SHIVNAGAR VIDYA PRASARAK MANDAL'S
COLLEGE OF ENGINEERING, MALEGAON (BK)-BARAMATI
(EN 307/6275) APPROVED BY AICTE NEW DELHI, GOVT. OF MAHARASHTRA, MUMBAI,
ACCREDITED BY NAAC, BANGALORE
AFFILIATED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, ID. NO. PU/PN/ENGG/084/1990

Institute Code - 6275

DEPARTMENT OF COMPUTER ENGINEERING

Third Year Of Computer Engineering

Course – 2019

310258 : Laboratory Practice II

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

Prepared By-
Prof. Yogesh R. Khalate
Assistant Professor Computer Engineering

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 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

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

Programming tools recommended: -

Information Security : - C/C++/Java
Software Modeling and Architectures:-Front end:HTML5, Bootstarp, JQuery, JS etc.
Backend: MySQL /MongoDB/NodeJS

Virtual Laboratory:

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

TABLE OF CONTENTS

GRAPH SEARCH

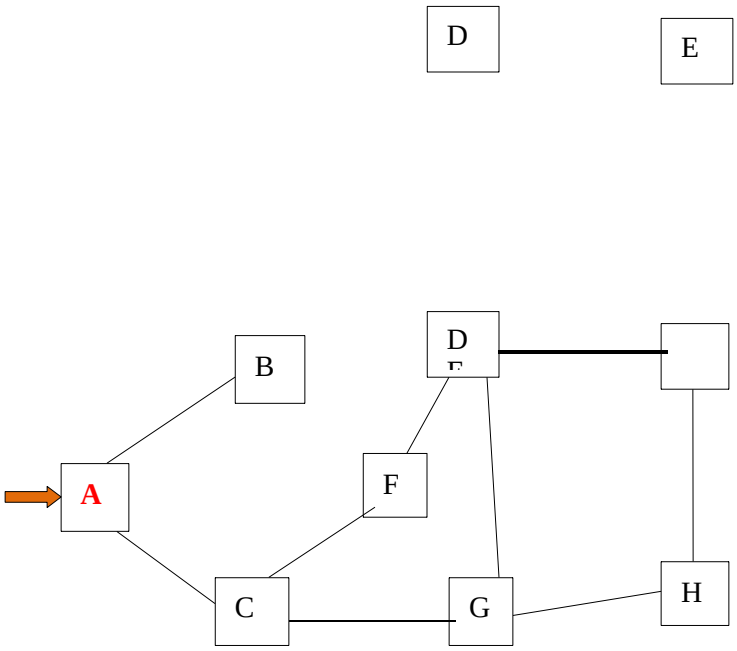
- Breadth first Search
- Depth first Search

Usage

- Transportation networks (airline carrier, airports as node and direct flights as edges (direct edge).
- Communication networks (a collection of computers as nodes and the physical link between them as edges).
- Information networks (World Wide Web can be viewed as directed graph, the Web pages are nodes and the hyperlink between the pages are directed edges).
- Social Network (People are nodes and friendship is an edge).

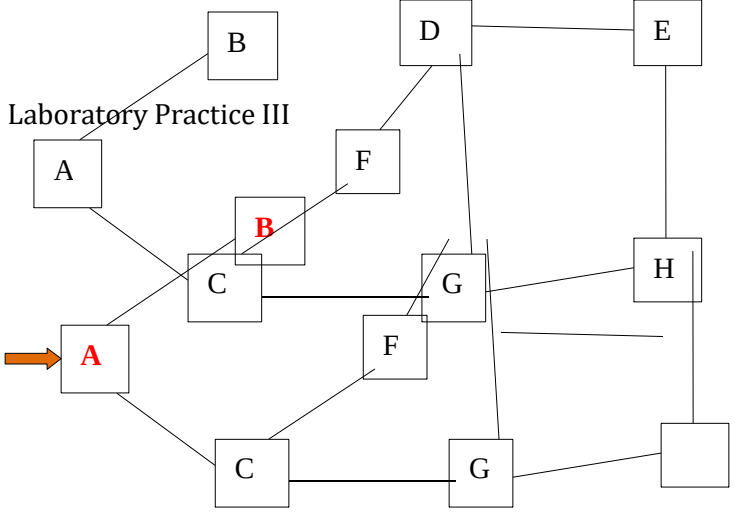
Breadth first Search(level order traversal): BFS begins at a root node and inspects all the neighboring nodes. Then for each of those neighbor nodes in turn, it inspects their neighbor nodes which were unvisited, and so on

Example:



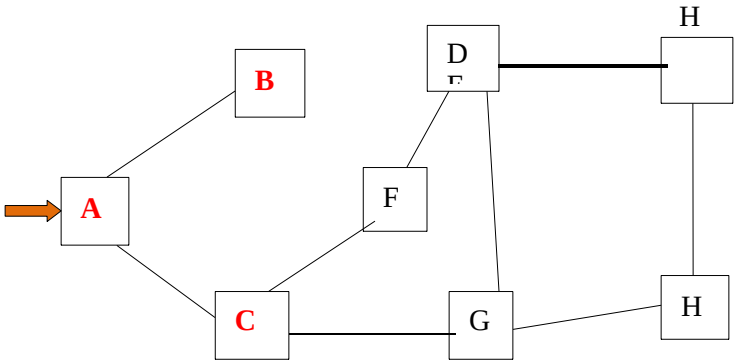
For implementation of BFS weuse queue as a data structure

Q {A}
Delete from queue vertex {A} & add their adjacent not visited vertices one bye one to the queue.
Print{}



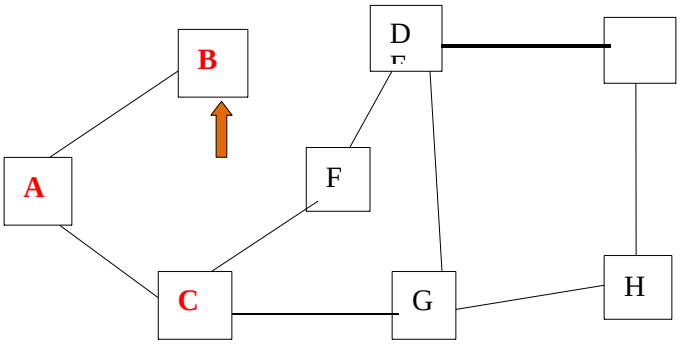
Q {B}

Print{A}

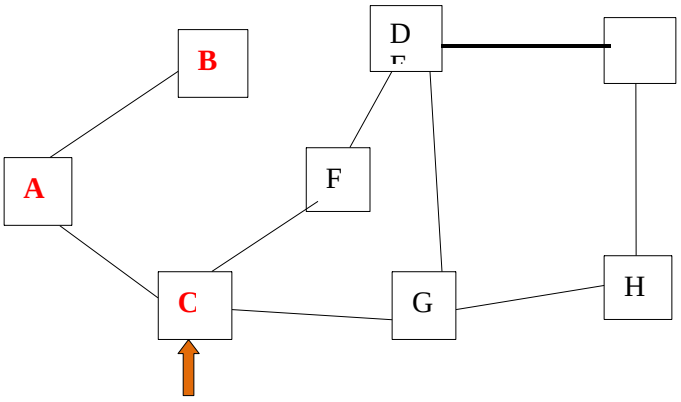


Q {B,C}

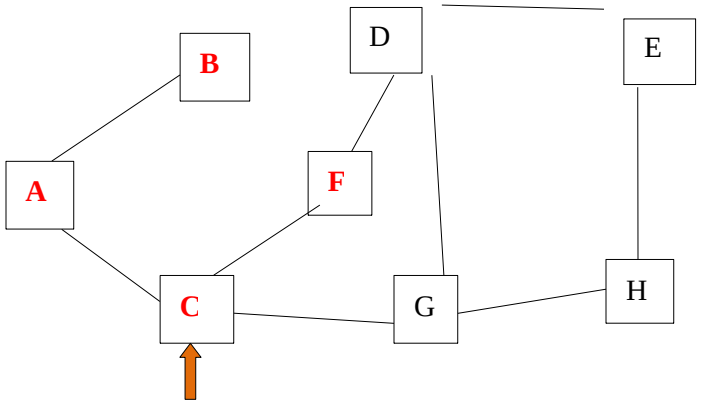
Print{A}



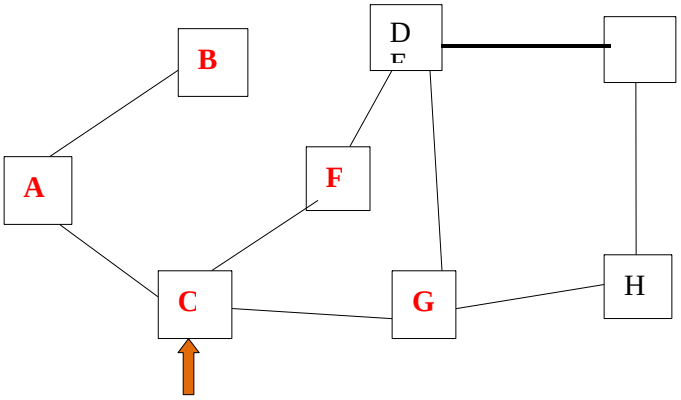
Q {B,C}
Delete vertex {B} & add not visited adjacent vertices of B to queue
Print{A}



