Anil Kumar Verma
Gaurav Sharma
Kuldeep Singh

2016

maestro
A Wiley Brand

WILEY

ACING THE GATE

COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

FREE ONLINE TEST INSIDE

- Check your preparedness
- Take practice test(s)
- View analysis of your test performance

- Content by eminent authors with excellent credentials
- Systematic approach covering complete spectrum of the discipline at both macro (chapters) and micro (sections) levels
- Problem-solving skills built through solved and unsolved questions
  - covering all topics
  - organized in the order of difficulty
  - with complete solutions and detailed explanations
- Online resources available at **gate.wileymaestro.com**

WILEY ACING THE GATE

COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

maestro
A Wiley Brand

WILEY ACING THE GATE

COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

Anil Kumar Verma
Associate Professor
Department of Computer Science and Engineering
Thapar University, Patiala

Gaurav Sharma
Assistant Professor
School of Computing Science and Engineering
Galgotias University, Greater Noida

Kuldeep Singh
Department of Computer Science and Engineering
Thapar University, Patiala

WILEY

WILEY ACING THE GATE

COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

Copyright © 2015 by Wiley India Pvt. Ltd., 4435-36/7, Ansari Road, Daryaganj, New Delhi-110002.

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or scanning without the written permission of the publisher.

Limits of Liability: While the publisher and the author have used their best efforts in preparing this book, Wiley and the author make no representation or warranties with respect to the accuracy or completeness of the contents of this book, and specifically disclaim any implied warranties of merchantability or fitness for any particular purpose. There are no warranties which extend beyond the descriptions contained in this paragraph. No warranty may be created or extended by sales representatives or written sales materials.

Disclaimer: The contents of this book have been checked for accuracy. Since deviations cannot be precluded entirely, Wiley or its author cannot guarantee full agreement. As the book is intended for educational purpose, Wiley or its author shall not be responsible for any errors, omissions or damages arising out of the use of the information contained in the book. This publication is designed to provide accurate and authoritative information with regard to the subject matter covered. It is sold on the understanding that the Publisher is not engaged in rendering professional services.

Other Wiley Editorial Offices:

John Wiley & Sons, Inc. 111 River Street, Hoboken, NJ 07030, USA

Wiley-VCH Verlag GmbH, Pappellaee 3, D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 42 McDougall Street, Milton, Queensland 4064, Australia

John Wiley & Sons (Asia) Pte Ltd, 1 Fusionpolis Walk #07-01 Solaris, South Tower Singapore 138628

John Wiley & Sons Canada Ltd, 22 Worcester Road, Etobicoke, Ontario, Canada, M9W 1L1

First Edition: 2015

ISBN: 978-81-265-5086-9

ISBN: 978-81-265-8199-3 (ebk)

www.wileyindia.com

PREFACE

The objective of writing this book is to help the GATE aspirants in understanding the various subjects of Computer Science and Information Technology. Being a national level exam, it measures the candidate's ability on different parameters, namely, basic fundamentals, concepts, interpretation and analytical ability. This examination assumes more importance as the GATE score has been used by public sector undertakings (PSUs) for recruiting graduate-level engineers at the entry level, JRF at CSIR Labs, etc. Even certain foreign universities (like Singapore, Germany) have also started recognizing the GATE score, for using it as a benchmark for providing scholarships for students interested in pursuing PG and PhD programs.

The major strength of this book lies in the demystifying complex concepts by unpeeling levels of abstraction down to the basic level. The book follows a balanced approach on systematic representation of the text. Each chapter begins with the basics and then in-depth concepts are described with the help of various example(s)/numerical(s) and schematic diagrams. Different ways of interpretation are also highlighted while providing the solutions to previous year GATE question papers and then the chapter is culminated with a summary of important terms/formulas used in that chapter. Therefore, the book is self-contained.

The organization of this book reflects our belief that significant amount of practice is required for achieving the complete mastery. Therefore, each chapter contains a set of multiple questions of different levels (approximately 100 questions in each chapter) for self-assessment/mock test preparation, designed by keeping in mind the latest pattern.

While writing this book, we have drawn upon the long experience of teaching the subjects as well as our own understanding of the subjects. The contents have been carefully compiled and have been presented in a lucid manner to the reader. Utmost care has also been taken to relieve the candidate from the unnecessary burden of recollecting the concepts during the preparation, so that the student qualifies with a high GATE score.

We sincerely hope that students looking for an up-to-date, excellent self-study, compact and comprehensive book will be more than happy to have this book.

It is innate that some error(s) might have crept in, we shall be thankful to the reader for highlighting the same as well as the suggestions in the improvement of the text presented.

ACKNOWLEDGEMENTS

Writing a book, is like a major project and is rarely the work of a single individual.

We thank our teachers and seniors for providing the wealth of knowledge, sharing the depth and breadth of their experience and answering a barrage of our questions. Our numerous students, whose valuable discussions are a key constituent of this book.

During the compilation and writing of this work, we have been fortunate enough to receive useful suggestions, timely assistance and humble support from our colleagues and friends.

Special thanks to our family members who were extremely generous with their time being consumed for writing this book and tolerated us when we were trying tooth-and-nail to meet the deadlines.

We are grateful to the Wiley India, particularly, Siddhartha Deb, Publishing Editor, for initiating all of us into this venture and Sruthi Guru, Developmental Editor, for the continuous cooperation and assistance for the timely feedback. Without their conscientious effort and personal involvement, this piece of work would not have been possible. We would also like to thank the production team, Rakesh Poddar and Samir Das, for their efforts.

We wish to acknowledge the efforts of our reviewer(s) for providing countless suggestions for improving the content and presentation.

Thank you!

Anil Kumar Verma (akvtu.1001@gmail.com)

Gaurav Sharma (sharmagaurav86317@gmail.com)

Kuldeep Singh (kuldeepisp@gmail.com)

ABOUT THE AUTHORS

Anil Kumar Verma is currently working as an Associate Professor in the Department of CSE, Thapar University, Patiala. He is a member of MISCI (Turkey), LMCSI (Mumbai), GMAIMA (New Delhi) and certified software quality auditor by MoCIT, Govt. of India. He received his B.S., M.S. and PhD, majoring in Computer Science and Engineering. He has a rich teaching experience of over 23 years, which include the post of Lecturer at M.M.M. Engineering College (now, MMM University of Technology), Gorakhpur from 1991 onwards, apart from being a visiting faculty to many institutions. He has visited USA, South Korea and Japan for academic purposes, and published over 150 papers in refereed journals and conferences (India and abroad). He is member of various program committees for different international/national conferences and is on the review board of various international journals. His research interests include wireless networks, routing algorithms and securing Ad hoc networks. He can be reached at akvtu.1001@gmail.com.

Gaurav Sharma is currently working as an Assistant Professor in the School of Computing Science and Engineering at Galgotias University, Greater Noida. He received his M.E.(CSE) and PhD degrees from Thapar University, Patiala, and has published various papers in refereed journals and conferences. He is a member of IEEE and ACM, and of various program committees for different international/national conferences, besides being on the review board of various internationally renowned journals. His research interests includes routing and security issues in Ad hoc networks. He can be reached at sharmagaurav86317@gmail.com.

Kuldeep Singh is currently a PhD research scholar and freelance consultant at Thapar University, Patiala, working in the area of Ad hoc Network Security. He has done his M.E. (IS) from Thapar University. Prior to that he had done his MCA and qualified GATE and CSIR-UGC NET. He has commendable and growing list of publications, and is a member of reputed computer societies like ACM. He can be reached at kuldeepisp@gmail.com.

ACING THE GATE

ABOUT GATE

Purpose of Examination

The Graduate Aptitude Test in Engineering (GATE) essentially examines the aptitude of an engineering graduate. It differs from other examinations as it calls for an aptitude based learning and approach on engineering topics. Being a national level exam, GATE measures the candidate's ability on different parameters, namely, basic fundamentals, concepts, interpretation and analytical ability. This examination assumes more importance as the GATE score has been used by public sector undertakings (PSUs) for recruiting graduate-level engineers at the entry level, JRF at CSIR Labs, etc. Even certain foreign universities (like Singapore, Germany) have also started recognizing the GATE score, for using it as a benchmark for providing scholarships for students interested in pursuing PG and PhD programs.

1. **Financial assistance:** A GATE qualified student pursuing PG is liable to receive a stipend of ₹12,000/- pm for 24 months from MHRD, India and other Government agencies.
2. **Jobs in Public Sector Undertakings:** Major PSUs such as — Indian Oil Corporation Ltd. (IOC), Power Grid Corporation of India Ltd.(PGCIL), Oil and Natural Gas Corporation(ONGC), Punjab State Power Corporation Ltd. (PSPCL), Bharat Broadband Network Limited (BBNL), etc. are recruting candidates based on their GATE score.
3. **Foreign Universities:** National University, Singapore,; Nanyang University, Singapore; and some universities in Germany do consider the GATE score.

Applying for the Examination

1. **Eligibility:** The candidate fulfilling the following eligibility criteria can apply for the examination:
   Bachelor's degree in Engineering/Technology/Architecture/ Pharmacy (Post-Diploma/Post B.Sc./4 years after 10+2)

   OR

   Master's degree in any branch of Science/ Mathematics/ Statistics/ Computer Applications or equivalent
   Note: Students currently in the final year of the above courses can also apply.

2. **Application Procedure:** GATE Online Application Processing System (GOAPS) Website opens for enrolment, application filling, application submission starts in month of September. For details, refer to the website of the current organizing institute.
3. **Examination Schedule:** GATE is conducted in **online** (Computer Based Test (CBT), **mode only** and is spread over two weeks. It starts in the last week of January to first week of February. The GATE (CS/IT) paper is held in three sessions.
4. **Announcement of Results:** The results are announced on the GATE Online Application Processing System (GOAPS) Website managed by IITs/IISc in the month of March.

Structure of Examination

The question paper comprises 65 questions carrying 100 marks to be attempted in 3 hours. The paper is distributed over General Ability, Engineering Mathematics and Technical Subject.

1. **General Ability** accounts for 15% of the question paper containing 10 questions carrying 15 marks (5 questions of 1 mark each and 5 questions of 2 marks each).
2. **Engineering Mathematics** accounts for 15% of the question paper.
3. **Computer Science and Information Technology (CS & IT)** accounts for 70% of the question paper.

The emphasis is on the following type of questions:

1. **Recall:** Based on facts, principles or laws involved. So, the candidate has to answer them from his/her memory or simple computation.
2. **Comprehension:** Based on candidate's understanding of basic concepts that help him to draw conclusions.
3. **Application:** Based on the ability of the candidate to apply his/her knowledge through computation or logical reasoning.
4. **Analysis and Synthesis:** Based upon diagrams/data/tables/images that help in analysis. It also involves separating useful information from the irrelevant information.

The **types of questions** asked are **Multiple Choice Questions** in which one correct option out of the four given options needs to be chosen; and **Numerical Answer Questions** in which the correct answer needs to be keyed in using an online keyboard.

The marking scheme is as follows:

1. **Multiple Choice Questions:** These questions carry 1 or 2 marks. There is **negative marking** for every incorrect answer. For every wrong answer, 1/3 is deducted for a 1-mark question and 2/3 marks are deducted for a 2-mark question.
2. **Numerical Answer Questions:** These questions carry 1 or 2 marks. No choices are given for these questions. There is no negative marking for these questions.

Computerized Mode of Examination

From 2014 onwards, GATE is conducted mandatorily in online mode for all subject papers. Prior to this, it was conducted online for 4 papers in 2011 and 6 papers in 2012 and 2013.

In the Online Computer Based Test (CBT) mode, the candidate is required to select the correct option for MCQ type and enter the answer (numerical value) for numerical answer type question using a mouse on a virtual keyboard (keyboard of the computer will be disabled). Blank paper pads for rough work are provided for rough work, and these have to be returned back after the examination with roll numbers mentioned in them.

At the end of the 3-hour window, the computer will automatically close the screen from further actions. Another screen with submit button will appear which needs to be clicked to end the examination.

ATTRIBUTES FOR SUCCESS IN EXAMINATION

Preparing for the Examination

Knowing Your Subject (Knowledge of Material)

The exhaustive syllabi could be overwhelming as it covers the entire course studied over a period of 6−7 semesters. Thus, the emphasis must be on "knowing the material", and not just being familiar with the material.

1. A systematic organization of the topics can be understood by going through the examination syllabus of the subject and aligning it with the course studied.
2. Practicing questions from topics having more weightage to the ones having lower weightage helps understand the fundamentals of concepts and their application.
3. Familiarize with the use of proper syntactical expressions, assumptions, laws, theorems, standard symbols and notations for solving various problems based on programming codes and logic.

Preparation of a Study Plan

A well-chartered study plan taking into account the focus areas and learning objectives based on analysis of previous year papers is important to build a connect between knowing the subject and its quantitative applications. The most relevant topics for each unit and unit-wise weightage is provided in the part opener preceding each chapter opener. These can be helpful in deriving focus areas.

1. The study plan may include a schedule of chapter or sub-topic level study with equal focus on revising fundamentals and practicing sufficient questions based on each topic.
2. The focus areas can be evaluated based on individual assessment into strong and weak areas. The time devoted to revise fundamentals in weak areas should be compensated with the time spent in solving questions in strong focus areas.
3. The next step would be to spend sufficient time in practicing questions in weak areas as well as assess the progress made.
4. After having enough confidence on both strong and weak areas, mock tests covering all the topics can be attempted. Alternately, one-fourth of MCQs from each topic can be attempted.

Refining Problem-Solving Skills

The examination tests the ability to *analyze information* provided, *make judgments*, apply the *understanding* of topic to arrive at the best-suited answer.

1. A quick revision of all the important formulas and unit summary would be useful before attempting practice tests or mock tests.
2. Through the solved examples, the most effective way to solve a problem and the possible mistakes that led to the wrong options may be identified.
3. Attempt as many previous years' GATE questions based on the topic and follow the above approach for self-evaluation.
4. Practice more and more questions based on the problem areas and rectify mistakes committed in the earlier practice rounds.

Taking the Examination

Effective Preparation and Time Management

Towards the end of study plan, it is important to focus on exam preparedness and time management skills.

1. It is important to enhance problem-solving skills by continuously practicing questions. While doing so, it is important to give equal importance to both 1-mark and 2-mark type questions. Although the time required for solving 1-mark questions might be a little less than the 2-mark questions. It is important to practice both types to enhance the speed and proficiency to attempt all questions types.
2. The time taken to solved a complete paper replicating the examination paper, that is, a mock test or practice test can be evaluated, and the number of incorrectly answered questions can be identified.
3. The above exercise is helpful in determining the questions which took longer time to solve, and those for which preparation was lacking in some form. More and more of these questions can be practiced upon to increase efficiency and accuracy for solving.

Maximizing your Exam-Taking Efficiency

As the examination is conducted in online (computer based test) mode, it is helpful to familiarize oneself with the examination procedure and instructions by going through the screen layout, button functions, etc., by accessing the mock test available on GAOPS website.

1. It is important to take proper rest and sleep well in the night before the examination, to avoid having a cluttered memory on the day of the examination.
2. Read the instructions carefully within the allotted time, and raise any equipment/software concerns immediately to the invigilator.
3. Once the examination begins, be calm and adopt a positive attitude. Read the question carefully and then the options. Carefully note the negatives in the question, such as "Which of the statements in incorrect?", etc., and work accordingly.
4. Set different time limits for the different sections, giving the maximum time for the technical section.
5. If any question appears unfamiliar or it is difficult to recollect the topic on which the question is based upon, then mark it for review and move to the next question. Reserve some time to revisit such questions and try to narrow down the possible responses and make an educated guess.
6. Constantly monitor the progress against the time remaining, which is displayed on the top right corner of the screen.

BEST OF LUCK!!!

CONTENTS

Preface **v**

Acing the GATE **ix**

About GATE **ix**

Attributes for Success in Examination **xi**

1 Digital Electronics **3**

1.1 Introduction 3
1.2 Number System 3
1.2.1 Conversions of Number System 4
1.2.2 Complement of a Number 4
1.2.3 Representation of Negative Numbers 5
1.2.4 The IEEE Standard for Floating Point Numbers 5
1.3 Boolean Logic 6
1.3.1 Boolean Algebra Laws (Huntington's Postulates) 7
1.3.2 Duality Theorem 7
1.3.3 Consensus Theorem 7
1.3.4 Positive Logic and Negative Logic 8
1.4 Digital Circuits 11
1.4.1 Combinational Circuits 11
1.4.2 Sequential Circuits 16
1.5 Digital Logic Families 26

Important Formulas 28
Solved Examples 28
Gate Previous Years' Questions 33
Practice Exercises 46
Answers to Practice Exercises 51

2 Computer Organization and Architecture 57

2.1 Introduction 57
2.2 Computer Architecture 57
2.2.1 Register Set 57
2.2.2 Quantitative Principles to Design High-Performance Processor 58
2.3 Machine Instructions and Addressing Modes 58
2.3.1 Machine Instructions 58
2.3.2 Addressing Modes 59
2.4 Arithmetic Logic Unit 61
2.4.1 Arithmetic Micro-Operations 61
2.4.2 Logic Micro-Operations 61
2.5 CPU Control Design 62
2.5.1 Instruction Execution 63
2.5.2 CPU Data Path 64
2.5.3 Control Unit Design 64
2.5.4 RISC versus CISC Processors 65
2.6 I/O Interface (Interrupt and DMA Mode) 65
2.7 Instruction Pipelining 67
2.8 Memory Hierarchy 72
2.8.1 Main Memory 72
2.8.2 Secondary Memory 73
2.8.3 Cache Memory 73
2.8.4 Cache Mapping Techniques 74
Important Formulas 77
Solved Examples 77
GATE Previous Years' Questions 80
Practice Exercises 95
Answers to Practice Exercises 100

3 Programming and Data Structures 105

3.1 Introduction 105
3.2 Basic Terminology 105
3.2.1 Programming Languages 105
3.2.2 Classification of High-Level Languages 106
3.2.3 Procedural Programming and Object-Oriented Programming 106
3.2.4 Data Types 107
3.2.5 Control Flow Statements 108
3.2.6 Array 110
3.2.7 Function and Recursion 111
3.2.8 Pointers 113

3.2.9 Parameter Passing 115
3.2.10 Structures and Unions 117
3.2.11 Enumerated Data Types 117
3.2.12 Scoping 117
3.2.13 Binding 118
3.2.14 Abstract Data Types 119
3.3 Stack 119
3.3.1 PUSH Operation on Stack 119
3.3.2 POP Operation on Stack 119
3.3.3 Application of Stack 120
3.4 Queue 122
3.4.1 Basic Operations on Queue 122
3.4.2 Types of Queue 123
3.4.3 Applications of Queues 123
3.5 Linked List 123
3.5.1 Basic Operations 123
3.5.2 Linked List Implementation 124
3.6 Trees 125
3.6.1 Binary Tree 125
3.6.2 Types of Binary Trees 125
3.6.3 Array Representation of Binary Trees 126
3.6.4 Tree Applications 126
3.6.5 Binary Search Tree 126
3.6.6 Graphs 127
Important Formulas 137
Solved Examples 138
GATE Previous Years' Questions 142
Practice Exercises 165
Answers to Practice Exercises 172

4 Algorithms 177

4.1 Introduction 177
4.2 Algorithm 177
4.2.1 Properties of Algorithm 177
4.2.2 Steps to Solve a Problem 177
4.2.3 Algorithm Analysis 178
4.2.4 Asymptotic Notation 178
4.2.5 Recurrence Relations 180
4.3 Hashing 182
4.3.1 Hash Table 182
4.3.2 Hashing Functions 182
4.3.3 Collisions 182
4.4 Binary Heap 184
4.4.1 Insertion in Min-Heap Tree 185
4.4.2 Deletion in Min-Heap Tree 185
4.4.3 Time Complexity 186

4.5 Searching and Sorting 186
4.5.1 Linear Search 186
4.5.2 Binary Search 187
4.5.3 Bubble Sort 187
4.5.4 Selection Sort 187
4.5.5 Insertion Sort 188
4.5.6 Heap Sort 188
4.5.7 Merge Sort 190
4.5.8 Quick Sort 192
4.5.9 Randomized Quick Sort 193
4.5.10 Counting Sort 193
4.5.11 Comparison of Sorting Techniques 194
4.6 Graph 194
4.6.1 Types of Graph 194
4.6.2 Types of Simple Graph 194
4.6.3 Graph Representation 195
4.7 Greedy Approach 195
4.7.1 Applications of Greedy Approach 196
4.7.2 Fractional Knapsack 196
4.7.3 Huffman Coding 197
4.7.4 Minimum Spanning Tree 198
4.7.5 Single-Source Shortest Path 200
4.8 Graph Traversal 201
4.8.1 Breadth-First Traversal 201
4.8.2 Depth-First Traversal 202
4.9 Dynamic Programming 203
4.9.1 Applications of Dynamic Programming 204
4.9.2 Fibonacci Series 204
4.9.3 0/1 Knapsack 204
4.9.4 Longest Common Subsequence 205
4.10 All-Pair Shortest Path 205
4.10.1 Floyd's Algorithm 205
4.11 Concepts of Complexity Classes 205
4.11.1 P-Complexity Class 205
4.11.2 NP Complexity Class 206
4.11.3 NP-Complete 206
4.11.4 NP-Hard 206
Important Formulas 207
Solved Examples 207
GATE Previous Years' Questions 210
Practice Exercises 227
Answers to Practice Exercises 232

5 Theory of Computation 237

5.1 Introduction 237
5.2 Finite Automata 237
5.2.1 Finite Automaton Model Characteristics 238
5.2.2 Technical Terms 238
5.2.3 Grammar 239

5.2.4 Noam Chomsky Grammar Classification 239
5.2.5 Chomsky Hierarchy 240
5.2.6 Different Grammar Types 240
5.3 Finite Automata and Regular Language 241
5.3.1 Deterministic Finite Automata 241
5.3.2 Non-deterministic Finite Automata 241
5.3.3 Comparison of DFA and NFA 241
5.3.4 DFA and NFA Design 241
5.3.5 Applications of DFA/NFA 243
5.3.6 Regular Expression and Grammar 243
5.3.7 Pumping Lemma 244
5.3.8 Myhill–Nerode Theorem 245
5.3.9 Transducer or Finite State Machine 245
5.3.10 Conversion in Finite Automata 246
5.3.11 Properties of Regular Sets/Languages 249
5.3.12 Some Important DFA 250
5.4 Pushdown Automata 250
5.4.1 Model of PDA 250
5.4.2 Transition Function 251
5.4.3 Non-deterministic PDA 251
5.4.4 Deterministic PDA 251
5.4.5 Parsing 252
5.5 Context-Free Grammar and Languages 252
5.5.1 Context-Free Grammar 252
5.5.2 Standard Context-Free Language 252
5.5.3 Derivation Tree or Parse Tree 253
5.5.4 Ambiguous Grammar 254
5.5.5 Removal of Ambiguity 254
5.5.6 Context-Free Grammar Simplification 254
5.5.7 Chomsky Normal Form 256
5.5.8 Greibach Normal Form 256
5.5.9 Identification of Language 256
5.6 Turing Machine 256
5.6.1 Model of a Turing Machine 257
5.6.2 Designing a Turing Machine 258
5.6.3 Recursive and Recursive Enumerable Languages 258
5.6.4 Variation of Turing Machine 258
5.7 Closure and Decidability 258
Important Formulas 260
Solved Examples 260
GATE Previous Years' Questions 263
Practice Exercises 276
Answers to Practice Exercises 282

6 Compiler Design 287

6.1 Introduction 287
6.2 Compilers and Interpreters 287
6.2.1 Compiler 287

6.2.2 Interpreter 288
6.2.3 Phases of a Compiler 288
6.2.4 Grouping of Phases 290
6.3 Lexical Analyzer 290
6.3.1 Functions of a Lexical Analyzer 290
6.3.2 Implementation of a Lexical Analyzer 290
6.4 Parser 291
6.4.1 Context-Free Grammar 292
6.4.2 Derivation Tree or Parse Tree 292
6.4.3 Ambiguous Grammar 293
6.4.4 Top-Down Parser 293
6.4.5 Bottom-Up Parser 297
6.5 Syntax-Directed Translation 302
6.5.1 Attributes of Syntax-Directed Translation 302
6.5.2 Types of Syntax-Directed Translation 302
6.5.3 Applications of SDT 302
6.6 Runtime Environment 303
6.6.1 Storage Organization 303
6.6.2 Activation Record and Activation Trees 304
6.6.3 Procedure Call Return Model 304
6.6.4 Lexical Versus Dynamic Scoping 305
6.6.5 Symbol Table 305
6.7 Intermediate Code Generation 305
6.7.1 Intermediate Representations 305
6.8 Code Optimization 307
6.8.1 Types and Levels of Optimization 307
6.8.2 Primary Source of Optimization 307
Important Formulas 308
Solved Examples 309
GATE Previous Years' Questions 310
Practice Exercises 318
Answers to Practice Exercises 323

7 Operating System 327

7.1 Introduction 327
7.2 Types of Operating System 328
7.2.1 Batch Operating System 328
7.2.2 Multiprogramming Operating System 328
7.2.3 Multitasking Operating System 328
7.2.4 Multiprocessor Operating System 328
7.2.5 Real-Time Operating System 329
7.3 Process Management 329
7.3.1 Process 329
7.3.2 Schedulers 330
7.4 CPU Scheduling 331
7.4.1 Scheduling Algorithms 331

7.5 Process Synchronization 335
7.5.1 Types of Processes 335
7.5.2 Interprocess Communication 336
7.5.3 Classical Synchronization Problems 336
7.5.4 Race Condition 336
7.5.5 Critical Section 337
7.5.6 Peterson's Algorithm for Two Processes 337
7.5.7 Semaphores 338
7.5.8 Deadlock and Starvation 339
7.6 Deadlocks 339
7.6.1 Resource Types 340
7.6.2 Characteristics of a Deadlock 340
7.6.3 Resource Allocation Graph 340
7.6.4 Deadlock Prevention 341
7.6.5 Deadlock Avoidance 342
7.6.6 Deadlock Detection and Recovery 346
7.7 Threads 347
7.7.1 Benefits of Threads 347
7.7.2 Types of Threads 347
7.8 Memory Management 348
7.8.1 Partitioning 348
7.8.2 Partition Allocation Policies 348
7.8.3 Loading and Linking 349
7.8.4 Paging 350
7.8.5 Multi-Level Paging 352
7.8.6 Segmentation 354
7.8.7 Virtual Memory 355
7.8.8 Page Replacement Algorithms 355
7.9 File System 359
7.9.1 Directory Structures 359
7.9.2 Allocation Methods 359
7.10 I/O Systems 363
7.10.1 Disk Structure 363
7.10.2 Disk Scheduling Algorithms 366
7.11 Protection and Security 368
7.11.1 Security 368
7.11.2 Security Environment 369
7.11.3 Cryptography 369
7.11.4 Attacks from within the System 370
7.11.5 Attacks from Outside the System 370
7.11.6 Anti Virus Approaches 371
Important Formulas 371
Solved Examples 372
GATE Previous Years' Questions 374
Practice Exercises 391
Answers to Practice Exercises 398

8 Databases **403**

8.1 Introduction 403
8.1.1 Traditional File Processing Approach 403
8.1.2 Database Management System 404
8.2 Components of Database Systems 404
8.3 DBMS Architecture 405
8.3.1 3-Tier Architecture 405
8.3.2 Data Independence 405
8.4 Data Models 405
8.4.1 Relational Model 406
8.4.2 ER Model 407
8.5 Database Design 410
8.5.1 Integrity Constraints 410
8.5.2 Normal Forms 411
8.5.3 Attribute Closure 411
8.5.4 Key 411
8.5.5 Decomposition 412
8.6 Query Languages (SQL) 414
8.6.1 SQL Commands 414
8.7 File Structures 416
8.7.1 Sequential Files 416
8.7.2 Indexing 416
8.7.3 B Tree 416
8.7.4 B+ Trees 416
8.8 Transactions and Concurrency Control 417
8.8.1 Transactions 417
8.8.2 Schedule 417
8.8.3 Classification of Schedule Based on Recoverability 419
8.8.4 Classification of Schedule Based on Serializability 420
8.8.5 Concurrency Control Protocol 423
Important Formulas 424
Solved Examples 425
GATE Previous Years' Questions 427
Practice Exercises 444
Answers to Practice Exercises 449

9 Information Systems and Software Engineering **453**

9.1 Introduction 453
9.2 Information Systems 453
9.2.1 Components of Information Systems 454
9.2.2 Types of Information Systems 454
9.3 Software 456
9.3.1 Characteristics of Software 456
9.3.2 Software Engineering 457
9.4 Process Models 458
9.4.1 Conventional Process Model 458
9.4.2 Evolutionary Process Model 459

9.5 Measurement of Metrics 461
9.5.1 Metrics 461
9.5.2 Size-Oriented Metrics 462
9.5.3 Effort and Schedule (Duration) Estimation 463
9.6 Risk Analysis 466
9.6.1 Risk Identification 466
9.6.2 Risk Projection or Risk Estimation 466
9.7 Software Development Life Cycle 467
9.7.1 Requirement 467
9.7.2 Design 470
9.7.3 Coding 472
9.7.4 Testing 472
Important Formulas 477
Solved Examples 477
GATE Previous Years' Questions 479
Practice Exercises 483
Answers to Practice Exercises 487

10 Computer Networks 491

10.1 Introduction 491
10.2 Network 491
10.2.1 Network Topology 492
10.3 LAN Technologies 492
10.3.1 Ethernet 492
10.3.2 Token Bus 493
10.3.3 Token Ring 493
10.4 ISO/OSI Stack 494
10.4.1 ISO/OSI Model 494
10.5 Routing Algorithms 498
10.6 Network Layer Protocols 501
10.6.1 Internet Protocol (IPv4) 501
10.6.2 ICMP 505
10.7 Layer 4: Transport Layer 505
10.8 Congestion 505
10.8.1 Congestion Versus Flow Control 506
10.9 User Datagram Protocol and Transmission Control Protocol 506
10.9.1 User Datagram Protocol 506
10.9.2 Transmission Control Protocol 506
10.10 Sockets 507
10.11 Layer 5: Session Layer 507
10.12 Layer 6: Presentation Layer 508
10.13 Layer 7: Application Layer 508
10.14 Devices 509
10.15 Network Security 510
10.15.1 Basic Concepts in Cryptography 510
10.15.2 Digital Signature 511
10.15.3 Firewall 511

Important Formulas 512
Solved Examples 513
GATE Previous Years' Questions 515
Practice Exercises 527
Answers to Practice Exercises 532

11 Web Technologies 537

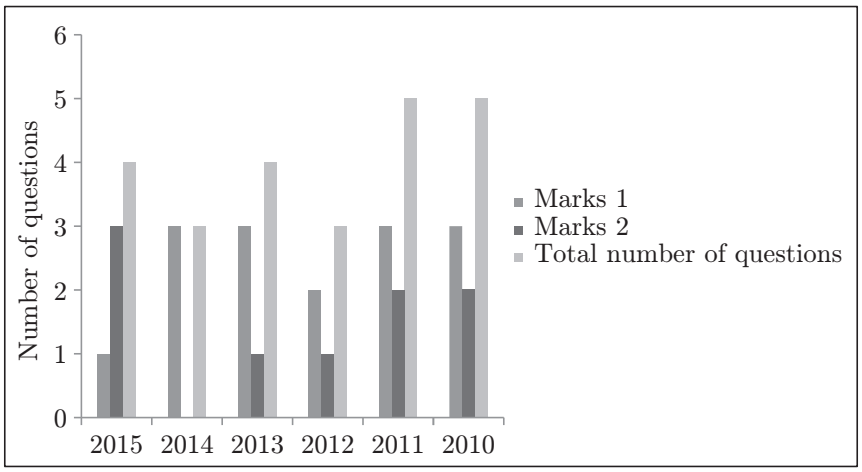
11.1 Introduction 537
11.2 HTML 538
11.2.1 HTML Tags 538
11.2.2 HTML Attributes 538
11.2.3 HTML Elements 539
11.2.4 HTML Character Entities 539
11.2.5 HTML Formatting 539
11.2.6 HTML Phrase Tags 540
11.2.7 HTML Background 540
11.2.8 HTML Lists 540
11.2.9 HTML Links 540
11.2.10 HTML Images 540
11.2.11 HTML Tables 540
11.2.12 HTML Frames 541
11.2.13 HTML Forms 541
11.2.14 HTML Marquees 541
11.3 Cascading Style Sheets 541
11.4 XML 541
11.4.1 Advantages of XML 542
11.4.2 Document-Type Definition 542
11.4.3 Tag Rules for XML Documents 542
11.4.4 Types of XML Documents 543
11.4.5 XHTML 543
11.4.6 Document Object Model (DOM) 543
11.4.7 XUL 543
11.4.8 Flash and Silverlight 543
11.4.9 User-Interface Language 543
11.5 Basic Concepts of Client−Server Computing 544
11.5.1 Server Types 544
11.5.2 Stateless and Stateful Servers 544
11.5.3 Thin Client and Fat Servers 544
11.5.4 Functions of a Client 544
11.5.5 Functions of a Server 544
11.5.6 Topologies for Client−Server 545
11.5.7 Types of Client−Server Model 545
11.5.8 Merits and Demerits of Client−Server 546
11.6 J2EE platform 546
11.6.1 J2EE Services 546
11.6.2 J2EE Architecture 547
11.6.3 EJB Container 547
11.6.4 J2EE Application Server 547

Important Formulas 547
Solved Examples 548
GATE Previous Years' Questions 550
Practice Exercises 551
Answers to Practice Exercises 554

Solved GATE (CS) 2014 Set 1 **555**
Solved GATE (CS) 2014 Set 2 **567**
Solved GATE (CS) 2014 Set 3 **579**
Solved GATE (CS) 2015 Set 1 **591**
Solved GATE (CS) 2015 Set 2 **605**
Solved GATE (CS) 2015 Set 3 **617**
Index **631**

UNIT I: DIGITAL ELECTRONICS

LAST SIX YEARS' GATE ANALYSIS



Concepts on which questions were asked in the previous six years

Year	Concepts
2015	K-maps, Half and full adder, JK flip-flop, Boolean operators, Prime implicates, Base values, Johnsen counter, Binary operator, D flip-flop
2014	Minterm expressions, Combinational functional blocks, Boolean expressions, flip−flops
2013	2's complement, Truth tables, Logic gates, RAM chips
2012	K-maps, Truth tables, IEEE single precision
2011	Flip−flops, Boolean expressions, Logic gates
2010	K-maps, 2's complement, Multiplexer, Boolean expression, Sequential circuits

CHAPTER 1

DIGITAL ELECTRONICS

Syllabus: Digital electronics: logic functions, minimization, design and synthesis of combinational and sequential circuits, number representation and computer arithmetic (fixed and floating point).

1.1 INTRODUCTION

Digital electronics represents discrete signals instead of signals in a continuous range. It uses two binary levels 0's (corresponding to false) and 1's (corresponding to true). The main reason for advancement in digital electronics is integrated circuits (ICs).

1.2 NUMBER SYSTEM

Number system is an age old method to represent numerals (Fig. 1.1). Decimal number system is the most common number system and binary number system is used by computers.

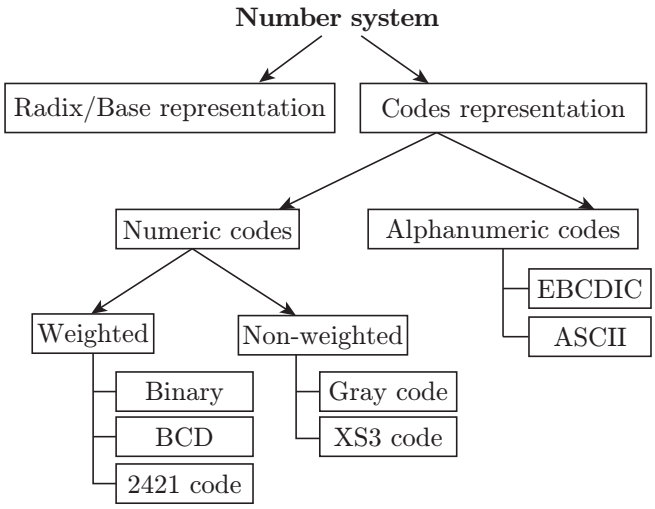


Figure 1.1 Number system.

Table 1.1 | Types of number systems

Base/Radix	Unique Numbers	Terminology	Example
10	0 to 9	Decimal number system	10.47
8	0 to 7	Octal number system	65.32
2	0 to 1	Binary number system	110.11
16	0 to 15 0 to 9 are numerals and 10 to 15 are represented with alphabets 10 = *A*, 11 = *B*, 12 = *C*, … 15 = *F*	Hexadecimal number system	BAD.1A

Radix or base is the number of unique digits, so the different base or radix can be expressed as represented in Table 1.1. For example, $(57)_{10}$ – here 10 is the base (or radix) and 57 is the number.

Example 1.1

Let us understand the logic of representation by considering $(57.1)_{10}$. We know 57 means 50 + 7 + (1/10). Now, representing the same by making use of radix (which is 10), we get

$$5\times(10)^1+7\times(10)^0+1\times(10)^{-1}$$

1.2.1 Conversions of Number System

A number of a particular number system can be converted into another number system, as follows:

1. **Decimal number system to any number system:** Consider the example of $(57.3)_{10}$. To find the binary equivalent, we have

2|57			
2|28	1		
2|14	0	$2\times0.3=0.6$	0
2|7	0	$2\times0.6=1.2$	1
2|3	1	$2\times0.2=0.4$	0
2|1	1	$2\times0.4=0.8$	0
$(111001)_2$		$(0.0100)_2$	

Therefore, $(57.3)_{10}=(111001.0100)_2$

2. **Any number system to decimal number system:** Consider the example of $(111001.0100)_2$. To find the decimal equivalent we have

$$\begin{aligned}(111001.0100)_2 &= 1\times2^5+1\times2^4+1\times2^3+0\times2^2\\ &\quad+0\times2^1+1\times2^0+0\times2^{-1}\\ &\quad+1\times2^{-2}+0\times2^{-3}+0\times2^{-4}\\ &=32+16+8+0+0+1+0\\ &\quad+0.25+0+0=(57.25)_{10}\end{aligned}$$

Another example is of $(101.11)_2=(5.75)_{10}=(5.6)_8=(5.C)_{16}$.

1.2.2 Complement of a Number

Any given number will have two complements:

1. **Radix minus one complement (R-1's complement):** It is obtained by subtracting the given number from the highest possible number. For example, if there is a two-digit decimal number, it will be subtracted from 99; whereas a 3-bit number in a binary system will be subtracted from 111.

   Note: The 1's complement of a binary number is obtained by interchanging the 1's and 0's. For example, 1's complement of $(1011)_2$ is $(0100)_2$.

2. **Radix complement (R's complement):** This is also known as true complement. It is obtained by adding one to R-1's complement. For example, $(57)_{10}$ – its 9's complement is 42 and its 10's complement is 43.

1.2.2.1 Subtraction Using R-1's Complement

For the following subtractions,

$$\begin{array}{r} A \\ -B \\ \hline \\ \hline \end{array}$$

the steps involved are as follows:

Step 1. Find R-1's complement of *B*.

Step 2. Add *A* and *B*.

Step 3. If there is carry, then add this carry to the answer.

If there is no carry, then find the R-1's complement of the answer and place a negative sign in front of the answer.

Problem 1.1: $(101)_2 - (011)_2$

Solution:

Step 1. 1's Complement of 011 is 100.

Step 2. The addition of these two numbers is (101 + 100 = 1 001).

Step 3. There is a carry of 1, so now we will add this carry to the answer (1 + 001), which is $(010)_2$.

Problem 1.2: $(011)_2 - (101)_2$

Solution:

Step 1. 1's Complement of 101 is 010.

Step 2. The addition of these two numbers is (011 + 010 = 0 101).

Step 3. There is a no carry, so now we will find 1's complement of the answer obtained (which is 010) and place a negative sign in front of it. So, the answer is $(-010)_2$.

1.2.2.2 Subtraction Using R's Complement

In this method, we find R's complement of the subtrahend and we do not add the carry, as in step 2. Rest of the steps are the same.

Problem 1.3: $(101)_2 - (011)_2$

Solution:

Step 1. 2's complement of 011 is 101.

Step 2. The addition of these two numbers is (101 + 101 = 1 010).

Step 3. There is a carry of 1, so now we will not add this carry to the answer (001), which is $(010)_2$.

Problem 1.4: $(011)_2 - (101)_2$

Solution:

Step 1. 2's complement of 101 is 011.

Step 2. The addition of these two numbers is (011 + 011 = 0 110).

Step 3. There is a no carry, so now we will find 2's complement of the answer obtained (which is 010) and place a negative sign in front of it. So, the answer is $(-010)_2$.

1.2.3 Representation of Negative Numbers

In$-(57)_{10}$ = (1111001), the most significant bit is 1, showing that the number is negative binary number.

1.2.4 The IEEE Standard for Floating Point Numbers

The IEEE (Institute of Electrical and Electronics Engineers) has provided a standard for floating point numbers and their arithmetic. This standard specifies how single precision (32-bit) and double precision (64-bit) floating point numbers are to be represented.

1. **Single Precision:** The IEEE single precision floating point standard representation requires a 32-bit word. The first bit to the left is the sign bit, the next eight bits are the exponent bits, and the final 23 bits form the fraction part (Fig. 1.2).

   The value of a normalized number $= (-1)^s \times 1.m \times 2^{e^{-127}}$

   where $s = 1$ for negative number, $s = 0$ for positive number, m = mantissa and e = exponent.

2. **Double Precision:** The IEEE double precision floating point standard representation requires a 64-bit word. The first bit is the sign bit, the next eleven bits are the exponent bits, and the final 52 bits form the fraction part (Fig. 1.3).

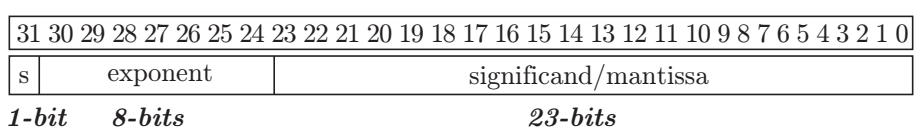


Figure 1.2 | Single Precision (32 bits).

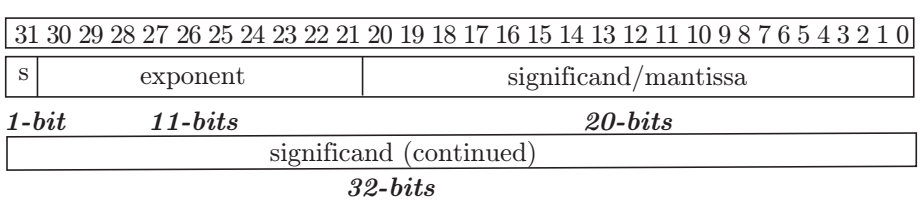


Figure 1.3 | Double Precision (64 bits).

Problem 1.5: Consider the decimal number: +116.25. Represent it using single precision IEEE floating point standard.

Solution: The equivalent binary representation is +1110100.01

$$\begin{aligned}\text{Normalizing number} &= +1.11010001 \times 2^{6}\\ &= +1.11010001 \times 2^{133-127}\\ &= +1.11010001 \times 2^{10000101-01111111}\end{aligned}$$

In IEEE 754 format:

Sign	Exponent	Fraction
0	10000101	11010001000000000000000

1.3 BOOLEAN LOGIC

The concept of Boolean logic was proposed by George Boole, in which any information or data is represented in the form of two states, which are complement to each other (such as 0 and 1, hot and cold, black and white, north and south, etc.).

Logic gates form the basic building block of the digital system. A logic gate is an electronic device that takes logical decisions based on given input combinations and its output can be represented by Boolean expression. The symbols used to represent logic gates are shown in Fig. 1.4.

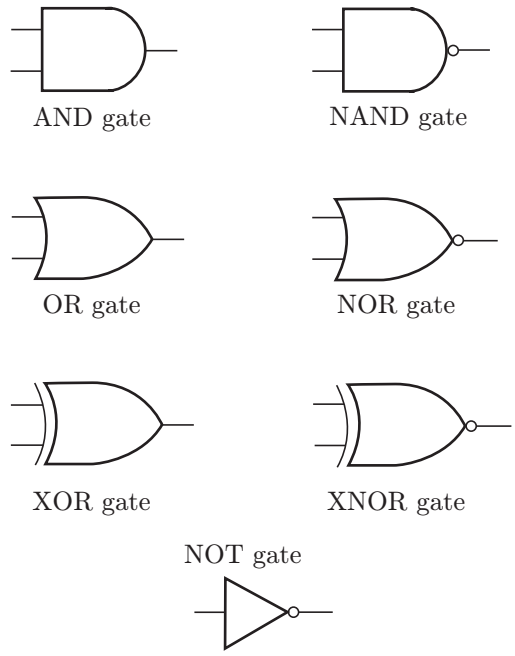


Figure 1.4 | Logic gates symbols.

Table 1.2 represents 15 basic operations that can be performed.

Table 1.2 | Basic operations in Boolean algebra

Inputs		Operations				
X	***Y***	**Null**	**Identity**	**Inhibit**		**Transfer**
				X* Inhibit *Y	***Y* Inhibit *X***	
0	0	0	1	0	0	0
0	1	0	1	0	1	1
1	0	0	1	1	0	0
1	1	0	1	0	0	1

Inputs		Operations					
X	***Y***	**AND**	**OR**	**NOT**		**XOR**	**XNOR**
				NOT *X*	**NOT *Y***		
0	0	0	0	1	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	0	1	1	0
1	1	1	1	0	0	0	1

(*Continued*)

Table 1.2 | Continued

Inputs		Operations			
X	Y	NAND	NOR	Implication	
				X Implication Y	Y Implication X
0	0	1	1	1	1
0	1	1	0	1	0
1	0	1	0	0	1
1	1	0	0	1	1

Table 1.2 is interpreted as follows:

1. Null – All the outputs are zero.
2. Identity – All the outputs are one.
3. X inhibit Y – Represented as $(x \cdot \sim y)$
4. Y inhibit X – Represented as $(y \cdot \sim x)$
5. Transfer – Same as y
6. AND – $x \cdot y$ (Similar to intersection in set theory)
7. OR – $x + y$ (Similar to union in set theory)
8. NOT X – $\sim x$ (Also known as complement of x)
9. NOT Y – $\sim y$ (Also known as complement of y)
10. XOR – $(\sim x \cdot y) + (x \cdot \sim y)$ (Also known as parity checker)
11. XNOR – $(x \cdot y) + (\sim x \cdot \sim y)$
12. NAND – $\sim(x \cdot y)$
13. NOR – $\sim(x + y)$
14. X implication Y $(\sim x + y)$
15. Y implication X $(\sim y + x)$

Problem 1.6: It is defined that $x * y = x'y' + xy$ and $z = x * y$. Among the following P, Q and R, which statement is correct?

P: $x = y * z$

Q: $y = x * z$

R: $x * y * z = 1$

Solution:

x	Y	z	P	Q	R
0	0	1	0	0	1
0	1	0	0	1	1
1	0	0	1	0	1
1	1	1	1	1	1

1.3.1 Boolean Algebra Laws (Huntington's Postulates)

The laws of Boolean Algebra are given by Huntington's postulates:

1. $X + 0 = X$
2. $X + (\sim X) = 1$
3. $X + X = X$
4. $X + 1 = 1$
5. $X + Y = Y + X$
6. $X + (Y + Z) = (X + Y) + Z$
7. $X \cdot (Y + Z) = X \cdot Y + X \cdot Z$
8. $\sim(X + Y) = \sim X \cdot \sim Y$ (Also called DeMorgan's law)
9. $X + (X \cdot Y) = X$
10. $\sim(\sim X) = X$

1.3.2 Duality Theorem

Every expression remains valid if we interchange the operator and identity elements. Following laws are the dual of the respective laws written above.

1. $X \cdot 1 = X$
2. $(X \cdot \sim X) = X \cdot X' = 0$
3. $X \cdot X = X$
4. $X \cdot 0 = 0$
5. $X \cdot Y = Y \cdot X$
6. $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$
7. $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$
8. $(X \cdot Y)' = X' + Y'$
9. $X \cdot (X + Y) = X$

1.3.3 Consensus Theorem

It is applicable only if a Boolean function has

1. three variables
2. each variable is used two times
3. only one variable in complemented or non-complemented form

Then, the term related to the complemented and non-complemented variable is the answer.

Example 1.2

$$AB + A'C + BC = AB + A'C$$

1.3.4 Positive Logic and Negative Logic

Usually, the two (binary) levels are represented in two different ways:

1 (signifying high) and 0 (signifying low) – Known as positive logic

or

1 (signifying low) and 0 (signifying high) – Known as negative logic

1.3.4.1 Canonical and Standard Forms

The minterms and the maxterms are given in Table 1.3.

Product-of-sums (POS) form:

$$\begin{aligned} F &= (x'y'z') + (xy'z') + (xyz) \\ &= m_0 + m_4 + m_7 \\ &= \sum m(0,4,7) \end{aligned}$$

Sum-of-products (SOP) form:

$$\begin{aligned} F &= (X+Y+Z)\cdot(X'+Y+Z)\cdot(X+Y+Z) \\ &= M_0 \cdot M_4 \cdot M_7 \\ &= \prod M(0,4,7) \end{aligned}$$

In order to convert minterm to maxterm, put all those numbers in maxterm which are not present in minterm, and vice versa.

$$\sum m(0,1,5) = \prod M(2,3,4,6,7)$$

1.3.4.2 Minimization of Expression

A Boolean expression which cannot be further reduced is said to be in minimal form. Each term in minimal Boolean expression is called prime implicant. The following are the two methods for minimization:

1. **Boolean Theorems (postulates):** Using the Boolean postulates, we can simplify an expression using algebraic method. This method is relatively difficult as we need to remember all the postulates.

Problem 1.7: Simplify the Boolean function $X = (A+B)(A+C)$.

Solution:

$$\begin{aligned} X &= AA + AC + AB + BC \\ &= A + AC + AB + BC \\ &= A(1+C) + AB + BC \\ &= A(1) + AB + BC \\ &= A + AB + BC \\ &= A(1+B) + BC \\ &= A(1) + BC \\ &= A + BC \end{aligned}$$

Problem 1.8: Simplify the Boolean function $X = A + A'B$.

Solution:

$$\begin{aligned} X &= A(1+B) + A'B \\ &= A\cdot B + A + A'B = A\cdot B + A'B + A \\ &= (A+A')\cdot B + A = 1\cdot B + A = B + A \end{aligned}$$

Table 1.3 | Minterms and Maxterms

		Minterms (SOP)		Maxterms (POS)	
Decimal	**Binary**	**Term**	**Designation**	**Term**	**Designation**
0	000	$x'y'z'$	m_0	$X + Y + Z$	M_0
1	001	$x'y'z$	m_1	$X + Y + Z'$	M_1
2	010	$x'yz'$	m_2	$X + Y' + Z$	M_2
3	011	$x'yz$	m_3	$X + Y' + Z'$	M_3
4	100	$xy'z'$	m_4	$X' + Y + Z$	M_4
5	101	$xy'z$	m_5	$X' + Y + Z'$	M_5
6	110	xyz'	m_6	$X' + Y' + Z$	M_6
7	111	xyz	m_7	$X' + Y' + Z'$	M_7

2. **Karnaugh-Map Method (K-Map method):** Karnaugh-map (K-map) involves graphical representation for simplifying Boolean expressions. It consists of group of adjacent cells in which each cell corresponds to one of the combinations of maxterms or minterms of n variables and is represented by group of literals. For n-variable K-map, there are 2^n cells required (Figs. 1.5–1.7). From one cell to the neighbouring cell there must be only one literal change. While solving the K-map from higher-order group to lower-order group only, the simplification must be done. In K-map, leftmost column and rightmost column of the same row are considered to be adjacent.
   If $K = 1$, it represents that all the cells are 1 so there is no need of circuit.

1.3.4.3 Grouping of Cells for Simplification

The grouping of cells for simplification is done in the following ways:

1. **Pairs:** If two adjacent cells either in row or in column contain 1's, they are said to be in pairs. Grouping of two adjacent cells removes one variable and output will contain common variables in two terms.
2. **Quads:** We can group four adjacent cells in K-map forming a quad. Grouping a quad eliminates the two variables that appear in both normal and complemented form. Resultant terms contain $(n - 2)$ variables, where n is the number of variables.
3. **Octet:** A group of eight adjacent cells is called octet. Grouping of 1's eliminates three variables. Resultant terms contain $(n - 3)$ variables, where n is the number of variables.

1.3.4.4 SOP Simplification Using K-Map

A higher priority will be given to more possible covering, that is, octet then to quad and then to pairs. The following procedure is followed:

1. Check the SOP expression and convert to standard and canonical form if required.
2. Draw the K-map based on the number of variables.
3. On the basis of SOP canonical form, 1's are entered to corresponding cells. Group the adjacent cells and take the common variable from each grouping.
4. Combine each group variable to obtain simplified expression.

1.3.4.5 POS Simplification Using K-Map

1. Check the POS expression and convert it to standard and canonical form if required.
2. Draw the K-map based on the number of variables.
3. Enter 0's in corresponding cells and group the adjacent cells.
4. Take the common variable from each grouping and write terms in OR form. The uncomplemented variable is assigned 0 and complemented is assigned 1.
5. Combine each OR term from the different groups to obtain simplified expression.

A \ B	0	1
0	00	01
1	10	11

Two-variable K-map

A \ B	0	1
0	0	1
1	2	3

Cell designation

A \ B	B′	B
A′	A′B′	A′B
A	AB′	AB

Cells with minterm

A \ B	B	B′
A	A + B	A + B′
A′	A′ + B	A′ + B′

Cells with maxterm

Figure 1.5 | Two-variable K-map.

A \ BC	00	01	11	10
0	000	001	011	010
1	100	101	111	110

Three-variable K-map

A \ BC	00	01	11	10
0	0	1	3	2
1	4	5	7	6

Cell designation

A \ BC	B′C′	B′C	BC	BC′
A′	A′B′C′	A′B′C	A′BC	A′BC′
A	AB′C′	AB′C	ABC	ABC′

Cells with minterm

A \ BC	B + C	B + C′	B′ + C′	B′ + C
A	A + B + C	A + B + C′	A + B′ + C′	A + B′ + C
A	A′ + B + C	A′ + B + C′	A′ + B′ + C′	A′ + B′ + C

Cells with maxterm

Figure 1.6 | Three-variable K-map.

AB \ CD	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

Four-variable K-map

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

Cell designation

AB \ CD	$C'D'$	$C'D$	CD	CD'
$A'B'$	$A'B'C'D'$	$A'B'C'D$	$A'B'CD$	$A'B'CD'$
$A'B$	$A'BC'D'$	$A'BC'D$	$A'BCD$	$A'BCD'$
AB	$ABC'D'$	$ABC'D$	$ABCD$	$ABCD'$
AB'	$AB'C'D'$	$AB'C'D$	$AB'CD$	$AB'CD'$

Cells with minterm

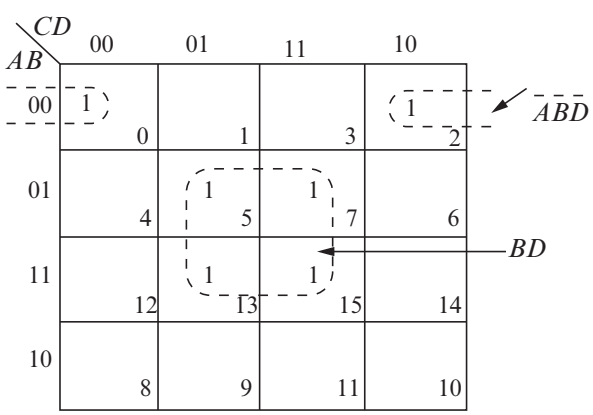
AB \ CD	$C+D$	$C+D'$	$C'+D'$	$C'+D$
$A+B$	$A+B+C+D$	$A+B+C+D'$	$A+B+C'+D'$	$A+B+C'+D$
$A+B'$	$A+B'+C+D$	$A+B'+C+D'$	$A+B'+C'+D'$	$A+B'+C'+D$
$A'+B'$	$A'+B'+C+D$	$A'+B'+C+D'$	$A'+B'+C'+D'$	$A'+B'+C'+D$
$A'+B$	$A'+B+C+D$	$A'+B+C+D$	$A'+B+C'+D'$	$A'+B+C'+D$

Cells with maxterm

Figure 1.7 Four-variable K-map.

Problem 1.9: Simplify $F(A,B,C,D) = \sum m(0, 2, 5, 7, 13, 15)$.

Solution:



The simplified Boolean expression is $F = A'B'D' + BD$.

Problem 1.10: Simplify $F(A,B,C,D) = ACD + A'B + D'$

Solution: Converting to standard SOP form. The first term ACD has one missing variable B.

$$ACD = ACD(B + B') = ABCD + AB'CD$$

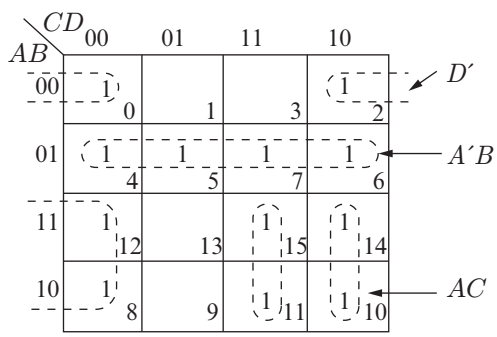
Similarly,

$$A'B = A'B(C + C')(D + D')$$
$$= A'BCD + A'BCD' + A'BC'D + A'BC'D'$$

$$D' = D'(B + B')(C + C') + (A + A')$$

Combining all terms, we get

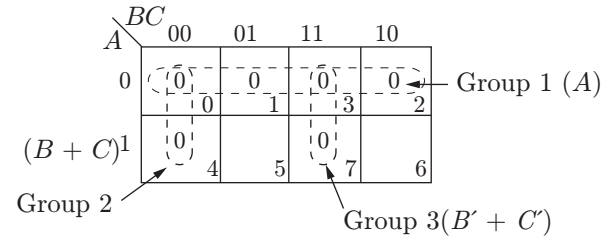
$$F = ABCD + AB'CD + A'BCD + A'BCD'$$
$$+ A'BC'D + A'BC'D' + ABC'D'$$
$$+ AB'CD' + AB'C'D' + A'B'CD' + A'B'C'D'$$



The simplified Boolean expression is $F = D' + A'B + AC$.

Problem 1.11: Simplify $S = \Pi M(0,1,2,3,4,7)$

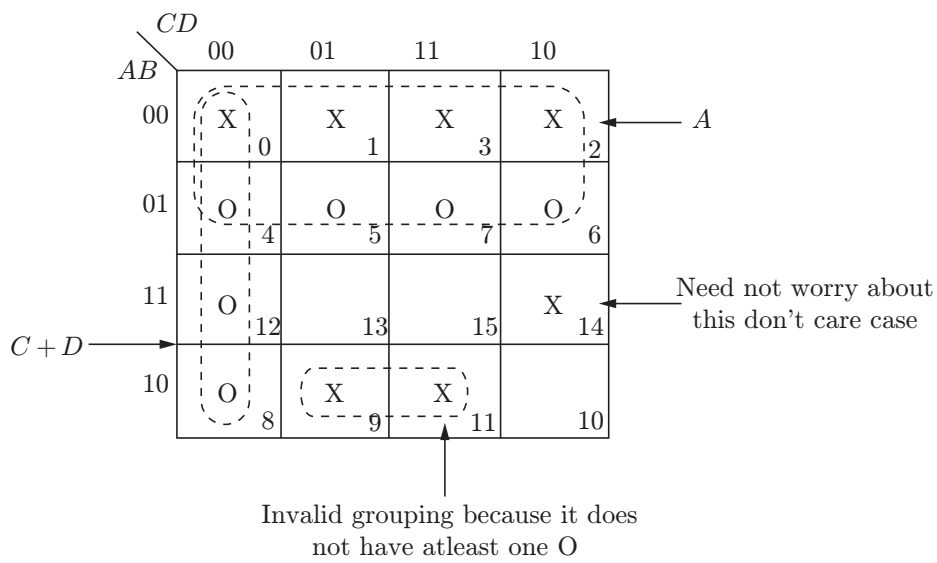
Solution:



The simplified Boolean expression is $S = A \cdot (B + C) \cdot (B' + C')$

Problem 1.12: Simplify $F(A, B, C, D) = \Pi M(4, 5, 6, 7, 8, 12) \cdot d(0, 1, 2, 3, 9, 11, 14)$

Solution: A logic circuit output may be either 0 or 1 but some circuit has output that will never occur for certain combinations of inputs. These conditions are don't care conditions and represented by X in K-map.



The simplified Boolean expression is $F = A(C + D)$.

1.4 DIGITAL CIRCUITS

The classification of digital circuitsis shown in Fig. 1.8.

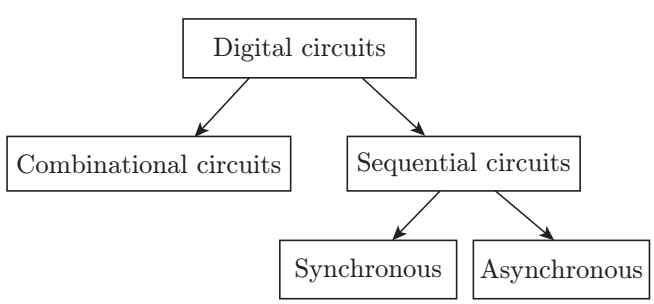


Figure 1.8 Types of digital circuits.

1.4.1 Combinational Circuits

A logic circuit whose output at any instant of time depends only on the present input is called combinational circuit. It contains no memory element.

1.4.1.1 Half Adder

A half adder is a combinational circuit for the addition of two 1-bit numbers. X and Y are inputs and outputs are sum (S) and carry (C) (Fig. 1.9 and Table 1.4).

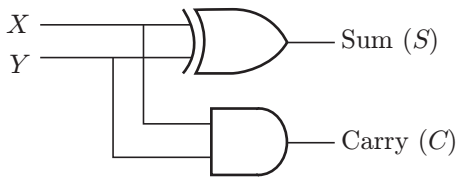


Figure 1.9 Logic diagram of a half adder.

Table 1.4 Truth table

Inputs		Outputs	
X	***Y***	**Carry (*C*)**	**Sum (*S*)**
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The sum and carry are as follows:

$$S = X'Y + XY'$$
$$C = XY$$

1.4.1.2 Full Adder

A full adder is a combinational circuit that performs the addition of 3 bits (2 significant bits and previous carry). It consists of three inputs and two outputs, two input bits that are to be added, the third input is the carry from the previous position (Fig. 1.10 and Table 1.5).

Table 1.5 Truth table

Inputs			Outputs	
X	***Y***	C_{in}	**Sum (*S*)**	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

The sum and carry are as follows:

$$S = X'Y'C_{in} + X'YC'_{in} + XYC_{in}$$
$$C_{out} = AC_{in} + XY + YC_{in}$$

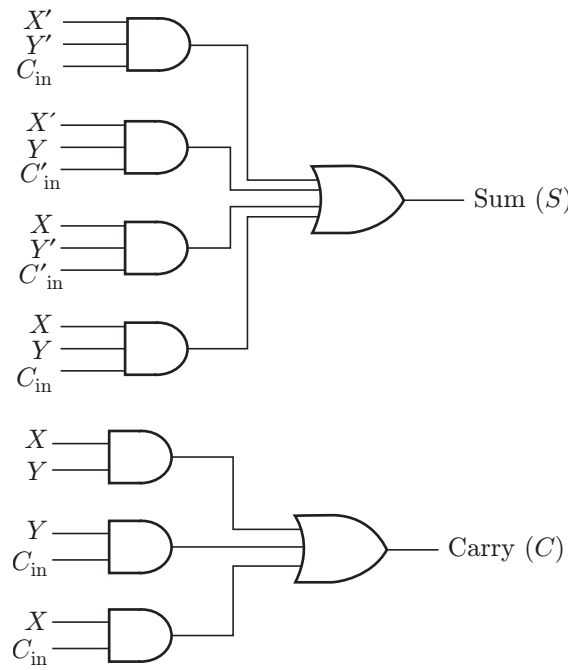


Figure 1.10 Logic diagram of a full adder.

1.4.1.3 Half Subtractor

A half subtractor is a combinational circuit that subtracts 2 bits (minuend and subtrahend) and produces their differences and borrow as output (Fig. 1.11 and Table 1.6).

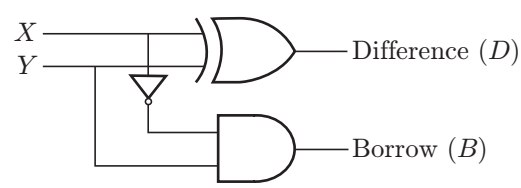


Figure 1.11 Logic diagram of a half subtractor.

Table 1.6 Truth table

Inputs		Outputs	
X	***Y***	**Difference (*D*)**	**Borrow (*B*)**
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

The difference and borrow are as follows:

$$D = XY' + X'Y$$
$$B = XY$$

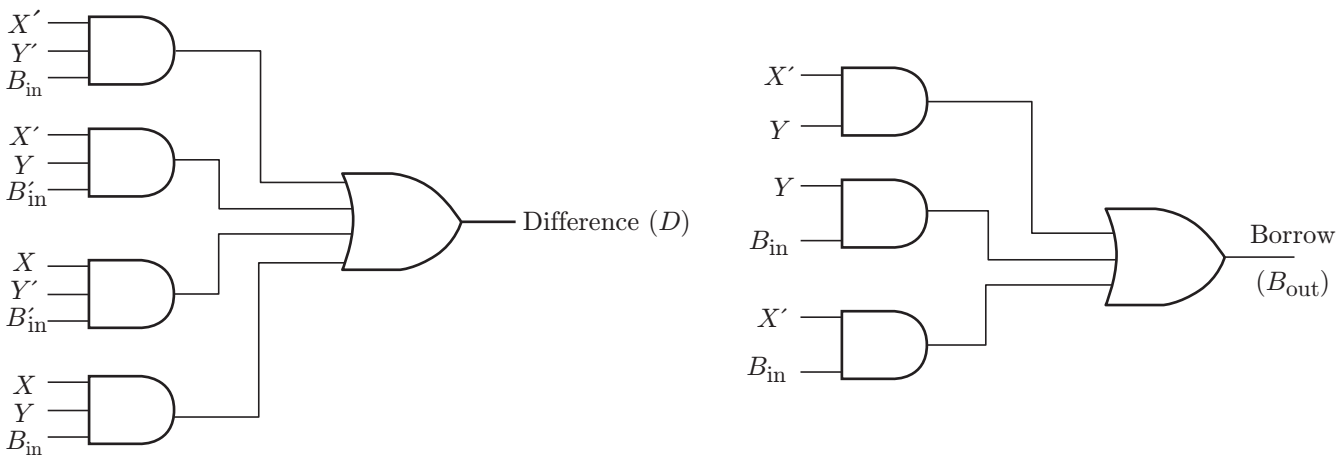


Figure 1.12 | Logic diagram of a full subtractor.

1.4.1.4 Full Subtractor

A full subtractor is a combinational circuit that performs subtraction involving 3 bits (minuend bit, subtrahend bit and borrow from previous stage) (Table 1.7 and Fig. 1.12).

Table 1.7 | Truth table

Inputs			Outputs	
X	Y	B_{in}	Difference (D)	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

The difference and borrow are as follows:

$$D = X'Y'B_{in} + X'YB'_{in} + XY'B'_{in} + XYB_{in}$$
$$B = X'Y + X'YB_{in} + YB_{in}$$

1.4.1.5 Parallel Adder (Ripple Carry Adder)

A parallel adder consists of full adders connected in cascade, that is, carry output of each adder is connected to the carry input of the next higher-order adder (Fig. 1.13). An n-bit parallel adder is constructed using $(n - 1)$ full adders and 1 half adder. As full adder consists of 2 half adders and 1 OR gate, so n-bit parallel adder can be made from $(2n - 1)$ half adders and $(n - 1)$ OR gates.

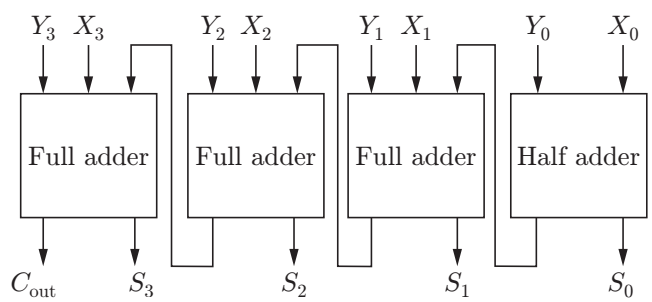


Figure 1.13 | Block diagram of a parallel adder.

1.4.1.6 Look-Ahead Carry Header

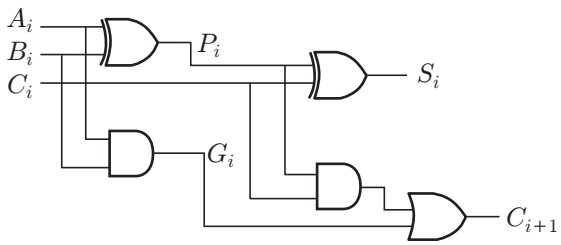
In a parallel adder, carry propagation delay is present. To overcome this difficulty, look-ahead carry adder is used. It uses logic gates to look at lower-order bits of augend and addend to check if higher-order carry is generated or not. It uses two functions, carry generate and carry propagate, for the same (Fig. 1.14).

Problem 1.13: How many AND and OR gates are required to design 7-bit look-ahead carry adder.

Solution: To design a 7-bit look-ahead carry adder, we need

$$\text{AND gate} = \frac{n(n+1)}{2} = \frac{7(7+1)}{2} = 28$$
$$\text{OR gate} = n = 7$$

where n is the number of bits.



Full Adder circuit with P_i and G_i

Carry propagation $P_i = A_i \oplus B_i$

Carry generation $G_i = A_i B_i$

Output sum $S_i = P_i \oplus C_i$

Output carry $C_{i+1} = G_i + P_i C_i$

Figure 1.14 | Logic diagram of a look-ahead adder.

1.4.1.7 Decoder

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines (Fig. 1.15). The size of a decoder is n to m where $m \leq 2^n$. It is used to route input data to a specified output line for memory addressing (Fig. 1.16 and Table 1.8).

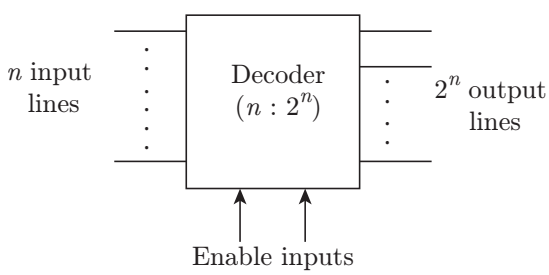


Figure 1.15 | Block diagram of a decoder.

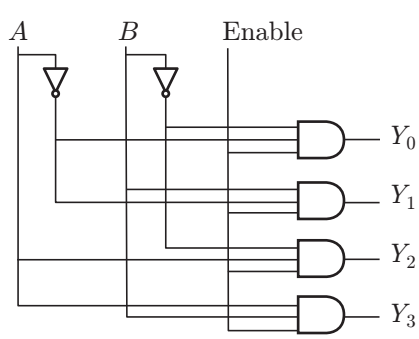


Figure 1.16 | Logic diagram of a 2X4 decoder.

Table 1.8 | Truth table of 2X4 decoder

Inputs		Outputs			
A	***B***	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

1.4.1.8 Encoder

An encoder is a combinational circuit that has 2^n (or fewer) inputs and n output lines (Fig. 1.17). It accepts an active level on one of its inputs representing a decimal or octal digit and converts it to a coded output.

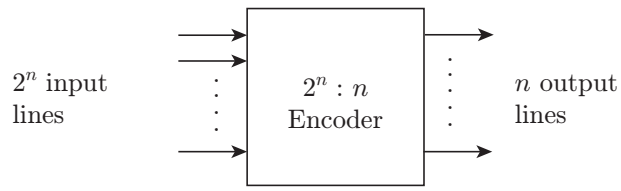


Figure 1.17 | Block diagram of an encoder.

1.4.1.9 Multiplexer

A multiplexer is a combinational circuit that provides inputs one by one at single output; and which input will be out at any instant depends upon the combination at select lines (Fig. 1.18). It converts parallel data to serial data (Fig. 1.19 and Table 1.9).

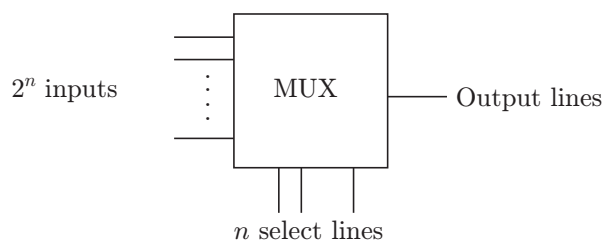


Figure 1.18 | Block diagram of a multiplexer.

Table 1.9 | Truth table of 4 to 1 MUX

Select Lines		Outputs
S_1	S_0	***Y***
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

The Boolean function is

$$Y = I_0S_0'S_1' + I_1S_0S_1' + I_0S_0'S_1 + I_0S_0S_1$$

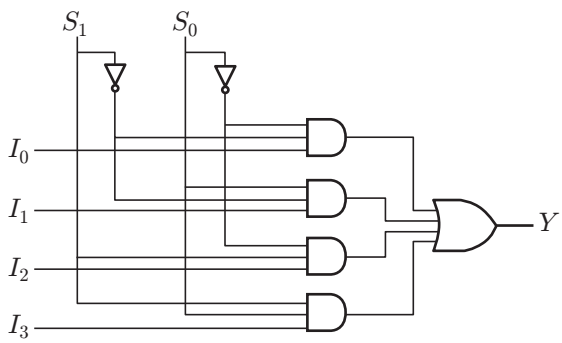


Figure 1.19 | Logic diagram of 4 to 1 MUX.

1.4.1.10 Boolean Function Implementation Using Multiplexers

1. Take one variable for input lines and others for selection lines.
2. Write the minterms with the variable in complemented form in the first row and with variable selected in uncomplemented form in the second row.
3. Then encircle the minterms which are present in the function.
   - If there is circled variable in the column, then we put 0 on the corresponding line.
   - If there are circled variables, then we put 1 on the line.
   - If bottom variable is circled and top is not circled, A is used in input line.
   - If bottom variable is not circled and top is circled, A' is used as input to that line.

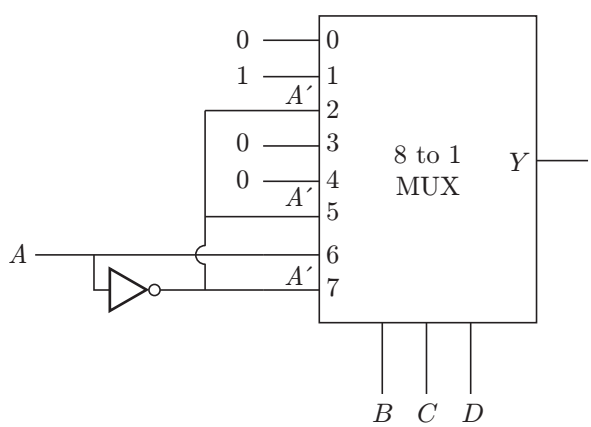
Problem 1.14: Implement the function $F(A, B, C, D) = \Sigma m(1, 2, 5, 7, 9, 14)$ using MUX.

Solution: Select variable A for input as B, C and D for selection lines.

As there are four variables so size of MUX is $2^{N-1} = 2^3 = 8$ to 1 MUX.

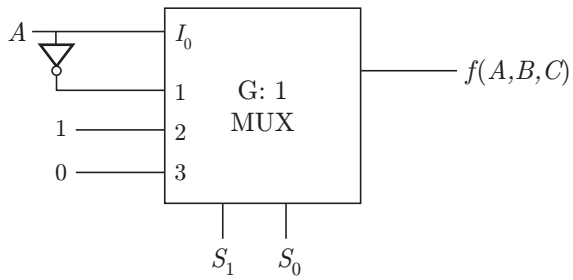
1. Minterms with A' are $0 - 7$.
2. Minterms with A are $8 - 15$.

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	25
	0	1	A'	0	0	A'	A	A'



Logic diagram

Problem 1.15: The combinatorial circuit below is equivalent to _____.



Solution:

	I_0	I_1	I_2	I_3
A'	0	1	2	3
A	4	5	6	7
	A	A'	1	0

It is equivalent to $\sum m(1, 2, 4, 6)$.

1.4.1.11 Demultiplexer

A demultiplexer is a combinational circuit that receives information on a single line and transmits this on one of 2^n of possible output line (Fig. 1.20). The selection of possible output line is controlled by bit values of n select lines.

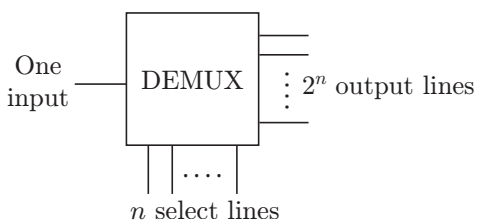


Figure 1.20 Block diagram of a demultiplexer.

Implementation of higher order decoders and multiplexers by using lower order. Whenever odd number implementation is there, one NOT gate is required.

Example 1.3

Design 4 ×16 decoders using 2 × 4 decoders

16/4 = 4

4/4 = 1

Total = 5

Example 1.4

Design 8 × 256 decoders using 4 × 64 decoders

256/64 = 4

4/64 = 0

Total = 4

Example 1.5

Design 6 × 64 decoders using 2 × 4 decoders

64/4 = 16

16/4 = 4

4/4 = 1

Total = 21

1.4.2 Sequential Circuits

A logic circuit whose output at any instant of time depends on the present input as well as past output is known as sequential circuit. It contains memory element.

1.4.2.1 Flip–Flops

A flip–flop or bistable multivibrator is a synchronous bistable circuit made up of logic gates that can exist in either of two stable states in presence of a falling or raising edge of a clock signal. Flip–flops can change its state by means of some external signal. It can "store" binary information. There are different types of flip–flops based on the manner the inputs affect their output state:

1. *RS* flip–flop
2. *D* flip–flop
3. *JK* flip–flop
4. *T* flip–flop

RS Flip–Flop (Direct Coupled)

This sequential logic circuit has two stable states ($Q = 1$ set state and $Q = 0$ reset state) achieved by giving proper inputs to R and S inputs (Figs. 1.21 and 1.22 and Table 1.10). If the circuit is in one particular state it continues to remain in that state and can store 1-bit data.

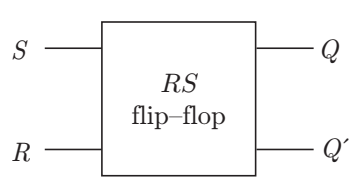


Figure 1.21 Block diagram of an *RS* flip–flop.

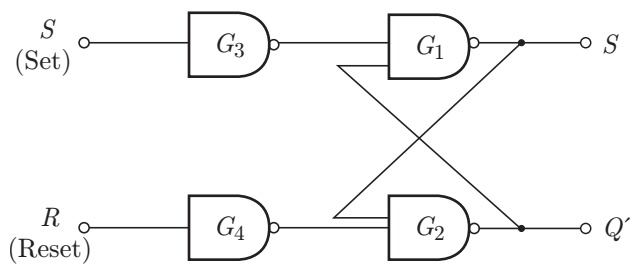


Figure 1.22 | Logic diagram of an *RS* flip−flop.

Table 1.10 | Truth table of *RS* flip−flop

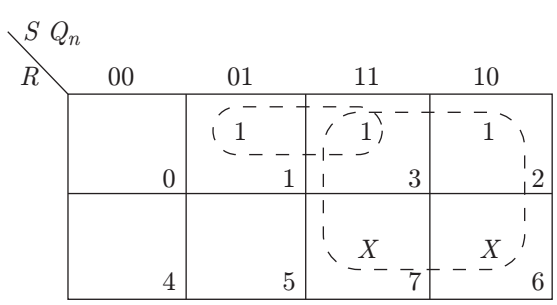
Inputs	Outputs	
S_n	R_n	Q_{n+1}
0	0	Q_n (No change)
0	1	0 (Reset)
1	0	1 (Set)
1	1	Prohibited

The characteristic equation which gives the algebraic description of the next state of a flip−flop obtained by K-map simplification is as follows (see Table 1.11):

$$Q_{n+1} = S + R'Q_n$$

Table 1.11 | Truth table

Flip−Flop Inputs		Present State	Next State
R_n	S_n	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	X
1	1	1	X



$Q_{n+1} = S + R'Q_n$

Clocked *RS* Flip−Flop

This circuit sets or resets the memory cell in synchronization with the clock pulse (Figs. 1.23 and 1.24 and Table 1.12). If clock pulse is not present, the gates G3 and G4 are inhibited; their outputs are 1.

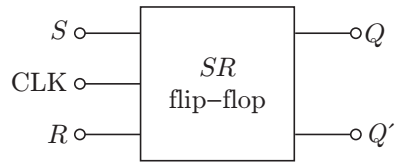


Figure 1.23 | Block diagram of a clocked *SR* flip−flop.

Table 1.12 | Truth table of *SR* flip−flop

Clock	Flip−Flop Inputs		Outputs
CLK	S_n	R_n	Q_{n+1}
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	Prohibited
0	X	X	Q_n

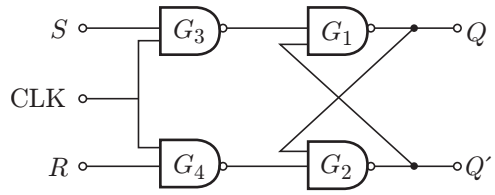
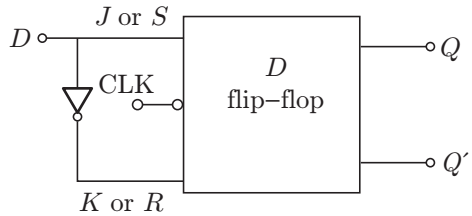


Figure 1.24 | Logic diagram of a clocked *SR* flip−flop.

D Flip−Flop

This circuit has only one input terminal, and output is followed by input. It has delay of exactly one clock cycle (Figs. 1.25 and 1.26 and Table 1.13). The input to *R* is through an inverter from *S*.



JK or *SR* Flip−Flop converted to *D*

Figure 1.25 | Block diagram of a *D* flip−flop.

Table 1.13 Truth table of D flip−flop

Flip−Flop Inputs	Present State	Next State
D_n	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

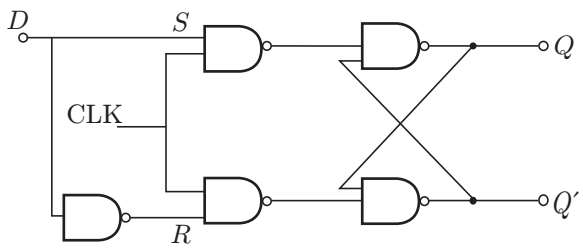


Figure 1.26 Logic diagram of a D flip−flop.

The characteristic equation is $Q_{n+1} = D$ (Fig. 1.27)

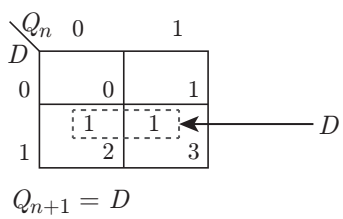


Figure 1.27 K-Map

JK Flip−Flop

The uncertainty in SR flip−flop when $S_n = R_n = 1$ can be avoided by using JK flip−flop (Figs. 1.28 and 1.29 and Table 1.14). The output of JK flip−flop is connected back to the inputs of gate. The inputs J and K are ANDed with Q and Q' to obtain S and R inputs:

$$S = JQ'$$
$$R = J \cdot Q$$

The flip−flop sets when input J is high and resets when input K is high. If $J = K = 1$, the flip−flop will toggle on the next positive clock.

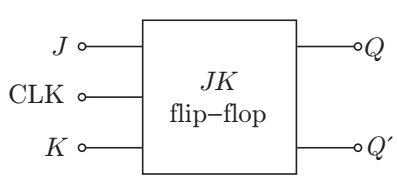


Figure 1.28 Block diagram of a JK flip−flop.

Table 1.14 Truth table of JK flip−flop

Flip−Flop Inputs		Present State	Next State
J_n	K_n	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

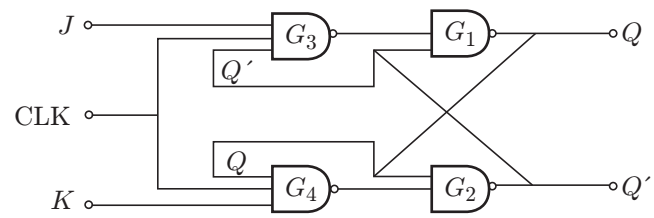


Figure 1.29 Logic diagram of a JK flip−flop.

The characteristic equation is $Q_{n+1} = JQ'_n + KQ_n$ (Fig. 1.30).

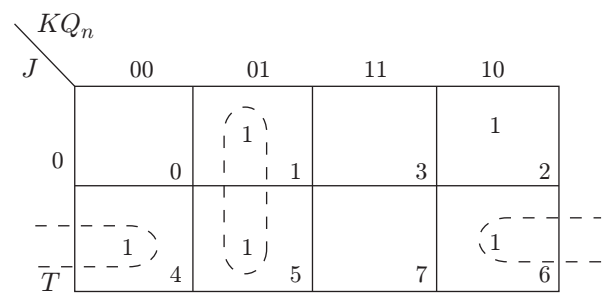


Figure 1.30 K-Map

Repetition of toggle for a single-clock pulse is known as **race-around problem**. The output of JK flip−flop will oscillate between 0 and 1 during a single-clock pulse so that at the end of clock pulse the value of Q is uncertain. This problem can be avoided by decreasing the pulse width, by edge triggering or increasing propagation delay.

Master−Slave *JK* Flip−flop

It is a cascade of two RS flip−flops (master and slave) with feedback from outputs of slave to input of master to avoid race-around problem. Positive clock pulses are applied to first flip−flop and pulses are inverted before applying to slave flip−flop (Fig. 1.31).

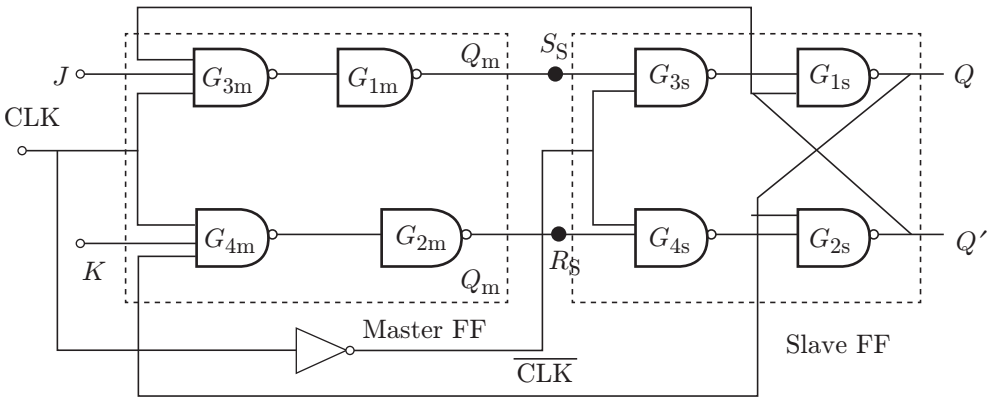


Figure 1.31 | Block diagram of a master−slave *JK* flip−flop.

T Flip−flop

It is also known as toggle flip−flop, which is obtained by connecting both inputs of *JK* flip−flop (Fig. 1.32 and Table 1.15).

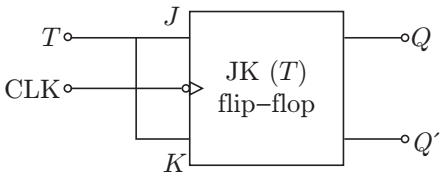
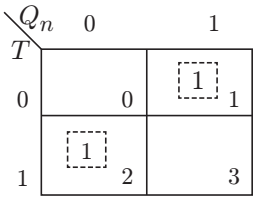


Figure 1.32 | Block diagram of a *T* flip−flop.

Table 1.15 | Truth table of *T* flip−flop

Flip−Flop Inputs	Present State	Next State
T_n	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0



$$Q_{n+1} = T'Q_n + TQ'_n$$

Figure 1.33 | K-Map

Excitation Table of Flip−Flop

The excitation table of flip−flop states is given in Table 1.16.

Conversion Equation

The flip−flop conversion equations are given in Table 1.17.

Table 1.16 | Excitation Table of Flip−flop States

Flip−Flop States		RS Flip−Flop Inputs		JK Flip−Flop Inputs		D Flip−Flop Inputs	T Flip−Flop Inputs
Present State (Q_n)	Next State (Q_{n+1})	R	S	J	K		
0	0	X	0	0	X	0	0
0	1	0	1	1	X	1	1
1	0	1	0	X	1	0	1
1	1	0	X	X	0	1	0

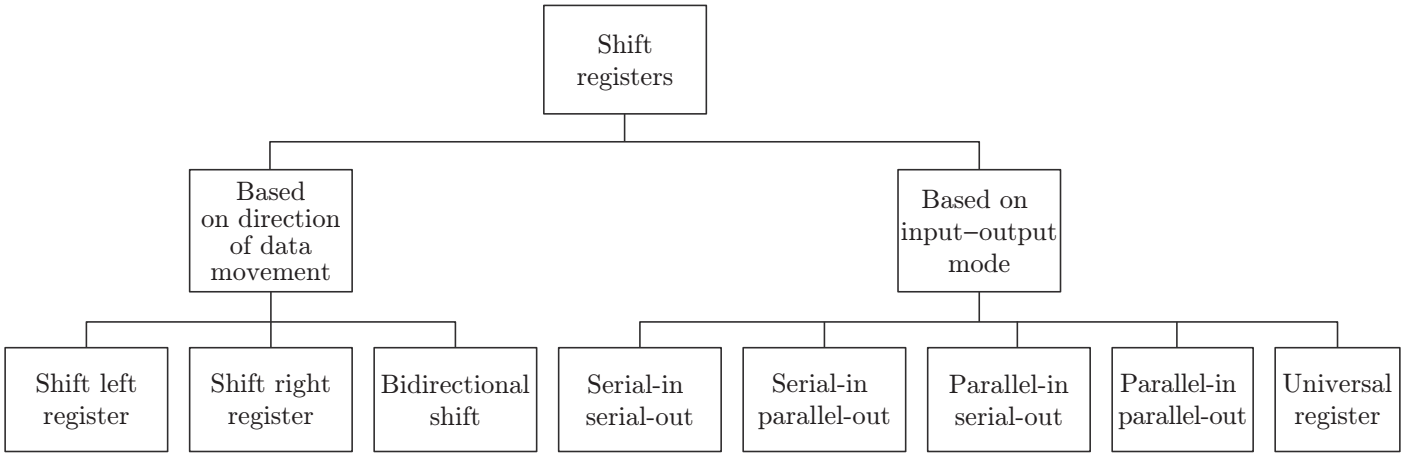


Figure 1.34 | Types of registers

Table 1.17 | Flip–flop conversion equations

1	SR to JK	$S = JQ'$ $R = K \cdot Q$
2	SR to D	$S = D$ $R = D'$
3	SR to T	$S = TQ'$ $R = TQ$
4	JK to SR	$J = S$ $K = R$
5	JK to D	$J = D$ $K = D'$
6	JK to T	$J = T$ $K = T$
7	D to JK	$D = TQ' + K'Q$
8	D to SR	$D = S + R'Q$
9	D to T	$D = T \oplus Q$
10	T to SR	$T = SQ' + RQ$
11	T to D	$T = D \oplus Q$
12	T to JK	$T = JQ' + KQ$

Applications of Flip–Flops

Registers

Registers are collection of flip–flops, and each flip–flop is capable of storing single bit of information. They make stored information available to the logic elements for the computation. There are different types of registers as shown in Fig. 1.34.

1. **Serial-In/Serial-Out Shift Register:** It accepts data serially on single input line and produces output in serial form. Data can be shifted right or left.
   - **Shift right register:** In shift right register, serial data is entered from the left side of a register and leaves from the right side. For example, a 4-bit number 1101 ($Q_D Q_C Q_B Q_A$) is entered serially into first flip–flop D from the left side (Fig. 1.35).

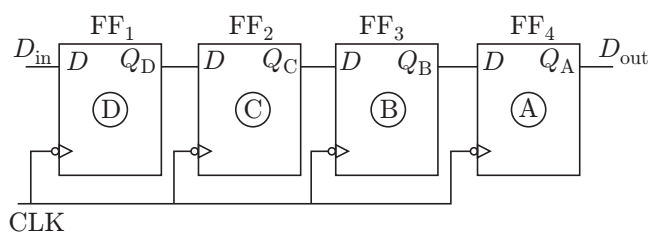


Figure 1.35 | 4-Bit shift right register.

The content of register after each clock pulse is shown in Table 1.18.

Table 1.18 | Content of register after each clock pulse

Clock Pulse	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	0	1	0
4	1	1	0	1

- **Shift left register:** In shift left register, serial data is entered from the right side of the register and leaves from the left side. For example, a 4-bit number 1101 ($Q_D Q_C Q_B Q_A$) is entered serially into input of shift left register (Fig. 1.36).

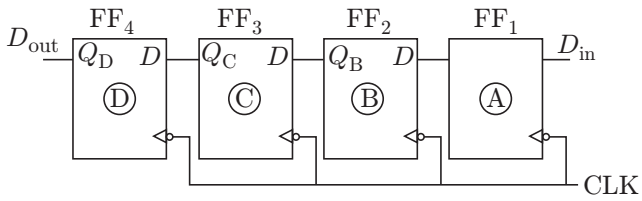


Figure 1.36 | 4-Bit shift left register.

The content of register after each clock pulse is given in Table 1.19.

Table 1.19 | Content of register after each clock pulse

Clock Pulse	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	0
4	1	1	0	1

Note: A universal register can perform all the operations of shift register.

2. **Serial-In to Parallel-Out (SIPO):** The register is loaded with data serially, one bit at a time, with the stored data being available at the output in parallel form (Fig. 1.37).
The content of register after each clock pulse is given in Table 1.20.

Table 1.20 | Content of register after each clock pulse

Clock Pulse	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	0	0	0	0

3. **Parallel-In to Serial-Out (PISO):** The parallel data is loaded simultaneously into the register and is shifted out of the register serially, one bit at a time (Fig. 1.38).

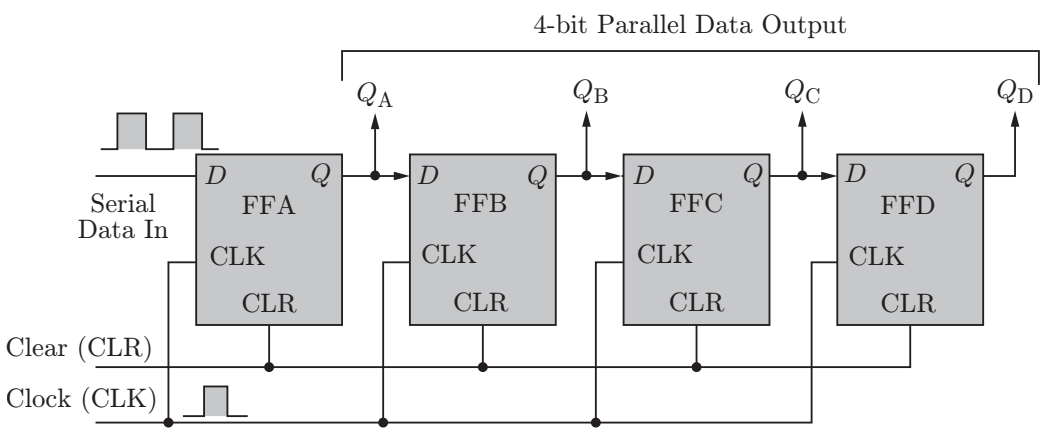


Figure 1.37 | 4-Bit SIPO register.

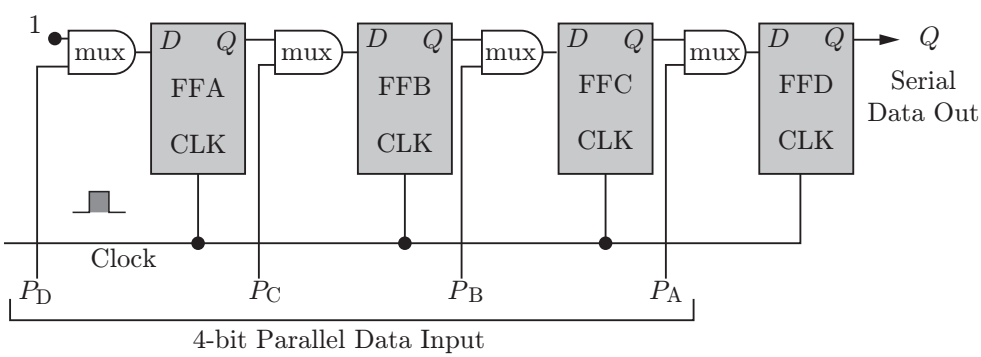


Figure 1.38 | 4-Bit PISO register.

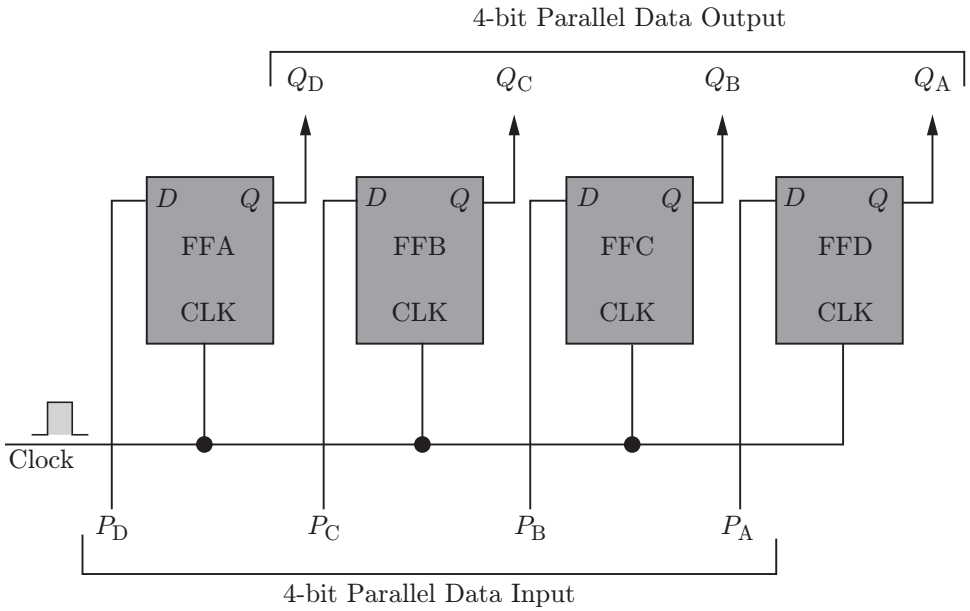


Figure 1.39 | 4-Bit PIPO register.

4. **Parallel-In to Parallel-Out (PIPO):** The parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse (Fig. 1.39).

Counters

A counter is a sequential logic circuit that is capable of counting the number of clock pulses arriving at its input in a cyclic sequence. It is just a register that shifts through a predetermined sequence of states upon the application of input clock pulses. There are two different types of counters – synchronous counter (parallel counter) and asynchronous counter (serial or ripple counter) (Table 1.21).

Table 1.21 | Differences between asynchronous and synchronous counter.

Asynchronous Counter	Synchronous Counter
The output of first flip–flop drives the clock for next flip–flop. So all flip–flops are not clocked simultaneously.	There is no connection between output of first flip–flop and clock input of next flip–flop. All the flip–flops are clocked simultaneously.
Speed is low.	Speed is high.
Simple logic circuit as number of states increases.	Complex design as number of states increases.
Example: Ring counter, Johnson counter.	Example: Binary ripple counter, Up-down counter.

The counter with n flip–flops has a maximum modulus 2^n so it can count from 0 to 2^{n-1}. It can also produce MOD numbers less than 2^n by skipping the states.

1. **4-Bit Asynchronous Up Counter:** Figures 1.40–1.42 show a 4-bit asynchronous counter constructed using JK flip–flops. The clock input of first flip–flop (the one with the Q_0 output) is connected to the external clock. The clock of second flip–flop drives from output Q_0 of the first flip–flop and so on. Each flip–flop input ($J = K = 1$) is connected to HIGH voltage so it toggles when negative edge is arrived at clock input and counts from 0000 to 1111.

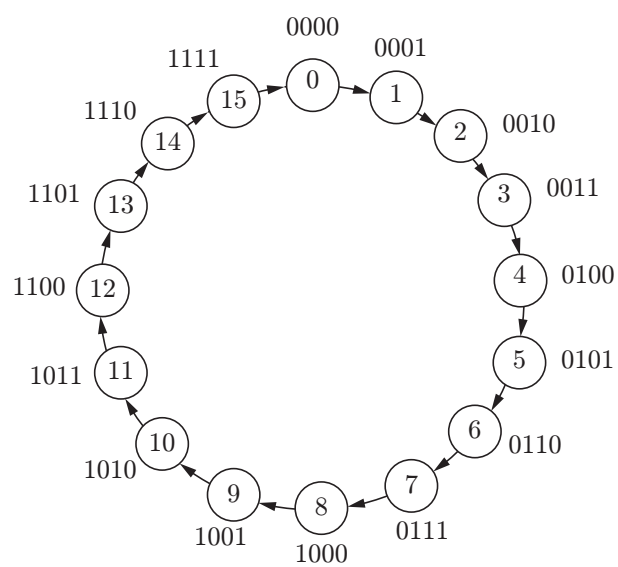


Figure 1.40 | State diagram of a 4-bit asynchronous counter.

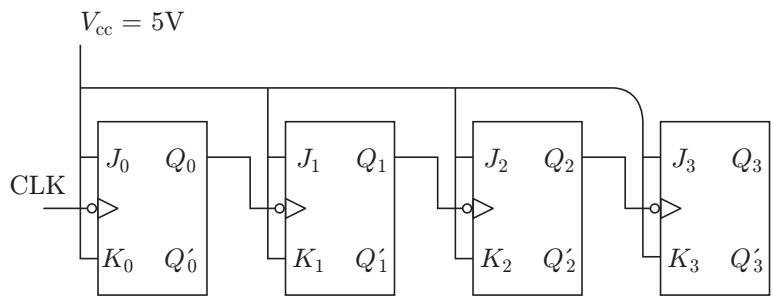


Figure 1.41 | Logic diagram for a 4-bit asynchronous counter.

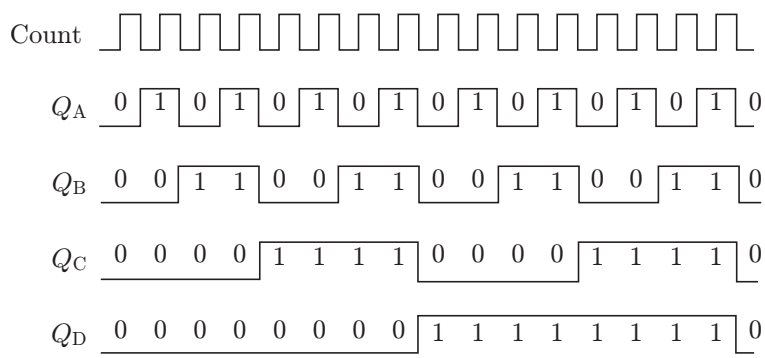


Figure 1.42 | Timing diagram for a 4-bit asynchronous counter.

2. **MOD-10 Asynchronous Counter:** A MOD-10 asynchronous counter has 10 states and number of flip−flops required are $2^4 \geq 10$ (Figs. 1.43 and 1.44). It counts the sequence from 0000 to 1001 and after that it returns to the initial state.

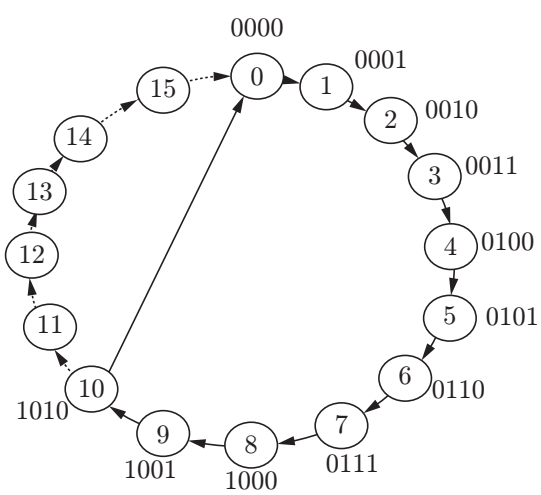


Figure 1.43 | State diagram of a 4-bit MOD-10 counter.

3. **4-Bit Synchronous Counter:** In a 4-bit synchronous counter, the least significant bit of flip−flop is connected at high level and other inputs of *JK* flip−flop are driven by some combination of flip−flop outputs. Figure 1.45 shows the logic diagram of a 4-bit synchronous counter. The output of flip−flop0 always toggle, and that of flip−flop1 toggles when Q_0 is 1, otherwise it maintains the previous state. The flip−flop 2 toggles when both Q_1 and Q_0 are 1. The flip−flop 3 toggles when Q_2, Q_1 and Q_0 are 1 (Table 1.22).

Table 1.22 | Truth table for 4-bit synchronous counter

Count	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	0	0
15	1	1	0	1

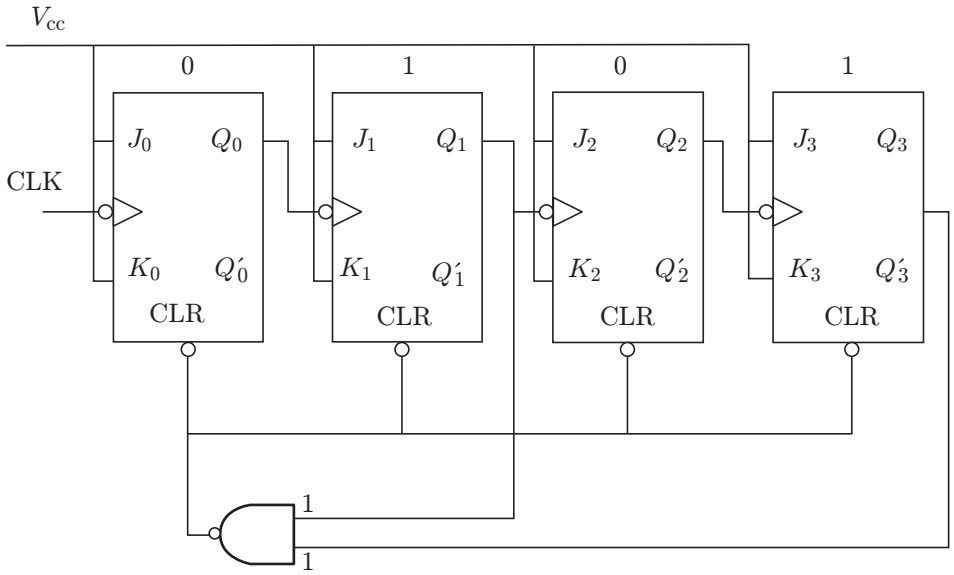


Figure 1.44 | Logic diagram for a MOD-10 asynchronous counter.

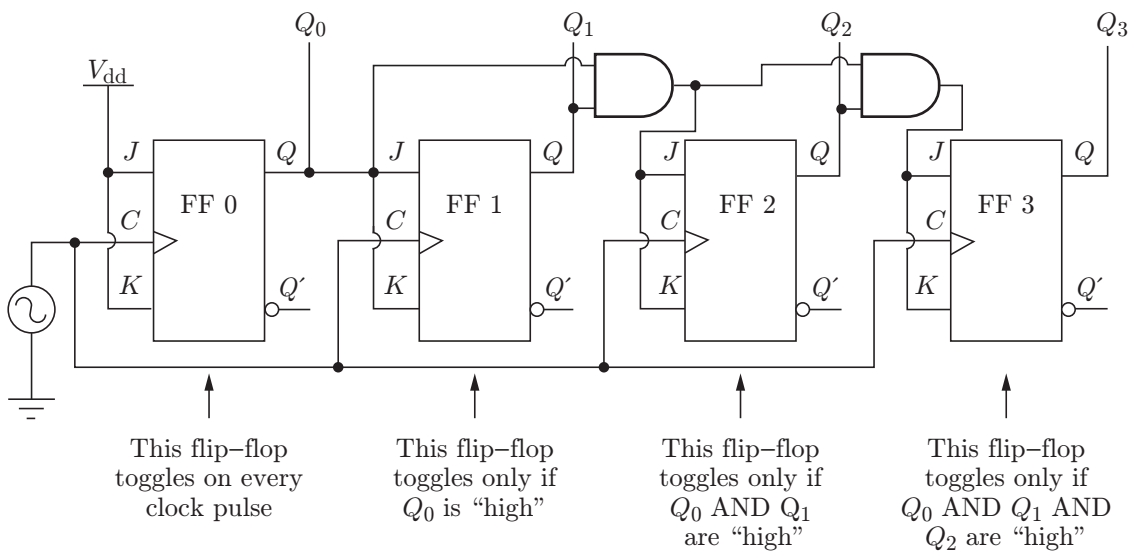


Figure 1.45 | Logic diagram of a 4-bit synchronous counter.

4. MOD-5 Synchronous Counter:

Step 1: Number of states is five, so counting sequence 0 to 4.

Flip−flops required are $2^n \geq 5$, so $n = 3$.

Let us label these flip−flops as A, B and C.

Step 2: Obtain the excitation table for *JK* flip−flop (Table 1.23).

Step 3: Develop state table using the excitation table (Table 1.24).

Step 4: K-map simplification for finding expressions corresponding to each input (Fig. 1.46).

Table 1.23 | Excitation table for *JK* flip−flop

Flip−Flop States		JK Flip−Flop Inputs	
Present State (Q_n)	**Next State (Q_{n+1})**	***J***	***K***
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Table 1.24 | State table

Present State			Next State			Flip–Flop Inputs					
Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	0	0	0	X	1	0	X	0	X
1	0	1	X	X	X	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X	X	X

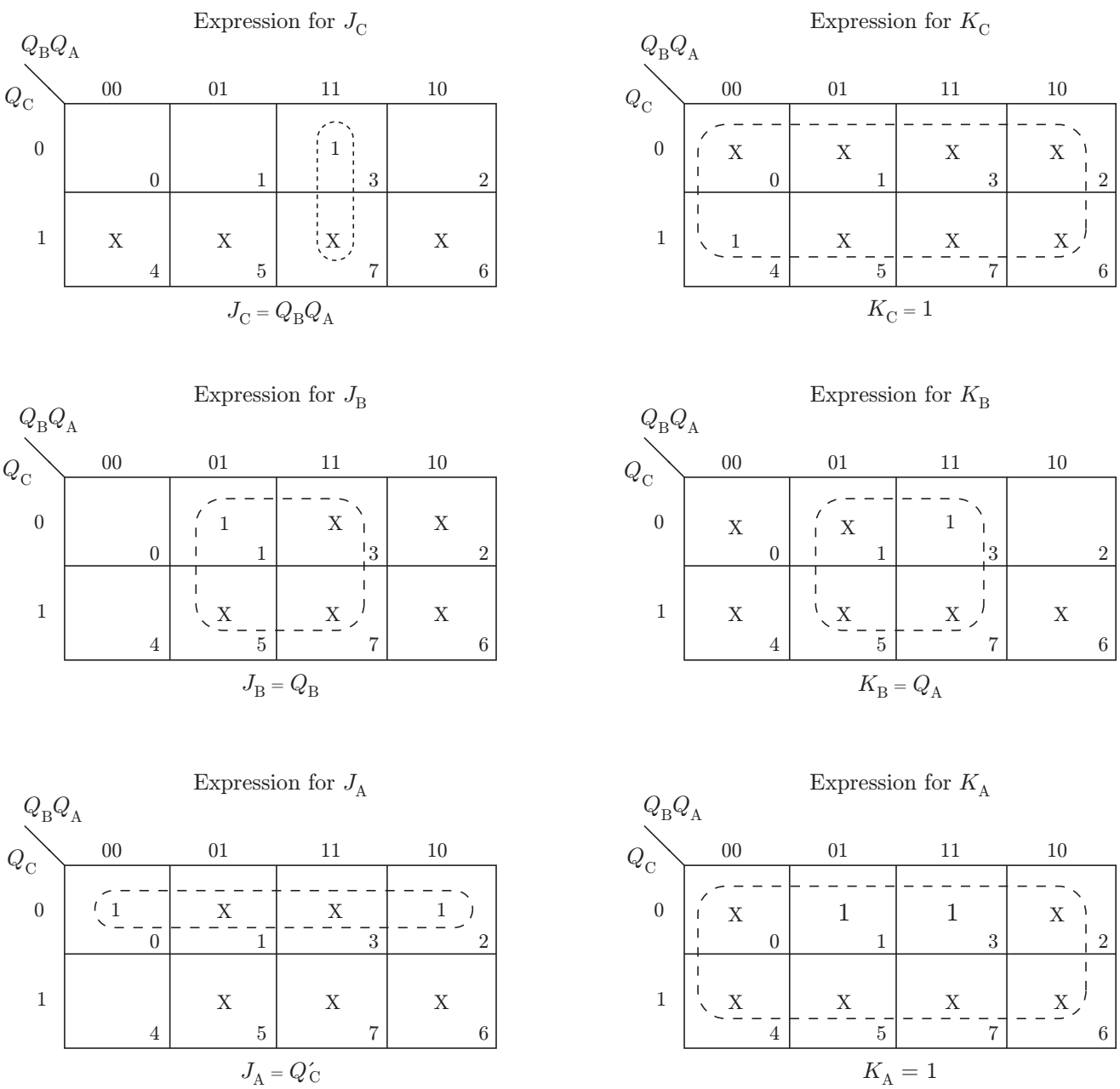


Figure 1.46 | K-map simplification.

Step 5: Logic diagram of MOD-5 counter (Fig. 1.47).

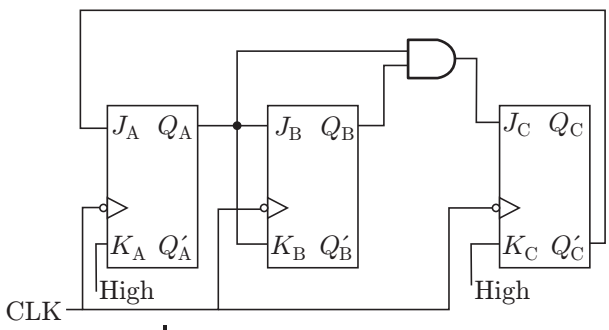


Figure 1.47 | Logic diagram of a MOD-5 counter.

5. **Shift register counter:** It is basically a shift register whose serial output is connected back to serial input in order to produce special sequences.
6. **Ring counter:** A ring counter is a circular shift register in which output of last flip−flop is connected to the input of the first in a ring (Fig. 1.48 and Table 1.25). In this, only one flip−flop is in state one while others are in their zero states. Each state repeats after every n clock cycles, where n is the number of flip−flops used. Initially the counter is reset by applying low signal to active low CLR input. All flip−flops reset except the first flip−flop.

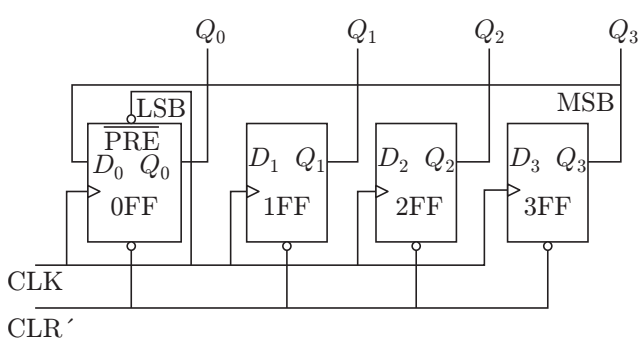


Figure 1.48 | Logic diagram of a 4-bit ring counter.

Table 1.25 | Truth table of 4-bit ring counter

CLK	CLR′	Q_3(MSB)	Q_2	Q_1	Q_0
0	0	0	0	0	1
1	1	0	0	1	0
1	1	0	1	0	0
1	1	1	0	0	0

(*Continued*)

Table 1.25 | Continued

CLK	CLR′	Q_3(MSB)	Q_2	Q_1	Q_0
1	1	0	0	0	1
1	1	0	0	1	0
1	1	0	1	0	0
1	1	1	0	0	0

7. **Johnson counter:** A Johnson counter (switch-tail ring counter, twisted-ring counter or walking-ring counter) is a variation of shift register counter, where the complement of the output of the last flip−flop is fed back as input to the first flip−flop (Fig. 1.49 and Table 1.26). The counter has counting sequence length of twice the length of the shift register, that is, number of flip−flops.

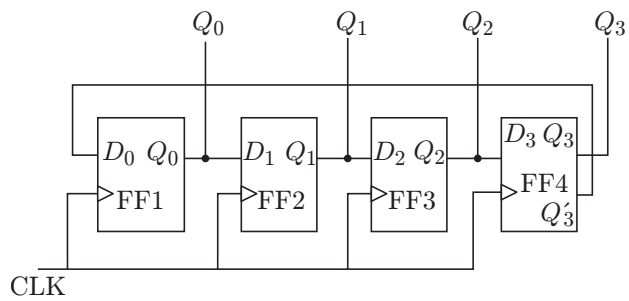


Figure 1.49 | 4-Bit Johnson counter.

Table 1.26 | Truth table of 4-bit Johnson counter

CLK	Q_0	Q_1	Q_2	Q_3	Q'_3
0	0	0	0	0	1
1	1	0	0	0	1
2	1	1	0	0	1
3	1	1	1	0	1
4	1	1	1	1	0
5	0	1	1	1	0
6	0	0	1	1	0
7	0	0	0	1	0
8	0	0	0	0	1

1.5 DIGITAL LOGIC FAMILIES

Refer Fig. 1.50 and Table 1.27 for classification and pros and cons of logic families.

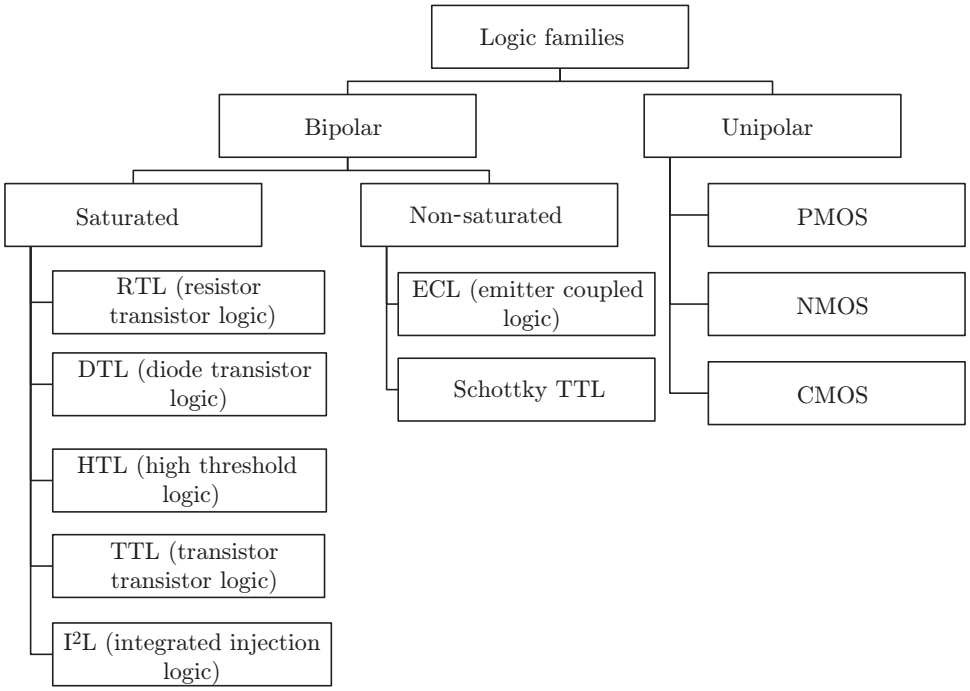


Figure 1.50 | Digital logic families.

Table 1.27 | Pros and cons of logic families

Logic Families	Pros	Cons
RTL	Economical and simple to design, can easily interface with other logic families	Low threshold and fan out, poor noise immunity, low speed and low power dissipation
DTL	High speed with 30-ns propagation delay, low power dissipation	Poor noise immunity and fan out
TTL	Fastest saturating logic family, high-speed Schottky, low-power Schottky, noise margin about 0.4 V, more fan out, can drive up to 10 gates	Power dissipation is more than MOS gates, noise immunity is not very high
ECL	Highest speed because transistors operate in active region, propagation delay of 2-ns, parameters resistant to temperature change	Very low noise margin, highest power dissipation, high cost
NMOS	Very low power dissipation, large fan out up to 20, very high noise margin	High power dissipation than CMOS, low speed of operation, large propagation delay per gate
I^2L	Composed of BJT so high speed of operation, low power dissipation, low power supply requirement	Lower packing density than NMOS, lower noise margin, external resistance required for proper functioning
CMOS	Large fan out (>50), lowest power dissipation, very high noise immunity, temperature resistant	Increased cost, less packing density than NMOS

IMPORTANT FORMULAS

1. Null − All the outputs are zero.
2. Identity − All the outputs are one.
3. X inhibit Y − Represented as $(x \cdot \sim y)$
4. Y inhibit X − Represented as $(y \cdot \sim x)$
5. Transfer − Same as y
6. AND − $x \cdot y$ (Similar to intersection in set theory)
7. OR − $x + y$ (Similar to union in set theory)
8. NOT X− $\sim x$ (Also known as complement of x)
9. NOT Y− $\sim y$ (Also known as complement of y)
10. XOR − $(\sim x \cdot y) + (x \cdot \sim y)$ (Also known as parity checker)
11. XNOR − $(x \cdot y) + (\sim x \cdot \sim y)$
12. NAND − $\sim(x \cdot y)$
13. NOR − $\sim(x + y)$
14. X implication Y $(\sim x + y)$
15. Y implication X $(\sim y + x)$
16. $X + 0 = X$
17. $X + (\sim X) = 1$
18. $X + X = X$
19. $X + 1 = 1$
20. $X + Y = Y + X$
21. $X + (Y + Z) = (X + Y) + Z$
22. $X \cdot (Y + Z) = X \cdot Y + X \cdot Z$
23. $\sim(X + Y) = \sim X \cdot \sim Y$ (Also called **DeMorgan's law**)
24. $X + (X \cdot Y) = X$
25. $\sim(\sim X) = X$

Duality Theorem:

26. $X \cdot 1 = X$
27. $(X \cdot \sim X) = X \cdot X' = 0$
28. $X \cdot X = X$
29. $X \cdot 0 = 0$
30. $X \cdot Y = Y \cdot X$
31. $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$
32. $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$
33. $(X \cdot Y)' = X' + Y'$
34. $X \cdot (X + Y) = X$

Half Adder:

35. $S = X'Y + XY'$
36. $C = XY$

Full Adder:

37. $S = X'Y'C_{\text{in}} + X'YC'_{\text{in}} + XY'C'_{\text{in}} + XYC_{\text{in}}$
38. $C_{\text{out}} = AC_{\text{in}} + XY + YC_{\text{in}}$

Half Subtractor:

39. $D = XY' + X'Y$
40. $B = X'Y$

Full Subtractor:

41. $D = X'Y'B_{\text{in}} + X'YB'_{\text{in}} + XY'B'_{\text{in}} + XYB_{\text{in}}$
42. $B = X'Y + X'YB_{\text{in}} + YB_{\text{in}}$

Characteristic Equations:

43. SR flip−flop: $Q_{n+1} = S + R'Q_n$
44. D flip−flop: $Q_{n+1} = D$
45. JK flip−flop: $Q_{n+1} = JQ'_n + KQ_n$
46. T flip−flop: $Q_{n+1} = TQ'_n + T'Q_n$

SOLVED EXAMPLES

1. A correct output is achieved from a master−slave JK flip−flop only if its inputs are stable while the

   (a) Clock is LOW
   (b) lave is ready to receive the input from master
   (c) Master gives a reset signal to slave
   (d) Clock is HIGH

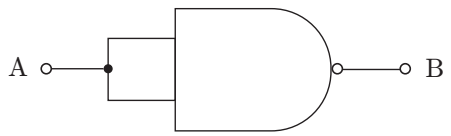
   Solution: The clock is high, only the flip−flops are stable when the inputs are stable.

   Ans. (d)

2. An inverter circuit can be realized with how many NAND gates?

(a) 1 (b) 2
(c) 3 (d) Not possible

Solution: An inverter or logic NOT gate can be made using standard NAND gate by connecting together **ALL** their inputs to a common input signal, for example,



Two-input NAND gate equivalent

Ans. (a)

3. A modulus-12 ring counter requires a minimum of _____.

(a) 10 flip−flops (b) 12 flip−flops
(c) 8 flip−flops (d) 6 flip−flops

Solution: In the ring counter, n flip-flops are used for the mod-n. 12 flip−flops because ring counter uses states n, so here modulus is 12.

Ans. (b)

4. How is a strobe signal used when serially loading a shift register?

(a) To turn the register set and reset
(b) To control the number of clocks
(c) To determine which output has a HIGH value
(d) To cascade it with other shift register

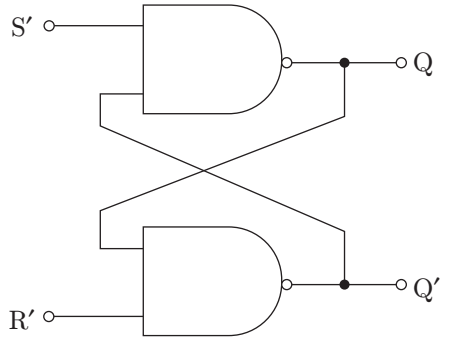
Solution: In digital circuits, a ***shift register*** is a cascade of flip−flops, sharing the same clock each time. This is called clocking or ***strobing*** to the register to set the clocks.

Ans. (b)

5. An active-HIGH *SR* latch has a 0 on the *S* input and a 1 on the *R* input. Now when the *R* input goes to 0, the latch will be _____.

(a) SET (b) RESET
(c) CLR (d) STROBE

Solution:



The SR (Set-Reset) flip−flop is a simple sequential circuit which consists of two gates connected as shown above. The output of each gate is connected to one of the inputs of the other gate, thus giving positive feedback or cross-coupling. The circuit has two active low inputs NOT S and NOT R and two outputs Q and NOT Q. When there is 0 on the S input and a 1 on the R input, then NOT S is 0 and NOT R is 1, so Q is 0 and NOT Q is 1. Thus, Q is reset to 0 by 0 on NOT R.

Ans. (b)

6. The NAND or NOR gates are referred to as "universal" gates because

(a) they are easier to implement.
(b) they can be used to build all the other types of gates.
(c) they are available in different IC families.
(d) they are available everywhere in the world.

Solution: NAND and NOR gates are universal because all Boolean functions that can be implemented by AND, OR and NOT gates can be implemented by NAND and NOR gates.

Ans. (b)

7. Match the following:

IC Family	GATE
TTL	NOR
CMOS	NOT
ECL	NAND

(a) TTL:NAND; ECL:NOR; CMOS:NOT
(b) TTL:NAND; ECL:NOT; CMOS:NOR
(c) TTL:NOT; ECL:NAND; CMOS:NOR
(d) TTL:NOR; ECL:NOT; CMOS:NAND

Solution: The NAND function is actually the simplest, most natural mode of operation for this TTL design. The basic gates are OR/NOR gates for ECL Construction. Inverters can be constructed using two complementary transistors in a CMOS configuration.

Ans. (a)

8. Which of the following satisfies the commutative law but not the associative law?

(a) NOR (b) OR
(c) XOR (d) NAND

Solution
Let $\uparrow$ signify the NAND operation. Then there exist propositions p, q, r such that

$$p\uparrow (q\uparrow r) \nvdash (p\uparrow q) \uparrow r$$

This implies NAND is not associative.

$$p\uparrow q = q\uparrow p$$

This implies NAND is commutative.

Ans. (d)

9. Match the following:

IC Family	Characteristics
X. TTL	High fan-out
Y. ECL	Low propagation delay
Z. CMOS	High power dissipation

Code:

	X	Y	Z
(a)	3	2	1
(b)	3	1	2
(c)	2	1	3
(d)	1	2	3

Solution: The power dissipation is dependent on the power supply voltage, frequency, output load, and input rise time. Power dissipation for TTL gates is usually 10 mW per gate. ECL is the fastest logic gate and has least propagation delay of 2 ns. CMOS gates have high fan-outs.

Ans. (a)

10. Which of the following statements is false?

A. The 2's complement of 0 is 0.
B. In 2's complement, the leftmost bit cannot be used to express a quantity.
C. For an n-bit word (2's complement) which includes the sign bit, there are $2n - 1$ positive integers, $2n + 1$ negative integers and one 0 for a total of $2n$ unique states.
D. In 2's complement, the significant information is contained in the 1's of positive numbers and 0's of the negative numbers.

(a) A (b) B (c) C (d) D

Solution: For an n-bit word which includes the sign bit, there are $2^{n-1} - 1$ positive integers, 2^{n-1} negative integers and one 0, for a total of 2^n unique states.

Ans. (c)

11. The dual of any Boolean expression is obtained by

(a) Interchanging POS and SOP terms
(b) Interchanging 0's and 1's
(c) Interchanging Boolean sums and Boolean products and also interchanging 0's and 1's
(d) Interchanging Boolean sums and Boolean products

Solution: The dual of a Boolean expression is obtained by interchanging Boolean sums and products and interchanging 0's and 1's. For example, The dual of $x(y + 0) = x + (y \times 1)$.

Ans. (c)

12. A sequential circuit gives an output depending upon

(a) its present state inputs only.
(b) its present-1 state input only.
(c) both present- and past-state input.
(d) the size of the memory.

Solution: Sequential logic is a logic circuit in which the output depends not only on the present value of its input signals but on their past history. This is in contrast to *combinational logic*, where the output depends only on the present input.

Ans. (c)

13. Synchronization is achieved by a timing device called a _____.

(a) Strobe
(b) Reset
(c) CLK
(d) Master clock generator

Solution: Synchronization is achieved by a timing device called a clock generator that produces a periodic train of clock pulses.

Ans. (d)

14. A complete microcomputer system contains

(a) Microprocessor
(b) Memory
(c) I/O device
(d) All of the above

Solution: A microcomputer contains a microprocessor, ROM and RAM, I/O ports and bus or system of interconnecting wires, housed in a motherboard.

Ans. (d)

15. How many different Boolean functions of degree 4 are there?

(a) 2^4 (b) 2^8
(c) 2^{12} (d) 2^{16}

Solution: Different Boolean functions of degree n is given by 2^{2^n}.

Ans. (d)

16. A Boolean operator $\oplus$ is defined as follows:

$1 \oplus 1 = 1$, $1 \oplus 0 = 0$, $0 \oplus 1 = 0$ and $0 \oplus 0 = 1$
What will be the truth value of the expression $(x \oplus y) \oplus z = x \oplus (y \oplus z)$?

(a) Cannot be defined, as $\oplus$ is defined for two inputs.
(b) Always true.
(c) Always false.
(d) True when any one of the inputs is true.

Solution:

x	y	z	x⊕y	(x⊕y)⊕z	y⊕z	x⊕(y⊕z)
0	0	0	1	0	1	0
0	0	1	1	1	0	1
0	1	0	0	1	0	1
0	1	1	0	0	1	0
1	0	0	0	1	1	1
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

Ans. (b)

17. Let * be defined as $x * y = \overline{x} + y$. Let $z = x*y$, then value of $z*x$ is

(a) $\overline{x} + y$ (b) x (c) 0 (d) 1

Solution: Substituting value of $z = x*y$ in the expression:

$$z * x = (x * y) * x = (\overline{x} + y) * x = (\overline{\overline{x} + y}) + x$$
$$= x \cdot \overline{y} + x$$

Now using absorption law: $a + ab = a$

$$x \cdot \overline{y} + x = x$$

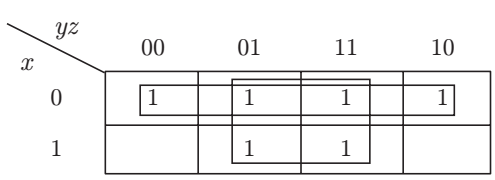
So, result is: $z*x = x$

Ans. (b)

18. Let $f(x,y,z) = \overline{x} + \overline{y}z + xz$ be a switching function. Which one of the following is valid?

(a) $\overline{y}x$ is a prime implicant of f.
(b) xz is a minterm of f.
(c) xz is an implicant of f.
(d) y is a prime implicant of f.

Solution:

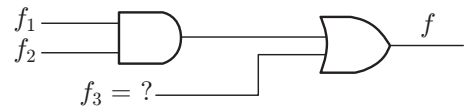


Number of possible joins are 2, so there are two prime implicants, that is, x' and z.

Both the prime implicants are essential. xz is an implicant of f.

Ans. (c)

19. Consider the logic circuit shown in the following figure. The functions f_1, f_2 and f (in canonical sum-of-products form in decimal notation) are



$f_1(w, x, y, z) = \sum (8, 9, 10)$
$f_2(w, x, y, z) = \sum (7, 8, 12, 13, 14, 15)$
$f(w, x, y, z) = \sum (8, 9)$

The function f_3 is

(a) $\sum$ 9, 10 (b) $\sum$ 9
(c) $\sum$ 1, 8, 9 (d) $\sum$ 8, 10, 18

Solution: By the logic circuit, function f is given as

$$f = f_1 f_2 + f_3$$
$$f_1 = \sum (8, 9, 10)$$
$$f_2 = \sum (7, 8, 12, 13, 14, 15)$$

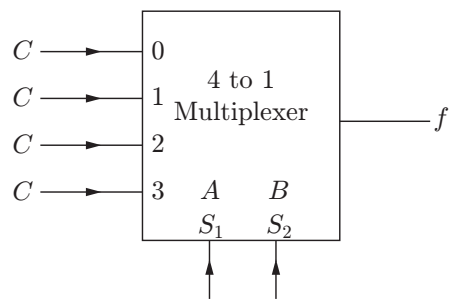
As according to logic, common minterm in f_1 and f_2 is 8, so

$$f_1 f_2 = \sum (8)$$

to get output as $f = \sum (8, 9)$, f_3 can be either $\sum (8, 9)$ or $f_3 = \sum 9$

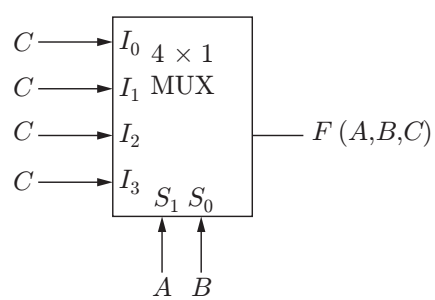
Ans. (b)

20. Consider the circuit shown in the following figure; f implements



(a) $\overline{A}\overline{B}C + \overline{A}B\overline{C} + ABC$ (b) $A + B + C$
(c) $A \oplus B \oplus C$ (d) C

Solution: The circuit for function $f(A, B, C)$ is given as follows



The output of function will be

$$\overline{A}\overline{B}C + \overline{A}BC + A\overline{B}C + ABC = \overline{A}C(\overline{B} + B) + AC(\overline{B} + B)$$
$$= \overline{A}C + AC = C$$

Ans. (d)

21. Which of the following functions implements the Karnaugh map shown below?

AB \ CD	00	01	11	10
00	0	0	1	0
01	X	X	1	X
11	0	1	1	0
10	0	1	1	0

(a) $\bar{A}B + CD$ (b) $D(C + A)$

(c) $AD + \bar{A}B$ (d) $(C + D)(\bar{C} + D)(A + B)$

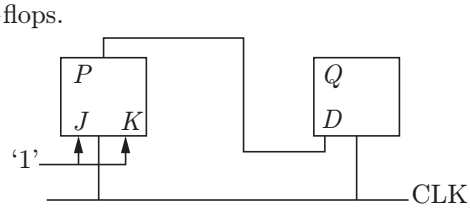
Solution: Karnaugh map of the function is given as follows:

AB \ CD	00	01	11	10
00	0	0	1	0
01	X	X	1	X
11	0	1	1	0
10	0	1	1	0

SOP form for the given function is $f(A, B, C, D) = CD + AD = D(C + A)$

Ans. (b)

22. Refer to the following arrangement of master−slave flip−flops.



It has the initial state of P, Q as 0, 1 (respectively). After the clock cycles, the output state P, Q is (respectively)

(a) 1, 0 (b) 1, 1 (c) 0, 0 (d) 0, 1

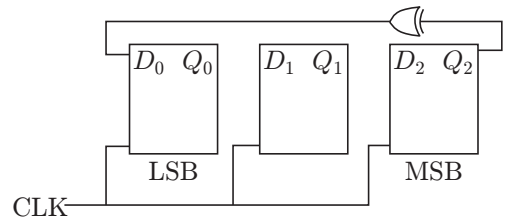
Solution:

	J	K	D		
	1	1	(P)	P	Q
Initially				0	1
	1	1	0	1	0
	1	1	1	0	1

For $J = 1$ and $K = 1$, the output is complement to previous state.

Ans. (a)

23. Consider the circuit given below with initial state $Q_0 = 1$, $Q_1 = Q_2 = 0$. The state of the circuit is given by the value $4Q_2 + 2Q_1 + Q_0$.



Which one of the following is the correct state sequence of the circuit?

(a) 1, 3, 4, 6, 7, 5, 2 (b) 1, 2, 5, 3, 7, 6, 4

(c) 1, 2, 7, 3, 5, 6, 4 (d) 1, 6, 5, 7, 2, 3, 4

Solution:

$D_2(Q_1)$	$D_1(Q_0)$	D_0 $(Q_1 \oplus Q_2)$	Q_2	Q_1	Q_0	$4Q_2 + 2Q_1 + Q_0$
Initially			**0**	**0**	**1**	**1**
0	1	0	0	1	0	2
1	0	1	1	0	1	5
0	1	1	0	1	1	3
1	1	1	1	1	1	7
1	1	0	1	1	0	6
1	0	0	1	0	0	4
0	0	1	0	0	1	1

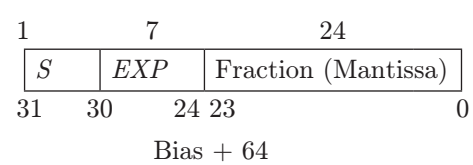
Therefore, the correct state sequence of the circuit is 1, 2, 5, 3, 7, 6, 4. Ans. (b)

24. A 32-bit floating-point number is represented by a 7-bit signed exponent, and a 24-bit fractional mantissa. The base of the scale factor is 16.

(a) The range of the exponent is _____.

(b) The range of the exponent is _____ if the scale factor is represented in excess −64 format.

Solution: The given floating point number format is



The above floating-point architecture is IBM floating-point architecture.

The number is represented as the following formula

$$(-1)^S \times 0.\text{Mantissa} \times 10^{\text{EXP} - 64}$$

(a) The range of the exponent is the range of the 7-bit signed numbers, which is −64 to +63.

(b) The range of the exponent if the scale factor is represented in excess −64 format is 0 to 127.

25. Booth's coding in 8-bits for the decimal number −57 is

(a) 0 − 100 + 1000 (b) 0 − 100 + 100 − 1
(c) 0 − 1 + 100 − 10 + 1 (d) 00 − 10 + 100 − 1

Solution: −57 can be represented in 2's complement form: 1 1 0 0 01 1 1

Booth's coding for decimal number −57 is

1 1 0 0 0 1 1 1$\boxed{0}$

$\Downarrow$

0 −1 0 0 +1 0 0 −1 −1

Ans. (b)

26. The 2' complement representation of the decimal value −15 is

(a) 1111 (b) 11111
(c) 111111 (d) 10001

Solution: +15 = 1 1 1 1
−15 in 2's complement = 10001

Ans. (d)

GATE PREVIOUS YEARS' QUESTIONS

1. Assuming all numbers are in 2's complement representation, which of the following numbers is divisible by 11111011?

(a) 11100111 (b) 11100100
(c) 11010111 (d) 11011011

(GATE 2003: 1 Mark)

Solution:

2's Complement	Binary Number	Decimal Value
11100111	00011001	25
11100100	00011100	28
11010111	00101001	41
11011011	00100101	37

11111011 is 5 in decimal. Only 25 is divisible by 5.

Ans. (a)

2. The following is a scheme for floating point number representation using 16 bits.

Bit Position	15	14.....9	8.....0
	s	e	m
	sign	exponent	mantissa

Let s, e, and m be the numbers represented in binary in the sign, exponent, and mantissa fields, respectively. Then the floating point number represented is

$$\begin{cases} (-1)^s(1+m\times 2^{-9})2^{e-31} & \text{if the exponent} \neq 111111 \\ 0 & \text{otherwise} \end{cases}$$

What is the maximum difference between two successive real numbers representable in this system?

(a) 2^{-40} (b) 2^{-9}
(c) 2^{22} (d) 2^{31}

(GATE 2003: 2 Marks)

Solution:
Largest positive number, m = 11111111
exponent, e = 111110
Second largest positive number = 11111110
exponent, e = 111110
Difference = $2^{31}\,(2 - 2^{-9} - 2 + 2^{-8}) = 2^{22}$

Ans. (c)

3. Consider an array multiplier for multiplying two n bit numbers. If each gate in the circuit has a unit delay, the total delay of the multiplier is

(a) $\Theta(1)$ (b) $\Theta(\log n)$ (c) $\Theta(n)$ (d) $\Theta(n^2)$

(GATE 2003: 1 Mark)

Solution:

$$Delay = 1+2+\cdots+n = \frac{n(n+1)}{2} = \Theta(n^2)$$

Ans.(d)

4. A 1-input, 2-output synchronous sequential circuit behaves as follows:

Let z_k, n_k denote the number of 0's and 1's, respectively, in initial k bits of the input ($z_k + n_k = k$). The circuit outputs 00 until one of the following conditions holds.

(a) $z_k - n_k = 2$. In this case, the output at the kth and all subsequent clock ticks is 10.
(b) $n_k - z_k = 2$. In this case, the output at the kth and all subsequent clock ticks is 01.

What is the minimum number of states required in the state transition graph of the above circuit?

(a) 5 (b) 6 (c) 7 (d) 8

(GATE 2003: 2 Marks)

Solution: Number of transition states are 7.

Ans. (c)

5. The literal count of a Boolean expression is the sum of the number of times each literal appears in the expression. For example, the literal count of $(xy + xz')$ is 4. What are the minimum possible literal counts of the product-of-sum and sum-of-product representations, respectively, of the function given by the following Karnaugh map? Here, X denotes "don't care".

XY \ ZW	00	01	11	10
00	X	1	0	1
01	0	1	X	0
11	1	X	X	0
10	X	0	0	X

(a) (11, 9) (b) (9, 13)
(c) (9, 10) (d) (11, 11)

(GATE 2003: 2 Marks)

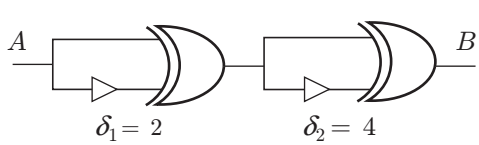
Solution:

XY \ ZW	00	01	11	10
00	X	1	0	1
01	0	1	X	0
11	1	X	X	0
10	X	0	0	X

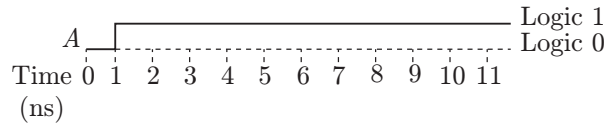
SOP: $wy + w'y' + z'wx' + xyz'$
Literal count: 10
POS: $(y' + z')(w' + z')(z' + y)(x + z + w)$
Literal count: 9

Ans. (c)

6. Consider the following circuit composed of XOR gates and non-inverting buffers.



The non-inverting buffers have delays $\delta_1 = 2$ ns and $\delta_2 = 4$ ns as shown in the figure. Both XOR gates and all wires have zero delay. Assume that all gate inputs, outputs and wires are stable at logic level 0 at time 0. If the following waveform is applied at input A, how many transition(s) (change of logic levels) occur(s) at B during the interval from 0 to 10 ns?



(a) 1 (b) 2
(c) 3 (d) 4

(GATE 2003: 2 Marks)

Solution: Total number of transitions is 3.
1: 0 to 0
2: 0 to 1
3: 1 to 1

Ans. (c)

7. A Boolean function $x'y' + xy + x'y$ is equivalent to

(a) $x' + y'$ (b) $x + y$
(c) $x + y'$ (d) $x' + y'$

(GATE 2004: 1 Mark)

Solution:
$x'y' + xy + x'y$
$= x'(y' + y) + xy$ (as $y' + y = 1$)
$= x' + xy$ (as $x + x'y = x + y$)
$= x' + y$

Ans. (d)

8. In an SR latch made by cross-coupling two NAND gates, if both S and R inputs are set to 0, then it will result in

(a) $Q = 0,\ Q' = 1$ (b) $Q = 1,\ Q' = 0$
(c) $Q = 1,\ Q' = 1$ (d) Indeterminate states

(GATE 2004: 1 Mark)

Solution: If both S and R are 0, using SR latch with NAND gate output state will be invalid state.

Ans. (d)

9. If 73_x (in base-x number system) is equal to 54_y (in base-y number system), the possible values of x and y are

(a) 8, 16 (b) 10, 12
(c) 9, 13 (d) 8, 11

(GATE 2004: 1 Mark)

Solution: By doing hit and trial with the options, we arrive at
If $x = 8$, $(73)_8 = 7 \times 8^1 + 3 \times 8^0 = 59$
If $y = 11$, $(54)_{11} = 5 \times 11^1 + 4 \times 11^0 = 59$

Ans. (d)

10. What is the result of evaluating the following two expressions using three-digit floating point arithmetic with rounding?

$$(113 + -111) + 7.51$$
$$113 + (-111 + 7.51)$$

(a) 9.51 and 10.0, respectively
(b) 10.0 and 9.51, respectively
(c) 9.51 and 9.51, respectively
(d) 10.0 and 10.0, respectively

(GATE 2004: 1 Mark)

Solution:
$(113 + -111) + 7.51 = 9.51$
$113 + (-111 + 7.51) = 10.0$

Ans. (a)

11. Consider a multiplexer with X and Y as data inputs and Z as control input. $Z = 0$ selects input X, and $Z = 1$ selects input Y. What are the connections required to realize the 2-variable Boolean function $f = T + R$, without using any additional hardware?

(a) R to X, 1 to Y, T to Z
(b) T to X, R to Y, T to Z
(c) T to X, R to Y, 0 to Z
(d) R to X, 0 to Y, T to Z

(GATE 2004: 2 Marks)

Solution: Connect R to X, 1 to Y and T to Z.

Ans. (a)

12. A circuit outputs a digit in the form of 4 bits, where 0 is represented by 0000, 1 by 0001, …, 9 by 1001. A combinational circuit is to be designed which takes these 4 bits as input and output 1 if the digit ≥5, and 0 otherwise. If only AND, OR and NOT gates are used, what is the minimum number of gates required?

(a) 2 (b) 3
(c) 4 (d) 5

(GATE 2004: 2 Marks)

Solution:

a	*B*	*c*	*d*	Output
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	0	1	0
0	1	0	0	0
0	1	1	1	1

(*Continued*)

Continued

A	*B*	*c*	*d*	Output
0	1	1	0	1
0	1	0	1	1
1	0	0	0	1
1	0	1	1	1

Now using K-map, the expression becomes $a + bd + bc \rightarrow a + b(d+c)$.
Hence, three gates will be required.

Ans. (b)

13. Which are the essential prime implicants of the following Boolean function?

$$f(a, b, c) = a'c + ac' + b'c$$

(a) $a'c$ and ac' (b) $a'c$ and $b'c$
(c) $a'c$ only (d) ac' and bc'

(GATE 2004: 2 Marks)

Solution:
$$\begin{aligned} f(a, b, c) &= a'c + ac' + b'c \\ &= a'(b + b')c + a(b + b')c' + (a + a')b'c \\ &= a'bc + a'b'c + abc' + ab'c' + a'b'c \end{aligned}$$

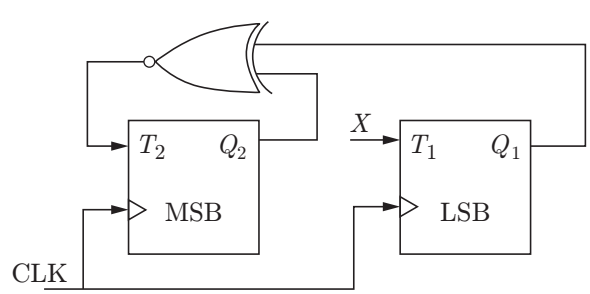
Using K-map:

c \ ab	00	01	11	10
0			1	1
1	1	1		

here ca' and $c'a$ are essential prime implicants, we cannot remove them.

Ans. (a)

14. Consider the partial implementation of a 2-bit counter using T flip−flops following the sequence 0-2-3-1-0, as shown in the following figure.



To complete the circuit, the input X should be

(a) Q_2' (b) $Q_2 + Q_1$
(c) $(Q_1 \oplus Q_2)'$ (d) $Q_1 \oplus Q_2$

(GATE 2004: 2 Marks)

Solution: Sequence 0 − 2 − 3 − 1 − 0 means 00 − 11 − 01 − 00
If $x = Q_1 \oplus Q_2$, then the given output will be produced.

Ans. (d)

15. A 4-bit carry look-ahead adder, which adds two 4-bit numbers, is designed using AND, OR, NOT, NAND, NOR gates only. Assuming that all the inputs are available in both complemented and uncomplemented forms and the delay of each gate is one time unit, what is the overall propagation delay of the adder? Assume that the carry network has been implemented using two-level AND−OR logic.

(a) 4 time units (b) 6 time units
(c) 10 time units (d) 12 time units

(GATE 2004: 2 Marks)

Solution: Equations for carry look-ahead adder are as follows:

$$C_1 = G_0 + P_0 \cdot C_0$$
$$C_2 = G_1 + P_1 \cdot C_1$$
$$C_3 = G_2 + P_2 \cdot C_2$$
$$C_4 = G_3 + P_3 \cdot C_3$$

The total number of gate delay will be 6.

Ans. (b)

16. Let A = 1111 1010 and B = 0000 1010 be two 8-bit 2's complement numbers. Their product in 2's complement is

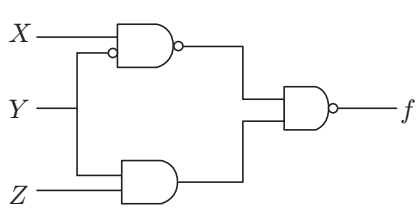
(a) 1100 0100 (b) 1001 1100
(c) 1010 0101 (d) 1101 0101

(GATE 2004: 2 Marks)

Solution:
A = 1111 1010 = 0000 0110 = −6
B = 0000 1010 = 10
$A \times B = -60$
−60 = 11000100

Ans. (a)

17. Consider the following circuit.



Which one of the following is TRUE?

(a) f is independent of X.
(b) f is independent of Y.
(c) f is independent of Z.
(d) None of X, Y, Z is redundant.

(GATE 2005: 1 Mark)

Solution:

$$f = XY' + (YZ)'$$
$$= Y' + Z'$$

So, f is independent of X.

Ans. (a)

18. The range of integers that can be represented by an n-bit 2's complement number system is

(a) -2^{n-1} to $(2^{n-1}-1)$
(b) $-2(2^{n-1}-1)$ to $(2^{n-1}-1)$
(c) -2^{n-1} to 2^{n-1}
(d) $-2(2^{n-1}+1)$ to $(2^{n-1}-1)$

(GATE 2005: 1 Mark)

Solution: Negative numbers are one more than positive number. So, the range of 2's complement is 2^{n-1} to $(2^{n-1} - 1)$.

Ans. (a)

19. The hexadecimal representation of 657_8 is

(a) 1AF (b) D78
(c) D71 (d) 32F

(GATE 2005: 1 Mark)

Solution:
$(657)_8 = (110\ 101\ 111)_8$
$(0001\ 1010\ 1111)_{16} = (1AF)_{16}$

Ans. (a)

20. The switching expression corresponding to $f(A, B, C, D) = \sum(1,4,5,9,11,12)$is

(a) $BC'D' + A'C'D + AB'D$
(b) $ABC' + ACD + B'C'D$
(c) $ACD' + A'BC' + AC'D'$
(d) $A'BD + ACD' + BCD'$

(GATE 2005: 1 Mark)

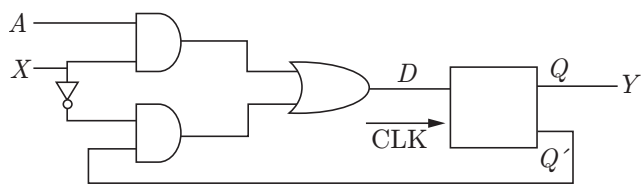
Solution:

$AB \backslash CD$	00	01	11	10
00		1		
01	1	1		
11	1			
10		1	1	

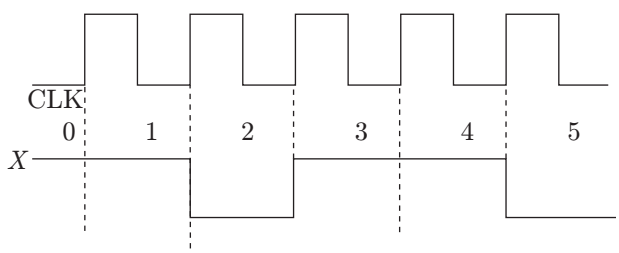
$f = BC'D' + A'C'D + AB'D$

Ans. (a)

21. Consider the following circuit involving a positive-edge triggered D FF.



Consider the following timing diagram. Let A_i represent the logic level on the line A in the ith clock period.



Let A' represent the complement of A. The correct output sequence on Y over the clock periods 1 through 5 is

(a) $A_0A_1A_1'A_3A_4$ (b) $A_0A_1A_2'A_3A_4$

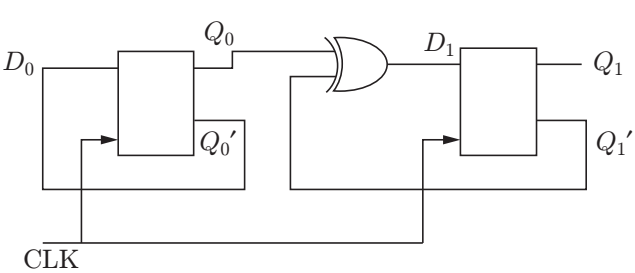
(c) $A_1A_2A_2'A_3A_4$ (d) $A_1A_2'A_3A_4A_5'$

(GATE 2005: 2 Marks)

Solution: The correct output sequence on Y is $A_0A_1A_1'A_3A_4$.

Ans. (a)

22. Consider the following circuit.



The flip−flops are positive-edge triggered D FFs. Each state is designated as a 2-bit string Q_0Q_1. Let the initial state be 00. The state transition sequence is

(a) 00-11-01 (b) 00-11

(c) 00-10-01-11 (d) 00-11-01-10

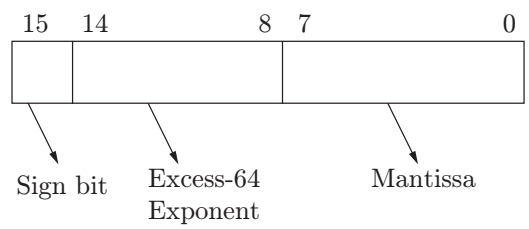
(GATE 2005: 2 Marks)

Solution:

D_0	D_1	Q_0	Q_1
1	1	1	1
0	1	0	1
1	0	1	0
0	0	0	0

Ans. (d)

Linked Answer Questions 23 and 24: Consider the following floating-point format.



Mantissa is a pure fraction in sign-magnitude form.

23. The decimal number 0.239×2^{13} has the following hexadecimal representation (without normalization and rounding off):

(a) 0D 24 (b) 0D 4D

(c) 4D 0D (d) 4D 3D

(GATE 2005: 2 Marks)

Solution:
$0.239 \times 2^{13} = 0.00111101 \times 2^{13}$
$64 + 13 = 77$

0	100 1101	0011 1101
	4D	3D

Ans. (d)

24. The normalized representation for the above format is specified as follows. The mantissa has an implicit 1 preceding the binary (radix) point. Assume that only 0's are padded in while shifting a field. The normalized representation of the above number (0.239×2^{13}) is

(a) 0A 20 (b) 11 34

(c) 49 D0 (d) 4A E8

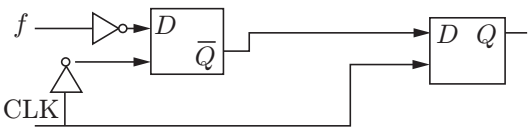
(GATE 2005: 2 Marks)

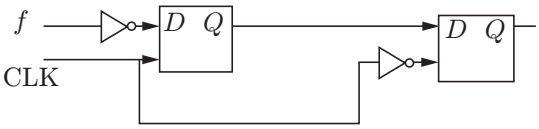
Solution: Using normalized form
1.11101×2^{10}
$64 + 10 = 74$

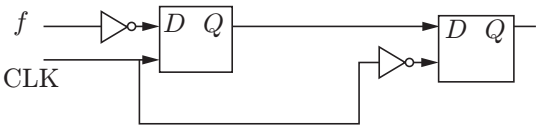
0	100 1010	11101000
	4A	E8

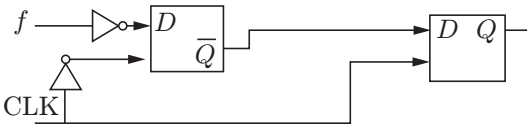
Ans. (d)

25. You are given a free running clock with a duty cycle of 50% and a digital waveform of which changes only at the negative edge of the clock. Which one of the following circuits (using clocked D flip−flops) will delay the phase off by 180°?

(a)


(b)


(c)


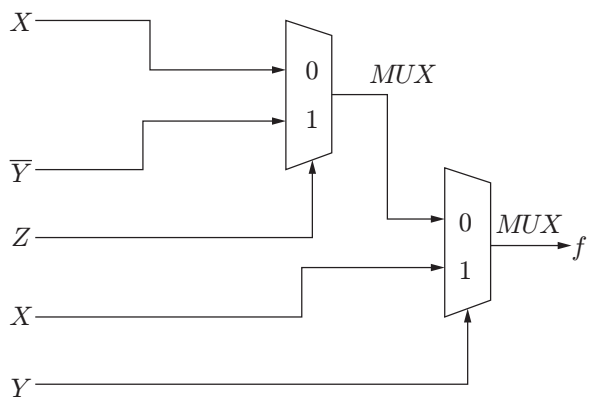
(d)


(GATE 2006: 1 Mark)

Solution: Left flip−flop is activated, the clock for right will be 0. There will be delay of 180°.

Ans. (c)

26. Consider the circuit shown. Which one of the following options correctly represents $f(x, y, z)$?



(a) $x\overline{z} + xy + \overline{y}z$ (b) $x\overline{z} + xy + \overline{yz}$

(c) $xz + xy + yz$ (d) $xz + x\overline{y} + \overline{y}z$

(GATE 2006: 1 Mark)

Solution:

$$f_1 = \overline{z}x + z\overline{y}$$

$$f = \overline{y}(\overline{z}x + z\overline{y}) + yx$$

$$= \overline{yz}x + \overline{y}zy + yx = yx + \overline{yz}x + z\overline{y}$$

$$= yx + \overline{y}(z + \overline{z}x) = yx + \overline{y}(z + x)$$

$$= yx + \overline{y}z + \overline{y}x$$

Now use K-map, we have

X \ YZ	00	01	11	10
0		1		
1	1	1	1	1

Hence, $xy + \overline{y}z + x\overline{z}$ is the correct option.

Ans. (a)

27. Given two 3-bit numbers $a_2a_1a_0$ and $b_2b_1b_0$ and c, the carry in the function that represents the carry generate function when these two numbers are added is

(a) $a_2b_2 + a_2a_1b_1 + a_2a_1a_0b_0 + a_2a_0b_1b_0$
$+ a_1b_2b_1 + a_1a_0b_2b_0 + a_0b_2b_1b_0$

(b) $a_2b_2 + a_2b_1b_0 + a_2a_1b_1b_0 + a_1a_0b_2b_1$
$+ a_1a_0b_2 + a_1a_0b_2b_0 + a_2a_0b_1b_0$

(c) $a_2 + b_2 + (a_2 \oplus b_2)\ (a_1 + b_1 + (a_1 \oplus b_1)(a_0 + b_0))$

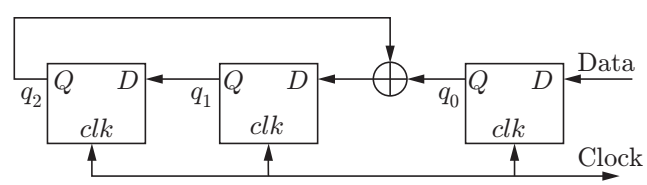
(d) $a_2b_2 + \overline{a}_2a_1b_1 + \overline{a}_2\overline{a}_1a_0b_0 + \overline{a}_2a_0\overline{b}_1b_0$
$+ a_1\overline{b}_2b_1 + \overline{a}_1a_0\overline{b}_2b_0 + a_0\overline{b}_2\overline{b}_1b_0$

(GATE 2006: 1 Mark)

Solution: Option (d) generates carry c always.

Ans. (d)

28. Consider the circuit in the diagram. The $\oplus$ operator represents XOR. The D flip−flops are initialized to zeroes (cleared).



The following data 100110000 is supplied to the "data" terminal in nine clock cycles. After that the values of $q_2q_1q_0$ are

(a) 000 (b) 001

(c) 010 (d) 101

(GATE 2006, 1 Mark)

Solution:

Clock	Input	q_2	q_1	q_0
0	1	0	0	1
1	0	0	1	0
2	0	1	0	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	0
6	0	0	1	0
7	0	1	0	0
8	0	**0**	**1**	**0**

Ans. (c)

29. Consider a Boolean function $f(w, x, y, z)$. Suppose that exactly one of its inputs is allowed to change at a time. If the function happens to be true for two input vectors $i_1 = (w_1, x_1, y_1, z_1)$ and $i_2 = (w_2, x_2, y_2, z_2)$, we would like the function to remain true as the input changes from i_1 to i_2 (i_1 and i_2 differ in exactly 1 bit position), without becoming false momentarily. Let $f(w, x, y, z) = (5, 7, 11, 12, 13, 15)$. Which of the following cube covers of f will ensure that the required property is satisfied?

(a) $w'xz$, wxy', $xy'z$, xyz, wyz
(b) wxy,$w'xz$, wyz
(c) $wxyz$, xz, $wx'yz$
(d) wzy, wyz, wxz,$w'xz$, $xy'z$, xyz

(GATE 2006: 1 Mark)

Solution:

$yz \backslash wx$	00	01	11	10
00				
01		1	1	
11	1	1	1	
10			1	

From the K-map, we have $w'xz$, wxy', $xy'z$, xyz, wyz.

Ans. (a)

30. We consider the addition of two 2's complement numbers $b_{n-1}b_{n-2}\ldots b_0$ and $a_{n-1}a_{n-2}\ldots a_0$. A binary adder for adding unsigned binary numbers is used to add the two numbers. The sum is denoted by $c_{n-1}c_{n-2}\ldots c_0$ and the carry-out by c_{out}.

Which one of the following options correctly identifies the overflow condition?

(a) $\overline{c_{\text{out}}(a_{n-1} \oplus b_{n-1})}$
(b) $a_{n-1}b_{n-1}\overline{c_{n-1}} + \overline{a_{n-1}b_{n-1}c_{n-1}}$
(c) $c_{\text{out}} \oplus c_{n-1}$
(d) $a_{n-1} \oplus b_{n-1} \oplus c_{n-1}$

(GATE 2006: 1 Mark)

Solution: The overflow will be identified c_{out} and c_{n-1} are XORed.

Ans. (c)

31. Consider numbers represented in 4-bit gray code. Let $h_3h_2h_1h_0$ be the gray code representation of a number n and let $g_3g_2g_1g_0$ be the gray code of $(n+1)$ (modulo 16) value of the number. Which one of the following functions is correct?

(a) $g_0(h_3h_2h_1h_0) = \sum(1, 2, 3, 6, 10, 13, 14, 15)$
(b) $g_1(h_3h_2h_1h_0) = \sum(4, 9, 10, 11, 12, 13, 14, 15)$
(c) $g_2(h_3h_2h_1h_0) = \sum(2, 4, 5, 6, 7, 12, 13, 15)$
(d) $g_3(h_3h_2h_1h_0) = \sum(0, 1, 6, 7, 10, 11, 12, 13)$

(GATE 2006: 1 Mark)

Solution: $(n + 1)$ mod 16 is a 5-bit number; range is from 0 to 31.

$$g_2(2,4,5,6,7,12,13,15)$$

Ans. (c)

32. What is the maximum number of different Boolean functions involving n Boolean variables?

(a) n^2
(b) 2^n
(c) 2^{2^n}
(d) 2^{n^2}

(GATE 2007: 1 Mark)

Solution: Maximum Boolean functions with n elements can be 2^{2^n}.

Ans. (c)

33. How many 3-to-8 line decoders with an enable input are needed to construct a 6-to-64 line decoder without using any other logic gates?

(a) 7
(b) 8
(c) 9
(d) 10

(GATE 2007: 1 Mark)

Solution: We need to construct 6 to 64 decoder so apply the following:

$$\sum_{i=1}^{\log 64} \frac{64}{8^i} = \frac{64}{8} + \frac{64}{64} = 9$$

Ans. (c)

34. Consider the following Boolean function of four variables:

$$f(w,x,y,z) = \sum (1,3, 4, 6, 9, 11,12,14)$$

The function is

(a) independent of one variable
(b) independent of two variables
(c) independent of three variables
(d) dependent on all the variables

(GATE 2007: 1 Mark)

Solution: $f(w,x,y,z) = \sum(1, 3, 4, 6, 9, 11, 12, 14)$ can be represented in K-map as

$wx \backslash yz$	00	01	11	10
00		1	1	
01	1			1
11	1			1
10		1	1	

Quadruple is formed, indicating the function is independent of two variables.

Ans. (b)

35. Let $f(w, x, y, z) = \sum (0, 4, 5, 7, 8, 9, 13, 15)$. Which of the following expressions are NOT equivalent to f?

(P) $x'y'z' + w'xy' + wy'z + xz$
(Q) $w'y'z' + wx'y' + xz$
(R) $w'y'z' + wx'y' + xyz + xy'z$
(S) $x'y'z' + wx'y' + w'y$

(a) P only (b) Q and S
(c) R and S (d) S only

(GATE 2007: 2 Marks)

Solution:

$wx \backslash yz$	00	01	11	10
00	1			
01	1	1	1	
11		1	1	
10	1	1		

Option (d) does not contain essential prime implicants.

Ans. (d)

36. Define the connective * for the Boolean variables X and Y as $X^*Y = XY + X'Y'$. Let $Z = X^*Y$. Consider the following expressions P, Q and R.

P: $X = Y^*Z$ Q: $Y = X^*Z$ R: $X^*Y^*Z = 1$

Which of the following is TRUE?

(a) Only P and Q are valid.
(b) Only Q and R are valid.
(c) Only P and R are valid.
(d) All P, Q, R are valid.

(GATE 2007: 2 Marks)

Solution:
P: $Y^*Z = YZ + Y'Z' = Y(XY + X'Y') + Y'(X'Y + XY') = X$
So, P is valid.
Q: $Y = X^*Z$
$X^*Z = XZ + X'Z' = X(XY + X'Y') + X'(X'Y + XY') = Y$
So, Q is valid.
R: $X^*Y^*Z = 1$
$X^*Y^*Z = Z^*Z = ZZ + Z'Z' = 1$
So, R is also valid.

Ans. (d)

37. Suppose only one multiplexer and one inverter are allowed to be used to implement any Boolean function of n variables. What is the minimum size of the multiplexer needed?

(a) 2^n line to 1 line (b) 2^{n+1} line to 1 line
(c) 2^{n-1} line to 1 line (d) 2^{n-2} line to 1 line

(GATE 2007: 2 Marks)

Solution: Since only one inverter is allowed, so $n-1$ variable will be used for selection and one as input. 2^{n-1} line to 1 line multiplexer will be required.

Ans. (c)

38. In a look-ahead carry generator, the carry generate function G_i and the carry propagate function P_i for inputs A_i and B_i are given by

$$P_i = A_i \oplus B_i \text{ and } G_i = A_i B_i$$

The expressions for the sum bit S_i and the carry bit C_{i+1} of the look-ahead carry adder are given by $S_i = P_i \oplus C_i$ and $C_{i+1} = G_i + P_i C_i$, where C_0 is the input carry.

Consider a two-level logic implementation of the look-ahead carry generator. Assume that all P_i and G_i are available for the carry generator circuit and that the AND and OR gates can have any number of inputs. The number of AND gates and OR gates needed to implement the look-ahead carry generator for a 4-bit adder with S_3, S_2S_1, S_0, and C_4 as its outputs are, respectively,

(a) 6, 3 (b) 10, 4
(c) 6, 4 (d) 10, 5

(GATE 2007: 2 Marks)

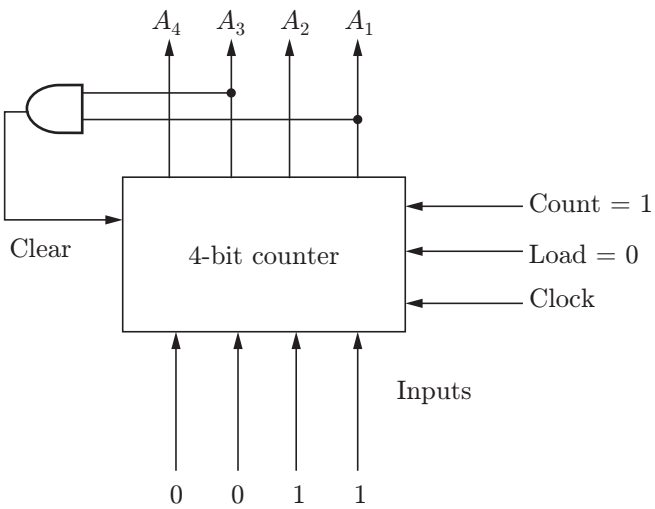
Solution: Two 1-bit adders are required, where each 1-bit adder required 5 AND and 2 OR gates. So, 10 AND gates and 4 OR gates are required.

Ans. (b)

39. The control signal functions of a 4-bit binary counter are given below (where X is "don't care"):

Clear	Clock	Load	Count	Function
1	X	X	X	Clear to 0
0	X	0	0	No change
0		1	X	Load input
0		0	1	Count next

The counter is connected as follows:



Assume that the counter and gate delays are negligible. If the counter starts at 0, then it cycles through the following sequence:

(a) 0, 3, 4 (b) 0, 3, 4, 5
(c) 0, 1, 2, 3, 4 (d) 0, 1, 2, 3, 4, 5

(GATE 2007: 2 Marks)

Solution: The counter will count if count = 1 and load = 0, the sequence will be 0, 1, 2, 3, 4.

Ans. (c)

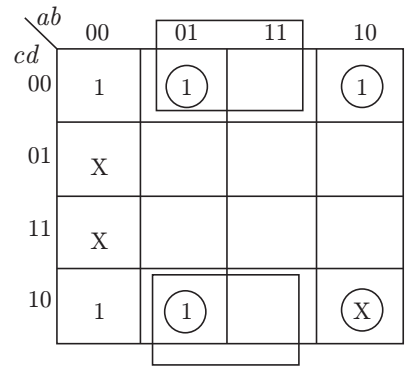
40. In the Karnaugh map shown below, X denotes a don't care term. What is the minimal form of the function represented by the Karnaugh map?

cd \ ab	00	01	11	10
00	1	1		1
01	X			
11	X			
10	1	1		X

(a) $\overline{b}\cdot\overline{d}+\overline{a}\cdot\overline{d}$ (b) $\overline{a}\cdot\overline{b}+\overline{b}\cdot\overline{d}+\overline{a}b\cdot\overline{d}$
(c) $\overline{b}\cdot\overline{d}+\overline{a}\cdot b\cdot\overline{d}$ (d) $\overline{a}\cdot\overline{b}+\overline{b}\cdot\overline{d}+\overline{a}\cdot\overline{d}$

(GATE 2008: 1 Mark)

Solution:



So, $\overline{b}\cdot\overline{d}+\overline{a}\cdot\overline{d}$

Ans. (a)

41. In the IEEE floating-point representation, the hexadecimal value 0x00000000 corresponds to

(a) The normalized value 2^{-127}
(b) The normalized value 2^{-126}
(c) The normalized value +0
(d) The special value +0

(GATE 2008: 1 Mark)

Solution: IEEE floating point standard represent number in normalized form. This is the smallest positive number, that is, +0.

Ans. (d)

42. Let r denote number system radix. The only value(s) of r that satisfy the equation $\sqrt{121_r}=11_r$ is/are

(a) Decimal 10 (b) Decimal 11
(c) Decimal 10 and 11 (d) Any value > 2

(GATE 2008: 1 Mark)

Solution:

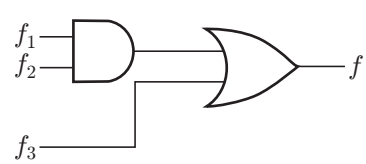
$$(1+2r^2+r^3)^{1/2}=1+r$$

$$\Rightarrow 1+2r^2+r^3=(1+r)^2=1+r^2+2r$$

$$r^3+r^2-2r=0\Rightarrow r>2$$

Ans. (d)

43. Given f_1, f_3 and f in canonical sum of products form (in decimal) for the circuit



$f_1=\sum m(4, 5, 6, 7, 8)$
$f_3=\sum m(1, 6, 15)$
$f=\sum m(1, 6, 8, 15)$

then f_2 is

(a) $\sum m(4, 6)$ (b) $\sum m(4, 8)$
(c) $\sum m(6, 8)$ (d) $\sum m(4, 6, 8)$

(GATE 2008: 1 Mark)

Solution: To get $f = \sum m(1, 6, 8, 15)$, f_2 should set 6 and 8.

Ans. (c)

44. If P, Q, R are Boolean variables, then

$$(P+\bar{Q})(P\cdot\bar{Q}+P\cdot R)(\bar{P}\cdot\bar{R}+\bar{Q})$$

simplifies to

(a) $P\cdot\bar{Q}$ (b) $P\cdot\bar{R}$
(c) $P\cdot\bar{Q}+R$ (d) $P\cdot\bar{R}+Q$

(GATE 2008: 2 Marks)

Solution:

$$\begin{aligned}(P+\bar{Q})(P\cdot\bar{Q}+P\cdot R)(\bar{P}\cdot\bar{R}+\bar{Q})\\ &= (P\cdot\bar{Q}+P\cdot\bar{Q}+P\cdot\bar{Q}\cdot R+PR)(\bar{P}\cdot\bar{R}+\bar{Q})\\ &= P\cdot\bar{Q}+P\cdot\bar{Q}+P\cdot\bar{Q}\cdot R+P\cdot\bar{Q}\cdot R\\ &= P\cdot\bar{Q} \quad (\text{as } 1+R=1)\end{aligned}$$

Ans. (a)

45. $(1217)_8$ is equivalent to

(a) $(1217)_{16}$ (b) $(028\text{F})_{16}$
(c) $(2297)_{10}$ (d) $(0\text{B}17)_{16}$

(GATE 2009: 1 Mark)

Solution:
$(1217)_8 = (001\ 010\ 001\ 111)_8$
$(0010\ 1000\ 1111) = (28\text{F})_{16} = (028\text{F})_{16}$

Ans. (b)

46. What is the minimum number of gates required to implement the Boolean function $(AB + C)$ if we have to use only 2-input NOR gates?

(a) 2 (b) 3
(c) 4 (d) 5

(GATE 2009: 1 Mark)

Solution:
$(AB + C) = (A + C)(B + C)$
$= ((A + C)' + (B + C)')'$
Thus, it requires three NOR gates.

Ans. (b)

47. The binary operation $\square$ is defined as follows

P	Q	$P \square Q$
T	T	T
T	F	T
F	T	F
F	F	T

Which one of the following is equivalent to $P \vee Q$?

(a) $\neg Q \square \neg P$ (b) $P \square \neg Q$
(c) $\neg P \square Q$ (d) $\neg P \square \neg Q$

(GATE 2009: 1 Mark)

Solution: Try using different values of P and Q. For example, P = True and Q = False.

Ans. (b)

48. How many 32 K $\times$ 1 RAM chips are needed to provide a memory capacity of 256 K-bytes?

(a) 8 (b) 32
(c) 64 (d) 128

(GATE 2009: 1 Mark)

Solution:

$$\text{Number of chips required} = \frac{256\times 8}{32\times 1} = 64$$

Ans. (c)

49. The minterm expansion of $f(P,Q,R) = PQ + Q\bar{R} + P\bar{R}$ is

(a) $m_2 + m_4 + m_6 + m_7$ (b) $m_0 + m_1 + m_3 + m_5$
(c) $m_0 + m_1 + m_6 + m_7$ (d) $m_2 + m_3 + m_4 + m_5$

(GATE 2010: 1 Mark)

Solution:

$$\begin{aligned}f(P,Q,R) &= PQ + QR' + PR'\\ &= PQ(R+R') + (P+P')QR' + P(Q+Q')R'\\ &= PQR + PQR' + P'QR' + PQ'R'\\ &= m_7 + m_6 + m_2 + m_4\end{aligned}$$

Ans. (a)

50. P is a 16-bit signed integer. The 2's complement representation of P is $(\text{F87B})_{16}$. The 2's complement representation of $8\times P$ is

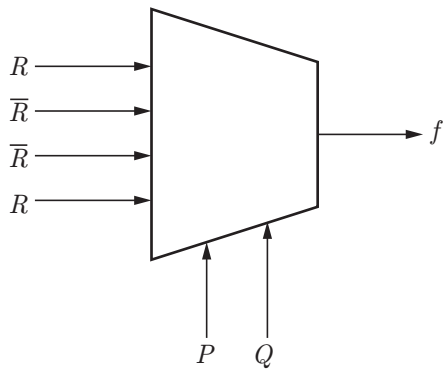
(a) $(\text{C3D8})_{16}$ (b) $(187\text{B})_{16}$
(c) $(\text{F878})_{16}$ (d) $(987\text{B})_{16}$

(GATE 2010: 1 Mark)

Solution:
$(\text{F87B})_{16} = (1925)_2$
$1925 \times 8 = (15400)_2$ = 2's complement of $(15400)_2$
$= (\text{C3D8})_{16}$

Ans. (a)

51. The Boolean expression for the output f of the multiplexer shown below is



(a) $\overline{P \oplus Q \oplus R}$ (b) $P \oplus Q \oplus R$

(c) $\overline{P \oplus Q \oplus R}$ (d) $P + Q + R$

(GATE 2010: 1 Mark)

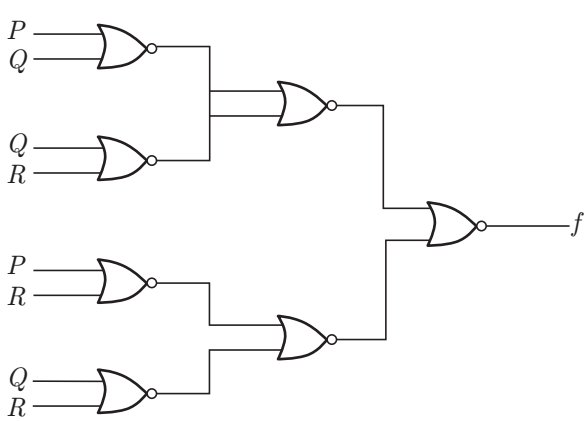
Solution:

P	Q	Input	Output
0	0	R	$P'Q'R$
0	1	R'	$P'QR'$
1	0	R'	$PQ'R'$
1	1	R	PQR

$P'Q'R + P'QR' + PQ'R' + PQR = P \oplus Q \oplus R$

Ans. (b)

52. What is the Boolean expression for the output f of the combinational logic circuit of NOR gates given below?



(a) $\overline{Q+R}$ (b) $\overline{P+Q}$

(c) $\overline{P+R}$ (d) $\overline{P+Q+R}$

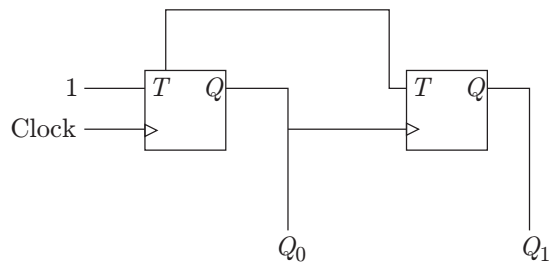
(GATE 2010: 2 Marks)

Solution:

$$\overline{\overline{P+Q}+\overline{Q+R}}+\overline{\overline{P+R}+\overline{Q+R}}$$
$$= (\overline{P+Q}+\overline{Q+R})\cdot(\overline{P+R}+\overline{Q+R})$$
$$= (\overline{P}\,\overline{Q}+\overline{Q}\,\overline{R})(\overline{P}\,\overline{R}+\overline{Q}\,\overline{R})$$
$$= \overline{P}\,\overline{Q}\,\overline{P}\,\overline{R}+\overline{P}\,\overline{Q}\,\overline{Q}\,\overline{R}+\overline{Q}\,\overline{R}\,\overline{P}\,\overline{R}+\overline{Q}\,\overline{R}\,\overline{Q}\,\overline{R}$$
$$= \overline{P}\,\overline{Q}\,\overline{R}+\overline{P}\,\overline{Q}\,\overline{R}+\overline{P}\,\overline{Q}\,\overline{R}+\overline{Q}\,\overline{R}$$
$$= \overline{P}\,\overline{Q}\,\overline{R}+\overline{Q}\,\overline{R}$$
$$= \overline{Q}\,\overline{R}(\overline{P}+1) = \overline{Q}\,\overline{R} = \overline{Q+R}$$

Ans. (a)

53. In the sequential circuit shown below, if the initial value of the output Q_1Q_0 is 00, what are the next four values of Q_1Q_0?



(a) 11,10,01,00 (b) 10,11,01,00

(c) 10,00,01,11 (d) 11,10,00,01

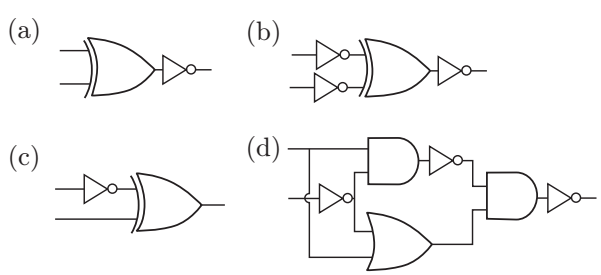
(GATE 2010: 2 Marks)

Solution:

Clock	Q_1	Q_0
1	0	0
2	1	1
3	1	0
4	0	1
5	0	0

Ans. (a)

54. Which one of the following circuits is NOT equivalent to a 2-input XNOR (exclusive NOR) gate?



(GATE 2011: 1 Mark)

Solution: Option (d) does not show XNOR gate functionality.

Ans. (d)

55. The simplified SOP (Sum of Product) form of the Boolean expression

$(P+\bar{Q}+\bar{R})\cdot(P+\bar{Q}+\bar{R})\cdot(P+Q+\bar{R})$ is

(a) $(\bar{P}\cdot Q+\bar{R})$ (b) $(P+\bar{Q}\cdot\bar{R})$

(c) $(\bar{P}\cdot Q+R)$ (d) $(P\cdot Q+R)$

(GATE 2011: 1 Mark)

Solution:

$P\backslash QR$	00	01	11	10
0	0	1	1	1
1	0	0	0	0

which implies $(P+\bar{R})(P+\bar{Q})$

Ans. (b)

56. The minimum number of D flip−flops needed to design a Mod-258 counter is

(a) 9 (b) 8

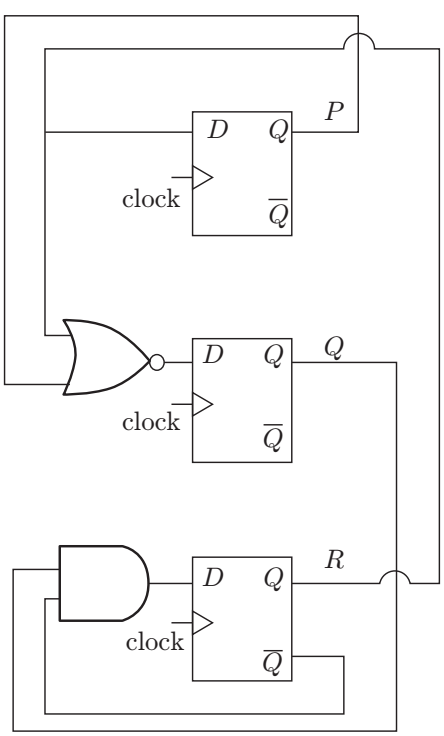
(c) 512 (d) 258

(GATE 2011: 1 Mark)

Solution: $2^n \geq 258 = 9$ K flip−flops

Ans. (a)

Common Data Questions 57 and 58: Consider the following circuit involving three D-type flip−flops used in a certain type of counter configuration.



57. If at some instance prior to the occurrence of the clock edge, P, Q and R have a value 0, 1 and 0, respectively, what shall be the value of PQR after the clock edge?

(a) 000 (b) 001

(c) 010 (d) 011

(GATE 2011: 2 Marks)

Solution:

Clock	$D_1 = R$	$D_2 = (P+R)'$	$D_3 = QR'$	P	Q	R
1	0	1	0	0	1	0
2	0	1	1	0	1	1
3	1	0	0	1	0	0
4	0	0	0	0	0	0

After 010, value of PQR is 011.

Ans. (d)

58. If all the flip−flops were reset to 0 at power on, what is the total number of distinct outputs (states) represented by PQR generated by the counter?

(a) 3 (b) 4

(c) 5 (d) 6

(GATE 2011: 2 Marks)

Solution: Clock cycles can be seen from the table given in Question 57. The number of district outputs is 4.

Ans. (b)

59. The truth table

X	Y	$f(X,Y)$
0	0	0
0	1	0
1	0	1
1	1	1

represents the Boolean function

(a) X (b) $X+Y$

(c) $X\oplus Y$ (d) Y

(GATE 2012: 1 Mark)

Solution: From given truth table, equation formed is

$$XY' + XY \Rightarrow X(Y' + Y) \Rightarrow X$$

Ans. (a)

60. The decimal value 0.5 in IEEE single-precisionfloating-point representation has

(a) Fraction bits of 000…000 and exponent value of 0

(b) Fraction bits of 000…000 and exponent value of −1

(c) Fraction bits of 100…000 and exponent value of 0
(d) No exact representation

(GATE 2012: 1 Mark)

Solution: Binary representation of 0.5 is 0.1000, which can be written as 1.000×2^{-1}
Here, exponent $= -1$ and fraction bits are 000…000.

Ans. (b)

61. What is the minimal form of the Karnaugh map shown below? Assume that **X** denotes a don't care term.

$cd \backslash ab$	00	01	11	10
00	1	X	X	1
01	X			1
11				
10	1			X

(a) $\overline{b}\,\overline{d}$
(b) $\overline{b}\,\overline{d} + \overline{b}\overline{c}$
(c) $\overline{b}\,\overline{d} + a\overline{b}\overline{c}d$
(d) $\overline{b}\,\overline{d} + \overline{b}\overline{c} + \overline{c}\overline{d}$

(GATE 2012: 2 Marks)

Solution:

1	X	X	1
X			1
1			X

From above grouping, the equation obtained is:

$$\overline{c}\overline{b} + \overline{d}\,\overline{b}$$

Ans. (b)

62. The smallest integer that can be represented by an 8-bit number in 2's complement form is

(a) -256
(b) -128
(c) -127
(d) 0

(GATE 2013: 1 Mark)

Solution: Smallest integer for 2's complement is -2^{n-1}
So here $n = 8$, $-2^{8-1} = -128$

Ans. (b)

63. In the following truth table, $V = 1$ if and only if the input is valid.

Inputs				Outputs		
D_0	D_1	D_2	D_3	X_0	X_1	V
0	0	0	0	x	x	0
1	0	0	0	0	0	1
x	1	0	0	0	1	1
x	x	1	0	1	0	1
x	x	x	1	1	1	1

What function does the truth table represent?

(a) Priority encoder
(b) Decoder
(c) Multiplexer
(d) Demultiplexer

(GATE 2013: 1 Mark)

Solution: From the given truth table it is clear that it is 4:2 priority encoder.

Ans. (a)

64. Which one of the following expressions does NOT represent exclusive NOR of x and y?

(a) $xy + x'y'$
(b) $x \oplus y'$
(c) $x' \oplus y$
(d) $x' \oplus y'$

(GATE 2013: 1 Mark)

Solution: $x' \oplus y' = (x'y + y'x) = x \oplus y$, this represents XOR. So, option (d) is not exclusive NOR.

Ans. (d)

65. A RAM chip has a capacity of 1024 words of 8 bits each (1K × 8). The number of 2 × 4 decoders with enable line needed to construct a 16K × 16 RAM from 1K × 8 RAM is

(a) 4
(b) 5
(c) 6
(d) 7

(GATE 2013: 2 Marks)

Solution: Number of chips required $= \dfrac{16 \times 16}{1 \times 8} = 16 \times 2$

So, it requires 16 vertical and 2 horizontal chips.
For this five 2 × 4 decoders are required.

Ans. (b)

PRACTICE EXERCISES

Set 1

1. Find the value of x in (257) base 10 = (41) base x?

(a) 61 (b) 64
(c) 127 (d) 256

2. What is the result of 01A2 + 001A = ?

(a) 01BC (b) 01CB
(c) 01CC (d) 01DC

3. The simplified function of Boolean function $F(W, X, Y, Z) = \Sigma(0, 1, 2, 5, 8, 9, 10)$ is

(a) $(W' + X')\ (Y' + Z')\ (X' + Z)$.
(b) $(W + X)\ (Y' + Z')\ (X' + Z')$.
(c) $(W' + X')\ (Y + Z)\ (X + Z)$.
(d) $(W' + X)\ (Y + Z')\ (X' + Z')$.

4. What is the hexadecimal representation of (1356) base 8?

(a) 2FE (b) 2DE
(c) 2EE (d) EEE

5. How many 3×8 decoders are needed to construct 4×16 decoder?

(a) 2 (b) 4
(c) 3 (d) 24/64

6. If 73 (in base x number system) is equal to 54 (in base y number system), the possible values of x and y are

(a) 11,8 (b) 8,13
(c) 11,11 (d) 8,11

7. Total number of 8×1 MUX required for implementation of 256×1 MUX is

(a) 37. (b) 35.
(c) 36. (d) 38.

8. Number of 8×1 MUX needed at second last level is (assume level having only $1(8 \times 1)$ MUX to be last level)

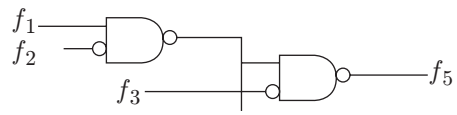
(a) 1 (b) 2
(c) 8 (d) 4

9. Match the columns

Column I	Column II
I. $xy + z(x \text{ XOR } y)$	A. $\sum m(1,2,4,7)$
II. x (XOR) y (XOR) z	B. $\sum m(3,5,6,7)$
III. $xy + yz + zx$	C. $\sum m(1,2,5,6)$

(a) I-A, II-C, III-B (b) I-B, II-A, III-B
(c) I-A, II-B, III-A (d) I-A, II-C, III-B

10. Consider the following circuit:



Given: $f_1(x, y, z) = \sum m(1, 2, 6, 7)$, $f_3(x, y, z) = \sum m(0, 3)$, $f_5(x, y, z) = \sum m(0, 1, 3, 6)$

How many different minterms combinations are possible for f_2?

(a) 15 (b) 12
(c) 8 (d) 6

11. The two numbers represented in signed 2's complement form is A = 11101101 and B = 11100110. If B is subtracted from A, the value obtained in signed 2's complement form is

(a) 10000111 (b) 00000111
(c) 10001001 (d) 11001001

12. Simplify the Boolean function

$F(w, x, y, z) = \sum(1,3,10) + \sum d(0,2,8,12)$

(a) $F = w'x' + x'z$ (b) $F = wx' + x'z'$
(c) $F = w'x + x'z'$ (d) $F = w'x' + x'z'$

13. Which of the following is the complement of function $F = x(y'z' + yz)$

(a) $x + yz' + y'z'$ (b) $x' + yz' + y'z$
(c) $x' + y'z' + y'z$ (d) $x' + y'z' + y'z'$

14. Match the following:

IC Family	Characteristics
P. TTL	(i) High component density
Q. ECL	(ii) Low power consumption
R. MOS	(iii) Evolution of "diodetransistor-logic"
S. CMOS	(iv) High-speed digital circuits

	P	Q	R	S
(a)	(i)	(ii)	(iii)	(iv)
(b)	(i)	(iii)	(iv)	(ii)
(c)	(iii)	(iv)	(i)	(ii)
(d)	(iv)	(iii)	(ii)	(i)

15. What is the base value of binary number system?

(a) 0 (b) 2
(c) 1 (d) 01

16. What is the maximum value that we can represent using N bits?

(a) $2n$ (b) 2^{n-1}
(c) $2^n - 2$ (d) $2^n - 1$

17. The number of bits required to represent 128 different numbers is

(a) 5. (b) 7.
(c) 6. (d) 8.

18. Consider the following truth table, where A, B, C represents the inputs and F the output function

A	*B*	*C*	*F*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

(a) $B'(A + C')(A' + C')$ (b) $B(A + C')(A' + C)$
(c) $B'(A' + C')(A' + C)$ (d) $B(A + C)(A' + C')$

19. Which of the following expressions is equivalent to $A \oplus B \oplus C$?

(a) $(A + B + C)(A' + B' + C')$
(b) $(A + B + C)(A' + B' + C)$
(c) $ABC + A'(B \oplus C) + B'(A \oplus C)$
(d) $(A' + B + C)(A' + B' + C')$

20. A half adder is also known as

(a) NOR circuit. (b) XNOR circuit.
(c) NAND circuit. (d) XOR circuit.

21. Which of the following is minimum error code?

(a) ANSI code (b) Grey code
(c) XS3 code (d) ASCII code

22. Register is a

(a) set of capacitors and inductors used to provide input instructions in a digital computer
(b) set memory cells used to store the I/O data.
(c) temporary storage unit within the CPU.
(d) permanent storage unit within the CPU.

23. A T FF can be realized by SR FF by?

(a) Connecting S to Q'.
(b) Connecting R is connected to Q'.
(c) Connecting S and R to Q.
(d) S and R are connected to Q and Q', respectively.

24. Which FF has its output clearly defined on all the combinations inputs?

(a) SR FF (b) D FF
(c) JK FF (d) T FF

25. The main difference between JK and RS flip−flop is that

(a) JK flip−flop do not need a clock pulse whereas RS flip−flop requires the same.
(b) The output Q is connected to J and K is grounded whereas it is opposite in R and S.
(c) JK flip−flop accepts both inputs as 1.
(d) JK flip−flop is acronym of Johnson−Keith's multivibrator and RS of reset−set multivibrator.

26. AND gate provides a LOW output _____.

(a) when any one input is LOW
(b) always
(c) when both the inputs are HIGH
(d) when both the inputs are LOW

27. How many pins does the 4049 IC have?

(a) 22 (b) 16
(c) 20 (d) 40

28. Which of the following expressions is in the sum-of-products (SOP) form?

(a) $A'B + C'D'$ (b) $AB'(CD)$
(c) $(A + B')(C + D)$ (d) $(A)B'(C + D')$

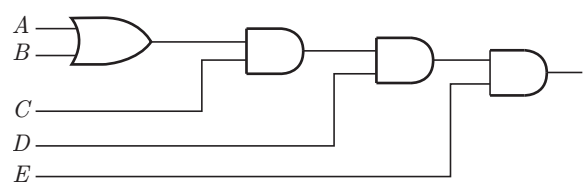
29. The significance of MSD in a decimal number system is

(a) the middle digit of a stream of numbers.
(b) the digit to the right of the decimal point.
(c) the last digit on the right.
(d) the digit with the most weight.

30. The Generic Array Logic (GAL) device is _____.

(a) complement of PAL.
(b) reprogrammable device.
(c) CMOS device.
(d) CMOS device which is also reprogrammable.

31. Derive the Boolean expression for the logic circuit shown below:



(a) $C(A + B)DE$ (b) $C(A + B)D + E$
(c) $ABCDE$ (d) $(DE) + (ABC)$

32. An XNOR gate gives a HIGH output with the following input combination

(a) A = LOW, B = HIGH
(b) A = HIGH, B = LOW
(c) A = LOW, B = LOW
(d) It is not possible to get a HIGH output in this gate

33. A shift register that has a parallel input, or a bidirectional serial load and having internal shift features, can be termed as

(a) Tristate device.
(b) Parallel-in shift-out (PISO) register.
(c) Universal shift register.
(d) Bidirectional shift register.

34. Which of the following logic families has the least propagation delay?

(a) 54LYY (b) BiCMOS
(c) ECL (d) 74SXX

35. Convert binary 111111110010 into hexadecimal.

(a) $(EF2)_{16}$ (b) $(FF2)_{16}$
(c) $(FE2)_{16}$ (d) $(FD2)_{16}$

36. Convert the binary number $(1001.0010)_2$ to decimal.

(a) 90.125 (b) 9.125
(c) 91.25 (d) 12.125

37. The alphanumeric code widely used for input and output to/from a computer?

(a) Biquinary Code (b) ASCII
(c) ANSI Code (d) EBCDIC

38. $(8B3F)_{16} = (?)_2$

(a) 111101101111 (b) 01110101111
(c) 1011001111101111 (d) 1000101100111111

39. Solving $-11 + (-2)$ will yield which 2's complement answer?

(a) 10011001 (b) 1000 1001
(c) 1111 0011 (d) 1110 1001

40. Dividing $(1100010)_2$ by $(0101)_2$ yields $(X)_{10}$. The value of X is _____.

41. The solution to $(0110)_{16} + (10010)_{16} = (?)_{16}$

(a) 10120_{16} (b) 11120_{16}
(c) 10020_{16} (d) $00B0_{16}$

42. Which of the following is correct for full adders?

(a) Full adders can be used for multiplication also.
(b) Full adders are used to make half adders and quad adders.
(c) Full adders have three inputs including a carry input.
(d) In a parallel full adder, the first stage may be a half adder.

43. What combination of the inputs a *JK* flip−flop toggles?

(a) $J = 0$, $K = 0$ (b) $J = 0$, $K = 1$
(c) $J = 1$, $K = 0$ (d) $J = 1$, $K = 1$

44. In a 6-bit Johnson counter sequence, the total number of states or bit patterns is _____.

Set 2

1. The simultaneous equations on the Boolean variables x, y, z and w, are

$$x + y + z = 1$$
$$xy = 0$$
$$xz + w = 1$$
$$xy + \overline{z}\overline{w} = 0$$

have the following solution for x, y, z and w, respectively.

(a) 0100 (b) 1101
(c) 1011 (d) 1000

2. Given the following Karnaugh map, which one of the following represents the minimal sum-of-product of the map?

yz \ wx	00	01	11	10
00	0	X	0	X
01	X	1	X	1
11	0	X	1	0
10	0	1	X	0

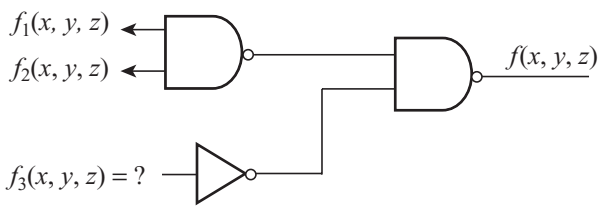
(a) $xy + y'z$
(b) $wx'y' + xy + xz$
(c) $w'x + y'z + xy$
(d) $xz + y$

3. Minimum SOP for $f(w, x, y, z)$ shown in Karnaugh-map below is

yz \ wx	00	01	11	10
00	0	1	1	0
01	X	0	0	1
11	X	0	0	1
10	0	1	1	X

(a) $xz + y'z$ (b) $xz' + zx'$
(c) $x'y + zx'$ (d) None of the above

4. Consider the following logic circuit whose inputs are functions f_1, f_2, f_3 and output f.



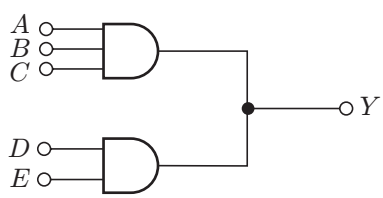
Given that $f_1(x, y, z) = \sum(0, 1, 3, 5)$; $f_2(x, y, z) = \sum(6, 7)$ and $f_3(x, y, z) = \sum(1, 4, 5)$ then f is

(a) $\sum(1, 4, 5)$ (b) $\sum(6, 7)$
(c) $\sum(0, 1, 3, 5)$ (d) $\sum(4, 5, 7)$

5. $f(A, B) = A' + B$. Simplified expression for function $f(f(x + y, y), z)$ is

(a) $x' + z$ (b) xyz
(c) $xy' + z$ (d) None of the above

6. Two NAND gates having open collector outputs are tied together as shown in the following figure. The logic function Y, implemented by the circuit is



(a) $Y = \bar{A}\bar{B}\bar{C} + \bar{D}\bar{E}$ (b) $Y = ABC + DE$
(c) $Y = \bar{A}\bar{B}\bar{C} \cdot \bar{D}\bar{E}$ (d) $Y = ABC \cdot DE$

7. What is the equivalent Boolean expression in product-of-sums form for the Karnaugh map given in the following figure.

CD \ AB	00	01	11	10
00		1	1	
01	1			1
11	1			1
10		1	1	

(a) $B\bar{D} + \bar{B}D$
(b) $(B + \bar{C} + D)(\bar{B} + C + \bar{D})$
(c) $(B + D)(\bar{B} + \bar{D})$
(d) $(B + \bar{D})(\bar{B} + D)$

8. Which of the following sets of component(s) is/are sufficient to implement any arbitrary Boolean function?

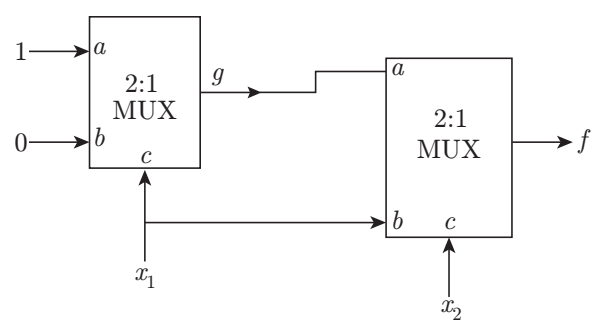
(a) XOR gates, NOT gates
(b) 2 to 1 Multiplexors
(c) AND gates, XOR gates
(d) Three-input gates that output $(A \cdot B) + C$ for the inputs A, B and C

9. Which function does NOT implement the Karnaugh map given in the following figure?

$wz\rightarrow$ / $xy\downarrow$	00	01	11	10
00	0	x	0	0
01	0	x	1	1
11	1	1	1	1
10	0	x	0	0

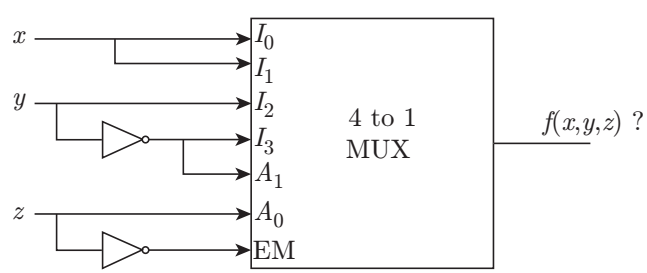
(a) $(w + x)y$ (b) $xy + yw$
(c) $(w + x)(w + y)(x + y)$ (d) None of the above

10. Consider the circuit shown below. The output of a 2:1 multiplexer is given by the function f?



(a) $x_1'x_2' + x_1x_2$ (b) $x_1'x_2 + x_1x_2'$
(c) $x_1' + x_2$ (d) $x_1' + x_2'$

11. Consider the following multiplexer where I_0, I_1, I_2, I_3 are four data input lines selected by two address line combinations $A_1A_0 = 00, 01, 10, 11$, respectively, and f is the output of the multiplexer, EN is the Enable input.



The function $f(x, y, z)$ implemented by the above circuit is

(a) xyz' (b) $xy + z$
(c) $x + y$ (d) None of the above

12. Which of the given number has its IEEE-754 32-bit floating-point representation as
(0 10000000 110 0000 0000 0000 0000 0000)

(a) 2.5 (b) 3.0
(c) 3.5 (d) 4.5

13. Consider the following state table in figure for a sequential machine. The number of states in the minimized machine will be

	0	1
A	D, 0	B, 1
B	A, 0	C, 1
C	A, 0	B, 1
D	A, 1	C, 1

(a) 4 (b) 3
(c) 2 (d) 1

14. A ROM is used to store the table for multiplication of two 8-bit unsigned integers. The size of ROM required is

(a) 256K × 16 (b) 64K × 8
(c) 4K × 16 (d) 64K × 16

15. Booth's algorithm for integer multiplication gives worst performance when the multiplier pattern is

(a) 101010……1010 (b) 100000…..001
(c) 111111……1111 (d) 011111…..1110

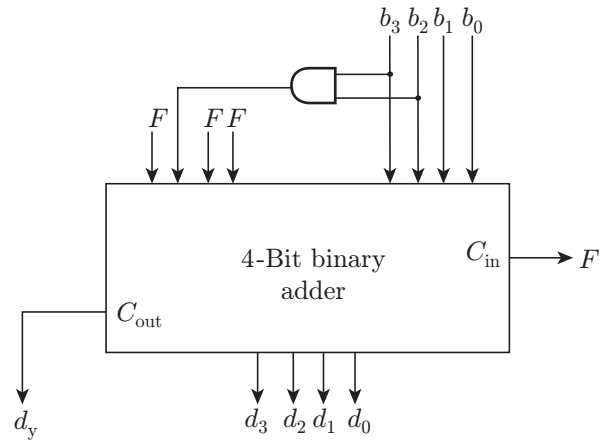
16. Consider the following floating point number representation.

31	24	23	0
Exponent		Mantissa	

The exponent is in 2's complement representation and mantissa is in the sign magnitude representation. The range of the magnitude of the normalized numbers in this representation is

(a) 0 to 1 (b) 0.5 to 1
(c) 2^{-23} to 0.5 (d) 0.5 to $(1 - 2^{-23})$

17. Consider the circuit in the following figure which has a 4-bit binary number $b_3b_2b_1b_0$ as input and a 5-bit binary number $d_4d_3d_2d_1d_0$ as output. The circuit implements



(a) Binary to Hex conversion
(b) Binary to BCD conversion
(c) Binary to grey code conversion
(d) Binary to radix −12 conversion

18. Given $\sqrt{(224)_r} = (13)_{r'}$

The value of the radix r is
(a) 10 (b) 8 (c) 5 (d) 6

19. Zero has two representations in:

(a) sign magnitude (b) 1's complement
(c) 2's complement (d) both (a) and (b)

20. The number 43 in 2's complement representation is

(a) 01010101 (b) 11010101
(c) 00101011 (d) 10101011

21. Consider the values $A = 2.0 \times 10^{30}$, $B = -2.0 \times 10^{30}$, $C = 1.0$ and the sequence

$X := A + B$
$Y := A + C$
$X := X + C$
$Y := Y + B$

(a) $X = 1.0,\ Y = 1.0$ (b) $X = 0.0,\ Y = 1.0$
(c) $X = 1.0,\ Y = 0.0$ (d) $X = 0.0,\ Y = 0.0$

22. The 2's complement representation of $(-539)_{10}$ in hexadecimal is

(a) ABE (b) DBC
(c) DE5 (d) 9E7

23. The decimal value 0.25

(a) is equivalent to binary 0.1
(b) is equivalent to binary 0.01
(c) is equivalent to binary 0.00111
(d) cannot be represented precisely in binary

24. Sign extension is the step in

(a) floating point multiplication.
(b) signed 16-bit integer addition.

(c) arithmetic left shift.
(d) converting a signed integer from one size to another.

25. The state of a 12-bit register is 100010010111. It is represented in BCD, what is its content?

(a) 2199 (b) 3423
(c) 564 (d) 897

26. Simplification of given Boolean expressions $= \{(a + b' + c')(a' + c')''\}$ to a minimum number of literals is

(a) $(c' + a'b')$ (b) $(a' + b'c')$
(c) $(b' + a'c')$ (d) $(a' + b' + c')$

27. Simplification of the following Boolean function $\{F(A, B, C, D) = \sum(3, 7, 11, 13, 14, 15)\}$ using K-maps:

(a) $CD + ABC + BCD$ (b) $CD + ABC + ABD$
(c) $CD + ACD + ABD$ (d) $CD + ABC + ACD$

28. A flip−flop has a 4-ns delay from the time the clock edge occurs to the time the output is complemented in a 10-binary ripple counter. The ripple counter operates reliably at which maximum frequency?

(a) 25 MHz (b) 30 MHz
(c) 32 MHz (d) 40 MHz

29. How many parity check bits must be included with the data word to achieve single-error correction when data word contains 16 bits?

(a) 4 Bits (b) 5 Bits
(c) 6 Bits (d) 7 Bits

30. How many 32K × 8 RAM chips are needed to provide a memory capacity of 256K bytes?

(a) 4 (b) 5
(c) 6 (d) 8

ANSWERS TO PRACTICE EXERCISES

Set 1

1. (b)	**9.** (b)	**17.** (b)	**25.** (c)	**33.** (c)	**41.** (a)
2. (a)	**10.** (a)	**18.** (d)	**26.** (a)	**34.** (c)	**42.** (d)
3. (a)	**11.** (b)	**19.** (c)	**27.** (b)	**35.** (b)	**43.** (d)
4. (c)	**12.** (d)	**20.** (c)	**28.** (a)	**36.** (b)	**44.** (12)
5. (a)	**13.** (b)	**21.** (b)	**29.** (d)	**37.** (b)	
6. (d)	**14.** (c)	**22.** (c)	**30.** (b)	**38.** (d)	
7. (a)	**15.** (b)	**23.** (d)	**31.** (a)	**39.** (c)	
8. (d)	**16.** (d)	**24.** (c)	**32.** (c)	**40.** (3)	

Set 2

1. (c) Considering each option

(a) Given $x = 0$, $y = 1$, $z = 0$ and $w = 0$

i. $x + y + z = 0 + 1 + 0 = 1$
ii. $xy = 0 \cdot 1 = 0$
iii. $xz + w = 0 \cdot 0 + 0 = 0$

but $xz + w$ should be 1, so option (a) is incorrect.

(b) Given $x = 1$, $y = 1$, $z = 0$ and $w = 1$

i. $x + y + z = 1 + 1 + 0 = 1$
ii. $xy = 1 \cdot 1 = 1$

However, given $xy = 0$
Therefore, option (b) is not the solution.

(c) Given $x = 1$, $y = 0$, $z = 1$ and $w = 1$

i. $x + y + z = 0 + 0 + 1 = 1$
ii. $xy = 1 \cdot 0 = 0$
iii. $xz + w = 1 \cdot 1 + 1 = 1$
iv. $xy + z'w' = 1 \cdot 0 + 0 \cdot 0 = 0$

It satisfies all the given conditions. So, (c) is the correct option.

2. (a)

yz \ wx	00	01	11	10
00	0	X	0	X
01	X	1	X	1
11	0	X	1	0
10	0	1	X	0

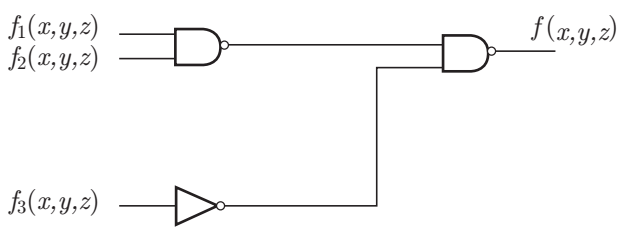
SOP is $xy + y'z$.

3. (c)

yz \ wx	00	01	11	10
00	0	1	1	1
01	x	0	0	0
11	x	0	0	0
10	0	1	1	x

Sum-of-product for given function is $x'y + zx'$.

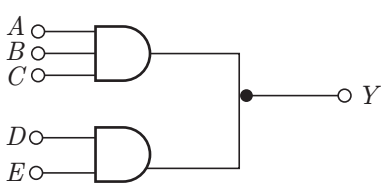
4. (a)



$$\begin{aligned} f_1f_2 &= f_1(x, y, z) \text{ NAND } f_2(x, y, z) \\ &= \sum (0, 1, 2, 3, 4, 6, 7) \\ f_3 &= \sum (1, 4, 5) \\ f_3' &= \sum (0, 1, 2, 3, 6, 7) \\ f &= f_1f_2 \text{ NAND } f_3' = (f_{12}f_3')' \\ &= (\sum (0, 1, 2, 3, 6, 7))' = \sum (1, 4, 5) \end{aligned}$$

5. (c) $f(f(x + y, y), z) = f((x + y)' + y, z) = f(x'y' + y, z)$
$= f(x' + y, z) = (x' + y)' + z = xy' + z$

6. (c)



Therefore, $Y = \bar{A}\bar{B}\bar{C} \cdot \bar{D}\bar{E}$

7. (c) The Karnaugh map for POS can be given by

AB \ CD	00	01	11	10
00	0	0	1	1
01	X	X	1	x
11	1	1	1	0
10	0	1	1	0

POS form of f is given as
$F(C, D, A, B) = (B + D)(\bar{B} + \bar{D})$

8. (c) Functionally complete operation set is the set of logic functions from which all other functions can be realized.

[XOR, AND] is a functionally complete set.

9. (c) For SOP form

xy \ wz	00	01	11	10
00	0	X	0	1
01	0	X	1	1
11	1	1	1	1
10	0	X	0	0

f in SOP can be given as

$$f(x, y, w, z) = xy + yw$$

or $$y(x + w)$$

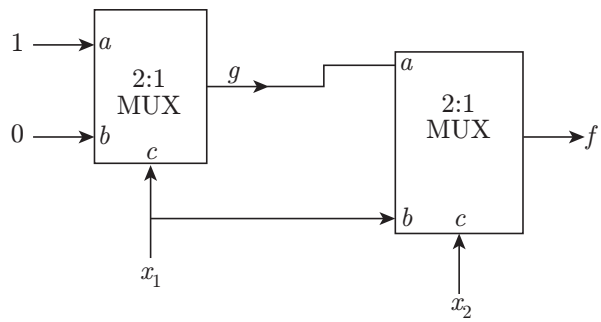
Therefore, options (a) and (b) are true.
For POS form

xy \ wz	00	01	11	10
00	0	X	0	0
01	0	X	1	1
11	1	1	1	1
10	0	X	0	0

$$f(x, y, w, z) = y(x + w)$$

Option (c) is incorrect.

10. (a)



The output g is $g = 1\ x_1' + 0\ x_1 = x_1'$
The output f is

$$f = gx_2' + x_1x_2$$
$$f = x_1'x_2' + x_1x_2$$

11. (a)

$$= \text{EN}(\bar{A}_1\bar{A}_0I_0 + \bar{A}_1A_0I_1 + A_1\bar{A}_0I_2 + A_1A_0I_3)$$
$$= z'(xyz' + yzx + 0 + y'zy')$$
$$= xyz' + xyzz' + 0 \cdot z' + z' \cdot zy'$$
$$= xyz' + 0 + 0 + 0 = xyz'$$

12. (c) Sign bit $S = 0 \Rightarrow$ positive number

$E = 1000\ 0000\text{B} = 128\text{D}$ (in normalized form)
Fraction is 1.11B (with an implicit leading 1) $= 1 + 1 \times 2^{-1} + 1 \times 2^{-2} = 1.75\text{D}$
The number is $+ 1.75 \times 2^{(128\ -\ 127)} = +3.5\text{D}$

13. (b)

x	**0**	**1**
A	D, 0	B, 1
B	A, 0	C, 1
C	A, 0	B, 1
D	A, 1	C, 1

These two states work same

Combining two states B and C into single state "BC".

x	**0**	**1**
A	D, 0	BC, 1
BC	A, 0	BC, 1
D	A, 0	BC, 1

Therefore, the number of states in the minimized machine will be 3.

14. (d) To store the table for multiplication of two 8-bit unsigned integers, size of ROM required is

$$2^{2N} \times 2N = 2^{16} \times 16 = 64\text{K} \times 16$$

15. (a)

Booth recording of a multiplier are

(a) 1 0 1 0 1 0 ..... 1 0 1 0
$\Downarrow$
−1 +1 −1 +1 −1 +1.....−1 +1 −1 0

(b) 1 0 0 0 0 0 .....0 0 0 1
$\Downarrow$
−1 + 0 0 0 0 0.....−0 0 +1 −1

(c) 1 1 1 1 1 1 ..... 1 1 1 1
$\Downarrow$
0 0 0 0 0 0..... 0 0 0 −1

(d) 0 1 1 1 1 1 ..... 1 1 1 0
$\Downarrow$
1 0 0 0 0 0 ..... 0 0 −1 0

Therefore, option (a) gives the worst performance because it contains maximum number of 1's.

16. (d)

31	24	23	0
Exponent		Mantissa	

The range of the magnitude of the normalized number: 0.5 to $(1 - 2^{-23})$

17. (d) Whenever both b_3 and b_2 is 1, then 01 00 is added to the number.

$$b_3b_2b_1b_0 = (1100)_B = (12)_{10}$$

So, in all the numbers greater than equal to 12, 0100 is added. Therefore, the circuit is binary to radix −12 converter.

18. (c)

(a) When the radix r is 10
Then $\sqrt{(224)_{10}} \approx 14.967 \neq (13)_{10}$

(b) When the radix is 8
Then $\sqrt{(224)_8} \approx \sqrt{(148)_{10}} = (12.165)_{10}$ and $(13)_8 = (11)_{10}$. Hence, $\sqrt{(224)_8} \neq (13)_8$
Option (b) is not correct.

(c) When the radix is 5
$\sqrt{(224)_5} \approx \sqrt{(64)_{10}} = (8)_{10}$ and $(13)_5 = (8)_{10}$.
Therefore, $\sqrt{(224)_5} = (13)_5$

Option (c) is correct.

(d) When the radix is 6

$$\sqrt{(224)_6} = \sqrt{(88)_{10}} \approx (9.38)_{10} \text{ and } (13)_6 = (9)_{10}.$$

Therefore, $\sqrt{(224)_6} \approx (13)_6$

Option (d) is not correct.

19. (d) Zero has two representations in

Sign magnitude as (i) 0 0 0 0, (ii) 1 0 0 0
1's Complement as (i) 0 0 0 0, (ii) 1 1 1 1

20. (c) Positive 2's complement numbers are represented as same simple binary. So +43 in binary is 0 0 1 0 1 0 1 1.

21. (a) Given that $A = 2.0 \times 10^{30}$, $B = -2.0 \times 10^{30}$, $C = 1.0$

(i) $X = A + B = 2.0 \times 10^{30} + (-2.0 \times 10^{30}) = 0.0$

(ii) $Y = A + C = 2.0 \times 10^{30} + (1.0)$

(iii) $Y = X + C = 0.0 + 1.0 = 1.0$

(iv) $Y = Y + B = 2.0 \times 10^{30} + 1.0 + (-2.0 \times 10^{30}) = 1.0$

Values of X and Y are 1.0 and 1.0, respectively.

22. (c) Compute a binary representation of the magnitude of the number and then take 2's complement of binary representation.

$$\begin{aligned} +539 &= 0010\ 0001\ 1011 \\ -539 &= 1101\ 1110\ 0101 \\ &= (\text{DE5})_{16} \end{aligned}$$

23. (b)

$$\begin{aligned} 0.25 \times 2 &= 0.50 \quad 0 \\ 0.50 \times 2 &= 1.00 \quad 1 \end{aligned}$$

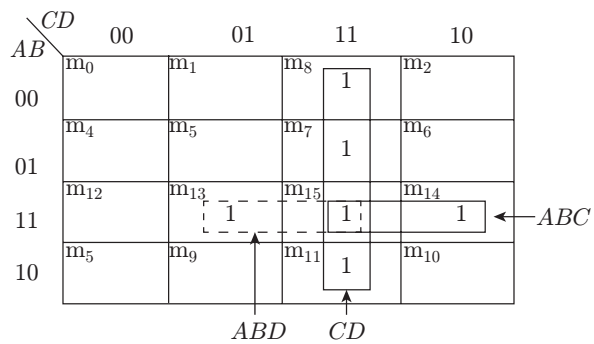
Therefore, the decimal value 0.25 is equivalent to binary 0.01.

24. (d) Sign extension is the operation for increasing the number of bits of a binary number while preserving the number's sign (positive/negative) and value.

25. (d)

26. (a) $(a + b' + c')(a' + c')$
$\Rightarrow aa' + ac' + a'b' + b'c' + a'c' + c'c'$
$\Rightarrow c' + b'(a' + c')$
$\Rightarrow c' + a'b'$

27. (b)



$F = CD + ABC + ABD$

28. (a) In worst case, all 10 flip−flops are complemented. The maximum delay is 10×3 ns $= 30$ ns. So, the maximum frequency is $10^9/40 = 25$ MHz.

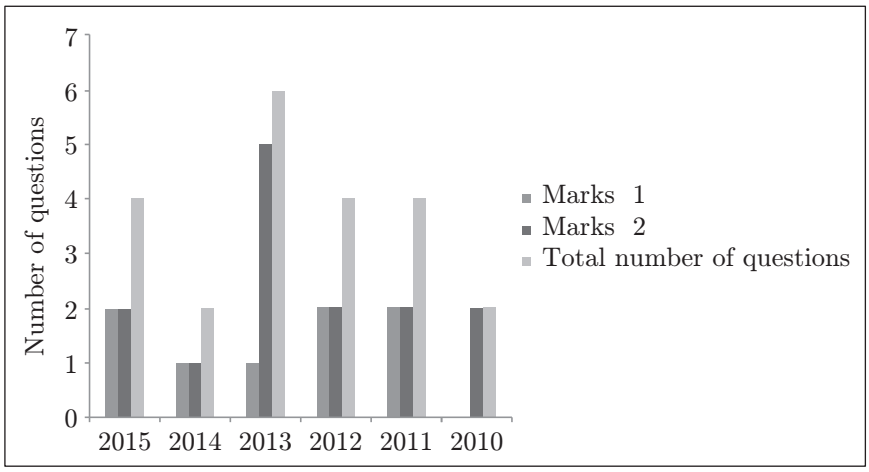
29. (b) Solving for n in terms of k, we obtain $2^k - 1 - k \geq n$

For data word length $(n) = 16$ bit we require 5 bit.

30. (d) $256/32 = 8$ chips

UNIT II: COMPUTER ORGANIZATION AND ARCHITECTURE

LAST SIX YEARS' GATE ANALYSIS



Concepts on which questions were asked in the previous six years

Year	Concepts
2015	Cache memory, Memory, Pipelining, Main memory, Pipeline, Anti-dependency, Instruction format, Byte Addressable memory, Pipeline, Three-address code
2014	Memory access, pipelining
2013	Cache memory, hard disk, pipelining, registers, micro-operations
2012	Pipelining, ROM, cache, file system
2011	Cache memory, interrupts
2010	Memory, pipelining, registers, cache memory

CHAPTER 2

COMPUTER ORGANIZATION AND ARCHITECTURE

Syllabus: Machine instructions and addressing modes, ALU and data path, CPU control design, memory interface, I/O interface (interrupt and DMA mode), instruction pipelining, cache and main memory, secondary storage

2.1 INTRODUCTION

Computer architecture and organization is the science of interconnecting hardware components, designing and configuring the hardware/software interface to fulfill functional and performance goals of a computer. This chapter outlines the basic hardware structure of a modern digital programmable computer, the basic laws for performance evaluation, designing the control and data path hardware for a processor, concept of pipelining for executing machine instructions simultaneously and designing fast memory and storage systems.

Computer architecture deals with the structure and behaviour of computer system as viewed by the user. It encompasses instruction formats, the instruction set architecture (ISA) and addressing modes.

Computer organization deals with the operation and interconnection of the various hardware components.

2.2 COMPUTER ARCHITECTURE

2.2.1 Register Set

The computer needs registers for processing and manipulating data and for holding memory addresses that are available to the machine-code programmer. Some registers for a basic computer are given in Table 2.1.

Table 2.1 Types of registers and their functions

Register Symbol	Register Name	Function
DR	Data register	Holds memory operand
ACC	Accumulator	Special purpose processor register
AR	Address register	Holds address for memory

(*Continued*)

Table 2.1 | Continued

Register Symbol	Register Name	Function
IR	Instruction register	Holds an instruction that is to be executed
PC	Program counter	Holds address of instruction to be executed next
TR	Temporary register	Holds temporary data if required
INPR	Input register	Holds input character
OUTR	Output register	Holds output character

2.2.2 Quantitative Principles to Design High-Performance Processor

Amdahl's law focused on performance gain after enhancing the system. The performance gain is denoted by S_{overall} and ET stands for execution time.

$$S_{\text{overall}} = \frac{\text{Performance of the system with enhancement}}{\text{Performance of the system without enhancement}}$$

$$S_{\text{overall}} = \frac{1/\text{ET}_{\text{new}}}{1/\text{ET}_{\text{old}}} \tag{2.1}$$

$$S_{\text{overall}} = \frac{\text{ET}_{\text{old}}}{\text{ET}_{\text{new}}}$$

After enhancement, the system consists of two portions: unenhanced and enhanced portion.

$$\text{ET}_{\text{new}} = \text{ET of the unenhanced portion} + \text{ET of enhanced portion}$$

To calculate ET_{new}, the following two factors are needed:

1. **Fraction$_{\text{enhance}}$ (F):** It indicates how much portion of the old system undergoes enhancement.
2. **Speed$_{\text{enhance}}$ (S):** It indicates how many times the new portion is running faster than the old portion.

$$S = \frac{\text{Performance}_{\text{new}}F}{\text{Performance}_{\text{old}}F} = \frac{1/\text{ET}_{\text{new}}F}{1/\text{ET}_{\text{old}}F} = \frac{\text{ET}_{\text{old}}F}{\text{ET}_{\text{new}}F}$$

So, $$\text{ET}_{\text{new}}F = \frac{\text{ET}_{\text{old}}F}{S}$$

On the basis of the above factor,

$$\text{ET}_{\text{new}}F = \text{ET}_{\text{old}}(1-F) + \frac{\text{ET}_{\text{old}}F}{S}$$

Substitute the value of ET in Eq. (2.1):

$$\text{ET}_{\text{new}}F = \text{ET}_{\text{old}}(1-F) + \frac{\text{ET}_{\text{old}}F}{S}$$

Let $\text{ET}_{\text{old}} = 1$,

$$S_{\text{overall}} = \frac{1}{(1-F)+(F/S)} = \left[(1-F)+\frac{F}{S}\right]^{-1}$$

When the system consists of multiple frequent cases, where i is the number of frequent cases:

$$S_{\text{overall}} = \left[\left(1-\sum F_i\right)+\sum\frac{F_i}{S}\right]^{-1}$$

Problem 2.1: Consider a hypothetical processor used in mathematical model simulation. It consists of two functional units, floating point and integer. The floating point is enhanced then it runs two times faster, but only 10% of the instructions are floating point. What is the speed up?

Solution: Here $S = 2$, $F = 0.1$

$$S_{\text{overall}} = \left[(1-0.1)+\frac{0.1}{2}\right]^{-1} = 1.052$$

2.3 MACHINE INSTRUCTIONS AND ADDRESSING MODES

Machine instruction is an individual machine code. The complete set of all machine codes recognized by a particular processor makes its Instruction Set. Instructions can be grouped according to the function they perform. The number of ways by which arguments for these machine instructions can be specified constitutes the addressing modes for a processor.

2.3.1 Machine Instructions

An instruction is a command to the microprocessor to perform a given task. Most computer instructions are classified as follows:

1. **Data transfer instructions:** These instructions move data from one place to another in the computer without changing the data content. Example: LOAD, MOVE, IN, OUT, PUSH, STORE.
2. **Data manipulation instructions:** These instructions perform arithmetic, logical and shift operations on data. Example: ADD, SUB, MUL, DIV, INC, AND, XOR, OR, SHR, SHL, ROR, ROL.
3. **Program control instructions:** These instructions may change the address value in program counter and cause the normal sequential flow to change.

On the basis of the number of address fields in an instruction, they are classified as follows:

1. **Three-address instruction:** Computer with three-address instruction format can use each address field to specify two sources and a destination, which can be either a processor register or a memory operand. It results in short program but requires too many bits to specify three addresses.

   Example: ADD R_1, A, B $\quad (R_1 \leftarrow M[A] + M[B])$

2. **Two-address instruction:** Each address field can specify either a processor register or a memory word.

   Example: MOV R_1, A ($R_1 \leftarrow M[A]$);
   MUL R_1, R_2 ($R_1 \leftarrow R_1 * R_2$)

3. **One-address instruction:** It used an implied accumulator (AC) register for all data manipulation. The other operand is in register or memory.

   Example: LOAD A (AC $\leftarrow M[A]$);
   ADD B (AC $\leftarrow$ AC + $M[B]$)

4. **Zero-address instruction:** A stack organized computer does not use an address field for the instruction ADD and MUL.

Problem 2.2:

(a) ISA of a processor consists of 64 registers, 125 instructions and 8 bits for immediate mode. In a given program, 30% of the instructions take one input register and have one output register, 30% have two input registers and one output register, 20% have one immediate input, and one output register, and remaining have two immediate input, 1 register input and one output register. Calculate the number of bits required for each instruction type. Assume that the ISA requires that all instructions be a multiple of 8 bits in length.

(b) Compare the memory space required with that of variable length instruction set.

Solution:

(a) Since there are 125 instructions so we need 7 bits to differentiate them as 64 < 125 < 128. For 64 registers, we need 6 bits and 8 bits for immediate mode.
For Type 1, 1 reg in, 1 reg out: 7 + 6 + 6 = 19 bits ~ 32 bits
For Type 2, 2 reg in, 1 reg out: 7+ 6 + 6 + 6 = 26 bits ~ 32 bits
For Type 3, 1 imm in, 1 reg out: 7 + 6 + 8 = 21 bits ~ 32 bits
For Type 4, reg in, 2 imm in, 1 reg out: 7 + 6 + 8 + 8 + 6 = 35 bits ~48 bits

(b) As the largest instruction type requires 48 bit instructions, the fixed-length instruction format uses 48 bits per instruction. Variable length instruction format uses $0.3 \times 32 + 0.3 \times 32 + 0.2 \times 32 + 0.2 \times 48 = 36 =$ bits on average, that is, 25% less space.

2.3.2 Addressing Modes

The addressing mode specifies how effective address of an operand is calculated from an instruction. Computers use various addressing mode techniques to:

1. provide programming flexibility to users through use of pointers to memory, counter for loop control, data indexing and program relocation.
2. reduce the size of the addressing field of the instruction.

Let us suppose $[x]$ means contents at location x for all the addressing modes.

2.3.2.1 Types of Addressing Modes

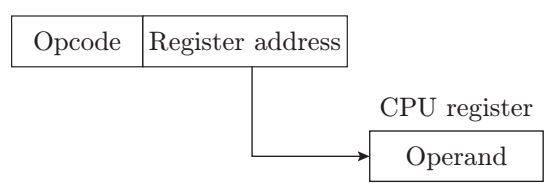
1. **Implied mode:** In this mode, the operands are implicitly stated in the instruction. For example, register reference instructions such as CMA (complement accumulator), CLA (clear accumulator) and zero-address instructions that use stack organization.
2. **Immediate mode:** In this mode, the operand is specified in the instruction itself, that is, address field is replaced by an actual operand. Immediate mode instructions are useful for initializing registers to a constant value. For example, used for initializing CPU registers to some constant value such as MOV R_1, #34.

   Instruction with immediate mode

Opcode	Data Operand

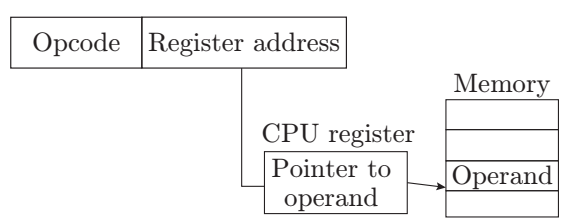
3. **Register (direct) mode:** In this mode, the operands are in CPU registers. An n-bit register field can specify any one of 2^n registers. Example: ADD R_1 will add the contents of an accumulator and contents of R_1, that is, ACC = [ACC] + [R_1].

   Instruction with register direct mode



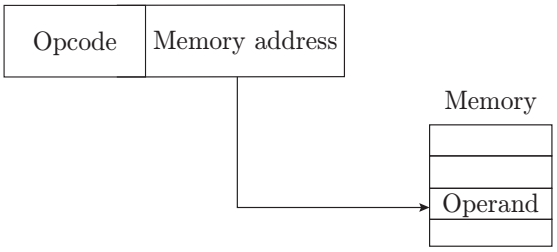
4. **Register (indirect) mode:** In this mode, the instruction format specifies a CPU register which contains an effective address of the operand residing in memory. This mode ensures less number of bits to specify a register value than to specify a memory location. Example: ADD @ R_1 will add the contents of an accumulator with contents of the register R_1, that is, AC = [ACC] + [[R_1]].

   Instruction with register indirect mode



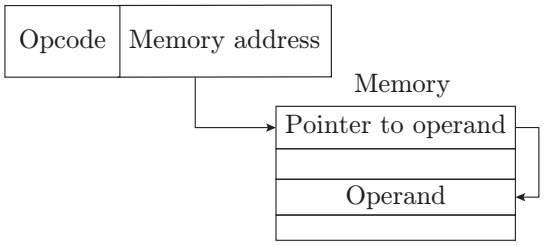
5. **Auto-increment or Auto-decrement mode:** This is similar to register indirect mode except the register containing effective address is incremented or decremented after (or before) its value is used to access memory.
6. **Direct address mode:** In this mode, the effective address of an operand is equal to the address part of the instruction. Example: ADD A instruction adds content of memory cell A to accumulator, that is, ACC = [ACC] + $M[A]$.

   Instruction with direct address mode



7. **Indirect address mode:** In this mode, memory address specified by address field contains the address of (pointer to) the operand. Example: ADD @ A will add the contents of the memory cell A, that is, ACC = [ACC] + $M[M[A]]$.

   Instruction with indirect address mode



8. **Relative address mode:** In this mode, the effective address of an operand is obtained by adding the content of a program counter to the address part of the instruction. The address part of the instruction can be either positive or negative represented in 2's complement. The result obtained after adding the content of the program counter to the address field produces an effective address whose position in memory is relative to the address of the next instruction.
9. **Index address mode:** In this mode, the effective address of an operand is obtained by adding the content of an index register to the address part of the instruction. The index register is a special CPU register that stores an index value and the address field of the instruction stores the base address of a data array in the memory. The distance between the base address and the address of the operand is the index value that is stored in the index register. The index register can be incremented to facilitate access to consecutive operands stored in arrays using the same instruction.
10. **Base register addressing mode:** In this mode, the effective address of an operand is obtained by adding the content of a base register to the address part of the instruction. This is somewhat similar to the indexed addressing mode except that the base register stores base or beginning address instead of an index register. It is used for program relocation.

Problem 2.3: A two-word instruction LOAD is stored at location 300 with its address field in the next location. The address field has value 600 and value stored at 600 is 500 and at 500 is 650. The words stored at 900, 901 and 902 are 400, 401 and 402, respectively. A processor register R contains the number 800 and index register has value 100. Evaluate the effective address and operand if addressing mode of the instruction is as follows:

1. Direct
2. Indirect
3. Relative
4. Immediate
5. Register indirect
6. Index

Solution: Memory layout is as follows

300	LOAD
301	600
500	650
600	500
700	900
800	700
900	400
901	401
902	402

Addressing Mode	Effective Address	Operand
Direct	600	500
Indirect	500	650
Relative	902	402
Immediate	301	600
Register indirect	800	700
Index	700	900

Problem 2.4: A relative mode branch type instruction is stored in memory at an address equivalent to decimal 600 and the branch is made to an address equivalent to decimal 400. What is the value of the relative address field of the instruction (in decimal)?

Solution: Relative address = 400 − 601 = −201

Table 2.2 | Arithmetic circuit function table

Select			Input to Adder Y	Output of Binary Adder $D = A + Y + C_{in}$	Micro-Operation
S_1	S_0	C_{in}			
0	0	0	0	$D = A$	Transfer A
0	0	1	0	$D = A + 1$	Increment A
0	1	0	B	$D = A + B$	Add
0	1	1	B	$D = A + B + 1$	Add with carry
1	0	0	$\bar{B}$	$D = A + \bar{B}$	Subtract with borrow
1	0	1	$\bar{B}$	$D = A + \bar{B} + 1$	Subtract
1	1	0	1	$D = A - 1$	Decrement A
1	1	1	1	$D = A$	Transfer A

2.4 ARITHMETIC LOGIC UNIT

Arithmetic logic unit (ALU) is a combinational circuit that performs all arithmetic and logic operations so that the entire register transfer operation from the source registers through the ALU and into the destination register can be performed during one clock pulse period.

2.4.1 Arithmetic Micro-Operations

The basic arithmetic micro-operations such as addition, subtraction, increment, decrement and shift are performed on numeric data stored in registers. The basic component of arithmetic is parallel binary adder, and by controlling the input to adder, different micro-operations can be realized. Figure 2.1 depicts a 2-bit arithmetic circuit which includes two full-adder circuits and two multiplexers for choosing different arithmetic micro-operations. There are two 2-bit input numbers A and B and 2-bit output D. The two inputs from A go directly to X inputs of full adder. The output of multiplexer goes to input Y of full adder.

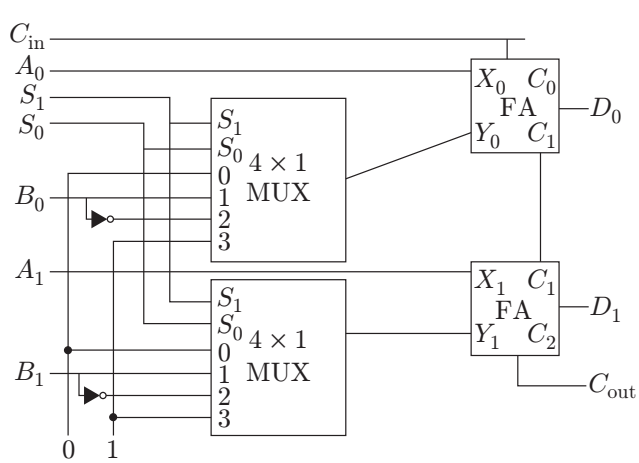


Figure 2.1 | A 2-bit arithmetic circuit.

By controlling the output Y of multiplexers with two selection inputs S_1 and S_0 and C_{in} either 0 or 1, we can generate the eight arithmetic micro-operations (Table 2.2).

2.4.2 Logic Micro-Operations

Logic micro-operations such as AND, OR, Exclusive OR, etc., consider each bit of register separately and specify binary operations for strings of bits (Table 2.3).

Table 2.3 | Types of micro-operations

Micro-operation	Name
$F \leftarrow 0$	Clear
$F \leftarrow A \wedge B$	AND
$F \leftarrow A \wedge \bar{B}$	
$F \leftarrow A$	Transfer A
$F \leftarrow \bar{A} \wedge B$	
$F \leftarrow B$	Transfer B
$F \leftarrow A \oplus B$	Exclusive OR
$F \leftarrow A \vee B$	OR
$F \leftarrow \overline{A \vee B}$	NOR
$F \leftarrow \overline{A \oplus B}$	Exclusive NOR
$F \leftarrow \bar{B}$	Complement B
$F \leftarrow A \vee \bar{B}$	
$F \leftarrow \bar{A}$	Complement A
$F \leftarrow \bar{A} \vee B$	
$F \leftarrow \overline{A \wedge B}$	NAND
$F \leftarrow$ all 1's	Set to all 1's

Logical micro-operations are capable of manipulating individual bits or a portion of word stored in CPU registers. Let us consider the data in a register A. In another register, B is the operand that will be used to modify the contents of A using logic micro-operations. Some of the applications are as follows:

1. **Selective set operation:** In this, If a bit in B is set to 1, that same position in A sets to 1, otherwise that bit in A retains its previous value.

$$\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ \hline 1 & 1 & 1 & 0 \end{array} \quad \begin{array}{l} A_t \\ B\,(\text{To set some bits in } A) \\ A_{t+1}(A \leftarrow A + B) \end{array}$$

2. **Selective complement operation:** If a bit in B is set to 1, that same position in A gets complemented from its original value, otherwise it remains unchanged.

$$\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ \hline 0 & 1 & 1 & 0 \end{array} \quad \begin{array}{l} A_t \\ B\,(\text{To complement some bits in } A) \\ A_{t+1}(A \leftarrow A \oplus B) \end{array}$$

3. **Selective clear operation:** If a bit in B is set to 1, that same position in A sets to 0, otherwise it remains unchanged.

$$\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 0 \end{array} \quad \begin{array}{l} A_t \\ B\,(\text{To clear some bits in } A) \\ A_{t+1}(A \leftarrow A \cdot B') \end{array}$$

4. **Mask operation:** If a bit in B is set to 0, that same position in A sets to 0, otherwise it remains unchanged.

$$\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 \end{array} \quad \begin{array}{l} A_t \\ B\,(\text{To clear some bits in } A) \\ A_{t+1}(A \leftarrow A \cdot B) \end{array}$$

5. **Clear operation:** If the bits in the same position in A and B are the same, they are cleared in A, else they are set in A.

$$\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ \hline 0 & 1 & 1 & 0 \end{array} \quad \begin{array}{l} A_t \\ B \\ A_{t+1}\,(A \leftarrow A \oplus B) \end{array}$$

6. **Insert operation:** It is used to insert a specific bit pattern into A register, leaving the other bit positions unchanged. This is accomplished by two sub-operations: masking operation to clear the desired bit positions, followed by OR operation to introduce the new bits into the desired positions. Suppose you wanted to introduce 10 into the low order two bits of A:

1101 A (Original) and 1110 A (Desired)

$$\begin{array}{cccc} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 1 & 1 & 1 & 0 \end{array} \quad \begin{array}{l} A\ (\text{Original}) \\ \text{Mask} \\ A\,(\text{Intermediate}) \\ \text{Added bits} \\ A\ (\text{Desired}) \end{array}$$

2.5 CPU CONTROL DESIGN

Central processing unit (CPU), or the brain of a computer, performs the data processing operations. It consists of three major parts: register set that stores intermediate data during instruction execution, ALU performs the required micro-operations and control unit that supervises all other elements for the transfer of information from one register to the other. The main function of a CPU is to fetch an instruction from the memory and execute it. CPU is divided into three types of organizations:

1. **Single accumulator organization:** In this, one operand is implied in the accumulator, a special purpose register, and the other operand is a register or the memory. Example: ADD R_1 ($R_1 \leftarrow$ AC + R_1), LOAD A (AC $\leftarrow A$), STORE T ($M[T] \leftarrow$ AC).
2. **General register organization:** In this, the CPU will have several general purpose registers which lead to shorter and efficient programs because registers are faster. Example: ADD R_1, R_2 ($R_1 \leftarrow R_1 + R_2$). Figure 2.2 shows bus organization for three registers R_1, R_2 and R_3. The output of these registers and one from the external input is connected to two multiplexers A and B. The two

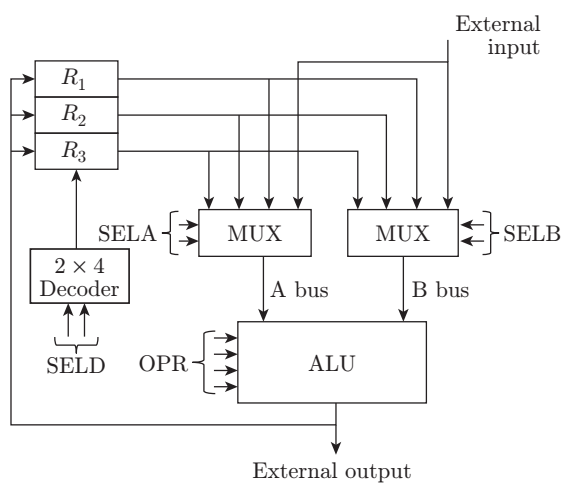


Figure 2.2 General register organization.

select lines SELA and SELB from multiplexers A and B select one of the input and feed to ALU. OPR specifies one of the possible operation codes that ALU will perform on the data inputs and the output is transferred either to one of the registers using 2 × 4 decoder or to the external output say memory. The control word (Fig. 2.3) for the two-operand instruction is as follows:

2 bits	2 bits	2 bits	4 bits
SELA	SELB	SELD	OPR

Figure 2.3 A control word.

3. **Stack organization:** Stack may consist of number of registers or a part of main memory in which data items are stored in consecutive locations that are accessed by LIFO (last in, first out) mechanism. As there is limited number of registers, a part of memory is implemented as stack for storage and retrieval of intermediate data. Stack pointer (SP) keeps a track of the top item of a stack. The process of inserting a new item onto a stack is known as push accomplished by first incrementing stack pointer and then inserting an item from the data register.

$$\text{SP} \leftarrow \text{SP} + 1$$

$$M[\text{SP}] \leftarrow \text{DR}$$

The process of removing an item from the top of a stack is known as pop performed by first transferring data into DR and then decrementing SP.

$$\text{DR} \leftarrow M[\text{SP}]$$

$$\text{SP} \leftarrow \text{SP} - 1$$

Problem 2.5: A system has CPU organized in the form of general register organization consisting of 16 registers, each storing 32-bit data. Assume the ALU has 35 operations.

(a) How many multiplexers are there in A bus and B bus, and what is the size of each multiplexer?
(b) How many selection inputs are needed for MUX A and MUX B?
(c) How many inputs and outputs are there in a decoder?
(d) How many inputs and outputs are there in ALU for data, including input and output carries?
(e) Formulate a control word for the system.

Solution:

(a) 32 Multiplexers, each of size 16×1.
(b) 4 Inputs each, to select one of 16 registers.
(c) 4 to 16 – Line decoder
(d) $32 + 32 + 1 = 65$ data input lines
(e) $32 + 1 = 33$ data output lines

4 bits	4 bits	4 bits	6 bits
SELA	SELB	SELD	OPR

2.5.1 Instruction Execution

A CPU generally executes one instruction at a time sequentially and a sequence of such instructions is known as a program. The CPU executes the instructions that reside in the main memory. In order to execute an instruction, the CPU has to fetch the instruction first from the main memory into one of its registers. It then **decodes** the instruction, that is, it decides what the instruction intended to do, fetch operands required and finally **executes** the instruction. This process is repeated continuously for a complete program and is known as the **fetch–execute** cycle (Fig. 2.4). The following steps are performed for executing an instruction:

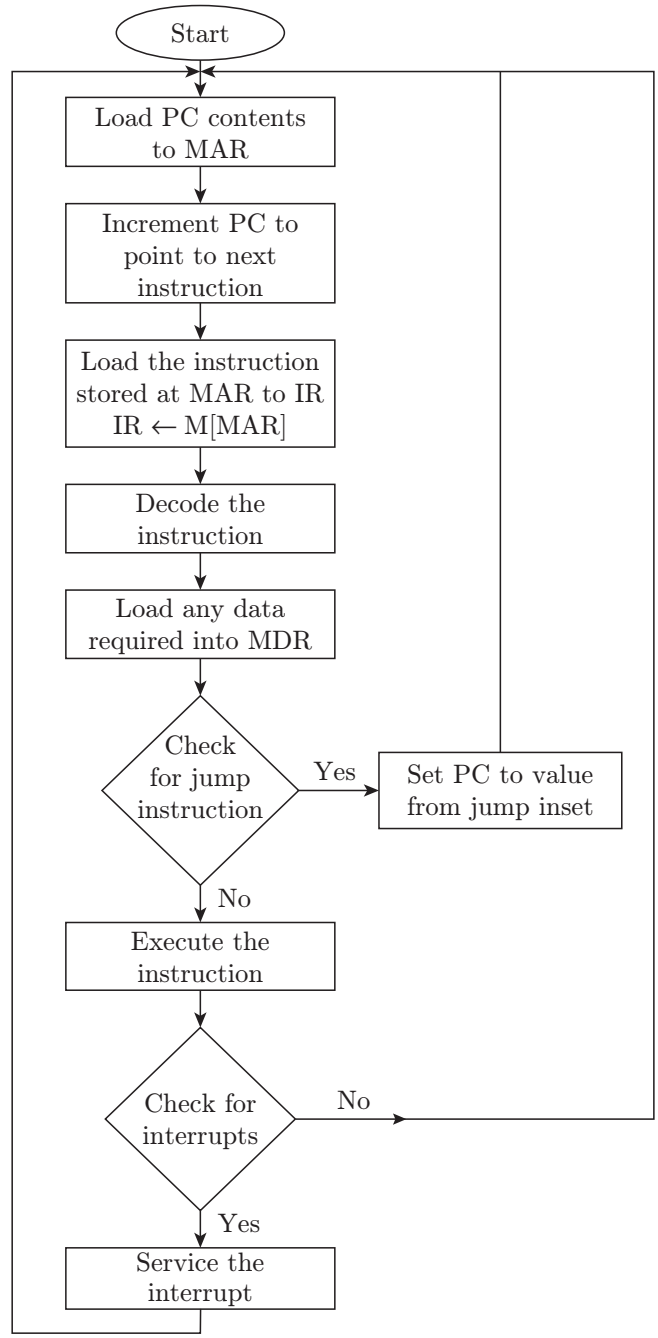


Figure 2.4 Instruction cycle.

1. **Fetching the instruction:** The next instruction is fetched from the memory address that is saved in the program counter, and memory content fetched is stored in instruction register (IR). The program counter then points to the next instruction that will be read in the next cycle.
2. **Decode the instruction:** During this cycle, the instruction inside the IR gets interpreted by the decoder.

3. **Operand fetch:** In case of a direct or indirect memory instruction, the execution begins in the next clock cycle. If the instruction has an indirect address, the effective address of the operand is read from the main memory, and the required data is fetched from the memory into memory data registers. If the instruction has direct address, nothing is done at this clock cycle.
4. **Execute the instruction:** The control unit of the CPU passes the instruction decoded by decoder as a sequence of control signals to the different functional units of the CPU to execute the tasks required by the instruction such as reading values from registers or input devices, performing mathematical or logic micro-operations by ALU, and writing the result back to a register or main memory.

2.5.2 CPU Data Path

CPU contains data paths that are responsible for routing data between the functional units of a computer. The following are the different data path structures available for routing:

1. **Single bus structure:** In this architecture, all CPU registers are connected to the same bus. Data can be transferred either between CPU registers or between CPU register and ALU at a given clock pulse. The speed of operation is slow as only one operand can be transferred in one clock cycle and addition operation ($R_1 \leftarrow R_2 + R_3$) occurs in three clock cycles.
2. **Two bus structure:** All general purpose CPU registers are connected to both buses say bus A and bus B; but special purpose registers are divided into two groups, say group 1 connecting bus A to program counter and one input of ALU and group 2 connecting bus B to MDR (Memory Data Register) and other input of ALU. The two operands are transferred to ALU in 1 clock cycle and the addition operation ($R_1 \leftarrow R_2 + R_3$) occurs in 2 clock cycles.
3. **Three bus structure:** The performance can be further be improved by using three buses such that addition operation ($R_1 \leftarrow R_2 + R_3$) can occur in one clock cycle.

2.5.3 Control Unit Design

Control unit is considered as brain of a CPU that controls various units in the data path. The performance of control unit is important as it determines the clock cycle of the processor. Control unit can be designed either by hardwired or by microprogram.

1. **Hardwired control:** Control unit is made up of sequential and combinatorial circuits to generate the control signals and interpret instructions (Fig. 2.5). The instruction decoder decodes the instruction loaded in instruction register. The step decoder generates a separate control line for each step in the control sequence. The encoder gets its input signal from the decoder, step decoder, external input and condition codes and generates individual control signals. It is faster and more efficient but less flexible and is difficult to add new feature or correct mistakes in original design.

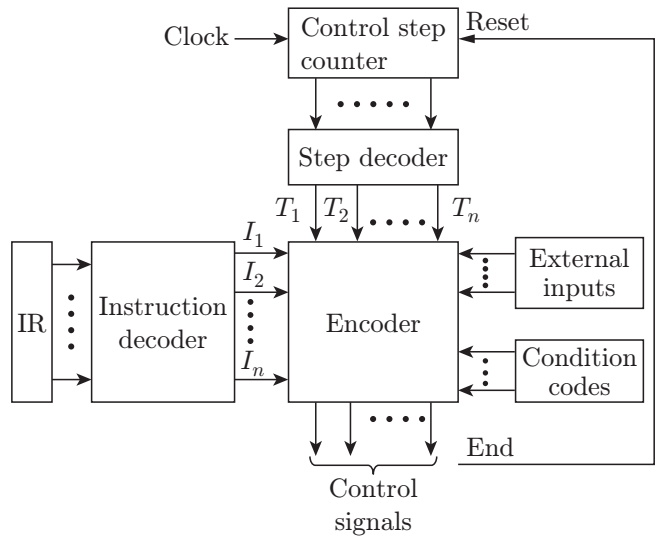


Figure 2.5 | Block diagram of hardwired control unit.

2. **Micro-programmed control:** Control signals are generated by using programming known as micro-programs that constitutes micro-instructions (control word) (Fig. 2.6). Memory that is part

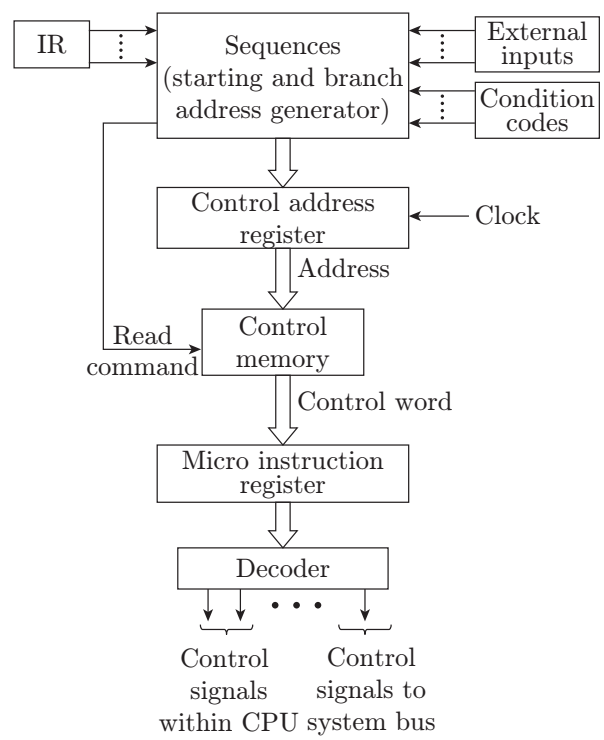


Figure 2.6 | Block diagram of micro-programmed control unit.

of CPU is known as control memory and stores micro-instructions. The micro-program sequencer generates the address of micro-instruction according to instruction stored in instruction register. The address of micro-instruction to be executed is available in content addressable register. Micro-program sequencer issues read command to read micro-instruction from control memory into micro-instruction register which on execution generates control signals for various parts of a processor. This control unit design is more flexible to accommodate new features and less error prone but quite slower than the hardwired unit.

The format of the control word is

Branch condition	Flag	Control signal	Control memory address

On the basis of the type of control word supported, it is divided into two types:

1. **Horizontal micro-programmed control unit:** In this design, the control signals are represented in the form of 1 bit per control signal and it supports longer control word.
2. **Vertical micro-programmed control unit:** In this design, the control signal is represented by using encoding format.

Problem 2.6: Consider a control unit which has 1024 control word memory; it supports 48 control signals and 8 flag conditions. What is the size of the control word in bits and control memory in bytes?

Solution:

(a) Using horizontal programmed control unit

0 bits	3 bits	48 bits	10 bits
Branch condition	Flag	Control signal	Control memory

Size of control word = 61 bits
Control memory = $(1024 \times 61)/8 = 128 \times 61$ bytes

(b) Using vertical programmed control unit

0 bits	3 bits	48 bits	10 bits
Branch condition	Flag	Control signal	Control memory

log 48 ~ 6 bits

Size of control word = 19 bits
Control memory = $(1024 \times 19)/8 = 128 \times 19$ bytes

2.5.4 RISC versus CISC Processors

The differences between reduced and complex instruction set computers is given in Table 2.4

Table 2.4 | RISC versus CISC

RISC (Reduced Instruction Set Computers)	CISC (Complex Instruction Set Computers)
Rich register set	Less number of registers
Supports less addressing modes	Supports more number of addressing modes
Supports fixed length instruction	Supports variable length instruction
Successful pipeline with one instruction per cycle	Unsuccessful pipeline
Example: ARM, Motorola	Example: Pentium processors

2.6 I/O INTERFACE (INTERRUPT AND DMA MODE)

I/O interface bridges the differences between CPU and peripheral devices and provides a method for transferring information between internal storage and external I/O devices. There are the following three modes of I/O transfer:

1. **Programmed I/O:** The I/O device does not have direct access to memory. It requires execution of several instructions by the CPU and the CPU has to wait for the I/O device to be ready for either reception or transmission of data.
2. **Interrupt initiated I/O:** In this, instead of waiting, the control is transferred from a currently running program to another service program as a result of an external/internal generated request.
   - **Hardware interrupts:** These interrupts are present in the hardware pins.
   - **Software interrupts:** These are the instructions used in the program whenever the required functionality is needed.
   - **Maskable interrupts:** These interrupts may be enabled or disabled explicitly.
   - **Non-maskable interrupts:** These interrupts are always there in the enable state. We cannot disable them by explicit conditions (flags).
   - **Vectored interrupts:** These interrupts are associated with the static vector address.
   - **Non-vectored interrupts:** These interrupts are associated with dynamic vector address.
   - **External interrupts:** These interrupts are generated by external devices such as I/O.
   - **Internal interrupts:** These devices are generated by the internal components of the processor such as temperature sensor, power failure, error instruction, etc.

- **Synchronous interrupts:** These interrupts are controlled by the fixed time interval. All the interval interrupts are called as synchronous interrupt.
- **Asynchronous interrupts:** These interrupts are initiated based on the feedback of previous instructions. All the external interrupts are called as asynchronous interrupt.

3. **Direct memory access (DMA):** It is one of several methods for coordinating the data transfers between an I/O device and the core processing unit or memory in a computer. It refers to transfer of data directly between a fast storage device and memory bypassing CPU because of its limited speed. DMA provides a significant improvement in terms of latency and throughput as it allows the I/O device to access the memory directly, without using the processor. There are certain advantages of using DMA for data transfer:
   - DMA saves processor's MIPS as the core can operate in parallel.
   - DMA saves power because it requires less circuitry than the processor to transfer data.
   - DMA has no modulo block size restrictions.

   Direct memory access (DMA) controller takes over the control of buses to manage the transfer directly between the I/O device and memory. Bus request (BR) and Bus grant (BG) signals are used by the DMA controller to request the CPU to relinquish control of the buses and get the control of system buses (Fig. 2.7). The DMA controller consists of 3 different registers: an address register, a control register and a word counter register. To transfer a block of data between an I/O device and memory, the controller stores initial values in the address register. The DMA channel then transfers the block of information from or to memory according to the control register. The starting address of the block in memory is given by the address register, and the length of the bytes to transfer is given by the word count register. The controller decrements a word counter each time it moves a data byte.

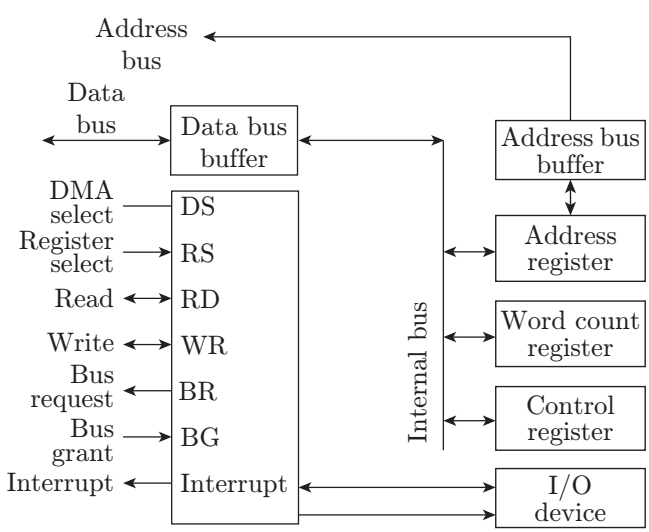


Figure 2.7 | Block diagram of the DMA controller.

There are several modes of operation of DMA:

- **Burst or block transfer mode:** In this mode, the entire block of data is transferred once the DMA controller is granted access to the system bus by the CPU. The bytes of data in the block are transferred before releasing control of the system buses back to the CPU. The only disadvantage of this mode is that it renders the CPU inactive for some long periods of time.
- **Cycle stealing mode:** In this mode, the DMA controller obtains access to the system buses like burst mode; but after one byte of data transfer, the control of the system bus is released back to the CPU via BG. It is then continually requested again via BR, transferring one byte of data per request, until the entire block of data has been transferred. This mode is suitable for the systems in which the CPU cannot be disabled for the considerable length of time as in burst transfer modes such as for controllers monitoring the data in real time. The advantage is that CPU is not idled for as long as in burst mode, but the data block is not transferred as quickly.
- **Transparent mode:** It is the slowest yet more efficient data transfer mode in terms of overall system performance. The DMA controller transfers data only when the CPU is busy in performing operations that do not use the system buses. So, the CPU never stops executing its programs but the biggest disadvantage is complex hardware circuitry that needs to determine when the CPU is not using the system buses.

A DMA read transfers data from the memory to the I/O device, while DMA write transfers data from an I/O device to memory. The functional behaviour of a DMA transfer outlined in Fig. 2.8:

- The CPU transmits the following information to a DMA controller:
  (a) beginning address in memory which is stored in address register in DMA controller.
  (b) Number of words to transfer which is stored in word count register in DMA Controller.
  (c) direction (memory-to-I/O device or I/O device-to-memory), port ID, DMA mode of transfer and end of block transfer either through interrupt request or no interrupt request which is stored in control register as command word.
- The processor them relinquishes control of address, data and control buses to DMA Controller and returns to other processing activities while the DMA controller starts the data transfer between I/O device and memory.

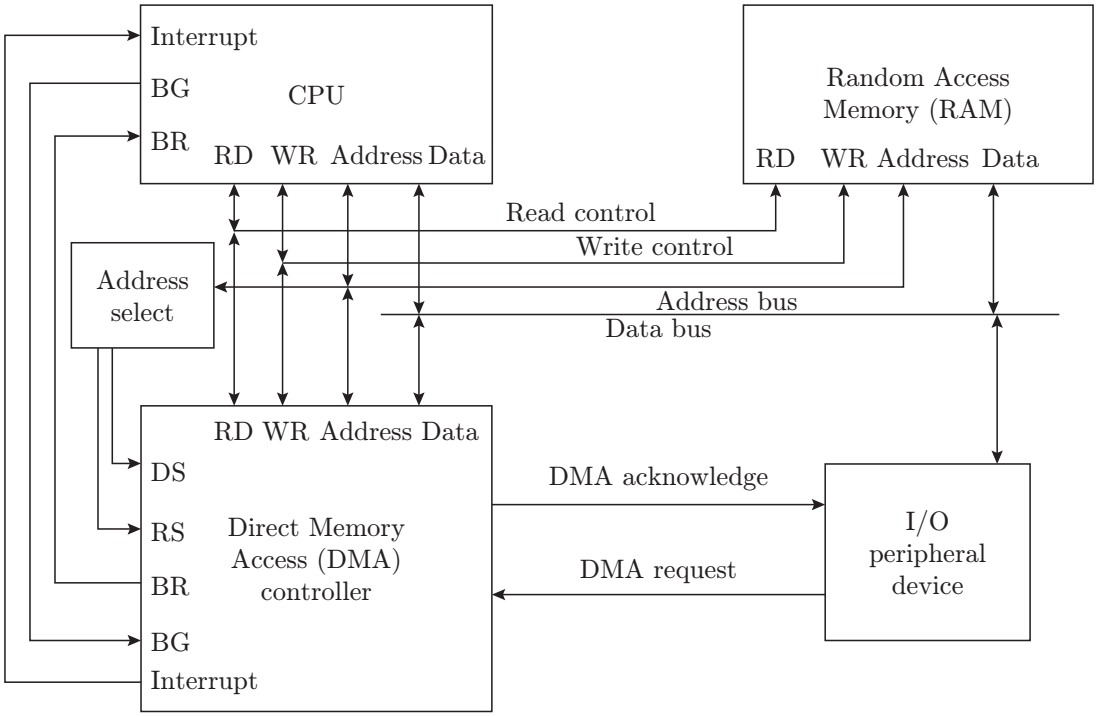


Figure 2.8 | DMA controller interconnection with memory, CPU and I/O devices.

- When the DMA controller accesses memory, it synchronizes this memory request with an idle period of the processor, thus disabling the processor, or requesting a halt of the processor, and awaits an acknowledgement.
- After the completion of the block transfer, the DMA controller either raises an interrupt request if the interrupts are enabled or indicates the completion in its status register and the processor recognizes I/O completion (either by interrupt signal or by reading the status register) and gets its system buses back and normal processing starts. The device has to initiate a new data transfer through DMA request signal which is again acknowledged by CPU through DMA acknowledge signal via DMA controller.

2.7 INSTRUCTION PIPELINING

In early computers, each instruction completely finished before the execution of the next one began. The hardware circuits needed to perform different operations of an instruction cycle are different and most part of these processor circuits are idle at a given moment of time. These processor circuits wait for the other parts of the processor to complete its part of execution first. Instruction pipelining is a technique for overlapping the execution of several instructions to reduce the overall execution time of a set of instructions and there is no need to wait of the most part of the processor circuits for the other parts of the processor to complete their part of execution. Pipeline speed is limited by the slowest pipeline stage.

