Sinhgad Institutes

Sinhgad Technical Education Society's

SINHGAD ACADEMY OF ENGINEERING,

PUNE-411048

DEPARTMENT OF COMPUTER ENGINEERING

Laboratory Practice II

Course :310258

Prepared by:

- Mrs.A.P.Pimpalkar
- Ms.S.M.Shelke
- Mr.G.S.Nikam
- Mr.V.K.Sambhar
- Mrs.V.S.Khandagale
- Ms. S.A. Mhaske

Vision

उत्तमपुरुषान् उत्तमाभियंत्रृन् निर्मातुं कटीबध्दः वयम् !

"We are committed to produce not only good engineersbut good human beings, also."

Mission

"Holistic development of students and teachers is what we believe in and work for. We strive to achieve this by imbibing a unique value system, transparent work culture, excellent academic and physical environment conducive to learning, creativity and technology transfer. Our mandate is to generate, preserve and share knowledge for developing a vibrant society."

Department of Computer Engineering

Vision

"To build the Department as a Centre of Excellence for students in Computer Engineering."

Mission

"We believe in developing value based system for student and staff by providing healthy and transparent work culture to cultivate new ideas in the field of engineering and technology which will contribute to build a vibrant Society."

Programme Outcomes (POs)

PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. Problem analysis: Identify, formulate, research literature, and analyze complex engineering problemsreaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professionalengineering practice.

PO7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. Project management and finance: Demonstrate knowledge and understanding of the engineeringand management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Programme Specific Outcomes (PSOs)

Computer Engineering graduate will be able to,

PSO1: **Project Development:** Successfully complete hardware and/or software related system or application projects, using the phases of project development life cycle to meet the requirements of service and product industries; government projects; and automate other engineering stream projects.

PSO2: Domain Expertise: Demonstrate effective application of knowledge gained from different computer domains like, data structures, data bases, operating systems, computer networks, security, parallel programming, in project development, research and higher education.

PSO3: Career Development: Achieve successful Career and Entrepreneurship- The ability to employ modern computer languages, environments, and platforms in creating innovative career paths to be an entrepreneur, and a zest for higher studies

Savitribai Phule Pune University
Third Year of Computer Engineering (2019 Course)
310258:Laboratory Practice II

Home

Teaching Scheme	Credit: 02	Examination Scheme and Marks
Practical: 04 Hours/Week		**Term Work: 50 Marks** **Practical: 25 Marks**

Companion Course: Artificial Intelligence (310253), Elective II (310254)

Course Objectives:

- To learn and apply various search strategies for AI
- To Formalize and implement constraints in search problems
- To understand the concepts of Information Security / Augmented and Virtual Reality/Cloud Computing/Software Modeling and Architectures

Course Outcomes:

On completion of the course, learner will be able to

- **Artificial Intelligence**
  - **CO1:** Design a system using different informed search / uninformed search or heuristic approaches
  - **CO2:** Apply basic principles of AI in solutions that require problem solving, inference, perception, knowledge representation, and learning
  - **CO3:** Design and develop an interactive AI application
- **Information Security**
  - **CO4:** Use tools and techniques in the area of Information Security
  - **CO5:** Use the cryptographic techniques for problem solving
  - **CO6:** Design and develop security solution

OR

- **Augmented and Virtual Reality**
  - **CO4:** Use tools and techniques in the area of Augmented and Virtual Reality
  - **CO5:** Use the representing and rendering system for problem solving
  - **CO6:** Design and develop ARVR applications

OR

- **Cloud Computing**
  - **CO4:** Use tools and techniques in the area of Cloud Computing
  - **CO5:** Use cloud computing services for problem solving
  - **CO6:** Design and develop applications on cloud

OR

- **Software Modeling and Architectures**
  - **CO4:** Use tools and techniques in the area Software Modeling and Architectures
  - **CO5:** Use the knowledge of Software Modeling and Architectures for problem solving
  - **CO6:** Design and develop applications using UML as fundamental tool

Guidelines for Instructor's Manual

The instructor‘s manual is to be developed as a reference and hands-on resource. It should include prologue (about University/program/ institute/ department/foreword/ preface), curriculum of the course, conduction and Assessment guidelines, topics under consideration, concept, objectives, outcomes, set of typical applications/assignments/ guidelines, and references.

Guidelines for Student's Laboratory Journal

The laboratory assignments are to be submitted by student in the form of journal. Journal consists of Certificate, table of contents, and handwritten write-up of each assignment (Title, Date of Completion, Objectives, Problem Statement, Software and Hardware requirements, Assessment grade/marks and assessor's sign, Theory- Concept in brief, algorithm, flowchart, test cases, Test Data Set(if applicable), mathematical model (if applicable), conclusion/analysis. Program codes with sample output of all performed assignments are to be submitted as softcopy. As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers as part of write-ups and

program listing to journal must be avoided. Use of DVD containing students programs maintained by Laboratory In-charge is highly encouraged. For reference one or two journals may be maintained with program prints in the Laboratory.

Guidelines for Laboratory /Term Work Assessment

Continuous assessment of laboratory work should be based on overall performance of Laboratory assignments by a student. Each Laboratory assignment assessment will assign grade/marks based on parameters, such as timely completion, performance, innovation, efficient codes, punctuality and

Guidelines for Practical Examination

Problem statements must be decided jointly by the internal examiner and external examiner. During practical assessment, maximum weightage should be given to satisfactory implementation of the problem statement. Relevant questions may be asked at the time of evaluation to test the student's understanding of the fundamentals, effective and efficient implementation. This will encourage, transparent evaluation and fair approach, and hence will not create any uncertainty or doubt in the minds of the students. So, adhering to these principles will consummate our team efforts to the promising start of student's academics.

Guidelines for Laboratory Conduction

The instructor is expected to frame the assignments by understanding the prerequisites, technological aspects, utility and recent trends related to the topic. The assignment framing policy need to address the average students and inclusive of an element to attract and promote the intelligent students. Use of open source software is encouraged. Based on the concepts learned. Instructor may also set one assignment or mini-project that is suitable to respective branch beyond the scope of syllabus.

Operating System recommended :- 64-bit Windows OS and Linux

Programming tools recommended: -

Information Security : - C/C++/Java

Augmented and Virtual Reality :- Unity, C#, Blender, VRTK, ARTK, Vuforia

VR Devices: HTC Vive, Google Daydream and Samsung gear VR.

Software Modeling and Architectures:-Front end:HTML5, Bootstarp, JQuery, JS etc.

Backend: MySQL /MongoDB/NodeJS

Virtual Laboratory:

Software Modeling and Architectures : http://vlabs.iitkgp.ernet.in/se

Information Security : http://cse29-iiith.vlabs.ac.in

Part I : Artificial Intelligence

Suggested List of Laboratory Experiments/Assignments

Sr. No.	Group A All assignments are compulsory
1.	Implement depth first search algorithm and Breadth First Search algorithm, Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.
2.	Implement A star Algorithm for any game search problem.
3.	Implement Greedy search algorithm for any of the following application: I. Selection Sort II. Minimum Spanning Tree III. Single-Source Shortest Path Problem IV. Job Scheduling Problem V. Prim's Minimal Spanning Tree Algorithm VI. Kruskal's Minimal Spanning Tree Algorithm VII. Dijkstra's Minimal Spanning Tree Algorithm
	Group B
4.	Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem.
5.	Develop an elementary catboat for any suitable customer interaction application.

	Group C
6.	Implement any one of the following Expert System I. Information management II. Hospitals and medical facilities III. Help desks management IV. Employee performance evaluation V. Stock market trading VI. Airline scheduling and cargo schedules
Part II : Elective II	
Suggested List of Laboratory Experiments/Assignments	
Sr. No.	**Assignment Name**
	Information Security (Any five)
1.	Write a Java/C/C++/Python program that contains a string (char pointer) with a value \Hello World'. The program should AND or and XOR each character in this string with 127 and display the result.
2.	Write a Java/C/C++/Python program to perform encryption and decryption using the method of Transposition technique.
3.	Write a Java/C/C++/Python program to implement DES algorithm.
4.	Write a Java/C/C++/Python program to implement AES Algorithm.
5.	Write a Java/C/C++/Python program to implement RSA algorithm.
6.	Implement the different Hellman Key Exchange mechanism using HTML and JavaScript. Consider the end user as one of the parties (Alice) and the JavaScript application as other party (bob).
7.	Calculate the message digest of a text using the MD5 algorithm in JAVA.
	Cloud Computing (All assignments are compulsory)
1.	Case study on Microsoft azure to learn about Microsoft Azure is a cloud computing platform and infrastructure, created by Microsoft, for building, deploying and managing applications and services through a global network of Microsoft-managed data centers. OR Case study on Amazon EC2 and learn about Amazon EC2 web services.
2.	Installation and configure Google App Engine. OR Installation and Configuration of virtualization using KVM.
3.	Creating an Application in SalesForce.com using Apex programming Language.
4.	Design and develop custom Application (Mini Project) using Sales force Cloud.
5.	**Mini-Project** Setup your own cloud for Software as a Service (SaaS) over the existing LAN in your laboratory. In this assignment you have to write your own code for cloud controller using open-source technologies to implement **with HDFS**. Implement the basic operations may be like to divide the file in segments/blocks and upload/ download file on/from cloud in encrypted form.
	Augmented and Virtual Reality (Assignments 1,2, 3,7 are mandatory, any 2 from 4, 5 & 6)
1.	Installation of Unity and Visual Studio, setting up Unity for VR development, understanding documentation of the same.
2.	Demonstration of the working of HTC Vive, Google Daydream or Samsung gear VR.
3.	Develop a scene in Unity that includes:

	i. A cube, plane and sphere, apply transformations on the 3 game objects. ii. Add a video and audio source.
4.	Develop a scene in Unity that includes a cube, plane and sphere. Create a new material and texture separately for three Game objects. Change the color, material and texture of each Game object separately in the scene. Write a C# program in visual studio to change the color and material/texture of the game objects dynamically on button click.
5.	Develop and deploy a simple marker based AR app in which you have to write a C# program to play video on tracking a particular marker.
6.	Develop and deploy an AR app, implement the following using Vuforia Engine developer portal: i. Plane detection ii. Marker based Tracking(Create a database of objects to be tracked in Vuforia) iii. Object Tracking
7.	**Mini-Projects/ Case Study** Create a multiplayer VR game (battlefield game). The game should keep track of score, no. of chances/lives, levels(created using different scenes), involve interaction, animation and immersive environment. **OR** Create a treasure hunt AR application which should have the following features: i. A help button for instruction box to appear. ii. A series of markers which would give hints on being scanned. iii. Involve interaction, sound, and good UI.
	Software Modeling and Architectures **(Problem statement 1, 2 , 5 are mandatory and any one from 3 and 4)**
1.	Consider a library, where a member can perform two operations: issue book and return it. A book is issued to a member only after verifying his credentials. Develop a use case diagram for the given library system by identifying the actors and use cases and associate the use cases with the actors by drawing a use case diagram. Use UML tool.
2.	Consider online shopping system. Perform the following tasks and draw the class diagram using UML tool. Represent the individual classes, and objects Add methods Represent relationships and other classifiers like interfaces
3.	Consider the online shopping system in the assignment 2. Draw the sequence diagram using UML tool to show message exchanges
4.	Consider your neighboring travel agent from whom you can purchase flight tickets. To book a ticket you need to provide details about your journey i.e., on which date and at what time you would like to travel. You also need to provide your address. The agency has recently been modernized. So, you can pay either by cash or by card. You can also cancel a booked ticket later if you decide to change your plan. In that case you need to book a new ticket again. Your agent also allows you to book a hotel along with flight ticket. While cancelling a flight ticket you can also cancel hotel booking. Appropriate refund as per policy is made in case of cancellation. Perform the following tasks and draw the use case diagram using UML tool. a. Identify the use cases from a given non-trivial problem statement. b. Identify the primary and secondary actors for a system. c. Use to generalization of use cases and «include» stereotypes to prevent redundancy in the coding phase

Mini-Projects	
5.	Select a moderately complex system and narrate concise requirement Specification for the same. Design the system indicating system elements organizations using applicable architectural styles and design patterns with the help of a detailed Class diagram depicting logical architecture. Specify and document the architecture and design pattern with the help of templates. Implement the system features and judge the benefits of the design patterns accommodated.