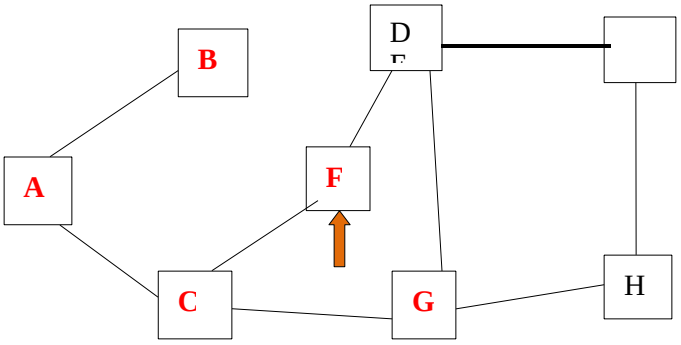
Q {C}
Delete vertex {C} & add not visited adjacent vertices of C to queue one bye one
Print{A, B}



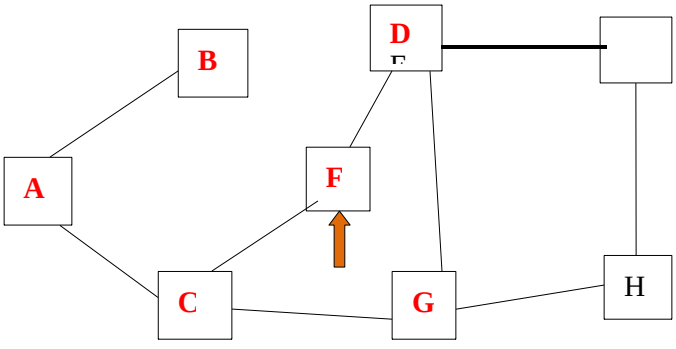
Q {F}
Print{A, B,C}



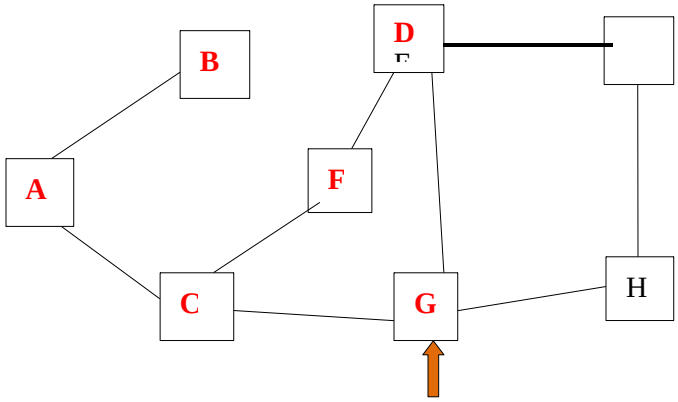
Q {F,G}
Print{A, B,C}



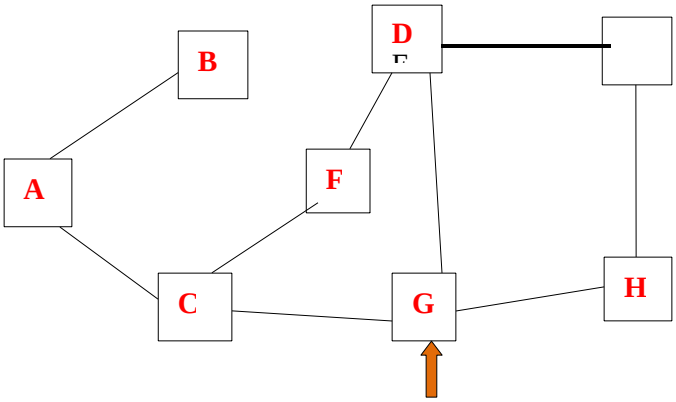
Q {F,G}
Delete vertex {F} & add not visited adjacent vertices of F to queue one bye one
Print{A, B,C}



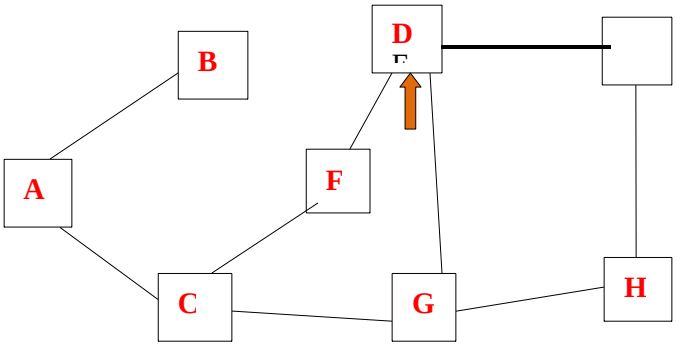
Q {G,D}
Print{A,
B,C,F}



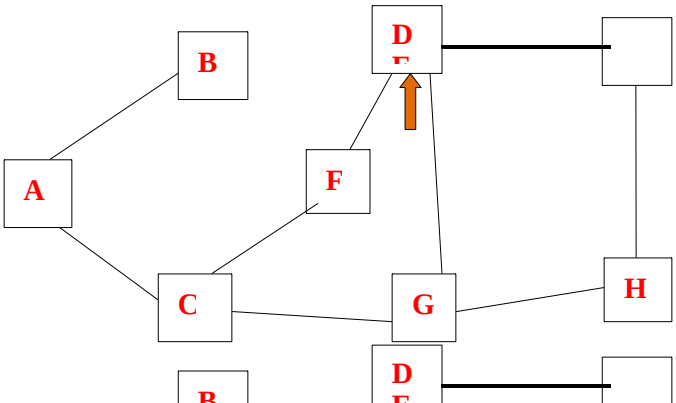
Q {G,D}
Delete vertex {G} & add not visited adjacent vertices of G to queue one bye one
Print{A, B,C,F}



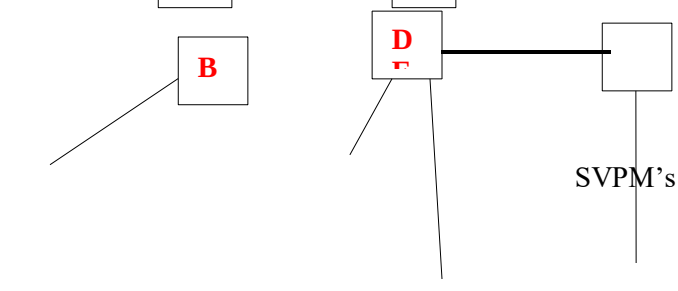
Q {D,H}
Print{A,
B,C,F,G}



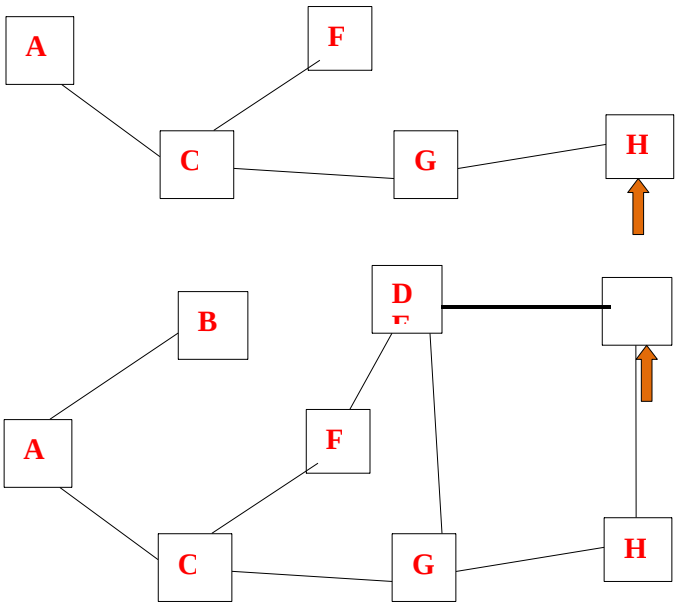
Q {D,H}
Delete vertex {D} & add not visited adjacent vertices of D to queue one bye one
Print{A, B,C,F,G}



Q {H,E}
Print{A, B,C,F,G,D}

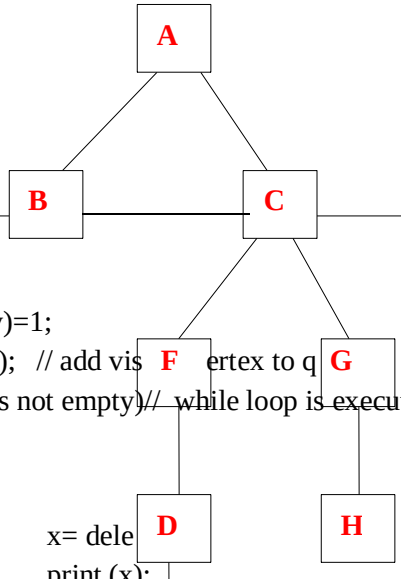


Q {E}
Delete vertex {H} & add not visited adjacent vertices of H to queue one bye one
Print{A, B,C,F,G,D,H}



Q {}
Delete vertex {E} & add not visited adjacent vertices of E to queue one bye one
Print{A, B,C,F,G,D,H,E}
Q is empty and we have traverse all the vertices in order **{ A, B C F G D H E }**

Tree after BFS run




```
BFS(V)
{
   Visited (v)=1;
   Add(v, Q);   // add vis ertex to q
  While(Q is not empty)// while loop is executing
Vtimes
  {
                x= dele
                print (x);
        For all w adjacent to x
        {
                    If( w is not
        {   visited(w)
        =1;
                    Add (w, Q);
        }
  }
}
```

1) If we represent the graph G by adjacency matrix then the running time of BFS algorithm is $O(V^2)$ because for find adjacent vertices of a particular vertex in adjacency matrix we require V time and for all the vertices running time will be V^2 , where V is thenumber of nodes.