Throughput of a processor is the rate at which operations get executed. Latency is the amount of time that a single operation takes to execute. In an unpipelined computer, throughput = 1/latency, as each operation executes by itself and for pipelined computer, throughput > 1/latency, since execution of instruction is overlapped.

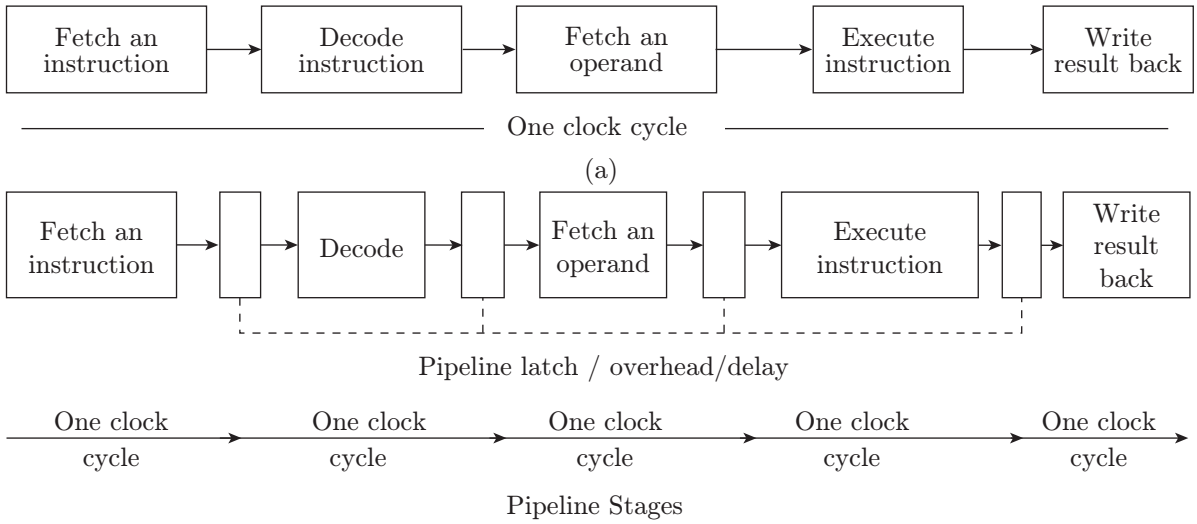
Consider a k-segment pipeline with a clock cycle time T_p used to execute n tasks (Fig. 2.9). An equivalent non-pipelined system takes T_n time to complete each task. The speed up of a pipelined system over a non-pipelined system is given by the following relation:

$$S = \frac{n \times T_n}{(k + n - 1) \times T_p}$$

Theoretically, maximum speed up that a pipelined system can achieve is given by the following equation:

$$S = \frac{kT_p}{T_n} = k$$

Pipelining Hazards: These hazards reduce the ideal speed up gained by pipelining by preventing the next instruction in the sequence from being executing during its designated clock pulse. Hazards forces the pipeline to be stalled. There are three types of hazards:



step →	1	2	3	4	5	6	7	8
Segment ↓ Fetch	I_1	I_2	I_3	I_4				
Decode		I_1	I_2	I_3	I_4			
Fetch operand			I_1	I_2	I_3	I_4		
Execute				I_1	I_2	I_3	I_4	
Write back					I_1	I_2	I_3	I_4

(c)

Figure 2.9 (a) Unpipelined processor. (b) Pipelined five-stage processor. (c) Timing diagram of a five-stage instruction pipeline.

1. **Structural hazards:** These result from resource conflicts when the hardware cannot support instructions that need simultaneous execution in pipeling.
2. **Data hazards:** They arise when an instruction depends on the result of a previous instruction and that result is not yet calculated.

   There are three situations in which data hazards can occur:

   - Read after write (RAW), a true dependency
   - Write after read (WAR), an anti dependency
   - Write after write (WAW), an output dependency

   Consider two instructions `inst1` and `inst2` occurring, with `inst1` occurring before `inst2` in the program order.

   - **Read after write (RAW):** A read after write (RAW) data hazard is a situation in which an instruction **refers** to a result which is yet not been calculated, that is, in this `inst2` tries to read a source before `inst1` writes to it. This situation arises if the read operation by instruction takes place before write done by other instruction. For example,

```
inst1: R3 <-R1 + R2
inst2: R4 <-R3 + R2
```

The first instruction calculates a value by adding values in registers R1 and R2 and saves the result in register R3, and the second instruction uses this saved value to calculate a result for register R4. However, in a pipeline, when operands for the second operation are fetched, the results from the first instruction will not have been saved yet, and so there arises a data dependency. It can be said that there is a data dependency with instruction `inst2`, as it is dependent on the completion of instruction `inst1`.

- **Write after read (WAR):** A write after read (WAR) data hazard refers to a situation in which there is a problem with concurrent execution, that is, `inst2` tries to write a destination before it is read by `inst1`. This situation arises if write operation completes first by instruction before the read operation takes place by other instruction. For example,

```
inst1: R4 <-R1 + R3
inst2: R3 <-R1 + R2
```

  If a situation arises in which there is a chance that `inst2` may get completed before `inst1` (i.e., with concurrent execution) we must note that we do not store the result of register `R3` before inst1 has had a chance to fetch the operands.

- **Write after write (WAW):** A write after write (WAW) data hazard refers to a situation in which there is a concurrent execution environment, that is, `inst2` tries to write an operand before it is written by `inst1`.This situation arises if write operation by an instruction occurs in the reverse order of the intended sequence. For example,

```
inst1: R2 <-R1 + R3
inst2: R2 <-R4 + R5
```

  The WB (write back) of `inst2` must be delayed until the execution of `inst1`.

3. **Control hazards:** They arise from the pipelining of branches and other instructions that change the value of PC.

Speed up from pipelining

$$= \frac{\text{Average instruction time unpipelined}}{\text{Average instruction time pipelined}}$$

Speed up from pipelining

$$= \frac{\text{CPI unpipelined} \times \text{Clock cycle pipelined}}{\text{CPI pipelined} \times \text{Clock cycle pipelined}}$$

$$\text{Ideal CPI} = \frac{\text{CPI unpipelined}}{\text{Pipeline depth}}$$

Speed up from pipelining

$$= \frac{\text{Ideal CPI} \times \text{Pipeline depth} \times \text{Clock cycle unpipelined}}{\text{CPI pipelined} \times \text{Clock cycle pipelined}}$$

Speed up from pipelining

$$= \frac{\text{Ideal CPI} \times \text{Pipeline depth} \times \text{Clock cycle unpipelined}}{(\text{Ideal CPI} + \text{Pipeline stall}) \times \text{Clock cycle pipelined}}$$

Assuming ideal CPI as 1, speed up is:

Speed up from pipelining

$$= \frac{\text{Pipeline depth} \times \text{Clock cycle unpipelined}}{(1 + \text{Pipeline stall}) \times \text{Clock cycle pipelined}}$$

where CPI is cycles per instruction.

Problem 2.7: Consider a four-stage pipeline processor. The number of cycles needed by the four instructions I_1, I_2, I_3 and I_4 in stages instruction fetch, decode, operand fetch and execute are shown below. Assume I_2 is the branch instruction. Draw the timing space diagram.

	S_1	S_2	S_3	S_4
I_1	2	1	1	1
I_2	1	2	3	1
I_3	1	1	1	2
I_4	2	1	3	1

Solution:

STEP →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Fetch	I_1	I_1	I_2	I_3	–	–	–	–	–	I_3	I_4	I_4					
Decode			I_1	I_2	I_2	–	–	–	–	–	I_3	–	I_4				
Operand Fetch				I_1		I_2	I_2	I_2	–	–	–	I_3	–	I_4	I_4	I_4	
Execute					I_1	–	–	–	I_2	–	–	–	I_3	I_3	–	–	I_4

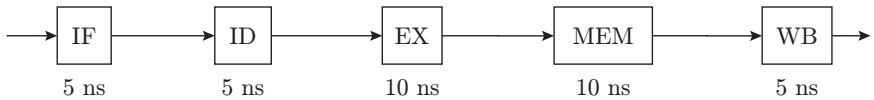
Problem 2.8: Assume a simple 5-stage pipeline (IF, ID, E, DF, W) each stage takes a single cycle. Assuming there are no cache misses. How many cycles would the following code take to execute if there is no special hardware to improve performance in the presence of hazards?

```
MOV   edx,[ecx+100]
MOV   ebx,[ecx+104]
ADD   edx,ebx
MOV   [ecx+108],ebx
MOV   eax,[ecx+100]
ADD   ebx,eax
```

Solution: The above code takes 14 cycles to execute, as shown below:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
IF	ID	DF	E	W									
	IF	ID	DF	E	W								
		IF	ID	DF	stall	E	W						
			IF	ID	stall	DF	stall	W					
				IF	ID	stall	DF	stall	stall	E	W		
					IF	ID	stall	DF	stall	stall	stall	E	W

Problem 2.9: In the below figure, calculate the total execution time after which the result of the fourth task entering the pipe above ready?



Solution:

	5	10	15	20	25	30	35	40	45	50	55	60	65
Inst1	IF	ID	EX	EX	MEM	MEM	WB						
Inst2		IF	ID		EX	EX	MEM	MEM	WB				
Inst3			IF	ID			EX	EX	MEM	MEM	WB		
Inst4				IF	ID				EX	EX	MEM	MEM	WB

Therefore, the total execution time is 65 ns.

Problem 2.10: What is the mean overhead of a pipeline with 8 stages and an execution time per stage of 2 ns?

Solution: The mean overhead = (Stages − 1) × Execution time per stage = $(8-1)\times 2 = 7\times 2 = 14$ ns

Problem 2.11: How many stages has a pipeline that achieves a speed of 9.9 for 100 operations?

Solution:

$$\text{Speed} = \frac{n\times k}{k-1+n} \Rightarrow 9.9 = \frac{n\times 90}{(90-1)+n} \Rightarrow n = 11$$

Problem 2.12: Calculate the time required to perform 1000 operations in a 6-staged pipeline with an execution time of 3 ns per stage?

Solution:

$$T_p = (k-1+n)\times T = (6-1+1000)\times 3 = 3.015\ \mu\text{s}$$

Problem 2.13: Calculate the mean overhead of a pipeline with 7 stages and an execution time per stage of 2 ns?

Solution: Mean overhead of pipeline =

$$\frac{(k\times T_p - T_n)}{k} = (k-1)\times T = (7-1)\times 2 = 12\ \text{ns}$$

Problem 2.14: Consider a pipeline with 5 stages: IF, ID, EX, M and W. Assume that each stage requires one clock cycle. Show how the following program segment for adding 2 arrays is processed and compare the clock cycles needed in non-pipelined system with pipelined system when result of the branch instruction i.e. content of is available after WB stage.

```
LOAD R4 #400
L1: LOAD R1, 0 (R4);
LOAD R2, 400 (R4);
ADD R3, R1, R2;
STORE R3, 0 (R4);
SUB R4, R4, #4;
BNEZ R4, L1;
```

Solution: Number of cycles = [Initial instruction + (Number of instructions in the loop L1) × Number of loop cycles] × Number of clock cycles/instruction (CPI)

$$= [1 + (6) \times 400/4] \times 5 = 3005$$

Timing diagram for one loop iteration in a pipelined system is as follows:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LOAD R4 #400	IF	ID	EX	M	W										
LOAD R1, 0 (R4)		IF	ID	EX	M	W									
LOAD R2, 400 (R4)			IF	ID	stall	stall	EX	M	W						
ADD R3, R1, R2				IF	ID	stall	stall	EX	stall	M	W				
STORE R3, 0 (R4)					IF	ID	stall	DF	stall	stall	E	W			
SUB R4,R4, #4							IF	ID	stall	Ex	M	W			
BNEZ R4, L1								IF	stall	ID	stall	stall	EX	M	W

Number of cycles in the loop = 15

Number of clock cycles for segment execution on pipelined processor

$$= 1 + (\text{Number of clock cycles in the loop L1}) \times \text{Number of loop cycles}$$

$$= 1 + 15 \times 400/4 = 1501$$

$$\text{Speedup} = \frac{\text{Number of Clock cycles for the program execution on non-pipelined processor}}{\text{Number of Clock cycles for the segment execution on pipelined processor}}$$

$$= \frac{3005}{1501} = 2 \text{ times}$$

Problem 2.15: Consider a 5-stage pipeline with stages: For all following questions we assume that: (a) Pipeline contains stages: IF (Instruction Fetch), IS (Issue), FO (Fetch operand), E (Execute) and W (Write). (b) Each stage except E requires one clock cycle and system has 4 Functional Units for floating point operations, FP load/store, FP addition/subtraction, FP multiplication and FP division, (c) Execution stage for Load/Store operations requires 1 clock cycle, for ADD or SUB operations requires 1 clock cycle, for MUL operation requires 3 clock cycles and for DIV operation requires 4 clock cycles. All memory references hit in cache. Pipeline has forwarding circuitry for all FUs, except FP-Load/Store where operand is ready after W-stage.

Timing diagram of is presented below:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LOAD F6, 20(R5)	IF	IS	FO	E	W											
LOAD F2, 28(R5)		IF	ISD	FO	E	W										
MUL F0, F2, F4			IF	IS	stall	stall	FO	E	E	E	W					
SUB F8, F6, F3				IF	IS	FO	E	W								
DIV F10, F0, F6					IF	IS	stall	stall	stall	stall	FO	E	E	E	E	W
ADD F6, F8, F2						IF	IS	FO	E	W						
STORE F8, 50(R5)							IF	IS	FO	E	W					

Identify the hazards in the following instructions from the following list (Structural, Data, Control, RAW, WAR, WAW, None)

1. MULT F0, F2, F4 and STORE F8, 50(R5)
2. DIV F10, F0, F6 and ADD F6, F8, F2
3. MULT F0, F2, F4 and DIV F10, F0, F6
4. DIV F10, F0, F6 and ADD F6, F8, F2

Solution: 1. Structural; 2. Data; 3. RAW; 4. WAR.

2.8 MEMORY HIERARCHY

The storage media can be categorized in hierarchy according to their speed and cost (Fig. 2.10). As we move down the hierarchy, access time increases and cost per bit decreases.

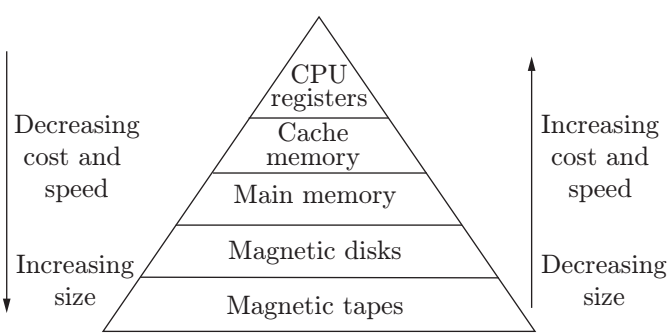


Figure 2.10 Memory hierarchy.

2.8.1 Main Memory

It is the central storage unit that directly communicates with the CPU. It is designed using semiconductor-integrated circuits and needs constant power supply to maintain the information. It is expensive as compared to auxiliary storage so it has limited capacity. Example: R/W (read/write) memory or RAM (random access memory) and ROM (read only memory). Integrated RAM chips are available in two modes:

1. **Static RAM:** It stores the binary information in flip flops and information remains valid until power is supplied. It has faster access time and is used in implementing cache memory.
2. **Dynamic RAM:** It stores the binary information as a charge on the capacitor. It requires refreshing circuitry to maintain the charge on the capacitors after few milliseconds. It contains more memory cells per unit area as compared to SRAM.

2.8.1.1 Memory Interfacing

If the required memory for the computer is larger than the capacity of one chip, it is necessary to connect multiple RAM and ROM chips to a CPU through the data and address buses (Fig. 2.11). The low-order address bus lines select the word within a chip and other lines select a particular chip through its chip select inputs. Assume a computer system needs 256 bytes of RAM and 512 bytes of ROM. The configuration of RAM chip is 128×8 and ROM chip is 512×8. The RAM and ROM chips required are as follows:

$$\text{Number of RAM chips} = 256/128 = 2$$
$$\text{Number of ROM chips} = 512/512 = 1$$

The memory interconnection is depicted in the following diagram:

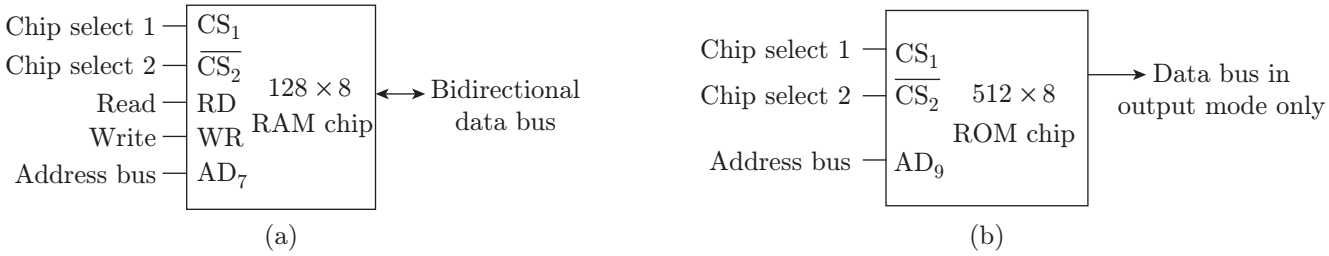


Figure 2.11 (a) RAM chip. (b) ROM chip.

Problem 2.16: A computer employs RAM chips of 256×8 and ROM chips of 1024×16. The computer system needs 2K bytes of RAM and 4K bytes of ROM and four interface units each with four registers. Draw a memory address map for the system and give the address range in hexadecimal for RAM and ROM chips.

Solution: RAM 2048/256 = 8 chips; 2048 = 211; 256 = 28

ROM 4096/1024 = 4 chips; 4096 = 212; 1024 = 210

Interface $4 \times 4 = 16$ registers; 16 = 24

Component	Address	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
RAM	0000-O7FF	0	0	0	0	0	↔	x	x	x	x	x	x	x	x		
								3×8 decoder									
ROM	4000-4FFF	0	1	0	0	↔	x	x	x	x	x	x	x	x	x	x	
								2×4 decoder									
Interface	8000-800F	1	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x

2.8.2 Secondary Memory

Secondary memory, also known as auxiliary memory or external memory, can store a large amount of data at lesser cost per byte than the main memory. They are non-volatile in nature, that is, data is not lost when the device is powered off. The most common auxiliary storage devices used in consumer systems are flash memory, optical disks and magnetic disks.

1. **Flash memory:** Flash memory is an electronic non-volatile fastest computer storage device that can be electrically erased and reprogrammed. Example: flash drives and solid state drive.
2. **Optical disk:** Optical disks are low-cost mass storage devices from which read and write operations are performed using laser technology. Optical disks can store huge amounts of data up to 6 GB (6 billion bytes). Different types of optical disks are CD-ROM (compact disk read-only), WORM (write-once read-many), EO (erasable optical disks) and DVD.
3. **Magnetic disk:** A magnetic disk is composed of a circular platter made of metal or plastic and coated with magnetized material on both sides. Multiple disks are stacked over one another on the spindle with read/write heads on each surface. Bits are stored as spots on magnetized surface along concentric circles called tracks. Tracks are further divided into wedge-shaped sectors.
4. **Magnetic tapes:** It consists of tape made up of plastic covered with magnetic oxide coating. Tapes are mounted on reels. Bits are recorded as magnetic spots on tape along several tracks. R/W heads are mounted in each track so that data can be recorded and read as a sequence of characters. Seven or nine bits are recorded to form a character together with a parity bit. Data is recorded in contiguous blocks separated by inter-record gaps.

2.8.3 Cache Memory

It is a special memory that compensates the speed mismatch between processor and main memory access time. It temporarily stores frequently used instructions and data for faster processing by the CPU. Cache hit ratio is calculated to measure its performance. If a data

item requested by the CPU is found in cache it is called hit otherwise it is a miss. Hit ratio is defined as ratio of number of hits divided by total CPU references to memory.

$$\text{Hit ratio } (h) = \frac{\text{Number of hits}}{\text{Number of hits} + \text{Number of misses}}$$

$$\text{Average access time} = \text{Hit ratio} \times T_c + (1 - \text{Hit ratio})(T_c + T_m)$$

where T_c is cache access time and T_m is the main memory access time.

2.8.3.1 Elements of Cache Design

The various elements of cache design are as follows:

1. **Cache size:** It should be optimum, small enough to keep average cost per bit close to the main memory and large enough to keep overall average access time close to the cache memory.
2. **Mapping function:** It describes the mapping of main memory block to cache block. There are three different mapping techniques: fully associative, direct mapped and set associative cache organization.
3. **Replacement algorithm:** When a new memory block is required in cache, one of the existing blocks must be replaced by a new block. Example: FIFO (first in, first out), LRU (least recently used).
4. **Write policy:** Cache memory follows write-through and write-back updating policies. In write-through policy, cache controller copies data immediately to main memory as data is written in cache. The data in main memory is always valid, but this approach reduces system performance. In write back, update to memory block is delayed until the updated cache block is replaced by a new block.

2.8.4 Cache Mapping Techniques

The cache memory can store a reasonable number of blocks, but this number is always small as compared to blocks in the main memory to keep average cost per bit low. The correspondence between memory blocks and cache block is specified by the following mapping techniques. Consider a cache memory consisting of 2K words with 128 blocks of 16 words each. Number of bits required to address a cache block is 11 bits. Main memory has 64K words and bits required to address is 16 (Fig. 2.12).

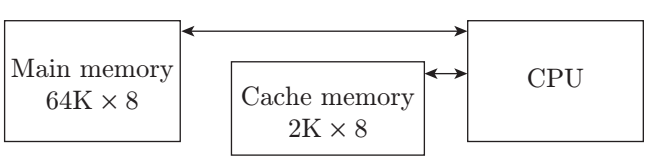


Figure 2.12 | Cache mapping example.

2.8.4.1 Direct Mapping

In this technique, each block from the main memory has only one possible location in the cache memory. In this example, say a block from main memory maps onto a block (i mod 128) of the cache. If there are 2^n words in the cache memory and 2^m words in the main memory, then m-bit main memory address is divided into two fields: n bits for index field to access the cache and $(m - n)$ bits for the tag field. Each word in cache consists of the data and the associated tag. Whenever a new block is brought into cache, tag is stored along with data bits. Index field is further divided into block and word if there are multiple words (say k) in a block. The lower k bits select one of the k words in a block known as word field. The block field is used to distinguish a block from other blocks.

Tag $(m - n)$ bits	Index (n bits)

Tag $(m - n)$ bits	Block $(n - k)$ bits	Word (k bits)

When CPU generates a memory request, the block field points to a particular block location in the cache. The high-order tag field is compared with tag bits associated with that cache location. If they match, then the desired word is in that block of cache. If there is no match, then the block containing the required word must be loaded to cache first (Fig. 2.13).

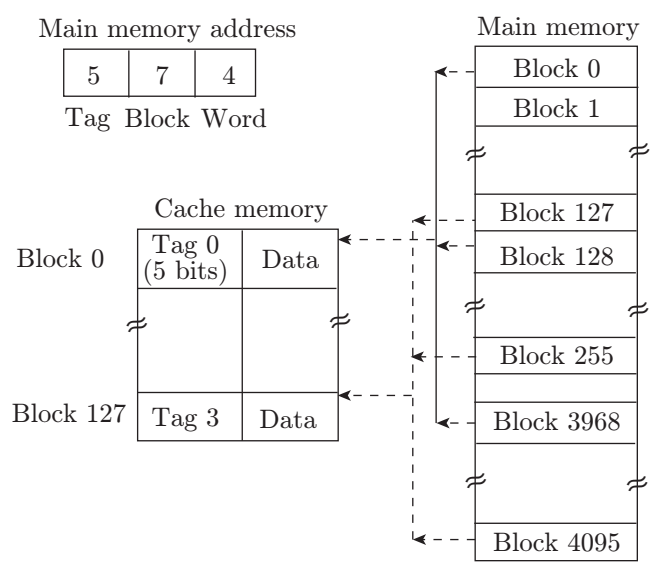


Figure 2.13 | Direct mapped cache organization.

The demerit of direct mapping is that hit ratio drops considerably if two or more words having same index and different tags are accessed consecutively one after the other.

2.8.4.2 Fully Associative Mapping

In this technique, a main memory block can be placed into any cache block location. It is the most flexible cache organization. The main memory address is divided into

two fields: word and tag. The associative memory stores both the address (tag) and data of the main memory. Figure 2.14 shows the mapping of different blocks into cache. High-order 12 bits of CPU address is placed in the argument register of the associative memory and compared to tag bits of each block of the cache to see if the desired block is present. Once the desired block is present, 4-bit word is used to extract necessary word from the cache.

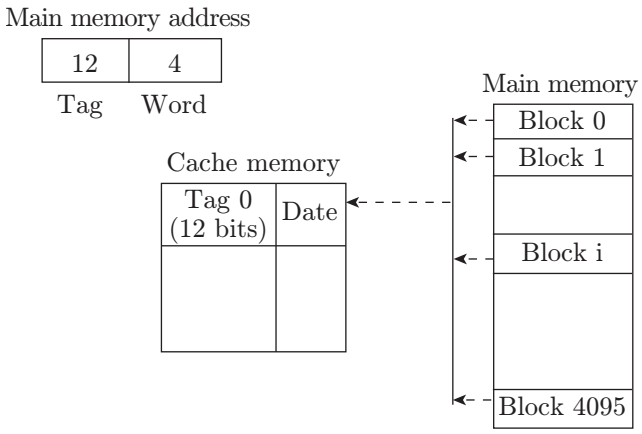


Figure 2.14 | Associative mapped cache organization.

It is necessary to compare high-order bits of main memory with all tag bits corresponding to each block to find whether a given block is present in cache, so it is the most expensive.

2.8.4.3 Set-Associative Mapping

As fully associative mapping is an expensive solution and direct mapping does not allow words with same index but different tag to exist in cache, set associative mapping is a combination of both. It is an improvement over direct mapping where contention problem is solved by having several choices for block placement. The figure below shows two-way set associative cache because each block of main memory has two choices for block placement in cache. A block i in the main memory can be in any block belonging to set i mod S of cache, where S is the number of sets. The block 0, 64, 128, … and so on of main memory can map into any of the two blocks in set 0.

The main memory address is divided into three fields: low-order bits for word field, set field to determine the desired block from all possible sets and high-order bits for the tag field. Each word in cache consists of data and the associated tag.

Tag	Set	Word

When the CPU generates a memory request, the set field points to a particular set of the cache which might contain the desired block. The high-order tag field is then compared associatively to the tags corresponding to the matched set. If a match occurs, the corresponding word is read from cache else main memory is referred and block containing that word is brought into cache for future reference (Fig. 2.15).

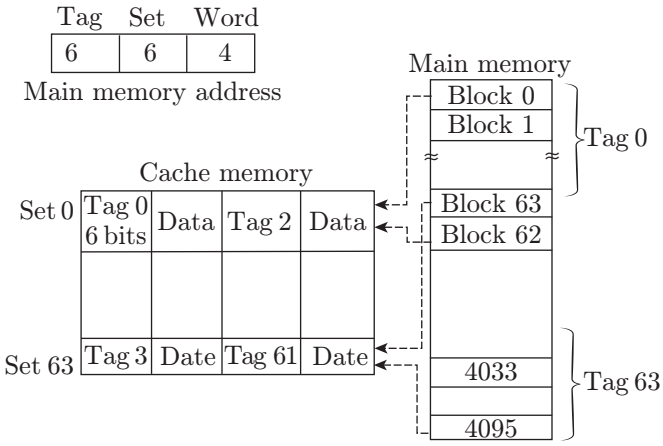


Figure 2.15 | Set-associative mapped cache organization.

Problem 2.17: Consider a memory hierarchy system containing a cache, a main memory and a virtual memory. Assuming, cache access time of 5 ns, and 80% hit ratio . The access time of the main memory is 100 ns, and it has a 99.5% hit rate. The access time of the virtual memory is 10 ms. Calculate the average access time of the memory hierarchy.

Solution: As we know, the hit rate of virtual memory is 100%, the average access time for requests that reach the main memory as (l00 ns × 0.995) + (10 ns × 0.005) = 50,099.5 ns. Given this, the average access time for requests that reach the cache is (5 ns × 0.80) + (50,099.5 ns × 0.20) = 10,024 ns.

Problem 2.18: A computer uses RAM chips of 1024 × 1 capacity.

(a) How many chips are needed to provide a memory capacity of 16K bytes?
(b) How many of these lines will be common to all chips?

Solution:

(a) Chips are needed to provide a memory capacity of 16K bytes = 16 × 8 = 128 chips
(b) Using 14 address lines (16K = 2^{14}), we have 10 lines specifying the chip address which is common to all chips.

Problem 2.19: Consider a 2-way set associative cache consisting of 256 blocks of 8 words each, and assume that the main memory is addressable by 16-bit address and it consists of 4K blocks. Calculate the number of bits in each of the TAG, BLOCK/SET and word fields for different mapping techniques?

Solution: For direct mapping, word field is of 3 bits to identify 8 different words in a block ($2^3 = 8$). As cache memory consists of 256 blocks so ($2^8 = 256$) 8 bits are required to address a block because there is one-to-one correspondence of block k in main memory to block (k mod 256) in cache memory. The remaining 5 (16 – 8 – 3) high-order address bits are tag bits. Thus, the main memory address for direct mapping is divided as follows:

Tag	Block	Word
5 bits	8 bits	3 bits

For fully associative mapping, number of word bits are same, that is, 3 bits. Cache memory stores both tag and data. The high-order tag bits of an address generated by CPU are compared with tag bits of each block so number of block bits is zero. All remaining bits (except word bits) are identified as tag bits. Thus, the main memory address for fully associative mapping is divided as follows:

Tag	Word
13 bits	3 bits

For two-way set-associative mapping, cache memory is mapped with two blocks per set. The word field has same number of bits, 3 bits. There are 128 (256/2) sets so 7 bits are required to uniquely identify these 128 sets. The remaining 6 (16 – 7 – 3) bits are used as tag bits. Thus, the main memory address for set-associative mapping is divided as follows:

Tag	Set	Word
6 bits	7 bits	3 bits

Problem 2.20: The access time of a cache memory is 200 ns and that of main memory is 2000 ns. It is estimated that 70% of the memory requests are for read and remaining 30% for write. The hit ratio for read accesses only is 0.9. A write-through procedure is used.

(a) What is the average access time of the system considering only memory read cycles?
(b) What is the average access time of the system for both read and write requests?
(c) What is the hit ratio taking into consideration the write cycles also?

Solution:

(a) Average access time = $0.9 \times 200 + 0.1 \times 2200 = 180 + 220 = 400$ ns
(b) Average access time = $0.3 \times 2000 + 0.7 \times 400 = 600 + 280 = 880$ ns
(c) Hit ratio = $0.7 \times 0.9 = 0.63$

Problem 2.21: A 4-way set-associative cache memory uses blocks of four words. The cache can accommodate a total of 1024 words from the main memory. The main memory size is 128K × 32.

(a) Formulate all pertinent information required to construct the cache memory.
(b) What is the size of the cache memory?

Solution:

(a) Main memory is 128K = 2^{17} so 17 bits address is generated. For a set size of 4, the number of sets in cache is 256/4 = 64, so 6 bits are required. Each block consists of 4 words so 2 bits are required for the word field.

Tag	Set	Word
9 bits	6 bits	2 bits

(b) Since cache is 4-way set associative, 4 blocks per set are stored in cache memory.

	Tag 1 9 bits	Data 1 32 bits	Tag 2 9 bits	Data 2 32 bits	Tag 3 9 bits	Data 3 32 bits	Tag 4 9 bits	Data 4 32 bits
Set 0 -	Tag	Data	Tag	Data	Tag	Data	Tag	Data
Set 128								

Size of cache memory is $128 \times 4(9 + 32) = 5248$ bits.

Problem 2.22: Suppose physical memory is of 2GB and each word is of 16 bits. There is a cache containing 2K words of data, and each cache block contains 16 words. For each of the direct mapped and 2-way set associative cache configurations, specify how the address would be partitioned.

Solution:

(a) **For direct mapping**, the word field is of 4 bits identify 16 different words in a block ($2^4 = 16$). As the cache memory consists of 2K words which is equivalent to 2K/16 = 128 blocks so ($2^7 = 128$) 7 bits are required to address a block. The remaining 20 (31 − 7 − 4) high-order address bits are tag bits. Thus, the main memory address for direct mapping is divided as follows:

20 bits	7 bits	4 bits
Tag	Block	Word

(b) Since cache is 2-way set associative, 2 blocks per set are stored in cache memory. So, the number of sets is 128/2 = 64.

21 bits	6 bits	4 bits
Tag	Set	Word

Problem 2.23: Consider a direct mapped cache of size 32 KB with block size 32 bytes. The CPU generates 32-bit addresses. What are number of bits needed for addressing block in cache and number of tag bits?

Solution:

Tag	Block	Word
32 − 10 − 5 bits	10 bits	5 bits

The number of bits needed for addressing block in cache and number of tag bits are 10, 17, respectively.

IMPORTANT FORMULAS

- Amdahl's law

$$S_{\text{overall}} = \left[(1 - \sum F_i) + \sum \frac{F_i}{S}\right]^{-1}$$

- The speed up of pipelined system over non-pipelined system is given by:

$$S = \frac{n \times T_n}{(k + n - 1) \times T_p}$$

- Maximum speed up that a pipelined system can achieve is given by:

$$S = \frac{kT_p}{T_n} = k$$

- Hit ratio (h)

$$= \frac{\text{Number of hits}}{\text{Number of hits} + \text{Number of misses}}$$

- Average access time

$$= \text{Hit ratio} \times T_c + (1 - \text{Hit ratio})(T_c + T_m)$$

where T_c is cache access time and T_m is main memory access time.

- Direct mapping

Tag ($m - n$) bits	Block ($n - k$) bits	Word (k bits)

- Set-associative mapping

Tag	Set	Word

SOLVED EXAMPLES

1. The principle of locality is used in

(a) Interrupt (b) Registers
(c) DMA (d) Cache memory

Solution: It is used in cache memory to help the program access small amounts of address space at any instant.

Ans. (d)

2. Which memory unit has lowest access time?

(a) Cache (b) Registers
(c) Optical disk (d) Main memory

Solution: Registers are used for processing and manipulating data and for holding memory addresses that are available to the machine-code programmer. So, they have lowest access time.

Ans. (b)

3. During DMA transfer, the DMA controller takes over the buses to manage the transfer

(a) Directly from CPU to memory
(b) Directly from memory to CPU
(c) Directly between the memory and registers
(d) Directly between the I/O device and memory

Solution: DMA controller manages transfer between I/O device and memory.

Ans. (d)

4. Booth's algorithm is used for the arithmetic operation of

(a) addition. (b) subtraction.
(c) multiplication. (d) division.

Solution: It is a multiplication algorithm that multiplies two signed binary numbers in 2's complement notation.

Ans. (c)

5. The reason for improvement in CPU performance during pipelining is

(a) reduced memory access time.
(b) increased clock speed.
(c) introduction of parallelism.
(d) increase in cache memory.

Solution: Instruction-level parallelism is implemented within a single processor to allow faster CPU throughput.

Ans. (c)

6. Use of cache memory enhances

(a) I/O access time
(b) memory access time.
(c) effective memory access time.
(d) secondary storage access time.

Solution: Cache memory compensates the speed mismatch between processor and main memory access time.

Ans. (c)

7. An instruction cycle refers to

(a) fetching an instruction.
(b) executing an instruction.
(c) fetching, decoding and executing an instruction.
(d) reading and executing an instruction.

Solution: It involves fetching, decoding and executing the instruction.

Ans. (c)

8. A hardware interrupt is also called

(a) an internal interrupt.
(b) an external interrupt.
(c) a processor interrupt.
(d) a clock interrupt.

Solution: Hardware interrupt is present in hardware pins.

Ans. (b)

9. Priority is provided by _____ for access to memory by various I/O channels and processors.

(a) a register
(b) a counter
(c) the processor scheduler
(d) a controller

Solution: Controller sets priority for memory access by various I/O devices and processes.

Ans. (d)

10. By applying the principle of temporal locality, processes are likely to reference pages that _____

(a) have been referenced recently.
(b) are located at address near recently referenced pages in memory.
(c) have been preloaded into memory.
(d) have to be reloaded into memory.

Solution: Temporal locality refers to reuse of resources referenced within a short time frame.

Ans. (a)

11. Which of the following is a correct statement related to L2 cache memory?

(a) The level 1 cache is always faster than the level 2 cache.
(b) The level 2 cache is used to mitigate the dynamic slowdown every time a level 1 cache miss occurs.
(c) Level 2 cache comes as on board only.
(d) In modern day computer, the level 2 cache is considered an internal cache.

Solution: L2 level of cache is placed between the L1 and RAM. The L1 cache is always the fastest.

Ans. (a)

12. What is the control unit's function in the CPU?

(a) To decode program instructions
(b) To transfer data to primary storage
(c) To perform logical operations
(d) To store arithmetic operations

Solution: Control unit controls several units of CPU and helps decode program instructions.

Ans. (a)

13. CPU fetches the data and instructions from _____

(a) ROM (b) control unit
(c) RAM (d) coprocessors chip

Solution: The CPU fetches data and instructions from RAM and executes it.

Ans. (c)

14. Which of the following affects the processing power of the CPU?

(a) Data bus, addressing schemes
(b) Clock speed, addressing schemes
(c) Clock speed, data bus
(d) Clock speed, data bus, addressing schemes

Solution: CPU processing speed is affected by clock cycle, data bus for routing data and addressing modes.

Ans. (d)

15. The contents of the flag register after execution of the following program by 8085 microprocessor will be:

Program
SUB A
MVI B, $(01)_H$
DCR B
HLT

(a) $(54)_H$ (b) $(00)_H$ (c) $(01)_H$ (d) $(45)_H$

Solution:

SUB A	Subtract contents of memory location whose contents are stored in A
MVI B, $(01)_H$	Move B with $(01)_H$
DCR B	Decrement B gives $(54)_H$
HLT	

Ans. (a)

16. An N-bit carry look-ahead adder, where N is a multiple of 4, employs ICS 74181 (40-bit ALU) and 74182 (4-bit carry look-ahead generator).

The minimum addition time using the best architecture for adder is

(a) Proportional to N
(b) Proportional to log N
(c) A constant
(d) None of the above

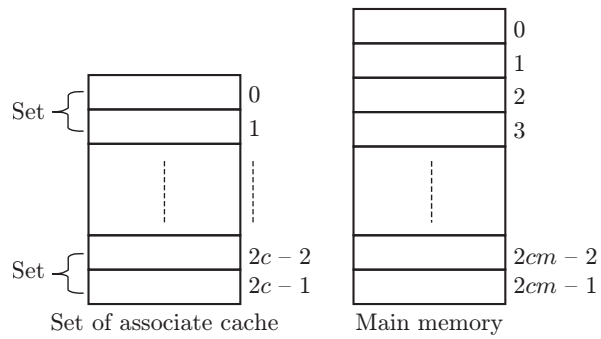
Solution: The addition time for N-bit carry look-ahead adder is always a constant.

Ans. (c)

17. The main memory of a computer has 2 cm blocks while the cache has $2c$ blocks. If the cache uses the set-associative mapping scheme with two blocks per set, then block k of the main memory maps to the set:

(a) (k mod m) of the cache
(b) (k mod c) of the cache
(c) (k mode $2c$) of the cache
(d) (k mod $2cm$) of the cache

Solution:



Number of sets in cache

$$= \frac{\text{Number of blocks in cache}}{\text{Number of blocks in one set}} = \frac{2c}{2}$$

Number of sets in cache = c
So to map block k, main memory maps to the set (k mod c) of the cache.

Ans. (b)

18. Which of the following is/are advantage of virtual memory?

(a) Faster memory to memory on an average
(b) Processes can be given protected address spaces
(c) Linker can assign addresses independent of where the program will be loaded in physical memory
(d) Programs larger than the physical memory size can be run

Solution: Virtual memory allows programs to have a larger space than the physical memory size to run. So, the programmer does not have to worry about physical memory size.

Ans. (d)

19. The number of full and half adders required to add 16-bit numbers is

(a) 8 half adders, 8 full adders
(b) 1 half adder, 15 full adders
(c) 16 half adders, 0 full adder
(d) 4 half adders, 12 full adders

Solution: To add N-bit numbers:
Number of half adders required = 1
Number of full adders required = $N - 1$
Therefore, to add 16-bit number only, 1 half adder and 15 full adders are required.

Ans. (b)

20. Which of the following requires a device driver

(a) Register (b) Cache
(c) Main memory (d) Disk

Solution: Disk is the I/O device attached externally to the processor. Therefore, disk requires a device driver.

Ans. (d)

21. More than one word are put in one cache block to

(a) exploit the temporal locality of reference in a program.
(b) exploit the spatial locality of reference in a program.
(c) reduce the miss penalty.
(d) none of the above.

Solution: There are two types of locality of references temporal and spatial locality.

The concept of spatial locality, instead of fetching just one item from the main memory to the cache, is useful to fetch several items that reside at adjacent address as well.
So, option (b) is correct.

Ans. (b)

22. Which of the following statements is false?

(a) Virtual memory implements the translation of a program's address space into physical memory address space.
(b) Virtual memory allows each program to exceed the size of the primary memory.
(c) Virtual memory increases the degree of multiprogramming.
(d) Virtual memory reduces the context-switching overhead.

Solution: Virtual memory increases the context-switching overhead.

Ans. (d)

23. Advantage of synchronous sequential circuits over asynchronous ones is

(a) faster operation.
(b) ease of avoiding problems due to hazards.
(c) lower hardware requirement.
(d) better noise immunity.

Solution: Because of less delay, synchronous sequential circuits have faster operation than asynchronous ones.

Ans. (a)

24. The total size of address space in a virtual memory system is limited by

(a) the length of MAR.
(b) the available secondary storage.
(c) the available main memory.
(d) all of the above.

Solution: Virtual memory depends only on the available size of the secondary memory.

Ans. (b)

25. Comparing the time T_1 taken for a single instruction on a pipelined CPU with time T_2 taken on a non-pipelined but identical CPU, we can say that

(a) $T_1 \leq T_2$
(b) $T_1 \geq T_2$
(c) $T_1 < T_2$
(d) T_1 plus T_2 is the time taken for one instruction fetch cycle

Solution: In case of one instruction, non-pipelined CPU takes less time as compared to pipelined CPU. This is due to buffer delays for pipelining.

Ans. (b)

GATE PREVIOUS YEARS' QUESTIONS

1. For a pipelined CPU with a single ALU, consider the following situations:

I. The $j + 1$-st instruction uses the result of the jth instruction as an operand.
II. The execution of a conditional jump instruction.
III. The j-th and $j + 1$-st instructions require the ALU at the same time.

Which of the above can cause a hazard?

(a) I and II only (b) II and III only
(c) III only (d) All the three

(GATE 2003: 1 Mark)

Solution: All the three statements cause hazards.

Ans. (d)

Common Data Questions 2 and 3: Consider the following assembly language program for a hypothetical processor. A, B and C are 8-bit registers. The meanings of various instructions are shown as comments:

MOV B, #0;	$B \leftarrow 0$
MOV C, #8;	$C \leftarrow 8$
Z: CMP C, #0;	Compare C with 0
JZ X;	Jump to X if zero flag is set

SUB C, #1;	$C \leftarrow C - 1$
RRC A, #1;	Right rotate A through carry by one bit. Thus:
;	if the initial values of A and the carry flag are $a_7..a_0$ and; c_0 respectively, their values after the execution of this; instruction will be c_0 $a_7..a_1$ and a_0, respectively.
JC Y;	Jump to Y if carry flag is set
JMP Z;	Jump to Z
Y: ADD B, #1;	$B \leftarrow B + 1$
JMP Z;	Jump to Z
X:	

2. If the initial value of register A is A_0, the value of register B after the program execution will be

(a) The number of 0 bits in A_0
(b) The number of 1 bits in A_0
(c) A_0
(d) 8

(GATE 2003: 2 Marks)

Solution: After execution, register B will contain number of 0 bits in A_0.

Ans. (a)

3. Which of the following instructions when inserted at location X will ensure that the value of register A after program execution is the same as its initial value?

(a) RRC A, #1
(b) NOP; no operation
(c) LRC A, #1; left rotate A through carry flag by one bit
(d) ADD A, #1

(GATE 2003: 2 Marks)

Solution: RRC A, #1

Ans. (a)

4. Which of the following addressing modes are suitable for program relocation at run time?

(i) Absolute addressing (ii) Base addressing
(iii) Relative addressing (iv) Indirect addressing

(a) (i) and (iv) (b) (i) and (ii)
(c) (ii) and (iii) (d) (i), (ii) and (iv)

(GATE 2004: 1 Mark)

Solution: Both base addressing and relative addressing modes are suitable.

Ans. (c)

Common Data Questions 5 and 6: Consider the following program segment for a hypothetical CPU having three-user registers R_1, R_2 and R_3.

Instruction	Operation	Instruction Size (in words)
MOV R_1, 5000	; $R_1 \leftarrow$ Memory [5000]	2
MOV R_2 (R_1)	; $R_2 \leftarrow$ Memory [(R_1)]	1
ADD R_2, R_3	; $R_2 \leftarrow R_2 + R_3$	1
MOV 6000, R_2	; Memory [6000] $\leftarrow R_2$	2
HALT	; Machine halts	1

5. Consider that the memory is byte addressable with size 32 bits, and the program has been loaded starting from memory location 1000 (decimal). If an interrupt occurs while the CPU has been halted after executing the HALT instruction, the return address (in decimal) saved in the stack will be

(a) 1007 (b) 1020
(c) 1024 (d) 1028

(GATE 2004: 2 Marks)

Solution:

MOV R_1, 5000	; 2	1000 to 1007
MOV R_2 (R1)	; 1	1008 to 1011
ADD R_2, R_3	; 1	1012 to 1015
MOV 6000, R_2	; 2	1016 to 1023
HALT	; 1	1024 to 1027

Ans. (c)

6. Let the clock cycles required from various operations be as follows:

Register to/from memory transfer: 3 clock cycles ADD with both operands in register: 1 clock cycle Instruction fetches and decodes: 2 clock cycles per word
The total number of clock cycles required to execute the program is

(a) 29 (b) 24
(c) 23 (d) 20

(GATE 2004: 2 Marks)

Solution:

		Clock Cycles
MOV R_1, 5000	; 2	8
MOV R_2 (R1)	; 1	5
ADD R_2, R_3	; 1	1
MOV 6000, R	; 2	8
HALT	; 1	2

Total 24 clock cycles are required.

Ans. (b)

7. Consider a small two-way set-associative cache memory, consisting of four blocks. For choosing the block to be replaced, use the least recently used (LRU) scheme. The number of cache misses for the following sequence of block addresses 8, 12, 0, 12, 8 is

(a) 2 (b) 3 (c) 4 (d) 5

(GATE 2004: 2 Marks)

Solution: Sequence is: 8, 12, 0, 12, 8

	12	12	12
8	8	0	8

Total number of miss = 4

Ans. (c)

8. A hard disk with a transfer rate of 10 MB/s is constantly transferring data to memory using DMA. The processor runs at 600 MHz, and takes 300 and 900 clock cycles to initiate and complete DMA transfer, respectively. If the size of the transfer is 20 KB, what is the percentage of processor time consumed for the transfer operation?

(a) 5.0% (b) 1.0% (c) 0.5% (d) 0.1%

(GATE 2004: 2 Marks)

Solution: Given that data transfer rate = 10 MB/s and size of transfer = 20 KB
So, % processor time consumed = 20 KB × 100/10 MB/s = 0.1%

Ans. (d)

9. A 4-stage pipeline has the stage delays as 150, 120, 160 and 140 ns, respectively. Registers that are used between the stages have a delay of 5 ns each. Assuming constant clocking rate, the total time taken to process 1000 data items on this pipeline will be

(a) 120.4 μs (b) 160.5 μs (c) 165.5 μs (d) 590.0 μs

(GATE 2004: 2 Marks)

Solution: Maximum stage delay = 160 + 5 = 165 ns
$T = [4 + (1000 - 1)]\ 165 = 165.5$ μs

Ans. (c)

10. Consider a multiplexer with X and Y as data inputs and Z as control input. Z = 0 selects input X, and Z = 1 selects input Y. What are the connections required to realize the 2-variable Boolean function f = T + R, without using any additional hardware?

(a) R to X, 1 to Y, T to Z
(b) T to X, R to Y, T to Z
(c) T to X, R to Y, 0 to Z
(d) R to X, 0 to Y, T to Z

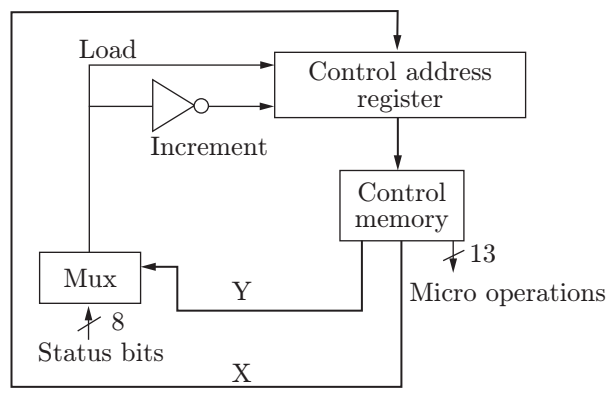
Solution: Connect R to X, 1 to Y and T to Z.

Ans. (a)

11. The microinstructions stored in the control memory of a processor have a width of 26 bits. Each microinstruction is divided into three fields: a micro-operation field of 13 bits, a next address field (X), and a MUX select field (Y). there are 8 status bits in the inputs of the MUX. How many bits are there in the X and Y fields, and what is the size of the control memory in number of words?

(a) 10, 3, 1024 (b) 8, 5, 256
(c) 5, 8, 2048 (d) 10, 3, 512

(GATE 2004: 2 Marks)



Solution: MUX has 8 input bits, 3 select lines are required. So, Y = 3. Total bits = 26

X = 26 – 13 – 3 = 10. So, memory required = 2^{10} = 1024

Ans: (a)

12. Which one of the following is true for a CPU having a single interrupt request line and a single interrupt grant line?

(a) Neither vectored interrupt nor multiple interrupting devices are possible.
(b) Vectored interrupts are not possible but multiple interrupting devices are possible.
(c) Vectored interrupts and multiple interrupting devices are both possible.
(d) Vectored interrupt is possible but multiple interrupting devices are not possible.

(GATE 2005: 1 Mark)

Solution: Vectored interrupts are not possible but multiple interrupting devices are possible.

Ans. (b)

13. Normally user programs are prevented from handling I/O directly by I/O instructions in them. For CPUs having explicit I/O instructions, such I/O protection is ensured by having the I/O instructions privileged. In a CPU with memory mapped I/O, there is no explicit I/O instruction. Which one of the following is true for a CPU with memory mapped I/O?

(a) I/O protection is ensured by operating system routine(s).
(b) I/O protection is ensured by a hardware trap.
(c) I/O protection is ensured during system configuration.
(d) I/O protection is not possible.

(GATE 2005: 1 Mark)

Solution: I/O protection is ensured by operating system routine(s).

Ans. (a)

14. Consider a three word machine instruction ADD A[R0], @B

The first operand (destination) "A [R0]" uses indexed addressing mode with R0 as the index register. The second operand (source) "@B" uses indirect addressing mode. A and B are memory addresses residing at the second and the third words, respectively. The first word of the instruction specifies the opcode, the index register designation and the source and destination addressing modes.

During execution of ADD instruction, the two operands are added and stored in the destination (first operand).

The number of memory cycles needed during the execution cycle of the instruction is:

(a) 3 (b) 4
(c) 5 (d) 6

(GATE 2005: 2 Marks)

Solution: A[R0] = require 1 memory cycle

The 3rd and 4th operands are indirectly addressed so each requires 2 cycles = 4 memory cycles.

1 memory cycle will be required to store the result. So, total of 6 memory cycles are required.

Ans. (d)

15. Match each of the high level language statements given on the left-hand side with the most natural addressing mode from those listed on the right-hand side.

(1) `A[I] = B[J]`	(a) Indirect addressing
(2) `while (*A++)`	(b) Indexed addressing
(3) `int temp =*x`	(c) Auto increment

(a) (1, c), (2, b), (3, a) (b) (1, a), (2, c), (3, b)
(c) (1, b), (2, c), (3, a) (d) (1, a), (2, b), (3, c)

(GATE 2005: 2 Marks)

Solution: A[I] = indexed addressing
*A++ = auto increment
Temp = *x = indirect addressing

Ans. (c)

16. Consider a direct mapped cache of size 32 KB with block size 32 bytes. The CPU generates 32-bit addresses. The number of bits needed for cache indexing and the number of tag bits are, respectively

(a) 10, 17 (b) 10, 22 (c) 15, 17 (d) 5, 17

(GATE 2005: 2 Marks)

Solution: Cache size = 32 KB = 15 bits are required

Index bits = 15
Tag bits 32 – 15 = 17

Ans. (a)

17. A 5-stage pipelined CPU has the following sequence of stages:

IF: Instruction fetch from instruction memory
RD: Instruction decodes and register read
EX: Execute ALU operation for data and address computation
MA: Data memory access for write access, the register read at RD state is used.
WB: Register write back.

Consider the following sequence of instructions:

I_1: L R_0, loc 1; $R_0 \Leftarrow M$[loc1]
I_2: A R_0, R_0 1; $R_0 \Leftarrow R_0 + R_0$
I_3: S R_2, R_0 1; $R_2 \Leftarrow R_2 - R_0$

Let each stage take one clock cycle. What is the number of clock cycles taken to complete the above sequence of instructions starting from the fetch of I_1?

(a) 8 (b) 10 (c) 12 (d) 15

(GATE 2005: 2 Marks)

Solution: 12 clock cycles are taken, as shown below

Clocks	1	2	3	4	5	6	7	8	9	10	11	12
I_1	IF	RD	EX	MA	WB							
I_2		IF	–	–	–	RD	EX	MA	WB			
I_3			IF	–	–	–	RD	–	–	EX	MA	WB

Ans. (c)

18. A device with data transfer rate 10 KB/s is connected to a CPU. Data is transferred byte-wise. Let the interrupt overhead be 4 s. The byte transfer time between the device interfaces register and CPU or memory is negligible. What is the minimum performance gain of operating the device under interrupt mode over operating it under program-controlled mode?

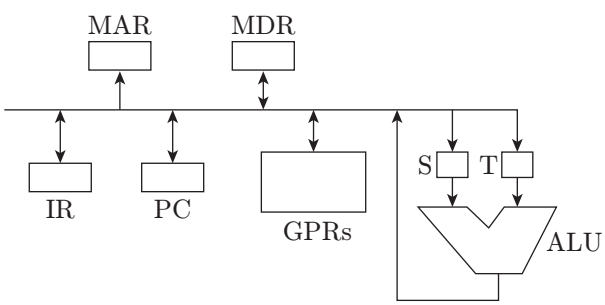
(a) 15 (b) 25
(c) 35 (d) 45

(GATE 2005: 2 Marks)

Solution: Data transfer rate = 10 KB/s
Interrupt overhead = 4×10^{-2} s
10 KB is sent = 1 s
1 B is sent = 1/10K = $100 - 10^{-2}$ s
Minimum performance gain = $100 \times 10^{-2}/4 \times 10^{-2}$
= 25

Ans. (b)

Common Data Questions 19 and 20: Consider the following data path of a CPU. The ALU, the bus and all the registers in the data path are of identical size. All operations including incrementation of the PC and the GPRs are to be carried out in the ALU. Two clock cycles are needed for memory read operation − the first one for loading address in the MAR and the next one for loading data from the memory bus into the MDR.



19. The instruction "add R_0, R_1" has the register transfer interpretation $R_0 \Leftarrow R_0 + R_1$. The minimum number of clock cycles needed for execution cycle of this instruction is

(a) 2 (b) 3
(c) 4 (d) 5

(GATE 2005: 2 Marks)

Solution: There will be three cycles−(1) R_{1out}, S_{in}, (2) R_{2out}, T_{in} and (3) S_{out}, T_{out}, ALU_{add}, R_{in}.

Ans. (b)

20. The instruction "call R_n, sub" is a two-word instruction. Assuming that PC is incremented during the fetch cycle of the first word of the instruction, its register transfer interpretation is

$$R_n \Leftarrow \text{PC}+1;$$
$$\text{PC} \Leftarrow M[\text{PC}];$$

The minimum number of CPU clock cycles needed during the execution cycle of this instruction is
(a) 2 (b) 3 (c) 4 (d) 5

(GATE 2005: 2 Marks)

Solution: The minimum number of CPU clock cycles needed during the execution cycle = 4. This is because
1 cycle is required to transfer already incremented value of PC
2 cycles for getting data in MDR
1 to load value of MDR in PC

Ans. (c)

21. Consider a disk drive with the following specifications:

16 surfaces, 512 tracks/surface, 512 sectors/track, 1 KB/sector, rotation speed 3000 rpm. The disk is operated in cycle stealing mode whereby whenever one 4 byte word is ready it is sent to memory; similarly, for writing, the disk interface reads a 4 byte word from the memory in each DMA cycle. Memory cycle time is 40 nsec. The maximum percentage of time that the CPU gets blocked during DMA operation is:

(a) 10 (b) 25
(c) 40 (d) 50

(GATE 2005: 2 Marks)

Solution:
Data transfer in one rotation = 512 × 1024 Bytes

1 rotation takes = $\frac{60}{3000}$ s

512KB is transferred in = $\frac{60}{3000}$ s

1 byte will be transferred = $\frac{60}{3000} \times 512 \times 1024$

4 bytes will be transferred

$$= 60 \times \frac{4}{3000} \times 512 \times 1024 = 152.58 \text{ ns}$$

$$\text{Block } \% = \frac{40}{152.28} = 26\%$$

Ans. (b)

22. A CPU has 24-bit instructions. A program starts at address 300 (in decimal). Which one of the following is a legal program counter (all values in decimal)?

(a) 400 (b) 500
(c) 600 (d) 700

(GATE 2006: 1 Mark)

Solution: Size of instruction = 24 bits; Start address = 300. Legal address will be multiple of three, that is, 300.

Ans. (c)

23. A CPU has a cache with block size 64 bytes. The main memory has k banks, each bank being c bytes wide. Consecutive c byte chunks are mapped on consecutive banks with wrap around. All the k banks can be accessed in parallel, but two accesses to the same bank must be serialized. A cache block access may involve multiple iterations of parallel bank accesses depending on the amount of data obtained by accessing all the k banks in parallel. Each iteration requires decoding the bank numbers to be accessed in parallel and this takes $k/2$ ns. The latency of one bank access is 80 ns. If $c = 2$ and $k = 24$, the latency of retrieving a cache block starting at address zero from the main memory is

(a) 92 ns (b) 104 ns
(c) 172 ns (d) 184 ns

(GATE 2006: 2 Marks)

Solution:
Time for one parallel process = $k/2$ + latency
Time for one byte = $24/2 + 80 = 92$
Total time for c bytes = $2 \times 92 = 184$ ns

Ans. (d)

24. A CPU has a five-stage pipeline and runs at 1 GHz frequency. Instruction fetch happens in the first stage of the pipeline. A conditional branch instruction computes the target address and evaluates the condition in the third stage of the pipeline. The processor stops fetching new instructions following a conditional branch until the branch outcome is known. A program executes 10^9 instructions out of which 20% are conditional branches. If each instruction takes one cycle to complete on average, the total execution time of the program is

(a) 1.0 s (b) 1.2 s
(c) 1.4 s (d) 1.6 s

(GATE 2006: 2 Marks)

Solution:
Total execution time of the program

$$= 10^9 + 0.20 \times 2 \times 10^9 = 1.4 \text{ s}$$

Ans. (c)

25. Consider a new instruction named branch-on-bit-set (mnemonic bbs). The instruction "bbs reg, pos, label" jumps to label if bit in position pos of register operand reg is one. A register is 32 bits wide and the bits are numbered 0 to 31, bit in position 0 being the least significant. Consider the following emulation of this instruction on a processor that does not have bbs implemented.

temp ← reg & mask
Branch to label if temp is non-zero.

The variable temp is a temporary register. For correct emulation, the variable mask must be generated by

(a) mask ← $0 \times 1 \ll$ pos
(b) mask ← $0 \times ffffffff \gg$ pos
(c) mask ← pos
(d) mask ← $0 \times f$

(GATE 2006: 2 Marks)

Solution: As there is only one bit with pos, the other bits need to be set to 0 in temp. The mask register must have 1 in pos position, for which pos number of left shifts over 1 need to be made.

Ans. (a)

Common Data Questions 26 and 27: Consider two cache organizations. The first one is 32 KB, 2-way set associative with 32-byte block size. The second one is of the same size but direct mapped. The size of an address is 32 bits in both cases. A 2-to-1 multiplexer has a latency of 0.6 ns while a k-bit comparator has a latency of $k/10$ ns. The hit latency of the set-associative organization is h_1 while that of the direct mapped one is h_2.

26. The value of h_1 is

(a) 2.4 ns (b) 2.3 ns
(c) 1.8 ns (d) 1.7 ns

(GATE 2006: 2 Marks)

Solution:
Address bits = 32
Block size = 32 B = 5 bits
Size of cache = 32KB/32 = 1KB = 10 bits
For 2-way set-associative memory = index bits = 9, tag bits = 18
$k/10 = 18/10 = 1.8$ + latency = $1.8 + 0.6 = 2.4$

Ans. (a)

27. The value of h_2 is

(a) 2.4 ns (b) 2.3 ns
(c) 1.8 ns (d) 1.7 ns

(GATE 2006: 2 Marks)

Solution:
In direct memory access: Tag bits = 17; Index = 10 bits; word = 5 bits
$k/10 = 17/10 = 1.7$ + latency = $1.7 + 0.6 = 2.3$

Ans. (b)

Common Data Questions 28 and 29: A CPU has a 32-KB direct-mapped cache with 128-byte block size. Suppose A is a two-dimensional array of size 512×512 with elements that occupy 8 bytes each. Consider the following two C code segments, P_1 and P_2.

```
P1: for (i=0; i<512; i++) {
          for (j=0; j<512; j++) {
              x +=A[i][j];
            }
        }
P2: for (i=0; i<512; i++) {
          for (j=0; j<512; j++) {
             x +=A[j][i];
            }
        }
```

P_1 and P_2 are executed independently with the same initial state, namely, the array A is not in the cache and i, j, x are in registers. Let the number of cache misses experienced by P_1 be M_1 and that for P_2 be M_2.

28. The value of M_1 is

(a) 0 (b) 2048
(c) 16384 (d) 262144

(GATE 2006: 2 Marks)

Solution:

Memory = 32 KB
Block size = 128 B
Number of blocks = 256
Number of elements in block = 256/8 =16
P_1 cache misses = $M_1 : 512 \times 512/16 = 16384$

Ans. (c)

29. The value of the ratio M_1/M_2 is

(a) 0 (b) 1/16
(c) 1/8 (d) 16

(GATE 2006: 2 Marks)

Solution:
P_2 number of cache misses = $M_2 = 512 \times 512$
Ratio of M_1:M_2 = 1:16

Ans. (b)

30. Consider a 4-way set-associative cache consisting of 128 lines with a line size of 64 words. The CPU generates a 20-bit address of a word in main memory. The number of bits in the TAG, LINE and WORD fields are, respectively:

(a) 9, 6, 5 (b) 7, 7, 6
(c) 7, 5, 8 (d) 9, 5, 6

(GATE 2007: 1 Mark)

Solution: For 64 words, $\log_2 64 = 6$ bits are required. For lines = 128/4 = 32 lines, 5 bits are required

Tag bits	Line	Word
9	5	6

Ans. (d)

31. Consider a pipelined processor with the following four stages:

IF: Instruction fetch
ID: Instruction decode and operand fetch
EX: Execute
WB: Write back
The IF, ID and WB stages take one clock cycle each to complete the operation. The number of clock cycles for the EX stage depends on the instruction. The ADD and SUB instructions need 1 clock cycle and the MUL instruction needs 3 clock cycles in the EX stage. Operand forwarding is used in the pipelined processor. What is the number of clock cycles taken to complete the following sequence of instructions?

ADD R_2, R_1, R_0 $R_2 \leftarrow R_1 + R_0$
MUL R_4, R_3, R_2 $R_4 \leftarrow R_3 \times R_2$
SUB R_6, R_5, R_4 $R_6 \leftarrow R_5 - R_4$

(a) 7 (b) 8 (c) 10 (d) 14

(GATE 2007: 1 Mark)

Solution:

Clock	1	2	3	4	5	6	7	8
I_1	IF	ID	EX	WB				
I_2		IF	ID	EX	EX	EX	WB	
I_3			IF	ID	_	_	EX	WB

Using operand forwarding, 8 clock cycles are required.

Ans. (b)

32. In a simplified computer, the instructions are

OP R_J, R_i – Performs R_J OP R_i and stores the result in register R_i.
OP m, R_i – Performs val OP R_i and stores the result in R_i. val denotes the content of memory location m.
MOV, mR_i – Moves the content of memory location m to register R_i.
MOV R_i, m – Moves the content of register R_i to memory location m.

The computer has only two registers, and OP is either ADD or SUB. Consider the following basic block:

$$t_1 = a + b$$
$$t_2 = c + d$$
$$t_3 = e - t_2$$
$$t_4 = t_1 - t_3$$

Assume that all operands are initially in memory. The final value of the computation should be in memory. What is the minimum number of MOV instructions in the code generated for this basic block?

(a) 2 (b) 3
(c) 5 (d) 6

(GATE 2007: 1 Mark)

Solution: The instructions generated in the code for this basic block are as follows:

MOV a, R_i
ADD b, R_i
MOV c, R_j
ADD d, R_j
SUB e, R_j
SUB R_i, R_j
MOV m, R_i

Ans. (b)

Common Data Questions 33–35: Consider the following program segment. Here R_1, R_2 and R_3 are general purpose registers.

Instruction	Operation	Instruction Size (No. of Words)
MOV R_1, (3000)	$R_1 \leftarrow m[3000]$	2
LOOP: MOV R_2, (R_3)	$R_2 \leftarrow M[R_3]$	1
ADD R_2, R_1	$R_2 \leftarrow R_1 + R_2$	1
MOV (R_3), R_2	$M[R_3] \leftarrow R_2$	1
INC R_3	$R_3 \leftarrow R_3 + 1$	1
DEC R_1	$R_1 \leftarrow R_1 - 1$	1
BNZ LOOP	Branch on not zero	2
HALT	Stop	1

Assume that the content of memory location 3000 is 10 and the content of the register R_3 is 2000. The content of each of the memory locations from 2000 to 2010 is 100. The program is loaded from the memory location 1000. All the numbers are in decimal.

33. Assume that the memory is word addressable. The number of memory references for accessing the data in executing the program completely is

(a) 10 (b) 11
(c) 20 (d) 21

(GATE 2007: 1 Mark)

Solution: Given that $R_1 = 10$, so the loop will run 10 times

$$10 \times 2 + 1 = 21$$

Ans. (d)

34. Assume that the memory is word addressable. After the execution of this program, the content of memory location 2010 is

(a) 100 (b) 101
(c) 102 (d) 110

(GATE 2007: 1 Mark)

Solution: It will remain 100, because the loop will exit as the value in R_1 becomes 0 when address in R_3 becomes 2010.

Ans. (a)

35. Assume that the memory is byte addressable and the word size is 32 bits. If an interrupt occurs during the execution of the instruction "INC R_3", what return address will be pushed on to the stack?

(a) 1005 (b) 1020
(c) 1024 (d) 1040

(GATE 2007: 1 Mark)

Solution: Memory is byte addressable, take 4 bytes per word. So at INC R_3, stack will contain 1024.

Ans. (c)

36. Consider a disk pack with 16 surfaces, 128 tracks per surface and 256 sectors per track. 512 bytes of data are stored in a bit serial manner in a sector. The capacity of the disk pack and the number of bits required to specify a particular sector in the disk are respectively:

(a) 256 Mbyte, 19 bits (b) 256 Mbyte, 28 bits
(c) 512 Mbyte, 20 bits (d) 64 Gbyte, 28 bits

(GATE 2007: 1 Mark)

Solution:
Disk capacity = 16 surfaces × 128 tracks × 256 sectors × 512 bytes = 256 MB
Total number of sectors = $16 \times 128 \times 256 = 2^{19}$

Ans. (a)

Linked Answer Questions 37 and 38: Consider a machine with a byte addressable main memory of 162 bytes. Assume that a direct mapped data cache consisting of 32 lines of 64 bytes each is used in the system. A 50×50 two-dimensional array of bytes is stored in the main memory starting from memory location 1100H. Assume that the data cache is initially empty. The complete array is accessed twice. Assume that the contents of the data cache do not change in between the two accesses.

37. How many data cache misses will occur in total?

(a) 48 (b) 50 (c) 56 (d) 59

(GATE 2007: 2 Marks)

Solution:
Main memory = 2^{16} B
Block size = 64 B
Number of blocks = 32
Number of elements = $50 \times 50 = 2500$
Starting from location 1100 means from 68th block
Number of blocks = 2500/64 = 40 blocks
Initially cache is empty for 32 misses, then 8 are remaining from total 40 for one access
Array is traversed twice, so data cache misses = 40 + 8 + 8 = 56

Ans. (c)

38. Which of the following lines of the data cache will be replaced by new blocks in accessing the array for the second time?

(a) line 4 to line 11 (b) line 4 to line 12
(c) line 0 to line 7 (d) line 0 to line 8

Solution:
Applying k mod c to find the location: 68 mod 32 = 4 to 11

Ans. (a)

39. Which of the following is/are true for the auto-increment addressing mode?

I. It is useful in creating self-relocating code.
II. If it is included in an Instruction Set Architecture, then an additional ALU is required for effective address calculation.
III. The amount of increment depends on the size of the data item accessed.

(a) I only (b) II only
(c) III only (d) II and III only

(GATE 2008: 2 Marks)

Solution: Only statement (III) is true.

Ans. (c)

40. Which of the following must be true for the RFE (Return from Exception) instruction on a general purpose processor?

I. It must be a trap instruction.
II. It must be a privileged instruction.
III. An exception cannot be allowed to occur during execution of an RFE instruction.

(a) I only (b) II only
(c) I and II only (d) I, II and III only

(GATE 2008: 2 Marks)

Solution: RFE (Return from Exception) is a privileged trap instruction that is executed when exception occurs, so an exception is not allowed to execute. Option (d) is the correct option.

Ans. (d)

41. For a magnetic disk with concentric circular tracks, the seek latency is not linearly proportional to the seek distance due to

(a) non-uniform distribution of requests
(b) arm starting and stopping inertia
(c) higher capacity of tracks on the periphery of the platter
(d) use of unfair arm scheduling policies

(GATE 2008: 2 Marks)

Solution: Tracks on magnetic disks are concentric and seek latency from one sector to other which may or may not be in different tracks. This seek distance is not proportional to latency since the tracks at periphery have higher diameter, and hence higher capacity to store data.

Ans. (b)

42. Which of the following are NOT true in a pipelined processor?

I. Bypassing can handle all RAW hazards.
II. Register renaming can eliminate all register carried WAR hazards.
III. Control hazard penalties can be eliminated by dynamic branch prediction.

(a) I and II only (b) I and III only
(c) II and III only (d) I, II and III

(GATE 2008: 2 Marks)

Solution: All the statements are true.

Ans. (d)

43. For inclusion to hold between two cache levels L1 and L2 in a multi-level cache hierarchy, which of the following are necessary?

I. L1 must be a write-through cache
II. L2 must be a write-through cache
III. The associativity of L2 must be greater than that of L1
IV. The L2 cache must be at least as large as the L1 cache

(a) IV only (b) I and IV only
(c) I, II and IV only (d) I, II, III and IV

(GATE 2008: 2 Marks)

Solution: L1 and L2 cache are placed between CPU and they can be both write through cache but not necessarily.
Associativity does not matter.
L2 cache must be at least as large as L1 cache, since all the words in L1 are also in L2.

Ans. (a)

44. The use of multiple register windows with overlap causes a reduction in the number of memory accesses for

I. Function locals and parameters
II. Register saves and restores
III. Instruction fetches

(a) I only (b) II only (c) III only (d) I, II and III

(GATE 2008: 2 Marks)

Solution: Multiple register windows with overlap causes a reduction in the number of memory accesses for register saves and restores.

Ans. (b)

45. Consider a machine with a 2-way set-associative data cache of size 64 KB and block size 16 bytes. The cache is managed using 32 bit virtual addresses and the page size is 4 KB. A program to be run on this machine begins as follows:

```
double ARR [1024]  [1024];
int i, j;
    /* Initialize array ARR to 0.0 */
       for(i=0; i<1024; i++)
           for(j=0; j<1024; j++)
              ARR [i] [j] =0.0;
```

The size of double is 8 bytes. Array ARR is located in memory starting at the beginning of virtual page 0xFF000 and stored in row major order. The cache is initially empty and no pre-fetching is done. The only data memory references made by the program are those to array ARR.
The total size of the tags in the cache directory is

(a) 32 Kbits (b) 34 Kbits
(c) 64 Kbits (d) 68 Kbits

(GATE 2008: 2 Marks)

Solution:
Virtual address = 32 bits
2-way cache size = 64 KB
1 set will contain = 32 KB entries = 15 bits
Block size = 16 bytes = 4 bits

Tag bits	Set bits	Word bits
17	11	4

Tag size = $17 \times 2 \times 1024$ = 34 kbits

Ans. (b)

46. Consider the data given in the above question. Which of the following array elements has the same cache index as ARR[0][0]?

(a) ARR[0][4] (b) ARR[4][0]
(c) ARR[0][5] (d) ARR[5][0]

(GATE 2008: 2 Marks)

Solution: Total elements can come in one slot 2048. After 2048 elements, same cache index will be on [2][0] and [4][0].

Ans. (b)

47. The cache hit ratio for this initialization loop is

(a) 0% (b) 25% (c) 50% (d) 75%

(GATE 2008: 2 Marks)

Solution: Cache hit ratio is found out as follows:

$$\frac{1024}{2048} = \frac{1}{2} = 50\%$$

As we can see in the above, there will be 50% hits.

Ans. (c)

Linked Answer Questions 48 and 49: Delayed branching can help in the handling of control hazards.

48. For all delayed conditional branch instructions, irrespective of whether the condition evaluates to true or false:

(a) The instruction following the conditional branch instruction in memory is executed.
(b) The first instruction in the fall through path is executed.
(c) The first instruction in the taken path is executed.
(d) The branch takes longer to execute than any other instruction.

(GATE 2008: 2 Marks)

Solution: The first instruction following the branch instruction is always executed (irrespective of whether the branch is taken or not).

Ans. (b)

49. The following code is to run on a pipelined processor with one branch delay slot:

I_1: ADD $R_2 \leftarrow R_7 + R_8$
I_2: SUB $R_4 \leftarrow R_5 - R_6$
I_3: ADD $R_1 \leftarrow R_2 + R_3$
I_4: STORE Memory $[R_4] \leftarrow R_1$
BRANCH to Label if $R_1 == 0$

Which of the instructions I_1, I_2, I_3 or I_4 can legitimately occupy the delay slot without any other program modification?

(a) I_1 (b) I_2 (c) I_3 (d) I_4

(GATE 2008: 2 Marks)

Solution: Instruction I_2 contains delayed slot. I_4 has data dependency in I_2.

Ans. (b)

50. How many 32K × 1 RAM chips are needed to provide a memory capacity of 256 K bytes?

(a) 8 (b) 32
(c) 64 (d) 128

Solution:

$$\text{Number of chips required} = \frac{256\text{K} \times 8}{32\text{K} \times 1} = 64$$

Ans. (c)

51. A CPU generally handles an interrupt by executing an interrupt service routine

(a) as soon as an interrupt is raised.
(b) by checking the interrupt register at the end of the fetch cycle.
(c) by checking the interrupt register after finishing the execution of the current instruction.
(d) by checking the interrupt register at fixed time intervals.

Solution: Interrupts are handled by checking the interrupt register after finishing the execution of current instruction.

Ans. (c)

52. Consider a 4-stage pipeline processor. The number of cycles needed by the four instructions I_1, I_2, I_3, I_4 in stages S_1, S_2, S_3, S_4 is shown below:

	S_1	S_2	S_3	S_4
I_1	2	1	1	1
I_2	1	3	2	2
I_3	1	1	1	3
I_4	1	2	2	2

What is the number of cycles needed to execute the following loop?

```
for (i=1 to 2) {I1; I2; I3; I4;}
```

(a) 16 (b) 23 (c) 28 (d) 30

(GATE 2009: 2 Marks)

Solution:

Clock	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
I_1	S_1	S_1	S_2	S_3	S_4										
I_2			S_1	S_2	S_2	S_2	S_3	S_3	S_4	S_4					
I_3				S_1	S_1		S_2		S_3		S_4	S_4	S_4		
I_4							S_1		S_2	S_2	S_3	S_3		S_4	S_4

For two iterations = 15 × 2 = 30 clock cycles

Ans. (d)

53. Consider a 4-way set-associative cache (initially empty) with total 16 cache blocks. The main memory consists of 256 blocks and the request for memory blocks is in the following order:

0, 255, 1, 4, 3, 8, 133, 159, 216, 129, 63, 8, 48, 32, 73, 92, 155

Which one of the following memory block will NOT be in cache if LRU replacement policy is used?

(a) 3 (b) 8 (c) 129 (d) 216

(GATE 2009: 2 Marks)

Solution:
To decide the location (address), mod 4 is applied.

Set 0	0, 4, 8,216 → 48, 32, 8, 92
Set 1	1, 133, 129, 73
Set 2	155
Set 3	255, 3, 159, 63

216 will not be there in cache.

Ans. (d)

Common Data Questions 54 and 55: A hard disk has 63 sectors per track, 10 platters each with 2 recording surfaces and 1000 cylinders. The address of a sector is given as a triple <*c*, *h*, *s*>, where c is the cylinder number, h is the surface number and s is the sector number. Thus, the 0th sector is addressed as <0, 0, 0>, the 1st sector as <0, 0, 1>, and so on.

(GATE 2009: 2 Marks)

54. The address <400, 16, 29> corresponds to sector number:

(a) 505035 (b) 505036
(c) 505037 (d) 505038

Solution:
Total surfaces = 10 × 2 = 20
Address is: 400 × 20 × 63 + 16 × 63 + 29 = 505037

Ans. (c)

55. The address of 1039th sector is

(a) <0, 15, 31> (b) <0, 16, 30>
(c) <0, 16, 31> (d) <0, 17, 31>

Solution:
Address of 1039th sector = 16 × 31 + 31 = 1039

Ans. (c)

56. A main memory unit with a capacity of 4 megabytes is built using 1M × 1-bit DRAM chips. Each DRAM chip has 1 K rows of cells with 1K cells in each row. The time taken for a single refresh operation is 100 ns. The time required to perform one refresh operation on all the cells in the memory unit is

(a) 100 nanoseconds
(b) 100×2^{10} nanoseconds
(c) 100×2^{20} nanoseconds
(d) 3200×2^{20} nanoseconds

(GATE 2010: 1 Mark)

Solution:
Main memory = 4 MB
Number of DRAM chips = 4 MB/ 1M × 1 bit = 32
Total cells = 32 × 1K × 1K
Time taken to refresh all the cells = 32 × 1K × 1K × 100 ns

Ans. (d)

57. The weight of a sequence a_0, $a_1/2$, … a_{n-1} of real numbers is defined as $a_0 + a_1/2 + \cdots + a_{n-1}/2^{n-1}$. A subsequence of a sequence is obtained by deleting some elements from the sequence, keeping the order of the remaining elements the same. Let X denote the maximum possible weight of a subsequence of a_0, a_1, …, a_{n-1} and Y the maximum possible weight of a subsequence of a_1, a_2, … a_{n-1}. Then X is equal to

(a) max(Y, $a_0 + Y$)
(b) max(Y, $a_0 + Y/2$)
(c) max(Y, $a_0 + 2Y$)
(d) $a_0 + Y/2$

(GATE 2010: 2 Marks)

Solution: The concepts involve the Dynamic Programming in Algorithms.

Given that
X = max weight from the sequence (a_0, a_1, a_2, … a_{n-1}) $= a_0 + \frac{a_1}{2} + \frac{a_2}{4} + \cdots + \frac{a_n - 1}{2^{(n-1)}}$

Y= max weight from the sequence (a_1, a_2, … a_{n-1})

$$= a_1 + \frac{a_2}{2} + \frac{a_3}{4} + \cdots + \frac{a_n - 1}{2^{(n-2)}}$$

Here we have to find X in terms of Y. So,

$$X = \left[a_1 + \frac{a_2}{2} + \frac{a_3}{4} + \cdots + \frac{a_{n-1}}{2^{(n-2)}}\right] = Y$$

$$\text{If } \left[a_0 + \frac{a_1}{2} + \frac{a_2}{4} + \cdots + \frac{a_{n-1}}{2^{(n-1)}}\right] < \left[a_1 + \frac{a_2}{2} + \frac{a_3}{4} + \cdots + \frac{a_{n-1}}{2^{(n-2)}}\right]$$

OR

$$X = \left[a_0 + \frac{a_1}{2} + \frac{a_2}{4} + \cdots + \frac{a_{n-1}}{2^{(n-1)}}\right] = a_0 + \frac{Y}{2}$$

$$\text{If } \left[a_0 + \frac{a_1}{2} + \frac{a_2}{4} + \cdots + \frac{a_{n-1}}{2^{(n-1)}}\right] > \left[a_1 + \frac{a_2}{2} + \frac{a_3}{4} + \cdots + \frac{a_{n-1}}{2^{(n-2)}}\right]$$

Hence, we sum up as

$$\text{X} = \text{MAX}(Y,\ a_0 + Y/2)$$

Ans. (b)

58. A 5-stage pipelined processor has instruction fetch (IF), instruction decode (ID), operand fetch (OF), perform operation (PO) and write operand (WO) stages. The IF, ID, OF and WO stages take 1 clock cycle each for any instruction. The PO stage takes 1 clock cycle for ADD and SUB instructions, 3 clock cycles for MUL instruction and 6 clock cycles for DIV instruction, respectively. Operand forwarding is used in the pipeline. What is the number of clock cycles needed to execute the following sequence of instructions?

Instruction	Meaning of Instruction
I_0: MUL R_2, R_0, R_1	$R_2 \leftarrow R_0 \times R_1$
I_1: DIV R_5, R_3, R_4	$R_5 \leftarrow R_3/R_4$
I_2: ADD R_2, R_5, R_2	$R_2 \leftarrow R_5 + R_2$
I_3: SUB R_5, R_2, R_6	$R_5 \leftarrow R_2 - R_6$

(a) 13 (b) 15 (c) 17 (d) 19

(GATE 2010: 2 Marks)

Solution:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
I_0	IF	ID	OF	PO	PO	PO	WO								
I_1		IF	ID	OF	_	_	PO	PO	PO	PO	PO	PO	WO		
I_2			IF	ID			OF	_	_	_	_	_	PO	WO	
I_3				IF			ID						OF	PO	WO

With operand forwarding, 15 clock cycles are needed.

Ans. (b)

59. The program below uses six temporary variables *a*, *b*, *c*, *d*, *e*, *f*.

```
a = 1
b = 10
c = 20
d = a + b
e = c + d
f = c + e
b = c + e
e = b + f
d = 5 + e
return d + f
```

Assuming that all operations take their operands from registers, what is the minimum number of registers needed to execute this program without spilling?

(a) 2 (b) 3
(c) 4 (d) 6

(GATE 2010: 2 Marks)

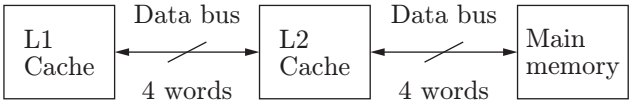
Solution: Let us take three Registers R1, R2 and R3

R1	R2	R3
a = 1	b = 10	c = 20
d = 11	b = 10	c = 20
d = 11	e = 21	c = 20
f = 41	e = 21	c = 20
b = 41	e = 21	c = 20
e = 42	b = 41	f = b = 41
d = 47	e = 42	f = 41

All the operations will be completed using three registers only.

Ans. (b)

Common Data Questions 60 and 61: A computer system has an L1 cache, an L2 cache, and a main memory unit connected as shown below. The block size in L1 cache is 4 words. The block size in L2 cache is 16 words. The memory access times are 2 nanoseconds, 20 nanoseconds and 200 nanoseconds for L1 cache, L2 cache and main memory unit, respectively.



60. When there is a miss in L1 cache and a hit in L2 cache, a block is transferred from L2 cache to L1 cache. What is the time taken for this transfer?

(a) 2 nanoseconds (b) 20 nanoseconds
(c) 22 nanoseconds (d) 88 nanoseconds

(GATE 2010: 2 Marks)

Solution: Access to L1 cache = 2 ns
Access to L2 cache = 20 ns
Block size of L2 is 16 words; data bus size is 4 words.
So, time taken for this transfer = $4 \times 22 = 88$ ns.

Ans. (d)

61. When there is a miss in both L1 cache and L2 cache, first a block is transferred from main memory to L2 cache, and then a block is transferred from L2 cache to L1 cache. What is the total time taken for these transfers?

(a) 222 nanoseconds (b) 888 nanoseconds
(c) 902 nanoseconds (d) 968 nanoseconds

(GATE 2010: 2 Marks)

Solution: Block transfer from main memory to L2 cache = $2 + 4 \times (20 + 200) = 2 + 880 = 882$ ns
L2 to L1 = $882 + 20 + 66 = 968$ ns

Ans. (d)

62. A computer handles several interrupt sources of which the following are relevant for this question.

- Interrupt from CPU temperature sensor (raises interrupt if CPU temperature is too high)
- Interrupt from Mouse (raises interrupt if the mouse is moved or a button is pressed)
- Interrupt from Keyboard (raises interrupt when a key is pressed or released)
- Interrupt from Hard Disk (raises interrupt when a disk read is completed)

Which one of these will be handled at the HIGHEST priority?

(a) Interrupt from Hard Disk
(b) Interrupt from Mouse
(c) Interrupt from Keyboard
(d) Interrupt from CPU temperature sensor

(GATE 2011: 1 Mark)

Solution: Interrupt from CPU temperature sensor will be handled at the highest priority.

Ans. (d)

63. Consider a hypothetical processor with an instruction of type LW R_1, 20 (R_2), which during execution reads a 32-bit word from memory and stores it in a 32-bit register R_1. The effective address of the memory location is obtained by the addition of a constant 20 and the contents of register R_2. Which of the following best reflects the addressing mode implemented by this instruction for the operand in memory?

(a) Immediate addressing
(b) Register addressing
(c) Register indirect scaled addressing
(d) Base indexed addressing

(GATE 2011: 1 Mark)

Solution: Effective address = contents of register $R_2 + 20$.

Ans. (d)

64. On a non-pipelined sequential processor, a program segment, which is a part of the interrupt service routine, is given to transfer 500 bytes from an I/O device to memory.

Initialize the address register
Initialize the count to 500
LOOP: Load a byte from device
Store in memory at address given by address register
Increment the address register
Decrement the count
If count!= 0 go to LOOP

Assume that each statement in this program is equivalent to a machine instruction which takes one clock cycle to execute if it is a non-load/store instruction. The load-store instructions take two clock cycles to execute.

The designer of the system also has an alternate approach of using the DMA controller to implement the same transfer. The DMA controller requires 20 clock cycles for initialization and other overheads. Each DMA transfer cycle takes two clock cycles to transfer one byte of data from the device to the memory.

What is the approximate speed up when the DMA controller based design is used in place of the interrupt-driven program-based input output?

(a) 3.4 (b) 4.4
(c) 5.1 (d) 6.7

(GATE 2011: 2 Marks)

Solution: By using load store instructions: $1 + 1 + 500 \times (2 + 2 + 1 + 1 + 1) = 2 + 500 \times 7 = 3502$
By using DMA: $20 + 500 \times 2 = 1020$
Speed up = 3502/1020 = 3.4

Ans. (a)

65. An 8 KB direct-mapped write-back cache is organized as multiple blocks, each of size 32 bytes. The processor generates 32-bit addresses. The cache controller maintains the tag information for each cache block comprising of the following:

1 Valid bit
1 Modified bit

As many bits as the minimum needed to identify the memory block mapped in the cache.

What is the total size of the memory needed at the cache controller to store metadata (tags) for the cache?

(a) 4864 bits (b) 6144 bits
(c) 6656 bits (d) 5376 bits

(GATE 2011: 2 Marks)

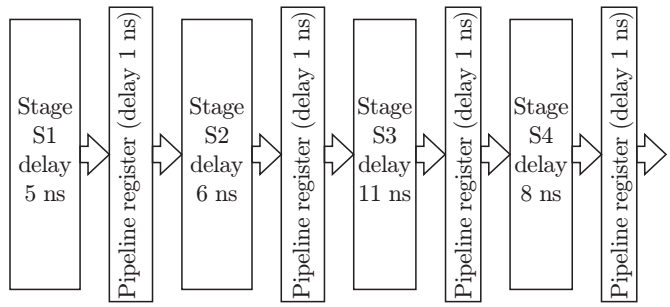
Solution: Number of blocks = 8K/32 = 256 blocks

Tag bits	Block bits	Word bits
19	8	5

Memory size for tag bits = 19 + 2 = 21
Total size of memory for tags = $21 \times 256 = 5376$ bits

Ans. (d)

66. Consider an instruction pipeline with four stages (S1, S2, S3 and S4) each with combinational circuit only. The pipeline registers are given in the figure.



What is the approximate speed up of the pipeline in steady state under ideal conditions when compared to the corresponding non-pipeline implementation?

(a) 4.0 (b) 2.5 (c) 1.1 (d) 3.0

Solution:

$$\text{Speed up} = \frac{\text{Time taken by non-pipelined}}{\text{Time taken by pipelined}} = \frac{5+6+11+8}{11+1} = 2.5$$

Ans. (b)

67. The decimal value 0.5 in IEEE single-precision floating-point representation has

(a) Fraction bits of 000…000 and exponent value of 0
(b) Fraction bits of 000…000 and exponent value of −1
(c) Fraction bits of 100…000 and exponent value of 0
(d) No exact representation

(GATE 2012: 1 Mark)

Solution: Binary representation of 0.5 is 0.1000.
which can be written as $= 1.000 \times 2^{-1}$
Here, exponent = −1 and fraction bits are 000…000.

Ans. (b)

68. Register renaming is done in pipelined processors

(a) As an alternative to register allocation at compile time
(b) For efficient access to function parameters and local variables
(c) To handle certain kinds of hazards
(d) As part of address translation

(GATE 2012: 1 Mark)

Solution: Register renaming is done to handle WAR/WAW hazards.

Ans. (c)

69. The amount of ROM needed to implement a 4-bit multiplier is

(a) 64 bits (b) 128 bits (c) 1 Kbits (d) 2 Kbits

Solution: Amount of ROM required
$= 2^{2k} \times 2k$ (where k = number of bits)
$= 2^{2\times4} \times 2 \times 4$
$= 2$ Kbits

Ans. (d)

70. A computer has a 256 KB, 4-way set-associative, write-back data cache with block size of 32 bytes. The processor sends 32-bit addresses to the cache controller. Each cache tag directory entry contains, in addition to address tag, 2 valid bits, 1 modified bit and 1 replacement bit.

The number of bits in the tag field of an address is

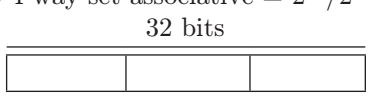
(a) 11 (b) 14 (c) 16 (d) 27

(GATE 2012: 2 Marks)

Solution:

$$\text{Number of blocks} = \frac{256 \text{ KB}}{32 \text{ B}} = 2^{13} \text{ blocks}$$

Due to 4-way set associative $= 2^{13}/2^2 = 2^{11}$



Ans. (c)

71. The size of the cache tag directory is

(a) 160 Kbits (b) 136 bits
(c) 40 Kbits (d) 32 bits

(GATE 2012: 2 Marks)

Solution: Tag directory contains: Tag bits + 4 additional bits = 20 bits
Size of cache tag directory $= 20 \times 2^{13} = 160$ Kbits

Ans. (a)

72. In a k-way set associative cache, the cache is divided into v sets, each of which consists of k lines. The lines of a set are placed in sequence one after another. The lines in set s are sequenced before the lines in set (s+1). The main memory blocks are numbered 0 onwards. The main memory block numbered j must be mapped to any one of the cache lines from

(a) (j mod v) * k to (j mod v) * k + (k-1)
(b) (j mod v) to (j mod v) + (k-1)
(c) (j mod k) to (j mod k) + (v-1)
(d) (j mod k) * v to (j mod k) * v + (v-1)

(GATE 2013: 1 Mark)

Solution: Number of sets = v
Number of main memory blocks = j
Number of lines = k (from 0 to k-1)
Position will be (j mod v)*k to (j mod v)*k +k-1

Ans. (a)

73. Consider the following sequence of micro-operations.

MBR ← PC
MAR ← X
PC ← Y
Memory ← MBR

Which one of the following is a possible operation performed by this sequence?

(GATE 2013: 2 Marks)

(a) Instruction fetch
(b) Operand fetch
(c) Conditional branch
(d) Initiation of interrupt service

Solution: Program counter value is stored in memory by MBR and gets a new address by Y. This indicates initialization of interrupt service routine.

Ans. (d)

74. Consider a hard disk with 16 recording surfaces (0-15) having 16384 cylinders (0-16383) and each cylinder contains 64 sectors (0-63). Data storage capacity in each sector is 512 bytes. Data are organized cylinder-wise and the addressing format is <cylinder no., surface no., sector no.>. A file of size 42797 KB is stored in the disk and the starting disk location of the file is <1200, 9, 40>. What is the cylinder number of the last sector of the file, if it is stored in a contiguous manner?

(GATE 2013: 2 Marks)

(a) 1281 (b) 1282 (c) 1283 (d) 1284

Solution: Number of sectors required to store the file $= \dfrac{42797 \times 1024}{512} = 85594$ sectors

Number of sectors in a cylinder $= 16 \times 64 = 1024$

Total number of cylinders required $= \dfrac{85594}{1024} = 84$

Last sector will be stored on 1284th cylinder.

Ans. (d)

75. Consider an instruction pipeline with five stages without any branch prediction: Fetch Instruction (FI), Decode Instruction (DI), Fetch Operand (FO), Execute Instruction (EI) and Write Operand (WO). The stage delays for FI, DI, FO, EI and WO are 5 ns, 7 ns, 10 ns, 8 ns and 6 ns, respectively. There are intermediate storage buffers after each stage and the delay of each buffer is 1 ns. A program consisting of 12 instructions I_1, I_2, I_3, … I_{12} is executed in this pipelined processor. Instruction I4 is the only branch instruction and its branch target is I9. If the branch is taken during the execution of this program, the time (in ns) needed to complete the program is

(a) 132 (b) 165 (c) 176 (d) 328

(GATE 2013: 2 Marks)

Solution: Clock pulse duration

$= \text{Maximum stage delay} + \text{Buffer delay}$

$= 11 \times 5 + (4 - 1) \times 11 = 88 \text{ ns}$

$n + k - 1 = 88 + 88 - 11 = 165$

Ans. (b)

Common Data Questions 76 and 77: The following code segment is executed on a processor which allows only register operands in its instructions. Each instruction can have atmost two source operands and one destination operand. Assume that all variables are dead after this code segment.

```
c = a + b;
d = c * a;
e = c + a;
x = c * c;
if (x > a) {
   y = a * a;
}
else {
  d = d * d;
  e = e * e;
}
```

76. Suppose the instruction set architecture of the processor has only two registers. The only allowed compiler optimization is code motion, which moves statements from one place to another while preserving correctness. What is the minimum number of spills to memory in the compiled code?

(a) 0 (b) 1
(c) 2 (d) 3

(GATE 2013: 2 Marks)

Solution: Suppose R_1 and R_2 hold a and b, respectively. Here for code optimization, code motion is allowed. The machine code is;

```
R2 ← R1 + R2     (c = a + b)
R2 ← R2 * R2     (x = c * c)
if (R2 > R1)
{
  R2 ← R1 * R1    [y = a*a]
}
else
{
  R2 ← c          [read value c from memory]
  R2 ← R2 * R1    [d = c * a;]
  R2 ← R2 * R2    [d = d * d;]
  R2 ← R2 * R1    [e = c + a;]
  R2 ← R2 * R2    [e = e * e;]
}
```

Only 'c' is spilled variable here, which needs to be stored and loaded from memory.

Ans. (b)

77. What is the minimum number of registers needed in the instruction set architecture of the processor to compile this code segment without any spill to memory? Do not apply any optimization other than optimizing register allocation.

(a) 3 (b) 4 (c) 5 (d) 6

(GATE 2013: 2 Marks)

Solution: Let $R_1 = a$, $R_2 = b$. The machine code is

```
  c = a + b;     R2 ← R1 + R2
  d = c * a;     R3 ← R2 * R1
  e = c + a;     R4 ← R2 + R1
  x = c * c;     R2 ← R2 * R2
if (x > a) {     R2 > R1
  y = a * a;     R1 ← R1 * R1
 }
else {
  d = d * d;     R3 ← R3 * R3
  e = e * e;     R4 ← R4 * R4
 }
```

Four registers are required.

Ans. (b)

PRACTICE EXERCISES

Set 1

1. The three different kinds of buses supporting the architecture of a computer are

(a) Address bus, data bus, I/O bus
(b) Front bus, data bus, address bus
(c) Control bus, data bus, address bus
(d) I/O bus, address bus, instruction bus

2. The term 'WORD' of a CPU equals to

(a) The maximum addressable memory size
(b) The width of a CPU register (integer or float point)
(c) The width of the address bus
(d) The total number of general purpose CPU registers

3. Which of the following statement is false about CISC ARCHITECTURE?

(a) CISC machine instructions may include complex addressing modes, which require many clock cycles to carry out.
(b) CISC control units are typically micro-programmed, allowing the instruction set to be more flexible.

(c) In the CISC instruction set, all arithmetic/logic instructions must be register based for fast processing.
(d) CISC architectures may perform better in network centric applications than RISC.

4. The register that holds the address of the location to or from which data are to be transferred is called

(a) Index register
(b) Accumulator
(c) Memory address registers
(d) Memory data registers

5. Which one of the following is not a type of I/O channel?

(a) Multiplexer (b) Selector
(c) Block multiplexer (d) None of the above

6. The performance of a pipelined processor is degraded if _____

(a) the pipeline stages have different delays
(b) consecutive instructions are to be executed serially
(c) the pipeline stages share hardware resources
(d) all of the above

7. The minimum time delay between the initiation of two independent memory operations is called

(a) Access time (b) Cycle time
(c) Rotational time (d) Latency time

8. The register which keeps track of the execution of a program and which contains the memory address of the instruction currently being executed is known as _____

(a) index register
(b) memory address register
(c) program counter
(d) instruction registers

9. For interval arithmetic, the best rounding technique used is _____

(a) rounding to plus and minus infinity
(b) rounding to zero
(c) rounding to nearest zero
(d) rounding to the next number

10. Hardwired control unit are faster than micro-programmed control unit because

(a) they do not consist of slower memory elements.
(b) they do not have slower elements such as gates, flip flops and registers.
(c) they consist of elements based on VVLSI design technology.
(d) they contain high-speed digital components.

11. The most relevant addressing mode to write position-independent code is

(a) direct mode (b) auto mode
(c) relative mode (d) indexed mode

12. A CPU uses 24-bit instruction. A program starts at address 300 (in decimal). Which one of the following is a legal program counter content (all values in decimal)?

(a) 324 (b) 512
(c) 600 (d) 700

13. An attempt to access a location not owned by a program is called

(a) data fault (b) address fault
(c) instruction fault (d) page fault

14. Which of the following statement about relative addressing mode is FALSE?

(a) It enables reduced program code
(b) It allows indexing of array element with same instruction
(c) It enables easy relocation of data
(d) It enables faster address calculation than absolute addressing

15. Compared to CISC processors, RISC processors contain

(a) more register and smaller instruction set
(b) larger instruction set and less registers
(c) less registers and smaller instruction set
(d) more registers and larger instruction set

16. Micro-programmed control cannot be implemented in RISC architecture because

(a) it tends to slow down the processor.
(b) it consumes more chip areas and large instruction set.
(c) handling a large number of registers is impossible in micro-programmed system.
(d) the 1 instruction/cycle timing requirement for RISC is difficult to achieve in micro-programmed based architecture.

17. Relocation of the code is easier in _____ irrespective of the program code

(a) indirect addressing
(b) indexed addressing
(c) base register addressing
(d) absolute addressing

18. In inverted page table organization, the size of the page table depends on

(a) the number of processes
(b) the size of page
(c) the size of main memory
(d) the number of frames in the main memory

19. When using the concept of locality of reference, the page reference being made by a process

(a) will always be to the page used in the previous page reference
(b) is likely to be one of the pages used in the past few page references
(c) will always to be one of the pages existing in the main memory
(d) will always lead to page fault

20. If the new version of processor is not made compatible to programs written for its older version, it could be able to process at a faster speed

(a) the statement is true.
(b) the statement is false.
(c) the speed cannot be predicted.
(d) speed has nothing to do with the compatibility.

21. A certain snooping cache can snoop only on address line. Which of the following is true?

(a) This would adversely affect the system if the write-through protocol is used.
(b) It would run well if the write-through protocol is used.
(c) Data snooping is mandatory to be implemented on data line.
(d) Data snooping may not be required.

22. When the frequency of the input signal to a CMOS gate is increased, the average power dissipation

(a) decreases exponentially
(b) increases
(c) decreases
(d) increases exponentially

23. The disadvantage of hardwired control units with flip flop is

(a) design becomes complex
(b) it requires more number of flip flops
(c) control circuit speed does not match with flip flops
(d) flip–flops can handle the data unit not the control unit

24. In a vectored interrupt,

(a) the branch address is assigned to a fixed location in memory.
(b) the interrupting source supplies the branch information to the processor through an interrupt vector.
(c) the branch address is obtained from a register in the processor.
(d) the branch address is obtained from program counter.

25. A device employing INTA line for device interrupt puts the CALL instruction on the data bus while

(a) $\overline{\text{INTA}}$ is active. (b) HOLD is active.
(c) $\overline{\text{READY}}$ is active. (d) READY is active.

26. On receiving an interrupt from an I/O device, the CPU

(a) branches off to halt (or wait) for a predetermined time
(b) branches off to the interrupt service after completion of the current instruction
(c) branches off to the interrupt service routine immediately
(d) hands over control of address bus and data bus to the interrupting device

27. Using large block size in a fixed block size file system leads to

(a) better disk throughput but poorer disk space utilization
(b) better disk throughput and better disk space utilization
(c) it does not matter as the total memory size is same
(d) poorer disk throughput but better disk space utilization

28. Which of the following statements are true about paging?

(a) It divides memory into units of equal size.
(b) It permits implementation of virtual memory.
(c) It suffers from internal fragmentation.
(d) It suffers from external fragmentation.

29. The number of entries in an inverted page table

(a) is equal to the number of processes.
(b) is equal to the number of page frames in the main memory.
(c) is equal to the size of the page frame.
(d) is equal to the number of page frames in cache memory.

30. In a virtual memory system, the addresses used by the programmer belongs to

(a) memory space (b) physical space
(c) address space (d) main memory space

31. Power consumption of processors can be vastly reduced by making use of _____ based transistors to implement the ICs.

(a) NMOS only
(b) TTL Schottky and PMOS
(c) PMOS only
(d) NMOS and PMOS

32. Address symbol table is generated by the

(a) memory management software
(b) assembler
(c) table match of associative memory
(d) generated by CPU

33. How many 128×8 RAM chips are needed to have a total RAM of 2048 bytes?

(a) 8 (b) 16 (c) 24 (d) 32

34. In 8085 microprocessor, how many I/O devices can be interfaced in I/O mapped I/O technique?

(a) Either 256 input devices or 256 output devices
(b) 8 I/O devices
(c) 256 input devices and 256 output devices
(d) 512 input-output devices

35. After reset, the CPU starts the execution of instruction from memory address

(a) 1111_H (b) 8000_H
(c) 0000_H (d) $FFFF_H$

36. In a microprocessor system, suppose TRAP, HOLD and RESET pin got activated at the same time, while the processor was executing some instructions, the system will

(a) execute the TRAP instruction
(b) execute the HOLD instruction
(c) execute the RESET instruction
(d) none of these instructions will be executed

37. In 8085 microprocessor, the programmer cannot access which flag directly?

(a) Sign flag (b) Carry flag
(c) Auxiliary carry flag (d) Parity flag

38. Which of the following is a pseudo-instruction for 8085?

(a) SPHL (b) CMP
(c) NOP (d) END

39. The term "cycle stealing" refers to:

(a) Interrupt-based data transfer
(b) DMA-based data transfer
(c) Polling mode data transfer
(d) Clock cycle overriding

40. Which of the following architecture is not suitable for the following SIMD architecture?

(a) Vector processor (b) PLA-based processor
(c) Von Neumann (d) PAL-based processor

41. In a multiprogramming system, which of the following concepts is used?

(a) Data parallelism (b) Paging
(c) L1 cache (d) DMA

42. PAL circuit can be defined as

(a) fixed OR and programmable AND logic.
(b) programmable OR and programmable AND logic.
(c) fixed AND and programmable OR logic.
(d) fixed OR and fixed AND logic.

43. If the clock input applied to a cascaded Mod-6 and Mod-4 counter is 48 kHz. Then the output of the cascaded arrangement shall be of:

(a) 4.8 kHz (b) 12 kHz
(c) 8 kHz (d) 48 kHz

44. If there are four ROM ICs of 8K and two RAM ICs of 4K words, then the address range of Ist RAM is (assume initial addresses correspond to ROMs)

(a) $(8000)_H$ to $(9FFF)_H$ (b) $(5000)_H$ to $(7FFF)_H$
(c) $(8000)_H$ to $(8FFF)_H$ (d) $(5000)_H$ to $(9FFF)_H$

45. The method for updating the main memory as soon as a word is removed from the cache is called

(a) Write-through (b) Write-back
(c) Write-save (d) Cache-save

Set 2

1. The most appropriate matching for the following pairs

X. Indirect addressing	1. Loops
Y. Immediate addressing	2. Pointers
Z. Auto-decrement addressing	3. Constants

(a) X – 3 Y – 2 Z – 1 (b) X – 1 Y – 3 Z – 2
(c) X – 2 Y – 3 Z – 1 (d) X – 3 Y – 1 Z – 2

2. Which of the following is not a form of memory?

(a) Instruction cache
(b) Instruction register
(c) Instruction opcode
(d) Translation look aside buffer

3. In serial data transmission, every byte of data is padded with a '0' in the beginning and one or two 1' s at the end of byte because

(a) Receiver is to be synchronized for byte reception.
(b) Receiver recovers lost '0' and '1' from these padded bits.
(c) Padded bits are useful in parity computation.
(d) None of these.

4. In 2's complement addition, overflow

(a) is flagged whenever there is carry from sign bit addition
(b) cannot occur when a positive value is added to a negative value
(c) is flagged when the carries from sign bit and previous bit match
(d) none of the above

5. The performance of a pipelined processor suffers if

(a) the pipeline stages have different delays
(b) consecutive instructions are dependent on each other
(c) the pipeline stages share hardware resources
(d) all of the above

6. Match the pairs in the following questions
By writing the corresponding letters only

(A) IEEE 488 (P) Specifies the interface for connecting a single device
(B) IEEE 796 (Q) Specified the bus standard for connecting a computer to other devices including CPUs
(C) IEEE 696 (R) Specifies the standard for an instrumentation bus
(D) RS232-C (S) Specifies the bus standard for the "back plane" bus called multibus

(a) (A) – (P); (B) – (R); (C) – (S); (D) – (Q)
(b) (A) – (P); (B) – (Q); (C) – (R); (D) – (S)
(c) (A) – (Q); (B) – (S); (C) – (R); (D) – (P)
(d) (A) – (R); (B) – (S); (C) – (P); (D) – (R)

7. When an interrupt occurs, an operating system

(a) Ignores the interrupt
(b) Always change state of interrupted process after processing the interrupt
(c) Always resumes execution of interrupted process after processing the interrupt
(d) May change state of interrupted process to 'blocked' and schedule another process.

8. RAID configuration of disks are used to provide

(a) Fault-tolerance (b) High speed
(c) High data density (d) None of the above

9. Arrange the following configuration for CPU in decreasing order of operating speeds: hardwired control, vertical microprogramming, horizontal microprogramming.

(a) Hardwired control, vertical microprogramming, horizontal microprogramming.
(b) Hardwired control, horizontal microprogramming, vertical microprogramming.
(c) Horizontal microprogramming, vertical microprogramming, hardwired control.
(d) Vertical microprogramming, horizontal microprogramming, hardwired control.

10. A processor needs software interrupt to

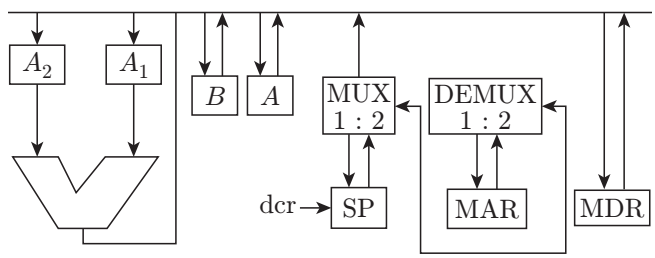
(a) test the interrupt system of the processor
(b) implement co-routines
(c) obtain system services which need execution of privileged instructions
(d) return from subroutine

11. Which is the most appropriate match for the items in the first list with the items in the second list?

List–I	List–II
X. Indirect addressing	I. Array implementation
Y. Index addressing	II. Writing relocatable code
Z. Base register addressing	III. Passing array as parameter

(a) (X, III) (Y, I) (Z, II) (b) (X, II) (Y, III) (Z, I)
(c) (X, III) (Y, II) (Z, I) (d) (X, I) (Y, IIII) (Z, II)

12. Consider the following data path of a simple non-piplelined CPU. The registers A, B, A_1, A_2, MDR, the bus and the ALU are 8-bit wide. SP and MAR are 16-bit registers. The MUX is of size 8 × (2:1) and the DEMUX is of size 8 × (1.2). Each memory operation takes 2 CPU clock cycles and uses MAR (memory address register) and MDR (memory data register). SP can be decremented locally



The CPU instruction "push r", where r = A or B has the specification

$$M[\text{sp}] \leftarrow r$$
$$\text{SP} \leftarrow \text{SP} - 1$$

How many CPU clock cycles are needed to execute the "push r" instruction?

(a) 2 (b) 3 (c) 4 (d) 5

13. In the absolute addressing mode,

(a) the operand is inside the instruction.
(b) the address of the operand is inside the instruction.

(c) the register containing the address of the operand is specified inside the instruction.
(d) the location of the operand is implicit.

14. What are the states of the auxiliary carry (AC) and carry flag (CY) after executing the following 8085 program?

MVI H, 5DH
MIV L, 6BH
MOV A, H
ADD L

(a) AC = 0, CY = 0 (b) AC = 1, CY = 1
(c) AC = 1, CY = 0 (d) AC = 0, CY = 1

15. Horizontal microprogramming

(a) does not require use of signal decoders
(b) results in larger-sized microinstructions than vertical microprogramming
(c) uses 1 bit for each control signal
(d) all of the above

16. For the daisy chain scheme of connecting I/O devices, which of the following statements is true?

(a) It gives non-uniform priority to various devices.
(b) It gives uniform priority to all devices.
(c) It is only useful for connecting slow devices to a processor device.
(d) It requires a separate interrupt pin on the processor for each device.

17. A microprogram control unit is required to generate a total of 25 control signals. Assume that during any microinstruction at most two control signals are active. Minimum number of bits required in the control word to generate the required control signal is _____.

18. The correct matching for the following pairs is

A. DMA I/O	1. High-speed RAM
B. Cache	2. Disk
C. Interrupt I/O	3. Printer
D. Condition code register	4. ALU

(a) A − 4, B − 3, C −1, D − 2
(b) A − 2, B − 1, C −3, D − 4
(c) A − 4, B − 3, C −2, D − 1
(d) A − 2, B − 3, C −4, D − 1

19. I/O redirection

(a) implies changing the name of a file
(b) can be employed to use an existing file as input file for a program
(c) implies connecting two programs through a pipe
(d) none of the above

20. The main difference(s) between a CISC and a RISC processor is/are that a RISC processor typically

(a) has fewer instruction
(b) has fewer addressing modes and more registers
(c) is easier to implement using hardwired control logic
(d) all of these

21. How big is a 4-way set-associative cache made up of 16 bits word, 4 words per line and having 1024 sets?

(a) 32 KB (b) 64 KB
(c) 128 KB (d) 256 KB

21. Consider the expression to be evaluated as $X = (M + N \times O)/(P \times Q)$. How many three address instruction are required to evaluate the above given expression?

(a) 3 (b) 4 (c) 5 (d) 6

23. In $X = (M + N \times O)/(P \times Q)$, how many one-address instructions are required to evaluate it?

(a) 4 (b) 6 (c) 8 (d) 10

24. A decimal number has 64 digits. The number of bits needed for its equivalent binary representation is

(a) 200 (b) 213 (c) 246 (d) 277

25. Determine the speed up obtained from pipelining if latencies for each stage in single cycle processor is given as:

IF	ID	ALU	MEM	WB
45 ns	20 ns	52 ns	44 ns	18 ns

ANSWERS TO PRACTICE EXERCISES

Set 1

1. (c) **3.** (c) **5.** (d) **7.** (b) **9.** (a)
2. (b) **4.** (c) **6.** (d) **8.** (c) **10.** (a)

11. (c)	**18.** (d)	**25.** (a)	**32.** (b)	**39.** (b)
12. (c)	**19.** (b)	**26.** (b)	**33.** (b)	**40.** (c)
13. (b)	**20.** (a)	**27.** (a)	**34.** (c)	**41.** (b)
14. (d)	**21.** (a)	**28.** (c)	**35.** (c)	**42.** (a)
15. (a)	**22.** (b)	**29.** (b)	**36.** (d)	**43.** (a)
16. (a)	**23.** (b)	**30.** (c)	**37.** (c)	**44.** (c)
17. (c)	**24.** (a)	**31.** (d)	**38.** (d)	**45.** (b)

Set 2

1. (c) Indirect addressing mode → Pointer access

Immediate addressing mode → Constant access
Auto decrement addressing mode → Loops

2. (c) An opcode is the portion of a machine language instruction that specifies the operation to be performed.

3. (a) Bits in the beginning and at the end of byte are known as start and stop bits, respectively. Start and stop bits are used for synchronization purpose.

4. (b) When two 2's complement numbers are added, overflow occurs in the following situations:

Both operands are positive and the result is negative.
Both operands are negative and the result is positive.
So option (b) is true, overflow does not occur when positive and negative numbers are added.

5. (d) Different hazards are caused due to various dependencies. Different dependencies for pipelined processor are as follows:

Structural dependency is due to different delays in pipelined stages.
Control dependency is due to consecutive instructions are dependent on each other.
Data dependency is due to hardware resources sharing.

6. (c) (A) IEEEE 488 – (Q)
(B) IEEEE 796 – (S)
(C) IEEEE 696 – (R)
(D) RS232-C – (P)

7. (c) When interrupt is caused, the execution of current instruction is stopped. After handling interrupt, the program resumes its execution.

8. (a) RAID is random array of independent disks that combines multiple disk drive components into a logical unit. RAID configuration provides fault-tolerance and high speed.

9. (b) Horizontal micro programming has high parallelism than vertical. So, the speed order is:

Hardwired control > horizontal microprogramming > vertical microprogramming

10. (c) A processor needs software interrupts to obtain system services which need execution of privileged instructions.

11. Pointers are used for accessing indirect addresses.

Indirect addressing mode → Pointer access
In case of immediate addressing, the constant represent the effective address.
Immediate addressing mode → Constant access
Auto decrement/increments are used for loop variables.
Auto decrement addressing mode → Loops

12. (a) Push '*r*' memory operation needs 2 clocks.

13. (b) In absolute addressing mode, the address of the operand is inside the instruction.

14. (c)

	Carry					Auxilary carry		
	0	1	1	1	1	1	1	1
(5D)#	(0	1	0	1	1	1	0	$1)_B$
+(6D)#	+ (0	1	1	0	1	0	1	$1)_B$
	(1	1	0	0	1	0	0	$0)_B$

AC = 1 and CY = 0

15. (d) Features of horizontal microprogramming are

(i) It does not require use of signal decoders
(ii) It results in larger-sized microinstructions than vertical microprogramming
(iii) It uses 1 bit for each control signal

16. (a) The daisy chaining method of establishing priority consists of a serial connection of all devices that request an interrupt. The device with the highest priority is placed in the first position, followed by lower-priority devices up to the device with the lowest priority, which is placed last in the chain. The farther the device is from the first position, the lower is its priority. Therefore, daisy chain gives non-uniform priority to various devices.

17. (10) To generate 25 control signals, 5 bits are required. To generate two control signals, the following scheme will be used.

5 bits to identify first and 5 bits to identify second including the case when one of them is not present. So total bits required = 10.

18. (b) DMA I/O – Disk

Cache – High-speed RAM
Interrupt I/O – Printer
Condition code register – ALU

19. (c) I/O redirection implies connection two programs through a pipe.

20. (d) The major characteristics of a RISC processor are

(i) Relatively few instruction
(ii) Relatively few addressing modes
(iii) More registers
(iv) Hardwired rather than microprogrammed control

21. (a) Number of bytes per line = 16 bit × 4 byte = 8 bytes

Cache size = 8 × 4 × 1024 = 32 KB

22. (b) Three-address instructions are as follow:

MUL R1, N, O
MUL R2, P, Q
ADD R3, M, R1
DIV X, R3, R2

23. (c) LOAD P (AC ← P)

MPY Q (AC ← AC × Q)
STORE X (X ← AC)
LOAD N (AC ← N)
MPY O (AC ← AC × O)
ADD M (AC ← AC + M)
DIV X (AC ← AC/X)
STORE X (X ← AC)

24. (b) The number of bits is

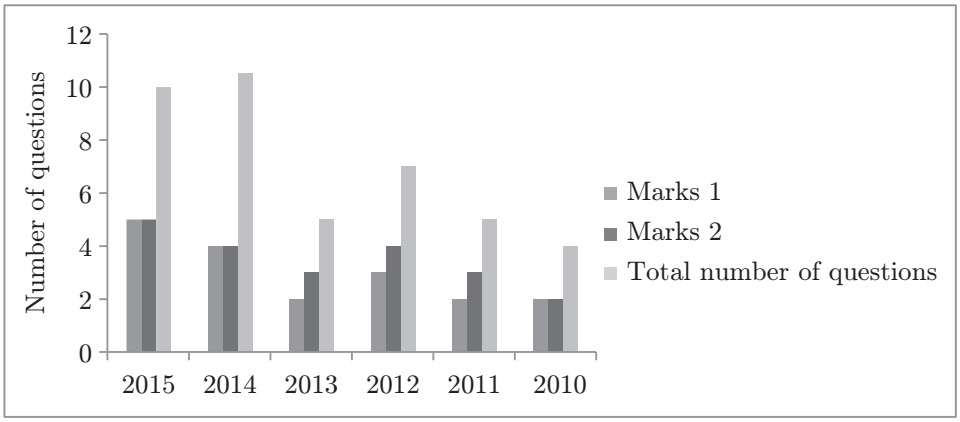
$$10^{64} - 1 = 2^x - 1$$
$$\Rightarrow 10^{64} = 2^x = x = \log_2 10^{64} \approx 213$$

25. (3.44) Speed up obtained by pipelining

$$= \frac{(45 + 20 + 52 + 44 + 18)}{52} = 3.44$$

UNIT III: PROGRAMMING AND DATA STRUCTURES

LAST SIX YEARS' GATE ANALYSIS



Concepts on which questions were asked in the previous six years

Year	Concept
2015	Binary tree, Minimum-maximum heap, B+tree, Functions, Recursion, Kth Smallest element in C, Hashing, Stack, C strings, Postfix equation, Binary search tree, Merge sort, Pointers, Switch-case, Storage classes, Quick sort, Array, Complexity
2014	Loops, Inorder traversal, Binary search tree, Tree traversal
2013	Binary search tree, Heap sort, Control statements, Recursion, Queues, Array and Pointer
2012	Switch case, Queue, Trees, Functions, Storage classes
2011	Array, Heaps, Binary search tree, Recursion
2010	Binary tree, Pointers, Heaps, Recursion, Link list

CHAPTER 3

PROGRAMMING AND DATA STRUCTURES

Syllabus: Programming and data structures: programming in C; functions, recursion, parameter passing, scope, binding; abstract data types, arrays, stacks, queues, linked lists, trees, binary search trees, binary heaps.

3.1 INTRODUCTION

A computer is a machine that manipulates information. Computer science is the study of how information is organised, manipulated and can be utilised. Data structure is a way of organising data in computer's memory or disk so that it can be used efficiently. Examples of several common data structures are arrays, linked lists, queues, stacks, binary trees and hash tables. Different kinds of data structures are suited to different kinds of applications, and some are highly specialised to specific tasks. Different types of data structures are used in different types of applications, which are specialised to perform the specific task. Data structures provide a means to manage large amounts of data efficiently, such as large databases and Internet indexing services. Data structures are designed to manage a large amount of data to suit a specific purpose so that it can be accessed and worked in an appropriate manner. Usually, efficient data structures are a key in designing efficient algorithms. In programming, efficient data structures are used to design efficient programs, both in terms of time and memory. In the following text, the programming part is discussed followed by data structure.

3.2 BASIC TERMINOLOGY

This section defines the basic terms used in computer programming.

3.2.1 Programming Languages

A computer performs the tasks commanded by the user. The command is given through a sequence of lines of code that are understood by the computer. The art of

writing such a code (or a program) is called programming. According to famous programmer Niklaus Wirth,

PROGRAMS = DATA + ALGORITHMS

Every programming language has two important pillars- *syntax* (refers to grammatical representation) and *semantic* (refers to the meaning). A correct program should be correct from syntactic view as well as from semantic view. The different kinds of translators check only the syntactic representation and the programmer takes care of the semantic representation. There are different generations of programming languages:

1. **First Generation/1GL/(Machine level programming):** The program is written in a sequence of 0s and 1s (binary number system). Such a language is more closer to the machine and the programs are machine dependent and difficult to debug. The only advantage is that there is no need for translator software (as the program is already in machine code) and execution is very fast.
2. **Second Generation /2GL/(Assembly level programming):** The program is written using instructions based on mnemonics and hexadecimal number system. Such a language is also machine-dependent. *Assembler* is the translator used to convert assembly level code to machine level code. The programs more reader friendly so are easier to debug.
3. **Third Generation /3GL/(High-level programming):** The program is written using English-like statements and decimal number system. Such a language is more closer to the user and the programs are machine independent. There is a need of translator (compiler/interpreter) to convert the high- level code to machine level code. Such languages are user-friendly. Such languages are *file-oriented*.
4. **Fourth Generation /4GL/(Non-procedural programming):** Such languages allow the user to specify the result instead of the describing how the result is to be obtained. All kinds of query languages belong to this generation. Such languages are *database-oriented*.
5. **Fifth Generation /5GL/(Natural Language Programming):** They are similar to 4GL and eliminates the user or programmer to learn a specific language. The language used, resembles the human speech closely. Example- CLOUT, Q & A, HAL (Human Access Language).

3.2.2 Classification of High-Level Languages

Depending upon their usage, the high-level languages are classified as: procedural, non-procedural and problem oriented languages.

1. **Procedural Languages:** These are those languages which follow a certain procedure. These are three different types-:
   - *Algorithmic languages:* The programmer specifies the steps (algorithm) to be followed for accomplishing a particular task. Such languages have built-in expressions, functions and procedures. For example, C, COBOL, PASCAL.
   - *Object-oriented languages:* Instead of using functions, objects are used. The data and the functions are combined in defining an object. Some of the important features are: Abstraction, Encapsulation, Polymorphism and Inheritance. For example, C++, Java.
   - *Scripting languages:* It is a kind of language that combines different components together to perform a difficult task. For example, Visual Basic script, PERL.
2. **Non-Procedural Languages:** 4GL languages are defined as non-procedural languages. These are two different types-:
   - *Functional language:* The program consists of functions, such languages are used in the field of artificial intelligence. For example, LISt Processing (LISP).
   - *Logic-based language:* It is a set of rules that are defined and the answer is retrieved using if−then rules. For example, PROgramming in LOGic (PROLOG).
3. **Problem-Oriented Languages:** Especially designed to solve certain kind of problems. These are of two different types:
   - *Numerical problems:* These contains the built-in functions, used to solve mathematical problems. For example, Mathematica.
   - *Publishing:* It contains the built-in functions, used in publishing industry. For example, LATEX.

3.2.3 Procedural Programming and Object-Oriented Programming

3.2.3.1 Procedural Programming

It uses a list of instructions in a stepwise manner to computer. It relies on procedures. Procedures are also known as routines or subroutines. A procedure consists of a series of computational steps that has to be carried out. Procedural programming languages are also known as top-down languages.

3.2.3.2 Object-Oriented Programming

Object-oriented programming (OOP) is a problem-solving approach in which all computations are carried

out using objects. An object is an entity of a program. It is used to perform actions and interactions with other elements of the program. These are the fundamental units of OOP, for example, a person.

The other fundamental concepts of OOP are as follows:

1. **Class:** It is a blueprint for an object. It does not define any data, rather it will define the object of the class it consists and the operations performed on the object.
2. **Abstraction:** It is used to provide essential information in a program without presenting the details. For example, in a database system, only data is shown, all the information regarding creating, storing, etc., is hidden.
3. **Encapsulation:** It is used to place data and functions at the same place.
4. **Inheritance:** It is the process of forming a new class from an existing class. The new class is known as the derived class and the existing class is known as the base class. It uses the concept of reusability of code.
5. **Polymorphism:** It is used to provide different meanings or functions to the existing operators or functions.
6. **Overloading:** It is a branch of polymorphism. When the existing operator or function is made to operate on a new data type, then it is known as overloading.

3.2.3.3 Structured Programming

A kind of programming approach that revolutionized the programming concept, proposed by E. Dijkstra'. In 1968, Dijkstra submitted the famous article in ACM Journal, highlighting the drawbacks of using the GOTO statement. He advocated that program written without the use of GOTO statement (or JUMP statement) were more stable. Nowadays, more and more programmers are following such kind of approach in writing programs. Other notable attributes of structured programming are:

1. **Top-down analysis:** A large problem is subdivided into smaller sub-problems.
2. **Modularization:** Dividing the program into small independent modules.

3.2.4 Data Types

It constitutes the type of values a variable can take and a set of operations that can be applied to those values. Data types in programming language can be broadly classified as shown in Fig. 3.1.

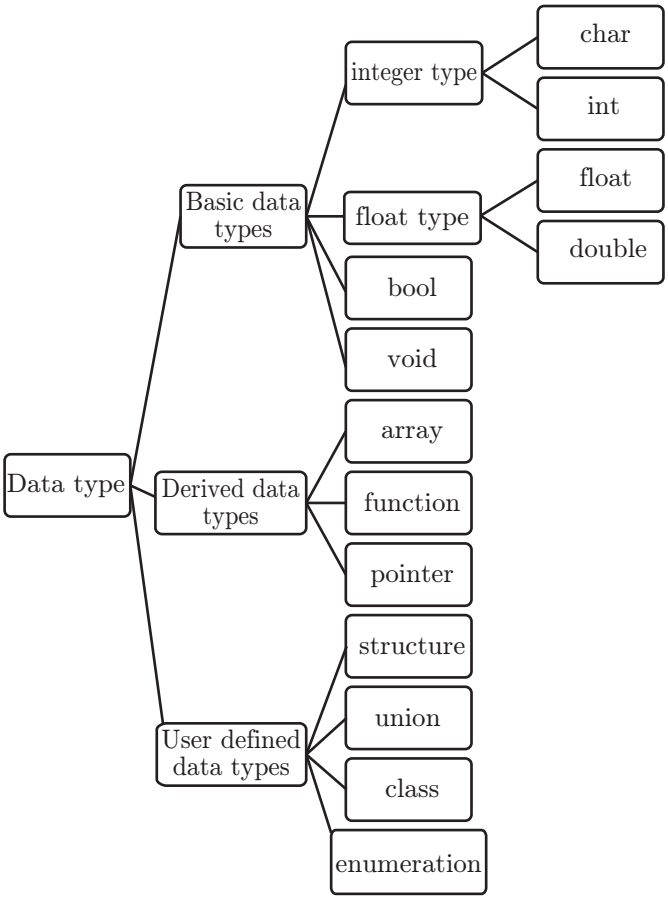


Figure 3.1 | Classification of data types in programming languages.

Table 3.1 gives detail about basic data types with its storage sizes and value ranges.

Table 3.1 | Basic data types, their storage size and value ranges

Type	Storage Size	Value Range (Depends on Compiler Type)
char	1 byte	−128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	−128 to 127
int	2 or 4 bytes	−32,768 to 32,767 or −2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295

(*Continued*)

Table 3.1 | Continued

Type	Storage Size	Value Range (Depends on Compiler Type)
short	2 bytes	−32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	−2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295
float	4 bytes	3.4E − 38 to 3.4E + 38
double	8 bytes	1.7E − 308 to 1.7E + 308
long double	10 bytes	3.4E − 4932 to 1.1E + 4932

C language is a weakly typed language so it allows implicit and explicit type conversions (Fig. 3.2). The compiler promotes each term in a binary expression to the highest precision operand.

3.2.5 Control Flow Statements

Programs consist of several statements and execution is not limited to a linear sequence of statements. During its execution, a program may repeat segments of code, or take decisions and branches. For that purpose, any programming language provides control flow statements that specifies the control flow of program or the order of execution of statements (Fig. 3.3).

3.2.5.1 Conditional Statements

These are used to execute a statement or a group of statements on the basis of certain conditions. These conditions are as follows:

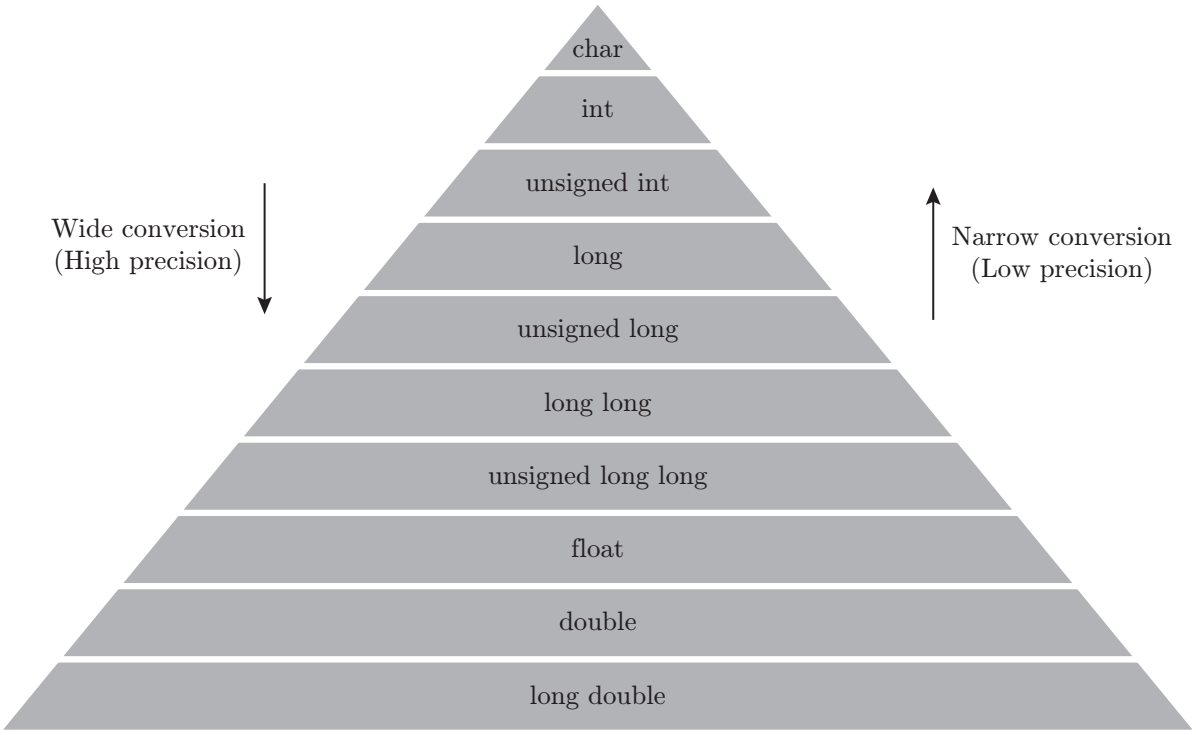


Figure 3.2 | Type conversion in C.

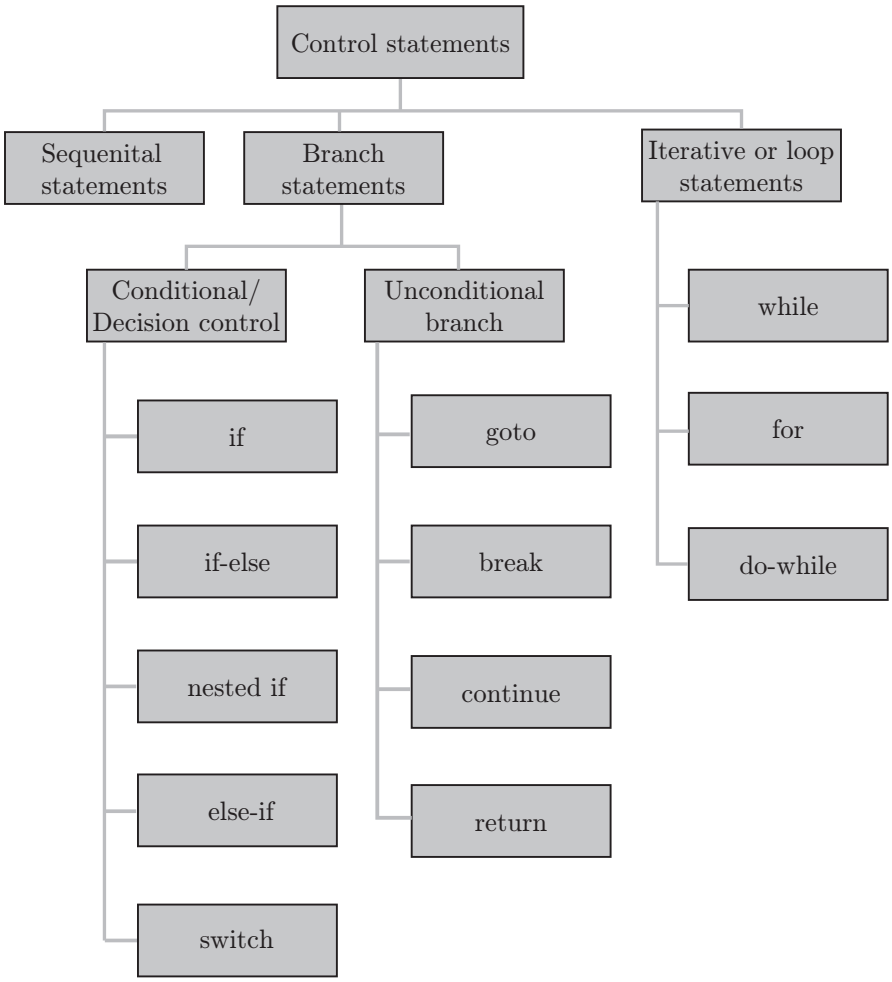


Figure 3.3 | Control flow statements.

1. **If:** When the condition is satisfied then the statements inside the parenthesis {} will be executed.
2. **If-Else:** If the condition is true then the statements between if and else will be executed. If it is false then the statement after else will be executed. In nested if, one or more if-else block(s) lie within the parent if statement.
3. **Else-If:** If the condition is true then the statements between if and else if will be executed. If it is false then the condition in else if is checked, and if it is true it will be executed.
4. **Switch:** It is also known as matching case statements. It is used to test for equality against a list of values. It matches the value in variable with any of the case inside the switch. If it matches the case defined in the switch then that match will be executed. If none of the cases match the statement then default will be executed.

3.2.5.2 Loops

These are used to execute a block of code several number of times. The statements are executed sequentially. Loop executes a statement or a group of statements multiple times. When the execution of statements in the loop is unknown, then this concept is known as odd loop.

Loops are of the following three types:

1. **while loop:** It repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
2. **for loop:** It executes a sequence of statements multiple times.
3. **do…while loop:** It is similar to while statement, except that it tests the condition at the end of the loop body.

Loops can be nested, that is, execution of one loop inside another loop.

There are various control statements used for changing execution from its normal sequence. These are as follows:

1. **break statement:** It terminates the loop or switch and transfers the execution to the statement immediately following the loop or switch.
2. **continue statement:** It skips the remainder of the loop body and immediately begins execution from the top.
3. **goto statement:** It transfers control to the labelled statement.
4. **return statement:** It transfers control from the function to the calling function.

3.2.6 Array

Array handles similar types of data. For example, storing marks of 100 students.

An array is a set of homogenous elements, which uses contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

3.2.6.1 Declaring an Array

Array can be declared as follows:

```
type arrayName [arraySize];
```

where

1. `type` represents the type of element an array stores. For example, if array stores integer elements then type of array is 'int'.
2. `arrayName` represents the name given to the array. It can be any string.
3. `arraySize` represents the number of elements the array stores. It is showed in `[]`.

For example, an array of 10 integers can be defined as follows:

```
int arr[10]
```

Arrays are of the following two types:

1. One-dimensional arrays
2. Multi-dimensional arrays

Initialisation of One-Dimensional Array

Arrays can be initialised at declaration time in this source code as follows (Fig. 3.4):

```
int age[5] = {2, 4, 35, 5, 8}
```

age[0]	age[1]	age[2]	age[3]	age[4]
2	4	35	5	8

Figure 3.4 | Initialisation of one-dimensional array.

Note that the first element is numbered 0 and so on.

Here, the size of the array *age* is five times the size of int because there are five elements.

Starting address of an array is also known as the base address of the array. Suppose, the starting address of age[0] is 1000 and the size of int is 4 bytes. Then, the next address (address of age[1]) will be 1004, address of age[2] will be 1008 and so on.

3.2.6.2 Important Things to Remember in C Arrays

Following are the important points for arrays:

1. Suppose you want to declare an array of 10 students. For example: arr[10]. Only array members from arr[0] to arr[9] are used. But, if element arr[10], arr[13], etc. are used, then the compiler may not show error but may cause fatal error during program execution.
2. The size of the array may not be defined during initialisation. For example,

   ```
   int age[] = {2, 4, 35, 5, 8};
   ```

   In this case, the compiler determines the size of the array by calculating the number of elements of an array.
3. If we do not provide value to any position in the array it will store 0 by default. For example,

   ```
   int age[5] = {2, 4, 35, 5};
   ```

   In this above case, first four places in the array will be filled by given values and the last will be filled by 0.

   ```
   int age[5]= { };
   ```

   Now array `age` is initialised with 0 at every position.

3.2.6.3 Address Calculation in an Array

The address can be calculated as follows:

One-Dimensional Arrays

In one dimension, an array '*A*' is declared as follows:

```
A[lb... ...ub]
```

where lb is the lower bound of the array and ub is the upper bound of the array.

Suppose we want to calculate the *i*th element address, then

```
Address (arr[i]) = BA + (i – lb) * c;
```

where BA is the base address of array, lb is the lower bound of array and c is the size of each element.

Two-Dimensional Arrays

In two dimensions, an array '*A*' is declared as follows:

```
A[lb1 ... ...ub1][lb2 ... ... ub2]
```

where lb_1 is the lower bound for row, lb_2 is the lower bound for column, ub_1 is the upper bound for row and ub_2 is the upper bound for column.

1. **Row Major Order:**

   ```
   Address(arr[i][j]) = BA+[i-lb1)*Nc+
                        (j-lb2)]*c
   ```

2. **Column Major Order:**

   ```
   Address(arr[i][j]) = BA+[j-lb2)*Nr+
                        (i-lb1)]*c
   ```

   where BA is the base address, Nr is the number of rows = $(lb_2 - lb_1 + 1)$ and Nc is the number of columns = $(ub_2 - ub_1 + 1)$.

Problem 3.1: Consider the following array:

```
a[1 ... ... 500]
```

Base address = 1000

Size of each element = 3 bytes

Find address of `a[397]`.

Solution:

```
Address (a[397]) = BA + (i - lb) * c
                 = 1000 + (397 - 1) * 3
                 = 1000 + 396 * 3
                 = 2188
```

Problem 3.2: Consider the following array:

```
a[50 ... ... 299][299 ... ... 500]
```

Base address = 0

Size of each element = 5

Find the address of a[250][499] by row major order.

Solution:

```
Address(arr[i][j]) = BA+[i-lb1)*Nc
                     +(j-lb2)]*c
Address(a[250][499]) = BA+[(250-50)*201
                       +(499-299)]*5
                     = 0+(200*202+200)*5
                     = 203000
```

3.2.7 Function and Recursion

A function is a group of statements, which is used to perform a task and recursion is a programming technique, which is used to express operations in terms of themselves.

3.2.7.1 Function

A function is a set of statements that together perform a task. Every C program has at least one function which is main(), and programs can have additional functions.

In other words, a function is a self-contained block, which takes some input and gives output after processing (Fig. 3.5). A function is known with various names such as a method or a sub-routine or a procedure, etc.

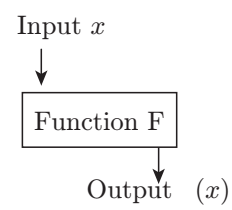


Figure 3.5 | A view of function.

Example 3.1

```
intavg(inta,intb,intc){
  float result;
  result = (a+b+c)/3;
  return result;
  }
```

Parts of a Function

A function has the following parts:

1. Function prototype
2. Definition of a function
3. Function call
4. Actual and formal arguments
5. Return statement

These parts have been discussed in detail as follows:

1. **Function prototype:** Function prototype contains function declaration, which tells the compiler about a function name, return type, argument types and function calling. Function declaration is necessary before calling it. A full declaration includes the return type and the type of arguments. Function declaration will be as follows:

   ```
   return_type function_name (parameter list);
   ```

For the above definition of function, the `avg()` function declaration is as follows:

```
intavg(int a, int b, int c);
```

In function declaration, the arguments are not important, only their type is required. So, the following declaration is also correct:

```
intavg(int, int, int);
```

2. **Definition of a function:** Function definition consists of the following parts:
   - *Function name*: This is the actual name of the function. It is unique. Function signature is constituted with the function name and parameter list together.
   - *Parameters*: At the time of function declaration, the parameters are passed into the function. The parameter list shows the type, value of parameter and number of arguments in the function. Parameters field is optional so that a function may contain no parameters.
   - *Function body*: The function body contains a collection of statements that defines what the task is performing by that function.
   - *Return type*: A function returns a value. The return type is that data type in which a function should return a value. Some functions perform the desired operations without returning a value. There are two possibilities, the function either returns a value or not. When the function does not return a value then the void data type used.

   In the above example, the function name is `avg`, the function has three integer-type parameters and the function calculates the average of those three numbers.
3. **Function call:** In function call, the control is transferred to the called function when the program calls the respective function. The name of the function must match with the function exactly with the name defined in the function prototype. A called function performs a defined task, and when its return statement is executed, it returns program control back to the main program. To call the function, the first step is to pass the arguments with function name, and then the return value stores in a specific manner.
4. **Actual and formal arguments:** A parameter is a special kind of variable used in a function to refer to the data provided as input to the function. These pieces of data are called arguments. Syntactically, we can pass any number of parameters to a function. Parameters are specified within a pair of parenthesis. These parameters are separated by commas (,). Parameter written in a function definition is called 'formal parameter'. Parameter written in a function call is called 'actual parameter'.
5. **Return statement:** When a return statement gets executed, then the function returns the value and transfers the control to the calling function. Execution continues in the calling function by following the remaining statements. A return statement can also return a value to the calling function.

Types of C Functions

The following are two types of functions in C on the basis of whether it is defined by a user or not:

1. **Pre-defined function:** The function whose definition is already stored in the library of the respective language is called pre-defined function. These functions are provided in the system or particular language library. For example, in C language, printf(), scanf(), gets(), puts() are available in stdio.h header file. These functions are also called library functions or built-in functions.
2. **User-defined function:** User-defined functions are functions created by the user at the time of writing the program for modularity purpose. For example,

```
int max (int num1, int num2){
int result;
if(num1 > num2)
      result = num1;
else
      result = num2;
return result;
}
```

Function Advantages

Function has the following advantages:

1. It provides the reusability of code in an easy manner.
2. It makes debugging and editing easier.
3. It reduces the size of a program.
4. It makes the logic of the program easier to design and understand.

3.2.8.2 Recursion

When the function calls itself then it is called recursion. In C language, there is provision for the function to call itself. When using the recursion, it is necessary to have exit condition from the function; otherwise, the result will be infinite.

Writing Recursive Functions

A recursive function has the following two parts:

1. **Base case:** It is the non-recursive solution and a stopping condition.
2. **Recursive case:** It is the recursive solution of the problem.

It is simply a specification of the general recursive function we have seen many times.

```
return_typefunction_name(Pass appropriate
  arguments)
{
 if a simple case, return the simple value
  /* base case or stopping condition*/
 else call function itself with simpler
  version of problem
}
```

Example 3.2

```
int sum(int n){
if(n==0)  /* base case or stopping
  condition */
return n;
else
returnn+sum(n-1); /*self-call to function
  sum() */
}
```

In this program, sum() function is called itself in else condition. If n is equal to 0 then the function returns the value passing in the argument, but if n is not equal to 0 then the function calls itself. Let $n = 5$ initially, then next time 4 is passed to the function and the value of the parameter is decreased by 1 until the condition is satisfied. At the end, when n becomes 0, the value of n is returned, which is from 5 to 1.

$$
\begin{aligned}
&\text{sum}(5)\\
&= 5 + \text{sum}(4)\\
&= 5 + 4 + \text{sum}(3)\\
&= 5 + 4 + 3 + \text{sum}(2)\\
&= 5 + 4 + 3 + 2 + \text{sum}(1)\\
&= 5 + 4 + 3 + 2 + 1 + \text{sum}(0)\\
&= 5 + 4 + 3 + 2 + 1 + 0\\
&= 5 + 4 + 3 + 2 + 1\\
&= 5 + 4 + 3 + 3\\
&= 5 + 4 + 6\\
&= 5 + 10\\
&= 15
\end{aligned}
$$

Advantages of Recursion

Recursion has the following advantages:

1. Unnecessary calling of functions is avoided.
2. In recursion, the function calls anywhere in the program so it if flexible in nature.
3. In some situations, the recursion is compulsory in a programming. For example, to find the factorial of a given number.

3.2.8 Pointers

A pointer is a variable which stores the address of some other variables. The declaration of a pointer is done as follows:

```
Data type *variable_name;
```

Example 3.3

```
int *p; /* pointer to an integer */
int b = 20;
p = &b; /* address of variable b is stored
  in p */

printf ("Address of variable b is: %d", &b);
printf ("Address of variable b is: %d, p);
printf ("Value of variable b is: %d", *p);
  /* *p displays the value at stored address */
```

The operating systems may stop to access memory and cause the program to crash. To avoid such problem, pointers should always be initialized before use. These are initialized using free memory in the following way:

1. **Memory allocation to pointers:**

   ```
   int *p = malloc (sizeof (*p));
   ```

2. **De-allocating memory of pointer:**

   ```
   free (p);
   ```

3. **Pointer to pointer:**

```
int *p    /* pointer to an integer */
int **ptr  /* pointer to pointer */
int b = 20;
p = &val;   /* address of variable b is
  stored in p */
ptr = &p;   /* address of pointer is
  stored in ptr */

To print value of b:
*p = 20;
**ptr = 20;
```

Pointer Arithmetic

1. **Incrementing a pointer:** Incrementing a pointer variable depends on data type of the pointer variable. It is generally used in array in which elements are stored at contiguous memory locations. For example,

   ```
   ptr ++ = ptr + 1 = ++ ptr = ptr + 1 *
     size_of(data type);
   (address + 1= ++ address = address++
     gives an address value)
   ```

```
#include<stdio.h>
int main(){
int *ptr=(int *)200; /*Assume integer
  takes 4 bytes*/
ptr=ptr+1;
printf("New Value of ptr : %u",ptr);
return 0;
}
```

Output

```
New Value of ptr: 204
```

2. **Decrementing a pointer:** Decrementing a pointer to a data variable will cause its value to be decremented by size of variable because memory required to store a variable (integer) varies from compiler to compiler.Example

```
ptr-- = ptr - 1 = --ptr = ptr - 1 *
  size_of(data type);
(address - 1= --address = address--
  gives an address value)
```

3. **Adding an integer value with pointer:** In C Programming, it is legal to add any integer number to pointer variable.

```
ptr + n = (ptr) + (n * size of data type);
```

```
#include<stdio.h>
int main()
{
int *ptr=(int *)100;
ptr=ptr+3;
printf("New Value of ptr : %u",ptr);
return 0;
}
```

Output

```
New Value of ptr : 112
```

4. **Subtracting an integer value from pointer:** In C programming, it is legal to subtract any integer number from pointer variable.

```
ptr - n = (ptr) - (n * size of data type)
```

```
#include<stdio.h>
int main()
{
int *ptr=(int *)100;
ptr=ptr - 3;
printf("New Value of ptr : %u",ptr);
return 0;
}
```

Output

```
New Value of ptr : 88
```

5. **Subtracting two pointers:** It means subtracting two pointers and it gives total number of two objects between them.

```
Actual Result = (ptr2 - ptr1) / Size
  of Data Type
```

```
Suppose the address of variable n is
  200 and integer takes 4 bytes.
```

```
#include<stdio.h>
int main()
{
    int n, *ptr1, *ptr2;
    ptr1 = &n;
    ptr2 = ptr1 + 2;
    printf("Difference is: %d",ptr2 -
      ptr1);
    return 0;
}
```

Output

```
Difference is 2
ptr1 = 200
ptr2 = 200 + 2 * 4 = 208
ptr2-ptr1 = (208-200)/4 = 2
Both pointers numerically differ by 8
  and technically by 2 objects
```

3.2.8.1 Dangling Pointer

Dangling pointer arises when an object is deleted or de-allocated, without modifying the value of pointer, so that the pointer still points to the memory location of the de-allocated memory. In other words, a pointer pointing to a non-existing memory location is known as a dangling pointer. For example,

```
void main ()
  {
      int *p;
          {
              int i=10; //This is a Block.
              p=&i;//Variable i is local
                so i's lifetime is only
                within the block.
          }
      printf ("%d", *p); //p will point to
        de-allocated memory.
      *p=*p+100;
      printf ("%d",*p);
  }
```

This program is perfectly running and giving output as 100 200. Actually this should not have happened because in this program p points to a location **i** which has been deleted from the memory so **p** becomes a dangling pointer but still the program is running.

Sometimes, the dangling pointer constitutes an undefined behaviour as may be the reason that memory has not been overwritten by anything else. For example,

```
int *demoFun()//returns the address of
  variable i
  {
      int i=10;
      return &i;
  }
int main()
 {
  int *p;
  p=demoFun();   /*function returning
    address is assigning to pointer
    variable.*/
   printf("%d",*p);      //10
   *p=*p+20;
   printf("%d",*p);      // Garbage Value
 }
```

Output

```
- 10
Some Garbage Value
```

According to the program, variable **i** have been deleted after completion of `demoFun()` function but still the pointer is able to access the memory location of **i** because this memory is not overwritten by another variable. It must be the same reason that dangling pointer constitutes an undefined behaviour. But when we perform another operation, complier gives some garbage value because base pointer **p** does not get any data to that particular location.

3.2.9 Parameter Passing

There are various types of parameter-passing techniques in programming languages. The parameter passed is based on the program requirement. According to the parameter passing, the passing parameter depends on the way the calling function calls the called function.

The C programming language supports the following two types of parameter-passing techniques:

1. Call by value
2. Call by reference

3.2.9.1 Call-by-Value Parameter-Passing Technique

When we call a function, we pass the parameter to the called function and it is by default passed by the value of variable in the calling function. This is called 'call by value'.

Example 3.4

```
void exchange(int x, int y);
main()
{
int a = 10, b = 20;
printf("\na = %d b = %d", a, b);
exchange(a, b);
printf("\na = %d b = %d", a, b);
}

exchange(int x, int y)
{
int t;
t = x;
x = y;
y = t;
printf("\nx = %d y = %d", x, y);
}
```

Before calling the function exchange(), values of a and b are 10 and 20, respectively. After calling exchange(), x and y also receive 10 and 20, respectively. In exchange(), we are interchanging values of x and y. So, the new values of x and y are 20 and 10, respectively. After successful execution of the function exchange(), control transfers to the main function, but in the main function, values of a and b remain unchanged, that is, 10 and 20, respectively.

So, the output of the above program is as follows:

```
a=10 b=20
x=20 y=10
a=10 b=20
```

In the call-by-value mechanism, only the formal argument (variables receiving value in called function from calling function, that is, x and y in the above program) gets changed and do not reflect in actual argument (variable in function call, that is, a and b in function call to exchange(a, b)).

3.2.9.2 Call-by-Reference Parameter-Passing Technique

In call by reference, the parameter is passing through the actual address of the value. The address of the variable never changes throughout the execution of the program. There is a possibility to access and modify the value of the variable by using a pointer. In case of any changes in the parameter inside the function, there is a change of the actual parameter. For example, the declaration of a variable is as follows:

`int x = 20;` which tells the compiler:

1. To allocate 2-byte memory location on stack frame of that particular function in which it is declared.
2. To hold value of type integer and also associate this address with name x.
3. This memory location is initialised with value 20.

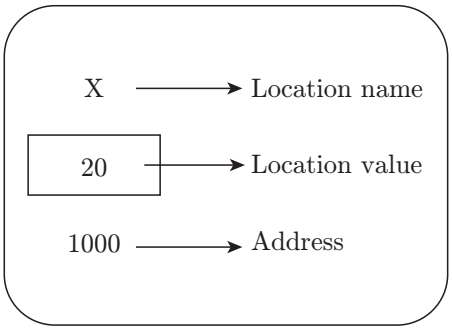


Figure 3.6 | Memory map in call by reference.

The selection of address for a particular variable is compiler dependent and is always a whole number. In Fig. 3.6, the value 20 can be accessed by using variable name x and address 1000. The symbol '&' is used to represent the 'address of'.

For example, expression &x in the above case returns address of variable x, which happens to be 1000.

An address is always positive, we use operator %u for unsigned integer. *' is a special operator in C used as 'value at address'. It is also called indirection operator.

For example, *(&x), which means value at address(&x), that is, value at address (1000). The example given below returns the address of a variable, the value of the variable and the value of variable using indirection operator.

Example 3.5

```
#include<stdio.h>
void main( )
{
int x = 20;
printf("\nAddress of x = %u", &x);
printf("\nValue of x = %d", x);
printf("\nValue of x = %d", *( &x ));
}
```

Output

```
Address of x = 1000
Value of x = 20
Value of x = 20
```

```
/* Call by Reference Parameter passing
  technique */
#include<stdio.h>
void exchange(a,b);
void main( )
{
int a = 10, b = 20;
printf("\na = %d b = %d", a, b);
exchange(&a, &b);
printf("\na = %d b = %d", a, b);
}
void exchange(int *x, int *y)
{
int t;
t = *x;
*x = *y;
*y = t;
}
```

Output

```
a = 10 b = 20
a = 20 b = 10
```

Consider the declaration of a variable. `int *x;` does not mean that x is going to hold integer value, rather it simply means it is going to hold address (memory location) of integer type of variable. We are passing address of variables a, b to function exchange(), so while receiving we are using two pointers to integer.

Generally, we use call-by-value mechanism if we do not need the changes made in calling function to reflect in the called function. But if we require the changes to reflect in the called function, we use the call-by-reference mechanism.

Also, if we want to return more than one value to the called function from the calling function, we go for call-by-reference value, which is not possible ordinarily.

Problem 3.3: Find the output of the following programs

(a)

```
#include<stdio.h>
void main()
{
   int a=3, *b;
   b = &a;
   printf("%d\n", a**b*a+*b);
}
```

(b)

```
#include<stdio.h>
int main()
{
    int a=10, *b, *c;
    b=&a; /* Assume address of a is 100
      and integer is 4 byte size */
    c=b;
    *b++=*c++;
    a++;
    printf("a=%d, b=%d, c=%d\n", a,
      b, c);
    return 0;
}
```

(c)

```
#include<stdio.h>
int main()
{
   char s[10] = "Point";
   char *const p=s;
   *p='J';
   printf("%s\n", s);
   return 0;
}
```

Solution:

The outputs of the programs are:

(a) `30` (b) `a=11, b=104, c=104` (c) `Point`

3.2.10 Structures and Unions

Structures are used to represent a record (e.g. library). A union is a special data type in C, which is used to store different data types in the same memory location. Table 3.2 shows the comparison of the structures and union.

Table 3.2 Comparison between structure and union

Structure	Union
1. To define a structure, the '`struct`' keyword is used.	1. To define union, the '`union`' keyword is used.
2. When a structure variable is created, the compiler allocates memory for this variable equal to the sum of size of structure elements.	2. When a union variable is created, the compiler allocates memory for that variable equal to the maximum size of union element.
3. Each structure element within a structure is assigned to unique memory location.	3. Individual members of the union share the memory allocated.
4. More than one structure element can be initialised at once.	4. Only one member can be initialised at a time.
5. Individual members can be accessed at a time.	5. Only one member can be accessed at a time.

3.2.11 Enumerated Data Types

Enumerated data types assume values which are previously declared. For example,

```
enum month {jan, feb, mar, apr, may, jun,
  jul, aug, sep, oct, nov, dec};
enum month this_month;
this_month = apr;
```

3.2.12 Scoping

The scope is a part of a program where the identifier may have direct access, and beyond that scope, a variable cannot be accessed. In programming, variables can be declared in the following three ways:

1. Inside the main function
2. Outside the main function
3. Function parameter definition

3.2.12.1 Local Variables

A local variable is that variable which is declared in the body of the main function. It can be used only by the function in which it declares. Other function has no information about local variable. Formal parameter of a function is also treated as a local variable to that function. In the following example, num, sum, r are local variables.

Example 3.6

```
#include<stdio.h>
int main(){
/* Local Variable Declaration*/
int num,sum=0,r;
printf("Enter a number: ");
scanf("%d",&num);
 while(num){
      r=num%10;
      num=num/10;
      sum=sum+r;
  }
  printf("Sum of digits of number: %d",sum);
 return 0;
}
```

Variables num, sum and r are local to the function main(), and cannot be accessible outside the main function.

3.2.12.2 Global Variables

Global variables are those variables which are defined outside the body of the main function. The global variables will hold their value throughout the lifetime of the program. They can be accessed inside any of the functions defined for the program. Following is an example using local variable and global variable.

Example 3.7

```
#include<stdio.h>
/* Global Variable Declaration*/
int r;
int main(){
/* Local Variable Declaration*/
 int num,sum=0;
 printf("Enter a number: ");
  scanf("%d",&num);
  while(num){
      r=num%10;
      num=num/10;
      sum=sum+r;
  }
  printf("Sum of digits of number: %d",sum);
  return 0;
}
```

Variables num, sum are local to the function main(), and cannot be accessible outside the main function. But the variable *r* is declared outside the main as a global variable, it can be accessed by any function in this program.

If local variable and global variable have the same name, then preference will be given to the value of the local variable inside the function.

Example 3.8

```
#include<stdio.h>
/* Global Variable Declaration*/
int a=10;
int main(){
/* Local Variable Declaration*/
  int a=5,b=20,sum;
  sum= a+b;
  printf("Sum of number: %d",sum);
 return 0;
}
```

In the above example, variable '*a*' is declared as global and local both. And variable '*b*' is declared as local to the main function. When the sum of '*a*' and '*b*' are calculated by a compiler then priority is given to that '*a*' which is declared as local. So the program will print 25 as the sum of the variables.

3.2.12.3 Storage Classes

A storage class defines the scope and visibility of variables and functions in C program. The following four types of storage classes are available:

1. Register
2. Auto
3. Static
4. Extern

Storage, default value, scope and life of these four storage classes are given in Table 3.3.

Table 3.3 | Storage, default value, scope and life of the storage classes

Class	Auto	Register	Static	Extern
Storage	Memory	CPU registers	Memory	Memory
Default	Garbage value	Garbage value	Zero	Zero
Scope	Local to block	Local to block	Local	Global
Life	Until block	Until block	Persists between different function calls	Till the end of program

3.2.13 Binding

It is an association of an attribute with a program component in a program. For example, the data type of the value of a variable is an attribute, which is associated with the variable name. The binding time for an attribute is the time at which the binding occurs.

In C programming, the binding time for a variable type is when the program is compiled, but the value of the variable is not bound until the program executes.

3.2.13.1 Early Binding

In early binding, the linker copies the referenced module into the executable image of the program at compilation/link time. The referenced module is a part of the executable image.

1. **Advantages of early binding:**
   - It is simple and the best performer.
   - It is at least twice as fast as late binding.
   - The executable is stand alone.
2. **Disadvantages of early binding:**
   - The executable image is large.
   - If the referenced module changes then each executable must be re-linked.

- If there is more than one executable using the module at the same time, then multiple copies of it are used in memory.

3.2.13.2 Late Binding

In late binding, the linker copies only a stub code for the referenced module into the executable image. It is also known as dynamic binding. The stub code loads the referenced module into memory at load/run time.

1. **Advantages of late binding:**
   - The size of the executable image is small.
   - On changing the referenced module, re-link of the executable is not required.
   - Most operating systems can share the referenced module.
2. **Disadvantages of late binding:**
   - The executable is dependent on the shared library at runtime.
   - Since the module is shared, it must be re-entrant and thread safe.

3.2.14 Abstract Data Types

An abstract data type is an object with a specification of the components. It is independent of implementation details.

Examples:

1. **Stack:** Operations such as push an item onto the stack, pop an item from the stack, ask if the stack is empty; implementation may be as array or linked list.
2. **Queue:** Operations such as add to the end of the queue, delete from the beginning of the queue, ask if the queue is empty; implementation may be as array or linked list or heap.
3. **Search structure:** Operations such as insert an item, ask if an item is in the structure and delete an item; implementation may be as array, linked list or tree.

3.3 STACK

A stack is an ordered collection of items in which the item is added and removed in order. In stack, the element is added on the top of the stack. Only one element, which is placed on the top, is removed from the top of the stack. The stack is called last-in first-out (LIFO) structure. Stack can be implemented by linked list and array.

The basic operations on stack are as follows

1. **PUSH:** To add an element to the stack.
2. **POP:** To remove an element from the stack.

3.3.1 PUSH Operation on Stack

When we try to add elements to the stack then the operation is called PUSH operation on stack (Fig. 3.7). The element is placed only on the top of the stack and then the stack position gets increased by 1.

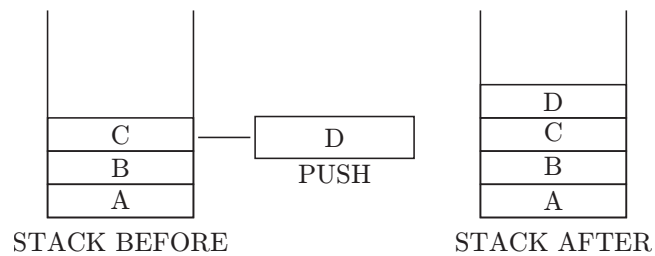


Figure 3.7 PUSH operation on stack.

3.3.1.1 Code for PUSH Operation

```
void push(int token)
{
char a;
if(top==MAX-1)
    {
printf("Stack full");  → is Full Condition
return;                   check
    }
do
    {
printf("\nEnter the token to be inserted:");
scanf("%d",&token);
top=top+1;
stack[top]=token;
printf("do you want to continue insertion
  Y/N");
        a=getch();
    }
while(a=='y');
}
```

3.3.2 POP Operation on Stack

Whenever we try to remove an element from the stack then the operation is called POP operation on stack (Fig. 3.8).

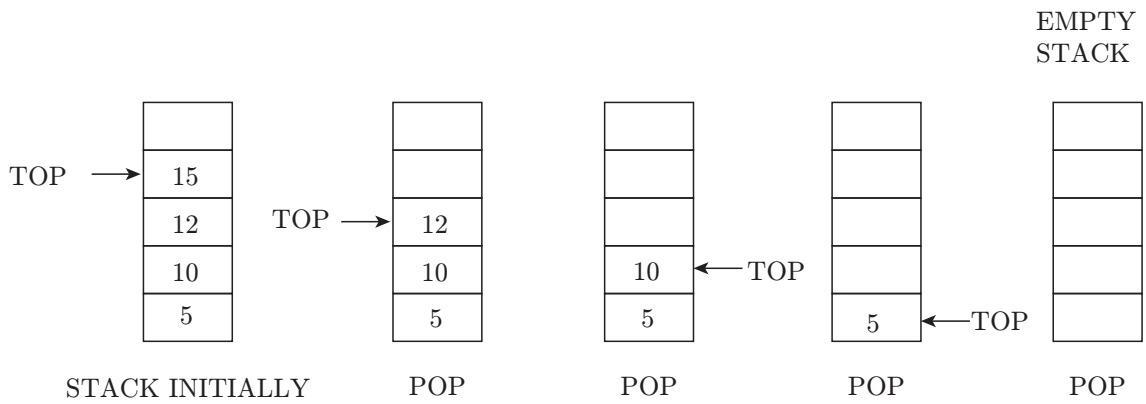


Figure 3.8 | POP operation on stack.

3.3.2.1 Code for POP Operation

The following is the code of POP function in which we use the return statement to send the result back to the calling function

```
int pop()
{
int t;
if(top==-1)
     {
printf("Stack empty");  → Is Empty Condition
return -1;                  check
     }
     t=stack[top];
top=top-1;
return t;
}
```

3.3.3 Application of Stack

Following are the applications on stack:

1. **Parsing:** Stacks are used by compilers to check the syntax of a program and for generating executable code.
2. **Reversing a list:** Stack can be used for reversing a list.
3. **Calling function:** When a function is called all local storage for the function is allocated on the stack and return address is also stored on the stack.
4. **Recursive function:** Stack can be used for implementing recursive functions.
5. **Expression conversion:**
   - *Infix*: An infix expression is one in which operators are located between their operands. This is how we are accustomed to writing expressions in standard mathematical notation.
   - *Postfix*: In postfix notation, the operator immediately follows its operands.
   - *Prefix*: In prefix notation, operands immediately follow operator.

Infix to Postfix Conversion

Algorithm for infix to postfix conversion:

1. You create a stack and an output string.
2. Then you read the infix string one token at a time, each token is either an identifier or an operator:
   - If the token is an identifier (e.g. a variable or a constant), then you append it onto the end of the output string.
   - If the token is an operator, then you compare its precedence with the precedence of the operator on the top of the stack.
     (a) If the operator on the top of the stack has lower precedence, then push the new operator onto the stack.
     (b) Else, pop the operator from the top of the stack and append it to the output string. Then push the new operator onto the stack.
     (c) Otherwise,
       (i) If the token is an open parenthesis, then push it onto the stack.
       (ii) If the token is a close parenthesis, then pop and append the operators from the stack until the open parenthesis is popped. Discard both parentheses.

Problem 3.4: Convert the given expression into postfix.

$$\text{Expression} = (A + B) * (C - D)$$

Solution:

Expression	Stack	Output	Comment
$(A + B) * (C - D)$	Empty		Initial
$A + B) * (C - D)$	(		Push
$+ B) * (C - D)$	(	A	Print
$B) * (C - D)$	(+	A	Push
$) * (C - D)$	(+	AB	Print
$* (C - D)$		$AB+$	Pop until (and print
$(C - D)$	*	$AB+$	Push
$C - D)$	*(	$AB+$	Push
$- D)$	*(	$AB + C$	Print
$D)$	*(−	$AB + C$	Push
)	*(−	$AB + CD$	Print
End	*	$AB + CD-$	Pop until (
End	Empty	$AB + CD-$	Pop every element

Infix to Prefix Conversion

Algorithm for infix to prefix conversion:

1. Reverse the given infix expression.
2. Make every '(' as ')' and every ')' as '('.
3. Apply algorithm infix to postfix on the expression that comes from step 2.
4. And finally reverse the output of step 3.

Problem 3.5: Convert the given expression into prefix:

Expression = $(A + B\,\hat{}\,C) * D + E\,\hat{}\,5$

Solution:

Step 1: Reverse the infix expression.

$5\,\hat{}\,E + D*)\ C\,\hat{}\,B + A\ ($

Step 2: Make every '(' as ')' and every ')' as '('

$5\,\hat{}\,E + D * (C\,\hat{}\,B + A)$

Step 3: Convert expression to postfix form.

Expression	Stack	Output	Comment
$5\,\hat{}\,E + D * (C\,\hat{}\,B + A)$	Empty	−	Initial
$\hat{}\,E + D * (C\,\hat{}\,B + A)$	Empty	5	Print
$E + D * (C\,\hat{}\,B + A)$	^	5	Push
$+D * (C\,\hat{}\,B + A)$	^	$5E$	Push

(*Continued*)

Table Continued

Expression	Stack	Output	Comment
$D * (C\wedge B + A)$	$+$	$5E\wedge$	Pop and push
$*(C\wedge B + A)$	$+$	$5E\wedge D$	Print
$(C\wedge B + A)$	$+*$	$5E\wedge D$	Push
$C\wedge B + A)$	$+*($	$5E\wedge D$	Push
$\wedge B + A)$	$+*($	$5E\wedge DC$	Print
$B + A)$	$+*(\wedge$	$5E\wedge DC$	Push
$+ A)$	$+*(\wedge$	$5E\wedge DCB$	Print
$A)$	$+*(+$	$5E\wedge DCB\wedge$	Pop and push
$)$	$+*(+$	$5E\wedge DCB\wedge A$	Print
End	$+*$	$5E\wedge DCB\wedge A+$	Pop until '('
End	Empty	$5E\wedge DCB\wedge A+*+$	Pop every element

Step 4: Reverse the expression.

$+*+A\wedge BCD\wedge E5$

3.4 QUEUE

Queue is a special abstract data type storage structure. The access to elements in a queue is restricted, unlike arrays. Enqueue and dequeue are two main operations used for the insertion of element in a queue and removal of an element from a queue, respectively. An item can be inserted at the end known as rear of the queue and removed from the end known as front of the queue. It is also called a first-in first-out (FIFO) list.

It has the following five properties:

1. **Capacity:** It stands for the maximum number of elements a queue can hold.
2. **Size:** It stands for the current size of the queue.
3. **Elements:** It is the array of elements.
4. **Front:** It is the index of the first element (the index at which the element has been removed).
5. **Rear:** It is the index of the last element (the index at which the element has been inserted).

3.4.1 Basic Operations on Queue

Queue has the following basic operations:

1. **Enqueue:** Inserts item x at the rear of the queue q. Queue can be declared as follows:

```
voidEnqueue(queue, x)
{
  if(front==0&& rear==MAX-1)
{
    printf("\n Queue Overflow ");
}
  elseif(front==-1&&rear==-1)
  {
      front=rear=0;
      queue[rear]=x;

  }
  else
  {
      rear++;
      queue[rear]=x;
  }
}
```

2. **Dequeue:** Removes the front element from q and returns its value. It can be declared as follows:

```
voidDequeue(queue)
{
  intx;
  if(front==-1)
  {
      printf("\n Underflow");
  }
  x=queue[front];
  if(front==rear)
{
     front=rear=-1;
}
else
  {
    if(front==MAX-1)
      front=0;
```

```
        else
          front++;
           printf("\n The deleted element
             is: %d", x);
      }

    }
```

3.4.2 Types of Queue

Queue is of the following types:

1. **Simple queue or linear queue:** In this data structure, operations, such as, insertion and deletion occurs at the rear and front of the list, respectively (Fig. 3.9).

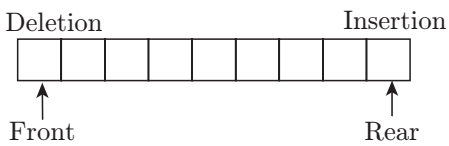


Figure 3.9 | Simple queue.

2. **Circular queue:** A circular queue is a queue where all nodes are treated as circular (Fig. 3.10). The first node follows the last node.

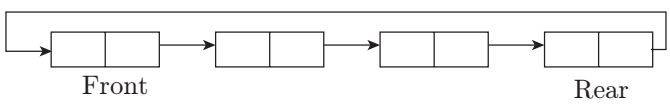


Figure 3.10 | Circular queue.

3. **Priority queue:** It contains items which have some preset priority (Fig. 3.11). When an element has to be removed from a priority queue; the item with the highest priority is removed first.

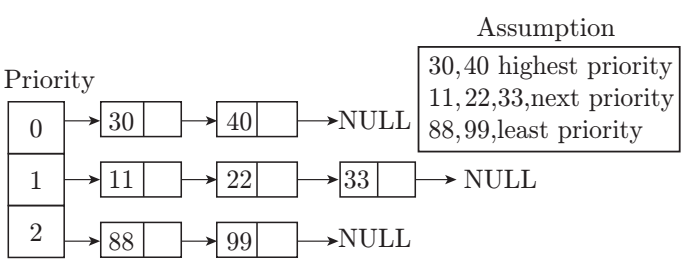


Figure 3.11 | Priority queue.

4. **Dequeue (double-ended queue):** In this, insertion and deletion occur at both the ends, that is, front and rear of the queue (Fig. 3.12).

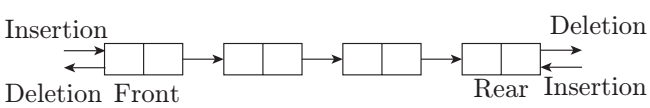


Figure 3.12 | Double-ended queue.

3.4.3 Applications of Queues

Queues have many applications in computer systems:

1. Jobs in a single-processor computer
2. Print spooling
3. Information packets in computer networks

3.5 LINKED LIST

Linked list is a dynamic data structure whose length can be increased or decreased at runtime. Linked list basically consists of memory blocks that are located at random memory locations. Node of a linked list consists of two parts, one is value or data and the other is pointer to the next node.

3.5.1 Basic Operations

Basic operations of a singly-linked list are as follows:

1. **Create:** Creates an empty node of linked list.
2. **Insert:** Inserts a new element at the end of the list.
3. **Delete:** Deletes any node from the list.
4. **Find:** Finds any node in the list.
5. **Print:** Prints the list.

Functions of these operations are listed as follows:

1. **Insert:** It takes the start node and data to be inserted as arguments. The new node is inserted at the end of the linked list, till it reaches the last node. Allocate the memory for the new node and put data in it. Store the address in the next field of the new node as NULL.
2. **Delete:** It takes the start node (as pointer) and data to be deleted as arguments. Reach the node for which the node next to it has to be deleted.
   - If that node points to NULL (i.e. pointer → next = NULL) then the element to be deleted is not present in the list.
   - Else, the pointer points to a node, whose next node has to be removed, declare a temporary node (temp), which points to the deleted node. Store the address of the node next to the temporary node in the next field of the node pointer (pointer → next = temp → next).

   Break the link of the node which is next to the pointer (which is also temp). Function free() will deallocate the memory.
3. **Find:** It takes the start node (as pointer) and data value of the node (key) to be found as arguments. The first node is dummy node, so start with the second node. Iterate through the entire linked list and search for the key. Until next field of the pointer is equal to NULL, check if pointer → data = key.

- If the condition holds, then the key is found.
- Else, move to the next node and search (pointer = pointer → next).
- If key is not found return 0, else return 1.

4. **Display:** Function takes the start node (as pointer) as an argument. If pointer = NULL, then there is no element in the list. Else, print the data value of the node (pointer → data) and move to the next node by recursively calling the print function with pointer → next sent as an argument.

3.5.2 Linked List Implementation

Linked list is implemented as follows:

1. **Node Structure of Linked List:**

```
structnode
{
    int data;
    structnode *next;
}*Start;
```

2. **Insert Node in Linked List:**

```
void insert(Start,x)
{
    structnode *temp,*New_Node;
    New_Node=(structnode *)
      malloc(sizeof (structnode));
    New_Node->data=num;
New_Node->next=NULL;

    temp=Start;
    if(temp==NULL)
    {
    Start=New_Node;
    exit(1)
    }
    else
    {
    while(temp->next!=NULL)
    {
        temp=temp->next;
    }
    Start=New_Node;
    }
}
```

3. **Delete Node in Linked List:**

```
int delete(int num)
{
    structnode *temp, *prev;
    temp=Start;
    while(temp!=NULL)
    {
    if(temp->data==num)
    {
        if(temp==Start)
        {
            Start=temp->next;
            free(temp);
            return 1;
            }
            else
            {
            prev->next=temp->next;
            free(temp);
            return 1;
            }
    }
    else
    {
        prev=temp;
        temp=temp->next;
    }
    }
    return 0;
}
```

4. **Find a Node in Linked List:**

```
int find (Struct node *Start,x)
{
struct node *temp;
temp=start;
if(temp==NULL)
{
printf("List is empty");
exit();
}
else
{
while(temp->data!=x && temp->
  next!=NULL)
{
temp=temp->next;
}
if(temp->data==x)
{
printf(" Node is found at address
  %u",temp);
return 1;
}
else
{
printf(" Node is not found ");
return 0;
}
}}
```

5. **Display Linked List:**

```
void display(start)
{
    r=start;
    if(r==NULL)
    {
    return;
    }
```

```
    while(r!=NULL)
    {
    printf("\n%d ",r->data);
    r=r->next;
    }
    printf("\n");
}
```

3.6 TREES

A tree is a collection of nodes and edges. When we connect each node with the help of edges then it constructs a tree. The following has the properties of a tree:

1. Starting node is called the root node.
2. Except the root node every other node(n) in a tree is surely connected by an edge from another node(y), the other node(y) is called the parent node of n. There is unique path from start node to end node.

3.6.1 Binary Tree

A binary tree is a tree data structure. In a binary tree, any node in the tree can have at most two children. Binary tree are used to construct binary search trees and binary heap. An example of a binary tree is shown in Fig. 3.13.

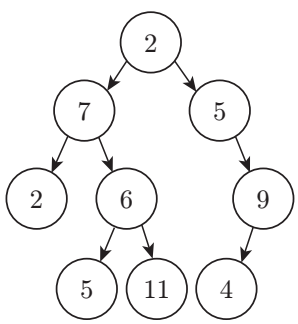


Figure 3.13 | Binary tree.

1. **Vocabulary and definitions:** We will formally define a tree and its components.
2. **Node:** A node is an important part of a tree. It can have a name, which we call the 'key'. It may also have additional information.
3. **Edge:** An edge is another important part of a tree. It connects two nodes to show a relationship between them. Every node (except the root) is connected with exactly one incoming edge from another node. Each node may have several outgoing edges.
4. **Root:** The root of the tree does not have incoming edges.
5. **Path:** A path is an ordered list of nodes, which are connected by edges. For example, $2 \rightarrow 7 \rightarrow 6 \rightarrow 11$ is a path.
6. **Children:** It is a set of nodes (S), which have incoming edges from the same node. In Fig. 3.13, 2 and 6 are the children of node 7.
7. **Parent:** It is the parent of all the nodes that are connected to the outgoing edges. In Fig. 3.13, node 5 is parent of node 9.
8. **Sibling:** The nodes in the tree having the same parent are said to be siblings. The nodes 2 and 6 are siblings in the tree shown in Fig. 3.13.
9. **Subtree:** It is a set of nodes and edges comprised of a parent and all the descendants of that parent.
10. **Leaf node:** It is a node that has no children. For example, node 5 and node 11 are the leaf node in Fig. 3.13.
11. **Level:** The level of a node (n) is the number of edges on the path from the root node to (n). For example, the level of node 4 in Fig. 3.13 is three.
12. **Height:** It is equal to the maximum level of any node in the tree. The height of the tree in Fig. 3.13 is three.

3.6.2 Types of Binary Trees

Binary trees are of the following types:

1. **Strictly binary tree:** Figure 3.14 shows the Strictly binary tree. It has the following properties:
   - If in a tree, non-leaf nodes have exactly two children then that tree is called strictly binary tree.
   - In strictly binary tree, every leaf nodes have degree 0 and every non-leaf nodes have degree 2.
   - A strictly binary tree with n leaf nodes has $(2n-1)$ total number of nodes. Thus, strictly binary tree always have odd number of nodes.

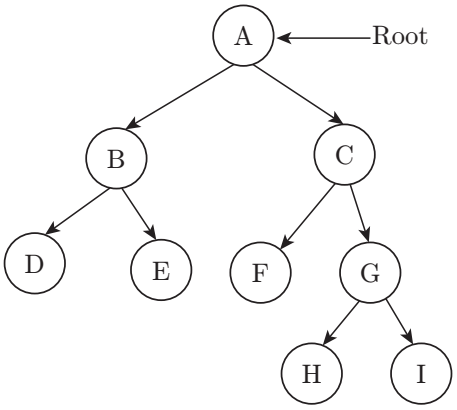


Figure 3.14 | Strictly binary tree.

2. **Complete binary tree:** Figure 3.15 shows the strictly binary tree. It has the following properties:

- A complete binary tree of depth d is strictly a binary tree having all the leaves at level d.
- The total number of nodes in a complete binary tree of depth d equals $(2^{d+1} - 1)$.

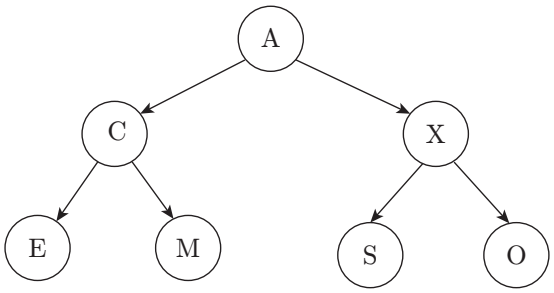


Figure 3.15 | Complete binary tree.

3. **Almost complete binary tree:** A binary tree of depth d is an almost complete binary tree (see Fig. 3.16) if:
   - Each leaf in the tree is at either level d or level $d - 1$.
   - For any node n_d in the tree with a right descendant at level d, all the left descendants of n_d that are leaves are also at level d.
   - An almost complete binary tree with N leaves that is not strictly binary has $2N$ nodes.

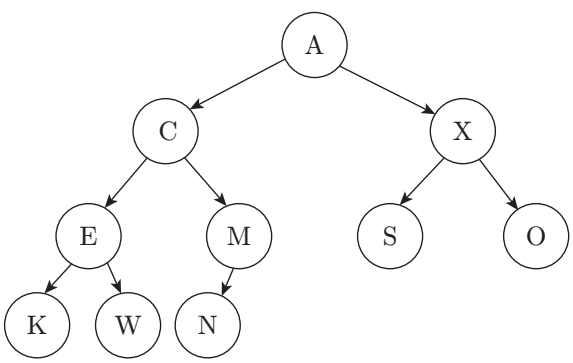


Figure 3.16 | Almost complete binary tree.

3.6.3 Array Representation of Binary Trees

An array can be used to store a binary tree using the following mathematical relationships.

If root data is stored at index n, then left child is stored at index $2*n$ and right child is stored at index $2*n + 1$.

Figure 3.17 demonstrates the storage representation.

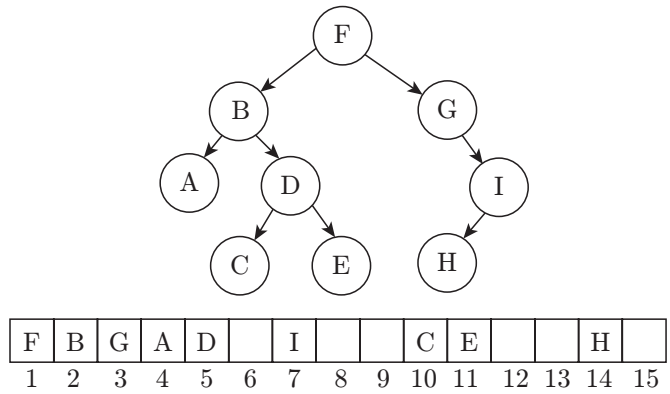


Figure 3.17 | Storage representation.

3.6.4 Tree Applications

The following are common uses of a tree:

1. Manipulate hierarchical data
2. Make information easy to search
3. Manipulate sorted lists of data
4. As a workflow for compositing digital images for visual effects
5. Router algorithms

3.6.5 Binary Search Tree

Binary search tree is a node-based binary tree data structure with the following properties:

1. The left subtree contains the nodes with keys less than the node's key.
2. The right subtree contains the nodes with keys greater than the node's key.
3. Both the right and left subtree should also be binary search tree.
4. There should not be any duplicate nodes.

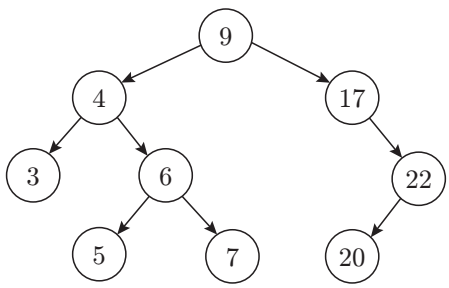
3.6.5.1 Tree Traversal in Binary Search Tree

There are three types of traversal techniques for tree:

1. **Preorder:**
   - Traverse the root
   - Traverse all the left external nodes starting with the left-most subtree
   - Traverse the right subtree starting at the left external node
2. **Postorder**
   - Traverse all the left external nodes starting with the left-most subtree
   - Traverse the right subtree starting at the left external node
   - Traverse the root

3. **Inorder**
   - Traverse the left-most subtree starting at the left external node
   - Traverse the root
   - Traverse the right subtree starting at the left external node

Problem 3.6: Consider the given tree and find the inorder, preorder and postorder of the tree.



Solution:

Inorder Traversal:

The inorder traversal of the above tree will output:

3, 4, 5, 6, 7, 9, 17, 20, 22

Preorder Traversal:

The preorder traversal of the above tree will output:

9, 4, 3, 6, 5, 7, 17, 22, 20

Postorder Traversal:

The postorder traversal of the above tree will output:

3, 5, 7, 6, 4, 20, 22, 17, 9

3.6.6 Graphs

A graph $G = (V, E)$ is composed of V (set of vertices) and E (set of edges), where edges connect the vertices. For example, see Fig. 3.18.

$V = \{a, b, c, d, e\}$ and $E = \{(a, b), (a, c), (a, d), (b, e), (c, d), (c, e), (d, e)\}$

3.6.6.1 Basic Terminology

Following are the basic terms used for graphs (see figure 3.19):

1. **Undirected graphs:** It consists of a set V of vertices and a set E of edges, each of them connecting to two different vertices.
2. **Directed graph:** In this, edges provide a connection from one vertex to another, but not necessarily in the opposite direction.
3. **Adjacent vertices:** Connected by an edge.
4. **Degree of a vertex:** It is defined as the number of adjacent vertices.
5. **Path:** It is defined as a sequence of vertices such that consecutive vertices are adjacent.
6. **Simple path:** Path with no repeated vertex.
7. **Cycle:** A simple path, where the first and last vertex are same.
8. **Connected graph:** When any two vertices are connected by some path.
9. **Subgraph:** When a subset of vertices and edges form a graph.
10. **Tree:** A connected graph without cycles.
11. **Balanced binary tree:** A binary tree in which no leaf is at much greater depth than any other leaf. Example of balanced binary search trees are red-black trees, AVL trees, etc. Examples of non-binary balanced search trees are B trees, B+ trees, etc.

3.6.6.2 Traversing a Graph

There are two types of traversal techniques for graph:

1. **Breadth first traversal:** It is a level-by-level traversal of an ordered tree. It starts with vertex I, then it traverses to neighbours of vertex i, then neighbours of neighbours of vertex i and so on. It uses a queue.

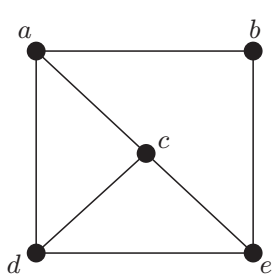


Figure 3.18 | A graph $G = (V, E)$.

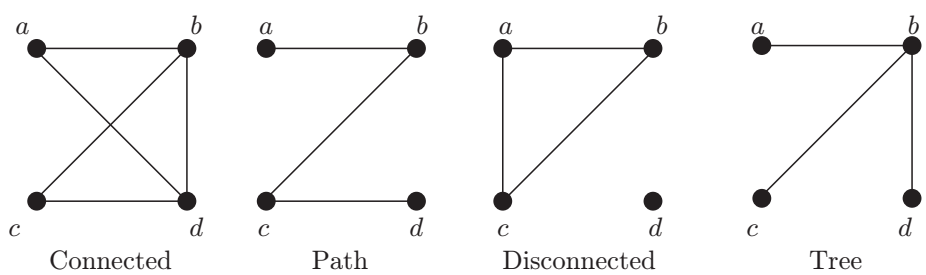
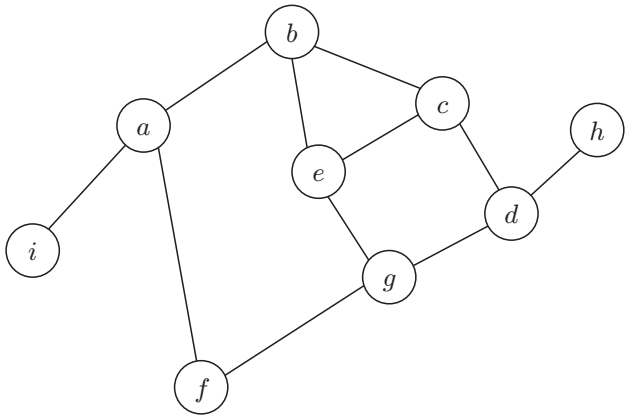
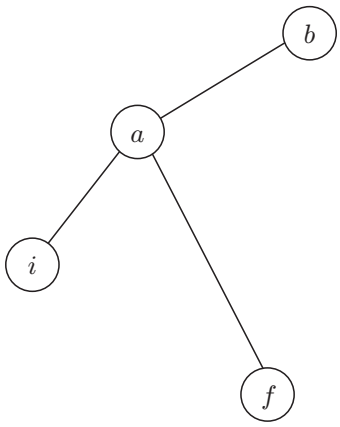


Figure 3.19 | Examples showing connected, path, disconnected and tree graphs.

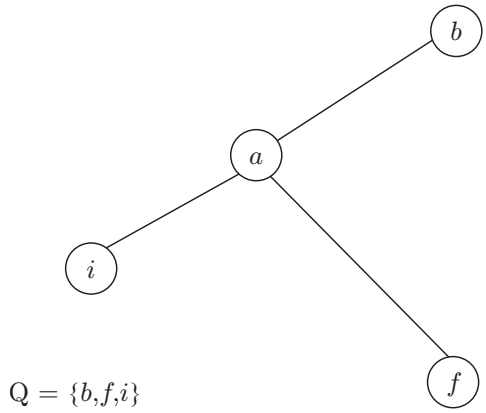
Example 3.9





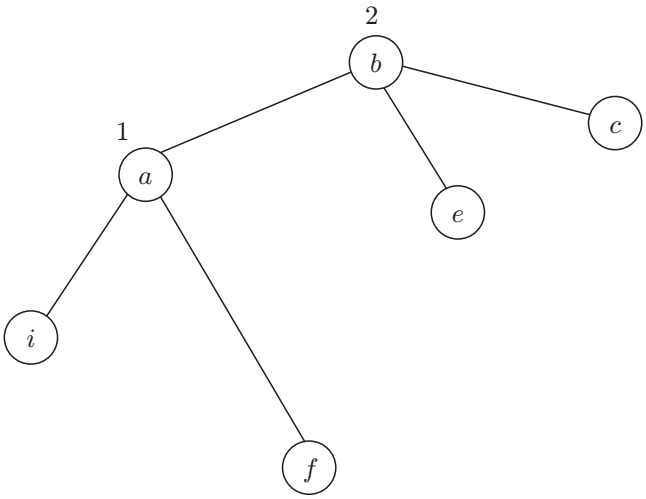
Q = $\{a\}$

(i)

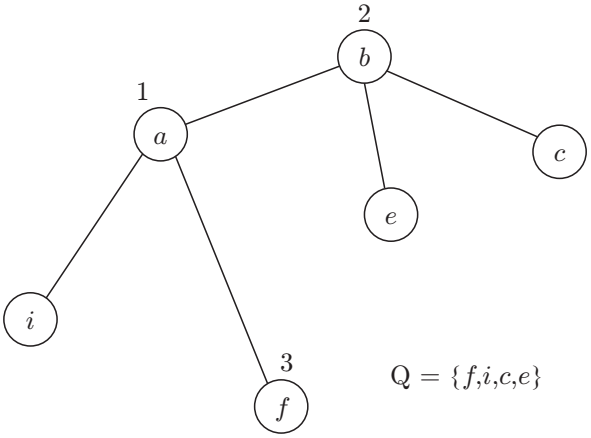


Q = $\{b,f,i\}$

(ii)

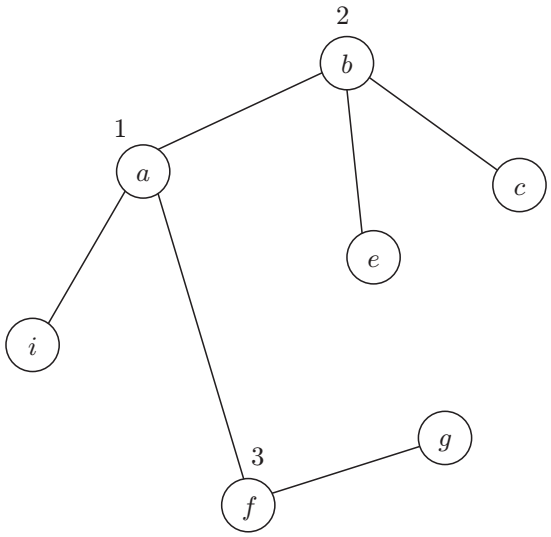


(iii)

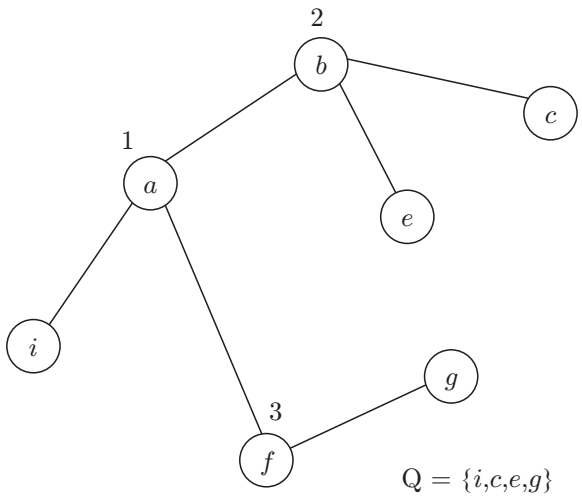


Q = $\{f,i,c,e\}$

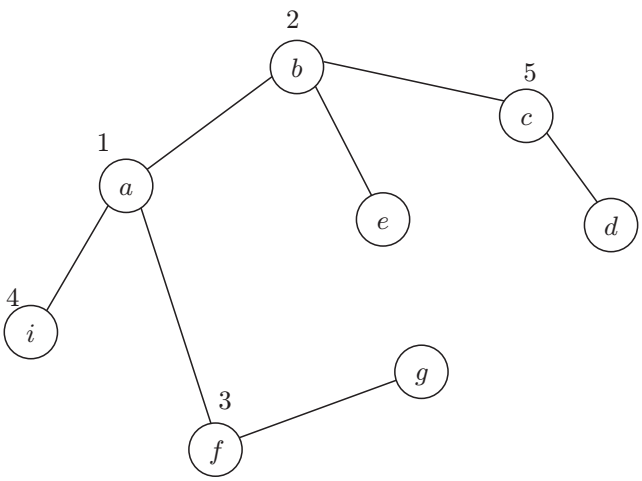
(iv)



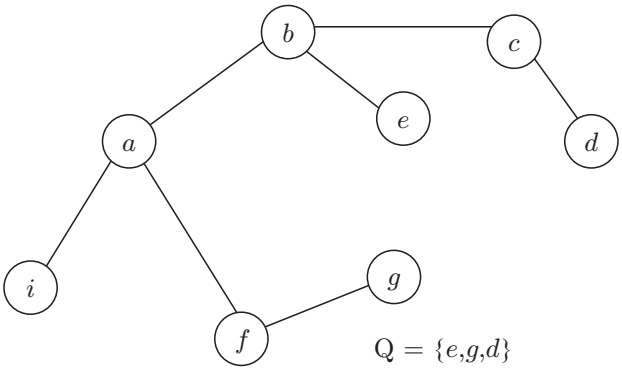
(v)



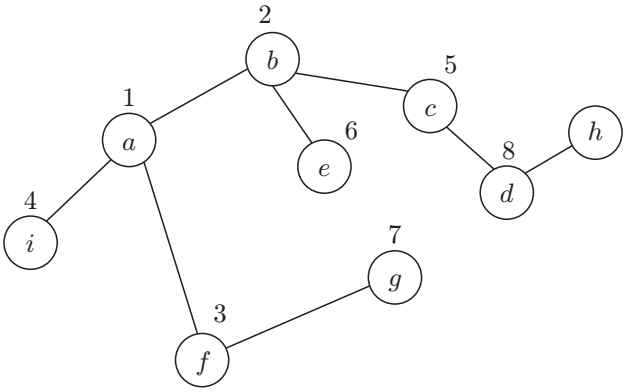
(vi)



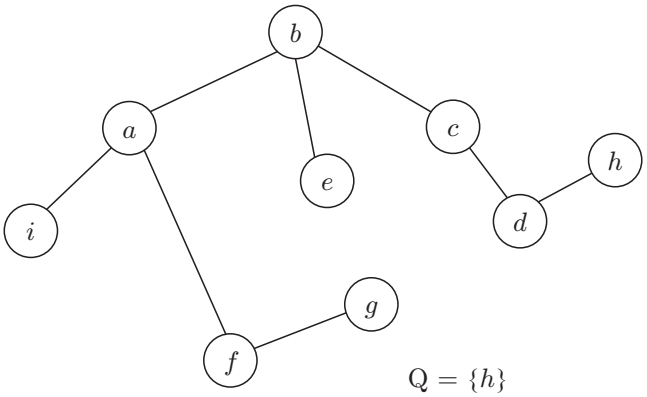
(vii)



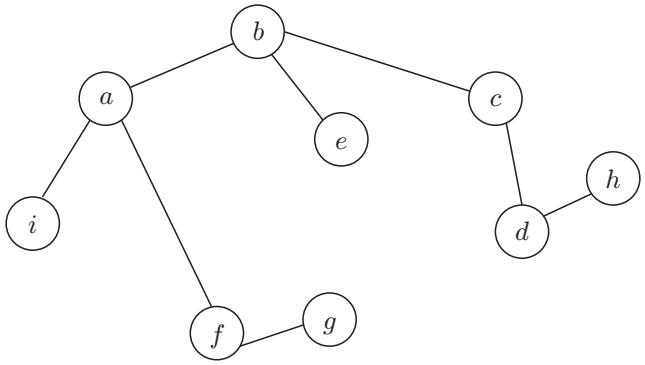
(viii)



(ix)



(x)

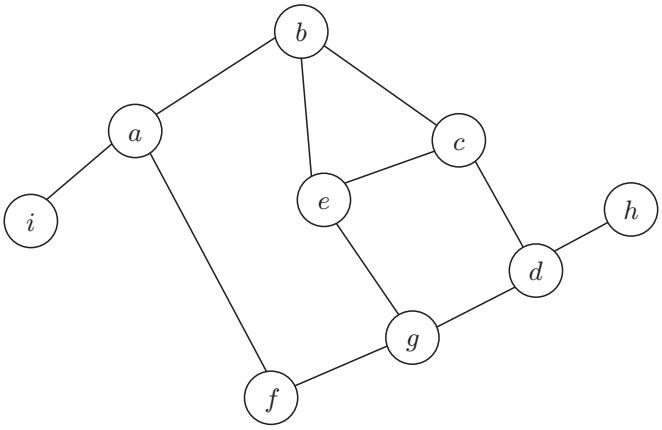


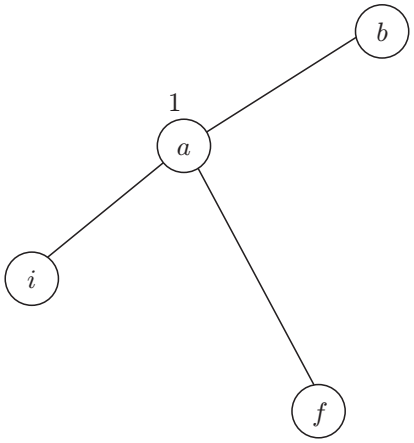
{a,b,f,i,c,e,g,d,h}

(xi)

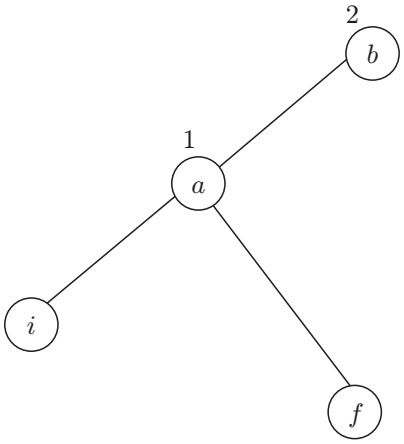
2. Depth first traversal: It is a preorder traversal of an ordered tree. It first traverses one subtree before returning to the current node and then traverses another subtree. It uses stack.

Example 3.10

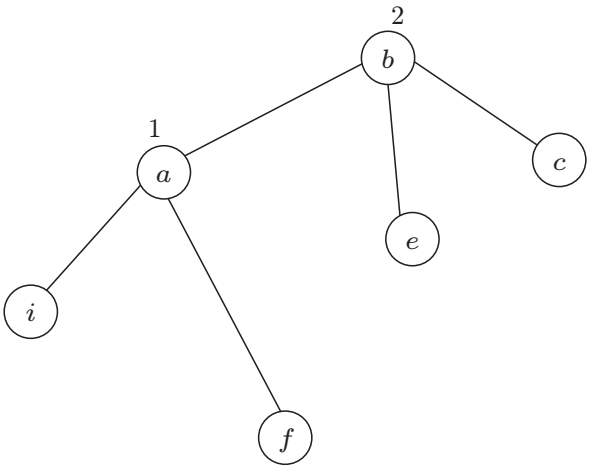




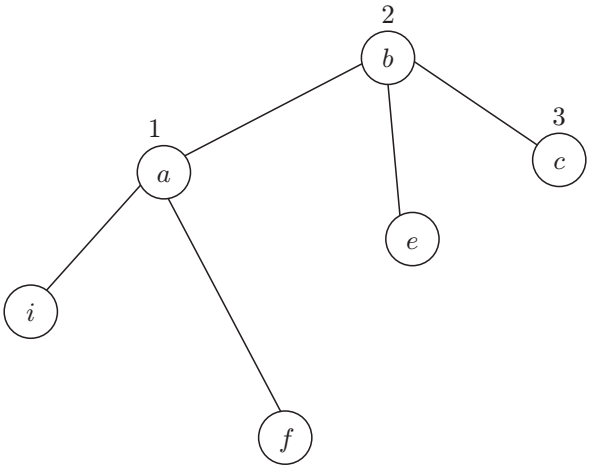
(i)



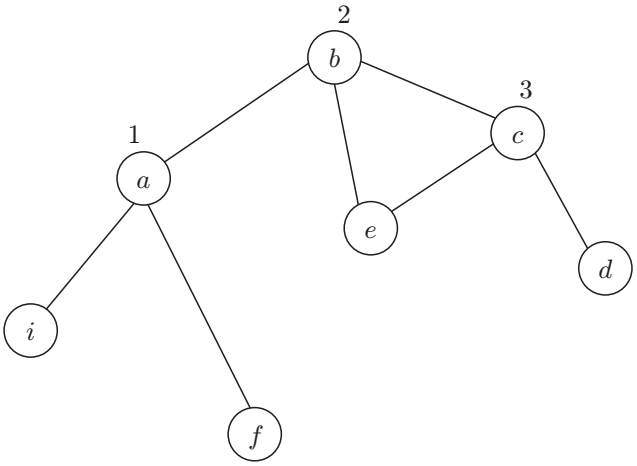
(ii)



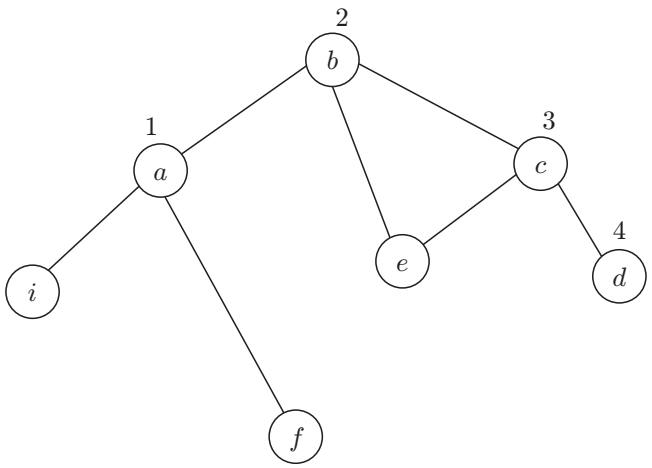
(iii)



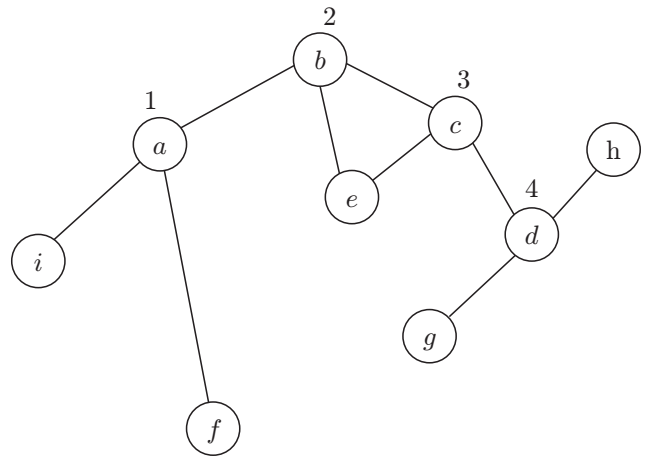
(iv)



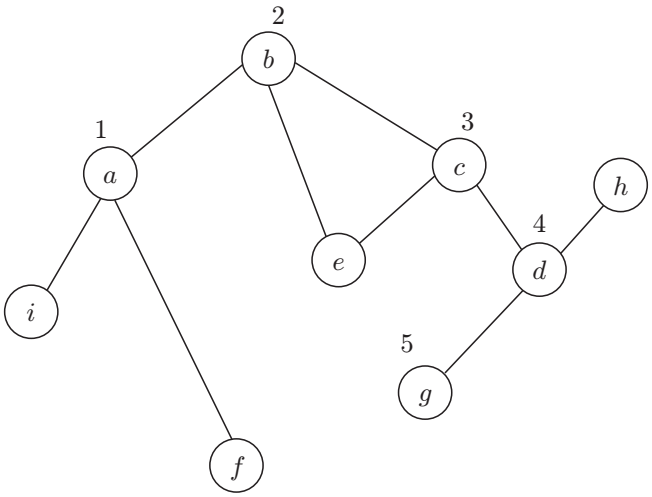
(v)



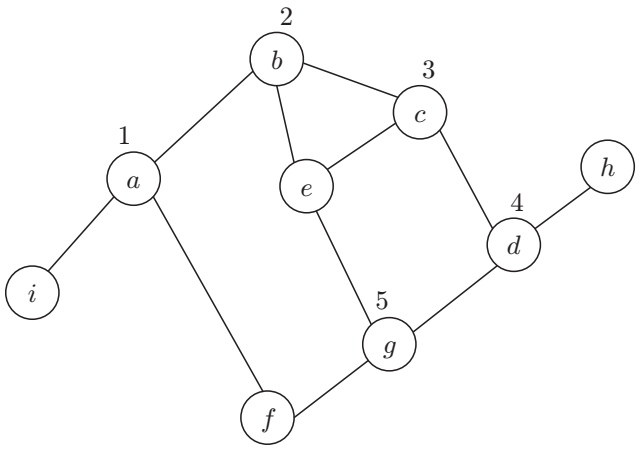
(vi)



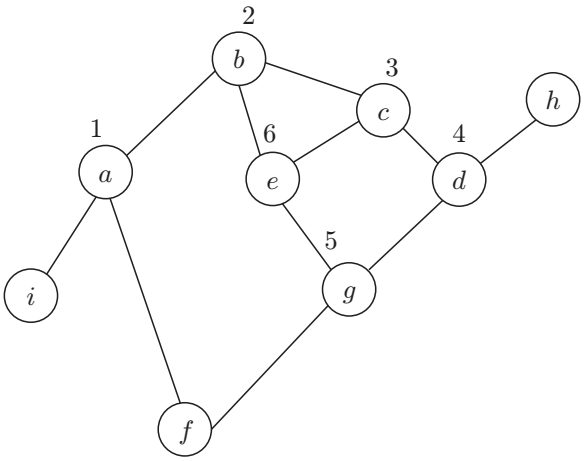
(vii)



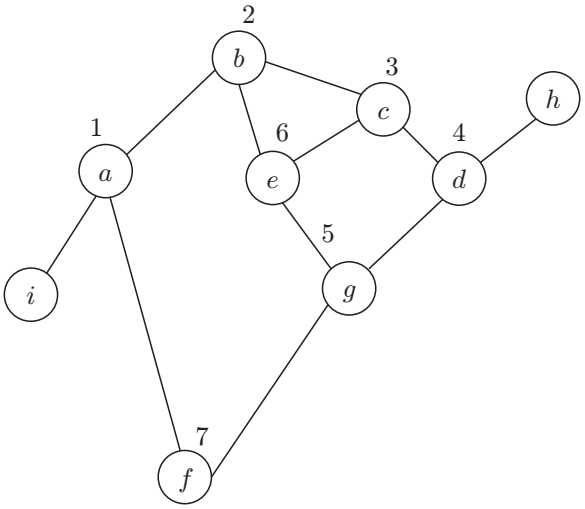
(viii)



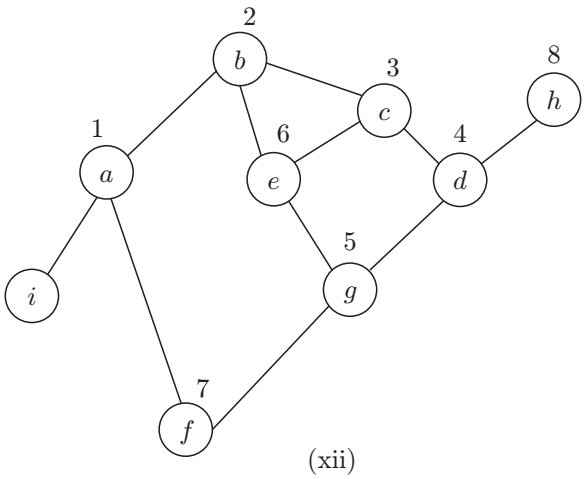
(ix)



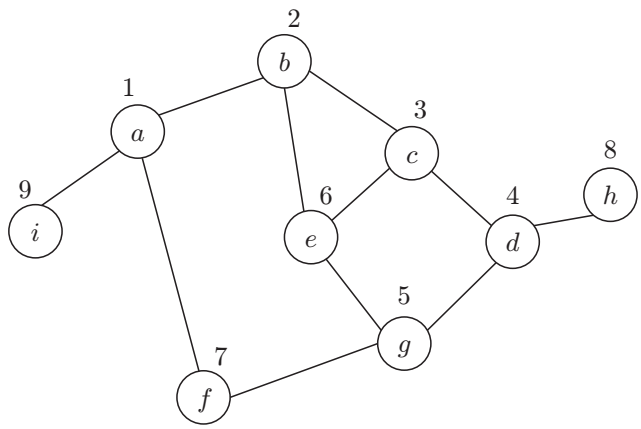
(x)



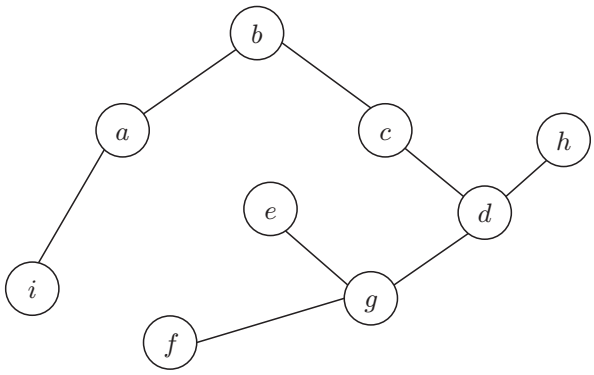
(xi)



(xii)



(xiii)



$\{a,b,c,d,g,e,f,h,i\}$

(ix)

3.6.6.3 Shortest Path

For a given directed graph, where each edge has a non-negative weight or cost, the shortest path problem is to find a path of least total weight from the source to every other vertex in the graph (Fig. 3.20).

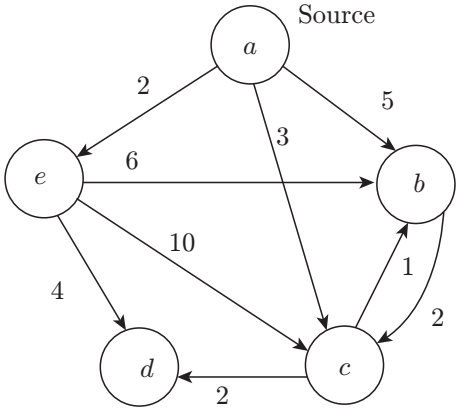
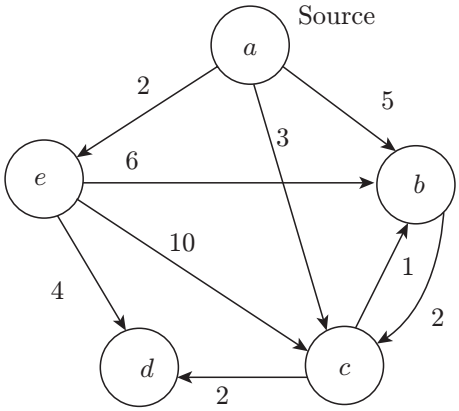


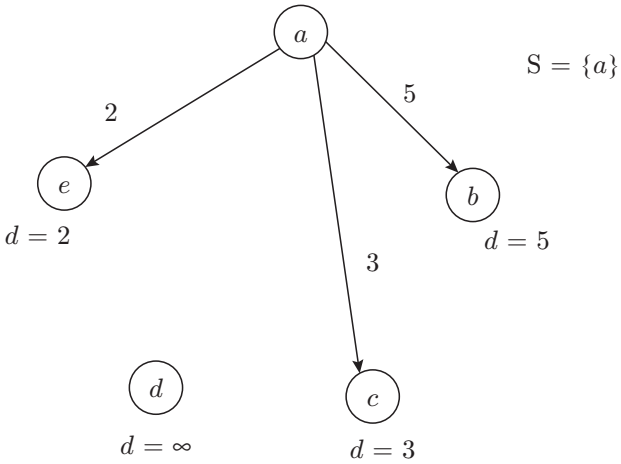
Figure 3.20 | Graph for finding the shortest path.

To find the shortest path, an approach known as Greedy method is used. The method is as follows:

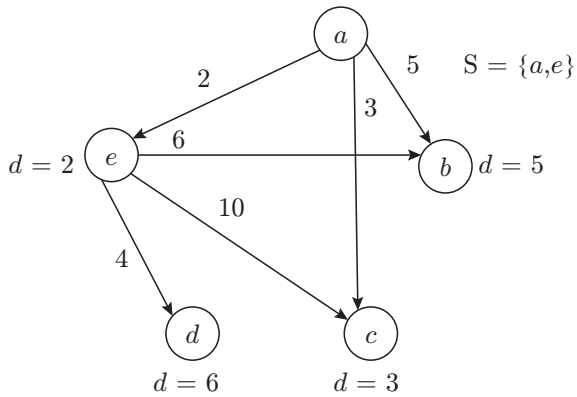
Choose the vertex with smallest weight and mark the path from source to the chosen vertex. Next, mark the same for other vertices.

Example 3.11

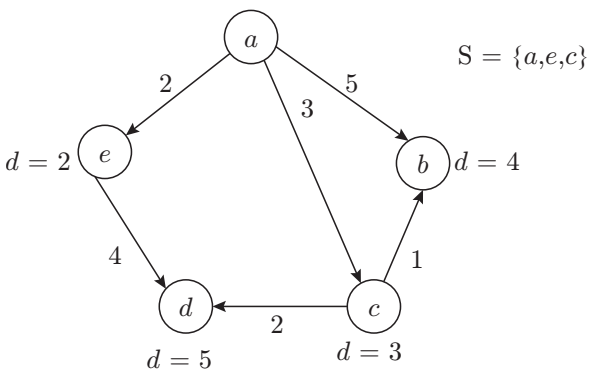




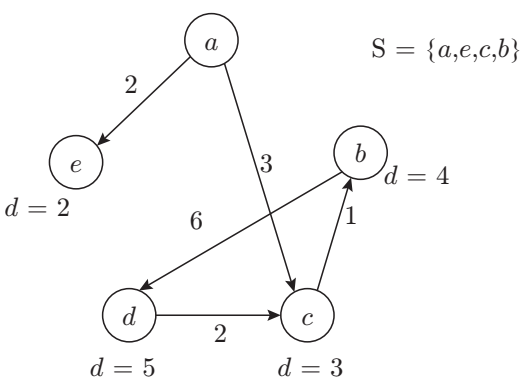
(i)



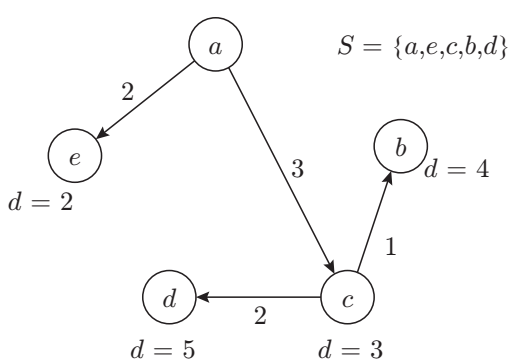
(ii)



(iii)



(iv)



(v)

3.6.6.4 Spanning Tree

A spanning tree of a graph G is a tree which contains all the vertices of the graph G. See Fig. 3.21.

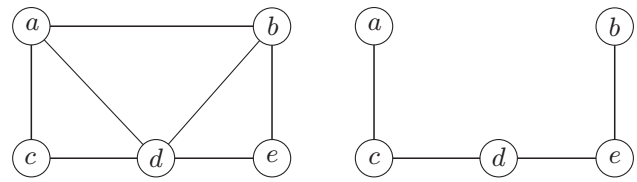


Figure 3.21 | Spanning tree of a graph.

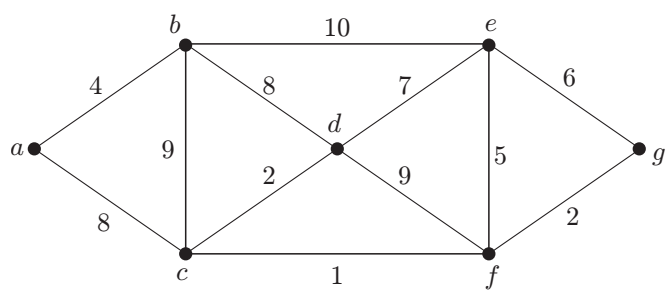
Minimum Spanning Tree

Minimum spanning tree in an undirected connected weighted graph is a spanning tree with minimum weight.

Prim's algorithm:

1. Begin with all the vertices.
2. Choose and draw any vertex.
3. Find the edge of least weight joining a drawn vertex to a vertex not currently drawn. Draw the weighted edge and the corresponding new vertex.
4. Repeat step 3 until all the vertices are connected, then STOP.

Example 3.12



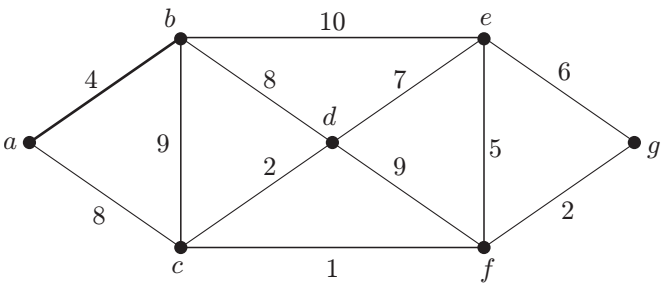
(i)

Step 1

$U = \{a\},\ V - U = \{b,\ c,\ d,\ e,\ f,\ g\}$

Closest		
U	**V – U**	**Low Cost**
b	A	4
c	A	8
d	A	∞
e	A	∞
f	A	∞
g	A	∞

Select vertex b to include in U.



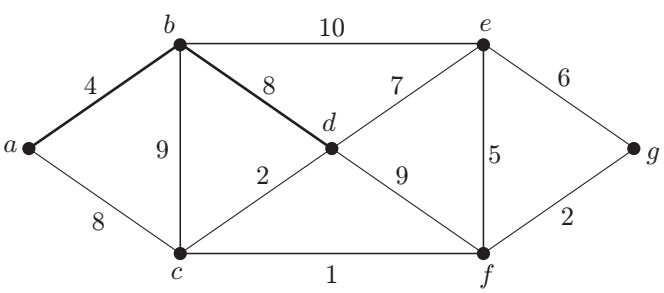
(ii)

Step 2

$U = \{a, b\}, \ V - U = \{c, d, e, f, g\}$

Closest		
U	**V – U**	**Low Cost**
C	A	8
D	B	8
E	B	10
F	A	∞
G	A	∞

Select vertex d to include in U.



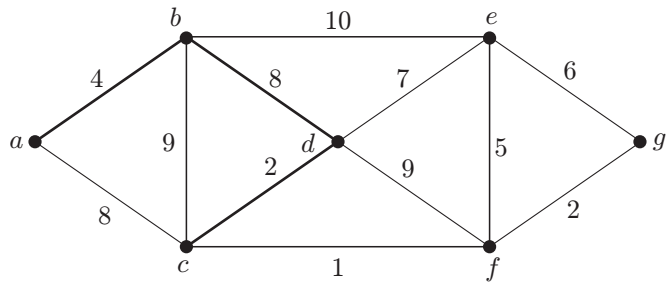
(iii)

Step 3

$U = \{a, b, d\}, \ V - U = \{c, e, f, g\}$

Closest		
U	**V – U**	**Low Cost**
c	D	2
e	D	7
f	D	9
g	A	∞

Select vertex c to include in U.



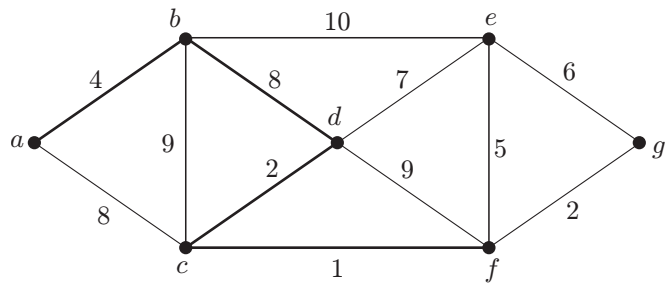
(iv)

Step 4

$U = \{a, b, c, d\}, \ V - U = \{e, f, g\}$

Closest		
U	**V – U**	**Low Cost**
e	D	7
f	C	1
g	A	∞

Select vertex f to include in U.



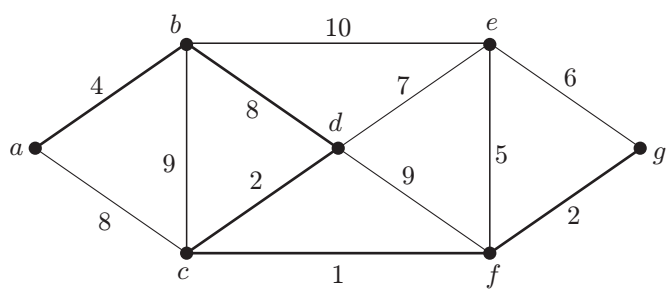
(v)

Step 5

$U = \{a, b, c, d, f\}, \ V - U = \{e, g\}$

Closest		
U	**V – U**	**Low Cost**
e	F	5
g	F	2

Select vertex g to include in U.



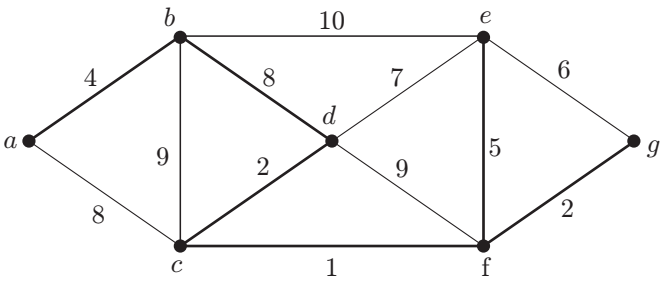
(vi)

Step 6

$U = \{a, b, c, d, f, g\}$, $V - U = \{e\}$

Closest		
U	**V – U**	**Low Cost**
e	*F*	5

Select vertex e to include in U.



(vii)

Step 7

$U = \{a, b, c, d, e, f, g\}$, $V - U = \{\}$

MST Complete

3.6.6.5 AVL Tree

AVL tree is a binary tree which satisfies the height balance property (Fig. 3.22). It has a time complexity of $O(\log(n))$.

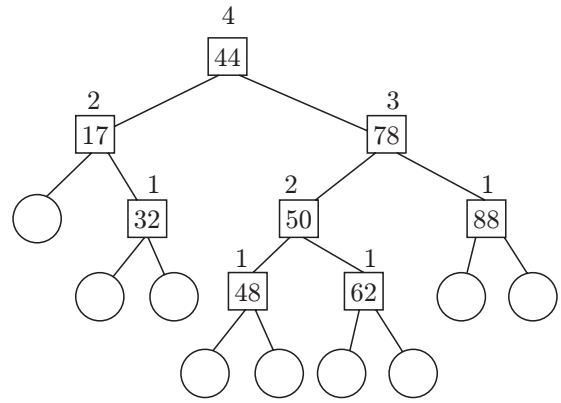


Figure 3.22 | AVL tree.

AVL has the following properties:

1. **Balance Factor of a Node:** It is the difference in heights of its two subtrees ($h_R - h_L$).
2. **Balanced Node:** Node with $|BF| \leq 1$; if $|BF| > 1$, the node is unbalanced.
3. **Balance Factor of Binary Trees:** It corresponds to the balance factor of its root node.
   - Tree is 'left-heavy' if BF ≤ -1
   - Tree is 'equal-height' if BF $= 0$
   - Tree is 'right-heavy' if BF $\geq +1$
4. **Balance Factor of AVL Trees:** BF of each node in an AVL tree can only $\in \{-1, 0, 1\}$.

Rotation

It is the restructuring of the tree which maintains the binary search tree property.

1. **Types of Rotation:**
   - *Single-Left rotation*: Refer Fig. 3.23.

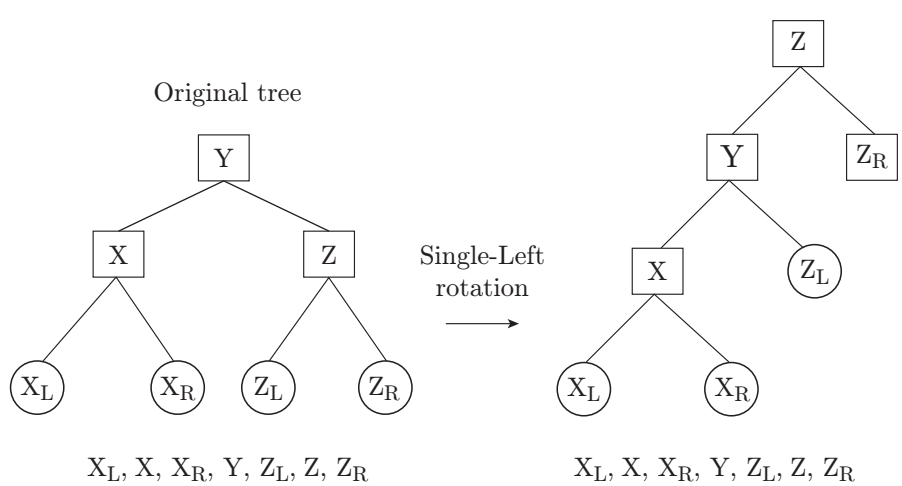


Figure 3.23 | Single-Left rotation.

- *Single-Right rotation*: Refer Fig. 3.24.

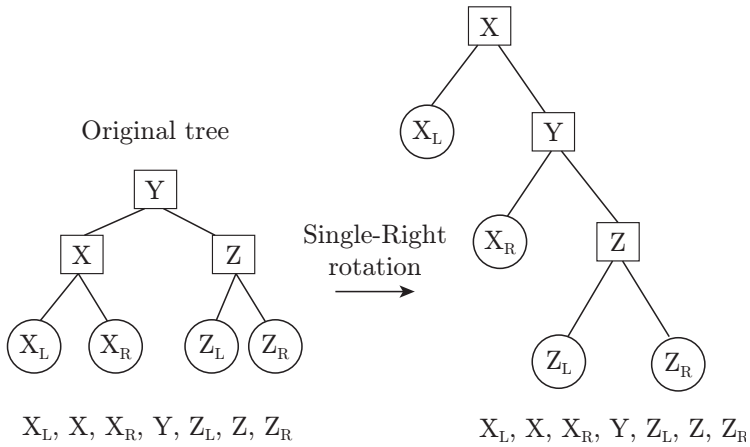


Figure 3.24 | Single-Right rotation.

- *Double left-right rotation*: Refer Fig. 3.25.

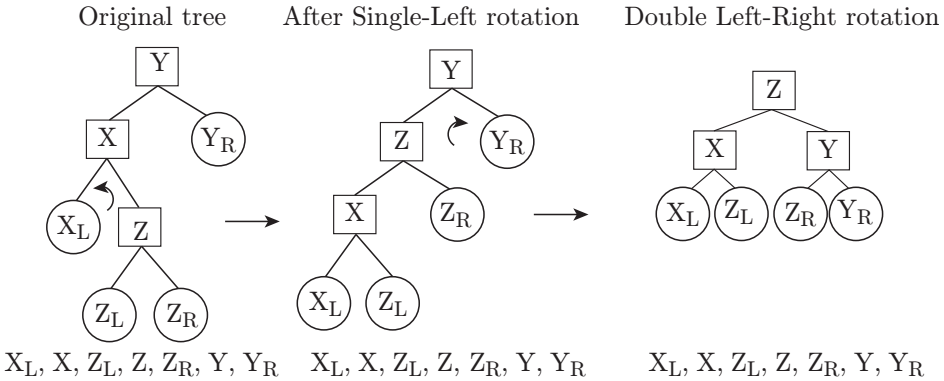


Figure 3.25 | Double left-right rotation.

- *Double right-left rotation*: Refer Fig. 3.26.

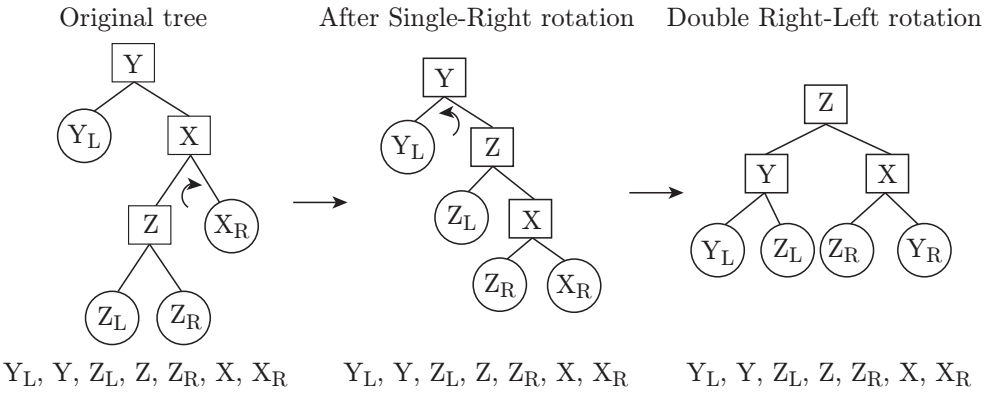


Figure 3.26 | Double right-left rotation.

3.6.6.6 Binary Heap

A complete binary tree is said to be binary heap if all the elements below in hierarchy are either greater than or equal to the root element. This binary heap (see Fig. 3.27) is known as min-heap. In addition, if all the elements are lesser or equal to the root element, this is called max-heap.

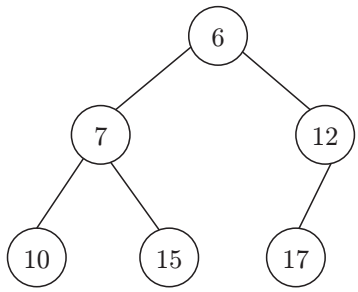


Figure 3.27 | Binary heap.

1. The height of a binary heap is $O(\log n)$. The runtime of deleteMin, insert and remove is $O(\log n)$.
2. The root node is the highest priority element. To remove the highest/lowest priority object (root), heap data structure is preferred. Priority queue is the best example to be implemented.
3. The running time of heap sort is $O(n\log n)$.

IMPORTANT FORMULAS

1. Insertion/Deletion in array: $O(n)$
2. Split/Merge in array: $O(n)$
3. Time complexity of Push(): $O(1)$
4. Time complexity of Pop(): $O(1)$
5. Time complexity of Enqueue(): $O(1)$
6. Time complexity of Dequeue(): $O(1)$
7. Number of elements in queue =

$$\begin{cases} \text{rear} - \text{front} + 1, & \text{if rear} = \text{front} \\ \text{rear} - \text{front} + n, & \text{otherwise} \end{cases}$$

8. Number of nodes in full binary tree: $2^{h+1}-1$ [h: levels]
9. Number of leaf nodes in full binary tree: 2^h
10. Number of waste pointers in complete binary tree of n nodes: $n + 1$
11. Complexity of inorder, preorder and postorder tree traversal: $O(n)$
12. Minimum number of moves for tower of Hanoi: $2^n - 1$
13. Number of unique binary trees: $\dfrac{{}^{2n}_{\ n}C}{n+1}$
14. **One-dimensional arrays**

    In one dimension, an array 'A' is declared as: A[lb … … ub]

    where lb is the lower bound of array and ub is the upper bound of array.

    Suppose we want to calculate the ith element address, then

    $$\text{Address (arr}[i]) = \text{BA} + (i - \text{lb}) * c$$

    where BA is the base address of array and lb is the lower bound of array and c is the size of each element
15. **Two-dimensional arrays**

    In two dimensions, an array 'A' is declared as:

    $$A[\text{lb}_1 \ldots \ldots \text{ub}_1][\text{lb}_2 \ldots \ldots \text{ub}_2]$$

    where lb_1 is the lower bound for row, lb_2 is the lower bound for column, ub_1 is the upper bound for row and ub_2 is the upper bound for column.

    Row major order

    $$\text{Address (arr}[i][j]) = \text{BA} + [i - \text{lb}_1) * \text{Nc} + (j - \text{lb}_2)] * c$$

    Column major order

    $$\text{Address (arr}[i][j]) = \text{BA} + [j - \text{lb}_2) * \text{Nr} + (i - \text{lb}_1)] * c$$

    where BA is the base address, Nr is the number of rows = $(\text{lb}_2 - \text{lb}_1 + 1)$, and Nc is the number of columns = $(\text{ub}_2 - \text{ub}_1 + 1)$.

SOLVED EXAMPLES

1. Variables of function call are allocated in

(a) registers and stack. (b) cache and heap.
(c) stack and heap. (d) registers and heap.

Solution: The variables of function call can be stored in stack and heap.

Ans. (c)

2. Predict the output of:

```
void main()
{
     float x=1.1;
     double y= 1.1;
if(x==y)
printf("I see You");
else
printf("I hate You");
     }
```

(a) I see You
(b) I hate You
(c) Cannot compare double and float
(d) Run time error

Solution: The stored value may or may not be exact, the precision of the value depends upon the number of bytes. Double stores value 1.1 with more precision than float because double takes 8 bytes, whereas float takes 4 bytes.

Ans. (b)

3. Global variable conflicts due to multiple file occurrence is resolved during

(a) load time.
(b) execution time.
(c) link time.
(d) parsing phase of compiler.

Solution: A global variable provides access from multiple files. Due to merging of one file code to another code, conflict occurs in the variable names and this cause an error at link time.

Ans. (c)

4.
```
#define f(a,b) a+b
#define g(a,b)a*b
main()
{
     int m;
     m=2*f(3,g(4,5));
     printf("\n m is %d", m);
}
```

What is the value of m?

(a) 70 (b) 50
(c) 46 (d) 69

Solution: `m=2*f(3,g(4,5));`
The function g(4,5) will return 20 by performing multiplication operations. Then function f (3,20) will return 23 by performing addition operation. In the last output of function f (3,20) is multiplied with 2. So, the final answer will be 46.

Ans. (c)

5. A program to find the factorial of a given number is written (a) using recursion and (b) without using recursion. Which of these will result in stack overflow for a given number?

(a) Only first program with recursion
(b) Both may result for the same number
(c) Only second program
(d) None of these

Solution: When the factorial program is written using recursion only then will it use stack, so stack overflow will occur in recursive program only.

Ans. (a)

6. If two strings are identical then strcmp() returns:

(a) True (b) 1
(c) −1 (d) 0

Solution: strcmp() is a pre-defined function of C library which returns value by calculating value=(second string length − first string length). Hence, if both the strings are identical then it will return 0.

Ans. (d)

7. In tree construction, which one will be suitable and efficient data structure?

(a) Queue (b) Linked list
(c) Heap (d) String

Solution: Linked list is the best suitable and efficient data structure for constructing a tree because an item can be inserted and deleted from linked list with less cost.

Ans. (b)

8. Which of the following is not true about spanning tree?

(a) It is a tree derived from a graph.
(b) All the nodes of a network appear on the tree only once.

(c) Spanning tree cannot have at most two edges repeated.
(d) Spanning tree cannot be minimum and maximum.

Solution: A spanning tree is a subgraph of a given graph G, which covers all the vertices of G and should be a tree. Spanning tree could be minimum or maximum.

Ans. (d)

9. Which of the following has compilation error in C?

(a) int $n = 17$;
(b) char ch = 99;
(c) float f = (float) 99.32;
(d) #include<stdio.h>

Solution: The syntax for inclusion of header file is `#include <stdio.h>`. There should be space between #include and <stdio.h>.

Ans. (d)

10. _____ is often used to prove the correctness of a recursive function.

(a) Associativity
(b) Commutativity
(c) Mathematical induction
(d) Factorial

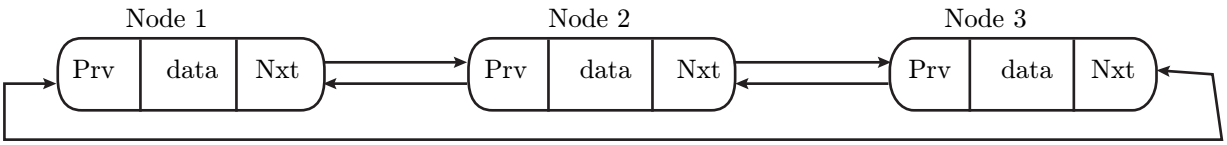
Solution: Mathematical induction is the method through which correctness of recursive function can be proved.

Ans. (c)

11. In a doubly linked list, the number of pointers affected for an insertion operation will be:

(a) 4
(b) 0
(c) 1
(d) Depends upon the nodes of the doubly linked list.

Solution: Let a node X be inserted after node 1, then both pointer of node X(Prv and Nxt) will be affected and node 1's Nxt pointer will point to node X and node 2's Prv pointer will also point to node X. So, total four pointers will be affected.

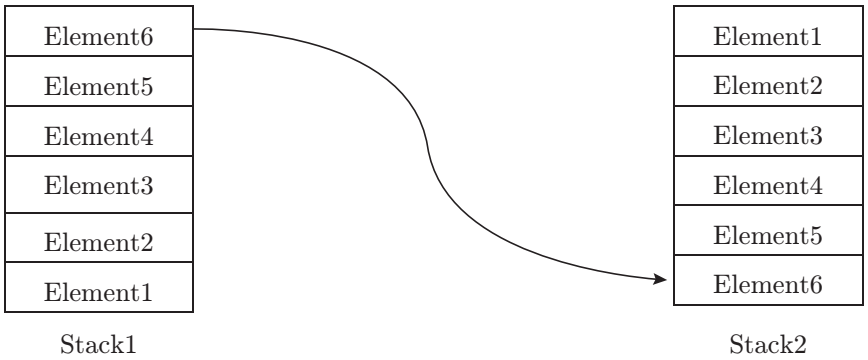


Ans. (a)

12. The queue data structure is to realised by using stack. The number of stacks needed would be:

(a) It cannot be implemented.
(b) 2 stacks.
(c) 4 stacks.
(d) 1 stacks.

Solution: Let there are 6 elements (1, 2, 3, 4, 5 and 6). First we will push them in stack 1 and then pop from stack 1 and push in stack2. The same behaviour will be observed when we will pop from stack2



So, only 2 stacks are required. So, option (b) is correct.

Ans. (b)

13. Postfix expression for the infix expression

$$(A*B - (C*D))/(F/D + E)$$

(a) $AB*CD* - FD/E+/$ (b) $AB*CD - *FDE//+$
(c) $AB*CD - FD*E//+$ (d) $AB*CD - *FDE/+/$

Solution:

Infix Expression	Postfix Expression	Stack
$(A*B - (C*D))/(F/D + E)$		
$*B - (C*D))/(F/D + E)$	A	(
$B - (C*D))/(F/D + E)$	A	(*
$- (C*D))/(F/D + E)$	AB	(*
$(C*D))/(F/D + E)$	AB*	(–
$*D))/(F/D + E)$	AB*C	(–(
$D))/(F/D + E)$	AB*C	(–(*
$/(F/D + E)$	AB*CD	(–(*))
$/(F/D + E)$	AB*CD*–	
$(F/D + E)$	AB*CD*–	/
$/D + E)$	AB*CD*–F	/
$D + E)$	AB*CD*–F	/(/
$+ E)$	AB*CD*–FD	/(/
$E)$	AB*CD*–FD/	/(+
	AB*CD*–FD/E	/(+)
	AB*CD*–FD/E+/	

Ans. (a)

14. There are six nodes. The different types of trees that can be realised are _____.

(a) 10 (b) 64
(c) 58 (d) 65

Solution: 58 trees can be realized.

Ans. (c)

15. In which of the following case(s) is it possible to obtain different results for call-by-reference and call-by-name parameter passing

(a) Passing an expression as a parameter
(b) Passing an array as a parameter
(c) Passing a pointer as a parameter
(d) Passing an array elements as a parameter

Solution: Passing an element of an array by call by name behaves similar to call by value. Therefore, if array elements are passed as parameter then they can produce different results for call-by-reference and call-by-name parameter passing.

Ans. (d)

16. A variant record in Pascal is defined by

```
typevarirec = record
     number: integer;
     case (var1, var2) of
          var1; (x, y : integer)
           var2: (p,q: real)
     end
end
```

Suppose an array of 100 records was declared on a machine which uses 6 bytes for an integer and 10 bytes for a real. How much space would the compiler have to reserve for the array

(a) 3800 (b) 3200
(c) 2800 (d) 4000

Solution: There are 100 records

So 600 B for 'record no': [100 × 6]

1200 B for var(x, y): [so, (6 + 6)*100]

2000 B for var(p, q) : [so, (10 + 10)*100]

Ans. (a)

17. What is the result of the following program?

```
Program side-effect (input, output)
var x, result: integer;
Function f (var x: integer): integer;
begin
     x: x+1;
     f:=x;
end
begin
     x:=5;
     result:=f(x)*f(x);
     writeln (result)
end
```

(a) 5 (b) 25
(c) 36 (d) 42

Solution: Function 'f' uses 'x' as local variable. So value of 'x' does not change globally and both the time $x = 5$ is passed to the function 'f'.

Ans. (c)

18. Consider the following C declaration

```
struct {
short s [5]
union{
float y;
long z;
}u;
}t;
```

Assume that objects of the type short, float and long occupy 2 bytes, 4 bytes and 8 bytes, respectively. The memory requirement for variable t, ignoring alignment considerations, is

(a) 22 bytes (b) 18 bytes
(c) 14 bytes (d) 10 bytes

Solution: The structure is created with total size of 'short and union' short have size of 2 byte, as it is an array of five elements so it will take 10 bytes. Union will consider the maximum of {float y (4 bytes) and long z(8 bytes)} 12 bytes. So, total size will be 22 bytes (10 + 12).

Ans. (a)

19. In a compact single-dimensional array representation for lower triangular matrices (i.e. all the elements above the diagonal are zero) of size $n \times n$, non-zero elements (i.e. elements of the lower triangle) of each row are stored one after another, starting from the first row, the index of the (i,j)th element of the lower triangular matrix in this new representation is

(a) $i + j$ (b) $i + j - 1$

(c) $(j-1)+\dfrac{i-1}{2}$ (d) $i+\dfrac{j(j-1)}{2}$

Solution: To find location in lower triangular matrix formula is given as

$$LOC(i,j) = (j-1) + \frac{i(i-1)}{2}$$

Ans. (c)

20. The following sequence of operation is performed on a stack:

PUSH(10), PUSH(20), POP, PUSH(10), PUSH(20), POP, POP, POP, PUSH(20), POP

The sequence of values popped out is

(a) 20, 10, 20, 10, 20 (b) 20, 20, 10, 10, 20
(c) 10, 20, 20, 10, 20 (d) 20, 20, 10, 20, 10

Solution:

Operation	Stack	Popped Elements List
Push	10	
Push	10, 20	
Pop	10	20
Push	10, 10	20
Push	10, 10, 20	20
Pop	10, 10	20, 20
Pop	10	20, 20, 10
Pop		20, 20, 10, 10
Push	20	20, 20, 10, 10
Pop		20, 20, 10, 10, 20

So, option (b) is correct.

Ans. (b)

21. The postfix expression for the infix expression

$$A + B * (C + D)/F + D * E \text{ is}$$

(a) $AB + CD +* F/D + E*$
(b) $ABCD + *F/DE *++$
(c) $A * B + CD/F * DE++$
(d) $A + * BCD/F * DE++$

Solution: Given expression $A + B * (C + D)/F + D * E$

Postfix conversion steps

$$A + B * CD + F/+ DE*$$
$$A + BCD +*F/DE*+$$
$$ABCD +*F/DE*++$$

So, option (b) is correct.

Ans. (b)

GATE PREVIOUS YEARS' QUESTIONS

1. Consider the following C function.

```
float f(float x, int y) {
float p, s; int i;
for (s=1,p=1,i=1; i<y; i++) {
p *= x/i;
s+=p;
}
return s;
}
```

For large values of y, the return value of the function f best approximates

(a) x^y (b) e^x (c) $\ln(1 + x)$ (d) x^x

(GATE 2003: 1 Mark)

Solution:

Applying iterative method:

i = 1	p = x/1	s = 1 + x
i =2	p = x * x/2	s = 1 + x + x^2/2
i = 3	p = x^3/ 6	s = 1 + x + x^2/2 + x^3/ 6
from above iterations: s = 1 + x + x^2/2 + x^3/ 6 + ⋯⋯ s = 1 + x + x^2/2! + x^3/ 3! + x^4/ 4! + ⋯⋯ = e^x		

Ans. (b)

2. Assume the following C variable declaration

```
int * A[10], B[10][10];
```

Of the following expressions

I. $A[2]$ II. $A[2][3]$
III. $B[1]$ IV. $B[2][3]$

which will not give compile-time errors if used as left-hand sides of assignment statements in a C program?

(a) I, II and IV only (b) II, III and IV only
(c) II and IV only (d) IV only

(GATE 2003: 1 Mark)

Solution: B[1] is address of second one dimensional array. Address can not be assigned to simple variable. So it will produce compile time error.

Ans. (a)

3. Let $T(n)$ be the number of different binary search trees on n distinct elements. Then $T(n) = \sum_{k=1}^{n} T(k-1)T(x)$, where x is

(a) $n - k + 1$ (b) $n - k$
(c) $n - k - 1$ (d) $n - k - 2$

(GATE 2003: 1 Mark)

Solution:

$$T(n) = \sum_{k=0}^{n} T(k-1)T(x)$$
$$T(n) = T(x)\ [T(0) + \ldots + T(n-1)]$$
$$x = n - k + 1$$

Ans. (a)

4. Suppose the numbers 7, 5, 1, 8, 3, 6, 0, 9, 4, 2 are inserted in that order into an initially empty binary search tree. The binary search tree uses the usual ordering on natural numbers. What is the inorder traversal sequence of the resultant tree?

(a) 7 5 1 0 3 2 4 6 8 9 (b) 0 2 4 3 1 6 5 9 8 7
(c) 0 1 2 3 4 5 6 7 8 9 (d) 9 8 6 4 2 3 0 1 5 7

(GATE 2003: 1 Mark)

Solution: In-order traversal of a binary search tree arrange elements in increasing order.

Ans. (c)

5. A data structure is required for storing a set of integers such that each of the following operations can be done in $O(\log n)$ time, where n is the number of elements in the set.

I. Deletion of the smallest element
II. Insertion of an element if it is not already present in the set

Which of the following data structures can be used for this purpose?

(a) A heap can be used but not a balanced binary search tree.
(b) A balanced binary search tree can be used but not a heap.
(c) Both balanced binary search tree and heap can be used.
(d) Neither balanced binary search tree nor heap can be used.

(GATE 2003: 2 Marks)

Solution: Time taken by heap to insert and delete n elements is $O(n)$. Only balanced binary search tree takes $O(\log n)$ time for insertion and deletion.

Ans. (b)

6. Let S be a stack of size $n \geq 1$. Starting with the empty stack, suppose we push the first n natural numbers in sequence, and then perform n pop operations. Assume that Push and Pop operations take X seconds each, and Y seconds elapse between the end of one such stack operation and the start of the next operation. For $m = 1$, define the stack-life of m as the time elapsed from the end of Push(m) to the start of the pop operation that removes m from S. The average stack-life of an element of this stack is

(a) $n(X + Y)$.
(b) $3Y + 2X$.
(c) $n(X + Y) - X$.
(d) $Y + 2X$

(GATE 2003: 2 Marks)

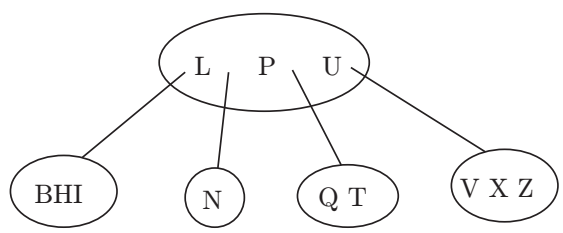
Solution : Time taken by Pop and Push opérations : $X + X = 2X$

Time delay between two stack operations $= Y$

Total time taken $= 2X + Y$

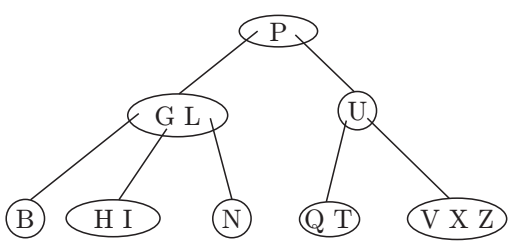
Ans. (d)

7. Consider the following 2−3−4 tree (i.e. B-tree with a minimum degree of two) in which each data item is a letter. The usual alphabetical ordering of letters is used in constructing the tree.

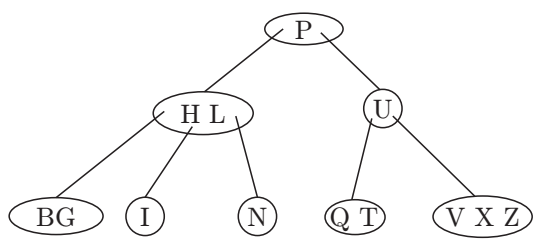


What is the result of inserting G in the above tree?

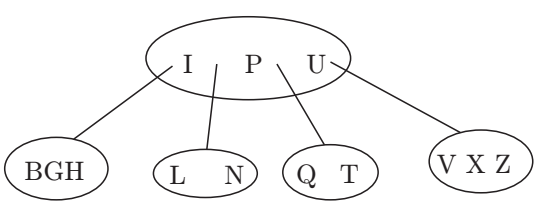
(a)



(b)



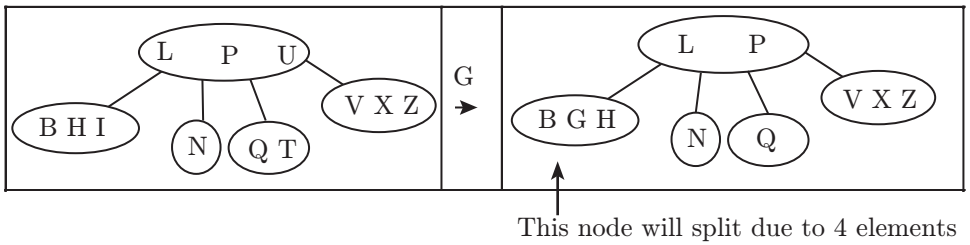
(c)



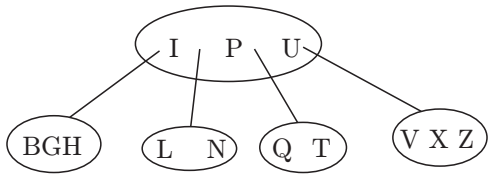
(d) None of the above

(GATE 2003: 2 Marks)

Solution: 2-3-4 B tree has minimum degree 2 and maximum 4. According to max degree, atmost 3 elements can be there in node.



Result will be



Ans. (c)

Common Data Questions 8 and 9: The following program fragment is written in a programming language that allows global variables and does not allow nested declarations of functions.

```
global int i = 100, j = 5;
void P(x) {
int i = 10;
print(x + 10);
i = 200;
j = 20;
print (x);
}
main() {P(i + j);}
```

8. If the programming language uses static scoping and call by need parameter passing mechanism, the values printed by the above program are

(a) 115, 220. (b) 25, 220.
(c) 25, 15. (d) 115, 105.

(GATE 2003: 2 Marks)

Solution: In static scoping, the variables and values are resolved at compile time.

P(100 + 5) = P(105)

Print x + 10 will print 115

Print x will print 105

Ans. (d)

9. If the programming language uses dynamic scoping and call-by-name parameter-passing mechanism, the values printed by the above program are

(a) 115, 220. (b) 25, 220.
(c) 25, 15. (d) 115, 105.

(GATE 2003: 2 Marks)

Solution: In dynamic scoping, the variables and values are resolved at run time.
P(100 + 5) = P(105)
First Print x + 10 will print 115
Second Print x will update x = 200 + 20 = 220 and will print 220

Ans. (a)

10. In the following C program fragment, j, k, n and TwoLog_n are integer variables, and A is an array of integers.
The variable n is initialised to an integer ≥ 3, and TwoLog_n is initialised to the value of n $2^*\lceil \log_2(n) \rceil$

```
for (k=3; k <= n; k++)
                A[k] = 0;
    for (k=2; k <= TwoLog_n; k++)
                for (j=k+1; j <= n; j++)
                  A[j] = A[j] || (j%k);
    for (j=3; j <= n; j++)
            if (!A[j]) printf("% d", j);
```

The set of numbers printed by this program fragment is

(a) $\{m | m \leq n, (\exists i)\ [m = i!]\}$
(b) $\{m | m \leq n, (\exists i)\ [m = i^2]\}$
(c) $\{m | m \leq n,\ m \text{ is prime}\}$
(d) $\{\}$.

(GATE 2003: 2 Marks)

Solution: The condition ! A[j] is equivalent to A[j] = = 0. But in array A no element is zero. So it will print nothing.

Ans. (d)

11. Consider the C program shown below.

```
#include <stdio.h>
#define print(x) printf("% d —, x)
int x;
void Q(int z) {
            z += x; print(z);
}
void P(int *y) {
            int x = *y+2;
            Q(x); *y = x-1;
            print(x);
}
main(void) {
            x = 5;
            P(&x)
            print(x);
 }
```

The output of this program is

(a) 12 7 6 (b) 22 12 11
(c) 14 6 6 (d) 7 6 6

(GATE 2003: 2 Marks)

Solution:

P(&x) where x = 5

x = *y + 2 = 5 + 2 = 7

Q(x) = Q (7)

→ Here z = 7

→ z = z + x = 7 + 5 = 12

→ Print 12

*y = x − 1 = 7 − 1 = 6

print(x) = print 7

Value returned by function P is 6. So main will print 6.

So, answer is 12 7 6

Ans. (a)

12. Consider the function *f* defined below:

```
struct item {
 int data;
 struct item * next;
};
int f(struct item *p) {
 return ((p == NULL) || (p ->next ==
   NULL) ||
 ((p->data <= p -> next -> data) &&
 f(p-> next)));
}
```

For a given linked list *p*, the function *f* returns 1 if and only if

(a) The list is empty or has exactly one element.
(b) The elements in the list are sorted in non-decreasing order of data value.
(c) The elements in the list are sorted in non-increasing order of data value.
(d) Not all elements in the list have the same data value.

(GATE 2003: 2 Marks)

Solution: The given function returns 1 if the elements are unique and are sorted increasing order.

Ans. (b)

13. Consider the following class definitions in a hypothetical object oriented language that supports inheritance and uses dynamic binding. The language should not be assumed to be either Java or C++, though the syntax is similar.

```
class P                 | Class Q subclass of
{                       | P
void f(int i)           | {
      {                 | void f(int i)
      print(i);         |      {
      }                 |      print(2*i);
}                       |      }
                        | }
```

Now consider the following program fragment:

```
Px = new Q();
Qy = new Q();
Pz = new Q();
x.f(1); ((P)y)·f(1); z·f(1);
```

where ((P)y) denotes a typecast of y to P. The output produced by executing the above program fragment will be

(a) 1 2 1 (b) 2 1 1 (c) 2 1 2 (d) 2 2 2

(GATE 2003: 2 Marks)

Solution:

```
Px = newQ();
Qy = newQ();
Pz = newQ();
x : f(1); print 2 * i = 2
((P)y) : f(1);
z : f(1) print 2 * i = 2
```

It will print 2 because typecast to parent class cannot prevent overriding. So, function f(1) of class Q will be called not f(1) of class P.

Ans. (d)

14. Which of the following is NOT an advantage of using shared, dynamically linked libraries as opposed to using statically linked libraries?

(a) Smaller sizes of executable files
(b) Lesser overall page fault rate in the system
(c) Faster program startup
(d) Existing programs need not be re-linked to take advantage of newer versions of libraries

(GATE 2003: 2 Marks)

Solution: The shared, dynamic linked libraries use smaller sizes of executable files, lesser overall page fault rate and faster startup of programs.

Ans. (d)

15. The goal of structured programming is to

(a) Have well-indented programs.
(b) Be able to infer the flow of control from the compiled code.
(c) Be able to infer the flow of control from the program text.
(d) Avoid the use of GOTO statements.

(GATE 2004: 1 Mark)

Solution: The goal of structured programming is that the user can control the flow according his requirement.

Ans. (c)

16. Consider the following C function

```
void swap (int a, int b)
{
int temp;
temp = a ;
a = b ;
b= temp ;
}
```

In order to exchange the values of two variables x and y:

(a) Call swap(x, y)
(b) Call swap(&x, &y)
(c) swap(x, y) cannot be used as it does not return any value.
(d) swap(x, y) cannot be used as the parameters are passed by value.

(GATE 2004: 1 Mark)

Solution: The value will not be interchanged. The values are passed by value, not by reference.

Ans. (d)

17. A single array $A[1..\text{MAXSIZE}]$ is used to implement two stacks. The two stacks grow from opposite ends of the array. Variables top 1 and top 2 (top 1 < top 2) point to the location of the topmost element in each of the stacks. If the space is to be used efficiently, the condition for 'stack full' is

(a) (top 1 = MAXSIZE/2) and (top 2 = MAXSIZE/2 + 1)
(b) top 1 + top 2 = MAXSIZE
(c) (top 1 = MAXSIZE/2) or (top 2 = MAXSIZE)
(d) top 1 = top 2 − 1

(GATE 2004: 1 Mark)

Solution: If we are to use space efficiently then size of any stack can be more than MAXSIZE/2.

Both stacks will grow from both ends and if any of the stack top reaches near to the other top then stacks are full. So the condition will be top1 = top2 −1 (given that top1 < top2)

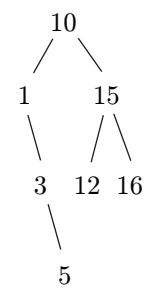
Ans. (d)

18. The following numbers are inserted into an empty binary search tree in the given order: 10, 1, 3, 5, 15, 12, 16. What is the height of the binary search tree (the height is the maximum distance of a leaf node from the root)?

(a) 2 (b) 3
(c) 4 (d) 6

(GATE 2004: 1 Mark)

Solution:



Ans. (b)

19. Given the following input (4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199) and the hash function x mod 10, which of the following statements are true?

I. 9679, 1989, 4199 hash to the same value.
II. 1471, 6171 hash to the same value.
III. All elements hash to the same value.
IV. Each element hashes to a different value.

(a) I only (b) II only
(c) I and II only (d) III or IV

(GATE 2004: 1 Mark)

Solution:

Mod 10 of all the elements in the list is given below:

4322 mod 10 = 2
1334 mod 10 = 4
1471 mod 10 = 1
9679 mod 10 = 9
1989 mod 10 = 9
6171 mod 10 = 1
6173 mod 10 = 3
4199 mod 10 = 9

From the list, it is clearly seen that 9679, 1989 and 4199 have same hash values 9. But, 1471 and 6171 have hash value of 1. So, options I and II are correct.

Ans. (c)

20. The best data structure to check whether an arithmetic expression has balanced parentheses is a

(a) queue. (b) stack.
(c) tree. (d) list.

(GATE 2004: 1 Mark)

Solution: Stack is used to check balancing of parenthesis.

Ans. (b)

21. Level order traversal of a rooted tree can be done by starting from the root and performing

(a) Preorder traversal. (b) Inorder traversal.
(c) Depth first search. (d) Breadth first search.

(GATE 2004: 1 Mark)

Solution: Breath first search gives the level order traversal.

Ans. (d)

22. Consider the following C function:

```
int f(int n)
{ staticint i = 1 ;
      if (n >= 5) return n;
      n = n+i;
      i++;
      return f(n);
      }
```

The value returned by $f(1)$ is

(a) 5. (b) 6.
(c) 7. (d) 8.

(GATE 2004: 2 Marks)

Solution:

Iteration	n	i
0	1	1
1	$1 + 1 = 2$	2
2	$2 + 2 = 4$	3
3	$4 + 3 = 7$	4
4	$7 \geq 5$ Return 7	

Ans. (c)

23. Consider the following program fragment for reversing the digits in a given integer to obtain a new integer. Let $n = d_1\ d_2\ \ldots\ d_m$.

```
int n, rev;
        rev = 0;
        while (n > 0) {
        rev = rev * 10 + n % 10;
        n = n / 10 ;
            }
```

The loop invariant condition at the end of the i iteration is:

(a) $n = d_1 d_2 \ldots d_{m-1}$ and rev $= d_m d_{m-1} \ldots d_{m-1+1}$
(b) $n = d_{m+1} \ldots d_{m-1} d_m$ or rev $= d_{m-1} \ldots d_2 d_1$
(c) $n \neq$ rev
(d) $n = d_1 d_2 \ldots_1 d_m$ or rev $= d_m \ldots d_2 d_1$

(GATE 2004: 2 Marks)

Solution: A loop invariant is a condition that must be true before and after each iteration of a loop. So $n = d_1 d_2 \ldots d_{m-1}$ and rev $= d_m d_{m-1} \ldots d_{m-1+1}$ provides the correct loop invariant condition.

Ans. (a)

24. Consider the following C program segment:

```
char p [20]
char * s = "string";
int length = strlen (s);
for (i = 0 ; i < length; i++)
p[i] = s[length - i];
print f("%",p);
```

The output of the program is

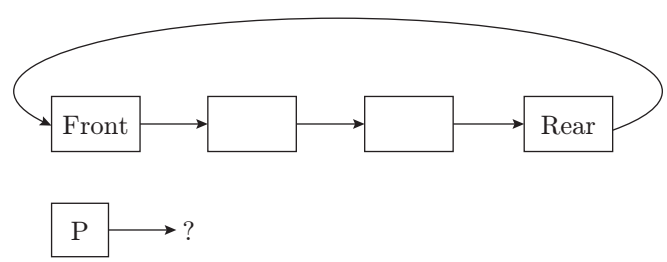
(a) gnirts. (b) String.
(c) gnirt. (d) No output is printed.

(GATE 2004: 2 Marks)

Solution: In C value of pointer variable cannot be assigned to normal character variable.

Ans. (d)

25. A circularly linked list is used to represent a queue. A single variable p is used to access the queue. To which node should p point such that both the operations enqueue and dequeue can be performed in constant time?



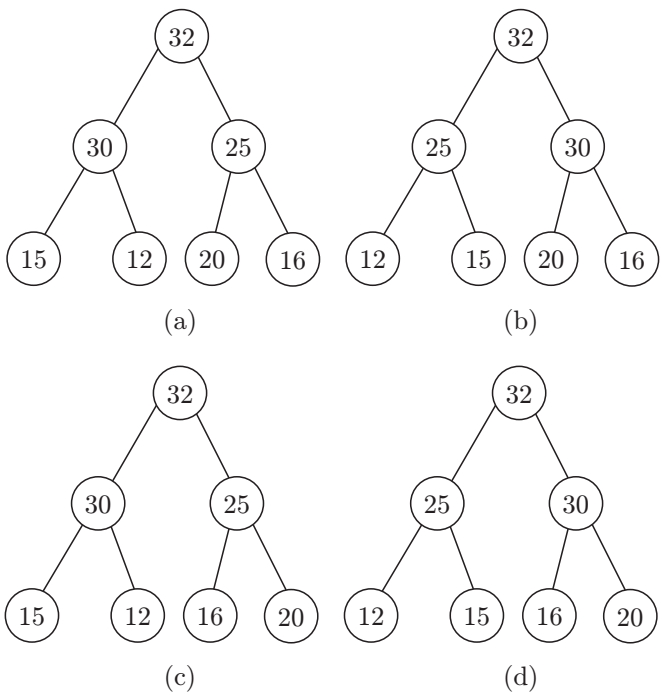
(a) Rear node
(b) Front node
(c) Not possible with a single pointer
(d) Node next to front

(GATE 2004: 2 Marks)

Solution: Required condition is not possible with single pointer.