Learning Resources

Text Books:

Artificial Intelligence

1. Stuart Russell and Peter Norvig, "Artificial Intelligence: A Modern Approach", Third edition, Pearson, 2003, ISBN :10: 0136042597
2. Deepak Khemani, "A First Course in Artificial Intelligence", McGraw Hill Education(India), 2013, ISBN : 978-1-25-902998-1
3. Elaine Rich, Kevin Knight and Nair, "Artificial Intelligence", TMH, ISBN-978-0-07-008770-5

Information Security

1. Atul Kahate, "Cryptography and Network Security", 3e, McGraw Hill Education
2. Prakash C. Gupta, "Cryptography and Network Security", PHI
3. V.K. Pachghare, "Cryptography and Information Security", PHI Learning

Cloud Computing

1. A. Srinivasan, J. Suresh," Cloud Computing: A Practical Approach for Learning and Implementation", Pearson, ISBN: 978-81-317-7651-3
2. Rajkumar Buyya, Christian Vecchiola, S. Thamarai Selvi, "Mastering Cloud Computing", McGraw Hill Education, ISBN-13:978-1-25-902995-0

Augmented and Virtual Reality

1. William R Sherman and Alan B Craig, "Understanding Virtual Reality: Interface, Application and Design", (The Morgan Kaufmann Series in Computer Graphics)". Morgan Kaufmann Publishers, San Francisco, CA, 2002
2. Alan B Craig, "Understanding Augmented Reality, Concepts and Applications", Morgan Kaufmann Publishers, ISBN:978-0240824086

Software Modeling and Architectures

1. Jim Arlow, Ila Neustadt, "UML 2 and the unified process –practical object-oriented analysis and design", Addison Wesley, Second edition, ISBN 978-0201770605
2. Len Bass, Paul Clements, Rick Kazman, "Software Architecture in Practice", Second Edition, Pearson ,ISBN 978-81-775-8996-2
3. Hassan Gomaa, "Software Modeling and Design- UML, Use cases, Patterns and Software Architectures", Cambridge University Press, 2011, ISBN 978-0-521-76414-8
4. Erich Gamma, "Design Patterns", Pearson, ISBN 0-201-63361-2

Reference Books:

1. Nilsson Nils J , "Artificial Intelligence: A new Synthesis", Morgan Kaufmann Publishers Inc. San Francisco, CA, ISBN: 978-1-55-860467-4
2. Patrick Henry Winston, "Artificial Intelligence", Addison-Wesley Publishing Company, ISBN: 0-201-53377-4
3. Andries P. Engelbrecht, "Computational Intelligence: An Introduction", 2nd Edition-Wiley India-

ISBN: 978-0-470-51250-0

Information Security

1. William Stallings, Lawrie Brown, "Computer Security Principles and Practice", 3rd_Edition, Pearson
2. William Stallings, "Cryptography and Network Security Principals and Practice", Fifth edition, Pearson
3. Nina Godbole, Sunit Belapure, "Cyber Security", Wiley, ISBN: 978-81-265-2179-1

Augmented and Virtual Reality

1. Steven M. LaValle, "Virtual Reality", Cambridge University Press, 2016
2. Alan B Craig, William R Sherman and Jeffrey D Will, "Developing Virtual Reality Applications: Foundations of Effective Design", Morgan Kaufmann, 2009.
3. Schmalstieg / Hollerer, "Augmented Reality: Principles & Practice", Pearson Education India; First edition (12 October 2016),ISBN-10: 9332578494
4. Sanni Siltanen, "Theory and applications of marker-based augmented reality", Julkaisija – Utgivare Publisher. 2012. ISBN 978-951-38-7449-0

Cloud Computing

1. James Bond ,"The Enterprise Cloud", O'Reilly Media, Inc. ISBN: 9781491907627
2. Dr. Kris Jamsa, "Cloud Computing: SaaS, PaaS, IaaS, Virtualization and more", Wiley Publications, ISBN: 978-0-470-97389-9
3. Anthony T. Velte Toby J. Velte, Robert Elsenpeter, "Cloud Computing: A Practical Approach", 2010, The McGraw-Hill.

Software Modeling and Architectures

1. Gardy Booch, James Rambaugh, Ivar Jacobson, "The unified modeling language user guide" , Pearson Education, Second edition, 2008, ISBN 0-321-24562-8.
2. Lan Sommerville, "Software Engineering", 9th edition, ISBN-13: 978-0-13-703515-1 ISBN-10: 0-13-703515-2.

@The CO-PO Mapping Matrix

CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	2	-	2	-	3	-	-	2	2	2	1	2
CO2	1	-	2	2	3	2	-	2	2	2	1	2
CO3	1	-	2	2	3	2	-	2	2	2	2	2
CO4	1	-	2	-	3	-	-	2	2	2	2	2
CO5	1	-	2	-	3	-	-	2	2	2	2	2
CO6	1	-	2	-	3	-	-	2	2	2	2	2

INDEX

Assignment No.	**1**
Title	Implement depth first search algorithm and Breadth First Search algorithm, Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.
Roll No.	
Class	T.E.
Date	
Subject	**Artificial Intelligence**
Signature	

Assignment No. 1

Title : Implement depth first search algorithm and Breadth First Search algorithm, Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.

Objectives: From this experiment, the student will be able to

• Understand and implement BFS and DFS algorithm.

• Analyze time complexity of the search algorithm.

Theory:

Depth First Search or DFS for a Graph

Depth First Traversal (or Search) for a graph is similar to Depth First Traversal of a tree. The only catch here is, that, unlike trees, graphs may contain cycles (a node may be visited twice). To avoid processing a node more than once, use a boolean visited array. A graph can have more than one DFS traversal.

Example:

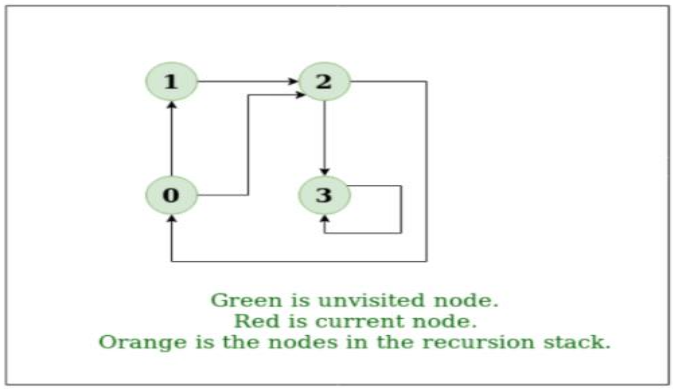
Input: n=4, e=6

0 -> 1, 0 -> 2, 1 -> 2, 2 -> 0, 2 -> 3, 3 -> 3

Output: DFS from vertex 1: 1 2 0 3

Explanation:

DFS Diagram:



Input: n = 4, e = 6

2 -> 0, 0 -> 2, 1 -> 2, 0 -> 1, 3 -> 3, 1 -> 3

Output: DFS from vertex 2 : 2 0 1 3

Explanation:

DFS Diagram:

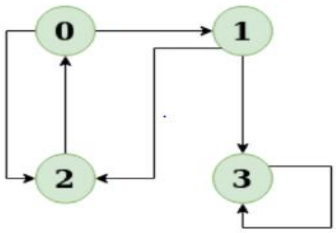
Input: n = 4, e = 6

2 -> 0, 0 -> 2, 1 -> 2, 0 -> 1, 3 -> 3, 1 -> 3

Output**:** DFS from vertex 2 : 2 0 1 3

Explanation:

DFS Diagram:



Green is unvisited node.
Red is current node.
Orange is the nodes in the recursion stack.

DFS of Graph

Depth-first search is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.

So the basic idea is to start from the root or any arbitrary node and mark the node and move to the adjacent unmarked node and continue this loop until there is no unmarked adjacent node. Then backtrack and check for other unmarked nodes and traverse them. Finally, print the nodes in the path.

Follow the below steps to solve the problem:

•Create a recursive function that takes the index of the node and a visited array.

•Mark the current node as visited and print the node.

•Traverse all the adjacent and unmarked nodes and call the recursive function with the index of the adjacent node.

Handling A Disconnected Graph:

This will happen by handling a corner case. All the vertices may not be reachable from a given vertex, as in a disconnected graph. To do a complete DFS traversal of such graphs, run DFS from all unvisited nodes after a DFS. The recursive function remains the same.

Follow the below steps to solve the problem:

• Create a recursive function that takes the index of the node and a visited array.

• Mark the current node as visited and print the node.

• Traverse all the adjacent and unmarked nodes and call the recursive function with the index of the adjacent node.

• Run a loop from 0 to the number of vertices and check if the node is unvisited in the previous DFS, then call the recursive function with the current node.

Algorithm for DFS:

1. Create single member queue comprising of root node.

2. If 1st Member of Queue is GOAL then goto Step 5.

3. If first member of queue is not GOAL then remove it and add to CLOSE or Visited

Queue. Consider its Children/ successor, if any add them from FRONT END.

4. If queue is not empty then goto Step 2, If queue is empty then goto Step 6

5. Print “SUCCESS” and stop.

6. Print “FALIURE” and stop.

.

Breadth First Search or BFS for a Graph

Breadth-First Traversal (or Search) for a graph is similar to Breadth-First Traversal of a tree .The only catch here is, that, unlike trees, graphs may contain cycles, so we may come to the same node again. To avoid processing a node more than once, we divide the vertices into two categories:

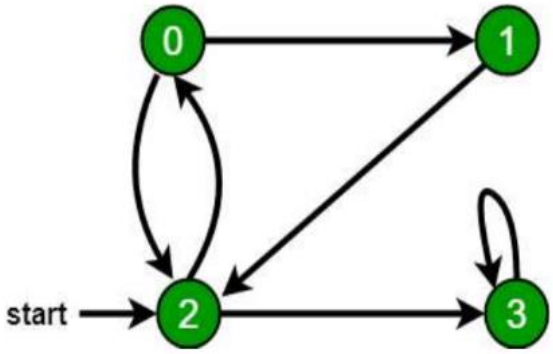
• Visited and

• Not visited.

A Boolean visited array is used to mark the visited vertices. For simplicity, it is assumed that all vertices are reachable from the starting vertex. BFS uses a queue data structure for traversal.

Example:

In the following graph, we start traversal from vertex 2.



When we come to vertex 0, we look for all adjacent vertices of it.

• 2 is also an adjacent vertex of 0.

• If we don't mark visited vertices, then 2 will be processed again and it will become a non-terminating process.

There can be multiple BFS traversals for a graph. Different BFS traversals for the above graph:

2, 3, 0, 1

2, 0, 3, 1

Implementation of BFS traversal:

Follow the below method to implement BFS traversal.

• Declare a queue and insert the starting vertex.

• Initialize a visited array and mark the starting vertex as visited.

• Follow the below process till the queue becomes empty:

• Remove the first vertex of the queue.

• Mark that vertex as visited.

• Insert all the unvisited neighbors of the vertex into the queue.

Algorithm for BFS:

1. Create single member queue comprising of root node.

2. If 1st Member of Queue is GOAL then goto Step 5.

3. If first member of queue is not GOAL then remove it and add to CLOSE or Visited Queue. Consider its Children/ successor, if any add them from BACK/REAR [FIFO]

4. If queue is not empty then goto Step 2, If queue is empty then goto Step 6

5. Print "SUCCESS" and stop.

6. Print "FALIURE" and stop.

Conclusion:

BFS is an uniformed search technique. It selects the shallowest unexpanded node in the search tree for expansion. It is complete, optimal for unit step costs and has time and space complexity of $O(b^d)$.

DFS is an uninformed search technique. It searches deeper into the problem space. It has time complexity as $O(b^m)$ and space complexity of O(b*m). DFS is implemented in C or java.

Viva Questions:

• Explain time and space complexity of BFS.

• DFS can be viewed as a special case of depth limited search with l=∞. Give reason to support the above statement.

• Explain any two uninformed search strategy used in problem solving by searching.

Assignment No.	**2**
Title	Implement A star (A*) Algorithm for any game search problem.
Roll No.	
Class	T.E.
Date	
Subject	**Artificial Intelligence**
Signature	

Assignment No. 2

Title: Implement A star (A*) Algorithm for any game search problem.

Objectives: From this experiment, the student will be able to

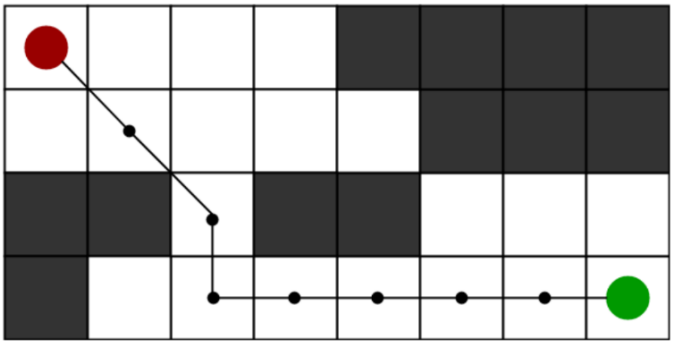
• Understand working of game search problem.

• Implement A* search algorithm.

Theory:

A* Search Algorithm

To approximate the shortest path in real-life situations, like- in maps, games where there can be many hindrances. We can consider a 2D Grid having several obstacles and we start from a source cell (colored red below) to reach towards a goal cell (colored green below)



What is A* Search Algorithm?

A* Search algorithm is one of the best and popular technique used in path-finding and graph traversals.

Why A* Search Algorithm?

Informally speaking, A* Search algorithms, unlike other traversal techniques, it has <brains=. What it means is that it is really a smart algorithm which separates it from the other conventional algorithms. This fact is cleared in detail in below sections. And it is also worth mentioning that many games and web-based maps use this algorithm to find the shortest path very efficiently (approximation).

Explanation :

Consider a square grid having many obstacles and we are given a starting cell and a target cell. We want to reach the target cell (if possible) from the starting cell as quickly as possible. Here A* Search Algorithm comes to the rescue. What A* Search Algorithm does is that at each step it picks the node

according to a value-8f' which is a parameter equal to the sum of two other parameters – 8g' and 8h'. At each step it picks the node/cell having the lowest 8f', and process that node/cell. We define8g' and 8h' as simply as possible below

g = the movement cost to move from the starting point to a given square on the grid, following the path generated to get there.

h = the estimated movement cost to move from that given square on the grid to the final destination. This is often referred to as the heuristic, which is nothing but a kind of smart guess.

We really don't know the actual distance until we find the path, because all sorts of things can be in the way (walls, water, etc.). There can be many ways to calculate this 8h9 which are discussed in the later sections.