2) If we represent the graph G by link lists then the running time of BFS algorithm is O(E + V), where E is the number of edges and V is the number of nodes.

BFS Applications

Breadth-first search can be used to solve many problems in graph theory, forexample:

- Finding all nodes within one connected component
- Finding the shortest path between two nodes u and v
- Finding the diameter of a graph.
- To obtain number of connected component we use BFS
- If in the uniform weighted graphs (all edges weight are same) to find out shortest path from source we can use BFT.
- Testing a graph for bipartiteness. If there are two vertices x,y in the same level (layer) L_i that are adjacent then the graph is not bipartite.

Depth-First Search (DFS)

We don't visit the nodes level by level. As long as there is an unvisited node adjacent to the current visited node we continue. Once we are stuck, trace back and go to a different branch.DFS use stack as data structure.

Depth-First Search Algorithm

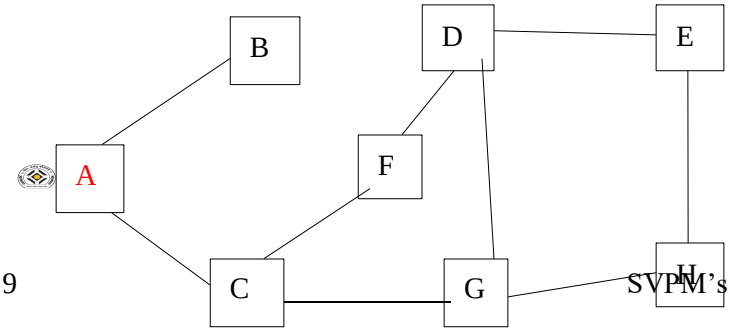
```
DFS(V)
{
   Visited
   (v)=1;
   Print(v);
   For all w adjacent to v
         {If( w is not
         visited)
                  {
                   DFS(w);
                  }

   }
```

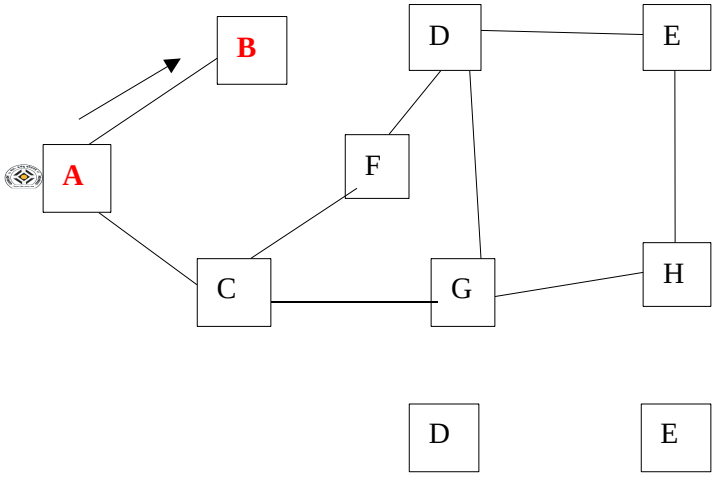
DFS Running time

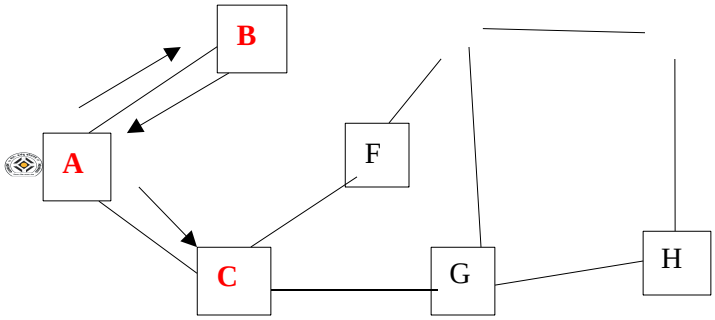
3) If we represent the graph G by adjacency matrix then the running time of DFS algorithm is $O(V^2)$ because for find adjacent vertices of a particular vertex in adjacency matrix we require V time and for all the vertices running time will be V^2 , where V is thenumber of nodes.

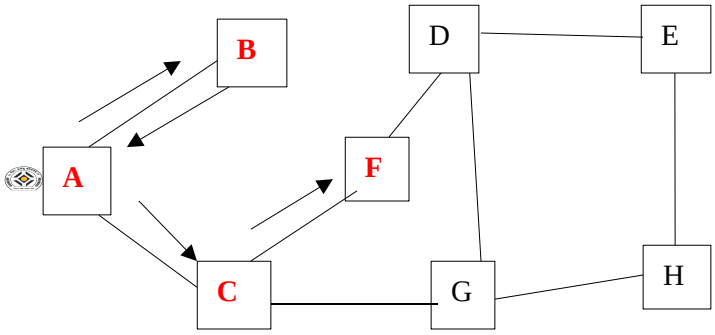
4) If we represent the graph G by link lists then the running time of DFS algorithm is O(E + V), where E is the number of edges and V is the number of nodes.

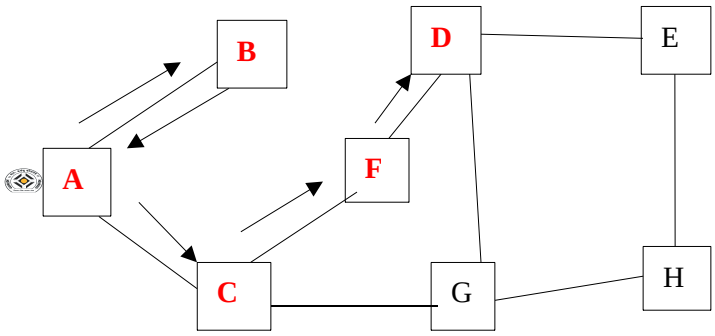
Example:

