



Northeastern University

Program Structure And Algorithms

Jagadeesh Vasudevamurthy Ph. D

j.vasudevamurthy@northeastern.edu

**Northeastern University
College of Engineering
Department of Information Systems
4 N 2nd St Suite 150,
San Jose, CA 95113**

**Draft 18
Jan 2025**

This material is copyrighted and is intended for exclusive access by registered students.

Any unauthorized copying or posting in any electronic form will result in the maximum legal penalties under USA law.



*sarasvatī namastubhyam
vara de kāmarūpiṇī¹
vidyārambham̄ kariṣyāmi
siddhirbhavatu me sadā*

O Goddess Saraswathi;
salutations to you, the giver of boons,
the one who fulfills desires.
I shall begin my studies.
May there always be
accomplishment for me

This work is dedicated to Goddess Saraswathi

GRADING



Nelson Mandela

Letter	Total Marks
A	97.0 - 100
A-	92.0 - 96.9
B+	85.0 - 91.9
??	<85

The instructor reserves the right to change the above grading without informing students if required.

At the entrance gate of a university in South Africa, the following message was posted for contemplation

"Destroying any nation does not require the use of atomic bombs or the use of long range missiles... It only requires lowering the quality of education and allowing cheating in the examinations by the students..."

Patients die at the hands of such doctors...

Buildings collapse at the hands of such engineers...

Money is lost at the hands of such economists & accountants...

Humanity dies at the hands of such religious scholars...

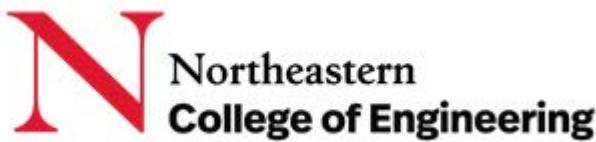
Justice is lost at the hands of such judges...

"The collapse of education is the collapse of the nation."

20% Midterm

20% Final

60% Homeworks



INFO 6205: Program Structures and Algorithms

Course Information

Course Title: **Program Structure and Algorithms**

Course Number: INFO6205

Term and Year: Spring 2025

Credit Hour: 4

CRN

Course Format: On- On-Ground /ZOOM as decided by Northeastern

Instructor Information

Full Name: Jagadeesh Vasudevamurthy Ph.D.

Email Address: j.vasudevamurthy@northeastern.edu

Office Hours: Click or tap here to enter text.

Instructor Biography

<https://www.linkedin.com/in/jagadeesh-vasudevamurthy-6796591/>

Teaching Assistant Information

Course Prerequisites

Graduate Level CSYE 6200 Minimum Grade of B- or Undergraduate Level INFO 5100 Minimum Grade of B- or Graduate Level INFO 5100 Minimum Grade of B-

Course Description

This course covers fundamental programming constructs and their performance. These include lists, stacks, queues, trees, trees, trees, trees, tries, and graphs. The course emphasizes several problems- solving techniques: brute force, recursion, divide-and-conquer, dynamic programming, greedy algorithms, iterative improvement and backtracking. The course also covers both theoretical and experimental measurement of performance, as well as the concept of complexity. The course will also illustrate the various design techniques with problems in graph theory as it applies to social networking paradigms.

Data structures and algorithms are two facets of one fundamental technique of programming. It is impossible to have one without the other. The class will be detail oriented and will provide an essential component for anyone contemplating a career as a software developer. Although the subject could be studied using any language, the language of this class is Python 3.7 and above

Course Learning Outcomes

1. Describe, explain, and use abstract data types including stacks, queues, lists, tree, hash, and graphs.

2. Describe, explain, and implement using varieties of algorithmic techniques like divide and conquer, greedy algorithms, dynamic programming, and back tracking.
3. Describe the asymptotic performance of the algorithms studied in this course and understand the practical implications of that information.
4. Read, criticize, and analyze complexity of Python programs written by some other author.
5. Solve many of the interview problems efficiently on the Leetcode and Hacker Rank interview web sites and confidently attend Google, Amazon, and Facebook interviews. This course is a gold mine for students seeking jobs.

Required Tools and Course Textbooks.

No textbook required. You must install Jupyter notebook to solve the assignment.

Course Schedule/Topics Covered.

For Wednesday classes 01/08/2025 (all dates below -4 days)

Week	Date	In Class Topic	Assignment Due
1	01/12	. Introduction 2. Basic data type 3. Pass by value 4. Swapping two objects in constant time 5. Class and objects 6. Int class 7. Data structure of Int 8. Need for private and public 9. Convert int to Python list 10. Convert Python list to int 11. How to reverse in place Python List 12. Need for operator overloading 1. __str__ 2. __len__ 3. __getitem__ 4. __setitem__ 5. __add__ 6. __sub__ 7. __lt__ 8. __eq__	01/19
2	01/19	1. Introduction 2. OOP 3. Class 4. Objects 5. Test bench 6. Solution 7. Product of Array Except Self 8. How to write testbench as a class 9. O(n^2) time complexity 9. O(n) time complexity 10. O(n) space complexity 11 O(1) space complexity	01/26

		12. How to use Leetcode to evaluate software	
3	01/26	1. Pass by value 2. Implementing python list as a dynamic growable array 3. Table doubling algorithm 4. Amortized cost 5. O(1) 6. Problem in prepending a list 7. Problem in deleting an element from slist 8. Need for a singly linked list 9. append, prepend, find and delete objects from slist 10. a[i] in a slist 11. stack 12. Queue 13. Deque	02/02
4	02/02	1. Need for recursion 2. Factorial using iteration and recursion 3. Printing a digit of a number iteration and recursion 3. Return the reverse value of an integer 4. Need for helper function 5. Merge sort 6. Recurrence tree	02/09
5	02/09	1. Solution to HW 2. Need for hash. 3. Building hash from basic Python 4. Using hash.	02/16
6	02/16	Graph Data Structure. Representation of million node graphs 1 class Graph 2. Build a graph from a file 3 Dump of a graph as a text file 4. Visualizing graphs using Graphviz package	02/23
7	02/23	1. Binary tree 2. Complete binary tree 3. Need for heap 4. Max heap and min heap 5. Representing heap an array 6. Finding left, right and parent in THETA(1) 7. How to build heap in nlogn and O(n) time 8. Heap sort 9. How to use heap from Python Library	NO HW
8	03/02	Midterm	
9	03/09	1. Greedy algorithm 2. Need for dynamic programming 3. Memorization and optimal table building	03/16

		4. Coin change problem 5. How to get answers back 6. 0/1 Knapsack problem	
10	03/16	1. Need for graph 2. Transportation problem 3. Minimal spanning tree 4. Course selection 5. Activity problem 6. Directed and undirected graph 7. Directed acyclic graph (DAG) 8. Graph representation using matrices 9. Graph representation using fan-in and fan-out list	03/23
11	03/23	1. DFS using time stamps BFS. 2. BFS 3. Topological sort 4. Dijkstra Algorithm	03/30
12	03/30	Binary tree 1. Need for left and right pointers 2. Why is parent pointer not required? 3. Tree traversals 4. Preorder, In order, post order, and Level order traversal 5. Level order traversal. 6. Tree visualization using Graphviz.	04/06
13	04/06	Huffman encoding	04/13
14	04/13	1. Binary search tree 2. Implementing Tree hash using BST 3. Need for Trie Data structure. 4. Implementing Trie	04/20
15	04/20	1. Why Disjoint Set? 2. Need for union and find 3. Disjoint set data structure 4. Union by size 5. Path compression 6. Inverse Ackerman function	NO HW
16	04/27	Final	

Assignment Grading

- Attendance – 5%
- 12 Programming assignments -45%
- Midterm Exam – 25%
- Final Exam – 25%

Grading Scale

Letter	Total Marks
A	97.0 - 100
A-	92.0 - 96.9
B+	85.0 - 91.9
??	<85

The instructor reserves the right to change the above grading without informing students if required.

**20% Midterm
20% Final
60% Homeworks**

Attendance/Late Work Policy

Attendance Policy

Students registered in MGEN courses (INFO, CSYE, and DAMG) are allowed **a maximum of 2 absences per course, with 3 or more absences resulting in an automatic 'F' for that course**. Students are expected to inform their instructors of any absences in advance of the class; if a student is sick long-term or experiences a medical issue that prevents class attendance, it is strongly encouraged that they speak with their Academic Advisor (coe-mgen-gradadvising@northeastern.edu) to learn more about the Medical Leave of Absence. Should a student anticipate being unable to attend 3 or more classes, they should discuss their situation with their Academic Advisor to explore other types of leave in accordance with the University's academic and global entry expectations. International students should review the Office of Global Services webpage to understand their visa compliance requirements.

Teaching Assistants (TAs) or Instructional Assistants (IAs) will be present at each class to collect student attendance.

Late Work Policy

Students must submit assignments by the deadline in the time zone noted in the syllabus. Students must communicate with the faculty prior to the deadline if they anticipate work will be submitted late. Work submitted late without prior communication with faculty will not be graded.

End-of-Course Evaluation Surveys

Your feedback regarding your educational experience in this class is particularly important to the College of Engineering. Your comments will make a difference in the future planning and presentation of our curriculum.

At the end of this course, please take the time to complete the evaluation survey at <https://neu.evaluationkit.com>. Your survey responses are **completely anonymous and confidential**. For courses 6 weeks in length or shorter, surveys will be open one week prior to the end of the courses; for courses greater than 6 weeks in length, surveys will be open for two weeks. An email will be sent to your Northeastern University Mail account notifying you when surveys are available.

Academic Integrity

A commitment to the principles of academic integrity is essential to the mission of Northeastern University. The promotion of independent and original scholarship ensures that students derive the most from their educational experience and their pursuit of knowledge. Academic dishonesty violates the most fundamental values of an intellectual community and undermines the achievements of the entire University.

As members of the academic community, students must become familiar with their rights and responsibilities. In each course, they are responsible for knowing the requirements and restrictions regarding research and writing, examinations of whatever kind, collaborative work, the use of study aids, the appropriateness of assistance, and other issues. Students are responsible for learning the conventions of documentation and acknowledgment of sources in their fields. Northeastern University expects students to complete all examinations, tests, papers, creative projects, and assignments of any kind according to the highest ethical standards, as set forth either explicitly or implicitly in this Code or by the direction of instructors.

Go to <http://www.northeastern.edu/osccr/academic-integrity-policy/> to access the full academic integrity policy.

MGEN Student Feedback

Students who would like to provide the MGEN unit with anonymous feedback on this particular course, Teaching Assistants, Instructional Assistants, professors, or to provide general feedback regarding their program, may do so using this survey: https://neu.co1.qualtrics.com/jfe/form/SV_cTIAbH7ZRaaw0Ki

University Health and Counseling Services

As a student enrolled in this course, you are fully responsible for assignments, work, and course materials as outlined in this syllabus and in the classroom. Over the course of the semester if you experience any health issues, please contact UHCS.

For more information, visit <https://www.northeastern.edu/uhs>.

Student Accommodations

Northeastern University and the Disability Resource Center (DRC) are committed to providing disability services that enable students who qualify under Section 504 of the Rehabilitation Act and the Americans with Disabilities Act Amendments Act (ADAAA) to participate fully in the activities of the university. To receive accommodations through the DRC, students must provide appropriate documentation that demonstrates a current substantially limiting disability.

For more information, visit <https://drc.sites.northeastern.edu>.

Library Services

The Northeastern University Library is at the hub of campus intellectual life. Resources include over 900,000 print volumes, 206,500 e-books, and 70,225 electronic journals.

For more information and for education specific resources, visit <https://library.northeastern.edu>
Network Campus Library Services: [Northeastern University Library Global Campus Portals](#)

24/7 Canvas Technical Help

For immediate technical support for Canvas, call 617-373-4357 or email help@northeastern.edu

Canvas Student Resources: <https://canvas.northeastern.edu/student-resources/>

For assistance with my Northeastern e-mail, and basic technical support:

Visit ITS at <https://its.northeastern.edu>

Email: help@northeastern.edu

ITS Customer Service Desk: 617-373-4357

Diversity and Inclusion

Northeastern University is committed to equal opportunity, affirmative action, diversity, and social justice while building a climate of inclusion on and beyond campus. In the classroom, members of the University community work to cultivate an inclusive environment that denounces discrimination through innovation, collaboration, and an awareness of global perspectives on social justice.

Please visit <http://www.northeastern.edu/oidi/> for complete information on Diversity and Inclusion

Title IX

Title IX of the Education Amendments of 1972 protects individuals from sex or gender-based discrimination, including discrimination based on gender-identity, in educational programs and activities that receive federal financial assistance.

Northeastern's Title IX Policy prohibits Prohibited Offenses, which are defined as sexual harassment, sexual assault, relationship or domestic violence, and stalking. The Title IX Policy applies to the entire community, including male, female, transgender students, faculty, and staff.

In case of an emergency, please call 911.

Please visit <https://www.northeastern.edu/ouec> for a complete list of reporting options and resources both on- and off-campus.

Contents

1 Python 3 Primer	25
1.1 Introduction	25
1.2 Installing Anaconda and Jupyter	25
1.3 Installing Visual Studio Code	27
1.4 Basic Python 3	30
1.5 String	59
1.6 List	69
1.7 Tuple	86
1.8 Set	91
1.9 Dictionary	94
1.10 List Comprehension	109
1.11 Heap	121
1.12 Naming convention	124
1.13 <i>class Util</i>	124
2 Analysis of Algorithms	131
2.1 Introduction	131
2.2 Algorithm	131
2.2.1 Finding a number in an unsorted array	133
2.2.2 Finding a fake coin	133
2.3 First look at complexity	135
2.4 Constant time/space algorithms	137

CONTENTS

2.5 Logarithmic complexity algorithms	137
2.5.1 Logarithm	137
2.5.2 Guess a number	140
2.5.3 Some more Logarithmic complexity algorithms examples . .	143
2.5.4 Harmonic series	143
2.6 Linear $O(n)$ algorithms	144
2.7 $O(n \log n)$ algorithms	146
2.8 Quadratic algorithms	147
2.8.1 Quadratic algorithms examples	149
2.9 Generating prime numbers	149
2.9.1 Expected output	153
2.10 Exponential algorithms	154
2.10.1 Geometric series	154
2.10.2 Rice on a chess board	156
2.10.3 Printing a truth table	156
2.10.4 Tower of Hanoi	158
2.11 Execution time for algorithms with given time complexities	160
2.12 Big O Notation	161
2.12.1 $O()$ definition	161
2.12.2 $O()$ Fact 1	163
2.12.3 $O()$ Fact 2	165
2.12.4 $O()$ Fact 3	165
2.12.5 Sum and product rules	167
2.13 Big $\Omega()$ notation	167
2.13.1 $\Omega()$ definition	167
2.14 Big $\Theta()$ notation	169
2.14.1 $\Theta()$ definition	169
2.15 Problem set	174
3 Recursion	185

CONTENTS

3.1	Introduction	185
3.2	Factorial of a number	185
3.2.1	Time and space complexity	187
3.3	$\sum_{i=0}^n i$	187
3.4	Computing number of binary digits to represent a decimal number	189
3.4.1	Time and space complexity	189
3.5	Reverse a number	191
3.6	Reverse a <i>slist</i>	191
3.7	Fibonacci numbers	192
3.8	Tower of Hanoi	195
3.9	Permutation	198
3.10	Recursion trees	201
3.11	Master Theorem	209
3.11.1	Solution	213
3.12	Problem set	217
4	Python List Implementation	221
4.1	Introduction	221
4.2	<i>List</i> Class	221
4.2.1	Expected Output	228
5	Singly Linked List	231
5.1	Introduction	231
5.2	<i>Slist</i> Class	231
5.3	Problem set	232
6	Doubly Linked List	237
6.1	Introduction	237
6.2	<i>dlist</i> Class	237
7	Stack Queue and Deque	239

CONTENTS

7.1	Introduction	239
7.2	<i>Stack</i> Class Using Python List	239
7.3	<i>MinStack</i> Class Using Stack	241
7.4	<i>Queue</i> Class Using Python List	241
7.5	<i>Stack</i> Class Using <i>Queue</i>	243
7.6	<i>Queue</i> Class Using <i>Stack</i>	245
7.7	<i>Deque</i> Class	247
7.8	Problem set	249
8	Searching and sorting	257
8.1	Introduction	257
8.2	Selection sort	257
8.2.1	Array based selection sort	257
8.2.2	Linked list based selection sort	259
8.3	Insertion sort	259
8.3.1	Array based insertion sort	259
8.3.2	Linked list based insertion sort	261
8.4	Binary search	261
8.4.1	Array based binary search	261
8.4.2	Linked list based binary search	263
8.5	Insertion binary sort	263
8.5.1	Array based insertion binary sort	263
8.5.2	Linked list based insertion binary sort	266
8.6	Merge sort	266
8.6.1	Array based merge sort	266
8.6.2	Linked list based merge sort	273
8.7	Quick sort	273
8.7.1	Array based quick sort	273
8.7.2	Linked list based quick sort	280
8.8	Quick select	280

CONTENTS

8.9 Counting Sort	282
8.10 Straight radix sort: from LSB to MSB	282
8.11 Radix exchange sort: from MSB to LSB	286
8.12 Problem set	286
9 Hash	287
9.1 Introduction	287
9.2 What is a hash?	287
9.3 <i>hash</i> Class	292
9.3.1 Expected output	302
9.4 Problem set	303
10 Heap	307
10.1 Introduction	307
10.2 What is a heap?	307
10.3 Representation of heap as an array	309
10.4 Finding maximum element of a heap	309
10.5 Inserting an element to the heap	311
10.6 Deleting maximum element from the heap	311
10.7 Writing class Heap	313
10.8 Building a heap of n elements from top to bottom	316
10.9 Building a heap of n elements from bottom to top	318
10.10 Heap sort algorithm	321
10.11 Heap in Python 3	321
10.12 Problem set	324
11 Graphviz - Graph Visualization Software	329
11.1 Introduction	329
11.2 Graphviz package	329
11.3 Example 1	331

CONTENTS

12 Binary Tree	333
12.1 Introduction	333
12.2 Various binary trees	333
12.3 Building heap data structure from user data	335
12.4 Building binary tree from user data	337
12.4.1 TreeNode.py	337
12.4.2 BuildTree.py	339
12.5 Converting a binary tree data structure to a dot file	339
12.5.1 WriteDot.py	339
12.6 Examples	341
12.6.1 Example 1	341
12.6.2 Example 2	341
12.6.3 Example 3	342
12.6.4 Example 4	344
12.7 Displaying a binary tree	344
12.7.1 Dot2Pdf.py	346
12.8 Tree traversal	349
12.8.1 Example 1	349
12.8.2 Example 2	349
12.8.3 Example 3	351
12.9 Algorithms on Tree	351
12.9.1 Postorder traversal: Number of nodes in a binary tree . . .	351
12.9.2 Height and depth of a tree	352
12.9.3 Postorder traversal: Depth or Height of a binary tree . . .	354
12.9.4 Height of a node in a tree	356
12.9.5 Depth of a node in a tree	356
12.9.6 Postorder tree traversal: Deleting a tree	358
12.9.7 Preorder traversal: Printing all paths of a tree	358
12.9.8 Inorder traversal	360

CONTENTS

12.10 Statistics	360
12.11 Huffman coding	360
12.12 Implementing Huffman coding	363
12.12.1 Leetcode 535. Encode and Decode TinyURL	367
13 Binary Search Tree	369
13.1 Introduction	369
13.2 Definition of BST	369
13.3 Is BST?	371
13.4 Building BST	371
13.5 Searching an element in BST	372
13.6 Minimum and maximum of a node	373
13.7 Successor of a node	375
13.8 Predecessor of a node	375
13.9 Deleting an element from BST	377
13.9.1 Case 1: Deleting a leaf from BST	377
13.9.2 Case 2: Deleting a node that has one kid from BST	377
13.9.3 Case 3: Deleting a node that has two kids from BST	379
13.9.4 Deletion in action	380
14 Trie Data structure	383
14.1 Introduction	383
14.2 Need for Trie data structure	383
14.3 Printing trie	385
14.3.1 Printing all paths in a binary tree	385
14.3.2 Printing trie	385
14.4 Finding all prefixes of a word	387
14.5 Printing numbers in lexicographic order	389
15 Disjoint sets	391
15.1 Introduction	391

CONTENTS

15.2 Need for Union and Find Data structure	391
15.3 Smart Union Algorithm	393
15.4 Problem set	397
16 Greedy Algorithms	403
16.1 Introduction	403
17 Dynamic Programming	405
17.1 Introduction	405
17.2 Why greedy algorithm may fail to give solution?	405
17.3 Fibonacci numbers	407
17.3.1 Fibonacci numbers using divide and conquer	407
17.3.2 Fibonacci numbers using dynamic programming	407
17.4 Coin change problem	408
17.5 Maximum subset independent set of a path graph	412
17.5.1 Expected output	412
17.6 0/1 Knapsack problem	422
17.7 Single source shortest path algorithm using dynamic programming	425
18 Graphs	433
18.1 Introduction	433
18.2 Graph applications	433
18.2.1 Transportation problem	433
18.2.2 Minimum connector problem	435
18.2.3 Scheduling problem	435
18.2.4 Activity network or Topological sorting problem	437
18.2.5 Critical path analysis	440
18.2.6 Flow problem	442
18.3 Graph examples	443
18.4 Graph representation using matrices	446
18.4.1 Undirected graph representation	446

CONTENTS

18.4.2 Undirected weighted graph representation	446
18.4.3 Directed graph representation	448
18.4.4 Directed weighted graph representation	448
18.5 Graph representation using fanins and fanouts lists	450
18.6 graphviz package	452
18.7 User Data	462
18.8 Graph Data structure	462
18.8.1 class GraphInterface	462
18.8.2 class Node	464
18.8.3 class Edge	464
18.8.4 class Graph	465
18.9 Dump a graph as a text file	467
18.10 Build a graph from a file	467
18.11 Write a graph as a dot file	469
18.11.1 Various dot file examples	469
18.12 Loops in a graph	471
18.13 Depth first search using time stamps	473
18.13.1 Depth first search on a undirected graph with no loop	477
18.13.2 Depth first search on a undirected graph with loop	478
18.13.3 Depth first search on a directed graph with no loop	480
18.13.4 Depth first search on a directed graph with loop	482
18.13.5 Depth first search on a directed graph with no loop	484
18.13.6 Depth first search on a directed graph with no loop	486
18.13.7 Implementing DFS using time stamps	488
18.14 Depth first search using colors	488
18.15 Breadth first search	491
18.15.1 Implementing BFS	496
18.16 Application of Breadth first search	498
18.16.1 Snake and Ladder	498

CONTENTS

18.17 Single source shortest path algorithms on a non negative weighted directed graph	498
18.17.1 Dijkstra's algorithm	498
18.17.2 Concept of relaxation	503
18.17.3 Dijkstra's algorithm in action	505
18.17.4 Implementing Dijkstra's algorithm	509
18.18 Topological sort	512
18.18.1 Topological sort on a directed acyclic graph	512
18.18.2 Topological sort on a directed graph	514
18.18.3 Topological sort algorithm	516
18.18.4 Topological sort examples	518
19 Algorithms Patterns	521
19.1 Introduction	521
19.2 Sliding Windows Patterns	521
19.2.1 Sliding window Maximum Average	521
19.2.2 Sliding window maximum	521
20 Algorithms for Interviews	523
20.1 Introduction	523
20.2 LeetCode Problem 38: Count and say	523
20.2.1 Expected output	524
20.2.2 Leetcode output	525
20.3 LeetCode Problem 66: Plus one	527
20.3.1 Expected output	527
20.3.2 Leetcode output	529
20.4 LeetCode Problem 2: Add Two Numbers	530
20.4.1 Leetcode output	530
20.5 LeetCode Problem 977: Square of a sorted array	532
20.5.1 Expected output	532
20.5.2 Leetcode output	534

CONTENTS

20.6 LeetCode Problem 233: Product of Array Except Self	536
20.6.1 Expected output	536
20.6.2 Leetcode output	539
20.7 Find duplicate elements in a Python list	539
20.7.1 Expected output	541
20.8 LeetCode Problem 88: Merge Sorted Array	545
20.8.1 Expected output	546
20.8.2 Leetcode output	549
20.9 LeetCode Problem 238. Product of Array Except Self	551
20.9.1 Expected output	551
20.9.2 Leetcode output	554
20.10 Finding a fake coin	554
20.11 LeetCode Problem 34: Find First and Last Position of Element in Sorted Array	557
20.11.1 Expected output	557
20.11.2 Leetcode output	560
20.12 LeetCode Problem 204: Count Primes	562
20.12.1 Leetcode output	564
20.13 LeetCode Problems 225, 235, 632 and 641 using <i>slist</i>	564
20.14 Friends	569
20.15 Hop	571
20.16 LeetCode Problem 162. Find Peak Element	573
20.16.1 Expected output	573
20.16.2 Leetcode output	576
20.17 LeetCode Problem 46: Permutations	578
20.17.1 Solution Idea	578
20.17.2 Expected output	580
20.17.3 Leetcode output	583
20.18 LeetCode Problem 909: Snake and Ladder. HackerRank: The Quick- est Way Up	585

CONTENTS

20.18.1 LeetCode Problem 909: Snakes and Ladders	585
20.18.2 Coordinate system	585
20.18.3 Example 1	587
20.18.4 Example 2	587
20.18.5 Example 3	589
20.18.6 HackerRank: The Quickest Way Up	591
20.18.7 Expected output	597
20.19 Buying and Selling stocks	597
20.19.1 $\Theta(n^2)$ time and $\Theta(1)$ space algorithm	598
20.19.2 $\Theta(n \log_2 n)$ time and $\Theta(\log_2 n)$ space algorithm	599
20.19.3 $\Theta(n)$ time and $\Theta(\log_2 n)$ space algorithm	601
20.19.4 $\Theta(n)$ time and $\Theta(1)$ space algorithm	603
20.19.5 Expected output	603
20.19.6 Leetcode	605
20.20 LeetCode Problem 322: Coin Change	607
20.20.1 Expected output	607
20.20.2 Leetcode output	627
20.21 House Robber 198	629
20.22 Destination city	631
20.23 Is Graph Bipartite?	633
20.23.1 Reading and writing the graph	633
20.23.2 Graph bipartite solution	636
20.24 Clone an undirected graph	640
20.25 Course schedule	643
20.26 Shortest path from a given source	647
20.27 Grow or Shrink	649
20.28 Selection Day Puzzle	649
20.29 Merging Communities	651
20.30 LeetCode 643. Maximum Average Subarray I	652

CONTENTS

20.30.1 Expected output	652
20.30.2 Leetcode output	652
20.31 LeetCode Problem 239. Sliding Window Maximum	654
20.31.1 Expected output	654
20.31.2 Leetcode output	655
20.31.3 Geeks for Geeks output	657

CONTENTS

VASUDEVAMURTHY

Chapter 1

Python 3 Primer

1.1 Introduction

1.2 Installing Anaconda and Jupyter

-----INSTALLING FRESH -----

If you something, uninstall

1. Go to add or remove program and uninstall
Anaconda3 2019.10(Python 3.7.4 64-bit)

Google

latest anconda for windows 10

ANACONDA WILL GET PYTHON ALSO.

For Windows

Python 3.9

64-Bit Graphical Installer 510 MB

NOW INSTALL FRESH. YOU GET PYTHON ALSO

64 bit Graphical installer (462MB)

JUST ME (OTHERWISE CREATES ADMIN ISSUES)

DEFAULT IS OFF. SELECT it so that path will be added

SELECT (Right mark) Add anaconda to my Path environment variable

SELECT BOTH IN GUI

Will install in this directory

C:\Users\jag\anaconda3

----- TO SEE ALL ENV OF ipython and jupyter-----
in cmdtool type

conda env export -n base

----- Which Version-----
IN cmdtool

type: python

Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc.
on win32

----- Which anaconda -----
conda list anaconda

C:\Users\jag>conda list anaconda
packages in environment at C:\Users\jag\anaconda3:

Name Version Build Channel
anaconda 2021.11 py39_0
anaconda-client 1.9.0 py39haa95532_0
anaconda-navigator 2.1.1 py39_0
anaconda-project 0.10.1 pyhd3eb1b0_0

----- UPDATING CONDA. DO THIS EVERY MONTH-----

cd C:\Users\jag\Anaconda3

conda update --all

C:\Users\jag\Anaconda3>conda update --all
Collecting package metadata (current_repodata.json): done
Solving environment: done

1.3 Installing Visual Studio Code

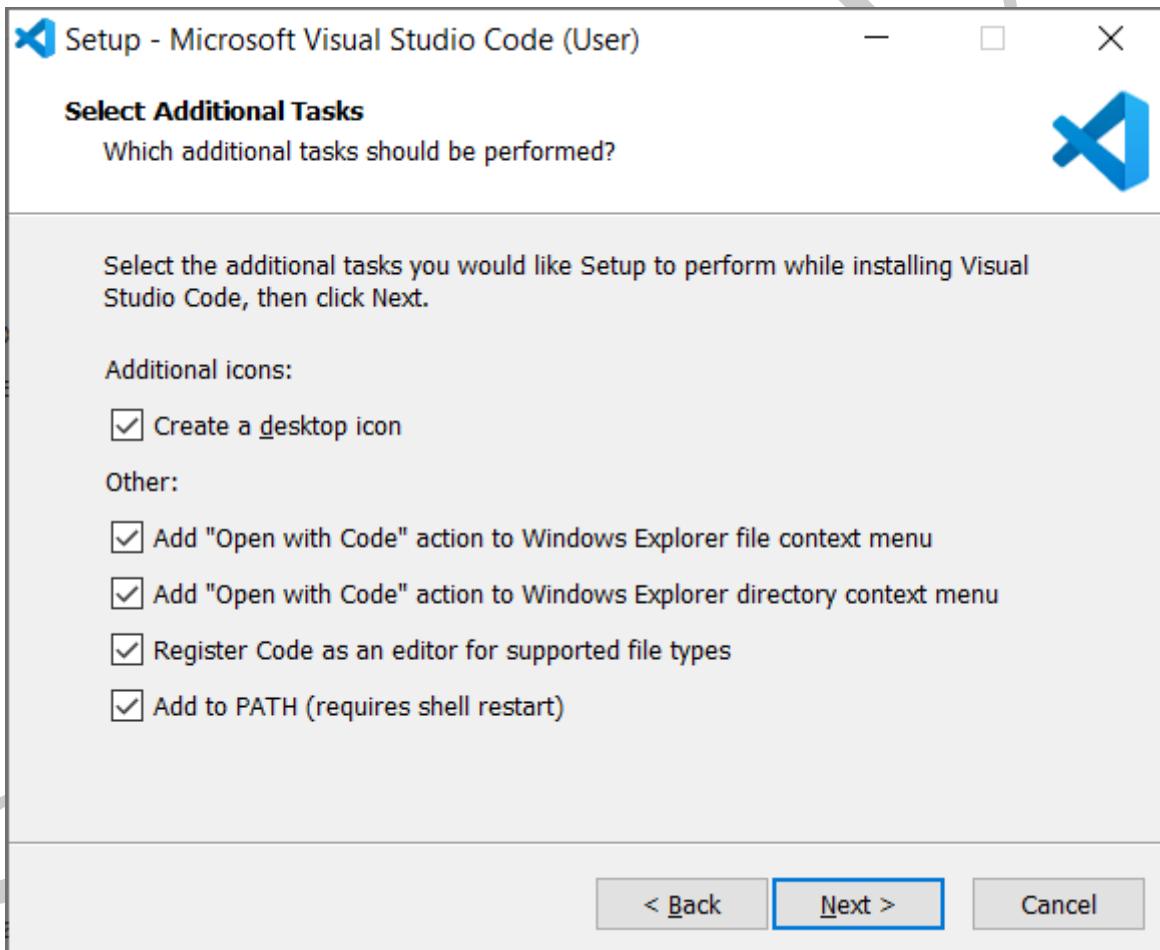
Build and Debug using Visual Studio Code

Google visual code

First find

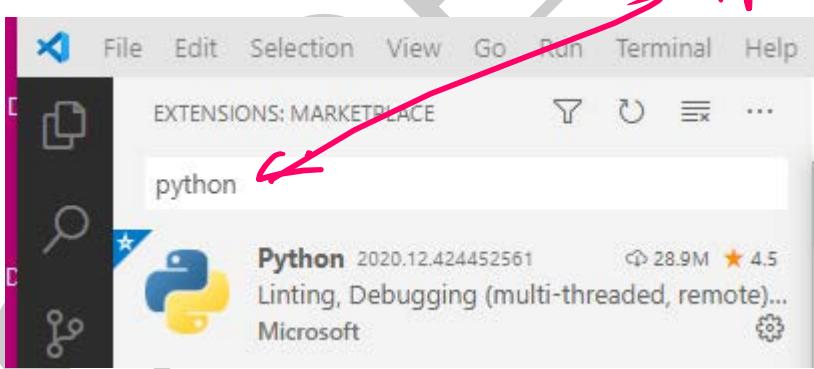
Download

<https://code.visualstudio.com/download>



Setting Up VSCode For Python Programming

https://www.youtube.com/watch?v=W--_EOzdTHk



Extensions Click on it

Type Python

Select this

1.4 Basic Python 3



python Cheat Sheet



Math	Power & Logarithmic	Angular Conversion	Constants
Number Theoretic	exp(x) log(x, base) log1p(x) log10(x) ceil(x) copySIGN(x,y) fabs(x) factorial(x) floor(x) fmod(x,y) frexp(x) fsum(iterable) isinf(x) isnan(x) ldexp(x,i) modf(x) trunc()	degrees(x) radians(x) Hyperbolic Functions	math.pi The mathematical constant pi = 3.141592... up to the available precision. math.e The mathematical constant e = 2.718281... up to the available precision.
	Trigonometric Functions	acosh(x) asinh(x) atanh(x) cosh(x) sinh(x) tanh(x)	
	acos(x) asin(x) atan(x) atan2(y,x) cos(x) hypot(x,y)		
	sin(x) tan(x)		

String Formatting

Formatting Operations

'd' Signed integer decimal 'l' Signed integer decimal 'o' Signed octal value 'v' Obsolete type - it was identical to 'd'
 'x' Signed hexadecimal (lowercase) 'X' Signed hexadecimal (uppercase) 'e' Floating point exponential format (lowercase)
 'E' Floating point exponential format (uppercase) 'f' Floating point decimal format 'F' Floating point decimal format
 'g' Floating point format. Uses the lowercase exponential format if the exponent is less than -4 or not less than precision, otherwise it uses the decimal format
 'G' Floating point format. Uses the uppercase exponential format if the exponent is less than -4 or not less than precision, otherwise it uses the decimal format
 'c' Single character (accepts either integer or single character string) 's' String (converts any Python object using repr())
 'r' String (converts any Python object using str()) '%' No argument is converted, adds a % character in the end result

File

Methods

close() flush() fileno()
isatty() next() read(size)
readline([size]) readlines([size])
xreadlines() seek(offset[, whence])
tell() truncate([size])
write(str) writelines(sequence)

Attributes

closed encoding
errors mode
name newlines
softspace

Class

Special Methods

__new__(cls) __lt__(self, other) __init__(self, args)
__le__(self, other) __del__(self) __gt__(self, other)
__repr__(self) __ge__(self, other) __str__(self)
__eq__(self, other) __cmp__(self, other)
__ne__(self, other) __index__(self) __nonzero__(self)
__hash__(self) __getatt__(self, name)
__getattribute__(self, name) __setattr__(self, name, value)
__delattr__(self, name) __call__(self, args, kwargs)

Random

Functions

seed(x[, getstate()]) vonmisesvariate(mu,kappa)
setstate(state) jumpahead(n) paretovariate(alpha)
getrandbits(k) randint(a,b) weibullvariate(alpha,beta)
randrange([start], stop, step) lognormvariate(mu,sigma)
choice(seq) shuffle(x[, random]) normalvariate(mu, sigma)
sample(population,k) random() gammavariate(alpha,beta)
uniform(a,b) triangular(low,high,mode) gauss(mu,sigma)
betaeviate(alpha,beta) expovariate(lambd)

Array

Array Methods

append(x) buffer_info() a[0,1,2,3,4,5]
byteswap() count() a[1] [1,2,3,4,5]
extend(iterable) fromfile(f,[n]) a[0,1,2,3] len(a) 6
fromlist(l[, offset]) fromstring(s) a[1:3] [1,2] a[0] 0
fromunicode(index) a[1:-1] [1,2,3,4] a[3] 5
insert(i[, pop]) remove(x) a[-1] 5
reverse() tofile(f[, offset]) a[-2] 4
tostring() tounicode()

OS

OS Variables

altsep Alternative separator
curdir Current dir string
depth Default search path
devnull Path of null device
extsep Extension separator

pardir Parent dir string
pathsep Path separator
sep Path separator
name Name of OS
linesep Line separator

platform

Current platform
stdin, stdout, stderr File objects for I/O
version_info Python version info
winver Version number

String

String Methods

capitalize() center(width[, fillchar]) count(sub[, start[, end]])
decode(encoding[, errors][, locale]) encode(encoding[, errors][, locale])
endswith(suffix[, start[, end]]) expandtabs(tabsize)
find(sub[, start[, end]]) format(*args, **kwargs) ljust(width[, start[, end[, fillchar]]])
index(sub[, start[, end]]) isdigit() islower() isupper() rfind(sub[, start[, end]])
isprintable() join(text) partition(sep[, sep[, sep]]) replace(old, new[, count])
ljust(width[, start[, end]]) rindex(sub[, start[, end]])
rjust(width[, fillchar]) rpartition(sep[, sep[, sep]]) rsplit(sep[, maxsplit])
rstrip(chars) split(sep[, sep[, sep]]) strip(chars[, chars[, chars]])
startswith(prefix[, start[, end]]) strip(chars[, chars[, chars]])
translate(table[, start[, end]]) upper() zfill(width)
zfill(width[, start[, end]])

SYS Arg V

args Command-line args
builtin_module_names Linked C modules
check_interval Signal check frequency
exec_prefix Root directory
executable Name of Executable
exitfunc Exit function name
modules Loaded modules
path Search path

SYS Arg V

sys.argv[0] foo.py

sys.argv[1] bar

sys.argv[2] -c

sys.argv[3] quux

sys.argv[4] -h

Date Time

Date Object

replace(year[, month, day][, hour][, minute][, second][, microsecond])
timetuple() weekday() isoweekday() isocalendar()
isoformat() strftime(format[, tzinfo]) strptime(%Y-%m-%d %H:%M:%S %Z)[, tzinfo])
astimezone(tzinfo) asctime() strptime(%Y-%m-%d %H:%M:%S %Z)[, tzinfo])
isoformat() strptime(%Y-%m-%d %H:%M:%S %Z)[, tzinfo])

Set & Mapping

Mapping Types

len(d) d[key] d[key]=value
del(d[key]) key in d key not in d
dict clear() copy() items()
fromkeys(keys[, value]) keys()
get(key[, default]) has_key(key)
has_value(key)
iteritems() iterkeys()
itervalues() pop(key[, default])
update(other)
update(other)
values()

Time Object

replace(hour[, minute[, second[, microsecond[, tzinfo]]]])

strftime(format[, tzinfo]) strptime(%Y-%m-%d %H:%M:%S %Z)[, tzinfo])

Datetime Object

date() time() timetuple() weekday() isoweekday() isocalendar()
replace(year[, month[, day[, hour[, minute[, second[, microsecond[, tzinfo]]]]]])
astimezone(tzinfo) asctime() strptime(%Y-%m-%d %H:%M:%S %Z)[, tzinfo])
isoformat() strptime(%Y-%m-%d %H:%M:%S %Z)[, tzinfo])

Date Formatting

%A Abbreviated weekday (Mon) %a Weekday (Monday)
 %B Abbreviated month name (Nov) %b Month name (November)
 %C Date and time %d Day (leading zero) (01 to 31)
 %H 24-hour (leading zero) (00 to 23) %I 12-hour (leading zero) (01 to 12)
 %j Day of year (001 to 366) %m Month (01 to 12) %M Minute (00 to 59)
 %p AM or PM %S Second (00 to 61) %U Week number (00 to 53)
 %w Weekday (2 to 6) %W Week number (00 to 53) %x Date
 %z Time zone (%H:%M) %Y Year without century (00 to 99) %y Year (2010)
 %Z Time zone (%Z) %c A stamp (%c character (%))

```
1 # Basic Python3
2 # Copyright: Jagadeesh Vasudevamurthy
3 # filename: basic.ipynb
```

```
1 # Basic imports
```

```
In [1]: 1 import sys
2 import os
3 print(sys.version)
```

```
3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
```

Printing

```
In [2]: 1 print('Hello World using single quote')
```

```
Hello World using single quote
```

```
In [3]: 1 print("Hello World using double quote")
```

```
Hello World using double quote
```

```
In [4]: 1 print('''Hello World using triple quote''')
```

```
Hello World using triple quote
```

```
In [5]: 1 print('"Hello World within double quote using single quote"')
```

```
"Hello World within double quote using single quote"
```

```
In [6]: 1 print(''''Hello World within double quote using triple quote'''')
```

```
"Hello World within double quote using triple quote"
```

```
In [7]: 1 print(3 + -5)
```

```
-2
```

In [8]:

```

1 x = 5
2 y = 8
3 print(x,y)
4 print("Python gives a space after x and EOLN after y")
5 print("x =", x, "y =", y)
6 print("Python default separator (sep =) is a space")
7 print("Python default eoln (end =) is a EOLN")

```

```

5 8
Python gives a space after x and EOLN after y
x = 5 y = 8
Python default separator (sep =) is a space
Python default eoln (end =) is a EOLN

```

In [9]:

```

1 x = 5
2 y = 8
3 print("User can change both sep= and end =")
4 print('' I am changing sep = space by sep =" -> " ''')
5 print('' I am changing end = EOLN by end =" ] " ''')
6 print("x", x, sep = '->', end = ' ')
7 print("realend")

```

```

User can change both sep= and end =
I am changing sep = space by sep =" -> "
I am changing end = EOLN by end =" ] "
x->5]realend

```

In [10]:

```

1 print("To show how to print without and with line feed")
2 x = 5
3 y = 8
4 print(x,y,end="")
5 x = 5
6 y = 8
7 print(x,y)

```

```

To show how to print without and with line feed
5 85 8

```

In [11]:

```

1 x = "UC"
2 y = "Santa Clara"
3 print("I am studying at", x, ".Campus is at",y)
4 print("----FIX ---")
5 print("I am studying at", x,end=". ")
6 print("Campus is at", y)

```

```

I am studying at UC .Campus is at Santa Clara
----FIX ---
I am studying at UC. Campus is at Santa Clara

```

Basic Data types: int, float, bool, string

Concept of name (LHS), value (RHS)

types(Inferred by python)

Type of name (LHS) need not have to announced.

All names on RHS must be available. Constants are already available(5, 3.14)

Python will infer type of the name automatically

As a result, the variables has no fixed type like C or Java

```
In [12]: 1 k = 200
          2 print("type of k", type(k), "Value of k =", k, "id =", id(k))

type of k <class 'int'> Value of k = 200 id = 2301260358288
```

```
In [13]: 1 print("In C or java you can say int junk; What is the value of junk?")
          2 print("Python will not allow this.")
          3 #junk
          4 print("you cannot declare name without assignment like int junk in C")

In C or java you can say int junk; What is the value of junk?
Python will not allow this.
you cannot declare name without assignment like int junk in C
```

```
In [14]: 1 k = 200
          2 print("type of k", type(k), "Value of k =", k, "id =", id(k))
          3 print("note 1 below is constant and available")
          4 k = k + 1 #++k
          5 print("type of k", type(k), "Value of k =", k, "id =", id(k))

type of k <class 'int'> Value of k = 200 id = 2301260358288
note 1 below is constant and available
type of k <class 'int'> Value of k = 201 id = 2301260358320
```

```
In [15]: 1 k = 78.90
          2 print("type of k", type(k), "Value of k =", k, "id =", id(k))
          3 k = 2
          4 print("type of k", type(k), "Value of k =", k, "id =", id(k))

type of k <class 'float'> Value of k = 78.9 id = 2301313589392
type of k <class 'int'> Value of k = 2 id = 2301260163408
```

```
In [16]: 1 k = 678906789.878
          2 print("type of k", type(k), "Value of k =", k, "id =", id(k))

type of k <class 'float'> Value of k = 678906789.878 id = 2301313590864
```

```
In [18]: 1 k = True
          2 print("type of k", type(k), "Value of k =", k, "id = ", id(k))
```

```
type of k <class 'bool'> Value of k = True id = 140703779215464
```

```
In [19]: 1 k = False
          2 print("type of k", type(k), "Value of k =", k, "id =", id(k))
```

type of k <class 'bool'> Value of k = False id = 140703779215496

```
In [20]: 1 k = 'a'  
2 print("type of k", type(k), "Value of k =", k, "id =", id(k))
```

```
type of k <class 'str'> Value of k = a id = 2301262827632
```

```
In [21]: 1 k = "a"
2 print("type of k", type(k), "Value of k =", k, "id = ", id(k))
```

```
type of k <class 'str'> Value of k = a id = 2301262827632
```

```
In [22]: 1 k = '''a'''  
2 print("type of k", type(k), "Value of k =", k, "id = ", id(k))
```

type of k <class 'str'> Value of k = a id = 2301262827632

```
In [23]: 1 k = "pack my box with five dozen Liquor jugs"
2 print("type of k", type(k), "Value of k =", k, "id =", id(k))
```

```
type of k <class 'str'> Value of k = pack my box with five dozen Liquor jugs  
id = 2301313645136
```

bool

```
In [24]: 1 k = True
          2 print("type of k", type(k), "Value of k =", k)
```

type of k <class 'bool'> Value of k = True

```
In [25]: 1 k = False  
2 print("type of k", type(k), "Value of k =", k)
```

type of k <class 'bool'> Value of k = False

In [26]:

```

1 print("To show how to build AND")
2 k = True
3 print("type of k", type(k), "Value of k =", k)
4 j = False
5 print("type of j", type(j), "Value of j =", j)
6 m = k and j
7 print("type of m", type(m), "Value of m =", m)

```

To show how to build AND

type of k <class 'bool'> Value of k = True
 type of j <class 'bool'> Value of j = False
 type of m <class 'bool'> Value of m = False

In [27]:

```

1 print("To show how to build OR")
2 k = True
3 print("type of k", type(k), "Value of k =", k)
4 j = False
5 print("type of j", type(j), "Value of j =", j)
6 m = k or j
7 print("type of m", type(m), "Value of m =", m)

```

To show how to build OR

type of k <class 'bool'> Value of k = True
 type of j <class 'bool'> Value of j = False
 type of m <class 'bool'> Value of m = True

In [28]:

```

1 print("To show how to build XOR")
2 k = True
3 print("type of k", type(k), "Value of k =", k)
4 j = False
5 print("type of j", type(j), "Value of j =", j)
6 m = (k and (not j)) or (not(k) and j) #XOR
7 print("type of m", type(m), "Value of m =", m)

```

To show how to build XOR

type of k <class 'bool'> Value of k = True
 type of j <class 'bool'> Value of j = False
 type of m <class 'bool'> Value of m = True

In [29]:

```

1 print("AND truth table")
2 for k in [False,True]:
3     for j in [False,True]:
4         m = (k and j)
5         print(k,id(k),j,id(j),m,id(m))

```

AND truth table

False 140703779215496 False 140703779215496 False 140703779215496
 False 140703779215496 True 140703779215464 False 140703779215496
 True 140703779215464 False 140703779215496 False 140703779215496
 True 140703779215464 True 140703779215464 True 140703779215464

```
In [30]: 1 print("OR truth table")
2 for k in [False,True]:
3     for j in [False,True]:
4         m = (k or j)
5         print(k,j,m)
```

OR truth table
 False False False
 False True True
 True False True
 True True True

```
In [31]: 1 print("XOR truth table")
2 for k in [False,True]:
3     for j in [False,True]:
4         m = (k and (not j)) or (not(k) and j) #XOR
5         print(k,j,m)
```

XOR truth table
 False False False
 False True True
 True False True
 True True False

```
In [32]: 1 print("XNOR truth table")
2 for k in [False,True]:
3     for j in [False,True]:
4         m = not((k and (not j)) or (not(k) and j)) #XNOR
5         print(k,j,m)
```

XNOR truth table
 False False True
 False True False
 True False False
 True True True

Immutable VS Mutable basic types

```
In [33]: 1 x = 10
2 y = x #y is pointing to x
3 print("BEFORE: x = ",x, " y = ",y, type(x), type(y), id(x), id(y))
4 print("AT THIS TIME x and y are not separate objects")
5 x = 5 #x will be changed to 5
6 print("AFTER changing x = 5: x = ",x, " y = ",y, type(x), type(y), id(x), id(y))
7 print("Now x and y are separate objects")
```

BEFORE: x = 10 y = 10 <class 'int'> <class 'int'> 2301260163664 2301260163
 664
 AT THIS TIME x and y are not separate objects
 AFTER changing x = 5: x = 5 y = 10 <class 'int'> <class 'int'> 23012601635
 04 2301260163664
 Now x and y are separate objects 37

This is because int is NOT 4 bytes like C

```
In [34]: 1 x = 10456781678916789678111111111116666677777777777777744444444444444
2 y = x #y is pointing to x
3 print("BEFORE: x = ",x, " y = ",y, type(x), type(y), id(x), id(y))
4 print("AT THIS TIME x and y are not separate objects")
5 x = 5 #x will be changed to 5
6 print("AFTER changing x = 5: x = ",x, " y = ",y, type(x), type(y), id(x), id(y))
7 print("Now x and y are separate objects")
```

BEFORE: x = 10456781678916789678111111111666667777777777777777444444
44444444444444444444 y = 1045678167891678967811111111116666677777777777777
77777744444444444444444444444444 <class 'int'> <class 'int'> 2301313900784 23
01313900784
AT THIS TIME x and y are not separate objects
AFTER changing x = 5: x = 5 y = 10456781678916789678111111111666667777
777777777777777444444444444444444444 <class 'int'> <class 'int'> 2301260
163504 2301313900784
Now x and y are separate objects

/ and //operator

```
In [35]: 1 i = 7
2 print(i/2)
```

3.5

```
In [36]: 1 i = 7
2 print(i//2)
```

3

```
In [37]: 1 print("How to find the midpoint")
2 l = 0
3 r = 7
4 m = (l + r)/2
5 print("l =",l , "r =",r, " m =",m)
```

How to find the midpoint
l = 0 r = 7 m = 3.5

```
In [38]: 1 print("How to find the midpoint")
2 l = 0
3 r = 7
4 m = (l + r)//2
5 print("l =",l , "r =",r, " m =",m)
```

How to find the midpoint
l = 0 r = 7 m = 3

```
In [39]: 1 print("How to find the midpoint")
2 l = 0
3 r = 7
4 m = l + (r - 1)//2
5 print("l =",l , "r =",r, " m =",m)
```

How to find the midpoint
l = 0 r = 7 m = 3

Type conversion

Conversion function in C++

```
In [40]: 1 k = 7889
2 print("type of k", type(k), "Value of k =", k)
3 s = str(k)
4 print("type of s", type(s), "Value of s =", s)
5 print("We have converted int 7889 to string '7889'")
6 k = int(s)
7 print("type of k", type(k), "Value of k =", k)
8 print("Now we have converted string '7889' to int 7889")
9 print("k = int('45z') #This gives an error, because this cannot be converted")
```

type of k <class 'int'> Value of k = 7889
 type of s <class 'str'> Value of s = 7889
 We have converted int 7889 to string '7889'
 type of k <class 'int'> Value of k = 7889
 Now we have converted string '7889' to int 7889
 k = int('45z') #This gives an error, because this cannot be converted to a valid integer

Control statement: if

```
In [41]: 1 x = 90
2 if (x > 90):
3     print("your mark is = ", x, ".You are fantastic")
4     print("your mark is = ", x, ".You are amazing")
5 print("You are great")
```

You are great

```
In [42]: 1 x = 91
2 if (x > 90):
3     print("your mark =", x, end=". ")
4     print("You are amazing")
5 print("You are great")
```

your mark = 91. You are amazing
 You are great

Control statement: if else

In [43]:

```
1 print("In C if (x > 20) { }")
2 print ("In Python we use : instead of {")
3 print("In Python indentation is forced by language")
4 print("all code below : must have the same tab or space. Dont mix tab and")
5 print()
6 x = 900
7 if (x > 90):
8     print("1")
9     print("2")
10    print("3")
11    print("4")
12 else:
13     print("5")
14     print("6")
15     print(7)
16 print("junk")
```

In C if (x > 20) { }
In Python we use : instead of {
In Python indentation is forced by language
all code below : must have the same tab or space. Dont mix tab and space

```
1
2
3
4
junk
```

In [44]:

```

1 print("In C if (x > 20) { }")
2 print ("In Python we use : instead of {")
3 print("In Python indentation is forced by language")
4 print("all code below : must have the same tab or space. Dont mix tab and")
5 print()
6 x = 9
7 if (x > 90):
8     print("1")
9     print("2")
10    print("3")
11    print("4")
12 else:
13     print("5")
14     print("6")
15     print(7)
16 print("junk")

```

In C if (x > 20) { }
 In Python we use : instead of {
 In Python indentation is forced by language
 all code below : must have the same tab or space. Dont mix tab and space

```

5
6
7
junk

```

Control statement: if elif

In [45]:

```

1 x = 5
2 if (x > 90):
3     print("1")
4     print("2")
5     print("3")
6     print("4")
7 elif (x > 85):
8     print("5")
9     print("6")
10    print('7')
11 elif (x > 80):
12     print("8")
13     print("9")
14     print('10')
15 else:
16     print(-420)
17 print("ahah")

```

-420
 ahah

In [46]:

```
1 x = 86
2 if (x > 90):
3     print("1")
4     print("2")
5     print("3")
6     print("4")
7 elif (x > 85):
8     print("5")
9     print("6")
10    print('7')
11 elif (x > 80):
12     print("8")
13     print("9")
14     print('10')
15 else:
16     print(-420)
17 print("ahah")
```

```
5
6
7
ahah
```

In [47]:

```
1 x = 81
2 if (x > 90):
3     print("1")
4     print("2")
5     print("3")
6     print("4")
7 elif (x > 85):
8     print("5")
9     print("6")
10    print('7')
11 elif (x > 80):
12     print("8")
13     print("9")
14     print('10')
15 else:
16     print(-420)
17 print("ahah")
```

```
8
9
10
ahah
```

Find the maximum of three numbers

In [48]:

```

1 def findmax(x:'int', y:'int', z:'int')->'int':
2     max = 0
3     print("x =",x, "y =", y , "z =", z)
4     if (x > y): #x is bigger
5         if (x > z): # x is bigger
6             max = x
7         else:
8             max = z
9     else:
10        if (y > z): # y is bigger
11            max = y
12        else:
13            max = z
14    return max
15
16 m = findmax(55,100,200)
17 print("max m =",m)
18 m = findmax(100,100,100)
19 print("max m =",m)
20 m = findmax(100,200,55)
21 print("max m =",m)
22 m = findmax(100.89,200.90,55.0)
23 print("max m =",m)
24 #m = findmax("cat",26,"Lion")
25 #print("max m =",m)

```

```

x = 55 y = 100 z = 200
max m = 200
x = 100 y = 100 z = 100
max m = 100
x = 100 y = 200 z = 55
max m = 200
x = 100.89 y = 200.9 z = 55.0
max m = 200.9

```

range (Start, Stop, Step)

range() takes three arguments.

Out of the three 2 arguments are optional. i.e., Start and Step are the optional arguments. That means Stop IS REQUIRED. Stops at Stop-1

So by default, it takes start = 0 and step = 1.

A start argument is a starting number of the sequence. i.e., lower limit. By default, it starts with 0 if not specified.

A stop argument is an upper limit. i.e. generate numbers up to this number, The range() function doesn't include this number in the

result.

The step is a difference between each number in the result. The

In [49]:

```

1 print('''ONLY ONE argument. That is STOP''')
2 for i in range(10):
3     print(i, " ", end='')
4 print()

```

ONLY ONE argument. That is STOP
0 1 2 3 4 5 6 7 8 9

In [50]:

```

1 print('''TWO argument. That is START and STOP''')
2 for i in range(3,10):
3     print(i, " ", end='')
4 print()

```

TWO argument. That is START and STOP
3 4 5 6 7 8 9

In [51]:

```

1 print('''Three argument. That is START STOP STEP''')
2 for i in range(5,17,2):
3     print(i, " ", end='')
4 print()

```

Three argument. That is START STOP STEP
5 7 9 11 13 15

In [52]:

```

1 print("range(17,5,-2) generates sequence starting 17, ending at less than")
2 for i in range(17,5,-2):
3     print(i, " ", end='')
4 print()

```

range(17,5,-2) generates sequence starting 17, ending at less than 5, at a distance of -2
17 15 13 11 9 7

In [53]:

```

1 print("Example 1: range(17,5,2) generates empty sequence")
2 for i in range(17,5,2):
3     print(i, " ", end='')

```

Example 1: range(17,5,2) generates empty sequence

In [54]:

```

1 print("Example 2: range(5,17,-2) generates empty sequence")
2 for i in range(5,17,-2):
3     print(i, " ", end='')

```

Example 2: range(5,17,-2) generates empty sequence

In [55]:

```

1 print("Example 3: range(5,17,16) generates sequence of one element ",end='')
2 for i in range(5,17,16):
3     print(i)

```

Example 3: range(5,17,16) generates sequence of one element 5

In [56]:

```

1 print("Example 4: range(12,1,-3) stops at 3, because next one is 0 which is smaller than 1")
2 for i in range(12,1,-3):
3     print(i, " ", end='')
4 print()

```

Example 4: range(12,1,-3) stops at 3, because next one is 0 which is smaller than 1
12 9 6 3

Is Range and list are equal?

In [57]:

```

1 for i in [0,1,2,3,4,5]:
2     print(i, " ", end=' ')
3 print()
4 for i in range(0,5):
5     print(i, " ", end=' ')
6 print()
7 print("Is list == range ?. In python 3, NO but it is true in Python 2")
8 print(range(0,6) == [0,1,2,3,4,5])
9 print("Can we convert range to a list?")
10 print("Yes. This is also called as type conversion")
11 #L = list(range(1, 1001)). This is not working
12 l = []
13 l.extend(range(0,5))
14 print("type of l", type(l), "Value of l =", l)
15 l = l + [420]
16 for i in l:
17     print(i, " ", end=' ')
18 print()

```

0 1 2 3 4 5
0 1 2 3 4
Is list == range ?. In python 3, NO but it is true in Python 2
False
Can we convert range to a list?
Yes. This is also called as type conversion
type of l <class 'list'> Value of l = [0, 1, 2, 3, 4]
0 1 2 3 4 420

For and while

In [58]:

```

1 print("Using for")
2 for i in range(0,5):
3     print(i,end = ' ')
4 print()

```

Using for
0 1 2 3 4

```
In [59]: 1 print("Using while")
2 i = 0 ;
3 while (i < 5):
4     print(i,end = ' ')
5     i = i + 1
6 print()
```

Using while
0 1 2 3 4

```
In [60]: 1 print("Using for in list")
2 list = [78,"cat", True, 7.89]
3 for i in list:
4     print(i,end = ' ')
5 print()
6 print("Using while in list")
7 i = 0 ;
8 while (i < len(list)):
9     print(list[i],end = ' ')
10    i = i + 1
11 print()
```

Using for in list
78 cat True 7.89
Using while in list
78 cat True 7.89

When to use for

```
In [61]: 1 print("Use for when you know the range")
2 def factors(n):
3     num = 1
4     print("Factors of ",n, " are: 1 ", end='')
5     m = n//2 + 1 #for 100, you need to test until 50 only
6     for i in range(2,m):
7         if (n % i == 0):
8             num = num + 1
9             print(i, " ", end=' ')
10    print()
11    print("number of factors of ", n, " = ", num)
12
13 factors(1000)
14 factors(19867989)
```

Use for when you know the range
Factors of 1000 are: 1 2 4 5 8 10 20 25 40 50 100 125 200 250
500
number of factors of 1000 = 15
Factors of 19867989 are: 1 3 6622663
number of factors of 19867989 = 3

When to use while

In [62]:

```

1 print("Use for when you know the range")
2 print("Use While when you don't know the range")
3 print("For example to find integer value of a square root")
4 def squareroot(n:int)->'int':
5     i = 2 ;
6     while (i * i <= n):
7         i = i + 1
8     return i-1 ;
9
10 n = 100
11 print("Integer square root of( ", n, " ) = ", squareroot(n))
12 n = 29
13 print("Integer square root of( ", n, " ) = ", squareroot(n))
14 n = 1000
15 print("Integer square root of( ", n, " ) = ", squareroot(n))

```

Use for when you know the range
 Use While when you don't know the range
 For example to find integer value of a square root
 Integer square root of(100) = 10
 Integer square root of(29) = 5
 Integer square root of(1000) = 31

Breaking out of while loop

In [63]:

```

1 p = 10
2 i = 0
3 broke = False #to remember whether if broke or not
4
5 while (i < 10):
6     if (i == p):
7         broke = True
8         print("I am breaking at ", i)
9         break
10    i=i+1
11 if (broke == False):
12     print("I did not break", i)

```

I did not break 10

In [64]:

```

1 p = 5
2 i = 0
3 broke = False #to remember whether if broke or not
4
5 while (i < 10):
6     if (i == p):
7         broke = True
8         print("I am breaking at ", i)
9         break
10    i=i+1
11 if (broke == False):
12     print("I did not break", i)

```

I am breaking at 5

Breaking out of while loop - Python way

In [65]:

```

1 p = 10
2 for i in range(0,10):
3     if (i == p):
4         print("I am breaking at ", i)
5         break ;
6 else:
7     print("This line is executed only when the break never happened. Nothing")
8     print("I did not break")

```

This line is executed only when the break never happened. Nothing has to be remembered

I did not break

In [66]:

```

1 p = 5
2 for i in range(0,10):
3     if (i == p):
4         print("I am breaking at ", i)
5         break
6 else:
7     print("This line is executed only when the break never happened. Nothing")
8     print("I did not break", i)

```

I am breaking at 5

What is the output of this program?

In [67]:

```

1 def xxx(z:'int')->'none':
2     print("z inside xxx ", z)
3     print("type of z", type(z), "Value of z =", z,"id of z = ", id(z))
4     z = 100
5     print("type of z", type(z), "Value of z =", z,"id of z = ", id(z))
6     print("z inside xxx after assigning 100 =", z)
7
8
9 z = 10
10 print("type of z", type(z), "Value of z =", z,"id of z = ", id(z))
11 print("z before xxx ", z)
12 xxx(z)
13 print("type of z", type(z), "Value of z =", z,"id of z = ", id(z))
14 print("z after xxx ", z)

```

```

type of z <class 'int'> Value of z = 10 id of z = 2301260163664
z before xxx 10
z inside xxx 10
type of z <class 'int'> Value of z = 10 id of z = 2301260163664
type of z <class 'int'> Value of z = 100 id of z = 2301260355024
z inside xxx after assigning 100 = 100
type of z <class 'int'> Value of z = 10 id of z = 2301260163664
z after xxx 10

```

HOW TO YOU FIX?

In [68]:

```

1 def xxx(a:'array of size 1')->'none':
2     print("a[0] inside xxx ", a[0])
3     a[0] = 100
4     print("a[0] inside xxx after assigning 100 =", a[0])
5
6
7 z = 10
8 print("z before xxx ", z)
9 a = [z]
10 xxx(a)
11 print("z after xxx ", z)
12 z = a[0]
13 print("z after z = a[0] ", z)

```

```

z before xxx 10
a[0] inside xxx 10
a[0] inside xxx after assigning 100 = 100
z after xxx 10
z after z = a[0] 100

```

HOW TO YOU FIX? Dissected

In [69]:

```

1 def xxx(a:'array of size 1')->'none':
2     print("inside xxx :           type of a", type(a), "Value of a")
3     a[0] = 100
4     print("inside xxx after a[0] = 100: type of a", type(a), "Value of a")
5
6
7 z = 10
8 print("type of z", type(z), "Value of z =", z, "id of z =", id(z))
9 a = [z]
10 print("type of a", type(a), "Value of a =", a, "id of a[0] =", id(a[0]))
11 xxx(a)
12 print("z after xxx: type of z", type(z), "Value of z =", z, "id of z =", id(z))
13 z = a[0]
14 print("z = a[0]:   type of z", type(z), "Value of z =", z, "id of z =", id(z))

```

```

type of z <class 'int'> Value of z = 10 id of z = 2301260163664
type of a <class 'list'> Value of a = [10] id of a[0] = 2301260163664
inside xxx :           type of a <class 'list'> Value of a = [10] id o
f a[0] = 2301260163664
inside xxx after a[0] = 100: type of a <class 'list'> Value of a = [100] id
of a[0] = 2301260355024
z after xxx: type of z <class 'int'> Value of z = 10 id of z = 230126016366
4
z = a[0]:   type of z <class 'int'> Value of z = 100 id of z = 230126035502
4

```

Why this is required?

In [70]:

```

1 class Alg:
2     def __init__(self, work):
3         self._work = work
4         self._alg()
5         #return self._work    ##TypeError: __init__() should return None, n
6
7     def _alg(self):
8         self._work = 100
9
10 work = 0
11 a = Alg(work)
12 print("work =", work)

```

How do you fix?

In [71]:

```

1 class Alg:
2     def __init__(self, work:'list of size 1'):
3         self._work = work
4         self._alg()
5
6     def _alg(self):
7         self._work[0] = 100
8
9 work = [0]
10 a = Alg(work)
11 print("work =", work[0])

```

work = 100

Will this Swap?

In [72]:

```

1 def swap(x:int,y:int)->'none':
2     print("In Swap x = ", x, " y = ", y)
3     t = x
4     x = y
5     y = t ;
6     print("Out Swap x = ", x, " y = ", y)
7
8 x = 100
9 y = 200
10 print("Before Swap x = ", x, " y = ", y)
11 swap(x,y)
12 print("After Swap x = ", x, " y = ", y)

```

Before Swap x = 100 y = 200
 In Swap x = 100 y = 200
 Out Swap x = 200 y = 100
 After Swap x = 100 y = 200

Will this Swap? Dissected

In [73]:

```

1 show = True
2 def swap(x:int,y:int)->'none':
3     print("In Swap x = ", x, " y = ", y)
4     if (show):
5         print("type of x", type(x), "Value of x =", x,"id of x = ", id(x))
6         print("type of y", type(y), "Value of y =", y,"id of y = ", id(y))
7     t = x
8     if (show):
9         print("after t = x")
10    print("type of x", type(x), "Value of x =", x,"id of x = ", id(x))
11    print("type of t", type(t), "Value of t =", t,"id of t = ", id(t))
12 x = y
13 if (show):
14     print("after x = y")
15     print("type of y", type(y), "Value of y =", y,"id of y = ", id(y))
16     print("type of x", type(x), "Value of x =", x,"id of x = ", id(x))
17 y = t ;
18 if (show):
19     print("after y = t")
20     print("type of t", type(t), "Value of t =", t,"id of t = ", id(t))
21     print("type of y", type(y), "Value of y =", x,"id of y = ", id(x))
22 print("Out Swap x = ", x, " y = ", y)
23
24 x = 100
25 y = 200
26 if (show):
27     print("type of x", type(x), "Value of x =", x,"id of x = ", id(x))
28     print("type of y", type(y), "Value of y =", y,"id of y = ", id(y))
29 print("Before Swap x = ", x, " y = ", y)
30 swap(x,y)
31 print("After Swap x = ", x, " y = ", y)
32 if (show):
33     print("type of x", type(x), "Value of x =", x,"id of x = ", id(x))
34     print("type of y", type(y), "Value of y =", y,"id of y = ", id(y))

```

```

type of x <class 'int'> Value of x = 100 id of x = 2301260355024
type of y <class 'int'> Value of y = 200 id of y = 2301260358288
Before Swap x = 100 y = 200
In Swap x = 100 y = 200
type of x <class 'int'> Value of x = 100 id of x = 2301260355024
type of y <class 'int'> Value of y = 200 id of y = 2301260358288
after t = x
type of x <class 'int'> Value of x = 100 id of x = 2301260355024
type of t <class 'int'> Value of t = 100 id of t = 2301260355024
after x = y
type of y <class 'int'> Value of y = 200 id of y = 2301260358288
type of x <class 'int'> Value of x = 200 id of x = 2301260358288
after y = t
type of t <class 'int'> Value of t = 100 id of t = 2301260355024
type of y <class 'int'> Value of y = 200 id of y = 2301260358288
Out Swap x = 200 y = 100
After Swap x = 100 y = 200
type of x <class 'int'> Value of x = 100 id of x = 2301260355024
type of y <class 'int'> Value of y = 200 id of y = 2301260358288

```

Function that swap that swap by passing list

In [74]:

```
1 def swap(a:'list of size 2')->'None':
2     print("In Swap a[0]= ", a[0], " a[1] = ", a[1])
3     t = a[0]
4     a[0] = a[1]
5     a[1] = t
6     print("out Swap a[0]= ", a[0], " a[1] = ", a[1])
7
8
9 x = 100
10 y = 200
11 print("Before Swap x = ", x, " y = ", y)
12 a = [x,y]
13 swap(a)
14 x = a[0]
15 y = a[1]
16 print("After Swap x = ", x, " y = ", y)
```

```
Before Swap x = 100 y = 200
In Swap a[0]= 100 a[1] = 200
out Swap a[0]= 200 a[1] = 100
After Swap x = 200 y = 100
```

Function that swap that swap by passing list dissected

In [75]:

```
1 def swap(a:'list of size 2')->'None':  
2     print("-----ENTER SWAP-----")  
3     print("type of a", type(a), "Value of a =", a,"id of a = ", id(a))  
4     print("type of a[0]", type(a[0]), "Value of a[0] =", a[0],"id of a = "  
5         print("type of a[1]", type(a[1]), "Value of a[1] =", a[1],"id of a = "  
6             t = a[0]  
7             print("type of t", type(t), "Value of t =", t,"id of t = ", id(t))  
8             a[0] = a[1]  
9             a[1] = t  
10            print("type of a", type(a), "Value of a =", a,"id of a = ", id(a))  
11            print("type of a[0]", type(a[0]), "Value of a[0] =", a[0],"id of a = "  
12            print("type of a[1]", type(a[1]), "Value of a[1] =", a[1],"id of a = "  
13            print("-----LEAVE SWAP -----")  
14  
15  
16 x = 100  
17 y = 200  
18 print("type of x", type(x), "Value of x =", x,"id of x = ", id(x))  
19 print("type of y", type(y), "Value of y =", y,"id of y = ", id(y))  
20 print("Before Swap x = ", x, " y = ", y)  
21 a = [x,y]  
22 print("type of a", type(a), "Value of a =", a,"id of a = ", id(a))  
23 print("type of a[0]", type(a[0]), "Value of a[0] =", a[0],"id of a = ", id  
24 print("type of a[1]", type(a[1]), "Value of a[1] =", a[1],"id of a = ", id  
25 swap(a)  
26 print("type of a", type(a), "Value of a =", a,"id of a = ", id(a))  
27 print("type of a[0]", type(a[0]), "Value of a[0] =", a[0],"id of a = ", id  
28 print("type of a[1]", type(a[1]), "Value of a[1] =", a[1],"id of a = ", id  
29 print("-----x and y BEFORE interchange -----")  
30 print("type of x", type(x), "Value of x =", x,"id of x = ", id(x))  
31 print("type of y", type(y), "Value of y =", y,"id of y = ", id(y))  
32 print("-----x and y AFTER interchange -----")  
33 x = a[0]  
34 y = a[1]  
35 print("type of x", type(x), "Value of x =", x,"id of x = ", id(x))  
36 print("type of y", type(y), "Value of y =", y,"id of y = ", id(y))  
37 print("After Swap x = ", x, " y = ", y)
```

```

type of x <class 'int'> Value of x = 100 id of x = 2301260355024
type of y <class 'int'> Value of y = 200 id of y = 2301260358288
Before Swap x = 100 y = 200
type of a <class 'list'> Value of a = [100, 200] id of a = 2301313901504
type of a[0] <class 'int'> Value of a[0] = 100 id of a = 2301260355024
type of a[1] <class 'int'> Value of a[1] = 200 id of a = 2301260358288
-----ENTER SWAP-----
type of a <class 'list'> Value of a = [100, 200] id of a = 2301313901504
type of a[0] <class 'int'> Value of a[0] = 100 id of a = 2301260355024
type of a[1] <class 'int'> Value of a[1] = 200 id of a = 2301260358288
type of t <class 'int'> Value of t = 100 id of t = 2301260355024
type of a <class 'list'> Value of a = [200, 100] id of a = 2301313901504
type of a[0] <class 'int'> Value of a[0] = 200 id of a = 2301260358288
type of a[1] <class 'int'> Value of a[1] = 100 id of a = 2301260355024
-----LEAVE SWAP -----
type of a <class 'list'> Value of a = [200, 100] id of a = 2301313901504
type of a[0] <class 'int'> Value of a[0] = 200 id of a = 2301260358288
type of a[1] <class 'int'> Value of a[1] = 100 id of a = 2301260355024
-----x and y BEFORE interchange -----
type of x <class 'int'> Value of x = 100 id of x = 2301260355024
type of y <class 'int'> Value of y = 200 id of y = 2301260358288
-----x and y AFTER interchange -----
type of x <class 'int'> Value of x = 200 id of x = 2301260358288
type of y <class 'int'> Value of y = 100 id of y = 2301260355024
After Swap x = 200 y = 100

```

Function that swap that by returning list

In [76]:

```

1 def swap(x:int,y:int)->'list':
2     print("In Swap x = ", x, " y = ", y)
3     a = [y,x]
4     return a
5
6 x = 100
7 y = 200
8 print("Before Swap x = ", x, " y = ", y)
9 a = swap(x,y)
10 print("after calling swap ", "x = ", x, " y = ", y)
11 x = a[0]
12 y = a[1]
13 print("After Swap x = ", x, " y = ", y)

```

```

Before Swap x = 100 y = 200
In Swap x = 100 y = 200
after calling swap x = 100 y = 200
After Swap x = 200 y = 100

```

A list is given.

The function f computes some other numbers in another list t

Contents of t will be written to l

In [86]:

```

1 def f(l:'list')->'None':
2     print("In function f:",l)
3     print("type of l", type(l), "Value of l =", l,"id = ", id(l))
4     t = [4,5,6]
5     print("t = ",t)
6     print("type of t", type(t), "Value of t =", t,"id = ", id(t))
7     l = t.copy()
8     print("type of t", type(t), "Value of t =", t,"id = ", id(t))
9     print("type of l", type(l), "Value of l =", l,"id = ", id(l))
10    print("Getting out function f:",l)
11
12    l = [1,2,3]
13    print("Before calling f:",l)
14    print("type of l", type(l), "Value of l =", l,"id = ", id(l))
15    f(l)
16    print("After calling f:",l)
17    print("type of l", type(l), "Value of l =", l,"id = ", id(l))

```

Before calling f: [1, 2, 3]
type of l <class 'list'> Value of l = [1, 2, 3] id = 2301312901440
In function f: [1, 2, 3]
type of l <class 'list'> Value of l = [1, 2, 3] id = 2301312901440
t = [4, 5, 6]
type of t <class 'list'> Value of t = [4, 5, 6] id = 2301314230144
type of t <class 'list'> Value of t = [4, 5, 6] id = 2301314230144
type of l <class 'list'> Value of l = [4, 5, 6] id = 2301313406208
Getting out function f: [4, 5, 6]
After calling f: [1, 2, 3]
type of l <class 'list'> Value of l = [1, 2, 3] id = 2301312901440

How do you Fix?

In [78]:

```

1 def f(l:'list')->'None':
2     print("In function f:",l)
3     t = [4,5,6]
4     print("t = ",t)
5     i = 0
6     for e in t:
7         l[i] = e
8         i = i + 1
9     print("Getting out function f:",l)
10
11 l = [1,2,3]
12 print("Before calling f:",l)
13 f(l)
14 print("After calling f:",l)

```

Before calling f: [1, 2, 3]
In function f: [1, 2, 3]
t = [4, 5, 6]
Getting out function f: [4, 5, 6]
After calling f: [4, 5, 6]

rotate

In [79]:

```

1 n = 3
2 a = '01001101'
3 k = 8
4 for i in range(n-1,0,-1):
5     print(a[i])

```

0
1

Bit operation

In [80]:

```

1 k = 0
2 n = 80
3 k |= (1 << n)
4 print("type of k", type(k), "Value of k =", k,"id = ", id(k))
5

```

type of k <class 'int'> Value of k = 1208925819614629174706176 id = 23013139
83472

C++ Test1

C++ TEST2

In []: 1

1.5 String

Strings

Copyright: Jagadeesh Vasudevamurthy 

filename: string.ipynb

Basic imports

In [1]:

```
1 import sys
2 import os
3 print(sys.version)
```

3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]

No char type

In [2]:

```
1 print("No char type. Single character is also string")
2 s = 'h'
3 print("type of s", type(s), "Value of s =", s)
```

No char type. Single character is also string
type of s <class 'str'> Value of s = h

In [3]:

```
1 s = 'jag'
2 print("type of s", type(s), "Value of s =", s)
```

type of s <class 'str'> Value of s = jag

In [4]:

```
1 s = "jag"
2 print("type of s", type(s), "Value of s =", s)
```

type of s <class 'str'> Value of s = jag

In [5]:

```
1 s = '''jag'''
2 print("type of s", type(s), "Value of s =", s)
```

type of s <class 'str'> Value of s = jag

In [6]:

```
1 s = "\jag"
2 print("type of s", type(s), "Value of s =", s)
```

type of s <class 'str'> Value of s = \jag

```
In [7]: 1 s = """Lovers's"""
2 print("type of s", type(s), "Value of s =", s)
```

type of s <class 'str'> Value of s = "Lovers's"

Empty String. Strings are immutable

```
In [8]: 1 s = None
2 if (s == None):
3     print("I am none")
4     print("type of s", type(s), "Value of s =", s, "id(s) =", id(s))
5 s = ""
6 print("type of s", type(s), "Value of s =", s, "id(s) =", id(s))
7 if not s:
8     print("I am empty string")
9 s = s + str(6)
10 print("type of s", type(s), "Value of s =", s, "id(s) =", id(s))
11 s = s + "jag"
12 print("type of s", type(s), "Value of s =", s, "id(s) =", id(s))
13 if not s:
14     print("empty string")
15 else:
16     print("I am Not empty string")
```

I am none
type of s <class 'NoneType'> Value of s = None id(s) = 140733301529816
type of s <class 'str'> Value of s = id(s) = 1803039688304
I am empty string
type of s <class 'str'> Value of s = 6 id(s) = 1803116858672
type of s <class 'str'> Value of s = 6jag id(s) = 1803116855664
I am Not empty string

Extracting elements of a string

In [9]:

```
1 s = "hello"
2 print(s)
3 print("length of s", len(s))
4 print("type of s", type(s), "Value of s =", s, "id(s) =", id(s))
5 print("s[0] = ", s[0])
6 print("s[1] = ", s[1])
7 print("s[2] = ", s[2])
8 print("s[3] = ", s[3])
9 print("s[4] = ", s[4])
10 print("Reverse accessing")
11 print("s[-1] = ", s[-1])
12 print("s[-2] = ", s[-2])
13 print("s[-3] = ", s[-3])
14 print("s[-4] = ", s[-4])
15 print("s[-5] = ", s[-5])
16 print("first element of the string is accessed using s[0]", s[0])
17 print("last element of the string is accessed using s[-1]", s[-1])
18 print("We don't have to know the length of the string")
```

```
hello
length of s 5
type of s <class 'str'> Value of s = hello id(s) = 1803113040624
s[0] = h
s[1] = e
s[2] = l
s[3] = l
s[4] = o
Reverse accessing
s[-1] = o
s[-2] = l
s[-3] = l
s[-4] = e
s[-5] = h
first element of the string is accessed using s[0] h
last element of the string is accessed using s[-1] o
We don't have to know the length of the string
```

Extracting substrings

In [10]:

```

1 s = "hello"
2 print("type of s", type(s), "Value of s =", s, "id(s) =", id(s))
3 print("length of s", len(s))
4 y = s[1:4]
5 print("type of y", type(y), "Value of y =", s, "id(y) =", id(y))
6 print("s[1] to s[3] = s[1:4] = ", y)
7 print("s[i:j] starts at i and ends at j-1")
8 print("Get first two characters of a string = s[0:2]", s[0:2])
9 print("Get first two characters of a string = s[:2]", s[:2])
10 print("Get last two characters of a string = s[3:len(s)]", s[3:len(s)])
11 print("Get last two characters of a string = s[3:]", s[3:])
12 print("s[3:1] is wrong. Returns empty string", s[3:1])
13 print("s[0:7] is wrong. Returns string up to correct function", s[0:7])
14 s = "23"
15 i = int(s)
16 print(i)

```

```

type of s <class 'str'> Value of s = hello id(s) = 1803113040624
length of s 5
type of y <class 'str'> Value of y = hello id(y) = 1803116727152
s[1] to s[3] = s[1:4] = ell
s[i:j] starts at i and ends at j-1
Get first two characters of a string = s[0:2] he
Get first two characters of a string = s[:2] he
Get last two characters of a string = s[3:len(s)] lo
Get last two characters of a string = s[3:] lo
s[3:1] is wrong. Returns empty string
s[0:7] is wrong. Returns string up to correct function hello
23

```

String are immutable

In [11]:

```

1 s = "hello"
2 print("type of s", type(s), "Value of s =", s, "id(s) =", id(s))
3 #s[3] = 'k' ## Strings cannot be modified. Immutable
4 print("Strings cannot be modified")
5 m = s[0:3] + 'k' + s[4:len(s)]
6 print("type of m", type(m), "Value of m =", m, "id(m) =", id(m))
7 s = s[0:3] + 'k' + s[4:len(s)]
8 print("type of s", type(s), "Value of s =", s, "id(s) =", id(s))
9 print("Replacing string hello. But we lost s = hello to:", s)

```

```

type of s <class 'str'> Value of s = hello id(s) = 1803113040624
Strings cannot be modified
type of m <class 'str'> Value of m = helko id(m) = 1803116853872
type of s <class 'str'> Value of s = helko id(s) = 1803116851632
Replacing string hello. But we lost s = hello to: helko

```

Assignment does not copy Strings

In [12]:

```

1 s = "hello"
2 print("type of s", type(s), "Value of s =", s, "id(s) =", id(s))
3 s1 = s ;
4 print("type of s1", type(s1), "Value of s1 =", s1, "id(s1) =", id(s1))
5 print("s and s1 are same string, See id above. Both are pointing to the same")

```

```

type of s <class 'str'> Value of s = hello id(s) = 1803113040624
type of s1 <class 'str'> Value of s1 = hello id(s1) = 1803113040624
s and s1 are same string, See id above. Both are pointing to the same string

```

Problem with Strings

In [13]:

```

1 s = "hello"
2 print("Note Python has no char type")
3 print("type of s", type(s), "Value of s =", s, "id(s) =", id(s))
4 s1 = ""
5 for c in s:
6     print("type of c", type(c), "Value of c =", c, "id(c) =", id(c))
7     s1 = s1 + c
8     print("type of s1", type(s1), "Value of s1 =", s1, "id(s1) =", id(s1))
9 print("type of s1", type(s1), "Value of s1 =", s1, "id(s1) =", id(s1))
10 print("s = ", s, "s1 = ", s1,"Both are pointing to different address. See id")
11 id(s), id(s1)

```

```

Note Python has no char type
type of s <class 'str'> Value of s = hello id(s) = 1803113040624
type of c <class 'str'> Value of c = h id(c) = 1803041971248
type of s1 <class 'str'> Value of s1 = h id(s1) = 1803041971248
type of c <class 'str'> Value of c = e id(c) = 1803041866096
type of s1 <class 'str'> Value of s1 = he id(s1) = 1803116869552
type of c <class 'str'> Value of c = l id(c) = 1803041902448
type of s1 <class 'str'> Value of s1 = hel id(s1) = 1803116870768
type of c <class 'str'> Value of c = l id(c) = 1803041902448
type of s1 <class 'str'> Value of s1 = hell id(s1) = 1803116880304
type of c <class 'str'> Value of c = o id(c) = 1803040209200
type of s1 <class 'str'> Value of s1 = hello id(s1) = 1803116880752
type of s1 <class 'str'> Value of s1 = hello id(s1) = 1803116880752
s = hello s1 = hello Both are pointing to different address. See id

```

Out[13]: (1803113040624, 1803116880752)

Fixing with List

In [14]:

```

1 s = "hello"
2 print("type of s", type(s), "Value of s =", s, "id(s) =", id(s))
3 l = []
4 for c in s:
5     l.append(c)
6     print("type of l", type(l), "Value of l =", l, "id(l) =", id(l))
7 s2 = ""
8 s1 = s2.join(l)
9 print("type of s1", type(s1), "Value of s1 =", s1, "id(s1) =", id(s1))
10 print("s = ", s, "s1 = ", s1,"Both are pointing to different address. See id")
11 id(s), id(s1)

```

```

type of s <class 'str'> Value of s = hello id(s) = 1803113040624
type of l <class 'list'> Value of l = ['h'] id(l) = 1803116924608
type of l <class 'list'> Value of l = ['h', 'e'] id(l) = 1803116924608
type of l <class 'list'> Value of l = ['h', 'e', 'l'] id(l) = 1803116924608
type of l <class 'list'> Value of l = ['h', 'e', 'l', 'l'] id(l) = 1803116924608
8
type of l <class 'list'> Value of l = ['h', 'e', 'l', 'l', 'o'] id(l) = 1803116924608
type of s1 <class 'str'> Value of s1 = hello id(s1) = 1803116871472
s = hello s1 = hello Both are pointing to different address. See id

```

Out[14]: (1803113040624, 1803116871472)

What is the output of this program?

In [15]:

```

1 def f(s:'string'):
2     s = s + "abc"
3
4 s = "jag"
5 print("s BEFORE calling function f", s)
6 f(s)
7 print("s AFTER calling function f", s)
8

```

```

s BEFORE calling function f jag
s AFTER calling function f jag

```

pass by value through list

In [16]:

```

1 def f(s:'List of string size 1'):
2     print("s in function f BEFORE ", s[0])
3     s[0] = s[0] + "abc"
4     print("s in function f AFTER", s[0])
5
6 s = []
7 s.append("")
8 f(s)
9 print("s AFTER calling function f", s[0])
10 print(s[0])
11
12 s = []
13 s.append("jag")
14 f(s)
15 print("s AFTER calling function f", s[0])
16 print(s[0])

```

```

s in function f BEFORE
s in function f AFTER abc
s AFTER calling function f abc
abc
s in function f BEFORE jag
s in function f AFTER jagabc
s AFTER calling function f jagabc
jagabc

```

Operation on Strings

In [17]:

```

1 s = "hello"
2 s.upper()
3 print("String are immutable. You cannot change s after created")
4 print(s)

```

```

String are immutable. You cannot change s after created
hello

```

In [18]:

```

1 s = "hello"
2 s.upper()
3 print(s)
4 u = s.upper()
5 print(u)
6 id(s), id(u)

```

```

hello
HELLO

```

Out[18]: (1803113040624, 1803116851504)

In [19]:

```

1 s = "hello"
2 print(s)
3 u = s.upper()
4 print(u)
5 t = s + u # addition
6 print(t)
7 id(s), id(u), id(t)

```

hello
HELLO
helloHELLO

Out[19]: (1803113040624, 1803116858544, 1803116851504)

In [20]:

```

1 s = "hello"
2 print(s)
3 u = t*5 # concatenation
4 print(u)
5 id(s), id(u)

```

hello
helloHELLOhelloHELLOhelloHELLOhelloHELLOhelloHELLO

Out[20]: (1803113040624, 1803117008576)

In [21]:

```

1 s = "    JUNK "
2 u = s.strip() # Removes space
3 print(s)
4 print(u)
5 id(s), id(u)

```

JUNK
JUNK

Out[21]: (1803116851248, 1803116855024)

In [22]:

```

1 s = "***    JUNK    ***"
2 u = s.strip('*') # Removes *
3 print(s)
4 print(u)
5 id(s), id(u)

```

*** JUNK ***
JUNK

Out[22]: (1803117092144, 1803116881072)

In [23]:

```
1 s = "***    JUNK    ***"
2 s1 = s.strip('*') # Removes *
3 print(s)
4 print(s1)
5 s2 = s1.strip() # Removes blanks
6 print(s2)
7 print(id(s), id(s1), id(s2))
8 s2 = s2 + str(100)
9 print(s2)
10 print(id(s), id(s1), id(s2))
```

```
***    JUNK    ***
JUNK
JUNK
1803117092592 1803117092464 1803117092144
JUNK100
```

Out[23]: (1803117092592, 1803117092464, 1803117101104)

1.6 List

list

Copyright: Jagadeesh Vasudevamurthy

filename: list.ipynb

Basic imports

In [1]:

```
1 import sys
2 import os
3 print(sys.version)
```

3.7.7 (default, May 6 2020, 11:45:54) [MSC v.1916 64 bit (AMD64)]

Lists. Hetrogenous container and Mutable

List = []

In [2]:

```
1 s =[1,2,5]
2 print("type of s", type(s), "Value of s =", s, "Length of s =", len(s), "id(s) = ", id(s))
3 for i in range(0,len(s)):
4     print("s[",i,"]",s[i],"id[",i, "]",id(s[i]))
5 print("Creating new s")
6 s = ["Mysore", 56, True, 9.89]
7 print("type of s", type(s), "Value of s =", s, "Length of s =", len(s), "id(s) = ", id(s))
8 for i in range(0,len(s)):
9     print("s[",i,"]",s[i],"id[",i, "]",id(s[i]))
```

type of s <class 'list'> Value of s = [1, 2, 5] Length of s = 3 id(s) = 2125951159048
s[0] 1 id[0] 140718491476368
s[1] 2 id[1] 140718491476400
s[2] 5 id[2] 140718491476496
Creating new s
type of s <class 'list'> Value of s = ['Mysore', 56, True, 9.89] Length of s = 4 id(s) = 2125949955976
s[0] Mysore id[0] 2125951487152
s[1] 56 id[1] 140718491478128
s[2] True id[2] 140718490954064
s[3] 9.89 id[3] 2125950268144

Lists are mutable and can be updated in

place

Concept of Referential Arrays

```
In [3]: 1 s =[1,2,5,10]
2 print("type of s", type(s), "Value of s =", s, "Length of s =", len(s), "id(s) =")
3 for i in range(0,len(s)):
4     print("s[",i,"]",s[i],"id[",i, "]",id(s[i]))
5 s[2] = ['a','e','i','o','u']
6 print("type of s", type(s), "Value of s =", s, "Length of s =", len(s), "id(s) =")
7 for i in range(0,len(s)):
8     print("s[",i,"]",s[i],"id[",i, "]",id(s[i]))
9 s[3] = [78,[89,90,True],"UCSC"]
10 print("type of s", type(s), "Value of s =", s, "Length of s =", len(s), "id(s) =")
11 for i in range(0,len(s)):
12     print("s[",i,"]",s[i],"id[",i, "]",id(s[i]))
```

```
type of s <class 'list'> Value of s = [1, 2, 5, 10] Length of s = 4 id(s) = 2125951748360
5951748360
s[ 0 ] 1 id[ 0 ] 140718491476368
s[ 1 ] 2 id[ 1 ] 140718491476400
s[ 2 ] 5 id[ 2 ] 140718491476496
s[ 3 ] 10 id[ 3 ] 140718491476656
type of s <class 'list'> Value of s = [1, 2, ['a', 'e', 'i', 'o', 'u'], 10] Length of s = 4 id(s) = 2125951748360
s[ 0 ] 1 id[ 0 ] 140718491476368
s[ 1 ] 2 id[ 1 ] 140718491476400
s[ 2 ] ['a', 'e', 'i', 'o', 'u'] id[ 2 ] 2125951159944
s[ 3 ] 10 id[ 3 ] 140718491476656
type of s <class 'list'> Value of s = [1, 2, ['a', 'e', 'i', 'o', 'u'], [78, [89, 90, True], 'UCSC']] Length of s = 4 id(s) = 2125951748360
s[ 0 ] 1 id[ 0 ] 140718491476368
s[ 1 ] 2 id[ 1 ] 140718491476400
s[ 2 ] ['a', 'e', 'i', 'o', 'u'] id[ 2 ] 2125951159944
s[ 3 ] [78, [89, 90, True], 'UCSC'] id[ 3 ] 2125951748808
```

Changing the first element

```
In [4]: 1 s = [8,5]
2 print("type of s", type(s), "Value of s =", s, "Length of s =", len(s), "id(s) =")
3 s[0] = 6
4 print("type of s", type(s), "Value of s =", s, "Length of s =", len(s), "id(s) =")
```

```
type of s <class 'list'> Value of s = [8, 5] Length of s = 2 id(s) = 2125951484
296
type of s <class 'list'> Value of s = [6, 5] Length of s = 2 id(s) = 2125951484
296
```

Changing the end of list

In [5]:

```

1 s = [8,5]
2 print("type of s", type(s), "Value of s =", s, "Length of s =", len(s), "id(s) =", id(s))
3 s[-1] = "jag"
4 print("type of s", type(s), "Value of s =", s, "Length of s =", len(s), "id(s) =", id(s))

```

```

type of s <class 'list'> Value of s = [8, 5] Length of s = 2 id(s) = 2125951173
896
type of s <class 'list'> Value of s = [8, 'jag'] Length of s = 2 id(s) = 212595
1173896

```

Extracting elements of a list

In [6]:

```

1 s = ["Mysore", 56, True, 9.89]
2 print("type of s", type(s), "Value of s =", s, "Length of s =", len(s), "id(s) =", id(s))
3 print("s[0] = ", s[0], "type(s[0]) is ", type(s[0]))
4 print("s[1] = ", s[1], "type(s[1]) is ", type(s[1]))
5 print("s[2] = ", s[2], "type(s[2]) is ", type(s[2]))
6 print("s[3] = ", s[3], "type(s[3]) is ", type(s[3]))
7 print("s[-1] = ", s[-1], "type(s[-1]) is ", type(s[-1]))
8 print("s[-2] = ", s[-2], "type(s[-2]) is ", type(s[-2]))
9 print("s[-3] = ", s[-3], "type(s[-3]) is ", type(s[-3]))
10 print("s[-4] = ", s[-4], "type(s[-4]) is ", type(s[-4]))

```

```

type of s <class 'list'> Value of s = ['Mysore', 56, True, 9.89] Length of s =
4 id(s) = 2125951450632
s[0] = Mysore type(s[0]) is <class 'str'>
s[1] = 56 type(s[1]) is <class 'int'>
s[2] = True type(s[2]) is <class 'bool'>
s[3] = 9.89 type(s[3]) is <class 'float'>
s[-1] = 9.89 type(s[-1]) is <class 'float'>
s[-2] = True type(s[-2]) is <class 'bool'>
s[-3] = 56 type(s[-3]) is <class 'int'>
s[-4] = Mysore type(s[-4]) is <class 'str'>

```

Extracting parts of a list

In [7]:

```

1 s = ["Mysore", 56, True, 9.89]
2 print("type of s", type(s), "Value of s =", s, "Length of s =", len(s), "id(s) =",
3 a = s[0:3] ;
4 print("Getting first 3 elements s[0:3], type of s", type(a), "Value of a ='", a,
5 a = s[:3] ;
6 print("Getting first 3 elements s[:3], type of s", type(a), "Value of a ='", a,
7 a = s[2:len(s)] ;
8 print("Getting last 2 elements s[2:len(s)], type of a", type(a), "Value of a =",
9 a = s[2:] ;
10 print("Getting last 2 elements s[2:], type of a", type(a), "Value of a =", a)

```

```

type of s <class 'list'> Value of s = ['Mysore', 56, True, 9.89] Length of s =
4 id(s) = 2125951375944
Getting first 3 elements s[0:3], type of s <class 'list'> Value of a = ['Mysor
e', 56, True] Length of a 3 2125951450632
Getting first 3 elements s[:3], type of s <class 'list'> Value of a = ['Mysor
e', 56, True] Length of a 3 2125951159944
Getting last 2 elements s[2:len(s)], type of a <class 'list'> Value of a = [Tru
e, 9.89] Length of a 2 2125951450632
Getting last 2 elements s[2:], type of a <class 'list'> Value of a = [True, 9.8
9] Length of a 2 2125951159944

```

Difference between List and String

In [8]:

```

1 s = ["Mysore", 56, True, 9.89]
2 print("type of s", type(s), "Value of s =", s, "Length of s =", len(s), "id(s) =",

```

```

type of s <class 'list'> Value of s = ['Mysore', 56, True, 9.89] Length of s =
4 id(s) = 2125951456776

```

In [9]:

```

1 s = ["Mysore", 56, True, 9.89]
2 a = s[0:1] ;
3 print("Getting first elements s[0:1], type of s", type(a), "Value of a =", a,
4 a = s[0]
5 print("Getting first elements s[0], type of s", type(a), "Value of a =", a)
6 print("python differentiate list[0:1] as a list but list[0] as an object")

```

```

Getting first elements s[0:1], type of s <class 'list'> Value of a = ['Mysor
e'] Length of a = 1 2125951414664
Getting first elements s[0], type of s <class 'str'> Value of a = Mysore Lengt
h of a = 6 2125951687792
python differentiate list[0:1] as a list but list[0] as an object

```

In [10]:

```

1 s = "abcd"
2 a = s[1:2]
3 b = s[1]
4 print("type of s", type(s), "Value of s =", s, "Length of s =", len(s), id(s))
5 print("type of a", type(a), "Value of a =", a, "Length of a =", len(a), id(a))
6 print("type of b", type(b), "Value of b =", b, "Length of b =", len(b), id(b))
7 print("python does not differentiate. string[0:1] as well as string[0] is a string")

```

```

type of s <class 'str'> Value of s = abcd Length of s = 4 2125951402032
type of a <class 'str'> Value of a = b Length of a = 1 2125876480816
type of b <class 'str'> Value of b = b Length of b = 1 2125876480816
python does not differentiate. string[0:1] as well as string[0] is a string

```

Nested Lists

In [11]:

```

1 s = [[["Mysore", 56], [True, 9.89], 23, [87]]]
2 print("type of s", type(s), "Value of s =", s, "Length of s", len(s))
3 print("type of s[0]", type(s[0]), "Value of s[0] =", s[0], "Length of s[0]")
4 print("type of s[1]", type(s[1]), "Value of s[1] =", s[1], "Length of s[1]")
5 print("type of s[2]", type(s[2]), "Value of s[2] =", s[2]) #We cannot print
6 print("type of s[3]", type(s[3]), "Value of s[3] =", s[3], "Length of s[3]")

```

```

type of s <class 'list'> Value of s = [[['Mysore', 56], [True, 9.89], 23, [87]]]
Length of s 4
type of s[0] <class 'list'> Value of s[0] = ['Mysore', 56] Length of s[0] = 2
type of s[1] <class 'list'> Value of s[1] = [True, 9.89] Length of s[1] = 2
type of s[2] <class 'int'> Value of s[2] = 23
type of s[3] <class 'list'> Value of s[3] = [87] Length of s[3] = 1

```

In [12]:

```

1 s = [[2,[6.8]],True,['UCSC']]
2 print("type of s", type(s), "Value of s =", s, "Length of s", len(s))
3 print("type of s[0]", type(s[0]), "Value of s[0] =", s[0], "Length of s[0]")
4 print("type of s[0][0]", type(s[0][0]), "Value of s[0][0] =", s[0][0])
5 print("type of s[0][1]", type(s[0][1]), "Value of s[0][1] =", s[0][1], "Length of s[0][1]")
6 print("type of s[0][1][0]", type(s[0][1][0]), "Value of s[0][1][0] =", s[0][1][0])
7 print("type of s[1]", type(s[1]), "Value of s[1] =", s[1])
8 print("type of s[2]", type(s[2]), "Value of s[2] =", s[2], "Length of s[2]")
9 print("type of s[2][0]", type(s[2][0]), "Value of s[2][0] =", s[2][0])

```

```

type of s <class 'list'> Value of s = [[2, [6.8]], True, ['UCSC']] Length of s 3
type of s[0] <class 'list'> Value of s[0] = [2, [6.8]] Length of s[0] = 2
type of s[0][0] <class 'int'> Value of s[0][0] = 2
type of s[0][1] <class 'list'> Value of s[0][1] = [6.8] Length of s[0][1] = 1
type of s[0][1][0] <class 'float'> Value of s[0][1][0] = 6.8
type of s[1] <class 'bool'> Value of s[1] = True
type of s[2] <class 'list'> Value of s[2] = ['UCSC'] Length of s[2] = 1
type of s[2][0] <class 'str'> Value of s[2][0] = UCSC

```

Modifying Lists

```
In [13]: 1 s = [[2,[6.8]],True,['UCSC']]
2 print("type of s", type(s), "Value of s =", s, "Length of s", len(s))
3 s[1] = False
4 s[2][0] = "UCB"
5 print("type of s", type(s), "Value of s =", s, "Length of s", len(s))
6 print("Strings are immutable. Lists are mutable")
7 s[0][1][0] = 3.145678
8 print("type of s", type(s), "Value of s =", s, "Length of s", len(s))
```

```
type of s <class 'list'> Value of s = [[2, [6.8]], True, ['UCSC']] Length of s
3
type of s <class 'list'> Value of s = [[2, [6.8]], False, ['UCB']] Length of s
3
Strings are immutable. Lists are mutable
type of s <class 'list'> Value of s = [[2, [3.145678]], False, ['UCB']] Length
of s 3
```

Immutable VS Mutable basic types

```
In [14]: 1 x = 10
2 y = x #y is pointing to x
3 print("BEFORE: x = ",x, " y = ",y, id(x), id(y))
4 x = 5 #x will be changed to 5
5 print("AFTER changing x = 5: x = ",x, " y = ",y, id(x), id(y))
```

```
BEFORE: x = 10 y = 10 140718491476656 140718491476656
AFTER changing x = 5: x = 5 y = 10 140718491476496 140718491476656
```

Immutable VS Mutable string

```
In [15]: 1 x = "abc"
2 y = x #y is pointing to x
3 print("BEFORE: x = ",x, " y = ",y, id(x), id(y))
4 x = "xyz" #x cannot be changed. A new string x is created with value xyz
5 print("AFTER changing x = xyz: x = ",x, " y = ",y, id(x), id(y))
6 print("Immutable objects are copied. If you change one object, the other ot
```

```
BEFORE: x = abc y = abc 2125876583472 2125876583472
AFTER changing x = xyz: x = xyz y = abc 2125951674096 2125876583472
Immutable objects are copied. If you change one object, the other objects will
not change
```

75 Immutable VS Mutable basic list

In [16]:

```

1 x = ["first", "second"]
2 y = x ; #y is pointing to x
3 print("BEFORE: x = ",x, " y = ",y, id(x), id(y))
4 x[1] = 23 ;
5 print("AFTER changing x[1] = 23: x = ",x, " y = ",y,id(x), id(y))
6 print("Mutable objects are NOT copied. If you change one object, all object"

```

BEFORE: x = ['first', 'second'] y = ['first', 'second'] 2125951484296 2125951484296
1484296
AFTER changing x[1] = 23: x = ['first', 23] y = ['first', 23] 2125951484296 2125951484296
Mutable objects are NOT copied. If you change one object, all objects will be changed

How to make an array of numbers?

In [17]:

```

1 s = []
2 for i in range(1,10,2):
3     s.append(i)
4 print("type of s", type(s), "Value of s =", s, "Length of s", len(s))

```

type of s <class 'list'> Value of s = [1, 3, 5, 7, 9] Length of s 5

How to make a matrix of integers?

In [18]:

```
1 m = []
2 r = 3
3 c = 2
4 k = 0 ;
5 for i in range(0,r):
6     row = []
7     m.append(row)
8     for j in range(0,c):
9         k = k + 1
10        row.append(k)
11
12 for i in range(0,r):
13     row = m[i]
14     print(row)
15
16 for i in range(0,r):
17     print("Row ", i , ":" ,end ="")
18     for j in range(0,c):
19         print(m[i][j], " " , end ="")
20     print() ;
```

```
[1, 2]
[3, 4]
[5, 6]
Row 0 :1 2
Row 1 :3 4
Row 2 :5 6
```

Copying Lists

In [19]:

```

1 a = [23, "UCSC"]
2 b = a ;
3 print("BEFORE a = ", a , id(a), " b = ",b, id(b))
4
5 print("Concept 1: A slice creates a new sublist from list")
6 print("Concept 2: s[0:k] == s[:k]. Create a new list from s[0] to s[k-1]")
7 print("Concept 3: s[k:len(s)] == s[k: ]. Create a new list from s[k] to s[r]
8 print("Concept 4: s[:] gives a copy of full slice. Create a new list from s
9
10 s = a[0:1]
11 print("s = a[0:1] = ", s, "length of s =",len(s), "type of s =", type(s),
12
13
14 s = a[:1]
15 print("s = a[:1] = ", s, "length of s =",len(s), "type of s =", type(s), i
16
17
18 s = a[1:]
19 print("s = a[1:] = ", s, "length of s =",len(s), "type of s =", type(s), i
20
21 s = a[:]
22 print("s = a[:] = ", s, "length of s =",len(s), "type of s =", type(s), id(s)
23

```

```

BEFORE a = [23, 'UCSC'] 2125951856200 b = [23, 'UCSC'] 2125951856200
Concept 1: A slice creates a new sublist from list
Concept 2: s[0:k] == s[:k]. Create a new list from s[0] to s[k-1]
Concept 3: s[k:len(s)] == s[k: ]. Create a new list from s[k] to s[n-1]
Concept 4: s[:] gives a copy of full slice. Create a new list from s[0] to s[n-1]
s = a[0:1] = [23] length of s = 1 type of s = <class 'list'> 2125951803464
s = a[:1] = [23] length of s = 1 type of s = <class 'list'> 2125951375560
s = a[1:] = ['UCSC'] length of s = 1 type of s = <class 'list'> 2125951803464
s = a[:] [23, 'UCSC'] length of s = 2 type of s = <class 'list'> 2125951375560

```

Equality of list

a == b checks whether two lists have value. Deep equal

a is b checks whether two lists points to same memory location.

In [20]:

```

1 a = [23, "UCSC"]
2 b = a
3 print("BEFORE a = ", a , "id of a ", id(a), " b = ",b,"id of b ", id(b))
4 print("BEFORE a == b", (a==b)) ## DEEP equality
5 print("BEFORE object a refers to object b = a is b", (a is b)) ##SHALLOW eq
6 a[1] = "UCB"
7 print("After a = ", a , "id of a ", id(a), " b = ",b,"id of b ", id(b))
8 print("AFTER a == b", (a==b)) ## DEEP equality
9 print("AFTER object a refers to object b = a is b", (a is b)) ##SHALLOW eq

```

BEFORE a = [23, 'UCSC'] id of a 2125951748360 b = [23, 'UCSC'] id of b 2125951748360
BEFORE a == b True
BEFORE object a refers to object b = a is b True
After a = [23, 'UCB'] id of a 2125951748360 b = [23, 'UCB'] id of b 2125951748360
AFTER a == b True
AFTER object a refers to object b = a is b True

In [21]:

```

1 a = [23, "UCSC"]
2 b = [23, "UCSC"]
3 c = [23, "Ucsc"]
4 print("BEFORE a = ", a , "id of a ", id(a), " b = ",b,"id of b ", id(b), " c "
5 print("BEFORE a == b", (a==b)) ## DEEP equality
6 print("BEFORE a == c", (a==c)) ## DEEP equality
7 print("BEFORE object a refers to object b = a is b", (a is b))##SHALLOW eq
8 a[1] = "UCB"
9 print("AFTER a = ", a , "id of a ", id(a), " b = ",b,"id of b ", id(b))
10 print("AFTER a == b", (a==b)) ## DEEP equality
11 print("AFTER object a refers to object b = a is b", (a is b)) ##SHALLOW eq

```

BEFORE a = [23, 'UCSC'] id of a 2125949955976 b = [23, 'UCSC'] id of b 2125951184008 c = [23, 'Ucsc'] id of c 2125951856008
BEFORE a == b True
BEFORE a == c False
BEFORE object a refers to object b = a is b False
AFTER a = [23, 'UCB'] id of a 2125949955976 b = [23, 'UCSC'] id of b 2125951184008
AFTER a == b False
AFTER object a refers to object b = a is b False

List concatenation

In [22]:

```

1 a = [23, "UCSC"]
2 b = [100]
3 print("First a = ", a , "id of a ", id(a), " b = ",b,"id of b ", id(b))
4 c = a
5 print("Before a = ", a , " c = ", c, "(a == c)", (a == c))
6 print("Before a = ", a , " c = ", c, "(a is c)", (a is c))
7 c = a + b ;
8 print("c = a + b; To show concatenation(+) always produces new list")
9 print("After a = ", a , " c = ", c, "(a == c)", (a == c))
10 print("After a = ", a , " c = ", c, "(a is c)", (a is c))
11 print("After b = ", b)

```

```

First a = [23, 'UCSC'] id of a 2125951747976 b = [100] id of b 21259518558
80
Before a = [23, 'UCSC'] c = [23, 'UCSC'] (a == c) True
Before a = [23, 'UCSC'] c = [23, 'UCSC'] (a is c) True
c = a + b; To show concatenation(+) always produces new list
After a = [23, 'UCSC'] c = [23, 'UCSC', 100] (a == c) False
After a = [23, 'UCSC'] c = [23, 'UCSC', 100] (a is c) False
After b = [100]

```

Extending a list

In [23]:

```

1 a = [23, "UCSC"]
2 b = a
3 print("a = ", a , "id of a ", id(a), " b = ",b,"id of b ", id(b))
4 print("Appending a list using +. This creates a new list ")
5 a = a + ["MIT"]
6 print("a = ", a , "id of a ", id(a), " b = ",b,"id of b ", id(b))
7 print("Because appending a list using + created a new list, b is still pointing to old value")
8 print("Appending a list using append will not create a new list ")

```

```

a = [23, 'UCSC'] id of a 2125951856008 b = [23, 'UCSC'] id of b 2125951856
008
Appending a list using +. This creates a new list
a = [23, 'UCSC', 'MIT'] id of a 2125951450632 b = [23, 'UCSC'] id of b 212
5951856008
Because appending a list using + created a new list, b is still pointing to old value
Appending a list using append will not create a new list

```

In [24]:

```

1 a = [100]
2 b = a ;
3 print("BEFORE a = ", a , "id of a ", id(a), " b = ",b,"id of b ", id(b))
4 a.append("[USCF]")
5 print("a = ", a , "id of a ", id(a), " b = ",b,"id of b ", id(b))
6 print("Because appending a list using append, did not create a new list, b

```

BEFORE a = [100] id of a 2125951748168 b = [100] id of b 2125951748168
a = [100, '[USCF]'] id of a 2125951748168 b = [100, '[USCF]'] id of b 2125
951748168

Because appending a list using append, did not create a new list, b is still pointing to a

In [25]:

```

1 print("Extending a list with another list, will not create a new list")
2 a = [23, "UCSC"]
3 b = [5.6,True]
4 c = a
5 print("BEFORE a = ", a , "id of a ", id(a), " b = ",b,"id of b ", id(b)," c
6 a.extend(b)
7 print("AFTER a = ", a , "id of a ", id(a), " b = ",b,"id of b ", id(b)," c =
8 print("This is NOT same as a = a + b")

```

Extending a list with another list, will not create a new list
BEFORE a = [23, 'UCSC'] id of a 2125951450184 b = [5.6, True] id of b 2125
951414728 c = [23, 'UCSC'] id of c 2125951450184
AFTER a = [23, 'UCSC', 5.6, True] id of a 2125951450184 b = [5.6, True] id
of b 2125951414728 c = [23, 'UCSC', 5.6, True] id of c 2125951450184
This is NOT same as a = a + b

In [26]:

```

1 a = [23, "UCSC"]
2 b = [5.6,True]
3 c = a
4 print("BEFORE a = ", a , "id of a ", id(a), " b = ",b,"id of b ", id(b)," c
5 a = a + b
6 print("AFTER a = ", a , "id of a ", id(a), " b = ",b,"id of b ", id(b)," c =

```

BEFORE a = [23, 'UCSC'] id of a 2125951426632 b = [5.6, True] id of b 2125
951450696 c = [23, 'UCSC'] id of c 2125951426632
AFTER a = [23, 'UCSC', 5.6, True] id of a 2125951450184 b = [5.6, True] id
of b 2125951450696 c = [23, 'UCSC'] id of c 2125951426632

List membership

In [27]:

```
1 a = [23, "UCSC", 4.73, "UCSB"]
2 print(a)
3 print("23 in a =", 23 in a)
4 print("4.73 in a =", 4.73 in a)
5 print("'UCSB' in a =", 'UCSB' in a)
6 print("'UCsB' in a =", 'UCsB' in a)
7 if (4.73 in a):
8     a.remove(4.73)
9 print(a)
10 a = [23, "UCSC", 4.73, "UCSB", 23, "UCSC", 4.73, "UCSB"]
11 print(a)
12 while ("UCSB" in a):
13     a.remove("UCSB")
14 print(a)
```

```
[23, 'UCSC', 4.73, 'UCSB']
23 in a = True
4.73 in a = True
'UCSB' in a = True
'UCsB' in a = False
[23, 'UCSC', 'UCSB']
[23, 'UCSC', 4.73, 'UCSB', 23, 'UCSC', 4.73, 'UCSB']
[23, 'UCSC', 4.73, 23, 'UCSC', 4.73]
```

Other list functions

In [28]:

```
1 l = list(range(5,13))
2 a = l
3 print(l)
4 print(a)
5 print("id of l ", id(l), "id of a ", id(a))
6 print("After l.reverse(). Reverse in place")
7 l.reverse()
8 print(l)
9 print(a)
10 print("id of l ", id(l), "id of a ", id(a))
11 print('-----')
12 print("After l.sort(). sort in ascending order in place")
13 l.sort()
14 print(l)
15 print(a)
16 print("id of l ", id(l), "id of a ", id(a))
```

```
[5, 6, 7, 8, 9, 10, 11, 12]
[5, 6, 7, 8, 9, 10, 11, 12]
id of l  2125951806792 id of a  2125951806792
After l.reverse(). Reverse in place
[12, 11, 10, 9, 8, 7, 6, 5]
[12, 11, 10, 9, 8, 7, 6, 5]
id of l  2125951806792 id of a  2125951806792
-----
After l.sort(). sort in ascending order in place
[5, 6, 7, 8, 9, 10, 11, 12]
[5, 6, 7, 8, 9, 10, 11, 12]
id of l  2125951806792 id of a  2125951806792
```

Complexity of Python list

```
In [ ]: 1
```

In [29]:

```
1 import os
2 path = "C:\\\\Users\\\\jag\\\\OneDrive\\\\vasu\\\\work\\\\py3\\\\objects\\\\py3\\\\py3\\\\pdf\\\\"
3 rpath = os.path.relpath(path)
4 from IPython.display import IFrame
5 display(IFrame(rpath, width=800, height=600))
```

1.7 Tuple

Tuples in Python

Copyright: Jagadeesh Vasudevamurthy

filename: tuple.ipynb

string " " list [] tuple () set {} Dictionary {}

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Like list (unlike array) tuples can have heterogeneous objects of any size

Creating empty list and empty Tuple

In [4]:

```
1 list = []
2 print(list)
3 tuple = ()
4 print(tuple)
```

```
[]  
()
```

Creating list and tuple

In [5]:

```
1 list = ['physics', 'chemistry', 1997, 2000]
2 print(list)
3 tuple = ('physics', 'chemistry', 1997, 2000);
4 print(tuple)
```

```
['physics', 'chemistry', 1997, 2000]
('physics', 'chemistry', 1997, 2000)
```

In [6]:

```

1 tup1 = ('physics', 'chemistry', 1997, 2000);
2 tup2 = (1, 2, 3, 4, 5, 6, 7 );
3 print("tup1[0]: ", tup1[0]);
4 print("tup2[1:5]: ", tup2[1:5]);

```

tup1[0]: physics
tup2[1:5]: (2, 3, 4, 5)

In [7]:

```

1 list = ['physics', 'chemistry', 1997, 2000]
2 print(list)
3 tuple = ('physics', 'chemistry', 1997, 2000);
4 print(tuple)
5 list[0] = 67
6 print(list)
7 #tuple[0] = 67
8 print('''tuple[0] = 67 YOU WILL GETTypeError: 'tuple' object does not supp

```

['physics', 'chemistry', 1997, 2000]
('physics', 'chemistry', 1997, 2000)
[67, 'chemistry', 1997, 2000]
tuple[0] = 67 YOU WILL GETTypeError: 'tuple' object does not support item assignment

Accessing Values in Tuples

In [8]:

```

1 tup1 = ('physics', 'chemistry', 1997, 2000)
2 tup2 = (1, 2, 3, 4, 5, 6, 7 )
3 print("tup1[0]: ", tup1[0])
4 print("tup2[1:5]: ", tup2[1:5])

```

tup1[0]: physics
tup2[1:5]: (2, 3, 4, 5)

In [9]:

```

1 tup = ('physics', 'chemistry', 1997, 2000)
2 for i in tup:
3     print(i)

```

physics
chemistry
1997
2000

Basic Tuples Operations

```
In [10]: 1 t = (1,2,3)
          2 print(t)
          3 print(len(t))
          4
```

```
(1, 2, 3)
3
```

```
In [11]: 1 t1 = (1,2,3)
          2 print(t1)
          3 t2 = (100,200)
          4 print(t2)
          5 t = t1 + t2
          6 print(t)
```

```
(1, 2, 3)
(100, 200)
(1, 2, 3, 100, 200)
```

```
In [12]: 1 hi = "hi"
          2 print(hi)
          3 t = hi * 4
          4 print(t)
```

```
hi
hihihihi
```

```
In [13]: 1 t = (1,2,3)
          2 print(t)
          3 print(3 in t)
```

```
(1, 2, 3)
True
```

```
In [14]: 1 t = (1,2,3)
          2 print(t)
          3 print(4 in t)
```

```
(1, 2, 3)
False
```

```
In [15]: 1 t1 = (1,2,3)
          2 t2 = (1,2,3)
          3 print(t1 == t2)
```

True

```
In [16]: 1 t1 = (1,2,3)
          2 t2 = (3,2,1)
          3 print(t1 == t2)
```

False

```
In [17]: 1 t = (1,2,3)
          2 print(max(t))
```

3

```
In [18]: 1 t = (1,2,3)
          2 print(min(t))
```

1

```
In [ ]:
```

1

1.8 Set

Set Data Structure

Copyright: Jagadeesh Vasudevamurthy

filename: set.ipynb

string " " list [] tuple () set {} Dictionary {}

set

```
1 k = {}
2 print("type of k", type(k), "Value of k =", k)
```

In [1]:
1 k = set() # {} is both for set and dictionary. So you have to explicitly say
2 print("type of k", type(k), "Value of k =", k)

```
type of k <class 'set'> Value of k = set()
```

```
1 # adding to set
```

In [2]:
1 k.add(4)
2 k.add(2)
3 k.add(5)
4 print("type of k", type(k), "Value of k =", k)
5 print("Note that order is lost")

```
type of k <class 'set'> Value of k = {2, 4, 5}
Note that order is lost
```

In [3]:
1 n = 4
2 if (n in k):
3 print(n, "Is there in set", k)

```
4 Is there in set {2, 4, 5}
```

```
In [4]: 1 n = 7  
2 if (n not in k):  
3     print(n, "Is NOT there in set",k)
```

7 Is NOT there in set {2, 4, 5}

```
In [5]: 1 n = 4  
2 if (n in k):  
3     k.remove(n)  
4 print("type of k", type(k), "Value of k =", k)
```

type of k <class 'set'> Value of k = {2, 5}

```
In [6]: 1 num = len(k)  
2 print("Number of elements in set k",num)
```

Number of elements in set k 2

```
In [7]: 1 for e in k:  
2     print(e)
```

2
5

```
In [ ]: 1
```

1.9 Dictionary

Dictionary Data Structure

Copyright: Jagadeesh Vasudevamurthy

filename: dictionary.ipynb

string " " list [] tuple () set {} Dictionary {}

Dictionary or hash

Creating empty dictionary

```
In [17]: 1 d = {}
2 print("type of d", type(d), "Value of d =", d)
```

```
type of d <class 'dict'> Value of d = {}
```

Creating dictionary with key,value pairs

Key is SSN (int). Value is name(string)

```
In [18]: 1 d = {616412450: 'Jag', 510998776: 'bob', 510998777: 'bob'}
2 print("type of d", type(d), "Value of d =", d)
```

```
type of d <class 'dict'> Value of d = {616412450: 'Jag', 510998776: 'bob', 51
0998777: 'bob'}
```

Adding new key,value pair to an existing dictionary

```
In [19]: 1 d = {616412450: 'Jag', 510998776: 'bob', 510998777: 'bob'}
2 print("type of d", type(d), "Value of d =", d)
3 d[16786910] = "police"
4 print("type of d", type(d), "Value of d =", d)
```

```
type of d <class 'dict'> Value of d = {616412450: 'Jag', 510998776: 'bob', 51
0998777: 'bob'}
```

```
type of d <class 'dict'> Value of d = {616412450: 'Jag', 510998776: 'bob', 51
0998777: 'bob', 16786910: 'police'}
```

is in dictionary

In [20]:

```

1 d = {616412450: 'Jag', 510998776: 'bob', 510998777: 'bob'}
2 print("type of d", type(d), "Value of d =", d)
3 k = 510998776
4 if (k in d):
5     print(k, "is in dictionary. Its Value is", d[k])
6 else:
7     print(k, "is in NOT in dictionary")

```

```

type of d <class 'dict'> Value of d = {616412450: 'Jag', 510998776: 'bob', 51
0998777: 'bob'}
510998776 is in dictionary. Its Value is bob

```

NOT in dictionary

In [21]:

```

1 d = {616412450: 'Jag', 510998776: 'bob', 510998777: 'bob'}
2 print("type of d", type(d), "Value of d =", d)
3 k = 'bob'
4 if (k in d):
5     print(k, "is in dictionary. Its Value is", d[k])
6 else:
7     print(k, "is in NOT in dictionary")

```

```

type of d <class 'dict'> Value of d = {616412450: 'Jag', 510998776: 'bob', 51
0998777: 'bob'}
bob is in NOT in dictionary

```

Inserting new key (Value can be same). <key, value> will be inserted

```

d = {1: 'Jag', 2: 'bob', 3: 'bob'} print("type of d", type(d), "Value of d =", d) k = 100 d[k] = 'bob'
print("type of d", type(d), "Value of d =", d)

```

Inserting same key with new value. value will be changed

In [22]:

```

1 d = {1: 'Jag', 2: 'bob', 3: 'bob'}
2 print("type of d", type(d), "Value of d =", d)
3 k = 2
4 d[k] = 'john'
5 print("type of d", type(d), "Value of d =", d)

```

```

type of d <class 'dict'> Value of d = {1: 'Jag', 2: 'bob', 3: 'bob'}
type of d <class 'dict'> Value of d = {1: 'Jag', 2: 'john', 3: 'bob'}

```

Inserting same key with same value. Nothing will be added

In [23]:

```

1 d = {1: 'Jag', 2: 'bob'}
2 print("type of d", type(d), "Value of d =", d)
3 k = 2
4 d[k] = 'bob'
5 print("type of d", type(d), "Value of d =", d)

```

```

type of d <class 'dict'> Value of d = {1: 'Jag', 2: 'bob'}
type of d <class 'dict'> Value of d = {1: 'Jag', 2: 'bob'}

```

Getting a value record given key

In [24]:

```

1 d = {1: 'Jag', 2: 'bob', 3: 'bob'}
2 print("type of d", type(d), "Value of d =", d)
3 k = 2
4 if k in d:
5     v = d[k]
6     print("key ", k, "has Value", v)
7 else:
8     print("there is no key", k)
9 v = d[1]
10 print(v)

```

```

type of d <class 'dict'> Value of d = {1: 'Jag', 2: 'bob', 3: 'bob'}
key 2 has Value bob
Jag

```

In [25]:

```

1 d = {1: 'Jag', 2: 'bob', 3: 'bob'}
2 print("type of d", type(d), "Value of d =", d)
3 k = 101
4 if k in d:
5     v = d[k]
6     print("key ", k, "has Value", v)
7 else:
8     print("there is no key", k)

```

```

type of d <class 'dict'> Value of d = {1: 'Jag', 2: 'bob', 3: 'bob'}
there is no key 101

```

Deleting a key

```
In [26]: 1 d = {1: 'Jag', 2: 'bob', 3: 'bob'}
2 print("type of d", type(d), "Value of d =", d)
3 k = 2
4 if k in d:
5     del d[k]
6 else:
7     print("there is no key", k)
8 print("type of d", type(d), "Value of d =", d)
```

```
type of d <class 'dict'> Value of d = {1: 'Jag', 2: 'bob', 3: 'bob'}
type of d <class 'dict'> Value of d = {1: 'Jag', 3: 'bob'}
```

Iterating over a dictionary

```
In [27]: 1 d = {1: 'Jag', 2: 'bob', 3: 'bob'}
2 print("type of d", type(d), "Value of d =", d)
3 for key, value in d.items():
4     print("Key", key, "Value", value)
```

```
type of d <class 'dict'> Value of d = {1: 'Jag', 2: 'bob', 3: 'bob'}
Key 1 Value Jag
Key 2 Value bob
Key 3 Value bob
```

Python hash()

The hash() method returns the hash value of an object if it has one.

```
In [28]: 1 n = 18100
2 print("n =", n, "hash =", hash(n))
```

```
n = 18100 hash = 18100
```

```
In [29]: 1 n = "Jag"
2 print("n =", n, "hash =", hash(n))
```

```
n = Jag hash = 3117224974171390311
```

Writing hash for user defined objects

VASUDEVAMURTHY

In [30]:

```
1 class Obj:
2     def __init__(self, x, y):
3         # All keys below.
4         # Decision is taken based on key. Do not change key after inserted
5         self._x = x
6         # All values below
7         # We can change these values
8         self._y = y
9
10    def change_key(self,x:'int')->'None':
11        self._x = x
12
13    def change_value(self,x:'int')->'None':
14        self._y = x
15
16
17    def __str__(self)->'string':
18        x = hash(self)
19        s = str(self._x) + "->" + str(self._y) + " key is " + str(self.
20        return s
21
22 ##### WRITE 5 below starts #####
23    def _get_key(self)->'int':
24        return self._x
25
26    def __hash__(self)->'int':
27        return (hash(self._get_key()))
28
29    #if you don't write __hash__ you get Unhashable type Obj
30    #''' # uncomment and test
31
32    def __hash__(self)->'int':
33        h = hash(self._get_key())
34        # DO NOT convert to positive using abs
35        ## Keep in mind that the hash value might be negative,
36        #depending on the string's content and the Python
37        #implementation, but it's perfectly valid to use it as a
38        #hash value for dictionaries or sets.
39        return h
40    #'''# uncomment and test
41
42    def __lt__(self, other)->'bool':
43        if (self._get_key() < other._get_key()):
44            return True
45        return False
46
47    def __eq__(self, other)->'bool':
48        # (a == b) = !(a < b) && !(b < a)
49        x = ((not(self < other)) and (not(other < self)))
50        return x
51
52    def __ne__(self, other)->'bool':
53        return(not(self == other))
```

55

WRITE 5 below starts

Testing Obj

VASUDEVAMURTHY

In [31]:

```
1 class TestObj:
2     def __init__(self):
3         self._test1()
4
5     def _P(self,d:'dict',t:'string')->"None":
6         k = list(d.keys())
7         v = list(d.values())
8         n1 = len(k)
9         n2 = len(v)
10        assert(n1 == n2)
11        print("-----", t, "starts -----")
12        for i in range(n1):
13            print("Obj =",k[i],"Value =",v[i])
14        print("-----", t, "ends -----")
15
16    def _test1(self):
17        # Creating instances of Obj
18        p1 = Obj(1, 10) #key is 1. value does not matter
19        p2 = Obj(2, 20)
20        p3 = Obj(1, 30) #Note key is 1. Not 3
21        p4 = Obj(4, 40)
22
23        # Creating a dictionary
24        # Obj UDT as keys. Value is the string
25        print("add p1 and p2 to dictionary")
26        d = {}
27        d[p1] = "p1" #WITHOUT __hash__ you get Unhashable type Obj
28        d[p2] = "p2"
29        self._P(d,"dict after adding p1 and p2")
30
31
32        print("what happens if use p3 as the key?")
33        if p3 in d:
34            v = d[p3]
35            print(p3,"Value", v)
36        else:
37            print(p3, "NOT THERE")
38        print("From the eyes of dict p3 is p1 as the key is 1. So it finds")
39
40        print("Let us add p3 and p4 to dictionary")
41        d[p3] = "p3"
42        d[p4] = "p4"
43        print("Let us print dictionary. Note we have added p1 p2 p3 and p4")
44        self._P(d,"dict after adding p1,p2,p3 and p4. Note p3 is p1 in the")
45        print("From the eyes of dict p3 is p1 as the key is 1. It just cha")
46
47
48        print("If the key is same, value will be changed. Key cannot be ch")
49        print("Note that p3 = Obj(1, 30), But we still have 10 in p1")
50        print("Number of items in d at this Obj is 3. NOT 4")
51        assert(len(d) == 3)
52
53
54        print("At this point local Obj p1 and p3 is as follows")
55        v = d[p1]           103
56        print("local p1",p1,"Value", v)
57        v = d[p3]
```

```
58     print("local p3",p3,"Value", v)
59
60
61     print("Let us change the value")
62     p1.change_value(100)
63     p2.change_value(200)
64     p3.change_value(300)
65     p4.change_value(400)
66
67     self._P(d,"dict after changing values")
68     print("Note Key is same, You can change OTHER fields")
69     print("ABOVE SHOWS all objects are pointers IN PYTHON, C++ is dif")
70
71     print("At this local Obj p1 and p3 is as follows")
72     v = d[p1]
73     print("local p1",p1,"Value", v)
74     v = d[p3]
75     print("local p3",p3,"Value", v)
76
77
78     print("Let us change the key")
79     p1.change_key(-100)
80     p2.change_key(-200)
81     p3.change_key(-300)
82     p4.change_key(-400)
83
84     self._P(d,"dict after changing keys")
85
86     #This will crash
87     if (False):
88         print("At this Obj p3 is as follows")
89         v = d[p3]
90         print("Key",p3,"Value", v)
91
92         if p3 in d:
93             v = d[p3]
94             print("Key",p3,"Value", v)
95         else:
96             print("Key", p3, "NOT THERE")
97             print("It did not find p3 because nothing was inserted into dict")
98
99         p3.change_key(1)
100        p3.change_value(-300) #value does not matter
101        self._P(d,"dict after changing key of p3 to 1")
102        if p3 in d:
103            v = d[p3]
104            print("Key",p3,"Value", v)
105        else:
106            print("Key", p3, "NOT THERE")
107            print("It did not find p3 because in the eye of dict p3 is p1")
108
109
110        p1.change_key(1)
111        p1.change_value(-300) #value does not matter
112        self._P(d,"dict after changing key of p1 to 1")
113        if p1 in d:
114            v = d[p1]
```

```
115     print("Key",p1,"Value", v)
116     print("It did find p1 because the key is 1 now")
117 else:
118     print("Key", p1, "NOT THERE")
```

test

In [32]: 1 TestObj()

```
add p1 and p2 to dictionary
----- dict after adding p1 and p2 starts -----
Obj = 1->10 key is 1 hash =1 Value = p1
Obj = 2->20 key is 2 hash =2 Value = p2
----- dict after adding p1 and p2 ends -----
what happens if use p3 as the key?
1->30 key is 1 hash =1 Value p1
From the eyes of dict p3 is p1 as the key is 1. So it finds p1
Let us add p3 and p4 to dictionary
Let us print dictionary. Note we have added p1 p2 p3 and p4
----- dict after adding p1,p2,p3 and p4. Note p3 is p1 in the eye of dict st
arts -----
Obj = 1->10 key is 1 hash =1 Value = p3
Obj = 2->20 key is 2 hash =2 Value = p2
Obj = 4->40 key is 4 hash =4 Value = p4
----- dict after adding p1,p2,p3 and p4. Note p3 is p1 in the eye of dict en
ds -----
From the eyes of dict p3 is p1 as the key is 1. It just changes the value fro
m string p1 to string p3
If the key is same, value will be changed. Key cannot be changed
Note that p3 = Obj(1, 30), But we still have 10 in p1
Number of items in d at this Obj is 3. NOT 4
At this point local Obj p1 and p3 is as follows
local p1 1->10 key is 1 hash =1 Value p3
local p3 1->30 key is 1 hash =1 Value p3
Let us change the value
----- dict after changing values starts -----
Obj = 1->100 key is 1 hash =1 Value = p3
Obj = 2->200 key is 2 hash =2 Value = p2
Obj = 4->400 key is 4 hash =4 Value = p4
----- dict after changing values ends -----
Note Key is same, You can change OTHER fields
ABOVE SHOWS all objects are pointers IN PYTHON, C++ is different
At this local Obj p1 and p3 is as follows
local p1 1->100 key is 1 hash =1 Value p3
local p3 1->300 key is 1 hash =1 Value p3
Let us change the key
----- dict after changing keys starts -----
Obj = -100->100 key is -100 hash =-100 Value = p3
Obj = -200->200 key is -200 hash =-200 Value = p2
Obj = -400->400 key is -400 hash =-400 Value = p4
----- dict after changing keys ends -----
Key -300->300 key is -300 hash =-300 NOT THERE
It did not find p3 because nothing was inserted into dict with has hash key o
f 300
----- dict after changing key of p3 to 1 starts -----
Obj = -100->100 key is -100 hash =-100 Value = p3
Obj = -200->200 key is -200 hash =-200 Value = p2
Obj = -400->400 key is -400 hash =-400 Value = p4
----- dict after changing key of p3 to 1 ends -----
Key 1->-300 key is 1 hash =1 NOT THERE
It did not find p3 because in the eye of dict p3 is p1
----- dict after changing key of p1 to 1 starts -----
Obj = 1->-300 key is 1 hash =1 Value = p3
Obj = -200->200 key is -200 hash =-200 Value = p2
Obj = -400->400 key is -400 hash =-400 Value = p4
----- dict after changing key of p1 to 1 ends -----
```

```
Key 1->-300 key is 1 hash =1 Value p3  
It did find p1 because the key is 1 now
```

```
Out[32]: <__main__.TestObj at 0x2178193f7f0>
```

```
In [ ]: 1
```

1.10 List Comprehension

Comprehensions

Copyright: Jagadeesh Vasudevamurthy

filename: Comprehensions.ipynb

Basic imports

In [130]:

```
1 import sys
2 import os
3 print(sys.version)
```

3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]

In Python, list comprehension is a concise way to create new lists based on existing lists or other iterable objects. In other words, it lets you create a new list based on the values in an already created list. You can combine loops, conditional statements, and expressions into a single line of code with the help of list comprehension.

Idea List = []

Syntax: list_comp = [expression for item in iterable if condition]

1. list_comp is the new Python list that we are creating
2. Must have for and in.
3. expression: It represents the operation or a call to a method to be applied to each item in the iterable. The result of which will be added to the new list list_comp.
4. item: It is the variable that represents each item in the iterable. item is used in expression
5. iterable: It is the existing list, set, tuple, string, or any other iterable object that can return its elements one at a time.
6. if condition (optional): This part is used to filter the elements from the iterable based on a condition.