Algorithm:

We create two lists – Open List and Closed List (just like Dijkstra Algorithm)

// A* Search Algorithm

1. Initialize the open list

2. Initialize the closed list put the starting node on the open list (you can leave its f at zero)

3. While the open list is not empty

a) find the node with the least f on the open list, call it "q"

b) pop q off the open list

c) generate q's 8 successors and set their parents to q

d) for each successor

i) if successor is the goal, stop search

ii) else, compute both g and h for successor

successor.g = q.g + distance between successor and q

successor.h = distance from goal to successor (This can be done using many ways, we will discuss three heuristics- Manhattan, Diagonal and Euclidean Heuristics)

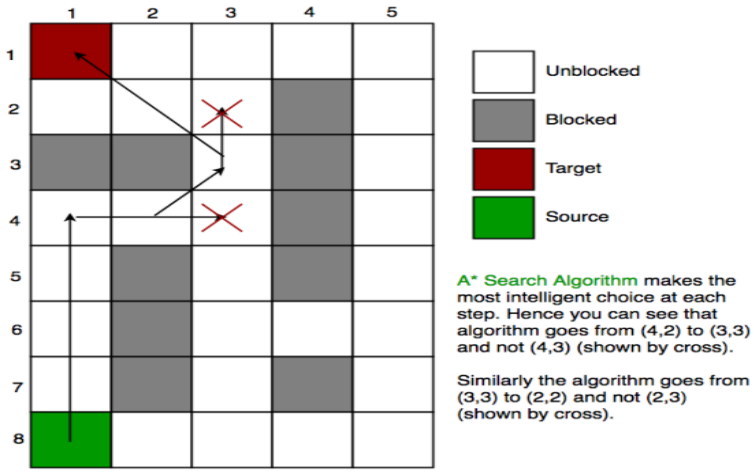
successor.f = successor.g + successor.h

iii) if a node with the same position as successor is in the OPEN list which has a lower f than successor, skip this successor

iv) if a node with the same position as successor is in the CLOSED list which has a lower f than successor, skip this successor otherwise, add the node to the open list end (for loop)

e) Push q on the closed list end (while loop)

So suppose as in the below figure if we want to reach the target cell from the source cell, then the A* Search algorithm would follow path as shown below. Note that the below figure is made by considering Euclidean Distance as a heuristics.



Heuristics

We can calculate g but how to calculate h?

We can do things.

A) Either calculates the exact value of h (which is certainly time consuming).

OR

B) Approximate the value of h using some heuristics (less time consuming).

We will discuss both of the methods.

A) Exact Heuristics –

We can find exact values of h, but that is generally very time consuming.

Below are some of the methods to calculate the exact value of h.

1) Pre-compute the distance between each pair of cells before running the A* Search Algorithm.

2) If there are no blocked cells/obstacles then we can just find the exact value of h without any pre-Computation using the distance formula/Euclidean Distance

B) Approximation Heuristics –

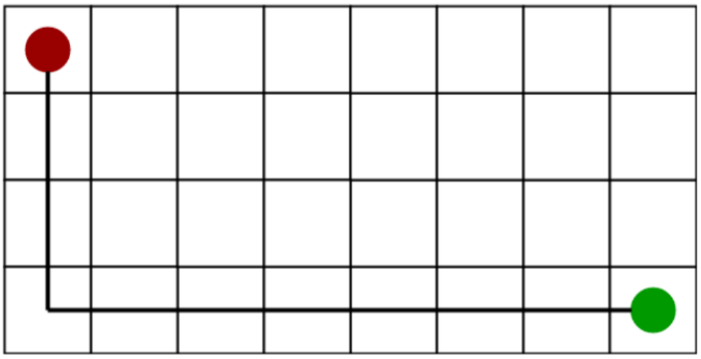
There are generally three approximation heuristics to calculate h –

1) Manhattan Distance –

• It is nothing but the sum of absolute values of differences in the goal9s x and y coordinates and the Current Cell's x and y coordinates respectively, i.e.,

h = abs (current_cell.x – goal.x) + abs (current_cell.y – goal.y)

• When to use this heuristic? – When we are allowed to move only in four directions only (right, left, top, bottom) The Manhattan Distance Heuristics is shown by the below figure (assume red spot as source cell and green spot as target cell).



2) Diagonal Distance-

• It is nothing but the maximum of absolute values of differences in the goal's x and y coordinates and the current cell's x and y coordinates respectively, i.e.,

dx = abs(current_cell.x – goal.x)

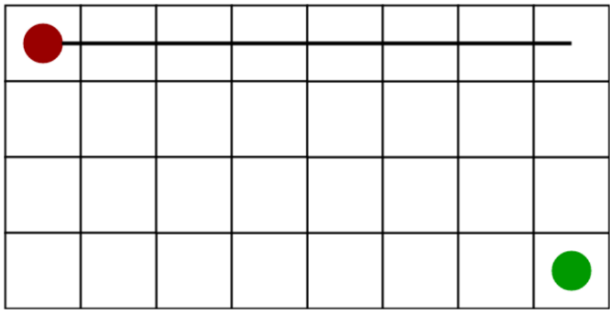
dy = abs(current_cell.y – goal.y)

h = D * (dx + dy) + (D2 - 2 * D) * min(dx, dy)

Where D is length of each node (usually = 1) and

D2 is diagonal distance between each node (usually =sqrt (2)).

• When to use this heuristic? – When we are allowed to move in eight directions only (similar to a move of a King in Chess) The Diagonal Distance Heuristics is shown by the below figure (assume red spot as source cell and green spot as target cell).



3) Euclidean Distance-

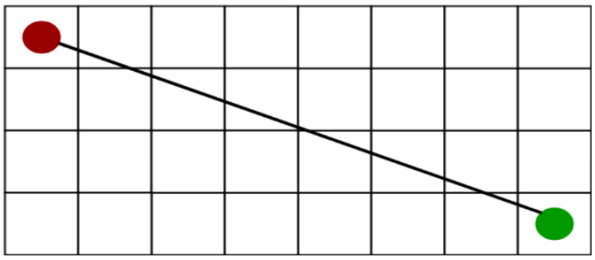
• As it is clear from its name, it is nothing but the distance between the current cell and the goal cell using

the distance formula

h = sqrt((current_cell.x – goal.x)2 + (current_cell.y – goal.y)2)

• When to use this heuristic? – When we are allowed to move in any directions.

The Euclidean Distance Heuristics is shown by the below figure (assume red spot as source cell and green spot as target cell).



Relation (Similarity and Differences) with other algorithms-

Dijkstra is a special case of A* Search Algorithm, where h = 0 for all nodes.

Limitations:

Although being the best path finding algorithm around, A* Search Algorithm doesn't produce the Shortest path always, as it relies heavily on heuristics / approximations to calculate – h.

Conclusion:

A* search algorithm used to find the shortest path through a search space to a goal state using a heuristic.

A* algorithm is executed and path with minimum cost from source node to destination node is calculated.

Viva Questions:

- State single source shortest path algorithm (Dijkstra's algorithm).
- What is A* Search Algorithm?
- Why is A* Search Algorithm Preferred?
- What is a Heuristic Function?
- How does the A * algorithm work?
- Does Google Maps use the A* algorithm?
- Is A* better than Dijkstra?

Assignment No.	**3**
Title	Implement Greedy search algorithm for any of the following application: I. Selection Sort II. Minimum Spanning Tree III. Single-Source Shortest Path Problem IV. Job Scheduling Problem V. Prim's Minimal Spanning Tree Algorithm VI. Kruskal's Minimal Spanning Tree Algorithm VII. Dijkstra's Minimal Spanning Tree Algorithm
Roll No.	
Class	T.E.
Date	
Subject	**Artificial Intelligence**
Signature	

Assignment No. 3

Aim:- Implement Greedy search algorithm for Prim's Minimal Spanning Tree Algorithm

Operating System recommended: - 64-bit Windows OS and Linux

Programming tools recommended: -

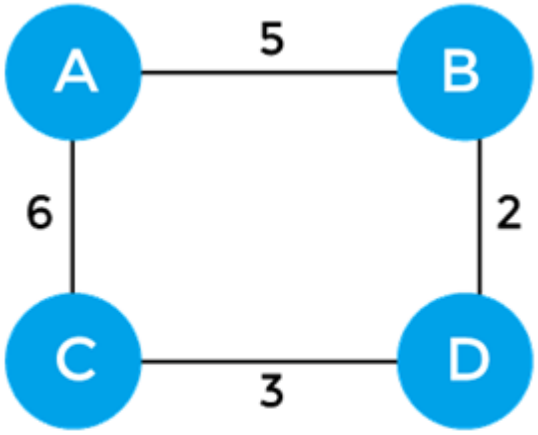
THEORY

Minimum Spanning tree

A minimum spanning tree can be defined as the spanning tree in which the sum of the weights of the edge is minimum. The weight of the spanning tree is the sum of the weights given to the edges of the spanning tree. In the real world, this weight can be considered as the distance, traffic load, congestion, or any random value.

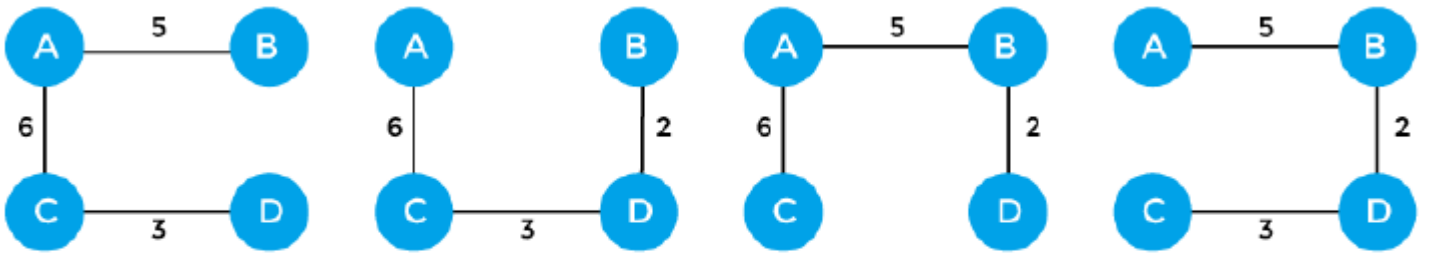
Example of minimum spanning tree

Let's understand the minimum spanning tree with the help of an example.

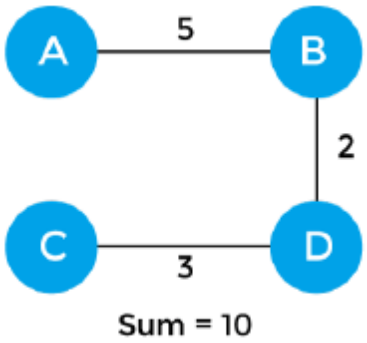


Weighted graph

The sum of the edges of the above graph is 16. Now, some of the possible spanning trees created from the above graph are –



So, the minimum spanning tree that is selected from the above spanning trees for the given weighted graph is



Applications of minimum spanning tree

The applications of the minimum spanning tree are given as follows -

o Minimum spanning tree can be used to design water-supply networks, telecommunication networks, and electrical grids.

o It can be used to find paths in the map.

Algorithms for Minimum spanning tree

A minimum spanning tree can be found from a weighted graph by using the algorithms given below -

o Prim's Algorithm

o Kruskal's Algorithm

Prim's algorithm –

It is a greedy algorithm that starts with an empty spanning tree. It is used to find the minimum spanning tree from the graph. This algorithm finds the subset of edges that includes every vertex of the graph such that the sum of the weights of the edges can be minimized.

Prim's algorithm starts with the single node and explores all the adjacent nodes with all the connecting edges at every step. The edges with the minimal weights causing no cycles in the graph got selected.

How does the prim's algorithm work?

Prim's algorithm is a greedy algorithm that starts from one vertex and continue to add the edges with the smallest weight until the goal is reached. The steps to implement the prim's algorithm are given as follows -

- First, we have to initialize an MST with the randomly chosen vertex.
- Now, we have to find all the edges that connect the tree in the above step with the new vertices.
  From the edges found, select the minimum edge and add it to the tree.
- Repeat step 2 until the minimum spanning tree is formed.

The applications of prim's algorithm are -

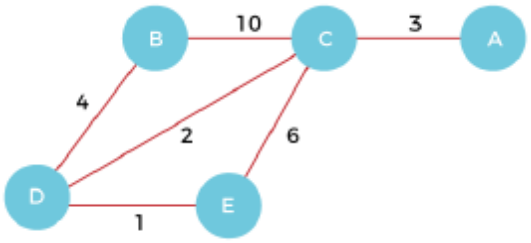
- Prim's algorithm can be used in network designing.

- It can be used to make network cycles.
- It can also be used to lay down electrical wiring cables.

Example of prim's algorithm

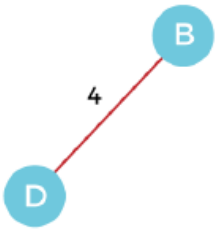
Now, let's see the working of prim's algorithm using an example. It will be easier to understand the prim's algorithm using an example.

Suppose, a weighted graph is -

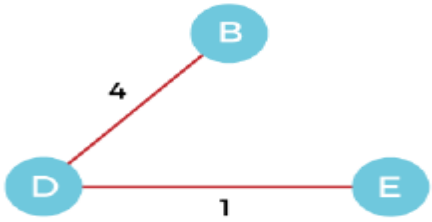


Step 1 - First, we have to choose a vertex from the above graph. Let's choose B.