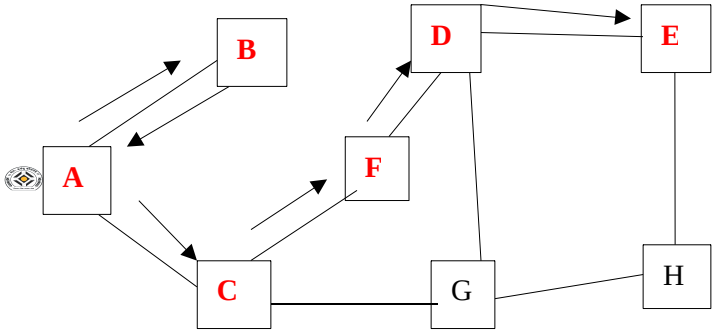



B
D
E
A
F
C
G
H
D
E

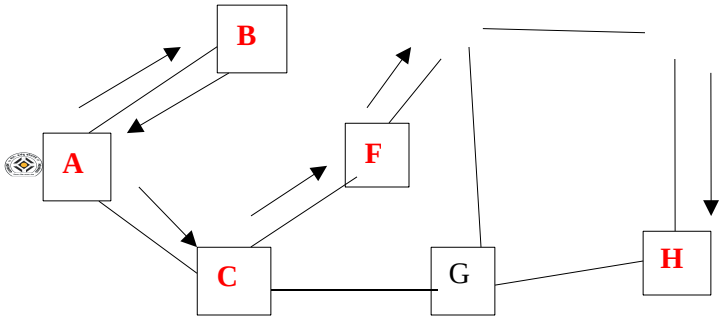

B
A
F
C
G
H

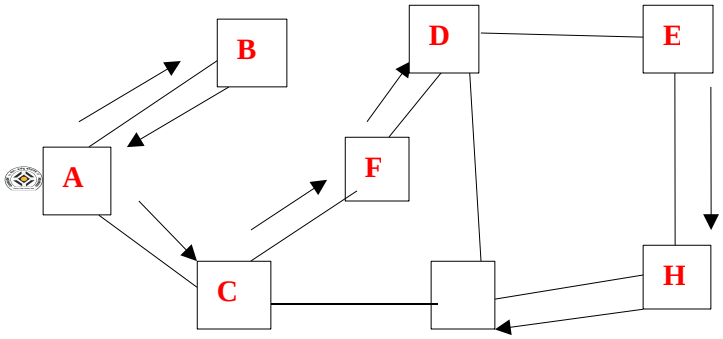

D
E
B
A
F
C
G
H


D
E
B
A
F
C
G
H


D
E
B
A
F
C
G
H

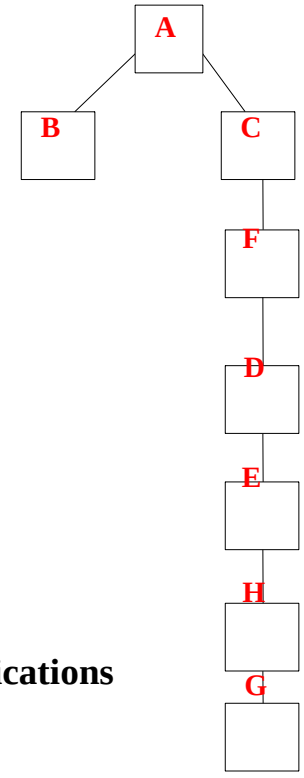

D
E





DFS{A,B,C,F,D,E,H,G}

Edges of Graph G that are not in DFS –{C-G, D-G}Tree after DFS run



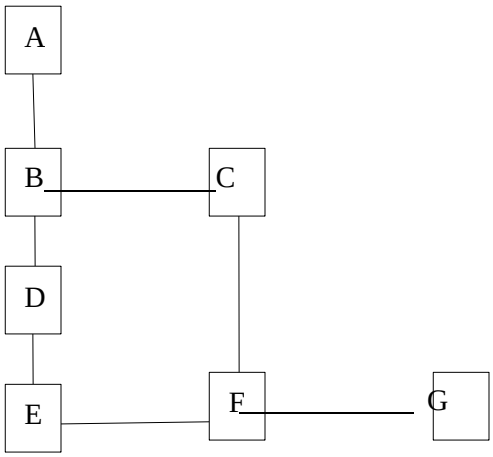
DFS Applications

- We use DFS to find a cycle or shortest cycle.
- We also use DFS to find strong components in digraph.
- Check whether graph is connected or not.
- Finding the number of connected component.
- Find articulation point in a graph. In the given connected graph by deletingany vertex if the graph is disconnected then the deleted vertex is called
  articulation point.

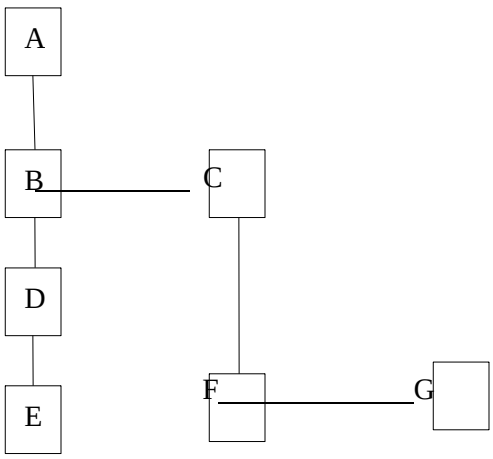
Example of BFS & DFS

V{A,B,C,D,E,F,G}

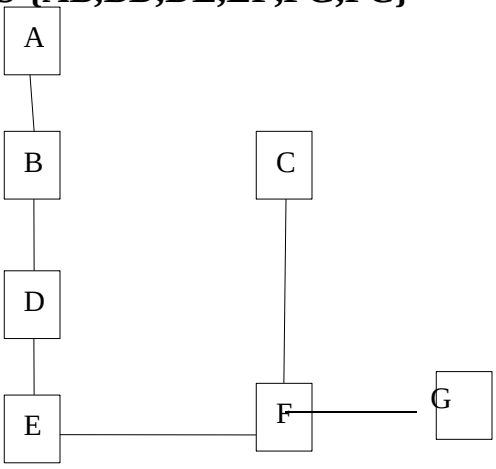
E{AB,BD,DE,BC,EF,CF,FG}



BFS-{AB,BD,BC,DE,CF,FG}



DFS-{AB,BD,DE,EF,FG,FC}



2	**Implement A star Algorithm for any game search problem**

PROBLEM STATEMENT:

Implement A star Algorithm for any game search problem

THEORY:

A Algorithm-*

- A* Algorithm is one of the best and popular techniques used for path finding and graph traversals.
- A lot of games and web-based maps use this algorithm for finding the shortest path efficiently.
- It is essentially a best first search algorithm.

Working-

A* Algorithm works as-

- It maintains a tree of paths originating at the start node.
- It extends those paths one edge at a time.
- It continues until its termination criterion is satisfied.

A* Algorithm extends the path that minimizes the following function-

$$f(n) = g(n) + h(n)$$

Here,

- 'n' is the last node on the path
- g(n) is the cost of the path from start node to node 'n'
- h(n) is a heuristic function that estimates cost of the cheapest path from node 'n' to the goal node

Algorithm-

- The implementation of A* Algorithm involves maintaining two lists- OPEN and CLOSED.
- OPEN contains those nodes that have been evaluated by the heuristic function but have not been expanded into successors yet.
- CLOSED contains those nodes that have already been visited.

The algorithm is as follows-

Step-01:

- Define a list OPEN.
- Initially, OPEN consists solely of a single node, the start node S.

Step-02:

If the list is empty, return failure and exit.

Step-03:

- Remove node n with the smallest value of f(n) from OPEN and move it to list CLOSED.
- If node n is a goal state, return success and exit.

Step-04:

Expand node n.

Step-05:

- If any successor to n is the goal node, return success and the solution by tracing the path from goal node to S.
- Otherwise, go to Step-06.

Step-06:

For each successor node,

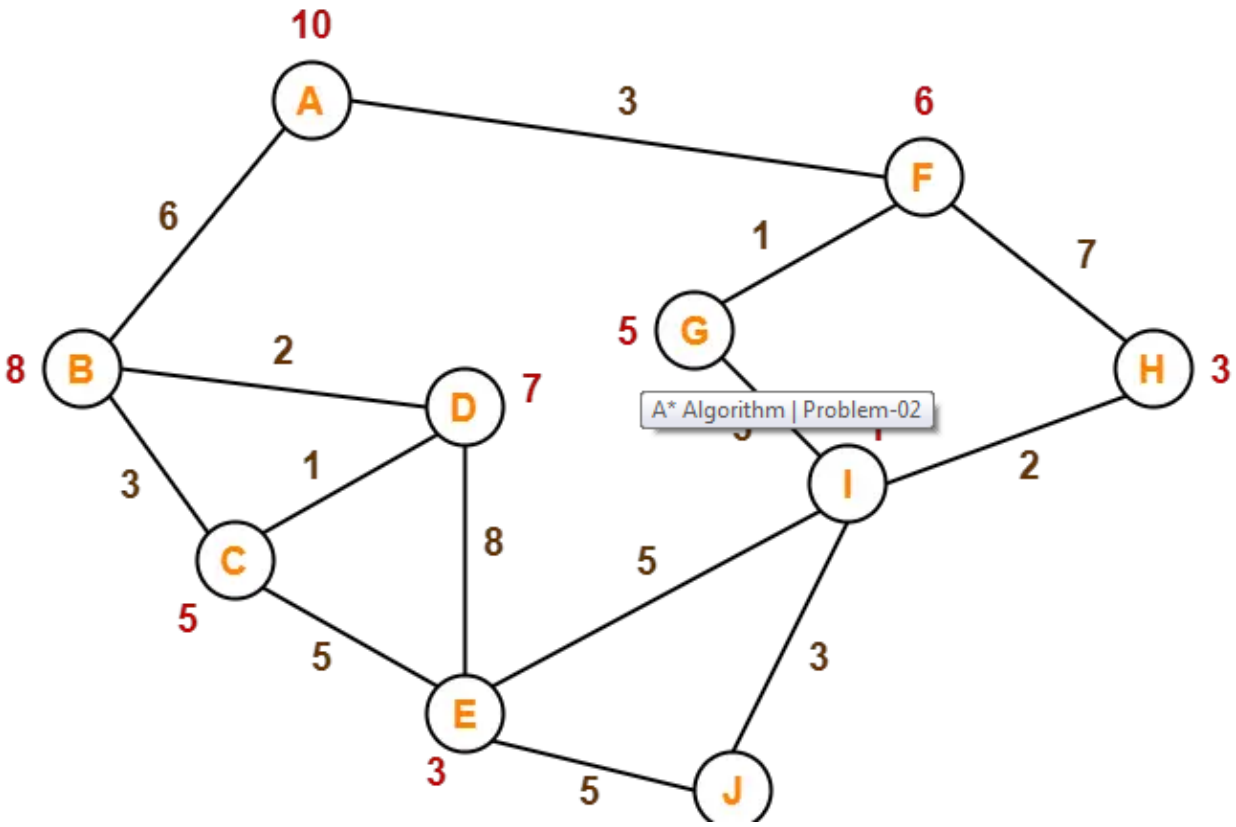
- Apply the evaluation function f to the node.
- If the node has not been in either list, add it to OPEN.

Step-07:

Go back to Step-02.

Problem-01:

Consider the following graph-



The numbers written on nodes represent the heuristic value.

Find the most cost-effective path to reach from start state A to final state J using A* Algorithm.

Solution-

Step-01:

- We start with node A.
- Node B and Node F can be reached from node A.

A* Algorithm calculates f(B) and f(F).

- f(B) = 6 + 8 = 14
- f(F) = 3 + 6 = 9

Since f(F) < f(B), so it decides to go to node F.

Path- A → F

Step-02:

Node G and Node H can be reached from node F.

A* Algorithm calculates f(G) and f(H).

- f(G) = (3+1) + 5 = 9
- f(H) = (3+7) + 3 = 13

Since f(G) < f(H), so it decides to go to node G.

Path- A → F → G

Step-03:

Node I can be reached from node G.