Clone a list

1. Basic List Comprehension

In [131]:

```
1 a = [1,2,3]
2 print("address of a = ", id(a), "value - ", a) ;
3 b = []
4 for e in a:
5     b.append(e)
6 print("address of b = ", id(b), "value - ", b) ;
```

```
address of a = 2408420073920 value - [1, 2, 3]
address of b = 2408420125184 value - [1, 2, 3]
```

In [132]:

```
1 a = [1,2,3]
2 print("address of a = ", id(a), "value - ", a) ;
3 b = [e for e in a]
4 print("address of b = ", id(b), "value - ", b) ;
```

```
address of a = 2408419278976 value - [1, 2, 3]
address of b = 2408419275584 value - [1, 2, 3]
```

Multiply each element of the list by 2

2. List Comprehension With A Condition

In [133]:

```
1 a = [1,2,3]
2 b = []
3 for e in a:
4     b.append(e * 2)
5 print(b)
```

```
[2, 4, 6]
```

In [134]:

```
1 a = [1,2,3]
2 b = [e*2 for e in a]
3 print(b)
```

```
[2, 4, 6]
```

3. List Comprehension With Multiple Conditions

```
In [135]: 1 a = [1, 2, 3, 4, 5]
2 print(a)
3 b = [e**2 for e in a if (e%2==0) or (e==1)]
4 print(b)
```

```
[1, 2, 3, 4, 5]
[1, 4, 16]
```

Convert to upper case

```
In [136]: 1 a = ["ucb", "ne", "ucla"]
2 b = []
3 for e in a:
4     b.append(e.upper())
5 print(b)
```

```
['UCB', 'NE', 'UCLA']
```

```
In [137]: 1 a = ["ucb", "ne", "ucla"]
2 b = [e.upper() for e in a]
3 print(b)
```

```
['UCB', 'NE', 'UCLA']
```

Make a list of only odd number

4. List Comprehension Using Custom Functions

```
In [138]: 1 def isodd(n:'int')->'bool':
2     if (n % 2):
3         return True
4     return False
```

```
In [139]: 1 a = [e for e in range(10)]
2 print("a = ", a)
3 b = []
4 for e in a:
5     if (isodd(e)):
6         b.append(e)
7 print(b)
```

```
a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 3, 5, 7, 9]
```

```
In [140]: 1 a = [e for e in range(10)]
2 print("a = ", a)
3 b = [e for e in a if isodd(e)]
4 print("b = ", b)
```

```
a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
b = [1, 3, 5, 7, 9]
```

5. List Comprehension Using The Ternary Operator

```
In [141]: 1 a = [1, 2, 3, 4, 5]
2 print(a)
3 b = []
4 for e in a:
5     if (e%2):
6         b.append('odd')
7     else:
8         b.append('even')
9 print(b)
```

```
[1, 2, 3, 4, 5]
['odd', 'even', 'odd', 'even', 'odd']
```

```
In [142]: 1 a = [1, 2, 3, 4, 5]
2 print(a)
3 b = [('odd' if e%2 else 'even') for e in a]
4 print(b)
```

```
[1, 2, 3, 4, 5]
['odd', 'even', 'odd', 'even', 'odd']
```

6. List Comprehension Using enumerate

```
In [143]: 1 a = ["ucb", "ne", "ucla"]
2 print(a)
3 i = 0 ;
4 b = []
5 for university in a:
6     b.append((i, university))
7     i+=1
8 print("As tuple",b)
9 i = 0 ;
10 c = []
11 for university in a:
12     c.append([i, university])
13     i+=1
14 print("As list",c)
```

```
['ucb', 'ne', 'ucla']
As tuple [(0, 'ucb'), (1, 'ne'), (2, 'ucla')]
As list [[0, 'ucb'], [1, 'ne'], [2, 'ucla']]
```

```
In [144]: 1 a = ["ucb", "ne", "ucla"]
2 print(a)
3 b = [(i,university) for i,university in enumerate(a)]
4 print("As tuple",b)
5 c = [[i,university] for i,university in enumerate(a)]
6 print("As list",c)
```

```
['ucb', 'ne', 'ucla']
As tuple [(0, 'ucb'), (1, 'ne'), (2, 'ucla')]
As list [[0, 'ucb'], [1, 'ne'], [2, 'ucla']]
```

7. List Comprehension Using zip

```
In [145]: 1 letters = ['a', 'b', 'c']
2 numbers = [4, 5, 6] #remove 6 and see
3 symbols = ['!', '@', '#']
4 a = []
5 n = len(letters)
6 for i in range(n):
7     a.append((letters[i], numbers[i], symbols[i]))
8 print(a)
```

```
[('a', 4, '!'), ('b', 5, '@'), ('c', 6, '#')]
```

```
In [146]: 1 letters = ['a', 'b', 'c']
2 numbers = [4, 5, 6] #remove 6 and see
3 symbols = ['!', '@', '#']
4 a = [ (l, n, s) for l, n, s in zip(letters, numbers, symbols) ]
5 print(a)
```

```
[('a', 4, '!'), ('b', 5, '@'), ('c', 6, '#')]
```

8. List Comprehension With Nested Loops

In [147]:

```

1 letters = ['a', 'b', 'c']
2 numbers = [4, 5]
3 a = []
4 for l in letters:
5     for n in numbers:
6         a.append(l + str(n))
7 print(a)

```

['a4', 'a5', 'b4', 'b5', 'c4', 'c5']

In [148]:

```

1 letters = ['a', 'b', 'c']
2 numbers = [4, 5]
3
4 a = [l+str(n) for l in letters for n in numbers]
5 print(a)

```

['a4', 'a5', 'b4', 'b5', 'c4', 'c5']

9. List Comprehensions With Smaller List Comprehensions Inside

In [149]:

```

1 a = [1,2,3]
2 print(a)
3 b = []
4
5 for e in a:
6     alist = []
7     for i in range(e):
8         alist.append(a[i])
9     b.append([alist])
10 print(b)

```

[1, 2, 3]
[[[1]], [[1, 2]], [[1, 2, 3]]]

In [150]:

```

1 a = [1, 2, 3]
2 print(a)
3
4 b = [[i+1 for i in range(e)] for e in a]
5 print(b)

```

[1, 2, 3]
[[1], [1, 2], [1, 2, 3]]

Flatten a matrix

```
In [151]: 1 matrix = [[1, 2],  
2             [4, 5],  
3             [7, 8]]  
4 print(matrix)
```

```
[[1, 2], [4, 5], [7, 8]]
```

```
In [152]: 1 a = []  
2 for arow in matrix:  
3     for e in arow:  
4         a.append(e)  
5 print(a)
```

```
[1, 2, 4, 5, 7, 8]
```

```
In [153]: 1 a = [e for arow in matrix for e in arow]  
2 print(a)
```

```
[1, 2, 4, 5, 7, 8]
```

Filter odd numbers from a nested list

```
In [154]: 1 a = []  
2 for arow in matrix:  
3     for e in arow:  
4         if (e % 2):  
5             a.append(e)  
6 print(a)
```

```
[1, 5, 7]
```

```
In [155]: 1 a = [e for arow in matrix for e in arow if (e % 2)]  
2 print(a)
```

```
[1, 5, 7]
```

Cartesian Product:

If A and B are sets, then the Cartesian product is the set of pairs(a,b) where 'a' is in A and 'b' is in B.



red = [1,2,3,4,5]



blue = [6,7,8,9,10]

If we are selecting 1 ball from the **red box** and 1 from the **blue box**, what are all the possible outcomes I can get?

In [156]:

```

1 red = [1,2,3,4,5]
2 blue = [6,7,8,9,10]
3 f = []
4 k = 0
5 for r in red:
6     for b in blue:
7         k += 1
8         f.append((r,b))
9 print(f)
10 print("k =",k)
```

```

[(1, 6), (1, 7), (1, 8), (1, 9), (1, 10), (2, 6), (2, 7), (2, 8), (2, 9), (2,
10), (3, 6), (3, 7), (3, 8), (3, 9), (3, 10), (4, 6), (4, 7), (4, 8), (4, 9),
(4, 10), (5, 6), (5, 7), (5, 8), (5, 9), (5, 10)]
k = 25
```

In [157]:

```

1 red = [1,2,3,4,5]
2 blue = [6,7,8,9,10]
3 k = 0
4 f = [(r,b) for r in red for b in blue]
5 print(f)
6 print("k =",k)
```

```

[(1, 6), (1, 7), (1, 8), (1, 9), (1, 10), (2, 6), (2, 7), (2, 8), (2, 9), (2,
10), (3, 6), (3, 7), (3, 8), (3, 9), (3, 10), (4, 6), (4, 7), (4, 8), (4, 9),
(4, 10), (5, 6), (5, 7), (5, 8), (5, 9), (5, 10)]
k = 0
```

Problem 2: Matching Students

Class A (4 students)	Class B (4 students)	Expected Outcome:
A111: 70 marks	B121: 90 marks	(A111, 'B122', 57)
A112: 25 marks	B122: 13 marks	(A111, 'B123', 4)
A113: 11 marks	B123: 66 marks	(A111, 'B124', 15)
A114: 80 marks	B124: 55 marks	(A112, 'B122', 12)
		(A112, 'B123', 14)
		(A114, 'B122', 67)
		(A114, 'B123', 14)
		(A114, 'B124', 25)

You want to match the students in a competition so that Class A has the higher chance to win.
To do that, you want to show all combinations where students in
Class A has higher score than students in **Class B**.

In [158]:

```

1 # Conventional Approach
2 classA = [("A111",70),("A112",25),("A113",11),("A114",80)]
3 classB = [("B121",90),("B122",13),("B123",66),("B124",55)]
4 matchA = []
5 # First Loop each element in classA
6 for (idA, scoreA) in classA:
7     # Loop each element in classB for a given class A element
8     for (idB, scoreB) in classB:
9         # Calculate the difference
10        diff = scoreA - scoreB
11        # store the result only when diff>0
12        if (diff>0):
13            matchA.append((idA, idB, diff))
14
15 for e in enumerate( matchA):
16     print(e)

(0, ('A111', 'B122', 57))
(1, ('A111', 'B123', 4))
(2, ('A111', 'B124', 15))
(3, ('A112', 'B122', 12))
(4, ('A114', 'B122', 67))
(5, ('A114', 'B123', 14))
(6, ('A114', 'B124', 25))

```

```
In [159]: 1 # List comprehension approach
2 matchA = [(idA,idB,scoreA-scoreB) for (idA,scoreA) in classA for (idB,scor
3 for e in enumerate(matchA):
4     print(e)

(0, ('A111', 'B122', 57))
(1, ('A111', 'B123', 4))
(2, ('A111', 'B124', 15))
(3, ('A112', 'B122', 12))
(4, ('A114', 'B122', 67))
(5, ('A114', 'B123', 14))
(6, ('A114', 'B124', 25))
```

Dictionary

{ } is Dictionary and also set

[] is list

```
In [160]: 1 l = ["jag", "bob", "jag"]
2 print(type(l), id(l), l)
3
4 s = {"jag", "bob", "jag"}
5 print(type(s), id(s), s)
6
7 d = {"jag":20,
8       "bob":40,
9       "rob":3}
10 print(type(d), id(d), d)

<class 'list'> 2408420256320 ['jag', 'bob', 'jag']
<class 'set'> 2408419983872 {'bob', 'jag'}
<class 'dict'> 2408420194432 {'jag': 20, 'bob': 40, 'rob': 3}
```

```
In [161]: 1 for k,v in d.items():
2     print("key",k, "value",v)

key jag value 20
key bob value 40
key rob value 3
```

```
In [162]: 1 keys = []
2 for k,v in d.items():
3     keys.append(k)
4 print(keys)

['jag', 'bob', 'rob']
```

```
In [163]: 1 keys = [k for k,v in d.items()]
2 print(keys)
['jag', 'bob', 'rob']
```

```
In [164]: 1 values = []
2 for k,v in d.items():
3     values.append(v)
4 print(values)
[20, 40, 3]
```

```
In [165]: 1 values = [v for k,v in d.items()]
2 print(values)
[20, 40, 3]
```

```
In [ ]: 1
```

1.11 Heap

VASUDEVAMURTHY

Heap in Python

Copyright: Jagadeesh Vasudevamurthy

filename: heap.ipynb

```
In [1]: import heapq
```

```
In [2]: class Node:
    def __init__(self, a:'int', b:'int'):
        self._a = a
        self._b = b

    ##Override __lt__ in Python 3, __cmp__ only in Python 2
    def __lt__(self, rhs:'Node')->'bool':
        if (self._a < rhs._a): # Change to > for max heap
            return True
        return False
```

```
In [3]: class MinHeap:
    def __init__(self):
        self._q = []
    def insert(self, a:'list'):
        for e in a:
            n = Node(e, -e)
            heapq.heappush(self._q, n)

    def add(self, n:'Node'):
        heapq.heappush(self._q, n)

    def get_top(self)->'Node':
        return self._q[0]

    def get_top_and_remove(self)->'Node':
        n = heapq.heappop(self._q)
        return n

    def deleteAll(self):
        while (len(self._q)):
            n = heapq.heappop(self._q)
            print(n._a, n._b);
```

```
In [4]: def test_Heap():
    a = [5, 8, 2, 8]
    h = MinHeap()
    h.insert(a)
    print("After inserting an array", a)
    n = h.get_top()
```

```
print("HeapTop has",n._a,n._b)
n = h.get_top_and_remove()
print("Removed element is",n._a,n._b)
n = h.get_top()
print("Now HeapTop has",n._a,n._b)
x = 3
n = Node(3,100)
h.add(n)
n1 = h.get_top()
print("HeapTop has after adding 3, 100 is",n1._a,n1._b)
n = Node(23,10)
h.add(n)
n1 = h.get_top()
print("HeapTop has after adding 23,10 is",n1._a,n1._b)
h.deleteAll()
```

In [5]: `test_Heap()`

```
After inserting an array [5, 8, 2, 8]
HeapTop has 2 -2
Removed element is 2 -2
Now HeapTop has 5 -5
HeapTop has after adding 3, 100 is 3 100
HeapTop has after adding 23,10 is 3 100
3 100
5 -5
8 -8
8 -8
23 10
```

In []:

In []:

1.12. NAMING CONVENTION

1.12 Naming convention

1 Class Name: Must start with Capital
EX: class Bank();

2 Functions and Variable Names:
Function names should be lowercase, with words separated by underscores as necessary to improve readability
Example: num_work_done

3 _single_leading_underscore
EXAMPLE: _work_done
This convention is used for declaring private variables, functions, methods and classes in a module.
Anything with this convention are ignored
in from module import *.
However, of course, Python does not supports truly private, so we can not force somethings private ones and also can call it directly from other modules.
So sometimes we say it “weak internal use indicator”.

4 Double Trailing Underscore __init__:
Indicates special methods defined by Python language.

Figure 1.1: Naming convention

1.13 class Util

```
1 #####  
2 # Util.py  
3 # Author: Jagadeesh Vasudevamurthy  
4 # Copyright: Jagadeesh Vasudevamurthy 2020  
5 #####  
6 #####  
7 #####  
8 # NOTHING CAN BE CHANGED IN THIS FILE  
9 #####  
10 #####  
11 #####  
12 # All imports  
13 #####  
14 import sys # For getting Python Version  
15 import random  
16 import math  
17 from time import process_time  
18  
19 class Util():  
20     pass  
21  
22 #####  
23 # generate_random_number start to end INCLUDED  
24 # start to end INCLUDED  
25 #####  
26 def generate_a_random_number(self,start:int,end:int)->'int':  
27     v = random.randrange(start,end+1);  
28     return v  
29  
30 #####  
31 # generate_random_number GENERATES N random numbers between  
32 # start to end INCLUDED  
33 # if onlypositive is False, generates both pos and negative number  
34 # randrange(beg, end, step) :-  
35 # beginning number (included in generation),  
36 # last number (excluded in generation) a  
37 # nd step ( to skip numbers in range while selecting).  
38 #####  
39 def generate_random_number(self, N:int, onlypositive:bool, start:int,  
end:int)->'List of integer':  
40     a = []  
41     for i in range(N):  
42         v = self.generate_a_random_number(start,end);  
43         if (onlypositive == False):  
44             if ((i % 2) == 0): ##//Even. Half are positive, Half are negative  
45                 v = -v ;  
46             a.append(v)  
47     return a  
48
```

C:\Users\jag\OneDrive\vasu\work\py3\objects\py3\sumN\Util.py

```

49 ######
50 # swap
51 #####
52 def swap(self,a:'List of integer', i:'int', j:'int'):
53     t = a[i]
54     a[i] = a[j]
55     a[j] = t
56
57 #####
58 # generate shuffled number between 0 to n
59 # n-1 not encluded
60 #####
61 def generate_suffled_number_between_1_to_n(self, n:int)->'List of integer':
62     a = []
63     for i in range(n):
64         a.append(i)
65
66     for i in range(n):
67         j = self.generate_a_random_number(0,n-1);
68         k = self.generate_a_random_number(0,n-1);
69         self.swap(a,j,k)
70     return a
71
72 #####
73 # generate n numbers in ascending order from 0 to n-1
74 #####
75 def generate_n_numbers_inAscending_order(self, n:int)->'List of integer':
76     a = []
77     for i in range(n):
78         a.append(i)
79     return a
80
81 #####
82 # generate n numbers in descending order from n-1 to 0
83 #####
84 def generate_n_numbers_in_descending_order(self, n:int)->'List of integer':
85     a = []
86     for i in range(n-1,-1,-1):
87         a.append(i)
88     return a
89
90 #####
91 # generate n same k number
92 #####
93 def generate_n_same_k_number(self, n:int,k:'int')->'List of integer':
94     a = []
95     for i in range(n):
96         a.append(k)
97     return a

```

```
98
99 ######
100 # print_index(10)
101 # 0 1 2 3 4 5 6 7 8 9
102 #####
103 def print_index(self, n:int):
104     for i in range(n):
105         print("{:4d}".format(i),end="")
106     print()
107
108 #####
109 # a = [7,8,9, 23, 100]
110 # print_list(a)
111 # 7 8 9 23 100
112 #####
113 def print_list(self, a:'list'):
114     for i in range(len(a)):
115         print("{:4d}".format(a[i]),end="")
116     print()
117
118 #####
119 # a = [7,8,9, 1, 100]
120 # crash
121 #####
122 def assertAscending_range(self, a:'list', start:int, includingend:int):
123     t = a[start]
124     for i in range(start+1,includingend):
125         if (a[i] < t):
126             assert(False)
127
128 #####
129 # a = [7,8,9, 1, 100]
130 # crash
131 #####
132 def assertAscending(self, a:'list'):
133     if (len(a)):
134         self.assertAscending_range(a,0,len(a))
135
136 #####
137 # log to the next possible integer
138 #####
139 def log_upper_bound(self, n:int, b:int)->'int':
140     f = math.log(n,b)
141     c = math.ceil(f)
142     return c
143
144 #####
145 # log to the smallest possible integer
146 #####
```

C:\Users\jag\OneDrive\vasu\work\py3\objects\py3\sumN\Util.py

```

147     def log_lower_bound(self, n:'int', b:'int')->'int':
148         f = math.log(n,b)
149         c = math.floor(f)
150         return c
151
152     ######
153     # sqrt to the next possible integer
154     #####
155     def sqrt_upper_bound(self, n:'int')->'int':
156         f = math.sqrt(n)
157         c = math.ceil(f)
158         return c
159
160     ######
161     # sqrt to the smallest possible integer
162     #####
163     def sqrt_lower_bound(self, n:'int')->'int':
164         f = math.sqrt(n)
165         c = math.floor(f)
166         return c
167
168     ######
169     # TEST DRIVERS BELOW
170     #####
171     def _test_random(self,N:int, onlypositive:bool, start:int, end:int)->'None':
172         a = self.generate_random_number(N,onlypositive,start,end);
173         assert(len(a) == N)
174         self.print_index(N)
175         self.print_list(a)
176
177     def _test_bed(self):
178         self._test_random(10, True,30,100)
179         self._test_random(10, False,30,40)
180
181         n = 10
182         a = self.generate_suffled_number_between_1_to_n(n)
183         self.print_index(n)
184         self.print_list(a)
185
186         a = self.generate_n_numbers_in_descending_order(n)
187         self.print_index(n)
188         self.print_list(a)
189
190         a = self.generate_n_numbers_inAscending_order(n)
191         self.print_index(n)
192         self.print_list(a)
193
194         a = self.generate_n_same_k_number(n,7)
195         self.print_index(n)

```

C:\Users\jag\OneDrive\vasu\work\py3\objects\py3\sumN\Util.py

```
196     self.print_list(a)
197
198
199
200
201 #####
202 # main
203 #####
204 def main():
205     print(sys.version)
206     t = Util()
207     t._test_bed()
208
209
210 #####
211 # start up
212 #####
213 if (__name__ == '__main__'):
214     main()
```

5

VASUDEVAMURTHY

Chapter 2

Analysis of Algorithms

2.1 Introduction

2.2 Algorithm

Algorithm

Algorithm is an effective method for solving a problem expressed as a finite sequence of instructions

Mohammed al-Khowarizmi

Mohammad ibn Mūsā al-Khwārizmī



A stamp issued December 5, 1993 in the Soviet Union, commemorating al-Khwarizmi's (approximate) 1200th birthday.

Born	c. 780
Died	c. 850
Ethnicity	Persian
Known for	ON APPROXIMATELY mathematics

Algorithm LargestNumber

Input: A List L that has N numbers and $N \geq 1$
Output: The largest number in the list L.

```

largest = L[0]
for (i = 1 to N-1) {
    if (L[i] > largest)
        largest = item L[i]
}
return largest

```

1. Precise
2. Unambiguous
3. Mechanical
4. Efficient
5. Correct

1. Is it correct ?
2. How much time does it take, in terms of n
3. And can we do better?

Figure 2.1: Algorithm

2.2.1 Finding a number in an unsorted array



Figure 2.2: Finding a number in an unsorted array

2.2.2 Finding a fake coin



You are given 70 gold coins, one of which is a fake coin (the fake coin has lesser weight). You have a scale, shown in the picture. What is the smallest number of weighing you can do in order to find the fake coin?

Figure 2.3: Finding fake coin

2.3 First look at complexity

2.3. FIRST LOOK AT COMPLEXITY

Complexity simplified

Someone asks you how much a textbook cost?

You will say: less than 100 ≤ 100 $O(n)$ WORST
At the worst 100\$
 (you won't say less than 1 million dollars)

If I buy a used/marked textbook? ≥ 10
 At least 10\$ (Best is 10\$)

Let us say all textbooks are exactly 40\$ $= 40$ $\underline{\Omega(n)}$ BEST
 $\underline{\Theta(n)}$ EXACT

<https://www.freecodecamp.org/news/big-theta-and-asymptotic-notation-explained/>

- “The delivery will be there within your lifetime.” (big-O, upper-bound)
- “I can pay you at least one dollar.” (big-omega, lower bound)
- “The high today will be 25°C and the low will be 19°C.” (big-theta, narrow range)
- “It’s a kilometer walk to the beach.” (big-theta, exact)

```
for i in range(N): {           Worst case: n times
    print("UCSC")             Best case: n times
}
}                                This takes some constant time THETA(1)   SPACE: THETA(1)

def findmax(l:'list')->'int':
    largest = l[0]           Worst case: n times
    n = len(l)               Best case: n
    for i in range(1,n):
        if (l[i] > largest):
            largest = l[i]   This takes some constant time THETA(1)   Space complexity THETA(1)

```

$$1, \log(n), n, n \cdot \log n, n^2, n^3, n^6, \frac{n}{2}, \frac{n}{3}, n^n, n!$$

Same techniques for both time and space

Figure 2.4: Complexity in simple words

2.4 Constant time/space algorithms

O(1) Algorithm

<pre> 1 def const(n:'int')->'int': 2 i = 45 + n ; 4 steps 3 j = 100 * n; independent of n 4 k = i + j ; 4 variables 5 return k independent of n 6 7 n = 4 8 k = const(4) 9 print("n = ",n, "k = ", k) 10 11 n = 1000 12 k = const(4) 13 print("n = ",n, "k = ", k) </pre> <p>n = 4 k = 449 n = 1000 k = 449</p>	<pre> 1 def const(n:'int')->'int': 2 return ((45 + n) + (100 * n)) 1 step 3 4 n = 4 independent of n 5 k = const(4) 1 variable 6 print("n = ",n, "k = ", k) independent of n 7 8 n = 1000 9 k = const(4) 10 print("n = ",n, "k = ", k) </pre> <p>n = 4 k = 449 n = 1000 k = 449</p>
--	---

```

void swap(int coffee, int tea) {
    int temp = coffee ;
    coffee = tea ;
    tea = temp;
}

```

3 steps independent of objects

Figure 2.5: Constant time/space algorithms

2.5 Logarithmic complexity algorithms

2.5.1 Logarithm

Logarithm

How many time
I should multiply
1 by 2 to get 8?

How many time
I should divide
8 by 2 to get 1?

$$1 * 2 * 2 * 2 = 8$$

$$2^? = 8$$

exponent
 $((8/2)/2)/2$

$$\log_2(8) = ?$$

base

$$\log 100 == \log_{10} 100$$

Engineers

$$\log_e 7.389 == \ln(7.389) \approx 2$$

Math

$e = \text{Euler Number} = 2.71828$

$$\log_2 16 == \lg(16) = 4$$

CS

$$5^? = 625$$

$$\log_5(625) = ?$$

$$5 * 5 * 5 * 5 = 625$$

$$\log_5(625) = 4$$

$$\log_b n = \frac{\log_2 n}{\log_2 b}$$

$$2^{10} = 1024 \quad \log_2(1024) = 10$$

$$2^0 = 1 \quad \log_2 1 = 0$$

$$2^1 = 2 \quad \log_2 2 = 1$$

what is $\log_2 64 \quad 2^8 = 64$

$$\log_2 64 = 8$$

$$\log_b(xy) = \log_b(x) + \log_b(y).$$

$$\log_b(b^x) = x \log_b(b) = x.$$

$$b^{\log_b(y)} = y$$

Figure 2.6: Logarithm

Logarithm Table			
x	$\log_{10}x$	\log_2x	$\log_e x$
1	0.000000	0.000000	0.000000
2	0.301030	1.000000	0.693147
3	0.477121	1.584963	1.098612
4	0.602060	2.000000	1.386294
5	0.698970	2.321928	1.609438
6	0.778151	2.584963	1.791759
7	0.845098	2.807355	1.945910
8	0.903090	3.000000	2.079442
9	0.954243	3.169925	2.197225
10	1.000000	3.321928	2.302585
20	1.301030	4.321928	2.995732
30	1.477121	4.906891	3.401197
40	1.602060	5.321928	3.688879
50	1.698970	5.643856	3.912023
60	1.778151	5.906991	4.094345
70	1.845098	6.129283	4.248495
80	1.903090	6.321928	4.382027
90	1.954243	6.491853	4.499810
100	2.000000	6.643856	4.605170
200	2.301030	7.643856	5.298317
300	2.477121	8.228819	5.703782
400	2.602060	8.643856	5.991465
500	2.698970	8.965784	6.214608
600	2.778151	9.228819	6.396930
700	2.845098	9.451211	6.551080
800	2.903090	9.643856	6.684612
900	2.954243	9.813781	6.802395
1000	3.000000	9.965784	6.907755
10000	4.000000	13.287712	9.210340

Figure 2.7: Logarithm table

$$\log_b n = \frac{\log_2 n}{\log 2}$$

Independent of n
 $O(\log n)$

Base does not
 matter

increases by
 constant amount
 independent of n

2.5.2 Guess a number

log n Behavior

A bartender offers 10000\$ bet. If you win, you get 10000\$ or you should pay him 10000\$

You choose a number between 1 to 1,000,000. Bartender will tell that number in 20 guesses.

After each guess, you should tell bar attender:

1. TOO HIGH
2. TOO LOW
3. YOU(bar attender) are right.

If bartender cannot get your choosen number in 20 guesses, you will win 10000\$. Other wise, you have to pay 10000\$ to the bartender.

Will you take that bet ?

How many MAXIMUM guesses are required to answer a number between 1- 10 ?

$$4 = \log_2 10$$

In general $\log_2 N$ guesses

$$\log_2 1000000 = 19.93 = 20 \text{ guesses}$$
$$2^{20} = 1048576$$

Figure 2.8: log n algorithm

log n Behavior

How many MAXIMUM guesses are required to answer a number between 1- 10 ?

$$4 = \log_2 10$$

In general $\log_2 N$ guesses

Number to be guessed 1

- 1: l= 1 r = 10 m = 5
- 2: l= 1 r = 4 m = 2
- 3: l= 1 r = 1 m = 1

Number to be guessed 2

- 1: l= 1 r = 10 m = 5
- 2: l= 1 r = 4 m = 2

2

Number to be guessed 3

- 1: l= 1 r = 10 m = 5
- 2: l= 1 r = 4 m = 2
- 3: l= 3 r = 4 m = 3

Number to be guessed 4

- 1: l= 1 r = 10 m = 5
- 2: l= 1 r = 4 m = 2
- 3: l= 3 r = 4 m = 3
- 4: l= 4 r = 4 m = 4

Number to be guessed 5

- 1: l= 1 r = 10 m = 5

1

Number to be guessed 6

- 1: l= 1 r = 10 m = 5
- 2: l= 6 r = 10 m = 8
- 3: l= 6 r = 7 m = 6

Number to be guessed 7

- 1: l= 1 r = 10 m = 5
- 2: l= 6 r = 10 m = 8
- 3: l= 6 r = 7 m = 6
- 4: l= 7 r = 7 m = 7

Number to be guessed 8

- 1: l= 1 r = 10 m = 5
- 2: l= 6 r = 10 m = 8

Number to be guessed 9

- 1: l= 1 r = 10 m = 5
- 2: l= 6 r = 10 m = 8
- 3: l= 9 r = 10 m = 9

3

Number to be guessed 10

- 1: l= 1 r = 10 m = 5
- 2: l= 6 r = 10 m = 8
- 3: l= 9 r = 10 m = 9
- 4: l= 10 r = 10 m = 10

4

Figure 2.9: Guessing a number between 1 to 10

Guess a number between l to h

```

1 def guess(g:'int',l:'int',h:'int')->'list of pos, steps':
2     max = int(round(math.log((h - 1),2)))
3     print("is ", g, " there between ",l, "and",h, " Both inclusive. Max steps should be ")
4     a = [-1,0]
5     while (l <= h):
6         a[1] = a[1] + 1;
7         m = (h - 1)//2 + 1 ;
8         print("step = ", a[1], "l = ", l, "h = ", h, "m = ",m)
9         if (g == m):|
10             a[0] = m
11             assert(a[1] <= max)|
12             return a
13         if (m < g):
14             l = m + 1
15         else:
16             h = m - 1
17     return a
18
19 def guesstop(n:'int',l:'int',h:'int'):
20     a = guess(n,l,h)
21     if (a[0] == -1):
22         print(n, "is not there after doing ", a[1], " steps of checking")
23     else:
24         print(n, "Found after doing ", a[1], " steps of checking")
25
1 g = 35
2 l = 1
3 h = 100
4 guesstop(g,l,h)
1 g = 101
2 l = 1
3 h = 100
4 guesstop(g,l,h)

```

Figure 2.10: Python code for guessing a number between l to r

2.5.3 Some more Logarithmic complexity algorithms examples

O(log n) Algorithms

```
def Quiz1(n:'int')->'int':  
    k = 0  
    i = 1  
    while (i < n):  
        k = k + 1  
        i = i * 2  
    return k
```

```
def Quiz2(n:'int')->'int':  
    k = 0  
    while (n > 1):  
        k = k + 1  
        n = n // 2  
    return k
```

Figure 2.11: log n algorithms

2.5.4 Harmonic series

2.6. LINEAR $O(N)$ ALGORITHMS

Harmonic series

In mathematics, the **harmonic series** is the [divergent infinite series](#):

$$\sum_{n=1}^{\infty} \frac{1}{n} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \dots$$

$$H_n \approx \ln n + \gamma + \frac{1}{2n} - \frac{1}{12n^2}$$

is quite good, where $\gamma \approx 0.5772156649$ is the [Euler-Mascheroni constant](#).

$$h(n) = \log_e n + 0.57721566$$

$$\Theta(\log n)$$

Figure 2.12: Harmonic series

2.6 Linear $O(n)$ algorithms

O(n) algorithms

```
int find1(int [] a, int tofind) {  
    int n = a.length ;  
    int j = 0 ;  
    while (j < n) {  
        if (a[j] == tofind) {  
            return j;  
        }else {  
            j++ ;  
        }  
    }  
    return -1;  
}
```

**2*n
compare**

```
int find2(int [] a, int tofind) {  
    int n = a.length ;  
    assert(a[n- 1] == tofind) ;  
    int j = 0 ;  
    while (1) {  
        if (a[j] == tofind) {  
            if (j == n- 1) {  
                return -1 ;  
            }  
            return j;  
        }else {  
            j++ ;  
        }  
    }  
}
```

**n+1
compare**

Figure 2.13: O(n) algorithms

2.7 $O(n \log n)$ algorithms

$O(n \log_2 n)$ algorithms

```
def Quiz3(n: 'int')->'int':
    k = 0
    for i in range(0,n):
        j = 2
        while (j <= n):
            k = k + 1
            j = j * 2
    return k
```

Figure 2.14: $O(n \log n)$ algorithms

Difference between n and $n \log n$ algorithm

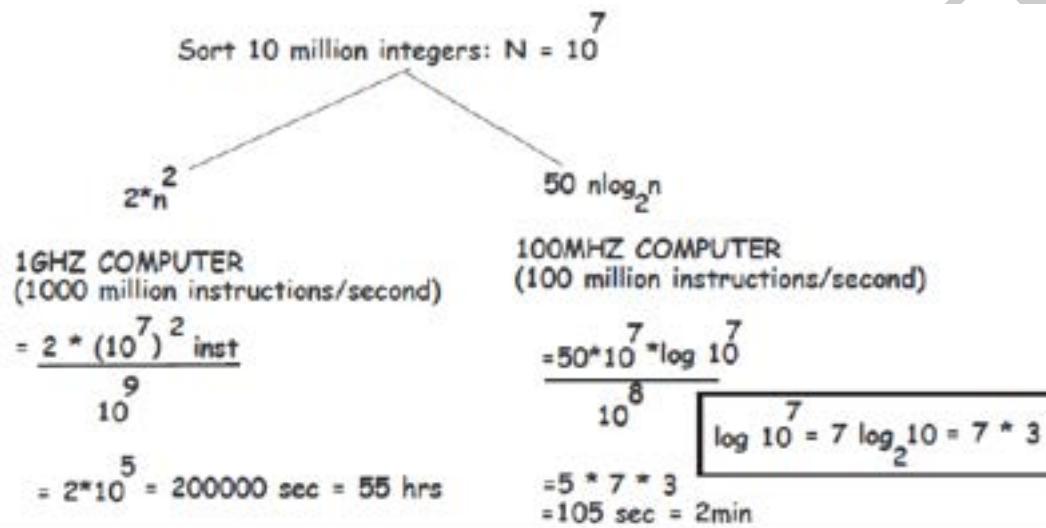


Figure 2.15: CPU time difference between n^2 and $n \log_2 n$ algorithms

2.8 Quadratic algorithms

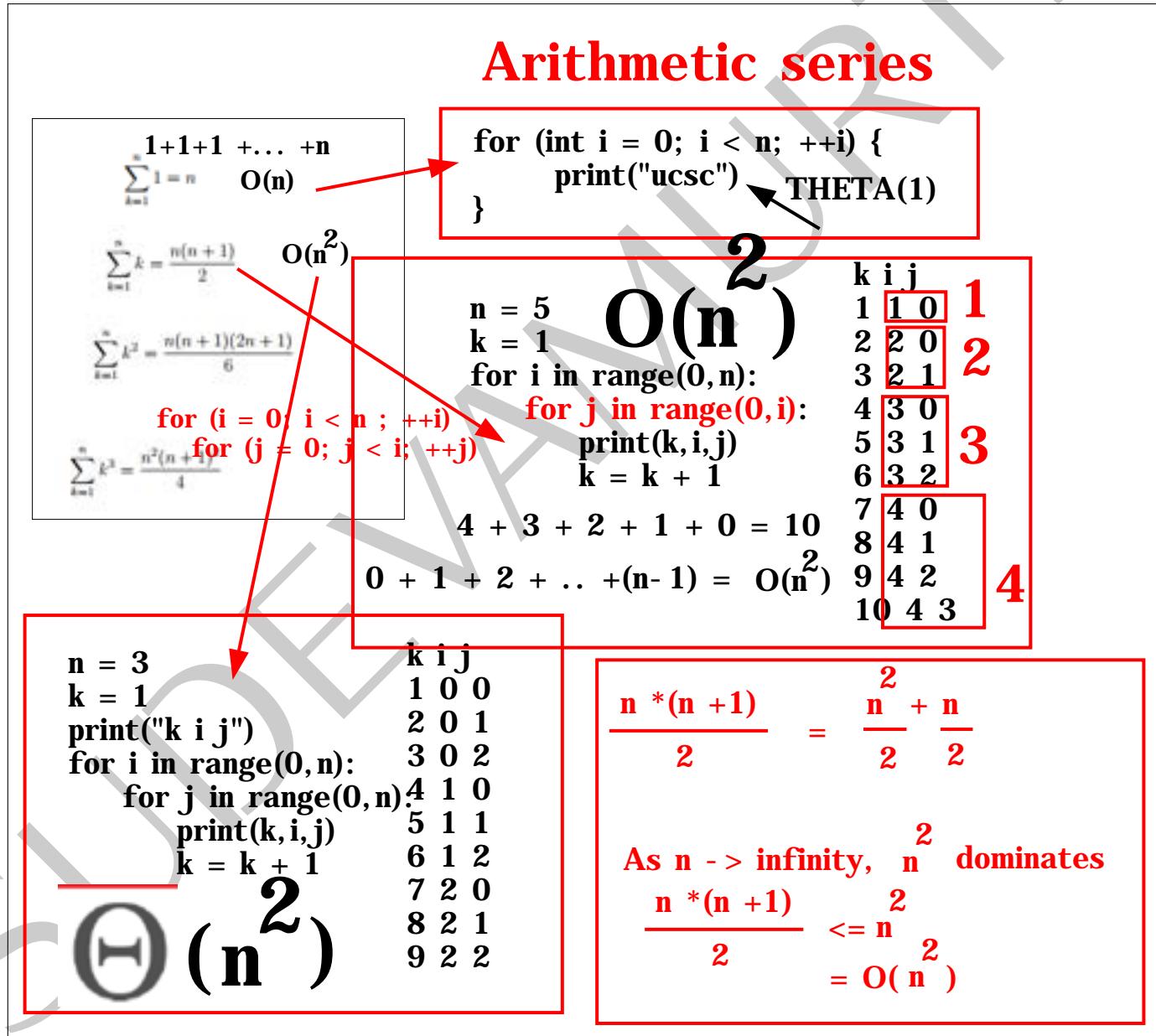


Figure 2.16: Arithmetic series

2.8.1 Quadratic algorithms examples

Quadratic algorithms

```
void doubleLoop(int n) {
    int k = 0;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            ++k;
        }
    }
    //What is n and k
}
```

```
void doubleLoopI(int n) {
    int k = 0;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < i; ++j) {
            ++k;
        }
    }
    //What is n and k
}
```

```
void doubleLoopC(int n, int C) {
    int k = 0;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < C; ++j) {
            ++k;
        }
    }
    //What is n and k
}
```

```
void tripleLoop(int n) {
    int k = 0;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            for (int p = 0; p < n; ++p) {
                //What is i j and p
                ++k;
            }
        }
    }
    //What is n and k
}
```

Figure 2.17: Quadratic algorithms

2.9 Generating prime numbers

2.9. GENERATING PRIME NUMBERS

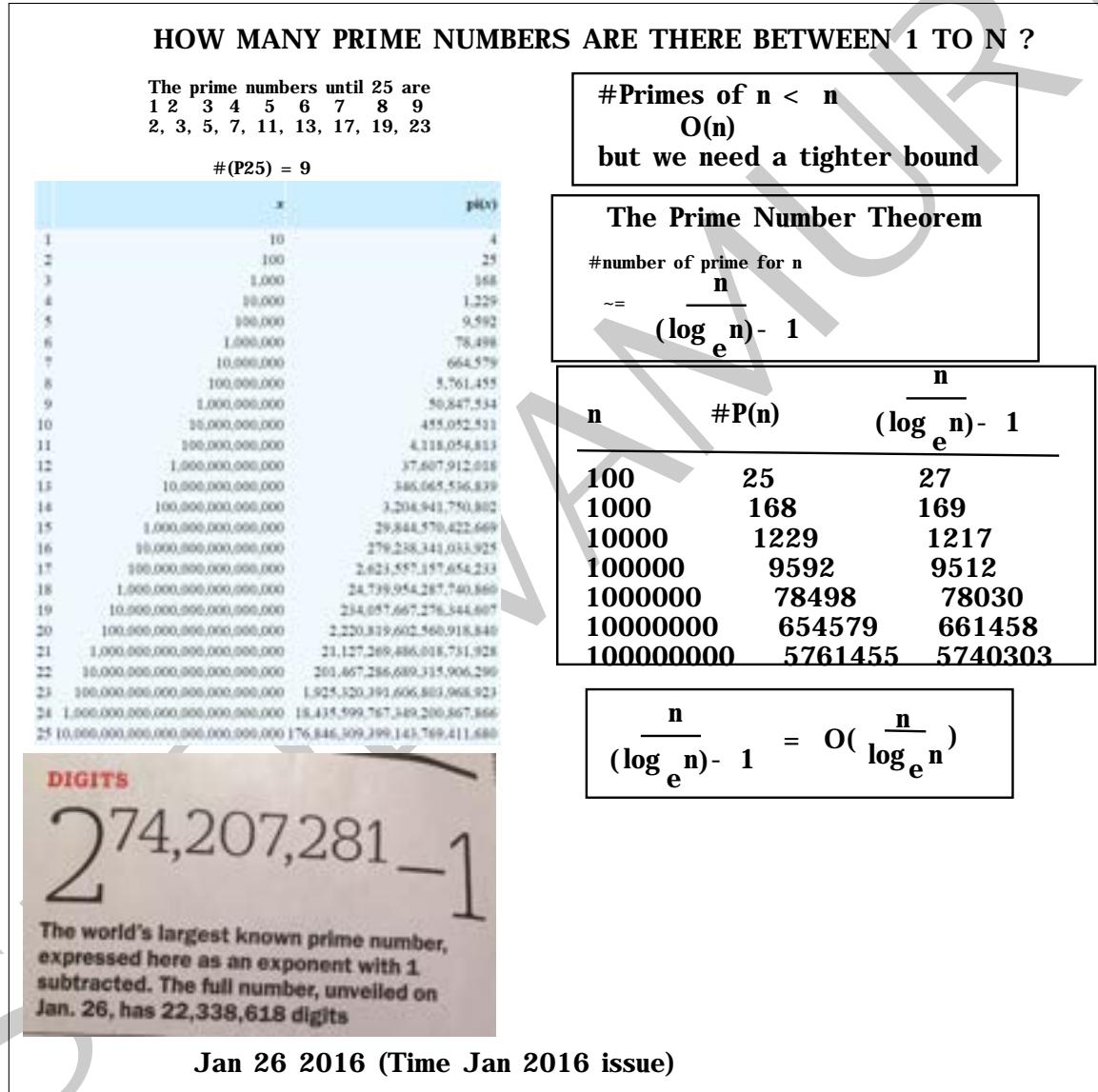


Figure 2.18: Computing numbers of prime numbers

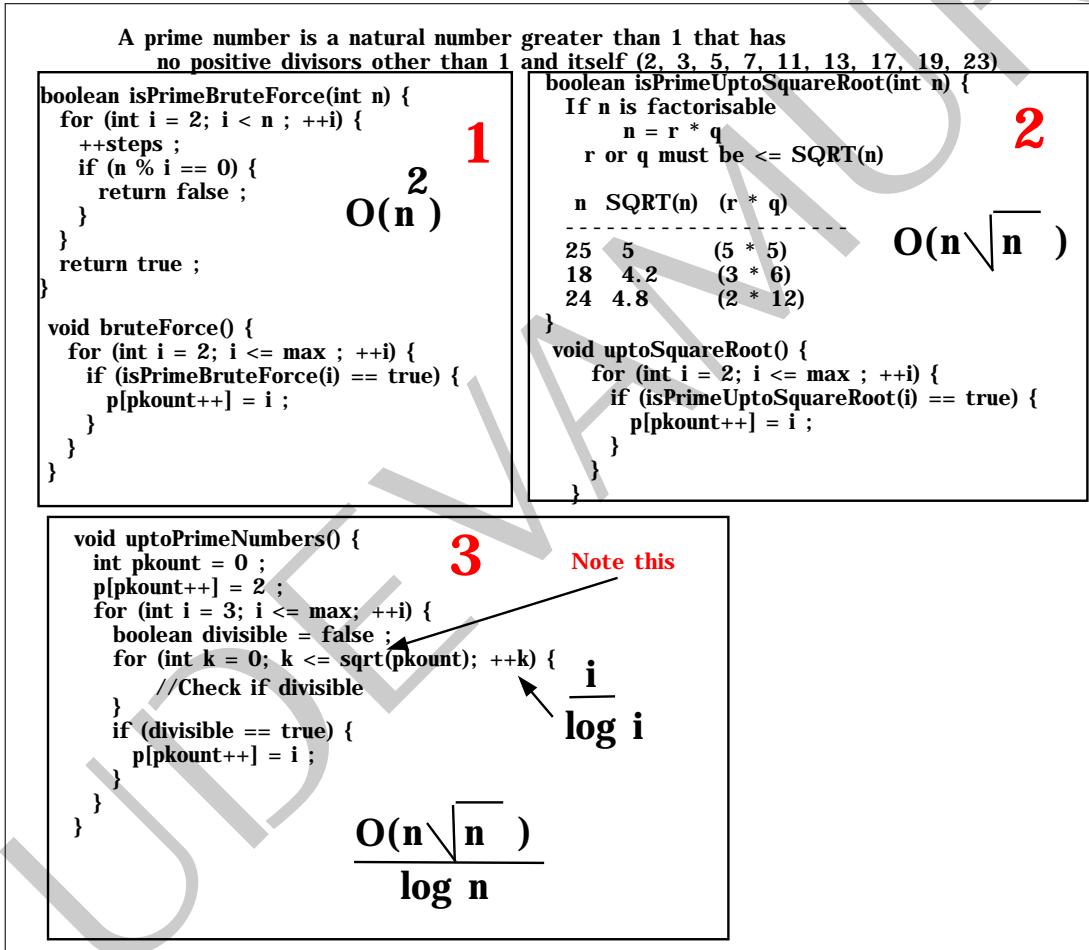


Figure 2.19: Three algorithms for generating prime numbers

2.9. GENERATING PRIME NUMBERS

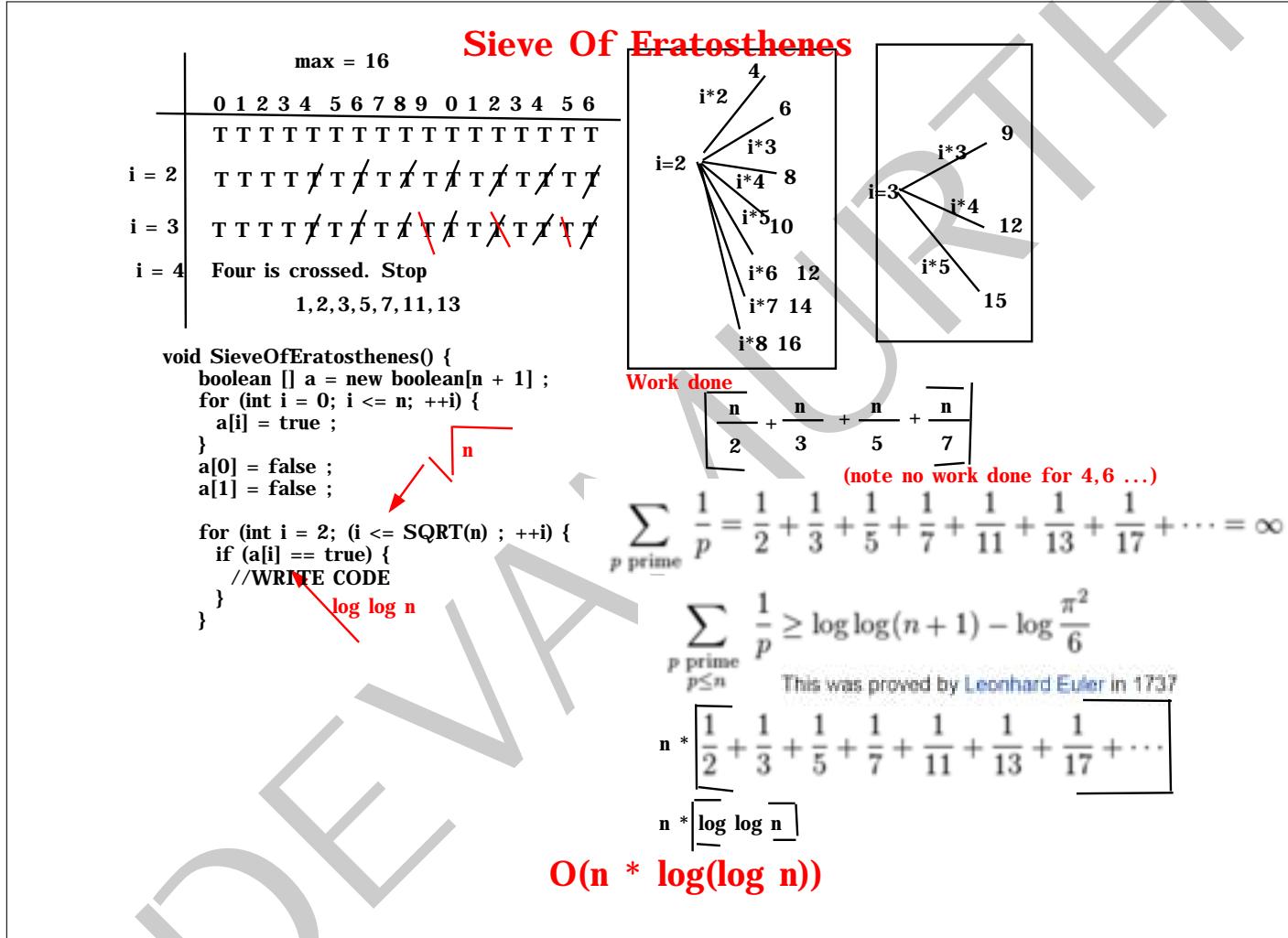


Figure 2.20: Sieve of Eratosthenes algorithm

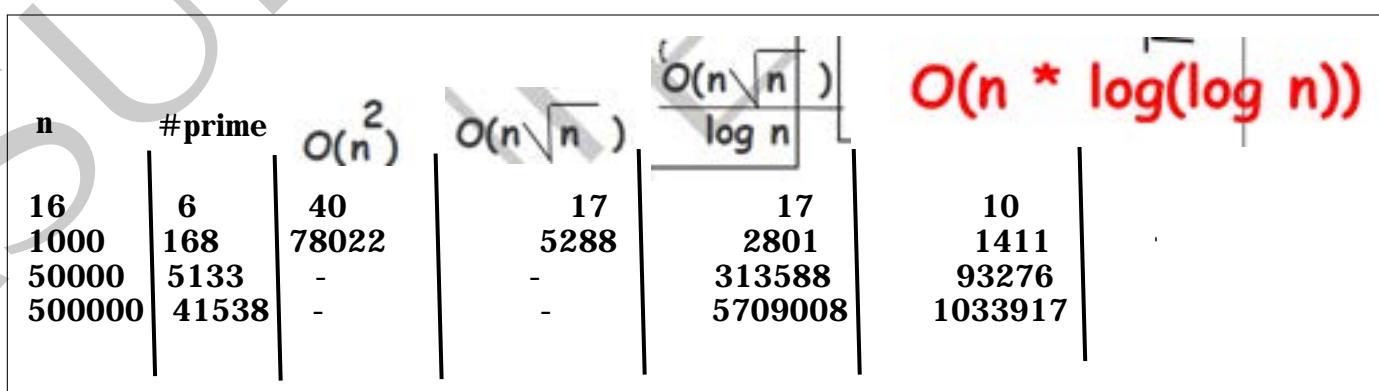


Figure 2.21: Number of steps with all the four methods

2.9.1 Expected output

```
Testing Prime.py Starts
----- 100 -----
----- n*SquareRoot(n) Method-----
100 has 25 Prime. Took 187 steps to compute
Total CPU time in sec = 0.0
----- uptoPrime Method-----
100 has 25 Prime. Took 132 steps to compute
Total CPU time in sec = 0.0
----- n*log(n) Method-----
100 has 25 Prime. Took 306 steps to compute
Total CPU time in sec = 0.0
----- 1000 -----
----- n*SquareRoot(n) Method-----
1000 has 168 Prime. Took 4789 steps to compute
Total CPU time in sec = 0.0
----- uptoPrime Method-----
1000 has 168 Prime. Took 2302 steps to compute
Total CPU time in sec = 0.0
----- n*log(n) Method-----
1000 has 168 Prime. Took 3413 steps to compute
Total CPU time in sec = 0.0
----- 10000 -----
----- n*SquareRoot(n) Method-----
10000 has 1229 Prime. Took 112528 steps to compute
Total CPU time in sec = 0.015625
----- uptoPrime Method-----
10000 has 1229 Prime. Took 38754 steps to compute
Total CPU time in sec = 0.015625
----- n*log(n) Method-----
10000 has 1229 Prime. Took 36983 steps to compute
Total CPU time in sec = 0.015625
----- 100000 -----
----- n*SquareRoot(n) Method-----
100000 has 9592 Prime. Took 2695695 steps to compute
Total CPU time in sec = 0.546875
----- uptoPrime Method-----
100000 has 9592 Prime. Took 694437 steps to compute
Total CPU time in sec = 0.1875
----- n*log(n) Method-----
```

2.10. EXPONENTIAL ALGORITHMS

```
100000 has 9592 Prime. Took 393080 steps to compute
Total CPU time in sec = 0.078125
----- 1000000 cannot compute using n^2 method -----
----- upto prime Method-----
1000000 has 78498 Prime. Took 13427403 steps to compute
Total CPU time in sec = 3.34375
----- n*log(n) Method-----
1000000 has 78498 Prime. Took 4122050 steps to compute
Total CPU time in sec = 0.921875
----- 10000000 cannot compute using n^2 method -----
----- upto prime Method-----
10000000 has 664579 Prime. Took 281144939 steps to compute
Total CPU time in sec = 93.609375
----- n*log(n) Method-----
10000000 has 664579 Prime. Took 42850053 steps to compute
Total CPU time in sec = 12.234375
----- 100000000 cannot compute using n^2 method -----
----- upto prime Method-----
100000000 has 5761455 Prime. Took 6270928471 steps to compute
Total CPU time in sec = 2077.84375
----- n*log(n) Method-----
100000000 has 5761455 Prime. Took 442570206 steps to compute
Total CPU time in sec = 135.03125
ALL TESTS PASSED
Testing Prime.py ENDS
```

2.10 Exponential algorithms

2.10.1 Geometric series

Geometric series

$$a + ar + ar^2 + \dots + ar^{(n-1)}$$

$$\sum_{k=0}^{n-1} (ar^k) = a \left(\frac{1-r^n}{1-r} \right)$$

0 to (n-1) is n

a is the first term
r is the "common ratio" between terms
n is the number of terms

0 to (n-1) is n

$$\sum_{i=0}^{n-1} 2^i = 2^0 + 2^1 + 2^2 + \dots + 2^{n-1} = 2^n - 1$$

a = 1 (first term)
r = 2 (doubles each time)
total n

Geometric series that converges absolutely

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \dots = \sum_{n=1}^{\infty} \left(\frac{1}{2}\right)^n = \frac{\frac{1}{2}}{1-\frac{1}{2}} = 1.$$

$$1 + 1/2 + 1/4 + 1/8 + 1/16 + \dots = 2 \quad \Theta(1)$$

a = 1/2 (first term)
r = 1/2 (reduces by half in each round)

$$1/2 * \left(\frac{1}{1/2}\right)^n = 1 \quad \left(\frac{1}{2}\right) \xrightarrow{n \rightarrow \infty} 0$$

Example: Grains of Rice on a Chess Board

On the page [Binary Data](#), we give an example of grains of rice on a chess board. The question is asked:

When we place rice on a chess board:

- 1 grain on the first square,
- 2 grains on the second square,
- 4 grains on the third and so on,
- ...

... doubling the grains of rice on each square ...

... how many grains of rice in total?



Figure 2.22: Geometric series

2.10. EXPONENTIAL ALGORITHMS

2.10.2 Rice on a chess board

$O(2^n)$ algorithms Rice on chess board



This tale, be it factual or not, is often used by mathematicians to explain the concept of exponential growth. It is difficult to fathom that by using this simple formula the accumulative amount is **18,446,744,073,709,551,615** grains of rice, and that in only 64 steps.

How much is that?

1,000,000	Million
1,000,000,000	Billion
1,000,000,000,000	Trillion
1,000,000,000,000,000	Quadrillion
1,000,000,000,000,000,000	Quintillion
1,000,000,000,000,000,000,000	Sextillion

Figure 2.23: Rice on chess board

2.10.3 Printing a truth table

$O(2^n)$ algorithms

Write a program that prints truth table of n inputs

For $n = 3$

000 TEST your program for $n = 4, 8$, and 10

001

010

011 WILL this world exists for $n = 64$?

100 Why not?

101

110

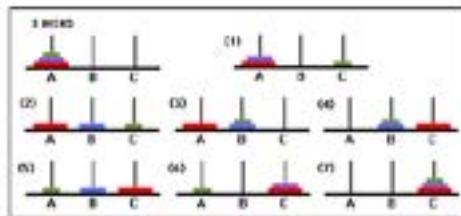
111

Figure 2.24: Printing a truth table

2.10.4 Tower of Hanoi

$O(2^n)$ algorithms Tower of Hanoi

WHEN WORLD WILL COLLAPSE?



TOWER OF BRAHMA has 64 disks

To complete you require $2^{64} - 1$ moves

$18,446,744,073,709,551,615$ moves

If the priests worked day and night,
making one move every second it
would take slightly more than 580
billion years to accomplish the job!

Figure 2.25: Tower of Hanoi

2.11. EXECUTION TIME FOR ALGORITHMS WITH GIVEN TIME COMPLEXITIES

2.11 Execution time for algorithms with given time complexities

Assume n= 1 takes .00001 second							Let us purchase 1000 times faster laptop			
Size n							Size of Largest Problem Instance Solvable in 1 hour			
Time complexity function	10	20	30	40	50	60				
n	.00001 second	.00002 second	.00003 second	.00004 second	.00005 second	.00006 second				
n^2	.0001 second	.0004 second	.0009 second	.0016 second	.0025 second	.0036 second				
n^3	.001 second	.008 second	.027 second	.064 second	.125 second	.216 second				
n^4	1 second	3.2 seconds	24.3 seconds	1.7 minutes	5.2 minutes	12.0 minutes				
2^n	.001 second	1.0 second	17.9 minutes	12.7 days	35.7 years	366 centuries				
3^n	.09 second	58 minutes	6.5 years	3831 centuries	2×10^6 centuries	1.3×10^7 centuries				

Time complexity function	With present computer	With computer 100 times faster	With computer 1000 times faster
n	N_1	$100 N_1$	$1000 N_1$
n^2	N_2	$10 N_2$	$11.6 N_2$
n^3	N_3	$4.54 N_3$	$10 N_3$
n^4	N_4	$2.5 N_4$	$3.98 N_4$
2^n	N_5	$N_5 + 0.04$	$N_5 + 0.97$
3^n	N_6	$N_6 + 6.19$	$N_6 + 6.26$

Listed from slowest to fastest growth:

- 1
- $\log n$
- n
- $n \log n$
- n^2
- n^3
- 2^n
- $n!$

Figure 2.26: Execution time

Difference between n and $n \log n$ algorithm

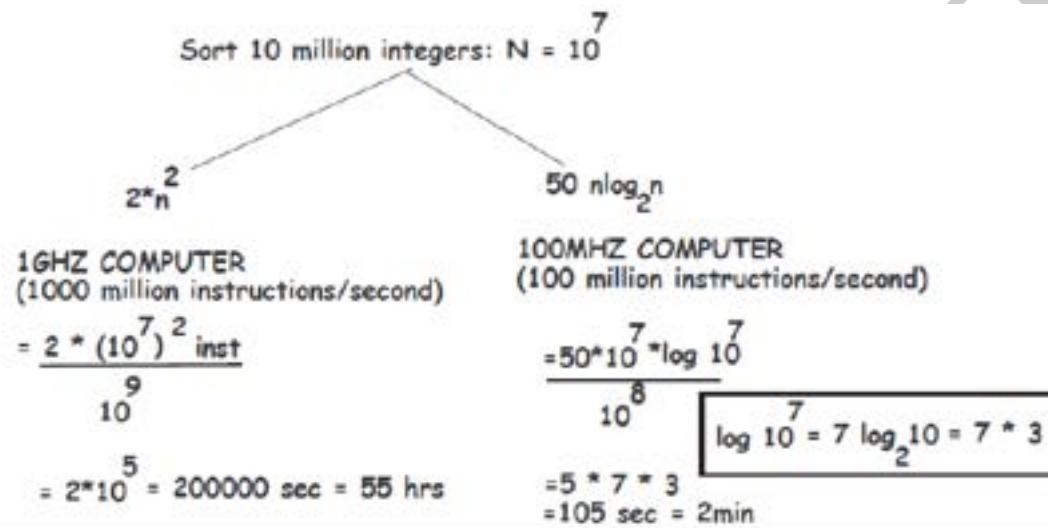


Figure 2.27: CPU time difference between n^2 and $n \log_2 n$ algorithms

2.12 Big O Notation

2.12.1 $O()$ definition

2.12. BIG O NOTATION

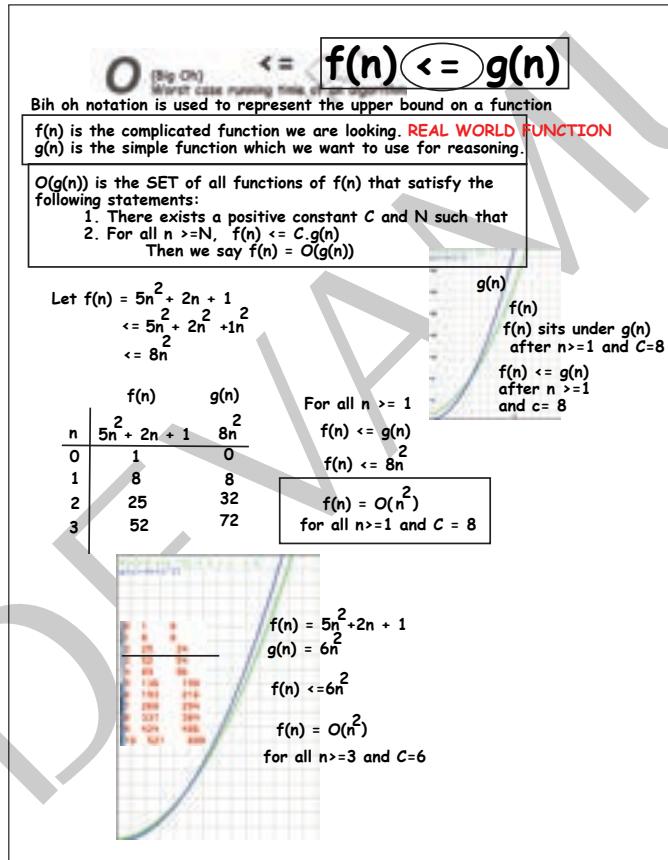


Figure 2.28: $O()$ definition

2.12.1.1 $O()$ Example 1

O
(Big Oh)
Worst case running time of an algorithm
 $\leq f(n) \leq g(n)$

$f(n)$ is the complicated function we are looking.
 $g(n)$ is the simple function which we want to use for reasoning.

$O(g(n))$ is the SET of all functions of $f(n)$ that satisfy the following statements:

1. There exists a positive constant C and N such that
2. For all $n \geq N$, $f(n) \leq C \cdot g(n)$
 Then we say $f(n) = O(g(n))$

Let $f(n) = 6 \cdot 2^{\frac{n}{n}} + n^2$

$$\begin{aligned} &<= 6 \cdot 2^n + n^2 \\ &\quad n \\ &<= 7 \cdot 2^n \end{aligned}$$

For all $n \geq 4$

$$\begin{aligned} f(n) &\leq g(n) \\ f(n) &\leq 7 \cdot 2^n \end{aligned}$$

$f(n) = O(2^{\frac{n}{n}})$
 for all $n \geq 4$ and $C = 7$

n	f(n)	g(n)
0	6	7
1	13	14
2	28	28
3	57	56
4	112	112
5	217	224
6	420	448
7	817	896
8	1600	1792
9	3153	3584

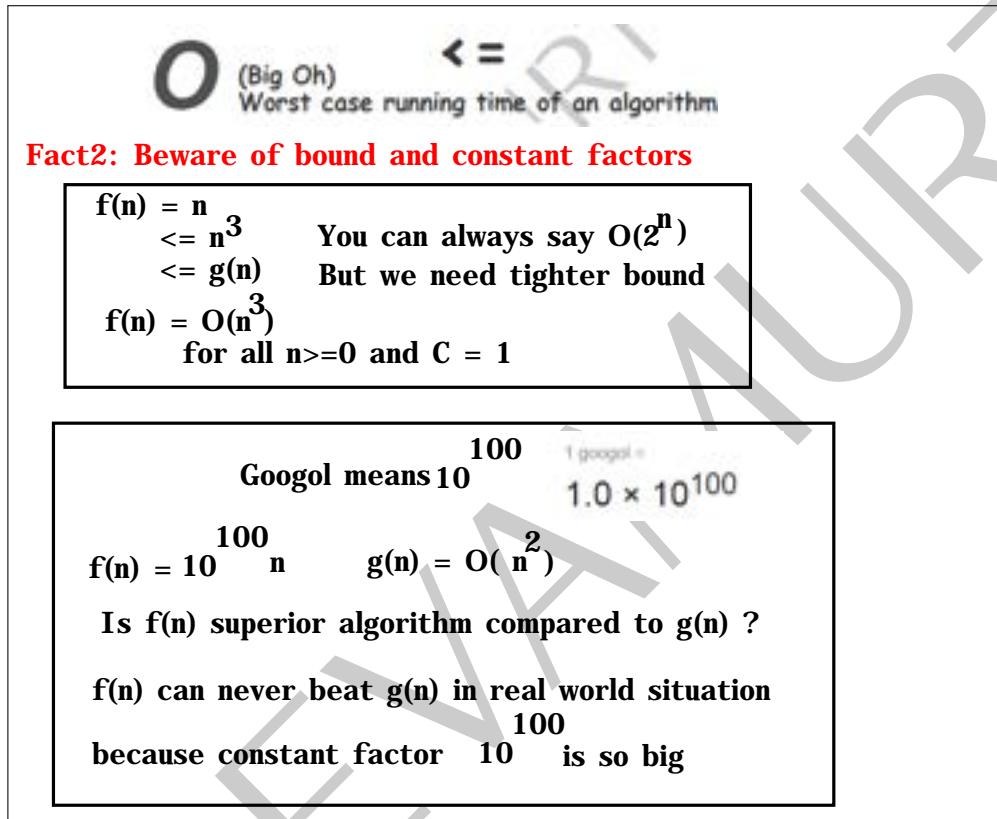
Figure 2.29: $O()$ example

2.12.2 $O()$ Fact 1

O	(Big Oh)	\leq	Worst case running time of an algorithm
Fact1: Big-Oh notation does not care about constant factors in terms of running time			
$f(n) = 1,000,000 n$	$\leq 1,000,000 n$	$\leq g(n)$	million n \leq million n. This is always true
$f(n) = O(n)$	for all $n \geq 0$ and $C=1,000,000$		
Complicated 1,000,000 n is written in simplest possible form O(n)			
Consider a program written by 3 programmers			
n	$n/2$	n	$100n$
100	50	100	100000
200	100	200	200000
ratio	2	2	2
			For all the programs, as n is doubled from 100 to 200, the growth rate is constant factor 2.
			Hence all the 3 algorithms are in $O(n)$
n	$100n$	n^2	
1	100	1	n^2 algorithm is better than n algorithm until $n = 100$
5	500	25	
10	1000	100	
20	2000	400	After that n^2 algorithm doubles. It is not a constant factor like above
50	5000	2500	
80	8000	6400	
100	10000	10000	BIG-OH is an UPPER BOUND ONLY
150	15000	22500	You can say how FAST your algorithm is,
200	20000	40000	You cannot say how SLOW is your algorithm
1000	100000	1000000	$\frac{n^2}{100n} = \frac{n}{100}$ worst depends on n

Figure 2.30: $O()$ fact 1

2.12.3 $O()$ Fact 2



O (Big Oh) \leq Worst case running time of an algorithm

Fact2: Beware of bound and constant factors

$$f(n) = n^3 \leq n^3 \quad \text{You can always say } O(2^n)$$

$$\leq g(n) \quad \text{But we need tighter bound}$$

$$f(n) = O(n^3) \quad \text{for all } n >= 0 \text{ and } C = 1$$

Googol means 10^{100}

$f(n) = 10^{100} n \quad g(n) = O(n^2)$

Is $f(n)$ superior algorithm compared to $g(n)$?

$f(n)$ can never beat $g(n)$ in real world situation
because constant factor 10^{100} is so big

Figure 2.31: $O()$ fact 2

2.12.4 $O()$ Fact 3

2.12. BIG O NOTATION

Fact3: Big- Oh notation is used to pick dominating terms

$$f(n) = n^3 + n^2 + 100n + 2000 ;$$

$$f(n) = O(n^3)$$

As $n \rightarrow \infty$, n^3 dominates, or grow much faster than n^2 and $100n$
 We are telling our algorithm is as fast n algorithm

Note that Big- Oh tells how fast is your algorithm.
 It cannot say how slow is your algorithm.

Theorem 1:

$$f(n) = a_0 + a_1 n + a_2 n^2 + a_3 n^3 + \dots + a_k n^k$$

$$\leq a_0 n^k + a_1 n^k + a_2 n^k + a_3 n^k + \dots + a_k n^k$$

$$\leq (a_0 + a_1 + a_2 + a_3 + \dots + a_k) n^k \quad \text{for } n \geq 1$$

$$\leq C n^k$$

$$f(n) = O(n^k) \quad \text{for } n \geq 1$$

$$\text{and } C = (a_0 + a_1 + a_2 + \dots + a_k)$$

Theorem 2:

Let us say have two phases in your algorithm.

$f_1(n) = O(g_1(n))$ Phase1

$f_2(n) = O(g_2(n))$ Phase2

then $f_1(n) + f_2(n) = O(\max(g_1(n), g_2(n)))$ NOTE IT IS NOT SUM

proof:

$f_1(n) = O(g_1(n))$ means $f_1(n) \leq C \cdot g_1(n)$ for all $n > n_1$

$f_2(n) = O(g_2(n))$ means $f_2(n) \leq D \cdot g_2(n)$ for all $n > n_2$

Let us choose $n_3 = \max(n_1, n_2)$;

$M = \max(C, D)$;

That means for $n > n_3$, $f_1 + f_2 \leq C \cdot g_1 + D \cdot g_2$

$\leq M \cdot g_1 + M \cdot g_2$

$\leq M(g_1 + g_2)$

$\leq M(2 \cdot \max(g_1, g_2))$

$\leq 2M(\max(g_1, g_2))$

Now $C = 2 \cdot M$

$n = n_3$

For $n > n_3$, and $C = 2 \cdot M$, $f_1(n) + f_2(n) = O(\max(g_1(n), g_2(n)))$

This also proves, if you have ' n ' phases in your algorithm
 the only phase that dominates is the one which takes maximum time.

Overall running time of an algorithm depend on the solving this
 BOTTLE NECK phase.

2.12.5 Sum and product rules

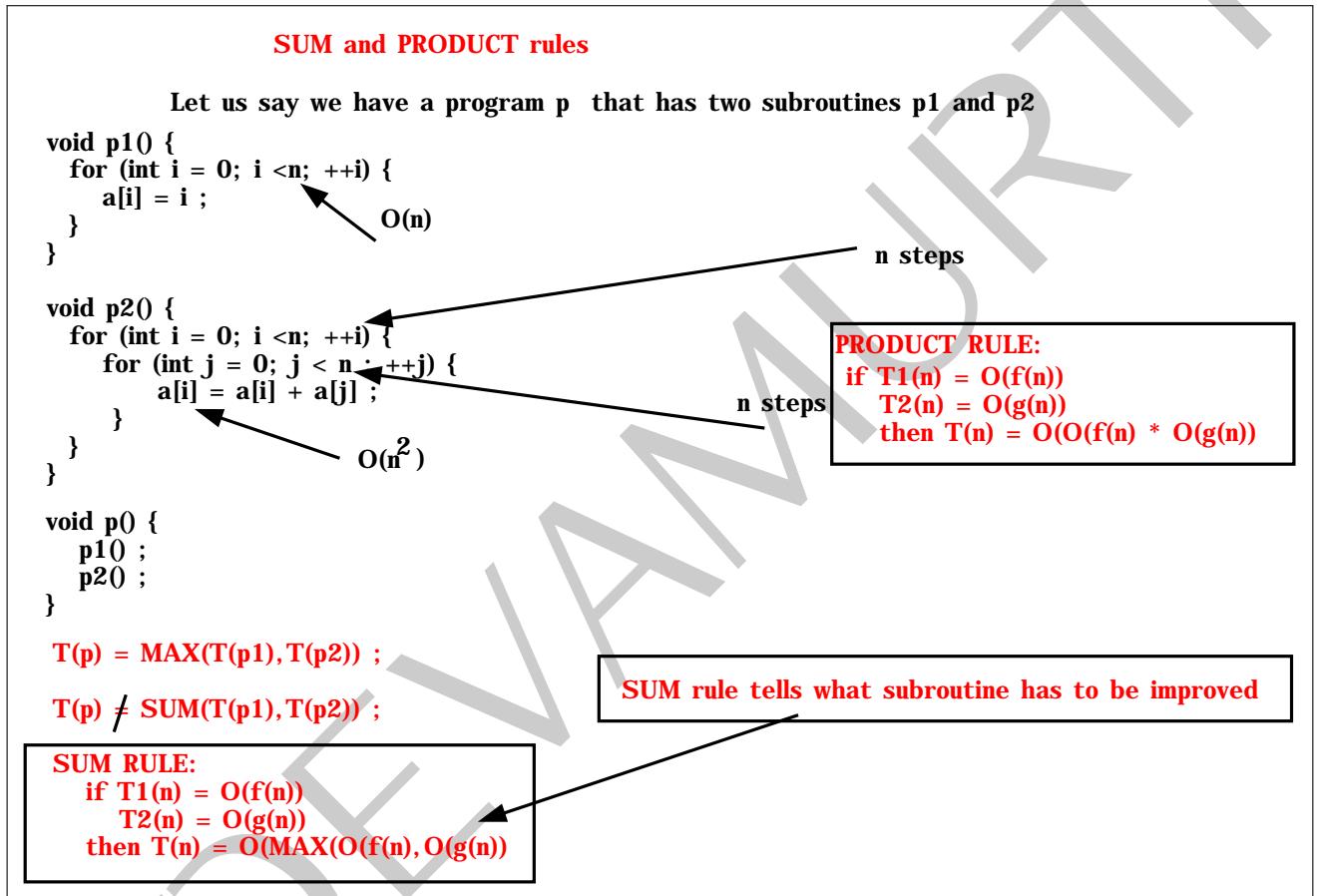


Figure 2.33: Sum and product rules

2.13 Big $\Omega()$ notation

2.13.1 $\Omega()$ definition

2.13. BIG $\Omega()$ NOTATION

Ω (Omega) \geq $f(n) \geq g(n)$
 Best case running time of an algorithm

f(n) is the complicated function we are looking.
g(n) is the simple function which we want to use for reasoning.

$\Omega(g(n))$ is the SET of all functions of $f(n)$ that satisfy the following statements:
 1. There exists a positive constant D and N such that
 2. For all $n \geq N$, $f(n) \geq D.g(n)$
 Then we say $f(n) = \Omega(g(n))$

Let $f(n) = 5n^2 + 2n + 1$
 $\geq 5n^2$

n	$f(n) = 5n^2 + 2n + 1$	$g(n) = 5n^2$
0	1	0
1	8	5
2	25	20
3	52	45

For all $n \geq 0$ $f(n) \geq g(n)$
 $f(n) \geq 5n^2$

$f(n) = \Omega(n^2)$
 for all $n \geq 0$ and $D = 5$

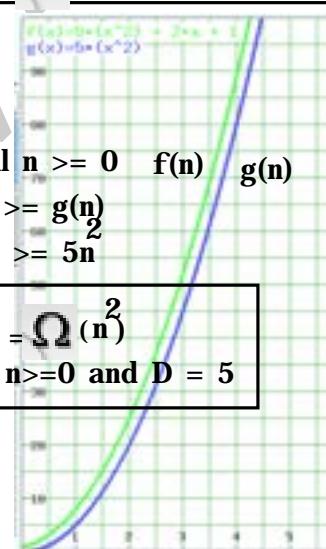


Figure 2.34: $\Omega()$ definition

2.14 Big $\Theta()$ notation

2.14.1 $\Theta()$ definition

Θ (Theta) Worst case = Best case = $f(n) \Theta g(n)$

$f(n)$ is the complicated function we are looking.
 $g(n)$ is the simple function which we want to use for reasoning.

$\Theta(g(n))$ is the SET of all functions of $f(n)$ that are both in $O(g(n))$ and $\Omega(g(n))$
 There exists a positive constants C and D and $n >= 0$ that satisfy the following statements:
 For all $n >= N$, $D.g(n) <= f(n) <= C.g(n)$
 Then we say $f(n) = \Theta(g(n))$

Let $f(n) = 5n^2 + 2n + 1$

$$5n^2 \leq 5n^2 + 2n + 1 \leq 8n^2$$

n	$8n^2$	$5n^2$	$f(n) = 5n^2 + 2n + 1$
0 0	0	0	1
1 8	8	5	9
2 32	32	25	29
3 72	72	52	57
4 128	128	80	101
5 200	200	125	150
6 288	288	193	204
7 392	392	260	283

For all $n >= 1$

$$8n^2 \geq 5n^2 + 2n + 1$$

For all $n >= 1$

$$5n^2 \leq 5n^2 + 2n + 1$$

$$5n^2 \leq 5n^2 + 2n + 1 \leq 8n^2$$

$$f(n) = \Theta(n^2)$$

for all $n >= 1, D=5, C=8$

Figure 2.35: $\Theta()$ definition

2.14. BIG Θ() NOTATION

Facts about Θ

An algorithm is $\Theta(g(n))$ if and only

1. if its worst-case running time is $O(g(n))$ and
2. its best-case running time is $\Omega(g(n))$.

For any two functions $f(n)$ and $g(n)$, we have
 $f(n) = \Theta(g(n))$
if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$

Example 1:

```
for (int i = 0; i < n; ++i) {  
    a[i] = i ;  
}
```

Example 2:
Is Binary search $\Theta(\log n)$ or $O(\log n)$?

It is NOT $\Theta(\log n)$ because you can always find the element in the first step, i.e. there is a lower bound $\Omega(1)$

Figure 2.36: Facts about $\Theta()$ notation

<https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation>

Linear search

Worst case: $c_1 \cdot n + c_2$;

As we've argued, the constant factor c_1 and the low-order term c_2 don't tell us about the rate of growth of the running time. What's significant is that the worst-case running time of linear search grows like the array size n . The notation we use for this running time is $\Theta(n)$. That's the Greek letter "theta," and we say "big-Theta of n " or just "Theta of n ."

Worst case running is $\Theta(n)$
Best case running is $\Theta(1)$
RUNNING TIME: $O(n)$

<https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation>

Binary search

only above. For example, although the worst-case running time of binary search is $\Theta(\lg n)$, it would be incorrect to say that binary search runs in $\Theta(\lg n)$ time in *all* cases. What if we find the target value upon the first guess? Then it runs in $\Theta(1)$ time. The running time of binary search is never worse than $\Theta(\lg n)$, but it's sometimes better. It would be convenient to have a form of asymptotic notation that means "the running time grows at most this much, but it could grow more slowly." We use "big-O" notation for just such occasions.

Binary search worst case: $\Theta(\lg n)$
Binary search best case: $\Theta(1)$
RUNNING TIME:
It is incorrect to say $\Theta(\lg n)$. So we say $O(\lg n)$

Printing an array

Worst case: $\Theta(n)$
Best case: $\Theta(n)$
Running time: $\Theta(n)$

Figure 2.37: Facts about $\Theta()$ notation

2.14. BIG Θ() NOTATION

What is Big Theta? When should I use Big Theta as opposed to big O?

<https://www.quora.com/What-is-Big-Theta-When-should-I-use-Big-Theta-as-opposed-to-big-O>

Formally, the only place you see bit-Theta is when the complexity is guaranteed and usually only in proofs. Adding two fixed-length numbers, for example, is $\Theta(1)$, no matter what. Printing or modifying every element of an array is $\Theta(n)$.

Informally, you'll generally call everything big-O, even if it isn't. It's technically valid, since big-O is the upper-bound and the upper-bound of a fixed function is obviously that function, but it can sound a little bit awkward.

<https://cs.stackexchange.com/questions/23068/how-do-o-and-%CE%A9-relate-to-worst-and-best-case>

Consider the following algorithm (or procedure, or piece of code, or whatever):

```
Contrive(n)
1. if n = 0 then do something Theta(n^3)
2. else if n is even then
3.   flip a coin
4.   if heads, do something Theta(n)
5.   else if tails, do something Theta(n^2)
6. else if n is odd then
7.   flip a coin
8.   if heads, do something Theta(n^4)
9.   else if tails, do something Theta(n^5)
```

What is the asymptotic behavior of this function?

In the best case (where n is even), the runtime is $\Omega(n)$ and $O(n^2)$, but not Θ of anything.

In the worst case (where n is odd), the runtime is $\Omega(n^4)$ and $O(n^5)$, but not Θ of anything.

In the case $n = 0$, the runtime is $\Theta(n^3)$.

This is a bit of a contrived example, but only for the purposes of clearly demonstrating the differences between the bound and the case. You could have the distinction become meaningful with completely deterministic procedures, if the activities you're performing don't have any known Θ bounds.

Figure 2.38: Facts about $\Theta()$ notation

Θ or O

<https://stackoverflow.com/questions/471199/what-is-the-difference-between-%ce%98n-and-on>

Big O means your algorithm will execute in no more steps than in given expression(n^2)

Big Omega means your algorithm will execute in no fewer steps than in the given expression(n^2)

When both condition are true for the same expression, you can use the big theta notation....

$\Theta(n^2)$

```
for (int i = 0; i < n; ++i)
    for (int j = 0; j < n; ++j)
        print(i,j)
```

$O(n)$

```
for (int i = 0; i < n; ++i)
    if (a[i] == k) //some k passed. K may there or not there
        break;
```

for i = 1 to n do
something in $O(1)$ that doesn't change n and i and isn't a jump

is $\Theta(n)$, $O(n^2)$, $\Omega(n^3)$, $\mathcal{O}(n^{1423424})$. big-oh is just an upper bound, which makes it easier to calculate because you don't have to find a tight bound.

The above loop is **only** `big-theta(O)` however.

What's the complexity of the [sieve of eratosthenes](#)? If you said `O(n log n)`, you wouldn't be wrong, but it wouldn't be the best answer either. If you said `big-theta(n log n)`, you would be wrong.

Figure 2.39: When to use Θ and O

2.15. PROBLEM SET

2.15 Problem set

Problem 2.15.1. Answer the questions below by hand and post the scanned document on Canvas.



Solve for only 70 coins

You are given 70 gold coins, one of which is a fake coin (the fake coin has lesser weight). You have a scale, shown in the picture. What is the smallest number of weighing you can do in order to find the fake coin?

1. Give an algorithm (Use figures/pseudo code) to solve the above program
2. Animate using above algorithm. how how will find fake coin which is in position 16
3. Animate using above algorithm. how how will find fake coin which is in position 22
4. Animate using above algorithm. how how will find fake coin which is in position 45
5. What is the minimum number of weighing you require? -----
6. What is the maximum number of weighing you require? -----

2.15. PROBLEM SET

Problem 2.15.2. Answer the questions below by hand and post the scanned document on Canvas.

Lighter or Heavier?



1. $n > 2$ identical coins
2. Exactly one coin is lighter or heavier(fake coin)
Remaining ($n-1$) coins have exact weight(Genuine)
3. Balance with no weights
4. You should find whether the fake coin is
Lighter or Heavier compared to Genuine coin

1. Give an algorithm (Use figures/pseudo code) to solve the above program
2. Animate using the above algorithm, how you will identify lighter or heavier for the following cases

1	0	1	2	3	4	5	6	7
	2	2	100	2	2	2	2	2

2	0	1	2	3	4
	2	2	2	2	100

3	0	1	2	3	4	5	6
	2	2	2	2	2	2	1

4	0	1	2	3
	2	1	2	2

Time complexity :
Space complexity:

2.15. PROBLEM SET

Problem 2.15.3. Find the complexity of each procedure by hand and post the scanned document on Canvas. All the questions are taken from interviews.

```

1  /**
2   * File Name: Complexity.java
3   *
4   * C:\work\java\objects\quiz\Complexity.java
5   *
6   * @author Jagadeesh Vasudevamurthy
7   *
8   */
9  public class Complexity{
10
11     private static void P(String s, int n, long k) {
12         System.out.println(s + " For input " + n + " work done is " + k ) ;
13     }
14
15     private static void p1(int n) {
16         long k = 0 ;
17         int x = 100 ;
18         int y = 89000 ;
19         ++k ;
20         int z = x + y * y + 8000 ;
21         P("p1",n,k);
22     }
23
24     private static void p2(int n) {
25         long k = 0 ;
26         int x = 100 ;
27         int y = 89000 ;
28
29         ++k;
30         int z = x + y * y + 8000 ;
31
32         for (int i = 0; i < n ; ++i) {
33             ++k ;
34             int w = i * x + y ;
35         }
36         P("p2",n,k);
37     }
38
39     private static void p3(int n) {
40         long k = 0 ;
41         int x = 100 ;
42         int y = 89000 ;
43
44         ++k;
45         int z = x + y * y + 8000 ;
46         for (int i = 0; i < n ; ++i) {
47             ++k ;
48             int w = i * x + y ;
49         }
50         for (int i = 0; i < n ; ++i) {
51             for (int j = 0; j < n ; ++j) {
52                 ++k ;
53                 int w = (i * x + y)*j ;
54             }
55         }
56         P("p3",n,k);
57     }
58
59     private static void p4(int n) {
60         long k = 0 ;
61         for (int i = 0; i < n; ++i) {
62             for (int j = 0; j < n/2; ++j) {
63                 ++k ;
64             }
65         }
66         P("p4",n,k);
67     }
68
69     private static void p5(int n) {

```

```

70 long k = 0 ;
71 for (int i = 0; i < n; ++i) {
72     for (int j = 0; j < i; ++j) {
73         for (int z = 0; z < 77; ++z) {
74             ++k ;
75         }
76     }
77 }
78 P("p5",n,k);
79 }
80
81 private static void p6(int n) {
82     long k = 0 ;
83     for (int i = 0; i < n; ++i) {
84         for (int j = 0; j < n*n; ++j) {
85             for (int z = 0; z < n*n*n; ++z) {
86                 ++k ;
87             }
88         }
89     }
90     P("p6",n,k);
91 }
92
93 private static void p7(int n) {
94     long k = 0 ;
95     for (int i = 0; i < n; ++i) {
96         for (i = 0; i < n*n; ++i) {
97             for (i = 0; i < n*n*n; ++i) {
98                 ++k ;
99             }
100        }
101    }
102    P("p7",n,k);
103 }
104
105 private static void p8(int n) {
106     long k = 0 ;
107     for (int i = 0; i < n; ++i) {
108         for (int j = 0; j < n*n; ++j) {
109             for (j = 0; j < n*n*n; ++j) {
110                 ++k ;
111             }
112         }
113     }
114     P("p8",n,k);
115 }
116
117 private static void p9(int n) {
118     long k = 0 ;
119     int t = n ;
120     while (n > 1) {
121         ++k;
122         n = n/2 ;
123     }
124     P("p9",t,k);
125 }
126
127 private static void p10(int n) {
128     long k = 0 ;
129     int t = n ;
130     while (n > 0) {
131         ++k;
132         n = n/3 ;
133     }
134     P("p10",t,k);
135 }
136
137 private static void p11(int n) {
138     long k = 0 ;

```

```

139     int t = n ;
140     while (n > 0) {
141         ++k;
142         n = n-1 ;
143     }
144     P("p11",t,k);
145 }
146
147 private static void p12(int n) {
148     long k = 0 ;
149     int t = n ;
150     while (n > 0) {
151         ++k;
152         n = n-5 ;
153     }
154     P("p12",t,k);
155 }
156
157 private static void interview1(int n) {
158     int s = 0 ;
159     for(int i=1;i<=n;i++) {
160         for(int j=1;j<=n;j+=i) {
161             for(int k=1;k<=n;k+=2) {
162                 ++s;
163             }
164         }
165     }
166     P("interview1",n,s);
167 }
168
169 private static void p13(int n) {
170     long k = 0 ;
171     int t = n ;
172     while (n >= 0) {
173         ++k;
174         n = n/2;
175         n = n-1;
176     }
177     P("p13",t,k);
178 }
179
180 private static void p14(int n) {
181     long k = 0 ;
182     int i = 1;
183     while (i < n) {
184         ++k;
185         i = i * 2 ;
186     }
187     P("p14",n,k);
188 }
189
190 private static void p15(int n) {
191     long k = 0 ;
192     int i = 1;
193     while (i < n) {
194         ++k;
195         i = i * 5 ;
196     }
197     P("p15",n,k);
198 }
199
200 private static void p16(int n) {
201     long k = 0 ;
202     int i = 1;
203     while (i < n) {
204         ++k;
205         i = i * 2 ;
206         ++k;
207         i = i + 1;

```

```

208     ++k;
209     i = i * 3;
210     ++k;
211     i = i - 1;
212 }
213 P("p16",n,k);
214 }
215
216 private static void p17(int n) {
217     long k = 0 ;
218     for (int i = 0; i < n; i=i+1) {
219         for (int j = 1; j <= n; j=j*2) {
220             for (int p = n-1; p > 1; p=p/2) {
221                 ++k ;
222             }
223         }
224     }
225     P("p17",n,k);
226 }
227
228 private static void p18(int n) {
229     long k = 0 ;
230     for (int i = 0; i < n; i=i+1) {
231         int j = 2 ;
232         while (j <= n) {
233             j = j * 2 ;
234             ++k ;
235         }
236     }
237     P("p18",n,k);
238 }
239
240 private static void p19(int n) {
241     long k = 0 ;
242     for (int i = 0; i < n; i=i+1) {
243         int j = 2 ;
244         while (j < n) {
245             j = j * j ;
246             ++k ;
247         }
248     }
249     P("p19",n,k);
250 }
251
252 private static void p20(int n) {
253     long k = 0 ;
254     int logntobase2 = (int)(Math.log(n)/Math.log(2));
255     //System.out.println("n = " + n) ;
256     //System.out.println("logntobase2 = " + logntobase2) ;
257     for (int i = 0; i < n; i=i+1) {
258         int j = 2 ;
259         while (j < logntobase2) {
260             j = j * 2 ;
261             ++k ;
262         }
263     }
264     P("p20",n,k);
265 }
266
267 private static void p21(int n) {
268     long k = 0 ;
269     int sqrt = (int) Math.sqrt((int)n) ;
270     //System.out.println("n = " + n) ;
271     //System.out.println("sqrt = " + sqrt) ;
272     for (int i = 0; i < n; i=i+1) { 182
273         int j = 2 ;
274         while (j < sqrt) {
275             j = j * j ;
276             ++k ;

```

```
277     }
278 }
279 P("p21",n,k);
280 }
281
282 private static long p22_r(int n,long k) {
283     ++k;
284     if (n <= 2) {
285         return k;
286     }else {
287         int x = (int)Math.sqrt((int)n);
288         return p22_r(x,k);
289     }
290 }
291
292 private static void p22(int n) {
293     long k = 0;
294     k = p22_r(n,k);
295     P("p22",n,k);
296 }
297
298 }
```

2.15. PROBLEM SET

Problem 2.15.4. LeetCode Problem 977: Square of a sorted array. Please refer to the section 20.5

Problem 2.15.5. LeetCode Problem 38: Count and say. Please refer to the section 20.2

Problem 2.15.6. Find duplicate elements in a Python list. Please refer to the section 20.7

Problem 2.15.7. LeetCode Problem 238: Product of Array Except Self. Please refer to the section 20.9

Chapter 3

Recursion

3.1 Introduction

3.2 Factorial of a number

Factorial of a number n

```
!5 = 5 * 4 * 3 * 2 * 1  
!n = n * n-1 * n-2 * ... * 2 * 1
```

```
def factI(self,n:'int')->'int':  
    ans = 1  
    for i in range(2,n+1):  
        ans = ans * i  
    return ans
```

```
!5 = 5 * 4 * 3 * 2 * 1  
!4 = 4 * 3 * 2 * 1  
  
!5 = 5 * !4
```

```
def factR(self,n:'int')->'int':  
    if (n < 2):  
        return 1  
    return (n * self.factR(n- 1))
```

Stack overflow for n = 1000
RecursionError: maximum recursion depth exceeded in comparison

Figure 3.1: Factorial of a number

3.2.1 Time and space complexity

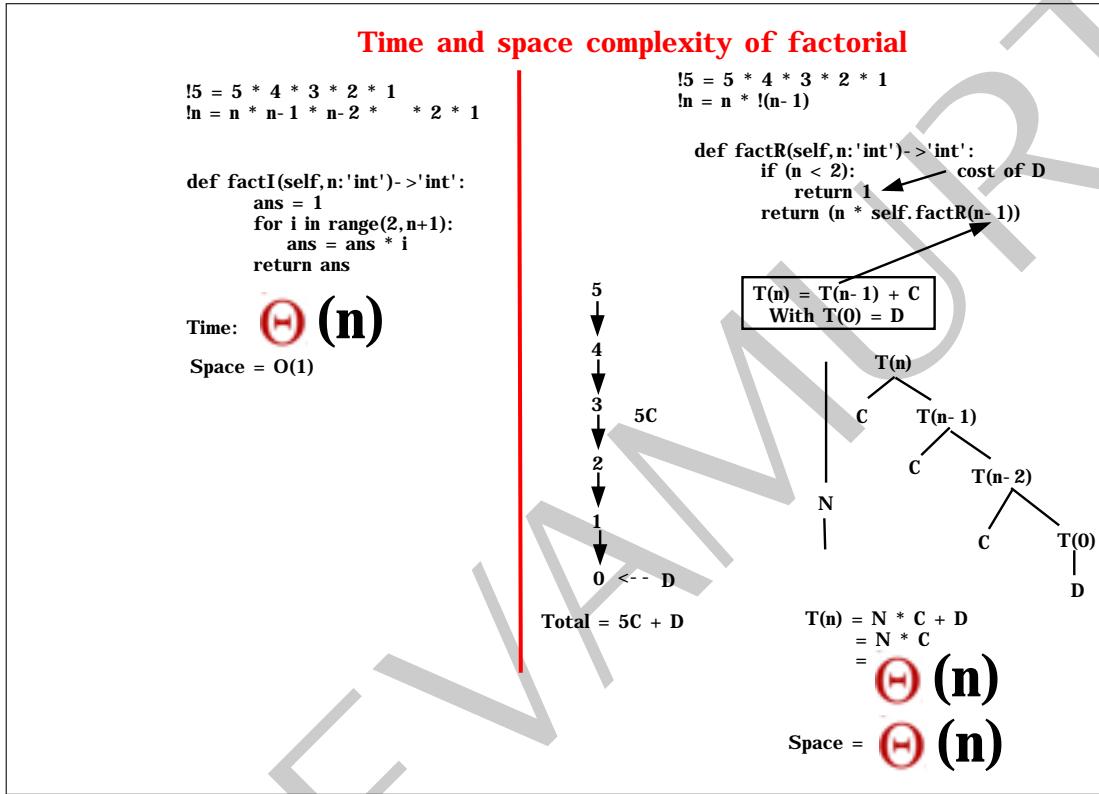


Figure 3.2: Time and space complexity

$$3.3 \quad \sum_{i=0}^n i$$

3.3. $\sum_{i=0}^N i$

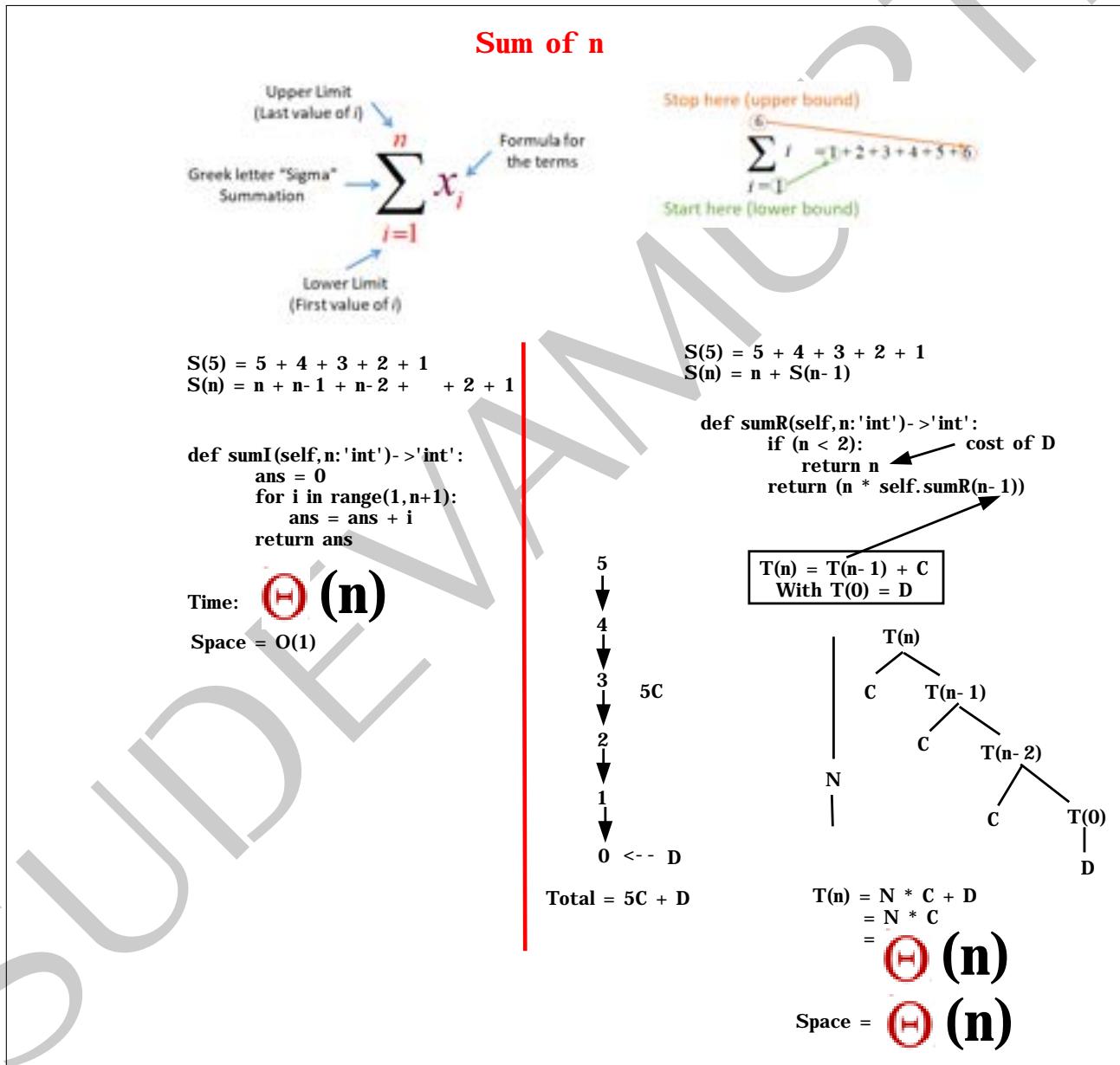


Figure 3.3: $\sum_{i=0}^n i$

3.4 Computing number of binary digits to represent a decimal number

Number of Binary Digit to Represent a Decimal number N

D	B
0	0
1	1
2	01
3	11
4	100
5	101
15	1111
16	10000

```

def numBinaryDigit(n):
    ans = 1 #n = 0 and 1 requires 1 digit
    if (n > 1):
        n = 0
        while (n > 0):
            ans = ans + 1
            n = n // 2
    return ans;

```

$2)15(7$
 $2)7(3$
 $2)3(1$
 $2)1(0$

Time = Theta($\log_2 n$)
Space = O(1)

```

def numBinaryDigitR_r(int n):
    if (n > 1):
        return (1 + self.numBinaryDigitR_r(n // 2))
    return 1 //n = 0 and 1 requires 1 digit
}

```

Figure 3.4: Computing number of binary digits to represent a decimal number

3.4.1 Time and space complexity

3.4. COMPUTING NUMBER OF BINARY DIGITS TO REPRESENT A DECIMAL NUMBER

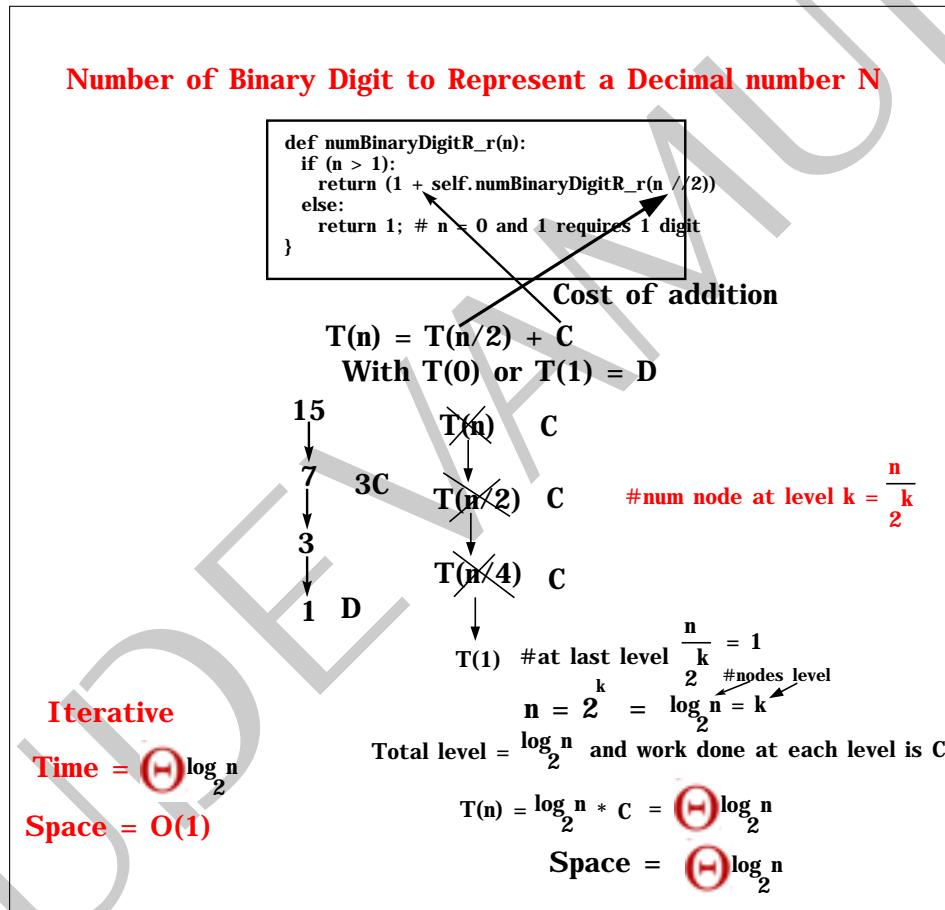
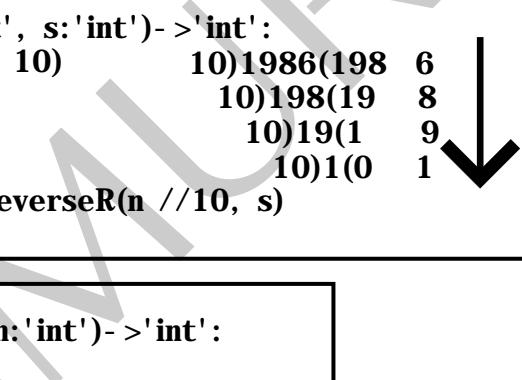


Figure 3.5: Time and space complexity

3.5 Reverse a number

Reverse an integer

<pre>n a ----- 0 0 1 1 10 1 100 1 1986 6891 12345 54321</pre>	<pre>def _reverseR(self, n:'int', s:'int')->'int': s = s * 10 + (n % 10) if (n < 10): return s else: return self._reverseR(n // 10, s)</pre>	<pre>10)1986(198 6 10)198(19 8 10)19(1 9 10)1(0 1</pre> 
--	--	--

```
def reverseR(self, n:'int')->'int':
    s = 0
    return self._reverseR(n, s)
```

a = reverseR(1986)

Time = $\Theta(\log_{10} n)$

Space = $\Theta(\log_{10} n)$

Figure 3.6: Reversing a number

3.6 Reverse a *slist*

3.7 Fibonacci numbers

Fibonacci numbers

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}
0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597	2584	4181	6765

```
def FibI(self, n: 'int') -> 'int':
    if (n < 2):
        return n
    lastlast = 0
    last = 1
    for i in range(2, n+1):
        ans = lastlast + last
        lastlast = last
        last = ans
    return ans
```

$\Theta(n)$
Constant space

```
def FibR(self, n: 'int') -> 'int':
    if (n < 2):
        return n
    return (self.FibR(n- 1) + self.FibR(n- 2))
```

Time: $O(1.6^n)$
Space: $O(n)$

```
[root@asus ~]#-----Testing Fibonacci-----
Fib( 0 ) Iterative = 0 CPU = 0.0
Fib( 0 ) Recursive = 0 CPU = 0.0
Fib( 1 ) Iterative = 1 CPU = 0.0
Fib( 1 ) Recursive = 1 CPU = 0.0
Fib( 9 ) Iterative = 34 CPU = 0.0
Fib( 9 ) Recursive = 34 CPU = 0.0
Fib( 10 ) Iterative = 55 CPU = 0.0
Fib( 10 ) Recursive = 55 CPU = 0.0
Fib( 14 ) Iterative = 377 CPU = 0.0
Fib( 14 ) Recursive = 377 CPU = 0.0
Fib( 20 ) Iterative = 6765 CPU = 0.0
Fib( 20 ) Recursive = 6765 CPU = 0.0
Fib( 40 ) Iterative = 102334155 CPU = 0.0
Fib( 40 ) Recursive = 102334155 CPU = 40.453125
Fib( 45 ) Iterative = 1134983170 CPU = 0.0
Fib( 45 ) Recursive = 1134983170 CPU = 447.40625
What CPU time it can take for FibR(50)?
```

Figure 3.7: Fibonacci numbers

3.7. FIBONACCI NUMBERS

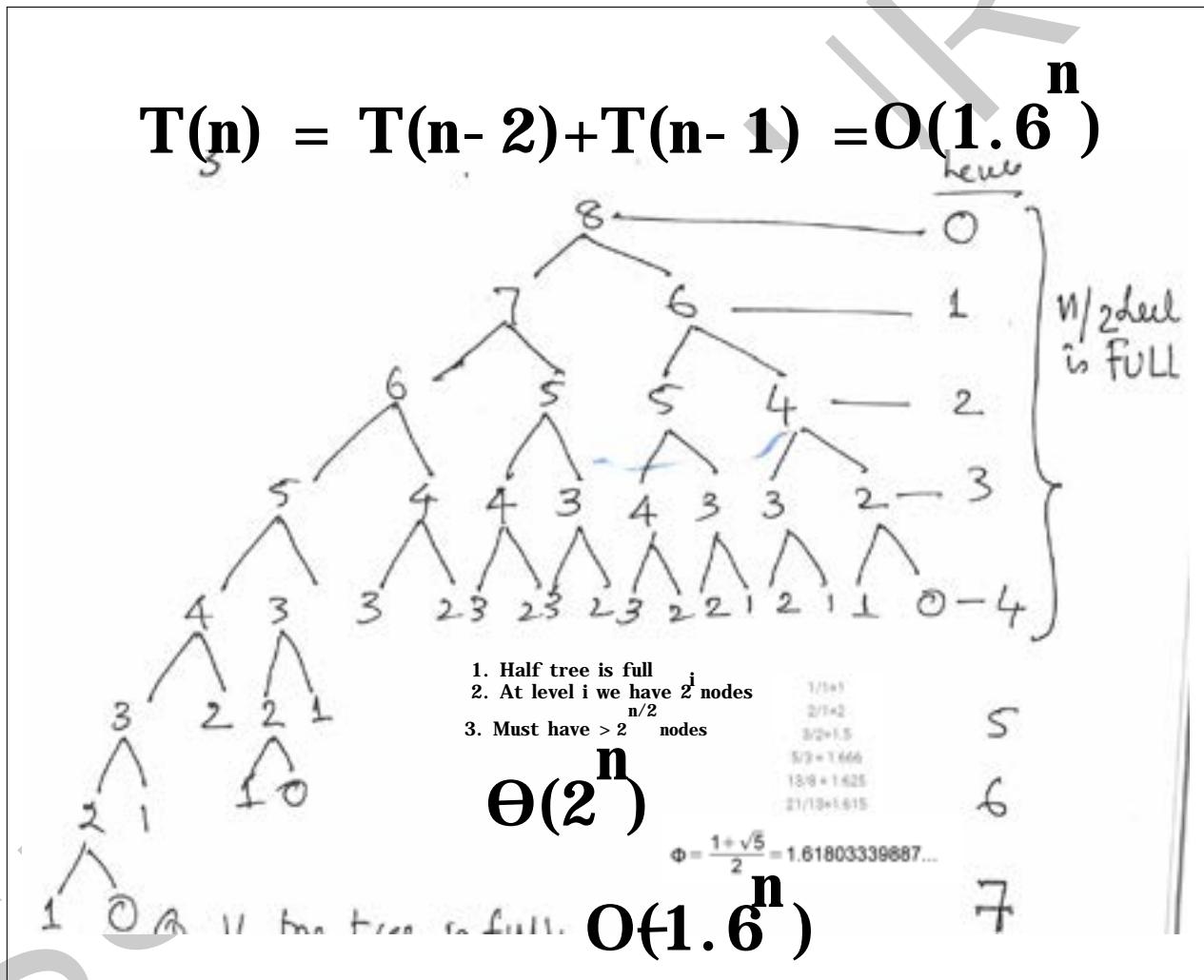


Figure 3.8: Complexity of computing Fibonacci numbers using recursion

3.8 Tower of Hanoi

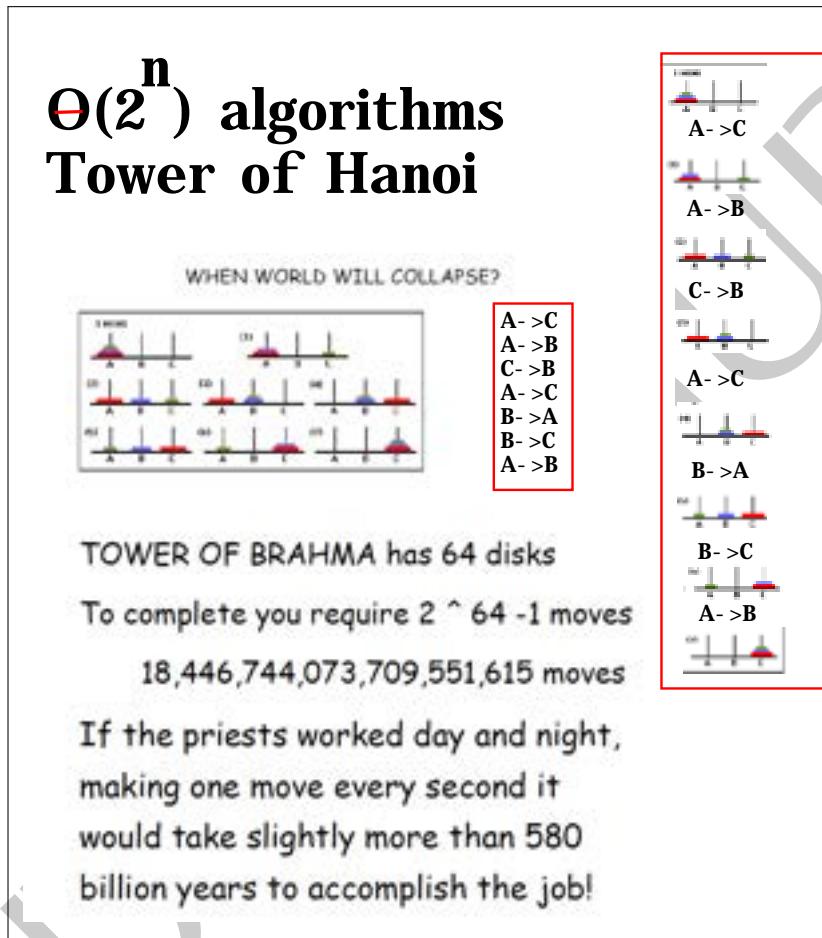


Figure 3.9: Tower of Hanoi

3.8. TOWER OF HANOI

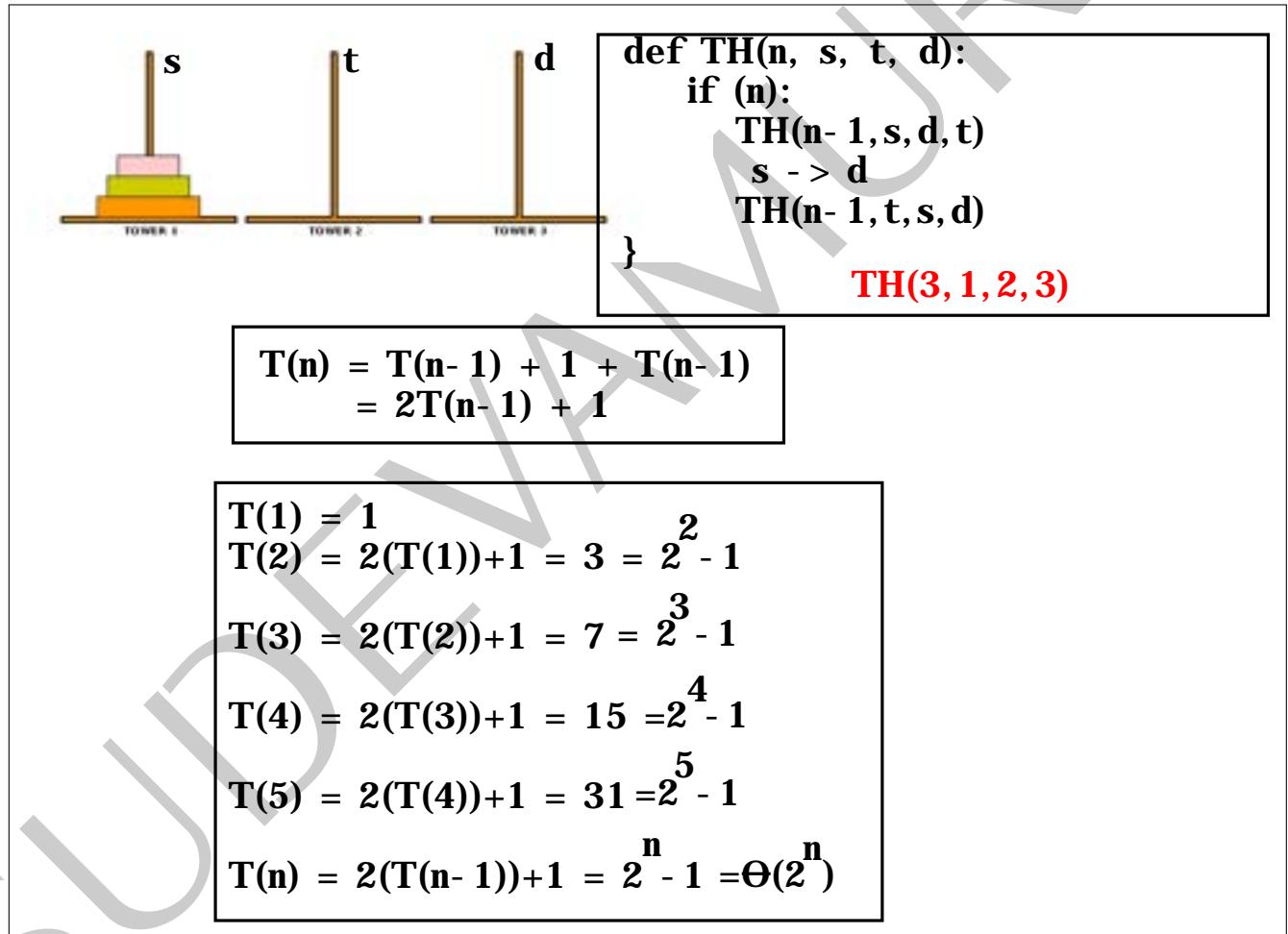


Figure 3.10: Tower of Hanoi Algorithm

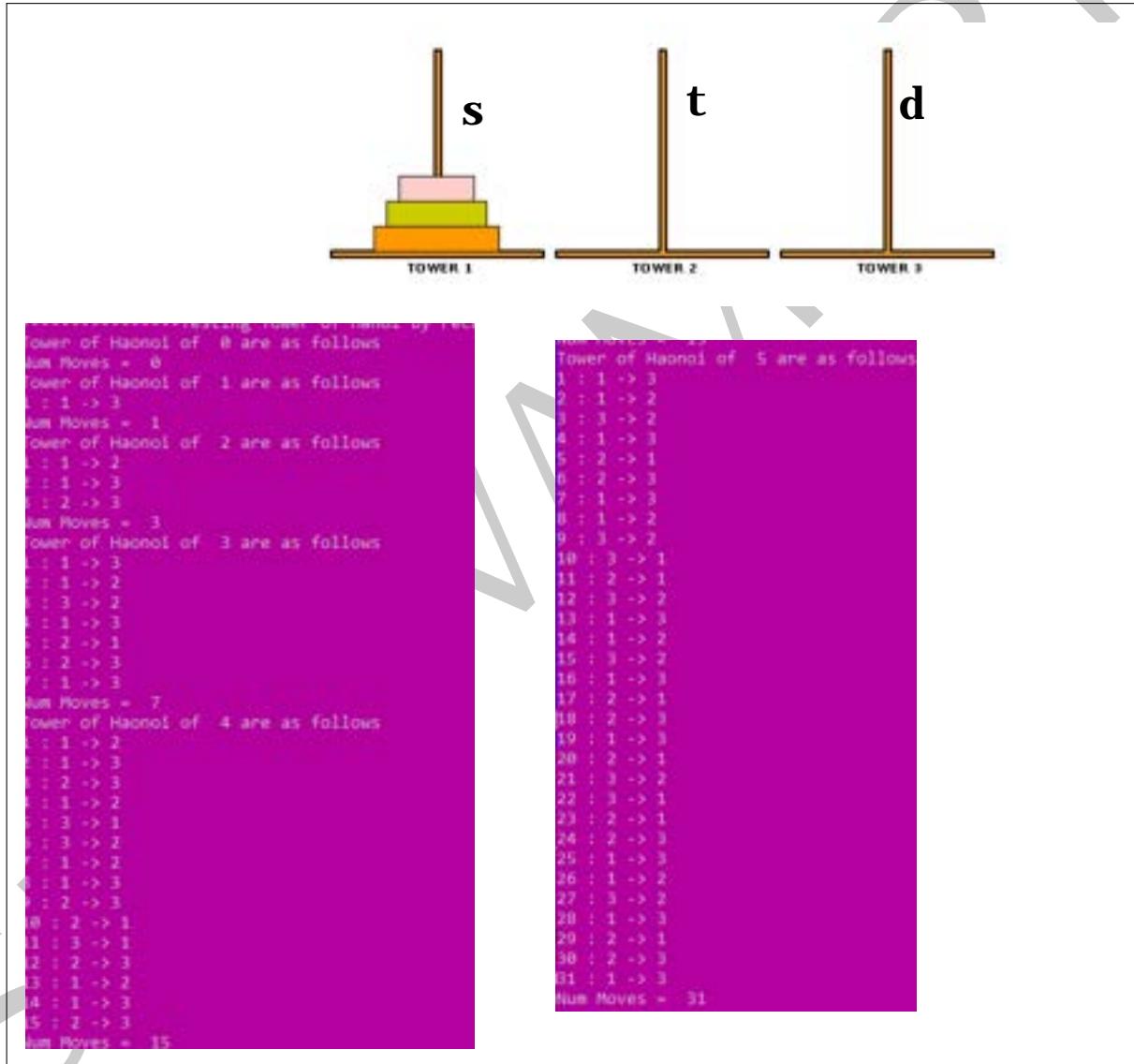


Figure 3.11: Tower of Hanoi solution

3.9 Permutation

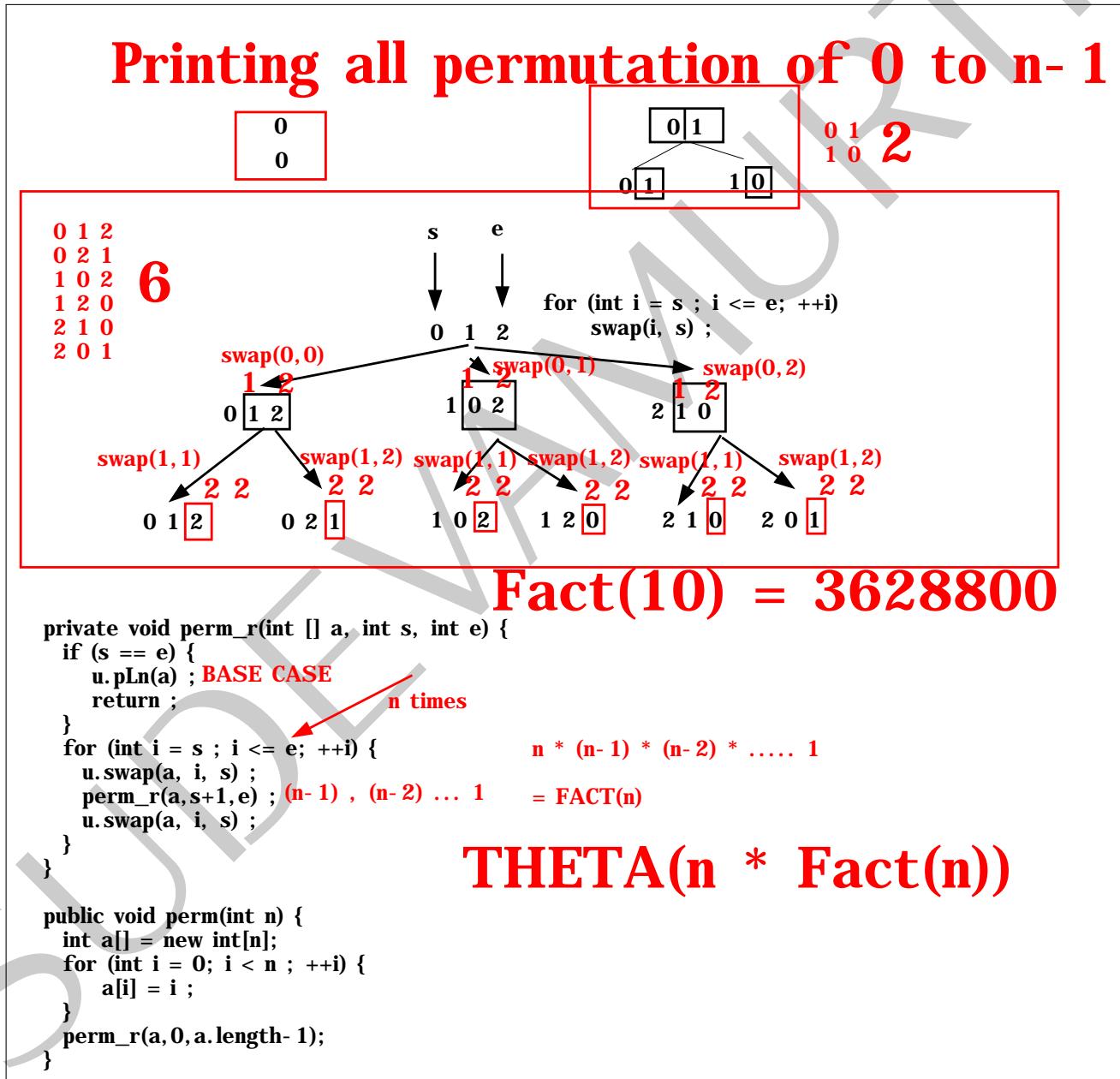


Figure 3.12: Permutation

Printing all permutation of 0 to n- 1

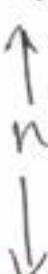
```
Permutation of  0 are as follows
Num permutation =  1
Permutation of  1 are as follows
Num permutation =  1
Permutation of  2 are as follows
1 :[0, 1]
2 :[1, 0]
Num permutation =  2
Permutation of  3 are as follows
1 :[0, 1, 2]
2 :[0, 2, 1]
3 :[1, 0, 2]
4 :[1, 2, 0]
5 :[2, 1, 0]
6 :[2, 0, 1]
Num permutation =  6
Permutation of  4 are as follows
1 :[0, 1, 2, 3]
2 :[0, 1, 3, 2]
3 :[0, 2, 1, 3]
4 :[0, 2, 3, 1]
5 :[0, 3, 2, 1]
6 :[0, 3, 1, 2]
7 :[1, 0, 2, 3]
8 :[1, 0, 3, 2]
9 :[1, 2, 0, 3]
10 :[1, 2, 3, 0]
11 :[1, 3, 2, 0]
12 :[1, 3, 0, 2]
13 :[2, 1, 0, 3]
14 :[2, 1, 3, 0]
15 :[2, 0, 1, 3]
16 :[2, 0, 3, 1]
17 :[2, 3, 0, 1]
18 :[2, 3, 1, 0]
19 :[3, 1, 2, 0]
20 :[3, 1, 0, 2]
21 :[3, 2, 1, 0]
22 :[3, 2, 0, 1]
23 :[3, 0, 2, 1]
24 :[3, 0, 1, 2]
Num permutation =  24
```

Figure 3.13: Permutation Solution

3.10 Recursion trees

$$\boxed{\begin{aligned} T(n) &= T(n-1) + C \\ T(1) &= 1 \end{aligned}}$$

KID



CANDY

KID getting Candy

$$\boxed{fib(n) = n * fib(n-1)}$$

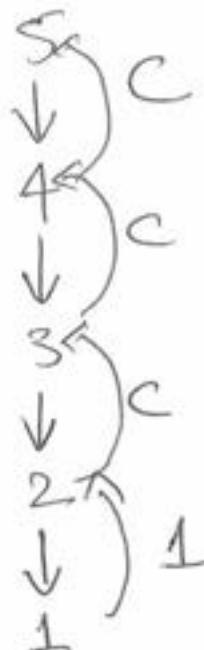
Finding Fibonacci of n

$$T(n) = T(n-1) + C$$

T
n

$$\begin{array}{c} \cancel{T(n)} \quad C \\ | \\ \cancel{T(n-1)} \quad C \\ | \\ \cancel{T(n-2)} \quad C \\ | \\ T(n-3) \end{array}$$

$$T(1) = 1$$

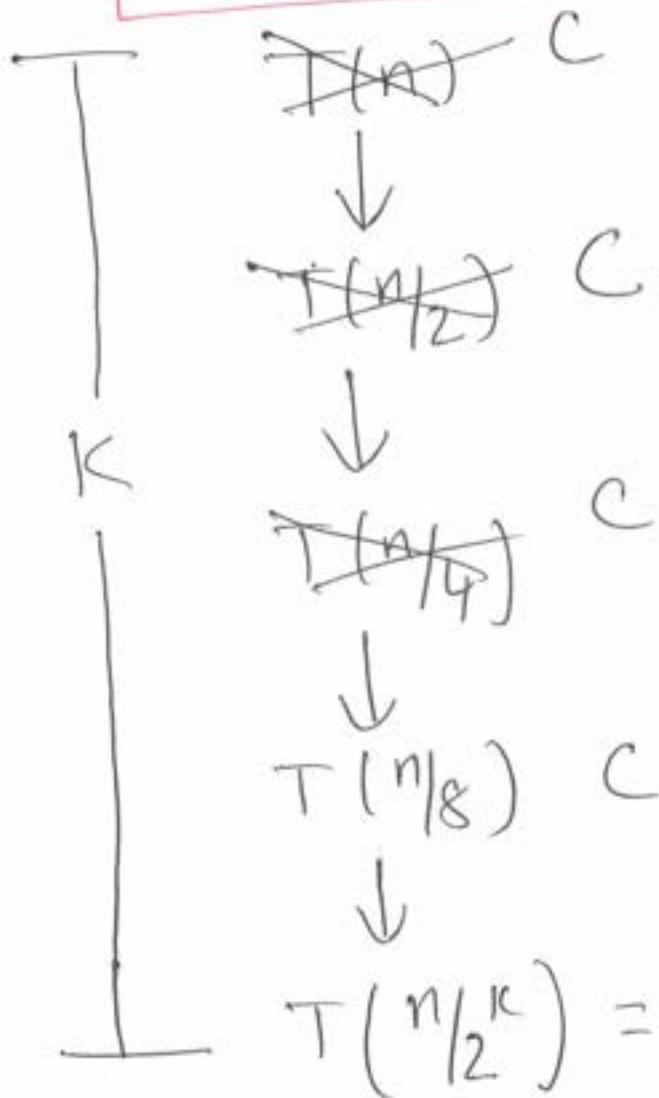


$$\begin{aligned} \text{Total Work} &= \left(\underbrace{C + C + C + C + \dots}_{n-1} \right) n + 1 \\ &= C(n-1) + 1 \\ &= Cn - C + 1 = \Theta(n) \end{aligned}$$

$$\begin{cases} T(n) = T(n/2) + c \\ T(1) = 1 \end{cases}$$

2

FINDING A WORD IN DICTIONARY



16
↓
8
↓
4
↓
2

$$2^4 = 16$$

$$\boxed{\log_2 16 = 4}$$

$$\frac{n}{2^K} = 1 \quad \therefore n = 2^K \quad \therefore K = \log_2 n$$

$$\begin{aligned} \text{Total Work} &= K \cdot c \\ &= \log_2 n \cdot c = \Theta(\log_2 n) \end{aligned}$$

$$T(n) = 2T\left(\frac{n}{2}\right) + C$$

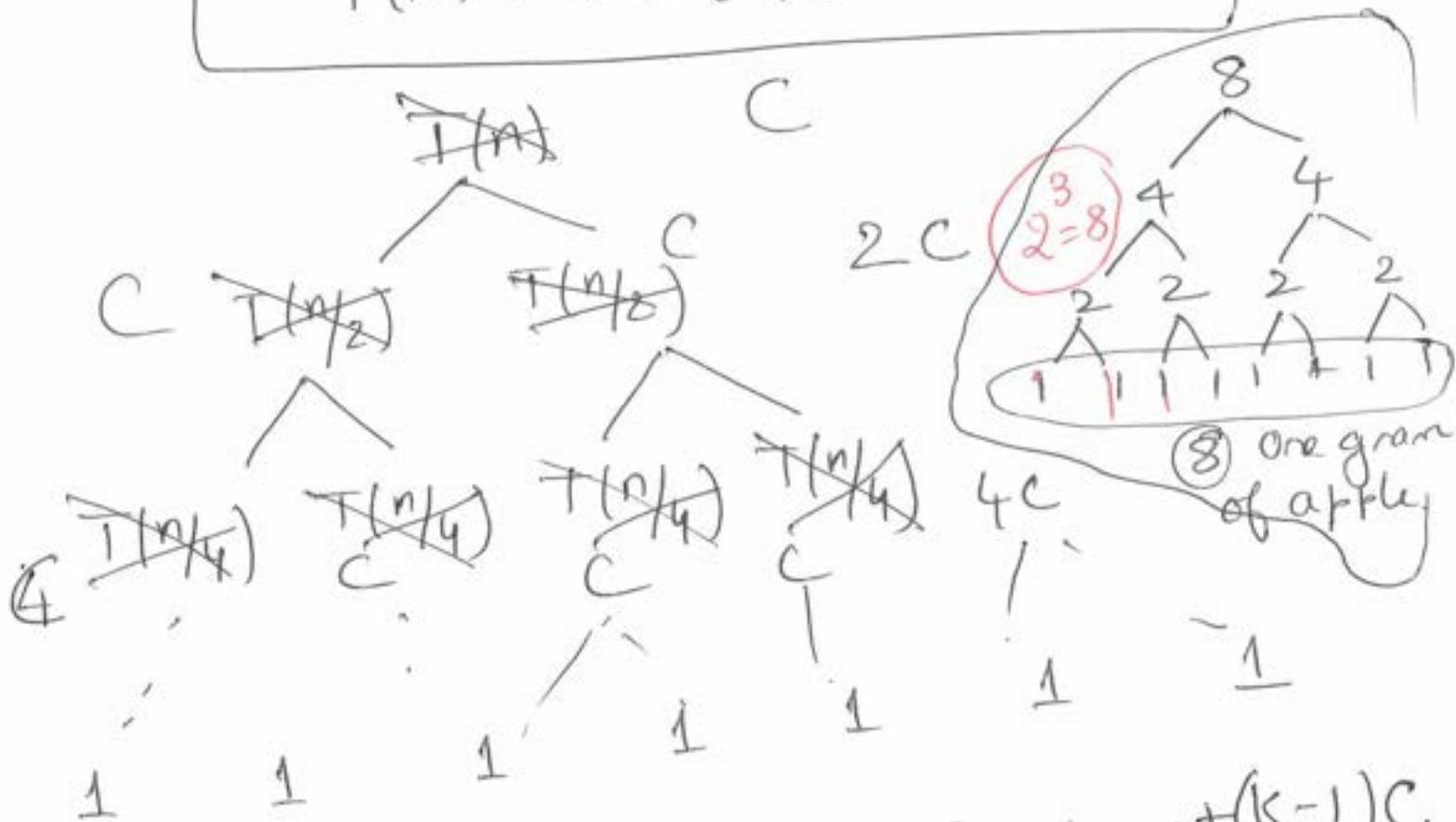
$$T(1) = 1$$

3

n grams of Apple.
cut apple by Half and eat

$$T(n) \neq T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + C$$

$$T(n) = 2T\left(\frac{n}{2}\right) + C$$



$$\text{Total work } T(n) = C + 2C + 4C + 8C + \dots + (2^{k-1})C$$

$$T(n) = C \left(2^0 + 2^1 + 2^2 + \dots + 2^{k-1} \right)$$

$$= C (2^k - 1) = 2^k C$$

But $2^k = n$

$$T(n) = C(n+1) \\ = Cn = \Theta(n)$$

$$\boxed{\begin{aligned} T(n) &= T(n/2) + n \\ T(1) &= 1 \end{aligned}} \quad (4)$$

$$\begin{array}{ccc} T(n) & n & n/2^0 \\ | & & \\ T(n/2) & n/2 & n/2^1 \\ | & & \\ T(n/4) & n/4 & n/2^2 \\ | & & \\ \vdots & & \\ T(1) = & n/2^K = 1 & \end{array}$$

$$\frac{n}{2^K} = 1$$

$$2^K = n$$

$$n = 2^K$$

$$K = \log_2 n$$

$$\text{Total work} = n \left(\frac{1}{2^0} + \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^K} \right)$$

$$= n \left(1 + \underbrace{0.5 + 0.25 + \dots}_{\frac{1}{205}} \overbrace{+ \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n} = 1} \right)$$

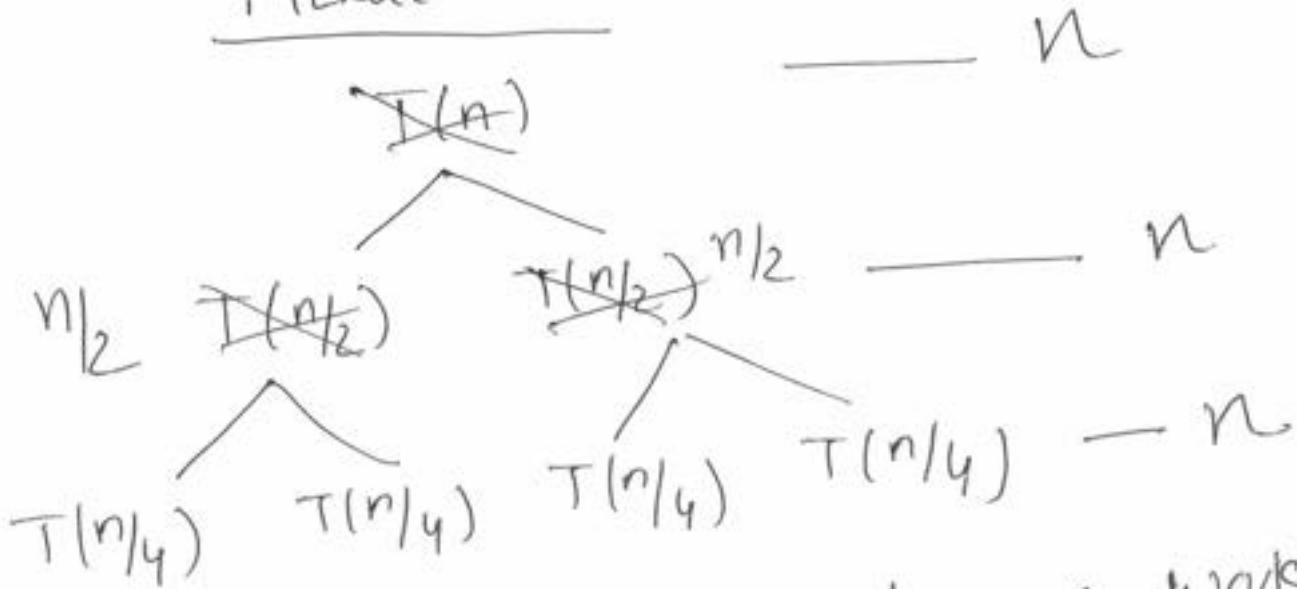
$$n^{(2)} = \Theta(n) \quad \boxed{T(n) = \Theta(n)}$$

(5)

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

$$T(1) = 1$$

MERGE SORT



In every level we are doing n work

$$T\left(\frac{n}{2^k}\right) = 1 \quad \frac{n}{2^k} = 1$$

$$k = \log_2 n$$

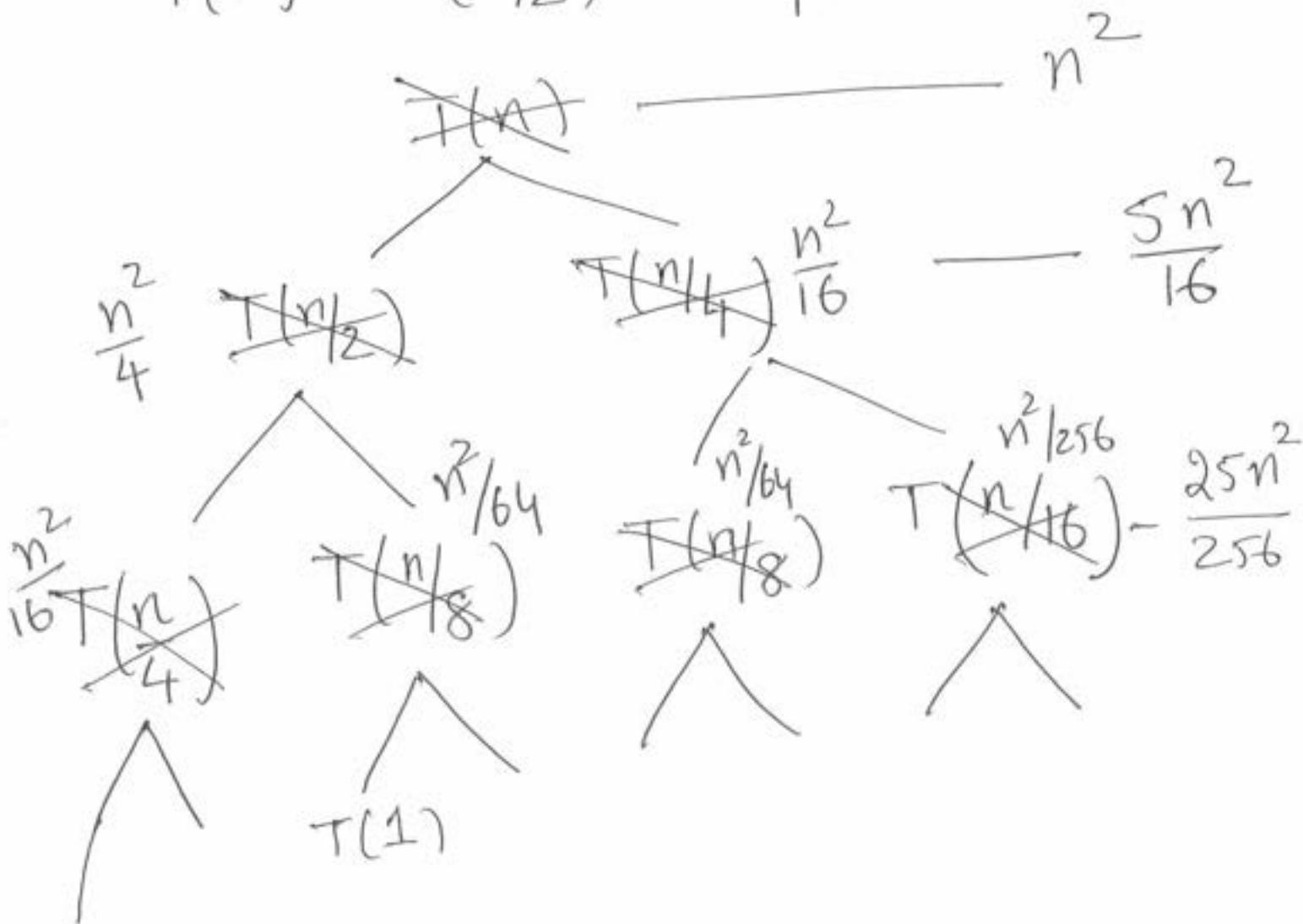
Tree has $\log_2 n$ levels.