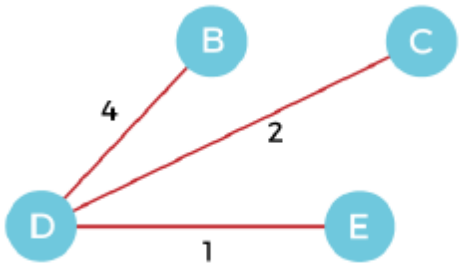
Step 2 - Now, we have to choose and add the shortest edge from vertex B. There are two edges from vertex B that are B to C with weight 10 and edge B to D with weight 4. Among the edges, the edge BD has the minimum weight. So, add it to the MST.



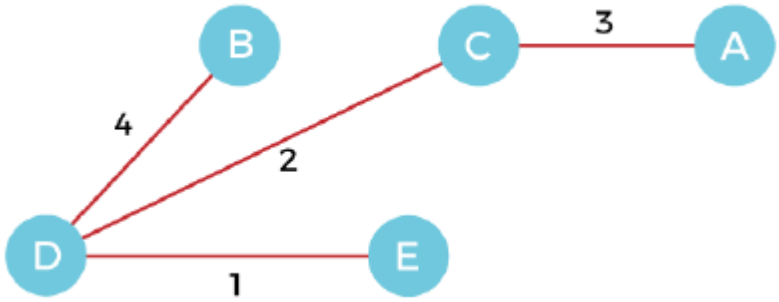
Step 3 - Now, again, choose the edge with the minimum weight among all the other edges. In this case, the edges DE and CD are such edges. Add them to MST and explore the adjacent of C, i.e., E and A. So, select the edge DE and add it to the MST.



Step 4 - Now, select the edge CD, and add it to the MST.



Step 5 - Now, choose the edge CA. Here, we cannot select the edge CE as it would create a cycle to the graph. So, choose the edge CA and add it to the MST.



So, the graph produced in step 5 is the minimum spanning tree of the given graph. The cost of the MST is given below - Cost of MST = 4 + 2 + 1 + 3 = 10 units.

Algorithm

1. Step 1: Select a starting vertex
2. Step 2: Repeat Steps 3 and 4 until there are fringe vertices
3. Step 3: Select an edge 'e' connecting the tree vertex and fringe vertex that has minimum weight
4. Step 4: Add the selected edge and the vertex to the minimum spanning tree T
5. [END OF LOOP]
6. Step 5: EXIT

Complexity of Prim's algorithm

Now, let's see the time complexity of Prim's algorithm. The running time of the prim's algorithm depends upon using the data structure for the graph and the ordering of edges. Below table shows some choice

Data structure used for the minimum edge weight	Time Complexity
Adjacency matrix, linear searching	$O(\|V\|^2)$
Adjacency list and binary heap	$O(\|E\| \log \|V\|)$
Adjacency list and Fibonacci heap	$O(\|E\|+ \|V\| \log \|V\|)$

Conclusion

Prim's algorithm can be simply implemented by using the adjacency matrix or adjacency list graph representation, and to add the edge with the minimum weight requires the linearly searching of an array of weights. It requires O(|V|2) running time. It can be improved further by using the implementation of heap to find the minimum weight edges in the inner loop of the algorithm.

Assignment No.	**4**
Title	Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem.
Roll No.	
Class	T.E.
Date	
Subject	**Artificial Intelligence**
Signature	

Assignment No. 4

Title:- Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem.

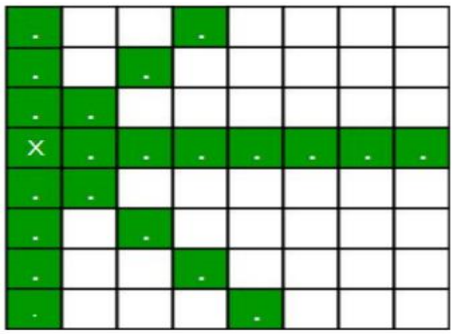
Objectives:- To study SQL DML statements

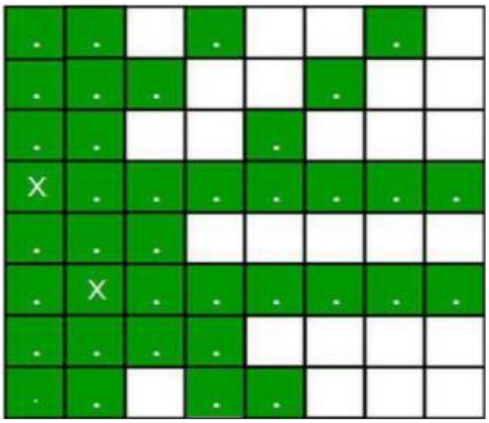
THEORY: Queen Problem using Branch and Bound

The N queens puzzle is the problem of placing N chess queens on an N×N chessboard so that no two queens threaten each other. Thus, a solution requires that no two queens share the same row, column, or diagonal.

The backtracking Algorithm for N-Queen is already discussed here. In a backtracking solution, we backtrack when we hit a dead end. In Branch and Bound solution, after building a partial solution, we figure out that there is no point going any deeper as we are going to hit a dead end.

Let's begin by describing the backtracking solution. The idea is to place queens one by one in different columns, starting from the leftmost column. When we place a queen in a column, we check for clashes with already placed queens. In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution. If we do not find such a row due to clashes, then we backtrack and return false.





Placing 1st queen on (3, 0) and 2nd queen on (5, 1)

For the 1st Queen, there are total 8 possibilities as we can place 1st Queen in any row of first column. Let's place Queen 1 on row 3. After placing 1st Queen, there are 7 possibilities left for the 2nd Queen . But wait, we don't really have 7 possibilities. We cannot place Queen 2 on rows 2, 3 or 4 as those cells are under attack from Queen 1. So, Queen 2 has only 8 – 3 = 5 valid positions left.

After picking a position for Queen 2, Queen 3 has even fewer options as most of the cells in its column are under attack from the first 2 Queens.

We need to figure out an efficient way of keeping track of which cells are under attack. In previous solution we kept an 8-by-8 Boolean matrix and update it each time we placed a queen, but that required linear time to update as we need to check for safe cells.

Basically, we have to ensure 4 things:

1. No two queens share a column.

2. No two queens share a row.

3. No two queens share a top-right to left-bottom diagonal.

4. No two queens share a top-left to bottom-right diagonal.

Number 1 is automatic because of the way we store the solution. For number 2, 3 and 4, we can perform updates in O (1) time. The idea is to keep three Boolean arrays that tell us which rows and which diagonals are occupied.

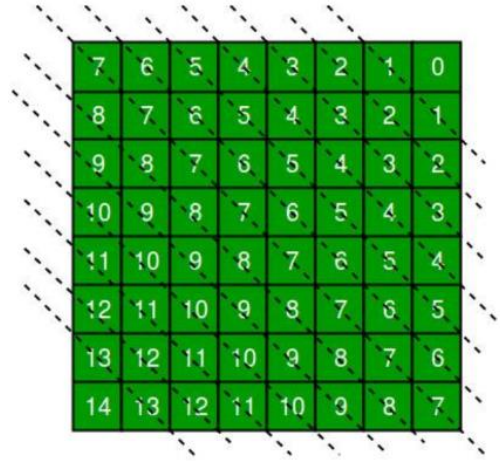
Let's do some pre-processing first. Let9s create two N x N matrix one for / diagonal and other one for \ diagonal. Let's call them slashCode and backslashCode respectively. The trick is to fill them in such a way that two queens sharing a same /diagonal will have the same value in matrix slashCode, and if they share same \- diagonal, they will have the same value in backslashCode matrix.

For an N x N matrix, fill slashCode and backslashCode matrix using below formula –

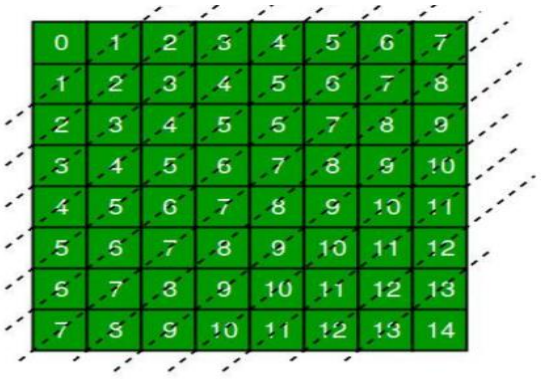
slashCode[row][col] = row + col

backslashCode[row][col] = row – col + (N-1)

Using above formula will result in below matrices



r -c + 7



r + c

The 8N – 19 in the backslash code is there to ensure that the codes are never negative because we will be using the codes as indices in an array. Now before we place queen i on row j, we first check whether row j is used (use an array to store row info). Then we check whether slash code (j + i) or backslash code (j – i + 7) are used (keep two arrays that will tell us which diagonals are occupied). If yes, then we have to try a different location for queen i. If not, then we mark the row and the two diagonals as used and recurse on queen i + 1. After the recursive call returns and before we try another position for queen i, we Need to reset the row, slash code and backslash code as unused again.

Conclusion:

Thus we have studied backtracking and branch and bound. Students tried to understand when these algorithmic approaches can be applied to problems of interest.

Assignment No.	**5**
Title	Develop an elementary chatbot for any suitable customer interaction application.
Roll No.	
Class	T.E. Computer
Date	
Subject	**Artificial Intelligence**
Signature	

Assignment No: 5

Aim:- Develop an elementary chatbot for any suitable customer interaction application.

Operating System recommended: - 64-bit Windows OS and Linux

Programming tools recommended:

Theory:-

What is a chatbot?

A chatbot is a computer program designed to have a conversation with human beings over the internet. It's also known as conversational agents, which communicate and collaborate with human users, through text messaging, in order to accomplish a specific task. Basically, there are two types of chatbots. The one that uses Artificial Intelligence, and another one is based on multiple choice scripts.

Both types of chatbots aim to create a more personalized content experience for the users, whether that's while watching a video, reading articles or buying new shoes.

These Chatbots hold the promise of being the next generation of technology that people use to interact online with business enterprises. These Chatbots offer a lot of advantages, one of which is that, because Chatbots communicate using a natural language, users don't need to learn yet another new website interface, to get comfortable with the unavoidable quirks.

Chatbots are capable to interpret human speech, and decide which information is being sought. Artificial intelligence is getting smarter each day, and brands that are integrating Chatbots with the artificial intelligence, can deliver one-to-one individualized experiences to consumers.

Why chatbot?

Chatbots can be useful in many aspects of the customer experience, including providing customer service, presenting product recommendations and engaging customers through targeted marketing campaigns. If a customer has an issue with a product, she can connect with a chatbot to explain the situation and the chatbot can input that information to provide a recommendation of how to fix the product. On the recommendation side. Brands are also using chatbots to connect their customers with thought leaders and add personality to their products. In all cases, brands seem to be having great success and experiencing increased engagement and revenue.

Chatbots are easy to use and many customers prefer them over calling a representative on the phone because it tends to be faster and less invasive. They can also save money for companies

and are easy to set up. Chatbots are relatively new and most companies haven't implemented them yet, it's only natural that users are interested in them. Hence, people want to discover what chatbots can and cannot do. The number of businesses using chatbots has grown exponentially. Chatbots have increased from 30,000 in 2016 to over 100,000 today. Every major company has announced their own chatbot and 60% of the youth population uses them daily. These statistics prove that chatbots are the new-gen tech. No more waiting for the right time to incorporate them into your business. The time is now. By the year 2020, nearly 80% of businesses will have their own chatbot. Billions of people are already using chatbots, so it's time your business did too.

Benefits of chatbot?

Chatbots are being made to ease the pain that the industries are facing today.

- The purpose of chat bots is to support and scale business teams in their relations with customers.
- Chatbots may sound like a futuristic notion, but according to Global Web Index statistics, it is said that 75% of internet users are adopting one or more messenger platforms.
- Although research shows us that each user makes use of an average of 24 apps a month, wherein 80% of the time would be in just 5 apps. This means you can hardly shoot ahead with an app, but you still have high chances to integrate your chatbot with one of these platforms.

1. Available 24*7:

I'm sure most of you have experienced listening to the boring music playing while you're kept on hold by a customer care agent. On an average people spend 7 minutes until they are assigned to an agent. Gone are the days of waiting for the next available operative. Bots are replacing live chat and other forms of contact such as emails and phone calls.

2. Handling Customers:

We humans are restricted to the number of things we can do at the same time. A study suggests that humans can only concentrate on 3–4 things at the same time. If it goes beyond that you are bound to meet errors.

Chatbots on the other hand can simultaneously have conversations with thousands of people. Companies like Taco Bell and Domino's are already using chatbots to arrange delivery of parcels.

3. Helps you save Money:

If you are a business owner you are bound have a lot of employees who need to be paid for the work they do. And these expenses just keep adding up as business grows. Chatbots are a onetime investment which helps businesses reduces down on staff required. You could integrate a customer support chatbot in your business to cater to simple queries of customers and pass on only the complex queries to customer support agents.

4. Provides 100% satisfaction to customers:

Humans react to others based on their mood and emotions. If a agent is having a good attitude or is in good mood he will most probably talk to customers in a good way. In contrary to this the customer will not be satisfied. Whereas chatbots are bound by some rules and obey them as long as they're programmed to. They always treat a customer in the most polite and perfect way no matter how rough the person is.