A* Algorithm calculates f(I).

f(I) = (3+1+3) + 1 = 8

It decides to go to node I.

Path- A → F → G → I

Step-04:

Node E, Node H and Node J can be reached from node I.

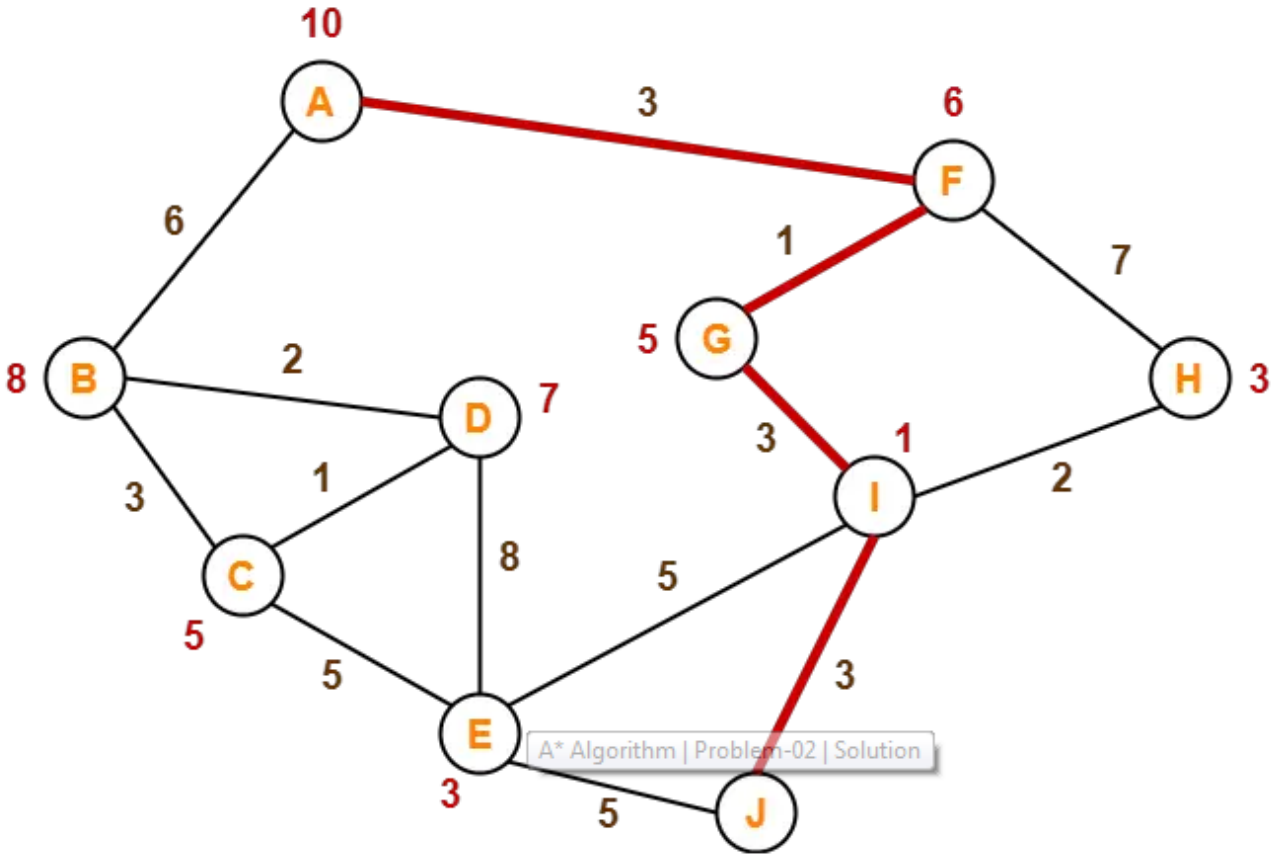
A* Algorithm calculates f(E), f(H) and f(J).

- f(E) = (3+1+3+5) + 3 = 15
- f(H) = (3+1+3+2) + 3 = 12
- f(J) = (3+1+3+3) + 0 = 10

Since f(J) is least, so it decides to go to node J.

Path- A → F → G → I → J

This is the required shortest path from node A to node J.



Important Note-

It is important to note that-

- A* Algorithm is one of the best path finding algorithms.
- But it does not produce the shortest path always.
- This is because it heavily depends on heuristics.

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** **Dijkstra's Minimal Spanning Tree Algorithm**

PROBLEM STATEMENT:

Implement Greedy search algorithm Single-Source Shortest Path Problem.

THEORY:

What is a Greedy Method?

- A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It doesn't worry whether the current best result will bring the overall optimal result.
- The algorithm never reverses the earlier decision even if the choice is wrong. It works in a top- down approach.
- This algorithm may not produce the best result for all the problems. It's

because it always goes for the local best choice to produce the global best result.

Advantages of Greedy Approach

- The algorithm is **easier to describe**.
- This algorithm can **perform better** than other algorithms (but, not in all cases).

Drawback of Greedy Approach

- As mentioned earlier, the greedy algorithm doesn't always produce the optimal solution. This is the major disadvantage of the algorithm
- For example, suppose we want to find the longest path in the graph below from root to leaf.

Greedy Algorithm

1. To begin with, the solution set (containing answers) is empty.
2. At each step, an item is added to the solution set until a solution is reached.
3. If the solution set is feasible, the current item is kept.
4. Else, the item is rejected and never considered again.

Single Source Shortest Path (SSSP) Problem

Given a directed graph G = (V,E), with non-negative costs on each edge, and a selected source node v in V, for all w in V, find the cost of the least cost path from v to w.

The *cost* of a path is simply the sum of the costs on the edges traversed by the path.

This problem is a general case of the more common subproblem, in which we seek the least cost path from v to a particular w in V. In the general case, this subproblem is no easier to solve than the SSSP problem.

Dijkstra's Algorithm

Dijkstra's algorithm is a *greedy* algorithm for the SSSP problem.

- A "greedy" algorithm always makes the locally optimal choice under the assumption that this will lead to an optimal solution overall.
- Example: in making change using the fewest number of coins, always start with the largest coin possible.

Data structures used by Dijkstra's algorithm include:

- a cost matrix C, where C[i,j] is the weight on the edge connecting node i to node j. If there is no such edge, C[i,j] = infinity.
- a set of nodes S, containing all the nodes whose shortest path from the source node is known. Initially, S contains only the source node.
- a distance vector D, where D[i] contains the cost of the shortest path (so far) from the source node to node i, using only those nodes in S as intermediaries.

The Algorithm

```
DijkstraSSSP (int N, rmatrix &C, vertex v, rvector &D)
{
  set S;
  int i;
  vertex k, w;
  float cw;

  Insert(S,v);
  for (k = 0; k < N; k++) D[k] = C[v][k];
  for (i = 1; i < N; i++) {
    /* Find w in V-S st. D[w] is minimum */
    cw = INFINITY;
    for (k = 0; k < N; k++)
      if (! Member(S,k) && D[k] < cw) {
```

```
            cw = D[k];
            w = k;
          }
      Insert(S, w);
      /* Forall k not in S */
      for (k = 0; k < N; k++)
          if (! Member(S,k))
           /* Shorter path from v to k using w? */
            if (D[w] + C[w][k] < D[k])
                  D[k] = D[w] + C[w][k];
   }
 } /* DijkstraSSSP */
```

How Dijkstra's Algorithm Works

On each iteration of the main loop, we add vertex w to S, where w has the least cost path from the source v (D[w]) involving only nodes in S.

We know that D[w] is the cost of the least cost path from the source v to w (even though it only uses nodes in S).

If there is a lower cost path from the source v to w going through node x (where x is not in S) then

- D[x] would be less than D[w]
- x would be selected before w
- x would be in S

Analysis of Dijkstra's Algorithm

Consider the time spent in the two loops:

1. The first loop has O(N) iterations, where N is the number of nodes in G.
2. The second (and outermost) loop is executed O(N) times.
    - The first nested loop is O(N) since we examine each vertex to determine whether or not

it is in V-S.

- The second nested loop is O(N) since we examine each vertex to determine whether or not it is in V-S.

The algorithm is O(N^2).

If we assume that there are many fewer edges than the maximum possible, we can do better than this, however.

Assume the following implementation details

- Use an adjacency list representation for the graph
- Maintain the set of nodes V-S as a priority queue
  - a partially ordered tree implementation of the priority queue allows updates in O(log N)

The new analysis is

1. The first loop still has O(N) iterations.
2. The second loop is still executed O(N) times.
   - The first nested loop is DeleteMin on the priority queue, which is O(log N).
   - Since each edge is considered at most once in the second nested loop (over the whole execution), and each access to the priority queue requires O(log N), the entire time spent in this loop is O(|E| log N).
3. The algorithm is O(|E| log N + N log N).

For N <= |E| <= N^2 the result is O(|E| log N), which is much better than O(N^2) in many cases.

4	**Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem**

PROBLEM STATEMENT:

Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem

THEORY:

8 Queens Problem using Backtracking

You are given an 8x8 chessboard, find a way to place 8 queens such that no queen can attack any other queen on the chessboard. A queen can only be attacked if it lies on the same row, or same column, or the same diagonal of any other queen. Print all the possible configurations.