So total work done $T(n) = n \cdot \log_2 n$

$$T(n) = \Theta(n \log_2 n)$$

⑥

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + n^2$$

 $T(1)$

How MANY Leaves are there?

This is not so obvious as we are recurring with different speeds

We are solving n problem as $\frac{n}{2} + \frac{n}{4} = \frac{3n}{4}$ problemSo Number of leaves must be less than n smaller than

$$\begin{aligned} & n^2 + \frac{5n^2}{16} + \frac{25n^2}{256} + \dots + \frac{5^K n^2}{16^K} \\ & \leq n^2 \left(1 + \frac{5}{16} + \frac{5^2}{16^2} + \frac{5^3}{16^3} + \dots + \frac{5^K}{16^K} \right) \leq 2n^2 \end{aligned}$$

$$\begin{aligned} & 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots \\ & = 2 \end{aligned}$$

$$T(n) = T(n-1) + T(n-2) + C$$

$$T(1) = 1$$

$$T(0) = 0$$

$$\left[\begin{array}{l} \text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2) \\ \text{fib}(1) = 1 \\ \text{fib}(0) = 0 \end{array} \right]$$

$$\left[T(n) = \Theta(1 \cdot 6^n) \right]$$

$$T(n) = T(n-1) + T(n-1) + C$$

$$T(1) = 1$$

$$\left[\begin{array}{l} \text{Th(int n, int s, int t, int d)} \\ \quad \text{if}(n) \{ \\ \quad \quad \text{Th}(n-1, s, d, t) \\ \quad \quad \text{3 3 } \xrightarrow{s \rightarrow d} \text{Th}(n-1, t, s, d) \end{array} \right]$$

$$\left[T(n) = \Theta(2^n) \right]$$

3.11 Master Theorem

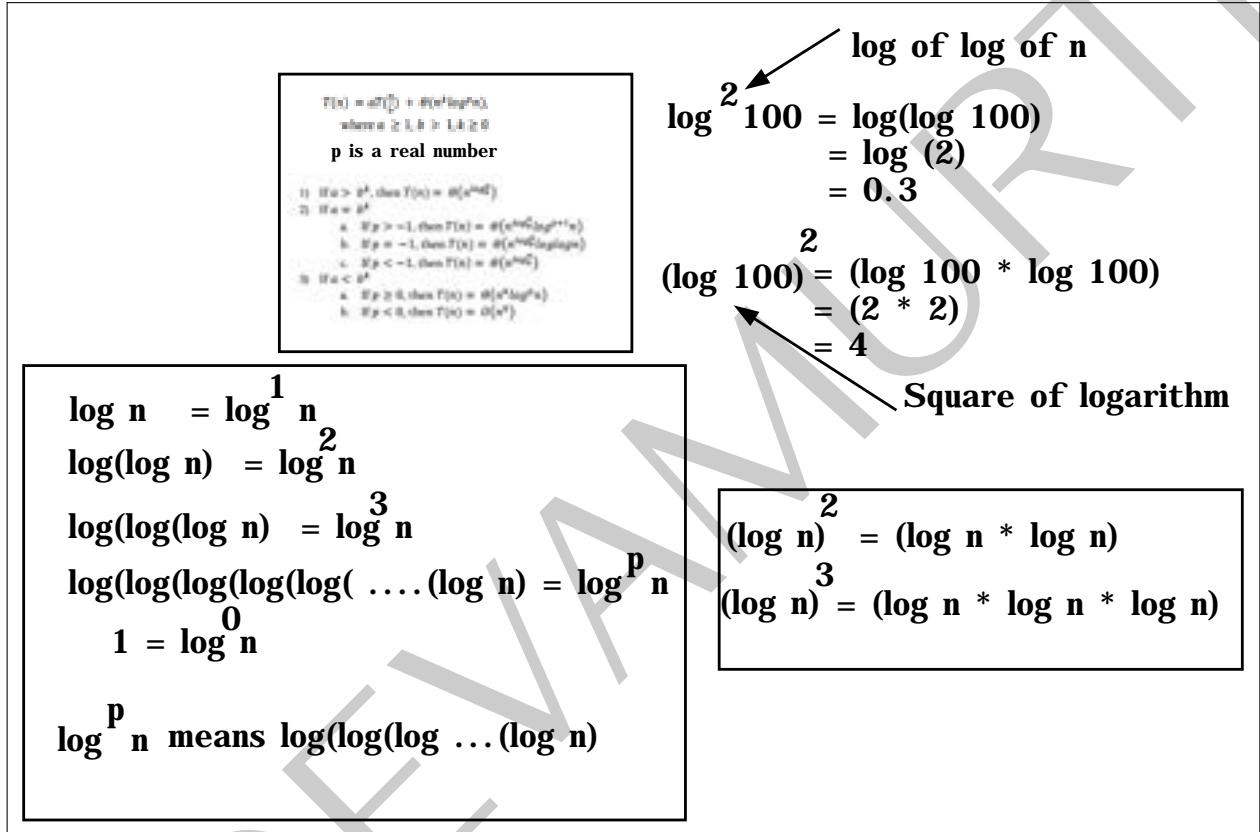


Figure 3.14: Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^k \log^p n),$$

where $a \geq 1, b > 1, k \geq 0$

p is a real number

1) If $a > b^k$, then $T(n) = \Theta(n^{\log_b^k})$

2) If $a = b^k$

a. If $p > -1$, then $T(n) = \Theta(n^{\log_b^k} \log^{p+1} n)$

b. If $p = -1$, then $T(n) = \Theta(n^{\log_b^k} \log \log n)$

c. If $p < -1$, then $T(n) = \Theta(n^{\log_b^k})$

3) If $a < b^k$

a. If $p \geq 0$, then $T(n) = \Theta(n^k \log^p n)$

b. If $p < 0$, then $T(n) = O(n^k)$

$$1. T(n) = 3T\left(\frac{n}{2}\right) + n^2$$

$$2. T(n) = 7T\left(\frac{n}{2}\right) + n^2$$

$$3. T(n) = 4T\left(\frac{n}{2}\right) + n^2$$

$$4. T(n) = 3T\left(\frac{n}{4}\right) + n \log n$$

$$5. T(n) = 4T\left(\frac{n}{2}\right) + \log n$$

$$6. T(n) = T(n-1) + n$$

$$7. T(n) = 4T\left(\frac{n}{2}\right) + n^2 \log n$$

$$8. T(n) = 5T\left(\frac{n}{2}\right) + n^2 \log n$$

$$9. T(n) = 3T\left(\frac{n}{3}\right) + n/\log n$$

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^k \log^p n),$$

where $a \geq 1, b > 1, k \geq 0$

p is a real number

- 1) If $a > b^k$, then $T(n) = \Theta(n^{\log_b^k})$
- 2) If $a = b^k$
 - a. If $p > -1$, then $T(n) = \Theta(n^{\log_b^k} \log^{p+1} n)$
 - b. If $p = -1$, then $T(n) = \Theta(n^{\log_b^k} \log \log n)$
 - c. If $p < -1$, then $T(n) = \Theta(n^{\log_b^k})$
- 3) If $a < b^k$
 - a. If $p \geq 0$, then $T(n) = \Theta(n^k \log^p n)$
 - b. If $p < 0$, then $T(n) = O(n^k)$

10) $T(n) = 2T(n/4) + C$

11) $T(n) = T(n/4) + \log n$

12) $T(n) = T(n/2) + T(n/4) + n^2$

13) $T(n) = 2T(n/4) + \log n$

14) $T(n) = 3T(n/3) + n \log n^2$

15) $T(n) = 8T((n-\sqrt{n})/4) + n$

16) $T(n) = 2T(n/4) + \sqrt{n}$

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^k \log^p n),$$

where $a \geq 1, b > 1, k \geq 0$

p is a real number

1) If $a > b^k$, then $T(n) = \Theta(n^{\log_b^a})$

2) If $a = b^k$

a. If $p > -1$, then $T(n) = \Theta(n^{\log_b^a} \log^{p+1} n)$

b. If $p = -1$, then $T(n) = \Theta(n^{\log_b^a} \log \log n)$

c. If $p < -1$, then $T(n) = \Theta(n^{\log_b^a})$

3) If $a < b^k$

a. If $p \geq 0$, then $T(n) = \Theta(n^k \log^p n)$

b. If $p < 0$, then $T(n) = O(n^k)$

$$17) T(n) = 2T(n/4) + n^{0.5}$$

$$18) T(n) = 16T(n/4) + n!$$

$$19) T(n) = 3T(n/2) + n$$

$$20) T(n) = 4T(n/2) + cn$$

$$21) T(n) = 3T(n/3) + n/2$$

$$22) T(n) = 4T(n/2) + n/\log n$$

$$23) T(n) = 7T(n/3) + n^2$$

$$24) T(n) = 8T_{212}(n/3) + 2^n$$

$$25) T(n) = 16T(n/4) + n$$

3.11.1 Solution

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^k \log^p n),$$

where $a \geq 1, b > 1, k \geq 0$

p is a real number

- 1) If $a > b^k$, then $T(n) = \Theta(n^{\log_b^k})$
- 2) If $a = b^k$
 - a. If $p > -1$, then $T(n) = \Theta(n^{\log_b^k} \log^{p+1} n)$
 - b. If $p = -1$, then $T(n) = \Theta(n^{\log_b^k} \log \log n)$
 - c. If $p < -1$, then $T(n) = \Theta(n^{\log_b^k})$
- 3) If $a < b^k$
 - a. If $p \geq 0$, then $T(n) = \Theta(n^k \log^p n)$
 - b. If $p < 0$, then $T(n) = O(n^k)$

$$T(n) = 3T\left(\frac{n}{2}\right) + n^2$$

a = 3

b = 2

k = 2 p = 0

$$\begin{array}{c} a \\ \hline 3 \end{array} \quad \begin{array}{c} b^k \\ \hline 2^2 \\ 2 \end{array} \quad p = 0 \quad \text{Case 3a} \quad \Theta(n^2 \log^0 n) \\ 3 \leq 4 \\ \underline{\text{Can 3}} \quad \quad \quad = \Theta(n^2)$$

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^k \log^p n),$$

where $a \geq 1, b > 1, k \geq 0$

p is a real number

- 1) If $a > b^k$, then $T(n) = \Theta(n^{\log_b^a})$
- 2) If $a = b^k$
 - a. If $p > -1$, then $T(n) = \Theta(n^{\log_b^k} \log^{p+1} n)$
 - b. If $p = -1$, then $T(n) = \Theta(n^{\log_b^k} \log \log n)$
 - c. If $p < -1$, then $T(n) = \Theta(n^{\log_b^k})$
- 3) If $a < b^k$
 - a. If $p \geq 0$, then $T(n) = \Theta(n^k \log^p n)$
 - b. If $p < 0$, then $T(n) = O(n^k)$

$$T(n) = \lceil T\left(\frac{n}{2}\right) + n^2 \rceil$$

$(a=1)$ $\boxed{b=2}$ $\boxed{k=2 \quad p=0}$

$$\frac{a}{1} \quad \frac{b^k}{2^2} \quad \boxed{T(n) = \Theta(n^{\log_2 1})}$$

$1 > 4$
Case 1

(3)

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^k \log^p n),$$

where $a \geq 1, b > 1, k \geq 0$

p is a real number

- 1) If $a > b^k$, then $T(n) = \Theta(n^{\log_b^k})$
- 2) If $a = b^k$
 - a. If $p > -1$, then $T(n) = \Theta(n^{\log_b^k} \log^{p+1} n)$
 - b. If $p = -1$, then $T(n) = \Theta(n^{\log_b^k} \log \log n)$
 - c. If $p < -1$, then $T(n) = \Theta(n^{\log_b^k})$
- 3) If $a < b^k$
 - a. If $p \geq 0$, then $T(n) = \Theta(n^k \log^p n)$
 - b. If $p < 0$, then $T(n) = O(n^k)$

$$T(n) = 4T\left(\frac{n}{2}\right) + n^2$$

$$a = 4$$

$$b = 2$$

$$k = 2$$

$$p = 0$$

$$\frac{a}{b^k} = \frac{4}{2^2} = \frac{4}{4} = 1$$

$$4 = 4$$

$$(\text{Case 2})$$

$$(\text{Case 2a})$$

$$T(n) = \Theta\left(n^{\log_2 4} \log^{(0+1)} n\right)$$

$$T(n) = \Theta(n^2 \log n)$$

$$\log_2 \frac{4}{4} = \log_2 2^2 = 2 \log_2 2 = 2$$

3.12 Problem set

Problem 3.12.1. Implement Permutations- Leetcode problem: 46 as shown in figures 20.17.

3.12. PROBLEM SET

Problem 3.12.2. Implement **Buying and Selling stocks**. Leetcode problem: 121 as shown in figures 20.19.

Problem 3.12.3. Implement **Hop** as shown in figures 20.15.

3.12. PROBLEM SET

Problem 3.12.4. Implement **One dimensional peak**. Leetcode problem: 162 as shown in figures 20.16.

Chapter 4

Python List Implementation

4.1 Introduction

4.2 List Class

Lists:			
Operation	Example	Complexity	Notes
Index	<code>l[1]</code>	$O(1)$	
Store	<code>l[i] = 0</code>	$O(1)$	
Length	<code>len(l)</code>	$O(1)$	
Append	<code>l.append(5)</code>	$O(1)$	mostly: $O(n)$ covers details
Pop	<code>l.pop()</code>	$O(1)$	same as <code>l.pop(-1)</code> , popping at end
Clear	<code>l.clear()</code>	$O(1)$	similar to <code>l = []</code>
Slice	<code>l[a:b]</code>	$O(b-a)$	<code>l[1:5]</code> : $O(1)$ / <code>l[:]</code> : $O(len(l))=O(N)$
Extend	<code>l.extend(...)</code>	$O(len(...))$	depends only on len of extension
Construction	<code>list(...)</code>	$O(len(...))$	depends on length of ... iterable
check ==, !=	<code>l1 == l2</code>	$O(N)$	
Insert	<code>l[a:b] + ...</code>	$O(N)$	
Delete	<code>del l[i]</code>	$O(N)$	depends on i; $O(N)$ in worst case
Containment	<code>x in/not in l</code>	$O(N)$	linearly searches list
Copy	<code>l.copy()</code>	$O(N)$	Same as <code>l[:]</code> which is $O(N)$
Remove	<code>l.remove(...)</code>	$O(N)$	
Pop	<code>l.pop(1)</code>	$O(N)$	$O(N-i)$: <code>l.pop(0)</code> : $O(N)$ (see above)
Extreme value	<code>min(l)/max(l)</code>	$O(N)$	linearly searches list for value
Reverse	<code>l.reverse()</code>	$O(N)$	
Iteration	<code>for v in l:</code>	$O(N)$	Worst: no return/break in loop
Sort	<code>l.sort()</code>	$O(N \log N)$	key/reverse mostly doesn't change
Multiply	<code>k*l</code>	$O(k N)$	$5*1$ is $O(N)$; <code>len(l)*l</code> is $O(N^2)$

Figure 4.1: Complexity of Python List

list implementation as dynamic array

Copyright: Jagadeesh Vasudevamurthy

filename: list.ipynb

WRITE CODE BELOW

In []:

```
1 # # List.py
2 # Implementation of Python LIST as dynamic array
3 # Author: Jagadeesh Vasudevamurthy
4 # Copyright: Jagadeesh Vasudevamurthy 2021
5 ######
6
7 #####
8 # ALL imports here
9 #####
10 import ctypes
11
12 #####
13 # List (Java Like List implemented as dynamic array)
14 #####
15 class List(object):
16     def __init__(self):
17         self._k = 0 ;
18         self._capacity = 8
19         self._a = self._allocate(self._capacity)
20
21     ######
22     # ALL Public functions below
23     # WRITE CODE BELOW
24     #####
25
26     ######
27     # ALL private functions below
28     # WRITE CODE BELOW
29     #####
30
31     ######
32     # Time:O(1)
33     # Space:O(1)
34     # Cannot be changed
35     #####
36     def _allocate(self,new_cap):
37         return (new_cap * ctypes.py_object)()
38
39
40     ######
41     # Time:O(n)
42     # Space:O(n)
43     # Cannot be changed
44     #####
45     def _grow(self):
46         os = self._capacity
47         ns = os * 2
48         print("Grow from", os, "to", ns, ". This is not O(1)")
49         b = self._allocate(ns) # New bigger array
50
51         for k in range(os): # Reference all existing values
52             b[k] = self._a[k]
53         self._a = b # Call b as the new bigger array
54         self._capacity = ns # Reset the capacity
55
56         223
```

ListTest.py

CANNOT BE CHANGED

In []:

```

1 ######
2 # ListTest.py
3 # Test Implementation of Python LIST as dynamic array object
4 # Author: Jagadeesh Vasudevamurthy
5 # Copyright: Jagadeesh Vasudevamurthy 2021
6 #####
7
8 #####
9 # All imports here
10 #####
11 import sys # For getting Python Version
12 #from List import *
13
14 #####
15 # Test List_grow_append_find
16 #####
17 class List_grow_append_change_find():
18     def __init__(self):
19         self.test_int()
20
21     def test_int(self):
22         print("----- Testing grow, append, change and find int-----")
23         d = List() # My List
24         l = [] # PYTHON LIST
25         N = 100
26         for i in range(N):
27             l.append(i *100)
28             d.append(i*100)
29         print("size of list=", len(d))
30         print("Now you can change the list from position 0 to",N-1, "in cons")
31         print("d[2] = ",d[2], "l[2] = ",l[2])
32         d[2] = -5
33         l[2] = -5
34         assert(d[2] == l[2])
35         print("d[2] = ",d[2], "l[2] = ",l[2])
36         print("To show d[i] and l[i] is constant time")
37         n = [1,2,15,99]
38         for e in n:
39             assert(d[e] == l[e])
40             print(d[e], l[e])
41         print("To show find is O(n) time")
42         if (100 in d) :
43             print("100 in List")
44         if (111 not in d) :
45             print("111 not in List")
46         print("To make sure Find in PYTHON LIST ")
47         if (100 in l) :
48             print("100 in PYTHON List")
49         if (111 not in l) :
50             print("111 not in PYTHON List")
51
52
53 #####
54 # Test List_delete
55 #####
56 class List_delete():

```

```
57  def __init__(self):
58      self._test()
59
60  def _test1(self,n:'list', item,fszie:'int'):
61      d = List() # My List
62      l = [] # PYTHON LIST
63      for e in n:
64          d.append(e)
65          l.append(e)
66      print("-----Before delete -----")
67      print("mylist d = ", d)
68      print("pylist l = ", l)
69      d.delete(item)
70      l.remove(item)
71      print("-----After deleting all", item, "from the array-----")
72      print("mylist d = ", d)
73      print("pylist l = ", l)
74      assert(len(d) == fszie)
75
76  def _test2(self,n:'integer'):
77      d = List() # My List
78      l = [] # PYTHON LIST
79      for e in range(n):
80          d.append(e)
81          l.append(e)
82      print("-----Before delete by position -----")
83      print("mylist l = ", l)
84      print("pylist d = ", d)
85      first = True ;
86      while True:
87          dsize = len(d)
88          if (dsize == 0):
89              break
90          pos = 0
91          if (dsize > 1):
92              if (first):
93                  pos = 0
94                  first = False
95              else:
96                  pos = dsize -1 ;
97                  first = True
98              print(" ----- delete element in position ", pos, " -----")
99              del(l[pos]) ;
100             del(d[pos]) ;
101             print("pylist l = ", l)
102             print("mylist d = ", d)
103             assert(len(d) == len(l))
104             for i in range(len(l)):
105                 assert(d[i] == l[i])
106
107
108  def _test(self):
109      print("----- Testing delete-----")
110      n = [1,2,99,2]
111      k = 2
112      self._test1(n,k,2)226
113
```

```
114     n = [1,1,1,1]
115     k = 1
116     self._test1(n,k,0)
117
118     n = [1,2,2,2]
119     k = 1
120     self._test1(n,k,3)
121
122     n = [2,2,2,1]
123     k = 1
124     self._test1(n,k,3)
125
126     n = ["jag","jag","bob", "tom", "jag"]
127     k = "jag"
128     self._test1(n,k,2)
129
130     print("----- Testing delete given position-----")
131     n = 10
132     self._test2(n)
133     print("Testing List ENDS")
134
135 ######
136 # MAIN
137 #####
138 def main():
139     print("Testing List starts")
140     print(sys.version)
141     ops = {
142         1: List_grow_append_change_find,
143         2: List_delete,
144     }
145     chosen_operation_function = ops.get(1) ## CHANGE 1 depending on assignme
146     chosen_operation_function();
147     chosen_operation_function = ops.get(2) ## CHANGE 1 depending on assignme
148     chosen_operation_function();
149     print("Testing List ends")
150
151 #####
152 # start up
153 #####
154 if (__name__ == '__main__'):
155     main()
156
```

4.2.1 Expected Output

```
Testing List starts
3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
----- Testing grow, append, change and find int-----
Grow from 8 to 16 . This is not O(1)
Grow from 16 to 32 . This is not O(1)
Grow from 32 to 64 . This is not O(1)
Grow from 64 to 128 . This is not O(1)
size of list= 100
Now you can change the list from position 0 to 99 in constant time
d[2] = 200 l[2] = 200
d[2] = -5 l[2] = -5
To show d[i] and l[i] is constant time
100 100
-5 -5
1500 1500
9900 9900
To show find is O(n) time
100 in List
To make sure Find in PYTHON LIST
100 in PYTHON List
111 not in PYTHON n List
----- Testing delete -----
-----Before delete -----
mylist d = [1 2 99 2 ]
pylist l = [1, 2, 99, 2]
-----After deleting all 2 from the array-----
mylist d = [1 99 ]
pylist l = [1, 99, 2]
-----Before delete -----
mylist d = [1 1 1 1 ]
pylist l = [1, 1, 1, 1]
-----After deleting all 1 from the array-----
mylist d = []
pylist l = [1, 1, 1]
-----Before delete -----
mylist d = [1 2 2 2 ]
pylist l = [1, 2, 2, 2]
-----After deleting all 1 from the array-----
mylist d = [2 2 2 ]
```

```
pylist l = [2, 2, 2]
-----Before delete -----
mylist d = [2 2 2 1 ]
pylist l = [2, 2, 2, 1]
-----After deleting all 1 from the array-----
mylist d = [2 2 2 ]
pylist l = [2, 2, 2]
-----Before delete -----
mylist d = [jag jag bob tom jag ]
pylist l = ['jag', 'jag', 'bob', 'tom', 'jag']
-----After deleting all jag from the array-----
mylist d = [bob tom ]
pylist l = ['jag', 'bob', 'tom', 'jag']
----- Testing delete given position-----
Grow from 8 to 16 . This is not O(1)
-----Before delete by position -----
mylist l = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
pylist d = [0 1 2 3 4 5 6 7 8 9 ]
----- delete element in position 0 -----
pylist l = [1, 2, 3, 4, 5, 6, 7, 8, 9]
mylist d = [1 2 3 4 5 6 7 8 9 ]
----- delete element in position 8 -----
pylist l = [1, 2, 3, 4, 5, 6, 7, 8]
mylist d = [1 2 3 4 5 6 7 8 ]
----- delete element in position 0 -----
pylist l = [2, 3, 4, 5, 6, 7, 8]
mylist d = [2 3 4 5 6 7 8 ]
----- delete element in position 6 -----
pylist l = [2, 3, 4, 5, 6, 7]
mylist d = [2 3 4 5 6 7 ]
----- delete element in position 0 -----
pylist l = [3, 4, 5, 6, 7]
mylist d = [3 4 5 6 7 ]
----- delete element in position 4 -----
pylist l = [3, 4, 5, 6]
mylist d = [3 4 5 6 ]
----- delete element in position 0 -----
pylist l = [4, 5, 6]
mylist d = [4 5 6 ]
----- delete element in position 2 -----
pylist l = [4, 5]
```

4.2. LIST CLASS

```
mylist d = [4 5 ]
----- delete element in position 0 -----
pylist l = [5]
mylist d = [5 ]
----- delete element in position 0 -----
pylist l = []
mylist d = []
Testing List ENDS
Testing List ends
```

Chapter 5

Singly Linked List

5.1 Introduction

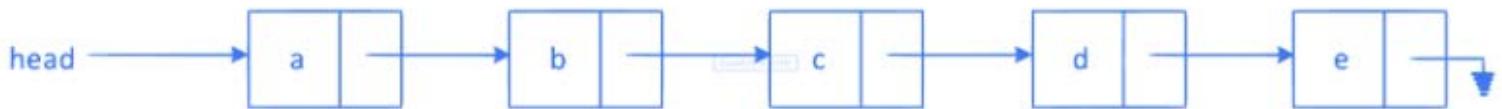
5.2 *Slist* Class

5.3. *PROBLEM SET*

5.3 Problem set

Problem 5.3.1. Answer the questions below by hand and post the scanned document on Canvas.

Consider the problem of reversing a singly linked list. To take an example, given the linked list below:



1. Show step by step how do you reverse the above list
2. What is the time complexity?
3. What is the space complexity?

let **a** = [35, 1, 89, 47, 9, 100, 5]

s->None

You CANNOT SORT a.

1

2

Show STEP BY STEP how you will insert n elements of array(Python list a) to an empty slist s maintaining the sorted ascending order

1. What is the time complexity?
2. What is the space complexity?

s-> 35 -> 1 -> 89 -> 47 -> 9 -> 100 ->5 -> NONE

YOU CANNOT USE PYTHON LIST TO STORE

Show STEP BY STEP how you can reverse slist s

3

1. What is the time complexity?
2. What is the space complexity?

s-> 35 -> 1 -> 89 -> 47 -> 9 -> 100 ->5 -> NONE

Show STEP BY STEP how you can sort slist s in ascending order
YOU CANNOT USE PYTHON LIST TO STORE

1. What is the time complexity?
2. What is the space complexity?

4

5.3. PROBLEM SET

Problem 5.3.2. LeetCode Problem 2: Add Two Numbers

Please refer to the section 20.4

Problem 5.3.3. LeetCode Problems 225, 235, 632 and 641 using *Slist*

Please refer to the section 20.13

VASUDEVAMURTHY

Chapter 6

Doubly Linked List

6.1 Introduction

6.2 *dlist* Class

VASUDEVAMURTHY

Chapter 7

Stack Queue and Deque

7.1 Introduction

7.2 *Stack Class Using Python List*

Stack using Python List
Note Stack has infinite size

```
class Stack(self):
    def __init__(self):
        self._a = []
ALL operation must be CONSTANT

    def push(self, T):
    def pop(self)->'T':
        #return None if empty
        #else return T

    def top(self)->'T':
        #return None if empty
        #else return T

    def empty(self)->'Bool':
    def is_full(self)->'Bool':
    def space(self)->'int':
        #Max space used in entire life
        return self._maxspace
    def __len__(self)->'int':
```

Figure 7.1: class Stack

7.3 MinStack Class Using Stack

```
MinStack using Stack
Note MinStack has infinite size
class MinStack(self):
    def __init__(self):
        #You can use only Stack
        ALL operation must be CONSTANT
    def push(self, T):

        def pop(self):
            #return None
            #Just removes the T from MinStack

        def top(self) -> 'T':
            #return None if empty
            #else return T

        def getMin(self) -> 'T':
            #return None if empty
            #else return MIN T
            #MUST BE THETA(1) Time
            #ZERO grade will be given
            #if getMIN is NOT THETA(1)
```

Figure 7.2: class MinStack

7.4 Queue Class Using Python List

Queue using Python List

Note Queue has a known size n

```
class Queue(n:'int'):
    def __init__(self):
        self._a = []
ALL operation must be CONSTANT

    def enQueue(self, T)->'bool':
        # False means Queue is full
        # True means enQueued

    def deQueue(self)->'bool':
        # does not return T
        # True: dequeued
        # False: Queue was empty

    def Front(self)->'T':
        return -1 if Queue is empty
        else return T

    def Rear(self)->'T':
        return -1 if Queue is empty
        else return T

    def isEmpty(self)->'Bool':
        pass

    def isFull(self)->'Bool':
        pass

    def __len__(self)->'int':
        pass
```

Figure 7.3: class Queue

7.5 Stack Class Using Queue

Stack using Queue

```

class StackUsingQueue(n: 'int'=100):
    def __init__(self):
        #YOU CAN USE ONLY Queue

    def push(self, T)
        #returns nothing

    def pop(self) -> 'T':
        # If empty return None
        # else T
        #WHAT IS THE COMPLEXITY?

    def top(self) -> 'T':
        return None if Queue is empty
        else return T

    def peek(self) -> 'T':
        #peek is same as top
        return self.top()
    def empty(self) -> 'Bool':

    def is_full(self) -> 'Bool':
    def space(self) -> 'int':
    def __len__(self) -> 'int':

```

Figure 7.4: class Stack using only Queue

7.6 Queue Class Using Stack

Queue using Stack

```
class QueueUsingStack():
    def __init__(self):
        #YOU CAN USE ONLY STACK

    def push(self, T)
        #returns nothing

    def pop(self)->'T':
        # If empty return None
        # else T
        #WHAT IS THE COMPLEXITY?

    def top(self)->'T':
        return None if Queue is empty
        else return T

    def peek(self)->'T':
        #peek is same as top
        return self.top()
    def empty(self)->'Bool':
        pass
    def is_full(self)->'Bool':
        pass
    def space(self)->'int':
        pass
    def __len__(self)->'int':
        pass
```

Figure 7.5: class Queue using only Stack

7.7 Deque Class

7.7. DEQUE CLASS

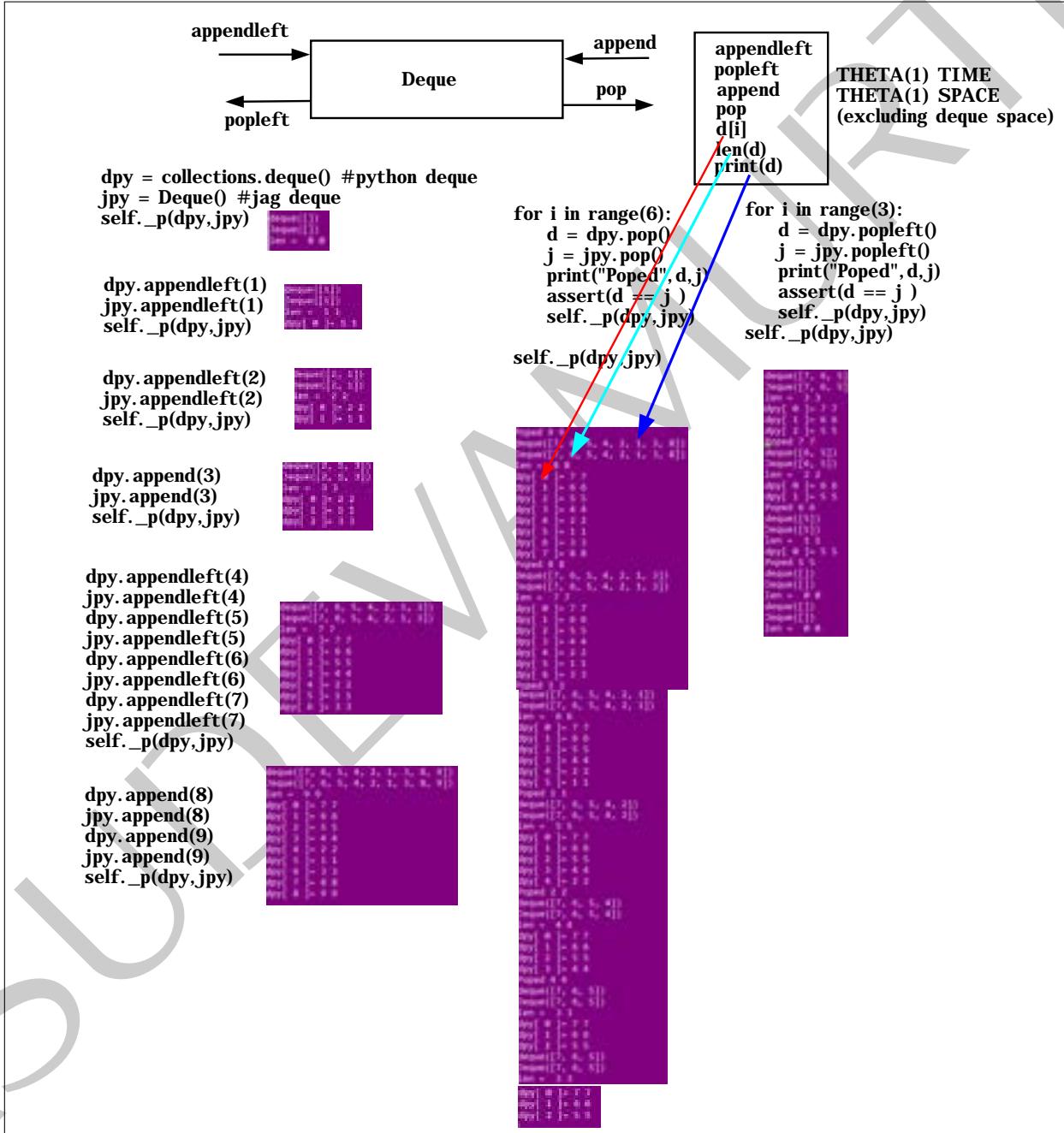
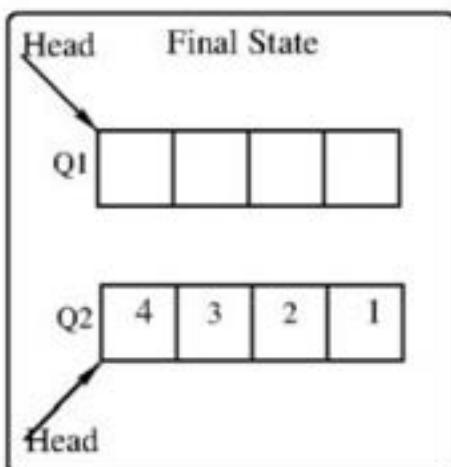
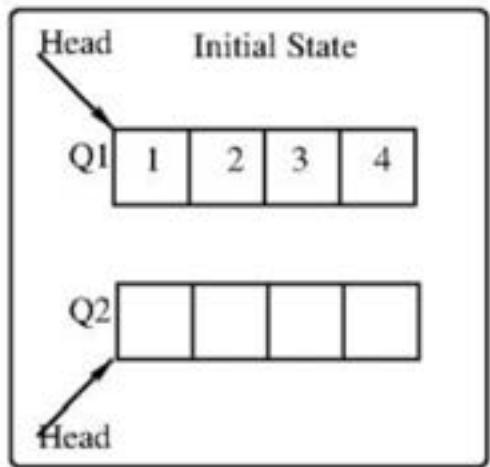


Figure 7.6: class Deque

7.8 Problem set

Problem 7.8.1. Answer the questions below by hand and post the scanned document on Canvas.

Consider the queues Q1 containing four elements and Q2 containing none (shown as the Initial State in the figure). The only operations allowed on these two queues are *Enqueue(Q, element)* and *Dequeue(Q)*. The minimum number of Enqueue operations on Q1 required to place the elements of Q1 in Q2 in reverse order (shown as the Final State in the figure) without using any additional storage is _____.



1

Must show each step of the algorithm through pictures

```
s = QueueUsingStack()
for i in range(0, 20, 2):
    s.push(i)
    s.push(i+1)
    print(s.peek(), " ", end='') WRITE ANS
    s.pop()
print()
```

2

Must show each step of the algorithm through pictures

```
s = StackUsingQueue()
for i in range(8):
    s.push(i)

for i in range(6):
    if (show):
        print(s.top(), " ", end='')
    s.pop()
if (show):
    print()
```

3

**Show through figures the value of m
You must show how you can get m in constant time**

```
s = MinStack()
m = s.getMin()
WHAT IS m
s.push(-2)
s.push(0)
s.push(-3)
m = s.getMin()
What is m
s.pop()
m = s.top()
what is m
m = s.getMin()
What is m
```

4

Problem 7.8.2. Implement stack and queue as explained in 7.2, 7.3, 7.4, 7.5, and 7.6. All tests must pass from Notebook L4list.ipynb. The expected output is as follows:

```
3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Testing stack using_python_list_of_unknown_size
Testing on 5 elements started using_python_list_of_unknown_size
4 3 2 1
Testing on 5 elements Passed
Testing on 5 elements started using_python_list_of_unknown_size
4 3 2 1 0
Testing on 5 elements Passed
Testing on 100 elements started using_python_list_of_unknown_size
99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79
Testing on 100 elements Passed
Testing on 500000 elements started using_python_list_of_unknown_size
Testing on 500000 elements Passed
Testing on 500001 elements started using_python_list_of_unknown_size
Testing on 500001 elements Passed
All the above function must have O(1) time and O(1) space for A grade
s = empty
after adding 6 = 6
after removing front = empty
after adding 5 = 5
after removing front = empty
after removing front = empty
s = empty
after adding 0 : 0
after adding 1 : 0 1
after adding 2 : 0 1 2
after adding 3 : 0 1 2 3
after removing front = 1 2 3
after adding 100 : 1 2 3 100
after removing front = 2 3 100
after adding 101 : 2 3 100 101
after removing front = 3 100 101
after adding 102 : 3 100 101 102
after removing front = 100 101 102
after adding 103 : 100 101 102 103
after removing front = 101 102 103
after adding 104 : 101 102 103 104
```

7.8. PROBLEM SET

```
after removing front = 102 103 104
after adding 105 : 102 103 104 105
after removing front = 103 104 105
after adding 106 : 103 104 105 106
after removing front = 104 105 106
after adding 107 : 104 105 106 107
after removing front = 105 106 107
after adding 108 : 105 106 107 108
after removing front = 106 107 108
after adding 109 : 106 107 108 109
after removing front = 107 108 109
after adding 110 : 107 108 109 110
----- removing 2 and adding 2 -----
107 108 109 110
after removing 2 front = 109 110
after adding 101 102 : 109 110 101 102
after removing 2 front = 101 102
after adding 102 103 : 101 102 102 103
after removing 2 front = 102 103
after adding 103 104 : 102 103 103 104
after removing 2 front = 103 104
after adding 104 105 : 103 104 104 105
after removing 2 front = 104 105
after adding 105 106 : 104 105 105 106
after removing 2 front = 105 106
after adding 106 107 : 105 106 106 107
after removing 2 front = 106 107
after adding 107 108 : 106 107 107 108
after removing 2 front = 107 108
after adding 108 109 : 107 108 108 109
after removing 2 front = 108 109
after adding 109 110 : 108 109 109 110
after removing 2 front = 109 110
after adding 110 111 : 109 110 110 111
----- removing 3 and adding 3 -----
109 110 110 111
after removing 3 front = 111
after adding 101 102 103 : 111 101 102 103
after removing 3 front = 103
after adding 102 103 104 : 103 102 103 104
after removing 3 front = 104
```

CHAPTER 7. STACK QUEUE AND DEQUE

```
after adding 103 104 105 : 104 103 104 105
after removing 3 front = 105
after adding 104 105 106 : 105 104 105 106
after removing 3 front = 106
after adding 105 106 107 : 106 105 106 107
after removing 3 front = 107
after adding 106 107 108 : 107 106 107 108
after removing 3 front = 108
after adding 107 108 109 : 108 107 108 109
after removing 3 front = 109
after adding 108 109 110 : 109 108 109 110
after removing 3 front = 110
after adding 109 110 111 : 110 109 110 111
after removing 3 front = 111
after adding 110 111 112 : 111 110 111 112
Testing on 5 elements started
Number of elements after enqueue 5
0 1 2 3
Number of elements after 1
Testing on 5 elements Passed
Testing on 5 elements started
Number of elements after enqueue 5
0 1 2 3 4
Number of elements after 0
Testing on 5 elements Passed
Testing on 500000 elements started
Number of elements after enqueue 500000
Number of elements after 8602
Testing on 500000 elements Passed
Testing on 500001 elements started
Number of elements after enqueue 500001
Number of elements after 0
Testing on 500001 elements Passed
All the above function must have O(1) time and O(1) space for A grade
Testing stack using_queue
Testing on 5 elements started using_queue
4 3 2 1
Testing on 5 elements Passed
Testing on 5 elements started using_queue
4 3 2 1 0
Testing on 5 elements Passed
```

7.8. PROBLEM SET

Testing on 100 elements started using_queue

99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84

Testing on 100 elements Passed

test_queue_using_stack Started

Number of elements after Push 5

0 1 2 3

Number of elements after 1

0 1 2 3 4 5 6 7 8 9

Number of elements after 10

test MinStack Started

Minimum 0

Minimum -100

All my test passed

NOW ALL THESE LEETCODE MUST PASS FOR FULL GRADE

622. Design Circular Queue: <https://leetcode.com/problems/design-circular-queue/>

225. Implement Stack using Queues: <https://leetcode.com/problems/implement-stack-using-queues/>

232. Implement Queue using Stacks: <https://leetcode.com/problems/implement-queue-using-stacks/>

155. Min Stack: <https://leetcode.com/problems/min-stack/>

CHAPTER 7. STACK QUEUE AND DEQUE

Now use the same Notebook **L4list.ipynb**. All Leetcode must pass as shown below. Note that NO code has to be written. If my Notebook **L4list.ipynb** passes, Leetcode must pass.

Problem 7.8.3. Implement deque as explained 7.7. The expected output is as follows:

VASUDEVAMURTHY

Chapter 8

Searching and sorting

8.1 Introduction

8.2 Selection sort

8.2.1 Array based selection sort

8.2. SELECTION SORT

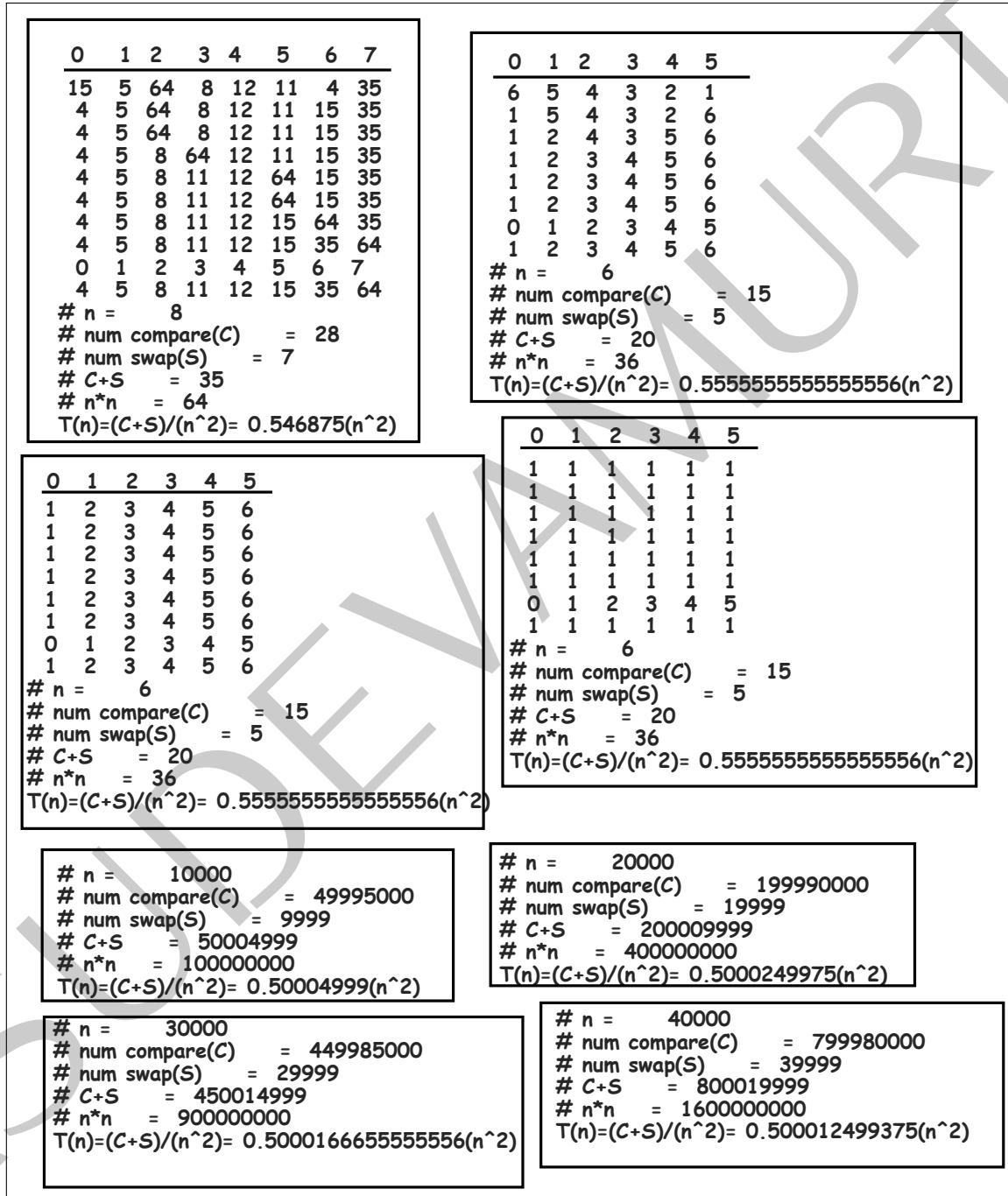


Figure 8.1: Animation of selection sort algorithm

8.2.2 Linked list based selection sort

8.3 Insertion sort

8.3.1 Array based insertion sort

```

0 1
15 5
Iteration 1: Inserting item 5 which is in position 1 to new position 0
0 1
5 15
# n = 2
# num compare(C) = 1
# num swap(S) = 1
# C+S = 2
# n*n = 4
T(n)=(C+S)/(n^2)= 0.5(n^2)

0 1 2 3 4 5 6 7
15 5 64 8 12 11 4 35
Iteration 1: Inserting item 5 which is in position 1 to new position 0
0 1 2 3 4 5 6 7
5 15 64 8 12 11 4 35
Iteration 2: Inserting item 64 which is in position 2 to new position 2
0 1 2 3 4 5 6 7
5 15 64 8 12 11 4 35
Iteration 3: Inserting item 8 which is in position 3 to new position 1
0 1 2 3 4 5 6 7
5 8 15 64 12 11 4 35
Iteration 4: Inserting item 12 which is in position 4 to new position 2
0 1 2 3 4 5 6 7
5 8 12 15 64 11 4 35
Iteration 5: Inserting item 11 which is in position 5 to new position 2
0 1 2 3 4 5 6 7
5 8 11 12 15 64 4 35
Iteration 6: Inserting item 4 which is in position 6 to new position 0
0 1 2 3 4 5 6 7
4 5 8 11 12 15 64 35
Iteration 7: Inserting item 35 which is in position 7 to new position 6
0 1 2 3 4 5 6 7
4 5 8 11 12 15 35 64
# n = 8
# num compare(C) = 20
# num swap(S) = 15
# C+S = 35
# n*n = 64
T(n)=(C+S)/(n^2)= 0.546875(n^2)

```

Figure 8.2: Animation of insertion sort algorithm

8.3. INSERTION SORT

```

0 1 2 3 4 5
1 2 3 4 5 6
Iteration 1: Inserting item 2 which is in position 1 to new position 1
0 1 2 3 4 5
1 2 3 4 5 6
Iteration 2: Inserting item 3 which is in position 2 to new position 2
0 1 2 3 4 5
1 2 3 4 5 6
Iteration 3: Inserting item 4 which is in position 3 to new position 3
0 1 2 3 4 5
1 2 3 4 5 6
Iteration 4: Inserting item 5 which is in position 4 to new position 4
0 1 2 3 4 5
1 2 3 4 5 6
Iteration 5: Inserting item 6 which is in position 5 to new position 5
0 1 2 3 4 5
1 2 3 4 5 6
# n = 6
# num compare(C) = 5
# num swap(S) = 0
# C+S = 5
# n*n = 36
T(n)=(C+S)/(n^2)= 0.138888888888889(n^2)

0 1 2 3 4 5
1 1 1 1 1 1
Iteration 1: Inserting item 1 which is in position 1 to new position 1
0 1 2 3 4 5
1 1 1 1 1 1
Iteration 2: Inserting item 1 which is in position 2 to new position 2
0 1 2 3 4 5
1 1 1 1 1 1
Iteration 3: Inserting item 1 which is in position 3 to new position 3
0 1 2 3 4 5
1 1 1 1 1 1
Iteration 4: Inserting item 1 which is in position 4 to new position 4
0 1 2 3 4 5
1 1 1 1 1 1
Iteration 5: Inserting item 1 which is in position 5 to new position 5
0 1 2 3 4 5
1 1 1 1 1 1
# n = 6
# num compare(C) = 5
# num swap(S) = 0
# C+S = 5
# n*n = 36
T(n)=(C+S)/(n^2)= 0.138888888888889(n^2)

# n = 10000
# num compare(C) = 25117244
# num swap(S) = 25107251
# C+S = 50224495
# n*n = 100000000
T(n)=(C+S)/(n^2)= 0.50224495(n^2)

# n = 20000
# num compare(C) = 100147440
# num swap(S) = 100127451
# C+S = 200274891
# n*n = 400000000
T(n)=(C+S)/(n^2)= 0.5006872275(n^2)

# n = 30000
# num compare(C) = 225744013
# num swap(S) = 225714027
# C+S = 451458040
# n*n = 900000000
T(n)=(C+S)/(n^2)= 0.5016200444444444(n^2)

# n = 40000
# num compare(C) = 400988352
# num swap(S) = 400948361
# C+S = 801936713
# n*n = 1600000000
T(n)=(C+S)/(n^2)= 0.501210445625(n^2)

```

Figure 8.3: Animation of insertion sort algorithm

8.3.2 Linked list based insertion sort

8.4 Binary search

8.4.1 Array based binary search

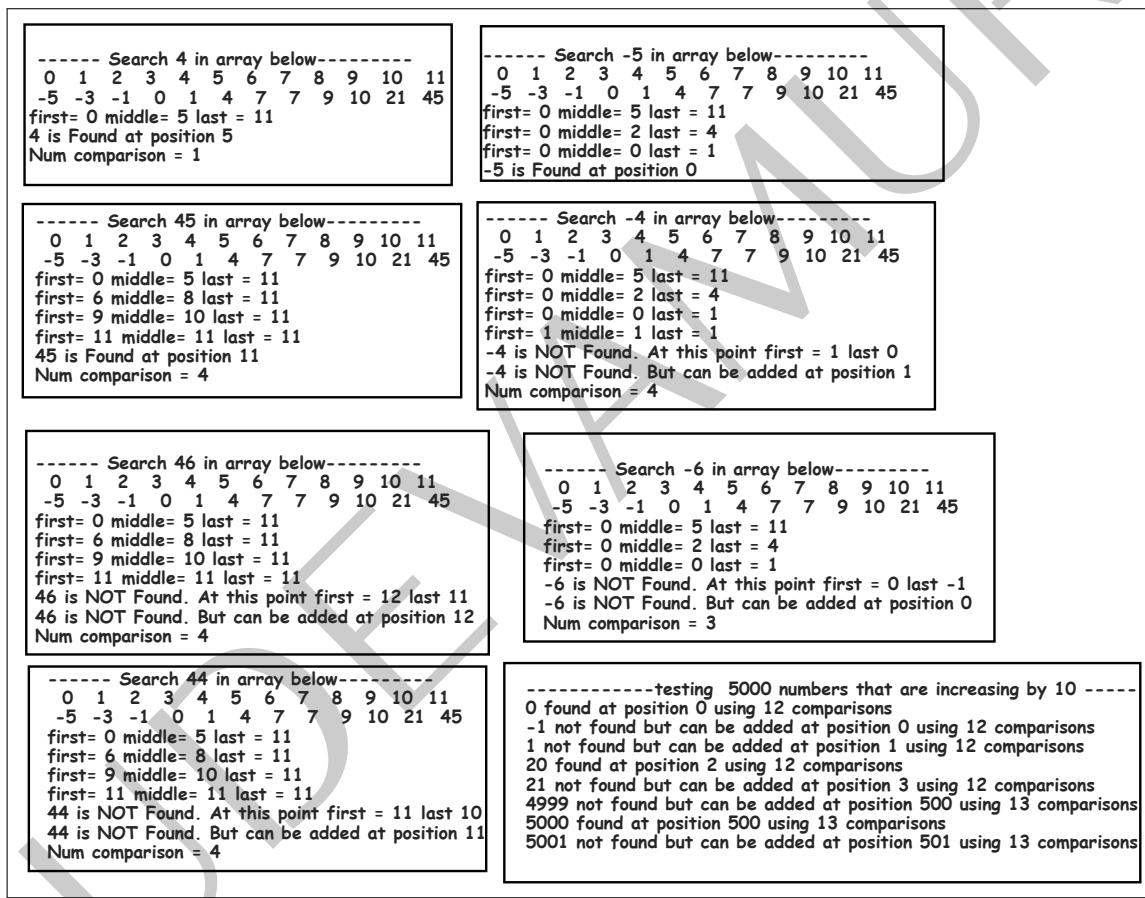


Figure 8.4: Animation of binary search algorithm

8.4. BINARY SEARCH

```

class ArrayBinarySearch {

    public static int[] bsearch(int [] a, int first, int last, int toFind, boolean display){
        if (display) {
            System.out.println("----- Search " + toFind + " in array below-----");
        }
        IntUtil u = new IntUtil();
        int n = a.length ;
        int numCompare = 0 ;
        u.assertAscending(a,first,last) ;
        if (display) {
            u.pLn(n) ;
            u.pLn(a,0,n);
        }
        int [] ans = new int [3] ; //0 found/not found, 1 position found/inserted 2 num compare
        while (first <= last) { //Even one element
            int middle = (last - first)/2 + first ;
            if (display) {
                System.out.println("first= " + first + " middle= " + middle + " last = " + last);
            }
            numCompare++;
            ans[2] = numCompare ;
            if (a[middle] == toFind) {
                ans[0] = 1 ; //answer is found
                ans[1] = middle ; //Pos
                if (display) {
                    System.out.println(toFind + " is Found at position " + ans[1]);
                    System.out.println("Num comparison = " + numCompare) ;
                }
                return ans ;
            }
            if (toFind < a[middle]) {
                last = middle - 1 ;
            }else {
                first = middle + 1 ;
            }
        }
        //Now search is over
        //middle is the index above or below target
        if (display) {
            System.out.println(toFind + " is NOT Found. At this point first = " + first + " last " + last);
        }
        ans[0] = 0 ; //answer is Not found
        assert(first > last) ;
        ans[1] = first ;
        assert(ans[1] >= 0 && ans[1] <= last + 1); //can be added at 0 or last +1
        if (display) {
            System.out.println(toFind + " is NOT Found. But can be added at position " + ans[1]);
            System.out.println("Num comparison = " + numCompare) ;
            System.out.println("-----");
        }
        return ans ;
    }

    public static int[] bsearch(int [] a, int toFind, boolean display){
        return (bsearch(a,0,a.length-1,toFind,display));
    }
}

```

$T(n) = T(n/2) + C$
 $T(n)$ at $T(1)$
 $\frac{n}{2} = 1$
 $n = 2$
 $k = \log_2 n = \Theta(\log_2 n)$
 $\frac{n}{2^k} = 1$
 $n = 2^k$
 $k = \log_2 n$

$T(n/2)$
 $T(n/4)$
 $T(n/8)$
 $T(1)$

----- Search - 4 in array below -----

0	1	2	3	4	5	6	7	8	9	10	11
-5	-3	-1	0	1	4	7	7	9	10	21	45

first= 0 middle= 5 last = 11
 first= 0 middle= 2 last = 4
 first= 0 middle= 0 last = 1
 first= 1 middle= 1 last = 1

-4 is NOT Found. At this point first = 1 last 0
 -4 is NOT Found. But can be added at position 1
 Num comparison = 4

Figure 8.5: Binary search from class ArrayBinarySearch

8.4.2 Linked list based binary search

8.5 Insertion binary sort

8.5.1 Array based insertion binary sort

```

0 1
15 5
Iteration 1: Inserting item 5 which is in position 1 to new position 0
0 1
5 15
# n = 2
# num compare(C) = 1
# num swap(S) = 1
# C+S = 2
# n*n = 4
T(n)=(C+S)/(n^2)= 0.5(n^2)

0 1 2 3 4 5 6 7
15 5 64 8 12 11 4 35
Iteration 1: Inserting item 5 which is in position 1 to new position 0
0 1 2 3 4 5 6 7
5 15 64 8 12 11 4 35
Iteration 2: Inserting item 64 which is in position 2 to new position 2
0 1 2 3 4 5 6 7
5 15 64 8 12 11 4 35
Iteration 3: Inserting item 8 which is in position 3 to new position 1
0 1 2 3 4 5 6 7
5 8 15 64 12 11 4 35
Iteration 4: Inserting item 12 which is in position 4 to new position 2
0 1 2 3 4 5 6 7
5 8 12 15 64 11 4 35
Iteration 5: Inserting item 11 which is in position 5 to new position 2
0 1 2 3 4 5 6 7
5 8 11 12 15 64 4 35
Iteration 6: Inserting item 4 which is in position 6 to new position 0
0 1 2 3 4 5 6 7
4 5 8 11 12 15 64 35
Iteration 7: Inserting item 35 which is in position 7 to new position 6
0 1 2 3 4 5 6 7
4 5 8 11 12 15 35 64
# n = 8
# num compare(C) = 15
# num swap(S) = 15
# C+S = 30
# n*n = 64
T(n)=(C+S)/(n^2)= 0.46875(n^2)

```

Figure 8.6: Animation of insertion binary sort algorithm

8.5. INSERTION BINARY SORT

```

0 1 2 3 4 5
6 5 4 3 2 1
Iteration 1: Inserting item 5 which is in position 1 to new position 0
0 1 2 3 4 5
5 6 4 3 2 1
Iteration 2: Inserting item 4 which is in position 2 to new position 0
0 1 2 3 4 5
4 5 6 3 2 1
Iteration 3: Inserting item 3 which is in position 3 to new position 0
0 1 2 3 4 5
3 4 5 6 2 1
Iteration 4: Inserting item 2 which is in position 4 to new position 0
0 1 2 3 4 5
2 3 4 5 6 1
Iteration 5: Inserting item 1 which is in position 5 to new position 0
0 1 2 3 4 5
1 2 3 4 5 6
# n = 6
# num compare(C) = 8
# num swap(S) = 15
# C+S = 23
# n*n = 36
T(n)=(C+S)/(n^2)= 0.6388888888888888(n^2)

0 1 2 3 4 5
1 2 3 4 5 6
Iteration 1: Inserting item 2 which is in position 1 to new position 1
0 1 2 3 4 5
1 2 3 4 5 6
Iteration 2: Inserting item 3 which is in position 2 to new position 2
0 1 2 3 4 5
1 2 3 4 5 6
Iteration 3: Inserting item 4 which is in position 3 to new position 3
0 1 2 3 4 5
1 2 3 4 5 6
Iteration 4: Inserting item 5 which is in position 4 to new position 4
0 1 2 3 4 5
1 2 3 4 5 6
Iteration 5: Inserting item 6 which is in position 5 to new position 5
0 1 2 3 4 5
1 2 3 4 5 6
# n = 6
# num compare(C) = 11
# num swap(S) = 0
# C+S = 11
T(n)=(C+S)/(n)= 1.8333333333333333(n)
# nlogn = 15.509775004326936
T(n)=(C+S)/(nlogn)= 0.7092301465966596(n*logn)
# n*n = 36
T(n)=(C+S)/(n^2)= 0.3055555555555556(n^2)

# n = 10000
# num compare(C) = 119012
# num swap(S) = 25198695
# C+S = 25317707
# n*n = 100000000
T(n)=(C+S)/(n^2)= 0.25317707(n^2)

# n = 20000
# num compare(C) = 258052
# num swap(S) = 99765319
# C+S = 100023371
# n*n = 400000000
T(n)=(C+S)/(n^2)= 0.2500584275(n^2)

# n = 30000
# num compare(C) = 404652
# num swap(S) = 225286768
# C+S = 225691420
# n*n = 900000000
T(n)=(C+S)/(n^2)= 0.25076824(n^2)

# n = 40000
# num compare(C) = 555973
# num swap(S) = 398613273
# C+S = 399169246
# n*n = 1600000000
T(n)=(C+S)/(n^2)= 0.24948077875(n^2)

```

Figure 8.7: Animation of insertion binary sort algorithm

```

class ArrayInsertionBinarySort extends ArraySort{
    @Override
    protected void sort(boolean ascend) {
        int n = a.length ;
        for (int i = 1 ; i < n; ++i) { O(n)
            //We are inserting ith item O(n(logn + n)) = O(n2)
            int item = a[i] ;
            // By definition 0 to (i-1) array is sorted
            int [] ans = ArrayBinarySearch.bsearch(a, 0, (i-1), item, false) ;
            if (display) {
                u.pLn(n) ;
                u.pLn(a, 0, n);
                String s = "Iteration " + i + ":" ;
                System.out.println(s + "Inserting item " + item + " which is in position " + i + " to new position " + ans[1]) ;
            }
            numCompare = numCompare + ans[2] ;
            //int [] ans = new int [3] ; //0 found/not found, 1 position found/inserted 2 num compare
            int pos = ans[1] ; //Can be inserted here
            //Now shift right
            for (int j = i-1; j >= pos; --j) { O(n)
                a[j+1] = a[j] ;
                numSwap++ ;
            }
            a[pos] = item ; //Inserted in correct position
        }
    }

    public static void main(String[] args) {
        System.out.println("InsertionBinarySort.java");
        ArrayInsertionBinarySort u = new ArrayInsertionBinarySort() ;
        u.testBench();
    }
}

```

if comparison cost is much greater than swapping

Figure 8.8: InsertionSort versus BinaryInsertionSort

8.6. MERGE SORT

8.5.2 Linked list based insertion binary sort

8.6 Merge sort

8.6.1 Array based merge sort

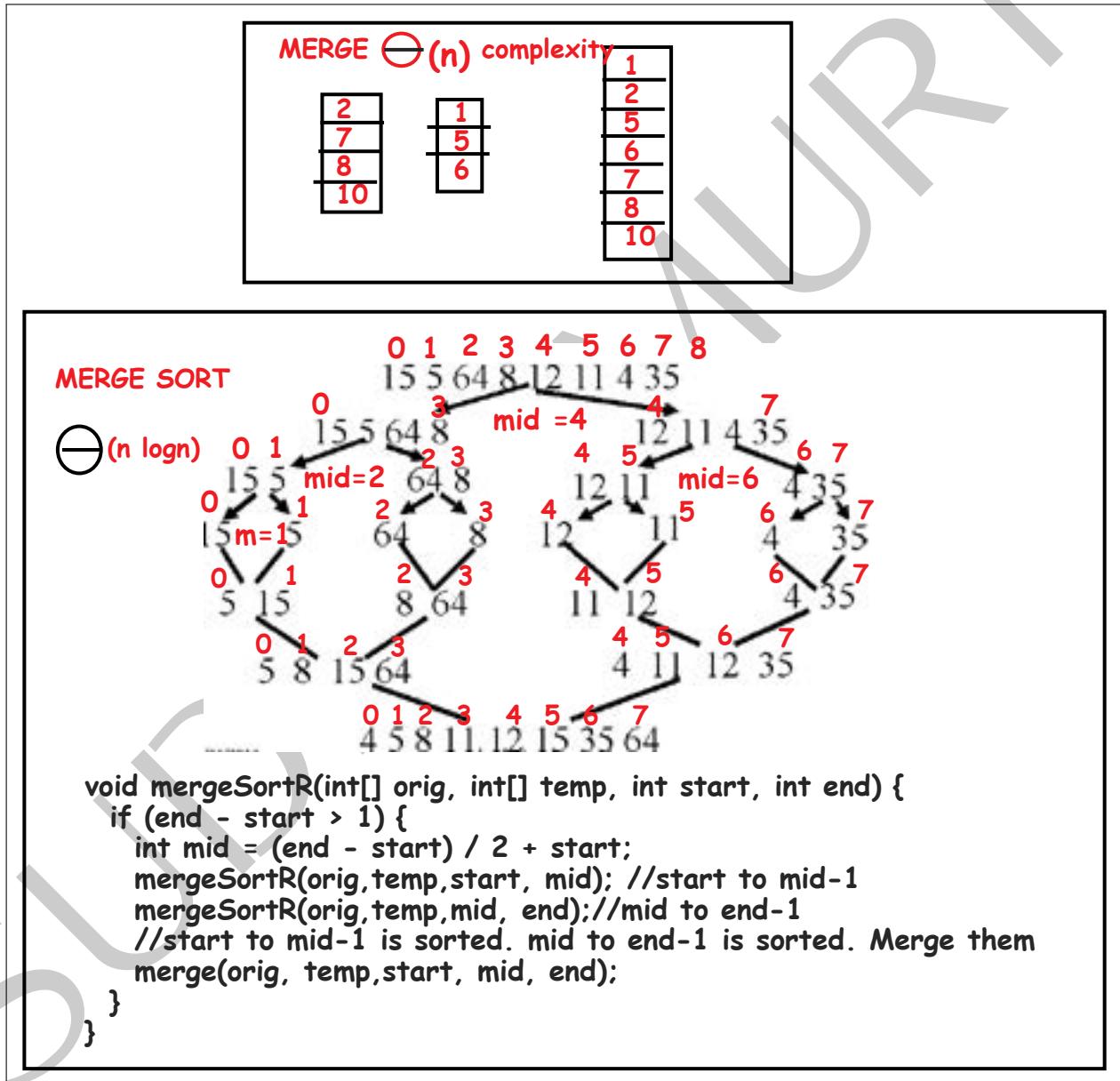


Figure 8.9: Merge and merge sort

8.6. MERGE SORT

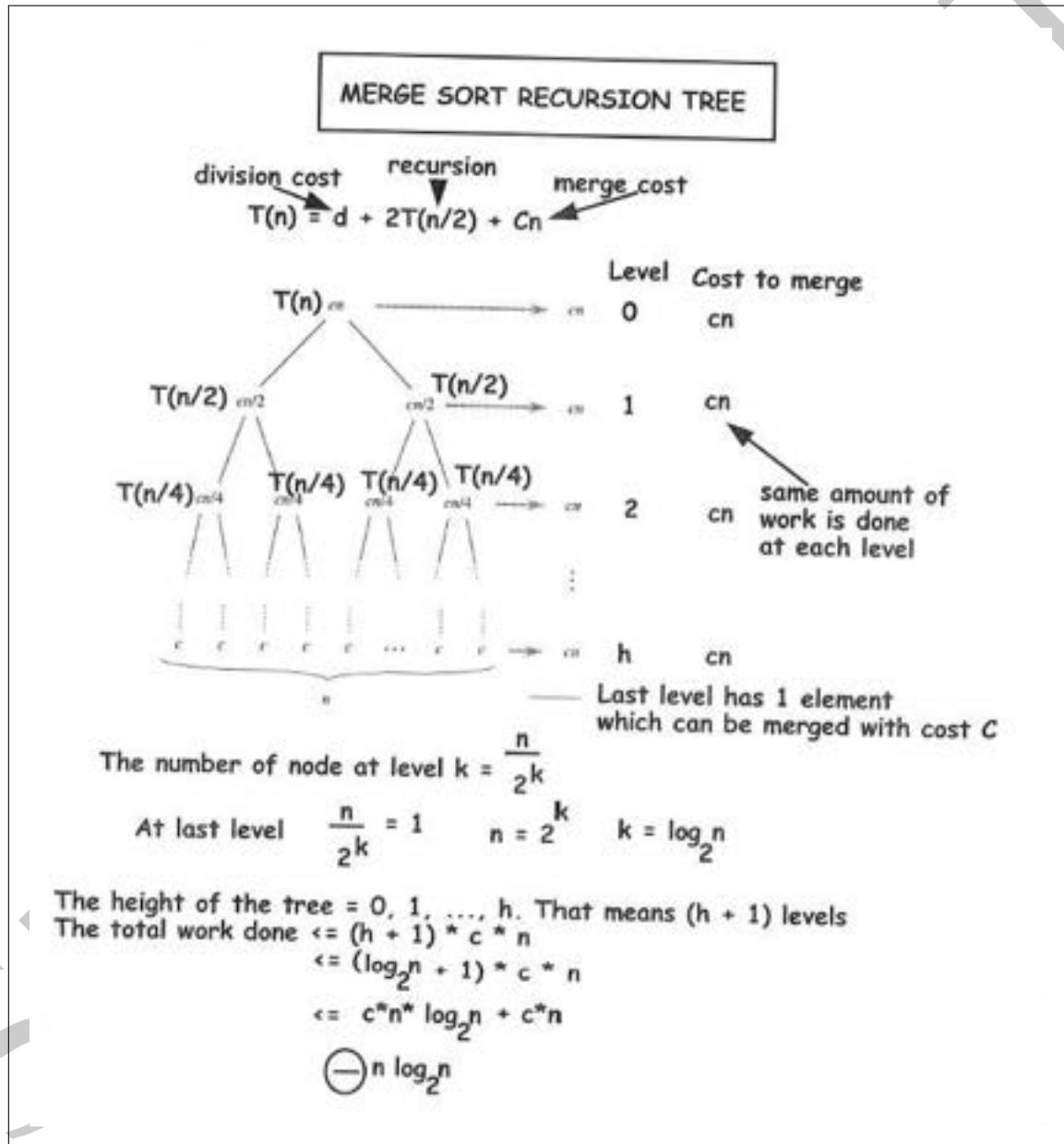


Figure 8.10: Complexity of merge sort

What happens if you can invent merge routine of $O(1)$ complexity?

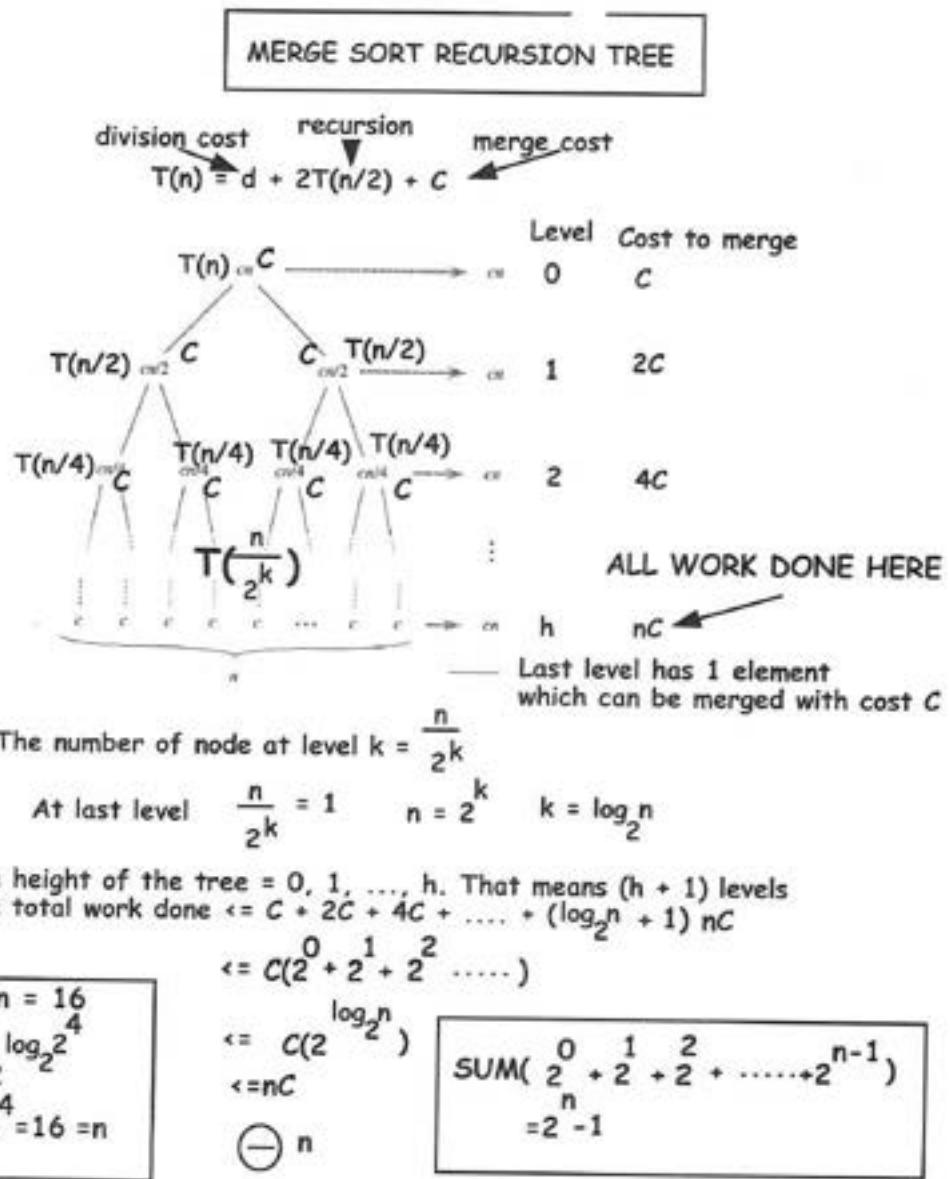


Figure 8.11: Complexity of merge sort if we invent $O(1)$ merge routine

8.6. MERGE SORT

What happens if merge routine is of complexity $O(n^2)$?

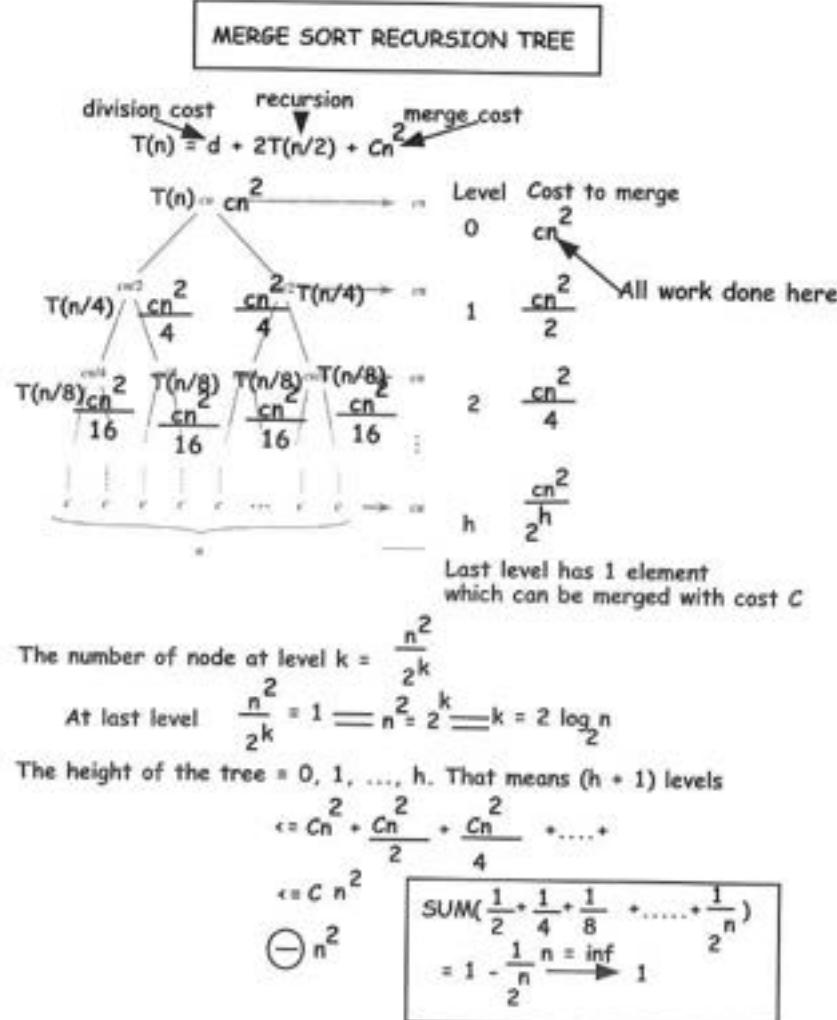


Figure 8.12: Complexity of merge sort if we use $O(n^2)$ merge routine

```
class ArrayMergeSort extends ArraySort{
```

$\Theta(n)$

```
    private void merge(int start, int mid, int end, int [] t) {
        int i = start ;
        int j = mid ;
        int k = 0 ;

        while (i < mid && j < end) {
            numCompare++ ;
            numSwap++ ;
            if (a[i] < a[j]) {
                t[k++] = a[i++] ;
            }else {
                t[k++] = a[j++] ;
            }
        }
        while (i < mid) {
            numSwap++ ;
            t[k++] = a[i++] ;
        }
        while (j < end) {
            numSwap++ ;
            t[k++] = a[j++] ;
        }
        // put back
        for (int f = 0; f < k; ++f) {
            numSwap++ ;
            int p = start + f ;
            int v = t[f] ;
            a[p] = v ;
        }
    }
```

start=4, end=8
mid=6

$t[0]$

$a[4..7]$

$a[4..7]$

$a[0..3]$

$a[0..7]$

```
private void mergeSortR(int start, int end, int [] t) {
    int mid = (end - start) / 2 + start ;
    if (end - start > 1) { //More than 1 element
        numRecursion++ ;
        mergeSortR(start,mid,t); //start to mid-1
        numRecursion++ ;
        mergeSortR(mid,end,t); //mid to end-1
        //start to mid-1 is sorted. mid to end-1 is sorted. Merge them
        merge(start,mid,end,t);
    }
}
```

$\Theta(n \log n)$

@Override

```
protected void sort(boolean ascend) {
    int [] t = new int[a.length] ;
    mergeSortR(0,a.length,t) ;
```

TEMP ARRAY REQUIRED

Figure 8.13: Sort routine from class ArrayMergeSortSort

8.6. MERGE SORT

# n = 10000 # num compare(C) = 120488 # num swap(S) = 267232 # C+S = 387720 $T(n)=(C+S)/(n\log n)= 2.91788374797097(n*\log n)$	# n = 20000 # num compare(C) = 260993 # num swap(S) = 574464 # C+S = 835457 $T(n)=(C+S)/(n\log n)= 2.9236905734322507(n*\log n)$
# n = 30000 # num compare(C) = 408708 # num swap(S) = 894464 # C+S = 1303172 $T(n)=(C+S)/(n\log n)= 2.9207299303160923(n*\log n)$	# n = 40000 # num compare(C) = 561684 # num swap(S) = 1228928 # C+S = 1790612 $T(n)=(C+S)/(n\log n)= 2.9281882657527665(n*\log n)$
-----testing 25001 SORTED ASCENDING numbers----- -----SORT1 Start----- # n = 25001 # num compare(C) = 178762 # num swap(S) = 734496 # C+S = 913258 $T(n)=(C+S)/(n\log n)= 2.5003157760604076(n*\log n)$	

Figure 8.14: Run time complexity of class `ArrayMergeSortSort`

8.6.2 Linked list based merge sort

8.7 Quick sort

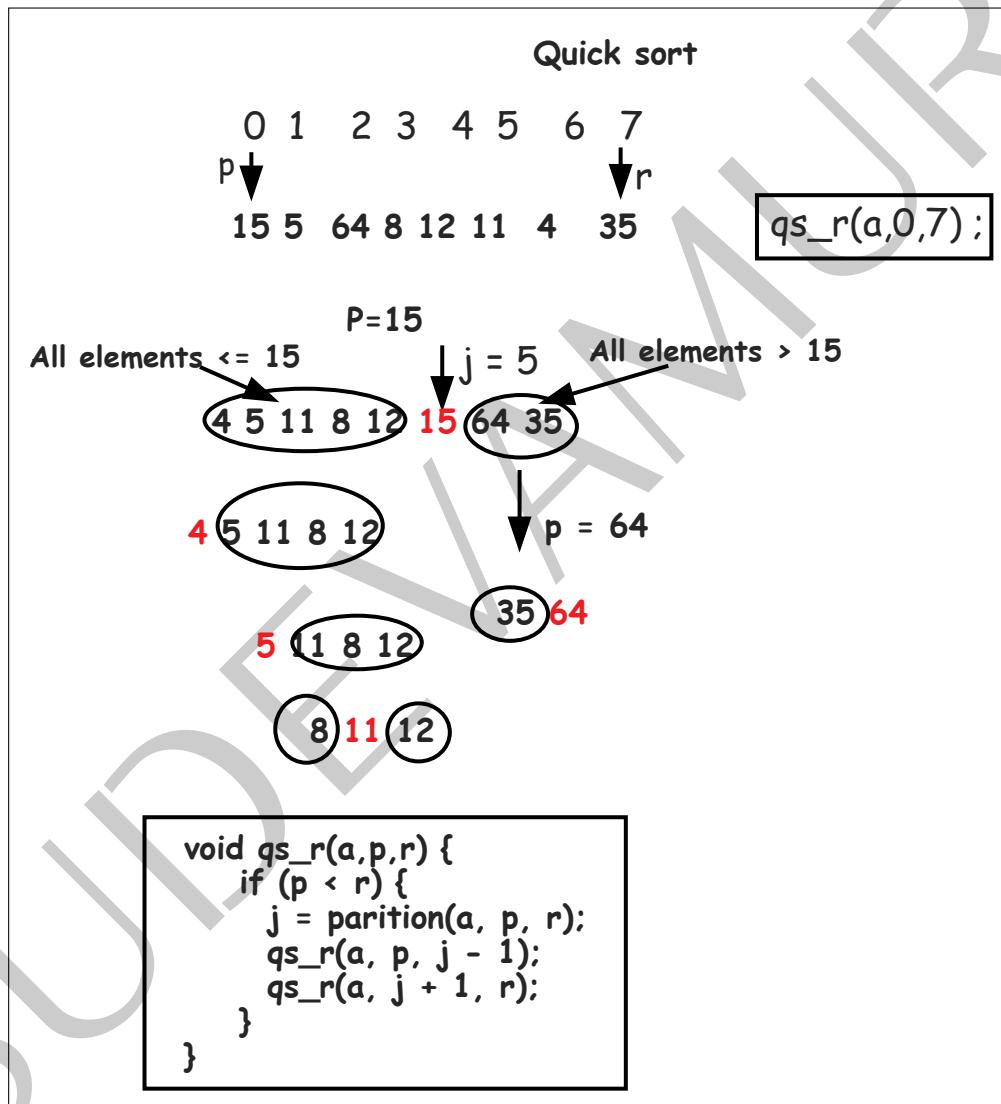


Figure 8.15: Partition and quick sort

8.7.1 Array based quick sort

8.7. QUICK SORT

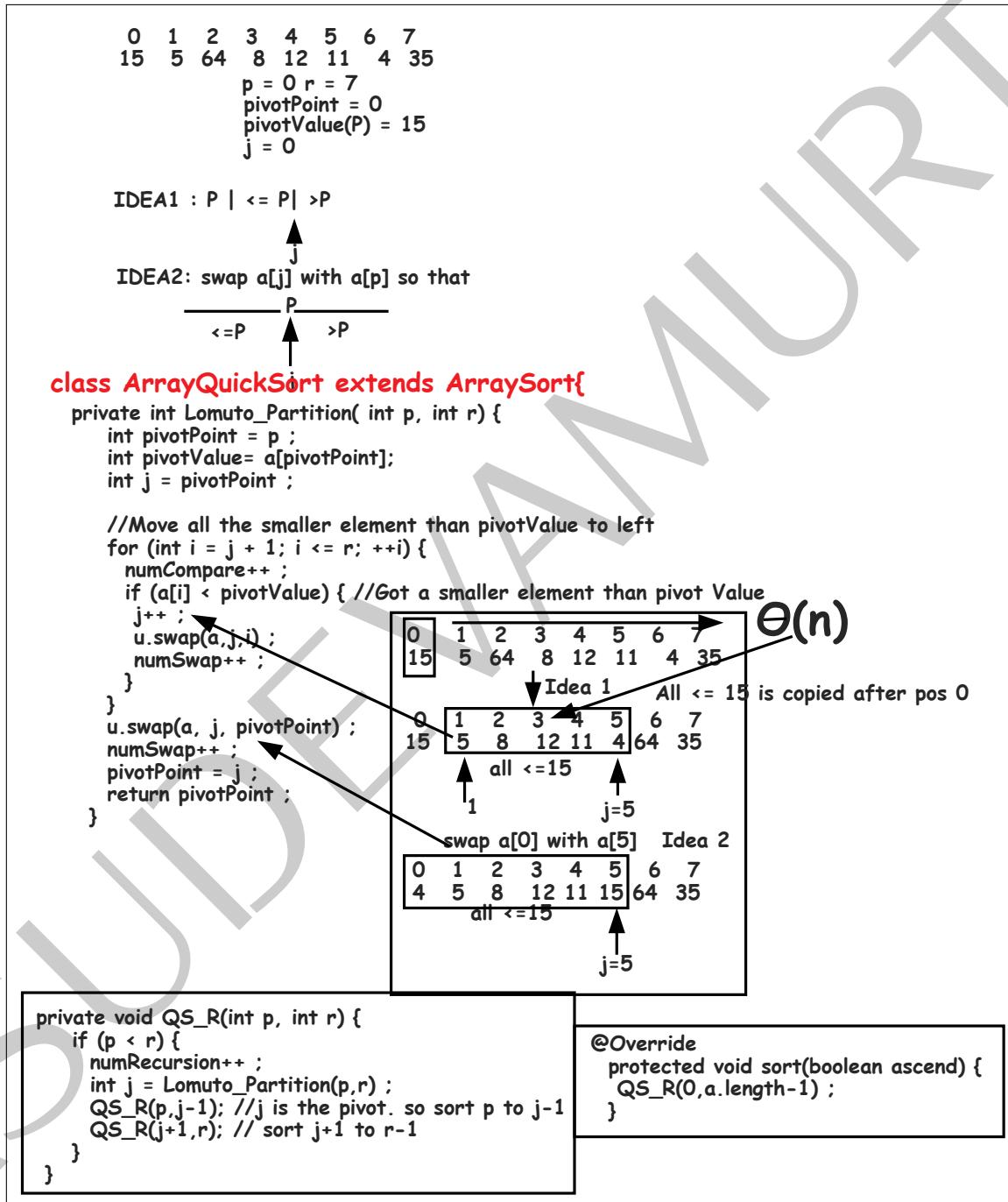


Figure 8.16: Lomuto-Partition and quick sort

CHAPTER 8. SEARCHING AND SORTING

<p>Sort from 0 to 7 using 15 as pivot Value</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> <tr><td>15</td><td>5</td><td>64</td><td>8</td><td>12</td><td>11</td><td>4</td><td>35</td></tr> </table> <p>Pivot Value = 15. Pivot position = 5</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> <tr><td>4</td><td>5</td><td>8</td><td>12</td><td>11</td><td>15</td><td>64</td><td>35</td></tr> </table>	0	1	2	3	4	5	6	7	15	5	64	8	12	11	4	35	0	1	2	3	4	5	6	7	4	5	8	12	11	15	64	35	<p>Sort from 0 to 5 using 6 as pivot Value</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr> </table> <p>Pivot Value = 6. Pivot position = 5</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>1</td><td>5</td><td>4</td><td>3</td><td>2</td><td>6</td></tr> </table>	0	1	2	3	4	5	6	5	4	3	2	1	0	1	2	3	4	5	1	5	4	3	2	6
0	1	2	3	4	5	6	7																																																		
15	5	64	8	12	11	4	35																																																		
0	1	2	3	4	5	6	7																																																		
4	5	8	12	11	15	64	35																																																		
0	1	2	3	4	5																																																				
6	5	4	3	2	1																																																				
0	1	2	3	4	5																																																				
1	5	4	3	2	6																																																				
<p>Sort from 0 to 4 using 4 as pivot Value</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr><td>4</td><td>5</td><td>8</td><td>12</td><td>11</td></tr> </table> <p>Pivot Value = 4. Pivot position = 0</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> <tr><td>4</td><td>5</td><td>8</td><td>12</td><td>11</td><td>15</td><td>64</td><td>35</td></tr> </table>	0	1	2	3	4	4	5	8	12	11	0	1	2	3	4	5	6	7	4	5	8	12	11	15	64	35	<p>Sort from 0 to 4 using 1 as pivot Value</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr><td>1</td><td>5</td><td>4</td><td>3</td><td>2</td></tr> </table> <p>Pivot Value = 1. Pivot position = 0</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>1</td><td>5</td><td>4</td><td>3</td><td>2</td><td>6</td></tr> </table>	0	1	2	3	4	1	5	4	3	2	0	1	2	3	4	5	1	5	4	3	2	6								
0	1	2	3	4																																																					
4	5	8	12	11																																																					
0	1	2	3	4	5	6	7																																																		
4	5	8	12	11	15	64	35																																																		
0	1	2	3	4																																																					
1	5	4	3	2																																																					
0	1	2	3	4	5																																																				
1	5	4	3	2	6																																																				
<p>Sort from 1 to 4 using 5 as pivot Value</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr><td>4</td><td>5</td><td>8</td><td>12</td><td>11</td></tr> </table> <p>Pivot Value = 5. Pivot position = 1</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> <tr><td>4</td><td>5</td><td>8</td><td>12</td><td>11</td><td>15</td><td>64</td><td>35</td></tr> </table>	0	1	2	3	4	4	5	8	12	11	0	1	2	3	4	5	6	7	4	5	8	12	11	15	64	35	<p>Sort from 1 to 4 using 5 as pivot Value</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr><td>1</td><td>5</td><td>4</td><td>3</td><td>2</td></tr> </table> <p>Pivot Value = 5. Pivot position = 4</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>1</td><td>2</td><td>4</td><td>3</td><td>5</td><td>6</td></tr> </table>	0	1	2	3	4	1	5	4	3	2	0	1	2	3	4	5	1	2	4	3	5	6								
0	1	2	3	4																																																					
4	5	8	12	11																																																					
0	1	2	3	4	5	6	7																																																		
4	5	8	12	11	15	64	35																																																		
0	1	2	3	4																																																					
1	5	4	3	2																																																					
0	1	2	3	4	5																																																				
1	2	4	3	5	6																																																				
<p>Sort from 2 to 4 using 8 as pivot Value</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr><td>4</td><td>5</td><td>8</td><td>12</td><td>11</td></tr> </table> <p>Pivot Value = 8. Pivot position = 2</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> <tr><td>4</td><td>5</td><td>8</td><td>12</td><td>11</td><td>15</td><td>64</td><td>35</td></tr> </table>	0	1	2	3	4	4	5	8	12	11	0	1	2	3	4	5	6	7	4	5	8	12	11	15	64	35	<p>Sort from 1 to 3 using 2 as pivot Value</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>1</td><td>2</td><td>4</td><td>3</td><td>5</td><td>6</td></tr> </table> <p>Pivot Value = 2. Pivot position = 1</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>1</td><td>2</td><td>4</td><td>3</td><td>5</td><td>6</td></tr> </table>	0	1	2	3	4	5	1	2	4	3	5	6	0	1	2	3	4	5	1	2	4	3	5	6						
0	1	2	3	4																																																					
4	5	8	12	11																																																					
0	1	2	3	4	5	6	7																																																		
4	5	8	12	11	15	64	35																																																		
0	1	2	3	4	5																																																				
1	2	4	3	5	6																																																				
0	1	2	3	4	5																																																				
1	2	4	3	5	6																																																				
<p>Sort from 3 to 4 using 12 as pivot Value</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr><td>4</td><td>5</td><td>8</td><td>12</td><td>11</td></tr> </table> <p>Pivot Value = 12. Pivot position = 4</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> <tr><td>4</td><td>5</td><td>8</td><td>11</td><td>12</td><td>15</td><td>64</td><td>35</td></tr> </table>	0	1	2	3	4	4	5	8	12	11	0	1	2	3	4	5	6	7	4	5	8	11	12	15	64	35	<p>Sort from 2 to 3 using 4 as pivot Value</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>1</td><td>2</td><td>4</td><td>3</td><td>5</td><td>6</td></tr> </table> <p>Pivot Value = 4. Pivot position = 3</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table>	0	1	2	3	4	5	1	2	4	3	5	6	0	1	2	3	4	5	1	2	3	4	5	6						
0	1	2	3	4																																																					
4	5	8	12	11																																																					
0	1	2	3	4	5	6	7																																																		
4	5	8	11	12	15	64	35																																																		
0	1	2	3	4	5																																																				
1	2	4	3	5	6																																																				
0	1	2	3	4	5																																																				
1	2	3	4	5	6																																																				
<p>Sort from 6 to 7 using 64 as pivot Value</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> <tr><td>4</td><td>5</td><td>8</td><td>11</td><td>12</td><td>15</td><td>64</td><td>35</td></tr> </table> <p>Pivot Value = 64. Pivot position = 7</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> <tr><td>4</td><td>5</td><td>8</td><td>11</td><td>12</td><td>15</td><td>35</td><td>64</td></tr> </table>	0	1	2	3	4	5	6	7	4	5	8	11	12	15	64	35	0	1	2	3	4	5	6	7	4	5	8	11	12	15	35	64	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table> <p># n = 6</p> <p># num compare(C) = 15 # num swap(S) = 14 # C+S = 29 # nlogn = 15.509775004326936 T(n)=(C+S)/(nlogn)= 1.8697885683002844(n*logn) # n*n = 36 T(n)=(C+S)/(n^2)= 0.8055555555555556(n^2)</p>	0	1	2	3	4	5	1	2	3	4	5	6												
0	1	2	3	4	5	6	7																																																		
4	5	8	11	12	15	64	35																																																		
0	1	2	3	4	5	6	7																																																		
4	5	8	11	12	15	35	64																																																		
0	1	2	3	4	5																																																				
1	2	3	4	5	6																																																				
<p># n = 8</p> <p># num compare(C) = 18 # num swap(S) = 13 # C+S = 31 # nlogn = 24.0 T(n)=(C+S)/(nlogn)= 1.2916666666666667(n*logn)</p>																																																									

Figure 8.17: Animation of quick sort algorithm

8.7. QUICK SORT

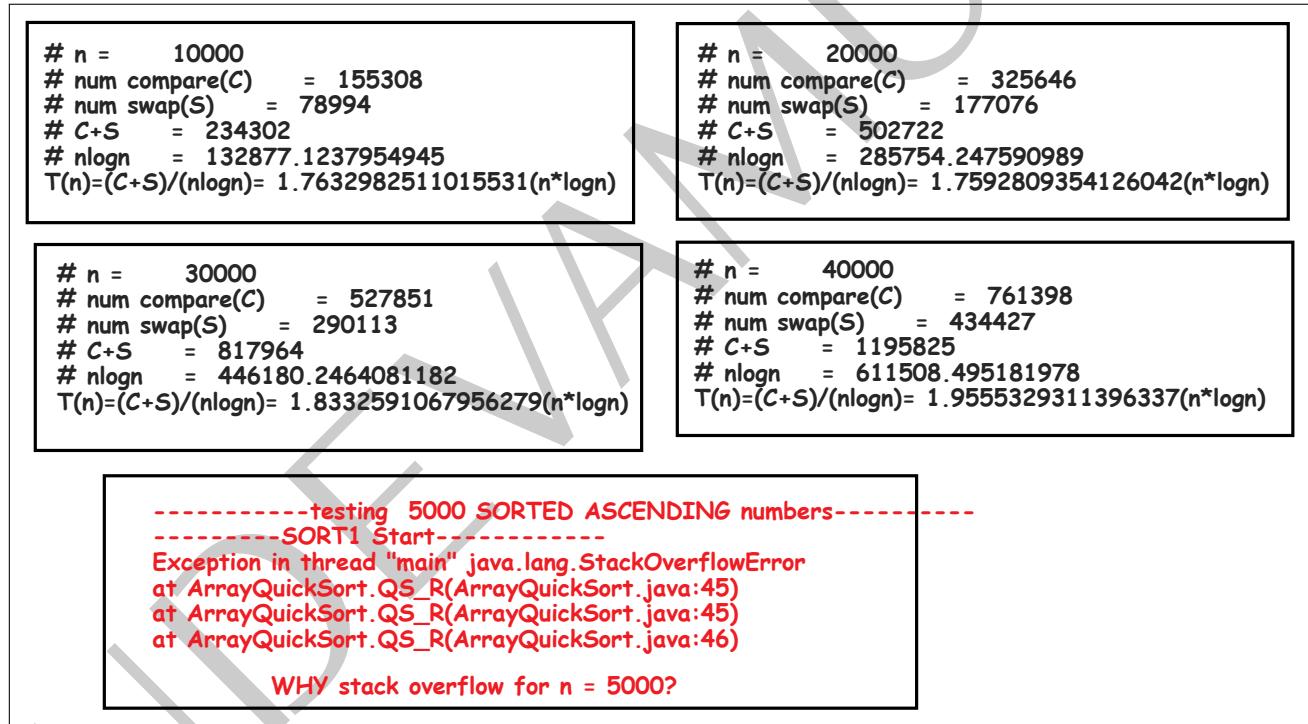


Figure 8.18: Animation of quick sort algorithm

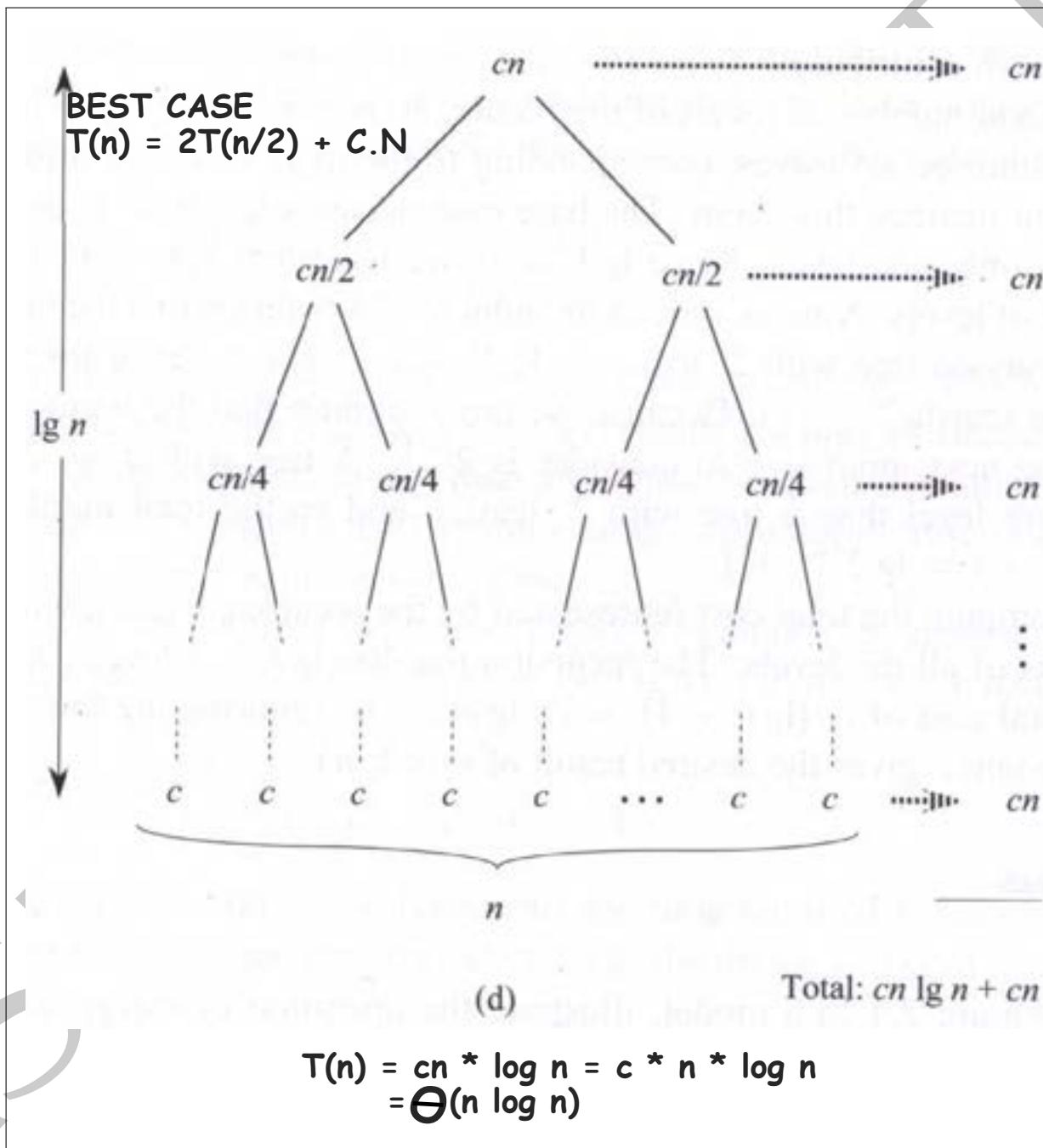


Figure 8.19: Best case complexity of quick sort

8.7. QUICK SORT

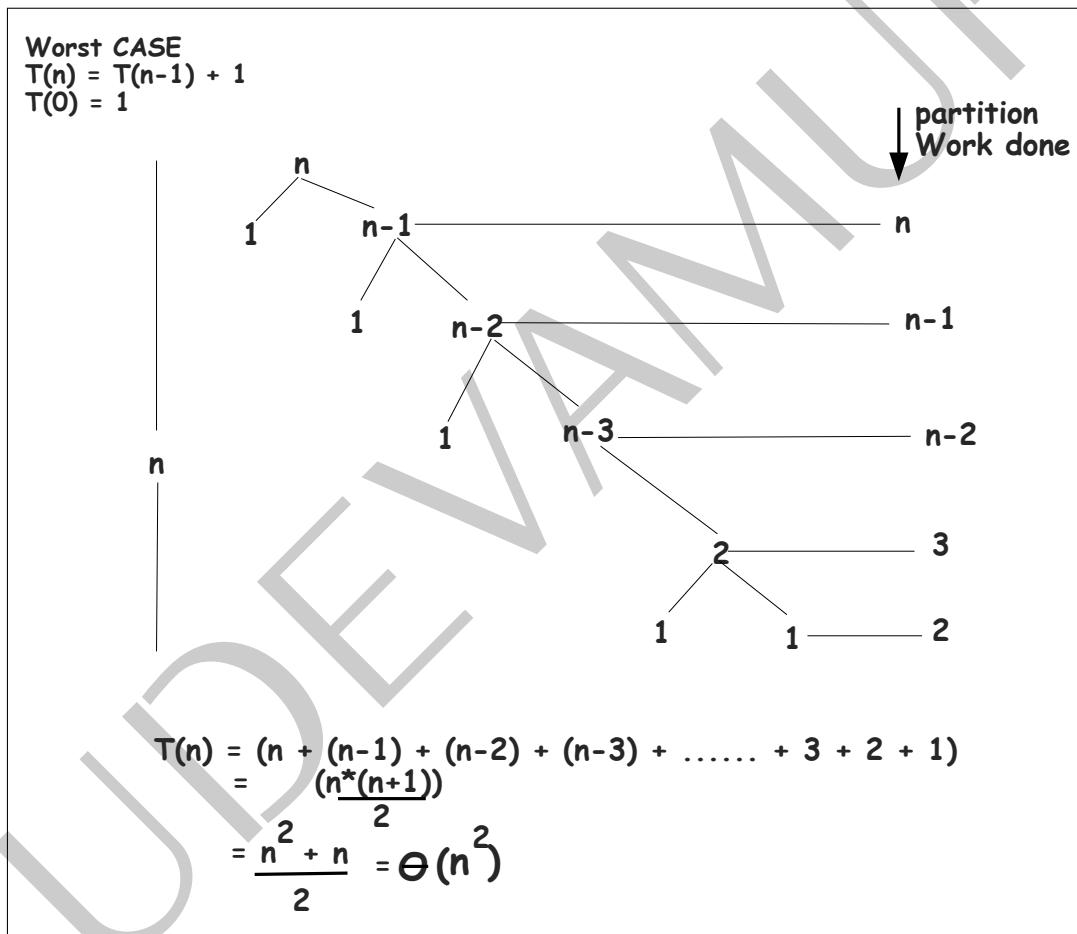


Figure 8.20: Worst case complexity of quick sort

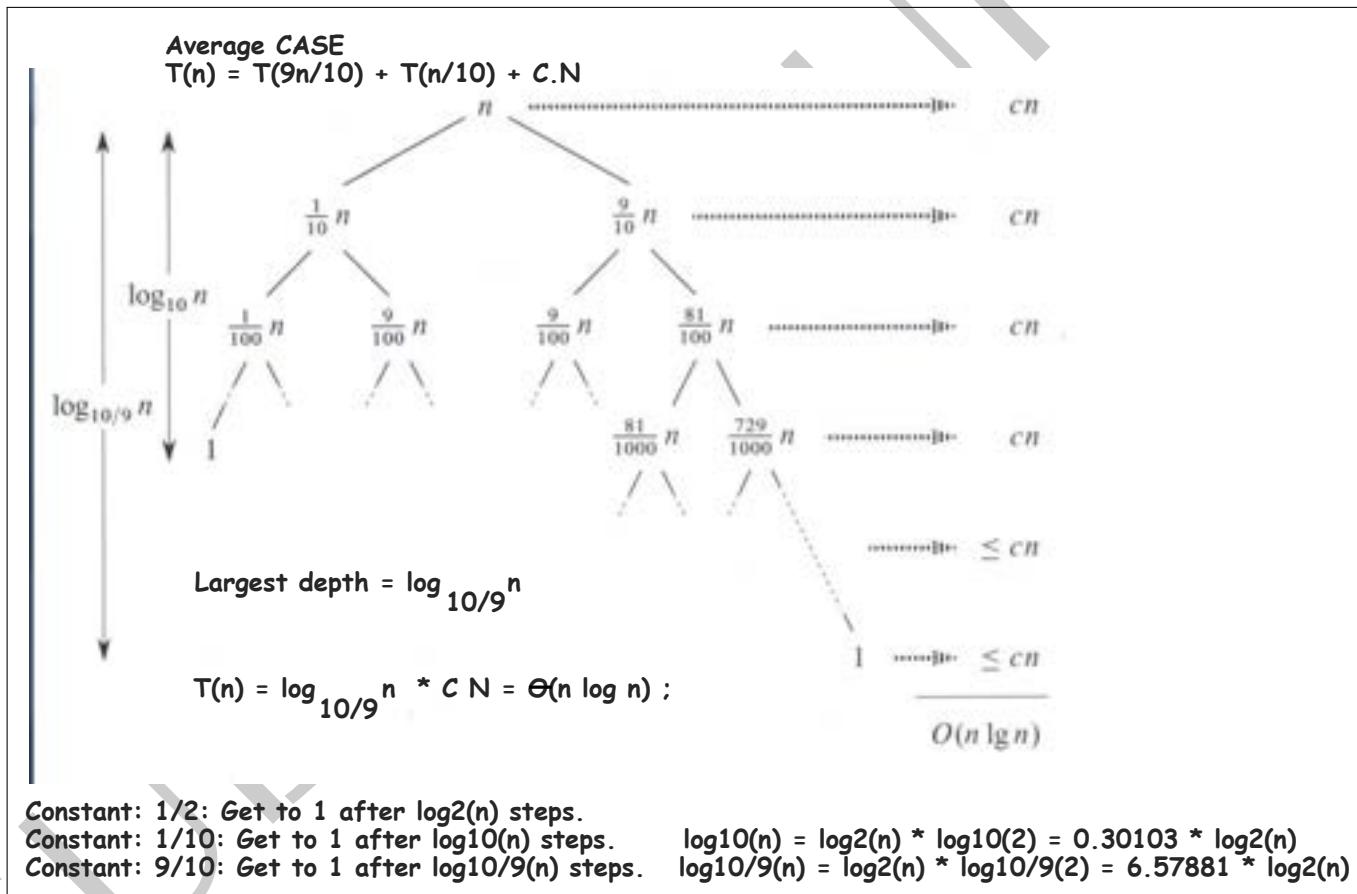


Figure 8.21: Expected complexity of quick sort

8.8. QUICK SELECT

8.7.2 Linked list based quick sort

8.8 Quick select

Finding kth element in an array

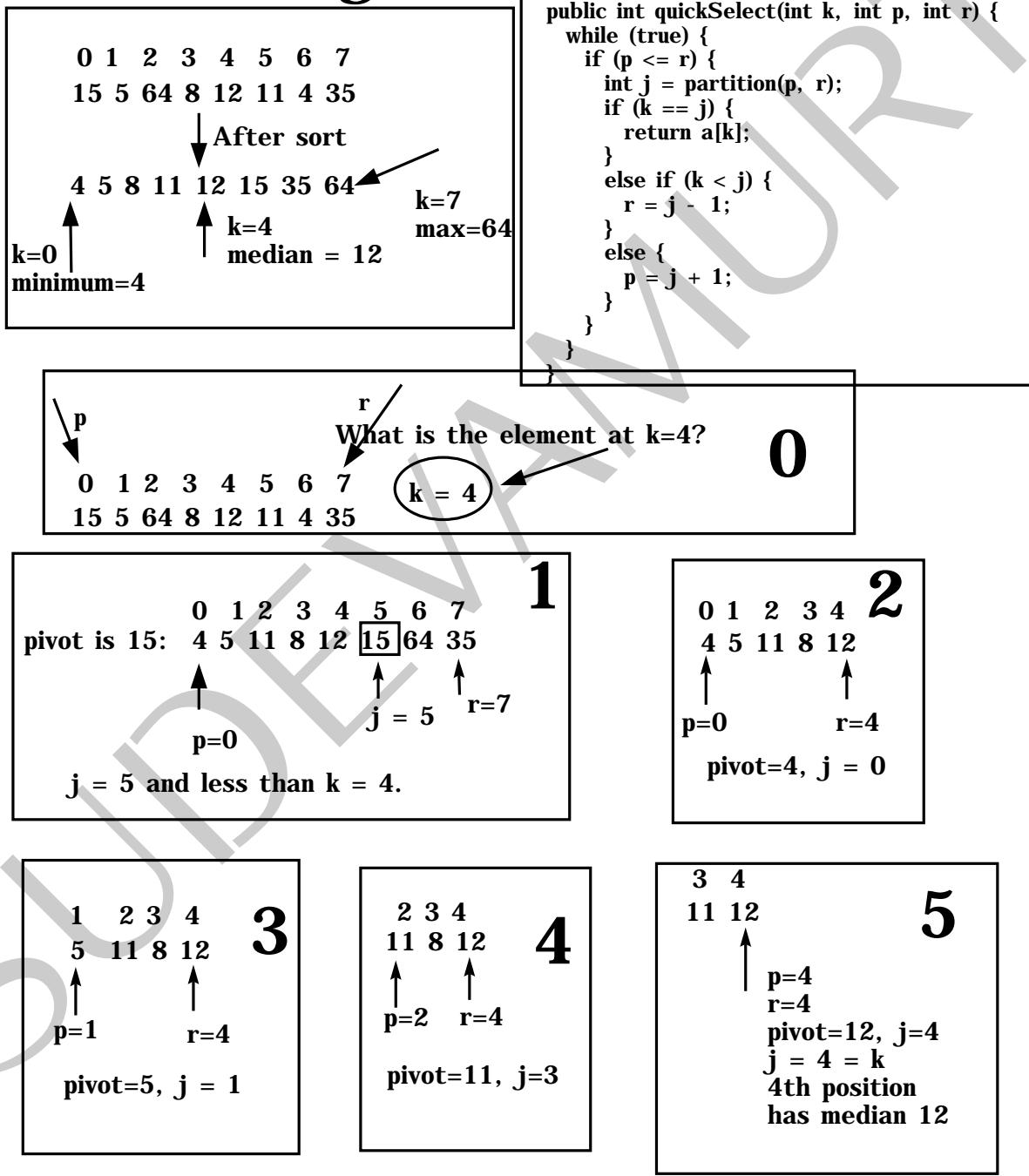


Figure 8.22: Finding k-th element using partition algorithm of quick sort

8.9. COUNTING SORT

8.9 Counting Sort

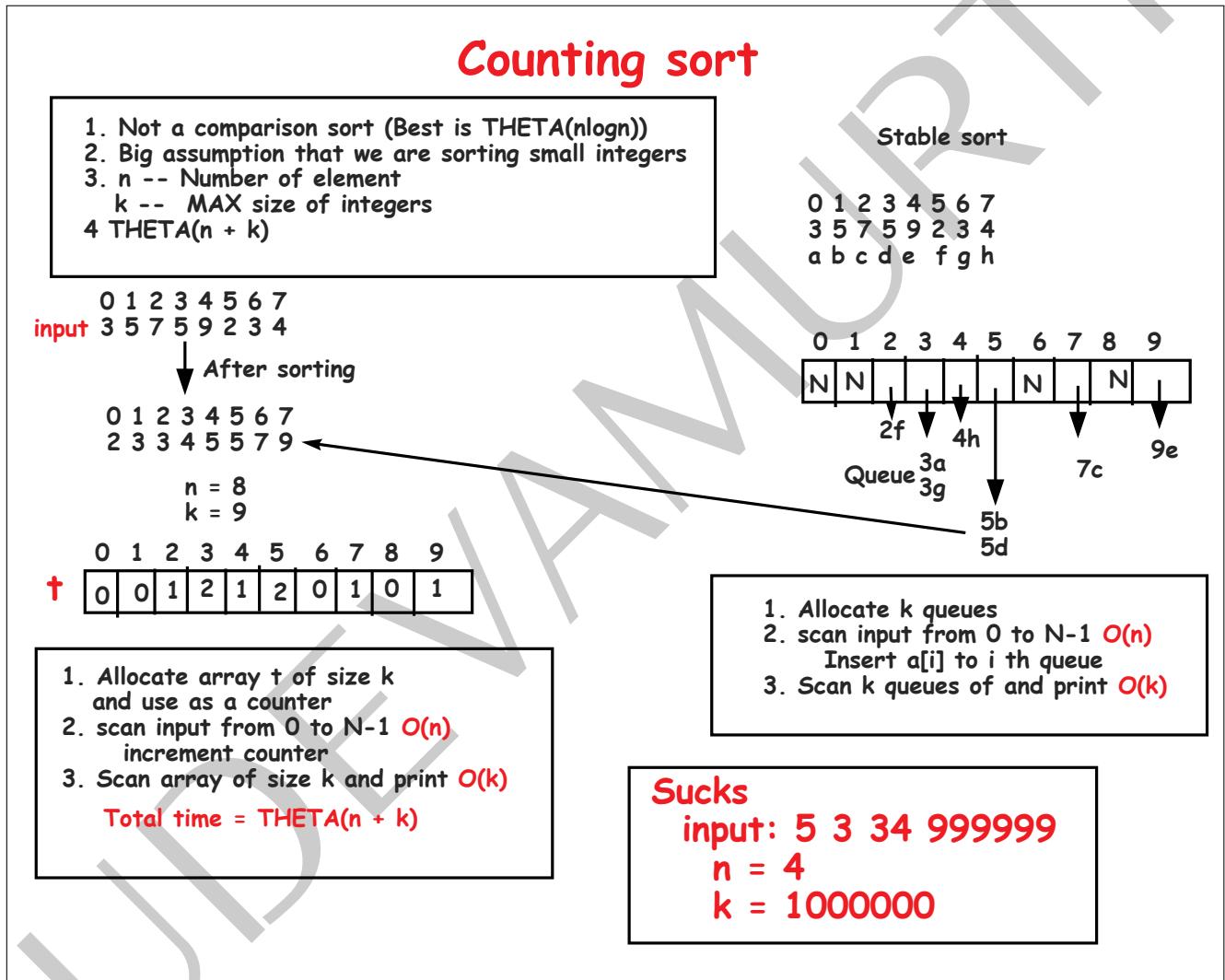


Figure 8.23: Counting sort and stable counting sort

8.10 Straight radix sort: from LSB to MSB

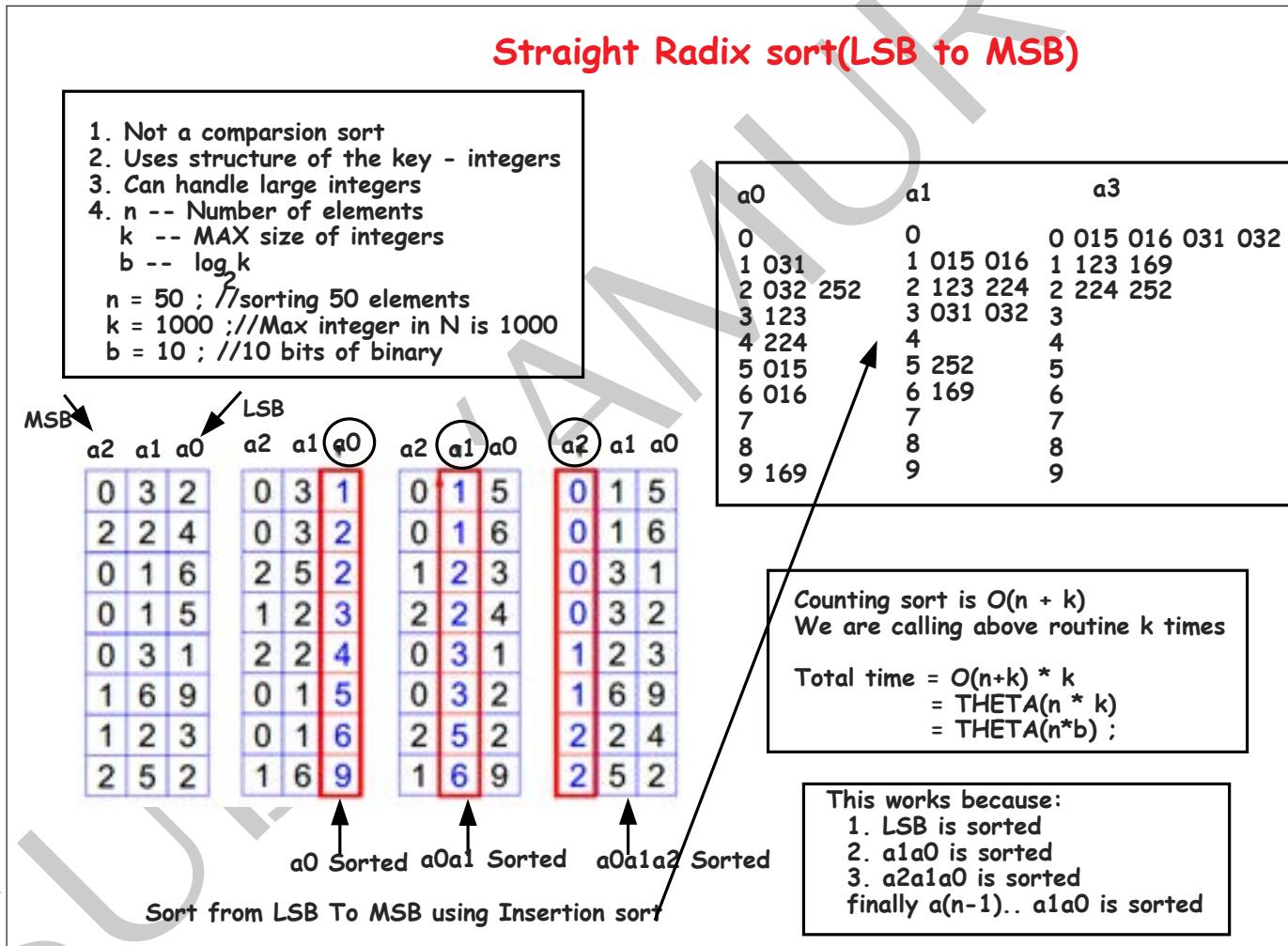


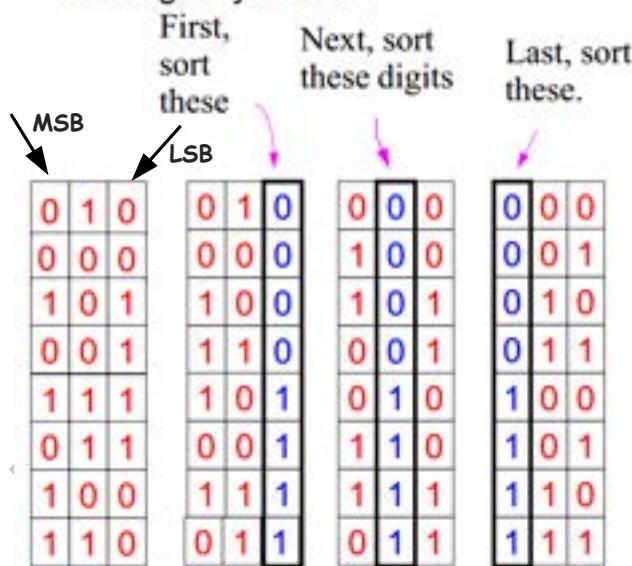
Figure 8.24: Straight radix sort, from LSB to MSB. Uses decimal number for illustration

8.10. STRAIGHT RADIX SORT: FROM LSB TO MSB

Straight Radix Sort using Binary digits(LSB to MSB)

Examines bits from *right to left*

```
for k := 0 to b-1  
    sort the array in a stable way,  
    looking only at bit k
```



Note order of these bits after sort.

Figure 8.25: Straight radix sort, from LSB to MSB using binary representation of a number

```

class StraightRadixSort extends ArraySort{

    private int numberOfBits() {
        int n = a.length ;
        if (n > 0) {
            int max = a[0] ;
            u.myassert(max >= 0) ;
            for (int i = 1; i < n ; ++i) {
                u.myassert(a[i] >= 0) ;
                if (a[i] > max) {
                    max = a[i] ;
                }
            }
            if (max == 0) {
                return 1 ;
            }
            int b = 0 ;
            while (max != 0) {
                max = max/2 ;
                b++ ;
            }
            return b ; Computes max  
number of bits  
required
        }
        return 0 ;
    }

    private void rs1(int bit) {
        int n = a.length ;
        int mask = 1 << bit ;
        int [] zeroa = new int[n] ;
        int t0 = 0 ;
        int [] onea = new int[n] ;
        int t1 = 0 ;

        for (int i = 0 ; i < n; ++i) {
            numCompare++ ;
            numSwap++ ;
            if (((a[i] & mask) != 0) { //Looking for 1
                onea[t1++] = a[i] ;
            }else {
                zeroa[t0++] = a[i] ;
            }
        }
        int k = 0 ;
        for (int i = 0; i < t0; ++i) {
            numSwap++ ;
            a[k++] = zeroa[i] ;
        }
        for (int i = 0; i < t1; ++i) {
            numSwap++ ;
            a[k++] = onea[i] ;
        }
        u.myassert(k == n) ;
    }

    private void rs(int nb) {
        for (int i = 0; i < nb; ++i) { Θ(n+k)
            rs1(i);
        }
    }

    @Override
    protected void sort(boolean ascend) {
        int nb = numberOfBits();
        rs(nb) ;
    }
}

```

Figure 8.26: Code and complexity of straight radix sort

8.11. RADIX EXCHANGE SORT: FROM MSB TO LSB

8.11 Radix exchange sort: from MSB to LSB

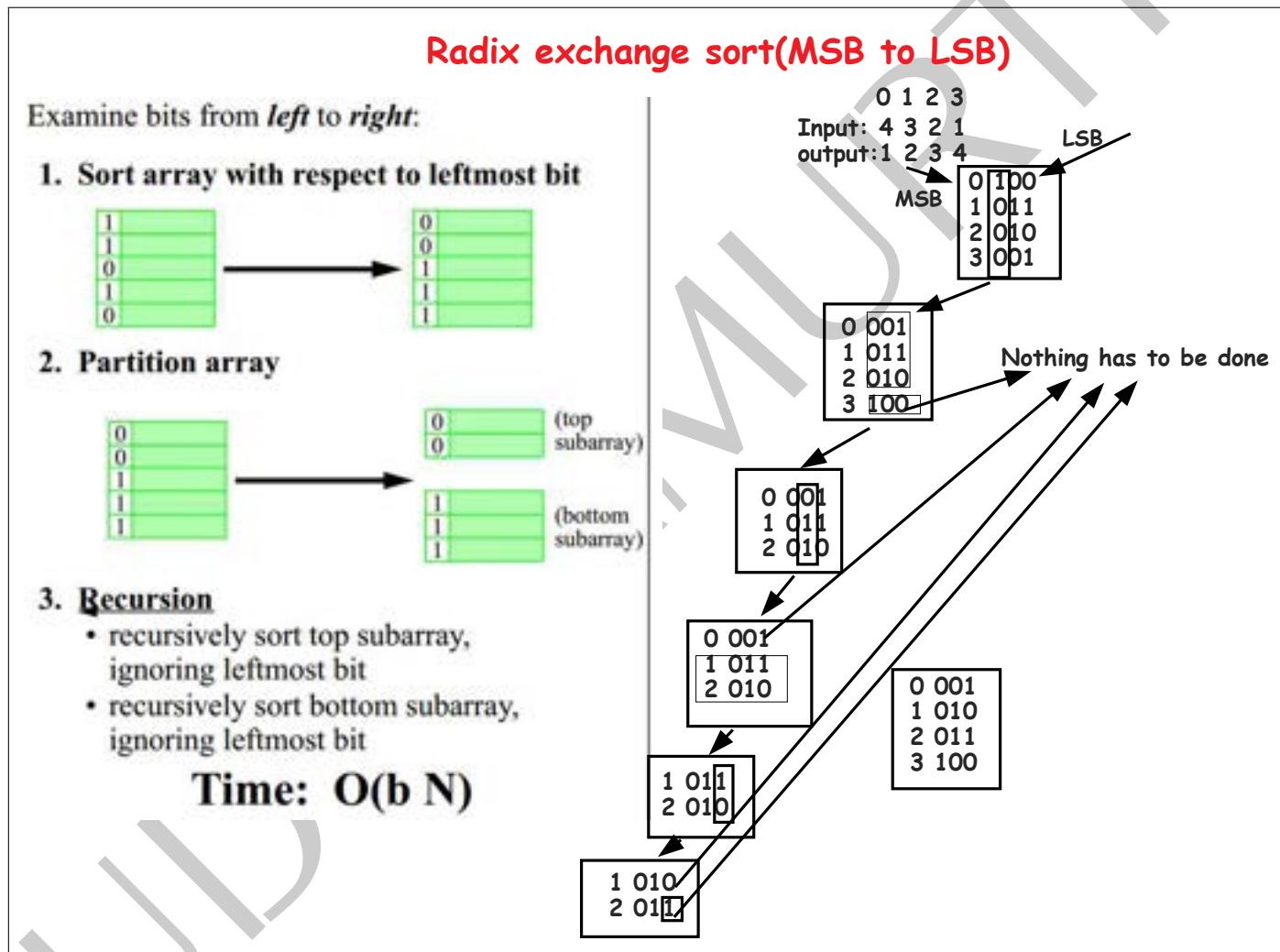


Figure 8.27: Radix exchange sort, from MSB to LSB

8.12 Problem set

Chapter 9

Hash

9.1 Introduction

9.2 What is a hash?

9.2. WHAT IS A HASH?

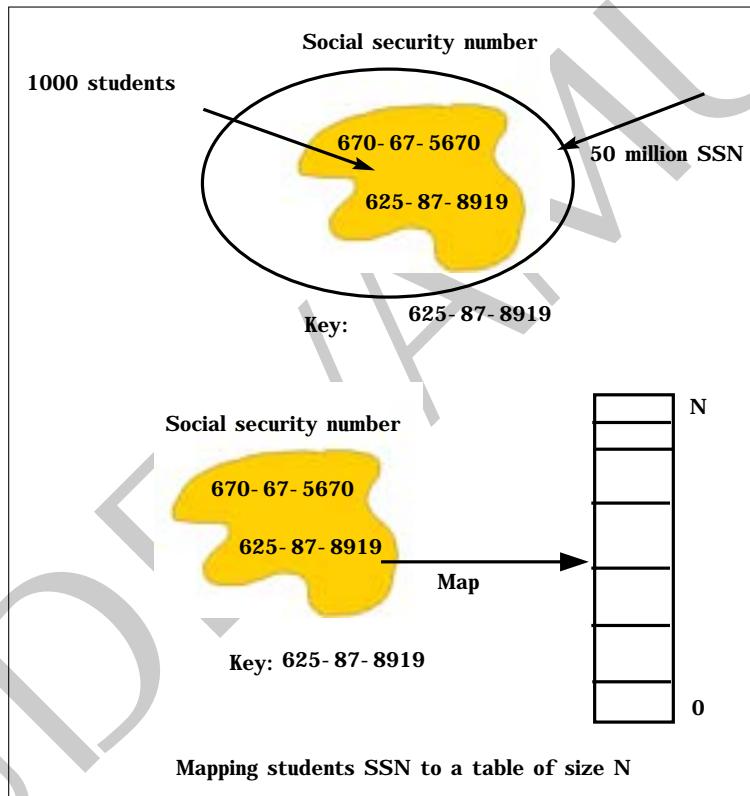


Figure 9.1: Why hash is required?

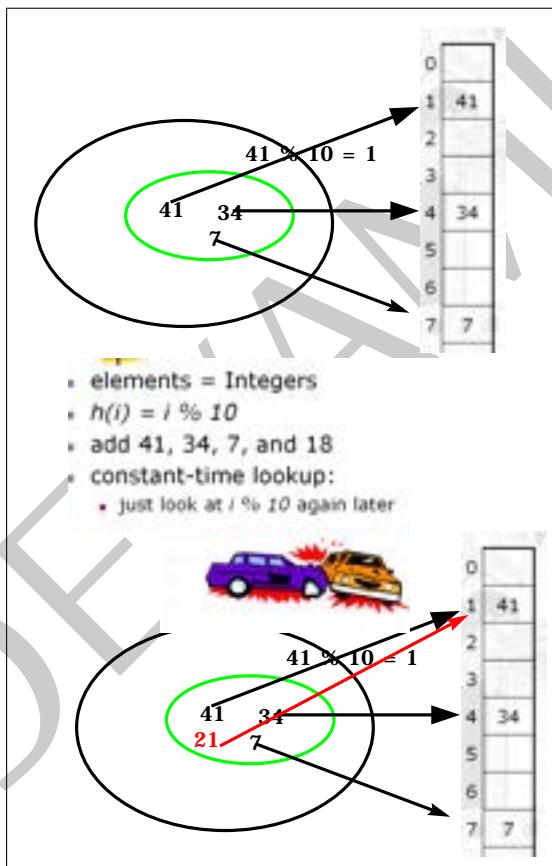


Figure 9.2: Why collisions happens?