5. Automation of repetitive work:

Let's be honest, no one likes doing the same work again and again over brief period of time. In the case of humans, such tasks are prone to errors. Chatbots now help automate tasks which are to be done frequently and at the right time. Also, now there are numerous slack bots which automate repetitive tasks. This helps People save time and increase productivity.

6. Personal Assistant:

People could use Bots as a fashion advisor for clothing recommendations, or ask trading tips from a finance bot, suggest places to visit from a travel bot and so forth. This would help the users get a more personal touch from the chatbot.

How chatbot can drive revenue for you?

Below we have compiled reasons why chatbots are important for your business and how can they help in increasing revenues:

a. Higher user customer engagement

Most businesses these days have a web presence. But with being on the internet, boundaries of day and night, availability and unavailability have changed, so have user expectations. This is probably the biggest reason to use them. Bots give the user an interactive experience.

b. Mobile-ready and immediate availability

Along with a web presence, it has also become increasingly important for brands to have a mobile presence - mobile apps, mobile-optimized websites.

Considering how chat has been around on the mobile for ages, most chatbot implementations don't need you to work on tweaking their UI, they are ready to implement and so available to your customers immediately Instead, implementing a chatbot - which works on the mobile browser or a messaging-app which the user regularly uses - makes it all the more reason for a customer to be engaged with the brand.

c. It can drive sales

Chatbots can be intelligent. Depending on a user's preferences or purchases, it can send products to customers which are more likely to convert into sales. Or it can send coupons to users for in-store purchases/discounts. Bots can also be used to link the user to your mCommerce site/app so they can buy the product directly from the convenience of their phones Sell Intelligently.

- Product Recommendations: Push proactive recommendations to users based on their preferences and search and order history.
- Enable order booking over chat

d. Minimal cost - Maximum return

The best part about bots is they are cheap. Chatbot provide the necessary infrastructure and APIs for creating these bots. They require minimal maintenance and since it is automated, there is no labor-intensive work that goes in there.

e. Customer Service

- **Track Order:** Keep users up to date with order status. Schedule or reschedule delivery to a provided address or request to pick it up at any other Best Buy outlet.
- **Stock outs:** Notify users when desired product is available and place order over a chat.
- **Returns and Replacements :** No waiting time to reach customer care. Customers can instantly place request to replace or return an order.

Application across Industries :

According to a new survey, 80% of businesses want to integrate chatbots in their business model by 2020. So which industries can reap the greatest benefits by implementing consumer-facing chatbots? According to a chatbot, these major areas of direct-to-consumer engagement are prime:

Chatbots in Restaurant and Retail Industries

Famous restaurant chains like Burger King and Taco bell has introduced their Chatbots to stand out of competitors of the Industry as well as treat their customers quickly. Customers of these restaurants are greeted by the resident Chatbots, and are offered the menu options- like a counter order, the Buyer chooses their pickup location, pays, and gets told when they can head over to grab their food.

Chatbots in Hospitality and Travel

For hoteliers, automation has been held up as a solution for all difficulties related to productivity issues, labour costs, a way to ensure consistently, streamlined production processes across the system.

Chatbots in Health Industry

Chatbots are a much better fit for patient engagement than Standalone apps. Through these Health-Bots, users can ask health related questions and receive immediate responses. These responses are either original or based on responses to similar questions in the database. The impersonal nature of a bot could act as a benefit in certain situations, where an actual Doctor is not needed.

Chatbots in E-Commerce

Mobile messengers- connected with Chatbots and the E-commerce business can open a new channel for selling the products online. E-commerce Shopping destination “Spring” was the early adopter. E-commerce future is where brands have their own Chatbots which can interact with their customers through their apps.

Process

Stage #1: Chatty Bot welcomes you

Teach your assistant to introduce itself in the console.

Stage #2: Print your name

Introduce yourself to the bot.

Stage #3: Guess the age

Use your knowledge of strings and numbers to make the assistant guess your age.

Stage #4: Learning numbers

Your assistant is old enough to learn how to count. And you are experienced enough to apply a for loop at this stage!

Stage #5: Multiple Choice

At this point, the assistant will be able to check your knowledge and ask multiple-choice questions. Add some functions to your code and make the stage even better.

How To Run The Project?

To run this project, you must have installed Python on your PC. After downloading the project, follow the steps below:

Step1: Extract/Unzip the file

Step2: Go inside the project folder, open cmd then type bot.py and enter to start the system.

OR

Step2: Simply, double-click the bot.py file and you are ready to go.

Conclusion: Thus, we learn how to create a chatbot for any application,

Assignment No.	**6**
Title	**Implement any one of the following Expert system** I. Information management II. Hospitals and medical facilities III. Help desks management IV. Employee performance evaluation V. Stock market trading VI. Airline scheduling and cargo schedules
Roll No.	
Class	T.E. Computer
Date	
Subject	**Artificial Intelligence**
Signature	

Assignment No: 06

Aim:- Implement any one of the following Expert system

I. Information management

II. Hospitals and medical facilities

III. Help desks management

IV. Employee performance evaluation

V. Stock market trading

VI. Airline scheduling and cargo schedules

Operating System recommended :- 64-bit Windows OS and Linux

Programming tools recommended:

Theory:-

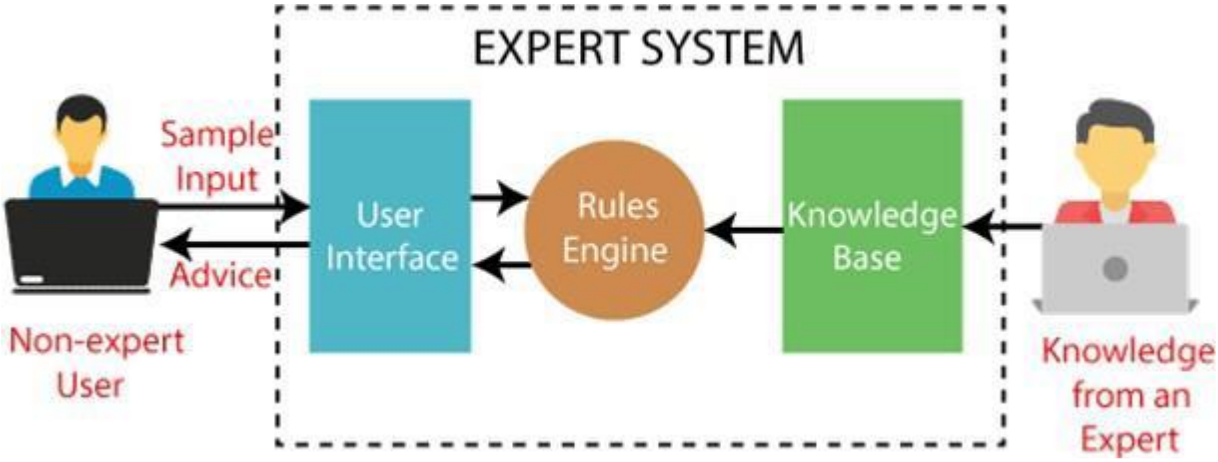
What is an Expert System?

An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert. It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries.

The expert system is a part of AI, and the first ES was developed in the year 1970, which was the first successful approach of artificial intelligence. It solves the most complex issue as an expert by extracting the knowledge stored in its knowledge base. The system helps in decision making for compsex problems using both facts and heuristics like a human expert. It is called so because it contains the expert knowledge of a specific domain and can solve any complex problem of that particular domain. These systems are designed for a specific domain, such as medicine, science**,** etc.

The performance of an expert system is based on the expert's knowledge stored in its knowledge base. The more knowledge stored in the KB, the more that system improves its performance. One of the common examples of an ES is a suggestion of spelling errors while typing in the Google search box.

Below is the block diagram that represents the working of an expert system:



Below are some popular examples of the Expert System:

- **DENDRAL:** It was an artificial intelligence project that was made as a chemical analysis expert system. It was used in organic chemistry to detect unknown organic molecules with the help of their mass spectra and knowledge base of chemistry.

- **MYCIN:** It was one of the earliest backward chaining expert systems that was designed to find the bacteria causing infections like bacteraemia and meningitis. It was also used for the recommendation of antibiotics and the diagnosis of blood clotting diseases.

- **PXDES:** It is an expert system that is used to determine the type and level of lung cancer. To determine the disease, it takes a picture from the upper body, which looks like the shadow. This shadow identifies the type and degree of harm.

- **CaDeT:** The CaDet expert system is a diagnostic support system that can detect cancer at early stages.

Characteristics of Expert System

- **High Performance:** The expert system provides high performance for solving any type of complex problem of a specific domain with high efficiency and accuracy.

- **Understandable:** It responds in a way that can be easily understandable by the user. It can take input in human language and provides the output in the same way.

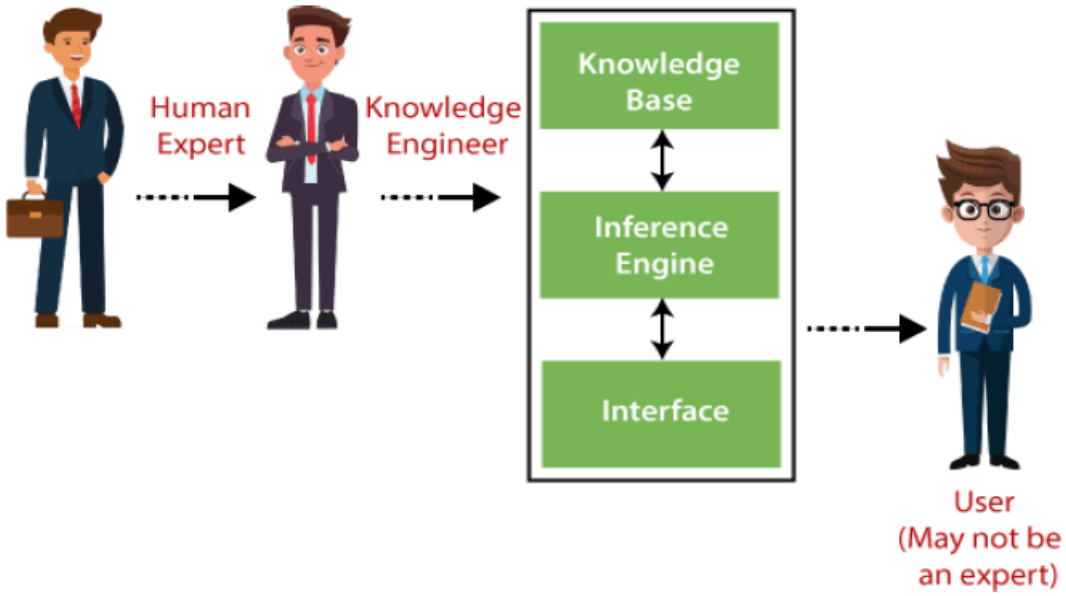
- **Reliable:** It is much reliable for generating an efficient and accurate output.

- **Highly responsive:** ES provides the result for any complex query within a very short period of time.

Components of Expert System

An expert system mainly consists of three components:

- User Interface
- Inference Engine
- Knowledge Base



Participants in the development of Expert System

There are three primary participants in the building of Expert System:

1. **Expert:** The success of an ES much depends on the knowledge provided by human experts. These experts are those persons who are specialized in that specific domain.

2. **Knowledge Engineer:** Knowledge engineer is the person who gathers the knowledge from the domain experts and then codifies that knowledge to the system according to the formalism.

3. **End-User:** This is a particular person or a group of people who may not be experts, and working on the expert system needs the solution or advice for his queries, which are complex.

Advantages of Expert System

o These systems are highly reproducible.

o They can be used for risky places where the human presence is not safe.

o The performance of these systems remains steady as it is not affected by emotions, tension, or fatigue.

Limitations of Expert System

- The response of the expert system may get wrong if the knowledge base contains the wrong information.
- Like a human being, it cannot produce a creative output for different scenarios.
- Its maintenance and development costs are very high.
- Knowledge acquisition for designing is much difficult.
- For each domain, we require a specific ES, which is one of the big limitations.
- It cannot learn from itself and hence requires manual updates.

Applications of Expert System

- **In designing and manufacturing domain**

It can be broadly used for designing and manufacturing physical devices such as camera lenses and automobiles.

- **In the knowledge domain**

These systems are primarily used for publishing the relevant knowledge to the users. The two popular ES used for this domain is an advisor and a tax advisor.

- **In the finance domain**

In the finance industries, it is used to detect any type of possible fraud, suspicious activity, and advise bankers that if they should provide loans for business or not.

- **In the diagnosis and troubleshooting of devices**

In medical diagnosis, the ES system is used, and it was the first area where these systems were used.

- **Planning and Scheduling**

The expert systems can also be used for planning and scheduling some particular tasks for achieving the goal of that task.

Conclusion:

In this way design the expert system for real world application.

INDEX

EX.NO:1

DISPLAY THE RESULT

AIM:

Write a Java/C/C++/Python program that contains a string (char pointer) with a value \Hello World9. The program should AND OR and XOR each character in this string with 127 and display the result

DESCRIPTION:

(a) String

The string is the one-dimentional array of characters terminated by null('\0'). Each and every character in the array consumes one byte of memory, and the last character must always be 0. The termination character('\0') is used to identify where the string ends. In C language string declaration can be done in two ways.

1. By char array
2. By string literal

EXAMPLE:

Let's see the example of declaring string by char array in C language.