To solve this problem, we will make use of the Backtracking algorithm. The backtracking algorithm, in general checks all possible configurations and test whether the required result is obtained or not. For thr given problem, we will explore all possible positions the queens can be relatively placed at. The solution will be correct when the number of placed queens = 8.

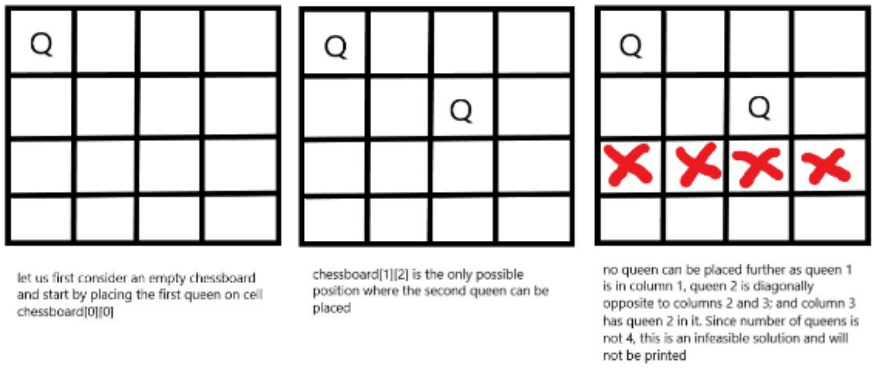
The time complexity of this approach is O(N!).

Input Format - the number 8, which does not need to be read, but we will take an input number for the sake of generalization of the algorithm to an NxN chessboard.

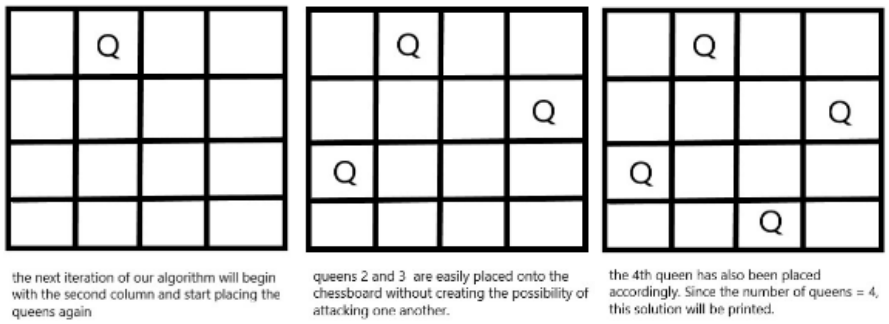
Output Format - all matrices that constitute the possible solutions will contain the numbers 0(for empty cell) and 1(for a cell where queen is placed). Hence, the output is a set of binary matrices.

Visualisation from a 4x4 chessboard solution :

In this configuration, we place 2 queens in the first iteration and see that checking by placing further queens is not required as we will not get a solution in this path. Note that in this configuration, all places in the third rows can be attacked.



As the above combination was not possible, we will go back and go for the next iteration. This means we will change the position of the second queen.



In this, we found a solution.

Now let's take a look at the backtracking algorithm and see how it works:

The idea is to place the queens one after the other in columns, and check if previously placed queens cannot attack the current queen we're about to place.

If we find such a row, we return true and put the row and column as part of the solution matrix. If such a column does not exist, we return false and backtrack*

Pseudocode

START

1. begin from the leftmost column
2. if all the queens are placed,
   return true/ print configuration
3. check for all rows in the current column
   a) if queen placed safely, mark row and column; and

```
      recursively check if we approach in the current
      configuration, do we obtain a solution or not
   b) if placing yields a solution, return true
   c) if placing does not yield a solution, unmark and
      try other rows
4. if all rows tried and solution not obtained, return
   false and backtrack
END
```

Link: https://iq.opengenus.org/8-queens-problem-backtracking/

8 Queens Problem using Branch and Bound:

The N-Queens problem is a puzzle of placing exactly N queens on an NxN chessboard, such that no two queens can attack each other in that configuration. Thus, no two queens can lie in the same row,column or diagnol.

The branch and bound solution is somehow different, it generates a partial solution until it figures that there's no point going deeper as we would ultimately lead to a dead end.

In the backtracking approach, we maintain an 8x8 binary matrix for keeping track of safe cells (by eliminating the unsafe cells, those that are likely to be attacked) and update it each time we place a new queen. However, it required O(n^2) time to check safe cell and update the queen.

In the 8 queens problem, we ensure the following:

1. no two queens share a row
2. no two queens share a column
3. no two queens share the same left diagnol
4. no two queens share the same right diagnol

we already ensure that the queens do not share the same column by the way we fill out our auxiliary matrix (column by column). Hence, only the left out 3 conditions are left out to be satisfied.

.

Applying the branch and bound approach :

The branch and bound approach suggets that we create a partial solution and use it to ascertain whether we need to continue in a particular direction or not. For this problem, we create 3 arrays to check for conditions 1,3 and 4. The boolean arrays tell which rows and diagnols are already occupied. To achieve this, we need a numbering system to specify which queen is placed

The indexes on these arrays would help us know which queen we are analysing.

Preprocessing - create two NxN matrices, one for top-left to bottom-right diagnol, and other for top-right to bottom-left diagnol. We need to fill these in such a way that two queens sharing same top-left_bottom-right diagnol will have same value in slashDiagonal and two queens sharing same top-right_bottom-left diagnol will have same value in backSlashDiagnol.

7	6	5	4	3	2	1	0
8	7	6	5	4	3	2	1
9	8	7	6	5	4	3	2
10	9	8	7	6	5	4	3
11	10	9	8	7	6	5	4
12	11	10	9	8	7	6	5
13	12	11	10	9	8	7	6
14	13	12	11	10	9	8	7

slash diagnol[row][col] = row + col

0	1	2	3	4	5	6	7
1	2	3	4	5	6	7	8
2	3	4	5	6	7	8	9
3	4	5	6	7	8	9	10
4	5	6	7	8	9	10	11
5	6	7	8	9	10	11	12
6	7	8	9	10	11	12	13
7	8	9	10	11	12	13	14

backslash diagnol[row][col] = row-col+(N-1)

For placing a queen *i* on row *j*, check the following :

1. whether row 'j' is used or not
2. whether slashDiagnol 'i+j' is used or not
3. whether backSlashDiagnol 'i-j+7' is used or not

If the answer to any one of the following is true, we try another location for queen **i** on row **j**, mark the row and diagnols; and recur for queen **i+1**.

Time Complexity

It has the same time complexity **O(N!)** as that of the backtracking algorithm, though it's performance is much better due to partial solution creation.

Link : https://iq.opengenus.org/8-queens-problem-using-branch-and-bound/

5	**Develop an elementary catboat for any suitable customer interaction application.**

PROBLEM STATEMENT:

Develop an elementary catboat for any suitable customer interaction application

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 of interpreting 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, chatbots can be used to share popular products with customers that they might find useful and can act as a sort of personal shopper or concierge service to find the perfect gift, meal or night out for a customer with just a few basic questions. 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 chatbots?

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, where in 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.

Types of Chatbots

Chatbots are categorized into two different types. Let us look at both and see how they function.
Rule-based chatbots.Chatbots follow a set of established rules or flows to respond to questions posted by a user. All your simple applications contain rule-based chatbots, which respond to queries based on the rules theyare trained on. For instance, a weather application, where you ask for a weather forecast and it fetches the data from different sources and responds with the information.
Rule-based chatbots may not be able to hold complex conversations. It can only accomplish the tasks it is programmed to perform unless more improvements are made by the developer.

Machine Learning-based chatbots Chatbots that are based on machine learning can hold more complex conversations as they try to process the question and understand the meaning behind the question. It learns from the previous conversation and enables itself to handle more complex questions in the future.

Now let's go through some of the benefits that chatbots provide:
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. Since chatbots are basically virtual robots they never get tired and continue to obey your command. They will continue to operate every day throughout the year without requiring to take a break. This improves your customer satisfaction and helps you rank highly in your sector.

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. No matter what time of the day it is or how many people are contacting you, every single one of them will be answered instantly. 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 to 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 one time investment which helps businesses reduce 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 an agent is having a good attitude or is in a good mood he will most probably talk to customers in a good way. In contrast 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. Also, in the travel and hospitality industry where travelers do not speak the same language, a bot can be trained to communicate in the language of the traveler.

5. Automation of repetitive work: Let's be honest, no one likes doing the same work again and again over a 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. For example, there are new items bought from your eCommerce site or there is a bug reported then it sends a short summary to a slack channel.

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. Also, the chatbot will remember all your choices and provide you with relevant choices the next time you visit it.

To create your own chatbot:

1 Identify your business goals and customer needs.
2 Choose a chatbot builder that you can use on your desired channels.
3 Design your bot conversation flow by using the right nodes.
4 Test your chatbot and collect messages to get more insights.
5 Use data and feedback from customers to train your bot.

How can chatbots 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. It makes customers feel they are working with someone to help resolve their issue. If done right, bots can help customers find what they are looking for and make them more likely to return.

Customer Engagement

Clearance Sale : Notify users about on-going clearance sale of products relevant to the users at their nearest outlets. Product Finder : Enable consultative selling without the need of a call center It offer Notification : Notify users about offers, product launches on products/ services they've shown interest in, and products that's back in stock

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.