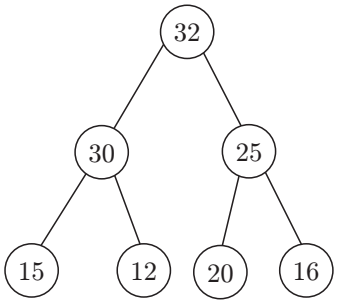
Ans. (c)

26. The elements 32, 15, 20, 30, 12, 25, 16 are inserted one by one in the given order into a maxHeap. The resultant maxHeap is



(GATE 2004: 2 Marks)

Solution: Root is greater than both left and right children.



Ans. (a)

27. Assume that the operators $+$, $-$, $\times$ are left associative and $\wedge$ is right associative. The order of precedence (from highest to lowest) is $\wedge$, $\times$, $+$, $-$.

The postfix expression corresponding to the infix expression $a + b \times c - d \wedge e \wedge f$ is

(a) $abc \times + def \wedge \wedge -$
(b) $abc \times + de \wedge f \wedge -$
(c) $ab + c \times d - e \wedge f \wedge$
(d) $- + a \times bc \wedge \wedge def$

(GATE 2004: 2 Marks)

Solution:

$a + b \times c - d \wedge e \wedge f$

$a + b \times c - d \wedge e\ f \wedge$

$a + b \times c - d\ e\ f \wedge \wedge$

$a + b\ c \times - d\ e\ f \wedge \wedge$

$a\ b\ c \times + - d\ e\ f \wedge \wedge$

$a\ b\ c \times + d\ e\ f \wedge \wedge -$

Ans. (a)

28. Consider the following C program:

```
        main ( )
{ int x, y, m, n;
scanf ("%d %d", &x, &y);
/ *Assume x > 0 and y > 0 */
m = x; n = y;
 while (m! = n)
        { if (m > n)
                m = m - n;
        else
                n = n - m;
         }
printf("%d",n);
}
```

The program computes

(a) $x + y$ using repeated subtraction.
(b) x mod y using repeated subtraction.
(c) The greatest common divisor of x and y.
(d) The least common multiple of x and y.

(GATE 2004: 2 Marks)

Solution: The given program finds greatest common divisor of x and y.

Ans. (c)

29. What does the following algorithm approximate? (Assume $m > 1$, $\varepsilon > 0$)

```
 x = m;
 y = 1;
while (x - y > ε)
       {    x = (x + y)/2;
            y = m/x;
       }
print(x);
```

(a) $\log m$
(b) m^2
(c) $m^{1/2}$
(d) $m^{1/3}$

(GATE 2004: 2 Marks)

Solution: The given program finds the square root of program.

Ans. (c)

30. Consider the following C program segment:

```
struct CellNode{
         structCellNode
           *leftChild;
         int element;
         structCellNode
           *rightChild;
         };
intDosomething (structCellNode *ptr)
{
  int value = 0;
  if (ptr ! = NULL)
    { if (ptr -> leftChild ! = NULL)
     value = 1 + DoSomething (ptr -
       >leftChild);
     if (ptr -> rightChild ! = NULL)
     value = max(value,1 +
       DoSomething (ptr->rightChild));
           }
  return (value);
}
```

The value returned by the function DoSomething when a pointer to the root of a non-empty tree is passed as argument is

(a) The number of leaf nodes in the tree.
(b) The number of nodes in the tree.
(c) The number of internal nodes in the tree.
(d) The height of the tree.

(GATE 2004: 2 Marks)

Solution: DoSomething( ) returns max(height of left child + 1, height of left child + 1). So given that pointer to root of tree is passed to DoSomething( ), it will return height of the tree. Note that this implementation follows the convention where height of a single node is 0.

Ans. (d)

31. Choose the best matching between the programming styles in Group 1 and their characteristics in Group 2.

Group 1	Group 2
I. Functional	A. Command-based, procedural
II. Logic	B. Imperative, abstract data types
III. Object-oriented	C. Side-effect free, declarative, expression evaluation
IV. Imperative	D. Declarative, clausal representation, theorem proving

(a) I-B, II-C, III-D, IV-A
(b) I-D, II-C, III-B, IV-A
(c) I-C, II-D, III-A, IV-B
(d) I-C, II-D, III-B, IV-A

(GATE 2004: 2 Marks)

Solution: Functional languages have no side effects and follow the declarative approach. Logic languages are declarative and used for theorem proving. Object-oriented languages use abstract data types. Imperative languages are command based and procedural.

Ans. (d)

32. What does the following C-statement declare?

```
int (*f) (int * );
```

(a) A function that takes an integer pointer as argument and returns an integer.
(b) A function that takes an integer as argument and returns an integer pointer.
(c) A pointer to a function that takes an integer pointer as argument and returns an integer.
(d) A function that takes an integer pointer as argument and returns a function pointer.

(GATE 2005: 1 Mark)

Solution:

⇒ int (* f) (int *)

⇒ return type is int, (*f) means pointer to function with argument as integer pointer.

Ans. (c)

33. An abstract data type (ADT) is:

(a) Same as an abstract class.
(b) A data type that cannot be instantiated.
(c) A data type for which only the operations defined on it can be used, but none else.
(d) All of the above.

(GATE 2005: 1 Mark)

Solution: ADT is user-defined data type that specifies the set of operations.

Ans. (c)

34. A common property of logic programming languages and functional languages is:

(a) Both are procedural languages.
(b) Both are based on λ calculus.
(c) Both are declarative.
(d) Both use Horn-clauses.

(GATE 2005: 1 Mark)

Solution: Both are declarative because we declare variable before use.

Ans. (c)

35. Which one of the following are essential features of an object-oriented programming language?

I Abstraction and encapsulation
II Strictly typedness
III Type-safe property coupled with sub-type rule
IV Polymorphism in the presence of inheritance

(a) I and II only (b) I and IV only
(c) I, II and IV only (d) I, III and IV only

(GATE 2005: 1 Mark)

Solution: Common property of OOP is abstraction and encapsulation and polymorphism in the presence of inheritance.

Ans. (b)

36. A program P reads in 500 integers in the range [0, 100] representing the scores of 500 students. It then prints the frequency of each score above 50. What would be the best way for P to store the frequencies?

(a) An array of 50 numbers
(b) An array of 100 numbers
(c) An array of 500 numbers
(d) A dynamically allocated array of 550 numbers

(GATE 2005: 1 Mark)

Solution: We need to know the frequency of numbers above 50. So, array of size 50 will be sufficient.

Ans. (a)

37. Consider the following C program:

```
void foo (int n, int sum 0) {
int k = 0, j = 0;
if (n==0) return;
k = n % 10; j = n / 10;
sum = sum + k;
foo (j, sum);
printf ("%d,", k);
 }
int main () {
int a = 2048, sum = 0;
foo (a, sum);
printf("%d\n", sum);
}
```

What does the above program print?

(a) 8, 4, 0, 2, 14 (b) 8, 4, 0, 2, 0
(c) 2, 0, 4, 8, 14 (d) 2, 0, 4, 8, 0

(GATE 2005: 2 Marks)

Solution:

Iteration	k	j	foo(n, sum)
1, n = 2048, Sum = 0	8	204	foo(204,8)
2, n = 204, Sum = 8	4	20	foo(20, 12)
3, n= 20, Sum = 12	0	2	foo(2,12)
4, n = 2, Sum = 12	2	0	foo(0,14), for this call function will return.

For recursive function calls, stack is used to store values. So the values will be printed as 2 0 4 8. And value of sum is 0. Because sum is passed by value not by reference. Changes are not reflected back to the main program.

Ans. (d)

38. Consider the following C program:

```
double foo(double); /* Line 1 */
int main() {
double da, db;
// input da
db = foo(da);
}
double foo(double a) {
return a;
 }
```

The above code compiled without any error or warning. If line 1 is deleted, the above code will show:

(a) No compile warning or error.
(b) Some compiler warnings not leading to unintended results.
(c) Some compiler warnings due to type-mismatch eventually leading to unintended results.
(d) Compiler errors.

(GATE 2005: 2 Marks)

Solution: Compilation error will be thrown by deleting Line1. Function is called before it is declared anywhere.

Ans. (c)

39. Postorder traversal of a given binary search tree T produces the following sequence of keys

10, 9, 23, 22, 27, 25, 15, 50, 95, 60, 40, 29

Which one of the following sequences of keys can be the result of an inorder traversal of the tree T?

(a) 9, 10, 15, 22, 23, 25, 27, 29, 40, 50, 60, 95
(b) 9, 10, 15, 22, 40, 50, 60, 95, 23, 25, 27, 29
(c) 29, 15, 9, 10, 25, 22, 23, 27, 40, 60, 50, 95
(d) 95, 50, 60, 40, 27, 23, 22, 25, 10, 9, 15, 29

(GATE 2005: 2 Marks)

Solution: Inorder traversal is always sorted in increasing order.

Ans. (a)

40. A priority queue is implemented as a max-heap. Initially, it has five elements. The level-order traversal of the heap is given below:

10, 8, 5, 3, 2

Two new elements '1' and '7' are inserted in the heap in that order. The level-order traversal of the heap after the insertion of the elements is:

(a) 10, 8, 7, 5, 3, 2, 1 (b) 10, 8, 7, 2, 3, 1, 5
(c) 10, 8, 7, 1, 2, 3, 5 (d) 10, 8, 7, 3, 2, 1, 5

(GATE 2005: 2 Marks)

Solution:

```
   10            10            10              10
  /  \          /  \          /  \            /   \
 8    5        8    5        8    5          8     7
/ \           / \  /        / \  / \        / \   / \
3  2          3  2 1        3  2 1  7       3  2  1  5
```

Level order traversal is 10, 8, 7, 3, 2, 1, 5

Ans. (d)

41. How many distinct binary search trees can be created out of 4 distinct keys?

(a) 5 (b) 14 (c) 24 (d) 42

(GATE 2005: 2 Marks)

Solution: Number of distinct binary trees = $1/1+n\ ({}^{2n}\mathrm{C}_n) = 14$

Ans. (b)

42. In a complete k-ary tree, every internal node has exactly k children. The number of leaves in such a tree with n internal nodes is:

(a) nk (b) $(n-1)k+1$
(c) $n(k-1)+1$ (d) $n(k-1)$

(GATE 2005: 2 Marks)

Solution: If there are n internal nodes, then number of leave nodes will be $n(k-1)+1$

Ans. (c)

43. Consider line number 3 of the following C program:

```
int min ( ) {            /* Line 1 */
int I, N;                  /* Line 2 */
fro (I =0, I<N, I++);  /* Line 3 */
}
```

Identify the compiler's response about this line while creating the object-module:

(a) No compilation error
(b) Only a lexical error
(c) Only syntactic errors
(d) Both lexical and syntactic errors

(GATE 2005: 2 Marks)

Solution: Misspellings of int keyword in line 2 and there should be 'for' instead of 'fro' in line 3.

Ans. (c)

Linked Answer Questions 44 and 45: Let s and t be two vertices in a undirected graph $G = (V, E)$ having distinct positive edge weights. Let $[X, Y]$ be a partition of V such that $s \in X$ and $T \in Y$. Consider the edge e having the minimum weight amongst all those edges that have one vertex in X and one vertex in Y.

44. The edge e must definitely belong to:

(a) the minimum weighted spanning tree of G
(b) the weighted shortest path from s to t
(c) each path from s to t
(d) the weighted longest path from s to t

(GATE 2005: 2 Marks)

Solution: Minimum weight edge is included in minimum spanning tree.

Ans. (a)

45. Let the weight of an edge e denote the congestion on that edge. The congestion on a path is defined to be the maximum of the congestions on the edges of the path. We wish to find the path from s to t having minimum congestion. Which one of the following paths is always such a path of minimum congestion?

(a) a path from s to t in the minimum weighted spanning tree
(b) a weighted shortest path from s to t
(c) an Euler walk from s to t
(d) a Hamiltonian path from s to t

(GATE 2005: 2 Marks)

Solution: Path from s to t in the minimum weighted spanning tree is the path of minimum congestion.

Ans. (a)

46. In a binary max-heap containing n numbers, the smallest element can be found in time

(a) $O(n)$ (b) $O(\log n)$
(c) $O(\log \log n)$ (d) $O(1)$

(GATE 2006: 1 Mark)

Solution: In max heap smallest element will be present in the leaf node. To get it, n nodes need to be traversed.

Ans. (b)

47. Let T be a depth first search tree in an undirected graph G. Vertices u and v are leaves of this tree T. The degrees of both u and v in G are at least 2. Which one of the following statements is true?

(a) There must exist a vertex w adjacent to both u and v in G
(b) There must exist a vertex w whose removal disconnects u and v in G
(c) There must exist a cycle in G containing u and v
(d) There must exist a cycle in G containing u and all its neighbours in G.

(GATE 2006: 1 Mark)

Solution: There must exist a vertex w adjacent to both u and v in G. So, option (a) is true.

Ans. (a)

48. A scheme for storing binary trees in an array X is as follows. Indexing of X starts at 1 instead of 0. The root is stored at $X[1]$. For a node stored at $X[i]$, the left child, if any, is stored in $X[2i]$ and the right child, if any, in $X[2i + 1]$. To be able to store any binary tree on n vertices, the minimum size of X should be

(a) $\log_2 n$ (b) n
(c) $2n + 1$ (d) $2^n - 1$

(GATE 2006: 1 Mark)

Solution: In case of right skewed tree, the size of array will be $2^n - 1$.

Ans. (d)

49. An element in an array X is called a leader if it is greater than all elements to the right of it in X. The best algorithm to find all leaders in an array

(a) Solves it in linear time using a left to right pass of the array
(b) Solves it in linear time using a right to left pass of the array
(c) Solves it using divide and conquer in time $\Theta(n \log n)$
(d) Solves it in time $\Theta(n^2)$

(GATE 2006: 1 Mark)

Solution: Best algorithm to find leader is pass the array from right side. With each traversal of node, maximum will be updated. Hence leader can be updated in O(n). So, option (b) is correct.

Ans. (b)

50. An implementation of a queue Q, using two stacks S_1 and S_2, is given below:

```
void insert (Q, x) {
      push (S1, x);
}
void delete (Q) {
      if (stack-empty(S2)) then
            if (stack-empty(S1)) then {
                  print("Q is empty");
                  return;
            }
      else while (!(stack-empty(S1))){
            x=pop(S1);
            push(S2,x);
            }
       x=pop(S2);
 }
```

Let n insert and m ($\leq$ n) delete operations be performed in an arbitrary order on an empty queue Q. Let x and y be the number of push and pop operations performed, respectively, in the process. Which one of the following is true for all m and n?

(a) $n + m \leq x < 2n$ and $2m \leq y \leq n + m$
(b) $n + m \leq x < 2n$ and $2m \leq y \leq 2n$
(c) $2m \leq x < 2n$ and $2m \leq y \leq n + m$
(d) $2m \leq x \leq 2n$ and $2m \leq y \leq 2n$

(GATE 2006: 2 Marks)

Solution:
Number of push operation = insert operation + delete operation
$\Rightarrow n + m$
$\Rightarrow n + m \leq x$
$\Rightarrow n + m \leq x < 2n$ and $2m \leq y \leq n + m$

Ans. (a)

51. Consider the following C function in which a[n] and b[m] are two sorted integer arrays and c[n + m] be another integer array.

```
void xyz(int a[], int b [], int c []){
            int i,j,k;
            i=j=k=0;
            while ((i<n) && (j<m))
                  if (a[i] < b[j])
                    c[k++] = a[i++];
                  else c[k++] = b[j++];
      }
```

Which of the following condition(s) hold(s) after the termination of the while loop?

(i) $j < m$, $k = n + j - 1$ and $a[n - 1] < b[j]$ if $i = n$
(ii) $i < n$, $k = m + i - 1$ and $b[m - 1] \leq a[i]$ if $j = m$

(a) Only (i)
(b) Only (ii)
(c) Either (i) or (ii) but not both
(d) Neither (i) nor (ii)

(GATE 2006: 2 Marks)

Solution:
$x[i] \wedge \sim y[i])$ = only 1's in x where corresponding entry in y is $0 \rightarrow x - y$
$\sim x[i] \wedge y[i]$ = only 1s in y where corresponding entry in x is $0 \rightarrow y - x$.
The operator "V" results in union of the above two sets.

Ans. (c)

52. Given two arrays of numbers $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ where each number is 0 or 1, the fastest algorithm to find the largest span (i, j) such that $a_i + a_{i+1} + \ldots + a_j = b_i + b_{i+1} + \ldots + b_j$, or report that there is not such span,

(a) Takes $O(3^n)$ and $\Omega(2^n)$ time if hashing is permitted.
(b) Takes $O(n^3)$ and $\Omega(n^{2.5})$ time in the key comparison model.
(c) Takes $\Theta(n)$ time and space.
(d) Takes $O(\sqrt{n})$ time only if the sum of the $2n$ elements is an even number.

(GATE 2006: 2 Marks)

Solution: To find the largest span (i, j), average $\Theta(n)$ time and space is required.

Ans. (c)

53. Consider these two functions and two statements S_1 and S_2 about them:

```
int work1(int *a,
  int i, int j)
    {
    int x = a[i+2];
    a[j] = x+1;
    return a[i+2]
      - 3;
    }
```

```
int work2(int *a, int
  i, int j)
    {
    int t1 = i+2;
      int t2 = a[t1];
    a[j] = t2+1;
    return t2 - 3;
    }
```

S_1: The transformation from work 1 to work 2 is valid, that is, for any program state and input arguments, work 2 will compute the same output and have the same effect on program state as work 1.

S_2: All the transformations applied to work 1 to get work 2 will always improve the performance (i.e reduce CPU time) of work 2 compared to work 1.

(a) S_1 is false and S_2 is false
(b) S_1 is false and S_2 is true
(c) S_1 is true and S_2 is false
(d) S_1 is true and S_2 is true

(GATE 2006: 2 Marks)

Solution:
S_1: True, both work1 and work2 perform the same task
S_2: True, work2 will improve the performance because it reduces calculations using temporary variables to store results.
S_1 and S_2 are correct statements.

Ans. (d)

54. Consider the following code written in a pass-by-reference language such as FORTRAN and these statements about the code.

```
subroutine swap(ix,iy)
      it = ix
L1 : ix = iy
L2 :iy = it
    end
    ia = 3
    ib = 8
    call swap (ia, 1b+5)
    print *, ia, ib
    end
```

S_1: The compiler will generate code to allocate a temporary nameless cell, initialise it to 13 and pass the address of the cell swap
S_2: On execution, the code will generate a runtime error on line L1
S_3: On execution, the code will generate a runtime error on line L2
S_4: The program will print 13 and 8
S_5: The program will print 13 and -2
Exactly the following set of statement(s) is correct:

(a) S_1 and S_2 (b) S_1 and S_4
(c) S_3 (d) S_1 and S_5

(GATE 2006: 2 Marks)

Solution: S_1 and S_2 are correct statements.

Ans. (a)

55. Consider this C code to swap two integers and these five statements: the code

```
void swap (int *px, int *py) {
      *px = *px - *py;
      *py = *px + *py;
      *px = *py - *px;
}
```

S_1: Will generate a compilation error
S_2: May generate a segmentation fault at runtime depending on the arguments passed

S_3: Correctly implements the swap procedure for all input pointers referring to integers stored in memory locations accessible to the process
S_4: Implements the swap procedure correctly for some but not all valid input pointers
S_5: May add or subtract integers and pointers

(a) S_1 (b) S_2 and S_3
(c) S_2 and S_4 (d) S_2 and S_5

(GATE 2006: 2 Marks)

Solution:
S_2: May generate segmentation fault if value at pointers px or py is constant or px or py points to a memory location that is invalid.
S_4: May not work for all inputs as arithmetic overflow can occur.

Ans. (c)

Common Data Questions 56 and 57: A 3-ary max-heap is like a binary max-heap, but instead of 2 children, nodes have 3 children. A 3-ary heap can be represented by an array as follows: The root is stored in the first location, $a[0]$, nodes in the next level, from left to right, are stored from $a[1]$ to $a[3]$. The nodes from the second level of the tree from left to right are stored from $a[4]$ location onward. An item x can be inserted into a 3-ary heap containing n items by placing x in the location $a[n]$ and pushing it up the tree to satisfy the heap property.

56. Which one of the following is a valid sequence of elements in an array representing 3-ary max-heap?

(a) 1, 3, 5, 6, 8, 9 (b) 9, 6, 3, 1, 8, 5
(c) 9, 3, 6, 8, 5, 1 (d) 9, 5, 6, 8, 3, 1

(GATE 2006: 2 Marks)

Solution: The 3-ary max heap can be formed by 9, 5, 6, 8, 3, 1.

Ans. (d)

57. Suppose the elements 7, 2, 10 and 4 are inserted, in that order, into the valid 3-ary max-heap found in the above question Q.56. Which one of the following is the sequence of items in the array representing the resultant heap?

(a) 10, 7, 9, 8, 3, 1, 5, 2, 6, 4
(b) 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
(c) 10, 9, 4, 5, 7, 6, 8, 2, 1, 3
(d) 10, 8, 6, 9, 7, 2, 3, 4, 1, 5

(GATE 2006: 2 Marks)

Solution: After inserting 7, 2, 10, 4, the heap sequence will be 10, 7, 9, 8, 3, 1, 5, 2, 6, 4.

Ans. (a)

58. Consider the following segment of C code:

```
int j, n;
    j = 1;
      while (j <=n)
        j = j*2;
```

The number of comparisons made in the execution of the loop for any $n > 0$ is:

(a) $\lceil \log_2 n \rceil + 1$ (b) n
(c) $\lceil \log_2 n \rceil$ (d) $\lceil \log_2 n \rceil + 1$

(GATE 2007: 1 Mark)

Solution: Check using values $n = 3,4,5\ldots$.

Ans. (d)

59. The maximum number of binary trees that can be formed with three unlabeled nodes is:

(a) 1 (b) 5
(c) 4 (d) 3

(GATE 2007: 1 Mark)

Solution: Maximum number of binary trees $= 2^n - n = 2^3 - 3 = 5$

Ans. (b)

60. The height of a binary tree is the maximum number of edges in any root to leaf path. The maximum number of nodes in a binary tree of height h is:

(GATE 2007: 1 Mark)

(a) $2^h - 1$ (b) $2^{h-1} - 1$ (c) $2^{h+1} - 1$ (d) 2^{h+1}.

Solution: Maximum number of node with height $h = 1 + 2 + 4 + \ldots + 2^h = 2^{h+1} - 1$

Ans. (c)

61. The following postfix expression with single-digit operands is evaluated using a stack:

$$8\ 2\ 3\ \wedge\ /\ 2\ 3\ *\ +\ 5\ 1\ *\ -$$

Note that $\wedge$ is the exponentiation operator. The top two elements of the stack after the first * is evaluated are:

(a) 6, 1 (b) 5, 7
(c) 3, 2 (d) 1, 5

(GATE 2007: 2 Marks)

Solution:

Expression	TOS
8	8
2	8,2
3	8,2,3
^	8,8
/	1
2	1,2
3	1,2,3
*	1,6

Top of stack contain 1,6.

Ans. (a)

62. The inorder and preorder traversal of a binary tree are d b e a f c g and a b d e c f g, respectively. The postorder traversal of the binary tree is:

(a) d e b f g c a (b) e d b g f c a
(c) e d b f g c a (d) d e f g b c a

(GATE 2007: 2 Marks)

Solution: Construct equivalent tree from inorder and preorder traversal.

```
      a
     / \
    b   c
   / \ / \
  d e f g
```

Postorder traversal is: d e b f g c a

Ans. (a)

63. Consider a hash table of size seven, with starting index zero, and a hash function $(3x + 4)$ mod 7. Assuming the hash table is initially empty, which of the following is the contents of the table when the sequence 1, 3, 8, 10 is inserted into the table using closed hashing? Note that - denotes an empty location in the table.

(a) 8, - , - , - , - , - , 10 (b) 1, 8, 10, - , - , - , 3
(c) 1, - , - , - , - , - , 3 (d) 1, 10, 8, - , - , - , 3

(GATE 2007: 2 Marks)

Solution:
Hash function is: $(3x + 4)$ mod 7
X = 1 7 mod 7 = 0
X = 3 13 mod 7 = 6
X = 8 28 mod 7 = 0 since 0 occupied, it will be on 1st position
X = 10 34 mod 7 = 6 since 6,0,1 is occupied it will at 2nd position
So 1 8 10 - - - 3

Ans. (b)

64. Consider the following C function:

```
 int f(int n)
{static int r = 0;
            if (n <= 0) return 1;
             if (n > 3)
            {r = n;
            return f(n-2)+2;
             }
            return f(n-1)+r;
}
```

What is the value of f (5)?

(a) 5 (b) 7 (c) 9 (d) 18

(GATE 2007: 2 Marks)

Solution:

f(5)	r = 5 , 2 + f(3)
f(3)	2 + 5 + f(2)
f(2)	7 + 5 + f(1)
f(1)	12 + 5 + f(0)
f(0)	17 + 1 = 18

Ans. (d)

65. A complete n-ary tree is a tree in which each node has n children or no children. Let I be the number of internal nodes and L be the number of leaves in a complete n-ary tree. If $L = 41$, and $I = 10$, what is the value of n?

(a) 3 (b) 4 (c) 5 (d) 6

(GATE 2007: 2 Marks)

Solution:

$I[n - 1] + 1 = L$
$N = (L - 1)/I + 1 = 40/10 + 1 = 5$

Ans. (c)

66. Consider the following C program segment where CellNode represents a node in a binary tree:

```
      structCellNode {
              structCellNode *leftChild;
                int element;
      structCellNode *rightChild;
      };
intGetValue (structCellNode *ptr) {
      int value = 0;
      if (ptr != NULL) {
                if ((ptr->leftChild ==
                  NULL) &&
                        (ptr->rightChild
                          == NULL))
                         value = 1;
      else
```

```
        value = value + GetValue
          (ptr->leftChild) + GetValue
          (ptr->rightChild);
        }
return(value);
}
```

The value returned by GetValue when a pointer to the root of a binary tree is passed as its argument is:

(a) The number of nodes in the tree
(b) The number of internal nodes in the tree
(c) The number of leaf nodes in the tree
(d) The height of the tree

(GATE 2007: 2 Marks)

Solution: As we can see in else part of given program, left and right subtrees are traversed recursively and value is incremented if they have leaf nodes.

Ans. (c)

67. Consider the process of inserting an element into a max-heap, where the max-heap is represented by an array. Suppose we perform a binary search on the path from the new leaf to the root to find the position for the newly inserted element, the number of comparisons performed is:

(a) $\Theta(\log_2 n)$ (b) $\Theta(\log_2 \log_2 n)$
(c) $\Theta(n)$ (d) $\Theta(n \log_2 n)$

(GATE 2007: 2 Marks)

Solution: Inserting an element in binary tree takes $\Theta(1)$ and binary search takes $\Theta(\log n)$.

Ans. (a)

68. An array of n numbers is given, where n is an even number. The maximum as well as the minimum of these n numbers need to be determined. Which of the following is TRUE about the number of comparisons needed?

(a) At least $2n - c$ comparisons, for some constant, c are needed.
(b) At most $1.5 - 2$ comparisons are needed.
(c) At least $n\log_2 n$ comparisons are needed.
(d) None of the above.

(GATE 2007: 2 Marks)

Solution: Using divide and conquer technique problem is divided into two parts of $n/2$.

$T(n) = 2\ T(n/2) + 2$

Solution of this equation is $3/2\ n - 2$ comparisons

Ans. (b)

69. Which combination of the integer variables x, y and z makes the variable a get the value 4 in the following expression?

$a = (x > y)\ ?\ ((x > z)\ ?\ x{:}z){:}((y > z)\ ?\ y{:}z)$

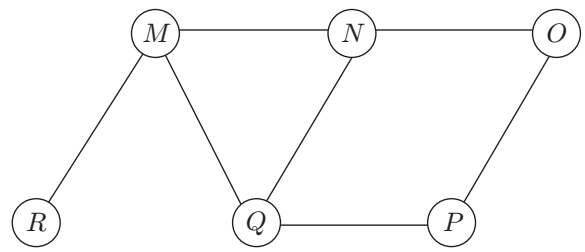
(a) $x = 3,\ y = 4,\ z = 2$
(b) $x = 6,\ y = 5,\ z = 3$
(c) $x = 6,\ y = 3,\ z = 5$
(d) $x = 5,\ y = 4,\ z = 5$

(GATE 2008: 1 Mark)

Solution: In option A, $3 > 4$ is false so it moves to else part. In else condition, $4 > 2$ is true and 4 will be printed.

Ans. (a)

70. The breadth first search algorithm has been implemented using the queue data structure. One possible order of visiting the nodes of the following graph is



(a) *MNOPQR* (b) *NQMPOR*
(c) *QMNPRO* (d) *QMNPOR*

(GATE 2008: 1 Mark)

Solution: In BFS, all the children of node will be traversed first.

Ans. (a)

71. You are given the postorder traversal, P, of a binary search tree on the n elements 1, 2, ..., n. You have to determine the unique binary search tree that has P as its postorder traversal. What is the time complexity of the most efficient algorithm for doing this?

(a) Θ ($\log n$)
(b) Θ (n)
(c) Θ ($n \log n$)
(d) None of the above, as the tree cannot be uniquely determined

(GATE 2008: 2 Marks)

Solution:

Step 1: The last element in the postorder traversal P will act as the root element.

Step 2: Now in P the elements that are smaller than root will form the left subtree
Step 3: The elements in P that are greater than root will form the right subtree
Step 4: Binary search could be used to find the left and right tree elements

Ans. (c)

72. We have a binary heap on n elements and wish to insert n more elements (not necessarily one after another) into this heap. The total time required for this is

(a) Θ (log n) (b) Θ (n)
(c) Θ ($n$log n) (d) Θ (n^2)

(GATE 2008: 2 Marks)

Solution: Time require to re-heapify is O(n).

Ans. (c)

73. What is printed by the following C program?

```
int f(int x, int *py, int **ppz)
{
    int y, z;
    **ppz + = 1; z = *ppz;
    *py + = 2; y = *py;
    x + = 3;
    return x + y + z:
}
void main()
{
    int c, *b, **a;
    c = 4; b = &c; a = &b;
    print f("%d", f(c, b, a));
}
```

(a) 18 (b) 19
(c) 21 (d) 22

(GATE 2008: 2 Marks)

Solution:
**ppz = 4 +1 = 5
z = 5
y = 5 + 2 = 7
x = 7 + 3 = 10
x + y + z = 10 + 7 + 5 = 22

Ans. (d)

74. Choose the correct option to fill ? 1 and ? 2 so that the program below prints an input string in reverse order. Assume that the input string is terminated by a newline character.

```
void reverse (void) {
            int c;
            if (?1) reverse ();
            ?2
}
main () {
   print f ("Enter Text");
   print f ("\ n");
   reverse ();
   print f ("\ n");
}
```

(a) ? 1 is (getchar ( )! = 'n')
? 2 isgetchar (c);

(b) ? 1 is (c = getchar ( ))! = 'n')
? 2 isgetchar (c);

(c) ? 1 is (c ! = 'n')
? 2 isputchar (c);

(d) ? 1 is ((c = getchar ( ))! = 'n')
? 2 isputchar (c);

(GATE 2008: 2 Marks)

Solution: ?1 checks the end of string, if not end of string it calls reverse function.
?2 prints the reverse string.

Ans. (d)

75. The following C function takes a single-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1,2,3,4,5,6,7 in the given order. What will be the contents of the list after the function completes execution?

```
struct node {
        int value;
        struct node * next;
};
Void rearrange (struct node * list) {
struct node * p, * q;
int temp;
if (!list || !list → next) return;
p = list; q = list →> next;
while q {
        temp = p → value; p →> value
          = q →value;
        q →value = temp; p = q → next
        q = p? p → next : 0;
        }
}
```

(a) 1,2,3,4,5,6,7 (b) 2,1,4,3,6,5,7
(c) 1,3,2,5,4,7,6 (d) 2,3,4,5,6,7,1

(GATE 2008: 2 Marks)

Solution: The program is simply swapping two consecutive numbers.
Here, input: 1,2,3,4,5,6,7
Output: 2,1,4,3,6,5,7

Ans. (b)

76. P and Q are two propositions. Which of the following logical expressions are equivalent?

I. $P \vee \sim Q$
II. $\sim (\sim P \wedge Q)$
III. $(P \wedge Q) \vee (P \wedge \sim Q) \vee (\sim P \wedge \sim Q)$
IV. $(P \wedge Q) \vee (P \wedge \sim Q) \vee (\sim P \wedge Q)$

(a) Only I and II (b) Only I, II and III
(c) Only I, II and IV (d) All of I, II III and IV

(GATE 2008: 2 Marks)

Solution: IV represent $P \vee Q$. Assuming values of P and Q can provide us the solution.

Ans. (b)

77. A B-tree of order 4 is built from scratch by 10 successive insertions. What is the maximum number of node splitting operations that may take place?

(GATE 2008: 2 Marks)

(a) 3 (b) 4 (c) 5 (d) 6

Solution: Insertion of 3 keys

10 20 30

Insertion of 4th key (1st split)

```
      30
     / \
10*20  40
```

Insertion of 5th key no split
To maximize splits, let us insert a value in a node that has key in access. Let us insert 5

```
        30
      /   \
5*10*20    40
```

Insertion of 6th key (2nd Split)
To maximize splits, let us insert a value in a node that has key in access. Let us insert 6

```
      8*30
     /  |  \
5  10 * 20 40
```

Insertion of 7th key
To maximize splits, let us insert a value in a node that has key in access. Let us insert 15

```
        8*30
      /  |  \
5 10*15*20  40
```

Insertion of 8th key (3rd Split)
To maximize splits, let us insert a value in a node that has key in access. Let us insert 12

```
        8*12*30
      / / |   \
5  10 15*20  40
```

Insertion of 9th key
To maximize splits, let us insert a value in a node that has key in access. Let us insert 17

```
        8*12*30
      / / \    \
5  10 15*17*20 40
```

Insertion of 10th key (4th and 5th Splits)
To maximize splits, let us insert a value in a node that has key in access. Let us insert 13

```
        12
      /   \
    8       15*30
   / \    / | \     \
5   10  13 17*20   40
```

Ans. (a)

Common Data Questions 78 and 79: Consider the following C functions:

```
int f1 (int n)
{
   if (n == 0 | | n == 1)
     return n;
else
  return (2* f1 (n - 1) + 3 * f1 (n - 2))
}
int f2 (int n)
{
  int i:
  int X[N], Y[N], Z[N];
  X[0] = Y[0] = Z[0] = 0;
  X[1] = 1; Y[1] = 2; Z[1] = 3;
  for (i = 2; i <= n; i ++) {
            X[i] = Y[i - 1] + Z[i - 2];
            Y[i] = 2 * X[i];
            Z[i] = 3 * X[i];
   }
return X[n];
}
```

78. The running time of $f_1(n)$ and $f_2(n)$ are

(a) $\Theta(n)$ and $\Theta(n)$ (b) $\Theta(2^n)$ and $\Theta(n)$
(c) $\Theta(n)$ and $\Theta(2^n)$ (d) $\Theta(2^n)$ and $\Theta(2^n)$

(GATE 2008: 2 Marks)

Solution: For f_1 recurrence relation is,

$T(n) = T(n-1) + T(n-2)$

This type of relations has non-polynomial complexity.

Ans. (b)

79. $f_1(8)$ and $f_2(8)$ return the values

(a) 1661 and 1640 (b) 59 and 59
(c) 1640 and 1640 (d) 1640 and 1661

(GATE 2008: 2 Marks)

Solution: Implementing the code will give the output 1640, 1640.

Ans. (c)

Common Data for Questions 80 and 81: Consider the following C program that attempts to locate an element x in an array $Y[\,]$ using binary search. The program is erroneous.

```
1.  f(int Y[10], int x) {
2.    int u, j, k;
3.    i = 0; j = 9;
4.    do {
5.              k = (i + j) / 2;
6.              if (Y[k] < x) i = k;
                  else j = k;
7.          }  while ((Y[k]! = x) &&
                  (i < j));
8.  if (Y [k] == x) print f ("x is in
      the array");
9.  else print f ("x is not in the
      array");
10. }
```

80. On which of the following contents of Y and x does the program fail?

(a) Y is [1 2 3 4 5 6 7 8 9 10] and $x < 10$.
(b) Y is [1 3 5 7 9 11 13 15 17 19] and $x < 1$.
(c) Y is [2 2 2 2 2 2 2 2 2 2] and $x > 2$.
(d) Y is [2 4 6 8 10 12 14 16 18 20] and $2 < x < 20$ and x is even.

(GATE 2008: 2 Marks)

Solution: Binary search fails for Y [2 2 2 2 2 2 2 2 2] and $x > 2$

Ans. (b)

81. The correction needed in the program to make it work properly is

(a) Change line 6 to: if (Y[k] < x) i = k + 1; else j = k − 1;
(b) Change line 6 to: if (Y[k] < x) i = k − 1; else j = k + 1;
(c) Change line 6 to: if (Y[k] <= x) i = k; else j = k;
(d) Change line 7 to:} while ((Y[k] == x) && (i < j));

(GATE 2008: 2 Marks)

Solution: Change line 6 according to option A to correct the program.

Ans. (c)

82. Consider the program below:

```
# include <stdio.h>
int fun(int n, int * f_p) {
int t, f;
if (n <= 1) {
* f_p =1;
return 1;
}
t = fun (n - 1, f_p);
f = t+ * f_p;
* f_p = t;
return f;
}
int main() {
int x = 15;
printf (" % d\ n", fun(5, & x));
return 0;
}
```

The value printed is

(a) 6 (b) 8 (c) 14 (d) 15

(GATE 2009: 1 Mark)

Solution:

Iteration	n	*f_p	t	f
	5	15 → 3	5	8 → Ans
	4	15 → 2	3	5
	3	15 → 1	2	3
	2	15 → 1	1	2
	1	15		

The function will return 8.

Ans. (b)

83. What is the maximum height of any AVL tree with 7 nodes? Assume that the height of a tree with a single node is 0.

(a) 2 (b) 3 (c) 4 (d) 5

(GATE 2009: 2 Marks)

Solution: The height of an AVL tree storing n items is $O(\log(n))$.

Ans. (b)

Linked Answer Questions 84 and 85: Consider a binary max-heap implemented using an array.

84. Which one of the following array represents a binary max-heap?

(a) {25,12,16,13,10,8,14}
(b) {25,14,13, 16,10,8,12}
(c) {25,14,16,13,10,8,12}
(d) {25,14,12,13,10,8,16}

(GATE 2009: 2 Marks)

Solution:
Conditions for max heap:

- Data at every node should be greater than its children nodes
- Left child is at 2i+1 and right at 2i+2

Ans. (c)

85. What is the content of the array after two delete operations on the correct answer to the previous question?

(a) {14, 13,12,10, 8} (b) {14,12,13, 8,10}
(c) {14,13, 8, 12,10} (d) {14,13,12, 8,10}

(GATE 2009: 2 Marks)

Solution:
Deletion can be done by following steps:

- Replace the root with last node
- Re-heapify the tree

```
        14
       /  \
      /    \
     13     12
    / \
   /   \
  8     10
```

Ans. (d)

86. In a binary tree with n nodes, every node has an odd number of descendants. Every node is considered to be its own descendant. What is the number of nodes in the tree that has exactly one child?

(a) 0 (b) 1
(c) $(n-1)/2$ (d) $n-1$

(GATE 2010: 1 Mark)

Solution: The given tree is called full binary tree which has 0 or 2 children. No node will have exactly one child node.

Ans. (a)

87. What does the following program print?

```
#include <stdio.h>
void f (int *p, int * g) {
             p = q;
            *p = 2;
            }
int i = 0, j =1;
int main ( ){
            f(&i, & j);
            print f ("%d%d / n", i, j ;
            return 0;
            }
```

(a) 2 2 (b) 2 1 (c) 0 1 (d) 0 2

(GATE 2010: 1 Mark)

Solution: i and j are passed by address, any modification by function will be reflected back.
When $p = q$ means address of q is changed.
$*p = 2$ will modify value of q to 2.

Ans. (d)

88. Which languages necessarily need heap allocation in the runtime environment?

(a) Those that support recursion
(b) Those that use dynamic scoping
(c) Those that allow dynamic data structures
(d) Those that use global variables

(GATE 2010: 1 Mark)

Solution: Runtime environment require dynamic memory allocation. Heap data structure is used for dynamic memory allocation.

Ans. (a)

89. What is the value printed by the following C program?

```
#include <stdio.h>
       int f(int * a, int n)
       {
                  if (n<= 0)return 0;
                  else if(*a% 2= = 0)
                    return * a + f(a
                    +1,n - 1);
                  else return * a -
                    f(a + 1, n - 1);
       }
       int main( )
       {
                  int a[ ] = {12, 7,
                    13, 4, 11, 6};
                  printf("%d",f(a,6));
                  return 0;
       }
```

(a) −9 (b) 5 (c) 15 (d) 19

(GATE 2010: 2 Marks)

Solution:

f(a,6)	12 + f(7,5)
f(7,5)	12 + 7 − f(13,4)
f(13,4)	12 + 7 −(13 − f(4,3))
f(4,3)	12 + 7 −(13 − (4 + f(11,2)))
f(11,2)	12 + 7 − (13 − (4 + 11 − f(6,1)))
f(6,1)	12 + 7 − (13 − (4 + 11 − 6)) = 15

Ans. (c)

90. The following C function takes a simply linked list as input argument. It modifies the list by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank.

```
typedefstruct node {
int value;
            struct node *next;
}           Node;
Node *move_to_front(Node *head) {
Node *p, *q;
  if ((head = = NULL: || (head->next =
     = NULL)) return head;
            q = NULL; p = head;
            while (p-> next !=NULL) {
            q=P;
            p=p->next;
}


  ____________________________
            return head;
            }
```

Choose the correct alternative to replace the blank line.

(a) q = NULL; $p \rightarrow$ next = head; head = p;
(b) $q \rightarrow$ next = NULL; head = p; $p \rightarrow$ next = head;
(c) head = p; $p \rightarrow$ next = q; $q \rightarrow$ next = NULL;
(d) $q \rightarrow$ next = NULL; $p \rightarrow$ next = head; head = p;

(GATE 2010: 2 Marks)

Solution: When the while loop ends, q contains address of second last node and p contains address of last node. So, we need to do the following things after while loop.

(i) Set next of q as NULL (q->next = NULL).
(ii) Set next of p as head (p->next = head).
(iii) Make head as p (head = p)

Step (ii) must be performed before step (iii). If we change head first, then we lose track of head node in the original linked list.

Ans. (d)

91. What does the following fragment of C program print?

```
char c [] = "GATE2011";
char *p = c;
printf("%s", p + p[3] - p[1]):
```

(a) GATE2011 (b) E2011
(c) 2011 (d) 011

(GATE 2011: 1 Mark)

Solution:
Let base address is 100
So, 100 + E − A = 104

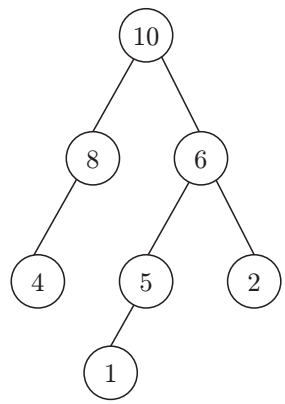
100	101	102	103	104	105	106	107
G	A	T	E	2	0	1	1

2011 will be printed.

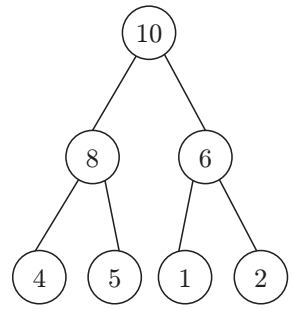
Ans. (c)

92. A max-heap is a heap where the value of each parent is greater than or equal to the value of its children. Which of the following is a max-heap?

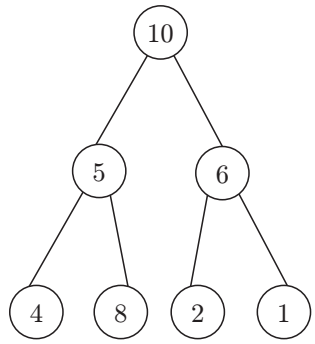
(a)



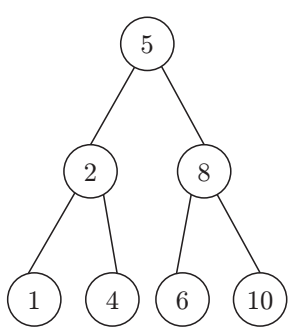
(b)



(c)



(d)



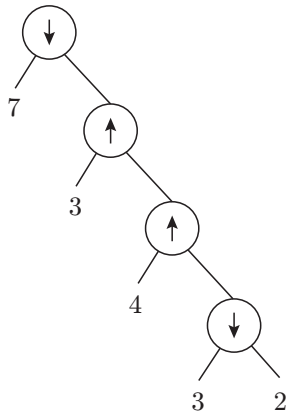
(GATE 2011: 1 Mark)

Solution: Heap is also complete binary tree.

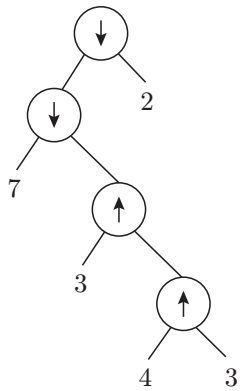
Ans. (b)

93. Consider two binary operators " and '↓' with the precedence of operator ↓ being lower than that of the operator . Operator is right associative while operator ↓ is left associative. Which one of the following represents the parse tree for expression (7↓343↓2)?

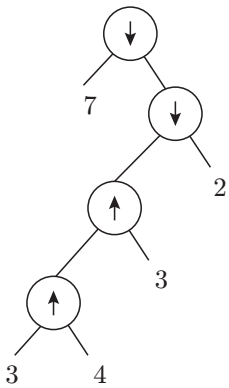
(a)



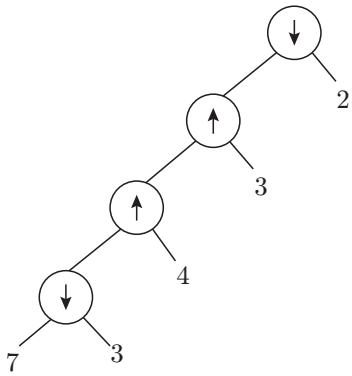
(b)



(c)



(d)



(GATE 2011: 2 Marks)

Solution:
Given expression is:
Precedence of ↑ is higher, so first the sub-expression (3↑ 4 ↑3) will be solved. In this 4 ↑ 3 will be solved first because ↑ is right to left associative. So, the expression will be solved as ((7 ↓(3↑(4 ↑3))). ↓ operator is left to right associative, so ((7 ↓(3↑(4 ↑3))) will be solved first. Build the in order expression considering operator precedence.

((7 ↓(3↑(4 ↑3)))↓2

Ans. (b)

Common Data Questions 94 and 95: Consider the following recursive C function that takes two arguments:

```
unsigned int foo (unsigned int n,
  unsigned int r) {
if (n>0) return ((n% r) + foo (n/r, r) );
else return 0;
}
```

94. What is the return value of the function foo when it is called foo(345, 10)?

(a) 345 (b) 12 (c) 5 (d) 3

(GATE 2011: 2 Marks)

Solution:

n = 345 and r = 10	345 % 10 + foo(34,10) = 5 + ….
n = 34, r = 10	5 + 4 + foo(3,10)
n = 3, r = 10	9 + 3 + foo (0,10)
n = 0, r = 10	12 + 0 = 12

Ans. (b)

95. What is the return value of the function foo when it is called foo(513, 2)?

(a) 9 (b) 8 (c) 5 (d) 2

(GATE 2011: 2 Marks)

Solution:

n = 513 and r = 2	513 % 2 + foo(256, 2) = 1 + ….
n = 256, r = 2	1 + 0+ foo(128,2)
n = 128, r = 2	1 + foo (64,2)
n = 64, r = 2	1 + foo(32,2)
….. n = 2, r = 2	…. 1 + foo(1,2)
n = 1, r = 2	1 + 1 + 0 = 2

Ans. (d)

96. What will be the output of the following C program segment?

```
char inChar = 'A' ;
switch(inChar) {
case 'A' : printf("Choice A\ n") ;
case 'B' :
case 'C' : printf("Choice B") ;
case 'D' :
case 'E' :
default :printf("No Choice") ; }
```

(a) No choice
(b) Choice A
(c) Choice A choice B no choice
(d) Program gives no output as it is erroneous

(GATE 2012: 1 Mark)

Solution: No break statement is available, it will execute all the cases.
Choice A
Choice B No Choice

Ans. (c)

97. Suppose a circular queue of capacity $(n - 1)$ elements is implemented with an array of n elements. Assume that the insertion and deletion operations are carried out using REAR and FRONT as array index variables, respectively. Initially, REAR = FRONT = 0. The conditions to detect queue full and queue empty are

(a) full: (REAR+1) mod n == FRONT
empty: REAR == FRONT

(b) full: (REAR+1) mod n == FRONT
empty: (FRONT+1) mod n == REAR

(c) full: REAR == FRONT
empty: (REAR+1) mod n == FRONT

(d) full: (FRONT+1) mod n == REAR
empty: REAR == FRONT

(GATE 2012: 2 Marks)

Solution:
Condition for queue full:
(Rear + 1) mod n = = front
Condition for queue free:
Rear = = front

Ans. (a)

98. The height of a tree is defined as the number of edges on the longest path in the tree. The function shown in the pseudocode below is invoked as height(root) to compute the height of a binary tree rooted at the tree pointer root.

```
int height (treeptr n)
{ if (n == NULL) return -1;
  if (n → left == NULL)
      if (n → right == NULL) return 0;

      else return [B1] ;          // Box 1

  else { h1 = height (n → left);
         if (n → right == NULL)
           return (1+h1);
         else { h2 = height (n → right);
             return [B2] ;      // Box 2
             }
       }
}
```

The appropriate expressions for the two boxes B1 and B2 are

(a) B1: (1 + height (n → right))
B2: (1 + max (h1, h2))

(b) B1: (height (n → right))
B2: (1 + max (h1,h2))

(c) B1: height (n → right)
B2: max (h1, h2)

(d) B1: (1 + height (n → right))
B2: max (h1, h2)

(GATE 2012: 2 Marks)

Solution: The above pseudocode is traversing tree to calculate the tree height.
Box 1: it should be replaced with the code that will traverse the right sub tree, so
B1: (1 + height(n → right))
Box2: Now both left and right sub tree height is calculated.
So the maximum out of both will be chosen.
B2: (1 + max(h1, h2)).

Ans. (a)

Common Data Questions 99 and 100: Consider the following C code segment:

```
int a, b, c = 0;
void prtFun(void);
main( )
{  staticint a = 1;  /*Line 1*/
prtFun( );
    a += 1;
prtFun( );
printf(" \n %d %d ", a, b);
}

voidprtFun(void)
{  staticint a = 2;  /*Line 2*/
   int b = 1;
       a += ++b;
       printf(" \n %d %d ", a, b);

}
```

99. What output will be generated by the given code segment?

(a)	(b)
3 1	4 2
4 1	6 1
4 2	6 1

(c)	(d)
4 2	3 1
6 2	5 2
2 0	5 2

(GATE 2012: 2 Marks)

Solution: A is static variable, its value will be preserved between different function calls.
First call to prtFun ( ): a = 2 and b = 1
a = 2 +2 = 4
Print 4 2
Second call to prtFun ( ): a = 4 and b = 1
a = 4 + 2 = 6
Print 6 2
Main printf function prints : 2 0

Ans. (c)

100. What output will be generated by the given code segment if:

Line 1 is replaced by autoint a = 1;
Line 2 is replaced by registerint a = 2;

(a)	(b)
3 1	4 2
4 1	6 1
4 2	6 1

(c)	(d)
4 2	4 2
6 2	4 2
2 0	2 0

(GATE 2012: 2 Marks)

Solution: Line 2 is replaced with register storage class, the values will not be preserved.
So it will print
4 2
4 2
2 0

Ans. (d)

101. Consider an undirected random graph of eight vertices. The probability that there is an edge between a pair of vertices is 1/2. What is the expected number of unordered cycles of length three?

(a) 1/8 (b) 1 (c) 7 (d) 8

(GATE 2013: 1 Mark)

Solution: P(e) = (1/2)
Number of ways vertices can be chosen of cycle three = ${}^{8}C_3 \times (1/2)^3 = 7$

Ans. (c)

102. Which of the following statements is/are **TRUE** for undirected graphs?

P: Number of odd degree vertices is even.
Q: Sum of degrees of all vertices is even.

(GATE 2013: 1 Mark)

(a) P only (b) Q only
(c) Both P and Q (d) Neither P nor Q

Solution: Both statements P and Q are correct.

Ans. (c)

103. What is the return value of $f(p, p)$, if the value of p is initialised to 5 before the call? Note that the first parameter is passed by reference whereas the second parameter is passed by value.

```
int f (int&x, int c) {
c = c - 1;
```

```
if(c==0) return 1;
x = x + 1;
return f(x,c) * x;
}
```

(a) 3024 (b) 6561
(c) 55440 (d) 161051

(GATE 2013: 2 Marks)

Solution: For p = 5,
f(6, 4)*6 = 9*8*7*6 = 3024

Ans. (b)

104. The preorder traversal sequence of a binary search tree is 30, 20, 10, 15, 25, 23, 39, 35, 42. Which one of the following is the postorder traversal sequence of the same tree?

(a) 10, 20, 15, 23, 25, 35, 42, 39, 30
(b) 15, 10, 25, 23, 20, 42, 35, 39, 30
(c) 15, 20, 10, 23, 25, 42, 35, 39, 30
(d) 15, 10, 23, 25, 20, 35, 42, 39, 30

(GATE 2013: 2 Marks)

Solution: Postorder traversal: Left Right Root 15, 10, 23, 25, 20, 35, 42, 39, 30 is the correct order.

Ans. (d)

105. Consider the following operation along with enqueue and dequeue operations on queues, where k is a global parameter.

```
MultiDequeue(Q){
m = k;
while (Q is not empty) and (m > 0) {
Dequeue(Q);
m = m - 1;
}
}
```

What is the worst case time complexity of a sequence of n queue operations on an initially empty queue?

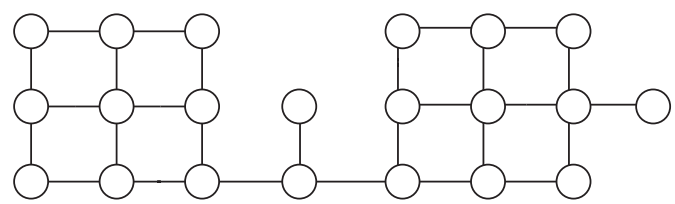
(a) $\Theta(n)$ (b) $\Theta(n + k)$
(c) $\Theta(n^k)$ (d) $\Theta(n^2)$

(GATE 2013: 2 Marks)

Solution: Any combination of n Enqueue and Dequeue operations will take Θ(n)

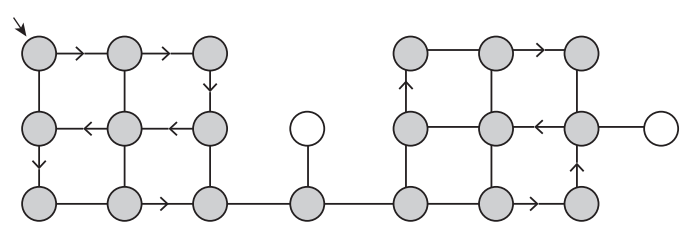
Ans. (a)

106. Suppose depth first search is executed on the graph below starting at some unknown vertex. Assume that a recursive call to visit a vertex is made only after first checking that the vertex has not been visited earlier. Then the maximum possible recursion depth (including the initial call) is _____.



(a) 17 (b) 18
(c) 19 (d) 20

Solution: By following the given path in graph, maximum possible depth is 19.



Ans. (c)

PRACTICE EXERCISES

Set 1

1. Predict the output of:

```
void main()
{
     int const*p=5;
     printf("%d",++(*p));
}
```

(a) 5 (b) 6
(c) Compile time error (d) Runtime error

2. Declaration of a pointer is given as (*ptr)[10]. Pick the correct answer.

(a) ptr is a pointer to an integer quantity.
(b) ptr is a pointer to a 10-element integer array.

(c) ptr is a 10-element array of pointers to integer quantities.
(d) ptr is a pointer to an 11-element integer array.

3.
```
struct a
{
    int a;
    char b;
    int c;
};
Union b
{
    char a;
    int b;
    int c;
};
```

Which of these is correct?

(a) Size of *a* is always different from size of *b*.
(b) Size of *a* is always same as size of *b*.
(c) *a* and *b* cannot be same because of non-homogeneity.
(d) It might result in runtime error.

4. `char a[5]="hello"` which of the following is correct?

(a) In array we cannot do the operation
(b) Size of *a* is too small
(c) Size of *a* is too large
(d) Runtime error

5.
```
void f(int i)
{
    int j;
    for(j=0; j<16; j++)
    {
        if(i & (0x8000>>j))
        printf("1");
        else
        printf("0");
    }
}
```

What is the purpose of this program?

(a) Its output is hex representation of i.
(b) Its output is octal representation of i.
(c) Its output is decimal representation of i.
(d) Its output is binary representation of i.

6 . What is the meaning of this pointer declaration: `char ptr(int(*a)[]);`

(a) Invalid statement
(b) ptr is a function that accepts an argument which is a pointer to an integer array and returns a character quantity.
(c) ptr is a function that accepts an argument which is an array of pointers to integers and returns a character of integer representation.
(d) Compile time error

7.
```
char func(intval)
{
    char ch=val;
    return ch;
}
void main()
{
    float a=115.76;
    printf("%d",func(a));
}
```

The output will be:

(a) 0 (b) 115.76
(c) 116 (d) Runtime error

8.
```
staticint func(intval)
{
    staticint sum;
    sum+=val;
    return sum;
}
void main()
{
    int i, n=9;
    for(i=1;i<n--;i++)
    func(i*2);
    printf("%d", func(0));
}
```

What is the output?

(a) 18 (b) 20
(c) 10 (d) 22

9. A variable *P* is called a pointer if

(a) *P* contains the address of an element in DATA.
(b) *P* points to the address of first element in DATA.
(c) *P* points to the address of last element in DATA.
(d) *P* contains the DATA and the address of DATA.

10. Size of (int) is

(a) always 2 bytes.
(b) depends on compiler being used.
(c) depends on computer hardware being used.
(d) always 32 bits.

11. Predict the output of:

```
main()
{
    char s[]="man";
    int i;
    for(i=0;s[i];i++)
    printf("\n%c%c%c%c",*(s+i),i[s]);
}
```

(a) Will execute successfully
(b) Incorrect syntax
(c) May execute on some compiler
(d) Will result in execution error

12. Global variables in different files are checked

(a) at load time.
(b) at assembly time.
(c) at linking time.
(d) during execution time.

13. Indirect addressing uses:

(a) Arrays (b) Pointers
(c) Structures (d) Array with pointers

14.
```
typedef char *ch;
const ch P;
```
In this code, what is P?

(a) P is a constant.
(b) P is a character constant.
(c) P is a character type.
(d) P is a pointer of character type.

15. Which of the following will have efficient space complexity and lesser time complexities?

(a) Incomplete binary tree
(b) Complete binary tree
(c) B+ tree
(d) Left-inclined binary tree

16. Binary tree can be represented as

(a) Linked list and graph. (b) Array and tree.
(c) Array and linked list. (d) Stack and queue.

17. The worst case height of AVL tree with n nodes is

(a) 1.44 $\log(n)$ (b) 1.44 $\log(n + 1)$
(c) 1.44 $\log(n + 2)$ (d) 1.44 $n\log n$

18. Which of the following sorting algorithm has almost the same worst case and best case complexity?

(a) Quick sort (b) Merge sort
(c) Heap sort (d) Shell sort

19. Which of the following is max-heap property if A is an array and i is the index of the array?

(a) A[Parent(i) $\geq$ A[i]] (b) A[A[i]>Parent(i)]
(c) A[Parent(i) $\leq$ A[i]] (d) A[A[i]<Parent(i)]

20. Complete binary tree can be implement by making use of

(a) array. (b) dequeue.
(c) priority queue. (d) stack.

21. If a binary search tree traversed in inorder then the number of nodes will be printed in

(a) ascending order. (b) descending order.
(c) random order. (d) none of these.

22. Which of the following is not an application of binary search trees?

(a) Finding a particular alphabet in a given string
(b) Sorting of characters in a string
(c) Faster searching as compared to linear search
(d) Minimising the length of a string

23. Which of the following is known as two-way list?

(a) Grounded header list
(b) Circular header list
(c) Linked list with header and trailer
(d) None of the above

24. Which of the following operators cannot be overloaded in C++?

(a) + (b) ?
(c) = = (d) : :

25. The correct way to round off a floating number x to an integer value is

(a) $y = (\text{int})\ (x + 0.5)$
(b) $y = \text{int}\ (x - 0.5)$
(c) $y = \text{int}\ (x + 0.5)$
(d) $y = (\text{int})\ ((\text{int})x + 0.5)$

26. Given two sorted list of size 'm' and 'n', respectively. The number of comparison needed in the worst case by the merge sort algorithm will be

(a) $m * \log n$ (b) min (m, n)
(c) avg (m, n) (d) $m + n - 1$

27. Consider a forest of t-trees having n vertices. The number of edges will be:

(a) $n\log t$ (b) $n - t$
(c) $n * t$ (d) $(n * t)/2$

28. Consider a binary search tree, _____ traversal will result in nodes arranged in ascending order?

(a) Inorder (b) Polish
(c) Reverse polish (d) Depth first search

29. The balance factor of height balance trees is:

(a) −1, 0, 1 (b) 1, 0, 1
(c) −1, 2, −1 (d) 1, 2, 1

30. The term 'push' and 'pop' is related to the

(a) array. (b) linked list.
(c) queue. (d) stack.

31. A digraph when represented in an adjacency matrix, the matrix is a

(a) weight matrix. (b) asymmetric matrix.
(c) symmetric matrix. (d) unit matrix.

32. The merge sort algorithm is based upon

(a) divide and conquer technique.
(b) 0/1 knapsack technique.
(c) optimisation technique.
(d) simulated annealing technique.

33. A linked list is a _____ kind of data structure.

(a) Static
(b) Dynamic
(c) Static as well as dynamic
(d) Neither dynamic nor static

34. When the stack is empty, the value of the top of stack(TOS) pointer is

(a) 1
(b) 0
(c) −1
(d) depends on the memory address

35. In an empty circular queue, the front and rear are?

(a) −1, −1 (b) 0, 0
(c) 0, 1 (d) 1, 1

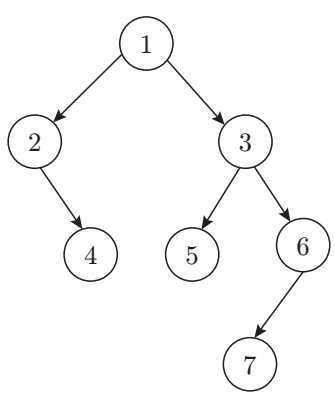
36. Which of the following data structures is non-linear type?

(a) Queue (b) Lists
(c) String (d) Graph

37. A full binary tree has a height of 3, then the number of nodes of such a tree is _____.

38. For the following tree, give the postorder traversal

(a) 2465371 (b) 2456371
(c) 4257631 (d) 7654321



39. Find the root in the given preorder traversal

ABDFGECH

(a) A (b) B
(c) H (d) G

40. If Inorder = *EDFBGAJHCI*, and Preorder = *ABDEFGCHJI,* the root is

(a) A (b) I
(c) J (d) B

41. A graph having n vertices and k components can have at most:

(a) $n + k/2$ edges
(b) $n*k/2$ edges
(c) $(n - k)(n - k + 1)/2$ edges
(d) $(n - k)(n - k + 2)/2$ edges

42. A graph in which all nodes have the same degree is called

(a) Euler graph. (b) Complete graph.
(c) Regular graph. (d) Colour graph.

43. The time complexity of hashing is

(a) $O(n)$ (b) $O(n \log (n - 1))$
(c) $O(n^3 - 1)$ (d) $O(n^2 \log (n - 1))$

44. The number of fields required in a doubly linked is _____.

(a) 7 (b) 2
(c) 3 (d) 1

45. The runtime complexity of building max-heap is

(a) $O(n)$ (b) $O(\log n)$
(c) $O(n^{n-1})$ (d) $O(n^2 \log n/2)$

46. A binary tree is of level 7, the maximum number of nodes can be

(a) 129 (b) 128
(c) 255 (d) 256

Set 2

1. What does the following code do?

```
Var a, b, c; integer
begin
a: = a+b;
b: =a-b;
c: =a-b
end
```

(a) Exchanges (a) and (b)
(b) Doubles (a) and stores in (b)
(c) Doubles (b) and stores in (a)
(d) Leaves (a) and (b) unchanged

2. FORTRAN implementation do not permit recursion because

(a) They use static allocation for variables
(b) They use dynamic allocation for variables
(c) Stacks are not available on all machines
(d) It is not possible to implement recursion on all machines

3. An unrestricted use of the 'goto' statement is harmful because

(a) It makes it more difficult to verify programs.
(b) It increases the running time of the programs.
(c) It increases the memory required for the programs.
(d) It results in the compiler generating longer machine code.

4. What is the value of X printed by the following program?

```
Program COMPUTE (input, output);
Var X: integer
Procedure FIND (X: real)
      Begin X: sqrt(X); end
begin
                X:=2
     Find (X)
Writeln(X)
end
```

(a) 2 (b) 4
(c) Runtime error (d) None of the above

5. Consider the following C function definition

```
int  Trial (int a, int b, int c)
{
       if ((a>=b)&& (c <b) return b;
       else if (a >=b)
         return Trial (a, c, b);
       else return Trial (b, a, c);
}
```

The function trial:

(a) Finds the maximum of a, b and c.
(b) Finds the minimum of a, b and c.
(c) Finds the middle number of a, b and c.
(d) None of these.

6. The following C declarations

```
  struct node {
  int i;
  float j;
};
  struct node *s[10];
```

Define s to be

(a) An array, each element of which is a pointer to a structure of type node.
(b) A structure of 2 fields, each field being a pointer to an array of 10 elements.
(c) A structure of 3 fields: an integer, a float and an array of 10 elements.
(d) An array, each element of which is a structure of type node.

7. Aliasing in the context of programming language refers to

(a) multiple variables having the same memory location.
(b) multiple variables having the same value.
(c) multiple variables having the same identifier.
(d) multiple uses of the same variable.

8. Consider the following three C functions:

```
[P1] int* g(void)
       {
int x = 10;
return (&x);
        }
[P2] int* px;
{
int *px;
  *px= 10
return  px;
}
[P3] int* g(void)
{
int *px;
px = (int*)malloc(size of (int));
 *px =10;
return  px;
}
```

Which of the above three functions is likely to cause problems with pointers?

(a) Only P_3 (b) Only P_1 and P_3
(c) Only P_1 and P_2 (d) P_1, P_2 and P_3

9. The value of j at the end of the execution of the following C program

```
int incr (int i)
{
     staticint count = 0;
     count = count + i
     return (count);
}
main (){
     int  i, j;
     for (i=0; i<=4; i++)
     j =incr(i)
              }
```

is

(a) 10 (b) 4
(c) 6 (d) 7

10. What is printed by the print statements in the program P_1, assuming call-by-reference parameter passing?

```
Program P1()
{
     x = 10;
     y = 3;
     func1 (y, x, x);
     print x;
     print y;
}
Func1(x, y, z)
{
     y = y+4;
     z = x + y +z
}
```

(a) 10, 3 (b) 27, 7
(c) 31, 3 (d) None of these

11. Let A be a two-dimensional array declared as follows:

A: array [1…10] [1..15] of integer
Assuming that each integer takes one memory location, the array is stored in row-major order and the first element of the array is stored at location 100. What is the address of the element $A[i][j]$?

(a) $15i + j + 84$ (b) $15j + i + 84$
(c) $10i + j + 89$ (d) $10j + i + 89$

12. Suppose you are given an array $s[1 \ldots n]$ and a procedure reverse (s, i, j) which reverses the order of elements in between positions i and j (both inclusive). What does the following sequence do, where $1 \leq k \leq n$

reverse $(s, 1, k)$;
reverse $(s, k + 1, n)$;
reverse $(s, 1, n)$

(a) Rotates s left by k position
(b) Leaves s unchanged
(c) Reverses all elements of s
(d) None of the above

13. Which of the following permutations can be obtained in the output (in the same order) using a stack assuming that the input is the sequence 1, 2, 3, 4, 5 in that order?

(a) 3, 4, 5, 1, 2 (b) 3, 4, 5, 2, 1
(c) 1, 5, 2, 3, 4 (d) 5, 4, 3, 1, 2

14. Which of the following is essential for converting an infix expression to the post-fix form efficiently?

(a) An operator stack
(b) An operand stack
(c) An operand stack and an operator stack
(d) A parse tree

15. A priority queue Q is used to implement a stack that stores characters. PUSH(C) is implemented INSERT(Q, C, K) where K is an appropriate integer key chosen by the implementation. POP is implemented as DELETEMIN(Q). For a sequence of operations, the keys chosen are in

(a) non-increasing order.
(b) non-decreasing order.
(c) strictly increasing order.
(d) strictly decreasing order.

16. What value would the following function return for the input $x = 95$?

```
Function fun (x: integer): integer;
Begin
    if x > 100 then fun = x - 10
    else fun: = : fun (fun(x+11))
End
```

(a) 89 (b) 90
(c) 91 (d) 92

17. In a circular-linked list organisation, insertion of a record involves modification of

(a) one pointer. (b) two pointer.
(c) multiple pointer. (d) no pointer.

18. Linked lists are not suitable data structures of which one of the following problems?

(a) Insertion sort
(b) Binary search
(c) Radix sort
(d) Polynomial manipulation

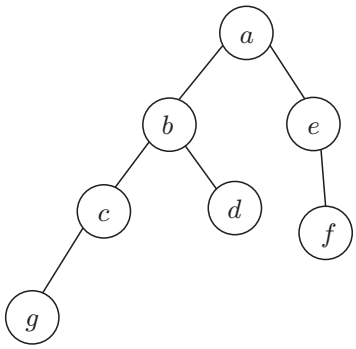
19. In the worst case, the number of comparisons needed to search singly linked list of length n for a given element is

(a) $\log_2 n$ (b) $n/2$
(c) $\log_2 n - 1$ (d) n

20. A binary tree T has n leaf nodes. The number of nodes of degree 2 in T is

(a) $\log_2 n$ (b) $n - 1$
(c) n (d) 2^n

21. In the balanced binary tree in the figure given below, how many nodes will become unbalanced when a node is inserted as a child of the node 'g'?



(a) 1 (b) 3
(c) 7 (d) 8

22. A binary search tree is generated by inserting in order of the following integers: 50, 15, 62, 5, 20, 58, 91, 3, 8, 37, 60, 24. The number of nodes in the left sub-tree and right sub-tree of the root, respectively, are

(a) (4, 7) (b) (7, 4)
(c) (8, 3) (d) (3, 8)

23. A binary search tree is used to locate the number 43. Which one of the following sequences is not possible?

(a) 61 52 14 17 40 43
(b) 2 3 50 40 60 43
(c) 10 65 31 48 37 43
(d) 81 61 52 14 41 43

24. A binary search tree contains the values 1, 2, 3, 4, 5, 6, 7, 8. The tree is traversed in preorder and the values are printed out. Which of the following sequences is a valid output?

(a) 53124786 (b) 53126487
(c) 53241678 (d) 53124768

25. Which of the following statements is false?

(a) A tree with n nodes has $(n - 1)$ edges.
(b) A labelled rooted binary tree can be uniquely constructed given its postorder and preorder traversal results.
(c) A complete binary tree with n internal nodes has $(n + 1)$ leaves.
(d) The maximum number of nodes in a binary tree of height h is $(2^{h+1} - 1)$

26. A complete n-ary tree is one in which every node has 0 or n sons. If x is the number of internal nodes of a complete n-ary tree, the number of leaves in it is given by

(a) $x(n - 1) + 1$ (b) $xn - 1$
(c) $xn + 1$ (d) $x(n + 1)$

27. Let LASTPOST, LASTIN and LASTPRE denote the latest vertex visited in a postorder, inorder and preorder traversal, respectively, of a complete binary tree. Which of the following is always true?

(a) LASTIN = LASTPOST
(b) LASTIN = LASTPRE
(c) LASTPRE = LASTPOST
(d) None of these

28. The number of leaf nodes in a rooted tree of n nodes with each node having 0 to 4 children is

(a) $n/2$ (b) $(2n - 1)/4$
(c) $(n + 1)/4$ (d) $(2n + 1)/4$

29. Consider the following program:

```
#define funct(x) x*x+x
int main()
{
     int x;
     x=23+funct(4)*funct(3);
     printf("%d",x);
     return 0;
}
```

What will be the output of the above program?

(a) 126 (b) 196
(c) 149 (d) 132

Output of the following program will be

```
int main
            {
int i, a[6]= 000, 001, 010, 011, 100,
  101, 110, 111;
for i=0;i<6;i++
printf"%d", a[i];
                return0;
             }
```

(a) 0, 1, 2, 3, 4, 5
(b) 0, 1, 10, 11, 100, 101
(c) 0, 1, 8, 9, 100, 101
(d) None of these

30. The minimum and maximum number of keys in the internal nodes of B tree with order 4 is, respectively

(a) 1, 3 (b) 2, 4
(c) 1, 4 (d) 2, 3

31. Consider the following code segment.

```
void foo(int x, int y)
{
x+=y;
y+=x;
}
main()
{
int x=5.5;
foo(x,x);
}
```

What is the final value of x in both call by value and call by reference, respectively?

(a) 5 and 16 (b) 5 and 12
(c) 5 and 20 (d) 12 and 20

32. What is the maximum height of any AVL tree with 7 nodes? Assume that the height of a tree with a single node is 0.

ANSWERS TO PRACTICE EXERCISES

Set 1

1. (c)	**11.** (a)	**21.** (a)	**31.** (b)	**41.** (d)
2. (b)	**12.** (c)	**22.** (d)	**32.** (a)	**42.** (c)
3. (a)	**13.** (b)	**23.** (d)	**33.** (b)	**43.** (a)
4. (b)	**14.** (a)	**24.** (d)	**34.** (c)	**44.** (c)
5. (a)	**15.** (b)	**25.** (a)	**35.** (a)	**45.** (b)
6. (b)	**16.** (c)	**26.** (d)	**36.** (d)	**46.** (b)
7. (a)	**17.** (c)	**27.** (b)	**37.** (3)	
8. (b)	**18.** (c)	**28.** (a)	**38.** (c)	
9. (a)	**19.** (a)	**29.** (a)	**39.** (a)	
10. (b)	**20.** (a)	**30.** (d)	**40.** (a)	

Set 2

1. (a) The given program is for swapping values of two variables.

2. (a) In recursion, stack is used to store the latest value of variables, means value of variables change with the recursive call. But FORTRAN uses static allocation for variables which does not allow changes in every different function call.

3. (a) A goto statement does not increase the running time or memory requirement of the program. It also does not contribute to longer machine code. But the use of goto can result in unstructured code with multiple entry and exit points which can cause an issue for program verification.

4. (b) Global variable value not changed because of static scoping.

5. (d) Just take $a = 21$, $b = 8$ and $c = 14$ and verify the options.

6. (a) S is an array, each element of which is a pointer to a structure of type node.

7. (a) Aliasing describes a situation in which a data location in memory can be accessed through different symbolic names in the program.

8. (c) In P_1, there is no parameter passing and type casting so it is wrong.

In P_2, pointer is defined as pointer function which is wrong.

9. (a) Count is declared as static variable in incr( ). So, statement static int count = 0 will assign count to 0 only in the first incr call. Remaining calls to incr function will use old values of count.

at first incr(0) call count will become 0
after incr(1) call count will become 1
after incr(2) call count will become 3
after incr(3) call count will become 6
after incr(4) call count will become 10

10. (c) Initial value of x and y will be 10, 3 and address are 1000, 2000, respectively. Then after first statement execution in the function, value at address 1000 will be {(10 as old value) + (4) = 14} because y at the called function refers to address of x. After second statement execution in the function, value at address 1000 will be $\{3 + 14 + 14 = 31\}$ because 'x' at called function refers to y in the calling function and 'z' at the called function refers to address of x. So final values will be

x(address = 1000) = 31

y(address = 2000) = 3

11. (a) To find location of an element in a two-dimensional array by row-major order is:

$A[i][j]$ = Base address + $(i - 1)*r + (j - 1)$, where r = no. of rows.

Given base address = 100 and number of rows = 15

$$\begin{aligned} &= 100 + (i - 1) \times 15 + (j - 1) \\ &= 100 + 15i - 15 + j - 1 \\ &= 100 + 15i + j - 16 \\ &= 84 + 5i + j \end{aligned}$$

11. (a) Let an array $S[1 \ldots 8]$

1	2	3	4	5	6	7	8

$S = 7$, $k = 3$

reverse (S, 1, 3) we get [3, 2, 1, 4, 5, 6, 7, 8]

reverse (S, 4, 8) we get [3, 2, 1, 8, 7, 6, 5, 4]

reverse (S, 1, 8) we get [4, 5, 6, 7, 8, 1, 2, 3]

after performing above three reverse, we can see it is equivalent to the option (a).

12. (b) Take a stack and perform push and pop operation to verify the given options.

13. (a) During the conversion of infix to postfix expression, operands do not change their position. So, only operator stack is needed.

14. (d) Stack works on concept of first-in last-out. But according to the question, it should work like queue means first-in first-out. For that, we give the higher priority to the first element. So, all elements should form strictly decreasing order.

15. (c) fun (98)

fun fun(109)

fun (99)

fun (fun (200))

fun (100)

fun (fun(111))

fun (101)

fun = 101−10 = 91

Return value of fun is 91. So, option (c) is correct.

16. (b) If a record is inserted in the circular-linked list then two pointers have to be modified. Let a record k be inserted between the node m and n, then

(a) Address field of node m contains address of node x.

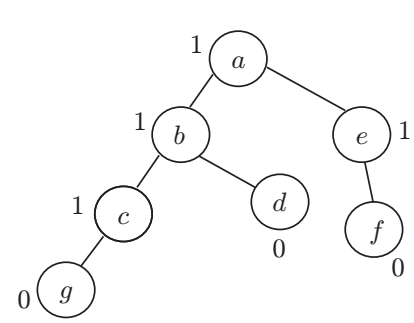
(b) Address field of node x contains address of node n.

17. (b) With linked list binary search will take $O(n)$ time to sort the element list. Thus, binary search performs worst with linked list.

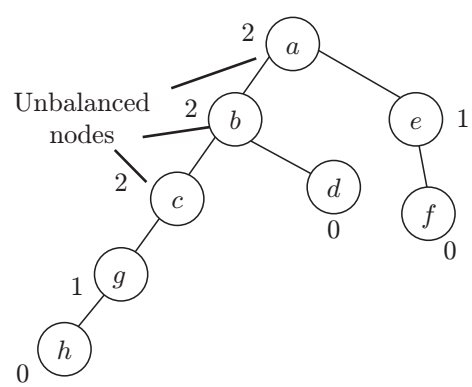
18. (d) On linked list we can perform only linear search. The number of comparisons required using linear search is $O(n)$.

19. (b) In a binary tree, two nodes are connected by a single node to having degree 2. If a tree has n leaf nodes, then the number of nodes with degree 2 will always be less than n. So option (b) is the strong answer.

20. (b)

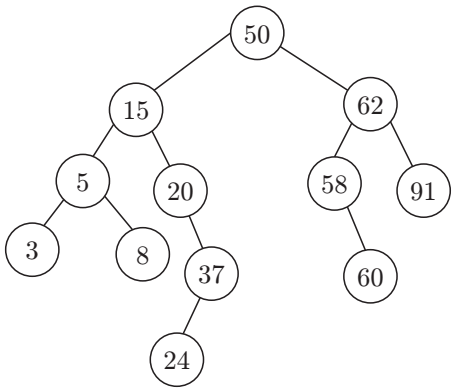


After a node h is added as a child of g



Nodes a, b and c become unbalanced.

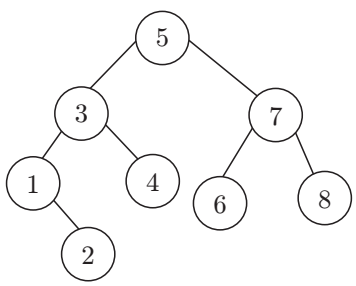
21. (b) The binary search tree is as follows:



In the above binary tree, number of nodes in the left sub-tree of root is 7 and in the right sub-tree of the root is 4.

22. (b) Not possible
(b) 2 3 50 40 0 43

23. (d) Inorder sequence: 53124768



24. (b) A labelled rooted binary tree cannot be constructed uniquely when inorder traversal is given along with postorder or preorder traversal.

25. (a) Total number of nodes in n-ary tree $= ni + 1$, where i is internal nodes _____ 1

Total nodes = internal nodes (i) + leaf nodes (l) _____ 2

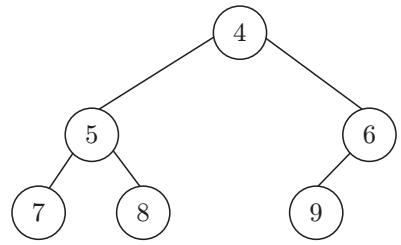
From 1 and 2

$i + l = ni + 1$
$l = (n - 1)I + 1$

Replace I with x, because in question internal node is given as x.
So number of leaf nodes in n-ary tree with x internal nodes is $(n - 1)x + 1$.

26. (d) In the question, it is given that the tree is a complete binary tree. For a complete binary tree, the last visited node will always be the same for inorder traversal and preorder traversal because in both the traversal right child is traversed in the last.

In postorder traversal, root node is traversed in the end. So we can remove option (a) and (c).
Let us take an example for option (b).



Inorder: 7 5 8 4 9 6
Preorder: 4 5 7 8 6 9
When a tree is a complete binary tree with full at last level, then only last-visited node will always be same for inorder traversal and preorder traversal.

Ans. (d)

27. (d) Total number of nodes in n-ary tree $= n$(internal nodes) + 1

Given $n = 4i + 1$
Total nodes = internal nodes (i) + leaf nodes (l)
$n = i + l$
$i = n - l$
$n = 4(n - l) + 1$
$n = 4n - 4l + 1$
$4l = (2n + 1)$
$l = (2n + 1)/4$

28. (c) According to program,

$F(n) = n + 2F(n - 1)$ for $n \geq 2$
$= 0$ for $n = 1$
$F(4) = 4 + 2(3 + 2(2 + 2(0)))) = 18$
$F(3) = 3 + 2(2 + 2(0)))) = 7$

Ans. (c)

29. (c) 0, 1, 8, 9, 100, 101, 110 is an octal number format.

30. (b) B tree of order k, every internal node must have $\lfloor k/2 \rfloor$ to k children and number of keys will be one less than the children.

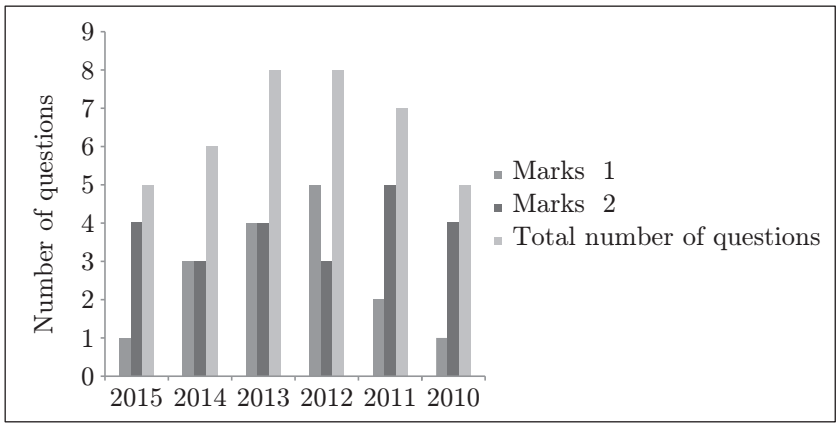
31. (c) In call by value, there is no change in the value of x because function foo( ) operates on copy of x.

In call by reference, first value of x changes to $5 + 5$ and then it changes to $10 + 10 = 20$.

32. (3) Draw AVL tree to find the height.

UNIT IV: ALGORITHMS

LAST SIX YEARS' GATE ANALYSIS



Concepts on which questions were asked in the previous six years

Year	Concept
2015	Algorithms, Unsorted array, BFS, Spanning tree, Recurrence relation, Job sequence, Planar graph, Hash table, Complexity, Asymptotic notations, Polynomial time reducible, SAT Problem, Connected graphs, Algorithm design paradigms, Graphs
2014	Arithmetic expressions, Rooted tree, Graph, Depth-first search, Sorting algorithms and their complexity, Balanced binary, NP-complete problem, Hashing
2013	Sorting algorithms, Time complexity, NP problems, Bellman–Ford algorithm, Graphs, Heap sort, queue
2012	Time complexity, Recurrence relation, Balanced binary search tree, NP problem, Dijkstra's algorithm, Merge sort
2011	Arrays, Binary heap, Dynamic programming, Time complexity, Hashing, Minimum spanning tree (MST)
2010	Undirected graphs, Spanning tree, Hashing

CHAPTER 4

ALGORITHMS

Syllabus: Algorithms: Analysis, Asymptotic notation, Notions of space and time complexity, Worst- and average-case analysis; Design: Greedy approach, Dynamic programming, Divide-and-conquer; Tree and graph traversals, Connected components, Spanning trees, Shortest paths; Hashing, Sorting, Searching. Asymptotic analysis (best, worst, average cases) of time and space, Upper and lower bounds, Basic concepts of complexity classes – P, NP, NP-hard, NP-complete.

4.1 INTRODUCTION

In computer science, an algorithm provides a set of step-by-step instructions for solving various problems. The set of instructions should be unambiguous, finite and provide a clear termination point. An algorithm accepts some input values and produces outputs. The algorithm can be built in simple English language or in programming language. This subject helps in solving and analysing complex problems through logical thinking.

4.2 ALGORITHM

An algorithm is a well-defined procedure which takes a set of values as an input and produces a set of values as the output.

4.2.1 Properties of Algorithm

There are certain properties algorithms have:

1. An algorithm has zero or more inputs.
2. It should produce at least one output.
3. It should terminate within the finite time.
4. It should be deterministic (ambiguous).
5. It is generic to all programming languages.

4.2.2 Steps to Solve a Problem

1. Define the problem, which means what is input and what should be the output.
2. Identify the constraints which are to be satisfied.
3. Design the algorithm for a given problem.
4. Draw the flowchart.
5. Verify the process.
6. Implement the coding.
7. Perform time and space analysis.

4.2.3 Algorithm Analysis

If a problem has more than one solution, then the technique used to decide which solution is best is known as algorithm analysis.

4.2.3.1 A Posteriori and A Priori Analysis

Analysis is done on two parameters: time complexity and space complexity. Time complexity means time taken by an algorithm to solve a problem. Space complexity means memory space required by an algorithm to solve a problem. There are two types of analysis:

1. ***A Posteriori* Analysis:** It is a software- and hardware-dependent analysis. It keeps on changing from one system to other system. This type of analysis tells the exact time and space complexity, meaning how much an algorithm takes time and space for solving a problem on the given system.
2. ***A Priori* Analysis:** It is a software- and hardware-independent analysis. It is constant for all the systems. This type of analysis tells the approximate time and space complexity. It is the determination of the order of magnitude of a statement, meaning how many times a statement is executing.

4.2.3.2 Asymptotic Analysis

Suppose there are given two algorithms for a task, how do we find out which one is better?

This can be done by actually implementing both the algorithms and running them on the same computer with different input values. Then the comparison is made in terms of time. Algorithm that takes less time is considered better. But there are so many problems with this approach.

1. It might be possible that for certain inputs first algorithms performs better than second and for other set of inputs second algorithm performs better. So,the decision of choosing best among them becomes difficult.
2. With the change in operating machine, performance of algorithm varies. On one machine first algorithm can work better and on another machine, the second works better. This is another problem associated with this approach.

Above issues can be handled using asymptotic analysis techniques for analysing algorithms. In asymptotic analysis, analysis of algorithm in done in terms of input size. Actual running time is not computed, variation of time with input size is calculated.

Now, understand that how the issues with above technique are resolved by asymptotic analysis technique using the below example:

Example 4.1

Let us consider an example of a search problem (searching a given item) in a sorted array. One way to search is linear search (the order of growth is linear) and the other way is binary search. Suppose we have two machines with different processing speeds. We run linear search on the fast machine and binary search on slow machine. For small input size, the linear search will perform better. But as we increase the size of input, binary search will perform better. The reason behind this is,the order of growth of binary search is logarithmic and that of linear search is linear. So, by using this technique the issues of machine dependency are resolved.

4.2.4 Asymptotic Notation

Asymptotic notation is based on two assumptions, which hold in most of the cases and have their importance. It is very much important to understand the assumptions and limitations of asymptotic notations:

1. **Input size:** It is interesting to know how the running time of an algorithm grows for a large input size n.
2. **Constant:** The running time of an algorithm also depends on various constants. But for a large input n, the constant factors would be ignored.

For representing best-case, average-case, and worst-case complexity, there are three different notations as given in the following sub-sections.

4.2.4.1 Theta Notation (Θ)

Theta notation shows the tightest upper bound and the tightest lower bound to the given function (Fig. 4.1). It is used to define the average-case running time. For a given function $f(n)$, we denote $\Theta(g(n))$ as $\Theta(g(n)) = \{f(n)$: there exists positive constants $c_1 > 0$, $c_2 > 0$ and $n_0 > 0$ such that $0 < c_1(g(n)) \leq f(n) \leq c_2(g(n))\ \forall\ n \geq n_0\}$

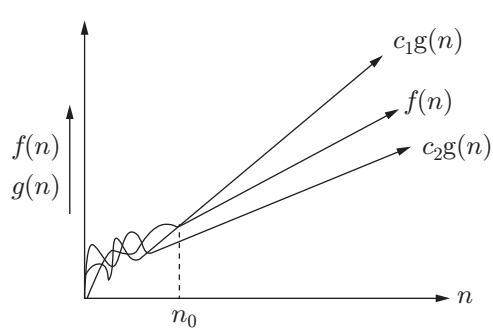


Figure 4.1 Theta notation (Θ)

Problem 4.1: Consider the following functions:

$$f(n) = n$$

$$g(n) = \frac{n}{8}$$

For $f(n) = \Theta(g(n))$, find the values of c_1, c_2 and n_0.

Solution: From the definition of theta (Θ) notation, we have

$$0 < c_1(g(n)) \leq f(n) \leq c_2(g(n)) \quad \forall n \geq n_0$$

By putting $c_1 = 1, 2, 3$ to satisfy the above condition, we get

$$c_1(g(n)) \leq f(n)$$

$$1\left(\frac{n}{8}\right) \leq n$$

$$c_1 = 1$$

By putting $c_2 = 1, 2, 3$ to satisfy the above condition, we get

$$f(n) \leq c_2(g(n))$$

$$8\left(\frac{n}{8}\right) \leq n$$

$$c_2 = 8$$

By putting $n_0 = 1, 2, 3, \ldots$, to satisfy the above condition, we get $n_0 = 1$.

4.2.4.2 Big-O Notation (O)

Big-O notation shows the upper bound to the given function (Fig. 4.2). It is used to define the worst-case running time. For a given function $f(n)$, we denote $O(g(n))$ as $O(g(n)) = \{f(n)$: there exist positive constants $c > 0$ and $n_0 > 0$ such that $0 < f(n) \leq c(g(n)) \forall n \geq n_0\}$

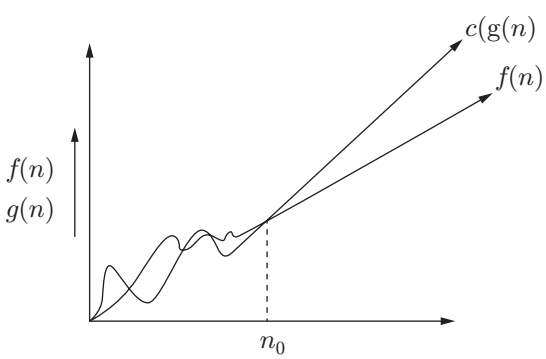


Figure 4.2 | Big-O notation (O).

Problem 4.2: Consider the following functions:

$$f(n) = n$$

$$g(n) = n$$

For $f(n) = O(g(n))$, find the values of c and n_0.

Solution: From the definition of Big-O notation, we have

$$f(n) \leq c(g(n)) \quad \forall n \geq n_0$$

By putting $c = 1$ and $n = 1$, we get

$$n \leq c \cdot n \ \forall \ n \geq n_0$$

The above condition is satisfied. So, $c = 1$ and $n_0 = 1$.

4.2.4.3 Omega Notation (Ω)

Omega notation shows the lower bound to the given function (Fig. 4.3). It is used to define the best-case running time. For a given function $f(n)$, we denote $\Omega(g(n))$ as $\Omega(g(n)) = \{f(n)$: there exist positive constants $c > 0$ and $n_0 > 0$ such that $0 < c(g(n)) \leq f(n) \ \forall \ n \geq n_0\}$.

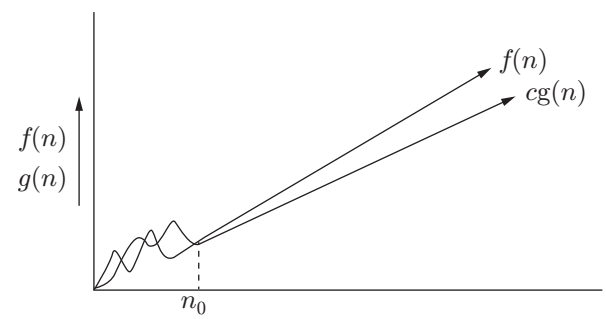


Figure 4.3 | Omega notation (Ω).

Problem 4.3: Consider the following functions:

$$f(n) = n^2$$

$$g(n) = n$$

For $f(n) = \Omega(g(n))$, find values of c and n_0.

Solution: From the definition of Omega notation, we have

$$c(g(n)) \leq f(n) \ \forall \ n \geq n_0$$

By putting $c = 1$ and $n = 1$, we get

$$c \cdot n \leq n^2 \ \forall \ n \geq n_0$$

The above condition is satisfied. So, $c = 1$ and $n_0 = 1$.

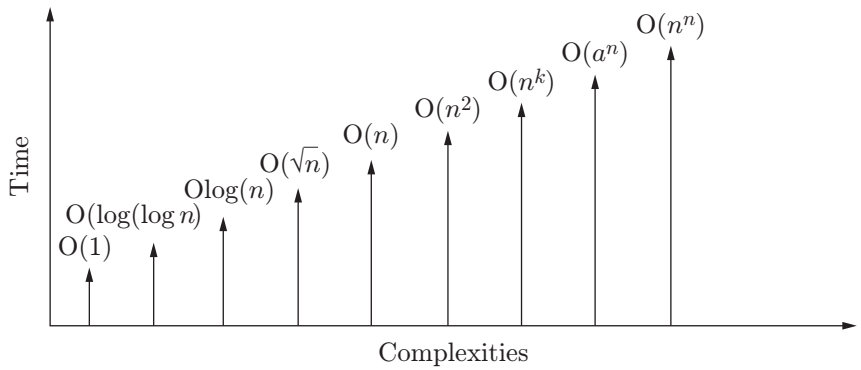


Figure 4.4 | Time complexities in increasing order.

4.2.4.4 Small-o Notation (o)

Small-o represents an upper bound which is not a tight upper bound. It is also used to define the worst-case running time. For a given function $f(n)$, we denote $o(g(n))$ as $o(g(n)) = \{f(n)$: there exist positive constants $c > 0$ and $n_0 > 0$ such that $0 < f(n) < c(g(n))\ \forall\ n \geq n_0\}$.

4.2.4.5 Small Omega Notation (ω)

Small omega (ω) represents a lower bound which is not a tight lower bound. It is used to define the best-case running time. For a given function $f(n)$, we denote $\omega(g(n))$ as $\omega(g(n)) = \{f(n)$: there exist positive constants $c > 0$ and $n_0 > 0$ such that $0 < c(g(n)) < f(n)\ \forall\ n \geq n_0\}$.

4.2.4.6 Properties of Asymptotic Notations

1. **Transitivity:** All asymptotic notations show transitivity.

   If $f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n))$, then $f(n) = \Theta(h(n))$.
   If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then $f(n) = O(h(n))$.
   If $f(n) = \Omega(g(n))$ and $g(n) = \Omega(h(n))$, then $f(n) = \Omega(h(n))$.
   If $f(n) = o(g(n))$ and $g(n) = o(h(n))$, then $f(n) = o(h(n))$.
   If $f(n) = \omega(g(n))$ and $g(n) = \omega(h(n))$, then $f(n) = \omega(h(n))$.

2. **Reflexivity:** Only Theta, Big-O, and Omega notations show reflexivity. Small-o and small-omega do not show reflexive property.

   $f(n) = \Theta(f(n))$ $\quad f(n) = O(f(n))$ $\quad f(n) = \Omega(f(n))$

3. **Symmetric:** Only Theta notation is symmetric.

   $f(n) = \Theta(g(n))$ if and only if $g(n) = \Theta(f(n))$.

4. **Transpose symmetry:**

   $f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$
   $f(n) = o(g(n))$ if and only if $g(n) = \omega(f(n))$

4.2.4.7 Types of Complexities

1. Constant time complexity: $O(1)$
2. Logarithmic time complexity: $O(\log(\log n))$, $O(\sqrt{\log n})$ and $O(\log n)$
3. Linear time complexity: $O(\sqrt{n})$ and $O(n)$
4. Polynomial time complexity: $O(n^k)$, where k is a constant and is >1
5. Exponential time complexity: $O(a^n)$, where $a > 1$

The complexities in increasing order are represented in Fig. 4.4.

4.2.5 Recurrence Relations

Recurrence relations are recursive definitions of mathematical functions or sequences. For example, the recurrence relation defines the famous Fibonacci sequence 1, 1, 2, 3, 5, 8, 13, …:

$$f(n) = f(n-1) + f(n-2)$$
$$f(1) = 1$$
$$f(0) = 1$$

Given a function defined by a recurrence relation, we want to find a 'closed form' of the function. In other words, we would like to eliminate recursion from the function definition. There are several techniques for solving recurrence relations, and these are discussed in the following sub-sections.

4.2.5.1 Iteration Method

This method is accurate but involves a lot of algebraic expressions which are difficult to keep track of. It can get very challenging for further complicated recurrence relations.

Problem 4.4: We have $T(n) = 1 \quad \text{if } n = 1$

$$T(n) = 4T\left(\frac{n}{2}\right) + n \quad \text{if } n > 1$$

$$T(n) = 4T\left(\frac{n}{2}\right) + n = 4\left(4T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

$$= 4\left(4\left(4T\left(\frac{n}{8}\right)+\frac{n}{4}\right)+\frac{n}{2}\right)+n$$

$$= 64T\left(\frac{n}{8}\right)+4n+2n+n$$

$$= n+2n+4n+64T\left(\frac{n}{8}\right)$$

$$= n+2n+4n+\cdots+2^j n+\cdots+4^i T\left(\frac{n}{2^i}\right)$$

Find the time complexity when $(n/2^i) = 1$.

Solution:

We have $i = \log n$. So, the time complexity of n can be represented as

$$T(n) = n+2n+4n+8n+2^i n+4^{\log n}\mathrm{O}(1)$$

We get $$n\left(\sum_{i=0}^{\log(n-1)} 2^i\right)+4^{\log n}\mathrm{O}(1)$$

We Know that $$\sum_{k=0}^{m} x^k = \frac{x^{m+1}-1}{x-1}$$

So, $$n\left(\frac{2^{\log(n-1)+1}-1}{2-1}\right)+4^{\log n}\mathrm{O}(1)$$

Solving, we get

$$n2^{\log n}-n+4^{\log n}\mathrm{O}(1) \Rightarrow n^2-n+n^{\log 4}\mathrm{O}(1)$$

$$\Rightarrow 2n^2\mathrm{O}(1)-n \Rightarrow \mathrm{O}(n^2)$$

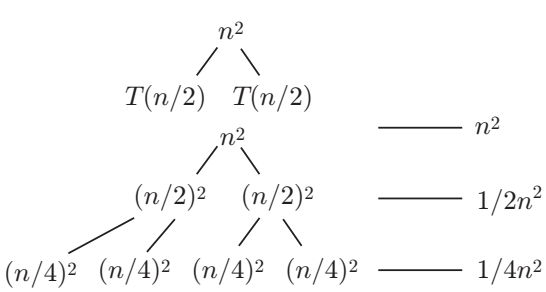
So, the running time complexity is given as: $T(n) = \mathrm{O}(n^2)$.

4.2.5.2 Recursion Tree

A recursion tree is useful for visualizing what happens when a recurrence is iterated. It represents a tree of recursive calls and the amount of work done at each call is shown with the help of a diagram.

Example 4.2

Consider the recurrence $T(n) = 2T(n/2)+n^2$.



How deep does the tree go?

We stop at the leaf, and we know we are at a leaf when we have a problem of size 1:

$$1 = \left(\frac{n}{2^i}\right)^2 \Rightarrow n^2 = 2^{2i} \Rightarrow n = 2^i \Rightarrow i = \log n$$

The amount of work done is given as

$$\sum_{i=0}^{\log n} \frac{n^i}{2} = \Theta n^2$$

This is geometrically decreasing in size, so it would not get any bigger than n^2.

4.2.5.3 Master Theorem

Master theorem provides a method to solve various recurrence relations. Although all the relations can't be solved using this method, but it is one of the useful method of solving complex relations of the form:

$$T(n) = aT\left(\frac{n}{b}\right)+f(n)$$

where a and b are constants. *f is a function of n. this recursive algorithm breaks up the input into sub-problems of size n/b each.*

Three cases of master theorem are as follows:

1. **Case 1:** If $f(n)$ is $\mathrm{O}(n^{\log_b a-\varepsilon})$ then $T(n)$ is $\Theta(n^{\log_b a})$. The rate of growth at leaves is faster than f, more work is done on leaves.
2. **Case 2:** If $f(n)$ is $\Theta(n^{\log_b a})$ then $T(n)$ as $\Theta(n^{\log_b a}\log n)$. The rate of growth of leaves is same as f. So, the same amount of work is done at every level of tree. The leaves grow at the same rate as f, so the same order of work is done at every level of the tree. The tree has $\mathrm{O}(\log n)$ times the work done on one level, yielding $T(n)$ as $\Theta(n^{\log_b a}\log n)$.
3. **Case 3:** $f(n)$ is $\Omega(n^{\log_b a+\varepsilon})$. Here f grows faster than the number of leaves, which means that asymptotically the total amount of work is dominated by the work done at the root node. For the upper bound, we also need an extra smoothness condition on f in this case, namely, that $af(n/b) \le cf(n)$ for some constant $c < 1$ and large n. In this case, $T(n)$ is $\Theta(f(n))$.

Example 4.3

Consider the following recurrence:

$$T(n) = 4T\left(\frac{n}{2}\right)+n$$

In the given relation, $a = 4$ and $b = 2$.

For this recurrence, there are $a = 4$ sub-problems, each dividing the input by $b = 2$, and the work done on each call is $f(n) = n$. Thus, $n^{\log_b a}$ is n^2 and $f(n)$ is $\mathrm{O}(n^{2-\varepsilon})$ for $\varepsilon = 1$, and Case 1 applies. Thus, $T(n)$ is $\Theta(n^2)$.

Example 4.4

Consider the following recurrence:

$$T(n) = 4T\left(\frac{n}{2}\right) + n^2$$

For this recurrence, there are again $a = 4$ sub-problems, each dividing the input by $b = 2$; but now the work done on each call is $f(n) = n^2$. Again, $n^{\log_b a}$ is n^2 and $f(n)$ is thus $\Theta(n^2)$, so Case 2 applies. Thus, $T(n)$ is $\Theta(n^2 \log n)$. Note that increasing the work on each recursive call from linear to quadratic has increased the overall asymptotic running time by only a logarithmic factor.

Example 4.5

Consider the following recurrence:

$$T(n) = 4T\left(\frac{n}{2}\right) + n^3$$

For this recurrence, there are again $a = 4$ sub-problems, each dividing the input by $b = 2$, but now the work done on each call is $f(n) = n^3$. Again, $n^{\log_b a}$ is n^2 and $f(n)$ is thus $\Omega(n^{2+\varepsilon})$ for $\varepsilon = 1$. Moreover, $4(n/2)^3 \leq kn^3$ for $k = 1/2$, so Case 3 applies. Thus, $T(n)$ is $\Theta(n^3)$.

4.3 HASHING

Hashing is technique that transforms a variable length input to a fixed length output. This technique is used for indexing, which helps in constant time search for insertion, deletion and information retrieval.

4.3.1 Hash Table

A hash table is a data structure used for hashing. It is an associative array that contains keys with values. In an array, indices are required to be integer, but a hash table allows floating point numbers, strings, array or structures as keys. Hash tables provide efficient lookup operation, that is, whenever a key is given, hash table provides associated value in almost constant time complexity. This transformation is done using hash function.

4.3.2 Hashing Functions

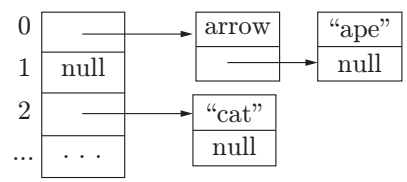
A hashing function is a transformation function, coded to return a value based on key. If h is a hash function that takes key k as parameter, will compute the index at which value is placed. If h is a hash function that takes key as a parameter, it will compute index where value is placed. It is important to note that hash function returns the same index value every time for same key.

4.3.3 Collisions

Collision occurs when one key value stores more than one value. Hash function producesthe same position for two values. Hash table cannot ensure to have single value corresponding to a key. This is because a hash table has fixed number of keys. Any new entry in hash table, after all the slots are filled, causes collision. However, good hash tables use some mechanism to avoid the situations of collisions or to handle them efficiently.

4.3.3.1 Separate Chaining

1. If more than one item is assigned the same hash index, "chain" them together.
2. In hash table, each position act as bucket that can store multiple data items.
3. There are the following two implementations:
   - Each bucket is itself an array: The disadvantages of bucket as array are
     (a) Memory wastage
     (b) Problem of overflow occurs when we try to add new item to a full bucket.
   - Each bucket is a linked list: The disadvantage of bucket as linked list is
     Pointers require more memory.



4.3.3.2 Open Addressing

1. Collision occurs if the hash function produces the same position occupied by another item. In this case, another open position is searched.

   Example: Hash code produced for 'White' is 22. But position 22 is already occupied, so it is placed at position 23.

2. There are three ways of finding an open position. This process is known as **probing**. To search any item in hash table, probing may be required.

   Example: When item "white" is searched, first we look for it at position 22. If it is not found there, we move to next location. The search will be stopped only when an empty position is encountered.

0	Arrow
1	
2	Cat
3	
4	Eye
5	
...	... ...
22	When
23	White
24	Yellow
25	Zero

4.3.3.3 Linear Probing

In case of linear probing the probe sequence is h(key), h(key) + 1, h(key) + 2, …, uptill required.

Examples: The item 'Ape' ($h = 0$) will be placed at position 1, because position 0 is already occupied. then 'Ball' ($h = 1$): try to place it at position 1, 1+1, 1+2. Position 3 is open so it will be placed here.

Advantage: If open position is available, linear probing will definitely find it.

Disadvantage: If there are clusters of filled positions, then length of subsequent probes increases. Probe length is the number of positions considered during a probe.

0	Arrow
1	Ape
2	Cat
3	Ball
4	Eye
5	
...	... ...
22	When
23	White
24	Yellow
25	Zero

4.3.3.4 Quadratic Probing

The probe sequence is

h(key), h(key) + 1, h(key) + 4, h(key) + 9, …, wrapping around as necessary

The offsets are perfect squares: $h + 12$, $h + 22$, $h + 32$, ….

Examples:

a. 'Ape' ($h = 0$): try 0, 0 + 1 − open!
b. 'Ball' ($h = 1$): try 1, 1 + 1, 1 + 4, 1 + 9 − open!

Advantage: It reduces clustering.

0	Arrow
1	Ape
2	Cat
3	
4	Eye
5	Ball
...	... ...
22	When
23	White
24	Yellow
25	Zero

Disadvantage: It may fail to find an existing open position. For example, let table size = 10, and x denote the occupied blocks. Trying to insert a key with h(key) = 0 offsets the probe sequence shown in italic.

0	x	
1	x	1 81
2		
3		
4	x	4 64

5	x	25
6	x	16 36
7		
8		
9	x	9 49

4.3.3.5 Double Hashing

It uses the following two hash functions:

1. h_1 computes the hash code.
2. h_2 computes the increment for probing.

The probe sequence is: h_1, $h_1 + h_2$, $h_1 + 2 \times h_2$, ….

Examples:

1. h_1 = our previous h
2. h_2 = number of characters in the string
3. 'Ape' ($h_1 = 0$, $h_2 = 3$): try 0, 0 + 3 – open!
4. 'Ball' ($h_1 = 1$, $h_2 = 4$): try 1 – open!

Advantages:

1. It combines the good features of linear and quadratic probing:
2. It reduces clustering
3. It will find an open position if there is one, provided the table size is a prime number

0	Arrow
1	Ball
2	Cat
3	Ape
4	Eye
5	
...	... ...
22	When
23	White
24	Yellow
25	Zero

4.3.3.6 Removing Items Under Open Addressing

Consider the following scenario:

Using linear probing:

(a) Insert 'Ape' ($h = 0$): Try 0, 0 + 1 – open!
(b) Insert 'Ball' ($h = 1$): Try 1, 1 + 1, 1 + 2 – open!
(c) Remove 'Ape'.
(d) Search for 'Ape': Try 0, 0 + 1 – no item.
(e) Search for 'Ball': Try 1 – no item, but 'Ball' is further down in the table.

1. On removing an item from a position, leave a special value in that position to indicate that an item was removed.
2. There are three types of positions: occupied, empty and removed.
3. Stop probing on encountering an empty position, but not when encountering a removed position.
4. Insert items in either empty or removed positions.

0	Arrow
1	1
2	Cat
3	Ball
4	Eye
5	
...	... ...
22	When
23	White
24	Yellow
25	Zero

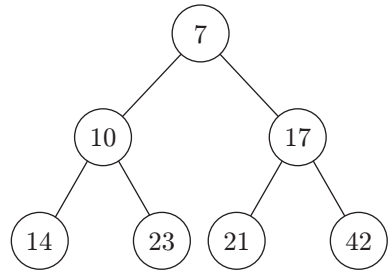
4.4 BINARY HEAP

A binary heap is a binary tree with two additional constraints:

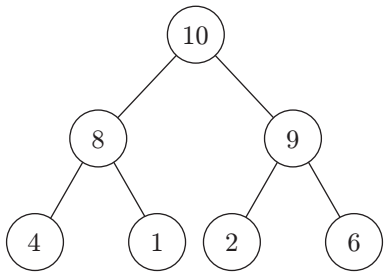
1. Binary heap should satisfy the property of complete binary tree.
2. It should satisfy the heap ordering property, that is, all the nodes should be either greater than or equal to or less than or equal to its children.

According to the heap ordering property, binary heaps are of two types:

1. **Min-Heap Tree:** In Min heap ordering, the value of each node is less or equal to its child node. Minimum element of the tree will be the root element. The largest element exists in the last level. The tree should be an almost complete binary tree.



2. **Max-Heap Tree:** In max heap, each node in the tree will be greater or equal to its child node. The maximum element of tree will be the root element. The smallest element exists in the last level. The tree should be an almost complete binary tree.



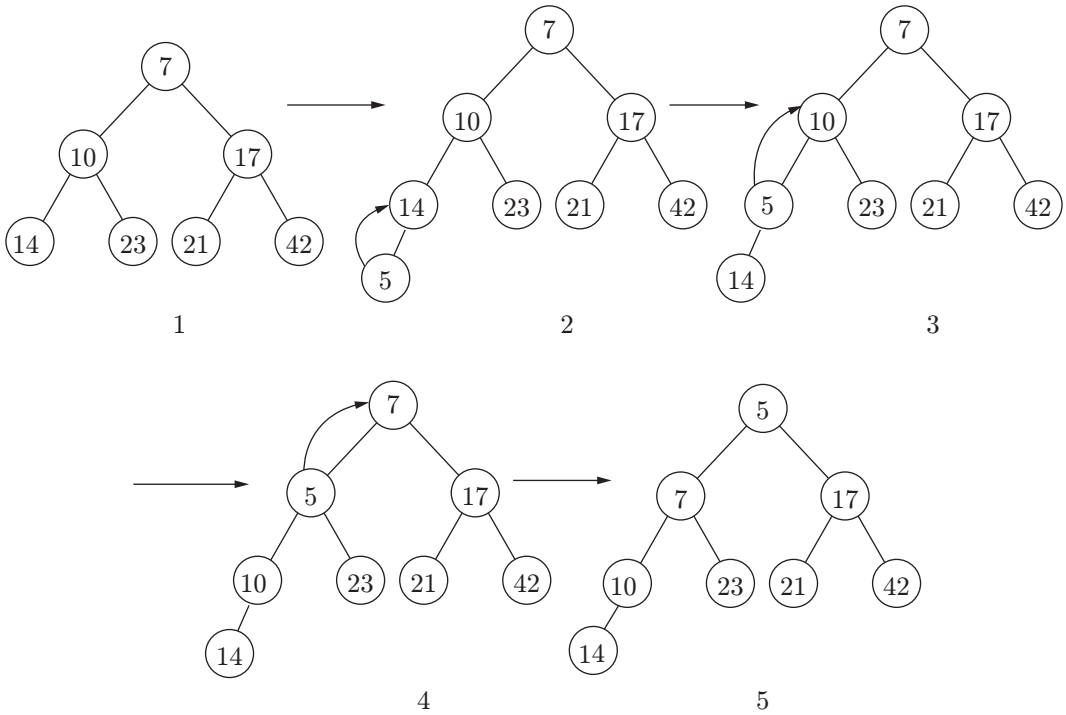
4.4.1 Insertion in Min-Heap Tree

The following are the steps to follow for insertion in the min-heap tree:

1. Add the new element to the last level of heap.
2. Compare the new element with its parent node.
3. If the order is correct then stop, otherwise swap the element with its parent.
4. Perform steps 2 and 3 until a new element is at position.

Example 4.6

1. Insert the given value (5) in the given min-heap tree.
2. Element 5 is inserted at position of left child of node 14. To satisfy the property of min-heap, swap (5, 14).
3. To satisfy the property of min-heap, swap (5, 10).
4. To satisfy the property of min-heap, swap (5, 7).
5. Now, element 5 is in its right position. This is the final tree.



4.4.2 Deletion in Min-Heap Tree

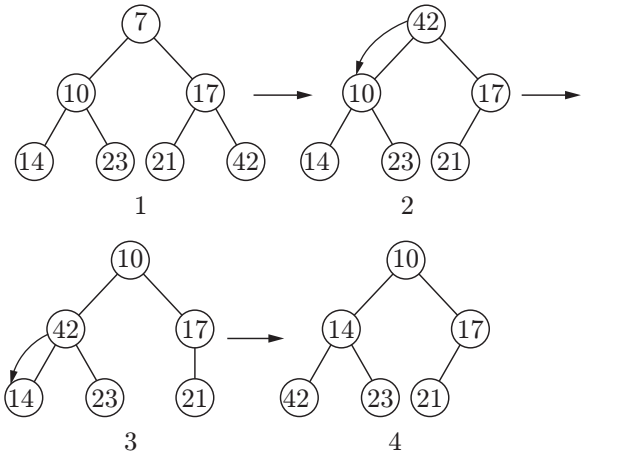
The following are the steps to follow for deletion in the min-heap tree:

1. Root of the heap is replaced with the element at the last level.
2. New root element is compared with its children; if it maintains ordering then stop.
3. Otherwise, swap the root with one of its children and return to previous step. (Smaller element is swapped for min-heap and larger child for max-heap)

Example 4.7

1. Deletion always occurs at the root level, so element 7 will be deleted. Before deletion of element 7 at the root, the element (42) is swapped to the root node from the extreme right-hand side of the last level and then 7 is deleted.
2. To maintain the min-heap property of the tree, swap 42 and min(left-child, right-child), which means swap 42 with 10.
3. To maintain the min-heap property of the tree, swap 42 and min(left-child, right-child), which means swap 42 with 14.

4. This represents the final tree, which is in the min-heap property.



4.4.3 Time Complexity

1. The worst-case running time for insertion is O(log n) because at most (log n) swaps are required to stable the min-heap property of the tree.
2. The worst-case running time for deletion is O(log n) because at most (log n) swaps are required to stable the min-heap property of the tree.

Some important points of binary heap are as follows:

1. Because of its structure, a heap with height k will have between 2^k and $2^{k+1} - 1$ elements. Therefore, a heap with n elements will have height $= \log_2 n$.
2. In heap data structure, element with highest or lowest priority is stored as root element. Heap is not considered as sorted structure, rather it is partially ordered. There is no relation among nodes at a given level.

   Heap is very useful data structure for implementing priority queues. Element with highest priority can be easily removed.
3. Since a binary heap is a complete binary tree, it can be easily represented as an array, and an array-based representation is space efficient. If the parent node is stored at index I, the left child can be calculated by $2 \times I + 1$ and right child by $2 \times I + 2$.

4.5 SEARCHING AND SORTING

Searching and sorting are the most common activities performed by the computer. These operations are the most important part for managing the data.

1. **Searching** is the algorithmic process of finding an item in a collection of items. The search result is either true or false depending on whether or not the item is present or not in the list.
2. **Sorting** is ordering a list of objects according to a given condition or property. For example, ordering a set of numbers in an ascending or descending order.
3. In **internal sorting**, the data which is to be sorted fits into the main memory. Internal sorting has an advantage of the random access nature of the main memory.
4. In **external sorting**, large amount of data is sorted. This large amount of data cannot be fit in main memory, so the data is placed in auxiliary memory. When sorting has to perform, small chunks of data are brought into main memory, sorted and placed back to a temporary file. At last all the sub-files are merged in one large file.
5. In **stable sorting**, the order of equal elements is preserved in a sorted list. Some of the sorting algorithms that show characteristics of stable sort are insertion sort, bubble sort, merge sort, etc.; whereas heap sort, quick sort, etc., are not stable sorts.
6. **In-place sorting:** In this, the input is transformed using a data structure with a small, constant amount of extra storage space. The input is usually overwritten by the output as the algorithm executes. Quicksort is commonly described as an in-place algorithm.
7. **Measurement of time complexity:** The time complexity of a sorting algorithm is measured by the number of critical operations performed. Examples of critical operations are as follows:
   - Key comparisons
   - Movements of records
   - Interchanges of two records
8. We use asymptotic analysis to denote the time efficiency of an algorithm. Efficiency of an algorithm also depends on the type of data for input, so we define best-, worst- and average-case efficiencies.

4.5.1 Linear Search

Linear search is a sequential way of finding an element in a given list. Linear search is a special case of brute force search. Its worst case is proportional to the number of elements in list.

4.5.1.1 Pseudocode for Linear Search

```
LINEAR_SEARCH (A[ ], x)
1. i = 1
2. while (A[i] = = x and i < n)
3. i = i + 1
4. if (A[i] = = x)
5. return i
6. else return 0
```

4.5.1.2 Time Complexity

From the analysis of above pseudocode, it can be observed that with the growth of input list, average and worst case complexities also grow linearly. In general, if there are n elements in the list, the worst-case requires n

comparisons. So, the complexity of linear search can be expressed in terms of some linear function.

So, running time $T(n) = O(n)$.

4.5.2 Binary Search

Binary search relies on a divide-and-conquer strategy to find an element within an already sorted list. It compares the element to be searched with the middle element of array. If the element is equal to the middle element, the search stops here and the position of element is returned. If it is less or greater than the middle element, then the values of sub-array are adjusted accordingly. The algorithm returns a unique value, that is, position of the element in the list, if the element is present.

4.5.2.1 Pseudocode for Binary Search

```
BINARY_SEARCH (A, value, left, right)
1. if right < left
2. return 'not found'
3. mid = floor((right-left)/2)+left
4. if A[mid] = value
5. return mid
6. if value < A[mid]
7. return BINARY_SEARCH(A, value, left, mid-1)
8. else
9. return BINARY_SEARCH(A, value, mid+1, right)
```

4.5.2.2 Time Complexity

The running time of the binary search is $\Theta(\log n)$.

4.5.3 Bubble Sort

Bubble sort algorithm is a comparison-based algorithm. It compares each element in the list with the next element. Elements are swapped if required. In every pass, one element comes to its position. This process repeats until a pass is made without disturbing any of the elements from their position.

4.5.3.1 Pseudocode for Bubble sort

```
BUBBLE_SORT(A)
1. for i ← 1 to n-1
2. for j ← 1 to n-1
3. If(A(j) > A(J+1))
4. Temp = A(j)
5. A(j) = A(j+1)
6. A(j+1) = Temp
```

Example 4.8

Sort the given elements (7, 5, 2, 4, 3 and 9) by using bubble sort.

After the first pass, we get

7, **5**, 2, 4, 3, 9

5, **7**, **2**, 4, 3, 9

5, 2, **7**, **4**, 3, 9

5, 2, 4, **7**, **3**, 9

5, 2, 4, 3, **7**, **9**

After the first pass, the first largest element (9) is on the top of array. So we have

5, 2, 4, 3, 7, 9

After the second pass, we get

2, 4, 3, 5, 7, 9

After the third pass, we get

2, 3, 4, 5, 7, 9

After the fourth pass, we get

2, 3, 4, 5, 7, 9

After the fifth pass, we get

2, 3, 4, 5, 7, 9

The sorted list is 2, 3, 4, 5, 7, 9

4.5.3.2 Time Complexity

It can be seen from the above example that in the bubble sort, each element passes one at a time and is placed in its correct position.

So, the number of passes = (number of elements − 1) = $(n - 1)$.

Cost per pass = $O(n)$.

Total cost = $O(n)(n - 1)$.

So, the worst-case runtime complexity is $O(n^2)$.

The average-case and best-case runtime complexity is $O(n^2)$.

4.5.4 Selection Sort

The selection sort first selects the smallest element from the unsorted list. Then that element is swapped with the first element in the list. In the next step, the size of the list is reduced by one because one element is present at its position. Next, the smallest element is swapped with the second element in list, and so on.

4.5.4.1 Pseudocode for Selection Sort

```
SELECTION_SORT(A)
1. for j ← 1 to n-1
2. smallest ← j
3. for i ← j + 1 to n
4. if A[ i ] < A[ smallest]
5. smallest ← i
6. Exchange A[ j ] ↔ A[ smallest]
```

Example 4.9

Sort the given elements (29, 64, 73, 34 and 20) by using selection sort.

We have

29, 64, 73, 34, 20

After the first pass, we get

20, 64, 73, 34, 29

After the second pass, we get

20, 29, 73, 34, 64

After the third pass, we get

20, 29, 34, 73, 64

After the fourth pass, we get

20, 29, 34, 64, 73

After the fifth pass, we get

20, 29, 34, 64, 73

The sorted list is 20, 29, 34, 64, 73.

4.5.4.2 Time Complexity

Number of passes = (Number of elements − 1)

Cost per pass = Swap + Comparisons = 1 + O(n)

Total cost = (n − 1)[1 + O(n)]

So, the worst-case runtime complexity is O(n^2).

The average-case and best-case runtime complexity is O(n^2).

4.5.5 Insertion Sort

The insertion sort inserts each element into its proper position. It chooses one element, inserts it to its position and shifts the rest of the list by one location. This process is repeated until no input element remains. Step-by-step procedure is given as follows.

1. Take an element from unsorted list.
2. Compare that element with sorted list from right to left.
3. Shift the list.

Here, it is assumed that first element is already sorted.

4.5.5.1 Pseudocode for Insertion Sort

```
INSERTION_SORT(A)
1. for j = 2 to n
2. key  ← A [j]
3. // Insert A[j] into the sorted sequence A[1..j-1]
4. j  ← i - 1
5. while i > 0 and A[i] > key
6. A[i+1]   ← A[i]
7. i  ← i - 1
8. A[j+1]   ← key
```

Example 4.10

Sorted part is bold in text and unsorted part is non-bold.

Suppose **30**, 10, 75, 33, 65 needs to be sorted using insertion sort.

After the first pass, we get

10, 30, 75, 33, 65

After the second pass, we get

10,30,75, 33, 65

After the third pass, we get

10, 30, 35, 75, 65

After the fourth pass, we get

10, 30, 35, 65, 75

The sorted list is 10, 30, 35, 65, 75.

4.5.5.2 Recurrence Relation of Insertion Sort

The recurrence relation for insertion sort is

$$T(n) = \begin{cases} \Theta(1), & n = 1 \\ T(n-1) + \Theta(n), & n > 1 \end{cases}$$

4.5.5.3 Time Complexity

In sorting, the most expensive part is the comparison of two elements. Zero comparisons are required to insert first element, one for second element, two comparisons for third element, and so on. Atmost $N - 1$ comparisons are required for the last element in the list.

Complexity is: $1 + 2 + 3 + 4 \cdots + (N - 1) = O(n^2)$

The worst-case running time is O(n^2)

The best-case running time is O(n)

4.5.6 Heap Sort

Heap sort is one of the comparison based sort that uses heap data structure. It is a special case of selection sort. Its typical implementation is not stable, but can be made stable. Heap can be built by the following two types:

1. **Min-Heap:** The parent node will be less than or equal to its children nodes. So, the minimum element will be at root.
2. **Max-Heap:** The parent node will be greater than or equal to its children nodes. The maximum element will be at the root.

4.5.6.1 Heap Sort Algorithm for Sorting in Increasing Order

1. First build a max-heap from the input List.
2. In max heap, the largest element is at the root. So, swap the root with last element. Re-heapify the

tree. Repeat the above steps until the size of the heap is greater than 1.

- **Heapify function:** Heapify is an important subroutine for manipulating max-heaps. Its inputs are an array A and an index i into the array. When Heapify is called, it is assumed that the binary tree rooted at left(i) and right(i) is max-heap, but that $A[i]$ may be smaller than its children, thus violating the max-heap property.
- **Build Heap function:** Heapify procedure is used in a bottom-up to convert an array A into a max-heap. The elements in the subarray $A[(\lfloor n/2 \rfloor + 1) \cdots n]$ are all leaves of the tree, and so each is a one-element heap to begin with. The procedure BuildHeap goes through the remaining nodes of the tree and runs Heapify on each node.

4.5.6.2 Pseudocode for Heapify Function

```
HEAPIFY(A, i)
1. le <- left(i)
2. ri <- right(i)
3. if (le<=heapsize) and (A[le]>A[i])
4. largest <- le
5. else
6. largest <- i
7. if (ri<=heapsize) and (A[ri]>A[largest])
8. largest <- ri
9. if (largest != i)
10. exchange A[i] <-> A[largest]
11. HEAPIFY(A, largest)
```

4.5.6.3 Pseudocode for BuildHeap Function

```
BUILD_HEAP(A)
1. heapsize <- length(A)
2. for i <- floor( length/2 ) down to 1
3. Heapify(A, i)
```

4.5.6.4 Pseudocode for Heap Sort

```
HEAP_SORT(A)
1.   BuildHeap(A)
2.   for i <- length(A) down to 2
3.      exchange A[1] <-> A[i]
4.      heapsize <- heapsize -1
5.      Heapify(A, 1)
```

4.5.6.5 Recurrence Relation of Heapify Function

The recurrence relation of Heapify() is

$$T(n) \leq \left\{ T\left(\frac{2n}{3}\right) + \Theta(1) \right\}$$

4.5.6.6 Time Complexity

The time complexity of Heapify() is O(log n). The time complexity of BuildHeap() is O(n log n), and the overall time complexity of the heap sort is O(n log n).

4.5.6.7 Applications of Heap Sort

1. It sorts a nearly sorted (or k-sorted) array.
2. It sorts k largest (or smallest) elements in an array.

Example 4.11

Let {6, 5, 3, 1, 8, 7, 2, 4} be the list that we want to sort from the smallest to the largest. (**Note**: For 'building the heap' step, larger nodes do not stay below smaller node parents. They are swapped with parents, and then recursively checked if another swap is needed, to keep larger numbers above smaller numbers on the heap binary tree.)

1. Build the heap:

Heap	Newly Added Element	Swap Elements
Nil	16	
16	20	
20 16		16 20
20 16 11	11	
20 16 11 8	8	
20 16 11 8	18	
20 16 11 8 18		16 18
20 18 11 8 16	14	
20 18 11 8 16 14		11 14
20 18 14 8 16 11		

2. Sort the list:

Heap	Swap Elements	Delete Element	Sorted Array	Details
20 18 14 8 16 11	20, 11			Swap 20 and 11 in order to delete 20 from the heap.
11 18 14 8 16 **20**		20		Delete 20 from the heap and add to the sorted array.
11, 18, 14, 8, 16, **20**	11, 18		20	Swap 11 and 18 as they are not in order in the heap.
18 11 14 8 16 **20**	11 16		20	Swap 11 and 16 as they are not in order in the heap.
17, 15, 16, 7, 10, **19** 18 16 14 8 11 **20**	18 11		20	Swap 18 and 11 in order to delete 17 from the heap.
11 16 14 8 **18 20**		18	20 18	Delete 18 from the heap and add to the sorted array.
11 16 14 8 **18 20**	11 14		20 18	Swap 11 and 14 as they are not in order in the heap.
11 14 16 8 **18 20**	14 16		20 18	Swap 14 and 16 in order to delete 14 from the heap.
16 14 11 8 **18 20**			20 18 16	Delete 16 from the heap and add to the sorted array.
8 14 11 **16 18 20**	8 14		20 18 16	Swap 14 and 8 as they are not in order in the heap.
14 8 11 **16 18 20**	14 11		20 18 16	Swap 14 and 11 in order to delete 14 from the heap.
10, 7, **15**, **16**, **17**, **19** 11 8 **14 16 18 20**		14	20 18 16 14	Delete 14 from the heap and add to the sorted array.
11 8 **14 16 18 20**	11 8		20 18 16 14	Swap 11 and 8 in order to delete 10 from the heap.
8, **11**, **15**, **16**, **17**, **19**		11	20 18 16 14 11	Delete 11 from the heap and add to the sorted array.
7, **10**, **15**, **16**, **17**, **19** **8 11 14 16 18 20**		8	20 18 16 14 11 8	Delete 8 from the heap and add to the sorted array.
			20 18 16 14 11	Completed

4.5.7 Merge Sort

Merge sort is a divide-and-conquer algorithm. It divides the input array in two subarrays, calls itself for the two sub-arrays and then merges the two sorted arrays. The MERGE(A, p, q, r) function is used for merging the two arrays. The MERGE(A, p, q, r) is a key process that assumes that array[$l \cdots m$] and array[$m + 1 \cdots r$] are sorted and merges the two sorted sub-arrays into one. Merge sort is a stable sort. It is not an in-place sorting technique.

4.5.7.1 Pseudocode for Merge Sort

```
MERGE_SORT (A, p, r)
1. if p < r
2. Then q = floor[(p + r)/2]
3. MERGE (A, p, q)
4. MERGE (A, q + 1, r)
5. MERGE (A, p, q, r)
```

Pseudocode for MERGE function is given as follows:

```
MERGE (A, p, q, r)
1. n1 ← q − p + 1
2. n2 ← r − q
3. create arrays L[1...n1 + 1] and R[1...n2 + 1]
4. For i ← 1 to n1
5. doL[i] ← A[p + i − 1]
6. For j ← 1 to n2
7. Do R[j] ← A[q + j]
```

```
8. L[n1 + 1]  ← ∞
9. R[n2 + 1]  ← ∞
10. i ← 1
11. j ← 1
12. For k  ← p to r
13. Do if L[i]  ≤ R[j]
14. then A[k]  ← L[i]
15. i ← i + 1
16. Else A[k]  ← R[j]
17. j ← j + 1
```

4.5.7.2 Recurrence Relation of Merge Sort

The recurrence relation for merge sort is

$$T(n) = \begin{cases} \Theta(1), & n = 1 \\ 2T\left(\dfrac{n}{2}\right) + \Theta(n), & n > 1 \end{cases}$$

4.5.7.3 Time Complexity

The above recurrence can be solved using either recurrence tree method or master theorem. It falls under case 2 of the master theorem, and the solution of the recurrence is O(n log n). The time complexity of merge sort in all three cases (worst, average and best) is the same, as merge sort always divides the array in two sub-arrays and takes linear time to merge the two arrays.

4.5.7.4 Applications of Merge Sort

1. Merge sort is useful for sorting linked lists in O(n log n) time. Other n log n algorithms such as heap sort and quick sort (average case n log n) cannot be applied to linked lists.
2. It is used in inversion count problem.
3. It is used in external sorting.

Note: Merge sort is not preferable for smaller-size element array.

Example 4.12

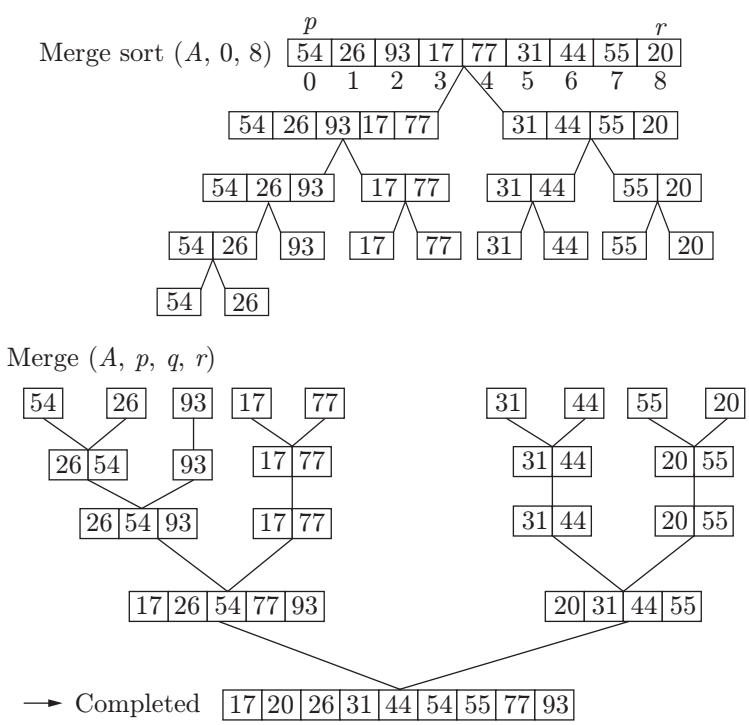
Sort the given list of elements: (54, 26, 93, 17, 77, 31, 44, 55 and 20).

Step 1: A recursive merge sort is called on the array of elements. Merge sort is a recursive function which divides an array into two parts; and calls itself recursively to divide the array into sub-parts until each part is having one element.

Step 2: After dividing the array, merge sort calls the MERGE function to put the subarray into a sorted order.

Array A:

54	26	93	17	77	31	44	55	20



4.5.8 Quick Sort

Quicksort is also divide-and-conquer strategy of sorting a list of elements like merge sort. This divides array of elements into sub-array $S[p \ldots r]$. The concept is described as follows:

1. **Divide:** Partition $S[p \ldots r]$ into two sub-arrays $S[p \ldots q-1]$ and $S[q+1 \ldots r]$ such that each element of $S[p \ldots q-1]$ is less than or equal to $S[q]$, which is, in turn, less than or equal to each element of $S[q+1 \ldots r]$. Compute the index q as part of this partitioning procedure.
2. **Conquer:** Sort the two sub-arrays $S[p \ldots q-1]$ and $S[q+1 \ldots r]$ by recursive calls to quicksort.
3. **Combine:** Since the sub-arrays are sorted in place, no work is needed to combine them; the entire array S is now sorted.

4.5.8.1 Pseudocode for Quick Sort

```
QUICKSORT(S, P, r)
1. If p < r
2. then q <- PARTITION(S, p, r)
3. QUICKSORT(S, p, q-1)
4. QUICKSORT(S, q+1, r)
```

4.5.8.2 Pseudocode for Partition

```
PARTITION(S, p, r)
1. x <- S[r]
2. i <- p-1
3. for j <- p to r-1
4. do if S[j] <= x
5. then i <- i+1
6. swap S[i] <-> S[j]
7. swap S[i+1] <-> S[r]
8. return i+1
```

4.5.8.3 Recurrence Relation of Quick Sort

1. **For worst case:** The worst case of quick sort occurs when the pivot we picked turns out to be the least element of the array to be sorted, in every step (i.e. in every recursive call). A similar situation will also occur if the pivot happens to be the largest element of the array to be sorted. Then recurrence relation of quick sort is
$$T(n) = T(1) + T(n-1) + \Theta(n)$$
2. **For best case:** The best case of quicksort occurs when the pivot we picked happens to divide the array into two almost equal parts, in every step. Thus, we have $k = n/2$ and $n - k = n/2$ for the original array of size n. Then the recurrence relation of quick sort is given as
$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

4.5.8.4 Time Complexity

The best-case and average-case running time of quick sort $= O(n \log n)$.

The worst-case running time of quick sort $= O(n^2)$.

Note: The worst case of quick sort occurs when the given elements are already sorted or almost sorted.

Example 4.13

Sort the given list of elements: (38, 81, 22, 48, 13, 69, 93, 14, 45, 58, 79, 72)

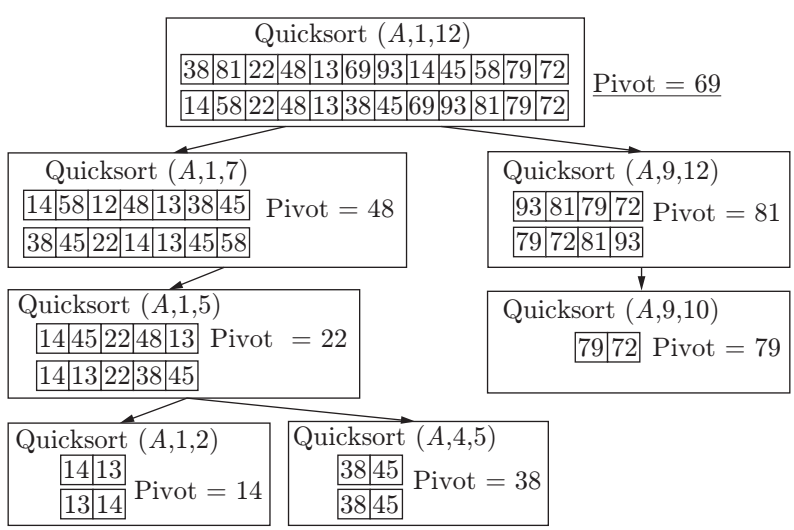
Step 1: Choose the pivot element which is at position [(left + right)/2].

Step 2: Now in partitioning phase: During the partitioning process:

(a) Elements at the left of the pivot are less than or equal to that of pivot.

(b) Elements at the right of pivot are greater than that of pivot.

The tree of recursive call is given as follows:



4.5.9 Randomized Quick Sort

In the randomized version of quick sort, we impose a distribution on the input. This does not improve the worst-case running time, but it improves the probability of getting the average-case running time.

In this version, we choose a random key for the pivot. Assume that the procedure Random(a, b) returns a random integer in the range [a, b); there are $b - a + 1$ integers in the range, and the procedure is equally likely to return one of them. The new partition procedure simply implements the swap before actually partitioning.

4.5.9.1 Pseudocode for Randomized Quick Sort

```
RANDOMIZED_PARTITION (A, p, r)

1. i ← RANDOM (p, r)
2. Exchange A[p] ← A[i]
3. return PARTITION (A, p, r)
```

The pseudocode for randomized quick sort has the same structure as quick sort, except that it calls the randomized version of the partition procedure.

```
RANDOMIZED_QUICKSORT (A, p, r)

1. If p<r then
2. Q ← RANDOMIZED_PARTITION (A, p, r)
3. RANDOMIZED_QUICKSORT (A, p, q)
4. RANDOMIZED_QUICKSORT (A, q+1, r)
```

4.5.9.2 Time Complexity

The best- and average-case running time of randomized quick sort = O(n log n).

The worst-case running time of randomized quick sort = O(n^2).

Note: The worst case of randomized quick sort occurs when all the given elements are equal.

4.5.10 Counting Sort

Counting sort is a sorting technique based on keys within a specific range. It works by counting the number of objects having distinct key values (kind of hashing), and then doing some arithmetic to calculate the position of each object in the output sequence.

4.5.10.1 Pseudocode for Counting Sort

```
COUNTING_SORT(A[], B[], k)
1. for i = 1 to k do
2. C[i] = 0
3. for j = 1 to length(A) do
4. C[A[j]] = C[A[j]] + 1
5. for 2 = 1 to k do
6. C[i] = C[i] + C[i-1]
7. for j = 1 to length(A) do
       B[C[A[j]]] = A[j]
       C[A[j]] = C[A[j]] - 1
```

Although this may look complicated, it is actually a very simple and clever algorithm.

1. An array A[ ] stores the initial data to be sorted.
2. An array C[ ] is used to count the occurrences of the data values.
3. An array B[ ] is used to store the final, sorted, list.
4. The first `for` loop initializes C[ ] to zero.
5. The second `for` loop increments the values in C[ ], according to their frequencies in the data.
6. The third `for` loop adds all the previous values, making C[ ] contain a cumulative total.
7. The fourth `for` loop writes out the sorted data into the array B[ ].

4.5.10.2 Time Complexity

O($n + k$), where n is the number of elements in the input array and k is the range of input.

Note:

1. Counting sort is efficient if the range of input data is not significantly greater than the number of objects to be sorted.
2. It is not a comparison-based sorting. Its running time complexity is O(n) with space proportional to the range of data.
3. This can be used as subroutine to some other algorithm. It is often used as a subroutine to another sorting algorithm, such as radix sort.
4. It uses a partial hashing to count the occurrence of the data object in O(1).
5. It can be extended to work for negative inputs also.

Example 4.14

Consider the data in the range 0 to 9.

Input data: 1, 4, 1, 2, 7, 5, 2

The steps to sort elements through counting sort are as follows:

1. Take a count array to store the count of each unique object:

   Index: 0 1 2 3 4 5 6 7 8 9

   Count: 0 2 2 0 1 1 0 1 0 0
2. Modify the count array such that each element at each index stores the sum of the previous counts:

   Index: 0 1 2 3 4 5 6 7 8 9

   Count: 0 2 4 4 5 6 6 7 7 7

The modified count array indicates the position of each object in the output sequence.

Table 4.1 | Sorting techniques

Sorting Algorithm	Average Time	Best Time	Worst Time	Space	Stability	In-Place Sorting
Bubble sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	Constant	Stable	Yes
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	Constant	Stable	Yes
Insertion sort	$O(n^2)$	$O(n)$	$O(n^2)$	Constant	Stable	Yes
Heap sort	$O(n \times \log(n))$	$O(n \times \log(n))$	$O(n \times \log(n))$	Constant	Instable	Yes
Merge sort	$O(n \times \log(n))$	$O(n \times \log(n))$	$O(n \times \log(n))$	Depends	Stable	No
Quick sort	$O(n \times \log(n))$	$O(n \times \log(n))$	$O(n^2)$	Constant	Stable	Yes
Count sort	$O(n)$	$O(n)$	$O(n)$	Constant	Stable	No

3. Output each object from the input sequence followed by decreasing its count by 1.
   (a) Process the input data: 1, 4, 1, 2, 7, 5, 2. Position of 1 is 2.
   (b) Put data 1 at index 2 in the output. Decrease count by 1 to place the next data 1 at an index 1 smaller than this index.
   (c) Put data 2 at index 4 in the output. Decrease count by 1 to place the next data 2 at an index 3 smaller than this index.
   (d) Now, move to the next element 3 which has count 4, and index 4 is already filled by 2 which means element 3 does not exist in the given input list.
   (e) Follow this process for all the elements.

4.5.11 Comparison of Sorting Techniques

The comparison of the various sorting techniques is given in Table 4.1.

4.6 GRAPH

A graph $G = (V, E)$ is a set of vertices and edges where each edge consist of pair of vertices.

1. Finite set of vertices are known as nodes of graph.
2. (u, v) is a pair of vertices called edge.
3. For directed graph (u, v) is an ordered pair that represents an edge from node u to node v.

4.6.1 Types of Graph

There are two types of graphs:

1. **Simple graph:** A graph is called a simple graph (Fig. 4.5) if
   - It contains no self-loop.
   - It does not have any parallel edge.

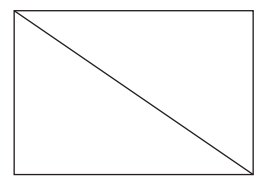


Figure 4.5 | Simple graph.

2. **Multigraph:** A graph with a self-loop and parallel edges is called a multigraph (Fig. 4.6).

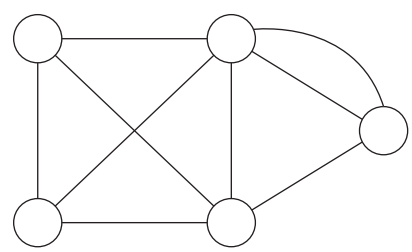


Figure 4.6 | Multigraph.

4.6.2 Types of Simple Graph

4.6.2.1 Null Graph

If in the given graph there is no edge, then it is called a null graph. In other words, a null graph does not have any edge between any two vertices (Fig. 4.7).

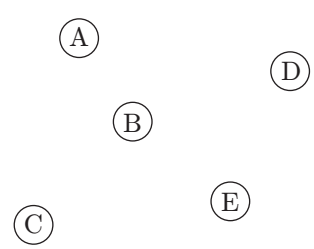


Figure 4.7 | Null graph.

4.6.2.2 Regular Graph

A graph is called a regular graph if all of its vertices have the same degree (Fig. 4.8).

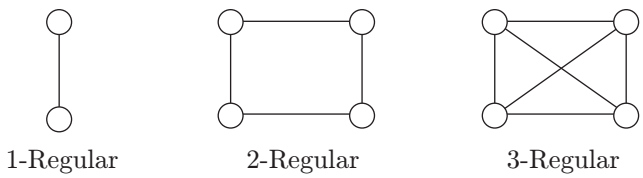


Figure 4.8 | Regular graph.

If a graph has n number of vertices and it is k-regular, where k is a positive constant, then the number of edges $= (n \times k)/2$.

4.6.2.3 Complete Graph

In the given graph, if every vertex is adjacent to all other remaining vertices, then that graph is called a complete graph (Fig. 4.9). If a graph is a complete graph and is having n vertices, then it is called a K_n graph.

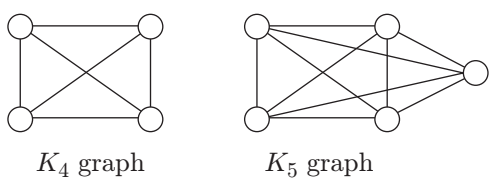


Figure 4.9 | Complete graphs.

A complete graph having n vertices has $[n \times (n-1)]/2$ edges.

Note: The number of simple graphs possible with n vertices $= 2^{[n(n-1)]/2}$.

4.6.3 Graph Representation

Graph data structure is useful in many real-time applications like they are used to represent complex networks. The network represented by graph can be of city defining paths, of telephone or circuit network.

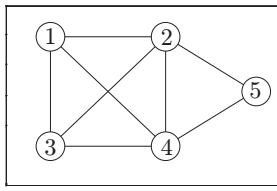
The following two are the most commonly used representations of a graph:

1. Adjacency matrix
2. Adjacency list

Incidence matrix and incidence list are two other representations for graph. Any representation can be chosen depending on the situation.

4.6.3.1 Adjacency Matrix

An adjacency matrix is a two-dimensional array of size $V \times V$, where V represents the number of vertices in a graph. If adj[i][j] = 1, this indicates there exists an edge between vertices i and j; otherwise, the value will be 0. The adjacency matrix for undirected graph is symmetric. Weights can also be represented by this matrix. The value adj[i][j] = w means weight of edge from i to j is w.



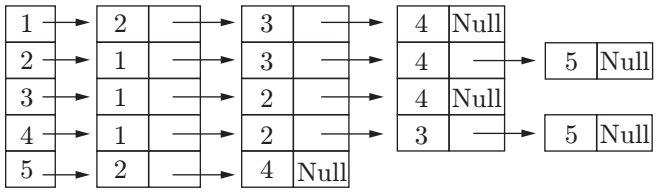
Vertex	1	2	3	4	5
1	0	1	1	1	0
2	1	0	1	1	1
3	1	1	0	1	0
4	1	1	1	0	1
5	0	1	0	1	0

This type of representation is very easy for implementation purpose. O(1) time is required to remove an edge. Testing that an edge exists between two nodes or not can be computed in O(1) time complexity.

Space requirement is more for both dense and sparse matrices is O(V^2). Addition of a vertex takes O(V^2) time.

4.6.3.2 Adjacency List

To implement adjacency list, an array of linked list is used. Array size is kept equal to the total number of vertices. For each vertex i, linked list of its adjacent nodes is stored in array[i]. Weights of edges can also be stored in linked list nodes. The representation of adjacency list is as follows:



The space required for adjacency list is: O(V+E), which is less than that of adjacency matrix. In worst-case, space consumed is O(V^2). Addition of vertex is easy, but finding an edge between vertices, say u and v, is not efficient. It requires O(V) complexity.

4.7 GREEDY APPROACH

A greedy algorithm is an algorithm for solving problems in an optimised way. In this approach,the algorithms execute repeatedly to maximize the outcome, considering the local conditions for some global problem. For some problems, it guarantees the optimal solution while for some it does not. The main strategy is to implement the problem so that it requires minimal resources.

The important terms are given as follows:

1. **Solution space:** The set of all possible solutions for given n inputs is called solution space.
2. **Feasible solution:** The feasible solution is one of the solutions from the solution space which satisfies the given condition.
3. **Optimal solution:** The optimal solution is one of the feasible solutions which optimizes our goal.

4.7.1 Applications of Greedy Approach

The following are the applications of the greedy approach:

1. Knapsack problem (fractional)
2. Job sequence with deadline
3. Huffman coding
4. Minimum cost spanning tree:
   a. Kruskal's algorithm
   b. Prim's algorithm
5. Singlesource shortest path:
   a. Dijkstra's algorithm
   b. Bellman–Ford algorithm

4.7.2 Fractional Knapsack

In this problem, items are given along with its weight and profit. The target is to maximize the profit considering the weight constraint. Unlike 0-1 knapsack problem, fraction of item can be considered. Fractional knapsack problem can be solved by Greedy approach.

Greedy Algorithm: The solution can be obtained by making different choices. At each stage, the best possible choice is made. In fractional knapsack, first of all profit value/weight ratios are calculated and then sorted in descending order. Item with highest value/weight ratio is chosen and added in the collection. The maximum weight is checked after adding each item. If the entire item cannot be included, then fraction of it is added to the collection.

Example 4.15

There are n items in a store. For $i = 1, 2, \ldots, n$, item i has weight $w_i > 0$ and worth $v_i > 0$. A thief can carry a maximum weight of W pounds in a knapsack. In this version of a problem, the items can be broken into smaller pieces, so the thief may decide to carry only a fraction x_i of object i, where $0 \leq x_i \leq 1$. Item i contributes $x_i w_i$ to the total weight in the knapsack, and $x_i v_i$ to the value of the load.

In symbol, the fraction knapsack problem can be stated as follows.

Maximize $nS_i = 1x_i v_i$ subject to constraint $nS_i = 1x_i w_i \leq W$. It is clear that an optimal solution must fill the knapsack exactly; otherwise, we could add a fraction of one of the remaining objects and increase the value of the load. Thus, in an optimal solution, $nS_i = 1x_i w_i = W$.

4.7.2.1 Pseudocode for Fractional Knapsack

```
Greedy-fractional-knapsack (w, v, W)

for i =1 to n
Do x[i] =0
weight = 0
while weight < W
Do i = best remaining item
if ( weight + w[i] ≤ W)
Then x[i] = 1
weight = weight + w[i]
else
x[i] = (w - weight) / w[i]
weight = W
Return x
```

4.7.2.2 Time Complexity

If the ratio of v_i/w_i is already sorted in decreasing order, then the time taken by the while loop will be O(n). So, the total time required will be O(n log n). If heap data structure is used, which keeps highest value/weight ratio at root. The construction of heap will take O(n) time and while loop will take O(log n) time (heap property is restored for each root removal). This does not improve the worst case, but it is faster if small number of items are to be filled in knapsack.

Example 4.16

Consider the following details:

1. Knapsack size = 20
2. Number of objects = 3

Object	Obj1	Obj2	Obj3
Profit	25	24	15
Weight	18	15	10

Solution:

Step 1: Calculate ratio (profit/weight):

Object	Obj1	Obj2	Obj3
Profit	25	24	15
Weight	18	15	10
Ratio	1.38	1.6	1.5

Step 2: Put according to decreasing ratio (profit/weight):

Object	Obj2	Obj3	Obj1
Profit	24	15	25
Weight	15	10	18
Ratio	1.6	1.5	1.38

Step 3: Put items into the knapsack till the knapsack is full:

(a) Put object 2 into the knapsack; it will add (value = 24 and weight = 15) to the knapsack. Knapsack has 5 units capacity remaining.
(b) Now object 3 cannot be put completely into the knapsack, so put a fraction of it. Add (value = 7.5 and weight = 5) to the knapsack. Knapsack is completely full, which means no other object can be put into it.

Step 4: Total knapsack value = 31.5 (which is optimum).

4.7.3 Huffman Coding

Huffman coding is a common technique of encoding, includinglossless data compression. The algorithm's output can be viewed as a variable-length code table for encoding a source symbol (such as a character in a file). Huffman coding is based on the frequency of occurrence of a data item (pixel in images). The main aim of Huffman coding is to encode the data that occurs frequently using less number of bits. Code books store codes that are constructed for images. Both the code book and encoded data are transmitted to decode the information.

Huffman codes can be constructed using following two ideas:

1. For an optimum code, symbols that have high probability should be assigned shorter code words.
2. To make optimum prefix code, two symbols that occur least frequently should be assigned same length.

4.7.3.1 Principle of Huffman Codes

1. Starting with the two least probable symbols, γ and δ, of an alphabet A, if the code word for γ is $[m]0$, the code word for δ would be $[m]1$, where $[m]$ is a string of 1s and 0s.
2. Now, the two symbols can be combined into a group, which represents a new symbol ψ in the alphabet set.
3. The symbol ψ has the probability $P(\gamma) + P(\delta)$. Recursively, determine the bit pattern $[m]$ using the new alphabet set.

4.7.3.2 Pseudocode for Huffman Coding

```
HUFFMAN(f[1...n])
1. T = empty binary tree
2. Q = priority queue of pairs (i, f[i]),
   i = 1...n, with f as comparison key
3. for each k = 1...n   - 1
4. i = extractMin(Q)
5. j = extractMin(Q)
6. f[n + k] = f[i] + f[j]
7. insertNode(T, n + k) with children i, j
8. insertRear(Q,(n + k, f[n + k]))
9. return T
```

4.7.3.3 Time Complexity

From the above pseudocode, we can see that with each iteration,the problem size is reduced by 1. Hence, there are exactly n iterations. The ith iteration consists of locating the two minimum values in a list of length $n - i + 1$. This is a linear operation, and so Huffman's algorithm clearly has a time complexityof $O(n^2)$.

However, it would be faster to sort the weights initially, and then maintain two lists. The first list consists of weights that have not been combined yet, and the second list consists of trees that have been formed by combining weights. This initial ordering is obtained at the cost of $O(n \log n)$. Obtaining the minimum two trees at each step then consists of two comparisons (comparing the heads of the two lists, and then comparing the larger item with the item after the smaller). The ordering of the second list can be maintained cheaply by using a binary search to insert new elements. Since at step i there are $i - 1$ elements in the second list, $O(\log i)$ comparisons are needed for insertion. Over the entire duration of the algorithm, the cost of keeping this list sorted is $O(n \log n)$. Therefore, the overall time complexity of Huffman's algorithm is $O(n \log n)$.

Example 4.17

In the following example five different symbols are given along with their frequency.

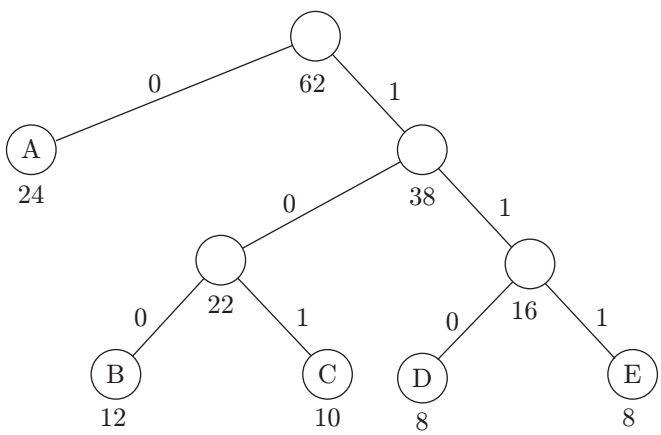
Symbol	Frequency
A	24
B	12
C	10
D	8
E	8

There are a total of 186 bits (with 3 bits per code word).

Here, the least frequency symbols are E and D. First of all, we will connect them and frequency of node will become 16. Then B and C are combined to make the

frequency 22. These will be two new nodes. In the next step, these two nodes are brought together and act as subordinate to the root node. A also acts as subordinate to root node.

4.7.3.4 Code Tree According to Huffman



Symbol	Frequency	Code	Code Length	Total Length
A	24	0	1	24
B	12	100	3	36
C	10	101	3	30
D	8	110	3	24
E	8	111	3	24

The total length is 138 bits.

4.7.4 Minimum Spanning Tree

Spanning tree is a sub-graph that connects all the vertices together. A single graph can have multiple spanning trees. Minimum spanning tree for a weighted undirected graph is the spanning tree with minimum total weights. The weight of spanning tree is addition of weights of all the edges in the spanning tree. An MST with N vertices, has $N - 1$ edges for a given graph.

4.7.4.1 Prim's Algorithm

Prim's algorithm is a method to find minimum spanning tree. In this algorithm minimum spanning tree formed should always be connected. Prim's algorithm is an example of Greedy approach.

Properties of Prim's Algorithm

Prim's algorithm has the following properties:

1. Prime's algorithm always results into a single tree.
2. Tree grows until it covers all the vertices of the graph.
3. At each step, an edge is added to the graph that has minimum weight and is the neighbour to the previous vertices.

Steps of Prim's Algorithm

1. Find the minimum weight edge from graph and mark it.
2. Consider the neighboring edges of the selected edges and mark the minimum weight edge among those edges.
3. Continue this until we cover n-1 edges.
4. The resultant tree is minimum spanning tree and it should always be connected.

Input: Undirected connected weighted graph $G = (V, E)$ in adjacency list representation, source vertex s in V and w is an array representing weights of edges.

Output: $p[1 \cdots |V|]$, representing the set of edges composing an MST of G.

Note: The field $\pi(v)$ names the parent of v in the tree.

Key(v) is the minimum weight of any edge connecting v to a vertex in tree. Key(u)$\leftarrow\infty$ means there is no such edge.

```
PRIM (G, w, s)
for each u in V(G)
do key(u) ← ∞
π(v) ← NIL
key [r] ← 0
Q ← V [G]
while Q != empty
do u ← Extract-Min(Q)
for each v in Adjacent[u]
do if v ∈ Q and w(u, v) < key [u]
then π[v] ← u
key [v] ← w(u, v)
```

Time Complexity

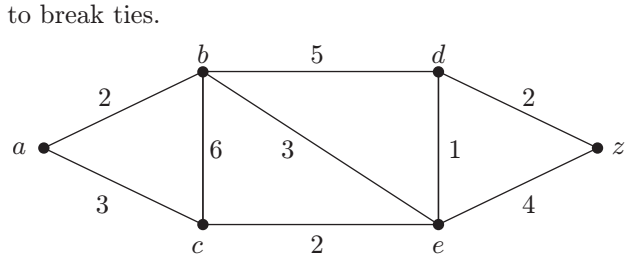
Prims algorithm use min heap data structure and its time complexity is O($V\log V+E\log V$).

Maximum edges in tree, $E = V - 1$

Therefore, O($V\log V + (V - 1)\log V$) = O($2V\log V - \log V$)

Thus, complexity = O($V\log V$)

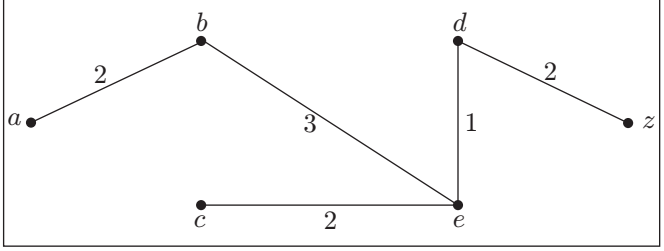
Problem 4.5: Use Prim's algorithm to find an MST in the following weighted graph. Use alphabetical order to break ties.



Solution: Prim's algorithm will proceed as follows.

1. First we add edge $\{d, e\}$ of weight 1.
2. Next, we add edge $\{c, e\}$ of weight 2.
3. Next, we add edge $\{d, z\}$ of weight 2.
4. Next, we add edge $\{b, e\}$ of weight 3.
5. And finally, we add edge $\{a, b\}$ of weight 2.

This produces a MST of weight 10.



4.7.4.2 Kruskal's Algorithm

One another algorithm to find the minimum spanning tree is Kruskal's. This method considers all minimum edges of graph one by one in increasing order and gives a resultant minimum spanning tree. In Kruskal's algorithm, it is not necessary for a tree to be connected.

Implementation steps:

1. Start from the source node.
2. Scan the weights and include the minimum weights to the tree in increasing order.
3. Stop if $N - 1$ edges are included.

Uses a disjoint-set data structure to determine whether an edge connects vertices in different components.

Data Structure

Before formalizing the above idea, let us quickly understand the disjoint-set data structure:

1. **Make-SET(v):** It creates a new set whose only member is pointed to by v. Note that for this operation, v must already be in a set.
2. **FIND-SET(v):** It returns a pointer to the set containing v.
3. **UNION(u, v):** It unites the dynamic sets that contain u and v into a new set that is a union of these two sets.

Algorithm

Starting with an empty set A, select the shortest edge that has not been chosen or rejected at every step, regardless of the position of this edge in the graph. The pseudocode for Kruskal algorithm is given:

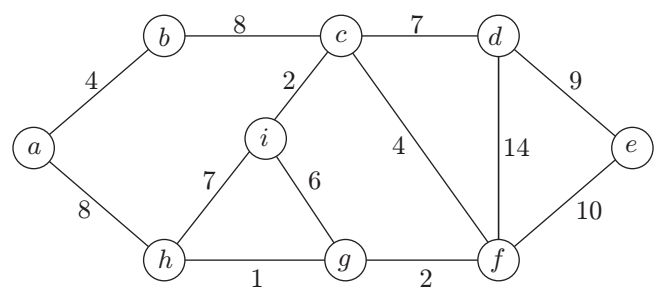
```
KRUSKAL(V, E, w)
A ← {φ} ▷ Set A will ultimately contain the
  edges of the MST
For each vertex v in V[G]
  Do MAKE-SET(v)
Sort E into non-decreasing order by weight w
For each edge (u, v)∈ E, taken in
  non-decreasing order by weight w
  do if FIND-SET(u)  ≠ FIND-SET(v)
  Then A ← A ∪ {(u, v)}
  UNION(u, v)
Return A
```

Time Complexity

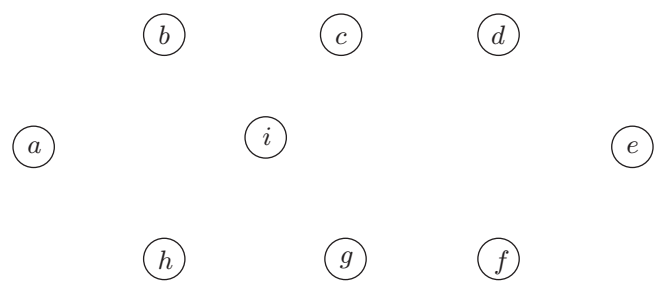
The edge weight can be compared in constant time. Initialization of priority queue takes $O(E \log E)$ time by repeated insertion. At each iteration of `while` loop, minimum edge can be removed in $O(\log E)$ time, which is $O(\log V)$, since the graph is simple. The total running time is $O((V + E) \log V)$, which is $O(E \log V)$ since the graph is simple and connected.

Example 4.18

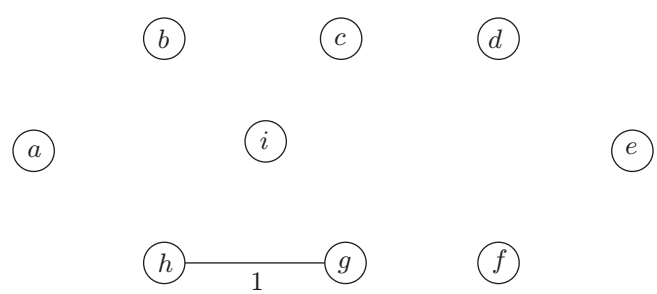
Let us use the Kruskal's algorithm to solve the following example.



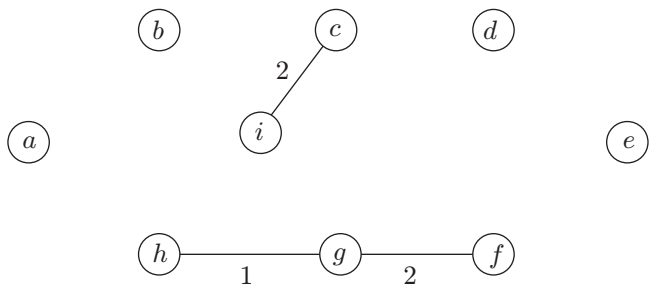
Step 1: Take only the vertices of graph.



Step 2: Choose the minimum weight edge from the original graph and draw that edge in disconnected graph containing n connected components as shown below:



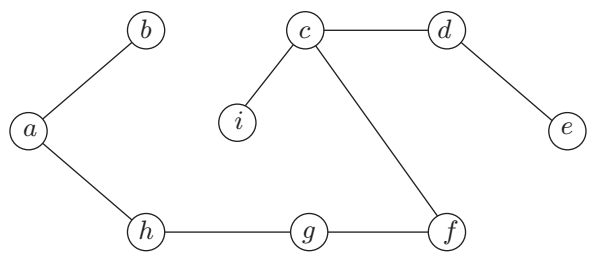
Step 3: Now select the next minimum weight edge and draw the corresponding edge in disconnected graph.



Here, there are two edges containing weight 2, add all possible edges of same weight until they form loop in the tree (since tree doesnot contain any loop but a graph may contain). That is why edge(g,f) and edge(i,c) both are added.

Step 4: Continue this process and check for the next smallest edge and add it into tree. Do this till we get $(n - 1)$ edges in the tree.

The final minimum spanning tree would be:



4.7.5 Single-Source Shortest Path

Given a weighted graph G, find the shortest path from a given vertex to every other vertex in G. Note that we can solve this problem quite easily with BFS traversal in the special case when all weights are equal. The Greedy approach to this problem is repeatedly selecting the best choice from those available at that time.

4.7.5.1 Dijkstra's Algorithm

Dijkstra's algorithm is a single source shortest path algorithm. It calculates the shortest distance starting from the start node till another node in the graph. While calculating the shortest distance, it excludes the longer distance paths. Dijkstra's algorithm does not operate on negative weights. For calculating shortest distance with negative edges, Bellman-Ford algorithm is used. Steps to perform Dijkstra's algorithm are as follows:

1. Assign all the nodes with distance infinity and 0 to initial node.Distance of start node s permanent and of all the other nodes is temporary.Set start node as active node.
2. Calculate the temporary distance of neighbours of active nodes. Distance is calculated by adding up the weights associated with those edges.
3. Update the distance, and set the current node an antecessor if such a calculated distance of a node is smaller as the current one. This step is also called update and is Dijkstra's central idea.
4. Set the node as active node which has minimal temporary distance. Mark the distance as permanent.
5. Repeat steps 2 to 4 until there are no nodes left with a permanent distance, whose neighbours still have temporary distances.

Dijkstra's Pseudocode

```
DIJKSTRA (G , S):
1.  for each vertex v in Graph:
2.  distance[v] := infinity
3.  previous[v] := undefined
4.  distance[ S ] := 0
5.  Q := the set of all nodes in Graph
6.  While Q is not empty:
7.  u := node in Q with smallest distance[ ]
8.  remove u from Q
9.  for each neighbour v of u:
10. alt := distance[u] + distance_between(u, v)
11. If alt < distance[v]
12. distance[v] := alt
13. previous[v] := u
14. Return previous[ ]
```

Time Complexity

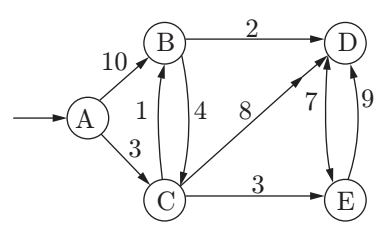
Like Prim's algorithm, Dijkstra's algorithm runs in $O(E \log V)$ time.

1. Dijkstra's algorithm with list: $O(V^2)$
2. Dijkstra's algorithm with modified binary heap: $O((E + V) \log V)$
3. Dijkstra's algorithm with Fibonacci heap: $O(E + V \log V)$

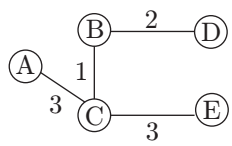
Example

The following are the steps:

1. Start from the source node; find the distance of all nodes reachable from the source.
2. Then pick the node which has the minimum distance, from the pool of nodes.
3. Calculate again the distance from the selected node, and follow the above two steps till the entire nodes are selected.



Source	A	B	C	D	E	
	∞	∞	∞	∞	∞	→ Cost
		Nil	Nil	Nil	Nil	→ Parent
	O	10	3	∞	∞	
→ A	Nil	A	A	Nil	Nil	
	O	4		11	6	
→ AC	Nil	C		C	C	
	O			6	6	
→ ACB	Nil			B	C	
	O				6	
→ ACBD	Nil				C	



4.7.5.2 Bellman–Ford Algorithm

The Bellman–Ford algorithm is used to compute single source shortest path to all other vertices of weighted directed graph. This algorithm also considers negative weights of graph edges. This algorithm is slower but is capable of even finding a negative weight cycle in graph.

Bellman-Ford Pseudocode

```
d[s] ← 0
for each v  ∈ V - {s}
do d[v]  ← ∞
for i  ← 1 to |V| - 1 do
for each edge (u, v) ∈ E do
if d[v] > d[u] + w(u, v) then
d[v]  ← d[u] + w(u, v)
π[v] ← u
for each edge (u, v) ∈ E
do if d[v] > d[u] + w(u, v)
then report that a negative-weight
    cycle exists
```

Time Complexity

The time complexity of Bellman-Ford algorithm is O(VE), where $E = V^2$, so the worst-case running time of Bellman–Ford algorithm is O(V^3).

4.8 GRAPH TRAVERSAL

Graph traversal means visiting all the nodes in a graph in some order. In graph traversal, some nodes may be visited more than once. This is because it is not necessary to know that the node has been explored before or not. The re-traversal becomes more for dense graph, which increase computation time. Tree traversal is a special case of graph traversal. The following are two graph traversal methods:

1. Breadth-first traversal (BFT)
2. Depth-first traversal (DFT)

4.8.1 Breadth-First Traversal

The breath-first traversal starts from the root node and then traverses all its neighbours. Then for each neighbour, its corresponding unvisited neighbour is processed. The pseudocode for BFT is given as follows:

```
BST (V)
1. visited(V) =1
2. add(V, Q)
3. while(Q is not empty)
4. u = delete(Q)
5. for all x adjacent to u
6. if( x is not visited)
7. visited(x) = 1
8. add(Q, x)
```

4.8.1.1 Time Complexity

The time complexity of BFT is O($V + E$), where V is the number of vertices in the graph and E is the number of edges in the graph.

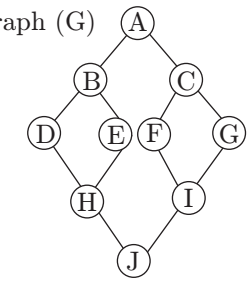
4.8.1.2 Application of BFT

The following are the applications of BFT. It can be used to find out

1. whether the given graph is connected.
2. the number of connected components in a graph.
3. whether the given graph contains a cycle.
4. the shortest path if all the edges have the same weight.
5. whether the given graph is a bipartite graph.

Problem 4.6: Consider the graph $G = (V, E)$ and find the BFT starting node A.

Graph (G)



Solution: A is the starting node and Q is a queue data structure to store the values.

BFT (A):

1. Visited (A) = 1
Add (Q, A)

A				

2. u = delete (Q) → A
For all adjacent nodes to A which are not already added in the queue, Add (Q, X), where X = (B, C) is the adjacent node to A.

B	**C**			

3. u = delete (Q) → B
For all adjacent nodes to B which are not already added in the queue, Add (Q, X), where X = (D, E) is the adjacent node to B.

C	**D**	**E**		

4. u = delete (Q) → C
For all adjacent nodes to C which are not already added in the queue, Add (Q, X), where X = (F, G) is the adjacent node to C.

D	**E**	**F**	**G**	

5. u = delete (Q) → D
For all adjacent nodes to D which are not already added in the queue, Add (Q, X), where X = (H) is the adjacent node to D.

E	**F**	**G**	**H**	

6. u = delete (Q) → E
For all adjacent nodes to E which are not already added in the queue, Add (Q, X), where X = (Φ) is the adjacent node to E.

F	**G**	**H**		

7. u = delete (Q) → F
For all adjacent nodes to F which are not already added in the queue, Add (Q, X), where X = (I) is the adjacent node to F.

G	**H**	**I**		

8. u = delete (Q) → G
For all adjacent nodes to G which are not already added in the queue, Add (Q, X), where X = (I) is the adjacent node to G.

H	**I**			

9. u = delete (Q) → H
For all adjacent nodes to H which are not already added in the queue, Add (Q, X), where X = (J) is the adjacent node to H.

I	**J**			

10. u = delete (Q) → I
For all adjacent nodes to I which are not already added in the queue, Add (Q, X), where X is the adjacent node to I.

J				

11. u = delete (Q) → J
For all adjacent nodes to J which are not already added in the queue, Add (Q, X), where X is the adjacent node to J.

12. The queue is empty, which means graph is traversed completely.

4.8.2 Depth-First Traversal

As already discussed, there are many ways of tree traversal. Consider the tree given in Fig. 4.10:

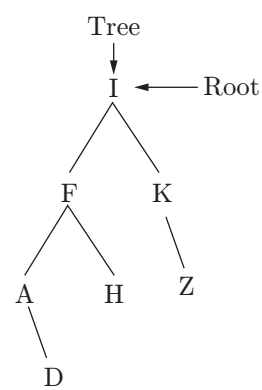


Figure 4.10 DFT of a tree.

Preorder traversal for given tree: I F A D H K Z

Preorder traversal also results in depth first traversal. This is because the traversal goes deeper before traversing its siblings. In this tree, the descendants of F, that is, A, H, D are traversed first and after that sibling of F, that is, K are traversed.

4.8.2.1 Pseudocode for DST

```
DFT (V)
1. visited (V) =1
2. for all x adjacent to V
3. if (x is not visited)
4. visited (x) =1
5. DFT (x)
```

The above is the recursive code for DFT.

4.8.2.2 Time Complexity

The time complexity of DFT is O($V + E$), where V is the number of vertices in the graph and E is the number of edges in the graph.

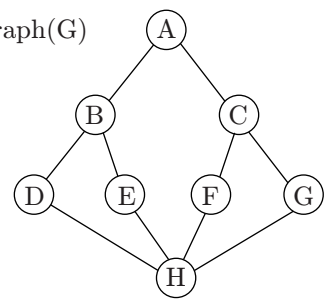
4.8.2.3 Application of DFT

The following are the applications of BFT. It can be used to find out

1. whether the given graph is connected.
2. the number of connected components in a graph.
3. whether the given graph contains a cycle.
4. whether the given directed graph is strongly connected.

Problem 4.7: Consider the graph $G = (V, E)$ and find the DFT starting node A.

Graph(G)



Solution: A is the starting node and S is a stack data structure to store the values. Stack is internally used by the recursive call.

`DFT (A)`

`1. Visited (A) =1`
`PUSH(S, A)`

A				

`2. POP(S) → A`
For all adjacent nodes to A which are not already added in the stack, `PUSH (S, X)`. Every neighbour element will be stored in the stack from right to left. Here, X = (B, C) is the adjacent node to A.

B	**C**			

`3. POP(S) → B`
For all adjacent nodes to B which are not already added in the stack, `PUSH (S, X)`. Every neighbour element will be stored in the stack from right to left. Here, X = (D, E) is the adjacent node to B.

D	**E**	**C**		

`4. POP (S) → D`
For all adjacent nodes to D which are not already added in the stack, `PUSH (S, X)`. Every neighbour element will be stored in the stack from right to left. Here, X = (H) is the adjacent node to D.

H	**E**	**C**		

`5. POP (S) → H`
For all adjacent nodes to H which are not already added in the stack, `PUSH (S, X)`. Every neighbour element will be stored in the stack from right to left. Here, X = (F, G) is the adjacent node to H.

F	**G**	**E**	**C**	

`6. POP (S) → F`
For all adjacent nodes to F which are not already added in the stack, `PUSH (S, X)`. Every neighbour element will be stored in the stack from right to left. Here, X = (Φ) is the adjacent node to F.

G	**E**	**C**		

`7. POP (S) → G`
For all adjacent nodes to G which are not already added in the stack, `PUSH (S, X)`. Every neighbour element will be stored in the stack from right to left. Here, X = (Φ) is the adjacent node to G.

E	**C**			

`8. POP (S) → E`
For all adjacent nodes to E which are not already added in the stack, `PUSH (S, X)`. Every neighbour element will be stored in the stack from right to left. Here, X = (Φ) is the adjacent node to E.

C				

`9. POP (S) C`
For all adjacent nodes to C which are not already added in the stack, `PUSH (S, X)`. Every neighbour element will be stored in the stack from right to left. Here, X = (Φ) is the adjacent node to C.

`10.` The stack is empty, which means the graph is traversed completely.

4.9 DYNAMIC PROGRAMMING

Dynamic programming is a method for solving complex problems by breaking them down into simpler sub-problems. It is applicable to problems exhibiting the properties of overlapping sub-problemsand optimal substructure.

Dynamic programming algorithms are used for optimization (e.g. finding the shortest path between two points, or the fastest way to multiply many matrices). A dynamic programming algorithm will examine all possible ways to solve the problem and will pick the best solution.

4.9.1 Applications of Dynamic Programming

The following are the applications of dynamic programming:

1. Fibonacci series
2. 0/1 knapsack
3. Largest common subsequence
4. All-pair shortest path

4.9.2 Fibonacci Series

Fibonacci series is the sequence produced by adding the previous two numbers in the sequence. First two numbers in sequences are 0 1 or 1 1. The recurrence relation to find the next term in the sequence is given below:

$$F_n = F_{n-1} + F_{n-2}$$

with initial values $F_1 = 0$ and $F_2 = 1$ or $F_1 = 1$ and $F_2 = 1$.

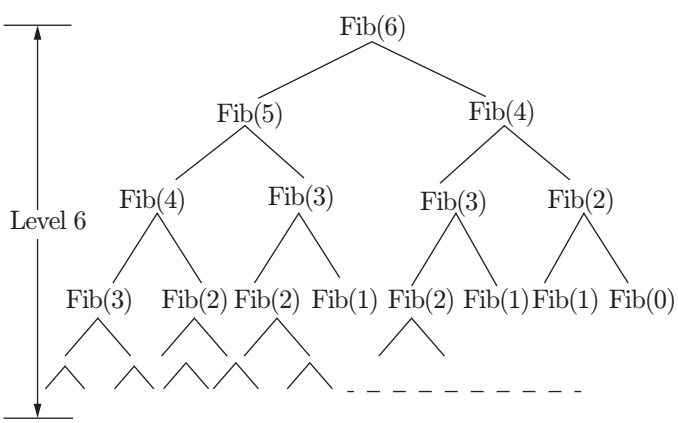
Example: Fibonacci series 0, 1, 1, 2, 3, 5, 8, 13, 21, ...

4.9.2.1 Recurrence Relation of Fibonacci

$$\text{fib}(n) = \begin{cases} 0, & \text{if } n < 0 \\ 1, & \text{if } n = 1 \\ \text{fib}(n-1) + \text{fib}(n-2), & \text{if } n > 1 \end{cases}$$

Example 4.19

The solution of Fibonacci through recursion tree is shown as follows:



Total number of nodes = $(2^n - 1)$.

So, the total number of function calls = 2^n, and one function call takes O(1) time.

The total running time = $O(2^n)$.

Note: Most of the recursive programs contain very less number of distinct calls and more number of repeated sub-problems or function calls. These problems are known as overlapping sub-problems.

4.9.2.2 Dynamic Solution of Fibonacci

In dynamic programing technique, we solve every problem exactly once and store the results in memory. That result is used later to get an optimum output (Fig. 4.11).

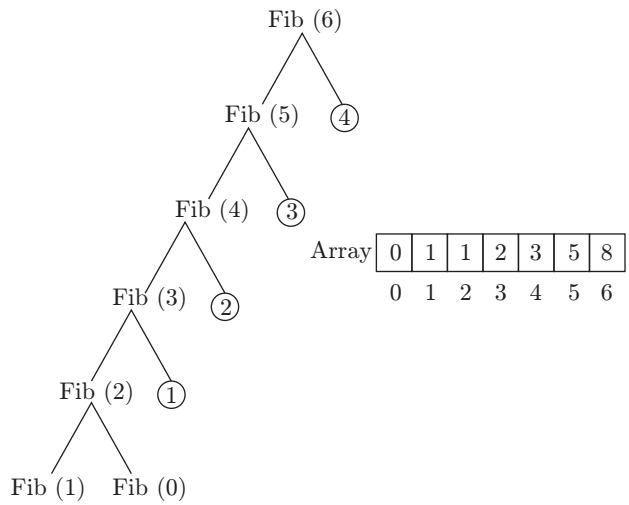


Figure 4.11 Dynamic programming approach for Fibonacci.

Pseudocode for Dynamic Solution of Fibonacci Series

```
1. Algorithm Fast-Fibonacci(n)
2. Let fib[0] and fib[1] be 1
3. for each i from 2 to n, do
4. Let fib[i] = fib[i - 2] + fib[i - 1]
5. end of loop
6. return fib[n]
```

Time Complexity

1. Fibonacci series without dynamic programing [space = $O(n)$ and time = $O(2^n)$]
2. Fibonacci series with dynamic programing [space = $O(n)$ and time = $O(n)$]

4.9.3 0/1 Knapsack

One of the problems in combinatorial optimization is knapsack problem. In this problem, the target is to maximize the profit by choosing items. Each item is associated with a weight and profit. The collection of items is formed such that the chosen total weight should be less than or equal to the given limit. The profit can be as large as possible.

4.9.3.1 Recurrence Relation 0/1 Knapsack

The recurrence relation of 0/1 Knapsack problem is

$$\mathrm{KS}(n,m) = \begin{cases} 0, & \text{if } m = 0 \text{ or } n = 0 \\ \mathrm{KS}(n-1,m) & \text{if } m < w[n] \\ \mathrm{Max}[\{\mathrm{KS}(n-1,m-w[n]+p[n])\} \\ \quad \{\mathrm{KS}(n-1,m)+0\}], & \text{otherwise} \end{cases}$$

4.9.3.2 Time Complexity

The time complexity of the knapsack problem is as follows:

1. Without dynamic programing = $O(2^n)$
2. With dynamic programing = $O(m^n)$

It is one of the NPC problems.

4.9.4 Longest Common Subsequence

A subsequence of a given sequence is just the given subsequence in which zero or more symbols are left out.

Let sequence $(S) = \{A, B, B, A, B, B\}$, subsequence $(S_1) = \{A, A\}$, subsequence $(S_2) = \{A, B, A, B\}$ and subsequence $(S_3) = \{B, A, B, A\}$.

4.9.4.1 Common Subsequence

In the given two sequences X and Y, we say that a sequence Z is a common subsequence of X and Y, if Z is a subsequence of both X and Y.

Example 4.20

$X = \{A, B, B, A, B, B\}$

$Y = \{B, A, A, B, A, A\}$

$Z = \{B, B, A\}$ is a common subsequence. But $Z = \{A, B, A, B\}$ is not a common subsequence.

4.9.4.2 Recurrence Relation of LCS

The recurrence relation of longest common subsequence is

$$\mathrm{LCS}(i,j) = \begin{cases} 0, & \text{if } i=0 \text{ or } j=0 \\ 1+\mathrm{LCS}(i-1,j-1), & \text{if } X[i]=Y[j] \\ \mathrm{Max}[\{\mathrm{LCS}(i-1,j)\}\{\mathrm{LCS}(i,j-1)\}], & \text{if } X[i]\neq Y[j] \end{cases}$$

4.9.4.3 Time Complexity

Complexity	LCS with Recursive Solution	LCS with Dynamic Programing
Time	If $(m = n)$ LCS $(m, n) = O(2^n)$ Else LCS$(m, n) = O(2^{m+n})$	$T(n) = O(m^n)$
Space	$O(m + n)$	$O(mn)$

4.10 ALL-PAIR SHORTEST PATH

Given a directed graph $G = (V, E)$, where each edge (v, w) has a non-negative cost $C[v, w]$, for all pairs of vertices (v, w), find the cost of the lowest-cost path from v to w. It is a generalization of the single-sourceshortest-path problem.

4.10.1 Floyd's Algorithm

Let $A^k(i, j)$ be the minimum cost required to go from vertex i to vertex j for which all intermediate vertices are in the set $(0, 1, 2, 3, \ldots, k)$ (Fig. 4.12).

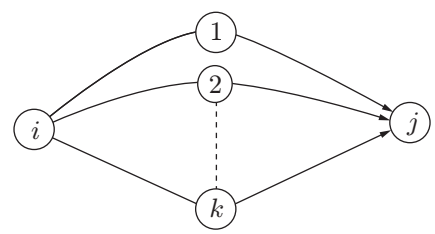


Figure 4.12 Different paths from i to j.

4.10.1.1 Recurrence Relation of Floyd's Algorithm

The recurrence relation of Floyd's algorithm is

$$A^k(i,j) = \mathrm{Min}\{A^{k-1}(i,j), A^{k-1}(i,k) + A^{k-1}(k,j)\}$$

4.10.1.2 Time Complexity

The running time complexity: $T(n) = O(V^3)$.

4.11 CONCEPTS OF COMPLEXITY CLASSES

4.11.1 P-Complexity Class

P in computational complexity theory is known as polynomial time or deterministic time. This is the most basic complexity class. This consists of all the problems that

are solvable in polynomial time. These problems can be solved in $O(n^k)$ time, where k is some constant. In class P, problems are efficiently solvable or traceable.

A language L is in class P iff there exists some Turing machine M such that:

1. Turing machine runs in polynomial time for all the inputs.
2. For all y in L, M must return 1, otherwise 0.

P includes many natural problems like linear programming, finding maximum matching and calculating the greatest common divisor. Prime number is also a part of class P now.

4.11.2 NP Complexity Class

NP class consists of problems that are verifiable in polynomial time. This class contains those languages that can be defined by second order logic.

Note: The most important open question in complexity theory is the P = NP problem (Fig. 4.13).

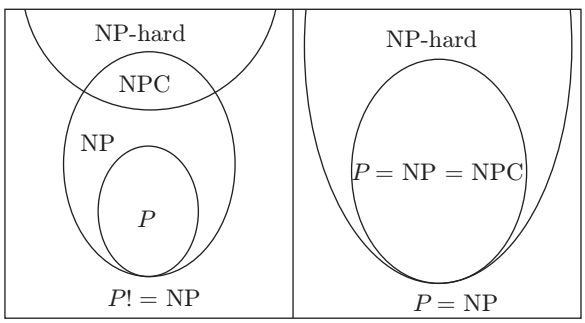


Figure 4.13 Relation between P, NP, NP-complete and NP-hard problems.

4.11.3 NP-Complete

These are the hardest problems in NP. In the computational complexity theory, the complexity class NP-complete (abbreviated NP-C or NPC) is a class of decision problems. A decision problem L is NP-complete if it is in the set of NP problems and also in the set of NP-hard problems.

NP-complete is a subset of NP, the set of all decision problems whose solutions can be verified in polynomial time; NP may be equivalently defined as the set of decision problems that can be solved in polynomial time on a non-deterministic TM. A problem p in NP is also NP-complete if every other problem in NP can be transformed into p in polynomial time.

The following list contains some well-known problems that are NP-complete when expressed as decision problems:

1. Boolean satisfiability problem (SAT)
2. Knapsack problem
3. Hamiltonian path problem
4. Travelling salesman problem
5. Subgraph isomorphism problem
6. Subset sum problem
7. Clique problem
8. Vertex cover problem
9. Independent set problem
10. Dominating set problem
11. Graph colouring problem

4.11.4 NP-Hard

NP-hard is a class that includes problems which are "at least as hard as hardest problem in NP". A problem is said to be NP hard if

1. It is in NP
2. It is NP-complete

This means a problem H is NP-hard if any problem L has polynomial time reduction from L $\rightarrow$ H. The problem can be a decision problem, optimization problem or a search problem.

Examples of NP-hard problem is the traveling salesman problem. It is an optimization problem of finding the least-cost cyclic route through all nodes of a weighted graph. There are decision problems that are NP-hard but not NP-complete, for example, the halting problem. This is the problem which asks, 'given a program and its input, will it run forever?' That is a yes/no question, so this is a decision problem.

Note:

1. Let 'A' be an NP-complete problem and 'B' be an unknown problem; if 'A' is a polynomial that reduces to 'B' ($A \leq_p B$), then 'B' is NP-hard.
2. Let 'A' be an NP-hard problem and 'B' be an unknown problem; if 'A' is a polynomial that reduces to 'B' ($A \leq_p B$), then 'B' is NP-hard.
3. Let 'A' be a P-class problem and 'B' be an unknown problem; if ($B \leq_p A$), then 'B' is a P-class problem.
4. Let 'A' be a P-class problem and 'B' be an unknown problem; if ($B \leq_p A$), then 'B' is an NP-class problem.

Problem 4.8: Both P and NP are closed under the operation of:

(A) Union (B) Intersection
(C) Kleene's closure (D) None of these

Solution: (D) Both P and NP are closed under concatenation and Kleene's closure. The Kleene closure of a language L, denoted by L^* is defined as

$$\{w_1 w_2 \ldots w_k - k \in N,\ w_i \in L \forall i = 1, 2, \ldots, k\}.$$

P is closed under Kleene's closure as for all $L \in P$, $L^* \in P$.

NP is closed under Kleene's closure as for all $L \in$ NP, $L^* \in$ NP.

IMPORTANT FORMULAS

1. Theta notation (Θ): $0 < c_1(g(n)) \le f(n) \le c_2(g(n))$ $\forall n \ge n_0$
2. Big-O notation (O): $0 < f(n) \le c(g(n)) \quad \forall n \ge n_0$.
3. Omega notation (Ω): $0 < c(g(n)) \le f(n) \quad \forall n \ge n_0$.
4. Small-o notation (o): $0 < f(n) < c(g(n)) \quad \forall n \ge n_0$.
5. Small omega notation (ω): $0 < c(g(n)) < f(n) \ \forall n \ge n_0$.
6. Master theorem: $T(n) = aT(n/b) + f(n)$.

   Case 1: $f(n) = O(n^{\log_b a - \varepsilon})$, then $T(n) = \Theta(n^{\log_b a})$.

   Case 2: $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.

   Case 3: $f(n) = \Omega(n^{\log_b a + \varepsilon})$.

   $af(n/b) \le cf(n)$ for some constant $c < 1$ and large n, then $T(n) = \Theta(f(n))$.
7. Linear search: $O(n)$.
8. Binary search: $\Theta(\log n)$.
9. Bubble sort: $O(n^2)$.
10. Selection sort: $O(n^2)$.
11. Insertion sort: $O(n^2)$.
12. Heap sort: $O(n \log n)$.
13. Merge sort: $O(n \log n)$.
14. Quick sort: $O(n^2)$ (worst case).
15. Quick sort: $O(n \log n)$ (average case).
16. Counting sort: $O(n + k)$.
17. If a graph has n number of vertices and it is k-regular, where k is a positive constant, then the number of edges $= (n \times k)/2$.
18. A complete graph having n vertices has $[n(n-1)]/2$ edges.
19. Number of simple graphs possible with n vertices $= 2^{[n(n-1)]/2}$.
20. Prim's algorithm: $O(E \log V)$.
21. Kruskal's algorithm: $O((V + E) \log V)$.
22. Dijkstra's algorithm: $O(E \log V)$.
23. Bellman–Ford algorithm: $O(V^3)$.
24. BFT and DFT: $O(V + E)$.

SOLVED EXAMPLES

1. Consider $T(n) = 6T(n/2) + n^3$; then $T(n)$ is equal to

   (a) $(n/2)$ (b) $O(n^3 \log n)$
   (c) $(n^2 \log n)$ (d) $O(n^3)$

   Solution:
   $T(n) = 6T(n/2) + n^3$; By using Master Theorem.
   $a = 6,\ b = 2,\ f(n) = n^3$
   $g(n) = (n^{\log_b a}) = n^{\log_2 6} = \Theta(n^{2.58})$
   $f(n) = \Omega(n^{\log_2 6 + \varepsilon})$, $f(n)$ is larger than $g(n)$, but is not a larger polynomial.
   By case 3, $T(n) = \Theta(f(n)) = n^3$

   Ans. (d)

2. Solve the following recurrence relation: $T(n) = 4T(n/2) + n$.

   (a) $O(n^2 \log n)$ (b) $O(n^2)$
   (c) $O(n/2)$ (d) $O(\log n)$

   Solution:
   $T(n) = 4T(n/2) + n$; By using Master Theorem.
   $a = 4,\ b = 2,\ f(n) = n$
   $g(n) = (n^{\log_b a}) = n^{\log_2 4} = \Theta(n^2)$
   $f(n) = O(n^{\log_2 4 - \varepsilon})$, $f(n)$ is smaller than $g(n)$, but is not a smaller polynomial.
   By case 1, $T(n) = \Theta(n^2)$

   Ans. (b)

3. Consider the following statements:

   1. An algorithm is the number of steps to be performed to solve a problem.
   2. An algorithm is the number of steps and their implementation in any language to a given problem to solve a problem.
   3. To solve a given problem there may be more than one algorithm.

Which of the following is true?

(a) 1 is correct. (b) 2 is correct.
(c) 1 and 3 are correct. (d) 2 and 3 are correct.

Solution: Both statements 1 and 3 are correct.

Ans. (c)

4. Compilers require efficient searching strategies for which they rely on

(a) Hash tables (b) String matching
(c) Binary search tables (d) Binary search trees

Solution: Hash tables have less complexity, so they are used by the compiler for efficient searching.

Ans. (a)

5. Consider a binary search tree having n elements. The time required to search a given element will be:

(a) $O(n)$ (b) $O(\log n)$
(c) $O(n^2)$ (d) $O(n \log n)$

Solution: The time required for search operation in a binary search depends on the height of the tree. In a complete binary tree with n nodes, search operation takes $\Theta(\log_2 n)$ in worst case scenario. But if all the nodes of tree are in linear chain, then the same operations will take $O(n)$ in worst-case scenario. So, option (a) is correct.

Ans. (a)

6. The average time required to perform a successful sequential search for an element in an array $A(1: n)$ is given by

(a) $\log n$ (b) n^2
(c) $(n + 1)/2$ (d) $(n \times n - 1/2)$

Solution: The average time required for sequential search or linear search in this case is $(n + 1)/2$.

Ans. (c)

7. Which sort will operate in quadratic time relative to the number of elements in the array (on the average)?

(a) Heap sort (b) Bubble sort
(c) Radix sort (d) Merge sort

Solution: The average time for bubble sort is $O(n^2)$, so it will operate in quadratic time.

Ans. (b)

8. Binary expression tree is traversed in _____ traversal.

(a) Postorder (b) Preorder
(c) Both A and B (d) Reverse polish order

Solution: Binary tree is traversed in both preorder as well as postorder traversal.

Ans. (c)

9. The time complexity of searching an element in a linked list of length n will be

(a) $O(n)$ (b) $O(n^2 - 1)$
(c) $O(\log n)$ (d) $O(\log n^2)$

Solution: The worst-case time complexity for searching an element in a linked list is $O(n)$, so option (a) is correct. If in the question it is not mentioned about (best or worst or average) time complexity then we always calculate as worst time complexity.

Ans. (a)

10. A hashing algorithm uses linear probing; the average search time will be less if the load factor

(a) is less than one (b) equals one
(c) is greater than one (d) equals two

Solution: As load time is the ratio of number of records present to the total number of records, so is load time is less than 1, the probability of collisions decreases and hence, search time reduces.

Ans. (a)

11. If a node has K children in B tree, then the node contains exactly _____ keys.

(a) K^2 (b) $K - 1$ (c) $K + 1$ (d) $\sqrt{K}$

Solution: The number of keys $= K - 1$.

Ans. (b)

12. Big-O estimate for the factorial function, that is, $n!$, is given by

(a) $O(n!)$ (b) $O(n^n)$
(c) $O(n \times n!)$ (d) $O(\sqrt{n})$

Solution: Factorial function $f(n) = n! = n(n-1) \times (n-2) \ldots 1$

$n! = n\,(n-1)(n-2) \ldots 1 \leq n \times n \ldots \times n = (n^n)$

So, Big O Estimation of $f(n!) = O(n^n)$

Ans. (b)

13. Let A and B be two $n \times n$ matrices. The efficient algorithm to multiply the two matrices has the time complexity $O(n^x)$, where x is _____.

Solution: Strassen's surprising algorithm can multiply two $n \times n$ matrices in $\Theta(n^{\log 7})$ time, which is equal to $O(n^{2.81})$ time.

Ans. (2.81)

14. The searching technique in which less number of comparisons are done is

(a) Binary searching
(b) Linear search searching
(c) Hashing
(d) Woline's search

Solution:

Search Complexity	Average Case	Worst Case
Binary	O(log n)	O(n)
Hash table	O(1)	O(n)
Linear search	O(n)	O(n)

Thus, hash table takes less search time.

Ans. (c)

15. Which of the following statements is false about Prim's algorithm?

(a) The time complexity is O(E log V) using a binary heap.
(b) It may use binomial max-heap to represent the priority queue.
(c) The time complexity is O($E + V$ log V) using a Fibonacci heap.
(d) Initially the roots key and nodes are set to zero.

Solution:
(a) True, The time complexity is O(E log V) using a binary heap.
(c) True, Prim's algorithm's time complexity is O($E + V$ log V) using a Fibonacci heap.
(d) True, Initial distance to the roots key and nodes is kept Zero.

Only option (b) is false.

Ans. (b)

16. Express the following recurrence relation in asymptotic notations:

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

(a) $\Theta(n \log n)$ (b) $\Theta(n \log^2 n)$
(c) $\Theta(n^2)$ (d) $O(n^3)$

Solution: By using Master Theorem,
$T(n) = 2T(n/2) + n$

$a = 2,\ b = 2,\ f(n) = n$

$g(n) = (n^{\log_b a}) = n^{\log_2 2} = \Theta(n)$

Order of $g(n)$ = order of $f(n)$, so by case 2, $T(n) = \Theta(f(n) \cdot \log n) = \Theta(n \log n)$

Ans. (a)

17. What is the height of n-node, k-ary heap?

(a) $\Theta(\log_k n)$ (b) $\Theta(nk)$
(c) $\Theta(k \log_2 n)$ (d) $\Theta(n/k)$

Solution: If h is the height of k-ary heap ($h \geq 2$) H, then the number of nodes 'n' in H such that

$$\frac{k(h)-1}{k-1} < n \leq \frac{k(h+1)-1}{k-1}$$

So $h = \Theta(\log_k n)$.

Ans. (a)

18. Solve the following given recurrence relation: $T(n) = 4T(n/2) + n^2$.

(a) $\Theta(n^2)$ (b) $\Theta(n^2 \log_2 n)$
(c) $\Theta(n \log_2 n)$ (d) None of the above

Solution: Using the master theorem, we have $a = 4$, $b = 2$ and $f(n) = n^2$. So

$$g(n) = n^{\log b \cdot a} = n^{\log 2 \cdot 4} = n^2$$

Here, order of $f(n)$ = order of $g(n)$, so we get

$$T(n) = O(n^{\log b \cdot a} \log_2 n) = O(n^2 \log_2 n)$$

Ans. (b)

19. Solve the following given recurrence relation: $T(n) = 16T(n/4) + n^3$.

(a) $\Theta(n \log_2 n)$ (b) $\Theta(n^2 \log_2 n)$
(c) $\Theta(n^3)$ (d) None of the above

Solution: From the master theorem, we have $a = 16$, $b = 4$ and $f(n) = n^2$. So

$$g(n) = n^{\log b \cdot a} = n^{\log 4 \cdot 16} = n^2$$

Order of $f(n) >$ order of $g(n)$, so we get

$$T(n) = O(f(n))$$

Ans. (c)

20. Consider the following two functions:

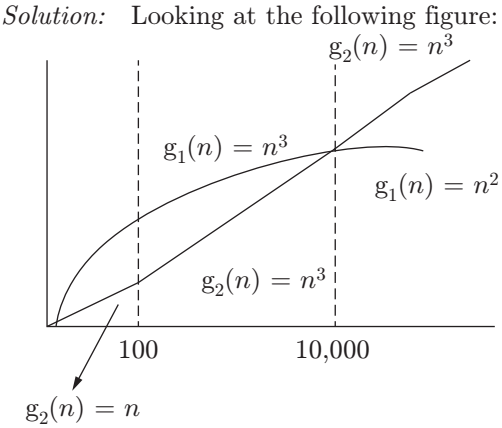
$$g_1(n) = \begin{cases} n^3, & \text{for } 0 \leq n \leq 10{,}000 \\ n^3, & \text{for } 0 \leq n \geq 10{,}000 \end{cases}$$

$$g_2(n) = \begin{cases} n, & \text{for } 0 \leq n \leq 10{,}000 \\ n^3, & \text{for } n > 10{,}000 \end{cases}$$

Which of the following is true?

(a) $g_1(n)$ is $O(g_2(n))$. (b) $g_1(n)$ is $O(n^3)$.
(c) $g_2(n)$ is $O(g_1(n))$. (d) $g_2(n)$ is $O(n)$.

Solution: Looking at the following figure:



Therefore, we get

$$n^2 \leq n^3 \quad \text{for } N \geq 10{,}000$$

$$g_1(n) = O(g_2(n))$$

Ans. (a)

GATE PREVIOUS YEARS' QUESTIONS

1. Consider the following three claims:

(I) $(n+k)^m = \Theta(n^m)$, where k and m are constants
(II) $2^{n+1} = O(2^n)$
(III) $2^{2n+1} = O(2^n)$

Which of these claims is correct?

(a) I and II (b) I and III
(c) II and III (d) I, II and III

(GATE 2003: 1 Mark)

Solution: Option I: Consider $k = 1$; the expansion of $(n+1)^m = f(n)$.
Option II: $2^{n+1} = 2.2^n = O(2^n)$.
Option III: $22^{n+1} = 2.2^{2n} = O(2^{2n})$.
Options I and II are correct.

Ans. (a)

2. Let G be an arbitrary graph with n nodes and k components. If a vertex is removed from G, the number of components in the resultant graph must necessarily lie between

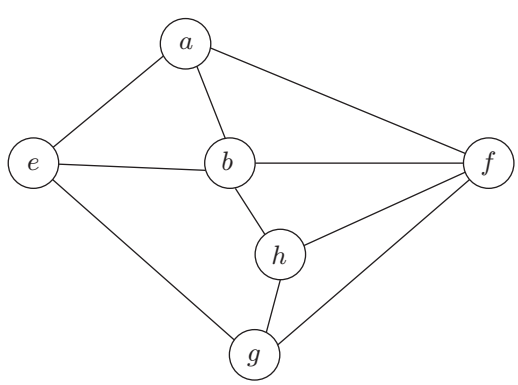
(a) k and n (b) $k-1$ and $k+1$
(c) $k-1$ and $n-1$ (d) $k+1$ and $n-k$

Solution: Minimum components comes out to be when the independent vertex, which itself is a component, is removed. In that case, the number of components left is $k-1$.

The maximum range comes when the removed vertex isolates all the vertices, In that case, the number of components comes out to be $n-1$.

Ans. (c)

3. Consider the following graph:



Among the following sequences:

(I) a b e g h f (II) a b f e h g
(III) a b f h g e (IV) a f g h b e

Which are depth-first traversals of the above graph?

(a) I, II and IV only (b) I and IV only
(c) II, III and IV only (d) I, III and IV only

(GATE 2003: 1 Mark)

Solution: Option II is incorrect. There is no edge between f and e.

Ans. (d)

4. The usual $\Theta(n^2)$ implementation of insertion sort to sort an array uses linear search to identify the position where an element is to be inserted into the already sorted part of the array. If, instead, we use binary search to identify the position, the worst-case running time will

(a) remain $\Theta(n^2)$ (b) become $\Theta(n(\log n)^2)$
(c) become $\Theta(n \log n)$ (d) become $\Theta(n)$

(GATE 2003: 1 Mark)

Solution: Worst-case complexity will remain $O(n^2)$. This is because binary search is used to locate the position of element will take $log_2(n)$ time, but inserting an element at its desired position will take $O(n)$ time in worst case.

Ans. (a)

5. In a heap with n elements with the smallest element at the root, the 7th smallest element can be found in time

(a) $\Theta(n \log n)$ (b) $\Theta(n)$
(c) $\Theta(\log n)$ (d) $\Theta(1)$

(GATE 2003: 1 Mark)

Solution: Minimum element as root means it is min heap. Searching 7th minimum element takes $O(\log n)$ time.

Ans. (c)

6. It has not been proved yet that $P = NP$.

Consider the language L defined as follows:

$$L = \begin{cases} (0+1)^* & \text{if } P = \text{NP} \\ \phi & \text{otherwise} \end{cases}$$

Which of the following statements is true?

(a) L is recursive.
(b) L is recursively enumerable but not recursive.
(c) L is not recursively enumerable.
(d) Whether L is recursive or not will be known after we find out if $P = \text{NP}$.

(GATE 2003: 1 Mark)

Solution: If a language L belongs to class P and class NP then it is called recursive.

Ans. (a)

7. A graph $G = (V, E)$ satisfies $|E| \leq 3\ |V| - 6$. The min-degree of G is defined as

$$\min_{v \in V}\{\text{degree}(v)\}$$

Therefore, min-degree of G cannot be

(a) 3 (b) 4 (c) 5 (d) 6

(GATE 2003: 2 Marks)

Solution: Substituting value of $E = V(V-1)/2$, we get $V = 3$ or 4. Thus, the minimum degree of each vertex could be 1 or 2. Hence, 3 cannot be the min degree.

Ans. (a)

8. Consider two languages L_1 and L_2, each on the alphabet $\sum$. Let f: $\sum \rightarrow \sum$ be a polynomial time computable bijection such that $(\forall\ x)[x \in L_1$ iff $f(x) \in L_2]$. Further, let f^{-1} also be polynomial time computable.

Which of the following CANNOT be true?

(A) $L_1 \in$ P and L_2 is finite
(B) $L_1 \in$ NP and $L_2 \in$ P
(C) L_1 is undecidable and L_2 is decidable
(D) L_1 is recursively enumerable and L_2 is recursive

(GATE 2003: 2 Marks)

Solution: L_1 is undecidable and L_2 is decidable are contradictory, as $L_1 \leq_p L_2$. Thus, if L_2 is decidable $\Rightarrow L_1$ is decidable.

Ans. (c)

Common Data Questions 8 and 9: In a permutation $a_1 \ldots a_n$ of n distinct integers, an inversion is a pair (a_i, a_j) such that $i < j$ and $a_i > a_j$.

9. If all permutations are equally likely, what is the expected number of inversions in a randomly chosen permutation of $1 \ldots n$?

(a) $\dfrac{n(n-1)}{2}$ (b) $\dfrac{n(n-1)}{4}$

(c) $\dfrac{n(n+1)}{4}$ (d) $2n[\log_2 n]$

(GATE 2003: 2 Marks)

Solution: Inversion means $i < j$ and $a_i > a_j$ in pair (a_i, a_j).

Average number of inversions $= \dfrac{1}{2}({}^n\mathrm{C}_2) = \dfrac{n(n-1)}{2}$

Ans. (b)

10. What would be the worst-case time complexity of the insertion sort algorithm, if the inputs are restricted to permutations of $1 \ldots n$ with at most n inversions?

(a) $\Theta(n^2)$ (b) $\Theta(n \log n)$

(c) $\Theta(n^{1.5})$ (d) $\Theta(n)$

(GATE 2003: 2 Marks)

Solution: With at most n inversions, insertion sort takes $\log n$ time. So, if there are n elements, complexity will be $\mathrm{O}(n \log n)$.

Ans. (a)

11. The cube root of a natural number n is defined as the largest natural number m such that $m^3 \leq n$. The complexity of computing the cube root of n (n is represented in binary notation) is

(a) $\mathrm{O}(n)$ but not $\mathrm{O}(n^{0.5})$
(b) $\mathrm{O}(n^{0.5})$ but not $\mathrm{O}((\log n)^k)$ for any constant $k > 0$
(c) $\mathrm{O}((\log n)^k)$ for some constant $k > 0$ but not $\mathrm{O}((\log \log n)^m)$ for any constant $m > 0$
(d) $\mathrm{O}((\log \log n)^k)$ for some constant $k > 0.5$ but not $\mathrm{O}((\log \log n)^{0.5})$

(GATE 2003: 2 Marks)

Solution: The complexity of computing the cube root of n is $\mathrm{O}(K^3\ n^2 \log(\mathrm{B}))$. Using binary search it takes $\mathrm{O}((\log n)^k)$ for some constant $k > 0$ but not $\mathrm{O}((\log n)^m$ for any constant $m > 0$.

Ans. (c)

12. Let $G = (V, E)$ be an undirected graph with a subgraph $G_1 = (V_1, E_1)$. Weights are assigned to edges of G as follows:

$$w(e) = \begin{cases} 0 & \text{if } e \in E_1 \\ 1 & \text{otherwise} \end{cases}$$

A single-source shortest path algorithm is executed on the weighted graph $(V, E, w)$with an arbitrary vertex v_1 of V_1 as the source. Which of the following can always be inferred from the path costs computed?

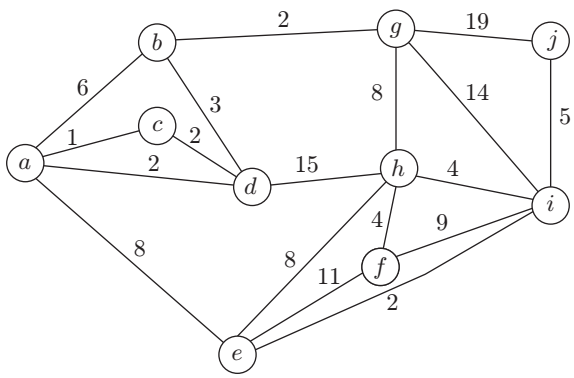
(a) Number of edges in the shortest paths from v_1 to all vertices of G.
(b) G_1 is connected.
(c) V_1 forms a clique in G.
(d) G_1 is a tree.

(GATE 2003: 2 Marks)

Solution: Let us take four vertices v_1, v_2, v_3 and v_4. The vertex v_1 is in G_1 such that v_1 is adjacent to v_3 and v_2 is adjacent to v_4. Now after apply shortest path algorithm we get $v_1 v_2 = 1$, $v_1 v_4 = 1$ and $v_1 v_3 = 0$. From these results, we can say that v_1 is not a clique in G_1.

Ans. (b)

13. What is the weight of a minimum spanning tree of the following graph?



(a) 29 (b) 31 (c) 38 (d) 41

(GATE 2003: 2 Marks)

Solution: A minimum spanning tree will contain $n - 1$ edges. Here, $10 - 1 = 9$ edges will be included. Total weight = 1 + 2 + 2 + 2 + 3 + 4 + 4 + 5 + 8 = 31.

Ans. (b)

14. The following are the starting and ending times of activities A, B, C, D, E, F, G and H, respectively, in chronological order:

'a_s b_s a_s a_e d_s a_e e_s f_s b_e d_e g_s e_e f_e h_s g_e h_e'

Here, x_s denotes the starting time and x_e denotes the ending time of activity X. We need to schedule the activities in a set of rooms available to us. An activity can be scheduled in a room only if the room is reserved for the activity for its entire duration. What is the minimum number of rooms required?

(a) 3 (b) 4 (c) 5 (d) 6

(GATE 2003: 2 Marks)

Solution: We have 'a_s b_s c_s a_e d_s c_e e_s f_s b_e d_e g_s e_e f_e h_s g_e h_e'. For this, the minimum number of rooms required is 4.

a_s	b_s	c_s	
d_s	b_s	c_s	
d_s	b_s	e_s	f_s
		e_s	f_s
g_s		e_s	f_s
g_s			
g_s	h_s		

Ans. (b)

15. Let $G = (V, E)$ be a directed graph with n vertices. A path from v_i to v_j in G is a sequence of vertices $(v_i, v_{i+1}, \ldots, v_j)$ such that $(v_k, v_{k+1}) \in E$ for all k in i through $j - 1$. A simple path is a path in which no vertex appears more than once.

Let A be an $n \times n$ array initialized as follows:

$$A[j,k] = \begin{cases} 1 & \text{if } (j,k) \in E \\ 0 & \text{otherwise} \end{cases}$$

Consider the following algorithm:

```
for i = 1 to n
for j = 1 to n
for k = 1 to n
A [j, k] = max (A[j, k] (A[j, i]+ A [i, k]);
```

Which of the following statements is necessarily true for all j and k after terminal of the above algorithm?

(a) $A[j, k] \leq n$.
(b) If $A[j, j] \geq n - 1$, then G has a Hamiltonian cycle.
(c) If there exists a path from j to k, $A[j, k]$ contains the longest path lens from j to k.
(d) If there exists a path from j to k, every simple path from j to k contain most $A[j, k]$ edges.

(GATE 2003: 2 Marks)

Solution: If there exists a path from j to k, every simple path from j to k, it must contain $A[j, k]$ edges.

Ans. (d)

16. Consider the label sequences obtained by the following pairs of traversals on a labelled binary tree:

(i) Preorder and postorder
(ii) Inorder and postorder
(iii) Preorder and inorder
(iv) Level order and postorder

Which of these pairs identifies a tree uniquely?

(a) (i) only (b) (ii) and (iii)
(c) (iii) only (d) (iv) only

(GATE 2004: 2 Marks)

Solution: Only if we have inorder traversal with any one of postorder or preorder traversal of tree, can we uniquely identify a tree.

Ans. (b)

17. Two matrices M_1 and M_2 are to be stored in arrays A and B, respectively. Each array can be stored in either row-major or column-major order in contiguous memory locations. The time complexity of an algorithm to compute M_1M_2 will be

(a) best if A is in row-major and B is in column-major order.
(b) best if both are in row-major order.
(c) best if both are in column-major order.
(d) independent of the storage scheme.

(GATE 2004: 2 Marks)

Solution: Complexity of matrices multiplication is irrespective of storage scheme of the arrays.

Ans. (d)

18. Suppose each set is represented as a linked list with elements in an arbitrary order. Which of the operations among union, intersection, membership and cardinality will be the slowest?

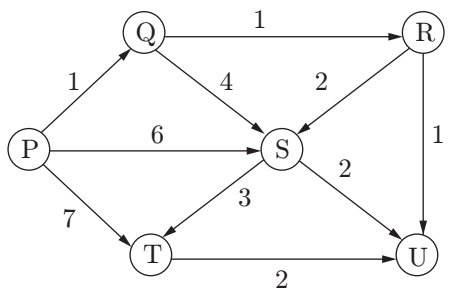
(a) Union only
(b) Intersection and membership
(c) Membership and cardinality
(d) Union and intersection

(GATE 2004: 2 Marks)

Solution: To calculate cardinality just counts the number of nodes in a list and for membership, just traverse the list and look for a match. So, cardinality and membership is not the slowest.
But For getting intersection and union of L_1 and L_2, we have to perform search in both list L_1 and L_2. So, they are the slowest one in the given list.

Ans. (d)

19. Suppose we run Dijkstra's single-source shortest-path algorithm on the following edge-weighted directed graph with vertex P as the source:



In what order do the nodes get included into the set of vertices for which the shortest path distances are finalized?

(a) P, Q, R, S, T, U (b) P, Q, R, U, S, T
(c) P, Q, R, U, T, S (d) P, Q, T, R, U, S

(GATE 2004: 2 Marks)

Solution: Nodes with minimum weight are included first. The sequence will be P, Q, R, U, S, T.

Ans. (b)

20. Let $\widehat{A}[1,\ldots,n]$ be an array storing a bit (1 or 0) at each location, and $f(m)$ is a function whose time complexity is $\Theta(m)$.

Consider the following program fragment written in a C-like language:

```
counter = 0;
for (i = 1 ; i < = n; i++)
{if (A [i] == 1) counter++;
else {f (counter); counter = 0;}
}
```

The complexity of this program fragment is
(a) $\Omega(n^2)$ (b) $\Omega(n \log n)$ and $O(n^2)$
(c) $\Theta(n)$ (d) $o(n)$

(GATE 2004: 2 Marks)

Solution: The maximum time taken by a `for` loop is n. So, the complexity is $\Theta(n)$.

Ans. (c)

21. The time complexity of the following C function is (assume $n > 0$):

```
int recursive (int n) {
if (n==1)
return (1);
else
return (recursive (n - 1) + recursive (n - 1));
}
```

(a) $O(n)$ (b) $O(n \log n)$
(c) $O(n^2)$ (d) $O(2^n)$

(GATE 2004: 2 Marks)

Solution: We have

$$T(n) = 2T(n-1), \quad n > 1$$
$$T(2) = 2T(1) = 2^1$$
$$T(3) = 2T(2) = 2^2$$
$$\ldots$$
$$T(n) = 2^{n-1} = O(2^n)$$

Ans. (d)

22. The recurrence equation

$$T(1) = 1$$
$$T(n) = 2T(n-1) + n, \quad n \geq 2$$

evaluates to:

(a) $2^{n+1} - n - 2$ (b) $2^n - n$
(c) $2^{n+1} - 2n - 2$ (d) $2^n + n$

(GATE 2004: 2 Marks)

Solution: We have

$$T(n) = 2T(n-1) + n, \quad n \geq 2$$
$$T(2) = 2T(1) + 2 = 4$$
$$T(3) = 2T(2) + 3 = 11$$
$$T(4) = 2T(3) + 4 = 26$$
$$\ldots$$
$$T(n) = 2^{n+1} - n - 2$$

Ans. (a)

23. Let $G_1 = (V, E_1)$ and $G_2 = V, E_2$ be connected graphs on the same vertex set V with more than two vertices. If $G_1 \cap G_2 = (V, E_1 \cap E_2)$ is not a connected graph, then the graph $G_1 \cup G_2 = (V, E_1 \cup E_2)$

(a) cannot have a cut vertex.
(b) must have a cycle.
(c) must have a cut-edge (bridge).
(d) has chromatic number strictly greater than those of G_1 and G_2.

(GATE 2004: 2 Marks)

Solution: Option (b) is the most appropriate option.

Ans. (b)

24. A program takes as input a balanced binary search tree with n leaf nodes and computes the value of a function $g(x)$ for each node x. If the cost of computing $g(x)$ is min{no. of leaf-nodes in left-subtree of x, number of leaf-nodes in right-subtree of x}, then the worst-case time complexity of the program is

(a) $O(n)$ (b) $O(n \log n)$
(c) $O(n^2)$ (d) $O(n^2 \log n)$

(GATE 2004: 2 Marks)

Solution: The complexity of $g(x)$ will be $O(n)$, using balanced binary tree.

Ans. (a)

25. The time complexity of computing the transitive closure of a binary relation on a set of n elements is known to be:

(a) $O(n)$ (b) $O(n \log n)$
(c) $O(n^{3/2})$ (d) $O(n^3)$

(GATE 2005: 1 Mark)

Solution: The transitive closure of a binary relation can be computed using Warshall's algorithm. The complexity of finding it is $O(n^3)$.

Ans. (d)

26. Suppose $T(n) = 2T(n/2) + n$, $T(0) = T(1) = 1$.

Which one of the following is false?

(a) $T(n) = O(n^2)$ (b) $T(n) = \Theta(n \log n)$
(c) $T(n) = \Omega(n^2)$ (d) $T(n) = O(n \log n)$

(GATE 2005: 2 Marks)

Solution: This can be solved using the master theorem.

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

Here we have $a = 2$, $b = 2$ and $f(n) = n$. So, it is the second case of master theorem. Therefore,

$$T(n) = \Theta(n^{\log_b a} \log n)$$

$$T(n) = \Theta(n \log n)$$

Options (a), (b) and (d) are true because options (a) and (d) are indicating the upper bound which is greater than $\Theta(n \log n)$. But, option (c) is false which indicates that the lower bound is n^2.

Ans. (c)

27. An undirected graph G has n nodes. Its adjacency matrix is given by an $n \times n$ square matrix whose (i) diagonal elements are 0s and (ii) non-diagonal elements are 1s. Which one of the following is *true*?

(a) Graph G has no minimum spanning tree (MST).
(b) Graph G has a unique MST of cost $n - 1$.
(c) Graph G has multiple distinct MSTs, each of cost $n - 1$.
(d) Graph G has multiple spanning trees of different costs.

(GATE 2005: 2 Marks)

Solution: Since diagonal elements are 0, there is no self-loop. And every node is connected to other $n - 1$ nodes.

Ans. (c)

28. Let $G(V,E)$ be an undirected graph with positive edge weights. Dijkstra's single source shortest path algorithm can be implemented using the binary heap data structure with time complexity:

(a) $O(|V|^2)$ (b) $O(|E| + |V| \log |V|)$
(c) $O(|V| \log |V|)$ (d) $O((|E| + |V|)\log |V|)$

(GATE 2005: 2 Marks)

Solution: Complexity using Dijkstra's algorithm for single source shortest path algorithm is

$$O(E + V \log V)$$

Ans. (b)

29. Consider the following two problems on undirected graphs:

α: Given $G(V,E)$, does G have an independent set of size $|V| - 4$?
β: Given $G(V,E)$, does G have an independent set of size 5?

Which one of the following is TRUE?

(a) α is in P and β is NP-complete
(b) α is NP-complete and β is in P
(c) α Both and β are NP-complete
(d) α Both and β are in P

(GATE 2005: 2 Marks)

Solution: Independent set decision problems of graph are NP-complete problems.

Ans. (c)

Linked Answer Questions 30 and 31: Consider the following C-function:

```
double foo (int n) {
int i;
double sum;
if (n==0) return 1.0;
else {
sum = 0.0;
for (i =0; i<n; i++)
sum +=foo(i);
return sum;
} }`
```

(GATE 2005: 2 Marks)

30. The space complexity of the above function is:

(a) O(1) (b) O(n)
(c) O(n!) (d) O(n^n)

Solution: The space complexity is the number of elements to be stored. This will depend on the height of tree produced. Maximum storage required is O(n).

Ans. (b)

31. Suppose we modify the above function foo() and store the values of foo(i), 0 < = I < n, as and when they are computed. With this modification, the time complexity for function foo() is significantly reduced. The space complexity of the modified function would be:

(a) O(1) (b) O(n)
(c) O(n^2) (d) O(n!)

Solution: Complexity will remain the same. With modification also, the required space will be same.

Ans. (b)

Linked Answer Questions 32 and 33: We are given 9 tasks T1, T2, ... T9. The execution of each task requires one unit of time. We can execute one task at a time. Each task T_i has a profit P_i and a deadline d_i. Profit P_i is earned if the task is completed before the end of the $(d_i)^{th}$ unit of time.

Task	T1	T2	T3	T4	T5	T6	T7	T8	T9
Profit	15	20	30	18	18	10	23	16	25
Deadline	7	2	5	3	4	5	2	7	3

32. Are all tasks completed in the schedule that gives maximum profit?

(a) All tasks are completed
(b) T1 and T6 are left out
(c) T1 and T8 are left out
(d) T4 and T6 are left out

(GATE 2005: 2 Marks)

Solution:

T2	T7	T9	T5	T3	T8	T1

T4 and T6 are left out.

Ans. (d)

33. What is the maximum profit earned?

(a) 147 (b) 165 (c) 167 (d) 175

(GATE 2005: 2 Marks)

Solution: Maximum profit = 20 +23 + 25 + 18 + 30 + 15 + 16 = 147

Ans. (a)

34. Let S be an NP-complete problem and Q and R be two other problems not known to be in NP. Q is polynomial time reducible to S, and S is polynomial time reducible to R. Which one of the following statements is true?

(a) R is NP-complete. (b) R is NP-hard.
(c) Q is NP-complete. (d) Q is NP-hard.

(GATE 2006: 1 mark)

Solution: R is NP-hard because S is polynomial time reducible to R.

Ans. (b)

35. A set X can be represented by an array $x[n]$ as follows:

$$x[i] = \begin{cases} 1 & \text{if } i \in X \\ 0 & \text{otherwise} \end{cases}$$

Consider the following algorithm in which x, y and z are Boolean arrays of size n:

```
algorithm zzz(x[] , y[], z [])
{
int i;
for (i=0; i<n; ++i)
z[i] = (x[i] ^ ~y[i]) V (~x[i] ^ y[i])
}
```

The set Z computed by the algorithm is:

(a) (X intersection Y)
(b) (X union Y)
(c) (X– Y) intersection (Y–X)
(d) (X– Y) union (Y–X)

(GATE 2006: 2 Marks)

Solution:

`x[i]^~y[i]` = only 1's in x where corresponding entry in y is 0 → $X - Y$

`~x[i]^y[i]` = only 1's in y where corresponding entry in x is 0 → $Y - X$

The operator 'V' results in a union of the above two sets.

Ans. (d)

36. Let SHAM3 be the problem of finding a Hamiltonian cycle in a graph $G = (V,E)$ with V divisible by 3 and DHAM3 be the problem of determining if a Hamiltonian cycle exists in such graphs. Which one of the following is true?

A	B	AoB
True	True	True
True	False	True
False	True	False
False	False	True

(a) Both DHAM3 and SHAM3 are NP-hard
(b) SHAM3 is NP-hard, but DHAM3 is not
(c) DHAM3 is NP-hard, but SHAM3 is not
(d) Neither DHAM3 nor SHAM3 is NP-hard

(GATE 2006: 2 Marks)

Solution: Both are NP-hard problems, and finding the Hamiltonian cycle is also an NP-hard problem.

Ans. (a)

37. Consider the following recurrence:

$T(n) = 2T\left(\left\lceil\sqrt{n}\right\rceil\right) + 1, \qquad T(1) = 1$

Which one of the following is true?

(a) $T(n) = \Theta(\log \log n)$ (b) $T(n) = \Theta(\log n)$
(c) $T(n) = \Theta(\text{sqrt}(n))$ (d) $T(n) = \Theta(n)$

(GATE 2006: 2 Marks)

Solution: Let $n = 2^m$. Then, we have

$$T(2^m) = T(2^{m/2}) + 1$$

Let $T(2^m) = S(m)$. Then, we have

$$S(m) = 2S\left(\frac{m}{2}\right) + 1$$

The above expression is a binary tree traversal recursion whose time complexity is $\theta(m)$. You can also prove this using the master theorem:

$$S(m) = \theta(m) = \theta(\log n)$$

since $n = 2^m$.
Now, let us go back to the original recursive function $T(n)$:

$$T(n) = T(2^m) = S(m) = \Theta(\log n)$$

Ans. (b)

38. Consider the polynomial $p(x) = a_0 + a_1x + a_2x^2 + a_3x^3$, where $a_i! = 0$, for all i. The minimum number of multiplications needed to evaluate p on an input x is

(a) 3 (b) 4 (c) 6 (d) 9

(GATE 2006: 2 Marks)

Solution: We have

$$p(x) = a_0 + a_1x + a_2x^2 + a_3x^2$$

$x^2 = 1$ multiplication

$a_1x + a_2x^2 + a_3x^3 = 3$ multiplications

Total four multiplications are required.

Ans. (b)

39. To implement Dijkstra's shortest path algorithm on unweighted graphs so that it runs in linear time, the data structure to be used is:

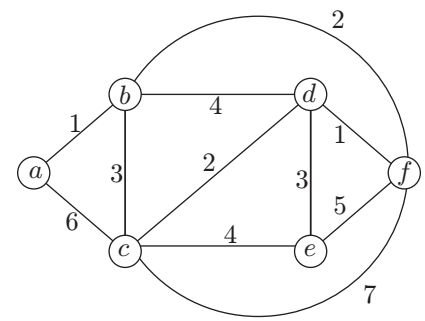
(a) Queue (b) Stack (c) Heap (d) B-tree

(GATE 2006: 2 Marks)

Solution: If we use Queue data structure to implement Dijkstra's shortest path algorithm on unweighted graphs Then shortest path will be calculated in linear time $O(|V| + |E|)$. Basically we do BFS traversal of graph to get the shortest path.

Ans. (a)

40. Consider the following graph:



Which one of the following cannot be the sequence of edges added, in that order, to a minimum spanning tree using Kruskal's algorithm?

(a) (a–b), (d–f), (b–f), (d–c), (d–e)
(b) (a–b), (d–f), (d–c), (b–f), (d–e)
(c) (d–f), (a–b), (d–c), (b–f), (d–e)
(d) (d–f), (a–b), (b–f), (d–e), (d–c)

(GATE 2006: 2 Marks)

Solution: Edges are added according to increasing number of weights. In option (d), d–e is added before d–c.

Ans. (d)

41. The median of n elements can be found in $O(n)$ time. Which one of the following is correct about the complexity of quick sort, in which median is selected as pivot?

(a) $O(n)$ (b) $O(n \log n)$
(c) $O(n^2)$ (d) $O(n^3)$

(GATE 2006: 2 Marks)

Solution: Randomized quick sort:

$T(n) = 2T(n/2) + 1 = O(n \log n)$.

Ans. (b)

42. Consider a weighted complete graph G on the vertex set $\{v_1, v_2, \ldots, v_n\}$ such that the weight of the edge (v_i, v_j) is $2|i - j|$. The weight of a minimum spanning tree of G is

(a) $n - 1$ (b) $2n - 2$ (c) ${}^nC_2$ (d) 2

(GATE 2006: 2 Marks)

Solution: Weight of the minimum spanning tree is

$$2\,|2-1| + 2\,|3-2| + 2\,|4-3| + 2\,|5-4| + \cdots + 2\,|n-(n-1)| = 2n - 2$$

Ans. (b)

43. Let w be the minimum weight among all edge weights in an undirected connected graph. Let e be a specific edge of weight w. Which of the following is *false*?

(a) There is a minimum spanning tree containing e.
(b) If e is not in a minimum spanning tree T, then in the cycle formed by adding e to T, all edges have the same weight.
(c) Every minimum spanning tree has an edge of weight w.
(d) e is present in every minimum spanning tree.

(GATE 2007: 2 Marks)

Solution: Options (a), (b) and (c) are correct. But option (d) is incorrect as there may be many edges of weight w in the graph and edge e may not be in some of the minimum spanning trees.

Ans. (d)

44. Consider the following C code segment:

```
int IsPrime(n)
{
int i,n;
for(i=2;i<=sqrt(n);i++)
if(n%i == 0)
{printf('Not Prime\n'); return 0;}
return 1;
}
```

Let $T(n)$ denote the number of times the `for` loop is executed by the program on input n. Which of the following is *true*?

(a) $T(n) = \text{O}(\text{sqrt}(n))$ and $T(n) = \Omega(\text{sqrt}(n))$
(b) $T(n) = \text{O}(\text{sqrt}(n))$ and $T(n) = \Omega(1)$
(c) $T(n) = \text{O}(n)$ and $T(n) = \Omega(\text{sqrt}(n))$
(d) None of the above

(GATE 2007: 2 Marks)

Solution: The for loop runs maximum for sqrt(n) and minimum for 1. Therefore,

$$T(n) = O(\text{sqrt}(n)) \text{ and } T(n) = \Omega(1)$$

Ans. (b)

45. In an unweighted, undirected connected graph, the shortest path from a node S to every other node is computed most efficiently in terms of time complexity by

(a) Dijkstra's algorithm starting from S
(b) Warshall's algorithm
(c) Performing a DFS starting from S
(d) Performing a BFS starting from S

(GATE 2007: 2 Marks)

Solution: Time complexity of BFS is $\text{O}(|E| + |V|)$, whereas time complexity of Dijkstra's is $\text{O}(|V|^2 + E)$ and Warshall's algorithm is $O(|V|^3)$ and DFS cannot be used for finding shortest path.

Ans. (d)

46. In the following C function, let $n \geq m$:

```
int gcd(n,m)
{
if (n%m ==0) return m;
n = n%m;
return gcd(m,n);
}
```

How many recursive calls are made by this function?

(a) $\Theta(\log_2 n)$ (b) $\Omega(n)$
(c) $\Theta(\log_2 \log_2 n)$ (d) $\Theta(\text{sqrt}(n))$

(GATE 2007: 2 Marks)

Solution:
Here, $T_n = 1 + \frac{1}{n}(T_0 T_1 + \cdots + T_{n-1}) = \Theta(\log_2 n)$

Ans. (a)

47. What is the time complexity of the following recursive function?

```
int DoSomething (int n)
{
if (n <= 2)
return 1;
else
return (DoSomething (floor(sqrt(n)))+n);
}
```

(a) $\Theta(n)$ (b) $\Theta(n \log_2 n)$
(c) $\Theta(\log n)$ (d) $\Theta(\log_2 \log_2 n)$

(GATE 2007: 2 Marks)

Solution: Recursive relation for the DoSomething() is

$(T(n) = T(\sqrt{n}) + C_1)$ if $n > 2$
Let $n = 2^m$, so $T(n) = T(2^m)$
Let $T(2^m) = S(m)$

From the above two expressions, we have

$T(n) = S(m)$
$S(m) = \text{S}(m/2) + C_1$
$S(m) = \text{O}(\log_2 m)$ (by Master Theorem)
$S(m) = \text{O}(\log_2 \log_2 n)$ (because $n = 2^m$)

From the above, we have $S(m) = T(n) = \text{O}(\log_2 \log_2 n)$

Ans. (d)

Linked Answer Questions 48 and 49: Suppose the letters a, b, c, d, e, f have probabilities $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{32}$, respectively.

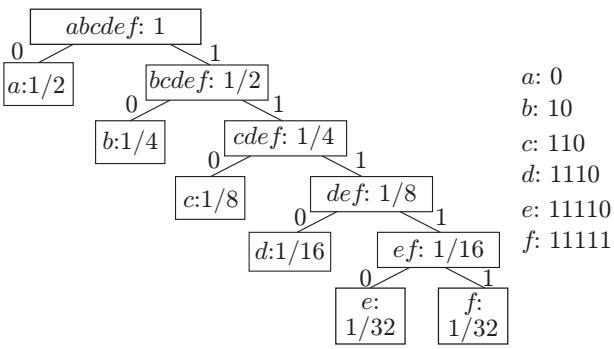
48. Which of the following is the Huffman code for the letter a, b, c, d, e, f?

(a) 0, 10, 110, 1110, 11110, 11111
(b) 11, 10, 011, 010, 001, 000
(c) 11, 10, 01, 001, 0001, 0000
(d) 110, 100, 010, 000, 001, 111

(GATE 2007: 2 Marks)

Solution:

a	b	c	d	e	f
1/2	1/4	1/8	1/16	1/32	1/32



Ans. (a)

49. What is the average length of the correct answer to the above question?

(a) 3 (b) 2.1875 (c) 2.25 (d) 1.9375

(GATE 2007: 2 Marks)

Solution: The average length =

$$1\times\frac{1}{2}+2\times\frac{1}{4}+3\times\frac{1}{8}+4\times\frac{1}{16}+5\times\frac{1}{32}+5\times\frac{1}{32}=1.9375$$

Ans. (d)

50. The most efficient algorithm for finding the number of connected components in an undirected graph on n vertices and m edges has time complexity

(a) $\Theta(n)$ (b) $\Theta(m)$
(c) $\Theta(m + n)$ (d) $\Theta(mn)$

(GATE 2008: 1 Mark)

Solution: DFS can be used to find out the number of connected components in an undirected graph. The time complexity of DFS is $O(m + n)$.

Ans. (c)

51. The minimum number of comparisons required to determine if an integer appears more than $n/2$ times in a sorted array of n integers is

(a) $\Theta(n)$ (b) $\Theta(\log n)$
(c) $\Theta(\log^* n)$ (d) $\Theta(1)$

(GATE 2008: 1 Mark)

Solution: The minimum number of comparison would be 1 since if an element exists more than $n/2$ times in a sorted list than it has to be in middle. The middle index of an array can be reached in a constant time.

Ans. (a)

52. G is a graph on n vertices and $2n - 2$ edges. The edges of G can be partitioned into two edge-disjoint spanning trees. Which of the following is NOT true for G?

(a) For every subset of k vertices, the induced sub-graph has at most $2k - 2$ edges.
(b) The minimum cut in G has at least two edges.
(c) There are two edge-disjoint paths between every pair of vertices.
(d) There are two vertex-disjoint paths between every pair of vertices.

(GATE 2008: 2 Marks)

Solution: Option (a) is true since the induced sub-graph on k vertices can be constructed as the union of the induced sub-graphs on k vertices of two edge disjoint spanning tree and each of the induced spanning trees has atmost $k - 1$ edges.
Option (b) is true otherwise G cannot be partitioned into two edge disjoint spanning trees.
Option (c) is true otherwise G has minimum cut of one edge.
Option (d) is false.

Ans. (d)

53. Consider the Quicksort algorithm. Suppose there is a procedure for finding a pivot element which splits the list into two sub-lists each of which contains at least one-fifth of the elements. Let $T(n)$ be the number of comparisons required to sort n elements. Then

(a) $T(n) \le 2T\left(\frac{n}{5}\right) + n$

(b) $T(n) \le T\left(\frac{n}{5}\right) + T\left(\frac{4n}{5}\right) + n$

(c) $T(n) \le 2T\left(\frac{4n}{5}\right) + n$

(d) $T(n) \le 2T\left(\frac{n}{2}\right) + n$

(GATE 2008: 2 Marks)

Solution: The recurrence relation is $T(n) = T(a) + T(b) + n, a \ge n/5$, $b \ge n/5$ and $a + b = n$

Thus, $a \le 4n/5$ and $b \le 4n/5$, so $T(n) \le 2T\left(\frac{4n}{5}\right) + n$

Ans. (c)

54. The subset-sum problem is defined as follows: Given a set S of n positive integers and a positive integer W, determine whether there is a subset of S whose elements sum to W. An algorithm Q solves this problem in $O(nW)$ time. Which of the following statements is false?

(A) Q solves the subset-sum problem in polynomial time when the input is encoded in unary
(B) Q solves the subset-sum problem in polynomial time when the input is encoded in binary
(C) The subset sum problem belongs to the class NP
(D) The subset sum problem is NP-hard

(GATE 2008: 2 Marks)

Solution: Since we know that subset sum is a NP-hard problem, thus problem may take linear time when the input encoded is unary.

Ans. (b)

Linked Answer Questions 55 and 56: The subsetsum problem is defined as follows. Given a set of n positive integers, $S = \{a_1, a_2, a_3, \ldots, a_n\}$ and positive integer W, is there a subset of S whose elements sum to W? A dynamic program for solving this problem uses a two-dimensional Boolean array X, with n rows and $W + 1$ columns. $X[i, j]$, $1 \le i \le n$, $0 \le j \le W$, is *true* if and only if there is a subset of $\{a_1, a_2, \ldots, a_i\}$ whose elements sum to j.

55. Which of the following is valid for $2 \le i \le n$ and $ai \le j \le W$?

(a) $X[i, j] = X[i-1, j] \vee X[i, j-a_i]$
(b) $X[i, j] = X[i-1, j] \vee X[i-1, j-a_i]$
(c) $X[i, j] = X[i-1, j] \vee X[i, j-a_i]$
(d) $X[i, j] = X[i-1, j] \vee X[i-1, j-a_i]$

(GATE 2008: 2 Marks)

Solution: $X[i, j]$ ($2 \le i \le n$ and $a_i \le j \le W$) is true if any of the following is true:
(1) Sum of weights excluding a_i is equal to j, that is, if $X[i-1, j]$ is true.
(2) Sum of weights including a_i is equal to j, that is, if $X[i-1, j-a_i]$ is true so that we get $(j-a_i) + a_i$ as j.

Ans. (b)

56. Which entry of the array X, if *true*, implies that there is a subset whose elements sum to W?

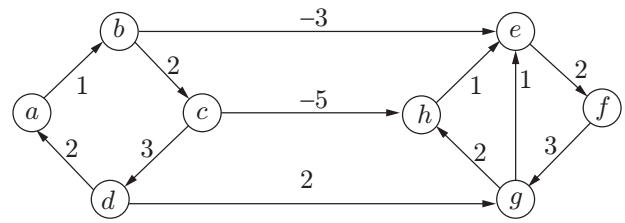
(a) $X[1, W]$ (b) $X[n, 0]$
(c) $X[n, W]$ (d) $X[n-1, n]$

(GATE 2008: 2 Marks)

Solution:
If we get the entry $X[n, W]$ as true, then there is a subset of $\{a_1, a_2, \ldots, a_n\}$ that has sum as W.

Ans. (c)

57. Dijkstra's single-source shortest path algorithm when run from vertex a in the following graph computes the correct shortest path distance to



(a) Only vertex a
(b) Only vertices a, e, f, g and h
(c) Only vertices a, b, c and d
(d) All of the vertices

(GATE 2008: 2 Marks)

Solution: Following the Dijkstra's algorithm's single-source shortest paths, we can reach all the vertices here.

Taking 'a' as the source: minimum distance is computed by checking all the adjacent nodes, as follows:

$$D(\text{a}) = 0$$
$$D(\text{b}) = \text{cost}(\text{a} \to \text{b}) = 1$$

Here, the minimum distance from source is computed, but since there is only one adjacent node, there is no need to check any other node for minimum.
Now, here b node has two adjacent nodes e and c. But since cost(b → e) has a negative weight, we ignore that path as Dijkstra does not entertain negative weights. Thus, we will proceed with node c:

$$D(\text{c}) = D(\text{b}) + \text{cost}(\text{b} \to \text{c}) = 2 + 1 = 3$$

Also c node could be reached only through b, so that is the only possible minimum weight of c.
As similar to node b, node c is also left with one child d due to negative edge removal c → h:

$$D(\text{d}) = D(\text{c}) + \text{cost}(\text{c} \to \text{d}) = 3 + 3 = 6$$

Similarly, proceeding further the same way, we can reach all nodes and compute the shortest path.

Ans. (d)

58. Consider the following functions:

$$f(n) = 2^n \qquad g(n) = n! \qquad h(n) = n^{\log n}$$

Which of the following statements about the asymptotic behaviour of $f(n)$, $g(n)$ and $h(n)$ is true?

(a) $f(n) = O(g(n))$; $g(n) = O(h(n))$
(b) $f(n) = \Omega(g(n))$; $g(n) = O(h(n))$
(c) $g(n) = O(f(n))$; $h(n) = O(f(n))$
(d) $h(n) = O(f(n))$; $g(n) = \Omega(f(n))$

(GATE 2008: 2 Marks)

Solution: The functions can be arranged in an increasing order as $h(n) < f(n) < g(n)$. Hence, option (d) is the correct one.

Ans. (d)

59. Let X be a problem that belongs to the class NP. Then which one of the following is *true*?

(a) There is no polynomial time algorithm for X.
(b) If X can be solved deterministically in polynomial time, then $P = \text{NP}$.
(c) If X is NP-hard, then it is NP-complete.
(d) X may be undecidable.

(GATE 2009: 1 mark)

Solution: Option (a) is incorrect because NP includes both P and NPC.
Option (b) is also incorrect because X may belong to P but it does not mean that P = NP.
Option (c) is correct because NP-complete set is intersection of NP and NP-hard sets.
Option (d) is incorrect because all NP problems are decidable in finite set of operations.

Ans. (c)

60. What is the minimum number of swaps required to sort n elements using selection sort, in the worst case?

(a) $\Theta(n)$ (b) $\Theta(n \log n)$
(c) $\Theta(n^2)$ (b) $\Theta(n^2 \log n)$

(GATE 2009: 1 mark)

Solution: Selection sort select the minimum element and the swap with first position. In the worst case, all n elements will be swapped.

Ans. (a)

61. The keys 12, 18, 13, 2, 3, 23, 5 and 15 are inserted into an initially empty hash table of length 10 using open addressing with hash function $h(k) = k$ and linear probing. What is the resultant hash table?

(A)

0	
1	
2	2
3	23
4	
5	15
6	
7	
8	18
9	

(B)

0	
1	
2	12
3	13
4	
5	5
6	
7	
8	18
9	

(C)

0	
1	
2	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

(D)

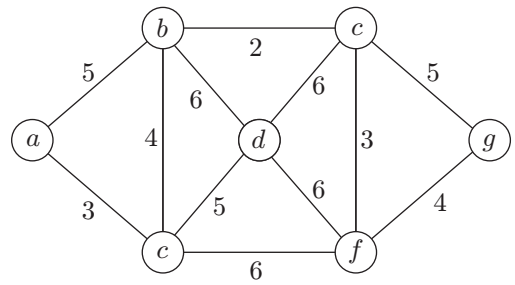
0	
1	
2	12,2
3	13, 3, 23
4	
5	5, 15
6	
7	
8	18
9	

(GATE 2009: 2 Marks)

Solution:
12 mod 10 → 2
18 mod 10 → 8
13 mod 10 → 3
2 mod 10 → 2 [2, 3 are occupied] so will go to 4
3 mod 10 → 3 [3, 4 are occupied] so will go to 5
23 mod 10 → 3 [3, 4, 5 are occupied] so will go to 6
5 mod 10 → 5 [5, 6 occupied] so will go to 7
15 mod 10 → 5 [5, 6, 7, 8 are occupied] will go to 9

Ans. (c)

62. Consider the following graph:



(GATE 2009: 2 Marks)

Which one of the following is NOT the sequence of edges added to the minimum spanning tree using Kruskal's algorithm?

(A) (b, e) (e, f) (a, c) (b, c) (f, g) (c, d)
(C) (b, e) (a, c) (e, f) (b, c) (f, g) (c, d)
(B) (b, e) (e, f) (a, c) (f, g) (b, c) (c, d)
(D) (b, e) (e, f) (b, c) (a, c) (f, g) (c, d)

(GATE 2009: 2 Marks)

Solution: In Kruskal's algorithm edges are chosen in increasing order of weights. So, (b, e) = 2, (e, f) = 3, (b, c) = 4, (a, c) = 3, (f, g) = 4 and (c, d) = 5.

Ans. (d)

63. The running time of an algorithm is represented by the following recurrence relation:

```
if n <= 3 then T(n) = n
else T(n) = T(n/3) + cn
```

Which one of the following represents the time complexity of the algorithm?

(a) $\Theta(n)$ (b) $\Theta(n \log n)$
(c) $\Theta(n^2)$ (d) $\Theta(n^2 \log n)$

(GATE 2009: 2 Marks)

Solution: As per the master theorem, case 3, we have $a = 1$, $b = 3$ and $f(n) = cn$. Thus

$$\log_b a = \log_3 1 = 0$$

$$n^0 < f(n)$$

So, the answer is $\Theta(n)$.

Ans. (a)

64. In quick sort, for sorting n elements, the $(n/4)$th smallest element is selected as pivot using an $O(n)$ time algorithm. What is the worst-case time complexity of the quick sort?

(a) $\Theta(n)$ (b) $\Theta(n \log n)$
(c) $\Theta(n^2)$ (d) $\Theta(n^2 \log n)$

(GATE 2009: 2 Marks)

Solution: The recursive expression is

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + cn$$

$$T(n) = \Theta(n \log n)$$

This expression is of randomized quick sort.

Ans. (b)

Common Data Questions 65 and 66: A subsequence of a given sequence is just the given sequence with some elements (possibly none or all) left out. We are given two sequences $X[m]$ and $Y[n]$ of lengths m and n, respectively, with indexes of X and Y starting from 0.

65. We wish to find the length of the longest common subsequence (LCS) of $X[m]$ and $Y[n]$ as $l(m, n)$, where an incomplete recursive definition for the function $l(i, j)$ to compute the length of the LCS of $X[m]$ and $Y[n]$ is given as follows:

```
l (i, j) 0, if either i=0 or j=0
= expr1, if i,j>0 and X[i -1] = Y[j - 1]
= expr2, if i,j>0 and X[i - 1] = Y[j - 1]
```

Which one of the following options is correct?

(a) expr1 $\equiv (i-1, j) + 1$
(b) expr1 $\equiv l(i, j-1)$
(c) expr2 $\equiv \max(l(i-1, j), l(i, j-1))$
(d) expr2 $\equiv \max(l(i-1, j-1), l(i, j))$

(GATE 2009: 2 Marks)

Solution: LCS problem is solved using dynamic programing in which every row is dependent upon previous row and column. So, if two characters at i^{th} row and j^{th}column do not match then the length is calculated as:

$$\text{expr2} = \max(l(i-1, j), l(i, j-1))$$

The values of $l(i, j)$ could be obtained by dynamic programming based on the correct recursive definition of $l(i, j)$ of the form given above, using an array $L[M, N]$, where $M = m + 1$ and $N = n + 1$, such that $L[i, j] = l(i, j)$.

Ans. (c)

66. Which one of the following statements would be *true* regarding the dynamic programming solution for the recursive definition of $l(i, j)$?

(a) All elements L should be initialized to 0 for the values of $l(i, j)$ to be properly computed.
(b) The values of $l(i, j)$ may be computed in a row-major order or column-major order of $L(M, N)$.
(c) The values of $l(i, j)$ cannot be computed in either row-major order or column-major order of $L(M, N)$.
(d) $L[p, q]$ needs to be computed before $L[r, s]$ if either $p < r$ or $q < s$.

(GATE 2009: 2 Marks)

Solution: The value can be computed in either row-major or column-major order.

Ans. (b)

67. Two alternative packages A and B are available for processing a database having 10^k records. Package A requires $0.0001\ n^2$ time units and package B requires $10n\log_{10}n$ time units to process n records. What is the smallest value of k for which the package B will be preferred over A?

(a) 12 (b) 10
(c) 6 (d) 5

(GATE 2010: 1 mark)

Solution:
$0.0001\ n^2 < 10\ n \log n \Rightarrow$

$10^{-5} n < \log n \Rightarrow 10^{-5} \times 10^k < k$ (as $\log_{10} n = k$)

Hence, $k - 5 > 0$ or $k > 6$. Thus, for $k = 6$, B will be preferred.

Ans. (c)

68. The weight of a sequence $a_0, a_1, \ldots a_{n-1}$ of real numbers is defined as

$$a_0 + a_1/2 + \cdots + a_{n-1} / 2^{n-1}$$

A subsequence of a sequence is obtained by deleting some elements from the sequence, keeping the order of the remaining elements the same. Let X denote the maximum possible weight of a subsequence of $a_0, a_1, \ldots, a_{n-1}$ and Y the maximum possible weight of a subsequence of $a_1, a_2, \ldots, a_{n-1}$. Then X is equal to

(a) $\max(Y, a_0 + Y)$ (b) $\max(Y, a_0 + Y)$
(c) $\max(Y, a_0 + 2Y)$ (d) $a_0 + Y/2$

Solution: This involves dynamic programming in algorithms. Given that
X= Maximum weight from the sequence $(a_0, a_1, a_2, \ldots a_{n-1}) = a_0 + \frac{a_1}{2} + \frac{a_2}{4} + \cdots + \frac{a_{n-1}}{2^{n-1}}$

Y = Maximum weight from the sequence $(a_1, a_2, \ldots a_{n-1}) = a_1 + \frac{a_2}{2} + \frac{a_3}{4} + \cdots + \frac{a_{n-1}}{2^{n-2}}$

Here, we have to find X in terms of Y.

So, $X = a_1 + \frac{a_2}{2} + \frac{a_3}{4} + \cdots + \frac{a_{n-1}}{2^{n-2}} = Y$

if $$a_0 + \frac{a_1}{2} + \frac{a_2}{4} + \cdots + \frac{a_{n-1}}{2^{n-1}} < a_1 + \frac{a_2}{2} + \frac{a_3}{4} + \cdots + \frac{a_{n-1}}{2^{n-2}}$$

OR

$$X = a_0 + \frac{a_1}{2} + \frac{a_2}{4} + \cdots + \frac{a_{n-1}}{2^{n-1}} = a_0 + Y/2 \text{ if}$$

$$a_0 + \frac{a_1}{2} + \frac{a_2}{4} + \cdots + \frac{a_{n-1}}{2^{n-1}} > a_1 + \frac{a_2}{2} + \frac{a_3}{4} + \cdots + \frac{a_{n-1}}{2^{n-2}}$$

Hence, we sum up as: $X = \max(Y, a_0 + Y/2)$

Ans. (b)

Common Data Questions 69 and 70: Consider a complete undirected graph with vertex set {0, 1, 2, 3, 4}. Entry W_{ij} in the following matrix W is the weight of the edge $\{i, j\}$:

$$W = \begin{pmatrix} 0 & 1 & 8 & 1 & 4 \\ 1 & 0 & 12 & 4 & 9 \\ 8 & 12 & 0 & 7 & 3 \\ 1 & 4 & 7 & 0 & 2 \\ 4 & 9 & 3 & 2 & 0 \end{pmatrix}$$

69. What is the minimum possible weight of a spanning tree T in this graph such that vertex 0 is a leaf node in the tree T?

(a) 7 (b) 8
(c) 9 (d) 1

(GATE 2010: 2 Marks)

Solution: To get the minimum spanning tree with vertex 0 as leaf, first remove 0th row and 0th column and then get the minimum spanning tree (MST) of the remaining graph. Once we have MST of the remaining graph, connect the MST to vertex 0 with the edge with minimum weight (we have two options as there are two 1's in the 0th row).

Ans. (d)

70. What is the minimum possible weight of a path P from vertex 1 to vertex 2 in this graph such that P contains at most three edges?

(a) 7 (b) 8
(c) 9 (d) 10

(GATE 2010: 2 Marks)

Solution: Path: $1 \to 0 \to 4 \to 2$
Weight: $1 + 4 + 3 = 8$

Ans. (b)

Linked Answer Questions 71 and 72: A hash table of length 10 uses open addressing with hash function $h(k) = k \bmod 10$, and linear probing. After inserting 6 values into an empty hash table, the v is as shown below:

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

71. Which one of the following choices gives a possible order in which the key values could have been inserted in the table?

(a) 46, 42, 34, 52, 23, 33
(b) 34, 42, 23, 52, 33, 46
(c) 46, 34, 42, 23, 52, 33
(d) 42, 46, 33, 23, 34, 52

(GATE 2010: 2 Marks)

Solution: Check the options according to linear probing.
The correct sequence is 46, 34, 42, 23, 52, 33.

Ans. (c)

72. How many different insertion sequences of the key values using the same hash function and linear probing will result in the hash table shown above?

(a) 10 (b) 20
(c) 30 (d) 40

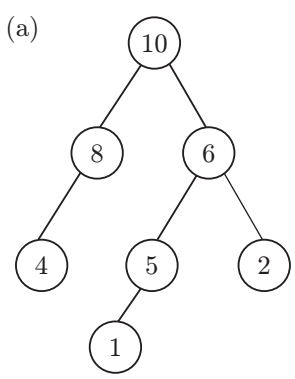
(GATE 2010: 2 Marks)

Solution: We have $n(n-1) = 6 \times 5 = 30$

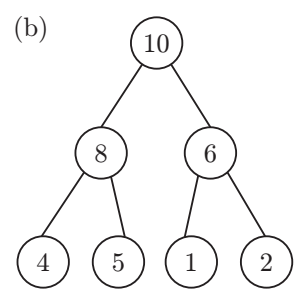
Ans. (c)

73. A max-heap is a heap where the value of each parent is greater than or equal to the value of its children. Which of the following is a max-heap?

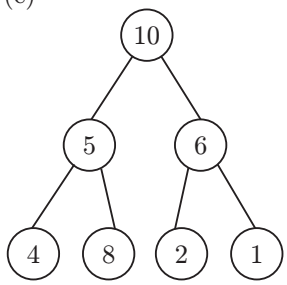
(a)



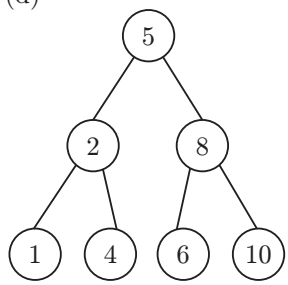
(b)



(c)



(d)



(GATE 2011: 1 mark)

Solution: Heap is also a complete binary tree.

Ans. (b)

74. An algorithm to find the length of the longest monotonically increasing sequence of numbers in an array $A[O: n-1]$ is given below.

Let L_i denote the length of the longest monotonically increasing sequence starting at index i in the array.

```
Initialize L_{n-L} = 1.
For all i such that 0 ≤ i ≤ n - 2
```

$$L_i = \begin{cases} 1 + L_{i+1} & \text{if } A[i] < A[i+1] \\ 1 & \text{otherwise} \end{cases}$$

Finally, the length of the longest monotonically increasing sequence is $\text{Max}(L_0, L_1, \ldots, L_{n-1})$.
Which of the following statements is *true*?

(a) The algorithm uses dynamic programming paradigm.
(b) The algorithm has a linear complexity and uses branch and bound paradigm.
(c) The algorithm has a non-linear polynomial complexity and uses branch and bound paradigm.
(d) The algorithm uses divide and conquer paradigm.

(GATE 2011: 1 Mark)

Solution: In the given algorithm, length of the longest monotonically increasing sequence is calculated as :L_i in terms of L_{i+1}. So, it uses dynamic programing technique.

Ans. (a)

75. We are given a set of n distinct elements and an unlabelled binary tree with n nodes. In how many ways can we populate the tree with the given set so that it becomes a binary search tree?

(a) 0 (b) 1 (c) $n!$ (d) $\frac{1}{n+1}\,{}^{2n}C_n$

(GATE 2011: 2 Marks)

Solution: We can populate only in one way because minimum value will be in the leftmost node, and the maximum value will be in the rightmost node. Recursively, we can define for other nodes.

Ans. (b)

76. Which of the given options provides the increasing order of asymptotic complexity of functions f_1, f_2, f_3 and f_4?

$f_1(n) = 2^n$; $f_2(n) = n^{3/2}$; $f_3(n) = n \log_2 n$;
$f_4(n) = n^{\log_2 n}$

(a) f_3, f_2, f_4, f_1 (b) f_3, f_2, f_1, f_4
(c) f_2, f_3, f_1, f_4 (d) f_2, f_3, f_4, f_1

(GATE 2011: 2 Marks)

Solution: Take n as a large value and evaluate functions. Let $n = 32$, then

f_1=4294967296 f_2 =181.019 f_3 =160
f_4 =33554432

$$f_3 < f_2 < f_4 < f_1$$

Ans. (a)

77. Consider a relational table r with sufficient number of records, having attributes $A_1, A_2, \ldots, A_n$ and let $1 \le p \le n$. Two queries $Q1$ and $Q2$ are given as follows:

Q1: $\Pi_{A_1,\ldots,A_p}(\sigma_{A_p=c}(r))$, where c is a constant

Q2: $\Pi_{A_1,\ldots,A_p}(\sigma_{c_1 \le A_p \le c_2}(r))$, where c_1 and c_2 are constants

The database can be configured to do ordered indexing on A_p or hashing on A_p. Which of the following statements is *true*?

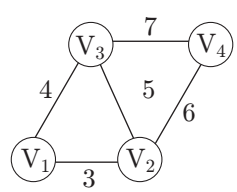
(a) Ordered indexing will always outperform hashing for both queries.
(b) Hashing will always outperform ordered indexing for both queries.
(c) Hashing will outperform ordered indexing on Q1, but not on Q2.
(d) Hashing will outperform ordered indexing on Q2, but not on Q1.

(GATE 2011: 2 Marks)

Solution:
Hashing will outperform ordered indexing on Q1, but not Q2.

Ans. (c)

Linked Answer Questions 78 and 79: An undirected graph $G(V, E)$ contains n ($n > 2$) nodes named v_1, v_2, …, v_n. Two nodes v_i and v_j are connected if and only if $0 <- i - j - \le 2$. Each edge (v_i, v_j) is assigned a weight $i + j$. A sample graph with $n = 4$ is shown as follows:



78. What will be the cost of the minimum spanning tree (MST) of such a graph with n nodes?

(a) $(1/12)\ (11n^2 - 5n)$ (b) $n^2 - n + 1$
(c) $6n - 11$ (d) $2n + 1$

(GATE 2011: 2 Marks)

Solution:
Let us have 2 nodes in the tree, then minimum spanning tree is

$$v_1 - v_2$$

Total weight = 3
Minimum spanning tree for 3 nodes would be

$$\begin{array}{l} v_1 - v_2 \\ | \\ v_3 \end{array}$$

Total weight= 3 + 4 = 7
Minimum spanning tree for 4 nodes would be

$$\begin{array}{l} v_1 - v_2 - v_4 \\ | \\ v_3 \end{array}$$

Total weight= 3 + 4 + 6 = 13
From the above examples, we observe that when we add i^{th} node, the weight of spanning tree increases by $2i - 2$. Let $T(n)$ be the weight of minimum spanning tree.

$$T(n) = \begin{cases} T(n-1) + (2n-2), & n > 2 \\ 3, & n = 1 \\ 0, & n = 2 \end{cases}$$

The recurrence can be written as sum of series $(2n-2) + (2n-4) + (2n-6) + (2n-8) + \ldots 3$, and solution of this recurrence is $n^2 - n + 1$.

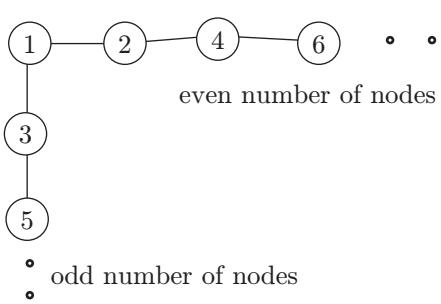
Ans. (b)

79. The length of the path from v_5 to v_6 in the MST of previous question with $n = 10$ is

(a) 11 (b) 25 (c) 31 (d) 41

(GATE 2011: 2 Marks)

Solution: Basic structure of all MSTs with more than 5 nodes(for the above given graph specification) would be following:



MST with 6 nodes have weight = 3 + 4 + 6 + 8 + 10 = 31

Ans. (c)

80. Assuming $P! = \text{NP}$, which of the following is true?

(a) NP-complete = NP
(b) NP-complete $\cap P = \Phi$
(c) NP-hard = NP
(d) P = NP-complete

(GATE 2012: 1 mark)

Solution:

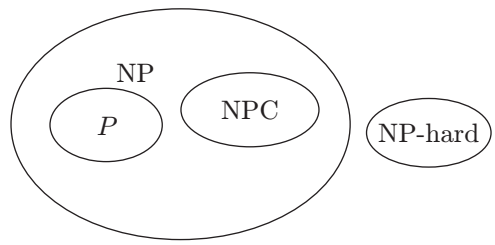


Diagram clearly shows that NP-complete $\cap P = \varepsilon$.

Ans. (b)

81. The worst-case running time to search for an element in a balanced binary search tree with $n2^n$ elements is

(a) $O(n \log n)$ (b) $O(n2^n)$
(c) $O(n)$ (d) $O(\log n)$

(GATE 2012: 1 mark)

Solution: The worst-case complexity of a balanced binary search tree is $\Theta(\log n)$, where n is the number of elements; here, n is $n2^n$, so

$$\log(n2^n) = \log n + n \log 2 = \log n + n$$

So,the complexity is $\Theta(n)$.

Ans. (c)

82. The recurrence relation capturing the optimal execution time of the Tower of Hanoi problem with n discs is

(a) $T(n) = 2T(n-2) + 2$
(b) $T(n) = 2T(n-1) + n$
(c) $T(n) = 2T(n/2) + 1$
(d) $T(n) = 2T(n-1) + 1$

(GATE 2012: 1 mark)

Solution: Recurrence relation for Tower of Hanoi is

$$T(n) = 2T(n-1) + 1$$

Ans. (d)

83. Let G be a simple undirected planar graph on 10 vertices with 15 edges. If G is a connected graph, then the number of bounded faces in any embedding of G on the plane is equal to

(a) 3 (b) 4
(c) 5 (d) 6

(GATE 2012: 1 mark)

Solution: We have

Given that vertices $(V) = 10$ and edges $(E) = 15$
Number of bounded and unbounded faces $(F) = ?$
We know that $V - E + F = 2$

$$10 - 15 + F = 2 \Rightarrow F = 7$$

So, the bounded faces $= F - 1 = 7 - 1 = 6$

Ans. (d)

84. Let $w(n)$ and $A(n)$ denote, respectively, the worst-case and average-case running time of an algorithm executed on an input of size n. Which of the following is *always true*?