Char ch[17]={'o','n','l','i','n','e','s','m','a','r','t','t','r','a','i','n','e','r','\0'};

As we know, array index starts from 0, so it will be represented as in the figure given below.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
o	n	l	i	n	e	s	m	a	r	t	t	r	a	i	n	e	r	\0

While declaring string, size is not mandatory. So we can write the above code as given below:

Char ch[]={,'n','l','i','n','e','s','m','a','r','t','t','r','a','i','n','e','r','\0'}

We can also define the string by the string literal in C language. For example: char str[]="onlinemarttrainer";

In such case, '\0' will be appended at the end of the string by the compiler.

(b) AND operation

There are two inputs and one output in binary AND operation. The inputs and result to a binary AND operation can only be 0 or 1. The binary AND operation will always produce a 1 output if both inputs are 1 and will produce a 0 output if both inputs are 0. For two different inputs, the output will be 0.

AND Truth table

Input		Output
X	Y	
0	0	0
0	1	0
1	0	0
1	1	1

C) XOR operation

There are two inputs and one output in binary XOR (exclusive OR) operation. It is similar to ADD operation which takes two inputs and produces one result. i.e. one output. The inputs and result to a binary AND operation can only be 0 or 1. The binary XOR operation will always produce a 1 output if either of its inputs are 1 and will produce a 0 output if both inputs are 0 or 1.

XOR Truth table

Input		Output
X	Y	
0	0	0
0	1	1
1	0	1
1	1	0

ALGORITHM:

STEP-1: Define the string

STEP-2: Perform AND operation

STEP-3: Perform XOR operation

STEP4: Display the result

CONCLUSION: Thus the AND or and XOR a string with a 127 had been implemented successfully using C language.

EX.NO:2

PERFORM ENCRYPTION AND DECRYPTION USING THE METHOD OF RAILFENCE– ROW & COLUMN TRANSPOSITION TECHNIQUE.

AIM:

Write a Java/C/C++/Python program to perform encryption and decryption using the method of Transposition technique

DESCRIPTION:

In the rail fence cipher, the plain text is written downwards and diagonally on successive "rails" of an imaginary fence, then moving up when were ach the bottom rail. When were ach the to p rail, the message is written downwards again until the whole plain text is written out. The message is then read off in rows.

EXAMPLE:

A	U	T	H	O	R
1	6	5	2	3	4
W	E	A	R	E	D
I	S	C	O	V	E
R	E	D	S	A	V
E	Y	O	U	R	S
E	L	F	A	B	C

yields the cipher

W I R E E R O S U A E V A R B D E V S C A C D O F E S E Y L .

ALGORITHM:

STEP-1: Read the Plain text.

STEP-2: Arrange the plain text in row column are matrix format.

STEP-3: Now read the keyword depending on the number of columns of the plain text.

STEP4: Arrange the characters of the key word in sorted order and the corresponding columns of the plain text.

STEP-5: Read the characters row wise or column wise in the former order to get the cipher text.

CONCLUSION:

Thus the railfence transposition algorithm had been executed successfully.

EX.NO:3

IMPLEMENTATION OF DES

AIM:

Write a Java/C/C++/Python program to implement DES algorithm.

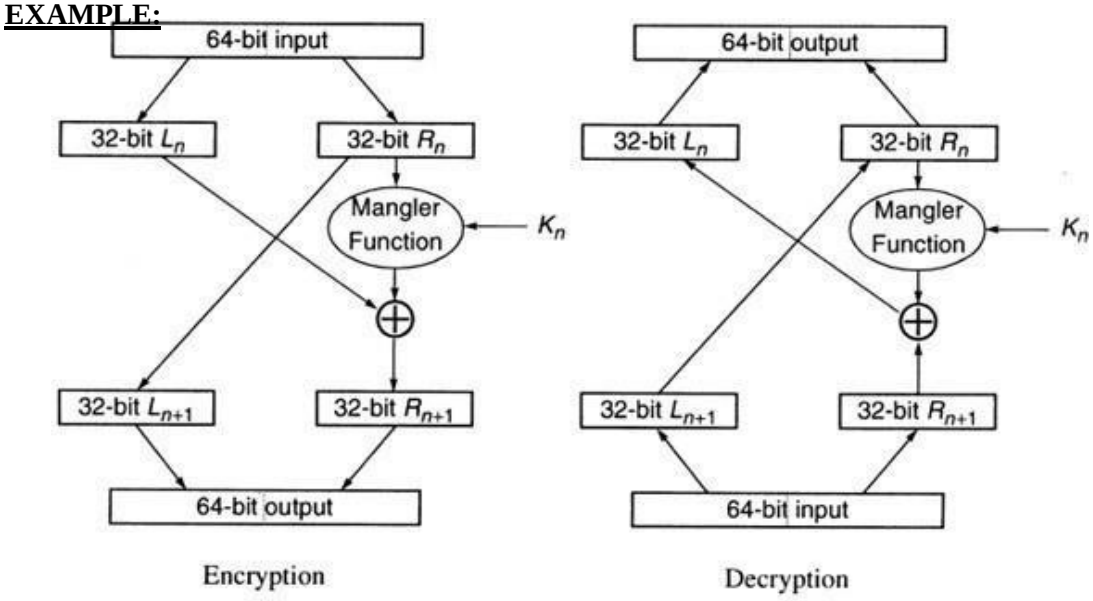
DESCRIPTION:

DES is a symmetric encryption system that uses64-bitblocks, 8bits of which are used for parity checks. The key therefore has a "useful" length of 56bits, which means that only 56 bits are actually used in the algorithm. The algorithm involves carrying out combinations, substitutions and permutations between the text to be encrypted and the key, while making sure the operations can be performed in both directions. The key is ciphered on 64 bits and made of 16 blocks of 4 bits, generally denoted k_1 to k_{16}. Given that "only" 56 bits are actually used for encrypting, there can be 2^{56} different keys.

The main parts of the algorithm are as follows:

- Fractioning of the text into 64-bit blocks
- Initial permutation of blocks
- Break down of the blocks into two parts: left and right ,named L and R
- Permutation and substitution steps repeated 16 times
- Re-joining of the left and right parts then inverse initial permutation

EXAMPLE:



Data Encryption Standard (DES)

The Data Encryption Standard (DES) is a Symmetric-key block cipher issued by the national Institute of Standards & Technology (NIST).

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). General Structure of DES is depicted in the following illustration −

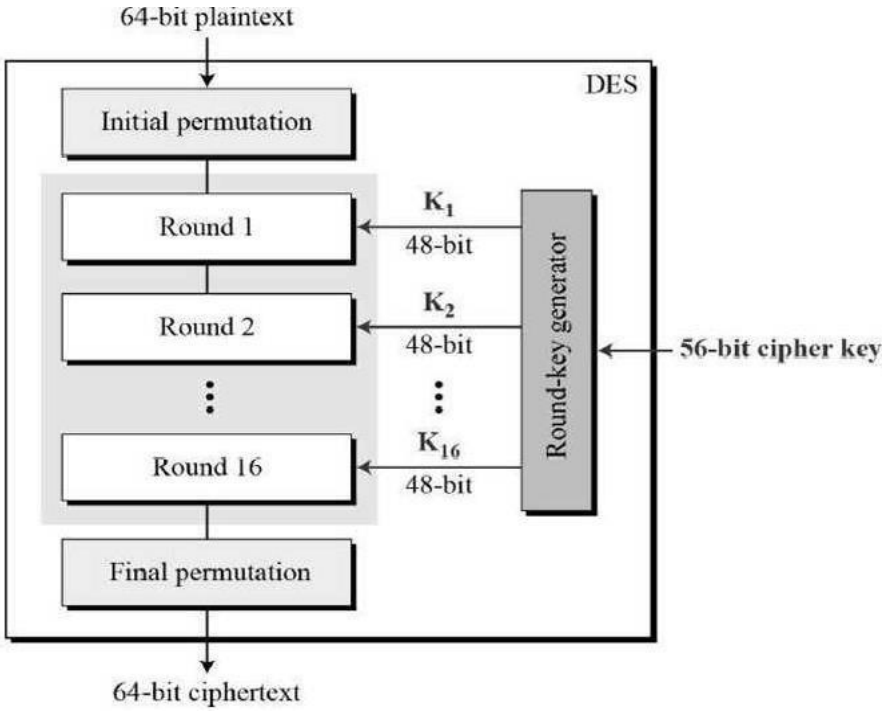


Figure 3.1: General Structure of DES

Since DES is based on the Feistel Cipher, all that is required to specify DES is −

- Round function
- Key schedule
- Any additional processing − Initial and final permutation

Initial and Final Permutation

The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES. The initial and final permutations are shown as follows

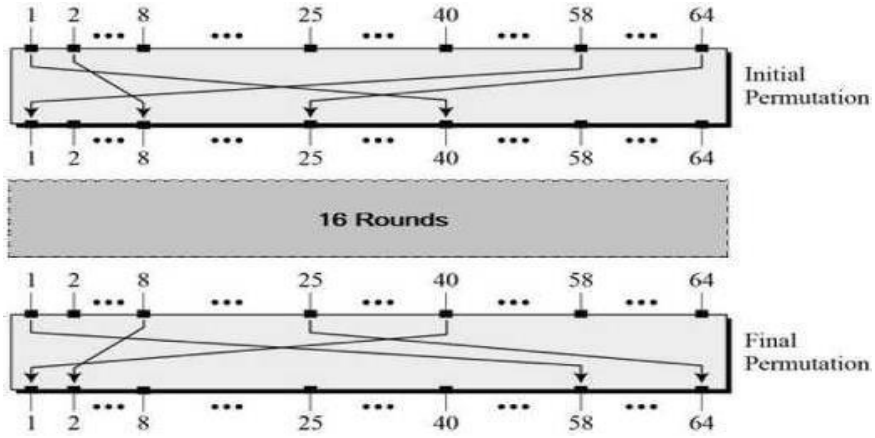


Figure 3.2 initial and final permutations

Round Function

The heart of this cipher is the DES function, f. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.

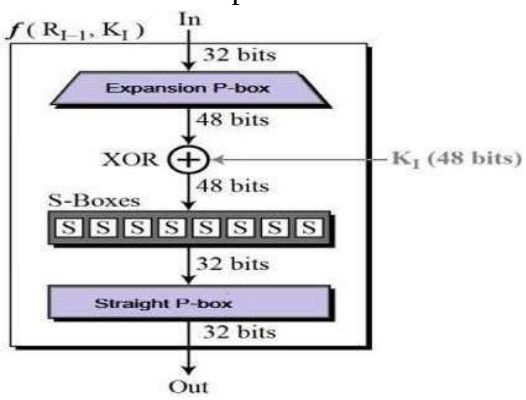
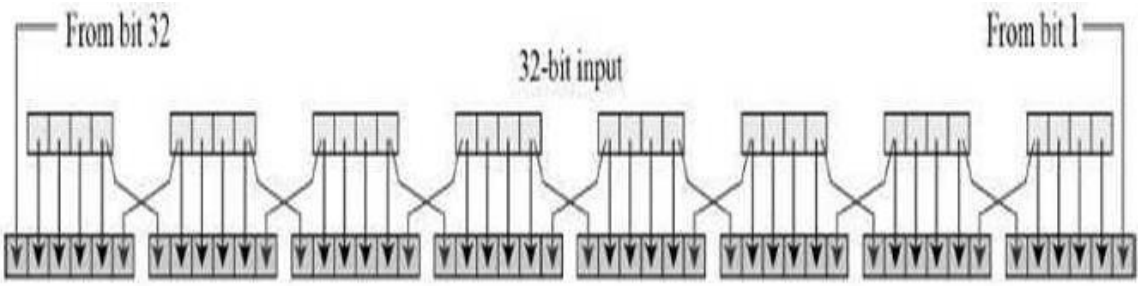


Figure 3.3 Round Functions

Expansion Permutation Box

Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits.



Permutation logic is graphically depicted in the following illustration:

Figure 3.4 Permutation logic

The graphically depicted Permutation logic is generally described as table in DES specification illustrated as shown:

Table 3.1 Permutation logic

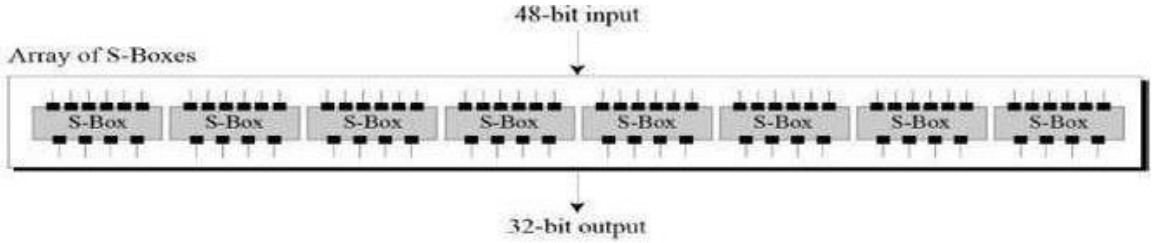
32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

XOR(Whitener)

After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.