9.2. WHAT IS A HASH?

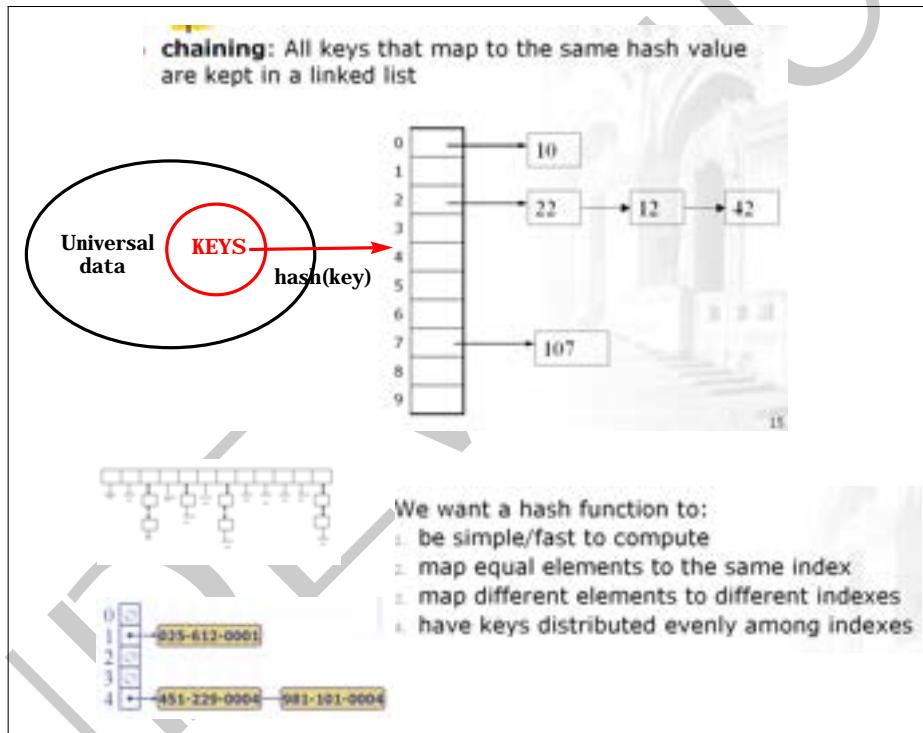


Figure 9.3: Properties of hash

String hash function

s is the String
n is the length of **s.length()**

3	2	1	0
C A L L			

in ASCII A=65, C=67 and L=76

$$\begin{aligned}
 & 31^3 \times s[0] + 31^2 \times s[1] + 31^1 \times s[2] + 31^0 \times s[3] \\
 & = 1995997 + 62465 + 2356 + 76 \\
 & = 2060894
 \end{aligned}$$

$31^{(n-1)} s[n-1] + 31^{(n-2)} s[n-2] + 31^{(n-3)} s[n-3] + 31^0 s[0]$

$$\begin{aligned}
 & 31^3 \times 67 + 31^2 \times 65 + 31^1 \times 76 + 31^0 \times 76 \\
 & = 76 + 31(76 + 31(65 + 31(67))) \\
 & = 76 + 31(76 + 31(65 + 2077)) \\
 & = 76 + 31(76 + 31(2142)) \\
 & = 76 + 31(76 + 66402) \\
 & = 76 + 31(66478) \\
 & = 76 + 2060818 \\
 & = 2060894
 \end{aligned}$$

Horner's Rule

integer hash function

```

def _hash_func1(self, key:'int'):
    key = ~key + (key << 15);
    key = key ^ (key >> 12);
    key = key + (key << 2);
    key = key ^ (key >> 4);
    key = key * 2057;
    key = key ^ (key >> 16);
    return key % (self._table_size)

```

```

def _hash_func(self, key:'int'):
    return key % (self._table_size)

```

Figure 9.4: Hash function

9.3 hash Class

Hash

Copyright: Jagadeesh Vasudevamurthy

filename:Hash.ipynb 

All import here

```
In [ ]: 1 import sys # For getting Python Version  
2 import random  
3 import math
```

ListNode class

```
In [ ]: 1 class ListNode:  
2     def __init__(self, val = 0, next= None):  
3         self.val = val  
4         self.next = next
```

Slist class

In []:

```
1 #####  
2 # Slist.py  
3 # SList obLect  
4 # Author: Jagadeesh Vasudevamurthy  
5 # Copyright: Jagadeesh Vasudevamurthy 2021  
6 #####  
7  
8 #####  
9 # ALL imports here  
10 #####  
11 #from ListNode import *  
12  
13 #####  
14 # class Slist  
15 #####  
16 class Slist():  
17     def __init__(self):  
18         #NOTHING CAN BE CHANGED HERE  
19         self._first = None  
20         self._last = None  
21  
22     #####  
23     # WRITE ALL public functions BELOW  
24     # YOU CAN HAVE ANY NUMBER OF PRIVATE FUNCTIONS YOU WANT  
25     #####  
26  
27     #####  
28     # ALL private functions BELOW  
29     # YOU CAN HAVE ANY NUMBER OF PRIVATE FUNCTIONS YOU WANT  
30     #####  
31
```

Hash Class

In []:

```

1 ######
2 # Hash.py
3 # Hash obLect
4 # Author: Jagadeesh Vasudevamurthy
5 # Copyright: Jagadeesh Vasudevamurthy 2021
6 #####
7
8 #####
9 # ALL imports here
10 #####
11 # #from ListNode import *
12 # from Slist import *
13
14 #####
15 # class Hash
16 #####
17 class Hash():
18     def __init__(self, size:'int'):
19         #WRITE YOUR DATA STRUCTURE HERE
20         self._table_size = size
21
22
23 #####
24 # WRITE ALL public functions BELOW
25 # YOU CAN HAVE ANY NUMBER OF PRIVATE FUNCTIONS YOU WANT
26 #####
27
28
29 #####
30 # WRITE ALL private functtions BELOW
31 # YOU CAN HAVE ANY NUMBER OF PRIVATE FUNCTIONS YOU WANT
32 #####
33
34 #####
35 # Time:    THETA(1)
36 # Space:   THETA(1)
37 # NOTHING CAN BE CHANGED BELOW
38 #####
39 def _hash_func1(self,key:'int'):
40     key = ~key + (key << 15);
41     key = key ^ (key >> 12);
42     key = key + (key << 2);
43     key = key ^ (key >> 4);
44     key = key * 2057;
45     key = key ^ (key >> 16);
46     return key % (self._table_size)
47
48 #####
49 # Time:    THETA(1)
50 # Space:   THETA(1)
51 # NOTHING CAN BE CHANGED BELOW
52 #####
53 def _hash_func(self,key:'int'):
54     return key % (self._table_size)
55

```

Util class

Nothing can be changed

No need to write any code here

```
In [ ]: 1 class Util():
2     pass
3
4     #####
5     # generate_random_number start to end INCLUDED
6     # start to end INCLUDED
7     #####
8     def generate_a_random_number(self,start:int,end:int)->'int':
9         v = random.randrange(start,end+1);
10        return v
11
12    #####
13    # generate_random_number GENERATES N random numbers between
14    # start to end INCLUDED
15    # if onlypositive is False, generates both pos and negative number
16    # randrange(beg, end, step) :-
17    # beginning number (included in generation),
18    # Last number (excluded in generation) a
19    # nd step ( to skip numbers in range while selecting).
20    #####
21    def generate_random_number(self, N:int, onlypositive:bool, start:int, er
22        a = []
23        for i in range(N):
24            v = self.generate_a_random_number(start,end);
25            if (onlypositive == False):
26                if ((i % 2) == 0): #####Even. Half are positive, Half are neg
27                    v = -v ;
28            a.append(v)
29        return a
30
31    #####
32    # swap
33    #####
34    def swap(self,a:'List of integer', i:'int', j:'int'):
35        t = a[i]
36        a[i] = a[j]
37        a[j] = t
38
39    #####
40    # generate shuffled number between 0 to n
41    # n-1 not encluded
42    #####
43    def generate_shuffled_number_between_0_to_n(self, n:int)->'List of integer':
44        a = []
45        for i in range(n):
46            a.append(i)
47
48        for i in range(n):
49            j = self.generate_a_random_number(0,n-1);
50            k = self.generate_a_random_number(0,n-1);
51            self.swap(a,j,k)
52        return a
53
54    #####
55    # generate n numbers in ascending order from 0 to n-1
56    #####
```

```
57  def generate_n_numbers_inAscending_order(self, n:int)->'List of integer'
58      a = []
59      for i in range(n):
60          a.append(i)
61      return a
62
63 ######
64 # generate n numbers in descending order from n-1 to 0
65 #####
66  def generate_n_numbers_in_descending_order(self, n:int)->'List of integers':
67      a = []
68      for i in range(n-1,-1,-1):
69          a.append(i)
70      return a
71
72 #####
73 # generate n same k number
74 #####
75  def generate_n_same_k_number(self, n:int,k:'int')->'List of integer':
76      a = []
77      for i in range(n):
78          a.append(k)
79      return a
80
81 #####
82 # print_index(10)
83 #    0   1   2   3   4   5   6   7   8   9
84 #####
85  def print_index(self, n:int):
86      for i in range(n):
87          print("{:4d}".format(i),end="")
88      print()
89
90 #####
91 # a = [7,8,9, 23, 100]
92 # print_list(a)
93 # 7   8   9  23 100
94 #####
95  def print_list(self, a:'list'):
96      for i in range(len(a)):
97          print("{:4d}".format(a[i]),end="")
98      print()
99
100 #####
101 # a = [7,8,9, 1, 100]
102 # crash
103 #####
104  def assertAscending_range(self, a:'list', start:int, includingend:int):
105      t = a[start]
106      for i in range(start+1,includingend):
107          if (a[i] < t):
108              assert(False)
109          t = a[i]
110
111 #####
112 # a = [7,8,9, 1, 100] 298
113 # crash
```

```
114 #####  
115 def assertAscending(self, a:'list'):  
116     if (len(a)):  
117         self.assertAscending_range(a,0,len(a))  
118 #####  
119 # Log to the next possible integer  
120 #####  
121 def logUpper_bound(self, n:'int', b:'int')->'int':  
122     f = math.log(n,b)  
123     c = math.ceil(f)  
124     return c  
125 #####  
126 # Log to the smallest possible integer  
127 #####  
128 def logLower_bound(self, n:'int', b:'int')->'int':  
129     f = math.log(n,b)  
130     c = math.floor(f)  
131     return c  
132 #####  
133 # sqrt to the next possible integer  
134 #####  
135 def sqrtUpper_bound(self, n:'int')->'int':  
136     f = math.sqrt(n)  
137     c = math.ceil(f)  
138     return c
```

Hash test class

NOTHING CAN BE CHANGED BELOW

In []:

```

1 ######
2 # HashTest.py
3 # Test Bench for Hash
4 # Author: Jagadeesh Vasudevamurthy
5 # Copyright: Jagadeesh Vasudevamurthy 2021
6 #####
7
8 ######
9 # NOTHING CAN BE CHANGED IN THIS FILE
10 #####
11
12 #####
13 # ALL imports here
14 #####
15 import sys # For getting Python Version
16 #from Util import *
17 #from Hash import *
18
19 #####
20 # class Hash test
21 #####
22 class HashTest():
23     def __init__(self):
24         self._u = Util()
25         self._test_int_hash()
26
27     def _test1(self,N,B,S,E):
28         print("----- TESTING add to hash ")
29         print("----- Adding",N,"Random numbers between", S, "T")
30         print("Perfect hash should have exactly", N//B, "elements in each bu")
31         d={}
32         h = Hash(B)
33         for i in range(N):
34             v= self._u.generate_a_random_number(S,E)
35             ## Note v can have duplicate
36             #If hash has to be like dict, we need to modify slist node with
37             # Python dict. key is i(unique) and value is v
38             d[i] = v #insert in dict
39             h.insert(v) #insert in hash
40         assert(len(h) == N)
41         assert(len(h) == len(d))
42         [min,max] = h.statistics()
43         print("MIN= ",min)
44         print("MAX= ",max)
45         print("Perfect hash should have exactly", len(h)//B, "elements")
46
47         print("----- Testing find -----")
48         for key,value in d.items():
49             v = h.find(value)
50             assert(v)
51         print("----- Testing delete -----")
52         for key,value in d.items():
53             v = h.delete(value)
54
55             300
56     def _test_int_hash(self):

```

```
57     N = 1000
58     B = N
59     S = 0
60     E = 1000
61     self._test1(N,B,S,E)
62
63     N = 100000
64     B = N
65     S = 111111111
66     E = 999999999
67     self._test1(N,B,S,E)
68
69     N = 500000
70     B = N
71     S = 111111111
72     E = 999999999
73     self._test1(N,B,S,E)
74
75     N = 5000000
76     B = N
77     S = 111111111
78     E = 999999999
79     self._test1(N,B,S,E)
80
81 #####
82 # MAIN
83 #####
84 #####
85 def main():
86     print("Basic Hash test starts")
87     print(sys.version)
88     t = HashTest()
89     print("Basic Hash test Passed. ")
```

main

In []: 1 main()

9.3. HASH CLASS

9.3.1 Expected output

```
Basic Hash test starts
3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
----- TESTING add to hash
----- Adding 1000 Random numbers between 0 To 1000 Bucket size 1000 ---
Perfect hash should have exactly 1 elements in each bucket
MIN= 0
MAX= 6
Perfect hash should have exactly 1 elements
----- Testing find -----
----- Testing delete -----
----- TESTING add to hash
----- Adding 100000 Random numbers between 11111111 To 99999999 Bucket size 100000 ---
Perfect hash should have exactly 1 elements in each bucket
MIN= 0
MAX= 7
Perfect hash should have exactly 1 elements
----- Testing find -----
----- Testing delete -----
----- TESTING add to hash
----- Adding 500000 Random numbers between 11111111 To 99999999 Bucket size 500000 ---
Perfect hash should have exactly 1 elements in each bucket
MIN= 0
MAX= 9
Perfect hash should have exactly 1 elements
----- Testing find -----
----- Testing delete -----
----- TESTING add to hash
----- Adding 5000000 Random numbers between 11111111 To 99999999 Bucket size 5000000 ---
Perfect hash should have exactly 1 elements in each bucket
MIN= 0
MAX= 9
Perfect hash should have exactly 1 elements
----- Testing find -----
----- Testing delete -----
Basic Hash test Passed.
```

9.4 Problem set

Problem 9.4.1. Answer the questions below by hand and post the scanned document on Canvas.

Let $a = [5, 28, 19, 5, 20, 33, 12, 17, 10]$

1

Show the hash table as a picture. Table size = 9

1. Hash function $h(n) = 4$.
2. Hash function $h(n) = n \bmod 9$
3. Hash function $h(n) = (n * 31) \bmod 9$
4. Hash function $h(n) = (n * 27) \bmod 9$

Let $a = ["UCSC", "UCSB", "UCB", "UCR", "NE", "MIT", "STANFORD"]$

Show the hash table as a picture. Table size = 10

Hash function: let n be the length of the string

2

$A[n-1] = \text{ASCII character of } n-1$

NOTE $\text{ASCII}(A) = 65$, $\text{ASCII}(Z) = 90$

$$h[s] = 31^{n-1} * A[n-1] + 31^{n-2} * A[n-2] + \dots + 31^0 * A[0]$$

Problem 9.4.2. Implement hash

Please hash as explained in section 9.3

VASUDEVAMURTHY

Chapter 10

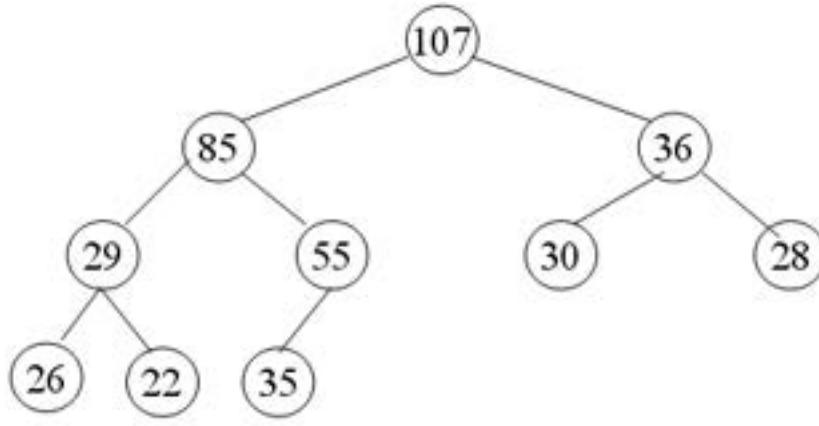
Heap

10.1 Introduction

10.2 What is a heap?

Heap

- A heap T is a complete Binary tree in which either T is empty or
- each item in $\text{Left}(T)$ is \leq Root item of T
- each item in $\text{Right}(T)$ is \leq Root item of T
- Left and Rights are heaps



- The ordering in a heap is *top-down, but not left or right*. Each root item is greater or equal to each of its children, but some left siblings may be greater than their right siblings and some be less. For example (85 > 36 but 29 < 55)

Figure 10.1: Max heap of int

10.3 Representation of heap as an array

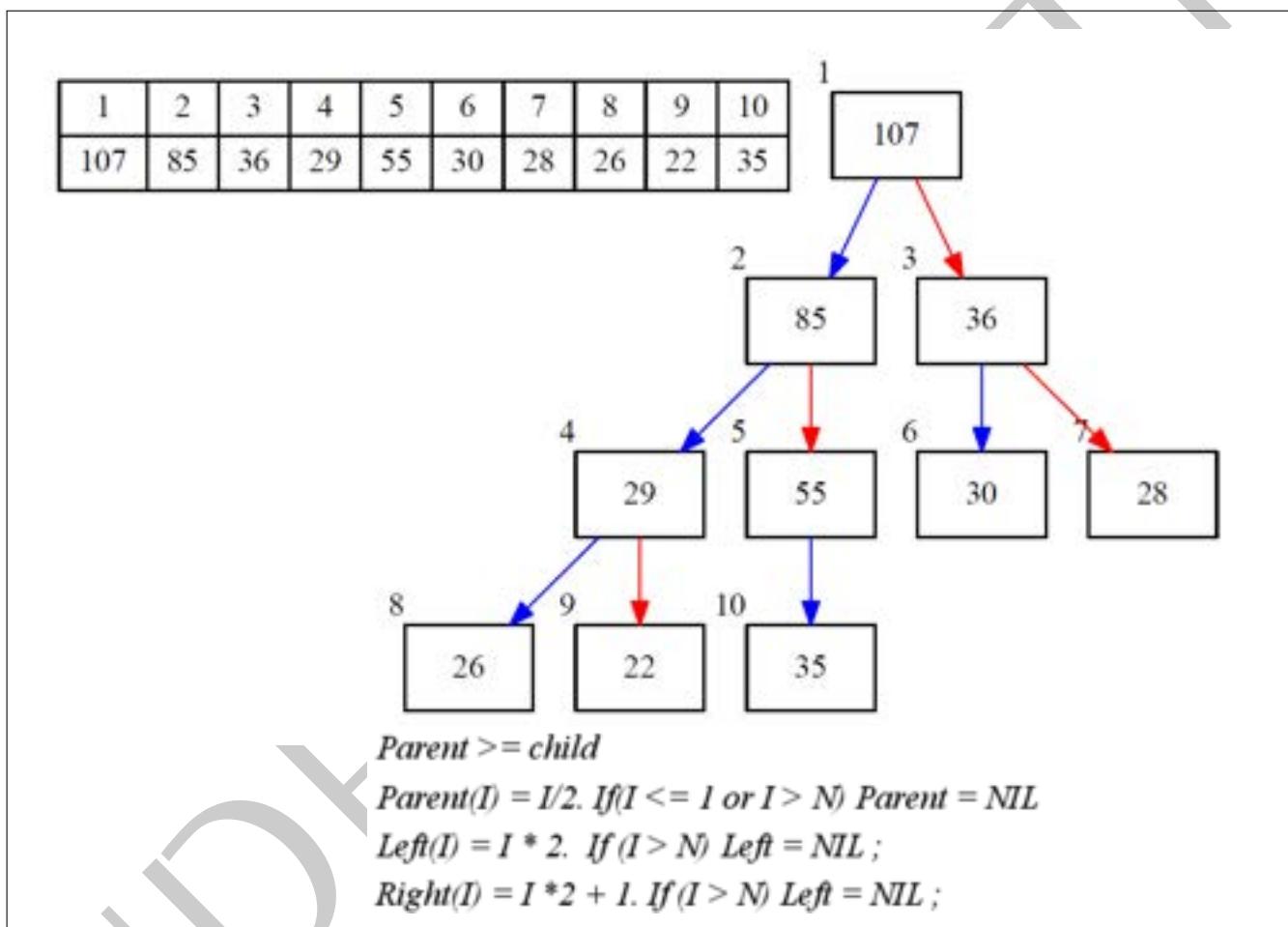


Figure 10.2: Representation of max heap of int

10.4 Finding maximum element of a heap

10.4. FINDING MAXIMUM ELEMENT OF A HEAP

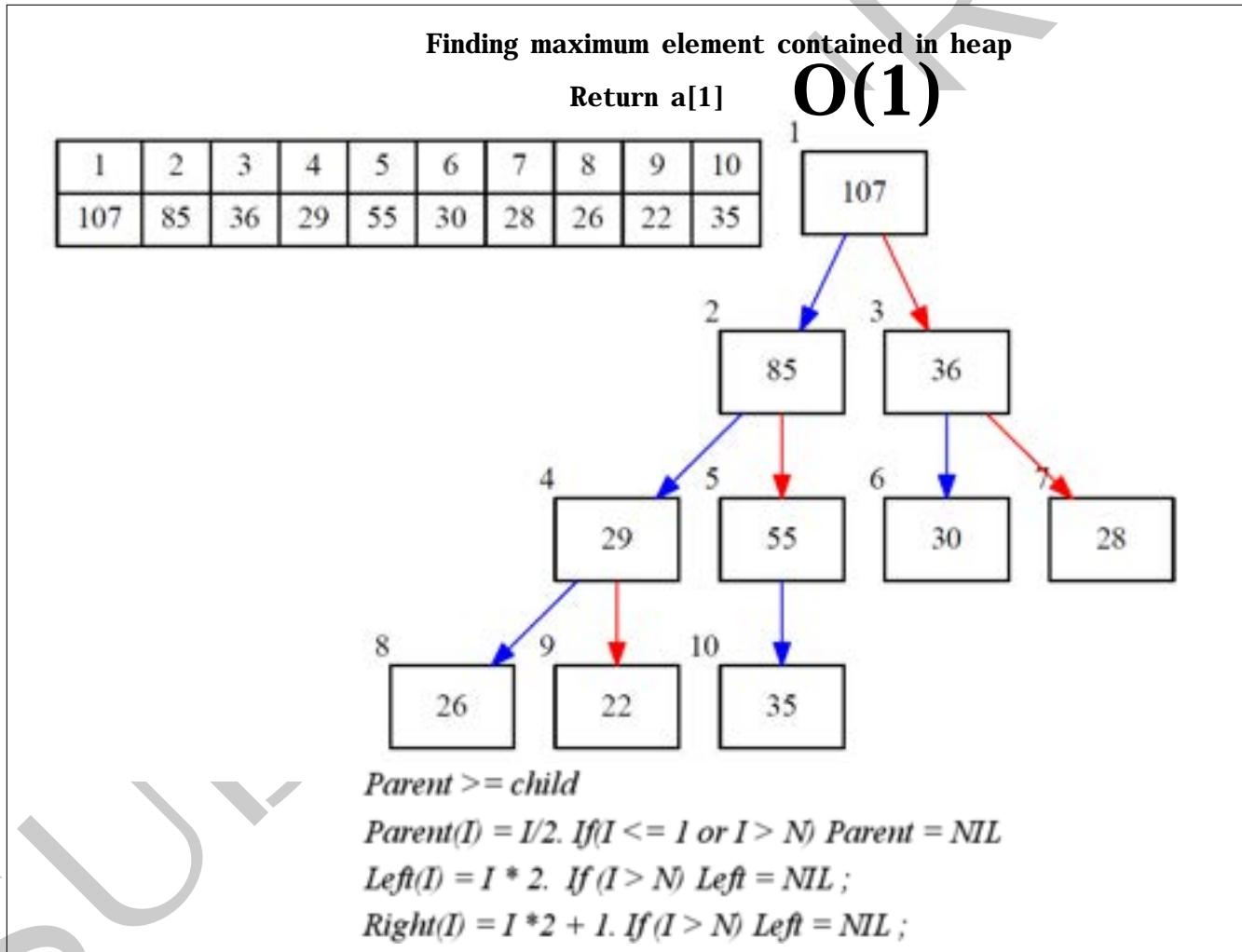


Figure 10.3: Finding max element

10.5 Inserting an element to the heap

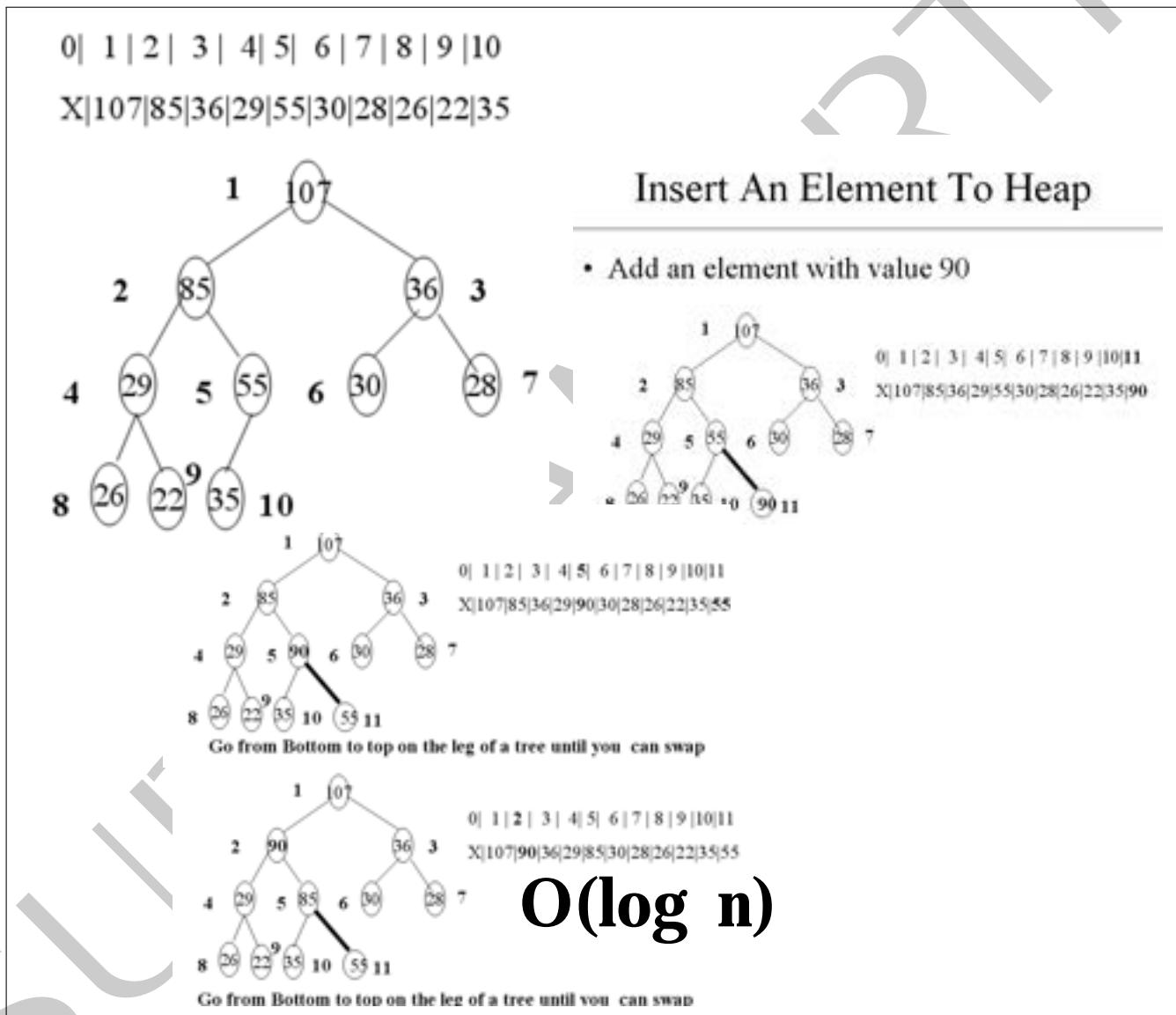


Figure 10.4: Inserting an element to the heap

10.6 Deleting maximum element from the heap

10.6. DELETING MAXIMUM ELEMENT FROM THE HEAP

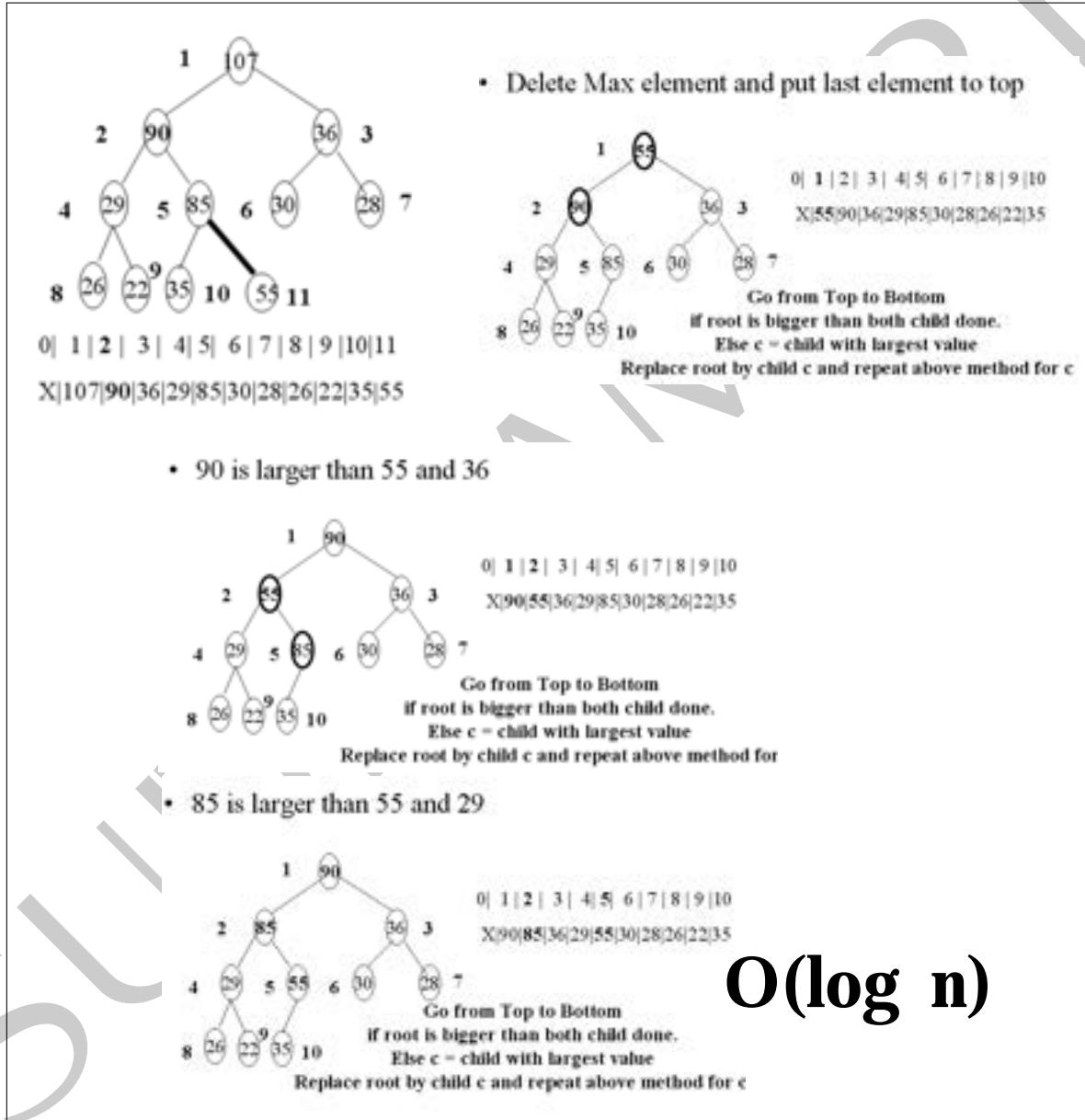


Figure 10.5: Deletion

10.7 Writing class Heap

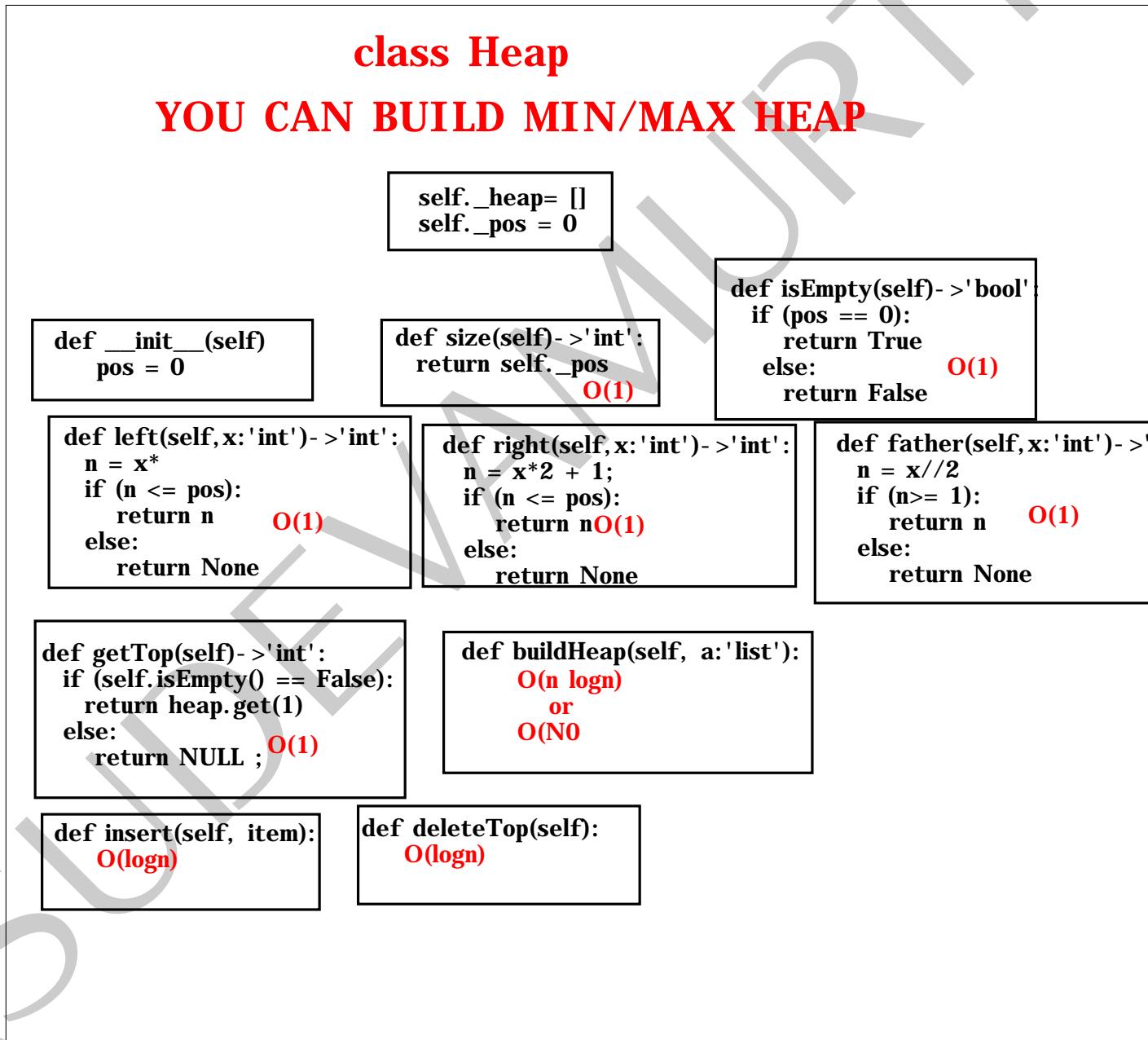


Figure 10.6: class Heap

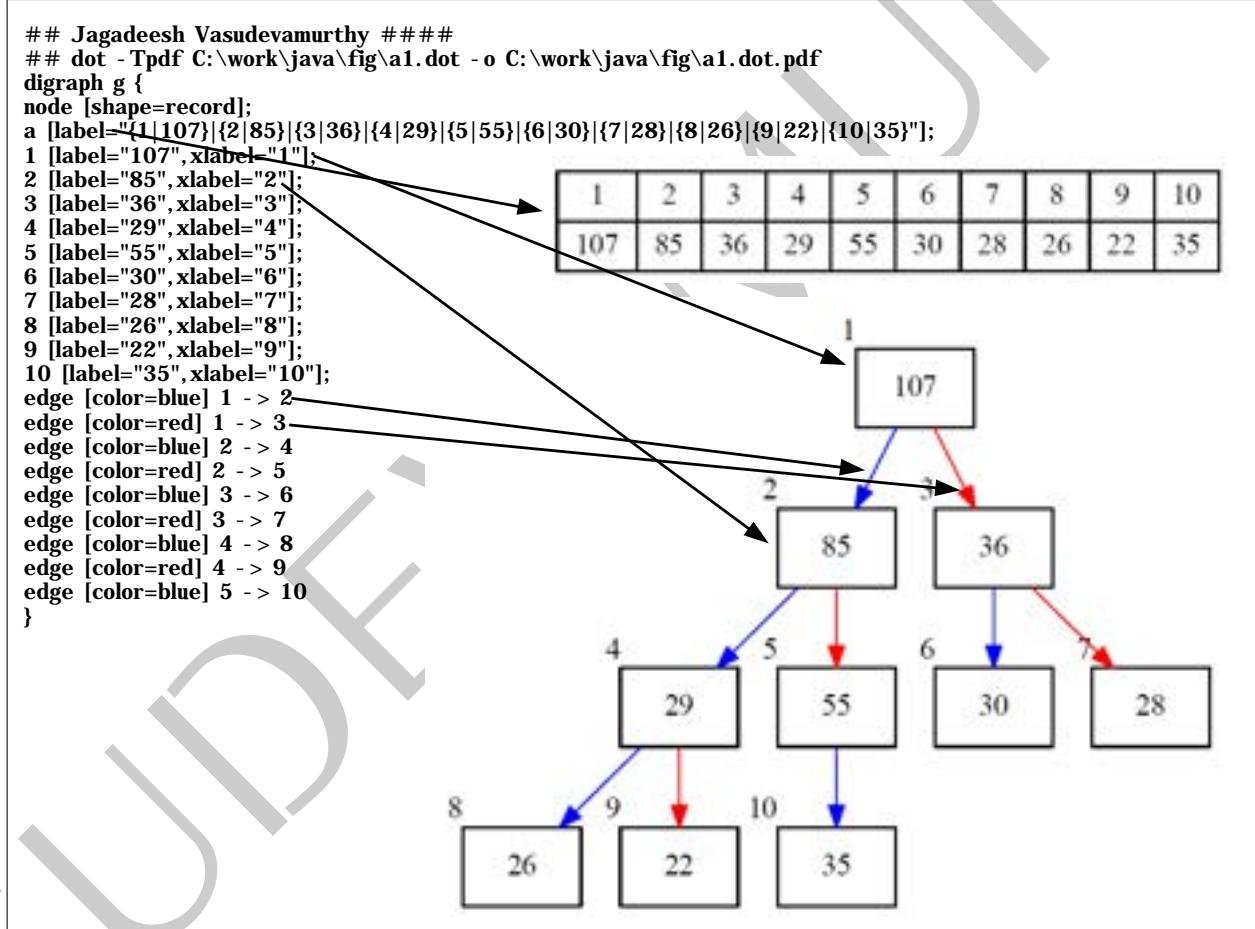


Figure 10.7: Printing heap as a dot file

10.8. BUILDING A HEAP OF N ELEMENTS FROM TOP TO BOTTOM

10.8 Building a heap of n elements from top to bottom

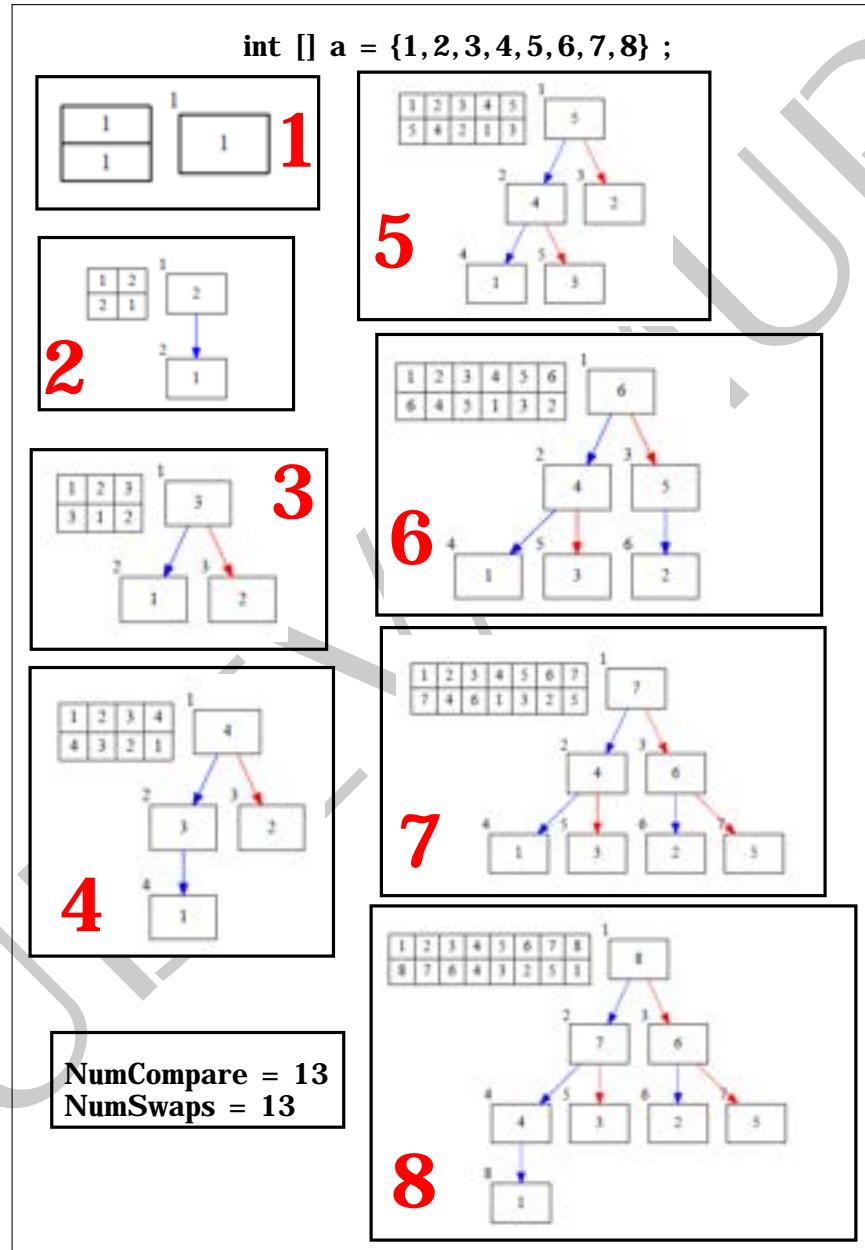


Figure 10.8: Building a heap of n elements from top to bottom

10.9 Building a heap of n elements from bottom to top

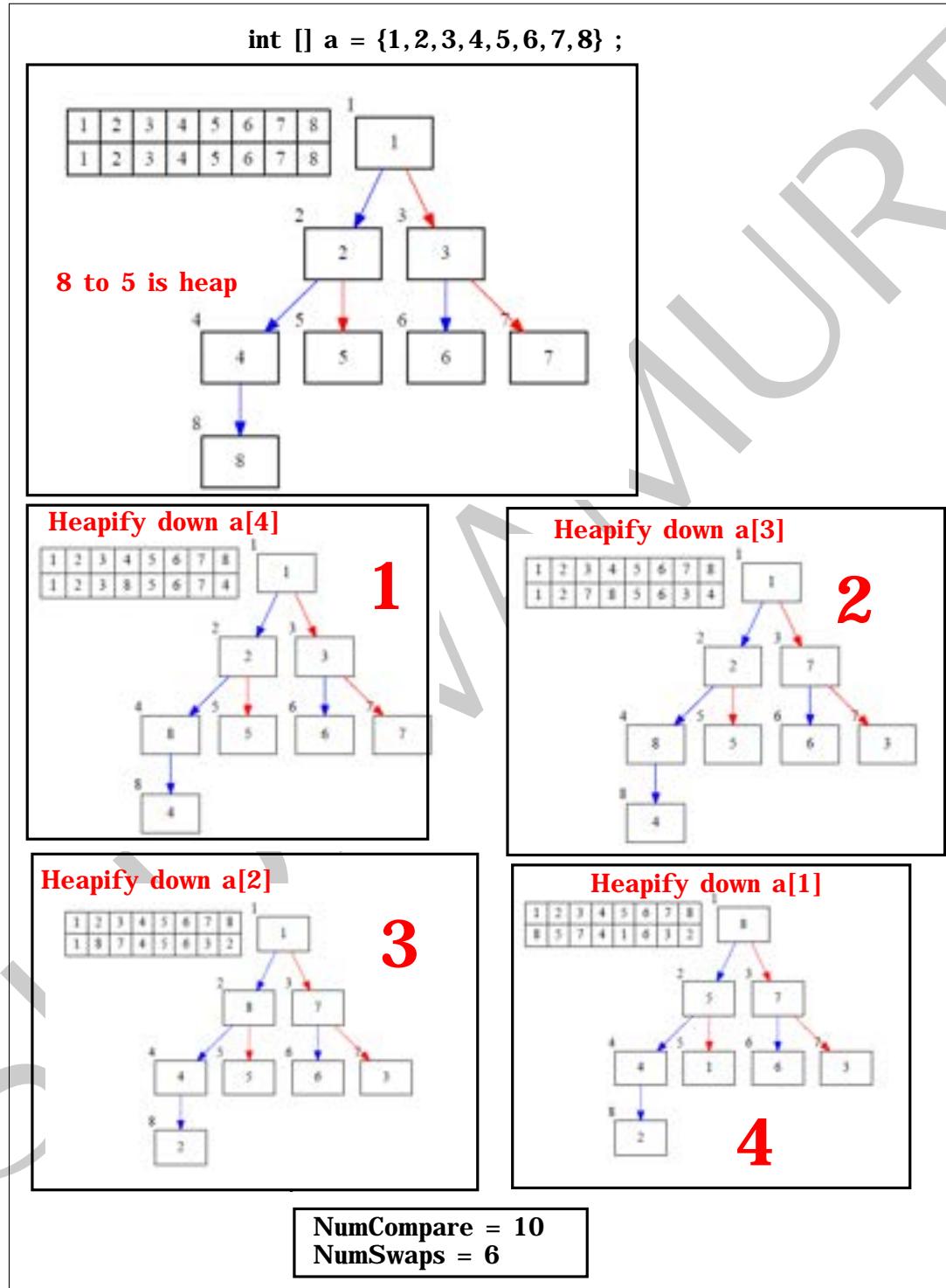


Figure 10.9: Building a heap of n elements from bottom to top

10.9. BUILDING A HEAP OF N ELEMENTS FROM BOTTOM TO TOP

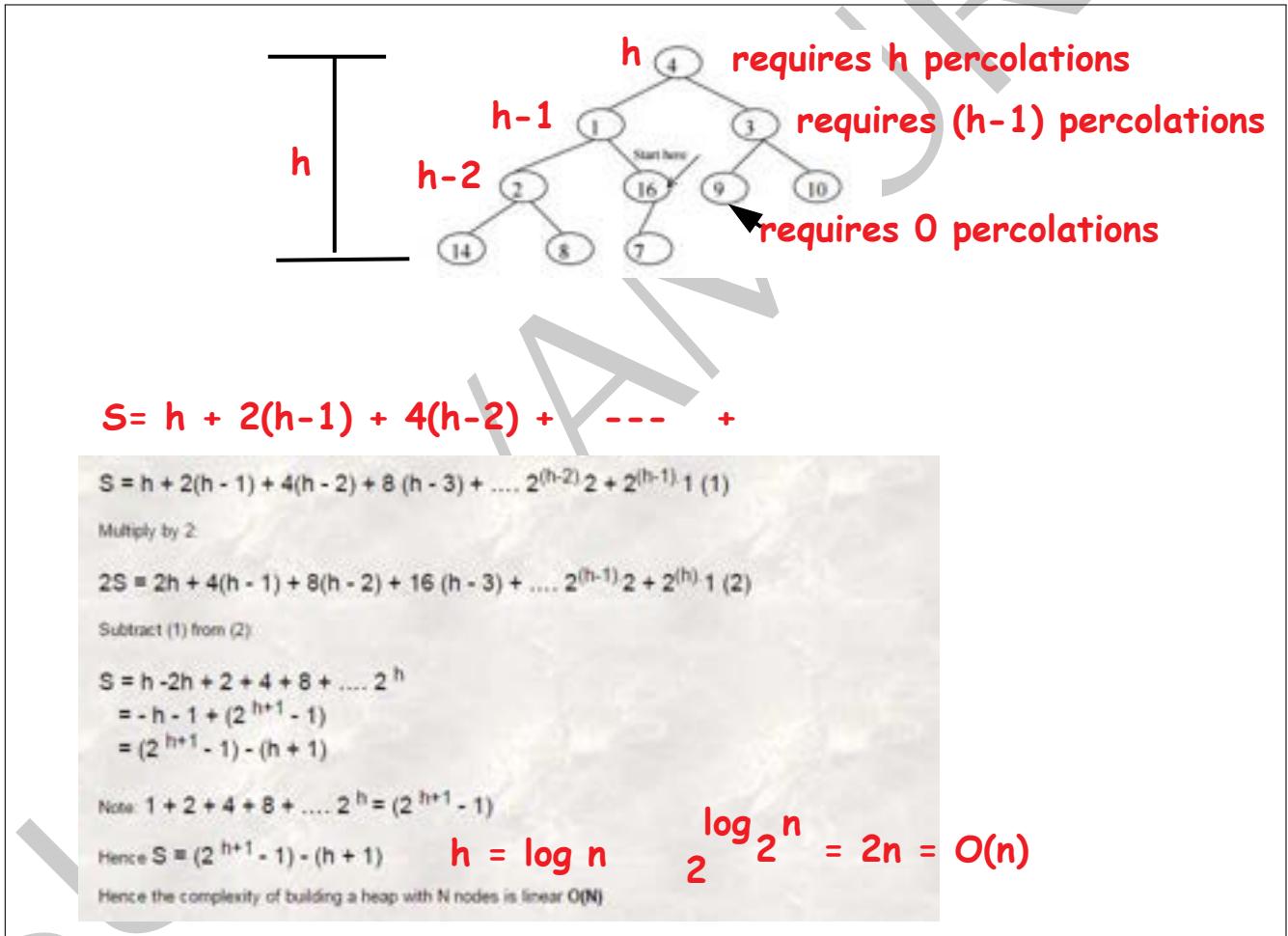


Figure 10.10: Analysis of building a heap of n elements from bottom to top

10.10 Heap sort algorithm

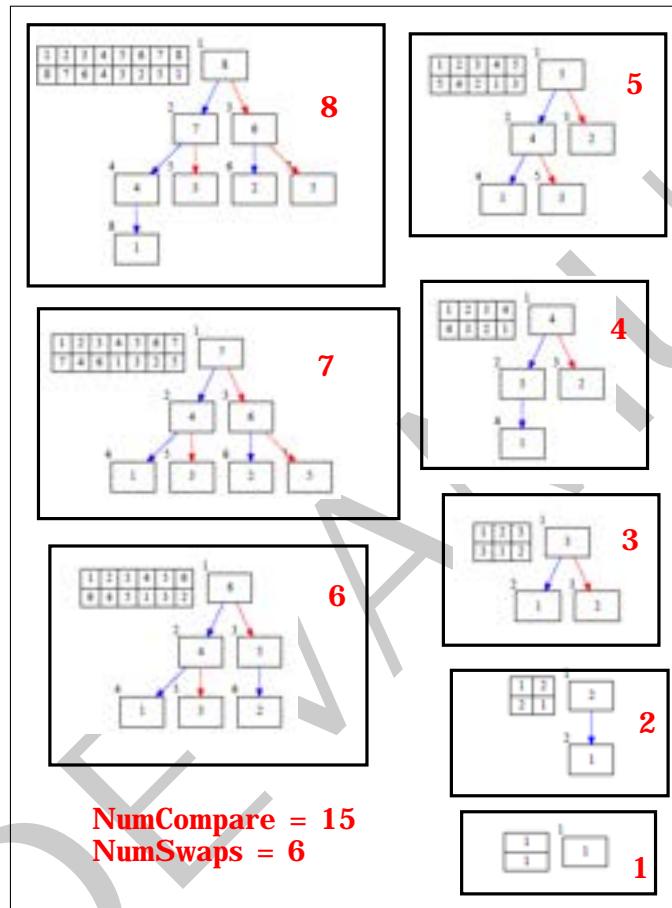


Figure 10.11: Heap sort

10.11 Heap in Python 3

Heap in Python

Copyright: Jagadeesh Vasudevamurthy

filename: heap.ipynb

```
In [11]: 1 import heapq
```

```
In [12]: 1 class Node:
2     def __init__(self, a:'int',b:'int'):
3         self._a = a
4         self._b = b
5
6     ##Override __lt__ in Python 3, __cmp__ only in Python 2
7     def __lt__(self, rhs:'int')->'bool':
8         if (self._a < rhs._a): # Change to > for max heap
9             return True
10            return False
```

```
In [13]: 1 class MinHeap:
2     def __init__(self):
3         self._q = []
4     def insert(self,a:'list'):
5         for e in a:
6             n = Node(e,-e)
7             heapq.heappush(self._q,n)
8
9     def add(self,n: 'Node'):
10        heapq.heappush(self._q,n)
11
12     def get_top(self)->'Node':
13         return self._q[0]
14
15     def get_top_and_remove(self)->'Node':
16         n = heapq.heappop(self._q)
17         return n
18
19     def deleteAll(self):
20         while (len(self._q)):
21             n = heapq.heappop(self._q)
22             print(n._a,n._b);
23
```

In [14]:

```
1 def test_Heap():
2     a = [5,8,2,8]
3     h = MinHeap()
4     h.insert(a)
5     print("After inserting an array",a)
6     n = h.get_top()
7     print("HeapTop has",n._a,n._b)
8     n = h.get_top_and_remove()
9     print("Removed element is",n._a,n._b)
10    n = h.get_top()
11    print("Now HeapTop has",n._a,n._b)
12    x = 3
13    n = Node(3,100)
14    h.add(n)
15    n1 = h.get_top()
16    print("HeapTop has after adding 3, 100 is",n1._a,n1._b)
17    n = Node(23,10)
18    h.add(n)
19    n1 = h.get_top()
20    print("HeapTop has after adding 23,10 is",n1._a,n1._b)
21    h.deleteAll()
```

In [16]:

```
1 test_Heap()
```

```
After inserting an array [5, 8, 2, 8]
HeapTop has 2 -2
Removed element is 2 -2
Now HeapTop has 5 -5
HeapTop has after adding 3, 100 is 3 100
HeapTop has after adding 23,10 is 3 100
3 100
5 -5
8 -8
8 -8
23 10
```

In []:

```
1
```

10.12. PROBLEM SET

10.12 Problem set

Problem 10.12.1. Implement **Grow or Shrink** as shown in figures 20.27.

Problem 10.12.2. Implement **Selection day puzzle** as shown in figures 20.28.

10.12. PROBLEM SET

Problem 10.12.3. Answer the questions below by hand and post the scanned document on Canvas.

Consider a max heap, represented by the array:

40, 30, 20, 10, 15, 16, 17, 8, 4.

Now consider that a value 35 is inserted into this heap.

Show the new heap

1

Must show both the array and array as heap

In a max heap containing n numbers, the smallest element can be found in time

Complexity: -----

Why?

2

Is the array with values

23, 17, 14, 6, 13, 10, 1, 5, 7, 12 a max- heap?

3

Show how you will build MAXHEAP from the list

$\langle 5, 3, 17, 10, 84, 19, 6, 22, 9 \rangle$.

Must show both the array and array as heap

Must show building from

1. Top to bottom
2. Bottom to top

4

Build minheap and maxheap for

$\langle 5, 13, 2, 25, 7, 17, 20, 8, 4 \rangle$.

5

Must show both the array and array as heap

Now keep on deleteing min from minheap
until minheap becomes empty

Keep on deleteing max from maxheap
until maxheap is empty

Show all the reported elements as a list

VASUDEVAMURTHY

Chapter 11

Graphviz - Graph Visualization Software

11.1 Introduction

11.2 Graphviz package

11.2. GRAPHVIZ PACKAGE

Graphviz - Graph Visualization Software
Envisioning connections

Windows
Stable and development Windows Install packages

graphviz	current stable release
Windows	graphviz-2.38.msi graphviz-2.38.zip

graphviz-2.38.zip 6/16/2015 4:16 PM Compressed (zipp...) 50,689 KB

Extract all files to a folder
example c:/jag/graphviz

```
1.dot
graph graphname {
    a -- b;
    b -- c;
    b -- d;
    d -- a;
}
```

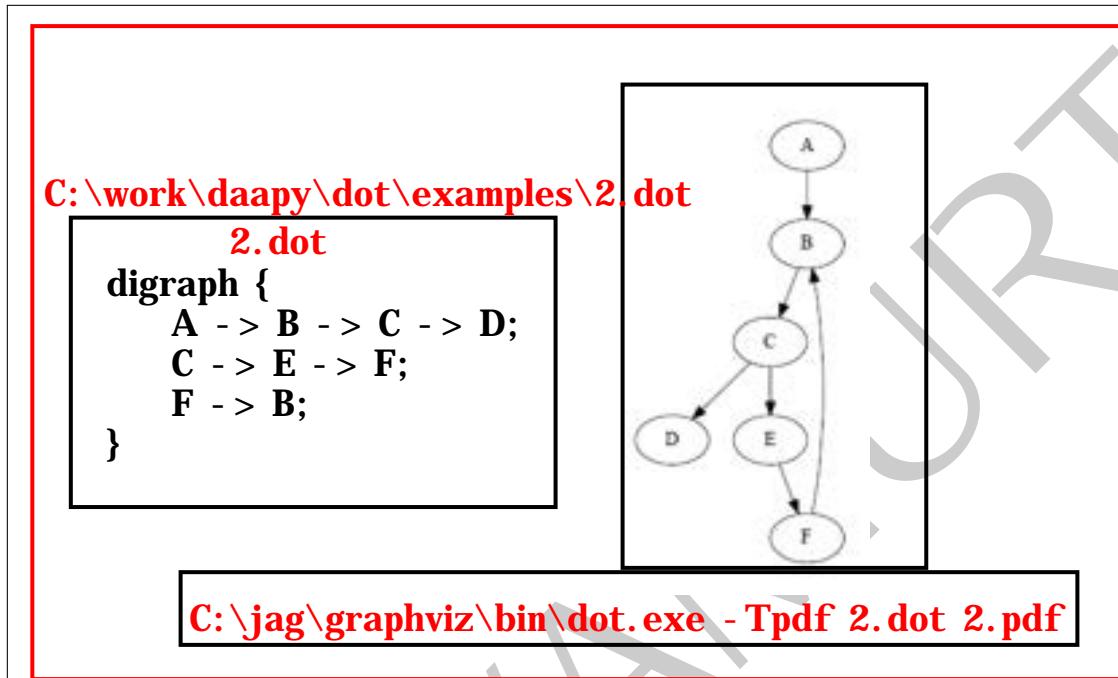
C:\jag\graphviz\bin\dot.exe -Tpdf 1.dot 1.pdf

<http://www.tonyballantyne.com/graphs.html>
<http://www.graphviz.org/Documentation/dotguide.pdf>

Figure 11.1: Installing Graphviz package

11.3 Example 1

11.3. EXAMPLE 1



C:\work\daapy\dot\examples\tree.dot

tree.dot

```
digraph G {
    {node[style=invis, label=""]; cx_30;}
    {rank=same; 20; 45; cx_30}
    {rank=same; 10; 25;}
    {rank=same; 40; 50}
    30 -> 20;
    30 -> 45;
    20 -> 10;
    20 -> 25;
    45 -> 40;
    45 -> 50;
    {edge[style=invis];
    //Distantiate nodes
    30 -> cx_30;
    20 -> cx_30 -> 45;

    //Force ordering between childs
    10:e -> 25:w;
    40:e -> 50:w;
    } C:\jag\graphviz\bin\dot.exe - Tpdf tree.dot tree.pdf
}
```

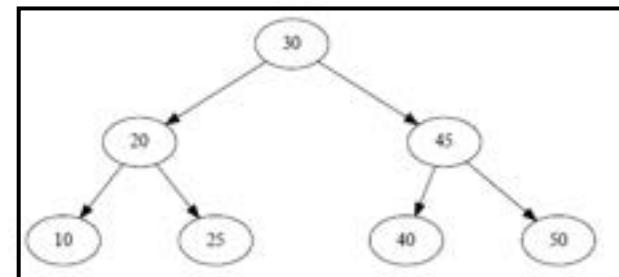


Figure 11.2: Example 1

Chapter 12

Binary Tree

12.1 Introduction

12.2 Various binary trees

12.2. VARIOUS BINARY TREES

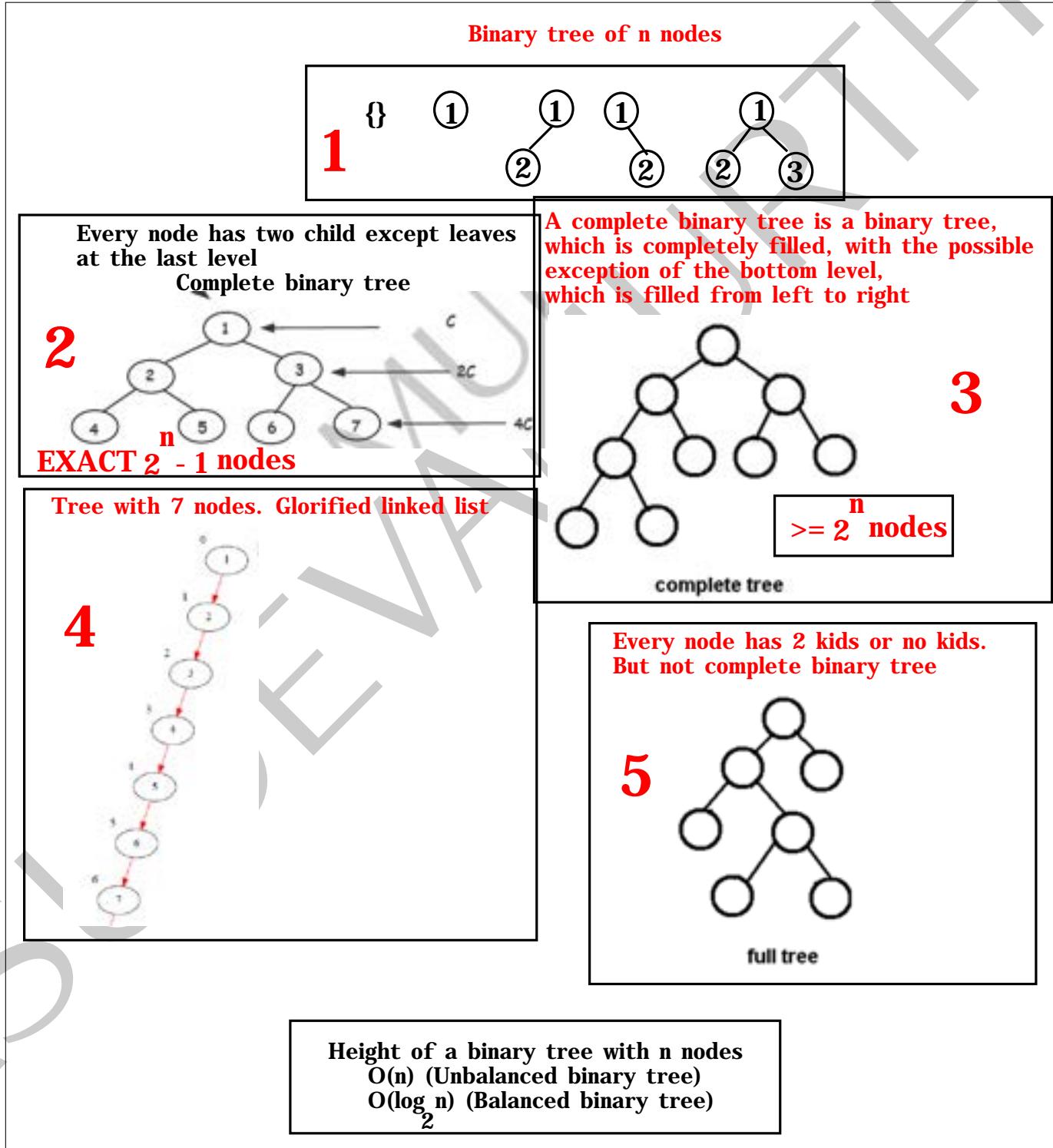


Figure 12.1: Various binary trees

12.3 Building heap data structure from user data

12.3. BUILDING HEAP DATA STRUCTURE FROM USER DATA

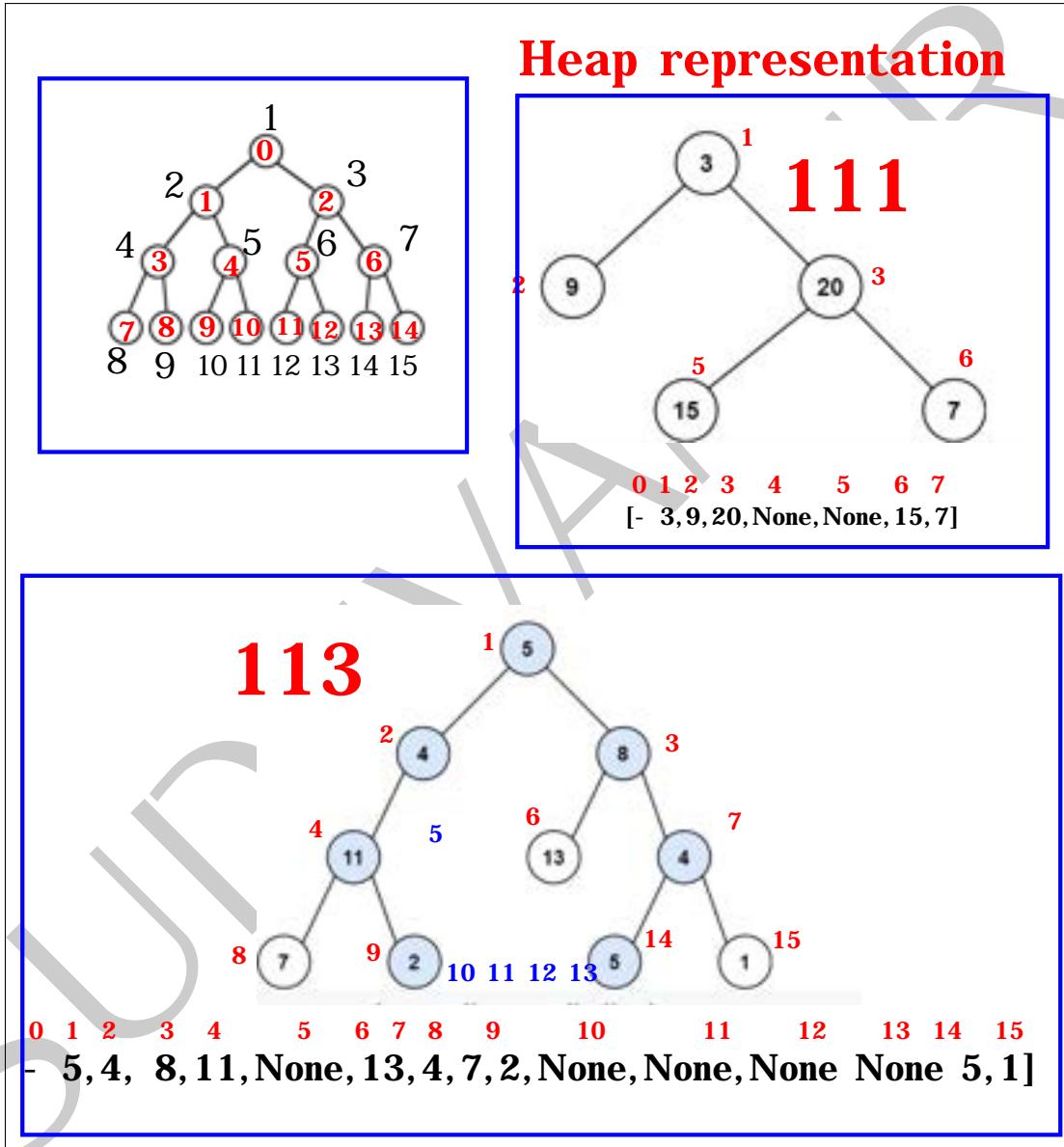


Figure 12.2: Heap data structure in memory

12.4 Building binary tree from user data

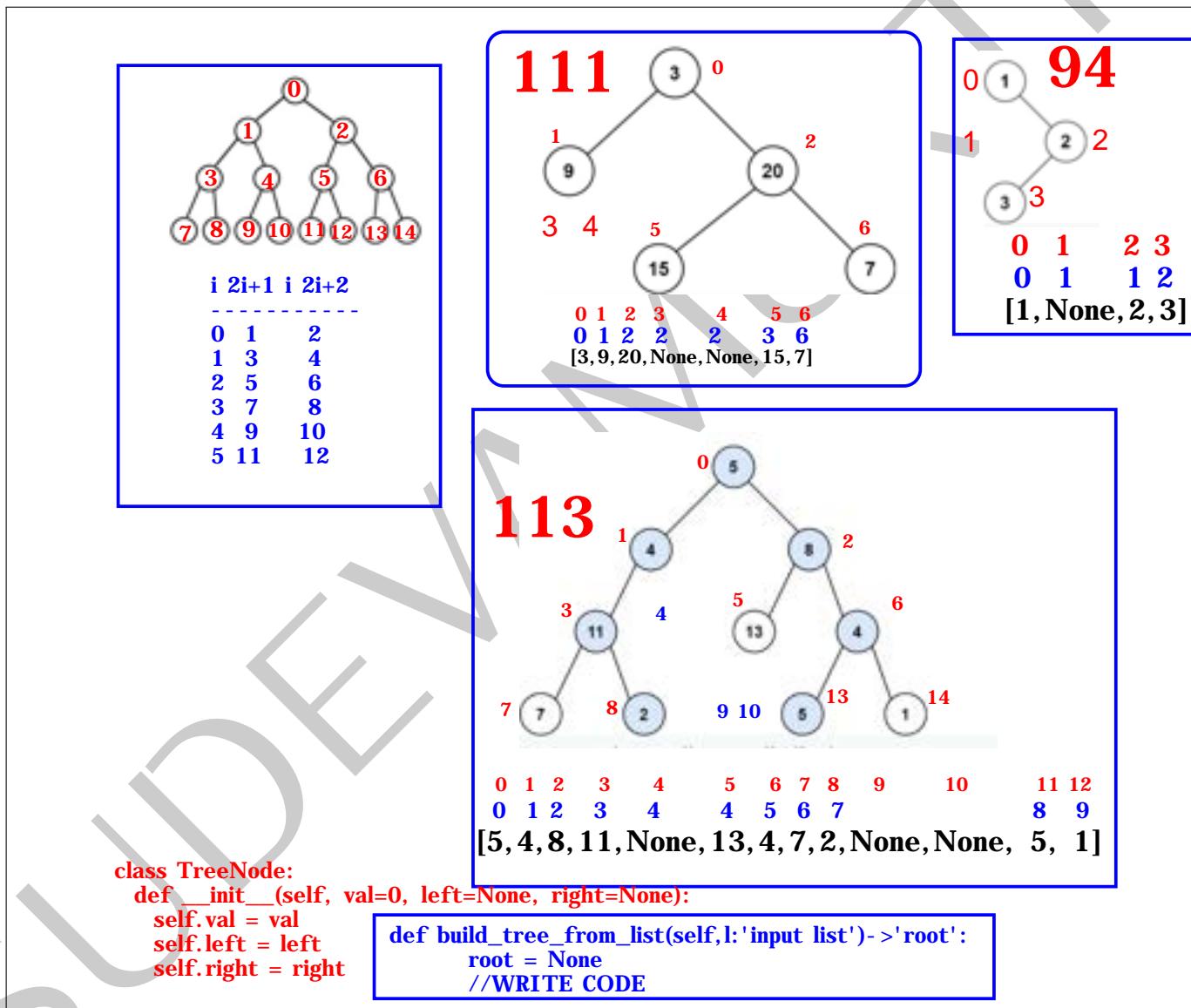


Figure 12.3: Building binary tree from user data

12.4.1 TreeNode.py

..g\OneDrive\vasu\work\py3\objects\tree\dir\TreeNode.py

```
1 #####  
2 # TreeNode.py  
3 # Author: Jagadeesh Vasudevamurthy  
4 # Copyright: Jagadeesh Vasudevamurthy 2022  
5 #####  
6 #####  
7 #####  
8 #       NOTHING CAN BE CHANGED BELOW  
9 #####  
10 class TreeNode:  
11     def __init__(self, val=0, left=None, right=None):  
12         self.val = val  
13         self.left = left  
14         self.right = right  
15
```

1

12.4.2 BuildTree.py

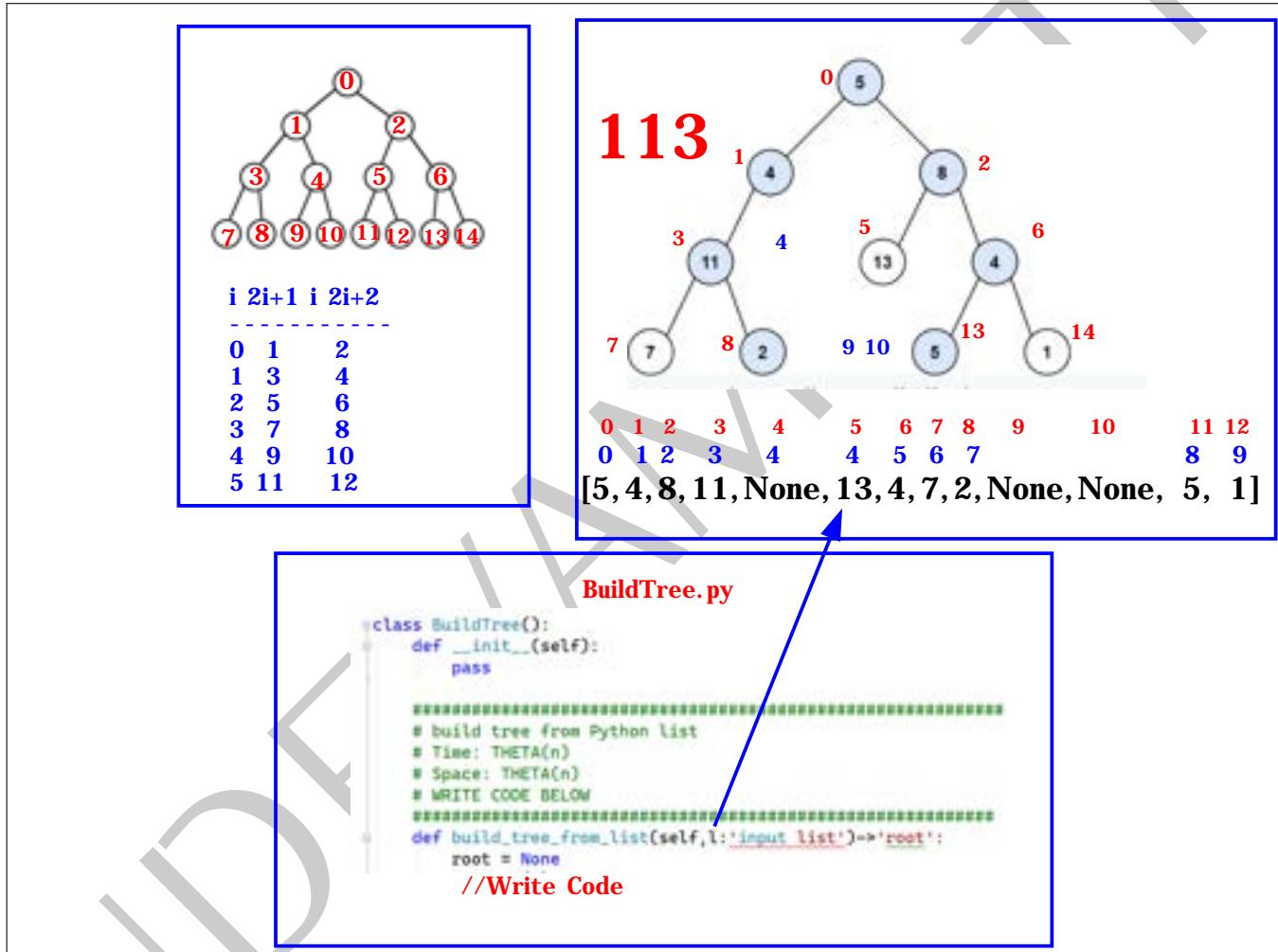


Figure 12.4: BuildTree.py

12.5 Converting a binary tree data structure to a dot file

12.5.1 WriteDot.py

WriteDot.py

```
class WriteDot():
    def __init__(self):
        pass
```

```
def write_tree_as_a_dot_file(self,
    root:'root of the tree',
    f:'file name', name:'string',
    input_list:'python list')->'string':
```

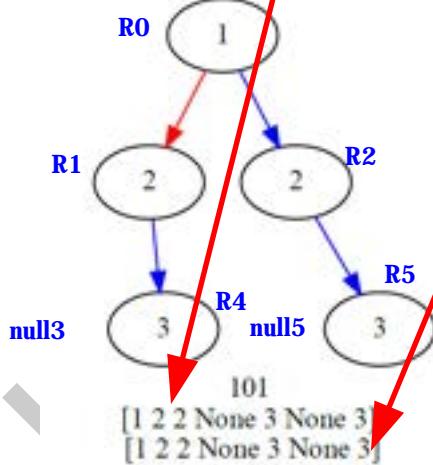


Figure 12.5: Writing dot file from tree data structure

12.6 Examples

12.6.1 Example 1

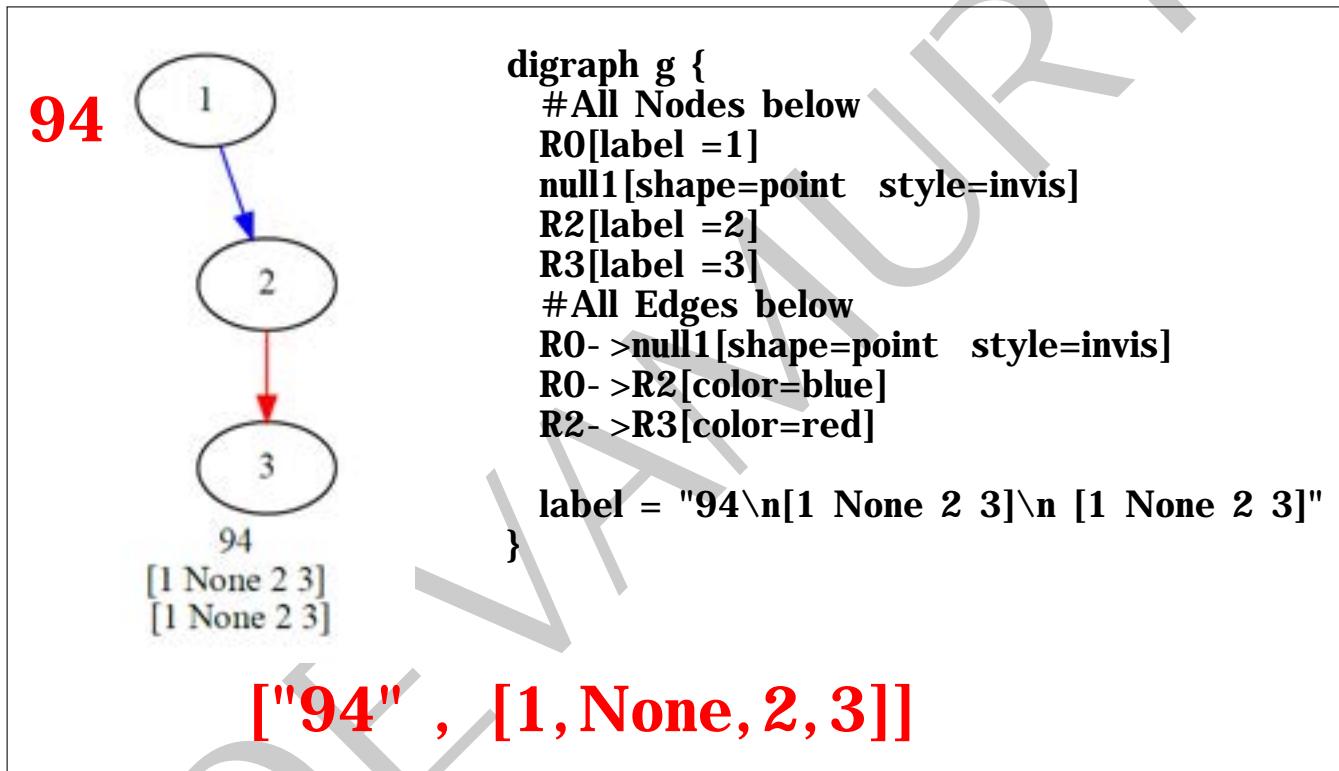


Figure 12.6: Example 1

12.6.2 Example 2

12.6. EXAMPLES

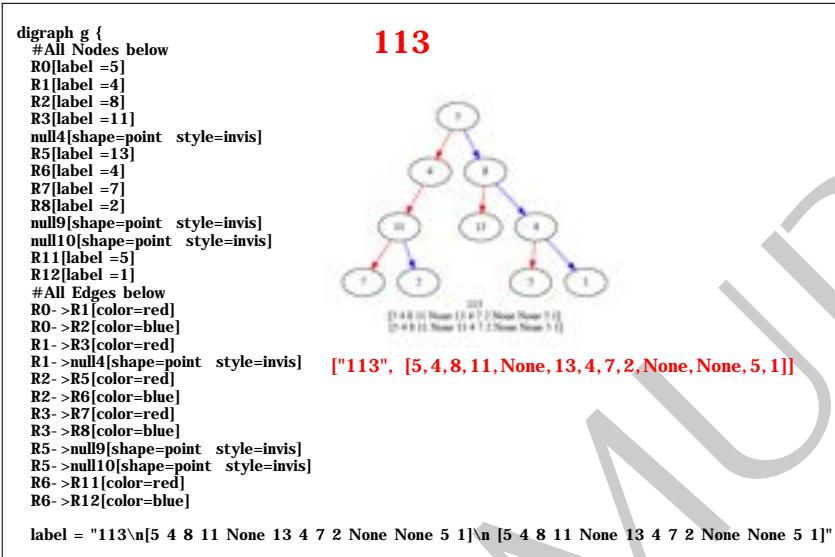


Figure 12.7: Example 2

12.6.3 Example 3

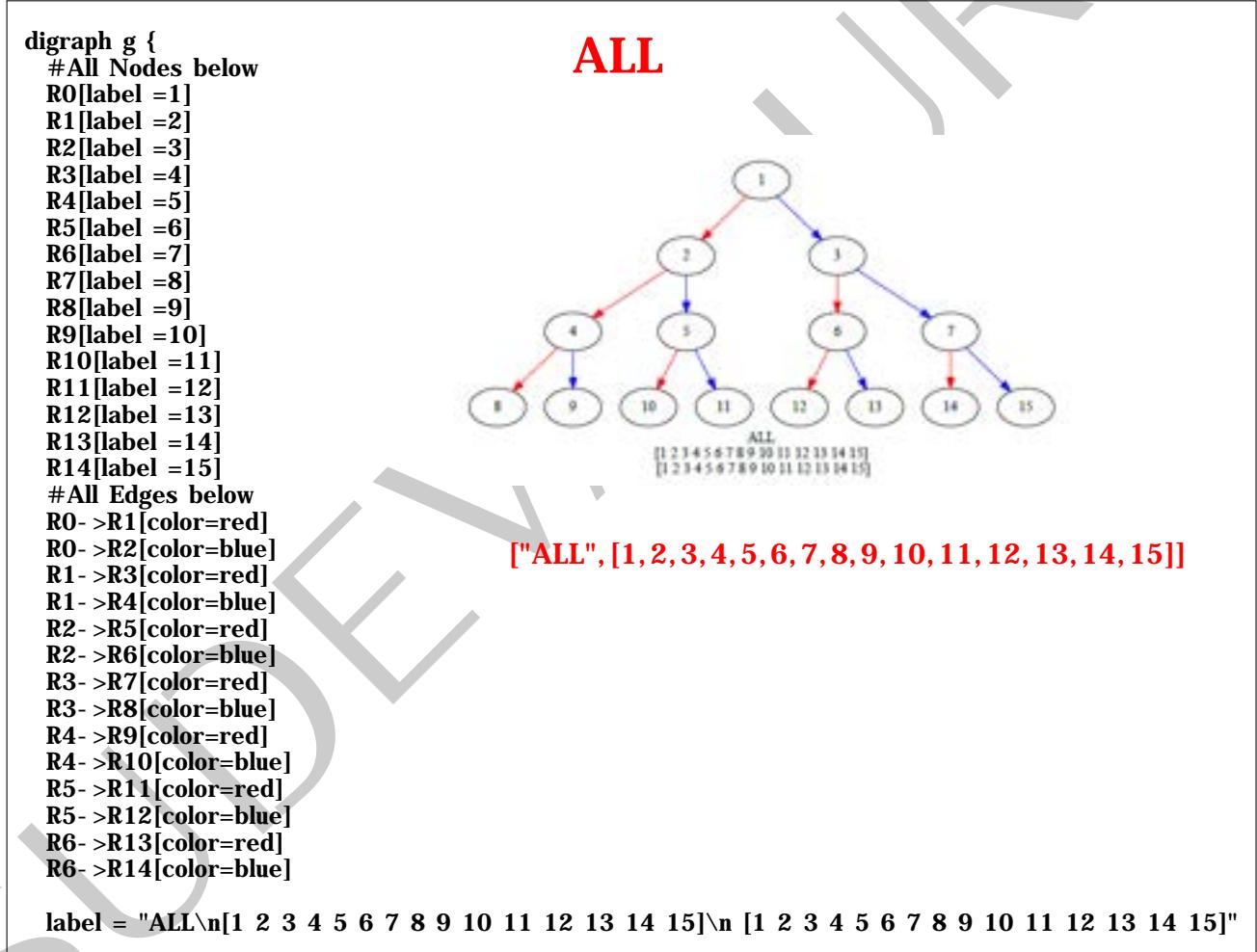


Figure 12.8: Example 3

12.7. DISPLAYING A BINARY TREE

12.6.4 Example 4

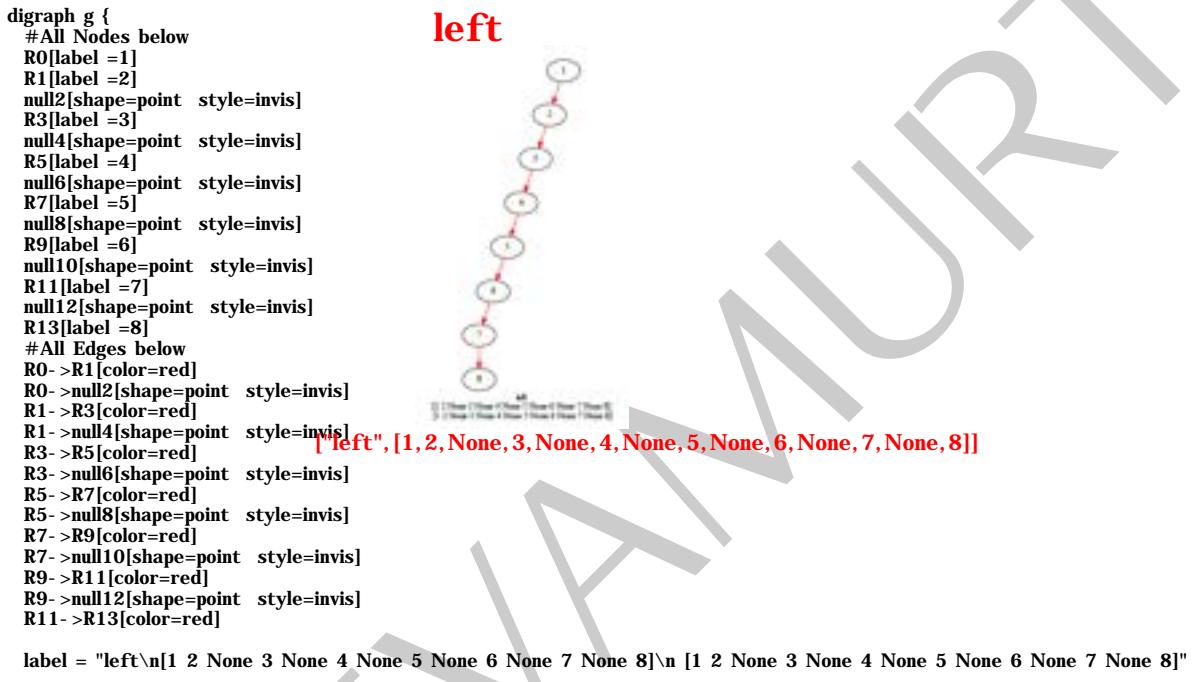


Figure 12.9: Example 4

12.7 Displaying a binary tree

Converting dot file to pdf

```
class Dot2Pdf():
    def __init__(self):
        #change these 2 lines
        self._output_dir = "C:\\\\Users\\\\jag\\\\OneDrive\\\\vasu\\\\work\\\\py3\\\\objects\\\\tree\\\\notebook\\\\dot\\\\"
    if (True):
        self._dot2pdf_exe = '"C:\\\\Program Files\\\\Graphviz\\\\bin\\\\dot.exe"'
        self._display_on_screen = False # make it True only on Jupyter
    else:
        self._dot2pdf_exe = ""

    def execute_dot_2_pdf(self, f:'string'):
        pass
```

Figure 12.10: Displaying a binary tree from the dot file

12.7. DISPLAYING A BINARY TREE

12.7.1 Dot2Pdf.py

```
1 #####  
2 # Dot2Pdf.py  
3 # Author: Jagadeesh Vasudevamurthy  
4 # Copyright: Jagadeesh Vasudevamurthy 2022  
5 #####  
6 #####  
7 #####  
8 # All imports  
9 # Comment all these imports in jupyter note book  
10 #####  
11 import os  
12 from IPython.display import IFrame  
13  
14 class Dot2Pdf():  
15     def __init__(self):  
16         #change these 2 lines  
17         self._output_dir = "C:\\\\Users\\\\jag\\\\OneDrive\\\\vasu\\\\work\\\\py3\\\\objects\\\\tree\\\\notebook\\\\dot\\\\"  
18         if (True):  
19             self._dot2pdf_exe = '"C:\\\\Program Files\\\\Graphviz\\\\bin\\\\dot.exe"'  
20             self._display_on_screen = False # make it True only on Jupyter  
21  
22     else:  
23         self._dot2pdf_exe =""  
24  
25 #####  
26 # NOTHING CAN BE CHANGED BELOW  
27 #####  
28 #####  
29 #####  
30 # Get output directory  
31 #####  
32     def output_dir(self)->'string':  
33         return self._output_dir  
34  
35 #####  
36 # Calling an dot2pdf executable  
37 # "C:\\Program Files\\Graphviz\\bin\\dot.exe" -Tpdf C:\\scratch\\outputs\\tree \\94.dot -o C:\\scratch\\outputs\\tree\\94.pdf  
38 #####  
39     def execute_dot_2_pdf(self, f:'string'):  
40         if (len(f)):  
41             dot_file = f +".dot"  
42             pdf_file = f +".pdf"  
43             s = self._dot2pdf_exe + " -Tpdf " + dot_file + " -o " + pdf_file  
44             print("Executing", s)  
45             os.system(s)  
46             if(self._display_on_screen):  
47                 rpath = os.path.relpath(pdf_file)
```

```
...ag\OneDrive\vasu\work\py3\objects\tree\dir\Dot2Pdf.py  
48     display(IFrame(rpath, width=800, height=400))  
49  
50  
51
```

2

VASUDEVAMURTHY

12.8 Tree traversal

12.8.1 Example 1

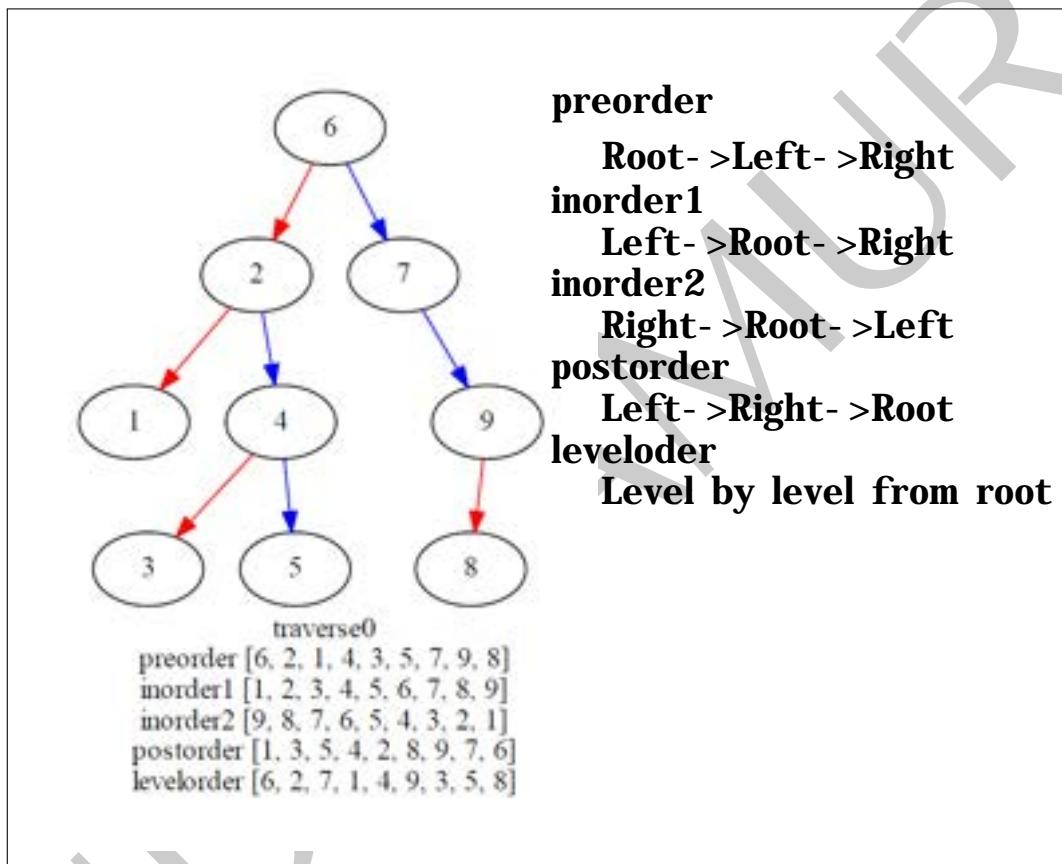


Figure 12.11: Example 1

12.8.2 Example 2

12.8. TREE TRAVERSAL

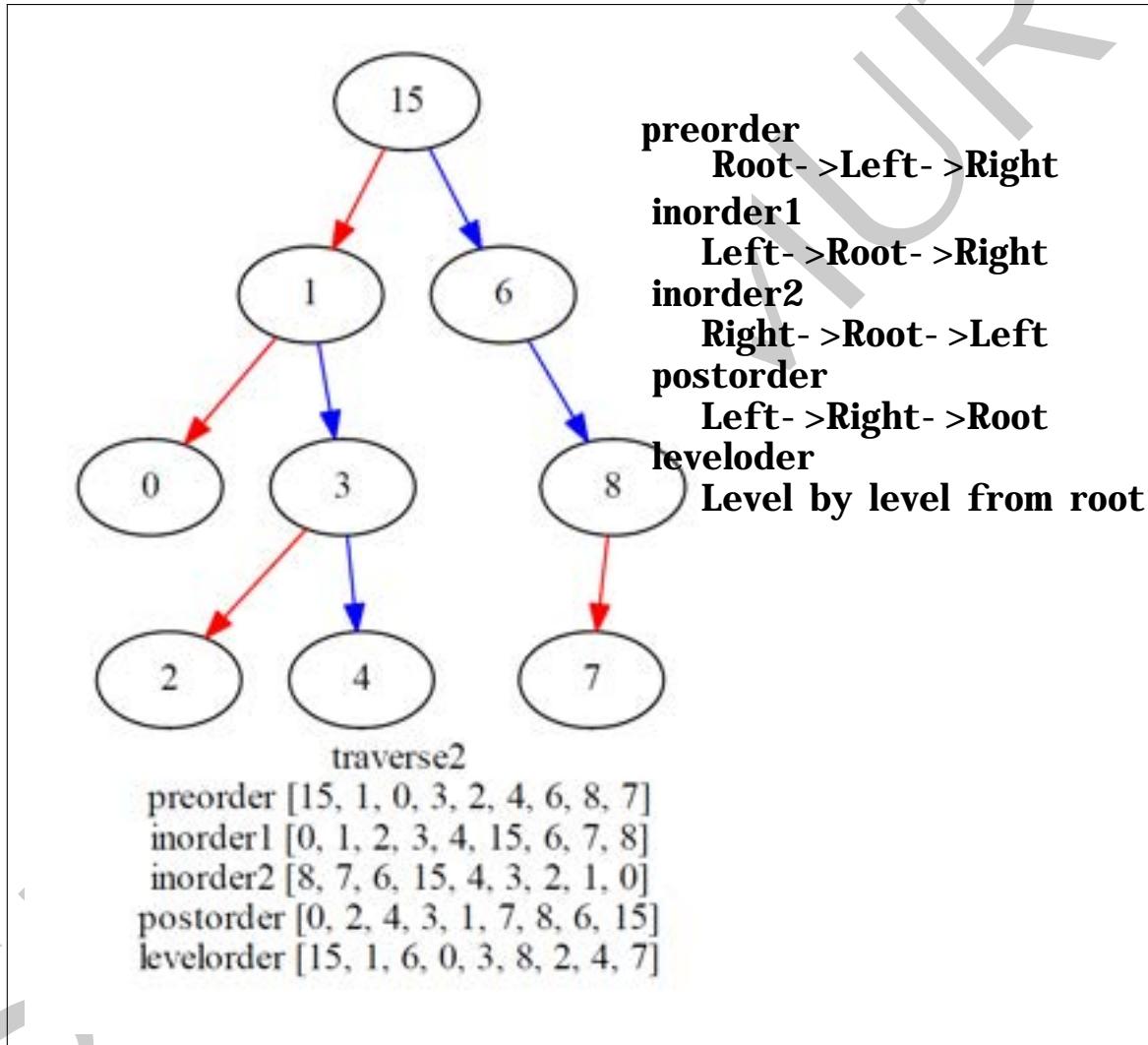


Figure 12.12: Example 2

12.8.3 Example 3

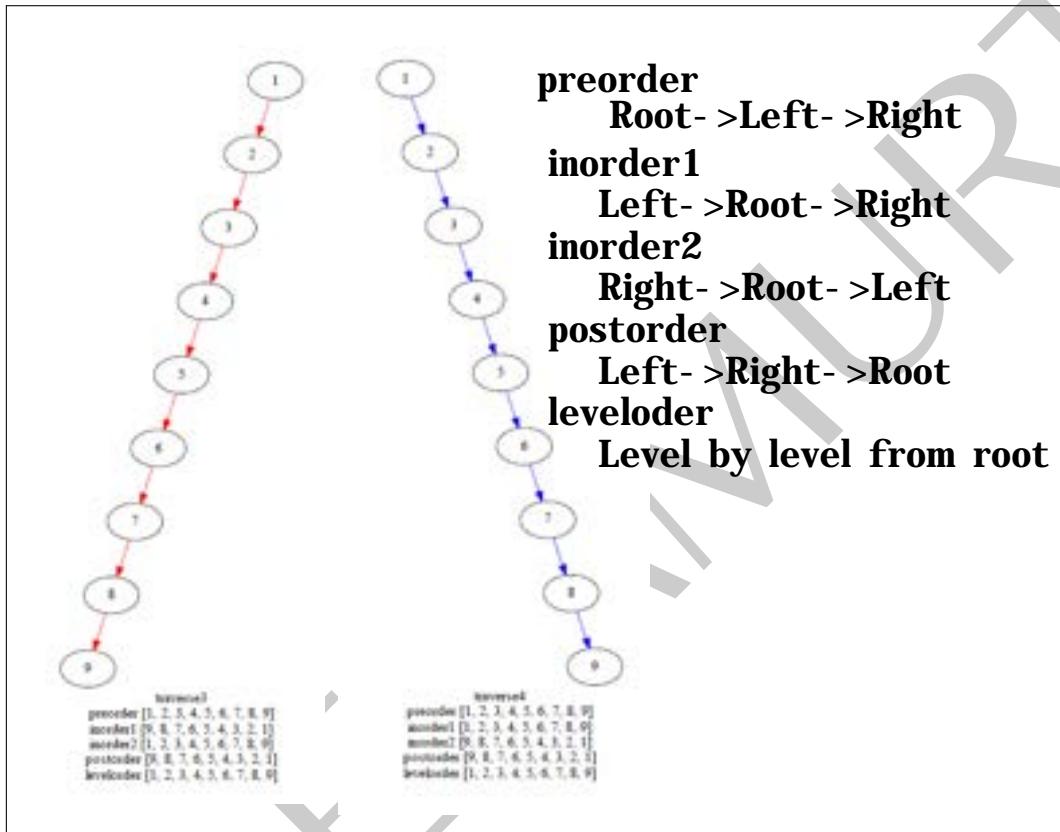


Figure 12.13: Example 3

12.9 Algorithms on Tree

12.9.1 Postorder traversal: Number of nodes in a binary tree

12.9. ALGORITHMS ON TREE

POSTORDER TRAVERSAL (LRV)

Number of nodes in a binary tree

$\text{num} = 1 + \text{Num nodes in left tree} + \text{Num nodes in right tree}$

Number of nodes of null tree = 0

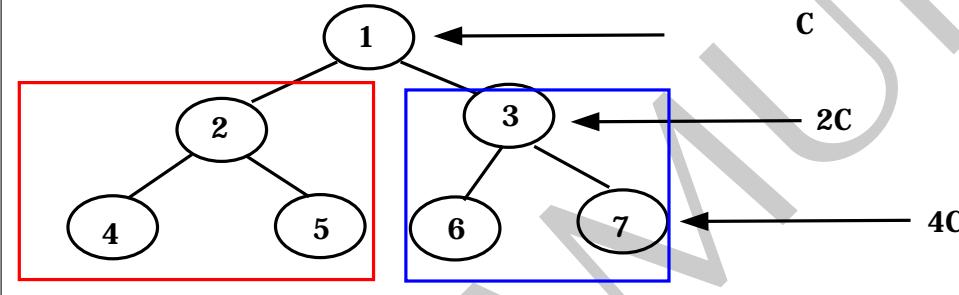


Figure 12.14: Number of nodes in a binary tree

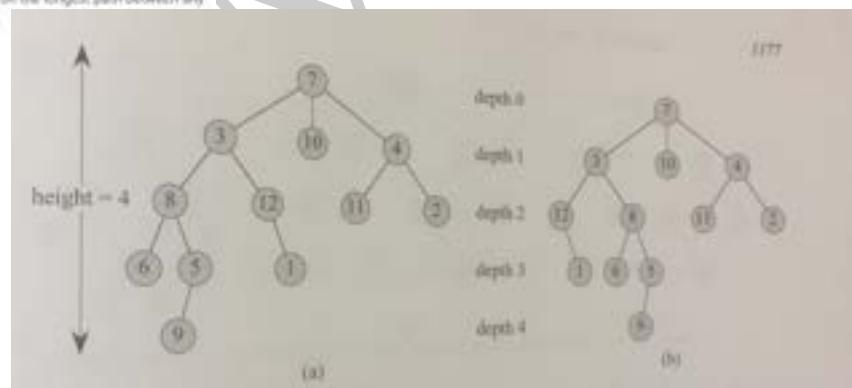
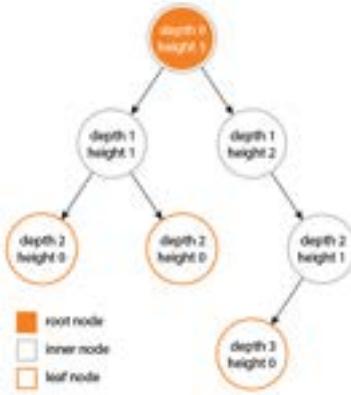
12.9.2 Height and depth of a tree

Depth and Height of a binary tree

104. Maximum Depth of Binary Tree

A binary tree's maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

- + The **height** of a tree would be the height of its root node, or equivalently, the depth of its deepest node.
- + The **diameter** (or **width**) of a tree is the number of nodes on the longest path between any two leaf nodes. The tree below has a diameter of 6 nodes.



The height of a tree is equal to the max depth of a tree.
The depth of a node and the height of a node are not necessarily equal.
See Figure B.6 of the 3rd Edition of Cormen et al. for an illustration of these concepts.

Depth: Draw a horizontal line at the root position and treat this line as ground. So the depth of the root is 0, and all its children are grown downward so each level of nodes has the current depth + 1.

Height: Same horizontal line but this time the ground position is external nodes, which is the leaf of tree and count upward.

Figure 12.15: Height and depth of a tree

12.9.3 Postorder traversal: Depth or Height of a binary tree

Postorder traversal: LRV

Depth of a node n is the LONGEST downward path (number of edges) from n to leaf node

Depth of a root = 0

```
Depth(node n)
if (n is root)
    return 0
return max(depth(n.left), depth(n.right)) + 1
```

Time: THETA(n)

Space: O(height of tree)

LeetCode definition

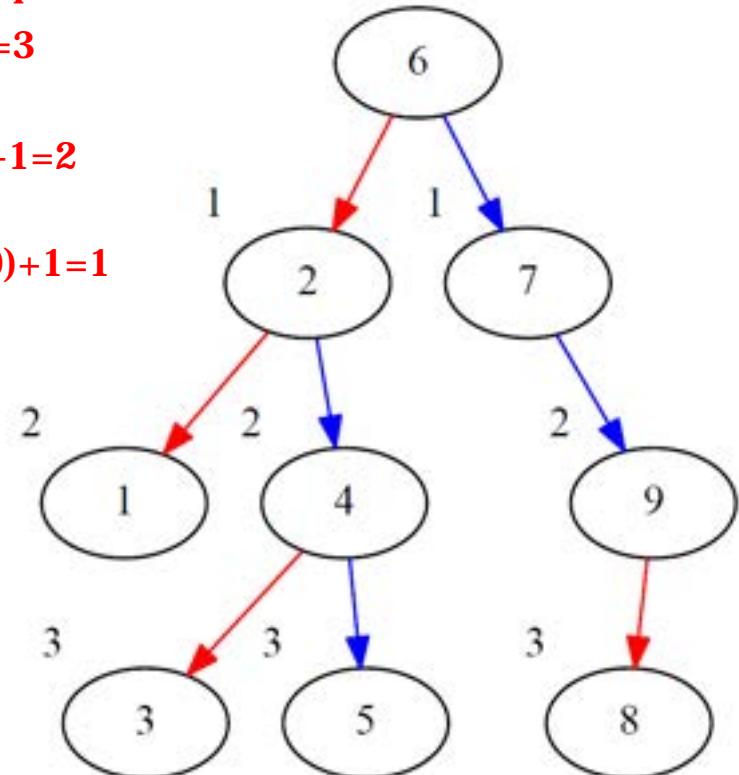
The maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

Given binary tree [3,9,20,null,null,15,7],

#LeetCode definition. Depth of root is 10

$$\begin{array}{c}
 3 \text{ } \max(1, 2)+1=3 \\
 / \backslash \\
 \max(0, 0)+1=1 \text{ } 9 \text{ } 20 \text{ } \max(1, 1)+1=2 \\
 / \backslash \\
 \max(0, 0)+1=1 \text{ } 15 \text{ } 7 \text{ } \max(0, 0)+1=1
 \end{array}$$

return its depth = 3.



prop0

Num Nodes = 9

Num Leaves = 4

Max Depth = 4

12.9. ALGORITHMS ON TREE

12.9.4 Height of a node in a tree

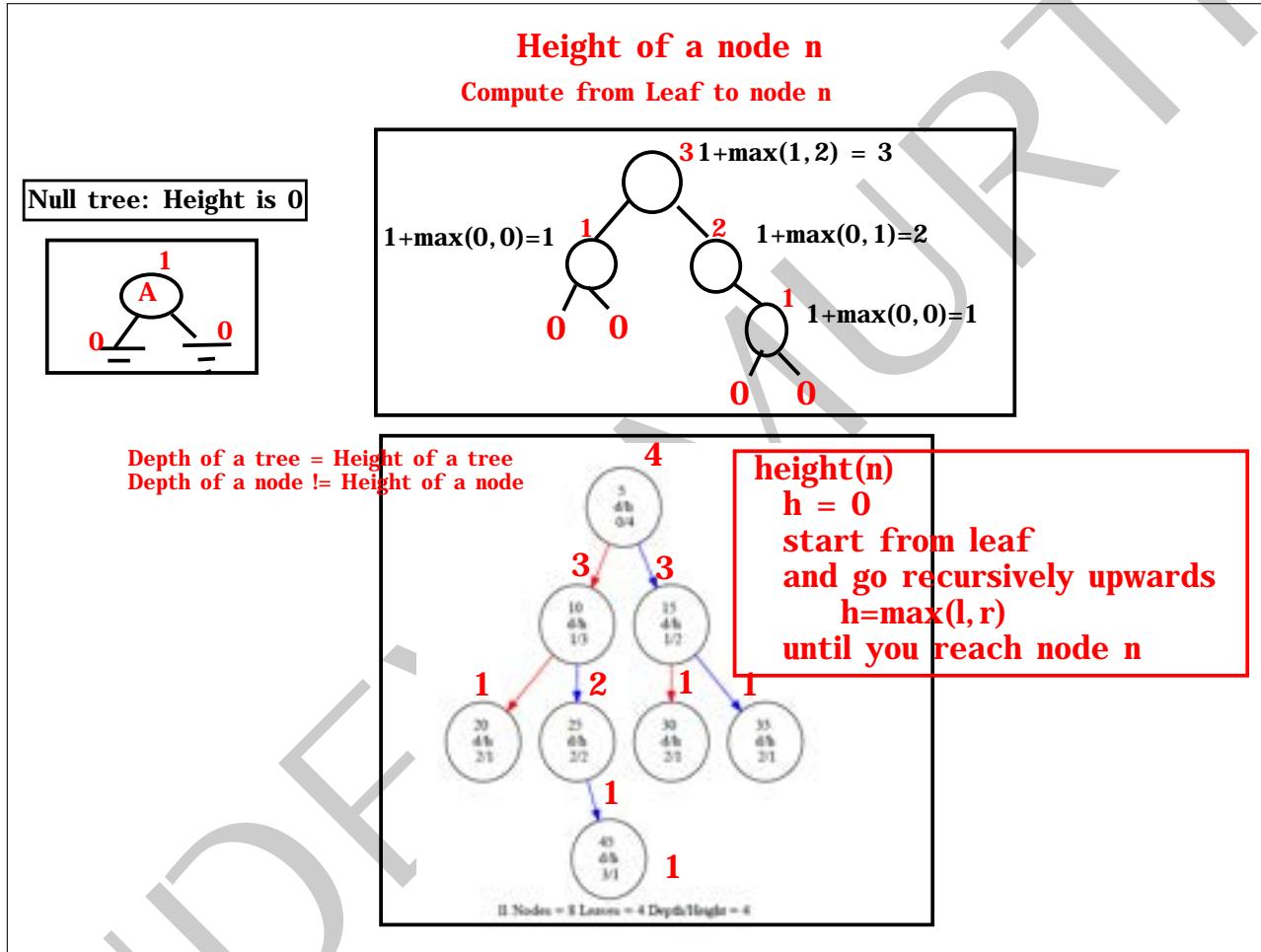


Figure 12.17: Height of a node in a tree

12.9.5 Depth of a node in a tree

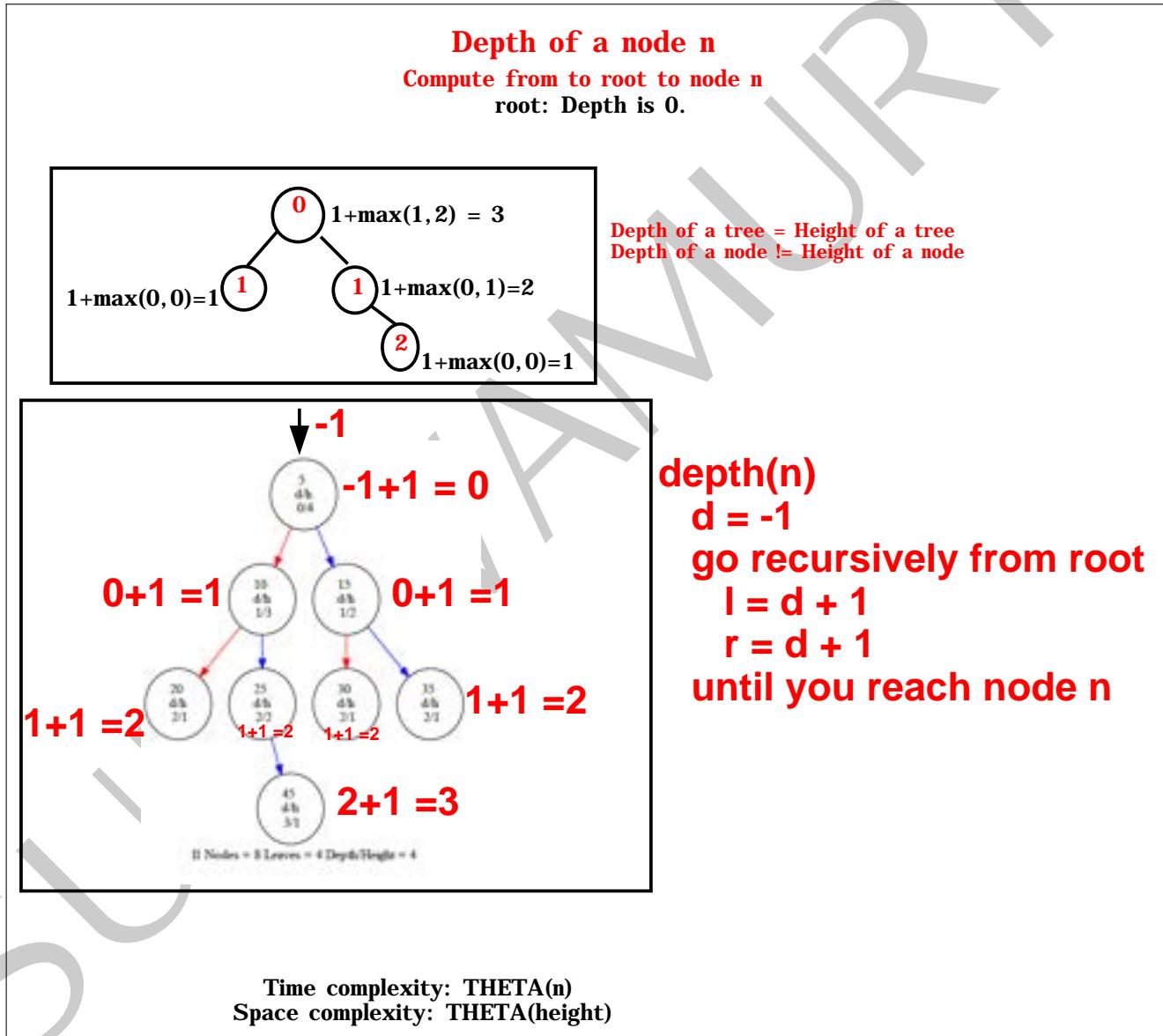


Figure 12.18: Depth of a node in a tree

12.9. ALGORITHMS ON TREE

12.9.6 Postorder tree traversal: Deleting a tree

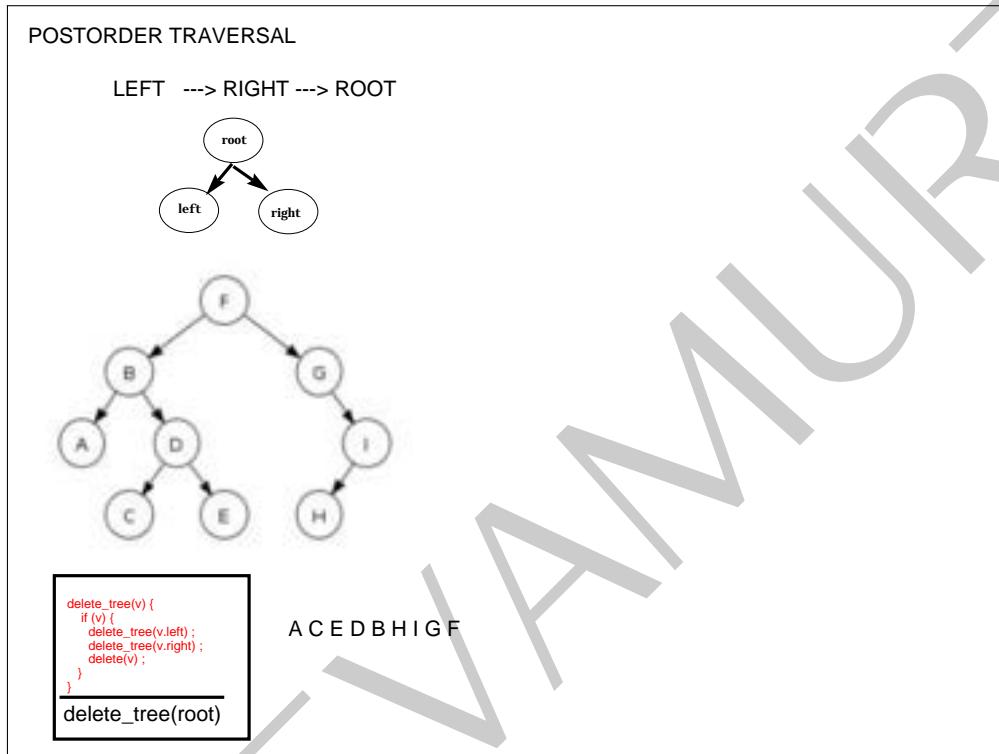


Figure 12.19: Need for postorder tree traversal

12.9.7 Preorder traversal: Printing all paths of a tree

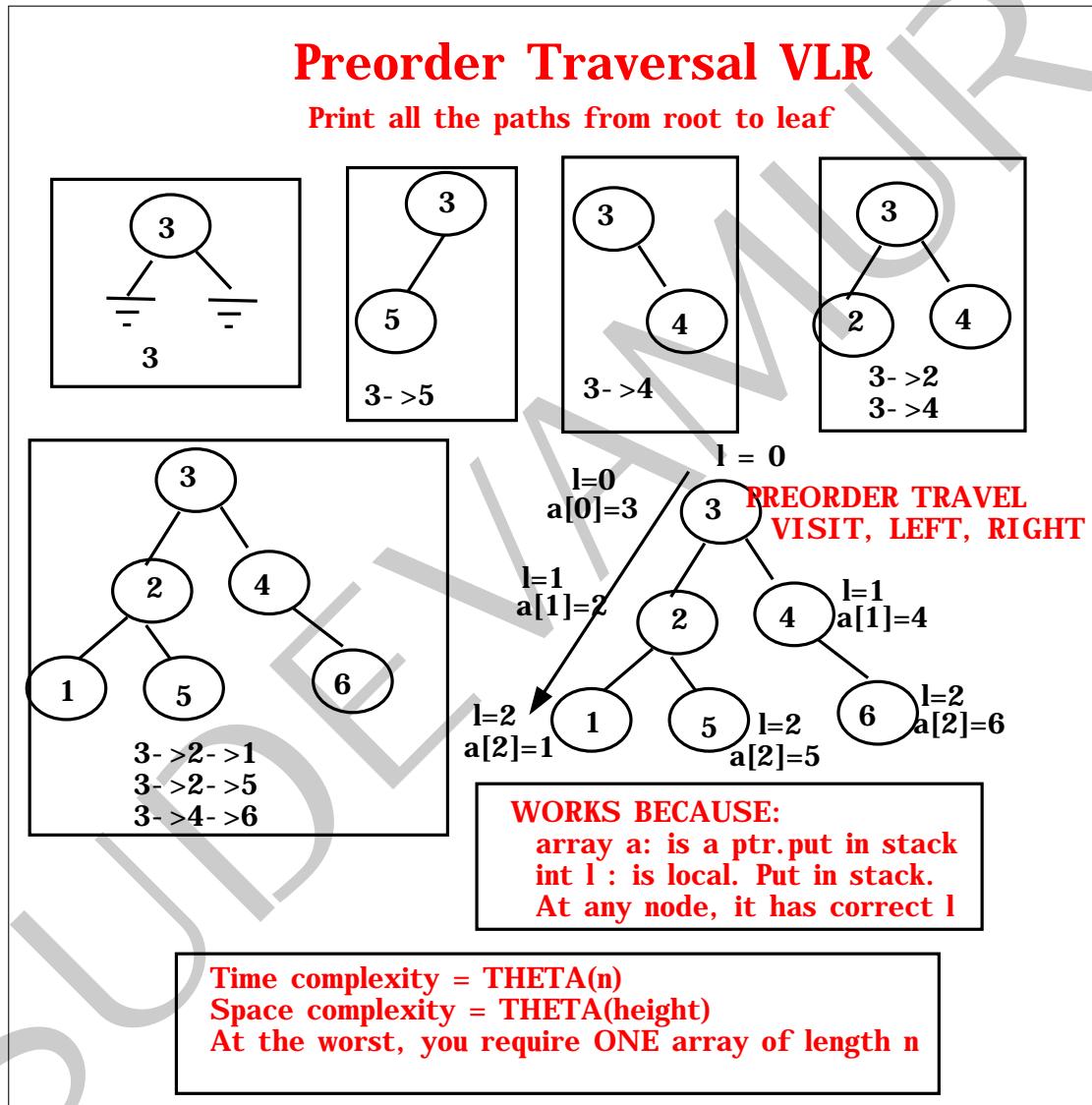


Figure 12.20: Printing all paths of a tree

12.10. STATISTICS

12.9.8 Inorder traversal

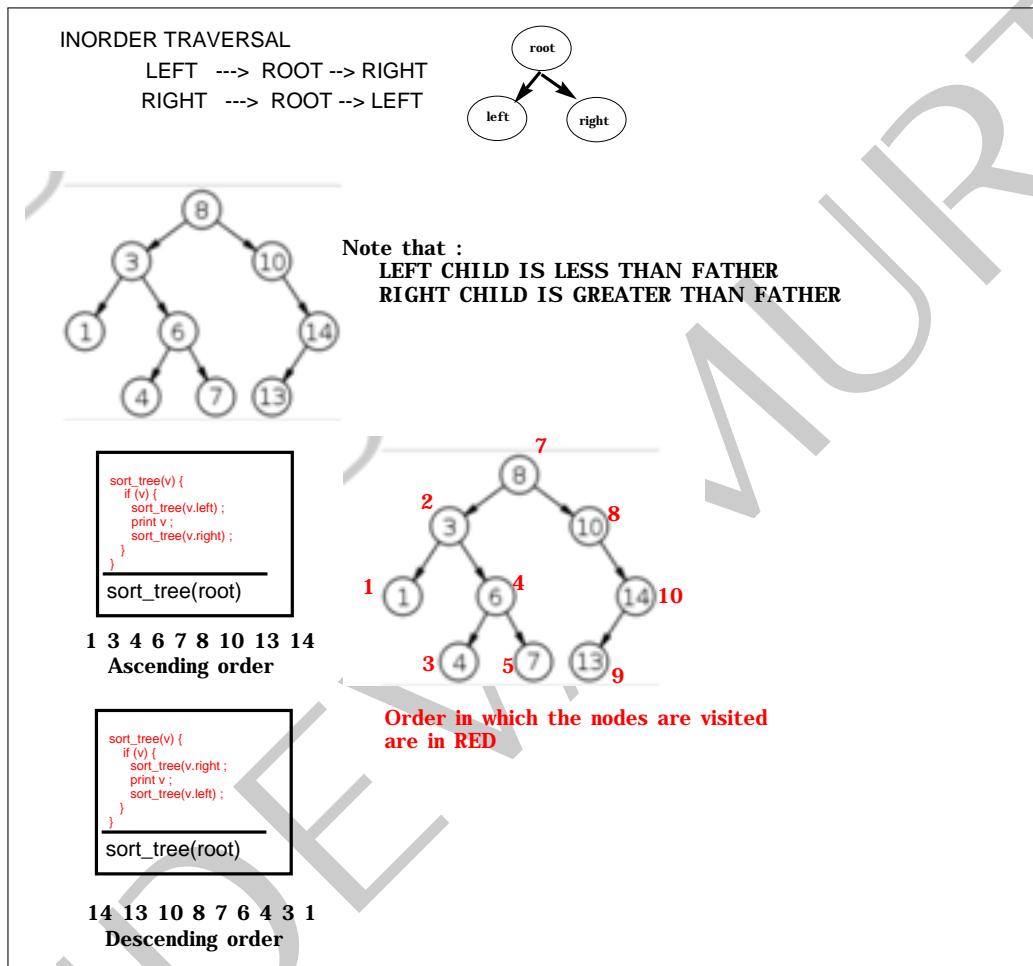


Figure 12.21: Need for inorder tree traversal

12.10 Statistics

12.11 Huffman coding

a	45000
b	13000
c	12000
d	16000
e	9000
f	5000

1000000

a = 000 b = 001 c = 010 d = 011 e = 100 f = 101

Total # bits = 1000000 * 3 = 3000000

Suppose

a = 0 b = 101 c = 100 d = 111 e = 1101 f = 1100

**Total # bits = 1(45) + 3(13 + 12 + 16) + 4(9 + 5)
= 224 * 1000 = 224000 bits**

3000000 -> 224000 = 25% reduction

How do we get variable length code, like

a = 0 b = 101 c = 100 d = 111 e = 1101 f = 1110

12.11. HUFFMAN CODING

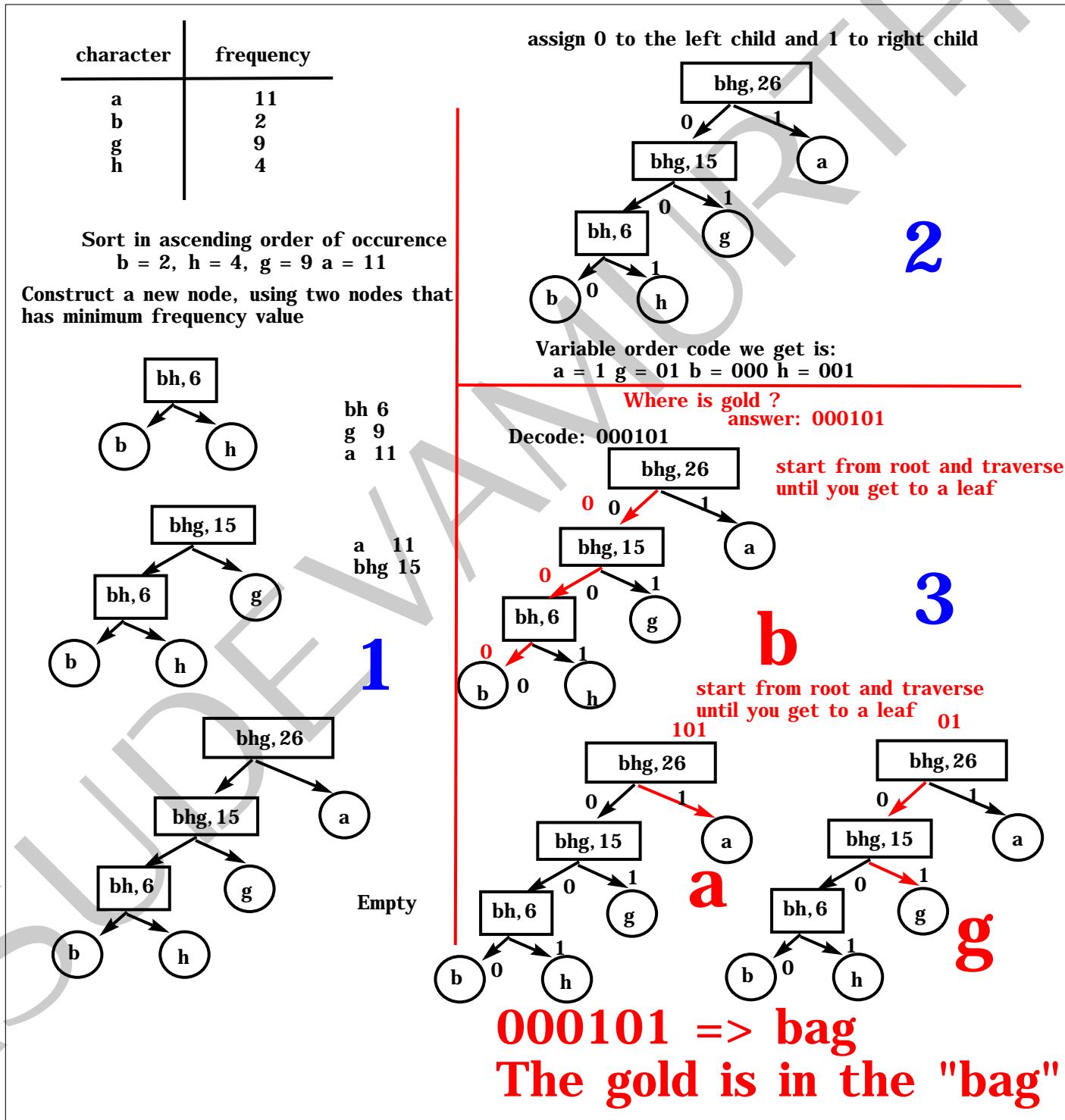


Figure 12.23: Compression using greedy algorithm

12.12 Implementing Huffman coding

DATA STRUCTURE

```
class Node():
    def __init__(self, freq, data, l = None, r = None):
        # Nothing can be added in class Node
        # Must use Node class. Hacker rank fails without this Node
        self.freq= freq #Note public
        self.data=data #Note public
        self.left = l #Note public
        self.right = r #Note public
    ## YOU CAN ADD ANY NUMBER OF PRIVATE FUNCTIONS ONLY
```

```
class Huffman():
    def __init__(self, show = False, f = None):
        #Nothing can be changed here
        self._s = None
        self._show = show
        self._f = f
        self._id = []
    ##YOU CAN ADD YOUR PRIVATE VARIABLE HERE

    ##### All public functions below #####
    # NOTHING CAN BE CHANGED BELOW
    #####
    def encode(self, s:'str')->'str':
        self._s = s
        return self._encode()
    def decode(self, e:'str')->'str':
        return self._decode(e)
```

Implement 2 functions

Figure 12.24: Data structure used

s = aaabbggggghhhhaaaggggaaaa_+@#

Figure 12.25: Six steps of the algorithm

12.12. IMPLEMENTING HUFFMAN CODING

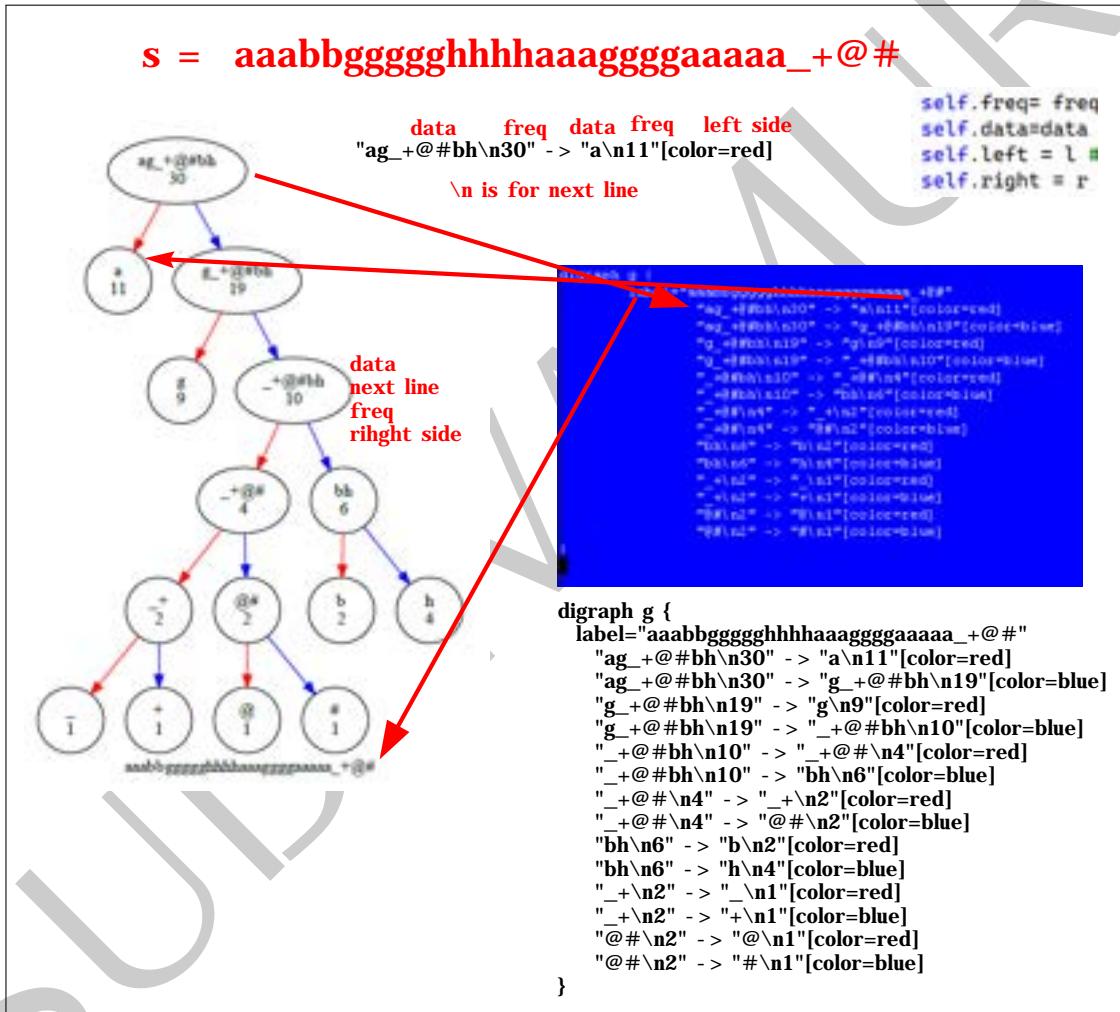


Figure 12.26: Writing dot file

12.12.1 Leetcode 535. Encode and Decode TinyURL

535. Encode and Decode TinyURL

<https://leetcode.com/problems/encode-and-decode-tinyurl/>

Success Details >

Runtime: 384 ms, faster than 5.15% of Python3 online submissions for Encode and Decode TinyURL.