(a) $A(n) = \Omega(W(n))$ (b) $A(n) = \Theta(W(n))$
(c) $A(n) = O(W(n))$ (d) $A(n) = o(W(n))$

(GATE 2012: 1 mark)

Solution:
$A(n) = O(W(n))$

Ans. (c)

85. Let G be a complete undirected graph on 6 vertices. If vertices of G are labelled, then the number of distinct cycles of length 4 in G is equal to

(a) 15 (b) 30
(c) 90 (d) 360

(GATE 2012: 2 Marks)

Solution: The number of distinct cycles of length 4 is

$$^6C_4 \times (4-1)! = {}^6C_4 \times 3! = 90$$

Ans. (c)

86. A list of n strings, each of length n, is sorted into lexicographic order using the mergesort algorithm. The worst-case running time of this computation is

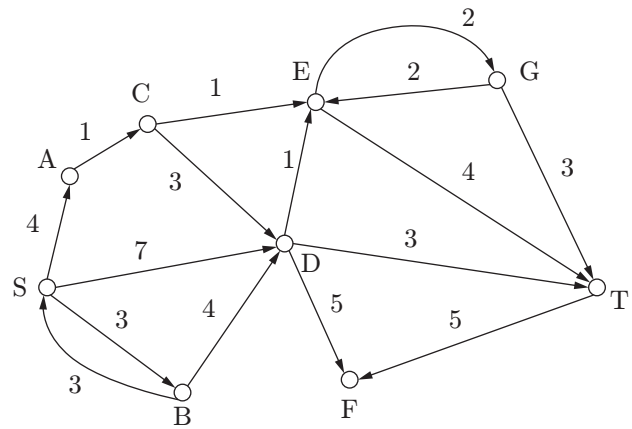
(a) $O(n \log n)$ (b) $O(n^2 \log n)$
(c) $O(n^2 + \log n)$ (d) $O(n^2)$

(GATE 2012: 2 Marks)

Solution: The worst-case runtime: $n \log n$.
For n strings: $n(n \log n)$, which gives $n^2 \log n$.

Ans. (b)

87. Consider the directed graph shown in the figure below. There are multiple shortest paths between vertices S and T. Which one will be reported by Dijkstra's shortest path algorithm? Assume that, in any iteration, the shortest path to a vertex v is updated only when a strictly shorter path to v is discovered.



(a) SDT (b) SBDT
(c) SACDT (d) SACET

(GATE 2012: 2 Marks)

Solution: Apply Dijkstra's shortest path algorithm.

S	A	B	C	D	E	G	T
0	–	–	–	–	–	–	–
0	4	3	–	7	–	–	–
0	4	3	5(A)	7	–	–	–
0	4	3	5(A)	7	6(C)	–	–
0	4	3	5(A)	7	6(C)	8(E)	10(E)

So, the minimum path will cover S → A → C → E → T, with total weight 10.

Ans. (d)

88. Which one of the following is the tightest upper bound that represents the number of swaps required to sort n numbers using selection sort?

(a) $O(\log n)$ (b) $O(n)$
(c) $O(n \log n)$ (d) $O(n^2)$

(GATE 2013: 1 mark)

Solution: The maximum number of swaps required for n elements is n.

Ans. (b)

89. Which one of the following is the tightest upper bound that represents the time complexity of inserting an object into a binary search tree of n nodes?

(a) $O(1)$ (b) $O(\log n)$
(c) $O(n)$ (d) $O(n \log n)$

(GATE 2013: 1 mark)

Solution: In case of skewed binary tree, the tightest upper bound of the insert node is $O(n)$.

Ans. (c)

90. What is the time complexity of Bellman-Ford single-source shortest path algorithm on a complete graph of n vertices?

(A) $\Theta(n^2)$ (B) $\Theta(n^2 \log n)$
(C) $\Theta(n^3)$ (D) $\Theta(n^3 \log n)$

(a)A (b)B (c)C (d)D

(GATE 2013: 1 mark)

Solution: The complexity for Bellman-Ford = $O(E \times V)$. Thus, we have

$$|E| = \frac{n(n-1)}{2} = n^2$$
$$|V| = n$$
$$|E| \times |V| = n^3$$

Ans. (c)

91. Which of the following statements are **TRUE**?

(1) The problem of determining whether there exists a cycle in an undirected graph is in P.
(2) The problem of determining whether there exists a cycle in an undirected graph is in NP.
(3) If a problem A is NP-Complete, there exists a non-deterministic polynomial time algorithm to solve A.

(a) 1, 2 and 3 (b) 1 and 2 only
(c) 2 and 3 only (d) 1 and 3 only

(GATE 2013: 1 mark)

Solution:
(1) True. Complexity for detecting cycle using DFS is $O(E+V) = O(V^2)$, which is solvable in polynomial time.
(2) True. Every problem in P is also NP.
(3) True. NP-Complete problems are solvable in non-deterministic time.

Ans. (a)

92. The line graph $L(G)$ of a simple graph G is defined as follows.

There is exactly one vertex $v(e)$ in $L(G)$ for each edge e in G.
For any two edges e and e' in G, $L(G)$ has an edge between $v(e)$ and $v(e')$, if and only if e and e' are incident with the same vertex in G.
Which of the following statements is/are *true*?

(P) The line graph of a cycle is a cycle.
(Q) The line graph of a clique is a clique.
(R) The line graph of a planar graph is planar.
(S) The line graph of a tree is a tree.

(a) P only (b) P and R only
(c) R only (d) P, Q and S only

(GATE 2013: 2 Marks)

Solution:
It can be observed by taking different examples for graph with cycle, clique and planner graph and with tree. Then we find that only option (a) is correct.

Ans. (a)

93. The number of elements that can be sorted in $\Theta(\log n)$ time using heap sort is

(A) $\Theta(1)$ (B) $\Theta(\sqrt{\log n})$
(C) $\Theta\left(\frac{\log n}{\log \log n}\right)$ (D) $\Theta(\log n)$

(a) A (b) B (c) C (d) D

(GATE 2013: 2 Marks)

Solution:
Heap sort sorts k elements in time $\Theta(k \log k)$

$$K = \Theta\left(\frac{\log n}{\log \log n}\right)$$
$$\Theta(k \log k) = \Theta\left(\frac{\log n}{\log \log n} \log \frac{\log n}{\log \log n}\right)$$
$$\Rightarrow \Theta\left(\frac{\log n}{\log \log n}(\log \log n - \log \log \log n)\right)$$
$$\Rightarrow \Theta\left(\frac{\log n}{\log \log n}(\log \log n)\right) \Rightarrow \Theta(\log n)$$

Ans. (d)

94. Consider the following function:

```
int unknown(int n) {
    int i, j, k = 0;
    for (i = n/2; i <= n; i++)
        for (j = 2; j <= n; j = j * 2)
            k = k + n/2;
    return k;
 }
```

(a) $\Theta(n^2)$ (b) $\Theta(n^{2\log n})$
(c) $\Theta(n^3)$ (d) $\Theta(n^{3\log n})$

(GATE 2013: 2 Marks)

Solution:

```
for(i=n/2; i<=n; i++) ---- > n/2,n/2+1...
n => n/2 times
{
for(j=2; j<=n; j=j*2) --------→ 2,4,8...
n => log n times
{
k = k + n/2;
}
return k;
}
```

Ans. (b)

95. Consider the following operation along with enqueue and dequeue operations on queues, where k is a global parameter:

```
MultiDequeue(Q){
   m = k
   while (Q is not empty and m > 0) {
      Dequeue(Q)
      m = m - 1
   }}
```

What is the worst-case time complexity of a sequence of n MultiDequeue() operations on an initially empty queue?

(a) $\Theta(n)$ (b) $\Theta(n + k)$
(c) $\Theta(nk)$ (d) $\Theta(n^2)$

(GATE 2013: 2 Marks)

Solution: Any combination of n enqueue and dequeue operations will take $\Theta(n)$.

Ans. (a)

PRACTICE EXERCISES

Set 1

1. What will be the height of a d-ary heap having n elements?

(a) $O(\log n \times d)$ (b) $O(d \log n)$
(c) $O(\log n/\log d)$ (d) $O(\log d/\log n)$

2. Quick sort gives $O(n \log n)$ worst-case performance if the pivot is selected as:

(a) First element of the array
(b) Median of first, last and middle elements
(c) Arithmetic mean of the first and last elements
(d) None of the above

3. A complete full binary tree with 10 leaves

(a) cannot have more than 4 nodes
(b) has exactly 19 nodes
(c) has exactly 17 nodes
(d) cannot have more than 19 nodes

4. What is the minimum number of edges to be removed from a complete bipartite graph of 6 nodes $K(6)$, so that graph becomes planar?

(a) 1 (b) 3 (c) 4 (d) 6

5. A digraph is represented using adjacency matrix, which is a

(a) Square matrix (b) Symmetric matrix
(c) Asymmetric matrix (d) Unit matrix

6. Merge sort uses

(a) divide and conquer approach
(b) heuristic approach
(c) greedy approach
(d) back tracking approach

7. If B is a circuit matrix of a graph having K components, e edges and n vertices, the rank of B is:

(a) $e + n + k$ (b) $e - n + k$
(c) $e - n - k$ (d) $e + n - k$

8. The number of nodes in a complete binary tree of level 5 is _____.

9. A graph is non-planar if and only if

(a) It does not contain subgraphs homeomorphic to k_5 and $k_{3,3}$.
(b) It does not contain subgraphs isomorphic to k_5 and $k_{3,3}$.
(c) It does contain subgraph isomorphic to k_5 and $k_{3,3}$.
(d) It does contain subgraph homeomorphic to k_5 and $k_{3,3}$.

10. To find all pairs of shortest distance in a graph, we can use

(a) dynamic programming approach
(b) bidirectional search approach
(c) merging approach
(d) recursion approach

11. Sorting is useful for

(a) report generation
(b) memory storage
(c) compiler
(d) making searching easier and efficient

12. A sorting technique that guarantees that records with the same primary key occur in the same order in the sorted list as in the original unsorted list is said to be

(a) internal (b) external
(c) stable (d) inconsistent

13. Stack cannot be used to

(a) evaluate an arithmetic expression in postfix form
(b) call different subroutines of a program
(c) allocate resources(like CPU) by the operating system
(d) implement recursion

14. A sorting method that uses a lesser number of swappings is

(a) quick sort (b) bubble sort
(c) selection sort (d) radix sort

15. Merge sort uses recursion for

(a) postorder traversal of a tree
(b) preorder traversal of a tree
(c) inorder traversal of a tree
(d) both b and c

16. A binary search tree

(a) gives the descending order for pre order traversal
(b) gives the ascending order for in order traversal
(c) gives the ascending order for post order traversal
(d) gives the ascending order for pre order traversal

17. Walk of a graph can be defined as a sequence of nodes and edges in which

(a) edges are not repeated
(b) edges may be repeated
(c) all nodes may be covered
(d) nodes may be repeated

18. A sorting technique that uses binary tree in which the larger number acts as a root of a subtree to all the numbers is called

(a) heap sort (b) quick sort
(c) merge sort (d) bucket sort

19. The reverse polish notation equivalent to the infix expression $((A + B) \times C + D)/(E + F + G)$ is

(a) $AB + C \times D + EF + G + /$
(b) $AB + CD\times + EF\ G +$
(c) $AB + C \times D + EF + G + /$
(d) $AB + CD\times + E + FG + /$

20. Big-O estimate for the logarithm of the factorial function, that is, $\log n!$, is given by

(a) $O(n \log n^2)$ (b) $O(n \log n)$
(c) $O(\log n!)$ (d) $O(\log \sqrt{n})$

21. If h is a hash function used to hash n keys into a table of size m, where $n < m$, then the expected number of collisions involving a particular key K is

(a) Less than 1
(b) Less than $\log n$
(c) Greater than $\log n$
(d) Greater than $\log n + 1$

22. In merge sort, what process is performed before merging?

(a) The array elements form a heap.
(b) Elements in each half of the array are sorted among themselves.
(c) Elements in the first half of the array are made equal to elements in second half of the array.
(d) The arrays are sorted and form a heap.

23. An algorithm is made up of two modules M1 and M2. If time complexities of modules M1 and M2 are $h(n)$ and $g(n)$, respectively, the time complexity of the algorithm is

(a) Mean($h(n)$, $g(n)$) (b) max($h(n)$, $g(n)$)
(c) $h(n) + g(n)$ (d) min($h(n)$, $g(n)$)

24. A vertex cover of an undirected graph $G(V, E)$ is a subset $V_1 \subseteq V$ vertices such that

(a) Each pair of vertices in V_1 is connected by an edge
(b) If $(u, v) \in E$, then $u \in V_1$ and $v \in V_1$
(c) If $(u, v) \in E$, then $u \in V_1$ or $v \in V_1$
(d) All pairs of vertices in V_1 are not connected by an edge

25. Which one of the following is not an informed search technique?

(a) Hill climbing search (b) AO* search
(c) A* search (d) Breadth-first search

26. The map colouring problem can be solved using which of the following techniques?

(a) Depth-first search
(b) Constraint satisfaction
(c) AO* search
(d) Bidirectional search

27. The value of postfix expression 8 3 4 + −3 8 2 / + × 2 3 + is _____.

28. Which of the following connected simple graphs has exactly one spanning tree?

(a) Graph with four nodes and five edges
(b) Graph with five nodes and five edges
(c) Graph with five nodes and four edges
(d) None of the above

29. Assume that there are n keys and each key is in the range $[0, m - 1]$. The runtime of bucket sort is

(a) $O(m - 1)$ (b) $O(m \log n)$
(c) $O(n \log m)$ (d) $O(n + m)$

30. Consider $f(x) = (x+1)\log(x^2+1) + 3x^2$; its time complexity in big-O notation is

(a) $O(x)$ (b) $O(x^2)$
(c) $O(x^3)$ (d) $O(\sqrt{n})$

31. How many edges are there in a forest of t trees containing a total of n vertices?

(a) $n \times t$ (b) $n - t$
(c) n/t (d) t/n

32. Which of the following sorting algorithms is stable?

(a) Quick sort and insertion sort
(b) Insertion sort and bubble sort
(c) Quick sort and heap sort
(d) Quick sort and bubble sort

33. The minimum number of edges in a connected cyclic graph with n vertices is

(a) $\sqrt{n}$ (b) $n - 1$
(c) n (d) $n + 1$

34. If there exists at least one path between every pair of vertices in a graph, the graph is called

(a) Complete graph (b) Hamiltonian graph
(c) Connected graph (d) Mesh

35. The depth of a complete binary tree with n nodes is

(a) $\log(n + 1) - 1$ (b) $\log(n) - 1$
(c) $\log(n - 1)$ (d) $\log(n)$

36. The maximum number of nodes contained by a binary tree of level l is

(a) $2l$ (b) 2^l
(c) l^2 (d) $\log(l - 1)$

37. $f(n) = O(g(n))$ implies

(a) $f(n) = O(g(n))$ only
(b) $f(n) = \Omega(g(n))$ only
(c) $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$
(d) $f(n) = \Theta(g(n))$ only

38. Assuming $P! = \text{NP}$, which of the following is true?

(a) $P = \text{NP}$
(b) NP-complete $\cap P = \Phi$
(c) NP-hard = NP
(d) NP-complete = P

39. Which of the following statements is *true*?

1. The problem of determining whether there exists a cycle in an undirected graph is in P.
2. The problem of determining whether there exists a cycle in an undirected graph is in NP.
3. If a problem A is NP-complete, there exists a non-deterministic polynomial time algorithm to solve A.

(a) 1, 2 and 3 (b) 1 and 3
(c) 2 and 3 (d) 1 and 3

40. Let X is a problem that belongs to the class NP. Which of the following is *true*?

(a) There is no polynomial time algorithm for X.
(b) If X can be solved, then $P = \text{NP}$.
(c) If X is NP-hard, then it is NP-complete.
(d) X may be represented by Bachmann-Landau notations.

41. Which of the following is not the in-place sorting?

(a) Insertion sort (b) Heap sort
(c) Merge sort (d) Selection sort

42. Which of the following does not use the concept of dynamic programming?

(a) Bellman-Ford algorithm
(b) Kruskal's algorithm
(c) Dijkstra's algorithm
(d) Prim's minimum spanning tree

43. We use dynamic programming approach when

(a) It can be implemented on 0–1 knapsack problem
(b) The solution has optimal substructure
(c) The given problem can be reduced to the 3-SAT problem
(d) The brute-force algorithm is not applicable

44. Which of the following algorithms is *not* a divide and conquer algorithm by nature?

(a) Dijkstra's algorithm (b) Heap sort
(c) Selection sort (d) Quick sort

45. What is recurrence for worst case of quick sort and what is the time complexity in worst case?

(a) Recurrence is $T(n) = T(n/2) + O(n/2)$ and time complexity is $O(n^2)$.
(b) Recurrence is $T(n) = T(n-1) + O(n)$ and time complexity is $O(n^2)$.
(c) Recurrence is $T(n) = 2T(n/2) + O(\log n)$ and time complexity is $O(n \log n)$.
(d) Recurrence is $T(n) = T(n/10) + O(n \times 9/10)$ and time complexity is $O(n^3)$.

Set 2

1. Map the following statements regarding quick sort with true (T) or false (F), respectively.

S1: Quick sort is an in-place sorting algorithm.
S2: Quick sort is stable.
S3: Quick sort performs best when numbers are sorted or almost sorted.
S4: The worst running time of quick sort using randomizedpartition is $O(n \log n)$.
S5: It is an example of greedy technique.

(a) TFFFF (b) TTFTT
(c) TFFTF (d) FTFTF

2. What will be the output after the second pass of bubble sort to sort the following number sequence in an ascending order?

3 5 7 6 4 2

(a) 3 5 6 4 2 7 (b) 3 5 4 6 2 7
(c) 3 5 4 2 6 7 (d) 2 3 4 6 5 7

3. The number of spanning trees for a complete graph with seven vertices is

(a) 2^5 (b) 7^5
(c) 3^5 (d) $2^{2\times 5}$

4. If BFS is run on a complete bipartite graph $K_{q,r}$, $q \geq 1$ and $r \geq 1$, starting with a vertex 'X', then maximum possible height of BFS tree, rooted at 'X', is:

(a) 1 (b) 2
(c) $q + r - 1$ (d) Max(q, r)

Common data questions 5 and 6: Consider the following algorithm

```
Add(k){//Assume n is a positive integer values
If(k<=1)
Return k;
Total=0;
For i=1 to k-1
Total =Total+Add(i);
Return Total;}
```

5. If $T(k)$ denotes the time complexity of the algorithm, then $T(k)$ is

(a) $\Theta(n^2)$ (b) $\Theta(n!)$
(c) $\Theta(2^n)$ (d) $\Theta(n^3)$

6. If $S(k)$ denotes the space complexity of the algorithm, then $S(k)$ (in terms of $T(k)$) is

(a) $\Theta(\log \log(T(n)))$
(b) $\Theta(\log(T(n)))$
(c) $\Theta(\log(T(n))/\log(n))$
(d) $\Theta\sqrt{(T(n))}$

7. If A is the best algorithm to check whether a given simple undirected graph $G(V, E)$ is a tree or not, then which of the following best describes the time complexity $T(V, E)$ of A? (Assume that G is represented as adjacency list.)

(a) $T(V, E)$=O($|V|$)
(b) $T(V, E)$= O($|V|$+$|E|$)
(c) $T(V, E)$= $\Theta(|V|^2)$
(d) $T(V, E) = \Theta\log(|V|+|E|)$

8. Express the following recurrence relation in asymptotic notation: $T(n) = 4T(n/4) + \sqrt{n}$.

(a) $\Theta(\sqrt{n})$ (b) $\Theta(n)$
(c) $\Theta(n^2)$ (d) $\Theta(n^{3/2})$

9. The depth of the binary tree with n nodes is

(a) log $(n + 1) - 1$ (b) log n
(c) $\log(n - 1) + 1$ (d) $\log(n) + \log(n - 1)$

10. Solve the following given recurrence relation: $T(n) = 2T(n/2) + n^2$.

(a) $\Theta(n^2)$ (b) $\Theta(n^2 \log_n n)$
(c) $\Theta(n \log_2 n)$ (d) None of the above

11. Given

(i) $\frac{1}{2}n^2 = \omega(n)$

(ii) $\frac{1}{2}n^2 = \omega(n^2)$

Which of the following is true?
(a) I is correct.
(b) II is correct.
(c) Both I and II are correct.
(d) None of them is correct.

12. $f(n)$=O($g(n)$) if and only if

(a) $g(n)$= O($f(n)$) (b) $g(n)$= $\omega(f(n))$
(c) $g(n)$= $\Omega(f(n))$ (d) None

13. What is the generating function $G(z)$ for the sequence of Fibonacci number?

(a) $\frac{z}{z^2 - z - 1}$ (b) $\frac{z}{1 - z - z^2}$
(c) $\frac{z}{1 + z - z^2}$ (d) $\frac{z}{1 + z + z^2}$

14. $\sum_{i=0}^{n} O(n)$, where O(n) stands for order n is

(a) O(n) (b) O(n^2) (c) O(n^3) (d) O($3n^2$)

15. In the worst case, the number of comparisons needed to search singly linked list of length n for a given element is

(a) $\log_2 n$ (b) $n/2$ (c) $\log_2 n - 1$ (d) n

16. Consider the following algorithm for searching for a given number x in an unsorted array $A[1 \ldots n]$ having n distinct values:

```
1. Choose an i uniformly at random from 1....n;
2. If A[i] = x then Stop else Goto 1;
```

Assuming that x is present on A, what is the expected number of comparisons made by the algorithm before it terminates?
(a) n (b) $n - 1$ (c) $2n$ (d) $n/2$

17. The running time of the following algorithm:

```
Procedure A(n)
If n < = 2 return (1) else return (A(|√n|))
```

is best described by
(a) O(n) (b) O(log n)
(c) O(log log n) (d) O(1)

18. Let P be the quick sort program to sort numbers in ascending order. Let t_1 and t_2 be the time taken by the program for the input [1 2 3 4] and [5 4 3 2 1], respectively. Which of the following holds?

(a) $t_1 = t_2$ (b) $t_1 > t_2$
(c) $t_1 < t_2$ (d) $t_1 = t_2 + 5\log 5$

19. Which of the following algorithm(s) can be used to sort n integers in the range $[1 \ldots n^3]$ in $O(n)$ time?

(a) Heap sort (b) Quick sort
(c) Merge sort (d) Radix sort

20. The recurrence relation that arises in relation with the complexity of binary search is

(a) $T(n) = T(n/2) + k$, where k is a constant
(b) $T(n) = 2T(n/2) + k$, where k is a constant
(c) $T(n) = T(n/2) + \log n$
(d) $T(n) = T(n/2) + n$

21. Which one of the following statements is false?

(a) Optimal binary search tree construction can be performed efficiently using dynamic programming.
(b) Breadth-first search cannot be used to find connected component of a graph.
(c) Given the prefix and postfix walks over a binary tree, the binary tree cannot be uniquely constructed.
(d) Depth-first search can be used to find connected components of a graph.

22. The recurrence relation $T(1) = 2$ and $T(n) = 3T(n/4) + n$ has the solution $T(n)$ equal to

(a) $O(n)$ (b) $O(\log n)$
(c) $O(n^{3/4})$ (d) None of the above

23. Quick sort is run on two inputs shown below to sort in ascending order

(i) 1, 2, 3, …, n
(ii) n, $n-1$, $n-2$, …, 2, 1

Let C_1 and C_2 be the number of comparisons made for the inputs (i) and (ii), respectively. Then

(a) $C_1 < C_2$
(b) $C_1 > C_2$
(c) $C_1 = C_2$
(d) We cannot compare for arbitrary n

24. Let $T(n)$ be the function defined by $T(1) = 1$, $T(n) = 2T\left(\lceil n/2 \rceil\right) + \sqrt{n}$ for $n \geq 2$. Which of the following statements is true?

(a) $T(n) = O(\sqrt{n})$ (c) $T(n) = O(\log n)$
(b) $T(n) = O(n)$ (d) None of the above

25. A sorting technique is called stable if

(a) it takes $O(n \log n)$ time
(b) it maintains the relative order of occurrence of non-distinct elements
(c) it uses divide and conquer paradigm
(d) it takes $O(n)$ space

26. If one uses straight two-way merge sort algorithm to sort the following elements in ascending order: 20, 47, 25, 8, 9, 4, 40, 30, 12, 17, then the order of these elements after second pass of the algorithm is

(a) 8, 9, 15, 20, 47, 4, 12, 7, 30, 30
(b) 8, 15, 20, 47, 4, 9, 30, 40, 12, 17
(c) 15, 20, 47, 4, 8, 9, 12, 30, 40, 17
(d) 4 8, 9, 15, 20, 47, 12, 17, 30, 40

27. Randomized quick sort is an extension of quick sort where the pivot is chosen randomly. What is the worst-case complexity of sorting n numbers using randomized quick sort?

(a) $O(n)$ (b) $(n \log n)$
(c) $O(n^2)$ (d) $O(n!)$

28. Kruskal's algorithm for finding a minimum spanning tree of a weighted graph G with n vertices and m edges has the time complexity of

(a) $O(n^2)$ (b) $O(mn)$
(c) $O(m^2)$ (d) $O(m \log n)$

29. A binary search tree is generated by inserting in order the following integers: 50, 15, 62, 5, 20, 58, 91, 3, 8, 37, 60, 24. The number of nodes in the left subtree and right subtree of the root, respectively, is

(a) (4, 7) (b) (7, 4)
(c) (8, 3) (d) (3, 8)

30. Which one of the following algorithm design techniques is used in finding all pairs of shortest distances in a graph?

(a) Dynamic programming (b) Backtracking
(c) Greedy (d) Divide and conquer

31. Give the correct matching for the following pairs:

A. $O(\log n)$	P. Selection
B. $O(n)$	Q. Insertion sort
C. $O(n \log n)$	R. Binary search
D. $O(n^2)$	S. Merge sort

(a) A – R; B – P; C – Q; D – S
(b) A – R; B – P; C – S; D – Q
(c) A – P; B – R; C – S; D – Q
(d) A – P; B – S; C – R; D – Q

32. Consider an undirected unweighted graph G. Let a breadth-first traversal of G be done starting from a node r. Let d(r, u) and d(r, v) be the lengths of the shortest paths from r to u and v, respectively, in G. If u is visited before v during the breadth-first traversal, which of the following statements is correct?

(a) $d(r, u) < d(r, v)$
(b) $d(r, u) > d(r, v)$
(c) $d(r, u) \leq d(r, v)$
(d) None of the above

33. How many undirected graphs (not necessarily connected) can be constructed out of a given set $V = \{V_1, V_2, \ldots, V_n\}$ of n vertices?

(a) $n(n-1)/2$ (b) 2^n
(c) $n!$ (d) $2^{(n(n-1)/2)}$

34. The most appropriate matching for the following pairs is:

X: depth-first search	1: heap
Y: breadth-first search	2: queue
Z: sorting	3: stack

(a) X– 1; Y– 2; Z– 3 (b) X– 3; Y– 1; Z– 2
(c) X– 3; Y– 2; Z– 1 (d) X– 2; Y– 3; Z– 1

ANSWERS TO PRACTICE EXERCISES

Set 1

1. (c)	**9.** (d)	**17.** (b)	**25.** (d)	**33.** (c)	**41.** (c)
2. (d)	**10.** (a)	**18.** (a)	**26.** (b)	**34.** (c)	**42.** (d)
3. (c)	**11.** (d)	**19.** (a)	**27.** (52)	**35.** (a)	**43.** (b)
4. (b)	**12.** (c)	**20.** (b)	**28.** (d)	**36.** (b)	**44.** (b)
5. (c)	**13.** (c)	**21.** (a)	**29.** (d)	**37.** (c)	**45.** (b)
6. (a)	**14.** (c)	**22.** (b)	**30.** (b)	**28.** (b)	
7. (c)	**15.** (b)	**23.** (b)	**31.** (b)	**39.** (a)	
8. (63)	**16.** (b)	**24.** (c)	**32.** (c)	**40.** (c)	

Set 2

1. (a) Quick sort is an in-place sorting technique but not stable. It performs worst when numbers are sorted. The worst running time of quick sort using randomized partition is $O(n^2)$. Quick sort is an example of a divide-and-conquer algorithm technique.

2. (c)

Pass 1:	Pass 2:	After second pass:
3 5 7 6 4 2	3 5 6 4 2 7	3 5 4 2 6 7
3 5 7 6 4 2	3 5 6 4 2 7	
3 5 6 7 4 2	3 5 4 6 2 7	
3 5 6 4 2 7	3 5 4 2 6 7	

3. (b) The number of spanning trees for K_n is $n^{n-2} = 7^{7-2} = 7^5$.

4. (b) Let the two sets be Q and R and if $X \in Q$, then the BFS tree contains X in level 0, all the nodes of R in level 1 and remaining nodes of Q other than X in level 2, so maximum height is 2.

5. (c) The recurrence relation $T(n)$ for time complexity of linear(n) is

$$T(k) = c + \sum_{i=1}^{k-1} T(i), \quad \text{if } k > 1 = c$$

$$= \text{else}$$

$$T(k) = c + \sum_{i=1}^{k-2} T(i) + T(k-1), \quad \text{for } k > 1$$

$$T(k) = T(k-1) + T(k-1) = 2 \times T(k-1)$$

So

$$T(k) = 2^{k-1}c$$

6. (b) Space complexity of large(k) is proportional to the height of recursion tree which is k. Thus, $S(k) = k = \Theta(\log(T(k))$, since $T(k) = \Theta(2^k)$.

7. (a) Can be done using DFS in $O(|V|)$ time.

8. (b) By case 1 of the Master Theorem

9. (a) The depth of a binary tree with n nodes is

$$2^{d+1} - 1 = N$$

So, we have

$$2^{d+1} = N + 1$$

Taking log, we get

$$(d+1)\log 2 = (N+1)$$

$$d + 1 = \log_2(N+1)$$

$$d = \log_2(N+1) - 1$$

10. (a) From the master theorem, here we have $a = 2$, $b = 2$ and $f(n) = n^2$. So

$$g(n) = n^{\log b \cdot a} = n$$

Here, order of $f(n) >$ order of $g(n)$, so we get

$$T(n) = O(f(n)) = O(n^2) \approx \Theta(n^2)$$

11. (a) For ω, there is $f(n) > cg(n)$. So there is no equality. Thus, I is correct and II is false.

12. (c) We have $f(n) = O(g(n))$ which means $f(n) < cg(n)$

It is true only if $g(n) > c(f(n))$ which means $g(n) = \Omega(f(n))$

13. (b) The generating function for the Fibonacci number $G(z)$ is

$$G(z) = \frac{z}{1 - z - z^2}$$

14. (b) We have

$$\sum_{i=0}^{n} O(n) = O(1) + O(2) + O(3) + \cdots + O(n)$$

$$= O\left(\frac{n(n+1}{2}\right) = O(n^2)$$

15. (d) The worst-case search time in a singly linked list is when the element is in the last position or the element does not exist in the linked list. So it is similar to linear search and requires 'n' comparisons in the worst case.

16. (a) (Number of comparisons for a case × probability for the case)

Case 1: If $A[i]$ is found in the first attempt, number of comparisons = 1. Thus, probability of the case = $1/n$.

Case 2: If $A[i]$ is found in the second attempt, number of comparisons = 2. Thus, probability of the case is given as

$$\frac{n-1}{n} \times \frac{1}{n}$$

Case 3: If $A[i]$ is found in the third attempt, number of comparisons = 3. Thus, probability of the case is given as

$$\frac{n-1}{n} \times \frac{n-1}{n} \times \frac{1}{n}$$

There are actually infinite such cases. So, we have the following infinite series:

$$\frac{1}{n} + \frac{n-1}{n} \times \frac{1}{n} \times 2 + f = \frac{n-1}{n} \times \frac{n-1}{n} \times \frac{1}{n} \times 3 + \cdots$$

After solving this series, we will get 'n' comparisons.

17. (c) Recursive relation for procedure $A(n)$ is

$$T(n) = T(\sqrt{n}) + c1 \quad \text{if } n > 2$$

Use substitution method to solve this.

18. (a) Quick sort takes the worst time if the elements are already in sorted orders (it could be in ascending or descending). As both the inputs are already sorted, it will take the same time in both the cases.

19. (d) Quick sort, merge sort and heap sort running time are (n log n), whereas radix sort is a non-comparative integer sorting algorithm that sorts data in $O(n)$ time.

20. (a) In binary search, only half of the array where the element exists is searched and the other half is left. So, we get

$$T\left(\frac{n}{2}\right) + K$$

21. (b) Depth-first search and breadth-first search are capable of finding whether a given graph is connected and are also capable of finding connected components of a graph in linear time. So, option (b) is false.

22. (a) Using case 1 of master theorem, we get $n^{\log_4 3} < n$

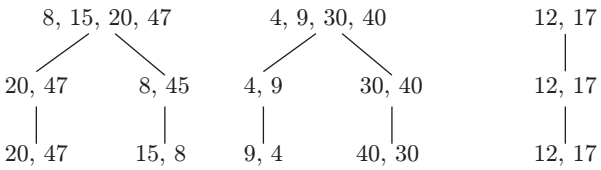
23. (c) Both of the given cases are quick sort worst-case problems, so the number of comparisons will be equal.

24. (b) Use case 1 of master theorem.

25. (b) A sorting algorithm is said to be stable if two objects with equal keys appear in the same order in the sorted output as they appear in the input unsorted array. Some sorting algorithms are stable by nature such as insertion sort, merge sort, bubble sort, etc. And some sorting algorithms are not, such as heap sort, quick sort, etc.

26. (b) Given: 20, 47, 15, 8, 9, 4, 40, 30, 12, 17.

Here, two-way merge sort is used, so a group of two is taken at once. The second pass is shown as follows:

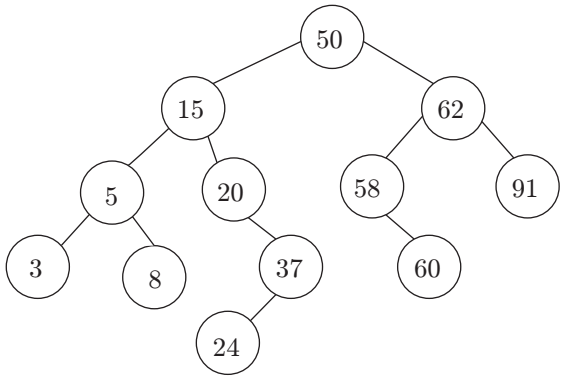


After second pass of the algorithm element, the order is 8, 15, 20, 47, 4, 9, 30, 40, 12, 17.

27. (c) Randomized quick sort worst-case complexity is $O(n^2)$; when all the elements are same, it took worst time.

28. (d)

29. (b) Look at the following figure:



The number of nodes in the left subtree and the right subtree of the root, respectively, is (7, 4). Therefore, option (b) is correct.

30. (a) Bellman-Ford algorithm is used to find all-pair shortest distances in a graph, and it is a dynamic programming technique.

31. (b) Merge sort took $O(n \log n)$ to sort a list of element. Binary search time complexity is $O(\log n)$. Insertion sort worst-case running time complexity is $O(n^2)$.

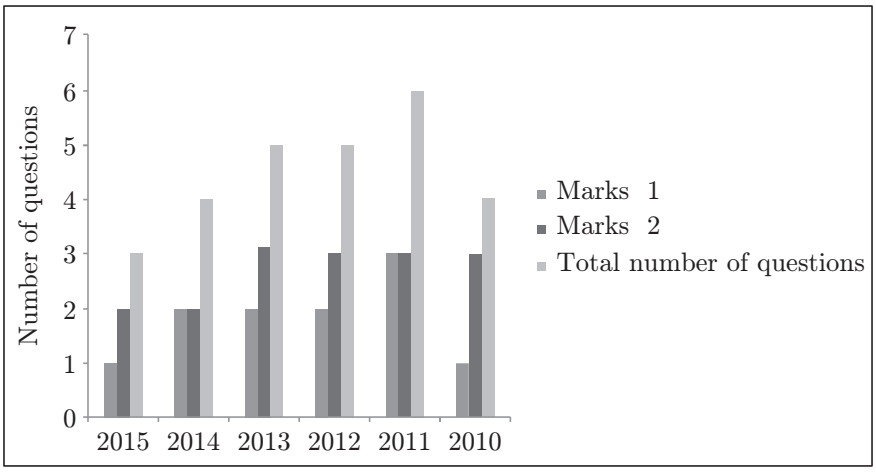
32. (c) $d(r, u)$ and $d(r, v)$ will be equal when u and v are at same level; otherwise, $d(r, u)$ will be less than $d(r, v)$.

33. (d) In an undirected graph, if it is a complete graph, then it has $n(n-1)/2$ edges. We can choose to have (or not have) any of the $n(n-1)/2$ edges. So, the total number of undirected graphs with n vertices is $2^{(n(n-1)/2)}$.

34. (c) Depth-first search uses stack for storing values; queue is used by breadth-first search. Heap is used for sorting the set of elements.

UNIT V: THEORY OF COMPUTATION

LAST SIX YEARS' GATE ANALYSIS



Concepts on which questions were asked in the previous six years

Year	Concepts
2015	Turing machine, Regular expression and languages, DFA, Context free grammar, Recursive language, NDPDA
2014	Regular languages, DFA, RE and REC
2013	Regular languages, DFA, CFL, CFG
2012	Regular languages, DFA and NFA, CFL
2011	Power of DFA, NFA and TM, Regular language, PDA
2010	Regular expression, Recursive language, LR and LL, CFL

CHAPTER 5

THEORY OF COMPUTATION

Syllabus: Regular languages and finite automata, context-free languages and pushdown automata, recursively enumerable sets and Turing machines, undecidability.

5.1 INTRODUCTION

Theory of computation is a branch of computer science and mathematics that studies whether a problem can be solved using an algorithm on a model of computation. This branch also studies that how efficiently that problem can be solved. This deals with two types of theories. First is computability theory, deals with up to which extent problem can be solved. Second is complexity theory, which deals with efficiency of solution. In this finite automata, push down automata, Turing machines, regular expressions and various grammars will be discussed.

Automata theory is the study of abstract machines (or more appropriately, abstract "mathematical" machines or systems) and the computational problems that can be solved using these machines. These abstract machines are called automata. Computability theory deals with the question of the extent to which a problem is solvable on a computer, whereas complexity theory considers not only whether a problem can be solved at all on a computer, but also how efficiently it can be solved. Two major aspects are considered in this–time complexity and space complexity.

Basic introduction and various technical terms for understanding of concept are described in the following section.

5.2 FINITE AUTOMATA

There are several things to be done in the designing and implementation of an automatic machine, but the most important question is that, what will be the behaviour of the machine? "Theory of Computation" is the subject which solves this purpose.

"Automata theory is the subject which describes the behaviour of automatic machines mathematically". In other words, we can say that "Automata Theory is the study of self-operating virtual machines to help in logical understanding of input and output process without or with intermediate stages of computation".

The model of automation is shown in Fig. 5.1.

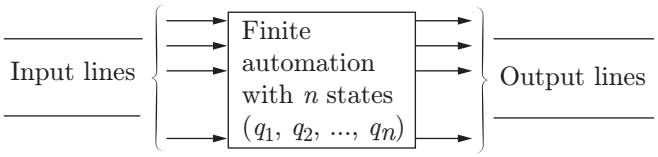


Figure 5.1 | Model of automation.

5.2.1 Finite Automaton Model Characteristics

The characteristics of finite automaton are described below:

1. **Input:** Input values from input alphabet Σ contains I_1, I_2, I_3, ... I_n at discrete time. are the input values from the input alphabet Σ at discrete time.
2. **Output:** Output O_1, O_2, O_3, ...; O_n is the output set for this computation model are different outputs of this model.
3. **States:** At any instance of time, the finite automaton will be in one of the states q_1, q_2, q_3, ..., q_n, are the set of states on which the finite automaton will be at some instance of time.
4. **States relation:** State relation describes that how to reach on next state using present input and output. At any instance of time, the next state of automaton is determined by the present input and present state.
5. **Output relation:** Either one state or number of states can be obtained as output. The next state is achieved using state relation. Either only state or both the input and state are obtained as output. When an automaton reads an input symbol, it moves to the next state, which is given by the state relation.

5.2.2 Technical Terms

Technical terms that are required in finite automaton are given as following:

1. **Alphabets:** A finite non-empty set of symbols are the alphabets of a language.

Example 5.1

$$\Sigma = \{a, b\}$$

$$\Pi = \{0, 1\}$$

Σ and Π are the sets which hold two alphabets. They are called binary alphabets for language.

$$(a+b)^* = \Sigma^* = \{\varepsilon, a, b, aa, bb, ab, ba, \ldots\}$$

where ε is called the identity element. Here, Σ^* contains every string possible by a, b.

$\Sigma^+ = \Sigma^* - \{\text{Null character } (\lambda)\}$, so Σ^+ will contain $\{a, b, aa, bb, ab, \ldots\}$.

2. **Strings:** Sequences of zero or more symbols are called string of a language. Let there be languages such as

   $L_1 = a^*$, then it will consist $\{\varepsilon, a, aa, aaa, aaaa, \ldots\}$ string set.
   $L_2 = ab^*$, then the string will be $\{a, ab, abb, abbb, \ldots\}$.
   $L_3 = 0^*1^*$, then it will consist $\{\varepsilon, 0, 1, 01, 001, 0001, 011, 0111, \ldots\}$ string set.
3. **Concatenation:** Let there be two strings $U = ab$ and $V = aab$. Then concatenation of these two languages is $UV = abaab$ and $VU = aabab$. Concatenations of two strings satisfy associative property but not commutative. So, $UV \neq VU$.
4. **Length of a string:** Let U, V and W be three strings. The length of a string tells the number of alphabets in that string.

   If $U = 011$, length of $|U| = 3$.
   If $V = ababa$, length of $|V| = 5$.
   If $W = abababb$, length of $|W| = 7$.
5. **Reversal of a string:** Let there be a string $W = aab$, then the reverse of W is

   $$W^r = baa$$

   If $W = W^r$, then the string is a palindrome. WW^r is called an even palindrome, and $W \times W^r$ is called an odd palindrome, where x is an alphabet. Let there be two strings U and V, then the reverse of UV is

   $$|UV|^r = V^r U^r$$
6. **Power of a string:** Let there be a string W, then

   $W^0 = \varepsilon$
   $W^1 = W$
   $W^2 = W \cdot W$
   $W^3 = W \cdot W^2$, so $W^n + 1 = W \cdot W^n = W^n \cdot W$

   Let there be a string $U = \{010\}$, then the power of U:

   $$U^0 = \varepsilon$$
   $$U^1 = \{010\}$$
   $$U^2 = \{010010\}$$
7. **Substring**: Any set of conjugative symbols is called a substring.

Example 5.2

Let $W = \{abc\}$ is a string. Substrings of W are (ε, a, b, c, ab, bc, abc), where ε is a substring of every string. If the length of a string is n, then the number of substrings possible is

$$\frac{n(n+1)}{2} + 1$$

8. **Prefix of a string:** Prefix of a string $W = \{x | xy = W\}$

   Consider a string $W = \{aabbcd\}$.
   Prefix of $W = \{\varepsilon, a, aa, aab, aabb, aabbc, aabbcd\}$
   If the length of a string is n, then the number of prefix of that string will be $(n + 1)$.

9. **Suffix of a string:** Suffix of a string $W = \{y \mid xy = W\}$

   Consider a string $W = \{aabbcd\}$.
   Suffix of $W = \{aabbcd, abbcd, bbcd, bcd, cd, d, \varepsilon\}$
   If the length of string is n, then the number of suffix of that string will be $(n + 1)$.

10. **Language:** A language is a collection of strings of finite length constructed from alphabets of symbol. Or, we can say that a language is a subset of every string made by selected alphabets.

    Let an alphabet set be $\{0, 1\}$, then Σ^* will contain all the strings made by 0 and 1. So language made by 0 and 1 will be the subset of Σ^*. Language $L \subseteq \Sigma^*$.

11. **Language union:** Let $L_1 = \{ab, ba, aab\}$ and $L_2 = \{ab, abb, bb\}$ be two languages, then $L_1 \cup L_2 = \{ab, ba, aab, abb, bb\}$.

12. **Language intersections:** Let $L_1 = \{ab, ba, aab\}$ and $L_2 = \{ab, abb, bb\}$ be two languages, then $L_1 \cap L_2 = \{ab\}$.

13. **Language complement:** Let L be a language and L' the complement of language L. So, $L' = \{\Sigma^* - L\}$

Example 5.3

Given language L as $\{aa, ab, baa\}$, then complement of language L $(L') = \{a, b\}^* - \{aa, ab, baa\}$.

14. **Language subtractions:** Let $L_1 = \{ab, ba, baa\}$ and $L_2 = \{aab, ba\}$ be two languages.

    Then $L_1 - L_2 = \{ab, baa\}$ and $L_2 - L_1 = \{aab\}$.

15. **Language XOR:** Let $L_1 = -ab, ba, baa''$ and $L_2 = -aab, ba''$ be two languages.

    $L_1 \oplus L_2 = (L_1 - L_2) \cup (L_2 - L_1) = (L_1 \cup L_2) - (L_1 \cap L_2)$
    $L_1 \oplus L_2 = \{ab, baa, aab\}$

16. **$L_1 \cdot L_2$:** Let $L_1 = \{ab, ba, aab\}$ and $L_2 = \{ba, ab\}$, then

    $L_1 \cdot L_2 = \{abba, abab, baba, baab, aabba, aabab\}$

17. **L^* and L^+:**

    $L^* = \{L^0 \cup L^1 \cup L^2 \cup L^3 \ldots\}$
    $L^+ = \{L^* - \lambda\}$

5.2.3 Grammar

Grammar is a set of rules which generates every string in a language. A grammar is described by four terms $G = (V, T, S, P)$, where

$V \rightarrow$ finite non-empty set of variables (represented by capital letters)

$S \rightarrow$ starting symbol

$T \rightarrow$ finite non-empty set of terminals (represented by small letters)

$P \rightarrow$ finite non-empty set of production rule

Example 5.4

$$S \rightarrow AB \text{ (production 1)}$$
$$A \rightarrow a \text{ (production 2)}$$
$$B \rightarrow b \text{ (production 3)}$$

where S is the starting point, A and B are variables and a, b are terminals. We have three productions here. In grammar, V, T, P should be non-empty and finite.

5.2.4 Noam Chomsky Grammar Classification

Grammar is classified into the following four types:

1. **Type 0:** This type of grammar is unrestricted, or phase structured, which generates all types of languages that can be recognized by a Turing machine. These languages are called recursive enumerable language. Unrestricted means these grammar have very less restrictions. Unrestricted grammar $(G) = u \rightarrow v$, where $u, v \in (V \cup T)^*$ and "u" must have at least one variable. V stands for variable and T for terminals.

Example 5.5

$$S \rightarrow aSb/\lambda$$
$$A \rightarrow aA/aB/Cb$$
$$Ba \rightarrow a/Da$$

2. **Type 1:** This is context-sensitive grammar (CSG) which is used to generate context-sensitive languages. CSG $(G) = u \rightarrow v$, where $u, v \in (V \cup T)^*$ and "u" must have at least one variable. There is one additional rule also, $|u| \leq |v|$, which means length of "u" is less than or equal to "v".

Example 5.6

$$S \rightarrow aSb/\lambda$$
$$Aa \rightarrow aA/aB/Cb$$
$$B \rightarrow a/Da$$

3. **Type 2:** This is context-free grammar (CFG) which is used to generate context-free languages. CFG $(G) = u \rightarrow v$, where $v \in (V \cup T)^*$ and $|u| \leq |v|$. In addition to the CSG grammar, one extra rule, that is, only single variable is allowed in "u".

Example 5.7

$$S \rightarrow aSb/\lambda$$
$$A \rightarrow aA/aB/Cb$$
$$B \rightarrow a/Da$$

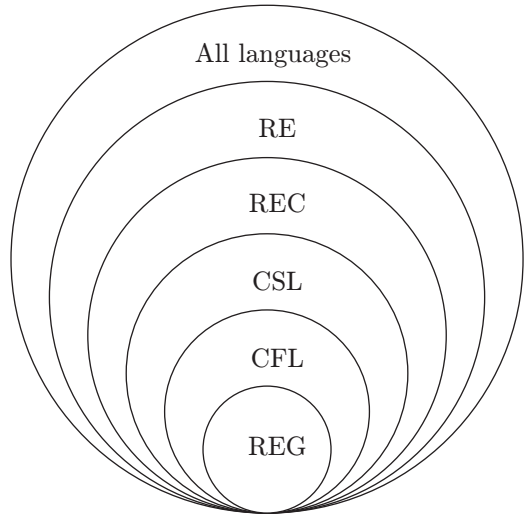
4. **Type 3:** This is regular grammar, therefore language generated by these grammars is called regular languages. Regular grammar $(G) = u \rightarrow v$, where $|u| \leq |v|$, $\{v \in (T^*, T^*V, VT^*)$ and $u \in V\}$ and "u" has only one variable.

Example 5.8

$$S \rightarrow aS/\lambda$$
$$A \rightarrow aA/aB/Cb$$
$$B \rightarrow a/Da$$

5.2.5 Chomsky Hierarchy

The hierarchy of grammar was described by Noam Chomsky in 1956, shown in Fig. 5.2.



REG—regular language; CFL—context-free language; CSL—context-sensitive language; REC—recursive language; RE—recursive enumerable.

Figure 5.2 | Chomsky hierarchy.

5.2.6 Different Grammar Types

There have been numerous grammar types proposed for different languages. Table 5.1 provides a summary of the same.

Table 5.1 | Comparison of different types of grammar

Type	Grammar	Accepted by	Restriction on Production ($x \rightarrow y$)	Example
Type 0	Unrestricted Grammar	Turing Machine	1. x must have at least one variable (V) 2. y can be any string which consist any combination of variable (V) and terminal (T)	AB → a | Ba
Type 1	Context Sensitive Grammar	Linear Bounded Automata	1. x must have at least one variable (V) 2. y can be any string which consist combination of variable (V) and terminal (T) 3. Length of y should be less than or equal to x.	Aa → aAb | aB | Cb
Type 2	Context Free Grammar	Push Down Automata	1. x can have exactly one Variable (V) 2. y can be any string which consist combination of variable (V) and terminal (T) 3. Length of y should be less than or equal to x.	A → aBa | aB | Ba
Type 3	Regular Grammar	Finite Automata	1. x can have exactly one Variable (V) 2. y can be any string which consist any combination from (VT*, T*V, T*), where V = one variable and T* = any number of terminals 3. Length of y should be less than or equal to x.	A → a | Bb | bB

5.3 FINITE AUTOMATA AND REGULAR LANGUAGE

Finite automata is classified into two categories–deterministic finite automata and non-deterministic finite automata.

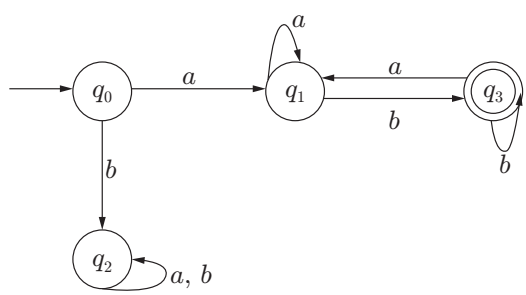
5.3.1 Deterministic Finite Automata

A deterministic finite automata (DFA) is defined by a five-tuple set $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite non-empty set of states.
2. Σ is a finite non-empty set of input called alphabets.
3. δ is a transition function which maps $Q \times \Sigma \rightarrow Q$.
4. $q_0 \in Q$, known as the initial state or starting state.
5. $F \subseteq Q$, known as a set of final states.

Example 5.9

Consider a DFA $M = (\{q_0, q_1, q_2, q_3\}\ \{a, b\}\ \{\delta\}\ \{q_0\}\ \{q_3\})$ as shown in the figure below.



The DFA can be represented in a transition table shown below, which can be determined by the transition function (δ).

δ	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_1	q_3
q_2	q_2	q_2
q_3	q_1	q_3

q_0 – Starting state

q_3 – Final state (accepting state)

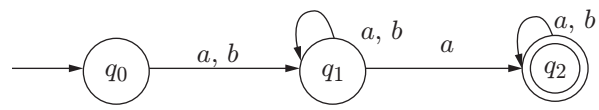
5.3.2 Non-deterministic Finite Automata

A non-deterministic finite automata (NFA) is defined by a five-tuple set $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite non-empty set of states.
2. Σ is a finite non-empty set of input called alphabets.
3. δ is a transition function which maps $Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$.
4. $q_0 \in Q$, known as the initial state or starting state.
5. $F \subseteq Q$, known as a set of final states.

Example 5.10

Consider an NFA $M = (\{q_0, q_1, q_2\}\ \{a, b\}\ \{\delta\}\ \{q_0\}\ \{q_2\})$ as shown in figure below.



The NFA can be represented in a transition table shown below, which can be determined by the transition function (δ).

δ	a	b
$\rightarrow q_0$	q_1	q_1
q_1	$\{q_1, q_2\}$	q_1
q_2	q_2	q_2

q_0 – Starting state

q_2 – Final state (accepting state)

5.3.3 Comparison of DFA and NFA

The comparison between DFA and NFA is given in Table 5.2.

5.3.4 DFA and NFA Design

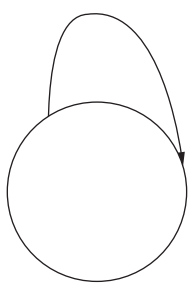
We have taken a few examples and discussed how to design a DFA and an NFA for these cases. The following are four types of states which are used to design a DFA and an NFA:

1. **Permanent accept state:** In this case, the initial and final states are the same. Due to self-loop this machine will always be on the same state, which is also final state. Hence, this is called permanent accept state.

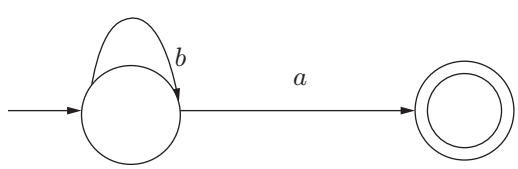
Table 5.2 | DFA vs. NFA

Title	DFA	NFA
Power	NFA and DFA powers are same	NFA and DFA powers are same
Supremacy	Some NFA are DFA, but not all	All DFA are NFA
Time complexity	Time needed to execute an input string is less than that required by NFA	Time needed to execute an input string is more than that required by DFA
Transition function	Maps $Q \times \Sigma \rightarrow Q$, number of next states at an input is exactly one	Maps $Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$, the number of next states at an input is zero, one or more.
Null moves	Does not allow null moves	Allows null moves
Moves	Only one move for single input alphabet	There can be choice (more than one move can be there) for single input alphabet

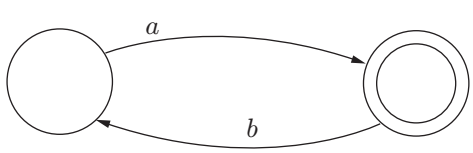
2. **Permanent reject state (trap state):** The given state is permanent reject state because there is self-loop to some non-final state.



3. **Temporary reject state:** The given state is temporary reject state because for input 'b', it has a loop on the non-final state. However, if after input 'b' there is another input 'a', then it will halt on final state.

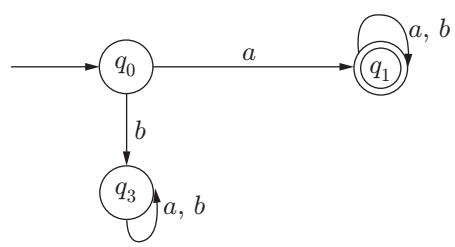


4. **Temporary accepting state:** The given state is temporary accepting state, as it reaches the final state if it has input symbol 'a', but if it gets another input symbol 'b' on that final state it will halt on non-final state.



Example 5.11

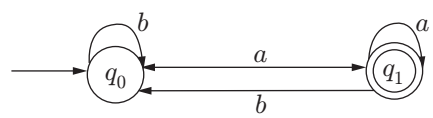
DFA which accepts string starting with "a"



Regular expression: $a(a + b)^*$

Example 5.12

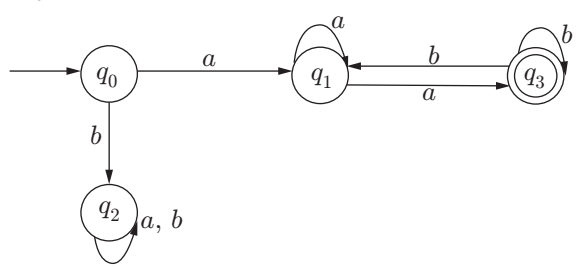
DFA which accepts string ending with "a"



Regular expression: $(a + b)^*a$

Example 5.13

DFA which accepts string starting with "a" and ending with "b"

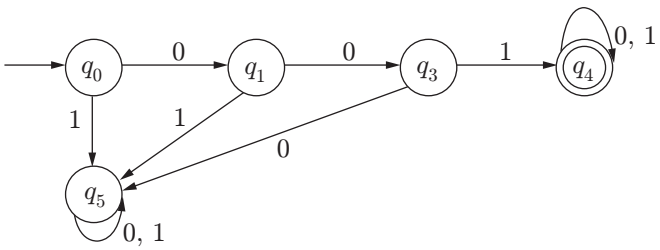


Regular expression:

1. $a(a + b)^* \cap (a + b)^*b$ (from statement)
2. $aa^*b(b + aa^*b)^*$

Example 5.14

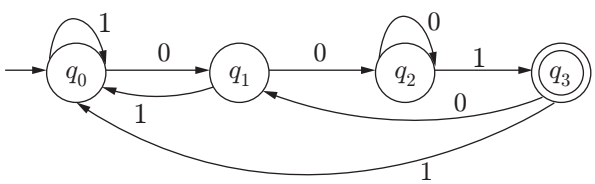
DFA which accepts string starting with "001"



Regular expression: 001(0 + 1)*

Example 5.15

DFA which accepts string ending with "001"



Regular expression: (0 + 1)*001

5.3.5 Applications of DFA/NFA

NFA and DFA have various applications as given below:

1. Lexical analyser
2. Text editor
3. Spell checker
4. Sequential circuit design
5. Unix graph (pattern search)

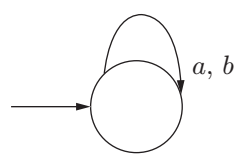
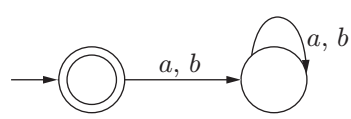
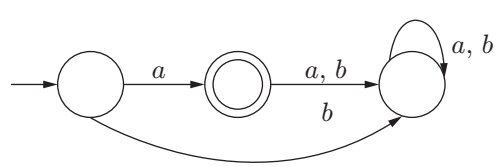
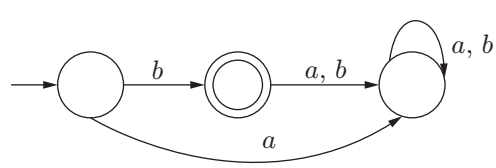
5.3.6 Regular Expression and Grammar

Finite automata accepts regular grammar. First of all, regular expressions are discussed and then how to construct regular grammar from finite automaton is described (Table 5.3).

1. **Regular expression:** A language is regular iff $\exists$ a regular expression r, that is, $L = L(r)$.
   - There can be more than one regular expression for a language L.
   - There can be more than one machine for a language L.
   - A language L can be produced by more than one grammar.

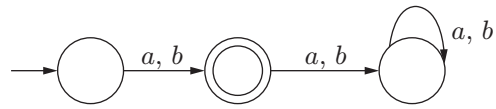
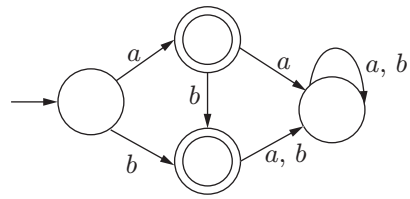
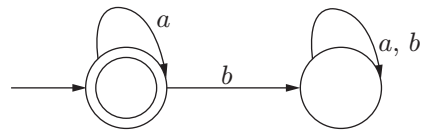
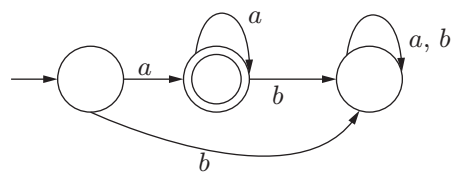
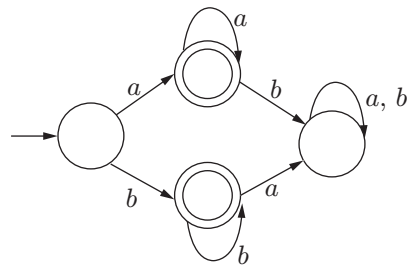
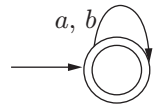
Symbols to represent regular expression are $\{a, b, \varepsilon, \Sigma, \varnothing, \lambda, +, ., *, (\ )\}$.

Table 5.3 Some standard regular languages and their regular expression, grammar and machine

Regular Expression (r)	Language (L)	Grammar (G)	Machine (M)
$\varnothing$	$\{\}$	$S \to A$	
Λ	$\{\lambda\}$	$S \to \varepsilon$	
A	$\{a\}$	$S \to a$	
B	$\{b\}$	$S \to b$	

(*Continued*)

Table 5.3 Continued

Regular Expression (r)	Language (L)	Grammar (G)	Machine (M)
$a + b$	$\{a, b\}$	$S \to a$ $S \to b$	
$a + b + ab$	$\{a, b, ab\}$	$S \to aab$ $S \to ab$ $S \to ba$	
a^*	$\{\lambda, a, aa, aaa, aaaa, \ldots\}$	$S \to aS$ $S \to \varepsilon$	
a^+	$\{a, aa, aaa, aaaa, \ldots\}$	$S \to aS$ $S \to a$	
$a^* + b^*$	$\{\lambda, a, aa, aaa, \ldots, b, bb, bbb, \ldots\}$	$S \to S_1/S_2$ $S_1 \to aS_1/\varepsilon$ $S_2 \to bS_2/\varepsilon$	
$(a + b)^*$	$\{\lambda, a, b, ab, ba, aab, aaab, baa, bab, bbb, baba, \ldots\}$	$S \to aS/bS$ $S \to a/b$	

2. **Regular grammar:** Type 3 grammar is also called regular grammar and it generates regular language. Regular grammar is divided into two types:

   Left linear grammar: $V \to VT^* + T^*$

   Right linear grammar: $V \to T^*V + T^*$

   A grammar is a regular grammar iff $\exists$ a left linear or right linear grammar for it.

5.3.7 Pumping Lemma

Pumping lemma is very useful to prove that a certain language is not a regular language. We know that the loop in finite automata (FA) makes it able to accept those strings which have length equal to and greater than its total number of states.

Let there be a string w which has a bigger length (more than its states), then we can break this string into three parts x, y (y should not be null) and z. Let FA has loop for y and $w = xyz \in L$ accepted by FA, so $w = xy^iz$ for $i = 0, 1, \ldots$ is also accepted by FA.

5.3.8 Myhill–Nerode Theorem

Myhill–Nerode theorem tells that the given language might be regular or not regular. A given language L is a regular language if and only if the set of equivalence classes of L is finite, or it satisfies following three conditions:

1. The language L divides the set of all possible strings into mutually separate classes.
2. If L is a regular language, then the number of classes created is finite.
3. If the number of classes that L creates is finite, then L is a regular language.

Problem 5.1: Consider a language L consists all strings over $\{a, b\}$ ending in b. Prove that L is a regular language.

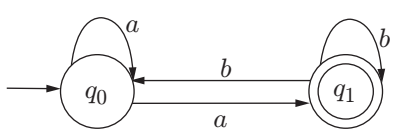
Solution: Let DFA $M = \{Q, S, \delta, q_0, F\}$ recognizes $L = \{b, bb, ab, aab, abb, \ldots\}$.

We have two classes C_0 and C_1 defined as follows:

$C_0 = \{$All strings that end in $a\}$

$C_1 = \{$All strings that end in b, i.e., accepted strings $(b, ab, abb, bbb, \ldots)\}$

Transition diagram for the DFA M is shown below.



Classes C_0 and C_1 correspond to states q_0 and q_1, respectively. Applying Myhill–Nerode theorem:

1. Two classes C_0 and C_1 are mutually exclusive.
2. L is regular and creates finite classes (here two classes).
3. The number of classes created by DFA is finite.

As all the three statements of Myhill–Nerode theorem are satisfied, hence L is regular.

5.3.9 Transducer or Finite State Machine

A finite state machine (FSM) is similar to FA except that it has the additional capability of producing an output.

$$\text{FSM} = \text{FA} + \text{Output capability}$$

There are two types of FSMs:

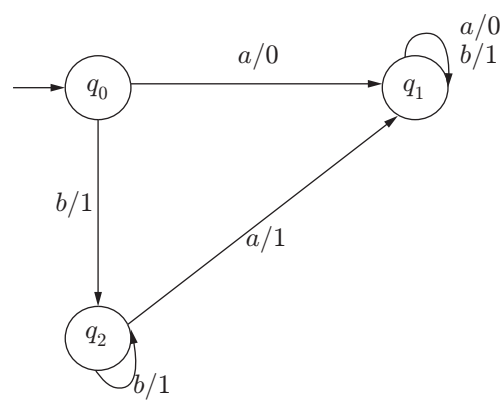
1. Mealy machines
2. Moore machines

5.3.9.1 Mealy Machines

In a Mealy machine, the output of an FSM is dependent on the present state and present input. A Mealy machine can be described by a six-tuple set $(Q, \Sigma, \Delta, \delta, \lambda, S)$, where

1. Q is a finite and non-empty set of states.
2. Σ is an input alphabet.
3. Δ is an output alphabet.
4. δ is a transition function which maps the present state and input symbol on to the next state $Q \times \Sigma \rightarrow Q$.
5. λ is the output function which maps $Q \times \Sigma \rightarrow \Delta$ ((present state + present symbol) $\rightarrow$ output).
6. $S \in Q$ is the starting state or initial state.

Problem 5.2: Consider the Mealy machine shown in the figure. Construct the transition table and find the output for input *aabba*.



Solution: Transition table for the above Mealy machine is:

Present State	Next State		Output	
	$x = a$	$x = b$	$x = a$	$x = b$
q_0	q_1	q_2	0	1
q_1	q_1	q_1	0	1
q_2	q_1	q_2	1	1

Input: *aabba*

Output: 00110

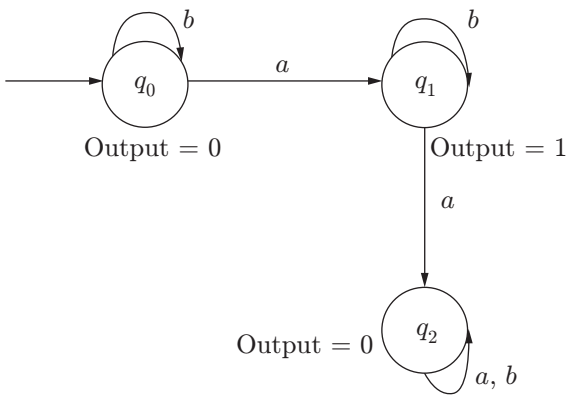
(**Note:** Output length of the Mealy machine is equal to the input length.)

5.3.9.2 Moore Machines

In Moore machine, the output of an FSM is dependent on the present state only. A Moore machine can be described by a six-tuple set $(Q, \Sigma, \Delta, \delta, \lambda, S)$, where

1. Q is a finite and non-empty set of states.
2. Σ is an input alphabet.
3. Δ is an output alphabet.
4. δ is a transition function which maps the present state and input symbol on to the next state $Q \times \Sigma \rightarrow Q$.
5. λ is the output function which maps $Q \rightarrow \Delta$ (present state $\rightarrow$ output)
6. $S \in Q$ is the starting state or initial state.

Problem 5.3: Consider the Moore machine shown in the figure. Construct the transition table and find the output for input *aabba*.



Solution: Transition table for the above Mealy machine:

Present State	Next State		Output
	$x = a$	$x = b$	
q_0	q_1	q_0	0
q_1	q_2	q_1	1
q_2	q_2	q_2	0

Input: *aabba*

Output: 010000

(**Note:** Output length of Moore machine is one greater than the input length because the first output symbol is additional without reading any symbol from the input.)

5.3.10 Conversion in Finite Automata

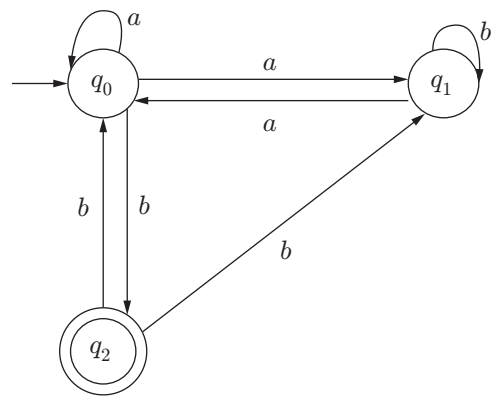
Power of NFA and DFA is the same, which means both can accept the same set of strings. Here we discuss interconversion of NFA and DFA.

5.3.10.1 NFA to DFA

To convert NFA to DFA, following points should be considered:

1. In NFA and DFA there is strictly one initial or starting state.
2. If there are n states in NFA, then equivalent DFA contains $\leq 2^n$ states.
3. DFA can simulate the behaviour of NFA by increasing the number of states.

Problem 5.4: Construct a DFA equivalent of given NFA.



Solution:

Step 1: Construct a transition table of the given NFA

δ	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	q_2
q_1	q_0	q_1
q_2	–	$\{q_0, q_1\}$

q_0 – Starting state

q_2 – Final state (accepting state)

Step 2: Construct a DFA transition table with the help of an NFA transition table. Put the first row same from the NFA transition table, then put one by one states from the right side. After the first row, put $\{q_0 q_1\}$ in the transition column and find the next state on a from q_0 and q_1 separately and write them under column a, repeat the same process for b. Now pick another state q_2 from the right side and follow the same process. This process has to be repeated until all the unique value set comes as a state under the transition table.

Now make the starting state of DFA the same state as in NFA. And make all those states final which

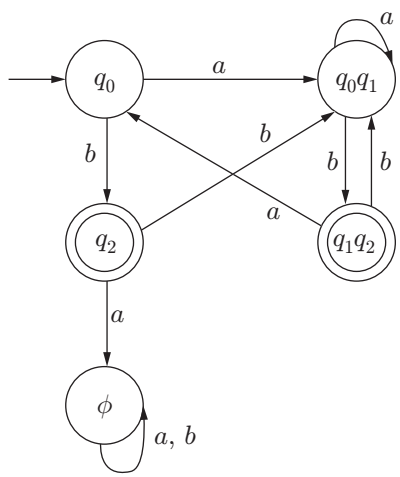
include the final state of NFA; in our example it was q_2, so q_1q_2 and q_2 states will be final states in DFA.

δ	a	b
$\rightarrow q_0$	$\{q_0q_1\}$	q_2
$\{q_0q_1\}$	$\{q_0q_1\}$	$\{q_1q_2\}$
q_2	ϕ	$\{q_0q_1\}$
$\{q_1q_2\}$	q_0	$\{q_0q_1\}$
ϕ	ϕ	ϕ

$\longrightarrow$ Represent starting state

◯ Represent final or accepting state

Step 3: Draw DFA from the transition table.

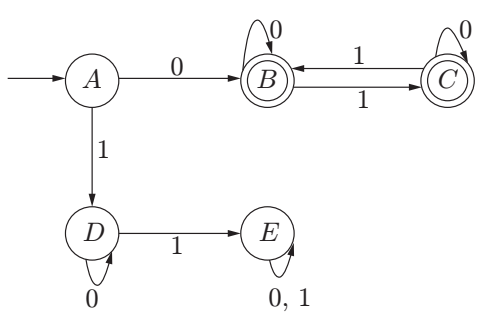


5.3.10.2 DFA to Minimum DFA

Minimization of DFA means reducing the unnecessary states. During minimization, the following steps are considered.

1. DFA with n states when converted into minimal DFA will contain N states, where $1 \leq N \leq n$.
2. We have to eliminate the same states from the given DFA to convert it into minimal DFA.
3. Two states $q_1 \equiv q_2$ iff $\forall\,(w \in \Sigma^*)$, $\{\delta(q_1, w), \delta(q_2, w)\}$ both are from the same set.
4. If a final state (accepting state) is equivalent to a non-final state, then it cannot be reduced.

Problem 5.5: Convert the given DFA into minimal DFA.



Solution:

Step 1: Make two sets of final and non-final states.

Iteration 1 = $(\{A, D, E\}\ \{B, C\})$

Step 2: Check within the set if two of them are equivalent states or not.

Let us first check in $\{A, D, E\}$, first we pick A and D.

Present State	Next State on Input	
	$X = 0$	$X = 1$
A	B	D
D	D	E
E	E	E

We can clearly see that state A is going outside on state B at input 0, which is in another set, whereas the other two states are indicating within the non-final state set. So, we will put state A in a new set.

Iteration 2 = $(\{A\}\ \{D, E\}\ \{B, C\})$

Repeat step 2 until last two iterations become same.

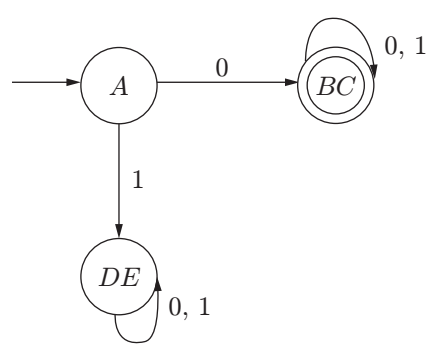
Now check for B and C.

Present State	Next State on Input	
	$X = 0$	$X = 1$
B	B	C
C	C	B

States B and C are not going outside from their own set at inputs 0 and 1, so there will be no change.

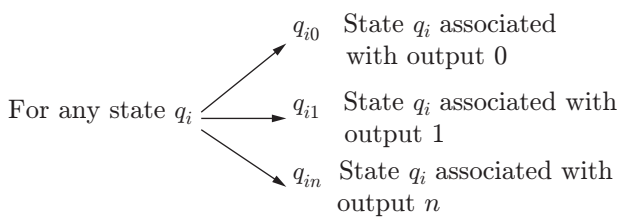
Iteration 3 = $(\{A\}\ \{D, E\}\ \{B, C\})$

We get two same iterations so we will stop this process. We will merge those states which lie in a set. There are three states in minimal DFA (A), (DE) and (BC). Now we can draw the minimal DFA.



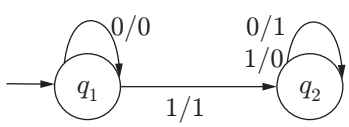
5.3.10.3 Mealy to Moore Machine

In a Mealy machine, the output depends upon both the present state and the input, and the Moore machine output depends only on the present state. While converting Mealy to Moore we develop a procedure so that all the states of Mealy machines, which are associated with different outputs, find them. After that, split those states into n parts if the states have n outputs.



Through this procedure all the states are associated with a single output, so all the states are considered as unique state. We can understand this concept clearly with an example.

Problem 5.6: Consider the 2's complement Mealy machine given below and convert it into Moore machine. At input "1101", the output is "0011". Input string will be read from the right side.



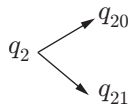
Solution: First we will make a transition table for the given Mealy machine.

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
q_1	q_1	q_2	0	1
q_2	q_2	q_2	1	0

In the transition table, we have to find those states which are associated with more than one output in the next state column.

For q_1 at $(q_1, 0)$, we have output 0.

For state q_2 at $(q_2, 0)$ output is 1 and at $(q_2, 1)$ output is 0. So there is a need to split q_2.

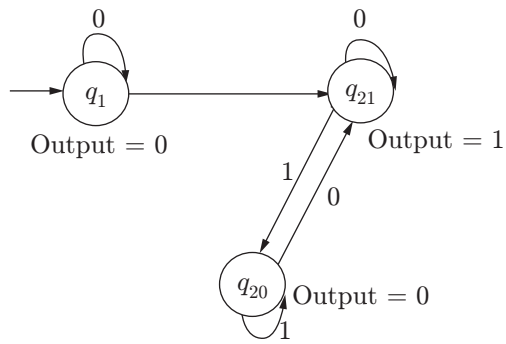


Now the Moore machine has three states $\{q_1, q_{20}, q_{21}\}$ with output $\{0, 0, 1\}$, respectively.

Now we will make a transition table of the Moore machine.

Present State	Next State		Output
	$x = 0$	$x = 1$	
$\rightarrow q_1$	q_1	q_{21}	0
q_{20}	q_{21}	q_{20}	0
q_{21}	q_{21}	q_{20}	1

Moore machine of 2's complement:



Note: m states, n outputs Mealy machine $\equiv$ Moore machine contains $\leq mn + 1$ state.

5.3.10.4 Moore to Mealy Machine

Consider the Moore machine given in Fig. 5.3 and convert it into Mealy machine.

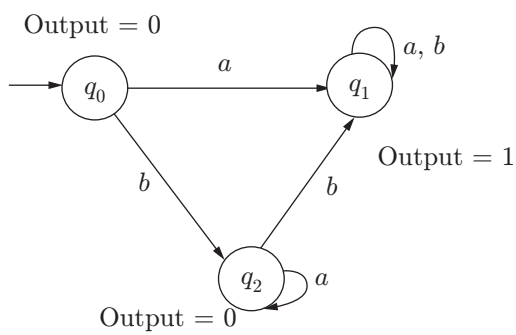


Figure 5.3 | Moore machine.

In the transition table of Moore machine (Table 5.4), the states (q_0, q_1, q_2) have outputs (0, 1, 0), respectively.

Table 5.4 | Transition table of Moore machine

Present State	Next State		Output
	$x = a$	$x = b$	
$\rightarrow q_0$	q_1	q_2	0
q_1	q_1	q_1	1
q_2	q_2	q_1	0

So, now we can make a transition table for the Mealy machine (Table 5.5).

Table 5.5 Transition table of Mealy machine

Present State	Next State		Output	
	$x = a$	$x = b$	$x = a$	$x = b$
$\rightarrow q_0$	q_1	q_2	1	0
q_1	q_1	q_1	1	1
q_2	q_2	q_1	0	1

Mealy machine equivalent to given Moore machine (Fig. 5.4):

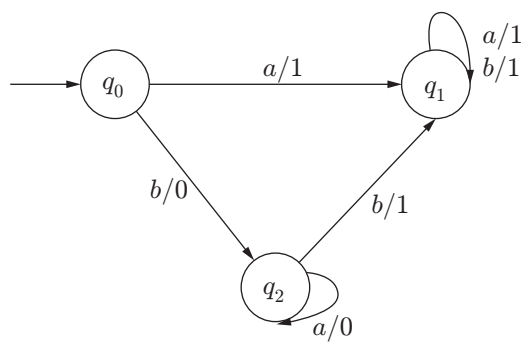


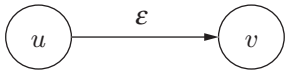
Figure 5.4 Mealy machine.

Note: m states, n outputs Moore machine $\equiv$ Mealy machine contains $\leq m$ states.

5.3.10.5 NFA with ε-Moves to NFA Without ε-Moves

There are three steps to convert NFA with ε moves to NFA without ε moves:

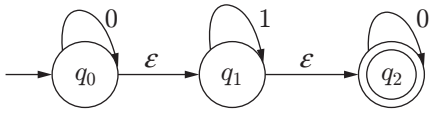
1. If there is an ε move from u to v, then every arrow starting from v is also starting from u.



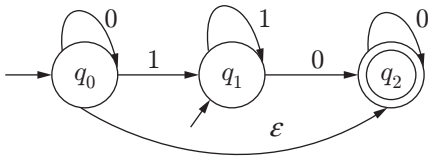
2. When ε move removes from u to v, and if u is a starting state, then make v also a starting state.
3. When ε move removes from u to v, and if v is a final state, then make u also a final state.

Problem 5.7: Convert given NFA with ε moves to NFA without ε moves.

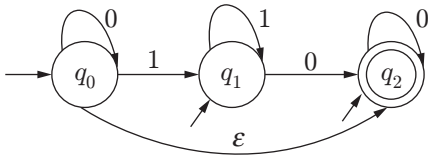
Solution:



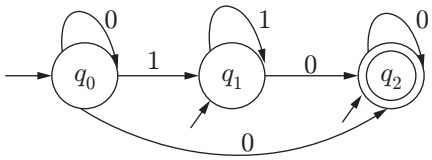
Step 1: Follow above steps to remove first ε move between q_0 and q_1.



Step 2: Follow above steps to remove ε move between q_1 and q_2.



Step 3: Follow above steps to remove ε move between q_0 and q_2.



After removing all ε moves, this is the final NFA.

Note: If NFA with ε-moves has n states then NFA without ε-moves will have exactly n states.

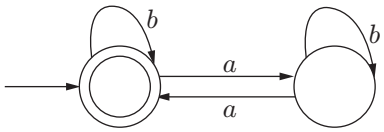
5.3.11 Properties of Regular Sets/Languages

1. Classes of two regular sets S_1 and S_2 are closed under union operation, means $S_1 \cup S_2$ is also a regular set.
2. Classes of two regular sets S_1 and S_2 are closed under concatenation operation, means S_1S_2 is also a regular set.
3. A class of regular set S_1 is closed under Kleene closure operation, means S_1^* is also a regular set.
4. If S is a regular set on some alphabet Σ, then complement of S is denoted by $\overline{S}$ is also a regular set. Steps to make complement of an NFA are as follows:
   - Change all final states to non-final states of NFA for Σ.
   - Change all non-final states to final states of NFA for Σ.
5. Classes of two regular sets S_1 and S_2 are closed under intersection operation, means $S_1 \cap S_2$ is also a regular set.
6. If S is a regular set on some alphabet Σ, then transpose of S is denoted by S^R is also a regular set.

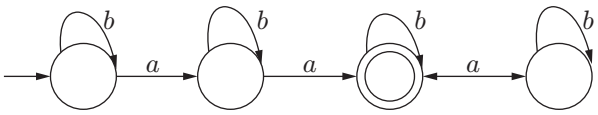
5.3.12 Some Important DFA

Some examples of important DFAs are given below:

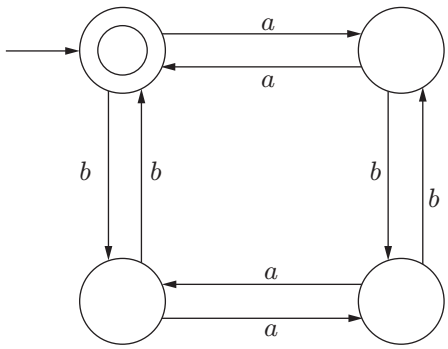
1. Containing even number of a's. This can be written as $L = \{w \mid na(w) \bmod 2 = 0\}$:



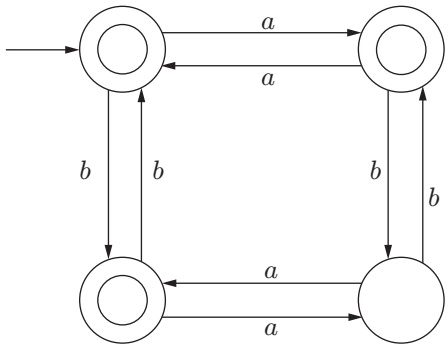
2. DFA having exactly two a's:



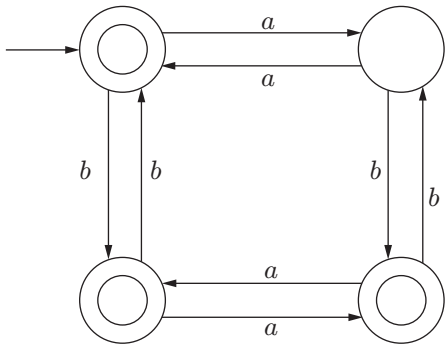
3. $L = \{w \mid na(w) \bmod 2 = 0 \text{ and } nb(w) \bmod 2 = 0\}$:



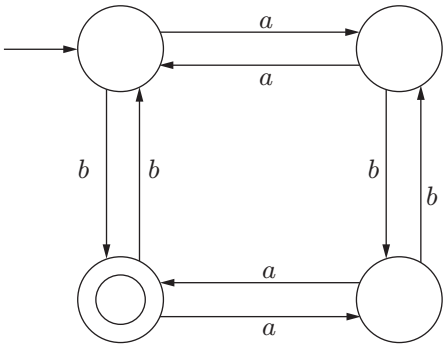
4. $L = \{w \mid na(w) \bmod 2 = 0 \text{ or } nb(w) \bmod 2 = 0\}$:



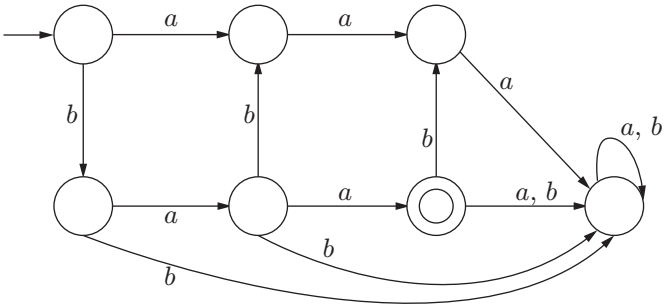
5. $L = \{w \mid na(w) \bmod 2 = 0 \text{ or } nb(w) \bmod 2 = 1\}$:



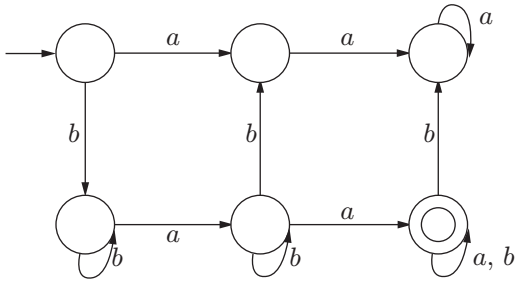
6. $L = \{w \mid na(w) \bmod 2 = 0 \text{ and } nb(w) \bmod 2 = 1\}$:



7. DFA which accepts exactly two a's and one b:



8. DFA which accepts at least two a's and at least one "b":



5.4 PUSHDOWN AUTOMATA

Pushdown automata (PDA) are devices that recognize context-free languages. PDA has finite memory in terms of a stack. A pushdown automaton is a non-deterministic device. The deterministic version of automata accepts only a subset of CFL known as deterministic context-free language (DCFL).

5.4.1 Model of PDA

PDA can be defined as a seven-tuple set $\{Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\}$, where

1. Q is a finite and non-empty set of states.
2. Σ is a finite and non-empty set of input symbol.

3. Γ is a finite non-empty set of stack alphabets.
4. δ is the transition function which maps $Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow Q \times \Gamma^*$
5. q_0 is the initial state (start state).
6. $Z_0 \in \Gamma$ is the starting (top most) stack symbol.
7. F is the set of final states and $F \subseteq Q$.

Figure 5.5 shows a model of PDA.

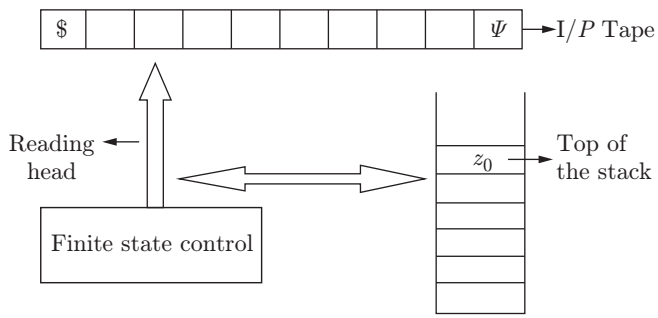


Figure 5.5 Model of PDA.

5.4.2 Transition Function

Three types of operations are performed under the transition function (δ) of PDA.

1. **Push element into stack:** Let the current stack symbol be z_0, current state be q_0 and current input symbol be "a". So after reading "a", z_2 is to be pushed into stack and next state may or may not be changed according to the transition function written as

   δ(current state, current input symbol, top of stack symbol) $\rightarrow$ (next state, new top of stack)

   $$\delta(q_0, a, z_0) = (q_1, az_0)$$

2. **Pop element from stack:** If "a" is input symbol on state q_0 with stack top z_1 and z_1 is popped up after processing "a" and next state is q_1. Then transition function δ is written as

   δ(current state, current input symbol, top of stack symbol) $\rightarrow$ (next state, top of stack)

   $$\delta(q_0, a, z_1) = (q_1, \lambda)$$

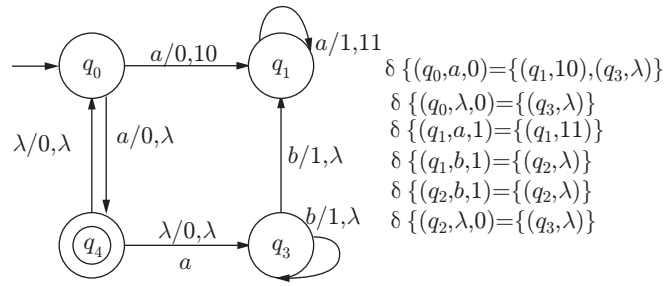
3. **No change in state element:** If on an input symbol "a" at state q_0 on stack top z_1 nothing is pushed or popped, then transition function can be written as

   $$\delta(q_0, a, z_1) = (q_1, z_1)$$

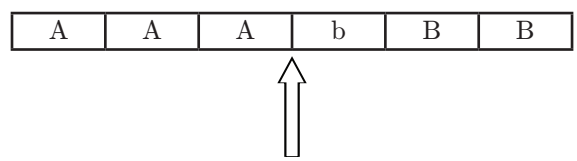
5.4.3 Non-deterministic PDA

Non-deterministic pushdown automata (NPDA) have more than one choice for a single input such as NFA. Transition function of NPDA is $Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow Q \times \Gamma^*$. An NPDA accepts an input string if a sequence leads to some final state of PDA or cause PDA to empty its stack. A non-deterministic automaton is equivalent to a CFG and more powerful than a deterministic PDA.

Example 5.16



This NPDA will recognize the string "*aaabbb*" by the following moves:



Input head start from left and starting stack symbol is 0.

1. $(q_0, a, 0) \vdash (q_1, 10)$
2. $(q_1, a, 10) \vdash (q_1, 110)$
3. $(q_1, a, 110) \vdash (q_1, 1110)$
4. $(q_1, b, 1110) \vdash (q_2, 110)$
5. $(q_2, b, 110) \vdash (q_2, 10)$
6. $(q_2, b, 10) \vdash (q_2, 0)$
7. $(q_2, \lambda, 0) \vdash (q_3, \lambda)$

As $q_3 \in F$, the string is accepted.

5.4.4 Deterministic PDA

A deterministic pushdown automata (DPDA) is one for which every input string has a unique way through the machine. A PDA $M = \{Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\}$ is deterministic if it satisfies the following two conditions:

1. $\delta(q, a, z)$ is either empty or only single move.
2. $\delta(q, \lambda, z) \neq \varnothing$ means $\delta(q, a, z) = \varnothing$ for each $a \in \Sigma$.

The language accepted by DPDA is known as deterministic context-free language (DCFL), and it is noticeable that not all CFLs are DCFLs.

Example 5.17

Language $L = \{a^n \times b^n : n \geq 0\}$, it can be accepted by the DPDA because all 'a' characters are inserted into stack till character 'x' come and then symbol 'a' is removed as many times as many symbol 'b' occur. In the last stack will be empty.

5.4.5 Parsing

Parsing is a technique to construct a parse or a derivation tree. It is to check whether there is some leftmost derivation for a given string using a certain grammar. Parse tree generated through parsing is also used by a syntax analyzer in a compiler for checking a string is according to a given grammar or not.

5.4.5.1 Top-Down Parsing

In leftmost derivation, we start with initial and replace the leftmost variable in each step and finally get the string. This approach is known as top-down parsing. In the parse, start symbol is root, terminals are leaves (input symbols) and other nodes are variables. We start from the root and replacing the intermediate nodes one by one from left to right reach the leaves. This approach is also known as recursive descent.

Here, we have constructed the leftmost derivation looking ahead one symbol in the given string. A grammar having this property is known as LL(1). In general, if a leftmost derivation is constructed by looking at the k symbol ahead in the given string, then the grammar is known as LL(k) grammar, where $k \geq 1$.

Steps of construction of top-down parser:

Step 1: Eliminate left recursion (if any) from the given grammar.

Step 2: Eliminate left factoring (if any) from the given grammar.

Step 3: If resulting grammar is LL(k) for some $k \geq 1$, then construct a parsing table.

5.4.5.2 Bottom-Up Parsing

In bottom-up parsing, we start with the input string and replace the terminals by respective variables such that these replacements lead to the starting symbol of the grammar. So every step takes input string close to the starting symbol. This approach is the reverse of top-down approach. It reads the input from left and uses the rightmost derivation in reverse order. Bottom-up parser is also known as shift-reduce (SR) parser. There are three actions in bottom-up parsing:

Step 1: Shift the current input (token) on the stack and read the next token.

Step 2: Reduce by some suitable LHS of production rule.

Step 3: Accept, final reduction which leads to starting symbol.

5.5 CONTEXT-FREE GRAMMAR AND LANGUAGES

A grammar $G = (V_n, T, S, P)$ is said to be a CFG if the production of G is of the form $A \to \alpha$, where $\alpha \in (V_n \cup S)^*$ and $A \in V_n$.

As we know that a CFG has no context either on left or on right, this is the reason why it is also known as context-free.

Problem 5.8: Consider a grammar $G = (V_n, T, S, P)$ having production $\{S \to aSa|bSb|x\}$. Check the production and find the language generated.

Solution: Let P_1: $S \to aSa$

P_2: $S \to bSb$

P_3: $S \to x$

(a, b, x) are terminals and S is a variable. As all the productions are of the form $A \to \alpha$, where $\alpha \in (V_n \cup S)^*$ and $A \in V_n$, hence G is a CFG, and it will produce context-free language.

Language generated: $L(G) = \{w\, x\, w^{\mathrm{R}}: w \in (a + b)^*\}$

5.5.1 Context-Free Grammar

A grammar $G = (V_n, T, S, P)$ is said to be a CFG if production $P\ \{u \to v\}$ follows these conditions:

1. $u \to v$, where $v \in (V \cup T)^*$
2. $|u| \leq |v|$ (length of u is less than v)
3. Only single variable is allowed in the left side (means u has single variable only)

Example 5.18

$$S \to aSb/\lambda$$
$$A \to aA/aB/Cb$$
$$B \to a/Da$$

5.5.2 Standard Context-Free Language

There are some standard languages under CFL such as:

1. $a^n b^n$, $a^n b^{2n}$, $a^{2n} b^n$, $a^n b^{2n+3}$, $a^{2n+2} b^{3n+5}$, etc.
2. $n_a(w) = n_b(w)$ (means number of "a" equals to number of "b" in a language), $a^m b^n$, where ($m = n$, $m > n$, $m < n$, etc.)
3. Palindrome such as [ww^{R} (even palindrome), waw^{R} (odd palindrome), etc.]

4. Generate language from grammar:
   - Grammar is $S \rightarrow aSbb|\varepsilon$
     Language generated by the above grammar is $a^n b^{2n}$.
   - Grammar is $S \rightarrow aSb|bSa|a|b$
     Language generated by the above grammar is $w\ x\ w^R$, $(x \in a,\ b)$.
5. Make grammar for given language:
   - Language $L = \{a^{2n}b^n\}$, where $n \geq 1$
     Grammar for given language: $S \rightarrow aaSb|\varepsilon$
   - Language $L = \{w\#w^R\colon w \in (a,\ b),\ \# \neq \Sigma\}$
     Grammar for a given language: $S \rightarrow aSb|bSa|\#$

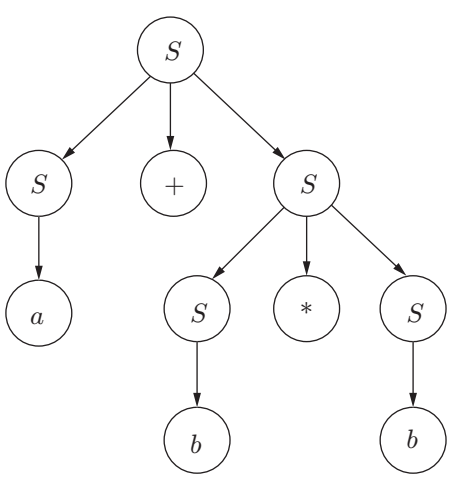
5.5.3 Derivation Tree or Parse Tree

The string generated by CFG $G = (V_n,\ T,\ S,\ P)$ is represented by a hierarchical structure called tree. A derivation tree or parse tree for a CFG is a tree that satisfies the following conditions:

1. If $A \rightarrow \alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n$ is a production in G, then A becomes the father of nodes labelled as $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n$.
2. The root has label S (starting symbol).
3. Every vertex (or node) has a label.
4. Internal nodes should be labelled with variables only.
5. The leaf nodes are labelled with ε or terminal symbols.
6. The collection of leaves from left to right yields the string w.

Problem 5.9: Consider the grammar $S \rightarrow S + S|S^*S|a|b$. Construct derivation (or parse) tree for the string $w = a + b^*b$.

Solution:



5.5.3.1 Leftmost Derivation Tree

A derivation tree is called as leftmost derivation (LMD) tree if the ordering of decomposed variable is from left to right. Thus, for generating a string $w = aab$ from grammar:

$S \rightarrow AB$ (production 1)

$A \rightarrow aaA$ (production 2)

$A \rightarrow \varepsilon$ (production 3)

$B \rightarrow bB$ (production 4)

$B \rightarrow \varepsilon$ (production 5)

LMD:

$S \rightarrow \underline{A}B$ by production 1)

$S \rightarrow aa\underline{A}B$ (by production 2)

$S \rightarrow aa\underline{B}$ (by production 3)

$S \rightarrow aab\underline{B}$ (by production 4)

$S \rightarrow aab$ (by production 5)

5.5.3.2 Rightmost Derivation Tree

A derivation tree is called rightmost derivation (RMD) tree if the ordering of decomposed variable is from right to left. Thus, for generating a string $w = aab$ from the above grammar:

RMD:

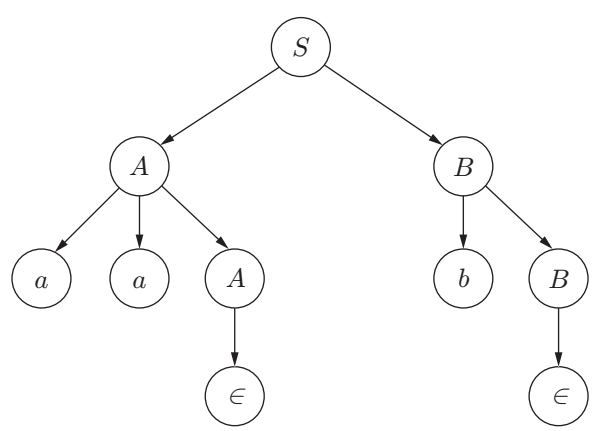
$S \rightarrow A\underline{B}$ (by production 1)

$S \rightarrow Ab\underline{B}$ (by production 4)

$S \rightarrow \underline{A}b$ (by production 5)

$S \rightarrow aa\underline{A}b$ (by production 2)

$S \rightarrow aab$ (by production 3)

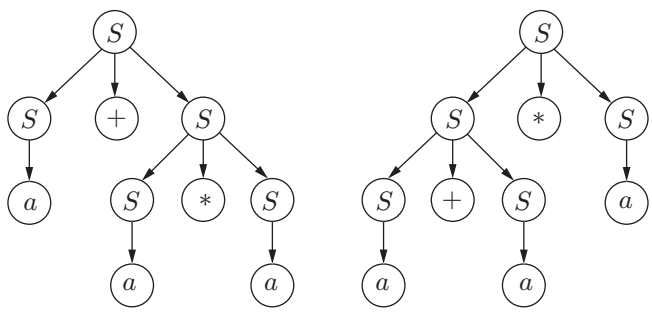


Left to right

5.5.4 Ambiguous Grammar

A grammar G is called ambiguous if for some string $w \in L(G)$, there exists two or more derivation tree (two or more LMD or two or more rightmost derivation tree). Let us consider a CFG grammar having production:

$S \to S + S|S^*S|a|b$, for string $w = a + a^*a$ have more than one LMD tree.



Note: A language (L) is called ambiguous language if and only if every grammar which generates it is ambiguous. The only known ambiguous language is $\{a^n b^m c^n\} \cup \{a^n b^m c^m\}$.

Left recursion and left factoring are the major cause for a grammar to be ambiguous. But presence of these in a grammar does not mean that grammar is ambiguous, and similarly absence of these does not mean that grammar is unambiguous.

5.5.5 Removal of Ambiguity

Ambiguities are left recursion and left factoring. In this subsection, removal of various ambiguities is discussed step by step.

5.5.5.1 Removal of Left Recursion

A production of grammar $G = (V_n, T, S, P)$ is said to be left recursive grammar if it has one of the production in a given form.

$A \to A\alpha$, where A is a variable and $\alpha \in (V_n \cup S)^*$.

Elimination of left recursion: Let the variable A have left recursive problem as follows:

$A \to A\alpha_1|A\alpha_2|A\alpha_3|\ldots|A\alpha_n|\beta_1|\beta_2|\beta_3|\ldots|\beta_m$, where β_1, β_2, β_3, …, β_m do not begin with A, then we replace A production by:

$$\{A \to \beta_1 A^{|}|\beta_2 A^{|}|\beta_3 A^{|} \ldots|\beta_n A^{|}|\},$$

where $A^{|} \to \alpha_1 A^{|}|\alpha_2 A^{|}|\alpha_3 A^{|}|\ldots|\alpha_n A^{|}|\varepsilon$

Problem 5.10: Let grammar $S \to S + S|S^*S|a|b|c$

Solution: After removing left factoring, the productions are replaced by

$$S^{|} \to + SS^{|}|^*SS^{|}|\varepsilon$$
$$S \to aS^{|}|bS^{|}|cS^{|}$$

5.5.5.2 Removal of Left Factoring

In grammar G, two or more production of variable A are said to have left factoring if the production are in the following form:

$$A \to \alpha\beta_1|\alpha\beta_2|\alpha\beta_3|\ldots|\alpha\beta_m$$

where $\{\beta_1|\beta_2|\beta_3|\ldots|\beta_m\} \in (V_n \cup S)^*$ and does not start with α. All these production have common left factor α.

Elimination of left factoring: Let variable A have (left factoring) production as follows:

$A \to \alpha\beta_1|\alpha\beta_2|\alpha\beta_3|\ldots|\alpha\beta_m|\gamma_1|\gamma_2|\gamma_3|\ldots|\gamma_m$, where γ_1, γ_2, γ_3, …, γ_m and β_1, β_2, β_3, …, β_m do not contain α as a prefix, then we replace this production by

$$A \to \alpha A^{|}|\gamma_1|\gamma_2|\gamma_3|\ldots|\gamma_m$$
$$A \to \beta_1|\beta_2|\beta_3|\ldots|\beta_m$$

Problem 5.11: Let grammar $A \to abc|abd|abe$ remove left factoring.

Solution: After removing left factoring, the productions are replaced by

$$A \to abA^{|}$$
$$A \to c|d|e$$

5.5.6 Context-Free Grammar Simplification

For simplification of context-free language, it is necessary to eliminate variable, terminal and production having the following properties:

1. Unit production ($A \to B$)
2. Null production ($A \to \lambda$)
3. Useless production, a variable which does not occur even in a single derivation

5.5.6.1 Removal of Unit Production

A production (P) in CFG is said to be a unit production if it is in the following form:

$A \to B$ where A, $B \in V$ (variable). There are three steps to remove a unit production.

Step 1: Find $\widehat{P} = P - (\text{unit production})$

Step 2: Find unit production and draw a unit production dependency graph.

Step 3: Add new rules to the grammar.

Problem 5.12: Remove the unit production from the given CFG grammar.

$$S \to Aa|B$$
$$B \to A|bb$$
$$A \to a|bc|B$$

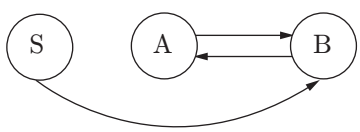
Solution:

Step 1: Find a non-unit production $(\widehat{P}) = P - (\text{unit production})$

$$S \to Aa$$
$$B \to bb$$
$$A \to a|bc$$

Step 2: Find a unit production and draw a unit production dependency graph.

$$S \to B$$
$$B \to A$$
$$A \to B$$



Step 3: Add new rules to the grammar. Check that from each variable through other variables on which terminals we can reach.

$$S \to a|bc|bb$$
$$A \to bb$$
$$B \to a|bc$$

Final grammar after removing a unit production is as follows:

$$S \to Aa|a|bc|bb$$
$$A \to a|bc|bb$$
$$B \to bb|a|bc$$

Note: Removal of a unit production has made B and the associated production useless.

5.5.6.2 Removal of Null Production

$A \to \lambda$ is a null production and A is a nullable variable; λ should be removed because it creates difficulties for the compilers. There are two steps to remove a null production.

Step 1: Identify the nullable variable.

Step 2: Find $\widehat{P}$ production without null.

Problem 5.13: Remove null (λ) production from the given CFG grammar.

$$S \to ABaC$$
$$A \to BC$$
$$B \to b|\lambda$$
$$C \to D|\lambda$$
$$D \to d$$

Solution:

Step 1: Identify nullable variables.

Variables B and C directly produce null $N = \{B, C\}$. But variable A is related to B and C, so A will also produce a null production.

$$N = \{A, B, C\}$$

Step 2: Find $\widehat{P}$ production without null.

$$S \to ABaC$$
$$A \to BC$$
$$B \to b$$
$$C \to D$$
$$D \to d$$

Generate grammar without null production ($G^{|}$) by putting null value to nullable variables one by one in production without null. For example, put $A = \lambda$ in $S \to ABaC$, we will get $S \to BaC$. Do this process for all nullable variables.

$$S \to ABaC|BaC|AaC|ABa|aC|aB|Aa|a$$
$$A \to BC|B|C$$
$$B \to b$$
$$C \to D$$
$$D \to d$$

5.5.6.3 Removal of Useless Production

Let $G = (V_n, T, S, P)$ be a CFG. A variable $A \in V$ is said to be useless if there is not even a single derivation which uses that production.

Example 5.19

$$S \to A$$
$$A \to aA|a$$
$$B \to bA$$

The variable B is useless and so is the production $B \to bA$. Although b can drive a terminal string, there is no way to reach on B from the starting variable S.

Problem 5.14: Consider a given grammar and eliminate useless production.

$$S \to aS|A|C$$
$$A \to a$$
$$B \to aa$$
$$C \to aCb$$

Solution:

Step 1: First find out those variables which lead to terminal strings. Because $A \to a$ and $B \to aa$, variables A and B directly come under this set. But due to the dependency on A, variable S also generates terminal string $S \to A \to a$. So,

$$S \to aS|A$$
$$A \to a$$
$$B \to aa$$

Step 2: Now we have to find those variables which cannot be reached from the starting variable. B is the only variable which cannot be reached from S. So B is a useless variable. Grammar after eliminating useless production and variables is as follows:

$$S \to aS|A$$
$$A \to a$$

5.5.7 Chomsky Normal Form

Chomsky normal form (CNF) has restriction on the length of the right-hand side of a production, either two variables or single terminal is allowed in the right side of a production such as $S \to AB$, $S \to a$.

Problem 5.15: Convert the given grammar into CNF form:

$$S \to aAD$$
$$A \to aB|bAB$$
$$B \to b$$
$$D \to d$$

Solution:

$S \to aAD$ or $\{S \to XD, X \to YA, Y \to a\}$

$A \to aB$ or $\{A \to ZB, Z \to a\}$

$A \to bAB$ or $\{A \to KB, K \to LA, L \to b\}$

This grammar can be written as in CNF $G^{|}$ as

$V_n = \{S, A, B, D, K, L, X, Y, Z\}$

$\Sigma = \{a, b, d\}$

$P^{|} = \{S \to XD, X \to YA, Y \to a, A \to ZB, Z \to a, A \to KB, K \to LA, L \to b\}$

5.5.8 Greibach Normal Form

A CFG is called in Greibach normal form (GNF) if all productions have the form $\{A \to ax$ or $A \to a\}$, where $a \in T$ and $x \in V^*$. Means RHS of production are either a terminal or a terminal followed by variables.

Problem 5.16: Convert the given grammar into GNF

$$S \to AB$$
$$A \to aA|bB|b$$
$$B \to b$$

Solutions: The GNF of the given grammar is

$$S \to aAB|bBB|bB$$
$$A \to aA|bB|b$$
$$B \to b$$

5.5.9 Identification of Language

1. If language is finite then it will surely be a regular language. Example: Let $L = \{a^m b^n, m + n = 10\}$ as m and n are finite, so it is a regular language. But it does not mean that if a language is infinite then it cannot be regular such as (a^*b^*) is an infinite and regular language.
2. If there is a comparison between m and n then it will be CFL. Example: Let $L = \{a^m b^n \ (m \geq n)\}$.
3. Let language $L = \{a^m b^n\}$, and if m or n is non-linear then it will not be accepted by PDA, so it will fall in the category of CSL.
4. If there are more than one comparison then it will be CSL. Example: Language $L = \{a^m b^n c^o, m \geq n$ and $o \geq n\}$.
5. All modular machines are regular language.
6. All palindrome languages are CFL language.

5.6 TURING MACHINE

In 1936, an English mathematician Alan Turing suggested the concept of Turing machine (TM). This machine can simulate the behaviour of a general purpose

computer. A TM is a generalization of a PDA, which uses a tape instead of tape and stack. The length of the tape is assumed to be infinite. A tape is divided into cells, and one cell can hold one input symbol. The head of the TM is capable to read and write on the tape and can move left or right or remain static.

TMs accept the languages defined by type 1 grammar, and these languages are called recursively enumerable or type 1 languages.

5.6.1 Model of a Turing Machine

A TM consists of a tape which is finite at the left end and infinite at the right end and has a finite control and read/write head (Fig. 5.6).

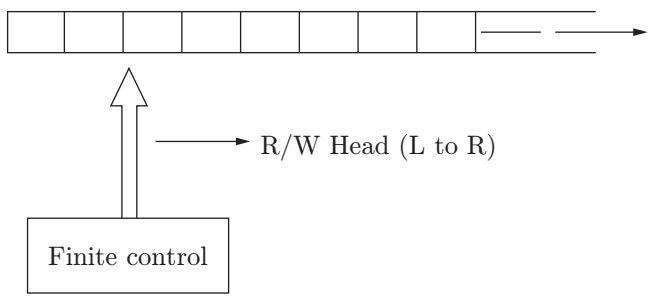


Figure 5.6 Model of a turing machine.

5.6.1.1 Mathematical Description of a Turing Machine

A TM can be described by a seven-tuple set (Q, Σ, Γ, δ, q_0, #, F), where

1. Q is the finite set of states (not including the halt state (h)).
2. Σ is the input alphabet which is a subset of the tape alphabet (not including the blank symbol #).
3. Γ is the finite set of symbols called the tape alphabet.
4. δ is the transition function which maps from $Q \times \Gamma \rightarrow Q \times \Gamma \times [L, R]$.
5. $\# \in \Gamma$ is a special symbol called blank.
6. $q_0 \in Q$ is the initial state.
7. $F \subseteq Q$ is the set of final states.

A TM can be deterministic or non-deterministic depending on the number of moves in a transition. If a TM has at the most one move in a transition, then it is a deterministic Turing machine (DTM). If there is more than one move, then it is a non-deterministic Turing machine (NTM). A standard TM reads one cell of the tape and changes its state (optional) and moves left or right by one cell.

The transition function (δ) is defined as $\delta(p, a) = (q, b, D)$, where p is the present state, q is the next state $a, b \in \Gamma$ and D is the movement {(left (L) or right (R)}.

After reading an input $a \in \Sigma$, TM does one of the following:

1. Replaces a by b and moves right (R) as $\delta(p, a) = (q, b, R)$.
2. Without write, anything moves right (R) as $\delta(p, a) = (q, a, R)$.
3. Replace a by b and move left (L) as $\delta(p, a) = (q, b, L)$.
4. Without write, anything moves left (L) as $\delta(p, a) = (q, a, L)$.

The change of state in the above transition is optional (means may or may not change).

5.6.1.2 Language Reorganization by a Turing Machine

A TM can be used as a language recognizer. It recognizes all languages (regular, CFL, CSL, type 0).

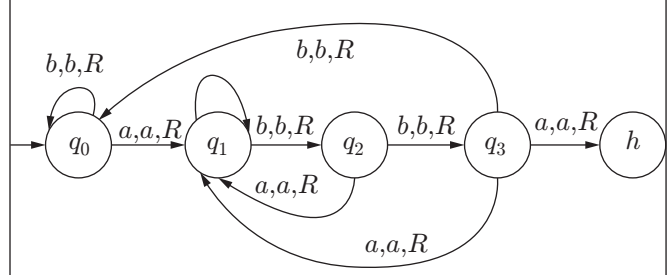
Let w be a string to be written on tap. The Turing machine is started from initial state q_0, read write head is placed on the left most symbol of w. with the sequence of moves, if TM enters to a final state and halts then string w is said to be accepted by Turing machine.

Problem 5.17: Design a TM for $L = \{w\colon w \in (a + b)^*$ ending in substring $abb\}$

Solution: Let TM $M = \{(q_0, q_1, q_2, q_3, h), (a, b), (a, b, \#), \delta, q_0, \#, h\}$ accepts language L, where δ is defined as follows:

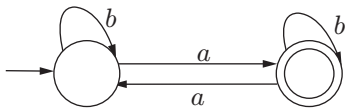
$\delta(q_0, b) = (q_0, b, R)$
$\delta(q_0, a) = (q_1, a, R)$
$\delta(q_1, a) = (q_1, a, R)$
$\delta(q_1, b) = (q_2, b, R)$
$\delta(q_2, a) = (q_1, a, R)$
$\delta(q_2, b) = (q_3, b, R)$
$\delta(q_3, a) = (q_1, a, R)$
$\delta(q_3, b) = (q_0, b, R)$
$\delta(q_3, \#) = (h, \#, S)$

The transition diagram is shown in the following figure.

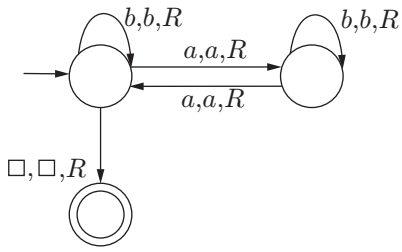


5.6.2 Designing a Turing Machine

Let us consider a problem for which we will design a TM. Suppose we have to design a TM which will halt on a final state if a string has an even number of "a"s'. So, first we will design a DFA which accepts a string having an even number of "a"s', and after that we will design a TM for the same problem.



This regular machine will accept only those strings which have an even number of a's. Now we will make a final state as non-final and add one final state on blank input.

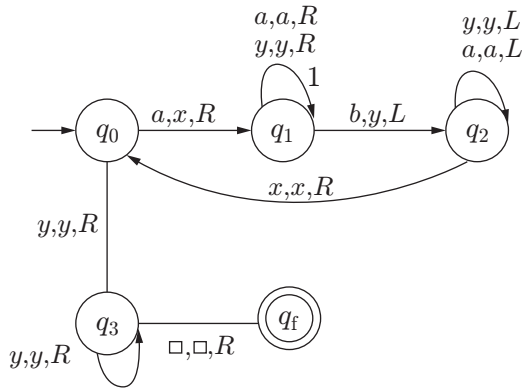


Problem 5.18: Design a TM which accepts a language $L = \{a^n b^n,\ n \geq 1\}$.

Solution: First make a transition function move based on an input in the tape which is shown in the figure.

$\delta(q_0,\ a) = (q_1,\ x,\ R)$
$\delta(q_1,\ a) = (q_1,\ a,\ R)$
$\delta(q_1,\ y) = (q_1,\ y,\ R)$
$\delta(q_1,\ b) = (q_2,\ y,\ L)$
$\delta(q_2,\ y) = (q_2,\ y,\ L)$
$\delta(q_2,\ a) = (q_2,\ a,\ L)$
$\delta(q_2,\ x) = (q_0,\ x,\ R)$
$\delta(q_0,\ y) = (q_3,\ y,\ R)$
$\delta(q_3,\ y) = (q_3,\ y,\ R)$
$\delta(q_3,\ \#) = (q_f,\ \#,\ R)$

A TM for a given language $L = \{a^n b^n,\ n \geq 1\}$.



5.6.3 Recursive and Recursive Enumerable Languages

The properties of recursive and recursive enumerable languages are as follows:

1. A language L is recursive enumerable (RE) if and only if there exists a TM that accepts it, machine must halt and accept $\forall\ w \in L$.
2. A language L is said to be recursive (REC) if and only if there exists a TM that accepts it and which halts on $\forall\ w \in \Sigma^*$.
3. REC is a decidable language.
4. RE is a semi-decidable language.
5. Both RE and REC have enumerable procedure (EP). A procedure is called an EP iff it lists all members of L in a finite amount of time.
6. Only REC has membership algorithm.
7. A set S is countable if it has an EP.
8. If A and B are countable then $(A \times B)$ is also countable.
9. Real numbers are uncountable infinite.
10. If a language L is RE then L has an EP but $\bar{L}$ does not have an EP.
11. If a language L is REC then both L and $\bar{L}$ have an EP. Every subset of countable set is countable, so every language L is countable because $L \subseteq \Sigma^*$ which is countable. It means every language has an EP but not those languages which are "not RE" or outside RE.
12. Set of all TM is countable.
13. If L is REC then $\bar{L}$ also will be REC.
14. If L is RE then $\bar{L}$ may or may not be RE.
15. If $\bar{L}$ is RE then L may or may not be RE.
16. Languages (regular, CFL, CSL, REC, RE) are countable infinite.
17. If both L and $\bar{L}$ is Turing enumerable then they are REC.

5.6.4 Variation of Turing Machine

Different variants of Turing machines are available. These variations are given as follows:

1. Multi-track Turing machine
2. Multi-tape Turing machine
3. Non-deterministic Turing machine
4. Universal Turing machine
5. Offline Turing machine
6. Turing machine with semi-infinite machine

5.7 CLOSURE AND DECIDABILITY

There are different closure and decidability properties for various languages. Tables 5.6 and 5.7 describe these properties for different languages.

Table 5.6 | Closure property of languages

Operation	Languages					
	Regular	DCFL	CFL	CSL	REC	RE
$\cup$	✓	×	✓	✓	✓	✓
$\cap$	✓	×	×	✓	✓	✓
L^c	✓	✓	×	×	✓	×
-	✓	×	✓	✓	✓	✓
*	✓	×	✓	✓	✓	✓
$L_1 \cup R$	✓	×	✓	✓	✓	✓
$L_1 \cap R$	✓	✓	✓	✓	✓	✓
L^R	✓	×	✓	–	–	–
$h(L)$	✓	×	✓	–	–	–
$h^{-1}(L)$	✓	✓	✓	–	–	–
L_1/L_2	✓	✓	–	–	–	–

✓{Surely exists}

× {May or may not exist}

– {Not known}

Table 5.6 is interpreted as follows:

1. $\{\cup\}$ means that if there are two languages L_1 and L_2 which are regular then their union will be regular, whereas if languages are DCFL then their union will not be DCFL.
2. $\{L_1 \cup R\}$ means that from two languages if one is regular (R) and the other is different, then their union will be result in a language similar to the second language. For example, if one language is regular and the other is DCFL, then their union will be a DCFL.
3. L^R represents reverse of a language; if there are two CFLs L_1 and L_2 then their union will be a CFL and their intersection will not be a CFL.
4. (.) shows concatenation operation.
5. (*) denotes multiplication operation.

Table 5.7 | Decidability table for languages

Algorithm Exists	Languages				
	Regular	CFL	CSL	REC	RE
Membership: $w \in \Sigma^*$, L then $w \in L$?	✓	✓	✓	✓	✓
Finiteness: Given language L is finite or infinite?	✓	✓	×	×	×
Emptiness: Language L is empty or not?	✓	✓	×	×	×
Equivalence: Two languages L_1 and L_2 are equal or not?	✓	×	×	×	×
Ambiguity: Language L is ambiguous or not?	✓	×	×	×	×
Regularity: Language is regular or not?	✓	×	×	×	×
Disjointness: Two language intersection is empty or not?	✓	×	×	×	×
Everything: Language L, then $L = \Sigma^*$	✓	×	×	×	×

✓ {Denotes that algorithm exists to decide about the question (decidable)}

× {Denotes that algorithm does not exist about that question (means undecidable)}

IMPORTANT FORMULAS

1. Number of states in a machine which accept m a' s and n b' s:
$$[(m + 1)^*(n + 1) + 1]$$
2. Mod n machines have n states.
3. Mod machine have no trap states.
4. If a machine accepts string length exactly n, then it has $(n + 2)$ states.
5. If a machine accepts string length $\leq n$, then it has $(n + 2)$ states.
6. If a machine accepts string length $\geq n$, then it has $(n + 1)$ states.
7. m States, n outputs Mealy machine $\equiv$ Moore machine containing $\leq mn + 1$ states.
8. m States, n outputs Moore machine $\equiv$ Mealy machine containing $\leq m$ states.
9. Output length of Mealy machine is equal to input length.
10. Output length of Moore machine is one greater than the input length because first output symbol is additional without reading any symbol from the input.
11. If language is finite then it will surely be regular. Example: Let $L = \{a^m b^n,\ m + n = 10\}$ as m and n are finite so it is a regular language. But it does not mean that if a language is infinite then it cannot be regular such as (a^*b^*) is an infinite and regular language.
12. If there is a comparison between m and n then it will be a CFL. Example: Let $L = \{a^m b^n,\ (m \geq n)\}$.
13. Let language $L = \{a^m b^n\}$ and if m or n is non-linear then it will not be accepted by PDA, so it will fall in the category of CSL.
14. If there are more than one comparison then it will be a CSL. Example: Language $L = \{a^m b^n c^o,\ m \geq n$ and $o \geq n\}$.
15. All modular machines are regular language.
16. All palindrome languages are CFL language.

Variation of NFA/DFA

17. DFA/NFA + (left $\leftrightarrow$ right) move $\equiv$ (2 − DFA)/(2 − NFA) means whatever a 2 − DFA can do, that can be done by simple DFA with left – right move.
18. DFA/NFA + (left $\leftrightarrow$ right) move + (read/write) head $\equiv$ (2 − DFA)/(2 − NFA).
19. DFA + 1 stack $\equiv$ DPDA).
20. NFA + 1 stack $\equiv$ NPDA).
21. DFA + 2 stack $\equiv$ TM.
22. DFA/NFA + 2 counter $\equiv$ TM.
23. DFA/NFA + 2 stack $\equiv$ DFA/NFA + 2 counter.
24. Power of {(DFA/NFA) < (DFA/NFA + 1 counter) < (DFA/NFA + 1 stack)}
25. {(DFA/NFA + 1 counter) < (DFA/NFA + 2 counter)}.

SOLVED EXAMPLES

1. Which of the following is a regular expression?

   (a) $L = \{0^m 1^n \mid m, n \geq 0\}$
   (b) $L = \{0^n 1^n \mid n \geq 0\}$
   (c) $L = \{xx \mid x \in \{0, 1\}^*\}$
   (d) $L = \{1^{n^2} 0^n \mid n \geq 0\}$

   Solution: In option (a), the given expression will generate any number of 0's followed by any number of 1's, which can be accepted by FA. So, it is a regular expression.

   In option (b), the given expression will generate number of 0's followed by number of 1's (both should be same in number), which will be accepted by PDA. So, it is not a regular expression.

   In option (c), the given expression will generate any string "x" twice, which will be accepted by PDA. So it is not a regular expression.

   In option (d), the given expression will generate twice the n number of 1's followed by n number of 0's, which cannot be accepted by FA. So, it is not a regular expression.

   Ans. (a)

2. The class of context-free language is not closed under

   (a) union
   (b) star
   (c) repeated concatenation
   (d) intersection

Solution: Context-free language is not close under intersection. Please refer table 5.6 (closure property of languages) in text.

Ans. (d)

3. Which of the following is true for regular sets $A = (1101 + 1)^*$ and $((1101)^*1^*)^*$?

(a) $A \subset B$ (b) $B \subset A$
(c) $A = B$ (d) A and B are incomparable

Solution: Both the regular expressions will generate the same set of strings, so both the regular sets are equal.

Ans. (c)

4. The language generated by the following grammar is $S \rightarrow aSa|bSb|\varepsilon$

(a) $a^m b^n$ $n \geq 0$, $m \geq 0$
(b) $a^n b^m$ $n \geq 1$, $m \geq 1$
(c) Odd length palindrome
(d) Even length palindrome

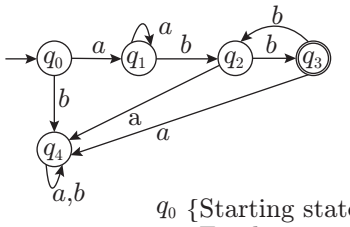
Solution: In the given grammar, variable S will generate a pair of symbol 'a' or symbol 'b' with variable S in the middle of them, and this is in loop so it can be generated any number of times. In the last variable, S will be replaced by ε. Some of the strings which can be generated by the given grammar are $\{aa, bb, abba, aabbaa, baab, \text{etc}\}$. So, the given grammar will generate even length palindrome.

Ans. (d)

5. Number of states in DFA accepting the following language is $L = \{a^n b^{2m} | n, m \geq 1\}$

(a) 2 (b) n
(c) m (d) 5

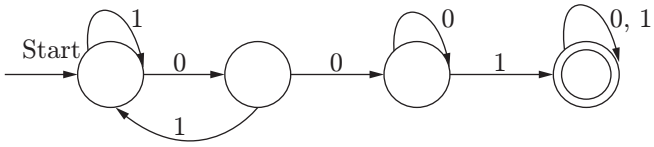
Solution: DFA for given language $L = a^n b^{2m} | n, m \geq 1$ is



q_0 {Starting state}
q_3 {Final state}

Ans. (d)

6. Consider the following DFS automaton M.



Let S denote the seven bits binary strings in which the first, fourth and last bits are 1. The number of strings in S that are accepted by M is

(a) 6 (b) 5
(c) 4 (d) 7

Solution: Strings which will be accepted by machine M can be found by fixing 1 at places first, fourth and last. We will have following four strings:

1001001 1001011 1001111 1111001

Ans. (c)

7. Which of the following CFG will generate the language $L = \{a^m b^n c^p d^q \mid m + n = p + q\}$

(a) $S \rightarrow aSd|A|B$
$A \rightarrow aAc|C$
$B \rightarrow bBd|D$
$C \rightarrow bCc|\varepsilon$

(b) $S \rightarrow aSd|A|B$
$A \rightarrow aAc|D$
$B \rightarrow bBd|C$
$C \rightarrow CBc|\varepsilon$

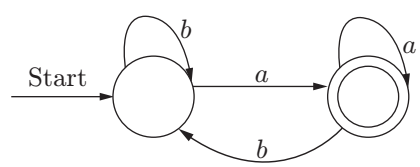
(c) $S \rightarrow aSd|A|B$
$A \rightarrow aAB|C$
$B \rightarrow bBd|C$
$C \rightarrow bCc|\varepsilon$

(d) $S \rightarrow aSd|A|B$
$A \rightarrow aAc|C$
$B \rightarrow bBd|C$
$C \rightarrow bCc$

Solution: Strings generated by L should have sum of number of 'a' and 'b' equal to total number of 'c' and 'd'. Only in option (c), the grammar can generate such string which has number of $(a + b)$ equal to number of $(c + d)$.

Ans. (c)

8. Consider the FA shown in the figure below, which language is accepted by the FA



(a) $b + (a + b)^*b$ (b) $(b + a + b)^*a$
(c) $b + a^*b$ (d) $b + a^*b^*$

Solution: Given that FA will accept any string which has the last symbol 'a'. So, only option (b) expression $(b + a + b)^*$ a can generate all combinations of strings which end with symbol 'a'.

Ans. (b)

9. Let $L = L_1 \cap L_2$, where L_1 and L_2 are languages defined below

$L_1 = \{a^m b^m c a^n b^m \mid m, n \geq 0\}$ and $L_2 = \{a^i b^j c^k \mid i, j, k \geq 0\}$. Then L is

(a) regular but not recursive
(b) context-free
(c) context-free but not regular
(d) recursively enumerable but not context-free

Solution: Language L_1 is CFL and language L_2 is regular. So, the result of $L_1 \cap L_2$ will be context free.

Ans. (c)

10. Which of the following is/are correct?

I. A language is context-free if and only if it is accepted by the PDA.
II. PDA is a finite automata with push-down stack.

(a) Only I is true
(b) Only II is true
(c) Both I and II are true
(d) Both I and II are false

Solution: Both I and II are true.

Ans. (c)

11. How many strings of length less than 4 contains the language described by the regular expression $(x + y)^*y(a + ab)^*$?

(a) 3 (b) 9
(c) 10 (d) 11

Solution: From the regular expression $(x + y)^*$ $y(a + ab)^*$, it can be seen that it contains 11 strings as follows:
$\{xy, yy, y, ya, yab, xya, yya, yaa, xxy, yyy, xyy\}$

Ans. (d)

12. Consider the following laws:

L_1: $(L^*)^* = L^*$
L_2: $\varnothing^* = \varnothing$
L_3: $\varepsilon^* = \varepsilon$
L_4: $L^+ = L^* + \varepsilon$
L_5: $L^* = \varepsilon + L^+$

Which of the above laws hold for regular expression?

(a) L_1, L_2 and L_5 (b) L_1, L_3 and L_5
(c) L_2, L_3 and L_4 (d) L_1, L_4 and L_5

Solution:

L_1: both the expression generates any string made up of input symbol of L. it also include ε.

L_2: $\varnothing^*$ can generate $\{\varnothing$ and $\varepsilon\}$

L_3: ε^* can only generate ε.

L_4: L^+ contain all the string made up of input symbol of L excluding ε.

L_5: L^* contain all the string made up of input symbol of L including ε.

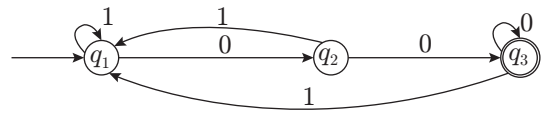
Son only L_1, L_3 and L_5 are applicable.

Ans. (b)

13. How many states does the DFA constructed for the set of all strings ending with "00" have?

(a) 1 (b) 3
(c) 2 (d) 2^2

Solution: DFA will construct as:



Ans. (b)

14. Which of the following CFG cannot be simulated by an FSM?

(a) $S \to Sa|b$
(b) $S \to aSb|ab$
(c) $S \to abX$, $X \to cY$, $Y \to d|aX$
(d) None of these

Solution: Given Grammar $S \to aSb|ab$ generates language $\{a^nb^n\}$ means language contains equal number of a's and b's. So, it cannot be simulated by finite state machine.

Ans. (b)

15. Let $r = 1(1 + 0)^*$, $s = 11^*0$ and $t = 1^*0$ be three regular expressions. Which one of the following is true?

(a) $L(s) \subseteq L(r)$ and $L(s) \subseteq L(t)$
(b) $L(r) \subseteq L(s)$ and $L(s) \subseteq L(t)$
(c) $L(s) \subseteq L(t)$ and $L(s) \subseteq L(r)$
(d) $L(t) \subseteq L(s)$ and $L(s) \subseteq L(r)$

Solution:

$r = 1(1 + 0)^*$, the language corresponds to r having all strings starting with 1.
$s = 11^*0$, the language corresponds to s having all strings starting with 1 followed by any number of 1 and ending with 0.
$t = 1^*0$, the language corresponds to t having all strings ending with 0.
So, $L(s) \subseteq L(r)$ and $L(s) \subseteq L(t)$. Therefore, option (a) is correct.

Ans. (a)

16. Which two of the following four regular expressions are equivalent?

(i) $(00)^*(\varepsilon + 0)$ (ii) $(00)^*$
(iii) 0^* (iv) $0(00)^*$

(a) (i) and (ii) (b) (ii) and (iii)
(c) (i) and (iii) (d) (iii) and (iv)

Solution:

(i) $(00)^*(\varepsilon + 0)$ represents any number of 0's
(ii) $(00)^*$ represents even no. of 0's
(iii) 0^* represents any number of 0's
(iv) $0(00)^*$ represents odd number of 0's

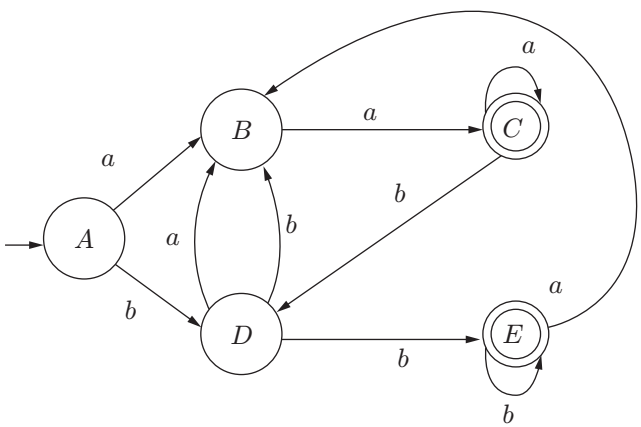
So, (i) and (iii) are the same expressions.

Ans. (c)

17. Let L be the set of all binary strings whose last two symbols are the same. The number of states in the minimum state deterministic finite state automation accepting L is

(a) 2 (b) 5
(c) 8 (d) 3

Solution: Minimum number of states in DFA which accept all binary strings whose last two symbols are the same is 5.



Ans. (b)

18. Consider the languages L_1, L_2, L_3 as given below:

$$L_1 = \{1^p 0^q | p,\ q \in N\}$$
$$L_2 = \{1^p 0^q | p,\ q \in N \text{ and } p = q\}$$
$$L_3 = \{1^p 0^q | p,\ q,\ r \in N \text{ and } p = q = r\}$$

Which of the following statements is NOT TRUE?

(a) Pushdown Automata (PDA) can be used to recognize L_1 and L_2.
(b) L_1 is a regular language.
(c) All the three languages are context-free.
(d) Turing machine can be used to recognize all the three languages.

Solution: Language L_1 is regular because it can be accept by NFA. Language L_2 and L_3 are CFL because they have comparison in between p and q.

Ans. (c)

19. Which of the following language over $\{a,\ b,\ c\}$ is accepted by a deterministic pushdown automata?

(a) $wcw^R \mid w \in \{a,\ b\}^*\}$
(b) $ww^R \mid w \in \{a,\ b,\ c\}^*\}$
(c) $\{a^n b^n c^n \mid n \geq 0\}$
(d) $\{w \mid w$ is a palindrome over $\{a,\ b,\ c\}$

Solution: The language $L = wcw^R \mid w \in \{a,\ b\}^*\}$ is accepted by DPDA because it is the only one which is DCFL. For the above language everything is pushed into the stack until "c" appears. When "c" appears all elements are removed from the stack if they match every symbol in w^R.

Ans. (a)

20. Which one of the following is the strongest correct statement about a finite language over some finite alphabet $\sum$?

(a) It could be undecidable.
(b) It is Turing machine recognizable.
(c) It is a regular language.
(d) It is a context-sensitive language.

Solution: Every finite language is regular and also CFL, CSL, RE (Turing machine recognizable) because regular language $\subset$ CFL $\subset$ CSL $\subset$ RE. So, the strongest correct statement is option (c).

Ans. (c)

GATE PREVIOUS YEARS' QUESTIONS

1. Ram and Shyam have been asked to show that a certain problem Π is NP-complete. Ram shows a polynomial time reduction from the 3-SAT problem to Π, and Shyam shows a polynomial time reduction from Π to 3-SAT. Which of the following can be inferred from these reductions?

(a) Π is NP-hard but not NP-complete.
(b) Π is in NP, but is not NP-complete.
(c) Π is NP-complete.
(d) Π is neither NP-hard nor in NP.

(GATE 2003: 1 Mark)

Solution: Reduction from the 3-SAT problem to Π is NP hard and reduction from Π to 3-SAT shows that Π is in NP.

As Π is in NP and it is NP hard, so it can be inferred that Π is NP complete.

Ans. (c)

2. If strings of a language L can be effectively enumerated in lexicographic (i.e., alphabetic) order, which of the following statements is true?

(a) L is necessarily finite.
(b) L is regular but not necessarily finite.

(c) L is context-free but not necessarily regular.
(d) L is recursive but not necessarily context-free.

(GATE 2003: 1 Mark)

Solution: L is recursive because recursive languages can be effectively enumerated in lexicographic order.

Ans. (d)

3. Consider the set $\sum^*$ of all strings over the alphabet $\sum = \{0, 1\}$. $\sum^*$ with the concatenation operator for strings

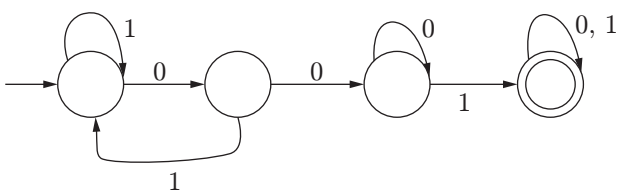
(a) does not form a group
(b) forms a non-commutative group
(c) does not have a right identity element
(d) forms a group if the empty string is removed from $\sum^*$

(GATE 2003: 1 Mark)

Solution: Fact

Ans. (a)

4. Consider the following deterministic finite state automaton M.



Let S denote the set of seven bit binary strings in which the first, the fourth, and the last bits are 1. The number of strings in S that are accepted by M is

(a) 1 (b) 5
(c) 7 (d) 8

(GATE 2003, 2 Marks)

Solution: Strings that are accepted are: 1001001, 1001011, 1001101, 1001111, 1111001, 1101001, and 1011001.

Ans. (c)

5. Let $G = (\{S\}, (a, b\}R, S)$ be a context-free grammar where the rule set R is

$$S \rightarrow a\,S\,b \mid S\ S \mid \varepsilon$$

Which of the following statements is true?

(a) G is not ambiguous.
(b) There exists $x, y, \in L(G)$ such that $xy \notin L(G)$.
(c) There is a deterministic pushdown automaton that accepts $L(G)$.
(d) We can find a deterministic finite state automaton that accepts $L(G)$.

(GATE 2003: 2 Marks)

Solution: Option (a) is False. G is ambiguous due to the possibility of two parsing trees for the same string.

Option (b) is True. Let $x = ab$ and $y = ba$, so $abba$ does not belong to $L(G)$.

Ans. (b)

6. A single tape Turing machine M has two states q_0 and q_1, of which q_0 is the starting state. The tape alphabet of M is $\{0, 1, B\}$ and its input alphabet is $\{0, 1\}$. The symbol B is the blank symbol used to indicate end of an input string. The transition function of M is described in the following table

	0	1	B
q_0	q_1, 1, R	q_1, 1, R	Halt
q_1	q_1, 1, R	q_0, 1, L	q_0, B, L

The table is interpreted as illustrated below.

The entry (q_1, 1, R) in row q_0 and column 1 signifies that if M is in state q_0 and reads 1 on the current tape square, then it writes 1 on the same tape square, moves its tape head one position to the right and transitions to state q_1.

Which of the following statements is true about M?

(a) M does not halt on any string in $(0 + 1)^+$.
(b) M does not halt on any string in $(00 + 1)^*$.
(c) M halts on all string ending in a 0.
(d) M halts on all string ending in a 1.

(GATE 2003: 2 Marks)

Solution: M does not halt on any string in $(0+1)^+$.

Ans. (a)

7. Define languages L_0 and L_1 as follows:

$L_0 = (<M, w, 0>|M$ halts on $w)$
$L_1 = \{<M, w, 1>|M$ does not halt on $w\}$

Here $<M, w, i>$ is a triplet, whose first component, M, is an encoding of a Turing machine, second component, w, is a string and third component, i, is a bit. Let $L = L_0 \cup L_1$. Which of the following is true?

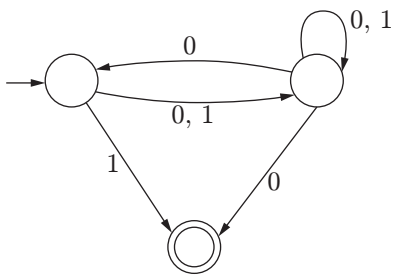
(a) L is recursively enumerable, but $\overline{L}$ is not.
(b) $\overline{L}$ is recursively enumerable, but L is not.
(c) Both L and $\overline{L}$ are recursive.
(d) Neither L nor $\overline{L}$ is recursively enumerable.

(GATE 2003: 2 Marks)

Solution: L contains strings that halts on w and does not halt on w. L can be recursively enumerable if it contains valid strings. So, L is not recursively enumerable, but $\overline{L}$ is recursively enumerable.

Ans. (b)

8. Consider the NFA M shown below.

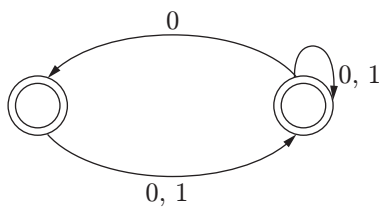


Let the language accepted by M be L. Let L_1 be the language accepted by the NFA M_1 obtained by changing the accepting state of M to a non-accepting state and by changing the non-accepting states of M to accepting states. Which of the following statements is true?

(a) $L_1 = \{0,1\}^* - L$ (b) $L_1 = \{0,1\}^*$
(c) $L_1 \subseteq L$ (d) $L_1 = L$

(GATE 2003: 2 Marks)

Solution:



Strings accepted by resulting NFA: $\{\in,0,1,00,01,10,11....\} = (0+1)^*$

Ans. (b)

9. The problems 3-SAT and 2-SAT are

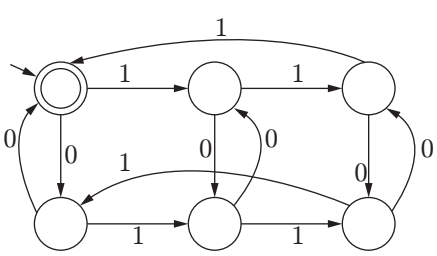
(a) both in P.
(b) both NP-complete
(c) NP-complete and in P, respectively
(d) undecidable and NP-complete, respectively

(GATE 2004: 1 Mark)

Solution: 3-SAT are NP-complete problems. 2-SAT are P problems.

Ans. (c)

10. The following finite state machine accepts all those binary strings in which the number of 1's and 0's are, respectively,



(a) divisible by 3 and 2
(b) odd and even
(c) even and odd
(d) eivisible by 2 and 3

(GATE 2004: 2 Marks)

Solution: The machine accepts strings that have 1's divisible by 3 and 0's divisible by 2. For example, strings 11100, 10011, 00111 will be accepted by this machine and strings 1001, 10111 will be rejected.

Ans. (a)

11. The language $[a^m b^n c^{m+n} \mid m, n \geq 1)$ is

(a) regular
(b) context-free but not regular
(c) context-sensitive but not context-free
(d) type 0 but not context-sensitive

(GATE 2004: 2 Marks)

Solution: The given language can be accepted by PDA, but cannot be accepted by finite automata. So, it is only context-free not regular. Finite automata cannot accept context free language, it only accepts regular language.

Ans. (b)

12. Consider the following grammar G:

$$S \to bS \mid aA \mid b$$
$$A \to bA \mid aB$$
$$B \to bB \mid aS \mid a$$

Let $N_a(w)$ and $N_b(w)$ denote the number of a's and b's in a string w, respectively. The language $L(G) \subseteq \{a,b\}^*$ generated by G is

(a) $\{w|N_a(w) > 3N_b(w)\}$
(b) $\{w|N_b(w) > 3N_b(w)\}$
(c) $\{w|N_a(w) = 3k,\ k \in \{0, 1, 2, ...\}\}$
(d) $\{w|N_b(w) = 3k,\ k \in \{0, 1, 2, ...\}\}$

(GATE 2004: 2 Marks)

Solution: Strings formed from given grammar are: b, bb, $babaab$, $abbb$, and so on. This means the number of a's are multiples of 3.

According to given grammar, string accepted is $baaa$. Options (a), (b) and (d) are false. Only option (c) is correct.

Ans. (c)

13. L_1 is a recursively enumerable language over Σ. An algorithm A effectively enumerates its words as w_1, w_2, w_3, ... Define another language L_2 over $\Sigma \cup \{\#\}$ as $\{w_i \# w_j: w_j \in L_1, i < j\}$. Here # is a new symbol. Consider the following assertions:

S_1: L_1 is recursive implies L_2 is recursive
S_2: L_2 is recursive implies L_1 is recursive

Which of the following statements is true?

(a) Both S_1 and S_2 are true.
(b) S_1 is true but S_2 is not necessarily true.
(c) S_2 is true but S_1 is not necessarily true.
(d) Neither is necessarily true.

(GATE 2004: 2 Marks)

Solution: For L_1, the membership algorithm can be constructed as follows. For an arbitrary $w_i \# w_j$,

(i) If w_i, $w_j \in L_1$ and $i < j$ then $w_i \# w_j$
(ii) If $w_i \notin L_1$ or $w_j \notin L_2$ or $i \geq j$, then $w_i \# w_j \notin L_2$.

This implies that L_2 is also recursive, so S_1 is true. A similar membership algorithm is not possible for L_2, so S_2 is not necessarily true.

Ans. (b)

14. Consider three decision problems P_1, P_2 and P_3. It is known that P_1 is decidable and P_2 is undecidable. Which one of the following is TRUE?

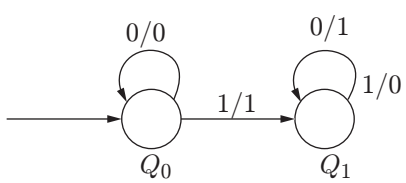
(a) P_3 is decidable if P_1 is reducible to P_3.
(b) P_3 is undecidable if P_3 is reducible to P_2.
(c) P_3 is undecidable if P_2 is reducible to P_3.
(d) P_3 is decidable if P_3 is reducible to P_2's complement.

(GATE 2005, 2 Marks)

Solution: Any problem P is undecidable if any known undecidable problem can be reduced to P. So, option (c) is correct.

Ans. (c)

15. The following diagram represents a finite state machine which takes as input a binary number from the least significant bit.



Which one of the following is TRUE?

(a) It computes 1's complement of the input number
(b) It computes 2's complement of the input number
(c) It increments the input number
(d) It decrements the input number

(GATE 2005: 2 Marks)

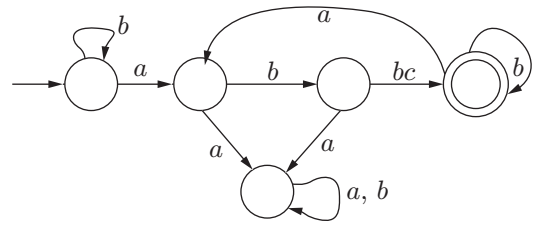
Solution: Output function is:

PS	0	1
0	0	1
1	1	0

The output computes 2's complement of input number.

Ans. (b)

16. Consider the machine M:



The language recognized by M is
(a) $\{w \in \{a, b\}^*$ every a in w is followed by exactly two b's$\}$.
(b) $\{w \in \{a, b)^*$ every a in w is followed by at least two b's$\}$.
(c) $\{w \in (a, b\}^*$ w contains the substring "abb"$\}$.
(d) $\{w \in \{a, b\}^*$ w does not contain "aa" as a substring$\}$.

(GATE 2005: 2 Marks)

Solution: DFA contains every "a" followed by at least two b's due to self-loops.

Ans. (b)

17. Let N_f and N_p denote the classes of languages accepted by non-deterministic finite automata and non-deterministic push-down automata, respectively. Let D_f and D_p denote the classes of languages accepted by deterministic finite automata and deterministic push-down automata, respectively.

Which one of the following is TRUE?

(a) $D_f \subset N_f$ and $D_p \subset N_p$
(b) $D_f \subset N_f$ and $D_p = N_p$
(c) $D_f = N_f$ and $D_p = N_p$
(d) $D_f = N_f$ and $D_p \subset N_p$

(GATE 2005: 2 Marks)

Solution: Power of NFA and DFA is same. But non-deterministic PDA is more powerful than deterministic PDA.

Ans. (d)

18. Consider the languages

$L_1 = \{a^n b^n c^m \mid n, m > 0\}$ and $L_2 = \{a^n b^m c^m \mid n, m > 0\}$

Which one of the following statements is FALSE?

(a) $L_1 \cap L_2$ is a context-free language.
(b) $L_1 \cup L_2$ is a context-free language.
(c) L_1 and L_2 are context-free languages.
(d) $L_1 \cap L_2$ is a context-sensitive language.

(GATE 2005: 2 Marks)

Solution: L_1 and L_2 both are context-free languages. Their intersection may or may not be context-free. So, option (a) is false.

Ans. (a)

19. Let L_1 be a recursive language, and let L_2 be a recursively enumerable but not a recursive language. Which one of the following is TRUE?

(a) $\overline{L_1}$ is recursive and $\overline{L_2}$ is recursively enumerable.
(b) $\overline{L_1}$ is recursive and $\overline{L_2}$ is not recursively enumerable.
(c) $\overline{L_1}$ and $\overline{L_2}$ are recursively enumerable.
(d) $\overline{L_1}$ is recursively enumerable and $\overline{L_2}$ is recursive.

(GATE 2005: 2 Marks)

Solution: Recursive languages are closed under complementation but recursive enumerable languages are not.

Ans. (b)

20. Consider the languages

$$L_1 = \{ww^R \mid w \in \{0, 1\}^*\}$$
$$L_2 = \{w \# w^R \mid w \in \{0, 1\}^*\},$$

where # is a special symbol

$$L_3 = \{ww \mid w \in \{0, 1\}^*\}$$

Which one of the following is TRUE?

(a) L_1 is a deterministic CFL.
(b) L_2 is a deterministic CFL.
(c) L_3 is a CFL, but not a deterministic CFL.
(d) L_3 is a deterministic CFL.

(GATE 2005: 2 Marks)

Solution: L_2 can be accepted by DPDA, so it is a deterministic context free language. L_1 is non-deterministic context free and L_3 in not context free language.

Ans. (b)

21. Let $L_1 = \{0^{n+m}1^n0^m | n, m \geq 0\}$,
$L_2 = \{0^{n+m}1^{n+m}0^m | n, m \geq 0\}$ and
$L_3 = \{0^{n+m}1^{n+m}0^{n+m} | n, m \geq 0\}$

Which of these languages are NOT context-free?

(a) L_1 only
(b) L_3 only
(c) L_1 and L_2
(d) L_2 and L_3

(GATE 2006: 1 Mark)

Solution: Only L_1 is accepted by pushdown automata as only L_1 can be written as a context-free language. L_2 is not context free, while L_3 is context sensitive.

Ans. (d)

22. If s is a string over $(0 + 1)^*$, then let $n_0(s)$ denote the number of 0's in s and $n_1(s)$ the number of 1's in s. Which one of the following languages is not regular?

(a) $L = [s \in (0 + 1)^* | n_0(s)$ is a 3-digit prime$\}$
(b) $L = \{s \in (0 + 1)^* |$for every prefix s' of s, $|n_0(s') - n_1(s')| \leq 2\}$
(c) $L = \{s \in (0 + 1)^* | \, |n_0(s) - n_1(s) \leq 4\}$
(d) $L = \{s \in (0 + 1)^* | n_0(s) \bmod 7 = n_1(s) \bmod 5 = 0\}$

(GATE 2006: 2 Marks)

Solution: Option (a) is finite, can be accepted by DFA.
Option (b), only prefix comparison is possible by DFA, so it is regular.
Option (c), whole string is compared, which is not possible by DFA because it does not have memory.
Option (d), mod machines can be made using DFA.

Ans. (c)

Linked Answer Questions 23 and 24:

23. Which one of the following grammars generates the language $L = \{a^i\, b^i \mid i \neq j)$?

(a) $S \to AC|CB$
$C \to aCb|a|b$
$A \to aA|\in$
$B \to Bb|\in$

(b) $S \to aS|Sb|a|b$

(c) $S \to AC|CB$
$C \to aCb|\in$
$A \to aA|\in$
$B \to Bb|\in$

(d) $S \to AC|CB$
$C \to aCb|\in$
$A \to aA|a$
$B \to Bb|b$

(GATE 2006: 2 Marks)

Solution: Both options (a) and (b) can produce equal number of a's and b's, which should not be produced according to grammar. Option (d) cannot produce all the strings. So, option (c) is the correct answer.

Ans. (c)

24. In the correct grammar above, what is the length of the derivation (number of steps starring from S) to generate the string $a^l\, b^m$ with $l \neq m$?

(a) max(l, m) + 2
(b) $l + m + 2$
(c) $l + m + 3$
(d) max(l, m) + 3

(GATE 2006: 2 Marks)

Solution: This can be checked by producing various strings of different length.

$$S \to AC$$
$$S \to aC$$
$$S \to aaCb$$
$$S \to aab$$

Here $l = 2$ and $m = 1$, which satisfies the equation max $(l, m) + 2 = 4$.

Ans. (a)

25. For $s \in (0 + 1)^*$, let $d(s)$ denote the decimal value of s (e.g. $d(101) = 5$).

Let $L = \{s \in (0 + 1)^*\text{l } d(s) \bmod 5 = 2 \text{ and } d(s) \bmod 7 \neq 4\}$

Which one of the following statements is true?

(a) L is recursively enumerable, but not recursive.
(b) L is recursive, but not context-free.
(c) L is context-free, but not regular.
(d) L is regular.

(GATE 2006: 2 Marks)

Solution: DFA is possible for L. If for any language, we can make DFA then that language will be regular language. So, L is regular.

Ans. (d)

26. Consider the following statements about the context-free grammar:

$$G = \{S \to SS,\ S \to ab,\ S \to ba,\ S \to \varepsilon\}$$

I. G is ambiguous.
II. G produces all strings with equal number of a's and b's.
III. G can be accepted by a deterministic PDA.

Which combination below expresses all the true statements about G?

(a) I only (b) I and III only
(c) II and III only (d) I, II, and III

(GATE 2006: 2 Marks)

Solution:

I. G is ambiguous. Null can be produced using multiple rules [$S \to \varepsilon$ and $S \to SS$]. Two parse trees will be formed.
II. It cannot produce all the combinations of equal a's and b's such as string $aabb$ is not possible.
III. Language accepted is $(ab + ba)^*$, which is regular. Regular languages are also accepted by DPDA.

Ans. (b)

27. Let L_1 be a regular language, L_2 be a deterministic context-free language and L_3 a recursively enumerable, but not recursive, language. Which one of the following statements is false?

(a) $L_1 \cap L_2$ is a deterministic CFL.
(b) $L_3 \cap L_1$ is recursive.
(c) $L_1 \cup L_2$ is context-free.
(d) $L_1 \cap L_2 \cap L_3$ is recursively enumerable.

(GATE 2006: 2 Marks)

Solution:
Option (a) is true. DCFL is closed under regular intersection.
Option (b) is false. Intersection is recursively enumerable.
Options (c) and (d) are true.

Ans. (b)

28. Consider the regular language $L = (111 + 11111)^*$. The minimum number of states in any DFA accepting this languages is

(a) 3 (b) 5
(c) 8 (d) 9

(GATE 2006: 2 Marks)

Solution: Nine states are required. The length of string will be 8. Total states required are 8 + 1 =9.

Ans. (d)

29. Which of the following problems is undecidable?

(a) Membership problem for CFGs
(b) Ambiguity problem for CFGs
(c) Finiteness problem for FSAs
(d) Equivalence problem for FSAs

(GATE 2007: 1 Mark)

Solution: No algorithm exists to check the ambiguity of grammar.

Ans. (b)

30. Which of the following is **TRUE**?

(a) Every subset of a regular set is regular.
(b) Every finite subset of a non-regular set is regular.
(c) The union of two non-regular sets is not regular.
(d) Infinite union of finite sets is regular.

(GATE 2007: 1 Mark)

Solution: In option (a), $a^n b^n$ is subset of $a^*\ b^*$; but it is not regular.
Option (b) is correct.
Option (c) is union of two context-free languages, and thus is regular.
Option (d) is infinite union of finite sets, and thus is not regular.

Ans. (b)

31. A minimum state deterministic finite automaton accepting the language

$L = \{w|\ w \in \{0, 1\}^*$, number of 0's and 1's in w are divisible by 3 and 5, respectively$\}$ has

(a) 15 states (b) 11 states
(c) 10 states (d) 9 states

(GATE 2007: 2 Marks)

Solution: Divisible by 3 and $5 = 3 \times 5 = 15$ states are required.

Ans. (a)

32. The language $L = \{0^i\, 21^i\ i \geq 0\}$ over the alphabet $\{0, 1, 2\}$ is

(a) not recursive
(b) recursive and is a deterministic CFL
(c) a regular language
(d) not a deterministic CFL but a CFL

(GATE 2007: 2 Marks)

Solution: L is accepted by DPDA. So, it is deterministic CFL.

Ans. (b)

33. Which of the following languages is regular?

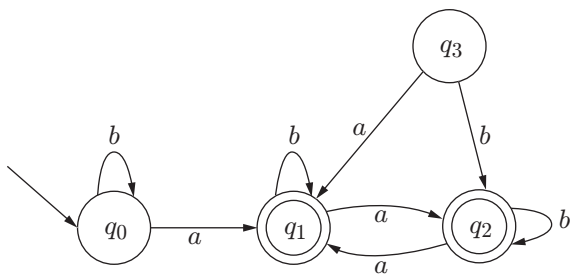
(a) $\{ww^R / w \in \{0, 1\}^+\}$
(b) $\{ww^R x / x, w \in \{0, 1\}^+\}$
(c) $\{wxw^R / x, w \in \{0, 1\}^+\}$
(d) $\{xww^R / x, w \in \{0, 1\}^+\}$

(GATE 2007: 2 Marks)

Solution: Finite automata do not have memory element. So, they cannot remember the symbols. Hence, options (a) and (d) are incorrect.

Ans. (c)

Common Data Questions 34 and 35: Consider the following finite state automaton



34. The language accepted by this automaton is given by the regular expression

(a) $b^*ab^*ab^*ab^*$
(b) $(a + b)^*$
(c) $b^*a\,(a + b)^*$
(d) $b^*ab^*ab^*$

(GATE 2007: 2 Marks)

Solution: One a is compulsory. Minimum string is 'a' that is accepted by the automaton. $b^*a\,(a + b)^*$ is regular and accepted by the automaton.

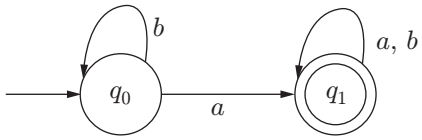
Ans. (c)

35. The minimum state automaton equivalent to the above FSA has the following number of states

(a) 1
(b) 2
(c) 3
(d) 4

(GATE 2007: 2 Marks)

Solution: Minimized DFA is:



Ans. (b)

36. Which of the following is true for the language $\{a^P \mid P \text{ is a prime}\}$?

(a) It is not accepted by a Turing Machine.
(b) It is regular but not context-free.
(c) It is context-free but not regular.
(d) It is neither regular nor context-free, but accepted by a Turing machine.

(GATE 2008: 1 Mark)

Solution: The options are checked by using pumping lemma for regular and context-free languages. It is found that the language is not regular, and hence not context free; but is accepted by the Turing machine.

Ans. (d)

37. Which of the following are decidable?

I. Whether the intersection of two regular languages is infinite.
II. Whether a given context-free language is regular.
III. Whether two pushdown automata accept the same language.
IV. Whether a given grammar is context-free.

(a) I and II
(b) I and IV
(c) II and III
(d) II and IV

(GATE 2008: 1 Mark)

Solution: Options II and III are undecidable. The intersection of two regular languages is determined by an algorithm, and it can be determined whether the given grammar is context-free or not. So, options I and II are decidable.

Ans. (b)

38. If L and $\bar{L}$ are recursively enumerable, then L is

(a) Regular
(b) Context-free
(c) Context-sensitive
(d) Recursive

(GATE 2008: 1 Mark)

Solution: A theorem states that when L and $\bar{L}$ both are RE, then L is recursive.

Ans. (d)

39. Which of the following statements is false?

(a) Every NFA can be converted to an equivalent DFA.
(b) Every non-deterministic Turing machine can be converted to an equivalent deterministic Turing machine.
(c) Every regular language is also a context-free language.
(d) Every subset of a recursively enumerable set is recursive.

(GATE 2008: 2 Marks)

Solution: Option (d) is not necessarily true. Every recursive set is recursively enumerable, but the reverse is not necessarily true.

40. Given below are two finite state automata (→ indicates the start state and F indicates the final state)

Y:

	a	b
→1	1	2
2(F)	2	1

Z:

	a	b
→1	2	2
2(f)	1	1

Which of the following represents the product automaton $Z \times Y$?

(a)

	a	b
→P	S	R
Q	R	S
R(F)	Q	P
S	Q	P

(b)

	a	b
→P	Q	S
Q	R	S
R(F)	Q	P
S	Q	P

(c)

	a	b
→P	S	Q
Q	R	S
R(F)	Q	P
S	P	Q

(d)

	a	b
→P	S	Q
Q	S	R
R(F)	Q	P
A	Q	P

(GATE 2008: 2 Marks)

Solution: Option (a) represents the product correctly.

Ans. (a)

41. Which of the following statements are true?

I. Every left-recursive grammar can be converted to a right-recursive grammar and vice versa.
II. All ε productions can be removed from any context-free grammar by suitable transformations.
III. The language generated by a context-free grammar all of whose productions are of the form $X \to w$ or $X \to wY$ (where w is a string of terminals and Y is a non-terminal) is always regular.
IV. The derivation trees of strings generated by a context-free grammar in Chomsky normal form are always binary trees.

(a) I, II, III and IV
(b) II, III and IV only
(c) I, III and IV only
(d) I, II and IV only

(GATE 2008: 2 Marks)

Solution: Option II: If ε belongs to $L(G)$, then the statement becomes false.

Ans. (c)

42. Match the following.

List I	List II
(E) Checking that identifiers are declared before their use	(P) $L = \{a^n b^m c^n d^m \mid n \geq 1, m \geq 1\}$
(F) Number of formal parameters in the declaration of a function agrees with the number of actual parameters in use of that function	(Q) $X \to XbX \mid XcX \mid dXf \mid g$
(G) Arithmetic expressions with matched pairs of parantheses	(R) $L = \{wcw \mid w \in (a \mid b)^*\}$
(H) Palindromes	(S) $X \to bXb \mid cXc \mid \varepsilon$

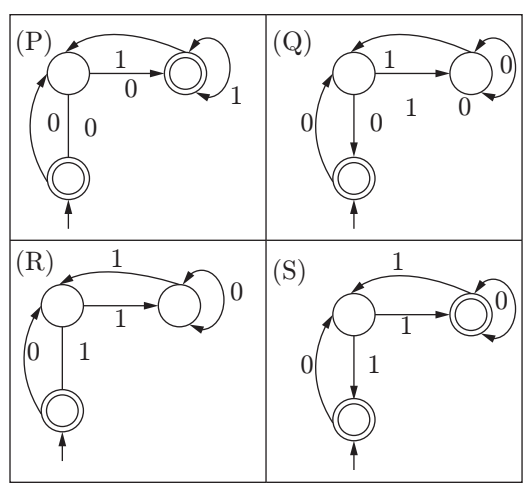
Codes:
(a) E – P, F – R, G – Q, H – S
(b) E – R, F – P, G – S, H – Q
(c) E – R, F – P, G – Q, H – S
(d) E – P, F – R, G – S, H – Q

(GATE 2008: 2 Marks)

Solution:
E – R: In R, first w is checking the declaration of an identifier.
F – P: Actual parameters $a^n b^m$ and formal parameters $c^n d^m$
G –Q: Arithmetic expressions with matched pair of parenthesis
H – S: S is generating palindrome strings.

Ans. (c)

43. Match the following NFAs with the regular expressions they correspond to



1. $\varepsilon + 0(01^*1 + 00)^*01^*$
2. $\varepsilon + 0(10^*1 + 00)^*0$
3. $\varepsilon + 0(10^*1 + 10)^*1$
4. $\varepsilon + 0(10^*1 + 10)^*10^*$

Codes:
(a) P – 2, Q – 1, R – 3, S – 4
(b) P – 1, Q – 3, R – 2, S – 4
(c) P – 1, Q – 2, R – 3, S – 4
(d) P – 3, Q – 2, R – 1, S – 4

(GATE 2008: 2 Marks)

Solution: P $\rightarrow \varepsilon + 0(01^*1 + 00)^*01^*$
Q $\rightarrow \varepsilon + 0(10^*1 + 00)^*0$
R $\rightarrow \varepsilon + 0(10^*1 + 10)^*1$
S $\rightarrow \varepsilon + 0(10^*1 + 10)^*10^*$

Ans. (c)

44. Which of the following are regular sets?

I. $\{a^n b^{2m} | n \geq 0, m \geq 0\}$
II. $\{a^n b^m | n = 2m\}$
III. $\{a^n b^m | n \neq m\}$
IV. $\{xcy | x, y \in (a, b)^*\}$

(a) I and IV only (b) I and III only
(c) I only (d) IV only

(GATE 2008: 2 Marks)

Solution: In options II and III, m and n are dependent. As DFA do not have memory, cannot be accepted by DFA. Options I and IV are regular sets.

Ans. (a)

45. $S \rightarrow aSa|bSb|a|b$

The language generated by the above grammar over the alphabet $\{a, b\}$ is the set of

(a) all palindromes.
(b) all odd length palindromes.
(c) strings that begin and end with the same symbol.
(d) all even length palindromes.

(GATE 2009: 1 Mark)

Solution: Strings generated by this grammar are odd length palindromes, for example, *aaa*, *aba*, *ababa*.

Ans. (b)

46. Which one of the following languages over the alphabet {0, 1} is described by the regular expression (0 + 1)*0(0 + 1)*0(0 + 1)*?

(a) The set of all strings containing the substring 00.
(b) The set of all strings containing at most two 0's.
(c) The set of all strings containing at least two 0's.
(d) The set of all strings that begin and end with either 0 or 1.

(GATE 2009, 1 Mark)

Solution: Given regular expression must have atleast two zeros 0(0 + 1)*0….

Ans. (c)

47. Which one of the following is FALSE?

(a) There is a unique minimal DFA for every regular language.
(b) Every NFA can be converted to an equivalent PDA.
(c) Complement of every context-free language is recursive.
(d) Every non-deterministic PDA can be converted to an equivalent deterministic PDA.

(GATE 2009: 1 Mark)

Solution: Both non-deterministic and deterministic PDA have different powers. Non-deterministic PDA can accept ambiguous grammars also.

Ans. (d)

48. Match all items in Group 1 with correct options from those given in Group 2.

Group 1	**Group 2**
P. Regular expression	**1.** Syntax analysis
Q. Pushdown automata	**2.** Code generation
R. Dataflow analysis	**3.** Lexical analysis
S. Register allocation	**4.** Code optimization

Codes:
(a) P – 4, Q – 1, R – 2, S – 3
(b) P – 3, Q – 1, R – 4, S – 2
(c) P – 3, Q – 4, R – 1; S – 2
(d) P – 2, Q – 1, R – 4, S – 3

(GATE 2009: 1 Mark)

Solution: Regular expressions: Lexical analysis
Pushdown automata: Syntax analysis
Dataflow analysis: Code optimization
Register allocation: Code generation

Ans. (b)

49. Given the following state table of an FSM with two states A and B, one input and one output;

Present State *A*	**Present State *B***	**Input State *C***	**Next State *A***	**Next State *B***	**Output**
0	0	0	0	0	1
0	1	0	1	0	0
1	0	0	0	1	0

(*Continued*)

Present State A	Present State B	Input State C	Next State A	Next State B	Output
1	1	0	1	0	0
0	0	1	0	1	0
0	1	1	0	0	1
1	0	1	0	1	1
1	1	1	0	—	1

If the initial state is $A = 0$, $B = 0$, what is the minimum length of an input string which will take the machine to the state $A = 0$, $B = 1$ with output = 1?

(a) 3 (b) 4 (c) 5 (d) 6

(GATE 2009: 2 Marks)

Solution:

Present State	Output
(0, 0) →	1
(0, 1) →	0
(1, 0) →	0
(0, 1) →	1

With $A = 0$, $B = 1$ and output = 1 will require a string of length 3.

Ans. (a)

50. Let $L = L_1 \cap L_2$, where L_1 and L_2 are languages as defined below:

$$L_1 = \{a^m b^m c a^n b^n \mid m, n^3 \geq 0\}$$
$$L_2 = \{a^i b^i c^k \mid i, j, k \geq 0\}$$

Then L is

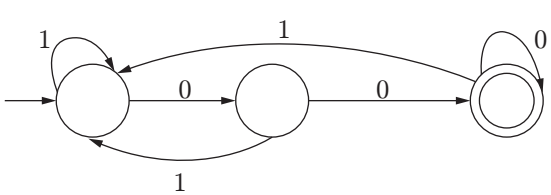
(a) not recursive.
(b) regular.
(c) context-free but not regular.
(d) recursively enumerable but not context-free.

(GATE 2009: 2 Marks)

Solution: Intersection of two regular languages is regular, but L_1 is not regular.

Ans. (c)

51. In the following figure, DFA accepts the set of all strings over {0, 1} that



(a) begin either with 0 or 1
(b) end with 0
(c) end with 00
(d) contain the substring 00

(GATE 2009, 2 Marks)

Solution: The given DFA accepts all the strings ending with 00.

Ans. (c)

52. Let L_1 be a recursive language. Let L_2 and L_3 be languages that are recursively enumerable but not recursive. Which of the following statements is not necessarily true?

(*a*) $L_2 - L_1$ is recursively enumerable.
(*b*) $L_1 - L_3$ is recursively enumerable.
(*c*) $L_2 \cap L_3$ is recursively enumerable.
(*d*) $L_2 \cup L_3$ is recursively enumerable.

(GATE 2010: 1 Mark)

Solution:
Subtraction of recursive language and recursive enumerable is not necessarily recursive enumerable.

Ans. (b)

53. Let $L = \{w \in (0 + 1)^* | w$ has even number of 1's$\}$, i.e., L is the set of all bit strings with even number of 1's. Which one of the regular expressions below represents L?

(a) (0*10*1)* (b) 0*(10*10*)*
(c) 0*(10*1)*0* (d) 0*1(10*1)*10*

(GATE 2010: 2 Marks)

Solution: Option (c) produces odd 1's also, option (a) produces strings ending with 1 and option (d) does not produce zero 1's.

Ans. (b)

54. Consider the languages

$$L_1 = \{0^i 1^j \mid i \neq j\},\ L_2 = \{0^i 1^j \mid i = j\},$$
$$L_3 = \{0^i 1^j \mid i = 2j + 1\},\ L_4 = \{0^i 1^j \mid i \neq 2j\}$$

Which one of the following statements is true?

(a) Only L_2 is context-free
(b) Only L_2 and L_3 are context-free
(c) Only L_1 and L_2 are context-free
(d) All are context-free

(GATE 2010: 2 Marks)

Solution: All are context-free languages as they are recognized by pushdown automata.

Ans. (d)

55. Let w be any string of length n in $\{0, 1\}^*$.
Let L be the set of all substrings of w.
What is the minimum number of states in a non-deterministic finite automaton that accepts L?

(a) $n - 1$ (b) n
(c) $n + 1$ (d) 2^{n-1}

(GATE 2010: 2 Marks)

Solution: $n + 1$ states are required for string of length n.

Ans. (c)

56. Let P be a regular language and Q be a context-free language such that $Q \subseteq P$. (For example, let P be the language represented by the regular expression p^*q^* and Q be $[p^n q^n \mid n \in N]$. Then which of the following is ALWAYS regular?

(a) $P \cap Q$ (b) $P - Q$
(c) $\Sigma^* - p$ (d) $\Sigma^* - Q$

(GATE 2011: 1 Mark)

Solution: Regular languages are closed under complementation.

Ans. (c)

57. The lexical analysis for a modern computer language such as Java needs the power of which one of the following machine models in a necessary and sufficient sense?

(a) Finite state automata
(b) Deterministic pushdown automata
(c) Non-deterministic pushdown automata
(d) Turing machine

(GATE 2011: 1 Mark)

Solution: Tokens are recognized in lexical analysis phase of compiler.

Ans. (c)

58. Which of the following pairs have DIFFERENT expressive power?

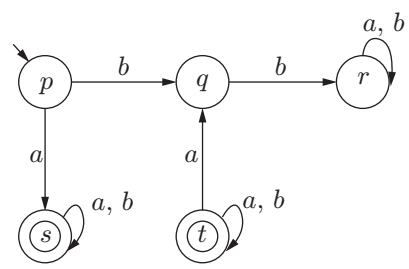
(a) Deterministic finite automata (DFA) and non-deterministic finite automata (NFA)
(b) Deterministic pushdown automata (DPDA) and non-deterministic pushdown automata (NPDA)
(c) Deterministic single-tape Turing machine and Non-deterministic single-tape Turing machine
(d) Single-tape Turing machine and multi-tape Turing machine

(GATE 2011: 1 Mark)

Solution: NDPDA is more powerful than DPDA.

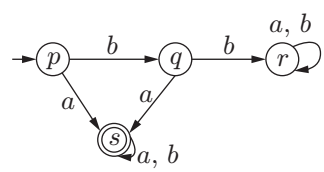
Ans. (b)

59. A deterministic finite automaton (DFA) D with alphabet $\Sigma = \{a, b\}$ is given below.

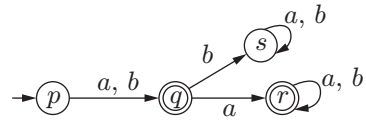


Which of the following finite state machines is a valid minimal DFA which accepts the same language as D?

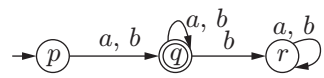
(a)



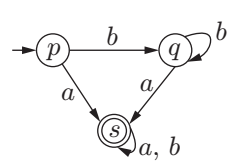
(b)



(c)



(d)



(GATE 2011: 2 Marks)

Solution: Given DFA does not accept "*bba*" which is accepted by option (c) and (d), and string "*b*" accepted by option (b). So, option (a) is answer.

Ans. (a)

60. Definition of a language L with alphabet $\{a\}$ is given as following.

$L = \{a^{nk} \mid k < 0$, and n is a positive integer constant$\}$
What is the minimum number of states needed in a DFA to recognize L?

(a) $(k + 1)$ (b) $(n + 1)$
(c) 2^{k+1} (d) 2^{n+1}

(GATE 2011: 2 Marks)

Solution: Let $n = 4$, so $L = a^{4k}$

Five states are required for this machine. $(n + 1)$ states are required for such a machine.

Ans. (b)

61. Consider the languages L_1, L_2 and L_3 as given below.

$L_1 = \{0^p 1^q | p, q \in N)$
$L_2 = \{0^p 1^q | p, q \in N$ and $p = q\}$
$L_3 = \{0^p 1^q 0^r | p, q, r \in N, r \in N$ and $p = q = r)$.

Which of the following statements is NOT TRUE?

(a) Pushdown automata (PDA) can be used to recognize L_1 and L_2.
(b) L_1 is a regular language.
(c) All the three languages are context-free.
(d) Turing machines can be used to recognize all the languages.

(GATE 2011: 2 Marks)

Solution: L_1 is a regular language; L_2 is a context-free language, and L_3 is a context-sensitive language.

Ans. (c)

62. Given the language $L = \{ab, aa, baa\}$, which of the following strings are in L^*?

1. *abaabaaabaa*
2. *aaaabaaaa*
3. *baaaaabaaaab*
4. *baaaaaba*

(a) 1, 2 and 3 (b) 2, 3 and 4
(c) 1, 2 and 4 (d) 1, 3 and 4

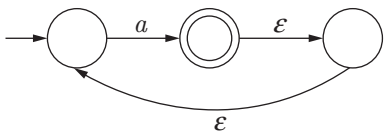
(GATE 2012: 1 Mark)

Solution: *baaaaabaaaab* is not derived from the given language.

Ans. (c)

63. What is the complement of the language accepted by the NFA shown below?

Assume $\Sigma = \{a\}$ and ε is the empty string.

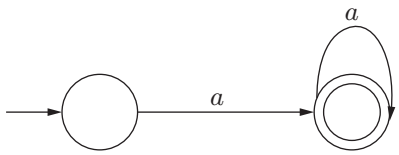


(a) $\varnothing$ (b) $\{\varepsilon\}$
(c) a (d) $\{a, \varepsilon\}$

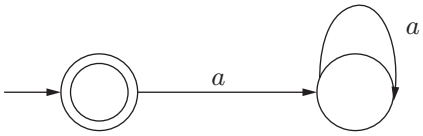
(GATE 2012: 1 Mark)

Solution: Language accepted by this machine is a^+.

First convert NFA to DFA:



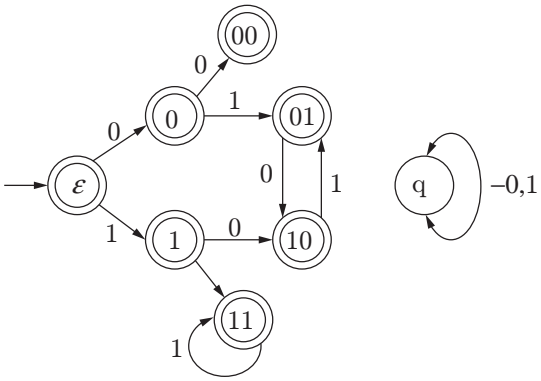
Take compliment of this DFA:



Language accepted is $\{\varepsilon\}$.

Ans. (b)

64. Consider the set of strings on $\{0, 1\}$ in which *every substring of* 3 *symbols* has at most *two* zeros. For example, 001110 and 011001 are in the language, but 100010 is not. All strings of length less than 3 are also in the language. A partially completed DFA that accepts this language is shown below.



The missing arcs in the DFA are

(a)

	00	01	10	11	q
00	1	0			
01				1	
10	0				
11			0		

(b)

	00	01	10	11	q
00		0			1
01		1			
10				0	
11		0			

(c)

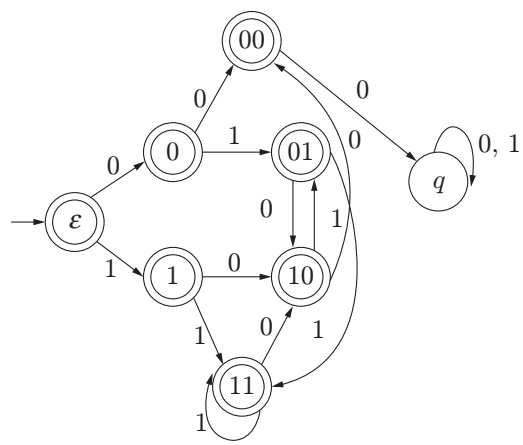
	00	01	10	11	q
00		1			0
01		1			
10			0		
11		0			

(d)

	00	01	10	11	q
00		1			0
01				1	
10	0				
11			0		

(GATE 2012: 2 Marks)

Solution: Complete DFA is given below:



Ans. (d)

65. Consider the following logical inferences.

I_1: If it rains then the cricket match will not be played. The cricket match was played.
Inference: There was no rain.
I_2: If it rains then the cricket match will not be played. It did not rain.
Inference: The cricket match was played.

Which of the following is **TRUE**?

(a) Both I_1 and I_2 are correct inferences
(b) I_1 is correct but I_2 is not a correct inference
(c) I_1 is not correct but I_2 is a correct inference
(d) Both I_1 and I_2 are not correct inferences

(GATE 2012: 2 Marks)

Solution:
R: It rains.
C: Cricket match played.
I_1: R → ∼ C can be written as ~R U ∼C

$$\frac{\begin{array}{r} \sim R \text{ U } \sim C \\ C \end{array}}{\sim R}$$

I_2: R → ∼ C can be written as ∼R U ∼C

$$\frac{\begin{array}{r} \sim R \text{ U } \sim C \\ \sim R \end{array}}{\sim R \text{ U } C}$$

I_1 is correct but I_2 is not a correct inference.

Ans. (b)

66. Which of the following problems are decidable?

1. Does a given program ever produce an output?
2. If L is a context-free language, then, is L also context-free?
3. If L is a regular language, then, is L also regular?
4. If L is a recursive language, then, is L also recursive?

(a) 1, 2, 3, 4 (b) 1, 2
(c) 2, 3, 4 (d) 3, 4

(GATE 2012: 2 Marks)

Solution: Regular and recursive languages are closed under complementation, but CFLs are not.

Ans. (d)

67. Consider the languages $L_1 = \Phi$ and $L_2=\{a\}$. Which one of the following represents $L_1L_2^* \cup L_1^*$?

(a) $\{\varepsilon\}$ (b) Φ
(c) a^* (d) $\{\varepsilon, a\}$

(GATE 2013: 1 Mark)

Solution:

$$L_1^* = \varepsilon. \text{ So, } L_1L_2^* \cup L_1^* = \varepsilon$$

Ans. (a)

68. Which of the following statements is/are **FALSE**?

1. For every non-deterministic Turing machine, there exists an equivalent deterministic Turing machine.
2. Turing recognizable languages are closed under union and complementation.
3. Turing decidable languages are closed under intersection and complementation.
4. Turing recognizable languages are closed under union and intersection.

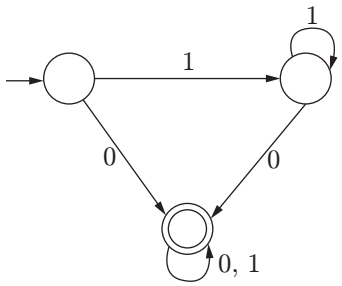
(a) 1 and 4 only (b) 1 and 3 only
(c) 2 only (d) 3 only

(GATE 2013: 1 Mark)

Solution:
Option (a) is true. The power of non-deterministic and deterministic Turing machines is equivalent.
Option (b) is True.
Option (c) is false. They are not closed under complementation.
Option (d) is true.

Ans. (c)

69. Consider the DFA A given below.



Which of the following are **FALSE?**

1. Complement of $L(A)$ is context-free.
2. $L(A) = L((11^*0 + 0)(0 + 1)^*0^*1^*)$
3. For the language accepted by A, A is the minimal DFA.
4. A accepts all strings over $\{0, 1\}$ of length at least 2.

(a) 1 and 3 only (b) 2 and 4 only
(c) 2 and 3 only (d) 3 and 4 only

(GATE 2013, 2 Marks)

Solution: Option (3) is false because minimal DFA can be constructed using two states only. Option (4) is false, DFA accepts string "0", of length 1.

Ans. (d)

70. Which of the following is/are undecidable?

1. G is a CFG. Is $L(G) = \varnothing$?

2. G is a CFG. Is $L(G) = \Sigma^*$?

3. M is a Turing machine. Is $L(M)$ regular?

4. A is a DFA and N is an NFA. Is $L(A) = L(N)$?

(a) 3 only (b) 3 and 4 only
(c) 2 and 3 only (d) 2 and 3 only

(GATE 2013: 2 Marks)

Solution: CFG is empty or not is decidable, and to test whether language accepted by DFA and NFA is same is also decidable.

Ans. (d)

71. Consider the following languages.

$$L_1 = \left\{0^p 1^q 0^r \,|p, q, r \geq 0\right\}$$

$$L_2 = \left\{0^p 1^q 0^r \,|p, q, r \geq 0, p \neq r\right\}$$

Which one of the following statements is **FALSE**?

(GATE 2013: 2 Marks)

(a) L_2 is context-free.
(b) $L_1 \cap L_2$ is context-free.
(c) Complement of L_2 is recursive.
(d) Complement of L_1 is context-free but not regular.

Solution: L_1 is regular and complement of Regular language is also regular.

Ans. (d)

PRACTICE EXERCISES

Set 1

1. What is the meaning of the regular expression Σ^* 0101 Σ^*?

(a) Any string containing "1" as substring
(b) Any string containing "01" as substring
(c) Any string containing "0101" as substring
(d) All strings containing "0101" as substring

2. Which of the following is called a regular operation?

(a) Union, star
(b) Concatenation
(c) Star, concatenation
(d) Concatenation, star, union

3. In Mealy machine, if we enter the string of length 5, then what will be the output string length?

(a) 4 (b) 5
(c) 6 (d) 1

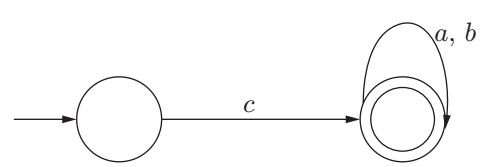
4. The string 1101 does not refer to the set represented by

(a) $110^*(0 + 1)$ (b) $1(0 + 1)^* 101$
(c) $(10)^*(01)^*(00 + 11)$ (d) $(11 + 00) + 01$

5. The grammar $S \to 0S1$, $S \to 0S$, $S \to S1$, $S \to 0$ will generate a

(a) Context-free language
(b) Regular language
(c) Context-sensitive language
(d) Push-Down Automata

6. Which of the string is accepted by a given NFA?



(a) $c.(a \cup b)^*$ (b) $c.a^*b^*$
(c) $c \cdot (ab)^*$ (d) $c \cdot (ab)$

7. Let L denotes the language generated by the grammar $S \to 0S0|00$. Which of the following is true?

(a) $L = 0^+$
(b) L is regular but not 0^+
(c) L is context-free but not regular
(d) L is context-free

8. Given an arbitrary non-deterministic finite automaton (NDFA) with N states, the maximum number of states in an equivalent minimized DFA is

(a) N^2 (b) 2^N
(c) $2N$ (d) $2^*N!$

9. A language accepted by pushdown automaton is

(a) Regular (b) Context-free
(c) Context-sensitive (d) Recursive

10. Which one of the following is a regular language?

(a) $\{ww \mid n \leq m,\ w \in \{0, 1\}^*\}$
(b) $\{ww^R \mid w \in \{0^*, 1^*\}\}$
(c) $\{w^n w^m \mid n \leq m,\ w \in \{0, 1\}\}$
(d) $\{w(w^R \mid w \in \{0, 1\}^*\}$

11. DFA and NFA when compared on computational power are

(a) DFA $\subseteq$ NFA
(b) NFA $\subseteq$ DFA
(c) DFA = NFA
(d) They cannot be compared

12. Number of states in DFA accepting the strings containing at most $2a$'s and at most $2b$'s over $\{a, b\}$ is

(a) 4 (b) 10
(c) 6 (d) 11

13. Find the minimum number of states in the PDA accepting language

$$L = \{a^n b^m \mid n > m;\ m, n > 0\}$$
$$\Sigma = \{a, b\}, \text{ and } \Gamma = \{z, a\}$$

(a) n^*m (b) 0
(c) 4 (d) 1

14. The context-free grammar which generates the language $L = \{a^n b^m \mid m = n \bmod 3\}$ is

(a) $S \to AB$
$A \to aaabbA|\varepsilon$
$B \to aabb|\varepsilon$

(b) $S \to AB$
$A \to AA|\varepsilon$
$B \to BBBA|\varepsilon$

(c) $S \to AB$
$A \to aaaA|\varepsilon$
$B \to ab|aabb|\varepsilon$

(d) $S \to BA$
$A \to BA|\varepsilon$
$B \to AAAB|\varepsilon$

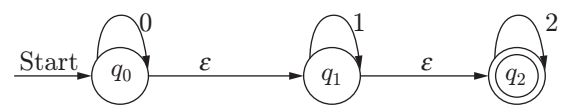
15. Consider the following context-free languages

$L = \{0^m 1^n | m \neq n;\ m, n \geq 0\}$

$L_1 = \{0^m 1^n | m > n \geq 0\}$ and $L_2 = \{0^m 1^n | n > m \geq 0\}$ then

(a) $L = L_1{}^* L_2$ (b) $L = \varnothing$
(c) $L = L_1 \cup L_2$ (d) $L = L_1 \cap L_2$

16. Consider the following NFA with ε moves



If the given NFA is converted to NFA without ε moves, which of the following denotes the set of final states?

(a) $\{q_2\}$ (b) $\{q_0, q_2\}$
(c) $\{q_0, q_1, q_2\}$ (d) $\{q_1, q_2\}$

17. Which of the following strings will not be accepted by the given NFA in the above question?

(a) 001122 (b) 1122
(c) 0121 (d) 22

18. If L and $\bar{L}$ are recursively enumerable then L is

(a) Regular
(b) Recursive enumerable
(c) Context-sensitive
(d) Recursive

19. Let L_1 be a recursive language. Let L_2 and L_3 be languages that are recursively enumerable but not recursive. Which of the following statements is not necessarily true?

(a) $L_2 - L_1$ is recursively enumerable
(b) $L_1 - L_3$ is recursively enumerable
(c) $L_2 \cap L_1$ is recursively enumerable
(d) $L_2 \cup L_1$ is recursively enumerable

20. Let w be any string of length n in $\{0, 1\}^*$. Let L be the set of all substrings of w. What is the minimum number of states in a non-deterministic finite automaton that accepts L?

(a) $n - 1$ (b) n
(c) $n + 1$ (d) 2^{n-1}

21. A minimum state deterministic finite automaton accepting the language $L = \{w \mid w \in \{0, 1\}^*$, number of 0's and 1's in w are divisible by 3 and 4, respectively$\}$ has

(a) 12 states (b) 11 states
(c) 10 states (d) 8 states

22. $A = \{<M>|$Turing machine that accepts only one string$\}$. The language A is

(a) context-free
(b) recursively enumerable
(c) non-recursively enumerable
(d) regular

23. A push-down automata recognizing $a^n b^n$ has minimum of n states where n is

(a) 3 (b) 0
(c) 1 (d) 2

24. Given the following statements:

S_1: Every context-sensitive language L is recursive.
S_2: There exists a recursive language that is not context-sensitive.

Which statement is correct?

(a) Only S_1 is correct.
(b) Only S_2 is correct.
(c) Both S_1 and S_2 are not correct.
(d) Both S_1 and S_2 are correct.

25. The Greibach normal form grammar for the language $L = \{a^n b^{n+1} \mid n \geq 0\}$ is

(a) $S \rightarrow aSB,\ B \rightarrow bSB|\varepsilon$
(b) $S \rightarrow aSB,\ B \rightarrow bSB|b$
(c) $S \rightarrow aSB|b,\ B \rightarrow b$
(d) $S \rightarrow aSb|b,\ B \rightarrow a$

26. Recursive enumerable languages are not closed under

(a) Union (b) Intersection
(c) Reversal (d) Complement

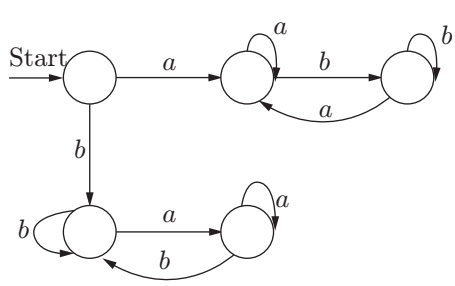
27. Consider the following statements:

I. Recursive languages are closed under complementation.
II. Recursively enumerable languages are closed under complementation.
III. Recursively enumerable languages are closed under union.

Which of the above statements are true?

(a) II only (b) I and III
(c) I and II (d) I, II and III

28. Consider the transition diagram of a DFA as given below



Which should be the final state(s) of the DFA if it should accept strings starting with "a" and ending with "b"?

(a) $\{q_0\}$ (b) $\{q_1\}$
(c) $\{q_0, q_1\}$ (d) $\{q_3\}$

29. Which of the following strings will be accepted in the above case if q_0 and q_1 are accepting states?

1. *ababab*
2. *babaaa*
3. *aaaba*

(a) 1, 2 (b) 2, 3
(c) 1, 3 (d) None of these

30. Which of the following represents the set of accepting states if the language to be accepted contains string having the same starting and ending symbols?

(a) $\{q_0\}$ (b) $\{q_0, q_3\}$
(c) $\{q_3\}$ (d) $\{q_3, q_1\}$

31. If L and ~L are recursive enumerable then L is

(a) regular and context-free
(b) only context-free
(c) regular and context-sensitive
(d) recursive

32. The following grammar

$G = (N, T, P, S)$
$N = \{S, A, B, C, D, E\}$
$T = (a, b, c\}$
P: $S \rightarrow ABCD,\ BCD \rightarrow DE,\ D \rightarrow aD,\ D \rightarrow a,\ E \rightarrow bE,\ E \rightarrow c$ is

(a) Type 1
(b) Type 1 but not Type 3
(c) Type 2 but not Type 3
(d) Type 0 but not Type 1

33. Context-free languages are closed under

(a) Concatenation and union
(b) Kleene closure and union
(c) Concatenation and union
(d) Concatenation, union, Kleene closure

34. The number of DFAs with the set of states $\{1,2, \ldots n\}$ $(n \geq 1)$ over the alphabet $\{0, 1\}$, with 1 as the initial state is

(a) 2^{n*2} (b) n^2
(c) $n^2 2^{n/2}$ (d) $n^{2n} 2^n$

35. The problem whether the given CFG is finite or infinite is

(a) Undecidable
(b) NP hard and decidable

(c) NP hard
(d) NP complete

36. Let $L = \{w{:}w$ has $3k + 1$ b's $\forall\ k \geq 0\}$and a minimized finite automata D accepting L is constructed. The number of states in D will be

(a) 2 (b) 3
(c) 5 (d) 0

37. Which of the following problems are decidable?

1. Does a given program ever produce an output?
2. If L is a context-free language, then is L' (complement of L) also context-free?
3. If L is a regular language, then is L' also regular?
4. If L is a recursive language, then is L' also recursive?

(a) 1, 2, 3, 4 (b) 1, 2,
(c) 2, 3, 4 (d) 3, 4

38. Number of states of the FSM required simulating behaviour of a computer with a memory capable of storing "m" words, each of length "n"

(a) $m \times 2^n$ (b) 2^{mn}
(c) 2^{m+n} (d) All of these

39. Which of the following statement is correct?

(a) $A = \{$If $a^n b^n |\ n = 0, 1, 2, 3, \ldots\}$ is a regular language.
(b) Set B of all strings of equal number of a's and b's denies a regular language.
(c) $L(A^*B^*) \cap B$ gives the set A.
(d) None of these

40. P, Q, R are three languages; if P and R are regular and if $PQ = R$, then

(a) Q has to be regular.
(b) Q cannot be regular.
(c) Q need not be regular.
(d) Q cannot be a CFL.

41. Which of the following definitions below generate the same language as L, where

$$L = \{x^n y^n \text{ such that } n \geq 1\}?$$

1. $E \rightarrow xEy|xy$
2. $xy|(x^+xyy^+)$
3. x^+y^+

(a) 1 only (b) 1 and 2
(c) 2 and 3 (d) 2 only

42. Which of the following statement is correct?

(a) All languages cannot be generated by CFG.
(b) Any regular language has an equivalent CFG.
(c) Some non-regular languages cannot be generated by CFG.
(d) Both (b) and (c).

43. What can be said about a regular language L over $\{a\}$ whose minimal finite state automation has two states?

(a) L must be $\{a^n|\ n$ is odd$\}$
(b) L must be $\{a^n|\ n$ is even$\}$
(c) L must be $\{a^n|\ n > 0\}$
(d) Either L must be $\{a^n|\ n$ is odd$\}$ or L must be $\{a^n|\ n$ is even$\}$

44. Let L be a language recognizable by a finite automation. The language REVERSE $(L) = \{w$ such that w is the reverse of v where $v \in L\}$ is a

(a) Regular language
(b) Context-free language
(c) Context-sensitive language
(d) Recursively enumerable language

45. Consider the following grammar

$S \rightarrow Ax|By$
$A \rightarrow By|Cw$
$B \rightarrow x|Bw$
$C \rightarrow v$

Which of the regular expression describes the same set of strings as the grammar?

(a) $xw^*y + xw^*yx + ywx$
(b) $xwy + xw^*xy + ywx$
(c) $xw^*y + xwxyx + ywx$
(d) $xwxy + xww^*y + ywx$

Set 2

1. Let R_1 and R_2 be regular sets defined over the alphabet, then

(a) $R1 \cap R_2$ is not regular
(b) $R_1 \cup R_2$ is not regular
(c) $\sum^* - R_1$ is regular
(d) $R_1{}^*$ is not regular

2. Which of the following definitions generates the same language as L?

Where $L = \{x^n y^n |\ n \geq 1\}$

(a) $E \rightarrow xEy|xy$ (b) $xy|x^+xyy^+$
(c) $|x^+y^+$ (d) None of these

3. Let $L \subseteq \sum^*$ where $\sum = \{a, b\}$, which of the following is true?

(a) $L = \{x|\ x$ has an equal number of a's and b's$\}$ is regular.

(b) $L = \{a^n\ b^n |\ n \geq 1\}$ is regular.
(c) $L = \{x |\ x$ has more a's than b's$\}$ is regular.
(d) $L = \{a^m\ b^n |\ m \geq n \geq 1\}$ is regular.

4. Which one of the following regular expression over $\{0, 1\}$ denotes the set of all strings not containing 100 as a substring?

(a) 0*(1 + 0)* (b) 0*101*0
(c) 0*1*01* (d) 0*(10 + 1)*

5. If the regular set A is represented by $A = (01 + 1)^*$ and the regular set B is represented by $B = ((01)^*1^*)^*$, which of the following is true?

(a) $A \subset B$
(b) $B \subset A$
(c) A and B are incomparable
(d) $A = B$

6. Let S and T be language over $\sum = \{a, b\}$ represented by the regular expressions $(a + b^*)^*$ and $(a + b)^*$, respectively, which of the following is true?

(a) $S \subset T$ (b) $T \subset S$
(c) $S = T$ (d) $S \cap T = \varnothing$

7. Consider the following statements:

S_1: $\{0^{2n} |\ n \geq 1\}$ is a regular language.
S_2: $\{0^m 1^n 0^{m+n} |\ m \geq 1$ and $n \geq 1\}$ is a regular language.

Which of the following is true about S_1 and S_2?

(a) Only S_1 is correct.
(b) Only S_2 is correct.
(c) Both S_1 and S_2 are correct.
(d) None of S_1 and S_2 is correct.

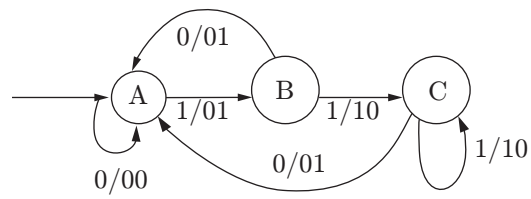
8. Give an arbitrary non-deterministic finite automation (NFA) with N states, the maximum number of states in an equivalent minimized DFA is

(a) N^2 (b) 2^N
(c) $2N$ (d) $N!$

9. Consider a DFA over $\sum = \{a, b\}$ accepting all strings which have number of a's divisible by 6 and number of b's divisible by 8. What is the minimum number of states that the DFA will have

(a) 8 (b) 14
(c) 15 (d) 48

10. The finite state machine described by the following state diagram with A as starting state, where an arc label is x/y and x stands for 1-bit input and y stands for 2-bit output



(a) Outputs the sum of the present and the previous bits of the input
(b) Outputs 01 whenever the input sequence contains 11
(c) Outputs 00 whenever the input sequence contains 10
(d) None of the above

11. The smallest DFA which accepts the language $L = \{x |$ length of x is divisible by $3\}$ has

(a) 2 states (b) 3 states
(c) 4 states (d) 5 states

12. A context-free grammar is ambiguous if

(a) The grammar contains useless on-terminals
(b) It produces more than one parse tree for some sentence
(c) Some production has two non-terminals side by side on the right-hand side
(d) None the above

13. Fortran is a

(a) regular language
(b) context-free language
(c) context-sensitive language
(d) none of these

14. Context-free language and regular languages are both closed under the operation(s) of

(a) union and intersection
(b) intersection and complement
(c) union and concatenation
(d) complementation and concatenation

15. Deterministic context-free languages are

(a) closed under union
(b) closed under complementation
(c) closed under intersection
(d) closed under concatenation

16. Consider the grammar with the following production

$S \rightarrow a\alpha b | b\alpha | ab$
$S \rightarrow \alpha S | b$
$S \rightarrow \alpha bb | ab$
$S\alpha \rightarrow bdb | b$

The above grammar is

(a) context-free grammar
(b) regular grammar
(c) context-sensitive grammar
(d) LR(k)

17. If L_1 and L_2 are context-free languages and R a regular set, one of the languages below is not necessarily a context-free language. Which one?

(a) L_1L_2 (b) $L_1 \cap L_2$
(c) $L_1 \cap R$ (d) $L_1 \cup L_2$

18. Which of the following statement is false?

(a) Every finite subset of a non-regular set is regular.
(b) Every subset of a regular set is regular.
(c) Every finite subset of a regular set is regular.
(d) The intersection of two regular sets is regular.

19. If two finite state machines are equivalent, they should have the same number of

(a) states (b) edges
(c) states and edges (d) none of these

20. Context-free languages are closed under

(a) union, intersection
(b) union, Kleene closure
(c) intersection, complement
(d) eomplement, Kleene closure

21. Let L_D be the set of all languages accepted by a PDA by final state and L_E the set of all languages accepted by empty stack. Which of the following is true?

(a) $L_D = L_E$ (b) $L_D \supset L_E$
(c) $L_D \subset L_E$ (d) None of these

22. If L_1 is a context-free language and L_2 is regular, which of the following is false?

(a) $L_1 - L_2$ is not context-free.
(b) $L_1 \cap L_2$ is context-free.
(c) ~L_1 is not context-free.
(d) ~L_2 is regular.

23. The language accepted by pushdown automation in which the stack is limited to 10 items is best described as

(a) context-free
(b) regular
(c) deterministic context-free
(d) recursive

24. Recursive languages are

(a) a proper superset of context-free languages
(b) also called 0 languages
(c) recognizable by Turing machines
(d) all of the above

25. The productions

$E \rightarrow E + E$
$E \rightarrow E - E$
$E \rightarrow E^*E$
$E \rightarrow E/E$
$E \rightarrow id$

(a) generate an inherently ambiguous language
(b) generate an ambiguous language but not inherently so
(c) are unambiguous
(d) can generate all possible fixed length valid computation for carrying out addition, subtraction, multiplication and division, which can be expressed in one expression

26. In which of the cases stated below is the following statement true?

"For every non-deterministic machine M_1 there exists an equivalent deterministic machine M_2 recognizing the same language"

I. M_1 is non-deterministic finite automation
II. M_1 is a non-deterministic PDA
III. M_1 is a deterministic Turing machine
IV. there is no machine M_1 for which the given statement is true

(a) I and II (b) I and III
(c) I, II and III (d) II and IV

27. Which of the following conversion is not possible (algorithmically)?

(a) Regular grammar to context-free grammar
(b) Non-deterministic FSA to deterministic FSA
(c) Non-deterministic PDA to deterministic PDA
(d) Non-deterministic Turing machine to deterministic Turing machine

28. Which one of the following is not decidable?

(a) Give a Turing machine M, a string s and an integer k, M accepts s within k steps
(b) Equivalence of two Turing machines
(c) Languages accepted by a given finite state machine is non-empty
(d) Languages accepted by a context-free grammar is non-empty

29. Which of the following is true?

(a) The complement of a recursive language is recursive.

(b) The complement of a recursively enumerable language is recursively enumerable.
(c) The complement of a recursive language is either recursive or recursively enumerable.
(d) The complement of a context-free language is context-free.

30. The C language is

(a) A context-free language
(b) A context-sensitive language
(c) A regular language
(d) Parsable fully only by a Turing machine

31. Any string of terminals that can be generated by the following CFG is

$S \rightarrow XY$

$X \rightarrow aX|bX|a$

$Y \rightarrow Ya|Yb|a$

(a) Has at least one "b"
(b) Should end in "a"
(c) Has no consecutive a's or b's
(d) Has at least two a's

32. Which of the following problems are undecidable?

(I) Membership problem in context-free languages
(II) Whether a given context-free language is regular
(III) Whether a finite state automation halts on all inputs
(IV) Membership problem for type 0 languages

(a) II and IV (b) I and II
(c) III and IV (d) I and IV

33. For two regular languages

$$L_1 = (a + b)^*a \text{ and } L_2 = b(a + b)^*$$

The intersection of L_1 and L_2 is given by

(a) $(a + b)^*ab$ (b) $ab(a + b)^*$
(c) $a(a + b)^*b$ (d) $b(a + b)^*a$

34. Consider the following problem X.

"Given a Turing machine M over the input alphabet $\sum$, any state q of M and a word $\sum^*$, does the computation of M on w visit the state q"

Which of the following statements about X is correct?

(a) X is decidable.
(b) X is undecidable but partially decidable.
(c) X is undecidable and not even partially decidable.
(d) X is not a decision problem.

ANSWERS TO PRACTICE EXERCISES

Set 1

1. (d)	**9.** (c)	**17.** (c)	**25.** (c)	**33.** (d)	**41.** (a)
2. (d)	**10.** (d)	**18.** (d)	**26.** (c)	**34.** (d)	**42.** (d)
3. (b)	**11.** (c)	**19.** (b)	**27.** (b)	**35.** (a)	**43.** (b)
4. (c)	**12.** (b)	**20.** (c)	**28.** (b)	**36.** (b)	**44.** (a)
5. (b)	**13.** (c)	**21.** (a)	**29.** (c)	**37.** (d)	**45.** (a)
6. (a)	**14.** (c)	**22.** (b)	**30.** (b)	**38.** (b)	
7. (b)	**15.** (c)	**23.** (a)	**31.** (d)	**39.** (c)	
8. (b)	**16.** (c)	**24.** (d)	**32.** (d)	**40.** (c)	

Set 2

1. (c) Regular languages are closed under union, interaction, Kleene closure and complementation. So option (c) is correct and remaining options (a), (b) and (d) are false.

2. (a) $L = \{x^n y^n | n \geq 1\}$

The given language L produces all strings having equal number of x and y, where y follows symbol x. Only expression $\Rightarrow E \rightarrow Ey \,|\, xy$ can produce equal number of x and y, where y follows x.

3. (d) Regular language is accepted by DFA. DFA does not have memory, so it is not able to do comparison between two values, that they are equal or not (means equal number of a's and b's). So only option (d) does not have comparison, which means it can be accepted by DFA.

4. (a)

a. 0*(1 + 0)* can generate string 0000100, which contains substring 100.
b. 0*101*0 generates strings which does not contain 100 as substring.
c. 0*1*01* generates strings which does not contain 100 as substring.
d. 0*(10+1)* generates all strings which does not contain 100 as substring.

5. (d) Strings in language generated by both the regular expression is equivalent.

So, $A = B$

6. (c) Language generated by both the regular expression is equivalent, so $(a + b^*)^* = (a + b)^*$.

7. (a) $L_1 = \{0^{2n} | \ n \geq 1\}$, L_1 is regular language because it produces even number of 0's. $L_2 = \{0^m 1^n 0^{m+n} | \ m \geq 1 \text{ and } n \geq 1\}$, L_2 is context-free language because there are two comparisons. So, S_1 is correct but S_2 is not correct.

8. (b) NFA with N states, an equivalent minimized DFA having maximum 2^N states.

9. (d) A DFA over $\sum = \{a, b\}$ accepting all strings which have number of a's divisible by m and number of b's divisible by n have m^*n states. In our example we have $6 \times 8 = 48$ states.

10. (a) FSM outputs the sum of the present and previous bits of the input. Let us suppose input is 101, then output is 0 + 1 (last 2 bits) = 01. When input is 11 then output is 1 + 1 = 10.

11. (b) The minimal DFA which accepts the language $L = \{x | \text{ length of } x \text{ is divisible by } n\}$ always have n states.

12. (b) A context-free grammar is ambiguous if there is a word which has two different derivation trees.

13. (c) Fortran is a context-sensitive languages.

14. (c) Regular languages are closed under the operation of union, interaction, reversal, concatenation and complementation. Context-free languages are closed under the operation of concatenation and union.

15. (b) Deterministic context-free languages are closed under complementation.

16. (c)

$S \rightarrow a\alpha b | b\alpha | ab$
$S \rightarrow \alpha S | b$
$S \rightarrow \alpha bb | ab$
$S\alpha \rightarrow bdb | b$

The given production ($S\alpha \rightarrow bdb|b$) contains terminal on the left-hand side, which means this grammar cannot be context-free grammar and regular. So, this grammar is context-sensitive.

17. (b) If L_1 and L_2 are context-free language then they are closed under

(i) Concatenation $L_1 L_2$
(ii) Union $L_1 \cup L_2$
(iii) Under regular intersection $L_1 \cap R$

But they are not closed under intersection $L_1 \cap L_2$.

18. (b) For (a) and (c), if a set is finite then it means we can create DFA for that. It does not matter that it belongs to non-regular or regular. If we are able to draw DFA for that set then it means that it is regular. So every finite set of regular or non-regular is regular.

For option (d), regular language is closed under intersection.

Let $\sum = (a, b)$, $\sum^*$ is a regular set.

$L_1 = \{a^n b^n | \ n \geq 0\}$

$L_2 = \{a^n b^n | \ nm \geq 0\}$

Here, $L_1 \subset \sum^*$ and $L_2 \subset \sum^*$

But L_1 is CFL and L_2 is regular. So from the above explanation, every subset of regular set need not be regular.

19. (d) Two finite state machines are said to be equivalent if they can accept the same language.

20. (b) Context-free languages are closed under union and Kleene closure and not closed under intersection and complement.

21. (a) PDA accepts its input by two ways:
(a) Accepting by entering into final state
(b) Accepting by empty stack

The acceptance or recognition power of PDA of both approaches are same because there exists algorithm to convert PDA accepting by empty stack to PDA accepting by final state and vice versa, Therefore $L_D = L_E$.

22. (a) L_1 = Context-free language (CFL)

L_2 = Regular
$L_1 - L_2 = L_1 \cap L_2^C$

L_2^C is a regular language because regular is closed under complement and so $L_1 \cap L_2^C$ is context-free because languages are closed under regular intersection. CFL is closed under regular intersection, so $L_1 \cap L_2$ is a context-free language. CFL is not closed under complementation, so $\sim L_1$ is not a context-free language. Regular languages are closed under complementation, so $\sim L_1$ is regular.

23. (b) A PDA with a stack limited to containing limited items is equivalent to a DFA, and language accepted by DFA is regular. So, regular is the best word to describe it.

24. (d) Recursive language is a superset of CFL. Recursive language is called type 0 or unrestricted language and accepted by Turing machine.

25. (b) Given grammar will generate ambiguous grammar because for a single string there can be two parse tree according to a given grammar.

26. (b)

(a) Power of NFA and DFA is same because there exists an algorithm to convert every NFA to DFA.
(b) Recognition power of NPDA and DPDA are not same because there is no algorithm to convert NPDA to DPDA.
(c) Power of NTM and DTM is same because there exists an algorithm to convert every NTM to DTM.
(d) Since for finite automaton and Turing machines the above statements are true, so this option is false.

27. (c) Since recognition power of NPDA (non-deterministic PDA) and DPDA (deterministic PDA) are not same because there is no algorithm to convert NPDA to DPDA. Means there exists some languages such as $L = \{ww^R | w \in (a, b^*\}$ which are accepted by NPDA but not by DPDA.

28. (b)

(a) Give a Turing machine M, a string s and an integer k, M accepts s within k steps is a decidable problem.
(b) Equivalence of two Turing machines is an undecidable problem.
(c) Emptiness of regular language is a decidable problem.
(d) Emptiness of CFL's is a decidable problem.

29. (a) Recursive languages are closed under complementation so complement of a recursive language is also recursive. As every recursive language is also recursive enumerable. So the complement of recursive languages is also recursive enumerable. So both options (a) and (c) are true but (a) is the strongest answer.

30. (b) The C language is a context-sensitive language.

31. (d) Has at least 2 numbers of a's

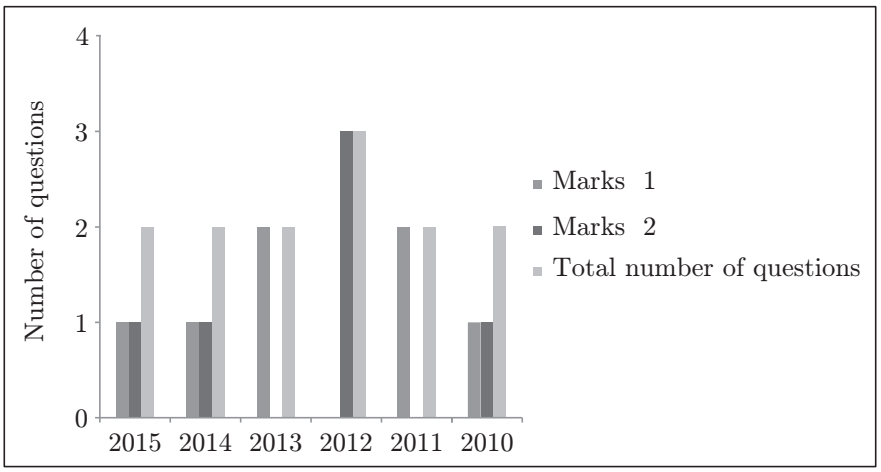
32. (a) Only options (II) and (IV) are undecidable because there exists no algorithm to check that a CFL is regular and also for membership problem for RE language.

33. (d) Language $L_1 = \{(a + b)^*a\}$ is a set of strings possible with "a" and "b" and which ends with terminal "a".
Language $L_2 = \{b(a + b)^*\}$ is a set of all strings possible with "a" and "b" and which starts with terminal "b".
So their intersection will give set of strings which starts with "a" and ends with "b", that is, $b(a + b)^*a$.

34. (b) There is algorithm to reduce halting problem of Turing machine to state entry problem. The given problem X is a state-entry problem. As halting problem of Turing machine is undecidable, so problem X also must be undecidable. The language corresponding to state-entry problem is RE but not REC. So X is undecidable, but partially decidable.

UNIT VI: COMPILER DESIGN

LAST SIX YEAR'S GATE ANALYSIS



Concepts on which questions were asked in the previous six years

Year	Concepts
2015	Control flow graph, Compiler phases, Parsers, LL(1) and LR(1) Grammar, Shift reducing parser
2014	Parsing, Static and dynamic memory allocation
2013	Parser, Languages
2012	Parsing, Grammar, Decidability
2011	Decidability, Languages
2010	Grammar, Compiler keywords

CHAPTER 6

COMPILER DESIGN

Syllabus: Compiler design: Lexical analysis, Parsing, Syntax-directed translation, Runtime environments, Intermediate and target code generation, Basics of code optimization.

6.1 INTRODUCTION

Computer understands programs written in machine language. Building program in machine language is a tedious and an error-prone task for human. So the programs are written in high-level languages which are easily understood by human. A compiler is a program that converts high-level program into low-level machine language that is understood by machine. This subject deals with how a compiler is designed and organized. While writing a compiler, the compilation process is divided into various phases. These phases operate in sequence; each phase takes input from the previous phase and provides output to the next phase.

6.2 COMPILERS AND INTERPRETERS

6.2.1 Compiler

A compiler is a special program that reads statements written in a language (called source language) and then converts them into another language (called target language). In other words, a compiler is a program which translates statements written in high-level language (i.e. Java, C#, Visual Basic) into machine-level language (Fig. 6.1). In the process of translation, a compiler also checks for errors if any.

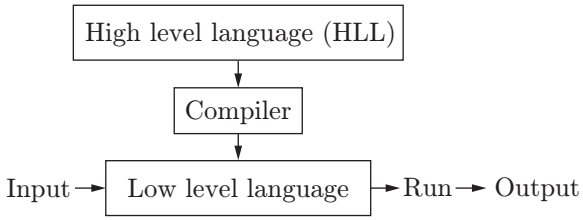


Figure 6.1 | Compiler.

6.2.2 Interpreter

An interpreter takes a single instruction as input and converts it into machine-level language and shows errors in the statement if any (Fig. 6.2). It requires less memory than a compiler, and execution of conditional control statements are slower. Debugging is easier in an interpreter.

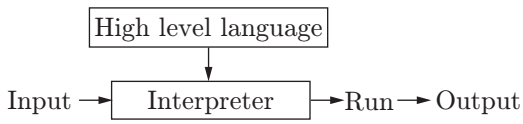


Figure 6.2 | Interpreter.

6.2.3 Phases of a Compiler

A compiler takes a source program written in high-level language as input and produces an equivalent set of machine instruction as output. The compiler process is complex, so it is divided into six sub-processes which are also known as phases of a compiler. The following are different phases of a compiler (Fig. 6.3):

1. Lexical analyzer
2. Syntax analyzer
3. Semantic analyzer
4. Intermediate code generator
5. Code optimization
6. Target code

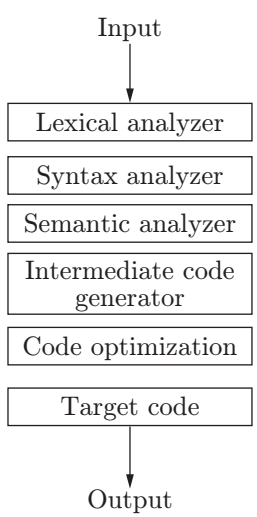


Figure 6.3 | Phases of a compiler.

6.2.3.1 Lexical Analyzer

Lexical analyzer reads the source program character by character at a time and unites them into a stream of tokens. Token is a group of character which represents keywords, operators and identifiers. Character sequence formed by tokens is called "lexeme". Lexical analyzer is also known as lexer, tokenizer or scanner. If a lexical analyzer gets an invalid token, then it will generate an error. The lexical analyzer assigns an id to each token according to their occurrence. It makes entries of each identifier into the symbol table. As a lexical analyzer cannot enter all information regarding an identifier such as type and scope, the remaining information is inserted by the other phases of the compiler into the symbol table. For detail explanation about lexical analyzer, refer to Section 6.3.

1. **Symbol table:** A symbol table is a data structure which stores identifier information related to its declaration and appearance in a program such as identifier id, type, scope, value and sometimes location. Link list and hashing are the common techniques which are used to construct symbol tables.

Example 6.1

Statement written in a source program is

$$A = B + C*25;$$

Tokens generated by lexical analyzer

$A \rightarrow$ Identifier
$= \rightarrow$ Assignment operator
$B \rightarrow$ Identifier
$+ \rightarrow$ Add operator
$C \rightarrow$ Identifier
$* \rightarrow$ Multiplication operator
$25 \rightarrow$ Constant

6.2.3.2 Syntax Analyzer or Parser

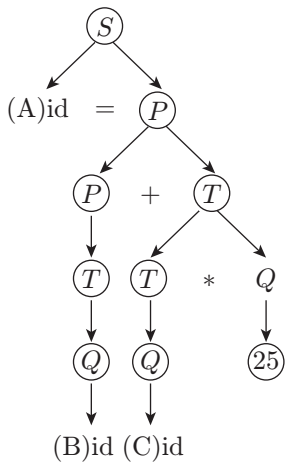
A syntax analyzer is the second and important phase of a compiler. It receives the tokens from the output of a lexical analyzer as an input. Parser performs two functions:

1. A parser checks that the input tokens from a lexical analyzer are valid or not according to the specified grammar of source language.
2. It generates a parse tree according to the given grammar to the source language. The grammar of source language is given below:

$S \rightarrow \text{id} = P$
$P \rightarrow P + T/T$
$T \rightarrow T*Q/Q$
$Q \rightarrow$ id/Integer constant

Example 6.2

Parse tree of a given problem $A = B + C*25$.



Parse tree of $A = B + C*25$

6.2.3.3 Semantic Analyzer

When the parse tree is generated by a syntax analyzer and passed as an input to the semantic analyzer, then the semantic analyzer computes the additional information related to the recognized tokens, such as operator, operand, expression or statement, and inserts that information into the symbol table. The information stored in the symbol table is frequently used by the other phases of the compiler. During the semantic analysis, the type of identifier is checked. In our example, let all identifiers be float, and 25 be treated as an integer constant. If required, the semantic analyzer will perform an implicit-type conversion, and if it is not possible, then it will throw an error. This can be easily understood by the example given in Fig. 6.4.

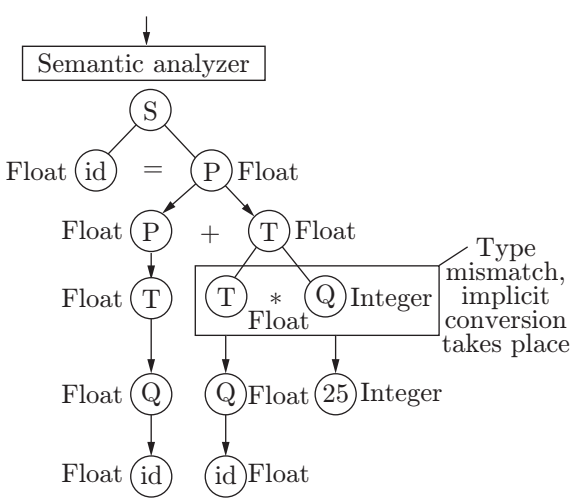


Figure 6.4 Semantic analyzer of $A = B + C*25$.

So, here in our example, implicit conversion took place. Implicit-type casting is also known as coercion.

6.2.3.4 Intermediate Code Generator

The intermediate code generator phase takes a tree as an input produced by a semantic analyzer and produces an intermediate code. The intermediate code thus generated has mainly two properties: it should be easy to produce and easy to translate into target program. An intermediate code can be represented in variety of forms. One of the forms is the three-address form, which is very similar to the assembly language in which every memory location acts like a register. The intermediate code of our example is

Source code:

$$A = B + C * 25$$

Intermediate code:

$$T_1 = C * 25$$
$$T_2 = B + T_1$$
$$A = T_2$$

6.2.3.5 Code Optimization

The code optimization phase is to reduce the size of the code and improve the performance of the code generated by an intermediate code phase. The most important part of optimized code is to minimize the amount of time taken by the code to execute and less common is to minimize the amount of memory used by the code. Optimized code of our example is

Intermediate code:	Optimized code:
$T_1 = C * 25$	$T_1 = C * 25$
$T_2 = B + T_1$	$A = B + T_1$
$A = T_2$	

6.2.3.6 Target Code

The target code is the final phase of the compiler which normally converts the input obtained from the code optimization phase into the target code (machine code or assembly code). The target code of our example is as follows:

MOV	R_1,	C
MUL	R_1,	25
MOV	R_2,	B
ADD	R_2,	R_1
STORE	A,	R_2

This is the final output of the compiler.

6.2.4 Grouping of Phases

Phases deal with the logical organization of a compiler. In an implementation, activities from more than one phase are often grouped together. Basically, phases are grouped into two parts. The first part is known as the front end, which consists of initial four phases (lexical analyzer, syntax analyzer, semantic analyzer and intermediate code) along with symbol table operations and error handling. The second part, also known as back end, consists of the last two phases (code optimization and target code). The back end also includes error handling and symbol table operation (Fig. 6.5).

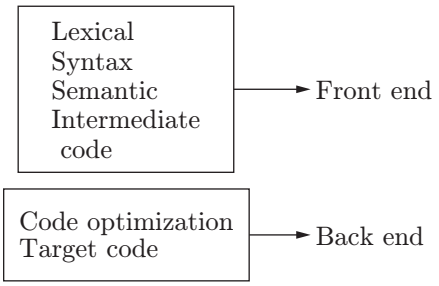


Figure 6.5 | Grouping of compiler phases.

6.2.4.1 Compiler Construction Tools

The compiler writers use software tools such as debuggers, version managers, profilers and so on. The following is a list of some useful compiler construction tools:

1. **Parser generators:** These produce syntax analyzer from context-free grammar as input.
2. **Scanner generators:** These automatically produce lexical analyzer from a specification based on regular expressions.
3. **Syntax-directed translation engines:** These produce collection of routines from parse tree, generating the intermediate code.
4. **Automatic code generators:** These take collection of rules that define the translation of each operation of the intermediate language into machine language for the target machine.
5. **Data-flow engines:** Data flow analysis is required to perform good code optimization and data flow engines facilitates the gathering of information about how values are transmitted from one part of a program to another part of that program.

6.3 LEXICAL ANALYZER

Lexical analyzer is the starting phase of a compiler. It reads the source program character by character at a time and unites them into a stream of tokens. Tokens consist of identifiers, operators and operand. A lexical analyzer also stores the name and id of identifiers in the symbol table (Fig. 6.6).

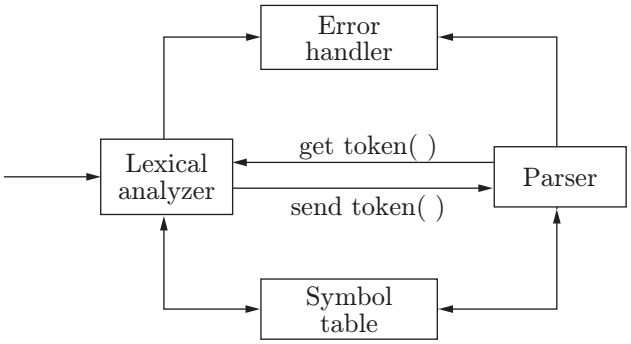


Figure 6.6 | Lexical analyzer.

6.3.1 Functions of a Lexical Analyzer

A lexical analyzer has the following functions.

1. Lexical analyzer divides the given source code or program into some meaning full words called tokens.
2. It eliminates the comment lines.
3. It finds integer and floating point constant.
4. It eliminates white-space character such as blank space and tab.
5. It helps in giving error message by providing row and column numbers.
6. It identifies identifier, keywords, operators and constants.

6.3.2 Implementation of a Lexical Analyzer

Method of implementing a lexical analyzer or scanner is regular expression and finite automaton. Some background information related to regular expression and finite automaton are given in the following sections which will help to understand how a scanner works.

6.3.2.1 Regular Expression Review

1. **Symbol:** Letters, digits and special symbols are examples of a symbol.
2. **Alphabet:** A finite set of symbols through which we build large structures. An alphabet is denoted by Σ, for example, $\Sigma = \{0, 1\}$.
3. **String:** A finite set of symbol made up of alphabets, for example, a, b are alphabets and $aaab$, $abba$ are the strings.
4. **Empty string:** A string which has zero symbols, and represented by ε.

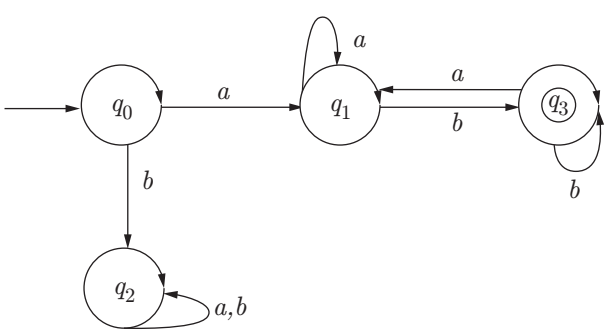
5. **Formal language:** A set of all possible strings which can be generated from given alphabets, and represented by Σ^*.
6. **Regular expression:** The rules that define the set of words that are valid tokens in a formal language. These rules are made by three operators:
   - Alternation $x|y$ (x or y)
   - Repetition x^* (x is repeated 0 or more times)
   - Concatenation xy

6.3.2.2 Finite Automata Review

Once we have all type of tokens defined by regular expression, we can create a finite automaton for recognizing them. A finite automaton has the following:

1. A finite set of states, one of which is the start state or initial state, and some (maybe none) of which are final states.
2. An alphabet Σ of possible input symbols.
3. A finite set of transitions that specifies for each state and for each symbol of the input alphabet, which defines that for an input symbol which will be the next state to go.

Example 6.3



where q_0 is the initial state and q_3 is the final state. The given finite automaton accepts a language which has a string starting with 'a' and ending with 'b'.

6.3.2.3 Recognition of Tokens

In this section, we will explain how a token is recognized by a lexical analyzer. A lexical analyzer uses finite automaton to recognize a token. Transition diagram is shown below which consists of stages and arcs. Arcs show the transition from one state to another state. Transition diagram describes the working of a lexical analyzer when called by the parser to get query for the next token. The positions are shown by the circles called states which are connected by edges. Here is a finite automaton which recognizes an integer (Fig. 6.7).

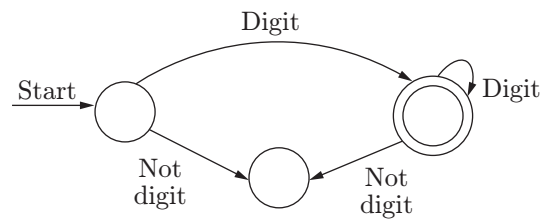


Figure 6.7 Finite automata for recognizing an integer.

Problem 6.1: Consider the following C program, find the number of tokens.

```
float average(int a, int b)
{
float c;
c = (a + b)/2;
return c;
}
```

Solution: Every token individual has been underlined. Counting the number of underlines, we have the number of tokens as 27.

```
float average (int a , int b )
{
float c ;
c = ( a + b ) / 2 ;
return c ;
}
```

Problem 6.2: Find the number of tokens in the following C statement.

```
printf("k = %d", i);
```

Solution: The number of tokens is 7.

```
printf("k = %d", i);
```

6.4 PARSER

A parser is a part of a compiler. It takes sequence of tokens from the lexical analyzer as an input and then builds a data structure in the form of a parse tree (Fig. 6.8). A parser's main purpose is to determine if

the input data may be derived from the start symbol of the grammar. Depending upon the method how the parse tree is derived, we have two types of parsers–top-down parser and bottom-up parser (discussed shortly).

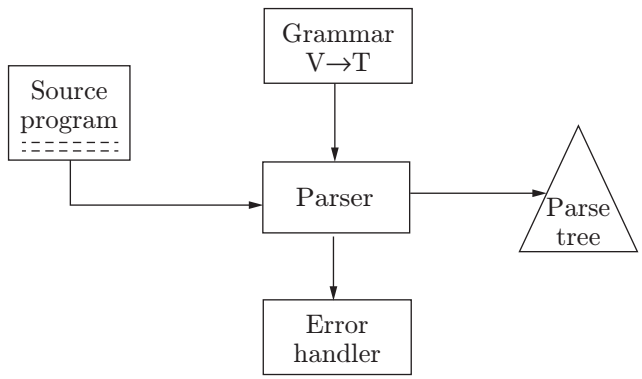


Figure 6.8 | Parser.

6.4.1 Context-Free Grammar

A grammar $G = (V_n, T, S, P)$ is said to be a context-free grammar (CFG) if the production $P = \{u \to v\}$ of G are of the form $u \to v$ and satisfy the following conditions:

1. $u \to v$, where $v \in (V \cup T)^*$, and V stands for variable and T for terminal
2. $u \to v$, where $u \in V_n$
3. $|u| \le |v|$ (length of u is less than v)
4. Only single variable is allowed in left side (means u has single variable only)

As we know that a CFG has no context either left or right, this is the reason why it is also known as context-free.

Problem 6.3: Consider a grammar $G = (V_n, T, S, P)$ having production $S \to aSa|bSb|x$. Check the production and find the language generated.

Solution:

Let P_1: $S \to aSa$

P_2: $S \to bSb$

P_3: $S \to x$

(a, b, x) are terminals and S is a variable. As all the production are of the form $A \to \alpha$, where $\alpha \in (V_n \cup S)^*$ and $A \in V_n$, hence G is a CFG. And it will produce context-free language.

Language generated: $L(G) = \{wxw^R : w \in (a+b)^*\}$

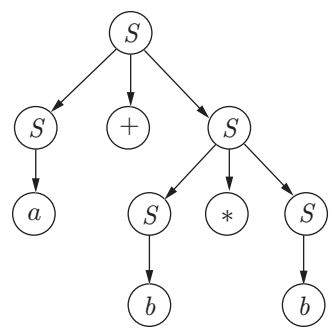
6.4.2 Derivation Tree or Parse Tree

The string generated by a CFG $G = (V_n, T, S, P)$ is represented by a hierarchical structure called tree. A derivation tree or parse tree for a CFG is a tree that satisfies the following condition:

1. If $A \to \alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n$ is a production in G, then A becomes the father of nodes, labelled $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n$.
2. The root has label S (starting symbol).
3. Every vertex (or node) has a label.
4. Internal nodes should be labels with variables only.
5. The leaves nodes are labelled with ε or terminal symbol.
6. The collection of leaves from left to right yields the string w.

Problem 6.4: Consider the grammar $S \to S + S|S*S|a|b$. Construct a derivation (or parse) tree for the string $w = a + b*b$.

Solution:



6.4.2.1 Leftmost Derivation Tree

A derivation tree is called a leftmost derivation (LMD) tree if the ordering of decomposed variable is from left to right. Thus, for generating string $w = aab$ from grammar:

$S \to AB$ (production 1)

$A \to aaA$ (production 2)

$A \to \varepsilon$ (production 3)

$B \to bB$ (production 4)

$B \to \varepsilon$ (production 5)

LMD:

$S \to \underline{A}B$ (by production 1)

$S \to aa\underline{A}B$ (by production 2)

$S \to aa\underline{B}$ (by production 3)

$S \to aab\underline{B}$ (by production 4)

$S \to aab$ (by production 5)

6.4.2.2 Rightmost Derivation Tree

A derivation tree is called rightmost derivation tree (RMD) if the ordering of decomposed variable is from right to left. Thus, for generating string $w = aab$ from the above grammar:

RMD:

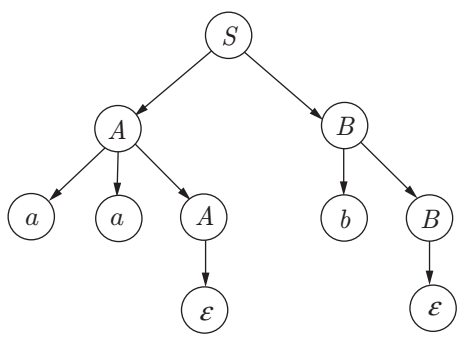
$S \to AB$ (by production 1)

$S \to AbB$ (by production 4)

$S \to Ab$ (by production 5)

$S \to aaAb$ (by production 2)

$S \to aab$ (by production 3)

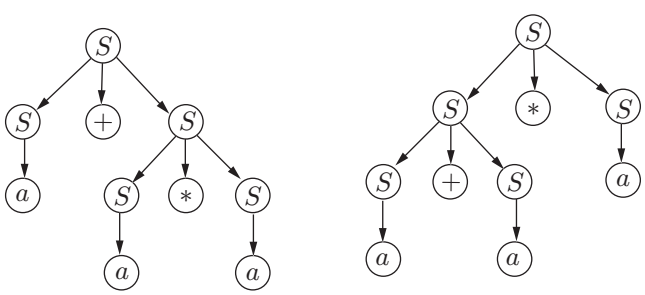


Left to right

6.4.3 Ambiguous Grammar

A grammar G is called ambiguous if for some string $w \in L(G)$, there exist two or more derivation tree (two or more LMD or two or more RMD tree). Let us consider a CFG grammar having production:

$S \to S + S|S^*S|a|b$, for string $w = a + a^*a$ have more than one LMD tree.



Note: A language (L) is called ambiguous if and only if every grammar which generates it is ambiguous. The only known ambiguous language is $\{a^n b^m c^n\} \cup \{a^n b^m c^m\}$.

Left recursion and left factoring is the major cause for a grammar to be ambiguous. But presence of these in a grammar does not mean that grammar is ambiguous; and similarly absence of these does not mean that the grammar is unambiguous.

6.4.3.1 Removal of Ambiguity

1. **Removal of left recursion:** A production of grammar $G = (V_n, T, S, P)$ is said to be left recursive grammar if it has one of the productions in the given form:

   $A \to A\alpha$, where A is a variable and $\alpha \in (V_n \cup S)^*$

   Elimination of left recursion: Let the variable A have left recursive problem as following:

   $$A \to A\alpha_1|A\alpha_2|A\alpha_3|\ldots|A\alpha_n|\beta_1|\beta_2|\beta_3|\ldots|\beta_m$$

   where $\beta_1, \beta_2, \ldots, \beta_m$ do not begin with A. Then we replace A production by:

   $$\{A \to \beta_1 A'|\beta_2 A'|\beta_3 A'|\ldots|\beta_n A'|\}$$

   where $A' \to \alpha_1 A'|\alpha_2 A'|\alpha_3 A'|\ldots|\alpha_n A'|\varepsilon$

Example 6.4

Let grammar $S \to S + S|S^*S|a|b|c$

To eliminate left recursion, the grammar S is replaced by

$$S' \to +SS'|^*SS'|\varepsilon$$
$$S \to aS'|bS'|cS'$$

2. **Removal of left factoring:** In grammar G, two or more productions of variable A are said to have left factoring if the productions are in the form:

   $$A \to \alpha\beta_1|\alpha\beta_2|\alpha\beta_3\ldots|\alpha\beta_m$$

   where $\{\beta_1|\beta_2|\beta_3\ldots|\beta_m\} \in (V_n \cup S)^*$ and does not start with α. All these production have common left factor α.

   Elimination of left factoring: Let variable A have (left factoring) production as follows:

   $$A \to \alpha\beta_1|\alpha\beta_2|\alpha\beta_3\ldots|\alpha\beta_m|\gamma_1|\gamma_2|\gamma_3|\ldots|\gamma_m$$

   where $\gamma_1, \gamma_2, \gamma_3, \ldots, \gamma_m$ and $\{\beta_1, \beta_2, \beta_3\ldots\beta_m\}$ do not contain α as a prefix, then we replace this production by

   $$A \to \alpha A'|\gamma_1|\gamma_2|\gamma_3|\ldots|\gamma_m$$
   $$A' \to \beta_1|\beta_2|\beta_3\ldots|\beta_m$$

Example 6.5

Let grammar $A \to abc|abd|abe$

To remove left factoring, we have

$$A \to abA'$$
$$A' \to c|d|e$$

6.4.4 Top-Down Parser

In top-down parser, parse tree construction starts from the root and proceeds to the leaf. Top-down parser uses

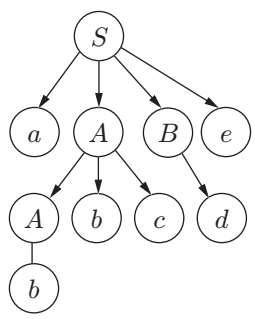
LMD for constructing the parse tree. When a variable contains more than one choice, choosing the correct production is always going to be difficult. In top-down parsing, no left recursion and no left factoring exist.

Problem 6.5: String $w = abbcde$, grammar G is given below:

$$S \rightarrow aABe$$
$$A \rightarrow Abc/b$$
$$B \rightarrow D$$

Draw a parse tree for the production of a string w with the help of grammar G.

Solution:



Parse tree for string $w = abbcde$.

Top-down parsing can be performed by the following two methods:

1. Top-down parsing with backtracking
2. Top-down parsing without backtracking

6.4.4.1 Top-Down Parsing with Backtracking

One of the most straightforward forms of parsing is recursive descent parsing (RDP). This is a top-down process in which the parser attempts to verify that the syntax of the input stream is correct as it is read from left to right. The pseudocode of recursive descent parser is given as follows:

```
RDP(S)
{
Choose a production S→x1,x2,x3,x4
for(i=1 to n)
{
if(xi is variable)
RDP(i)
else if(xi==lookahead)
increment i/p pointer
else
error(choose another production)
}
}
```

Problem with Recursive Descent Parser

1. Recursion is used to generate parse tree.
2. More time is wasted in backtracking.
3. Recursive descent parser can enter into an infinite loop if the given grammar contains left recursion.
4. Time complexity of RDP is $O(2^n)$.

6.4.4.2 Top-Down Parser without Backtracking

A predictive parser is a special class of recursive decent parser. The goal of predictive parsing is to construct a top-down parser that does not require backtracking. Predictive parsing technique can only be used for class of LL(k) grammars, where k is some integer. In LL(k) grammar, by seeing k tokens a recursive decent parser decides which production should be examined so that the LL(k) grammars are able to exclude all grammars that are ambiguous and having left recursion (Fig. 6.9).

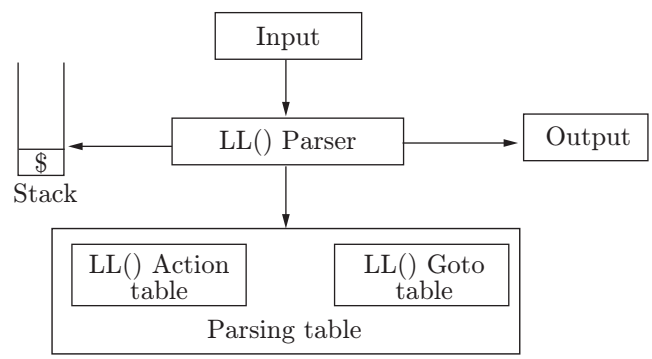


Figure 6.9 | Predictive parser.

6.4.4.3 Algorithm of Predictive Parser

Let x be the top of stack and 'a' be the look-ahead symbol. Then

1. If ($x == a == \$$), then successful complete.
2. If ($x == a == \$$), then pop the stack element and increment input pointer.
3. If (x is a variable), then see the LL(1) action parsing table M.
   If $M[x_1, a] = x \rightarrow uvw$, then replace x by uvw in the reverse order.
4. If $M[x_1, a]$ = blank, then parsing error.

Problem 6.6: Draw a top-down predictive parser using the following parser table.

Grammar of given language

$$S \rightarrow (L)/a$$
$$L \rightarrow SL'$$
$$L' \rightarrow \varepsilon/,SL'$$

String $w = (a, a, a)$

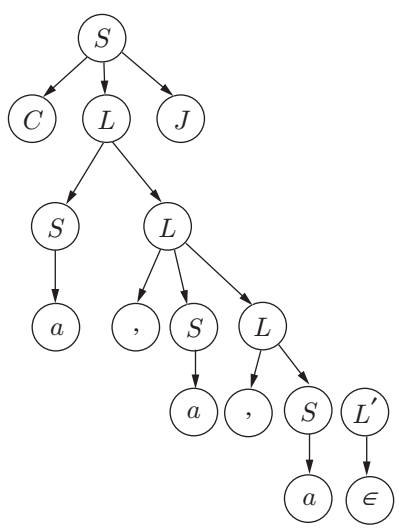
Parsing Table

	A	,	(	)	$
S	$S \to a$		S → (L)		
L	$L \to SL'$		$L \to SL'$		
L'		$L' \to, SL'$		$L' \to \varepsilon$	

Solution:

S. No.	Input	Stack	Production
1.	$(\underline{a}, a, a)\$$	$\$S$	$S \to (L)$
		$\$)L\underline{(}$	
2.	$\underline{a}, a, a)\$$	$\$)L$	$L \to SL'$
		$\$)L'S$	$S \to a$
		$\$)L'a$	
3.	$\underline{,}\ a, a)\$$	$\$)L'$	$L' \to, SL'$
		$\$)L'S\underline{,}$	
4.	$\underline{a}, a)\$$	$\$)L'S$	$S \to a$
		$\$)L'\underline{a}$	
5.	$\underline{,}\ a)\$$	$\$)L'S\underline{,}$	$L' \to,SL'$
6.	$\underline{a})\$$	$\$)L'S$	$S \to a$
		$\$)L'\underline{a}$	
7.	$\underline{)}\ \$$	$\$)L'$	$L' \to \varepsilon$
		$\$\ \underline{)}$	
8.	$\underline{\$}$	$\$$	

By following these steps we can generate a parse tree for the string $w = (a, a, a)$.



Parse tree for the string $w = (a, a, a)$.

6.4.4.4 LL(1) Parsing Table Construction

1. **First set:** First(A) gives a set of all terminals that may begin in a string derived from A. To get first of A, start finding all strings generated by that variable in a language and then pick the first element of every string.

   Rules:
   - If $S \to ab/cd/ef$ then
     First(S) = $\{a, c, e\}$
   - If $S \to ab$ then
     First(S) = $\{a\}$
     First(ε) = (ε)
     First(a) = $\{a\}$, where 'a' is terminal
   - If $S \to AB$
     $A \to a$
     $B \to b$
     then First(s) = $\{a\}$
   - If $S \to aS/b$ then
     First(S) = $\{a, b\}$
   - If $S \to AB$
     $A \to a/c/d/\varepsilon$
     $B \to b$
     then First(S) = $\{a, c, d, b\}$
2. **Follow set:** Follow(A) gives a set of all terminals that may follow immediately to the right of A. ε can never be a part of any follow set.

   Rules:
   - If S is the starting symbol, then \$ is also one of the element of Follow(S).
   - If $S \to aBCDE$, then
     Follow(C) = First(DE) = First(D)
   - If $S \to AB$ or $S \to ABC$ and $C \to \varepsilon$, then
     Follow(B) = Follow(S)

Problem 6.7: Find the first and the follow sets for the given grammar:

$E \to TE'$
$E' \to \varepsilon/ + TE'$
$T \to FT'$
$T' \to \varepsilon/*FT'$
$F \to \text{id}/(E)$

Solution:

Variable	First( )	Follow( )
E	id, (	\$,)
E'	+, ε	\$,)
T	id, (	+, \$,)
T'	*, ε	+, \$,)
F	id, (	*, +, \$,)

Problem 6.8: Find the first and the follow sets for the given grammar:

$$S \rightarrow (L)/a$$
$$L \rightarrow SL'$$
$$L' \rightarrow \varepsilon/,SL'$$

Solution:

Variable	First( )	Follow( )
S	(, a	,,), \$
L	(, a	)
L'	,, ε	)

6.4.4.5 Algorithm for Constructing a Parsing Table

If a grammar G is given, the steps to construct a parsing table is as follows:

For each production $A \rightarrow \alpha$, do the following

1. Add $A \rightarrow \alpha$ under $M[A, b]$, where $b \in$ First(α).
2. If First(α) contains ε, then add $A \rightarrow \alpha$ under $M[A, c]$, where $c \in$ Follow(A).

Number of parsing table entries = $\{V^*(T + 1)\}$, where V is the number of variables in the given grammar and T is the number of terminals in the given grammar.

Problem 6.9: Construct the parsing table from the following grammar:

$$E \rightarrow TE'$$
$$E' \rightarrow \varepsilon/ + TE'$$
$$T \rightarrow FT'$$
$$T' \rightarrow \varepsilon/*FT'$$
$$F \rightarrow \text{id}/(E)$$

Solution:

The first and the follow sets for the given grammar are as follows:

Variable	First( )	Follow( )
E	id, (	\$,)
E'	+, ε	\$,)
T	id, (	+, \$,)
T'	*, ε	+, \$,)
F	id, (	*, +, \$,)

Number of cells in parsing table = $\{V^*(T + 1)\}$

So, table entries = $\{5^*(5 + 1)\} = 30$

Parsing table constructed with the help of the first and follow sets of given grammar:

	Id	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
F	$F \rightarrow$ id			$F \rightarrow (E)$		

How to Check a Given Grammar (G) Is LL(1) or Not?

The following are two ways to identify that a given grammar is LL(1) or not:

1. By constructing parsing table, if parsing table does not have more than one production entry in any of the cell then only it is LL(1).
2. By checking the following two conditions:
   - If a given grammar G does not contain null(ε) production:
     Let grammar G be
     $$A \rightarrow \alpha_1/\alpha_2/\alpha_3$$
     Then α_1, α_2, α_3 should be pair wise disjoint. That is,
     $$\text{First}(\alpha_1) \cap \text{First}(\alpha_2) = \varnothing$$
     $$\text{First}(\alpha_2) \cap \text{First}(\alpha_3) = \varnothing$$
     $$\text{First}(\alpha_3) \cap \text{First}(\alpha_1) = \varnothing$$
   - If grammar G contains null(ε) production:
     Let grammar G be
     $$A \rightarrow \alpha_1/\alpha_2/\varepsilon$$
     Find First(α_1), First(α_2) and Follow(A), they should be pair wise disjoint.
     $$\text{First}(\alpha_1) \cap \text{Follow}(A) = \varnothing$$
     $$\text{First}(\alpha_2) \cap \text{Follow}(A) = \varnothing$$

Problem 6.10: Check whether the following grammar is LL(1) or not.

$$S \rightarrow E/a$$
$$E \rightarrow a$$

Solution: In the given grammar there is no null production, so

$$\text{First}(S) = a$$

$$\text{First}(E) = a$$

According to the given condition 1, First$(S) \cap$ First$(A) \neq \varnothing$. So the given grammar is not LL(1).

Problem 6.11: Check the following grammar is LL(1) or not.

$$S \rightarrow aABb$$
$$A \rightarrow a/\varepsilon$$
$$B \rightarrow d/\varepsilon$$

Solution: In the given grammar there is null production, so find first and follow of those productions which have null production.

$$\left.\begin{cases} \text{First}(A) = a \\ \text{Follow}(A) = d, b \end{cases}\right\} \text{No common element}$$

$$\left.\begin{cases} \text{First}(B) = d \\ \text{Follow}(B) = b \end{cases}\right\} \text{No common element}$$

Important Points

1. Every regular grammar need not be LL(1) because that grammar may contain left factoring.
2. Any ambiguous grammar cannot be LL(1).
3. If any grammar contains left factoring, then it cannot be LL(1) grammar.
4. If a given grammar contains left recursion, then it cannot be LL(1) grammar.

6.4.5 Bottom-Up Parser

In bottom-up parser, parse tree construction starts from children and proceeds to root. Bottom-up parser uses RMD for constructing the parse tree. A substring which will give reduction of string is called handle.

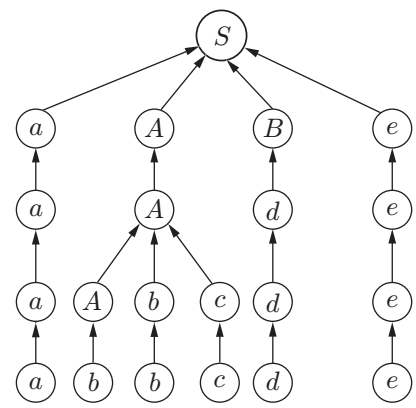
The difficulty with bottom-up parser is identifying the right handle which will give one required variable so that we will go to start symbol.

Problem 6.12: String $w = abbcde$, grammar G is given below:

$$S \rightarrow aABe$$
$$A \rightarrow Abc/b$$
$$B \rightarrow D$$

Draw a parse tree for reduction of a string w with the help of the grammar G.

Solution: The parse tree for reduction of the string $w = abbcde$.



6.4.5.1 Classification of Bottom-Up Parser

The bottom up parser is classified as shown in Fig. 6.10.

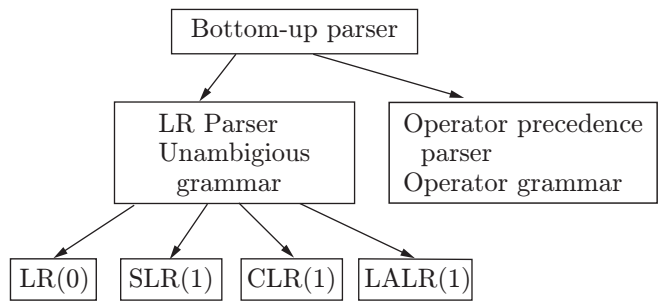


Figure 6.10 Classification of bottom-up parser.

LR Parser

In computer science, an LR parser is a type of bottom-up parser that efficiently handles context-free languages in guaranteed linear time. An LR parser reads input from left to right and produces an RMD. The canonical LR (CLR), look-ahead LR (LALR) and simple LR (SLR) parsers are common variants of LR parsers. For all types of LR parsers, parsing algorithm is same but parsing tables are different.

LR Parsing Algorithm

Let S be the state on top of the stack and 'a' be the look-ahead symbol, then

1. If action $[S, a] = S_i$, then shift 'a' and 'i', and also increment the i/p pointer.
2. If action $[S, a] = r_j$ and r_j is $\alpha \rightarrow \beta$, then pop $2|\beta|$ symbols and replace by α. If S_{m-1} is the state below α, then push Goto $[S_{m-1}, \alpha]$.

3. If action $[S, a]$ = accepted, then successful parsing.
4. If action $[S, a]$ = blank, then parsing error.

Stack Operations

1. **Shift:** Shifts the next input symbol onto the top of the stack.
2. **Reduce:** Replaces a set of grammar symbol or handle on the top of the stack with the LHS of a production rule.
3. **Accept:** Declares successful completion of the parsing.

LR(0) Parser

1. **LR(0) parsing table construction:** For constructing LR(0) parsing table we have to follow the given steps:
   - Construct augmented grammar.
     Let given grammar G be
     $$S \to AA$$
     $$A \to aA/b$$
     Augmented grammar G' will be
     $$S' \to S$$
     $$S \to AA$$
     $$A \to aA/b$$
     $S' \to S$ is called the augmented production.
   - Find closure $\{I_0 = \text{closure } (S' \to \cdot S)\}$.
     To find closure of a production I, follow these steps:
     - In closure of I, add I also to closure (I).
     - If I is $A \to B \cdot CD$ and $C \to EF$ is in the given grammar G, then add $C \to \cdot EF$ to closure (I).
     - Repeat these steps for every newly added LR(0) item.

     This can be understood clearly by the given example; it uses the above grammar to find closure.
     Closure $(S' \to \cdot S) = \{S' \to \cdot S | S \to \cdot AA | A \to \cdot aA | A \to \cdot b\}$
     Closure $(S \to \cdot AA) = \{S \to \cdot AA | A \to \cdot aA | A \to \cdot b\}$
     Closure $(A \to \cdot aA) = \{A \to \cdot aA\}$
   - Using I_0 construct deterministic finite automata (DFA).
   - Reduce DFA into LR(0) parsing table.
   - Find Goto (I, X).
     Goto (I, X) function is used to fill the second part of the parsing table, which have entries related to move after X.

Example 6.6

Consider the above grammar for the Goto function.

- Goto $(S' \to \cdot S, S) = S' \to S$.
- Goto $(S' \to \cdot AA, A) = S \to S \cdot | A \to \cdot aA | A \to \cdot b$
- Goto $(A \to \cdot aA, a) = A \to a \cdot A | A \to \cdot aA | A \to \cdot b$

Problem 6.13: Consider the following grammar and construct the LR parsing table.

$S \to AA$ (production number 1)

$A \to aA$ (production number 2)

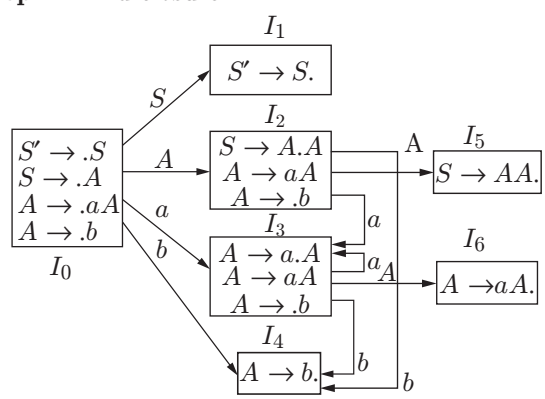
$A \to b$ (production number 3)

Solution:

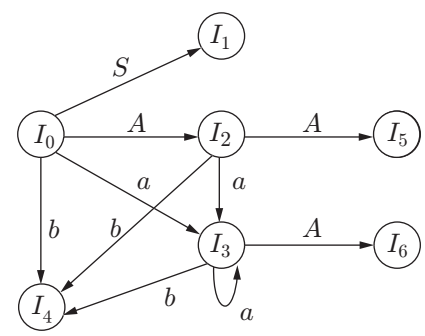
Step 1: Construct the augmented grammar.

$S' \to S$
$S \to AA$
$A \to aA$
$A \to b$

Step 2: Find closure.



Step 3: Construct DFA.



Step 4: Construct LR(0) parsing table.

- S_i denotes the shift move in the above DFA, where 'i' denotes the state number.
- r_i represents reduction move in the above DFA, where 'i' denotes the production number.

State	Action			Goto	
	A	***b***	**$**	***S***	***A***
I_0	S_3	S_4		1	2
I_1			Accepted		
I_2	S_3	S_4			5
I_3	S_3	S_4			6
I_4	r_3	r_3	r_3		
I_5	r_1	r_1	r_1		
I_6	r_2	r_2	r_2		

Problem 6.14: Check whether the following grammar is LR(0) or not.

$S \rightarrow (L)$ (production number 1)

$S \rightarrow a$ (production number 2)

$L \rightarrow L, S$ (production number 3)

$L \rightarrow S$ (production number 4)

Solution:

Step 1: Construct the augmented grammar.

$S' \rightarrow S$

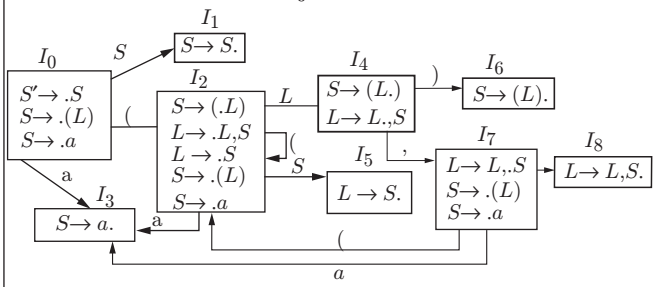
$S \rightarrow (L)$

$S \rightarrow a$

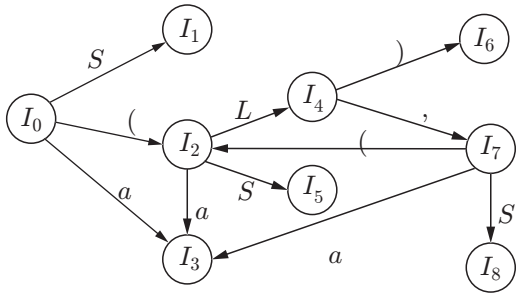
$L \rightarrow L, S$

$L \rightarrow S$

Step 2: Find closure ($I_0 = S' \rightarrow S$).



Step 3: Construct DFA.



Step 4: Construct LR parsing table.

State	Action					Goto	
	A	(	)	,	\$	S	L
I_0	S_3	S_2				1	
I_1					Accepted		
I_2	S_3	S_2				5	4
I_3	r_2	r_2	r_2	r_2	r_2		
I_4			S_6	S_7			
I_5	r_4	r_4	r_4	r_4	r_4		
I_6	r_1	r_1	r_1	r_1	r_1		
I_7	S_3	S_2				8	
I_8	r_3	r_3	r_3	r_3	r_3		

The given grammar is LR(0) because there are no multiple entries in the same cell.

Important Points

1. If there are two reductions on any state of DFA, then in table entry we have to put both in that cell. In that situation parser will not be able to decide for which reduction it has to parse, so this problem is called reduce-reduce (R-R) problem.
2. If in any state one production is reduced and another is shifted then also parser is not able to parse it. This problem is called shift-reduce (S-R) problem.
3. Due to DFA no shift-shift (S-S) problem occurred.
4. For conflict, there should be two productions and at least one of them should be reduced.
5. Goto section does not participate in conflict.

Problem 6.15: Check if the following grammar is LR(0) or not.

$E \rightarrow T + E$ .....(production number 1)

$E \rightarrow T$ .....(production number 2)

$T \rightarrow$ id .....(production number 3)

Solution:

Step 1: Construct an augmented grammar.

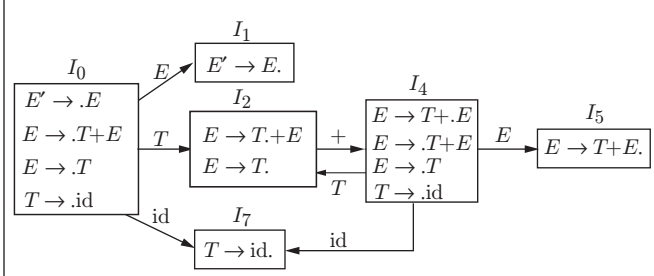
$E' \rightarrow E$

$E \rightarrow T + E$

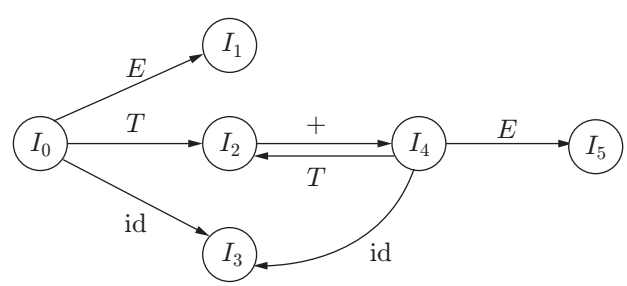
$E \rightarrow T$

$T \rightarrow$ id

Step 2: Find closure.



Step 3: Construct DFA.



Step 4: Construct LR() parsing table.

	Action			Goto	
	+	Id	$	E	T
I_0		S_3		1	2
I_1			Accepted		
I_2	S_4/r_2	r_2	r_2		
I_3	r_3	r_3	r_3		
I_4	S_3			5	2
I_5	r_1	r_1	r_1		

The given grammar is not LR(0) because there are multiple entries (S_4/r_2) in the same cell. This conflict is called as **SR (shift-reduce) conflict**.

SLR(1) Parser

SLR(1) parser is one of the variants of LR parser. An SLR parser is efficient at finding the single correct bottom-up parser in a single scan without backtracking. SLR(1) parser table has only one difference from LR(0) parsing table, reduction entries are made only in the specific location. Let production $E \rightarrow T$ be reduced, then entry of the reduced production will be in those column which comes in Follow(T). If there are multiple entries in the same cell, then the given grammar will not be SLR(1).

From Problem 6.15, we will construct parse table only for SLR(1) (Table 6.1). State 2 has shift and reduce entry, but we have to find that where we have to put the reduce entry. The production which is reduced is $E \rightarrow T$, so we will put r_2 at places which come in Follow(T).

$$\text{Follow } (T) = \{+, \$\}$$

Table 6.1 | Parsing table for SLR(1)

	Action			Goto	
	+	Id	$	E	T
I_0		S_3		1	2
I_1			Accepted		
I_2	S_4/r_2		r_2		
I_3	r_3	r_3	r_3		
I_4	S_3			5	2
I_5	r_1	r_1	r_1		

There are two entries in the same cell, so this is not SLR(1) grammar.

Canonical LR Parser

Canonical LR parser is a simplified version of an LR parser and is also known as CLR parser. CLR parser is the most powerful parser of the LR parser family. Closure and Goto function are processed differently than in SLR parser.

1. **Closure function of CLR:**
   Closure(I)
   - If I is $A \rightarrow B{\cdot}CD$, \$ and $C \rightarrow {\cdot}EF$ is in G, then add $C \rightarrow {\cdot}EF$, {First(D), \$}.
   - Also add I to closure(I).
   - Repeat above steps for every newly added items.
2. **Goto function of CLR:**
   Goto(I, X)
   - Add I to Goto(I, X) and also move dot(·) after X.
   - Apply closure to the result obtained in the previous step.

Problem 6.16: Construct parsing table for the following grammar and also check that the given grammar is CLR(1) or not?

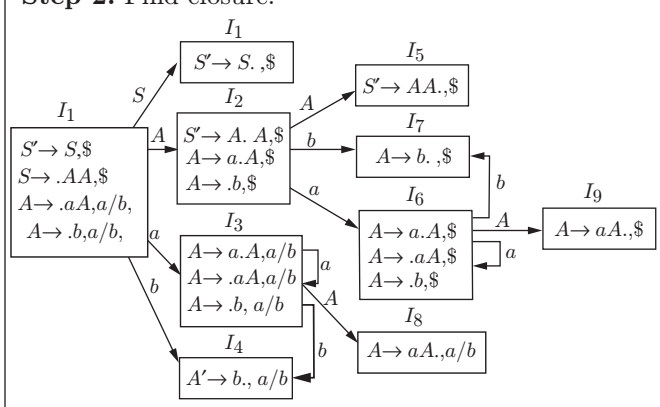
$$S \rightarrow AA$$
$$A \rightarrow aA$$
$$A \rightarrow b$$

Solution:

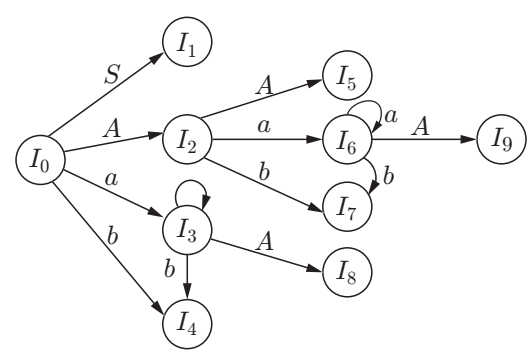
Step 1: Construct an augmented grammar.

$$S' \rightarrow S$$
$$S \rightarrow AA$$
$$A \rightarrow aA$$
$$A \rightarrow b$$

Step 2: Find closure.



Step 3: Construct DFA.



Step 4: Construct parsing table.

	Action			Goto	
State	A	B	$	S	A
I_0	S_3	S_4		1	2
I_1			Accepted		

(*Continued*)

Continued

State	Action			Goto	
	A	B	\$	S	A
I_2	S_6	S_7			5
I_3	S_3	S_4			8
I_4	r_3	r_3			
I_5			r_1		
I_6	S_6	S_7			9
I_7			r_3		
I_8	r_2	r_2			
I_9			r_2		

Here, in our example we have three productions which are reduced.

$S \to AA$ production 1 (represented by r_1)

$A \to aA$ production 2 (represented by r_2)

$A \to b$ production 3 (represented by r_3)

To know in which place r_3 will get entry, check what the look-ahead terminals in the closure are of the given production of the grammar. Production $A \to b$ has $-a$, b as look-ahead terminals. So reduction r_3 will get entry under a and b column. Given grammar is CLR(1) grammar because there are two entries in the same cell.

Note: CLR(1) parser is more powerful than other variants of LR parser, but it is costlier than the other due to more number of states in the DFA of CLR(1), even though two states are exactly the same other than look-ahead symbol. So by merging those two states together, CLR(1) can be minimized and the minimized CLR(1) is called LALR(1).

Operator Precedence Parser: It is based upon bottom-up parsing technique that follow shift-reduce parsing method. Operator precedence parser interprets an operator precedence grammar. Operator precedence parser is capable of parsing all LR(1) grammars.

Operator grammar: A grammar G is said to be operator grammar iff

1. G does not have null production.
2. G does not have two adjacent variables on the right-hand side of the production.

Problem 6.17: Which is/are operator grammar in the given set?

(a) $E \to E + E|E^*E|\text{id}$
(b) $E \to AB,\ A \to a,\ B \to b$
(c) $E \to E + E|E^*E|\text{id}|\varepsilon$

Solution:

(a) Operator grammar
(b) Not operator grammar
(c) Not operator grammar

Problem 6.18: Convert the following grammar into operator grammar.

$P \to SR|S$
$R \to bSR|bS$
$S \to WbS|W$
$W \to L^*W|L$
$L \to \text{id}$

Solution:

The operator grammars for the given grammars are:

$P \to SbP|SbS|S$
$R \to bP|bS$
$S \to WbS|W$
$W \to L^*W|L$
$L \to \text{id}$

Important Points

1. The relation between CLR(1), LALR(1), SLR(1), LR(0) and LL(1) is shown in Fig. 6.11.

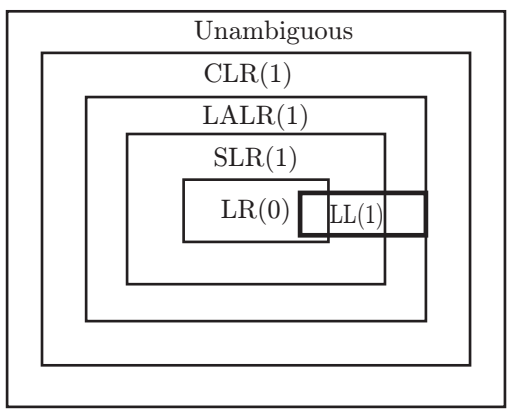


Figure 6.11 Relation between CLR(1), LALR(1), SLR(1), LR(0) and LL(1).