Substitution Boxes

The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and



a 4-bit output. Refer the following illustration –

Figure 3.5 S-Boxes

The S-box rule is illustrated below –

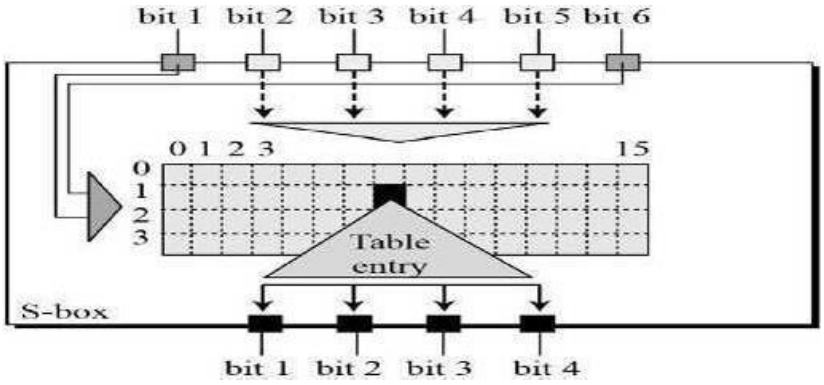


Figure 3.6 S-Box Rules

There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32 bit section.

Straight Permutation − The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

Table 3.2 Straight Permutation

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Key Generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration −

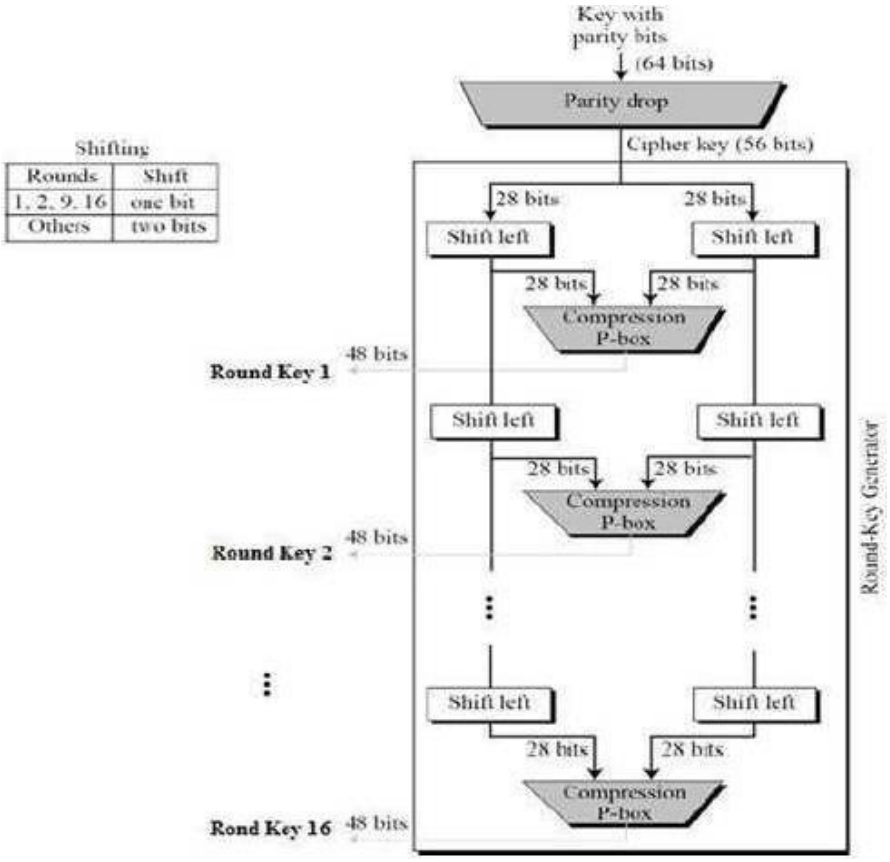


Figure 3.7 the process of key generation

The logic for Parity drops, shifting, and Compression P-box is given in the DES description

DES Analysis

The DES satisfies both the desired properties of block cipher. These two properties make cipher very strong.

•Avalanche effect − A small change in plaintext results in the very great change in the cipher text.

•Completeness − Each bit of cipher text depends on many bits of plaintext.

During the last few years, cryptanalysis has found some weaknesses in DES when key selected are weak keys. These keys shall be avoided.

DES has proved to be a very well designed block cipher. There have been no significant cryptanalytic attacks on DES other than exhaustive key search.

ALGORITHM:

STEP-1: Read the 64-bit plain text.

STEP-2: Split it into two 32-bit blocks and store it in two different arrays.

STEP-3: Perform XOR operation between these two arrays.

STEP-4: The output obtained is stored as the second 32-bit sequence and the original second 32-bit sequence forms the first part.

STEP-5: Thus the encrypted 64-bit cipher text is obtained in this way. Repeat the same process for the remaining plain text characters.

CONCLUSION:

Thus the data encryption standard algorithm had been implemented successfully using Java language.

EX.NO:4

IMPLEMENTATION OF AES

AIM:

Write a Java/C/C++/Python program to implement AES algorithm.

.

DESCRIPTION:

Theory Concepts:

The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six times faster than triple DES.
A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow.

The features of AES are as follows −

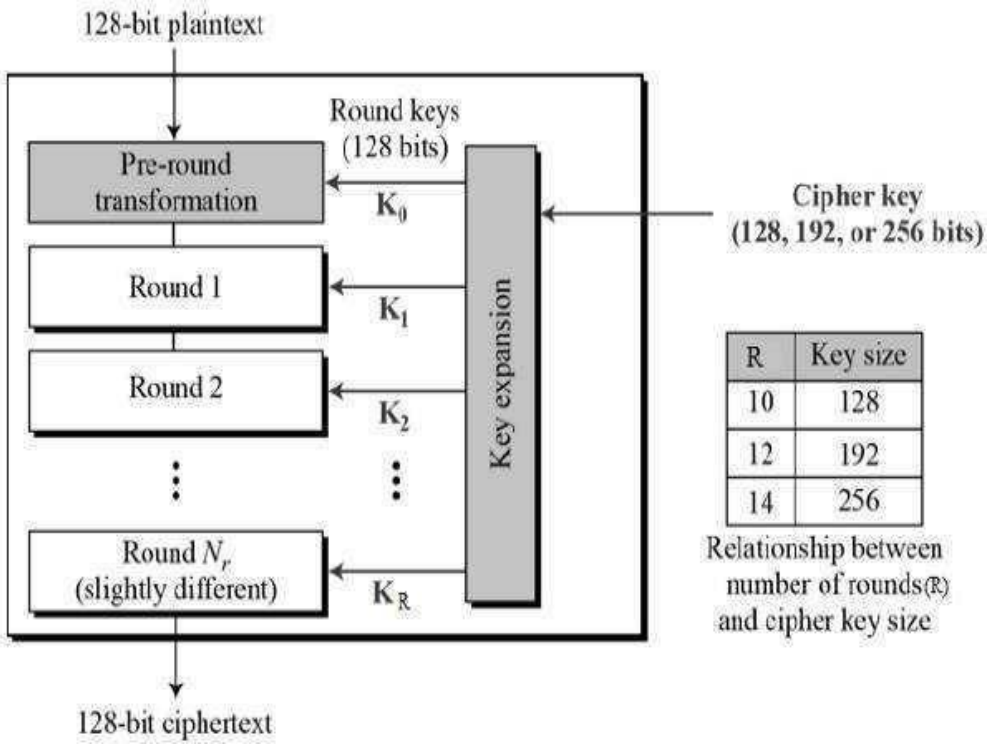
- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C ,Java and Python

Operation of AES

AES is an iterative rather than Feistel cipher. It is based on 'substitution–permutation network'. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).
Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix −

Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

The schematic of AES structure is given in the following illustration −



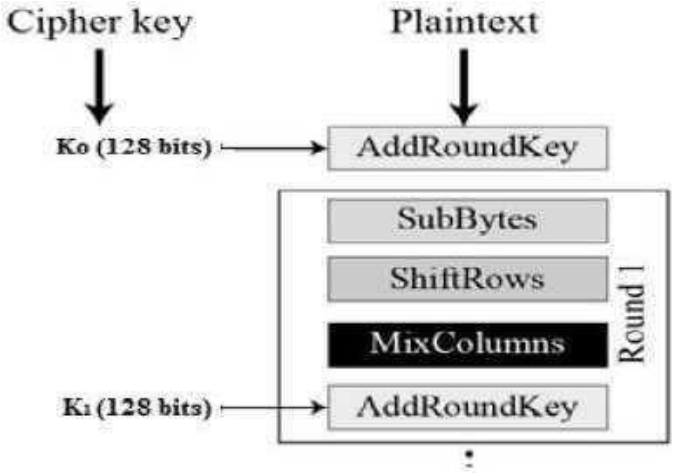
R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

Figure 4.1 AES structure

Encryption Process

Here, we restrict to description of a typical round of AES encryption. Each round comprise of four sub-processes. The first round process is depicted below −



CONCLUSION:

Thus the Advanced encryption standard algorithm had been implemented successfully using Java language

EX.NO:5

IMPLEMENTATION OF RSA

AIM:

To write a C program to implement the RSA encryption algorithm.

DESCRIPTION:

RSA is an algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm. Asymmetric means that there are two different keys. This is also called public key cryptography, because one of them can be given to everyone. A basic principle behind RSA is the observation that it is practical to find three very large positive integers e, d and n such that with modular exponentiation for all integer m:

$$(m^e)^d = m(\mathrm{mod} n)$$

The public key is represented by the integers n and e; and, the private key, by the integer d. m represents the message. RSA involves a public key and a private key. The public key can be known by everyone and is used for encrypting messages. The intention is that messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key.

RSA(Rivest, Shamir & Adleman)

RSA is an algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm. Asymmetric means that there are two different keys. This is also called public key cryptography, because one of the keys can be given to anyone. The other key must be kept private. The algorithm is based on the fact that finding the factors of a large composite number is difficult: when the integers are prime numbers, the problem is called prime factorization. It is also a key pair (public and private key) generator.

- RSA makes the public and prívate keys by multiplying two large prime numbers *p* and *q*
  - It's easy to find & multiply large prime No. ($n=pq$)
  - It is very difficult to factor the number *n* to find *p* and *q*
  - Finding the private key from the public key would require a factoring operation
  - The real challenge is the selection & generation of keys.

- RSA is complex and slow, but secure
- 100 times slower than DES on s/w & 1000 times on h/w

The Rivest-Shamir-Adleman (RSA) algorithm is one of the most popular and secures public-key encryption methods. The algorithm capitalizes on the fact that there is no efficient way to factor very large (100-200 digit) numbers.

Using an encryption key (*e*,*n*), the algorithm is as follows:

1. Represent the message as an integer between 0 and (*n*-1). Large messages can be broken up into a number of blocks. Each block would then be represented by an integer in the same range.
2. Encrypt the message by raising it to the *e*th power modulo *n*. The result is a ciphertext message C.
3. To decrypt ciphertext message C, raise it to another power *d* modulo *n*

The encryption key (*e*,*n*) is made public. The decryption key (*d*,*n*) is kept private by the user.

How to Determine Appropriate Values for *e*, *d*, and *n*

1. Choose two very large (100+ digit) prime numbers. Denote these numbers as *p* and *q*.
2. Set *n* equal to *p* * *q*.
3. Choose any large integer, *d*, such that GCD(*d*, ((*p*-1) * (*q*-1))) = 1
4. Find *e* such that *e* * *d* = 1 (**mod** ((*p*-1) * (*q*-1)))

Rivest, Shamir, and Adleman provide efficient algorithms for each required operation[4].

How secure is a communication using RSA?

Cryptographic methods cannot be proven secure. Instead, the only test is to see if someone can figure out how to decipher a message without having direct knowledge of the decryption key. The RSA method's security rests on the fact that it is extremely difficult to factor very large numbers. If 100 digit numbers are used for *p* and *q*, the resulting *n* will be approximately 200 digits. The fastest known factoring algorithm would take far too long for an attacker to ever break the code. Other methods for determining *d* without factoring *n* are equally as difficult.

Any cryptographic technique which can resist a concerted attack is regarded as secure. At this point in time, the RSA algorithm is considered secure.

How Does RSA Works?

RSA is an **asymmetric** system, which means that a key pair will be generated (we will see how soon) , a **public** key and a **private** key , obviously you keep your private key secure and pass around the public one.

The algorithm was published in the 70's by Ron **R**ivest, Adi **S**hamir, and Leonard **A**dleman, hence RSA, and it sort of implement's a trapdoor function such as Diffie's one.

RSA is rather slow so it's hardly used to encrypt data, more frequently it is used to encrypt and pass around **symmetric** keys which can actually deal with encryption at a **faster** speed.

RSA Security:

- It uses prime number theory which makes it difficult to find out the key by reverse engineering.
- Mathematical Research suggests that it would take more than 70 years to find P & Q if N is a 100 digit number.

Algorithm

The RSA algorithm holds the following features −

- RSA algorithm is a popular exponentiation in a finite field over integers including prime numbers.
- The integers used by this method are sufficiently large making it difficult to solve.
- There are two sets of keys in this algorithm: private key and public key.

You will have to go through the following steps to work on RSA algorithm −

Step 1: Generate the RSA modulus

The initial procedure begins with selection of two prime numbers namely p and q, and then calculating their product N, as shown −

N=p*q

Here, let N be the specified large number.

Step 2: Derived Number (e)

Consider number e as a derived number which should be greater than 1 and less than (p-1) and (q-1). The primary condition will be that there should be no common factor of (p-1) and (q-1) except 1

Step 3: Public key

The specified pair of numbers **n** and **e** forms the RSA public key and it is made public.

Step 4: Private Key

Private Key **d** is calculated from the numbers p, q and e. The mathematical relationship between the numbers is as follows −

ed = 1 mod (p-1) (q-1)

The above formula is the basic formula for Extended Euclidean Algorithm, which takes p and q as the input parameters.