Memory Usage: 14.3 MB, less than 8.99% of Python3 online submissions for Encode and Decode TinyURL.

```
#####
# All imports
import heapq
#####

class Huffman():
    def __init__(self, show = False, f = None):
        #Nothing can be changed here
        self._s = None
        self._show = show
        self._f = f
        self._id = []
        ##YOU CAN ADD YOUR PRIVATE VARIABLE HERE

#####
##LEETCODE
## https://leetcode.com/problems/encode-and-decode-tinyurl/
## Step1: Delete all code and comment in Leetcpcde
## Step2: Cut and paste this file in Leetcode
## Step3: Make False to True
## All tests must pass in one shot.
## Post screen shot of Leetcode passed as put
#####
if (True):
    Codec = Huffman
```

WRITE CODE

Must be True

Chapter 13

Binary Search Tree

13.1 Introduction

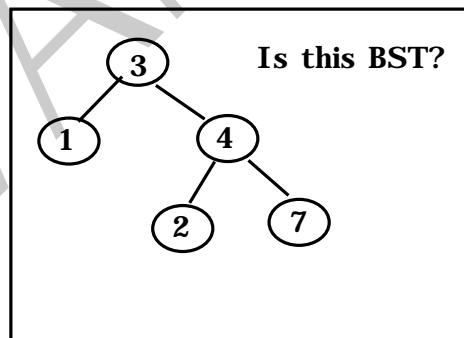
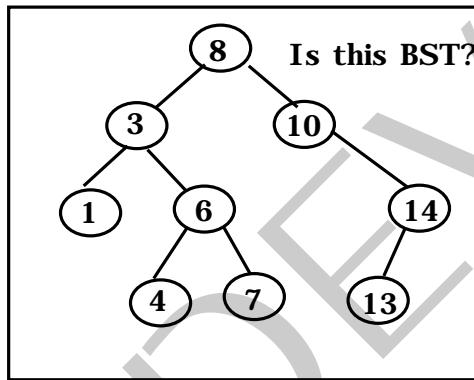
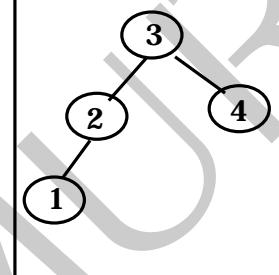
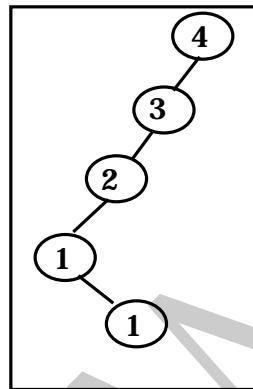
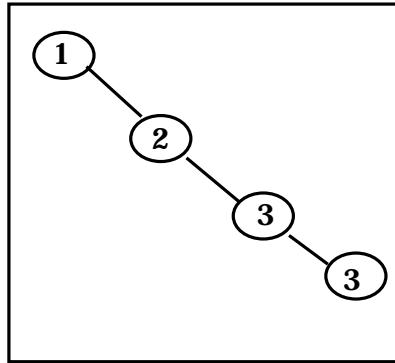
13.2 Definition of BST

13.2. DEFINITION OF BST

Binary Search Tree

A BST T is a binary tree such that either T is empty or

1. Each item in Left(T) < root(T)
2. Each item in Right(T) >= root(T)
3. Left(T) and Right(T) are BST



HEAP

A (Max)Heap T is a complete binary tree such that either T is empty or

1. Each item in Left(T) <= root(T)
2. Each item in Right(T) <= root(T)
3. Left(T) and Right(T) are heaps

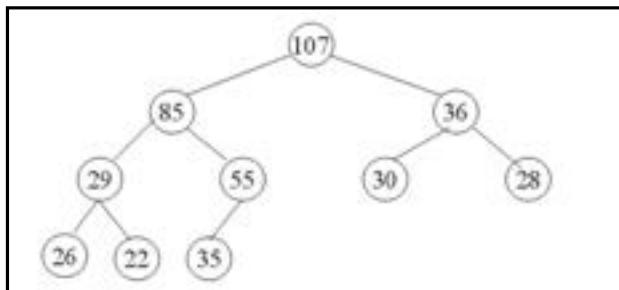


Figure 13.1: Definition BST

13.3 Is BST?

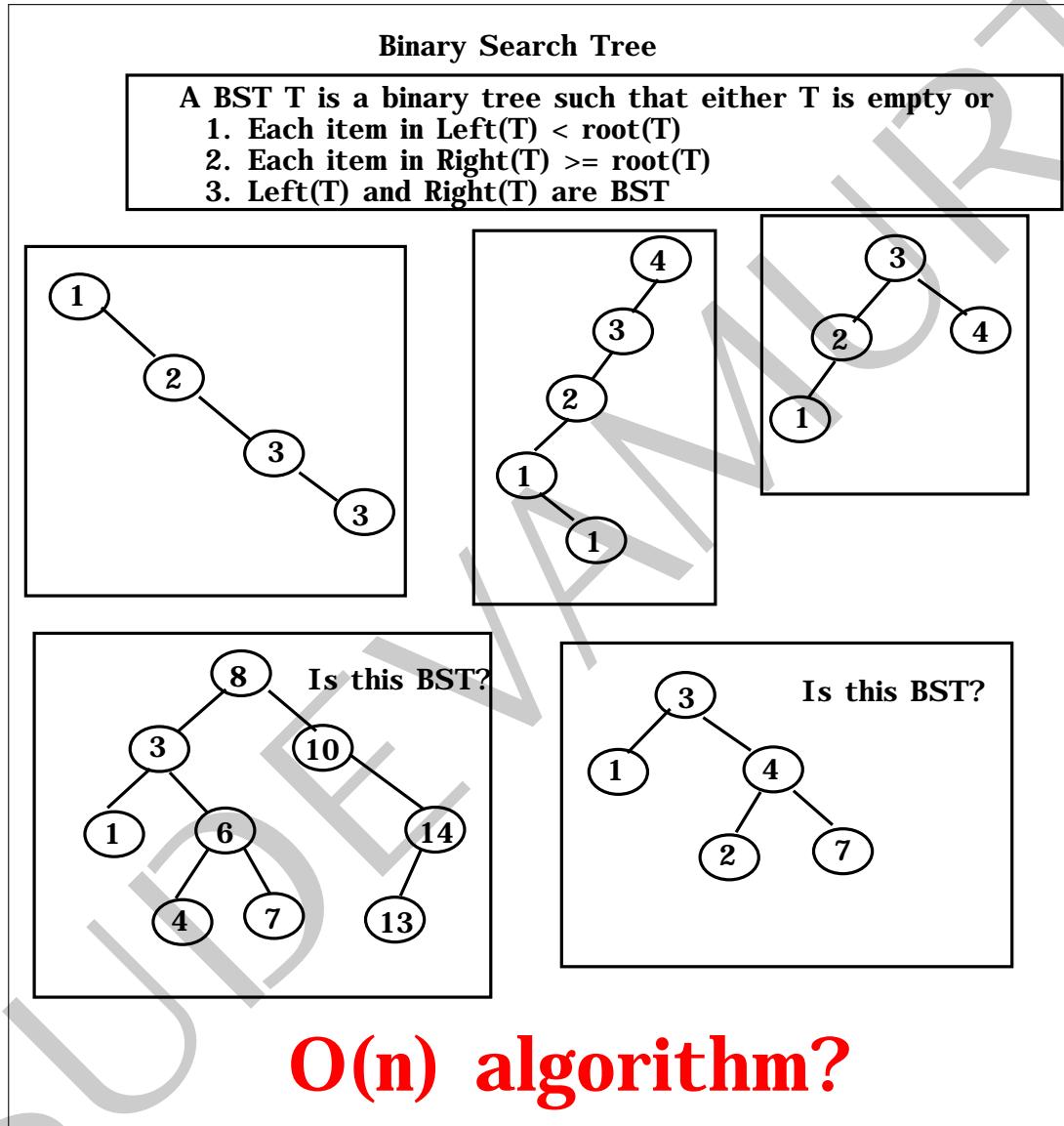


Figure 13.2: Testing a BST

13.4 Building BST

13.5. SEARCHING AN ELEMENT IN BST

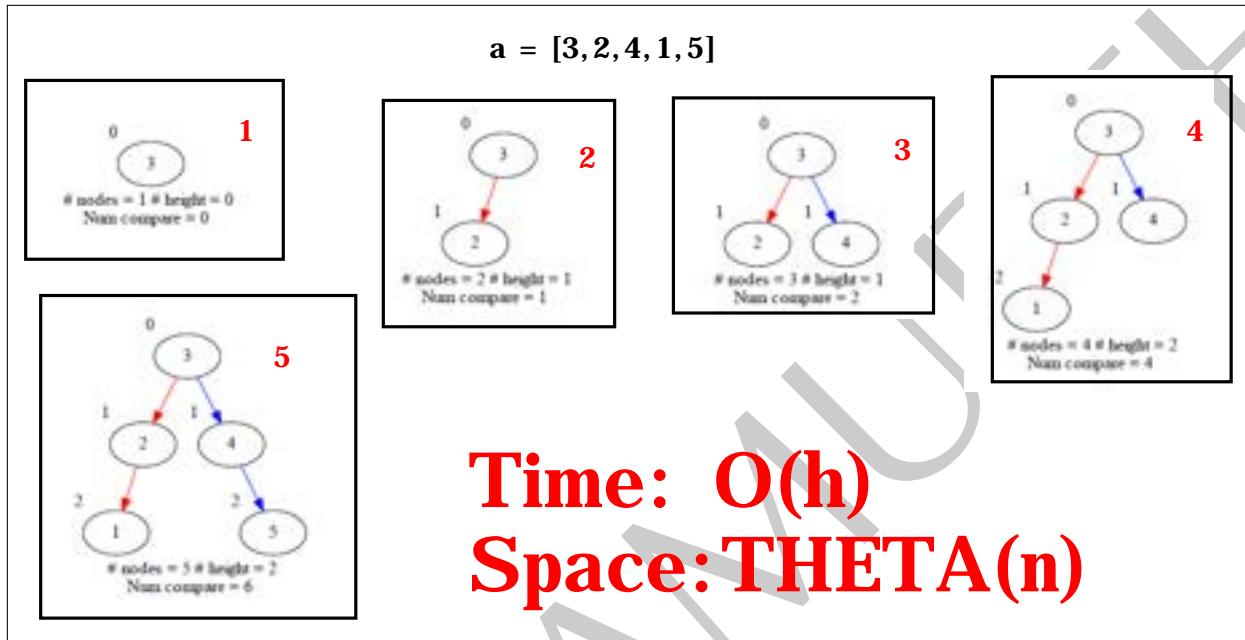


Figure 13.3: Building BST

13.5 Searching an element in BST

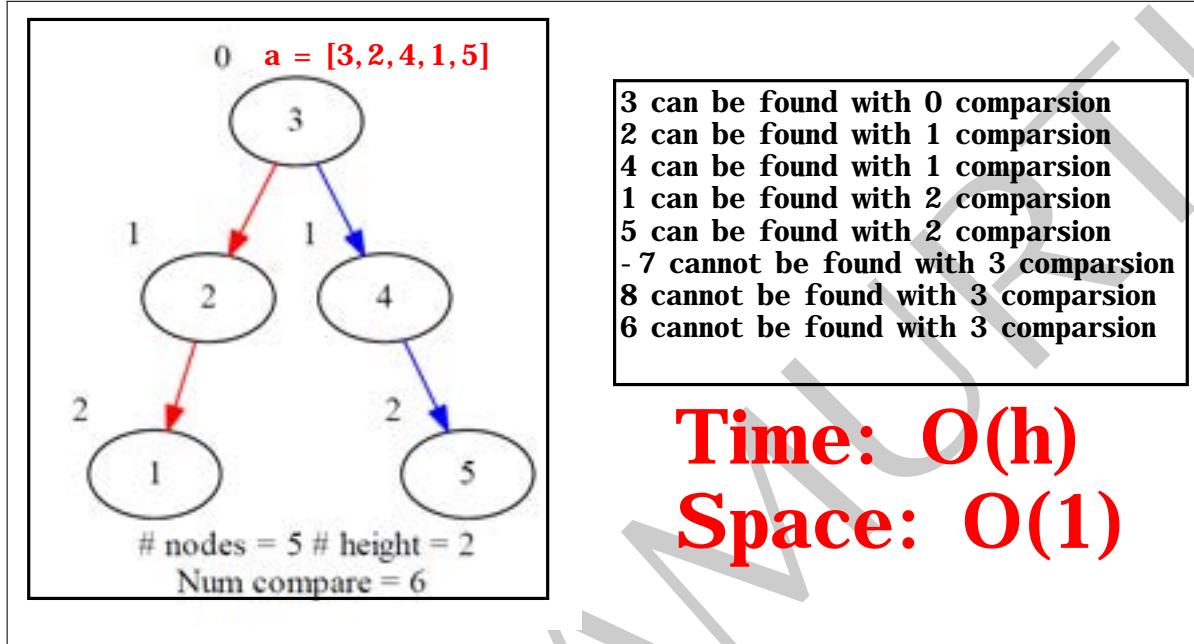


Figure 13.4: Searching an element BST

13.6 Minimum and maximum of a node

13.6. MINIMUM AND MAXIMUM OF A NODE

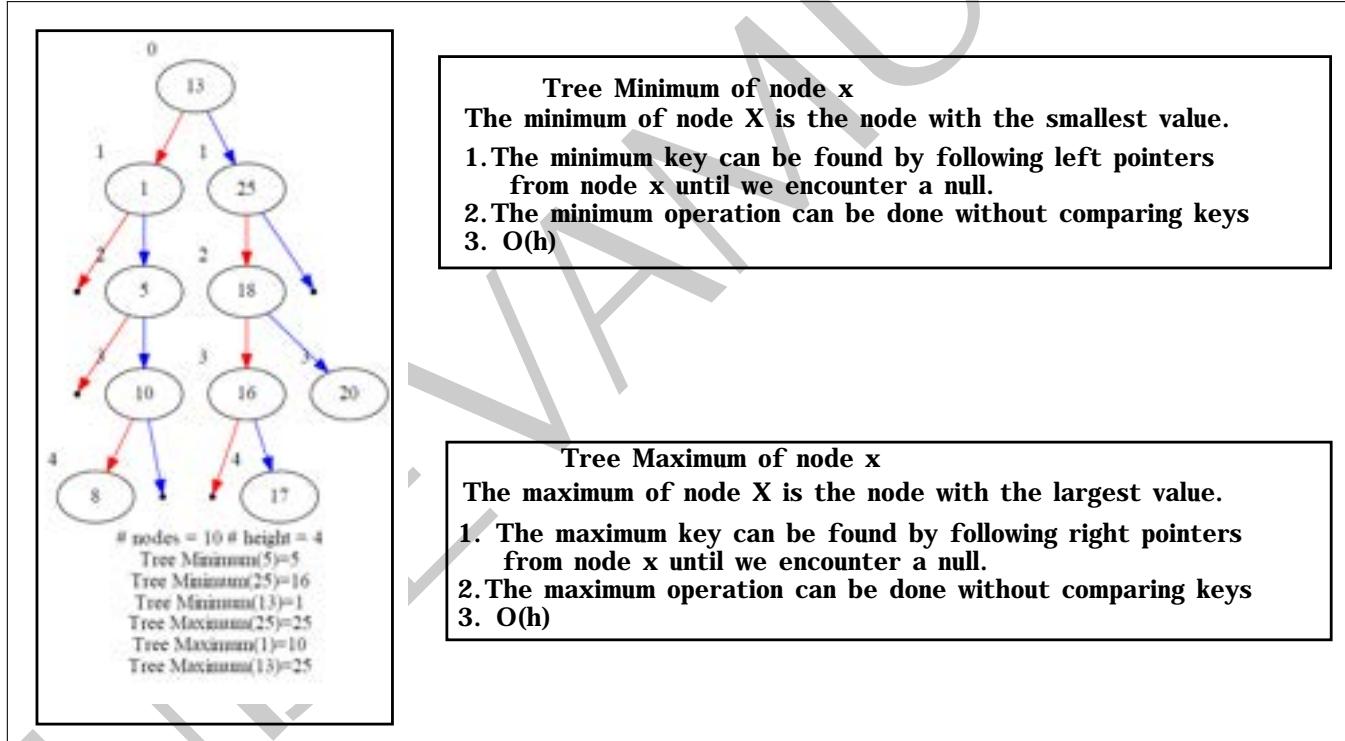
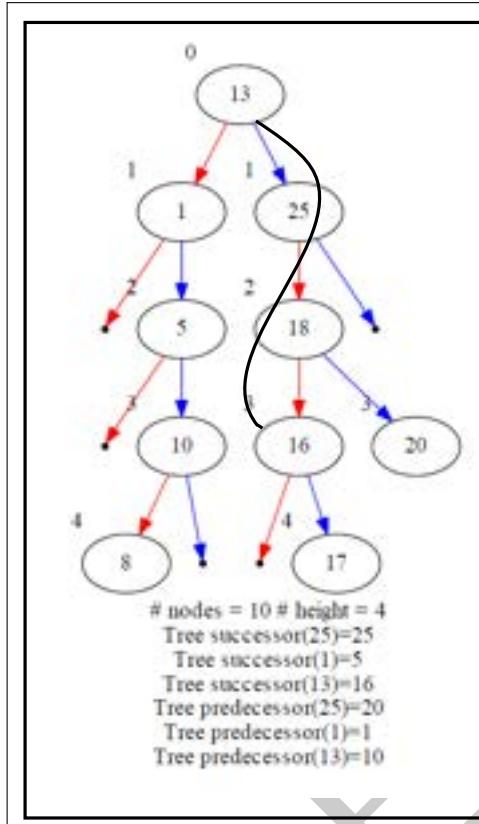


Figure 13.5: Finding minimum and maximum of a node

13.7 Successor of a node



Tree Successor of a node x

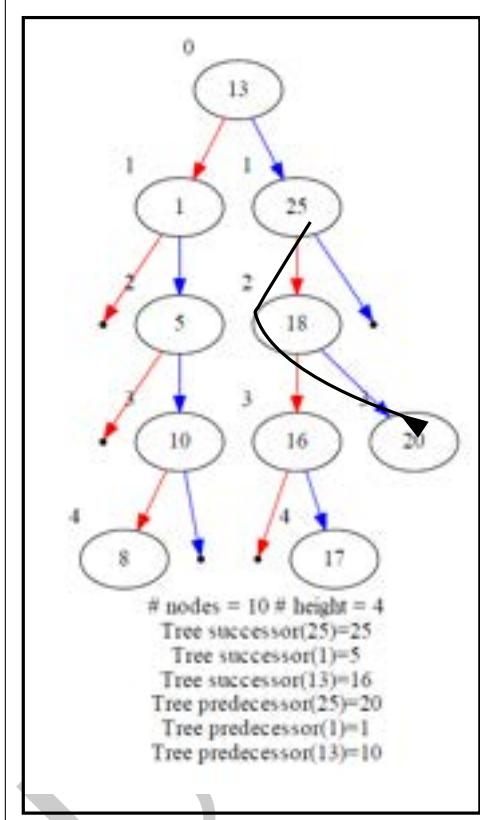
The successor of a node x is the node with the next possible smallest value

1. The successor key is found by right followed by left pointer
2. Successor has NO left child
3. Successor operation is done without comparing key
4. O(h)

Figure 13.6: Finding successor of a node

13.8 Predecessor of a node

13.8. PREDECESSOR OF A NODE



Tree Predecessor of a node x

The predecessor of a node x is the node with the next possible largest value

1. The predecessor key is found by left followed by right pointer
2. Predecessor has NO right child
3. Predecessor operation is done without comparing key
4. O(h)

Figure 13.7: Finding predecessor of a node

13.9 Deleting an element from BST

13.9.1 Case 1: Deleting a leaf from BST

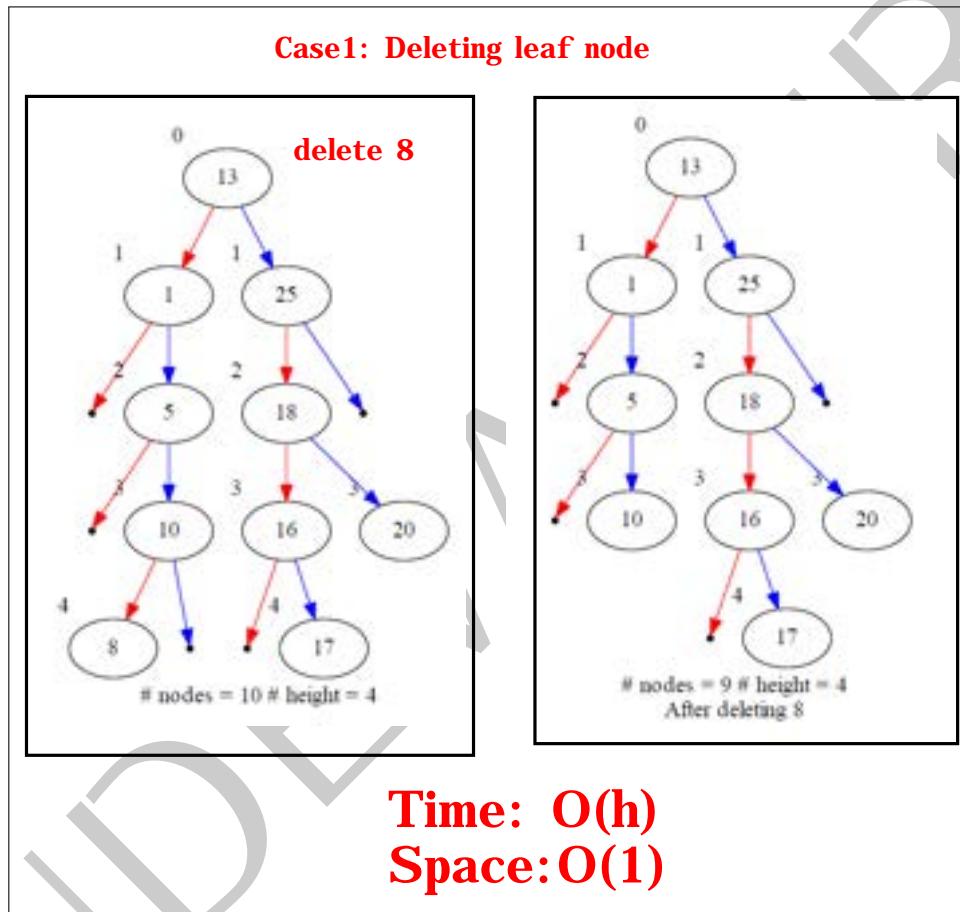


Figure 13.8: Delete a leaf from BST

13.9.2 Case 2: Deleting a node that has one kid from BST

13.9.2.1 Case 2a: Kid is a leaf

13.9. DELETING AN ELEMENT FROM BST

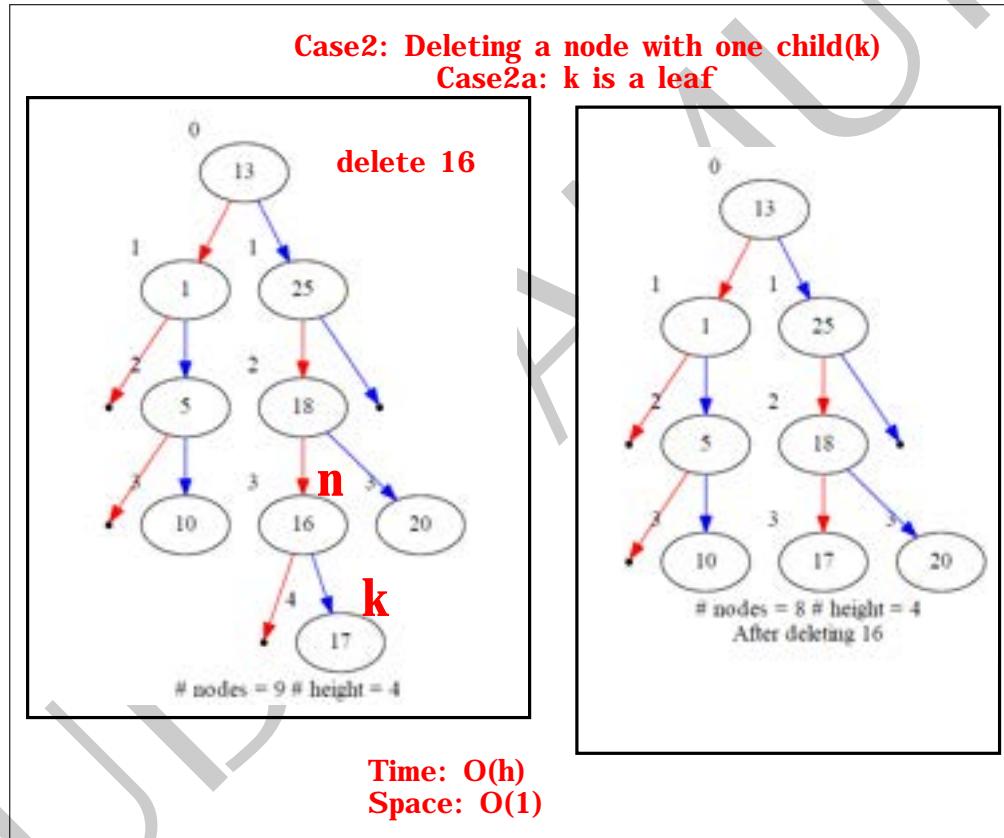


Figure 13.9: Delete a node that has one kid(which is a leaf) from BST

13.9.2.2 Case 2b: Kid is a non leaf

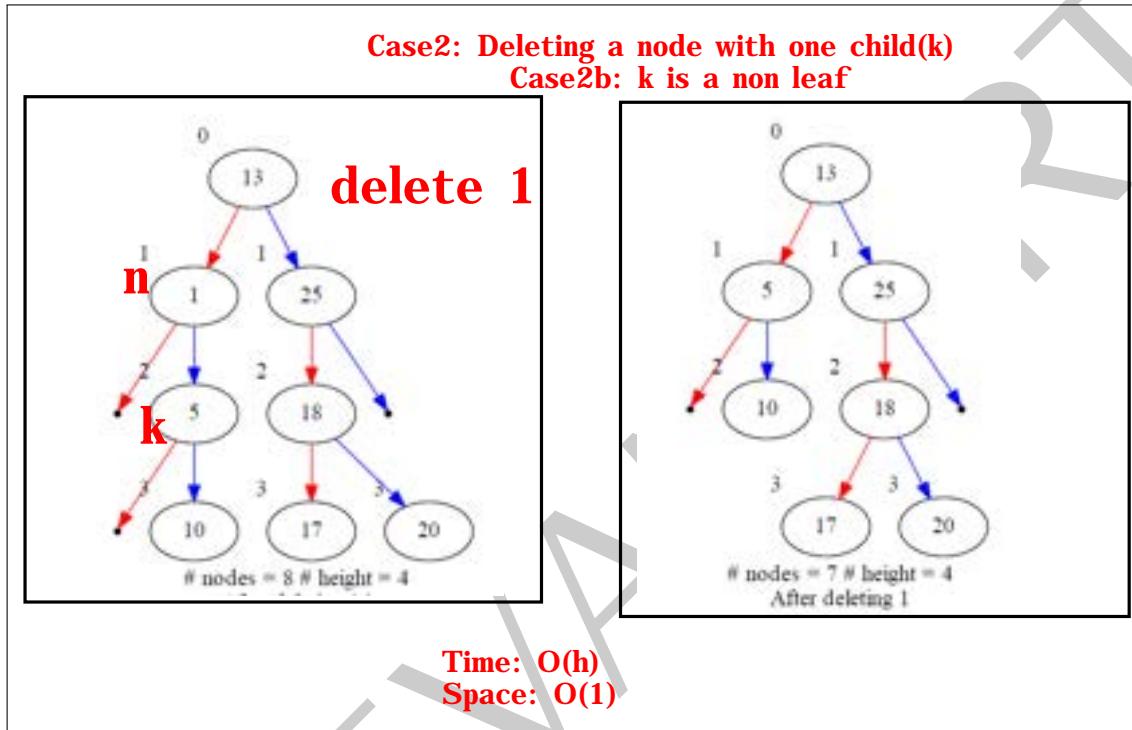


Figure 13.10: Delete a node that has one kid(which is a non leaf) from BST

13.9.3 Case 3: Deleting a node that has two kids from BST

13.9. DELETING AN ELEMENT FROM BST

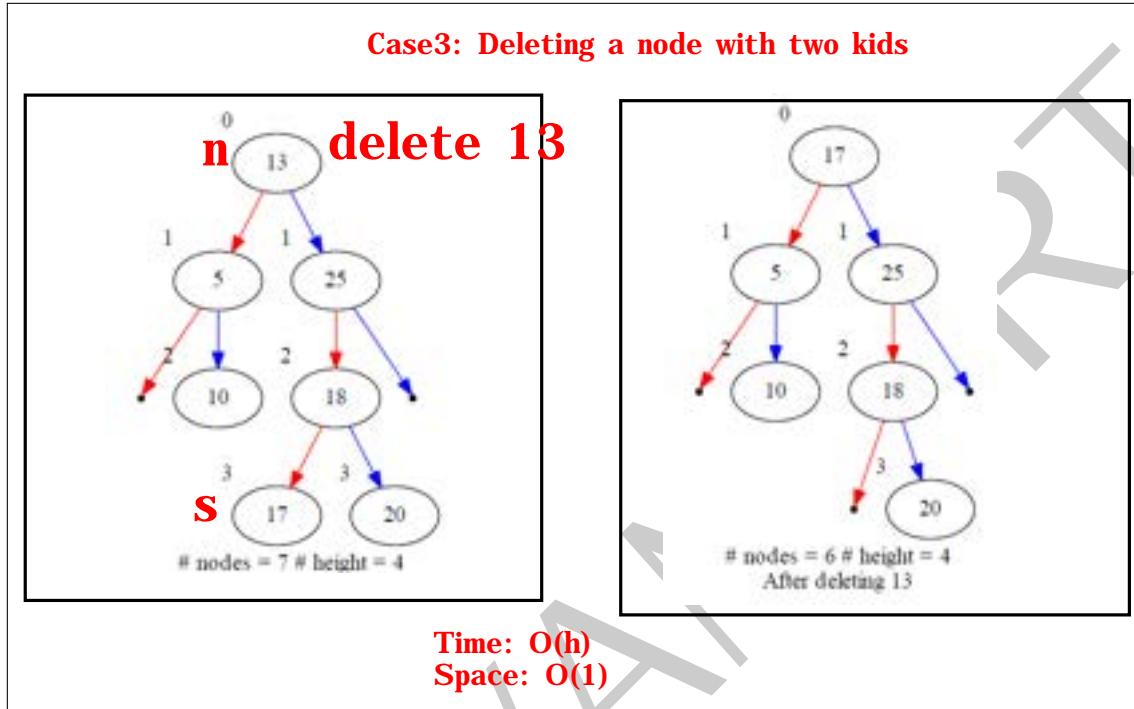


Figure 13.11: Delete a node that has two kids from BST

13.9.4 Deletion in action

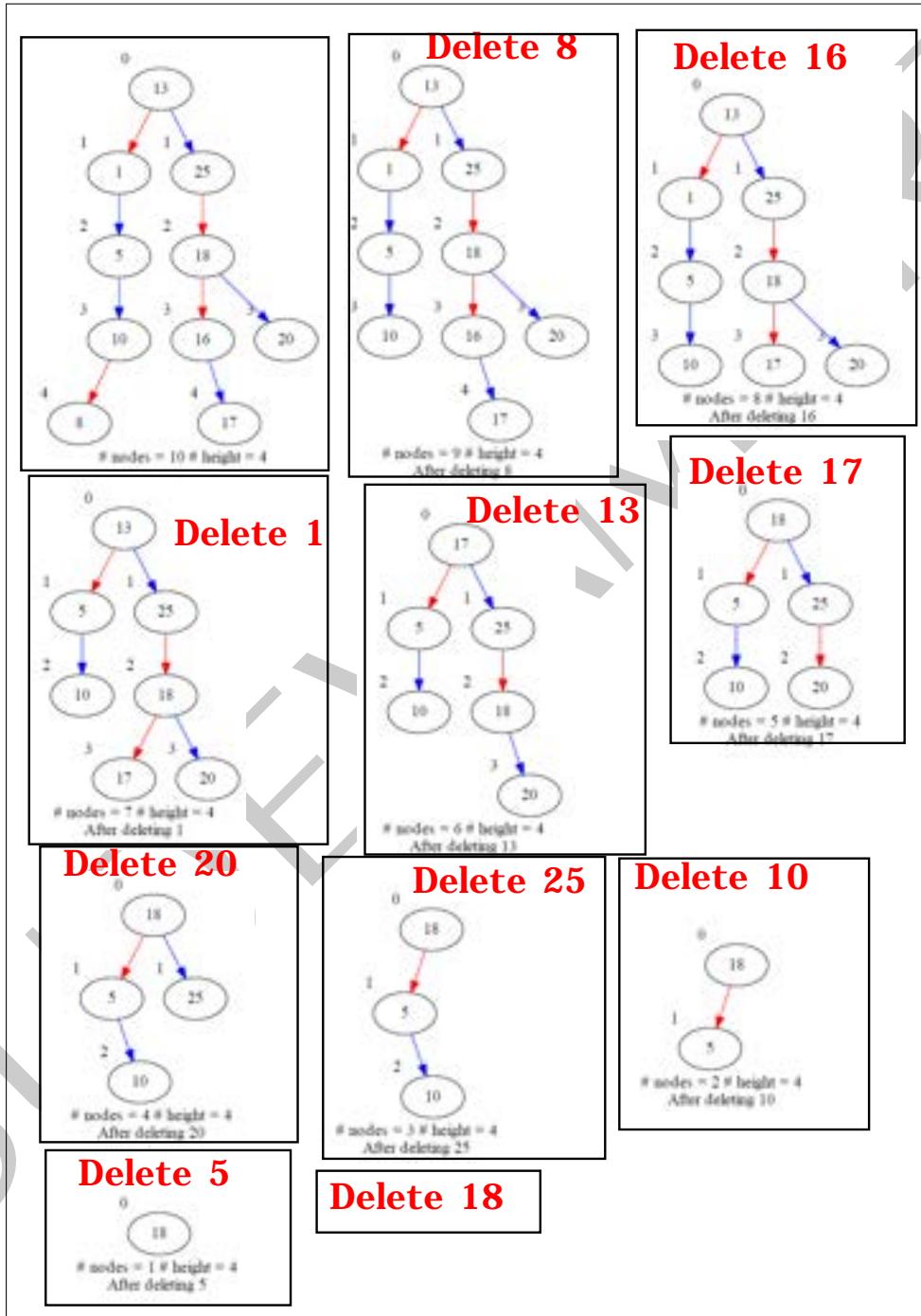


Figure 13.12: Delete a node from BST

VASUDEVAMURTHY

Chapter 14

Trie Data structure

14.1 Introduction

14.2 Need for Trie data structure

14.2. NEED FOR TRIE DATA STRUCTURE

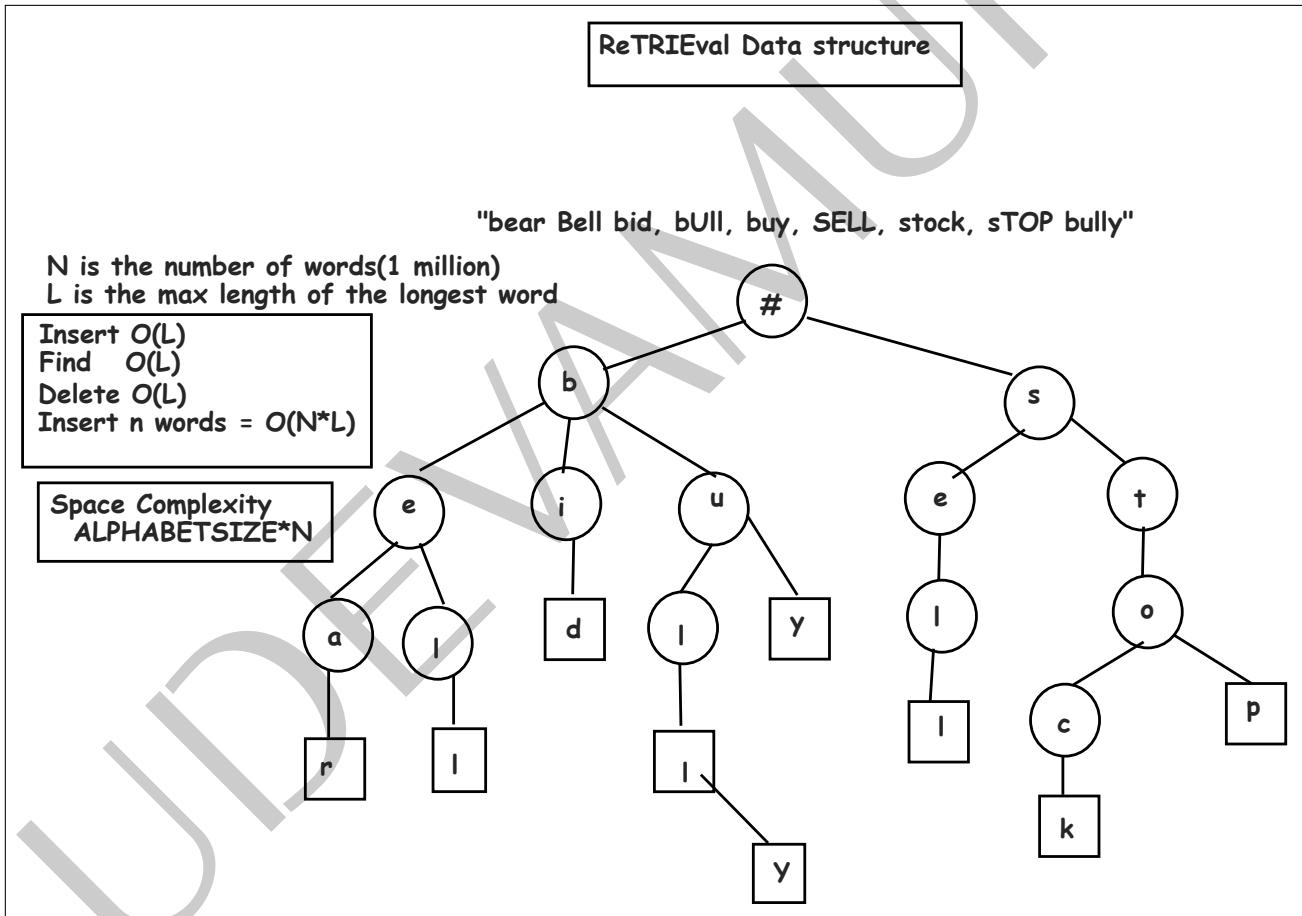


Figure 14.1: Need for Trie data structure

14.3 Printing trie

14.3.1 Printing all paths in a binary tree

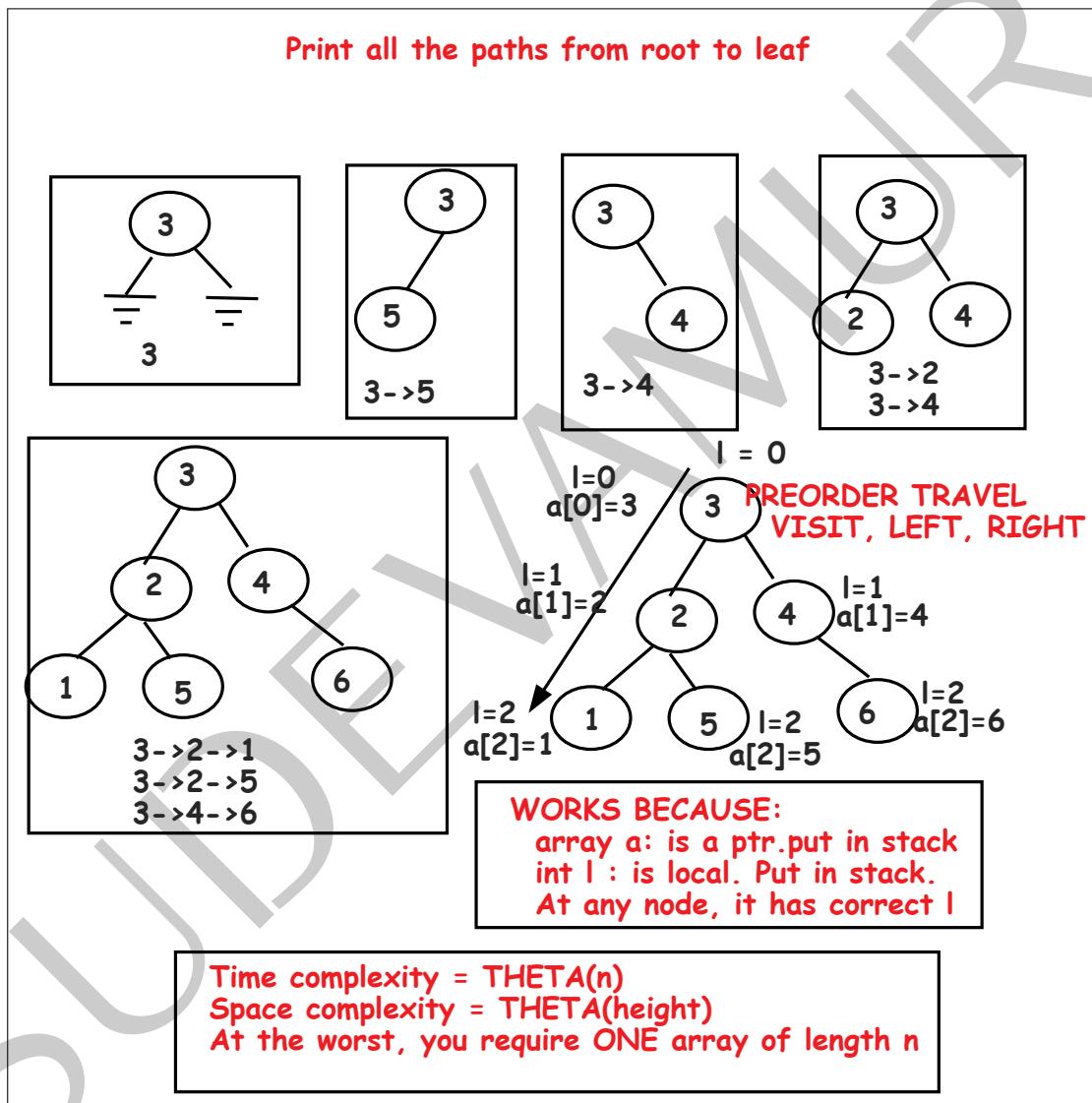


Figure 14.2: Printing all paths of a tree

14.3.2 Printing trie

14.3. PRINTING TRIE

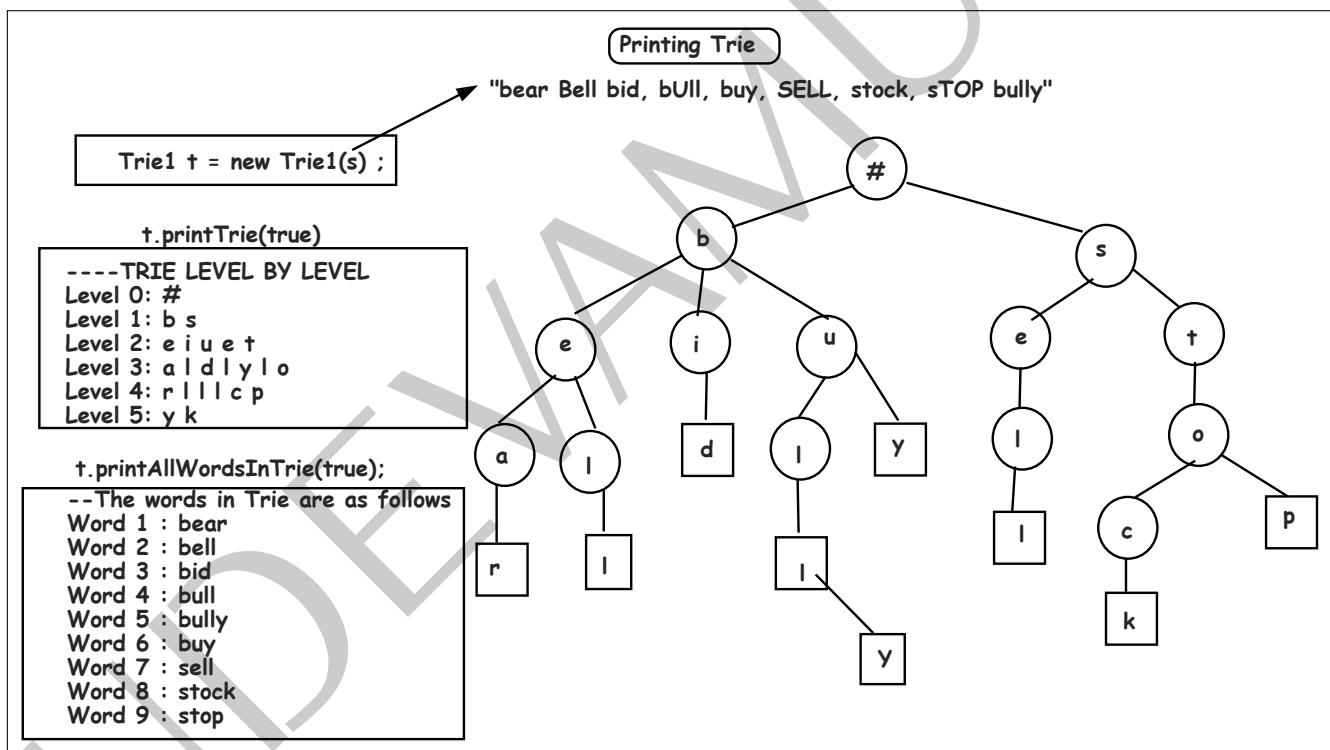


Figure 14.3: Printing trie

14.4 Finding all prefixes of a word

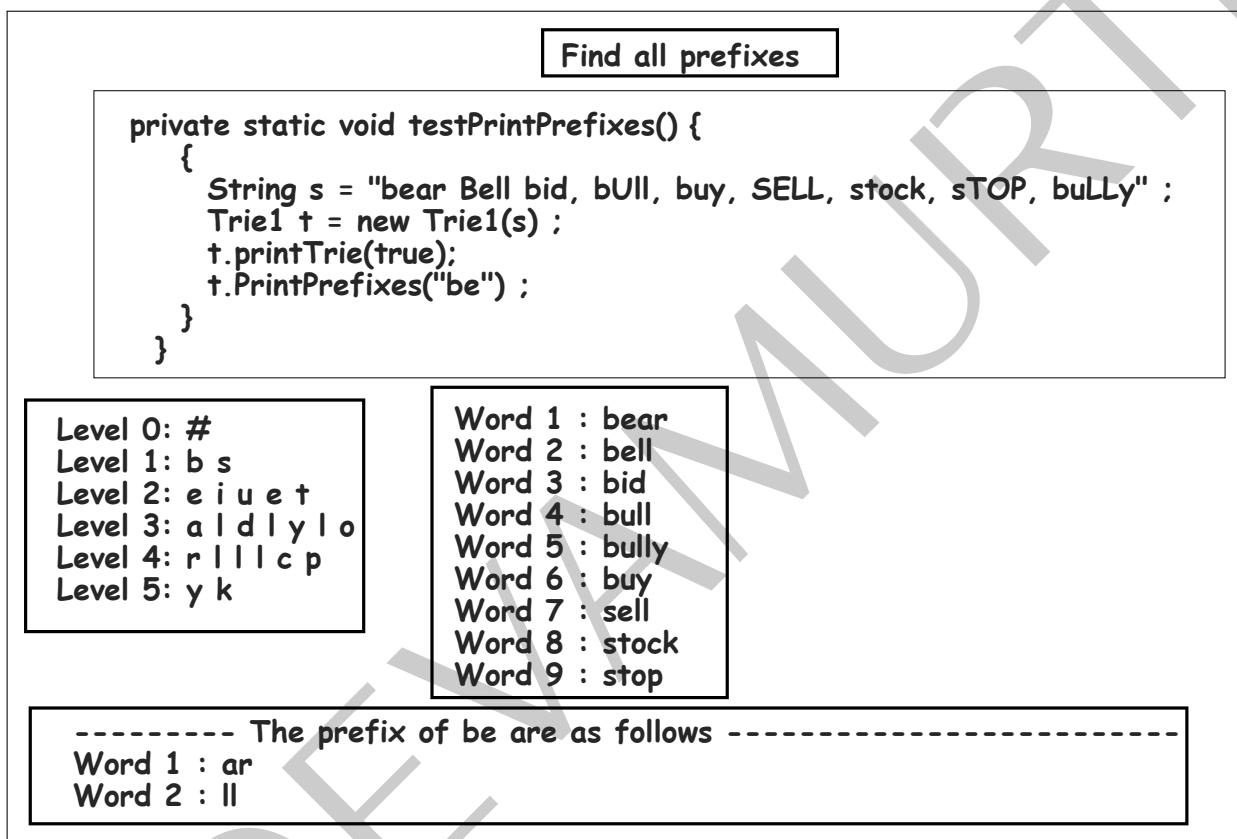


Figure 14.4: Printing all prefixes of a word

14.4. FINDING ALL PREFIXES OF A WORD

```
Find all prefixes

private static void test2() {
    String fileName = "C:/work/java/names/firstname.txt";
    Trie1 t = new Trie1();
    try {
        BufferedReader in = new BufferedReader(new FileReader(fileName));
        String str = null;
        int k = 0;
        while ((str = in.readLine()) != null) {
            int i = 0;
            String name = new String();
            while (str.charAt(i) != ' ') {
                name = name + str.charAt(i++);
            }
            t.insert(name);
            //System.out.println(k++ + " " + name);
        }
        in.close();
    } catch (IOException e) {}
    int k = t.printAllWordsInTrie(true);
    u.myassert(k == 5163);
    t.PrintPrefixes("be");
}
```

----- The prefix of be are as follows -----

Word 1 : a	Word 34 : rnard
Word 2 : ata	Word 35 : rnarda
Word 3 : atrice	Word 36 : rnardina
Word 4 : atris	Word 37 : rnardine
Word 5 : atriz	Word 38 : rnardo
Word 6 : ati	Word 39 : rniece
Word 7 : ulah	Word 40 : rnetta
Word 8 : be	Word 41 : rnice
Word 9 : cki	Word 42 : rnie
Word 10 : ckie	Word 43 : rniece
Word 11 : cky	Word 44 : rnita
Word 12 : e	Word 45 : rry
Word 13 : len	Word 46 : rt
Word 14 : lie	Word 47 : rta
Word 15 : linda	Word 48 : rtha
Word 16 : lkis	Word 49 : rtie
Word 17 : ll	Word 50 : rram
Word 18 : lla	Word 51 : ryl
Word 19 : lle	Word 52 : ss
Word 20 : lve	Word 53 : sie
Word 21 : n	Word 54 : th
Word 22 : nedict	Word 55 : thanie
Word 23 : nita	Word 56 : thann
Word 24 : nito	Word 57 : thany
Word 25 : njamin	Word 58 : theil
Word 26 : nnett	Word 59 : tsey
Word 27 : nnie	Word 60 : tsey
Word 28 : nny	Word 61 : tte
Word 29 : nton	Word 62 : tthe
Word 30 : renice	Word 63 : ttnia
Word 31 : rna	Word 64 : tty
Word 32 : rnadette	Word 65 : ttyann
Word 33 : rnadine	Word 66 : ttre
	Word 67 : ula
	Word 68 : ulah
	Word 69 : v
	Word 70 : verlee
	Word 71 : verley
	Word 72 : verly

Figure 14.5: Printing all prefixes of a word

14.5 Printing numbers in lexicographic order

Given an integer N, print numbers from 1 to N in lexicographic order -- Bloomberg LP

```
private static void printOneToNinLexicographicOrder() {
{
    int n = 25 ;
    Trie1 t = new Trie1(n) ;
}
}
```

----- TRIE LEVEL BY LEVEL -----

Level 0: #

Level 1: 1 2 3 4 5 6 7 8 9

Level 2: 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

----- The words in Trie are as follows -----

Word 1 : 1			
Word 2 : 10	Word 12 : 2	Word 19 : 3	
Word 3 : 11	Word 13 : 20	Word 20 : 4	
Word 4 : 12	Word 14 : 21	Word 21 : 5	
Word 5 : 13	Word 15 : 22	Word 22 : 6	
Word 6 : 14	Word 16 : 23	Word 23 : 7	
Word 7 : 15	Word 17 : 24	Word 24 : 8	
Word 8 : 16	Word 18 : 25	Word 25 : 9	
Word 9 : 17			
Word 10 : 18			
Word 11 : 19			

Figure 14.6: Printing numbers in lexicographic order

VASUDEVAMURTHY

Chapter 15

Disjoint sets

15.1 Introduction

15.2 Need for Union and Find Data structure

15.2. NEED FOR UNION AND FIND DATA STRUCTURE

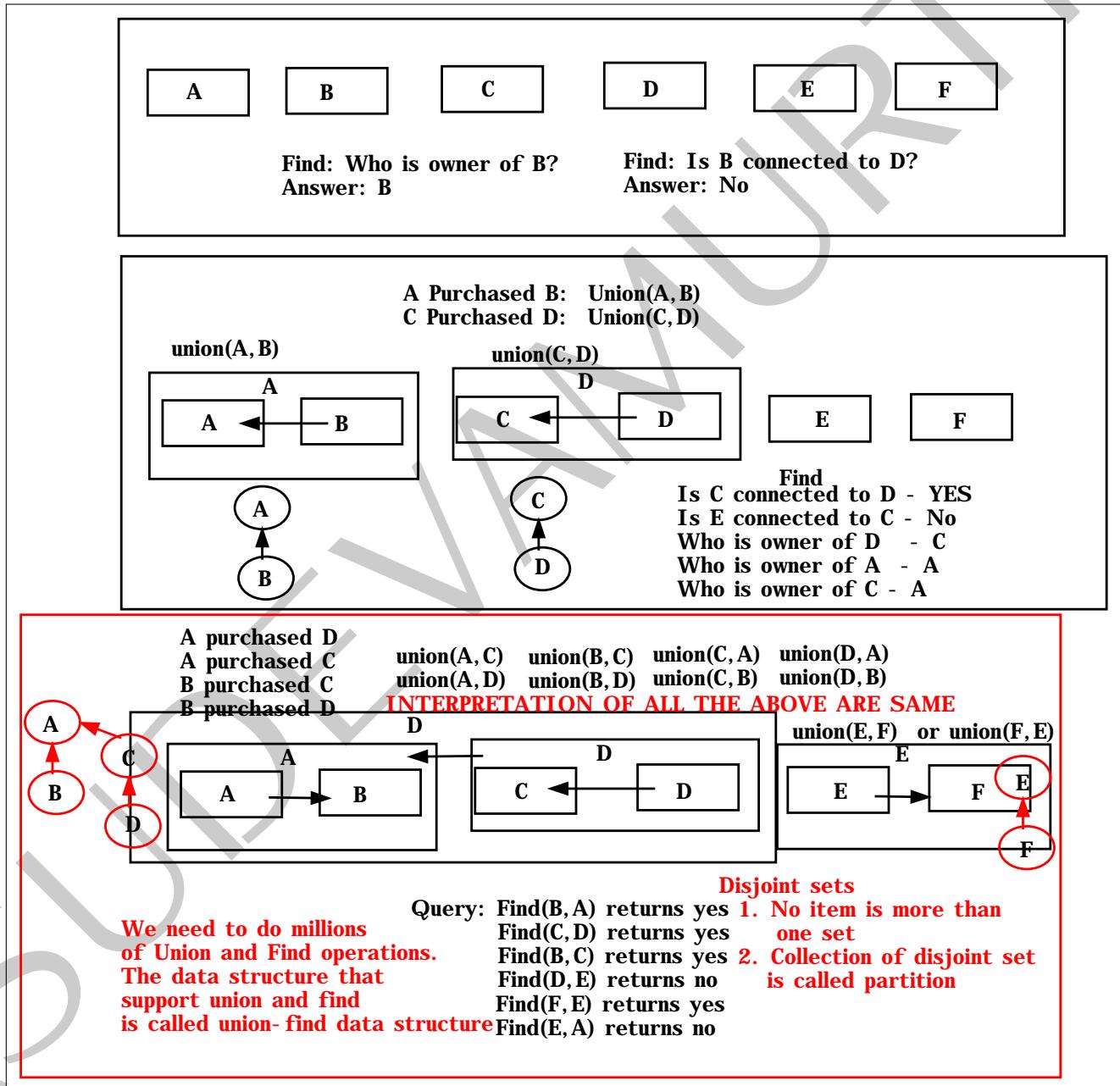
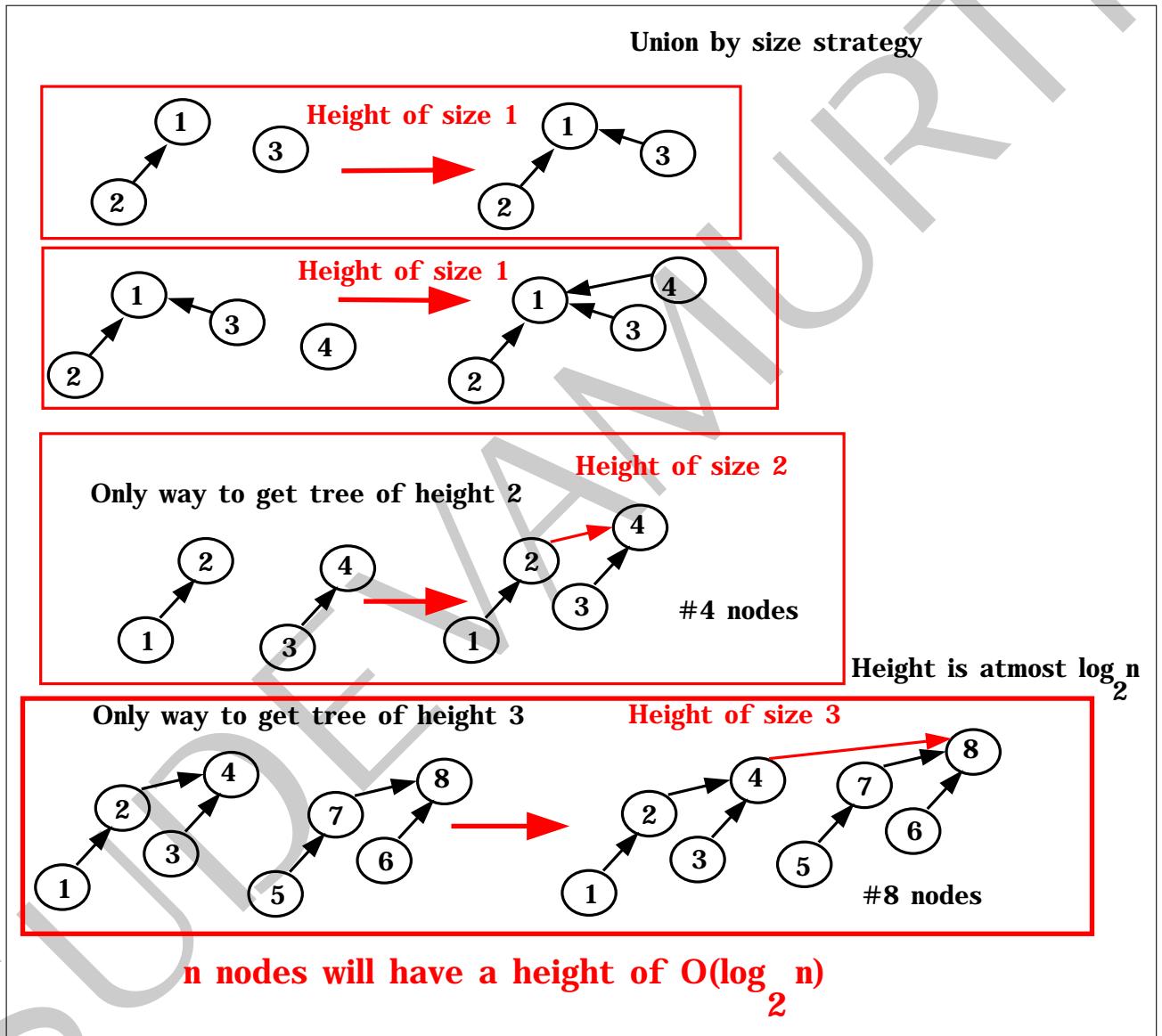


Figure 15.1: Need for Union and Find Data structure

15.3 Smart Union Algorithm

Figure 15.2: To show the height of the tree is at most $O(\log_2 n)$

15.3. SMART UNION ALGORITHM

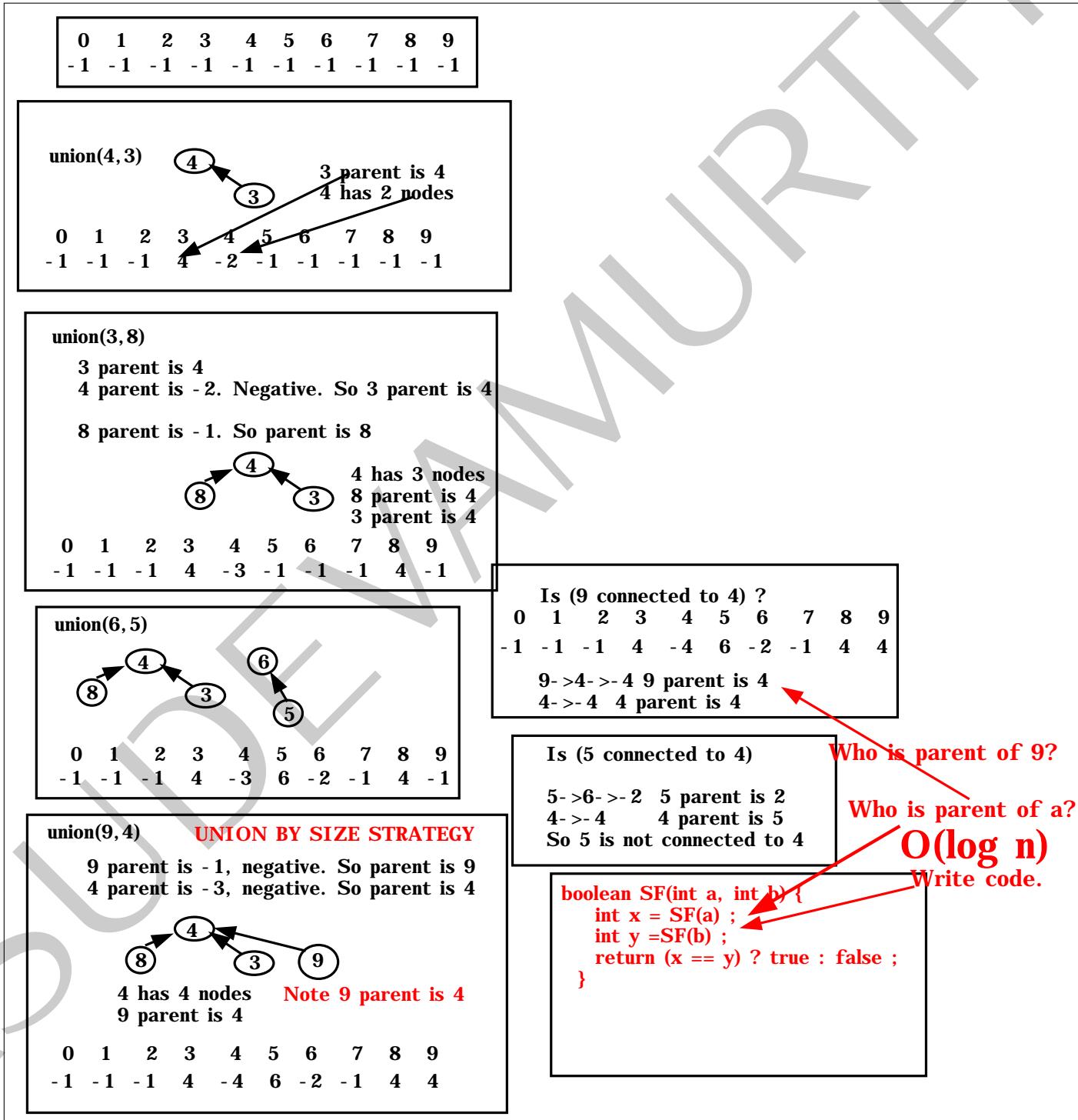


Figure 15.3: Smart union by size strategy algorithm

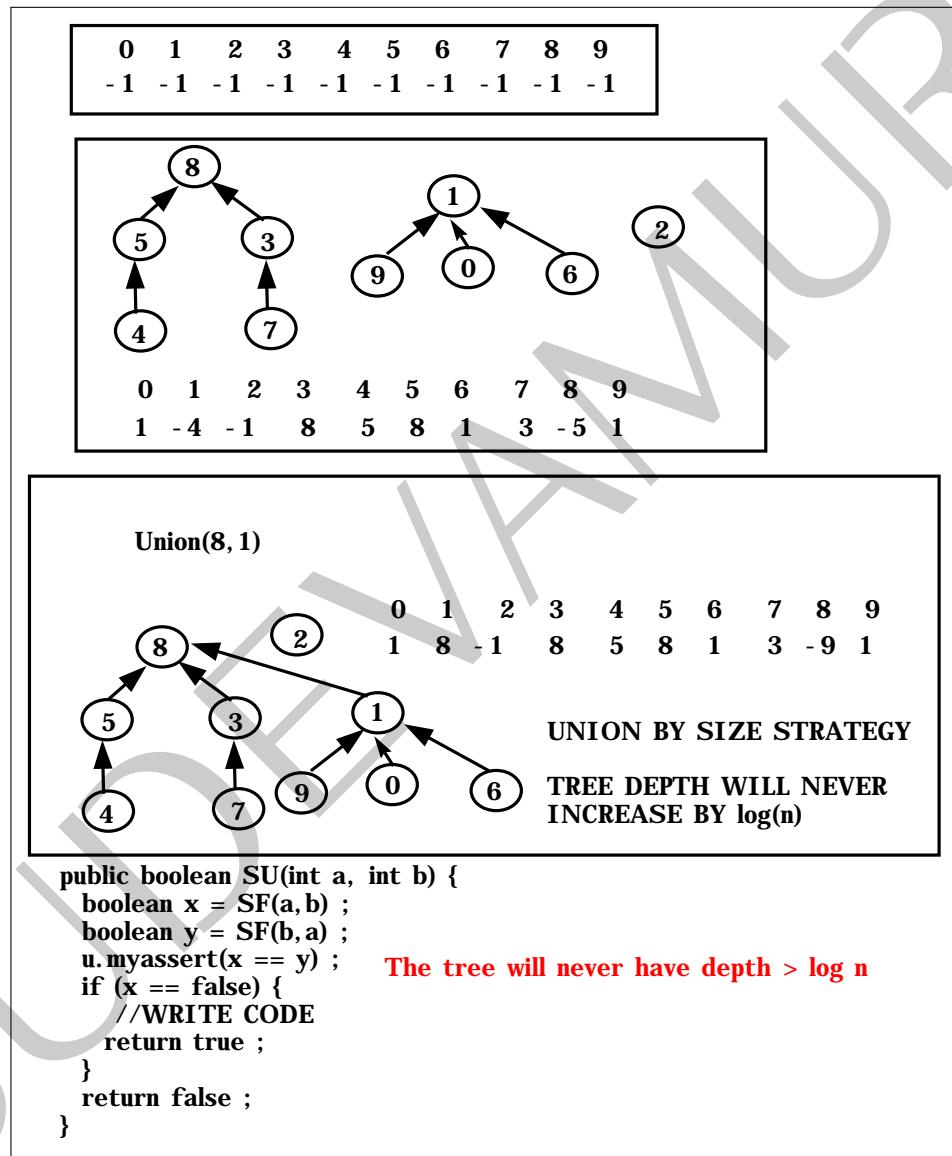


Figure 15.4: Smart union by size strategy algorithm

15.3. SMART UNION ALGORITHM

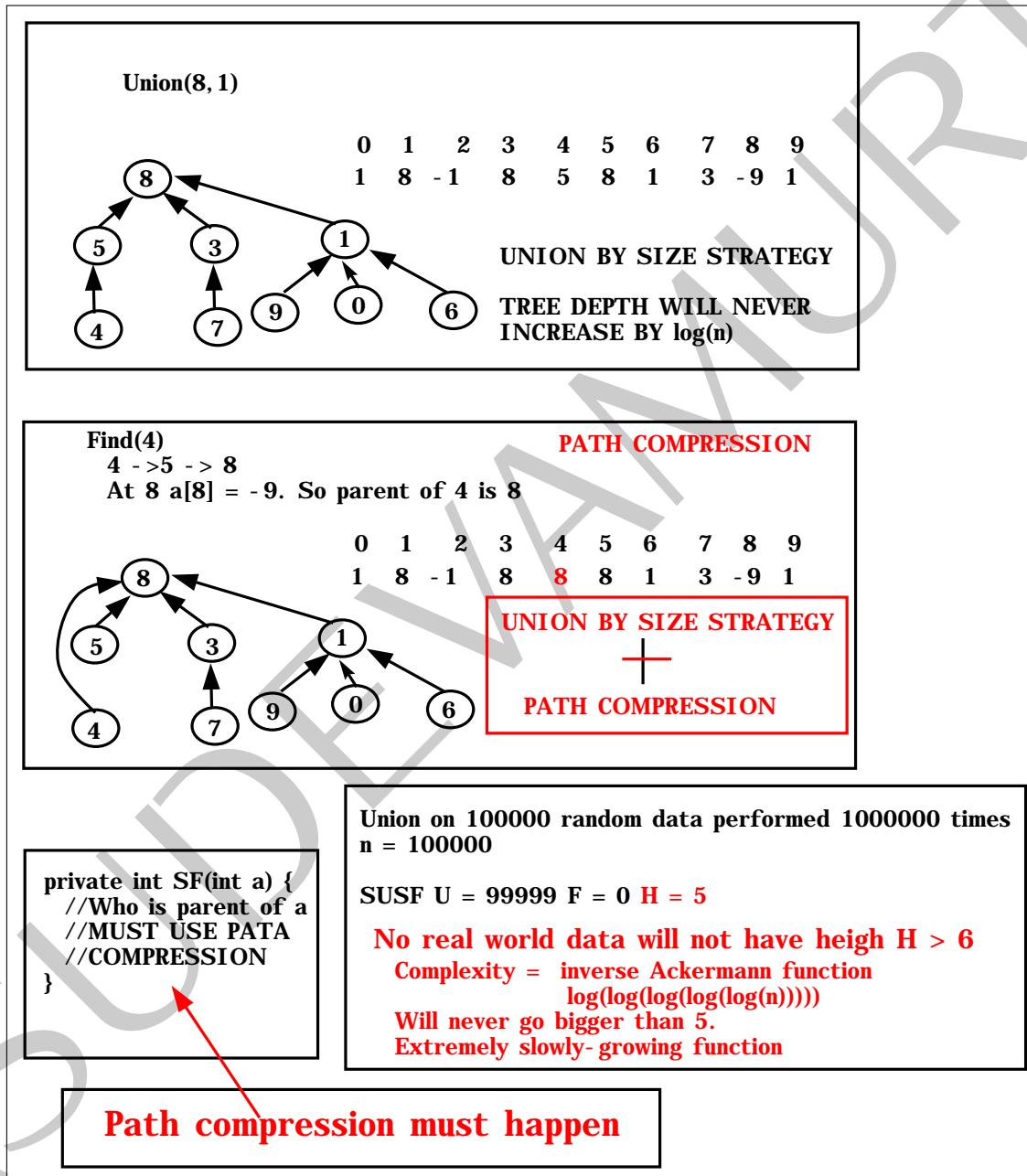


Figure 15.5: Smart union by size strategy with path compression algorithm

15.4 Problem set

Problem 15.4.1. Implement Merging Communities (Hacker Rank) as shown in figures 20.29.

15.4. PROBLEM SET

Problem 15.4.2. Implement Kruskal algorithm as shown in figures 15.6 and 15.7. You must use SUSF algorithm for finding the loop.

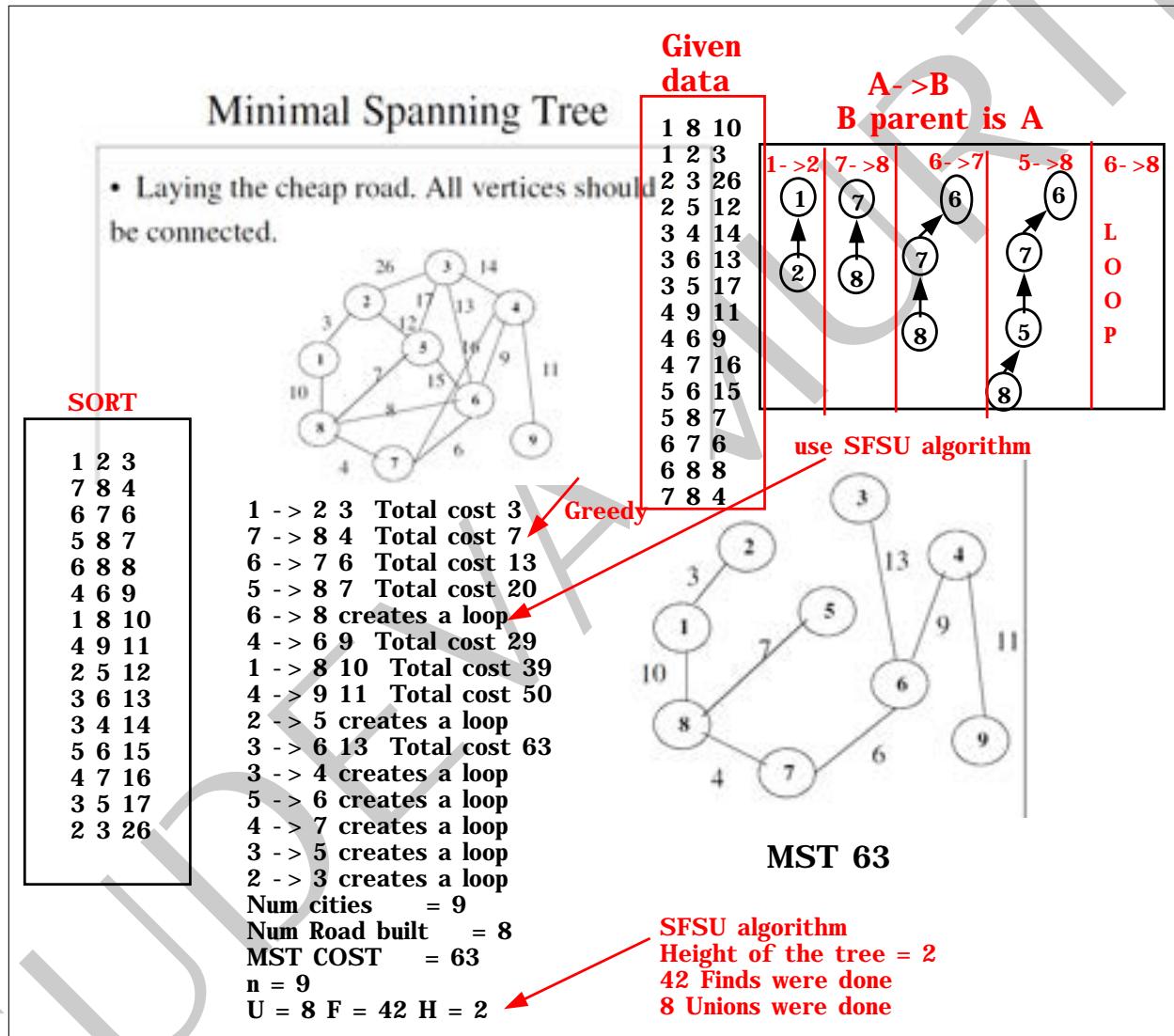


Figure 15.6: Kruskal Algorithm in action

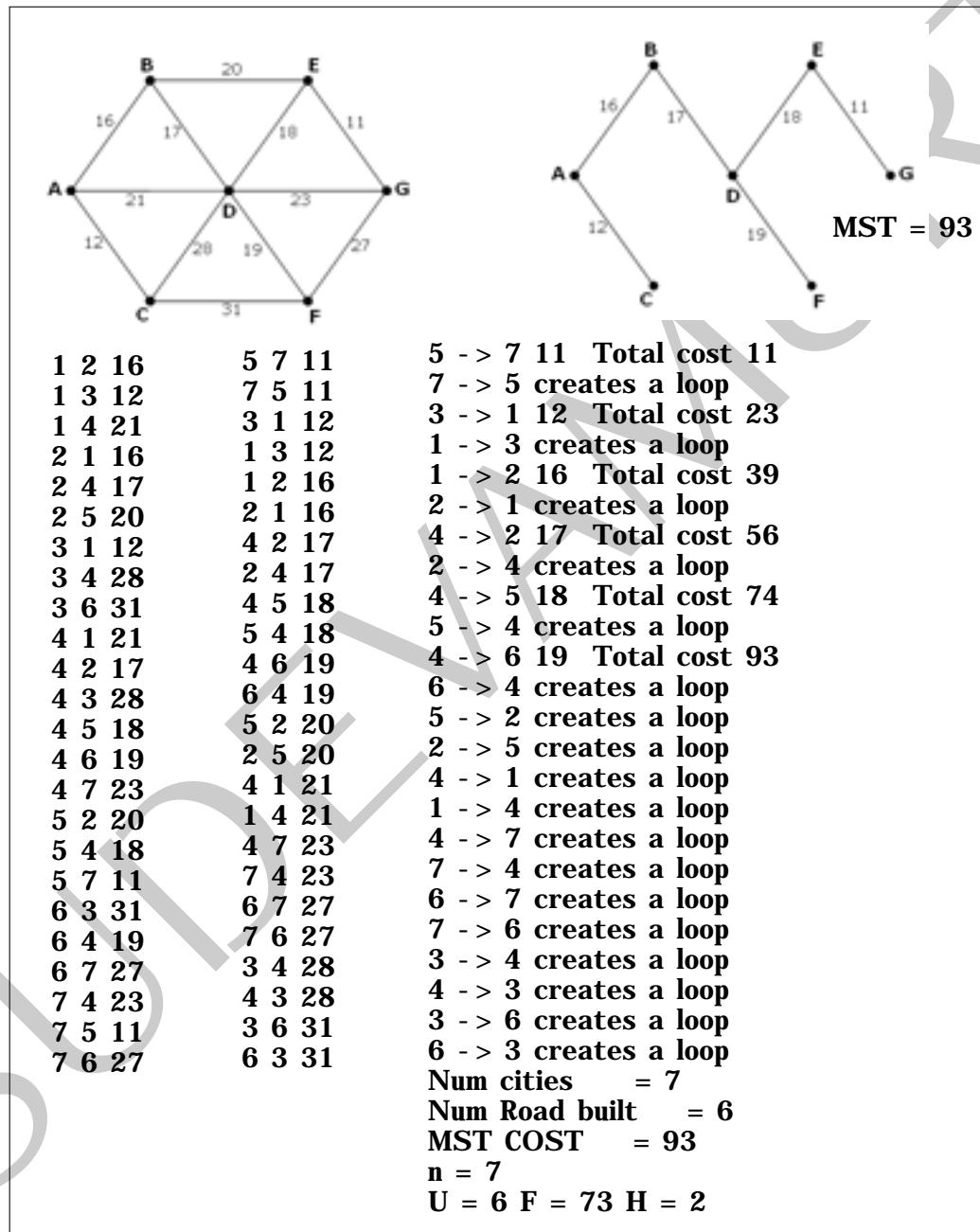


Figure 15.7: Kruskal Algorithm in action

15.4. PROBLEM SET

Problem 15.4.3. Answer the questions below by hand and post the scanned document on Canvas.

```

N = 17
s = SUSF(N)
for i = 1; i <=15; i = i +2
    s.U(i,i+1)
for i = 1; i <= 13; i = i + 4
    s.U(i, i + 2)
s.U(1,5)
s.U(11,13)
s.U(1,10)
s.num(2)
s.num(9)

```

1

MUST SHOW BOTH ARRAY AND TREE of UF

How many elements are there in 2 and 9?
 What will be the height of the tree?

```

N = 10
s = SUSF(N)
e1 = [ [4,3], [3,8], [6,5], [9,4], [2,1] ]
e2 = [ [5,0], [7,2], [6,1], [7,3] ]

```

2

for e in e1: MUST SHOW BOTH ARRAY AND TREE of UF
 s.U(e[0],e[1])
 Is 8 and 9 connected -> TRUE/FALSE
 Is 5 and 4 connected -> TRUE/False

for e in e2:
 s.U(e[0],e[1]) <- - WHAT WILL BE THE HEIGHT
 OF TREE?

VASUDEVAMURTHY

Chapter 16

Greedy Algorithms

16.1 Introduction

VASUDEVAMURTHY

Chapter 17

Dynamic Programming

17.1 Introduction

17.2 Why greedy algorithm may fail to give solution?

17.2. WHY GREEDY ALGORITHM MAY FAIL TO GIVE SOLUTION?

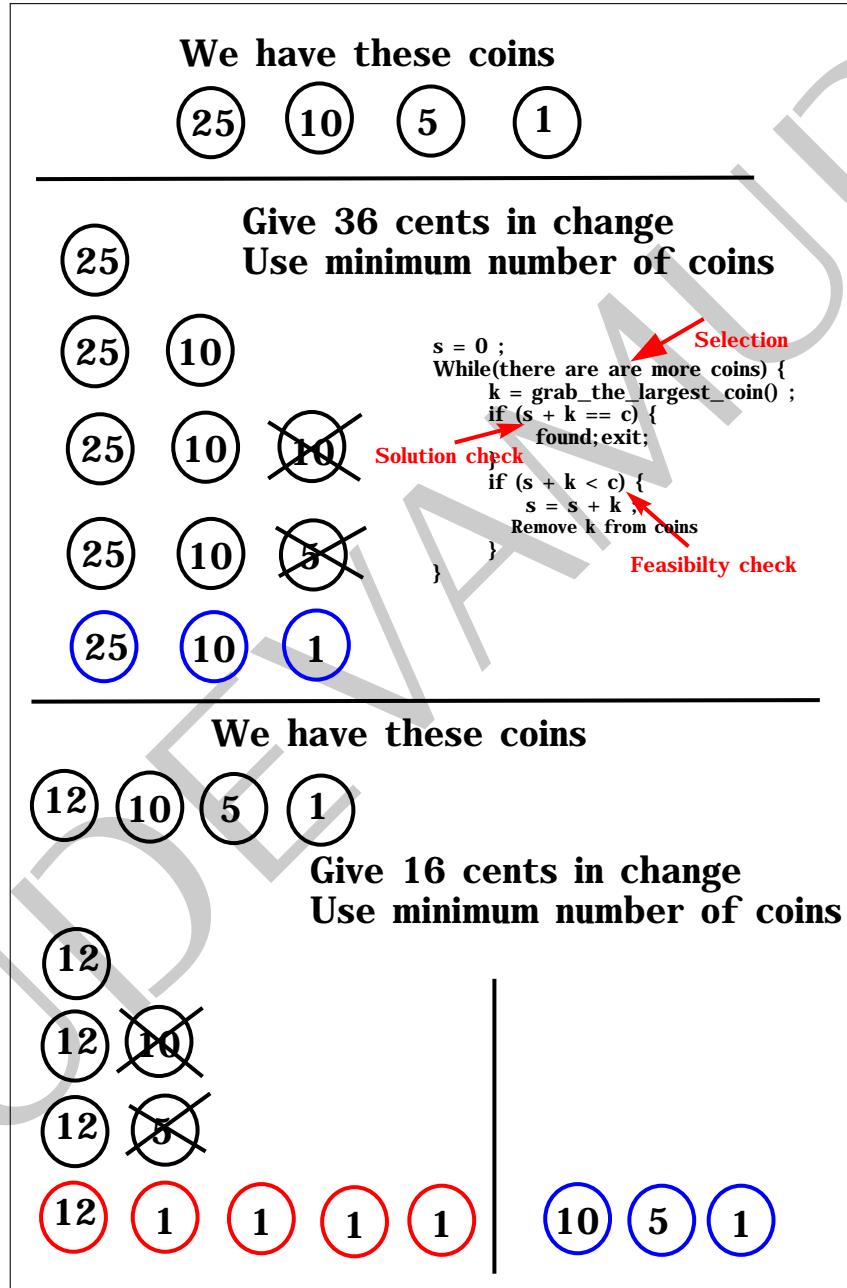


Figure 17.1: Correct and incorrect solutions using greedy algorithm

17.3 Fibonacci numbers

17.3.1 Fibonacci numbers using divide and conquer

17.3.2 Fibonacci numbers using dynamic programming

17.4 Coin change problem

Amazon Interview Programming Problem

The StampDispenser class represents a postage stamp vending machine. The machine contains stamps of different values. The machine will always have a stamp with a value of 1 cent and the machine will never run out of any type of stamp. The machine should allow a consumer of the class to calculate the minimum number of stamps that the machine can dispense to fill a given request.

Your task is to complete one of the provided implementations of the StampDispenser class: C++, C#, or Java.

```
/**  
 * Facilitates dispensing stamps for a postage stamp machine.  
 */  
public class StampDispenser  
{  
    /**  
     * Constructs a new StampDispenser that will be able to dispense the given  
     * types of stamps.  
     *  
     * @param stampDenominations The values of the types of stamps that the  
     * machine should have. Should be sorted in descending order and  
     * contain at least a 1.  
     */  
    public StampDispenser(int[] stampDenominations)  
    {  
    }  
  
    /**  
     * Returns the minimum number of stamps that the machine can dispense to  
     * fill the given request.  
     *  
     * @param request The total value of the stamps to be dispensed.  
     */  
    public int calcMinNumStampsToFillRequest(int request)  
    {  
        return 0;  
    }  
}
```

As an example, suppose an instance of StampDispenser was created with stampDenominations, {90, 30, 24, 10, 6, 2, 1}, and calcMinNumStampsToFillRequest(int) was called with request, 34. The call should return 2, as 34 cents can best be filled by one 24 cent stamp and one 10 cent stamp.

Things to keep in mind:

1. Assume that a junior programmer is going to read your code. You should include comments and any other aides that you use to communicate your code to other developers.
2. Optimize the code for speed.
3. The code should compile and work.
4. The code should work for countries with high denomination values where stamp values of 1000 or 9000 are common.

Figure 17.2: Amazon Interview Question

17.4. COIN CHANGE PROBLEM

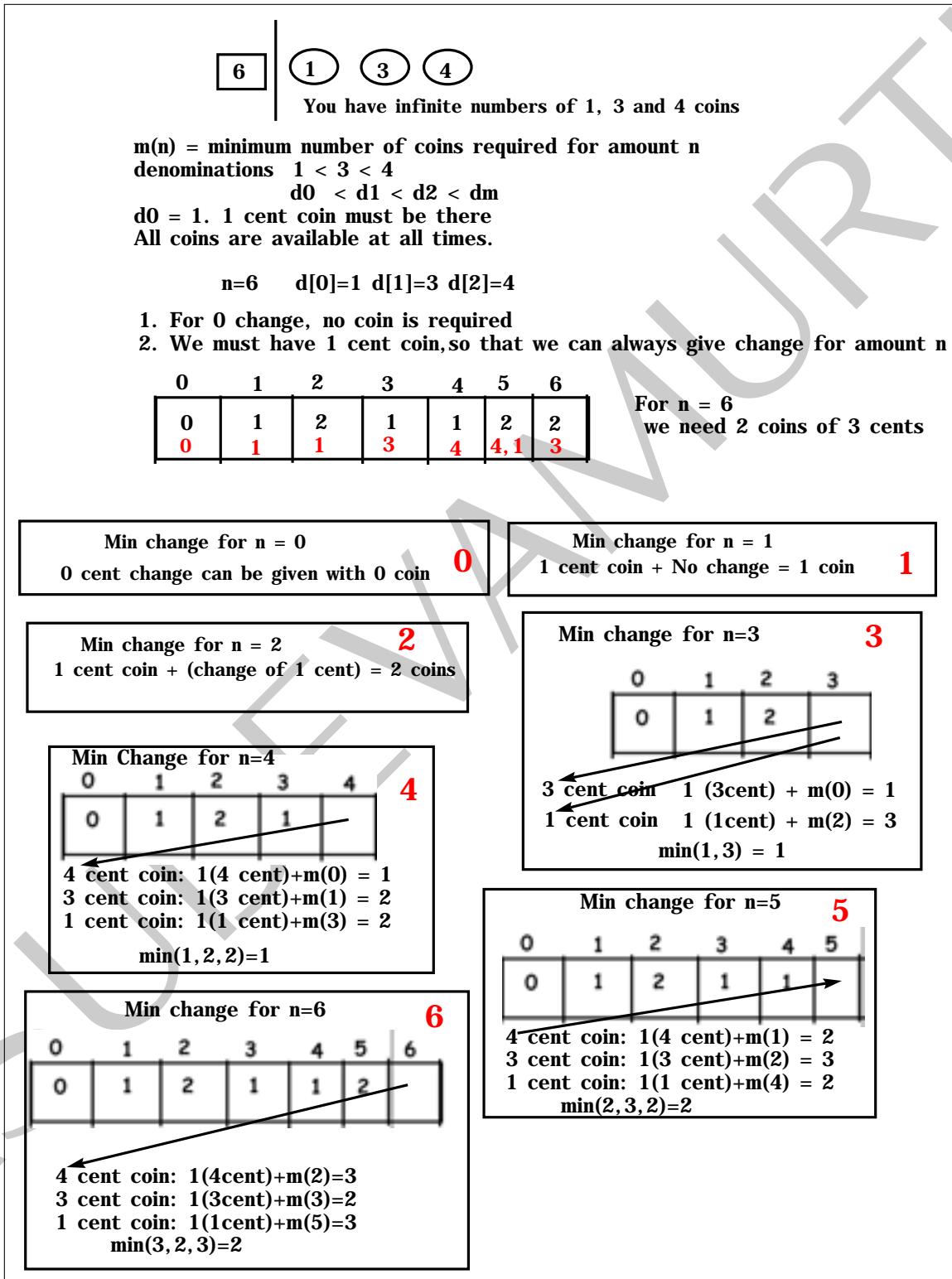


Figure 17.3: Building optimal table using dynamic programming

Give minimum change for 34 cents using coins {1,2,6,10,24,30,90}														
i =	0	1	2	3	4	5	6	7	8	9	10	11	12	13
m array	0	1	1	2	2	3	1	2	2	3	1	2	2	3
k array	0	1	2	1	2	1	6	1	2	1	10	1	2	1
.
14	15	16	17	18	19	20								
3	4	2	3	3	4	2								
2	1	6	1	2	1	10								
20	21	22	23	24	25	26	27	28	29	30	31	32		
2	3	3	4	1	2	2	3	3	4	1	2	2		
10	1	2	1	24	1	2	1	2	1	30	1	2		
minimum change for 23 cents can be achieved using 4 coin														
1:::Pick coin 1. Current val= 1. Remaining val= 22														
2:::Pick coin 2. Current val= 3. Remaining val= 20														
3:::Pick coin 10. Current val= 13. Remaining val= 10														
4:::Pick coin 10. Current val= 23. Remaining val= 0														

Figure 17.4: Solution to Amazon Interview Question

17.5 Maximum subset independent set of a path graph

Maximum Weight Independent Set of a Path Graph

Independent set is a set of vertices such that any two vertices in the set do not have a direct edge between them.

It is given that there is at least one way to traverse from any vertex in the graph to another, i.e. the graph has one connected component.

$\{A\} = 1$
 $\{B\} = 2$
 $\{C\} = 3$
 $\{D\} = 4$
 $\{A, C\} = 4$
 $\{A, D\} = 5$
 $\{B, D\} = 6$

$\{1, 2, 4\}$

$\{1\} = 1$
$\{2\} = 2$
$\{4\} = 4$
$\{1, 4\} = 5$

$\{6, 1, 2, 7, 3, 5\}$

$\{6\} = 6$	$\{6, 2\} = 8$	$\{1, 7\} = 8$
$\{1\} = 1$	$\{6, 7\} = 13$	$\{1, 3\} = 4$
$\{2\} = 2$	$\{6, 3\} = 9$	$\{1, 5\} = 6$
$\{7\} = 7$	$\{6, 5\} = 11$	$\{2, 3\} = 5$
$\{3\} = 3$	$\{6, 2, 5\} = 13$	
$\{5\} = 5$	$\{6, 7, 5\} = 18$	

A brute force algorithm would work but would require exponential running time

A greedy algorithm could have us making an incorrect "myopic" decision that would produce an invalid overall answer

A divide and conquer algorithm could work, but would require non-trivial and expensive work to resolve conflicts (adjacency) upon merging subproblems together

A dynamic programming approach can solve the problem in linear time

Figure 17.5: problem definition

17.5.1 Expected output

MaxPath problem STARTS

----- test1 -----

1 2 4

----- Brute Force -----

```

1:{1} = 1
2:{2} = 2
3:{4} = 4

```

```
4:{1,4} = 5
{1,4} = 5 work = 8
```

1 2 4

----- Dynamic programming -----

0	1	2	3
0	1	2	4
0	1	2	5
0	1	2	4

```
{4,1} = 5 work = 3
```

----- test2 -----

1 2 3 4

----- Brute Force -----

```
1:{1} = 1
2:{2} = 2
3:{3} = 3
4:{1,3} = 4
5:{4} = 4
6:{1,4} = 5
7:{2,4} = 6
{2,4} = 6 work = 19
```

1 2 3 4

----- Dynamic programming -----

0	1	2	3	4
0	1	2	3	4
0	1	2	4	6
0	1	2	3	4

```
{4,2} = 6 work = 4
```

----- test3 -----

6 1 2 7 3 5

----- Brute Force -----

```
1:{6} = 6
2:{1} = 1
3:{2} = 2
4:{6,2} = 8
5:{7} = 7
6:{6,7} = 13
7:{1,7} = 8
8:{3} = 3
9:{6,3} = 9
```

17.5. MAXIMUM SUBSET INDEPENDENT SET OF A PATH GRAPH

```
10:{1,3} = 4  
11:{2,3} = 5  
12:{6,2,3} = 11  
13:{5} = 5  
14:{6,5} = 11  
15:{1,5} = 6  
16:{2,5} = 7  
17:{6,2,5} = 13  
18:{7,5} = 12  
19:{6,7,5} = 18  
20:{1,7,5} = 13  
{6,7,5} = 18 work = 94
```

```
6   1   2   7   3   5  
----- Dynamic programming -----  
0   1   2   3   4   5   6  
0   6   1   2   7   3   5  
0   6   6   8   13  13  18  
0   6   6   2   7   7   5  
{5,7,6} = 18 work = 6  
----- test4 -----  
3 100   4   1   5 100   6  
----- Brute Force -----
```

```
1:{3} = 3  
2:{100} = 100  
3:{4} = 4  
4:{3,4} = 7  
5:{1} = 1  
6:{3,1} = 4  
7:{100,1} = 101  
8:{5} = 5  
9:{3,5} = 8  
10:{100,5} = 105  
11:{4,5} = 9  
12:{3,4,5} = 12  
13:{100} = 100  
14:{3,100} = 103  
15:{100,100} = 200  
16:{4,100} = 104  
17:{3,4,100} = 107  
18:{1,100} = 101
```

```

19:{3,1,100} = 104
20:{100,1,100} = 201
21:{6} = 6
22:{3,6} = 9
23:{100,6} = 106
24:{4,6} = 10
25:{3,4,6} = 13
26:{1,6} = 7
27:{3,1,6} = 10
28:{100,1,6} = 107
29:{5,6} = 11
30:{3,5,6} = 14
31:{100,5,6} = 111
32:{4,5,6} = 15
33:{3,4,5,6} = 18
{100,1,100} = 201 work = 201
-----
```

3	100	4	1	5	100	6	
----- Dynamic programming -----							
0	1	2	3	4	5	6	7
0	3	100	4	1	5	100	6
0	3	100	100	101	105	201	201
0	3	100	100	1	5	100	100

{100,1,100} = 201 work = 7

----- test5 -----

5	99	100	99	7
---	----	-----	----	---

----- Brute Force -----

```

1:{5} = 5
2:{99} = 99
3:{100} = 100
4:{5,100} = 105
5:{99} = 99
6:{5,99} = 104
7:{99,99} = 198
8:{7} = 7
9:{5,7} = 12
10:{99,7} = 106
11:{100,7} = 107
12:{5,100,7} = 112
{99,99} = 198 work = 43
-----
```

17.5. MAXIMUM SUBSET INDEPENDENT SET OF A PATH GRAPH

```
5 99 100 99 7
-----
----- Dynamic programming -----
0 1 2 3 4 5
0 5 99 100 99 7
0 5 99 105 198 198
0 5 99 100 99 99
{99,99} = 198 work = 5
-----
----- test6 -----
1 3 1 3 100
-----
----- Brute Force -----
1:{1} = 1
2:{3} = 3
3:{1} = 1
4:{1,1} = 2
5:{3} = 3
6:{1,3} = 4
7:{3,3} = 6
8:{100} = 100
9:{1,100} = 101
10:{3,100} = 103
11:{1,100} = 101
12:{1,1,100} = 102
{3,100} = 103 work = 43
-----
1 3 1 3 100
-----
----- Dynamic programming -----
0 1 2 3 4 5
0 1 3 1 3 100
0 1 3 3 6 103
0 1 3 3 3 100
{100,3} = 103 work = 5
-----
----- test7 -----
6 3 10 8 2 10 3 5 10 5 3
-----
----- Brute Force -----
1:{6} = 6
2:{3} = 3
3:{10} = 10
4:{6,10} = 16
5:{8} = 8
6:{6,8} = 14
7:{3,8} = 11
```

```
8:{2} = 2
9:{6,2} = 8
10:{3,2} = 5
11:{10,2} = 12
12:{6,10,2} = 18
13:{10} = 10
14:{6,10} = 16
15:{3,10} = 13
16:{10,10} = 20
17:{6,10,10} = 26
18:{8,10} = 18
19:{6,8,10} = 24
20:{3,8,10} = 21
21:{3} = 3
22:{6,3} = 9
23:{3,3} = 6
24:{10,3} = 13
25:{6,10,3} = 19
26:{8,3} = 11
27:{6,8,3} = 17
28:{3,8,3} = 14
29:{2,3} = 5
30:{6,2,3} = 11
31:{3,2,3} = 8
32:{10,2,3} = 15
33:{6,10,2,3} = 21
34:{5} = 5
35:{6,5} = 11
36:{3,5} = 8
37:{10,5} = 15
38:{6,10,5} = 21
39:{8,5} = 13
40:{6,8,5} = 19
41:{3,8,5} = 16
42:{2,5} = 7
43:{6,2,5} = 13
44:{3,2,5} = 10
45:{10,2,5} = 17
46:{6,10,2,5} = 23
47:{10,5} = 15
48:{6,10,5} = 21
```

17.5. MAXIMUM SUBSET INDEPENDENT SET OF A PATH GRAPH

49:{3,10,5} = 18
50:{10,10,5} = 25
51:{6,10,10,5} = 31
52:{8,10,5} = 23
53:{6,8,10,5} = 29
54:{3,8,10,5} = 26
55:{10} = 10
56:{6,10} = 16
57:{3,10} = 13
58:{10,10} = 20
59:{6,10,10} = 26
60:{8,10} = 18
61:{6,8,10} = 24
62:{3,8,10} = 21
63:{2,10} = 12
64:{6,2,10} = 18
65:{3,2,10} = 15
66:{10,2,10} = 22
67:{6,10,2,10} = 28
68:{10,10} = 20
69:{6,10,10} = 26
70:{3,10,10} = 23
71:{10,10,10} = 30
72:{6,10,10,10} = 36
73:{8,10,10} = 28
74:{6,8,10,10} = 34
75:{3,8,10,10} = 31
76:{3,10} = 13
77:{6,3,10} = 19
78:{3,3,10} = 16
79:{10,3,10} = 23
80:{6,10,3,10} = 29
81:{8,3,10} = 21
82:{6,8,3,10} = 27
83:{3,8,3,10} = 24
84:{2,3,10} = 15
85:{6,2,3,10} = 21
86:{3,2,3,10} = 18
87:{10,2,3,10} = 25
88:{6,10,2,3,10} = 31
89:{5} = 5

90:{6,5} = 11
91:{3,5} = 8
92:{10,5} = 15
93:{6,10,5} = 21
94:{8,5} = 13
95:{6,8,5} = 19
96:{3,8,5} = 16
97:{2,5} = 7
98:{6,2,5} = 13
99:{3,2,5} = 10
100:{10,2,5} = 17
101:{6,10,2,5} = 23
102:{10,5} = 15
103:{6,10,5} = 21
104:{3,10,5} = 18
105:{10,10,5} = 25
106:{6,10,10,5} = 31
107:{8,10,5} = 23
108:{6,8,10,5} = 29
109:{3,8,10,5} = 26
110:{3,5} = 8
111:{6,3,5} = 14
112:{3,3,5} = 11
113:{10,3,5} = 18
114:{6,10,3,5} = 24
115:{8,3,5} = 16
116:{6,8,3,5} = 22
117:{3,8,3,5} = 19
118:{2,3,5} = 10
119:{6,2,3,5} = 16
120:{3,2,3,5} = 13
121:{10,2,3,5} = 20
122:{6,10,2,3,5} = 26
123:{5,5} = 10
124:{6,5,5} = 16
125:{3,5,5} = 13
126:{10,5,5} = 20
127:{6,10,5,5} = 26
128:{8,5,5} = 18
129:{6,8,5,5} = 24
130:{3,8,5,5} = 21

17.5. MAXIMUM SUBSET INDEPENDENT SET OF A PATH GRAPH

131:{2,5,5} = 12
132:{6,2,5,5} = 18
133:{3,2,5,5} = 15
134:{10,2,5,5} = 22
135:{6,10,2,5,5} = 28
136:{10,5,5} = 20
137:{6,10,5,5} = 26
138:{3,10,5,5} = 23
139:{10,10,5,5} = 30
140:{6,10,10,5,5} = 36
141:{8,10,5,5} = 28
142:{6,8,10,5,5} = 34
143:{3,8,10,5,5} = 31
144:{3} = 3
145:{6,3} = 9
146:{3,3} = 6
147:{10,3} = 13
148:{6,10,3} = 19
149:{8,3} = 11
150:{6,8,3} = 17
151:{3,8,3} = 14
152:{2,3} = 5
153:{6,2,3} = 11
154:{3,2,3} = 8
155:{10,2,3} = 15
156:{6,10,2,3} = 21
157:{10,3} = 13
158:{6,10,3} = 19
159:{3,10,3} = 16
160:{10,10,3} = 23
161:{6,10,10,3} = 29
162:{8,10,3} = 21
163:{6,8,10,3} = 27
164:{3,8,10,3} = 24
165:{3,3} = 6
166:{6,3,3} = 12
167:{3,3,3} = 9
168:{10,3,3} = 16
169:{6,10,3,3} = 22
170:{8,3,3} = 14
171:{6,8,3,3} = 20

172:{3,8,3,3} = 17
173:{2,3,3} = 8
174:{6,2,3,3} = 14
175:{3,2,3,3} = 11
176:{10,2,3,3} = 18
177:{6,10,2,3,3} = 24
178:{5,3} = 8
179:{6,5,3} = 14
180:{3,5,3} = 11
181:{10,5,3} = 18
182:{6,10,5,3} = 24
183:{8,5,3} = 16
184:{6,8,5,3} = 22
185:{3,8,5,3} = 19
186:{2,5,3} = 10
187:{6,2,5,3} = 16
188:{3,2,5,3} = 13
189:{10,2,5,3} = 20
190:{6,10,2,5,3} = 26
191:{10,5,3} = 18
192:{6,10,5,3} = 24
193:{3,10,5,3} = 21
194:{10,10,5,3} = 28
195:{6,10,10,5,3} = 34
196:{8,10,5,3} = 26
197:{6,8,10,5,3} = 32
198:{3,8,10,5,3} = 29
199:{10,3} = 13
200:{6,10,3} = 19
201:{3,10,3} = 16
202:{10,10,3} = 23
203:{6,10,10,3} = 29
204:{8,10,3} = 21
205:{6,8,10,3} = 27
206:{3,8,10,3} = 24
207:{2,10,3} = 15
208:{6,2,10,3} = 21
209:{3,2,10,3} = 18
210:{10,2,10,3} = 25
211:{6,10,2,10,3} = 31
212:{10,10,3} = 23

17.6. 0/1 KNAPSACK PROBLEM

```
213:{6,10,10,3} = 29  
214:{3,10,10,3} = 26  
215:{10,10,10,3} = 33  
216:{6,10,10,10,3} = 39  
217:{8,10,10,3} = 31  
218:{6,8,10,10,3} = 37  
219:{3,8,10,10,3} = 34  
220:{3,10,3} = 16  
221:{6,3,10,3} = 22  
222:{3,3,10,3} = 19  
223:{10,3,10,3} = 26  
224:{6,10,3,10,3} = 32  
225:{8,3,10,3} = 24  
226:{6,8,3,10,3} = 30  
227:{3,8,3,10,3} = 27  
228:{2,3,10,3} = 18  
229:{6,2,3,10,3} = 24  
230:{3,2,3,10,3} = 21  
231:{10,2,3,10,3} = 28  
232:{6,10,2,3,10,3} = 34  
{6,10,10,10,3} = 39 work = 3719
```

6 3 10 8 2 10 3 5 10 5 3
----- Dynamic programming -----
0 1 2 3 4 5 6 7 8 9 10 11
0 6 3 10 8 2 10 3 5 10 5 3
0 6 6 16 16 18 26 26 31 36 36 39
0 6 6 10 10 2 10 10 5 10 10 3
{3,10,10,10,6} = 39 work = 11
MaxPath problem ENDS

17.6 0/1 Knapsack problem

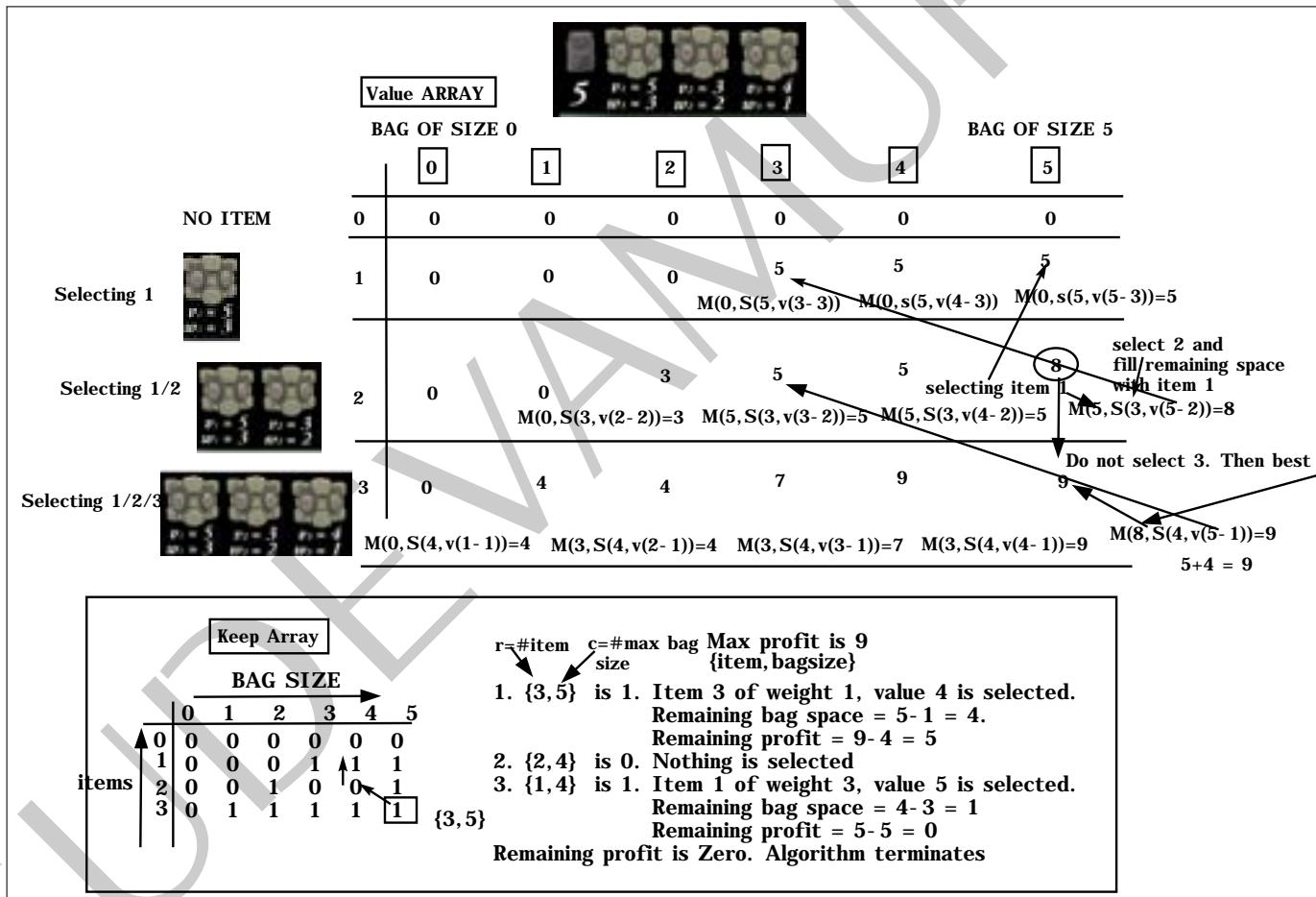


Figure 17.6: Building optimal table using dynamic programming

17.6. 0/1 KNAPSACK PROBLEM

```

----- V MATRIX -----
0 0 0 0 0 0
0 0 0 5 5 5
0 0 3 5 5 8
0 4 4 7 9 9
                w = [3,2,1]
                v = [5,3,4]
                Bag size 5

----- k matrix -----
0 0 0 0 0 0
0 0 0 1 1 1
0 0 1 0 0 1
0 1 1 1 1 1
i =
1 2 3
w =
3 2 1
v =
5 3 4
Max Value of 9 can obtained from items {3,1} that has values {4+5=9}

----- V matrix -----
0 0 0 0 0 0
0 0 12 12 12 12
0 10 12 22 22 22
0 10 12 22 30 32
0 10 15 25 30 37
                w = [2,1,3,2]
                v = [12,10,20,15]
                Bag size 5

----- k matrix -----
0 0 0 0 0 0
0 0 1 1 1 1
0 1 0 1 1 1
0 0 0 0 1 1
0 0 1 1 0 1
i =
1 2 3 4
w =
2 1 3 2
v =
12 10 20 15
Max Value of 37 can obtained from items {4,2,1} that has values {15+10+12=37}

----- V matrix -----
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 10 10 10 10 10
0 0 0 0 40 40 40 40 40 50
0 0 0 0 40 40 40 40 40 50
0 0 0 50 50 50 50 50 50 50
                int[] w = {5,4,6,3};
                int[] v = {10,40,30,50};
                Bag size 10

----- k matrix -----
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1
0 0 0 0 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 1
0 0 0 1 1 1 1 1 1 1
i =
1 2 3 4
w =
5 4 6 3
v =
10 40 30 50
Max Value of 90 can obtained from items {4,2} that has values {50+40=90}

```

Figure 17.7: Animation of 0/1 knapsack problem using dynamic programming

17.7 Single source shortest path algorithm using dynamic programming

17.7. SINGLE SOURCE SHORTEST PATH ALGORITHM USING DYNAMIC PROGRAMMING

Find the minimum distance between 0 and 6

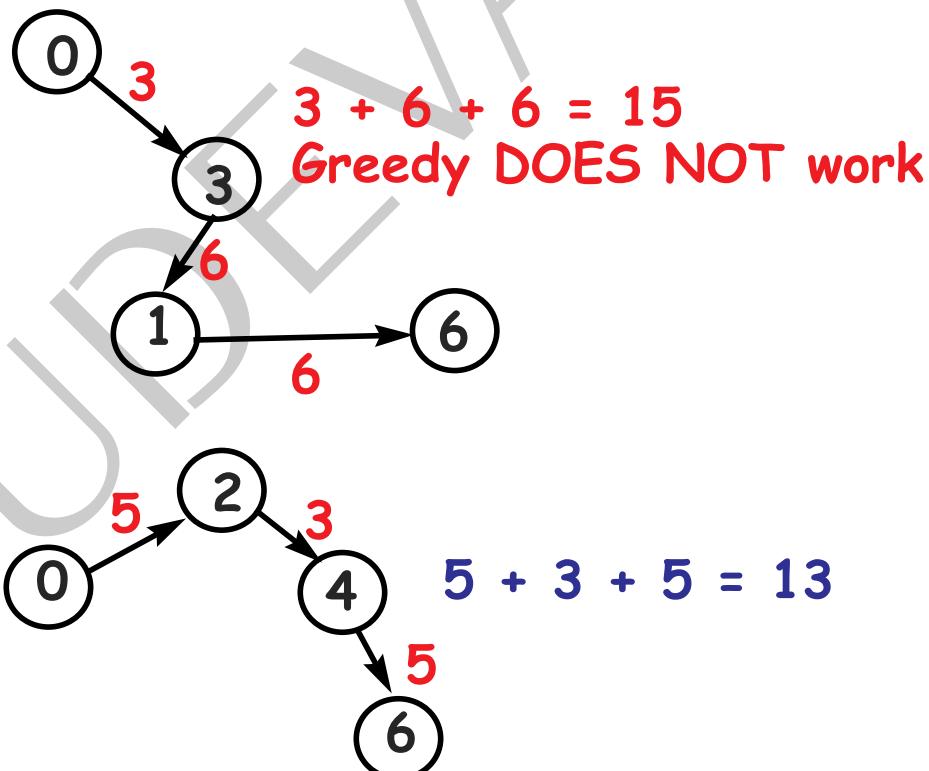
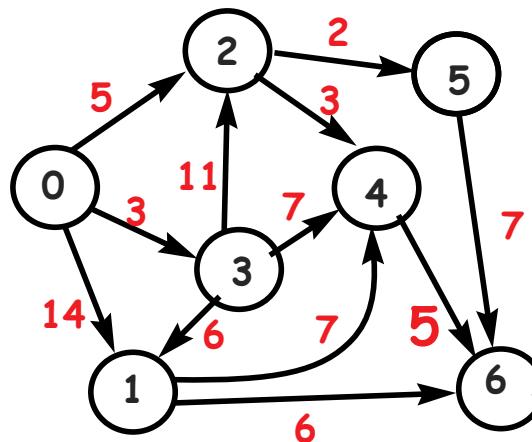


Figure 17.8: Greedy algorithm that does not work

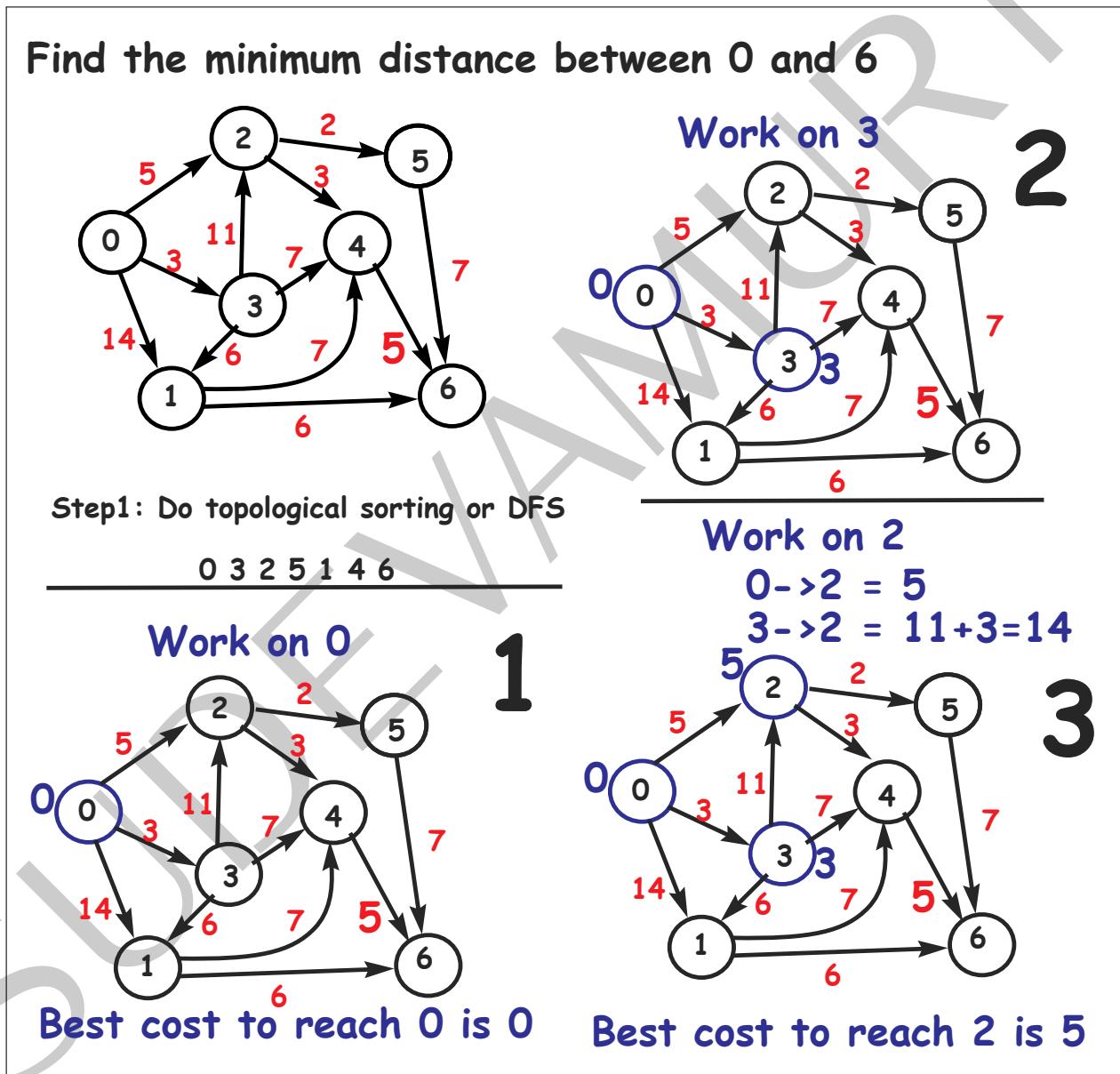


Figure 17.9: Dynamic programming

17.7. SINGLE SOURCE SHORTEST PATH ALGORITHM USING DYNAMIC PROGRAMMING

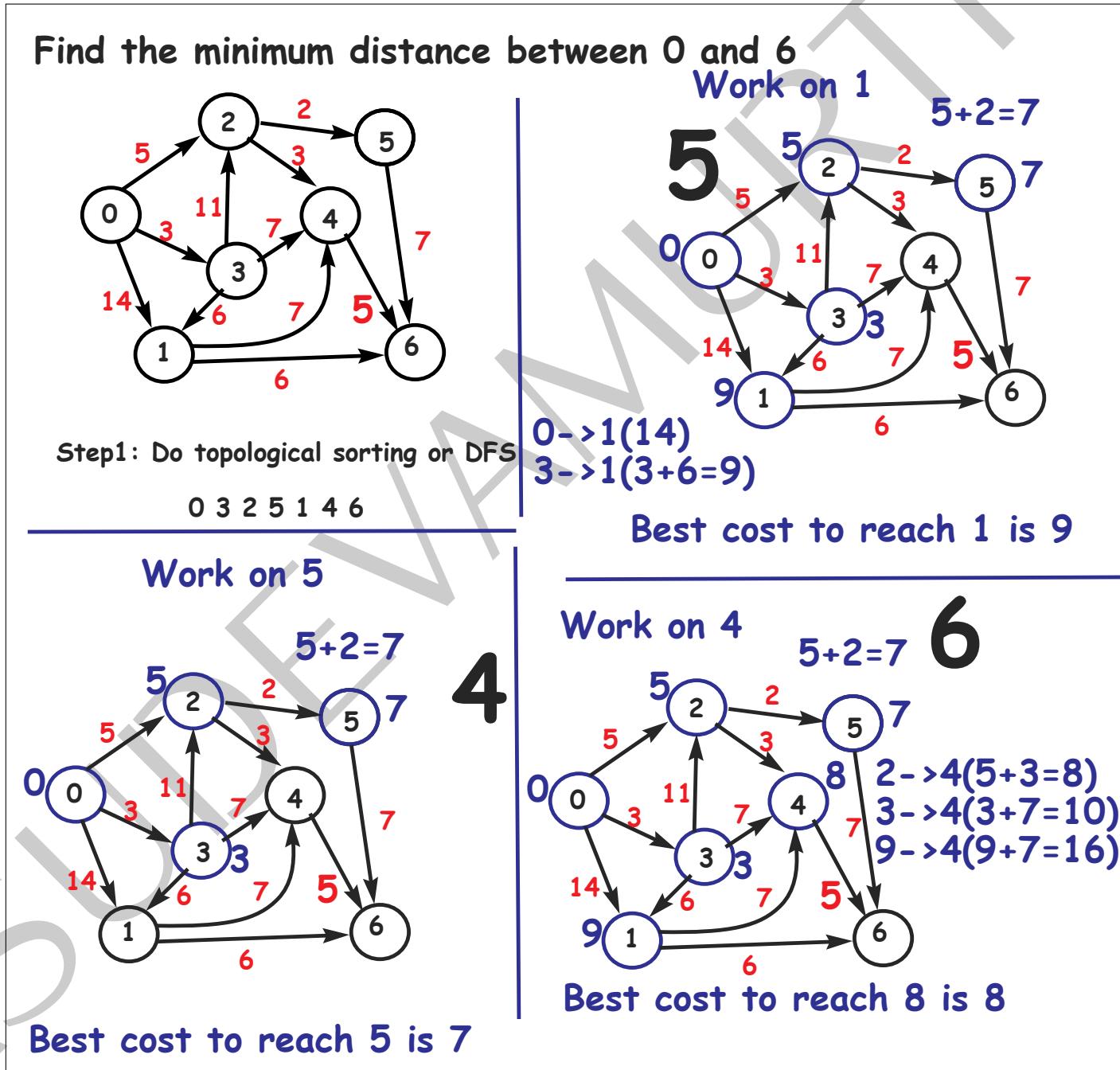


Figure 17.10: Dynamic programming (continued)

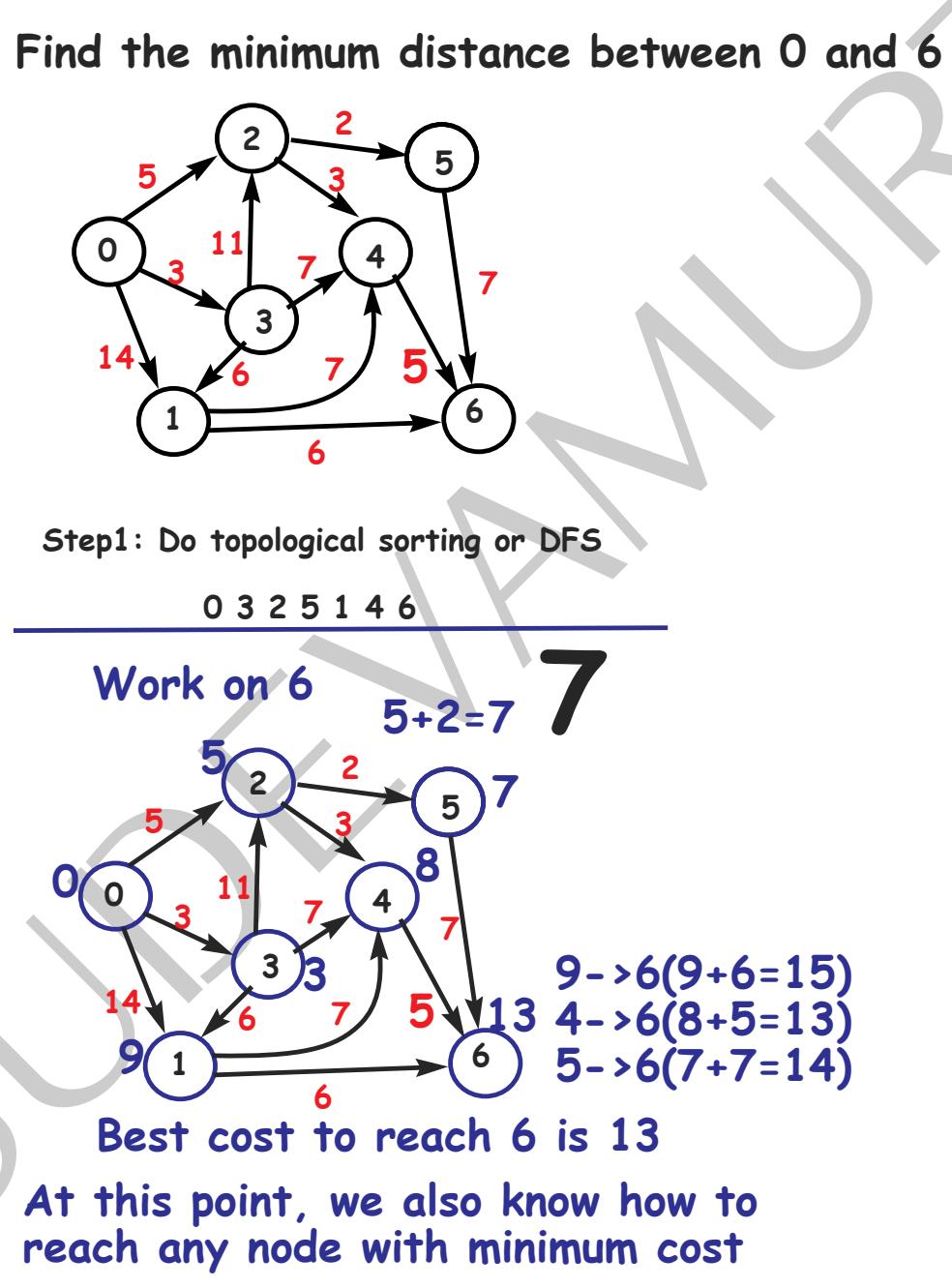


Figure 17.11: Dynamic programming (continued)

17.7. SINGLE SOURCE SHORTEST PATH ALGORITHM USING DYNAMIC PROGRAMMING

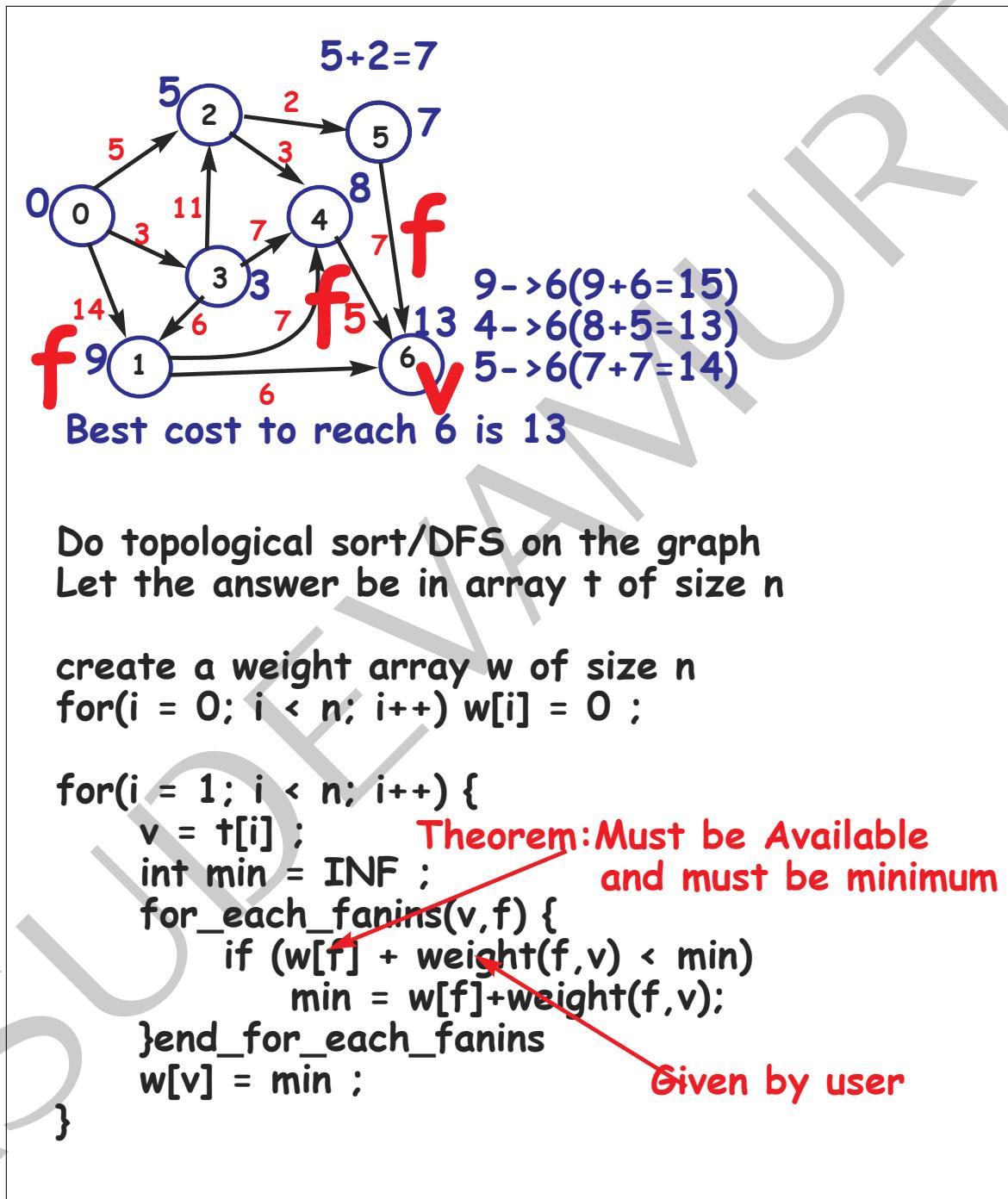


Figure 17.12: Dynamic programming algorithm

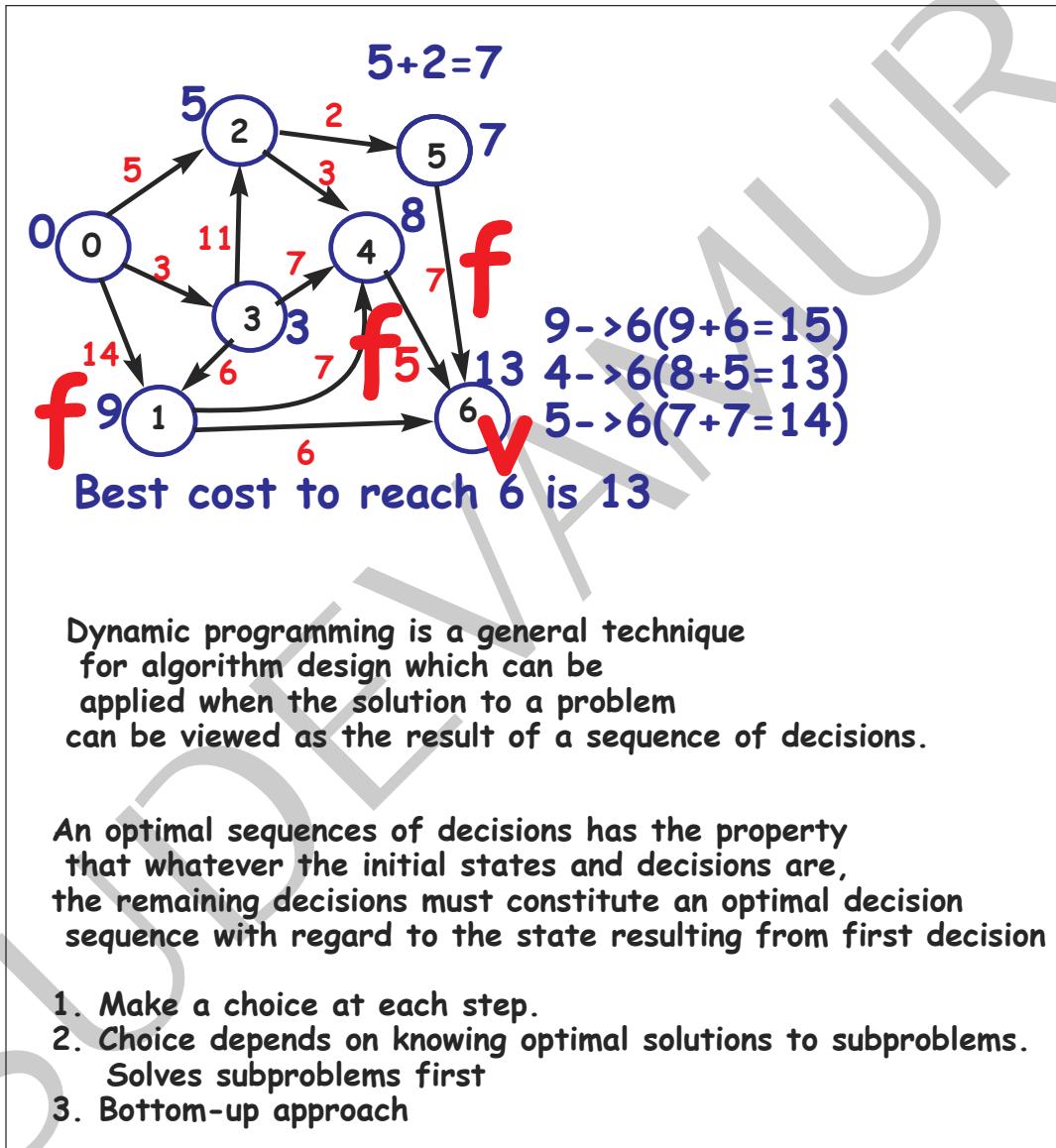


Figure 17.13: Principle of optimality

17.7. SINGLE SOURCE SHORTEST PATH ALGORITHM USING DYNAMIC PROGRAMMING

VASUDEVAMURTHY

Chapter 18

Graphs

18.1 Introduction

18.2 Graph applications

18.2.1 Transportation problem

18.2. GRAPH APPLICATIONS

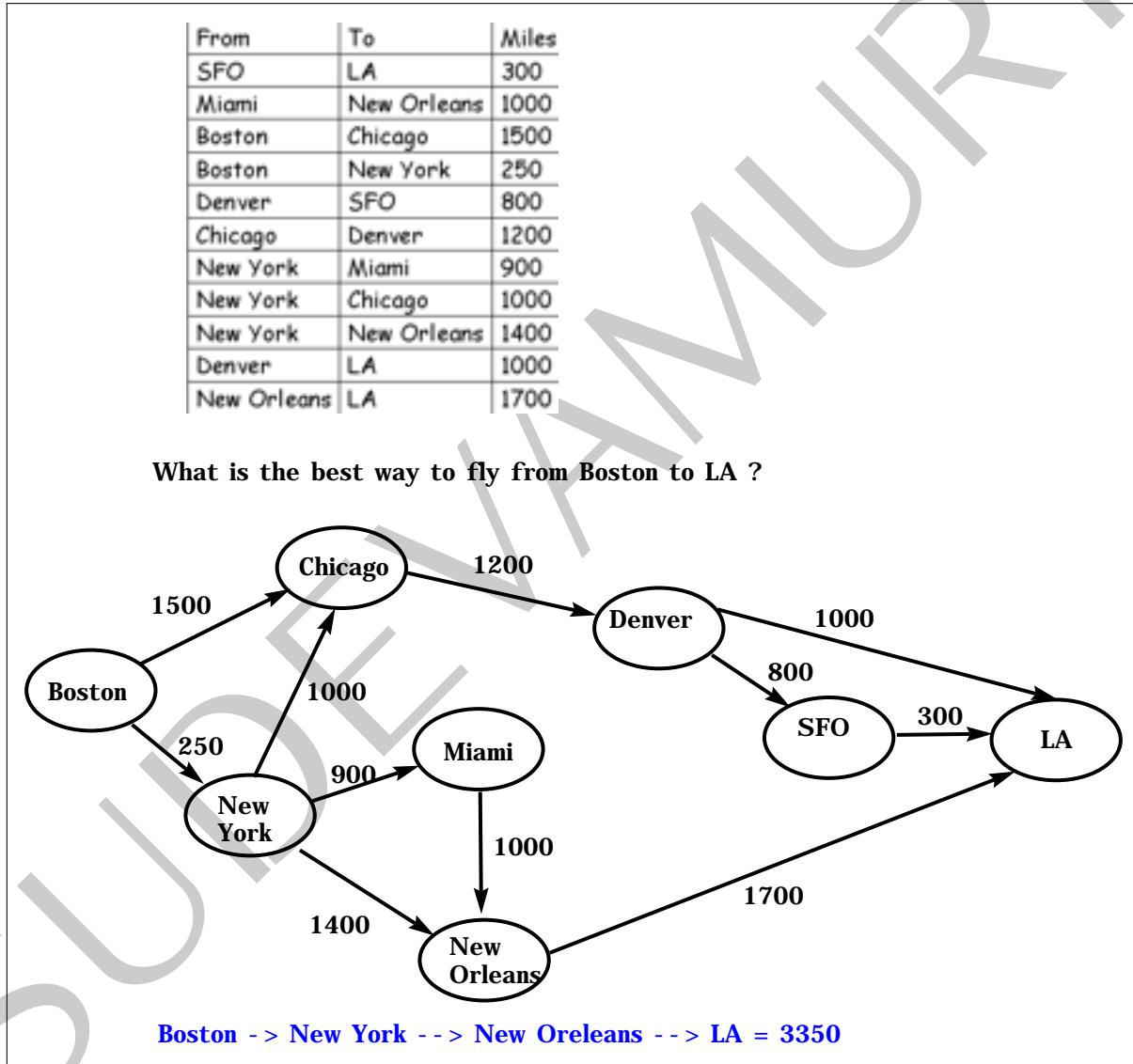


Figure 18.1: What is the best way to fly from Boston to LA?