2. The relation between CLR(1), LALR(1), SLR(1), LR(0) and operator precedence parser is shown in Fig. 6.12.

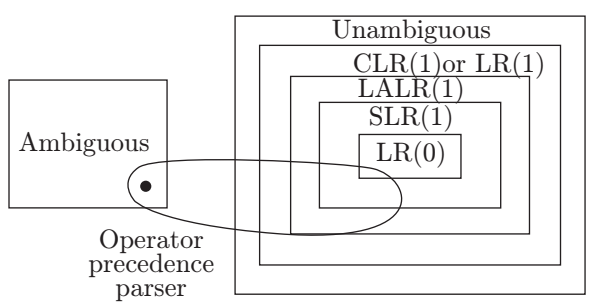


Figure 6.12 Relation between CLR(1), LALR(1), SLR(1), LR(0) and operator precedence parser.

3. If grammar G is LL(1), then
   - It may be LR(0).
   - It may be SLR(1).
   - It will surely be LALR(1).
4. Every LL(1) grammar is surely LALR(1), but if any grammar is LALR(1) then it may or may not be LL(1).
5. If any grammar is LALR(1), then it will surely be CLR(1).
6. CLR(1) is also known as LR(1).
7. Every LL(1) grammar will surely be CLR(1).
8. All LL(k) parsers are subset of LR(k) parser.
9. If the number of states in LR(0), SLR(1), CLR(1) and LALR(1) is n_1, n_2, n_3, n_4, respectively, then the relation between them is

$$n_1 = n_2 = n_4 \leq n_3$$

6.5 SYNTAX-DIRECTED TRANSLATION

Syntax-directed translation (SDT) is a method of compiler implementation which attaches semantic rules with every production of a CFG while translating a string into a sequence of actions. An SDT can be implemented by first constructing a parse tree and then performing the actions in a pre-order traversal.

Example 6.7

Let a production be $E \rightarrow E_1 + E_2$, then SDT will be

$$E \rightarrow E_1 + E_2 \left\{ \begin{array}{c} E \cdot \text{val} = E_1 \cdot \text{val} + E_2 \cdot \text{val} \\ \text{Print}\ (E \cdot \text{val}); \end{array} \right\}$$

6.5.1 Attributes of Syntax-Directed Translation

The following are two types of attributes supported by the SDT:

1. **Synthesized attribute:** An attribute is said to be synthesized only if its value is calculated in terms of its children in the parse tree.
2. **Inherited attribute:** An attribute is said to be inherited attribute only if its value is calculated in terms of its parent or children or both in the parse tree.

6.5.2 Types of Syntax-Directed Translation

An SDT is basically divided into two types: S-attribute definition and L-attribute definition (Table 6.2).

Table 6.2 | Characteristics of S-attribute and L-attribute definitions

S Attribute Definition	L Attribute Definition
S-attribute definition uses only synthesized attributes.	L-attribute definition uses both synthesized and inherited attributes.
Semantic rules should be placed at the rightmost place of the right-hand side.	Semantic rules can be placed anywhere on the right-hand side of its production.
Evaluated by a bottom-up or post-order traversal of a parse tree.	Evaluated by a bottom-up or pre-order traversal of a parse tree.

6.5.3 Applications of SDT

1. Evaluating arithmetic expression
2. Creating syntax tree
3. Converting infix to postfix
4. Converting infix to prefix
5. Generating intermediate code
6. Converting binary to decimal
7. Storing type information into symbol table

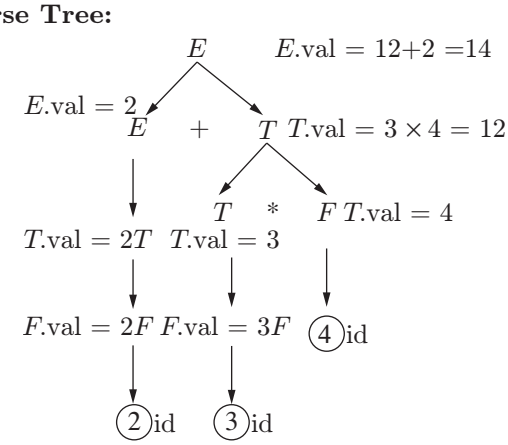
Problem 6.19: Construct SDT to evaluate the given arithmetic expression:

Input: 2 + 3*4
Output: 14

Solution:

+Grammar	Semantic Rules
$E \rightarrow E + T$	E.val = E.val + T.val Print (E.val);
$E \rightarrow T$	E.val = T.val
$T \rightarrow T*F$	E.val = T.val * F.val
$T \rightarrow F$	T.val = F.val
$F \rightarrow$ id	F.val = id

Parse Tree:



Parse tree for given arithmetic expression 2 + 3*4.

Problem 6.20: Consider the following SDT:

$S \rightarrow TR$

$R \rightarrow +T$ {print (+);} R

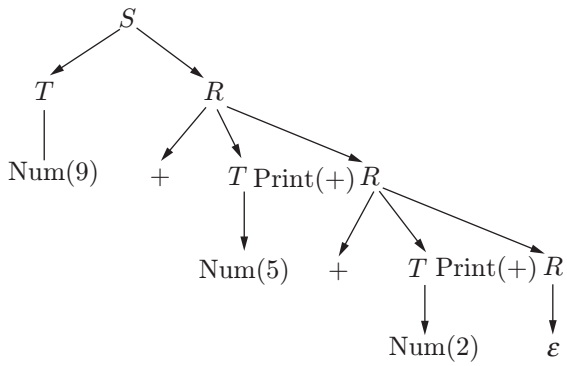
$R \rightarrow \varepsilon$

$T \rightarrow$ Num {print (num);}

For input string $(9 + 5 + 2)$, what will be the output?

Solution:

We first draw the parse tree and then determine the output.



Parse tree for given arithmetic expression $(9 + 5 + 2)$.

Output = 95 + 2+

Problem 6.21: Construct an SDT to convert infix expression to postfix expression.

Input: $a + b^*c$

Output: $abc^* +$

Solution:

The SDT is constructed as follows:

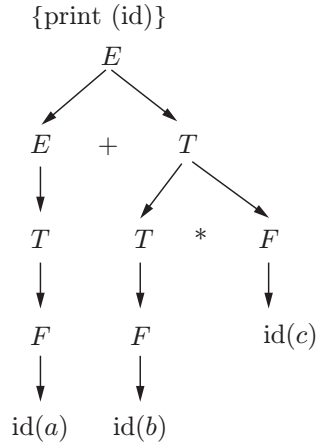
$E \rightarrow E + T$ {print (+)}

$E \rightarrow T$

$T \rightarrow T^*F$ {print (*)}

$T \rightarrow F$

$F \rightarrow$ id {print (id)}



Parse tree for given infix expression $a + b^*c$.

Problem 6.22: Construct an SDT to convert infix expression to prefix expression.

Input: $a + b^*c$

Output: $+ a^*b\ c$

Solution:

The SDT is constructed as follows:

$E \rightarrow$ {print (+)} $E + T$

$E \rightarrow T$

$T \rightarrow$ {print (*)} T^*F

$T \rightarrow F$

$F \rightarrow$ id {print (id)}

6.6 RUNTIME ENVIRONMENT

To generate the target code, there is a need to know about the environment where the source program will execute. It consists mapping of names and objects in the memory, procedure activation, storage allocation, library routines and exception handling, scopes and extents of declarations and symbol table organization.

6.6.1 Storage Organization

Runtime memory needs to be subdivided as follows to hold the different components of an executing program (Fig. 6.13):

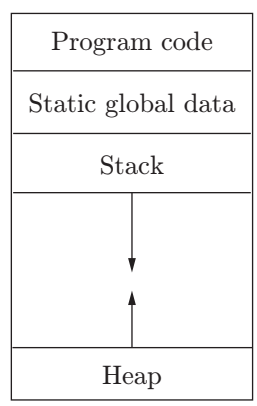


Figure 6.13 | Storage organization.

1. **Program code:** Refers to static area used by the generated target code which is fixed at compile time.
2. **Static global data:** Refers to storage space for data, which does not change during the execution of the program.
3. **Stack:** Manages activation of procedures at runtime. The area usually grows towards lower address.
4. **Heap:** Holds variable created at runtime.

There are two different approaches for runtime storage allocation, as given in Table 6.3.

Table 6.3 | Static and dynamic allocation

Static Allocation	Dynamic Allocation
Allocates all needed space when program starts.	Allocates space when it is needed.
Deallocates all space when program terminates.	Deallocates space when it is no longer needed.

6.6.2 Activation Record and Activation Trees

Activation is the function in execution mode. The function code is the static part whereas execution is the counterpart. The storage associated with an activation of a procedure is called **activation record.**

1. **Activation record content:**
   - *Temporary values*: Values generated as a result of the expression evaluations which cannot be put in registers.
   - *Local data*: Local data to belong to the procedure.
   - *Saved machine*: Keeps the context or machine status (register, PC, etc.).
   - *Access link*: Points to non-local data in other AR.
   - *Control link*: Points to the caller's activation record the return value space of the called function, if any.
   - *Actual parameters*: Used by the calling procedure to pass parameters to called procedures; registers are used to pass these information.

For a recursive procedure or function, several activations may be alive simultaneously. Activation tree shows the path through which control enters and leaves activations for single run of a program. Each node represents an activation of a function, if an arrow is facing a child node from the parent node that means the child function is called by the parent function. Sibling 'a' is left to 'b' means that function 'a' is called before function b.

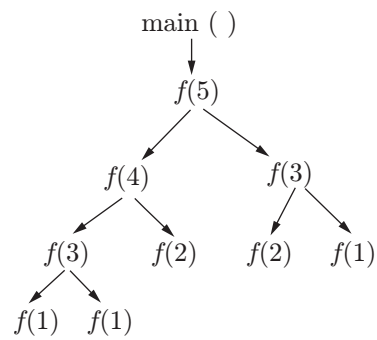
Problem 6.23: Draw an activation tree of the given example.

```
int a[10];
int main(){
int i;
for (i=0; i<10; i++){
a[i] = f(i);
     }
}
int f (int n) {
if (n<3)  return 1;
  return f(n-1)+f(n-2);
           main ends
        }
```

Solution: The activation takes place as follows:

```
System starts main
        enter f(5)
            enter f(4)
                enter f(3)
            enter f(2)
                exit f(2)
                enter f(1)
                exit f(1)
               exit f(3)
               enter f(2)
               exit f(2)
            exit f(4)
            enter f(3)
              enter f(2)
             exit f(2)
             enter f(1)
             exit f(1)
            exit f(3)
        exit f(5)
```

The activation tree for $f(5)$ is as follows:



Activation tree of $f(5)$.

The following observations are made based upon the activation tree:

1. Order of activation corresponds to the pre-order traversal of the tree.
2. Order of deactivation corresponds to the post-order traversal of the tree.
3. If an activation of p calls q, then p will not terminate before q.

6.6.3 Procedure Call Return Model

Every machine's architecture and every language are slightly different from each other. The basic steps followed by a function call are as follows:

1. Before a function call, the calling routine:
   - Saves any necessary registers
   - Pushes the arguments onto the stack for the target call
   - Sets up the static link (if appropriate)
   - Pushes the return address onto the stack
   - Jumps to the target

2. During a function call, the target routine:
   - Saves any necessary registers
   - Sets up the new frame pointer
   - Makes space for any local variables
   - Does its work
   - Tears down frame pointer and static link
   - Restores any saved registers
   - Jumps to saved return address
3. After a function call, the calling routine:
   - Removes return address and parameters from the stack
   - Restores any saved registers
   - Continues executing

6.6.4 Lexical Versus Dynamic Scoping

Lexical (or Static Scoping)	Dynamic Scoping
Binding of variable to declarations is done at compile time.	Binding of variable to declarations is done at compile time.
Lexical scope rules specify the association of variables with declaration based on the textual order of the source code.	Dynamic scoping can be achieved by copying a function verbatim at the place of call.
Innermost enclosing block rule.	Most recent occurrence rule.
Name resolution is independent of caller.	Name resolution depends on caller.
Used by most modern languages: C/C++, Pascal, etc.	Formerly used in some interpreted languages (older versions of LISP).

6.6.5 Symbol Table

It is a data structure used by compiler to store all the information of tokens generated by lexical analyzer. After the lexical analysis phase, semantic analyzer performs type checking on the input code. During type checking, semantic analyzer checks whether the use of names (such as type names, variables, functions) is consistent with their definition in the source code. Consequently, if there are any inconsistencies or misuses found during type checking then it is shown as an error. This is the task of a symbol table.

Operations that can be performed on a symbol table are as follows:

1. Insert
2. Delete
3. Search

Symbol table can be implemented by:

1. Ordered list
2. Unordered list
3. Hash table
4. Tree

Possible entries in a symbol table are as follows:

1. Name
2. Data type
3. Size
4. ID
5. Scope information
6. Storage allocation

6.7 INTERMEDIATE CODE GENERATION

The source program can be directly converted into the target language. But there are many benefits of having an intermediate code or machine-independent code such as increased abstraction, clear separation between front end and back end, and easily retarget and code optimization technique can also be applied (Fig. 6.14).

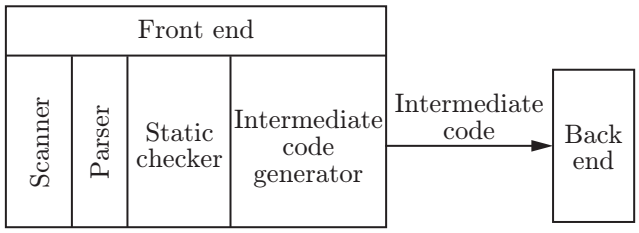


Figure 6.14 Intermediate code generation.

6.7.1 Intermediate Representations

The commonly used representations of intermediate code are as follows:

1. Syntax tree
2. Postfix notation
3. Three-address code

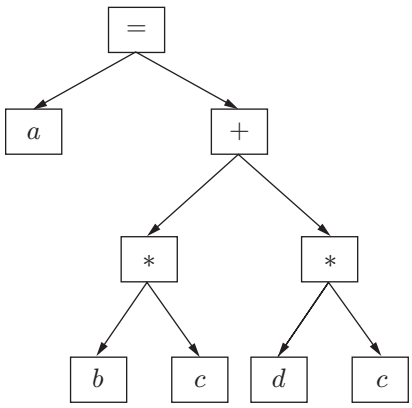
6.7.1.1 Syntax Tree

A syntax tree is the tree representation of the abstract syntactic structure of source code. Due to its abstract nature it is also called as abstract syntax tree. Each of the tree node denotes a construct occurring in source code.

Example 6.8

Statement written in source language:

$a = b*c + d*c$



Syntax tree of $a = b*c + d*c$.

6.7.1.2 Postfix Notation

Postfix notation is also known as 'reverse polish' notation. Any expression can be written unambiguously. Interpreters can be built easily for postfix notation by using the stack data structure. In the postfix notation, an operator follows the operand.

Example 6.9

The source language statement:

$$a = b*c + d*c$$

Can be rewritten in postfix notation as:

$$abc*dc* + =$$

6.7.1.3 Three-Address Statement

Three-address statement is a linearized representation of an abstract syntax, in which names of the temporaries correspond to the nodes. The intermediate values name allows three-address code to be easily rearranged, which is convenient for optimization technique. The reason to call 'three-address code' is that each statement generally contains three addresses, two for the operands and one for the result. Representations of the three-address statement are quadruples, triples and indirect triples.

1. **Quadruples:** A quadruple is a record structure having four fields, namely op, arg1, arg2 and result. The op field contains an internal code for operator. To translate binary expression 'x operator y' into three-address code, operator is placed in 'op' column; x in 'arg1' column, y in 'arg2' column and a new temporary variable stores their calculated result in the result column.

Example 6.10

The quadruples for the assignment $a = b*c + d*e$ generates the following three-address code:

	Op	arg1	arg2	Result
(0)	=	C		t_1
(1)	*	B	t_1	t_2
(2)	=	D		t_3
(3)	*	E	t_3	t_4
(4)	+	t_2	t_4	t_5
(5)	=	t_5		A

2. **Triples:** Temporary variables or memory names in the symbol table can be avoided by the position number of the statement that computes it. If we used this method then three address statements can be represented by records with three columns operation, arg1 and arg2. The column arg1 and arg2, for the arguments of operation, are either pointers to the symbol table or pointers into the triple structure. As numbers of used fields are three, so this format is known as triples.

Example 6.11

	Operation	Arg1	Arg2
(0)	=	c	
(1)	*	b	(0)
(2)	=	c	
(3)	*	b	(2)
(4)	+	(1)	(3)
(5)	=	a	(4)

3. **Indirect triples:** Triples are very difficult to optimize because for optimization it required moving of intermediate code and other triples connected to it also have to be updated. So, a new variant of

triples is called indirect triples is being used which is easier to optimized. Indirect triples perform listing pointers to triples, rather than listing the triples themselves.

Example 6.12

#	Stmt		#	Op	Arg1	Arg2
(0)	(14)	→	(14)		C	
(1)	(15)	→	(15)	*	B	(14)
(2)	(16)	→	(16)		C	
(3)	(17)	→	(17)	*	B	(16)
(4)	(18)	→	(18)	+	(15)	(17)
(5)	(19)	→	(19)	=	A	(18)

Program Triple container

6.8 CODE OPTIMIZATION

Code optimization phase is to reduce the size of the code and improve the performance of the code generated by intermediate code phase. The most important part of optimized code is to minimize the amount of time taken by the code to execute and less common is to minimize the amount of memory used by the code. The basic requirement optimization methods should comply with is that an optimized program must have the same output and side effects as its non-optimized version.

6.8.1 Types and Levels of Optimization

Optimization can be performed by automatic optimizers or programmers. An optimizer is a built-in unit of a compiler. Modern processors have the capability to optimize the execution order of code instructions. Optimizations are classified mainly into two types, one is high-level and the other is low-level optimizations. High-level optimizations are generally performed by the programmer who handles source code of the programs such as classes, functions, control statements, and procedures. Low-level optimizations are performed at the stage when source code is compiled into a set of machine instructions, and it is at this stage that automated optimization is usually employed.

6.8.2 Primary Source of Optimization

Optimization can be done by using different ways. Some of the primary source of optimization is discussed below:

6.8.2.1 Dead-Code Elimination

Dead-code is a section of program code which is executed but the result produced by that section is never used. So, it can be removed from the program code because it does not have any effect on the functionality of program.

Let we have temp1 = temp2 − temp3, and temp1 is never used further in the program then we can eliminate this whole instruction.

6.8.2.2 Constant Propagation

Constant propagation is the process of substituting a constant value by the subsequent uses of a variable while there is no intervening changes in the value of that variable. Consider the given example below:

```
int a = 10;
int b = 12 - a / 2;
return b = b * (35/a +2);
```

Variable 'a' propagating the constant value 10 in the given code.

6.8.2.3 Constant Folding

Constant folding is the process of evaluating the constant expression at compile time. Constant folding process improves run-time performance and also reduces code size by evaluating constant at compile time. In program code below, the expression (4–2) can be evaluated at compile time and replaced with the constant 2.

```
int sub( )
{
Return (4-2);
}
```

6.8.2.4 Elimination of Common Sub-Expression

If two or more operations are similar and produce same results then it is efficient way that to compute at once and refer that results further rather than re-evaluate it. In the given program code below, temp 3 and temp 6 produce similar results and temp 4 and temp 5 also

produce similar results, so we can compute them once and refer results further below in the code.

```
main()
{
int x, y, z;

x = (1+20)* -x;
y = x*x+(x/y);
y = z = (x/y)/(x*x);
}

straight translation:

tmp1 = 1 + 20;
tmp2 = -x;
x = tmp1 * tmp2;
tmp3 = x * x;
tmp4 = x/y;
y = tmp3 + tmp4;
tmp5 = x/y;
tmp6 = x * x;
z = tmp5/tmp6;
y = z;
```

What sub-expressions can be eliminated? How can valid common sub-expressions (live ones) be determined? Here is an optimized version, after constant folding and propagation and elimination of common sub-expressions:

```
tmp2 = -x;
x = 21 * tmp2;
tmp3 = x * x;
tmp4 = x/y;
y = tmp3 + tmp4;
tmp5 = x/y;
z = tmp5/tmp3;
y = z;
```

6.8.2.5 Copy Propagation

Copy propagation is the different way of optimization, in which assignment a=b for some variable 'a' and 'b', we can replace later uses of 'a' with use of 'b' (it is to be assumed that there is no change to either variable in-between). The code on the left side makes a copy of tmp1 in tmp2 and a copy of tmp3 in tmp4. When we do optimization then on the right, we eliminated those unnecessary copies and propagated the original variable into later uses.

```
tmp2 = tmp1;
tmp3 = tmp2 * tmp1;
tmp4 = tmp3;
tmp5 = tmp3 * tmp2;
c = tmp5 + tmp4;
tmp3 = tmp1 * tmp1;
tmp5 = tmp3 * tmp1;
c = tmp5 + tmp3;
```

IMPORTANT FORMULAS

1. Time complexity of RDP is $O(2^n)$.
2. Every regular grammar need not be LL(1), because that grammar may contain left factoring.
3. Any ambiguous grammar cannot be LL(1).
4. If any grammar contains left factoring, then it can't be LL(1) grammar.
5. If given grammar contains left recursion, then it can't be LL(1) grammar.
6. If grammar G is LL(1), then
   - It may be LR(0).
   - It may be SLR(1).
   - It will surely be LALR(1).
7. Every LL(1) grammar will surely be LALR(1), but if any grammar is LALR(1) then it may or may not be LL(1).
8. If any grammar is LALR(1), then it will surely be CLR(1).
9. CLR(1) is also known as LR(1).
10. Every LL(1) grammar will surely be CLR(1).
11. All LL(k) parsers are subset of LR(k) parser.
12. If the number of states in LR(0), SLR(1), CLR(1) and LALR(1) is n_1, n_2, n_3, n_4, respectively, then relation between them is

$$n_1 = n_2 = n_4 \leq n_3$$

SOLVED EXAMPLES

1. Which of the following derivations does a top-down parser use while parsing an input string? The input is assumed to be scanned in left to right order.

(a) Leftmost derivation
(b) Topmost derivation
(c) Rightmost derivation
(d) Leftmost derivation traced out in reverse

Solution: Top–Down parser uses leftmost derivation for constructing parse tree.

Ans. (a)

2. The process of assigning load addresses to various parts of the program and adjusting the code and data in the program to reflect the assigned addresses is called

(a) address assembly (b) parsing
(c) relocation (d) indexing

Solution: The process adjusting the code and data in the program to reflect the assigned addresses is called relocation.

Ans. (c)

3. Which of the following statements is true?

(a) An unambiguous grammar need not have same leftmost and rightmost derivation.
(b) An LL(1) parser is not a top-down parser.
(c) LALR is less powerful than SLR.
(d) An ambiguous grammar can be LR(k) for any k.

Solution: Only option (a) is true. LL(1) parser is a top–down parser; LALR is more powerful.

Ans. (a)

4. Given the following expression grammar:

$E \rightarrow E*F|F + E|F$

$F \rightarrow F - F|\text{id}$

Which of the following is true?

(a) * has higher precedence than +
(b) − has higher precedence than *
(c) + and − have same precedence
(d) + has higher precedence than *

Solution: id has higher precedence than any other symbol.

Ans. (b)

5. The number of tokens in the following C statement is

```
printf("i=%d, &i =%x",i&i);
```

(a) 13 (b) 6
(c) 10 (d) 11

Solution: The tokens in C tokens include identifiers, keywords, constants, operators, string literals and other separators. Thus, there are 10 tokens in the given printf statement.

Ans. (c)

6. The identifications of common sub-expression and replacement of run-time computations by compile-time computations is

(a) local optimization
(b) global optimization
(c) constant folding
(d) common fielding

Solution: Constant folding is the process of evaluating the constant expression at compile time.

Ans. (c)

7. In a compiler the module that checks every character of the source text is called

(a) the machine dependant optimizer
(b) the code checker
(c) the lexical analyzer
(d) the syntax analyzer

Solution: Lexical analyzer reads the source program character by character at a time and unites them into a stream of tokens.

Ans. (c)

8. Match the following.

Group 1	**Group 2**
A. Lexical analysis	P. Directed acyclic graphs
B. Code optimization	Q. Syntax trees
C. Code generation	R. PDA
D. Abelian groups	S. Finite automaton

(a) A-S, B-P, C-R, D-Q
(b) A-Q, B-R, C-S, D-P
(c) A-S, B-Q, C-R, D-P
(d) A-P, B-S, C-R, D-Q

Solution: A-S, B-P, C-R, D-Q

Ans. (a)

9. Which of the following strings can definitely be said to be token without looking at the next input character while compiling a Pascal program?

1. Begin
2. Goto
3. <>

(a) 1 (b) 2
(c) 3 (d) All of the above

Solution: Begin and Goto cannot be properly grouped as keywords without considering the delimiter. But <> can be considered as Boolean operator, not matter what follows the symbol.

Ans. (c)

10. Let us suppose, in a programming language an identifier is permitted to be a letter followed by any number of letters or digits. If L and D denote the sets of letters and digits, respectively, which of the following expressions defines an identifier?

(a) $(L.D)^*$ (b) $L(L + D)$
(c) $L + (L.D)^*$ (d) $L(L.D)^*$

Solution: An identifier expression cannot have character such as *.

Ans. (b)

11. Consider LR(1) collections of grammar (G), in state I_0 (i.e. first state) what is the follow set of LR(1) item, $T \to .TF$?

(a) $\$| +$ (b) $\$| + |a|b$
(c) $\$| + |a$ (d) $+ |a|b$

Solution:
I_0:
$E' \to .E, \$$
$E \to .E + T, \$| +$
$E \to .T, \$| +$
$T \to .TF, \$| + |a|b$
$T \to .F, \$| + |a|b$
$F \to .F^*, \$| + |a|b|^*$
$F \to .a, \$| + |a|b|^*$
$F \to .b, \$| + |a|b|^*$

Ans. (b)

12. Which grammar rules violate the requirement of the operator grammar? A, B, C are variables and a, b, c are terminals.

i. $A \to BC$ ii. $A \to CcBb$
iii. $A \to BaC$ iv. $A \to \varepsilon$

(a) i only (b) i and ii
(c) i and iii (d) i and iv

Solution: Operator grammar has two properties: First, there should not be any null production. Second, there should not be any production which has two consecutive variables in the right-hand side of production.

Ans. (d)

13. Consider the following grammar:

$S \to (L)/a$
$L \to SL'$
$L' \to \varepsilon/+SL'$

In the predictive parser table M of the grammar, the entries $M[S, (\,]$ and $M[L',)]$, respectively,

(a) $S \to (L)$ and $L' \to ,SL'$
(b) $S \to a$ and $L' \to \varepsilon$
(c) $S \to (L)$ and $L' \to \varepsilon$
(d) $S \to a$ and $L' \to ,SL'$

Solution:

Parsing table

	A	**,**	**(**	**)**	**$**
S	$S \to a$		$S \to (L)$		
L	$L \to SL'$		$L \to SL'$		
L'		$L' \to ,SL'$		$L' \to \varepsilon$	

Ans. (c)

14. Which is equivalent CFG without useless symbol for the given grammar:

$S \to PQ|p$, $P \to p$

(a) $S \to PQ$ (b) $S \to p$
(c) $S \to P$, $P \to p$ (d) $S \to P|p$, $P \to p$

Solution: P and Q are the useless symbol because Q does not produce anything and P produces p terminal through $P \to p$, which can directly be generated by the starting symbol. So, option (b) is correct, because the given grammar generates terminal p only, which can be generated by $S \to p$.

Ans. (b)

GATE PREVIOUS YEARS' QUESTIONS

1. Which of the following suffices to convert an arbitrary CFG to an LL(1) grammar?

(a) Removing left recursion alone
(b) Factoring the grammar alone
(c) Removing left recursion and factoring the grammar
(d) None of the above

(GATE 2003: 1 Mark)

Solution: To convert a CFG to an LL(1) grammar, left recursion and left factoring must be removed. Otherwise, the grammar becomes ambiguous.

Ans. (c)

2. Assume that the SLR parser for a grammar G has n_1 states and the LALR parser for G has n_2 states. The relationship between n_1 and n_2 is

(a) n_1 is necessarily less than n_2.
(b) n_1 is necessarily equal to n_2.
(c) n_1 is necessarily greater than n_2.
(d) None of the above.

(GATE 2003: 1 Mark)

Solution: States for SLR parser and LALR parser are equal.

Ans. (b)

3. In a bottom-up evaluation of a syntax-directed definition, inherited attributes can

(a) always be evaluated.
(b) be evaluated only if the definition is L attributed.
(c) be evaluated only if the definition has synthesized attributes.
(d) never be evaluated.

(GATE 2003: 1 Mark)

Solution: In a bottom-up evaluation of a syntax-directed definition, inherited attributes can be evaluated using synthesized attributes.

Ans. (c)

4. Which of the following statements is FALSE?

(a) In statically typed languages, each variable in a program has a fixed type.
(b) In untyped languages, values do not have any types.
(c) In dynamically typed languages, variables have no types.
(d) In all statically typed languages, each variable in a program is associated with values of only a single type during the execution of the program

(GATE 2003: 1 Mark)

Solution: In dynamically typed languages such as C, variables have no types.

Ans: (c)

5. Consider the grammar shown below:

$S \rightarrow iEtSS' | a$
$S' \rightarrow eS | \varepsilon$
$E \rightarrow b$

In the predictive parse table, M, of this grammar, the entries $M[S',e]$ and $M[S',\$]$, respectively, are

(a) $\{S' \rightarrow eS\}$ and $\{S'\varepsilon\}$
(b) $\{S' \rightarrow eS\}$ and { }
(c) $\{S' \rightarrow \varepsilon\}$ and $\{S' \rightarrow \varepsilon\}$
(d) $\{S' \rightarrow eS, S' \rightarrow \varepsilon\}$ and $\{S' \rightarrow \varepsilon\}$

(GATE 2003: 2 Marks)

Solution: Predictive parser table is

Non-Terminal A	b	E	i	t	\$
S	$S \rightarrow a$		$S \rightarrow iEiSS'$		
S'		$S' \rightarrow \varepsilon$ $S' \rightarrow eS$			$S' \rightarrow \varepsilon$
E	$E \rightarrow b$				

$M[S', e] = \{S' \rightarrow \varepsilon, S' \rightarrow eS\}$
$M[S', \$] = S' \rightarrow \varepsilon$

Ans. (d)

6. Consider the grammar shown below:

$S \rightarrow CC$
$C \rightarrow cC | d$

This grammar is

(a) LL(1)
(b) SLR(1) but not LL(1)
(c) LALR(1) but not SLR(1)
(d) LR(l) but not LALR(1)

(GATE 2003: 2 Marks)

Solution: The given grammar is LR(1). Every SLR(1) grammar is LR(1) but not every LALR(1) grammar is LR(1).

Ans. (d)

7. Consider the translation scheme shown below:

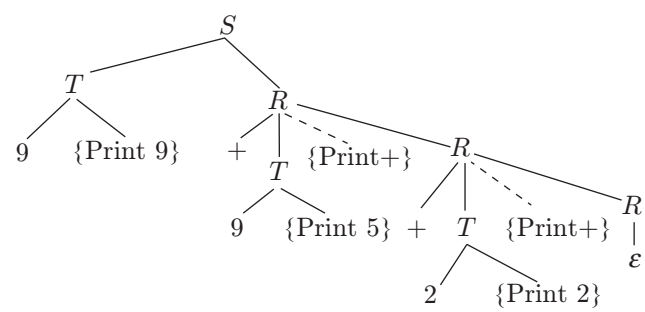
$S \rightarrow TR$
$R \rightarrow +T$ {print('+');} $R|\varepsilon$
$T \rightarrow$ num {print(num.val);}

Here num is a token that represents an integer and num.val represents the corresponding integer value. For an input string '9 + 5 + 2', this translation scheme will print

(a) 9 + 5 + 2
(b) 9 5 + 2 +
(c) 9 5 2 + +
(d) + + 9 5 2

(GATE 2003: 2 Marks)

Solution: Tree for translation scheme



Thus, the translational scheme will print 9 5 + 2 +

Ans. (b)

8. Consider the syntax-directed definition shown below.

$S \rightarrow$ id: $= E$	`{gen(id.place = E.place;);}`

$E \rightarrow E1 + E2$	`{t = newtemp( );`
	`gen(t = E1.place + E2.place;);`
	`E.place = t;}`

$E \rightarrow$ id	`{E.place = id.place;}`

Here, gen is a function that generates the output code, and newtemp is a function that returns the name of a new temporary variable on every call. Assume that t_i's are the temporary variable names generated by newtemp. For the statement '$X := Y + Z$', the 3-address code sequence generated by this definition is

(a) $X = Y + Z$
(b) $t_1 = Y + Z$; $X = t_1$
(c) $t_1 = Y$; $t_2 = t_1 + Z$; $X = t_2$
(d) $t_1 = Y$; $t_2 = Z$; $t_3 = t_1 + t_2$; $X = t_3$

(GATE 2003: 2 Marks)

Solution: Three-address code for $X = Y + Z$ is

$$t_1 = Y$$
$$t_2 = Z$$
$$t_3 = t_1 + t_2$$
$$X = t_3$$

Ans. (d)

9. Consider a program P that consists of two source modules M_1 and M_2 contained in two different files. If M_1 contains a reference to a function defined in M_2, the reference will be resolved at

(a) edit time. (b) compile time.
(c) link time. (d) load time.

(GATE 2004: 1 Mark)

Solution: Modules can be compiled separately, but they are linked to resolve references at link time.

Ans. (c)

10. Which of the following grammar rules violate the requirements of an operator grammar? P, Q, R are non-terminals, and r, s, t are terminals.

(i) $P \rightarrow QR$
(ii) $P \rightarrow QsR$
(iii) $P \rightarrow \varepsilon$
(iv) $P \rightarrow QtRr$

(a) (i) only (b) (i) and (iii) only
(c) (ii) and (iii) only (d) (iii) and (iv) only

(GATE 2004: 1 Mark)

Solution: The rule for operator grammar is that no production drive ε and it should not contain two non-terminals.

$P \rightarrow QR$ and $P \rightarrow \varepsilon$ violate the rules.

Ans. (b)

11. Consider the grammar rule $E - E_1 - E_2$ for arithmetic expressions. The code generated is targeted to a CPU having a single-user register. The subtraction operation requires the first operand to be in the register. If E_1 and E_2 do not have any common sub-expression, in order to get the shortest possible code

(a) E_1 should be evaluated first.
(b) E_2 should be evaluated first.
(c) evaluation of E_1 and E_2 should necessarily be interleaved.
(d) order to evaluation of E_1 and E_2 is of no consequence.

(GATE 2004: 1 Mark)

Solution: If we evaluate E_2 first then we can save this to memory location and evaluation of E_1 is kept in registers only. But if E_1 is solved first, it should be saved to memory. Then evaluate E_2 and save to memory. Both need to be fetched again. So, option (b) is the optimized answer.

Ans. (b)

12. Consider the grammar with the following translation rules and E as the start symbol.

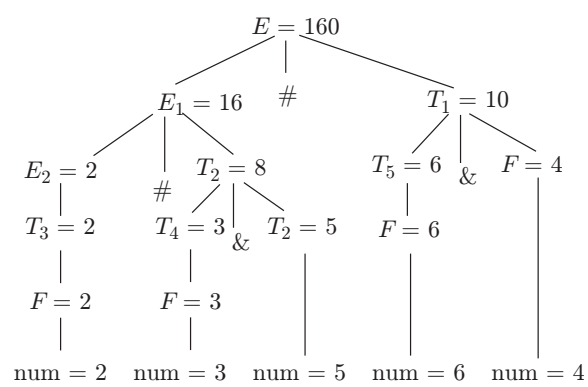
$E \rightarrow E_1 \# T$ {E.value = E_1.value * T.value}
$|T$ {E.value = T.value}
$T \rightarrow T_1$ & F {T.value = T_1.value + F.value}
$|F$ {T.value = F.value}
$F \rightarrow$ num {F.value = num.value}

Compute E.value for the root of the parse tree for the expression: 2 # 3 & 5 # 6 & 4.

(a) 200 (b) 180
(c) 160 (d) 40

(GATE 2004: 2 Marks)

Solution: The parse tree is



Ans. (c)

13. The grammar $A \rightarrow AA|(A)|\varepsilon$ is not suitable for predictive parsing because the grammar is

(a) ambiguous.
(b) left recursive.
(c) right recursive.
(d) an operator grammar.

(GATE 2005: 1 Mark)

Solution: The given grammar is ambiguous because it have more than one parse tree for sting "( )".

Ans. (a)

14. Consider the grammar:

$E \rightarrow E + n | E \times n | n$

For a sentence $n + n \times n$, the handles in the right-sentential form of the reduction are

(a) n, $E + n$ and $E + n \times n$
(b) n, $E + n$ and $E + E \times n$
(c) n, $n + n$ and $n + n \times n$
(d) n, $E + n$ and $E \times n$

(GATE 2005: 2 Marks)

Solution:
$n + n \times n$

Using $E \rightarrow n$	$E + n \times n$
Using $E \rightarrow E + n$	$E \times n$
Using $E \rightarrow E \times n$	E

Handles are n, $E + n$, $E \times n$.

Ans. (d)

15. Consider the grammar:

$S \rightarrow (S) | a$

Let the number of states in SLR(1), LR(1) and LALR(1) parsers for the grammar be n_1, n_2 and n_3, respectively. The following relationship holds good:

(a) $n_1 < n_2 < n_3$ (b) $n_1 = n_3 < n_2$
(c) $n_1 = n_2 = n_3$ (d) $n_1 \geq n_3 \geq n_2$

(GATE 2005: 2 Marks)

Solution: Number of states: SLR = LALR < LR.

Ans. (b)

Linked Answer Questions 16 and 17: Consider the following expression grammar. The semantic rules for expression evaluation are stated next to each grammar production.

$E \rightarrow$ number	E.val = number.val
$\mid E$ '+' E	$E(1)$.val = $E(2)$.val + $E(3)$.val
$\mid E$ '×' E	$E(1)$.val = $E(2)$.val × $E(3)$.val

16. The above grammar and the semantic rules are fed to a yacc tool (which is an LALR(1) parser generator) for parsing and evaluating arithmetic expressions. Which one of the following is true about the action of yacc for the given grammar?

(a) It detects recursion and eliminates recursion.
(b) It detects reduce-reduce conflict, and resolves.
(c) It detects shift-reduce conflict, and resolves the conflict in favour of a shift over a reduce action.
(d) It detects shift-reduce conflict, and resolves the conflict in favour of a reduce over a shift action.

(GATE 2005: 2 Marks)

Solution:

$E \rightarrow$ number $| E$ '+' $E | E$ '×' E

yacc compiler detects shift-reduce conflict, and resolves the conflict in favour of a shift over a reduce action.

Ans. (c)

17. Assume the conflicts in Part (a) of this question are resolved and an LALR(1) parser is generated for parsing arithmetic expressions as per the given grammar. Consider an expression $3 \times 2 + 1$. What precedence and associativity properties does the generated parser realize?

(a) Equal precedence and left associativity; expression is evaluated to 7.
(b) Equal precedence and right associativity; expression is evaluated to 9.
(c) Precedence of '×' is higher than that of '+', and both operators are left associative; expression is evaluated to 7.
(d) Precedence of '+' is higher than that of '×', and both operators are left associative; expression is evaluated to 9.

(GATE 2005: 2 Marks)

Solution: X has higher precedence than +. So 3×2 will be solved first. Hence, $3 \times 2 = 6 + 1 = 7$.

Ans. (c)

18. Consider the following grammar:

$S \rightarrow S^*E$
$S \rightarrow E$
$E \rightarrow F + E$
$E \rightarrow F$
$F \rightarrow$ id

Consider the following LR(0) items corresponding to the grammar above.

(i) $S \rightarrow S^*.E$
(ii) $S \rightarrow F. + E$
(iii) $S \rightarrow F + .E$

Given the items above, which two of them will appear in the same set in the canonical sets of items for the grammar?

(a) (i) and (ii) (b) (ii) and (iii)
(c) (i) and (iii) (d) None of the above

(GATE 2006: 1 Mark)

Solution: If $S \rightarrow S^*.E$ is in LR(0) then $E \rightarrow F + .E$ will also be there because both of them have '.' (dot) before E.

Ans. (c)

19. Consider the following translation scheme:

$S \rightarrow ER$
$R \rightarrow {}^*E\{\text{print}(\text{'*'});\}R | \varepsilon$

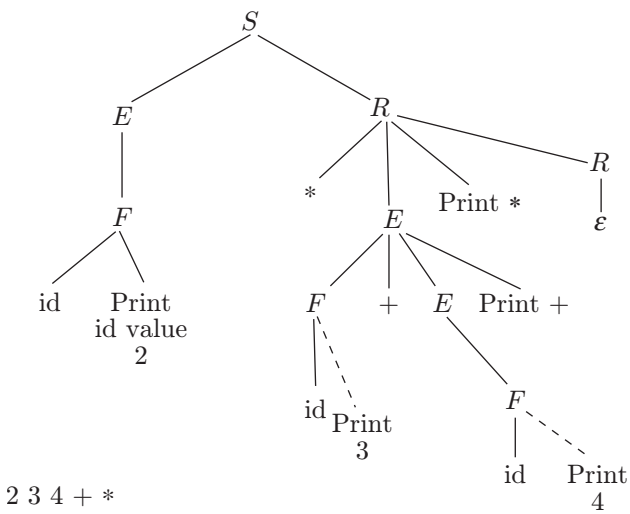
$E \rightarrow F + E\{\text{print}(+);\}|F$
$F \rightarrow (S)\text{id}\{\text{print}(\text{id.value});\}$

Here id is a token that represents an integer and id.value represents the corresponding integer value. For an input 2*3 + 4, this translation scheme prints

(a) 2*3 + 4 (b) 2* + 34
(c) 23*4 (d) 234 + *

(GATE 2006: 2 Marks)

Solution:
Parse tree is:



2 3 4 + *

Ans. (d)

20. Consider the following grammar:

$S \rightarrow FR$
$R \rightarrow {}^*S|\varepsilon$
$F \rightarrow \text{id}$

In the predictive parser table, M, of the grammar, the entries $M[S, \text{id}]$ and $M[R, \$]$, respectively,

(a) $\{S \rightarrow FR\}$ and $\{R \rightarrow S\}$
(b) $\{S \rightarrow FR\}$ and $\{\ \}$
(c) $\{S \rightarrow FR\}$ and $\{R \rightarrow {}^*S\}$
(d) $\{F \rightarrow \text{id}\}$ and $\{R \rightarrow \varepsilon\}$

(GATE 2006: 2 Marks)

Solution:
Parsing table

Non-Terminal	id	*	$
S	$S \rightarrow FR$		
R		$R \rightarrow {}^*S$	$R \rightarrow \varepsilon$
F	$F \rightarrow \text{id}$		

From this table, we find option (a) is correct.

Ans. (a)

21. Which one of the following is a top-down parser?

(a) Recursive descent parser
(b) Operator precedence parser
(c) An LR(k) parser
(d) An LALR(k) parser

(GATE 2007: 1 Mark)

Solution: Recursive decent parsing is top-down parsing.

Ans. (a)

22. Consider the grammar with non-terminals $N = \{S, C, S_1\}$, terminals $T = \{a, b, i, t, e\}$, with S as the start symbol, and the following set of rules:

$S \rightarrow iCtSS_1|a$
$S_1 \rightarrow eS|\varepsilon$
$C \rightarrow b$

The grammar is NOT LL(1) because

(a) it is left recursive. (b) it is right recursive.
(c) it is ambiguous. (d) it is not context-free.

(GATE 2007: 2 Marks)

Solution: Two parse trees are possible for the above grammar. So, the given grammar is ambiguous.

Ans. (c)

23. Consider the following two statements:

A. Every regular grammar is LL(1).
B. Every regular set has an LR(1) grammar.

Which of the following is TRUE?

(a) Both A and B are true.
(b) A is true and B is false.
(c) A is false and B is true.
(d) Both A and B are false.

(GATE 2007: 2 Marks)

Solution: A left recursive grammar is not LL(1).

Ans. (c)

24. Which of the following describes a handle (as applicable to LR parsing) appropriately?

(a) It is the position in a sentential form where the next shift or reduce operation will occur.
(b) It is non-terminal whose production will be used for reduction in the next step.
(c) It is a production that may be used for reduction in a future step along with a position in the sentential form where the next shift or reduce operation will occur.
(d) It is the production p that will be used for reduction in the next step along with a position

in the sentential form where the right-hand side of the production may be found.

(GATE 2008: 1 Mark)

Solution: Fact

Ans. (d)

25. Some code optimizations are carried out on the intermediate code because

(a) They enhance the portability of the compiler to other target processors.
(b) Program analysis is more accurate on intermediate code than on machine code.
(c) The information from dataflow analysis cannot otherwise be used for optimization.
(d) The information from the front end cannot otherwise be used for optimization.

(GATE 2008: 1 Mark)

Solution: Fact

Ans. (b)

26. An LALR(1) parser for a grammar G can have shift-reduce (S-R) conflicts if and only if

(a) The SLR(1) parser for G has S-R conflicts.
(b) The LR(1) parser for G has S-R conflicts.
(c) The LR(0) parser for G has S-R conflicts.
(d) The LALR(1) parser for G has reduce-reduce conflicts.

(GATE 2008: 2 Marks)

Solution: The LR(1) parser for G has Shift-Reduce conflicts.

Ans. (b)

27. Which of the following are true?

I. A programming language which does not permit global variables of any kind and has no nesting of procedures/functions, but permits recursion can be implemented with static storage allocation.
II. Multi-level access link (or display) arrangement is needed to arrange activation records only if the programming language being implemented has nesting of procedures/functions.
III. Recursion in programming languages cannot be implemented with dynamic storage allocation.
IV. Nesting procedures/functions and recursion require a dynamic heap allocation scheme and cannot be implemented with a stack-based allocation scheme for activation records.
V. Programming languages which permit a function to return a function as its result cannot be implemented with a stack-based storage allocation scheme for activation records.

(a) II and V only (b) I, III and IV only
(c) I, II and V only (d) II, III and V only

(GATE 2008: 2 Marks)

Solution:

I: Multi-level access link (or display) arrangement is needed to arrange activation records only if the programming language being implemented has nesting of procedures/functions is true.
V: Programming languages which permit a function to return a function as its result cannot be implemented with a stack-based storage allocation scheme for activation records is true.

Ans. (a)

28. Match all items in Group 1 with correct options from those given in Group 2.

Group 1	**Group 2**
P. Regular expression	1. Syntax analysis
Q. Pushdown automata	2. Code generation
R. Dataflow analysis	3. Lexical analysis
S. Register allocation	4. Code optimization

(a) P-4, Q-1, R-2, S-3 (b) P-3, Q-1, R-4, S-2
(c) P-3, Q-4, R-1, S-2 (d) P-2, Q-1, R-4, S-3

(GATE 2009: 1 Mark)

Solution: Regular expressions: Lexical analysis

Pushdown automata: Syntax analysis
Dataflow analysis: Code optimization
Register allocation: Code optimization

Ans. (b)

29. Which of the following statements are TRUE?

A. There exist parsing algorithms for some programming languages whose complexities are less than $\Theta(n^3)$.
B. A programming language which allows recursion can be implemented with static storage allocation.
C. No L-attributed definition can be evaluated in the framework of bottom-up parsing.
D. Code improving transformations can be performed at both source language and intermediate code level.

(a) A and B (b) A and D
(c) C and D (d) A, C and D

(GATE 2009: 2 Marks)

Solution: Statements A and D are true.

Ans. (b)

30. Which data structure in a compiler is used for managing information about variables and their attributes?

(a) Abstract syntax tree (b) Symbol table
(c) Semantic stack (d) Parse table

(GATE 2010: 1 Mark)

Solution: Symbol table is a data structure used by a compiler to store variables and their attributes.

Ans. (b)

31. The grammar $S \rightarrow aSa|bS|c$ is

(a) LL(1) but not LR(1)
(b) LR(1) but not LR(1)
(c) Both LL(1) and LR(1)
(d) Neither LL(1) nor LR(1)

(GATE 2010: 2 Marks)

Solution: The given grammar is regular and LR(1). It is also LL(1) as multiple entries for one iteration are present.

Ans. (c)

32. In a compiler, keywords of a language are recognized during

(a) parsing of the program.
(b) the code generation.
(c) the lexical analysis of the program.
(d) dataflow analysis.

(GATE 2011: 1 Mark)

Solution: Tokens are recognized in lexical analysis phase of compiler.

Ans. (c)

33. The lexical analysis for a modern computer language such as Java needs the power of which one of the following machine models in a necessary and sufficient sense?

(a) Finite state automata
(b) Deterministic pushdown automata
(c) Non-deterministic pushdown automata
(d) Turing machine

(GATE 2011: 1 Mark)

Solution: Lexical analyzer is implemented using finite state automata.

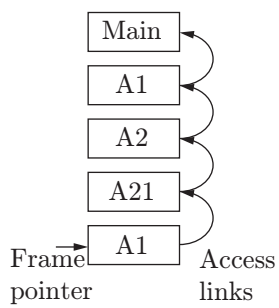
Ans. (a)

34. Consider the program given below, in a block-structured pseudo-language with lexical scoping and nesting of procedures permitted.

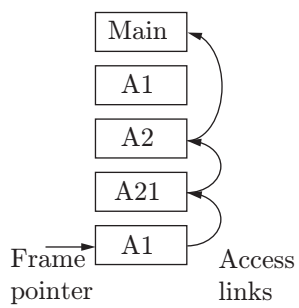
```
Program main;
      Var ...
      Procedure A1;
            Var ...
            Call A2;
      End A1
      Procedure A2;
            Var ...

            Procedure A21;
                  Var ...
                  Call A1;
            End A21

            Call A21;
      End A2

      Call A1;
End main.
```

Consider the calling chain: Main → A1 →A2 → A21 → A1
The correct set of activation records along with their access links is given by

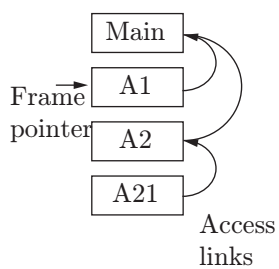
(a)



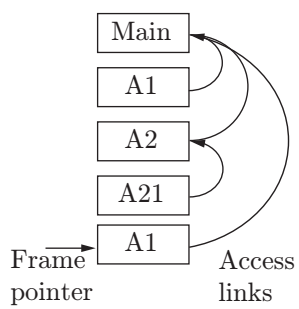
(b)



(c)



(d)



(GATE 2012: 2 Marks)

Solution: Access link is defined as link to activation record of closest lexically enclosing block in program text, so the closest enclosing blocks representing for A1, A2 and A21 are main, main and A2.

Ans. (d)

Linked Answer Questions 35 and 36: For the grammar below, a partial LL(1) parsing table is also presented along with the grammar. Entries that need to be filled are indicated as E_1, E_2, and E_3. ε is the empty string, \$ indicates end of input, and, | separates alternate right-hand sides of productions.

$S \to aAbB|bAaB|\varepsilon$
$A \to S$
$B \to S$

	A	**B**	**\$**
S	E_1	E_2	$S \to \varepsilon$
A	$A \to S$	$A \to S$	Error
B	$B \to S$	$B \to S$	E_3

35. The FIRST and FOLLOW set for the non-terminals A and B are

(a) FIRST(A) = $\{a, b, \varepsilon\}$ = FIRST(B)
(b) FOLLOW(A) = $\{a, b\}$
(c) FOLLOW(B) = $\{a, b, \$\}$
(d) FIRST(A) = $\{a, b, \$\}$
(e) FIRST(B) = $\{a, b, \varepsilon\}$
(f) FOLLOW(A) = $\{a, b\}$
(g) FOLLOW(B) = $\{\$\}$
(h) FIRST(A) = $\{a, b, \varepsilon\}$ = FIRST(B)
(i) FOLLOW(A) = $\{a, b\}$
(j) FOLLOW(B) = $\varnothing$
(k) FIRST(A) = $\{a, b, \varepsilon\}$ = FIRST(B)
(l) FOLLOW(A) = $\{a, b\}$
(m) FOLLOW(B) = $\{a, b\}$

(GATE 2012: 2 Marks)

Solution: First(A) gives set of all terminals that begin in string derived from A, and Follow(A) means it will consist set of all terminals that may follow immediately to the right of A.

First(A) = $\{a, b, \varepsilon\}$
First(B) = $\{a, b, \varepsilon\}$
Follow(A) = $\{a, b\}$
Follow(B) = $\{a, b, \$\}$

Ans. (a)

36. The appropriate entries for E_1, E_2 and E_3 are

(a) E_1:$S \to aAbB$, $A \to S$
(b) E_2:$S \to bAaB$, $B \to S$
(c) E_3:$B \to S$
(d) E_1:$S \to aAbB$, $S \to \varepsilon$
(e) E_2:$S \to bAaB$, $S \to \varepsilon$
(f) E_3:$S \to \varepsilon$
(g) E_1:$S \to aAbB$, $S \to \varepsilon$
(h) E_2:$S \to bAaB$, $S \to \varepsilon$
(i) E_3:$B \to S$
(j) E_1:$A \to S$, $S \to \varepsilon$
(k) E_2:$B \to S$, $S \to \varepsilon$
(l) E_3:$B \to S$

(GATE 2012: 2 Marks)

Solution:

First(S) = $\{a, b, \varepsilon\}$ and Follow(S) = $\{a, b, \$\}$
First(B) = $\{a, b, \varepsilon\}$ and Follow(B) = $\{a, b, \$\}$
So,

	a	**B**	**ε**
S	$S \to aAbB$	$S \to bAaB$	$S \to \varepsilon$
	$S \to \varepsilon$	$S \to \varepsilon$	
B	$B \to S$	$B \to S$	$B \to S$

Ans. (c)

37. What is the maximum number of reduce moves that can be taken by a bottom-up parser for a grammar with no epsilon- and unit-production (i.e., of type $A \to \varepsilon$ and $A \to a$) to parse a string with n tokens?

(a) $n/2$ (b) $n - 1$
(c) $2n - 1$ (d) $2n$

(GATE 2013: 1 Mark)

Solution: $n - 1$ moves can be taken by bottom up parser.

Ans. (b)

38. Consider the following two sets of LR(1) items of an LR(1) grammar:

$X \to c.X$, c/d
$X \to .cX$, c/d
$X \to .d$, c/d
$X \to c.X$, \$
$X \to .cX$, \$
$X \to .d$, \$

Which of the following statements related to merging of the two sets in the corresponding LALR parser is/are FALSE?

A. Cannot be merged since look aheads are different.
B. Can be merged but will result in S-R conflict.
C. Can be merged but will result in R-R conflict.
D. Cannot be merged since goto on c will lead to two different sets.

(a) A only (b) B only
(c) A and D only (d) A, B, C and D

(GATE 2013: 1 Mark)

Solution: All the statements are false.

Ans. (d)

PRACTICE EXERCISES

Set 1

1. Type checking is normally done during

(a) parsing
(b) syntax analysis
(c) syntax-directed translation
(d) semantic-based optimization

2. Consider the following grammar:

$S \rightarrow m|mn|mno$

Choose the correct statement from the following:

(a) The grammar is LL(1).
(b) The grammar is LL(2).
(c) The grammar is LL(3).
(d) None of the above.

3. A shift-reduce parser carries out the actions specified within braces immediately after reducing with the corresponding rule of grammar.

$S \rightarrow xx\ W$ {print "1"}
$S \rightarrow y$ {print "2"}
$W \rightarrow Sz$ {print "3"}

What is the translation of *xxxxyzz* using the syntax-directed translation scheme described by the above rules?

(a) 23131 (b) 11233
(c) 11231 (d) 33211

4. The pass numbers for each of the following activities

1. Object code generation
2. Literals added to literal table
3. Listing printed
4. Address resolution of local symbols that occur in a two-pass assembler, respectively, are

(a) 2, 2, 1, 2 (b) 2, 1, 2, 1
(c) 2, 1, 1, 2 (d) 1, 2, 2, 2

5. Which of the following statements is true?

(a) LALR parser is more powerful than canonical LR parser.
(b) LALR parser cannot be compared with LALR parser.
(c) Canonical LR parser is more powerful than LALR parser.
(d) The parsers canonical LR and LALR have the same power.

6. Consider the given grammar:

$S' \rightarrow S$
$S \rightarrow aXd|bYd|aYe|bXe$
$X \rightarrow c$
$Y \rightarrow c$

The grammar is

(a) LR(1) but not LALR(1)
(b) LALR(1) but not LL(1)
(c) SLR(1) but not LR(0)
(d) None of the above

7. Which of the following statements is true?

(a) An unambiguous grammar does not have the same leftmost and rightmost derivation.
(b) An LL(1) parser is a left-right parser.
(c) LALR is more as powerful as SLR.
(d) An ambiguous grammar can be defined as LR(k) for any k.

8. Generation of intermediate code based on an abstract machine model is useful in compilers because

(a) It makes implementation of lexical analysis and syntax analysis easier.
(b) Syntax translations are easier for intermediate code generation.
(c) It enhances the portability of the compiler system program.
(d) It is difficult to generate executable code from high-level language programs.

9. A linker is given object modules for a set of programs that were compiled separately. What information needs to be included in an object module?

(a) Version of the compiler being used
(b) Relocation code
(c) Symbol table and various macros
(d) Absolute addresses of internal symbols

10. In the following grammar

$X ::= X\ \Theta\ Y/Y$
$Y ::= Z\ \mathrm{O}\ Y/Z$
$Z ::= \mathrm{id}$

Which of the following is true?

(a) 'Θ' is left associative while 'O' is right associative.
(b) It cannot be defined as the rules defined are few.
(c) 'Θ' is right associative while 'O' is left associative.
(d) Both 'Θ' and 'O' are right associative.

11. Which of the following statements are true/false, map them appropriately, with respect to syntax-directed definitions?

A. The terminals are defined for inherited attributes.
B. Value of attributes of terminals is generally supplied by lexical analyzer.
C. The start symbol does not have an inherited attribute.
D. Attribute grammar is used to define the special cases of the attributes.

(a) TTFT (b) TFTF
(c) FFTF (d) FTTF

12. Which of the following statements are true?

A. All unambiguous grammars are regular.
B. Every LL(K) and LR(K) grammar is unambiguous.
C. A regular language can never be inherently ambiguous.
D. A regular grammar contains special string for syntax-directed definitions.

(a) A and D only (b) B and C only
(c) A and B only (d) C and D only

13. What is the maximum number of reduce moves that can be taken by a bottom-up parser for a grammar with no epsilon and unit production (i.e., of type $A \rightarrow \varepsilon$ and $A \rightarrow a$) to parse a string with n tokens?

(a) $n + 1/2$ (b) $n - 1$
(c) n (d) $\sqrt{n}$

14. The grammar $S \rightarrow aSa|bS|c$ can be defined as

(a) It is only LR(1).
(b) It is only LL(1).
(c) Both LL(1)and LR(1).
(d) Neither LL(1) nor LR(1).

15. Match all items in Group 1 with correct options from those given in Group 2.

Group 1	**Group 2**
P. Regular expression	1. Lexical analysis
Q. Register allocation	2. Code generation
R. Dataflow analysis	3. Syntax analysis
S. Pushdown automata	4. Code optimization

(a) P-1, Q-2, R-3, S-4 (b) P-1, Q-2, R-4, S-3
(c) P-3, Q-1, R-2, S-4 (d) P-4, Q-3, R-2, S-1

16. Which of the following statements are true?

(a) A programming language which allows loops can be implemented with static storage allocation.
(b) There exists parsing algorithms for some programming languages whose complexities are less than $O(n^3)$.
(c) No L-attributed definition can be evaluated in the framework of bottom-up parsing.
(d) Some undefined symbols are also defined in the symbol table.

17. Which of the following describes a handle (as applicable to LR parsing) appropriately?

(a) It is the production p in a sentential form where the sentence begins and reduce operation occurs.
(b) It is the production p used for reduction in the next step and the next shift operation will occur.
(c) It is a production p that may be used for reduction in a future step along with a position in the sentential form where the next shift or reduce operation will occur.
(d) It is the production p that will be used for reduction in the next step along with a position in the sentential form where the right hand side of the production may be found.

18. An LALR(1) parser for a grammar G can have shift-reduce (S-R) conflicts if and only if

(a) The LR(1) parser for G has no S-R conflicts.
(b) The LR(1) parser for G has S-R conflicts.
(c) The SLR(1) parser for G has S-R conflicts.
(d) The LALR(1) parser for G has reduce-reduce conflicts.

19. Which one of the following is a top-down parser?

(a) Recursive descent parser
(b) Shift left associative parser
(c) SLR(k) parser
(d) LR(k) parser

20. Which of the following is true?

(a) Every regular grammar is not LL(1) and every regular set does not have an LR(1) grammar.
(b) Every regular grammar is LL(1) and every regular set does not have an LR(1) grammar.
(c) Every regular grammar is not LL(1) and every regular set has an LR(1) grammar.
(d) Every regular grammar is LL(1) and every regular set has an LR(1) grammar.

21. Some functions of code optimizations are carried out on the intermediate code of the compiler because

(a) They enhance the portability of the compiler to other target computer.
(b) Program analysis is more accurate and runs faster.
(c) It saves time in performing dataflow analysis.
(d) Provides a better understanding of the object file.

22. Which one of the following is FALSE?

(a) A simple basic block enters the sequence at the beginning and exits at the end.
(b) Expression analysis can be used for common sub-expression elimination.
(c) Dead-code elimination can be done by live variable analysis.
(d) $x = 4 * 5 \Rightarrow x = 20$ is an example of common sub-expression elimination.

23. One of the purposes of using intermediate code in compilers is to

(a) Improve memory and stack management.
(b) Improve detection of syntax and semantic errors.
(c) Improve the reusing of the machine-independent code optimizer.
(d) Improve the register allocation and cache management.

24. Which of the following statement(s) is/are true about LALR(1) parsers?

S_1: LALR(1) parsers have same number of states as SLR(1) parsers (core LR(0) items are the same).
S_2: LALR(1) is derived from LR(1) with no shift-reduce conflict will also have no shift-reduce conflict.
S_3: LALR(1) may create reduce-reduce conflict which was not in LR(1) from which LALR(1) is derived.

(a) None of the statement is true
(b) S_1 and S_3
(c) S_1 and S_2
(d) S_1, S_2 and S_3

25. Which of the following suffices to convert an arbitrary CFG to an LL(1) grammar?

(a) Removing left recursion alone and right associative productions
(b) Removing the grammar and the productions but not the start symbol
(c) Factoring the grammar and removing the left associative productions
(d) None of the above

26. The major limitation of a shift reduce parser is to overcome

(a) Representation of productions in top-down parser
(b) Reduce-reduce conflict only
(c) Both shift reduce conflict and reduce-reduce conflict
(d) Shift conflicts only

27. The compiler type that converts all operands to the type of the largest operand is called

(a) Type promotion (b) Type evaluation
(c) Type bootstrap (d) Type operand testing

28. High-level knowledge which relates to the use of sentences in different contexts and how the context affect the meaning of the sentences is called

(a) Pedagogy (b) Syntactic
(c) Morphological (d) Pragmatic

29. The process adjusting the code and data in the program to reflect the assigned addresses is called _____.

(a) Assembly (b) Linking
(c) Absolute addressing (d) Relocation

30. A top-down parser while parsing an input string from left to right makes use of:

(a) Leftmost derivation
(b) Topmost derivation
(c) Rightmost derivation traced out in reverse
(d) Bottom-up derivation traced in reverse

31. Which of the following statement(s) is/are true?

S_1: If G is unambiguous, then every right-sentential form has a unique handle.
S_2: Shift-reduce parsers use a stack and input buffer.

(a) Neither S_1 nor S_2 (b) S_1 only
(c) Both S_1 and S_2 (d) S_2 only

32. Which of the following is/are type(s) of tools used in various phases of compiler?

(a) Control flow graph, binary sort, cyclic graphs
(b) Three-address code, bootstrap loader, cyclic graphs
(c) Directed acyclic graph, assembler, binary linker
(d) Control flow graph, three-address code, directed acyclic graph

33. Which of the following is true?

(a) If a grammar is LL(k), it is also ~(LR($k-1$)).
(b) If a grammar is not LR(k), it is ~(SLR(k)).
(c) If a grammar is CLR(k), it is also SLR(k).
(d) If a grammar is not LALR(k), it is not SLR(k).

34. Consider the following grammar:

$E \rightarrow AB$
$A \rightarrow +BA/\varepsilon$
$B \rightarrow B * A/\text{id}$

Then which of the following is true?
(a) Both +, * are left associative.
(b) Only * is left associative.
(c) Only + is left associative.
(d) Both +, * are not left associative.

35. The type of conflicts that can rise in LR(0) techniques are

(a) Shift-right only
(b) Reduce-reduce only
(c) Shift-left only
(d) Both shift-reduce and reduce-reduce

36. Shift-reduce parsers are

(a) Shift parsers (b) Bottom-up parsers
(c) Predictive parsing (d) Reducing parsers

37. Consider the following grammar:

$S \rightarrow Aa|bAc|Bc|bBa$
$A \rightarrow x$, $B \rightarrow x$

(a) The grammar is LR(1) but not LALR(1)
(b) LALR(1) but not LL(1)
(c) SLR(1) but not LR(0)
(d) None of the above

38. Which of the following statements is false?

(a) An unambiguous grammar has same leftmost and rightmost derivation.
(b) An LL(1) parser is a top-down parser.
(c) LALR is more powerful than SLR.
(d) An ambiguous grammar can never be LR(k) for any k.

39. Consider the grammar given below:

$A \rightarrow AA|a|\varepsilon$

String '$a\ \ aaa$' has

(a) A unique derivation tree in the grammar
(b) Four leftmost derivations in the grammar
(c) Four rightmost derivations in the grammar
(d) An infinite number of leftmost and rightmost derivations

40. Consider the grammar R:

$E \rightarrow E + E|E*E|(E)|\text{id}$

Consider the other grammar S:

$E \rightarrow E + T|T$
$T \rightarrow T*F|F$
$F(E)|\text{id}$

Choose the incorrect statement.

(a) S generates the same language as R.
(b) R is ambiguous but not LL(1) as it is left recursive.
(c) S is SLR(1).
(d) S gives a higher precedence to + over *.

41. What is the maximum number of reduce moves that can be taken by a bottom-up parser for a grammar with no epsilon and unit production (i.e. of type $A \rightarrow$? and $A \rightarrow a$) to parse a string with n tokens?

(a) $n + 1$ (b) $n - 1$ (c) n (d) 2^n

42. The following grammar is

$S \rightarrow Aa|bAc|dc|bda$
$A \rightarrow d$
(a) Not SLR(1) and not CLR(1)
(b) CLR(1) but not SLR(1)
(c) SLR(1) but not CLR(1)
(d) SLR(1) and CLR(1)

43. Write true (T)/false (F) for each of the following statements:

A. Symbol table is used only in first three phases of a compiler.
B. A pretty printer analyses a program and prints it in such a way that the structure of the program becomes clearly visible.
C. YACC build up LALR parsing table.
D. If a grammar is LALR(1), it is not necessarily SLR(1).

(a) TTFT (b) TTTF (c) FTTF (d) FTTT

44. Consider two binary operators * and ^ with precedence of operator * being lower than of ^. Operator * is left associative while operator ^ is right associative. The value of 3*2^3*4 is _____.

45. Consider the following grammar:

$S \rightarrow aB|aAb$
$A \rightarrow bAb|a$
$B \rightarrow aB|\varepsilon$

How many back tracks are required to generate the string 'aab' from the above grammar?

(a) Nil (b) 2 (c) 1 (d) 3

46. Which of the following statements are TRUE?

(a) There does not exist any parsing algorithm which can parse in parallel.
(b) A programming language which does not use loops makes use of static storage allocation.
(c) No L-attributed definition can be evaluated in the framework of bottom-up parsing.
(d) Code improving transformations can be performed at both the source language as well as the intermediate code level.

47. Choose the correct one from the following

(a) An SLR(1) grammar is always LL(0).
(b) A grammar that is LR(0) is necessarily LL(1).
(c) A grammar that is LR(0) may not be LL(1).
(d) None of the above.

48. Consider the grammar shown below:

$T \rightarrow PP$
$P \rightarrow sP/r$

The grammar is

(a) LL(1)
(b) SLR(1) but not LL(1)
(c) LALR(1) but not SLR(1)
(d) LR(1) but not LALR(1)

49. Which statement is false about parsers?

(a) Top-down parsing algorithms use a leftmost derivation.
(b) Bottom-up parsing algorithms use the reverse of a rightmost derivation.
(c) An LR(1) parser may sometimes exist for some ambiguous grammars.
(d) An LL(1) parser is a top-down parser.

50. Consider the following grammar:

$L \rightarrow eLsL|sLeL|\varepsilon$

Choose the correct statement:

(a) The string '*eses*' has two parse trees
(b) The grammar is unambiguous.
(c) The grammar is LL(1) and LR(1).
(d) None of the above.

Set 2

1. The number of tokens in the following C statements:

```
printf ("a = %d, b = %d", a, b);
```

(a) 8 (b) 9 (c) 10 (d) 11

2. Consider the grammar with the following translation rules and S as the start symbol.

$S \rightarrow S$ @ T {S.value = S.value * T.value}
$\quad |T$ {S.value = T.value}
$T \rightarrow T$ \$ F {T.value = T_1.value + F.value}
$\quad |F$ {T.value = F.value}
$F \rightarrow$ num {F.value = num.value}

Compute S.value for the root of the parse tree for the expression: 2 @ 8 \$ 5 @ 6 \$ 4.

(a) 50 (b) 180 (c) 260 (d) 340

3. Consider the following grammar:

$S \rightarrow WX|Y$
$W \rightarrow wWx|wx$
$X \rightarrow yXz|yz$
$Y \rightarrow wYz|wZz$
$Z \rightarrow xZy|xy$

Which language (L) is generated by the grammar?

(a) $\{w^n x^n y^n z^n: n \geq 1\}$
(b) $\{w^n x^n y^m z^m: n > 1, m > 1\}$
(c) $\{w^n x^n y^m z^m: n \geq 1, m \geq 1\} \cup \{w^n x^m y^m z^n: n \geq 1, m \geq 1\}$
(d) $\{w^m x^n y^o z^p: m, n, o, p > 1\}$

4. How many parse trees can be generated for the string '*wwxxyyzz*' from the grammar given in the previous question?

(a) 4 (b) 3 (c) 2 (d) 1

5. The grammar $E \rightarrow EE$, $E \rightarrow (E)$, $E \rightarrow$ **( )**, $E \rightarrow t$ is

(a) Not LL(1) as it is left recursive
(b) Not LL(1) as it is ambiguous
(c) Not LR(1) as it is left recursive
(d) None of the above

6. Which one of the following string can definitely be said to be a token without looking at the next input char?

(a) < (b) > (c) + (d) *

7. Consider the grammar:

$E \rightarrow TE'$
$E' \rightarrow \varepsilon/+TE'$
$T \rightarrow FT'$
$T' \rightarrow \varepsilon/*FT'$
$F \rightarrow a/[E]$

7a. Find out the First( ) of variable T?

(a) a, [(b) a, [,]
(c) +, ε (d) *, ε

7b. Find out First( ) of variable E'?

(a) a, [(b) a, [,]
(c) +, ε (d) *, ε

7c. What will be the Follow( ) of E?

(a) +, * (b) a, [
(c) \$,] (d) \$, [

7d. What will be the Follow( ) of T?

(a) +, *, a (b) +, a, [
(c) *, \$,] (d) +, \$,]

8. Which of the following statements is incorrect?

(a) LALR is more powerful than SLR.
(b) An LL(1) parser uses top-down parsing approach.
(c) An unambiguous grammar has the same leftmost and rightmost derivation.
(d) An ambiguous grammar can never be LR(k) for any k.

9. Assume that the LALR parser for a grammar G_1 has n_1 states and the CLR parser for G_2 has n_2 states. The relationship between n_1 and n_2 is

(a) $n_1 > n_2$ (b) $n_1 \geq n_2$
(c) $n_1 < n_2$ (d) $n_1 \leq n_2$

10. Which of the following grammar (G) is an operator grammar?

(a) $\{S \to SS|Sa|Sb|a\}$
(b) $\{S \to AB|\varepsilon,\ A \to a,\ B \to b\}$
(c) $\{S \to aA,\ A \to a|b|\varepsilon\}$
(d) $\{S \to AaS|b,\ A \to a\}$

11. Consider the following grammar (G):

$E \to E + T|T$
$T \to TF|F$
$F \to F^*|a|b$

The above given grammar is

(a) Not LR(0) due to S-R problem
(b) Not LR(0) due to R-R problem
(c) LR(0)
(d) None of the above

12. Consider the above grammar. If a grammar is LL(1) then

(a) It will be LR(0). (b) It will be SLR(1).
(c) It will be LALR(1). (d) None of the above.

13. Consider the CFG given below with $\{S,\ X,\ Y\}$ non-terminal and $\{a,\ b\}$ terminals. S is the starting symbol and set of production rules are

$S \to bX|aB$
$A \to a|aS|bAA$
$B \to b|bS|aBB$

Which of the following strings is generated by the grammar?

(a) $aabbab$ (b) $abbbab$
(c) $bbaaa$ (d) $aabbb$

14. For the correct answer string to the above question, how many derivation trees are possible?

(a) 1 (b) 2
(c) 3 (d) 4

15. To manage and store information about variables and their attributes, compiler uses

(a) Parse table (b) Semantic stack
(c) Symbol table (d) Tree

16. Consider the grammar

$S \to AbBaCc|\varepsilon$
$A \to aAb|ba$
$B \to bBc|cb$
$C \to cCa|ac$

16a. Find the First( ) of S?

(a) $\{\varepsilon\}$ (b) $\{a,\ \varepsilon\}$
(c) $\{a,\ b,\ \varepsilon\}$ (d) $\{a,\ b,\ c,\ \varepsilon\}$

16b. Find the Follow( ) of B?

(a) $\{a\}$ (b) $\{b\}$
(c) $\{a,\ b\}$ (d) $\{a,\ c\}$

16c. Consider the grammar $(G) = \{A \to A + A|a|\varepsilon\}$, which one is the correct after removing left recursion?

(a) $\{A' \to AA',\ A' \to |a|\varepsilon\}$
(b) $\{A' \to A + A'|\varepsilon,\ A' \to |a\}$
(c) $\{A' \to A + A'|a|\varepsilon\}$
(d) $\{A' \to +AA'|\varepsilon,\ A \to aA'\}$

ANSWERS TO PRACTICE EXERCISES

Set 1

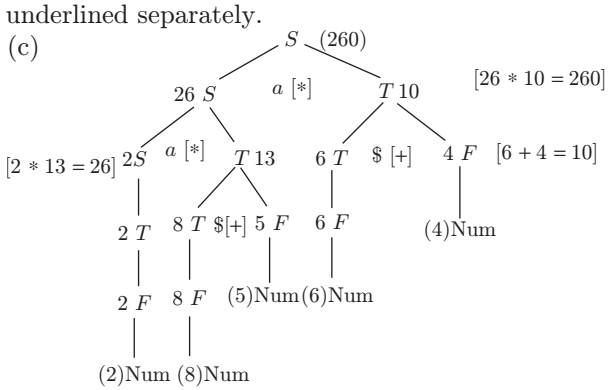
1. (a)	**10.** (b)	**19.** (d)	**28.** (c)	**37.** (c)	**46.** (c)
2. (c)	**11.** (d)	**20.** (c)	**29.** (d)	**38.** (b)	**47.** (b)
3. (a)	**12.** (d)	**21.** (a)	**30.** (a)	**39.** (b)	**48.** (a)
4. (c)	**13.** (a)	**22.** (d)	**31.** (c)	**40.** (d)	**49.** (c)
5. (d)	**14.** (a)	**23.** (d)	**32.** (a)	**41.** (a)	**50.** (c)
6. (b)	**15.** (b)	**24.** (a)	**33.** (b)	**42.** (d)	
7. (c)	**16.** (d)	**25.** (c)	**34.** (d)	**43.** (a)	
8. (d)	**17.** (b)	**26.** (d)	**35.** (b)	**44.** (12338)	
9. (b)	**18.** (d)	**27.** (d)	**36.** (b)	**45.** (a)	

Set 2

1. (b) `printf ( " a = %d , b = %d" , a , b ) ;`

Number of tokens in the above statements is 9, are underlined separately.

2. (c)

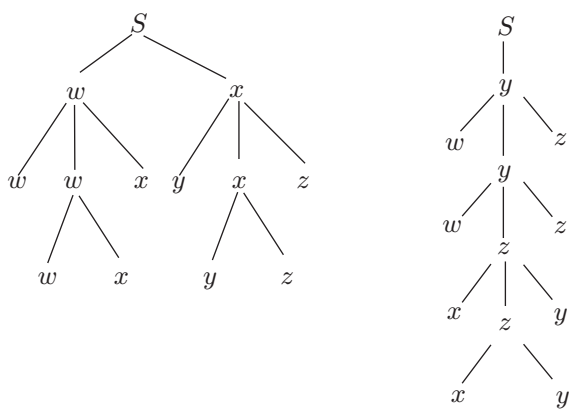


3. (c) Grammar

1. $S \rightarrow WX|Y$
2. $W \rightarrow wWx|wx$
3. $X \rightarrow yXz|yz$
4. $Y \rightarrow wYz|wZz$
5. $Z \rightarrow xZy|xy$

From the second and third production we can get ($w^n x^n y^m z^m$: $n \geq 1$, $m \geq 1$) and from the fourth and fifth production we can get $\{w^n x^m y^m z^n$: $n \geq 1$, $m \geq 1\}$, so option (c) is correct.

4. (c)



5. (b) The given grammar is ambiguous, so it is not LL(1), and hence not LR(1).

6. (d) The multiplication string itself is a complete string which does not require any further character.

7a. (a) First(T) consists the terminal which is the first in the production of T. So, First(T) = First(F) = $\{a, [\ \}$

7b. (c) First( ) of $E' = \{a, [\ \}$

7c. (c) Follow(E) consists of three things:

- **i.** The terminal which follow variable E in the right side of any production.
- **ii.** If any variable follow E in the right side of production, then first of that variable will be in follow of E.
- **iii.** If E is the starting symbol, then add $ sign into the follow of E.

So, Follow(E) ={$,]}

7d. (d) Refer to solution of Q9.

8. (c) Since an unambiguous grammar has leftmost derivation and rightmost derivations but both of them are not same.

9. (d) In LALR, all the states of CLR which are differ only by look-ahead symbol are merged. So, the number of states in LALR may be less than or equal to the number of states in CLR.

10. (d) A grammar (G) is said to be an operator grammar if and only if

- **i.** Grammar G does not have any null production.
- **ii.** G does not have any two adjacent variables on the right-hand side of the production.

11. (a) Draw parsing table for the grammar, for details procedure of parsing table refer text.

12. (c) Check the relationship diagram between LL(1) and LR(0), SLR(1), LALR(1) (refer text).

13. (a) $S \rightarrow aB \rightarrow aaBB \rightarrow aabB \rightarrow aabbS \rightarrow aabbaB \rightarrow aabbab$

14. (b) Two parse trees are possible.

15. (c) Symbol table is used to store all the information about a variable such as variable name, id and type.

16a. (c)

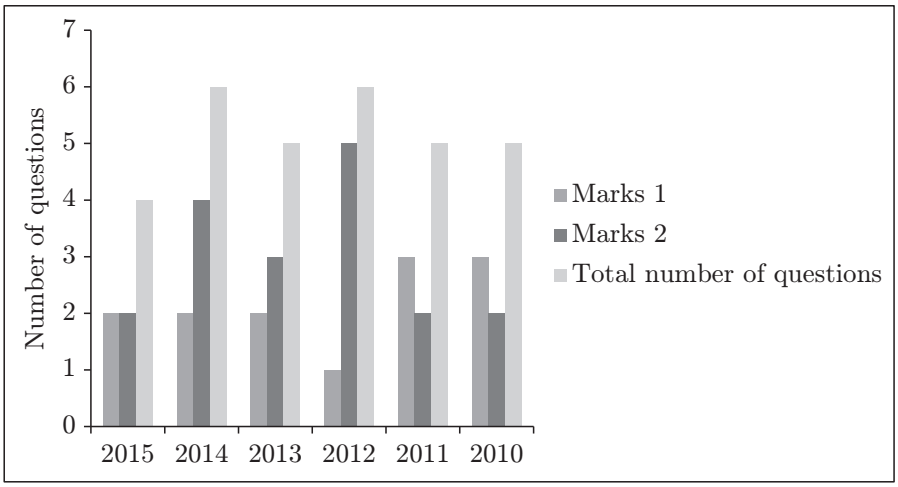
16b. (d)

16c. (d) Steps to remove left recursion:

$A' \rightarrow +AA'|\varepsilon$
$A \rightarrow aA'$

UNIT VII: OPERATING SYSTEM

LAST SIX YEARS' GATE ANALYSIS



Concepts on which questions were asked in the previous six years

Year	Concept
2015	Shared variable, SCAN algorithm, Page replacement, Processes, Process synchronization, Deadlock, Process scheduling, TLB, Best fit algorithm, Page table
2014	Page Frames, Deadlock, Translation Lookaside Buffer (TLB), Disk Scheduling Algorithm
2013	Scheduling, Disk Management, Semaphores, Concurrent Processes, Memory Management
2012	System Calls (Fork), Scheduling Algorithms, Concurrent Processes
2011	DMA Concept, Interrupts, Lightweight Processes, User—Kernel Mode, Scheduling Algorithms
2010	Deadlock, Critical Section, Page table, Concurrent Process Concept

CHAPTER 7

OPERATING SYSTEM

Syllabus: Operating system: Processes, Threads, Inter-process communication, Concurrency, Synchronization, Deadlock, CPU scheduling, Memory management and virtual memory, File systems, I/O systems, Protection and security

7.1 INTRODUCTION

An operating sysem (OS) can be defined as an intermediate program between the user and the computer hardware. On the user end, it handles application programs and at the other end it makes use of system call(s) to instruct the hardware to perform certain task as instructed by the user. There are different types of operating systems to accomplish various tasks. Mainframe operating systems are designed to optimize hardware utilization. The real-time operating systems are designed for timeliness, the time-sharing systems are designed for efficient usage of the system and other resources. Some of the examples of OS are: UNIX, Linux (different flavors, such as, Redhat, Suse, Fedora, Debian etc.), MacOS, Microsoft Windows, etc. Nowadays, the mobiles that we use are also equipped with operating system. Some of the well-known OS for mobiles are — Symbian and Android.

The function of the operating systems is to provide an environment in which a user can execute his/her commands in a **convenient** and **efficient** manner. Operating system is also called **resource allocator**. The placement of operating systems in a computing environment is shown in Fig. 7.1.

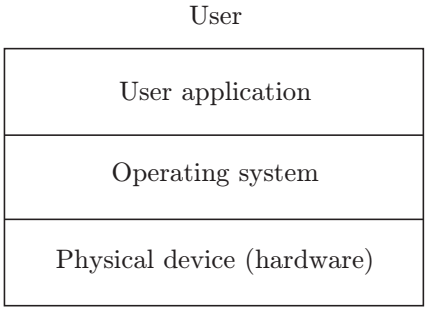


Figure 7.1 Location of an operating system in a computing environment.

System Call is a request made by a user program to get the service of an operating system.

To consider the "inside view" of the system, the OS is written as a collection of procedures, each of which can call any other, whenever required. Each parameter has a well-defined interface with input parameters and output results. A simplified layered structure of Dijkstra's "THE" operating system is shown in Fig. 7.2:

LAYER	FUNCTION
5	The operator (user)
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory management
0	Process allocation and multiprogramming

Figure 7.2 | Simplified structure of Dijkstra's THE operating system.

7.2 TYPES OF OPERATING SYSTEM

7.2.1 Batch Operating System

In a batch operating system, tasks are submitted to a pool, and only after the completion of one job, the next job is executed (Fig. 7.3). In this system, the CPU idle time (free time) is high and throughput of the system is low. Throughput is the number of jobs executed per unit time.

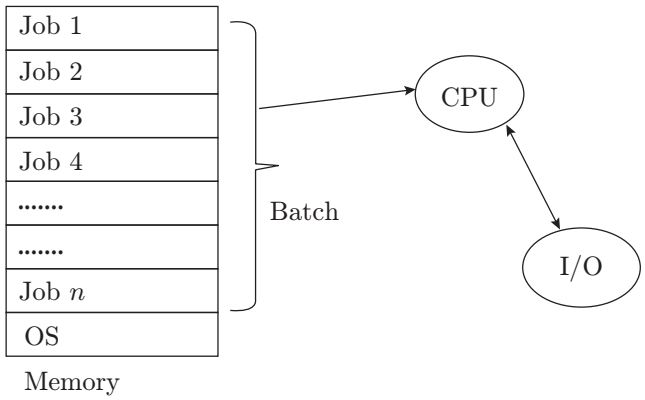


Figure 7.3 | A batch operating system.

7.2.2 Multiprogramming Operating System

A single user cannot keep either the CPU or the I/O devices busy at all times. Multiprogramming increases CPU utilization by organizing jobs so that the CPU is not idle at any time (Fig. 7.4). Jobs to be executed are maintained in the memory simultaneously and the OS switches among these jobs for their execution. When one job waits for some input (WAIT state), the CPU switches to another job. This process is followed for all jobs in memory. When the wait for the first job is over, the CPU is back to serve it. The CPU will be busy till all the jobs have been executed. Thus, increasing the CPU utilization and throughput.

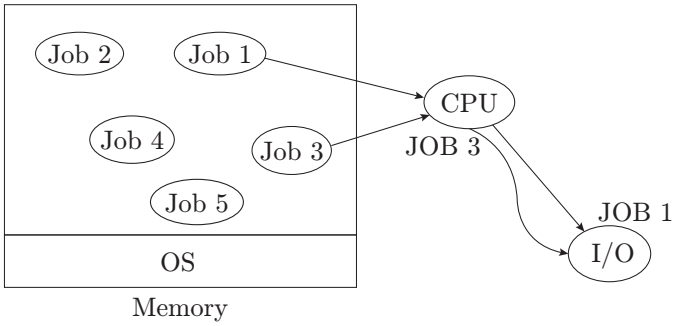


Figure 7.4 | A multiprogramming operating system.

7.2.3 Multitasking Operating System

Multitasking system is the extension of multiprogramming system. In this system, jobs will be executed in the time-sharing mode. It uses CPU scheduling and multiprogramming techniques to provide a small portion of a timeslot to each user. Each user has at least one separate program in memory.

7.2.4 Multiprocessor Operating System

Multiprocessor operating system consists of more than one processor for execution (Fig. 7.5). It improves the throughput of the system and it is also more reliable than the single processor system because if one processor breaks down then the other can share the load of that processor (Table 7.1).

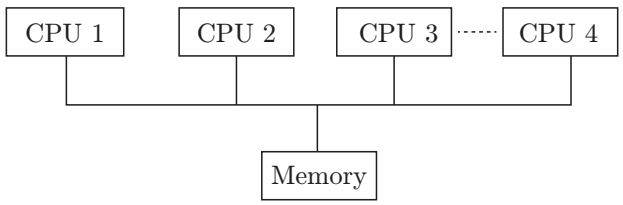


Figure 7.5 | A multiprocessor operating system.

Table 7.1 | Comparison between Batch Processing and Timesharing OS

	Batch Processing	Timesharing
AIM	Maximize processor utilization	Minimize response time
COMMANDS	Job Control Language (JCL) provided along with the job	As per the commands provided at the terminal

7.2.5 Real-Time Operating System

A real-time operating system (RTOS) is the one which deals with real-time application requests (Fig. 7.6). The input data must be processed without any delays. The processing time is shorter and the waiting time is minimum. If an RTOS meets a deadline approximately, it is known as a soft RTOS, but if deadline has to be met deterministically (hard deadline), it is a hard RTOS.

The events that an RTOS may have to respond to can be either periodic (occurring at regular intervals) or aperiodic (occurring at irregular intervals). Further, if there are n periodic events and an event i occurs with period P_i and requires T_i seconds of CPU time, then the load can be handled if

$$\sum_{i=1}^{n}(T_i / P_i) \leq 1$$

An RTOS that meets this criteria is said to be *schedulable*.

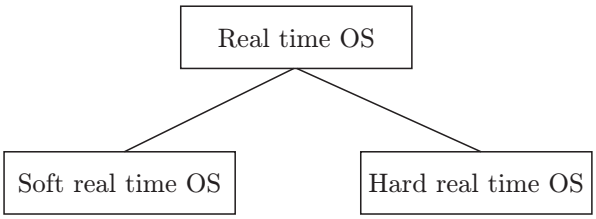


Figure 7.6 | Classification of a real-time operating system.

An operating system is concerned with the management functions of the following resources: (a) process management, (b) memory management, (c) file management and (d) device management.

7.3 PROCESS MANAGEMENT

7.3.1 Process

A program under execution is called a process. In other words, a program is in the main memory and the occupied processor is called a process. A process has various attributes, states and operation.

7.3.1.1 Attributes of a Process

A process is distinct from another based upon the following attributes:

1. **Process id:** It is a unique identification number which is assigned by the OS at the time of process creation.
2. **Process state:** It contains the state of the process where the process is currently residing. The state may be new, ready, running, waiting, halted and so on.
3. **Program counter:** It is having the address of the next instruction to be executed.
4. **Priority:** It is also a parameter assigned by the OS at the time of process creation.
5. **General purpose register**
6. **List of open files**
7. **List of closed files**

All the attributes of a process are stored in the process control block (PCB) (Fig. 7.7).

Process Id	***Process State***
Process counter	Priority
List of open files	List of close files
General purpose register information	Accounting information
I/O status information	CPU scheduling information
Memory management information	Context data memory pointers

Figure 7.7 | A process control block.

7.3.1.2 Process States

Any process is defined by a unique characteristic of its state, which informs about the current activity of the process. A process may be in one of the following states during its execution:

1. **New:** Whenever a new process is created.
2. **Running:** Whenever the process is able to get attention from the busy CPU, that is, the CPU is actually processing the instruction.
3. **Waiting:** The process waits for its turn as it requires some resource(s) which is/are held by another process(es).
4. **Ready:** A process has been allocated all the resources required and is waiting for the attention

from CPU because the CPU is busy in executing some other task.

5. **Terminated:** A process is in terminated state when the processor has finished the execution of a particular task.
6. **Suspended ready:** If processes are more than the capacity of the memory, then they will be suspended and go to suspended ready state, which is in the secondary memory.
7. **Suspended wait:** If processes are more than the capacity of memory, then they will be suspended and go to suspended waiting state, which is in the secondary memory.

7.3.2 Schedulers

An operating system has three types of scheduler (Fig. 7.8):

1. **Long-term scheduler:** It is responsible for bringing a newly created process into the system.
2. **Short-term scheduler:** It is responsible for selecting one of the processes from the ready state and scheduling that process into the running state.
3. **Mid-term scheduler:** It is responsible for suspending and resuming the processor.

7.3.3.1 Dispatcher

It is responsible for saving and loading the context of a process. The context switching is done by a dispatcher.

7.3.3.2 Degree of Multiprogramming

The number of jobs present in the memory is called as the degree of multiprogramming. Long-term scheduler controls the degree of multiprogramming.

Note: The best data structure to implement ready queue is double linked list.

7.3.3.3 Time Periods

A process has various time periods as follows:

1. **Arrival time (AT):** When a process arrives into ready state, it is called as arrival time.

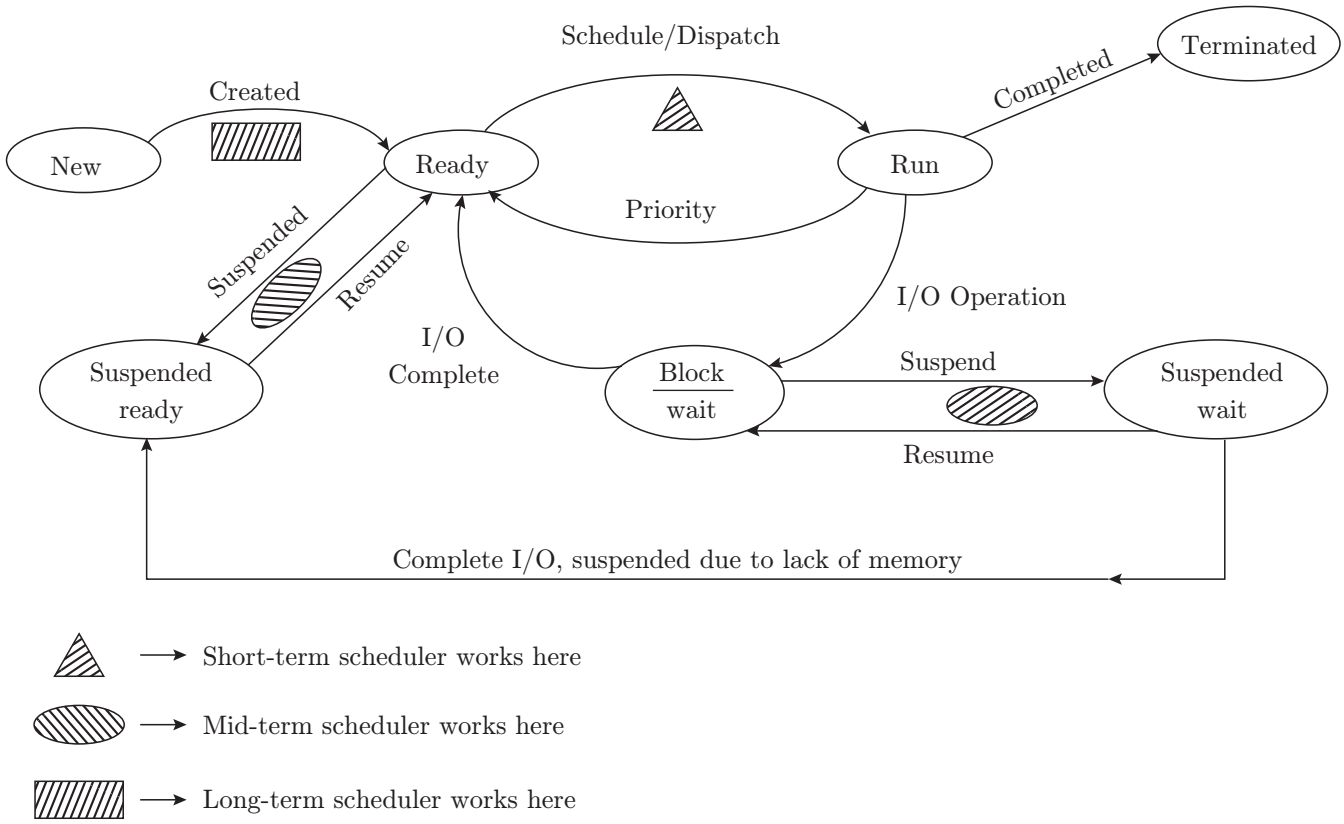


Figure 7.8 | Lifecycle of a scheduler.

2. **Burst time (BT):** The time required by a process for its execution is called as burst time.
3. **Completion time (CT):** The time when a process completes its execution is called as completion time.
4. **Turn-around time (TAT):** The time difference between completion time and arrival time is called as turnaround time.

$$TAT = CT - AT$$

5. **Waiting time (WT):** Waiting time for a process is the time duration which is spent in waiting queue by that process.

$$WT = TAT - BT$$

6. **Response time:** The first scheduled time for a process is called response time.

7.4 CPU SCHEDULING

CPU scheduling is the technique to put a process from ready queue to running state. The objective of CPU scheduling is to minimize the turnaround time and average waiting time of a process. The short-term scheduler is used to apply the CPU scheduling in the ready state to select the process to schedule on to the processor (running state).

7.4.1 Scheduling Algorithms

There are many CPU scheduling algorithms as follows:

1. First come, first serve (FCFS)
2. Shortest job first (SJF)
3. Shortest remaining time first (SRTF)
4. Round robin
5. Priority based
6. Highest response ratio next (HRRN)
7. Multilevel queue
8. Multilevel feedback queue

7.4.1.1 First-Come, First-Serve Scheduling

In this simple scheme, the allocation of CPU is on the basis of first come, first serve (FCFS). A queue data structure is maintained for FCFS scheduling. When the wait of the process is over and it enters into the ready state, its PCB is linked onto the tail of the queue. The process at the head of the queue is allocated to the processor and the running process is removed from the queue. The FCFS scheduling is non-preemptive.

Example 7.1

Consider the following workload table of three processes:

Process	Burst Time (ms)
P_1	15
P_2	4
P_3	2

Let us suppose that the process arrives at time = 0 in the order P_1, P_2, P_3. By applying the FCFS scheduling algorithm, we get the result shown in the following Gantt chart:

P_1	P_2	P_3

0 15 19 21

Waiting Time:

Process	Waiting Time (ms) WT = (TAT – BT)
P_1	0
P_2	15
P_3	19

Average waiting time = (Total waiting time)/3
= (0 + 15 + 19)/3 = 34/3 = 11.33 ms

Turnaround Time:

Process	Turnaround Time (ms) TAT = (CT – AT)
P_1	15
P_2	19
P_3	21

Average turnaround time = (15 + 19 + 21)/3
= 55/3 = 18.33 ms.

7.4.1.2 Shortest-Job-First Scheduling

In SJF scheduling, the job that requires CPU attention for lesser time is chosen first for CPU attention. The data structure used is a queue in which the jobs are arranged on the basis of CPU time required by the process (in ascending order). If there are two processes that have the same requirement of CPU time, then the order in which they arrived (FCFS scheduling) is used to break the tie. This algorithm can be either pre-emptive or non-preemptive. Pre-emptive SJF scheduling is sometimes called shortest-time-first scheduling (STF).

Example 7.2

Consider the same Example 7.1:

Process	Burst Time (ms)
P_1	15
P_2	4
P_3	2

Assuming that all these processes arrive at the same time, using SJF scheduling, we would schedule these processes according to the following Gantt chart:

P_3	P_2	P_1

0 2 6 21

Waiting Time:

Process	Waiting Time (ms) WT = (TAT − BT)
P_1	6
P_2	2
P_3	0

Average waiting time = (6 + 2 + 0)/3 = 8/3 = 2.66 ms

Turnaround Time:

Process	Turnaround Time (ms) TAT = (CT − AT)
P_1	21
P_2	6
P_3	2

Average turnaround time = (21 + 6 + 2)/3 = 29/3
= 9.66 ms

7.4.1.3 Shortest Remaining Time First

SRTF is a pre-emptive version of the SJF. It permits processes that enter the ready list to pre-empt the running process if the time for the new process (or for its next burst) is less than the remaining time for the running process (or for its current burst).

Example 7.3

Consider the following example. We have the workload table of four processes.

Process	Arrival Time (ms)	Burst Time (ms)
P_1	0	5
P_2	1	2

(*Continued*)

Continued

Process	Arrival Time (ms)	Burst Time (ms)
P_3	2	8
P_4	3	3

1. Now, P_1 is executed first, because it is the only process available at that time ($t = 0$).
2. At $t = 1$, we have a new process P_2 which requires only 2 ms whereas the already executing process P_1 still requires 4 ms to complete. Therefore, the CPU will start executing P_2 because it has the shortest running time out of the two available processes P_1 and P_2.
3. At $t = 2$, we have three processes, namely P_1, P_2 and P_3, P_2 has the shortest remaining time, therefore it will continue to execute.
4. At $t = 3$, process P_2 terminates and a new process P_4 enters the ready queue. Now, we have three processes P_1, P_3 and P_4 waiting for CPU attention. P_4 has the shortest remaining time (3 ms) as compared to P_1 (4 ms) and P_3 (8 ms).

The Gantt chart is shown below:

P_1	P_2	P_4	P_1	P_3

0 1 3 6 10 18

Turnaround Time:

Process	Turnaround Time (ms) TAT = (CT − AT)
P_1	10 − 0 = 10
P_2	3 − 1 = 2
P_3	18 − 2 = 16
P_4	6 − 3 = 3

Average turnaround time = (10 + 2 + 16 + 3)/4
= 7.75 ms

Waiting Time:

Process	Waiting Time (ms) WT = (TAT − BT)
P_1	10 − 5 = 5
P_2	2 − 2 = 0
P_3	18 − 8 = 10
P_4	3 − 3 = 0

Average waiting time = (5 + 0 + 8 + 10)/4 = 5.75 ms

7.4.1.4 Round-Robin Scheduling

Round-robin scheduling is basically similar to FCFS scheduling, but pre-emption is added to switch between

processes. A small unit of time, generally from 10 to 100 ms, called a time quantum (or time slice), is defined. The ready queue is treated as a circular queue. The round-robin (RR) scheduling algorithm is designed especially for time-sharing systems.

Example 7.4

Consider the following example. We have the workload table of four processes, and time quantum of 4 ms is used.

Process	Burst Time (ms)
P_1	21
P_2	3
P_3	3
P_4	5

Using RR scheduling, we would schedule these processes according to the following Gantt chart:

P_1	P_2	P_3	P_4	P_1	P_4	P_1	P_1	P_1	P_1

0 4 7 10 14 18 19 23 27 31 35

Waiting Time:

Process	Waiting Time (ms) WT = (TAT − BT)
P_1	35 − 24 = 11
P_2	7 − 3 = 4
P_3	10 − 3 = 7
P_4	19 − 5 = 14

Average waiting time = (11 + 4 + 7 + 14)/4 = 9 ms

Turnaround Time:

Process	Turnaround Time (ms) TAT = (CT − AT)
P_1	35
P_2	7
P_3	10
P_4	19

Average turnaround time = (35 + 7 + 10 + 19)/4
= 17.75 ms

7.4.1.5 Priority Scheduling

In priority scheduling, each process is assigned a priority and the highest priority process is allocated to the CPU first. The data structure used is a priority queue, that is, the process with the highest priority is in the front of the queue. If there are two processes that have the same priority, then their order of arrival (FCFS scheduling) is used to break the tie. The major drawback of this algorithm is starvation, that is, a low-priority process have to wait for an infinite time because the high priority processes gets executed first and after some time another high priority process comes and the low priority process is again left out.

Example 7.5

Consider the following example. We have the workload table of five processes.

Process	Burst Time (ms)	Priority
P_1	15	3
P_2	1	1
P_3	5	4
P_4	1	5
P_5	10	2

Using priority scheduling, we would schedule these processes according to the following Gantt chart:

P_2	P_5	P_1	P_3	P_4

0 1 11 26 31 32

Waiting Time:

Process	Waiting Time (ms)
P_1	11
P_2	0
P_3	26
P_4	31
P_5	1

Average waiting time = (11 + 0 + 26 + 31 + 1)/5
= 69/5 = 13.8 ms

Turnaround Time:

Process	Turnaround Time (ms)
P_1	26
P_2	1
P_3	31

(*Continued*)

Continued

Process	Turnaround Time (ms)
P_4	32
P_5	11

Average turnaround time = (26 + 1 + 31 + 32 + 11)/5
= 20.2 ms

7.4.1.6 Highest Response Ratio Next (HRRN)

In HRRN algorithm, a processor is assigned to a process on the basis of response ratio. Response ratio is calculated by the following formula:

$$\text{Response ratio} = \frac{w + s}{s}$$

where w = waiting time and s = service time or burst time.

The HRRN algorithm favours the shorter jobs and limits the waiting time of larger jobs. It is non-preemptive scheduling algorithm.

Example 7.6

Consider the following example. We have the workload table of five processes.

Process	Burst Time (ms)	Arrival Time
P_1	3	0
P_2	6	2
P_3	4	4
P_4	5	6
P_5	2	8

Using HRRN scheduling, we would schedule these processes according to the following Gantt chart:

P_1	P_2	P_3	P_5	P_4

0 3 9 13 15 20

1. Now P_1 is executed first, because it is the only process available at that time ($t = 0$).
2. At $t = 2$, we have a new process P_2 which requires only 6 ms whereas the already executing process P_1 still requires 1 ms to complete. Therefore, the CPU will continue to execute P_1 because it is non-preemptive scheduling.
3. At $t = 3$, we have only one process P_2, so P_2 will get executed.
4. At $t = 9$, we have three processes P_3, P_4 and P_5. To select which process will be executed next, we have to calculate the response ratio for all of the three processes. The process having the highest response ratio will be executed first.
   At $t = 9$,
   Response ratio of $P_3 = (w + s)/s = (5 + 4)/4$
   $= 2.25$
   Response ratio of $P_4 = (w + s)/s = (3 + 5)/5$
   $= 1.60$
   Response ratio of $P_5 = (w + s)/s = (1 + 2)/2 = 1.5$
   Process response ration of P_3 is high, so it will be executed next.
5. Repeat the above steps for all the processes.

Waiting Time:

Process	Waiting Time (ms) WT = TAT − BT
P_1	0
P_2	1
P_3	5
P_4	9
P_5	5

Average waiting time = (0 + 1 + 5 + 9 + 5)/5 = 20/5
= 4 ms

Turnaround Time:

Process	Turnaround Time (ms)
P_1	3
P_2	7
P_3	9
P_4	14
P_5	7

Average turnaround time = (3 + 7 + 9 + 14 + 7)/5
= 8 ms

7.4.1.7 Multilevel Queue Scheduling

In multilevel queue scheduling, the ready queue is partitioned into several separate queues such that each process belongs to some queue. Various properties for allocating jobs to different queues may depend upon memory size, process priority or process type, or any other parameter(s). Each queue may follow different scheduling algorithms. Figure 7.9 depicts the multilevel queue scheduling.

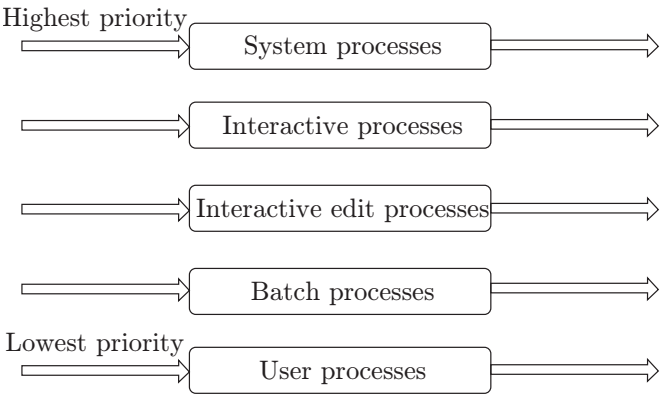


Figure 7.9 | Multilevel queue scheduling.

7.4.1.8 Multilevel Feedback Queue Scheduling

In multilevel feedback queue scheduling, the multilevel queue scheduling approach has been enhanced. Here, the process is not permanently assigned to a queue but a process is allowed to move between queues. The major advantage of this approach is that it prevents starvation, because a process which is a low-priority process can be moved to a higher-priority queue after some time.

Figure 7.10 depicts a three-level feedback queue scheduler. The scheduler first executes all processes in the first queue. When this queue is empty, it will execute processes in the next queue (having a higher quantum or time-slice). Similarly, processes in subsequent queue will be executed only if the previous queue(s) are empty. Further, a process that arrives for lower queue (having lower quantum) will pre-empt a process in queue having a higher quantum.

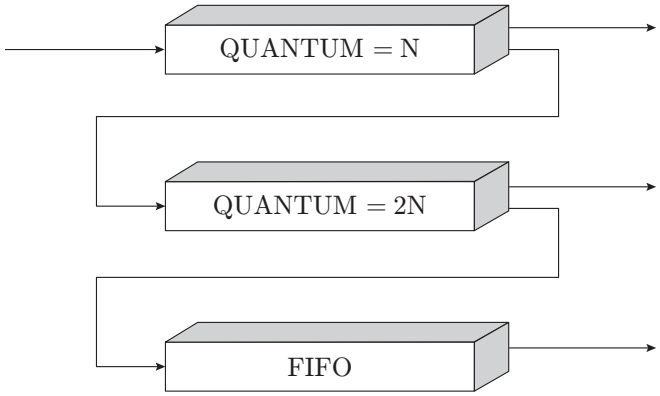


Figure 7.10 | Multilevel feedback queue scheduling.

Note: Apart from the above well-known scheduling algorithms, there are two other approaches also.

1. **Guaranteed scheduling:** This scheduling algorithm guarantees the user that particular amount of attention will be given to user. For example, if there are n users, then we can say that each user will get about $1/n$ of the CPU time. This seems to be a good scheduling algorithm theoretically but is harder to implement due to overheads involved in switching of processes.
2. **Lottery scheduling:** This is similar to lottery. Whenever a scheduling decision has to be made, any process is selected at random and is given more amount of CPU time as compared to other processes similar to the lottery winner.

There are two different categories of schedulers–Dynamic and Static. A dynamic scheduling can be defined as a scheduling defined at run-time; whereas in static the scheduling decision is taken before the system starts running. There are some popular dynamic scheduling algorithms such as:

1. **Rate monotonic algorithm:** A priority value is assigned to each process depending upon the frequency of the occurrence. A process with high frequency occurrence gets a high priority. So, at run-time the highest priority process gets the CPU.
2. **Earliest deadline first:** The processes are sorted according to the deadline (say, in RTOS) and the CPU time is allocated to the first process in the list, that is, which has to meet the deadline urgently.
3. **Least laxity algorithm:** A process that has the least spare-time (laxity) is given to the CPU.

7.5 PROCESS SYNCHRONIZATION

7.5.1 Types of Processes

A process consists of two major elements namely, the program code and a data set. The processes can be classified as follows:

1. **Independent process:** This is a process which cannot affect or get affected by execution of another process.
2. **Cooperating process:** This is a process that affects and gets affected by execution of other processes.
3. **Concurrent processes:** These processes are independent, interacting processes which are executed simultaneously over a period of time.

When two processes want to use or access a single global variable, then the problem of inconsistency occurs. The problem can be solved using the process synchronization.

7.5.2 Interprocess Communication

Inter-process communication is a technique to exchange data among multiple processes or threads. IPC allows programmers to schedule activities to different processes. One process controls the whole communication. There are different IPC techniques available:

- Message passing
- Shared memory
- Remote procedure calls
- Synchronization

7.5.3 Classical Synchronization Problems

Classical synchronization problems are used to implement synchronization mechanisms and critical sections. Some of the interesting problems are: **The Producer-Consumer Problem, The Readers-Writers Problem, The Dining Philosophers Problem, Sleeping Barber, Cigarette Smoker Problem**, etc. The following sections discuss only the first three of them.

7.5.3.1 The Producer-Consumer Problem

To further understand the concept, let us take an example of producer—consumer problem. A producer produces data that is consumed by a consumer (e.g., spooler and printer). A buffer holds the produced data, which is not yet consumed. There exist several producers and consumers, and initially, we assume the buffer is unbounded.

Let count be a global variable. Count = 5

Producer: Register1 = Count [5]

Producer: Register1 + = 1 [6]

Consumer: Register2 = Count [5]

Consumer: Register2 = Register2 -1 [4]

Producer: Count = Register 1 [6]

Consumer: Count = Register 2 [4]

As we can see, output will be either 4 or 6. But it should be 5.

Here, output depends on the order of execution of statements. Synchronization is required, otherwise output will be inconsistent.

7.5.3.2 The Readers-Writers Problem

In this problem, there are many concurrent processes that try to access the same file. Some processes try to read data from file or some want to insert or change the contents in the file. According to Bernstein constraint, many readers may access the file at the same time, but when the writer is accessing the file, no one should be allowed to access it. This can be ensured by following methods:

1. Assign readers more priority than writers. If a writer has no permission to access the file, any reader can get access to it. But this may cause infinite waiting for the writer.
2. Another method can be used to assign writers large priority over readers. When writer requests for file, all the current readers finish their access and new readers have to wait until the writer finishes its task. This may cause infinite waiting for reader.

7.5.3.3 The Dining Philosopher's Problem

Another classic problem of synchronization is the dining philosopher's problem. Here five philosophers sit around a circular table with five chairs. There are five plates and forks placed in a circular manner (Fig. 7.11). When the philosopher wants to eat, s/he picks up forks from left and right of his plate. When finishes eating, s/he keeps both the forks back. This ensures that no philosopher starves. Two problems are faced in this:

1. Each philosopher picks up one fork and none of them gets a second fork.
2. No philosopher should starve due to another.

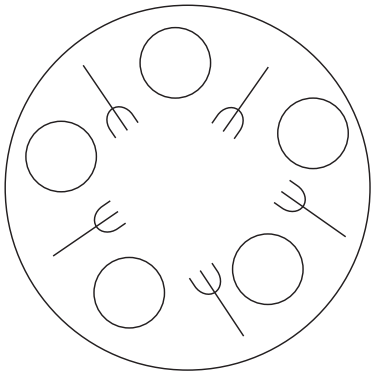


Figure 7.11 | The Dining Philosopher's Table.

7.5.4 Race Condition

When several processes have access to shared data, then the final value of this shared data will eventually depend upon the action performed by the last process. This is a situation in which the result depends upon which process is executed first. Such a situation or condition is known

as "race condition". The producer—consumer example discussed in the previous section is one such example. The part of the program where the shared memory is accessed is known as critical section. The following are the four conditions that are used to provide a good solution to this problem:

1. No assumption has to be made about the number of CPUs and their speed.
2. No two processes may be executing their critical section at the same time.
3. No process has to wait indefinitely to enter its critical region.
4. No process executing outside its critical section can block other processes.

7.5.5 Critical Section

Critical section is a part of program global variables which are manipulated by the processes.

7.5.5.1 Critical Section Problem

Critical section problem occurs when n number of processes are competing for critical section. The problem is to ensure that if one process is in the critical section, no other process is allowed to enter into critical section at the same time.

```
do {
      [ Entry section ]
              Critical Section
      [ Remainder section ]
              Remainder Section
   } while(1);
```

Busy Waiting

Spinning or busy waiting is a mechanism in which the process keeps on checking for the condition to be true. For example the process is checking repeatedly whether a lock is available or not. This technique can be used to generate the random time delays. This consumes so many CPU cycles.

```
do
{
    wait(s);
// critical section
    signal(s);
// remainder section
} while(1);
```

Processes keep on checking for semaphore value to be zero. This continuous looping problem is busy waiting. When semaphore does busy waiting, it is called 'spinlock'.

7.5.5.2 Solutions to Critical Section Problem

1. **Mutual exclusion:** If process P_i is executing in its critical section, then no other process can execute their critical sections.
2. **Progress:** If no process is executing in its critical section and there exist some process that wishes to enter into critical section, then the selection of process that will enter into critical section cannot be postponed indefinitely.
3. **Bounded waiting:** A bound must exist on the number of times that other processes are allowed to enter into critical section after a process has made the request for critical section and it is not granted.

7.5.6 Peterson's Algorithm for Two Processes

Peterson's algorithm is considered the most correct algorithm for process synchronization.

P_i	P_j

do{
flag[i] = true;
turn = j;
while(flag[j] and turn = j);
Critical Section
flag[i] = false
Remainder Section
}while(1);
	do{
	flag[j] = true;
	turn = i;
	while(flag[i] and turn = i);
	Critical Section
	flag[j] = false
	Remainder Section
	}while(1);

The above code ensures mutual exclusion, progress and bounded wait.

7.5.7 Semaphores

Semaphores are synchronization tools. These are the shared variables, which can be implemented as integer variables. These can be accessed via two indivisible atomic operations that are wait and signal.

7.5.7.1 Integer Implementation of Semaphores

The operations (wait and signal) are realized by making use of integer values.

1. **Wait:** It reduces the value of semaphore by 1 unit. It can be denoted by $P(S)$.
   Wait(S):

```
while S ≤ 0
      Do no_operation;
S--;
```

2. **Signal:** It increases the value of semaphore by 1. This can be denoted by $V(S)$.
   Signal(S):

```
S++;
```

3. **Disadvantages:**
   - We must keep the value of $S = 1$
   - Busy waiting which consumes many CPU cycles

7.5.7.2 Binary Semaphores

The value of binary semaphores can range between 0 and 1 only. Binary semaphores are also known as mutex locks as they ensure mutual exclusion.

Problem 7.1: Each process $P_i = 1, 2, \ldots 11$ is coded as follows:

```
P (mutex)
   Critical Section
V (mutex)
```

The code for P_{10} and P_{11} is identical except it uses V in the place of P and P in the place of V.

How many processes will be in critical section?

Solution:

Let $S = 1$

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}
P	P	P	P	P	P	P	P	P	P	V
V	V	V	V	V	V	V	V	V	V	P

Say P_1 enters in the critical section, $S_1 \rightarrow 0$.
Now P_{11} process can increment $S \rightarrow 1$ and will have access to critical section.
Now any process can enter into critical section, because S is incremented by P_{11}.
Three processes will enter into critical section.

Note: 1. With change in one sequence, three processes will enter in the critical section.

2. With change in two sequences, five processes will enter into the critical section and so on.

7.5.7.3 Counting Semaphores

This semaphore makes use of a 'count' variable. This variable is initialized by the numeric value of the finite number of instances of a resource. The count value is decremented whenever the instance of that resource is allocated to a process. So, if the count value of a resource is zero, it means that all instances of that resource have been allocated and no process can enter into critical section. Further, when a process releases a resource this count value is incremented.

Implementation of Counting Semaphore

1. Wait operation

Wait (S):

```
{
        C--;
        if(C < 0)
        {
             Block the process and put
               in blocked queue
        }
}
```

2. Signal operation

Signal (S):

```
{
        C++;
        if(C ≤ 0)
        {
           Remove process from block queue
             and keep in ready queue
        }
}
```

Example 7.7

Counting semaphore = 7

$P = 20$ [decrement semaphore]

$V = 15$ [increment semaphore]

Value after implementing P and V: $C - P + V$

$$7 - 20 + 15 = 2$$

Table 7.2 summarizes the common concurrency mechanisms

Table 7.2 | Common concurrency mechanisms

1.	Semaphore	An integer value is used for signalling between different processes. Only three operations are performed – initialize, increment and decrement. Also known as counting semaphore.
2.	Binary Semaphore	Only two values 0 and 1 are used.
3.	Mutex	Similar to binary semaphore (lock and unlock values are used) but the major difference is that the process which has locked the mutex is the only one that can unlock it.
4.	Monitor	It is a line of code that contains access procedures, initialization code and variables. The access procedures are critical sections.
5.	Event Flag	It is a memory word (such as accumulator flags), which is used for synchronization.
6.	Condition Variable	A particular data type is used to block a given process until some specified condition is not met.
7.	Mailboxes	The two processes can exchange information through mailbox and also can take its help for synchronizing their activities.
8.	Spinlocks	A process executes in an infinite loop until the locked variable is available.

(*Continued*)

7.5.8 Deadlock and Starvation

P_0	P_1
wait(S);	wait(Q);
wait(Q);	wait(S);
.	.
.	.
.	.
.	.
signal(S);	signal(Q);
signal(Q);	signal(S);

In the above code, let $S = 1$, $Q = 1$

Process P_0 will make $S \rightarrow 0$

Process P_1 will make $Q \rightarrow 0$

Both will not be able to move further. This is the deadlock state.

7.6 DEADLOCKS

A deadlock is a situation where computer programs sharing the same resources are permanently blocked. As a result of which, each program prevents the other from acquiring the resources required for its completion. Let us consider an example, there are two friends (processes) and they intend to write a letter. To accomplish this task they need two pens and two pages (resources). Now, one of them has pages (papers) and the other a pen. A possible solution would have been that one person can give his resource to another so that the other person completes the job (write the letter) and after his job is finished this person can also complete the job. But none of them is willing to part the resources s/he has. This is a simple case of deadlock.

7.6.1 Resource Types

There are two types of resources, namely, pre-emptable and non-preemptable resources.

1. **Pre-emptable:** These resources can be taken away from the process with no ill effects, for example, CPU, memory.
2. **Non-preemptable:** These resources can be taken away from the process (without causing any ill effect). For example, CD, USB drive, plotter, printer, etc. We cannot give such resources to other processes in the middle of their jobs.

7.6.2 Characteristics of a Deadlock

The following four conditions must exist simultaneously for a deadlock to occur:

1. **Mutual exclusion condition:** The resources involved are non-shareable. Each resource must be assigned to only one single process at a time.
2. **Hold and wait condition:** The process currently holds a resource and is waiting to acquire another requested resource held by other process(es).
3. **Non-preemptive condition:** Resource(s) that have already been acquired by a process cannot be taken back forcibly unless and until they are released by the holding process.
4. **Circular wait condition:** The processes form a circular list or chain, where each process in the list is waiting for a resource held by another process.

To understand these four concepts clearly, let us take an example of the road as a resource, shown in Fig. 7.12.

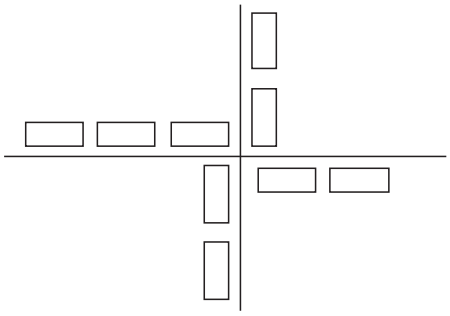


Figure 7.12 | A traffic deadlock.

1. **Mutual exclusion condition:** Only one vehicle can be moved.
2. **Hold and wait condition:** Each vehicle occupies some part of the road and is waiting to move to the other side of road.
3. **Non-preemptive condition:** Road space occupied by a vehicle cannot be taken away.
4. **Circular wait condition:** Each vehicle on the road is waiting for the next vehicle to move.

7.6.3 Resource Allocation Graph

A resource allocation graph is a graph which has two different types of nodes, the process nodes (represented by circle) and resource nodes (represented by rectangles). For different instances of a resource, there is a dot in the resources nodes (rectangle). For example, if there are two identical drives, we will depict it with two dots.

The directed edges among these nodes represent resource allocation and release. An edge from the resource to the process node denotes that it has been acquired by the process, whereas an edge from the process node to the resource node denotes that the process is requesting for the resource. For example, consider the resource allocation graph in Fig. 7.13 and we can easily conclude that there is a deadlock.

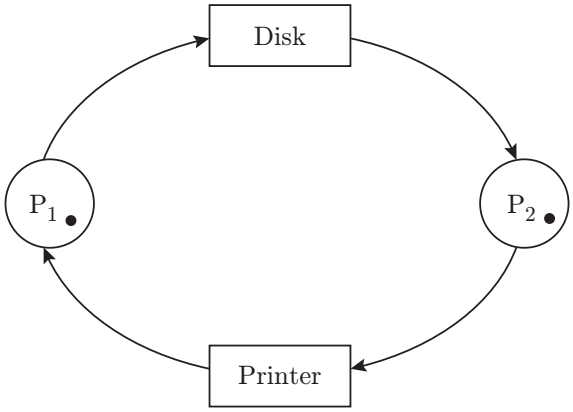


Figure 7.13 | Resource allocation graph.

By making use of resource allocation graph we can check for deadlock (i.e., the graph has a circle) and if there is no circle, it indicates the lack of circular-wait condition, hence deadlock will not occur. This is depicted in Fig. 7.14.

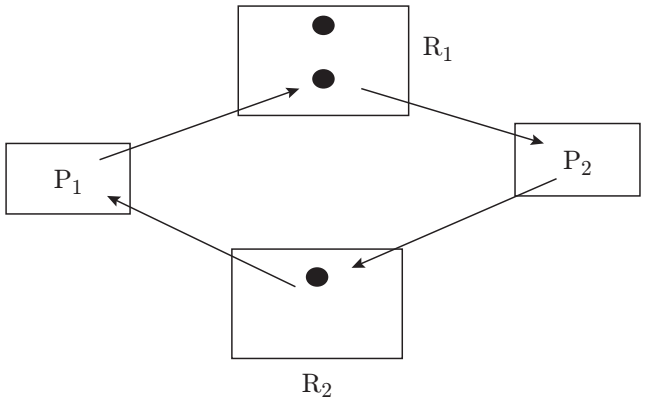


Figure 7.14 | A resource allocation graph having a cycle and no deadlock. (The figure also depicts an alternative way of representing resource allocation graphs.)

The following conditions are depicted in the resource allocation graph shown above:

1. **There are three sets of process (P), resource (R) and edges (E):**
   $P = \{P_1, P_2\}$
   $R = \{R_1, R_2\}$
   $E = \{R_1 \rightarrow P_1, P_1 \rightarrow R_2, R_2 \rightarrow P_2, P_2 \rightarrow R_1\}$
2. **Resource instances:**
   Two instances of R_1
   One instance of R_2
3. **Process states:** Process P_1 is holding an instance of resource type R_1 and is waiting for an instance of resource type R_2. Further, process P_2 is holding an instance of resource type R_2 and is waiting for an instance of resource type R_1.

The following tips help us to check the resource allocation graph easily and predict the presence of cycles.

1. If there is no cycle in the resource allocation graph, there is no deadlock.
2. If there is a cycle in the graph and each resource has only one instance, then there is a deadlock. In such a case, the presence of a cycle is the necessary condition in order for deadlock to occur (as there is only one instance of the resource).
3. If there is a cycle in the graph, and each resource has more than one instance, there may or may not be a deadlock. Therefore, we conclude that a cycle in the resource allocation graph is a necessary but not a sufficient condition for a deadlock in the case of multiple instances of a resource.

The following table summarizes the above discussion

Possibility of Deadlock (03 conditions)	Existence of Deadlock (04 conditions)
Mutual Exclusion, No Preemption & Hold and wait	Mutual Exclusion, No Preemption, Hold and wait & Circular Wait

A deadlock can be dealt by following any of the three strategies:

1. Deadlock prevention
2. Deadlock avoidance
3. Deadlock detection and recovery

7.6.4 Deadlock Prevention

To prevent a deadlock, we must ensure that any one of the following four conditions listed below (Section 7.5.2) should not hold. Let us consider them one by one:

1. **Mutual exclusion condition:** Let us take the example of tossing a coin, getting a head or tail is mutually exclusive, which means that we cannot have both head and tail. The mutual exclusion condition means that there are some resources (such as CD Drive, Printer, etc.) which once allocated to a process cannot be taken back unless and until the process releases it (i.e., non-preemptable resources). So, such resources should be allocated to only one process at a time. They cannot be shared. Although, this does not mean that none of the resources can be shared. Certain pre-emptable resources (such as memory, CPU) can be shared, because they will never result in a deadlock. So, those resources which cannot be shared are mutually exclusive and this condition needs to be taken care by OS.
2. **Hold and wait condition:** In order to ensure this condition never occurs in the system, we have to devise a strategy **such** that a process requests all its required resources at one time and block**s** the process until all requests can be granted. This approach has its drawback that a process cannot know what resources would be required before it actually starts execution, and sometimes it may have to wait for a long time in order to get all its required resources; whereas it could have used some of the already allocated resources and released them back to the pool to be used by other processes. For example, a program may require five CDs, there should be five free drives before the execution begins, while the program might be using only one drive to begin with. Thus, this approach leads to serious waste of resources.
3. **Non-Preemption condition:** The non-preemption condition can be overcome by the following protocol. If a process is holding some resources and requests another resource that cannot be immediately allocated to it (which means, the process must wait), then all resources currently being held are pre-empted (released). These pre-empted resources are added to the list of resources. For example, when a process releases resources, the process may lose all its work to that point. The major disadvantage of this strategy is the possibility of indefinite postponement (starvation). However, this protocol is often applied to resource whose state can be easily saved and restored later, such as CPU registers and memory space. It cannot generally be applied to printers and drives.
4. **Circular wait condition:** A linear ordering of different resources can be used to allocate the resources to a process, which means that if a process has been allocated a resource, then it can only be allocated a resource in the linear order.

Another way to implement it could be use some kind of indexing or a subscript. Such a strategy is helpful in avoiding cycle(s) in resource allocation graph and results in overcoming the circular wait condition.

7.6.5 Deadlock Avoidance

The side effects of preventing deadlocks are similar to the side effects of walking carefully so that you donot fall as compared to walking freely. These are as follows:

1. Low device utilization
2. Reduced system throughput

Another approach is to anticipate a deadlock before it actually occurs. This method differs from deadlock prevention, which guarantees that deadlock cannot occur by denying any of the four necessary conditions of the deadlock, whereas a 'Deadlock Avoidance' algorithm dynamically examines the resource-allocation state to ensure that there can never be a circular wait condition. There are three basic pillars to attain this objective:

1. Safe state algorithm
2. Resource allocation graph algorithm
3. Banker's algorithm

7.6.5.1 Safe State

A safe state can be defined as a state in which there is no deadlock. This can be achieved if

1. All the requested resources are allocated to the process.
2. If a process needs an unavailable resource, it may wait until the same has been released by a process to which it has already been allocated. If such a sequence does not exist, it is known as unsafe state.

Taking care of these two rules will never result in a deadlock.

7.6.5.2 Resource Allocation Graph Algorithm

The resource allocation graphs can also be used for deadlock avoidance. The major limitation of this strategy is that it is not applicable to a resource allocation system with multiple instance of each resource type.

7.6.5.3 Banker's Algorithm (Proposed by Dijkstra's)

Banker's algorithm is used to avoid deadlocks, is discussed in the following text:

1. **Basic principle:** It is similar to a banking system, that is, to ensure that the bank never allocates its available cash such that it can no longer satisfy the needs of all its customers.
2. **Working:** When a new process enters the system, it must declare the maximum number of instances of each resources type it may need. This number should not exceed the total number of resources in the system.

Now, when a user requests a set of resources, the system must determine whether the allocation of these resources will leave the system in a safe or in unsafe state. If in safe state, then the resources requested are allocated, otherwise the process must wait until some other process releases the requested resource.

We need the following data structures:

1. **Available:** A vector of length m indicates the number of available resources of each type. If Available[j] = k, there are k instances available of resource type R_j.
2. **Max:** An $n \times m$ matrix defines the maximum demand of each process, if Max[i, j] = k, then process P_i may request at most 'k' instances of resource type R_j.
3. **Allocation:** An $n \times m$ matrix defines the number of resources of each type currently allocated to each process.
   - If Allocation[i, j] = k
   - Then, process P_i is currently allocated 'k' instances of resource type R_j.
4. **Need:** An $n \times m$ matrix indicates the remaining resources needed for each process.
   - If Need[i,j] = k
   - Then, process P_i may need 'k' more instances of resource type R_j to complete its task.
   - Further, Need[i, j] = Max[i, j] − Allocation[i, j]

The Banker's algorithm consists of the following two parts:

1. **Safety algorithm:** The algorithm for finding out whether a system is in a safe state can be defined as follows:
   - Let 'Work' and 'Finish' be vectors of lengths m and n, respectively.
     Work: = Available
     Finish [i]: = False, for i = 1,2,3, …, n
   - Find an 'i', such that
     Finish[i]: = False
     Need $i \leq$ work, if no such i exists, go to step 4.
   - Work: = Work + Allocation
     Finish[i]: = true
     Go to step 2.
   - If Finish[i]: = true for all i, then the system is in a safe state.

2. Resource request algorithm

If $\text{Request}_i[j] = k$, then process P_i wants k instances of resources type R_j. When a request for resources is made by process P_i, the following actions are taken:

- If $\text{Request}_i \leq \text{Need}_i$, go to step 2.
  Otherwise, raise an error condition because the process has exceeded its maximum claim.
- If $\text{Request}_i \leq \text{Available}$, go to step 3.
  Otherwise, P_i must wait, as the resources are not available.
- Let the system pretend to have allocated the requested resources to process P_i by modifying the state as follows:
  Available: = Available − Request_i;
  Allocation_i: = Allocation_i + Request_i;
  $\text{Need}_i = \text{Need}_i - \text{Request}_i$

Now, if the resulting transactions are safe, process P_i is allocated its resources.

Time Complexity

Banker's algorithm considers each request as it occurs, and checks whether it leads to a safe state or not. If it does, the request is granted; otherwise, it is postponed for some time later. This algorithm has a time complexity of $O(n^2)$, where n is the number of processes.

Limitations of the Banker's Algorithm

The following are some limitations of the banker's algorithms:

1. A fixed number of processes and resources are required. Further, all the processes should know their requirement of resources before starting.
2. No other process can start until the algorithm has been completely executed.
3. It relies on the input information, the resources may get underutilized if inaccurate information is provided.
4. This algorithm itself takes good amount of time to execute, more information about processes and resources will result in more time to execute.
5. The requests granted should be in a finite time.
6. If a resource becomes unavailable (e.g., a CD drive stops functioning), it can result in an unsafe state.
7. It is difficult to have a priori information about the need of resources.

Problem 7.2: Consider a system having five processes and three resources types A, B and C. Resource type A has nine instances, B has four instances and C has seven instances. Initially, the snapshot of the given system is as follows:

Process	Allocated	Maximum	Available
	ABC	ABC	ABC
P_0	0,1,0	7,5,3	2,2,2
P_1	2,0,0	3,2,2	
P_2	3,0,2	9,0,2	
P_3	2,1,1	2,2,2	
P_4	0,0,2	4,3,3	

Using Banker's algorithm, how can we avoid the deadlock?

Solution: We will start with the first part of the Banker's algorithm, as follows:

Step 1.1: We will first calculate the Need matrix

$\text{Need}[i] = \text{Max}[i] - \text{Allocation}[i]$

Process	Need
	ABC
P_0	7,4,3
P_1	1,2,2
P_2	6,0,0
P_3	0,1,1
P_4	4,3,1

Step 1.2: Now, we will decide which process should execute first, that is, what are the processes that need resources which are less than available.

Process	Need	Available
	ABC	ABC
P_0	7,4,3	2,2,2
P_1	1,2,2	

So, we will start with P_1 (because its resource need is less than the available resource) and after its completion all the resources will be free; hence, after completion of P_1 the available matrix contents will be

Available
2 2 2
\+ 2 0 0 (resources held by P_1)
4 2 2

Step 1.3: Again let us see the status

Process	Need	Available
	ABC	ABC
P_0	7,4,3	4,2,2
P_1	Complete	
P_2	6,0,0	
P_3	0,1,1	
P_4	4,3,1	

Now, we observe that P_3 can be executed, and is considered to be executed.

Step 1.4: Again, let us see the status

Process	Need	Available
	ABC	ABC
P_0	7,4,3	6,3,3
P_1	Complete	
P_2	6,0,0	
P_3	Complete	
P_4	4,3,1	

So, P_4 executes and similarly we can say that P_2 and then P_0 will be executed, and as a result, the available matrix after execution of each process would be:

Available 6,3,3
P_4 executes 0,0,2 (+)
6,3,5

After the execution of P_2
Available 6,3,5
P_2 executes 3,0,2
9,3,7

After the execution of P_0

Available	9,3,7
P_0 executes	0,1,0
	9,4,7

Thus, the final status of the system under consideration will be:

Process	Need	Available
	ABC	ABC
P_0	Complete	9,4,7
P_1	Complete	
P_2	Complete	
P_3	Complete	
P_4	Complete	

Therefore, the proposed sequence of execution is [P_1, P_3, P_4, P_2, P_0]. Now, we will find whether this sequence of execution is a 'safe sequence'. We will now consider the resource request algorithm (the second portion) of the Banker's algorithm. In the beginning, we have:

Step 2.1:

Work = [2,2,2] (i.e., avail at the start)

Finish = [F, F, F, F, F] (the finish value of all processes is F meaning False)

So, P_1 is executed first, and we observe that

$Need_i <$ Work: [1,2,2] < [2,2,2] holds good, and as a result we have

Work = [4,2,2]

Finish = [F,T,F,F,F] (the finish value of P_1 is T meaning True)

Step 2.2:

Next, we have P_3 executed and in this case the condition $Need_i <$ Work holds good, and as a result we have,

Work = [6,3,3]

Finish = [F,T,F,T,F]

After the execution of P_4, we have

Step 2.3:

Work = [6,3,5]

Finish = [F,T,F,T,T]

After the execution of P_2, we have

Step 2.4:

Work = [9,3,7]

Finish = [F,T,T,T,T]

After the execution of P_0, we have

Step 2.5:

Work = [9,4,7]

Finish = [T,T,T,T,T]

In the whole process, we have never encountered a situation where the condition $Need_i <$ Work does not hold good. Therefore, we can execute our processes in the order [P_1, P_3, P_4, P_2, P_0] without encountering the deadlock.

7.6.6 Deadlock Detection and Recovery

Deadlock detection is a process of detecting whether a deadlock has occurred. Deadlock occurs if neither avoidance nor prevention has been employed. To detect a deadlock, we simulate, in a recursive manner, the most favoured execution of each unblocked process.

1. Any unblocked process can acquire the resources needed for its execution.
2. All the acquired resources are released.
3. These released resources can be used by the previously blocked processes, which are in need.

Repeat the steps 1 to 3 above. If any blocked process remains, they are in a deadlock state.

7.6.6.1 Deadlock Detection

To deal with this concept further, we can discuss the different types of resources available, namely, reusable resources and consumable resources (Table 7.3).

Table 7.3 | Reusable vs. consumable resources

Reusable Resources	Consumable Resources
Number of units are constant	Number of units may vary
Unit is either free or allocated; no sharing	Process may create new units
Process requests, acquires and releases the units	Process may consume the units
Example: memory, devices, files, tables	Example: messages, signals

We have already discussed about resource allocation graphs (Section 7.6.3) and now we will discuss about deadlock detection in two different scenarios:

1. **Single instance of each resource type:** If all the resources have only a single instance, then we use a variant of resource allocation graph known as **wait-for graph** (Fig. 7.15). A deadlock exists in the system if and only if the wait-for graph contains a cycle. To detect deadlocks, the system maintains the wait-for graph and an algorithm is invoked, periodically in order to check for cycle in the graph.
2. **Single instance of each resource type:** The wait-for graph scheme is not applicable to a resource system with multiple instances. The following algorithm is used for deadlock detection in such type of systems (somewhat similar to Banker's algorithm).

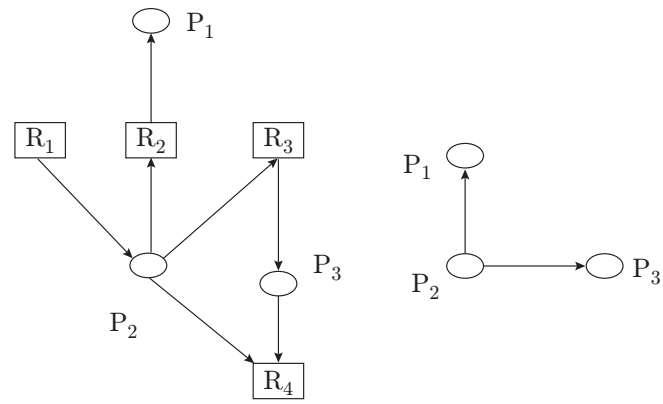


Figure 7.15 | A resource allocation graph and the corresponding wait-for graph.

- **Available:** A vector of length m indicates the number of available resources of each type.
- **Allocation:** An $n \times m$ matrix defines the number of resources of each type currently allocated to each process.
- **Request:** An $n \times m$ matrix defines the current request of each process.

The algorithm is as follows:

- Let Work and Finish be vectors of length m and n, respectively. Initially,

  Work = Available.
- Find an index i such that both the following conditions are false:

  Finish[i] = False

  $\text{Request}_i \leq$ Work. If no such i exists, go to step 4.
- Work = Work + Allocation_i

  Finish [i] = True.

  Go to step 2.
- If Finish[i] = False, for some value of i, $1 \leq i \leq n$, then the system is in deadlock. Moreover, if Finish[i] = False, then it states that the process P_i is deadlocked.

7.6.6.2 Recovery from Deadlock

If at all a deadlock has been detected, any one of the following techniques are used by the OS to recover from the deadlock.

1. **Recovery by Process Termination:** Once the deadlock has occurred, our approach is to overcome the deadlock. This can be achieved if the processes that have been allocated the resources submit them back to the resource pool. This submission is done in a one-by-one fashion until the deadlock

is removed. Now, the question is that which process should submit first or the order of submission. Some of the possible parameters could be:

- ***The priority of a process*:** A process having a low priority could be the first one to give up the resources.
- ***Number of resources held*:** A process holding most of the resources could be the victim for termination or a process having the least number of allocated resources be the first.
- ***Remaining time for completion*:** In this approach, a process which needs more time for completion is terminated.
- ***Process type*:** Whether the processes are interactive (which means that if we terminate one process, all interactive processes will be terminated because it might be possible that one process in execution needs some input from the process that was terminated, hence this will not serve our purpose) or Batch processes (in this case entire batch has to be taken care of).

2. **Recovery by checkpointing and rollback (resource pre-emption):** Another way to recover from deadlock is making use of checkpoints and rollback. These two strategies involve saving the states of a running process at certain intervals, so that they can be terminated and start their execution from any of the saved states. The major overhead is extra memory and complexity to store the intermediate states.

Deadlock recovery is a high cost process, so is used when it is very rare and their might be a need for process migration. Such a concept is more popular in databases.

7.7 THREADS

A process can have more than one thread. A light-weight process is called thread. A process is generally known as a single-threaded process. Multithreaded process has more than one thread.

A process has code, data and files. A thread shares code, data and files with all other threads. A thread has its own registers and stacks which are non-sharable (Fig. 7.16).

7.7.1 Benefits of Threads

1. Improved responsiveness
2. Faster context switching (context-switching time for thread < context-switching time for process)
3. Resource sharing
4. Enhanced throughput of system
5. Effective utilization of multiprocessor systems
6. Economical to implement

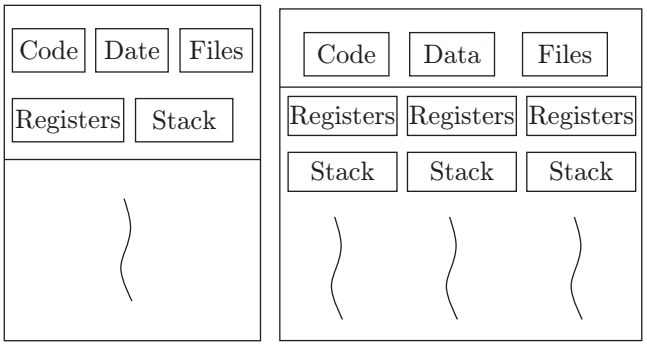


Figure 7.16 | Threads and their types.

7.7.2 Types of Threads

A process can be composed of a single thread or multi-threads. A thread (also known as lightweight process) includes the program counter, stack pointer and its own data area. A thread executes sequentially and is also interruptable. Whereas a process is composed of thread(s) and other system resources (data files, devices, RAM etc). The concept of multithreading is used to perform a number of independent tasks. There are two types of threads, namely,

1. User-level threads
2. Kernel-level threads

User-Level Threads	Kernel-Level Threads
These are created by user or programmer.	Kernel threads are created by operating system.
OS cannot recognize user threads.	These are recognized by OS.
If thread is performing I/O system call, then the entire process will be blocked.	If thread is performing I/O system call, then other threads will continue to execute.
User threads are dependent threads.	Kernel threads are independent threads.
It has less context-switching time.	It has more context-switching time.
User threads do not require hardware support.	Kernel threads require hardware support.

7.7.2.1 fork() Function

The fork is a system call which is used to create the child process. The fork() function returns a value of 0 (zero)

to the newly created child process. The fork() returns a positive integer (process id of child process) to the parent process. The fork() returns a negative value if a child process creation is unsuccessful.

Note: If a program contains n fork() calls, then the number of created child processes are $(2^n - 1)$.

Problem 7.3: Consider the following code and find the total number of processes.

```
main()
{
fork ();
fork ();
fork ();
printf ("hello");
}
```

Solution:

Number of child process $= (2^n - 1) = 2^3 - 1 = 7 = 7$

Total number of process = Parent process + Child process $= (1 + 7) = 8$

7.8 MEMORY MANAGEMENT

The OS manages both the primary memory (RAM) and the secondary memory (hard disk). In the following section, we discuss various primary management algorithms followed by secondary memory management algorithms (hard disk scheduling algorithms). The OS manages the memory in an efficient and orderly manner. The OS has the following main responsibilities related to the memory—Isolation of a process (non-interference), Allocation and management (based upon an algorithm, the user does not decide but knows where the content is), Protection and access control (security) and reliable long-term storage.

7.8.1 Partitioning

It is a technique to divide memory into several parts for process use. Basically, two partitioning methods are used:

1. Fixed **p**artition **s**cheme: Memory is divided into several fixed-sized partitions (also known as static partitioning) (Fig. 7.17). The partition can contain only one process, therefore, the number of partitions is directly equal to the number of processes (concept of multiprogramming). So, whenever a process needs this resource any free partition is allocated and when the process is complete it releases the particular partition to be re-used by another process.

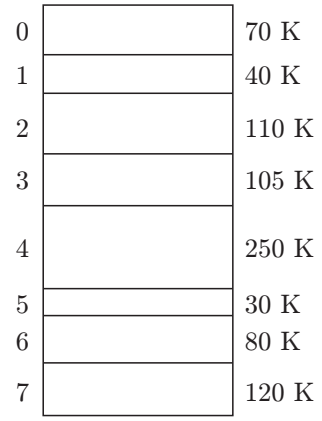


Figure 7.17 | A fixed memory partition scheme.

2. Variable partition scheme: Variable-sized partition is also known as dynamic partitioning (Fig. 7.18). In this scheme, initially the memory will be full contiguous free block. Whenever it gets a request from a process, accordingly partition is made. The processes are accommodated in the memory according to their requirements and thus internal fragmentation is overcome. But external fragmentation exists.

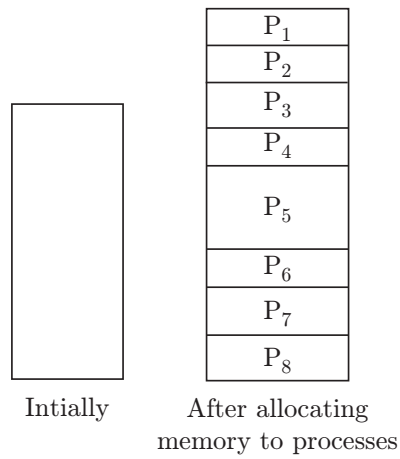


Figure 7.18 | A variable memory partition scheme.

7.8.2 Partition Allocation Policies

Whenever a process requests this resource, the allocation of free memory (partitions or holes) is done based upon different strategies. There are four different strategies:

1. **First Fit:** Allocation is made to the first free partition that is capable of meeting the requirements

of the process. The search can start from either end (up or down).

2. **Next Fit:** A variation of first fit, the allocation is started from the previous partition that was allocated.
3. **Best Fit:** The entire list of free partitions is scanned and the partition that fits best, that is, which results in a lesser free space after allocation is chosen.
4. **Worst Fit:** It is the exact opposite of best fit. The resource is allocated a partition that results in higher free space. The idea behind this strategy is that, a small process when allocated with a bigger partition will be able to make it free in a lesser time, this free partition may be used at a later stage by a process which might require a large partition; so that larger process does not have to wait for resource allocation.

Problem 7.4: Request from the processes is 300K, 25K, 125K and 50K, respectively.

The above request could be satisfied with? (Assume variable partition scheme)

(a) Best fit but not first fit
(b) First fit but not best fit
(c) Both best and first fit
(d) Neither best nor first fit

Solution:

We have to check for only two algorithms best fit and first fit (as given in question). Let us discuss by considering one algorithm at a time, as follows:

(a) In Best Fit:

1. First of all, the 300K jobs will use 350K slot. Free space in the slot will be = 350 − 300 = 50K.
2. Next process of 25K will use remaining 50K slot (available as free from the operation above), as per the best fit algorithm. So, the free space left = 50 − 25 = 25K.
3. Third process of 125K will be accommodated by 150K space, so we are left with 150 − 125 = 25K.
4. Fourth process requires 50K. Now, we have two slots of 25K left but our last process is of 50K size.

Therefore, the need of all the processes is not satisfied.

(b) In First Fit:

1. First process of 300K will use 350K slot. Remaining free space will be = 350 − 300 = 50K.
2. Next process of 25K will use remaining 150K slot (as per first fit algorithm). So, free space left = 150 − 25 = 125K slot.
3. Third process of 125K will be accommodating by 125K slot which is left free after the allocation of the memory to second process.
4. Now, we have only one slot of 50K left and our process requires 50K memory.

Observing, the execution of both the algorithms, we can say that by using the first fit algorithm, all processes can be executed. Therefore, option (b) is the correct answer.

7.8.3 Loading and Linking

A program may consist of different modules which are to be 'linked' and then 'loaded' in the Main Memory for execution. An analogy to understand it could be that a program in C is first linked (as it makes use of different header files) and then loaded in memory for execution. It is hereby worth-mentioning that the image of the program gets loaded into memory.

7.8.3.1 Loading

Loading is copying a program from secondary storage into main memory so it is ready to run. In some cases, loading just involves copying the data from disk to memory, in others it involves allocating storage, setting protection bits or arranging for virtual memory to map virtual addresses to disk pages. Loading can be done through the following two ways:

1. **Static Loading:** It is the technique in which the entire program loads into memory before the start of execution of the program. Some points of static loading are as follows:
   - Inefficient utilization of memory
   - Faster program execution
   - Use of static linking if static loading is used
2. **Dynamic Loading:** It is the technique in which the program is loaded into memory on demand. Some points of dynamic loading are as follows:
   - Efficient utilization of memory
   - Slower program execution
   - Use of dynamic linking if dynamic loading is used

The majority of OS is used dynamic loading and dynamic linking.

7.8.3.2 Linking (or Address Binding)

Association of program instruction and data to the actual physical memory location is called as address binding.

Instruction	Address
I_1	20
I_2	30
I_3	40
I_4	50

Address binding track to increment program counter. There are three types of address binding:

1. **Compiler Time:** If the compiler takes responsibility of associating a program and data to the actual physical memory location, it is called compile-time address binding. The compiler needs to interact with the operating system memory manager to perform compile-time address binding. This type of address binding is done before loading the program into memory.
2. **Load-time Address Binding:** This type of address binding is performed after loading the program into memory. The load-time address binding is performed by a loader.
3. **Run-Time (dynamic) Address Binding:** In the dynamic address binding, address binding is postponed till the time of execution. Dynamic binding is performed by the processor. The majority of OS uses dynamic binding.

7.8.4 Paging

Paging is one of the memory-management schemes where a computer can store and retrieve data from secondary storage for use in the main memory (Fig. 7.19). This technique maps processor-generated logical address to physical address. As we know,

Logical address > Physical address

There are few terms used in paging:

1. **Logical Address Space (LAS):** Defines the size of logical memory.
2. **Physical Address Space (PAS):** Defines the size of physical memory.

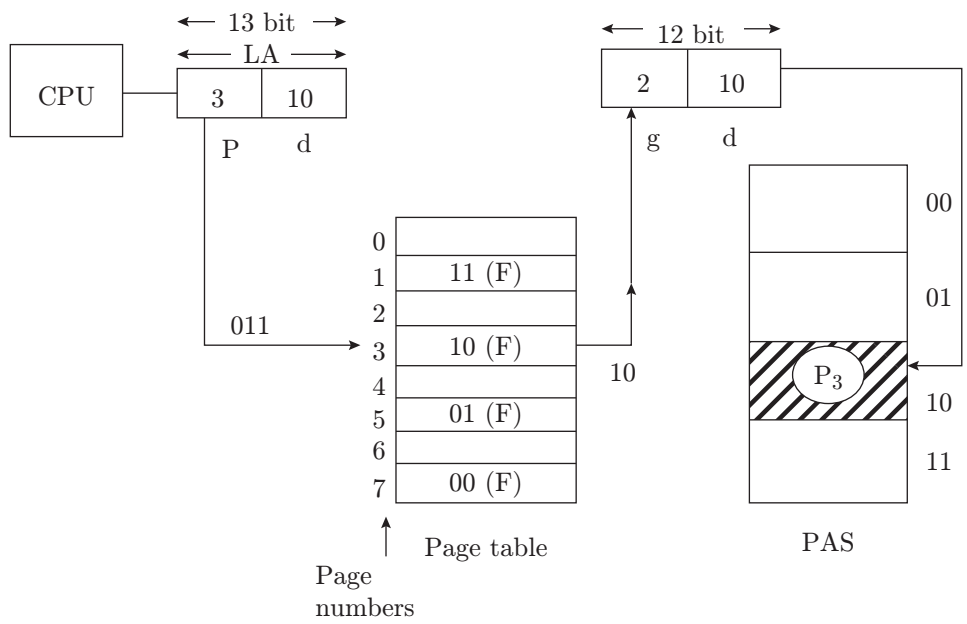


Figure 7.19 | A memory paging scheme.

3. **Logical address (LA):** Number of bits to represent logical address space.
4. **Physical address (PA):** Number of bits to represent physical address space.
5. **Page:** Logical memory is divided into equal parts called pages.
6. **Frame:** Physical memory is divided into equal parts called frames.

Assume that processor-generated logical address (LA) is 13-bit long and physical address (PA) is 12-bit long, then

$$\text{Logical address space} = 2^{13} \text{ words} = 2^3 \times 2^{10} \text{ words} = 8\text{K words}$$

$$\text{Physical address space} = 2^{12} \text{ words} = 2^2 \times 2^{10} \text{ words} = 4\text{K words}$$

The logical address is divided into equal size pages. In our case, page size is given as 1K words. So,

$$\text{Number of pages} = \frac{\text{LAS}}{\text{Page size}}$$

$$\text{Number of pages} = \frac{8\text{K}}{1\text{K}} = 8$$

To represent eight pages, we required 3 bits. So, logical address is divided into two parts P (page number or number of bits required to represent the pages of LAS) and d (page size or number of bits required to represent the page size of LAS). This means LA is divided into $P = 3$ and $d = 10$ bits.

The physical address is divided into equal-size frames. Generally, frame size = page size, given as 1K words. So,

$$\text{Number of frames} = \frac{\text{PAS}}{\text{Frame size}}$$

$$\text{Number of pages} = \frac{4\text{K}}{1\text{K}} = 4$$

To represent four frames, we required 2 bits. So, physical address is divided into two parts f (frame number or number of bits required to represent the frames of PAS) and d (frame size or number of bits required to represent the frame size of PAS). This means PA is divided into $f = 2$ and $d = 10$ bits.

Figure Description:

1. Whenever we apply paging, we have to maintain a page table.
2. Number of entries in page table is same as the number of pages in LAS.
3. The page table entries contain frame number (binary format).
4. P is mapped to page table and corresponding entries go to PAS.

Problem 7.5: Consider the following system:

Number of pages = 2K

Page size = 4K words

Physical address = 18 bits

Calculate logical address space and number of frames.

Solution:

1. We know that

$$\text{Number of pages} = \frac{\text{LAS}}{\text{Page size}}$$

So, LAS = Number of pages × Page size

$$\text{LAS} = (2\text{K} \times 4\text{K}) \text{ words} = 2^{11} \times 2^{12} \text{ words} = 2^{23} \text{ words} = 8\text{M words}$$

2. $$\text{Number of frames} = \frac{\text{PAS}}{\text{Frame size}} = \frac{2^{18}}{2^{12}}$$

Because frame size = page size, the number of frames is 2^6 frames = 64 frames.

Problem 7.6: Consider a system with LA = 32 bits, PAS = 64 MB and page size is 4 KB. The memory is byte addressable. Page table entry size is 2 bytes. What is the approximate page table size?

Solution:

LA = 32 bits. So, LAS = 2^{32} bytes

$$\text{Number of pages} = \frac{\text{LAS}}{\text{Page size}} = \frac{2^{32}}{2^{12}} = 2^{20}$$

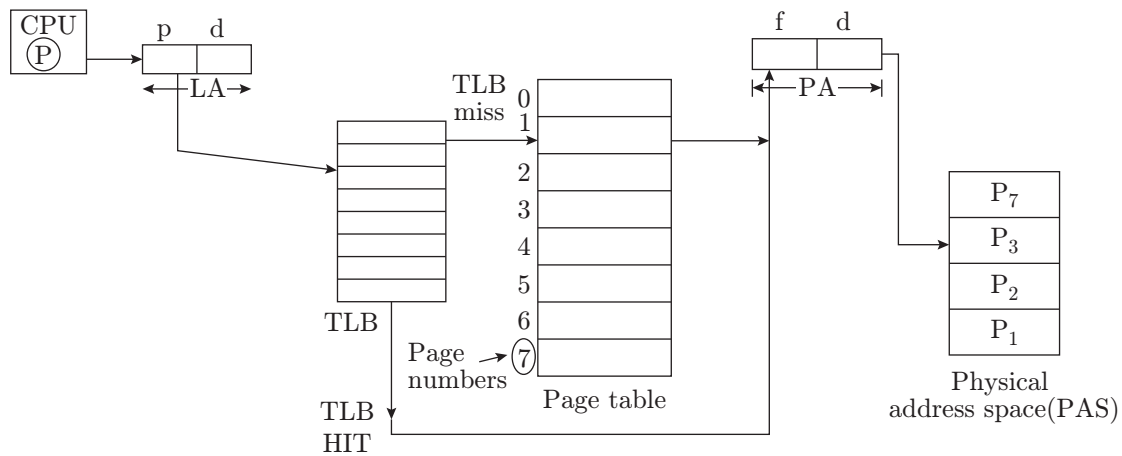
Page table size = Number of entries × Page table entry size ⇒ $2^{20} \times 2$ bytes = 2MB (because number of page table entries = number of pages)

7.8.4.1 Paging Performance with TLB

Translation lookaside buffer is a cache memory used by memory management unit to improve the translation speed of virtual address. The use of virtual memory becomes time consuming without TLB. This buffer keeps the record of all the recently accessed pages.TLB works as shown in Fig. 7.20.

CPU first searches the page in TLB, if the page is found in TLB then physical address is calculated. But if there is TLB miss, then CPU goes to the page table. It checks the page in page table, if found then it calculates the effective address and make the entry in TLB too. But if page is not found in page table, it generates the page fault.

The following points show how the performance is improved by using TLB:



LA ⟶ Logical address

PA ⟶ Physical address

TLB ⟶ Trantition look-aside buffer

P = Page number
f = Frame number
d = Page size or frame size

Figure 7.20 | TLB.

1. If page table is kept in the main memory and the main memory access time is M, then formula for effective memory access time for page table is

$$E_{\text{MAT}} = 2M$$

2. Translation look-aside buffer (TLB) is added to improve the performance of page table. TLB contains the frequently referenced page number and corresponding frame number. If the TLB access time is C and TLB hit ratio is X, then effective memory access time for page table is

$$E_{\text{MAT}} = X(C+M) + (1-X)(C+2M)$$

Problem 7.7: Consider a system with TLB access time 20 ns and main memory access time 100 ns. Calculate the effective memory access time if TLB hit ratio is 95%.

Solution:

Given that $C = 20$ ns, $M = 90$ ns, $X = 95\%$. Without TLB effective memory access time

$$E_{\text{MAT}} = 2M = 2 \times 100 = 200 \text{ ns}$$

With TLB, effective memory access time

$$\begin{aligned} E_{\text{MAT}} &= X(C+M) + (1-X)(C+2M) \\ &= 0.95(20+100) + (1-0.95)(20+2\times 100) \\ &= 114 + 11 = 125 \text{ ns} \end{aligned}$$

7.8.5 Multi-Level Paging

Paging technique faces problem of large memory requirement due to the large logical address space. Multi-level paging technique provides the solution to the problem that occurred in paging technique (Fig. 7.21).

Example:

Assume that processor-generated logical address (LA) is 35-bit long and physical address (PA) is 26-bit long, then

$$\begin{aligned} \text{Logical address space} &= 2^{35} \text{ words} \\ &= 2^5 \times 2^{30} \text{ words} \\ &= 32\text{G words} \end{aligned}$$

$$\begin{aligned} \text{Physical address space} &= 2^{26} \text{ words} \\ &= 2^6 \times 2^{20} \text{ words} \\ &= 64\text{M words} \end{aligned}$$

The logical address is divided into equal size pages. In our case, page size is given as 4K words. So,

$$\begin{aligned} \text{Number of pages} &= \frac{\text{LAS}}{\text{Page size}} \\ &= \frac{32\text{G words}}{4\text{K words}} = 8\text{M pages} \end{aligned}$$

To represent 8M pages, we require 23 bits. So, the logical address is divided into two parts P (page number or number of bits required to represent the pages of LAS) and d (page size or number of bits required to represent the page size of LAS). This means LA is divided into $P = 23$ and $d = 12$ bits. And P is further divided into two parts P_1 and P_2.

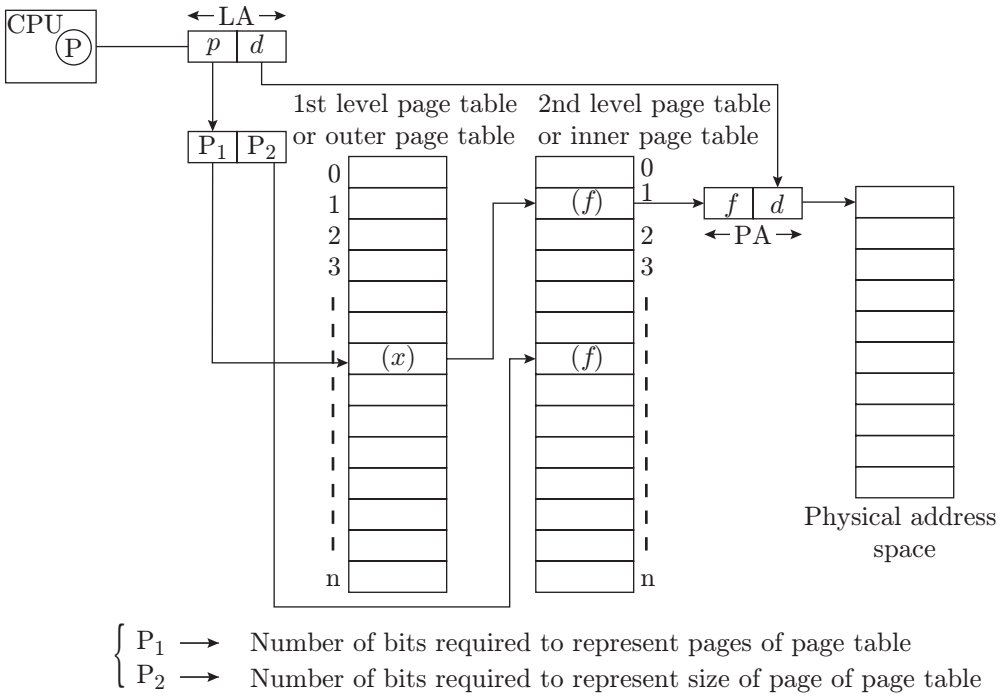


Figure 7.21 | Multi-level paging scheme.

1. P_1 is the page number or number of bits required to represent the pages of page table.
2. P_2 is the page size or number of bits required to represent the page size of page table.

The physical address is divided into equal-size frames. Generally, frame size = page size, given as 4K words. So,

$$\text{Number of frames} = \frac{\text{PAS}}{\text{Frame size}} = \frac{64\text{M words}}{4\text{K words}} = 2^{14}$$

$$= 16\text{K frames}$$

To represent 16K frames, we required 14 bits. So, physical address is divided into two parts f (frame number or number of bits required to represent the frames of PAS) and d (frame size or number of bits required to represent the frame size of PAS). Means PA is divided into $f = 14$ and $d = 12$ bits.

Problems associated with multilevel paging: Memory requirements are low for multilevel paging but address translation time is more due to more number of levels. This can be solved by using Translation lookaside buffer.

7.8.5.1 Inverted Paging

In paging technique, every process has its own page table and because of this there is a problem of huge page table size. To overcome overhead of maintaining the page table for every page, inverted paging is used (Fig. 7.22). In inverted paging, only one table is maintained for all

Problem 7.8: Consider a system with logical address = 34 bits, physical address = 29 bits and page size = 16 KB. The memory is byte addressable and the page table entry size is 8 bytes.

(a) What is the conventional page table size?
(b) What is the inverted page table size?

Solution:

(a) Conventional page table size

$$\text{Number of pages} = \frac{\text{LAS}}{\text{Page size}} = \frac{2^{34}}{2^{14}} = 2^{20}$$

Page table size = Number of pages × Page table entry size = 2^{20} × 8 bytes × 8 B = 8 MB

(b) Inverted page table size

$$\text{Number of frames} = \frac{\text{PAS}}{\text{Frame size}} = \frac{2^{29}}{2^{14}} = 2^{15}$$

Page table size = Number of frames × Page table entry size

$$= 2^{15} \times 8 \text{ bytes} \times 8 \text{ B}$$

$$= 256 \text{ KB}$$

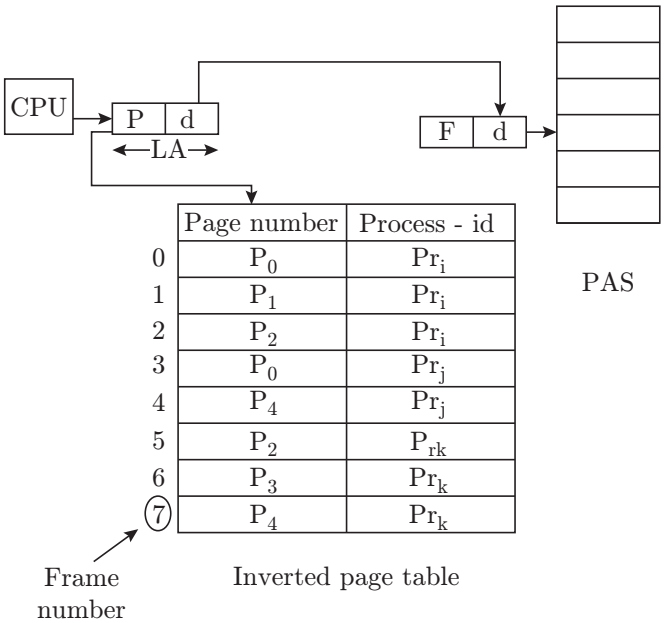


Figure 7.22 | An inverted paging scheme.

the processes. The number of entries in the inverted page table is the same as number of frames in the PAS.

7.8.6 Segmentation

Paging technique does not follow user's view of memory allocation. To achieve user's view of memory allocation, segmentation technique is used (Fig. 7.23). In this technique, logical address space is divided into several segments. Segments may vary in size.

S is the segment number; it is equal to the number of bits required to represent the segment of LAS.

D is the number of bits required to represent the segment size. It is also known as segment offset. So,

Number of segments in segment table
= Number of segments in LAS

Segmentation also suffers from external fragmentation.

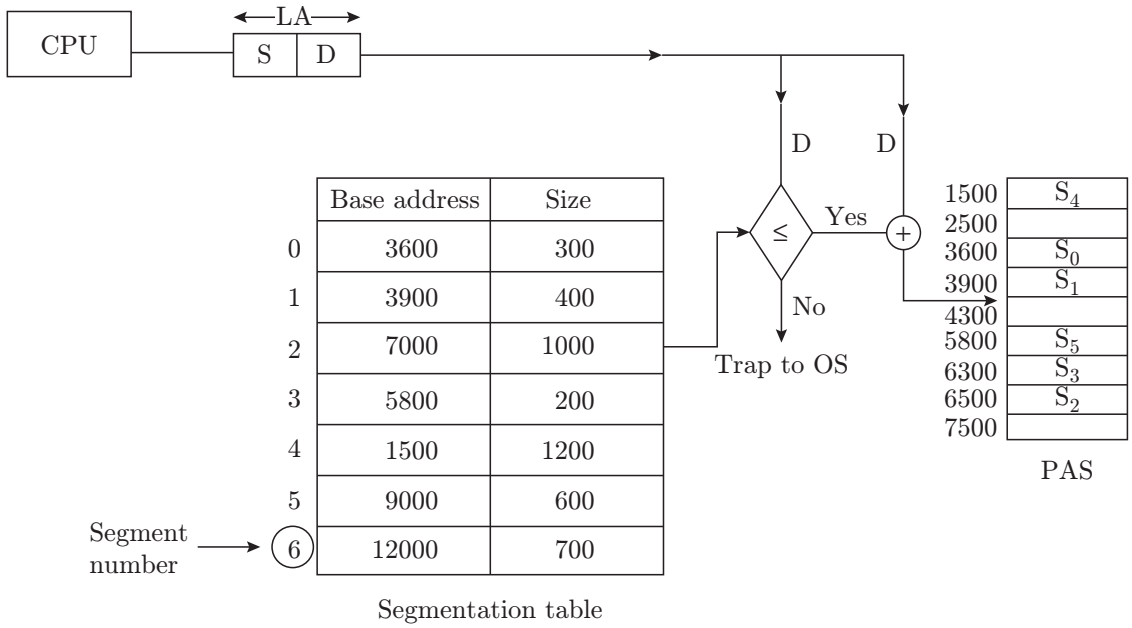


Figure 7.23 | A memory segmentation scheme.

7.8.7 Virtual Memory

Virtual memory is a memory management technique that is implemented using both hardware and software. It maps memory addresses used by a program, called virtual addresses, into physical addresses in computer memory. Virtual memory gives an illusion to the programmer that programs which are larger in size than actual memory can be executed. Virtual memory can be implemented with demand paging.

7.8.7.1 Demand Paging

Demand paging is a method of virtual memory management (Fig. 7.24). In a system that uses demand paging, the operating system copies a disk page into physical memory only if an attempt is made to access it and that page is not already in memory (i.e., if a page fault occurs).

Figure Description:

Step 1: Processor is looking for page number 4. But page 4 is not available in the memory, that is, page fault. Without page 4 currently executing, program will stop.

Step 2: The signal is send to OS that processor is looking for page 4, which is currently not available in the main memory.

Step 3: OS will search for required page 4 in LAS.

Step 4: OS will replace the page from LAS to PAS by using the page replacement algorithm.

Step 5: OS updates the page table accordingly.

Step 6: The signal is sent to the processor, which says the required page is brought into the memory and then the processor continues the program execution.

7.8.7.2 Effective Memory Access Time

The time taken to service a page fault is called as page fault service time. If the main memory access time is M, page fault service time is S, which is much larger than M ($S \gg M$) and page fault rate is P, then

Effective memory access time,

$$E_{\text{MAT}} = P \times S + (1 - P) \times M$$

Problem 7.9: Consider a system with page fault service time (S) = 100 ns, main memory access time (M) = 20 ns, and page fault rate (P) = 65%. Calculate effective memory access time.

Solution:

$$\begin{aligned} E_{\text{MAT}} &= P \times S + (1 - P) \times M \\ &= 0.65 \times 100 + (1 - 0.65) \times 20 \\ &= 0.65 \times 100 + 0.35 \times 20 \\ &= 65 + 7 = 72 \text{ ns} \end{aligned}$$

Table 7.4 summarizes the different memory management techniques.

7.8.8 Page Replacement Algorithms

Whenever there is a demand of any page which is not in memory then that must be loaded. But if there is no free frame in memory, one page is selected for replacement. Page replacement algorithm decides which page is to be replaced at the time of page fault.

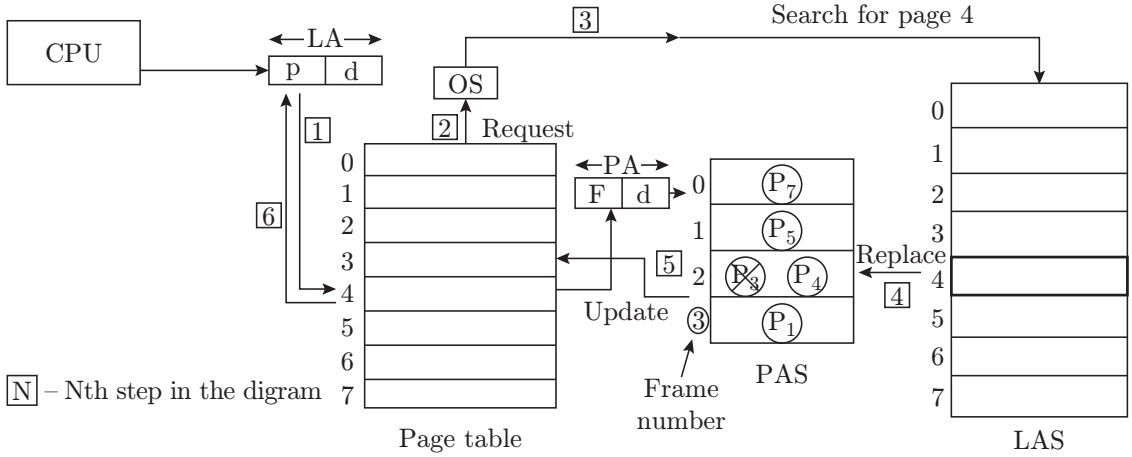


Figure 7.24 | A demand paging scheme.

Table 7.4 | Comparison of different memory management techniques

Technique	Pros	Cons
Fixed Partitioning	Simple, No overhead on OS	Internal fragmentation
Dynamic Partitioning	No internal fragmentation, Efficient use of RAM	External fragmentation, Inefficient use of CPU
Paging	No external fragmentation	Very little internal fragmentation
Segmentation	No internal fragmentation, Less overhead as compared to dynamic partitioning	External fragmentation
Virtual Memory Paging	No external fragmentation, High degree of multiprogramming	Overhead on OS
Virtual Memory Segmentation	No internal fragmentation, Higher degree of multiprogramming, Support for protection and sharing	Extra overhead on OS

7.8.8.1 First In, First Out

In this technique, the new page will replace the page which is entered first in the memory frame. This means that page is selected which has been in the main memory for the longest time. Queue data structure is used.

Example 7.8

We have a reference string: 1, 2, 2, 1, 0, 3, 1, 2, 4, 1, 0

Reference String

1	2	2	1	0	3	1	2	4	1	0

1	1	1	1	1	3	3	3	4	4	4
	2	2	2	2	2	1	1	1	1	0
				0	0	0	2	2	2	2
Miss	Miss	Hit	Hit	Miss	Miss	Miss	Miss	Miss	Hit	Miss

Step 1: Suppose we have three empty frames. We insert values in frames 1, 2. Then there is a hit for 2 and 1. Because they are already present in frames, so the next 0 will be inserted in frame.

1
2
0

Step 2: When page 3 is referenced, there is no free frame in memory; 3 will be replaced with the page which is in memory for the longest time. That page is 1.

~~1~~	3
2	
0	

Step 3: Next page referred is 1, which will be replaced with 2 and so on.

~~3~~ 4
~~2~~ ~~1~~ 0
~~0~~ 2

Number of hits: 2, 1, 1 = 3

Number of page faults = 1,2,0,3,1,2,4,0 = 8

$$\text{Hit ratio} = \frac{\text{Number of hits}}{\text{Total page references}} = \frac{3}{11}$$

Problems with FIFO technique: FIFO suffers from Belady's anomaly. According to **Belady's anomaly**, sometimes with the increase in number of frames, page fault rate also increases. The performance of the OS drops instead going up (Fig. 7.25).

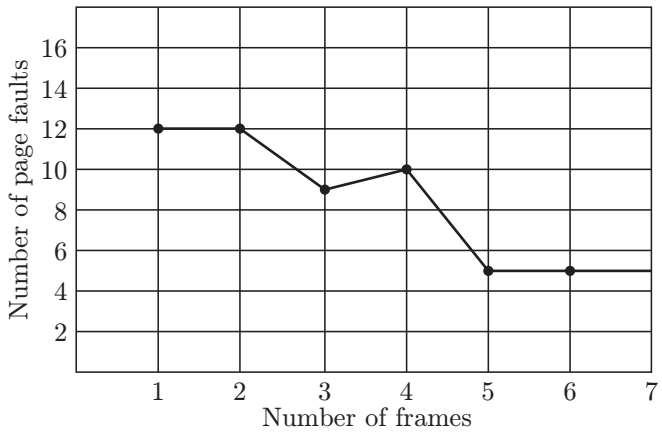


Figure 7.25 | Belady's anomaly.

7.8.8.2 Optimal Replacement Algorithm

In the case of optimal page replacement algorithm, page will be replaced that will not be referenced in future.

Reference String

1	2	2	1	0	3	1	2	4	1	0

1	1	1	1	1	1	1	1	1	1	0
	2	2	2	2	2	2	2	4	4	4
				0	3	3	3	3	3	3
Miss	Miss	Hit	Hit	Miss	Miss	Hit	Hit	Miss	Hit	Miss

Step 1: We have three empty frames. We insert values in frames 1, 2. Then there is a hit for 2 and 1. Because they are already present in frames, so the next 0 will be inserted in the frame.

1
2
0

Step 2: When page 3 is referenced, there is no free frame in memory; 3 will be replaced with the page which will not be used for long. That page is 0.

1	
2	
~~0~~	3

Step 3: Next page referred is 1, 2 will be hit. Now page 4 is demanded. No free frame is there in memory. So it will be replaced by either 2 or 3. Both are not required in future. To decide among the two, FIFO is used. 2 will be replaced with 4.

1	
~~2~~	4
3	

Step 4: Next in reference string is 1. This is already present, so it is a hit. Next is 0, 0 will also use FIFO to break the tie among 1, 4, 3. And 0 will be replaced with 1.

~~1~~	0
4	
3	

- When there is a tie for replacement in optimal page replacement (OPR), FIFO is used to break the tie.

  Number of page faults = 6

  Number of hits = 5
- OPR is the most efficient algorithm. Lowest number of page faults occurs for this algorithm.
- Realization of this algorithm is difficult, because OS needs to know that how long it will take to refer that page again.

7.8.8.3 Least Recently Used

Least recently used (LRU) page replacement algorithm replaces the page which has not been used recently.

Reference String

1	2	2	1	0	3	1	2	4	1	0

1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	3	3	3	4	4	4
				0	0	0	2	2	2	0
Miss	Miss	Hit	Hit	Miss	Miss	Hit	Miss	Miss	Hit	Miss

Step 1: We have three empty frames. We insert values in frames 1, 2. Then there is a hit for 2 and 1. They are already present in frames. So, next 0 will be inserted in frame.

1
2
0

Step 2: When page 3 is referenced, there is no free frame in memory. So, 3 will be replaced with the page which is not recently used. That page is 2.

1	
~~2~~	3
0	

Step 3: Next page referred is 1, which will be hit. Now page 2 is demanded. No free frame is there in memory. So it will be replaced by 2, due to LRU page.

1	
3	
~~0~~	2

Step 4: Next in reference string is 4. This will be replaced with 1.

1	
~~3~~	4
2	

Step 5: Next in reference string is 1. This will be hit. Next, page 0 will be replaced with 2.

1	
4	
~~2~~	0

Number of hits = 4

Number of page faults = 7

- Counter is maintained after each page reference with the last access time.
- Expensive.

Frame allocation: Each process needs minimum number of pages. The allocation can be done in the following ways.

Fixed allocation:

- **Equal allocation:** Each process gets equal number of frames. For example, there are 80 frames and 4 processes. So, each process will get 20 frames.
- **Proportional allocation:** This is a dynamic way of frame allocation. Allocation is done according to the size of process.

  Allocation for $P_i = \frac{s_i}{S} \times m$

  Here P_i: ith process

  s_i: size of ith process

  S: total size

  m: total number of frames

Priority allocation: This method uses priority instead of size to allocate pages.

7.8.8.4 Global versus Local Replacement

As discussed earlier, a page replacement algorithm decides which page is to be replaced at the time of page fault. Now, the questions that arise are:

1. Which page (that we have seen through different page replacement algorithms, above), and
2. From where (to be discussed in the following section).

Global Replacement: The process can replace any page available in the main memory when page fault occurs. All the pages in the system are viewed as common pool. Further, fixed allocation method cannot be implemented using this technique. One of the major problems with this scheme is that each process will spend more time in recovering lost pages, which is called 'thrashing'.

Local Replacement: When the page fault occurs, local replacement algorithm can replace the pages from the same process. Other processes will not suffer by this scheme.

1. Local page replacement does not remove thrashing, but it reduces it to some extent.
2. Working set model is solution of thrashing.

7.8.8.5 Thrashing

A program that causes page faults more frequently is said to be thrashing, which means that the system more busy in dealing with page faults instead of doing some constructive work. This leads to the decrease in throughput of the system.

Further, it has been observed that there is a degree of multiprogramming, which is optimal for any system's performance, because if we increase the degree of multiprogramming it might result in thrashing (Fig. 7.26).

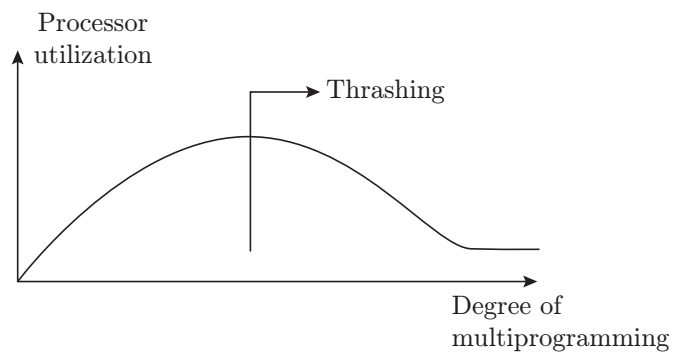


Figure 7.26 | Thrashing in multiprogramming

There are different remedies to overcome thrashing, one of them is discussed below:

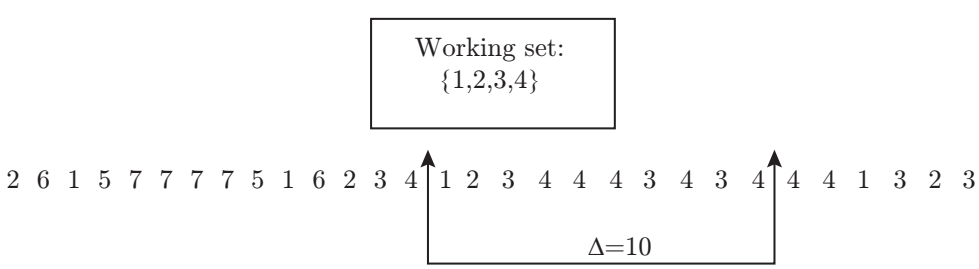
Working set model: According to Peter Denning, working set is a collection of pages that are being used by some process during a finite time interval. This gives an idea about approximate set of pages the process will require in future.

$$\Delta = \text{Working set window}$$

Example:

Window size = 10

As shown in the figure below, active pages are in set.



7.9 FILE SYSTEM

7.9.1 Directory Structures

The directory structures are used to manage and organize data. This can be done in the following ways:

1. The disks can be split into one or more partitions. These partitions provide separate areas within one disk and are treated as a separate storage devices.
2. Each partition contains information about files. The device directory contains information such as name, location, size and type.

7.9.1.1 Single-Level Directory

It is the simplest directory structure (Fig. 7.27), in which there is one directory which contains all the files. Major limitation lies in the number of files (memory size) a directory can hold and there is more than one user that needs to access the files.

7.9.1.2 Two-Level Directory

To overcome the above problems, a separate directory can be created for each user, known as – User File Directory (UFD). The major limitation is the sharing of files among the users. This scheme is represented in Fig. 7.28.

7.9.1.3 Tree-Structured Directories

Directory structure is extended to a tree of arbitrary height. In this, users can create their own subdirectories and manage their own files. There is a root directory too. Such a type of file system is more popular in present day OS. A path name is the path from the root, through all the subdirectories, to a specified file. There are two types of path names:

1. **Absolute path name:** It begins at the root and follows a path down to the specified file, giving directory names on the path. For example, root/spell/mail/sys/units.
2. **Relative path name:** It defines a path from the current directory. Considering the same example as above, let us suppose that we are working in a current directory named root/spell/mail, then the relative path can be sys/units, instead of specifying the full path name.

7.9.1.4 Acyclic Graph Directories

An acyclic graph is a graph with no cycles and is a natural generalization of the tree-structured directory scheme (Fig. 7.29). A tree structure prohibits the sharing of files or directories, whereas an acyclic graph allows directories to have shared subdirectories and file, which means the same file or subdirectory may be in two different directories. With a shared file, only one actual file exists, so any changes made by one person are immediately visible to the other.

7.9.2 Allocation Methods

The direct access nature of disks allows us flexibility in the implementation of files and usually many files are stored on the same disk. The main problem is how to allocate space to these files so that disk space is utilized effectively and files can be accessed quickly. Three

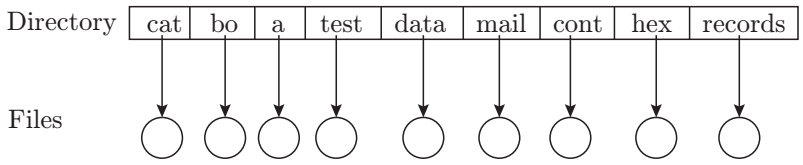


Figure 7.27 | A single-level directory.

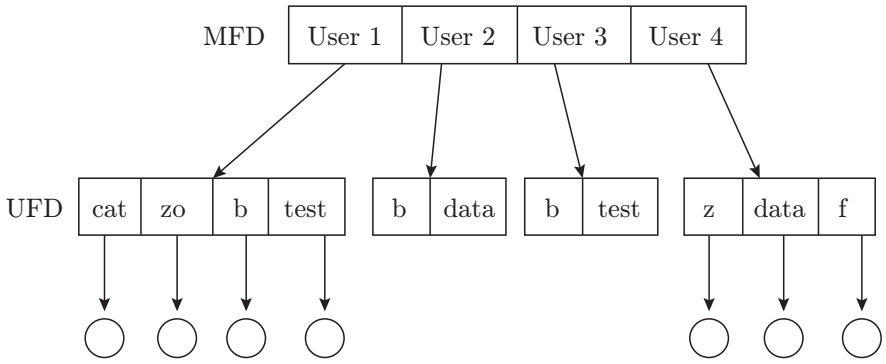


Figure 7.28 | Two-level directory structures.

Note: User 2 and User 4 have a file named data, thus name-collision problem is solved.

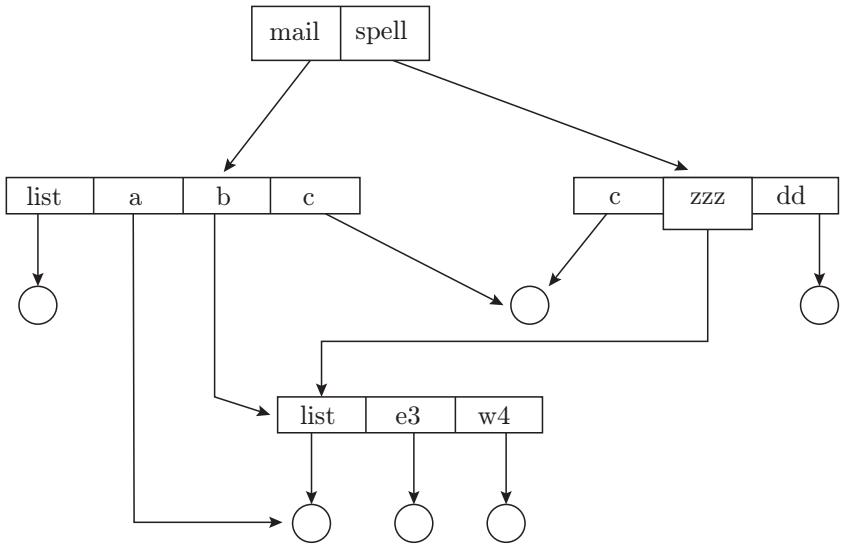


Figure 7.29 | Acyclic graph directory structures (the *sub* directory list is shared by *zzz* and *b*).

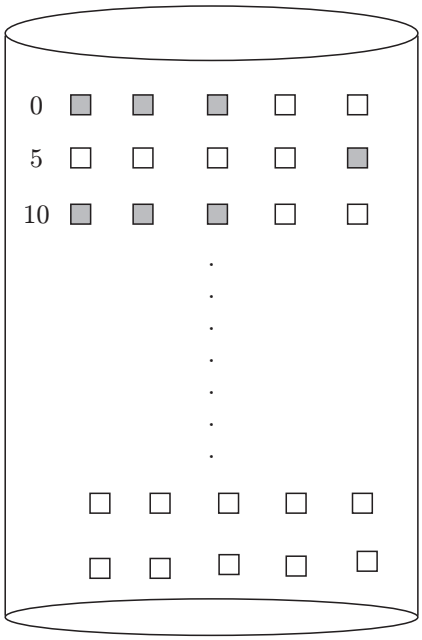
widely used methods of allocating disk space are discussed in the following text.

7.9.2.1 Contiguous Allocation

In contiguous allocation method, each file occupies a set of contiguous blocks on the disk. All the blocks are continuously kept together. There is an index that specifies the starting address and the length (size) indicating the blocks (Fig. 7.30). Different allocation strategies are first-fit, best-fit or worst-fit (already discussed earlier). This technique suffers from internal and external fragmentation. It supports sequential and random access.

7.9.2.2 Linked Allocation

In linked allocation, each data block contains the address of the next block in the file (Fig. 7.31). In contrast to contiguous allocation, the disk blocks are scattered on disk. The directory entry contains an index, which points to the starting block as well as the ending block. Each block in turn contains an address of the next block, similarly, all the blocks are linked together. The last block does not have a



File	Start	Length
Cnt	0	3
List	9	4

Figure 7.30 | Contiguous allocation of disk space.

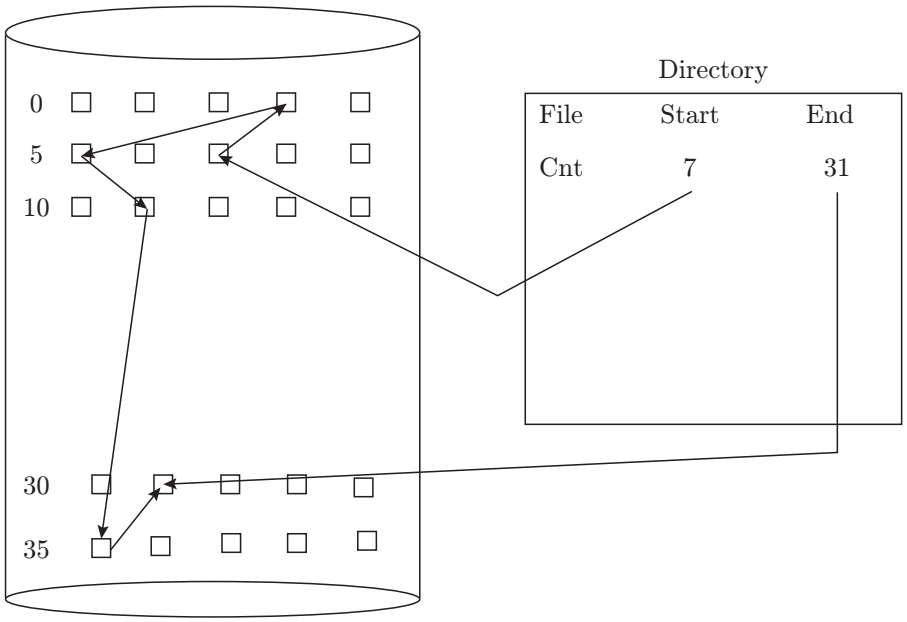


Figure 7.31 | Linked allocation of disk space.

next block address (just as linked list). The major advantage being, no external fragmentation with a limitation of slow access (because we have to traverse all the blocks in the list before the required block). Another major drawback is the breaking of the link (similar to dangling references).

7.9.2.3 Indexed Allocation

There is an index block that contains the address of every block that contains the information associated with that file (Fig. 7.32). This approach is better as provides dynamic access without any external fragmentation and there is very less loss of information in case of breaking of links. Let us take an example of a loss of pointer, in this case only information about that block will be lost whereas in linked allocation information stored in the rest of blocks will also be lost.

It is hotly debated how large the index block should be. Every file must have an index block, so we want the index block to be as small as possible. If the index block is too small, however, it will not be able to hold enough pointers for a large file, and a mechanism will have to be

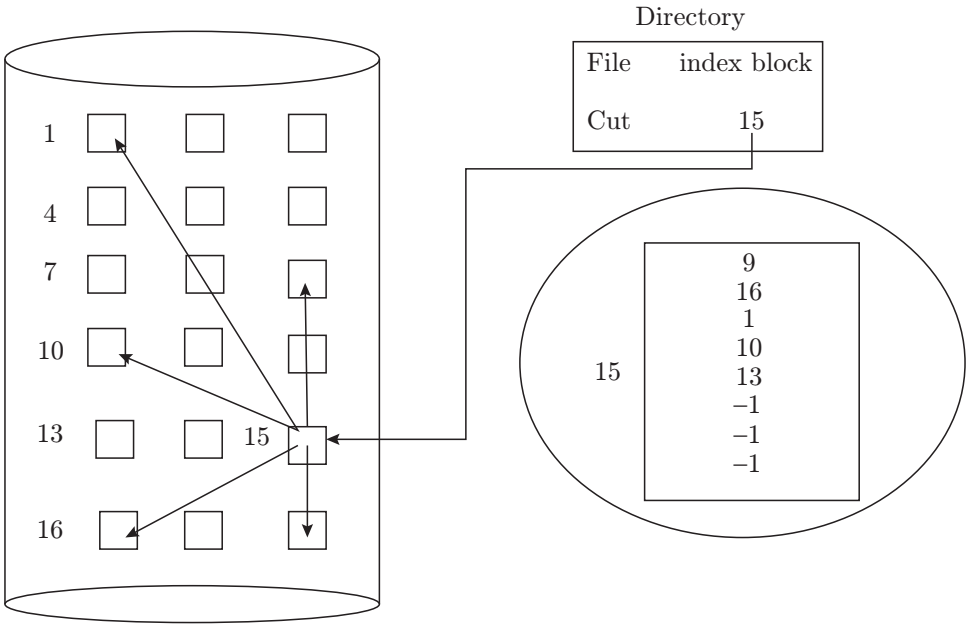


Figure 7.32 | Indexed allocation of disk space (-1 represents not used).

available to deal with this issue. The following are the possible solutions:

1. **Linked scheme:** An index block is normally of one disk space. Thus, it can be read and written directly by itself.
2. **Multilevel index:** The first index block contains the set of pointers to secondary index blocks, which in turn contain pointers to the actual data blocks.
3. **Combined scheme:** It is used in the UNIX File System (UFS) i-nodes (see Fig. 7.33, which represents the UFS i-node scheme), where first few data block pointers of the index block are stored directly in the i-node and then single, double and triple indirect pointers provide access to more data blocks as needed.

   Total size of file system

$$= \left\{ \text{Number of direct DBA} + \left(\frac{\text{DBA size}}{\text{DBA}}\right) + \left(\frac{\text{DBA size}}{\text{DBA}}\right)^2 + \left(\frac{\text{DBA size}}{\text{DBA}}\right)^3 + \cdots \right\} \times \text{DB size}$$

   where DBA is disk block address.

- Number of disk block possible in one disk block = $\left(\frac{\text{DBA size}}{\text{DBA}}\right)$
- If asked for maximum file size, then calculate for highest power only such as Problem 7.10.

Problem 7.10: Consider the UNIX i-node which uses 12 direct DBAs, 1 single indirect, 1 double indirect, 1 triple indirect. The disk block address requires 32 bits and the disk block size is 1 KB. What is the maximum file size?

(a) 8 GB (b) 16 GB
(c) 32 GB (d) 64 GB

Solution:

$$\text{Maximum file size} = \left(\frac{\text{DBA size}}{\text{DBA}}\right)^3 \times \text{DB size}$$

$$= \left(\frac{1\text{ KB}}{4\text{ bytes}}\right)^3 \times 1\text{ KB}$$

$$= \left(\frac{2^{10}}{2^2}\right)^3 \times 2^{10}\text{ bytes}$$

$$= 16\text{ GB}$$

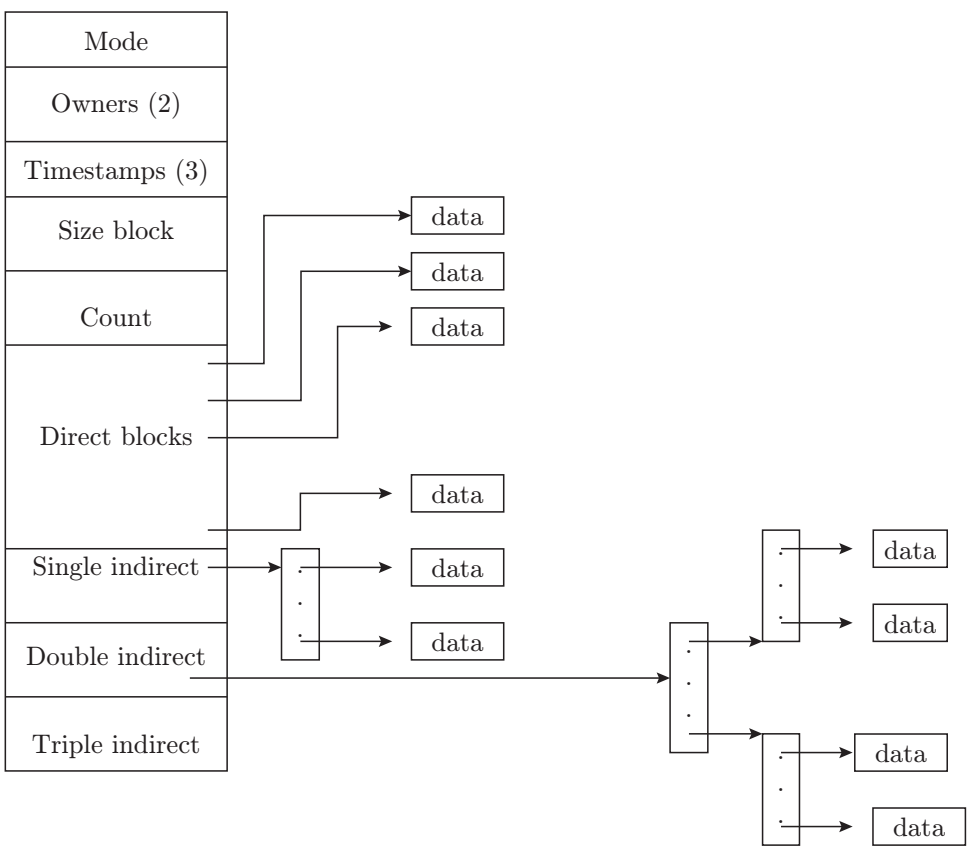


Figure 7.33 | The UNIX i-node combined scheme.

7.10 I/O SYSTEMS

There are different devices that are connected with a computer. These devices are broadly classified as:

1. **Machine readable:** Most of the electronic equipments such as hard disk, floppy disk, CD, USB, bar code reader, actuators, etc.
2. **Human readable:** Devices such as printer, keyboard and Video Display Unit (VDU), usually called monitor screen (but is different from monitor, as used in OS). Such devices transfer the data at a slower rate as compared to the data transferred between the memory and CPU. To overcome such differences in the rate of transfer, the concept of SPOOLing (Simultaneous Peripheral Operations OnLine) is used.
3. **Communication:** Network Interface Cards (NIC) are used for network communication or modems, etc.

There are different techniques used for I/O, such as Programmed I/O, Interrupt driven I/O and DMA I/O. These are covered in detail in Chapter 2.

Problem 7.11: Consider a disk which has 16 plotters, each plotter is divided into two surfaces. Each surface has 1K tracks and each track has 512 sectors. Each sector can store 2 KB data.

(a) Find the capacity of the disk.
(b) How many bits are required to identify a sector?

Solution:

(a) Capacity of disk = (Plotters × Surface × Tracks × Sector × Sector Size) bytes

$= (16 \times 2 \times 1024 \times 512 \times 2048)$ bytes

$= (32 \times 2^{10} \times 2^{9} \times 2^{11})$ bytes

$= 32 \times 2^{3}$ bytes

$= 32$ GB

(b) Number of bits required to identify a sector

= (Bits for plotter + bits for surface + bits for tracks + bits for sectors)

$= (4 + 1 + 10 + 9)$ bits

$= 24$ bits

7.10.1 Disk Structure

Due to advent of technology, the gap in the access speeds of a processor (as well as primary memory, i.e., RAM) and disks (the secondary memory, i.e., hard disks) have increased. In order to provide a faster access to user, it is equally important for the OS to manage the disks efficiently by making use of suitable algorithms. A simple disk structure is shown in Fig. 7.34.

Note: Number of cylinders per track = Number of tracks per surface

7.10.1.1 Time Related to Disk

1. **Seek time:** The amount of time required to move the read/write head from its current position to desired track.
2. **Rotational latency:** The amount of time to rotate the track when the read/write head comes to desired sector position. In simple disk, rotational latency is the time to rotate 1/2 disk to the access.
3. **Transfer time:** The amount of time taken to transfer the required data is called as transfer time.

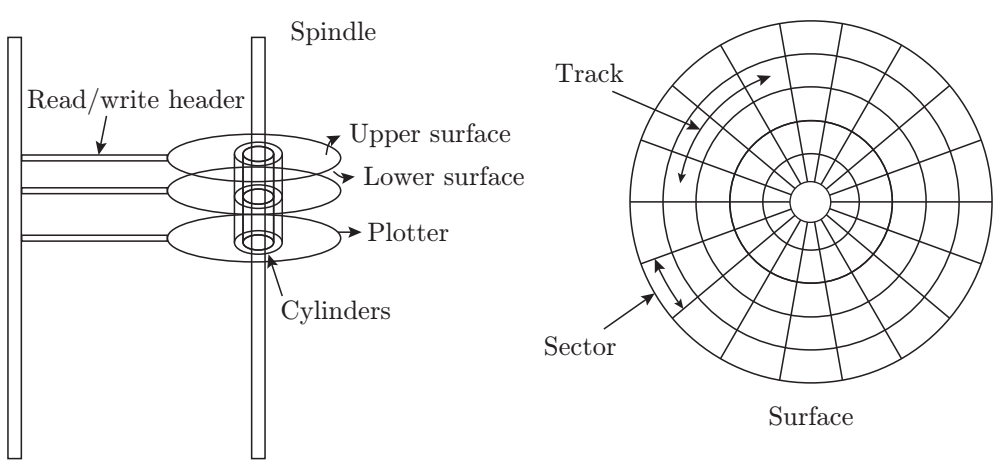


Figure 7.34 | The disk structure.

Transfer time depends on the total size of the track and the rotational rate of the disk.

4. **Transfer Rate:** The number of bytes transferred in one unit time is called as the transfer rate of the disk.

7.10.1.2 Disk Interleaving

When the disk rotation speed is higher, then read/write head skips sector 2 after reading sector 1 because it does not get enough time to read the next sector. This problem is solved by the disk interleaving technique (Fig. 7.35):

1. **Single interleaving:** There is a gap between two consecutive sectors.
2. **Double interleaving:** There are two gaps between two consecutive sectors.

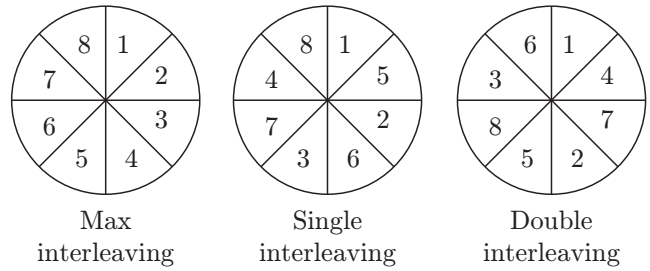


Figure 7.35 | Disk interleaving.

Note:

1. In single interleaving technique, to read a track, two rotations are required.
2. In double interleaving technique, to read a track, 2.75 rotations are required.

Problem 7.12: Consider a disk which has average seek time of 32 ns and rotational rate of 360 rpm (round per minute). Each track of the disk has 512 sectors, each of size 512 bytes.

(a) What is the time taken to read four continuous sectors?
(b) What is the data transfer rate?

Solution:

(a) Time to read four continuous sectors = (Seek time + rotational latency + transfer time)

I. Seek time = 32 ns = 32×10^{-9} s

II. Rotational latency
$\Rightarrow$360 rotations took 60 s
1 rotation $\rightarrow$ (60/360) s
1/2 rotation $\rightarrow$ (60/360) $\times$ (1/2) $\Rightarrow$ 0.083 s

III. Transfer time
$\Rightarrow$1 Track can be read in one rotation, and 1 track has 512 sectors each having 512 bytes.

So, (512 $\times$ 512) bytes data can be read in one rotation time (1/6) s.

256 KB data read time = (1/6) s

$$1 \text{ KB data read time} = \left(\frac{1}{6 \times 256}\right)\text{s}$$

$$2 \text{ KB data read time} = 2 \times \left(\frac{1}{6 \times 256}\right)\text{s}$$

(1 sector size= 512 byte, so four sector = 2KB) = 0.0013 s

Time to read four continuous sectors = (32×10^{-9}+ 0.083 + 0.0013) s = 0.0843 s

(b) Transfer rate = Number of bytes transferred in one unit time (1 s)

1/6 s $\rightarrow$ 256 KB

1 s $\rightarrow$ (256 $\times$ 6) KB = 1536 Kbps

Problem 7.13: Consider an interleaving disk where the track is divided into eight sectors, each of size 2K. Seek time is 30 ms and rotation rate of disk is 3600 rpm. If half rotation is required to place the read/write head at the start of the sector, then

(a) How much time is required reading all eight sectors of the track using double-interleaving disk?
(b) What is the time difference to read all eight sectors if it is a non-interleaving disk?
(c) What is the data transfer rate using double-interleaving disk?

Solution:

(a) Time to read eight continuous sectors = (Seek time + rotational latency + transfer time)

I. Seek time = 30 ms
II. Rotational latency = (1/2) rotation
3600 rotations took 60 s

1 rotation → (60/3600) = 16.66 ms

1/2 rotation → (16.66) × (1/2) = 0.0083 = 8.33 ms

III. Transfer time
We have to read all of the eight sectors mean one whole track. 1 Track can be read in 2.75 rotations (double interleaving).

So, transfer time of eight sectors = Transfer time of 1 track = 2.75 Rotation time

Time to read eight continuous sectors = (Seek time + rotational latency + transfer time)
= (Seek time + (1/2) rotation time + 2.75 rotation time)
= (Seek time + 3.25 rotation time)
=–(30 + 3.25 × (16.66)} = 84.14 ms

(b) Time to read eight continuous sectors = (Seek time + rotational latency + transfer time)

I. Seek time = 30 ms
II. Rotational latency ⇒ 8.33 ms
III. Transfer time
We have to read all of the eight sectors means one whole track.

1 Track can be read in 1 rotation (non-interleaving).

So, transfer time of eight sectors = Transfer time of 1 track = 1 Rotation time

Time to read eight continuous sectors = (Seek time + rotational latency + transfer time)
= (Seek time + (1/2) rotation time + 1 rotation time)
= (Seek time + 1.5 rotation time)
= {(30 + 1.5 × (16.66)} = 55 ms

Time difference between double interleaving and non-interleaving = (84.14 − 55)
= 29.14 ms

(c) Transfer rate = Number of bytes transferred in one unit time (1 s)

2.75 Rotation time → 16 KB

60 s = 3600 rotations

1 s = 3600/60 = 60 rotations

2.75 rotation = 16 KB data

1 rotation = (16/2.75) KB

60 rotations = {(16/2.75) × 60} KB ≈ 349 KB

So, transfer rate ≈ 349 Kbps

7.10.2 Disk Scheduling Algorithms

One of the responsibilities of the operating system is to use the hardware efficiently, which means that we should have fast disk access time and a broader disk bandwidth. Whenever a process needs I/O to or from the disk, it issues a system call to the operating system. The request specifies several pieces of information, such as:

1. Whether this operation is input or output?
2. What the disk address for the transfer is?
3. What the memory address for the transfer is?
4. What the number of bytes to be transferred is?

If the desired disk drive and controller are available, the request can be serviced immediately. If the driver or controller is busy, any new requests for service will be placed on the queue of pending requests for that device. We require disk-scheduling algorithms to service these pending requests.

Problem 7.14: Consider, for example, a disk queue with requests for I/O to blocks on cylinders

88, 83, 7, 13, 90, 79, 11, 63

The disk head is initially at cylinder 53.

Solution:

The head will first move from 53 to 88 (being the first request), then to 83, 7, 13, 90, 79, 11, and finally to 63, for a total head movement of 339 cylinders. This schedule is shown in Fig. 7.36.

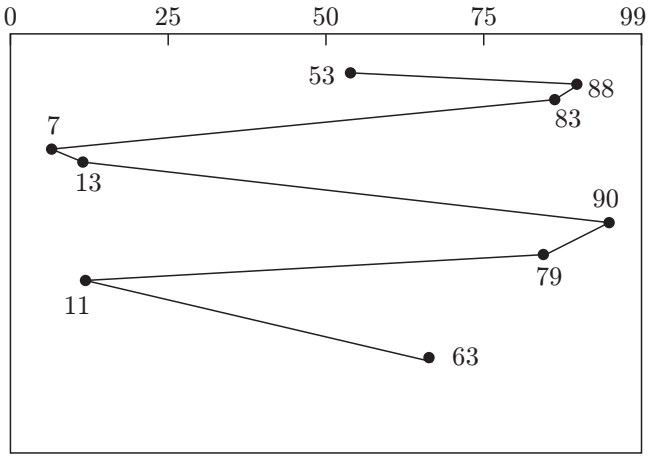


Figure 7.36 FCFS disk-scheduling algorithm.

Total head movement = (53∼88) + (88∼83) + (83∼7) + (13∼7) + (99∼13) + (79∼90) + (11∼79) + (11∼63)
= 35 + 5 + 76 + 6 + 86 + 11 + 68 + 52
= 339 cylinders

7.10.2.1 FCFS Scheduling

The simplest form of the disk scheduling is, of course, the first-come, first-served (FCFS) algorithm. This algorithm is intrinsically fair, but it generally does not provide the fastest service.

7.10.2.2 SSTF Scheduling

A variation of shortest job first (SJF), applied in CPU scheduling. It serves the request that has a minimum seek time from the current head position. To calculate the minimum seek time, we look for the minimum number of cylinders to be traversed. The major limitation is — starvation, as encountered in SJF algorithm.

Problem 7.15: Consider, the same problem, a disk queue with requests for I/O to blocks on cylinders

88, 83, 7, 13, 90, 79, 11, 63

The disk head is initially at cylinder 53.

Solution:

The head is positioned at cylinder 53, the minimum distance it has to move is 10 when it moves to cylinder 63, which is minimum, then from 63 the next access will be 79, then 83, to 88, then to 90, to 13, 11 and finally 7. This schedule is shown in Fig. 7.37.

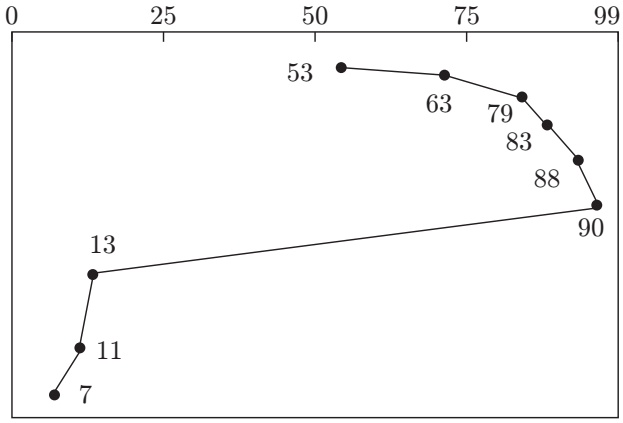


Figure 7.37 SSTF disk-scheduling algorithm.

Total head movement = (53∼63) + (63∼79) + (79∼83) + (83∼88) + (88∼90) + (90∼13) + (13∼11) + (11∼7)

= 10 + 4 + 4 + 5 + 2 + 77 + 2 + 4 = 108 cylinders

7.10.2.3 SCAN Scheduling

Also, called the elevator algorithm, here the disk arm begins serving the requests at one end and moves to the other end (it does not come back, as in earlier algorithms) and then returns back serving the requests, similar to an elevator. This technique eventually results in reduced variance as compared to SSTF.

Problem 7.16: Consider, the same previous problem, a disk queue with requests for I/O to block on cylinders

88, 83, 7, 13, 90, 79, 11, 63

The disk head is initially at cylinder 53 and the disk arm is moving towards 0.

Solution:

The head is positioned at cylinder 53 and the disk arm is moving towards 0. So, the cylinders accessed will be 12, 11 and 7, the disk arm will touch 0 and then will read 63, then 79, 83, 88 and finally 90. This schedule is shown in Fig. 7.38.

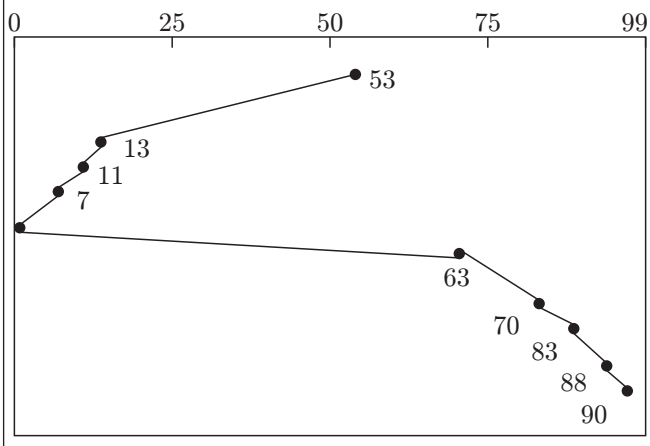


Figure 7.38 | SCANdisk-scheduling algorithm.

Total head movement = (53∼13) + (13∼11) + (11∼7) + (7∼0) + (0∼63) + (63∼79) + (79∼83) + (83∼88) + (88∼90)

= 40 + 2 + 4 + 7 + 63 + 16 + 4 + 5 + 2
= 143 cylinders

It should take into account the head movement from cylinder 7 to 0 and 0 to 63 also.

7.10.2.4 C-SCAN Scheduling

This algorithm treats the cylinders as a circular list, hence provides a uniform wait time as compared to the SCAN algorithm discussed above. The major difference is that the requests are served in any one direction of the motion of the head, not in both the directions as in SCAN. This can be compared by looking at the following numerical problem.

Problem 7.17: Consider, the same problem, a disk queue with requests for I/O to blocks on cylinders

88, 83, 7, 13, 90, 79, 11, 63

The disk head is initially at cylinder 53 and the disk arm is moving towards 0.

Solution:

The head is positioned at cylinder 53 and the disk arm is moving towards 0. So, the cylinders accessed will be 13,11 and 7, the disk arm will touch 0 and then will return to the end (in this case) and start reading 90, 88, 83, 79, and finally read 63. This schedule is shown in Fig. 7.39.

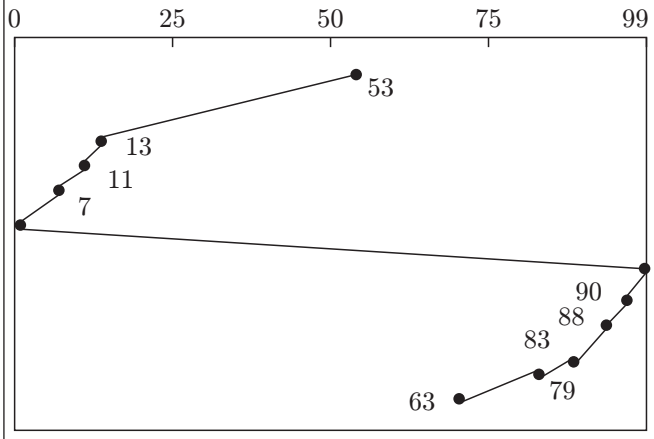


Figure 7.39 | C-SCAN disk-scheduling algorithm.

Total head movement = (53∼13) + (13∼11) + (11∼7) + (7∼0) + (0∼99) + (99∼90) + (90∼88) + (88∼83) + (83∼79) + (79∼63)

= 40 + 2 + 4 + 7 + 99 + 9 + 2 + 5 + 4 + 16
= 188 cylinders

7.10.2.5 LOOK and C-LOOK Scheduling

LOOK and C-LOOK are the variants of SCAN and C-SCAN algorithm, respectively. Here the head does not move till the end of the cylinder, it only moves to the farthest cylinder which requires to be served. After serving the farthest request, it changes the direction (in case of LOOK) or jumps to another end (in case of C-LOOK). The same is illustrated below.

Problem 7.18: Consider, the same problem, a disk queue with requests for I/O to blocks on cylinders

88, 83, 7, 13, 90, 79, 11, 63

The disk head is initially at cylinder 53 and the disk arm is moving towards 0.

Solution:

The head is positioned at cylinder 53 and the disk arm is moving towards 0. So, the cylinders accessed will be 13, 11 and 7, now the disk arm will not touch 0 (as in SCAN algorithm) but will read 63, then 79, 83, 88 and finally 90. This schedule is shown in Fig. 7.40.

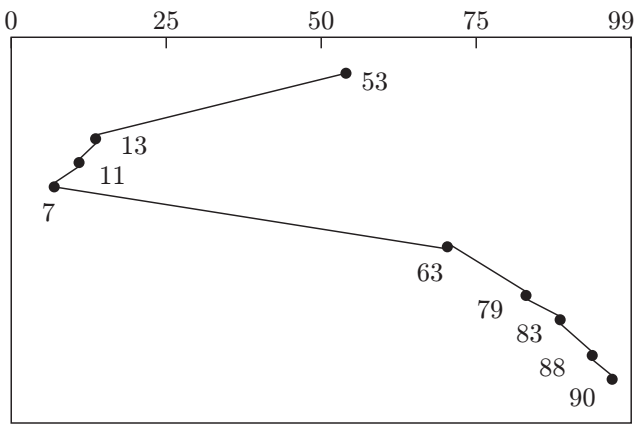


Figure 7.40 LOOK disk-scheduling algorithm.

Total head movement = (53∼13) + (13∼11) + (11∼7) + (7∼63) + (63∼79) + (79∼83) + (83∼88) + (88∼90)

= 40 + 2 + 4 + 56 + 16 + 4 + 5 + 2 = 129 cylinder

Problem 7.19: Consider, the same example, a disk queue with requests for I/O to blocks on cylinders

88, 83, 7, 13, 90, 79, 11, 63

The disk head is initially at cylinder 53 and the disk arm is moving towards 0. Simulate the C-LOOK disk-scheduling algorithm.

Solution:

The head is positioned at cylinder 53 and the disk arm is moving towards 0. So, the cylinders accessed will be 13,11 and 7, the disk arm will not touch 0 but will return to 90, then read 88, 83, 79 and finally read 63. This schedule is shown in Fig. 7.41.

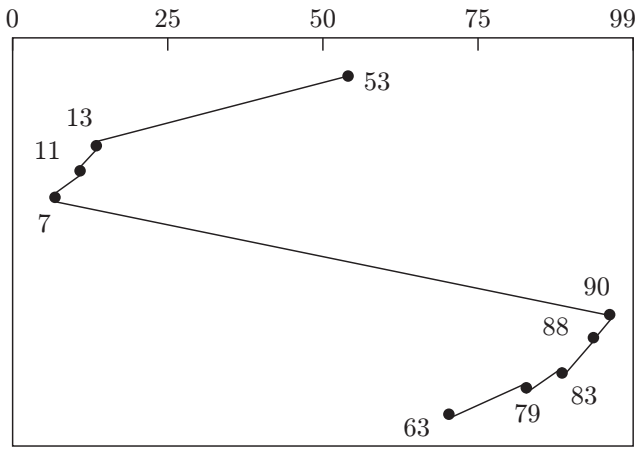


Figure 7.41 C-LOOK disk-scheduling algorithm.

Total head movement = (53∼13) + (13∼11) + (11∼7) + (7∼90) + (90∼88) + (88∼83) + (83∼79) + (79∼63)

= 40 + 2 + 4 + 83 + 2 + 5 + 4 + 16 = 156 cylinder

7.10.2.6 Selection of a Disk-Scheduling Algorithm

Given so many disk-scheduling algorithms, how do we choose the best one? SSTF is common and has a natural appeal because it increases performance over FCFS. SCAN and C-SCAN perform better for system that place a heavy load on the disk, because they are less likely to have starvation problem. Therefore, it is dependent on the type of application we want to use.

7.11 PROTECTION AND SECURITY

As huge information is stored in computer systems, the need to protect it is equally important. Usually, security and protection are used interchangeably.

7.11.1 Security

There are three important aspects, known as CIA triad, which are basic to providing security. These are:

1. **Confidentiality:** It involves data confidentiality as well as user privacy.
2. **Integrity:** It involves two terms—data integrity (which means that data is handled in an authorized manner) and system integrity (which means

that the system is free from unauthorised access or manipulation).

3. **Availability:** It means the timely access to user (for which he has the rights).

7.11.2 Security Environment

Security has many facets. Three of the most important ones are the nature of the threats, the nature of intruders and accidental data loss.

There are two different kinds of attack. A *'Passive Attack'* does not affect the system resources but observes the various acts of the system (such as – trying to know the password by looking over the shoulders of a user), whereas in an *'Active Attack'* the system resources are affected (such as – an unauthorised person deleting a file).

7.11.2.1 Threats

From a security perspective, computer systems have three general goals, with corresponding threats to them (see Table 7.5):

Table 7.5 | Goal and threats.

Goal	Threat
Data confidentiality	Exposure of data
Data integrity	Tempering with data
System availability	Denial of Service (DOS)

7.11.2.2 Intruders

In the securityperspective, intruders are defined as unauthorized people, who are entering into the system they do not have business with. Intruders are also known as '*adversaries'*, and are capable of deploying active attacks as well as passive attacks.

7.11.2.3 Accidental Data Loss

Malicious intruders cause threats to the system. Hence, valuable data can be lost by accident. The common causes of accidental data loss are as follows:

1. **Hardware or software errors:** CPU malfunctions, unreadable disks or tapes, program bugs, etc.
2. **Human errors:** Incorrect data entry, wrong tape or disk mounted, wrong program run, lost disk or tape, etc.
3. **Natural disaster:** Fires, floods, earthquakes, wars, riots, or rats gnawing tapes or floppy disks, etc.

7.11.3 Cryptography

First we will discuss about cryptography, in brief, to understand more about security.

The aim of cryptography is to encrypt plaintext (message) into cipher text in such a way that only the authorized person or user can decrypt it back to plaintext.

In symmetric cryptography (or private key cryptography), one makes use of a single key. This key is used for encryption at the sender side and the same key is used at the receiver side to decrypt the message.

Whereas, in asymmetric cryptography (or public key cryptography), two keys are used in which one is private and the other one is public. The public key is used to encrypt the message, whereas the private key, which is only known to the receiver, is used for decryption.

The encryption process is illustrated in the Fig. 7.42 below:

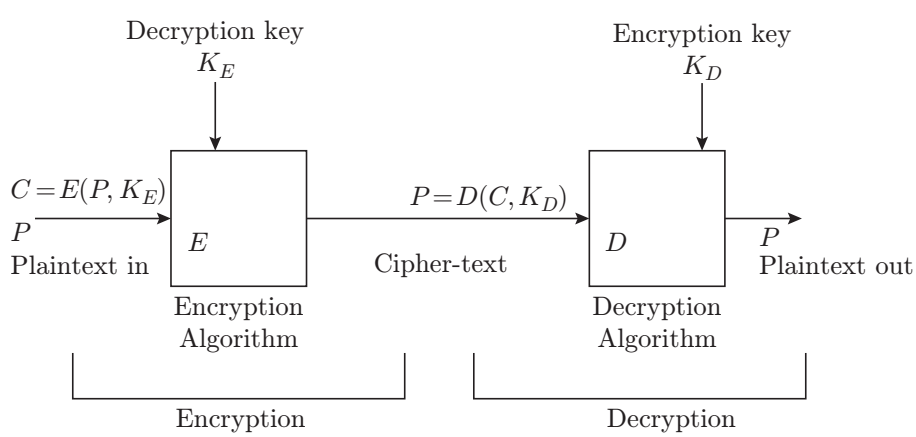


Figure 7.42 | Mechanism of cryptography.

7.11.4 Attacks from within the System

When an adversary enters into a system, he or she can start doing damage. The damage is of the following types:

1. **Trojan horses:** Trojan horse is a program, which contains a code hidden within a block of code, used to modify/delete/encrypt the user's data (functioning is similar to the Trojan horse, we read in history). The Trojan horse cannot be differentiated from a normal code of program by reading the source code of the program housing the Trojan horse.
2. **Login spoofing:** Related to Trojan horse, it is a technique to obtain a user's password. The user is provided with the same lookalike page, as his login and he is unable to differentiate between the actual page and this page; so he enters his login password and the vital information does not remain private.
3. **Logic bombs:** Similar to a bomb, this code explodes (causes destruction) when certain condition(s) are met like – a particular day of time or a particular date or when the user is running a particular file. Such a code is embedded in a legitimate program, as in a Trojan horse. These codes can modify or delete a file or can even cause a system to halt or do any other damage.
4. **Trap doors:** These are also small programs embedded in the programs to quickly gain access at a later time. These are also known as '*backdoors*'. It is very difficult to block these backdoors by an OS.
5. **Buffer overflow:** It is a program written to block memory or buffer. It allows adversary to modify portions of the target process address space.

7.11.5 Attacks from Outside the System

In the previous sections, we have discussed about the threats largely caused by users already (logged) in the system; however, for machines connected to the network, there are external threats too.

7.11.5.1 Virus Damage Scenarios

A virus is a program that can do anything a program can do. It replicates itself, infects user's data or files, stops the operation of the system or functioning of the system. It gets its name because of the similarities in its functionality to a biological virus. The common acronym for virus is Vital Information and Resources Under Siege. Nowadays, there are certain programs known as *'Kit'*, capable of generating new viruses automatically. Some of the well-known anti-virus softwares are: Symantec, McAfee, AVG, Kaspersky, etc.

Viruses can be divided into different classes based upon the following characteristics:

1. Environment
2. Operating system (OS)
3. Destructive capabilities

(Not to forget: there also exist other 'harmful' programs or so called '**malware**', such as Trojan horses).

1. **Environment:** Viruses can be described as:
   - **File viruses:** Most common virus that infects the executable (*.exe) files.
   - **Boot viruses:** The virus infects the boot sector program, the contents are replaced by infected code. Nowadays, modern OS are capable of thwarting the attack by such kind of virus.
   - **Macro viruses:** Macros are defined as a group of lines, technically speaking a subroutine, which is capable of infecting documents like – MSOffice, that make use of OLE (Object Linking and Embedding) formats.
   - **Network viruses:** These viruses make use of protocols and commands of computer network or email to spread themselves in the system.

   There is a large number of combinations possible, for example, file-boot viruses infecting both files and boot sectors on the disks. Another example of the combo – network macrovirus, not only infect the documents which are being edited, but also send copies of itself by email.

2. **Mode of Operation:** Viruses can be described as follows:
   - **TSR:** A terminate-and-stay-resident (TSR) virus infects a computer and remains in RAM and then performs an action for which it was planted. The virus becomes a part of the OS and remains active always. In contrast a non-resident virus is active for a limited time only.
   - **Stealth algorithm:** These viruses do not reveal their identity.
   - **Self-encrypting and polymorphic:** Such viruses are not detected by antivirus as they do not have a fixed signature. They are able to change themselves with every infection, and therefore, are very hard to detect.

3. **Destructive capabilities:** Viruses can be described as follows:
   - **Harmless:** Do not affect the computing resources.
   - **Not dangerous:** Have a limited effect on computing resources. For example, a certain sound can be produced after the computer is ON for say 2 hours.

- **Dangerous:** They disrupt the normal working of computing resources, such as you might not be able to read from the USB port of the computer.
- **Very dangerous:** Certain viruses result in loss of data, such as erasing system files which required when a system is to be booted, so as a result, the system does not work (boot).

7.11.6 Anti Virus Approaches

The best approach is the protection against the virus. If at all the system has been infected by virus, the following steps are commonly used:

1. **Detect:** Locate it (by checking the file size of vital files).
2. **Identify:** Identify the virus (may be your antivirus scanner might tell you).
3. **Remove:** All the traces of the virus are removed (disinfected by the antivirus scanner).

The antivirus scanners use different approaches to protect the system, some of these approaches are:

1. **Signature scanning:** Each virus has a certain signature (a code), which is located and deleted.
2. **Heuristic scanning:** Makes use of heuristics (rules of thumb or a guess) based upon certain analysis.
3. **Generic decryption:** The virus decrypts itself after the execution of code.
4. **Behaviour blocking:** Its function is similar to a firewall, it blocks the execution of a suspicious piece of code.

IMPORTANT FORMULAS

1. Turnaround Time: TAT = CT − AT [(CT: completion time, AT: arrival time)]
2. Waiting Time: WT = TAT − BT [(BT: burst time)]
3. Waiting Time for Round Robin: $(n-1)q + nc$
4. Complexity of Safe Sequence using Banker's Algorithm: $m \times n^2$(n: processes, m: resources)
5. To Avoid Deadlock:
   - Total need $< m + n$
   - Max need of each process should be between 1 and m resources
6. Response Ratio $= \dfrac{w+s}{s}$(w = waiting time, s = service time or burst time)
7. Thread maintains different registers and stack but shares code, data and files.
8. Need$[I]$ = Max$[I]$ − Allocation$[I]$
9. If program contains n fork() call, then the number of created child process are $(2^n - 1)$.
10. Number of pages $= \dfrac{\text{LAS}}{\text{Page size}}$
11. Number of frames $= \dfrac{\text{PAS}}{\text{Frame size}}$
12. $E_{\text{MAT}} = X(C+M) + (1-X)(C+2M)$
13. Effective memory access time $(\text{E}_{\text{MAT}}) = P \times S + (1-P) \times M$
14. Total size of file system

$$= \left\{ \text{Number of direct DBA} + \left(\frac{\text{DBA size}}{\text{DBA}}\right) + \left(\frac{\text{DBA size}}{\text{DBA}}\right)^2 + \left(\frac{\text{DBA size}}{\text{DBA}}\right)^3 + \cdots \right\} \times \text{DB size}$$

15. Number of disk block possible in one disk block $= \left(\dfrac{\text{DBA size}}{\text{DBA}}\right)$
16. Number of cylinder per track = Number of tracks per surface
17. Capacity in disk = (Plotters × Surface × Tracks × Sector × Sector size) bytes
18. Number of bits required to identify a sector = (Bits for plotter + bits for surface + bits for tracks + bits for sectors)
19. Time to read four continuous sectors = (Seek time + rotational latency + transfer time)

SOLVED EXAMPLES

1. The term thrashing is used to define

(a) A reduce page I/O
(b) A decreased degree of multiprogramming
(c) An excessive page I/O
(d) Improvement(s) in the system performance

Solution: Thrashing means that the system more busy in dealing with page faults instead of doing some constructive work. Thus, it leads to excessive page I/O.

Ans. (c)

2. Page fault occurs when

(a) The page is not in cache memory.
(b) The page is in main memory.
(c) The page is not in main memory.
(d) The page has an address, which cannot be loaded.

Solution: If page is not found in page table, it generates the page fault.

Ans. (c)

3. If $m =$ number of resources, $n =$ number of processes in the system, then Banker's algorithm although a general algorithm may require _____ operations.

(a) $n \times n \times m$
(b) $(m \times m) \times n$
(c) $(m \times n) \times n$
(d) $(n \times m) \times m$

Solution: Complexity of Safe Sequence using Banker's Algorithm is $m \times n^2$
where n is the number of processes, and m is the number of resources.

Ans. (a, c)

4. Mutual exclusion problem occurs

(a) Between two disjoint processes that do not interact.
(b) Among processes that share resources.
(c) Among processes that do not use the same resource.
(d) Between two processes that uses different resources of different machines.

Solution: In this problem, if process P_i is executing in its critical section, then no other process can execute their critical sections.

Ans. (b)

5. Linux operating system uses

(a) i-Node scheduling.
(b) fair pre-emptive scheduling.
(c) RR scheduling.
(d) highest penalty ratio next.

Solution: It uses fair pre-emptive scheduling.

Ans. (b)

6. In modern operating systems such as Windows 2000, all the processor-dependent code is isolated in a dynamic link library called

(a) NTFS file system
(b) Hardware abstraction layer
(c) Microkernel
(d) CORBA Process Manager

Solution: The hardware abstraction layer (HAL) provides logical link between NT-based operating systems and physical hardware of the computer.

Ans. (b)

7. Translation look-aside buffer (TLB) is

(a) A cache-memory in which item to be searched is compared one by one with the keys.
(b) A cache-memory in which item to be searched is compared with all the keys simultaneously.
(c) A main memory in which item to be searched is compared one by one with the keys.
(d) An associative memory in which item to be searched is compared with all the keys simultaneously.

Solution: TLB is a cache memory used by memory management unit to improve the translation speed of virtual address.

Ans. (d)

8. Windows 2000 operating system include the following process states:

(a) Ready, running and waiting
(b) Ready, standby, running, waiting, transition and terminated
(c) Ready, running, waiting, transition, sleep and terminated
(d) Standby, running, transition, sleep and terminated

Solution: It includes ready, standby, running, waiting, transition and terminated states.

Ans. (b)

9. A memory management algorithm, realizing virtual memory, partially swaps out a process. This is similar to which kind of CPU scheduling.

(a) Short-term scheduling
(b) Long-term scheduling

(c) Medium-term scheduling
(d) Mutual exclusion

Solution: Medium-term scheduling is one in which decision is taken to add the number of processes that are present either partially or fully in the main memory. This is similar to CPU scheduling. In short-term scheduling the available process is executed by the processor. In long-term scheduling, the process is added to the pool of processes to be executed.

Ans. (c)

10. The working set model is used in memory management to implement the concept of

(a) Segmentation (b) Principle of locality
(c) Paging (d) Thrashing

Solution: Fact

Ans. (b)

11. Which scheduling algorithm gives a minimum average waiting time?

(a) RR (b) SJF
(c) FCFS (d) Priority

Solution: Shortest job first algorithm.

Ans. (b)

12. If the disk is rotating at 3600 rpm, determine the effective data transfer rate which is defined as the number of bytes transferred per second between disk and memory. (Given size of track = 512 bytes)

(a) 20Kbps (b) 30 Kbps
(c) 40Kbps (d) 50Kbps

Solution: 3600 rotation → 60 s

1 rotation → 60/3600 = 1/60 s

Transfer rate = (60 × track size) per second
= 60 × 512 bytes/s
= 30 Kbps

Ans. (b)

13. Which of the following features will characterize an OS as multiprogrammed OS?

1. More than one program may be loaded into main memory at the same time.
2. If a program waits for certain event, another program is immediately scheduled.
3. If the execution of a program terminates, another program is immediately scheduled.

(a) Only
(b) (1) and (2) only
(c) (1) and (3) only
(d) (1), (2) and (3) only

Solution: Multiprogrammed OS performs all the given three tasks.

Ans. (d)

14. The P and V operations on counting semaphores, where s is a counting semaphore are defined as follows:

P(s):$s = s - 1$
if $s < 0$ then wait;
$V(s)$:$s = s + 1$
if $s = 0$ then wake up a process waiting on s.
Assume that Pb and Vb are the wait and signal operation on binary semaphore. Two binary semaphores Xb and Yb are used to implement the semaphore operation $P(s)$ and $V(s)$ as follows:

```
P(s): Pb(Xb);
s=s-1
if(s<0){
Vb(Xb);
Pb(Yb);
}
else Vb(Xb);
V(s): Pb(Yb);
s= s +1
if(s<=0)
Vb(Yb);
```

The initial value of Xb and Yb are, respectively,

(a) 0 and 0 (b) 0 and 1
(c) 1 and 0 (d) 1 and 1

Solution: From the give code, we get $P(s)$ and $V(s)$ decrements and increment the value of semaphore, respectively. So, from the given conditions, we conclude that to avoid mutual exclusion the value of Xb should be 1 and that of Yb should be 0.

Ans. (c)

15. Consider the following table of arrival time and burst time for three processes P_0, P_1 and P_2.

Process	Arrival Time	Burst Time
P_0	0	9
P_1	1	4
P_2	2	9

The pre-emptive shortest job first scheduling algorithm is used. Scheduling is carried out only at arrival or completion of process. The average waiting time for the three processes is ___.

Solution: Waiting time for $P_0 = 5 - 1 = 4$ s
Waiting time for $P_1 = 1 - 1 = 0$ s
Waiting time for $P_2 = 13 - 2 = 4$ s
Average waiting time = 4 + 0 + 11/3 = 5 s

Ans. (5)

16. A CPU-scheduling algorithm determines an order for the execution of its scheduled processes. Given k processes to be scheduled on one processor, how many different schedules are possible? Give a formula in terms of k.

(a) $\sqrt{k}$

(b) $\frac{1}{k}$

(c) k^2

(d) $k!$

Solution: $k!(k \text{ factorial} = k \times (k-1) \times (k-2) \times \cdots \times 2 \times 1)$

Ans. (d)

GATE PREVIOUS YEARS' QUESTIONS

1. Using a larger block size in a fixed block size file system leads to

(a) better disk throughput but poorer disk space utilization.
(b) better disk throughput and better disk space utilization.
(c) poorer disk throughput but better disk space utilization.
(d) poorer disk throughput and poorer disk space utilization.

(GATE 2003: 1 Mark)

Solution: Larger block size in a fixed block size file system reduces the number of disk access, but at the same time increases chances of internal fragmentation.

Ans. (a)

2. In a system with 32-bit virtual addresses and 1-KB page size, use of one-level page tables for virtual to physical address translation is not practical because of

(a) the large amount of internal fragmentation.
(b) the large amount of external fragmentation.
(c) the large memory overhead in maintaining page tables.
(d) the large computation overhead in the translation process.

(GATE 2003: 1 Mark)

Solution: Page size = 2^{10} bytes
Total number of pages required = $2^{32}/2^{10} = 2^{22}$, which is very large.
There will be overhead in maintaining page tables.

Ans. (c)

3. A uni-processor computer system only has two processes, both of which alternate 10 ms CPU bursts with 90 ms I/O bursts. Both the processes were created at nearly the same time. The I/O of both processes can proceed in parallel. Which of the following scheduling strategies will result in the least CPU utilization (over a long period of time) for this system?

(a) First come first served scheduling
(b) Shortest remaining time first scheduling
(c) Static priority scheduling with different priorities for the two processes
(d) Round robin scheduling with a time quantum of 5 ms

(GATE 2003: 2 Marks)

Solution: Round robin will result in least cpu utilization for time slice = 5 ms
Consider process A and B, both process have 10 ms cpu utilization and then 90 ms I/O. Sequence will be ABAB. After 15 ms process, A can do I/O and I/O can be done in parallel; therefore, A finishes I/O at 105th ms and B finishes I/O at 110th ms so cpu remains idle from 20th to 105th ms leading to a total of 85 ms.

Ans. (d)

4. Which of the following will always leads to an output starting with '001100110011'?

(a) P(S) at W, V(S) at X, P(T) at Y, V(T) at Z, S and T initially 1
(b) P(S) at W, V(T) at X, P(T) at Y, V(S) at Z, S initially 1, and T initially 0
(c) P(S) at W, V(T) at X, P(T) at Y, V(S) at Z, S and T initially 1
(d) P(S) at W, V(S) at X, P(T) at Y, V(T) at Z, S initially 1, and T initially 0

(GATE 2003: 2 Marks)

Solution: To print '001100110011'. Process P must start first and process Q must wait until P prints 00 then process P must wait until process Q prints 11 and so on.

In all of the four options P(T) at Y is given, so process Q have to waits for T and T must be

initialize to 0. Process P must signal T (V(T)) at X. similarly in all four option P(S) is given at W so S must be initially 1 and Process Q signal S at Z to print desired output.

Ans. (b)

Common Data Questions 5 and 6: A processor uses two-level page tables for virtual to physical address translation. Page tables for both levels are stored in the main memory. Virtual and physical addresses are both 32 bits wide. The memory is byte addressable. For virtual to physical address translation, the 10 most significant bits of the virtual address are used as index into the first-level page table while the next 10 bits are used as index into the second-level page table. The 12 least significant bits of the virtual address are used as offset within the page. Assume that the page table entries in both levels of page tables are 4 bytes wide.

Further, the processor has a translation look-aside buffer (TLB), with a hit rate of 96%. The TLB caches recently used virtual page numbers and the corresponding physical page numbers. The processor also has a physically addressed cache with a hit rate of 90%. Main memory access time is 10 ns, cache access time is 1 ns, and TLB access time is also 1 ns.

5. Assuming that no page faults occur, the average time taken to access a virtual address is approximately (to the nearest 0.5 ns)

(a) 1.5 ns (b) 2 ns
(c) 3 ns (d) 4 ns

(GATE 2003: 2 Marks)

Solution: Average time taken to access a virtual address
$= [0.96(1 + 0.9 \times 1 + 0.1 \times 1 + 10)] + [(0.04)(21 + 0.9 \times 1 + 0.1 \times 1 + 10)]$
$= 3.87 \approx 4$ ns

Ans. (d)

6. Suppose a process has only the following pages in its virtual address space: two contiguous code pages starting at virtual address 0×00000000, two contiguous data pages starting at virtual address 0×00400000, and a stack page starting at virtual address $0 \times$ FFFFF000. The amount of memory required for storing the page tables of this process is

(a) 8 KB (b) 12 KB
(c) 16 KB (d) 20 KB

(GATE 2003: 2 Marks)

Solution: Total memory required $= 4 \times 2^{12}$ bytes $= 16$ KB

Ans. (c)

7. Which of the following will always lead to an output staring with '001100110011'?

(a) P(S) at W, V(S) at X, P(T) at Y, V(T) at Z, S and T initially 1
(b) P(S) at W, V(T) at X, P(T) at Y, V(S) at Z, S initially 1, and T initially 0
(c) P(S) at W, V(T) at X, P(T) at Y, V(S) at Z, S and T initially 1
(d) P(S) at W, V(S) at X, P(T) at Y, V(T) at Z, S initially 1, and T initially 0

Solution: To get output 001100110011, P and Q should be executed alternatingly. P(S) at W, V(T) at X, P(T) at Y, V(S) at Z, S initially 1, and T initially 0 will give the correct output.

Ans. (b)

8. Consider the following statements with respect to user-level threads and kernel-supported threads

(i) Context switch is faster with kernel-supported threads.
(ii) For user-level threads, a system call can block the entire process.
(iii) Kernel-supported threads can be scheduled independently.
(iv) User-level threads are transparent to the kernel.

Which of the above statements are true?

(a) (ii), (iii) and (iv) only
(b) (ii) and (iii) only
(c) (i) and (iii) only
(d) (i) and (ii) only

(GATE 2004: 1 Mark)

Solution: For option (i): User level thread may load only those registers which are required, whereas kernel-level thread loads each registers, so context switching is slower in kernel-level thread.

For option (ii): Kernel do not have information about all the user-level thread, so for blocking a user-level thread a system call is used.

Options (iii) and (iv) are correlated: Kernel-level threads are well known to kernel, so kernel supported thread can be scheduled independently and user level thread are transparent to kernel.

Ans. (a)

9. Consider an operating system capable of loading and executing a single sequential user process at a time. The disk head scheduling algorithm used is first come first served (FCFS). If FCFS is replaced by shortest seek time first (SSTF), claimed by the vendor to give 50% better benchmark results, what is the expected improvement in the I/O performance of user programs?

(a) 50% (b) 40%
(c) 25% (d) 0%

(GATE 2004: 1 Mark)

Solution: I/O performance is determined by input and output devices. Disk scheduling will not improve the performance of I/O for users.

Ans. (d)

10. The minimum number of page frames that must be allocated to a running process in a virtual memory environment is determined by

(a) the instruction set architecture.
(b) page size.
(c) physical memory size.
(d) number of processes in memory.

(GATE 2004: 1 Mark)

Solution: The minimum number of page frames required by a process is based on instruction set architecture.

Ans.(a)

11. Consider the following set of processes, with the arrival times and the CPU-burst times given in milliseconds.

Process	Arrival Time	Burst Time
P_1	0	5
P_2	1	3
P_3	2	3
P_4	4	1

What is the average turnaround time for these processes with the pre-emptive shortest remaining processing time first (SRPT) algorithm?

(a) 5.50 (b) 5.75
(c) 6.00 (d) 6.25

(GATE 2004: 2 Marks)

Solution:

P_1	P_2	P_4	P_3	P_1

0 1 4 5 8 12

Turnaround time for

$$P_1 = 12 - 0 = 12$$
$$P_2 = 4 - 1 = 3$$
$$P_3 = 8 - 2 = 6$$
$$P_4 = 5 - 4 = 1$$

Average turnaround time $= (12 + 3 + 6 + 1)/4 = 5.50$

Ans. (a)

12. Consider a system with a two-level paging scheme in which a regular memory access takes 150 ns, and servicing a page fault takes 8 ms. An average instruction takes 100 ns of CPU time, and two memory accesses. The TLB hit ratio is 90%, and the page fault rate is one in every 10000 instructions. What is the effective average instruction execution time?

(a) 645 ns (b) 1050 ns
(c) 1215 ns (d) 1230 ns

(GATE 2004: 2 Marks)

Solution: Total memory access time $= 0.90 \times 150 + 0.10 \times 300 = 165$ ns
Time taken to execute instructions $= 100 + 2(165 + 5000 \times 8 \times 10^6) = 1230$ ns

Ans. (d)

13. Consider two processes P_1 and P_2 accessing the shared variables X and Y protected by two binary semaphores S_x and S_y, respectively, both initialized to 1. P and V denote the usual semaphore operators, where P decrements the semaphore value, and V increments the semaphore value. The pseudocodes of P_1 and P_2 are as follows:

P_1:	P_2:
While true do	{ While true do {
L_1;.....	L_3.....
L_2;.....	L_4;.....
X = X + 1;	Y = Y + 1;
Y = Y - 1;	X = Y - 1;
V(S_x);	V(S_y);
V(S_y);	V(S_x);

In order to avoid deadlock, the correct operators at L_1, L_2, L_3 and L_4 are, respectively,

(a) $P(S_y)$, $P(S_x)$; $P(S_x)$, $P(S_y)$
(b) $P(S_x)$, $P(S_y)$; $P(S_y)$, $P(S_x)$
(c) $P(S_x)$, $P(S_x)$; $P(S_y)$, $P(S_y)$
(d) $P(S_y)$, $P(S_y)$; $P(S_x)$, $P(S_y)$

(GATE 2004: 2 Marks)

Solution: To avoid deadlock, values of L_1, L_2, L_3 and L_4 should be considered as $P(S_x)$, $P(S_y)$, $P(S_x)$ and $P(S_y)$.

Ans. (d)

14. A Unix-style i-node has 10 direct pointers and one single, one double and one triple indirect pointers. Disk block size is 1 KB, disk block address is

32 bits, and 48-bit integers are used. What is the maximum possible file size?

(a) 2^{24} bytes (b) 2^{32} bytes
(c) 2^{34} bytes (d) 2^{48} bytes

(GATE 2004: 2 Marks)

Solution: Number of disk block pointers in one block = 2^{10} byte/32 bit = 256

Maximum size of file = 256 × 256 × 256 × 1KB = 2^{34} bytes

Ans. (c)

15. What is the swap space in the disk used for?

(a) Saving temporary html pages
(b) Saving process data
(c) Storing the super-block
(d) Storing device drivers

(GATE 2005: 1 Mark)

Solution: Swap space is used to store the process data.

Ans. (b)

16. Increasing the RAM of a computer typically improves performance because

(a) virtual memory increases.
(b) larger RAMs are faster.
(c) fewer page faults occur.
(d) fewer segmentation faults occur.

(GATE 2005: 1 Mark)

Solution: With the increase in memory, page faults will become less.

Ans. (c)

17. Suppose n processes $P_1, \ldots, P_n$ share m identical resource units, which can be reserved and released one at a time. The maximum resource requirement of process P_i is s_i, where $s > 0$. Which one of the following is a sufficient condition for ensuring that deadlock does not occur?

(a) $i, s_i < m$ (b) $i, s_i < n$

(c) $\sum_{i=1}^{n} s_i < (m+n)$ (d) $\sum_{i=1}^{n} s_i < (m \times n)$

(GATE 2005: 2 Marks)

Solution: To remain in safe state, the maximum resource requirement must be less than $m + n$.

Ans. (c)

18. Consider a disk drive with the following specifications:

16 surfaces, 512 tracks/surface, 512 sectors/track, 1 KB/sector, rotation speed 3000 rpm. The disk is operated in cycle stealing mode whereby whenever one 4 byte word is ready it is sent to memory; similarly, for writing, the disk interface reads a 4 byte word from the memory in each DMA cycle. Memory cycle time is 40 nsec. The maximum percentage of time that the CPU gets blocked during DMA operation is:

(GATE 2005: 2 Marks)

(a) 10 (b) 25 (c) 40 (d) 50

Solution: Data transfer in one rotation = 512 × 1024 Bytes

1 rotation takes = 60/3000 s
512KB is transferred in = 60/3000 s

$$1 \text{ byte will be transferred} = \frac{60}{3000 \times 512 \times 1024}$$

4 bytes will be transferred

$$= \frac{60 \times 4}{3000 \times 512 \times 1024} = 152.58 \text{ s}$$

$$\text{Block } \% = \frac{40}{152.58} = 26\%$$

Ans. (b)

19. Consider the following code fragment:

```
if (fork () ==0)
  {
  a = a + 5; printf("%d,%dn", a, &a);
  }
else
  {
  a = a - 5; printf("%d, %dn", a, &a);
  }
```

Let u *and* v be the values printed by the parent process, and x, y be the values printed by the child process. Which one of the following is TRUE?

(a) $u = x + 10$ and $v = y$
(b) $u = x + 10$ and $v! = y$
(c) $u + 10 = x$ and $v = y$
(d) $u + 10 = x$ and $v! = y$

(GATE 2005: 2 Marks)

Solution: Fork function returns 0 for child process and it gives Process id of child to parent process. Parent process executes else part of code. So $u = a - 5$ for parent and for child $x = a + 5$. Therefore, $u + 10 = x$. As there are two copies of a is created, so they have different memory locations. Hence, $v ! = y$.

Ans. (d)

Linked Answer Questions 20 and 21: We are given 9 tasks T_1, T_2, … T_9. The execution of each task requires one unit of time. We can execute one task at a time. Each task T_i has a profit P_i and a deadline d_i. Profit P_i is earned if the task is completed before the end of the $(d_i)^{th}$ unit of time.

Tank	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
Profit	15	20	30	18	18	10	23	16	25
Deadline	7	2	5	3	4	5	2	7	3

20. Are all tasks completed in the schedule that gives maximum profit?

(a) All tasks are completed
(b) T_1 and T_6 are left out
(c) T_1 and T_8 are left out
(d) T_4 and T_6 are left out

(GATE 2005: 2 Marks)

Solution:

T_2	T_7	T_9	T_5	T_3	T_8	T_1

T_4 and T_6 are left out.

Ans. (d)

21. What is the maximum profit earned?

(a) 147 (b) 165 (c) 167 (d) 175

(GATE 2005: 2 Marks)

Solution: Maximum profit = 20 + 23 + 25 + 18 + 30 + 15 + 16 = 147

Ans. (a)

22. Consider three CPU-intensive processes, which require 10, 20 and 30 time units and arrive at times 0, 2 and 6, respectively. How many context switches are needed if the operating system implements a shortest remaining time first scheduling algorithm? Do not count the context switches at time zero and at the end.

(a) 1 (b) 2
(c) 3 (d) 4

(GATE 2006: 1 Mark)

Solution:

P_1	P_2	P_3

0 10 30 60

Two times context switching will be done.

Ans. (b)

23. The atomic fetch-and-set x, y instruction unconditionally sets the memory location x to 1 and fetches the old value of x n y without allowing any intervening access to the memory location x. Consider the following implementation of P and V functions on a binary semaphore S.

```
void P (binary_semaphore *S)
     {
           unsigned y;
           unsigned *x = &(S->value);
           do
       {
         fetch-and-set x, y;
       } while (y);
     }
     void V (binary_semaphore *S)
     {
           S->value = 0;
     }
```

Which one of the following is true?

(a) The implementation may not work if context switching is disabled in P.
(b) Instead of using fetch-and-set, a pair of normal load/store can be used.
(c) The implementation of V is wrong.
(d) The code does not implement a binary semaphore.

(GATE 2006: 2 Marks)

Solution: If context switching is disabled, this implementation may not work. Then processes will not be synchronized.

Ans. (a)

24. A CPU generates 32-bit virtual addresses. The page size is 4 KB. The processor has a translation look-aside buffer (TLB) which can hold a total of 128 page table entries and is 4-way set associative. The minimum size of the TLB tag is

(a) 11 bits (b) 13 bits
(c) 15 bits (d) 20 bits

(GATE 2006: 2 Marks)

Solution: Virtual address = 32 bits

Page size = 4KB = 12 bits
Number of pages = 128 = 2^7 = 7 bits are required
In 4-way set-associative memory = 128/4 = 32 = 5 bits required
Tag bits = 32 − 12 − 5 = 15

Ans. (c)

25. A computer system supports 32-bit virtual addresses as well as 32-bit physical addresses. Since the virtual address space is of the same size as the physical address space, the operating system

designers decide to get rid of the virtual memory entirely. Which one of the following is true?

(a) Efficient implementation of multi-user support is no longer possible.
(b) The processor cache organization can be made more efficient now.
(c) Hardware support for memory management is no longer needed.
(d) CPU scheduling can be made more efficient now.

(GATE 2006: 2 Marks)

Solution: Special hardware support is required for address translation in case of virtual addressing.

Ans. (c)

26. Consider three processes (process id 0, 1, 2, respectively) with compute time bursts 2, 4 and 8 time units. All processes arrive at time zero. Consider the longest remaining time first (LRTF) scheduling algorithm. In LRTF, ties are broken by giving priority to the process with the lowest process id. The average turnaround time is

(a) 13 units (b) 14 units
(c) 15 units (d) 16 units

(GATE 2006: 2 Marks)

Solution:

Process	Burst Time	Remaining Time	Turnaround Time
P_0	2	2 − 1 = 1	12
P_1	4	4 − 1 = 3 − 1 = 2 − 1 = 1	13
P_3	8	8 − 4 = 4 − 1 = 3 − 1 = 2 − 1=1	14

P_2	P_1	P_2	P_1	P_2	P_0	P_1	P_2	P_0	P_1	P_2

0 4 5 6 7 8 9 10 11 12 13 14

Average turnaround time = 12 + 13 + 14 = 39/3 = 13

Ans. (a)

27. Consider three processes, all arriving at time zero, with total execution time of 10, 20 and 30 units, respectively. Each process spends the first 20% of execution time doing I/O, the next 70% of time doing computation, and the last 10% of time doing I/O again. The operating system uses a shortest remaining compute time first scheduling algorithm and schedules a new process either when the running process gets blocked on I/O or when the running process finishes its compute burst. Assume that all I/O operations can be overlapped as much as possible. For what percentage of time does the CPU remain idle?

(a) 0% (b) 10.6%
(c) 30.0% (d) 89.4%

(GATE 2006: 2 Marks)

Solution: Let three processes be p_0, p_1 and p_2. Their execution time is 10, 20 and 30, respectively. p_0 spends first 2 time units in I/O, 7 units of CPU time and finally 1 unit in I/O. p_1 spends first 4 units in I/O, 14 units of CPU time and finally 2 units in I/O. p_2 spends first 6 units in I/O, 21 units of CPU time and finally 3 units in I/O.

Ideal	P_0	P_1	P_2	ideal

0 2 9 23 44 47

Total time spent = 47
Idle time = 2 + 3 = 5
Percentage of idle time = (5/47) × 100 = 10.6%

Ans. (b)

28. Consider the following snapshot of a system running n processes. Process i is holding x_i instances of a resource R, 1 n. Currently, all instances of R are occupied. Further, for all i, process i has placed a request for an additional y_i instances while holding the x_i instances it already has. There are exactly two processes p and q such that $y_p = y_q = 0$. Which one of the following can serve as a necessary condition to guarantee that the system is not approaching a deadlock?

(a) min $(x_p, x_q) < \max_{k \neq p,q} y_k$
(b) $x_p + x_q \geq \min_{k \neq p,q} y_k$
(c) max $(x_p, x_q) > 1$
(d) min $(x_p, x_q) > 1$

(GATE 2006: 2 Marks)

Solution: Processes p and q have sufficient resources so they do not require any additional resources and both of them can complete and release all resources acquired by them which are $(x_p + x_q)$. If the resources released by process p and q are sufficient for another process which is waiting for y_k resources, then system will not be in deadlock.

Ans. (b)

Linked Answer Questions 29 and 30: Barrier is a synchronization construct where a set of processes synchronizes globally, that is, each process in the set arrives at the barrier and waits for all others to arrive and then all processes leave the barrier. Let the number of processes in the set be three and S be a binary semaphore with the usual P and V functions. Consider the following C implementation of a barrier with line numbers shown on left.

```
void barrier (void) {
1:    P(S);
2:    process_arrived++;
3:    V(S);
4:    while (process_arrived !=3);
5:    P(S);
6:    process_left++;
7:    if (process_left==3) {
8:    process_arrived = 0;
9:    process_left = 0;
10:   }
11:   V(S);
      }
```

The variables process_arrived and process_left are shared among all processes and are initialized to zero. In a concurrent program, all the three processes call the barrier function when they need to synchronize globally.

29. The above implementation of barrier is incorrect. Which one of the following is true?

(a) The barrier implementation is wrong due to the use of binary semaphore S.
(b) The barrier implementation may lead to a deadlock if two barriers in invocations are used in immediate succession.
(c) Lines 6 to 10 need not be inside a critical section.
(d) The barrier implementation is correct if there are only two processes instead of three.

(GATE 2006: 2 Marks)

Solution: Fact

Ans. (b)

30. Which one of the following rectifies the problem in the implementation?

(a) Lines 6 to 10 are simply replaced by process_arrived.
(b) At the beginning of the barrier the first process to enter the barrier waits until process_arrived becomes zero before proceeding to execute $P(S)$.
(c) Context switch is disabled at the beginning of the barrier and re-enabled at the end.
(d) The variable process_left is made private instead of shared.

(GATE 2006: 2 Marks)

Solution: Fact

Ans. (b)

31. Group 1 contains some CPU scheduling algorithms and Group 2 contains some applications. Match entries in Group 1 to entries in Group 2.

Group I	Group II
(P) Gang Scheduling	(1) Guaranteed Scheduling
(Q) Rate Monotonic Scheduling	(2) Real-time Scheduling
(R) Fair Share Scheduling	(3) Thread Scheduling

(a) P – 3; Q – 2; R – 1
(b) P – 1; Q – 2; R – 3
(c) P – 2; Q – 3; R – 1
(d) P – 1; Q – 3; R – 2

(GATE 2007: 1 Mark)

Solution: Gang scheduling: Used in parallel systems like for thread scheduling

Rate monotonic scheduling: Used in real-time systems

Fair share scheduling: Used for guaranteed scheduling

Ans. (a)

32. Consider a disk pack with 16 surfaces, 128 tracks per surface and 256 sectors per track. 512 bytes of data are stored in a bit serial manner in a sector. The capacity of the disk pack and the number of bits required to specify a particular sector in the disk are, respectively,

(a) 256 Mbyte, 19 bits
(b) 256 Mbyte, 28 bits
(c) 512 Mbyte, 20 bits
(d) 64 Gbyte, 28 bits

(GATE 2007: 1 Mark)

Solution: Disk capacity = 16 surfaces × 128 tracks × 256 sectors × 512 bytes = 256 MB

Total number of sectors = $16 \times 128 \times 256 = 2^{19}$

Ans. (a)

33. Consider the following statements about user-level threads and kernel-level threads. Which one of the following statements is FALSE?

(a) Context switch time is longer for kernel-level threads than for user-level threads.
(b) User-level threads do not need any hardware support.
(c) Related kernel-level threads can be scheduled on different processors in a multiprocessor system.
(d) Blocking one kernel-level thread blocks all related threads.

(GATE 2007: 1 Mark)

Solution: Kernel-level threads are managed by kernel. Blocking one thread may not block other threads.

Ans. (d)

34. An operating system uses shortest remaining time first (SRT) process scheduling algorithm. Consider the arrival times and execution times for the following processes:

Process	Execution Time	Arrival Time
P_1	20	0
P_2	25	15
P_3	10	30
P_4	15	45

What is the total waiting time for process P_2?

(a) 5 (b) 15
(c) 40 (d) 55

(GATE 2007: 2 Marks)

Solution:

P_1	P_2	P_3	P_2	P_4

0 20 30 40 55 70

Waiting time of P_2 = Completion time − (Arrival time + Execution time)
= 55 − (15 + 25)
= 15

Ans. (b)

35. A virtual memory system uses first in first out (FIFO) page replacement policy and allocates a fixed number of frames to a process. Consider the following statements:

P: Increasing the number of page frames allocated to a process sometimes increases the page fault rate.
Q: Some programs do not exhibit locality of reference.
Which one of the following is TRUE?

(a) Both P and Q are true, and Q is the reason for P.
(b) Both P and Q are true, but Q is not the reason for P.
(c) P is false, but Q is true.
(d) Both P and Q are false.

(GATE 2007: 2 Marks)

Solution: P: True, it is Belady's anomaly.
Q: True, but not reason for Belady's anomaly.

Ans. (b)

36. A single processor system has three resource types X, Y and Z, which are shared by three processes. There are 5 units of each resource type. Consider the following scenario, where the column **alloc** denotes the number of units of each resource type allocated to each process, and the column **request** denotes the number of units of each resource type requested by a process in order to complete execution. Which of these processes will finish **LAST**?

	alloc			request		
	X	Y	Z	X	Y	Z
P_0	1	2	1	1	0	3
P_1	2	0	1	0	1	2
P_2	2	2	1	1	2	0

(a) P_0
(b) P_1
(c) P_2
(d) None of the above, since the system is in a deadlock

(GATE 2007: 2 Marks)

Solution: Available resources: 0 1 2
P_1's request can be fulfilled, it will start executing.
Available after P_1 execution: 2 1 3
P_0's request can be fulfilled and then P_2 will execute.
P_2 will finish last.

Ans. (c)

37. Two processes, P_1 and P_2, need to access a critical section of code. Consider the following synchronization construct used by the processes:
In process P_1 and P_2, variables wants1 and wants2 both become true, but process P_1 and P_2 are stuck for infinite time in their while loops waiting for each other to finish. So, they do not prevent deadlock. But mutual exclusion is ensured.

P_1	P_2
`while (true) {`	`while (true) {`
`wants1 = true;`	`wants2 = true;`
`while (wants2==true);`	`while (wants1==true);`
`/* Critical Section */`	`/* Critical Section */`
`Wants1=false;`	`Wants2=false;`
`}`	`}`
`/* Remainder section */`	`/* Remainder section */`

Which one of the following statements is TRUE about the above construct?

(a) It does not ensure mutual exclusion.
(b) It does not ensure bounded waiting.
(c) It requires that processes enter the critical section in strict alternation.
(d) It does not prevent deadlocks, but ensures mutual exclusion.

(GATE 2007: 2 Marks)

Solution: When wants1 and wants2 both become true, both processes will be in wait state. But mutual exclusion is ensured.

Ans. (d)

Linked Answer Questions 38 and 39: A process has been allocated 3-page frames. Assume that none of the pages of the process are available in the memory initially. The process makes the following sequence of page references (reference string): 1, 2, 1, 3, 7, 4, 5, 6, 3, 1.

38. If optimal page replacement policy is used, how many page faults occur for the above reference string?

(a) 7 (b) 8
(c) 9 (d) 10

(GATE 2007: 2 Marks)

Solution:
Using optimal page replacement: page faults = 7

1
$2 \rightarrow 7 \rightarrow 4 \rightarrow 5 \rightarrow 6$
3

Ans. (a)

39. Least recently used (LRU) page replacement policy is a practical approximation to optimal page replacement. For the above reference string, how many more page faults occur with LRU than with the optimal page replacement policy?

(a) 0 (b) 1
(c) 2 (d) 3

(GATE 2007: 2 Marks)

Solution:
Using LRU: page faults = 9

$1 \rightarrow 4 \rightarrow 3$
$2 \rightarrow 7 \rightarrow 6$
$3 \rightarrow 5 \rightarrow 1$

$9 - 7 = 2$ more page faults when LRU is used.

Ans. (c)

40. The data blocks of a very large file in the Unix file system are allocated using

(a) contiguous allocation.
(b) linked allocation.
(c) indexed allocation.
(d) an extension of indexed allocation

(GATE 2008: 1 Mark)

Solution: i-node is used in Unix file system.

Ans.(d)

41. For a magnetic disk with concentric circular tracks, the seek latency is not linearly proportional to the seek distance due to

(a) non-uniform distribution of requests.
(b) arm starting and stopping inertia.
(c) higher capacity of tracks on the periphery of the platter.
(d) use of unfair arm scheduling policies.

(GATE 2008: 1 Mark)

Solution: For a magnetic disk with concentric circular tracks, the seek latency is not linearly proportional to the seek distance due to arm starting and stopping inertia.

Ans. (b)

42. In an instruction execution pipeline, the earliest that the data TLB (Translation Lookaside Buffer) can be accessed is

(a) before effective address calculation has started.
(b) during effective address calculation.
(c) after effective address calculation has completed.
(d) after data cache lookup has completed.

(GATE 2008: 2 Marks)

Solution: For effective address calculation, TLB can be accessed earliest.

Ans. (b)

43. The P and V operations on counting semaphores, where s is a counting semaphore, are defined as follows:

```
P(s) : s = s - 1;
            if s < 0 then wait;
V(s) : s = s + 1;
            if s <= 0 then wakeup a
              process waiting on s;
```

Assume that P_b and V_b the wait and signal operations on binary semaphores are provided. Two

binary semaphores x_b and y_b are used to implement the semaphore operations $P(s)$ and $V(s)$ as follows:

```
P(s) : Pb (xb);
            s = s -1;
            if (s < 0) {
               Vb (xb);
               Pb (yb);
            }
            else Vb (xb);

V(s) : Pb (xb);
            s = s + 1;
            if (s <=0) Vb (yb);
               Vb (xb);
```

The initial values of x_b and y_b are, respectively,

(a) 0 and 0 (b) 0 and 1
(c) 1 and 0 (d) 1 and 1

(GATE 2008: 1 Mark)

Solution: x_b and y_b are binary semaphores. $P(S)$ decrements the value of semaphore and $V(S)$ increments it. To ensure the property of mutual exclusion, value passed to x_b should be 1 and value for y_b should be 0.