You might argue that you have an app for that. Having an app for your brand is great, but having users discover that app, download it and use it to stay engaged is not an easy deal. 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 provides 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 a request to replace or return an order. Seek Reviews : Reach out to users to seek reviews on the products recently bought

Gift Recommendations

● Recommend relevant gifting options to users, accessing calendar events and understanding the likes and style of beneficiary.
● Opportunity to upsell gift cards for the users for every occasion.
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 have introduced their Chatbots to stand out from 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 also work to accept table reservations, take special requests and go the extra step to make the evening special for your guests.

Chatbots are not only good for the restaurant staff in reducing work and pain but can provide a better user experience for the customers.

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. Accurate and immediate delivery of information to customers is a major factor in running a successful online Business, especially in the price sensitive and competitive Travel and Hospitality industry. Chatbots particularly have gotten a lot of attention from the hospitality industry in recent months.

Chatbots can help hotels in a number of areas, including time management, guest services and cost reduction. They can assist guests with elementary questions and requests. Thus, freeing up hotel staff to devote more of their time and attention to time-sensitive, critical, and complicated tasks. They are often more cost effective and faster than their human counterparts. They can be programmed to speak to guests in different languages, making it easier for the guests to speak in their local language to communicate.

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 ease the access to healthcare and industry has favourable chances to serve their customers with personalised health tips. It can be a good example of the success of Chatbots and Service Industry combo.

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.

Chatbots, AI and Machine Learning pave a new domain of possibilities in the Fashion industry, from Data Analytics to Personal Chatbot Stylists. Fashion is such an industry where luxury goods can only be bought in a few physical boutiques and one to one customer service is essential. The Internet changed this dramatically, by giving the customers a seamless but a very impersonal experience of shopping. This particular problem can be solved by Chatbots.

Customers can be treated personally with bots, which can exchange messages, give required suggestions and information. Famous fashion brands like Burberry, Tommy Hilfiger have recently launched Chatbots for the London and New York Fashion Week respectively. Sephora a famous cosmetics brand and H&M– a fashion clothing brand have also launched their Chatbots.

Chatbots in Finance

Chatbots have already stepped into the Finance Industry. Chatbots can be programmed to assists the customers as Financial Advisor, Expense Saving Bot, Banking Bots, Tax bots, etc. Banks and Fintech have ample opportunities in developing bots for reducing their costs as well as human errors. Chatbots can work for customer's convenience, managing multiple accounts, directly checking their bank balance and expenses on particular things. Further about Finance and Chatbots have been discussed in our earlier blog: Chatbots as your Personal Finance Assistant.

Chatbots in Fitness Industry

Chat based health and fitness companies using Chatbot, to help their customers get personalized health and fitness tips. Tech based fitness companies can have a huge opportunity by developing their own Chatbots offering a huge customer base with personalized services. Engage with your fans like never before with news, highlights, game-day info, roster and more.

Chatbots and the Service Industry together have a wide range of opportunities and small to big all size of companies using chatbots to reduce their work and help their customers better.

Chatbots in Media

Big publisher or small agency, our suite of tools can help your audience chatbot experience rich and frictionless. Famous News and Media companies like The Wall Street Journal, CNN, Fox news, etc have launched their bots to help you receive the latest news on the go.

Chatbot in Celebrity:

With a chatbot you can now have one-on-one conversations with millions of fans. Chatbot in

Marketing

SMS Marketing

● Why promote just a coupon code that the customer does not know how to use?

S.N.J.B's. Late Sau. K B Jain College of Engineering, Chandwad 8

Laboratory Practice II Third Year Computer Engineering

● Improve conversions from your existing SMS campaigns.

● Talk to your customers when they want to using "Talk to an Agent" feature.

Email Marketing

● So your eMail has made a solid elevator pitch about your product.

● As a next step, is making customers fill an online form the most exciting way to engage with your customers?

● It's time to rethink the landing page.
● Instantly engage in a conversation with your customers.
● Address their concerns and queries

Social Media Triage

● How effectively are you addressing the negative sentiment around your brand on social media?
● Addressing queries instantly and effectively can convert even an angry customer into a loyal fan.
● Leverage a chatbot as your first response strategy and comfort that customer.

7. Questions:

Q 1: What is the use of a chat bot?

Q 2: Explain dialog flow in detail.

Q 3: What are the requirements for developing a chatbot?

Q 4: How do you evaluate a chatbot performance?

Q 5: How do I improve my chatbot accuracy?

8. Conclusion:

Smart solutions are important for the success of any business. From providing 24/7 customer service, improving current marketing activities, saving time spent on engaging with users to improving internal processes, chatbots can yield the much-needed competitive advantage. If you are looking to develop a chatbot, the best thing to do is to approach a company that will understand your business needs to develop a chatbot that helps you achieve your business goals.

7	**Write a Java/C/C++/Python program that contains a string (char pointer) with a value \HelloWorld'. The program should AND or and XOR each character in this string with 127 anddisplay the result**

PROBLEM STATEMENT:

Write a C program that contains a string (char pointer) with a value \HelloWorld'. The program should AND and XOR each character in this string with127 and display the result.

THEORY:

XOR Encryption is an encryption method used to encrypt data and is hard to crack by brute-force method, i.e

generating random encryption keys to match with the correct one.

In cryptography, the **simple XOR cipher** is a type of *additive cipher*, an encryption algorithm that operates according to the principles:
the exclusive disjunction (XOR) operation is sometimes called modulus 2 addition (or subtraction, which is identical). With this logic, a string of text can be encrypted by applying the bitwise XOR operator to every character using a given key. To decrypt the output, merely reapplying the XOR function with the key will remove the cipher.
Example
The string "Wiki" (01010111 01101001 01101011 01101001 in 8-bit ASCII) can be encrypted with the repeating key 11110011 as follows:

```
    01010111 01101001 01101011 01101001
    11110011 11110011 11110011 11110011
    -----------------------------------
=   10100100 10011010 10011000 10011010
```

And conversely, for decryption:

```
    10100100 10011010 10011000 10011010
    11110011 11110011 11110011 11110011
    -----------------------------------
=   01010111 01101001 01101011 01101001
```

Use and security

The XOR operator is extremely common as a component in more complex ciphers. By itself, using a constant repeating key, a simple XOR cipher can trivially be broken using frequency analysis. If the content of any message can be guessed or otherwise known then the key can be revealed. Its primary merit is that it is simple to implement, and that the XOR operation is computationally inexpensive. A simple repeating XOR (i.e. using the same key for xor operation on the whole data) cipher is therefore sometimes used for hiding information in cases where no particular security is required. The XOR cipher is often used in computer malware to make

reverse engineering more difficult.

If the key is random and is at least as long as the message, the XOR cipher is much more secure than when there is key repetition within a message. When the keystream is generated by a pseudo-random number generator, the result is a stream cipher. With a key that is truly random, the result is a one-time pad, which is unbreakable in theory.

The XOR operator in any of these ciphers is vulnerable to a known-plaintext attack, since *plaintext* *ciphertext* = *key*. It is also trivial to flip arbitrary bits in the decrypted plaintext by manipulating the ciphertext. This is called malleability.

Usefulness in cryptography

The primary reason XOR is so useful in cryptography is because it is "perfectly balanced"; for a given plaintext input 0 or 1, the ciphertext result is equally likely to be either 0 or 1 for a truly random key bit.

The table below shows all four possible pairs of plaintext and key bits. It is clear that if nothing is known about the key or plaintext, nothing can be determined from the ciphertext alone.

XOR Cipher Trace Table

Plaintext	Key	Ciphertext
0	0	0
0	1	1
1	0	1
1	1	0

ALGORITHM:

1.start

2.take the input ‘hello world’ which is assigned to variable ‘str’

3.perform XOR operation between the string and 127.

4.then print the result

5.stop. PROGRAM:

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

PROBLEM STATEMENT:

To implement a rail fence transposition technique in Java.

THEORY:

The railfence cipher is an easy to apply transposition cipher that jumbles up the order of the letters of a message in a quick convenient way. It also has the security of a key to make it a little bit harder to break.

The Rail Fence cipher works by writing your message on alternate lines across the page, and then reading off each line in turn For example, let's consider the **plaintext** "This is a secret message".

Rail Fence	T		I		I		A		E		R		T		E		S		G	
Encoding		H		S		S		S		C		E		M		S		A		E

To encode this message we will first write over two lines (the "rails of the fence") as follows:

Ciphertext T I I A E R T E S G H S S S C E M S A E

Note that all white spaces have been removed from the plain text.

The **ciphertext** is then read off by writing the top row first, followed by the bottom row:

ALGORITHM :

1. In the rail fence cipher, the plaintext is written downwards and diagonally on successive "rails" of an imaginary fence, then moving up when we reach the bottomrail.
2. When we reach the top rail, the message is written downwards again until the wholeplaintext is written out.
3. The message is then read off in rows.

CONCLUSION:

Thus the program for Rail Fence cipher encryption and decryption algorithm

hasbeen implemented and the output verified successfully.

Ex. No : 9 Date :	Data Encryption Standard (DES) Algorithm (User Message Encryption)

AIM:

To use Data Encryption Standard (DES) Algorithm for a practical application like User Message Encryption.