18.2.2 Minimum connector problem

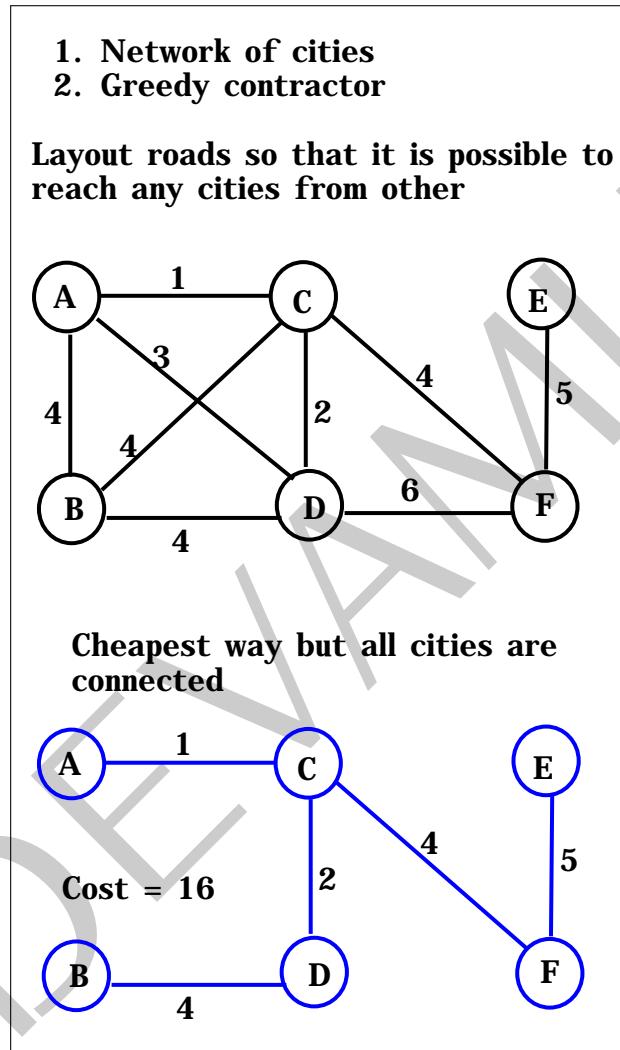


Figure 18.2: Laying cheapest road

18.2.3 Scheduling problem

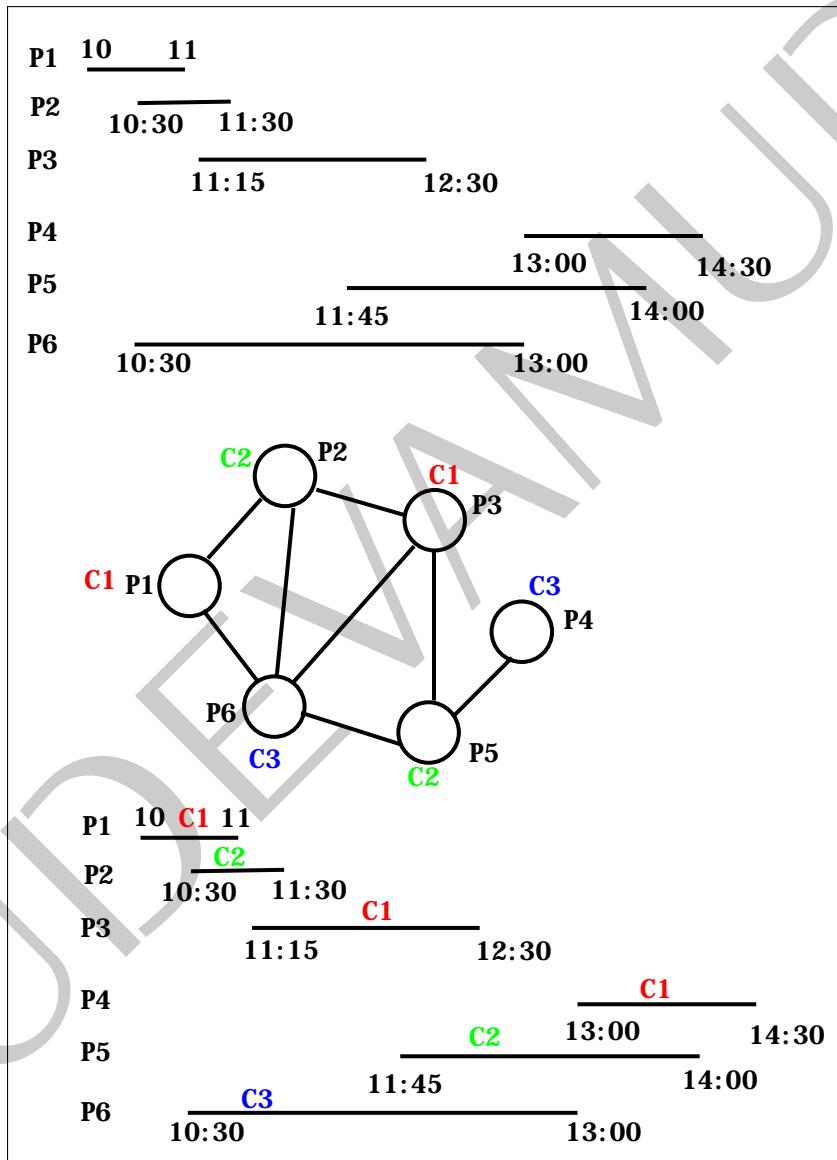


Figure 18.3: Minimum channels required to broadcast seven programs

18.2.4 Activity network or Topological sorting problem

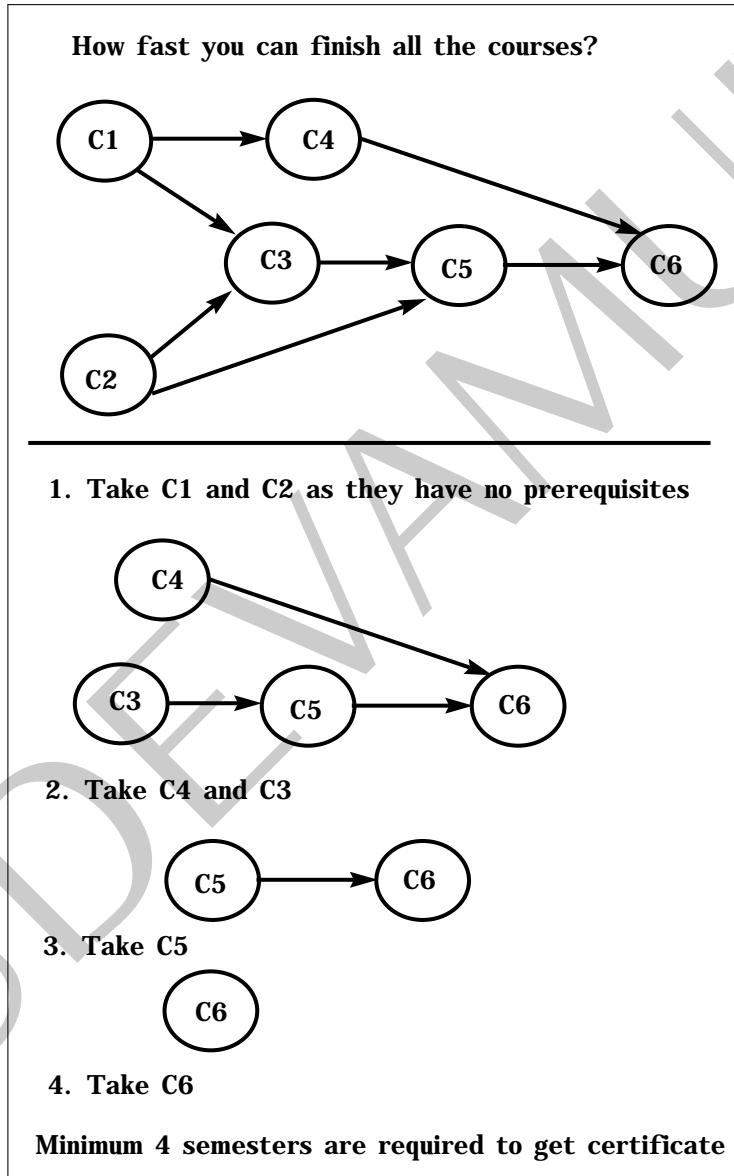


Figure 18.4: Completing courses in an university

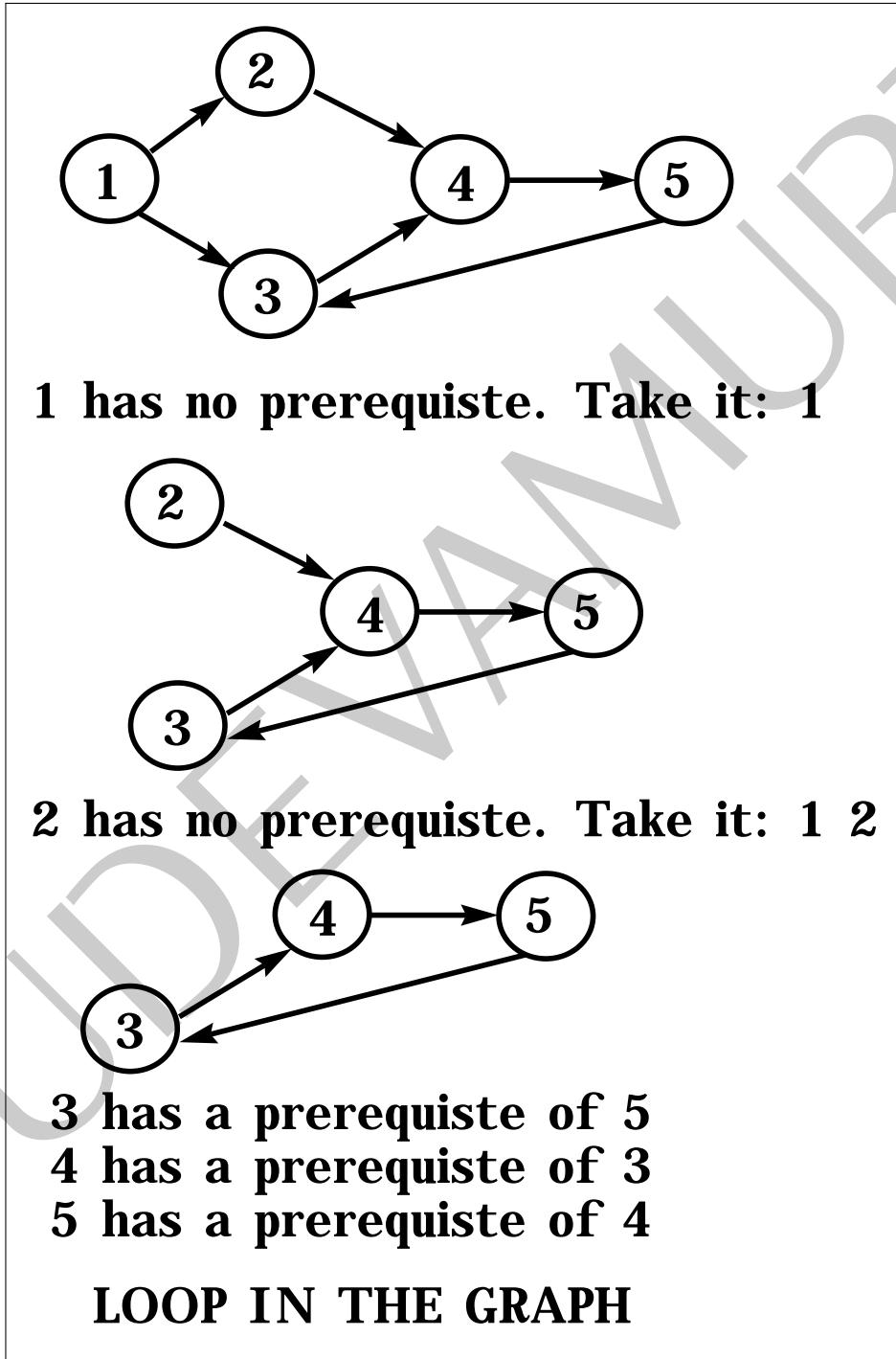
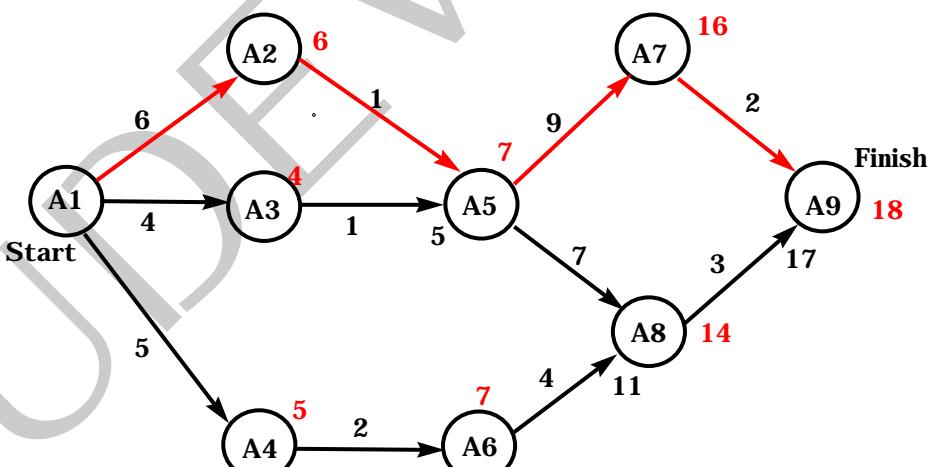
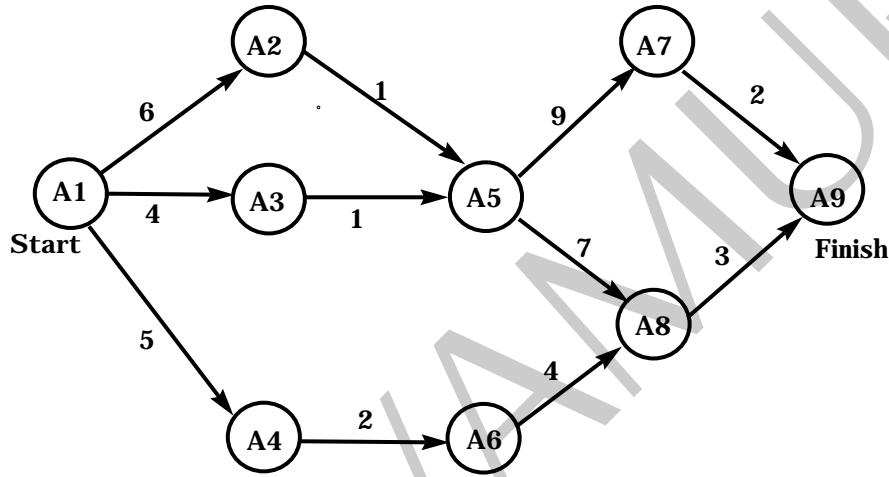


Figure 18.5: Impossible to complete courses

18.2.5 Critical path analysis

A1, A2, ... A9 are the activites in the projects.
How fast you can complete the project?
What is the critical path in your project?



You cannot finish the project in less than 18 days
Critical path is : A1, A2, A5, A7 and A9

Figure 18.6: Critical path of a project

18.2.6 Flow problem

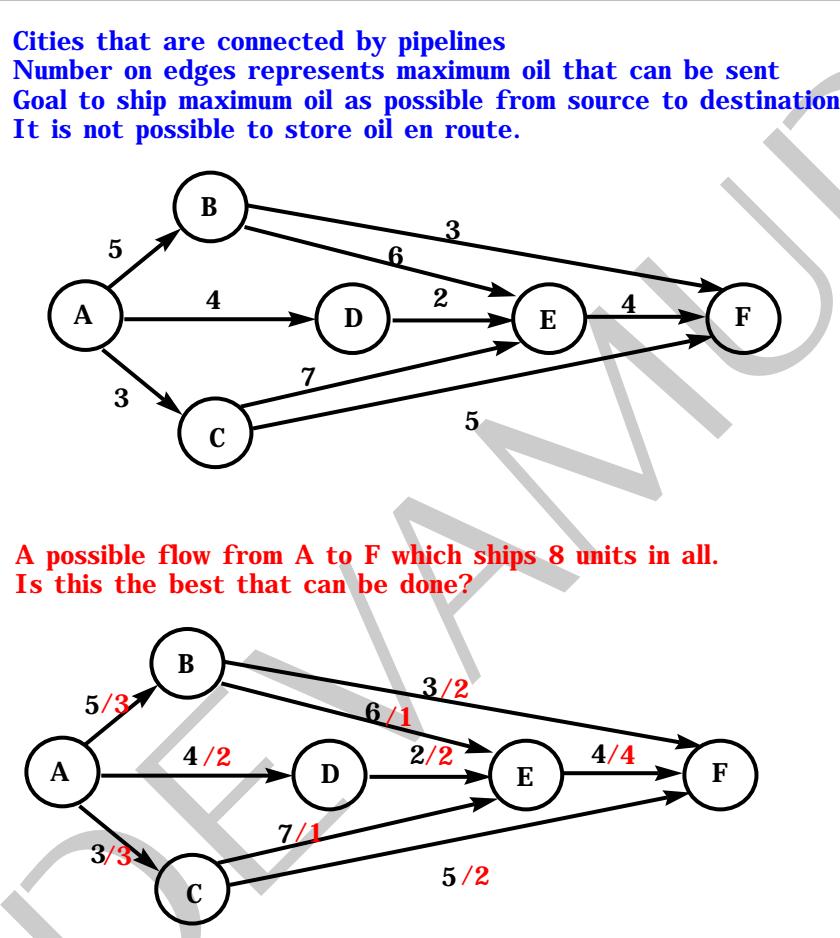


Figure 18.7: Maximum flow possible

18.3 Graph examples

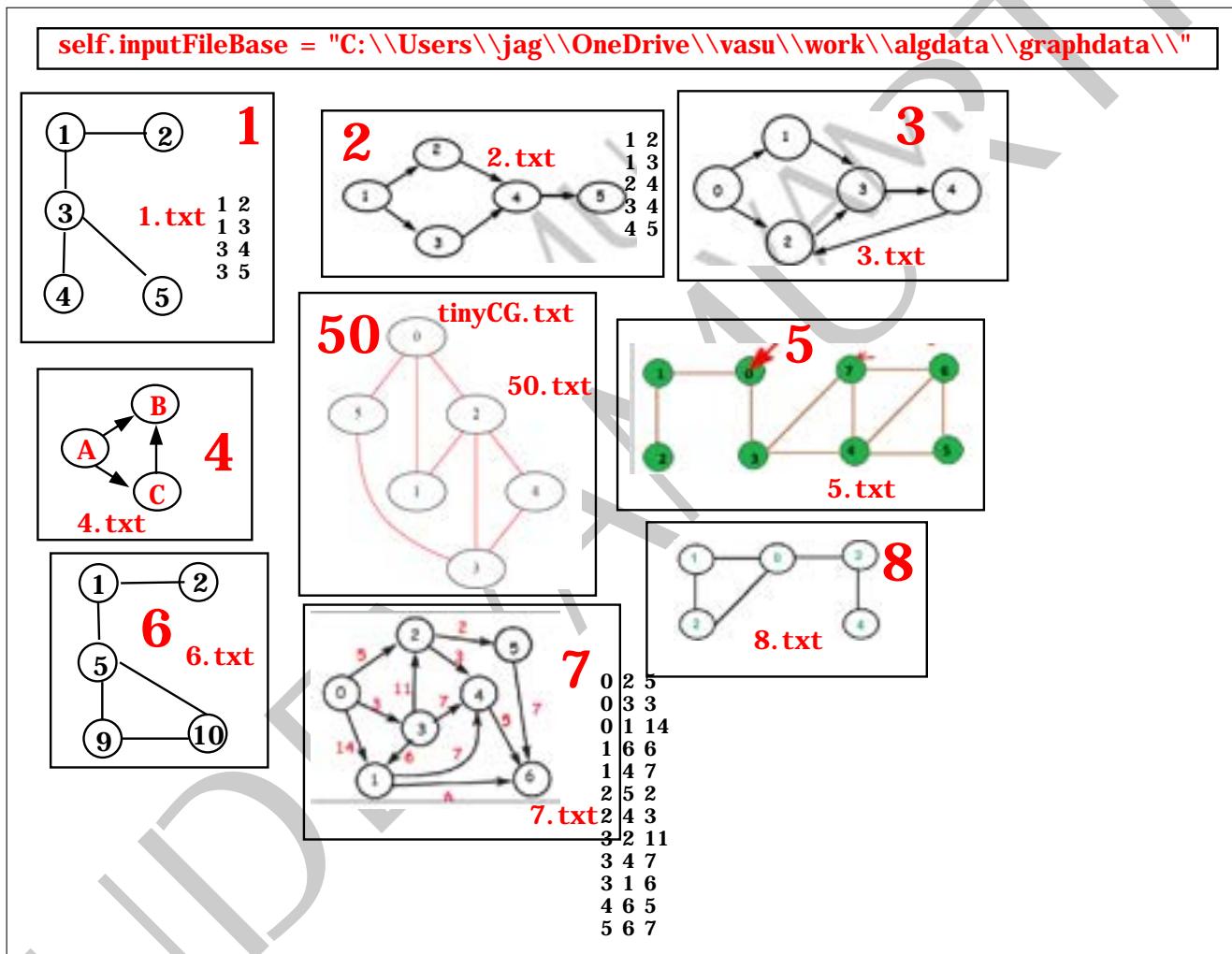


Figure 18.8: Various graphs

18.3. GRAPH EXAMPLES

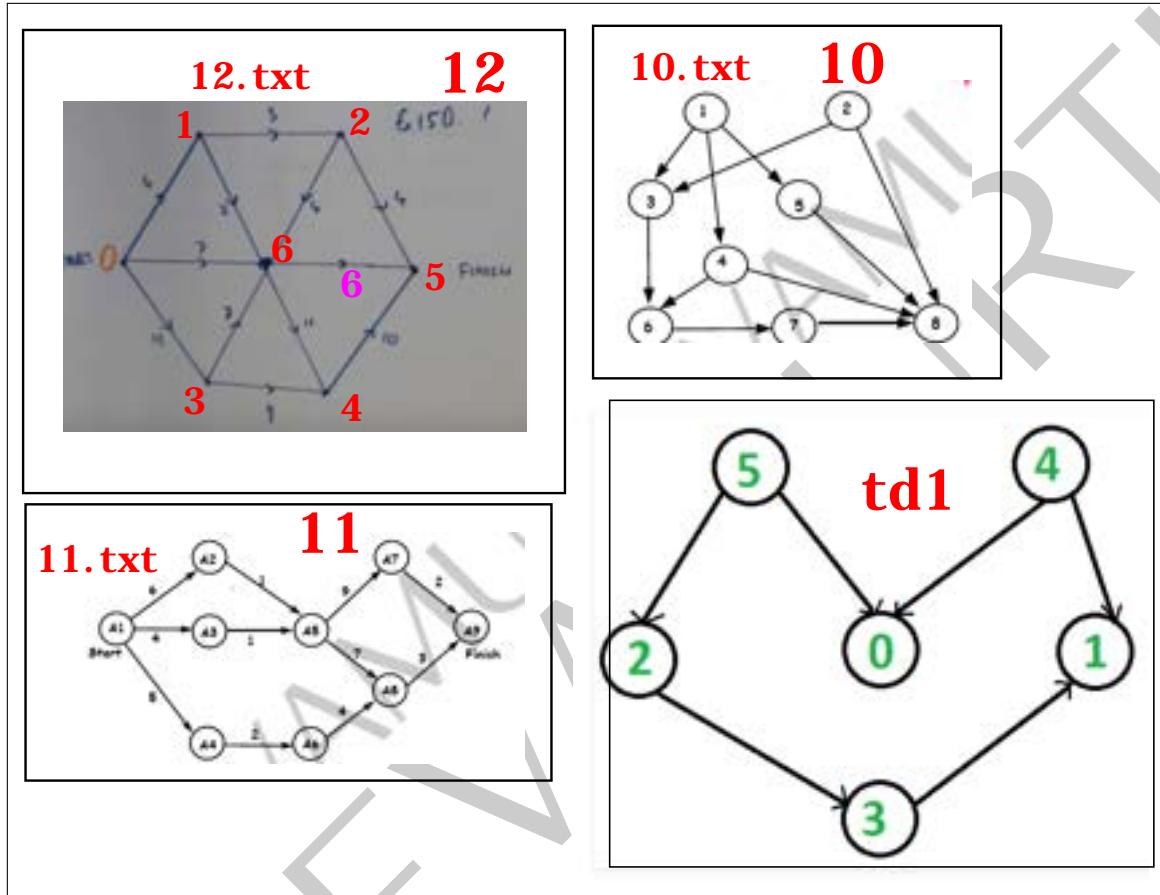


Figure 18.9: Various graphs

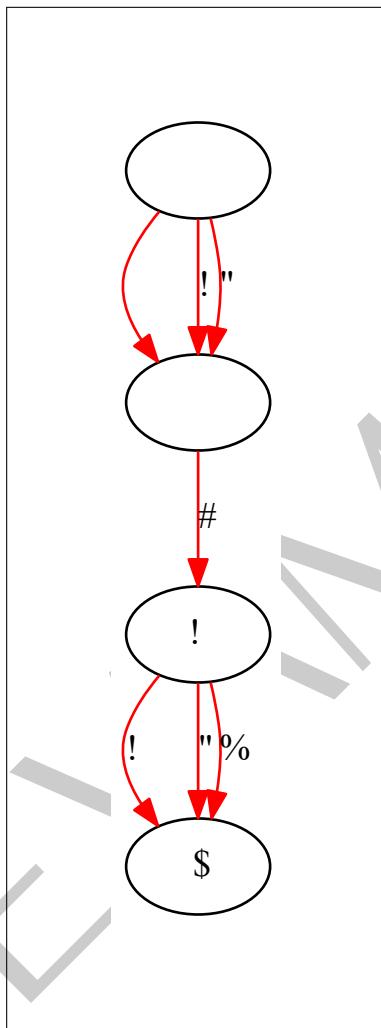


Figure 18.10: Parallel edges

18.4. GRAPH REPRESENTATION USING MATRICES

18.4 Graph representation using matrices

18.4.1 Undirected graph representation

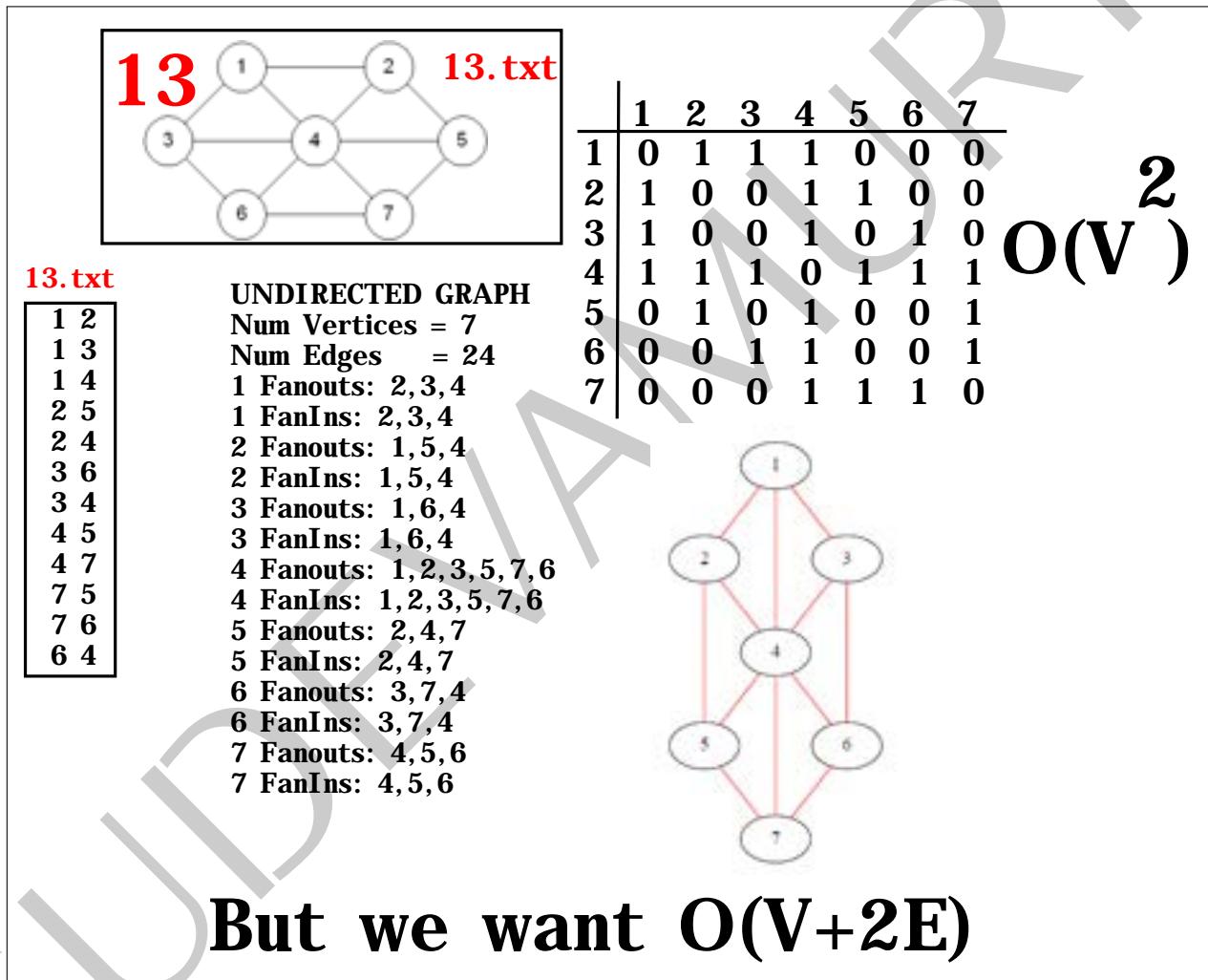


Figure 18.11: Representation of an undirected graph

18.4.2 Undirected weighted graph representation

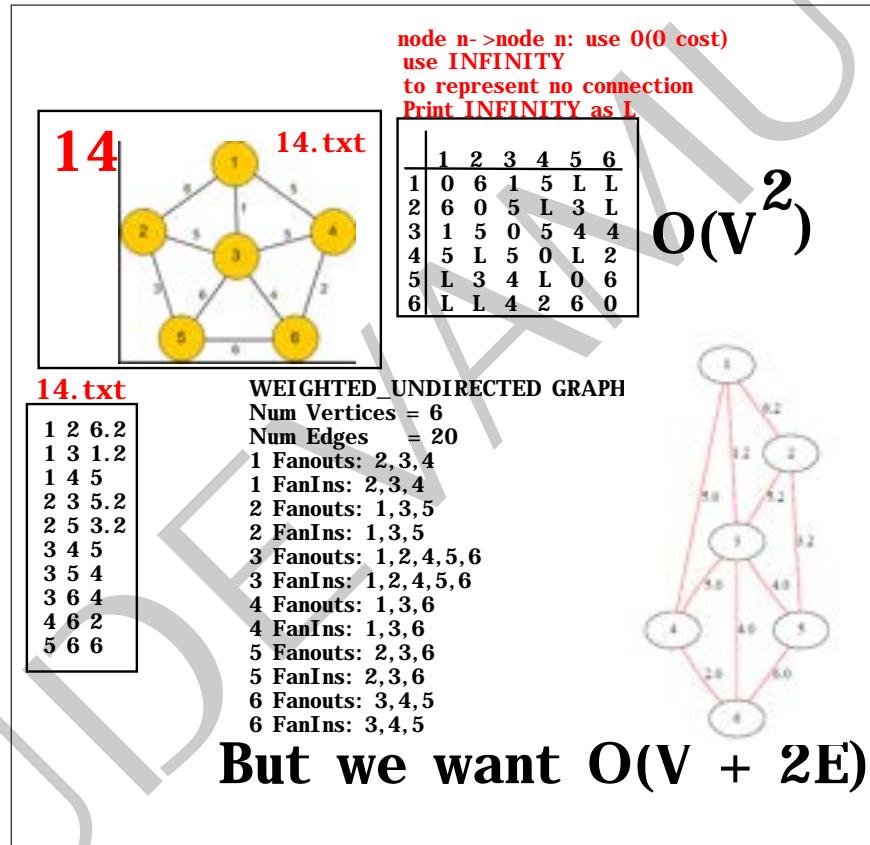


Figure 18.12: Representation of an undirected weighted graph

18.4. GRAPH REPRESENTATION USING MATRICES

18.4.3 Directed graph representation

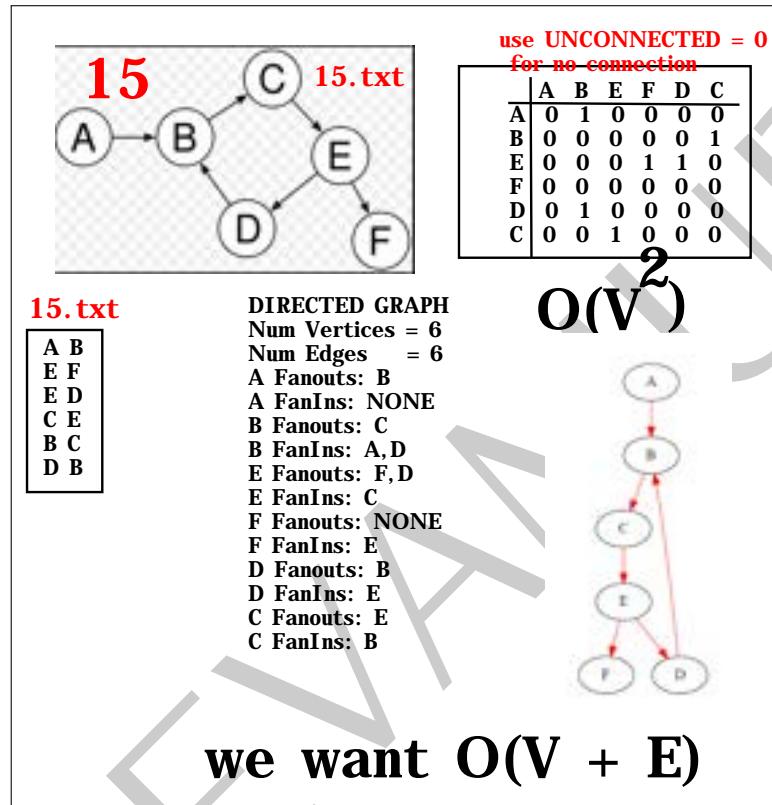


Figure 18.13: Representation of a directed graph

18.4.4 Directed weighted graph representation

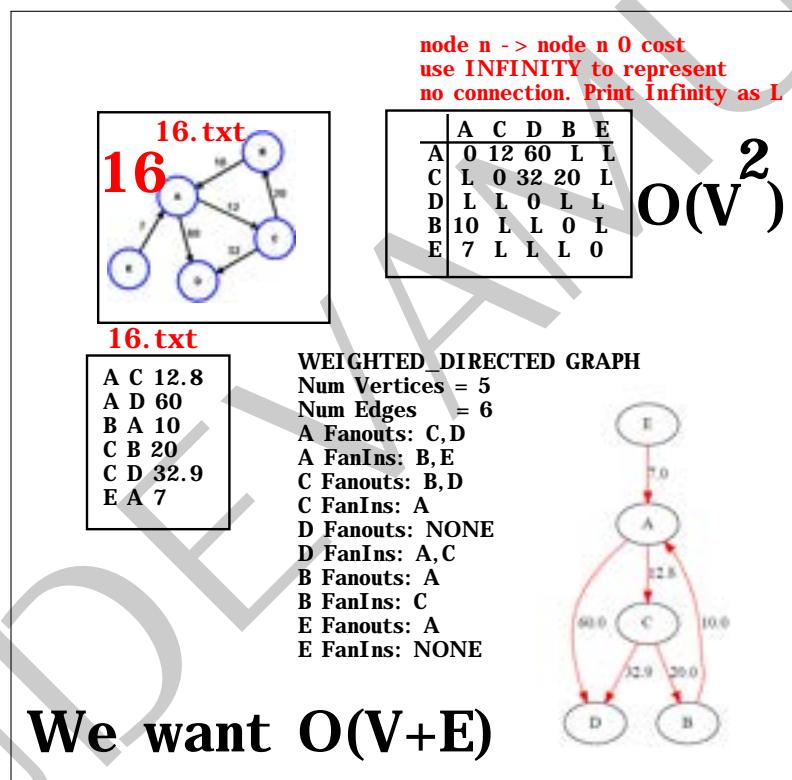


Figure 18.14: Representation of a directed weighted graph

18.5. GRAPH REPRESENTATION USING FANINS AND FANOUTS LISTS

18.5 Graph representation using fanins and fanouts lists

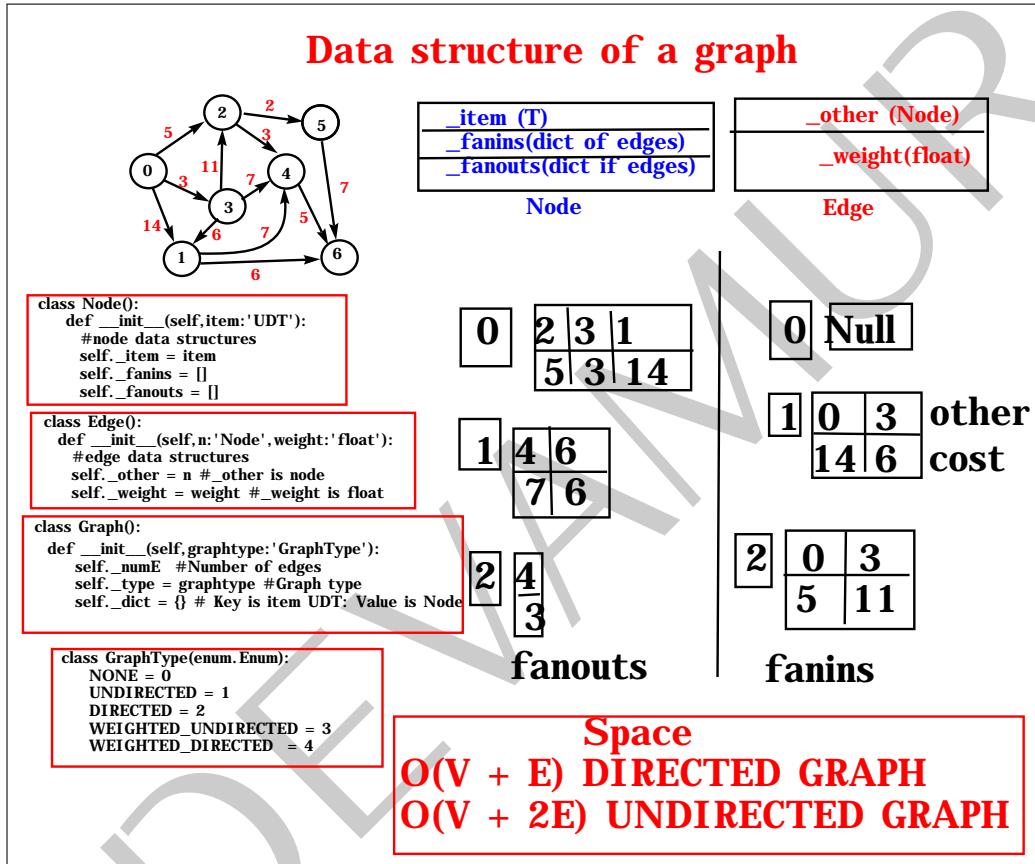


Figure 18.15: Graph data structure

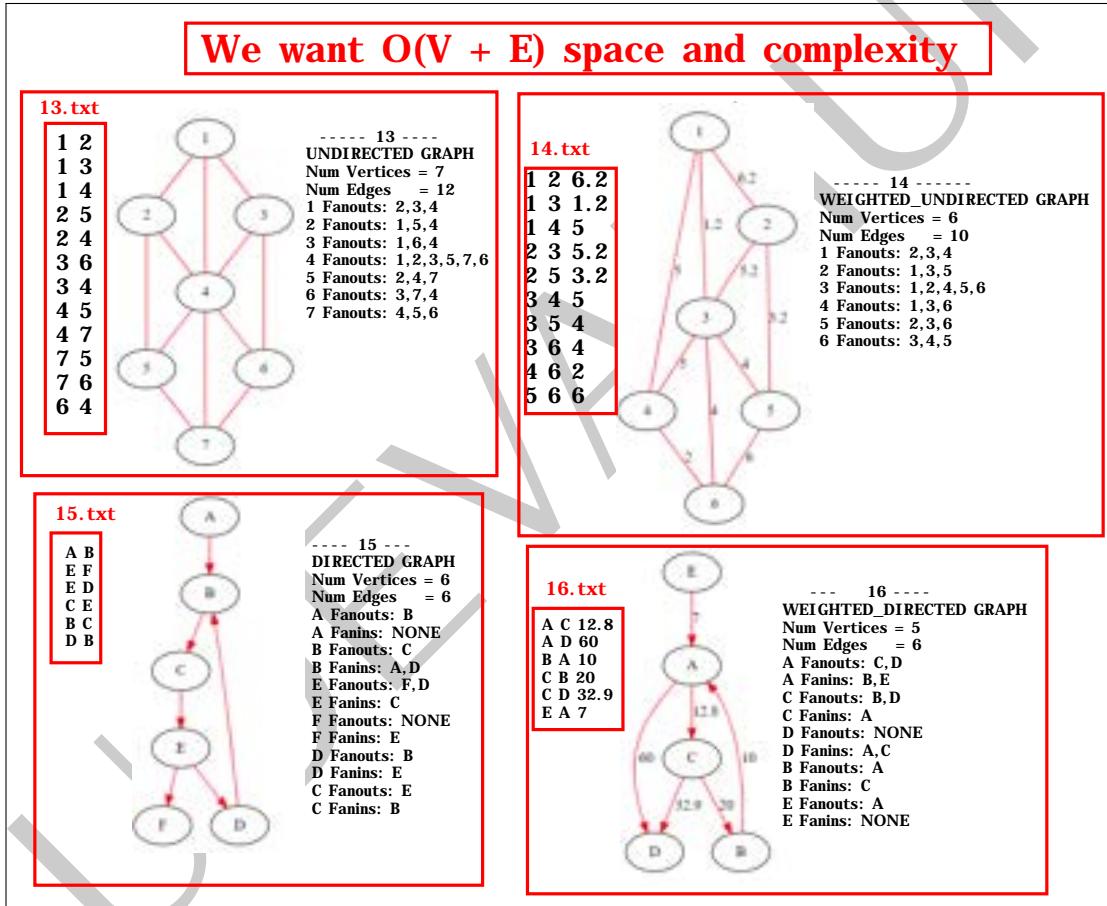


Figure 18.16: Representation of a graph

18.6 graphviz package

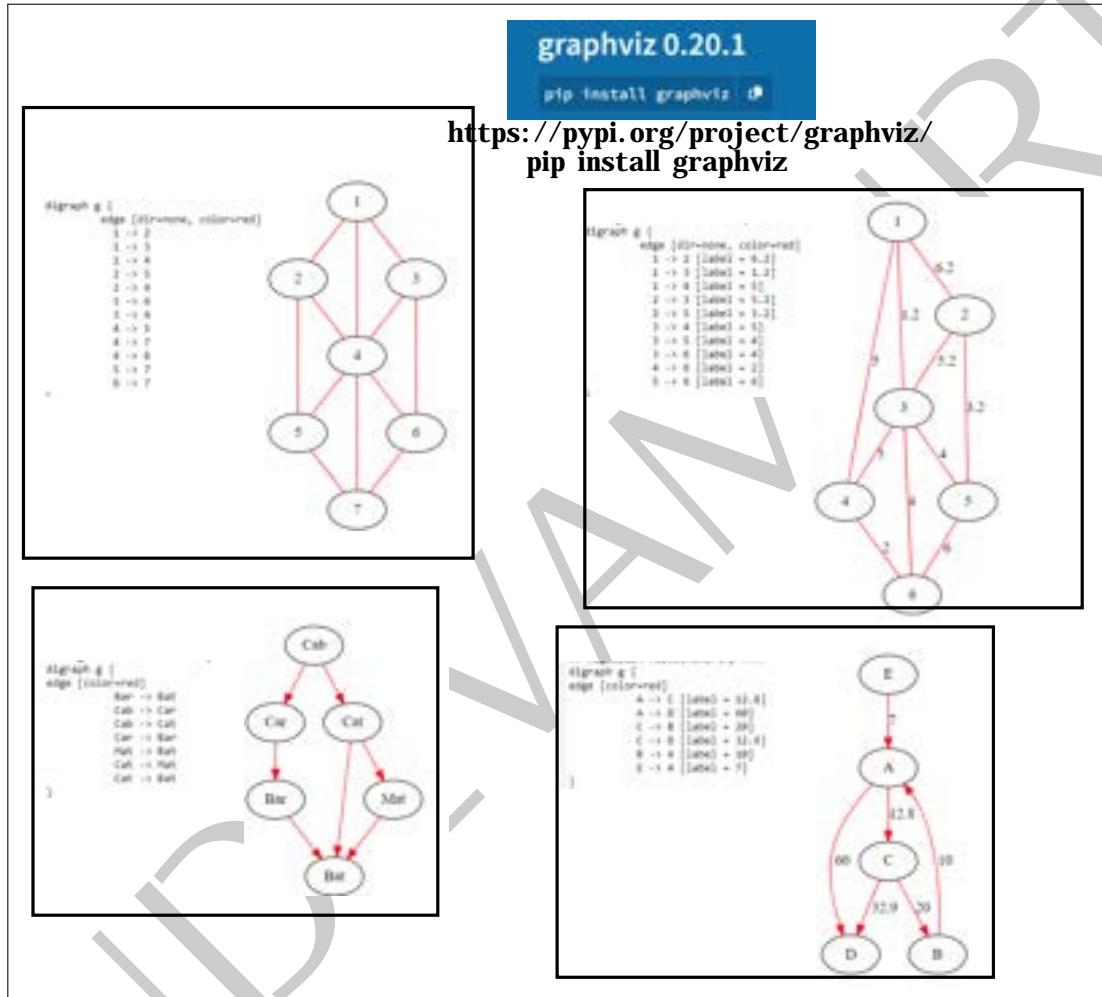


Figure 18.17: *graphviz* pacakge

Drawing graph

Copyright: Jagadeesh Vasudevamurthy

filename: graphviz.ipynb

Basic imports

In [42]:

```
1 import sys
2 import os
3 from graphviz import Digraph
4 print(sys.version)
```

3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]

Generic Read dot file

In [43]:

```
1 from graphviz import Source
2 def readDotFile(filename:'string')->'dot_graph':
3     Base = "C:\\\\Users\\\\jag\\\\OneDrive\\\\vasu\\\\work\\\\py3\\\\objects\\\\py3\\\\py3\\\\"
4     file = Base + filename + ".dot"
5     print(file)
6     with open(file) as f:
7         dot_graph = f.read()
8     print(dot_graph)
9     return dot_graph
10
```

Undirected Graphs With NO Weight

In [44]:

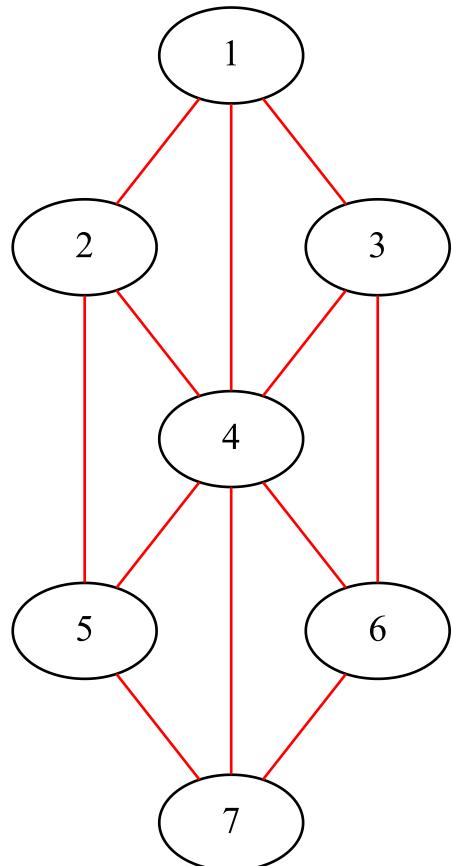
```
1 ...
2 #File: 13.dot
3 #Jagadeesh Vasudevamurthy
4 digraph g {
5     edge [dir=none, color=red]
6     1 -> 2
7     1 -> 3
8     1 -> 4
9     2 -> 5
10    2 -> 4
11    3 -> 6
12    3 -> 4
13    4 -> 5
14    4 -> 7
15    4 -> 6
16    5 -> 7
17    6 -> 7
18 }
19 ...
20 Source(readDotFile("13"))
```

```
C:\Users\jag\OneDrive\vasu\work\py3\objects\py3\py3\dot\13.dot
```

```
## Jagadeesh Vasudevamurthy ####
```

```
digraph g {
    edge [dir=none, color=red]
    1 -> 2
    1 -> 3
    1 -> 4
    2 -> 5
    2 -> 4
    3 -> 6
    3 -> 4
    4 -> 5
    4 -> 7
    4 -> 6
    5 -> 7
    6 -> 7
}
```

Out[44]:



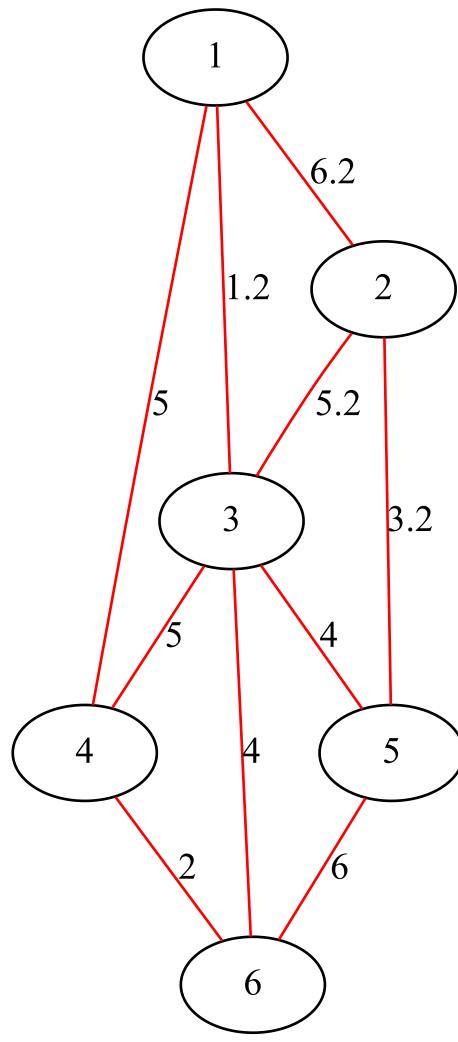
In [45]:

```
1 ...
2 #FILE: 14.dot
3 ## Jagadeesh Vasudevamurthy ####
4 digraph g {
5     edge [dir=none, color=red]
6     1 -> 2 [label = 6.2]
7     1 -> 3 [label = 1.2]
8     1 -> 4 [label = 5]
9     2 -> 3 [label = 5.2]
10    2 -> 5 [label = 3.2]
11    3 -> 4 [label = 5]
12    3 -> 5 [label = 4]
13    3 -> 6 [label = 4]
14    4 -> 6 [label = 2]
15    5 -> 6 [label = 6]
16 }
17 ...
18 Source(readDotFile("14"))
```

```
C:\Users\jag\OneDrive\vasu\work\py3\objects\py3\py3\dot\14.dot
```

```
## Jagadeesh Vasudevamurthy ####
digraph g {
    edge [dir=none, color=red]
    1 -> 2 [label = 6.2]
    1 -> 3 [label = 1.2]
    1 -> 4 [label = 5]
    2 -> 3 [label = 5.2]
    2 -> 5 [label = 3.2]
    3 -> 4 [label = 5]
    3 -> 5 [label = 4]
    3 -> 6 [label = 4]
    4 -> 6 [label = 2]
    5 -> 6 [label = 6]
}
```

Out[45]:



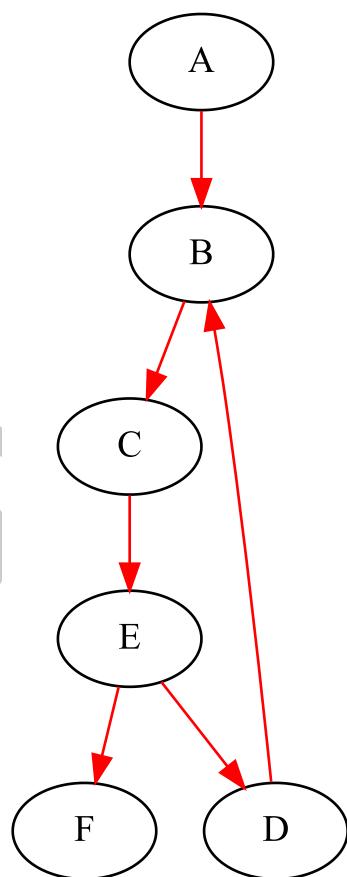
Directed Graphs With NO Weight

In [46]:

```
1 ...
2 #File: 15.dot
3 ## Jagadeesh Vasudevamurthy ####
4 digraph g {
5     edge [color=red]
6         A -> B
7         B -> C
8         E -> F
9         E -> D
10        D -> B
11        C -> E
12    }
13
14 ...
15 Source(readDotFile("15"))
```

```
C:\Users\jag\OneDrive\vasu\work\py3\objects\py3\py3\dot\15.dot
## Jagadeesh Vasudevamurthy ####
digraph g {
edge [color=red]
    A -> B
    B -> C
    E -> F
    E -> D
    D -> B
    C -> E
}
```

Out[46]:



Directed Graph With Weight

In [47]:

```

1  ...
2  #File: 16.dot
3  ## Jagadeesh Vasudevamurthy ####
4  digraph g {
5      edge [color=red]
6          A -> C [label = 12.8]
7          A -> D [label = 60]
8          C -> B [label = 20]
9          C -> D [label = 32.9]
10         B -> A [label = 10]
11         E -> A [label = 7]
12     }
13
14 ...
15 Source(readDotFile("16"))

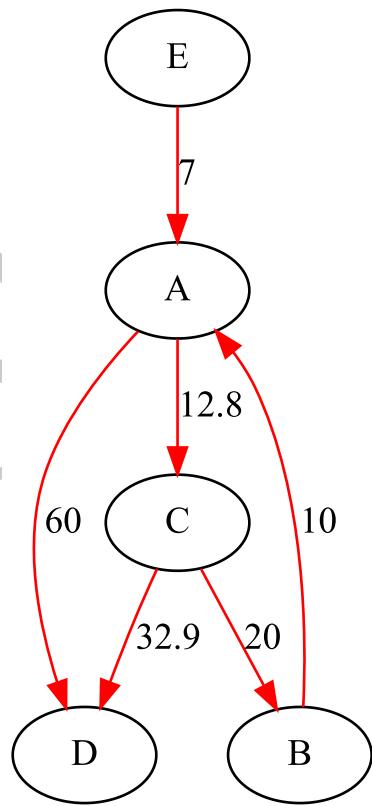
```

```

C:\Users\jag\OneDrive\vasu\work\py3\objects\py3\py3\dot\16.dot
## Jagadeesh Vasudevamurthy ####
digraph g {
edge [color=red]
    A -> C [label = 12.8]
    A -> D [label = 60]
    C -> B [label = 20]
    C -> D [label = 32.9]
    B -> A [label = 10]
    E -> A [label = 7]
}

```

Out[47]:



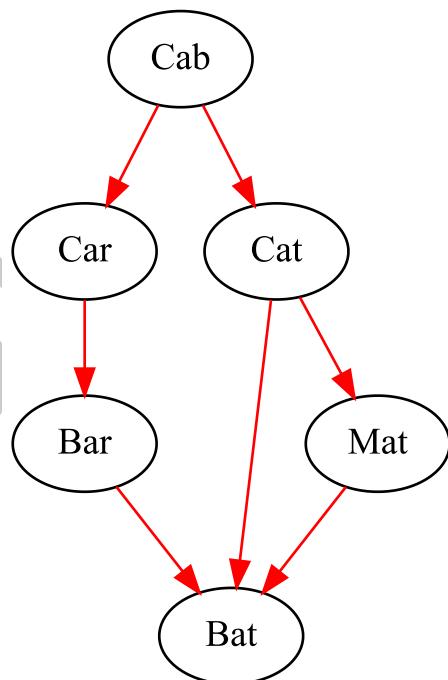
Directed Acyclic Graph (DAG)

In [48]:

```
1  ''
2  #File: cat.dot
3  ## Jagadeesh Vasudevamurthy ####
4  digraph g {
5      edge [color=red]
6          Bar -> Bat
7          Cab -> Car
8          Cab -> Cat
9          Car -> Bar
10         Mat -> Bat
11         Cat -> Mat
12         Cat -> Bat
13     }
14 ...
15 Source(readDotFile("cat"))
```

```
C:\Users\jag\OneDrive\vasu\work\py3\objects\py3\py3\dot\cat.dot
## Jagadeesh Vasudevamurthy ####
digraph g {
edge [color=red]
    Bar -> Bat
    Cab -> Car
    Cab -> Cat
    Car -> Bar
    Mat -> Bat
    Cat -> Mat
    Cat -> Bat
}
```

Out[48]:



VASUDEVAMURTHY

18.7 User Data

```
User Data
```

```
class Data:  
    def __init__(self, n: "string"):  
        self._name = n   ### _name is used as key for this project  
        self.age = 100   ## To show you can have anything,  
  
    def __get_key(self) -> "string":  
        return self._name  
  
    def __str__(self) -> "string":  
        return self._name  
  
    def get_real_name(self) -> "string":  
        return self._name  
    def __hash__(self)->'int':  
        k = self._get_key()  
        t = hash(k)  
        return t  
  
    def __lt__(self, other: "Data") -> "bool":  
        if not isinstance(other, type(self)):  
            assert False  
        n1 = self._get_key()  
        n2 = other._get_key()  
        return (n1 < n2)
```

==
!=
>
<=

Figure 18.18: User Data

18.8 Graph Data structure

18.8.1 class GraphInterface

Nameless Data Structure

```

class GraphInterface:
    def __init__(self):
        self._index = 0
        self._dict = {} # Key is item UDT: Value is index (0 to n-1)
        self._list = [] # Given number between 0 to n-1 get Data in O(1) time

    def find(self, d:'Data')->'int':
        ##calls DATA def __hash__(self)->'int':
        ##if you don't write hash
        ## TypeError: unhashable type: 'Data'
        if (d in self._dict):
            n = self._dict.get(d) #Key is Data Value is the 'int' THETA(1)
            assert(n >= 0 and n < self._index)
            return n
        return -1

    def insert(self, d:'Data')->'int':
        n = self.find(d)
        if (n == -1):
            #Not in the dict. Add to dict and to list. Note everything is pointer
            self._dict[d] = self._index #Key is Data Value is the 'int' THETA(1)
            self._list.append(d)
            n = self._index
            self._index = self._index + 1
        return n

    def __getitem__(self, n:'int')->'string':
        assert(n >= 0 and n < self._index)
        return self._list[n].get_real_name()

```

Graph knows only 0 to (n- 1)

**GraphInterface g
realname=g[i]**

Figure 18.19: Interface to user data

18.8.2 class Node

Node Class

```
class Node:  
    def __init__(self, n: "int"):  
        self._num = n  
        self._fanins = {} # dict of fanins of Node  
        self._fanouts = {} # dict of fanouts of Node  
  
    def get_num(self) -> "int":  
    def add_fan_out(self, e: "Edge") -> "None":  
    def add_fan_in(self, e: "Edge") -> "None":  
    def num_fan_outs(self) -> "int":  
    def num_fan_ins(self) -> "int":  
    def fanout_of_a_node_edges(self) -> "list of fanout edges":  
    def fanout_of_a_node(self) -> "list of fanout numbers":  
    def fanin_of_a_node_edges(self) -> "list of fanin edges":  
    def fanin_of_a_node(self) -> "list of fanin Nodes":  
    def node_has_fanout_edge(self, e: "Edge") -> "Edge or None":  
    def get_weight(self, to:"int") ->'float':  
    def node_has_fanout(self, e: "Edge") -> "int or None":  
    def node_has_fanin_edge(self, e: "Edge") -> "Edge or None":  
    def node_has_fanin(self, e: "Edge") -> "Int or None":  
    def node_get_fanout_node_weight(self, other_node: "Node") -> "float":  
    def node_get_fanin_node_weight(self, other_node: "Node") -> "float":
```

Figure 18.20: Node public functions

18.8.3 class Edge

Edge Class

```
class Edge:  
    def __init__(self, n: "int", weight: "float"):  
        self._other_num = n # _other node  
        self._weight = weight # _weight is float  
  
    #####  
    # All Public routines. YOU SHOULD ONLY CALL THESE ROUTINES  
    #####  
    def get_num(self) -> "int":  
        return self._other_num  
  
    def get_weight(self) -> "float":  
        return self._weight  
  
    def change_weight(self, w: "float") -> "None":  
        self._weight = w
```

Figure 18.21: Edge public functions

18.8.4 class Graph

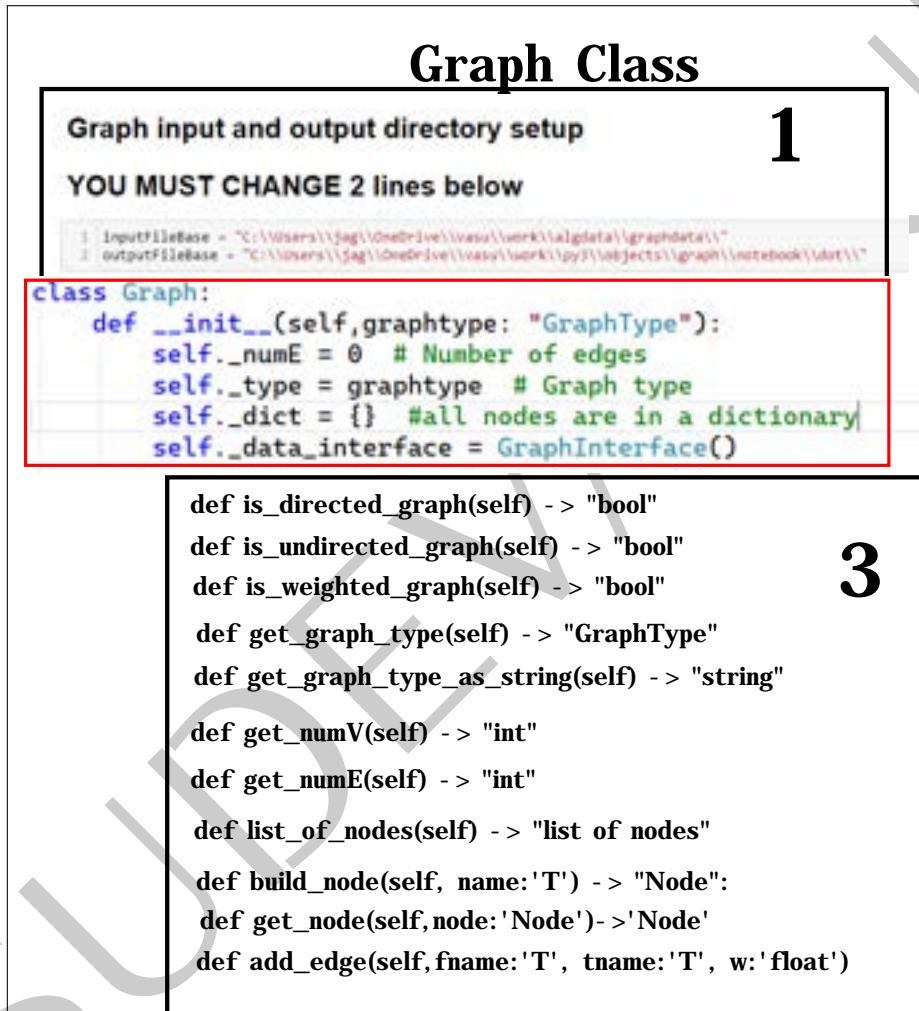


Figure 18.22: Graph public functions

18.9 Dump a graph as a text file

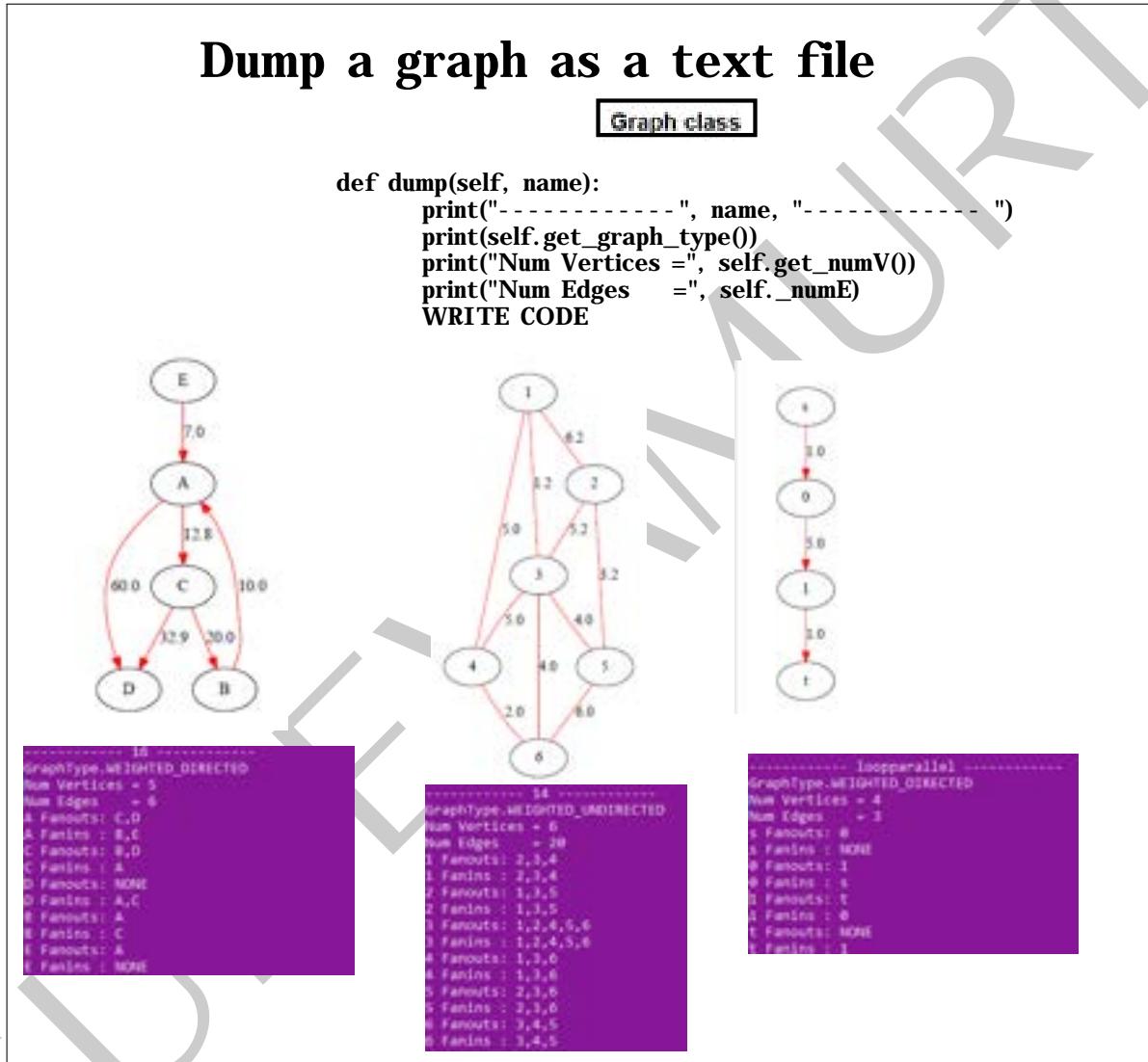


Figure 18.23: Dump a graph as a text file

18.10 Build a graph from a file

Read a file and build a graph

```
class GraphBuilder:
    def __init__(self, g: "graph", f: "string"):
        self._g = g
        # graph object
        self._f = f # File from which you are building graph
        self._g._g = self._build_graph()

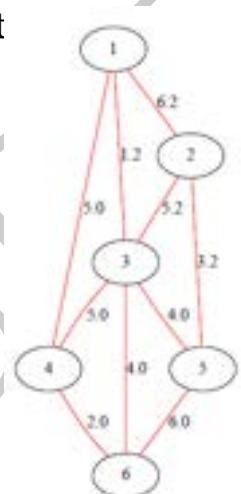
    def _build_graph(self) -> "None":
        WRITE CODE
```

```
class Graph:
    def build_graph(self, f: "file name"):
        b = GraphBuilder(self, f)
```

(undirected)

14.txt

```
1 2 6.2
1 3 1.2
1 4 5
2 3 5.2
2 5 3.2
3 4 5
3 5 4
3 6 4
4 6 2
5 6 6
```



(directed)

13.txt

```
Bar Bat
Cab Car
Mat Bat
Cab Cat
Cat Mat
Car Bar
Cat Bat
```

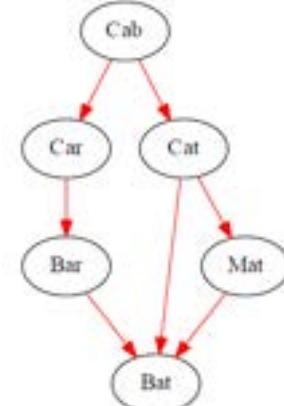


Figure 18.24: class GraphBuilder

18.11 Write a graph as a dot file

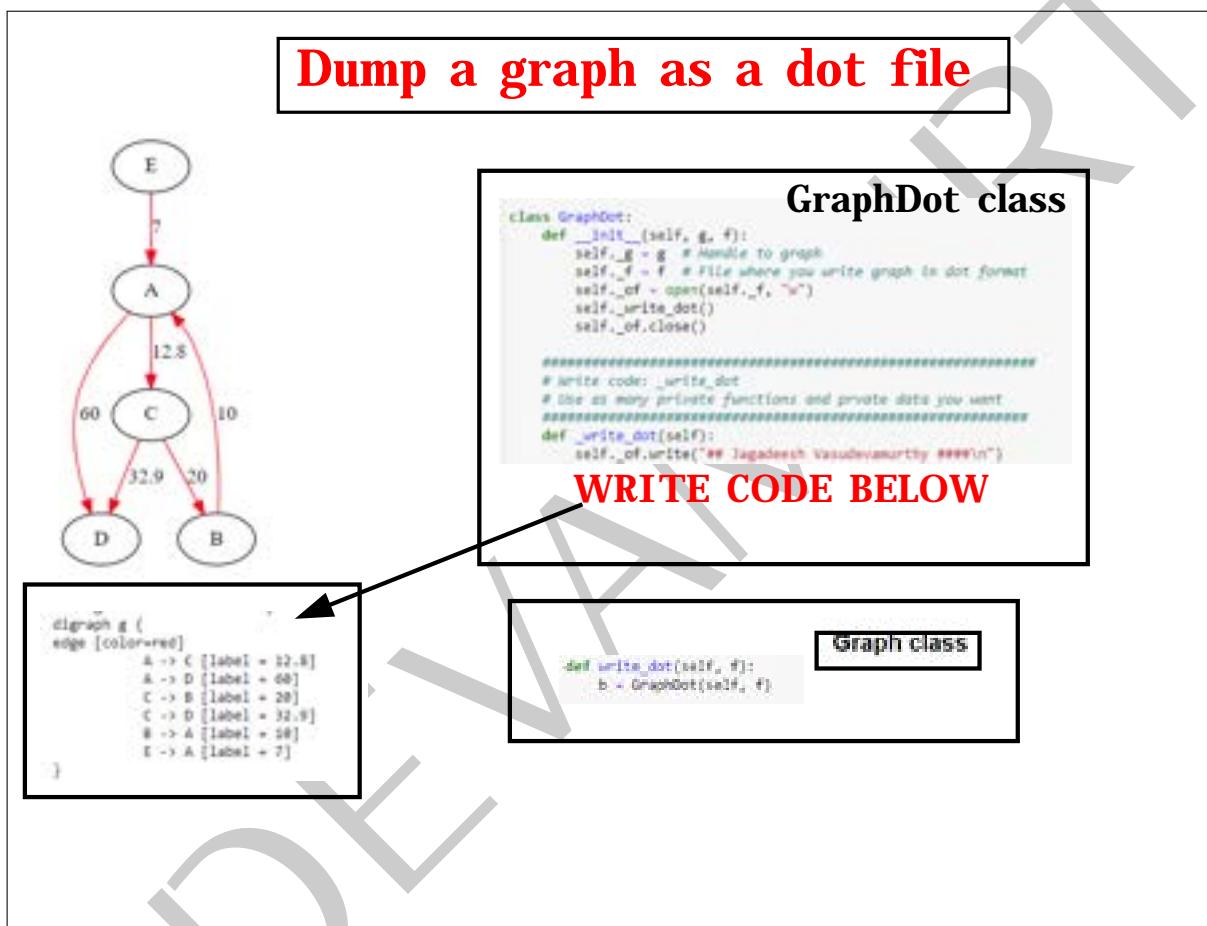


Figure 18.25: class GraphDot

18.11.1 Various dot file examples

18.11. WRITE A GRAPH AS A DOT FILE

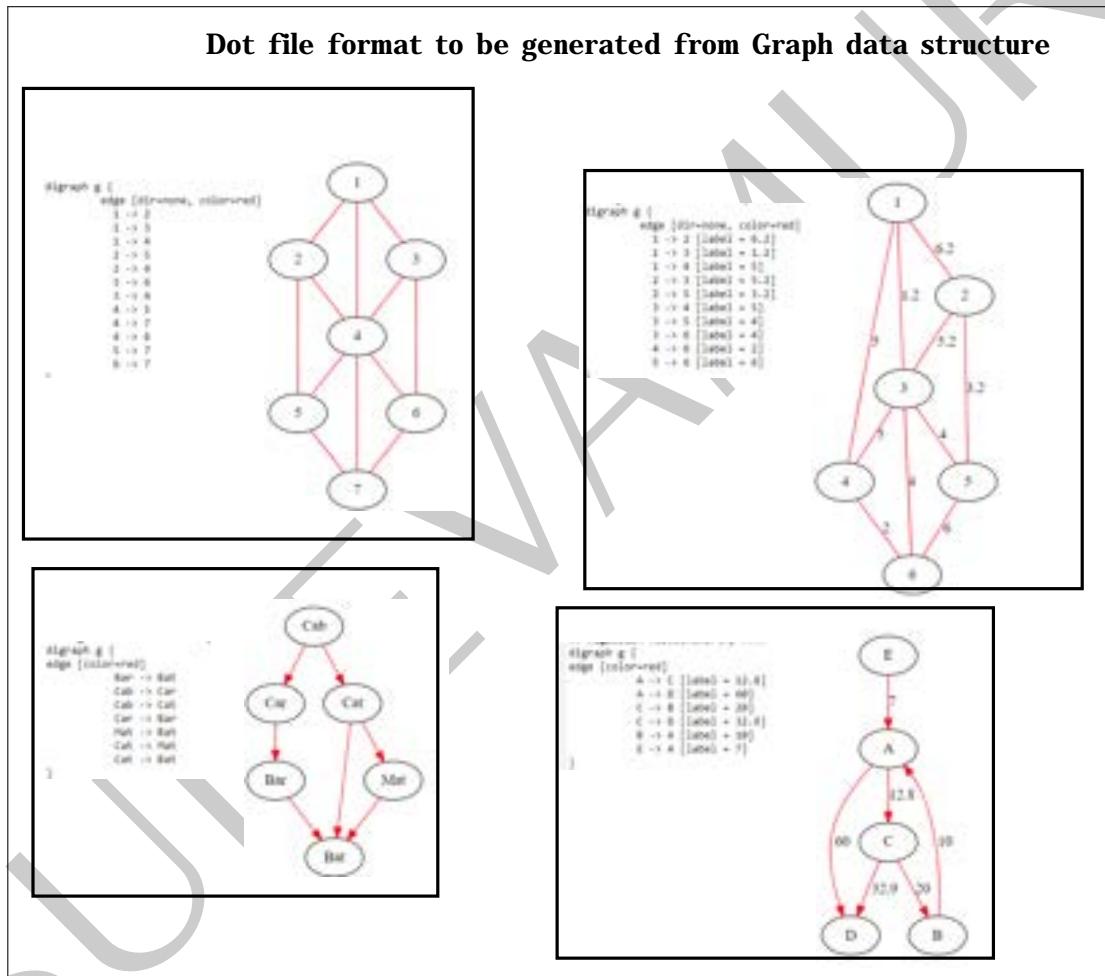


Figure 18.26: Various dot file examples

18.12 Loops in a graph

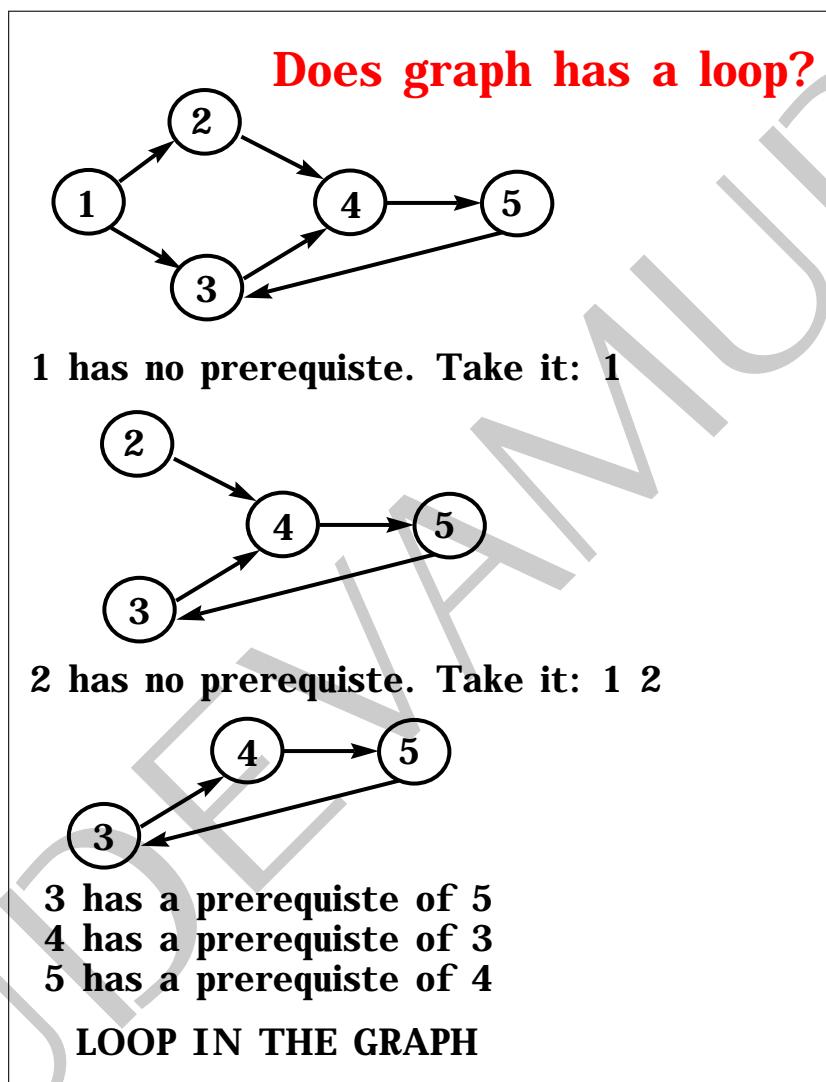


Figure 18.27: Completing courses in an university

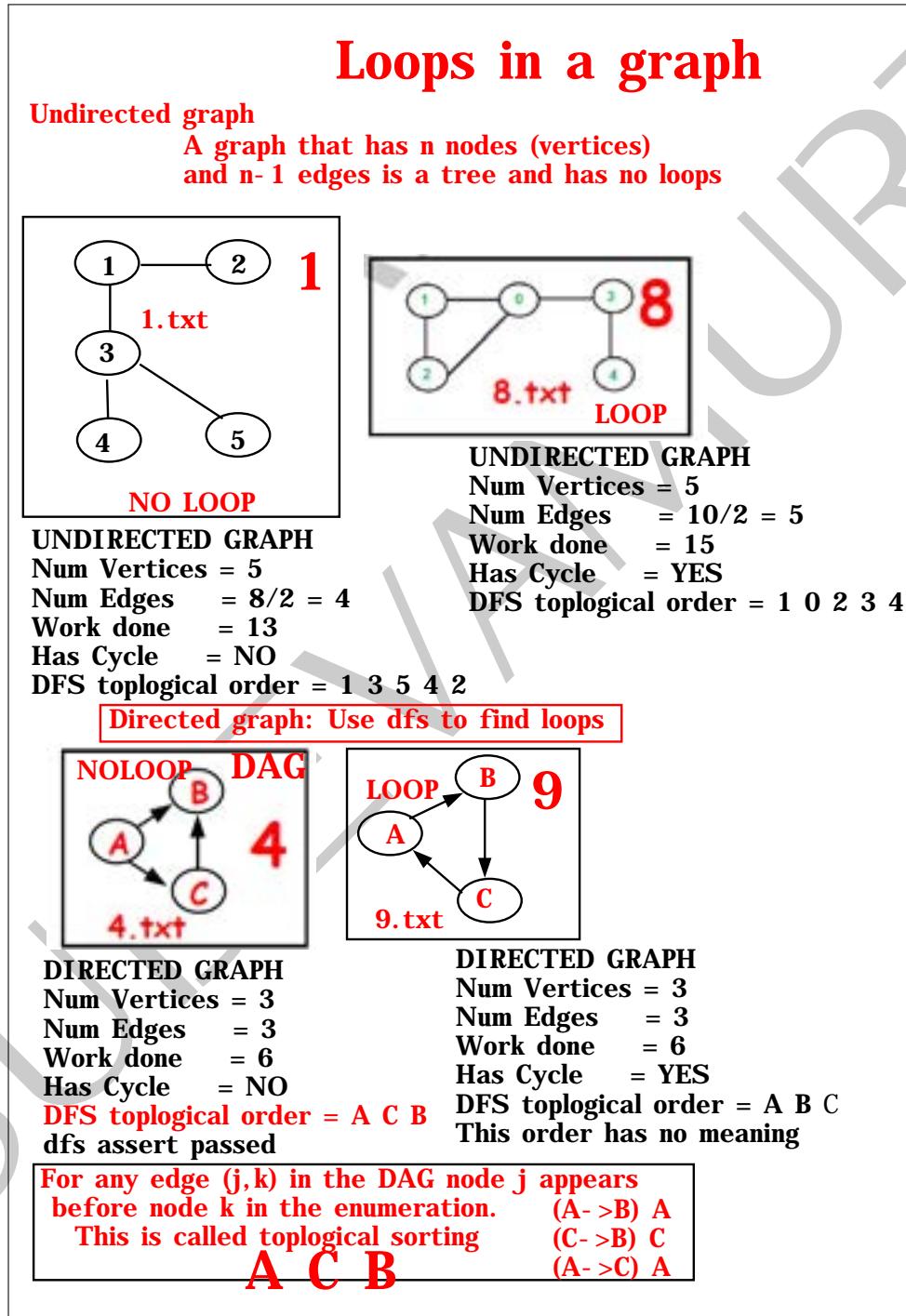


Figure 18.28: Loop in a graph

18.13 Depth first search using time stamps

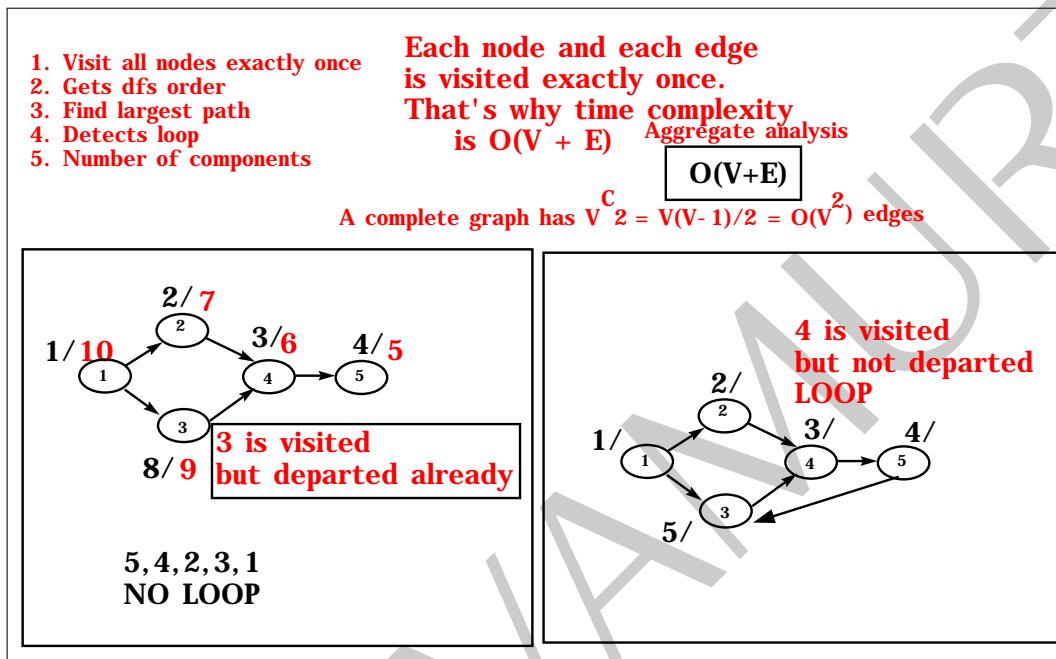


Figure 18.29: Depth first search on a directed graph using time stamps

18.13. DEPTH FIRST SEARCH USING TIME STAMPS

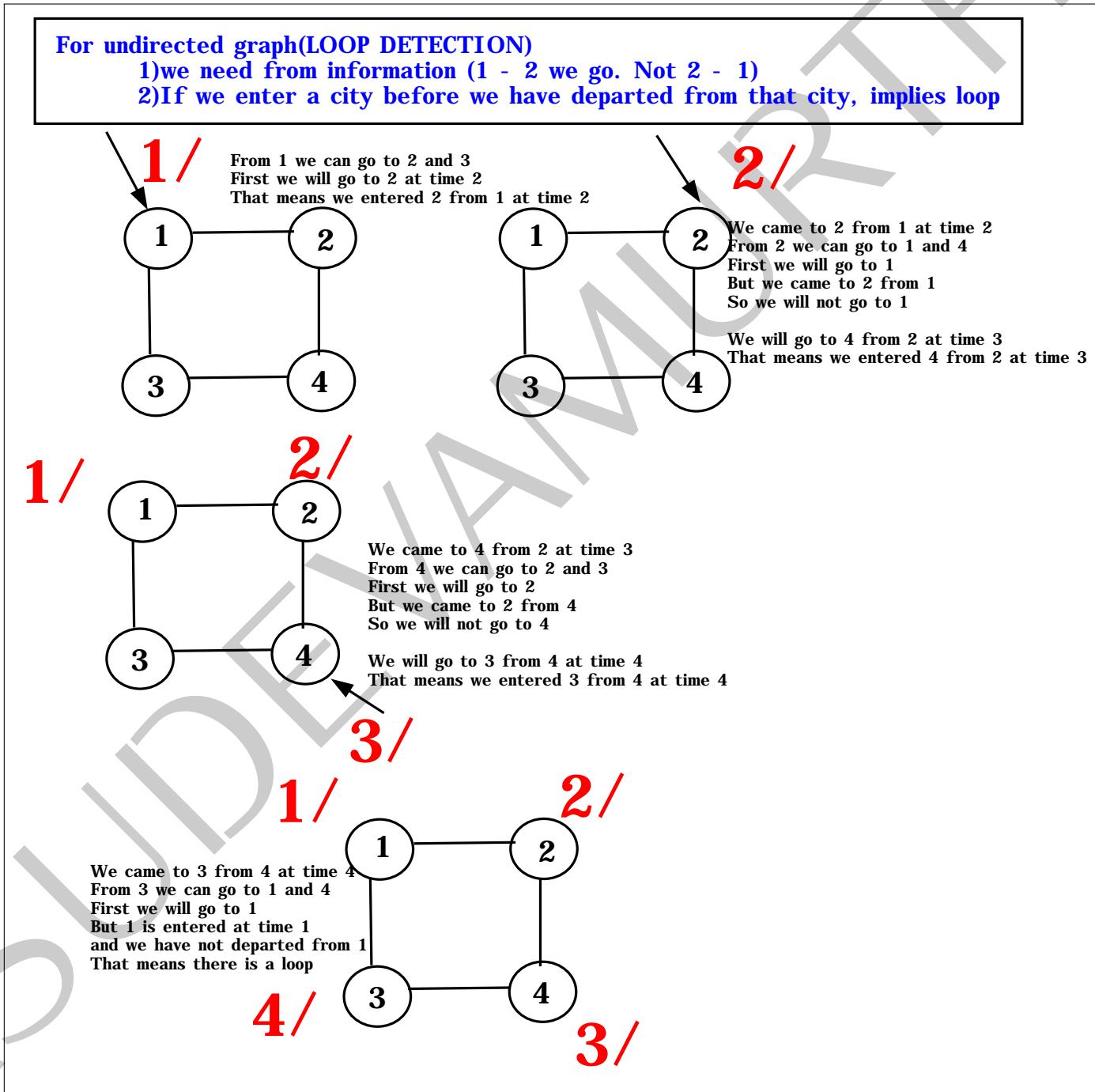


Figure 18.30: Depth first search on an undirected graph using time stamps and from

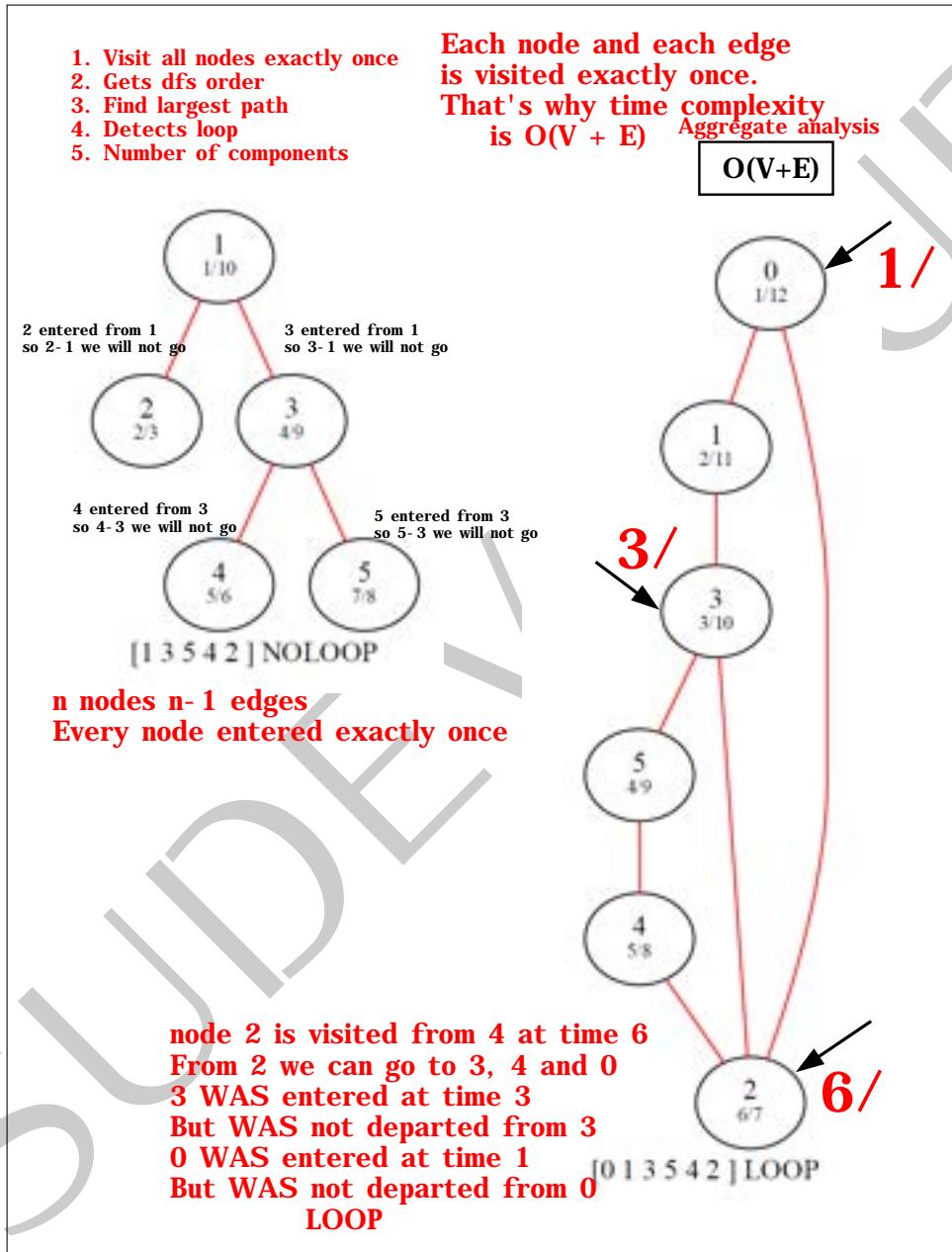


Figure 18.31: Depth first search on an undirected graph

18.13. DEPTH FIRST SEARCH USING TIME STAMPS

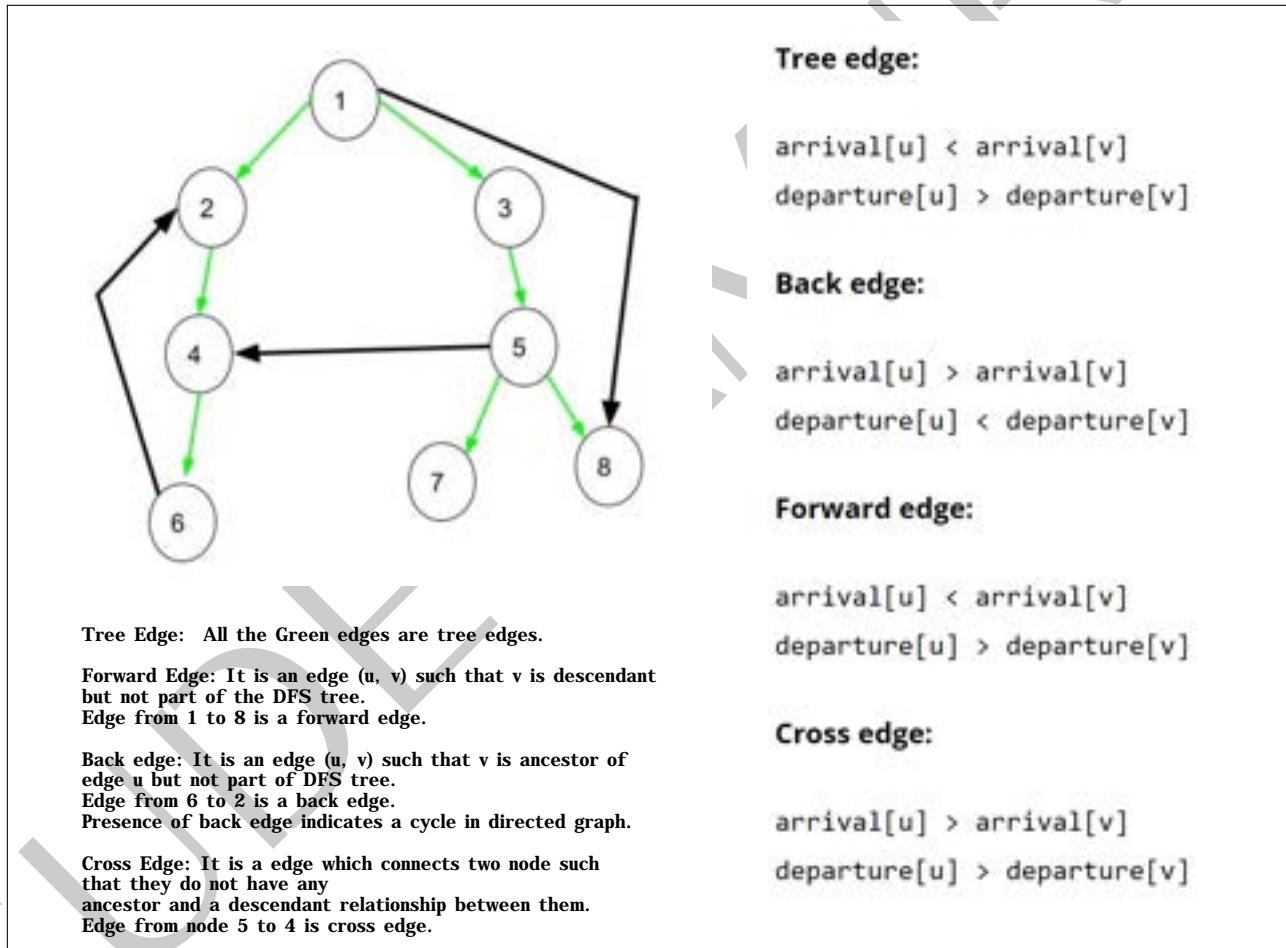


Figure 18.32: classification of edges in a directed graph

18.13.1 Depth first search on a undirected graph with no loop

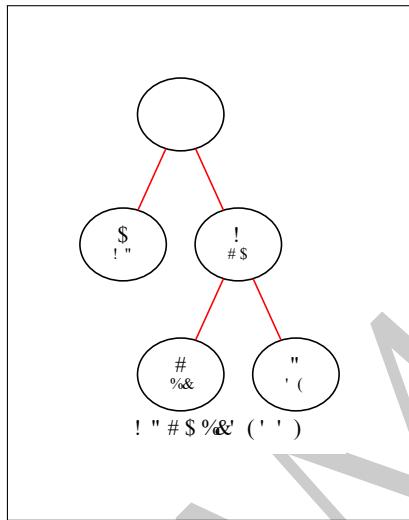


Figure 18.33: undirected graph with no loop

18.13. DEPTH FIRST SEARCH USING TIME STAMPS

18.13.2 Depth first search on a undirected graph with loop

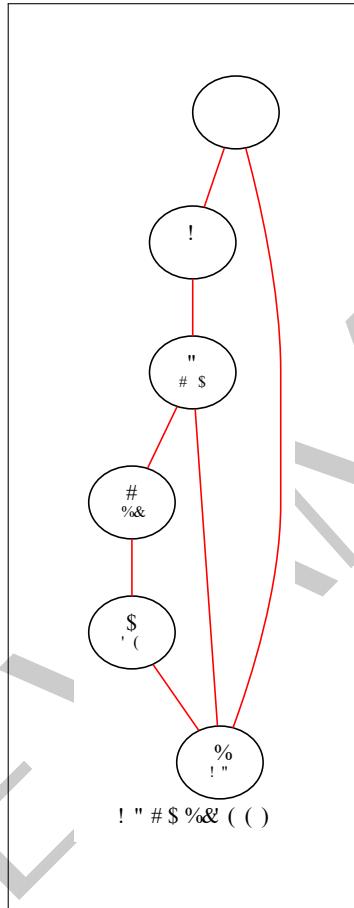


Figure 18.34: undirected graph with loop

```
## Jagadeesh Vasudevamurthy ####  
## dot -Tpdf C:\scratch\outputs\dot\udf1dfs.dot -o C:\scratch\outputs\dot\udf1dfs.d  
digraph g {  
    label = "[0 1 3 5 4 2 ] LOOP"  
    0[label = <0<BR /><FONT POINT-SIZE="10">1/12</FONT>>]  
    1[label = <1<BR /><FONT POINT-SIZE="10">2/11</FONT>>]  
    3[label = <3<BR /><FONT POINT-SIZE="10">3/10</FONT>>]  
    5[label = <5<BR /><FONT POINT-SIZE="10">4/9</FONT>>]  
    4[label = <4<BR /><FONT POINT-SIZE="10">5/8</FONT>>]  
    2[label = <2<BR /><FONT POINT-SIZE="10">6/7</FONT>>]  
    edge [dir=none, color=red]  
    0 -> 1  
    0 -> 2  
    1 -> 3  
    3 -> 5  
    3 -> 2  
    5 -> 4  
    4 -> 2  
}
```

18.13. DEPTH FIRST SEARCH USING TIME STAMPS

18.13.3 Depth first search on a directed graph with no loop

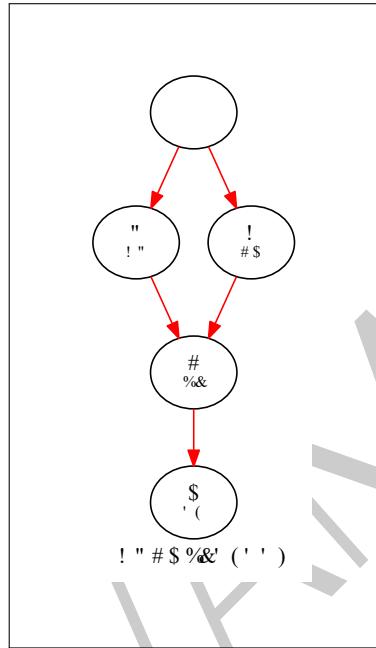


Figure 18.35: directed graph with no loop

```
## Jagadeesh Vasudevamurthy ####  
## dot -Tpdf C:\scratch\outputs\dot\2dfs.dot -o C:\scratch\outputs\dot\2dfs.dot.pdf  
digraph g {  
    label = "[1 3 2 4 5 ] NOLOOP"  
    1[label = <1<BR /><FONT POINT-SIZE="10">1/10</FONT>>]  
    2[label = <2<BR /><FONT POINT-SIZE="10">2/7</FONT>>]  
    3[label = <3<BR /><FONT POINT-SIZE="10">8/9</FONT>>]  
    4[label = <4<BR /><FONT POINT-SIZE="10">3/6</FONT>>]  
    5[label = <5<BR /><FONT POINT-SIZE="10">4/5</FONT>>]  
    edge [color=red]  
        1 -> 2  
        1 -> 3  
        2 -> 4  
        3 -> 4  
        4 -> 5  
}
```

18.13. DEPTH FIRST SEARCH USING TIME STAMPS

18.13.4 Depth first search on a directed graph with loop

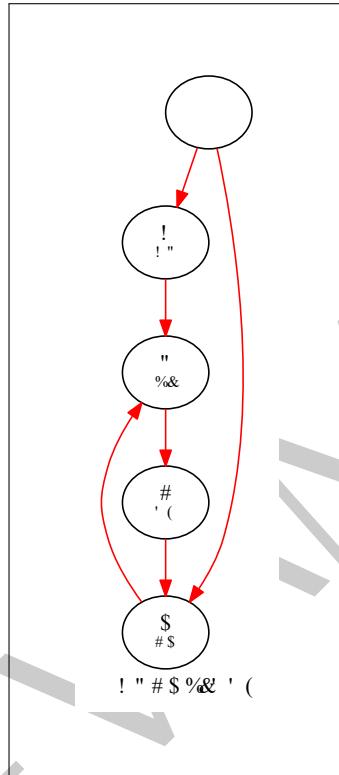


Figure 18.36: directed graph with loop

```
## Jagadeesh Vasudevamurthy ####  
## dot -Tpdf C:\scratch\outputs\dot\3dfs.dot -o C:\scratch\outputs\dot\3dfs.dot.pdf  
digraph g {  
    label = "[0 1 3 4 2 ] LOOP"  
    0[label = <0<br /><font point-size="10">1/10</font>>]  
    1[label = <1<br /><font point-size="10">2/9</font>>]  
    2[label = <2<br /><font point-size="10">5/6</font>>]  
    3[label = <3<br /><font point-size="10">3/8</font>>]  
    4[label = <4<br /><font point-size="10">4/7</font>>]  
edge [color=red]  
    0 -> 1  
    0 -> 2  
    1 -> 3  
    2 -> 3  
    3 -> 4  
    4 -> 2  
}
```

18.13. DEPTH FIRST SEARCH USING TIME STAMPS

18.13.5 Depth first search on a directed graph with no loop

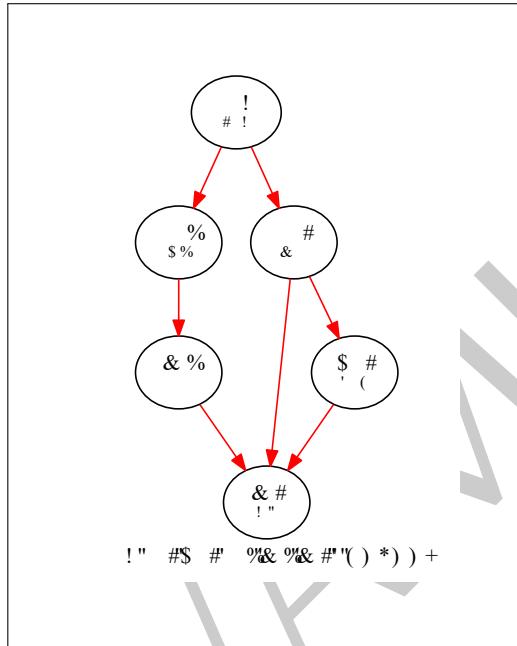


Figure 18.37: directed graph with no loop

```
## Jagadeesh Vasudevamurthy ####  
## dot -Tpdf C:\scratch\outputs\dot\catdfs.dot -o C:\scratch\outputs\dot\catdfs.dot  
digraph g {  
    label = "[Cab Cat Mat Car Bar Bat ] NOLOOP"  
    Bar[label = <Bar<BR /><FONT POINT-SIZE="10">1/4</FONT>>]  
    Bat[label = <Bat<BR /><FONT POINT-SIZE="10">2/3</FONT>>]  
    Cab[label = <Cab<BR /><FONT POINT-SIZE="10">5/12</FONT>>]  
    Car[label = <Car<BR /><FONT POINT-SIZE="10">6/7</FONT>>]  
    Mat[label = <Mat<BR /><FONT POINT-SIZE="10">9/10</FONT>>]  
    Cat[label = <Cat<BR /><FONT POINT-SIZE="10">8/11</FONT>>]  
    edge [color=red]  
    Bar -> Bat  
    Cab -> Car  
    Cab -> Cat  
    Car -> Bar  
    Mat -> Bat  
    Cat -> Bat  
    Cat -> Mat  
}
```

18.13. DEPTH FIRST SEARCH USING TIME STAMPS

18.13.6 Depth first search on a directed graph with no loop

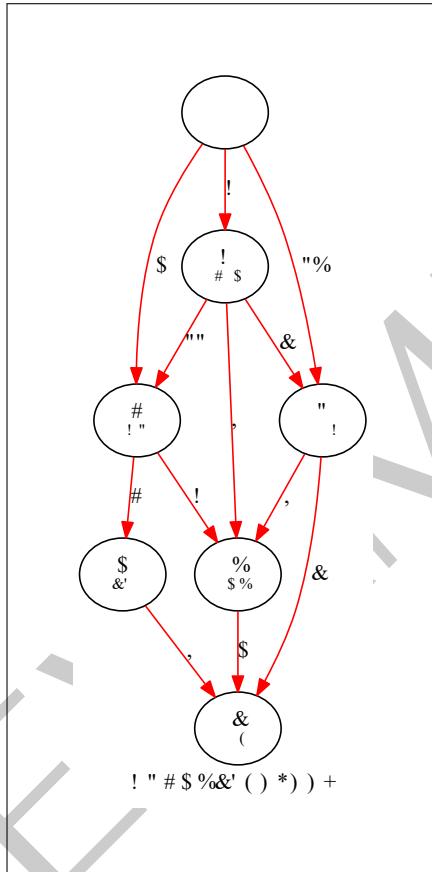


Figure 18.38: directed graph with no loop

```
## Jagadeesh Vasudevamurthy ####  
## dot -Tpdf C:\scratch\outputs\dot\7dfs.dot -o C:\scratch\outputs\dot\7dfs.dot.pdf  
digraph g {  
    label = "[0 3 1 2 5 4 6 ] NOLLOOP"  
    0[label = <0<BR /><FONT POINT-SIZE="10">1/14</FONT>>]  
    2[label = <2<BR /><FONT POINT-SIZE="10">2/9</FONT>>]  
    3[label = <3<BR /><FONT POINT-SIZE="10">10/13</FONT>>]  
    1[label = <1<BR /><FONT POINT-SIZE="10">11/12</FONT>>]  
    6[label = <6<BR /><FONT POINT-SIZE="10">4/5</FONT>>]  
    4[label = <4<BR /><FONT POINT-SIZE="10">3/6</FONT>>]  
    5[label = <5<BR /><FONT POINT-SIZE="10">7/8</FONT>>]  
edge [color=red]  
    0 -> 2 [label = 5]  
    0 -> 3 [label = 3]  
    0 -> 1 [label = 14]  
    2 -> 4 [label = 3]  
    2 -> 5 [label = 2]  
    3 -> 2 [label = 11]  
    3 -> 1 [label = 6]  
    3 -> 4 [label = 7]  
    1 -> 6 [label = 6]  
    1 -> 4 [label = 7]  
    4 -> 6 [label = 5]  
    5 -> 6 [label = 7]  
}
```

18.14. DEPTH FIRST SEARCH USING COLORS

18.13.7 Implementing DFS using time stamps

Implementing DFS using Timestamp

```
class GraphDfsUsingTimeStamp():
    def __init__(self,g:'graph',filename:'string',dfs_order:'list',
                 has_loop:'list of size 1',work:'list of size 1',dfs_traversal_output_file):
        ##NOTHING CAN BE CHANGED HERE
        self._g = g
        self._f = filename
        self._dfs_order = dfs_order
        self._has_loop = has_loop
        self._has_loop[0] = False
        self._work = work
        self._work[0] = 0
        self._dfs_traversal_output_file = dfs_traversal_output_file

        ##YOU CAN have any number of private variables and functions
        self._dfs()
        self._write_dot()

    def _write_dot(self):
        print("WRITE CODE")

    def _dfs(self):
        print("WRITE CODE")
```

```
class Graph():
    def dfs_using_time_stamp(self, f:'string',dfs_order:'list',has_loop:'list of size 1',
                           work:'list of size 1',dfs_traversal_output_file:'string'):
        b = GraphDfsUsingTimeStamp(self,f,dfs_order,has_loop,work,dfs_traversal_output_file)
```

DFS ORDER: 1 3 2 4 5
NOLOOP
Work Done 6
DFS ASSERT PASSED

Figure 18.39: Implementing DFS using time stamps

18.14 Depth first search using colors

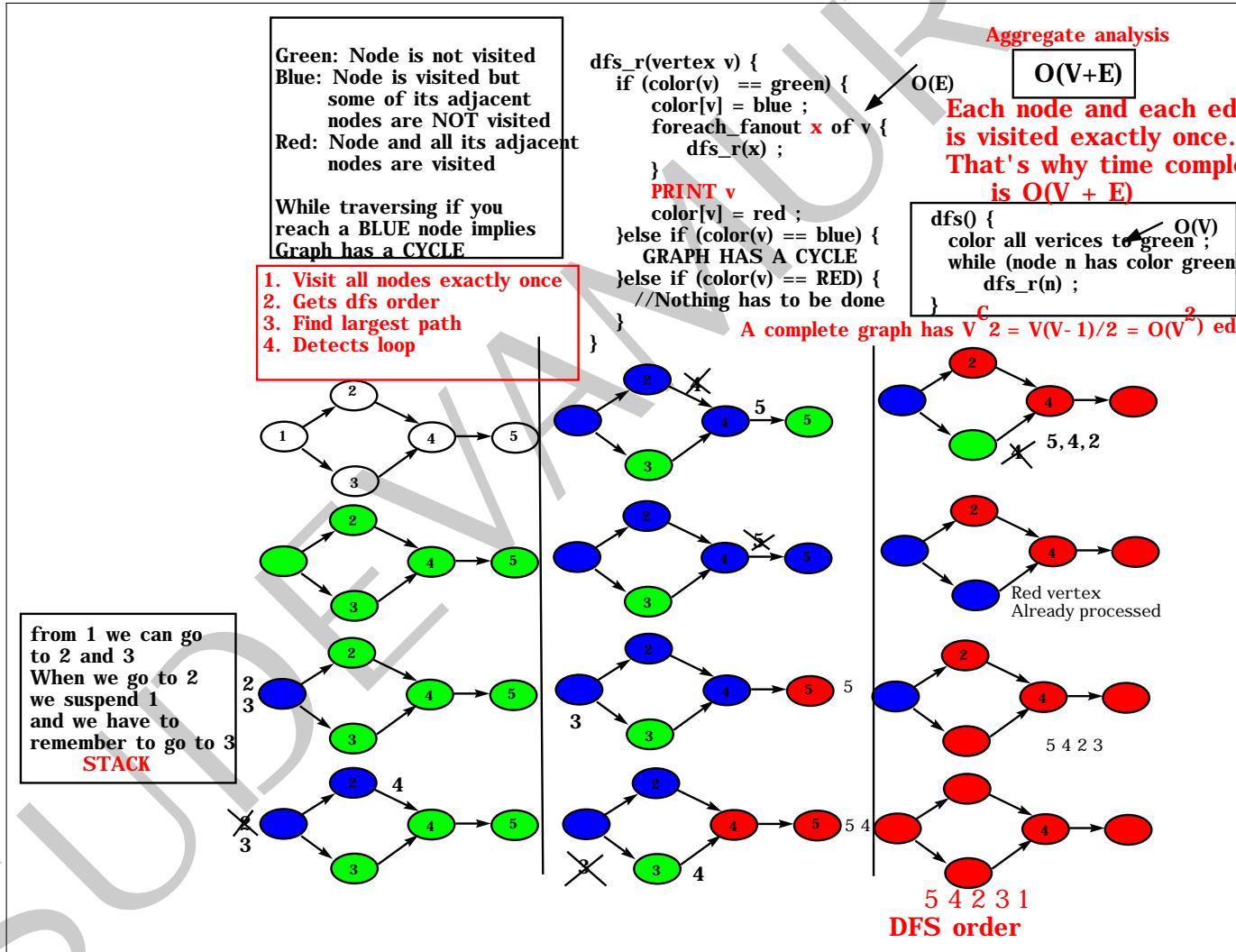


Figure 18.40: Depth first search on a graph that has no loop

18.14. DEPTH FIRST SEARCH USING COLORS

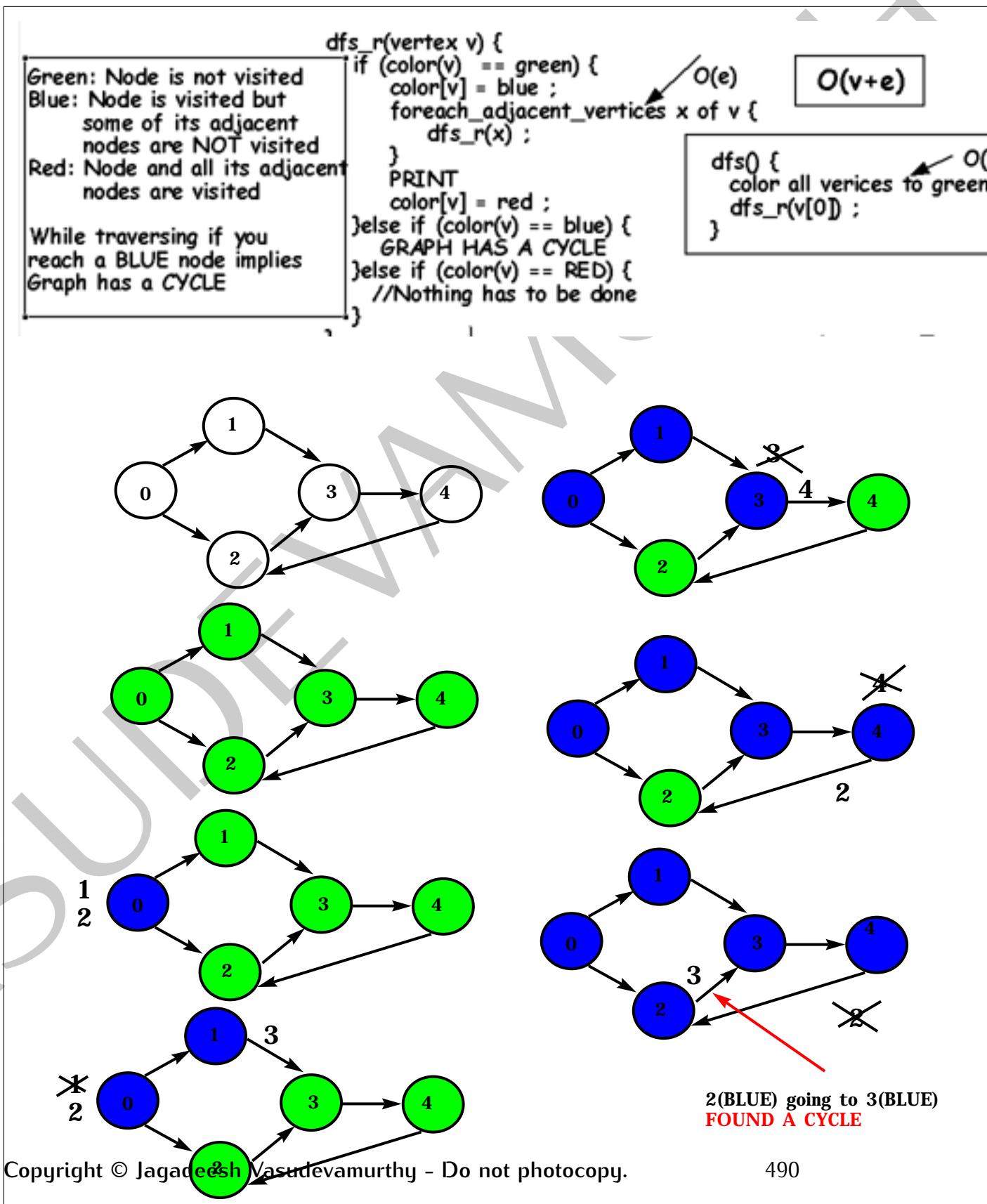


Figure 18.41: Depth first search on a graph that has a loop

18.15 Breadth first search

Breadth First Search

1. Explore the graph level by level
2. All nodes are visited exactly once
3. For every node n, we need to remember two things:

a) Is n has visited already?

We can track this: visited[n]

b) Have we visited all the adjacent nodes of n

We can track this with a queue

Insert all the fanout of a node to a queue

For_each_fanout f of n

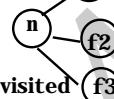
if (visited[f] == FALSE)

visited[f] = true;

q.add(f); //f fanout has to be visited

}

We need to process
f1, f2 and f3
QUEUE



```
bfs(node s) {
    for each node of G assign {level, from} = -1; O(V)
    s.level = 0 ;
    s.from = 0 ;
    q.enqueue(s) ;
    while (q is not empty) {
        node n = q.dequeue ;
        for_each_fanout_node(f of n) {  

            if (f.level == -1) {
                f.level = n.level + 1 ;
                f.from = n ;
                q.enqueue(f)
            }
        }
    }
}
```

O(V+2E)

This loop depends on fanout of each node
But max vertices is $E = nC_2$
But total fanout for G cannot be bigger than
1) E for directed graph
2) 2E for undirected graph

1. Finds shortest path if graph has same weight
2. We can find the path from s to t from 'from'
3. We know the distance from s to any node from 'level'
4. We can find all connected components
5. Each node is visited exactly once

Figure 18.42: Breadth first search

18.15. BREADTH FIRST SEARCH

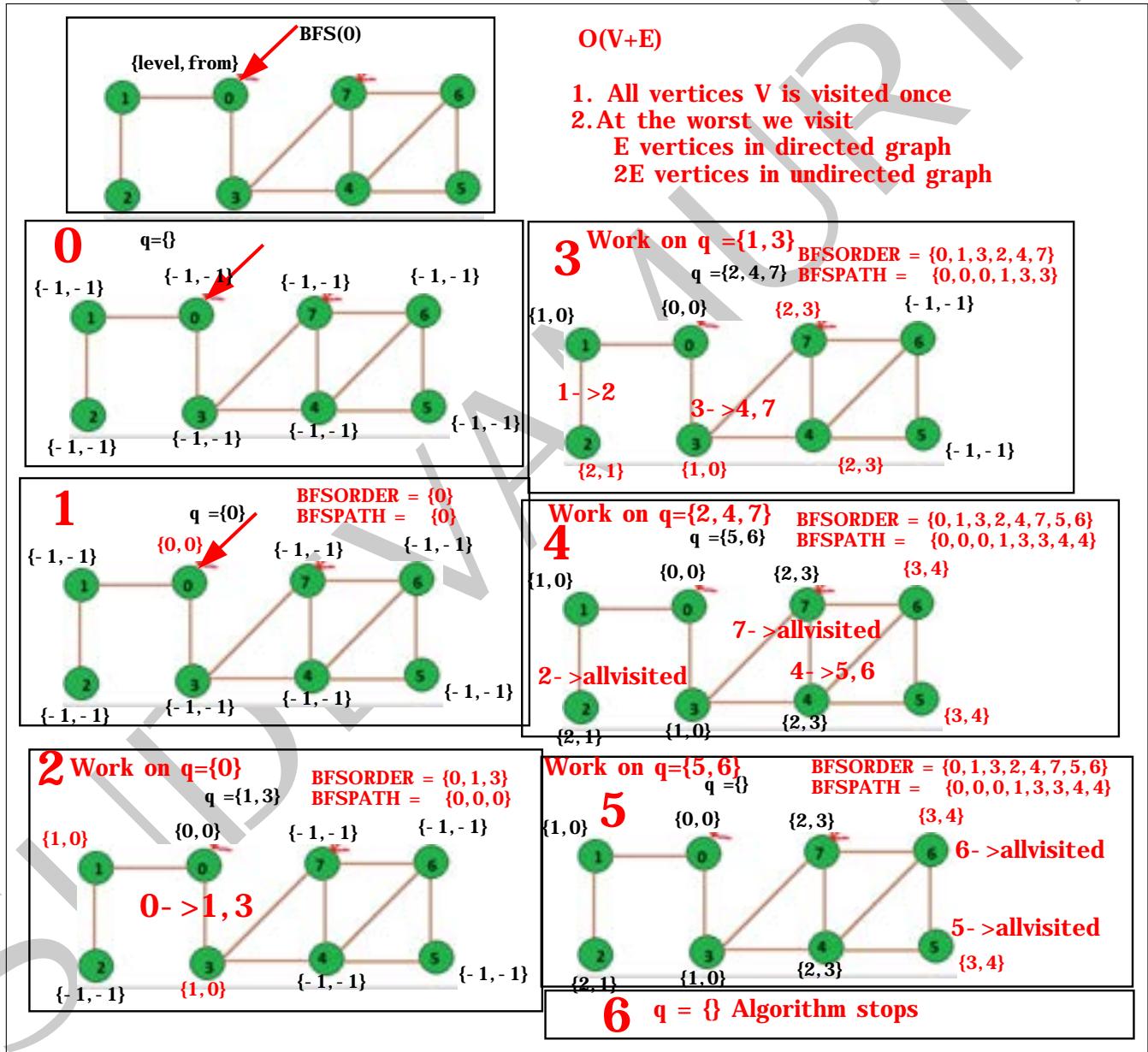


Figure 18.43: Breadth first search animation

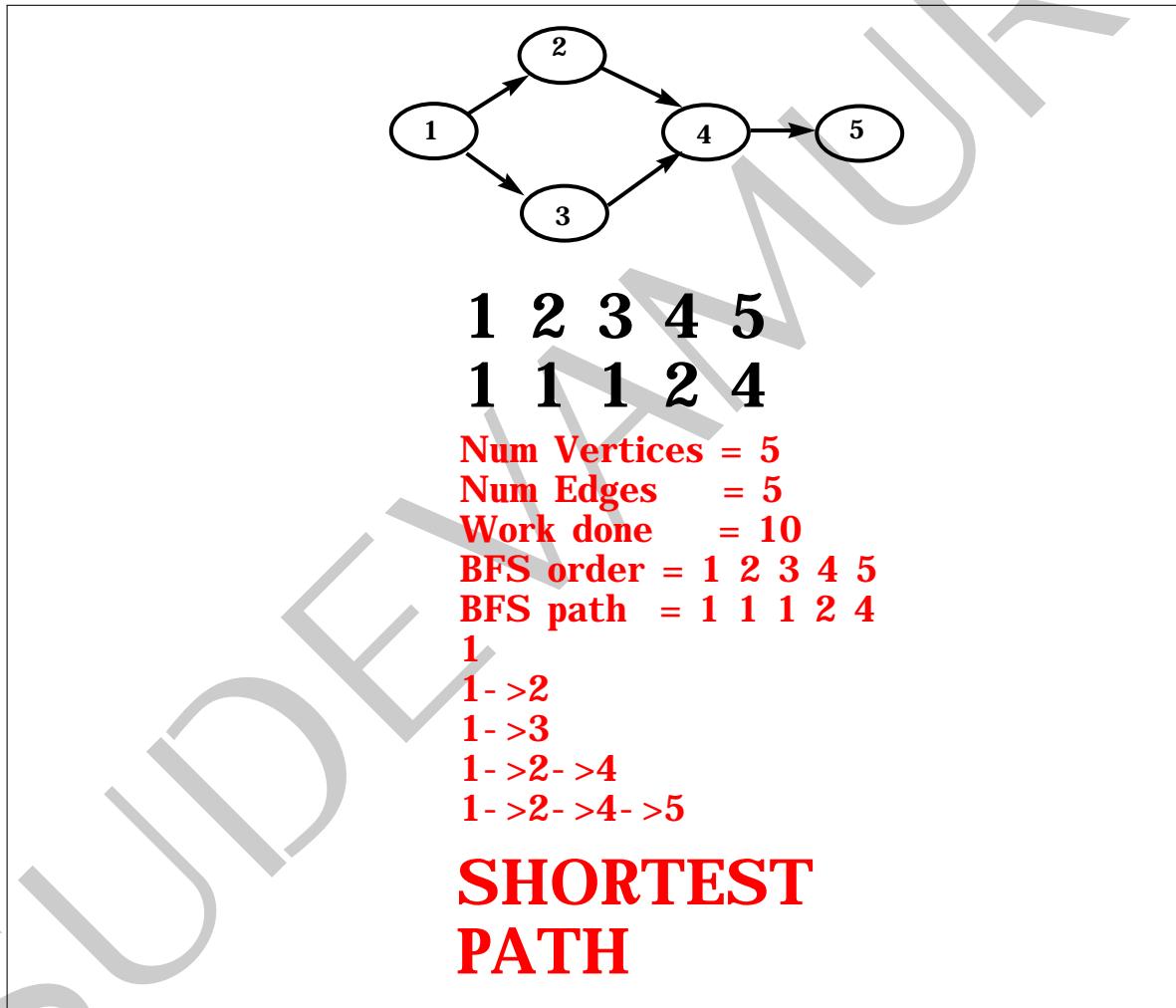


Figure 18.44: Breadth first search on a graph that has no loop

18.15. BREADTH FIRST SEARCH

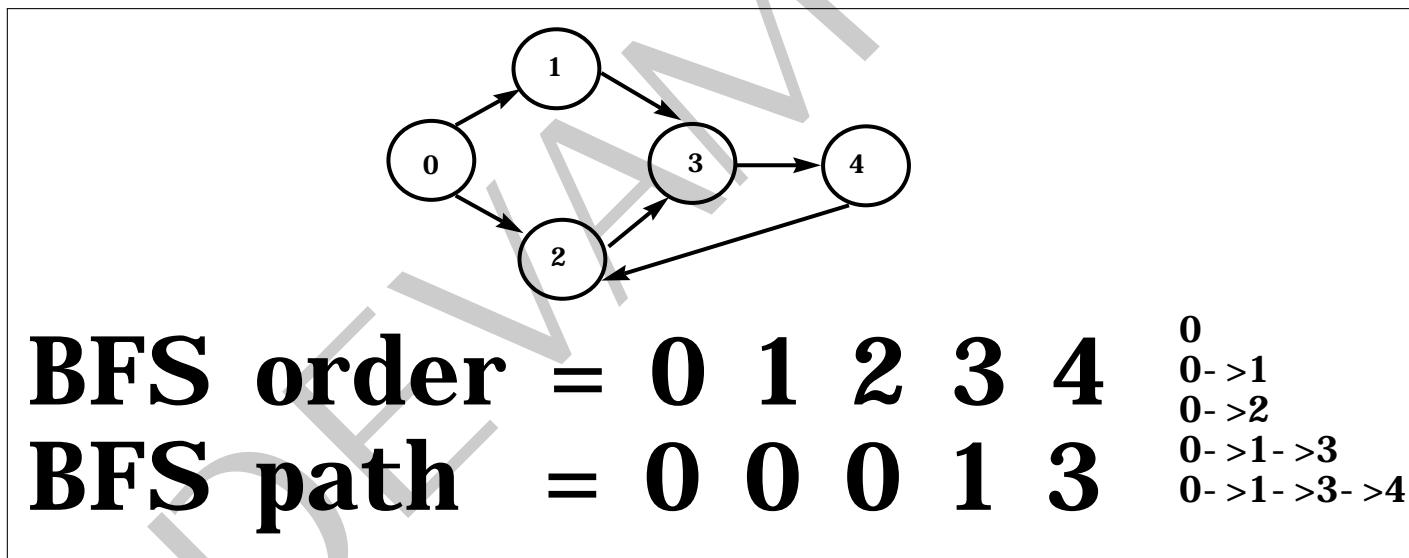


Figure 18.45: Breadth first search on a graph that has a loop

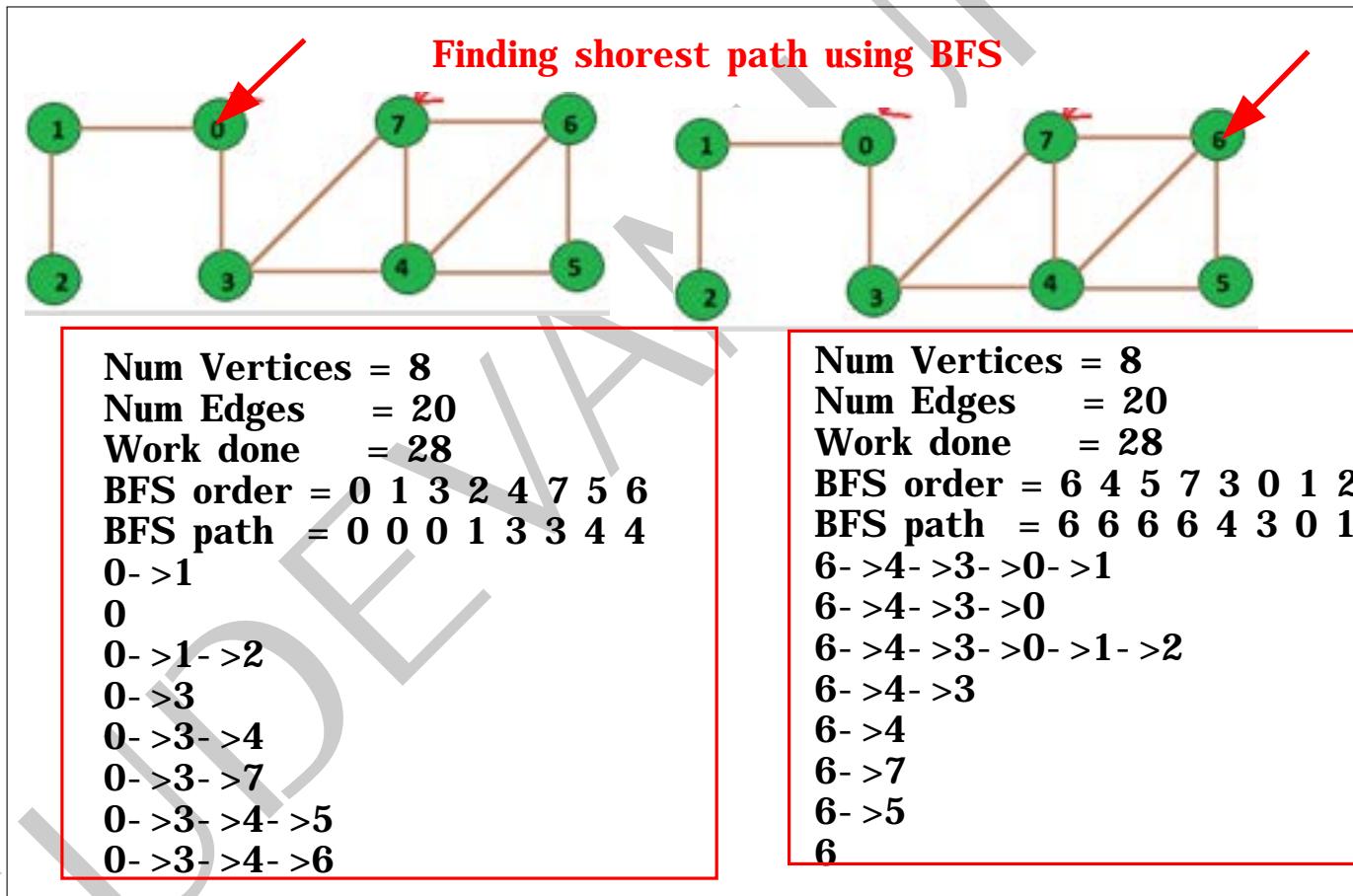


Figure 18.46: Breadth first search on a graph

18.15. BREADTH FIRST SEARCH

18.15.1 Implementing BFS

Implementing BFS algorithm

```
class GraphBfs:  
    def __init__(GraphBfs.py  
        self,  
        g: "graph",  
        graph_name: "string",  
        start_city_name: "string",  
        end_city:"string"  
        shortest_dist:'list of size 1',  
        work: "list of size 1",  
        bfs_traversal_output_file:"String",  
        show: 'bool'  
    ):  
        ## must write 3 routines  
        self._bfs()  
        self._find_shortest_path()  
        self._write_dot()  
  
def bfs(Graph.py  
        self,  
        gname: "graphname",  
        start: "start city name",  
        endt: "end city name",  
        dist: "list of size 1",  
        work: "list of size 1",  
        ofile: "bfs_traversal_output_file",  
        show: "bool",  
    ):  
        b = GraphBfs(self, gname, start, endt, dist, work, ofile, show)
```

Figure 18.47: Implementing BFS

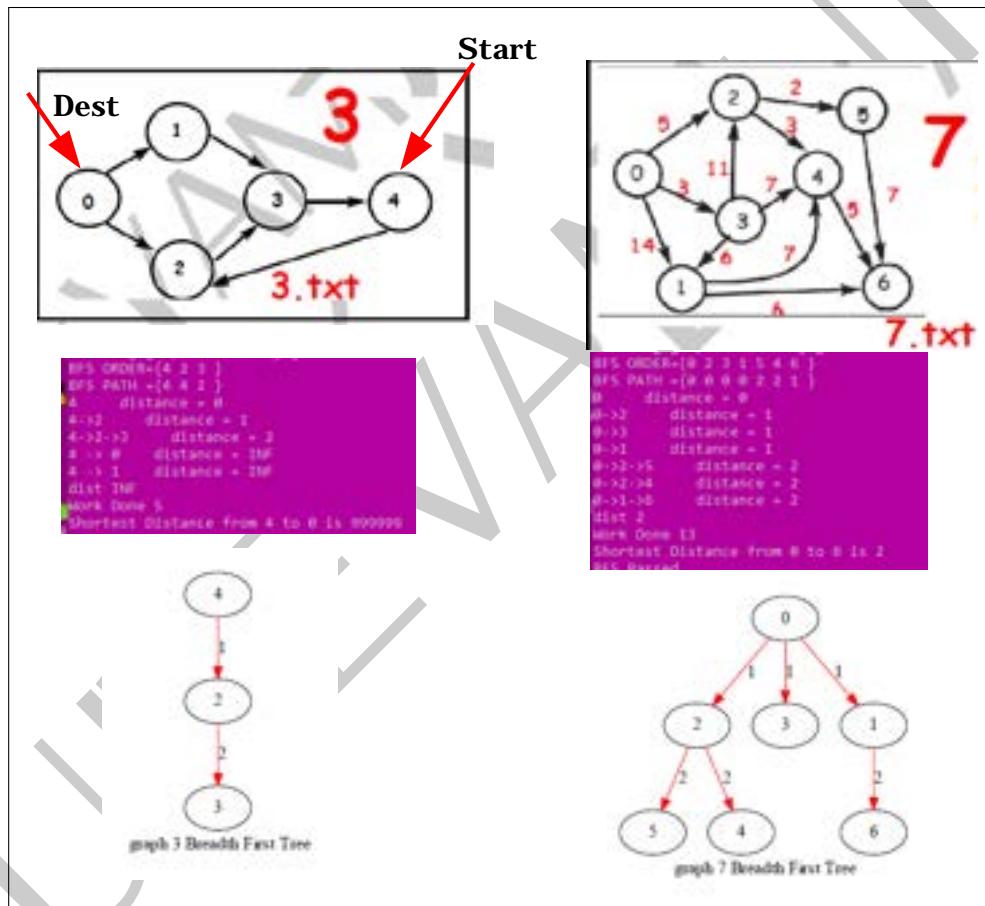


Figure 18.48: Implementing BFS

18.16. APPLICATION OF BREADTH FIRST SEARCH

18.16 Application of Breadth first search

18.16.1 Snake and Ladder

Refer to the section 20.18

18.17 Single source shortest path algorithms on a non negative weighted directed graph

18.17.1 Dijkstra's algorithm

A Note on Two Problems in Connexion with Graphs

By

E. W. DIJKSTRA

We consider n points (nodes), some or all pairs of which are connected by a branch; the length of each branch is given. We restrict ourselves to the case where at least one path exists between any two nodes. We now consider two problems.

Problem 1. Construct the tree of minimum total length between the n nodes. (A tree is a graph with one and only one path between every two nodes.)

In the course of the construction that we present here, the branches are subdivided into three sets:

I. the branches definitely assigned to the tree under construction (they will form a subtree);

II. the branches from which the next branch to be added to set I, will be selected;

III. the remaining branches (rejected or not yet considered).

The nodes are subdivided into two sets:

A. the nodes connected by the branches of set I,

B. the remaining nodes (one and only one branch of set II will lead to each of these nodes).

We start the construction by choosing an arbitrary node as the only member of set A, and by placing all branches that end in this node in set II. To start with, set I is empty. From then onwards we perform the following two steps repeatedly.

Step 1. The shortest branch of set II is removed from this set and added to set I. As a result one node is transferred from set B to set A.

Step 2. Consider the branches leading from the node, that has just been transferred to set A, to the nodes that are still in set B. If the branch under consideration is longer than the corresponding branch in set II, it is rejected; if it is shorter, it replaces the corresponding branch in set II, and the latter is rejected.

We then return to step 1 and repeat the process until sets II and B are empty. The branches in set I form the tree required.

The solution given here is to be preferred to the solution given by J. B. KRUSKAL [1] and those given by H. LOBERMAN and A. WEINBERGER [2]. In their solutions all the — possibly $\frac{1}{2}n(n-1)$ — branches are first of all sorted according to length. Even if the length of the branches is a computable function of the node coordinates, their methods demand that data for all branches are stored simultaneously. Our method only requires the simultaneous storing of

the data for at most n branches, viz. the branches in sets I and II and the branch under consideration in step 2.

Problem 2. Find the path of minimum total length between two given nodes P and Q .

We use the fact that, if R is a node on the minimal path from P to Q , knowledge of the latter implies the knowledge of the minimal path from P to R . In the solution presented, the minimal paths from P to the other nodes are constructed in order of increasing length until Q is reached.

In the course of the solution the nodes are subdivided into three sets:

A. the nodes for which the path of minimum length from P is known; nodes will be added to this set in order of increasing minimum path length from node P ;

B. the nodes from which the next node to be added to set A will be selected; this set comprises all those nodes that are connected to at least one node of set A but do not yet belong to A themselves;

C. the remaining nodes.

The branches are also subdivided into three sets:

I. the branches occurring in the minimal paths from node P to the nodes in set A;

II. the branches from which the next branch to be placed in set I will be selected; one and only one branch of this set will lead to each node in set B;

III. the remaining branches (rejected or not yet considered).

To start with, all nodes are in set C and all branches are in set III. We now transfer node P to set A and from then onwards repeatedly perform the following steps.

Step 1. Consider all branches r connecting the node just transferred to set A with nodes R in sets B or C. If node R belongs to set B, we investigate whether the use of branch r gives rise to a shorter path from P to R than the known path that uses the corresponding branch in set II. If this is not so, branch r is rejected; if, however, use of branch r results in a shorter connexion between P and R than hitherto obtained, it replaces the corresponding branch in set II and the latter is rejected. If the node R belongs to set C, it is added to set B and branch r is added to set II.

Step 2. Every node in set B can be connected to node P in only one way if we restrict ourselves to branches from set I and one from set II. In this sense each node in set B has a distance from node P : the node with minimum distance from P is transferred from set B to set A, and the corresponding branch is transferred from set II to set I. We then return to step 1 and repeat the process until node Q is transferred to set A. Then the solution has been found.

Remark 1. The above process can also be applied in the case where the length of a branch depends on the direction in which it is traversed.

Remark 2. For each branch in sets I and II it is advisable to record its two nodes (in order of increasing distance from P), and the distance between P and that node of the branch that is furthest from P . For the branches of set I this

is the actual minimum distance, for the branches of set II it is only the minimum thus far obtained.

The solution given above is to be preferred to the solution by L. R. FORD [3] as described by C. BERGE [4], for, irrespective of the number of branches, we need not store the data for all branches simultaneously but only those for the branches in sets I and II, and this number is always less than n . Furthermore, the amount of work to be done seems to be considerably less.

References

- [1] KRUSKAL jr., J. B.: On the Shortest Spanning Subtree of a Graph and the Travelling Salesman Problem. Proc. Amer. Math. Soc. 7, 48–50 (1956).
- [2] LOBERMAN, H., and A. WEINBERGER: Formal Procedures for Connecting Terminals with a Minimum Total Wire Length. J. Ass. Comp. Mach. 4, 428–437 (1957).
- [3] FORD, L. R.: Network flow theory. Rand Corp. Paper, P-923, 1956.
- [4] BERGE, C.: Théorie des graphes et ses applications, pp. 68–69. Paris: Dunod 1958.