Ans.(c)

44. Which of the following statements about synchronous and asynchronous I/O is NOT true?

(a) An ISR is invoked on completion of I/O in synchronous I/O but not in asynchronous I/O.
(b) In both synchronous and asynchronous I/O, an ISR (Interrupt Service Routine) is invoked after completion of the I/O.
(c) A process making a synchronous I/O call waits until I/O is complete, but a process making an asynchronous I/O call does not wait for completion of the I/O.
(d) In the case of synchronous I/O, the process waiting for the completion of I/O is woken up by the ISR that is invoked after the completion of I/O.

(GATE 2008: 1 Mark)

Solution: Interrupt is generated on completion of both synchronous and asynchronous I/O. In synchronous, the interrupt wakes up another process that is waiting for I/O, whereas in asynchronous, interrupt informs the other processes that the I/O is complete and they can process their I/O operation.

Ans. (b)

45. Which of the following is NOT true of deadlock prevention and deadlock avoidance schemes?

(a) In deadlock prevention, the request for resources is always granted if the resulting state is safe.
(b) In deadlock avoidance, the request for resources is always granted if the result state is safe.
(c) Deadlock avoidance is less restrictive than deadlock prevention.
(d) Deadlock avoidance requires knowledge of resource requirements *a priori*.

(GATE 2008: 1 Mark)

Solution: Deadlock avoidance scheme is less restrictive than deadlock prevention because in the latter, the request for a resource may not be granted even if the resulting state is safe.

Ans. (c)

46. A process executes the following code

```
for (i = 0; i < n; i + +) for ( );
```

The total number of child processes created is

(a) n (b) $2^n - 1$
(c) 2^n (d) $2^{n+1} - 1$

(GATE 2008: 1 Mark)

Solution:
1 fork system call creates 1 child processes.
2 fork calls create 3 child processes, and so on
With n fork calls, there will be $2^n - 1$ child processes created.

Ans. (b)

47. A processor uses 36-bit physical addresses and 32-bit virtual addresses, with a page frame size of 4 KB. Each page table entry is of size 4 bytes. A three-level page table is used for virtual to physical address translation, where the virtual address is used as follows

- Bits 30–31 are used to index into the first-level page table.
- Bits 21–29 are used to index into the second-level page table.
- Bits 12–20 are used to index into the third-level page table.
- Bits 0–11 are used as offset within the page.

The number of bits required for addressing the next level page table (or page frame) in the page table entry of the first, second and third level page tables are, respectively,

(a) 20, 20 and 20 (b) 24, 24 and 24
(c) 24, 24 and 20 (d) 25, 25 and 24

(GATE 2008: 1 Mark)

Solution:
Given that
Physical address size = 36 bits
Virtual address size = 32 bits
Physical memory size = 2^{36} bytes
Page frame size = 4K bytes = 2^{12} bytes

Number of bits required to access physical memory frame = 36 − 12 = 24, so in third level of page table to access an entry 24 bits are required.
To access second-level page table entry, we require 9-bit of virtual address and size of pages in second level of page table is 4 bytes. So, the size of second level page table is $(2^9) \times 4 = 2^{11}$ bytes.
There are $(2^{36})/(2^{11})$ possible locations to store this page table. Therefore, the second page table requires 25 bits to address it.
Similarly, we can calculate that for first level page table needs 25 bits to address it.

Ans. (b)

48. In which one of the following page replacement policies, Belady's anomaly may occur?

(a) FIFO (b) Optimal
(c) LRU (d) MRU

(GATE 2009: 1 Mark)

Solution: Belady's anomaly refers to the phenomenon where increasing the number of page frames results in increase in the number of page faults. This phenomenon is experienced when FIFO page replacement algorithm is used.

Ans. (a)

49. The essential content(s) in each entry of a page table is/are

(a) virtual page number
(b) page frame number
(c) both virtual page number and page frame number
(d) access right information

(GATE 2009: 1 Mark)

Solution:
Page frame number is must in page table.

Ans. (b)

50. Consider a system with four types of resources R_1 (3 units), R_2 (2 units), R_3 (3 units), R_4 (2 units). A non-preemptive resource allocation policy is used. At any given instance, a request is not entertained if it cannot be completely satisfied. Three processes P_1, P_2, P_3 request the sources as follows if executed independently.

Process P_1:	**Process P_2:**	**Process P_3:**
$t = 0$: requests 2 units of R_2	$t = 0$: requests 2 units of R_3	$t = 0$: requests 1 unit of R_4
$t = 1$: requests 1 unit of R_3	$t = 2$: requests 1 unit of R_4	$t = 2$: requests 2 units of R_1
$t = 3$: requests 2 units of R_1	$t = 4$: requests 1 unit of R_1	$t = 5$: releases 2 units of R_1
t=5: releases 1 unit of R_2 and 1 unit of R_1	$t = 6$: releases 1 unit of R_3	$t = 7$: requests 1 unit of R_2
	$t = 8$: Finishes	$t = 8$: requests 1 unit of R_3
$t = 7$: releases 1 unit of R_3		$t = 9$: Finishes
$t = 8$: requests 2 units of R_4		
$t = 10$: Finishes		

Which one of the following statements is TRUE if all three processes run concurrently starting at time $t = 0$?

(a) All processes will finish without any deadlock.
(b) Only P_1 and P_2 will be in deadlock.
(c) Only P_1 and P_3 will be in a deadlock.
(d) All three processes will be in deadlock.

(GATE 2009: 2 Marks)

Solution: Apply deadlock detection algorithmand see that there is no process waiting indefinitely for a resource.

Ans. (a)

51. Consider a disk system with 100 cylinders. The requests to access the cylinders occur in the following sequence:

4, 34, 10, 7, 19, 73, 2, 15, 6, 20

Assuming that the head is currently at cylinder 50, what is the time taken to satisfy all requests if it takes 1ms to move from one cylinder to adjacent one and shortest seek time first policy is used?

(a) 95 ms (b) 119 ms
(c) 233 ms (d) 276 ms

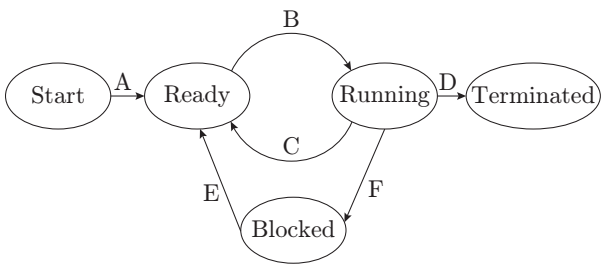
(GATE 2009: 2 Marks)

Solution: 4, 34, 10, 7, 19, 73, 2, 15, 6, 20
Since shortest seek time first policy is used, head will first move to 34. This move will cause 16 × 1 ms. After 34, head will move to 20 which will cause 14 × 1 ms and so on. So, cylinders are accessed in the following order 34, 20, 19, 15, 10, 7, 6, 4, 2, 73 and total time will be

$(16 + 14 + 1 + 4 + 5 + 3 + 1 + 2 + 2 + 71) \times 1 = 119$ ms

Ans. (b)

52. In the following process state transition diagram for a uniprocessor system, assume that there are always some processes in the ready state:



Now, consider the following statements:

I. If a process makes a transition D, it would result in another process making transition A immediately.
II. A process P_2 in blocked state can make transition E, while another process P_1 is in running state.
III. The OS uses pre-emptive scheduling.
IV. The OS uses non-preemptive scheduling.
Which of the above statements are TRUE?

(a) I and II (b) I and III
(c) II and III (d) II and IV

(GATE 2009: 2 Marks)

Solution: Arrow C shows pre-emptive scheduling. II is also true, a process can be in ready queue after completing I/O operation.

Ans. (c)

53. The `enter_CS()` and `leave_CS()` functions to implement critical section of a process are realized using test-and-set instruction as follows:

```
void enter_CS(X)
  {
   ( )
   while test-and-set(X) ;
  }
 void leave_CS(X)
  {
   X=0;
  }
```

In the above solution, X is a memory location associated with the CS and is initialized to 0. Now consider the following statements:

I. The above solution to CS problem is deadlock-free.
II. The solution is starvation-free.
III. The processes enter CS in FIFO order.
IV. More than one process can enter CS at the same time.

Which of the above statements is TRUE?

(a) I only (b) I and II
(c) II and III (d) IV only

(GATE 2009: 2 Marks)

Solution:
The code segment checks that deadlock should not occur, as the functions `test-and-set` and `leave_CS` initialize memory location x to 0.

Ans. (a)

54. A multilevel page table is preferred in comparison to a single-level page table for translating virtual address to physical address because

(a) it reduces the memory access time to read or write a memory location.
(b) it helps to reduce the size of page table needed to implement the virtual address space of a process.
(c) it is required by the translation look-aside buffer.
(d) it helps to reduce the number of page faults in page replacement algorithms.

(GATE 2009: 2 Marks)

Solution: To reduce the page table size, multilevel paging is used.

Ans. (b)

55. Consider the methods used by processes P_1 and P_2 for accessing their critical sections whenever needed, as given below. The initial values of shared Boolean variables S_1 and S_2 are randomly assigned.

Method used by P_1	Method used by P_2
while ($S_1 == S_2$);	while (S_1 != S_2);
Critica1 section	Critica1 section
$S_1 = S_2$;	S_2 = not (S_1);

Which one of the following statements describes the properties achieved?

(a) Mutual exclusion but not progress
(b) Progress but not mutual exclusion
(c) Neither mutual exclusion nor progress
(d) Both mutual exclusion and progress

(GATE 2010: 1 Mark)

Solution: Both P_1 and P_2 are checking different conditions, so ensuring mutual exclusion. Progress is not shown by P_1 and P_2, because they are making their own condition true again.

Ans. (a)

56. Two alternative packages A and B are available for processing a database having 10^1 records. Package A requires 0.0001 n^2 time units and package B requires $10n\log_{10}n$ time units to process n records. What is the smallest value of k for which package B will be preferred over A?

(a) 12 (b) 10 (c) 6 (d) 5

Solution:

$0.0001\ n^2 < 10\ n \log n$

For $k = 6$, B will be preferred.

Ans. (c)

57. A system uses FIFO policy for page replacement. It has 4-page frames with no pages loaded to begin with. The system first accesses 100 distinct pages in some order and then accesses the same 100 pages, but now in the reverse order. How many page faults will occur?

(a) 196 (b) 192 (c) 197 (d) 195

(GATE 2010: 1 Mark)

Solution: Page faults that occur $= 100 + 100 - 4$
$= 196$

4 is subtracted because the last four will be present in frames.

Ans. (a)

58. Which of the following statements are true?

I. Shortest remaining time first scheduling may cause starvation.
II. Pre-emptive scheduling may cause starvation.
III. Round robin is better than FCFS in terms of response time.

(a) I only (b) I and III only
(c) II and III only (d) I, II and III

(GATE 2010: 1 Mark)

Solution:

I. High remaining time jobs will starve.
II. In pre-emptive scheduling, low priority jobs will suffer.
III. Response time will be better round robin because each process will get chance to execute on periodic basis.

Ans. (d)

59. The following program consists of three concurrent processes and three binary semaphores. The semaphores are initialized as $S_0 = 1$, $S_1 = 0$, $S_2 = 0$.

Process P_0	Process P_1	Process P_2
`while (true)`	`wait (S1);`	`wait (S2);`
`{`	`release (S0);`	`release (S0);`
`wait (S0);`		
`print '0'`		
`release (S1);`		
`release (S2);`		
`}`		

How many times will process P_0 print '0'?

(a) At least twice (b) Exactly twice
(c) Exactly thrice (d) Exactly once

(GATE 2010: 2 Marks)

Solution: $S_0 = 1$, only P_0 will execute, it will print 0. P_0 will increment S_1 to 1, process P_1 will start executing which will again make S_0 to 1. Process P_0 will again print 0. Similarly, P_2 will perform the action. Hence, at least twice 0 will be printed.

Ans. (a)

60. A system has n resources R_0, ..., R_{n-1}, and k processes P_0, ..., P_{k-1}. The implementation of the resource request logic of each process P_i is as follows:

```
 if (i%2==0)
 {
 if (i<n) request Ri;
 if (i+2<n)request Ri+2;
 }
else
 {
   if (i<n) request Rn-i;
   if (i+2<n)request Rn-i-2;
  }
```

In which one of the following situations is a deadlock possible?

(a) $n = 40$, $k = 26$ (b) $n = 21$, $k = 12$
(c) $n = 20$, $k = 10$ (d) $n = 41$, $k = 19$

(GATE 2010: 2 Marks)

Solution: Using option (b), we have

if k = 12, then $p_{12-1} = p_{11} \Rightarrow i = 10$ then else block is executed and R_{n-i}, $R_{n-i-2} = R_{21-11}$, $R_{21-11-2} = R_{10}$, R_8. But R_{10} is also requested when k = 11, so this results in dead lock.

Ans. (b)

61. Let the time taken to switch between user and kernel modes of execution be t_1 while the time

taken to switch between two processes be t_2. Which of the following is TRUE?

(a) $t_1 > t_2$
(b) $t_1 = t_2$
(c) $t_1 < t_2$
(d) Nothing can be said about the relation between $t_1$and t_2

(GATE 2011: 1 Mark)

Solution: Process switching requires more time.

Ans. (c)

62. A thread is usually defined as a 'light weight process' because an operating system (OS) maintains smaller data structures for a thread than for a process. In relation to this, which of the following is TRUE?

(a) On per-thread basis, the OS maintains only CPU register state.
(b) The OS does not maintain a separate stack for each thread.
(c) On per-thread basis, the OS does not maintain virtual memory state.
(d) On per-thread basis, the OS maintains only scheduling and accounting information.

(GATE 2011: 1 Mark)

Solution: Operating System maintains separate stack, register and thread-specific data for different threads. OS does not maintain virtual memory state. So option (c) is correct.

Ans. (c)

63. Let the page fault service time be 10 ms in a computer with average memory access time being 20 ns. If one page fault is generated for every 10^6 memory accesses, what is the effective access time for the memory?

(a) 21 ns (b) 30 ns
(c) 23 ns (d) 35 ns

(GATE 2011: 1 Mark)

Solution: p = page fault rate = $1/10^6$

Effective access time = $p \times$ page fault service time + $(1 - p) \times$ memory access time

$= 1/10^6 \times 10 + [(1 - (1/10^6)] \times 20 \approx 30$ ns

Ans. (b)

64. Consider the following table of arrival time and burst time for three processes P_0, P_1 and P_2.

Process	Arrival Time	Burst Time
P_0	0 ms	9 ms
P_1	1 ms	4 ms
P_2	2 ms	9 ms

The pre-emptive shortest job first scheduling algorithm is used. Scheduling is carried out only at arrival or completion of processes. What is the average waiting time for the three processes?

(a) 5.0 ms (b) 4.33 ms
(c) 6.33 ms (d) 7.33 ms

(GATE 2011: 2 Marks)

Solution:

P_0	P_1	P_0	P_2

0 1 5 13 22

Waiting time

$P_0 = 5 - 1 = 4$ ms
$P_1 = 1 - 1 = 0$ ms
$P_2 = 13 - 2 = 11$ ms

Average waiting time = $4 + 11/3 = 5$ ms

Ans. (a)

65. An application loads 100 libraries at startup. Loading each library requires exactly one disk access. The seek time of the disk to a random location is given as 10 ms. Rotational speed of disk is 6000 rpm. If all 100 libraries are loaded from random locations on the disk, how long does it take to load all libraries? (The time to transfer data from the disk block once the head has been positioned at the start of the block may be neglected.)

(a) 0.50 s (b) 1.50 s
(c) 1.25 s (d) 1.00 s

(GATE 2011: 2 Marks)

Solution: Since transfer time can be neglected, the average access time is sum of average seek time and average rotational latency. Average seek time for a random location time is given as 10 ms. The average rotational latency is half of the time needed for complete rotation. It is given that 6000 rotations need 1 min. So one rotation will take 60/6000 s, which is 10 ms. Therefore, average rotational latency is half of 10 ms, which is 5 ms.

Average disk access time = Seek time + rotational latency
= 10 ms + 5 ms = 15 ms

For 100 libraries, the average disk access time will be 15×100 ms.

Ans. (b)

66. A process executes the code

```
fork();
fork();
fork();
```

The total number of child processes created is

(a) 3 (b) 4
(c) 7 (d) 8

(GATE 2012: 1 Mark)

Solution: fork() system call is used to create child processes.

Total number of processes: 2^n
Child processes: $2^n - 1$, here n represents the number of fork calls.

Ans. (c)

67. Consider the three processes, P_1, P_2 and P_3 shown in the table.

Process	Arrival Time	Time Units Required
P_1	0	5
P_2	1	7
P_3	3	4

The completion order of the three processes under the policies FCFS and RR2 (round-robin scheduling with CPU quantum of 2 time units) are

(a) FCFS: P_1, P_2, P_3
RR2: P_1, P_2, P_3
(b) FCFS: P_1, P_3, P_2
RR2: P_1, P_3, P_2
(c) FCFS: P_1, P_2, P_3
RR2: P_1, P_3, P_2
(d) FCFS: P_1, P_3, P_2
RR2: P_1, P_2, P_3

(GATE 2012: 2 Marks)

Solution: FCFS Scheduling:

P_1	P_2	P_3

0 5 12 16

Time of completion for P_1: $5 - 0 = 5$
P_2: $12 - 1 = 11$
P_3: $16 - 3 = 13$

Order of completion is P_1, P_2, P_3.

RR2 Scheduling:

P_1	P_2	P_1	P_3	P_2	P_2	P_3	P_2	P_2

0 2 4 6 8 10 11 13 15 16

Time of completion for P_1: $11 - 0 = 11$
P_2: $16 - 1 = 15$
P_3: $13 - 3 = 10$

Order of completion is P_1, P_3, P_2.

Ans. (c)

68. `Fetch_And_Add(X, i)` is an atomic read-modify-write instruction that reads the value of memory location X, increments it by the value i, and returns the old value of X. It is used in the pseudocode shown below to implement a busy-wait lock. L is an unsigned integer shared variable initialized to 0. The value of 0 corresponds to lock being available, while any non-zero value corresponds to the lock being not available.

```
AcquireLock(L){
        while (Fetch_And_Add(L,1))
                L = 1;
}
ReleaseLock(L){
        L = 0;
}
```

This implementation
(a) fails as L can overflow.
(b) fails as L can take on a non-zero value when the lock is actually available.
(c) works correctly but may starve some processes.
(d) works correctly without starvation.

(GATE 2012: 2 Marks)

Solution: Let a process has just released the lock and made L = 0 and there is one more process waiting for the lock, that process will then execute the AcquireLock(). Just after the L was made 0, let the waiting processes executed the line L = 1. And then all the waiting process will execute AcquireLock() and they cannot come out of the while loop.

Ans. (b)

69. A file system with 300 GB disk uses a file descriptor with 8 direct block addresses, 1 indirect block address and 1 doubly indirect block address. The size of each disk block is 128 bytes and the size of each disk block address is 8 bytes. The maximum possible file size in this file system is

(a) 3 KB
(b) 35 KB
(c) 280 bytes
(d) Dependent on the size of the disk

(GATE 2012: 2 Marks)

Solution:
Disk block size = 128 bytes
Disk block addresses = 8 bytes
Number of entries = 128/8 = 16
8 direct block addresses = $8 \times 128 = 1024$
1 indirect block addresses = $16 \times 128 = 2048$
1 doubly indexes = $16 \times 16 \times 128 = 32768$

35 KB

Ans. (b)

70. Consider the virtual page reference string

1, 2, 3, 2, 4, 1, 3, 2, 4, 1

On a demand paged virtual memory system running on a computer system with main memory size of 3 pages frames which are initially empty. Let LRU, FIFO and OPTIMAL denote the number of page faults under the corresponding page replacements policy. Then

(a) OPTIMAL < LRU < FIFO
(b) OPTIMAL < FIFO < LRU

(c) OPTIMAL = LRU
(d) OPTIMAL = FIFO

(GATE 2012: 2 Marks)

Solution:

LRU:	1 1 1 4 4 4 2 2 2	9 page faults
	2 2 2 2 3 3 3 1	
	3 3 1 1 1 4 4	
FIFO:	1 1 1 4 4 4	6 page faults
	2 2 2 1 1	
	3 3 3 2	
OPTIMAL:	1 1 1 1 1	5 page faults
	2 2 4 4	
	3 3 2	

OPTIMAL < FIFO < LRU

Ans.(b)

71. Consider the following transactions with data items P and Q initialized to zero:

T_1: read (P);
read (Q);
if P = 0 then Q := Q + 1;
write (Q).
T_2: read (Q);
read (P);
if Q = 0 then P := P + 1;
write (P).

Any **non-serial** interleaving of **T_1** and **T_2** for concurrent execution leads to

(a) a serializable schedule
(b) a schedule that is not conflict serializable
(c) a conflict serializable schedule
(d) a schedule for which a precedence graph cannot be drawn

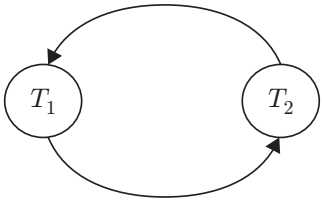
(GATE 2012: 2 Marks)

Solution: Two or more actions are said to be in conflict if:

(i) The actions belong to different transactions.
(ii) At least one of the actions is a write operation.
(iii) The actions access the same object (read or write).

Take the following schedule:

S: $r_1(P)$; $r_1(Q)$; $r_2(Q)$; $r_2(P)$; $w_1(Q)$; $w_2(P)$



Cycle exists between two transactions, so T_1 and T_2 are not conflict serializable.

Ans. (b)

72. A scheduling algorithm assigns priority proportional to the waiting time of a process. Every process starts with priority zero (the lowest priority). The scheduler re-evaluates the process priorities every T time units and decides the next process to schedule. Which one of the following is **TRUE** if the processes have no I/O operations and all arrive at time zero?

(a) This algorithm is equivalent to the first-come-first-serve algorithm.
(b) This algorithm is equivalent to the Round-Robin algorithm.
(c) This algorithm is equivalent to the shortest-job-first algorithm.
(d) This algorithm is equivalent to the shortest-remaining-time-first algorithm.

(GATE 2013: 1 Mark)

Solution: Because time quantum T is given here to reschedule priority. So, it is equivalent to Round Robin algorithm.

Ans. (b)

73. Consider a hard disk with 16 recording surfaces (0-15) having 16384 cylinders (0-16383) and each cylinder contains 64 sectors (0-63). Data storage capacity in each sector is 512 bytes. Data are organized cylinder-wise and the addressing format is <cylinder no., surface no., sector no.>. A file of size 42797 KB is stored in the disk and the starting disk location of the file is <1200, 9, 40>. What is the cylinder number of the last sector of the file, if it is stored in a contiguous manner?

(a) 1281 (b) 1282 (c) 1283 (d) 1284

(GATE 2013: 1 Mark)

Solution: The number of sectors required to store the file $= \dfrac{42797 \times 1024}{512} = 85594$ sectors

Number of sectors in a cylinder = 16 × 64 = 1024

Total number of cylinders required = 85594/1024
= 84

Last sector will be stored on 1284th cylinder.

Ans. (d)

74. A certain computation generates two arrays a and b such that $a[i]=f(i)$ for $0 \le i < n$ and $b[i]=g(a[i])$ for $0 \le i < n$. Suppose this computation is decomposed into two concurrent processes X and Y such that X computes the array a and Y computes the array b. The processes employ two binary semaphores R and S, both initialized to zero. The array a is shared by the two processes. The structures of the processes are shown below.

Process Y:	Process Y:

`private i;`
`for (i=0; i < n; i++) {`
`  a[i] = f(i);`
`  ExitX(R, S);`
`}`
	`private i;`
	`for (i=0; i < n; i++) {`
	`  EntryY(R, S);`
	`  b[i]=g(a[i]);`
	`}`

Which one of the following represents the CORRECT implementations of ExitX and EntryY?

(A)

```
ExitX(R, S) {
    P(R);
    V(S);
}
EntryY (R, S) {
    P(S);
    V(R);
}
```

(B)

```
ExitX(R, S) {
    V(R);
    V(S);
}
EntryY(R, S) {
    P(R);
    P(S);
}
```

(C)

```
ExitX(R, S) {
    P(S);
    V(R);
}
EntryY(R, S) {
    V(S);
    P(R);
}
```

(D)

```
ExitX(R, S) {
    V(R);
    P(S);
}
EntryY(R, S) {
    V(S);
    P(R);
}
```

(a) A (b) B
(c) C (d) D

(GATE 2013: 2 Marks)

Solution: P(S) is for wait, V(S) is for signal. In option (c), mutual exclusion is ensured. Deadlock also does not occur. Other options are leading either to deadlock or do not ensure mutual exclusion.

Ans.(c)

75. A shared variable x, initialized to zero, is operated on by four concurrent processes W, X, Y, Z as follows. Each of the processes W and X reads x from memory, increments by one, stores it to memory, and then terminates. Each of the processes Y and Z reads x from memory, decrements by two, stores it to memory, and then terminates. Each process before reading x invokes the P operation (i.e., *wait*) on a counting semaphore S and invokes the V operation (i.e., *signal*) on the semaphore S after storing x to memory. Semaphore S is initialized to two. What is the maximum possible value of x after all processes complete execution?

(a) -2 (b) -1 (c) 1 (d) 2

(GATE 2013: 2 Marks)

Solution: $S = 2$ (only two processes can run at the same time)
$x = 0$

W	X	Y	Z
P(S)	P(S)	P(S)	P(S)
R(x)	R(x)	R(x)	R(x)
x++	x++	x=x−2	x=x−2
W(x)	W(x)	W(x)	W(x)
V(S)	V(S)	V(S)	V(S)

Let X and Y are running:
X: R(x) → 0 Y: R(x) → 0
x++ = 1 x = x − 2 = −2
w(x) =1 w(x) = −2
Let X wrote after Y, so x will contain 1.
Now when W and Z will execute, W will increment it to 2 and Z will decrement to −1, so maximum possible values is 2.

Ans. (d)

Common Data Questions 76 and 77: A computer uses 46-bit virtual address, 32-bit physical address, and a three-level paged page table organization. The page table base register stores the base address of the first-level table (T_1), which occupies exactly one page. Each entry of T_1 stores the base address of a page of the second-level table (T_2). Each entry of T_2 stores the base address of a page of the third-level table (T_3). Each entry of T_3 stores a page table entry (PTE). The PTE is 32 bits in size. The processor used in the computer has a 1 MB 16-way set-associative virtually indexed physically tagged cache. The cache block size is 64 bytes.

76. What is the size of a page in KB in this computer?

(a) 2 (b) 4
(c) 8 (d) 16

(GATE 2013, 2 Marks)

Solution: All the page table resides in the physical memory (RAM) since the physical address is 32 bit addressable. Each level table holds a 32-bit (4-byte) address of the RAM.

Assume the size of a page is x bytes (which we need to find). As the size of level-1 table is also x (as given in the question that level-1 table accommodates exactly a page), it can hold $x/4$ base addresses of $x/4$ level-2 tables (which also accommodates exactly a page) and each level-2 table holds base addresses of $x/4$ level-3 tables.
The level-3 table has entities equal to $2^{46}/x$
That is, $(x/4)(x/4)(x/4) = 2^{46}/x$
On solving, $x = 8192$, which is in bytes so 8192/1024 = 8 KB

Ans. (c)

77. Consider the same data as in the above question. What is the minimum number of page colours needed to guarantee that no two synonyms map to different sets in the processor cache of this computer?

(a) 2 (b) 4
(c) 8 (d) 16

(GATE 2013: 2 Marks)

Solution:
Let page sixe is x bytes.
Size of level 1 table is also x (given in the question).
It can hold x/4 base addresses of x/4 level 2 tables and each level 2 table can hold base addresses of x/4 level 3 tables.
The level 3 table has entities equal to 2^{46}/x, that is, (x/4)(x/4)(x/4) = 2^{46}/x
On solving $x = 8192$ bytes = 8 K bytes

Ans. (c)

PRACTICE EXERCISES

Set 1

1. Context switching is part of

(a) Buffering
(b) Process execution state
(c) Interrupt handling
(d) Ready state

2. Interrupt is

(a) External to the execution of the current instruction
(b) Associated with the execution of the current instruction
(c) Explicit request
(d) Internal to the execution of the current program

3. Process no longer exist, but it leaves a record for its parent process to collect, the state of the process is

(a) spawned. (b) slipend.
(c) zombie. (d) running.

4. In round-robin scheduling algorithm, throughput will

(a) Remain as it is
(b) Will be decreased
(c) Will be increased
(d) It depends, may either increase or decrease

5. Paging is a solution to overcome

(a) Internal fragmentation
(b) External fragmentation
(c) Middle fragmentation
(d) Both internal and external fragmentation

6. Paging suffers from the following limitation(s)

(a) Internal fragmentation
(b) External fragmentation
(c) Middle fragmentation
(d) Both internal and external fragmentation

7. Segmentation is solution to overcome

(a) Internal fragmentation
(b) External fragmentation
(c) Middle fragmentation
(d) Both internal and external fragmentation

8. Segmentation suffers from the following limitation(s)

(a) Both internal and external fragmentation
(b) External fragmentation
(c) Middle fragmentation
(d) Internal fragmentation

9. Physical base address of each page in physical memory is associated with?

(a) Main memory
(b) Virtual memory
(c) Page table
(d) Cache memory

10. External fragmentation can be overcome by

1. Compaction
2. Paging
3. Segmentation

(a) 1 and 2 (b) 2 and 3
(c) 1 and 3 (d) 1 and 2 and 3

11. In semaphores, the term P and V stands for

(a) Wait, Signal
(b) Signal, Wait
(c) Both a and b
(d) None

12. The time taken to move the desired sector under the R/W head is called

(a) Seek time delay
(b) Rotational time delay
(c) Transfer time delay
(d) Queuing time delay

13. Banker's algorithm is applied for

(a) Deadlock detection
(b) Deadlock avoidance
(c) Deadlock recovery
(d) Both a and c

14. Solution to the thrashing is

(a) Working set model
(b) Increasing the degree of multiprogramming
(c) Increasing the size of page table
(d) Decreasing the size of the page

15. Consider a memory paging system where TLB access time is 20 ns and main memory access time is 200 ns. If the hit ratio in TLB is 90%, find the effective access time in ns is ____.

16. In the following resource allocation graph, every resource has single instance, then cycle means

(a) Deadlock exists
(b) Deadlock may exists
(c) Deadlock can be avoided
(d) Deadlock can be recovered

17. In a resource allocation graph, every resource has multiple instances, then cycle means

(a) Deadlock exists
(b) Deadlock may exists
(c) Cycle cannot exists
(d) Deadlock can be avoided

18. Semaphores provide a solution for

(a) Mutual exclusion
(b) Process synchronization
(c) Circular wait
(d) None of the above

19. When an interrupt occurs, an operating system

(a) Ignores the interrupt
(b) Stops the execution of the current instruction
(c) May change the state of the interrupt process to block and schedule another process
(d) Informs the process control block (PCB)

20. Disk I/O is measured in terms of

(a) Blocks (b) Bits
(c) Sectors (d) Tracks

21. Simplest method for deadlock recovery is

(a) Roll back the processes one by one
(b) Pre-empt the resource
(c) Lock one of the resources
(d) Kill one of the processes

22. Let the time taken to switch between user mode and kernel mode of execution be T_1 while time taken to switch between the user mode1 and user mode2 be T_2. Which of the following is true?

(a) $T_1 < T_2$
(b) $T_1 > T_2$
(c) $T_1 = T_2$
(d) T_1 and T_2 cannot be compared

23. A thread is usually defined as a light-weight process because an operating system (OS) maintains smaller data structure for a thread than for a process. In relation to this, which of the following statement is correct?

(a) OS maintains only scheduling and virtual memory information for each thread.
(b) OS maintains only CPU registers for each thread.
(c) OS does not maintain a separate stack for each thread.
(d) OS does not maintain virtual memory state for each thread.

24. The Windows operating system such as Windows XP uses the following file system:

(a) FAT-16, NTFS
(b) FAT-32, NTFS
(c) NTFS (NT File System) only
(d) FAT-16, FAT-32 and NTFS

25. Which of the following statements is not true about disk-arm scheduling algorithms?

(a) SSTF (shortest seek time first) algorithm increases performance of FCFS.
(b) The number of requests for disk service is not influenced by file allocation method.
(c) Caching the directories and index blocks in main memory cannot help in reducing disk arm movements.
(d) SCAN and C-SCAN algorithms are less likely to have a starvation problem.

26. _____ maintains the list of free disk blocks in a Unix operating system.

(a) i-Node
(b) Boot node pointer
(c) Super block
(d) File allocation table

27. A part of Windows 2000 operating system that is not transportable is

(a) Device Management Scheme
(b) Virtual Memory Management Scheme
(c) Process Management Scheme
(d) Graphical User Interface Scheme

28. Windows 32 API supports

(a) 16 Bit
(b) 32-Bit and 64-bit Windows
(c) Only 32-bit Windows
(d) 16-Bit, 32-bit and 64-bit Windows

29. How an operating system does make sure that a particular program has had a successful completion?

(a) The return value is 0.
(b) The return value is 1.
(c) The return value is –1.
(d) The program does not return a value.

30. Consider the following program

```
main()
{
fork();
fork();
fork();
}
```

How many new processes will be created?

(a) 9 (b) 6
(c) 7 (d) 5

31. A fence register is basically used for

(a) Memory protection
(b) Block protection
(c) File protection
(d) CPU protection

32. Consider the following program

```
main()
{
fork();
exec("c:/notepad.exe");
fork();
}
```

How many new processes will be created?

(a) 4 (b) 2
(c) 3 (d) 1

33. Fork system call creates the child process on successful completion of fork, which of the following value is returned in the child process?

(a) p_id of parent process
(b) p_id of child process
(c) p_id of both the child process and parent process
(d) zero

34. Suppose that a process is in blocked state waiting for some I/O devices, now when the I/O device is done with the serving to a process, it was earlier doing, the process changes its state to the

(a) Running state
(b) Ready state
(c) Suspended state
(d) Terminated state

35. What is shell?

(a) It is hardware component.
(b) It is a command interpreter.
(c) It is a compiler component.
(d) It is a tool used for CPU scheduling.

36. When an OS creates a process at the explicit request of another process, this action is known as

(a) Process cloning
(b) Process slipping
(c) Process spawning
(d) Process borne

37. In threads, which mapping is called pure user level threads?

(a) 1:1 Mapping
(b) N:1 Mapping
(c) N:M Mapping
(d) Null Mapping

38. Which type of scheduler controls degree of multiprogramming?

(a) Long-term scheduler
(b) Short-term scheduler
(c) Medium-term scheduler
(d) Hybrid-term scheduler

39. Consider the following program

P_1	P_2
P(mutex1)	P(mutex2)
P(mutex2)	P(mutex1)
.	.
.	.
.	.
V(mutex1)	V(mutex2)
V(mutex2)	V(mutex1)

Using this program

(a) Deadlock cannot occur.
(b) Deadlock can occur.
(c) Data is insufficient.
(d) Deadlock recovery cannot be done.

40. If thread library is implemented in a kernel space, then each function call will be considered as a

(a) System call
(b) Local function call
(c) Both a and b
(d) User call

41. An OS has to schedule three user processes each requiring 2 units of a particular resource R. The minimum number of units of R such that no deadlocks will ever arise is

(a) 3 (b) 2
(c) 4 (d) 6

42. Which of the following need not be saved on context switch between the various processes?

(a) General purpose register
(b) Translation look-aside buffer
(c) Program counter
(d) Stack register

43. A computer hardware has N number of CPUs, then the maximum number of processes that can be in running state on this hardware will be

(a) $N/2$ (b) $N!$
(c) N (d) $\log_2 N$

44. Page swapping and page replacement is also called

(a) Page cannibalizing
(b) Page copying
(c) Page snooping
(d) Page sharing

45. The commonly used term PCB means

(a) Program control block
(b) Process control block
(c) Process communication block
(d) Program communication block

46. Consider the following scenario, does it represent a:

Process	Allocated	Need	Available
	A B C	A B C	A B C
P_1	0 1 0	135	0 2 0
P_2	203	542	
P_3	323	241	
P_4	212	030	

(a) Safe sequence exists
(b) Safe sequence does not exists
(c) Insufficient data
(d) Deadlock recovery does not exist

47. Consider a process of 256 bytes and a system having main memory of size 1024 bytes with frame size of 8 bytes.

47a. The number of pages will be:

(a) 16 (b) 32
(c) 64 (d) 128

47b. How many frames are there in main memory?

(a) 128 (b) 256
(c) 512 (d) 64

47c. How many entries are there in page table?

(a) 16 (b) 32
(c) 64 (d) 8

47d. What could be the size of page table?

(a) 124 bits (b) 224 bits
(c) 145 bits (d) 128 bits

48. Consider the following snapshot of an operating system required to schedule the execution of the processes:

Processes	Arrival Time (in ms)	CPU Time Needed (in ms)	Priority
P_1	0	10	5
P_2	0	5	2
P_3	2	3	1
P_4	5	20	4
P_5	10	2	3

48a. If CPU scheduling policy is FCFS, then average waiting time will be

(a) 12.8 ms (b) 8 ms
(c) 16 ms (d) 15 ms

48b. If scheduling policy is SJF without pre-emption, then average waiting time will be

(a) 12.8 ms (b) 6.8 ms
(c) 17 ms (d) 15 ms

48c. If CPU scheduling policy is SJF with pre-emption, then average waiting time will be

(a) 8 ms (b) 14 ms
(c) 5.6 ms (d) 15 ms

48d. If CPU scheduling policy is priority scheduling without pre-emption, then average waiting time will be

(a) 12.8 ms (b) 11.8 ms
(c) 10.8 ms (d) 14 ms

48e. If CPU scheduling policy is priority scheduling with pre-emption, then average waiting time will be

(a) 19 ms (b) 7.6 ms
(c) 8 ms (d) 14.2 ms

Set 2

1. Which scheduling policy is most suitable for time-shared operating systems?

(a) Shortest job first
(b) Round robin
(c) First come first serve
(d) Elevator

2. Four jobs to be executed on a single processor system arrive at time 0+ in the order A, B, C, D. Their burst CPU time requirements are 4, 1, 8, 1 time units, respectively. The completion time of A under round-robin scheduling with time slice of one time unit is _____.

3. The sequence is an optimal non-preemptive scheduling sequence for the following jobs which leaves the CPU idle for _____ unit(s) of time.

Job	Arrival Time	Burst Time
1	0.0	9
2	0.6	5
3	1.0	1

(a) {3,2,1},1
(b) {2,1,3},0
(c) {3,2,1},0
(d) {1,2,3},5

4. The Banker's algorithm for avoiding the deadlock has got the following data structure, with four processes P_0,P_1,P_2,P_3 and resourcetypes A, B, C and D.

Process	Allocation	Maximum	Available
	A B C D	A B C D	A B C D
P_0	2 1 2 0	4 1 3 2	3 0 1 2
P_1	1 0 0 2	4 1 0 3	
P_2	2 0 2 1	5 4 4 1	
P_3	2 3 4 1	3 3 6 5	

What will be the safety sequence of the above processes to avoid the deadlock?

(a) P_0, P_3, P_1, P_2 (b) P_0, P_1, P_2, P_3
(c) P_0, P_1, P_3, P_2 (d) P_0, P_2, P_1, P_3

5. Assume the processes are arrived into the ready queue in the correct order in the above question at time t_o. What will be the average waiting time using round-robin scheduling with a time quantum 5 ms? (The CPU burst requirements of P_0, P_1, P_2, P_3 are 5 ms, 10 ms, 7 ms and 8 ms, respectively.)

(a) 14.5 ms (b) 13 ms
(c) 12.5 ms (d) 15 ms

6. When the result of a computation depends on the speed of the processes involved, there is said to be

(a) Cycle stealing (b) Race condition
(c) A time lock (d) A deadlock

7. Each process P_i, $I = 1 \ldots 9$ is coded as follows:

Repeat

```
P (mutex)
{critical section}
V (mutex)
```

Forever
The code uses P_{10} is identical except that it uses V (mutex) in place of P (mutex). What is the largest no of processes that can be inside the critical section at any moment?

(a) 1 (b) 2
(c) 3 (d) None of these

8. Consider Peterson's algorithm for mutual exclusion between two concurrent processes *i* and *j*. The program executed by process is shown below:

```
Repeat
  Flag[i] = true;
  Turn=j;
  While (P) do no -op;
  Enter critical section, perform actions,then
  Then
  Exit critical section
  Flag[i]=false;
  Perform other non-criticalsection actions.
  Until false;
```

For the program to guarantee mutual exclusion, the predicate P in the while loop should be

(a) Flag[j]=true and Turn=i
(b) Flag[j]=true and Turn=j
(c) Flag[i]=true and Turn=j
(d) Flag[i]=true and turn=i

9. Consider an operating system containing 'n' processes each requiring four resources. The maximum number of processes that can be executed without any deadlock if there are 18 resources is _____

10. The average page fault service time in a system is 25 ms and memory access time is 220 ns. If page fault rate is 0.0004, what will be the effective access time?

(a) 10 (b) 7.36
(c) 10.21 (d) 11.36

11. Consider n processes sharing the CPU in a round-robin fashion. Assuming that each process switch takes s seconds, what must be the quantum size q such that the overhead resulting from process switching is minimized but at the same time each process is guaranteed to get its turn at the CPU at least every t seconds?

(a) $q \leq \dfrac{t-ns}{n-1}$ (b) $q \geq \dfrac{t-ns}{n-1}$

(c) $q \leq \dfrac{t-ns}{n+1}$ (d) $q \geq \dfrac{t-ns}{n+1}$

12. A solution to the Dining Philosophers Problem which avoids deadlock is

(a) Ensure that all philosophers pick up the left fork before the right fork
(b) Ensure that all philosophers pick up the right fork before the left fork
(c) Ensure that one particular philosopher picks up the left fork before the right fork, and that all other philosophers pick up the right fork before the left fork
(d) None of these

13. A computer has six tape drives, with n processes competing for them. Each process may need two drives. What is the maximum value of n for the system to be deadlock free?

(a) 6 (b) 5
(c) 4 (d) 4

14. Which of the following is NOT a valid deadlock prevention scheme?

(a) Release all resources before requesting a new resource.
(b) Number the resources uniquely and never request a lower numbered resource than the last one requested.
(c) Never request a resource after releasing any resource.
(d) Request and all required resources be allocated before execution.

15. Consider the disk drive with the following specification:

Eight surfaces, 1024 tracks/surface, 512 sectors/track, 4KB/sector, rotation speed is 3000 rpm and the disk is operated in burst mode. The processor runs at 600 MHz and takes 300 and 900 clock cycle to initiate and complete DMA transfer, respectively, if the size of transferred data is 20KB.

What is the size of the hard disk?

(a) 16 GB (b) 8 GB
(c) 16 MB (d) 8 MB

16. What is the percentage of processor time consumed for the transfer operation in the above case?

(a) 0.69% (b) 27.77%
(c) 97% (d) 20%

17. Consider a system having m resources of the same type. These resources are shared by three processes A, B and C which have peak demands of 3, 5 and 7, respectively. For what value of m deadlock will not occur?

(a) 13 (b) 14
(c) 15 (d) 16

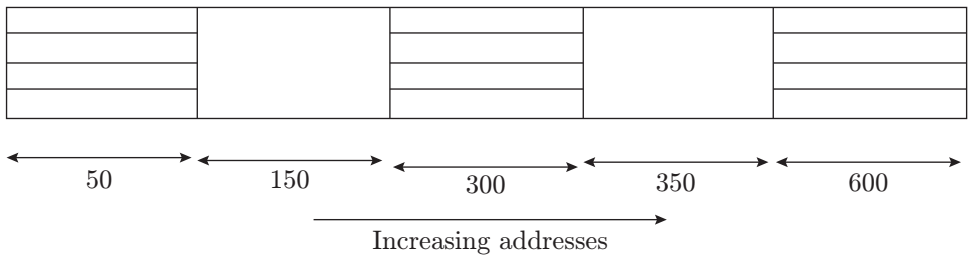
18. A ROM is used to store the table for multiplication of two 8-bit unsigned integers. The size of ROM required is

(a) 256 × 16 (b) 64K × 8
(c) 4K × 16 (d) 64K × 16

19. Dirty bit for a page in a page table

(a) helps avoid unnecessary writes on paging device
(b) helps maintain LRU information
(c) allows only read on a page
(d) none of the above

20. Consider the following heap (figure) in which blank regions are not in use and hatched regions are in use.



The sequence of requests for blocks of size 300, 25, 125, 50 can be satisfied if we use

(a) Either first-fit or best-fit policy(any one)
(b) First-fit but not best-fit policies
(c) Best-fit but not first-fit policy
(d) None of the above

21. The capacity of memory unit is defined by the number of words multiplied by the number of bits/word. How many separate address and data lines are needed for a memory of 4K × 16?

(a) 10 address, 16 data lines
(b) 11 address, 8 data lines
(c) 12 address, 16 data lines
(d) 12 address, 12 data lines

22. If an instruction takes iμs and a page fault takes an additional jμs, the effective instruction time if on the average a page fault occurs every k instruction is

(a) $i + \frac{j}{k}$ (b) $i + j * k$

(c) $\frac{i+j}{k}$ (d) $(i + j) * k$

23. Suppose the time to service a page fault is on the average 10 ms, while a memory access takes 1 μs. Then a 99.99% hit ratio results in average memory access time of

(a) 1.9999 ms
(b) 1 ms
(c) 9.999 μs
(d) 1.9999 μs

24. Consider a process that has been allocated 3 page frames. Assume that system uses pure demand paging. While undergoing execution, the process makes the following sequence of page references 1,4,7,1,6,3,1,3,2,4,9,1.

If optimal page replacement policy is used, then the number of page faults for the above reference string is

(a) 7 (b) 6
(c) 8 (d) None of these

25. If LRU is used for the same reference string in the above question then which of the following is true?

(a) Optimal has 22% lesser page faults than LRU
(b) Optimal has 33% lesser page faults than LRU
(c) Optimal has 44% lesser page faults than LRU
(d) None of these

26. What is the number of page faults by least recently used page replacements for a memory with 4 frames for the page reference string 2, 0, 1, 2, 4, 0, 5, 1, 4, 6, 4, 2, 1, 3, 0?

(a) 7 (b) 8
(c) 9 (d) 10

27. If number of frames is 3, then how many more page faults will occur using the same page reference string?

(a) 0 (b) 1
(c) 2 (d) 3

28. Which of the following disk-scheduling strategies is likely to give the best throughput?

(a) Farthest cylinder next
(b) Nearest cylinder next
(c) First come first serve
(d) Elevator algorithm

29. In the index allocation scheme of blocks to a file, the maximum possible size of the file depends on

(a) The size of blocks, and the size of the address of the blocks.
(b) The number of blocks used for the index, and the size of the blocks.
(c) The size of blocks, the number of blocks used for the index and the size of the address of the blocks.
(d) None of the above

30. Suppose two jobs, each of which needs 10 min of CPU time, start simultaneously. Assume 50% I/O wait time.

How long will it take for both to complete if they run sequentially?

(a) 10 (b) 20
(c) 30 (d) 40

31. How long will it take if they run in parallel in the above case?

(a) 10 (b) 20
(c) 30 (d) 40

32. A multilevel page table is preferred in comparison to a single-level page table for translating virtual address to physical address because

(a) It reduces the memory access time for read and write.
(b) It helps to reduce the size of page table needed to implement the virtual address space of a process.
(c) It is required by translation look-aside buffer.
(d) It helps to reduce the number of page faults in page replacement algorithms.

33. A process executes the following code for($i = 0$; $i < n$; i++) for ();the total number of child processes created is

(a) n (b) 2^{n-1}
(c) $2n$ (d) $2^{(n+1)-1}$

34. A UNIX style i-node has 10 direct pointers and one single, one double and one triple indirect pointers. Disk block size is 1 KB, disk block address is 32 bit and 48 bit integers are used. What is the maximum possible file size?

(a) 2^{24} bytes (b) 2^{32} bytes
(c) 2^{34} bytes (d) 2^{48} bytes

35. Match the following:

List I	List II
a. Critical region	1. Hoarse monitor
b. Wait/signal	2. Mutual exclusion
c. Working set	3. Principal of locality
d. Dead lock	4. Circular wait

(a) 2 1 3 4 (b) 1 2 4 3
(c) 2 3 1 4 (d) 1 3 2 4

ANSWERS TO PRACTICE EXERCISES

Set 1

1. (c)	**12.** (b)	**23.** (b)	**34.** (b)	**45.** (c)
2. (a)	**13.** (b)	**24.** (d)	**35.** (b)	**46.** (b)
3. (c)	**14.** (a)	**25.** (b)	**36.** (c)	**47a.** (b)
4. (b)	**15.** (240)	**26.** (c)	**37.** (a)	**47b.** (a)
5. (b)	**16.** (a)	**27.** (b)	**38.** (a)	**47c.** (b)
6. (a)	**17.** (b)	**28.** (d)	**39.** (b)	**47d.** (b)
7. (a)	**18.** (b)	**29.** (a)	**40.** (a)	**48a.** (a)
8. (b)	**19.** (c)	**30.** (c)	**41.** (c)	**48b.** (b)
9. (c)	**20.** (a)	**31.** (a)	**42.** (b)	**48c.** (c)
10. (a)	**21.** (d)	**32.** (b)	**43.** (c)	**48d.** (c)
11. (a)	**22.** (a)	**33.** (d)	**44.** (a)	**48e.** (b)

Set 2

1. (b) To schedule processes fairly, a round-robin scheduler generally employs time-sharing, giving each job a time slot or quantum (its allowance of CPU time), and interrupting the job if it is not completed by then. It is designed especially for time-sharing systems.

2. (9) Draw Gantt chart according to the round-robin algorithm with time quantum 1.

0	1	2	3	4	5	6	7	8	9	10
A	B	C	D	A	C	A	C	A	C	

So, completion time of A is 9.

3. (a) In the first sequence (3, 2, 1), CPU is ideal for 1 unit time because 3rd process arrives at time 1. Process with large burst time will not have to wait more so CPU will not be ideal more time.

4. (c) Refer text

5. (a) Refer text

6. (b) The condition is said to be race condition.

7. (d) Every time when process P_{10} enters into critical section, it performs the 'UP' operation, so after that one more processor can enter into it. This process continues till end and so 10 processes can enter into critical section at a time.

8. (b) To ensure the mutual exclusion into this program, we have to take care that 'j' should not be allowed to value 'turn'.

9. (7) $5n \leq n + 28 \Rightarrow n = 7$ satisfy the above equation.

10. (c) Effective access time = [(1 − p) × Memory access time + p × page fault service time]

$\Rightarrow [(1 - 0.0004) \times 0.22 + 0.0004 \times 2500]$ μs
$\Rightarrow$10.219

11. (a) Next execution of the process should occur within t seconds. Each process runs for q period, and if there are n processes $p_1\ p_2\ p_3 \cdots p_n\ p_1\ p_2 \cdots$ then p_1 turn comes again when it has completed time quanta for remaining process p_2 to p_n, that is, it would take at most $(n-1)q$ time.
So each process in round-robin gets its turn after $\leq (n-1)q$ time when we do not consider overheads, but if we consider overhead(s) then it would be $ns + (n-1)q$.
So, we have $ns + (n-1)q \leq t$ overhead will be reduced when time quantum is maximum allowable, that is, $q = (t - ns)/(n-1)$

12. (a) It is a condition to implement correctly Dining Philosopher Problem. We have to start checking from left fork to the right fork.

13. (a) Every process needs two resources to complete its task. If we allocate five resources to five processes as one process each, then no one process can finish its task. But if we give one more resource to any process then it will complete its task and free its resources to be used by the other. By which all process can finish their tasks. So answer is 6.

14. (c) This is not a valid deadlock prevention scheme.

15. (a) Disk size $= 8 \times 1024 \times 512 \times 4$ K $= 16 \times 2^{30}$ bytes = 16 GB

16. (b) Refer text

17. (d) The peak demand for each processes A, B and C is given as 3, 5 and 7, respectively. Suppose the peak demand for all the three processes happens simultaneously, then we will require $3 + 5 + 7 = 15$ resources to satisfy the demand.

18. (d) $2^8 \times 2^8 = 64\text{K} \times 16$

19. (d) The pages which have dirty bits are the pages written back in memory, which avoids unnecessary writing.

20. (b) Refer text

21. (c) $4\text{K} = 2^{12} \rightarrow 12$ addresses

22. (c) Calculate by the given formula ($E_{\text{MAT}} = p \times s + (1-p) \times m$)

where p = page fault rate, s = page fault service time, m = memory access time.

23. (a) Calculate by the given formula:

$$\text{Average memory access time} = \frac{[(\%\text{ of Page miss} \times \text{Page fault service time}) + (\%\text{ of Page hit} \times \text{Memory access time})]}{100}$$

24. (d)

1	7	7	1	6	3	1	3	2	4	9	1
1	1	1	1	1	1	1	1	1	1	1	1
	4	4	4	4	4	4	4	4	4	9	9
		7	7	6	3	3	3	2	2	2	2
F	F	F	H	F	F	H	H	F	H	F	H

25. (a)

1	4	7	1	6	3	1	3	2	4	9	1
1	1	1	1	1	1	1	1	1	4	4	4
	4	4	4	6	6	6	6	2	2	2	1
		7	7	7	3	3	3	3	3	9	9
F	F	F	H	F	F	H	H	F	F	F	H

26. (d)

String	**2**	**0**	**1**	**2**	**4**	**0**	**5**	**1**	**4**	**6**	**4**	**2**	**1**	**3**	**0**
Frame 1	**2**	2	2	0	0	1	2	4	0	5	5	1	6	4	2
Frame 2		**0**	0	1	1	2	4	0	5	1	1	6	4	2	1
Frame 3			**1**	2	2	4	0	5	1	4	6	4	2	1	3
Frame 4					**4**	0	**5**	**1**	4	**6**	4	**2**	1	**3**	**0**

Bolds are representing misses. So, total 10 misses.

27. (d)

String	**2**	**0**	**1**	**2**	**4**	**0**	**5**	**1**	**4**	**6**	**4**	**2**	**1**	**4**	**0**
Frame 1	2	2	2	0	1	2	4	0	5	1	1	6	4	2	1
Frame 2		0	0	1	2	4	0	5	1	4	6	4	2	1	4
Frame 3			**1**	2	**4**	**0**	**5**	**1**	**3**	**6**	4	**2**	**1**	**4**	**0**

Bolds are representing misses. So 13 misses. So difference is $13 - 10 = 3$.

28. (b) Nearest cylinder next approach gives the better results.

29. (c) Number of disk block possible in one disk block

$$= \left(\frac{\text{DBA size}}{\text{DBA}}\right)$$

Each process takes total of 10 min for I/O + 10 min for CPU, that is, 20 min
Total time for complete = 20 min + 20 min
= 40 mins

31. (b) Probability p for which time is spent in I/O
$p = 1/2$ or 50% I/O
CPU utilization $= 1 - p^2 = 0.75$
$T \times 0.75 = 20$

32. (b) The use of multilevel page table reduces the size of the page table needed to implement the virtual address space of a process.

33. (b) Total number of process is 2^n, but we subtract the main process, therefore total number of children = (2^{n-1})

34. (c) Number of disk block pointer that will fit in one block = 210/4 = 256

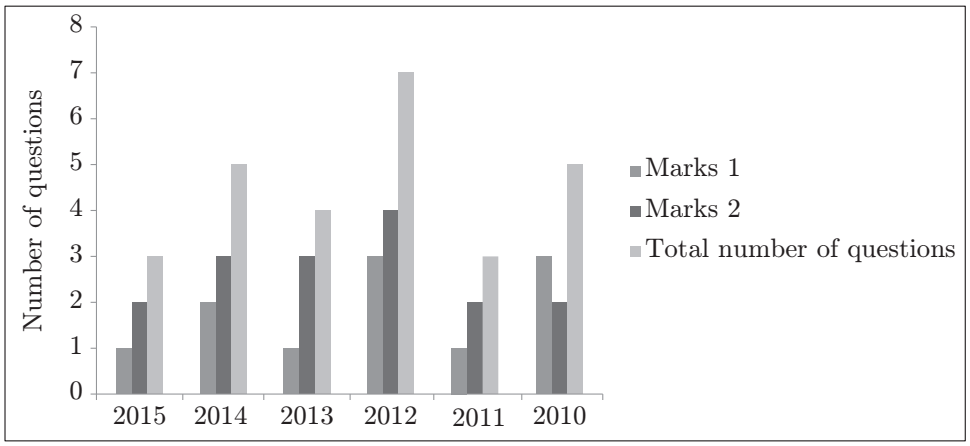
$$\text{Maximum file size} = 256 \times 256 \times 256 \times 1 \text{ KB} = 2^{34} \text{ bytes}$$

35. (a)

(a) Critical region: Non-sharable region of code.
(b) In concurrent programming, mutual exclusion is achieved by semaphore.
(c) Hoare's monitor: Monitors used for synchronization were invented by C. A. R. Hoare.
(d) condition fordeadlockto occur: Hold and wait, circular wait, mutual exclusion, no pre-emption

UNIT VIII: DATABASES

LAST SIX YEARS' GATE ANALYSIS



Concepts on which questions were asked in the previous six years

Year	Concepts
2015	SQL, ER Diagram, Disk, Functional Dependency, Transactions, B+ Trees, Transaction Properties, Transaction Recovery
2014	Relational algebra, SQL query, Serializability, Attributes of relational schema
2013	Indexing, Relational schemas, Normalizations, Functional dependencies
2012	ER and relational models, SQL query, Normalizations, Relational algebra
2011	SQL query, Indexing
2010	B+ tress, Relational schema, SQL query, Concurrency control protocols, Functional dependencies

CHAPTER 8

DATABASES

Syllabus: ER model, relational model (relational algebra, tuple calculus), database design (integrity constraints, normal forms), query languages (SQL), file structures (sequential files, indexing, B and B+ trees), transactions and concurrency control.

8.1 INTRODUCTION

Database is a collection of organized information so that it can easily be searched, retrieved, managed and updated. A database management system (DBMS) is a set of programs designed to manage a database. It enables users to store, retrieve and modify information in a database with utmost efficiency along with security features. DBMS is applicable to various day-to-day fields such as transactions in banking; airline/railway/hotel reservations; maintenance of student information in schools/universities; online retails; marketing and sales etc. It also allows its users to create their own databases. Different types of DBMS are available such as hierarchical, network, relational and object oriented.

8.1.1 Traditional File Processing Approach

File processing approach is generally more accurate and faster than the manual database system. Each user is responsible for the defining and implementation of the files required for the specific application. The implementation of required files sometimes creates redundancy of data, for example, one user keeps records for the savings account of the customer and another user may create the loan account of the same customer. This causes the duplicity of the records of the same customer. So, this practice is not feasible for real time applications. There is a need of centralized management of data. The data is created once and then accessed by different users. The data should be shared for different transactions. It should be self-describing in nature, which means the

database system contains not only data but it describes the description of the database structure.

8.1.2 Database Management System

A database is a collection of related data. Data is a collection of raw facts or figures, processed to form information. Database management system is a collection of programs for the creation and maintenance of database. It is an efficient and reliable approach to retrieve data for many users. It provides various functions such as:

1. **Redundancy control:** It provides redundancy by removing duplicity of data by following rules of normalization.
2. **Data independence:** It provides independence to application programs from details of data representation and storage. It also provides an abstract view of the data to insulate application code from such details.
3. **Data integrity:** It promotes and enforces some integrity rules for reducing data redundancy and increasing data consistency.
4. **Concurrency control:** It supports sharing of data, so, it has to provide an approach for managing concurrent access of the database. Hence, preserving the inconsistent state and integrity of the data.
5. **Transaction management:** It provides an approach to ensure that either all the updates for a given transaction will execute or that none of them would execute.
6. **Backup and recovery:** It provides mechanisms for backing up data periodically and recovering from different types of failures, thus, preventing loss of data.
7. **Non-Procedural query language:** It provides with query language for retrieval and manipulation of data.
8. **Security:** It protects unauthorized access in the database. It ensures the access to authorized users.

8.2 COMPONENTS OF DATABASE SYSTEMS

DBMS consists of several components, namely software, hardware, data, procedures and data access language. These components are responsible for the definition, collection, management and use of data within the environment. Figure 8.1 shows the components of database system. The description of each component is as follows:

1. **Software:** It is the collection of programs used by the computers within the database system. It is used to handle, control and manage the database. It includes the following software:
   - Operating system software like Microsoft Windows, Linux OS, Mac OS.
   - DBMS software such as Oracle 8I, MySQL, Access (Jet, MSDE), SQL Server etc.
   - Network softwares are used for sharing share the data of database among multiple users.
   - Application programs are developed like C++, VB, dotnet etc. are used to access database in dbms. These are used to access and manipulate the data in the database.
2. **Hardware:** It consists of all system's physical devices such as computers, storage devices, I/O channels, electromechanical devices etc. It also includes peripherals, such as, keyboard, mouse, modems, printers, etc.
3. **Data:** It is the collection of facts. The database contains the data and the metadata.
4. **Procedures:** There are the instructions and rules to design and use the database system. These includes the following:
   - Steps for the installation of DBMS
   - Steps to use the DBMS or application program
   - Steps for the backup of DBMS
   - Steps to change the structure of DBMS
   - Steps for the generation of reports.
5. **Data access language:** The users can use it to access the data to and from the database. The function of data access language is the entry of new data, manipulation of the existing data and the retrieval of the existing data in the database. The most popular database access language is SQL (Structured Query Language). Users can perform these functions with the help of commands. The role of administrator is to access, to create and to maintain the database.
6. **People:** Persons involved to access, to create and to maintain the database are called users. These are of various types according to the role performed by them (Fig. 8.1). These are as follows:
   - **System Administrator:** The role of system administrator is to supervise the general operations of DBMS.
   - **Database Administrator:** The role of database administrator (DBA) is to manage the DBMS.
   - **Database Designer:** The role of database designer is to design the structure of the database.
   - **Application Programmer:** The role of application programmer is to create the data entry forms, reports and procedures.
   - **End User:** The role of end user is to use the application programs by entering new data and manipulating and accessing existing data.

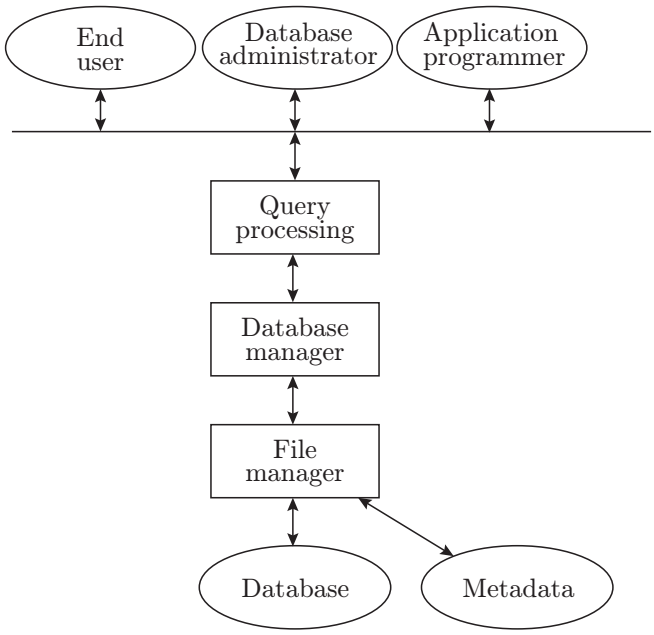


Figure 8.1 Components of database systems.

8.3 DBMS ARCHITECTURE

It is an approach to outlook the database by users. It is means for the representation of data in an understandable way to the users. DBMS architecture can be used to divide the whole system to related and independent modules. It can be of 1-tier, 2-tier 3-tier or n-tier.

8.3.1 3-Tier Architecture

DBMS can be most widely used as 3-tier architecture. In this architecture, the database is divided into three tiers depending upon the kind of users (see Fig. 8.2).

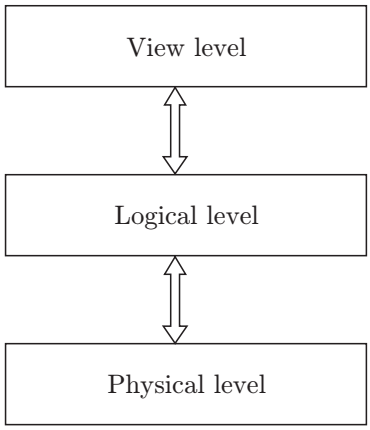


Figure 8.2 The 3-tier architecture of DBMS.

1. **Internal schema or physical level:** It is also called Database Tier. Database exists in this tier. It also includes query processing languages, all relations and their corresponding constraints. It describes the physical storage structure of the database.
2. **Conceptual schema or logical level:** It is also called Application Tier. Server and program exists in this tier. It also describes the structure of the database.
3. **External schema or view level:** It is also called Presentation Tier. End user exists in this tier. An end user is capable of multiple views of database. It also includes all views generated by applications.

8.3.2 Data Independence

Data independence is defined, as the change at one level does not affect the higher level. It is of two types:

1. **Logical data independence:** It is defined, as the change in conceptual schema does not affect the external schema. For example, if the format of the table changes, the data lies in the table should not be changed.
2. **Physical data independence:** It is defined, as the change in internal schema does not affect the conceptual schema. Thus, does not affect the external schema. For example, if the storage system has been changed, then it does not affect the logical structure of the database.

8.4 DATA MODELS

Data model is a collection of tools, which describes data, relationships, constraints and semantics. It gives the logical structure of the database. It describes the relationship of data in the database. Data models are of various types:

1. **Relational Model:** It is a collection of relations (tables). In relational model, each table is stored as a separate file.
2. **Entity–Relationship data model:** This model is based on the notion of real-world entities and relationship among them.
3. **Object-Based data model:** It defines the database as objects, its properties and its operations. In this model, objects with the similar structures comprise a class. The classes are organized into hierarchies. The operations on these classes are performed through methods.
4. **Semi-Structured data model:** It is also known as XML model. This model is used to exchange data over the web. It uses hierarchical tree structures. In this model, data can be represented as elements by using tags.

5. **Network model:** In this model, data is represented as record types. The data in this model has many-to-many relationship.
6. **Hierarchical model:** In this model, the data is represented as a hierarchical tree structure.

8.4.1 Relational Model

The database in relational model is represented as a collection of relations (tables). A relation is a kind of set. It is also a subset of a Cartesian product of an unordered set of ordered tuples. Relational model was proposed by E. F. Codd, which stores data in a tabular form. It consists of a table where rows represent records and columns represent the attributes. It has various terminologies as follows:

1. **Tuple:** It represents a single row of a table, which contains a single record for that relation.
2. **Relation instance:** It represents a finite set of tuples in the relational database system.
3. **Relation schema:** It represents the relation name, that is, table name, attributes and their names.
4. **Relation key:** It represents the unique key for the relation or table. Each row has one or more attributes, which can identify the row in the table uniquely.
5. **Attribute domain:** It represents the predefined value scope of each attribute.

8.4.1.1 Constraints in Relational Model

Constraints are the restrictions that one wishes to apply on database. The following constraints are applied on relational model.

1. **Key constraints:** Each relation has at least one minimal subset of attributes, which can identify a tuple uniquely.
   - No two tuples have identical value for key attributes.
   - Key attribute does not have NULL value.
2. **Domain constraints:** Attributes have specific domain values in real world. For example, value of age can only be positive.
3. **Referential integrity constraints:** If a relation refers to a key attribute of a different relation then that key element must exist.

8.4.1.2 Relational Algebra

It is a procedural query language. It takes instances of relations as input and returns instances of relations as output. Operators are used to perform queries.

Table 8.1 | Notations of fundamental operations of relational algebra

Fundamental Operation	Symbol
Select	σ
Project	π
Union	$\cup$
Intersection	$\cap$
Set difference	$-$
Cartesian product	$\times$
Rename	ρ
Natural join	$\bowtie$

Fundamental operations of relational algebra are as follows (see Table 8.1):

1. **Select:** It is used to select rows from a relation. It is denoted by σ.

   Syntax of select $\sigma_p(r)$, r is relation and p is prepositional logic.

   p uses connectors and operators $\wedge, \vee, =, \neq, <, >, \leq, \geq$.

   For example, $\sigma_{\text{empname}} = \text{"John"}$ (emp).
2. **Project:** It is used to project columns in a relation. It is denoted by π. The duplicate tuples are automatically eliminated.

   Syntax of projects $\pi_A(r)$, r is relation and A is the attribute name in a relation.

   For example, $\pi_{\text{empname, sal}}$ (emp)
3. **Union:** It returns a relation instance, which contains all tuples occurring in the first relation or in the second relation. It is denoted as $R \cup S$, where R and S are two relations. The duplicate tuples are automatically eliminated.

   This operation is valid for the following:
   - Both relations must have the same number of attributes.
   - Attribute domains must be compatible.
4. **Intersection:** It returns a relation instance, which contains all tuples occurring in both relations. It is denoted as $R \cap S$, where R and S are two relations.
5. **Set difference:** It returns a relation instance, which contains all tuples that occur in the first relation but not in the second relation. It is denoted as $R - S$, where R and S are two relations.
6. **Cartesian product:** It returns a relation instance, which contains all the fields of the first relation followed by all the fields of the second relation. It is denoted as $R \times S$, where R and S are two relations.

7. **Rename:** It returns a relation but without any name. It is used to rename the output relation. It is denoted as ρ.
8. **Joins:** It returns combined information from two or more relations.
   - *Condition joins:* It accepts a join condition c and a pair of relation instances as arguments, and returns a relation instance. It is denoted as $\sigma_c(R \times S)$.
   - *uijoin:* It is a special case of condition joins where the condition c contains equalities.
   - *Natural join:* It is a Cartesian product of two relations. It is denoted by $\bowtie$.

8.4.1.3 Tuple Calculus

It is a non-procedural query language. In this, number of tuple variables is specified. It is represented as $\{t \mid \text{Condition}\}$, wherte t is a tuple variable and Condition is a conditional expression.

Preliminaries of tuple calculus are as follows:

1. Constants
2. Predicates
3. Boolean and, or, not
4. $\exists$ there exists
5. $\forall$ for all

8.4.2 ER Model

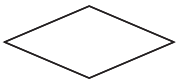
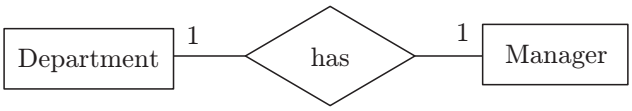
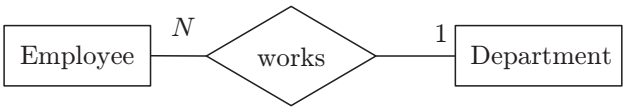
ER model represents the conceptual view of a database. It describes the relation of data to each other (Table 8.2). It was developed by Peter Chen in 1976. It views real-world data as systems of entities and relationships. ER model has three basic elements: entity, attribute and relationship. These are discussed in the following sections.

Table 8.2 | Entity-relationship objects

Element	Symbol	Example
Entity		Student, Employee
Weak entity		Order — Order item
Attribute		Name
Simple attribute		Order
Composite attribute		First_name, Last_name → Name
Derived attribute		Date_of_birth - - - Age
Single-valued attribute		Roll_number
Multi-value attribute		Phone_number

(*Continued*)

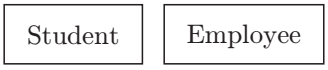
Table 8.2 | Continued

Element	Symbol	Example
Relationship		
One-to-One relationship		
One-to-Many relationship		
Many-to-One relationship		
Many-to-Many relationship		

8.4.2.1 Entity

Entities represent the real-world things. These are data objects which maintain different relationships with each other, for example, Employee, Department, etc. These are represented by means of rectangles.

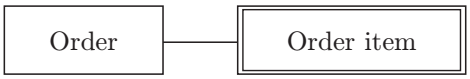
For example,



Entity set is a collection of similar types of entities. For example, all Employees set may contain all employees of all departments.

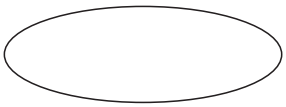
Weak entity depends on the existence of another entity. A weak entity cannot be identified by its own attributes. It uses a foreign key combined with its attributes to form the primary key. It is represented by means of double rectangles.

For example, Details of Employee's spouse, order item, etc.

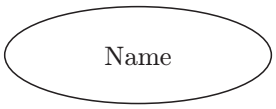


8.4.2.2 Attributes

Attribute is a property, trait or characteristic of an entity, relationship or another attribute. All attributes have values. A domain or range of values can be assigned to attributes. For example, name, class, age are attributes of the entity student. These are represented by ovals.



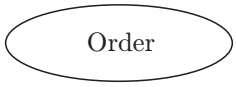
For example,



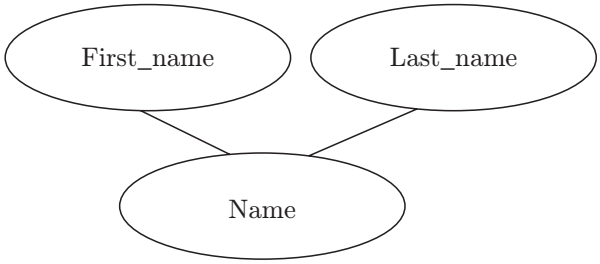
There are different types of attributes, listed as follows:

1. **Simple attribute:** Simple attributes contains atomic values. Atomic values cannot be divided

into sub parts. Examples are mobile number, roll number.

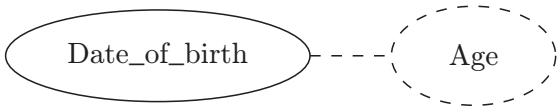


2. **Composite attribute:** Composite attributes are composition of many simple attributes. For example address can be divided into house number, street number, locality and city.

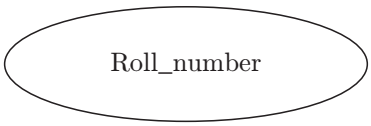


3. **Derived attribute:** Derived attributes are those whose value is derived from some other attribute in the database. For example, age of person can be calculated from date of birth. Dotted oval is used to represent derived attributes.

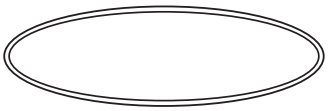
For example,



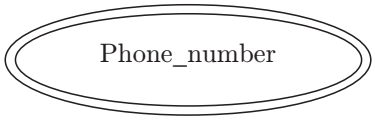
4. **Single-valued attribute:** Single valued attributes contain only one value for that attribute. For example age for person, blood group.



5. **Multi-value attribute:** In this, an attribute may contain more than one value. Multiple values are represented by double ovals.

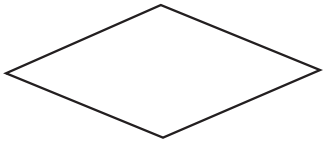


For example contact number, email ids.



8.4.2.3 Relationships

It represents the association among entities in a specified way. For example, employee entity has relation works at with department entity. Relationships are represented by diamond-shaped boxes.

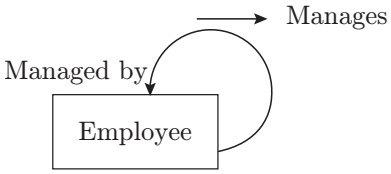


Some basic terminologies related to relationship are given below.

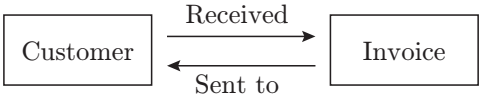
As we have discussed, relations are the core components in RDBMS, these relations are defined by two major characteristics—relationship set and the degree of relationship, defined in the following text.

1. **Relationship set:** Relationship of similar type is called relationship set. It has attributes. These attributes are called descriptive attributes.
2. **Degree of relationship:** It defines the number of participating entities in a relationship. They are of the following types:

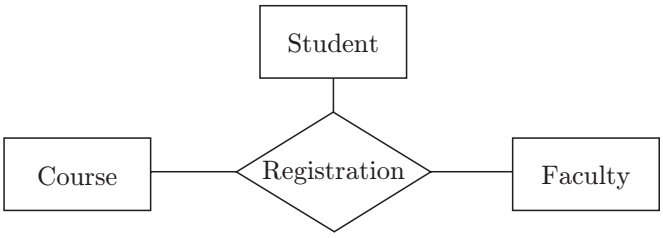
   - *Unary relationship (Degree 1):* One entity participates. For example,



   - *Binary relationship (Degree 2):* Two entities participate. For example,

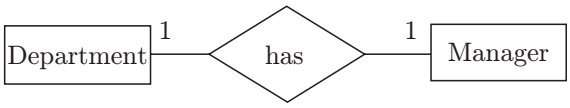


   - *Ternary relationship (Degree 3):* Three entities participate. For example,



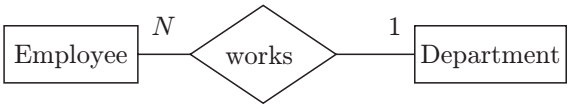
- *n-ary relationship (Degree n): n* entities participate.

3. **Cardinality:** It defines the number of instances of an entity, which can be associated to the number of instances of other entity via relationship set.
   - *One to one:* One instance of an entity is associated with at most one instance of another entity with the relationship. It is represented as '1-1'. For example,



   - *One to many:* One instance of an entity is associated with more than one instance of another entity with the relationship. It is represented as '1-*N*'. For example,

   - *Many to one:* More than one instance of an entity is associated with another entity with the relationship. It is represented as '*N*-1'. For example,



   - *Many to many:* More than one instance of an entity is associated with more than one instance of another entity with the relationship. It is represented as '*N*-*N*'. For example,

4. **Generalization:** It is a collection of entity sets, having similar characteristics, brought together into one generalized entity. For example, salaried and contract employees are generalized as employee.
5. **Specialization:** It is the process of identifying subsets of an entity set. It is a reverse process of generalization. For example, employees are specialized as salaried and contract as shown in Fig 8.3.

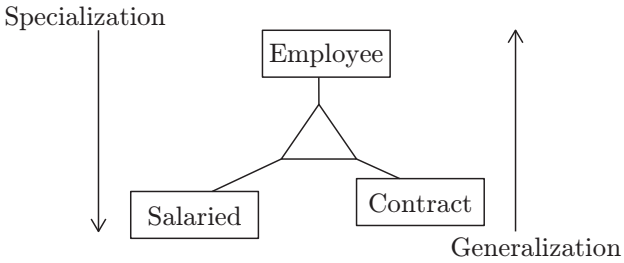


Figure 8.3 | Specialization and Generalization.

6. **Aggregation:** It allows a relationship set participate in another relationship set (see Fig. 8.4).

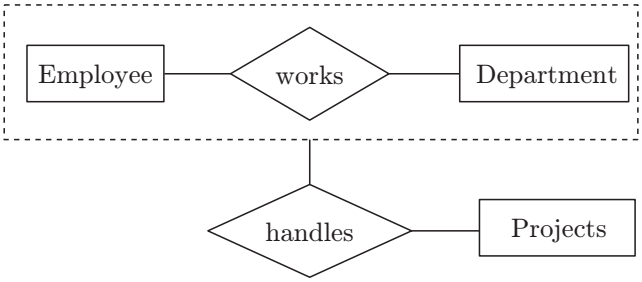


Figure 8.4 | Aggregation.

8.5 DATABASE DESIGN

The design of a database consists of the following steps:

1. Identifying entities
2. Identifying relationships
3. Identifying attributes
4. Presenting entities and relationships: Entity relationship diagram (ERD)
5. Assigning keys
6. Defining the attribute's data type
7. Normalization

8.5.1 Integrity Constraints

Integrity constrains are applied to maintain the consistency in a database. This helps in providing the unique answer to a given query on the database. For example, if the answer to particular query is 'x' then it should be 'x' if such a query is carried out again (without adding/deleting/modifying) on the same table.

1. **Domain integrity:** It defines a valid set of values for an attribute, for example, length or size, data type, etc.

2. **Entity integrity constraint:** It defines that the primary keys cannot be null. There must be a proper value in the primary key field.
3. **Referential integrity constraint:** It is specified between two tables. It is used to maintain the consistency among rows between the two tables.
4. **Foreign key integrity constraint:** There are two types of foreign key integrity constraints:
   - *Cascade update related fields:* Whenever the primary key of a row in the primary table is changed, the foreign key values are updated in the matching rows in the related table.
   - *Cascade delete related rows:* Whenever a row in the primary table has been deleted, the matching rows are automatically deleted in the related table.

8.5.2 Normal Forms

Normalization is a technique of organizing the database tables, such that they have minimum redundancy. Following are the normal forms that are to be achieved for the process of normalization. The underlying concept is that, if we say a table to be satisfying an 'x' level normal form, then it is understood that it has also satisfied the 'x-1' level normal form.

1. **First normal form:** It defines that all the attributes in a relation must have atomic domains.
2. **Second normal form:** It defines that every non-prime attribute should be fully functionally dependent on prime key attribute. A prime attribute is a part of prime key. The relation must be in the First normal form.
3. **Third normal form:** It defines that no non-prime attribute is transitively dependent on prime key attribute. The relation must be in the Second normal form.
4. **Boyce–Codd normal form:** It defines that for any non-trivial FD, $X \rightarrow A$, then X must be a superkey. It is an extension of the Third normal form.
5. **Fourth normal form:** It defines that for every multivalued dependency $X \rightarrow\rightarrow Y$ that holds over R, one of the following statements is true: (a) Y is the subset of X and (b) X is a superkey. Also, sometimes called, Multi-valued Dependency Normal Form (MDNF).
6. **Fifth normal form:** It denies that for every join dependency $\bowtie \{R_1, \ldots, R_n\}$ that holds over R, one of the following statements is true: (a) $R_i = R$ and (b) the join dependency is implied by the set of those FDs over R in which the left side is a key. Also, sometimes called, Project-Join Normal Form (PJNF).

8.5.3 Attribute Closure

Set of all attributes functionally determined by X is called closure of X. Closure set of X is denoted by X^+.

Problem 8.1: Functional dependencies $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$ is given, find closure of A, B, C and D.

Solution:

Closure of $A = A^+ = (A, B, C, D)$

Closure of $B = B^+ = (B, C, D)$

Closure of $C = C^+ = (C, D)$

Closure of $D = D^+ = (D)$

8.5.4 Key

It is a minimum set of attributes used to differentiate all the tuples of the table.

8.5.4.1 Superkey

It is a set of attributes that uniquely identifies each record in a table. It is a superset of candidate key. In other words, let R be the schema and X be the set of attributes over R. If closure of X (X^+) determines all the attributes of R, then X is called superkey.

Problem 8.2: Consider a schema $R(ABCDE)$ and FDs $\{AB \rightarrow C, C \rightarrow D, B \rightarrow E\}$. Find superkey.

Solution:

Find closure set of (A, B, C, AB).

Closure of $A = A^+ = \{A\}$

Closure of $B = B^+ = \{B, E\}$

Closure of $C = C^+ = \{C, D\}$

Closure of $AB = AB^+ = \{A, B, C, D, E\}$

So, AB is the superkey.

8.5.4.2 Candidate Key

It is an attribute or set of attribute that can act as a primary key of a table that uniquely identifies each record

in a table. Every candidate key is a superkey but not vice versa.

In other words, candidate key is the minimal superkey. If X is a superkey and none of the proper subset of X is a superkey, then X is called the minimal superkey or candidate key.

Problem 8.3: Consider a schema $R(ABCDE)$ and FDs $\{AB \rightarrow C,\ C \rightarrow D,\ B \rightarrow EA\}$. Find the superkey.

Solution:

Find closure set of $(A,\ B,\ C,\ AB)$

Closure of $A = A^+ = \{A\}$

Closure of $B = B^+ = \{B,\ E,\ A,\ C,\ D\}$

Closure of $C = C^+ = \{C,\ D\}$

Closure of $AB = AB^+ = \{A,\ B,\ C,\ D,\ E\}$

So, AB and B are superkeys. But only B is the candidate key.

8.5.4.3 Primary Key

It is one or more data attributes that uniquely identify an entity. It does not allow null values.

1. **Alternate key:** The candidate key, which is not selected as a primary key.
2. **Composite key:** It consists of two or more attributes.
3. **Foreign key:** It is an entity that is the reference to the primary key of another entity.

8.5.5 Decomposition

It is required to eliminate redundancy from the schema. If a relational schema R has redundancy in the data, then decompose R into two R_1 and R_2 schema. There are two properties, which should be maintained when we perform decomposition, it should be a lossless join as well as dependency preserving.

8.5.5.1 Lossless Join

Let R be a relation schema and let F be a set of functional dependency (FD) over R. R is decomposed into R_1 and R_2. R_1 and R_2 are called lossless-join decomposition if $R = R_1 \bowtie R_2$ or if we can recover original relation from the decomposed relation.

(a) *Algorithm for finding decomposition is lossless:*

Step 1: Union of all decomposed sub-relation should be equal to relation R.

$$R_1 \cup R_2 \cup R_3 \cup \ldots \cup R_n = R$$

Step 2: Any two sub-relations R_i and R_j can be merged into R_{ij} with $R_1 \cup R_2$ only if

i. $R_i \cap R_j \neq \varnothing$ (null)

ii. $R_i \cap R_j = X$ then closure of $X(X^+) \supseteq R_i$

or

$R_i \cap R_j = X$ then closure of $X(X^+) \supseteq R_j$

Step 3: Repeat step 2 until 'N' relations become one relation. If 'N' relations become single relation, then composition is called lossless, otherwise not.

Problem 8.4: Consider a schema $R\ (ABCDEFGHIJ)$ and functional dependencies $\{\text{FDs} = (AB \rightarrow C,\ A \rightarrow D,\ B \rightarrow F,\ F \rightarrow GH,\ D \rightarrow IJ)\}$ and decompositions

(a) $\{D = (ABCDE,\ BFGH,\ DIJ)\}$
(b) $\{D = (ABCD,\ DE,\ BF,\ FGH,\ DIJ)\}$

Check whether the decomposition is lossless or not.

Solution:

(a) Given

$$R_1 = (ABCDE)$$
$$R_2 = (BFGH)$$
$$R_3 = (DIJ)$$

Apply algorithm for lossless decomposition:

Step 1:

$$R_1 \cup R_2 \cup R_3 = \{(ABCDE) \cup (BFGH) \cup (DIJ)\}$$
$$= (ABCDEFGHIJ) = R$$

Step 1 satisfies the given condition, so it is true.

Step 2:

For R_1 and R_2:

$$R_1 \cap R_2 = (ABCDE) \cap (BFGH) = B$$

Find closure of $B = B^+ = \{B,\ F,\ G,\ H\}$.

Condition (ii) is satisfied, so R_1 and R_2 can be merged together. After merging R_1 and R_2,

$$R_{12} = (ABCDEFGH)$$

Now merge R_{12} and R_3.

$$R_{12} \cap R_3 = (ABCDEFGH) \cap (DIJ) = D$$

Find closure of $D = D^+ = \{D, I, J\}$.

Condition (ii) is satisfied, so R_{12} and R_3 can be merged together. After merging R_{12} and R_3

$$R_{123} = (ABCDEFGHIJ)$$

So, $R_{123} = R$, therefore, it is lossless decomposition.

(b) Given that

$$R_1 = (ABCD)$$
$$R_2 = (DE)$$
$$R_3 = (BF)$$
$$R_4 = (FGH)$$
$$R_5 = (DIJ)$$

Apply algorithm for lossless decomposition.

Step 1:

$$R_1 \cup R_2 \cup R_3 \cup R_4 \cup R_5 = \{(ABCD) \cup (DE) \cup (BF) \cup (FGH) \cup (DIJ)\} = (ABCDEFGHIJ) = R$$

Step 1 satisfies the given condition, so it is true.

Step 2:

For R_1 and R_2

$$R_1 \cap R_2 = (ABCD) \cap (DE) = D$$

Find closure of $D = D^+ = \{D, I, J\}$.

Condition (ii) of step 2 does not satisfy, so R_1 and R_2 cannot be merged together.

For R_1 and R_3

$$R_1 \cap R_3 = (ABCD) \cap (BF) = B$$

Find closure of $B = B^+ = \{B, F, G, H\}$.

Condition (ii) of step 2 is satisfied, so R_1 and R_3 can be merged together. After merging R_1 and R_3

$$R_{13} = (ABCDF)$$

For R_{13} and R_4

$$R_{13} \cap R_4 = (ABCDF) \cap (FGH) = F$$

Find closure of $F = F^+ = \{F, G, H\}$

Condition (ii) of step 2 is satisfied, so R_{13} and R_4 can be merged together. After merging R_{13} and R_4

$$R_{134} = (ABCDFGH)$$

For R_{134} and R_5

$$R_{134} \cap R_5 = (ABCDFGH) \cap (DIJ) = D$$

Find closure of $D = D^+ = \{D, I, J\}$

Condition (ii) of step 2 is satisfied, so R_{134} and R_5 can be merged together. After merging R_{134} and R_5

$$R_{1345} = (ABCDFGHIJ)$$

For R_{1345} and R_2

$$R_{1345} \cap R_2 = (ABCDFGHIJ) \cap (DE) = D$$

Find closure of $D = D^+ = \{D, I, J\}$.

Condition (ii) of step 2 does not satisfy, so R_{1345} and R_2 cannot be merged together.

So, finally we left with two decompositions which cannot be merged into a single relation. So condition given in step 3 does not satisfy, therefore, it is not lossless decomposition.

8.5.5.2 Dependency Preserving

All the dependency should be preserved after the decomposition of schema R.

Let R be the relational schema with FD set F decomposed into $R_1, R_2, R_3, \ldots, R_n$ with FD sets $F_1, F_2, F_3, \ldots, F_n$, respectively. The decomposition is said to be dependency preserved if

$$F_1 \cup F_2 \cup F_3 \cup \cdots \cup F_n = F$$

And composition is called non-dependency preserved if

$$F_1 \cup F_2 \cup F_3 \cup \cdots \cup F_n \subset F$$

(a) *Algorithm to check dependency is preserved or not:*

Step 1: Find all the FDs of sub-relations.

Step 2: Check that all the FDs of relation F is covered by FDs of sub-relation in any form directly or indirectly.

Problem 8.5: Consider a schema $R(ABCD)$ and FDs set $F = (A \to B, B \to C, C \to D, D \to A)$ and decompositions $D = (AB, BC, CD)$. Check decomposition is dependency preserving or not.

Solution:

Step 1: Find all the FDs of sub-relations.

	$R_1 = (AB)$	$R_2 = (BC)$	$R_3 = (CD)$
Direct Dependency	$A \to B$	$B \to C$	$C \to D$
Indirect Dependency	$B \to A$	$C \to B$	$D \to C$

To calculate indirect dependency, use closure property. In relation R_1, we have indirect dependency $B \rightarrow A$. We have to first find closure of all attributes in that decomposition (like in R_1 we have A and B) by using FD set F and then check that we are getting attribute A in closure of B. So closure of $B = B^+ = ABCD$. So we can write that indirectly as we have three dependencies $\{B \rightarrow A, B \rightarrow C$ and $B \rightarrow D\}$ but only $B \rightarrow A$ is valid for R_1 as we have only two attributes (A and B) in relation R_1.

Step 2: Check that all the FDs of relation F is covered by FDs of sub-relation in any form directly or indirectly.

(a) $A \rightarrow B$ (Directly covered by sub-relation FD)
(b) $B \rightarrow C$ (Directly covered by sub-relation FD)
(c) $C \rightarrow D$ (Directly covered by sub-relation FD)
(d) $D \rightarrow A$ (Indirectly covered by sub-relation FD by using $\{D \rightarrow C$ then $C \rightarrow B$ and then $B \rightarrow A\}$).

All the dependency is preserved, so it is dependency-preserved decomposition.

8.5.5.3 Relation between Two Functional Dependency (FD) Sets

Let F and G be two FD sets:

1. Set F and G are equal if and only if closure of $F(F^+)$ = Closure of $G(G^+)$
2. Set F and G are equal only if both of the below conditions satisfy:
   - F covers G: All FD in G is logically implied by F.
   - G covers F: All FD in F is logically implied by G.
3. If all FD in G is logically implied by F but all FD in F is not logically implied by G then $G \subset F$.

Problem 8.6: Let there be two FD sets F and G, given as follows:

$F = \{A \rightarrow B, B \rightarrow C\}$

$G = \{A \rightarrow B, AB \rightarrow C, B \rightarrow C, A \rightarrow C\}$

Find the relation between both FDs.

Solution:

(a) F covers G:

Functional Dependency	Covered by F
$A \rightarrow B$	Direct covered
$AB \rightarrow C$	Indirect covered ($AB^+ = ABC$), so AB can determine C.
$B \rightarrow C$	Direct covered
$A \rightarrow C$	Indirect covered ($A^+ = ABC$), so A can determine B and C.

(b) G covers F

Functional Dependency	Covered by G
$A \rightarrow B$	Direct covered
$B \rightarrow C$	Direct covered

In our example, F covers G and vice versa. So $F = G$.

8.6 QUERY LANGUAGES (SQL)

SQL stands for Structured Query Languages, which is a standard computer language for relational database management and data manipulation. It is used to query, insert, update and modify data in the table. Some common RDBMS that use SQL are Oracle, Microsoft SQL Server, Access, Ingres, Sybase, etc. Raymond Boyce and Donald Chamberlin developed it in the early 1970s at IBM, but commercially released by Relational Software Inc. (now, Oracle Corporation) in 1979. Looking into the history it was the software named- VULCAN, that was procured by Ashton Tate and then by FoxPro and later was purchased by Microsoft. Other popular softwares were/are – Clipper and Gupta Technologies.

8.6.1 SQL Commands

The SQL commands are used to interact with relational databases. These commands can be classified into the following groups (see Tables 8.3 and 8.4):

1. **Data Definition Language (DDL):** It contains metadata, that is, data about data. All the integrity constraints and data base schemas are defined through DDL. These commands are used to create, modify and delete database objects. CREATE, ALTER, TRUNCATE and DROP are part of DDL. TRUNCATE command is used to delete complete data from an existing table, while DROP command is used to remove a table definition and all data, indexes, triggers and constraints for a table.
   - Syntax of create command

     ```
     CREATE TABLE table_name( column1
     datatype, column2 datatype, column3
     ```

```
datatype, ..... columnN datatype,
PRIMARY KEY( one or more columns ) );
```

For example, `create table instructor (INST_ID char(5), name varchar(20), dept_name varchar(20), salary numeric(8,2));`

- Syntax of alter command

```
ALTER TABLE table_name ADD column_
name datatype;
ALTER TABLE table_name DROP COLUMN
column_name;
ALTER TABLE table_name MODIFY COLUMN
column_name datatype;
ALTER TABLE table_name ADD CONSTRAINT
Constraint UNI (column1, column2...);
ALTER TABLE table_name ADD CONSTRAINT
Constraint CHECK (CONDITION);
ALTER TABLE table_name ADD CONSTRAINT
PrimaryKey PRIMARY KEY (column1,
column2...);
```

For example, `ALTER TABLE CUSTOMERS ADD GRADE char(1);`

- Syntax of truncate command

```
TRUNCATE TABLE table_name;
```

For example, `TRUNCATE TABLE emp;`

- Syntax of drop command

```
DROP TABLE table_name;
```

For example, `DROP TABLE emp;`

Table 8.3 | SQL commands

Command	Description
CREATE	Creates a new table, a view of a table or other objects in the database
ALTER	Modifies an existing database (table)
DROP	Deletes a table, a view of a table or other objects in the database
SELECT	Retrieves data from one or more tables (database)
INSERT	Inserts new data record in the database
UPDATE	Modifies data in the database
DELETE	Deletes data from the database
GRANT	Gives a privilege to the user
REVOKE	Takes back privileges granted from the user

Table 8.4 | Clauses in SQL

Clause	Description
From	Equals to cross product
Where	Selects the tuples which satisfies the condition
Group by	Groups the table based on specified attribute
Having	Used to select groups based on condition

2. **Data Manipulation Language (DML):** DML is used to manipulate the data in database. It allows to insert, update and delete data items. SELECT, INSERT, DELETE and UPDATE are part of DML. Control statements BEGIN TRANSACTION, SAVEPOINT, COMMIT and ROLLBACK are also part of DML.

   - Syntax of select command

```
SELECT * FROM table_name;
SELECT column1, column2, columnN
FROM table_name;
```

For example, `SELECT empno, ename FROM emp;`

   - Syntax of insert command

```
INSERT INTO TABLE_NAME (column1,
column2, column3,...columnN)] VALUES
(value1, value2, value3,...valueN);
INSERT INTO TABLE_NAME VALUES
(value1,value2,value3,...valueN);
```

For example, `INSERT into emp (empno, ename, sal, dept) VALUES (100, ABC, 10000, Accounts);`

   - Syntax of delete command

```
DELETE FROM table_name WHERE [condi-
tion];
```

For example, `DELETE FROM emp WHERE empno = 8;`

   - Syntax of update command

```
UPDATE table_name SET column1 =
value1, column2 = value2....,
columnN = valueN WHERE [condition];
```

For example, `UPDATE mp SET sal = 12000 WHERE empid = 8;`

3. **Data Control Language (DCL):** To assign or revoke access rights data control language is used. GRANT and REVOKE are used for DCL.

- Syntax of grant command

```
GRANT privilege_name ON object_name
TO {user_name | PUBLIC | role_name}
[with GRANT option];
```

For example, `GRANT SELECT ON emp TO user1;`

- Syntax of revoke command

```
REVOKE privilege_name ON object_name
FROM {User_name | PUBLIC | Role_name};
```

For example, `REVOKE SELECT ON emp TO user1;`

8.7 FILE STRUCTURES

File structure mainly deals with how files are stored on the disk. Various file organisations are described below.

8.7.1 Sequential Files

It is a file organisation system where every file record contains an attribute to uniquely identify a particular record. Records are placed in a sequential order with some unique key.

8.7.2 Indexing

Indexing is a data structure mechanism to efficiently retrieve records from the database, for example, book index. It is defined based on its indexing attributes. Indexing is of three types:

1. **Primary index:** Indexing is based on ordering key field of file.
2. **Secondary index:** Indexing is based on non-ordering field of file.
3. **Clustering index:** Indexing is based on ordering non-key field of file.

Ordering field is the field on which the records of file are ordered. Ordered indexing is of two types: dense index and sparse index.

1. **Dense index:** For every search key value in the database, there is an index record. Index record has two parts: search key value and the pointer. The pointer is pointed to the actual record.
2. **Sparse index:** In this no index record is created for every search key.

8.7.3 B Tree

A B tree is a data structure that stores data in such a manner that search, insertions and deletions can be done in logarithmic time. B trees are a general form of binary trees where a node can have more than one child.The B-trees are efficient for those systems that read and write large blocks of data, that is, databases and file systems.

B trees are self-balancing trees. All the leaf nodes are at the same level. A B tree with order p has:

1. Root node may contain minimum 1 key
2. Minimum number of child $= \left(\frac{p}{2}\right) - 1$
3. Maximum number of children $= p$
4. Maximum keys $= p - 1$

The order of B-tree can be found as follows:

$$p \times P + (p-1)(K + P_r) \leq \text{Block size}$$

where p is order of the tree, P is the block pointer, P_r is the record pointer and K is the key pointer.

Problem 8.7: The order of B-tree index is the maximum number of children it can have. Suppose that a block pointer takes 6 bytes, the search field value takes 9 bytes, record pointer takes 7 bytes and the block size is 512 bytes. What is the order of the B tree?

Solution:

Given that block size = 512 B; record pointer (P_r) = 7 B; block pointer (P_B) = 6 B; key pointer (K) = 9 B. So, we have

$$p \times P + (p-1)(K + P_r) \leq \text{Block size}$$

$$6p + (p - 1)(9 + 7) = 512$$

$$p = 24$$

8.7.4 B+ Trees

It supports multilevel indexing. The leaf nodes of B+ tree represent actual data pointers. It ensures all leaf nodes are balanced, that is, at the same height. Leaf nodes are linked using link list.

B+ tree structure is such that B+ tree is of order n and it is fixed for every B+ tree.

The internal nodes contain at least $[n/2]$ pointers, except the root node and at most n pointers. Leaf nodes contain at least $[n/2]$ record pointers, at least $[n/2]$ key values, at most n record pointers, at most n key values, and every leaf node contains one block pointer P to point to next leaf node and forms a linked list.

Order of internal nodes can be calculated as:

$$p \times P_B + (p-1) \times k \leq \text{Block size}$$

Order of leaf nodes:

$$P_{\text{leaf}} \times [k + P_r] + P_B \leq \text{Block size}$$

where p is the order of internal nodes, P_{r} is record pointer, P_{B} is block pointer, k is key pointer, P_{leaf} is the order of leaf nodes.

Problem 8.8: The order of an internal node in a B+ tree index is the maximum number of children it can have. Suppose that a block pointer takes 6 bytes, the search field value takes 9 bytes, record pointer take 7 bytes and the block size is 512 bytes. What is the order of the internal node and leaf node?

Solution:

Given that block size = 512 B, record pointer (P_{r}) = 7 B, block pointer (P_{B}) = 6 B and key pointer (k) = 9 B.

Order of internal nodes:

$$p \times P_B + (p-1)\ k \leq \text{Block size}$$
$$6p + (p - 1)9 = 512$$
$$p = 34$$

Order of leaf nodes:

$$P_{\text{leaf}} \times [k + P_r] + P_B \leq \text{Block size}$$
$$P_{\text{leaf}}\ [9 + 7] + 6 = 512$$
$$P_{\text{leaf}} = 31$$

8.8 TRANSACTIONS AND CONCURRENCY CONTROL

Transactions are series of read and write operations on database. When many users access same database at the same time, some control mechanism is required by databases to stay in consistent state. Following sections will provide description about transactions and their control mechanism.

8.8.1 Transactions

It is a series of reads and writes of database objects. It maintains the integrity of a database while running multiple concurrent operations. Transactions have four properties that a DBMS must ensure to maintain data. These are known as ACID properties:

1. **Atomicity:** Either all actions are carried out or none (no partial transaction).
2. **Consistency:** After the execution of transaction, the database must be in a consistent state (no concurrent execution of other transactions).
3. **Isolation:** All the transactions are carried out and executed as the only transaction in the system (no transaction will affect the existence of any other transaction).
4. **Durability:** Persistence of data even if the system fails and restarts.

8.8.2 Schedule

A chronological execution sequence of transactions is called schedule. It is a list of actions reading, writing, aborting or committing from a set of transactions. A schedule can have many transactions. Schedule can be further divided into two types—serial schedule and concurrent schedule.

8.8.2.1 Serial Schedule

A schedule is called serial schedule if all transactions are executed in a non-interleaving manner. Let there are two transactions T_1 and T_2 schedule S then schedule S is called serial schedule if one transaction executes after another shown in Fig. 8.5.

T_1	T_2
Read (X)	
$X = X - Z$	
Write (X)	
Read (Y)	
$Y = Y + Z$	
Write (Y)	
	Read (X)
	$X = X + W$
	Write (X)

T_1	T_2
	Read (X)
	$X = X + W$
	Write (X)
Read (X)	
$X = X - Z$	
Write (X)	
Read (Y)	
$Y = Y + Z$	
Write (Y)	

Figure 8.5 | Serial schedule.

8.8.2.2 Concurrent Schedule

A schedule is called concurrent schedule if transactions are executed in an interleaving manner or simultaneous execution of two or more transactions. Let there be two transactions T_1 and T_2 in a schedule S, then schedule S is called concurrent schedule if both the transactions execute parallel. One of the scenarios is shown in Fig. 8.6.

T_1	T_2
Read (X)	
X = X − Z	
	Read (X)
	X = X + W
Write (X)	
	Write (X)
Read (Y)	
Y = Y + Z	
Write (Y)	

Figure 8.6 | Concurrent schedule.

8.8.2.3 Comparison between Serial and Concurrent Schedule

Table 8.5 shows the comparison between serial and concurrent schedule.

Table 8.5 | Serial schedule vs. concurrent schedule

Serial Schedule	Concurrent Schedule
All serial schedules are consistent schedules	All serial schedules are not consistent schedules
Less throughput	More throughput
Poor resource utilization	Good resource utilization
More response time	Less response time
For the given transactions, the number of serial schedule is very much less than the number of concurrent schedule	For the given transactions, the number of concurrent schedule is more than the number of serial schedule

8.8.2.4 Problems Occurring due to Concurrent Schedule

In concurrent schedule, more than one transaction is executed simultaneously; and due to this some problem arises with concurrent schedule given as follows:

1. **WR problem (Read after Write):** WR problem is also known as dirty read problem or uncommitted read problem. Let there be two transactions T_i and T_j of schedule S. If transaction T_j reads a data item which is written by T_i, but till that time transaction T_i is not committed, then WR problem can occur. If in transaction T_i rollback or failure occurs, then database will be inconsistent due to uncommitted read by T_j transaction (Fig. 8.7).

Transaction T_i	Transaction T_j
Read (A)	
A = A + X	
Write (A)	
	Dirty Read
	Read (A)
	Write (B)
	A = A + Y
Read (B)	
.....	
.....	
Failure or rollback	
	Write (A)

Figure 8.7 | Uncommitted read problem (WR problem).

Example: Let value of data item A = 1000, X = 100 and Y = 200, then transaction T_i will update the database with value 1100. Transaction T_j will read data item value as 1100. Transaction T_j will update database with value 1300. As transaction T_i fails, the database value should be 1200. Therefore, this problem is known as uncommitted read problem or dirty read problem.

2. **RW problem (Write after Read):** RW problem is also known incorrect summary problem or unrepeatable read problem (Fig. 8.8). Let there be two transactions T_i and T_j of schedule S. Transaction T_i read a data item and T_j also read similar data item. Transactions T_i and T_j both have write operation on that data item. If in both transactions, read operation occurs before commit of the other transaction and before writing back that data item in database by other transaction then RW problem will arise.

Transaction T_i	Transaction T_j
Read (A)	
If (A > 0)	
A = A + 1	
	Read (A)
	If (A > 0)
	A = A + 1
Write (A)	
Commit	Write (A)
	Commit

Figure 8.8 | Incorrect summary problem (RW problem)

Example: Let value of data item A = 100. Transaction T_i will read data item with value 100 and update database with value 101. Transaction T_j will read data item value as 100 because it will read before updating value of data item A by transaction T_i. Transaction T_j will update database with value 101. But data item 'A' value should be 102 because both the transaction have commit successfully. Therefore, this problem is known as unrepeatable read problem.

3. **WW Problem (Write after Write):** WW problem in concurrent schedule is also known as lost update problem (Fig. 8.9). Let there are two transactions T_i and T_j of schedule S. Transaction T_i reads a data item and updates it. Now, transaction T_j also writes similar data item with some other value and transaction T_j commits successfully. After commit of transaction T_j, transaction T_i will rollback. Thus, updated value of data item by transaction T_j will be lost.

Transaction T_i	Transaction T_j
Read (A)	
Write (A)	
	Write (A)
	Commit
Failure and Rollback	

Figure 8.9 Lost update problem (WW problem).

Example: Let the value of data item A=100. Transaction T_i read data item and update database with value 200. Let transaction T_j updated database with value 250 and commit successfully. After committing transaction T_j, transaction T_i rollback which set data item value to its initial value 100. So, this problem is known as lost update problem.

8.8.3 Classification of Schedule Based on Recoverability

8.8.3.1 Irrecoverable Schedule

Let there be two transactions T_i and T_j of schedule S. If transaction T_j reads a data item which is updated by transaction T_i and transaction T_j commits before the commit (or rollback) of transaction T_i, then the given schedule S is called irrecoverable schedule.

8.8.3.2 Recoverable Schedule

Let a schedule S have two transactions T_i and T_j. If transaction T_j reads a data item which is updated by transaction T_i and transaction T_j is not allowed to commit (or rollback) before the commit (or rollback) of transaction T_i, then the given schedule S is called recoverable schedule. Recoverable schedule may suffer from uncommitted read, lost update and incorrect summary problem.

8.8.3.3 Cascading Rollback Recoverable Schedule

Let there be four transactions (T_1, T_2, T_3, T_4) in a given schedule S. In schedule S, if rollback of a transaction (T_1) results in the rollback of the other transactions (T_2, T_3, T_4) (because of their dependency on each other), then this is called cascading rollback.

If a schedule is recoverable and has no cascading rollback, then it is called cascading rollback recoverable schedule. Incorrect summary and lost update problems may exist in cascading rollback recoverable schedule. Problems of no recoverability, Uncommitted Read problem (WR problem) and cascading rollback problem do not occur in such schedule.

8.8.3.4 Strict Recoverable Schedule

Let a schedule S have two transactions T_i and T_j. If S is called a strict recoverable schedule then it satisfies these two conditions:

1. Schedule S should be cascading rollback recoverable schedule.
2. If one transaction T_i writes a data item 'A', then the other transaction T_j is not allowed to write on that data item.

Only incorrect summary problem (RW problem) may arise in strict recoverable schedule. Irrecoverable, Uncommitted Read problem (WR problem), lost update and cascading rollback problems do not occur in such schedule.

8.8.3.5 Summary of Schedules Based on Recoverability

Schedule	Problem Exists	Problem Remove
Irrecoverable	All	None
Recoverable	Incorrect summary problem (RW), Uncommitted Read problem (WR problem), lost update (WW) and cascading rollback problem	Irrecoverable
Cascading rollback recoverable	Incorrect summary problem (RW) and lost update (WW).	Irrecoverable, Uncommitted Read problem (WR problem) and cascading rollback problem.
Strict recoverable	Incorrect summary problem (RW)	Irrecoverable, Uncommitted Read problem (WR problem), lost update (WW) and cascading rollback problem.

Problem 8.9: A concurrent schedule S has three transactions T_1, T_2, T_3. Transactions execute read/write operation in the following sequence:

$\text{Read}_1(\text{x})$, $\text{Read}_2(\text{z})$, $\text{Read}_3(\text{x})$, $\text{Read}_1(\text{z})$, $\text{Read}_2(\text{y})$, $\text{Read}_3(\text{y})$, $\text{Write}_1(\text{x})$, Commit_1, $\text{Write}_2(\text{z})$, $\text{Write}_3(\text{y})$, $\text{Write}_2(\text{y})$, Commit_3, Commit_2.

Find out the type of recoverable schedule?

Solution:

Transaction T_1	Transaction T_2	Transaction T_3
$\text{Read}_1(\text{x})$	-----	-----
-----	$\text{Read}_2(\text{z})$	-----
-----	-----	$\text{Read}_3(\text{x})$
$\text{Read}_1(\text{z})$	-----	-----
-----	$\text{Read}_2(\text{y})$	-----
-----	-----	$Read_3(\text{y})$
$\text{Write}_1(\text{x})$	-----	-----
Commit_1	-----	-----
-----	$\text{Write}_2(\text{z})$	-----
-----	-----	$\text{Write}_3(\text{y})$
-----	$\text{Write}_2(\text{y})$	-----
-----	-----	Commit_3
-----	Commit_2	-----

Step 1: Check for recoverable. By using the definition of recoverable schedule we can find that given schedule is recoverable or not. (In this problem, no transaction has read operation on a data item after that data item is written by another transaction, so the given schedule is recoverable.)

Step 2: Check for cascading recoverable. In cascading recoverable schedule, no uncommitted read is allowed. In the given problem, there is no uncommitted read so the given schedule is cascadless schedule.

Step 3: Check for strict recoverable. There should be commit operation between write operations on similar data items by two different transactions. In the given problem, transaction T_2 and transaction T_3 have write operations on data item (y), but no commit operation is performed by transaction T_3 before data item is updated by transaction T_2. Hence, the given schedule is not strict recoverable.

8.8.4 Classification of Schedule Based on Serializability

A concurrent transaction schedule is said to be a serializable schedule if its outcome (resulting database state) is equal to the outcome of the serial execution of transactions in the given concurrent schedule. Serializable schedule can be divided into two types: conflict serializability and view serializability.

8.8.4.1 Conflict Serializability

A concurrent schedule S is said to be conflict serializable schedule if S is conflict equivalent to a serial schedule.

1. **Conflict equivalent schedule:** Let there be two schedules S_1 and S_2. If after swapping consecutive non-conflict pairs of operation of transactions in schedule S_1 results in schedule S_2, then S_1 and S_2 schedules are called conflict equivalent schedules.
2. **Conflict and non-conflict pairs:** A pair of operations in a schedule is called conflict schedule if they have all three conditions:
   - At least one of the operations should be write operation.
   - Both the operations should be performed on the same data item.
   - Both operations should be performed by different transactions.

If a pair of operations do not fulfill the above three conditions, then the pair is call a non-conflict pair.

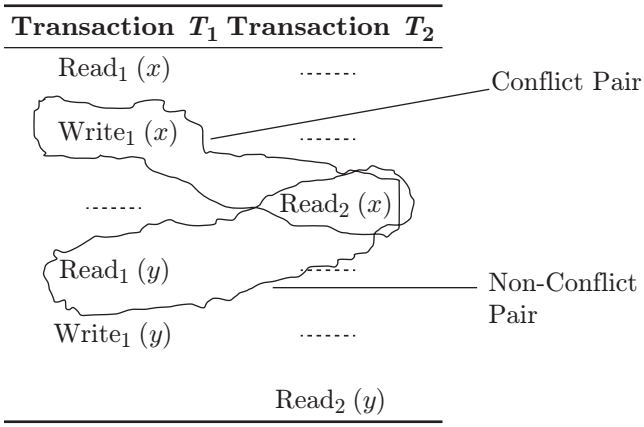
Problem 8.10: A schedule S_1 with two transactions T_1 and T_2 is given as:

$\text{Read}_1(x)$, $\text{Write}_1(x)$, $\text{Read}_2(x)$, $\text{Read}_1(y)$, $\text{Write}_1(y)$, $\text{Read}_2(y)$,....

Find conflict equivalent schedule to S_1.

Solution:

Schedule S_1:



Conflict and non-conflict pairs in S_1.

After swapping non-conflict pair in schedule S_1 we get another schedule called S_2. The schedules S_1 and S_2 are conflict equivalent schedules.

Schedule S_2:

Transaction T_1	Transaction T_2
Read$_1$ (x)	------
Write$_1$ (x)	------
Read$_1$ (y)	------
------	Read$_2$ (x)
Write$_1$ (y)	------
------	Read$_2$ (y)

Conflict equivalent schedules.

3. **Testing method of conflict serializable schedule:** To check that a given schedule is conflict serializable or not, we use precedence graph. If precedence drawn from a schedule does not contain a cycle then only it is a conflict serializable schedule. If precedence graph contains a cycle then it is not a conflict serial schedule. A precedence graph G contains:
   (a) Vertex set V: denotes the transactions of the schedule.
   (b) Edge between two vertex from V_i to V_j will be draw where $i \neq j$ and V_i's operation occur before V_j's operation, if V_i *and* V_j have following cases:

(i) V_i : Read (x) operation and V_j : Write (x) operation

(ii) V_i : Write (x) operation and V_j : Read (x) operation

(iii) V_i : Write (x) operation and V_j : Write (x) operation

Problem 8.11: Consider the given schedule S and draw precedence graph for S.

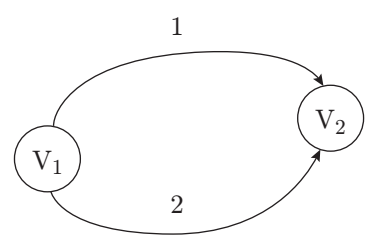
Transaction T_1	Transaction T_2
------	Read$_1$ (x)
Write$_1$ (x)	------
------	Read$_2$ (x)
Write$_1$ (y)	------
------	Read$_2$ (y)

Solution:

Vertices V_1 and V_2 represent transactions T_1 and T_2.

Edge 1 for
V_2 : Read (x) operation and V_1 : Write (x) operation

Edge 2 for
V_1 : Write (y) operation and V_2 : Read (y) operation



Note: If two schedules S_1 and S_2 are given then they will be conflict equivalent if they satisfy following conditions:

- Precedence graph of S_1 and S_2 should be equal.
- Each transaction operation should be equal in S_1 and S_2. (The two transaction will be equal of they have same number of operations and type of operation are also same.)

Problem 8.12: Consider the given schedule S and identify that schedule S is conflict serial schedule or not by the use of precedence graph.

Transaction T_1	Transaction T_2	Transaction T_3
Read (x)	------	------
Write (x)	------	------
------	Read (x)	------
------	------	Write (x)
------	Write (y)	------
Write (y)	------	------
------	------	Read (y)

Solution:

Vertices V_1, V_2 and V_3 represent transactions T_1, T_2 and T_3.

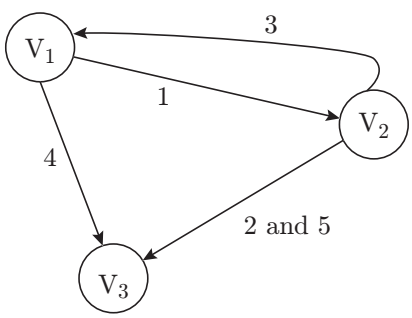
Edge 1 for
V_1 : Write (x) operation and V_2 : Read (x) operation

Edge 2 for
V_2 : Read (x) operation and V_3 : Write (x) operation

Edge 3 for
V_2 : Write (y) operation and V_1 : Write (y) operation

Edge 4 for
V_1 : Write (y) operation and V_3 : Read (y) operation

Edge 5 for
V_2 : Write (y) operation and V_3 : Read (y) operation



Precedence graph drawn from the given schedule contains cycle with vertices V_1 and V_2. So, the given schedule is not a conflict serializable schedule.

8.8.4.2 View Serializability

If a given concurrent schedule is view equivalent to one of the possible serial schedule, then it is called a view serializable schedule. Let there be two schedules S_1 and S_2 with the similar set of transactions. Schedules S_1 and S_2 will be view equivalent if they satisfy following three conditions:

1. If transaction T_i reads the initial value of data item (x) in schedule S_1, then transaction T_i must read the initial value of (x) in schedule S_2.
2. In schedule S_1, if transaction T_i reads the value of data item (x) produced by transaction T_j then T_i must reads the value of (x) that produced by transaction T_j in the schedule S_2.
3. If transaction T_i write the final value of data item (x) in schedule S_1, then transaction T_i must write the final value of (x) in schedule S_2.

Problem 8.13: Identify whether the given concurrent schedule is a view serializable schedule or not?

T_1	T_2	T_3
------	------	Read (x)
------	Read (x)	------
------	------	Write (x)
Read (x)	------	------
Write (x)	------	------

Solution: We have a concurrent schedule S_1 with three transactions, so the number of possible serializable schedules with three transactions is $8\,(2^3 = 8)$. So, we will pick one serializable schedule S_2 and if the given concurrent schedule is be view equivalent to that schedule, only then would the given schedule will be view serializable.

T_1	T_2	T_3
------	------	Read (x)
------	Read (x)	------
------	------	Write (x)
Read (x)	------	------
Write (x)	------	------

Schedule S_1

T_2	T_3	T_1
Read (x)		
	Read (x)	
	Write (x)	
		Read (x)
		Write (x)

Schedule S_2

Step 1: Initial Reads. Transactions T_3 and T_2 read initial value of data item (x) in schedule S_1 (will not consider T_1 because it reads updated value of x by T_3). T_3 and T_2 also read the initial value of (x) in serial schedule S_2. So, the first condition is true.

Step 2: W-R Conflicts. In the given schedule S_1, there is only one W-R conflict (between T_3 and T_1). Transaction T_1 reads the value of x produced by transaction T_3 in schedule. In schedule S_2, transaction T_1 also reads the value of x produced by transaction T_3 in schedule. So, the second condition is also true.

Step 3: Final Write. Transaction T_1 writes the final value of data item (x) in schedules S_1 and S_2 both. So, the third condition is also satisfied.

All the three conditions are satisfied, so the given concurrent schedule S_1 is view equivalent to one of the possible serial schedule S_2. Therefore, the given concurrent schedule is a view serializable schedule.

8.8.5 Concurrency Control Protocol

Concurrency control protocols are used to ensure serializability of transactions in a concurrent transaction execution environment. They can be lock based protocols and Time stamp ordering protocols.

8.8.5.1 Lock Based Protocol

Lock based protocols uses a mechanism through which a transaction has to obtain a read lock (for read operation) and write lock (for write operation) on a data item (x) without obtaining a lock transaction not allowed to perform any operation. There are two types of lock mechanisms supported by lock based protocol.

1. **Shared lock:** If a transaction T has shared lock on a data item (x), then it can only perform Read operation on that data item.
2. **Exclusive lock:** If a transaction T has exclusive lock on a data item (x), then it can perform Read or Write any operation on that data item.

Lock Compatibility Matrix is given in Table 8.6, which shows that if a transaction has shared/exclusive lock then for which lock it can make a request.

Table 8.6 | Lock compatibility matrix

Lock Hold / Lock Request	Shared Lock	Exclusive Lock
Shared Lock	Yes	No
Exclusive Lock	No	No

1. **Two Phase Locking Protocol (2PL):** Two phase locking protocol is used by the transaction to lock a data item before reading and writing to maintained consistency in the database. The protocol uses two phases to apply the locking scheme as:
   - *Expanding phase:* Transaction acquired locks in this phase and no locks are released.
   - *Shrinking phase:* Transaction releases locks in this phase and no locks are acquired.

   Note: Two phase locking protocol ensure conflict serializability, but may not be free from irrecoverable, deadlock and starvation.
2. **Two Phase Locking Protocol with Lock Up-gradation:** This technique allows transaction holding a shared lock on a data item can upgrade shared lock into the exclusive lock on the data item without performing unlock for shared lock.
3. **Strict Two Phase Locking Protocol:** In this protocol, if a transaction has exclusive lock on a data item, then it cannot release the lock until commit or rollback is performed by that transaction.
4. **Rigorous Two Phase Locking Protocol:** In this protocol, if a transaction has any lock (exclusive or shared) on a data item, then it cannot release the lock until commit or rollback performed by that transaction.

8.8.5.2 Time Stamp Ordering Protocol

The time stamp ordering protocol uses time stamp values (unique) of the transactions assigned by database to ensure serializability among transactions. A time stamp is a unique value assigned by the database in ascending order. Suppose we have three transactions T_1, T_2 and T_3 arriving in database, then assigned time stamp values will be as $T_1 < T_2 < T_3$. A data item has two types of time stamp.

1. **Read Time Stamp (x):** It is the highest transaction time stamp that has performed Read operation successfully on a data item (x). It is also denoted by RTS(x). Initial value of RTS(x) is zero.
2. **Write Time Stamp (x):** It is the highest transaction time stamp that has performed Write operation successfully on a data item (x). It is also denoted by WTS(x). Initial value of WTS(x) is zero.
3. **Basic Time Ordering Protocol**
   - Transaction T_i issues Read(x) operation
     (a) If (WTS(x) > Time Stamp(T_i))

     Rollback T_i

     (b) Otherwise allowed execution of Read(x) by transaction T_i
     Set RTS(x) = Maximum (RTS(x), Time Stamp (T_i))

- Transaction T issues Write(x) operation
  (a) If (RTS(x) > Time Stamp(T_i))

  Rollback T_i

  (b) If (WTS(x) > Time Stamp(T_i))

  Rollback T_i

Note: Basic Time Ordering protocol ensures serializability, equivalent serial schedule based on time stamp Ordering. It is deadlock free protocol, but basic time ordering protocol can have starvation and irrecoverable schedule (because it does not ensure order of commit).

4. **Strict Time Ordering Protocol:** In this protocol, an additional condition is added in the Basic Time Ordering protocol. This condition is as follows:

   Let a schedule have two transactions T_i and T_j where time stamp of (T_i) is less than time stamp of T_j. If transaction T_j issues Read(x)/ Write(x) operation with WTS(x) < Time Stamp (T_j) then (T_j) has to be delayed Read(x)/ Write(x) operation until commit or rollback of transaction T_i that has performed Write(x).

   Strict Time Ordering Protocol ensures serializability and deadlock free. But it may suffer from starvation.

5. **Deadlock Prevention:** To prevent deadlock situation in a database system it is very important to inspect all the operation performed by transactions in a schedule. There are various deadlock prevention scheme which uses time stamp ordering mechanism.

6. **Wait-Die Protocol:** Assume TimeStamp (T_1) < ... < TimeStamp (T_n) and T_i and T_j are any two transactions and transaction T_i is older than transaction T_j. This implies

   TimeStamp (T_i) < ... < TimeStamp (T_j).

   - If transaction T_i required a lock which is hold by transaction T_j, then transaction T_i is allowed to wait.
   - If transaction T_j required a lock which is hold by transaction T_i then transaction T_j will rollback.

7. **Wound-Wait Protocol:** Let TimeStamp (T_1) < ... < TimeStamp (T_n) and T_i and T_j are any two transactions and transaction T_i is older than transaction T_j. This implies

   TimeStamp (T_i) < TimeStamp (T_j)

   - If transaction T_i required a lock which is hold by transaction T_j then rollback transaction T_j.
   - If transaction T_j required a lock which is hold by transaction T_i then transaction T_j is allowed to wait.

IMPORTANT FORMULAS

1. All the 3NF and BCNF decomposition guarantee lossless join.
2. All 3NF and BCNF decomposition may not guarantee dependency preservation.
3. Any relation with two attributes is in BCNF.
4. Functional dependency F: $X \rightarrow Y$ implies that for any two tuples if $t_1[X] = t_2[X]$, they must have $t_1[Y] = t_2[Y]$.
5. CP: Child Pointer
6. KV: Key Value
7. Order of internal node in B+ tree is
   $P_I \cdot \text{CP} + (P_I - 1)\text{KV} \Leftarrow$ Block size

Inference Rules:

1. **Reflexivity**
   If $Y \subseteq X$, then $X \rightarrow Y$
2. **Augmentation**
   If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
3. **Transitivity**
   If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
4. **Union**
   If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
5. **Decomposition**
   If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
6. **Pseudo-transitivity**
   If $A \rightarrow B$ and $BC \rightarrow D$, then $AC \rightarrow D$
7. Number of rows in cross product = $r_1 \times r_2$
8. Number of attributes in cross product = $m + n$
9. Complete set of operations are $\{\sigma, \pi, \cup, -, \times\}$

10. In clustering index, file is ordered on non-key field.

11. Secondary indexes are created on unordered file on either key or non-key field.

12. Different kind of keys – Primary Key, Secondary Key, Candidate Key, Alternate key, Foreign Key, Superkey, Compound key (or Composite key) (or concatenated key).

SOLVED EXAMPLES

1. Which of the following operations does not modify the database?

(a) Sorting (b) Insertion at the beginning
(c) Append (d) Modify

Solution: Sorting does not involving addition/ insertion or modification of any element.

Ans. (a)

2. Which of the following operations is based on relational algebra?

(a) rename and union
(b) select and union
(c) only union
(d) rename, select and union

Solution: Rename, select and union are some of the operations of relational algebra.

Ans. (d)

3. ACID properties of a transaction are

(a) atomicity, constant, integrity, durability.
(b) atomicity, consistency, isolation, durability.
(c) atomicity, compact, in-built, durability.
(d) automatically, consistency, isolation, database.

Solution: ACID properties are atomicity, consistency, isolation and durability.

Ans. (b)

4. Which normal form is concerned with removing transitive dependency?

(a) 1NF (b) 2NF
(c) 3NF (d) BCNF

Solution: 3NF is also the highest normal form.

Ans. (c)

5. The concept Database Lock is used to overcome the problem of

(a) lost update and inconsistent data.
(b) uncommitted dependency and inconsistent data.
(c) inconsistent data only.
(d) lost updates, inconsistent data and uncommitted dependency.

Solution: Lost update, inconsistent data and uncommitted (WR) problems are overcome by database lock.

Ans. (d)

6. Database language may consist of the following?

(a) DDL and DML
(b) DML
(c) Query language
(d) DDL, DML, query language

Solution: DDL, DML, query language are all data base languages.

Ans. (d)

7. Which of the following is not a function of DBA?

(a) Network maintenance
(b) Routine maintenance
(c) Defining the schema
(d) Data access through authorization

Solution: The role of database administrator is to manage the DBMS.

Ans. (a)

8. In a relational database the category type of information is represented in

(a) tuple. (b) field.
(c) primary key. (d) database name.

Solution: Field shows type of information in relational database.

Ans. (b)

9. Which key is used to identify a tuple uniquely?

(a) Primary key (b) Tuple key
(c) Unique key (d) Domain key

Solution: Primary key.

Ans. (a)

10. An association of the information is represented by

(a) an attribute. (b) a relationship.
(c) a normal form. (d) the records.

Solution: Relationship shows association of information.

Ans. (b)

11. In which stage of database design, all the necessary fields and their types of a database are listed?

(a) Data definition (b) Data field definition
(c) E-R diagram (d) User definition

Solution: In data definition stage, all the fields and their types are listed.

Ans. (a)

12. The maximum length of an attribute of type text is

(a) 127. (b) 255.
(c) 256. (d) It is a variable.

Solution: 255

Ans. (b)

13. When a transaction has completed all its operations and is waiting for commit or rollback action, the state of transaction is

(a) partially commit. (b) ready commit.
(c) half commit. (d) commit and rollback.

Solution: The state of that transaction is partially commit.

Ans. (a)

14. Which of the following statement is not false?

(a) Time stamp protocols avoid deadlock.
(b) Locking technique is used to avoid deadlock.
(c) 2Phase locking does not provide serializability.
(d) 2Phase deals with input and storing phase.

Solution: Time stamp protocol is a deadlock free protocol.

Ans. (a)

15. Data integrity control makes use of _____ for maintaining the integrity of database.

(a) specific alphabets (uppercase and lowercase alphabets)
(b) passwords
(c) relational algebra
(d) storing on a backup hard disk

Solution: It promotes and enforces some integrity rules for the reducing data redundancy and increasing data consistency.

Ans. (a)

16. Data warehouse provides

(a) transaction responsiveness.
(b) storage, functionality responsiveness to queries.
(c) demand and supply responsiveness.
(d) storage of transactions.

Solution: It provides data storage and responsiveness to queries.

Ans. (a)

17. If D_1, D_2, … D_n are domains in a relational model, then the relation is a table, which is a subset of

(a) $D_1 \oplus D_2 \oplus \ldots \oplus D_n$ (b) $D_1 \times D_2 \times \ldots \times D_n$
(c) $D_1 \cup D_2 \cup \ldots \cup D_n$ (d) $D_1 \cap D_2 \cap \ldots \cap D_n$

Solution: $D_1 \times D_2 \times \ldots \times D_n$

Ans. (a)

18. Which normal form is considered adequate for normal relational database design?

(a) 2NF (b) 5NF
(c) 4NF (d) 3NF

Solution: Decomposition in BCNF is not always lossless and dependency preserving. So, 3NF is considered to be most adequate form in relational database.

Ans. (d)

19. Let $R = (A, B, C, D, E, F)$ be a relation scheme with the following dependencies $C \rightarrow F$, $E \rightarrow A$, $EC \rightarrow D$, $A \rightarrow B$. Which of the following is a key for R?

(a) CD (b) EC (c) AE (d) AC

Solution: To find key, we try to find closure.

$$(CD)+ = \{C, D, F\}$$
$$(EC)+ = \{A, B, C, D, E, F\}$$
$$(AE)+ = \{A, B, E\}$$
$$(AC)+ = \{A, B, C, F\}$$

Only EC satisfies the closure property.

Ans. (b)

20. Give the following relation instance:

x	y	z
1	4	2
1	5	3
1	6	3
3	2	2

Which of the following functional dependencies are satisfied by the instance?

(a) $XY \rightarrow Z$ and $Z \rightarrow Y$
(b) $YZ \rightarrow X$ and $Y \rightarrow Z$
(c) $YZ \rightarrow X$ and $X \rightarrow Z$
(d) $XZ \rightarrow Y$ and $Y \rightarrow X$

Solution: Functional dependency should uniquely identify the value. From the given options only the following satisfies this condition:

$$XY \to Z,\ YZ \to X,\ Y \to Z \textbf{ AND } Y \to X$$

Ans. (b)

21. Relation R is decomposed using a set of functional dependencies F, and relation S is decomposed using another set of functional dependencies G. One decomposition is definitely BCNF, the other is definitely 3NF, but it is not known which is which. To make a guaranteed identification, which on one of the following tests should be used on the decompositions? (Assume that the closures of F and G are available.)

(a) Dependency-preservation
(b) Lossless-join
(c) BCNF definition
(d) 3NF definition

Solution: All the 3NF and BCNF decomposition guarantee lossless join. All 3NF and BCNF decomposition does not guaranty dependency preservation. According to this, only option (c) can be correct.

Ans. (c)

22. Consider the join of a relation R with a relation S. If R has m tuples and S has n tuples then the maximum and minimum sizes of the join, respectively, are

(a) $m + n$ and 0.
(b) mn and 0.
(c) $m + n$ and $|m - n|$.
(d) mn and $m + n$.

Solution: Number of tuples in their join gets multiplied with number of tuples in both the relations. So, the maximum and minimum tuples will be mn and 0, respectively.

Ans. (b)

23. Given the relations

Employee(name, salary, deptno) and department (deptno, deptname, address), which of the following queries cannot be expressed using the basic relational algebra operations (σ, π, $\bowtie$, E, $\cap$, $-$)?

(a) Department address of every employee
(b) Employees whose name is the same as their department name
(c) The sum of all employees' salaries
(d) All employees of a given department

Solution: Aggregate functions such as sum, avg, min and max cannot be expressed in terms of basic relational algebra operations. These require extended relational algebra.

Ans. (c)

GATE PREVIOUS YEARS' QUESTIONS

1. Which of the following scenarios may lead to an irrecoverable error in a database system?

(a) A transaction writes a data item after it is read by an uncommitted transaction.
(b) A transaction reads a data item after it is read by an uncommitted transaction.
(c) A transaction reads a data item after it is written by a committed transaction.
(d) A transaction reads a data item after it is written by an uncommitted transaction.

(GATE 2003: 1 Mark)

Solution: In option (d), a problem will occur. Suppose first transaction A is reading data written by second transaction B, but B has not committed yet. If A acts on that data and eventually commits, but B rollback. A will have "acted on" data that was never committed. So it will lead to irrecoverable error in database.

Ans. (d)

2. Consider the following SQL query:

```
SELECT distinct a1,a2,K,anaa
FROM r1,r2,L,rmr
WHERE P
```

For an arbitrary predicate P, this query is equivalent to which of the following relational algebra expressions?

(a) $\Pi_{a_1,a_2,\ldots,a_n}\sigma_p(r_1 \times r_2 \times \cdots \times r_m)$

(b) $\Pi_{a_1,a_2,K,\ a_n}\sigma_p\ (r_1 \bowtie r_2 \bowtie \mathrm{L} \bowtie r_\mathrm{m})$

(c) $\Pi_{a_1,a_2,K,a_n}\sigma_p\left(r_1 \cup r_2 \cup L \cup r_m\right)$

(d) $\Pi_{a_1,a_2,K,a_n}\sigma_p\left(r_1 \cap r_2 \cap L \cap r_m\right)$

(GATE 2003: 1 Mark)

Solution: Π is projection that by default selects distinct value of attributes. Cross product will

consider all the tables $r_1, r_2, \ldots r_n$. Row (σ) selects all the rows satisfying predicate P.

Ans. (a)

3. Consider the following functional dependencies in a database:

```
Date_of_birth → Age
Age → Eligibility
Name → Roll_number
Roll_number → Name
Course_number → Course_name
Course_number → Instructor
(Roll_number, Course_number) → Grade
```

The relation (Roll_number, Name, Date_of_birth, Age) is

(a) In Second normal form but not in Third normal form
(b) In Third normal form but not in BCNF
(c) In BCNF
(d) In none of the above

(GATE 2003: 2 Marks)

Solution: The applicable FDs are:

```
Date_of_birth → Age
Age → Eligibility
Name → Roll_number
Roll_number → Name
```

The candidate keys are

(Roll_number, Date_of_birth) and (Name, Date_of_birth)

For candidate key the FD Date_of_birth → Age is a partial dependency.

So, the relation is in 1NF.

Candidate keys for given relation are: (date of birth, name) and (date of birth, roll_number)

Data of Birth → Age

Age (non-prime) is partially dependent upon Date of Birth which is prime attribute. According to 2NF definition, every non-prime attribute should be fully functional dependent on key, so it is not in 2NF. Therefore, it will not be in 3NF and BCNF also.

Ans. (d)

4. Consider the set of relations shown below and the SQL query that follows:

```
Students: (Roll_number, Name, Date_of_birth)
Courses: (Course number, Course_name, Instructor)
Grades: (Roll_number, Course_number, Grade)
SELECT distinct Name
FROM Students, Courses, Grades
WHERE Students.Roll_number = Grades.Roll_number
    and Courses.Instructor = Shyam
    and Courses.Course_number = Grades.Course_number
    and Grades.grade = A
```

Which of the following sets is computed by the above query?

(a) Names of students who have got an A grade in all courses taught by Shyam.
(b) Names of students who have got an A grade in all courses.
(c) Names of students who have got an A grade in at least one of the courses taught by Shyam.
(d) None of the above.

(GATE 2003: 2 Marks)

Solution: Use of distinct selects the name only once. Due to distinct selected names will be unique, so one person will be selected only once, in spite of getting grade A in any number of courses taught by Shyam.

Ans. (c)

5. Consider three data items D_1, D_2 and D_3, and the following execution schedule of transactions T_1, T_2 and T_3. In the diagram, $R(D)$ and $W(D)$ denote the actions reading and writing the data item D, respectively.

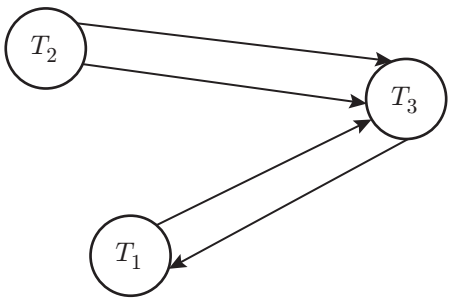
T	T_2	T_3
	$R(D_3)$;	
	$R(D_2)$;	
	$W(D_2)$;	
		$R(D_2)$;
		$R(D_3)$;
$R(D_1)$;		
$W(D_1)$;		
		$W(D_2)$;
		$W(D_3)$;
	$R(D_1)$;	
$R(D_2)$;		
$W(D_2)$;		
	$W(D_1)$;	

Which of the following statements is correct?

(a) The schedule is serializable as T_2; T_3; T_1.
(b) The schedule is serializable as T_2; T_1; T_3.
(c) The schedule is serializable as T_3; T_2; T_1.
(d) The schedule is not serializable.

(GATE 2003: 2 Marks)

Solution: Let us draw a flow diagram between the three schedules. An edge is drawn between two schedules if there exists read-write or write-write dependency between them.



The schedule has cycle between T_1 and T_3. So, the schedule is not serializable.

Ans. (d)

6. Let $R_1(A, B, C, D)$ and $R_2(D, E)$ be two relation schema where the primary keys are shown underlined, and let C be a foreign key in R_1 referring to R_2. Suppose there is no violation of the above referential integrity constraint in the corresponding relation instances r_1 and r_2. Which one of the following relational algebra expressions would necessarily produce an empty relation?

(a) $\Pi_D(r_2) - \Pi_C(r_1)$ (b) $\Pi_C(r_1) - \Pi_D(r_2)$
(c) $\Pi_D(r_1 \bowtie_{C \neq D} r_2)$ (d) $\Pi_D(r_1 \bowtie_{C = D} r_2)$

(GATE 2004: 1 Mark)

Solution: C is foreign key in relation R1 referring to relation R2. So, C would not contain any additional value that is not present in D. So $\Pi_C(r_1) - \Pi_D(r_2)$ will return no value.
Returns all the tuples which are in $\Pi_C(r_1)$ but not in $\Pi_D(r_2)$.

Ans. (b)

7. Consider the following relation schema pertaining to a student's database:

Students(<u>rollno</u>, name, address)

Enroll(<u>rollno</u>, courseno, coursename)

where the primary keys are shown underlined. The number of tuples in the student and Enroll tables are 120 and 8, respectively. What are the maximum and minimum number of tuples that can be present in (Student $*$ Enroll), where '$*$' denotes natural join?

(a) 8, 8 (b) 120, 8
(c) 960, 8 (d) 960, 120

(GATE 2004: 1 Mark)

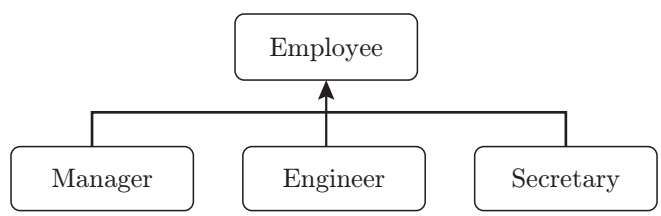
Solution: The maximum and minimum number of tuples presented in (Student $*$ Enroll) would be represented by the minimum of these 120 and 8, that is, min(120, 8) = 8
Natural join has inbuilt condition of equality. So in both the relations, minimum and maximum tuples will be 8.

Ans. (a)

8. It is desired to design an object-oriented employee record system for a company. Each employee has a name, unique id and salary. Employees belong to different categories and their salary is determined by their category. The functions getName, getId and compute salary are required. Given the class hierarchy below, possible locations for these functions are:

(i) getId is implemented in the superclass
(ii) getId is implemented in the subclass
(iii) getName is an abstract function in the superclass
(iv) getName is implemented in the superclass
(v) getName is implemented in the subclass
(vi) getSalary is an abstract function in the superclass
(vii) getSalary is implemented in the superclass
(viii) getSalary is implemented in the subclass

(GATE 2004: 2 Marks)



Choose the best design

(a) (i), (iv), (vi), (viii) (b) (i), (iv), (vii)
(c) (i), (iii), (v), (vi), (viii) (d) (ii), (v), (viii)

Solution: getid and getname can be placed in base class. Because all the subclasses will have the same implementation for these functions. But getsalary is dependent on category of employee. So, abstract function can be placed in base class and its implementation in the subclasses.

Ans. (a)

9. The relation scheme Student Performance (Name, CourseNo, RollNo, Grade) has the following functional dependencies:

Name, CourseNo → Grade
RollNo, CourseNo → Grade
Name → RollNo
RollNo → Name

The highest normal form of this relation scheme is

(a) 2NF (b) 3NF
(c) BCNF (d) 4NF

(GATE 2004: 2 Marks)

Solution: Name and RollNo are candidate keys. The attributes are not repeated. So, the relation schema is in 1 NF.
There is no partial dependency. So, the relation is in 2NF.
Grade is not fully dependent upon all candidate keys, so it is not in 3NF.
Candidate keys for given relation are: (rollNo, courseNo) and (name, courseNo) there is no partial and transitive dependencies. So the highest normal form is 3NF. name → rollNo and rollNo → name are violating the condition of BCNF.

Ans. (b)

10. Consider the relation Student (<u>name</u>, sex, marks), where the primary key is shown underlined, pertaining to students in a class that has at least one boy and one girl. What does the following relational algebra expression produce?

(Note: ρ is the rename operator).

$$\Pi_{\text{name}}(r_{\text{sex=female}}(\text{Student}) - \Pi_{\text{name}}(\text{Student}_{\text{sex=female}} \Lambda x = \text{male} \Lambda \text{marks} \leq m \rho_{n,x,m}(\text{Student}))$$

(a) Names of girl students with the highest marks
(b) Names of girl students with more marks than some boy student
(c) Names of girl students with marks not less than some boy student
(d) Names of girl students with more marks than all the boy students

(GATE 2004: 2 Marks)

Solution: The query is Part 1 − Part 2. Part 2 of the query results in names of girls having marks less than or equal to marks of all the boys. And Part 1 selects the names of all the girls in class. So, Part 1 − Part 2 will result in names of girl students having more marks than all the boy students.

Ans. (d)

11. The order of an internal node in a B+ tree index is the maximum number of children it can have. Suppose that a child pointer takes 6 bytes, the search field value takes 14 bytes, and the block size is 512 bytes. What is the order of the internal node?

(a) 24 (b) 25
(c) 26 (d) 27

(GATE 2004: 2 Marks)

Solution: Order of internal node is $P_I \cdot CP + (P_I - 1)$
KV <= Block Size
Child pointer (CP) = 6 B
Key value (KV) = 14 B
Block size = 512 B
$6 + (P_I - 1) \times 14 = 512$
$P_I = 26$

Ans. (c)

12. The employee information in a company is stored in the following relation:

```
Employee(name, sex, salary, deptName)
```

Consider the following SQL query:

```
SELECT deptName
FROM Employee
WHERE sex = 'M'
GROUP by deptName
HAVING avg(salary)>
 (SELECT avg (salary) FROM Employee)
```

It returns the names of the department in which

(a) the average salary is more than the average salary in the company.
(b) the average salary of male employees is more than the average salary of all male employees in the company.
(c) the average salary of male employees is more than the average salary of employees in the same department.
(d) the average salary of male employees is more than the average salary in the company.

(GATE 2004: 2 Marks)

Solution: Inner query selects the average salary of employees in the company. Outer query selects average salary of male employees. So, the whole query finds the department name of whose average salary of male employees is more than the average salary in the company.

Ans. (d)

13. Which one of the following is a key factor for preferring B+ trees to binary search trees for indexing database relations?

(a) Database relations have a large number of records.
(b) Database relations are sorted on the primary key.
(c) B+ trees require less memory than binary search trees.
(d) Data transfer from disks is in blocks.

(GATE 2005: 1 Mark)

Solution: Indexing performs well for large data blocks.
B+ trees are preferred over the binary search trees.
B+ trees transfer data from disk to primary memory in form of data blocks.
In case of B+ trees, data moves in terms of blocks.

Ans. (d)

14. Which one of the following statements about normal forms is FALSE?

(a) BCNF is stricter than 3NF.
(b) Lossless, dependency-preserving decomposition into 3NF is always possible.
(c) Lossless, dependency-preserving decomposition into BCNF is always possible.
(d) Any relation with two attributes is in BCNF.

(GATE 2005: 1 Mark)

Solution: It is not always possible to decompose a table in BCNF and preserve dependencies. For example, a set of functional dependencies $\{AB \rightarrow C, C \rightarrow B\}$ cannot be decomposed in BCNF.

Ans. (c)

15. Let r be a relation instance with schema $R = (A, B, C, D)$. We define $r_1 = \Pi_{A, B, C}(R)$ and $r_2 = \Pi_{A, D}(r)$. Let $s = r_1 * r_2$ where $*$ denotes natural join. Given that the decomposition of r into r_1 and r_1 is lossy, which one of the following is TRUE?

(a) $s \subset r$ (b) $r \cup s = r$
(c) $r \subset s$ (d) $r * s = s$

(GATE 2005: 1 Mark)

Solution:

Table r

A	B	C	D
1	10	100	1000
1	20	200	1000
	20	200	2000

Table r_1

A	B	C
1	10	100
1	20	200

Table r_2

A	D
1	1000
1	2000

Table s (natural join of r_1 and r_2)

A	B	C	D
1	10	100	1000
1	20	200	1000
1	20	100	200
1	20	200	2000

Ans. (c)

16. Let E_1 and E_2 be two entities in an E/R diagram with simple single-valued attributes. R_1 and R_2 are two relationships between E_1 and E_2, where R_1 is one-to-many and R_2 is many-to-many. R_1 and R_2 do not have any attributes of heir own. What is the minimum number of tables required to represent this situation in the relational model?

(a) 2 (b) 3
(c) 4 (d) 5

(GATE 2005: 2 Marks)

Solution:

E_1
l
m
n

E_2
x
y
z

R_2

E_1	E_2
L	x
L	y
M	y

The one-to-many relationships are represented with entity set from one side. In one-to-many relationships, each entity in the entity set can be associated with at most one entity of the other. So, the table is not formed for R_1. Hence, the tables are formed for R_2, E_1 and E_2.
Using cross-reference technique, minimum of 3 tables are required.

Ans. (b)

17. The following table has two attributes A and C where A is the primary key and C is the foreign key referencing A with on-delete cascade.

A	C
2	4
3	4

(*Continued*)

Continued

A	C
4	3
5	2
7	2
9	5
6	4

The set of all tuples that must be additionally deleted to preserve referential integrity when the tuple (2, 4) is deleted is

(a) (3, 4) and (6, 4)
(b) (5, 2) and (7, 2)
(c) (5, 2), (7, 2) and (9, 5)
(d) (3, 4), (4, 3) and (6, 4)

(GATE 2005: 2 Marks)

Solution: When (2,4) is deleted, as C is a foreign key referring A with delete on cascade, all entries with value 2 in C must be deleted.
So, (5, 2) and (7, 2) are deleted. As a result of this, 5 and 7 are deleted from A, which causes (9, 5) to be deleted.

Ans. (c)

18. The relation book (title, price) contains the titles and prices of different books. Assuming that no two books have the same price, what does the following SQL query list?

```
SELECT title
FROM book as B
WHERE (SELECT count(*)
        FROM book as T
        WHERE T.price > B.price) < 5
```

(a) Titles of the four most expensive books
(b) Title of the fifth most inexpensive book
(c) Title of the fifth most expensive book
(d) Titles of the five most expensive books

(GATE 2005: 2 Marks)

Solution: The outer query selects all titles from book table. For every selected book, the sub-query returns count of those books, which are more expensive than the selected book. The where clause of outer query will be true for five most expensive books.

Ans. (d)

19. Consider a relation scheme $R = (A, B, C, D, E, H)$ on which the following functional dependencies hold:

$$\{A \rightarrow B, BC \rightarrow D, E \rightarrow C, D \rightarrow A\}.$$

What are the candidate keys of R?

(a) AE, BE
(b) AE, BE, DE
(c) AEH, BEH, BCH
(d) AEH, BEH, DEH

(GATE 2005: 2 Marks)

Solution: Let S be a candidate key of relation R if the closure of S is all attributes of R and there is no subset of S whose closure is all attributes of R.

Closure of AEH: $AEH+ = \{ABCDEH\}$

Closure of BEH: $BEH+ = \{ABCDEH\}$

Closure of DEH: $DEH+ = \{ABCDEH\}$

Ans. (d)

20. Consider the following log sequence of two transactions on a bank account, with initial balance 12000, that transfer 2000 to a mortgage payment and then apply a 5% interest.

1. T_1 start
2. T_1 B old = 1200 new = 10000
3. T_1 M old = 0 new = 2000
4. T_1 commit
5. T_2 start
6. T_2 B old = 10000 new = 10500
7. T_2 commit

Suppose the database system crashes just before log record 7 is written. When the system is restarted, which one statement is true of the recovery procedure?

(a) We must redo log record 6 to set B to 10500.
(b) We must undo log record 6 to set B to 10000 and then redo log records 2 and 3.
(c) We need not redo log records 2 and 3 because transaction T_1 has committed.
(d) We can apply redo and undo operations in arbitrary order because they are idempotent.

(GATE 2006: 1 Mark)

Solution: When a transaction is committed, no need to redo or undo operations.

Ans. (c)

21. Consider the relation account (customer, balance) where customer is a primary key and there are no null values. We would like to rank customers according to decreasing balance. The customer with the largest balance gets rank 1. Ties are not broke but ranks are skipped: if exactly two customers

have the largest balance they each get rank 1, and rank 2 is not assigned.

Query 1: `SELECT A.customer, count(B.customer) FROM account A, account B WHERE A.balance ⇐ B.balance group by A.customer`

Query 2: `SELECT A.customer, 1+count(B.customer) FROM account A, account B WHERE A.balance < B.balance group by A.customer`

Consider these statements about Query 1 and Query 2.

I. Query 1 will produce the same row set as Query 2 for some but not all databases.
II. Both Query 1 and Query 2 are correct implementation of the specification.
III. Query 1 is a correct implementation of the specification but Query 2 is not.
IV. Neither Query 1 nor Query 2 is a correct implementation of the specification.
V. Assigning rank with a pure relational query takes less time than scanning in decreasing balance order assigning ranks using ODBC.

Which two of the above statements are correct?

(a) II and V (b) I and III
(c) I and IV (d) III and V

(GATE 2006: 2 Marks)

Solution: Both query1 and query2 perform the same task, but none of them sort the customer according to the decreasing balance. So, option (c) is correct.

Ans. (c)

22. Consider the relation enrolled (student, course) in which (student, course) is the primary key, and the relation paid (student, amount) where student is the primary key. Assume no null values and no foreign keys or integrity constraints. Given the following four queries:

Query 1: `SELECT student FROM enrolled WHERE student in (SELECT student FROM paid)`

Query 2: `SELECT student FROM paid WHERE student in (SELECT student FROM enrolled)`

Query 3: `SELECT E.student FROM enrolled E, paid P WHERE E.student = P.student`

Query 4: `SELECT student FROM paid WHERE exists`

`(SELECT * FROM enrolled WHERE enrolled.student = paid.student)`

Which one of the following statements is correct?

(a) All queries return identical row sets for any database
(b) Query 2 and Query 4 return identical row sets for all databases but there exist databases for which Query 1 and Query 2 return different row sets.
(c) There exist databases for which Query 3 returns strictly fewer rows than Query 2
(d) There exist databases for which Query 4 will encounter an integrity violation at runtime.

(GATE 2006: 2 Marks)

Solution: The output of Query 2, Query 3 and Query 4 will be identical. Query 1 produces duplicate rows. But row set produced by all the queries will be same. For example,

Table: Enrolled

Student	Course
A	C_1
B	C_1
A	C_2
C	C_1

Table: Enrolled

Student	Amount
A	100
B	100
D	200

Output of Query 1: A B

Output of Query 2: A B

Output of Query 3: A B

Output of Query 4: A B

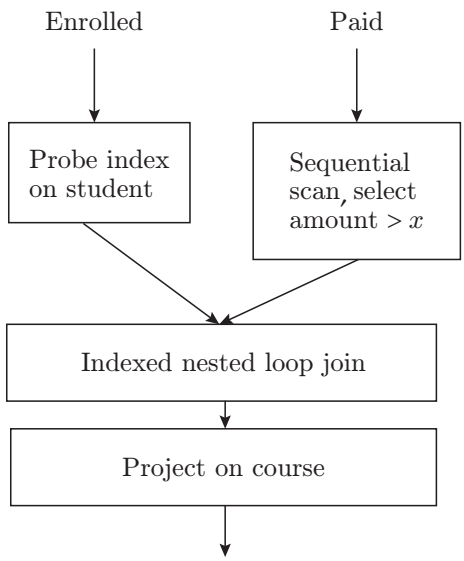
Ans. (a)

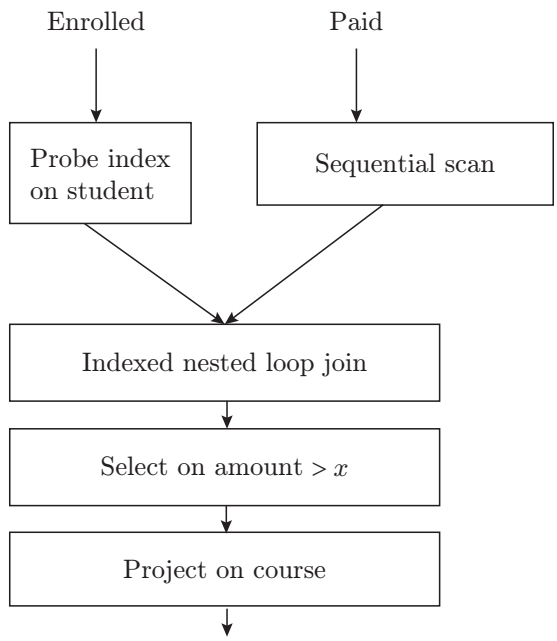
23. Consider the relation enrolled (student, course) in which (student, course) is the primary key, and the relation paid (student, amount) in which student is the primary key. Assume no null values and no foreign keys or integrity constraints. Assume that amounts 6000, 7000, 8000, 9000 and 10000 were each paid by 20% of the students. Consider these query plans (Plan 1 on left, Plan 2 on right) to "list all courses taken by students who have paid more than x".

A disk seek takes 4 ms, disk data transfer bandwidth is 300 MB/s and checking a tuple to see if amount is greater than x takes 10 μs. Which of the following statements is correct?

(a) Plan 1 and Plan 2 will not output identical row sets for all databases.
(b) A course may be listed more than once in the output of Plan 1 for some databases.
(c) For $x = 5000$, Plan 1 executes faster than Plan 2 for all databases.
(d) For $x = 9000$, Plan 1 executes slower than Plan 2 for all databases.

(GATE 2006: 2 Marks)





Solution: Plans need to load both tables' courses and enrolled. So, disk access time is the same for both plans.
Plan 2 does lesser number of comparisons compared to Plan 1.
So, join operation will require more comparisons as the second table will have more rows in Plan 2 compared to Plan 1.
The joined table of two tables will have more rows, so, more comparisons are needed to find amounts greater than x.
Plan 1 executes faster than Plan 2. First tuples are filtered then joined, whereas in plan 2 first tuples are joined and then filtered, which takes more time.

Ans. (c)

24. The following functional dependencies are given:

$AB \rightarrow CD$, $AF \rightarrow D$, $DE \rightarrow F$, $C \rightarrow G$, $F \rightarrow E$, $G \rightarrow A$

Which one of the following options is false?

(a) $\{CF\}+ = \{ACDEFG\}$
(b) $\{BG\}+ = \{ABCDG\}$
(c) $\{AF\}+ = \{ACDEFG\}$
(d) $\{AB\}+ = \{ABCDFG\}$

(GATE 2006: 2 Marks)

Solution: Closure of AF or $AF+ = \{ADEF\}$, closure of AF does not contain C and G.

The closure of {AF}+ is {A F D E}. It cannot drive all the members of set given in option (c). So, option (c) is false.

Ans. (c)

25. Information about a collection of students is given by the relation studinfo(studId, name, sex). The relation enroll(studId, courseId) gives which student has enrolled for (or taken) what course(s). Assume that every course is taken by at least one male and at least one female student. What does the following relational algebra expression represent?

$$\Pi_{\text{courseId}}(\Pi_{\text{studId}}(\sigma_{\text{sex}=\text{"female"}}(\text{studInfo})) \times \Pi_{\text{courseId}}(\text{enroll}))$$

(a) Courses in which all the female students are enrolled.
(b) Courses in which a proper subset of female students are enrolled.
(c) Courses in which only male students are enrolled.
(d) None of the above

(GATE 2007: 2 Marks)

Solution:

(i) Select studId of all female students and select all courseId of all courses.
(ii) The query performs a Cartesian product of the above select two columns in Step 1.
(iii) It subtracts enroll table from the result of Step 2.

This will remove all the (studId, courseId) pairs, which are present in enrol table.
If all female students have registered in courses, then this course will not be there in the subtracted result.
So, the complete expression returns courses in which a proper subset of female students is enrolled.

Ans. (b)

26. Consider the relation employee (name, sex, supervisorName) with name as the key. `supervisorName` gives the name of the supervisor of the employee under consideration. What does the following Tuple Relational Calculus query produce?

`{e.name|employee(e) ∧{(∀x) [¬employee(x) ∨ x.supervisorName ≠ e.name ∨ x.sex = "male")]}/` inserted on the left of expression

(a) Names of employees with a male supervisor.
(b) Names of employees with no immediate male subordinates.
(c) Names of employees with no immediate female subordinates.
(d) Names of employees with a female supervisor.

(GATE 2007: 2 Marks)

Solution: The query selects all those employees whose immediate subordinate is "male".

Ans. (b)

27. Consider the table employee (empId, name, department, salary) and the two queries Q_1, Q_2 below. Assuming that department 5 has more than one employee, and we want to find the employees who get higher salary than anyone in the department 5, which one of the statements is TRUE for any arbitrary employee table?

Q_1:
```
SELECT e.empId 1
FROM employee e
WHERE not exists (SELECT * FROM
employee s WHERE s.department = "5"
and s.salary >= e.salary)
```

Q_2:
```
SELECT e.empId
FROM employee e
WHERE e.salary > Any
(SELECT distinct salary FROM em-
ployee s WHERE s.department = "5")
```

(a) Q_1 is the correct query.
(b) Q_2 is the correct query.
(c) Both Q_1 and Q_2 produce the same answer.
(d) Neither Q_1 nor Q_2 is the correct query.

(GATE 2007: 2 Marks)

Solution: Q_1 selects an employee but does not compute the result as after the 'Where not exists', it does not have statement to produce the result.
Q_2 selects an employee who gets higher salary than anyone in the department 5. It correctly finds employees who get higher salary than anyone in the department 5.

Ans. (b)

28. Which one of the following statements is FALSE?

(a) Any relation with two attributes is in BCNF.
(b) A relation in which every key has only one attribute is in 2NF.
(c) A prime attribute can be transitively dependent on a key in a 3NF relation.
(d) A prime attribute can be transitively dependent on a key in a BCNF relation.

(GATE 2007: 2 Marks)

Solution: According to the definition of 3NF, a prime attribute can be transitively dependent on a key. Option (d) is incorrect because this does not satisfy the condition of BCNF.

Ans. (d)

29. The order of a leaf node in a B+ tree is the maximum number of (value, data record pointer) pairs it can hold. Given that the block size is 1 KB, data record pointer is 7 bytes long, the value field is 9 bytes long and a block pointer is 6 bytes long, what is the order of the leaf node?

(a) 63 (b) 64
(c) 67 (d) 68

(GATE 2007: 2 Marks)

Solution: Let the order of the leaf node be n.
Block size = 1 KB = 1024 bits

$$6 + 7n + (n - 1)9 = 1024$$
$$n = 64$$

Ans. (b)

30. Consider the following schedules involving two transactions. Which one of the following statements is TRUE?

S_1: $r_1(X)$; $r_1(Y)$; $r_2(X)$; $r_2(Y)$; $w_2(Y)$; $w_1(X)$

S_2: $r_1(X)$; $r_2(X)$; $r_2(Y)$; $w_2(Y)$; $r_1(Y)$; $w_1(X)$

(a) Both S_1 and S_2 are conflict serializable.
(b) S_1 is conflict serializable and S_2 is not conflict serializable.
(c) S_1 is not conflict serializable and S_2 is conflict serializable.
(d) Both S_1 and S_2 are not conflict serializable.

(GATE 2007: 2 Marks)

Solution:

Schedule S_1

T_1	T_2
$r_1(X)$	
$r_1(Y)$	
	$r_2(X)$
	$r_2(Y)$
	$w_2(Y)$
$w_1(X)$	

The schedule is not conflict serializable.

Schedule S_2

T_1	T_2
$r_1(X)$	
	$r_2(X)$
	$r_2(Y)$
	$w_2(Y)$
$r_1(Y)$	
$w_1(X)$	

The schedule is conflict serializable to T_2T_1.

Ans. (c)

31. A clustering index is defined on the fields which are of type

(a) Non-key and ordering
(b) Non-key and non-ordering
(c) Key and ordering
(d) Key and non-ordering

(GATE 2008: 1 Mark)

Solution: Clustering index is defined on the fields. If the records of the file are physically ordered on a non-key field, it will not have a distinct value for each record. So, the clustering index is defined on the fields of type non-key and ordering.

Ans. (a)

32. Which of the following tuple relational calculus expression(s) is/are equivalent to $\forall t \in r\ (P(t))$?

I. $\neg\exists t \in r\ (P(t))$

II. $\exists t \notin r\ (P(t))$

III. $\neg\exists t \in r(\neg P(t))$

IV. $\exists t \notin r\ (\neg P(t))$

(a) I only (b) II only
(c) III only (d) III and IV only

(GATE 2008: 1 Mark)

Solution: Using negation theorem, we find option (c) is true.

Ans. (c)

33. Let R and S be two relations with the following schema:

$R(\underline{P, Q}, R_1, R_2, R_3)$

$S(\underline{P, Q}, S_1, S_2)$

where $\{P, Q\}$ is the key for both schemas. Which of the following queries are equivalent?

I. $\Pi_P(R \bowtie S)$

II. $\Pi_P(R) \bowtie \Pi_P(S)$

III. $\Pi_P\left(\Pi_{P,Q}(R) \cap \Pi_{P,Q}(S)\right)$.

IV. $\Pi_P\left(\Pi_{P,Q}(R) - \left(\Pi_{P,Q}(R) - \Pi_{P,Q}(S)\right)\right)$

(a) Only I and II (b) Only I and III
(c) Only I, II and III (d) Only I, III and IV

(GATE 2008: 2 Marks)

Solution:

In I, P from natural join of R and S are selected.

In III, all P from intersection of (P, Q) pairs present in R and S.

IV is also equivalent to III because $(R - (R - S)) = R \cap S$.

II is not equivalent as it may also include P, where Q is not same in R and S.

Queries I, II and III performs the same operations, that is, they select attribute P.

Ans. (c)

34. Consider the following relational schemes for a library database:

Book(Title, Author, Catalog_no, Publisher, Year, Price)

Collection(Title, Author, Catalog_no)

within the following functional dependencies:

I. Title Author → Catalog_no
II. Catalog_no → Title Author Publisher Year
III. Publisher Title Year → Price

Assume {Author, Title} is the key for both schemes. Which of the following statements is true?

(a) Both Book and Collection are in BCNF.
(b) Both Book and Collection are in 3NF only.
(c) Book is in 2NF and Collection is in 3NF.
(d) Both Book and Collection are in 2NF only.

(GATE 2008: 2 Marks)

Solution: Collection is in BCNF as there is only one functional dependency Title Author → Catalog_no and {Author, Title} is key for collection.

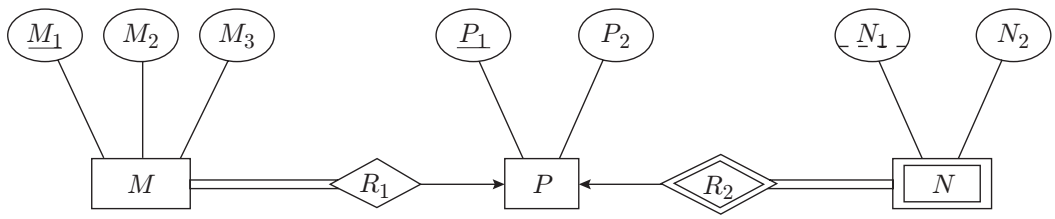
Book is not in BCNF because Catalog_no is not a key and there is a functional dependency Catalog_no → Title Author Publisher Year.

Book is not in 3NF because non-prime attributes (Publisher Year) are transitively dependent on key [Title, Author].

Book is in 2NF because every non-prime attribute of the table is dependent on either the key [Title, Author] or another non-prime attribute.

Ans. (c)

Linked Answer Questions 35 and 36: Consider the following ER diagram:



35. The minimum number of tables needed to represent M, N, P, R_1, R_2 is

(a) 2 (b) 3
(c) 4 (d) 5

(GATE 2008: 2 Marks)

Solution: Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the "many" side, containing the primary key of the "one" side.

Since R_1 is many to one and participation of M is total, M and R_1 can be combined to form the table $\{M_1, M_2, M_3, P_1\}$. N is a week entity set, so it can be combined with P.

Ans. (a)

36. Which of the following is a correct attribute set for one of the tables for the correct answer to the above question?

(a) $\{M_1, M_2, M_3, P_1\}$
(b) $\{M_1, P_1, N_1, N_2\}$
(c) $\{M_1, P_1, N_1\}$
(d) $\{M_1, P_1\}$

(GATE 2008: 2 Marks)

Solution:
For R_1 correct attribute set: M_1, M_2, M_3, P_1
For R_2 correct attribute set: N_1, N_2, P_1, P_2 with P_1 as primary key and N_1 as weak entity set.

Ans. (a)

37. Consider two transactions T_1 and T_2, and four schedules S_1, S_2, S_3, S_4 of T_1 and T_2 as given below:

T_1: $R_1[x]\,W_1[x]\,W_1[y]$

T_2: $R_2[x]R_2[y]\,W_2[y]$

S_1: $R_1[x]R_2[x]R_2[y]\,W_1[x]\,W_1[y]\,W_2[y]$

S_2: $R_1[x]R_2[x]R_2[y]\,W_1[x]\,W_2[y]\,W_1[y]$

S_3: $R_1[x]\,W_1[x]R_2[x]\,W_1[y]R_2[y]\,W_2[y]$

S_4: $R_2[x]R_2[y]R_1[x]\,W_1[x]\,W_1[y]\,W_2[y]$

Which of the above schedules are conflict-serializable?

(a) S_1 and S_2 (b) S_2 and S_3
(c) S_3 only (d) S_4 only

(GATE 2009: 2 Marks)

Solution: The serial schedule T_1T_2 has the following sequence of operations:

$$R_1[x]\,W_1[x]\,W_1[y]R_2[x]R_2[y]\,W_2[y]$$

And the schedule T_2T_1 has the following sequence of operations:

$$R_2[x]R_2[y]\,W_2[y]R_1[x]\,W_1[x]\,W_1[y]$$

The schedule S_2 is conflict-equivalent to T_2T_1 and S_3 is conflict-equivalent to T_1T_2.
Only S_2 and S_3 are conflict serialzable schedules. To test for conflict serializability, make the wait for graph. If there is a cycle in graph, it means not conflict serializable.

Ans. (b)

38. The following key values are inserted into a B+ tree in which order of the internal nodes is 3, and that of the leaf nodes is 2, in the sequence given below. The order of internal nodes is the maximum number of tree pointers in each node, and the order of leaf nodes is the maximum number of data items that can be stored in it. The B+ tree is initially empty.

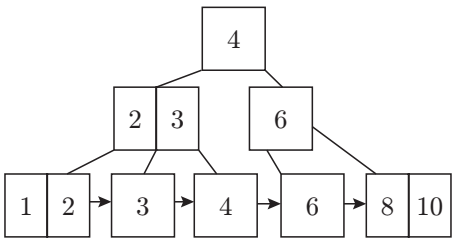
10, 3, 6, 8, 4, 2, 1

The maximum number of times leaf nodes would get split up as a result of these insertions is

(a) 2 (b) 3
(c) 4 (d) 5

(GATE 2009: 2 Marks)

Solution: Total number of splits here are 5.



Ans. (a)

39. Let R and S be relational schemes such that $R = \{a, b, c\}$ and $S = \{c\}$. Now consider the following queries on the database:

I. $\Pi_{R-S}(r) - \Pi_{R-S}(\Pi_{R-S}(r) \times s - \Pi_{R-S,S}(r))$

II. $\{t \mid t \in \Pi_{R-S}(r) \wedge \forall u \in s(\exists v \in r(u = v[s] \wedge t = v[R-S]))\}$

III. $\{t \mid t \in \Pi_{R-S}(r) \wedge \forall v \in r(\exists u \in s(u = v[s] \wedge t = v[R-S]))\}$

IV. `SELECT R.a, R.b FROM R, S WHERE R.c = S.c`

Which of the above queries are equivalent?

(a) I and II (b) I and III
(c) II and IV (d) III and IV

(GATE 2009: 2 Marks)

Solution: I and II describe the division operator in relational algebra and tuple relational calculus, respectively.

Ans. (a)

Common Data Questions 40 and 41: Consider the following relational schema:

`Suppliers(sid: integer, sname: string, city: string, street: string)`

`Parts(pid: integer, pname: string, color: string)`

`Catalog(sid: integer, pid: integer, cost: real)`

40. Consider the following relational query on the above database:

```
SELECT S.sname
FROM Suppliers S
WHERE S.sid NOT IN (SELECT C.sid
        FROM Catalog C
        WHERE C.pid NOT IN (SELECT P.pid
                FROM Parts P
                WHERE P.color <> 'blue'))
```

Assume that relations corresponding to the above schema are not empty. Which one of the following is the correct interpretation of the above query?

(a) Find the names of all suppliers who have supplied a non-blue part.
(b) Find the names of all suppliers who have not supplied a non-blue part.
(c) Find the names of all suppliers who have supplied only blue parts.
(d) Find the names of all suppliers who have not supplied only blue parts.

(GATE 2009: 2 Marks)

Solution: The sub-query "`SELECT P.pid FROM Parts P WHERE P.color <> 'blue'`" returns pids of parts, which are not blue.

The sub-query "`SELECT C.sid FROM Catalog C WHERE C.pid NOT IN (SELECT P.pid FROM Parts P WHERE P.color <> 'blue')`" returns sid of all those suppliers who have supplied blue parts.

The complete query returns the names of all suppliers who have supplied a non-blue part.

Inner query `SELECT P. pid FROM Parts P WHERE P. color <> 'blue'` will select pid of non-blue parts. Outer query of this will produce sid of those suppliers whose pid is not in the result of inner query. The query will result the supplier names who have supplied non-blue parts

Ans. (a)

41. Assume that, in the suppliers relation above, each supplier and each street within a city has a unique name, and (sname, city) forms a candidate key. No other functional dependencies are implied other than those implied by primary and candidate keys. Which one of the following is TRUE about the above schema?

(a) The schema is in BCNF.
(b) The schema is in 3NF but not in BCNF.
(c) The schema is in 2NF but not in 3NF.
(d) The schema is not in 2NF.

(GATE 2009: 2 Marks)

Solution: Candidate key: sname, city
Primary key: sname
Alternate key: sid, sname

city → street
sname → city
Transitivity: sname → city → street
So, it is in 3NF.

Ans. (b)

42. Consider a B+ tree in which the maximum number of keys in a node is 5. What is the minimum number of keys in any non-root node?

(a) 1 (b) 2
(c) 3 (d) 4

(GATE 2010: 1 Mark)

Solution: In B+ tree, root node has minimum two block pointers and maximum p block pointer, where p = order and key = order − 1.

In the non-root node, the minimum number of keys $= p/2 - 1$

Here, key = 5, order = 6

So, minimum number of keys in non-root node $= 6/2 - 1 = 2$

Ans. (b)

43. A relational schema for a train reservation database is given below:

```
Passenger(pid, pname, age)
Reservation(pid, class, tid)
```

Table: Passenger

pid	pname	Age
0	'Sachin'	65
1	'Rahul'	66
2	'Saurav'	67
3	'Anil'	69

Table: Reservation

pid	Class	Tid
0	'AC'	8200
1	'AC'	8201
2	'SC'	8201
5	'AC'	8203
1	'SC'	8204
3	'AC'	8202

What pids are returned by the following SQL query for the above instance of the tables?

```
SELECT pid
FROM Reservation WHERE class = 'AC' AND
EXISTS(SELECT * FROM Passenger
  WHERE age > 65 AND Passenger.pid =
      Reservation.pid)
```

(a) 1, 0 (b) 1, 2
(c) 1, 3 (d) 1, 5

(GATE 2010: 1 Marks)

Solution: The outer query selects four entries (with pid as 0, 1, 5, 3) from Reservation table. Out of these selected entries, the sub-query returns non-null values only for 1 and 3.

Ans. (c)

44. Which of the following concurrency control protocols ensure both conflict serializability and freedom from deadlock?

I. Two-phase locking
II. Timestamp ordering

(a) I only (b) II only
(c) Both I and II (d) Neither I nor II

(GATE 2010: 1 Mark)

Solution: Two-phase locking is a concurrency control method, which guarantees serializability. The protocol utilizes locks, applied by a transaction to data, which may block other transactions from accessing the same data during the transaction's life. 2PL may be lead to deadlocks which result from the mutual blocking of two or more transactions.

Timestamp ordering algorithm is a non-lock concurrency control method. In Timestamp based method, deadlock cannot occur as no transaction ever waits.

Ans. (b)

45. Consider the following schedule for transactions T_1, T_2 and T_3:

T_1	T_2	T_3
Read(X)		
	Read(Y)	
		Read(Y)
	Write(Y)	
Write(X)		
		Write(X)
	Read(X)	
	Write(X)	

Which one of the schedules below is the correct serialization of the above?

(a) $T_1 \rightarrow T_3 \rightarrow T_2$ (b) $T_2 \rightarrow T_1 \rightarrow T_3$
(c) $T_2 \rightarrow T_3 \rightarrow T_1$ (d) $T_3 \rightarrow T_1 \rightarrow T_2$

(GATE 2010: 2 Marks)

Solution: T_1 can complete before T_2 and T_3 as there is no conflict between Write(X) of T_1 and the operations in T_2 and T_3 which occur before Write(X) of T_1.

T_3 can complete before T_2 as the Read(Y) of T_3 does not conflict with Read(Y) of T_2. Similarly, Write(X) of T_3 does not conflict with Read(Y) and Write(Y) operations of T_2.

After topologically sorting, the sequence is $T_1 \rightarrow T_3 \rightarrow T_2$.

Ans. (a)

46. The following functional dependencies hold for relations $R(A, B, C)$ and $S(B, D, E)$

$B \rightarrow A$,

$A \rightarrow C$

The relation R contains 200 tuples and the relation S contains 100 tuples. What is the maximum number of tuples possible in the natural join $R \bowtie S$?

(a) 100 (b) 200
(c) 300 (d) 2000

(GATE 2010: 2 Marks)

Solution: B is a candidate key of R.

So, all 200 values of B must be unique in R.

There is no functional dependency given for S.

For the maximum number of tuples in output, there are two ways.

(i) All 100 values of B in S are same and there is an entry in R, which matches with this value. So, we are having 100 tuples in output.
(ii) All 100 values of B in S are different and these values are present in R also. So, we are having 100 tuples.

Ans. (a)

47. Consider a relational table with a single record for each registered student with the following attributes:

I. Registration_Number: Unique registration number for each registered student.
II. UID: Unique Identity number, unique at the national level for each citizen.
III. BankAccount_Number: Unique account number at the bank. A student can have multiple accounts or joint accounts. This attribute stores the primary account number.
IV. Name: Name of the Student.
V. Hostel_Room: Room number of the hostel.

Which of the following options is INCORRECT?

(a) BankAccount_Number is a candidate key.
(b) Registration_Number can be a primary key.
(c) UID is a candidate key if all students are from the same country.
(d) If S is a superkey such that $S \cap$ UID is NULL then $S \cup$ UID is also a superkey.

(GATE 2011: 1 Mark)

Solution: When two students hold joint account in that case BankAccount_Num will not uniquely determine other attributes.

Ans. (a)

48. Database table by name Loan_Records is given below:

Borrower	Bank_Manager	Loan_Amount
Ramesh	Sunderajan	10000.00
Suresh	Ramgopal	5000.00
Mahesh	Sunderajan	7000.00

What is the output of the following SQL query?

```
SELECT count(*) FROM(
    (SELECT Borrower, Bank_Manager FROM
    Loan_Records) AS S NATURAL JOIN
    (SELECT Bank_Manager, Loan_Amount
    FROM Loan_Records) AS T );
```

(a) 3 (b) 9
(c) 5 (d) 6

(GATE 2011: 2 Marks)

Solution: Output as Table S:

Borrower	Bank_Manager
Ramesh	Sunderajan
Suresh	Ramgopal
Mahesh	Sunderajan

Output as Table T:

Bank_Manager	Loan_Amount
Sunderajan	10000.00
Ramgopal	5000.00
Sunderajan	7000.00

Natural Join of S and T:

Borrower	Bank_Manager	Loan_Amount
Ramesh	Sunderajan	10000.00
Ramesh	Sunderajan	7000.00
Suresh	Ramgopal	5000.00
Mahesh	Sunderajan	10000.00
Mahesh	Sunderajan	7000.00

Ans. (c)

49. Consider a database table T containing two columns X and Y each of type integer.

After the creation of the table, one record ($X = 1$, $Y = 1$) is inserted in the table.

Let MX and MY denote the respective maximum values of X and Y among all records in the table at any point in time. Using MX and MY, new records are inserted in the table 128 times with X and Y values being $MX + 1$, $2 * MY + 1$, respectively. It may be noted that each time after the insertion, values of MX and MY change.

What will be the output of the following SQL query after the steps mentioned above are carried out?

```
SELECT Y FROM T WHERE X = 7;
```

(a) 127 (b) 255
(c) 129 (d) 257

(GATE 2011: 2 Marks)

Solution:

The value of X is calculated using MX + 1 and Y is calculated using 2 × MY + 1

For example, X = 1 and Y = 1

at the next step, X = MX + 1 = 1 + 1 = 2, Y = 2 × MY + 1 = 2 × 1 + 1 = 3

X	Y
1	1
2	3
3	7
4	15
5	31
6	63
7	127

Ans. (a)

50. Which of the following statements are **TRUE** about an SQL query?

I. An SQL query can contain a HAVING clause even if it does not have a GROUP BY clause.
II. An SQL query can contain a HAVING clause only if it has GROUP BY clause.
III. All attributes used in the GROUP BY clause must appear in the SELECT clause.
IV. Not all attributes used in the GROUP BY clause need to appear in the SELECT clause.

(a) I and III (b) I and IV
(c) II and III (d) II and IV

(GATE 2012: 1 Mark)

Solution:

I. HAVING clause can also be used with aggregate function. If we use a HAVING clause without a GROUP BY clause, the HAVING condition applies to all rows that satisfy the search condition.

II. To verify S,

```
CREATE TABLE temp (id INT, name
VARCHAR(100));
INSERT INTO temp VALUES (1, "abc");
INSERT INTO temp VALUES (2, "abc");
INSERT INTO temp VALUES (3, "bcd");
INSERT INTO temp VALUES (4, "cde");
SELECT Count(*) FROM temp GROUP BY name;
```

Result:

```
count(*)
2
1
1
```

Ans. (b)

51. Given the basic ER and relational models, which of the following is **INCORRECT**?

(a) An attribute of an entity can have more than one value.
(b) An attribute of an entity can be composite.
(c) In a row of a relational table, an attribute can have more than one value.
(d) In a row of a relational table, an attribute can have exactly one value or a NULL value.

(GATE 2012: 1 Mark)

Solution: In relational table, more than one value for an attribute will violate the condition of first normal form. So, option (c) is incorrect.

Ans. (c)

52. Which of the following is **TRUE**?

(a) Every relation is 3NF is also in BCNF.
(b) A relation R is in 3NF if every non-prime attribute of R is fully functionally dependent on every key of R.
(c) Every relation in BCNF is also in 3NF.
(d) No relation can be in both BCNF and 3NF. (GATE 2012, 1 mark)

Solution: Any relation that is in 'x'NF, will automatically be in 'x-1' NF. Now, let us see the order of NF as we discussed in the text, it is – 1NF, 2NF, 3NF, BCNF, 4NF and 5NF. Therefore, option (c) is correct.

Ans. (c)

53. Suppose $R_1(\underline{A}, B)$ and $R_2(\underline{C}, D)$ are two relation schemas. Let r_1 and r_2 be the corresponding relation instances. B is a foreign key that refers to C in R_2. If data in r_1 and r_2 satisfy referential integrity constrains, which of the following is **ALWAYS TRUE**?

(a) $\Pi_B(r_1) - \Pi_C(r_2) = \varnothing$
(b) $\Pi_C(r_2) - \Pi_B(r_1) = \varnothing$
(c) $\Pi_B(r_1) - \Pi_C(r_2)$
(d) $\Pi_B(r_1) - \Pi_C(r_2) \neq \varnothing$

(GATE 2012: 2 Marks)

Solution: B is a foreign key in r_1, which refers to C in r_2.

r_1 and r_2 satisfy referential integrity constraints.

So, every value that exists in column B of r_1 must also exist in column C of r_2.

Ans. (a)

54. Consider the following transactions with data items P and Q initialized to zero:

$\boldsymbol{T}_1$: read (P);
read (Q);
if P=0 then Q:=Q+1;
write (Q).

$\boldsymbol{T}_2$: read (Q);
read (P)
if Q=0 then P:=P+1;
write (P).

Any non-serial interleaving of T_1 and T_2 for concurrent execution leads to

(a) a serializable schedule
(b) a schedule that is not conflict-serializable
(c) a conflict-serializable schedule
(d) a schedule for which precedence graph cannot be drawn

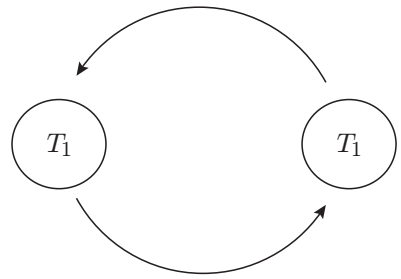
(GATE 2012: 2 Marks)

Solution: Two schedules are conflict-serializable if the actions belong to different transactions, one of the actions is a write operation, the actions access the same object (read or write).

Two schedules are conflict-equivalent if both schedules involve the same set of transactions, and the order of each pair of conflicting actions in both schedules is the same.

Take the following schedule:

S: $r_1(P)$; $r_1(Q)$; $r_2(Q)$; $r_2(P)$; $w_1(Q)$;$w_2(P)$



Cycle exists between two transactions, so T_1 and T_2 are not conflict serializable.

A schedule is conflict-serializable when the schedule is conflict-equivalent to one or more serial schedules. There are two possible serial schedules—T_1 followed by T_2 and T_2 followed by T_1. In both, one of the transactions reads the value written by another transaction as a first step.

Ans. (b)

Common Data Questions 55 and 56: Consider the following relations A, B and C:

A

Id	Name	Age
12	Arun	60
15	Shreya	24
99	Rohit	11

B

Id	Name	Age
15	Shreya	24
25	Hari	40
98	Rohit	20
99	Rohit	11

C

Id	Phone	Area
10	220	02
99	2100	01

55. How many tuples does the result of the following relational algebra expression contain? Assume that the schema of $A \cup B$ is the same as that of A.

$$(A \cup B) \bowtie_{\text{A.Id}>40 \bowtie \text{C.Id}<15} \text{C}$$

(a) 7 (b) 4
(c) 5 (d) 9

(GATE 2012: 2 Marks)

Solution:

$A \cup B$

Id	Name	Age
12	Arun	60
15	Shreya	24
99	Rohit	11
25	Hari	40
98	Rohit	20

$$(A \cup B) \bowtie_{\text{A.Id}>40 \bowtie \text{C.Id}<15} \text{C}$$

Id	Name	Age	Id	Phone	Area
12	Arun	60	10	2200	02
15	Shreya	24	10	2200	02
99	Rohit	11	10	2200	02
25	Hari	40	10	2200	02
98	Rohit	20	10	2200	02
99	Rohit	11	99	2100	01
98	Rohit	20	99	2100	01

Ans. (a)

56. How many tuples does the result of the following SQL query contain?

```
SELECT A.Id
FROM A
WHERE A.Age > ALL (SELECT B.Age
                   FROM B
                   WHERE B.Name = 'Arun')
```

(a) 4 (b) 3 (c) 0 (d) 1

(GATE 2012: 2 Marks)

Solution: ALL returns the values to be greater than all the values returned by the sub-query.
There is no entry with the name Arun in Table B. So, the sub-query will return NULL. If a sub-query returns NULL, then the condition becomes true for all rows of A.

Ans. (b)

57. An index is clustered, if

(a) It is on a set of fields that form a candidate key.
(b) It is on a set of fields that include the primary key.
(c) The data records of the file are organized in the same order as the data entries of the index.
(d) The data records of the file are organized not in the same order as the data entries of the index.

(GATE 2013: 1 Mark)

Solution: In clustered index, the leaf nodes contain the data page of the table concerned. Each index row contains a key value and a pointer to either the intermediate level (other than root and leaf nodes) or data row in leaf level. Therefore, they are organised in the same order.

Ans. (c)

58. Consider the following relational schema:

```
Students(rollno: integer, sname: string)
Courses(courseno: integer, cname: string)
Registration(rollno: integer, courseno; integer, percent: real)
```

Which of the following queries are equivalent to this query in English?

"Find the distinct names of all students who score more than 90% in the course numbered 107"

I. `SELECT DISTINCT S.sname FROM Students as S, Registration as R WHERE R.rollno=S.rollno AND R.Courseno=107 AND R.percent>90`

II. $\Pi_{\text{sname}}(\sigma_{\text{courseno}=107} \Lambda \text{ percent} > 90(\text{Registration students}))$

III. $\{T | \exists S \in \text{Student}, \exists \text{R} \in \text{Registration } (\text{S.rollno} = \text{R.rollno} \wedge \text{R.courseno} = 107 \wedge R.\text{percent} > 90 \wedge T.\text{ sname} = S.\text{name})\}$

$\{<\text{S}_N> \mid \exists S_R \exists R_P(<S_R, S_N> \in \text{Student} \wedge < S_R, 107, R_P> \in R_P > 90)\}$

(a) I, II, III and IV (b) I, II and III only
(c) I, II and IV only (d) II, III and IV only

(GATE 2013: 2 Marks)

Solution: All of the query statements are equivalent to the given query.

Ans. (a)

Linked Answer Questions 59 and 60: Relation R has eight attributes $ABCDEFGH$. Fields of R contain only atomic values. $F = \{CH \rightarrow G, A \rightarrow BC, B \rightarrow CFH, E \rightarrow A, F \rightarrow EG\}$ is a set of functional dependencies (FDs) so that $F+$ is exactly the set of FDs that hold for R.

59. How many candidate keys does the relation R have?

(a) 3 (b) 4
(c) 5 (d) 6

(GATE 2013: 2 Marks)

Solution: Candidate keys AD, BD, ED and FD

Ans. (b)

60. The relation R is

(a) In 1NF, but not in 2NF
(b) In 2NF, but not in 3NF
(c) In 3NF, but not in BCNF
(d) In BCNF

(GATE 2013: 2 Marks)

Solution: $A \rightarrow BC$, $B \rightarrow CFH$ and $F \rightarrow EG$ are partial dependencies.

So, they are in 1NF but not in 2NF.

Ans. (a)

PRACTICE EXERCISES

Set 1

1. A relational database can be defined as

(a) a place to store relational information.
(b) a database that relates to different operations.
(c) a database to relate the relationship of different databases.
(d) a database related to storing of information about humans.

2. The minimal superkeys are called

(a) minimum keys. (b) candidate keys.
(c) domain keys. (d) optimal keys.

3. Which of the following combinations represents a valid data model?

(a) E-R model, network data model, normal form data model
(b) Relational data model, hierarchical data model, SQL data model
(c) Object-based data model, table data model
(d) E-R model, relational data model, object-based data model

4. Which of the following is not a database model?

(a) Network database model
(b) Relational database model
(c) Object-oriented database model
(d) Normal form data model

5. Which one of the following allows accessing or maintaining data in a database?

(a) DCL (b) DML
(c) DDL (d) None of the above

6. Which of the following is a binary operation?

(a) Select (b) Project
(c) Rename (d) Union

7. Which of the following is a unary operation?

(a) Join (b) Project
(c) Intersection (d) Cartesian product

8. What is the least and ceiling number of rows in the join selection of two tables having m rows and n rows, respectively?

(a) 1 and $m \times n$ (b) 0 and $m + n$
(c) 1 and infinite (d) 0 and mn

9. An attribute of one table having the same value of a primary key of another table is termed as

(a) foreign key. (b) alternate key.
(c) candidate key. (d) transitive key.

10. A database has five attributes defined as $\{A, B, C, D, E\}$. The functional dependency is defined by

$\{A \rightarrow BC, E \rightarrow FG\}$

The candidate key of this table is

(a) A. (b) AE.
(c) E. (d) BC and FG.

11. Consider the following set of functional dependency

$\{AB \rightarrow CD, CD \rightarrow E, E \rightarrow A\}$ AB, CD are candidate keys.

It is in which normal form?

(a) 2NF (b) BCNF (c) 3NF (d) 4NF

12. The term physical data independence can be defined as

(a) If we modify the conceptual schema, then there is no need to modify the physical schema.
(b) If we modify the physical schema, then there is no need to modify the conceptual schema.

(c) If we modify the conceptual schema, then there is no need to modify the external schema.
(d) If we modify the middle level schema, then any one schema (internal/external) may be modified.

13. Relational algebra can be defined as

(a) Procedural. (b) Non-procedural.
(c) User dependent. (d) Domain dependent.

14. Which of the following is true?

(a) A relation in BCNF is always in 3NF.
(b) A relation in 3NF is always in BCNF.
(c) PJNF and 3NF are same.
(d) A relation in PJNF is not in 3NF.

15. RAD stands for _____.

(a) Read and destroy
(b) Read and develop
(c) Rapid application development
(d) Rapid application design

16. In DML, RECONNCT command cannot be used with

(a) CLEAN set. (b) FIXED set.
(c) OPTIONAL set. (d) USER set.

17. The UWA is used to communicate the contents of individual records between

(a) Host program and present record.
(b) Host program and host record.
(c) Host program and DBMS.
(d) Host program and host language.

18. Referential integrity is concerned with

(a) Primary key. (b) Foreign key.
(c) Alternate key. (d) Project_Join key.

19. Match the following:

List – I	List – II
I. DDL	A. LOCK TABLE
II. DML	B. COMMIT
III. TCL	C. Natural Difference
IV. BINARY Operation	D. REVOKE

Codes:

	I	II	III	IV
(a)	A	D	B	C
(b)	A	B	D	C
(c)	D	B	A	C
(d)	D	A	B	C

20. The value of particular field should be less than 50 if it is

(a) An integrity constraint.
(b) A referential constraint.
(c) A primary key constraint.
(d) A normalization constraint.

21. The ER model of a database includes

I. Different entity types.
II. Attributes for each entity type.
III. Relationships among entity types.
IV. Semantic integrity constraints are not defined.

Which of the above statements is correct?

(a) All of the above (b) Only I, II and III
(c) III and IV (d) II and III

22. The cardinality ratio and participation _____ of a table helps us to implement either the cross referencing design or mutual referencing design.

(a) Functionality (b) Constraints
(c) SQL queries (d) Domains

23. Consider the following schemas:

```
Branch = (Branch-name, Assets, Branch-city)
Customer = (Customer-name, Bank name,
            Customer-city)
Borrow = (Branch-name, loan number,
          customer account-number)
Deposit = (Branch-name, Accountnumber,
           Customer-name, Balance)
```

Using relational algebra, the query that finds customers having deposits have accountnumber > 23456 is

(a) $\Pi_{\text{customer-name}}(\sigma_{\text{account-number}} > 23456\ (\text{Deposit})$
(b) $\sigma_{\text{customer-name}}(\sigma_{\text{account-number}} > 23456\ (\text{Deposit})$
(c) $\Pi_{\text{customer-name}}(\sigma_{\text{account-number}} > 23456\ (\text{Borrow})$
(d) $\sigma_{\text{customer-name}}(\Pi_{\text{account-number}} > 23456\ (\text{Borrow})$

24. What deletes the entire file except the file structure?

(a) DELETE
(b) DELETE RECORDS
(c) ZAP
(d) RETAIN STRUCT

25. The function of a Transaction Manager is to

I. Maintain a log of transactions.
II. Maintain database images before and after a transaction.
III. Maintain appropriate concurrency control.
IV. Avoid deadlocks.

(a) I, II, III and IV (b) II and III
(c) I, III and IV (d) I, II and III

26. The term granularity relates to

(a) the size of table.
(b) the size of data item.
(c) the size of tuple.
(d) the size of primary key

27. Match the following:

I. OLAP	A. Back propagation
II. OLTP	B. Data warehouse
III. Decision tree	C. RDBMS
IV. Neural network	D. Classification

	I	II	III	IV
(a)	D	C	A	B
(b)	B	C	D	A
(c)	A	B	C	D
(d)	D	B	A	C

28. Which of the following statements is TRUE?

(a) A prime attribute can be transitively dependent on a key in 5NF relation.
(b) A prime attribute can be transitively dependent on a key in 3NF relation.
(c) A prime attribute can be transitively dependent on a key in PJNF relation.
(d) A prime attribute cannot be transitively dependent on a key in BCNF relation.

29. Which normal form is considered as adequate for a simple database design?

(a) 2NF (b) 3NF
(c) BCNF (d) PJNF

30. Which of the following is not a type of DBMS?

(a) Hierarchical (b) Network
(c) Relational (d) Graphical

31. Manager's salary details are to be hidden from Employee Table. This technique is called as

(a) internal level datahiding.
(b) physical level datahiding.
(c) external level datahiding.
(d) logical level datahiding.

32. A network schema is used to

(a) restrict to many-to-many relationship.
(b) permit many-to-many relationship.
(c) permit to store data in a database.
(d) help in storing one-to-many relationship.

33. Storing under unified scheme at a single site is called as

(a) data mining. (b) universal database.
(c) data warehousing. (d) advanced database.

34. The task of correcting and cleaning data is called as

(a) data organization. (b) pre-processing.
(c) data mining. (d) data structuring.

35. A clustering index consists of _____.

(a) declared and ordered primary key
(b) both grouped and ordered primary key and foreign key
(c) ordered foreign key
(d) grouped and ordered candidate key and alternate key

36. Consider the following schema:

```
Emp(Empcode, Name, Sex, Salary, Deptt)
```

A simple SQL query is executed as follows:

```
SELECT Deptt FROM Emp
WHERE sex = 'M'
GROUP by Dept
HAVING avg (Salary) > {select avg (Salary)
from Emp}
```

The output will be

(a) average salary of male employee is the average salary of the organization.
(b) average salary of male employee is less than the average salary of the organization.
(c) average salary of male employee is equal to the average salary of the organization.
(d) average salary of male employees in a department is more than the average salary of the organization.

37. Consider a relation $X(p, q, r)$, and the domains are represented by their atomic values. Further, the functional dependency $p \rightarrow q$, $q \rightarrow r$ is observed, the relation is in

(a) 1NF, not in 2NF
(b) 2NF, not in 3NF
(c) 3NF
(d) 1NF, not in 3NF

38. Match the following:

I. Foreign keys	A. Domain constraint
II. Private key	B. Referential integrity
III. Event control action model	C. Password
IV. Data security	D. Trigger

Codes:

	I	II	III	IV
(a)	C	B	A	D
(b)	B	A	D	C
(c)	C	D	A	B
(d)	A	B	C	D

Set 2

1. For a database relation $R(a, b, c, d)$, where the domains of a, b, c, d include only atomic values, only the following functional dependencies and those that can be inferred from them holds.

$a \rightarrow c$

$b \rightarrow d$

This relation is

(a) In First normal form but not in Second normal form.
(b) In Second normal form but not in Third normal form.
(c) In Third normal form.
(d) None of the above.

2. Let $R(a, b, c)$ and $S(d, e, f)$ be two relations in which d is the foreign key of S that refers to the primary key of R. Consider the following four operations on R and S:

I. Insert into R II. Insert into S
III. Delete from R IV. Delete from S

Which of the following is true about the referential integrity constraint above?

(a) None of I, II and IV can cause its violation.
(b) All of I, II, III and IV can cause its violation.
(c) Both I and IV can cause its violation.
(d) Both II and III can cause its violation.

3. There are five records in a database

Name	Age	Occupation	Category
Rama	27	CON	A
Abdul	22	ENG	A
Jenifer	28	DOC	B
Maya	32	SER	D
Dev	24	MUS	C

There is an index file associated with this and it contains the values 1, 3, 2, 5 and 4. Which one of the fields in the index built from?

(a) Age (b) Name
(c) Occupation (d) Category

4. Consider the following database relations containing the attributes

`Book_id`

`Subject_Category_of_books`

`Name_of_Author`

`Nationalilty_of_Author`

With Book_id as the primary key

(a) The highest normal form satisfied by this relation is — NF.
(b) Suppose the attributes Book_title and Author_address are added to the relation, and the primary key is changed to {name_of_Author, Book_title}, the highest normal form satisfied by the relation is — NF.

5. Consider the scheme $R = (S\ T\ U\ V)$ and the dependencies $S \rightarrow T$, $T \rightarrow U$, $U \rightarrow V$ and $V \rightarrow S$. Let $R = (R_1$ and $R_2)$ be a decomposition such that $R_1 \cap R_2 = \phi$. The decomposition is

(a) Not in 2NF
(b) In 2NF but not 3NF.
(c) In 3NF but not 2NF.
(d) In both 2NF and 3NF.

6. Consider a schema $R(A, B, C, D)$ and functional dependencies $A \rightarrow B$ and $C \rightarrow D$. Then the decomposition of R into $R_1(AB)$ and $R_2(CD)$ is

(a) Dependency preserving and lossless join.
(b) Lossless join but not dependency preserving.
(c) Dependency preserving but not lossless join.
(d) Not dependency preserving and not lossless join.

7. Relation R with an associated set of functional dependencies, F, is decomposed into BCNF. The redundancy (arising out of functional dependencies) in the resulting set of relation is

(a) Zero.
(b) More than zero but less than that of an equivalent 3NF decomposition.
(c) Proportional to the size of $F+$.
(d) Indeterminate.

8. From the following instance of a relation scheme $R(A, B, C)$, we can conclude that

A	B	C
1	1	1
1	1	0
2	3	2
2	3	2

(a) A functionally determines B and B functionally determines C.
(b) A functionally determines B and B does not functionally determine C.
(c) B does not functionally determine C.
(d) A does not functionally determine B and B does not functionally determine C.

9. Which of the following is/are correct?

(a) An SQL query automatically eliminates duplicates.
(b) An SQL query will not work if there are no indexes on the relations.
(c) SQL permits attribute names to be repeated in the same relation.
(d) None of the above.

10. Given relation $r(w, x)$ and $s(y, z)$, the result of select distinct w, x from r, s is guaranteed to be same as r, provided

(a) r has no duplicates and s is non-empty.
(b) r and s have no duplicates.
(c) s has no duplicates and r is non-empty.
(d) r and s have the same number of tuples.

11. Which of the following query transformations (i.e. replacing the LHS expression by the RHS expression) is incorrect? R_1 and R_2 are relations, C_1 and C_2 are selection conditions and A_1 and A_2 are attributes of R_1.

(a) $\sigma_{c_1}(\sigma_{c_2}(R_1)) \rightarrow \sigma_{c_2}(\sigma_{c_1}(R_1))$
(b) $\sigma_{c_1}(\Pi_{A_1}(R_1)) \rightarrow \Pi_{A_1}(\sigma_{c_1}(R_1))$
(c) $\sigma_{c_1}(R_1 \mathrm{E} R_2) \rightarrow \sigma_{c_1}(R_1) \mathrm{E}\, \sigma_{c_1}(R_1)$
(d) $\Pi_{A_2}(\sigma_{c_1}(R_1)) \rightarrow \sigma_{c_1}(\Pi_{A_2}(R_1))$

12. The relational algebra expression equivalent to the following tuple calculus expression:

$\{t | t \in r \wedge (t[A] = 10 \wedge t[B] = 20)\}$ is

(a) $\sigma_{(A = 10 \vee B = 20)}(r)$
(b) $\sigma_{(A = 10}(r)$ E $\sigma_{B = 20}(r)$
(c) $\sigma_{(A = 10}(r) \cap \sigma_{B = 20}(r)$
(d) $\sigma_{(A = 10}(r) - \sigma_{B = 20}(r)$

13. Which of the following relational calculus expressions is not safe?

(a) $\{t | \exists \in R_1(t[A] = u[A]) \wedge \neg \exists s \in R_2(t[A] = s[A])\}$
(b) $\{t | \forall u \in R_1(u[A] = \text{“}x\text{”} \Rightarrow \exists s \in R_2\ (t[A] = s[A] \wedge s[A] = u[A]))\}$
(c) $\{t | \neg\ (t \in R_1)$
(d) $\{t | \exists \in R_1(t[A] = u[A]) \wedge \exists s \in R_2(t[A] = s[A])\}$

14. With regard to the expressive power of the formal relational query languages, which of the following statement is true?

(a) Relational algebra is more powerful than relational calculus.
(b) Relational algebra has the same power as relational calculus.
(c) Relational algebra has the same power as safe relational calculus.
(d) None of the above.

15. For the schedule given below, which of the following is correct?

1 Read A
2 Read B
3 Write A
4 Read A
5 Write A
6 Write B
7 Read B
8 Write B

(a) This schedule is serializable and can occur in a scheme using 2PL protocol.
(b) This schedule is serialzable but cannot occur in a scheme using 2PL protocol.
(c) This schedule is not serializable but can occur in a scheme using 2PL protocol.
(d) This schedule is not serializable and cannot occur in a scheme using 2PL protocol.

16. Which of the following is correct?

(a) B trees are for storing data on disk and B+ trees are for main memory.
(b) Range queries are faster on B+ trees.
(c) B trees are for primary indexes and B+ trees are for secondary indexes.
(d) The height of a B+ tree is independent of the number of records.

17. B+ trees are preferred to binary trees in database because

(a) disk capacities are greater than memory capacities.

(b) disk access is much slower than memory access.
(c) disk data transfer rates are much less than memory data transfer rates.
(d) disks are more reliable than memory.

18. A B+ tree index is to be built on the name attribute of the relation STUDENT. Assume that all student names are of length 8 bytes, disk blocks are of size 512 bytes and index pointers are of size 4 bytes. Given this scenario, what would be the best choice of the degree (i.e., the number of pointers per node of the B+ tree?

ANSWERS TO PRACTICE EXERCISES

Set 1

1. (a)	**8.** (d)	**15.** (c)	**22.** (b)	**29.** (b)	**36.** (d)
2. (b)	**9.** (a)	**16.** (b)	**23.** (a)	**30.** (d)	**37.** (a)
3. (d)	**10.** (b)	**17.** (c)	**24.** (c)	**31.** (c)	**38.** (b)
4. (d)	**11.** (c)	**18.** (b)	**25.** (d)	**32.** (b)	
5. (a)	**12.** (b)	**19.** (d)	**26.** (b)	**33.** (c)	
6. (d)	**13.** (a)	**20.** (a)	**27.** (b)	**34.** (b)	
7. (b)	**14.** (a)	**21.** (a)	**28.** (d)	**35.** (a)	

Set 2

1. (a) The relation has the following functional dependencies:

$a \rightarrow c$
$b \rightarrow d$

$\{a\ b\}$ is key for the given relation.

Here prime attributes are a and b.

c and d are dependent on partial keys, this is not allowed in 2NF. So, the given relation is in 1 NF.

2. (d) Insertion into S and deletion from R can cause inconsistency. So, II and III can cause violation.

3. (c) The index is built from field occupation where column is sorted in alphabetic order CON, DOC, ENG, SER and MUS. Index is assigned according to alphabetic position.

4. (a) 2NF, (b) 3NF

5. (d) The relation schema $R = (S\ T\ U\ V)$ has following functional dependencies

$S \rightarrow T$

$T \rightarrow U$

$U \rightarrow V$

$V \rightarrow S$

Candidate keys for this relation are $\{S, T, U, V\}$

As given $R_1 \cap R_2 = \phi$

So, R_1 and R_2 might have two attributes. A relation with two attributes satisfies both 2NF and 3NF conditions.

6. (c) Dependency is preserved because there is no loss of functional dependencies.
Since $R_1(AB)$ and $R_2(CD)$ do not have any common attribute. So, it is lossy join.

7. (a) Redundancy is zero when we decompose into BCNF.

8. (b, c) F: $X \rightarrow Y$ implies that for any two tuples if $t_1[X] = t_2[X]$, they must have $t_1[Y] = t_2[Y]$.

9. (d) DISTINCT key word is used to remove duplicate rows along with SELECT keyword. SQL does not permit attribute names to be repeated in the same relation.

10. (a) Cross join with an empty relation produces the overall result as empty. Therefore, S is not empty. 'r' should not have any duplicates to have same rows as 'r'.

11. (a) $\sigma_{c_1}(\sigma_{c_1}(R))$ and $(\sigma_{c_2}(\sigma_{c_1}(R)))$ will not be equivalent if conditions c_1 and c_2 are not equivalent.

12. (c) Tuple calculus expression select tuples where $A = 10$ and $B = 20$ of relation R. So, the equivalent relational algebra expression is $\sigma_{(A\ =\ 10}(r) \cap \sigma_{B\ =\ 20}(r)$.

13. (c) The query $-t|\neg\ (t \in\ R_1)$ is syntactically correct. But it requests for all tuples that are not in R_1. That set of such t tuples is obviously infinite, in the context of such as the infinite domain set of all integers. Therefore, this is unsafe query.

14. (c) Both relational algebra and relational calculus has same expressive powers. Every query written in relation algebra can be expressed in relational calculus.

15. (d) Given schedule has $W_1(A)$, $R_2(A)$ and $W_2(B)$, $R_1(B)$ conflicting pairs. So the schedule is not serializable. Hence, it cannot occur in scheme using 2PL protocol.

16. (b) Most database systems use indexes built on some form of a B+ tree due to its many advantages, in particular its support for range queries. Leaf nodes are linked together in B+ trees, hence range queries are faster.

17. (b) In B+ trees, disk access is slow and numbers of hits are less. B+ trees have very high fan-out (typically on the order of 100 or more), which reduces the number of I/O operations required to find an element in the tree.

18. (43) Let n be the degree.

Given k key size (length of the name = 8 byte attribute of student)

Disk block size, $B = 512$ bytes

Index pointer size, $b = 4$ bytes

Degree of B+ tree can be calculated if we know the maximum number of key an internal node can have; the formula for that is

$(n - 1)k + n \times b =$ Block size

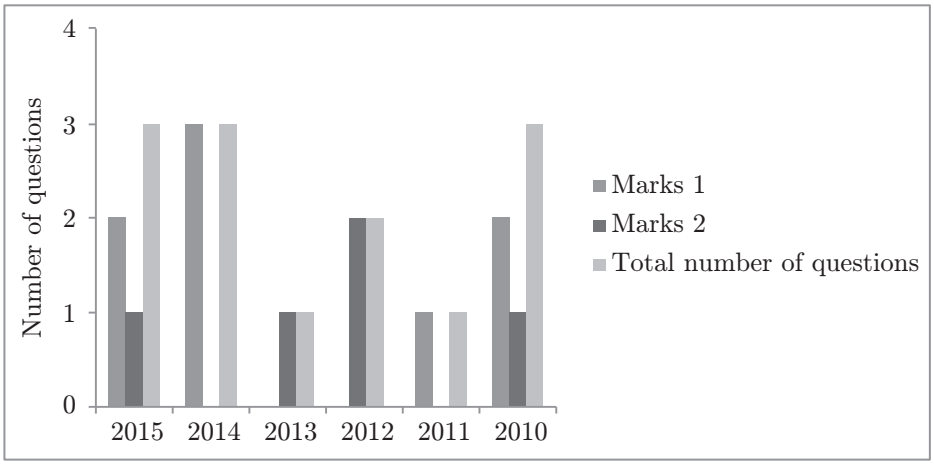
$(n - 1)8 + n \times 4 = 512$

$12n = 520$

$n = 520/12 = 43$

UNIT IX : INFORMATION SYSTEMS AND SOFTWARE ENGINEERING

LAST SIX YEARS' GATE ANALYSIS



Concepts on which questions were asked in the previous six years

Year	Concept
2015	Function point metric, Seeded faults, McCabe's complexity, COCOMO Model, SRS document, Code walkthrough and inception, Testing
2014	Cohesion and coupling
2013	Cohesion and coupling
2012	IEEE floating-point representation
2011	Lines of code, COCOMO model, SRS, black-box testing
2010	Software development life cycle

CHAPTER 9

INFORMATION SYSTEMS AND SOFTWARE ENGINEERING

Syllabus: Information gathering, Requirement and feasibility analysis, Data flow diagrams, Process specifications, Input/output design, Process life cycle, Planning and managing the project, Design, Coding, Testing, Implementation, Maintenance.

9.1 INTRODUCTION

Software engineering is the study of design, development and maintenance of software. Real-time problems are so large and complex that they cannot be solved by single developer. Software engineering defines teams and its quality. It is concerned about all the aspects of software development from feasibility of software to its maintenance. The team in software engineering consists of developers, testers, designers, customers, managers, etc. Every team member works to fulfil task assigned to him.

9.2 INFORMATION SYSTEMS

Information system consists of components such as storing and processing data. All business firms and organisations rely on them to carry out and manage their operations, interact with their customers and suppliers, and compete in the market. For example, these organisations could reach their customers through Internet for the processing of financial accounts and manage their human resources. Other users can rely on information system through online services such as social networking, auctions,

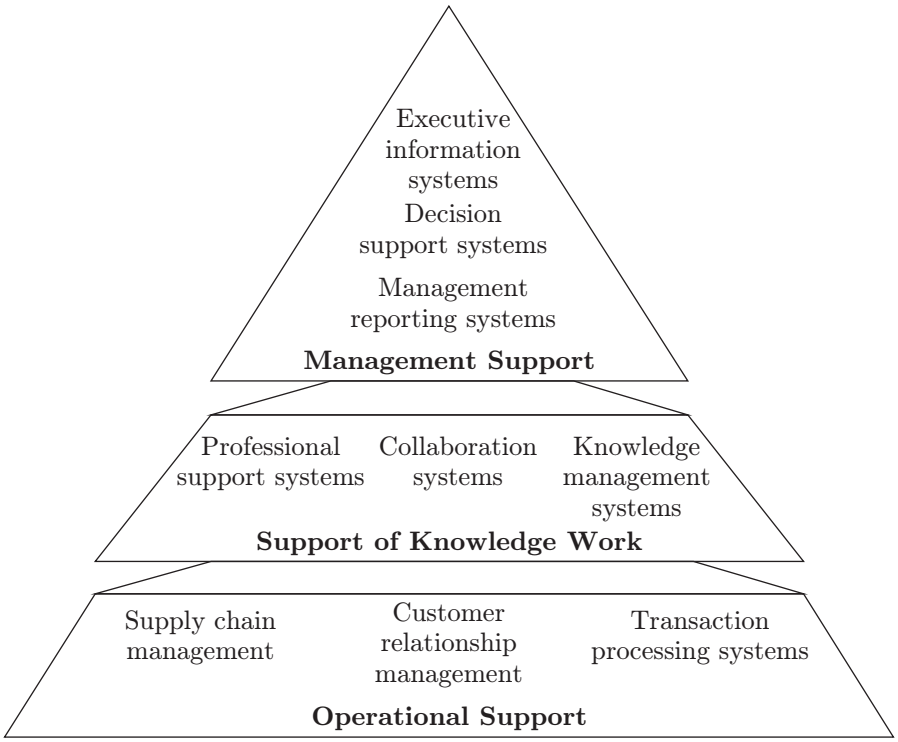


Figure 9.1 | An information system.

banking, entertainment, shopping, etc. Information and knowledge have become vital economic resources. Information systems support operations, knowledge work and management in organisations (Fig. 9.1).

9.2.1 Components of Information Systems

The main components of information systems are computer hardware and software, telecommunications, databases and data warehouses, human resources and procedures (Fig. 9.2). The hardware, software and telecommunications constitute information technology (IT), which is now ingrained in the operations and management of organisations.

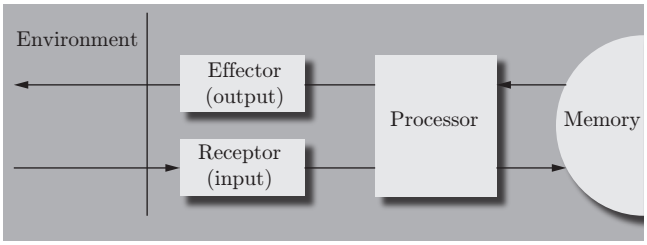


Figure 9.2 | Structure of an information system.

9.2.2 Types of Information Systems

9.2.2.1 Office Information System

Office information system (OIS) uses hardware, software and network to enhance the workflow and facilitate communications among employees. OIS is also known as office automation. With the help of OIS, employees or users can perform tasks electronically using computers and other electronic devices, instead of manually processing data. For example, a registration department can post the class schedule via email to students, whereas in a manual system, the registration department would have to photocopy the schedule and mail it to each student's house.

OIS supports a large range of business office activities such as creating and distributing graphics and/or documents, sending messages, scheduling and accounting. All levels of users utilise and benefit from the features of an OIS.

The software used in OIS includes word processing, spreadsheets, databases, presentation graphics, email, Web browsers, Web-page authoring, personal information management and groupware. OIS uses communication technology such as voice mail, fax, video conferencing and electronic data interchange (EDI) for the electronic exchange of text, graphics, audio and video. The hardware includes computers, modems, video cameras, speakers and microphones, scanners and fax machines.

9.2.2.2 Transaction Processing Systems

A transaction processing system (TPS) captures and processes data generated during an organisation's day-to-day

transactions. A transaction is a business activity such as a deposit, payment, order, reservation, etc.

Transaction processing includes the following activities:

1. Recording a business activity such as registration, order, payment, etc.
2. Confirming an action or triggering a response such as printing documents, sending a response, generating pay cheque, issuing receipts, etc.
3. Maintaining data such as adding new data, modifying data or deleting data.

For the processing of business data, TPS was the first computerised system developed as a function called data processing. It is an existing manual system, which allows faster processing, reduced clerical costs and improved customer service.

Online transaction processing (OLTP) is an example of TPS. The administrative users use OLTP to enter data and take print out of the entered data in the form of receipts.

9.2.2.3 Management Information Systems

Management information system (MIS) generates accurate, timely and organised information, and based on that user can make decisions, solve problems, supervise activities and track progress. MIS generates reports on a regular basis, a management information system called management reporting system (MRS). MIS interacts with TPS. On the basis of activities performed by TPS, MIS generates reports of daily activities. The three basic types of information it produces is as follows:

1. **Detailed information:** It confirms transaction processing activities, for example, detailed order report.
2. **Summary information:** It consolidates data into a format, which can be reviewed quickly and easily by a user, for example, an inventory summary report, which includes totals, tables or graphs.
3. **Exception information:** It filters data to report information, which is outside of a normal condition called the exception criteria. An exception criterion is defined as the range of what is considered normal activity or status; for example, exception report, which notifies the items it needs to reorder. It helps in saving time.

9.2.2.4 Decision Support Systems

Decision support system (DSS) is a computer-based information system designed to support decision-making activities such as business or organisation. A variety of DSSs exist having support of wide range of decisions. It uses data from two sources: internal and external.

1. **Internal sources of data:** A data from an organisation such as manufacturing, inventory, sales or financial data.
2. **External source of data:** A data from other organisations such as population trends, interest rates, construction rates, price of raw material, etc.

DSS may also include query language, statistical analysis capabilities, spreadsheets and graphics for the extraction and evaluation of data, if it is very large. DSS includes some kind of capabilities for the creation of a model of the factors affecting the decision. In these models the user can change the values of all or few parameters and project the results. There are lots of application software packages which are used for the decision-making process such as executive information system (EIS), which supports the information needs of executive management. EIS uses data in the form of charts or tabular forms to show trends, ratios and other managerial statistics.

9.2.2.5 Expert Systems

An **expert system** is an information system, which is responsible for capturing and storing human knowledge and imitating human reasoning and decision-making processes. It consists of two main components: knowledge base and inference rules.

1. **Knowledge base:** It is a combination of subject knowledge and experiences of human experts.
2. **Inference rules:** It is a set of logical judgement applied to knowledge base when a user describes a situation to the expert system.

It is used in decision-making at any level in an organisation. It is also used to resolve situations such as diagnosing illnesses, searching for oil, making soup, etc.

It is a part of an artificial intelligence. Artificial intelligence (AI) is an application of human intelligence to computers. It senses human actions based on logical assumptions and prior experience, then take appropriate action to complete the task. It can perform various functions such as speech recognition, logical reasoning and creative responses.

9.2.2.6 Integrated Information Systems

As technology widens its horizons, it becomes difficult to classify a system belonging uniquely to one of the above five types of information system. Today's applications supporting processing of transactions, management of information and decision-making need an integrated information system.

Types of Information Systems

The following are six major types of information systems corresponding to each organisational level (the four levels shown in Fig. 9.1):

1. Transaction processing systems (TPS) to serve the operational level of an organisation.
2. Knowledge work systems (KWS) to keep an organization up-to-date in knowledge in terms of technology, science, art and social thoughts.
3. Office automation systems (OAS) to serve the knowledge level of an organisation.
4. Decision-support systems (DSS) to analyze the existing information to forecast the effects of their decisions in the near future.
5. Management information systems (MIS) to serve the management level of the organisation.
6. Executive support systems (ESS) to serve the strategic level of an organisation.

9.3 SOFTWARE

Software is a program, and when it is executed the desired functionality and performance must be satisfied. Software is a data structure used to manipulate the information. It is a document that describes the operation and use of a program. Finally, it is a logical entity rather than physical entity.

9.3.1 Characteristics of Software

The following are the characteristics of a software:

1. Software is developed or engineered, but not manufactured in classical sense. Although the development approach for both the hardware design and software design is same, but after implementation of the hardware design we get the physical component and after implementation of the software design we get the logical component. In both cases quality is an important factor, that means when the output is operational the associated functionality must be satisfied.
2. Software does not wear out, but deteriorates. In hardware design, at early development, the failure rate is high due to undiscovered errors. After correcting the defects, the failure rate is decreased and the component starts running at a steady state. However, over a period of time, due to external conditions, such as temperature, air, dust and environmental maladies, the failure rate again increases and the hardware starts to wear out. When it undergoes wear out, we can replace the component with a new one. In software design, at early development, the failure rate is high due to undiscovered errors. After correcting the defects the failure rate is decreased and the software starts running within the constrained level. After a period of time, another change is injected in the existing software due to which the failure rate again increases, and after correction the failure rate decreases. So, during the lifetime of software, when the requirements keep changing, the software undergoes deterioration (Fig. 9.3).
3. The industry is moving towards component-based development, but software is still being custom-built. As components are error-free codes, they are reusable. With component-based development for developing some hardware, time and development cost are reduced as the design consists of IC numbers which are already available. So, at the time of implementation we can directly use them. For softwares also, we should use reusable components.

Software development also uses component names such as:

1. **Off the shelf:** This component is used in the project from the third party, that is, the component is not available in the library of the developing organisation.
2. **Fully experienced and partially experienced:** This component is derived based on the previous project, so can be directly used in the current project.
3. **New component:** This component is developed from the base line, means from the initial stage or scratch.

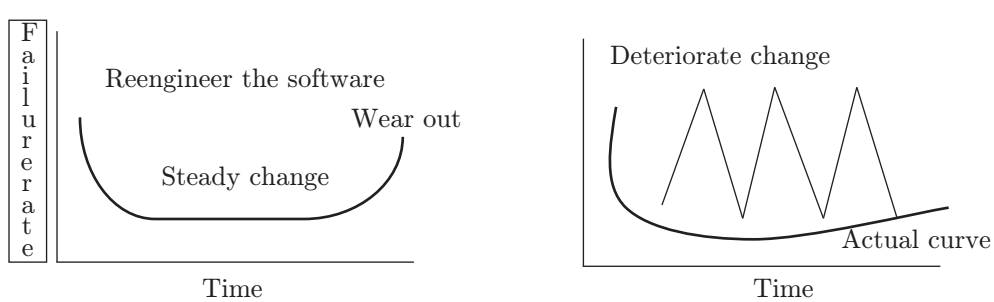


Figure 9.3 | Wear out and deterioration of software.

9.3.2 Software Engineering

Software engineering is the systematic, disciplined and quantitative approach for the development, operation and maintenance of the software.

Software engineering is described in the following two approaches:

1. **Layered approach:** The layered approach consists of the following four layers (Fig. 9.4):
   - ***Quality:*** Confirmation to explicitly stated form as mentioned in the document.
   - ***KPA:*** Key process areas such as planning, schedule, performance and quality attribute.
   - ***Methods:*** "How to" process model.
   - ***Tools:*** Automated and semi-automated tool available to develop the software.

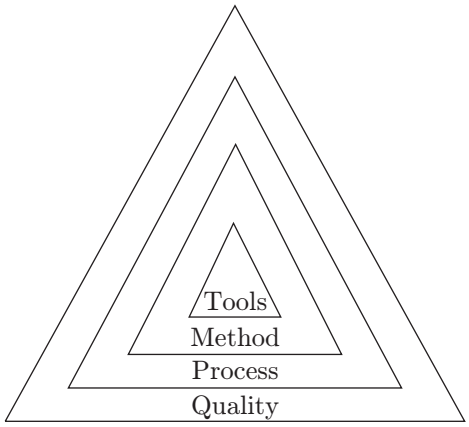


Figure 9.4 | Layered approach of software engineering.

2. **Generic approach:** The generic approach consists of the following three stages:
   - ***Definition:*** This stage is focused on 'What', that is, what information is processed, what functionality and performance is desired, what system behaviour is expected, what constraints are imposed and what validation criteria is used. In this stage, analysis and requirement operations are performed.
   - ***Validation:*** 'Are we building the right product?'. The right condition is injected in the testing phase of the development to cover the errors.
   - ***Development:*** This stage is focused on 'how', that is, how to process the information, how to define the data structure, how to maintain the interface, how to convert procedural description to machine readable code, and how to develop test case etc. This case performs design, coding and testing operations.

9.3.2.1 Support

After delivering, the project support stage is required to maintain the software. The following four types of supports or maintenance are required:

1. **Corrective support:** In this support, unidentified errors are going to be managed. The corresponding maintenance is called as corrective maintenance.
2. **Adaptive support:** Over a period of time, if the customer wants to change the platform, that is, CPU, memory, operating system or external interfaces, there is a need of adaptive maintenance.
3. **Perfective (enhancement) support:** Over a period of time, if the customer or the end user wants to introduce new functionality into the existing software to maintain the new software, perfective maintenance is required.
4. **Preventive support:** When a customer keeps on changing the requirement, the software undergoes deterioration. To maintain such a case, there is a need of preventive maintenance or re-engineering.

9.3.2.2 Software Process

A process gives the framework to develop efficient software. On the basis of the process used to develop the software, the organisation undergoes assignment to assess if there is a need of a maturity model.

The Software Engineering Institute (SEI) has introduced a maturity model to assess an organisation. This model is called as the capability maturity model (CMM). The CMM consists of the following five (5) maturity levels:

1. **Level 0 (Initial):** In this level, there is no standard maintained to develop the software. This level uses the adhoc procedure, and development becomes chaotic. Success depends on the individual levels.
2. **Level 1 (Repeatable):** In this level, based on the knowledge of previous project management, activities are developed to trace the cost, schedule and performance. There is no guarantee to the success scenario of the current project.
3. **Level 2 (Defined):** In this level, project management and engineering activities are defined. All the standards are there in the diagrammatical approach, they are not prepared for anticipated situation.
4. **Level 3 (Managed):** In this level, project management and engineering activities are quantitatively collected and documented. Quality attributes are also defined. Because of the lack of continuous improvement, minimum quality is assured.
5. **Level 4 (Optimised):** In this level, more profit is gained with minimum effort while maintaining the continuous improvement process model, innovative ideas, tools and techniques, and qualitative feedback from the customers.

9.4 PROCESS MODELS

There are two types of process models to develop the software (Fig. 9.5).

9.4.1 Conventional Process Model

In order to develop a software, a team of software engineers follow a development strategy specifying the process, methodology, tools and phases. This is referred to as process model. The different types of software models are as follows:

9.4.1.1 Waterfall Model

Waterfall model, also known as linear sequential model, provides a systematic and sequential approach for the development of a software starting from analysis phase to designing, coding, testing and support (Fig. 9.6).

1. **Requirement gathering:** On the basis of the business objective, the input domain and output domain are defined. According to the IEEE standard, requirement is defined as a condition or a capability required by the user to interact with the system or "A documented representation of conditional capability is called as requirement".
2. **Design:** In this stage, the following four operations are performed:
   - *Creating the data structure:* It identifies the data objects and attributes and creates the relationship between the data objects.
   - *Creating the software architecture:* It creates the blue print, that is, the skeleton to represent the inflows and outflows of the software.
   - *Identifying the interfaces:* It represents the interconnection between different modules present in the software.
   - *Defining procedural details:* It converts the high-level abstraction into low-level abstraction (algorithmic approach).
3. **Coding:** It translates the procedural details into machine readable form.
4. **Testing:** In this stage, different test cases are implemented to cover the logical errors and functional errors.
5. **Supports:** After delivering the software to the customer, support stage is involved to maintain the software and to adapt to the changes.

Note: When the requirements are clear, use this model to develop the software. But the final product will be available after a long time schedule.

9.4.1.2 Prototype Model

When customer requirements are not clear, the prototype model is used to finalise the requirement. The framework of prototype model is shown in Fig. 9.7.

In this model, listening to the customer is the entry point to prepare the software requirement specification (SRS). On the basis of the SRS, the designer performs the quick design, quick code. Quick test mock up. The prototype model is evaluated by the customer, and based on the feedback of the customer its iteration continues until the customer is satisfied. After finalising the SRS, the same process model is used to build the final product, which is called as open-end prototyping or evaluating prototyping model. After finalising the SRS, the developer can choose another process model to build the final product, which is called as throwaway prototyping or close-end prototyping.

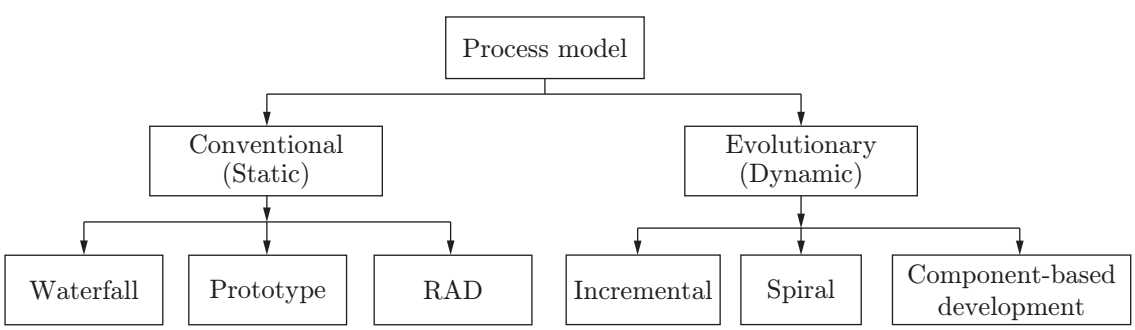


Figure 9.5 | Classification of process model.

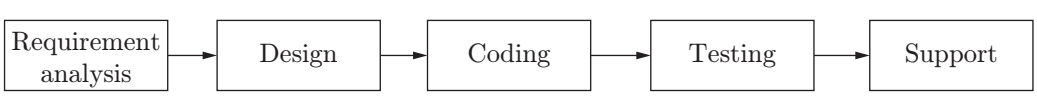


Figure 9.6 | Waterfall model.

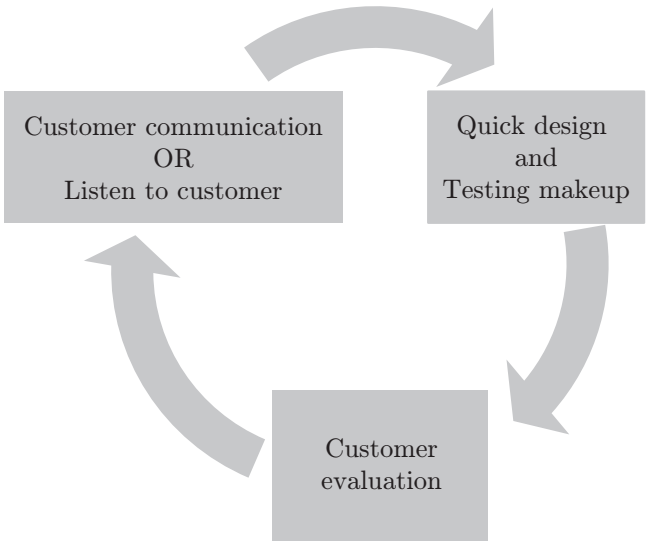


Figure 9.7 | Prototype model.

9.4.1.3 Rapid Application Development

The rapid application development (RAD) model is used when the requirements are clear but the time schedule is very short. In this model, the project is divided into modules and develops the model simultaneously. To maintain effective modularity make sure that the model has high cohesion and low coupling.

1. **Coupling:** It is a quantitative measure of functional strength of a module.
2. **Cohesion:** It is a quantitative measure of to what extent the module is independent from another module.

Framework of the RAD model is shown in Fig. 9.8.

1. **Business modelling:** It identifies the business objective. On the basis of business objective, we can gather data objects.
2. **Data modelling:** It creates the data structure, that is, identifies the data objects and attributes and relationship.
3. **Process modelling:** It creates the information flow in the project by modelling, adding or deleting the data object.
4. **Code generation:** By using a fourth-generation technique, it translates the procedural details into machine-readable format.

Test case: Different test cases are developed and implemented to cover the error. At the end of the test state, all the modules are available with error-free code, which can be integrated to build the final product. RAD model is suitable only when the organisation has efficient manpower.

9.4.2 Evolutionary Process Model

In the evolutionary process model, the software is developed in version basis because requirements are not static. When requirement changes, that change is reflected in the version. Under this case, different models are used to develop effective software.

9.4.2.1 Incremental Model

In this model, classic life cycle is used to develop the software. See Fig. 9.9.

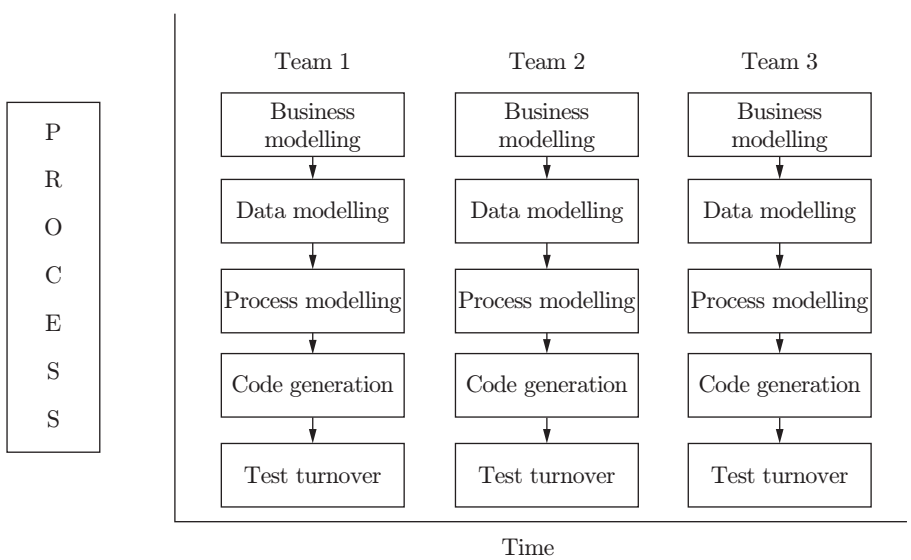


Figure 9.8 | RAD model.

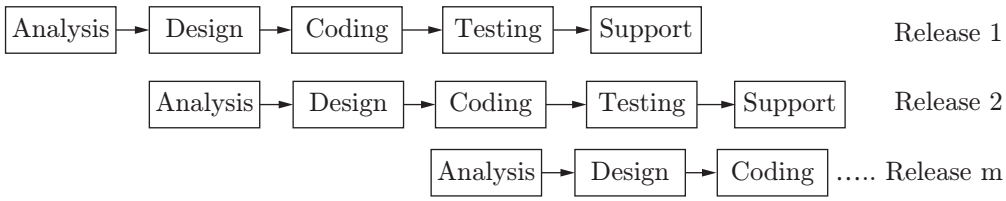


Figure 9.9 | Incremental model.

9.4.2.2 Spiral Model

Steps involved in spiral model (Fig. 9.10) are as follows:

Step 1: Customer communication

Step 2: Planning

Step 3: Risk analysis

Step 4: Engineering

Step 5: Construction and release

Step 6: Customer evaluation

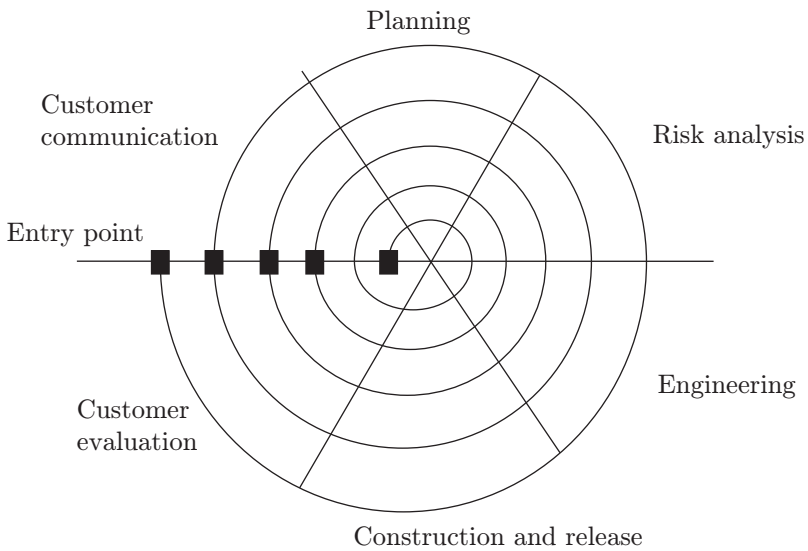


Figure 9.10 | Spiral model.

1. In the spiral model, the entry point is the customer communication to finalise the SRS. In the planning stage, we can prepare the estimation model to estimate the cost schedule and performance.
2. In the risk analysis stage, we can identify the technical and management risks. In the engineering stage, we can select the effective process model to develop the software.
3. In the construction and release stage, the procedural description is converted into machine-readable format and test cases also implemented. Later, the software is delivered to the customer.
4. In the customer evaluation stage, the software undergoes operation. During operation the customer identifies the change requirement and posts it to the development team. The change is reflected in the next release of the software.
5. In the spiral model, customer satisfaction is high, and at the same time the designer is able to design efficient design templates and deliver the code on budget. So, this model is called win-win spiral model.

9.4.2.3 Component-Based Development Model

In the component-based model, the components are used in the project development (Fig. 9.11).

After designing the project, before constructing the final software check the availability of the components in the library. If the component is available, use it in the current project; if not, construct the new component and place it in the library, then move to the next iteration.

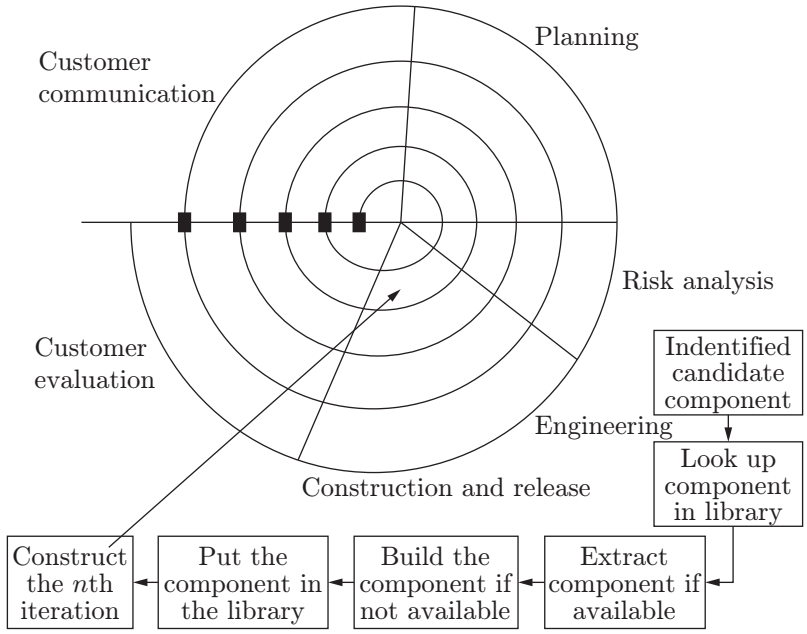


Figure 9.11 | Component-based development (CBD) model.

9.5 MEASUREMENT OF METRICS

Measurement is a quantitative indication which shows the size or complexity of a process.

9.5.1 Metrics

Metrics is a quantitative measure of an attribute of the project or process.

Measurement is divided into two types:

1. **Direct measurement:** In the direct measurement, the value is calculated based on direct approach. Example: Size of a project can be calculated based on the LOC (line of code).
2. **Indirect measurement:** Indirect measurement means the value can be calculated based on the other parameters. Example: Size of a project can be calculated by using the function point.

Software supports with four kind of P's:

1. People
2. Product
3. Project
4. Process

On the basis of the four P's, the software metrics are classified into the following three types:

1. **Product metrics:** This metrics describes the characteristics of the product such as size, complexity, performance and quality level.
2. **Project metrics:** It describes the project characteristics and execution. For example, effort cost and schedule; effort in terms of man-months. Effort of a project is 3 man-months means one developer works 3 months to develop the software.
3. **Process metrics:** It describes the characteristics of the process model to develop and maintain the software (Fig. 9.12). For example, calculates the defect rate and defect removal efficiency.

All the estimation models are empirical modules, that means, the models are developed based on the past experience without proof. Even though these models lead to success scenario of the software development.

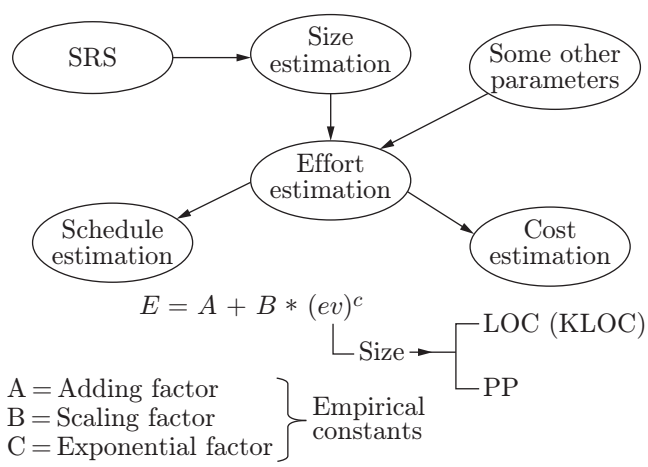


Figure 9.12 | Process metrics — an illustration.

9.5.2 Size-Oriented Metrics

The size of a project can be calculated using the following two ways:

1. Line of code (LOC)
2. Function point analysis (FPA)

9.5.2.1 Line of Code

If we want to estimate the size of a project, there is a need of taking the past project experience. Assume the LOC into three categories:

1. Optimistic LOC (S_{opt})
2. Most likely LOC (S_m)
3. Permissible LOC (S_{pess})

After identifying three levels of the source code, we can estimate the size of the project by using this following formula:

$$S = \frac{S_{opt} + 4S_m + S_{pess}}{6}$$

Problem 9.1: Consider a three-dimensional (3D) geometric application. The range of LOC estimated is $S_{opt} = 5600$, $S_m = 7900$ and $S_{pess} = 9600$. What is the expected size?

Solution:

$$S = \frac{5600 + 31600 + 9600}{6} = \frac{46800}{6} = 7800$$

So, the estimated size = 7800 LOC = 7.8 KLOC

9.5.2.2 Function Point Analysis

Function point analysis (FPA) is an indirect measurement to calculate the size of the project. Therefore, there is a need to consider the other parameters.

The function points are calculated based on the following five attributes:

1. Number of user input
2. Number of user output
3. Number of user enquiries
4. Number of files
5. Number of external interface

Every input is associated with the weighing factor. The weighing factor is defined based on the experience of the past project.

If the project is simple, the corresponding values are maintained in a table (Table 9.1). If project is complex, those values are also maintained in table.

Table 9.1 | Function point estimation

Attribute	Value	Weighing Factor		
		Simple	Average	Complex
I/P files		3	4	5
O/P files		4	5	7
Enquiries		3	4	6
Files		7	10	15
External interfaces		5	7	10

$$\text{Count value} = \text{Values} \times \text{Weighing factor}$$

$$\text{FP} = \text{Count value} \times \text{VAF}$$

Apart from the functional requirements, some other parameters are also involved in the project. So, we need to consider the effects of these factors on the project.

$$\text{VAF} = 0.65 + 0.01 \times \sum_{P=1 \text{ to } 14} F_i$$

where VAF is value adjustment factor.

Note: On the basis of the knowledge of the previous project, the weighing table is maintained as a simple/average/complex case. Calculate the count value by providing the multiplication between attribute values with the corresponding complexity levels. Default value of $\sum_{P=1 \text{ to } 14} F_i = 42$, if value adjustment complexity values are not given, use the default value.

Problem 9.2: Consider the software project with the number of input and output, enquiries, files and interfaces as 30(4), 25(5), 20(4), 10(10), 5(7). Consider () is a weighing factor. Assume all complexity adjustment factor are average = 3. Calculate the FP for the project.

Solution:

Values	Weighing Factor	Total
40	4	160
20	5	100
25	4	100
15	10	150
5	7	35
		Grand total = 545

Value of weighing factor = 545

FP = 545 × [(0.65) + 0.01 × 42] = 583.15

Problem 9.3: Consider 3D geometry application. The size of the project can be measured in terms of FP with the following parameters:

USER	
I/O	$3 \times 4 = 12$
O/P	$2 \times 5 = 10$
Enquiries	$2 \times 4 = 8$
Files	$1 \times 10 = 10$
External interfaces	$4 \times 7 = 28$

Assume the project complexity is average and value adjustment complexity is 1. Calculate the FP.

Solution:

$$\begin{aligned} \text{FP} &= 68 \times (0.65 + 0.01 \times 14) \\ &= 68 \times (0.65 + 0.14) \\ &= 68 \times (0.79) \\ &= 53.73 \end{aligned}$$

Problem 9.4: The effort of the above project is 5 person-months. What is the productivity?

Solution:

Given that 5 persons-month $\Rightarrow$ 53.72

5 person-months means that the work done by 5 persons in 1 month.

Person 5 $\rightarrow$ 53.72

Person 1 $\rightarrow$ 53.72/5 = 10.74

Productivity = 10.74

or we can say, 10.74 functional point developed by 1 person-month.

9.5.3 Effort and Schedule (Duration) Estimation

Effort and duration can be estimated by using different empirical model structure.

1. According to the software engineering laboratory (SEL) empirical model, the effort can be estimated as follows:

$$E = 1.4\ (\text{KLOC})^{0.93} \text{ person-months (units)}$$

The duration can be calculated as

$$D = 4.6\ (\text{KLOC})^{0.26} \text{ months (units)}$$

2. According to Walstonald-Felio method (W-F) (IBM):

The effort can be calculated as

$$E = 5.2\ (\text{KLOC})^{0.91} \text{ person-months}$$

Duration can be calculated as

$$D = 4.1\ (\text{KLOC})^{0.36}$$

Problem 9.5: Software development is expected to involve eight parameters of efforts. Calculate the following using SEL and WF models.

(a) The number of lines of code that can be produced.
(b) The duration of the development
(c) The productivity in LOC per person-year
(d) The average manning (person)

Solution:

By SEL method:

(a)
$$E = 1.4\ (\text{KLOC})^{0.93}$$
$$8P - 4 = 1.4\ (\text{KLOC})^{0.93}$$
$$(\text{KLOC})^{0.93} = \frac{(8 \times 12)}{1.4} = \frac{96}{1.4}$$
$$\text{KLOC} = \left(\frac{96}{1.4}\right)^{1/0.93} = 90.0414$$

So, LOC = 90041

(b) $D = 4.1\ (\text{KLOC})^{0.36} = 4.6 \times (90.0414)^{0.26}$
$= 4.6 \times 3.22 = 14.8 \sim 15$ months

(c) Productivity $= \dfrac{90041}{8} = 11255$

(d) Manning = Effort/Duration = $(8 \times 12)/15$ person-months = 6.4 persons

By WF method:

(a)
$$E = 5.2\ (\text{KLOC})^{0.91}$$
$$8P - 4 = 5.2\ (\text{KLOC})^{0.91}$$
$$(\text{KLOC})^{0.91} = \frac{(8 \times 12)}{5.2} = \frac{96}{5.2} = 18.46$$
$$\text{KLOC} = \left(\frac{96}{5.2}\right)^{1/0.91} = 24.63$$

So, LOC = 24630

(b) $D = 4.6\ (\text{KLOC})^{0.26} = 4.6 \times (24.63)^{0.26}$
$= 4.6 \times 2.3 = 10.6 \sim 11$ months

(c) Productivity $= \dfrac{24630}{8} = 3078.75$

(d) Manning = Effort/Duration = $(8 \times 12)/11$ person-months = 8.7 persons

9.5.3.1 Constructive Cost Model

By using constructive cost model (COCOMO), we can calculate the effort and time, this model is also an empirically derived model. Therefore, the structure of the effort calculation is

$$\text{Effort} = c_1 \times \text{EAF} \times (\text{size})^{c_2} \text{ (person-months)}$$

$$\text{Duration} = c_3 \times (\text{Effort})^{c_4} \text{ (months)}$$

where c_1 and c_3 are the empirically derived scaling factors and c_2 and c_4 are empirically derived exponential

factors. Size indicates in terms of KLOC or functional point. EAF (effort adjustment factor) (1 to 14 factors) rating (1–7) range.

COCOMO model is applicable in three different kinds of applications, as follows:

1. Organic mode (low complexity)
2. Semi-detached mode (average complexity)
3. Embedded mode (high complexity)

On the basis of the type of project, the constants are derived as follows:

	c_1	c_2	c_3	c_4
Simple	2.4	1.05	2.5	0.38
Average	3.0	1.12	2.5	0.35
Complex	3.6	1.20	2.5	0.32

Problem 9.6: A company needs to develop digital signal processing software for one of its newest invention. The software is expected to have 40000 LOC. The company needs to determine the efforts in person-months needed to develop this software using the basic COCOMO model.

The multiplicative factor for this model is given as 2.8 for the software development on the embedded system while exponential factor is given as 1.20. What is the estimated effort in person-months?

(a) 234.35 (b) 932.50
(c) 287.8 (d) 122.4

Solution:

Effort = 2.8 EAF × $(40000)^{1.20}$(EAF = by default =1)

$= 2.8 \times (40000)^{1.20}$

$= 2.8 \times (40)^{1.20}$

$= 234.35$

Problem 9.7: A company develops software for a digital signal processing application. The software size is expected to have 100000 LOC. LOC is sometimes called as DSE (delivery of source instructions).

The company needs to determine the effort in person-months and time in months to develop the software using COCOMO model under embedded mode. The effort adjustment factor is calculated based on the given table. What is the estimated effort and time for the above project.

S.No.	Cost Driven	Effect
1.	Language experience	1.0
2.	Schedule constraints	1.0
3.	Database size	1.0
4.	Turnaround time	1.0
5.	Virtual machine exp.	1.0
6.	Virtual machine volatility	1.0
7.	Use of software tools	0.88
8.	Modern programming practices	1.0
9.	Storage constraints	1.0
10.	Application experience	1.10
11.	Timing constraints	1.0
12.	Requirement reliability	1.15
13.	Product complexity	1.15
14.	Team capability	1.0

Solution:

AF = Multiplication of all cost-driven effect

(Multiple all) EAF = 1.28

$$E = c_1 \times \text{EAF} \times (\text{size})^{c_2}$$

As c_1= 2.0, so c_2 = 1.2 = 2.8 × 1.28 $(100)^{1.2}$ = 900 (person-months)

$D = 2.5 \times (\text{Effort})^{0.32} = 2.5 \times (900)^{.32} = 22.04$ months ~ 23 months

9.5.3.2 Defect Rate

It is a quality metric, defect is undiscovered error during the development of the software. The defect rate is defined as

$$\text{Defect rate} = \frac{\text{Number of defect} \times \text{Time}}{\text{Total or number of LOC}}$$

9.5.3.3 Defect Removal Efficiency

Defect removal efficiency (DRE) is also a quality metric. It is defined as

$$\text{DRE\%} = \frac{E}{E + D} \times 100$$

where E = error and d = defect.

9.5.3.4 Halstead Size-Oriented Metric

Halstead metric is used to count the number of lines preset in the software. To calculate the size of the project in terms of LOC, we are supposed to count the number of lines in the software. But this value does not give the correct estimation because comments and spaces and non-executable lines are also counted.

LOC estimation means only the count of executable statements. To identify the correct executable statement in the software, there is a need to divide the program in the form of lexemes.

Lexemes	Token
For	Keyword
=	Equal -op
+	Add -op
*	Mul -op
3	Constant

After dividing the program into token, we can separate the operator and operands. Operators are special words and special symbol operators, etc. Operands are constant, variable, identifier, etc.

Unique Operator	Occurrence of the Operator in the Software	Unique Operand	Occurrence of the Operand in the Software
$\Sigma = n_1$	$\Sigma = N_1$	$\Sigma = n_2$	$\Sigma = N_2$

where n_1 = the count of unique operator used in the software

n_2 = count of unique operands used in the software

N_1 = count of occurrence of the operator in the software

N_2 = count of occurrence of the operands in the software

The following metrics are defined based on the above database.

The vocabulary can be calculated by using $n = n_1 + n_2$

The size of the software can be calculated by

$$N = N_1 + N_2$$

When we are considering machine language then

$$N = 2 \times \text{LOC}$$

The volume of the program can be calculated by using

$$V = N \log_2 n$$

The estimated level of program is defined as

$$L = \frac{2n^2}{N_1 \times N_2}$$

The difficulty is defined as

$$D = \frac{1}{L}$$

Error is defined as

$$E = \frac{V}{L}$$

Estimated program length is calculated as

$$n_1 \times \log_2 n_1 + n_2 \times \log_2 n_2$$

Problem 9.8: Consider the following program and calculate the software matrix n, N, V, D, E

```
var a, b, c, d, m: integer'
Begin
    Read in (a, b, c, d)
    a: = a + a;
    b:= b + b;
    c: = c * d;
    m: = a + b - c;
    write in (m);
End
```

Solution:

Unique Operator	Occurrence of the Operator in the Software	Unique Operator	Occurrence of the Operand in the Software
=	4	a	6
+	3	b	6
*	1	c	5
–	1	d	3
,	7		
:	1		
;	7		
( )	2		
.	1		
Read in	1		
Write in	1		
Integer begin-end	1		
n_1 =13	$N_1 = 31$	$n_2 = 5$	$N_2 = 23$

$$n = n_1 + n_2 = 13 + 5 = 18$$

$$N = N_1 + N_2 = 31 + 23 = 54$$

$$D = \frac{1}{L}$$

$$L = \frac{2n_2}{n_1 \times N_2} = \frac{2 \times 5}{13 \times 23} = \frac{1}{0.033} = 30.30$$

$$V = n \cdot \log_2 n = 54 \cdot \log_2 18 = 54 \times 5 = 270$$

$$E = \frac{V}{L} = \frac{270}{0.033} = 8181.81$$

9.6 RISK ANALYSIS

Risk means extreme condition when the product becomes a failure. Risk is associated with two characteristics:

1. **Uncertainty:** It means may/may not occur, 100% probable risk.
2. **Loss:** It means when the risk becomes real there is a loss, otherwise no loss.

9.6.1 Risk Identification

After identifying the extreme condition, there is a need of analyses of the risk in terms of level of uncertainty and degree of loss. The level of uncertainty is maintained in the following four ways:

1. Negligible
2. Marginal
3. Critical
4. Catastrophic

We are going to identify critical and catastrophic risk condition and try to manage them in the software execution with the help of risk strategy.

9.6.1.1 Risk Strategy

There are two types of risk strategies:

1. **Reactive risk strategy:** It means monitoring the project for the probable risk. When the risk is present during execution of the project, it implements the plans to control the risks. It is also called as fire-lighting mode.
2. **Proactive risk strategy:** It means risk plans and actions are developed before the implementation of the software. So, by using these strategies we can avoid the risk, if possible, or manage the risk, if not possible. By combining both proactive and reactive strategies, the risk mitigation, monitoring and management (RMMM) plan is prepared.

9.6.1.2 Types of Risk

1. **Project risk:** This risk threatens the project plan, therefore schedule will be extended and development cost will increase.
2. **Technical risk:** These risks threaten the quality and timeliness of developing the software. When this risk becomes real, we cannot develop the project within the quality levels.
3. **Business risk:** These risks threaten the viability of software risk. Under this category, four different risks are defined:
   - *Market risk:* Build an excellent product but no users.
   - *Strategic risk:* Build a product that is no longer fit into the business objective.
   - *Management risk:* Lack of high-level support to inject the changes.
   - *Budget risk:* No synchronisation between cost and price.
4. **Known risk:** Uncover errors after a careful evaluation of the project.
5. **Predictable risk:** It is extrapolation from the past project experience.
6. **Unpredictable risk:** It acts as a joker in the deck.
7. **Generic risk:** It describes the characteristics of the project such as planning, performance, cost, effort, etc.
8. **Product-specific risk:** These risks describe the characteristic of the product such as the size, quality, performance, etc.

Risk Components

There are four different components present in the software to maintain the probability of probable risk:

1. **Performance:** It indicates the level of uncertainty to meet its specification.
2. **Cost:** It indicates the level of uncertainty to maintain project budget.
3. **Support:** It indicates the level of uncertainty to easily correct, adopt or enhance the existing software.
4. **Schedule:** It indicates the level of uncertainty to maintain the schedule and quality of the software.

9.6.2 Risk Projection or Risk Estimation

To estimate the cost of the risk, there is a need to identify the probability of the risk becoming real and the associated impact.

Risk Name	Category	Probability	Impact

By using the above database we can estimate the risk cost by using the following formula:

$$\text{Risk exposure (RE)} = P \times C$$

where P is the probability of risk becoming real and C is the cost.

Problem 9.9: To develop an image-processing application, the software uses component-based development approach '60'. Reusable software component are planned but 70% of the project reusable component are used. '18' components are developed from the baseline to complete the project. The cost component size is '100' LOC, and the cost for each LOC is \$14. What is the risk RE when the risk probability is 80%?

Solution:

$$\text{RE} = P \times C$$
$$P = 80\%$$
$$C = 18 \times 100 \times 14 = \$252000$$
$$\text{RE} = 0.8 \times 25200$$
$$= \$20160$$

9.7 SOFTWARE DEVELOPMENT LIFE CYCLE

The software development life cycle (SDLC) describes the generic approach to develop the software. The framework of the SDLC is shown in Fig. 9.13.

9.7.1 Requirement

According to the IEEE standard, the requirement is defined as a condition or capability needed by a user to solve a problem or to achieve an objective.

There are five types of requirements classified in the software engineering (Fig. 9.14):

1. **Business requirement:** These requirements describe the high-level business requirements. These requirements are documented in vision and scope document.

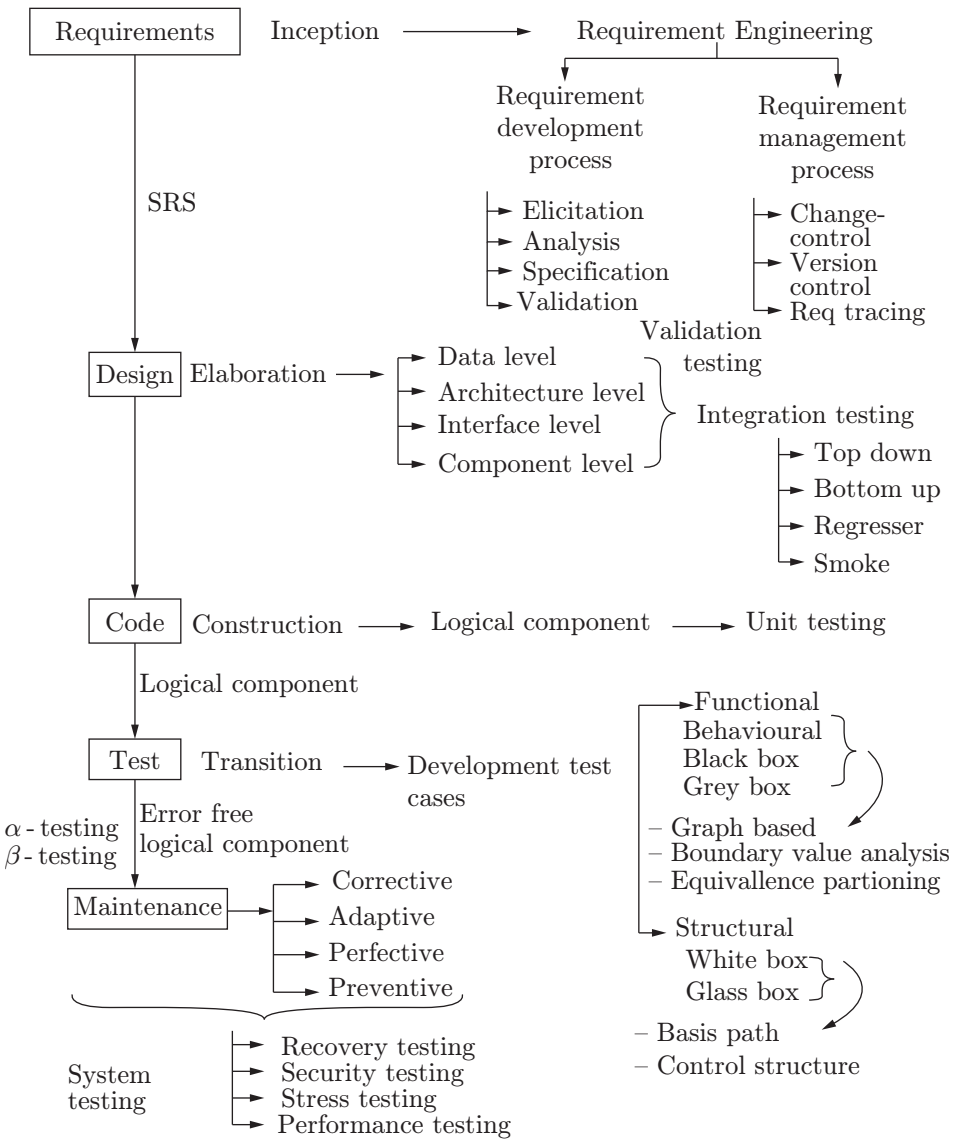


Figure 9.13 The software development life cycle.

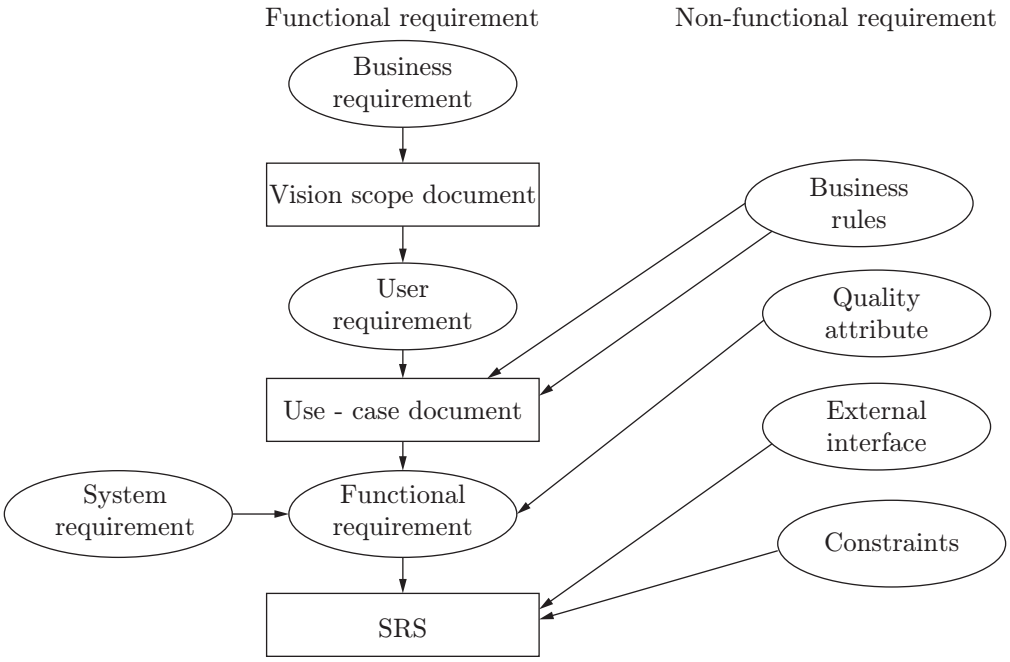


Figure 9.14 | Requirement phase of SDLC.

2. **User requirement:** This requirement describes the user need, user goal and user capability towards the system to achieve their objective. Those requirements are documented in use-case document.
3. **Functional requirements:** This requirement describes the functionality of the software, which is developed into the system to satisfy the user requirement. These requirements are documented in the SRS document.
4. **System requirement:** This requirement describes the subsystem requirement on which platform the developed product is running, that is, CPU, memory capacity and operating system. These are the subset of the functional requirement.
5. **Non-functional requirement:** This requirement describes the business rules (such as govt. acts, govt. policies, account plans) and quality attributes (availability, usability, maintainability, robustness, flexibility).

9.7.1.1 External Interfaces

Like how many other systems are communicating with the software and constraints (risk condition). All these requirements are documented in the SRS.

Note: SRS document consists of goals of the software, functional requirement and non-functional requirements.

In order to develop the right system, the right requirements need to be injected into the software development process, and for developing the right requirement there is a need of requirement engineering.

9.7.1.2 Requirement Engineering

Requirement engineering includes the systematic, disciplined, quantifiable approach to the development, operation and maintenance of the requirement. The following two processes take place in requirement engineering:

1. **Requirement management process:** These activities are used to manage the requirement throughout the project development and lifetime of the product.
2. **Requirement development process:** The framework used to develop the requirement is shown in Fig. 9.15.

On the basis of the goal of the software, elicit the requirements from different sources and create the data dictionary.

On the basis of the data dictionary, create the structure of the software. On the basis of the structure of the software, prepare the technical specifications. All the customer capabilities are maintained in the SRS document.

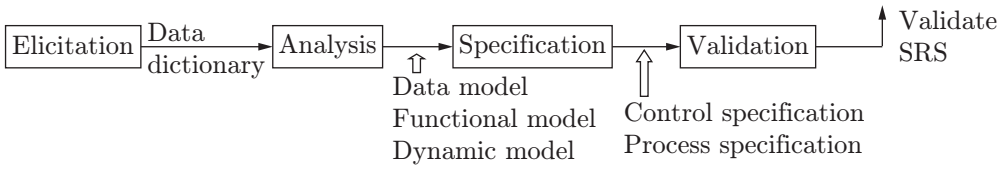


Figure 9.15 | Requirement process.

After reviewing the SRS document twice or thrice, the customer and the developer sign on the SRS document. This document (SRS) is then forwarded to the design face.

9.7.1.3 Elicitation

In this state, by using different techniques, elicit the requirement from different sources towards goal of the project. The different elicitation techniques are as follows:

1. **CORE (Controlled Requirement Expression):** In CORE technique, elicit the requirement by using view point analysis. In this analysis, maintain the sequence of data object, action and consequence.
2. **IBIS (Issue-based Information System):** In IBIS technique, elicit the requirement based on the question and answer analysis. This technique maintains the issue, position and justification.
3. **FODA (Future-oriented Domain Analysis):** In FODA technique, elicit the requirement by creating different models to specify the structure of the end product. This technique maintains the sequence context model — function model — dynamic model (behaviour or data model).
4. **QFD (Quality Function Deployment):** In QFD technique, elicit the requirement towards quality deployment.
5. **JAD (Joint Application Development):** In JAD technique, elicit the requirement by conducting a quick meeting between customer, end user, analyst and developers.
6. **Prototype:** In prototype technique, elicit the requirement by creating the quick design mock-ups and taking continuous feedback from the customer.

Note: The output of the elicitation stage is data dictionary, it consists of data objects related to the software.