Encryption Formula

Consider a sender who sends the plain text message to someone whose public key is **(n,e).** To encrypt the plain text message in the given scenario, use the following syntax −

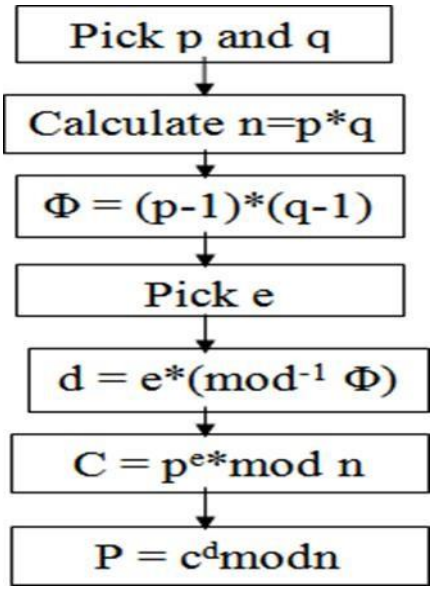
C = Pe mod n

Decryption Formula

The decryption process is very straightforward and includes analytics for calculation in a systematic approach. Considering receiver **C** has the private key **d**, the result modulus will be calculated as −

Plaintext = Cd mod n

Example



1. P=7, Q=17
2. 119=7*17
3. (7-1)*(17-1)= 6*16 =96 factor 2 & 3, so E=5
4. (D*5) mod (7-1)*(17-1)=1, so D=77
5. CT=10^5 mod 119 =100000 mod 119 =40
6. Send 40
7. PT=40^{77} mod 119 = 10

ALGORITHM:

STEP-1: Select two co-prime numbers as p and q.

STEP-2: Compute n as the product of p and q.

STEP-3: Compute (p-1)*(q-1) and store it in z.

STEP-4: Select a random prime number e that is less than that of z.

STEP-5: Compute the private key,dase*mod^{-1}(z).

STEP-6: The cipher text is computed as $message^{e}$*mod n.

STEP-7: Decryption is done as $cipher^{d}$ mod n.

CONCLUSION:

Thus we learn that to how to Encrypt and Decrypt the message by using RSA Algorithm.

EX.NO:6

IMPLEMENTATION OF DIFFIEHELLMAN KEY EXCHANGE ALGORITHM

AIM:

To implement the Diffie-Hellman Key Exchange algorithm using C language.

DESCRIPTION:

Diffie–Hellman Key Exchange establishes a shared secret between two parties that can be used for secret communication for exchanging data over a public network. It is primarily used as a method of exchanging cryptography keys for use in symmetric encryption algorithms like AES. The algorithm in itself is very simple. The process begins by having the two parties, Alice and Bob. Let's assume that Alice wants to establish a shared secret with Bob.

Diffie-Hellman key Exchange (DH)

In the mid- 1970's, Whitefield Diffie, a student at the Stanford University met with Martin Hellman, his professor &the two began to think about it. After some research & complicated mathematical analysis, they came up with the idea of AKC. Many experts believe that this development is the first & perhaps the only truly revolutionary concept in the history of cryptography

Silent Features of Diffie-Hellman key Exchange (DH)

1. Developed to address shortfalls of *key distribution* in symmetric key distribution.
2. A *key exchange algorithm,* not an encryption algorithm
3. Allows two users to share a *secret key* securely over a public network
4. Once the key has been shared Then both parties can use it to encrypt and decrypt messages using symmetric cryptography
5. Algorithm is based on "difficulty of calculating discrete logarithms in a finite field"
6. These keys are mathematically related to each other.
7. ''Using the public key of users, the session key is generated without transmitting the private key of the users.''

Diffie-Hellman Key Exchange/Agreement Algorithm with Example

1. Firstly, Alice and Bob agree on two large prime numbers, n and g. These two integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.

   Let n = 11, g = 7.

2. Alice chooses another large random number x, and calculates A such that:
   $A = g^x \bmod n$

   Let x = 3. Then, we have, $A = 7^3 \bmod 11 = 343 \bmod 11 = 2$.

3. Alice sends the number A to Bob.

   Alice sends 2 to Bob.

4. Bob independently chooses another large random integer y and calculates B such that:
   $B = g^y \bmod n$

   Let y = 6. Then, we have, $B = 7^6 \bmod 11 = 117649 \bmod 11 = 4$.

5. Bob sends the number B to Alice.

   Bob sends 4 to Alice.

6. A now computes the secret key K1 as follows:
   $K1 = B^x \bmod n$

   We have, $K1 = 4^3 \bmod 11 = 64 \bmod 11 = 9$.

7. B now computes the secret key K2 as follows:
   $K2 = A^y \bmod n$

   We have, $K2 = 2^6 \bmod 11 = 64 \bmod 11 = 9$.

Diffie -Hellman Key exchange

1. Public values:

- large prime p, generator g (primitive root of p)

2. Alice has secret value x, Bob has secret y

3. Discrete logarithm problem: given x, g, and n, find A

4. A → B: $g^x \pmod n$

5. B → A: $g^y \pmod n$

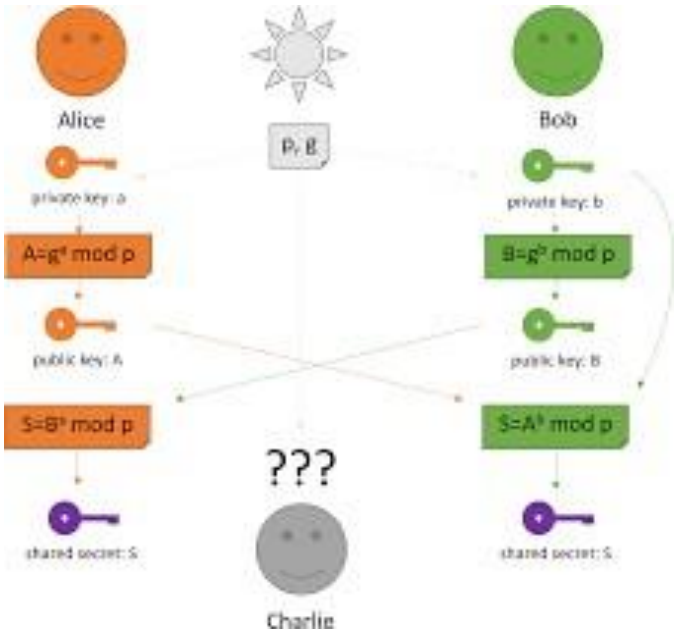
6. Bob computes $(g^x)^y = g^{xy} \pmod n$

7. Alice computes $(g^y)^x = g^{xy} \pmod n$

8. Symmetric key= $g^{xy} \pmod n$

Limitation: Vulnerable to "man in the middle" attacks*

EXAMPLE:



ALGORITHM:

STEP-1: Both Alice and Bob shares the same public keys g and p.

STEP-2: Alice selects a random public key a.

STEP-3: Alice computes his secret key A as g^a mod p.

STEP-4: Then Alice sends A to Bob.

STEP-5: Similarly Bob also selects a public key b and computes his secret key as B and sends the same back to Alice.

STEP-6 : Now both of them compute their common secret key as the other one's secret key power of a mod p.

CONCLUSION:

Thus we have studied and implement Diffie-Hellmen key exchange algorithm

EX.NO:7

IMPLEMENTATION OF MD5

AIM:

To write a C program to implement the MD5 hashing technique.

DESCRIPTION:

MD5 processes a variable-length message into a fixed-length output of 128 bits. The input message is broken up into chunks of 512-bit blocks. The message is padded so that its length is divisible by 512.The padding works as follows: first a single bit ,1 ,is appended to the end of the message. This is followed by as many zeros as are required to bring the length of the message upto 64 bits less than a multiple of 512.The remaining bits are filled up with 64bits representing the length of the original message, modulo 2^{64}.

The main MD 5 algorithm

operates on a 128-bit state, divided into four 32-bit words, denoted A, B, C, and D. These are initialized to certain fixed constants. The main algorithm then uses each 512-bit message block in turn to modify the state.

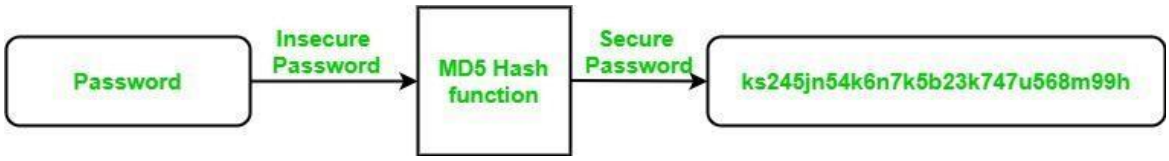
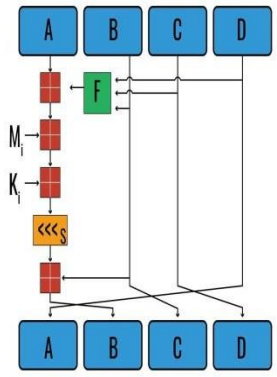


Fig 7.1 Simplified Block diagram



EXAMPLE:

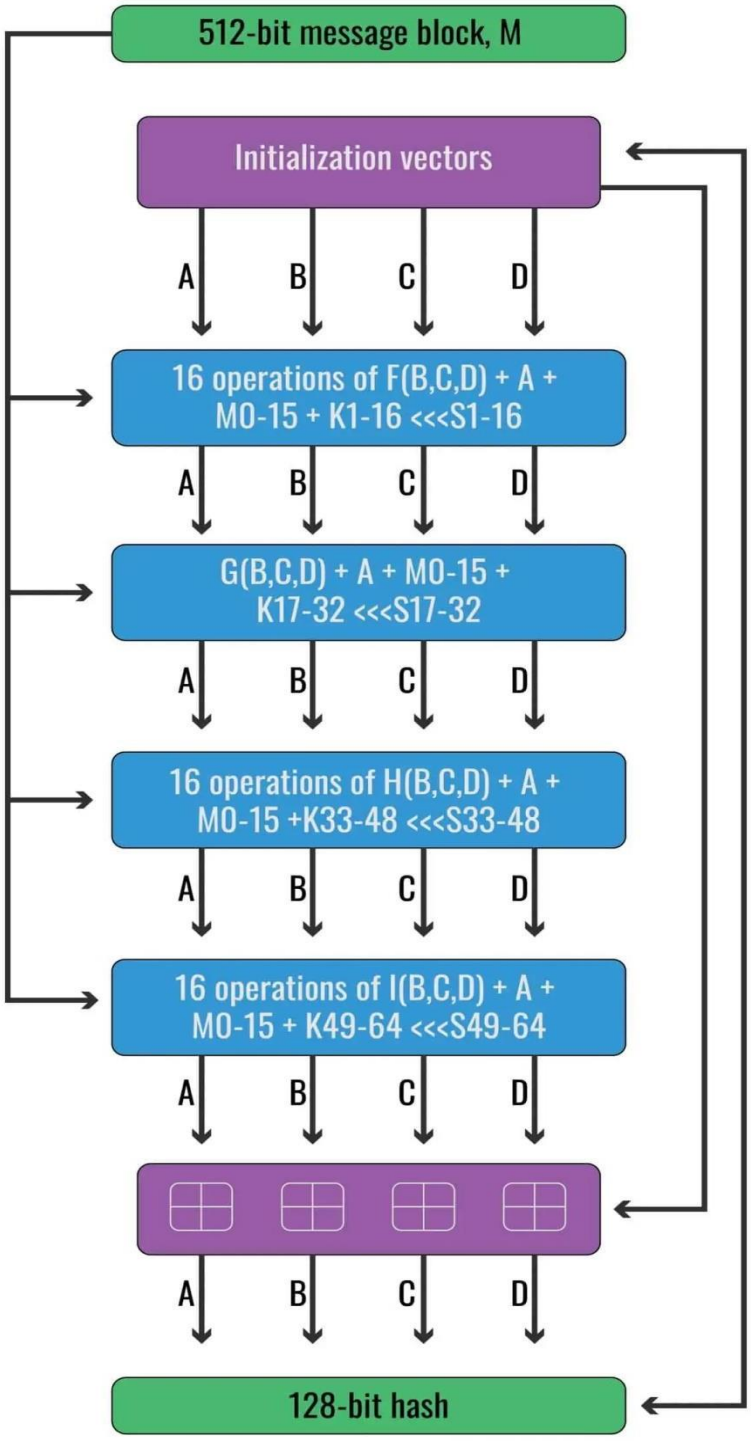


Fig 7.3 Structure of MD5

ALGORITHM:

STEP-1: Read the 128-bit plain text.

STEP-2: Divide into four blocks of 32-bits named as A,B,C and D.

STEP3: Compute the functions f, g, hand I with operations such as, rotations, permutations, etc,.

STEP4: The output of these functions are combined together as F and performed circular shifting and then given to key round.

STEP-5: Finally, right shift of' s'times are performed and the results are combined together to produce the final output.

CONCLUSION:

Thus we have studied and implement of MD5 hashing algorithm

STEP3: Compute the functions f, g, hand I with operations such as, rotations, permutations, etc,.

STEP4: The output of these functions are combined together as F and performed circular shifting and then given to key round.

STEP-5: Finally, right shift of‘ s’times are performed and the results are combined together to produce the final output.

CONCLUSION:

Thus we have studied and implement of MD5 hashing algorithm