Mathematisch Centrum
2e Boerhaavestraat 49
Amsterdam-O

(Received June 11, 1959)

18.17. SINGLE SOURCE SHORTEST PATH ALGORITHMS ON A NON NEGATIVE WEIGHTED DIRECTED GRAPH

Edsger Wybe Dijkstra



Born	May 11, 1930 Rotterdam, Netherlands
Died	August 6, 2002 (aged 72) Nuenen, Netherlands
Fields	Computer science
Institutions	Mathematisch Centrum Eindhoven University of Technology The University of Texas at Austin
Doctoral advisor	Adriaan van Wijngaarden
Doctoral students	Nico Habermann Martin Rem David Naumann Cornelis Hemerik Jan Tijmen Udding Johannes van de Snepscheut Antonetta van Gasteren
Known for	Dijkstra's algorithm Structured programming THE multiprogramming system Semaphore
Notable awards	Turing Award Association for Computing Machinery

Figure 18.49: Dijkstra

Dijkstra's Algorithm

Finds the shortest path from a vertex s to any other vertex w .

Graph G must be directed and all edge weights must be ≥ 0 .

Graph does not need to be a DAG.

Graph G can have cycles

Greedy Algorithm

USES THREE CONCEPTS:

1. BFS
2. Min Heap(Priority Queue) instead of Queue
3. Relaxation

Figure 18.50: Dijkstra's algorithm concepts

18.17.2 Concept of relaxation

18.17. SINGLE SOURCE SHORTEST PATH ALGORITHMS ON A NON NEGATIVE WEIGHTED DIRECTED GRAPH

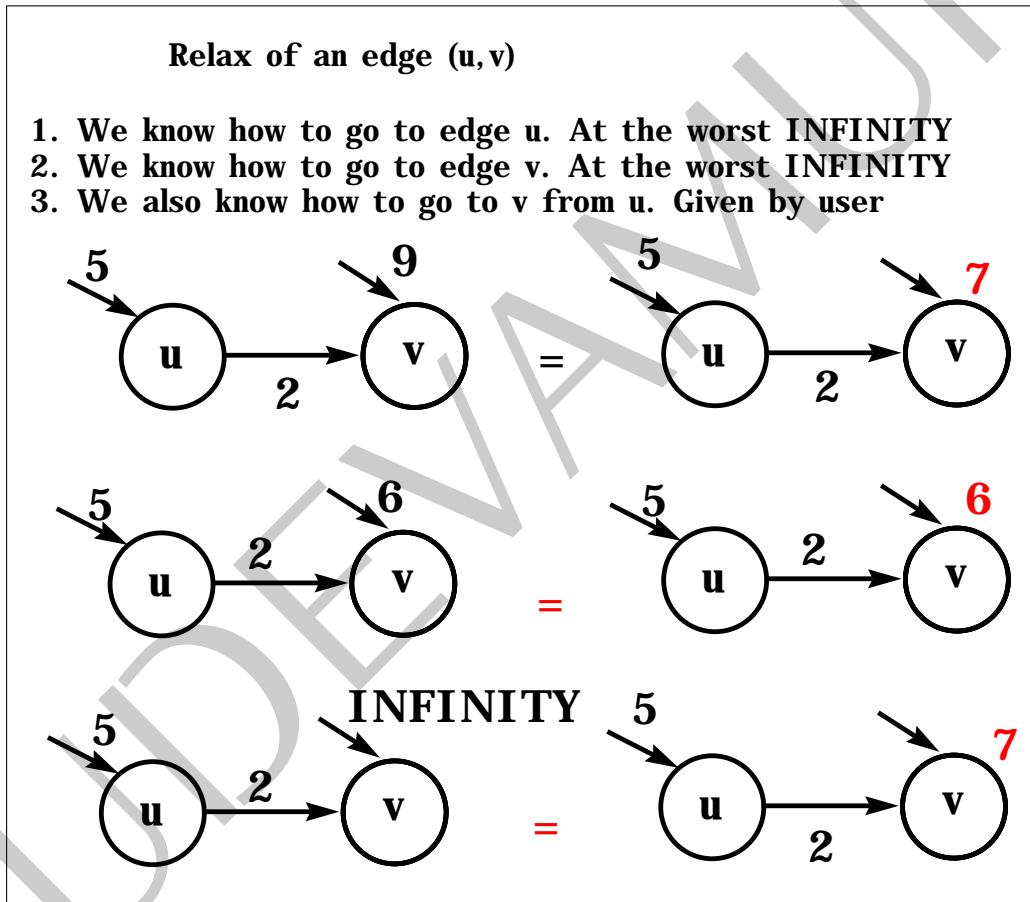
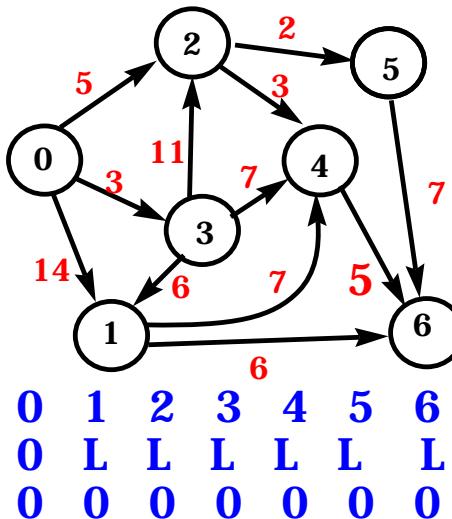


Figure 18.51: Relaxation of an edge

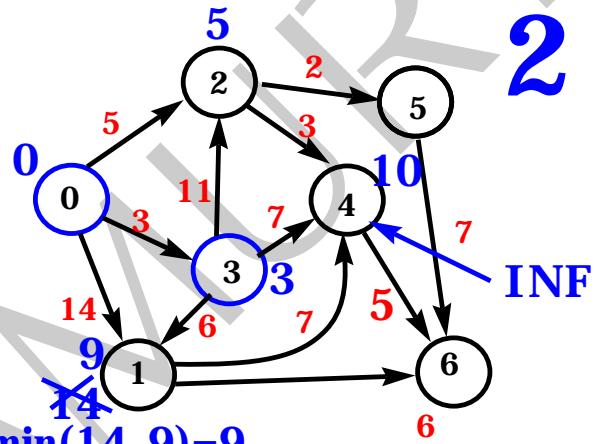
18.17.3 Dijkstra's algorithm in action

18.17. SINGLE SOURCE SHORTEST PATH ALGORITHMS ON A NON NEGATIVE WEIGHTED DIRECTED GRAPH

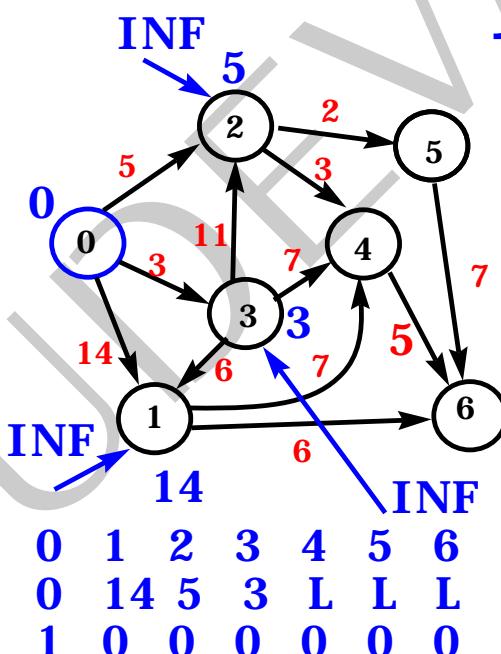
Find the minimum distance between 0 and 6



Work on vertex 3 since it has the min value 3



Work on vertex 0



Work on vertex 2 since it has the min value 5

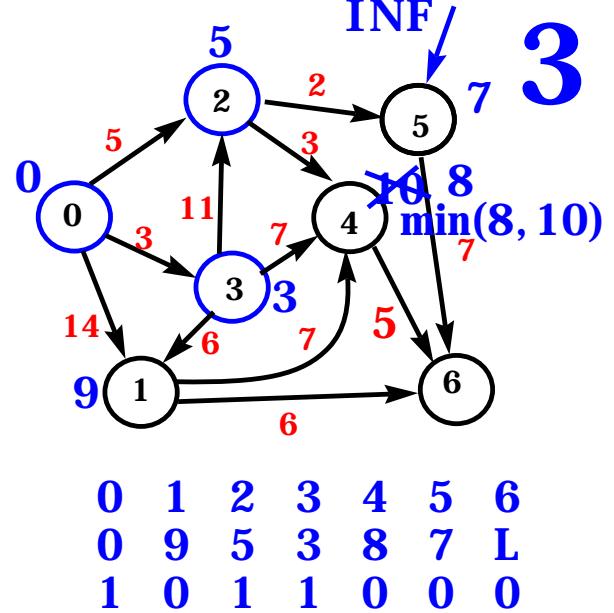


Figure 18.52: Dijkstra's algorithm in action

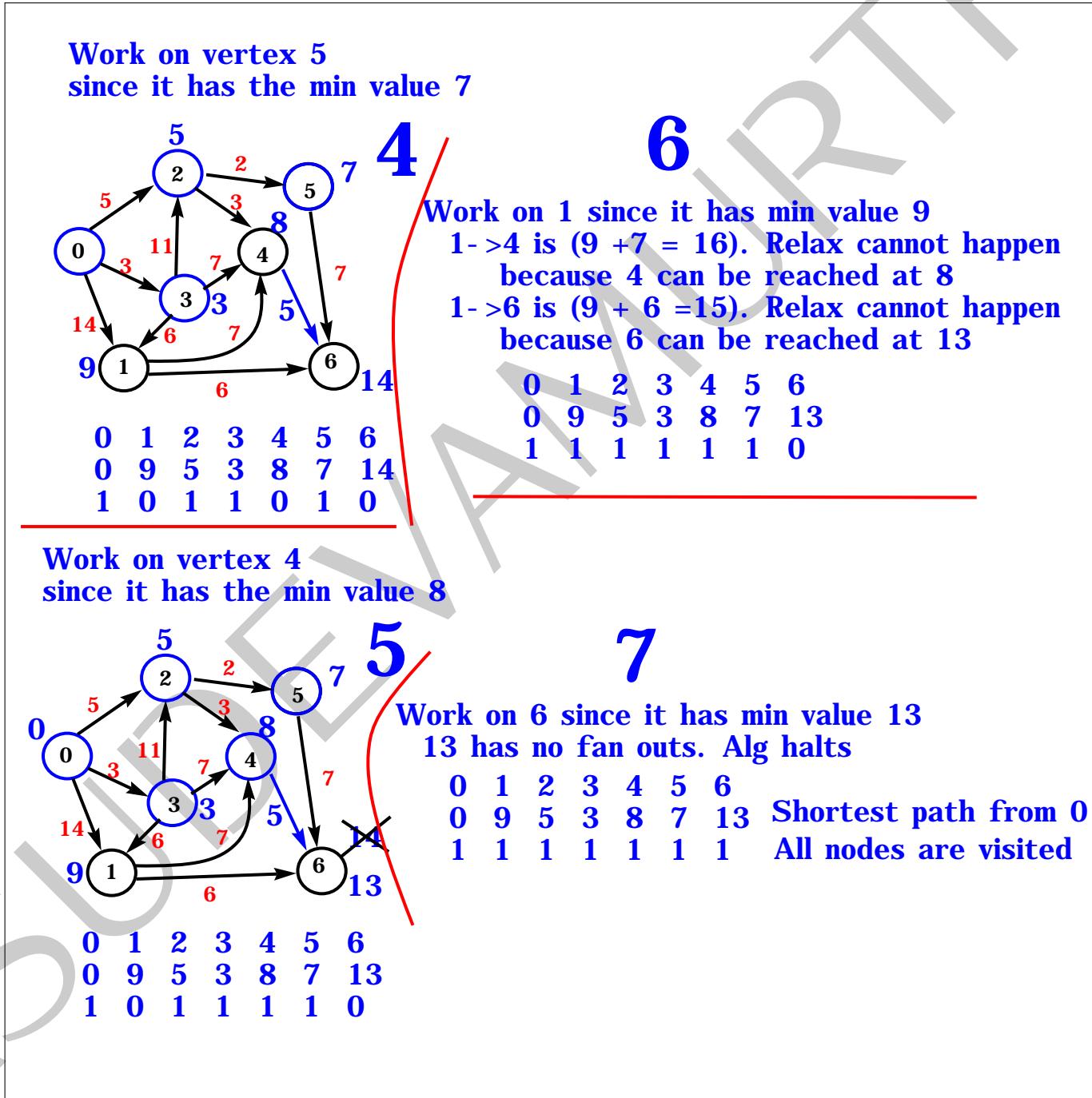
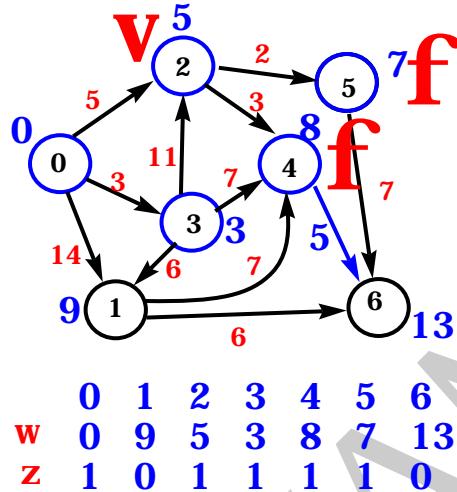


Figure 18.53: Dijkstra's algorithm in action, continued

Complexity of Dijkstra's algorithm



```

n = number_of_vertices
w[n] = weight array initialized to infinite.
w[startingpoint] = 0 ;
z[n] = visited array initialized to false
do { O(V+E)
    v = get_a_vertex_that_has_minimum_weight_and_not_visited
        (w[v] is minimum and z[v] is false)
    if (!v) DONE RETURN
    for_all_fanouts(v, f) {
        relax(v, f, w(v, f)) ;
    }
    z[v] = true ; //v is true
}while(1) O(V*(V+E))

```

O(V) can be reduced **O(logV)** using minheap

O(log V * (V+E))

18.17.4 Implementing Dijkstra's algorithm

18.17. SINGLE SOURCE SHORTEST PATH ALGORITHMS ON A NON NEGATIVE WEIGHTED DIRECTED GRAPH

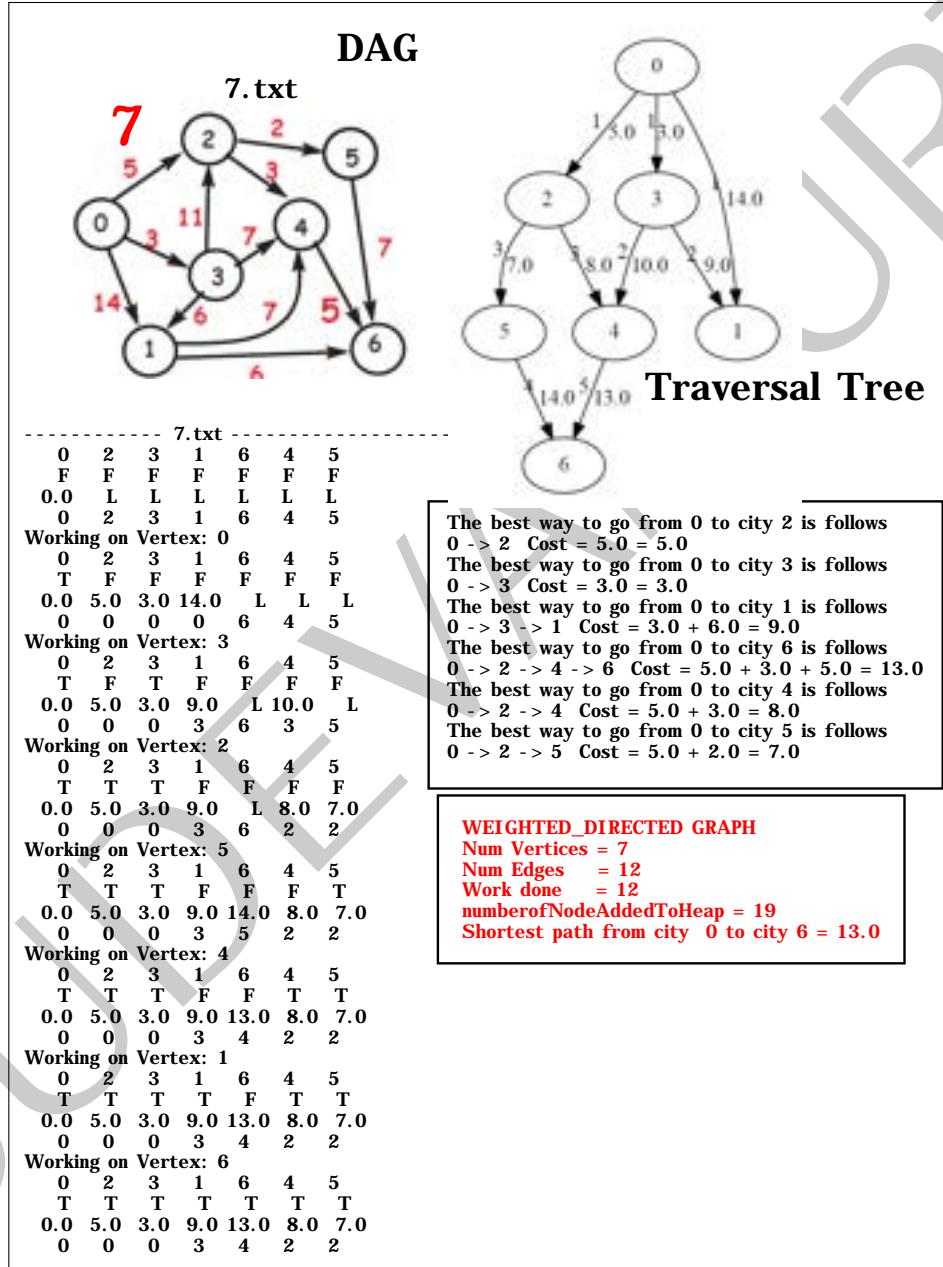


Figure 18.55: Dijkstra on directed weighted graph

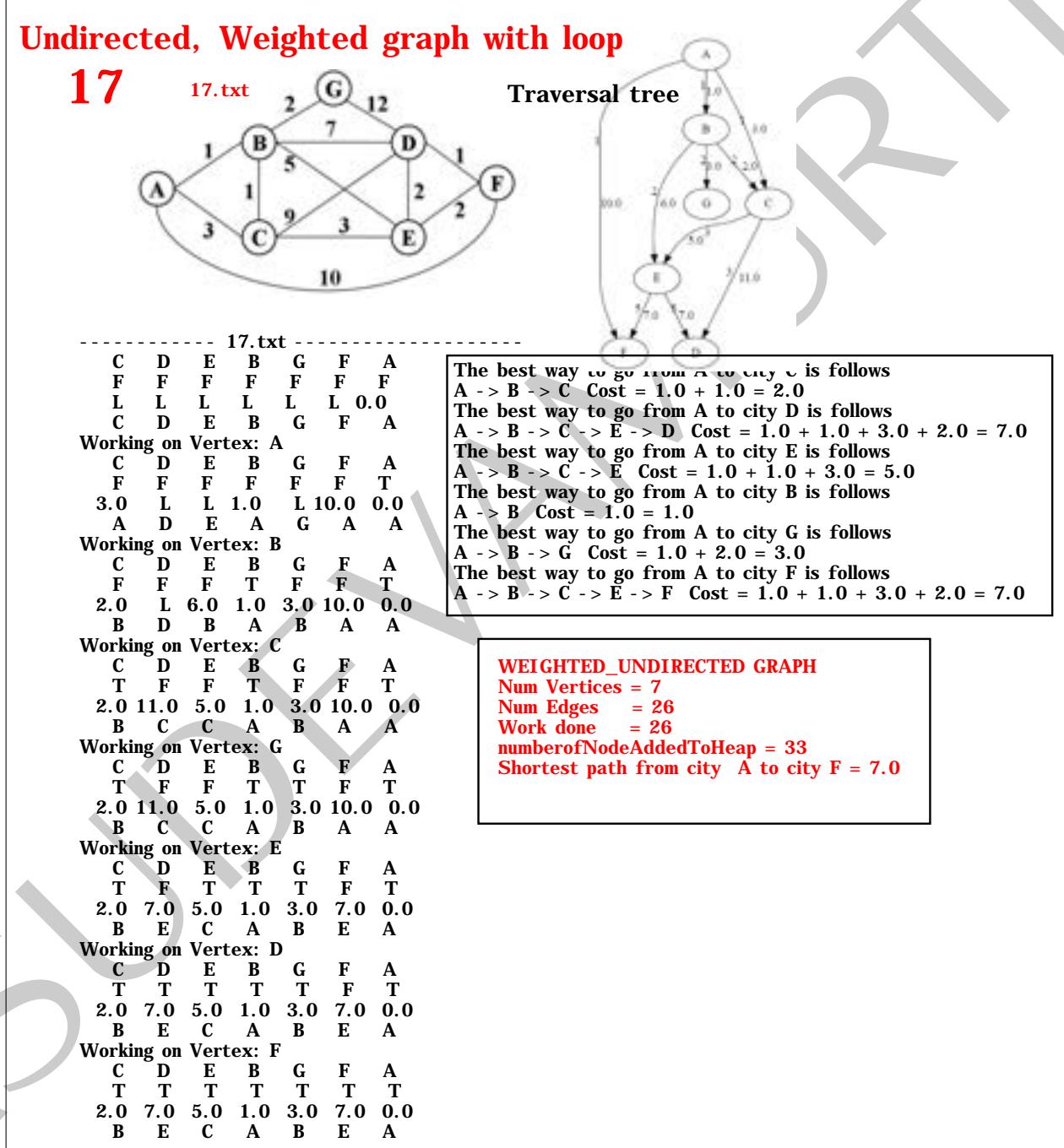


Figure 18.56: Dijkstra on undirected weighted graph

18.18 Topological sort

18.18.1 Topological sort on a directed acyclic graph

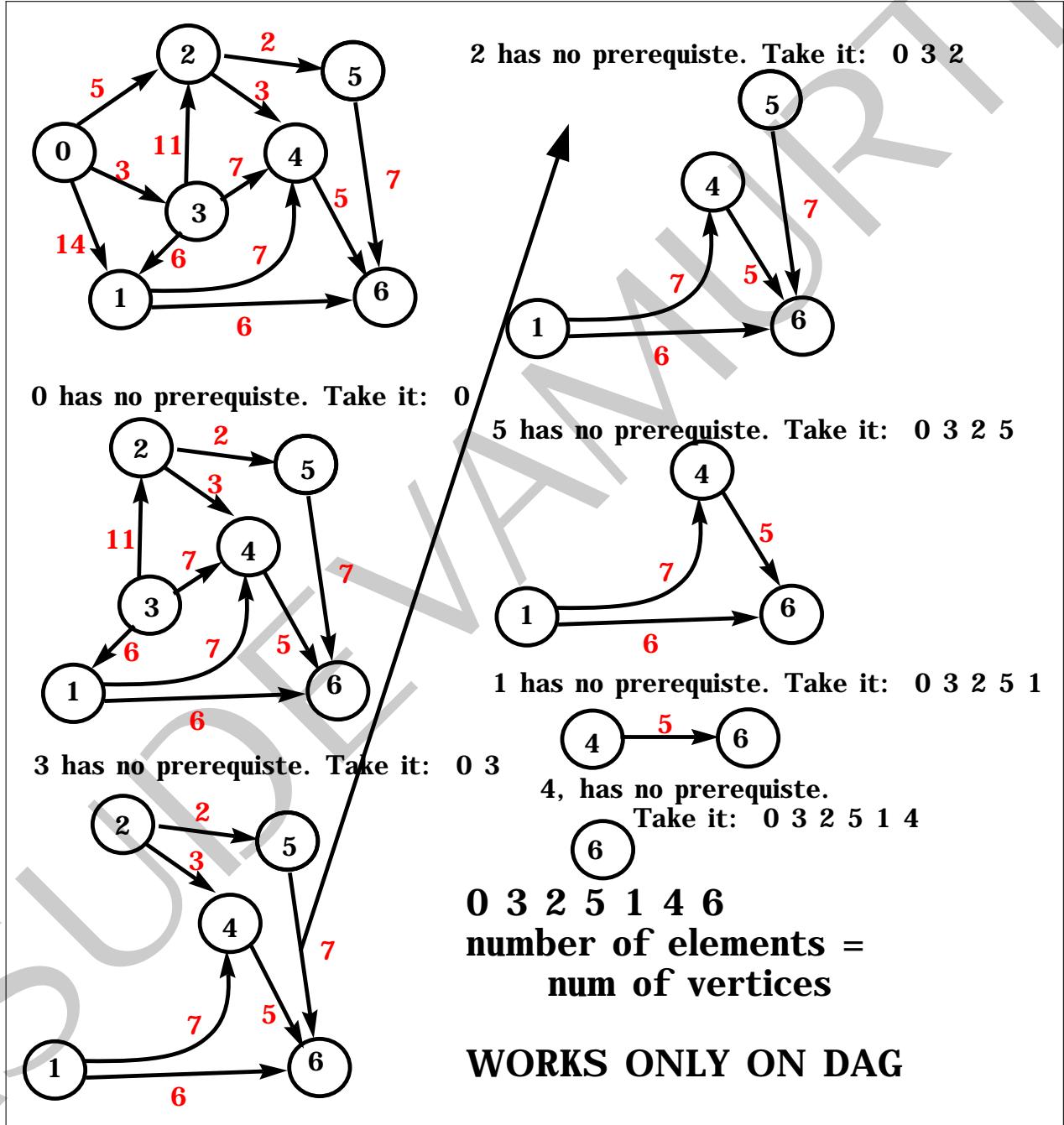


Figure 18.57: Topological sort on a DAG

18.18.2 Topological sort on a directed graph

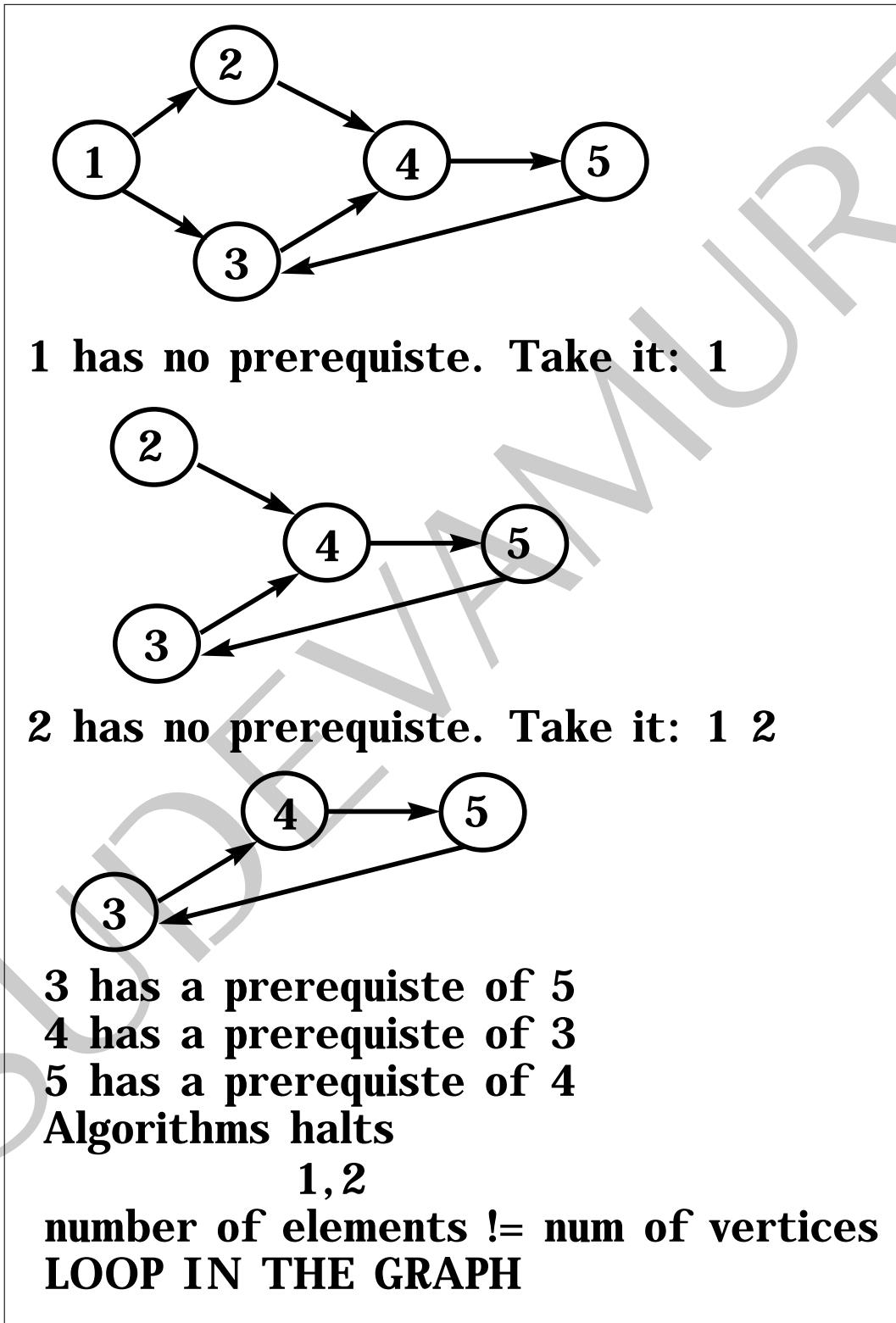


Figure 18.58: Topological sort on a graph that has cycle

18.18.3 Topological sort algorithm

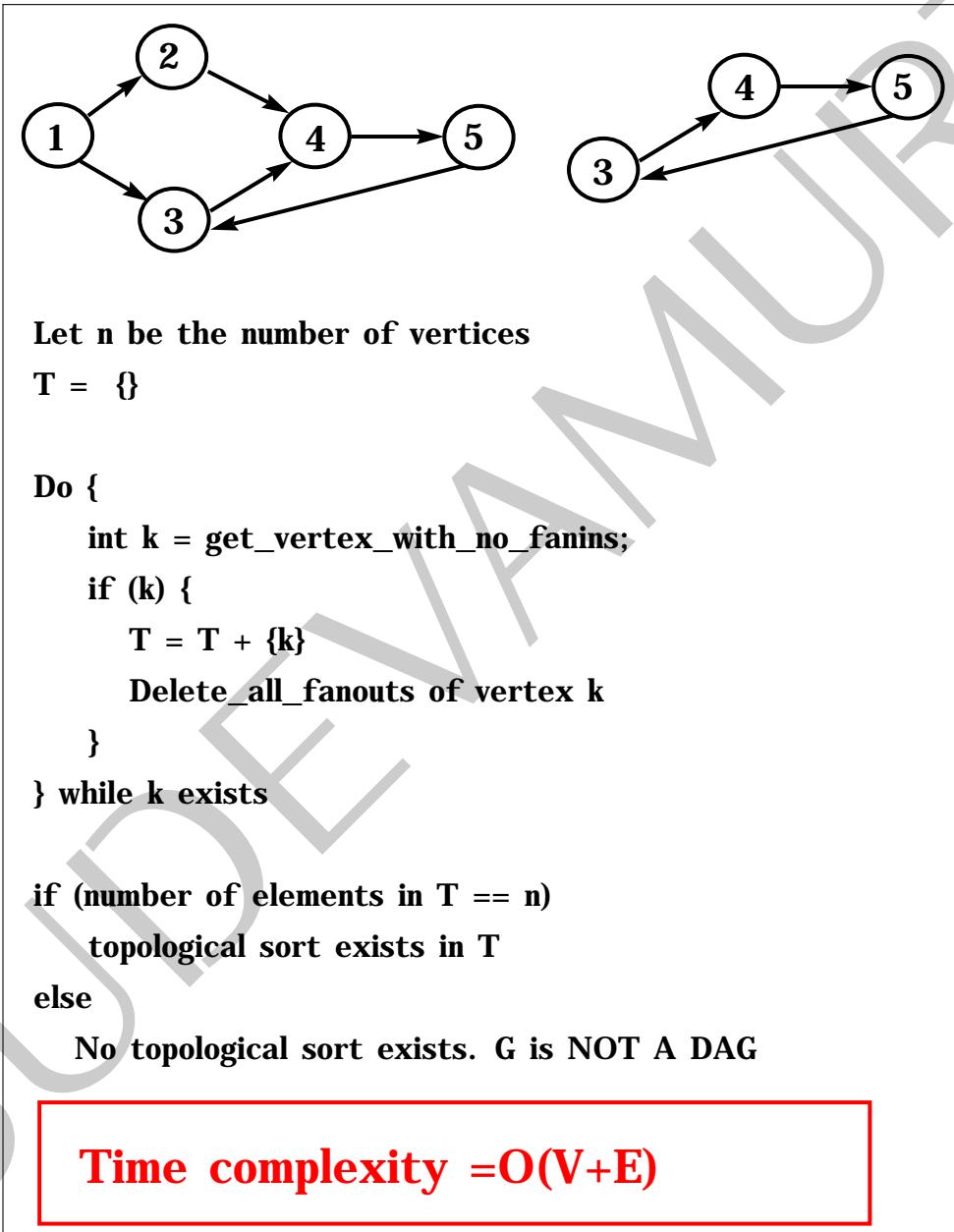


Figure 18.59: Topological sort algorithm

18.18.4 Topological sort examples

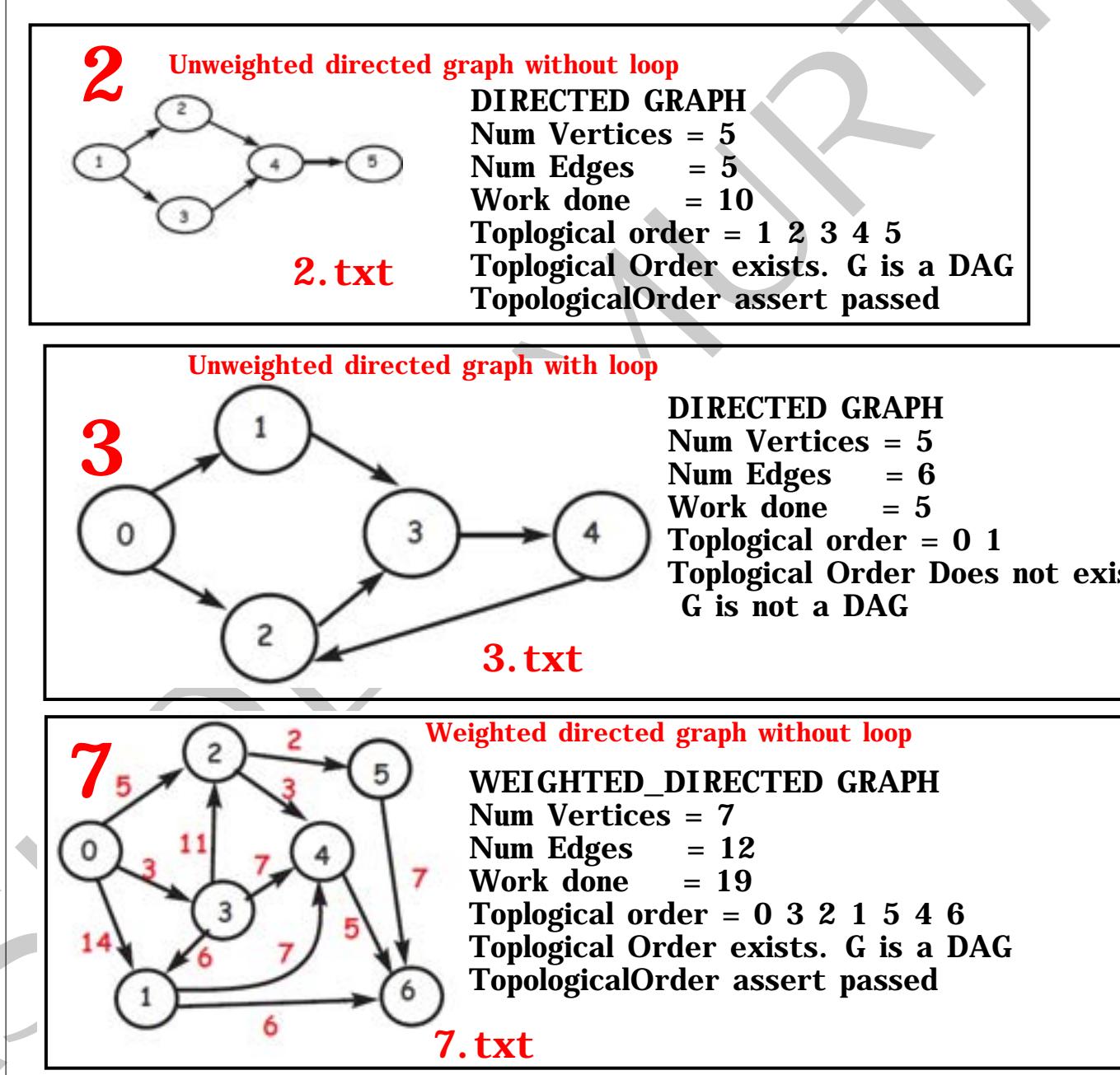


Figure 18.60: Topological sort examples

VASUDEVAMURTHY

Chapter 19

Algorithms Patterns

19.1 Introduction

19.2 Sliding Windows Patterns

19.2.1 Sliding window Maximum Average

Please refer to the section 20.30.

19.2.2 Sliding window maximum

Please refer to the section 20.31.

VASUDEVAMURTHY

Chapter 20

Algorithms for Interviews

20.1 Introduction

20.2 LeetCode Problem 38: Count and say

LOOK AND SAY PUZZLE
Leetcode 38. Count and Say

<https://leetcode.com/problems/count-and-say/>

"3322251"
two 3's, three 2's, one 5, and one 1
2 3 + 3 2 + 1 5 + 1 1
"23321511"

n=7

1	1
2	11
3	21
4	1211
5	111221
6	312211
7	13112221

Return the n string

string is given: 9999999999
return: string 109

Figure 20.1: Problem definition

20.2. LEETCODE PROBLEM 38: COUNT AND SAY

20.2.1 Expected output

```
n = 1 Length = 1
n = 1 1
n = 2 Length = 2
n = 2 11
n = 3 Length = 2
n = 3 21
n = 4 Length = 4
n = 4 1211
n = 5 Length = 6
n = 5 111221
n = 6 Length = 6
n = 6 312211
n = 7 Length = 8
n = 7 13112221
n = 8 Length = 10
n = 8 1113213211
n = 9 Length = 14
n = 9 31131211131221
n = 10 Length = 20
n = 10 13211311123113112211
n = 11 Length = 26
n = 11 11131221133112132113212221
n = 12 Length = 34
n = 12 3113112221232112111312211312113211
n = 13 Length = 46
n = 13 1321132132111213122112311311222113111221131221
n = 14 Length = 62
n = 14 11131221131211131231121113112221121321132132211331222113112211
n = 15 Length = 78
n = 15 311311222113111231131112132112311321322112111312211312111322212311322113212211311221131123113
n = 16 Length = 102
n = 16 13211321322113311213211331121113122112132113121113222112311311222113112221131123113
n = 17 Length = 134
n = 17 11131221131211132221232112111312212321123113112221121113122113112211311123113322112
n = 18 Length = 176
n = 18 31131122211311123113321112131221123113112211121312211213211322112311311222113112221131123113
n = 19 Length = 226
n = 19 13211321322113311213212311211131122211213211321223112111311222112111312211211131221121113122112
n = 20 Length = 302
```

```
n = 20 111312211312111322212321121113121112131121321123113213221121113122113121  
n = 21 Length = 408  
n = 21 311311222113111231133211121312211231131123112111331121113122112132113121  
n = 22 Length = 528  
n = 22 1321132132211331121321231231121113112221121321133112132112312321123113121  
n = 23 Length = 678  
n = 23 1113122113121113222123211211131211121311121321123113213221121113122123211  
n = 24 Length = 904  
n = 24 311311222113111231133211121312211231131123112111331121113122112132113121  
n = 25 Length = 1182  
n = 25 1321132132211331121321231231121113112221121321133112132112312321123113121  
n = 26 Length = 1540  
n = 26 1113122113121113222123211211131211121311121321123113213221121113122123211  
n = 27 Length = 2012  
n = 27 311311222113111231133211121312211231131123112111331121113122112132113121  
n = 28 Length = 2606  
n = 28 1321132132211331121321231231121113112221121321133112132112312321123113121  
n = 29 Length = 3410  
n = 29 1113122113121113222123211211131211121311121321123113213221121113122123211  
n = 30 Length = 4462  
n = 30 311311222113111231133211121312211231131123112111331121113122112132113121  
n = 31 Length = 5808  
n = 31 1321132132211331121321231231121113112221121321133112132112312321123113121  
ALL TESTS PASSED
```

20.2.2 Leetcode output

20.2. LEETCODE PROBLEM 38: COUNT AND SAY

The screenshot shows a LeetCode problem page for "Count and Say". The title is "38. Count and Say" and the URL is <https://leetcode.com/problems/count-and-say/>. The status is "Success" with a "Details" link. The runtime is 118 ms, faster than 5.34% of Python3 online submissions, and the memory usage is 14.3 MB, less than 76.51% of Python3 online submissions. Below the stats is a "Next challenge" link. The code area contains the following Python code:

```
class Solution:
    def __init__(self):
        pass

    #THIS IS LEETCODE
    def countAndSay(self, n: 'int') -> 'string':
        ## NOTHING CAN BE CHANGED HERE
        return self._alg_integer(n)

    ## n is str here
    def _alg_string(self, n: 'string') -> 'string':
        YOU WRITE CODE
```

Figure 20.2: Problem 38: Count and say passed

20.3 LeetCode Problem 66: Plus one

66. Plus One

<https://leetcode.com/problems/plus-one/>

You are given a large integer represented as an integer array digits, where each digits[i] is the ith digit of the integer. The digits are ordered from most significant to least significant in left- to- right order. The large integer does not contain any leading 0's.

Increment the large integer by one and return the resulting array of digits.

Input: [1, 2, 3]

Output: [1, 2, 4]

Explanation: The array represents the integer 123.

Example 2:

Input: [4, 3, 2, 1]

Output: [4, 3, 2, 2]

Explanation: The array represents the integer 4321.

Input: digits = [9]

Output: [1, 0]

Explanation: $9 + 1 = 10$

Figure 20.3: Problem definition

20.3.1 Expected output

Testing PlusOneBase.py Starts

1

2

2

20.3. LEETCODE PROBLEM 66: PLUS ONE

2

9	
1	0
1	0
1	0

1	2	3
1	2	4
1	2	4
1	2	4

1	7	8	9
1	7	9	0
1	7	9	0
1	7	9	0

9	8	9
9	9	0
9	9	0
9	9	0

9	9	
1	0	0
1	0	0
1	0	0

9 9 8
9 9 9
9 9 9
9 9 9

```
100000 tests start 100000
100000 tests passed
ALL TESTS PASSED
Testing PlusOneBase.py Starts
```

20.3.2 Leetcode output

66. Plus One

<https://leetcode.com/problems/plus-one/>

SUCCESS Details ↗

Runtime: 45 ms, faster than 26.57% of Python3 online submissions for Plus One.

Memory Usage: 14.3 MB, less than 48.47% of Python3 online submissions for Plus One.

class Solution:

```
def __init__(self):
    pass
```

LEETCODE implementation

```
def plusOne(self, digits: List[int]) -> List[int]:
    ## NOTHING CAN BE CHANGED HERE
    return self._alg(digits)
```

```
def _alg(self, a:List[int]) -> List[int]:
    //WRITE CODE
```

Figure 20.4: Problem 66: Plus one passed

20.4. LEETCODE PROBLEM 2: ADD TWO NUMBERS

20.4 LeetCode Problem 2: Add Two Numbers

Add Two Numbers

<https://leetcode.com/problems/add-two-numbers/>

2. Add Two Numbers

Medium · 25,193 · 92,270 · 17 Accepted · 103 Submissions

You are given two non-empty linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers, and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Example 1:

```
graph LR; N1((2)) --> N2((4)); N2 --> N3((3)); N4((5)) --> N5((6)); N5 --> N6((4)); N7((7)) --> N8((0)); N8 --> N9((8));
```

Input: $l1 = [2,4,3]$, $l2 = [5,6,4]$
Output: $[7,0,8]$
Explanation: $342 + 465 = 807$.

Example 2:

Input: $l1 = [9]$, $l2 = [9]$
Output: $[8]$

Example 3:

Input: $l1 = [9,9,9,9,9,9,9]$, $l2 = [9,9,9,9]$
Output: $[8,9,9,9,9,9,9,9]$

Figure 20.5: Problem definition

20.4.1 Leetcode output

2. Add Two Numbers

<https://leetcode.com/problems/add-two-numbers/>

Success Details ↗

Runtime: 70 ms, faster than 88.44% of Python3 online submissions for Add Two Numbers.

Memory Usage: 14.1 MB, less than 10.01% of Python3 online submissions for Add Two Numbers.

```
class Solution:
    methods CAN BE CHANGED BELOW
    def addTwoNumbers(self, l1: 'ListNode', l2: 'ListNode') -> 'ListNode':
        show = False
        assert_answers = True
        ans = [None] #list of size 1
        h = ListNode(11,ans,assert_answers,show)
        return ans[0]

class ListNode:
    def __init__(self, l1: 'ListNode', l2: 'ListNode', ans:'listNode of size 1', assert_answers:'bool', show:'bool'):
        -> None:
            METHODS CAN BE CHANGED BELOW
            self._a = SList()
            self._a._first = l1
            self._b = SList()
            self._b._first = l2
            self._ans = ans
            self._sign()

    def _alg(self):
        t = self._a + self._b
        self._ans[0] = t._first

class SList():
    def __init__(self):
        METHODS CAN BE CHANGED HERE
        self._fFirst = None
        self._last = None
```

Cannot change this code

Cannot change this code

Write code below

Figure 20.6: Problem 02: Add Two Numbers passed

20.5. LEETCODE PROBLEM 977: SQUARE OF A SORTED ARRAY

20.5 LeetCode Problem 977: Square of a sorted array

977. Squares of a Sorted Array

<https://leetcode.com/problems/squares-of-a-sorted-array/>

Given an integer array `nums`, sorted in **non-decreasing** order, return an array of **the squares of each number** sorted in non-decreasing order.

Example 1:

```
Input: nums = [-4,-1,0,3,10]
Output: [0,1,9,16,100]
Explanation: After squaring, the array becomes [16,1,0,-1,-4]. After sorting, it becomes [0,1,4,9,16].
```

Example 2:

```
Input: nums = [-7,-3,2,3,11]
Output: [4,9,9,49,121]
```

Constraints:

- $1 \leq \text{nums.length} \leq 10^4$
- $-10^4 \leq \text{nums}[i] \leq 10^4$
- nums is sorted in **non-decreasing** order.

Figure 20.7: L0977Squaresofasortedarray: Square of a sorted array

20.5.1 Expected output

Testing L0977Test Starts

```
-----  
alg = NSquareTime_1_Space n = 0 work = 0 CPU time in sec = 0.0  
alg = NlognTime_N_Space n = 0 work = 0 CPU time in sec = 0.0  
alg = NTime_1_Space n = 0 work = 0 CPU time in sec = 0.0  
-----  
0 1 2  
1 2 3  
1 4 9  
alg = NSquareTime_1_Space n = 3 work = 9 CPU time in sec = 0.0  
1 4 9  
alg = NlognTime_N_Space n = 3 work = 7 CPU time in sec = 0.0  
1 4 9  
alg = NTime_1_Space n = 3 work = 6 CPU time in sec = 0.0  
-----  
0 1  
-2 1  
1 4  
alg = NSquareTime_1_Space n = 2 work = 5 CPU time in sec = 0.0  
1 4  
alg = NlognTime_N_Space n = 2 work = 4 CPU time in sec = 0.0  
1 4  
alg = NTime_1_Space n = 2 work = 4 CPU time in sec = 0.0  
-----  
0 1 2  
-3 -2 3  
4 9 9  
alg = NSquareTime_1_Space n = 3 work = 9 CPU time in sec = 0.0  
4 9 9  
alg = NlognTime_N_Space n = 3 work = 7 CPU time in sec = 0.0  
4 9 9  
alg = NTime_1_Space n = 3 work = 6 CPU time in sec = 0.0  
-----  
0 1 2 3 4  
-4 -1 0 3 10  
0 1 9 16 100  
alg = NSquareTime_1_Space n = 5 work = 20 CPU time in sec = 0.0  
0 1 9 16 100  
alg = NlognTime_N_Space n = 5 work = 16 CPU time in sec = 0.0  
0 1 9 16 100  
alg = NTime_1_Space n = 5 work = 10 CPU time in sec = 0.0  
Random tests on Array of size 0  
alg = NSquareTime_1_Space n = 0 work = 0 CPU time in sec = 0.0  
alg = NlognTime_N_Space n = 0 work = 0 CPU time in sec = 0.0  
alg = NTime_1_Space n = 0 work = 0 CPU time in sec = 0.0  
Random tests on Array of size 1001  
alg = NSquareTime_1_Space n = 1001 work = 502502 CPU time in sec = 0.09375  
alg = NlognTime_N_Space n = 1001 work = 10978 CPU time in sec = 0.0  
alg = NTime_1_Space n = 1001 work = 2002 CPU time in sec = 0.0  
Random tests on Array of size 2002  
alg = NSquareTime_1_Space n = 2002 work = 2007005 CPU time in sec = 0.34375  
alg = NlognTime_N_Space n = 2002 work = 23958 CPU time in sec = 0.0  
alg = NTime_1_Space n = 2002 work = 4004 CPU time in sec = 0.0  
Random tests on Array of size 3003  
alg = NSquareTime_1_Space n = 3003 work = 4513509 CPU time in sec = 0.875  
alg = NlognTime_N_Space n = 3003 work = 37694 CPU time in sec = 0.0  
alg = NTime_1_Space n = 3003 work = 6006 CPU time in sec = 0.015625  
Random tests on Array of size 4004  
alg = NSquareTime_1_Space n = 4004 work = 8022014 CPU time in sec = 1.484375  
alg = NlognTime_N_Space n = 4004 work = 51920 CPU time in sec = 0.0  
alg = NTime_1_Space n = 4004 work = 8008 CPU time in sec = 0.0
```

20.5. LEETCODE PROBLEM 977: SQUARE OF A SORTED ARRAY

20.5.2 Leetcode output

977. Squares of a Sorted Array

```
class Solution:
    #YOU CANNOT CHANGE THIS INTERFACE
    def sortedSquares(self, nums: List[int]) -> List[int]:
        ans = []
        work = [0]
        #p = L0977("NSquareTime_1_Space",nums,ans,work,False)
        #p = L0977("NlognTime_N_Space",nums,ans,work,False)
        p = L0977("NTime_1_Space",nums,ans,work,False)
        return ans
```

Time Limit Exceeded [Details >](#)

Last executed input

Success [Details >](#)

Runtime: 182 ms, faster than 64.95% of Python3 online submissions for Squares of a Sorted Array.

Memory Usage: 19.8 MB, less than 5.85% of Python3 online submissions for Squares of a Sorted Array.

Success [Details >](#)

Runtime: 176 ms, faster than 71.33% of Python3 online submissions for Squares of a Sorted Array.

Memory Usage: 19.8 MB, less than 5.85% of Python3 online submissions for Squares of a Sorted Array.

Figure 20.8: Problem 977: Square of a sorted array passed

20.6. LEETCODE PROBLEM 233: PRODUCT OF ARRAY EXCEPT SELF

20.6 LeetCode Problem 233: Product of Array Except Self

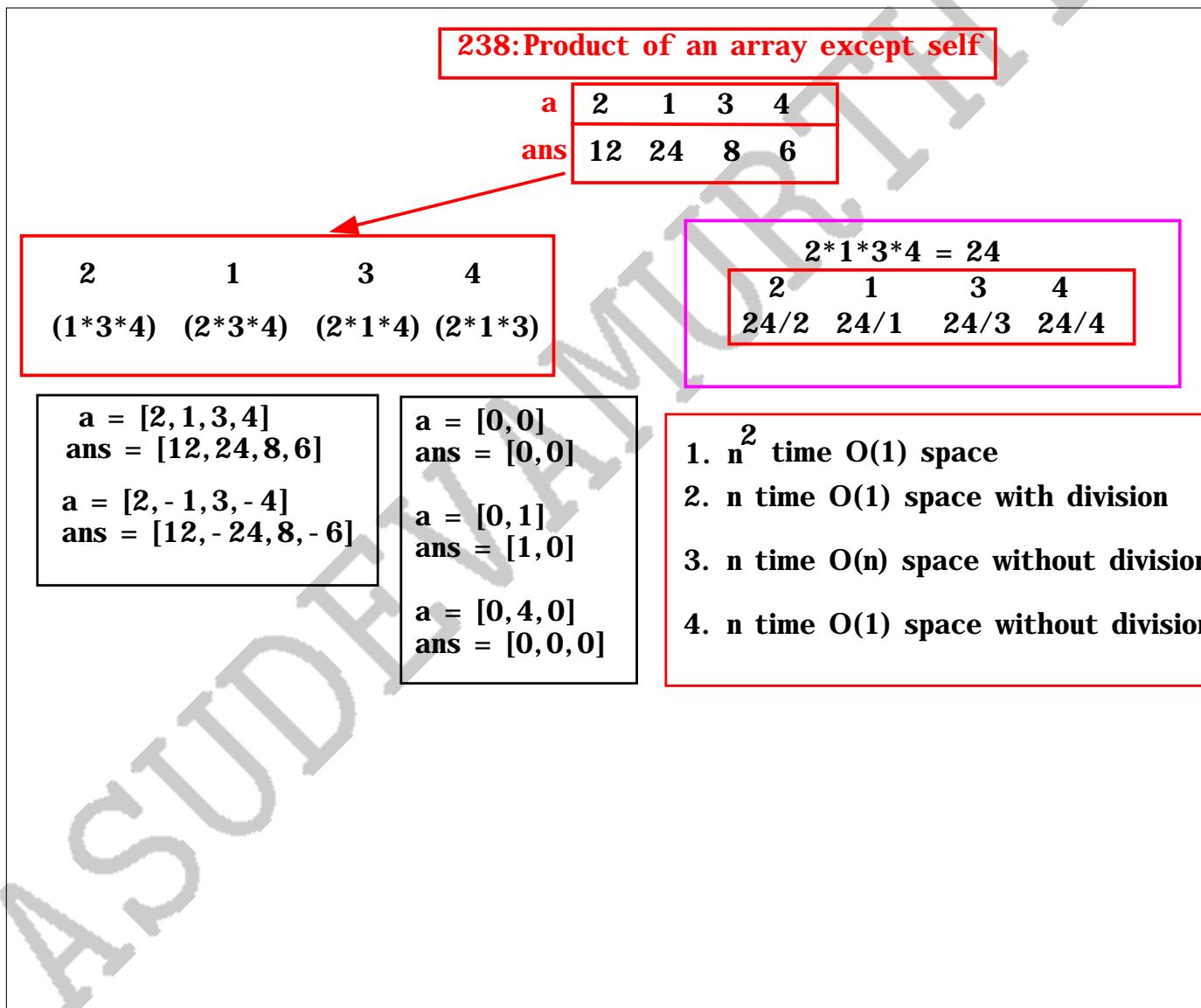


Figure 20.9: chaf: Product of Array Except Self

20.6.1 Expected output

Testing L0238Test Starts

-----PROBLEM 1 -----

0	1	2	3
2	1	3	4
12	24	8	6

n = 4

Brute Force: Work= 16 CPU time in Sec 0.0

Use Division: Work= 12 CPU time in Sec 0.0

n time n space: Work= 10 CPU time in Sec 0.0

n time 1 space: Work= 22 CPU time in Sec 0.0

-----PROBLEM 2 -----

0	1	2	3
2	-1	3	-4
12	-24	8	-6

n = 4

Brute Force: Work= 16 CPU time in Sec 0.0

Use Division: Work= 12 CPU time in Sec 0.0

n time n space: Work= 10 CPU time in Sec 0.0

n time 1 space: Work= 22 CPU time in Sec 0.0

-----PROBLEM 3 -----

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9

362880 0 0 0 0 0 0 0 0 0 0

n = 10

Brute Force: Work= 100 CPU time in Sec 0.0

Use Division: Work= 30 CPU time in Sec 0.0

n time n space: Work= 28 CPU time in Sec 0.0

n time 1 space: Work= 58 CPU time in Sec 0.0

-----PROBLEM 4 -----

0	1
0	0
0	0

n = 2

Brute Force: Work= 4 CPU time in Sec 0.0

Use Division: Work= 4 CPU time in Sec 0.0

n time n space: Work= 4 CPU time in Sec 0.0

n time 1 space: Work= 10 CPU time in Sec 0.0

-----PROBLEM 5 -----

0	1
0	1
1	0

n = 2

20.6. LEETCODE PROBLEM 233: PRODUCT OF ARRAY EXCEPT SELF

Brute Force: Work= 4 CPU time in Sec 0.0
Use Division: Work= 6 CPU time in Sec 0.0
n time n space: Work= 4 CPU time in Sec 0.0
n time 1 space: Work= 10 CPU time in Sec 0.0
-----PROBLEM 6 -----
0 1 2
0 4 0
0 0 0
n = 3
Brute Force: Work= 9 CPU time in Sec 0.0
Use Division: Work= 6 CPU time in Sec 0.0
n time n space: Work= 7 CPU time in Sec 0.0
n time 1 space: Work= 16 CPU time in Sec 0.0
Random tests on Array of size 1001
n = 1001
Brute Force: Work= 1002001 CPU time in Sec 0.546875
Use Division: Work= 3003 CPU time in Sec 0.0
n time n space: Work= 3001 CPU time in Sec 0.015625
n time 1 space: Work= 6004 CPU time in Sec 0.03125
Random tests on Array of size 1973
n = 1973
Brute Force: Work= 3892729 CPU time in Sec 2.828125
Use Division: Work= 5919 CPU time in Sec 0.015625
n time n space: Work= 5917 CPU time in Sec 0.125
n time 1 space: Work= 11836 CPU time in Sec 0.125
Random tests on Array of size 2945
n = 2945
Brute Force: Work= 8673025 CPU time in Sec 5.515625
Use Division: Work= 8835 CPU time in Sec 0.0
n time n space: Work= 8833 CPU time in Sec 0.0
n time 1 space: Work= 17668 CPU time in Sec 0.015625
Random tests on Array of size 3917
n = 3917
Brute Force: Work= 15342889 CPU time in Sec 6.921875
Use Division: Work= 11751 CPU time in Sec 0.0
n time n space: Work= 11749 CPU time in Sec 0.015625
n time 1 space: Work= 23500 CPU time in Sec 0.015625
Random tests on Array of size 4889
n = 4889
Brute Force: Work= 23902321 CPU time in Sec 30.546875
Use Division: Work= 14667 CPU time in Sec 0.0625

```
n time n space: Work= 14665 CPU time in Sec 1.390625
n time 1 space: Work= 29332 CPU time in Sec 1.484375
Testing L0238Test ENDS
```

20.6.2 Leetcode output

238. Product of Array Except Self
<https://leetcode.com/problems/product-of-array-except-self/>



The screenshot shows four successful submissions for the 'Product of Array Except Self' problem on LeetCode. Each submission has a code snippet, runtime, and memory usage.

- Submission 1:** Brute Force (Time Limit Exceeded)

```
if (True):
    b = L0238(nums,ans,work,"Brute Force")
```

Runtime: 313 ms, faster than 5.04% of Python3 online submissions for Product of Array Except Self.
Memory Usage: 24.4 MB, less than 22.80% of Python3 online submissions for Product of Array Except Self.
- Submission 2:** Use Division

```
if (True):
    b = L0238(nums,ans,work,"Use Division")
```

Success Details >
Runtime: 313 ms, faster than 5.04% of Python3 online submissions for Product of Array Except Self.
Memory Usage: 24.4 MB, less than 22.80% of Python3 online submissions for Product of Array Except Self.
- Submission 3:** n time n space

```
if (True):
    b = L0238(nums,ans,work,"n time n space")
```

Success Details >
Runtime: 328 ms, faster than 5.04% of Python3 online submissions for Product of Array Except Self.
Memory Usage: 25.6 MB, less than 11.97% of Python3 online submissions for Product of Array Except Self.
- Submission 4:** n time 1 space

```
if (True):
    b = L0238(nums,ans,work,"n time 1 space")
```

Success Details >
Runtime: 399 ms, faster than 5.04% of Python3 online submissions for Product of Array Except Self.
Memory Usage: 24.5 MB, less than 22.44% of Python3 online submissions for Product of Array Except Self.

```
class Solution:
    def productExceptSelf(self, nums: List[int]) -> List[int]:
        work = [0]
        ans = []
        if (False):
            b = L0238(nums,ans,work,"Brute Force")
        if (False):
            b = L0238(nums,ans,work,"Use Division")
        if (False):
            b = L0238(nums,ans,work,"n time n space")
        if (True):
            b = L0238(nums,ans,work,"n time 1 space")
        return ans
```

Figure 20.10: Problem 238: Product of Array Except Self passed

20.7 Find duplicate elements in a Python list

Find duplicate numbers in a given list a

1. List a, of size of n, given
2. Each list element is between 0 to (n-1)

$n = 4$
0 1 2 3
3 2 1 0
No dup

$n = 4$
0 1 2 3
2 3 1 3
duplicate {3}

0 1 2 3 4 5 6
1 2 3 1 3 6 6
duplicate {1, 3, 6}

Goal:

1. Collect all duplicates exactly once in the list d

0 1 2
1 1 1
wrong $d = \{1, 1\}$

0 1 2
1 1 1
 $d = \{1\}$

2. Given list a should be intact at the end of your algorithm

3. We need $O(n)$ time and $O(1)$ space algorithm

1. Implement $O(n^2)$ time and $O(1)$ space algorithm
2. Implement $O(n)$ time and $O(n)$ space algorithm
3. Implement $O(n)$ time and $O(1)$ space algorithm

You cannot use any library functions like set/map etc.
Must use only Python list []

Figure 20.11: Find duplicate elements in a Python list

20.7.1 Expected output

```
Testing duplicateN.py Starts
----- Test case  0  Size of n = 3
  0  1  2
  1  1  1
----- NsquareTimeConstantSpace -----
[1]
Num Duplicate 1 Work done = 13 CPU time in sec = 0.0
----- NTimeNSpace -----
[1]
Num Duplicate 1 Work done = 4 CPU time in sec = 0.0
----- NTimeConstantSpace -----
[1]
Num Duplicate 1 Work done = 13 CPU time in sec = 0.0
----- Test case  1  Size of n = 4
  0  1  2  3
  1  0  0  0
----- NsquareTimeConstantSpace -----
[0]
Num Duplicate 1 Work done = 18 CPU time in sec = 0.0
----- NTimeNSpace -----
[0]
Num Duplicate 1 Work done = 5 CPU time in sec = 0.0
----- NTimeConstantSpace -----
[0]
Num Duplicate 1 Work done = 18 CPU time in sec = 0.0
----- Test case  2  Size of n = 7
  0  1  2  3  4  5  6
  1  2  3  1  3  6  6
----- NsquareTimeConstantSpace -----
[1, 3, 6]
Num Duplicate 3 Work done = 39 CPU time in sec = 0.0
----- NTimeNSpace -----
[1, 3, 6]
Num Duplicate 3 Work done = 10 CPU time in sec = 0.0
----- NTimeConstantSpace -----
[1, 3, 6]
Num Duplicate 3 Work done = 35 CPU time in sec = 0.0
----- Test case  3  Size of n = 7
  0  1  2  3  4  5  6
```

20.7. FIND DUPLICATE ELEMENTS IN A PYTHON LIST

```
1 2 3 1 3 0 0
----- NsquareTimeConstantSpace -----
[0, 1, 3]
Num Duplicate 3 Work done = 39 CPU time in sec = 0.0
----- NTimeNSpace -----
[0, 1, 3]
Num Duplicate 3 Work done = 10 CPU time in sec = 0.0
----- NTimeConstantSpace -----
[0, 1, 3]
Num Duplicate 3 Work done = 35 CPU time in sec = 0.0
----- Test case 4 Size of n = 3
0 1 2
0 0 1
----- NsquareTimeConstantSpace -----
[0]
Num Duplicate 1 Work done = 11 CPU time in sec = 0.0
----- NTimeNSpace -----
[0]
Num Duplicate 1 Work done = 4 CPU time in sec = 0.0
----- NTimeConstantSpace -----
[0]
Num Duplicate 1 Work done = 15 CPU time in sec = 0.0
----- Test case 5 Size of n = 4
0 1 2 3
3 2 0 1
----- NsquareTimeConstantSpace -----
[]
Num Duplicate 0 Work done = 10 CPU time in sec = 0.0
----- NTimeNSpace -----
[]
Num Duplicate 0 Work done = 4 CPU time in sec = 0.0
----- NTimeConstantSpace -----
[]
Num Duplicate 0 Work done = 20 CPU time in sec = 0.0
All basic tests passed
-----Testing on 5000 Random numbers
----- Test case 6 Size of n = 5000
----- NsquareTimeConstantSpace -----
Num Duplicate 1241 Work done = 7651231 CPU time in sec = 3.609375
----- NTimeNSpace -----
Num Duplicate 1241 Work done = 6241 CPU time in sec = 0.0
```

```
----- NTimeConstantSpace -----  
Num Duplicate 1241 Work done = 22508 CPU time in sec = 0.015625  
----Testing on 10000 Random numbers  
----- Test case 7 Size of n = 10000  
----- NsquareTimeConstantSpace -----  
Num Duplicate 2535 Work done = 30613032 CPU time in sec = 12.703125  
----- NTimeNSpace -----  
Num Duplicate 2535 Work done = 12535 CPU time in sec = 0.0  
----- NTimeConstantSpace -----  
Num Duplicate 2535 Work done = 45054 CPU time in sec = 0.015625  
----Testing on 15000 Random numbers  
----- Test case 8 Size of n = 15000  
----- NsquareTimeConstantSpace -----  
Num Duplicate 3696 Work done = 68460255 CPU time in sec = 25.96875  
----- NTimeNSpace -----  
Num Duplicate 3696 Work done = 18696 CPU time in sec = 0.015625  
----- NTimeConstantSpace -----  
Num Duplicate 3696 Work done = 67280 CPU time in sec = 0.046875  
----Testing on list that is completely same 5000 All numbers are same  
----- Test case 9 Size of n = 5000  
----- NsquareTimeConstantSpace -----  
Num Duplicate 1 Work done = 24998 CPU time in sec = 0.015625  
----- NTimeNSpace -----  
Num Duplicate 1 Work done = 5001 CPU time in sec = 0.0  
----- NTimeConstantSpace -----  
Num Duplicate 1 Work done = 15004 CPU time in sec = 0.0  
----Testing on list that is completely same 10000 All numbers are same  
----- Test case 10 Size of n = 10000  
----- NsquareTimeConstantSpace -----  
Num Duplicate 1 Work done = 49998 CPU time in sec = 0.015625  
----- NTimeNSpace -----  
Num Duplicate 1 Work done = 10001 CPU time in sec = 0.015625  
----- NTimeConstantSpace -----  
Num Duplicate 1 Work done = 30004 CPU time in sec = 0.015625  
----Testing on list that is completely same 15000 All numbers are same  
----- Test case 11 Size of n = 15000  
----- NsquareTimeConstantSpace -----  
Num Duplicate 1 Work done = 74998 CPU time in sec = 0.03125  
----- NTimeNSpace -----  
Num Duplicate 1 Work done = 15001 CPU time in sec = 0.0  
----- NTimeConstantSpace -----
```

20.7. FIND DUPLICATE ELEMENTS IN A PYTHON LIST

```
Num Duplicate 1 Work done = 45004 CPU time in sec = 0.015625
-----Testing on list that is in decreasing order with no duplicates 5000 descending_order
----- Test case 12 Size of n = 5000
----- NsquareTimeConstantSpace -----
Num Duplicate 0 Work done = 12502500 CPU time in sec = 4.5
----- NTimeNSpace -----
Num Duplicate 0 Work done = 5000 CPU time in sec = 0.0
----- NTimeConstantSpace -----
Num Duplicate 0 Work done = 25000 CPU time in sec = 0.015625
-----Testing on list that is in decreasing order with no duplicates 10000 descending_order
----- Test case 13 Size of n = 10000
----- NsquareTimeConstantSpace -----
Num Duplicate 0 Work done = 50005000 CPU time in sec = 17.40625
----- NTimeNSpace -----
Num Duplicate 0 Work done = 10000 CPU time in sec = 0.0
----- NTimeConstantSpace -----
Num Duplicate 0 Work done = 50000 CPU time in sec = 0.015625
-----Testing on list that is in decreasing order with no duplicates 15000 descending_order
----- Test case 14 Size of n = 15000
----- NsquareTimeConstantSpace -----
Num Duplicate 0 Work done = 112507500 CPU time in sec = 39.5625
----- NTimeNSpace -----
Num Duplicate 0 Work done = 15000 CPU time in sec = 0.015625
----- NTimeConstantSpace -----
Num Duplicate 0 Work done = 75000 CPU time in sec = 0.03125
-----Testing on list that is in increasing order with no duplicates 5000 ascending_order
----- Test case 15 Size of n = 5000
----- NsquareTimeConstantSpace -----
Num Duplicate 0 Work done = 12502500 CPU time in sec = 4.34375
----- NTimeNSpace -----
Num Duplicate 0 Work done = 5000 CPU time in sec = 0.0
----- NTimeConstantSpace -----
Num Duplicate 0 Work done = 25000 CPU time in sec = 0.015625
-----Testing on list that is in increasing order with no duplicates 10000 ascending_order
----- Test case 16 Size of n = 10000
----- NsquareTimeConstantSpace -----
Num Duplicate 0 Work done = 50005000 CPU time in sec = 17.671875
----- NTimeNSpace -----
Num Duplicate 0 Work done = 10000 CPU time in sec = 0.0
----- NTimeConstantSpace -----
Num Duplicate 0 Work done = 50000 CPU time in sec = 0.015625
```

```
-----Testing on list that is in increasing order with no duplicates 15000 ascending
----- Test case 17 Size of n = 15000
----- NsquareTimeConstantSpace -----
Num Duplicate 0 Work done = 112507500 CPU time in sec = 50.46875
----- NTimeNSpace -----
Num Duplicate 0 Work done = 15000 CPU time in sec = 0.0
----- NTimeConstantSpace -----
Num Duplicate 0 Work done = 75000 CPU time in sec = 0.03125
All tests passed. You are a genius
If you liked this problem send me a box of chocolates
Testing duplicateN.py ENDS
```

20.8 LeetCode Problem 88: Merge Sorted Array

88. Merge Sorted Array

<https://leetcode.com/problems/merge-sorted-array/>

Given two sorted integer arrays `nums1` and `nums2`, merge `nums2` into `nums1` as one sorted array.

The number of elements initialized in `nums1` and `nums2` are `m` and `n` respectively. You may assume that `nums1` has enough space (size that is equal to `m + n`) to hold additional elements from `nums2`.

Example 1:

Input: `nums1` = [1,2,3,0,0,0], `m` = 3, `nums2` = [2,3,6], `n` = 3
Output: [1,2,2,3,3,6]

Example 2:

Input: `nums1` = [1], `m` = 1, `nums2` = [], `n` = 0
Output: [1]

Constraints:

- $0 \leq m, n \leq 200$
- $0 \leq m + n \leq 200$
- $\text{nums1.length} \leftarrow m + n$
- $\text{nums2.length} \leftarrow n$
- $-10^5 \leq \text{nums1}[i], \text{nums2}[i] \leq 10^5$

Figure 20.12: L0088: Merge Sorted Array

20.8. LEETCODE PROBLEM 88: MERGE SORTED ARRAY

20.8.1 Expected output

```
Testing L0088Test Starts
-----PROBLEM 1 -----
 0  1  2  3  4  5
 1  2  3  0  0  0
 0  1  2
 2  5  6
-----ANS-----
 0  1  2  3  4  5
 1  2  2  3  5  6
nlogntime_1space: work = 16 CPU time in sec = 0.0
ntime_nspace:    work = 12 CPU time in sec = 0.0
ntime_1space:   work = 4 CPU time in sec = 0.0
-----PROBLEM 2 -----
 0  1  2  3  4  5  6
 1  2  2 100  0  0  0
 0  1  2
 2  4  5
-----ANS-----
 0  1  2  3  4  5  6
 1  2  2  2  4  5 100
nlogntime_1space: work = 20 CPU time in sec = 0.0
ntime_nspace:    work = 14 CPU time in sec = 0.0
ntime_1space:   work = 4 CPU time in sec = 0.0
-----PROBLEM 3 -----
 0
 2
-----ANS-----
 0
 2
nlogntime_1space: work = 0 CPU time in sec = 0.0
ntime_nspace:    work = 2 CPU time in sec = 0.0
ntime_1space:   work = 0 CPU time in sec = 0.0
-----PROBLEM 4 -----
 0  1
 1  2
```

----- ANS -----

0 1
1 2

nlogntime_1space: work = 2 CPU time in sec = 0.0
ntime_nspace: work = 4 CPU time in sec = 0.0
ntime_1space: work = 0 CPU time in sec = 0.0

----- PROBLEM 5 -----

0 1 2 3
1 2 0 0
0 1
1 2

----- ANS -----

0 1 2 3
1 1 2 2

nlogntime_1space: work = 8 CPU time in sec = 0.0
ntime_nspace: work = 8 CPU time in sec = 0.0
ntime_1space: work = 3 CPU time in sec = 0.0

----- PROBLEM 6 -----

0 1 2 3
1 2 0 0
0 1
3 4

----- ANS -----

0 1 2 3
1 2 3 4

nlogntime_1space: work = 8 CPU time in sec = 0.0
ntime_nspace: work = 8 CPU time in sec = 0.0
ntime_1space: work = 2 CPU time in sec = 0.0

----- PROBLEM 7 -----

0 1 2 3
3 4 0 0
0 1
1 2

----- ANS -----

0 1 2 3
1 2 3 4

nlogntime_1space: work = 8 CPU time in sec = 0.0
ntime_nspace: work = 8 CPU time in sec = 0.0
ntime_1space: work = 4 CPU time in sec = 0.0

----- PROBLEM 8 -----

0 1 2 3

20.8. LEETCODE PROBLEM 88: MERGE SORTED ARRAY

```
1   3   0   0
0   1
2   4
-----
----- ANS -----
0   1   2   3
1   2   3   4
nlogntime_1space: work = 8 CPU time in sec = 0.0
ntime_nspace:     work = 8 CPU time in sec = 0.0
ntime_1space:     work = 3 CPU time in sec = 0.0
-----
----- PROBLEM 9 -----
0   1   2   3
2   4   0   0
0   1
1   2
-----
----- ANS -----
0   1   2   3
1   2   2   4
nlogntime_1space: work = 8 CPU time in sec = 0.0
ntime_nspace:     work = 8 CPU time in sec = 0.0
ntime_1space:     work = 4 CPU time in sec = 0.0
Random tests on Array of size 10000
nlogntime_1space: work = 285755 CPU time in sec = 0.0
ntime_nspace:     work = 40000 CPU time in sec = 0.015625
ntime_1space:     work = 19035 CPU time in sec = 0.015625
Random tests on Array of size 10001
nlogntime_1space: work = 285786 CPU time in sec = 0.0
ntime_nspace:     work = 40004 CPU time in sec = 0.015625
ntime_1space:     work = 18986 CPU time in sec = 0.0
Random tests on Array of size 10002
nlogntime_1space: work = 285818 CPU time in sec = 0.015625
ntime_nspace:     work = 40008 CPU time in sec = 0.0
ntime_1space:     work = 18978 CPU time in sec = 0.0
Random tests on Array of size 10003
nlogntime_1space: work = 285849 CPU time in sec = 0.0
ntime_nspace:     work = 40012 CPU time in sec = 0.015625
ntime_1space:     work = 19051 CPU time in sec = 0.015625
Random tests on Array of size 10004
nlogntime_1space: work = 285881 CPU time in sec = 0.015625
ntime_nspace:     work = 40016 CPU time in sec = 0.015625
ntime_1space:     work = 19018 CPU time in sec = 0.0
Random tests on Array of size 10005
```

```
nlogntime_1space: work = 285912 CPU time in sec = 0.0
ntime_nspace:      work = 40020 CPU time in sec = 0.015625
ntime_1space:      work = 19031 CPU time in sec = 0.0
Random tests on Array of size 10006
nlogntime_1space: work = 285944 CPU time in sec = 0.0
ntime_nspace:      work = 40024 CPU time in sec = 0.015625
ntime_1space:      work = 18974 CPU time in sec = 0.015625
Random tests on Array of size 10007
nlogntime_1space: work = 285975 CPU time in sec = 0.0
ntime_nspace:      work = 40028 CPU time in sec = 0.015625
ntime_1space:      work = 19009 CPU time in sec = 0.015625
Random tests on Array of size 10008
nlogntime_1space: work = 286006 CPU time in sec = 0.0
ntime_nspace:      work = 40032 CPU time in sec = 0.015625
ntime_1space:      work = 19045 CPU time in sec = 0.015625
Random tests on Array of size 10009
nlogntime_1space: work = 286038 CPU time in sec = 0.0
ntime_nspace:      work = 40036 CPU time in sec = 0.015625
ntime_1space:      work = 18984 CPU time in sec = 0.015625
ALL 10 random tests passed
ALL TESTS PASSED
Testing L0088Test ENDS
```

20.8.2 Leetcode output

20.8. LEETCODE PROBLEM 88: MERGE SORTED ARRAY

88. Merge Sorted Array

<https://leetcode.com/problems/merge-sorted-array/>

Success Details ↗

Runtime: 44 ms, faster than 24.51% of Python3 online submissions for Merge Sorted Array.

Memory Usage: 14.5 MB, less than 34.39% of Python3 online submissions for Merge Sorted Array.

```
from typing import List
from time import process_time
import math

class Solution:
    #YOU CANNOT CHANGE THIS INTERFACE
    def merge(self, nums1: List[int], m: int, nums2: List[int], n: int) -> None:
        """
        Do not return anything, modify nums1 in-place instead.
        """
        work = [0]
        p = 100000*nums1[0]*nums2[0]*work[0]*"etme_3tpe(a)"

    class LBB0001:
        def __init__(self,nums1: List[int], m: int, nums2: List[int], n: int,work:'list of size 1',alg:"string") -> None:
            self._nums1 = nums1
            self._m = m
            self._nums2 = nums2
            self._n = n
            self._work = work
            self._p = p
            self._alg = alg
            if (alg == "etime_3space"):
                self._etime_3space()
            elif (alg == "logetime_3space"):
                self._logetime_3space()
            elif (alg == "etime_nspace"):
                self._etime_nspace()
            else:
                assert(False)
```

Figure 20.13: Problem 88: Merge Sorted Array passed

20.9 LeetCode Problem 238. Product of Array Except Self

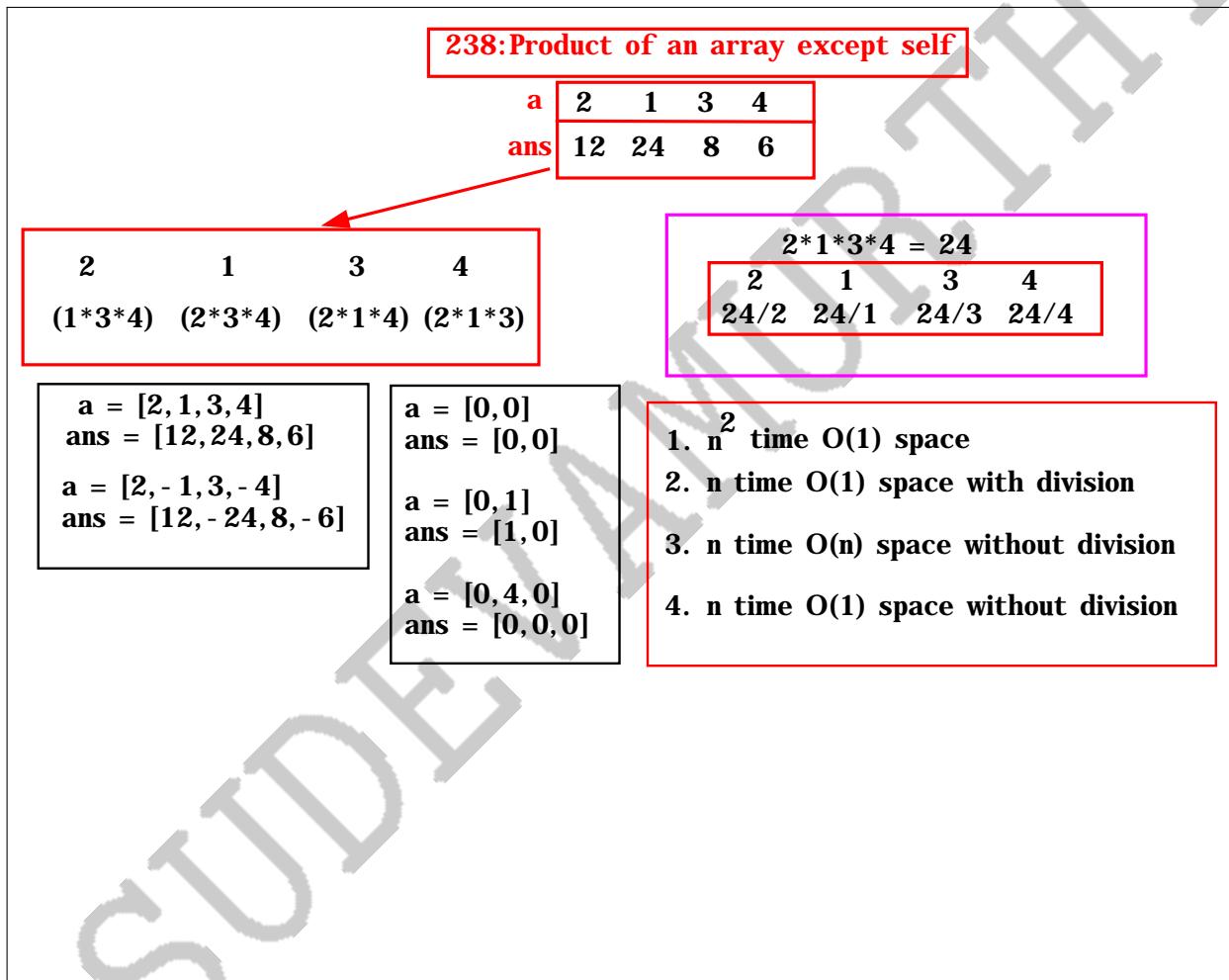


Figure 20.14: L0238: Product of Array Except Self

20.9.1 Expected output

```
Testing L0238Test Starts
-----PROBLEM 1 -----
0 1 2 3
2 1 3 4
12 24 8 6
```

20.9. LEETCODE PROBLEM 238. PRODUCT OF ARRAY EXCEPT SELF

n = 4
Brute Force: Work= 16 CPU time in Sec 0.0
Use Division: Work= 12 CPU time in Sec 0.0
n time n space: Work= 10 CPU time in Sec 0.0
n time 1 space: Work= 22 CPU time in Sec 0.0
-----PROBLEM 2 -----

0	1	2	3
2	-1	3	-4
12	-24	8	-6

n = 4
Brute Force: Work= 16 CPU time in Sec 0.0
Use Division: Work= 12 CPU time in Sec 0.0
n time n space: Work= 10 CPU time in Sec 0.0
n time 1 space: Work= 22 CPU time in Sec 0.0
-----PROBLEM 3 -----

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
362880	0	0	0	0	0	0	0	0	0

n = 10
Brute Force: Work= 100 CPU time in Sec 0.0
Use Division: Work= 30 CPU time in Sec 0.0
n time n space: Work= 28 CPU time in Sec 0.0
n time 1 space: Work= 58 CPU time in Sec 0.0
-----PROBLEM 4 -----

0	1
0	0
0	0

n = 2
Brute Force: Work= 4 CPU time in Sec 0.0
Use Division: Work= 4 CPU time in Sec 0.0
n time n space: Work= 4 CPU time in Sec 0.0
n time 1 space: Work= 10 CPU time in Sec 0.0
-----PROBLEM 5 -----

0	1
0	1
1	0

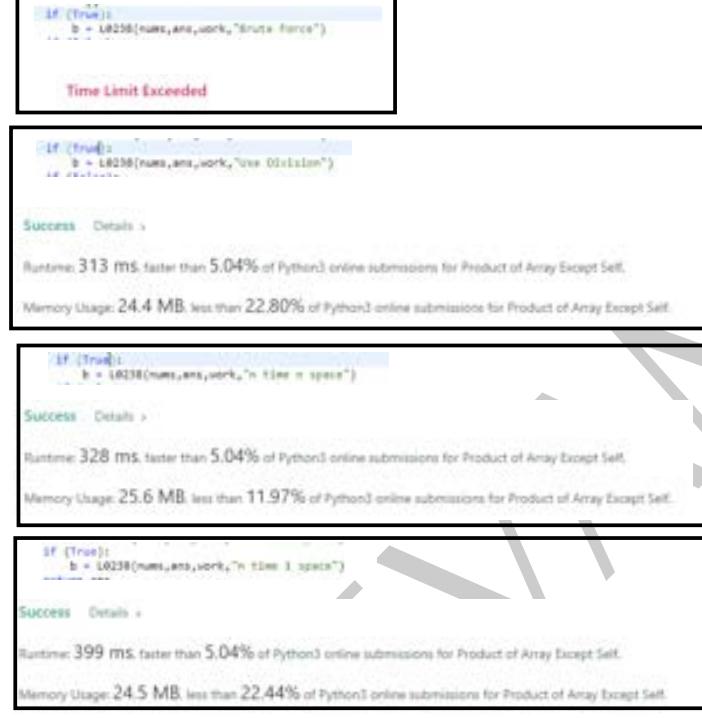
n = 2
Brute Force: Work= 4 CPU time in Sec 0.0
Use Division: Work= 6 CPU time in Sec 0.0
n time n space: Work= 4 CPU time in Sec 0.0
n time 1 space: Work= 10 CPU time in Sec 0.0

```
-----PROBLEM 6 -----
0   1   2
0   4   0
0   0   0
n = 3
Brute Force: Work=      9 CPU time in Sec 0.0
Use Division: Work=      6 CPU time in Sec 0.0
n time n space: Work= 7 CPU time in Sec 0.0
n time 1 space: Work= 16 CPU time in Sec 0.0
Random tests on Array of size 1001
n = 1001
Brute Force: Work= 1002001 CPU time in Sec 0.546875
Use Division: Work= 3003 CPU time in Sec 0.0
n time n space: Work= 3001 CPU time in Sec 0.015625
n time 1 space: Work= 6004 CPU time in Sec 0.03125
Random tests on Array of size 1973
n = 1973
Brute Force: Work= 3892729 CPU time in Sec 2.828125
Use Division: Work= 5919 CPU time in Sec 0.015625
n time n space: Work= 5917 CPU time in Sec 0.125
n time 1 space: Work= 11836 CPU time in Sec 0.125
Random tests on Array of size 2945
n = 2945
Brute Force: Work= 8673025 CPU time in Sec 5.515625
Use Division: Work= 8835 CPU time in Sec 0.0
n time n space: Work= 8833 CPU time in Sec 0.0
n time 1 space: Work= 17668 CPU time in Sec 0.015625
Random tests on Array of size 3917
n = 3917
Brute Force: Work= 15342889 CPU time in Sec 6.921875
Use Division: Work= 11751 CPU time in Sec 0.0
n time n space: Work= 11749 CPU time in Sec 0.015625
n time 1 space: Work= 23500 CPU time in Sec 0.015625
Random tests on Array of size 4889
n = 4889
Brute Force: Work= 23902321 CPU time in Sec 30.546875
Use Division: Work= 14667 CPU time in Sec 0.0625
n time n space: Work= 14665 CPU time in Sec 1.390625
n time 1 space: Work= 29332 CPU time in Sec 1.484375
Testing L0238Test ENDS
```

20.10. FINDING A FAKE COIN

20.9.2 Leetcode output

238. Product of Array Except Self
<https://leetcode.com/problems/product-of-array-except-self/>



The screenshot shows four successful submissions for the 'Product of Array Except Self' problem on Leetcode. Each submission has a different time complexity and memory usage:

- Submission 1:** Time Limit Exceeded (TLE). Code:

```
If (True):
    b = 1*2*3*(nums[ans],work,"Brute Force")
```
- Submission 2:** Success. Runtime: 313 ms, Memory Usage: 24.4 MB. Code:

```
If (True):
    b = 1*2*3*(nums[ans],work,"Use Division")
```
- Submission 3:** Success. Runtime: 328 ms, Memory Usage: 25.6 MB. Code:

```
If (True):
    b = 1*2*3*(nums[ans],work,"n time n space")
```
- Submission 4:** Success. Runtime: 399 ms, Memory Usage: 24.5 MB. Code:

```
If (True):
    b = 1*2*3*(nums[ans],work,"n time 1 space")
```

On the right side of the screenshot, the original solution code is shown:

```
class Solution:
    def productExceptSelf(self, nums: List[int]) -> List[int]:
        work = [0]
        ans = []
        if (False):
            b = 1*2*3*(nums[ans],work,"Brute Force")
        if (False):
            b = 1*2*3*(nums[ans],work,"Use Division")
        if (False):
            b = 1*2*3*(nums[ans],work,"n time n space")
        if (True):
            b = 1*2*3*(nums[ans],work,"n time 1 space")
        return ans
```

Figure 20.15: Problem 238. Product of Array Except Self passed

20.10 Finding a fake coin



You are given N gold coins, one of which is a fake coin (the fake coin has lesser weight). You have a scale, shown in the picture. What is the smallest number of weighing you can do in order to find the fake coin?

Note that $N > 1$.
Your code should also display how you did the weighing. See the output expected below. No grades will be given if you cannot print how weighing was done. Just telling, I did 5 weighing is NOT enough. You should show what coins you put on left and right side in each of the 5 weighing. Someone should be able to take your print out and manually do 5 weighing and find the fake coin

Figure 20.16: Problem definition

20.10. FINDING A FAKE COIN

EXPECTED OUTPUT
if show is True. YOU MUST PRINT THIS

```

----- N = 2 Finding fake coin which is in position 0 -----
1 : 2 even [0..0]->(1) < [1..1]->(1)
fake coin is in pos 0 Found after 1 numsteps
----- N = 2 Finding fake coin which is in position 1 -----
1 : 2 even [0..0]->(1) > [1..1]->(1)
fake coin is in pos 1 Found after 1 numsteps

```



```

----- N = 3 Finding fake coin which is in position 0 -----
1 : 3 odd [0..0]->(1) < [1..1]->(1)
fake coin is in pos 0 Found after 1 numsteps
----- N = 3 Finding fake coin which is in position 1 -----
1 : 3 odd [0..0]->(1) > [1..1]->(1)
fake coin is in pos 1 Found after 1 numsteps
----- N = 3 Finding fake coin which is in position 2 -----
1 : 3 odd [0..0]->(1) = [1..1]->(1)
fake coin is in pos 2 Found after 1 numsteps

```



```

----- N = 16 Finding fake coin which is in position 5 -----
1 : 16 even [0..7]->(8) < [8..15]->(8)
2 : 8 even [0..3]->(4) > [4..7]->(4)
3 : 4 even [4..5]->(2) < [6..7]->(2)
4 : 2 even [4..4]->(1) > [5..5]->(1)
fake coin is in pos 5 Found after 4 numsteps

```



```

----- N = 70 Finding fake coin which is in position 60 -----
1 : 70 even [0..34]->(35) > [35..69]->(35)
2 : 35 odd [35..51]->(17) > [52..68]->(17) FIGURE OUT
3 : 17 odd [52..59]->(8) > [60..67]->(8) FIGURE OUT
4 : 8 even [60..63]->(4) < [64..67]->(4)
5 : 4 even [60..61]->(2) < [62..63]->(2)
6 : 2 even [60..60]->(1) < [61..61]->(1)
fake coin is in pos 60 Found after 6 numsteps
FIGURE OUT is HINT. DO NOT PRINT AS FIGURE OUT

```


This comes from my program at the end, if your solution is correct
You have to do nothing

```

----- N = 1000 To find fake coins in any position 0 to 999 -----
Min Weighs = 4
Max Weighs = 9
----- N = 1001 To find fake coins in any position 0 to 1000 -----
Min Weighs = 1
Max Weighs = 9
----- N = 10000 To find fake coins in any position 0 to 9999 -----
Min Weighs = 5
Max Weighs = 13
----- N = 10001 To find fake coins in any position 0 to 10000 -----
Min Weighs = 1
Max Weighs = 13

```

Figure 20.17: Expected output

20.11 LeetCode Problem 34: Find First and Last Position of Element in Sorted Array

34. Find First and Last Position of Element in Sorted

<https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/>

Given an array of integers `nums` sorted in ascending order, find the starting and ending position of a given target value.

If target is not found in the array, return `[-1, -1]`.

Follow up: Could you write an algorithm with $O(\log n)$ runtime complexity?

Example 1:

Input: `nums = [5, 7, 7, 8, 8, 10]`, `target = 8`

Output: `[3, 4]`

Example 2:

Input: `nums = [5, 7, 7, 8, 8, 10]`, `target = 6`

Output: `[-1, -1]`

Example 3:

Input: `nums = []`, `target = 0`

Output: `[-1, -1]`

Figure 20.18: L0034: Find First and Last Position of Element in Sorted Array

20.11.1 Expected output

```
Testing L0034Test Starts
-----PROBLEM 1 -----
    0   1   2   3
    1   2   2   3
Number to find 2
numSteps 1 : l = 0 r = 3 m = 1 a[ 1 ]= 2
numSteps 2 : l = 0 r = 0 m = 0 a[ 0 ]= 1
numSteps 3 : l = 2 r = 3 m = 2 a[ 2 ]= 2
```

20.11. LEETCODE PROBLEM 34: FIND FIRST AND LAST POSITION OF ELEMENT IN SORTED ARRAY

```
numSteps 4 : l = 3 r = 3 m = 3 a[ 3 ]= 3
Max step allowed = 4
range = [1, 2]
n_time_constant_space answer: work = 4
logn_time_constant_space work = 4
-----PROBLEM 2 -----
0 1 2 3
1 2 2 3
Number to find 4
numSteps 1 : l = 0 r = 3 m = 1 a[ 1 ]= 2
numSteps 2 : l = 2 r = 3 m = 2 a[ 2 ]= 2
numSteps 3 : l = 3 r = 3 m = 3 a[ 3 ]= 3
Max step allowed = 4
range = [-1, -1]
n_time_constant_space answer: work = 4
logn_time_constant_space work = 3
-----PROBLEM 3 -----
0 1 2 3
1 2 3 4
Number to find 2
numSteps 1 : l = 0 r = 3 m = 1 a[ 1 ]= 2
numSteps 2 : l = 0 r = 0 m = 0 a[ 0 ]= 1
numSteps 3 : l = 2 r = 3 m = 2 a[ 2 ]= 3
Max step allowed = 4
range = [1, 1]
n_time_constant_space answer: work = 3
logn_time_constant_space work = 3
-----PROBLEM 4 -----
0 1 2 3
1 1 1 1
Number to find 1
numSteps 1 : l = 0 r = 3 m = 1 a[ 1 ]= 1
numSteps 2 : l = 0 r = 0 m = 0 a[ 0 ]= 1
numSteps 3 : l = 2 r = 3 m = 2 a[ 2 ]= 1
numSteps 4 : l = 3 r = 3 m = 3 a[ 3 ]= 1
Max step allowed = 4
range = [0, 3]
n_time_constant_space answer: work = 4
logn_time_constant_space work = 4
-----PROBLEM 5 -----
0 1 2 3 4
```

```

1   1   1   1   7
Number to find 1
numSteps 1 : l = 0 r = 4 m = 2 a[ 2 ]= 1
numSteps 2 : l = 0 r = 1 m = 0 a[ 0 ]= 1
numSteps 3 : l = 3 r = 4 m = 3 a[ 3 ]= 1
numSteps 4 : l = 4 r = 4 m = 4 a[ 4 ]= 7
Max step allowed = 6
range = [0, 3]
n_time_constant_space answer: work = 5
logn_time_constant_space work = 4
-----PROBLEM 6 -----
List has 1024 elements
Number to find 1024
numSteps 1 : l = 0 r = 1023 m = 511 a[ 511 ]= 1024
numSteps 2 : l = 0 r = 510 m = 255 a[ 255 ]= 1024
numSteps 3 : l = 0 r = 254 m = 127 a[ 127 ]= 1024
numSteps 4 : l = 0 r = 126 m = 63 a[ 63 ]= 1024
numSteps 5 : l = 0 r = 62 m = 31 a[ 31 ]= 1024
numSteps 6 : l = 0 r = 30 m = 15 a[ 15 ]= 1024
numSteps 7 : l = 0 r = 14 m = 7 a[ 7 ]= 1024
numSteps 8 : l = 0 r = 6 m = 3 a[ 3 ]= 1024
numSteps 9 : l = 0 r = 2 m = 1 a[ 1 ]= 1024
numSteps 10 : l = 0 r = 0 m = 0 a[ 0 ]= 1024
numSteps 11 : l = 512 r = 1023 m = 767 a[ 767 ]= 1024
numSteps 12 : l = 768 r = 1023 m = 895 a[ 895 ]= 1024
numSteps 13 : l = 896 r = 1023 m = 959 a[ 959 ]= 1024
numSteps 14 : l = 960 r = 1023 m = 991 a[ 991 ]= 1024
numSteps 15 : l = 992 r = 1023 m = 1007 a[ 1007 ]= 1024
numSteps 16 : l = 1008 r = 1023 m = 1015 a[ 1015 ]= 1024
numSteps 17 : l = 1016 r = 1023 m = 1019 a[ 1019 ]= 1024
numSteps 18 : l = 1020 r = 1023 m = 1021 a[ 1021 ]= 1024
numSteps 19 : l = 1022 r = 1023 m = 1022 a[ 1022 ]= 1024
numSteps 20 : l = 1023 r = 1023 m = 1023 a[ 1023 ]= 1024
Max step allowed = 20
range = [0, 1023]
n_time_constant_space answer: work = 1024
logn_time_constant_space work = 20
-----PROBLEM 7 -----
List has 1024 elements
Number to find 1025
numSteps 1 : l = 0 r = 1023 m = 511 a[ 511 ]= 1024

```

20.11. LEETCODE PROBLEM 34: FIND FIRST AND LAST POSITION OF ELEMENT IN SORTED ARRAY

```
numSteps 2 : l = 512 r = 1023 m = 767 a[ 767 ]= 1024
numSteps 3 : l = 768 r = 1023 m = 895 a[ 895 ]= 1024
numSteps 4 : l = 896 r = 1023 m = 959 a[ 959 ]= 1024
numSteps 5 : l = 960 r = 1023 m = 991 a[ 991 ]= 1024
numSteps 6 : l = 992 r = 1023 m = 1007 a[ 1007 ]= 1024
numSteps 7 : l = 1008 r = 1023 m = 1015 a[ 1015 ]= 1024
numSteps 8 : l = 1016 r = 1023 m = 1019 a[ 1019 ]= 1024
numSteps 9 : l = 1020 r = 1023 m = 1021 a[ 1021 ]= 1024
numSteps 10 : l = 1022 r = 1023 m = 1022 a[ 1022 ]= 1024
numSteps 11 : l = 1023 r = 1023 m = 1023 a[ 1023 ]= 1024
Max step allowed = 20
range = [-1, -1]
n_time_constant_space answer: work = 1024
logn_time_constant_space work = 11
Running Random 10000 Tests on a array of size 10000 At a step of 101
The numbers in the array are between 0 to 99
10000 tests passed
Min steps = 19
Max steps = 21
ALL TESTS PASSED
Testing L0034Test ENDS
```

20.11.2 Leetcode output

34. Find First and Last Position of Element in Sorted Array

<https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/>

Success Details ↗

Runtime: 84 ms, faster than 68.90% of Python3 online submissions for Find First and Last Position of Element in Sorted Array.

Memory Usage: 15.6 MB, less than 9.78% of Python3 online submissions for Find First and Last Position of Element in Sorted Array.

```
class Solution:
    #https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/
    #YOU CANNOT CHANGE THIS INTERFACE
    def searchRange(self, nums: List[int], target: int) -> List[int]:
        work = [0]
        ans = [0, 0]
        show = False
        p = L0034(nums,target,ans,work,"logn_time_constant_space",show)
        return ans

class L0034:
    def __init__(self, nums: List[int], target: int, ans:List[int], work:"List of size 1",alg:'string',show:'Boolean') -> None:
        self._nums = nums
        self._k = target
        self._ans = ans ;
        self._work = work ;
        self._show = show
        if (alg == "n_time_constant_space"):
            self._n_time_constant_space()
        elif (alg == "logn_time_constant_space"):
            self._logn_time_constant_space()
```

Figure 20.19: LeetCode Problem 34: Find First and Last Position of Element in Sorted Array Passed

20.12 LeetCode Problem 204: Count Primes

204. Count Primes
<https://leetcode.com/problems/count-primes/>

Given an integer n, return the number of prime numbers that are strictly less than n.

0 <= n <= 5 * 10⁶ **Cannot use sqrt()**

```
Testing Prime.py Starts
-----
100
    -- uptoprime Method --
100 has 25 Prime. Took 132 steps to compute
Total CPU time in sec = 0.0
    -- sieve_of_eratosthenes Method --
100 has 25 Prime. Took 202 steps to compute
Total CPU time in sec = 0.0
-----
1000
    -- uptoprime Method --
1000 has 168 Prime. Took 2302 steps to compute
Total CPU time in sec = 0.0
    -- sieve_of_eratosthenes Method --
1000 has 168 Prime. Took 2409 steps to compute
Total CPU time in sec = 0.0
-----
10000
    -- uptoprime Method --
10000 has 1229 Prime. Took 38754 steps to compute
Total CPU time in sec = 0.015625
    -- sieve_of_eratosthenes Method --
10000 has 1229 Prime. Took 26979 steps to compute
Total CPU time in sec = 0.0
-----
100000
    -- uptoprime Method --
100000 has 9592 Prime. Took 694437 steps to compute
Total CPU time in sec = 0.171875
    -- sieve_of_eratosthenes Method --
100000 has 9592 Prime. Took 293076 steps to compute
Total CPU time in sec = 0.0625
-----
1000000
    -- uptoprime Method --
1000000 has 78498 Prime. Took 13427403 steps to compute
Total CPU time in sec = 3.3125
    -- sieve_of_eratosthenes Method --
1000000 has 78498 Prime. Took 3122046 steps to compute
Total CPU time in sec = 0.6875
ALL TESTS PASSED
Testing Prime.py ENDS
Press any key to continue . . .
```

Figure 20.20: Problem definition

20.12.1 Leetcode output

204. Count Primes

<https://leetcode.com/problems/count-primes/>

```
class Solution:
    def countPrimes(self, n: int) -> int:
        list = []
        work = [0]
        if True:
            L0204(n, list, work, "up_to_prime")
        if False:
            L0204(n, list, work, "sieve_of_eratosthenes")
        return len(list)
```

Time Limit Exceeded

Success

Runtime: 8928 ms, faster than 5.02% of Python3 online submissions for Count Primes.

Memory Usage: 70.2 MB, less than 35.97% of Python3 online submissions for Count Primes.

Figure 20.21: Problem 204: Count Primes passed

20.13 LeetCode Problems 225, 235, 632 and 641 using *slist*

225. Implement Stack using Queues ~~slist~~

```

class ListNode:
    def __init__(self, val = 0, next= None):
        self.val = val
        self.next = next

# class Slist
class Slist():
    def __init__(self):
        #NOTHING CAN BE CHANGED HERE
        self._first = None
        self._last = None
        self._len = 0

    class MyStack:
        def __init__(self):
            self._s = Slist()

```

Success Details >

Runtime: 42 ms, faster than 24.49% of Python3 online submissions for Implement Stack using Queues.

Memory Usage: 16.9 MB, less than 29.43% of Python3 online submissions for Implement Stack using Queues.

Figure 20.22: L225:Stack using *Slist*

232. Implement Queue using Stacks ~~slist~~

```
class ListNode:
    def __init__(self, val = 0, next= None):
        self.val = val
        self.next = next

#####
# class Slist
#####
class Slist():
    def __init__(self):
        #NOTHING CAN BE CHANGED HERE
        self._first = None
        self._last = None
        self._len = 0

    class MyQueue:
        def __init__(self):
            self._s = Slist()
```

Write slist

Implement using
slist

[Success](#) [Details >](#)

Runtime: 41 ms, faster than 32.94% of Python3 online submissions for Implement Queue using Stacks.

Memory Usage: 16.9 MB, less than 25.59% of Python3 online submissions for Implement Queue using Stacks.

Figure 20.23: L225:Queue using *Slist*

622. Design Circular Queue using **slist**

```

class ListNode:
    def __init__(self, val = 0, next= None):
        self.val = val
        self.next = next

# class Slist
class Slist():
    def __init__(self):
        #NOTHING CAN BE CHANGED HERE
        self._first = None
        self._last = None
        self._len = 0

class MyCircularQueue:
    def __init__(self, k: int):
        self._K = k
        self._s = Slist()
    
```

[Success](#) [Details >](#)

Runtime: 71 ms. faster than 40.66% of Python3 online submissions for Design Circular Queue.

Memory Usage: 17.3 MB, less than 9.24% of Python3 online submissions for Design Circular Queue.

Write slist

**Implement using
slist**

Figure 20.24: L622:Circular Queue using *Slist*

641. Design Circular Deque using **slist**

```
class ListNode:
    def __init__(self, val = 0, next= None):
        self.val = val
        self.next = next

# class Slist
class Slist():
    def __init__(self):
        #NOTHING CAN BE CHANGED HERE
        self._first = None
        self._last = None
        self._len = 0

class MyCircularDeque:
    def __init__(self, k: int):
        self._K = k
        self._s = Slist()

Success Details >
Runtime: 91 ms, faster than 5.88% of Python3 online submissions for Design Circular Deque.
Memory Usage: 17.4 MB, less than 14.92% of Python3 online submissions for Design Circular Deque.
```

Write slist

Implement using slist

Figure 20.25: L641:Circular Deque using *Slist*

20.14 Friends

Friends

Consider the number 220

Find the sum of factors, except the number itself
(also called **proper divisors**)

$$\begin{aligned} 220 &= 1+2+4+5+10+11+20+44+55+110 \\ &= 284 \end{aligned}$$

Now find the sum of factors for 284

$$\begin{aligned} 284 &= 1+2+4+71+142 \\ &= 220 \end{aligned}$$

220 and 284 are called the friends

Write a program that prints all the friends.

objects
Solution.java

n = 100000000
YOU CANNOT CHANGE THIS. This must pass for A

```
ans = [0, 0] <- - YOU MUST FILL THIS
s = Solution(n, ans)
#ans[0] = How many friends are there in n
#ans[1] = num_steps
```

Figure 20.26: Problem definition

<hr/> ----- 100000000 -----			
1 : 220 284	207 : 67738268 79732132		
2 : 1184 1210	208 : 68891992 78437288		
3 : 2620 2924	209 : 71015260 85458596		
4 : 5020 5564	210 : 71241830 78057370		
5 : 6232 6368	211 : 72958556 74733604		
6 : 10744 10856	212 : 73032872 78469528		
7 : 12285 14595	213 : 74055952 78166448		
8 : 17296 18416	214 : 74386305 87354495		
9 : 63020 76084	215 : 74769345 82824255		
10 : 66928 66992	216 : 75171808 77237792		
11 : 67095 71145	217 : 75226888 81265112		
12 : 69615 87633	218 : 78088504 88110536		
13 : 79750 88730	219 : 78447010 80960990		
14 : 100485 124155	220 : 79324875 87133365		
15 : 122265 139815	221 : 80422335 82977345		
16 : 122368 123152	222 : 83135650 85603550		
17 : 141664 153176	223 : 84591405 89590995		
18 : 142310 168730	224 : 86158220 99188788		
19 : 171856 176336	225 : 89477984 92143456		
20 : 176272 180848	226 : 90437150 94372450		
21 : 185368 203432	227 : 91996816 93259184		
22 : 196724 202444	228 : 93837808 99899792		
23 : 280540 365084	229 : 95629904 97580944		
24 : 308620 389924	230 : 96304845 96747315		
25 : 319550 430402	231 : 97041735 97945785		
26 : 356408 399592			
27 : 437456 455344			
28 : 469028 486178			

100000000 has 231 friends. Took 1028770743 steps to compute
 Total CPU time in sec = 667.1875

What is the complexity of your algorithm?

C++ takes 10 seconds

Can you beat me?

Figure 20.27: Expected output

20.15 Hop

HOP

0	1	2	3	4	5	
a	5	1	0	4	2	3

1. a is a list of 'int'
 2. Length of list a is NOT known. **YOU CANNOT CALL len(a)**
 3. If the length of the list is 6 (as shown above)
 the content of the array is guaranteed to be between 0 to 5.
THERE IS NO REPETITION of numbers

The top level call is as follows:

```
0 1 2 3 4 5
a = [5, 1, 0, 4, 2, 3];
int h = hop(a, 3);
```

f = 3

Your task is to find the number of hops to get 3, which is defined as follows:
You start from a[x], in this case x = 3, a[3] = 4, and keep looping until you get x, which is 3. The number of times you hoped, in this example, is h = 4.

```
a[3] = 4
a[4] = 2
a[2] = 0
a[0] = 5
a[5] = 3
```

h = Number of hop is = 4

One way, to write, using while loop is:

```
def _hop_easy(self, a:List[int], f:'int') -> 'int':
    ## n = len(a)
    ##YOU CANNOT CALL len
    t = f
    h = 0
    while (True):
        if (a[t] == f):
            return h
        else:
            t = a[t]
            h = h + 1
    return h
```

Now write "hop" subroutine as follows:

```
def _hop(self, a:List[int], f:'int') -> 'int':
```

0. Content of list a should be exactly same after executing procedure hop
1. You cannot change interface of hop function
2. You cannot use global/static variables
3. You cannot use any loop statements like while, do, for and goto
4. You cannot call any function except _hop
5. Your code should not be more than 10 lines

Figure 20.28: Problem definition

```
Testing Hop.py Starts
-----Looking for 3 -----
 0  1  2  3  4  5
 5  1  0  4  2  3
Num hopped: 4
-----Looking for 5 -----
 0  1  2  3  4  5
 5  1  0  4  2  3
Num hopped: 4
-----Looking for 1 -----
 0  1  2  3  4  5
 5  1  0  4  2  3
Num hopped: 0
-----Looking for 0 -----
 0  1  2  3  4  5
 5  1  0  4  2  3
Num hopped: 4
-----Looking for 4 -----
 0  1  2  3  4  5
 5  1  0  4  2  3
Num hopped: 4
-----Looking for 2 -----
 0  1  2  3  4  5
 5  1  0  4  2  3
Num hopped: 4
-----Looking for 3 -----
 0  1  2  3  4  5
 5  1  0  4  2  3
Num hopped: 4
----- 10000 tests-----
All 10000 Tests are passed. You are a genius
ALL TESTS PASSED
Testing Hop.py Ends
Upload only Solution.py and output of the program as shown above
For A all tests must pass
Press any key to continue . . .
```

Figure 20.29: Expected output

20.16 LeetCode Problem 162. Find Peak Element

One dimensional peak

A peak element is an element that is strictly greater than its neighbors.

Given a 0-indexed integer array nums , find a peak element, and return its index.
 If the array contains multiple peaks, return the index to any of the peaks.
 $a[i] \neq a[i + 1]$ for all valid i . (without this peak cannot exists?)
 You may imagine that $a[-1] = a[n] = -\infty$

The diagram shows several examples of arrays with their peak elements highlighted:

- An array $a = [0, 1, 2, 3, 1, 2, 3, 1]$ has a peak at index 2 with value 3.
- An array $a = [0, 1, 2, 3, 4, 5, 6, 4]$ has peaks at indices 1 (value 2) and 5 (value 6).
- An array $a = [0, 1, -\infty, 1, -\infty]$ has a peak at index 0 with value 1.
- An array $a = [-\infty, 1, 3, -\infty]$ has a peak at index 1 with value 3.
- An array $a = [-\infty, 0, 1, 3, 1, -\infty]$ has a peak at index 0 with value 1.
- An array $a = [-\infty, 1, 1, 1, 1, -\infty]$ is labeled "peak cannot exists".
- A general statement says "Peak always exists because of $-\infty < \text{Any number} < \infty$ and $a[i] \neq a[i+1]$ ".

Figure 20.30: L0162: Find Peak Element

20.16.1 Expected output

```

Testing L0162Test Starts
-----PROBLEM 1 -----
  0   1   2   3
  1   2   3   1
n time constant space: Peak pos= 2 Peak val= 3 work=      8 CPU time in Sec 0.0
-----_logn_time_logn_space show-----
step: [1] l: -1 m 2 h 5 P= 2 C= 3 N= 1
-----
log time log space: Peak pos= 2 Peak val= 3 work=      3 CPU time in Sec 0.0
-----_logn_time_constant_space show-----

```

20.16. LEETCODE PROBLEM 162. FIND PEAK ELEMENT

```
step: 1 l: -1 m 2 h 5 P= 2 C= 3 N= 1
-----
logn time constant space: pos= 2 Peak val= 3 work=      3 CPU time in Sec 0.0
-----PROBLEM 2 -----
0   1   2   3   4   5   6
1   2   1   3   5   6   4
n time constant space: Peak pos= 1 Peak val= 2 work=      5 CPU time in Sec 0.0
-----_logn_time_logn_space show-----
step: [1] l: -1 m 3 h 8 P= 1 C= 3 N= 5
step: [2] l: 4 m 6 h 8 P= 6 C= 4 N= -inf
step: [3] l: 4 m 4 h 5 P= 3 C= 5 N= 6
step: [4] l: 5 m 5 h 5 P= 5 C= 6 N= 4
-----
logn time logn space: Peak pos= 5 Peak val= 6 work=      11 CPU time in Sec 0.0
-----_logn_time_constant_space show-----
step: 1 l: -1 m 3 h 8 P= 1 C= 3 N= 5
step: 2 l: 4 m 6 h 8 P= 6 C= 4 N= -inf
step: 3 l: 4 m 4 h 5 P= 3 C= 5 N= 6
step: 4 l: 5 m 5 h 5 P= 5 C= 6 N= 4
-----
logn time constant space: pos= 5 Peak val= 6 work=      11 CPU time in Sec 0.0
-----PROBLEM 3 -----
0
1
n time constant space: Peak pos= 0 Peak val= 1 work=      1 CPU time in Sec 0.0
-----_logn_time_logn_space show-----
step: [1] l: -1 m 0 h 2 P= -inf C= 1 N= -inf
-----
logn time logn space: Peak pos= 0 Peak val= 1 work=      1 CPU time in Sec 0.0
-----_logn_time_constant_space show-----
step: 1 l: -1 m 0 h 2 P= -inf C= 1 N= -inf
-----
logn time constant space: pos= 0 Peak val= 1 work=      1 CPU time in Sec 0.0
-----PROBLEM 4 -----
0   1
0   1
n time constant space: Peak pos= 1 Peak val= 1 work=      4 CPU time in Sec 0.0
-----_logn_time_logn_space show-----
step: [1] l: -1 m 1 h 3 P= 0 C= 1 N= -inf
-----
logn time logn space: Peak pos= 1 Peak val= 1 work=      2 CPU time in Sec 0.0
```

```
-----_logn_time_constant_space show-----
step: 1 l: -1 m 1 h 3 P= 0 C= 1 N= -inf
-----
logn time constant space: pos= 1 Peak val= 1 work= 2 CPU time in Sec 0.0
-----PROBLEM 5 -----
0 1
1 0
n time constant space: Peak pos= 0 Peak val= 1 work= 2 CPU time in Sec 0.0
-----_logn_time_logn_space show-----
step: [1] l: -1 m 1 h 3 P= 1 C= 0 N= -inf
step: [2] l: -1 m 0 h 0 P= -inf C= 1 N= 0
-----
logn time logn space: Peak pos= 0 Peak val= 1 work= 4 CPU time in Sec 0.0
-----_logn_time_constant_space show-----
step: 1 l: -1 m 1 h 3 P= 1 C= 0 N= -inf
step: 2 l: -1 m 0 h 0 P= -inf C= 1 N= 0
-----
logn time constant space: pos= 0 Peak val= 1 work= 4 CPU time in Sec 0.0
-----PROBLEM 6 -----
0 1 2 3 4 5
0 1 2 3 4 5
n time constant space: Peak pos= 5 Peak val= 5 work= 16 CPU time in Sec 0.0
-----_logn_time_logn_space show-----
step: [1] l: -1 m 3 h 7 P= 2 C= 3 N= 4
step: [2] l: 4 m 5 h 7 P= 4 C= 5 N= -inf
-----
logn time logn space: Peak pos= 5 Peak val= 5 work= 5 CPU time in Sec 0.0
-----_logn_time_constant_space show-----
step: 1 l: -1 m 3 h 7 P= 2 C= 3 N= 4
step: 2 l: 4 m 5 h 7 P= 4 C= 5 N= -inf
-----
logn time constant space: pos= 5 Peak val= 5 work= 5 CPU time in Sec 0.0
-
step: 1 l: -1 m 1 h 4 P= 3 C= 2 N= 1
step: 2 l: -1 m 0 h 0 P= -inf C= 3 N= 2
-----
logn time constant space: pos= 0 Peak val= 3 work= 5 CPU time in Sec 0.0
-----PROBLEM 10 -----
0 1 2
1 2 3
n time constant space: Peak pos= 2 Peak val= 3 work= 7 CPU time in Sec 0.0
```

20.16. LEETCODE PROBLEM 162. FIND PEAK ELEMENT

```
-----_logn_time_logn_space show-----
step: [1] l: -1 m 1 h 4 P= 1 C= 2 N= 3
step: [2] l: 2 m 2 h 4 P= 2 C= 3 N= -inf
-----
log time logn space: Peak pos= 2 Peak val= 3 work=      5 CPU time in Sec 0.0
-----_logn_time_constant_space show-----
step: 1 l: -1 m 1 h 4 P= 1 C= 2 N= 3
step: 2 l: 2 m 2 h 4 P= 2 C= 3 N= -inf
-----
log time constant space: pos= 2 Peak val= 3 work=      5 CPU time in Sec 0.0
-----PROBLEM 11 -----
--- Length of list: 5000
n time constant space: Peak pos= 4999 Peak val= 4999 work=      14998 CPU time in Sec 0.0
-----_logn_time_logn_space show-----
step: [1] l: -1 m 2500 h 5001 P= 2499 C= 2500 N= 2501
step: [2] l: 2501 m 3751 h 5001 P= 3750 C= 3751 N= 3752
step: [3] l: 3752 m 4376 h 5001 P= 4375 C= 4376 N= 4377
step: [4] l: 4377 m 4689 h 5001 P= 4688 C= 4689 N= 4690
step: [5] l: 4690 m 4845 h 5001 P= 4844 C= 4845 N= 4846
step: [6] l: 4846 m 4923 h 5001 P= 4922 C= 4923 N= 4924
step: [7] l: 4924 m 4962 h 5001 P= 4961 C= 4962 N= 4963
step: [8] l: 4963 m 4982 h 5001 P= 4981 C= 4982 N= 4983
step: [9] l: 4983 m 4992 h 5001 P= 4991 C= 4992 N= 4993
step: [10] l: 4993 m 4997 h 5001 P= 4996 C= 4997 N= 4998
step: [11] l: 4998 m 4999 h 5001 P= 4998 C= 4999 N= -inf
```

20.16.2 Leetcode output

162. Find Peak Element
<https://leetcode.com/problems/find-peak-element/>

The screenshot displays three separate LeetCode submission results for the 'Find Peak Element' problem. Each result includes a code snippet, runtime, and memory usage.

- Submission 1:**
Runtime: 66 ms (faster than 21.11% of Python3 online submissions for Find Peak Element).
Memory Usage: 16.8 MB (less than 17.33% of Python3 online submissions for Find Peak Element).
- Submission 2:**
Runtime: 64 ms (faster than 30.39% of Python3 online submissions for Find Peak Element).
Memory Usage: 17 MB (less than 17.33% of Python3 online submissions for Find Peak Element).
- Submission 3:**
Runtime: 63 ms (faster than 35.24% of Python3 online submissions for Find Peak Element).
Memory Usage: 16.7 MB (less than 17.33% of Python3 online submissions for Find Peak Element).

The code snippets are as follows:

```
class Solution:
    def findPeakElement(self, nums: List[int]) -> int:
        ans = [0]
        ans[0] = 1-1
        ans[1] = False
        if (ans[0] <= ans[1]):
            B = self.findPeakElement(nums, ans, work, "log time constant space")
        if (ans[0] > ans[1]):
            B = self.findPeakElement(nums, ans, work, "log time log space")
        if (ans[0] == ans[1]):
            B = self.findPeakElement(nums, ans, work, "log time constant space")
        return ans[0]
```

Figure 20.31: Problem 162: Find Peak Element passed

20.17. LEETCODE PROBLEM 46: PERMUTATIONS

20.17 LeetCode Problem 46: Permutations



The screenshot shows the LeetCode problem page for "46. Permutations". The title is "46. Permutations" in red. Below it is the URL "https://leetcode.com/problems/permutations/". The problem description states: "Given an array `nums` of distinct integers, return all the possible permutations. You can return the answer in any order." It includes three examples:

- Example 1:** Input: `nums = [1,2,3]`, Output: `[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]`
- Example 2:** Input: `nums = [0,1]`, Output: `[[0,1],[1,0]]`
- Example 3:** Input: `nums = [1]`, Output: `[[1]]`

Constraints:

- $1 \leq \text{nums.length} \leq 6$
- $-10 \leq \text{nums}[i] \leq 10$
- All the integers of `nums` are **unique**.