9.7.1.4 Analysis

This stage describes the structure of the end product in terms of data object, functionality, information flow and behaviour with respect to external event, by creating three types of model:

1. Context or data model (E-R diagram)
2. Functionality and information flow model (described by DFD)
3. Dynamic or behavioural model (described by STD)

1. **Context or data model (E-R Diagram):** This model describes the structure of the end product in terms of data object and relationship. Data model can be represented by using the entity, relationship diagram. E-R diagram consists of object, attribute and relationship.

   Object is defined as a representation of composite elements. Composite elements must have multiple properties. The property of the object is called as attribute. The relationship can be maintained between the object by using two concepts.

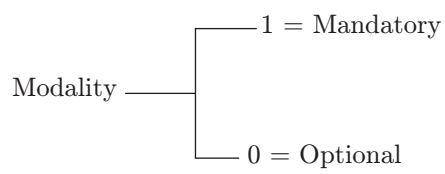
   - **Cardinality:** It indicates in how many ways one object can be communicated with another object. There are three types of cardinality:

     1: 1

     1: n

     M: N

   - **Modelling:** It indicates "is the communication link between the object mandatory or optional".



   Note: On the basis of the data model, the data structure is created in the data level design.

2. **Functional and information flow model:** The functional and information flow model describes the functionality of the end product, that means, what is the behaviour of the end product with a proper set of input. The functionality can be represented by using the data flow diagram (DFD). Data flow diagram is also called as Bubble chart or directed graph. Different levels of abstraction of the end product can be represented using different levels of DFD.

   - **Level-Zero DFD:** It describes the high-level abstraction of the functionality by maintaining the single process bubble with multiple input- and output-directed edges (Fig. 9.16).

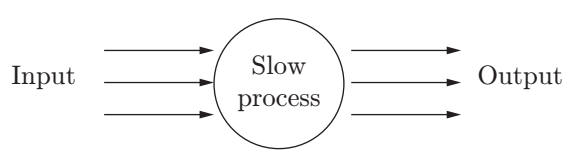


Figure 9.16 | Level-zero DFD.

   In this level of DFD, only the functionality is represented with absence of information flow.

   - **Level-1 DFD:** It describes the functionality and information flow of the software by maintaining different process bubbles. It describes the project in low-level abstraction. Low-level abstraction is easy to code (Fig. 9.17).

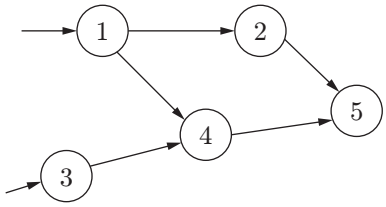


Figure 9.17 | Level-1 DFD.

3. **Dynamic model:** It describes the behaviour of the system with respect to external events. This model is represented using the state transition diagram (Fig. 9.18).

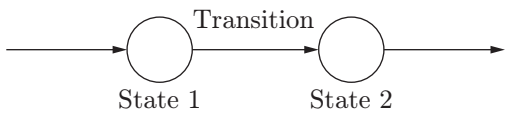


Figure 9.18 | State transition diagram.

- **Specification:** In this stage, the models are refined into control specification and process specification
  (a) **Control specification:** It shows the behaviour of the product.
  (b) **Process specification:** It refines the mathematical expression, algorithm and constraints.
- **Validation:** In this stage, various validation criteria are implemented to validate the requirement towards the right condition. The following reviews are done to validate the software:
  (a) Configuration review
  (b) Quick review
  (c) Detailed review

  "Validation means the black-box testing is performed at the requirement stage itself".
  After validating the SRS document, it is forwarded to the design stage. The output of the requirement development process is shown in Fig. 9.19.

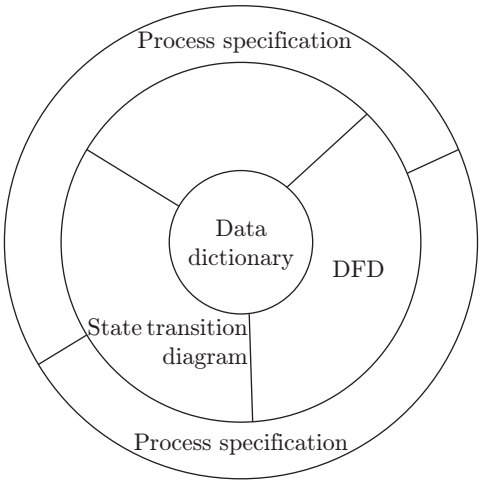


Figure 9.19 | Validation.

9.7.2 Design

The design stage represents the project in the low-level abstraction, that means, the customer description is translated into technical description. To do this process, the following four levels of design take place (Fig. 9.20):

1. **Data level:** Data level design describes the data structure. It takes help from the E-R diagram and data dictionary.
2. **Architecture level:** It defines the structure of the end product (skeleton). It takes help from the DFD of the last stage.
3. **Interface level:** Interface level design describes the number of interfaces required in the system. It takes help from the state transition diagram.
4. **Component level:** It represents the procedural description to implement the functionality in the software. It can take help from process specification and control specification (Fig. 9.21).

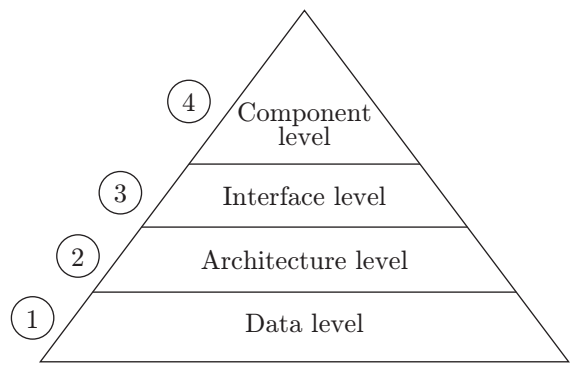


Figure 9.20 | Levels.

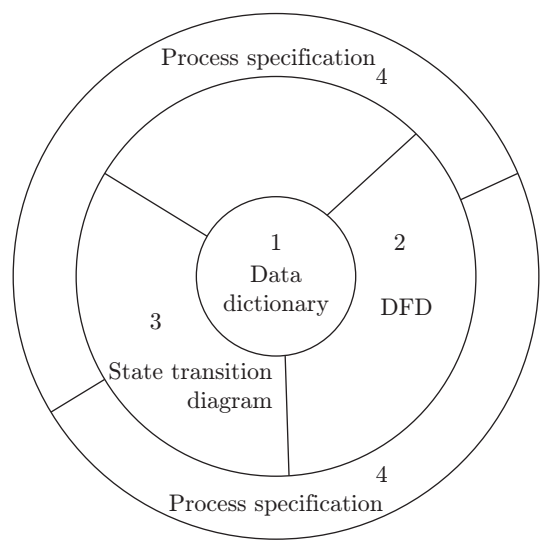


Figure 9.21 | Component.

9.7.2.1 Designing Concept

The following concepts should be kept in mind while designing a document:

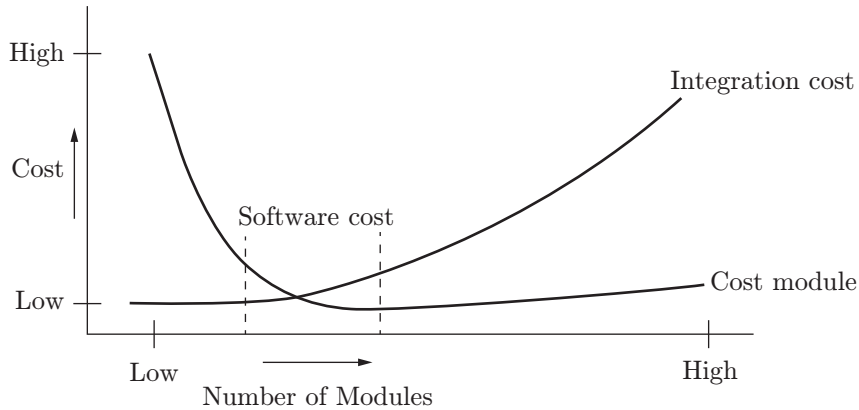


Figure 9.22 Number of modules versus cost of software development.

1. **Abstraction:** There are different levels of abstraction available to represent the statement. In high-level abstraction, the statement is represented in concrete manner, that means, internal structure is not provided. In low-level abstraction, all statements are defined in a single format with all details of implementation.
2. **Refinement:** It is a top-down strategy to simplify the statement structure. Refinement is also called as elaboration. After refinement of the statement, we can get the direct implementation statement.
3. **Modularity:** Software development process uses the modularity concept to reduce the development cost and time. Suppose the complexity of the problem is denoted by $C(P)$ and the effort of the problem is denoted by $E(P)$. The following formats represent the advantage of C.

   $C(P_1) \quad E(P_1)$

   $C(P_2) \quad E(P_2)$

   $C(P_1) > C(P_2)$

   $C(P_1) > E(P_2)$

   $C(P_1 + P_2) > C(P_1) + C(P_2)$

   $E(P_1 + P_2) > E(P_1) + E(P_2)$

   When the number of modules is less, the cost module for developing the module is high, and at the same time the integration cost is low.

   When the number of modules increases, the cost per module is low but integration cost is high. This relationship is shown in Fig. 9.22.

Modules should be designed in such a way that each module has specific functional requirements. Functional independence is measured using two terms cohesion and coupling.

9.7.2.2 Cohesion

It is a measure of relative functional strength of a module. Following types of cohesion exist:

1. **Logical cohesion:** Logical cohesion exists when a module that performs tasks are related logically.
2. **Temporal cohesion:** Temporal cohesion exists when a module contains tasks that are related in such a way that all the tasks must be executed within the same time limit.
3. **Procedure cohesion:** The procedural cohesion exists when the processing elements of a module are related and must be executed in a specific order.
4. **Communication cohesion:** The communication cohesion exists when all the processing elements concentrate on one area of a data structure.

9.7.2.3 Coupling

Coupling is a measure of relative independence among modules, that is, it is a measure of interconnection among modules. Following types of coupling exist:

1. **Data coupling:** Data coupling exists when a module accesses another module through an argument or through a variable.
2. **Stamp coupling:** Stamp coupling exists when a module accesses another module through a data structure.
3. **Control coupling:** Control coupling exists when the control is passed between modules using a control variable.
4. **External coupling:** External coupling exists when modules are connected to an environment external to the software.
5. **Common coupling:** Common coupling exists when the modules share the same global variable.
6. **Content coupling:** Content coupling exists when one module makes use of data or control information maintained in another module. Also, content coupling occurs when branches are made into the middle of a module.

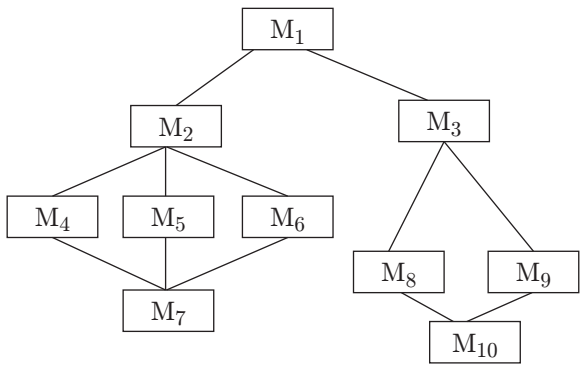
Note: Effective modular design must have high cohesion and low coupling.

9.7.2.4 Software Architecture

It shows the hierarchical structure of the software and also describes the logical connections between the modules. In this process, it identifies the data model, functional model and behaviour model.

Control Hierarchy

It indicates the control sequence in the software project. It represents the interaction between different modules (Fig. 9.23).



Fan-in ⇒ number of control in Fan-in $(M_7) = 3$

Fan-out ⇒ number of control out $(M_2) = 1$

Fan-out $(M_7) = 0$

$(M_2) = 3$

Figure 9.23 | An illustration of control hierarchy.

The control hierarchy defines various parameters of the software design:

1. **Depth:** It is a measure of the number of levels present in the software design.
2. **Width:** It is a measure of the maximum number of modules present at any one of the level.
3. **Fan-in:** It is a measure of the number of modules already controlling the given module.
4. **Fan-out:** It is a measure of the number of modules controlled by the given module.
5. **Structural partitioning:** It indicates the control sequence of development, maintenance, and integration of the software. It is divided into the following two types:
   - Horizontal
   - Vertical

Horizontal partitioning allows the level by level development and integration, that means, control flow is assigned to levels.

Vertical portioning means the control sequence is forwarded from root to leaf node, that is, depth-wise development and integration take place. Because of dependencies, this approach is not suggested to software development.

There are different advantages with horizontal partitioning:

1. Easy to develop
2. Easy to integrate
3. Easy to test
4. Easy to maintain

Data Structure

It shows the logical connection between the data objects.

Software Procedure

It describes the customer requirements into programming language statements. Information hiding is used in effective modularity. It shows the independence of the module. After the designing stage, we have the software design document (SDD). The SDD represents the programming description language oriented customer objective.

9.7.3 Coding

It translates the procedural-oriented description into machine-readable format by using appropriate programming language.

9.7.4 Testing

The objective of testing to measure to what extent the developer logical program becomes a failure. In the testing phase, different test cases are developed and implemented to cover the structural and functional errors. Functional testing includes conducting tests to check the functionality of the product with all proper sets of input, which also covers errors. Structural testing includes conducting tests to examine the internal structure of the developed product to cover the error.

9.7.4.1 Structural Testing

This testing is also called as white-box or glass-box testing. The objective of white-box testing is as follows:

1. Guarantee the all the statements present in the developed program exercise at least once.
2. Exercise logical expression towards true and false side.
3. Exercise all the loop boundaries within the operational limit and beyond the operational limit.
4. Exercise the data structure present in the program towards validity.

To reach the objectives of white-box testing, two different test operations are performed:

1. Basis path testing
2. Central structural testing

Basis Path Testing

This test is used to cover all the paths present in the development code. This testing guarantees that all the statements in the program must be exercised at least once to identify the path in the program, convert the program into graphical notation. On the basis of the control flow between the statements, the program is converted into graph notation called as flow graph or program graph. Flow graph is a directed graph.

Each node in the flow graph represents the program statement. The sequence of the statement is represented by using the following (Fig. 9.24):

Figure 9.24 | Sequence statement.

1. The if condition is represented by Fig. 9.25.

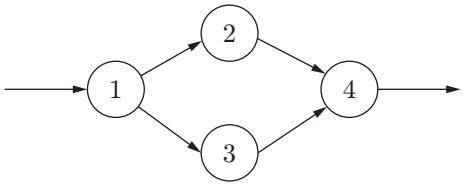


Figure 9.25 | IF condition.

2. The while statement is represented by Fig. 9.26.

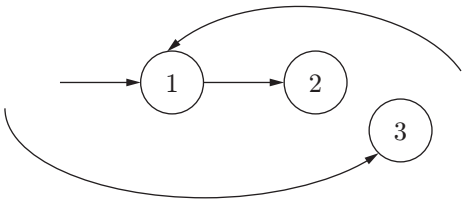


Figure 9.26 | WHILE statement.

3. Do–while statement is represented by using Fig. 9.27.

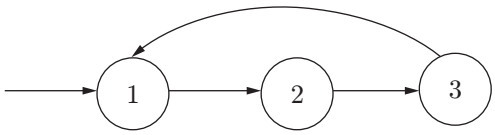


Figure 9.27 | DO–WHILE statement.

4. The switch case statement is represented by Fig. 9.28.

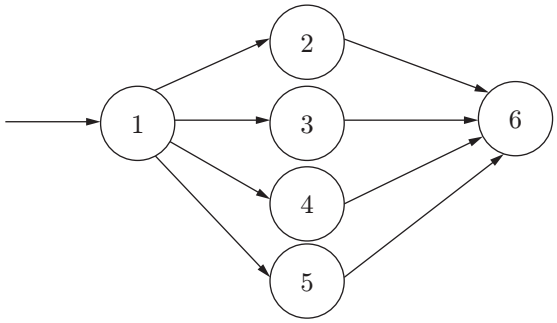


Figure 9.28 | SWITCH-CASE statement.

By using the cyclomatic complexity we can measure the logical complexity of the program.

Cyclomatic Complexity

It is defined as the quantitative measure of logical complexity of the program. Cyclomatic complexity can be calculated in three ways:

1. By using the number of edges and number of nodes we can measure the cyclomatic complexity as

   $V(G) = E - N + 2P$

   $E =$ Number of edges

   $N =$ Number of nodes

   $P =$ Number of disconnected component

2. By using the region we can estimate the cyclomatic complexity as

   $V(G) =$ Number of regions present in the graph

   When we count the cyclomatic complexity of a program by using basis path testing, outside of graph is considered as one region.

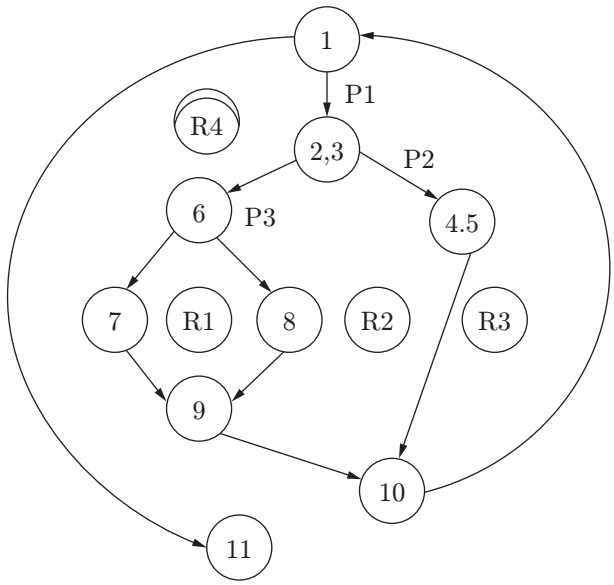
3. By using predictable nodes:

   $V(G) = P + 1$

   $P =$ Number of predicate nodes

   Predictable nodes are calculated by two or more nodes expanding from it.

Problem 9.10: Consider the following flow graph and calculate the cyclomatic complexity and also identify the independent path.



Flow graph depicting relation among different nodes.

Solution:

Cyclomatic complexity $= E - N + 2$

$= 11 - 9 + 2 = 4$

Number of regions $= 4 = R_1, R_2, R_3, R_4$

So, cyclomatic complexity $= 4$

Cyclomatic complexity $P + 1 = 3 + 1 = 4$

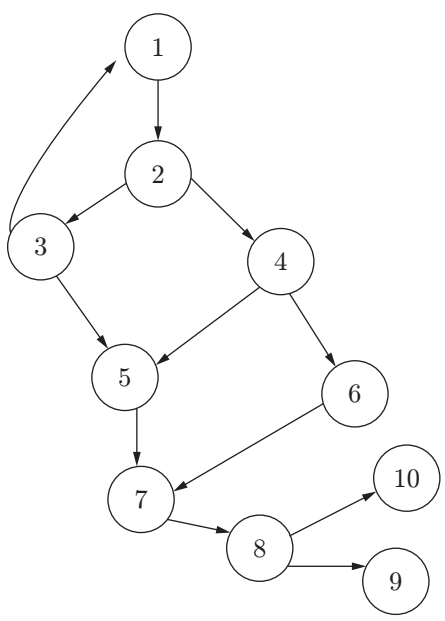
Path 1: 1-11

Path 2: 1-2-3-4-5-10-1-11

Path 3: 1-2-3-6-8-9-10-1-11

Path 4: 1-2-3-6-7-9-10-1-11

Problem 9.11: Consider the following program and calculate the cyclomatic complexity.



Flow graph depicting relation among nodes.

Solution:

Rule 3: $P + 1 = 4 + 1 = 5$

Rule 2: Number of regions $= 5$

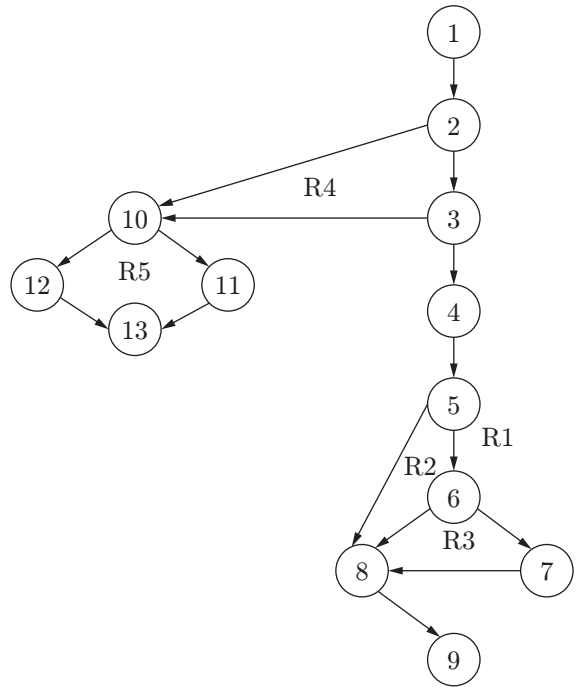
Rule 1 $= E - n + 2 = 13 - 10 + 2 = 5$

Path 1: 1-2-45-6-7-8-10

Path 2: 1-2-4-5-7-8-10

Path 3: 1-2-3-5-7-8-10 and more

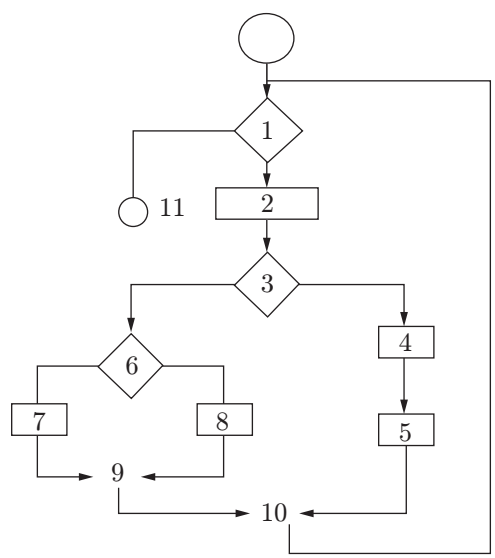
Problem 9.12: Consider the following graph and calculate the cyclomatic complexity.



Flow graph depicting relation among nodes.

Cyclomatic complexity $= 6$

Problem 9.13: Consider the following flow chart and calculate the cyclomatic complexity.



Flow chart depicting various operations of a program.

Cyclomatic complexity = Number of if statement + 1 = 3 + 1 = 4

The cyclomatic complexity of the program can be calculated by using the graph matrices.

Graph Matrices

The flow graph is converted into graph matrices by maintaining rows and columns. The number of rows and columns depends on the number of nodes in the flow graph. Based on the communication link between the nodes in the flow graph, the entry is reflected in corresponding position in the graph matrices.

If any row consists of more than one entry that node is called as predicate node, then take the weightage of predicate node by subtracting one from the total. After subtraction operation, add all the row values, it gives the predicate nodes. Use the formula to calculate cyclomatic complexity $V(G) = P + 1$.

Example 9.1

	1	2	3	4	5
1		b		c	1
2	a		d		
3		e		f	
4			g		
5	h				

Connection Matrix

	1	2	3	4	5
1	0	**1**	0	**1**	**1**
2	**1**	0	**1**	0	0
3	0	**1**	0	**1**	0
4	0	0	**1**	0	0
5	**1**	0	0	0	0

$3 - 1 = 2$

$2 - 1 = 1$

$2 - 1 = 1$

$1 - 1 = 0$

$1 - 1 = 0$

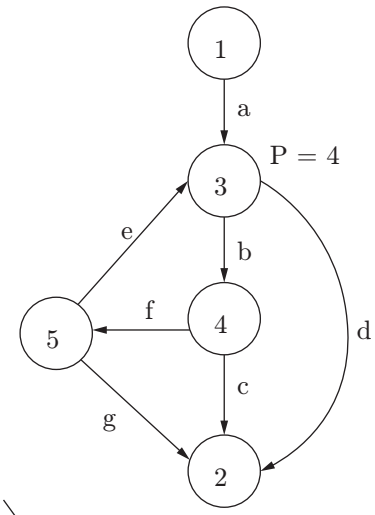
$P = 04$

	1	2	3	4	5
1		**1**		**1**	**1**
2	**1**		**1**		
3		**1**		**1**	
4			**1**		
5	**1**				

A graph matrix and its allocation.

Problem 9.14: Consider the following flow graph and create the graph matrix and calculate the cyclomatic complexity.

Note: If there are more than two outgoing nodes, then the predicate formula will not work.



Mode \ Link	1	2	3	4	5
1			a		
2					
3		d		b	
4		c			f
5		g	e		h

A flow graph and graph matrix. Solution:

So, cyclomatic complexity = $P + 1 = 5$

Control Structure Testing

This testing is used to cover the error from the control structure present in the program. There are three types of expressions:

1. Arithmetic (= , − , *, /) = value
2. Relational (>, <, <=, >=, =, ≠) = True/False (Boolean)
3. Boolean (&&, ||, ¬, /) = True/False (Boolean data type)

Examples:

1. $a + b$
2. $(a + b) > (c + d)$ = T/F
3. $((a + b) > (c + d)$ && $((a \times b) < ((c * d))$ = T/F

After evaluating the control expression, the output may be a Boolean data, that is, true/false.

In the control structure testing, examine all the logical expression towards true/false. During that evaluation, it covers the Boolean operator error, Boolean data or parenthesis symbol error, relational operator error and arithmetic operator errors.

In this testing, two test cases are developed to cover all the errors described in the above list.

1. Branch testing
2. Domain testing

9.7.4.2 Black-Box Testing

In this testing, various input test cases are developed and the product functionality is examined, whether it satisfies all possible inputs. This testing is also called behavioural testing or grey-box testing. In this strategy, various testing techniques are used to cover the functional errors such as

1. Equivalence partition
2. Boundary value analysis
3. Comparison

1. **Equivalence partition:** In this testing, the input domain is divided into equivalent partitions. Each position is called as class. One partition is the exact valid range and another partition is above the upper limit and another is below the lower limit.

Example 9.2

If the input box consists of 1 to 1000 values, no need of preparing the thousand (1000) test cases. By using equivalence partition, the above input box is divided into three test cases.

Test case is prepared between the ranges of {1–1000}

Test case is prepared {>1000}

Test case is prepared {<1}

2. **Boundary value analysis:** Most of the problems are possible at the extreme edges that are boundaries of input domain. Boundary value analysis focuses on identified errors at the boundary rather than finding those existing at the centre of the input domain.

Example 9.3

Test case for input box accepting number (1—1000) using boundary value analysis is

(a) Test cases with data exactly the input boundary of input domain, that is, 1 and 1000.
(b) Test data with values at just below the extreme edges of input domain, that is, 0 and 999.
(c) Test data with values just above the input boundary of input domain, that is, 2 and 1000.

Note: When the product supports the N variable that is tested using the boundary value analysis, that is, $4N + 1$ test cases.

9.7.4.3 Comparison Testing

When the software is developed in the real-time application domain, there is a predefined value estimated based on the product behaviour. After developing the software, compare the behaviour of the new product with predefined values and remove the errors.

9.7.4.4 Life Cycle Testing

Various testing plans are implemented at each stage of the software development life cycle.

9.7.4.5 Validation Testing

This testing is performed at requirement stage by conducting the black-box test plans on a prototype model.

9.7.4.6 Integration Testing

This testing is performed at the design stage. Software development uses the modularity concept and develops the module independently. After developing the module, there is a need of integration to generate the final system.

In that regard, integration testing is required to check the functionality. Integration testing is not focused on internal details of the modules.

1. **Top-down integration:** This testing focuses on the function of the module integration either depthwise or breadthwise. Depth-wise integration is a vertical partitioning and widthwise integration is horizontal partitioning.
2. **Bottom-up integration:** In this testing, the integration starts from leaf node and forwards to the root node. In this testing, the stubs and drivers are maintained.

9.7.4.7 Regression Testing

When a new module is added, there is a possibility of introducing new functionality to the existing system. Due to the new functionality, there is a possibility of errors occurring in the existing system. To cover those errors, all test cases are re-conducted. This is called regression testing.

9.7.4.8 Smoke Testing

This testing is used in the wrap-shrink component development. Wrap-shrink component development means within the short period of time the software is implemented. To perform this we use modularity and by adding or deleting the existing module we can develop the final product. In this way, smoke testing is used to cover the error.

9.7.4.9 Unit Testing

This testing is performed at the coding stage. After developing the module, use the white-box test plan to cover the structural errors in the module.

9.7.4.10 Alpha Testing

After developing the product, this test is conducted by the customer at the developer side to cover the error. In this testing, the developer's presence is there.

9.7.4.11 Beta Testing

After deploying the product to the customer side, this test is conducted by the customer or end user at the customer place. In this testing, the developer's presence is not there.

9.7.4.12 System Testing

After developing the product, various system-level testing operations are performed to check the reliability of the product.

1. **Recovery testing:** This testing focuses on the recovery of data due to crashes or power failure.
2. **Security testing:** This testing focuses on unauthorised accessing of product.
3. **Stress testing:** This testing is focused on load balancing. It is also known as volume testing.
4. **Performance testing:** This testing is a user acceptance testing in terms of quality and reliability.

IMPORTANT FORMULAS

1. Size $S = \dfrac{S_{\text{opt}} + 4S_{\text{m}} + S_{\text{pess}}}{6}$
2. FP = Count value × VAF
   where $\text{VAF} = 0.65 + 0.01 \times \sum_{P=1 \text{ to } 14} F_i$
3. According to the SEL empirical model, the effort and duration can be estimated as
   $E = 1.4\ (\text{KLOC})^{0.93}$ person-month (units)
   $D = 4.6\ (\text{KLOC})^{0.26}$ months (units)
4. According to Walstonald-Felio method (W-F) (IBM):
   $E = 5.2\ (\text{KLOC})^{0.91}$person-months
   $D = 4.1\ (\text{KLOC})^{0.36}$
5. COCOMO model
   $\text{Effort} = c_1 \times \text{EAF} \times (\text{size})^{c_2}$ (person-months)
   $\text{Duration} = c_3 \times (\text{Effort})^{c_4}$ (months)
6. RE = $P \times C$
   where P is the probability of risk becoming real and C is the cost.
7. Cyclomatic complexity
   $V(G) = E - N + 2P$, where E = number of edges, N = number of nodes and P = number of disconnected components.
   $V(G)$ = Number of regions present in the graph
   $V(G) = P + 1$, where P = number of predicate nodes

SOLVED EXAMPLES

1. The most important feature of spiral model is

   (a) Requirement analysis
   (b) Risk management
   (c) Quality management
   (d) Configuration management

   Solution: Risk management helps in developing a cost effective project where risks are analyzed and risk strategies are decided upon.

   Ans. (b)

2. Cyclomatic complexity of a flow graph G with n vertices and e edges is

   (a) $V(G) = e + n - 2$ (b) $V(G) = e - n + 2$
   (c) $V(G) = e + n + 2$ (d) $V(G) = e - n - 2$

   Solution: Cyclomatic complexity is a software metric developed by Thomas J. McCabe. It is used to indicate the complexity of a program. It directly measures the number of linearly independent paths through a program's source code. It is also known as conditional complexity. It is defined as $V(G) = e - n + 2$.

   Ans. (b)

3. Testing of software with actual data and in actual environment is called

   (a) Alpha testing (b) Beta testing
   (c) Regression testing (d) None of the above

   Solution: Users or an independent test team at the developer's site performs alpha testing. Alpha testing

is often employed as a form of internal acceptance testing, before the software goes to beta testing.

Beta testing is performed after alpha testing for user acceptance testing.

Regression testing focuses on finding defects after a major code change has occurred.

Ans. (b)

4. Object Request Broker (ORB) is

I. A software program that runs on the client as well as on the application server.
II. A software program that runs on the client side only.
III. A software program that runs on the application server, where most of the components reside.

(a) I, II and III (b) I and II
(c) II and III (d) I only

Solution: ORB manages interaction between clients and servers.

Ans. (d)

5. Key process areas of CMM level 4 are also classified by a process which is

(a) CMM level 2 (b) CMM level 3
(c) CMM level 5 (d) All of the above

Solution: There are five maturity levels in CMM model—(1) Initial, (2) Managed, (3) Defined, (4) Quantitatively Managed and (5) Optimizing. The organization at level 2 includes level 1. So, level 4 is included in level 5.

Ans. (c)

6. Validation means

(a) Are we building the product right
(b) Are we building the right product
(c) Certification of fields
(d) None of the above

Solution: Verification means are we building the product right, whereas validation means whether we are building the right product.

Ans. (b)

7. Function points can be calculated by

(a) UFP × CAF (b) UFP × FAC
(c) UFP × Cost (d) UFP × Productivity

Solution: Function Point FP = UFP × CAF. UFP stands for Unadjusted Function Point, while CAF stands for Complexity Adjustment Factor.

Ans. (a)

8. Superficially the term "object-oriented", means that, we organise software as a

(a) collection of continuous objects that incorporates both data structure and behaviour.
(b) collection of discrete objects that incorporates both discrete structure and behaviour.
(c) collection of discrete objects that incorporates both data structure and behaviour.
(d) collection of objects that incorporates both discrete data structure and behaviour.

Solution: This is the definition of object-oriented according to software engineering.

Ans. (c)

9. Which one of the following is not a key process area in CMM level 5?

(a) Defect prevention
(b) Process change management
(c) Software product engineering
(d) Technology change management

Solution: The key process areas at level 5 covers the following issues—Defect Prevention, Technology Change Management and Process Change Management. Software Product Engineering has been addressed in level 3.

Ans. (c)

10. Which of the following system components is responsible for ensuring that the system is working to fulfil its objective?

(a) Input (b) Output
(c) Feedback (d) Control

Solution: The components of the control system ensure smooth functioning of the system to achieve its objective.

Ans. (d)

11. Modifying the software to match changes in the ever-changing environment is called

(a) Adaptive maintenance
(b) Corrective maintenance
(c) Perfective maintenance
(d) Preventive maintenance

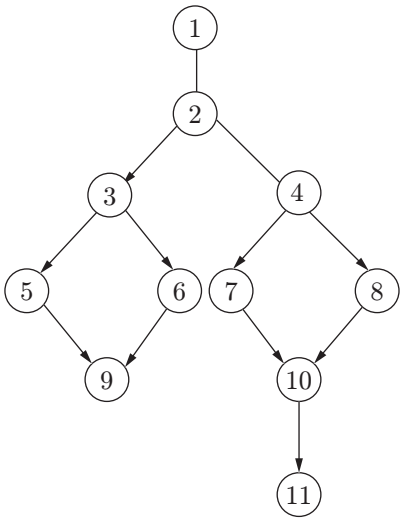
Solution: Corrective maintenance refers to modifications initiated by defects in the software.

Perfective maintenance refers to improving processing efficiency or performance, or restructuring the software to improve changeability.

Preventive maintenance refers to prevention of breakdowns.

Ans. (a)

12. Consider the following flow graph and calculate the cyclomatic complexity and also identify the independent path.



Solution: Cyclomatic complexity $= E - N + 2$
$= 12 - 11 + 2 = 3$

OR
Number of regions = 3
So, cyclomatic complexity = 3

Ans. (3)

13. Software measurement is useful to

(a) indicate quality of the product
(b) track progress
(c) form a baseline for estimation and prediction
(d) all of the above.

Solution: Software measurement is useful to indicate quality of the product, track progress, assess productivity and form a baseline for estimation and prediction.

Ans. (d)

14. Which of the following metric does not depend on the programming language used?

(a) Line of code (b) Function count
(c) Member of token (d) All of the above

Solution: Line of code and tokens depends upon the programming language used for developing programs.

Ans. (b)

15. If in a software project the number of user input, user output, enquiries, files and external interfaces are (15, 50, 24, 12, 8), respectively, with complexity average weighing factor. Find productivity if effort = 70 person-month.

Solution:
$15 \times 4 = 60$
$50 \times 5 = 50$
$24 \times 4 = 96$
$12 \times 10 = 120$
$8 \times 7 = 56$
Count value = 382
Productivity = Count value $\times$ $(0.65 + 42 \times 0.01)$
$= 382 \times (0.65 + 0.42)$
$= 382 \times (1.07)$
$= 408.74$

Ans. (408.74)

16. 58000 LOC gaming software is developed, with the effort of 3 person-year. What is the productivity of person-month?

(a) 1.9 KLOC (b) 1.6 KLOC
(c) 4.8 KLOC (d) 4.2 KLOC

Solution: $58000/36 = 1611.1 = 1.6$ KLOC

Ans. (b)

GATE PREVIOUS YEARS' QUESTIONS

1. The coupling between different modules of a software is categorized as follows:

I. Content coupling
II. Common coupling
III. Control coupling
IV. Stamp coupling
V. Data coupling

Coupling between modules can be ranked in the order of strongest (least desirable) to weakest (most desirable) as follows:

(a) I-II-III-IV-V
(b) V-IV-III-II-I
(c) I-III-V-II-IV
(d) IV-II-V-III-I

(GATE 2009: 1 Mark)

Solution: The types of coupling from least desirable to most desirable are as follows:

Content coupling → Common coupling → Control coupling → Stamp coupling → Data coupling.

Ans. (a)

2. Which of the following statements are TRUE?

I. The context diagram should depict the system as a single bubble.
II. External entities should be identified clearly at all levels of DFDs.
III. Control information should not be represented in a DFD.
IV. A data store can be connected to either another data store or an external entity.

(a) II and III (b) II and III
(c) I and III (d) I, II and III

(GATE 2009: 2 Marks)

Solution: The context diagram depicts the system as a single bubble. Control information should not be represented in DFD.

Ans. (c)

3. Consider the following statements about the cyclomatic complexity of the control flow graph of a program module. Which of these are TRUE?

I. The cyclomatic complexity of a module is equal to the maximum number of linearly independent circuits in the graph.
II. The cyclomatic complexity of a module is the number of decisions in the module plus one, where a decision is effectively any conditional statement in the module.
III. The cyclomatic complexity can also be used as a number of linearly independent paths that should be tested during path coverage testing.

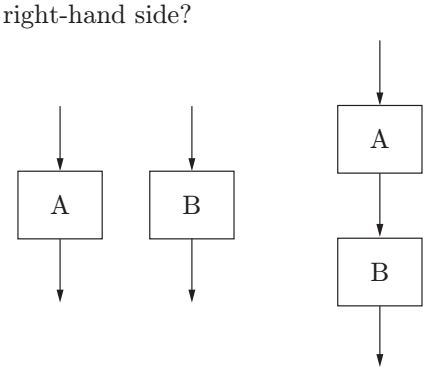
(a) I and II (b) II and III
(c) I and III (d) I, II and III

(GATE 2009: 2 Marks)

Solution: The cyclomatic complexity of a module is not equal to the maximum number of linearly independent circuits in the graph.

Ans. (b)

4. The cyclomatic complexity of each of the modules A and B shown below is 10. What is the cyclomatic complexity of the sequential integration shown on the right-hand side?



(a) 19 (b) 21
(c) 20 (d) 10

(GATE 2010: 1 Mark)

Solution: Cyclomatic complexity = Number of decision points + 1

Number of decision points in A = 10 − 1 = 9

Number of decision points in B = 10 − 1 = 9

Cyclomatic complexity is = (9 + 9) + 1 = 19

Ans. (a)

5. What is the appropriate pairing of items in the two columns listing various activities encountered in a software life cycle?

Column I	Column II
P. Requirements capture	1. Module development and integration
Q. Design	2. Domain analysis
R. Implementation	3. Structural and behavioural modelling
S. Maintenance	4. Performance tuning

(a) P-3, Q-2, R-4, S-1
(b) P-2, Q-3, R-1, S-4
(c) P-3, Q-2, R-1, S-4
(d) P-2, Q-3, R-4, S-1

(GATE 2010: 1 Mark)

Solution: Requirement capture: Domain analysis
Design: Structural and behavioural modelling
Implementation: Module development and integration
Maintenance: Performance tuning

Ans. (b)

6. The following program is to be tested for the statement coverage:

```
begin
if (a==b) {S1; exit;}
     else if (c==d) {S2;}
            else {S3; exit;}
S4;
end
```

The test cases T1, T2, T3 and T4 given below are expressed in terms of the properties satisfied by the values of variables a, b, c and d. The exact values are not given.

T1: a, b, c and d are all equal

T2: a, b, c and d are all distinct

T3: a = b and c != d

T4: a != b and c = d

Which of the test suites given below ensures coverage of statements S1, S2, S3 and S4?

(a) T1, T2, T3 (b) T2, T4
(c) T3, T4 (d) T1, T2, T4

(GATE 2010: 2 Marks)

Solution:
T1 covers S1
T2 covers S3
and T4 covers S2 and S4.
So, the test suit is T1,T2 and T4

Ans. (d)

7. Which one of the following is NOT desired in a good SRS document?

(a) Functional requirements
(b) Non-functional requirements
(c) Goals of implementation
(d) Algorithms for software implementation

(GATE 2011: 1 Mark)

Solution: SRS document is prepared at the starting phase of software development cycle and only client software requirement is noted down. Algorithm for software implementation is designed in later stages after the requirement phase.

Ans. (d)

8. A company needs to develop digital signal processing software for one of its newest inventions. The software is expected to have 40000 lines of code. The company needs to determine the effort in person-months needed to develop this software using the basic COCOMO model. The multiplicative factor for this model is given as 2.8 for the software development on embedded systems, while the exponentiation factor is given as 1.20. What is the estimated effort in person-months?

(a) 234.25 (b) 932.50
(c) 287.80 (d) 122.40

(GATE 2011: 1 Mark)

Solution: Estimated effort in person-months = $a_1(\text{KLOC})^{a2}$
Given that $a_1 = 2.8$ and $a_2 = 1.20$
So, KLOC = 40
$2.8\ (40)^{1.20} = 234.25$ person-months

Ans. (a)

9. A company needs to develop a strategy for software product development for which it has a choice of two programming languages L1 and L2. The number of lines of code (LOC) developed using L2 is estimated to be twice the LOC developed with L1. The product will have to be maintained for five years. Various parameters for the company are given in the table below.

Parameter	Language L1	Language L2
Man years needed for development	LOC/10000	LOC/10000
Development cost per man year	$$1000000	$$750000
Maintenance time	5 years	5 years
Cost of maintenance per year	$$1000000	$$50000

Total cost of the project includes cost of development and maintenance. What is the LOC for L1 for which the cost of the project using L1 is equal to the project using L2?

(a) 4000 (b) 5000 (c) 4333 (d) 4667

(GATE 2011: 1 Mark)

Solution: Number of lines of code for L1 = l
Number of lines of code for L2 =$2l$
Total cost for L1 = Total cost for L2

$$\frac{l}{10000} \times 1000000 + 5 \times 100000$$
$$= \frac{2l}{10000} \times 750000 + 5 \times 50000$$

Solving, we get $l = 5000$

Ans. (b)

10. The following is the comment written for a C function:

```
/*This function computes the roots of a
quadratic equation a.x^2 + b.x + c = 0.
The function stores two real roots in
*root1 and *root2, and returns the
status of validity of roots. It handles
four different kinds of cases.
  (i) When coefficient a is zero irre-
      spective of discriminant
 (ii) When discriminant is positive
(iii) When discriminant is zero
 (iv) When discriminant is negative.
Only in cases (ii) and (iii), the stored
roots are valid. Otherwise 0 is stored
in the roots. The function returns 0 when
the roots are valid and -1 otherwise.
The function also ensures root1 > = root2.
int get_QuadRoots (float a, float b, float
c, float *root1, float *root2);
*/
```

A software test engineer is assigned the job of doing black box testing. He comes up with the following test cases, many of which are redundant.

Test Case	Input Set			Expected Output Set		
	a	b	c	root1	root2	Return Value
T1	0.0	0.0	7.0	0.0	0.0	−1
T2	0.0	1.0	3.0	0.0	0.0	−1
T3	1.0	2.0	1.0	−1.0	−1.0	0
T4	4.0	−12.0	9.0	1.5	1.5	0
T5	1.0	−2.0	−3.0	3.0	−1.0	0
T6	1.0	1.0	4.0	0.0	0.0	−1

Which one of the following options provide the set of non-redundant tests using equivalence class partitioning approach from input perspective for black box testing?

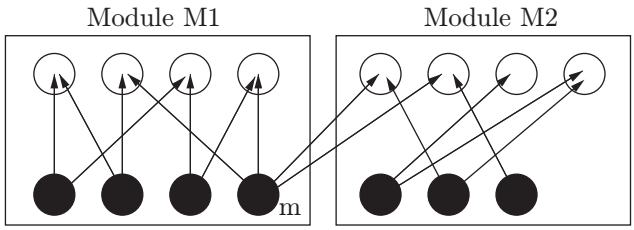
(a) T1, T2, T3, T6
(b) T1, T3, T4, T5
(c) T2, T4, T5, T6
(b) T2, T3, T4, T5

(GATE 2012: 2 Marks)

Solution
T1 and T2 checks the same condition of a = 0
T3 and T4, both have D = 0
These should not be in the same set.

Ans. (c)

11. The following figure represents access graphs of two modules M_1 and M_2. The filled circles represent methods and the unfilled circles represent attributes. If method *m* is moved to module M_2 keeping the attributes where they are, what can we say about the average cohesion and coupling between modules in the system of two modules?



(a) There is no change.
(b) Average cohesion goes up but coupling is reduced.
(c) Average cohesion goes down and coupling also reduces.
(d) Average cohesion and coupling increase.

(GATE 2013: 2 Marks)

Solution: Coupling = Number of external links/Number of modules

Cohesion = Number of internal links/Number of methods

Before modification, we have
Coupling = 2/2 =1
Cohesion for M_1 = 8/4 =2
Cohesion for M_2 = 6/3 =2

After modification, we have

Coupling = 2/2 =1
Cohesion for M_1 = 6/3 =2
Cohesion for M_2 = 8/4 =2
Hence, there is no change.

Ans. (a)

Common Data Questions 12 and 13: The procedure given below is required to find and replace certain characters inside an input character string supplied in array `A`. The characters to be replaced are supplied in array `oldc`, while their respective replacement characters are supplied in array `newc`. Array `A` has a fixed length of five characters, while arrays `oldc` and `newc` contain three characters each. However, the procedure is flawed.

```
void find_and_replace (char *A, char *oldc, char *newc){
    for (int i=0; i<5; i++)
        for (int j=0; j<3; j++)
            if (A[i] == oldc[j]) A[i] = newc[j];
}
```

The procedure is tested with the following four test cases.

(1) `oldc = "abc", newc = "dab"`
(2) `oldc = "cde", newc = "bcd"`
(3) `oldc = "bca", newc = "cda"`
(4) `oldc = "abc", newc = "bac"`

(GATE 2013: 2 Marks)

12. The tester now tests the program on all input strings of length five consisting of characters `'a'`, `'b'`, `'c'`, `'d'` and `'e'` with duplicates allowed. If the tester carries out this testing with the four test cases given above, how many test cases will be able to capture the flaw?

(a) Only one
(b) Only two
(c) Only three
(d) All four

Solution
The test cases 3 and 4 are the only cases that capture the flaw. The code does not work properly when an old character is replaced by a new character, and the new character is again replaced by another new character. This does not happen in test cases (1) and (2), it happens only in cases (3) and (4).

Ans. (b)

13. If array `A` is made to hold the string `"abcde"`, which of the above four test cases will be successful in exposing the flaw in this procedure?

(a) None (b) 2 only
(c) 3 and 4 only (d) 4 only

Solution
Test case `oldc = "abc", newc = "bac"` will detect the fault.

Ans. (c)

14. In the context of modular software design, which one of the following combinations is desirable?

(a) High cohesion and high coupling
(b) High cohesion and low coupling
(c) Low cohesion and high coupling
(d) Low cohesion and low coupling

Solution: Cohesion measures the internal dependency within module, whereas coupling presents intermodule dependence. So, there should be high cohesion and low coupling.

Ans. (b)

PRACTICE EXERCISES

Set 1

1. SRS is also known as specification of

(a) White box testing (b) Stress testing
(c) Integrated testing (d) Black-box testing

2. SRD stands for

(a) Software requirements definition
(b) Structured requirements definition
(c) Software requirements diagram
(d) Structured requirements diagram

3. The symbol represents

(a) mandatory 1 cardinality
(b) mandatory many cardinality
(c) optional 0 or 1 cardinality
(d) optional zero-many cardinality

4. Software testing is

(a) The process of establishing that errors are not present.
(b) The process of establishing confidence that a program does what it is supposed to do.
(c) The process of executing a program to show that it is working as per specifications.
(d) The process of executing a program with the intent of finding errors.

5. All activities lying on critical path have slack time equal to

(a) 0 (b) 1
(c) 2 (d) None of above

6. A software agent is defined as

I. Software developed for accomplishing a given task.
II. A computer program which is capable of acting on behalf of the user in order to accomplish a given computational task.
III. An open source software for accomplishing a given task.

(a) I (b) II
(c) III (d) All of the above

7. In terms of a system, finished products and information are examples of:

(a) Input (b) Output
(c) Feedback (d) Control

8. For a well-understood data processing application, it is best to use

(a) the waterfall model
(b) the prototyping model
(c) the evolutionary model
(d) the spiral model

9. A COCOMO model is

(a) Common Cost Estimation Model
(b) Constructive Cost Estimation Model
(c) Complete Cost Estimation Model
(d) Comprehensive Cost Estimation Model

10. IEEE 830-1993 is an IEEE recommended standard for

(a) Software requirement specification
(b) Software design
(c) Testing
(d) Both (a) and (b)

11. All the modules of the system are integrated and tested as complete system in the case of

(a) Bottom-up testing (b) Top-down testing
(c) Sandwich testing (d) Big-Bang testing

12. If the objects focus on the problem domain, then we are concerned with

(a) Object-Oriented Analysis
(b) Object-Oriented Design
(c) Object-Oriented Analysis and Design
(d) None of the above

13. A fault simulation testing technique is

(a) Mutation testing (b) Stress testing
(c) Black-box testing (d) White-box testing

14. If every requirement stated in the SRS has only one interpretation, SRS is said to be

(a) Correct (b) Unambiguous
(c) Consistent (d) Verifiable

15. The model in which the requirements are implemented by category is

(a) Evolutionary development model
(b) Waterfall model
(c) Prototyping
(d) Iterative enhancement model

16. Which one of the following is a fault base testing technique?

(a) Unit testing (b) Beta testing
(c) Stress testing (d) Mutation testing

17. Which of the following statements is true?

(a) Abstract data types are the same as classes.
(b) Abstract data types do not allow inheritance.
(c) Classes cannot inherit from the same base class.
(d) Object have state and behaviour.

18. If every requirement can be checked by a cost-effective process, then the SRS is

(a) Verifiable (b) Traceable
(c) Modifiable (d) Complete

19. The feature of the object-oriented paradigm which helps code reuse is

(a) Object (b) Class
(c) Inheritance (d) Aggregation

20. The worst type of coupling is

(a) Data coupling (b) Control coupling
(c) Stamp coupling (d) Content coupling

21. Modules X and Y operate on the same input and output data, then the cohesion is

(a) Sequential (b) Communicational
(c) Procedural (d) Logical

22. The desired level of coupling is

(a) No coupling (b) Control coupling
(c) Common coupling (d) Data coupling

23. Changes made to an information system to add the desired but not necessarily the required features is called

(a) Preventative maintenance
(b) Adaptive maintenance
(c) Corrective maintenance
(d) Perfective maintenance

24. Output comparators are used in

(a) Static testing of single module
(b) Dynamic testing of single module
(c) Static testing of single and multiple modules
(d) Dynamic testing of single and multiple modules

25. Coupling and cohesion can be represented using a

(a) Cause-effect graph (b) Dependence matrix
(c) Structure chart (d) SRS

26. KPA in CMM stands for

(a) Key Process Area
(b) Key Product Area
(c) Key Principal Area
(d) Key Performance Area

27. In the spiral model, 'risk analysis' is performed

(a) In the first loop
(b) In the first and second loop
(c) In every loop
(d) Before using spiral model

28. Each time a defect gets detected and fixed, the reliability of a software product

(a) Increases (b) Decreases
(c) Remains constant (d) Cannot say anything

29. The level at which the software uses scarce resources is

(a) Reliability (b) Efficiency
(c) Portability (d) All of the above

30. Alpha and beta testing are forms of

(a) Acceptance testing (b) Integration testing
(c) System testing (d) Unit testing

31. Which one of the following is not a risk management technique for managing the risk due to unrealistic schedules and budgets?

(a) Detailed multisource cost and schedule estimation
(b) Design cost
(c) Incremental development
(d) Information hiding

32. _____ of a system is the structure or structures of the system which comprise software elements, the externally visible properties of these elements and the relationship amongst them.

(a) Software construction (b) Software evolution
(c) Software architecture (d) Software reuse

33. In function point analysis, the number of complexity adjustment factors is

(a) 10 (b) 12 (c) 14 (d) 20

34. Regression testing is primarily related to

(a) Functional testing (b) Development testing
(c) Data flow testing (d) Maintenance testing

35. Which one of the following is not a definition of error?

(a) It refers to the discrepancy between a computed, observed or measured value and the true, specified or theoretically correct value.
(b) It refers to the actual output of software and the correct output.
(c) It refers to a condition that causes a system to fail.
(d) It refers to human actions that result in software containing a defect or fault.

36. Match the following:

List – I	**List – II**
a. Data coupling	i. Module A and Module B have shared data
b. Stamp coupling	ii. Dependency between modules is based on the fact they communicate by only passing of data
c. Common coupling	iii. When complete data structure is passed from one module to another
d. Content coupling	iv. When the control is passed from one module to the middle of another

(*Continued*)

Codes:

	a	b	c	d
(a)	iii	ii	i	iv
(b)	ii	iii	i	iv
(c)	ii	iii	iv	i
(d)	iii	ii	iv	i

37. A process which defines a series of tasks that have the following four primary objectives is known as

1. To identify all items that collectively define the software configuration.
2. To manage changes to one or more of these items.
3. To facilitate the construction of different versions of an application.

To ensure that software quality is maintained as the configuration evolves over time.

(a) Software Quality Management Process
(b) Software Configuration Management Process
(c) Software Version Management Process
(d) Software Change Management Process

38. Software Configuration Management is the discipline for systematically controlling

(a) The changes due to the evolution of work products as the project proceeds.
(b) The changes due to defects (bugs) being found and then fixed.
(c) The changes due to requirement changes
(d) All of the above

39. Which one of the following is not a step of requirement engineering?

(a) Requirement elicitation
(b) Requirement analysis
(c) Requirement design
(d) Requirement documentation

40. _____ is a process model that removes defects before they can precipitate serious hazards.

(a) Incremental model
(b) Spiral model
(c) Clean room software engineering
(d) Agile model

41. Equivalence partitioning is a _____ method that divides the input domain of a program into classes of data from which test cases can be derived.

(a) White-box testing
(b) Black-box testing
(c) Orthogonal array testing
(d) Stress testing

42. The following three golden rules are for:

(i) Place the user in control
(ii) Reduce the user's memory load
(iii) Make the interface consistent
(a) User satisfaction
(b) Good interface design
(c) Saving system's resources
(d) None of these

43. Software safety is a _____ activity that focuses on the identification and assessment of potential hazards that may affect software negatively and cause an entire system to fail.

(a) Risk mitigation, monitoring and management
(b) Software quality assurance
(c) Software cost estimation
(d) Defect removal efficiency

44. While estimating the cost of software, LOC and function points (FP) are used to measure which one of the following?

(a) Length of code
(b) Size of software
(c) Functionality of software
(d) None of the above

45. The factors that determine the quality of a software system are

(a) Correctness, reliability
(b) Efficiency, usability, maintainability
(c) Testability, portability, accuracy, error tolerances, expandability, access control, audit
(d) All of the above

46. _____ establishes information about when, why and by whom changes are made in software.

(a) Software configuration management
(b) Change control
(c) Version control
(d) An audit trail

47. _____ is an "umbrella" activity that is applied throughout the software engineering process.

(a) Debugging
(b) Testing
(c) Designing
(d) Software quality assurance

48. Are we building the right product?

This statement refers to

(a) Verification
(b) Validation
(c) Testing
(d) Software quality assurance

49. Main aim of software engineering is to produce

(a) program
(b) software
(c) within budget
(d) software within budget in the given schedule

Set 2

1. Company needs to develop digital signal processing software for one of its newest invention. The software is expected to have 36000 LOC. The company needs to determine the efforts in person-month needed to develop this software using the basic COCOMO model.

The multiplicative factor for this model is given as 2.6 for the software development on the embedded system while exponential factor is given as 1.30. The estimated effort in person-month is _____.

2. The relationship between a derived class (or subclass) and base class is referred to as

(a) Association
(b) Inheritance
(c) Polymorphism
(d) Instantiation
(e) Aggregation

3. A design is said to be a good design if the components are

(a) Strongly coupled
(b) Strongly coupled and weakly cohesive
(c) Strongly coupled and strongly cohesive
(d) Strongly cohesive and weakly coupled

4. If a control switch is passed as an argument, this is an example of _____ coupling.

(a) Content
(b) Common
(c) Control
(d) Stamp
(e) Data

5. The aim of software engineering is to produce software that is

(a) Fault-free
(b) Delivered on time
(c) Delivered within budget
(d) All of these are the aims of software engineering

6. A systematic approach to software development, as epitomised by the various life-cycle models, is useful in

(a) Helping us understand the nature of the software product
(b) Convincing the customer that we know what we are doing
(c) Filling texts on software engineering
(d) Managing the various activities necessary to get the job done

7. A process view in software engineering would consider which of the following

(a) Product performance
(b) Staffing
(c) Functionality
(d) Reliability
(e) Usability

8. Which of the following is not a 'concern' during the management of a software project?

(a) Product quantity
(b) Time
(c) Project/Product information
(d) Money

9. Black-box testing is also called

(a) Specification-based testing
(b) Structural testing
(c) Verification
(d) Unit testing
(e) Stress testing

10. Project risk factor is considered in?

(a) Spiral model
(b) Waterfall model
(c) Prototyping model
(d) Iterative enhancement model

ANSWERS TO PRACTICE EXERCISES

Set 1

1. (d)	**10.** (a)	**19.** (c)	**28.** (a)	**37.** (b)	**46.** (d)
2. (b)	**11.** (d)	**20.** (a)	**29.** (b)	**38.** (d)	**47.** (d)
3. (d)	**12.** (a)	**21.** (b)	**30.** (a)	**39.** (c)	**48.** (b)
4. (d)	**13.** (a)	**22.** (d)	**31.** (d)	**40.** (c)	**49.** (d)
5. (a)	**14.** (b)	**23.** (d)	**32.** (c)	**41.** (b)	
6. (b)	**15.** (a)	**24.** (d)	**33.** (c)	**42.** (b)	
7. (b)	**16.** (d)	**25.** (b)	**34.** (d)	**43.** (b)	
8. (a)	**17.** (b)	**26.** (a)	**35.** (c)	**44.** (b)	
9. (b)	**18.** (a)	**27.** (c)	**36.** (b)	**45.** (d)	

Set 2

1. (274.26) Effort = 2.6 EAF × $(36k)^{1.30}$ (EAF = by default =1)

$$= 2.6 \times (36k)^{1.30}$$
$$= 2.6 \times (36)^{1.20}$$
$$= 274.26$$

2. (b) A derived class inherits all the attributes and property of parent class (base class).

3. (d) The aim is to minimize the interaction between modules and maximize interaction within a module.

4. (c) Two modules are control coupled if one passes an element of control to another.

5. (d) The aim of software engineering is to produce software that is fault-free, delivered on time, delivered within budget and satisfies users' needs.

6. (d) A systematic approach to software development, as epitomised by the various life-cycle models, is useful in managing the various activities necessary to get the job done.

7. (b) Staffing is the apt choice among the given

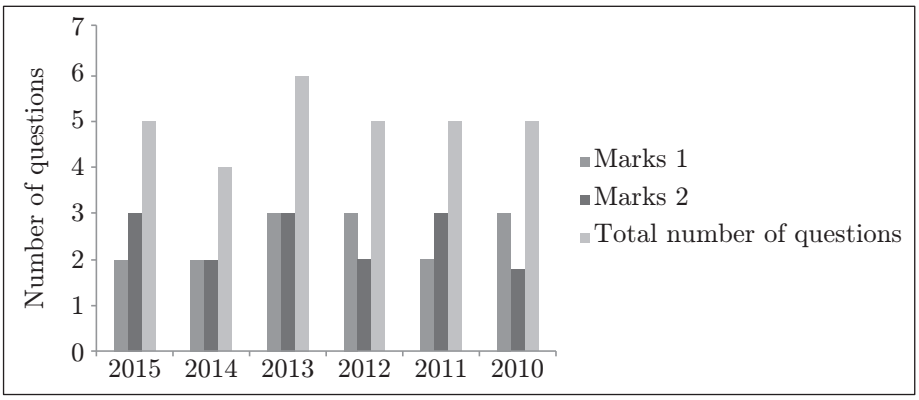
8. (a) Product quantity would not include during the management of a software project.

9. (a) Black-box testing is another name for specification-based testing.

10. (a) Aim of risk analysis phase in the spiral model is to eliminate the high-risk problems before they threaten the project operation or cost.

UNIT X: COMPUTER NETWORKS

LAST SIX YEARS' GATE ANALYSIS



Concepts on which questions were asked in the previous six years

Year	Concept
2015	TCP, IP Header, File organization, Symmetric key cryptography, LAN, Stop and wait protocol, CSMA/CD, Go-back-N sliding window protocol, IPv4 Addressing, Data transmission, HTTP, UDP datagram, Routing table
2014	IPv4 addresses, MTU, OSI layer, Frames, IP header, Classless inter-domain routing
2013	Transport layer protocols, Data link layer concepts, IPv4 concepts
2012	IPv4 concepts, Transport layer concepts, Application layer concepts, IP address, Propagation delay in networks
2011	Firewall concept, Distance vector, Application layer concepts, HTML
2010	IP address concepts, Application layer, Router concepts

CHAPTER 10

COMPUTER NETWORKS

Syllabus: Computer networks: ISO/OSI stack, LAN technologies (Ethernet, token ring), Flow-and error-control techniques, Routing algorithms, Congestion control, TCP/UDP and sockets, IP(v4), Application-layer protocols (ICMP, DNS, SMTP, POP, FTP, HTTP); Basic concepts of hubs, switches, gateways and routers; Network security: basic concepts of public key and private key cryptography, digital signature, firewalls.

10.1 INTRODUCTION

Computer network is a collection of autonomous devices interconnected via a medium. The medium may be a guided medium, a wireless medium or a satellite communication. To understand data communication, different layered architectures have been presented. All layers execute different protocols to communicate the data successfully from one end to the other. The intermediary devices such as switch, hub, router or gateway help in advancing this communication. Although the security requirements may be different depending upon the application, network security cannot be left aside when studying networks.

10.2 NETWORK

Two or more devices connecting to each other through any medium forms a network. To connect the devices there are two possible ways:

1. **Point-to-point connection:** This provides a dedicated link between the two devices.
2. **Multipoint connection:** This is also known as multi-drop connection. In this case, channel capacity is shared by more than two devices. In general, this is used in practice. All the five examples in Fig. 10.1 are multipoint.

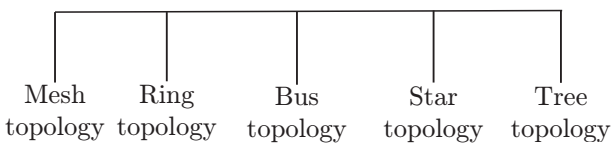


Figure 10.1 | Network topology.

10.2.1 Network Topology

The physical or logical view of interconnection among devices is known as topology.

Ring and mesh topology are examples of peer-to-peer relationship because here all the devices share the link equally. Bus, star and tree topologies are examples of primary—secondary relationship, where one device controls and the other devices have to transmit through it.

10.2.1.1 Bus Topology

In a bus network, all the devices are connected with one cable. The major benefit here is, we require less cable length than star topology and expansion is quite easy with the help of repeaters.

The issues associated with bus topology are as follows:

1. Only one device can send the data at one time. All the devices will listen at that time.
2. The data communication is only in one direction.
3. In a bus topology, if the network shuts down then there is problem in identifying the culprit device.

10.2.1.2 Ring Topology

In a ring topology, each deviceis connected exactly to two devices to form the ring. Repeaters are used to regenerate and retransmit each bit. Data travels around the network in one direction. Data travels in the form of token. Additional components do not affect the performance of the network. Even if the load on the network increases, the performance of a ring topology is better than a bus topology. The problems may be listed as follows:

1. Failure of one computer in the ring may lead to entire communication loss.
2. Network scaling is difficult.

For example, token ring is defined by IEEE 802.5 standard.

10.2.1.3 Star Topology

In a star topology, a hub is placed at the central location and all the devices are connected to the hub. All communication is possible through the hub only. If the hub is active, it may amplify or regenerate the signals. The following are the drawbacks of a star topology:

1. If the central hub fails, no communication is possible.
2. Cabling cost is more.

For example, Ethernet 10 base T is a popular example.

10.2.1.4 Mesh Topology

All the devices are connected through peer-to-peer links. A fully connected mesh will have $n(n-1)/2$ physical channels to connect n devices. The advantage of mesh topology is security and privacy. A dedicated link will eliminate traffic problems, and fault diagnose is easy here. But cabling cost and other hardware required make it difficult to implement in real practice. It is better to use this topology in backbone network and other topologies for further network configuration.

10.2.1.5 Tree Topology

A tree topology maintains the devices connected to a central hub as well as to some secondary hubs, which are again connected to the central hub. It allows an isolated network which prioritizes communication from different computers. It also faces the same problem as in a star topology; failure of central hub will crash the entire network. Also, it has high cabling cost.

10.3 LAN TECHNOLOGIES

LAN (local area network) is an integral part to create a network. There are many LAN technologies such as Ethernet, token ring, token bus, FDDI (fiber distributed data interface) and ATM LAN. Table 10.1 shows the comparison of Ethernet, token bus and token ring.

10.3.1 Ethernet

Xerox Corporation, Digital Equipment Corporation and Intel Corporation developed Ethernet LAN technology in 1976. Ethernet is based on the IEEE 802.3 specification. It is a linear-bus logical topology. Ethernet is the most widely used LAN technology in the world. It has passed four generations: Standard Ethernet (10 t Mbps), Fast Ethernet (100 Mbps), Gigabit Ethernet (1 Gbps) and Ten-Gigabit Ethernet (10 Gbps). Earlier Ethernet designed were of two types: Thicknet (thick coaxial main trunk cable of 10 mm and Thinnet (thin coaxial cable: RG-58 of 5 mm). In 1990, unshielded twisted-pair (10Base-T) Ethernet came into existence. Ethernet uses

Table 10.1 | Comparison of Ethernet, token bus and token ring

Attribute	Ethernet	Token Bus	Token Ring
Physical Topology	Linear	Linear	Star
Logical Topology	None	Ring	Ring
Connection	Random	By token	By token
Node Addition	Node added anywhere and anytime	Distributed algorithms are responsible for node addition	Between two specified nodes
Cable Used	Twisted pair, co-axial and fibre optic	Co-axial	Twisted pair and fibre optic
Cable Length	50–2000 m	200–500 m	50–1000 m
Frequency	10–100 Mbps	10 Mbps	4–100 Mbps
Frame Structure	1500 byte	8191 bytes	5000 bytes
IEEE Standard	802.3	802.4	802.5
Maintenance	No central maintenance	By distributed algorithms	By a designated monitor node
Performance	Immediately transmitted by the nodes, heavy traffic can reduce the effectiveness of transmission	Nodes must wait for the token if no other node is transmitting. During heavy traffic, token passing provides fair access to all nodes	Nodes must wait for token even if no other node is transmitting. During heavy traffic, token passing provides fair access to all nodes
Maximum Delay before Transmitting	None	Bounded, depending on distance spanned and number of nodes	Bounded, depending on distance spanned and number of nodes

Carrier-Sense Multiple Access with Collision Detection (CSMA/CD) access control scheme. The drawback of this topology is that if one of the links between the two adjacent nodes fails, the whole network fails.

10.3.2 Token Bus

Token bus is a bus (physical view) ring (logical view) topology. In token bus, nodes are connected linearly. However, they make a logical ring, as each node knows the address of its successor.

10.3.3 Token Ring

Token ring is a star (physical view) ring (logical view) topology. The hub acts as a connector. The wiring inside the hub makes the ring; the stations are connected to this ring through the two wire connections. It is more efficient; if a link goes down, it will bypass the hub and operate the other nodes. It also improves the scalability of the network.

In early token release,

$$\text{Throughput for single station or } N \text{ stations} = \frac{\text{Data}}{\text{Transmission time} + (\text{Ring latency}/\text{Number of stations})}$$

In delayed token release,

$$\text{Throughput for single station} = \frac{\text{Data}}{\text{Transmission time} + \text{Ring latency} + (\text{Ring latency}/\text{Number of stations})}$$

$$\text{Throughput for } N \text{ stations} = \frac{\text{Data}}{\text{Ring latency} + (\text{Ring latency}/\text{Number of stations})}$$

10.4 ISO/OSI STACK

There are two reference models based on the network architectures. One is the International Standards Organization/Open Systems Interconnection (Reference Model 1984) (ISO/OSI) model and other is the Transmission Control Protocol/Internet Protocol (TCP/IP) model. The layered architecture of both the models has been compared in Fig. 10.2.

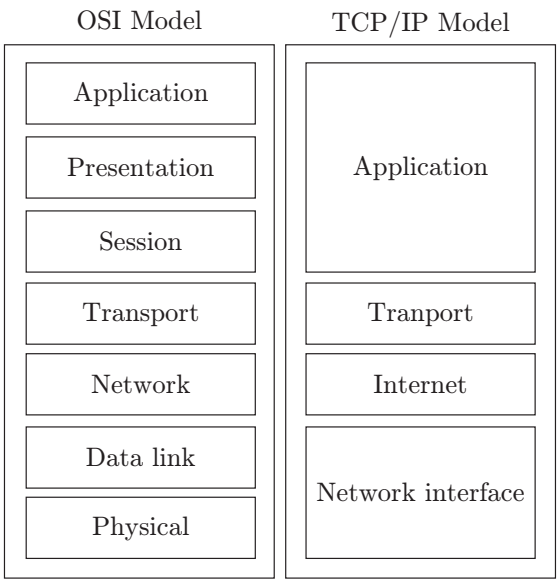


Figure 10.2 | OSI and TCP/IP model.

The ISO/OSI model has seven layers while TCP/IP has only four layers. Session and presentation layers are completely missing. Characteristics of the session layer are provided by the transport layer, and the application layer bears the accountability of the presentation layer. Host to network is the lowest layer in the TCP/IP model and its duty is to send IP packets to the network. In the ISO/OSI model, the network layer provides connection-oriented as well as connectionless services, but the TCP/IP model provides only connectionless services.

If M is a message and H is the header that is added at every layer and N layers are present in hierarchy, then the fraction of header that is passed in the total content is calculated as follows:

$$\text{Fraction of data} = \frac{M}{NH + M}$$

10.4.1 ISO/OSI Model

10.4.1.1 Layer 1: Physical Layer

The major responsibilities of physical layer are transmission of raw bit stream and to form the physical interface between two communicating devices. The conversion of analog to digital and digital to analog is performed at this layer.

The following are some issues listed which may create problems before/during transmission:

1. Compatibility of mechanical and electrical interfaces
2. Working of physical transmission media
3. Deciding on the number of bits per second to be sent
4. Finding out whether transmission is simplex or duplex
5. Establishing and terminating the initial communication when both sender and receiver are finished

$$\text{Transmission time} = \frac{\text{Message size (bits)}}{\text{Bandwidth (bits/s)}}$$

$$\text{Propagation time} = \frac{\text{Distance}}{\text{Velocity}}$$

Problem 10.1: Message size = 1 Kb

Bandwidth = 1 Mbps

$$\text{Transmission time} = \frac{1\text{ Kb}}{1\text{ Mbps}} = \frac{10^3 \times 2^3 \text{bits}}{10^6 \text{bits/s}} = 8 \times 10^{-3}\text{ s}$$

If link utilisation is 50%, what is the relation between transmission time and propagation time?

Solution:

Link utilization of sender

$$= \frac{\text{Transmission time}}{\text{Transmission time} + 2 \times \text{Propagation time}}$$

$$\frac{1}{2} = \frac{\text{Transmission time}}{\text{Transmission time} + 2 \times \text{Propagation time}}$$

Transmission time + 2 × Propagation time
= 2 × Transmission time

Therefore, Transmission time = 2 × Propagation time

$$\text{Transmission time} = 2 \times \text{Propagation time}$$

or $\quad$ Transmission time = Round-trip time

Let L be the message, B the bandwidth and R the round-trip time (RTT),

$$\text{Link utilization of sender} = \frac{L/B}{(L/B) + R}$$

$$\eta = \frac{L}{L + BR}$$

In the above expression,

1. if L = BR, $\eta = 50\%$
2. if $L > BR$, $\eta > 50\%$
3. if $L < BR$, $\eta < 50\%$
4. if $L >>> BR$, $\eta \simeq 100\%$
5. if $L <<< BR$, $\eta \simeq 0\%$

10.4.1.2 Layer 2: Data Link Layer

Data link layer provides reliable transfer of information between two adjacent nodes. Also, it provides frame-level error control and flow control. It provides

communication between machines on the same network. Communication between two devices can be simplex, half-duplex, or full-duplex. In simplex, the communication is unidirectional. In half-duplex, each device can both transmit and receive, but not at the same time. In full-duplex, both devices can transmit and receive simultaneously.

This layer is responsible for encoding and decoding i.e. converting bits to signals at sender site and recovering bits from received signals at receiver side; frame creation i.e. deciding a minimum unit for sending bits; error detection and/or correction of frames through parity or CRC and flow control using ARQ, sliding WINDOW etc. The functionality of data link layer is as follows:

1. **Encoding:** Signals propagate over a physical medium – (1) modulate electromagnetic waves (varying voltage); (2) encode binary data onto signals (e.g., 0 as low signal or non-return to zero, NRZ, and 1 as high signal or non-return to zero inverted, NRZI); make a transition from current signal to encode a 1 or stay at the current signal to encode a 0;(3) Manchester (transmit XOR of the NRZ-encoded data and the clock only 50% efficient).
   In Manchester encoding, a clock signal and data signal are mixed together by XORing operation. The clock makes a clock transition in every bit time, so it runs at twice the bit rate. When it is XORed with 0 then it makes low-to-high transition, it acts as a clock and when it is XORed with 1 then it makes high-to-low transition, it acts as a data signal.
2. **Framing:** The basic data unit at the date link layer is called a 'frame' which is a collection of bits in sequence boundary. In order to mark boundaries of frame starting and ending characters are used.
3. **Flow control:** It is a mechanism which informs the sender about the amount of data transmission before receiving an acknowledgement from the receiver. As the receiving device has limited speed for processing the incoming data and limited memory to store data, so it informs the sending device by sending few frames and stop. The receiving device has a buffer, a block of memory, for storing extra incoming data before processing.
4. **Error control:** It is a mechanism which informs the sender about the retransmission of the damaged and lost frames during transmission. It is a method of error detection and error correction. Automatic repeat request (ARQ) is a process in which whenever an error is detected, the receiving device sends the request for retransmission to sender.
5. **Techniques of flow and error control:**
   - *Stop-and-wait automatic repeat request:* In this protocol, the sender starts the timer and keeps the copy of the sent frame. If the timer expires and there is no acknowledgement (ACK) for the sent frame, the frame is resent, the copy is held and the timer is restarted. For the corrupted and lost ACK frame, sequence numbers can be used. In the data frame, a field is added for the sequence number. The sequence numbers are based on modulo-2 arithmetic. This protocol is also having acknowledgement numbers, which specifies the sequence number of the next frame expected by the receiver. A data frame uses a sequence number and an ACK frame uses an acknowledgement number. The control variable of sender keeps the sequence number for the next frame to be sent (0 or 1). The control variable of receiver keeps the number of the next frame expected. When a frame is sent, the value of the control variable of sender is incremented. When a frame is received, the value of the control variable of receiveris incremented. The stop-and-wait ARQ protocol is very inefficient if the channel is thick (large bandwidth) and long (long round-trip delay). Other drawback of this protocol is that it does not support pipelining, as it does not support multiple frames.Pipelining helps in improving the efficiency of the transmission, if the number of bits in transition is large with respect to the bandwidth-delay product.
     For stop-and-wait ARQ,

$$\text{Transmission time} = \frac{\text{Message size}}{\text{Bandwidth}}$$

$$\text{Propagation delay} = \frac{\text{Distance of the link}}{\text{Velocity}}$$

Link utilisation of sender or throughput is given by

$$\eta = \frac{\text{Transmission time}}{\text{Transmission time} + 2 \times \text{Propagation delay}}$$

   - *Go-back-N automatic repeat request:* This protocol helps in improving the efficiency of transmission by filling the pipe. It supports multiple frames during wait for acknowledgement. The sequence numbers are modulo 2^m, where m is the size of the sequence number field in bits. Sliding window defines the range of sequence numbers related to the sender and receiver. When the timer expires, the sender resends all outstanding frames. Stop-and-wait ARQ is a special case of Go-Back-N ARQ in which the size of the send window is 1. It supports one receiver window size. This protocol is very inefficient for a noisy link. In noisy link, frames are resending again and again, which uses bandwidth and slow down the transmission. For Go-Back-N ARQ,

     Sender window size $< 2m$

     Maximum sequence number $= 2m - 1$

     where m is the number of segment bits.

If maximum sequence number is s, then the number of sequence bits = log $(s+1)$

$$\text{Transmission time} = \frac{\text{Message size}}{\text{Bandwidth}}$$

$$\text{Propagation delay} = \frac{\text{Distance}}{\text{Velocity}}$$

Link utilisation of sender or throughput is given by

$$\eta = \frac{\text{Transmission time}}{\text{Transmission time} + 2 \times \text{Propagation delay}}$$

$$\text{Number of frames (Window size)} = \frac{\text{Total bits}}{\text{Frame size}}$$

- *Selective repeat automatic repeat request:* In this protocol, the damaged frame is resent in the network. It is efficient for noisy links, but it requires complex processing at the receiver end.

  Sender window size
  = Receiver window size $\leq 2^m - 1$

  If Q is the size of the window, then the number of sequence bits = $\log_2 Q + 1$

$$\text{Number of frames (Window size)} = \frac{\text{Total bits}}{\text{Frame size}}$$

Problem 10.2: Calculate the link utilisation for stop-and-wait flow control mechanism if the frame size is 4800 bits, bit rate is 9600 bps and distance between devices is 2000 km. Given propagation speed is 200000 km/s.

Solution:

$$\eta = \frac{4800/9600}{(4800/9600) + 2 \times (2000/200000)} = 0.96$$

6. **Parity bits:** Append a single parity bit to a sequence of bits
   - If using 'odd' parity, the parity bit is calculated as making the total number of 1's in the bit sequence odd;
   - If using 'even' parity, the parity bit makes the total number of 1's in the bit sequence even

   For example, if $-Q$ is for even parity, what's the parity bit for 00010101? The problem with parity bit is that it only detects when there are an odd number of bit errors.
7. **Polynomial codes:** It can detect errors on large chunks of data; has low overhead; is more robust than parity bit; and requires the use of a code polynomial.
8. **Cyclic Redundancy Check (CRC):** Example of a polynomial code

   Procedure:
   - Let r be the degree of the code polynomial. Append r zero bits to the end of the transmitted bit string. Call the entire bit string $S(x)$.
   - Divide $S(x)$ by the code polynomial using modulo-2 division.
   - Subtract the remainder from $S(x)$ using modulo-2 subtraction.

   The result is the checksummed message.
9. **Decoding a CRC procedure:**
   - Let n be the length of the checksummed message in bits.
   - Divide the checksummed message by the code polynomial using modulo-2 division. If the remainder is zero, there is no error detected.
10. **Choosing a CRC polynomial:** The longer the polynomial, the smaller the probability of undetected error.

Problem 10.3: If the frame is 1101011011 and generator is $x^4 + x + 1$, what would be the transmitted frame?

Solution: The polynomial $x^4 + x + 1$ corresponds to divisor 10011($k = 5$ bits)

Data word (1101011011) of N = 10 bits is augmented with $(k - 1)$ zero's.

Dividend = 11010110110000

```
          110000101
10011 ) 11010110110000
        10011
        ------
        010011
         10011
        -------
         0000010110
              10011
        ---------
              0010100
                10011
        ---------
                001110
```

After dividing the message 1101011011 by 10011 the remainder is 1110, which is CRC. The transmitted data is data + CRC, which is 1101011011 + 1110 = 11010110111110.

Data link layer is divided into two sublayers:

1. **Multiple access control sublayer:** It provides controlled access to shared transmission media.
2. **Logical link control sublayer:** It is responsible for error and flow control.

When multiple users share a common communication link, multiple access protocols are used to coordinate access to common link (Fig. 10.3).

1. **Random Access Protocols:** The following are the different random access protocols used for accessing the shared transmission channel:
   - *Pure ALOHA:* It says that whenever a station is having the data, they can send immediately. The time at which the collision occurs is called vulnerable time.

$$T_p = \text{Maximum propagation time} = \frac{\text{Distance between two stations}}{\text{Velocity}}$$

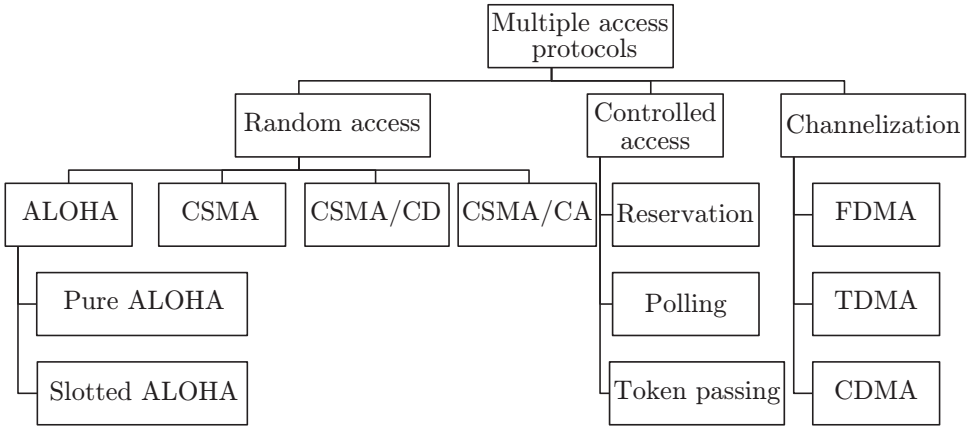


Figure 10.3 | Multiple access protocols.

Suppose T_{fr} is the average transmission time for a frame, then

$$T_{fr} = \frac{\text{Frame size}}{\text{Bandwidth}} = G$$

$$\text{Throughput } (S) = G \times e^{-2G}$$

$$\text{Vulnerable time} = 2 \times T_{fr}$$

$$S_{\max} = 18.4\%$$

- *Slotted ALOHA:* It says that if stations are ready with the data, they have to wait for the required time slot and can transmit data exactly at that timeslot.

$$\text{Throughput } (S) = G \times e^{-G}$$

$$\text{Vulnerable time} = T_{fr}$$

$$S_{\max} = 36.8\%$$

- *CSMA (Carrier Sense Multiple Access):* If a station is ready with the data, it senses the channel, and if channel found idle, data is transmitted, otherwise the station has to wait for random amount of time.

10.4.1.3 Layer 3: Network Layer

The network layer is responsible for host-to-host delivery and path selection between endsystems (routing). The fragmentation, reassembly and translation between different network types are also performed at this layer. In other words, communication between nodes is possible in different networks through this layer.

Packet delivery can be accomplished by using either a connection-oriented or a connectionless network service. In a connection-oriented protocol, the connection is established before sending the packets so route is established before and all the packets have to follow that route. Example: Frame relay and ATM uses this service.

In connectionless protocols, the network layer protocol treats each packet independently. The packets in a message may or may not follow the same path to their destination. For example, Internet uses this type of service.

Switching can be broadly divided into three categories: circuit switching, packet switching and message switching, as shown in Table 10.2.

Table 10.2 | Comparison of circuit, message and packet switching

Attribute	Circuit	Message	Packet
Dedicated Physical Path	Yes	No	No
Bandwidth Available	Fixed		Dynamic
Route Selection	Static	Dynamic (per message)	Dynamic (per packet)
Potentially Wasted Bandwidth	Yes	No	No
Stored and Forward Transmission	No	Stored	Queued not stored
Transmission Length	Unlimited	Maximum length	No maximum length
Same Route Follows	Yes	No	No

(*Continued*)

Table 10.2 | Continued

Attribute	Circuit	Message	Packet
Packets Arrive in Order	Yes		No
Call Setup	Required	Not Required	Not required
Congestion Route Blocking	At setup time, if user busy	No message blocking	On every packet
Possible Reordering	No	No	Yes
Response to Link Failure	Data loss	Rerouting/Retransmission	Rerouting/Retrans-mission
Message Delivery	Guaranteed	Depends on the network	Depends on the network
Delivery Time	Negligible	Long	Short
Path Establishment	Switch path for entire connection time	For each message	For each packet
Charging	Per minute	Per message	Per packet

The **Internet** is a global system of computer networks that are interconnected worldwide and all use the standard Internet protocol suite (TCP/IP) for linking.

1. **Internet as a Datagram Network:** The Internet, at the network layer, is a packet-switched network. Switching can be generally divided into three broad categories: circuit switching, packet switching, and message switching. Packet switching uses either the virtual circuit approach or the datagram approach.

   The Internet chooses the datagram approach to switching in the network layer, and uses the universal addresses defined in the network layer to route packets from the source to the destination. Switching at the network layer in the Internet uses the datagram approach to packet switching.
2. **Why Internet Uses Connectionless Network?** Delivery of a packet can be accomplished by using either a connection-oriented through TCP or a connectionless network service through UDP (see Table 10.3). In a connection-oriented service, the source has to make a connection with the destination before sending a data packet. Only after establishing connection, a sequence of packets from source to the destination can be sent on the same path that is established before in a sequential order. The connection is terminated only when all the packets of a particular message have been successfully. But the communication at the network layer in the Internet is connectionless. The reason is that Internet is made of so many heterogeneous networks that it is almost impossible to create a connection between every source and destination pair without knowing the nature of the networks in advance.

Table 10.3 | Comparison between TCP and UDP

Attribute	TCP	UDP
Connection Management	Connection oriented	Connectionless
End-to-End Connection	Dedicated connection	No dedicated connection
Reliability	Reliable	Unreliable
Transmission	Byte oriented	Message oriented
Acknowledgement	Yes	No
Retransmission	Automatically	If needed
Congestion Control	Yes	No
Flow Control	Yes	No
Fault Tolerance	No	No
Data Delivery	Strictly ordered	Unordered
Security	Yes	Yes
Overhead	Low	Very low
Transmission Speed	High	Very high
Data Quantity Suitability	Small to very large amount of data	Small to moderate amount of data

10.5 ROUTING ALGORITHMS

When the router receives a packet, which route this packet should follow to reach to destination is an important concern. This is one of the major responsibilities of network

layer. This decision is taken by router on the basis of the routing table maintained by it. The algorithm which decides the suitable route is known as routing algorithm. The desirable properties of any routing algorithm are correctness, fairness, stability, robustness and optimality.

These algorithms can be broadly divided into two categories:

1. **Adaptive algorithms:** The dynamic routing decision depends on the topology and traffic. Furthermore, adaptive algorithms have been divided into three forms:
   - ***Centralised:*** The decision is taken on the basis of global information and this is performed by a centralised node.
   - ***Isolated:*** The routing decision is taken based on local information. Generally, routers do not share information with their neighbours.
   - ***Distributed:*** A combination of local and global information.
2. **Non-adaptive algorithms:** The static routing decision is taken in advance and it is downloaded by the routers, that is, never change once initial route has been selected.

The properties of non-adaptive routing algorithms are as follows:

1. **Optimality principal:** It states that "if router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along the same route". In other words, suppose r_1 is the route from I to J and r_2 is rest of the route. Then, if any route is better than r_2, it could have improved the overall optimal route. Hence, r_1r_2 is optimal.
2. **Sink tree:** A set of all optimal routes from any source to a fixed destination form a tree called sink tree. Sink trees may be more than one with the same path length. The concern of sink tree here is to help routers find the best path.

In addition to adaptive and non-adaptive categorisation, routing algorithm can be simply categorised into the following:

1. Static routing (shortest path, flooding)
2. Flow-based routing
3. Dynamic routing (distance vector, link state routing)
4. Hierarchical routing
5. Routing for mobile hosts
6. Broadcast routing
7. Multicast routing

These are explained as follows:

1. **Shortest path routing:** This non-adaptive approach is based on the simplest and most widely used principle. Each node is treated as a router and each arc as communication link. To find a path between a pair of routers, the shortest path is chosen. The shortest path is chosen based on the number of hops or geographical distance in kilometres. Dijkstra's and Bellman–Ford's algorithm are the most famous shortest path algorithms.

Example 10.1

Flooding: This is again a non-adaptive algorithm and it sends a copy of the packet to every outgoing line except the line on which it was received. This guarantees the packet delivery to destination but a large number of packet copies will be generated. Sometimes this count approaches to infinity and the only solution is to discard the packet using any of the following approach:

(a) Using a hop counter to avoid forwarding of packets as number of hops reaches the diameter of the network.
(b) Keeping track of flooded packets.
(c) Selective forwarding by forwarding only those relevant packets which approaches to the right destination.

To avoid looping, a sequence number may be added to each packet's header. This helps in discarding those packets whose sequence number is lower than the one already received.

Note: Although flooding is inefficient in most applications, an exception may be in the case of military application where large number of routers are placed and robustness is highly desirable.

2. **Flow-based routing:** This is a non-adaptive algorithm which uses topology and traffic condition for deciding the route. If traffic on some route is more than average, then the route should be avoided to achieve optimal path.

   If line capacity and flow is given, delay can be determined easily using the following formula:

$$T = \frac{1}{\mu C - \lambda}$$

   where $1/\mu$ is mean packet size in bits, λ is the mean number of packets arrived per second and C is the line capacity in Kbps.

Example 10.2

Distance Vector/Distributed Bellman–Ford/ Ford–Fulk-erson Routing Algorithm: This is one of the popular examples of dynamic routing algorithm. Each router maintains a table (called vector) which helps in finding the best-known distance to any destination. It also indicates preferred outgoing line to be used to reach the destination. The performance metric can be the number of hops, time delay or number of packets in the queue.

Issues:

(a) Slowness in converging to the right answer and this problem is well known as count to infinity (can be solved using split-horizon algorithm).
(b) Line bandwidth should be a metric when choosing root.

Whenever a packet comes to a router, the neighbouring router will give their routing table and a new vector table is created at that router.

Example 10.3

Consider a network shown in Fig. 10.4:

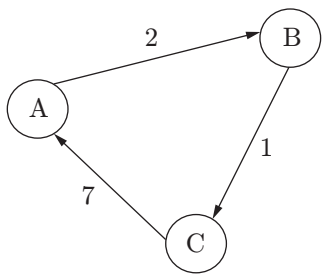


Figure 10.4

Step 1: Initialise cost of direct links and set to ∞cost from neighbours.

Node A table

		Cost to		
		A		C
From	A	0	2	7
	B	∞	∞	∞
	C	∞	∞	∞

Node B table

		Cost to		
		A	B	C
From	A	∞	∞	∞
	B	2	0	1
	C	∞	∞	∞

Node C table

		Cost to		
		A	B	C
From	A	∞	∞	∞
	B	∞	∞	∞
	C	7	1	0

Step 2: Each node periodically sends its own distance vector (DV) to neighbours. When node A receives DV from neighbour B, it keeps it and updates its own DV as follows:

$$D_A(\mathrm{B}) = \min_v \{ C(x, v) + D_v(\mathrm{B})\}$$

Node A updated table after receiving DV from node B and node C is:

$$D_A(\mathrm{B}) = \min\ \{2 + 0, 7 + 1\} = 2 \text{ and}$$
$$D_A(\mathrm{C}) = \min\ \{2 + 1, 7 + 0\} = 3$$

Node A table

		Cost to		
		A	B	C
From	A	0	2	3
	B	2	0	1
	C	7	1	0

Node B table

		Cost to		
		A	B	C
From	A	0	2	7
	B	2	0	1
	C	7	1	0

Node C table

		Cost to		
		A	B	C
From	A	0	2	7
	B	3	0	1
	C	3	1	0

Step 3: In similar fashion, algorithm proceeds until all nodes have updated tables.

Node A table

		Cost to		
		A	B	C
From	A	0	2	3
	B	2	0	1
	C	3	1	0

Node B table

		Cost to		
		A	B	C
From	A	0	2	3
	B	2	0	1
	C	3	1	0

Node C table

		Cost to		
		A	B	C
From	A	0	2	3
	B	2	0	1
	C	3	1	0

3. **Link state routing:** This is simply a modern replacement of distance vector routing. The steps of the algorithm are as follows:
   - Each router discovers the neighbours for their network addresses.
   - Measure delay or cost to each of these neighbours.
   - Construct a packet including network address and delays of all neighbours.
   - Send it to all routers.
   - Find the shortest path to all routers (Dijkstra's algorithm can be used).

Example 10.4

OSPF protocol (used in Internet) uses the link state algorithm: IS-IS (intermediate system–intermediate system) is another example used in Internet backbones and in some digital cellular systems.

4. **Hierarchical routing:** In the situation of telephone networks, all above routing algorithms fail because the size of the routing table is too large here. In this routing, routers are divided (placed) into different regions. A router will have the knowledge of other routers in its own region but unaware about the internal structure of other regions. This will reduce the size of the routing table.
5. **Broadcast and multicasting routing:** Broadcasting means sending packets to all other hosts in the network, whereas multicasting refers to sending packets to a specific group or a fixed number of hosts.

10.6 NETWORK LAYER PROTOCOLS

In the Internet model, or the TCP/IP suite, there are five main network layer protocols: ARP, RARP, IP, ICMP and IGMP (Fig. 10.5).

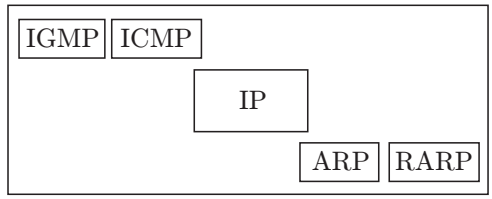


Figure 10.5 Network protocols.

The main protocol in this layer is IP. It is responsible for host to host delivery of packets from a source to destination. IP needs services of other protocols for better network performance. IP needs ARP to find MAC address of the next hop. As IP is an unreliable protocol, it needs ICMP (Internet Control Message Protocol) to handle unusual situations and errors. IGMP (Internet Group Message Protocol) is used for multicast delivery.

10.6.1 Internet Protocol (IPv4)

Internet Protocol is a layer-3 protocol of network layer of OSI model. It takes data segments from layer-4 transport layer and divides it into packets. Thus, it encapsulates data units received from the above layer and adds its own header information (Fig. 10.6).

Maximum size of IP header = 60 bytes

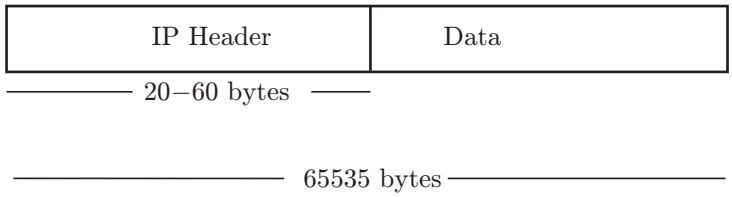
Minimum size of IP header = 20 bytes

If the size of header is 32 bytes in IP, calculate the number of option bytes.

Option bytes = 32 − 20 = 12 bytes

IP header details are as follows:

1. **Version:** Version number of Internet Protocol is 4 (e.g. IPv4).
2. **IHL:** Internet header length indicates the size of header available in the packet.
3. **TOS:** Type of service that is provided by the router to the packets such as minimum delay or cost.
4. **Total length:** Length of the entire IP packet (including IP header and IP payload).
5. **Identification:** If IP packet size is greater than the maximum transmission unit (MTU), it has to be fragmented during the transmission by the router, then all the fragments of the packet contain same identification number to identify original IP packet they belong to.
6. **Flags:** If IP packet is too large, these 'flags' tell if they can be fragmented or not. In 3-bit flag, the MSB is



Version (4 bits)	IHL (4 bits)	TOS (8 bits)	Total length (16 bits)	
Identification (Fragment ID) (16 bits)			Flags (3 bits)	Fragmentation offset (13 bits)
Time to live (8 bits)		Protocol (8 bits)	Header checksum (16 bits)	
32-Bit source address				
32-Bit destination address				
Options (if any, variable length, padded with 0's, 40 bytes maximum length)				

Figure 10.6 IPv4 packet header.

always set to '0'. The next bit is DF (do not fragment). If DF = 0, fragmentation can be done if required and buffered at the router of the receiver until all fragment comes, and if DF = 1, fragmentation should not be done. The third bit is MF (more fragment). If MF = 1, then datagram is not the last fragment. If MF = 0, then datagram is the last fragment.

7. **Fragment offset:** This offset tells the exact position of the fragment in the original IP packet.
8. **Time to live:** It controls the maximum number of routers visited by the datagram. To avoid looping in the network, every datagram is sent with some-time-to-live(TTL) value set. Each router receiving the datagram decreases TTL by 1 and when it becomes 0, the datagram is discarded.
9. **Protocol:** It tells the higher-level protocol which uses the service of the IP layer. For example, protocol number for ICMP is 1, IGMP is 2, TCP is 6 and UDP is 17.
10. **Header checksum:** This field stores checksum value of the entire header excluding data which is used to check if the packet has been received error-free.
11. **Source address:** 32-Bit address of the sender of the packet.
12. **Destination address:** 32-Bit address of the receiver of the packet.
13. **Options:** These options may contain values for various options such as strict source routing, security, record route, time stamp, etc.

Problem 10.4: If the total length bits are 0000000000 111111 and header length is 1001, calculate the size of the packet, header and payload.

Solution:

(a) Packet size = Decimal equivalent of 00000000 00111111 = 63 bytes
(b) Header size (1001) = $9 \times 4 = 36$ bytes
(c) Packet size = Header + Payload

Payload = Packet size − Header = 63 − 36
= 27 bytes

Problem 10.5: Suppose a router receives an IP packet containing 600 data bytes and has to follow a packet maximum transferable unit of 200 bytes. Assume that the IP header is 20 bytes long; specify the relevant values in each fragment header.

Solution: IP packet = 600 bytes
MTU = 200 bytes

IP header = 20 bytes

Maximum possible data length per fragment = 200 − 20 = 180 bytes

Data length of each fragment must be a multiple of 8 bytes so 22 × 8 = 176 bytes

Data packet must be divided into the following four frames:

(176 + 20) + (176 + 20) + (176 + 20) + (72 + 20) = 680

	Length	ID	MF	Fragment Offset
Original Packet	620 bytes	X	0	0
Fragment 1	196 bytes	Z	1	0
Fragment 2	196 bytes	Z	1	22
Fragment 3	196 bytes	Z	1	44
Fragment 4	92 bytes	Z	0	66

10.6.1.1 IPv4 Addresses

An **IPv4** address is a 32-bit address that can find the device on the Internet uniquely and universally. These addresses are unique, that is, these define only one connection by the device to the Internet. The total number of addresses used by this protocol which is called as address space is 2^n where n is the total number of bits. As IPv4 uses 32-bit addresses, the address space of this protocol is 2^{32}.

In IPv4, addresses are 32-bit binary numbers. However, for ease of use of people, theses binary patters are represented as dotted decimals. Therefore, there are two notations available to denote IPv4 addresses: binary and dotted decimal.

1. **Binary notation:** In this notation, IPv4 addresses are represented as 32 bits where each octet is said to be a byte. Therefore, the IPv4 address is usually said to be the 4-byte address. The example for this notation is as follows:

   01111101 10000011 00000110 00000001

2. **Dotted-decimal notation:** In this notation, each byte (8 bits) of 32-bit binary address, known as octet is separated with a dot, and then the binary number is converted into its decimal equivalent. The example for this notation is as follows:

   11000000 10101000 00001010 00001010
   is equivalent to 192.168.10.10

Classful Addressing

The architecture used by IPv4 is classful addressing where the addresses are divided into five classes: A, B, C, D and E. The first few bits reveal the class of address when the address is in binary and first byte reveals the class of address when the address is in dotted decimal notation. This architecture is called classful addressing (Table 10.4).

Table 10.4 | Classful addressing architecture of IPv4

	Leading Bits	Value Range	Starting Address	Ending Address
Class A	0	0–126	0.0.0.0	127.255.255.255
Class B	10	128–191	128.0.0.0	191.255.255.255
Class C	110	192–223	192.0.0.0	223.255.255.255
Class D	1110	224–239	224.0.0.0	239.255.255.255
Class E	1111	240+	240.0.0.0	255.255.255.255

Note: 127 reserved for loopback address.

The addresses of class A, B, C are unicast, class D addresses are multicast and class E addresses are reserved. The IP address in class A, B and C is divided into **netid** and **hostid**. In class A, one byte defines the netid and the other three bytes define the hostid. In class B, two bytes define the netid, while the other two bytes define the hostid. In class C, three bytes define the netid and one byte defines the hostid.

Mask: The length of the netid and hostid (in bits) is predetermined in classful addressing but we can also use a mask (also called the default mask) which is a 32-bit number made of contiguous 1's. The subnet mask is compared to the IP address from left to right, bit for bit. The 1's in the subnet mask represent the network portion and 0's represent the host portion. The subnet mask is created by placing a binary 1 in each bit position that represents the network portion and placing a binary 0 in each bit position that represents the host portion.

The subnet mask assigned along with IP address signifies which part of the IP address is network and which part is host.

There is a flaw in this type of architecture, that is, each class is divided into fixed number of blocks, and a lot of addresses are wasted as blocks A and B addresses are too large to consume, block C addresses are small in number, class D addresses are reserved for multicasting and class E addresses are reserved for future, which is another wastage of addresses. Therefore, these address scheme architecture is almost obsolete and leads to an introduction of classless addressing scheme as if there are no address classes.

Classless Addressing

In classless addressing, the addresses are granted in a block and the number of addresses in the block depends

upon the addresses needed by the entity. The addresses in the block must be contiguous, the first address must be evenly divisible by the total number of addresses and the number of addresses in a block must be power of 2. Mask in classless addressing can take the value in the range of 0 to 32. Classless Inter-Domain Routing provides the flexibility of borrowing bits of host part of the IP address and using them as smaller sub-networks called subnet. This process is known as **subnetting**. The addresses can be defined in IPv4 as a.b.c.d/n, where a.b.c.d defines one address of the block.

1. The first address in the block can be found by setting the rightmost 32 $-n$ bits to Os in the binary notation.
2. The last address in the block can be found by setting the 32 $-n$ rightmost bits in the binary notation of the address to Is.
3. The number of addresses in the block is the difference between the first and the last address,that is, 2^{32-n}.

Problem 10.6: Determine the subnet identifier, broadcast address and number of valid host addresses having a host with IP address of 196.142.4.29/24.

Solution:

Subnet identifier: 196.142.4.0

Broadcast address: 196.142.4.255

Number of valid host addresses: 254

The subnet mask of /24 in CIDR notation corresponds to 255.255.255.0. The above subnet mask has last eight bits set to zero (32 − 24 = 8), which means that we have 2^8 IP addresses available. Total number of valid hosts in a network is obtained by subtracting two addresses: subnet address and broadcast address $2^8 - 2 = 256 - 2 = 254$.

Problem 10.7: If the IP address of a system is 131.121.61.189, calculate the netid, first host, last host and directed broadcast address.

Solution:

IP address (class B)	131.121.61.189
Net mask	255.255.0.0
Net id	131.121.0.0
First host	131.121.0.1
Directed broadcast address (All host bits 1)	131.121.255.255

Problem 10.8: In class B, if subnet mask is 255.255.240.0, find the number of subnets and host in each subnet.

Solution:

11111111	11111111	1111	0000000
		0000	
Netid		Subnet bits	Host bits

Number of subnets = $2^4 - 2 = 14$ subnets

Number of hosts in each subnet = $2^{12} - 2$

= 4094 subnets

Problem 10.9: If IP address of a system is 199.11.171.189 and subnet mask is 255.255.255.224, calculate the subnet id.

Solution:

IP address	199.11.171.189	
Subnet mask	255.255.255.224	
Subnet id	199.11.171.160	
160 is equivalent to	101	00000
	Subnet Bits	Host Bits

Number of subnets = $2^3 - 2 = 6$

Number of hosts = $2^5 - 2 = 30$

First subnet id is 199.11.171.32 (001 00000).

Second subnet id is 199.11.171.64 (010 00000).

Last subnet id is 199.11.171.192 (110 00000).

First host of first subnet is 199.11.171.33 (001 00001).

Last host of last subnet is 199.11.171.222 (110 11110).

Supernetting is a part of classless addressing. In classless addressing, the addresses should be contiguous in a block. The first address should be exactly divisible by the number of addresses in a block. In supernet, bits are borrowed from netid.

Rules of supernetting:

1. The number of blocks must be power of 2.
2. The blocks must be continuous in the address space.
3. The third octet of the first address in the superblock must be exactly divisible by the number of blocks.

Problem 10.10: A company needs 1000 addresses, which of the following set of class C block can be used to form a supernet?

(a) 198.47.32.0, 198.47.33.0, 198.47.34.0
(b) 198.47.31.0, 198.47.32.0, 198.47.33.0, 198.47.34.0
(c) 198.47.32.0, 198.47.42.0, 198.47.52.0, 198.47.62.0
(d) 198.47.32.0, 198.47.33.0, 198.47.34.0, 198.47.35.0

Solution: (d)

(a) Not in power of 2 or blocks are 3.
(b) Third octet of first address is not divisible by 4.
(c) Blocks are not contiguous.

Problem 10.11: Which of the following can be the beginning address of a block that contains 16 addresses?

(a) 205.16.37.32
(b) 190.16.42.44
(c) 17.17.33.82
(d) 123.45.24.52

Solution:

(a) 32 is divisible by 16.

10.6.2 ICMP

As IP is a connectionless and unreliable protocol, it cannot report errors, so it takes help of ICMP to communicate updates or error information to other intermediate routers, devices or hosts.

Each ICMP message contains three fields: Type, Code and Checksum (Fig. 10.7). The Type field identifies the ICMP message, the Code field provides further information about the associated Type field and the Checksum field verifies the integrity of the message.

8 bits	8 bits	16 bits
Type	Code	Checksum
Rest of header		
Data section		

Figure 10.7 | ICMP message.

Extra options are specified in the rest of the header. ICMP places the message to be sent in the data section. Different types defined for ICMP messages are shown in Table 10.5.

Table 10.5 | Different types defined for ICMP messages

Category	Type	Message
Error Reporting Message	3	Destination Unreachable
	4	Source Quench
	11	Time Exceeded
	12	Parameter Problem
	5	Redirect Message
Query Message	0	Echo Reply
	8	Echo Request
	9	Router solicitation
	10	Router advertisement
	13	Timestamp Request
	14	Timestamp Reply
	17	Address Mask Request
	18	Address Mask Reply

10.7 LAYER 4: TRANSPORT LAYER

Although the reliability of the network layer is undoubtful, the transport layer has many responsibilities to carry out the following:

1. Managing connections and timers
2. Allowing reliable, connection-oriented byte stream from one end to other end
3. Multiplexing
4. Addressing
5. Performing segmentation
6. Packetizing
7. Handling error control and variable sized sliding window for flow control
8. Allocating bandwidth with congestion control

Issues:

(a) Headers, error detection, reliable communication
(b) Communication between processes (running on machines on possibly different networks)

10.8 CONGESTION

Congestion occurs when packets overload the subnet (means the number of packets sent to the network is greater than the capacity of the network), which results in performance degrade. The following are causes for congestion:

1. A sudden stream of packet from various sources reaching the same destination.
2. Slow links.
3. Slow processor.
4. Suppose an intermediate router has no free buffer, it will discard the packet. As the ender will not receive any acknowledgement, it will resend the packet and hence congestion will be there.

Congestion is unavoidable, but it is necessary to control it. It can be handled with the following approach:

1. Congestion information may be forwarded to the concerned node so that it is possible to limit senders (prevent one sender from overflowing the receiver) or reroute the packets.
2. Resource availability may reduce the congestion.
3. Prevent additional packets from entering the congestion region.

To control congestion in network traffic, leaky bucket algorithm is used. This algorithm shapes the bursty traffic into fixed rate traffic. It does so by averaging the data rate and dropping the packets if bucket is full.

10.8.1 Congestion Versus Flow Control

Congestion control ensures the ability to carry the offered traffic by any subnet. The major entities that effect the congestion are behaviour of hosts, routers as well as the factors that reduce the carrying capacity.

Flow control makes it sure that there is not much difference between the sending and receiving packet rate, that is, a fast sender does not send at a rate faster than the rate at which the receiver receives.

10.9 USER DATAGRAM PROTOCOL AND TRANSMISSION CONTROL PROTOCOL

10.9.1 User Datagram Protocol

UDP, or User Datagram Protocol, is an unreliable and connectionless protocol used by applications that transmit small amount of data at one time and do not require receipt of acknowledgement of data, for example, audio or video broadcasting (Fig. 10.8).

Source port (16 bits)	Destination port (16 bits)
Length (16 bits)	Checksum (16 bits)
Data	

Figure 10.8 UDP packet.

10.9.2 Transmission Control Protocol

TCP, or Transmission Control Protocol, provides a connection-oriented, reliable stream delivery through the use of sequenced acknowledgement with retransmission of packets when necessary. The TCP header structure is shown in Fig. 10.9.

Source port (16 bits)								Destination port (16 bits)
Sequence number(32 bits)								
Acknowledgement number (32 bits)								
HLEN (4 bits)	Reserved (6 bits)	U	A	P	R	S	F	Window (16 bits)
Checksum (16 bits)								Urgent pointer (16 bits)
Option + padding								
Data								
TCP - header structure								

Figure 10.9 TCP packet.

In TCP, sequence number is attached for every byte in the segment. The initial sequence number for the first data byte will be a random number that is generated by a random number generator in the range of 0 to $2^{32} - 1$. Acknowledgement number will always be the sequence number of the next expected data. The control bits and their description is given in Table 10.6.

Table 10.6 | Control bits

Control Bits	Description
U (URG)	Urgent pointer field significant
A (ACK)	Acknowledgement field significant
P (PSH)	Push function
R (RST)	Reset the connection
S (SYN)	Synchronize sequence numbers
F (FIN)	No more data from sender

10.10 SOCKETS

A **socket** is an endpoint for communication between client process and server process across a network. A **socket address** is the combination of an IP address and a port number. This address is used to send data packet to a particular process running on a machine. When two programs are executed, a client process and a server process are created, and these processes communicate with each other by reading from, and writing to, sockets.

There are few system calls, such as those given below:

1. Socket() returns a socket descriptor (like file descriptor), which is an integer value.
2. Bind() binds an address to a socket descriptor created by socket.
3. Listen() announces willingness to accept connections.
4. Accept() blocks the caller until a connection attempt arrives.
5. Connect() actively attempts to establish a connection.
6. Send() sends some data over the connection.
7. Receive() receives some data from the connection.
8. Close() releases the connection.

The following steps (as shown in Figure 10.10) occur when establishing a TCP connection between two computers using sockets:

1. The server instantiates a ServerSocket object, which denotes the port number on which the communication is to take place.
2. The server invokes accept() method of the ServerSocket class, which waits until a client connects to the server at the given port.
3. While the server is waiting, a client instantiates a Socket object, which specifies the server name and port number for the connection.
4. The constructor of the Socket class attempts to connect client to the specified server and port number. Once the communication is established, the client has a Socket object for communicating with the server.
5. The accept() method returns a reference to a new socket on the server that is connected to the socket of the client.

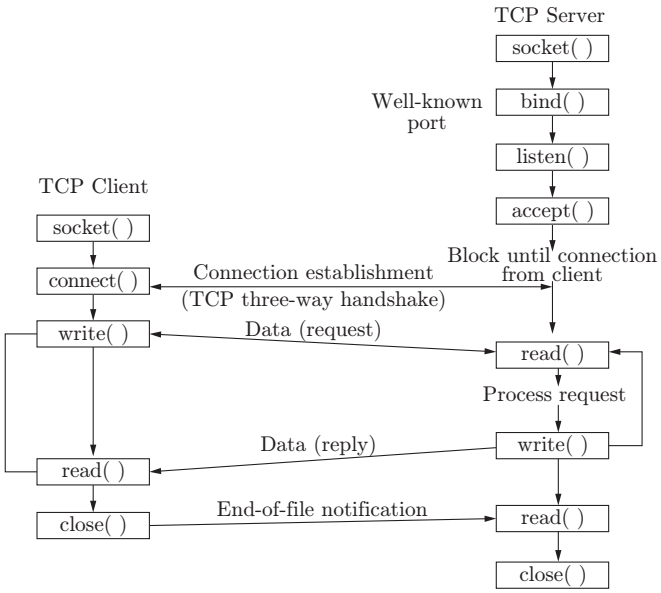


Figure 10.10 | Socket creation.

10.11 LAYER 5: SESSION LAYER

The services provided by session layer are as follows:

1. Establishes, manages and terminates a communication session with remote systems
2. Allows two machines to enter into a dialog (communication may be half duplex or full duplex)
3. Adds checkpoints or synchronisation points to a data stream.
4. Groups several user-level connections into a single 'session'.

Some protocol suites do not include the session layer.

Checkpoint: If we need a file of 1000 pages, it is suggested to insert checkpoints after every 100 pages to ensure that each 100-page unit is received and acknowledged independently. Meanwhile, if a crash occurs during the transmission of page 324, the only pages that need to be resent after system recovery are pages 301 to 324. The pages from 1 to 300 need not be resent.

10.12 LAYER 6: PRESENTATION LAYER

The following are the major concerns of presentation layer:

1. Syntax and semantics of the information exchanged between two systems.
2. Support to different encoding schemes used by different machines. It converts the sender-dependent format into a common format and the receiver converts it back to the receiver-dependent format.
3. Data encryption—to ensure privacy, sensitive information should be encrypted. Information encrypted to some other form is unreadable to others.
4. Data compression—to reduce the size of information to carry. In the case of multimedia, for example, text, audio and video, compression is a very useful tool.

Note: Many protocol suites do not include a presentation layer.

10.13 LAYER 7: APPLICATION LAYER

The major responsibility of application layer is to implement communication between two applications of the same type. There is a common misconception that every user application runs on application layer, but it runs only on those applications which interact with the communication system. For example, FTP, HTTP, SMTP/POP3/IMAP (email)are all application layer protocols, but a designing software or text editor cannot be considered as an application layer protocol.The following are popular application layer protocols:

1. **Internet Control Message Protocol (ICMP):** ICMP provides error reporting and query management mechanism for host, which lacks in IP. ICMP messages are of two types: error-reporting messages and query messages. The header of ICMP is of 8 bytes and data section is of variable size. First byte is ICMP type, second byte is code, the next two bytes specify the checksum field and rest of the header is specific for each message type.
2. **Domain Name System (DNS):** DNS is a supporting program, which is used by other programs used by the users, such as email. DNS is a client–server program used to find the IP address of an email recipient. In DNS client–server application, a sender sends an email to the email address of the receiver. The DNS client sends the request to the DNS server to map the email address to IP address. DNS has three different domains: generic, country and inverse domains. Generic domain specifies the registered hosts according to their generic behaviour. For example, com, org, edu, gov, net, etc. The country domain uses two character country abbreviations, for example, 'in' for India. The inverse domain is used to map an address to a name (Fig. 10.11).

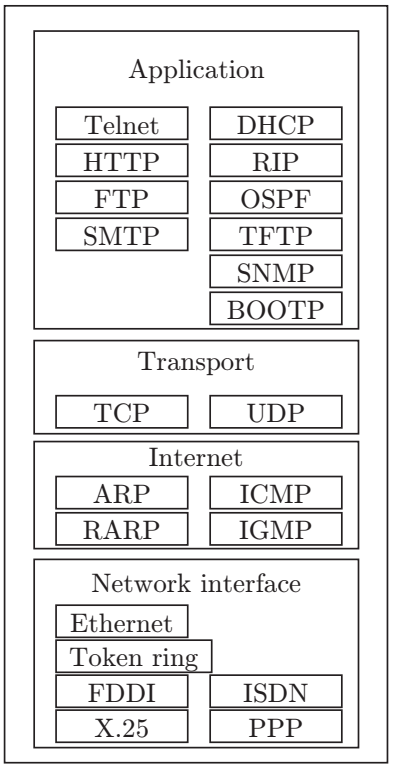


Figure 10.11 | TCP/IP protocols.

3. **Simple Mail Transfer Protocol (SMTP):** SMTP is a message transfer agent (MTA), which is used to transfer mail. A system should have client MTA for sending a mail and server MTA for receiving a mail. SMTP is used two times, between the sender and the sender's mail server and between the two mail servers. The main task of SMTP is to push the message from the client to the server.
4. **Post Office Protocol (POP):** POP3 (version 3) is a message access protocol which is used to extract the message for client to the server. It has two modes: the delete mode and the keep mode. In the delete mode, the mail is deleted from the mailbox after each retrieval, while in the keep mode, the mail remains in the mailbox after retrieval.
5. **File Transfer Protocol (FTP):** It is used for transferring files from one system to another. FTP establishes two connections between hosts, one for data transfer and the other for control information. FTP uses TCP Port 21 for the control connection

and TCP Port 20 for the data connection. The FTP client has three components: user interface, client control process and the client data transfer process. The server has two components: the server control process and the server data transfer process. The control connection remains open for entire FTP session, whereasthe data connection remains open for each file transfer.

6. **Hypertext Transfer Protocol (HTTP):** HTTP is used to access data on the World Wide Web (WWW). It works as a combination of FTP and SMTP. Unlike FTP, HTTP does not have any control connection and uses only one TCP connection. Unlike SMTP, it does not store and then forward the messages. It immediately sends the messages. HTTP uses a TCP Port 80.

10.14 DEVICES

The devices used for internetworking at different layers are specified in Fig. 10.12.

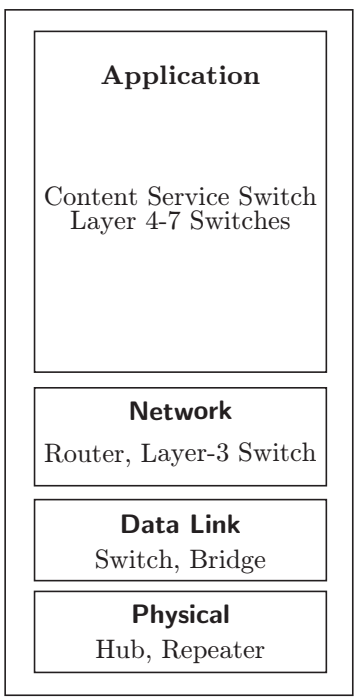


Figure 10.12 | Devices.

1. **Repeaters:** It is an electronic device which can receive the weak signal and retransmit it with higher speed. For example, if the LAN is connected for a long distance, then to cover a distance the signal is sent to a repeater which is attached with two LANs and retransmits the signal to a higher level. Thus, repeater connects two segments of network cable. It works at the physical layer of the OSI model.
2. **Bridge:** There is a limited number of stations that can be connected with a single LAN. So, a bridge is used to connect multiple LANs of the same type. It operates on a physical layer and data link layer. A bridge checks the physical address contained in the frame. It also uses table for filtering frames. It partitions the collision so that performance increases.
3. **Hub:** It is a network device which is used to connect various computers together. Hub is the central connection for all the computers, which connect through Ethernet. Hub can receive and send the information but cannot perform both tasks at the same time. This makes it slower than a switch. It is less expensive and less complex.
4. **Switch:** Switch is a small network device used to connect one or more computers through LAN. It is mostly used in home networks. Switches and hubs are used in the same network. Hubs increase the network by providing more ports, and switches divide the whole network into smaller networks. Switching reduces the amount of unnecessary traffic when every port sends the same information. Switch also reduces the possibility of collision in network.
5. **Router:** A router is a networking device which takes packets from one network and after analysis sends that packet to another network. When a data packet comes to the router, the router reads the destination address of the packet and sends it to the respective router which contains that destination address (listed in its routing table). A router is more intelligent than a hub because a hub only sends the information between the devices but the router analyses the packet and then forwards it to the other network. It controls the traffic on the network.
6. **Gateway:** It is a networking system capable of interconnecting one or more networks that has different base protocol. Gateway serves as an entry and exit point. Gateway, sometime called as protocol converter, is used in different layers. For example, a gateway can be used to convert a TCP/IP packet to a NetWare IPX packet.

Problem 10.12: If the capacity of router is 1 MB, data output rate is 8 Mbps. Tokens are generated at 6 Mbps. Calculate the time burst traffic is routed.

Solution: $C + \rho \cdot S = M \cdot S$

where M = output rate, C = capacity of router, S = time of burst traffic and ρ = token rate

$$10^6 + 6 \times 10^6 \cdot S = (8 \times 10^6) \cdot S$$
$$10^6(1 + 6S) = (8 \times 10^6) \cdot S$$
$$1 + 6S = 8S$$
$$2S = 1 \Rightarrow S = 0.5 \text{ s}$$

10.15 NETWORK SECURITY

Network security is an activity designed to protect the network and data in terms of usability, reliability, integrity and safety. It targets a variety of threats and prevents them to enter into the network. It is accomplished through hardware and software, for example, firewall, anti-virus and anti-spyware, cryptography, intrusion prevention systems (IPS) used to identify fast-spreading threats such as zero-day attacks,and virtualprivate networks (VPNs) used to provide secure remote access.

Network security components are as follows:

1. **Confidentiality:** It ensures the concealment of data to unauthorised individuals.
2. **Integrity:** It ensures that information is changed in a specified and authorised manner. There is no change in content or source by an unintended user.
3. **Availability:** It ensures that systems are available for the authorised users.

The trio form the term CIA (Confidentiality, Integrity and Availability).

10.15.1 Basic Concepts in Cryptography

Cryptography is the science of providing secure communication over insecure channels. Cryptography consists of two operations: encryption and decryption.

Encryption is the process in which data is ciphering so that only the intended recipient can know the message. Decryption is the process of deciphering the message.

Basically, encryption and decryption are two functions of a cryptographic algorithm mathematically related to each other. A cryptographic algorithm is widely known, but a key, which is used for the encryption/decryption, is kept secret.

Cryptography is of two types: symmetric cryptography and asymmetric cryptography. Both the approaches have their own pros and cons.

10.15.1.1 Symmetric Cryptography

In symmetric key cryptography, a single shared key is used for both encryption and decryption as shown in Fig. 10.13. Symmetric cryptographic primitives use block ciphers, stream ciphers, cryptographic hash functions, and message authentication codes (MACs). Block cipher uses a deterministic algorithm and operates on a block (fixed length of bits) with unaltered transformation. Stream cipher encrypts each bit individually to generate cipher text. Hash functions or one-way hash functions are used to map an arbitrary-length message string to fixed-size message string. The final value is called hash value.

Figure 10.13 Symmetric key cryptography.

The security of the symmetric key cryptography lies in the secrecy of the shared symmetric key. If the adversary captures the shared secret key, then it affects both confidentiality and authentication of the message. Examples of symmetric key cryptography are Twofish, Serpent, AES (Rijndael), Blowfish, RC4, RC5, 3DES, IDEA, SEAL, SNOW, etc.

10.15.1.2 Asymmetric Cryptography

In asymmetric cryptography, a private key is used for the decryption of a message while a public key is used for the encryption of the message as shown in Fig. 10.14. The private key needs to be kept confidential while the public key can be published freely. Asymmetric cryptography is also known as public key cryptography (PKC). PKC was introduced first by Diffie and Hellman in 1976. Public key algorithms are based on mathematical functions rather than substitution and transposition as in symmetric key cryptography. Examples of asymmetric key cryptography are Diffie-Hellman, RSA, Merkle-Hellman, Rabin, McEliece, El Gamal, Ellliptic curves, etc.

Figure 10.14 Asymmetric key cryptography.

RSA Algorithm

The RSA algorithm, developed in 1977 by Riverst, Shamir, Adelman at MIT, provides encryption and digital signatures. RSA is based on factoring of large numbers, which is not known to be NP-complete.

Encryption and decryption are as follows for a plaintext block M and cipher textblock:

$$C = M^e \bmod n$$
$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

where n and e are known to both the sender and receiver, but d is only known to the receiver. Thus, the public key is $P = \{n, e\}$ and the private key $S = \{d, n\}$. It is impossible to find d given e and n.

The detailed RSA algorithm is as follows:

1. **Key generation:**
   - Choose two prime numbers p and q, keep them secret.
   - Compute $n = pq$, n is public.
   - Calculate $\phi(n) = (p-1)(q-1)$
   - Choose e with $gcd(\phi(n), e) = 1; 1 < e < \phi(n)$, which is also public.
   - Compute $d = e^{-1} \bmod \phi(n)$ and keep it private.
   - The private key consists of $S = \{d, n\}$ and public key consists of $P = \{n, e\}$.
2. **Encryption:**
   Plaintext $M < n$
   Cipher text $C = M^e \bmod n$
3. **Decryption:**
   Ciphertext C
   Plaintext $M = C^d \bmod n$

10.15.2 Digital Signature

It provides the authenticity of the origin of information to the user and verifies the information is intact. Hence, it provides authentication and data integrity. It also provides non-repudiation, which ensures that the sender cannot deny the origin of information. It is based on the public key cryptography concept.

10.15.2.1 Digital Standard Algorithm

Digital Standard Algorithm (DSA) is based on the difficulty of discrete logarithm problem (DLP). It is also based on Elgamal and Schnorr system.

DSA involves the following four steps:

1. **Key generation:**
   - *Global Public Components:* p is a prime number with 512-1024 bits, q is a prime divisor of $(p-1)$ with 160 bits, g is an integer $g = h^{(q-1)/q} \bmod p$.
   - *Users Private Key:* x is random integer less than q.
   - *Users Public Key:* $y = g^x \bmod p$
2. **Signature:**
   - For each message M, generates random k
   - Computes $r = (g^k \bmod p) \bmod q$
   - Computes $s = k^{-1}(H(M) + xr) \bmod q$
   - Signature is (r, s)
3. **Verification:**
   - Computes $w = s^{-1} \bmod q$, $u_1 = H(M)w \bmod q$
   - Computes $u_2 = rw \bmod q$, $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$
   - Verify if $v = r$
4. **Correctness:**

$$\begin{aligned} v &= (g^{u_1} y^{u_2} \bmod p) \bmod q \\ &= (g^{H(M)w \bmod q} y^{rw \bmod q} \bmod p) \bmod q \\ &= (g^{H(M)w \bmod q} y^{xrw \bmod q} \bmod p) \bmod q \\ &= (g^{(H(M)w + xrw) \bmod q} \bmod p) \bmod q \\ &= (g^{(H(M)+xr)w \bmod q} \bmod p) \bmod q \\ &= (g^{(H(M)+xr)k(H(M)+xr)^{-1} \bmod q} \bmod p) \bmod q \\ &= (g^k \bmod p) \bmod q \\ &= r \end{aligned}$$

10.15.3 Firewall

It is a device that filters access to the protected network from the outsider network. It is an integrated collection of security measures, which are designed to prevent unauthorised electronic access to a network system. It has a predefined set of rules, which can protect private network from unauthorised access by filtering incoming or outgoing traffic. These predefined set of rules are called firewall policies.

10.15.3.1 Functions of Firewalls

1. Examining the packet header and filtering
2. Verifying the IP address or the port
3. Granting and denying access

Packets can be filtered on the basis of the following criteria:

1. Source IP address
2. Destination IP address
3. TCP/UDP source port
4. TCP/UDP destination port

10.15.3.2 Types of Firewalls

1. **Packet Filter (stateless):** It is router-based filters. It does not use any context for filtering the packets. Individual packets are accepted or rejected. It has following limitations: (i) filter rules are hard to set up, (ii) inadequate primitives, (iii) hard to manage access to RPC-based services.
2. **Stateful Filter:** It maintains records of all connections passing through it. It determines if a packet is either the start of a new connection, a part of an existing connection or is an invalid packet. It maintains tables of each active connection, including the IP addresses, ports and sequence numbers of packets.
3. **Application gateway:** It works as a proxy. It is made up of bastion hosts, which run special software to act as a proxy server. It inspects the contents of the traffic, blocking inappropriate contents, such as websites, viruses, vulnerabilities, etc.

10.15.3.3 Limitations of Firewall

1. It cannot protect against attacks that bypass the firewall.
2. It cannot protect fully against internal threats.
3. It cannot provide protection against malicious code problems such as viruses and Trojan horses, although some are capable of scanning the code.

IMPORTANT FORMULAS

Application	Protocol
SMTP	TCP
TELNET	TCP
SSH	TCP
HTTP	TCP
DNS	TCP & UDP
PING	ICMP

1. In early token release,

   Throughput for single station or N stations

$$= \frac{\text{Data}}{\text{Transmission time} + (\text{Ring latency}/\text{Number of stations})}$$

2. In delayed token release,

   Throughput for single station

$$= \frac{\text{Data}}{\text{Transmission time} + \text{Ring latency} + (\text{Ring latency}/\text{Number of stations})}$$

   Throughput for N stations

$$= \frac{\text{Data}}{\text{Ring latency} + (\text{Ring latency}/\text{Number of stations})}$$

3. $\text{Transmission time} = \dfrac{\text{Message size (bits)}}{\text{Bandwidth (bits/s)}}$

4. $\text{Propagation time} = \dfrac{\text{Distance}}{\text{Velocity}}$

5. For stop-and-wait ARQ

$$\text{Transmission time} = \frac{\text{Message size}}{\text{Bandwidth}}$$

$$\text{Propagation delay} = \frac{\text{Distance of the link}}{\text{Velocity}}$$

   Link utilisation of sender or throughput is given by

$$\eta = \frac{\text{Transmission time}}{\text{Transmission time} + 2 \times \text{Propagation delay}}$$

6. Pure ALOHA

   $\text{Throughput } (S) = G \times e^{-2G}$

   $\text{Vulnerable time} = 2 \times T_{fr}$

   $S_{\max} = 18.4\%$

7. Slotted ALOHA

   $\text{Throughput } (S) = G \times e^{-G}$

   $\text{Vulnerable time} = T_{fr}$

   $S_{\max} = 36.8\%$

SOLVED EXAMPLES

1. Which layer is responsible for delivery from process to process?

(a) Network (b) Transport
(c) Physical (d) Data link

Solution: Transport layer is responsible for end-to-end process communication.

Ans. (b)

2. The minimum size of an Ethernet frame is

(a) 18 bytes (b) 64 bytes
(c) 46 bytes (d) 56 bytes

Solution: Ethernet header = 18 Bytes [Dest. Mac (6) + Source Mac (6) + Length (2) + CRC (4)] Minimum Data Portion = 46 Bytes and Minimum Ethernet Frame Size = 64 Bytes

Ans. (b)

3. Using a 7-bit sequence number, what is the maximum size (in bits) of the sender and receiver window using stop-and-wait protocols?

(a) 1 and 1 (b) 1 and 7
(c) 7 and 1 (d) 7 and 7

Solution: Stop-and-wait protocol uses 1 bit sequence.

Ans. (a)

4. Vulnerable time in CSMA is

(a) It is double the transmission time as it includes the sensing time
(b) Half of average frame transmission time as collisions are less
(c) $2 \times$ (the average frame transmission time)
(d) None of the above

Solution: Vulnerable time for CSMA is the propagation time T_p needed for a signal to propagate from one end of the medium to the other.

Ans. (d)

5. Which class of IP addresses is used for multicasting?

(a) Class E (b) Class C
(c) Class A (d) Class D

Solution: Class D is used for multicasting. Refer Table 10.4.

Ans. (a)

6. The options field of IPv4 is used for?

(a) Time stamping, strict source routing
(b) Loose source routing, strict source routing
(c) Time stamping only
(d) Time stamping, loose source routing, strict source routing

Solution: It may contain values for various options, such as strict source routing, security, record route, time stamp, etc.

Ans. (d)

7. In a fully connected mesh network with d devices and c connections, there are _____ physical channels to link all devices.

(a) $d*(d-1)/2$ (b) $d*(d+1)/2*c$
(c) $(2*d + 2*c)/2$ (d) $2*(d+1)+c$

Solution: The total number of wired links required to establish a fully connected mesh network of d nodes can be calculated as $c = d\,(d-1)/2$.

Ans. (a)

8. In IPv4 header, the _____ field is used to determine to which datagram a newly arrived fragment belongs to.

(a) Identification (b) Datagram_id
(c) Fragment offset (d) Time to live

Solution: Identification field is used to identify original IP packet the fragments belong to.

Ans. (a)

9. Encryption and decryption is the responsibility of _____ layer.

(a) Session (b) Data link
(c) Application (d) Network

Solution: Application layer is responsible for the encryption and decryption processes.

Ans. (c)

10. Maximum throughput of an ALOHA network is

(a) 18.4% (b) 35.8% (c) 36.8% (d) 50%

Solution: When $G = 1$, the throughput is increased to the maximum value of 36.8%.

Ans. (c)

11. A terminal multiplexer has six 1200 bps terminals and 'N' 300 bps terminals connected to it. The outgoing line is 9600 bps. What is the maximum value of N?

(a) 4 (b) 6 (c) 8 (d) 12

Solution: Since, there are six 1200 bps terminals. So, $6 \times 1200 + n \times 300 = 9600 \Rightarrow n = 8$

Ans. (c)

12. The total number of wired links required to establish a fully connected mesh network of 9 nodes will be

(a) 36 (b) 56 (c) 72 (d) 64

Solution: Total number of wired links required to establish a fully connected mesh network of n nodes can be calculated as $c = n(n-1)/2$
So, for 9 nodes, total links are 36.

Ans. (a)

13. Considering a classful addressing, the IP address 128.252.144.84 denotes

(a) 0.0.0.0 as network ID and 128.252.252.84 as node ID
(b) 128.0.0.0 as network ID and 128.252.127.84 as node ID
(c) 128.252.0.0 as network ID and 128.252.144.84 as node ID
(d) 128.252.144.0 as network ID and 128.252.144.84 as node ID

Solution: The IP belongs to class B. The network id is 128.252.0.0 and the node id is 128.252.144.84.

Ans. (c)

14. In Ethernet CSMA/CD, the special bit sequence transmitted by media access management for collision handling is called

(a) Hamming code (b) CRC
(c) Jam (d) Preamble

Solution: Hamming code is a set of error-correction code, which is used to detect and correct bit errors. CRC (cyclic redundancy check) is used to detect data transmission errors.
Preamble is used in network communications for synchronizing transmission time between systems.

Ans. (c)

15. A Gateway operates at _____ layers.

(a) All layers except physical and application layer
(b) All the seven layers
(c) Only on session, transport and network layers
(d) Same layers on which the switch and bridge operates

Solution: Hubs, repeaters(or active hubs) operate at physical layer (1); Bridges, Switches operates at data link layer (2) ; Routers operate at network layer (3) ; content-switches (or web-switches or application-switches) operates at layers 4 to 7. Gateway operates at all the seven layers–physical, data link, network, transport (4), session (5), presentation (6), application (7).

Ans. (b)

16. Consider the network with subnet mask 153.224.0.0/13. Determine the last host address in the network.

(a) 153.208.255.255
(b) 153.224.255.254
(c) 153.231.255.255
(d) 153.231.255.254

Solution: Network id: 153.224.0.0
Subnet mask: 255.248.0.0
By doing XOR between network id and subnet mask, one gets network id= 153.231.0.0
We have 19 bits for host, so first host address is 153.224.0.1 and last host is 153.231.255.254

Ans. (d)

17. Consider a token ring LAN of length 12 km and having 40 stations, signal propagation speed is 8 ns/m and data rate is 100 Mbps. An average frame contains 220 bytes. Delay occurred at each station is equivalent to 10-bit delay. What is the utilisation of token ring approximately?

(a) 10% (b) 12%
(c) 15% (d) 21%

Solution: Transmission time of a frame

$$T_t = \frac{220 \times 8}{100 \times 10^6} \text{ s} = 17.6 \ \mu\text{s}$$

Propagation time around the ring

$$T_p = 10 \times 12000 \text{ ns} = 120 \ \mu\text{s}$$

Delay at each station = 1/10 μs, so delay at 40 stations will be = 4 μs
Now,

$$\text{Utilisation (U)} = \frac{T_t}{T_t + T_p + \text{delay at all stations}}$$
$$= \frac{17.6}{17.6 + 120 + 4} = \frac{17.6}{141.6}$$
$$= 0.124 \times 100 = 12.4\% \approx 12\%$$

Ans. (b)

18. A CSMA/CD-based network has transmission rate 120 Mbps, length 1 km and speed of signal is 10^9 m/s. What should be the minimum frame size?

(a) 120 B (b) 240 B
(c) 400 B (d) 440 B

Solution: We know that

$$\frac{\text{Length of packet}}{\text{Bandwidth}} = 2 \times \frac{\text{Distance}}{\text{Velocity}}$$

So, packet length $= 2 \times \frac{1000}{10^9} \times 120 \times 10^6 \Rightarrow 240$ bytes

Ans. (b)

19. How many 8-bit characters can be transmitted per second over a 7800 baud serial communication link using asynchronous mode of transmission with one start bit, 8 data bits and 1 parity bit?

(a) 600 (b) 660
(c) 780 (d) 1200

Solution: Total number of bits = 10

Modulation rate = 7800 baud

Numbers of 8-bit characters are transmitted = $\frac{7800}{10 \text{ bits}} = 780$

Ans. (c)

20. In a token ring network, the transmission speed is 20 bps and the propagation speed is 180 m/μs. The 1-bit delay in this network is equivalent to:

(a) 20 m of cable (b) 18 m of cable
(c) 9 m of cable (d) 5 m of cable

Solution: Given that $R = 2\times10^7$ bps, $B = 1$ bps, $V = 180$ m/μs; $d = ?$

$$B = R\times\frac{d}{v}$$

$$d = \frac{Bv}{R} = \frac{1\times180}{2\times10^7\times10^{-6}} = 9 \text{ m of cable}$$

Ans. (c)

21. Determine the maximum length of the cable (in km) for transmitting data at a rate of 400 Mbps in an Ethernet LAN with frames of size 8000 bits. Assume the signal speed in the cable to be 250000 km/s.

(a) 4 b) 5 (c) 5.5 (d) 7

Solution: Maximum length of cable can be calculated by the given problem:

$$\frac{8000 \text{ bits}}{400\times10^6 \text{ bits/s}} = \frac{2\times L}{25\times10^4 \text{km/s}}$$

$$L = 5 \text{ km}$$

Ans. (b)

22. In the IPv4 addressing format, the number of networks allowed in class C addresses is

(a) 2^{21} (b) 2^{22}
(c) 2^{23} (d) 2^{24}

Solution: In class C,
Net id = 24 bit
Host id = 8 bits
Out of 24 bits 3 bits are reserved for representation class 'C', that is, 110.
So, C class has 2^{21}networks.

Ans. (a)

23. Consider a 2.5 Mbps token ring LAN and frame size of 180 bytes. If the ring latency is 210 μs, then what will be the effective data rate of the LAN?

(a) 1.40 Mbps (b) 1.56 Mbps
(c) 1.76 Mbps (d) 1.84 Mbps

Solution:

$$\text{Transmission time} = \frac{\text{Data size}}{\text{Bandwidth}} = \frac{180\times8}{2.5\times10^6} = 0.00057 \text{ s}$$

Ring latency = Round-trip time = 210 μs = 0.00021 s
Total time required to transfer 1440 bits = (0.00057 + 0.00021) = 0.00078 s
So, effective data rate = 1440/0.00078 = 1.84 Mbps

Ans. (d)

24. How many number of parity bit is required in hamming code if message size is 8 bit?

(a) 2 (b) 4
(c) 6 (d) 8

Solution:
Given message (m) = 8-bit
To calculate the number of parity bits, we have the formula:

$2^r \geq m + r + 1$, by putting $r = 1, 2, 3, 4\ldots$
$r = 4$ will satisfy the given equality.

Ans. (b)

GATE PREVIOUS YEARS' QUESTIONS

1. Which of the following assertions is FALSE about the Internet Protocol (IP)?

(a) It is possible for a computer to have multiple IP addresses.
(b) IP packets from the same source to the same destination can take different routes in the network.
(c) IP ensures that a packet is discarded if it is unable to reach its destination within a given number of hops.
(d) The packet source cannot set the route of an outgoing packet; the route is determined only by the routing tables in the routers on the way.

(GATE 2003: 1 Mark)

Solution: The packet source can set the route of an outgoing packet; the route is determined only by the routing tables in the routers on the way.

Ans. (d)

2. Which of the following functionalities must be implemented by a transport protocol over and above the network protocol?

(a) Recovery from packet losses
(b) Detection of duplicate packets
(c) Packet delivery in the correct order
(d) End-to-end connectivity

(GATE 2003: 1 Mark)

Solution: End-to-end delivery is the responsibility of a transport layer.

Ans. (d)

3. The subnet mask for a particular network is 255.255.31.0. Which of the following pairs of IP addresses could belong to this network?

(a) 172.57.88.62 and 172.56.87.233
(b) 10.35.28.2 and 10.35.29.4
(c) 191.203.31.87 and 191.234.31.88
(d) 128.8.129.43 and 128.8.161.55

(GATE 2003: 2 Marks)

Solution: The two addresses should belong to the same network, if binary AND operation of both addresses with net mask should be same.

The last octets of IP addresses of 0 is 000 0000.
The last octets of IP address of 43 and 55 and their AND with net mask gives the same result.

Ans. (d)

4. A 2-km long broadcast LAN has 10^7 bps bandwidth and uses CSMA/CD. The signal travels along the wire at 2×10^8 m/s. What is the minimum packet size that can be used on this network?

(a) 50 bytes (b) 100 bytes
(c) 200 bytes (d) None of the above

(GATE 2003: 2 Marks)

Solution: Given that length $L = 2$ km $= 2000$ m

Bandwidth, $B = 10^7$ bps
Signal travels $S = 2 \times 10^8$ m/s

So, propagation delay $(T_p) = \dfrac{L}{S} = \dfrac{2\times10^3}{2\times10^8} = \dfrac{1}{10^5}$ s

In CSMA/CD network,

Round-trip delay $= 2 \times T_p = 2 \times 10^5$ s
The minimum packet size must take round-trip delay to transmit.
So, transmission delay (T_x) = Round-trip delay

Since $T_x = N/B$, where N is the number of bits to be transmitted.
Number of bits transmitted, N = Round-trip delay $\times B = 2 \times 10^5 \times 10^7 = 200$ bytes

Ans. (c)

5. Host A is sending data to host B over a full duplex link. A and B are using the sliding window protocol for flow control. The send and receive window sizes are 5 packets each. Data packets (sent only from A to B) are all 1000 bytes long and the transmission time for such a packet is 50 μs. Acknowledgement packets (sent only from B to A) are very small and require negligible transmission time. The propagation delay over the link is 200 μs. What is the maximum achievable throughput in this communication?

(a) 7.69×10^6 bps (b) 11.11×10^6 bps
(c) 12.33×10^6 bps (d) 15.00×10^6 bps

(GATE 2003: 2 Marks)

Solution:
Throughput = 1 window/RTT
Round-trip time (RTT) = Transmission time + 2 × Propagation time

$$= 50 \text{ ms} + 2 \times 200 \text{ ms} = 450 \text{ ms}$$

As the size of window is 5 packets and 1 packet contains 1000 bytes.
The total size of the packet in bytes is $5 \times 1000 = 50000$ bytes

Therefore, throughput $= \dfrac{5000 \text{ bytes}}{450\times10^{-6} \text{ s}}$

$$= 11.11\times10^6 \text{ bps}$$

Ans. (b)

6. Choose the best matching between Groups 1 and 2.

Group: 1	Group: 2
P. Data link layer	1. Ensures reliable transport of data over a physical point-to-point link
Q. Network layer	2. Encodes/decodes data for physical transmission
R. Transport layer	3. Allows end-to-end communication between two processes
	4. Routes data from one network node to the next

(a) P:1, Q:4, R:3 (b) P:2, Q:4, R:1
(c) P:2, Q:3, R:1 (d) P:1, Q:3, R:2

(GATE 2004: 1 Mark)

Solution: Data link layer provides reliable transmission of data over a physical link.
Network layer provides routing of data packets in the network.
Transport layer is responsible for end-to-end process communication.

Ans. (a)

7. Which of the following is NOT true with respect to a transparent bridge and a router?

(a) Both bridge and router selectively forward data packets.
(b) A bridge uses IP addresses while a router uses MAC addresses.
(c) A bridge builds up its routing table by inspecting incoming packets.
(d) A router can connect between a LAN and a WAN.

(GATE 2004: 1 Mark)

Solution: Both router and bridge selectively forward data packets and both can connect between a LAN and WAN. Routing and bridge builds their routing table by inspecting incoming packets but router and bridge both use MAC address. So, option (b) is correct.

Ans. (b)

8. How many 8-bit characters can be transmitted per second over a 9600 baudserial communication link using asynchronous mode of transmission with one start bit, eight data bits, two stop bits, and one parity bit?

(GATE 2004: 1 Mark)

(a) 600 (b) 800 (c) 876 (d) 1200

Solution:

$$\frac{9600}{1+8+2+1} = 800$$

Ans. (b)

9. A and B are the only two stations on an Ethernet. Each has a steady queue of frames to send. Both A and B attempt to transmit a frame, collide, and A wins the first back-off race. At the end of this successful transmission by A, both A and B attempt to transmit and collide. The probability that A wins the second back-off race is

(a) 0.5 (b) 0.625 (c) 0.75 (d) 1.0

(GATE 2004: 2 Marks)

Solution: At the first collision, both A and B will have first collision but A wins.
So, A's collision counter will be reinitialized to 0.
At the second collision, A and B will have first and second collisions, respectively.
So, A will have to select random number from [0,1].
But, B will have to select from [0, 1, 2, 3].
On mapping, the following are total favouring cases to A's winning.

0–1 0–2 0–3 1–2 1–3

So, the probability is $5/8 = 0.625$.

Ans. (b)

10. The routing table of a router is shown below:

Destination	Subnet Mask	Interface
128.75.43.0	255.255.255.0	Eth_0
128.75.43.0	255.255.255.128	Eth_1
192.12.17.5	255.255.255.255	Eth_3
Default		Eth_2

On which interface will the router forward packets addressed to destinations 128.75.43.16 and 192.12.17.10, respectively?

(a) Eth_1 and Eth_2 (b) Eth_0 and Eth_2
(c) Eth_0 and Eth_3 (d) Eth_1 and Eth_3

(GATE 2004: 2 Marks)

Solution: On performing AND operation between incoming IP address and subnet-mask and comparing the result with the destination.

If there is a match between multiple destinations, then select the destination with the longest length subnet mask.

128.75.43.16 matches with 128.75.43.0 and 128.75.43.0. But the packets addressed to 128.75.43.16 will be forwarded to Eth_1.

If a result is not matching with any of the given destinations, then the packet is forwarded to the default interface (here Eth_2).

Therefore, the packets addressed to 192.12.17.10 will be forwarded to Eth_2.

Ans. (a)

Common Data Questions 11 and 12: Consider three IP networks A, B and C. Host H_A in network A sends messages each containing 180 bytes of application data to a host H_C in network C. The TCP layer prefixes a 20-byte header to the message. This passes through an intermediate network B. The maximum packet size, including 20-byte IP header, in each network is

A: 1000 bytes
B: 100 bytes
C: 1000 bytes

The network A and B are connected through a 1 Mbps link, while B and C are connected by a 512 Kbps link (bps = bits per second).

11. Assuming that the packets are correctly delivered, how many bytes, including headers, are delivered to the IP layer at the destination for one application message, in the best case? Consider only data packets.

(a) 200 (b) 220
(c) 240 (d) 260

(GATE 2004: 2 Marks)

Solution: When all the 3 packets are delivered, the bytes are

$$(80 + 20) + (80 + 20) + (40 + 20) = 260$$

Ans. (d)

12. What is the rate at which application data is transferred to host H_C? Ignore errors, acknowledgements and other overheads.

(a) 325.5 Kbps (b) 354.5 Kbps
(c) 409.6 Kbps (d) 512.0 Kbps

(GATE 2004: 2 Marks)

Solution: Actual data sent =180 out of 260
So, data rate $(180/260) \times 512$ Kbps = 354.461 Kbps

Ans. (b)

13. Packets of the same session may be routed through different paths in:

(a) TCP, but not UDP (b) TCP and UDP
(c) UDP, but not TCP (d) Neither TCP nor UDP

(GATE 2005: 1 Mark)

Solution: Packets travel on network layer. TCP and UDP are transport layer protocols.

Ans. (b)

14. The address resolution protocol (ARP) is used for:

(a) finding the IP address from the DNS.
(b) finding the IP address of the default gateway.
(c) finding the IP address that corresponds to a MAC address.
(d) finding the MAC address that corresponds to an IP address.

(GATE 2005: 1 Mark)

Solution: ARP works to find the physical address of a machine.

Ans. (d)

15. The maximum window size for data transmission using the selective reject protocol with n-bit frame sequence numbers is

(a) 2^n (b) 2^{n-1} (c) $2^n - 1$ (d) 2^{n-2}

(GATE 2005: 1 Mark)

Solution: For selective reject protocol, window size = $2^n/2 = 2^{n-1}$

Ans. (b)

16. In a network of LANs connected by bridges, packets are sent from one LAN to another through intermediate bridges. Since more than one path may exist between two LANs, packets may have to be routed through multiple bridges. Why is the spanning tree algorithm used for bridgerouting?

(a) For shortest path routing between LANs
(b) For avoiding loops in the routing paths
(c) For fault tolerance
(d) For minimising collisions

(GATE 2005: 1 Mark)

Solution: Spanning tree protocol is used for bridge routing to avoid loops in routing paths.

Ans. (b)

17. An organisation has a class B network and wishes to form subnets for 64 departments. The subnet mask would be

(a) 255.255.0.0 (b) 255.255.64.0
(c) 255.255.128.0 (d) 255.255.252.0

(GATE 2005: 1 Mark)

Solution: For class B, 16 bits are reserved as network bits. To allocate 64 subnets, 6 bits are added to network bits. 22 bits are for network. Subnet mask is created by assigning 1 to all network bits.
So, mask is 255.255.252.0.

Ans. (d)

18. In a packet-switching network, packets are routed from source to destination along a single path having two intermediate nodes. If the message size is 24 bytes and each packet contains a header of 3 bytes, then the optimum packet size is

(a) 4 (b) 6 (c) 7 (d) 9

(GATE 2005: 2 Marks)

Solution: Optimal packet size is 9.

$9-3 = 6$ B will be transferred in one packet, so the total message will travel in 4 packets.

Ans. (d)

19. Suppose the round-trip propagation delay for a 10 Mbps Ethernet having 48-bit jamming signal is 46.4 s. The minimum frame size is

(a) 94 (b) 416
(c) 464 (d) 512

(GATE 2005: 2 Marks)

Solution: RTT = 46.4×10^{-6}s

10 Mb are sent in = 1 s
1 will be sent = $1/10^7$
48 bits will be in = 4.8×10^{-6} s
Total delay = 46.4 + 4.8 = 51.2
Minimum frame size = 51.2 × 10 = 512 bits

Ans. (d)

20. For which one of the following reasons does Internet Protocol (IP) use the time- to-live (TTL) field in the IP datagram header?

(a) Ensure packets reach destination within that time
(b) Discard packets that reach later than that time
(c) Prevent packets from looping indefinitely
(d) Limit the time for which a packet gets queued in intermediate routers

(GATE 2006: 1 Mark)

Solution: Decrementing TTL will prevent packet from looping.

Ans. (b)

21. Station A uses 32 byte packets to transmit messages to Station B using a sliding window protocol. The round trip delay between A and B is 80 milliseconds and the bottleneck bandwidth on the path between A and B is 128 kbps. What is the optimal window size that A should use?

(a) 20 (b) 40
(c) 160 (d) 320

(GATE 2006: 2 Marks)

Solution: Round-trip delay = 80 ms

Bandwidth = 128 Kbps
In 1 s = 128 Kbits are sent
In 80 ms = 128K × 80 × 10^{-3} = 128 × 80 bits

$$\text{Window size} = \frac{128 \times 80}{32 \times 8} = 40$$

Ans. (b)

22. Two computers C_1 and C_2 are configured as follows. C_1 has IP address 203.197.2.53 and netmask 255.255.128.0. C_2 has IP address 203.197.75.201 and netmask 255.255.192.0. Which one of the following statements is true?

(a) C_1 and C_2 both assume they are on the same network.
(b) C_2 assumes C_1 is on the same network, but C_1 assumes C_2 is on a different network.
(c) C_1 assumes C_2 is on the same network, but C_2 assumes C_1 is on a different network.
(d) C_1 and C_2 both assume they are on different networks.

(GATE 2006: 2 Marks)

Solution: Network address of C_1: 203.197.2.53 AND 255.255.128.0 = 203.197.0.0

Network address of C_2: 203.197.75.201 AND 255.255.192.0 = 203.197.0.0
C_1 assumes C_2 is on the same network, but C_2 assumes C_1 on different network.

Ans. (c)

23. Station A needs to send a message consisting of 9 packets to station B using a sliding window (window size 3) and Go-Back-N error control strategy. All packets are ready and immediately available for transmission. If every 5thpacket that A transmits gets lost (but no packets from B ever get lost), then what is the number of packets that A will transmit for sending the message to B?

(a) 12 (b) 14 (c) 16 (d) 18

(GATE 2006: 2 Marks)

Solution:

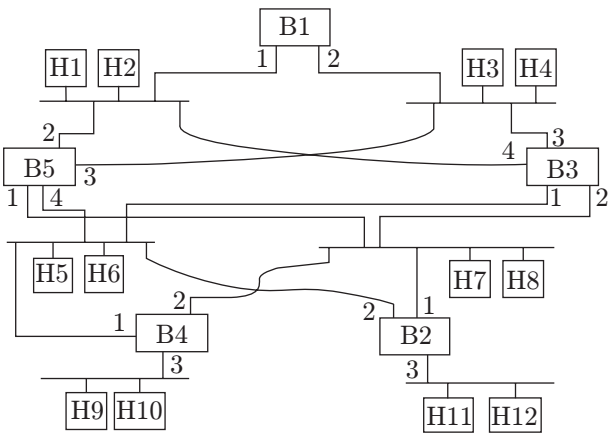
A		B
1	→	1
2	→	2
3	→	3
4	→	4
5	→	lost
6	→	discard
7	→	discard
8	→	5
9	→	6
10	→	lost 7
11	→	discard 8
12	→	discard 9
13	→	7
14	→	8
15	→	lost 9
16	→	9

Ans. (c)

Linked Answer Questions 24 and 25: Consider the diagram shown below where a number of LANs

are connected by (transparent) bridges. To avoid packets looping through circuits in the graph, the bridges organize themselves in a spanning tree. First, the root bridge is identified as the bridge with the least serial number. Next, the root sends out (one or more) data units to enable the setting up of the spanning tree of shortest paths from the root bridge to each bridge.

Each bridge identifies a port (the root port) through which it will forward frames to the root bridge. Port conflicts are always resolved in favour of the port with the lower index value. When there is a possibility of multiple bridges forwarding to the same LAN (but not through the root port), ties are broken as follows: bridges closest to the root get preference and between such bridges the one with the lowest serial number is preferred.



24. For the given connection of LANs by bridges, which one of the following choices represents the depth first traversal of the spanning tree of bridges?

(a) B_1, B_5, B_3, B_4, B_2 (b) B_1, B_3, B_5, B_2, B_4
(c) B_1, B_5, B_2, B_3, B_4 (d) B_1, B_3, B_4, B_5, B_2

Solution: DFS (depth first search) is analgorithm for traversing tree or graph. One starts at the root (selecting some arbitrary node as the root in the case of a graph) and explores as far as possible along each branch before backtracking. So depth first search traversal is B_1, B_5, B_3, B_4, B_2.

(GATE 2006: 2 Marks)

Ans. (a)

25. Consider the correct spanning tree for the previous question. Let host H_1 send out a broadcast ping packet. Which of the following options represents the correct forwarding table on B_3?

(a)

Hosts	Port
H_1, H_2, H_3, H_4	3
H_5, H_6, H_9, H_{10}	1
H_7, H_8, H_{11}, H_{12}	2

(b)

Hosts	Port
H_1, H_2	4
H_3, H_4	3
H_5, H_6	1
H_5, H_8, H_9, H_{10}, H_{11}, H_{12}	2

(c)

Hosts	Port
H_3, H_4	3
H_5, H_6, H_9, H_{10}	1
H_1, H_2	4
H_7, H_8, H_{11}, H_{12}	2

(d)

Hosts	Port
H_1, H_2, H_3, H_4	3
H_5, H_7, H_9, H_{10}	1
H_7, H_8, H_{11}, H_{12}	4

(GATE 2006: 2 Marks)

Solution: Forwarding table for B_3 is

Hosts	Port
H_1, H_2, H_3, H_4	3
H_5, H_6, H_9, H_{10}	1
H_7, H_8, H_{11}, H_{12}	2

Ans. (a)

26. In Ethernet, when Manchester encoding is used, the bit rate is

(a) Half the baud rate (b) Twice the baud rate
(c) Same as the baud rate (d) None of the above

(GATE 2007: 1 Mark)

Solution: For transmission of digital information, Manchester coding is used to convert digital information into electrical signals for transmission. It uses two baud for one bit.

Ans. (b)

27. Which one of the following uses UDP as the transport protocol?

(a) HTTP (b) Telnet
(c) DNS (d) SMTP

(GATE 2007: 1 Mark)

Solution: DNS uses services of UDP protocol.

Ans. (c)

28. There are n stations in a slotted LAN. Each station attempts to transmit with a probability p in each time slot. What is the probability that ONLY one station transmits in a given time slot?

(a) $np(p\ 1)^{n-1}$ (b) $(1-p)^{n-1}$
(c) $p(p\ 1)^{n-1}$ (d) $1-(p\ 1)^{n-1}$

(GATE 2007: 2 Marks)

Solution: Probability of sending by one station = $p(1-p)^{n-1}$

For n stations, it is $np(1-p)^{n-1}$

Ans. (a)

29. In a token ring network, the transmission speed is 10^7 bps and the propagation speed is 200 m/μs. The 1-bit delay in this network is equivalent to:

(a) 500 m of cable (b) 200 m of cable
(c) 20 m of cable (d) 50 m of cable

(GATE 2007: 2 Marks)

Solution: Transmission delay for 1 bit $t = 1/(10^7)$ = 0.1 μs.

200 m can be travelled in 1 μs. Therefore, in 0.1 μs, 20 m can be travelled.

Ans. (c)

30. The address of a class B host is to be split into subnets with a 6-bit subnet number. What is the maximum number of subnets and the maximum number of hosts in each subnet?

(a) 62 subnets and 262142 hosts
(b) 64 subnets and 262142 hosts
(c) 62 subnets and 1022 hosts
(d) 64 subnets and 1024 hosts

(GATE 2007: 2 Marks)

Solution: Here 16 + 6 = 22 bits are reserved for the network

Number of hosts = $2^{32-22} = 2^{10} - 2 = 1022$
Number of subnets = $2^6 - 2 = 62$

Ans. (c)

31. The message 11001001 is to be transmitted using the CRC polynomial $x^3 + 1$ to protect it from errors. The message that should be transmitted is

(a) 11001001000 (b) 11001001011
(c) 11001010 (d) 110010010011

(GATE 2007: 2 Marks)

Solution: The polynomial is equivalent to 1001. Divide 1001 with 11001001000 and find the remainder. Remainder is 011. Append 011 at the end of input string, it will be 11001001011.

Ans. (b)

32. The distance between two stations M and N is L km. All frames are K bits long. The propagation delay per kilometre is t s. Let R bits/s be the channel capacity. Assuming that processing delay is negligible, the minimum number of bits for the sequence number field in a frame for maximum utilisation, when the sliding window protocol is used, is

(a) $\left\lceil \log_2 \frac{2LtR+2K}{K} \right\rceil$ (b) $\left\lceil \log_2 \frac{2LtR}{K} \right\rceil$
(c) $\left\lceil \log_2 \frac{2LtR+K}{K} \right\rceil$ (d) $\left\lceil \log_2 \frac{2LtR+K}{2K} \right\rceil$

(GATE 2007: 2 Marks)

Solution: Distance between stations M and N = L km
Propagation delay = t s
Total propagation delay = Lt s
Frame size = K bits
Channel capacity = R bits/s
Transmission time = K/R
Let n be the window size.

$$\text{Utilisation} = \frac{n}{1+2a}, \text{ where } a = \frac{\text{Propagation time}}{\text{Transmission}}$$

$$= \frac{n}{1+(2LtR\,/\,K)} = \frac{nK}{2LtR+K}$$

For maximum utilisation: $nK = 2LtR + K$

Therefore, $$n = \frac{2LtR+K}{K}$$

Number of bits needed for n frames is log n.

Ans. (c)

33. Match the following:

Column I	Column II
(P) SMTP	(1) Application layer
(Q) BGP	(2) Transport layer
(R) TCP	(3) Data link layer
(S) PPP	(4) Network layer
	(5) Physical layer

(a) P – 2, Q – 1, R – 3, S – 5
(b) P – 1, Q – 4, R – 2, S – 3
(c) P – 1, Q – 4, R – 2, S – 5
(d) P – 2, Q – 4, R – 1, S – 3

(GATE 2007: 2 Marks)

Solution:
SMTP: Application layer for mail transfer
BGP: Network layer routing protocol
TCP: Transport layer transmission protocol
PPP: Data link layer protocol for direct connections

Ans. (b)

34. What is the maximum size of data that the application layer can pass on to the TCP layer below?

(A) Any size
(B) 2^{16} bytes – size of TCP header
(C) 2^{16} bytes
(D) 1500 bytes

(GATE 2008: 1 Mark)

Solution:
There is no limit of data passing at application layer.

Ans. (a)

35. Which of the following system calls results in the sending of SYN packets?

(a) Socket (b) Bind
(c) Listen (d) Connect

(GATE 2008: 1 Mark)

Solution: Connect() is called by the client and connection is established using three-way hand shake.

Ans. (d)

36. In the slow start phase of the TCP congestion control algorithm, the size of the congestion window

(a) does not increase.
(b) increases linearly.
(c) increase quadratically.
(d) increase exponentially.

(GATE 2008: 2 Marks)

Solution: In the slow start phase of the TCP congestion control algorithm, the size of the congestion window increases exponentially.

Ans. (d)

37. If a class B network on the Internet has a subnet mask of 255.255.248.0, what is the maximum number of hosts per subnet?

(a) 1022 (b) 1023
(c) 2046 (d) 2047

(GATE 2008: 2 Marks)

Solution: Number of network bits = 21

Number of host bits = 11
Number of hosts = $2^{11} - 2 = 2046$

Ans. (c)

38. A computer on a 10 Mbps network is regulated by a token bucket. The token bucket is filled at a rate of 2 Mbps. It is initially filled to capacity with 16 Mbits.

What is the maximum duration for which the computer can transmit at the full 10 Mbps?

(a) 1.6 s (b) 2 s
(c) 5 s (d) 8 s

(GATE 2008: 2 Marks)

Solution: Data transfer of token bucket = 10 Mbps
Rate of transfer = 2 Mbps

Initially filled capacity = 16 Mbits

$$\text{Maximum burst time} = \frac{b}{M-r} = \frac{16}{10-2} = 2 \text{ s}$$

Ans. (b)

39. A client process P needs to make a TCP connection to a server process S. Consider the following situation: the server process S executes a socket(), a bind() and a listen() system call in that order, following which it is pre-empted.

Subsequently, the client process P executes a socket() system call followed by connect() system call to connect to the server process S. The server process has not executed any accept() system call. Which one of the following events could take place?

(a) Connect() system call returns successfully
(b) Connect() system call blocks
(c) Connect() system call returns an error
(d) Connect() system call results in a core dump

(GATE 2008: 2 Marks)

Solution: The accept() call does not execute. So, connect() call did not get response for a time stamp to wait, therefore, connect() system call returns an error.

Ans. (c)

40. In the RSA public key cryptosystem, the private and public keys are (e,n) and (d,n), respectively, where $n = p^* q$ and p and q are large primes. Besides, n is public and p and q are private. Let M be an integer such that $0 < M < n$ and $\phi(n) = (p-1)(q-1)$. Now, consider the following equations:

I. $M' = M^e \bmod n$
$M = (M')^d \bmod n$
II. $ed \equiv 1 \bmod n$
III. $ed \equiv 1 \bmod \phi(n)$
IV. $M' = M^e \bmod \phi(n)$
$M = (M')^d \bmod \phi(n)$

Which of the above equations correctly represent the RSA cryptosystem?

(a) I and II (b) I and III
(c) II and IV (d) III and IV

(GATE 2009: 2 Marks)

Solution: Encryption: $M' = M^e \bmod n$
Decryption: $M = (M')^d \bmod n$
$ed \equiv 1 \bmod f(n)$

Ans. (b)

41. While opening a TCP connection, the initial sequence number is to be derived using a time-of-day (ToD) clock that keeps running even when the host is down. The low order 32 bits of the counter of the ToD clock is to be used for the initial sequence numbers. The clock counter increments once per millisecond. The maximum packet lifetime is given to be 64s.

Which one of the choices given below is closest to the minimum permissible rate at which sequence numbers used for packets of a connection can increase?

(a) 0.015/s (b) 0.064/s
(c) 0.135/s (d) 0.327/s

(GATE 2009: 2 Marks)

Solution: The frames from the sending station are numbered sequentially.

Ans. (b)

42. Let $G(x)$ be the generator polynomial used for CRC checking. What is the condition that should be satisfied by $G(x)$ to detect odd number of bits in error?

(a) $G(x)$ contains more than two terms.
(b) $G(x)$ does not divide $1+x^k$, for any not exceeding the frame length k.
(c) $1+x$ is a factor of $G(x)$.
(d) $G(x)$ has an odd number of terms.

(GATE 2009: 2 Marks)

Solution: To detect odd number of errors, $(x + 1)$ must be present.

Ans. (c)

Linked Answer Questions 43 and 44: Frames of 1000 bits are sent over a 10 bps duplex link between two hosts. The 6 propagation time is 25 ms. Frames are to be transmitted into this link to maximally pack them in transit (within the link).

43. What is the minimum number of bits (l) that will be required to represent the sequence numbers distinctly? Assume that no time gap needs to be given between transmissions of two frames.

(a) $l = 2$ (b) $l = 3$
(c) $l = 4$ (d) $l = 5$

(GATE 2009: 2 Marks)

Solution: Transmission delay of link = $1000/10^6$ = 1 ms

Propagation delay = 25 ms
Maximum 25 frames can be sent, for 25 frames bits required are 5.

Ans. (d)

44. Suppose that the sliding window protocol is used with the sender window size of 2^l, where l is the number of bits identified in the earlier part and l acknowledgements are always piggy backed. After sending 2^l frames, what is the minimum time the sender will have to wait before starting transmission of the next frame? (Identify the closest choice ignoring the frame processing time.)

(a) 16 ms (b) 18 ms (c) 20 ms (d) 22 ms

(GATE 2009: 2 Marks)

Solution: Size of window = 32

Round trip time = 2 × 25 ms = 50 ms
To send the next frame, the sender has to wait for 50 − 32 = 18 ms

Ans. (b)

45. One of the header fields in an IP datagram is the time-to-live (TTL) field. Which of the following statements best explains the need for this field?

(a) It can be used to prioritise packets.
(b) It can be used to reduce delays.
(c) It can be used to optimise throughput.
(d) It can be used to prevent packet looping.

(GATE 2010: 1 Mark)

Solution: Value of TTL is decremented to prevent packet looping.

Ans. (d)

46. Which one of the following is not a client–server application?

(a) Internet chat (b) Web browsing
(c) E-mail (d)Ping

(GATE 2010: 1 Mark)

Solution: ping is a command not an application. It is to check connectivity and there is no need to communicate with server.

Ans. (d)

47. Suppose computers A and B have IP addresses 10.105.1.113 and 10.105.1.91, respectively, and they both use the same net mask N. Which of the values of N given below should not be used if A and B should belong to the same network?

(a) 255.255.255.0 (b) 255.255.255.128
(c) 255.255.255.192 (d) 255.255.255.224

(GATE 2010: 2 Marks)

Solution: To belong to a different network, A and B should have different network address.

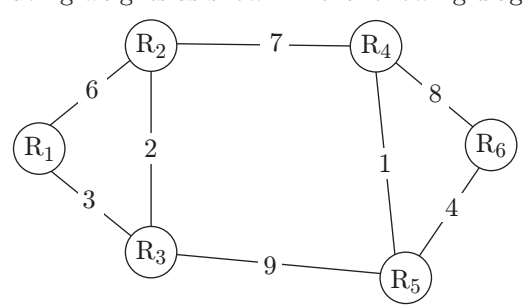
Perform logical AND operation on all the options.
For example,option (d)
IP address of A: 10.105.1. 01110001
Network mask: 255.255.255. 11100000
Network address of A: 10.105.1. 01100000

IP address of B: 10.105.1. 01011011
Network mask: 255.255.255. 11100000
Network address of A: 10.105.1. 01000000

Both have different network addresses.

Ans. (d)

Linked Answer Questions 48 and 49: Consider a network with 6 routers R_1 to R_6 connected with links having weights as shown in the following diagram



48. All the routers use the distance vector based routing algorithm to update their routing tables. Each router starts with its routing table initialised to contain an entry for each neighbour with the

weight of the respective connecting link. After all the routing tables stabilise, how many links in the network will never be used for carrying any data?

(a) 4 (b) 3 (c) 2 (d) 1

(GATE 2010: 2 Marks)

Solution: We can check one by one all shortest distances. When we check for all shortest distances for R_i we don't need to check its distances to R_0 to R_i-1 because the network graph is undirected. Following will be distance vectors of all nodes.

Shortest distances from R_1 to R_2, R_3, R_4, R_5 and R_6
R_1 (5, 3, 12, 12, 16)
Links used: R_1–R_3, R_3–R_2, R_2–R_4, R_3–R_5, R_5–R_6

Shortest distances from R_2 to R_3, R_4, R_5 and R_6
R_2 (2, 7, 8, 12)
Links used: R_2–R_3, R_2–R_4, R_4–R_5, R_5–R_6

Shortest distances from R_3 to R_4, R_5 and R_6
R_3 (9, 9, 13)
Links used: R_3–R_2, R_2–R_4, R_3–R_5, R_5–R_6

Shortest distances from R_4 to R_5 and R_6
R_4 (1, 5)
Links used: R_4–R_5, R_5–R_6

Shortest distance from R_5 to R_6
R_5 (4)

Links used: R_5–R_6

If we mark all the used links one by one, we can see that following links are never used.
R_1–R_2 R_4–R_6

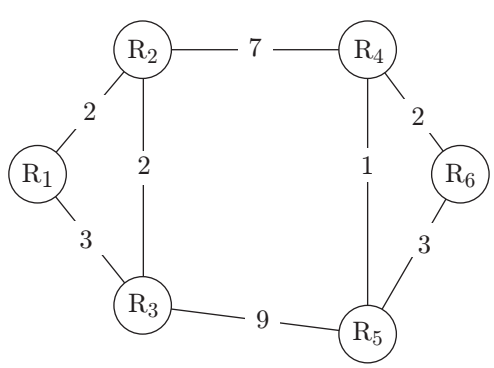
Ans. (c)

49. Suppose the weights of all unused links in the previous question are changed to 2 and the distance vector algorithm is used again until all routing tables stabilise. How many links will now remain unused?

(a) 0 (b) 1 (c) 2 (d) 3

(GATE 2010: 2 Marks)

Solution: After the weights of unused links() are changed to following graph.



Following will be distance vectors of all nodes
R_1 (2, 3, 9, 10, 11)
Links used: R_1–R_2, R_1–R_3, R_2–R_4, R_4–R_5, R_4–R_6
R_2 (2, 7, 8, 9)
Links used: R_2–R_3, R_2–R_4, R_4–R_5, R_4–R_6
R_3 (9, 9, 11)
Links used: R_3–R_2, R_2–R_4, R_3–R_5, R_4–R_6
R_4 (1, 2)
Links used: R_4–R_5, R_4–R_6
R_5 (3)
Links used: R_5–R_4, R_4–R_6
If we mark all the used links one by one, we can see that following links are never used.
R_5–R_6

Ans. (b)

50. A layer-4 firewall (a device that can look at all protocol headers up to the transport layer) CANNOT

(a) Block entire HTTP traffic during 9:00PM and 5:00 AM
(b) Block all ICMP traffic
(c) Stop incoming traffic from a specific IP address but allow outgoing traffic to the same IP address
(d) Block TCP traffic from a specific user on a multi-user system during 9:00 PM and 5:00 AM

(GATE 2011: 1 Mark)

Solution: As Layer-4 firewall cannot block traffic of Layer-5 (application layer) and HTTP is an application layer protocol. So, option (a) is correct.

Ans. (a)

51. Consider different activities related to email.

m_1: Send an email from a mail client to a mail server
m_2: Download an email from mailbox server to a mail client
m_3: Checking email in a web browser

Which is the application level protocol used in each activity?

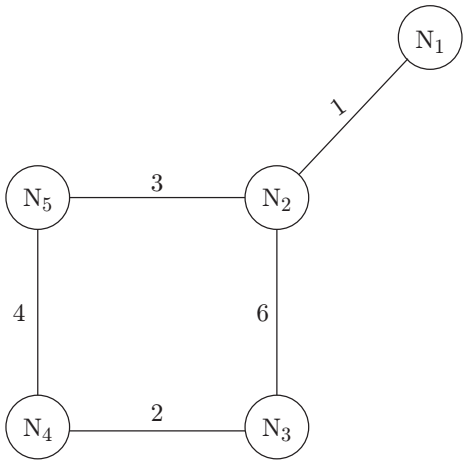
(a) m_i: HTTP m_2: SMTP m_3: POP
(b) m_i: SMTP m_2: FTP m_3: HTTP
(c) m_i: SMTP m_2: POP m_3: HTTP
(d) m_i: POP m_2: SMTP m_3: IMAP

(GATE 2011: 1 Mark)

Solution:
m_1: SMTP is responsible for mail transfer.
m_2: POP is responsible for downloading mail from mailbox server to mal client.
m_3: HTTP is responsible for viewing application on web browser.

Ans. (c)

Linked Answer Questions 52 and 53: Consider a network with five nodes, N_1 to N_5, as shown below.



The network uses a distance vector routing protocol. Once the routes have stabilised, the distance vectors at different nodes are as follows:

N_1:(0,1, 7, 8, 4) N_2: (1, 0, 6, 7, 3)
N_3: (7, 6, 0, 2, 6) N_4: (8,7, 2,0,4)
N_5: (4, 3, 6, 4, 0)

Each distance vector is the distance of the best-known path at that instance to nodes, N_1 to N_5, where the distance to itself is O. Also, all links are symmetric and the cost is identical in both directions. In each round, all nodes exchange their distance vectors with their respective neighbours. Then all nodes update their distance vectors. In between two rounds, any change in cost of a link will cause the two incident nodes to change only that entry in their distance vectors.

52. The cost of link N_2–N_3 reduces to 2 (in both directions). After the next round of updates, what will be the new distance vector at node N_3?

(a) (3, 2, 0, 2, 5) (b) (3, 2, 0, 2, 6)
(c) (7, 2, 0, 2, 5) (d) (7, 2, 0, 2, 6)

(GATE 2011: 2 Marks)

Solution: N_3: (3, 2, 0, 2, 5)

Ans. (a)

53. After the update in the previous question, the link N_1–N_2 goes down. N_2 will reflect this change immediately in its distance vector as cost, ∞. After the NEXT ROUND of update, what will be the cost to N_1 in the distance vector of N_3?

(a) 3 (b) 9 (c) 10 (d) ∞

(GATE 2011: 2 Marks)

Solution: N_3 to N_1 either by N_2 or by N_4

N_2–N_1 is ∞, so N_3 will choose to go via N_4 to N_1 (2 + 8).

Ans. (c)

54. Which of the following transport layer protocols is used to support electronic mail?

(a) SMTP (b) IP (c) TCP (d) UDP

(GATE 2012: 1 Mark)

Solution: Electronic mail does not require TCP connection between sender and receiver of email.

Ans. (c)

55. The protocol data unit (PDU) for the application-layer in the Internet stack is

(a) segment. (b) datagram.
(c) message. (d) frame.

(GATE 2012: 1 Mark)

Solution:
The protocol data unit (PDU) for Data link layer = Frame; Network layer = Datagram; Transport layer = Segment; Application layer = Message.

Ans. (c)

56. In the IPv4 addressing format, the number of networks allowed under class C addresses is

(a) 2^{14} (b) 2^7 (c) 2^{21} (d) 2^{24}

(GATE 2012: 1 Mark)

Solution:

Network bits	Host bits
24 bits	8 bits

Starting with 3 bits (110) reserved to recognise the class. So, the number of networks are 2^{21}.

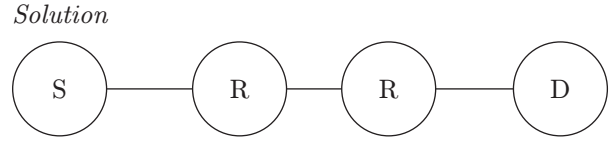
Ans. (c)

57. Consider a source computer (S) transmitting a file of size 10^6 bits to a destination computer (D) over a network of two routers (R_1 and R_2) and three links (L_1, L_2, and L_3). L_1 connects S to R_1; L_2 connects R_1 to R_2; and L_3 connects R_2 to D. Let each link be of length 100 km. Assume signals travel over each link at a speed of 10^8 meters per second. Assume that the link bandwidth on each link is 1Mbps. Let the file be broken down into 1000 packets each of size 1000 bits. Find the total sum of transmission and propagation delays in transmitting the file from S to D?

(a) 1005 ms (b) 1010 ms
(c) 3000 ms (d) 3003 ms

(GATE 2012: 2 Marks)

Solution



Transmission delay for 1 packet from each of S, R_1 and R_2 will take 1ms.

Propagation delay on L_1, L_2 and L_3 for one packet is 1 ms.
Transmission delay + Propagation delay = 2 ms
First packet will reach at 6th ms
Second packet will reach at 7th ms
1000 packets will reach at 1005th ms

Ans. (a)

58. Consider an instance of TCP's Additive Increase Multiplicative Decrease (AIMD) algorithm where the window size at the start of the slow start phase is 2 MSS and the threshold at the start of the first transmission is 8 MSS. Assume that a timeout occurs during the fifth transmission. Find the congestion window size at the end of the tenth transmission.

(a) 8 MSS (b) 14 MSS
(c) 7 MSS (d) 12 MSS

(GATE 2012: 2 Marks)

Solution:
In slow start, with each iteration, size of congestion window doubles. So,

T = 1 ws = 2
T = 2 ws = 4
T = 3 ws = 8

Threshold is 8. So now window size will increase by one using additive increase method.

T = 4 ws = 9
T = 5 ws = 10, here timeout occurs.

Hence, threshold = 10/2 = 5, ws is reset

T = 6 ws = 2
T = 7 ws = 4
T = 8 ws = 5 [threshold was 5, so apply additive increase]
T = 9 ws = 6
T = 10 ws = 7

Ans. (c)

59. An Internet Service Provider (ISP) has the following chunk of CIDR-based IP addresses available with it: 245.248.128.0/20. The ISP wants to give half of this chunk of addresses to organisation A, and a quarter to organisation B, while retaining the remaining with itself. Which of the following is a valid allocation of addresses to A and B?

(a) 245.248.136.0/21 and 245.248.128.0/22
(b) 245.248.128.0/21 and 245.248.128.0/22
(c) 245.248.132.0/22 and 245.248.132.0/21
(d) 245.248.136.0/24 and 245.248.132.0/21

(GATE 2012: 2 Marks)

Solution: Total numbers of addresses available are 4096.

Organisation A	:	2048
Organisation B	:	1024
Remaining	:	1024

Allocation of addresses to A: 11 bits are required for organisation A.

$\boxed{245.248.\ 10000}$ $\boxed{100.00000000}$

Address allocated is 245.248.136.0/21
Allocation of addresses to B: 10 bits are required for organisation B.

$\boxed{245.248.\ 100000}$ $\boxed{00.00000000}$

Address allocated is 245.248.128.0/22.

Ans. (a)

60. The transport layer protocols used for real time multimedia, file transfer, DNS and email, respectively, are

(a) TCP, UDP, UDP and TCP
(b) UDP, TCP, TCP and UDP
(c) UDP, TCP, UDP and TCP
(d) TCP, UDP, TCP and UDP

(GATE 2013: 1 Mark)

Solution: Real-time multimedia: UDP (session less protocol, used where fast data transfer is required)

File transfer: TCP
DNS: UDP
E-mail: TCP

Ans. (c)

61. Using public key cryptography, X adds a digital signature σ to message M, encrypts $< M, \sigma >$, and sends it to Y, where it is decrypted. Which one of the following sequences of keys is used for the operations?

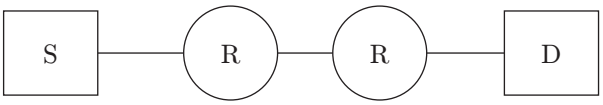
(a) Encryption: X's private key followed by Y's private key; Decryption: X's public key followed by Y's public key
(b) Encryption: X's private key followed by Y's public key; Decryption: X's public key followed by Y's private key
(c) Encryption: X's public key followed by Y's private key; Decryption: Y's public key followed by X's private key
(d) Encryption: X's private key followed by Y's public key; Decryption: Y's private key followed by X's public key

(GATE 2013: 1 Mark)

Solution: X uses private key to add digital signature, then uses Y's public key to encrypt. On the other end, Y first uses its private key followed by X's public key.

Ans. (d)

62. Assume that source S and destination D are connected through two intermediate routers labelled R.



Determine how many times each packet has to visit the network layer and the data link layer during a transmission from S to D.

(a) Network layer – 4 times and Data link layer – 4 times
(b) Network layer – 4 times and Data link layer – 3 times
(c) Network layer – 4 times and Data link layer – 6 times
(d) Network layer – 2 times and Data link layer – 6 times

(GATE 2013: 1 Mark)

Solution: Transmission from S to D will follow these steps:

At S: Network layer → Data link layer
At R1: Data link layer → Network layer → Data link layer
At R2: Data link layer → Network layer → Data link layer
At D: Data link layer → Network layer
Therefore, the packet has to visit 6 times data link layer and 4 times network layer.

Ans.(c)

63. Determine the maximum length of the cable (in km) for transmitting data at a rate of 500 Mbps in an Ethernet LAN with frames of size 10000 bits. Assume the signal speed in the cable to be 200000 km/s.

(a) 1 (b) 2
(c) 2.5 (d) 5

(GATE 2013: 2 Marks)

Solution: Data rate: 500 Mbps = 5×10^8 bps
Frame size: 10000 bits

Speed: 200000 km/s

$$5 \times 10^8 \text{ bits can travel} = 1\text{s}$$

$$10^4 \text{ bits will travel} = \frac{10^4}{5\times10^8} = \frac{1}{5\times10^4}$$

In 1 s, distance travelled = 2×10^5 km

$$\text{In } (1/5\times10^4) \text{ s, distance travelled is} = \frac{2\times10^5}{5\times10^4} = 4 \text{ km}$$

Hence, the maximum distance is 4/2 = 2 km

Ans. (b)

64. In an IPv4 datagram, the M bit is 0, the value of HLEN is 10, the value of total length is 400 and the fragment offset value is 300. The position of the datagram, the sequence numbers of the first and the last bytes of the payload, respectively, are

(a) Last fragment, 2400 and 2789
(b) First fragment, 2400 and 2759
(c) Last fragment, 2400 and 2759
(d) Middle fragment, 300 and 689

(GATE 2013: 2 Marks)

Solution: $M = 0$ indicates no more fragment means last fragment

Actual header length is $4 \times 10 = 40$
Total length = 400 bytes (40-byte header + 360-byte payload)
Fragment offset = $300 \times 8 = 2400$ bytes (measured in terms of 8 bytes)
First byte sequence number of last fragment is 2400.
Sequence number of last byte payload is $2400 + 360 - 1 = 2759$.

Ans. (c)

PRACTICE EXERCISES

Set 1

1. Which layer is associated with log in/log out from the network?

(a) Transport (b) Presentation
(c) Data link (d) Session

2. Which layer is associated with IP addresses?

(a) Session (b) Network
(c) Transport (d) Data link

3. The size of MAC address and IPv4 address is

(a) 8 bits and 24 bits (b) 32 bits and 48 bits
(c) 48 bits and 32 bits (d) 24 bits and 32 bits

4. The bit size of cyclic redundancy code in an Ethernet is

(a) CRC is not used in Ethernet
(b) 8
(c) 24
(d) 32

5. MTU stands for

(a) Minimum Transfer Unit
(b) Minimum Telephony Unit
(c) Maximum Transfer Unit
(d) Memory Transfer Unit

6. Using 7-bit sequence number, what is the maximum size (in bits) of the sender and receiver window using Go-Back-N ARQ?

(a) 127 and 1 (b) 1 and 127
(c) 127 and 127 (d) 1 and 1

7. Using a 5-bit sequence number, what is the maximum size of the sender and receiver window using selective repeat ARQ?

(a) 16 and 16 (b) 1 and 16
(c) 16 and 1 (d) Depend upon the bits selected to be repeated

8. Which protocol is being used in wireless to overcome collision?

(a) CSMA/CD
(b) CSMA/CA
(c) Both
(d) WEP

9. The host bits present in class B IP address are _____.

10. Which protocol is used to provide error reporting services to IP address?

(a) ICMP (b) ARP
(c) BGP (d) OSPF

11. Which protocol is used to find the physical address for a logical address?

(a) ARP (b) RARP
(c) IGMP (d) ICMP

12. Which protocol cannot be used to find the logical address if the server is residing out of your network?

(a) ARP (b) RARP
(c) BOOTP (d) IGMP

13. Which topology requires a central controller or hub?

(a) Bus (b) Star
(c) Mesh (d) Ring

14. The number of nibbles are reserved in the IP header for header length is _____.

15. How many hops can be travelled by the IP packet having TTL value 5?

(a) It cannot travel any hop.
(b) It can travel only single hop.
(c) It can travel five hops.
(d) It can travel four hops.

16. What does the acronym ISDN for?

(a) Indian Standard Data Network
(b) Integrated Services Digital Networks
(c) Intelligent Secure Digital Network
(d) Intelligent Services Digital Network

17. If N is the maximum sequence number, then the window size in selective repeat and Go-Back-N protocols are, respectively,

(a) N/2, N − 1 (b) N/2, N + 1
(c) N + 1/2, N − 1/2 (d) N − 1, N

18. CRC can detect all burst error of up to M errors, if generator polynomial $G(x)$ is of degree

(a) It does not matter (b) $M - 1$
(c) $M/2$ (d) $M + 1$

19. Which layers of the OSI reference model are host-to-host layers?

(a) Transport, session, presentation, application
(b) Session, presentation, application
(c) Datalink, transport, presentation, application
(d) Physical, datalink, network, transport

20. A nibble is used for packed sequence numbering in a sliding window protocol used in a computer network. What is the maximum window size?

(a) Depends upon the underlying network protocol
(b) Depends upon the number of nodes involved in the network
(c) 15
(d) 16

21. Infrared signals can be used in a closed area using _____ propagation.

(a) Sky (b) Ground
(c) Line of sight (d) Small signal

22. A bridge has access to _____ address in the same network.

(a) Physical (b) Logical
(c) Supernet (d) Link state

23. Match the following:

List – I	List – II
A. Physical layer	i. Resources to network access are used
B. Data link layer	ii. Packets are moved from one destination to other
C. Network layer	iii. Process to process message delivery
D. Transport layer	iv. Bit stream is transmitted
E. Application Layer	v. Frames are formed

Codes:

	A	B	C	D	E
(a)	iv	v	ii	iii	i
(b)	i	iii	iv	ii	v
(c)	iv	ii	i	v	iii
(d)	i	ii	iii	iv	v

24. The _____ is a unit to measure the signal strength in wireless networks

(a) Frequency (b) Bandwidth delay product
(c) Attenuation (d) Decibel

25. Which one of the following media is multi-drop?

(a) UTP cable (b) STP cable
(c) Thick coaxial cable (d) Multi-mode fibre optic cable

26. What is the baud rate of the standard 20 Mbps Ethernet?

(a) 10 megabaud (b) 40 megabaud
(c) 30 megabaud (d) 20 megabaud

27. Binary symmetric channel uses

(a) half-duplex protocol
(b) full-duplex protocol
(c) simplex protocol
(d) simplex protocol with separate data and control bits

28. Which of the following addresses is used to deliver a message to the correct application program running on a host?

(a) Port (b) DNS address
(c) Logical address (d) MAC address

29. Which of the following protocol is used for performing RPCs between applications in a language and system in an independent way?

(a) DHCP (b) SNMP
(c) SOAP (d) SMTP

30. Which layer of OSI reference model is responsible for decomposition of messages and generation of sequence numbers to ensure correct re-composition from end-to-end communication in a network?

(a) Physical (b) Session
(c) Transport (d) Data link

31. The VLF and LF bauds use _____ propagation for communication.

(a) Ground (b) Sky
(c) Line of sight (d) Space

32. Using the 8-bit sequence numbers, what is the maximum size of the sender and receiver window in selective repeat ARQ?

(a) 128 and 128
(b) 1 and 127
(c) 128 and 127
(d) 8 and 8

33. Vulnerable Slotted ALOHA is

(a) equal to average frame transmission time
(b) half of average frame transmission time
(c) two times the average frame transmission time
(d) none

34. A CSMA/CD network supports data rate of 10Mbps, the maximum distance between any two stations is found to be 2500 m for the correct operation of the collision detection process. What should be the maximum distance if the data rate is increased to 100 Mbps?

(a) 25000 m (b) 2500 m
(c) 250 m (d) 25 m

35. Consider an IP in CIDR notation as 220.19.18.87/24. The first and the last address of this network will be?

(a) 220.19.18.24 and 220.19.18.87
(b) 220.19.18.0 and 220.19.18.87
(c) 220.19.18.0 and 220.19.18.123
(d) 220.19.18.0 and 220.19.18.255

36. In ICMP, 8 byte of the IP datagram is included in ICMP data part because:

(a) It contains the IP address.
(b) It contains the port numbers.
(c) The statement is false, IP datagram is not included in ICMP data part.
(d) It is helpful in checking the errors with CRC used.

37. A receiver receives the IP packet with the first 8 bits as 10100011. The length of the IP packet is

(a) 163 bytes (b) 1 byte
(c) 127 bytes (d) 12 bytes

38. The router performs at _____ layer(s).

(a) Only physical
(b) Physical, datalink and network
(c) Physical, datalink and transport
(d) Datalink and network

39. A packet has arrived with the offset value 100 and HLEN value 5. The total length value is 100. What is the number of first byte and the last byte?

(a) 879 (b) 880
(c) 881 (d) 882

40. Which of the following is true?

(a) In TCP/IP-based services, the destination address is to be specified only during the initial stage of setup.
(b) Initial setup is required for UDP-based service.
(c) Packet sequencing is not guaranteed in TCP/IP-based services.
(d) Initial setup is not possible in UDP-based service.

41. How many characters (7 bits + 1 parity) can be transmitted over 2400 bps line, if the transfer is asynchronous (1 start and 1 stop)?

(a) 2400/8 (b) 2400/10
(c) 2400 × 8/2 (d) 256

42. In CRC, if the data unit is 110111001 and the divisor is 1011 then what is the dividend at the receiver?

(a) 110111001101
(b) 110111001000
(c) 110111001011
(d) 101110111001

43. With respect to ICMP, which of the following statements is false?

(a) ICMP reports about the routers.
(b) ICMP is also used to test connectivity.
(c) ICMP and BOOTP are equivalent in their usage.
(d) An ICMP message type is encapsulated for transmission.

44. A 3000-lm long trunk is used to transmit frames using Go-Back-N protocol. The propagation speed is 6 μs/km and the trunk data rate is 1.544 Mbps. We ignore the time it takes to receive the acknowledgement bits. Frame size is 64 bytes. What should be the maximum window size at the sender's side in order to achieve 100% efficiency?

(a) 3000 (b) 1544
(c) 110 (d) 64

45. In a pure ALOHA network, 100 stations share a link of 1 Mbps. The throughput of such a network having 1000 bits size of frames and each station is transmitting at the rate of 10 frames/s is

(a) 10% (b) 13.53%
(c) 12.8% (d) 18.16%

46. An HDLC does not support

(a) simplex
(b) multipoint link
(c) balanced multipoint
(d) full duplex

47. A hub in the network is

(a) a passive device
(b) an active device
(c) a server that serves every node
(d) a power supply concentrator

48. Consider the following message: $M = 1010001101$. The CRC of this message using the divisor polynomial $x^5 + x^4 + x^2 + 1$ is

(a) 1110 (b) 0101
(c) 1001 (d) 0010

49. The broadcast address for the subnet that IP address 192.168.26.8, 255.255.255.248 is a member of

(a) 192.168.26.225 (b) 192.168.26.255
(c) 192.168.127.255 (d) 192.168.26.0

50. A sender uses public key cryptography to send a secret message. Which of the following is true?

(a) Sender encrypts using receiver's public key
(b) Sender encrypts using his own public key
(c) Receiver decrypts using sender's private key
(d) Receiver decrypts using own private key

51. Which of the following functionality must be implemented by transport layer over and above the network protocol?

(a) Recovery from packet loss
(b) End to end connectivity
(c) Packet delivery in encapsulated abstract order
(d) Secure transmission from end to end

52. During the process of packet forwarding by the routers, _____ field is not updated.

(a) IP header source address
(b) IP header target address
(c) IP header TTL
(d) IP header checksum

53. Consider the network, using CIDR addressing, a PC having the IP address as 180.10.10.10/20. The numbers of addresses in such a network will be _____ ?

(a) 255 (b) 4094 (c) 4096 (d) 1024

Set 2

1. Consider the following two cases given below, where G denotes the generator polynomial and M denotes the received message:

Case 1: $G = 1100$, $M = 1010101$
Case 2: $G = 11001$, $M = 111000111011$
In which transmissions errors occur?

(a) 1 only (b) 2 only
(c) Both 1 and 2 (d) Neither 1 nor 2

2. A block of addresses is granted to a small organisation. We have given a random address 216.18.9.21/28.

What is the first address in the block?

(a) 216.18.9.0 (b) 216.18.9.16
(c) 216.18.9.18 (d) 216.18.9.21

3. How many total numbers of addresses are there in the block using the information in the above question?

(a) 16 (b) 32 (c) 64 (d) 128

4. A channel has 10 Kbps bit rate using stop-and-wait protocol and has propagation delay of 5 ms. For a frame with 360 bit, what will be the efficiency?

(a) 92% (b) 88%
(c) 84% (d) 78%

5. Consider a 3 Mbps token ring LAN and frame size is of 200 byte. If the ring latency is 150 μs, then what will be the effective data rate of the LAN?

(a) 2.10 Mbps (b) 2.25 Mbps
(c) 2.35 Mbps (d) 2.90 Mbps

6. Host A is sending data to host B over a full duplex link. A and B are using the sliding window protocol for flow control. The send and receive window sizes are 6 packets each. Data packets (sent only from A to B) are all 1000 bytes long and the transmission time for such a packet is 40 μs. Acknowledgement packets (sent only from B to A) are very small and require negligible transmission time. The propagation delay over the link is 180 μs. What is the maximum achievable throughput in this communication?

(a) 14.28×10^6 bps (b) 14.61×10^6 bps
(c) 19.33×10^6 bps (d) 23.40×10^6 bps

7. Station A uses 32 bytes packets to transmit messages to station B using a sliding window protocol. The round-trip delay between A and B is 60 ms and the bottleneck bandwidth on the path between A and B is between 128 Kbps. What is the optimal window size that A should use?

(a) 20 (b) 30
(c) 60 (d) 40

8. Frames of 1000 bits are sent over a 10^6 bps duplex link between the two hosts. The propagation time is 34 ms. Frames are to be transmitted into this link to maximally pack them in transit (within the link). What is the minimum number of bits (l) that will be required to represent the sequence of numbers distinctly? Assume that no time gap needs to be given between transmissions of two frames.

(a) $l = 4$ (b) $l = 5$
(c) $l = 6$ (d) $l = 7$

9. A 3-km long broadcast LAN has 10^7 bps bandwidth and uses CSMA/CD. The signal travels along the wire at 4×10^8 m/s. What is the minimum packet size that can be used on this network?

(a) 50 bytes (b) 100 bytes
(c) 200 bytes (d) None of these

10. Match the following:

(A) 204.26.37.47	i. Class A
(B) 145.12.13.0	ii. Class B
(C) 230.54.256.56	iii. Class C
(D) 255.255.255.255	iv. Class D
(E) 126.126.126.126	v. Class E
(F) 85.18.257.24	vi. Invalid IP

(a) A-iv	B-ii	C-iv	D-v	E-i	F-i
(b) A-iii	B-ii	C-vi	D-v	E-i	F-vi
(c) A-iii	B-ii	C-iv	D-iv	E-vi	F-i
(d) A-iii	B-ii	C-iv	D-vi	E-i	F-vi

11. In a class C network if subnet mask is given as 255.255.255.224. Calculate the number of subnet and number of host in each subnet (practically possible).

(a) 4 and 32 (b) 6 and 64
(c) 8 and 32 (d) 6 and 30

12. We need to make a super network out of 16 class C block, what is the supernet mask?

(a) 255.255.255.240
(b) 255.240.0.0
(c) 255.255.240.0
(d) 255.255.224.0

13. Given that bandwidth = 10 Mbps, distance = 4 km and velocity = 2×10^8 m/s. The number of bits are transmitted in RTT is _____.

14. Suppose the round-trip propagation delay for a 8 Mbps Ethernet having 48-bit jamming signal is 50 μs. The minimum frame size is _____.

15. The address of a class B host is to be split into subnets with a 7-bit subnet number. What is the maximum number of subnets and the maximum number of hosts in each subnet?

(a) 128 subnets and 1024 hosts
(b) 128 subnets and 1022 hosts
(c) 126 subnets and 512 hosts
(d) 126 subnets and 510 hosts

16. If a class B network on the Internet has a subnet mask of 255.255.248.0, what is the maximum number of hosts per subnet?

(a) 1022 (b) 1023
(c) 2046 (d) 2047

17. In a network,propagation delay is 100 μs and bandwidth is given as 100 Mbps. Calculate how many RTTs are required for transmitting 40000 bits in stop-and-wait ARQ.

(a) 2 (b) 4 (c) 7 (d) 9

18. If 5-bit sequence number is used, what is the sequence number after sending 100 frames in Go-Back-N ARQ?

(a) 2 (b) 3 (c) 4 (d) 5

19. If N is a maximum sequence number in sliding window of Go-Back-N ARQ. How many sequence bit will be there for use?

(a) $\log_2 N$ (b) $1/\log_2 N$
(c) $\log_2 N^2$ (d) $\log_2 N + 1$

20. If 6 bits are used for sequence number, then what is the sender window size and receiver window size in selective repeat scheme?

(a) 64 (b) 63
(c) 32 (d) 31

21. If 10 Base 5 cable is used and velocity is 2×10^8 m/s. What will be the frame size in CSMA/CD?

(a) 40 (b) 50
(c) 60 (d) 64

22. If the capacity of a router is 1.2 mb, output rate of data is .8 mb/s token that are generated is 6 mb/s. Calculate the time matrix busted traffic is routed.

(a) 0.6 s (b) 1.2 s
(c) 1.5 s (d) 2.0 s

23. If the total length bits are 0000000000111111 and HLEN = 1010, then which one is true about the size of header and payload?

(a) 40 and 63
(b) 40 and 23
(c) 24 and 63
(d) 24 and 23

24. In ICMP protocol, error reporting message type 11 denotes which of the message type?

(a) Destination unreachable
(b) Source quench
(c) Time exceeded
(d) Parameter problem

25. In a token ring network, the transmission speed is 32 bps and the propagation speed is 200 m/μs. The 4-bit delay in this network is equivalent to:

(a) 12 m of cable (b) 18 m of cable
(c) 22 m of cable (d) 25 m of cable

ANSWERS TO PRACTICE EXERCISES

Set 1

1. (d)	**8.** (b)	**15.** (c)	**22.** (a)	**29.** (c)	**36.** (b)	**43.** (c)	**50.** (a)
2. (b)	**9.** (16)	**16.** (b)	**23.** (a)	**30.** (c)	**37.** (d)	**44.** (c)	**51.** (b)
3. (c)	**10.** (a)	**17.** (a)	**24.** (d)	**31.** (a)	**38.** (b)	**45.** (b)	**52.** (b)
4. (d)	**11.** (a)	**18.** (d)	**25.** (c)	**32.** (a)	**39.** (a)	**46.** (c)	**53.** (b)
5. (c)	**12.** (b)	**19.** (a)	**26.** (b)	**33.** (a)	**40.** (d)	**47.** (a)	
6. (a)	**13.** (b)	**20.** (c)	**27.** (a)	**34.** (c)	**41.** (b)	**48.** (d)	
7. (a)	**14.** (4)	**21.** (c)	**28.** (a)	**35.** (d)	**42.** (c)	**49.** (b)	

Set 2

1. (a) In case 1: when M (1010101) by G (1100), we will get remainder which do not have all zeros.

In case 2: when M (111000111011) by G (11001), we will get all zeros in remainder.

2. (b) One network address: 216.18.9.21

Subnet mask: 255.255.255.240
By doing XOR between network id and subnet mask, we get first address of block = 216.18.9.16.

3. (a) Total number of addresses $= 2^{32-28} = 2^4 = 16$

4. (d) Transmission time $= \dfrac{\text{Data size}}{\text{Bandwidth}} = \dfrac{360}{10000} = 36$ ms

Utilization in percentage

$$= \frac{\text{Transmission time}}{\text{Transmission time} + (2 \times \text{Propagation time})} \times 100$$

$$= \frac{36}{36+10} \times 100 = 78.2 \approx 78\%$$

5. (c) Transmission time $= \dfrac{\text{Data size}}{\text{Bandwidth}} = \dfrac{200 \times 8}{3 \times 10^6}$
$= 0.00053\,\text{s}$

Ring latency = Round-trip time = 150 μs = 0.00015 s
Total time required to transfer 1600 bits = (0.00053 + 0.00015) = 0.00068 s
So, effective data rate = 1600/0.00068 = 2.35 Mbps

6. (a) We have

Window size (n) = 6 packets
Packet size = 1000 bytes
So, total packet size = 6 × 1000 = 6000 bytes
Total time = Transmission time + Propagation time
$= 6 \times 40 + 180\ \mu\text{s} = 420\ \mu\text{s} = 420 \times 10^{-6}\text{s}$

Maximum achievable throughput $= \dfrac{\text{Total size}}{\text{Total time}}$

$$= \frac{6000}{420 \times 10^{-6}} = \frac{6000 \times 10^6}{420}$$
$$= 14.28 \times 10^6 \text{ bps}$$

7. (b) Given round-trip delay (T) = 60 ms = 60×10^{-3}s

$R = 128$ Kbps $= 128 \times 10^3$ bps
$L = R \times T = 128 \times 10^3 \times 60 \times 10^{-3} = 128 \times 60$

So, optional window size (n) $= \dfrac{128 \times 60}{32 \times 8}$

$$= \frac{7680}{256} = 30$$

8. (c) Because of duplex link we need not wait for twice the propagation time for sending the frame. If the sender window size is N, then

Transmitting 10^6 bits require = 1 s
Therefore, $N \times 10^{-3}$ s = N ms
Therefore, N ms = 34 ms, $N = 34 < 2^6$
So, minimum number of bits required is 6.

9. (d) In CSMA/CD, the minimum frame size = 2 × z × bandwidth

Given that distance (d) = 3 km = 3×10^3 m, velocity (v) = 4×10^8 m/s and bandwidth = 10^7 bps.
Therefore, minimum pocket size

$$= 2 \times (7.5 \times 10^{-6}) \times 10^7 = 2 \times 7.5 \times 10$$
$$= 150 \text{ bits} = 150/8 \approx 19 \text{ bytes}$$

10. (b) Number greater than 255 is not allowed in valid IP. In option (f), 3rd octave has 257 value which is not valid. So, option (f) should match (vi). Similarly, option (c) should match (vi).

11. (d) Subnetmask 255.255.255.224 – 11111111. 11111111. 11111111. 11100000

Number of subnet $= 2^3 - 2 = 6$
Number of host in each subnet $= 2^5 - 2 = 30$
Two addresses are reduced because one is for network address and another is for broadcasting.

12. (c) Default mask of

class C = 255.	255.	255.	0
11111111	11111111	11111111	00000000

Required 16 class C blocks, so

11111111	11111111	1111<u>0000</u>	00000000

We have used last 4 bit for combining 16 class C blocks. So supernet mask is {255.255.240.0}

13. (400) RTT (round-trip time) = 2× PT (propagation time)

$$\text{Propagation time (PT)} = \frac{\text{Distance}}{\text{Velocity}} = \frac{4 \times 10^3}{2 \times 10^8}$$
$$= 2 \times 10^{-5}\text{s} = 20\ \mu\text{s}$$

RTT = 2 × PT = 2 × 20 = 40 μs
For 1 s, bits transmitted = 10^7 bits
So, for 40 μs, the buts transmitted = $40 \times 10^{-6} \times 10^7 = 400$ bits

14. (400) Frame size = $(2T_P)$ × (Bandwidth) = 50 ms × 8 Mbps = 400 Kbits

15. (d) The Class B is defined as follows

Network id = 16 bit
Host id = 16 bit

Maximum number of subnets
$= 2^7 - 2 = 128 - 2 = 126$
Maximum number of hosts in each subnet
$= 2^{16} - 7 - 2 = 2^9 - 2 = 510$

16. (c) Generally, the number of addresses usable for addressing specific hosts in each network is always $2^N - 2$, where N is the number of bits for host id.

The binary representation of subnet mask is 11111 111.11111111.11111000.00000000.

There are 21 bits set in subnet. So, 11 (i.e., 32 − 21) bits are left for host ids.

Total possible values of host ids is $2^{11} = 2048$.

Out of 2048 values, 2 addresses are reserved.

The address with all bits as 1 is reserved as broadcast address and address with all host id bits as 0 is used as network address of subnet.

17. (a) Given that bandwidth = 100 Mbps = 100 × $10^6 = 10^8$ bits and propagation time = 100 μs

We know that

$$\text{RTT} = 2 \times \text{PT} = 200\ \mu\text{s}$$
$$\text{Data transfer in 1 s} = 10^8$$

So, data transfer in 200 μs (in 1 RTT) = $200 \times 10^{-6} \times 10^8 = 20000$ bits
Required RTT for transmitting 40000 bits = 40000/20000 = 2

18. (b) Sequence bits (n) = 5
Maximum value of sequence number = $2^n - 1$

0	1	2	3	-	30	31	0	1	2	-	30	31	0	1	2	-	30	31	0	1	2	3
				-						-						-						
				-						-						-						
				-						-						-						

(0−30) = 31 frames
(31−29) = 31 frames
(30−28) = 31 frames
Total = 93 frames

Total frames left 6 and they will be counted as 29, 30, 31, 0, 1, 2.

So, after 99th frame, sequence number will be 3.

19. (d) If sequence bits are N, then maximum sequence number is = $2^N - 1$, so

Number of sequence bits = $\log_2 N + 1$

20. (c) If sequence bits are n, then window size = 2^{n-1}

So, for 6-bit sequence number, window size = $2^{6-1} = 2^5 = 32$

21. (b) Given that distance = 500 m and bandwidth = 10 Mbps

Frame transmission time = 2 × Propagation delay
Therefore,

$$\frac{\text{Frame size}}{\text{Bandwidth}} = 2 \times \frac{\text{Distance}}{\text{Velocity}}$$

$$\frac{\text{Frame size}}{10 \times 10^6\ \text{b/s}} = 2 \times \frac{500\ \text{m}}{2 \times 10^8\ \text{m/s}}$$

Frame size = 50 bits

22. (a) Formula to calculate time = $C + \rho S = MS$

where capacity (C) = 1.2×10^6, token generation rate (ρ) = 6 mb/s and output rate (M) = 8×10^6. Putting values into formula, we get

$$(1.2 \times 10^6) + 6 \times 10^6 \times S = (8 \times 10^6)\ S$$
$$10^6(1.2 + 6S) = (8 \times 10^6)\ S$$
$$S = \frac{1.2}{2} = 0.6\ \text{s}$$

23. (b) Packet size = $(0000000000111111)_2 = 63$

Header size = HLEN × 4 = 1010 × 4 = $(1010)_2 \times 4 = 40$
Packet size = Header + Payload
63 = 40 + Payload ⇒ Payload = 23

24. (c) Error message types

3→ destinations unreachable
4→ source quench
11→ time exceeded
12→ parameter problem

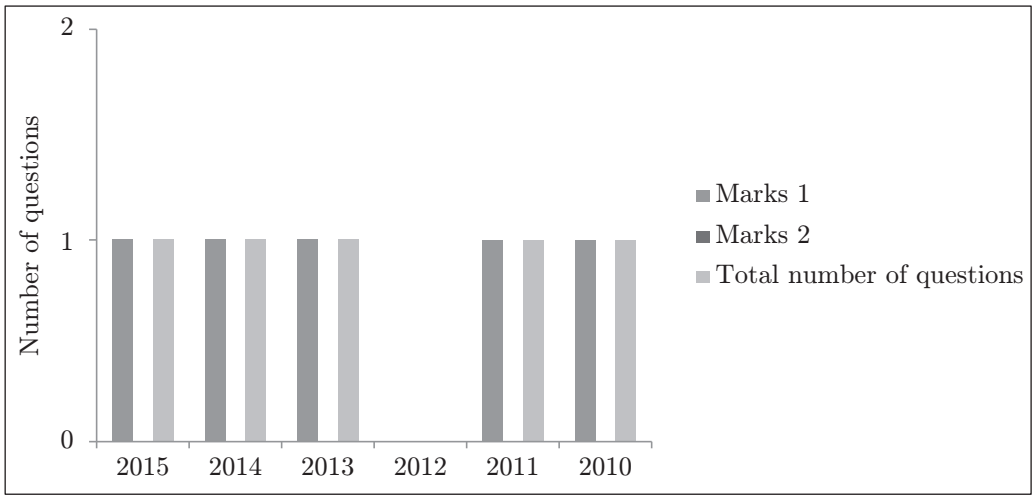
25. (d)

Given that $R = 32 \times 10^6$ bps, $B = 4$, $V = 200$ m/μs; $d = ?$

$$d = \frac{Bv}{R} = \frac{4 \times 200}{32 \times 10^6 \times 10^{-6}} = 25\ \text{m of cable}$$

UNIT XI: WEB TECHNOLOGIES

LAST SIX YEARS' GATE ANALYSIS



Concepts on which questions were asked in the previous six years

Year	Concept
2015	Unix Sockets, XML and HTML
2014	HTML Web page and http server
2013	Communication technology
2012	Nil
2011	HTML elements
2010	Client–server application

CHAPTER 11

WEB TECHNOLOGIES

Syllabus: Web technologies: HTML, XML, basic concepts of client–server computing.

11.1 INTRODUCTION

To communicate over the Internet, Web servers and Web browsers act as client−server programs. Web data is distributed over Internet using these programs. HTML is a markup language that is used for presentation and interaction with people in Web browsers. Web servers use IP address and port numbers for identification. Using Web browser, data request is sent using HTTP protocol. Apache, Tomcat and IIS are popular Web-server programs, whereas Internet Explorer (IE), Chrome, Safari, Opera and Firefox are popular Web browsers.

The World Wide Web (WWW), also termed as 'web', is an information medium which is accessed through the Internet. The following table describes the major milestones in the development of the WWW (Table 11.1).

Table 11.1 | Major milestones in development of Web

1980	Tim Berners-Lee invents the WWW, at CERN (the world famous nuclear research lab at Switzerland).
1990	Concepts like − HTTP, Web browser, Uniform Resource Identifier (URI) and HTML
1993	Launch of the first graphical web browser, named − MOSAIC at USA.
1994	Hosting of the first International WWW Conference, formation of W3C
1996	Commercialization of the Web
1998	Google was founded by Larry Page and Sergey Brin
1999	Concept of Dot-com, its boom and bust.

(*Continued*)

Table 11.1 | Continued

2002	Launch of Web 2.0
2004	Launch of FaceBook and use of Internet for social networking
Future	Semantic Web (Web 3.0)

11.2 HTML

HTML stands for hypertext markup language, widely used for the development of Web pages. It is used to display text, images and other objects in a specified way on the browser. Hypertext signifies the technique of linking HTML documents together. The link available on a Web page is called hypertext. Markup language signifies the marking of a text document with tags, which are used to display on Web browsers. It was designed in 1991 by Berners-Lee, and later, it was published in 1995 with standard specifications. The latest version of HTML-5 was published in 2012. World Wide Web Consortium (W3C) defines the specifications of HTML. HTML 4.0 was published in 1997 and then revised in 1998. HTML 4.01 was published in 1999.

HTML files are written in Notepad, a text editor on Windows, with a file extension. html. The output can be shown by opening it using a Web browser.

11.2.1 HTML Tags

HTML tags are used to mark up HTML elements and are surrounded by two characters '<' and '>' known as angle brackets (see Table 11.2). HTML tags generally comes in pair such as and . The first tag in a pair is the opening tag and the second tag is the ending tag. The text between the start and end tags is the element content. These are not case-sensitive, which means and <B> are the same.

A typical HTML document has the following structure:

```
<html>

<head>

                Document header related tags

</head>

<body>

                 Document body related tags

</body>

</html>
```

Table 11.2 | HTML tags

Tag	Description
<html>	Defines an HTML document
<head>	Defines the document's header, which can keep other HTML tags such as <title>, <link>, etc.
<title>	Defines the document title
<body>	Defines the document's body
<h1> – <h6>	Defines headings 1 to 6
<p>	Defines a paragraph
 	Inserts a single line break
	Defines a centering content
<hr>	Defines a horizontal rule
<pre>	Preserves formatting
<!-->	Defines a comment
<samp>	Defines a sample computer code
<var>	Defines a variable

11.2.2 HTML Attributes

Attributes are extra bits of information in tags. These are used to define the characteristics of an HTML element. These show inside the opening tag and their values are always inside quotation marks and consist of two parts: name is the property and value is the value of the property to be set. For example, `<body bgcolor="blue">`, bgcolor is the name and blue is the value.

11.2.2.1 Core Attributes

There are four core attributes used in HTML: id, title, class and style.

1. The id attribute can be used to uniquely identify any element within an HTML page. For example, `<p id="html">` This paragraph explains the definition of HTML.
2. The title attribute is used for the title of the element. The syntax of title attribute is same as that of the id attribute.
3. The class attribute is used to associate an element with a style sheet, and specifies the class of the element.
4. The style attribute is used to specify Cascading Style Sheet (CSS) rules within the element.

11.2.2.2 Generic Attributes

Attribute	Value	Description
align	Right, left, centre	Horizontally aligns tags
bgcolor	Numeric, hexadecimal, RGB values	Places a background colour behind an element
background	URL	Places a background image behind an element
class	User defined	Classifies an element for use with Cascading Style Sheets
height	Numeric value	Specifies the height of tables, images or table cells
id	User defined	Names an element for use with Cascading Style Sheets
title	User defined	Pop-up title of the elements
valign	Top, middle, bottom	Vertically aligns tags within an HTML element
width	Numeric value	Specifies the width of tables, images or table cells

11.2.3 HTML Elements

Elements are bits of information that make up Web pages. These reside in the tag pairs, for example, "My first Web page" and "This is my first Web page" are HTML elements.

```
<html>
   <head>
      <title>My first Web page</title>
   </head >
      <body>This is my first Web page</body>
</html>
```

11.2.4 HTML Character Entities

In HTML, some characters have special meaning; if we want to display such characters, character entities must be used in place of actual characters. A character entity has three parts:

ampersand (&), entity name or an entity number and semicolon (;).

`∧` specifies the beginning and ending of a special character, whereas Entity name is an abbreviation of that character.

For example, '<' is a beginning tag in HTML. To display this character we have to write '`<`'. There are some more examples listed in Table 11.3.

Table 11.3 | HTML Character Entities and their Description

Character	Character Entity	Description
	` `	Non-breaking space
<	`<`	Less than
>	`>`	Greater than
&	`&`	Ampersand
"	`"`	Quotation mark
'	`'`	Apostrophe

11.2.5 HTML Formatting

HTML formatting tags are basically designed to display special type of text. The tags will provide some additional feature than the normal text. Some of the formatting tags used in HTML are shown in Table 11.4.

Table 11.4 | HTML Formatting tags and their description

Tag	Description
<b>	Bold text
<big>	Larger text
<del>	Deleted text
	Grouping content
<i>	Italic text
<ins>	Inserted text
<pre>	Preformatted text
<small>	Smaller text
	Grouping content

(*Continued*)

Table 11.4 | Continued

Tag	Description
<strike>	Strike text
<sub>	Subscript text
<sup>	Superscript text
<tt>	Teletype (monospaced) text
<u>	Underlined text

Note: The <font> tag in HTML is deprecated

11.2.6 HTML Phrase Tags

The purpose of designing phrase tags is to add structural information to a text fragment. They are displayed in a similar way as other basic tags like <b>, <i>, <pre>, and <tt>, etc. The phrase tags used in HTML and their description is given in Table 11.5.

Table 11.5 | HTML phrase tags and their description

Tag	Description
<abbr>	Text abbreviation
<acronym>	Acronym element
<address>	Address text
<bdo>	Text direction
<blockquote>	A long quotation
<cite>	Text citation
<code>	Computer code text
<dfn>	HTML definition element, introducing a special term
	Emphasize text
<kbd>	Keyboard text
<mark>	Marked text
<q>	A short quotation
	Strong text
<small>	Smaller text

Programming variables are used when the content of the element is a variable; in most cases, the <pre> and <code> tags are also used alongwith it.

Program output is used while documenting line(s) of code, also used to check the output of the code or script.

11.2.7 HTML Background

The background can be a colour or an image and specified in the <body> tag.

11.2.7.1 Bgcolor

Bgcolor is an attribute and specifies a background colour for an HTML page. The value of this attribute can be a hexadecimal number, an RGB value or a colour name.

11.2.7.2 Background

It specifies background image for an HTML page. The value of this attribute is the uniform resource locator (URL) of the image.

11.2.8 HTML Lists

HTML lists are of three types: unordered lists, ordered lists and definition lists.

1. Unordered lists are used for non-sequential lists (bullets). <ul> … </ul> tag is used to define unordered lists and <li> tag is used to define each list item.
2. Ordered lists are used for sequential lists (incremental numbers). <ol> … </ol> tag is used to define ordered lists.
3. Definition list consists of two parts: a termand a description. <dl> tag is used to define a definition list; <dt> is used for definition term and <dd> is used for definition description.

11.2.9 HTML Links

An anchor tag <a> … </a> is used to define a link, the destination is also included in the anchor tag <a>, for example, `<a href="http://www.html.com"> HTML </a>`, href is an attribute.

11.2.10 HTML Images

<img> tag is used to display image. It does not have a closing tag.

1. **src attribute:** It has an src attribute, which contains URL of the image. For example, `<imgsrc="html.gif">`.
2. **alt attribute:** It also uses alt attribute to define an alternate text for an image.
3. **Image dimensions:** Width and height of an image can be used with <width> and <height> attributes.
4. **border attribute:** This attribute is used to specify border thickness in terms of pixels.

11.2.11 HTML Tables

The HTML tables are used to arrange data into rows and columns of cells. These are created using the <**table**> tag.

<**tr**> tag is used to create table rows and <**td**> tag is used to create data cells. <**th**> tag is used for table heading. Cellpadding and cellspacing attributes are used to adjust the white space in the table cells. Cellpadding defines the distance between cell borders and the content within a cell and cellspacing defines the width of the border. Colspan and rowspan attributes are used to merge two or more columns into a single column and row, respectively. For table backgrounds, bgcolor and background attributes are used. The height and width of the table can be changed with the height and width attributes, respectively. Caption tag is used for the caption of the table. <thead>, <tbody> and <tfoot> are used for creating portions of HTML table into header, body and footer, respectively.

11.2.12 HTML Frames

To divide the browser window into multiple sections, HTML frames are used. A separate HTML document can be loaded in each section. A frameset is simply a collection of such kind of frames in a browser window.

<frameset> tag is used to create frames in the browser window. The rows attribute is used for horizontal frames and cols attribute is used for vertical frames. The value of rows and cols is represented in percentage, for example, `<frameset cols="25%,50%,25%">`.

Each frame is indicated by the <frame> tag. The <frame> tag has mainly two attributes: name and src. For example, `<frame name="top" src="/html/top_frame.htm"/>`.

Other attributes of a frame are border, frameborder and framespacing. Border defines the width of the border of each frame in pixels. Frameborder takes value either 1 (yes) or 0 (no) for 3D border display. The framespacing defines the amount of space between frames in a frameset.

The <frame> tag has the following attributes: src, name, frameborder, marginwidth, marginheight, noresize, scrolling and longdesc.

The <iframe> tag is used to define a rectangular region, that can display another document having scrollbars and borders.

11.2.13 HTML Forms

These are used to collect data from the site visitor. There are various form elements: text fields, drop-down menus, combo-box, radio buttons, checkboxes, etc.

The <form> tag has the following attributes: action, method, target and enctype.

1. Action attribute is used for backend script, which is ready to process passed data.
2. Method attribute is used to upload data.
3. Target defines the target window, where the result of the script is displayed, and
4. Enctype defines the encoding of the data by the browser.

11.2.14 HTML Marquees

HTML marquee is a scrolling piece of text displayed on the Web page. The <marquee> tag is used for this.

11.3 CASCADING STYLE SHEETS

Cascading Style Sheets (CSS) is another method of styling content. It is a style sheet language which describes the appearance and format of the document written in any markup language. Generally, it is used to modify the style of web pages and user interfaces written in HTML and XHTML, plain XML, XUL, etc. Along with HTML and JavaScript, the CSS is a technology used by most websites to create attractive webpages, visually appealing user interfaces for web and mobile applications.

CSS is designed primarily to distinguish document content from document presentation, using elements such as the layout, colours, and fonts.

11.4 XML

XML stands for eXtensible Markup Language. It is an open standard to share data and information between computers and computer programs. It is a meta-markup language, which defines the syntax to define other semantic, domain-specific and structured markup languages. It also describes the structure and meaning of a document. It is a set of rules for defining semantic tags. XML documents always have a single root element. The element names are case-sensitive, always closed and correctly nested. Attributes of element are always quoted. The default entities defined are <, >, &, “, and ’. An XML document has a tree-like structure.

Syntax for an XML:

```
<Element Name>

<Element Name>Content</Element Name>

<Element Name>Content</Element Name>

<Empty Element Name/>
```

Example 11.1

<html> is a root element, <head> and <body> are two branches.

```
<html>

<head>

<title>XML</title></head>

<body>

This is XML</body>

</html>
```

Example 11.2

<team person1="Tom" person2="Dick">

```
<team>

<person1>Tom</person1>

<person2>Dick</person2>

</team>
```

Example 11.3

Example of an XML document:

```
<?xml version="1.0" encoding="UTF-8"?>

<greeting>Hello, world!</greeting>
```

1. **Namespaces:** An XML namespace is a collection of names identified by a universal resource identifier (URI) reference.
2. **XML semantics:** It is an articulation of XML elements that exist in a file, their relationship(s) to each other and their meaning.

11.4.1 Advantages of XML

1. Defining your own elements
2. Better organized documents
3. Sorting of database

11.4.2 Document-Type Definition

Document-type definition (DTD) defines the tags in the document; the tags contain other tags, the number and sequence of the tags, the attributes of tags and the values of those attributes.

11.4.2.1 Defining XML Vocabularies with DTDs

The semantic relationships are created by DTD and/or XML schemas. DTDs reside inside an XML document or outside an XML document. If it resides inside, it begins with a DOCTYPE declaration followed by the name of the XML document's root element and a list of all the elements and their relation to each other. If it resides outside an XML document, then the DOCTYPE declaration includes a pointer to a file where the XML elements are described.

11.4.2.2 Names and Numbers of Elements

Each element is qualified by the number of times it can occur in the XML document. This is done with the asterisk (*), question mark (?) and plus sign (+) symbols. Each symbol has a specific meaning:

1. **Asterisk (*):** The element may appear zero or more times.
2. **Question mark (?):** The element may appear zero or one time only.
3. **Plus sign (+):** The element appears at least once if not more times.

11.4.2.3 PCDATA

PCDATA stands for parsed character data. It is used to denote that content contains only text, text without markup.

11.4.2.4 Sequences

An element contains multiple sub-elements when bound together; the list of multiple elements is called a sequence. They can be grouped together as follows:

1. **Comma (,):** Denotes the expected order of the elements in the XML file.
2. **Parentheses (()):** Groups elements together.
3. **Vertical bar (|):** Denotes a Boolean union relationship between the elements.

11.4.3 Tag Rules for XML Documents

An XML document is a collection of XML elements. An XML element starts from (including) the element's start tag and ends at (including) the element's end tag. The tags in XML should follow the following restrictions:

1. XML tags cannot overlap.
2. Tags should be closed.
3. A tag that does not contain any text can contain the end marker at the end of the start tag.

11.4.4 Types of XML Documents

There are two ways to create an XML document:

1. **Well formed:** These documents follow the XML tag rules and do not follow the DTD. Document has a top-level element, all elements must have a starting and an ending tag, element names should be case-sensitive and elements must be nested properly.
2. **Valid:** A valid document must have followed the rules by DTD and should be well formed.

The comparison of HTML and XML is given in Table 11.6.

Table 11.6 | Comparison of HTML and XML

Attributes	HTML	XML
Internal Definition	Yes	Yes
External Definitions	No	Yes
Defined Tags	Yes	No
Add New Tags	No	Yes
Easily Add Objects	Yes	Yes
Easily Add Entities	Yes	Yes

Note: XML and HTML were designed with different goals:

1. XML was designed to describe data and to focus on what data is.
2. HTML was designed to display data and to focus on how data looks.

11.4.5 XHTML

It stands for eXtensible HyperText Markup Language. It was developed by W3C for the transition from HTML to XML. It is similar to HTML4.01, but it gives a consistent and well-organised format. It combines the features of both HTML and XML.

The XHTML document use lower case for all HTML elements and attributes. Certain tags in HTML are non-pair tags but XHTML requires end tags. All values must be quoted. XHTML define three DTD: Strict, Transitional and Frameset. XHTML 1.1 is a module based XHTML, which supports Ruby Annotation elements for the better understanding of East-Asian languages. It is compatible with HTML 4.0.

11.4.6 Document Object Model (DOM)

It is a World Wide Web Consortium (W3C) standard, used to access documents such as HTML and XML. It is a programming API. It defines the objects, properties and methods for accessing documents. It has three parts: Core DOM, XML DOM and HTML DOM.

11.4.7 XUL

Developed by Netscape and Mozilla, It stands for eXtensible User interface Language and pronounced as "Zool". It is a series of XML tags, which allows different operating platforms to exchange data or program. It supports Cascading Style Sheets, JavaScript, RDF, DOM and HTML.

11.4.8 Flash and Silverlight

Developed by Adobe Systems, Flash is a Rich Internet Application (RIA), used for playing audio and video in webpages. It is basically used for providing animation, interactivity to games, advertisements, etc. It is run on Microsoft Windows, Mac OS, QNX, Google TV and RIM.

Supported by a subset of. NET Framework, Silverlight is also a Rich Internet Application (RIA). In this, user interfaces are declared in eXtensible Application Markup Language (XAML). It supports Windows Media Video (WMV), Windows Media Audio (WMA), MPEG layer III (MP3).

11.4.9 User-Interface Language

These are used for graphical user interfaces and control. The objective of these interfaces is to use programs and script codes in the form of markup. Some of user interface markup languages are:

1. XUL (eXtensible User-interface Language) developed by Mozilla Foundation.
2. QML (Qt Meta Language) developed by Nokia, used for mobile applications such as touch input, fluid animation, etc. It is based on JavaScript declarative language.
3. UMIL (User Interface Markup Language) used to define user interfaces for computers. It is an XML based language.
4. UsiXML (User Interface XML) is a complaint markup language. It describes the user interface for multiple contexts of use.
5. XAL (eXtensible Application Language) developed by Nexaweb's Enterprise Web 2.0 suite, used for applications like Java client and Ajax Client.

Other user-interface languages are as follows: MXML, OpenLaszlo, ZUML, JavaFX, jInterface, Thinlet, Vexi, XHTML, XFDL, Xforms, XAML, XRC, EMML, GladeXML, SVG, etc.

11.5 BASIC CONCEPTS OF CLIENT–SERVER COMPUTING

Client–server consists of two parts: server, which provides services and client, which request services from the server (Fig. 11.1). The server in the network provides services to more than one client. There are different servers for different applications, such as file server, print server, mail server, etc. So, a client can request services from several servers on the network. Client and server can be on the same computer or on different computers linked by a network.

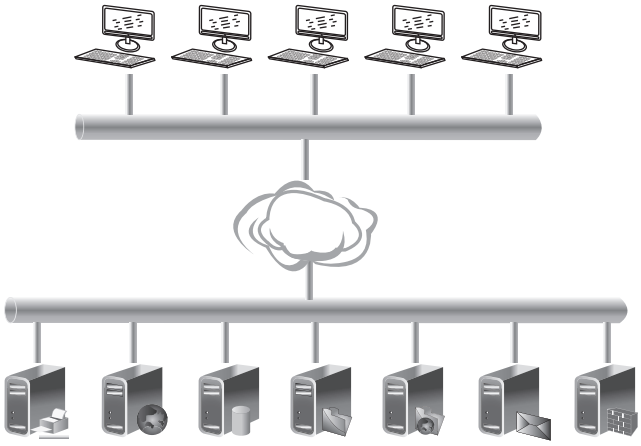


Figure 11.1 | Client–server model.

11.5.1 Server Types

For large and complex systems, the responsibility of the servers is distributed to several servers.

1. **File server:** All files reside in the file server. It is responsible for sharing files on the network. Example: Map Drive by Microsoft Windows, Google Drive, Network File Services (NFS) by Sun Micro Systems.
2. **Print server:** All the output devices, such as printer, are controlled by the print server. It is responsible for sharing printers on the network among clients.
3. **Application server:** It manages access to centralized application software. Example: shared database.
4. **Web server:** It manages Internet and/or Intranet. It shares documents. Thin clients use Web browser to request data and documents from servers. Example: Internet Information Server (IIS) of Microsoft Corporation, WebLogic and Oracle Application Server, etc.
5. **Mail server:** It manages the e-mails, messages, etc. Example: Gmail.
6. **Fax server:** It manages sending and receiving of fax.
7. **Database server:** It manages databases and responds to clients through SQL queries. It shares data in a database. Example: Oracle9i database server.
8. **Transaction servers:** It manages data and remote procedures. It shares data and high-level processing, such as OnLine Transaction Processing (OLTP), across a network. Example: Microsoft SQL Server, BEA Tuxedo, etc.

11.5.2 Stateless and Stateful Servers

1. **Stateless server:** It treats each request as an independent transaction. The request is not related to the previous request. It increases the overhead in each request, that is, extra information has to be included. Though, it simplifies the server design and does not affect the system if clients crash down. Example: Gopher protocol and Gopher+ protocol.
2. **Stateful server:** It does not treat each request as an independent transaction. Each request is related to the previous request. In stateful server, if clients crash frequently, state information may exhaust server's memory. Example: A remote file server.

11.5.3 Thin Client and Fat Servers

1. **Thin client:** It handles minimum processing on the client side.
2. **Fat client:** It handles large processing on the client side. It is used in traditional client–server models.
3. **Fat server:** It handles more functions on the server side. It is easy to manage than fat clients.

11.5.4 Functions of a Client

1. It manages user interface.
2. It accepts and checks the syntax of user inputs.
3. It generates database request and transmits them to the server.
4. It passes response back to the server.

11.5.5 Functions of a Server

1. It checks authorization for a client.
2. It ensures that a client does not violate integrity constraints.
3. It performs query/update processing and transmits responses to the client.
4. It maintains system catalogue.
5. It accepts and processes database requests from a client.

6. It provides concurrent database access.
7. It provides recovery control.

11.5.6 Topologies for Client–Server

Topology provides a physical connectivity of all clients and servers to each other. There are three types of topologies designed for client–server as follows:

1. **Single client single server:** In this topology, one client is directly connected to one server (Fig. 11.2). Example: One dedicated server connected to a single remote-sensing satellite.

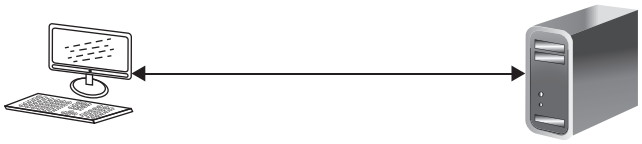


Figure 11.2 | Single client single server.

2. **Multiple clients single server:** In this topology, multiple clients are directly connected to one server (Fig. 11.3). Example: Different nodes connected to a server in a lab.

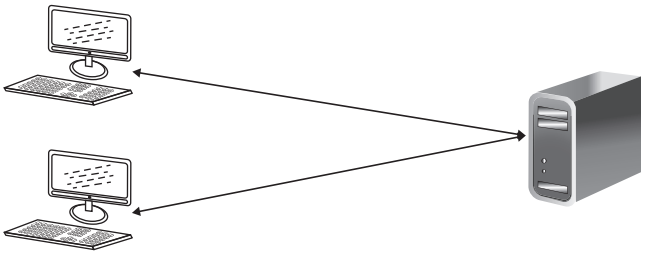


Figure 11.3 | Multiple clients single server.

3. **Multiple clients multiple servers:** In this topology, multiple clients are connected to multiple servers (Fig. 11.4). Example: People accessing facebook, Google and other applications.

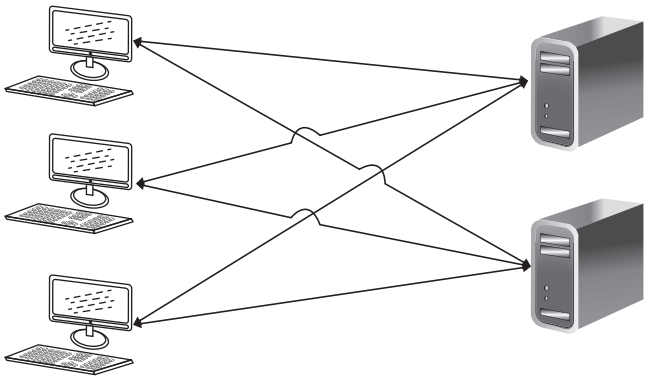


Figure 11.4 | Multiple clients multiple servers.

11.5.7 Types of Client–Server Model

There are three types of client–server models:

1. Two tier
2. Three tier
3. *N* tier

11.5.7.1 Two-Tier Architecture

In this, the business rules exist either at the client or at the server. The client is the first tier and the server is the second tier. Two-tier architecture at the client is called "fat client". The client uses a driver to translate the client's request into a database native library call.

Advantages

1. Client requests services directly from the server.
2. It is less complicated for implementation.
3. It provides attractive graphical user interface applications.
4. It has persistent connection between the client and the server.
5. It can be easily managed.

Disadvantages

1. Maintenance cost of application at client is high.
2. It increases load in the network.
3. Several PCs are required for individual applications.
4. Software distribution procedure is complicated in two-tier architecture.

11.5.7.2 Three-Tier Architecture

In this, business logic is located in the middle tier. Client's request is also handled by the middle tier. Only middle tier has to change if business rules are changed. The driver translates the request into a network protocol and makes a request via the proxy server. Application responsibilities in the three-tier architecture are of the following three types:

1. **Presentation (GUI) or user services:** It maintains the graphical user interface and generates output for the monitor. Presentation logic deals with screen formatting, windows management, input editing and what-if analysis.
2. **Application services or business rules:** It executes applications and controls the flow of the program. Business logic dealing with domain and range validation, data dependency validation and request/response architecture of inter-process communication level.
3. **Database services or data server:** It manages the databases. Server logic deals with data access, data management, data security and SQL parsing.

Advantages

1. Centralized application maintenance
2. Separate application logic from user interface control and data presentation
3. Easy and dynamic load balancing
4. Less expensive hardware (thin client)
5. Clients need not have native libraries loaded locally
6. Centralized drivers

Disadvantages

1. Client does not maintain a persistent database connection.
2. Separate proxy server is required. It increases network traffic.
3. Network protocol used by the driver may be proprietary.

11.5.7.3 N-Tier Architecture

In this, the developer designs components according to business rules and distributes the functionalities accordingly. It has better utilization and sharing of resources. The *N*-tier architecture provides finer-grained layers.

Advantages

1. Improved overall performance
2. Centralized business logic
3. More secure

11.5.8 Merits and Demerits of Client–Server

The client–server architecture facilitates with various features but it has some disadvantages also. The merits and demerits are discussed below:

1. **Merits:**
   - It increases the performance and reduces the workload on each system.
   - It gives workstation independence, which does not bind the users to a single operating system.
   - It allows different types of component systems such as client, network or server to work together.
   - It provides scalability in the network.
   - It preserves data integrity.
   - It provides data sharing, so data accessibility is good to users.
   - Due to centralized management of data, it becomes easy to secure and backup recovery of data by system administration.
   - It provides sharing of resources among diverse platforms.
   - It provides location independence of data processing.
   - The operating cost is less.
   - It has reduced hardware cost in terms of storage, printers and other output devices.
   - It has less communication cost.
2. **Demerits:**
   - Maintenance cost is high as it needs administrator. So, it also increases the training cost.
   - Hardware cost is increased in terms of high-powered platform with large amount of RAM and storage space.
   - It is more complex than PC.

11.6 J2EE PLATFORM

J2EE (Java2 Platform Enterprise Edition) is used to reduce cost involved in application design and development. It provides distributed multi-tiered application model, reusable components, web-service support and transaction control applications.

J2EE applications are made up of components. There are basically three types of components:

1. **Application clients and applets:** These components run on the client.
2. **Java Servlet and JavaServer Pages (JSP) technology components:** These are the web components that run on the server.
3. **Enterprise JavaBeans (EJB) components (enterprise beans):** These are the business components that run on the server.

11.6.1 J2EE Services

1. **HTTP API:** It is defined by the servlet and JSP interfaces.
2. **HTTPS:** It supports HTTP protocol over the SSL protocol.
3. **JDBC:** It is used for connectivity with relational database systems.
4. **JMS (Java Message Service):** It supports reliable point-to-point messaging as well as the publish-subscribe model.
5. **JTA(Java Transaction API):** It consists of two parts. (i) Application-level demarcation interface, which is used by the container and application components to demarcate transaction boundaries, and (ii) interface between the transaction manager and a resource manager used at the J2EE SPI level.
6. **JAF (Javabeans Activation Framework):** It is used for handling data in different MIME types, originating in different formats and locations.
7. **JNDI (Java Naming and Directory Interface):** It is used for naming and directory access.

8. **JAXP (Java API for XML Parsing):** It provides support for the industry standard SAX and DOM APIs for parsing XML documents. It also supports XSLT transform engines.
9. **RMI-IIOP:** It is a sub-system composed of APIs to use RMI-style programming. It supports both the J2SE native RMI protocol (JRMP) and the CORBA IIOP protocol.

11.6.2 J2EE Architecture

The Java2 SDK is an Enterprise Edition known as J2EE SDK. It is the reference implementation provided by Sun Microsystems, Inc. Figure 11.5 shows the major elements of the J2EE architecture.

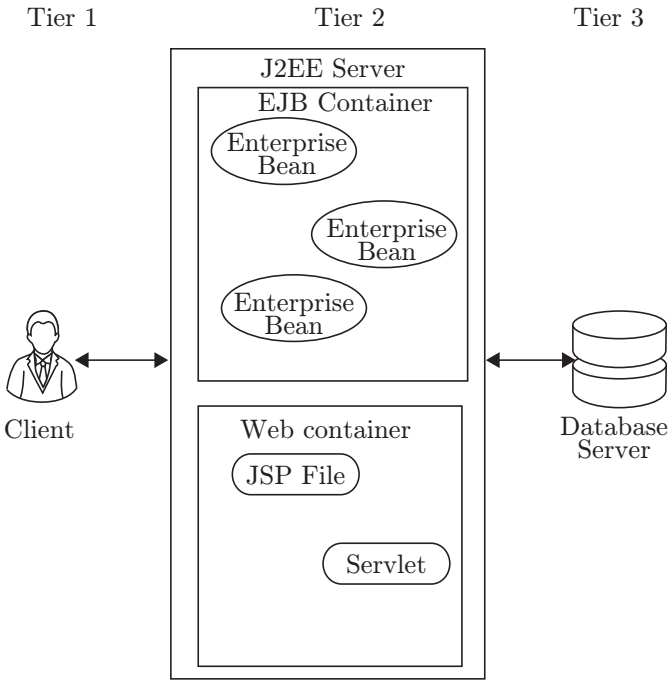


Figure 11.5 | J2EE architecture.

11.6.3 EJB Container

The container is a runtime environment, which controls the enterprise beans and provides them system-level services. Enterprise bean instances runs in an EJB container.

The container provides the following services to enterprise beans:

1. **Transaction Management:** The programmer need not to code for the transactions, rather, he has to declare the enterprise bean's transactional properties in the deployment descriptor file. Then, the container reads the file and handles the enterprise bean's transaction.
2. **Security:** Only authorized clients can invoke an enterprise bean's methods. Client has a particular role and each role is permitted to invoke certain methods.
3. **Remote Client Connectivity:** It manages the low level communication between clients and enterprise beans.
4. **Life Cycle Management:** It maintains all the states from the creation to the removal of enterprise beans.
5. **Database Connection Pooling:** The container manages the pool of database connections. The enterprise bean obtains a connection from the pool. These database connections can be re-used.
6. **Web Container:** It is a runtime environment for JSP files and servlets.

11.6.4 J2EE Application Server

It provides the following services:

1. **Naming and Directory:** It allows programs to locate services and components through the Java Naming and Directory Interface (JNDI) API.
2. **Authentication:** It enforces security by requiring users to log in.
3. **HTTP:** It enables web browsers to access servlets and JavaServer Page (JSP) files.
4. **EJB:** It allows clients to invoke methods on Enterprise Java Beans.

IMPORTANT FORMULAS

1. XML is case-sensitive.
2. XML is strict, so all the tags should be properly closed.
3. A tag that doesn't contain any text can contain the end marker at the end of the start tag.
4. Important tags in HTML:

<html>	Defines an HTML document
<head>	Defines the document's header, which can keep other HTML tags such as <title>, <link>, etc.
<title>	Defines the document title

<body>	Defines the document's body
<h1> – <h6>	Defines headings 1 to 6
<p>	Defines a paragraph
 	Inserts a single line break
	Defines a centering content
<hr>	Defines a horizontal rule
<pre>	Preserves formatting
<!-->	Defines a comment
<table>	Defines table in html
<tr>	Define rows in table
<td>	Defines columns in table

Important Abbreviations

API	Application Program Interface
ASP	Active Server Pages
CORBA	COmmon Resource Broker Achitecture
CSS	Cascading Style Sheets
DOM	Document Object Model
DTD	Document Type Definition
EJB	Enterprise Java Beans
HTML	HyperText Markup Language
HTTP/HTTPS	HyperText Transfer Protocol / Secure
J2EE	Java 2 Enterprise Edition
JAF	Javabeans Activation Framework
JMS	Java Message Service
JNDI	Java Naming and Directory Interface
JSP	Java Server Pages
JTA	Java Transaction API
MathML	Mathematical Markup Language
MIME	Multi-purpose Internet Mail Extensions
OLTP	OnLine Transaction Processing
RIA	Rich Internet Application
RSS	Rich Site Summary or Really Simple Syndication
SGML	Standard Generalized Markup Language
SVG	Scalable Vector Graphics
UIML	User Interface Markup Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WML	Wireless Markup Language
XAL	eXtensible Application Language
XAML	eXtensible Application Markup Language
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
XSL	eXtensible Stylesheet Language
XUL	XML User Interface Language

SOLVED EXAMPLES

1. The firewall is used for

(a) Sensing the size of the packet.
(b) Isolating intranet from extranet.
(c) Screening packets to/from the network and filtering of network traffic.
(d) Isolating Internet from virtual LAN.

Solution: Firewall is used to prevent unauthorized access of private networks through packet screening from outside user, especially Internet.

Ans. (c)

2. LDAP stands for

(a) lightweight digital audio protocol.
(b) lightweight directory access protocol.
(c) large domain access protocol.
(d) large data audio protocol.

Solution: The LDAP stands for Lightweight Directory Access Protocol which is a directory service protocol based on client-server model.

Ans. (b)

3. A digital signature is used to provide security makes use of

(a) Digitally scanned signatures.
(b) A unique ASCII code number of the sender.
(c) Private key encryption.
(d) Public key encryption.

Solution: Digital signature is an asymmetric cryptography technique which uses public key encryption mechanism to provide integerity, authentication and non-repudiation.

Ans. (d)

4. The <b> tag makes the enclosed text bold. The other alternative is to use

(a) (b) <dark>
(c) <big> (d) <highblack>

Solution:
 tag is used to define important text by bold the text.
<dark> — there is no tag in html named dark.
<big> tag is used to make text bigger than the normal text.
<highblack> — there is no tag in html named highblack.

Ans. (a)

5. What does HTML stand for?

(a) High tool and text markup language
(b) Heavy tool markup language
(c) Hypertext markup language
(d) Hypertext my language

Solution: HTML is a markup language which is used to design web pages. It stands for Hypertext Markup Language.

Ans. (a)

6. Who describes and controls the making of various Web standards?

(a) The World Wide Web Consortium
(b) IEEE and IETE
(c) Administrator and Mozilla
(d) IEEE

Solution: The World Wide Web Consortium (W3C) is the international standard organisation for managing and controlling web standard. IEEE and IETE are professional association societies for advancement of technology.

Ans. (a)

7. Which of the following HTML tag will provide the largest heading

(a) <h6> (b) <h1>
(c) <head8> (d) <heading6>

Solution: <h1> will provide largest heading, then <h2>, then <h3> and so on. <h6> gives the smallest heading.

Ans. (b)

8. The following HTML tag is used for inserting a line break?

(a)
 (b) <startbreak/>
(c) <lb/> (d) <LINEBRK/>

Solution:
 tag is used for inserting a line break. There is no tag in html with names<startbreak>, <lb/> and <LINEBRK>.

Ans. (a)

9. The correct syntax for adding yellow as a background colour in HTML is

(a) `<bodystyle="background-color:yellow">`
(b) `<backgroundcolor>yellow</background color>`
(c) `<color.background="yellow">`
(d) `<backgrndcolor="yellow">`

Solution: Syntax for background color is:

1. `<body bgcolor="color_name|hex_number|rgb_number">`

2. `<body style="background-color:color_name">`

Any of these two can be used.

Ans. (a)

10. The HTML tag used to make a text bold is

(a) <b> (b) <heavybold>
(c) <blackbold> (d) <bld>

Solution: <b> tag is used in HTML to make text bold.

Ans. (a)

11. The HTML tag used to make a text italic is

(a) <italic> (b) <i>
(c) <textitalic> (d) <slantingtext>

Solution: <i> tag is used in HTML to make text italic. There is no tag in html with names<italic>, <textitalic> and <slantingtext>.

Ans. (b)

12. The HTML tag used to create a hyperlink is

(a) `<ahref="http://www.myexams.com">MYEXAMS</a>`
(b) `<aurl="https://www.myexams.com">myexams</a>`
(c) `<anamelink="http://www.myexams.com">myexams.com</a>`
(d) `<a>http://www.MYEXAMS.COM</a>`

Solution: `<a href =" URL addrss" hyperlink name </a>` tag is used to create hyperlink in HTML.

Ans. (a)

13. Linking to another place in the same or another Web page require two A (Anchor) tags, the first with the _____ attribute and the second with the _____ attribute.

(a) NAME, LINK
(b) LINK, HREF
(c) HREF, NAME
(d) TARGET, VALUE

Solution: `<a href =" addrss"> hyperlink name </a>` tag is used to create hyperlink in HTML.

Ans. (c)

14. HTML is defined using SGML – an _____ standard, information processing-text and office systems (SGML) for text information processing.

(a) ISO – 8878 (b) ISO – 8879
(c) ISO – 8880 (d) ISO – 8881

Solution: SGML is an ISO standard (ISO — 8879) for defining markup languages.

Ans. (b)

15. The facilities available in the Internet are

A. Electronic mail.
B. Remote login.
C. File transfer.
D. Encryption technique.

(a) A and B (b) A, B and D
(c) A, B and C (d) B, C and D

Solution: Internet provides services like e-mails, file transfer and remote login; but it does not provide encryption and decryption techniques.

Ans. (c)

GATE PREVIOUS YEARS' QUESTIONS

1. Consider the HTML table definition given below:

```
<table border=1>
<tr><td rowsp an=2>ab</td>
<tdcolspan=2> cd </td></tr>
<tr><td>ef</td><td rowspan=2>gh</td>
</tr><tr><td colspan=2>ik</td></tr>
</table>
```

The number of rows in each column and the number of columns in each row are

(a) 2, 2, 3 and 2, 3, 2 (b) 2, 2, 3 and 2, 2, 3
(c) 2, 3, 2 and 2, 3, 2 (d) 2, 3, 2 and 2, 2, 3

(GATE 2009: 1 Mark)

Solution: Two td commands are used in the first tr command and three td commands are used in second tr command. So, it requires (2, 3, 2) rows in each column.

Ans. (c)

2. Which one of the following is not a client–server application?

(a) Internet chat (b) Web browsing
(c) E-mail (d) Ping

(GATE 2010: 1 Mark)

Solution: Internet chat, Web browsing and email are client–server applications. Ping is a connectivity checking of the network.

Ans. (d)

3. HTML (HyperText Markup Language) has language elements, which permit certain actions other than describing the structure of the Web document. Which one of the following actions is NOT supported by pure HTML (without any server or client-side scripting) pages?

(a) Embed Web objects from different sites into the same page.
(b) Refresh the page automatically after a specified interval.
(c) Automatically redirect to another page upon download.
(d) Display the client time as part of the page.

(GATE 2011: 1 Mark)

Solution: `<OBJECT> ... </OBJECT>` tag is used to embed web objects.
`<META HTTP-EQUIV="Refresh" CONTENT="5">` is used to refresh page after every 5 seconds.
`<META HTTP-EQUIV="Refresh" CONTENT="0; URL=another-page.html">` is used to redirect.
But for displaying the client time, there is no tag available.

Ans. (d)

4. Match the problem domains in **Group I** with the solution technologies in **Group II**.

Group I	Group II
(P) Services oriented computing	(1) Interoperability
(Q) Heterogeneous communicating systems	(2) BPMN
(R) Information representation	(3) Publish-find bind
(S) Process description	(4) XML

(a) P−1, Q−2, R−3, S−4
(b) P−3, Q−4, R−2, S−1
(c) P−3, Q−1, R−4, S−2
(d) P−4, Q−3, R−2, S−1

(GATE 2013: 1 Mark)

Solution: Service oriented computing involves the operations such as find, publish and bind between the entities (service consumer, service registry and service provider).
Through interoperability, heterogeneous systems can communicate and work together.
XML is used to represent information in a specific format.
BPMN is used to represent business process description in a business process model.

Ans. (c)

5. A graphical HTML browser resident at a network client machine Q accesses a static HTML Web page from a HTTP server S. The static HTML page has exactly one static embedded image, which is also at S. Assuming no caching, which one of the following is correct about the HTML Web page loading (including the embedded image)?

(a) Q needs to send at least 2 HTTP requests to S, each necessarily in a separate TCP connection to server S.
(b) Q needs to send at least 2 HTTP requests to S, but a single TCP connection to server S is sufficient.
(c) A single HTTP request from Q to S is sufficient, and a single TCP connection between Q and S is necessary for this.
(d) A single HTTP request from Q to S is sufficient, and this is possible without any TCP connection between Q and S.

Solution: Client has to make atleast two http requests—one for the web page and another for embedded image. One TCP connection is sufficient for both requests because TCP connection is alive for the request.

Ans. (b)

PRACTICE EXERCISES

Set 1

1. The importance of the markup tag helps the browser

(a) how to orient the Web page.
(b) how to display the apps message.
(c) how to display the page.
(d) how to fetch a Web page.

2. What is the default filename of a homepage of a website?

(a) catch.html (b) index.html
(c) try.html (d) home.html

3. The <Table> tag has which of the following attributes?

(a) row (b) cellpadding
(c) font (d) link

4. Which of the following is not a pair tag?

(a) <i> (b) <u>
(c) <b> (d) <img>

5. DTD definition is used along with XML to specify

(a) the data types of the contents of XML document.
(b) the front page of XML document.
(c) the links of XML to other documents.
(d) the links of XML document from other documents.

6. XSL definition is used along with XML definition to specify

(a) the data structures and the links of XML document.
(b) the presentation of XML document.
(c) the layered display of XML with other documents.
(d) the structure of XML document.

7. Which of the following is a platform-free language?

(a) FORTRAN77 (b) PASCAL
(c) EIFFEL (d) JAVA

8. The importance of an HTML tag is

(a) It specifies the structure of your page.
(b) It specifies the connection to a particular page.

(c) It specifies the marking of the Web page.
(d) It specifies formatting and layout instructions of the Web page.

9. HTML is a subset of

(a) SGML (b) SGHT
(c) STML (d) SHML

10. The format of an HTML document is

(a) ASCII (b) Special JPEG
(c) Special TIFF (d) Special Web

11. The advantages of XML over HTML are:

A. It allows processing of data stored in Web pages.
B. It uses meaningful tags which aids in understanding the nature of a document.
C. Is simpler than HTML.
D. It separates presentation and structure of document.

(a) A, B and C only (b) A, C and D only
(c) A, B and D only (d) B and D only

12. XML uses

(a) user-defined tags. (b) pre-defined tags.
(c) extensible tags. (d) pairing tags.

13. A URL specifies the following:

A. Protocol used
B. Domain name of server hosting Web page
C. Name of folder with required information
D. Name of document formatted using HTML

(a) A, B, C and D (b) B, C and D
(c) B, C and D (d) A, C and

14. The World Wide Web supports

A. Encryption.
B. HTML.
C. HTTP.
D. Firewall.

(a) A, B and C (b) B and C
(c) A, C and D (d) A, B, C and D

15. The time taken by Internet packets

(a) can be predetermined before transmission.
(b) depends upon the size of the packet.
(c) is irrelevant for audio packets.
(d) is irrelevant for video packets.

16. The DNS maps the IP addresses to

(a) a binary address as strings.
(b) an alphanumeric address.
(c) a hierarchy of domain names.
(d) a hexadecimal address.

17. How can you create an email link?

(a) `<ahref="mailto:xxx@yyy">`
(b) `<ahref="xxx@yyy.zzz">`
(c) `<mailhref="xxx@yyy.abc">`
(d) `<mail>xxx@yyy.pqr</mail>`

18. How can you open a link in a new browser window?

(a) `<ahref="url" target="newblank">`
(b) `<ahref="url" target="_blank">`
(c) `<ahref="url" target="subwindow">`
(d) `<ahref="url" target="nextwindow">`

19. What does CSS stand for?

(a) Computer secure scheme.
(b) Computer securesheets.
(c) Cascading style sheets.
(d) Cascading secure sheets.

20. What is the correct HTML for referring to an external style sheet?

(a) `<link rel="stylesheet" type="text/css" href="mystyle.css">`
(b) `<style src="mystyle.css">`
(c) `<stylesheet>mystyle.css</stylesheet>`
(d) None of these

21. An external style sheet in an HTML document is placed

(a) in the <body> section.
(b) in the <linker> section.
(c) in the <head> section.
(d) in the <top> section.

22. The HTML tag used to define an internal style sheet is

(a) <style> (b) <stylesheet>
(c) <css> (d) <internal link>

23. Which of the following HTML attribute defines inline styles?

(a) style (b) inlinefont
(c) objectdefine (d) inlinestyles

24. The correct CSS statement is

(a) `{color=black(body}`
(b) `body {color:black}`
(c) `body:color:=black`
(d) `{body(color:black)}`

25. Which of the following inserts a comment in a CSS file?

(a) // this is a comment
(b) /* this is a comment */
(c) * this is a comment*
(d) “this is a comment”

26. How can we change the background colour?

(a) `bgcolor:"newcolor"`
(b) `color=background (colorname)`
(c) `background-color:`
(d) `BGCOL="colorname"`

27. To add a background color for all <h1> elements, which of the following HTML syntax is used?

(a) `h1 {background-color:#FFFFFF}`
(b) `{background-color:#FFFFFF}.h1`
(c) `{background-color:#FFFFFF}.h1(all)`
(d) `h1.all{bgcolor= #FFFFFF}`

28. To change the text colour in HTML we use

(a) color: (b) latest_text_color=
(c) modifytextcolor: (d) newcolor:

29. JavaScript is defined under which HTML element?

(a) <jscript> (b) <script>
(c) <scriptjava> (d) <define.js>

30. The correct syntax to write "Hi There" in Javascript is

(a) `jscript.write("Hi There")`
(b) `response.write("Hi There")`
(c) `print("Hi There")`
(d) `print.jscript("Hi There")`

31. To define the use of JavaScript in an HTML document, it is described in?

(a) The <insertscript> section.
(b) The <head> section.
(c) Both the <head> section and the <body> section.
(d) The <linkjscript> section.

32. What is the correct syntax for referring to an external JavaScript called "ps.js"?

(a) `<script type="text.javascript" name="ps.js">`
(b) `<script type="text/javascript" link="ps.js">`
(c) `<script type="text/javascript" src="ps.js">`
(d) `<script type="text/javascript" external="ps.js">`

33. Which of the following syntax is used to write "Hi" in an alert box?

(a) `msgBox("Hi")`
(b) `alert("Hi")`
(c) `alertmsgBox("Hi")`
(d) `msgalertBox="Hi"`

34. Which of the following statements is used for creating a function?

(a) `function=Functionname()`
(b) `function Functionname()`
(c) `function:Functionname()`
(d) `function "Functionname"`

35. How do you call a function named "myFunction"?

(a) myFunction()
(b) call function myFunction
(c) callmyFunction()
(d) callfunction.myFunction

36. The correct syntax to define "if" statement for executing some code if "x" is equals to 2?

(a) `if(x=2) then` (b) `ifx==2.0 then`
(c) `if(x==2)` (d) `if(x:=2)`

37. The correct syntax for a conditional statement to execute following code if "x" is NOT equal to 8?

(a) `if` x`=! 8 then` (b) `if (`x`<> 8)`
(c) `if` x`<>8 then` (d) `if (`x`!= 8)`

38. What does XML stand for?

(a) Exclusive modern links
(b) Extensible markup language
(c) Extra memory links
(d) Exit markup language

Set 2

1. Which is required by the Internet?

(a) An international agreement to connect computers
(b) A local area network
(c) A commonly agreed set of rules to communicate between computers
(d) A wide area network

2. Which services are available on the World Wide Web

A. Decryption
B. HTTP
C. HTML
D. Firewalls

(a) A and B (b) B and C
(c) B and D (d) A and D

3. The HTML link properties are:

(a) :link, :visit, :hover, :active
(b) :link, :visited, :hover, :active
(c) :linked, :visit, :over, :active
(d) :link, :hover, :active, :inactive

4. Which of the following is used to specify border type (solid or dotted or double line etc.)

(a) Border-style
(b) Border-layout
(c) Border-attrib
(d) Border-width

5. To declare the version of XML, the correct syntax is.

(a) `<?xml version='1.0'/>`
(b) `<*xml version='1.0'/>`
(c) `<?xml version="1.0"/>`
(d) `</xml version='1.0'/>`

6. Which of the following is TRUE in context of an XML language

(a) Every XML document must have a DTD.
(b) There is no difference in lower case and upper case tags.
(c) All documents must have a root element.
(d) Closing tag is not mandatory.

7. In an HTML document, the correct place to refer to an external style sheet is

(a) In the <body> section.
(b) In <external> tag.
(c) At the top of the document.
(d) In the <head> section.

ANSWERS TO PRACTICE EXERCISES

Set 1

1. (c) A markup language, such as HTML, is used by Web browsers to display text and graphics. The text includes markup tags such as to indicate the start of a paragraph, and to indicate the end of a paragraph. HTML documents are often referred to as Web pages.

2. (b) A home page is the Web page that is chosen by the Web server as a default page which appears on accessing the top level of a Web site, without specifying a particular file name. For example, if someone visits http://www.home.com/, the Web server refers a list of default file names (such as "index.html", "index.htm", etc.) and displays the first matching file it finds. Hence, the following two addresses will work identically, and display the file named "index.html".

http://www.home.com/

http://www.home.com/index.html

3. (b) Attributes of <Table> tag are align, bgcolor, border, cellpadding, cellspacing, frame, rules, sortable, summary, width, etc.

4. (d) All tags that need to be closed are pair tags. <img> need not to be closed.

5. (a) DTD is document type definition that gives information about data types of content of documents it is used with.

6. (b) The Extensible Stylesheet Language Family (XSL) represents the XML document and XLL definition is used along with XML to specify the links with other documents.

7. (d) JAVA is platform independent.

8. (d) HTML is a markup language used by Web browsers to represent text and graphics; it specifies formatting and layout instructions of the Web page.

9. (a) HTML is a subset of SGML (Standard General Markup Language). XML is a highly functional subset of SGML. XHTML extends and subsets HTML.

10. (a) The format of an HTML document is ASCII.

11. (b)

12. (a)

13. (a) The URL consists of two parts separated by a colon and two forward slashes—*protocol identifier* which indicates the protocol to be used and *resource name* which specifies the IP address or the domain name of the resource.

14. (b) WWW supports HTML, HTTP, XML, XHTML.

15. (c)	**23.** (a)	**31.** (c)
16. (c)	**24.** (b)	**32.** (c)
17. (a)	**25.** (b)	**33.** (b)
18. (b)	**26.** (c)	**34.** (b)
19. (c)	**27.** (a)	**35.** (a)
20. (a)	**28.** (a)	**36.** (c)
21. (c)	**29.** (b)	**37.** (d)
22. (a)	**30.** (a)	**38.** (b)

Set 2

1. (c)	**4.** (a)	**7.** (d)
2. (b)	**5.** (c)	
3. (b)	**6.** (c)	

SOLVED GATE (CS) 2014

SET 1

(Engineering Mathematics and Technical Section)

Q. No. 1 – 25 Carry One Mark Each

1. Consider the statement: "Not all that glitters is gold" Predicate glitters (x) is true if x glitters and predicate gold (x) is true if x is gold. Which one of the following logical formulae represents the above statement?

(a) $\forall x$; glitters(x) $\Rightarrow \neg$gold(x)
(b) $\forall x$; gold(x) $\Rightarrow$ glitters(x)
(c) $\exists x$; gold(x) $\wedge \neg$glitters(x)
(d) $\exists x$; glitters(x) $\wedge\neg$ gold(x)

Solution: S: "Not all that glitters is gold" means there exist some glitters that are not gold. So, $\exists x$ is used along with AND operator.
$\exists x$; glitters(x) $\wedge \sim$gold(x)

Ans. (d)

2. Suppose you break a stick of unit length at a point chosen uniformly at random. Then the expected length of the shorter stick is _____.

Solution: The length of shorter stick ranges between 0 and 0.5 meters. So, the expected length is 0.25.

Ans. (0.25)

3. Let G = (V,E) be a directed graph where V is the set of vertices and E the set of edges. Then which one of the following graphs has the same strongly connected components as G?

(a) $G_1 = (V,E_1)$ where $E_1 = \{(u,v) \mid (u,v) \notin E)\}$
(b) $G_2 = (V,E_2)$ where $E_2 = \{(u,v) \mid (v,u) \notin E)\}$
(c) $G_3 = (V,E_3)[$ where $E_3 = \{(u,v) \mid$ there is a path of length ≤ 2 from u to v in E$\}$
(d) $G_4 = (V,E_4)$ where V_4 is the set of vertices in G which are not isolated

Solution: Using topological sort, strongly connected components can be found. According to the property, if (u,v) belongs to E_2, (v,u) will not belong to E.

Ans. (b)

4. Consider the following system of equations:

$3x + 2y = 1$
$4x + 7z = 1$
$x + y + z = 3$
$x - 2y + 7z = 0$

The number of solutions for this system is _____.

Solution: The given equations are
$3x + 2y = 1$
$4x + 7z = 1$
$x + y + z = 3$
$x - 2y + 7z = 0$

Augmented matrix $\rho(A : B) = \begin{bmatrix} 3 & 2 & 0 & 1 \\ 4 & 0 & 7 & 1 \\ 1 & 1 & 1 & 3 \\ 1 & -2 & 7 & 0 \end{bmatrix}$

Applying row operations, the resultant matrix will be:

$$\begin{bmatrix} 1 & 1 & 1 & 3 \\ 0 & -4 & 3 & -11 \\ 0 & 0 & -15 & -21 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$\rho(A) = 3$ = number of variables. So equations have a unique solution.

Ans. (1)

5. The value of dot product of the eigenvectors corresponding to any pair of different eigenvalues of a 4-by-4 symmetric positive definite matrix is _____.

Solution: The Eigen for symmetric positive matrix corresponding to different Eigen values are orthogonal. Hence, their dot product is zero always.

Ans. (0)

6. Let the function

$$f(\theta) = \begin{vmatrix} \sin\theta & \cos\theta & \tan\theta \\ \sin(\pi/6) & \cos(\pi/6) & \tan(\pi/6) \\ \sin(\pi/3) & \cos(\pi/3) & \tan(\pi/3) \end{vmatrix}$$

where $\theta \in \left[\frac{\pi}{6}, \frac{\pi}{3}\right]$ and $f'(\theta)$ denote the derivative of f with respect to θ. Which of the following statements is/are **TRUE**?

(I) There exists $\theta \in \left[\frac{\pi}{6}, \frac{\pi}{3}\right]$ such that $f'(\theta) = 0$

(II) There exists $\theta \in \left[\frac{\pi}{6}, \frac{\pi}{3}\right]$ such that $f'(\theta) \neq 0$

(a) I only (b) II only

(c) Both I and II (d) Neither I nor II

Solution: By applying mean value theorem, both I and II found to be true.