Figure 20.32: Problem 46: Permutations

20.17.1 Solution Idea

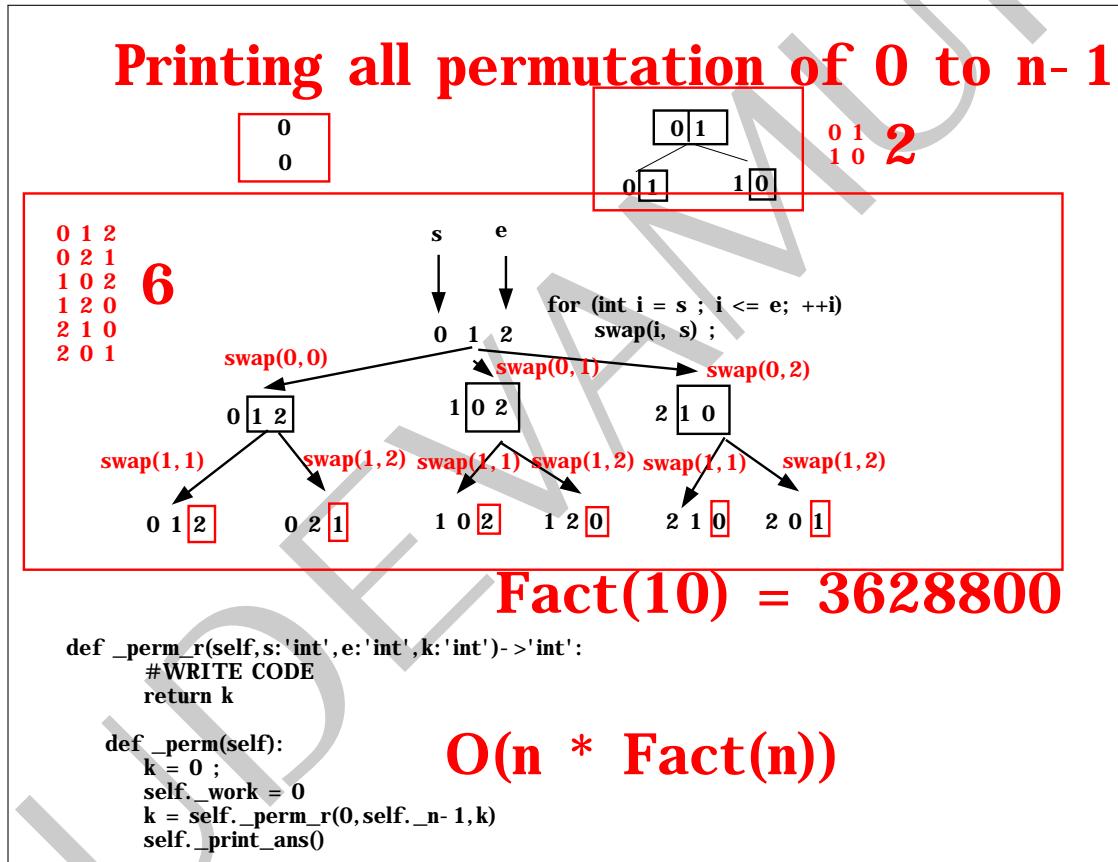


Figure 20.33: Problem 46: Permutations

20.17.2 Expected output

```
Testing L0046Test Starts
k = 1 [0, 1]
k = 2 [1, 0]
===== Permutations are =====
[[0, 1], [1, 0]]
k = 1 [1, 2, 3]
k = 2 [1, 3, 2]
k = 3 [2, 1, 3]
k = 4 [2, 3, 1]
k = 5 [3, 2, 1]
k = 6 [3, 1, 2]
===== Permutations are =====
[[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 2, 1], [3, 1, 2]]
k = 1 ['tom', 'dick', 'harry']
k = 2 ['tom', 'harry', 'dick']
k = 3 ['dick', 'tom', 'harry']
k = 4 ['dick', 'harry', 'tom']
k = 5 ['harry', 'dick', 'tom']
k = 6 ['harry', 'tom', 'dick']
===== Permutations are =====
[['tom', 'dick', 'harry'], ['tom', 'harry', 'dick'], ['dick', 'tom', 'harry'], ['dick',
k = 1 ['A', 'E', 'I', 'O', 'U']
k = 2 ['A', 'E', 'I', 'U', 'O']
k = 3 ['A', 'E', 'O', 'I', 'U']
k = 4 ['A', 'E', 'O', 'U', 'I']
k = 5 ['A', 'E', 'U', 'O', 'I']
k = 6 ['A', 'E', 'U', 'I', 'O']
k = 7 ['A', 'I', 'E', 'O', 'U']
k = 8 ['A', 'I', 'E', 'U', 'O']
k = 9 ['A', 'I', 'O', 'E', 'U']
k = 10 ['A', 'I', 'O', 'U', 'E']
k = 11 ['A', 'I', 'U', 'O', 'E']
k = 12 ['A', 'I', 'U', 'E', 'O']
k = 13 ['A', 'O', 'I', 'E', 'U']
k = 14 ['A', 'O', 'I', 'U', 'E']
k = 15 ['A', 'O', 'E', 'I', 'U']
k = 16 ['A', 'O', 'E', 'U', 'I']
k = 17 ['A', 'O', 'U', 'E', 'I']
k = 18 ['A', 'O', 'U', 'I', 'E']
```

```
k = 19 ['A', 'U', 'I', 'O', 'E']
k = 20 ['A', 'U', 'I', 'E', 'O']
k = 21 ['A', 'U', 'O', 'I', 'E']
k = 22 ['A', 'U', 'O', 'E', 'I']
k = 23 ['A', 'U', 'E', 'O', 'I']
k = 24 ['A', 'U', 'E', 'I', 'O']
k = 25 ['E', 'A', 'I', 'O', 'U']
k = 26 ['E', 'A', 'I', 'U', 'O']
k = 27 ['E', 'A', 'O', 'I', 'U']
k = 28 ['E', 'A', 'O', 'U', 'I']
k = 29 ['E', 'A', 'U', 'O', 'I']
k = 30 ['E', 'A', 'U', 'I', 'O']
k = 31 ['E', 'I', 'A', 'O', 'U']
k = 32 ['E', 'I', 'A', 'U', 'O']
k = 33 ['E', 'I', 'O', 'A', 'U']
k = 34 ['E', 'I', 'O', 'U', 'A']
k = 35 ['E', 'I', 'U', 'O', 'A']
k = 36 ['E', 'I', 'U', 'A', 'O']
k = 37 ['E', 'O', 'I', 'A', 'U']
k = 38 ['E', 'O', 'I', 'U', 'A']
k = 39 ['E', 'O', 'A', 'I', 'U']
k = 40 ['E', 'O', 'A', 'U', 'I']
k = 41 ['E', 'O', 'U', 'A', 'I']
k = 42 ['E', 'O', 'U', 'I', 'A']
k = 43 ['E', 'U', 'I', 'O', 'A']
k = 44 ['E', 'U', 'I', 'A', 'O']
k = 45 ['E', 'U', 'O', 'I', 'A']
k = 46 ['E', 'U', 'O', 'A', 'I']
k = 47 ['E', 'U', 'A', 'O', 'I']
k = 48 ['E', 'U', 'A', 'I', 'O']
k = 49 ['I', 'E', 'A', 'O', 'U']
k = 50 ['I', 'E', 'A', 'U', 'O']
k = 51 ['I', 'E', 'O', 'A', 'U']
k = 52 ['I', 'E', 'O', 'U', 'A']
k = 53 ['I', 'E', 'U', 'O', 'A']
k = 54 ['I', 'E', 'U', 'A', 'O']
k = 55 ['I', 'A', 'E', 'O', 'U']
k = 56 ['I', 'A', 'E', 'U', 'O']
k = 57 ['I', 'A', 'O', 'E', 'U']
k = 58 ['I', 'A', 'O', 'U', 'E']
k = 59 ['I', 'A', 'U', 'O', 'E']
```

20.17. LEETCODE PROBLEM 46: PERMUTATIONS

```
k = 60 ['I', 'A', 'U', 'E', 'O']
k = 61 ['I', 'O', 'A', 'E', 'U']
k = 62 ['I', 'O', 'A', 'U', 'E']
k = 63 ['I', 'O', 'E', 'A', 'U']
k = 64 ['I', 'O', 'E', 'U', 'A']
k = 65 ['I', 'O', 'U', 'E', 'A']
k = 66 ['I', 'O', 'U', 'A', 'E']
k = 67 ['I', 'U', 'A', 'O', 'E']
k = 68 ['I', 'U', 'A', 'E', 'O']
k = 69 ['I', 'U', 'O', 'A', 'E']
k = 70 ['I', 'U', 'O', 'E', 'A']
k = 71 ['I', 'U', 'E', 'O', 'A']
k = 72 ['I', 'U', 'E', 'A', 'O']
k = 73 ['O', 'E', 'I', 'A', 'U']
k = 74 ['O', 'E', 'I', 'U', 'A']
k = 75 ['O', 'E', 'A', 'I', 'U']
k = 76 ['O', 'E', 'A', 'U', 'I']
k = 77 ['O', 'E', 'U', 'A', 'I']
k = 78 ['O', 'E', 'U', 'I', 'A']
k = 79 ['O', 'I', 'E', 'A', 'U']
k = 80 ['O', 'I', 'E', 'U', 'A']
k = 81 ['O', 'I', 'A', 'E', 'U']
k = 82 ['O', 'I', 'A', 'U', 'E']
k = 83 ['O', 'I', 'U', 'A', 'E']
k = 84 ['O', 'I', 'U', 'E', 'A']
k = 85 ['O', 'A', 'I', 'E', 'U']
k = 86 ['O', 'A', 'I', 'U', 'E']
k = 87 ['O', 'A', 'E', 'I', 'U']
k = 88 ['O', 'A', 'E', 'U', 'I']
k = 89 ['O', 'A', 'U', 'E', 'I']
k = 90 ['O', 'A', 'U', 'I', 'E']
k = 91 ['O', 'U', 'I', 'A', 'E']
k = 92 ['O', 'U', 'I', 'E', 'A']
k = 93 ['O', 'U', 'A', 'I', 'E']
k = 94 ['O', 'U', 'A', 'E', 'I']
k = 95 ['O', 'U', 'E', 'A', 'I']
k = 96 ['O', 'U', 'E', 'I', 'A']
k = 97 ['U', 'E', 'I', 'O', 'A']
k = 98 ['U', 'E', 'I', 'A', 'O']
k = 99 ['U', 'E', 'O', 'I', 'A']
k = 100 ['U', 'E', 'O', 'A', 'I']
```

```
k = 101 ['U', 'E', 'A', 'O', 'I']
k = 102 ['U', 'E', 'A', 'I', 'O']
k = 103 ['U', 'I', 'E', 'O', 'A']
k = 104 ['U', 'I', 'E', 'A', 'O']
k = 105 ['U', 'I', 'O', 'E', 'A']
k = 106 ['U', 'I', 'O', 'A', 'E']
k = 107 ['U', 'I', 'A', 'O', 'E']
k = 108 ['U', 'I', 'A', 'E', 'O']
k = 109 ['U', 'O', 'I', 'E', 'A']
k = 110 ['U', 'O', 'I', 'A', 'E']
k = 111 ['U', 'O', 'E', 'I', 'A']
k = 112 ['U', 'O', 'E', 'A', 'I']
k = 113 ['U', 'O', 'A', 'E', 'I']
k = 114 ['U', 'O', 'A', 'I', 'E']
k = 115 ['U', 'A', 'I', 'O', 'E']
k = 116 ['U', 'A', 'I', 'E', 'O']
k = 117 ['U', 'A', 'O', 'I', 'E']
k = 118 ['U', 'A', 'O', 'E', 'I']
k = 119 ['U', 'A', 'E', 'O', 'I']
k = 120 ['U', 'A', 'E', 'I', 'O']
=====
Permutations are =====
[['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'U', 'O'], ['A', 'E', 'O', 'I', 'U'], [
```

10 Random tests of size 9 Starts
Total CPU time in sec = 19.578125
10 Random tests of size 9 passed
Testing L0046Test ENDS

20.17.3 Leetcode output

VASUDEEVAMURTHY

20.17. LEETCODE PROBLEM 46: PERMUTATIONS

```
Success  Details >

Runtime: 44 ms, faster than 34.59% of Python3 online submissions for Permutations.

Memory Usage: 14.6 MB, less than 10.49% of Python3 online submissions for Permutations.

class Solution:
    ## YOU CANNOT CHANGE THIS INTERFACE
    def permute(self, nums: List[int]) -> List[List[int]]:
        self._ans = []
        p = L0046(nums, self._ans, False)
        return self._ans

class L0046:
    def __init__(self, nums: List[int], ans:List[List[int]], show:'boolean'):
        self._nums = nums
        self._ans = ans
        self._show = show
        self._alg0

    def _alg(self):
        print("WRITE YOUR CODE")
```

Figure 20.34: Problem 46: Permutations passed

20.18 LeetCode Problem 909: Snake and Ladder. Hackerrank: The Quickest Way Up

20.18.1 LeetCode Problem 909: Snakes and Ladders

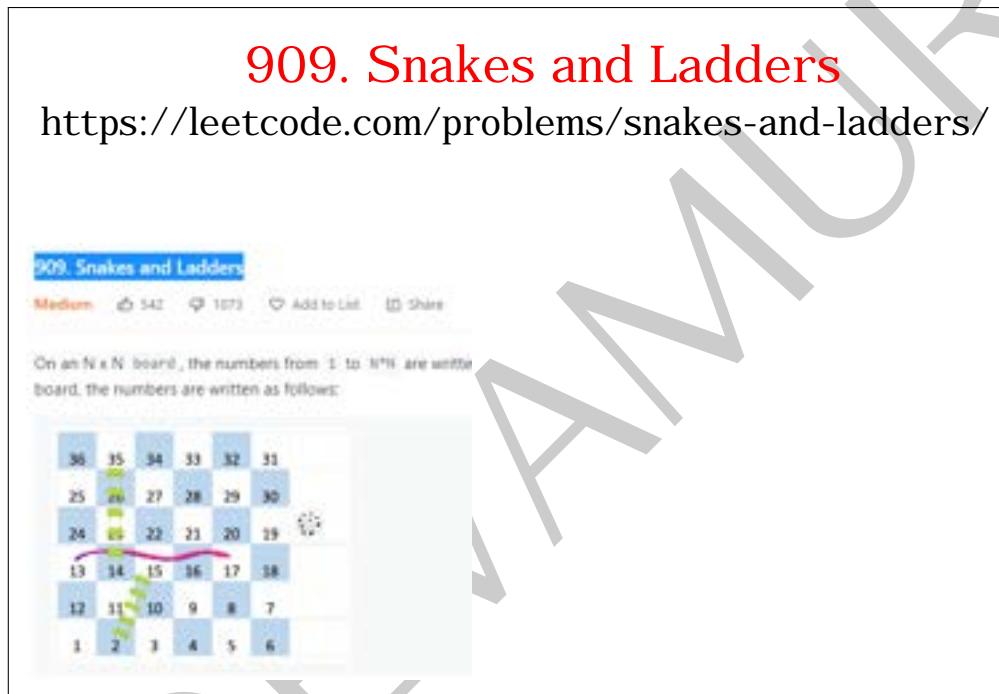


Figure 20.35: Problem 909: Snakes and Ladders

20.18.2 Coordinate system

20.18. LEETCODE PROBLEM 909: SNAKE AND LADDER
HACKERRANK:THE QUICKEST WAY UP

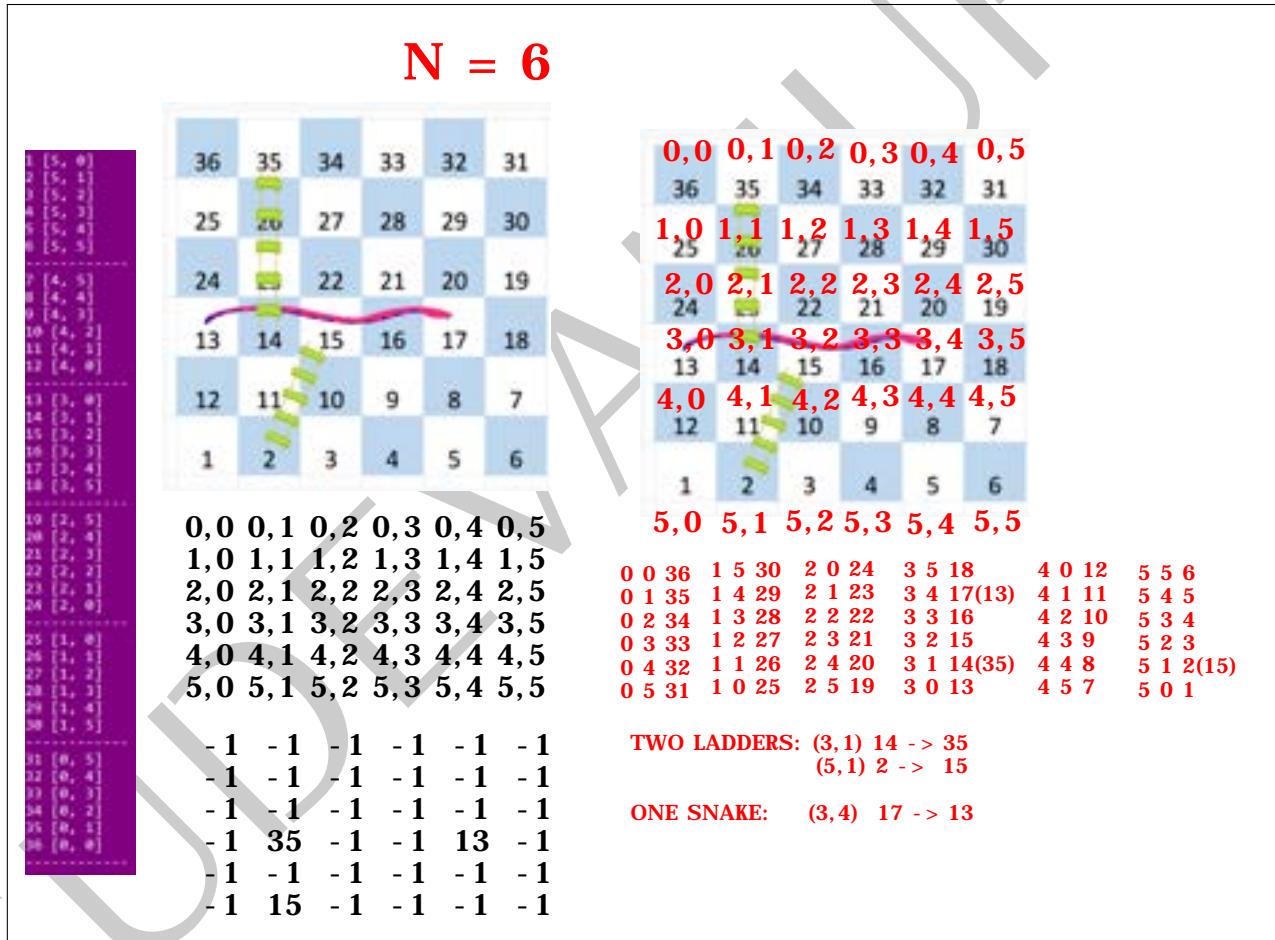


Figure 20.36: Coordinate system

20.18.3 Example 1

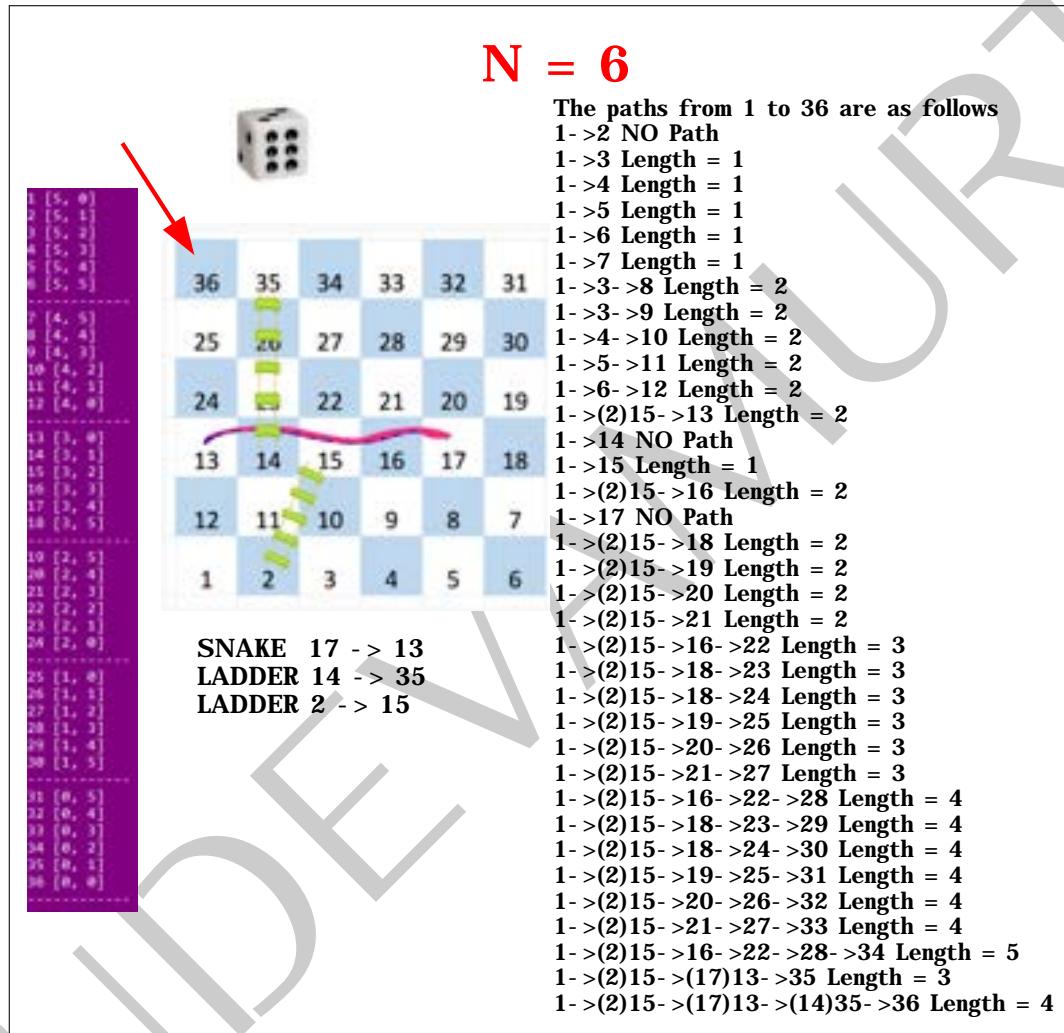


Figure 20.37: Example 1

20.18.4 Example 2

20.18. LEETCODE PROBLEM 909: SNAKE AND LADDER
HACKERRANK:THE QUICKEST WAY UP

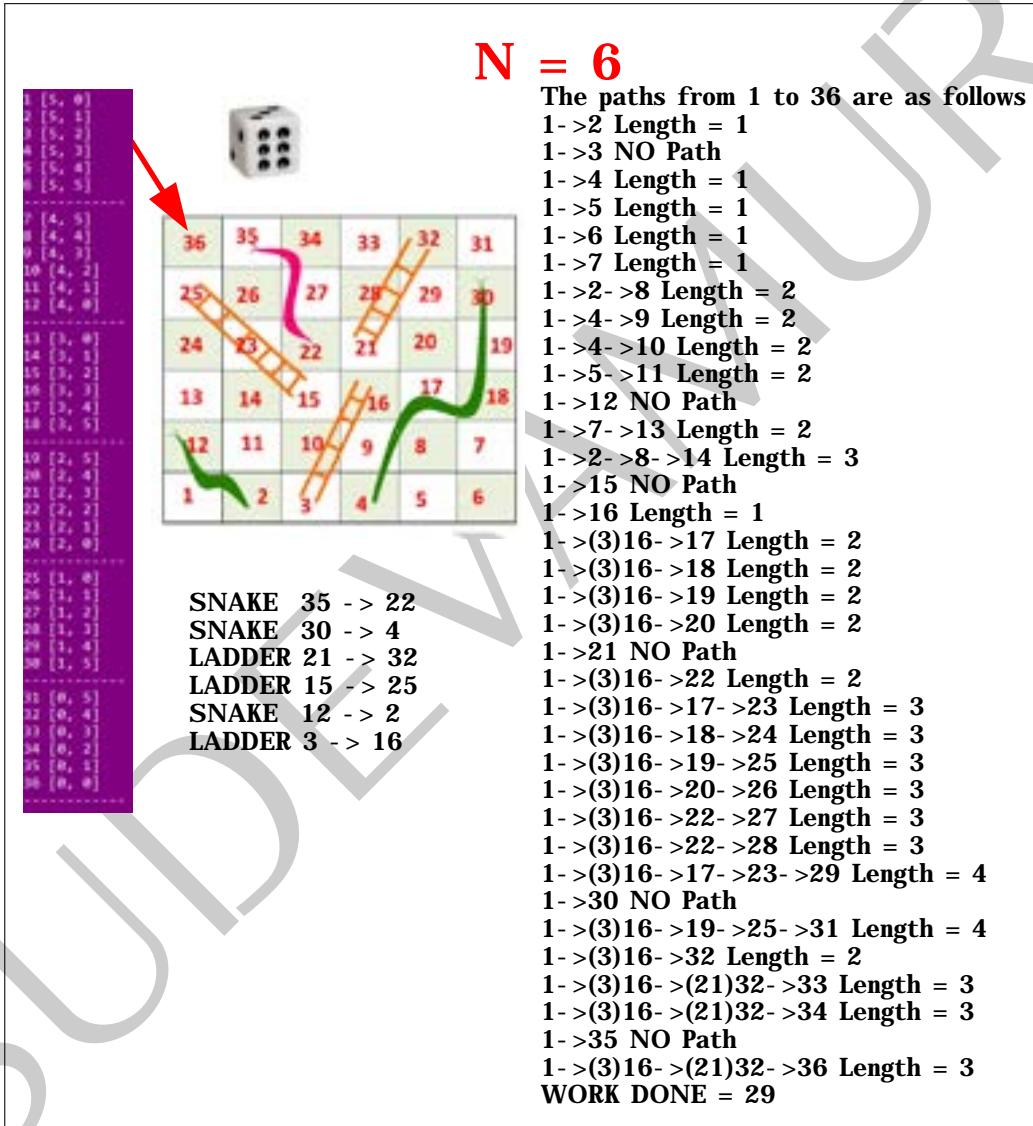


Figure 20.38: Example 2

20.18.5 Example 3

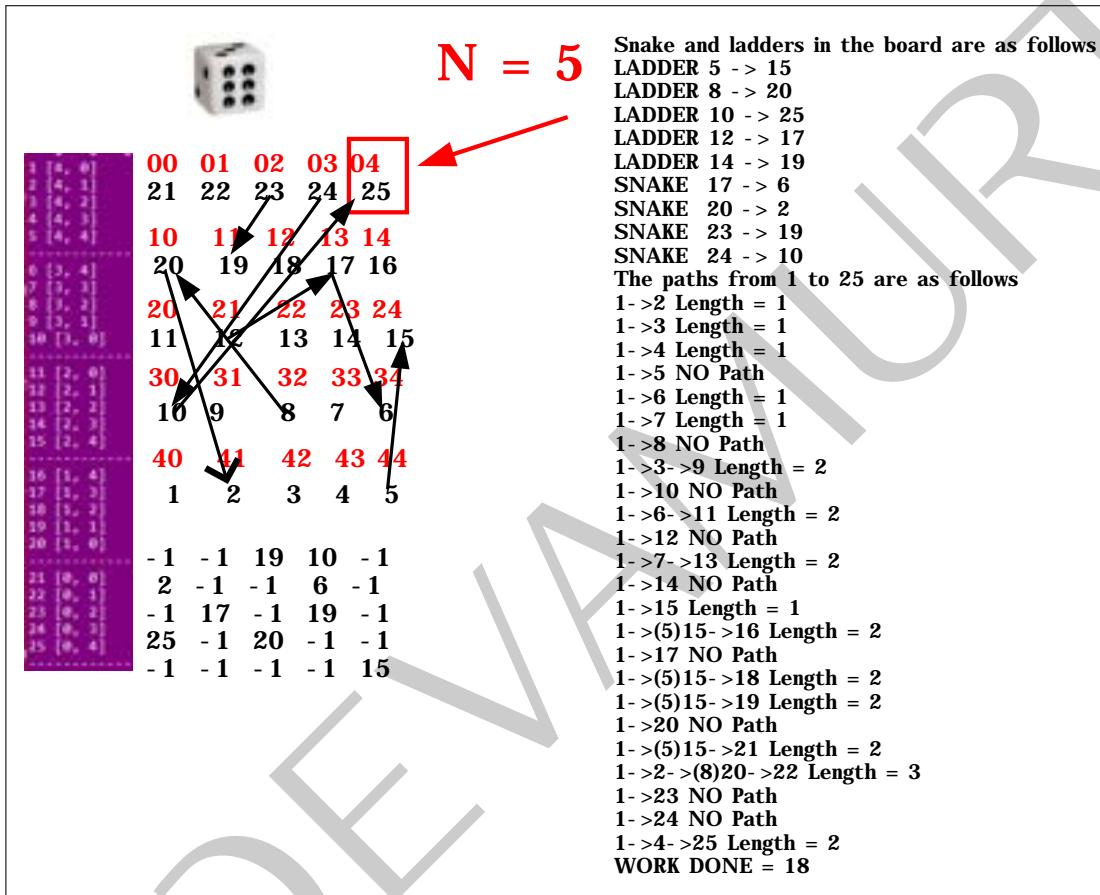


Figure 20.39: Example 3

20.18. LEETCODE PROBLEM 909: SNAKE AND LADDER
HACKERRANK:THE QUICKEST WAY UP

```
Success  Details +  
Runtime: 232 ms, faster than 9.29% of Python3 online submissions for Snakes and Ladders.  
Memory Usage: 14.8 MB, less than 5.92% of Python3 online submissions for Snakes and Ladders.  
  
class Solution:  
    ## YOU CANNOT CHANGE THIS INTERFACE  
    def snakesAndLadders(self, board: List[List[int]]) -> int:  
        num_moves = [-1]  
        work = [0]  
        p = L0909(board, 0, None, None, num_moves, work, False)  
        return num_moves[0]  
  
class L0909:  
    def __init__(self, board: List[List[int]], max:'int', ladders:'list',  
                 snakes:'list', num_moves:'List of size 1', work:'List of size 1', show:'boolean'):  
        self._b = board  
        self._max = max #note 0 for board  
        self._ladders = ladders  
        self._snakes = snakes  
        self._num_moves = num_moves  
        self._work = work  
        self._show = show  
  
        self._alg()
```

Figure 20.40: Problem 909: Snake and ladder passed

20.18.6 HackerRank: The Quickest Way Up

Snakes and Ladders: The Quickest Way Up

<https://www.hackerrank.com/challenges/the-quickest-way-up/problem>

Snakes and Ladders: The Quickest Way Up ★

Problem Submissions Leaderboard Discussions Editorial

Mashru takes out his board and Ladders game, stares at the board and wonders, "If I can always roll the die to whatever number I want, what would be the max number of rolls to reach the destination?"

Rules: The game is played with a board of 6 faces numbered 1 to 6.

- Starting from square 1, land on square 100 with the max roll of the die. If max roll the number rolled would place the player beyond square 100, no move is made.
- If a player lands at the base of a ladder, the player must climb the ladder. Ladders go up only.
- If a player lands at the mouth of a snake, the player must go down the snake and come out through the tail. Snakes go down only.

Function Description:

Complete the quickestWayUp function in the editor below. It should return an integer that represents the minimum number of moves required.

quickWayUp has the following parameter(s):

- ladders: a 2D integer array where each `ladders[i]` contains the start and end cell numbers of a ladder.
- snakes: a 2D integer array where each `snakes[i]` contains the start and end cell numbers of a snake.

Constraints:

- $1 \leq k \leq 100$
- $1 \leq n, m \leq 100$

The board is always 10×10 with squares numbered 1 to 100.

Another square 1 next square 100 will be the starting point of a ladder or snake.

A square will have at most one endpoint from either a snake or a ladder.

Figure 20.41: Problem description

Snake and ladder game

What is the minimal number of dice throws required to go from 1 to 36

```
{
    int[][] l = {
        {3, 16},
        {21, 32},
        {15, 25}
    };
    int[][] s = {
        {12, 2},
        {30, 4},
        {35, 22}
    };
    one("Board36", 36, l, s, 3);
}
```

Number of squares = 36
Work done = 59
Shortest path from 1 to 36 = 3

$1 \xrightarrow{2} 3(16) \xrightarrow{5} 21(32) \xrightarrow{4} 36$
3 throws of dice

Figure 20.42: Example1: Snake and ladder

Snake and ladder game

What is the minimal number of dice throws required to go from 1 to 100



`{
 int[][] l = {
 {32, 62},
 {42, 68},
 {12, 98}
 };
 int[][] s = {
 {95, 13},
 {97, 25},
 {93, 37},
 {79, 27},
 {75, 19},
 {49, 47},
 {67, 17}
 };
 one("Board100case1", 100, l, s, 3) ;
}`

Number of squares = 100
Work done = 238
Shorest path from 1 to 100 = 3

5 6 2

1 - -> 6 - -> 12(98) - -> 100

3 rolls

Figure 20.43: Example2: Snake and ladder

Snake and ladder game

What is the minimal number of dice throws required to go from 1 to 100



```
{  
    int[][] l = {  
        {8,52},  
        {6,80},  
        {42,26},//To trouble you  
        {2,72}  
    };  
    int[][] s = {  
        {51,19},  
        {39,11},  
        {29,37},//To trouble you  
        {3,81},///To trouble you  
        {59,5},  
        {79,23},  
        {7,53},  
        {43,33},  
        {21,77},  
    };  
    one("Board100case2", 100,l,s,5) ;  
}
```

Number of squares = 100
Work done = 411
Shorest path from 1 to 100 = 5

5 2 6 6 6
1 - - > 6(80) - - > 82 - - > 88 - - > 94 - - > 100
5 rolls

Figure 20.44: Example2: Snake and ladder

Snake and ladder game

What is the minimal number of dice throws required to go from 1 to 99



```
{  
    int[][] l = {  
        {2, 82},  
        {64, 99}  
    };  
    int[][] s = {  
        {84, 63},  
    };  
    SnakeAndLadder("SnakeisNotBad", 99, l, s) ;  
}
```

Number of squares = 99
Work done = 602
Shortest path from 1 to 99 = 3

**Number of squares = 99
Work done = 602
Shorest path from 1 to 99 = 3**

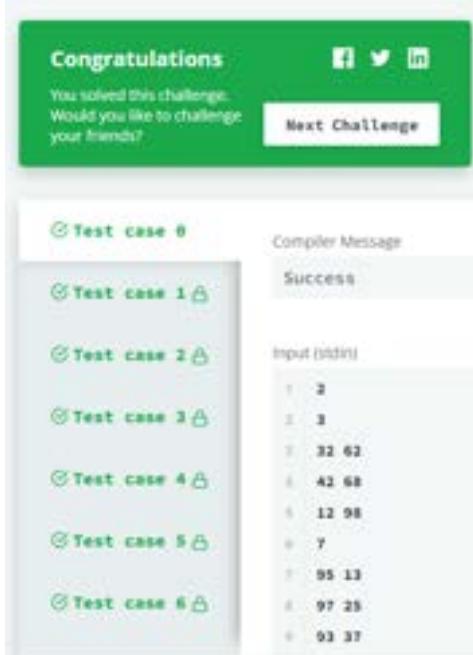
Snake is NOT bad

1 - -> 2(82) - - -> 84(63) - -> 64(99)
3 rolls

Figure 20.45: Example3: Snake is not bad. It helps

**20.18. LEETCODE PROBLEM 909: SNAKE AND LADDER.
HACKERRANK:THE QUICKEST WAY UP**

Snakes and Ladders: The Quickest Way Up
<https://www.hackerrank.com/challenges/the-quickest-way-up/problem>



Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message
Success

Input (stdin)

```
1 2
2 3
3 32 62
4 42 62
5 12 98
6 7
7 95 13
8 97 25
9 93 37
```

```
def quickestWayUp(ladders, snakes):
    num_moves = [-1]
    work = [0]
    max = 100
    p = L0909(None, max, ladders, snakes, num_moves, work, False)
    return num_moves[0]
```

Note same as LeetCode code

Figure 20.46: HackerRank passed

20.18.7 Expected output

20.19 Buying and Selling stocks

We are given an array of n integers representing stock prices on a single day.

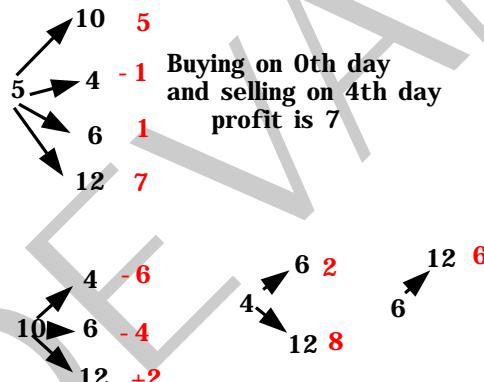
day	0	1	2	3	4
cost	5	10	4	6	12

We want to find a pair (buyDay, sellDay), with buyDay <= sellDay, such that if we bought the stock on buyDay and sold it on sellDay, we would maximize our profit.

$$\{\text{buyDay } 2, \text{ sellDay } 4\} = \text{profit} = (12 - 4 = 8\$)$$

0 1 2 3 4
5 10 4 6 12

Brute force



Time complexity = $\Theta(n^2)$

Space complexity = $\Theta(1)$

Figure 20.47: Problem description

20.19. BUYING AND SELLING STOCKS

20.19.1 $\Theta(n^2)$ time and $\Theta(1)$ space algorithm

See figure 20.47

20.19.2 $\Theta(n \log_2 n)$ time and $\Theta(\log_2 n)$ space algorithm

20.19. BUYING AND SELLING STOCKS

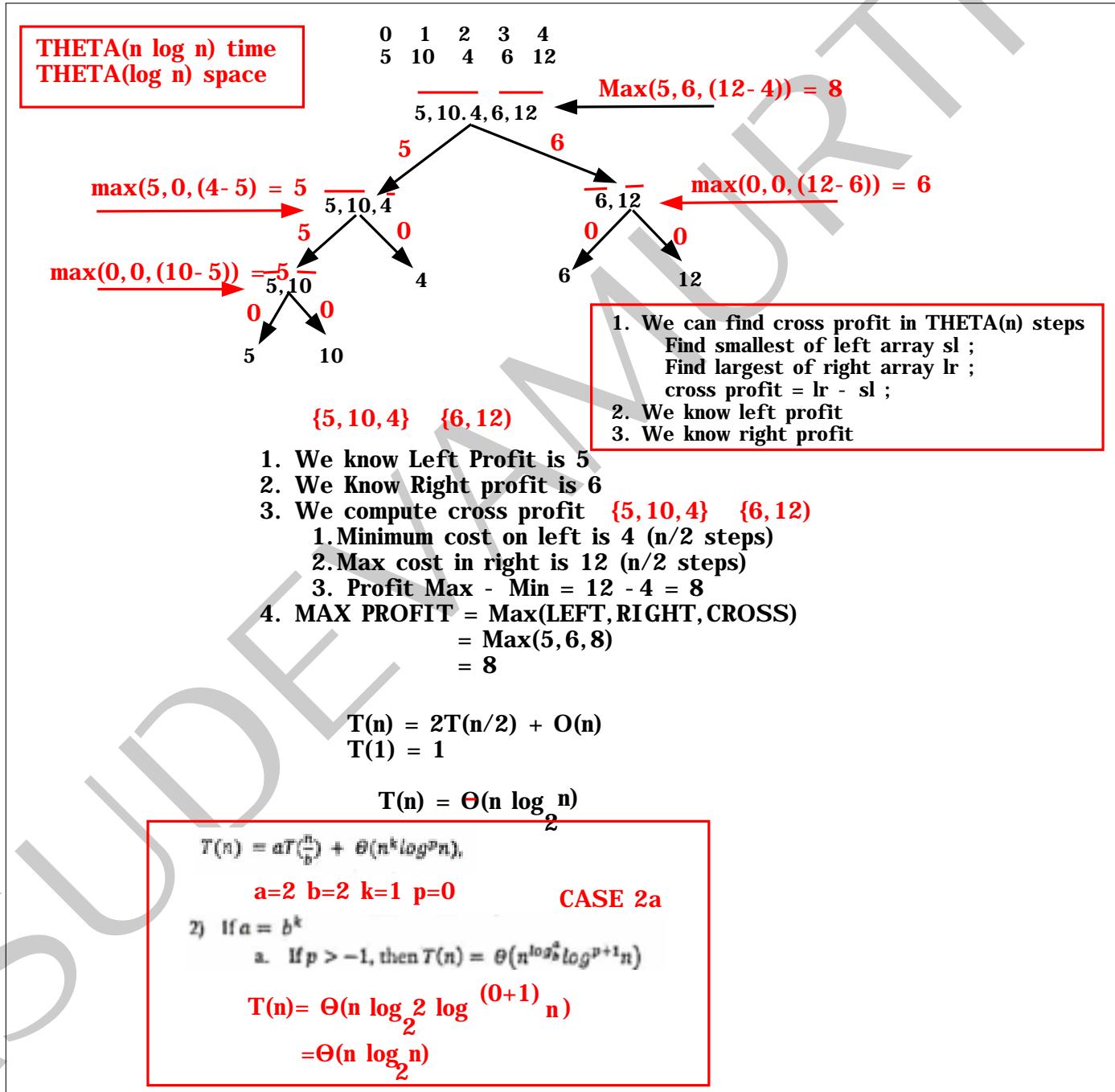


Figure 20.48: $\Theta(n \log_2 n)$ time and $\Theta(\log_2 n)$ space algorithm

20.19.3 $\Theta(n)$ time and $\Theta(\log_2 n)$ space algorithm

20.19. BUYING AND SELLING STOCKS

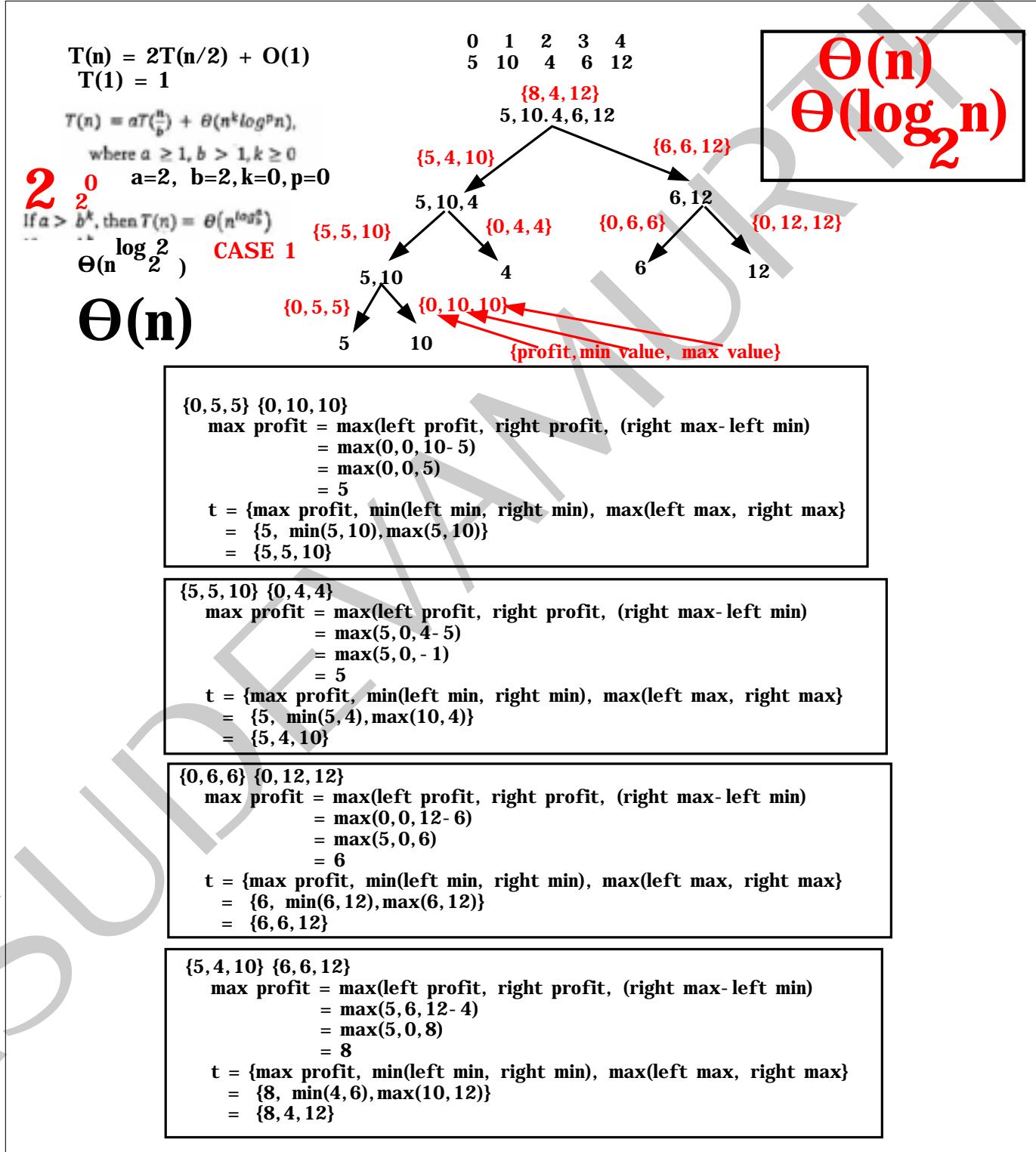


Figure 20.49: $\Theta(n)$ time and $\Theta(\log_2 n)$ space algorithm

20.19.4 $\Theta(n)$ time and $\Theta(1)$ space algorithm

20.19.5 Expected output

20.19. BUYING AND SELLING STOCKS

```
Testing Stock1.py Starts
 0  1  2  3  4
 5 10  4  6 12
nsquare_time_constant_space work = 10 Profit = 8
nlogn_time_nlogn_space work = 12 Profit = 8
nlogn_time_nlogn_space work = 5 Profit = 8
 0  1  2  3  4  5
 7  1  5  3  6  4
nsquare_time_constant_space work = 15 Profit = 5
nlogn_time_nlogn_space work = 16 Profit = 5
nlogn_time_nlogn_space work = 6 Profit = 5
----- 100 ascending tests-----
nsquare_time_constant_space work 4950
nlogn_time_nlogn_space work 672
nl_time_constant_space work 100
ascending tests passed
----- 100 descending tests-----
nsquare_time_constant_space work 4950
nlogn_time_nlogn_space work 672
nl_time_constant_space work 100
descending tests passed
----- 100 same value tests-----
nsquare_time_constant_space work 4950
nlogn_time_nlogn_space work 672
nl_time_constant_space work 100
same value tests passed
----- 500 random tests-----
nsquare_time_constant_space work 62375000
nlogn_time_nlogn_space work 2244000
nl_time_constant_space work 2500000
All 500 Random tests passed. You are a guru in stock trading
ALL TESTS PASSED
Testing Stock1.py Ends
Upload only Solution.py and output of the program as shown above
For A all tests must pass
Press any key to continue . . .
```

Figure 20.50: Expected output

20.19.6 Leetcode

121. Best Time to Buy and Sell Stock

<https://leetcode.com/problems/best-time-to-buy-and-sell-stock/>

Say you have an array for which the i th element is the price of a given stock on day i .

If you were only permitted to complete at most one transaction
(i.e., buy one and sell one share of the stock),
design an algorithm to find the maximum profit.

Note that you cannot sell a stock before you buy one.

Example 1:

Input: [7, 1, 5, 3, 6, 4]

Output: 5

Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = $6 - 1 = 5$.
Not $7 - 1 = 6$, as selling price needs to be larger than buying price.

Example 2:

Input: [7, 6, 4, 3, 1]

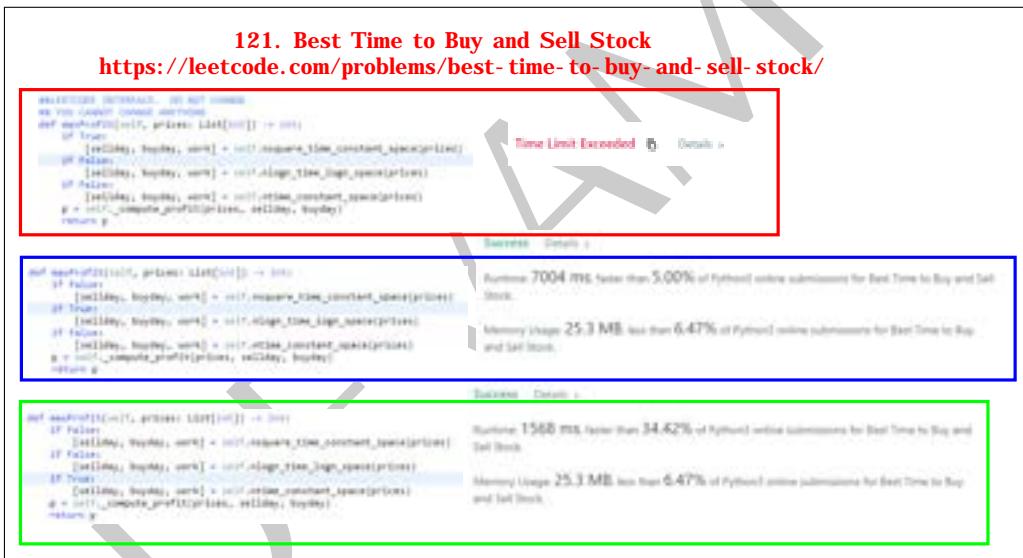
Output: 0

Explanation: In this case, no transaction is done, i.e. max profit = 0.

Figure 20.51: Leetcode description

20.19. BUYING AND SELLING STOCKS

121. Best Time to Buy and Sell Stock
<https://leetcode.com/problems/best-time-to-buy-and-sell-stock/>



```
class Solution:
    def maxProfit(self, prices: List[int]) -> int:
        if len(prices) < 2:
            return 0
        buyDay, sellDay = self.square_time_constant_space(prices)
        profit = self.compute_profit(prices, buyDay, sellDay)
        return profit
```

Time Limit Exceeded Details ↗

```
def maxProfit(self, prices: List[int]) -> int:
    if len(prices) < 2:
        return 0
    [buyDay, sellDay, profit] = self.square_time_constant_space(prices)
    if True:
        [buyDay, sellDay, profit] = self.sqrt_time_log_space(prices)
    if False:
        [buyDay, sellDay, profit] = self.linear_constant_space(prices)
    p = self._compute_profit(prices, buyDay, sellDay)
    return p
```

Runtime: 7004 ms, faster than 5.00% of Python3 online submissions for Best Time to Buy and Sell Stock.
Memory Usage: 25.3 MB, less than 6.47% of Python3 online submissions for Best Time to Buy and Sell Stock.

```
def maxProfit(self, prices: List[int]) -> int:
    if len(prices) < 2:
        return 0
    [buyDay, sellDay, profit] = self.square_time_constant_space(prices)
    if True:
        [buyDay, sellDay, profit] = self.sqrt_time_log_space(prices)
    if False:
        [buyDay, sellDay, profit] = self.linear_constant_space(prices)
    p = self._compute_profit(prices, buyDay, sellDay)
    return p
```

Runtime: 1568 ms, faster than 34.62% of Python3 online submissions for Best Time to Buy and Sell Stock.
Memory Usage: 25.3 MB, less than 6.47% of Python3 online submissions for Best Time to Buy and Sell Stock.

Figure 20.52: running all 3 algorithms on Leetcode

20.20 LeetCode Problem 322: Coin Change

322. Coin Change

<https://leetcode.com/problems/coin-change/>

You are given an integer array `coins` representing coins of different denominations and an integer `amount` representing a total amount of money.

Return the fewest number of coins that you need to make up that amount. If that amount of money cannot be made up by any combination of the coins, return `-1`.

You may assume that you have an infinite number of each kind of coin.

Example 1:

```
Input: coins = [1,2,5], amount = 11
Output: 3
Explanation: 11 = 5 + 5 + 1
```

Example 2:

```
Input: coins = [2], amount = 3
Output: -1
```

alg0

Figure 20.53: Problem 322: Coin Change

20.20.1 Expected output

Testing L0322Test Starts

----- Problem 1 -----

[0, 1, 2, 3, 4, 5, 6]

[0, 1, 2, 1, 1, 2, 2]

[0, 1, 1, 3, 4, 1, 3]

minimum change for 0 cents can be achieved using 0 coins.

minimum change for 1 cents can be achieved using 1 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 0

minimum change for 2 cents can be achieved using 2 coins.

20.20. LEETCODE PROBLEM 322: COIN CHANGE

1 : Give coin 1 . So far you have given= 1 .Remaining to give 1
2 : Give coin 1 . So far you have given= 2 .Remaining to give 0
minimum change for 3 cents can be achieved using 1 coins.
1 : Give coin 3 . So far you have given= 3 .Remaining to give 0
minimum change for 4 cents can be achieved using 1 coins.
1 : Give coin 4 . So far you have given= 4 .Remaining to give 0
minimum change for 5 cents can be achieved using 2 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 4
2 : Give coin 4 . So far you have given= 5 .Remaining to give 0
minimum change for 6 cents can be achieved using 2 coins.
1 : Give coin 3 . So far you have given= 3 .Remaining to give 3
2 : Give coin 3 . So far you have given= 6 .Remaining to give 0
WORK = 13

----- Problem 2 -----

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
[0, 1, 1, 2, 2, 1, 2, 2, 3, 3, 2, 3]
[0, 1, 2, 1, 2, 5, 1, 2, 1, 2, 5, 1]
minimum change for 0 cents can be achieved using 0 coins.
minimum change for 1 cents can be achieved using 1 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 0
minimum change for 2 cents can be achieved using 1 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 0
minimum change for 3 cents can be achieved using 2 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 2
2 : Give coin 2 . So far you have given= 3 .Remaining to give 0
minimum change for 4 cents can be achieved using 2 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 2
2 : Give coin 2 . So far you have given= 4 .Remaining to give 0
minimum change for 5 cents can be achieved using 1 coins.
1 : Give coin 5 . So far you have given= 5 .Remaining to give 0
minimum change for 6 cents can be achieved using 2 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 5
2 : Give coin 5 . So far you have given= 6 .Remaining to give 0
minimum change for 7 cents can be achieved using 2 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 5
2 : Give coin 5 . So far you have given= 7 .Remaining to give 0
minimum change for 8 cents can be achieved using 3 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 7
2 : Give coin 2 . So far you have given= 3 .Remaining to give 5
3 : Give coin 5 . So far you have given= 8 .Remaining to give 0
minimum change for 9 cents can be achieved using 3 coins.

```

1 : Give coin 2 . So far you have given= 2 .Remaining to give 7
2 : Give coin 2 . So far you have given= 4 .Remaining to give 5
3 : Give coin 5 . So far you have given= 9 .Remaining to give 0
minimum change for 10 cents can be achieved using 2 coins.
1 : Give coin 5 . So far you have given= 5 .Remaining to give 5
2 : Give coin 5 . So far you have given= 10 .Remaining to give 0
minimum change for 11 cents can be achieved using 3 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 10
2 : Give coin 5 . So far you have given= 6 .Remaining to give 5
3 : Give coin 5 . So far you have given= 11 .Remaining to give 0
WORK = 28
----- Problem 3 -----

```

```

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23
[0, 1, 1, 2, 2, 3, 1, 2, 2, 3, 1, 2, 2, 3, 3, 4, 2, 3, 3, 4, 2, 3, 3, 4, 1, 2, 2, 3
[0, 1, 2, 1, 2, 1, 6, 1, 2, 1, 10, 1, 2, 1, 2, 1, 6, 1, 2, 1, 10, 1, 2, 1, 24, 1, 25
minimum change for 0 cents can be achieved using 0 coins.
minimum change for 1 cents can be achieved using 1 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 0
minimum change for 2 cents can be achieved using 1 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 0
minimum change for 3 cents can be achieved using 2 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 2
2 : Give coin 2 . So far you have given= 3 .Remaining to give 0
minimum change for 4 cents can be achieved using 2 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 2
2 : Give coin 2 . So far you have given= 4 .Remaining to give 0
minimum change for 5 cents can be achieved using 3 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 4
2 : Give coin 2 . So far you have given= 3 .Remaining to give 2
3 : Give coin 2 . So far you have given= 5 .Remaining to give 0
minimum change for 6 cents can be achieved using 1 coins.
1 : Give coin 6 . So far you have given= 6 .Remaining to give 0
minimum change for 7 cents can be achieved using 2 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 6
2 : Give coin 6 . So far you have given= 7 .Remaining to give 0
minimum change for 8 cents can be achieved using 2 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 6
2 : Give coin 6 . So far you have given= 8 .Remaining to give 0
minimum change for 9 cents can be achieved using 3 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 8
2 : Give coin 2 . So far you have given= 3 .Remaining to give 6

```

20.20. LEETCODE PROBLEM 322: COIN CHANGE

3 : Give coin 6 . So far you have given= 9 .Remaining to give 0 minimum change for 10 cents can be achieved using 1 coins.

1 : Give coin 10 . So far you have given= 10 .Remaining to give 0 minimum change for 11 cents can be achieved using 2 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 10

2 : Give coin 10 . So far you have given= 11 .Remaining to give 0 minimum change for 12 cents can be achieved using 2 coins.

1 : Give coin 2 . So far you have given= 2 .Remaining to give 10

2 : Give coin 10 . So far you have given= 12 .Remaining to give 0 minimum change for 13 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 12

2 : Give coin 2 . So far you have given= 3 .Remaining to give 10

3 : Give coin 10 . So far you have given= 13 .Remaining to give 0 minimum change for 14 cents can be achieved using 3 coins.

1 : Give coin 2 . So far you have given= 2 .Remaining to give 12

2 : Give coin 2 . So far you have given= 4 .Remaining to give 10

3 : Give coin 10 . So far you have given= 14 .Remaining to give 0 minimum change for 15 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 14

2 : Give coin 2 . So far you have given= 3 .Remaining to give 12

3 : Give coin 2 . So far you have given= 5 .Remaining to give 10

4 : Give coin 10 . So far you have given= 15 .Remaining to give 0 minimum change for 16 cents can be achieved using 2 coins.

1 : Give coin 6 . So far you have given= 6 .Remaining to give 10

2 : Give coin 10 . So far you have given= 16 .Remaining to give 0 minimum change for 17 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 16

2 : Give coin 6 . So far you have given= 7 .Remaining to give 10

3 : Give coin 10 . So far you have given= 17 .Remaining to give 0 minimum change for 18 cents can be achieved using 3 coins.

1 : Give coin 2 . So far you have given= 2 .Remaining to give 16

2 : Give coin 6 . So far you have given= 8 .Remaining to give 10

3 : Give coin 10 . So far you have given= 18 .Remaining to give 0 minimum change for 19 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 18

2 : Give coin 2 . So far you have given= 3 .Remaining to give 16

3 : Give coin 6 . So far you have given= 9 .Remaining to give 10

4 : Give coin 10 . So far you have given= 19 .Remaining to give 0 minimum change for 20 cents can be achieved using 2 coins.

1 : Give coin 10 . So far you have given= 10 .Remaining to give 10

2 : Give coin 10 . So far you have given= 20 .Remaining to give 0

minimum change for 21 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 20
2 : Give coin 10 . So far you have given= 11 .Remaining to give 10
3 : Give coin 10 . So far you have given= 21 .Remaining to give 0
minimum change for 22 cents can be achieved using 3 coins.

1 : Give coin 2 . So far you have given= 2 .Remaining to give 20
2 : Give coin 10 . So far you have given= 12 .Remaining to give 10
3 : Give coin 10 . So far you have given= 22 .Remaining to give 0
minimum change for 23 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 22
2 : Give coin 2 . So far you have given= 3 .Remaining to give 20
3 : Give coin 10 . So far you have given= 13 .Remaining to give 10
4 : Give coin 10 . So far you have given= 23 .Remaining to give 0
minimum change for 24 cents can be achieved using 1 coins.

1 : Give coin 24 . So far you have given= 24 .Remaining to give 0
minimum change for 25 cents can be achieved using 2 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 24
2 : Give coin 24 . So far you have given= 25 .Remaining to give 0
minimum change for 26 cents can be achieved using 2 coins.

1 : Give coin 2 . So far you have given= 2 .Remaining to give 24
2 : Give coin 24 . So far you have given= 26 .Remaining to give 0
minimum change for 27 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 26
2 : Give coin 2 . So far you have given= 3 .Remaining to give 24
3 : Give coin 24 . So far you have given= 27 .Remaining to give 0
minimum change for 28 cents can be achieved using 3 coins.

1 : Give coin 2 . So far you have given= 2 .Remaining to give 26
2 : Give coin 2 . So far you have given= 4 .Remaining to give 24
3 : Give coin 24 . So far you have given= 28 .Remaining to give 0
minimum change for 29 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 28
2 : Give coin 2 . So far you have given= 3 .Remaining to give 26
3 : Give coin 2 . So far you have given= 5 .Remaining to give 24
4 : Give coin 24 . So far you have given= 29 .Remaining to give 0
minimum change for 30 cents can be achieved using 1 coins.

1 : Give coin 30 . So far you have given= 30 .Remaining to give 0
minimum change for 31 cents can be achieved using 2 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 30
2 : Give coin 30 . So far you have given= 31 .Remaining to give 0
minimum change for 32 cents can be achieved using 2 coins.

1 : Give coin 2 . So far you have given= 2 .Remaining to give 30

20.20. LEETCODE PROBLEM 322: COIN CHANGE

2 : Give coin 30 . So far you have given= 32 . Remaining to give 0 minimum change for 33 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 32

2 : Give coin 2 . So far you have given= 3 . Remaining to give 30

3 : Give coin 30 . So far you have given= 33 . Remaining to give 0 minimum change for 34 cents can be achieved using 2 coins.

1 : Give coin 10 . So far you have given= 10 . Remaining to give 24

2 : Give coin 24 . So far you have given= 34 . Remaining to give 0 minimum change for 35 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 34

2 : Give coin 10 . So far you have given= 11 . Remaining to give 24

3 : Give coin 24 . So far you have given= 35 . Remaining to give 0 minimum change for 36 cents can be achieved using 2 coins.

1 : Give coin 6 . So far you have given= 6 . Remaining to give 30

2 : Give coin 30 . So far you have given= 36 . Remaining to give 0 minimum change for 37 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 36

2 : Give coin 6 . So far you have given= 7 . Remaining to give 30

3 : Give coin 30 . So far you have given= 37 . Remaining to give 0 minimum change for 38 cents can be achieved using 3 coins.

1 : Give coin 2 . So far you have given= 2 . Remaining to give 36

2 : Give coin 6 . So far you have given= 8 . Remaining to give 30

3 : Give coin 30 . So far you have given= 38 . Remaining to give 0 minimum change for 39 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 38

2 : Give coin 2 . So far you have given= 3 . Remaining to give 36

3 : Give coin 6 . So far you have given= 9 . Remaining to give 30

4 : Give coin 30 . So far you have given= 39 . Remaining to give 0 minimum change for 40 cents can be achieved using 2 coins.

1 : Give coin 10 . So far you have given= 10 . Remaining to give 30

2 : Give coin 30 . So far you have given= 40 . Remaining to give 0 minimum change for 41 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 40

2 : Give coin 10 . So far you have given= 11 . Remaining to give 30

3 : Give coin 30 . So far you have given= 41 . Remaining to give 0 minimum change for 42 cents can be achieved using 3 coins.

1 : Give coin 2 . So far you have given= 2 . Remaining to give 40

2 : Give coin 10 . So far you have given= 12 . Remaining to give 30

3 : Give coin 30 . So far you have given= 42 . Remaining to give 0 minimum change for 43 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 42

2 : Give coin 2 . So far you have given= 3 .Remaining to give 40
3 : Give coin 10 . So far you have given= 13 .Remaining to give 30
4 : Give coin 30 . So far you have given= 43 .Remaining to give 0
minimum change for 44 cents can be achieved using 3 coins.
1 : Give coin 10 . So far you have given= 10 .Remaining to give 34
2 : Give coin 10 . So far you have given= 20 .Remaining to give 24
3 : Give coin 24 . So far you have given= 44 .Remaining to give 0
minimum change for 45 cents can be achieved using 4 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 44
2 : Give coin 10 . So far you have given= 11 .Remaining to give 34
3 : Give coin 10 . So far you have given= 21 .Remaining to give 24
4 : Give coin 24 . So far you have given= 45 .Remaining to give 0
minimum change for 46 cents can be achieved using 3 coins.
1 : Give coin 6 . So far you have given= 6 .Remaining to give 40
2 : Give coin 10 . So far you have given= 16 .Remaining to give 30
3 : Give coin 30 . So far you have given= 46 .Remaining to give 0
minimum change for 47 cents can be achieved using 4 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 46
2 : Give coin 6 . So far you have given= 7 .Remaining to give 40
3 : Give coin 10 . So far you have given= 17 .Remaining to give 30
4 : Give coin 30 . So far you have given= 47 .Remaining to give 0
minimum change for 48 cents can be achieved using 2 coins.
1 : Give coin 24 . So far you have given= 24 .Remaining to give 24
2 : Give coin 24 . So far you have given= 48 .Remaining to give 0
minimum change for 49 cents can be achieved using 3 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 48
2 : Give coin 24 . So far you have given= 25 .Remaining to give 24
3 : Give coin 24 . So far you have given= 49 .Remaining to give 0
minimum change for 50 cents can be achieved using 3 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 48
2 : Give coin 24 . So far you have given= 26 .Remaining to give 24
3 : Give coin 24 . So far you have given= 50 .Remaining to give 0
minimum change for 51 cents can be achieved using 4 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 50
2 : Give coin 2 . So far you have given= 3 .Remaining to give 48
3 : Give coin 24 . So far you have given= 27 .Remaining to give 24
4 : Give coin 24 . So far you have given= 51 .Remaining to give 0
minimum change for 52 cents can be achieved using 4 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 50
2 : Give coin 2 . So far you have given= 4 .Remaining to give 48
3 : Give coin 24 . So far you have given= 28 .Remaining to give 24

20.20. LEETCODE PROBLEM 322: COIN CHANGE

4 : Give coin 24 . So far you have given= 52 . Remaining to give 0 minimum change for 53 cents can be achieved using 5 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 52

2 : Give coin 2 . So far you have given= 3 . Remaining to give 50

3 : Give coin 2 . So far you have given= 5 . Remaining to give 48

4 : Give coin 24 . So far you have given= 29 . Remaining to give 24

5 : Give coin 24 . So far you have given= 53 . Remaining to give 0 minimum change for 54 cents can be achieved using 2 coins.

1 : Give coin 24 . So far you have given= 24 . Remaining to give 30

2 : Give coin 30 . So far you have given= 54 . Remaining to give 0 minimum change for 55 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 54

2 : Give coin 24 . So far you have given= 25 . Remaining to give 30

3 : Give coin 30 . So far you have given= 55 . Remaining to give 0 minimum change for 56 cents can be achieved using 3 coins.

1 : Give coin 2 . So far you have given= 2 . Remaining to give 54

2 : Give coin 24 . So far you have given= 26 . Remaining to give 30

3 : Give coin 30 . So far you have given= 56 . Remaining to give 0 minimum change for 57 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 56

2 : Give coin 2 . So far you have given= 3 . Remaining to give 54

3 : Give coin 24 . So far you have given= 27 . Remaining to give 30

4 : Give coin 30 . So far you have given= 57 . Remaining to give 0 minimum change for 58 cents can be achieved using 3 coins.

1 : Give coin 10 . So far you have given= 10 . Remaining to give 48

2 : Give coin 24 . So far you have given= 34 . Remaining to give 24

3 : Give coin 24 . So far you have given= 58 . Remaining to give 0 minimum change for 59 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 58

2 : Give coin 10 . So far you have given= 11 . Remaining to give 48

3 : Give coin 24 . So far you have given= 35 . Remaining to give 24

4 : Give coin 24 . So far you have given= 59 . Remaining to give 0 minimum change for 60 cents can be achieved using 2 coins.

1 : Give coin 30 . So far you have given= 30 . Remaining to give 30

2 : Give coin 30 . So far you have given= 60 . Remaining to give 0 minimum change for 61 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 60

2 : Give coin 30 . So far you have given= 31 . Remaining to give 30

3 : Give coin 30 . So far you have given= 61 . Remaining to give 0 minimum change for 62 cents can be achieved using 3 coins.

1 : Give coin 2 . So far you have given= 2 . Remaining to give 60

2 : Give coin 30 . So far you have given= 32 .Remaining to give 30
3 : Give coin 30 . So far you have given= 62 .Remaining to give 0
minimum change for 63 cents can be achieved using 4 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 62
2 : Give coin 2 . So far you have given= 3 .Remaining to give 60
3 : Give coin 30 . So far you have given= 33 .Remaining to give 30
4 : Give coin 30 . So far you have given= 63 .Remaining to give 0
minimum change for 64 cents can be achieved using 3 coins.
1 : Give coin 10 . So far you have given= 10 .Remaining to give 54
2 : Give coin 24 . So far you have given= 34 .Remaining to give 30
3 : Give coin 30 . So far you have given= 64 .Remaining to give 0
minimum change for 65 cents can be achieved using 4 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 64
2 : Give coin 10 . So far you have given= 11 .Remaining to give 54
3 : Give coin 24 . So far you have given= 35 .Remaining to give 30
4 : Give coin 30 . So far you have given= 65 .Remaining to give 0
minimum change for 66 cents can be achieved using 3 coins.
1 : Give coin 6 . So far you have given= 6 .Remaining to give 60
2 : Give coin 30 . So far you have given= 36 .Remaining to give 30
3 : Give coin 30 . So far you have given= 66 .Remaining to give 0
minimum change for 67 cents can be achieved using 4 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 66
2 : Give coin 6 . So far you have given= 7 .Remaining to give 60
3 : Give coin 30 . So far you have given= 37 .Remaining to give 30
4 : Give coin 30 . So far you have given= 67 .Remaining to give 0
minimum change for 68 cents can be achieved using 4 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 66
2 : Give coin 6 . So far you have given= 8 .Remaining to give 60
3 : Give coin 30 . So far you have given= 38 .Remaining to give 30
4 : Give coin 30 . So far you have given= 68 .Remaining to give 0
minimum change for 69 cents can be achieved using 5 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 68
2 : Give coin 2 . So far you have given= 3 .Remaining to give 66
3 : Give coin 6 . So far you have given= 9 .Remaining to give 60
4 : Give coin 30 . So far you have given= 39 .Remaining to give 30
5 : Give coin 30 . So far you have given= 69 .Remaining to give 0
minimum change for 70 cents can be achieved using 3 coins.
1 : Give coin 10 . So far you have given= 10 .Remaining to give 60
2 : Give coin 30 . So far you have given= 40 .Remaining to give 30
3 : Give coin 30 . So far you have given= 70 .Remaining to give 0
minimum change for 71 cents can be achieved using 4 coins.

20.20. LEETCODE PROBLEM 322: COIN CHANGE

1 : Give coin 1 . So far you have given= 1 .Remaining to give 70
2 : Give coin 10 . So far you have given= 11 .Remaining to give 60
3 : Give coin 30 . So far you have given= 41 .Remaining to give 30
4 : Give coin 30 . So far you have given= 71 .Remaining to give 0
minimum change for 72 cents can be achieved using 3 coins.
1 : Give coin 24 . So far you have given= 24 .Remaining to give 48
2 : Give coin 24 . So far you have given= 48 .Remaining to give 24
3 : Give coin 24 . So far you have given= 72 .Remaining to give 0
minimum change for 73 cents can be achieved using 4 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 72
2 : Give coin 24 . So far you have given= 25 .Remaining to give 48
3 : Give coin 24 . So far you have given= 49 .Remaining to give 24
4 : Give coin 24 . So far you have given= 73 .Remaining to give 0
minimum change for 74 cents can be achieved using 4 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 72
2 : Give coin 24 . So far you have given= 26 .Remaining to give 48
3 : Give coin 24 . So far you have given= 50 .Remaining to give 24
4 : Give coin 24 . So far you have given= 74 .Remaining to give 0
minimum change for 75 cents can be achieved using 5 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 74
2 : Give coin 2 . So far you have given= 3 .Remaining to give 72
3 : Give coin 24 . So far you have given= 27 .Remaining to give 48
4 : Give coin 24 . So far you have given= 51 .Remaining to give 24
5 : Give coin 24 . So far you have given= 75 .Remaining to give 0
minimum change for 76 cents can be achieved using 4 coins.
1 : Give coin 6 . So far you have given= 6 .Remaining to give 70
2 : Give coin 10 . So far you have given= 16 .Remaining to give 60
3 : Give coin 30 . So far you have given= 46 .Remaining to give 30
4 : Give coin 30 . So far you have given= 76 .Remaining to give 0
minimum change for 77 cents can be achieved using 5 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 76
2 : Give coin 6 . So far you have given= 7 .Remaining to give 70
3 : Give coin 10 . So far you have given= 17 .Remaining to give 60
4 : Give coin 30 . So far you have given= 47 .Remaining to give 30
5 : Give coin 30 . So far you have given= 77 .Remaining to give 0
minimum change for 78 cents can be achieved using 3 coins.
1 : Give coin 24 . So far you have given= 24 .Remaining to give 54
2 : Give coin 24 . So far you have given= 48 .Remaining to give 30
3 : Give coin 30 . So far you have given= 78 .Remaining to give 0
minimum change for 79 cents can be achieved using 4 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 78

2 : Give coin 24 . So far you have given= 25 .Remaining to give 54
3 : Give coin 24 . So far you have given= 49 .Remaining to give 30
4 : Give coin 30 . So far you have given= 79 .Remaining to give 0
minimum change for 80 cents can be achieved using 4 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 78
2 : Give coin 24 . So far you have given= 26 .Remaining to give 54
3 : Give coin 24 . So far you have given= 50 .Remaining to give 30
4 : Give coin 30 . So far you have given= 80 .Remaining to give 0
minimum change for 81 cents can be achieved using 5 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 80
2 : Give coin 2 . So far you have given= 3 .Remaining to give 78
3 : Give coin 24 . So far you have given= 27 .Remaining to give 54
4 : Give coin 24 . So far you have given= 51 .Remaining to give 30
5 : Give coin 30 . So far you have given= 81 .Remaining to give 0
minimum change for 82 cents can be achieved using 4 coins.
1 : Give coin 10 . So far you have given= 10 .Remaining to give 72
2 : Give coin 24 . So far you have given= 34 .Remaining to give 48
3 : Give coin 24 . So far you have given= 58 .Remaining to give 24
4 : Give coin 24 . So far you have given= 82 .Remaining to give 0
minimum change for 83 cents can be achieved using 5 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 82
2 : Give coin 10 . So far you have given= 11 .Remaining to give 72
3 : Give coin 24 . So far you have given= 35 .Remaining to give 48
4 : Give coin 24 . So far you have given= 59 .Remaining to give 24
5 : Give coin 24 . So far you have given= 83 .Remaining to give 0
minimum change for 84 cents can be achieved using 3 coins.
1 : Give coin 24 . So far you have given= 24 .Remaining to give 60
2 : Give coin 30 . So far you have given= 54 .Remaining to give 30
3 : Give coin 30 . So far you have given= 84 .Remaining to give 0
minimum change for 85 cents can be achieved using 4 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 84
2 : Give coin 24 . So far you have given= 25 .Remaining to give 60
3 : Give coin 30 . So far you have given= 55 .Remaining to give 30
4 : Give coin 30 . So far you have given= 85 .Remaining to give 0
minimum change for 86 cents can be achieved using 4 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 84
2 : Give coin 24 . So far you have given= 26 .Remaining to give 60
3 : Give coin 30 . So far you have given= 56 .Remaining to give 30
4 : Give coin 30 . So far you have given= 86 .Remaining to give 0
minimum change for 87 cents can be achieved using 5 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 86

20.20. LEETCODE PROBLEM 322: COIN CHANGE

2 : Give coin 2 . So far you have given= 3 .Remaining to give 84
3 : Give coin 24 . So far you have given= 27 .Remaining to give 60
4 : Give coin 30 . So far you have given= 57 .Remaining to give 30
5 : Give coin 30 . So far you have given= 87 .Remaining to give 0
minimum change for 88 cents can be achieved using 4 coins.

1 : Give coin 10 . So far you have given= 10 .Remaining to give 78
2 : Give coin 24 . So far you have given= 34 .Remaining to give 54
3 : Give coin 24 . So far you have given= 58 .Remaining to give 30
4 : Give coin 30 . So far you have given= 88 .Remaining to give 0
minimum change for 89 cents can be achieved using 5 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 88
2 : Give coin 10 . So far you have given= 11 .Remaining to give 78
3 : Give coin 24 . So far you have given= 35 .Remaining to give 54
4 : Give coin 24 . So far you have given= 59 .Remaining to give 30
5 : Give coin 30 . So far you have given= 89 .Remaining to give 0
minimum change for 90 cents can be achieved using 1 coins.

1 : Give coin 90 . So far you have given= 90 .Remaining to give 0
minimum change for 91 cents can be achieved using 2 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 90
2 : Give coin 90 . So far you have given= 91 .Remaining to give 0
minimum change for 92 cents can be achieved using 2 coins.

1 : Give coin 2 . So far you have given= 2 .Remaining to give 90
2 : Give coin 90 . So far you have given= 92 .Remaining to give 0
minimum change for 93 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 92
2 : Give coin 2 . So far you have given= 3 .Remaining to give 90
3 : Give coin 90 . So far you have given= 93 .Remaining to give 0
minimum change for 94 cents can be achieved using 3 coins.

1 : Give coin 2 . So far you have given= 2 .Remaining to give 92
2 : Give coin 2 . So far you have given= 4 .Remaining to give 90
3 : Give coin 90 . So far you have given= 94 .Remaining to give 0
minimum change for 95 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 94
2 : Give coin 2 . So far you have given= 3 .Remaining to give 92
3 : Give coin 2 . So far you have given= 5 .Remaining to give 90
4 : Give coin 90 . So far you have given= 95 .Remaining to give 0
minimum change for 96 cents can be achieved using 2 coins.

1 : Give coin 6 . So far you have given= 6 .Remaining to give 90
2 : Give coin 90 . So far you have given= 96 .Remaining to give 0
minimum change for 97 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 96

2 : Give coin 6 . So far you have given= 7 .Remaining to give 90
3 : Give coin 90 . So far you have given= 97 .Remaining to give 0
minimum change for 98 cents can be achieved using 3 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 96
2 : Give coin 6 . So far you have given= 8 .Remaining to give 90
3 : Give coin 90 . So far you have given= 98 .Remaining to give 0
minimum change for 99 cents can be achieved using 4 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 98
2 : Give coin 2 . So far you have given= 3 .Remaining to give 96
3 : Give coin 6 . So far you have given= 9 .Remaining to give 90
4 : Give coin 90 . So far you have given= 99 .Remaining to give 0
minimum change for 100 cents can be achieved using 2 coins.
1 : Give coin 10 . So far you have given= 10 .Remaining to give 90
2 : Give coin 90 . So far you have given= 100 .Remaining to give 0
WORK = 544

----- Problem 4 -----

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
[0, 1, 1, 2, 2, 3, 1, 2, 2, 3, 1, 2, 2, 3, 3, 4, 2, 3, 3, 4, 2, 3, 3, 4, 1, 2, 2, 3, 3]
[0, 1, 2, 1, 2, 1, 6, 1, 2, 1, 10, 1, 2, 1, 2, 1, 6, 1, 2, 1, 10, 1, 2, 1, 24, 1, 25]
minimum change for 0 cents can be achieved using 0 coins.
minimum change for 1 cents can be achieved using 1 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 0
minimum change for 2 cents can be achieved using 1 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 0
minimum change for 3 cents can be achieved using 2 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 2
2 : Give coin 2 . So far you have given= 3 .Remaining to give 0
minimum change for 4 cents can be achieved using 2 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 2
2 : Give coin 2 . So far you have given= 4 .Remaining to give 0
minimum change for 5 cents can be achieved using 3 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 4
2 : Give coin 2 . So far you have given= 3 .Remaining to give 2
3 : Give coin 2 . So far you have given= 5 .Remaining to give 0
minimum change for 6 cents can be achieved using 1 coins.
1 : Give coin 6 . So far you have given= 6 .Remaining to give 0
minimum change for 7 cents can be achieved using 2 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 6
2 : Give coin 6 . So far you have given= 7 .Remaining to give 0
minimum change for 8 cents can be achieved using 2 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 6

20.20. LEETCODE PROBLEM 322: COIN CHANGE

2 : Give coin 6 . So far you have given= 8 .Remaining to give 0 minimum change for 9 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 8

2 : Give coin 2 . So far you have given= 3 .Remaining to give 6

3 : Give coin 6 . So far you have given= 9 .Remaining to give 0 minimum change for 10 cents can be achieved using 1 coins.

1 : Give coin 10 . So far you have given= 10 .Remaining to give 0 minimum change for 11 cents can be achieved using 2 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 10

2 : Give coin 10 . So far you have given= 11 .Remaining to give 0 minimum change for 12 cents can be achieved using 2 coins.

1 : Give coin 2 . So far you have given= 2 .Remaining to give 10

2 : Give coin 10 . So far you have given= 12 .Remaining to give 0 minimum change for 13 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 12

2 : Give coin 2 . So far you have given= 3 .Remaining to give 10

3 : Give coin 10 . So far you have given= 13 .Remaining to give 0 minimum change for 14 cents can be achieved using 3 coins.

1 : Give coin 2 . So far you have given= 2 .Remaining to give 12

2 : Give coin 2 . So far you have given= 4 .Remaining to give 10

3 : Give coin 10 . So far you have given= 14 .Remaining to give 0 minimum change for 15 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 14

2 : Give coin 2 . So far you have given= 3 .Remaining to give 12

3 : Give coin 2 . So far you have given= 5 .Remaining to give 10

4 : Give coin 10 . So far you have given= 15 .Remaining to give 0 minimum change for 16 cents can be achieved using 2 coins.

1 : Give coin 6 . So far you have given= 6 .Remaining to give 10

2 : Give coin 10 . So far you have given= 16 .Remaining to give 0 minimum change for 17 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 16

2 : Give coin 6 . So far you have given= 7 .Remaining to give 10

3 : Give coin 10 . So far you have given= 17 .Remaining to give 0 minimum change for 18 cents can be achieved using 3 coins.

1 : Give coin 2 . So far you have given= 2 .Remaining to give 16

2 : Give coin 6 . So far you have given= 8 .Remaining to give 10

3 : Give coin 10 . So far you have given= 18 .Remaining to give 0 minimum change for 19 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 18

2 : Give coin 2 . So far you have given= 3 .Remaining to give 16

3 : Give coin 6 . So far you have given= 9 .Remaining to give 10

4 : Give coin 10 . So far you have given= 19 .Remaining to give 0
minimum change for 20 cents can be achieved using 2 coins.
1 : Give coin 10 . So far you have given= 10 .Remaining to give 10
2 : Give coin 10 . So far you have given= 20 .Remaining to give 0
minimum change for 21 cents can be achieved using 3 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 20
2 : Give coin 10 . So far you have given= 11 .Remaining to give 10
3 : Give coin 10 . So far you have given= 21 .Remaining to give 0
minimum change for 22 cents can be achieved using 3 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 20
2 : Give coin 10 . So far you have given= 12 .Remaining to give 10
3 : Give coin 10 . So far you have given= 22 .Remaining to give 0
minimum change for 23 cents can be achieved using 4 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 22
2 : Give coin 2 . So far you have given= 3 .Remaining to give 20
3 : Give coin 10 . So far you have given= 13 .Remaining to give 10
4 : Give coin 10 . So far you have given= 23 .Remaining to give 0
minimum change for 24 cents can be achieved using 1 coins.
1 : Give coin 24 . So far you have given= 24 .Remaining to give 0
minimum change for 25 cents can be achieved using 2 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 24
2 : Give coin 24 . So far you have given= 25 .Remaining to give 0
minimum change for 26 cents can be achieved using 2 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 24
2 : Give coin 24 . So far you have given= 26 .Remaining to give 0
minimum change for 27 cents can be achieved using 3 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 26
2 : Give coin 2 . So far you have given= 3 .Remaining to give 24
3 : Give coin 24 . So far you have given= 27 .Remaining to give 0
minimum change for 28 cents can be achieved using 3 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 26
2 : Give coin 2 . So far you have given= 4 .Remaining to give 24
3 : Give coin 24 . So far you have given= 28 .Remaining to give 0
minimum change for 29 cents can be achieved using 4 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 28
2 : Give coin 2 . So far you have given= 3 .Remaining to give 26
3 : Give coin 2 . So far you have given= 5 .Remaining to give 24
4 : Give coin 24 . So far you have given= 29 .Remaining to give 0
minimum change for 30 cents can be achieved using 1 coins.
1 : Give coin 30 . So far you have given= 30 .Remaining to give 0
minimum change for 31 cents can be achieved using 2 coins.

20.20. LEETCODE PROBLEM 322: COIN CHANGE

1 : Give coin 1 . So far you have given= 1 .Remaining to give 30
2 : Give coin 30 . So far you have given= 31 .Remaining to give 0
minimum change for 32 cents can be achieved using 2 coins.
1 : Give coin 2 . So far you have given= 2 .Remaining to give 30
2 : Give coin 30 . So far you have given= 32 .Remaining to give 0
minimum change for 33 cents can be achieved using 3 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 32
2 : Give coin 2 . So far you have given= 3 .Remaining to give 30
3 : Give coin 30 . So far you have given= 33 .Remaining to give 0
minimum change for 34 cents can be achieved using 2 coins.
1 : Give coin 10 . So far you have given= 10 .Remaining to give 24
2 : Give coin 24 . So far you have given= 34 .Remaining to give 0
WORK = 137

----- Problem 5 -----

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
[0, 1, 2, 3, 4, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 2, 3, 4, 5, 6, 2, 3, 4, 5, 6, 1]
[0, 1, 1, 1, 1, 5, 1, 1, 1, 10, 1, 1, 1, 1, 5, 1, 1, 1, 1, 10, 1, 1, 1, 1, 25]
minimum change for 0 cents can be achieved using 0 coins.
minimum change for 1 cents can be achieved using 1 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 0
minimum change for 2 cents can be achieved using 2 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 1
2 : Give coin 1 . So far you have given= 2 .Remaining to give 0
minimum change for 3 cents can be achieved using 3 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 2
2 : Give coin 1 . So far you have given= 2 .Remaining to give 1
3 : Give coin 1 . So far you have given= 3 .Remaining to give 0
minimum change for 4 cents can be achieved using 4 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 3
2 : Give coin 1 . So far you have given= 2 .Remaining to give 2
3 : Give coin 1 . So far you have given= 3 .Remaining to give 1
4 : Give coin 1 . So far you have given= 4 .Remaining to give 0
minimum change for 5 cents can be achieved using 1 coins.
1 : Give coin 5 . So far you have given= 5 .Remaining to give 0
minimum change for 6 cents can be achieved using 2 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 5
2 : Give coin 5 . So far you have given= 6 .Remaining to give 0
minimum change for 7 cents can be achieved using 3 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 6
2 : Give coin 1 . So far you have given= 2 .Remaining to give 5
3 : Give coin 5 . So far you have given= 7 .Remaining to give 0

minimum change for 8 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 7

2 : Give coin 1 . So far you have given= 2 . Remaining to give 6

3 : Give coin 1 . So far you have given= 3 . Remaining to give 5

4 : Give coin 5 . So far you have given= 8 . Remaining to give 0

minimum change for 9 cents can be achieved using 5 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 8

2 : Give coin 1 . So far you have given= 2 . Remaining to give 7

3 : Give coin 1 . So far you have given= 3 . Remaining to give 6

4 : Give coin 1 . So far you have given= 4 . Remaining to give 5

5 : Give coin 5 . So far you have given= 9 . Remaining to give 0

minimum change for 10 cents can be achieved using 1 coins.

1 : Give coin 10 . So far you have given= 10 . Remaining to give 0

minimum change for 11 cents can be achieved using 2 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 10

2 : Give coin 10 . So far you have given= 11 . Remaining to give 0

minimum change for 12 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 11

2 : Give coin 1 . So far you have given= 2 . Remaining to give 10

3 : Give coin 10 . So far you have given= 12 . Remaining to give 0

minimum change for 13 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 12

2 : Give coin 1 . So far you have given= 2 . Remaining to give 11

3 : Give coin 1 . So far you have given= 3 . Remaining to give 10

4 : Give coin 10 . So far you have given= 13 . Remaining to give 0

minimum change for 14 cents can be achieved using 5 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 13

2 : Give coin 1 . So far you have given= 2 . Remaining to give 12

3 : Give coin 1 . So far you have given= 3 . Remaining to give 11

4 : Give coin 1 . So far you have given= 4 . Remaining to give 10

5 : Give coin 10 . So far you have given= 14 . Remaining to give 0

minimum change for 15 cents can be achieved using 2 coins.

1 : Give coin 5 . So far you have given= 5 . Remaining to give 10

2 : Give coin 10 . So far you have given= 15 . Remaining to give 0

minimum change for 16 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 15

2 : Give coin 5 . So far you have given= 6 . Remaining to give 10

3 : Give coin 10 . So far you have given= 16 . Remaining to give 0

minimum change for 17 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 . Remaining to give 16

2 : Give coin 1 . So far you have given= 2 . Remaining to give 15

20.20. LEETCODE PROBLEM 322: COIN CHANGE

3 : Give coin 5 . So far you have given= 7 .Remaining to give 10
4 : Give coin 10 . So far you have given= 17 .Remaining to give 0
minimum change for 18 cents can be achieved using 5 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 17
2 : Give coin 1 . So far you have given= 2 .Remaining to give 16
3 : Give coin 1 . So far you have given= 3 .Remaining to give 15
4 : Give coin 5 . So far you have given= 8 .Remaining to give 10
5 : Give coin 10 . So far you have given= 18 .Remaining to give 0
minimum change for 19 cents can be achieved using 6 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 18
2 : Give coin 1 . So far you have given= 2 .Remaining to give 17
3 : Give coin 1 . So far you have given= 3 .Remaining to give 16
4 : Give coin 1 . So far you have given= 4 .Remaining to give 15
5 : Give coin 5 . So far you have given= 9 .Remaining to give 10
6 : Give coin 10 . So far you have given= 19 .Remaining to give 0
minimum change for 20 cents can be achieved using 2 coins.
1 : Give coin 10 . So far you have given= 10 .Remaining to give 10
2 : Give coin 10 . So far you have given= 20 .Remaining to give 0
minimum change for 21 cents can be achieved using 3 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 20
2 : Give coin 10 . So far you have given= 11 .Remaining to give 10
3 : Give coin 10 . So far you have given= 21 .Remaining to give 0
minimum change for 22 cents can be achieved using 4 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 21
2 : Give coin 1 . So far you have given= 2 .Remaining to give 20
3 : Give coin 10 . So far you have given= 12 .Remaining to give 10
4 : Give coin 10 . So far you have given= 22 .Remaining to give 0
minimum change for 23 cents can be achieved using 5 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 22
2 : Give coin 1 . So far you have given= 2 .Remaining to give 21
3 : Give coin 1 . So far you have given= 3 .Remaining to give 20
4 : Give coin 10 . So far you have given= 13 .Remaining to give 10
5 : Give coin 10 . So far you have given= 23 .Remaining to give 0
minimum change for 24 cents can be achieved using 6 coins.
1 : Give coin 1 . So far you have given= 1 .Remaining to give 23
2 : Give coin 1 . So far you have given= 2 .Remaining to give 22
3 : Give coin 1 . So far you have given= 3 .Remaining to give 21
4 : Give coin 1 . So far you have given= 4 .Remaining to give 20
5 : Give coin 10 . So far you have given= 14 .Remaining to give 10
6 : Give coin 10 . So far you have given= 24 .Remaining to give 0
minimum change for 25 cents can be achieved using 1 coins.

1 : Give coin 25 . So far you have given= 25 .Remaining to give 0
WORK = 63

----- Problem 6 -----

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]

[0, 1, 2, 3, 4, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 2, 3]

[0, 1, 1, 1, 1, 5, 1, 1, 1, 10, 1, 1, 1, 1, 5, 1]

minimum change for 0 cents can be achieved using 0 coins.

minimum change for 1 cents can be achieved using 1 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 0
minimum change for 2 cents can be achieved using 2 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 1

2 : Give coin 1 . So far you have given= 2 .Remaining to give 0

minimum change for 3 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 2

2 : Give coin 1 . So far you have given= 2 .Remaining to give 1

3 : Give coin 1 . So far you have given= 3 .Remaining to give 0

minimum change for 4 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 3

2 : Give coin 1 . So far you have given= 2 .Remaining to give 2

3 : Give coin 1 . So far you have given= 3 .Remaining to give 1

4 : Give coin 1 . So far you have given= 4 .Remaining to give 0

minimum change for 5 cents can be achieved using 1 coins.

1 : Give coin 5 . So far you have given= 5 .Remaining to give 0

minimum change for 6 cents can be achieved using 2 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 5

2 : Give coin 5 . So far you have given= 6 .Remaining to give 0

minimum change for 7 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 6

2 : Give coin 1 . So far you have given= 2 .Remaining to give 5

3 : Give coin 5 . So far you have given= 7 .Remaining to give 0

minimum change for 8 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 7

2 : Give coin 1 . So far you have given= 2 .Remaining to give 6

3 : Give coin 1 . So far you have given= 3 .Remaining to give 5

4 : Give coin 5 . So far you have given= 8 .Remaining to give 0

minimum change for 9 cents can be achieved using 5 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 8

2 : Give coin 1 . So far you have given= 2 .Remaining to give 7

3 : Give coin 1 . So far you have given= 3 .Remaining to give 6

4 : Give coin 1 . So far you have given= 4 .Remaining to give 5

5 : Give coin 5 . So far you have given= 9 .Remaining to give 0

20.20. LEETCODE PROBLEM 322: COIN CHANGE

minimum change for 10 cents can be achieved using 1 coins.

1 : Give coin 10 . So far you have given= 10 .Remaining to give 0
minimum change for 11 cents can be achieved using 2 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 10

2 : Give coin 10 . So far you have given= 11 .Remaining to give 0
minimum change for 12 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 11

2 : Give coin 1 . So far you have given= 2 .Remaining to give 10

3 : Give coin 10 . So far you have given= 12 .Remaining to give 0
minimum change for 13 cents can be achieved using 4 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 12

2 : Give coin 1 . So far you have given= 2 .Remaining to give 11

3 : Give coin 1 . So far you have given= 3 .Remaining to give 10

4 : Give coin 10 . So far you have given= 13 .Remaining to give 0
minimum change for 14 cents can be achieved using 5 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 13

2 : Give coin 1 . So far you have given= 2 .Remaining to give 12

3 : Give coin 1 . So far you have given= 3 .Remaining to give 11

4 : Give coin 1 . So far you have given= 4 .Remaining to give 10

5 : Give coin 10 . So far you have given= 14 .Remaining to give 0
minimum change for 15 cents can be achieved using 2 coins.

1 : Give coin 5 . So far you have given= 5 .Remaining to give 10

2 : Give coin 10 . So far you have given= 15 .Remaining to give 0
minimum change for 16 cents can be achieved using 3 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 15

2 : Give coin 5 . So far you have given= 6 .Remaining to give 10

3 : Give coin 10 . So far you have given= 16 .Remaining to give 0

WORK = 35

Problem 7 -----

[0]

[0]

[0]

minimum change for 0 cents can be achieved using 0 coins.

Problem 8 -----

[0, 1]

[0, 1]

[0, 1]

minimum change for 0 cents can be achieved using 0 coins.

minimum change for 1 cents can be achieved using 1 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 0

WORK = 1

----- Problem 9 -----

[0, 1, 2]

[0, 1, 2]

[0, 1, 1]

minimum change for 0 cents can be achieved using 0 coins.

minimum change for 1 cents can be achieved using 1 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 0

minimum change for 2 cents can be achieved using 2 coins.

1 : Give coin 1 . So far you have given= 1 .Remaining to give 1

2 : Give coin 1 . So far you have given= 2 .Remaining to give 0

WORK = 2

----- Problem 10 -----

[0, 1, 2, 3]

[0, -1, 1, -1]

[0, -1, 2, -1]

minimum change for 0 cents can be achieved using 0 coins.

minimum change for 1 cents can be achieved using -1 coins.

Change cannot be given for 1 cents

minimum change for 2 cents can be achieved using 1 coins.

1 : Give coin 2 . So far you have given= 2 .Remaining to give 0

minimum change for 3 cents can be achieved using -1 coins.

Change cannot be given for 3 cents

WORK = 2

----- Problem 11 -----

Expected minimal change is 20 . your answer is 20

WORK = 23904

----- Problem 12 -----

Expected minimal change is 8 . your answer is 8

WORK = 444

----- Problem 13 -----

Expected minimal change is 10000 . your answer is 10000

WORK = 10000

Testing L0322Test ENDS

20.20.2 Leetcode output

20.20. LEETCODE PROBLEM 322: COIN CHANGE

322. Coin Change
<https://leetcode.com/problems/coin-change/>

Success Details ↗

Runtime: 6108 ms faster than 5.01% of Python3 online submissions for Coin Change.

Memory Usage: 15 MB less than 29.38% of Python3 online submissions for Coin Change.

```
class Solution:
    # YOU CANNOT CHANGE THIS INTERFACE
    # LEETCODE INTERFACE
    def coinChange(self, coins: List[int], amount: int) -> int:
        # YOU CANNOT CHANGE ANYTHING IN THIS PROCEDURE
        work = [0]
        changes = [] #if change cannot be given, you must put -1 in changes[0]
        show = False
        p = l0322(coins,amount,changes,work,show)
        num_change = len(changes)
        if (num_change == 1):
            if (changes[0] == -1):
                num_change = -1
        return num_change

class l0322:
    def __init__(self, coins: List[int], amount:int, changes:list[int], work:list[int], show:bool):
        self._coins = coins
        self._n = amount
        self._ans = changes
        self._work = work
        self._show = show
        # YOU MUST GENERATE V table and X table
        self._x = []
        self._v = []
        # You can have any number of data structures here
        # MUST EXIST TWO ROUTINES
        self._alg()
        self._get_solution()
```

Figure 20.54: Problem 322: Coin Change. All LeetCode test passes

20.21 House Robber 198

20.21. HOUSE ROBBER 198

<https://leetcode.com/problems/house-robber/>

198. House Robber

Easy 6065 178 Add to List Share

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security system connected and **it will automatically contact the police if two adjacent houses were broken into on the same night.**

Given a list of non-negative integers representing the amount of money of each house, determine the maximum amount of money you can rob tonight **without alerting the police.**

Input: nums = [1, 2, 3, 1]
Output: 4
Explanation: Rob house 1 (money = 1) and then rob house 3 (money = 3).
Total amount you can rob = 1 + 3 = 4. (1, 3)

Input: nums = [2, 7, 9, 3, 1]
Output: 12
Explanation: Rob house 1 (money = 2), rob house 3 (money = 9) and rob house 5 (money = 1).
Total amount you can rob = 2 + 9 + 1 = 12. (2, 9, 1)

```
1 class Solution {  
2     private int[] a;  
3  
4     public int rob(int[] nums) {  
5         a = nums;  
6         return dp();  
7     }  
}
```

Make sure you will store all the items that was stolen in an array. When you add all the elements of this array, it should match to output number

Success Details >

Runtime: 0 ms, faster than 100.00% of Java online submissions for House Robber.

Memory Usage: 36.1 MB, less than 94.60% of Java online submissions for House Robber.

Figure 20.55: All LeetCode test passes. Make sure you will store all the items in an array

20.22 Destination city

1436. Destination City

<https://leetcode.com/problems/destination-city/>

You are given the array `paths`, where `paths[i] = [cityAi, cityBi]` means there exists a direct path going from `cityAi` to `cityBi`. Return the destination city, that is, the city without any path outgoing to another city.

It is guaranteed that the graph of paths forms a line without any loop, therefore, there will be exactly one destination city.

Example 1:

```
Input: paths = [["London","New York"],["New York","Lima"],["Lima","Sao Paulo"]]
Output: "Sao Paulo"
Explanation: Starting at "London" city you will reach "Sao Paulo" city
which is the destination city. Your trip consist of: "London" -> "New
York" -> "Lima" -> "Sao Paulo".
```

Figure 20.56: Problem definition

```
type of k <class 'list'> Value of k = [['London', 'New York'], ['New York', 'Lima'], ['Lima', 'Sao Paulo']]
Dest City Sao Paulo
type of k <class 'list'> Value of k = [['B', 'C'], ['D', 'B'], ['C', 'A']]
Dest City A
type of k <class 'list'> Value of k = [['A', 'Z']]
Dest City Z
ALL TESTS PASSED
Press any key to continue . . .
```

Figure 20.57: Expected output

```
class Solution():
    def __init__(self):
        pass

    ##Keep Leetcode interface
    def destCity(self, paths: List[List[str]])->'str':
        ## NOTHING CAN BE CHANGED HERE
        return self.__alg(paths)
```

[Success](#) [Details >](#)

Runtime: 56 ms, faster than 67.13% of Python3 online submissions for Destination City.

Memory Usage: 13.8 MB, less than 100.00% of Python3 online submissions for Destination City.

Figure 20.58: All tests in LeetCode must pass

20.23 Is Graph Bipartite?

<https://leetcode.com/problems/is-graph-bipartite/>

785. Is Graph Bipartite?

Given an undirected graph, return true if and only if it is bipartite.

Recall that a graph is bipartite if we can split its set of nodes into two independent subsets A and B such that every edge in the graph has one node in A and another node in B.

The graph is given in the following form:
graph[i] is a list of indexes j for which the edge between nodes i and j exists.
Each node is an integer between 0 and graph.length - 1.
There are no self edges or parallel edges:
graph[i] does not contain i, and it doesn't contain any element twice.

Example 1:

Input: [[1,3], [0,2], [1,3], [0,2]]

Output: true

Explanation:

The graph looks like this:



We can divide the vertices into two groups: {0, 2} and {1, 3}.

Example 2:

Input: [[1,2,3], [0,2], [0,1,3], [0,2]]

Output: false

Explanation:

The graph looks like this:



We cannot find a way to divide the set of nodes into two independent subsets.

Figure 20.59: Problem definition

20.23.1 Reading and writing the graph

20.23. IS GRAPH BIPARTITE?

Building graph and Writing dot file

Given an undirected graph,

The graph is given in the following form:
graph[i] is a list of indexes j for which the edge between nodes i and j exists.
Each node is an integer between 0 and graph.length - 1.
There are no self edges or parallel edges:
graph[i] does not contain i, and it doesn't contain any element twice.

Example 1:
Input: [[1, 3], [0, 2], [1, 3], [0, 2]]

The graph looks like this:

0 ----- 1
|
|
3 ----- 2

```
digraph g {  
    edge [dir=none, color=red]  
    0->1  
    0->3  
    1->2  
    2->3  
    label= "[[1, 3], [0, 2], [1, 3], [0, 2]]"  
}
```

b1in.dot

```
[[1, 3], [0, 2], [1, 3], [0, 2]]
```

Figure 20.60: Example 1

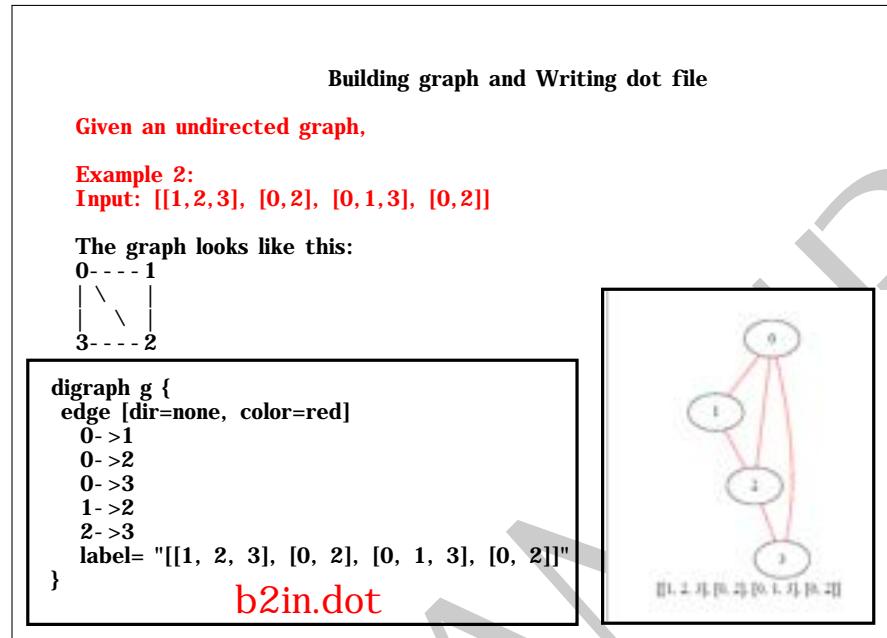


Figure 20.61: Example 2

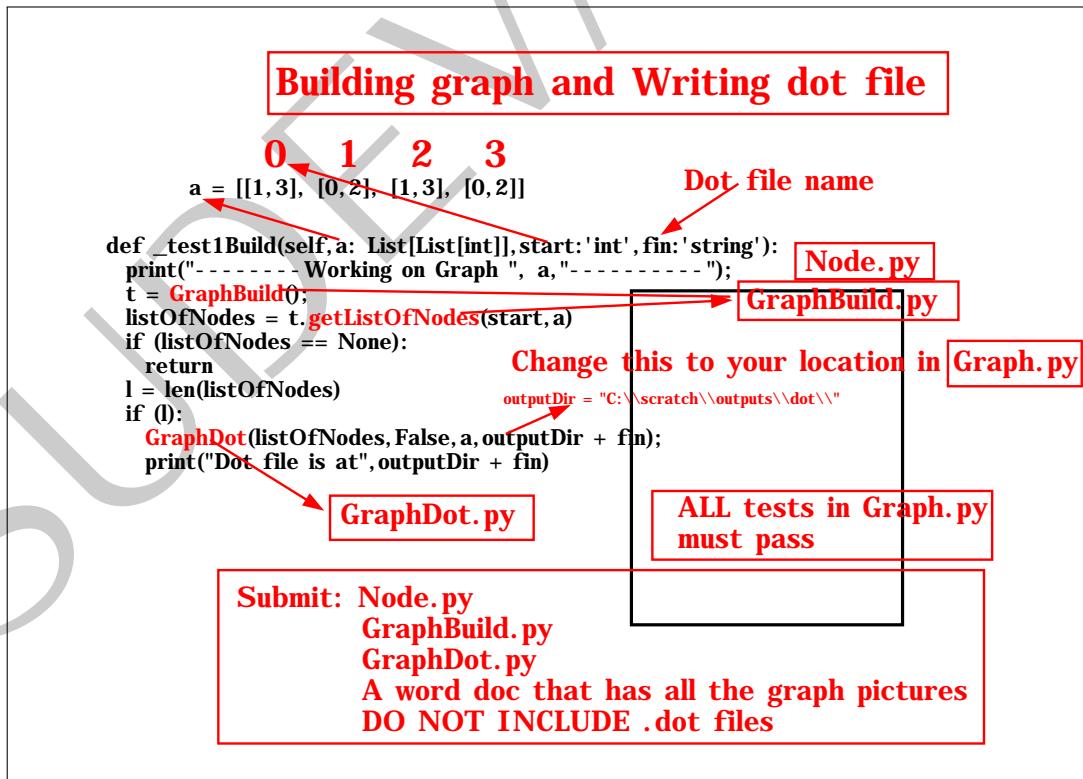


Figure 20.62: What to submit?

20.23. IS GRAPH BIPARTITE?

20.23.2 Graph bipartite solution

Example 1:

Input: [[1,3], [0,2], [1,3], [0,2]]

Output: true

Explanation:

The graph looks like this:

```
0 -.- 1  
| |  
3 -.- 2  
a = [[1,3], [0,2], [1,3], [0,2]]
```

We can divide the vertices into two groups: {0, 2} and {1, 3}.

```
graph g {  
    overlap:false; splines:true  
    edge [style:dotted, weight=10, lens:2]  
    subgraph cluster RED {  
        RED [pos=-1,0%]; color:red /* , style=invis */  
        0 -- RED  
        2 -- RED  
    }  
    subgraph cluster BLUE {  
        BLUE [pos=-1,0%]; color:blue /* , style=invis */  
        1 -- BLUE  
        3 -- BLUE  
    }  
    edge [style="", weight=1, len=1]  
    0 -.- 1  
    0 -.- 3  
    1 -.- 2  
    2 -.- 3  
    label: "[[1,3], [0,2], [1,3], [0,2]]"]  
}
```

Example 2:

Input: [[1,2,3], [0,2], [0,1,3], [0,2]]

Output: false

Explanation:

The graph looks like this:

```
0 -.- 1  
| |  
3 -.- 2
```

We cannot find a way to divide
the set of nodes into two independent subsets.

785. Is Graph Bipartite?

<https://leetcode.com/problems/is-graph-bipartite/>

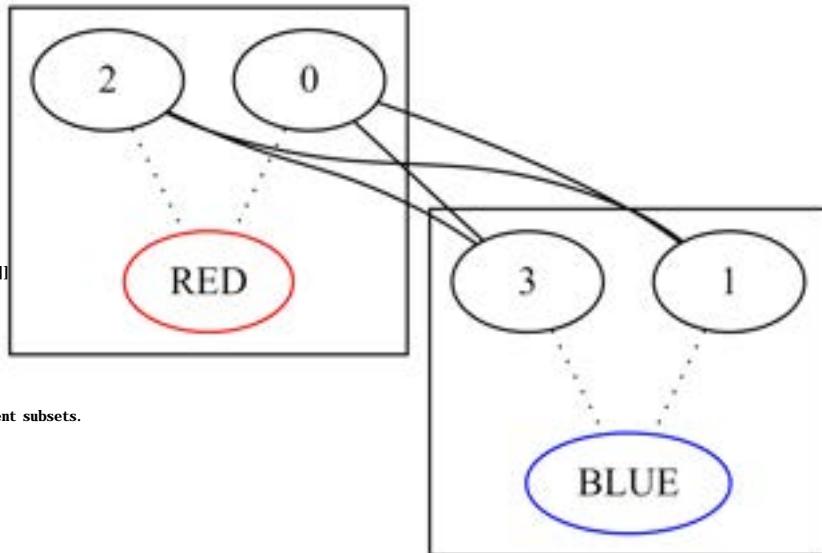


Figure 20.63: Bipartite problem definition

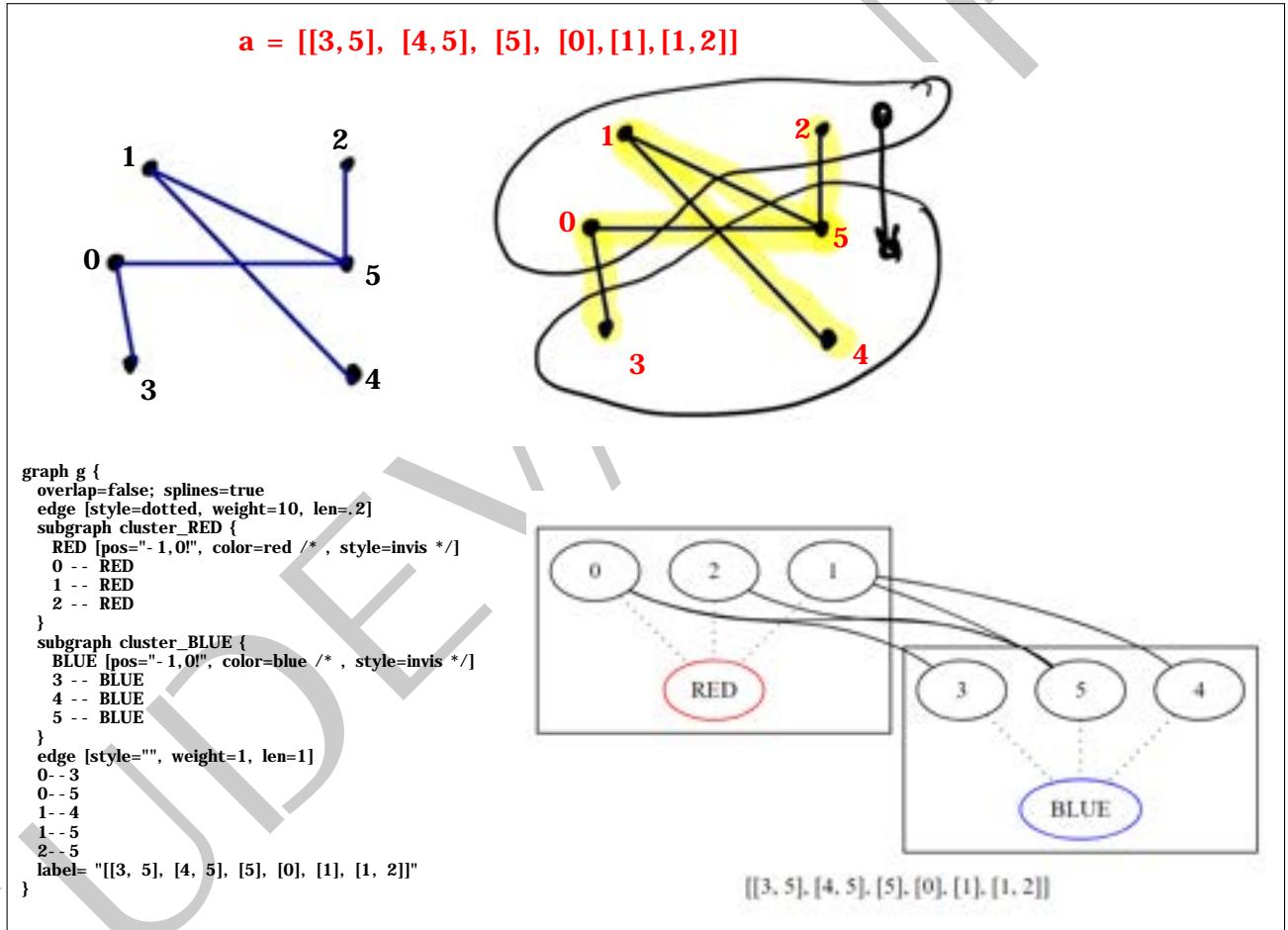


Figure 20.64: Another bipartite graph

20.23. IS GRAPH BIPARTITE?

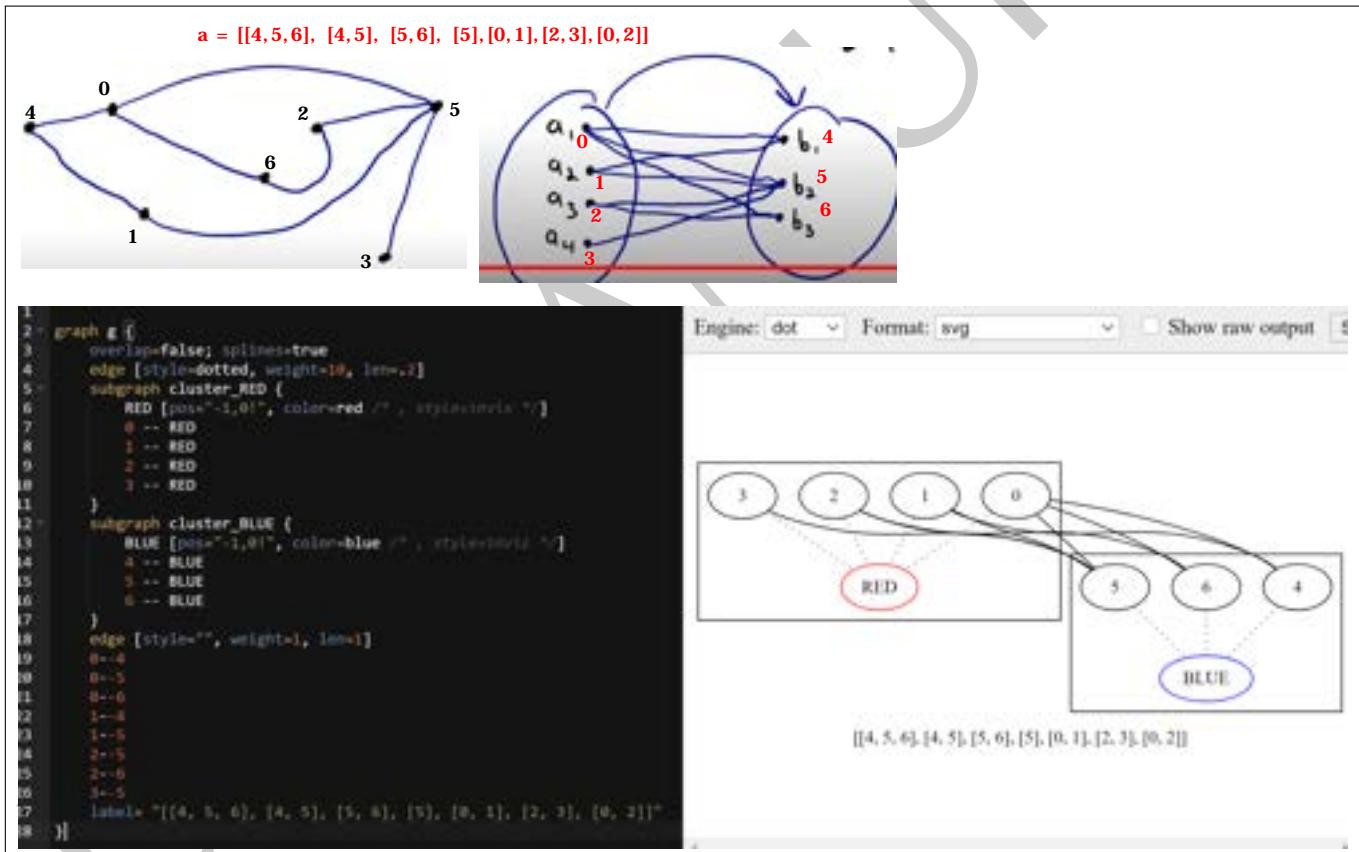


Figure 20.65: Another bipartite graph

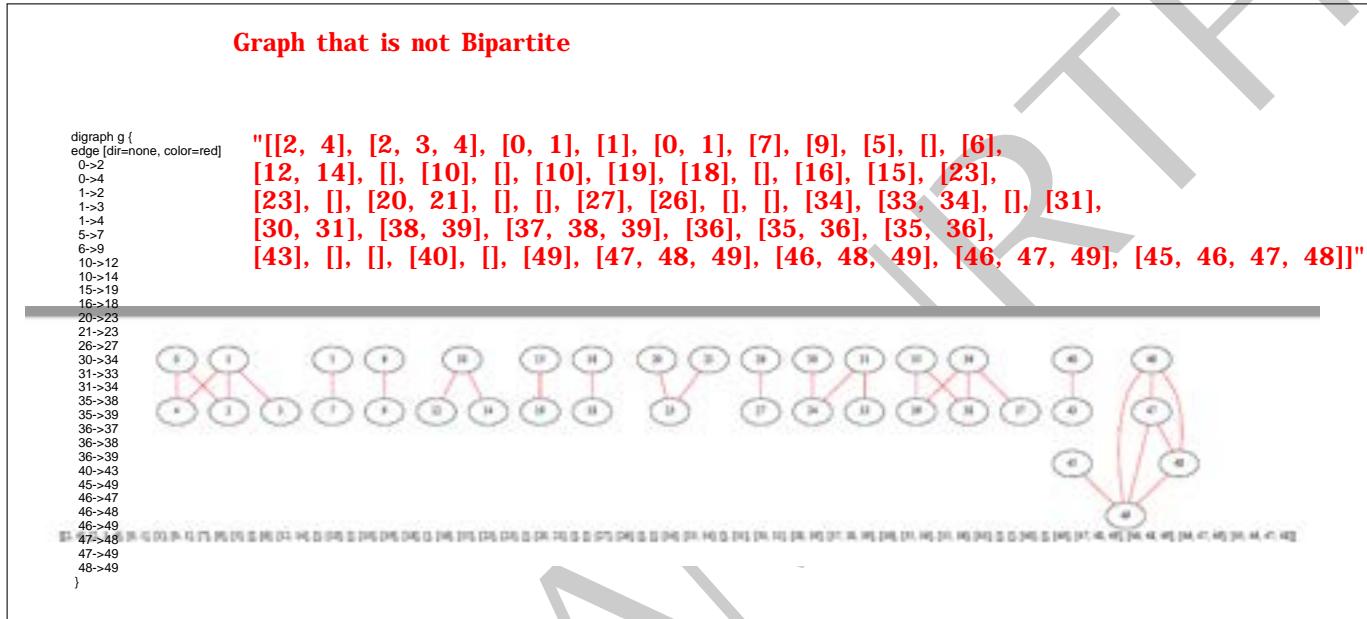


Figure 20.66: Not a bipartite graph

```

class Solution:
    ## KEEP same interface as Leetcode
    ## YOU CANNOT CHANGE anything
    def isBipartite(self, graph: List[List[int]]) -> bool:
        return self.__alg(graph)

    ##      WRITE CODE BELOW
    ##      You can have as many number of private functions
    ##      and private variables
    def __alg(self, graph: List[List[int]]) -> bool:

```

Success [Details >](#)

Runtime: 196 ms, faster than 38.94% of Python3 online submissions for Is Graph Bipartite?.

Memory Usage: 14.2 MB, less than 9.09% of Python3 online submissions for Is Graph Bipartite?.

Figure 20.67: All tests in LeetCode must pass

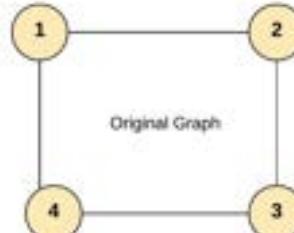
20.24 Clone an undirected graph

<https://leetcode.com/problems/clone-graph/>

clone connected undirected graph.

Input: adjList = [[2,4],[1,3],[2,4],[1,3]]
Output: [[2,4],[1,3],[2,4],[1,3]]

Explanation: There are 4 nodes in the graph.
1st node (val = 1)'s neighbors are 2nd node (val = 2) and 4th node (val = 4).
2nd node (val = 2)'s neighbors are 1st node (val = 1) and 3rd node (val = 3).
3rd node (val = 3)'s neighbors are 2nd node (val = 2) and 4th node (val = 4).
4th node (val = 4)'s neighbors are 1st node (val = 1) and 3rd node (val = 3).



Original Graph



Input: adjList = [[]]
Output: [[]]

Explanation: Note that the input contains one empty list. The graph consists of only one node with val = 1 and it does not have any neighbors.



Input: adjList = [[2],[1]]
Output: [[2],[1]]

Input: adjList = []
Output: []

Explanation: This is an empty graph, it does not have any nodes.

Figure 20.68: Problem definition

```
----- Working on Graph [[2, 4], [1, 3], [2, 4], [1, 3]] -----
Original Graph is in C:\scratch\outputs\dot\1in.txt
Cloned Graph is in C:\scratch\outputs\dot\1out.txt
Cloned Graph is equivalent to original graph
----- Working on Graph [] -----
Original Graph is in C:\scratch\outputs\dot\2in.txt
Cloned Graph is in C:\scratch\outputs\dot\2out.txt
Cloned Graph is equivalent to original graph
----- Working on Graph [] -----
----- Working on Graph [[2], [1]] -----
Original Graph is in C:\scratch\outputs\dot\4in.txt
Cloned Graph is in C:\scratch\outputs\dot\4out.txt
Cloned Graph is equivalent to original graph
ALL TESTS PASSED
Press any key to continue . . .
```

Figure 20.69: Expected output

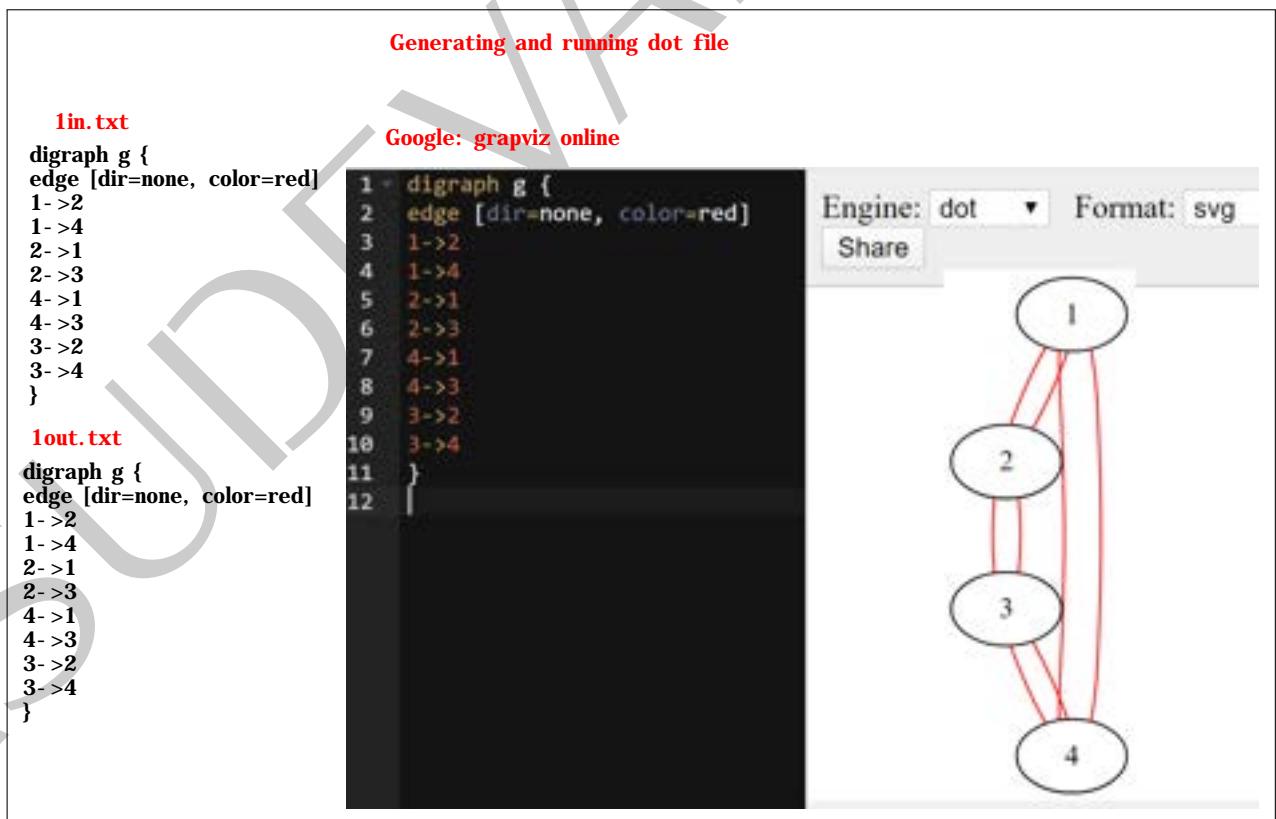


Figure 20.70: dot file and output

20.24. CLONE AN UNDIRECTED GRAPH

The screenshot shows a LeetCode submission for the problem "Clone an Undirected Graph". The code is as follows:

```
class Solution:
    def cloneGraph(self, node: 'Node') -> 'Node':
        n = self._clone(node)
        return n

    def _clone(self, firstNode: 'Node') -> 'Node':
```

Below the code, there is a "Success" message and a "Details" link. Performance metrics are listed below:

Runtime: 28 ms, faster than 98.24% of Python3 online submissions for Clone Graph.

Memory Usage: 14 MB, less than 74.07% of Python3 online submissions for Clone Graph.

Figure 20.71: All tests in LeetCode must pass

20.25 Course schedule

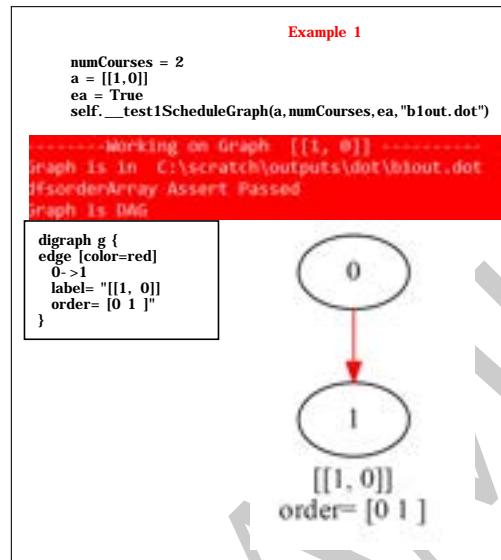


Figure 20.72: Problem definition

20.25. COURSE SCHEDULE

Example 2

```
numCourses = 2
a = [[1, 0], [0, 1]]
ea = False
self._test1ScheduleGraph(a, numCourses, ea, "b2out.dot")
```

----- Working on Graph [[1, 0], [0, 1]] -----
Graph is in C:\scratch\outputs\dot\b2out.dot
Graph is Not a DAG. dfsorderArray has no meaning.

graph LR; 0((0)) --> 1((1)); 1((1)) --> 0((0))

[[1, 0], [0, 1]]
order= []

Figure 20.73: Problem definition

Example 3

```
numCourses = 4
a = [[1, 0], [2, 0], [3, 1], [3, 2]]
ea = True
self._test1ScheduleGraph(a, numCourses, ea, "b3out.dot")
```

----- Working on Graph [[1, 0], [2, 0], [3, 1], [3, 2]] -----
Graph is in C:\scratch\outputs\dot\b3out.dot
dfsorderArray Assert Passed
Graph is DAG.

graph LR; 0((0)) --> 2((2)); 0((0)) --> 3((3)); 1((1)) --> 3((3)); 2((2)) --> 3((3))

[[1, 0], [2, 0], [3, 1], [3, 2]]
order= [0 2 1 3]

Figure 20.74: Problem definition

207. Course Schedule

<https://leetcode.com/problems/course-schedule/>

There are a total of numCourses courses you have to take, labeled from 0 to numCourses- 1.

Some courses may have prerequisites, for example to take course 0 you have to first take course 1, which is expressed as a pair: [0,1]

Given the total number of courses and a list of prerequisite pairs, is it possible for you to finish all courses?

Example 1:

Input: numCourses = 2, prerequisites = [[1,0]]
 Output: true
 Explanation: There are a total of 2 courses to take.
 To take course 1 you should have finished course 0. So it is possible.

Example 2:

Input: numCourses = 2, prerequisites = [[1,0],[0,1]]
 Output: false
 Explanation: There are a total of 2 courses to take.
 To take course 1 you should have finished course 0,
 and to take course 0 you should
 also have finished course 1. So it is impossible.

Figure 20.75: LeetCode problem definition

207. Course Schedule

```
class Solution:
    def canFinish(self, numCourses: int, prerequisites: List[List[int]]) -> bool:
        return self.__alg(numCourses, prerequisites)
```

[Success](#) [Details >](#)

Runtime: 192 ms, faster than 22.93% of Python3 online submissions for Course Schedule.

Memory Usage: 18 MB, less than 6.12% of Python3 online submissions for Course Schedule.

Figure 20.76: All tests in LeetCode must pass

210. Course Schedule II

<https://leetcode.com/problems/course-schedule-ii/>

There are a total of n courses you have to take, labeled from 0 to $n - 1$.

Some courses may have prerequisites, for example to take course 0 you have to first take course 1, which is expressed as a pair: [0, 1]

Given the total number of courses and a list of prerequisite pairs, return the ordering of courses you should take to finish all courses.

There may be multiple correct orders, you just need to return one of them. If it is impossible to finish all courses, return an empty array.

Example 1:

Input: 2, [[1, 0]]
 Output: [0, 1]
 Explanation: There are a total of 2 courses to take.
 To take course 1 you should have finished
 course 0. So the correct course order is [0, 1].

Example 2:

Input: 4, [[1, 0], [2, 0], [3, 1], [3, 2]]
 Output: [0, 1, 2, 3] or [0, 2, 1, 3]
 Explanation: There are a total of 4 courses to take.
 To take course 3 you should have finished both
 courses 1 and 2. Both courses 1 and 2 should be taken
 after you finished course 0.
 So one correct course order is [0, 1, 2, 3].
 Another correct ordering is [0, 2, 1, 3].

Figure 20.77: LeetCode problem definition

210. Course Schedule II

```
class Solution:
    def findOrder(self, numCourses: int, prerequisites: List[List[int]]) -> List[int]:
        dfsorderArray = self.__alg(numCourses, prerequisites)
        return dfsorderArray
```

Success Details >

Runtime: 272 ms, faster than 8.66% of Python3 online submissions for Course Schedule II.

Memory Usage: 17.9 MB, less than 7.14% of Python3 online submissions for Course Schedule II.

Figure 20.78: All tests in LeetCode must pass

20.26 Shortest path from a given source

<https://www.hackerrank.com/challenges/dijkstrashortreach/problem>

Dijkstra: Shortest Reach 2

Given an **undirected graph** and a **starting node**, determine the lengths of the shortest paths from the starting node to all other nodes in the graph.

If a node is unreachable, its distance is **-1**.

Nodes will be numbered consecutively from **1 to n**,
and edges will have varying distances or lengths.

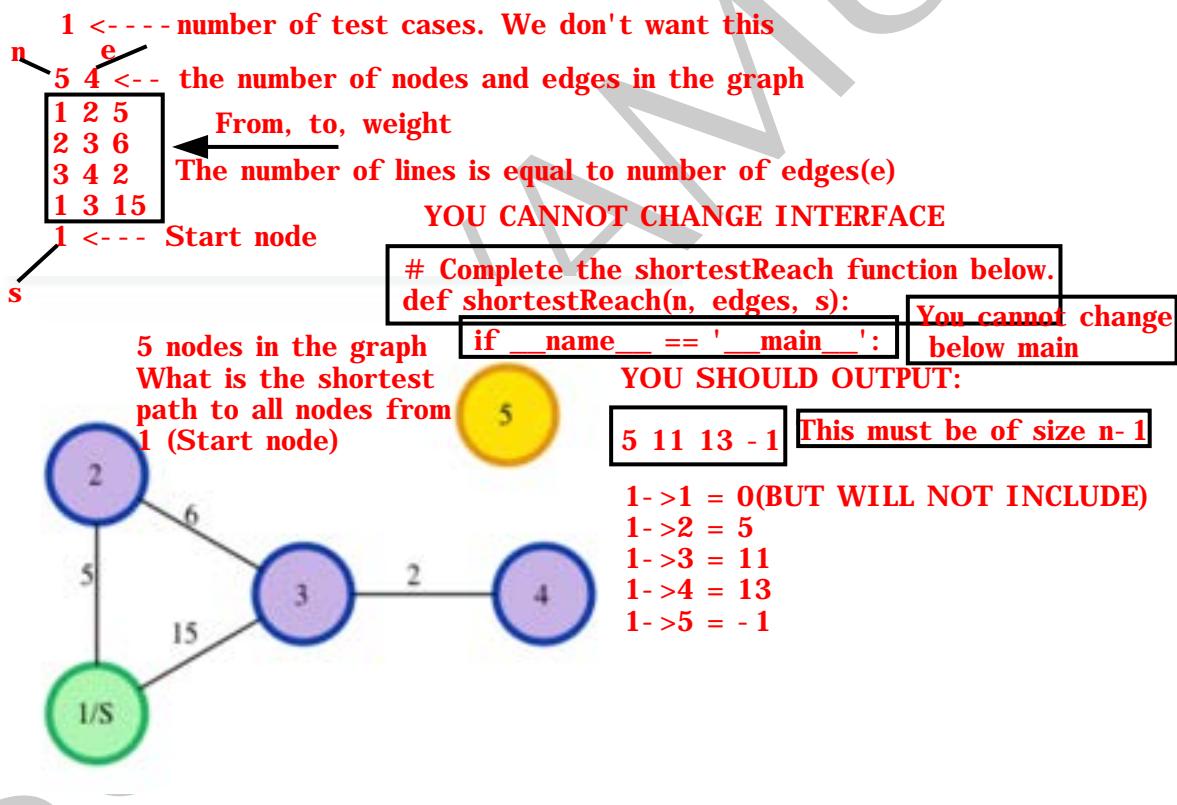


Figure 20.79: Problem definition

20.26. SHORTEST PATH FROM A GIVEN SOURCE

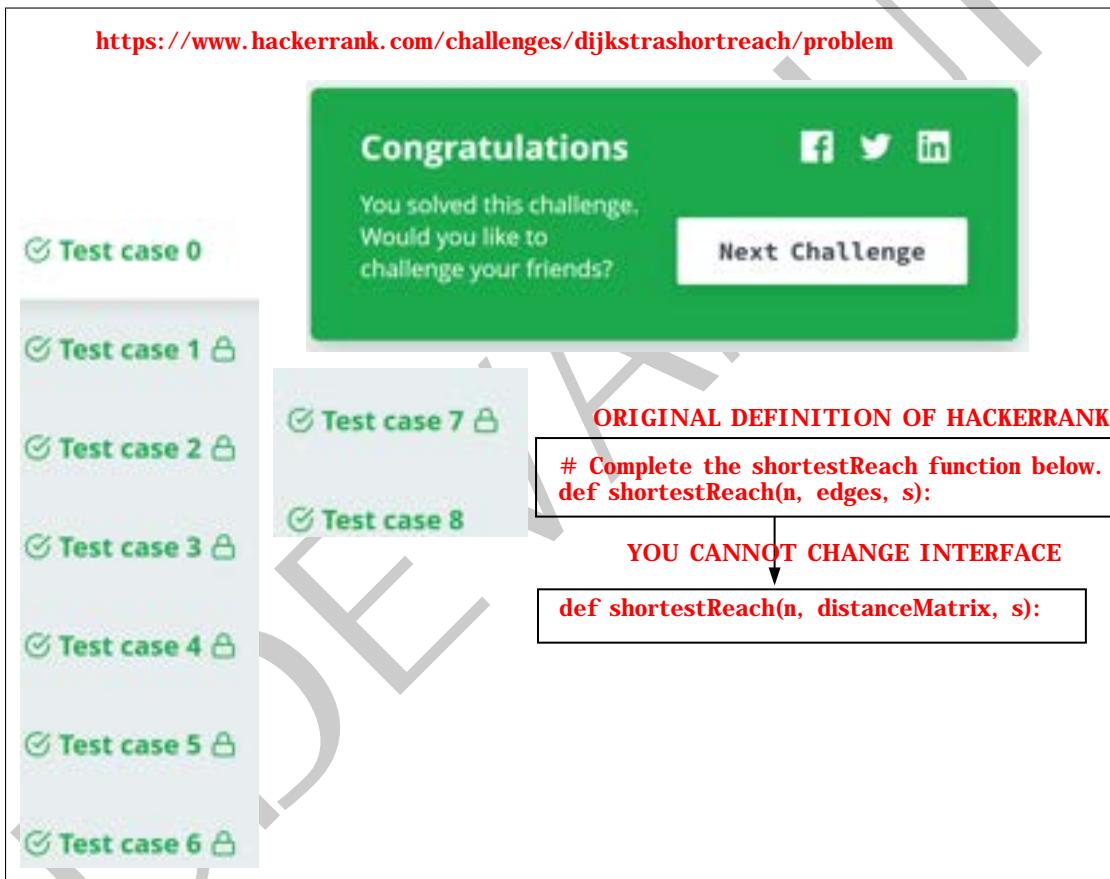


Figure 20.80: All tests in Hacker rank passed by changing definition. You should not do this

20.27 Grow or Shrink

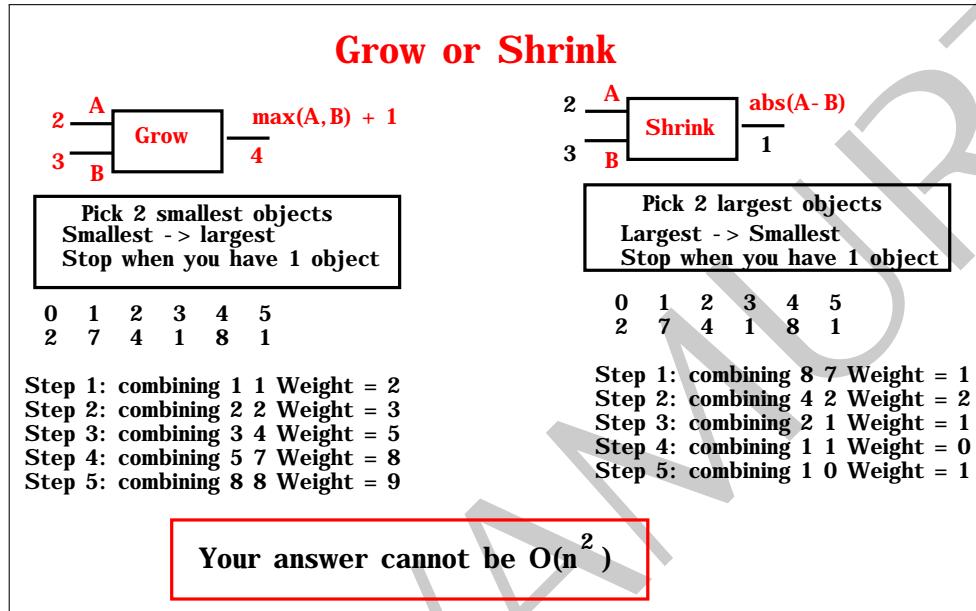


Figure 20.81: Grow or Shrink

20.28 Selection Day Puzzle

20.28. SELECTION DAY PUZZLE

Selection Day Puzzle

1. There are $N = 25$ students
2. We need to pick $S = 3$ fastest students in a running race competition
3. We have a racing track of capacity $RT = 5$
4. We don't have a watch
5. All students runs at a different phase.

Name of the student

Time taken to run

What is the minimum number of races needed to identify first, second and third positions?

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
5	11	4	15	23	0	10	19	17	1	24	6	12	20	7	8	22	3	21	14	18	16	9	13	2

The fastest 3 students after X races are: 5 9 24

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	20	12	11	23	15	3	14	24	22	18	7	17	6	13	21	16	19	8	5	10	1	2	4	9

The fastest 3 students after X races are: 0 21 22

1. How many races are required?
2. if (show == True)

Your output must be such that, any idiot, executes your steps should be able to conduct the race.

Figure 20.82: Selection day puzzle

20.29 Merging Communities

Implement Merging Communities as shown in figures 20.83.

The screenshot shows a challenge interface from Hackerrank. At the top, a red URL is displayed: `#https://www.hackerrank.com/challenges/merging-communities/problem?isFullScreen=true`. Below the URL, there is a large red box containing Python code. Inside this box, a red arrow points to the first line of the class definition with the text "YOU WRITE ONLY THIS CLASS". Another red arrow points to the line `s = SUSF(N)` with the text "Nothing can be changed below. You must use this code as is". A third red arrow points to the line `s.Union(a,b)` with the text "Both written in SUSF class". To the right of the code box, there is a "Congratulations" message: "You solved this challenge. Would you like to challenge your friends?", a "Next Challenge" button, and social sharing icons for Facebook, Twitter, and LinkedIn. Below the code area, there is a table showing test cases and their expected outputs:

Test Case	Input	Output
Test case 3	3 0 1	M 1 2
Test case 4	3 0 2	M 2 3
Test case 5	3 0 3	M 1 2
Test case 6	3 1	
Test case 7	3 2	
Test case 8	3 3	
Test case 9	3 3	

Figure 20.83: Merging Communities

20.30. LEETCODE 643. MAXIMUM AVERAGE SUBARRAY I

20.30 LeetCode 643. Maximum Average Subarray I

643. Maximum Average Subarray I

<https://leetcode.com/problems/maximum-average-subarray-i/>

You are given an integer array `nums` consisting of n elements, and an integer k .
Find a contiguous subarray whose length is equal to k that has the maximum average value and return this value.

`s = L0643("name", a, k, d, work1, self._show)`

Size of $n = 9$ Sub array size - 3

0	1	2	3	4	5	6	7	8
1	2	3	1	4	5	2	3	6

`SUM(a[0], a[1], a[2]) = 6 average = 2.0`
`SUM(a[1], a[2], a[3]) = 6 average = 2.0`
`SUM(a[2], a[3], a[4]) = 8 average = 2.6666666666666665`
`SUM(a[3], a[4], a[5]) = 10 average = 3.3333333333333335`
`SUM(a[4], a[5], a[6]) = 11 average = 3.6666666666666665`
`SUM(a[5], a[6], a[7]) = 10 average = 3.3333333333333335`
`SUM(a[6], a[7], a[8]) = 11 average = 3.6666666666666665`

`d = [4, 5, 6]`

max average = 3.6666666666666665

Figure 20.84: Problem definition

20.30.1 Expected output

$O(n*k)$ $\Theta(1)$

0	1	2	3	4	5	6	7	8
1	2	3	1	4	5	2	3	6

$\Theta(n)$ $O(1)$

```
for i=0 to n-k:
    sum = 0
    for j=i to i+k-1:
        sum += a[j]
    avg = sum / k
    if avg > max_avg:
        max_avg = avg
        max_index = i
print(max_avg)
print(max_index)
```

```
n = 9
k = 3
max_avg = 0
max_index = 0
for i=0 to n-k:
    sum = 0
    for j=i to i+k-1:
        sum += a[j]
    avg = sum / k
    if avg > max_avg:
        max_avg = avg
        max_index = i
print(max_avg)
print(max_index)
```

Figure 20.85: Expected output

20.30.2 Leetcode output

643. Maximum Average Subarray I

<https://leetcode.com/problems/maximum-average-subarray-i/>

```
s = Exam("NKTimeConstantSpace", nums, k, d, work0, False)
```

Time Limit Exceeded Details >

Last executed input: [1132, 2010, 334, 8891, 9799, 6196, 1212, 7034]

```
s = Exam("NTimeConstSpace", nums, k, d, work0, False)
```

Success Details >

Runtime: 6293 ms, faster than 5.00% of Python3 online submissions for Maximum Average Subarray I.

Memory Usage: 29.7 MB, less than 5.15% of Python3 online submissions for Maximum Average Subarray I.

Figure 20.86: Problem 643. Maximum Average Subarray I

20.31. LEETCODE PROBLEM 239. SLIDING WINDOW MAXIMUM

20.31 LeetCode Problem 239. Sliding Window Maximum

239. Sliding Window Maximum

<https://leetcode.com/problems/sliding-window-maximum/>

You are given an array of integers `nums`, there is a sliding window of size `k` which is moving from the very left of the array to the very right.
You can only see the `k` numbers in the window.
Each time the sliding window moves right by one position.

Return the max sliding window.

`s = L0239("name", a, k, d, work1, self, _show)`

<p>Size of <code>n</code> = 9 Sub array size - 3</p> <table border="1" style="margin-left: auto; margin-right: auto;"><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr><tr><td>1</td><td>2</td><td>3</td><td>1</td><td>4</td><td>5</td><td>2</td><td>3</td><td>6</td></tr></table> <p><code>Max(a[0], a[1], a[2]) = 3</code> <code>Max(a[1], a[2], a[3]) = 3</code> <code>Max(a[2], a[3], a[4]) = 4</code> <code>Max(a[3], a[4], a[5]) = 5</code> <code>Max(a[4], a[5], a[6]) = 5</code> <code>Max(a[5], a[6], a[7]) = 5</code> <code>Max(a[6], a[7], a[8]) = 6</code></p> <p><code>d = [3, 3, 4, 5, 5, 5, 6]</code></p>	0	1	2	3	4	5	6	7	8	1	2	3	1	4	5	2	3	6	<p>Size of <code>n</code> = 10 Sub array size - 4</p> <table border="1" style="margin-left: auto; margin-right: auto;"><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr></table> <p><code>Max(a[0], a[1], a[2], a[3]) = 3</code> <code>Max(a[1], a[2], a[3], a[4]) = 4</code> <code>Max(a[2], a[3], a[4], a[5]) = 5</code> <code>Max(a[3], a[4], a[5], a[6]) = 6</code> <code>Max(a[4], a[5], a[6], a[7]) = 7</code> <code>Max(a[5], a[6], a[7], a[8]) = 8</code> <code>Max(a[6], a[7], a[8], a[9]) = 9</code></p> <p><code>d = [3, 4, 5, 6, 7, 8, 9]</code></p>	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8																															
1	2	3	1	4	5	2	3	6																															
0	1	2	3	4	5	6	7	8	9																														
0	1	2	3	4	5	6	7	8	9																														

Figure 20.87: Problem definition

20.31.1 Expected output

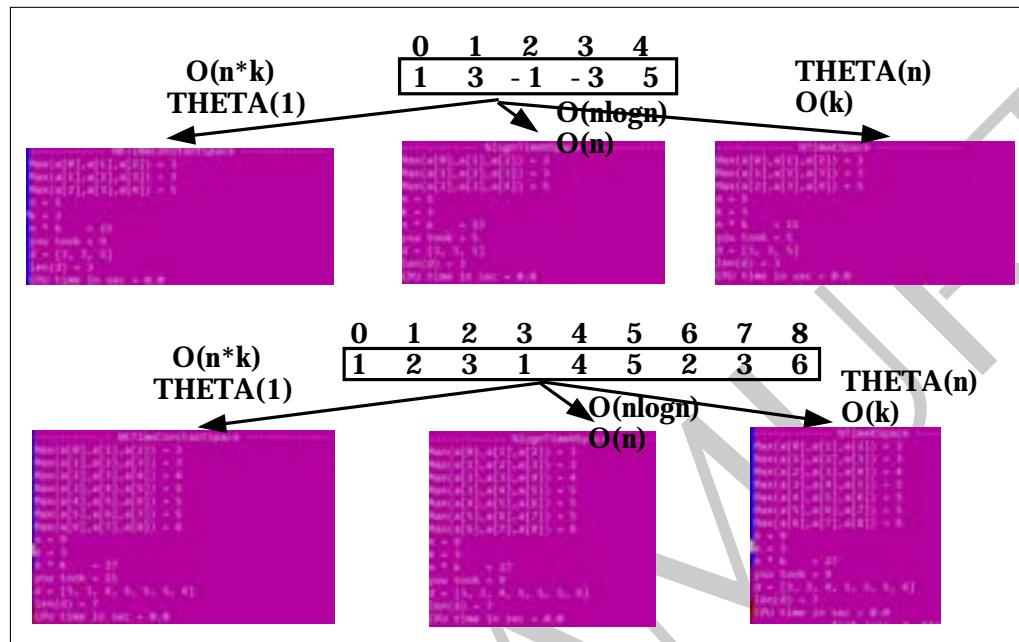


Figure 20.88: Expected output

20.31.2 Leetcode output

239. Sliding Window Maximum

<https://leetcode.com/problems/sliding-window-maximum/>

s = Exam("NKTimeConstantSpace", nums, k, d, work0, False)

Time Limit Exceeded Details »
Last executed input: 143,8457,6354,4398,9169,5724,5478,5958,5902,4484,5785,8149,3281,4827,9961,493,2995,19, 34008 View 413

s = Exam("NlognTimeNSpace", nums, k, d, work0, False)

Success Details »
Runtime: 2052 ms, faster than 5.00% of Python3 online submissions for Sliding Window Maximum.
Memory Usage: 37.1 MB, less than 9.08% of Python3 online submissions for Sliding Window Maximum.
Next challenges:

s = Exam("NTimeKSpace", nums, k, d, work0, False)

Success Details »
Runtime: 1357 ms, faster than 44.17% of Python3 online submissions for Sliding Window Maximum.
Memory Usage: 33.7 MB, less than 12.96% of Python3 online submissions for Sliding Window Maximum.

Figure 20.89: Problem 239. Sliding Window Maximum LeetCode

20.31.3 Geeks for Geeks output

Sliding Window Maximum (Maximum of all subarrays of size K)
<https://www.geeksforgeeks.org/sliding-window-maximum-maximum-of-all-subarrays-of-size-k/>

```
from collections import deque  
import heapq
```

Without this g2g will not run

```
s = Exam("NKTimeConstantSpace", arr, k, d, work0, False)
```

Runtime Error 

Test Cases Passed:

56 / 72

Time Limit Exceeded

```
s = Exam("NlognTimeNSpace", arr, k, d, work0, False)
```

Runtime Error 

Test Cases Passed:

56 / 72

Time Limit Exceeded

```
s = Exam("NTimeKSpace", arr, k, d, work0, False)
```

Problem Solved Successfully 

 You get marks only for the first correct submission if you solve the problem without viewing the full solution.

Test Cases Passed:

72 / 72

Your Total Score:

5

Total Time Taken:

1.91

Correct Submission Count:

2

Figure 20.90: Problem 239. Sliding Window Maximum Geeks for Geeks