Ans. (c)

7. Consider the following Boolean expression for F:

$$F(P, Q, R, S) = PQ + \overline{P}QR + \overline{P}Q\overline{R}S$$

The minimal sum-of products form of F is

(a) PQ + QR + QS (b) P + Q + R + S

(c) $\overline{P} + \overline{Q} + \overline{R} + \overline{S}$ (d) $\overline{P}R + \overline{P}\overline{R}S + P$

Solution:

$PQ + \overline{P}QR + \overline{P}Q\overline{R}S$

$PQ + \overline{P}Q(R + \overline{R}S)$

Applying property [A + BC = (A + B)(A + C)], we have

$PQ + \overline{P}Q(R + \overline{R})(R + S)$

$\Rightarrow PQ + \overline{P}Q(R + S)$

$\Rightarrow PQ + \overline{P}QR + \overline{P}QS$

$\Rightarrow Q(P + \overline{P}R) + \overline{P}QS$

$\Rightarrow Q(P + R) + \overline{P}QS$

$\Rightarrow QP + QR + \overline{P}QS$

$\Rightarrow Q(P + \overline{P})(P + S) + QR$

$\Rightarrow PQ + QS + QR$

Ans. (a)

8. The base (or radix) of the number system such that the following equation holds is _____.

Solution: $(312/20)_x = (13.1)_x$

$$\frac{3x^2 + x + 2}{2x} = \frac{x^2 + 3x + 1}{x} \Rightarrow x^2 - 5x = 0$$

$$\Rightarrow x = 0,\ 5$$

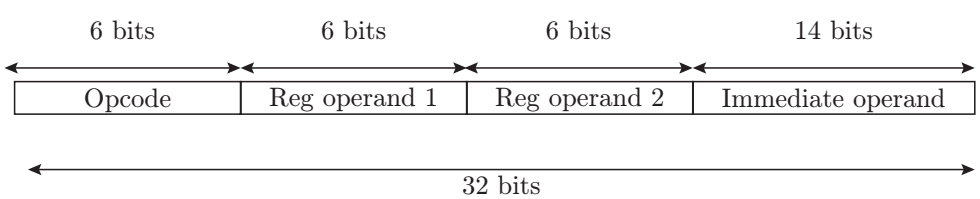
Base of number system cannot be 0. So, answer is 5.

Ans. (5)

9. A machine has a 32-bit architecture, with 1-word long instructions. It has 64 registers, each of which is 32 bits long. It needs to support 45 instructions, which have an immediate operand in addition to two register operands. Assuming that the immediate operand is an unsigned integer, the maximum value of the immediate operand is _____.

Solution: Given that instruction size = 32 bits; total instructions = 45, so opcode requires 6 bits; Total registers = 64, so 6 bits for register operands Immediate operand = 32 − 6 − 12 = 14

So, the maximum value of the immediate operand will be 16383



Ans. (16383)

10. Consider the following program in C language:

```
#include <stdio.h>
main()
{
 int i;
int *pi=&i;
scanf("%d", pi);
printf("%d \n", i+5);
}
```

Which one of the following statements is TRUE?

(a) Compilation fails.
(b) Execution results in a run-time error.
(c) On execution, the value printed is 5 more than the address of variable i.
(d) On execution, the value printed is 5 more than the integer value entered.

Solution: In the program, pi is a pointer variable that contains the address of i. scanf gets the value at address of i and printf prints value 5 more than i.

Ans. (d)

11. Let G be a graph with n vertices and m edges. What is the tightest upper bound on the running time of Depth First Search on G, when G is represented as an adjacency matrix?

(a) $\Theta(n)$ (b) $\Theta(n + m)$
(c) $\Theta(n^2)$ (d) $\Theta(m^2)$

Solution: Tightest upper bound for running depth first search using adjacency matrix is $O(V^2)$. Number of vertices here are n. So, $O(n^2)$ is the correct option.

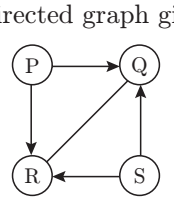
Ans. (c)

12. Consider rooted n node binary tree represented using pointers. The best upper bound on the time required to determine the number of sub trees having exactly 4 nodes is $\Theta(n^a \log_b n)$. Then the value of a + 10b is _____

Solution: On taking input, the root of the binary tree prints all the subtrees of size 4 in O(n) time, so a = 1, b = 0, and thus a + 10b = 1.

Ans. (1)

13. Consider the directed graph given below.



Which one of the following is TRUE?

(a) The graph does not have any topological ordering
(b) Both PQRS and SRQP are topological orderings
(c) Both PSRQ and SPRQ are topological orderings.
(d) PSRQ is the only topological ordering.

Solution: Topological order of a directed graph is the linear arrangement of vertices in which if there is an edge between u to v, then u comes before v in order. Option (c) satisfies this property, so both PSRQ and SPRQ are topological orderings.

Ans. (c)

14. Let P be a quick sort program to sort numbers in ascending order using the first element as the pivot. Let t_1 and t_2 be the number of comparisons made by P for the inputs [1 2 3 4 5] and [4 1 5 3 2], respectively. Which one of the following holds?

(a) $t_1 = 5$ (b) $t_1 < t_2$
(c) $t_1 > t_2$ (d) $t_1 = t_2$

Solution: Worst case of quick sort is when the list is already sorted. It requires n^2 comparisons. Whereas an unsorted list requires $n\log n$ comparisons. List 1 is sorted, so the number of comparisons are more for list 1. Therefore, (c) is the correct option ($t_1 > t_2$).

Ans. (c)

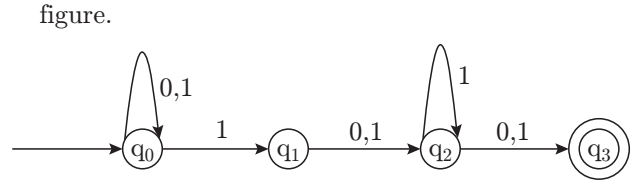
15. Which one of the following is TRUE?

(a) The language $L = \{a^n b^n \mid n \geq 0\}$ is regular.
(b) The language $L = \{a^n \mid n \text{ is prime}\}$ is regular.
(c) The language $L = \{w \mid w \text{ has } 3k + 1b\text{'s for some } k \in N \text{ with } \Sigma = \{a,b\}\}$ is regular.
(d) The language $L = \{ww \mid w \in \Sigma^* \text{ with } \Sigma = \{0,1\}\}$ is regular.

Solution:
(a) False. The language is not regular, because it requires memory for remembering number of a's and b's.
(b) False. This is a recursive language.
(c) True. The language is regular because DFA construction is possible.
(d) False. It is a context sensitive language.

Ans. (c)

16. Consider the finite automaton in the following figure.



What is the set of reachable states for the input string 0011?

(a) $\{q_1, q_2, q_3\}$ (b) $\{q_0, q_1\}$
(c) $\{q_0, q_1, q_2, q_3\}$ (d) $\{q_3\}$

Solution: With string '0011', we can reach $\{q_0, q_1, q_2\}$ states.

Ans. (a)

17. Which one of the following is FALSE?

(a) A basic block is a sequence of instructions where control enters the sequence at the beginning and exits at the end.
(b) Available expression analysis can be used for common subexpression elimination.
(c) Live variable analysis can be used for dead code elimination.
(d) $x = 4 \times 5 \Rightarrow x = 20$ is an example of common subexpression elimination.

Solution: $x = 4 \times 5 = 20$ is not a common subexpression elimination.

Ans. (d)

18. Match the following

(1) Waterfall model	(a) Specifications can be developed
(2) Evolutionary model	(b) Requirements compromises are inevitable
(3) Component based software	(c) Explicit recognition of risk
(4) Spiral development	(d) Inflexible partitioning of the project into stages

(a) 1-a, 2-b, 3-c, 4-d (b) 1-d, 2-a, 3-b, 4-c
(c) 1-d, 2-b, 3-a, 4-c (d) 1-c, 2-a, 3-b, 4-d

Solution: Waterfall model: Inflexible partitioning of the project into stages (backtracking is not possible to previous phase)
Evolutionary model: Specifications can be developed

Component based software: Requirements compromises are inevitable

Spiral development: Explicit recognition of risk

Ans. (b)

19. Suppose a disk has 201 cylinders, numbered from 0 to 200. At some time the disk arm is at cylinder 100, and there is a queue of disk access requests for cylinders 30, 85, 90, 100, 105, 110, 135 and 145. If Shortest-Seek Time First (SSTF) is being used for scheduling the disk access, the request for cylinder 90 is serviced after servicing _____ number of requests.

Solution: 30, 85, 90, 100, 105, 110, 135 and 145 will be sort according to SSTF:
100 →105 → 110 → 90 → 85 → 135 → 145 → 30

90 will be served after 3 requests.

Ans. (3)

20. Which one of the following is FALSE?

(a) User level threads are not scheduled by the kernel.
(b) When a user level thread is blocked, all other threads of its process are blocked.
(c) Context switching between user level threads is faster than context switching between kernel level threads.
(d) Kernel level threads cannot share the code segment.

Solution: Code, text and data are sharable in kernel threads. Only stack and registers are non-sharable.

Ans. (d)

21. Consider the relation scheme R = (E, F, G, H, I, J, K, L, M, N) and the set of functional dependencies {{E,F}→ {G},{F}→ {I,J}, {E,H} → {K,L} → {M}, {K} → {M}, {L} → {N} on R. What is the key for R?

(a) {E, F} (b) {E, F, H}
(c) {E, F, H, K, L} (d) {E}

Solution: Find the closure. If it contains all the elements in its closure, then it called a candidate key.
{EFH} + = {E, F, G, H, I, J, K, L, M, N}

Ans. (b)

22. Given the following statements:

S1: A foreign key declaration can always be replaced by an equivalent check assertion in SQL.

S2: Given the table R(a,b,c) where a and b together form the primary key, the following is a valid table definition.

```
CREATE TABLE S (
a INTEGER,
d INTEGER,
e INTEGER,
PRIMARY KEY (d),
FOREIGN KEY (a) references R)
```

Which one of the following statements is CORRECT?

(a) S1 is TRUE and S2 is a FALSE
(b) Both S1 and S2 are TRUE
(c) S1 is FALSE and S2 is a TRUE
(d) Both S1 and S2 are FALSE

Solution: S1: Employee (Name, Dept_ID)
Department (Dept_Name, Deptid)

In the given relation, Employee DeptID is a foreign key referencing Deptid (P.K) of relation Department.

Now, declare the foreign key by an equivalent check assertion as follows:-

CREATE TABLE Employee (

Name Varchar (10)

DeptID INT (6) check (DeptID IN (select Deptid from Department)),

PRIMARY KEY (Name));

The above use of check assertion is good to declare the foreign key as far as insertion is considered for relation Employee. But it will fail to implement changes done in Department relation in terms of deletion & updation. So, S1 is false.

S2: The given table definition is not valid due to invalid foreign key declaration. Attribute a is declared as foreign key which is a single valued attribute and it is referencing the primary key (a b) of relation R (a, b, c), which is a composite key. A single value attribute cannot refer a composite key. So, S2 is false.

23. Consider the following three statements about link state and distance vector routing protocols, for a large network with 500 network nodes and 4000 links

[S1] The computational overhead in link state protocols is higher than in distance vector protocols.

[S2] A distance vector protocol (with split horizon) avoids persistent routing loops, but not a link state protocol.

[S3] After a topology change, a link state protocol will converge faster than a distance vector protocol. Which one of the following is correct about S1, S2, and S3?

(a) S1, S2, and S3 are all true
(b) S1, S2, and S3 are all false.
(c) S1 and S2 are true, but S3 is false
(d) S1 and S3 are true, but S2 is false.

Solution: A distance vector protocol (with split horizon) avoids persistent routing loops, but not a link state protocol. So, S_2 is false.

Ans. (d)

24. Which one of the following are used to generate a message digest by the network security protocols?

(P) RSA (Q) SHA-1 (R) DES (S) MD5

(a) P and R only (b) Q and R only
(c) Q and S only (d) R and S only

Solution: Only SHA-1 and MD5 are used as message digest.

Ans. (c)

25. Identify the correct order in which the following actions take place in an interaction between a web browser and a web server.

1. The web browser requests a webpage using HTTP.
2. The web browser establishes a TCP connection with the web server.
3. The web server sends the requested webpage using HTTP.
4. The web browser resolves the domain name using DNS.

(a) 4, 2, 1, 3 (b) 1, 2, 3, 4
(c) 4, 1, 2, 3 (d) 2, 4, 1, 3

Solution: First of all, browser resolves IP address using DNS server, then connects to web server using TCP connection. After connection establishment, browser request for the page and web server responds with the page. So, option (a) is correct.

Ans. (a)

Q. No. 26 – 55 Carry Two Marks Each

26. Consider a token ring network with a length of 2 km having 10 stations including a monitoring station. The propagation speed of the signal is 2×10^8 m/s and the token transmission time is ignored. If each station is allowed to hold the token for 2 μsec, the minimum time for which the monitoring station should wait (in μsec) before assuming that the token is lost in _____.

Solution: Length = 2 km, stations = 10, signal speed = 2×10^8 m/s and token hold time = 2 μs

$$\text{RTT} = \frac{\text{Length}}{\text{Speed}} = \frac{2000 \text{ m}}{2 \times 10^8 \text{ m/s}} = 10 \ \mu\text{s}$$

Minimum wait time = RTT + Stations × Hold time

Minimum wait time = 10 + 10 × 2 = 30 μs

27. Let the size of congestion window of a TCP connection be 32 KB when a timeout occurs. The round trip time of the connection is 100 msec and the maximum segment size used is 2 KB. The time taken (in msec) by the TCP connection to get back to 32 KB congestion window in _____.

Solution: Given that RTT = 100 ms, congestion window size = 32 KB

$$\text{Time out} = \frac{\text{Congestion window}}{2} = \frac{32}{2} = 16 \text{ KB}$$

and MSS = 2 KB. Now,

2 KB → 1 RTT (Slow Start)

4 KB → 2 RTT

8 KB → 3 RTT

16 KB → 4 RTT (Threshold)

After threshold point is reached, there begins an additive increase phase.

18 KB → 5 RTT

20 KB → 6 RTT

22 KB → 7 RTT

.

.

30 KB → 11 RTT

32 KB

Therefore, total time = 11 RTTs
= 11 × 10
= 1100 ms

Ans. (1100)

28. Consider a selective repeat sliding window protocol that uses a frame size of 1 KB to send data on a 1.5 Mbps link with a one-way latency of 50 msec. To achieve a link utilization of 60%, the minimum number of bits required to represent the sequence number field is _____.

Solution: Given Frame Size = 1 KB, link capacity = 1.5 Mbps, one-way latency = 50 msec
RTT = 2 × 50 = 100 msec and link utilization = 60%

$$\text{Transmission Time (T)} = \frac{1000 \times 8}{1.5 \times 10^6} = 5.33 \text{ ms}$$

$$\text{Let window size} = n \text{ then link utilization} = \frac{n \times T}{(T + \text{RTT})}$$

$$\text{Therefore, } n = \frac{0.6(5.33 + 100)}{5.33} = 12$$

Window size in selective repeat = $2^{n-1} = 12$;
so $n - 1 = 4 \Rightarrow n = 5$

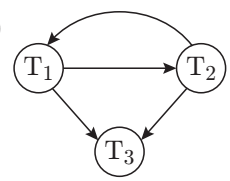
Ans. (5)

29. Consider the following four schedules due to three transactions (indicted by the subscript) using read and write on a data item x, denoted r(x) and w(x) respectively. Which one of them is conflict serializable?

(a) $r_1(x)$; $r_2(x)$; $w_1(x)$; $r_3(x)$; $w_2(x)$
(b) $r_2(x)$; $r_1(x)$; $w_2(x)$; $r_3(x)$; $w_1(x)$
(c) $r_3(x)$; $r_2(x)$; $r_1(x)$; $w_2(x)$; $w_1(x)$
(d) $r_2(x)$; $w_2(x)$; $r_3(x)$; $r_1(x)$; $w_1(x)$

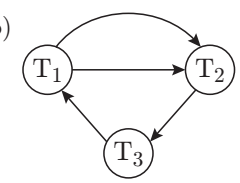
Solution To test conflict stabilizability, make precedence graph. If there is a cycle between any two nodes, then the given schedule is not conflict serializable.

(a)



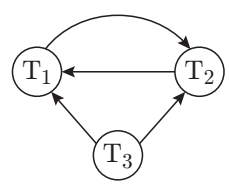
Cycle is there
So, not conflict serializable

(b)



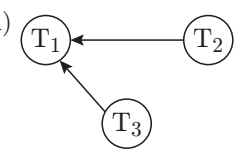
Cycle is there

(c)



Cycle is there

(d)



No cycle, so, it is a conflict serializable schedule

Ans. (d)

30. Given the following two statements:

S1: Every table with two single-valued attributes is in 1NF, 2NF, 3NF and BCNF

S2: AB→C, D→E, E→C is a minimal cover for the set of functional dependencies AB→C, D→E, AB→E, E→C

Which one of the following is **CORRECT**?

(a) S1 is TRUE and S2 is FALSE.
(b) Both S1 and S2 are TRUE.
(c) S1 is FALSE and S2 is TRUE.
(d) Both S1 and S2 are FALSE.

Solution:
S1: True, the highest normal form achieved by any relation with two single valued attributes is BCNF.

S2: False, because the closure of AB contains different set of attributes.

Ans. (a)

31. An operating system uses the Banker's algorithm for deadlock avoidance when managing the allocation of three resource types X, Y, and Z to three processes P0, P1, and P2. The table given below presents the current system state. Here, the allocation matrix shows the current number of resources of each type allocated to each process and the Max matrix shows the maximum number of resources of each type required by each process during its execution.

	Allocation			Max		
	X	Y	Z	X	Y	Z
P0	0	0	1	8	4	3
P1	3	2	0	6	2	0
P2	2	1	1	3	3	3

There are 3 units of type X, 2 units of type Y and 2 units of type Z still available. The system is currently in a safe state. Consider the following independent requests for additional resources in the current state:

REQ1: P0 requests 0 units of X, 0 units of Y and 2 units of Z

REQ2: P1 requests 2 units of X, 0 units of Y and 0 units of Z

Which one of the following is TRUE?

(a) Only REQ1 can be permitted.
(b) Only REQ2 can be permitted.
(c) Both REQ1 and REQ2 can be permitted.
(d) Neither REQ1 nor REQ2 can be permitted.

Solution:

Allocation	Max	Need	Available
X Y Z	X Y Z	X Y Z	X Y Z
0 0 3	8 4 3	8 4 0	3 2 0
3 2 0	6 2 0	3 0 0	
2 1 1	3 3 3	1 2 2	

REQ1:
P0 (0, 0, 2)
With (3, 2, 0), only request of P1 can be served. So, REQ1 cannot be fulfilled.

REQ2:
P1 (2,0,0)
With (3, 2, 0), request of P1 can be served. There is one safe sequence (P1, P2, P0). So, REQ2 can be fulfilled.

Ans. (b)

32. Consider the following set of processes that need to be scheduled on a single CPU. All the times are given in milliseconds

Process Name	Arrival Time	Execution Time
A	0	6
B	3	2
C	5	4
D	7	6
E	10	3

Using the *shortest remaining time first* scheduling algorithm, the average process turnaround time (in msec) is _____.

Solution

A	B	A	C	E	D

0 3 5 8 12 15 21

Average turnaround time =

$$\frac{(8-0)+(5-3)+(12-5)+(21-7)+(15-10)}{5}$$

$$= \frac{36}{5} = 7.2 \text{ ms}$$

Ans. (7.2)

33. Assume that there are 3 page frames which are initially empty. If the page reference string 1, 2, 3, 4, 2, 1, 5, 3, 2, 4, 6, the number of page faults using the optimal replacement policy is _____.

Solution:
1, 2, 3, 4, 2, 1, 5, 3, 2, 4, 6

		3	4	4	4	4	4	4	4
	2	2	2	2	2	2	2	2	6
1	1	1	1	1	1	5	3	3	3
F	F	F	F	H	H	F	F	H	F

Total page faults = 7

Ans. (7)

34. A canonical set of items is given below

s → L. > R

Q → R.

On input symbol < the set has

(a) a shift-reduce conflict and a reduce-reduce conflict.
(b) a shift-reduce conflict but not a reduce-reduce conflict.
(c) a reduce-reduce conflict but not a shift-reduce conflict.
(d) neither a shift-reduce nor a reduce-reduce conflict.

Solution: Input symbol < is not available in the given grammar. So, we cannot comment about the conflict that has arose. Option (d) is the correct answer.

Ans. (d)

35. Let L be a language and $\overline{L}$ be its complement. Which of the following is NOT a viable possibility?

(a) Neither L nor $\overline{L}$ is recursively enumerable (r.e.).
(b) One of L and $\overline{L}$ is r.e. but not recursive; the other is not r.e.
(c) Both L and $\overline{L}$ are r.e. but not recursive.
(d) Both L and $\overline{L}$ are recursive.

Solution: Recursively enumerable languages are not closed under complement. Thus, both L and $\overline{L}$ cannot be both recursive enumerable.

Ans. (c)

36. Which of the regular expressions given below represent the following DFA?

(I) 0*1(1+00*1)*
(II) 0*1*1+11*0*1
(III) (0+1)*1

(a) I and II only
(b) I and III only
(c) II and III only
(d) I, II, and III

Solution: The given DFA accepts all the strings ending with any number of 1's. All the three expressions are ending with 1 and are accepted by the given DFA.

Ans. (d)

37. There are 5 bags labeled 1 to 5. All the coins in a given bag have the same weight. Some bags have coins of weight 10 gm, others have coins of weight 11 gm. I pick 1, 2, 4, 8, 16 coins, respectively, from bags 1 to 5. Their total weight comes out to be 323 gm. Then the product of the labels of the bags having 11 gm coins is _____.

Solution From the given information, we have

$$a + 2b + 4c + 8d + 16e = 323$$

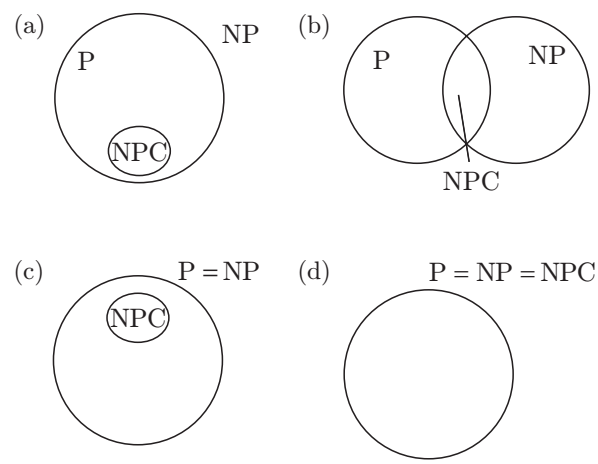
As other terms are multiples of 2, so a = odd = 11. Substituting, we get

$$2b + 4c + 8d + 16e = 312$$
$$\Rightarrow b + 2c + 4d + 8e = 156$$

So, b must be even = 10. Solving in similar manner, we find that $c = 11$, $d = 11$ and $e = 10$. So, the product of the labels of the bags having 11 gm coins is $1 \times 3 \times 4 = 12$.

Ans. (12)

38. Suppose a polynomial time algorithm is discovered that correctly computes the largest clique in a given graph. In this scenario, which one of the following represents the correct Venn diagram of the complexity classes P, NP and NP Complete (NPC)?



Solution Largest clique problem is known NP-Complete problem. In the question, it is given that there exists a polynomial time algorithm for largest clique problem. Thus, P = NP = NP-Complete.

Ans. (d)

39. The minimum number of comparisons required to find the minimum and the maximum of 100 numbers is _____.

Solution Number of comparisons for finding minimum and maximum of n numbers $= \frac{3}{2}n - 2$

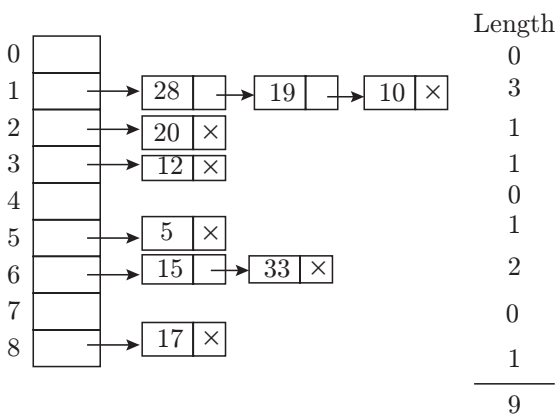
Here $n = 100$, so $(300/2) - 2 = 148$

Ans. (148)

40. Consider a hash table with 9 slots. The hash function is $h(k) = k \bmod 9$. The collisions are resolved by chaining. The following 9 keys are inserted in the order: 5, 28, 19, 15, 20, 33, 12, 17, 10. The maximum, minimum, and average chain lengths in the hash table, respectively, are

(a) 3, 0, and 1 (b) 3, 3, and 3
(c) 4, 0, and 1 (d) 3, 0, and 2

Solution:



Thus, average length $= \frac{9}{9} = 1$

Ans. (a)

41. Consider the following C function in which **size** is the number of elements in the array **E**:

```
int MyX(int *E, unsigned int size)
{
int Y = 0;
int Z;
int i, j, k;
for(i = 0; i < size; i++)
Y = Y + E[i];
for(i = 0; i < size; i++)
for(j = i; j < size; j++)
{
Z = 0;
for(k = i; k <= j; k++)
Z = Z + E[k];
if (Z > Y)
Y = Z;
}
return Y;
}
```

The value returned by the function **MyX** is the

(a) maximum possible sum of elements in any sub-array of array **E**.
(b) maximum element in any sub-array of array **E**.
(c) sum of the maximum elements in all possible sub-arrays of array **E**.
(d) the sum of all the elements in the array **E**.

Solution: The following code calculates sum of all the elements in array.

for(i = 0; i < size; i++)

Y = Y + E[i];

This Z = Z + E[k] calculates the sum of sub-array and compare it with Y. If sum is greater, then it is assigned to the Y. Hence, it calculates maximum possible sum of any subarray.

Ans. (a)

42. Consider the following pseudo code. What is the total number of multiplications to be performed?

```
D = 2
for i = 1 to n do
for j = i to n do
for k = j + 1 to n do
D = D * 3
```

(a) Half of the product of the 3 consecutive integers.
(b) One-third of the product of the 3 consecutive integers.
(c) One-sixth of the product of the 3 consecutive integers.
(d) None of the above.

Solution:

$$\sum(n-1) + \sum(n-2) + \sum 1 = \frac{n(n-1)(n+1)}{6}$$

Ans. (c)

43. Consider a 6-stage instruction pipeline, where all stages are perfectly balanced. Assume that there is no cycle-time overhead of pipelining. When an application is executing on this 6-stage pipeline, the speedup achieved with respect to non-pipelined

execution if 25% of the instructions incur 2 pipeline stall cycles is _____.

Solution: For non-pipeline 6 stages = 6 cycles are required

Pipeline time = 1 + (25/100) × 2 = 1.5

$$\text{Speed-up} = \frac{\text{Non-pipelined}}{\text{Pipeline time}} = \frac{6}{1.5} = 4$$

Ans. (4)

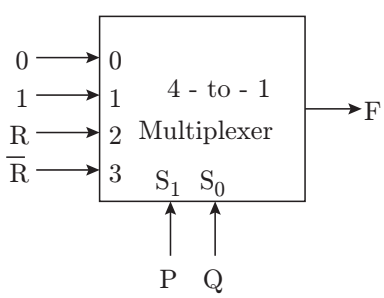
44. An access sequence of cache block addresses is of length N and contains n unique block addresses. The number of unique block addresses between two consecutive accesses to the same block address is bounded above k. What is the miss ratio if the access sequence is passed through a cache of associativity $A \geq k$ exercising least-recently-used replacement policy?

(a) n/N (b) $1/N$ (c) $1/A$ (d) k/n

Solution: The miss ratio is n/N.

Ans. (a)

45. Consider the 4-to-1 multiplexer with two lines S_1 and S_0 given below.



The minimal sum-of-products form of the Boolean expression for the output F of the multiplexer is

(a) $\overline{P}Q + QR + P\overline{Q}R$

(b) $\overline{P}Q + \overline{P}Q\overline{R} + PQ\overline{R} + P\overline{Q}R$

(c) $\overline{P}QR + \overline{P}Q\overline{R} + Q\overline{R} + P\overline{Q}R$

(d) $PQ\overline{R}$

Solution: $\overline{P}\overline{Q}0 + \overline{P}Q1 + P\overline{Q}R + PQ\overline{R}$

$\overline{P}Q + P\overline{Q}R + PQ\overline{R}$

$Q(\overline{P} + P\overline{R}) + P\overline{Q}R$

$Q(\overline{P} + P)(\overline{P} + \overline{R}) + P\overline{Q}R$

$Q\overline{P} + Q\overline{R} + P\overline{Q}R$

Ans. (a)

46. The function $f(x) = x \sin x$ satisfies the following equation. $f''(x) + f(x) + t\cos x = 0$. The value of t is _____.

Solution:

$f(x) = x \sin x$

$f''(x) + f(x) + t\cos x = 0$

$f'(x) = \sin x + x\cos x => f''(x) = \cos x + x(-\sin x) + \cos x = 2\cos x - x\sin x$

$2\cos x - x\sin x + x\sin x + t\cos x = 0$

$2\cos x + t\cos x = 0 \Rightarrow t = -2$

Ans. (−2)

47. A function f(x) is continuous the interval [0,2]. It is known that $f(0) = f(2) = -1$ and $f(1) = 1$.

Which one of the following statements must be true?

(a) There exists a y in the interval (0,1) such that $f(y) = f(y + 1)$
(b) For every y in the interval (0,1), $f(y) = f(2 - y)$
(c) The maximum value of the function in the interval (0.2) is 1
(d) There exists a y in the interval (0,1) such that $f(y) = f(2 - y)$

Solution: Let $g(x) = f(x) - f(x + 1)$ defined in [0,1]

$g(0) = f(0) - f(1) = -1 - 1 = -2$

$g(1) = f(1) - f(2) = 1 + 1 = +2$

$g(0)$ is negative and $g(1)$ is positive. Therefore, there exists some x for which $g(x) = 0$

$\Rightarrow g(x) = 0 = f(x) - f(x+1) \Rightarrow f(x) = f(x + 1)$

Ans. (a)

48. Four fair six-sided dice are rolled. The probability that the sum of the results being 22 is $X/1296$. The value of X is _____.

Solution Sum 22 comes with following possibilities:

6 + 6 + 6 + 4 = 22 (numbers can be arranged in 4 ways)

6 + 6 + 5 + 5 = 22 (numbers can arranged in 6 ways)

Therefore, the total ways $X = 4 + 6 = 10$

Ans. (10)

49. A pennant is a sequence of numbers, each number being 1 or 2. An n-pennant is a sequence of numbers with sum equal to n. For example, (1,1,2) is a 4-pennant. The set of all possible 1-pennants is {(1)}, the set of all possible 2-pennants is {(2), (1,1)} and the set of all 3-pennants is {(2,1), (1,1,1),

(1,2)}. Note that the pennant (1,2) is not the same as the pennant (2,1). The number of 10-pennants is _____.

Solution 10 pennants: 1 + 9 + 28 + 35 + 15 + 1 = 89

Ans. (89)

50. Let S denote the set of all functions f: $\{0,1\}^4 \rightarrow \{0,1\}^4$. Denote by N the number of functions from S to the set {0,1}. The value of $\log_2 \log_2 N$ is _____.

Solution: If a function is defined A → B, then

Size of A = |A| and Size of B = |B|

A = $\{0,1\}^4$ ⇒ Number of elements in A, that is, $|A| = 2^{16}$

Number of functions from A → B, $N = 2^{|A|}$

Therefore, $\log_2 \log_2 2^{|A|} = \log_2 |A| = \log_2 2^{16} = 16$

Ans. (16)

51. Consider an undirected graph G where self-loops are not allowed. The vertex set of G is {i,f): $1 \le i \le 12$, $1 \le j \le 12$}. There is an edge between (a,b) and (c,d) if $|a - c| \le 1$ and $|b - d| \le 1$ The number of edges in the graph is _____.

Solution According to the given graph:
4 vertices have degree 3, 40 vertices with degree 5 and 100 vertices with degree 8.

Total degree = 12 + 200 + 800 = 1012

So, 1012 = 2 edges ⇒ edges = 506

Ans. (506)

52. An ordered n-tuple $(d_1, d_2, \ldots, d_n)$ with $d_1^3\, d_2 \ldots, ^3d_n$ is called graphic if there exists a simple undirected graph with n vertices having degrees d_1, d_2, …, d_n, respectively. Which of the following 6-tuples is NOT graphic?

(a) (1, 1, 1, 1, 1, 1) (b) (2, 2, 2, 2, 2, 2)
(c) (3, 3, 3, 1, 0, 0) (d) (3, 2, 1, 1, 1, 0)

Solution Applying Havel Hakimi theorem

(a) (1,1,1,1,1,1)
0,1,1,1,1 ← Reorder
1, 1, 1, 1, 0
0, 1, 1, 0 ← Reorder
1, 1, 0, 0
0, 0, 0

(b) (2,2,2,2,2,2)
1,1,2,2,2 ← Reorder
2, 2, 2, 1, 1
1, 1, 1, 1
0, 1, 1 ← Reorder
1, 1, 0
0, 0

(c) (3,3,3,1,0,0)
2,2,0,0,0
1,−**1**,0,0 ← not graphic

(d) (3,2,1,1,1,0)
1,0,0,1,0 ← Reorder
1,1,0,0,0
0, 0, 0, 0

Ans. (c)

53. Which one of the following propositional logic formulas is TRUE when exactly two of p, q, and r are **TRUE**?

(a) $((p \leftrightarrow q) \wedge r) \vee (p \wedge q \wedge \sim r)$
(b) $(\sim (p \leftrightarrow q) \wedge r) \vee (p \wedge q \wedge \sim r)$
(c) $((p \rightarrow q) \wedge r) \vee (p \wedge q \wedge \sim r)$
(d) $(\sim (p \leftrightarrow q) \wedge r) \wedge (p \wedge q \wedge \sim r)$

Solution: Consider P = R = true and Q = false. All the options other than B will become false.

Ans. (b)

54. Given the following schema:

```
employees(emp-id, first-name, last-name,
hire-date, dept-id, salary)
departments(dept-id, dept-name,
manager-id, location-id)
```

You want to display the last names and hire dates of all latest hires in their respective departments in the location ID 1700. You issue the following query:

```
SQL>SELECT last-name, hire-date
FROM employees
WHERE (dept-id, hire-date) IN
(SELECT dept-id, MAX(hire-date)
FROM employees JOIN departments
USING(dept-id)
WHERE location-id = 1700
GROUP BY dept-id);
```

What is the outcome?

(a) It executes but does not give the correct result.
(b) It executes and gives the correct result.
(c) It generates an error because of pairwise comparison.
(d) It generates an error because the GROUP BY clause cannot be used with table joins in a sub-query.

Solution: Inner query joins employees and departments by using clause and apply filter location_id = 1700. Then the list will be grouped by dept_ids. This will select all the latest hire dates and dept_id. Outer query will result into last name and hire date of all the latest hires.

Ans. (b)

55. Consider two processors P_1 and P_2 executing the same instruction set. Assume that under identical conditions, for the same input, a program running on P_2 takes 25% less time but incurs 20% more CPI (clock cycles per instruction) as compared to the program running on P_1. If the clock frequency of P_1 is 1 GHz, then the clock frequency of P_2 (in GHz) is

Solution: Frequency of $P_1 = 1$ GHz $= 10^9$ Hz

1 cycle $\Rightarrow 1/10^9$ s $= 1$ ns

Suppose P_1 takes 5 cycles, then P_2 takes 20% more, that is, $5 + 5 \times (20/100) = 6$ cycles

P_2 takes 25% less time $= 5 - 5 \times (25/100) = 3.75$ ns

Let us say frequency of $P_2 = x$ GHz

x GHz $= x \times 10^9$ Hz

1 cycle $\Rightarrow 1/x$ ns

6 cycles $\Rightarrow 6/x$ ns $= 3.75$ ns

Therefore $x = 6/3.75 = 1.6$ GHz

Ans. (1.6)

SOLVED GATE (CS) 2014

SET 2

(Engineering Mathematics and Technical Section)

Q. No. 1 – 25 Carry One Mark Each

1. The security system at an IT office is composed of 10 computers of which exactly four are working. To check whether the system is functional, the officials inspect four of the computers picked at random (without replacement). The system is deemed functional if at least three of the four computers inspected are working. Let the probability that the system is deemed functional be denoted by p then $100p =$ _____.

Solution: P is the probability that system is functional means at least three systems are working, p will be calculated as:

P for at least 3 systems are working

$$= \frac{\text{All 4 systems working} + \text{3 systems working}}{\text{Number of ways to pick 4 systems from 10}}$$

P that 3 systems are working

$$= (\text{Number of systems not selected}) \times (\text{Number of ways to pick 3 from 4 selected systems})$$

$$= 6 \times 4 \times 3 \times 2 \times 4$$

Therefore,

P for at least 3 systems are working

$$= \frac{(4\times3\times2\times1)+(6\times4\times3\times2\times4)}{10\times9\times8\times7} = 0.1192$$

So, $100P = 11.92$

Ans. (11.92)

2. Each of the nine words in the sentence "**The quick brown fox jumps over the lazy dog**" is written on a separate piece of paper. These nine pieces of paper are kept in a box. One of the pieces is drawn at random from the box. The *expected* length of the word drawn is _____. (The answer should be rounded to one decimal place.)

Solution: The given 9 words are as follows:
The, Quick, Brown, Fox, Jumps, Over, The, Lazy, Dog

Words with length 3 are => THE, FOX, THE, DOG (4)

Words with length 4 are => OVER, LAZY (2)

Words with length 5 are => QUICK, BROWN, JUMPS (3)

Probability for drawn 3 length words = 4/9

Probability for drawn 4 length words = 2/9

Probability for drawn 5 length words = 3/9

$$\text{Expected word length} = \left\{\left(3\times\frac{4}{9}\right)+\left(4\times\frac{2}{9}\right)+\left(5\times\frac{3}{9}\right)\right\}$$

$$= 3.88$$

Ans. (3.88)

3. The maximum number of edges in a bipartite graph on 12 vertices is _____.

Solution: The maximum number of edges in bipartite graph $= \dfrac{n^2}{4}$

For $n = 12$, maximum number of edges in bipartite graph $= \dfrac{12\times12}{4} = 36$ edges

Ans. (36)

4. If the matrix A is such that

$$A = \begin{bmatrix} 2 \\ -4 \\ 7 \end{bmatrix} \begin{bmatrix} 1 & 9 & 5 \end{bmatrix}$$

Then the determinant of A is equal to _____.

Given matrix A is as:

$$\text{Determinate of } A = \begin{bmatrix} 2 & 18 & 10 \\ -4 & -36 & -20 \\ 7 & 63 & 35 \end{bmatrix} \Rightarrow$$

$$\begin{aligned} &\{2((-36\times 35)-(-20\times 63))\}+4((18\times 35)- \\ &(10\times 63))+7((-20\times 18)-(-36\times 10))\} \\ &=\{2((-1260)-(-1260))+4((630)- \\ &(630))+7((-360)-(-360))\} \\ &=\{2(0)+4(0)+7(0)\} \\ &=0 \end{aligned}$$

Ans. (0)

5. A non-zero polynomial $f(x)$ of degree 3 has roots at $x = 1$, $x = 2$ and $x = 3$. Which one of the following must be TRUE?

(a) $f(0)\, f(4) < 0$ (b) $f(0)\, f(4) > 0$
(c) $f(0) + f(4) > 0$ (d) $f(0) + f(4) < 0$

Solution: Given that $f(x)$ has degree 3. For $f(x) = 0$, roots are $x = 1, 2, 3$ which lies between 0 and 4. So, $f(0)$ and $f(4)$ have opposite sign. Therefore, $f(0)\, f(4)$ should be less than zero.

Ans. (a)

6. The dual of a Boolean function $F(x_1, x_2, \ldots x_n, +, \cdot, ')$ written as F^D, is the same expression as that of F with $+$ and $\cdot$ swapped. F is said to be self-dual if $F = F^D$. The number of self-dual functions with n Boolean variables is

(a) 2^n (b) 2^{n-1} (c) 2^{2^n} (d) $2^{2^{n-1}}$

Solution: A function is called self-dual if it has equal number of maxterms and minterms.

Total function $= 2^{2^n}$

Number of mutual exclusive terms $= 2^n/2$

Number of possible function by using minterms or maxterms taking any one at a time is (without mutual excusive terms) $= 2^{2^{n-1}}$

Ans. (d)

7. Let $k = 2^n$. A circuit is built by giving the output of an n-bit binary counter as input to an n-to-2^n bit decoder. This circuit is equivalent to a

(a) k-bit binary up counter.
(b) k-bit binary down counter.
(c) k-bit ring counter.
(d) k-bit Johnson counter.

Solution: In case of a k-bit ring encounter, single output will be 1 and remaining will be zero at a time.

Ans. (c)

8. Consider the equation $(123)_5 = (x8)_y$ with x and y as unknown. The number of possible solutions is _____.

Solution: Converting both sides of $(123)_5 = (x8)_y$ into decimal number system, we get

$$\begin{aligned} &1\times 5^2+2\times 5^1+3\times 5^0 = x\times y^1+8\times y^0 \\ &25+10+3 = xy+8 \\ &38 = xy+8 \Rightarrow xy = 30 \end{aligned}$$

Factors of 30 are -1, 2, 3, 15, 30

If $x = 1$ then $y = 30$
If $x = 2$ then $y = 15$
If $x = 3$ then $y = 10$

So, we have three solutions to the given problem.

Ans. (3)

9. A 4-way set-associative cache memory unit with a capacity of 16 KB is built using a block size of 8 words. The word length is 32 bits. The size of the physical address space is 4 GB. The number of bits for the TAG field is _____.

Solution:
A 4-way set associative cache memory has
Physical address size = 32 bits
Cache size = 16 K Bytes
Word length = 32 bits = 4 Bytes
Block size = 8 words = 8 × 4 = 32 Bytes

$$\text{Number of blocks} = \frac{\text{Cache size}}{\text{Block size}} = \frac{16\times 1024}{32} = 2^9$$

So, block of f set = 9 bits

$$\text{Number of sets} = \frac{2^9}{4} = 2^7$$

Set of f set = 7 bits

$$\text{Byte of } f \text{ set} = 32 \text{ bytes} = 2^5 \text{ bits} = 5 \text{ bits}$$

$$\begin{aligned} \text{TAG} &= \{\text{Physical address bits} - \\ &\quad (\text{Set of } f \text{ set} + \text{Byte of } f \text{ set})\} \\ &= \{32-(7+5)\} = 20 \text{ bits} \end{aligned}$$

10. Consider the function func shown below:

```
int func(int num) {
int count = 0;
while (num) {
count++;
num>>= 1;
}
return (count);
}
```

The value returned by func(435) is _____.

Solution: The given function has a while loop and termination condition for loop is, until num will not be zero. Function have a count variable which will count number of times will execute. Every time loop is executed, a right shift will be performed on the number. Initially, num = 435, binary equivalent = 110110011.

After first execution num = 011011001 and count = 1

After second execution num = 001101100 and count = 2

After third execution num = 000110110 and count = 3

After eight execution num = 000000001 and count = 8

After nine execution num = 000000000 and count = 9

The function will return count = 9.

Ans. (9)

11. Suppose n and p are unsigned int variables in a C program. We wish to set p to ${}^{n}C_3$. If n is large, which one of the following statements is most likely to set p correctly?

(a) $p = n \times (n-1) \times (n-2)/6$;
(b) $p = n \times (n-1)/2 \times (n-2)/3$;
(c) $p = n \times (n-1)/3 \times (n-2)/2$;
(d) $p = n \times (n-1)/2 \times (n-2)/6.0$;

Solution:

$P = {}^{n}C_3 = n \times (n-1) \times (n-2)/6$

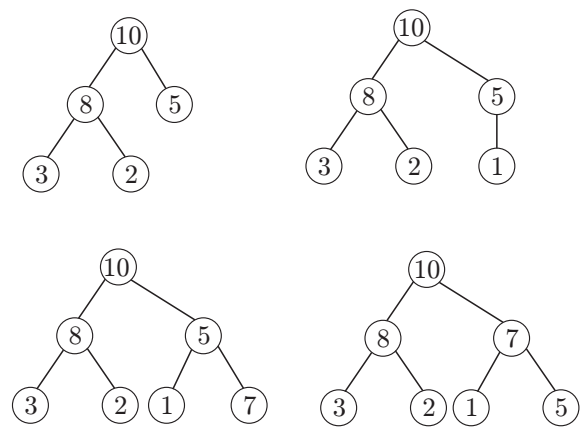
Computing $n \times (n-1) \times (n-2)$ together can go beyond the range; so, options (a) and (d) are not true. If n is even or odd then $n \times (n-1)/2$ will always result in integer value, thus more accuracy) whereas in the case of $n \times (n-1)/3$, it is not certain to get integer always so less accuracy. Hence, option (b) is more accurate.

Ans. (b)

12. A priority queue is implemented as a Max-Heap. Initially, it has 5 elements. The level-order traversal of the heap is: 10, 8, 5, 3, 2. Two new elements 1 and 7 are inserted into the heap in that order. The level-order traversal of the heap after the insertion of the elements is:

(a) 10, 8, 7, 3, 2, 1, 5
(b) 10, 8, 7, 2, 3, 1, 5
(c) 10, 8, 7, 1, 2, 3, 5
(d) 10, 8, 7, 5, 3, 2, 1

Solution:



Ans. (a)

13. Which one of the following correctly determines the solution of the recurrence relation with $T(1) = 1$?

$$T(n) = 2T\left(\frac{n}{2}\right) + \log n$$

(a) $\Theta(n)$
(b) $\Theta(n\log n)$
(c) $\Theta(n^2)$
(d) $\Theta(\log n)$

Solution:
Applying Master Theorem,

$T(n) = aT(n/b) + f(n)$

$T(n) = 2T(n/2) + \log n$;

$a = 2,\ b = 2,\ f(n) = \log n$

$\Rightarrow g(n) = (n^{\log_b a}) = n^{\log_2 2} = \Theta(n)$

$\Rightarrow f(n) = \log n = O(n^{\log_2 2-\varepsilon})$,

$\Rightarrow f(n)$ is smaller than $g(n)$ [as per Case 1 of Master Theorem]

$\Rightarrow T(n) = \Theta(n)$

So, option (a) is correct.

Ans. (a)

14. Consider the tree arcs of a BFS traversal from a source node W in an unweighted, connected, undirected graph. The tree T formed by the tree arcs is a data structure for computing

(a) the shortest path between every pair of vertices.
(b) the shortest path from W to every vertex in the graph.

(c) the shortest paths from W to only those nodes that are leaves of T.
(d) the longest path in the graph.

Solution: Application of BFS is to find the shortest path from u to v. In the given statement, W is the source node. It will calculate shortest path to every node from W. So, option (b) is correct.

Ans. (b)

15. If $L_1 = \{a^n \mid n \geq 0\}$ and $L_2 = \{b^n \mid n \geq 0\}$, Consider

(I) $L_1 \cdot L_2$ is a regular language
(II) $L_1 \cdot L_2 = \{a^n\ b^n \mid n \geq 0\}$

Which one of the following is CORRECT?

(a) Only (I) (b) Only (II)
(c) Both (I) and (II) (d) Neither (I) nor (II)

In option (1), concatenation of two regular languages is regular (property).
But in option (2) $L_1 \cdot L_2 = \{a^n\ b^n \mid n \geq 0\}$ is not regular because both the variables are independent in the given languages L_1 and L_2. So, only option (a) is correct.

Ans. (a)

16. Let A $\leq_m$ B denotes that language A is mapping reducible (also known as many-to-one reducible) to language B. Which one of the following is FALSE?

(a) If A $\leq_m$ B and B is recursive then A is recursive.
(b) If A $\leq_m$ B and A is undecidable then B is undecidable.
(c) If A $\leq_m$ B and B is recursively enumerable then A is recursively enumerable.
(d) If A $\leq_m$ B and B is not recursively enumerable then A is not recursively enumerable.

Solution: As language A is mapping reducible to language B, we have $f: \in^* \to \in^*$ where for every w, $w \in A \Leftrightarrow f(w) \in B$
Hence, if A $\leq_m$ B and B is not recursively enumerable then A is not recursively enumerable.

Ans. (d)

17. Consider the grammar defined by the following production rules, with two operators * and +

$$S \to T * P$$
$$T \to U \mid T * U$$
$$P \to Q + P \mid Q$$
$$Q \to Id$$
$$U \to Id$$

Which one of the following is TRUE?

(a) + is left associative, while * is right associative
(b) + is right associative, while * is left associative
(c) Both + and * are right associative
(d) Both + and * are left associative

Solution: From production rule P → Q + P, we can decide + is right associative and from T → T * U, we conclude * is left associative.

Ans. (b)

18. Which one of the following is NOT performed during compilation?

(a) Dynamic memory allocation
(b) Type checking
(c) Symbol table management
(d) Inline expansion

Solution: Dynamic memory allocation is done at the run time, not at compile time.

Ans. (a)

19. Which one of the following is TRUE?

(a) The requirements document also describes how the requirements that are listed in the document are implemented efficiently.
(b) Consistency and completeness of functional requirements are always achieved in practice.
(c) Prototyping is a method of requirements validation.
(d) Requirements review is carried out to find the errors in system design.

Solution: Prototype is constructed to check that requirements can be fulfilled or not.

Ans. (a)

20. A FAT (file allocation table) based file system is being used and the total overhead of each entry in the FAT is 4 bytes in size. Given a 100×10^6 bytes disk on which the file system is stored and data block size is 10^3 bytes, the maximum size of a file that can be stored on this disk in units of 10^6 bytes is _____.

Solution:

$$\text{Number of entries} = \frac{\text{Disk size}}{\text{Block size}} = \frac{10^8}{10^3} = 10^5$$

Entry size = 4 B, total space consumed = $4\text{B} \times 10^5$

Maximum file size can be stored = $10^8 - 0.4 \times 10^6 = 99.6 \times 10^6$ B

Ans. (99.6)

21. The maximum number of super keys for the relation schema R (E, F, G, H) with E as the key is _____.

Solution: Considering E as key, Total $2^3 = 8$ keys are possible.

Ans. (8)

22. Given an instance of the STUDENTS relation as shown below:

Student D	Student Name	Student	Email	Student Age	CPI
2345	Shankar	Shankar@math	X	9.4	
1287	Swati	Swati@ee	19	9.5	
7853	Shankar	Shankar@cse	19	9.4	
9876	Swati	Swati@mech	18	9.3	
8765	Ganesh	Ganesh@civil	19	8.7	

For (StudentName, StudentAge) to be a key for this instance, the value X should NOT be equal to _____.

Solution: Key (Student Name, student Age) should always be unique. If X = 19, the duplicate value will occur. This is not allowed. So, X cannot be 19.

Ans. (19)

23. Which one of the following is TRUE about the interior gateway routing protocols—Routing Information Protocol (RIP) and Open Shortest Path First (OSPF)?

(a) RILP uses distance vector routing and OSPF uses link state routing
(b) OSPF uses distance vector routing and RIP uses link state routing
(c) Both RIP and OSPF use link state routing
(d) Both RIP and OSPF use distance vector routing

Solution: Distance vector routing technique is used by RIP and OSPF uses link state routing.

Ans. (a).

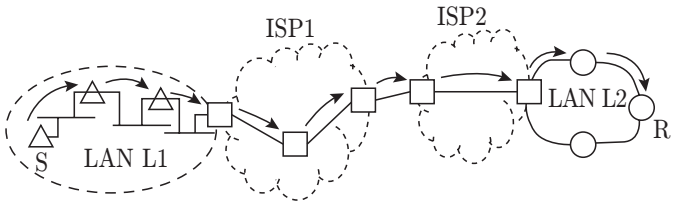
24. Which one of the following socket API functions converts an unconnected active TCP socket into a passive socket?

(a) Connect (b) Bind
(c) Listen (d) Accept

Solution: Listen function converts an unconnected active TCP socket into a passive socket.

Ans. (c)

25. In the diagram shown below, L1 is an Ethernet LAN and L2 is a Token-Ring LAN. An IP packet originates from sender S and traverses to R, as shown. The links within each ISP and across the two ISPs, are all 'point-to-point' optical links. The initial value of the TTL field is 32. The maximum possible value of the TTL field when R receives the datagram is _____.



Solution:
Initially, TTL = 32.

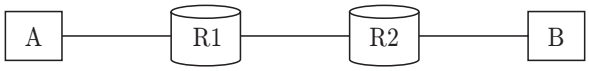
Number of routers = 5 and Last is receiver.

So, TTL value will be 32 − 6 = 26 at R.

Ans. (26)

Q. No. 26 – 55 Carry Two Marks Each

26. Consider the store and forward packet switched network given below. Assume that the bandwidth of each link is 10^6 bytes / sec. A user on host A sends a file of size 10^3 bytes to host B through routers R1 and R2 in three different ways. In the first case, a single packet containing the complete file is transmitted from A to B. In the second case, the file is split into 10 equal parts, and these packets are transmitted from A to B. In the third case, the file is split into 20 equal parts and these packets are sent from A to B. Each packet contains 100 bytes of header information along with the user data. Consider only transmission time and ignore processing, queuing and propagation delays. Also assume that there are no errors during transmission. Let T1, T2 and T3 be the times taken to transmit the file in the first, second and third case, respectively. Which one of the following is CORRECT?



(a) T1 < T2 < T3 (b) T1 > T2 > T3
(c) T2 = T3, T3 < T1 (d) T1 = T3, T3 > T2

Solution:
Given that bandwidth = 10^6 bytes/sec, file size = 1000 bytes, header size = 100 bytes

Case 1: A sends complete file to B

Total frame size = 1000 + 100 = 1100 bytes

Transmission time = $1100/10^6$ = 1100 μs

T1 = 3300 μs

Case 2: File size in chunks of 100 bytes

Total frame size = 100 + 100 = 200 bytes

Transmission time for 1 packet = $200/10^6$ = 200 μs

Transmission time for 10 packets = 2000 μs

T2 = 2000 + 200 + 200 = 2400 μs

Case 3: File size in chunks of 50 bytes

Total frame size = 50 + 100 = 150 bytes

Transmission time for 1 packet = $150/10^6$ = 150 μs

Transmission time for 20 packets = 3000 μs

T3 = 3000 + 150 + 150 = 3300 μs

T1 = T3 and T3 > T1

Ans. (d)

27. An IP machine Q has a path to another IP machine H via three IP routers R1, R2, and R3.

Q—R1—R2—R3—H

H acts as an HTTP server, and Q connects to H via HTTP and downloads a file. Session layer encryption is used, with DES as the shared key encryption protocol. Consider the following four pieces of information:

[I1] The URL of the file downloaded by Q

[I2] The TCP port numbers at Q and H

[I3] The IP addresses of Q and H

[I4] The link layer addresses of Q and H

Which of I1, I2, I3, and I4 can an intruder learn through sniffing at R2 alone?

(a) Only I1 and I2 (b) Only I1
(c) Only I2 and I3 (d) Only I3 and I4

Solution: Intruder can gain only I2 and I3 information. Because sniffing at R2 will able to capture only the TCP port numbers which are encapsulated in IP datagram and IP addresses of Q and H.

Ans. (c)

28. A graphical HTML browser resident at a network client machine Q accesses a static HTML webpage from a HTTP server S. The static HTML page has exactly one static embedded image which is also at S. Assuming no caching, which one of the following is correct about the HTML webpage loading (including the embedded image)?

(a) Q needs to send at least 2 HTTP requests to S, each necessarily in a separate TCP connection to server S
(b) Q needs to send at least 2 HTTP requests to S, but a single TCP connection to server S is sufficient
(c) A single HTTP request from Q to S is sufficient, and a single TCP connection between Q and S is necessary for this
(d) A single HTTP request from Q to S is sufficient, and this is possible without any TCP connection between Q and S

Solution: Q sends at least 2 HTTP request to S, single TCP connection is sufficient.

Ans. (b)

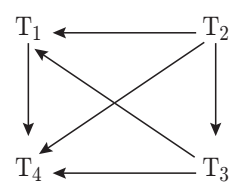
29. Consider the following schedule S of transactions T1, T2, T3, T4:

T1	T2	T3	T4
Writes [X]	Reads[X]		
Commit			
		Writes[X]	
		Commit	
	Writes[Y]		
	Reads [Z]		
	Commit		
			Reads[X]
			Reads[Y]
			Commit

Which one of the following statements is CORRECT?

(a) S is conflict-serializable but not recoverable
(b) S is conflict-serializable but is recoverable
(c) S is both conflict-serializable and recoverable
(d) S is neither conflict-serializable nor is it recoverable

Solution:



As there is no cycle in the graph, it is conflict serializable.

All the transactions are reading committed values, so schedule is recoverable.

Ans. (c)

30. Consider a join (relation algebra) between relations r(R) and s(S) using the nested loop method. There are 3 buffers each of size equal to disk block size, out of which one buffer is reserved for intermediate results. Assuming size(r(R)) < size(s(S)), the join will have fewer number of disk block accesses if

(a) relation r(R) is in the outer loop.
(b) relation s(S) is in the outer loop.
(c) join selection factor between r(R) and s(S) is more than 0.5.
(d) join selection factor between r(R) and s(S) is less than 0.5.

Solution: The relation r(R) should be in outer loop to have minimum number of disk accesses.

Ans. (a)

31. Consider the procedure below for the Producer-Consumer problem which uses semaphores:

Semaphore n = 0;

Semaphore s = 1;

```
void producer ( )
      {
      while (true)
      {
      Produce();
      SemWait(s);
      addToBuffer();
      semSignal(s);
      semSignal(n);
      }
      }
```

```
void consumer()
{
while(true)
semWait(s);
semWait(n);
removeFromBuffer();
semsignal(s);
consume();
}
}
```

Which one of the following is TRUE?

(a) The producer will be able to add an item to the buffer, but the consumer can never consume it.
(b) The consumer will remove no more than one item from the buffer.
(c) Deadlock occurs if the consumer succeeds in acquiring semaphore s when the buffer is empty.
(d) The starting value for the semaphore n must be 1 and not 0 for deadlock-free operation.

Solution: When consumer gets semaphore s and buffer is empty that means value of n is still zero. Thus, consumer goes into infinite loop at second step resulting in deadlock.

Ans. (c)

32. Three processes A, B and C each execute a loop of 100 iterations. In each iteration of the loop, a process performs a single computation that requires t_c CPU milliseconds and then initiates a single I/O operation that lasts for t_{io} milliseconds. It is assumed that the computer where the processes execute has sufficient number of I/O devices and the OS of the computer assigns different I/O devices to each process. Also, the scheduling overhead of the OS is negligible. The processes have the following characteristics:

Process id	t_c	t_{io}
A	100 ms	500 ms
B	350 ms	500 ms
C	200 ms	500 ms

The processes A, B, and C are started at times 0, 5 and 10 milliseconds, respectively, in a pure time sharing system (round robin scheduling) that uses a time slice of 50 milliseconds. The time in milliseconds at which process C would complete its first I/O operation is _____.

Solution: Time quantum = 50 ms

A	B	C	A	B	C	B	C	B	C	B

0 50 100 150 200 250 300 350 400 450 500 650

After completion of its CPU operations at 500 ms, C goes for I/O operations. This will last for 500 ms. Hence, process C will complete its first I/O at 500 + 500 = 1000 ms.

Ans. (1000)

33. A computer has twenty physical page frames which contain pages numbered 101 through 120. Now a program accesses the pages numbered 1, 2, …, 100 in that order, and repeats the access sequence THRICE. Which one of the following page replacement policies experiences the same number of page faults as the optimal page replacement policy for this program?

(a) Least-recently-used (b) First-in-first-out
(c) Last-in-first-out (d) Most-recently-used

Solution: Reference string = 1, 2, 3…. 100, 1, 2, 3…. 100, 1, 2, 3…. 100

In OPR technique, total page faults will be = 300 (100 every time)

This behaves same as most recently used, so numbers of page faults are same.

Ans. (d)

34. For a C program accessing **X[i] [j] [k]**, the following intermediate code is generated by a compiler. Assume that the size of an **integer** is 32 bits and the size of a **character** is 8 bits.

```
t0 = i * 1024
t1 = j * 32
t2 = k * 4
t3 = t1 + t0
t4 = t3 + t2
t5 = X[t4]
```

Which one of the following statements about the source code for the C program is CORRECT?

(a) X is declared as "int X[32] [32] [8]".
(b) X is declared as "int X[4] [1024] [32]".
(c) X is declared as "char X[4] [32] [8]".
(d) X is declared as "char X[32] [16] [2]".

Solution:
Let X be declared as X[A][B][C]

From t0 = i*1024, we have A * B = 1024.

From t1 = j*32, we have C*(size of Type) = 32.

From t2 = k*4, we have size of Type = 4.

Therefore, Type = int, A = 32, B = 32 and C = 8.

Ans. (a)

35. Let <M> be the encoding of a Turing machine as a string over S = {0,1}. Let L = {<M> | M is a Turning machine that accepts a string of length 2014}. Then, L is

(a) decidable and recursively enumerable
(b) undecidable but recursively enumerable
(c) undecidable and not recursively enumerable
(d) decidable but not recursively enumerable

Solution: Languages accepted by Turing machine are recursively enumerable. L is undecidable because Turing machine can halt or get stuck in infinite loop.

Ans. (b)

36. Let L_1 = {w ∈ {0,1}*|w has at least as many occurrences of (110)'s as (011)'s}. Let L_2 = {w ∈ {0,1}*|w has at least as many occurrence of (000)'s as (111)'s}. Which one of the following is TRUE?

(a) L_1 is regular but not L_2
(b) L_2 is regular but not L_1
(c) Both L_1 and L_2 are regular
(d) Neither L_1 nor L_2 are regular

Solution: Only L_1 is regular, construction of DFA is possible for L_1. L_2 is context free grammar, requires stack to remember count of 0's and 1's.

Ans. (a)

37. Consider two strings A "= q pqrr" and B = "pqprqrp". Let x be the length of the longest common subsequence (not necessarily contiguous) between A and B and let y be the number of such longest common subsequences between A and B. Then $x + 10y =$ _____.

Solution:
Given that A = "qpqrr" and B = "pqprqrp"

The LCS between A and B is having length 4, so x = 4 and possible common subsequences are as follows:

(1) qpqr
(2) pqrr
(3) qprr

So, $y = 3$. Hence, $x + 10y = 4 + 10 \times 3 = 34$.

Ans. (34)

38. Suppose P, Q, R, S, T are sorted sequences having lengths 20, 24, 30, 35, 50 respectively. They are to be merged into a single sequence by merging together two sequences at a time. The number of comparisons that will be needed in the worst case by the optimal algorithm for doing this is _____.

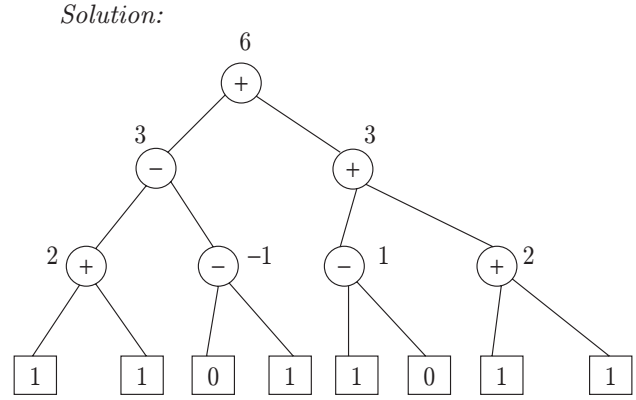
Solution: Merging two lists requires $m + n - 1$ comparisons.
20, 24, 30, 35, 50

Merge 20, 24 = 44 elements and 44 − 1 = 43 comparisons
Merge 30, 35 = 65 elements and 65 − 1 = 64 comparisons
Merge 44, 50 = 94 elements and 94 − 1 = 93 comparisons
Merge 94, 65 = 159 elements and 159 − 1 = 158 comparisons
Total comparisons = 43 + 64 + 93 + 158 = 358

Ans. (358)

39. Consider the expression tree shown. Each leaf represents a numerical value, which can either be 0 or 1. Over all possible choices of the values at the leaves, the maximum possible value of the expression represented by the tree is _____.

Solution:



Ans. (6)

40. Consider the following function

```
double f (double x) {
if (abs (x*x - 3) < 0. 01) return x;
else return f(x/2 + 1.5/x);
}
```

Give a value q (to 2 decimals) such that $f(q)$ will return q: _____.

Solution:

if(abs($p \times p^{-3}$) < 0.01) return p;

this will be true for $x = 1.72$

Ans. (1.72)

41. Suppose a stack implementation supports an instruction REVERSE, which reverses the order of elements on the stack, in addition to the PUSH and POP instructions. Which one of the following statements is TRUE with respect to this modified stack?

(a) A queue cannot be implemented using this stack.
(b) A queue can be implemented where ENQUEUE takes a single instruction and DEQUEUE takes a sequence of two instructions.
(c) A queue can be implemented where ENQUEUE takes a sequence of three instructions and DEQUEUE takes a single instruction.
(d) A queue can be implemented where both ENQUEUE and DEQUEUE take a single instruction each.

Solution:
An ENQUEUE & DEQUEUE operation takes four sequences of instructions:

```
Enqueue: Reverse, Push, Reverse
Dequeue: POP
```

(OR)

```
Enqueue: Push
Dequeue: Reverse, POP, Reverse
```

So, option (c) is correct.

Ans. (c)

42. Consider the C function given below

```
int f(int j)
{
static int i = 50;
int k;
if (i == j)
{
printf("something");
k = f(i);
return 0;
}
else return 0;
}
```

Which one of the following is TRUE?

(a) The function returns 0 for all values of j.
(b) The function prints the string something for all values of j.
(c) The function returns 0 when j = 50.
(d) The function will exhaust the runtime stack or run into an infinite loop when j = 50.

Solution: Condition i = = j becomes true for j = 50 and program will go to infinite loop.

Ans. (d)

43. In designing a computer's cache system, the cache block (or cache line) size is an important parameter. Which one of the following statements is correct in this context?

(a) A smaller block size implies better spatial locality
(b) A smaller block size implies a smaller cache tag and hence lower cache tag overhead
(c) A smaller block size implies a larger cache tag and hence lower cache hit time
(d) A smaller block size incurs a lower cache miss penalty

Solution: Smaller block size means more number of blocks can be accommodated in cache. This improves hit ratio and lower the miss penalty.

Ans. (d)

44. If the associativity of a processor cache is doubled while keeping the capacity and block size unchanged, which one of the following is guaranteed to be NOT affected?

(a) Width of tag comparator
(b) Width of set index decoder
(c) Width of way selection multiplexor
(d) Width of processor to main memory data bus

Solution: If associatively is doubled, then the index bits will decrease and tag bits will increase. So, the options (a), (b) and (c) will get affected.

Ans. (d)

45. The value of a float type variable is represented using the single-precision 32-bit floating point format of IEEE-754 standard that uses 1 bit for sign, 8 bits for biased exponent and 23 bits for mantissa. A float type variable X is assigned the decimal value of −14.25. The representation of X in hexadecimal notation is

(a) C1640000H (b) 416C0000H
(c) 41640000H (d) C16C0000H

Solution:

S	E	M
1 bit	8 bits	23 bits

$(14.25)_{10} = (1110.01)_2$

Now, normalize the binary number 1.11001×2^3

E = e + Bias

E = 3 + 127 = 130

Binary equivalent of 130 = $(10000010)_2$

Therefore, S = 1 (Since negative number)

M = (11001000…)

E = 10000010

1100	0001	0110	0100	0000	….
C	1	6	4	0	000

Ans. (a)

46. In the Newton-Raphson method, an initial guess of $x_0 = 2$ is made and the sequence x_0, x_1, x_2, … is obtained for the function $0.75x^3 - 2x^2 - 2x + 4 = 0$

Consider the statements

(I) $x_3 = 0$.

(II) The method converges to a solution in a finite number of iterations.

Which of the following is TRUE?

(a) Only I (b) Only II
(c) Both I and II (d) Neither I nor II

Solution: According to Newton–Raphson method:

$x_1 = x_0 - f_0/f_0'$

$f(x) = 0.75x^3 - 2x^2 - 2x + 4$

$f'(x) = 2.25x^2 - 4x - 2$

$x_0 = 2, f_0 = -2, \ f_0' = -1$

$x_1 = 2 - (-2)/(-1) = 0$

$f_1 = 4, \ f_1' = -2$

$x_2 = 0 - (4)/(-2) = 2$

$f_2 = -2, \ f_2' = -1$

$x_3 = 2 - (-2)/(-1) = 0$

Hence, (I) is true. Also the value diverges, thus option (II) is false.

Ans. (a)

47. The product of the non-zero eigenvalues of the matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

is _____.

Solution:

Eigen vector is AX = λX

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Convert the given matrix in diagonal matrix. The diagonal entries are called Eigen values.

Eigen values = 2, 3

Product of Eigen values = 2 × 3 = 6

Ans. (6)

48. The probability that a given positive integer lying between 1 and 100 (both inclusive) is NOT divisible by 2, 3 or 5 is _____.

Solution: Let A event → Divisible by 2, B event → Divisible by 3, C event → Divisible by 5

$P(A' \cap B' \cap C') = 1 - P(A \cup B \cup C)$

$P(A) = 50/100$, $P(B) = 33/100$, $P(C) = 20/100$

$P(A \cap B) = 16/100$, $P(B \cap C) = 6/100$,

$P(A \cap C) = 10/100$, $P(A \cap B \cap C) = 3/100$

$P(A \cup B \cup C) = P(A) + P(B) + P(C) - P(A \cap B) - P(B \cap C) - P(A \cap C) + P(A \cap B \cap C)$

$P(A \cup B \cup C) = (50 + 33 + 20 - 16 - 6 - 10 + 3)/100 = 74/100$

$P(A' \cap B' \cap C') = 1 - 74/100 = 26/100 = 0.26$

Ans. (0.259 to 0.261)

49. The number of distinct positive integral factors of 2014 is _____.

Solution:

2014 = 2 × 19 × 53 (product of prime factors)

Number of distinct positive integral factors of 2014 = (1 + 1) × (1 + 1) × (1 + 1) = 8

Ans. (8)

50. Consider the following relation on subsets of the set S of integers between 1 and 2014. For two distinct subsets U and V of S we say U < V if the minimum element in the symmetric difference of the two sets is in U.

Consider the following two statements:

S1: There is a subset of S that is larger than every other subset.

S2: There is a subset of S that is smaller than every other subset.

Which one of the following is CORRECT?

(a) Both S1 and S2 are true
(b) S1 is true and S2 is false
(c) S2 is true and S1 is false
(d) Neither S1 nor S2 is true

Solution:

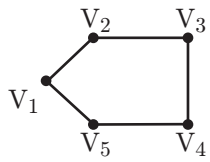
Both S1 and S2 are true.

Ans. (a)

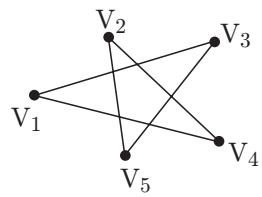
51. A cycle on n vertices is isomorphic to its complement. The value of n is _____.

Solution:

Consider $n = 5$,



Cycle graph

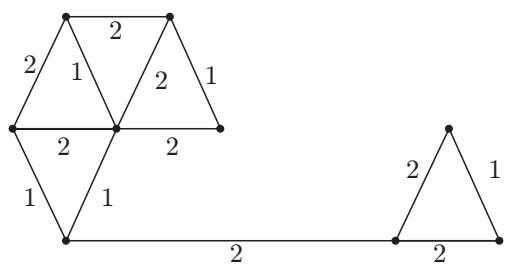


Complement of cycle graph

By checking properties these two graphs are isomorphic.

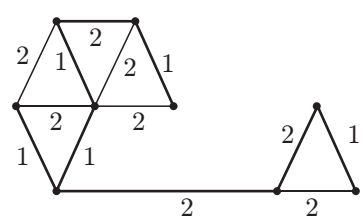
Ans. (5)

52. The number of distinct minimum spanning trees for the weighted graph below is



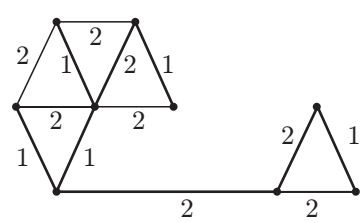
Solution:

1→

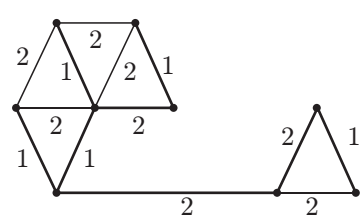


Applying Kruskal's

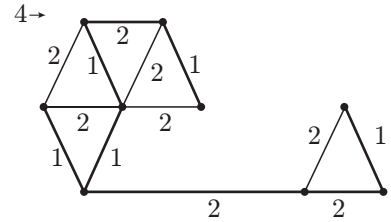
2→



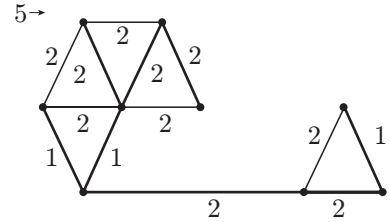
3→



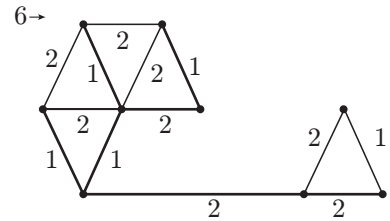
4→



5→



6→



Hence, number of distinct minimum spanning trees are 6.

Ans. (6)

53. Which one of the following Boolean expressions is NOT a tautology?

(a) $((a \to b) \wedge (b \to c)) \to (a \to c)$
(b) $(a \leftrightarrow c) \to (\sim b \to (a \wedge c))$
(c) $(a \wedge b \wedge c) \to (c \wedge a)$
(d) $a \to (b \to a)$

Solution: Tautology means all the output values should be true. Hence, check the expressions by trying all the possible combinations.

Ans. (b)

54. SQL allows duplicate tuples in relations, and correspondingly defines the multiplicity of tuples in the result of joins. Which one of the following queries always gives the same answer as the nested query shown below:

Select * from R where a in (select S. a from S)

(a) Select R. * from R, S where R. a = S. a
(b) Select distinct R. * from R, S where R. a = S. a
(c) Select R. * from R, (select distinct a from S) as S1 where R. a = S1. a
(d) Select R. * from R, S where R. a = S. a and is unique R

Solution:

Let R

A	B
A1	3
A2	4
A3	4

S

A	c
A1	5
A2	4
A2	4
A3	6

Query Output:

A	b
A1	3
A2	4
A2	4

(a)

A	b
A1	3
A2	4
A2	4
A2	4

(b)

A	c
A1	3
A2	4

(c)

A	b
A1	3
A2	4
A2	4

(d)

A	b
A1	3
A2	4

Hence, option (c) is correct.

Ans. (c)

55. Consider a main memory system that consists of 8 memory modules attached to the system bus, which is one word wide. When a write request is made, the bus is occupied for 100 nanoseconds (ns) by the data, address, and control signals. During the same 100 ns, and for 500 ns thereafter, the addressed memory module executes one cycle accepting and storing the data. The (internal) operation of different memory modules may overlap in time, but only one request can be on the bus at any time. The maximum number of stores (of one word each) that can be initiated in 1 millisecond is _____.

Solution:

For write request bus is occupied for = 100 ns

For storing the data requires = 100 ns

100 ns $\rightarrow$ 1 store

$100/10^9 \rightarrow$ 1 store

1 sec $\rightarrow 10^7$

10^{-3} s $\rightarrow 10^{-3} \times 10^7$ stores $= 10^4$, that is, 10,000 stores

Ans. (10000)

SOLVED GATE (CS) 2014

SET 3

(Engineering Mathematics and Technical Section)

Q. No. 1 – 25 Carry One Mark Each

---

1. Consider the following statements:

P: Good mobile phones are not cheap

Q: Cheap mobile phones are not good

L: P implies Q

M: Q implies P

N: P is equivalent to Q

Which one of the following about L, M, and N is CORRECT?

(a) Only L is TRUE.
(b) Only M is TRUE.
(c) Only N is TRUE.
(d) L, M and N are TRUE.

Solution:

P: good mobile phones are not cheap

Q: Cheap mobile phones are not good.

$P \to Q$ and $Q \to P$

Hence, $P \leftrightarrow Q$. So, L, M and N are true.

Ans. (d)

2. Let X and Y be finite sets and $f : X \to Y$ be a function. Which one of the following statements is TRUE?

(a) For any subsets A and B of X, $|f(A \cup B)| = |f(A)| + |f(B)|$
(b) For any subsets A and B of X, $f(A \cap B) = f(A) \cap f(B)$
(c) For any subsets A and B of X, $|f(A \cap B)| = \min\{|f(A)|, |f(B)|\}$
(d) For any subsets S and T of Y, $f^{-1}(S \cap T) = f^{-1}(S) \cap f^{-1}(T)$.

Solution:
Take example:

$X = \{1, 2, 3\}$ $Y = \{b, c\}$

$S = \{1, 2\}$ $T = \{b\}$

Ans. (d)

3. Let G be a group with 15 elements. Let L be a subgroup of G. It is known that $L \neq G$ and that the size of L is at least 4. The size of L is _____.

Solution: According to Lagrange's theorem, the order of subgroup divides order of group. Order of group is 15.

3, 5, 15 divide 15. According to the given condition, size should be atleast 4 and cannot be equal to that of group. So, 5 is the order of subgroup.

Ans. (5)

4. Which one of the following statements is TRUE about every $n \times n$ matrix with only real eigenvalues?

(a) If the trace of the matrix is positive and the determinant of the matrix is negative, at least one of its eigenvalues is negative.
(b) If the trace of the matrix is positive, all its eigenvalues are positive.
(c) If the determinant of the matrix is positive, all its eigenvalues are positive.
(d) If the product of the trace and determinant of the matrix is positive, all its eigenvalues are positive.

Solution: Determinant of matrices = Product of Eigen values

Ans. (a)

5. If V_1 and V_2 are 4-dimensional subspaces of a 6-dimensional vector space V, then the smallest possible dimension of $V_1 \cap V_2$ is _____.

Solution:
Let V be $\{e_1, e_2, e_3, e_4, e_5, e_6\}$

$V_1 = \{e_1, e_3, e_5, e_6\}$

$V_2 = \{e_2, e_3, e_4, e_5\}$

Smallest possible set of $V_1 \cap V_2 = 2$

Ans. (2)

6. If $\int_0^{2\pi} |x \sin x| \, dx = k\pi$, then the value of k is equal to _____.

Solution:

$$
\begin{aligned}
|\sin x| &= -\sin x \,[\text{from } \pi \text{ to } 2\pi] \\
&= \int_0^{\mu} x \sin x \, dx + \int_{\pi}^{2\pi} -x \sin x \, dx \\
&= x(-\cos x) - 1(-\sin x) \,|-(-x\cos x + \sin x)| \\
&= 4\pi
\end{aligned}
$$

Ans. (4)

7. Consider the following minterm expression for F.

F(P,Q,R,S) = $\sum$ 0, 2, 5, 7, 8, 10, 13, 15

The minterms 2, 7, 8 and 13 are do not care terms. The minimal sum-of-products form for F is

(a) $Q\bar{S} + \bar{Q}S$

(b) $\bar{Q}\bar{S} + QS$

(c) $\bar{Q}\bar{R}\bar{S} + \bar{Q}R\bar{S} + Q\bar{R}S + QRS$

(d) $\bar{P}\bar{Q}\bar{S} + \bar{P}QS + PQS + P\bar{Q}\bar{S}$

Solution:

PQ\RS	$\bar{R}\bar{S}$	$\bar{R}S$	**RS**	$R\bar{S}$
$\bar{P}\bar{Q}$	1			X
$\bar{P}Q$		1	X	
PQ		X	1	
$P\bar{Q}$	X			1

$QS + \bar{Q}\,\bar{S}$

Ans. (b)

8. Consider the following combinational function block involving four Boolean variables x, y, a, b where x, a, b are inputs and y is the output.

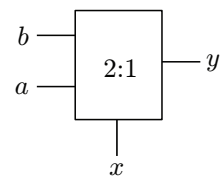
```
f (x, y, a, b)
{
if (x is 1) y = a;
else y = b;
}
```

Which one of the following digital logic blocks is the most suitable for implementing this function?

(a) Full adder (b) Priority encoder
(c) Multiplexor (d) Flip–flop

Solution:

$y = x' \, b + xa$



Ans. (c)

9. Consider the following processors (ns stands for nanoseconds). Assume that the pipeline registers have zero latency.

P1: Four-stage pipeline with stage latencies 1 ns, 2 ns, 2 ns, 1 ns.

P2: Four-stage pipeline with stage latencies 1 ns, 1.5 ns, 1.5 ns, 1.5 ns.

P3: Five-stage pipeline with stage latencies 0.5 ns, 1 ns, 1 ns, 0.6 ns, 1 ns.

P4: Five-stage pipeline with stage latencies 0.5 ns, 0.5 ns, 1 ns, 1 ns, 1.1 ns.

Which processor has the highest peak clock frequency?

(a) P1 (b) P2 (c) P3 (d) P4

Solution:
Maximum clock pulse will be of duration = 2 ns for P1

Minimum clock pulse duration = 1 ns for P3

Frequency $\propto$ 1/Clock pulse

Highest frequency will be of P3.

Ans. (c)

10. Let A be a square matrix size $n \times n$. Consider the following pseudocode. What is the expected output?

```
C = 100;
for i = 1 to n do
for j = 1 to n do
 {
```

```
Temp = A[i] [j] + C;
A[i][j] = A [j][i];
A[j][i] = Temp - C;
}
for i = 1 to n do
for j = 1 to n do
output (A[i][j]);
```

(a) The matrix A itself
(b) Transpose of the matrix A
(c) Adding 100 to the upper diagonal elements and subtracting 100 from lower diagonal elements of A
(d) None of these

Solution: A[i][j] and A[j][i] exchanging their values. Perform the operation for atleast three iterations.

Ans. (a)

11. The minimum number of arithmetic operations required to evaluate the polynomial $P(X) = X^5 + 4X^3 + 6X + 5$ for a given value of X, using only one temporary variable is _____.

Solution:

$P(X) = X^3(X^2 + 4) + 6X + 4$

$X^2 = 1$

$X^3 = 1$

$X^3 \times (X^2 + 4) =$ One multiplication + One addition

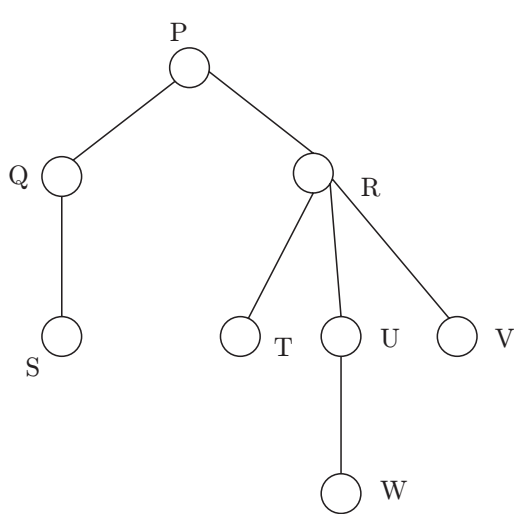
$=$ 2 operations

$6X = 1$ multiplication

$5 + 2$ additions $= 7$ operations

Ans. (7)

12. Consider the following rooted tree with the vertex labelled P as the root



The order in which the nodes are visited during an in-order traversal of the tree is

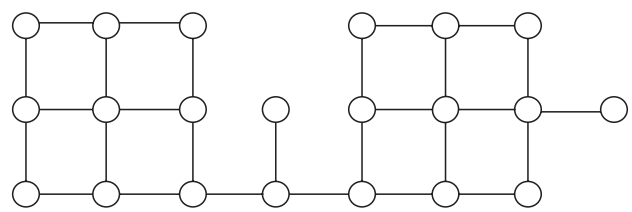
(a) SQPTRWUV (b) SQPTUWRV
(c) SQPTWUVR (d) SQPTRUWV

Solution:

Left Root Middle Right
SQPTRWUV

Ans. (a)

13. Suppose depth first search is executed on the graph below starting at some unknown vertex. Assume that a recursive call to visit a vertex is made only after first checking that the vertex has not been visited earlier. Then the maximum possible recursion depth (including the initial call) is _____.



Solution: In depth first search, the nodes are traversed in depth. Maximum possible depth is 19.

Ans. (19)

14. You have an array of n elements. Suppose you implement quick sort by always choosing the central element of the array as the pivot. Then the tightest upper bound for the worst case performance is

(a) $O(n^2)$ (b) $O(n \log n)$
(c) $\Theta(n \log n)$ (d) $O(n^2)$

Solution: Worst case is if the list is already sorted. So, worst case complexity is $O(n^2)$.

Ans. (a)

15. The length of the shortest string NOT in the language (over S = {a,b}) of the following regular expression is _____. a*b* (ba)* a*

Solution: String of length 3 'bab' is not present.

Ans. (3)

16. Let Σ be a finite non-empty alphabet and let 2^{Σ^*} be the power set of Σ *. Which one of the following is TRUE?

(a) Both 2^{Σ^*} and Σ^* are countable
(b) 2^{Σ^*} is countable Σ^* is uncountable
(c) 2^{Σ^*} is uncountable and Σ^* is countable
(d) Both 2^{Σ^*} and Σ^* are uncountable

Solution: Σ^* is countable infinite. The power set of countable infinite sets is uncountable.

Ans. (c)

17. One of the purposes of using intermediate code in compilers is to

(a) make parsing and semantic analysis simpler.
(b) improve error recovery and error reporting
(c) increase the chances of reusing the machine-independent code optimizer in other compliers.
(d) improve the register allocation.

Solution: Increase the chances of reusing the machine-independent code optimizer in other compliers.

Ans. (c)

18. Which of the following statements are CORRECT?

(1) Static allocation of all data areas by a compiler makes it impossible to implement recursion.
(2) Automatic garbage collection is essential to implement recursion.
(3) Dynamic allocation of activation records is essential to implement recursion.
(4) Both heap and stack are essential to implement recursion.

(a) 1 and 2 only (b) 2 and 3 only
(c) 3 and 4 only (d) 1 and 3 only

Solution: For recursion, dynamic memory allocation is required.

Ans. (d)

19. In the context of modular software design, which one of the following combinations is desirable?

(a) High cohesion and high coupling
(b) High cohesion and low coupling
(c) Low cohesion and high coupling
(d) Low cohesion and low coupling

Solution: Cohesion measures the internal dependency within module whereas coupling presents inter module dependence. So, there should be high cohesion and low coupling.

Ans. (b)

20. A system uses 3 page frames for storing process pages in main memory. It uses the Least Recently Used (LRU) page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults that will occur while processing the page reference string given below?

4, 7, 6, 1, 7, 6, 1, 2, 7, 2

Solution:
LRU with 3 frames

$4 \rightarrow 1$

$7 \rightarrow 2$

$6 \rightarrow 7$

Page faults = 6

Ans. (6)

21. What is the optimized version of the relation algebra expression $(\pi_{A1}(\pi_{A2}(\sigma_{F1}(r))))$, where A_1, A_2 are sets of attributes in with $A_1 \subset A_2$ and F_1, F_2 are Boolean expressions based on the attributes in r?

(a) $\pi_{A1}(\sigma(F_1 \wedge F_2)\ (r))$ (b) $\pi_{A1}(\sigma(F_1 \vee F_2)\ (r))$
(c) $\pi_{A2}(\sigma(F_1 \wedge F_2)\ (r))$ (d) $\pi_{A2}(\sigma(F_1 \vee F_2)\ (r))$

Solution: π select column and σ select row. As we can see from relational algebra expression, π select the final A1 attribute set. Row is selected which fulfills both the conditions of F1 and F2.
So, equivalent expression that can be written is: $\pi_{A1}(\sigma(F_1 \wedge F_2(r)))$

Ans. (a)

22. A prime attribute of a relation scheme R is an attribute that appears

(a) in all candidate keys of R.
(b) in some candidate key of R.
(c) in a foreign keys of R.
(d) only in the primary key of R.

Solution: An attribute that appears in candidate key is known as prime attribute.

Ans. (b)

23. In the following pairs of OSI protocol layer/sub-layer and its functionality, the INCORRECT pair is

(a) Network layer and Routing
(b) Data Link Layer and Bit synchronization
(c) Transport layer and End-to-end process communication
(d) Medium Access Control sub-layer and Channel sharing

Solution: The task of data link layer is to create frames. Bit synchronization is handled by physical layer.

Ans. (b)

24. A bit-stuffing based framing protocol uses an 8-bit delimiter pattern of 01111110. If the output bit-string after stuffing is 01111100101, then the input bit-string is

(a) 0111110100 (b) 0111110101
(c) 0111111101 (d) 0111111111

Solution: 01111110 is the delimiter pattern given. In output string 0111111**0**0101, 0 is stuffed. Input string is 01111110101.

Ans. (b)

25. Host A (on TCP/IP v4 network A) sends an IP datagram D to host B (also on TCP/IP V4 network B). Assume that no error occurred during the transmission of D. When D reaches B, which of the following IP header field(s) may be different from that of the original datagram D?

(i) TTL (ii) Checksum (iii) Fragment Offset

(a) (i) only (b) (i) and (ii) only
(c) (ii) and (iii) only (d) (i), (ii) and (iii)

Solution: TTL, checksum and fragment offset changes at every node.

Ans. (d)

Q. No. 26 – 55 Carry Two Marks Each

26. An IP router implementing Classless Inter-domain routing (CIDR) receives a packet with address 131.23.151.76. The router's routing table has the following entries:

Prefix	Output Interface Identifier
131.16.00/12	3
131.28.0.0/14	5
131.19.0.0/16	2
131.22.0.0/15	1

The identifier of the output interface on which this packet will be forwarded is _____.

Solution:
Calculating the network address:

First entry:

IP: 131. 00010111.151.76

Mask: 131. 00010000. 0.0 [12 n/w bits]

n/w address: 131.16.0.0 [Matched]

Second Entry:

IP: 131. 00010111.151.76

Mask: 131. 00011100. 0.0 [14 n/w bits]

n/w address: 131.20.0.0 [Not Matched]

Third Entry:

IP: 131. 00010111.151.76

Mask: 131. 00010011. 0.0 [16 n/w bits]

n/w address: 131.19.0.0 [Not Matched]

Fourth Entry:

IP: 131. 00010111.151.76

Mask: 131. 00010110. 0.0 [15 n/w bits]

n/w address: 131.20.0.0 [Matched]

according to longest mask, packet will be forwarded to interface 1.

Ans. (1)

27. Every host in an IPv4 network has a 1-second resolution real-time clock with battery backup. Each host needs to generate up to 1000 unique identifiers per second. Assume that each host has a globally unique IPv4 address. Design a 50-bit globally unique ID for this purpose. After what period (in seconds) will the identifiers generated by a host wrap around?

Solution:
50 bit unique id = 32 bit ip + 18 bit extra

With 18 bits = 2^{18} ids can be generated.

1000 ids take = 1 sec

2^{18} id will take = $2^{18}/1000 = 256$ sec.

Ans. (256)

28. An IP router with a Maximum Transmission Unit (MTU) of 1500 bytes has received an IP packet of size 4404 bytes with an IP header of length 20 bytes. The values of the relevant fields in the header of the third IP fragment generated by the router for this packet are

(a) MF bit: 0, Datagram Length: 1444; Offset: 370
(b) MF bit: 1, Datagram Length: 1424; Offset: 185
(c) MF bit: 1, Datagram Length: 1500; Offset: 370
(d) MF bit: 0, Datagram Length: 1424; Offset: 2960

Solution:
First fragment: 1480 + 20 MF = 1

Second fragment: 1480 + 20 MF = 1

Third fragment: 1444 + 20 MF = 0

Ans. (a)

29. Consider the transactions T1, T2, and T3 and the schedules S1 and S2 given below.

T1 : r1 (X) ; r1 (Z) ; w1 (X) ; w1 (Z)
T2 : r2 (X) ; r2 (Z) ; w2 (Z)
T3 : r3 (X) ; r3 (X) ; w3 (Y)

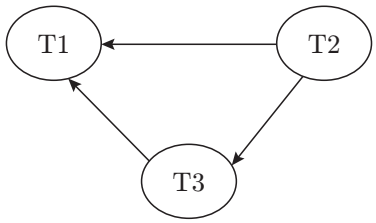
S1: r1(X); r3(Y); r3(X); r2(Y); r2(Z); w3(Y); w2(Z); r1(Z); w1(X); w1(Z)

S2: r1(X); r3(Y); r2(Y); r3(X); r1(Z); r2(Z); w3(Y); w1(X); w2(Z); w1(Z)

Which one of the following statements about the schedules is TRUE?

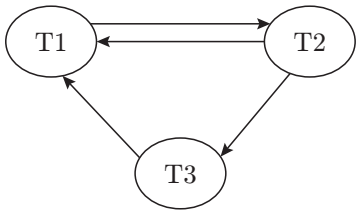
(a) Only S1 is conflict-serializable.
(b) Only S2 is conflict-serializable.
(c) Both S1 and S2 are conflict-serializable.
(d) Neither S1 nor S2 is conflict-serializable.

Solution: Precedence graph according to Schedule 1:



No cycle so S1 is conflict serializable.

Precedence graph according to Schedule 2:



Cycle exists, so S2 is not conflict serialzable.

Ans. (a)

30. Consider the relational schema given below, where eId of the relation dependent is a foreign key referring to empId of the relation employee. Assume that every employee has at least one associated dependent in the dependent relation.

employee (<u>empId</u>, empName, empAge)

dependent (<u>depId</u>, <u>eId</u>, depName, depAge)

Consider the following relational algebra query

$$\Pi_{\text{empId}}(\text{employee}) - \Pi_{\text{empId}}(\text{employee} \bowtie$$
$$(\text{empId} = \text{eID}) \wedge (\text{empAge} \leq \text{depAge})\ \text{dependent})$$

The above query evaluates to the set of empIds of employees whose age is greater than that of

(a) some dependent.
(b) all dependents.
(c) some of his/her dependents.
(d) all of his/her dependents.

Solution:

Part A – Part B

Part B selects those employee ids whose age is less than his dependents. When Part B is subtracted from all the employee ids, result will be set of those employee eids who are greater than all their dependents.

Ans. (d)

31. A system contains three programs and each requires three tape units for its operation. The minimum number of tape units which the system must have such that deadlocks never arise is _____.

Solution:

Exp:	Maximum	Allocate	Need	Available
P_1	−3	2	1	1
P_2	−3	2	1	
P_3	−3	2	1	

With the above given data, after allocating 2 units of tape to each process, with 1 available unit any of the 3 process can be satisfied in such a way, that No dead lock will be there.
So, answer is 7 tape units.
There are three processes, each require 3 resources. If we have 7 resources, we can allocate 2 to each process and 1 would be available. Any process can start executing.

Ans. (7)

32. An operating system uses shortest remaining time first scheduling algorithm for pre-emptive scheduling of processes. Consider the following set of processes with their arrival times and CPU burst times (in milliseconds):

Process	Arrival Time	Burst Time
P1	0	12
P2	2	4
P3	3	6
P4	8	5

The average waiting time (in milliseconds) of the processes is _____.

Solution:
According to SRJF

P1	P2	P3	P4	P1

0 2 6 12 17 27

Waiting time:

P1: 17 − 2 = 15

P2: 0

P3: 6 − 3 = 3

P4: 12 − 8 = 4

Average waiting time = 15 + 3 + 4 = 22/4 = 5.5 ms

Ans. (5.5)

33. Consider a paging hardware with a TLB. Assume that the entire page table and all the pages are in the physical memory. It takes 10 milliseconds to search the TLB and 80 milliseconds to access the physical memory. If the TLB hit ratio is 0.6, the effective memory access time (in milliseconds) is _____.

Solution: Effective memory access time = Cache hit × (TLB access + Main Memory) + Cache miss × (TLB access + 2 × Main Memory)

= 0.6 × (10 + 80) + 0.4 × (10 + 160)

=122 ms

Ans. (122)

34. Consider the basic block given below.

a = b + c
c = a + d
d = b + c
e = d − b
a = e + b

The minimum number of nodes and edges present in the DAG representation of the above basic block respectively are

(a) 6 and 6
(b) 8 and 10
(c) 9 and 12
(d) 4 and 4

Solution:
a = b + c
c = a + d
d = 2b + c + d
e = c
a = d

The maximum number of nodes and edges are (6, 6).

Ans. (a)

35. Which one of the following problems is undecidable?

(a) Deciding if a given context-free grammar is ambiguous.
(b) Deciding if a given string is generated by a given context-free grammar.
(c) Deciding if the language generated by a given context-free grammar is empty.
(d) Deciding if the language generated by a given context-free grammar is finite.

Solution: There is no algorithm to check ambiguity of CFG.

Ans. (a)

36. Consider the following languages over the alphabet = {0,1,c}

$$L_1 = \{0^n 1^n \mid n \geq 0\}$$

$$L_2 = \{wcw^r \mid w \in \{0,1\}^*\} \ \{\ \}$$

$$L_3 = \{ww^r \mid w \in \{0,1\}^*\}$$

Here w^r is the reverse of the string w. Which of these languages are deterministic context-free languages?

(a) None of the languages
(b) Only 1 L
(c) Only 1 L and 2 L
(d) All the three languages

Solution: L_1 and L_2 are accepted by DPDA, L_3 is accepted by NDPDA.

Ans. (c)

37. Suppose you want to move from 0 to 100 on the number line. In each step, you either move right by a unit distance or you take a shortcut. A shortcut is simply a pre-specified pair of integers i, j with $i < j$. Given a shortcut i, j if you are at position i on the number line, you may directly move to j. Suppose $T(k)$ denotes the smallest number of steps needed to move from k to 100. Suppose further that there is atmost 1 shortcut involving any number, and in particular from 9 there is a shortcut to 15. Let y and z be such that T(9) = 1 + min($T(y)$, $T(z)$). Then the value of the product yz is _____.

Solution:
T(9) = 1 + min(T(x), $T(y)$)

T(x) and T(y) can be two values: either move right or short cut

Moving right means: 10 and shortcut from 9 is 15

So product is 15 × 10 = 150.

Ans. (150)

38. Consider the decision problem 2CNFSAT defined as follows:

{ϕ|ϕ is a satisfiable propositional formula in CNF with at most two literal per clause}

For example, $\phi = (x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (x_2 \vee x_4)$ is a Boolean formula and it is in 2CNFSAT. The decision problem 2CNFSAT is

(a) NP-Complete.
(b) solvable in polynomial time by reduction to directed graph reachability.

(c) solvable in constant time since any input instance is satisfiable.
(d) NP-hard, but not NP-complete.

Solution: The decision problem 2CNFSAT is Solvable in polynomial time by reduction to directed graph reachability.

Ans. (b)

39. Suppose we have a balanced binary search tree T holding n numbers. We are given two numbers L and H and wish to sum up all the numbers in T that lie between L and H. Suppose there are m such numbers in T. If the tightest upper bound on the time to compute the sum is $O(n^a \log^b n + m^c \log^d n)$, the value of a + 10b + 100c + 1000d is _____.

Solution:
Time complexity is O (log n + m)

According to this, a = 0, b = 1, c = 1 and d = 0

Put these in given equation: a + 10b + 100c + 1000d = 0 + 10 + 100 + 0 = 110

Ans. (110)

40. Consider a hash table with 100 slots. Collisions are resolved using chaining. Assuming simple uniform hashing, what is the probability that the first 3 slots are unfilled after the first 3 insertions?

(a) $(97 \times 97 \times 97)/100^3$
(b) $(99 \times 98 \times 97)/100^3$
(c) $(97 \times 96 \times 95)/100^3$
(d) $(97 \times 96 \times 95)/(3! \times 100^3)$

Solution:

Using same index: $\dfrac{97 \times 97 \times 97}{100 \times 100 \times 100}$

Ans. (a)

41. Consider the pseudocode given below. The function Dosomething () takes as argument a pointer to the root of an arbitrary tree represented by the left-MostChild-rightSibling representation. Each node of the tree is of type treeNode.

```
typedef struct treeNode* treeptr;
Struct treeNode
{
Treeptr leftMostchild, rightSibiling;
};
Int Dosomething (treeptr tree)
{
int value =0;
if (tree ! = NULL) {
If (tree -> leftMostchild = = NULL)
else
value = Dosomething
(tree->leftMostchild);
value = value + Dosometing
(tree->rightsibiling);
}
return (value);
}
```

When the pointer to the root of a tree is passed as the argument to DoSomething, the value returned by the function corresponds to the

(a) number of internal nodes in the tree.
(b) height of the tree.
(c) number of nodes without a right sibling in the tree.
(d) number of leaf nodes in the tree.

Solution: Pseudocode recursively finds the leaf nodes and counts them.

Ans. (d)

42. Consider the C function given below. Assume that the array listA contains n (> 0) elements, sored in ascending order.

```
int ProcessArray (int * listA,
int x, int n)
{
int 1, j, k;
i = 0;
j = n - 1;
do {
k = (i + j) /2;
if (x < = listA [k])
j = k - 1;
If (listA [k] < = x)
i = k+1;
}while (1 < = j);
If (listA [k] = = x)
return (k) ;
else
return -1;
}
```

Which one of the following statements about the function **ProcessArray is CORRECT**?

(a) It will run into an infinite loop when x is not in listA.
(b) It is an implementation of binary search.
(c) It will always find the maximum element in listA.
(d) It will return – 1 even when x is present in listA.

Solution: Checking the logic of code clearly show that binary search has been implemented.

Ans. (b)

43. An instruction pipeline has five stages, namely, instruction fetch (IF), instruction decode and register fetch (ID/RF), instruction execution (EX), memory access (MEM), and register write back (WB) with stage latencies 1 ns, 2.2 ns, 2 ns, 1 ns, and 0.75 ns, respectively (ns stands for nanoseconds). To gain in terms of frequency, the designers have decided to split the ID/RF stage into three stages (ID, RF1, RF2) each of latency 2.2/3 ns. Also, the EX stage is split into two stages (EX1, EX2) each of latency 1 ns. The new design has a total of eight pipeline stages. A program has 20% branch instructions which execute in the EX stage and produce the next instruction pointer at the end of the EX stage in the old design and at the end of the EX2 stage in the new design. The IF stage stalls after fetching a branch instruction until the next instruction pointer is computed. All instructions other than the branch instruction have an average CPI of one in both the designs. The execution times of this program on the old and the new design are P and Q nanoseconds, respectively. The value of P/Q is _____.

Solution:

Old Design:

Stages = 5

Stall cycle = 2

Stall frequency = 20%

Maximum clock period = 2.2

Average access time (P) = $(0.8 + 0.6) \times 2.2 = 3.08$ ns

New Design:

Stages = 8

Stall cycle = 5

Stall frequency = 20%

Maximum clock period = 1

Average access time (Q) = $(0.8 + 0.12) \times 1 = 2$ ns

P/Q = 3.08/2 = 1.54

Ans. (1.54)

44. The memory access time is 1 nanosecond for a read operation with a hit in cache, 5 nanoseconds for a read operation with a miss in cache, 2 nanoseconds for a write operation with a hit in cache and 10 nanoseconds for a write operation with a miss in cache. Execution of a sequence of instructions involves 100 instruction fetch operations, 60 memory operand read operations and 40 memory operand write operations. The cache hit-ratio is 0.9. The average memory access time (in nanoseconds) in executing the sequence of instructions is _____.

Solution:

100 fetch operations: 90×1 ns $+ 10 \times 5 = 140$ ns

Memory read operations (60): $0.9 \times 60 \times 1 + 0.1 \times 60 \times 5 = 84$ ns

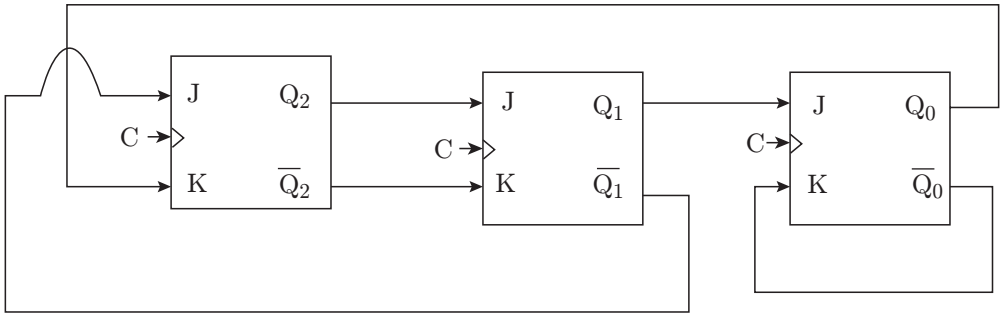
Memory write operations (40): $0.9 \times 40 \times 1 + 0.1 \times 40 \times 5 = 112$ ns

Average memory access time = 336/200 = 1.68

Ans. (1.68)

45. The above synchronous sequential circuit built using JK flip-flops is initialized with $Q_2Q_1Q_0 = 000$. The state sequence for this circuit for the next 3 clock cycles is

(a) 001, 010, 011
(b) 111, 110, 101
(c) 100, 110, 111
(d) 100, 011, 001



Solution:

Present State			Next State		
Q_2	Q_1	Q_0	$Q_2(J = Q_1', K = Q_0)$	$Q_1(J = Q_2, K = Q_2')$	$Q_0(J = Q_1, K = Q_0')$
0	0	0	1	0	0
1	0	0	1	1	0
1	1	0	1	1	1

Ans. (c)

46. With respect to the numerical evaluation of the definite integral, $K = \int_a^b x^2\,dx$, where a and b are given, which of the following statements is/are TRUE?

(I) The value of K obtained using the trapezoidal rule is always greater than or equal to the exact value of the definite integral.

(II) The value of K obtained using the Simpson's rule is always equal to the exact value of the definite integral.

(a) I only (b) II only
(c) Both I and II (d) Neither I nor II

Solution:

$$\int_0^1 x^2\,dx = \frac{x^3}{3} = \frac{1}{3} = 0.33333$$

As $N = 4$, we have

x	0	0.25	0.5	0.75	1
$y = x^2$	0	0.0625	0.25	0.5625	1
	y_0	y_1	y_2	y_3	y_4

Trapezoidal rule:

$$\int_0^1 x^2\,dx = \frac{0.25}{2}\left[(0+1) + 2(0.0625 + 0.25 + 0.5625)\right]$$

$$= 0.34375$$

Simpson's 1/3 rule:

$$\int_0^1 x^2\,dx = \frac{0.25}{2}\left[(0+1) + 2(0.25) + 4(0.0625 + 0.5625)\right]$$

$$= 0.3333$$

Ans. (c)

47. The value of the integral given below is

$$\int_0^{\pi} x^2 \cos x\,dx$$

(a) -2π (b) π (c) $-\pi$ (d) 2π

Solution:

$$\int_0^{\pi} x^2 \cos x\,dx = [x^2 \sin x - 2x(-\cos x) + 2(-\sin x)]_{0 \text{ to } \pi}$$
$$= -2\pi$$
$$= [x^2 \sin x - 2x(-\cos x) + 2(-\sin x)]_{0 \text{ to } \pi}$$
$$= -2\pi$$

Ans. (a)

48. Let S be a sample space and two mutually exclusive events A and B be such that $A \cup B = S$. If P(.) denotes the probability of the event, the maximum value of P(A)P(B) is _____.

Solution:
Maximum value of $P(A)\ P(B) = ?$

$P(A)P(B) = P(A)[1 - P(A)]$

Let $P(A) = x$

$f(x) = x - x^2$

$f'(x) = 1 - 2x = 0$

So, $x = 1/2$ and $f''(x) = -2$ and x is maximum at 1/2

$P(A)\ P(B) = 0.25$

49. Consider the set of all functions f :{0, 1, …, 2014} → {0, 1 ,…, 2014} such that $f(f(i)) = i$, for $0 \le i \le 2014$. Consider the following statements.

P. For each such function it must be the case that for every $i, f(i) = i$,
Q. For each such function it must be the case that for some $i, f(i) = i$,
R. Each such function must be onto.
Which one of the following is CORRECT?

(a) P, Q and R are true
(b) Only Q and R are true
(c) Only P and Q are true
(d) Only R is true

Solution: It is not necessary that for every $f(i) = i$, so statement P is incorrect.

Ans. (b)

50. There are two elements x,y in a group (G,*) such that every element in the group can be written as a product of some number of x's and y's in some order. It is known that

$$x * x = y * y = x * y * x * y = y * x * y * x = e$$

where e is the identity element. The maximum number of elements in such a group is _____.

Solution: Elements of this group are: {x, y, e, x*y}

Ans. (4)

51. If G is a forest with n vertices and k connected components, how many edges does G have?

(a) $\lceil n/k \rceil$ (b) $\lceil n/k \rceil$
(c) $n - k$ (d) $n - k + 1$

Solution: In forest each component is a tree. There are G_1, G_2, … G_k components. Each has $n - 1$ edges.

$$\sum_{i=1}^{k} n - i = n - k \text{ edges}$$

Ans. (c)

52. Let δ denote the minimum degree of a vertex in a graph. For all planar graphs on n vertices with $\delta \geq 3$, which one of the following is TRUE?

(a) In any planar embedding, the number of faces is at least $\frac{n}{2} + 2$
(b) In any planar embedding, the number of faces is less than $\frac{n}{2} + 2$
(c) There is a planar embedding in which the number of faces is less than $\frac{n}{2} + 2$
(d) There is a planar embedding in which the number of faces is at most $\frac{n}{\delta + 1}$

Solution:

Degree ≥ 3 is of n vertices.

Means $3n \leq 2e \rightarrow e \geq 3n/2$

We know that $e = n + r - 2$

$n + r - 2 \geq 3n/2$

or $r \geq n/2 + 2$

Ans. (a)

53. The CORECT formula for the sentence, "not all rainy days are cold" is

(a) ∀d (Rainy(d) ∧ ~Cold(d))
(b) ∀d (~Rainy(d) → Cold(d))
(c) ∃d (~Rainy(d) → Cold(d))
(d) ∃d (Rainy(d) ∧ ~Cold(d))

Solution:

"Not all rainy days are cold"

$$\sim \forall d\big(\text{rainy}(d) \rightarrow \text{cold}(d)\big) \rightarrow$$
$$\exists d\big(\text{rainy}(d) \sim \text{cold}(d)\big)$$

Ans. (d)

54. Consider the following relational schema:

`Employee (empId, empName, empDept)`

`Customer (custId,custName, salesRepId, rating)`

SalesRepId is a foreign key referring to empId of the employee relation. Assume that each employee makes a sale to at least one customer. What does the following query return?

```
SELECT empName
FROM employee E
WHERE NOT EXISTS (SELECT custId
                  FROM customer C
                  WHERE C. salesRepId = E. empId
AND C. rating < > 'GOOD')
```

(a) Names of all the employees with at least one of their customers having a 'GOOD' rating.
(b) Names of all the employees with at most one of their customers having a 'GOOD' rating.
(c) Names of all the employees with none of their customers having a 'GOOD' rating.
(d) Names of all the employees with all their customers having a 'GOOD' rating.

Solution: Inner query will result in the customer ids of those whose rating is not good. Outer query results in those employee names whose customer ids are not in the inner query. So, the result will display employees with good rating.

Ans. (d)

55. Let $\oplus$ denote the Exclusive OR (XOR) operation. Let '1' and '0' denote the binary constants. Consider the following Boolean expression for F over two variables P and Q.

$F(P,Q) = ((1 \oplus P) \oplus (P \oplus Q)) \oplus ((P \oplus Q) \oplus (Q \oplus 0))$

The equivalent expression for F is

(a) $P + Q$ (b) $\overline{P + Q}$
(c) $P \oplus Q$ (d) $\overline{P \oplus Q}$

Solution:

$$(P' \oplus (P \oplus Q)) \oplus ((P \oplus Q) \oplus Q)$$

$$(P' \oplus (PQ' + P'Q)) \oplus ((PQ' + P'Q) \oplus Q)$$

$$(P'(PQ' + P'Q) + P(PQ' + P'Q)') \oplus (Q(PQ' + P'Q)' + Q'(PQ' + P'Q))$$

$$Q' \oplus P$$

$$(P \oplus Q)'$$

Ans. (d)

SOLVED GATE (CS) 2015

SET 1

(Engineering Mathematics and Technical Section)

Q. NO. 1 — 25 Carry One Mark Each

1. If $g(x) = 1 - x$ and $h(x) = \dfrac{x}{x-1}$, then $\dfrac{g(h(x))}{h(g(x))}$ is

(a) $\dfrac{h(x)}{g(x)}$ (b) $\dfrac{-1}{x}$

(c) $\dfrac{g(x)}{h(x)}$ (d) $\dfrac{x}{(1-x)^2}$

Solution: We have $g(x) = 1 - x$, $h(x) = \dfrac{x}{x-1}$,

$$g(h(x)) = g\left(\frac{x}{x-1}\right) = 1 - \frac{x}{x-1} = \frac{1}{1-x}$$

$$h(g(x)) = h(1-x) = \frac{x-1}{x}$$

$$\frac{g(h(x))}{h(g(x))} = \frac{x}{(x-1)(1-x)} = \frac{h(x)}{g(x)}$$

Hence, (a) is the correct answer.

Ans. (a)

2. $\lim_{x\to\infty} x^{1/x}$ is

(a) ∞ (b) 0 (c) 1 (d) Not defined

Solution: In $\lim_{x\to\infty} x^{1/x}$ let us assume $y = \lim_{x\to\infty} x^{1/x}$

Taking logarithm on both sides,

$\log y = \log\left(\lim_{x\to\infty} x^{1/x}\right) = \lim_{x\to\infty} (\log x)/x$

RHS is ∞/∞ form, therefore L-Hopital rule can be applied

Thus, $\log y = \lim_{x\to\infty} 1/x = 0$. Hence, $y = 1$.

Ans. (c)

3. Match the following:

P: Prim's algorithm for minimum spanning tree	(i) Backtracking
Q: Floyd–Warshal algorithm for all pairs shortest paths	(ii) Greedy method
R: Merge sort	(iii) Dynamic programming
S: Hamiltonian circuit	(iv) Divide and conquer

(a) P-iii, Q-ii, R-iv, S-i
(b) P-i, Q-ii, R-iv, S-iii
(c) P-ii, Q-iii, R-iv, S-i
(d) P-ii, Q-i, R-iii, S-iv

Solution: Merge sort firstly divides the list and then merges it to sort. Hence, it uses divide and conquer strategy.

All pair shortest path algorithm uses the concept of overlapping while computing the shortest path. Hence, it uses dynamic programming.

Hamiltonian circuit means covering all the vertices and while doing so backtracking is required.

Prim's algorithm for minimum spanning tree is greedy method.

Ans. (c)

4. Which one of the following is the recurrence equation for the worst case time complexity of the Quicksort algorithm for sorting n (≥ 2) numbers? In the recurrence equations given in the options below, c is a constant.

(a) $T(n) = 2T(n/2) + cn$
(b) $T(n) = T(n-1) + T(1) + cn$

(c) $T(n) = 2T(n-1) + cn$
(d) $T(n) = T(n/2) + cn$

Solution: Quicksort's worst case is when the list is already sorted or when it is in reverse order. In both cases, the pivot element is already in sorted order and list will always be divided into two lists containing 1 and $n-1$ elements.

Thus, the recurrence relation formed is as follows:

$$T(n) = T(n-1) + T(1) + cn$$

Ans. (b)

5. The height of a tree is the length of the longest root-to-leaf path in it. The maximum and minimum number of nodes in a binary tree of height 5 are

(a) 63 and 6, respectively
(b) 64 and 5, respectively
(c) 32 and 6, respectively
(d) 31 and 5, respectively

Solution: If height of the binary tree is h, then

Minimum number of nodes in a tree $= h + 1$

Maximum number of nodes $= 2^{h+1} - 1$

Here, $h = 5$

Therefore, minimum nodes $= 5 + 1 = 6$

maximum nodes $= 2^6 - 1 = 63$

Hence, answer is 63 and 6, respectively.

Ans. (a)

6. Match the following:

P: Condition coverage	(I) Black-box testing
Q: Equivalence class partitioning	(II) System testing
R: Volume testing	(III) White-box testing
S: Alpha testing	(IV) Performance testing

(a) P-II, Q-III, R-I, S-IV
(b) P-III, Q-IV, R-II, S-I
(c) P-III, Q-I, R-IV, S-II
(d) P-III, Q-I, R-II, S-IV

Solution: Condition coverage means covering all the possibilities of both true and false conditions. White box testing covers the internal structure of software.

Equivalence class testing is dividing the input of software into small partitions of equivalent data, from which various test cases are derived. This is black box testing. Volume or stress testing is a type of system testing.

Alpha testing is done by the user/customer for checking its performance.

Ans. (d)

7. Which of the following is/are correct inorder traversal sequence(s) of binary search tree(s)?

I. 3, 5, 7, 8, 15, 19, 25
II. 5, 8, 9, 12, 10, 15, 25
III. 2, 7, 10, 8, 14, 16, 20
IV. 4, 6, 7, 9, 18, 20, 25

(a) I and IV only
(b) II and III only
(c) II and IV only
(d) II only

Solution: Inorder traversal of tree is the sorted order of nodes of the tree. Check the given sequences; the one which is in sorted order is the correct inorder traversal.

Ans. (a)

8. Which one of the following is TRUE at any valid state in shift – reduce parsing?

(a) Viable prefixes appear only at the bottom of the stack and not inside
(b) Viable prefixes appear only at the top of the stack and not inside
(c) The stack contains only a set of viable prefixes
(d) The stack never contains viable prefixes

Solution: The set of prefixes of right sentential forms that can appear on the stack of a shift-reduce parser are called viable prefixes. They come at the top of stack.

Ans. (b)

9. Which one of the following is NOT equivalent to $p \leftrightarrow q$?

(a) $(\rceil p \vee q) \wedge (p \vee \rceil q)$
(b) $(\rceil p \vee q) \wedge (q \rightarrow p)$
(c) $(\rceil p \wedge q) \vee (p \wedge \rceil q)$
(d) $(\rceil p \wedge \rceil q) \vee (p \wedge q)$

Solution: Drawing the truth table of all the options, we can find the equivalency of expressions.

Ans. (c)

10. For a set A, the power set of A is denoted by 2^A. If $A = \{5, \{6\}, \{7\}\}$, which of the following options are TRUE?

I. $\Phi \in 2^A$
II. $\Phi \subseteq 2^A$
III. $\{5, \{6\}\} \in 2^A$
IV. $\{5\{6\}\} \subseteq 2^A$

(a) I and III only
(b) II and III only
(c) I, II and III only
(d) I, II and IV only

Solution: A: The given set and 2^A: Power set of A, that is, set of subsets of A.

Φ is the subset of every set.

Therefore, statements I and II are true.

Since 5 does not belong to 2^A. Hence, statement IV is not true.

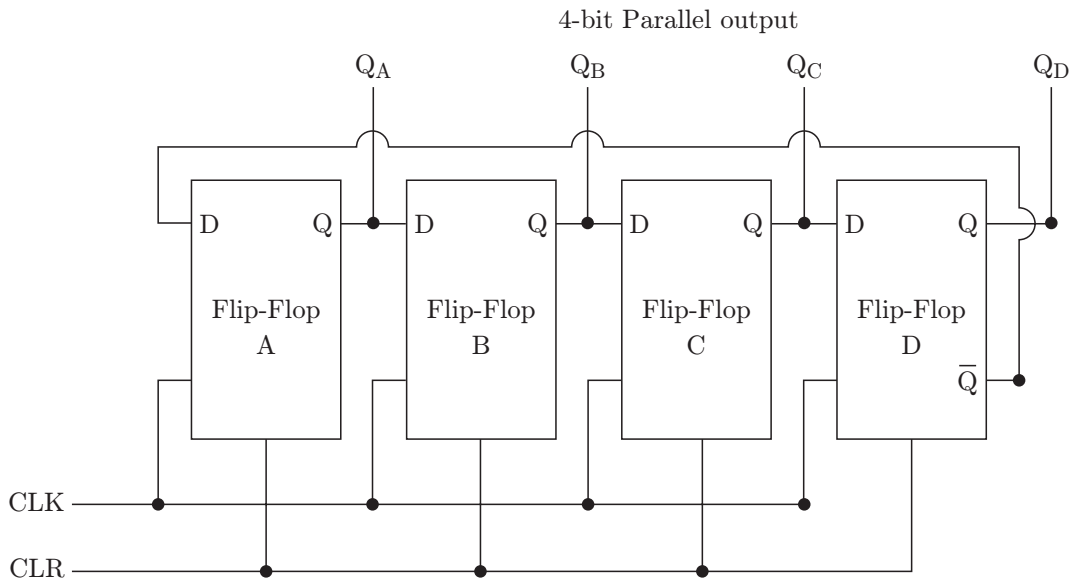
Ans. (c)

11. Consider a 4-bit Johnson counter with an initial value of 0000. The counting sequence of this counter is

(a) 0, 1, 3, 7, 15, 14, 12, 8, 0
(b) 0, 1, 3, 5, 7, 9, 11, 13, 15, 0
(c) 0, 2, 4, 6, 8, 10, 12, 14, 0
(d) 0, 8, 12, 14, 15, 7, 3, 1, 0

Solution: This inversion of Q is feedback to input D of flip-flop A and this causes the counter to "count" in a unique way. Counting Pattern is as follows:

Clock Pulse Number	Flip-Flop A	Flip-Flop B	Flip-Flop C	Flip-Flop D
Initial	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1



Ans. (d)

12. For computers based on three-address instruction formats, each address field can be used to specify which of the following.

S1: A memory operand
S2: A processor register
S3: An implied accumulator register

(a) Either S1 or S2 (b) Either S2 or S3
(c) Only S2 and S3 (d) All of S1, S2 and S3

Solution: Computer with three-address instruction format can use each address field to specify either processor register or memory operand.

Ans. (a)

13. Suppose two hosts use a TCP connection to transfer a large file. Which of the following statements is/are FALSE with respect to the TCP connection?

I. If the sequence number of a segment is m, then the sequence number of the subsequent segment is always $m + 1$.
II. If the estimated round trip time at any given point of time is t sec the value of the retransmission timeout is always set to greater than or equal to t sec.
III. The size of the advertised window never changes during the course of the TCP connection.
IV. The number of unacknowledged bytes at the sender is always less than or equal to the advertised window.

(a) III only (b) I and III only
(c) I and IV only (d) II and IV only

Solution: It is not necessary to have the subsequent sequence number as $m + 1$.

Size of window is not constant during TCP connection. It changes once the threshold value of congestion window reaches. So, statement III is false.

Ans. (b)

14. Suppose that everyone in a group of N people wants to communicate secretly with the $N-1$ others using symmetric key cryptographic system. The communication between any two persons should not be decodable by the others in the group. The number of keys required in the system as a whole to satisfy the confidentiality requirement is

(a) $2N$ (b) $N(N-1)$
(c) $N(N-1)/2$ (d) $(N-1)^2$

Solution: In symmetric key cryptography, for n users the number of possible keys is $n(n-1)/2$.

Ans. (c)

15. Which of the following statements is/are FALSE?

I. XML overcomes the limitations in HTML to support a structured way of organizing content.
II. XML specification is not case sensitive while HTML specification is case sensitive.
III. XML supports user defined tags while HTML uses pre-defined tags.
IV. XML tags need not be closed while HTML tags must be closed

(a) II only (b) I only
(c) II and IV only (d) III and IV only

Solution: HTML is case insensitive, while XML is case sensitive. Therefore, statement II is incorrect.

XML tags must be closed. Hence, statement IV is also false.

Ans. (c)

16. Which one of the following fields of an IP header is NOT modified by a typical IP router?

(a) Checksum (b) Source address
(c) Time to Live (TTL) (d) Length

Solution: Source address is the only thing that remains fixed and cannot be varied.

Ans. (b)

17. In one of the pairs of protocols given below, both the protocols can use multiple TCP connections between the same client and the server. Which one is that?

(a) HTTP, FTP (b) HTTP, TELNET
(c) FTP, SMTP (d) HTTP, SMTP

Solution: HTTP can use multiple TCP connections for each resource.

FTP uses Telnet protocol for control information on a TCP connection and another TCP connection for data exchange.

Ans. (a)

18. For any two languages L_1 and L_2 such that L_1 is context-free and L_2 is recursively enumerable but not recursive, which of the following is/are necessarily true?

I. $\overline{L_1}$ (complement of L_1) is recursive
II. $\overline{L_2}$ (complement of L_2) is recursive
III. $\overline{L_1}$ is context-free
IV. $\overline{L_1} \cup L_2$ is recursively enumerable

(a) I only (b) III only
(c) III and IV only (d) I and IV only

Solution: Context-free language is also recursive and recursive is closed under complement. Therefore, statement I is true.

Complement of recursive enumerable is not recursive. Hence, statement II is false.

Context-free language is not closed under complement. Hence, statement III is false.

$\sim L_1$ is recursive and hence also recursive enumerable. L_2 is recursive enumerable. RE languages are closed under union. Hence, statement IV is true.

Ans. (d)

19. Consider a system with byte-addressable memory, 32-bit logical addresses, 4 kilobyte page size and page table entries of 4 bytes each. The size of the page table in the system in megabytes is _____.

Solution: Logical address space = 32 bits

Page size = 4 KB = 2^{12}

Each page table entry size = 4 B

Number of pages = $2^{32}/2^{12} = 2^{20}$

Page table size = (Number of pages) × (Page table entry size) = 4 B × 2^{20} = 4 MB

Ans. (4)

20. The following two functions P1 and P2 that share a variable B with an initial value of 2 execute concurrently.

P1()	P2()
{	{
C = B - 1;	D = 2 * B;
B = 2 * C;	B = D - 1;
}	}

The number of distinct values that B can possible take after the execution is ____ .

Solution: Initial value of B = 2

There are four possibilities in which two processes can be executed:

P1 is executed first and then P2:

After this execution B = 3

P2 is executed first and then P1

After this execution B = 4

P1 is preempted by P2

After this execution B = 2

P2 is preempted by P1

After this execution B = 3

Therefore, three possible values of B = 2, 3, 4

Ans. (3)

21. SELECT operation in SQL is equivalent to

(a) the selection operation in relational algebra
(b) the selection operation in relational algebra, except that SELECT in SQL retains duplicates
(c) the projection operation in relational algebra
(d) the projection operation in relational algebra, except that SELECT in SQL retains duplicates

Solution: In SQL, the SELECT operation is used for selecting the columns. In a similar way, relational algebra uses project to select the columns. But SQL retains duplicates as well which is not so in relational algebra.

Ans. (d)

22. A file is organized so that the ordering of data records is the same as or close to the ordering of data entries in some index. Then that index is called

(a) dense. (b) sparse.
(c) clustered. (d) unclustered.

Solution: A clustered index is a type of index where the table records are physically re-ordered to match the index.

Ans. (c)

23. In the LU decomposition of the matrix $\begin{bmatrix} 2 & 2 \\ 4 & 9 \end{bmatrix}$, if the diagonal elements of U are both 1, then the lower diagonal entry l_{22} of L is _____.

Solution: In LU decomposition, we have

$$\begin{bmatrix} 2 & 2 \\ 4 & 9 \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} 1 & u_{12} \\ 0 & 1 \end{bmatrix}$$

Solving the above matrix, we have $l_{11} = 2$. Now,

$$l_{11}u_{12} = 2 \Rightarrow u_{12} = 1$$

Also, $l_{21} = 4$. So, $l_{21}u_{12} + l_{22} = 9 \Rightarrow l_{22} = 5$

Ans. (5)

24. The output of the following C program is _____.

```
void f1(int a, int b)
{
int c;
c=a; a=b; b=c;
}
void f2(int *a, int *b)
{
int c;
c=*a; *a=*b; *b=c;
}
int main()
{
int a=4, b=5, c=6;
f1(a, b);
f2(&b, &c);
printf("%d", c-a-b);
}
```

Solution: f1 (a, b): This is call by value. Values will not change.

f2 (&b, &c): This is call by reference. This swaps values of b and c.

Now $a = 4$, $b = 6$, $c = 5$

printf will print c − a − b = 5 − 6 − 4 = −5

Ans. (−5)

25. What are the worst-case complexities of insertion and deletion of a key in a binary search tree?

(a) $\Theta(\log n)$ for both insertion and deletion
(b) $\Theta(n)$ for both insertion and deletion
(c) $\Theta(n)$ for insertion and $\Theta(\log n)$ for deletion
(d) $\Theta(\log n)$ for insertion and $\Theta(n)$ for deletion

Solution: Worst case for insertion and deletion could be if we have to traverse all the nodes. In that case, the complexity is $\Theta(n)$ in both.

Ans. (b)

Q. NO. 26 — 55 Carry Two Marks Each

26. Suppose that the stop-and-wait protocol is used on a link with a bit rate of 64 kilobits per second and 20 milliseconds propagation delay. Assume that the transmission time for the acknowledgement and the processing time at nodes are negligible. Then the minimum frame size in bytes to achieve a link utilization of at least 50% is _____.

Solution: Minimum frame size means 100% utilization.

$$\text{RTT} = 20 \times 2 = 40 \text{ ms}$$

$$\text{Bit rate} = 64 \text{ Kbps}$$

64×10^3 bytes $\Rightarrow$ 1 sec

40 ms $\Rightarrow 64 \times 10^3 \times 40 \times 10^{-3} = 2560$ bits

Bytes $\Rightarrow \dfrac{2560}{8} = 320$ bytes

Hence, to achieve 50% utilization, bytes = 160

Ans. (160)

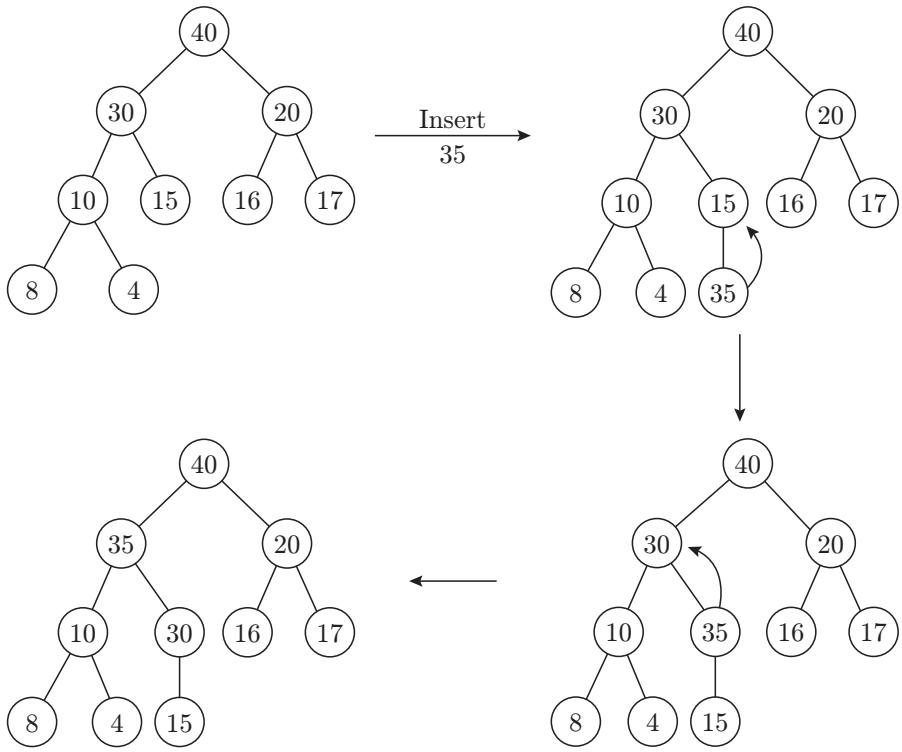
27. Consider a max heap, represented by the array: 40, 30, 20, 10, 15, 16, 17, 8, 4.

Array Index	1	2	3	4	5	6	7	8	9
Value	40	30	20	10	15	16	17	8	4

Now consider that a value 35 is inserted into this heap. After insertion, the new heap is

(a) 40, 30, 20, 10, 15, 16, 17, 8, 4, 35
(b) 40, 35, 20, 10, 30, 16, 17, 8, 4, 15
(c) 40, 30, 20, 10, 35, 16, 17, 8, 4, 15
(d) 40, 35, 20, 10, 15, 16, 17, 8, 4, 30

Solution:



40, 35, 20, 10, 30, 16, 17, 8, 4, 15

Ans. (b)

28. Consider the following C program segment.

```
while(first <= last)
{
if(array[middle] < search)
first = middle + 1;
else if (array[middle] = = search)
found = TRUE;
else last = middle - 1;
middle = (first + last)/2;
}
if (first > last) notPresent = TRUE;
```

The cyclomatic complexity of the program segment is ____.

Solution: Cyclomatic complexity = Number of predicate nodes + 1

Predicate node: Decision making node, that is, wherever we check for some condition.

Here in our code, the number of predicate nodes = 4

Therefore, cyclomatic complexity = 4 + 1 = 5.

Ans. (5)

29. Consider a LAN with four nodes S_1, S_2, S_3 and S_4. Time is divided into fixed-size slots, and a node can begin its transmission only at the beginning of a slot. A collision is said to have occurred if more than one node transmit in the same slot. The probability of generation of a frame in a time slot by S_1, S_2, S_3 and S_4 are 0.1, 0.2, 0.3 and 0.4, respectively. The probability of sending a frame in the first slot without any collision by any of these four stations is ____.

Solution:

$$P = P(S_1)\ P(\sim S_2)\ P(\sim S_3)\ P(\sim S_4) + P(\sim S_1)\ P(S_2)\ P(\sim S_3)\ P(\sim S_4) + P(\sim S_1)\ P(\sim S_2)\ P(S_3)\ P(\sim S_4) + P(\sim S_1)\ P(\sim S_2)\ P(\sim S_3)\ P(S_4)$$

$= 0.1 \times 0.8 \times 0.7 \times 0.6 + 0.9 \times 0.2 \times 0.7 \times 0.6 + 0.9 \times 0.8 \times 0.3 \times 0.6 + 0.9 \times 0.8 \times 0.7 \times 0.4$

$= 0.4404$

Ans. (0.4404)

30. The binary operator $\neq$ is defined by the following truth table.

p	q	p ≠ q
0	0	0
0	1	1
1	0	1
1	1	0

Which one of the following is true about the binary operator $\neq$?

(a) Both commutative and associative
(b) Commutative but not associative
(c) Not commutative but associative
(d) Neither commutative nor associative

Solution: Looking at the truth table, we analyze that this is the same as XOR operation. XOR is both commutative (i.e., x*y = y*x) and associative (i.e., a*(b*c) = (a*b)*c).

Ans. (a)

31. $\sum_{x=1}^{99} \frac{1}{x(x+1)} =$ _____.

Solution:

$$\sum_{x=1}^{99} \frac{1}{x(x+1)} = \frac{1}{1\times 2} + \frac{1}{2\times 3} + \frac{1}{3\times 4} + \cdots + \frac{1}{99\times 100}$$

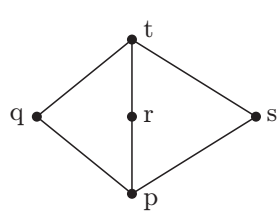
$$= \frac{2-1}{1\times 2} + \frac{3-2}{2\times 3} + \cdots + \frac{100-99}{99\times 100}$$

$$= 1 - \frac{1}{2} + \frac{1}{2} - \frac{1}{3} + \frac{1}{3} - \cdots + \frac{1}{99} - \frac{1}{100}$$

$$= 1 - \frac{1}{100} = 0.99$$

Ans. (0.99)

32. Suppose $\mathcal{L} = \{p,q,r,s,t\}$ is a lattice represented by the following Hasse diagram:



For any x, $y \in \mathcal{L}$, not necessarily distinct, $x \vee y$ and $x \wedge y$ are join and meet of x, y, respectively. Let $\mathcal{L}^3 = \{(x,y,z)\colon x,y,z \in \mathcal{L}\}$ be the set of all ordered triplets of the elements of $\mathcal{L}$. Let p_r be the probability that an element $(x, y, z) \in \mathcal{L}^3$ chosen equiprobably satisfies $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$. Then

(a) $p_r = 0$
(b) $p_r = 1$
(c) $0 < p_r \leq \frac{1}{5}$
(d) $\frac{1}{5} < p_r < 1$

Solution: Number of elements in $\mathcal{L}^3$

= Number of ways to choose 3 elements from 5

$= 5 \times 5 \times 5 = 125$ ways

Now, when we take $x = t$, then also the given condition is satisfied for any y and z.

Number of ways y and z can be taken $= 5 \times 5 = 25$ ways.

If we take $x = p$, $y = p$, $z = r$, the given condition is satisfied. So, $p_r > (25 / 125) > 1/5$

For $x = t$, $y = r$, $z = p$, the given condition is not satisfied. So, $p_r \neq 1$.

So, option (d) is correct.

Ans. (d)

33. Consider the operations

$f(X,Y,Z) = X'YZ + XY' + Y'Z'$ and

$g(X,Y,Z) = X'YZ + X'YZ' + XY.$

Which one of the following is correct?

(a) Both $\{f\}$ and $\{g\}$ are functionally complete
(b) Only $\{f\}$ is functionally complete
(c) Only $\{g\}$ is functionally complete
(d) Neither $\{f\}$ nor $\{g\}$ is functionally complete

Solution: A system of Boolean functions is functionally complete iff function satisfies following properties:

- at least one function that does not preserve zero
- at least one function that does not preserve one
- at least one function that is not linear
- at least one function that is not monotone
- at least one function that is not self-dual

$\{f\}$ satisfies all the above properties.

Ans. (b)

34. Let G be a connected planar graph with 10 vertices. If the number of edges on each face is three, then the number of edges in G is _____.

Solution: Using formula: $|R| + |V| = |E| + 2$

$3|R| = 2|E|$

$|R| = (2/3)|E|$

$$(2/3)|E| + |V| = |E| + 2 \mid |E| - (2/3)|E| = 8$$

$$|E| = 24$$

Ans. (24)

35. Let a_n represent the number of bit strings of length n containing two consecutive 1's. What is the recurrence relation for a_n?

(a) $a_{n-2} + a_{n-1} + 2^{n-2}$
(b) $a_{n-2} + 2a_{n-1} + 2^{n-2}$
(c) $2a_{n-2} + a_{n-1} + 2^{n-2}$
(d) $2a_{n-2} + 2a_{n-1} + 2^{n-2}$

Solution: Suppose $n = 3$, possible strings are $\Rightarrow$ 011, 110, 111

For $n = 2 \Rightarrow 11$ is the only possible string

$a_3 = a_1 + a_2 + 2^1 = 0 + 1 + 2 = 3 \Rightarrow$ True

$a_3 = a_1 + 2a_2 + 2^1 = 0 + 2 + 2 = 4 \Rightarrow$ False

$a_3 = 2a_1 + a_2 + 2^1 = 0 + 1 + 2 = 3 \Rightarrow$ True

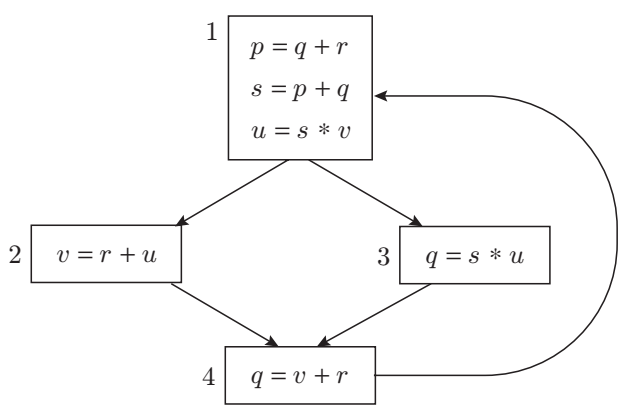
$a_3 = 2a_1 + 2a_2 + 2^1 = 0 + 2 + 2 = 4 \Rightarrow$ False

Now for options (a) and (c), check by considering $n = 4$. The answer comes out to be (a).

Ans. (a)

36. A variable x is said to be live at a statement S_i in a program if the following three conditions hold simultaneously:

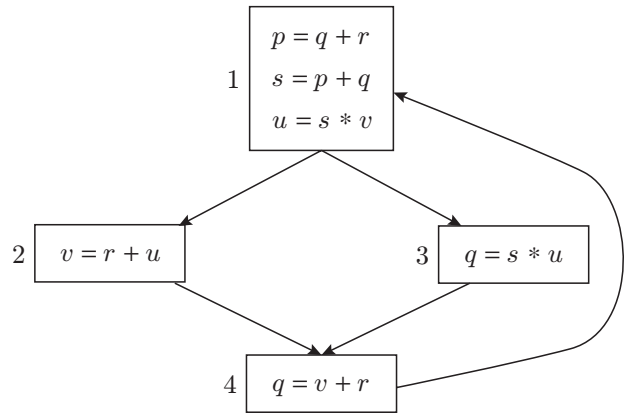
I. There exists a statement S_j that uses x
II. There is a path from S_i to S_j in the flow graph corresponding to the program
III. The path has no intervening assignment to x including at S_i and S_j



The variables which are live at the statement in basic block 2 and at the statement in basic block 3 of the above control flow graph are

(a) p, s, u
(b) r, s, u
(c) r, u
(d) q, v

Solution:



Checking the three conditions given in question,

r is live at block2 since r is used in block4 and has a path from block2 to block4 in the control flow graph. Also there isn't any modification of r in the path. Hence, r is live at block2. Similarly, check for u.

Ans. (c)

37. The least number of temporary variables required to create a three-address code in static single assignment form for the expression $q+r/3+s-t*5+u*v/w$ is _____.

Solution: We will need a temporary variable for storing the result of each binary operation.

q + r / 3 + s – t * 5 + u * v/w
t1 = r/3;
t2 = t*5;
t3 = u*v;
t4 = t3/w;
t5 = q + t1;
t6 = t5 + s;
t7 = t5 – t2;
t8 = t7 + t4
So, a total of 8 temporary variables are required.

Ans. (8)

38. Consider an Entity-Relationship (ER) model in which entity sets E_1 and E_2 are connected by an m:n relationship R_{12}. E_1 and E_3 are connected by a 1: n (1 on the side of E_1 and n on the side of E_3) relationship R_{13}.

E_1 has two single-valued attributes a_{11} and a_{12} of which a_{11} is the key attribute. E_2 has two single-valued attributes a_{21} and a_{22} of which is the key attribute. E_3 has two single-valued attributes a_{31} and a_{32} of which a_{31} is the key attribute. The relationships do not have any attributes.

If a relational model is the derived from the above ER model, then the minimum number of relations that would be generated if all the relations are in 3NF is _____.

Solution: The relations are as shown:

E_1: <a_{11}, a_{12}>

E_2: <a_{21}, a_{22}>

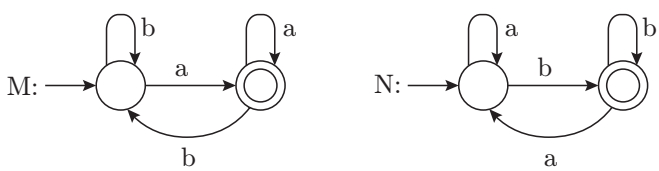
E_3: <a_{31}, a_{32}, a_{11}>

E_1–E_3: <a_{11}, a_{21}>for m:n relationship $E_1 - E_2$

We cannot combine any relation here as it will give rise to partial functional dependency and thus violate 3NF.

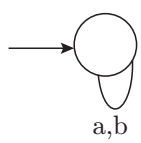
Ans. (4)

39.



Consider the DFAs M and N given above. The number of states in a minimal DFA that accepts the languages L(M) ∩ L(N) is _____.

Solution: M accepts the strings ending with 'a'. N accepts the strings ending with 'b'. Thus, their intersection should accept an empty string.

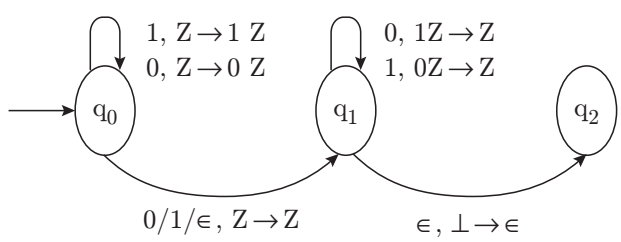


Ans. (1)

40. Consider the NPDA

$\langle Q = \{q_0, q_1, q_2\}, \Sigma = \{0, 1\}, \Gamma = \{0, 1, \Gamma\}\perp \delta, q_0, \perp, F = \{q_2\}\rangle$,

where (as per usual convention) Q is the set of states, Σ is the input alphabet, $\delta$is the stack alphabet, $\delta$is the state transition function, q_0 is the initial state, ⊥ is the initial stack symbol, and F is the set of accepting states. The state transition is as follows:



Which one of the following sequences must follow the string 101100 so that the overall string is accepted by the automaton?

(a) 10110 (b) 10010
(c) 01010 (d) 01001

Solution: This PDA is accepts all strings of the form x0x' or x1x', where x' is the reverse of the 1's complement of x. The given string 101100 has 6 letters and we are given 5 letter strings. So, x0 is done, with x = 10110. Now, the stack contains 10110 and remaining string should be opposite and reverse to it, so that the stack becomes empty. So, 1's complement of x = 01001 and x' = 10010.

Ans. (b)

41. Let G = (V, E) be a simple undirected graph, and s be a particular vertex in it called the source. For x ∈ V, let d(x) denote the shortest distance in G from s to x. A breadth first search (BFS) is performed starting at *s*. Let *T* be resultant BFS tree. If (u, v) is an edge of *G* that is not in *T*, then which one of the following CANNOT be the value of $d(u) - d(v)$?

(a) –1 (b) 0 (c) 1 (d) 2

Solution: In BFS traversal of given graph, we can either visit node *u* first or node *v*. Let us assume node u if visited first. Then all neighbors of node u will be in the queue to be visited. Since *v* is also a neighbor of u and v is not visited before, $d(v)$ will become $d(u) + 1$.

So, $d(u) - d(v) = 2$ is not possible.

Ans. (d)

42. Consider a uniprocessor system executing three tasks T_1, T_2 and T_3, each of which is composed of an infinite sequence of jobs (or instances) which arrive periodically at intervals of 3, 7 and 20 milliseconds, respectively. The priority of each task is the inverse of its period, and the available tasks are scheduled in order of priority, with the highest priority task scheduled first. Each instance of T_1, T_2 and T_3 requires an execution time of 1, 2 and 4 milliseconds, respectively. Given that all tasks initially arrive at the beginning of the 1st millisecond and task preemptions are allowed, the first instance of T_3 completes its execution at the end of _____ milliseconds.

Solution:

T1	T2	T1	T3	T1	T3	T2	T1	T2	T3

1 2 4 5 7 8 9 10 11 12 13

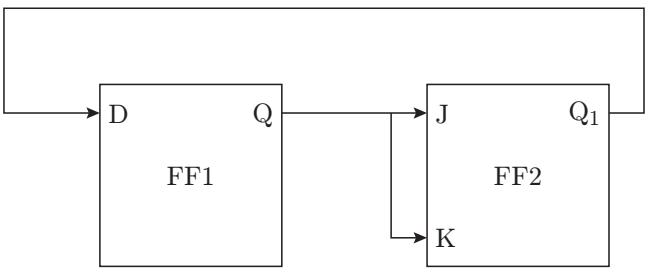
Ans. (13)

43. A positive edge-triggered D flip-flop is connected to a positive edge-triggered JK flip-flop as follows. The *Q* output of the D flip-flop is connected to both

the J and K inputs of the JK flip-flop, while the Q output of the JK flip-flop is connected to the input of the D flip-flop is connected to both the J and K inputs of the JK flip-flop, while the Q output of the JK flip-flop is connected to the input of the D flip-flop. Initially, the output of the D flip-flop is set to logic one and the output of the JK flip-flop is cleared. Which one of the following is the bit sequence(including the initial state) generated at the Q output of the JK flip-flop when the flip-flops are connected to a free-running common clock? Assume that J = K = 1 is the toggle mode and J = K = 0 is the state-holding mode of the JK flip-flop. Both the flip-flops have non-zero propagation delays.

(a) 0110110….. (b) 0100100….
(c) 011101110…. (d) 011001100…

Solution:



Clock	Q	Q_1
0 (initial)	1	0
1	0	1
2	1	1
3	1	0
4	0	1
5	1	1
6	1	0
7	0	1

Ans. (a)

44. Consider a disk pack with a seek time of 4 milliseconds and rotational speed of 10000 rotations per minute (RPM). It has 60 sectors per track and each sector can store 512 bytes of data. Consider a file stored in the disk. The file contains 2000 sectors. Assume that every sector access necessitates a seek, and the average rotational latency for accessing each sector is half of the time for one complete rotation. The total time (in milliseconds) needed to read the entire file is _____.

Solution: Seek time = 4 ms

RPM = 10000 rotations

that is, 60 seconds = 10000 rotations

$$1 \text{ rotation} = \frac{60}{10000} = 6 \text{ ms}$$

Rotational latency = 6/2 = 3 ms

One rotation = One track

Number of sectors in one track = 600 ⇒ 600 sectors = 6 ms

One sector ⇒ 6/600 = 0.01 ms

The file contains 2000 sectors.

Therefore, 2000 sectors take = 2000 × 0.01 = 20 ms

Hence, total time needed to read the file

= (Number of sectors) × (Seek time + Rotational latency) + (Time for 2000 sectors)

= 2000 × (4 + 3) + 20 =14020

Ans. (14000–14020)

45. Consider a non-pipelined processor with a clock rate of 2.5 gigahertz and average cycles per instruction of four. The same processor is upgraded to a pipelined processor with five stages; but due to the internal pipelined delay, the clock speed is reduced to 2 gigahertz. Assume that there are no stalls in the pipeline. The speed up achieved in this pipelined processor is _____.

Solution: Non-pipelined method:

Frequency = 2.5 GHz

Time required for one cycle = 1/(2.5) = 0.4 ms

Each instruction requires 4 cycles. Therefore, time for each instruction = 0.4 × 4 = 1.6 ms

Pipelined structure:

Frequency = 2 GHz

Time = 0.5 ms

Enhanced time for each stage in pipeline structure = time for one stage of pipeline = 0.5 ms

$$\text{Speed up} = \frac{\text{Non-pipelined structure time}}{\text{Enhanced time in pipeline}} = \frac{1.6}{0.5}$$

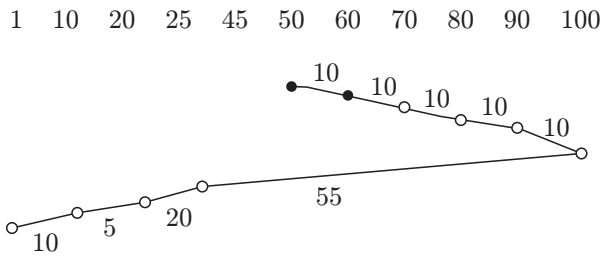
$$= 3.$$

Ans. (3.2)

46. Suppose the following disk request sequence (track numbers) for a disk with 100 tracks is given: 45, 20, 90, 10, 50, 60, 80, 25, 70. Assume that the initial position of the R/W head is on track 50. The additional distance that will be traversed by the R/W head when the Shortest Seek Time First (SSTF) algorithm is used compared to the SCAN (Elevator) algorithm (assuming that SCAN algorithm moves towards 100 when it starts execution) is _____ tracks.

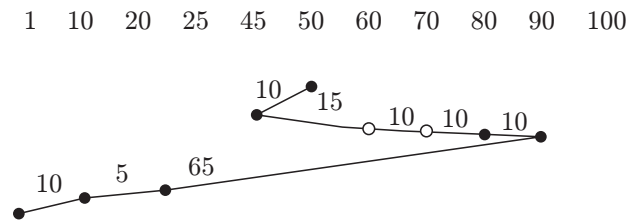
Solution:

SCAN:



Total Head Moments = 10+10+10+10+10+55
+20+5+10 = 140

SSTF:



Total Head Moments = 10+15+10+10+10+65
+5+10 = 130

Additional distance traversed = 140 − 130 = 10

Ans. (10)

47. Consider a main memory with five page frames and the following sequence of page references: 3, 8, 2, 3, 9, 1, 6, 3, 8, 9, 3, 6, 2, 1, 3. Which one of the following is true with respect to page replacement policies First In First Out (FIFO) and Least Recently Used (LRU)?

(a) Both incur the same number of page faults.
(b) FIFO incurs 2 more page faults than LRU
(c) LRU incurs 2 more page faults than FIFO
(d) FIFO incurs 1 more page faults than LRU

Solution:

FCFS:

					1	1	1	1	1	1	1	1	1	1
				9	9	9	9	9	9	9	9	2	2	2
		2	2	2	2	2	2	8	8	8	8	8	8	8
	8	8	8	8	8	8	3	3	3	3	3	3	3	3
3	3	3	3	3	3	6	6	6	6	6	6	6	6	6
			Hit						Hit	Hit	Hit		Hit	Hit
3	8	2	**3**	9	1	6	3	8	**9**	**3**	**6**	2	**1**	**3**

Page faults for FIFO = 9

LRU:

					1	1	1	1	1	1	1	2	2	2
				9	9	9	9	9	9	9	9	9	9	9
		2	2	2	2	2	2	8	8	8	8	8	1	1
	8	8	8	8	8	6	6	6	6	6	6	6	6	6
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
			Hit				Hit		Hit	Hit	Hit			Hit
3	8	2	**3**	9	1	6	**3**	8	**9**	**3**	**6**	2	1	**3**

Page faults for LRU = 9

Numbers of page faults are equal.
So, option (a) is correct.

Ans. (a)

48. $\int_{1/\pi}^{2/\pi} \frac{\cos(1/x)}{x^2}\, dx =$ _____.

Solution: Substituting $1/x = t$ in $\int_{1/\pi}^{2/\pi} \frac{\cos(1/x)}{x^2}\, dx$

Differentiating both sides, we get $-1/x^2\, dx = dt$

$t = 1/x = \pi$ and $\pi/2$

Therefore, $\int_{\pi/2}^{\pi} \cos(t)\, dt = \sin\pi - \sin\pi/2 = -1$

Ans. (−1)

49. Consider the following 2×2 matrix A where two elements are unknown and are marked by a and b. The eigenvalues of this matrix are −1 and 7. What are the values of a and b?

$$A = \begin{pmatrix} 1 & 4 \\ b & a \end{pmatrix}$$

(a) $a = 6,\ b = 4$
(b) $a = 4,\ b = 6$
(c) $a = 3,\ b = 5$
(d) $a = 5,\ b = 3$

Solution: Eigenvalues of matrix, $\lambda_1 = -1$ and $\lambda_2 = 7$

We know that, $\lambda_1 + \lambda_2 =$ Trace (i.e., sum of diagonal elements)

$$\lambda_1 \times \lambda_2 = \text{Determinant}$$

$$\lambda_1 + \lambda_2 = -1 + 7 = 6 = 1 + a \Rightarrow a = 5$$
$$\lambda_1 \times \lambda_2 = -7 = a - 4b \Rightarrow -7 = 5 - 4b \Rightarrow b = 3$$

Ans. (d)

50. An algorithm performs $(\log N)^{1/2}$ find operations, N insert operations, $(\log N)^{1/2}$ delete operations, and $(\log N)^{1/2}$ decrease-key operations on a set of data items with keys drawn form a linearly ordered set. For a delete operation, a pointer is provided to the record that must be deleted. For the decrease-key operation, a pointer is provided to the record that has its key decreased. Which one of the following data structures is the most suited for the algorithm to use, if the goal is to achieve the best asymptotic complexity considering all the operations?

(a) Unsorted array
(b) Min–heap
(c) Sorted array
(d) Sorted doubly linked list

Solution: Suppose if we use an unsorted array then, search operation will require $\Theta(N)$ time.
$(\log N)^{1/2}$ search operations will take $= \Theta(N(\log N)^{1/2})$ time.

Insertion will take a constant time, that is, $\Theta(1)$.
N insert operations need $= N \times \Theta(1) = \Theta(N)$ time.

Delete operations will also be done in constant time since the pointer to be deleted is provided and no time will be required for search.

$(\log N)^{1/2}$ delete operations will be completed in $= \Theta((\log N)^{1/2})$ time

Decrease key operation will also take $= \Theta((\log N)^{1/2})$ time

If we use sorted order array, insertion will take more time since we need to find its exact position in the list where to insert the data. And in the case of min heap, more time will be required in insertion and deletion since heapify algorithm needs to be implemented after each insertion or deletion.

Ans. (a)

51. Consider the following relations:

Students

Roll_No	Student_Name
1	Raj
2	Rohit
3	Raj

Performance

Roll_No	Course	Marks
1	Math	80
1	English	70
2	Math	75
3	English	80
2	Physics	65
3	Math	80

Consider the following SQL query.

SELECT S. Student_Name, sum (P.Marks)

FROM Student S, Performance P

WHERE S.Roll_No = P.Roll_No

GROUP BY S.Student_Name

The number of rows that will be returned by the SQL query is _____.

Solution: Since here group by Name is done, therefore marks for roll nos. 1 and 3 are added together giving 310 and marks for roll no. 2, that is, Rohit are added up to give 140. Thus, the resulting output table is:

Student_Name	Marks
Raj	310
Rohit	140

Ans. (2)

52. What is the output of the following C code? Assume that the address of x is 2000 (in decimal) and an integer requires four bytes of memory?

```
int main()
{
unsigned int x[4][3] = {(1,2,3),
{4,5,6}, {7, 8, 9}, {10, 11, 12}};
printf("%u,  %u,  %u",  x+3,  *(x+3),
*(x+2)+3);
}
```

(a) 2036, 2036, 2036
(b) 2012, 4, 2204
(c) 2036, 10, 10
(d) 2012, 4, 6

Solution:

1	2	3
4	5	6
7	8	9
10	11	12

As given in question, integer takes 4 bytes to store

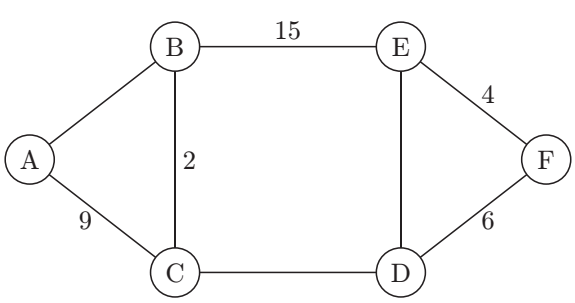
Here, X + 3 ⇒ 2000 + (3 × 3) × 4 = 2036

*(X + 3) ⇒ Value at address 2036, %u will print the address only, that is, 2036

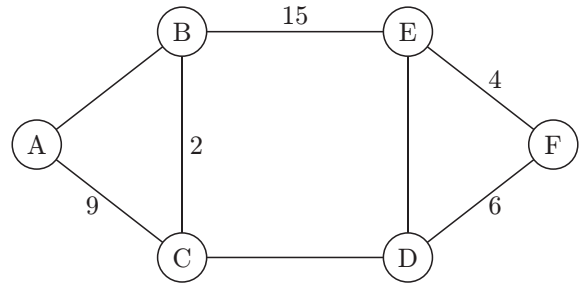
*(X + 2) + 3 ⇒ 2000 + (3 × 2)4 + 3 × 4 = 2036

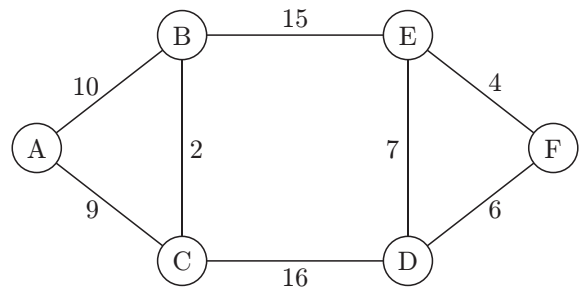
Ans. (a)

53. The graph shown below has 8 edges with distinct integer edge weights. The minimum spanning tree (MST) is of weight 36 and contains the edges: {(A, C), (B, C), (B, E), (E, F), (D, F)}. The edge weights of only those edges which are in the MST are given in the figure shown below. The minimum possible sum of weights of all 8 edges of this graph is _____.



Solution:





Sum = 10 + 9 + 2 + 15 + 16 + 7 + 4 + 6 = 69

Ans. (69)

54. Consider the following C function.

```
int fun1 (int n)
{
int i, j, k, p, q = 0;
for (i = 1; i < n; ++i)
{
p = 0;
for(j = n; j > 1; j = j/2)
++p;
for(k = 1; k < p; k = k*2)
++q;
}
return q;
}
```

Which one of the following most closely approximates the return value of the function fun1?

(a) n^3
(b) $n(\log n)^2$
(c) $n \log n$
(d) $n \log(\log n)$

Solution: First for loop will execute n times, the others are sequential not nested, so both execute $\log(n)$ and $\log(p)$ times but the value returned by fun1 is "q" which is calculated in the second loop. So, $n \times (\log(p))$, [where $p = \log(n)$] ⇒ $n \log(\log n)$times.

Ans. (d)

55. Consider the following pseudo code, where x and y are positive integers.

```
begin
q: = 0
r: = x
```

```
while r ≥ y do
    being
    r: = r - y
    q: = q + 1
    end
end
```

The post condition that needs to be satisfied after the program terminates is

(a) $\{r = qx + y \wedge r < y\}$
(b) $\{x = qy + r \wedge r < y\}$
(c) $\{y = qx + r \wedge 0 < r < y\}$
(d) $\{q + 1 < r - y \wedge y > 0\}$

Solution: In this program, q counts the number of times y is subtracted from x and at the remaining r value will act as remainder.

So, $x = qy + r$ and $r < y$.

Ans. (b)

SOLVED GATE(CS) 2015

SET 2

(Engineering Mathematics and Technical Section)

Q. No. 1 — 25 Carry One Mark Each

1. Consider the following two statements.

S1: If a candidate is known to be corrupt, then he will not be elected

S2: If a candidate is kind, he will be elected

Which one of the following statements follows from S1 and S2 per sound interference rules of logic?

(a) If a person is known to be corrupt, he is kind
(b) If a person is not known to be corrupt, he is not kind
(c) If a person is kind, he is not known to be corrupt
(d) If a person is not kind, he is not known to be corrupt

Solution: Let p: Candidate is known to be corrupt

q: Candidate is kind

r: Candidate is elected

S1: $p \rightarrow \sim r \Rightarrow r \rightarrow \sim p$

S2: $q \rightarrow r$

Therefore, $q \rightarrow \sim p$, that is, if a candidate is kind, he is not known to be corrupt.

Ans. (c)

2. The cardinality of the power set of {0, 1, 2, ... 10} is _____.

Solution: The cardinality of the power set of {0, 1, 2, ... , 10} is $2^{\text{number of elements in set}}$, that is, $2^{11} = 2048$.

Ans. (2048)

3. Let R be the relation on the set of positive integers such that aRb if and only if a and b are distinct and have a common divisor other than 1. Which one of the following statements about R is true?

(a) R is symmetric and reflexive but not transitive
(b) R is reflexive but not symmetric and not transitive
(c) R is transitive but not reflexive and not symmetric
(d) R is symmetric but not reflexive and not transitive

Solution: R is not reflexive since a and b are distinct as given in the question. Now, the given relation is true for $a = 4$ and $b = 8$; this is also true for $a = 8$ and $b = 4$. Hence, it satisfies the condition for symmetric relation, that is, if (a,b) satisfies then (b,a) should satisfy the relation.

R does not satisfy transitivity. For example, $a = 5$, $b = 10$, $c = 12$. So, (5,10) satisfies the relation, (10,12) also satisfies the relation, but (5,12) does not. Hence, the relation is not transitive.

Ans. (d)

4. The number of divisors of 2100 is _____.

Solution:

$2100 = 2^2 + 3 \times 5^2 \times 7$

Number of divisors $= (2 + 1) \times (1 + 1) \times (2 + 1) \times (1 + 1) = 36$

Ans. (36)

5. The larger of the two eigenvalues of the matrix $\begin{bmatrix} 4 & 5 \\ 2 & 1 \end{bmatrix}$ is _____.

Solution:

$(4-\lambda)\ (1-\lambda) - 10 = 0 \Rightarrow 4 - 4\lambda - \lambda + \lambda^2 = 10$

$\lambda^2 - 5\lambda - 6 = 0 \Rightarrow \lambda = 6, -1$

Hence, the larger value of $\lambda = 6$

Ans. (6)

6. An unordered list contains n distinct elements. The number of comparisons to find an element in this list that is neither maximum nor minimum is

(a) $\Theta(n \log n)$ (b) $\Theta(n)$
(c) $\Theta(\log n)$ (d) $\Theta(1)$

Solution: In an unordered list, if we consider any three elements, one of them would neither be maximum or minimum. Hence, in constant time we can find the required element.

Ans. (d)

7. The minimum number of JK flip-flops required to construct a synchronous counter with the count sequence (0, 0, 1, 1, 2, 2, 3, 3, 0, 0, ...) is _____.

Solution: Number of distinct states = 4. Hence, minimum flip-flops required is 2.

Ans. (2)

8. Assume that for a certain processor, a read request takes 50 nanoseconds on a cache miss and 5 nanoseconds on a cache hit. Suppose while running a program, it was observed that 80% of the processors read requests result in a cache hit. The average and access time in nanoseconds is _____.

Solution: Given that cache hit rate = 0.8, so cache miss rate = 0.2

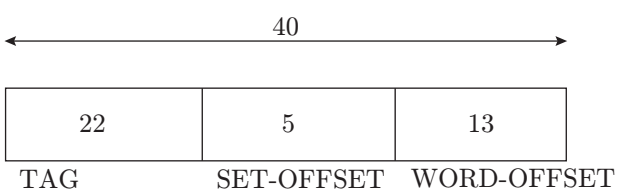
Time on cache miss = 50 ns and time on cache hit = 5 ns

So, average access time = 0.8(5) + 0.2(50) = 14 ns

Ans. (14)

9. A computer system implements a 40-bit virtual address, page size of 8 kilobytes, and a 128-entry translation look-aside buffer (TLB) organized into 32 sets each having four ways. Assume that the TLB tag does not store any process id. The minimum length of the TLB tag in bits is _____.

Solution:



Page size = 8 KB = $2^{10} \times 2^3 = 2^{13} \Rightarrow$ 13 byte word-offset.

Number of sets = 32 = $2^5 \Rightarrow$ 5 bytes set-offset

Remaining bytes are for tag = 40 − 18 = 22

Ans. (22)

10. Consider the following statements

I. The complement of every Turing decidable language is Turing decidable.
II. There exists some language which is in NP but is not Turing decidable.
III. If L is a language in NP, L is Turing decidable.

Which of the above statements is/are true?

(a) Only II (b) Only III
(c) Only I and II (d) Only I and III

Solution: Turing decidable languages are recursive languages. Recursive languages are closed under complement, that is, complement of every recursive language is recursive.

Ans. (d)

11. Consider the following function written in the C programming language.

```
void foo (char *a) {
if (*a && *a != ' ')
{
foo (a+1);
putchar *a;
}
}
```

The output of the above function on input "ABCD EFGH" is

(a) ABCD EFGH (b) ABCD
(c) HGFE DCBA (d) DCBA

Solution: If condition fails when the space is encountered, then output will be DCBA.

Ans. (d)

12. Consider a complete binary tree where the left and the right subtrees of the root are max-heaps. The lower bound for the number of operations to convert the tree to a heap is

(a) $\Omega(\log n)$ (b) $\Omega(n)$
(c) $\Omega(n \log n)$ (d) $\Omega(n^2)$

Solution: Left and right subtrees of binary tree are max heaps. Thus, to convert the tree into heap we only need to compare the roots. This would always be done on the number of levels which is $\log(n)$.

Ans. (a)

13. A binary tree T has 20 leaves. The number of nodes in T having two children is _____.

Solution: The number of nodes with two children is always one less than the number of leaf nodes.

Here, Leaf nodes = 20 ⇒ Number of two children nodes = 19.

Ans. (19)

14. Consider the following C function.

```
int fun(int n)
{
int x=1, k;
if (n==1) return x;
```

```
for (k=1; k<n; ++k)
x = x + fun (k) * fun (n-k);
return x;
}
```

The return value of fun (5) is _____ .

Solution:

For $n = 1$, $f(n) = 1$

For $n > 1$, $f(n) = 1 + \sum_{k=1}^{n-1} f(k) f(n-k)$

$f(5) = 51$

Ans. (51)

15. A software requirements specification (SRS) document should avoid discussing which one of the following?

(a) User interface issues
(b) Non-functional requirements
(c) Design specification
(d) Interfaces with third party software

Solution: Design specifications are not discussed in SRS document. Only design constraints are discussed

Ans. (c)

16. Consider two decision problems Q_1, Q_2 such that Q_1 reduces in polynomial time to 3-SAT and 3 -SAT reduces in polynomial time to Q_2. Then which one of following is consistent with the above statement?

(a) Q_1 is in NP, Q_2 is NP hard
(b) Q_1 is in NP, Q_2 is NP hard
(c) Both Q_1 and Q_2 are in NP
(d) Both Q_1 and Q_2 are NP hard

Solution: 3-SAT is NPC problem.

$Q_1 \leq_{\mathrm{p}}$ 3-SAT and 3-SAT $\leq_{\mathrm{p}} Q_2$

If an NP problem A is polynomial time reducible to another NP problem B, then B is NPC. But if we do not know the type of B problem, then B is said to be NP hard. Hence, Q_1 is in NP and Q_2 is NP hard.

Ans. (a)

17. Match the following:

(P) Lexical analysis	(1) Graph coloring
(Q) Parsing	(2) DFA minimization
(R) Register allocation	(3) Post-order traversal
(S) Expression evaluation	(4) Production tree

(a) P – 2, Q – 3, R –1, S – 4
(b) P – 2, Q – 1, R – 4, S – 3
(c) P – 2, Q – 4, R – 1, S – 3
(d) P – 2, Q – 3, R – 4, S – 1

Solution: Lexical analyzer uses DFA.

Parsing is done though production tree.

Register allocation can be used in graph coloring.

Post-order traversal is used for expression evaluations.

Ans. (c)

18. In the context of abstract-syntax-tree (AST) and control-flow-graph (CFG), which one of the following is TRUE?

(a) In both AST and CFG, let node, N_2 be the successor of node N_1. In the input program, the code corresponding to N_2 is present after the code corresponding in N_1.
(b) For any input program, neither AST nor CFG will contain a cycle.
(c) The maximum number of successors of a node in an AST and a CFG depends on the input program.
(d) Each node is AST and CFG corresponds to atmost one statement in the input program.

Solution: Option (a) is false in the case when CFG contains a cycle. Hence, option (b) is also false as it says CFG cannot contain cycle. Also, single block can contain multiple statements. Thus, option (d) is also false.

Ans. (c)

19. Consider the basic COCOMO model where E is the effort applied in person-months, D is the development time in chronological months, KLOC is the estimated number of delivered lines of code (in thousands) and a_b, b_b, c_b, d_b have their usual meanings. The basic COCOMO equations are of the form

(a) $E = a_b(KLOC)\exp(b_b),\ D = c_b(E)\exp(d_b)$
(b) $D = a_b(KLOC)\exp(b_b),\ E = c_b(D)\exp(d_b)$
(c) $E = a_b \exp(b_b),\ D = c_b(KLOC)\exp(d_b)$
(d) $E = a_b \exp(D_b),\ D = c_b(KLOC)\exp(b_b)$

Solution: Basic COCOMO model equations are of the form:

Effort $= a_b(KLOC)\ \exp(b_b)$

Development time $= c_b(\text{Effort})\ \exp(d_b)$

Ans. (a)

20. A system has 6 identical resources and N processes competing for them. Each process can request at most 2 resources. Which one of the following values of N could lead to a deadlock?

(a) 1 (b) 2
(c) 3 (d) 4

Solution: No deadlock occurs when $N = 1$, $N = 2$ and $N = 3$. When $N = 4$, deadlock may occur as two processes will get complete resources; and two will wait for the completion of other two.

Ans. (d)

21. Consider the following transaction involving two bank accounts x and y.

read (x) ; $x := x - 50$; write (x); read (y);
$y := y + 50$; write (y)

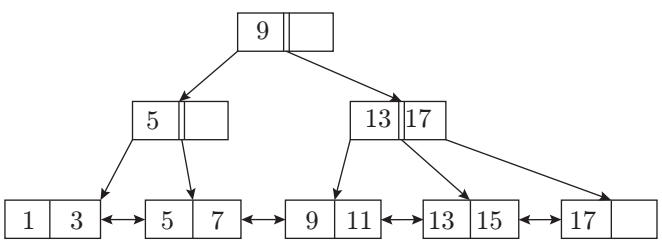
The constraint that the sum of the accounts x and y should remain constant is that of

(a) Atomicity (b) Consistency
(c) Isolation (d) Durability

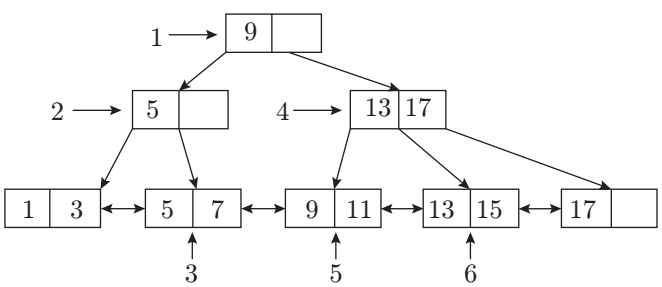
Solution: Consistency means the overall state of system should remain constant before and after the transaction. Hence, the sum remains constant with the property of consistency.

Ans. (b)

22. With reference to the B+ tree index of order 1 shown below, the minimum number of nodes (including the Root node) that must be fetched in order to satisfy the following query: "Get all records with a search key greater than or equal to 7 and less than 15" is _____.



Solution:



Ans. (6)

23. Identify the correct order in which a server process must invoke the function calls accept, bind, listen, and recv according to UNIX socket APL

(a) listen, accept, bind, recv
(b) bind, listen, accept, recv
(c) bind, accept, listen, recv
(d) accept, listen, bind, recv

Solution: The correct order is bind, listen, accept, recv. First three are used for connection establishment and recv is used for data transfer operation.

Ans. (b)

24. A link has a transmission speed of 10^6 bits/sec. It uses data packets of size 1000 bytes each. Assume that the acknowledgement has negligible transmission delay, and that its propagation delay is the same as the data propagation delay. Also assume that the processing delays at the nodes are negligible. The efficiency of the stop-and-wait protocol in this setup is exactly 25%. The value of the one-way propagation delay (in milliseconds) is _____.

Solution: In stop and wait, efficiency $= \dfrac{1}{(1+2a)}$ $= \dfrac{1}{4} \Rightarrow a = 3/2$

Transmission time, $T_t = \dfrac{1000 \times 8}{10^6} = 8$ ms

Propagation time, $T_p = a \times T_t = 3 \times \dfrac{8}{2} = 12$ ms

Ans. (12)

25. Which one of the following statements is NOT correct about HTTP cookies?

(a) A cookie is a piece of code that has the potential to compromise the security of an internet user
(b) A cookie gains entry to the user's work area through an HTTP header
(c) A cookie has an expiry date and time
(d) Cookies can be used to track the browsing pattern of a user at a particular site

Solution: A cookie cannot compromise the security of an internet user.

Ans. (a)

Q. No. 26 — 55 Carry Two Marks Each

26. Consider the following routing table at an IP router:

Network No.	Net Mask	Next Hop
128.96.170.0	255.255.254.0	Interface 0
128.96.168.0	255.255.254.0	Interface 1
128.96.166.0	255.255.254.0	R2
128.96.164.0	255.255.252.0	R3
0.0.0.0	Default	R4

For each IP address in Group I identify the correct choice of the next hop from Group II using the entries from the routing table above.

Group I	Group II
(i) 128.96.171.92	(a) Interface 0
(ii) 128.96.167.151	(b) Interface 1
(iii) 128.96.163.151	(c) R2
(iv) 128.96.165.121	(d) R3
	(e) R4

(a) i – a, ii – c, iii – e, iv – d
(b) i – a, ii – d, iii – b, iv – e
(c) i – b, ii – c, iii – d, iv – e
(d) i – b, ii – c, iii – e, iv – d

Solution:

(i)

IP Address:	128.96.171.92
Subnet Mask:	255.255.254.0

Network Address: 128.96.170.0

This matches with Interface 0.

(ii)

IP Address:	128.96.167.151
Subnet Mask:	255.255.254.0

Network Address: 128.96.166.0

This matches with R2.

(iii)

IP Address:	128.96.163.151
Subnet Mask:	255.255.252.0

Network Address: 128.96.170.0

This matches with R4.

(iv)

IP Address:	128.96.165.121
Subnet Mask:	255.255.254.0

Network Address: 128.96.164.0

This matches with R3.

So, option (a) is correct.

Ans. (a)

27. Host A sends a UDP datagram containing 8880 bytes of user data to host B over an Ethernet LAN. Ethernet frames may carry data up to 1500 bytes (i.e. MTU = 1500 bytes). Size of UDP header is 8 bytes and size of IP heard is 20 bytes. There is no option field in IP header. How many total number of IP fragments will be transmitted and what will be the contents of offset field in the last fragment?

(a) 6 and 925 (b) 6 and 7400
(c) 7 and 1110 (d) 7 and 8880

Solution: Datagram size = 1500 bytes
Data in one datagram = 1500 – 20 – 8 = 1472 bytes

Total data to send = 8880 bytes

$$\text{Number of datagrams} = \frac{8880}{1472} = 7$$

$$\text{and offset} = \frac{8872}{8} = 1109 + 1 = 1110$$

Ans. (c)

28. Assume that the bandwidth for a TCP connection is 1048560 bits/sec. Let α be the value of RTT in milliseconds (rounded off to the nearest integer) after which the TCP window scale option is needed. Let β be the maximum possible window size the window scale option. Then the values of α and β are

(a) 63 milliseconds, 65535×2^{14}
(b) 63 milliseconds, 65535×2^{16}
(c) 500 milliseconds, 65535×2^{14}
(d) 500 milliseconds, 65535×2^{16}

Solution: Bandwidth = 1048560 bits/sec
TCP window scale option is needed when size increases more than 65535 B.

$$\text{Time to send 65535 B} = \frac{65535 \times 8}{1048560 \times 1000} = 500 \text{ ms}$$

So, $\alpha = 500$ ms.

Maximum window size with window scale option (β) = 65535×2^{14}

Ans. (c)

29. Consider a simple checkpointing protocol and the following set of operations in the log.

(start, T4); (write, T4, y, 2, 3); (start, T1);
(commit, T4); (write, T1, z, 5, 7);
(checkpoint);
(start, T2); (write, T2, x, 1, 9);
(commit, T2); (start, T3),
(write, T3, z, 7, 2);

If a crash happens now and the system tries to recover using both undo and redo operations,

what are the contents of the undo lists and the redo list?

(a) Undo: T3,T1; Redo: T2
(b) Undo: T3,T1; Redo: T2,T4
(c) Undo: none; Redo: T2,T4,T3,T1
(d) Undo: T3,T1,T4; Redo: T2

Solution: The transactions that are not committed must not be saved in database. Hence, we undo transactions T1 and T3 here as they are not yet committed. Transaction that is after the checkpoint must be redone. Hence, we redo the transaction T2.

Ans. (a)

30. Consider two relations R_1(A,B) with the tuples (1,5), (3,7) and R_2 (A,C) = (1,7), (4,9). Assume that R(A,B,C) is the full natural outer join of R_1 and R_2. Consider the following tuples of the form (A,B,C): a = (1,5,null), b = (1,null,7) c = (3,null,9), d = (4,7,null), e = (1,5,7), f = (3,7,null), g = (4,null,9).Which one of the following statements is correct?

(a) R contains a, b, e, f, g but not c, d.
(b) R contains all of a, b, c, d, e, f, g
(c) R contains e, f, g but not a, b
(d) R contains e but not f, g

Solution: In option (c)

$R_1 \Rightarrow$

A	B
1	5
3	7

$R_2 \Rightarrow$

A	C
1	7
4	9

R ⇒

A	B	C
1	5	7
3	7	Null
4	Null	9

a = (1,5,null), b = (1,null,7) c = (3,null,9), d = (4,7,null), e = (1,5,7), f = (3,7,null), g = (4,null,9).

a, b, c and d are not the rows of relation R but e, f, g are the rows in relation R. Hence, option (c) is correct.

Ans. (c)

31. Consider six memory partitions of sizes 200 KB, 400 KB, 600 KB, 500 KB, 300 KB and 250 KB, where KB refers to kilobyte. These partitions need to be allotted to four processes of sizes 357 KB, 210 KB, 468 KB and 491 KB in that order. If the best fit algorithm is used, which partitions are NOT allotted to any process?

(a) 200 KB and 300 KB (b) 200 KB and 250 KB
(c) 250 KB and 300 KB (d) 300 KB and 400 KB

Solution: Partitions needed for the processes = 357 KB, 210 KB, 468 KB and 491 KB

Using best fit strategy, the places allocated to the processes are:

	357 KB	491 KB	468 KB		210 KB
200	400	600	500	300	250

Thus, two memory slots are never used.

Ans. (a)

32. Consider a typical disk that rotates at 15000 rotations per minute (RPM) and has a transfer rate of 50×10^6 bytes/sec. if the average seek time of the disk is twice the average rotational delay and the controller's transfer time is 10 times the disk transfer time, the average time (in milliseconds) to read or write a 512-byte sector of the disk is _____.

Solution:

RPM = 15000

15000 rotations ⇒ 60 sec

1 rotation ⇒ 4 ms = average seek time

Rotational latency = 4/2 = 2 ms

Transfer rate = 50×10^6 bytes/sec

50×10^6 bytes ⇒ 1 sec

512 bytes $\Rightarrow \dfrac{512}{(50 \times 10^6)} = 0.01$ ms ⇒ Disk transfer time

Controller's transfer time = 10 × (Disk transfer time) = 10 × 0.01 = 0.1 ms

Average time = 4 + 2 + 0.1 = 6.1 ms

Ans. (6.1)

33. A computer system implements 8 kilobyte pages and a 32-bit physical address space. Each page table entry contains a valid bit, a dirty bit, three permission bits, and the translation. If the maximum size of the page table of a process is 24 megabytes, the length of the virtual address supported by the system is _____ bits.

Solution: Page size = 8 KB = 2^{13}
Physical address space = 32 bits

Number of frames = $2^{32}/2^{13} = 2^{19}$ frames

Number of bits for page table entry = 19 + 1 V + 1 D + 3 P = 24 bits

Page table size = 24 MB = $(24 \times 2^{20} \times 8)$ bits
= 24 × (Number of pages)

So, number of pages = 2^{23}

Therefore, virtual address apace contains
(23 + 13) = 36 bits

Ans. (36)

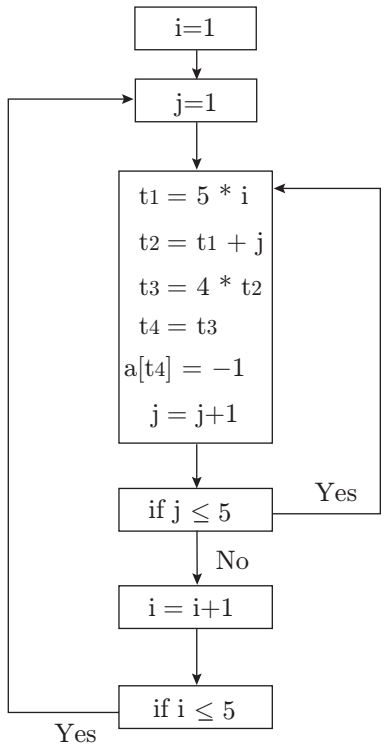
34. Consider the intermediate code given below.

(1) i = 1 (2) j = 1
(3) t1 = 5 * i (4) t2 = t1+ j
(5) t3 = 4 * t2 (6) t4 = t3
(7) a[t4] = −1 (8) j = j + 1
(9) if j < = 5 goto (3) (10) i = i + 1
(11) if i < 5 goto (2)

The number of nodes and edges in the control-flow-graph constructed for the above code, respectively, are

(a) 5 and 7 (b) 6 and 7
(c) 5 and 5 (d) 7 and 8

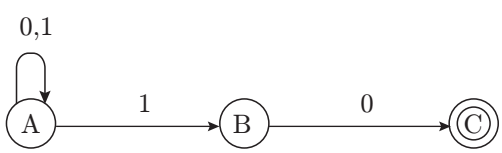
Solution:
Number of nodes = 6
Number of edges = 7



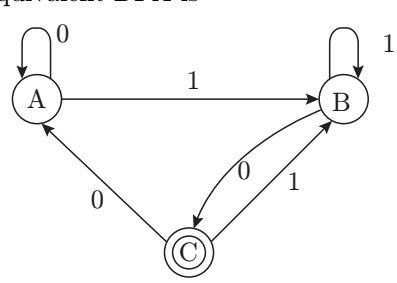
Ans. (b)

35. The number of states in the minimal deterministic finite automaton corresponding to the regular expression (0 + 1)*(10) is _____.

Solution:



The equivalent DFA is



Ans. (3)

36. Which of the following languages is/are regular?

L1:$\{wxw^R | w, x \in \{a,b\}^* \text{ and } |w|, |x| > 0\}$, w^R is the reverse of string w

L2: $\{a^n b^m \mid m \neq n \text{ and } m, n \geq 0\}$

L3: $\{a^p b^q c^r \mid p, q, r \geq 0\}$

(a) L_1 and L_3 only (b) L_2 only
(c) L_2 and L_3 only (d) L_3 only

Solution: L_1 and L_3 are regular since we can make a regular expression for them.

Ans. (a)

37. Given below are some algorithms, and some algorithm design paradigms.

(1) Dijkstra's shortest path	(i) Divide and conquer
(2) Floyd-Warshall algorithm to compute all pair shortest path	(ii) Dynamic programming
(3) Binary search on a sorted array	(iii) Greedy design
(4) Backtracking search on a graph	(iv) Depth-first search
	(v) Breadth-first search

Match the above algorithms on the left to the corresponding design paradigm they follow.

(a) 1 − i, 2 − iii, 3 − i, 4 − v
(b) 1 − iii, 2 − iii, 3 − i, 4 − v
(c) 1 − iii, 2 − ii, 3 − i, 4 − iv
(d) 1 − iii, 2 − ii, 3 − i, 4 − v

Solution: Dijkstra uses the Greedy approach.

All pair shortest path algorithm uses the concept of overlapping while computing the shortest path. Hence, it uses dynamic programming.

Binary search divides the list and then searches for an element. Hence, it uses divide and conquer strategy.

Backtracking is used in DFS. Once we have covered one path, we need to backtrack to move on to another.

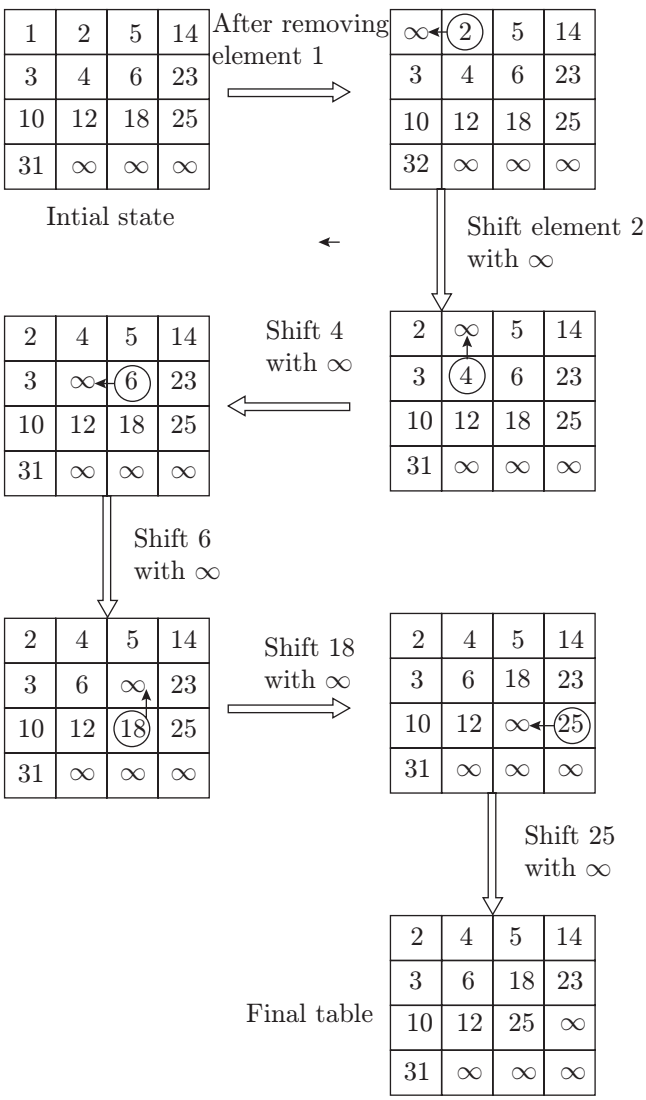
Ans. (c)

38. A Young tableau is a 2D array of integers increasing from left to right and from top to bottom. Any unfilled entries are marked with ∞, and hence there cannot be any entry to the right of, or below a ∞. The following Young tableau consists of unique entries.

1	2	5	14
3	4	6	23
10	12	18	25
31	∞	∞	∞

When an element is removed from a Young tableau, other elements should be moved into its place so that the resulting table is still a Young tableau (unfilled entries may be filled in with a ∞). The minimum number of entries (other than 1) to be shifted, to remove 1 from the given Young tableau is _____.

Solution:



So, minimum number of entries to be shifted = 5.

Ans. (5)

39. Suppose you are provided with the following function declaration in the C programming language

```
int partition (int a[], int n);
```

The function treats the first element of a[] as a pivot, and rearranges the array so that all elements less than or equal to the pivot is in the left part of the array, and all elements greater than the pivot is in the right part. In addition, it moves the pivot so that the pivot is the last elements of the left part. The return value is the number of elements in the left part.

The following partially given function in the C programming language is used to find the kth smallest element in an array a [ ] of size n using the partition function. We assume k ≤ n.

```
int kth_smallest(int a[], int n, int k)
{
int left_end partition (a,n);
if (left_end + 1 == k)
{
return a [left_end];
}
if ((left_end + 1) > k)
{
return kth_smallest (__________);
}
else
return kth_smallest (__________);
}
}
```

The missing argument lists are respectively

(a) (a, left _ end, k) and (a + left _ end +1, n − left _ end −1, k − left _ end −1)
(b) (a, left _ end, k) and (a, n − left _ end −1, k − left _ end −1)
(c) (a + left _ end +1, n − left end −1, k − left _ end −1) and (a, left _ end, k)
(d) (a, n − left _ end −1, k − left _ end −1) and (a, left _ end, k)

Solution: The given code is implementation of kth smallest element.

Replace the blanks with (a, left_end, k) and (a + left_end + 1, n − left_end − 1, k − left_end − 1).

Ans. (a)

40. Which one of the following hash functions on integers will distribute keys most uniformly over 10 buckets numbered 0 to 9 for i ranging from 0 to 2020?

(a) $h(i) = i^2 \bmod 10$
(b) $h(i) = i^3 \bmod 10$
(c) $h(i) = (11 * i^2) \bmod 10$
(d) $h(i) = (12 * i) \bmod 10$

Solution: If we take the first 10 elements, number of collisions taken by the hash function is given by option (b), which is less when compared to others.

Ans. (b)

41. The secant method is used to find the root of an equation f(x) = 0. It is started from two distinct estimates, x_a and x_b for the root. It is an iterative procedure involving linear interpolation to a root. The iteration stops if $f(x_b)$ is very small and then x_b is the solution. The procedure is given below. Observe that there is an expression which is missing and is marked by ? Which is the suitable expression that is to be put in place of ? so that it follows all steps of the secant method?

Secant

```
Initialize: xa, xb, ε, N
// ε =convergence indicator
// N = maximum no. of iterations
fb = f(xb)
i = 0
while (i < N and |fb| > ε) do
i = i + 1 // update counter
xt = ? // missing expression for
xa = xb
xb = xt // intermediate value
fb = f(xb) // function value at new xb
end while
if |fb| > ε then // loop is terminated
with i=N
write "Non-convergence"
else
write "return xb"
end if
```

(a) $x_b-(f_b-f(x_a))f_b/(x_b-x_a)$
(b) $x_a-(f_a-f(x_a))f_a/(x_b-x_a)$
(c) $x_b-(x_b-x_a)f_b/(f_b-f(x_a))$
(d) $x_a-(x_b-x_a)f_a/(f_b-f(x_a))$

Solution: The given pseudocode implements secant method. Option (c) provides the direct formula.
Secant method: $x_{n+1} = (f(x_{n-1}) - f_{n-1}(x_n))/f_n - f_{n-1}$
from (c), $x_b - ((x_b - x_a)\, f_b)/(f_b - f_a) = (f_b x_a - f_a x_b)/(f_b - f_a)$

Ans. (c)

42. Consider the C program below.

```
#include <stdio.h>
int *A, stkTop;
int stkFunc (int opcode, int val)
{
static int size =0, stkTop=0;
switch (opcode)
{
case-1: size = val; break;
case 0: if (stkTop < size) A (stktop++]
= val; break;
default: if (stktop) return A [--stkTop];
}
return -1;
}
int main()
{
int B[20]; A = B; stkTop = -1;
stkFunc(-1, 10);
stkFunc(0, 5);
stkFunc(0, 10);
printf("%d/n", stkFunc(1, 0) +
stkfunc(1, 0);
}
```

The value printed by the above program is _____.

Solution: stkFunc(−1,0) returns size = 10. Then, stkFunc(0,5) returns 5. Then StrFunc(0,10) returns 10. StkFunc(1,0) returns 5 and then 10, so answer is 15.

Ans. (15)

43. Consider the sequence of machine instruction given below:

MUL R5, R0, R1

DIV R6, R2, R3

ADD R7, R5, R6

SUB R8, R7, R4

In the above sequence, R0 to R8 are general purpose registers. In the instructions shown, the first register stores the result of the operation performed on the second and the third registers. This sequence of instructions is to be executed in a pipelined instruction processor with the following 4 stages (1) Instruction Fetch and Decode (IF), (2) Operand Fetch (OF), (3) Perform Operation (PO) and (4) Write back the result (WB) . The IF,OF and WB stages take 1 clock cycle each for any instruction The PO stage takes 1 clock cycle for ADD or SUB instruction, 3 clock cycles for MUL instruction and 5 clock cycles for DIV instruction.

The pipelined processor uses operand forwarding from the PO stage to the OF stage. The number of clock cycles taken for the execution of the above sequence of instructions is _____.

Solution:

1	2	3	4	5	6	7	8	9	10	11	12	13
I	I	P	P	P	W							
	O	O	_	_	P	P	P	P	P	W		
		I	_	_	O	_	_	_	_	P	W	
			I	_	_	O	_	_	_	_	P	W

Ans. (13)

44. Consider a processor with byte-addressable memory. Assume that all registers, including Program Counter (PC) and Program Status Word (PSW), are of size 2 bytes. A stack in the main memory is implemented from memory location $(0100)_{16}$ and it grows upward. The stack pointer (SP) points to the top element of the stack. The current value of SP is $(016E)_{16}$. The CALL instruction is of two words, the first word is the op-code and the second word is the starting address of the subroutine. (one word = 2 bytes). The CALL instruction is implemented as follows:

Store the current value of PC in the stack

Store the value of PSW register in the stack

Load the starting address of the subroutine in PC

The content of PC just before the fetch of a CALL instruction is $(5FA0)_{16}$. After execution of the CALL instruction, the value of the stack pointer is

(a) $(016A)_{16}$ (b) $(016C)_{16}$
(c) $(0170)_{16}$ (d) $(0172)_{16}$

Solution:

Given that memory is byte addressable. Implementation of CALL instruction is as follows:

- Store the current value of PC in the stack: Current stack pointer $+2 = (016E)_{16} + 2$
- Store the value of PSW register in the stack: $(016E)_{16} + 2 + 2 = = (0172)_{16}$

Ans. (d)

45. The number of min-terms after minimizing the following Boolean expression is ———.

$[D' + AB' + A'C + AC'D + A'C'D]'$

Solution: $F(A, B, C, D) = [D' + AB' + A'C + AC'D + A'C'D]'$

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	0
10	0	0	0	0

The number of min-terms = 1.

Ans. (1)

46. Let $f(x) = x^{-(1/3)}$ and A denote the area of the region bounded by $f(x)$ and the X-axis, when x varies from -1 to 1. Which of the following statements is/are TRUE?

I f is continuous in $[-1,1]$
II f is not bounded in $[-1,1]$
III A is nonzero and finite

(a) II only (b) III only
(c) II and III only (d) I, II and III

Solution:

$$\int_{-1}^{1} f(x)\,dx = \int_{-1}^{0} x^{-1/3}dx + \int_{0}^{1} x^{-1/3}dx = 0$$

So, only statement II is true.

Ans. (c)

47. Perform the following operations on the matrix

$$\begin{bmatrix} 3 & 4 & 45 \\ 7 & 9 & 105 \\ 13 & 2 & 195 \end{bmatrix}$$

(i) Add the third row to the second row
(ii) Subtract the third column from the first column.

The determinant of the resultant matrix is _____.

Solution: After applying row and column operations, the resultant matrix is:

$$B = \begin{bmatrix} -42 & 4 & 45 \\ -280 & 11 & 300 \\ -182 & 2 & 195 \end{bmatrix}$$

$$|B| = 0$$

Ans. (0)

48. The number of onto functions (surjective functions) from set $X = \{1,2,3,4\}$ to set $Y = \{a,b,c\}$ is ____.

Solution: The number of onto functions from X to Y is

$$\sum_{k=0}^{n-1} (-1)^{kn} c_k (n-k)^m, \text{ where } |X| = m \text{ and } |Y| = n$$

Here, $X = \{1, 2, 3, 4\}$, $Y = \{a, b, c\}$, $m = 4$ and $n = 3$

Applying the formula we get, the number of onto functions = 36

Ans. (36)

49. Let X and Y denote the sets containing 2 and 20 distinct objects respectively and F denote the set of all possible functions defined from X to Y. Let f be randomly chosen from F. The probability of f being one-to-one is _____.

Solution: $|X| = 2$ and $|Y| = 20$

Total number of functions from

$$X \text{ to } Y = 20^2 = 400$$

Total number of one to one functions

$$= {}^{20}P_2 = 380$$

$$\text{Probability} = \frac{380}{400} = 0.95$$

Ans. (0.95)

50. Consider the alphabet $\Sigma = \{0.1\}$, the null/empty string λ and the sets of strings X_0, X_1, and X_2 generated by the corresponding non-terminals of a regular grammar X_0, X_1, and X_2 are related as follows.

$X_0 = 1\ X_1$

$X_1 = 0\ X_1 + 1\ X_2$

$X_2 = 0\ X_1 + \{\lambda\}$

Which one of the following choices precisely represents the strings in X_0?

(a) 10(0*+(10)*)1
(b) 10(0*+(10)*)*1
(c) 1(0 +10)*1
(d) 10(0 +10*)*1+110(0 +10)*1

Solution: Strings formed by the given equations are represented by regular expression: 1(0 +10)*1.

Ans. (c)

51. A graph is self-complementary if it is isomorphic to its complement. For all self-complementary graphs on n vertices, n is

(a) A multiple of 4
(b) Even
(c) Odd
(d) Congruent to 0 mod 4, or, 1 mod 4

Solution: In a self-complementary graph, the number of vertices is equal to exactly half of number of edges in a complete graph, that is, $n(n-1)/4$. Hence, answer is (d).

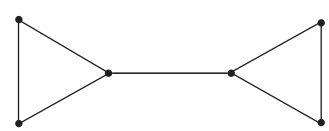
Ans. (d)

52. In a connected graph, a bridge is an edge whose removal disconnects a graph. Which one of the following statements is true?

(a) A tree has no bridges
(b) A bridge cannot be part of a simple cycle
(c) Every edge of a clique with size ≥ 3 is a bridge (A clique is any compete subgraph of a graph)
(d) A graph with bridges cannot have a cycle

Solution:

Option (a): False as in tree, every edge acts as a bridge.
Option (b): True
Option (c): False as every edge of complete graph is not bridge. Clique is also a complete graph.
Option (d): False as the graph is having a cycle and bridge.



Ans. (b)

53. Which one of the following well formed formulae is tautology?

(a) $\forall x\ \exists y\ R(x, y) \leftrightarrow \exists y\ \forall x\ R(x, y)$
(b) $(\forall x\ [\exists y\ R(x,y) \rightarrow S(x,y)]) \rightarrow \forall x \exists y S(x,y)$
(c) $[\forall x \exists y\ (P(x,y) \rightarrow R(x,y)] \leftrightarrow [\forall x \exists y (\neg P(x,y) \vee R(x,y)]$
(d) $\forall x \forall y\ P(x,y) \rightarrow \forall x \forall y P(y,x)$

Solution: Tautology is that which is always true in all cases. Check the expressions for this.

Ans. (c)

54. Which one of the following assertions concerning code inspection and code walkthrough is true?

(a) Code inspection is carried out once the code has been unit tested
(b) Code inspection and code walkthrough are synonyms

(c) Adherence to coding standards is checked during code inspection
(d) Code walkthrough is usually carried out by an independent test team

Solution: Code inspection generally checks that coding standards are followed or not.

Ans. (c)

55. A half adder is implemented with XOR and AND gates. A full adder is implemented with two half adders and one OR gate. The propagation delay of an XOR gate is twice that of an AND/OR gate. The propagation delay of an AND/OR gate is 1.2 microseconds. A 4-bit ripple-carry binary adder is implemented by using four full adders. The total propagation time of this 4-bit binary adder in microseconds is _____.

Solution: Total delay $= 4 \times 4.8 = 19.2\ \mu s$

Ans. (19.2)

SOLVED GATE(CS) 2015

SET 3

(Engineering Mathematics and Technical Section)

Q. No. 1 – 25 Carry One Mark Each

---

1. Consider the following C program segment.

```
#include <stdio.h>
int main()
{
char s1[7] = "1234", *p;
p = s1 + 2;
*p = '0';
printf("%s", s1);
}
```

What will be printed by the program?

(a) 12 (b) 120400
(c) 1204 (d) 1034

Solution: The given character array representation:

1	2	3	4	\0
100	101	102	103	

s1: Base address = 100

p = 100 + 2 = 102

*p = 102 = 0

1	2	0	4	\0
100	101	102	103	

Output will be: 1204

Ans. (c)

2. Suppose U is the power set of the set $S = \{1, 2, 3, 4, 5, 6\}$. For any $T \in U$, let $|T|$ denote the number of elements in T and T_1 denote the complement of T. For any T, $R \in U$, let $T \backslash R$ be the set of all elements in T which are not in R. Which one of the following is true?

(a) $\forall X \in U\ (|X| = |X'|)$
(b) $\exists X \in U\ \exists\ Y \in U\ (|X| = 2, |Y| = 5$ and $X \cap Y = 0)$
(c) $\forall X \in U\ \forall\ Y \in U\ (|X| = 2, |Y| = 3$ and $X \backslash Y = 0)$
(d) $\forall X \in U\ \forall\ Y \in U\ (X \backslash Y = Y' \backslash X')$

Solution:

$$S = \{1, 2, 3, 4, 5, 6\}$$

Let $X = \{1,2\}$

$$X' = S - X = \{3, 4, 5, 6\}$$
$$|X| \neq |X'|$$

Option (a) is false.

Let $X = \{1, 2, 4, 5, 6\}$ and $Y = \{1, 2, 3, 4, 5\}$

$$X \cap Y = \{1, 2, 4, 5\} \neq \varnothing$$

Option (b) is incorrect.

Let $X = \{1, 2\}$ and $Y = \{2, 4, 5\}$

$$X / Y = \{1\} \neq \varnothing$$

Option (c) is incorrect.

So, option (d) is the correct answer.

Ans. (d)

3. Consider the relation $X(P, Q, R, S, T, U)$ with the following set of functional dependencies

$F = \{$

$\{P, R\} \rightarrow \{S, T\}$,

$\{P, S, U\} \rightarrow \{Q, R\}$

$\}$

Which of the following is the trivial functional dependency in F^+, where F^+ is closure of F?

(a) $\{P, R\} \rightarrow \{S, T\}$ (b) $\{P, R\} \rightarrow \{R, T\}$
(c) $\{P, S\} \rightarrow \{S\}$ (d) $\{P, S, U\} \rightarrow \{Q\}$

Solution: Trivial function dependency $X \rightarrow Y$ holds if $Y \subseteq X$.
From the given set of functional dependencies, only $\{P, S\} \rightarrow \{S\}$ holds true.

Because $S \subseteq PS$.

Ans. (c)

4. The maximum number of processes that can be in *Ready* state for a computer system with n CPUs is

(a) n (b) n^2
(c) 2^n (d) independent of n

Solution: Any number of processes can wait in Ready queue. It does not depend on number of CPUs available.

Ans. (d)

5. Among simple LR (SLR), canonical LR, and look-ahead LR (LALR), which of the following pairs identify the method that is very easy to implement and the method that is the most powerful, in that order

(a) SLR, LALR
(b) Canonical LR, LALR
(c) LSR, canonical LR
(d) LALR, canonical LR

Solution: CLR is the most powerful parsing. Simple LR is easy to implement as there is no overhead of look ahead symbols.

Ans. (c)

6. Let # be a binary operator defined as

$X \# Y = X' + Y'$ where X and Y are Boolean variables.

Consider the following two statements

S1: $(P \# Q) \# R = P \# (Q \# R)$

S2: $Q \# R = R \# Q$

Which of the following is/are true for the Boolean variables P, Q and R?

(a) Only S1 is true
(b) Only S2 is true
(c) Both S1 and S2 are true
(d) Neither S1 nor S2 are true

Solution:

$$X \# Y = X' + Y'$$

$$\text{S1: } (P \# Q) \# R = P \# (Q \# R)$$

LHS

$$(P' + Q')' + R'$$
$$PQ + R'$$

RHS

$$P' + (Q' + R')'$$
$$P' + QR$$

LHS $\neq$ RHS

S1 is False.

S2: $Q \# R = R \# Q$

$Q' + R' = R' + Q'$

S2 is True.

Ans. (b)

7. Consider a software project with the following information domain characteristics for calculation of function point metric.

Number of external inputs (I) = 30

Number of external output (O) = 60

Number of external inquiries (E) = 23

Number of files (F) = 08

Number of external interfaces (N) = 02

It is given that the complexity weighting factors for I, O, E, F and N are 4, 5, 4, 10 and 7, respectively. It is also given that, out of fourteen value adjustment factors that influence the development effort, four factors are not applicable, each of the other four factors have value 3, and each of the remaining factors have value 4. The computed value of function point metric is _____.

Solution:

Number of external inputs (I) = 30 × 4 = 120

Number of external outputs (O) = 60 × 5 = 300

Number of external inquiries (E) = 23 × 4 = 92

Number of files (F) = 08 × 10 = 80

Number of external interfaces (N) = 02 × 7 = 14

Total unadjusted function points = 606

Ci = 4 × 3 + 6 × 4 = 36

VAF = 0.65 + [(Ci)/100] = 0.65 + 0.36 = 1.01

FP = UFP × VAF ⇒ FP = 606 × 1.01 = 612.06

Ans. (612.06)

8. In a web server, ten WebPages are stored with the URLs of the form

http://www.yourname.com/var.html; where, var is a different number from 1 to 10 for each Webpage. Suppose, the client stores the Webpage with var = 1 (say W1) in local machine, edits and then tests. Rest of the WebPages remains on the web server. W1 contains several relative URLs of the form "var.html" referring to the other WebPages. Which one of the following statements needs to be added in W1, so that all the relative URLs in W1 refer to the appropriate WebPages on the web server?

(a) <a href: "http://www.yourname.com/", href: "…var.html">
(b) <base href: "http://www.yourname.com/">
(c) <a href: "http://www.yourname.com/">
(d) <base href: "http://www.yourname.com/", range "…var.html">

Solution: Base tag with href is used: < base href :" http : // www.yourname.com / ", range :"...var. html" >

Ans. (b)

9. Consider the following statements

I. TCP connections are full duplex
II. TCP has no option for selective acknowledgement
III. TCP connections are message streams

(a) Only I is correct
(b) Only I and III are correct
(c) Only II and III are correct
(d) All of I, II, and III are correct

Solution:

I. True, TCP connection is full duplex connection because sender and receiver can exchange data in the same connection.

II. False, there is no selective acknowledgement in TCP, concept of cumulative acknowledgement is used.

III. False, Data is sent in byte streams.

Ans. (a)

10. Consider the equality $\sum_{i=0}^{n} i^3 = X$ and the following choices for X:

I. $\Theta(n^4)$ II. $\Theta\ (n^5)$
III. $O(n^5)$ IV. $\Omega(n^3)$

The equality above remains correct if X is replaced by

(a) Only I
(b) Only II
(c) I or III or IV but not II
(d) II or III or IV but not I

Solution:

$$X = 1^3 + 2^3 + 3^3 + 4^3 + \cdots + n^3 = \frac{n^2(n+1)^2}{4}$$

This is equivalent to $\Theta\ (n^4)$, $O(n^5)$ and $\Omega(n^3)$.

Ans. (c)

11. Consider a binary tree T that has 200 leaf nodes. Then, the number of nodes in T that have exactly two children are _____.

Solution:
Leaf $(p) = 200$

Total nodes $= 2p - 1 = 399$

Nodes with degree exactly two children $= 399 - 200 = 199$

Ans. (199)

12. Given a hash table T with 25 slots that stores 2000 elements, the load factor α for T is _____.

Solution:

$$\text{Load factor } (\alpha) = \frac{\text{Size}}{\text{Complexity}}$$

Size: Number of elements = 2000

Capacity: Number of slots = 25

$$\alpha = \frac{2000}{25} = 80$$

Ans. (80)

13. In the given matrix $\begin{bmatrix} 1 & -1 & 2 \\ 0 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix}$, one of the Eigen values is 1. The Eigen vectors corresponding to the Eigen value 1 are

(a) $\{\alpha(4, 2, 1) | \alpha \neq 0, \alpha \in \mathbb{R}\}$
(b) $\{\alpha(-4, 2, 1) | \alpha \neq 0, \alpha \in \mathbb{R}\}$
(c) $\{\alpha(2, 0, 1) | \alpha \neq 0, \alpha \in \mathbb{R}\}$
(d) $\{\alpha(-2, 0, 1) | \alpha \neq 0, \alpha \in \mathbb{R}\}$

Solution: Let X be the eigen vector corresponding to the eigen value $\lambda = 1$

Apply $AX = \lambda X$

Thus, we get Eigen vectors as $\{\alpha\ (-4,2,1) | \alpha \neq 0, \alpha \in \mathbb{R}\}$

Ans. (b)

14. The value of $\lim_{x\to\infty}\left(1+x^2\right)e^{-x}$ is _____.

(a) 0 (b) $\frac{1}{2}$

(c) 1 (d) ∞

Solution:

$$\lim_{x\to\infty}\left(1+x^2\right)e^{-x} = (\infty)^0 = 1$$

Ans. (c)

15. The number of 4 digit numbers having their digits in non-decreasing order (from left to right) constructed by using the digits belonging to the set {1, 2, 3} is _____.

Solution: 4 digit numbers having their digits in non-decreasing order (from left to right) from set {1, 2, 3}
Starting with 1: 1111, 1112, 1113, 1122, 1123, 1133, 1222, 1223, 1233, 1333 = 10

Starting with 2: 2222, 2223, 2233, 2333 = 4

Starting with 3: 3333 = 1

Total = 10 + 4 + 1 =15

Ans. (15)

16. In a room there are only two types of people, namely Type 1 and Type 2. Type 1 people always tell the truth and Type 2 people always lie. You give a fair coin to a person in that room, without knowing which type he is from and tell him to toss it and hide the result from you till you ask for it. Upon asking, the person replies the following

"The result of the toss is head if and only if I am telling the truth."

Which of the following options is correct?

(a) The result is head
(b) The result is tail
(c) If the person is of Type 2, then the result is tail
(d) If the person is of Type 1, then the result is tail

Solution: Type 1 people always tell the truth. Type 2 people always lie.

Statement: "The result of the toss is head if and only if I am telling the truth."

Option (a): This statement extracts that the person is of Type 1 and the result is head.

Option (b): We cannot draw conclusion about the Type of person. So, it is incorrect.

Option (c): If person is of Type 2, then he is telling lie that toss is head. Option (c) is incorrect.

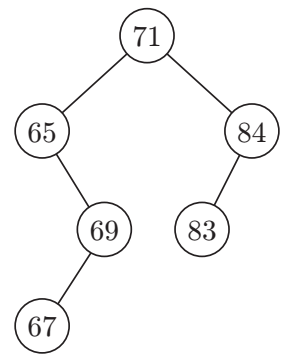
Option (d): If person is of Type 1, then he is telling the truth that toss is head. Hence, this is incorrect.

Ans. (a)

17. While inserting the elements 71, 65, 84, 69, 67, 83 in an empty binary search tree (BST) in the sequence shown, the element in the lowest level is

(a) 65 (b) 67
(c) 69 (d) 83

Solution: 67 is at the lowest level.

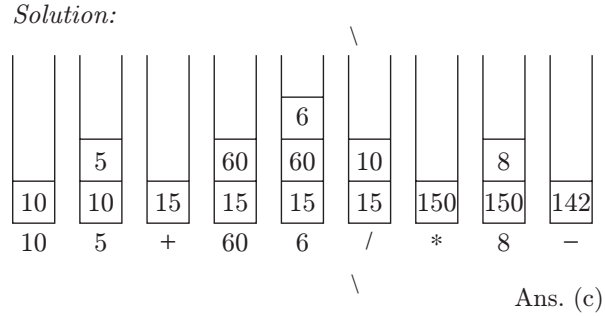


Ans. (b)

18. The result evaluating the postfix expression 10 5 + 60 6 / * 8 − is

(a) 284 (b) 213
(c) 142 (d) 71

Solution:



Ans. (c)

19. Consider the following relation

Cinema (theater, address, capacity)

Which of the following options will be needed at the end of the SQL query?

SELECT P1. address

FROM Cinema P1

Such that it always finds the addresses of theaters with maximum capacity?

(a) WHERE P1. capacity > = All (select P2. capacity from Cinema P2)
(b) WHERE P1. capacity > = Any (select P2. capacity from Cinema P2)
(c) WHERE P1. capacity > All (select max (P2. capacity) form Cinema P2)
(d) WHERE P1. capacity > Any (select max (P2. capacity) from Cinema P2)

Solution: Inner query selects the capacity of Theaters, > = All compares it with the all the values. So, option (a) will find address of theaters with maximum capacity.

Ans. (a)

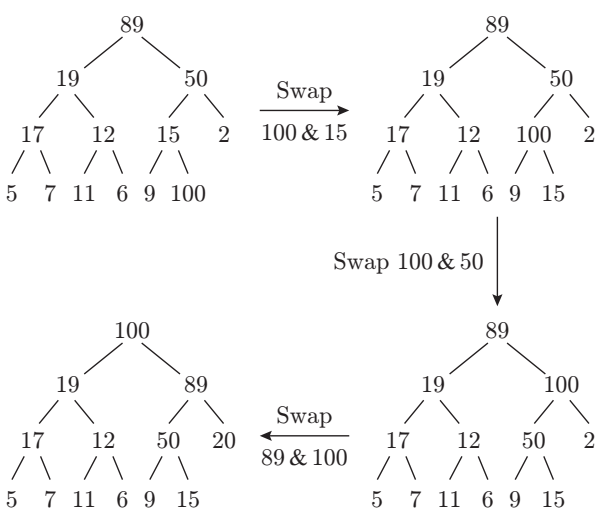
20. Consider the following array of elements

<89, 19, 50, 17, 12, 15, 2, 5, 7, 11, 6, 9, 100>

The minimum number of interchanges needed to convert it into a max-heap is

(a) 4 (b) 5
(c) 2 (d) 3

Solution:



Ans. (d)

21. Two processes X and Y need to access a critical section. Consider the following synchronization construct used by both the processes

Process X

```
/* other code for process X */
while (true)
{
varP = true;
while(varQ == true)
{
/* Critical Section */
varP = false;
}
}
/* other code for process X */
```

Process Y

```
/* other code for process Y */
while (true)
{
varQ = true;
while(varP == true)
{
/* Critical Section */
varQ = false;
}
}
/* other code for process Y */
```

Here, varP and varQ are shared variables and both are initialized to false. Which one of the following statements is true?

(a) The proposed solution prevents deadlock but fails to guarantee mutual exclusion
(b) The proposed solution guarantees mutual exclusion but fails to prevent deadlock
(c) The proposed solution guarantees mutual exclusion and prevents deadlock
(d) The proposed solution fails to prevent deadlock and fails to guarantee mutual exclusion

Solution: Proposed solution does not ensure mutual exclusion, because both varP and varQ can be true at the same time and enter to the critical section. But there is no deadlock state in this code, because X and Y do not lock required variables.

Ans. (a)

22. Let L be the language represented by the regular expression $\Sigma^*0011\Sigma^*$ where $\Sigma = \{0, 1\}$. What is the minimum number of states in a DFA that recognizes $\overline{L}$ (complement of L)?

(a) 4 (b) 5
(c) 6 (d) 8

Solution: 5 $(n + 1)$ states are sufficient to represent 0011.

Ans. (b)

23. Consider a software program that is artificially seeded with 100 faults. While testing this program, 159 faults are detected, out of which 75 faults are from those artificially seeded faults. Assuming that both real and seeded faults are of same nature and have same distribution, the estimated number of undetected real faults is _____.

Solution:

N: Non-seeded errors = 159 − 75 = 84
S: Seeded errors =100

s: Seeded errors detected = 75

$$\frac{N(S-s)}{s} = \frac{84(100-75)}{75} = 28$$

Ans. (28)

24. Consider a machine with a byte addressable main memory of 2^{20} bytes, block size of 16 bytes and a direct mapped cache having 2^{12} cache lines. Let the addresses of two consecutive bytes in main memory be $(E201F)_{16}$ and $(E2020)_{16}$. What are the tag and cache line address (in hex) for main memory address $(E201F)_{16}$?

(a) E, 201 (b) F, 201
(c) E, E20 (d) 2, 01F

Solution: Total bits = 20

Word

4	12	4

Ans. (a)

25. Consider a CSMA/CD network that transmits data at a rate of 100 Mbps (10^8 bits per second) over a 1 km (kilometer) cable with no repeaters. If the minimum frame size required for this network is 1250 bytes, what is the signal speed (km/sec) in the cable?

(a) 8000 (b) 10000
(c) 16000 (d) 20000

Solution: Frame size = 1250 bytes, Bandwidth = 100 Mbps, Distance = 1 km, Signal Speed = ?

In CSMA/CD,

$$L = \frac{2dB}{V} \Rightarrow V = 2\,\text{dB/L} = 20{,}000 \text{ km/sec}$$

Ans. (d)

Q. No. 26 – 55 Carry Two Marks Each

26. The velocity v (in kilometer/minute) of a motorbike which starts from rest, is given at fixed intervals of time t(in minutes) as follows:

t	2	4	6	8	10	12	14	16	18	20
v	10	18	25	29	32	20	11	5	2	0

The approximate distance (in kilometers) rounded to two places of decimals covered in 20 minutes using Simpson's 1/3rd rule is _____.

Solution: Simpson's 1/3rd rule:

$H = 2$

$$S = \frac{2}{3[(0+0)+2(18+29+20+5)+4(10+25+32+11+2)]} = 309.33$$

Ans. (309.33)

27. Assume that a mergesort algorithm in the worst case takes 30 seconds for an input of size 64. Which of the following most closely approximates the maximum input size of a problem that can be solved in 6 minutes?

(a) 256 (b) 512
(c) 1024 (d) 2048

Solution: Merge sort worst case time complexity = O($n$log n)

For input of size 64, it takes 30 seconds

$$n \log n = 30 \text{ for } n = 64$$
$$64 \log 64 = 30 \text{ seconds}$$

For 6 minutes, factor is 12. So, multiply by 12 on both sides

$$12 \times (64 \log 64) = 12 \times 30 \text{ seconds}$$
$$512 \log 512 \text{ generates the factor of } 12$$

So, maximum 512 size array can be sorted.

Ans. (b)

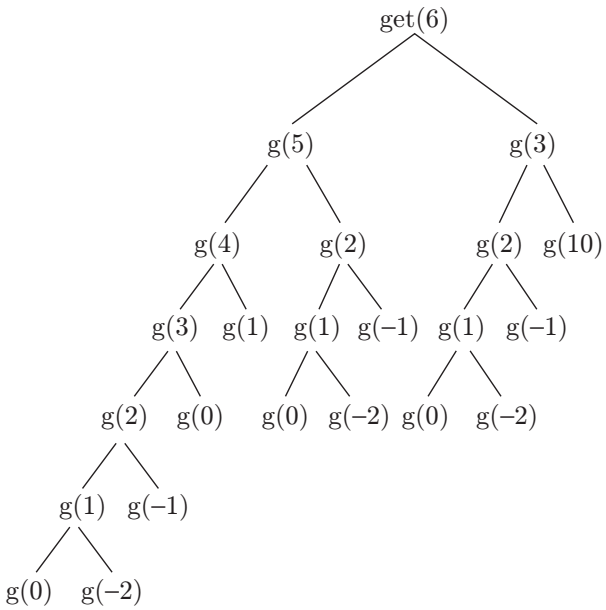
28. Consider the following recursive C function.

```
void get(int n)
{
if (n<1)return;
if (n<1) return;
get (n-1);
get (n-3);
printf("%d", n);
}
```

If get (6) function is being called in main() then how many times will the get() function be invoked before returning to the main()?

(a) 15 (b) 25
(c) 35 (d) 45

Solution:



Ans. (b)

29. Consider a B+ tree in which the search key is 12 bytes long, block size is 1024 bytes, record pointer is 10 bytes long and block pointer is 8 bytes long. The maximum number of keys that can be accommodated in each non-leaf node of the tree is _____.

Solution: In B+ trees, if order of tree is p then for internal nodes:
p(child pointer) + $(p-1)$(Key value) $\leq$ Block size

$p(8) + (p-1)12 \leq 1024$

$20p - 12 \leq 1024$

$p \leq 51 \Rightarrow p = 51$

So, for 51 order tree, we can save maximum 50 keys.

Ans. (50)

30. Given the function $F = P' + QR$, where F is a function in three Boolean variables P, Q and R and $P' = !P$, consider the following statements.

(S1) $F = \sum (4, 5, 6)$

(S2) $F = \sum (0, 1, 2, 3, 7)$

(S3) $F = \prod (4, 5, 6)$

(S4) $F = \prod (0, 1, 2, 3, 7)$

Which of the following is true?

(a) (S1)−False, (S2)−True, (S3)−True, (S4)−False

(b) (S1)−True, (S2)−False, (S3)−False, (S4)−True

(c) (S1)−False, (S2)−False (S3)−True, (S4)−True

(d) (S1)−True, (S2)−True, (S3)−False, (S4)−False

Solution:

$F = P' + QR = P'(Q + Q')(R + R') + (P+P')QR$

$F = P'Q + P'Q'(R + R') + PQR + P'QR$

$F = P'QR + P'QR' + P'Q'R + P'Q'R' + PQR + P'QR$

P \ QR	00	01	11	10
0	1	1	1	1
1	0	0	1	0

$$F = \sum(0, 1, 2, 3, 7)$$
$$= \prod(4, 5, 6)$$

S2 and S3 are true.

Ans. (a)

31. Language L_1 is polynomial time reducible to language L_2. Language L_3 is polynomial time reducible to L_2, which in turn is polynomial time reducible to language L_4.

Which of the following is/are true?

I. if $L_4 \in P$, then $L_2 \in P$
II. if $L_1 \in P$ or $L_3 \in P$, then $L_2 \in P$
III. $L_1 \in P$, if and only if $L_3 \in P$
IV. if $L_4 \in P$, then $L_1 \in P$ and $L_3 \in P$

(a) II only (b) III only
(c) I and IV only (d) I only

Solution: Given, $L_1 \leq_P L_2$

$L_3 \leq_P L_2$

$L_2 \leq_P L_4$

If L_4 is polynomial, then it implies that L_2 is also polynomial. Similarly, if L_2 is polynomial this also implies that L_1 and L_3 are also polynomial.

Ans. (c)

32. Consider the following C program.

```
#include <stdio.h>
int f1(void);
int f2(void);
int f3(void);
int x = 10;
int main()
{
int x = 1;
x + = f1() + f2() + f3() + f2();
printf ("%d", x);
return 0;
}
int f1() {int x = 25; x++; return x;}
int f2() {static int x = 50; x++; return x;}
int f3() {x *= 10; return x};
```

The output of the program is _____.

Solution:

$$x = x + f1() + f2() + f3() + f2()$$

After evaluating the functions through the given code, we get

f1() ⇒ 26

f2() ⇒ 51

f3() ⇒ 100

f2() ⇒ 52 [Static variable is there in f2(). Thus, its value is initialized only once and the value computed in 1st call of f2() is retained in the next call of function]

Ans. (230)

33. Consider the following C program.

```
#include <stdio.h>
int main()
{
static int a[] = {10, 20, 30, 40, 50};
static int *p[] = {a, a+3, a+4, a+1, a+2};
int **ptr = p;
ptr++;
printf("%d%d", ptr-p, **ptr);
}
```

The output of the program is _____.

Solution: a[] = {10,20,30,40,50}

p is a pointer to array. 1st element of p points to 10

2nd element points to 40

3rd element points to 50

4th element points to 20

5th element points to 30

Thus, implementing the code gives us 140.

Ans. (140)

34. Which of the following languages are context-free?

$L_1 = \{a^m b^n a^n b^m | m, n \geq 1\}$

$L_2 = \{a^m b^n a^m b^n | m, n \geq 1\}$

$L_3 = \{a^m b^n | m = 2n + 1\}$

(a) L_1 and L_2 only (b) L_1 and L_3 only
(c) L_2 and L_3 only (d) L_3 only

Solution: For identifying the context-free language, check whether the stack implementation is possible or not.

$$L_1 = \{a^m b^n a^n b^m | m, n \geq 1\}$$

Here when we put m number of a's and n number of b's in stack, top of stack will contain n number of b's. Thus, when n number of a's comes in the string then pop b's and when m number of b's comes then pop a's from stack. Thus, it is possible to count the a's and b's properly. Hence, this is a context-free language.

Similarly, check for others. Thus, L_1 and L_3 are context free.

Ans. (b)

35. Consider the following policies for preventing deadlock in a system with mutually exclusive resources.

I. Processes should acquire all their resources at the beginning of execution. If any resource is not available, all resources acquired so far are released.
II. The resources are numbered uniquely, and processes are allowed to request for resources only in increasing resource numbers
III. The resources are numbered uniquely, and processes are allowed to request for resources only in decreasing resource numbers
IV. The resources are numbered uniquely. A process is allowed to request only for a resource with resource number larger than its currently held resources

Which of the above policies can be used for preventing deadlock?

(a) Any one of I and III but not II or IV
(b) Any one of I, III, and IV but not II
(c) Any one of II, and III but not I or IV
(d) Any one of I, II, III, and IV

Solution: All the four methods can be used to prevent deadlock in the system.

Ans. (d)

36. In the network 200.10.11.144/27, the fourth octet (in decimal) of the last IP address of the network which can be assigned to a host is _____.

Solution:

IP address: 200.20.11.144 => 200.20.11.10010000

Subnet mask address: 225.225.225.11100000

Looking at the subnet address, we get to know that 1st three bits gives the subnet number and remaining are for host.

Thus, 100 10000

Last host bits will be: 100 11110 ⇒ 158

Ans. (158)

37. Consider a network connecting two systems located 8000 kilometers apart. The bandwidth of the network is 500×10^6 bits per second. The propagation speed of the media is 4×10^6 meters per second. It is needed to design a Go Back − N sliding window protocol for this network. The average packet size is 10^7 bits. The network is to be used to its full capacity. Assume that processing delays at nodes are negligible. Then, the minimum size in bits of the sequence number field has to be _____.

Solution:

Distance = 8000 km = 8×10^6 m

Speed = 4×10^6 m/s

Transmission time = Distance/Speed = 2 s

Bandwidth = 500×10^6 bps

Packet size = 10^7 bits

500×10^6 bits ⇒1 s

10^7 bits ⇒ 10^7 / 500×10^6 = 1/50 s

Propagation time = 1/50 s ⇒ RTT = 2/50 = 1/25 s

Full capacity network utilization is done. Therefore, efficiency is 100%.

$$\eta = \frac{w}{(1+2a)} = 1 \Rightarrow w = 1 + 2a$$

$$a = \frac{2}{0.02} = 100$$

$$2^n = 1 + 200 = 201 \Rightarrow n = 8$$

Ans. (8)

38. Consider the following reservation table for a pipeline having three stages S_1, S_2, and S_3.

	Time→				
	1	2	3	4	5
S_1	X				X
S_1		X		X	
S_3			X		

The minimum average latency (MAL) is _____.

Solution: For S_1 latency = 0 + 4

For S_2 latency = 1 + 3

For S_3 latency = 2

Average latency = 10/3 = 3.33

Ans. (3.33)

39. Consider the following code sequence having five instructions I_1 to I_5. Each of these instructions has the following format.

OP Ri, Rj, Rk

where operation Op is performed on contents of registers Rj and Rk and the result is stored in register Ri.

I_1: ADD R1, R2, R3

I_2: MUL R7., R1, R3

I_3: SUB R4, R1, R5

I_4: ADD R3, R2, R4

I_5: MUL R7, R8, R9

Consider the following three statements.

S1:There is an anti-dependence between instruction I_2 and I_5

S2:There is an anti-dependence between instructions I_2 and I_4

S3:Within an instruction pipeline an anti-dependence always creates one or more stalls

Which one of above statements is/are correct?

(a) Only S1 is true
(b) Only S2 is true
(c) Only S1 and S3 are true
(d) Only S2 and S3 are true

Solution: There occurs a name dependency here (write after read). This is anti-dependency and it always causes a stall of one or more cycles.

Ans. (b)

40. Consider the following two C code segments. Y and X are one- and two-dimensional arrays of size n and $n \times n$, respectively, where $2 \le n \le 10$. Assume that in both code segments, elements of Y are initialized to 0 and each element X[i] [j] of array X is initialized to i+j. Further assume that when stored in main memory all elements of X are in same main memory page frame.

Code segment 1:

//initialize elements of Y to 0

//initialize elements X[i] [j] of X to i+j

for (i = 0; i < n; i++)

Y[i] += X[0] [i];

Code segment 2:

//initialize elements of Y to 0

//initialize elements X[i] [j] of X to i+j

```
for (i = 0; i < n; i++)
Y[i] += X[i][0];
```

Which of the following statements is/are correct?

S1: Final contents of array Y will be same in both code segments

S2: Elements of array X accessed inside the for loop shown in code segment 1 are contiguous in main memory

S3: Elements of array X accessed inside the for loop shown in code segment 2 are contiguous in main memory

(a) Only S2 is correct
(b) Only S3 is correct
(c) Only S1 and S2 are correct
(d) Only S1 and S3 are correct

Solution: Y is initialized with all zeros. X is initialized with sum of its indices.

Y[i] + = x[0] [i] and Y[i] + = x[i] [0] will contain same values. Statement S1 is true. S2 is true because array is stored as x[0][0], x[0][1], x[0][2]…x[1][0], x[1][2] and so on.

Ans. (c)

41. Consider the following partial Schedule S involving two transactions T1 and T2. Only the read and the write operations have been shown. The read operation on data item P is denoted by read (P) and the write operation on data item P is denoted by write (P).

Time instance	Transaction -id	
	T1	T2
1	read(A)	
2	write(A)	
3		recd(C)
4		write(C)
5		read(B)
6		write(B)
7		read(A)
8		commit
9	read(B)	

Schedule S

Suppose that the transaction T1 fails immediately after time instance 9. Which one of the following statements is correct?

(a) T2 must be aborted and then both T1 and T2 must be re-started to ensure transaction atomicity
(b) Schedule S is non-recoverable and cannot ensure transaction atomicity
(c) Only T2 must be aborted and then restarted to ensure transaction atomicity
(d) Schedule S is recoverable and can ensure atomicity and nothing else needs to be done

Solution: T2 transaction reads the value written by T1 transaction. T2 is committed before T1 and T1 fails. Thus, T2 has read that value of A which no more exists in the database. Hence, the schedule S is non-recoverable schedule. Also, it doesn't ensure atomicity.

Ans. (b)

42. If the following system has non-trivial solution

$$px + qy + rz = 0$$
$$qx + ry + pz = 0$$
$$rx + py + qz = 0$$

then which one of the following options is TRUE?

(a) $p - q + r = 0$ or $p = q = -r$
(b) $p + q - r = 0$ or $p = -q = r$
(c) $p + q + r = 0$ or $p = q = r$
(d) $p - q + r = 0$ or $p = -q = -r$

Solution: For non-trivial solution, $|A| = 0$. Applying this on the given matrix, we get answer (c).

Ans. (c)

43. Consider the following C program:

```
# include<stdio.h>
int main( )
{
int i, j, k = 0;
j = 2 * 3 / 4 + 2.0 / 5 + 8 / 5;
k -= --j;
for (i = 0; i < 5: i++)
{
switch (i + k)
{
case 1:
case 2: printf (" n%d", i+k);
case 3: printf ("n%d", i+k);
default: printf ("n%d", i+k);
}
}
return 0;
}
```

The number of times printf statement is executed is _____.

Solution: j and k will be evaluated to be 2 and -1, respectively. After getting in the loop, and implementing every iteration, the printf is called 10 times.

Ans. (10)

44. If for non-zero x, $af(x) + bf\left(\frac{1}{x}\right) = \frac{1}{x} - 25$ where $a \neq b$ then $\int_1^2 f(x)\,\mathrm{d}x$ is

(a) $\frac{1}{a^2 - b^2}\left[a(\ln 2 - 25) + \frac{47b}{2}\right]$

(b) $\frac{1}{a^2 - b^2}\left[a(2\ln 2 - 25) - \frac{47b}{2}\right]$

(c) $\frac{1}{a^2 - b^2}\left[a(2\ln 2 - 25) + \frac{47b}{2}\right]$

(d) $\frac{1}{a^2 - b^2}\left[a(\ln 2 - 25) - \frac{47b}{2}\right]$

Solution:

$$af(x) + bf(1/x) = \frac{1}{x - 25} \qquad (1)$$

Replacing x by $1/x$, we get

$$af\,(1/x) + bf\,(x) = x - 25 \qquad (2)$$

Multiply Eq. (1) by a and Eq. (2) by b, and then subtract them, to get

$$f\,(x) = [1/(a^2 - b^2)\,][a/x - bx + 25(b - a)]$$

$$\int_1^2 f(x)\,\mathrm{d}x = \left[\frac{1}{(a^2 - b^2)}\right] \times \left[a(\ln x - 25x) - b\left(\frac{x^2}{2} - 25x\right)\right]_1^2$$

Solving this equation, we get the answer (c).

Ans. (a)

45. Let G be a connected undirected graph of 100 vertices and 300 edges. The weight of a minimum spanning tree of G is 500. When the weight of each edge of G is increased by five, the weight of a minimum spanning tree becomes _____.

Solution: In a spanning tree containing 100 vertices, number of edges will be 99.

Therefore, weight is increased by = 500 + 5 × 99 = 995

Ans. (995)

46. Two hosts are connected via a packet switch with 10^7 bits per second links. Each link has a propagation delay of 20 microseconds. The switch begins forwarding a packet 35 microseconds after it receives the same. If 10000 bits of data are to be transmitted between the two hosts using a packet size of 5000 bits, the time elapsed between the transmission of the first bit of data and the reception of the last bit of the data in microseconds is _____.

Solution: Transmission time of 5000 bits = 500 microseconds

Total transmission time of first bit = 500 + 20 + 35 + 20 = 575 microseconds

Total transmission time of last bit = 500 + 500 = 1000 microseconds

Total time = 1000 + 575 = 1575 microseconds

Ans. (1575)

47. For the processes listed in the following table, which of the following scheduling schemes will give the lowest average turnaround time?

Process	Arrival Time	Processing Time
A	0	3
B	1	6
C	4	4
D	6	2

(a) First Come First Serve
(b) Non-preemptive Shortest Job First
(c) Shortest Remaining Time
(d) Round Robin with Quantum value two

Solution:

Process	Arrival Time	Processing Time
A	0	3
B	1	6
C	4	4
D	6	2

Shortest remaining time first:

A	B	C	D	B

0 3 4 8 10 15

Turnaround time = Completion time − Arrival time

A ⇒ 3 − 0 = 3

B ⇒ 15 − 1 = 14

C ⇒ 8 − 4 = 4

D ⇒ 10 − 6 = 4

$$\text{Average turnaround time} = \frac{3+14+4+4}{4} = 6.25$$

Similarly checking for other cases, we will find that average turnaround time is minimum in SRTF.

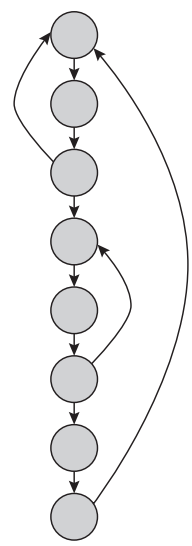
Ans. (c)

48. Consider three software items: Program–X, Control Flow Diagram of Program–Y and Control Flow Diagram of Program–Z as shown below

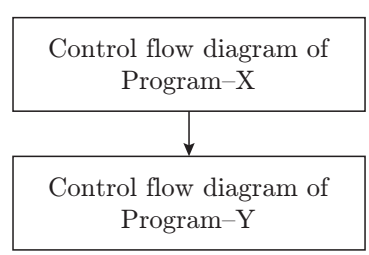
Program X:

```
sumcal(int maxint, int value)
{
int result =0, i = 0;
if(value < 0)
{
value = -value;
}
while((i< value) && (result < = maxint))
}
i = i +1;
result = result + 1;
}
if (result < = maxint)
{
printf(result);
}
else
{
printf("large");
}
printf("end of program");
}
```

Control Flow Diagram of Program − Y:



Control Flow Diagram of Program − Z:



The values of McCabe's Cyclomatic complexity of Program–X, Program–Y, and Program–Z, respectively, are

(a) 4, 4, 7
(b) 3, 4, 7
(c) 4, 4, 8
(d) 4, 3, 8

Solution: From Program–X:

Number of conditions and control statements = 3

Complexity = 3 + 1 = 4

Control Flow Diagram of Program–Y:

Complexity = Edges − Number of vertices + 2
= 10 − 8 + 2 = 4

Control Flow Diagram of Program–Z:

Complexity = X + Y − 1 = 4 + 4 − 1 = 7

Ans. (a)

49. Consider the equation $(43)_x = (y3)_8$ where x and y are unknown. The number of possible Solutions is _____.

Solution:

$$(43)_x = (y3)_8$$
$$\Rightarrow 4x + 3 = 8y + 3$$
$$\Rightarrow x = 2y$$

Moreover, $x \geq 5$ and $y \leq 7$

Hence, possible values of x and y are: (14,7), (12,6), (10,5), (8,4), (6,3)

Ans. (5)

50. Let R be a relation on the set of ordered pairs of positive integers such that ((p,q), (r,s)) ∈ R if and only if p−s = q−r. Which one of the following is true about R?

(a) Both reflexive and symmetric
(b) Reflexive but not symmetric
(c) Not reflexive but symmetric
(d) Neither reflexive nor symmetric

Solution:

$$((p,q)\ ,\ (r,s)) \in R$$
$$R \Rightarrow p - s = q - r$$

As we know, for reflexive we need (p,q) and (q,p) in relation R.

But $p - q \neq q - p$, for example, (4, 2) and (2, 4) $\Rightarrow 4 - 2 = 2$ and $2 - 4 = -2$

Hence, relation is not reflexive.

Similarly, for symmetric if (p,q) R (q,s) $\Rightarrow p - s = q - r$

Also, $r - q = s - p \Rightarrow$ (r,s) R (p,q)

Hence, it is symmetric.

Ans. (c)

51. Suppose X_i for $i = 1, 2, 3$ are independent and identically distributed random variables whose probability mass functions are

$$\Pr[X_i = 0] = \Pr[X_i = 1] = 1/2 \text{ for } i = 1, 2, 3.$$

Define another random variable $Y = X_1\ X_2 \oplus X_3$, where ⊕ denotes XOR. Then $\Pr[Y = 0|X_3 = 0] =$ _____.

Solution:
$X_3 = 0$ and $Y = X_1\ X_2$ XOR $X_3 \Rightarrow X_1\ X_2 = 0$

$X_1 = 0$ and $X_2 = 0$, $X_1 = 1$ and $X_2 = 0$, $X_1 = 0$ and $X_2 = 1$

$$P[Y{=}0/X_3 = 0] = P[(Y{=}0)\cap(X_3 = 0)]/P[X_3 = 0]$$
$$= (1/2)^3 \times 3/(1/2) = 3/4 = 0.75$$

Ans. (0.75)

52. The total number of prime implicants of the function $f(w, x, y, z) = \Sigma(0, 2, 4, 5, 6, 10)$ is _____.

Solution:

$wx \backslash yz$	00	01	11	10
00	*1	0	0	1
01	1	1*	0	1*
11	0	0	0	0
10	0	0	0	1*

Ans. (3)

53. Suppose c = ⟨c[0], …., c[k-]⟩ is an array of length k, where all the entries are from the set {0, 1}. For any positive integers a and n, consider the following pseudocode.

```
DOSOMETHING (c, a, n)
z ← 1
for i ← 0 to k - 1
do z ← z^2 mod n
if c[i] = 1
then z ← (z × a) mod n
return z
```

If $k = 4$, $c = \langle 1, 0, 1, 1\rangle$, $a = 2$ and $n = 8$, then the output of DOSOMETHING (c, a, n) is _____.

Solution: $k = 4$, $c = <1\ 0\ 1\ 1>$, $a = 2$ and $n = 8$
We call the function DOSOMETHING(c, a, n) and implement the code in the given steps.

After evaluation, we get answer 0.

Ans. (0)

54. Let $f(n) = n$ and $g(n) = n^{(1 + \sin n)}$, where n is a positive integer. Which of the following statements is/are correct?

I. $f(n) = O(g(n))$

II. $f(n) = \Omega(g(n))$

(a) Only I
(b) Only II
(c) Both I and II
(d) Neither I nor II

Solution:

$$f(n) = n \text{ and } g(n) = n^{1 + \sin n}$$

Minimum value of $g(n) = n^{1-1} = 1$

Maximum value of $g(n) = n^2$

In either case, it is not equal to $f(n)$. Hence, $f(n)$ can't be best or worst case for $g(n)$.

Ans. (d)

55. Consider the following grammar G

S → F | H

F → p | c

H → d | c

where S, F, and H are non-terminal symbols, p, d, and c are terminal symbols. Which of the following statement(s) is/are correct?

S1: LL(1) can parse all strings that are generated using grammar G

S2: LR(1) can parse all strings that are generated using grammar G

(a) Only S1
(b) Only S2
(c) Both S1 and S2
(d) Neither S1 nor S2

Solution: When we try to parse the given grammar with LL(1) parser or LR(1) parser, there occurs a conflict in both cases. Hence, it cannot be parsed with any of these parsers.

Ans. (d)

INDEX

A

Abstract data types, 119
Abstraction, 107
Addressing modes, 59–60
  auto-increment or auto-decrement mode, 60
  base register, 60
  direct mode, 60
  immediate mode, 59
  implied mode, 59
  index mode, 60
  indirect mode, 60
  register (direct) mode, 59
  register (indirect) mode, 59
  relative mode, 60
Adjacency list, 195
Adjacency matrix, 195
Algorithm
  analysis, 178
  asymptotic notation, 178–180
  defined, 177
  properties of, 177
  steps to solve a problem, 177
Algorithmic languages, 106
All-pair shortest path, 205
Alphabets of a language, 238
Ambiguities, removal of, 254
Ambiguous grammar, 254, 293
Amdahl's law, 58
AND gate, 6
Arithmetic logic unit (ALU)
  arithmetic micro-operations, 61
  logic micro-operations, 61–62
Arithmetic micro-operations, 61
Array
  address calculation in an, 110–111
  declaring, 110
  important points for, 110
  initialisation of one-dimensional, 110
  two-dimensional, 111
Asymptotic notation, 178–180
  big-O, 179
  complexities, 180
  omega, 179
  properties of, 180
  small-o, 180
  theta, 178
Asynchronous counter, 22
Asynchronous interrupts, 66

B

Banker's algorithm, 342–343
Base register addressing mode, 60
Bellman–Ford algorithm, 201
Binary heap, 137
  max-heap tree, 184
  min-heap tree, 184–185
  time complexity, 186
Binary number system, 3
Binary search
  pseudocode for, 187
  time complexities, 187
Binary search tree, 126
  traversal techniques, 126–127
Binary semaphores, 338
Binary trees, 125–126
  almost complete, 126
  array representation of, 126
  children, 125
  complete, 125–126
  edge, 125
  height, 125
  leaf node, 125
  level, 125
  node, 125
  parent, 125
  path, 125
  root, 125
  sibling, 125
  strictly, 125
  subtree, 125
Binding, 118–119
  early, 118–119
  late, 119
Bistable multivibrator, 16
4-bit synchronous counter, 23
  logic diagram for a, 23–24
  up counter, 22
Boolean algebra
  basic operations in, 6–7
  consensus theorem, 7
  duality theorem, 7
  huntington's postulates, 7
  negative logic, 8
  positive logic, 8
Boolean expression, 8
Boolean function, 15
  implementation using multiplexers, 15
Boolean logic, 6–10
Bubble sort algorithm
  pseudocode for, 187
  time complexities, 187

C

Cache memory, 73–74
  cache mapping techniques, 74–75
  cache size, 74
  direct mapping, 74
  full associative mapping, 74–75

mapping function, 74
set associative mapping, 75
write-back updating policies, 74
write-through policy, 74
Calling function, 120
Cascading Style Sheets (CSS), 541
Central processing unit (CPU)
data paths, 64
design, 64–65
general register organization, 62
hardwired control unit, 64
instruction execution, 63–64
micro-programmed control unit, 64–65
RISC *vs.* CISC processors, 65
single accumulator organization, 62
stack organization, 63
Chomsky normal form (CNF), 256
Class, 107
Clear accumulator (CLA), 59
Clear operation, 62
Client–server computing, 544–546
functions of a client, 544
functions of a server, 544–545
merits and demerits, 546
N-tier architecture, 546
server types, 544
three-tier architecture, 545–546
topologies for, 545
two-tier architecture, 545
Clocked *RS* flip–flops, 17
logic diagram of, 17
Code optimization phase
primary source of, 307–308
types and levels of, 307
Column major order, 111
Combinational circuits, 11–16
decoder, 14
demultiplexer, 16
encoder, 14
full adder, 12
full subtractor, 13
half adder, 11–12
half subtractor, 12
look-ahead carry adder, 13–14
multiplexer, 14
parallel adder, 13
Compiler, 287
construction tools, 290
grouping of phases, 290
phases of, 288–289
Complement accumulator (CMA), 59
Computer architecture, 57
quantitative principles to design high-performance processor, 58
registers and their functions, 57–58
Computer networks
application layer, 508–509
congestion and its causes, 505–506
connections, 491
devices used for internetworking, 509
digital signature, 511
firewalls, 511–512
International Standards Organization/Open Systems Interconnection (Reference Model 1984) (ISO/OSI) model, 494–498
LAN (local area network), 492–493
network layer protocols, 501–505
network security, 510–512
presentation layer, 508
routing algorithms, 498–499
session layer, 507
sockets, 507
topology, 492
Transmission Control Protocol (TCP), 506–507
transport layer, 505
User Datagram Protocol (UDP), 506
Computer organization, 57
Concatenation, 238
Concurrency control protocols, 423–424
Consensus theorem, 7
Constructive cost model (COCOMO), 463–464
Context-free grammar, 252
Context-free grammar simplification, 254–256
Control flow statements
break statement, 110
conditional statements, 108–110
continue statement, 110
goto statement, 110
return statement, 110
Control hazards, 69
Conversions of number system, 4
decimal number system to any number system, 4
number system to decimal number system, 4
Counters, 22–26
Counting semaphores, 338–339
Counting sort
pseudocode for, 193
time complexities, 192
CPU scheduling algorithms, 331–335
Critical section problem, 337
Cryptography, 510–511

D

Database design
attribute closure, 411
decomposition process, 412–414
integrity constraints, 410–411
keys, 411–412
process of normalization, 411
Database management system (DBMS), 403–404
architecture of, 405
components of, 404–405
Data models, 405
ER model, 407–410
relational model, 406–407
Data path structures of CPU, 64
single bus structure, 64
three bus structure, 64
two bus structure, 64
Data types in programming language, 107–108
Deadlocks, 339
avoidance of, 342–343
characteristics of, 340
detection of, 346
prevention of, 341–342
receovery from, 346–347
resource allocation graph, 340–341
Decimal number system, 3
Decoder, 14
logic diagram of, 14
Defect rate, 464
Defect removal efficiency (DRE), 464
Degree of multiprogramming, 330
Demultiplexer, 16
Derivation tree, 253
D flip–flops, 17–18
truth table of, 18
Digital circuits
classification of, 11
combinational circuits, 11–16
sequential circuits, 16–26
Digital electronics, 3
Digital logic families, 26–27
pros and cons of logic families, 27
Dijkstra's algorithm, 200–201
Dining philosopher's problem, 336
Direct memory access (DMA), 66–67
burst or block transfer mode, 66
channel, 66
controller, 66
cycle stealing mode, 66
data transfer mode, 66
interconnection with memory, CPU and I/O devices, 67
modes of operation, 66
Dispatcher, 330
Do … while loop, 109
Duality theorem, 7
Dynamic programming, 203–204
applications of, 204–2057

E

Early binding, 118–119
Else–if condition statement, 109
Encapsulation, 107
Encoder, 14
Enumerated data types, 117
Ethernet LAN technology, 492–493
Even palindrome, 238
Expression conversion, 120
infix to postfix conversion, 120
infix to prefix conversion, 121–122
External interrupts, 65

F

Fibonacci series, 204
File processing approach, 403–404
File structures
B trees, 416
B+ tree structure, 416–417
indexing attributes, 416
sequential files, 416
File system
allocation methods, 359–362
directory structures, 359
Finite automata, 237–238
conversion in, 246–249

deterministic finite automata (DFA), 241–243, 246–247, 250
grammar, 239–240
model characteristics, 238
non-deterministic finite automata (NFA), 241–243
technical terms, 238–239
Finite state machine (FSM)
mealy machines, 245
moore machines, 246
Flip–flops, 16
applications of, 20–26
clocked *RS*, 17
conversion equations, 19–20
D, 17–18
excitation table of, 19
JK, 18
obtained by K-map simplification, 17
RS, 16–17
T, 19
Floyd's algorithm, 205
Fork() function, 347–348
For loop, 109
Fractional knapsack, 196
pseudocode for, 196
time complexities, 196
Full adder, 12
logic diagram of, 12
Full subtractor, 13
logic diagram of a, 13
Function, 111
actual and formal arguments, 112
advantages, 112
call, 112
definition of, 112
parts of, 111–112
pre-defined, 112
prototype, 111–112
user-defined, 112
Function point analysis (FPA), 462

G

Global variables, 117–118
Grammar, 239–240
chomsky hierarchy, 239–240
regular, 244
regular expressions, 243–244
types, 240
Graphs
adjacent vertices, 127
AVL tree of, 135–136
balanced binary tree, 127
binary heap, 137
complete graph, 195
connected, 127
cycle, 127
degree of a vertex, 127
directed, 127
K_n graph, 195
multigraph, 194
null graph, 194
path, 127
regular graph, 195
representations of, 195
shortest paths, 132–133
simple graph, 194–195
spanning tree of, 133–135
subgraph, 127
traversal techniques for, 127–132
tree, 127
undirected, 127
Graph traversal
breath-first traversal (BFT), 201–202
depth-first traversal (DFT), 202–203
Greedy algorithm, 195
applications of, 196
important terms, 196
Greibach normal form (GNF), 256
Grouping of cells for simplification, 9

H

Half adder, 11–12
Half subtractor, 12
Hardware interrupts, 65
Hardwired control unit, 64
Hashing
collisions, 182–184
double, 183–184
functions, 182
hash table, 182
linear probing, 183
open addressing, 182–184
quadratic probing, 183
Heap sort
algorithm, 188–189
build heap function, 189
heapify function, 189
pseudocode for, 189
recurrence relation, 189
time complexities, 189
High-level languages, 106
HTML, 538–541
attributes, 538–539
backgrounds, 540
character entities, 539
elements, 539
formatting, 539
forms, 541
frames, 541
images, 540
links, 540
lists, 540
marquees, 541
phrase tags, 540
tables, 540–541
tags, 538
Huffman coding, 197
code tree according to, 198
principle of, 197
pseudocode for, 197
time complexities, 197
Huntington's postulates for Boolean algebra, 7

I

Identity element, 238
IEEE standard for floating point numbers, 5
double precision, 5
single precision, 5
If condition statement, 109
If–else condition statement, 109
Index address mode, 60
Information system, 453–454
components of, 454
decision support system (DSS), 455
expert system, 455
integrated, 455
management information system (MIS), 455
office information system (OIS), 454
transaction processing system (TPS), 454–455
Inheritance, 107
Insertion sort
pseudocode for, 188
recurrence relation for, 188
time complexities, 188
Insert operation, 62
Instruction execution
decoding instruction, 63
executing instruction, 63–64
fetching instruction, 63
operand fetching, 64
Instruction pipelining, 67–69
hazards in, 67–69
speed up of a pipelined system over a non-pipelined system, 67
Instruction set architecture (ISA), 57
Intermediate code generation, 305–307
Internal interrupts, 65
Interpreter, 287
Inter-process communication, 336
Interrupt initiated I/O, 65
I/O interface
direct memory access (DMA), 66–67
interrupts, 65–66
programmed, 65
I/O systems
disk scheduling algorithms, 366–368
disk structure, 363–364

J

J2EE platform, 546–547
application server, 547
architecture, 547
EJB container, 547
services, 546–547
JK flip–flops, 18
logic diagram, 18
master–slave, 18
truth table of, 18
Johnson counter, 26

K

Karnaugh-map (K-map), 9, 25
four-variable, 10
POS expression using, 9
SOP expression using, 9
three-variable, 9
two-variable, 9
0/1 knapsack, 204–205
Kruskal's algorithm, 199–200

L

Language complement, 239
Language intersections, 239
Languages, closure and decidability properties of, 258–259
Language subtractions, 239
Language union, 239
Language XOR, 239
Largest common subsequence, 205
Late binding, 119
Leftmost derivation (LMD) tree, 253
Length of a string, 238
Lexical analyzer
  functions of, 290
  implementation of, 290–291
LIFO (last in, first out) mechanism, 62
Linear search
  pseudocode for, 186
  time complexities, 186–187
Line of code (LOC), 462
Linked list
  basic operations, 123–124
  implementation of, 124–125
Local variables, 117
Logic gate, 6
  symbols, 6
Logic micro-operations, 61–62
Look-ahead carry adder, 13–14
  logic diagram of, 14
Loops, 109

M

Machine instructions, 58–59
  data manipulation instruction, 58
  data transfer instruction, 58
  one-address instruction, 59
  program control instruction, 58
  three-address instruction, 58
  two-address instruction, 59
  zero-address instruction, 59
Maskable interrupts, 65
Mask operation, 62
Master–slave *JK* flip–flops, 18
Maxterms, 8
Memory
  cache, 73–74
  main, 72
  m-bit main, 74
  secondary, 73
Memory management
  loading and linking, 349–350
  page replacement algorithms, 355–358
  paging, 350–354
  partition allocation policies, 348–349
  partitioning methods, 348
  segmentation technique, 354
  thrashing, 358
  virtual memory, 355
Merge sort
  applications of, 191
  pseudocode for, 190–191
  recurrence relation for, 191
  time complexities, 191
Metrics
  effort and schedule (duration) estimation, 463–465
  halstead, 465
  process, 461
  product, 461
  project, 461
  size-oriented, 462
Micro-programmed control unit, 64–65
  horizontal, 65
  vertical, 65
Micro-program sequencer, 65
Minimization of expresson, 8
Minimum spanning tree
  Kruskal's algorithm, 199–200
  Prim's algorithm, 198–199
Minterms, 8
MOD-10 asynchronous counter, 23–24
  logic diagram for a, 24
MOD-5 synchronous counter, 24
  logic diagram for a, 26
Multiplexer, 14
  block diagram of, 14
  boolean function implementation using, 15
Myhill–Nerode theorem, 245

N

NAND gate, 6
N-bit parallel adder, 13
Negative logic, 8
Non-maskable interrupts, 65
Non-procedural languages, 106
Non-vectored interrupts, 65
NOR gate, 6
NP-complete class, 206
NP-hard class, 206
Number system, 3–6
  binary, 3
  complement of a number, 4–5
  conversions of, 4
  decimal, 3
  IEEE standard for floating point numbers, 5
  representation of negative numbers, 5

O

Object-oriented languages, 106
Object-oriented programming (OOP), 106–107
Octet, 9
Odd palindrome, 238
Operating system (OS), 327
  batch, 328
  multiprocessor, 328
  multiprogramming, 328
  multitasking, 328
  real-time (RTOS), 329
  scheduler for, 330
OR gate, 6
Overloading, 107

P

Pairs, 9
Parallel adder, 13
  block diagram of a, 13
Parallel-in to parallel-out (PIPO), 22
Parallel-in to serial-out (PISO), 21
Parameter-passing techniques
  call-by-reference, 115–116
  call-by-value, 115
Parser, 291
  algorithm of predictive, 294
  bottom-up, 297–300
  canonical LR, 300–301
  context-free grammar (CFG) for, 292
  derivation tree or parse tree, 292
  leftmost derivation (LMD), 292
  operator precedence, 301
  rightmost derivation tree (RMD), 293
  top-down, 293–294
Parse tree, 253
Parsing, 120, 252
Parsing table, 295–296
P-complexity class, 205–206
Peterson's algorithm, 337–338
Pipelining hazards, 67–69
  data hazards, 68
  structural hazards, 68
Pointers, 113
  arithmetic, 113–114
  dangling, 114–115
Polymorphism, 107
Positive logic, 8
Power of a string, 238
Pre-defined function, 112
Prefix of a string, 239
Prime implicant, 8
Prim's algorithm, 198–199
Problem-oriented languages, 106
Procedural languages, 106
Procedural programming languages, 106
Process, 329
  attributes of, 329
  states, 329–330
  synchronization, 335–339
  time periods of, 330–331
Process models
  component-based model, 460–461
  incremental model, 459–460
  prototype model, 458–459
  rapid application development (RAD) model, 459
  spiral model, 460
  waterfall model, 458

Producer–consumer problem, 336
Product-of-sums (POS) form, 8
  using K-map, 9
Programmed I/O, 65
Programming languages, 105–106
  fifth generation /5GL/(natural language programming), 106
  first generation/1GL/(machine level programming), 106
  fourth generation /4GL/(non-procedural programming), 106
  second generation /2GL/(assembly level programming), 106
  third generation /3GL/(high-level programming), 106
Pumping lemma, 244–245
Pushdown automata (PDA)
  deterministic (NPDA), 251
  model of, 250–251
  non-deterministic (NPDA), 251
  transition function of, 251

Q

Quads, 9
Queue, 122
  applications of, 123
  basic operations, 122–123
  types of, 123
Quick sort
  pseudocode for, 192
  randomized version of, 193
  recurrence relation for, 192
  time complexities, 192

R

Race-around problem, 18
Race condition, 336–337
Radix complement, 4
  subtraction using, 6
Radix minus one complement, 4
  subtraction using, 4–5
Radix or base, 4
Read after write (RAW), 68
Readers–writers problem, 336
Recurrence relations
  iteration method, 180
  master theorem, 181
  recursion tree, 181
Recursion, 112
  advantages, 113
  function, 112–113
Recursive and recursive enumerable languages, 258
Recursive function, 120
Registers, 20
Regular sets/languages, properties of, 249
Relative address mode, 60
Return statement, 112
Reversal of a string, 238
Reversing a list, 120
Rightmost derivation (RMD) tree, 253
Ring counter, 26
Ripple carry adder, 13
Risk analysis
  identification of risk, 466
  risk projection or risk estimation, 466
  risk strategies, 466
  types of risk, 466
Row major order, 111
RS flip–flops, 16–17
  clocked, 17
  logic diagram of, 17
  truth table of, 17
Runtime environment
  activation record and activations trees, 304
  lexical *vs* dynamic scoping, 305
  procedure call return model, 304–305
  storage organization, 303–304
  symbol table, 305

S

Schedule, 417–419
  based on recoverability, 419
  based on serializability, 420–423
Scoping, 117
Scripting languages, 106
Searching, 186
Security of computers, 368
  from accidental data loss, 369
  anti-virus approaches, 371
  attacks from outside the system, 370–371
  attacks from within the system, 370
  cryptography, 369
  from threats, 369
Selection sort
  pseudocode for, 187
  time complexities, 188
Selective clear operation, 62
Selective complement operation, 62
Selective set operation, 62
Semaphores, 338
Sequential circuits, 16–26
Serial-in/serial-out shift register, 20–21
Serial-in to parallel-out (SIPO), 21
Shift left register, 21
Shift register counter, 26
Shift right register, 20
Sockets, 507
Software
  architecture, 472
  characteristics of, 456
  coding, 472
  design stage, 470–472
  engineering, 457
  interrupts, 65
  testing, 472–477
Software Development Life Cycle (SDLC)
  elicitation techniques, 469–470
  external interfaces, 468
  requirement, 467–468
  requirement engineering, 468–469
Sorting, 186
Spinning or busy waiting, 337
Stack, 119
  applications of, 120
  POP operation on, 119–120
  PUSH operation on, 119
Stack pointer (SP), 63
Standard context-free language, 252–253
Storage classes, 118
String of a language, 238
Structured programming, 107
Structured query languages (SQL), 414
  commands, 414–416
Structures, 117
Substring, 238
Suffix of a string, 239
Sum-of-products (SOP) form, 8
  using K-map, 9
Switch condition statement, 109
Synchronous counter, 22
Synchronous interrupts, 66
Syntax-directed translation (SDT)
  applications of, 302
  attributes supported by, 302
  characteristics, 302
  types of, 302

T

T flip–flops, 19
  truth table, 19
Threads
  benefits of, 347
  types of, 347–348
Three-variable K-map, 9
Token bus and ring, 493
Transactions, 417
Trees
  applications, 126
  binary, 125–126
  properties of, 125
Truth table, 12–14
Turing machine (TM), 256–258
Two-variable K-map, 9

U

Unions, 117
User-defined function, 112

V

Vectored interrupts, 65
Virtual memory, 355

W

Web technology
  Cascading Style Sheets (CSS), 541
  Document Object Model (DOM), 543
  flash, 543

milestones in development of, 537–538
silverlight, 543
user-interface languages, 543
World Wide Web (WWW), 537
While loop, 109
Write after read (WAR), 69
Write after write (WAW), 69

X

2X4 decoder, 14
XHTML, 543
XML, 541–543
advantages of, 542
documents, 543
tag rules for, 542
vocabularies, 542
XNOR gate, 6
XOR gate, 6
XUL, 543

The Graduate Aptitude Test in Engineering (GATE) essentially examines the aptitude of an engineering graduate. It differs from other examinations as it calls for an aptitude based learning and approach on engineering topics. To qualify this exam, an aspirant should have perfect understanding on the fundamental concepts, ability to interpret the fundamental quantities and their relationships, and apply these to problem solving.

The book uses a precise and systematic approach to present the subject of computer science and information technology for GATE (CS). The content of the book is built after thorough analysis of concepts asked in previous year question papers, and thus emerges as a fully competent book to cater to the needs of the GATE aspirants. Difficult topics such as theory of computation, compiler design, operating systems are clarified using simple steps and illustrations as well as supported by a large number of problems for practice; as these form the major portion of the GATE syllabus. Using this book, the aspirants would be able to revise their fundamentals of the subject and test their preparedness level through multitude of problems, thereby developing the aptitude required for success in GATE. Further, the book covers the exhaustive syllabus of the subject in such a manner that makes it useful for similar competitive examinations and undergraduate courses in CS and IT streams.

Key Features of the Book

- Content structured to meet the new pattern of GATE and guide the aspirants through various challenges.
- Opening chapter about GATE strategy and methodology and to help students through the preparation phase of the examination.
- Part opener for each unit with graphical representation of number of questions asked in previous years, and topic-wise analysis.
- Emphasis on concept clarity through programming codes, flowcharts, tabular representations, etc.
- Programming and quantitative aspect of the subject covered in form of programming examples, illustrative examples and solved problems to apply the preceding concept.
- Important points and formulas placed at the end of each chapter for a quick review.
- Solutions of previous years' GATE questions at the end of relevant chapters and the latest papers at the end of the book.
- Sufficient solved examples and practice exercises at the end of the chapters divided into 1-mark and 2-mark question sets as per GATE pattern.

About the Authors

Anil Kumar Verma is an Associate Professor in the Department of Computer Science and Engineering, Thapar University, Patiala. Besides having a PhD, he has a rich teaching experience of more than 23 years. He is a member of IEEE, ACM, LMCSI (Mumbai),GMAIMA (New Delhi) and certified software quality auditor by MoCIT, Govt. of India.

Gaurav Sharma is an Assistant Professor in the School of Computing Science and Engineering at Galgotias University, Greater Noida. He received his PhD from Thapar University, Patiala, and is a member of IEEE and ACM.

Kuldeep Singh is a freelance consultant and PhD research scholar at Thapar University, Patiala. He has done his M.E. (Information Security), and has qualified GATE and CSIR-UGC NET.

Wiley India Pvt. Ltd.
4435-36/7, Ansari Road, Daryaganj
New Delhi-110 002
Customer Care +91 11 43630000
Fax +91 11 23275895
csupport@wiley.com
www.wileyindia.com
www.wiley.com

WILEY

follow us on
facebook.com/WileyIndiaTestPrep
twitter.com/wileyindiapl
youtube.com/wileyindiapl
google.com/+wileyindia

ISBN 978-81-265-5086-9