Theory:

It is a 16-round Feistel cipher, and the plaintext and ciphertext is 64 bit and DES is most widely used encryption scheme adapted in 1977s by NIST and they made this as a standard and historically DES comes from the cipher which is called LUCIFER which was designed by IBM in 60s and they have modified this slightly.

So, it is a 16-round block cipher, so we have a plain text which is 64-bit and we have 16 rounds. F1, F2,..., F16 for each round we know we need another input which are called round key. So, you have 16 round keys and these round keys are coming from key scheduling algorithm, which takes the secret key shared between Alice and Bob and gives the round keys.

DES round function is called Feistel cipher.

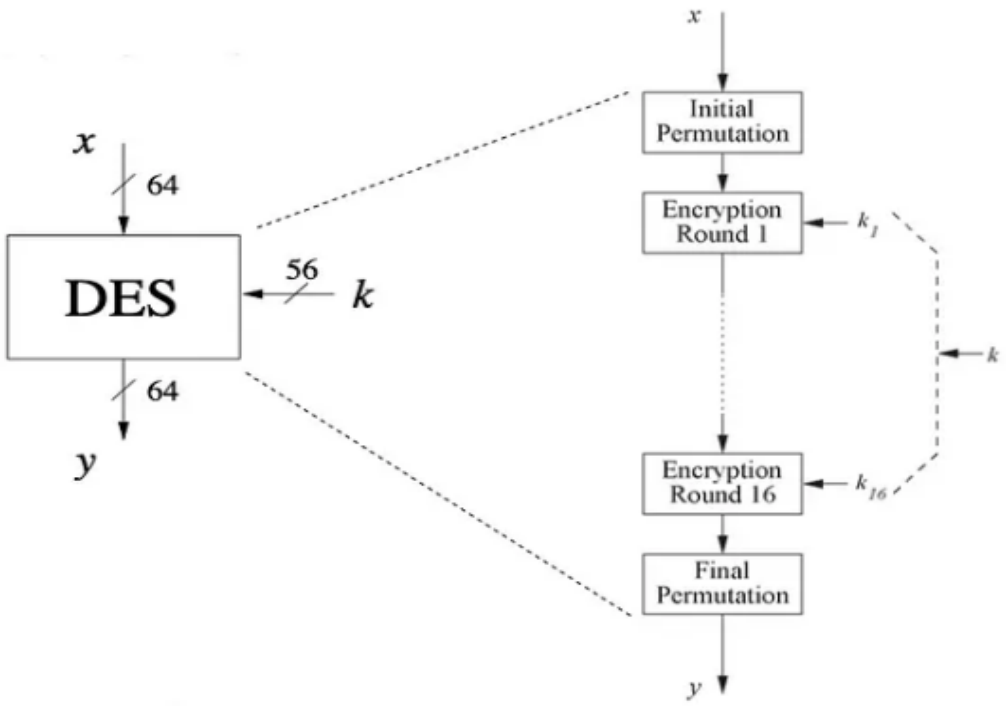
Feistel cipher (refer above image) has a 64-bit plaintext, we denote this by L0 and R0 each is 32-bit. Output is L1 and R1 both are 32-bit in size. So, R0 is directly copied to L1 and for R1, we have to apply function F which takes as input R0 and the round key and then it is bitwise XOR with L0 to give us R1.

So,

L1 = R0,

R1 = F(R0,Ki)^L0, where Ki is the round key of size 48-bit and F gives 32-bit output.

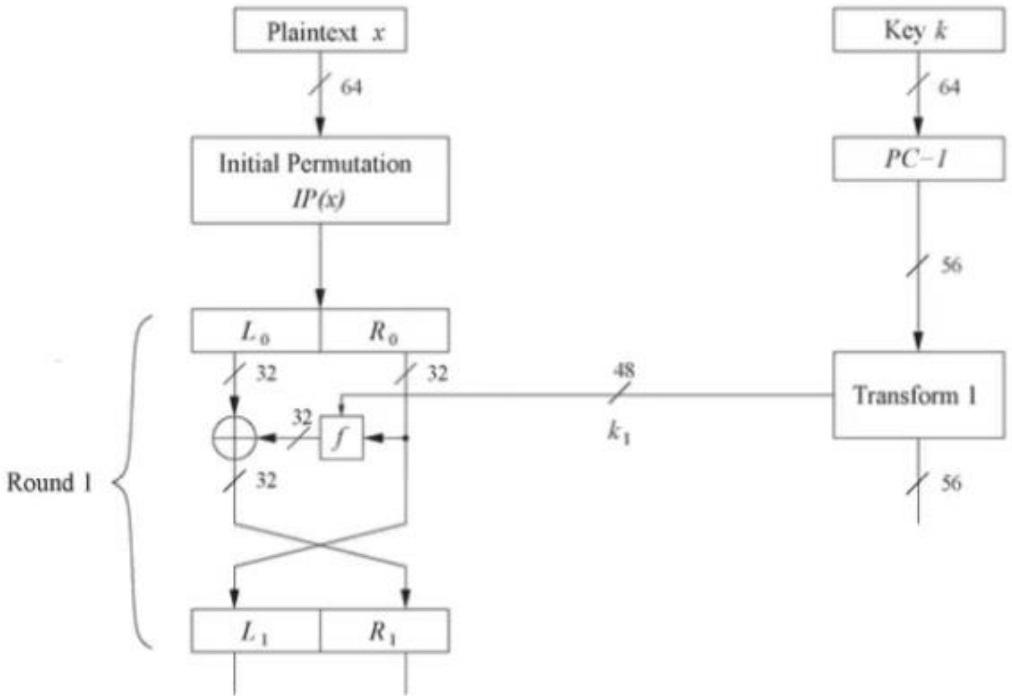
So, each round of DES is basically this Feistel function. How to get this round, we will talk about the key scheduling algorithm from the secret key.



The DES Feistel Network (1)

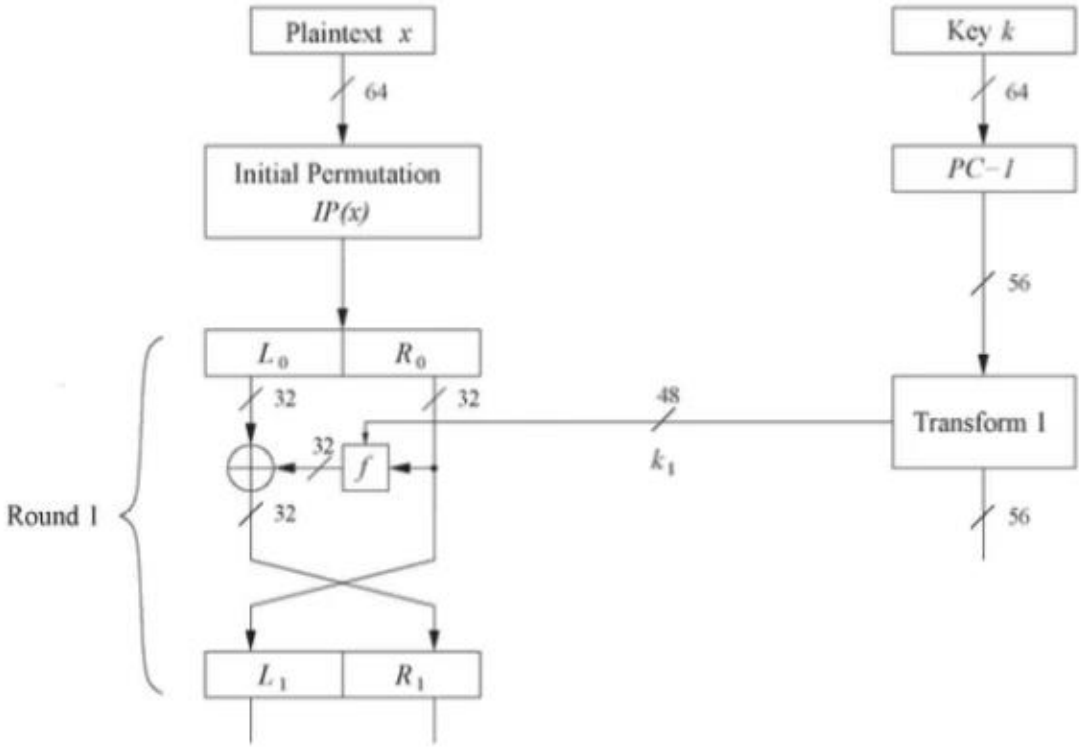
- DES structure is a *Feistel network*
- Advantage: encryption and decryption differ only in the key schedule
- Bitwise initial permutation, then 16 round
  - ◇ The plaintext is split into 32-bit halves $L_i$and R_i
  - ◇ *Ri* is fed into the function *f*, the output of which is then XORed with *Li*
- The Left and right half are swapped
- Rounds can be expressed as $L_i = R_{i-1}$, $R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$

Link: https://medium.com/@amit28amical/data-encryption-standard-des-code-in-java-4a45ad692bae



The DES Feistel Network (2)

- L and R swapped again at the end of the cipher, i.e., after round 16 followed by a final permutation


Plaintext x
64
Initial Permutation
IP(x)
L0
R0
32
32
32
f
32
Round 1
L1
R1
Key k
64
PC–1
56
48
k1
Transform 1
56

10	RSA Algorithm

PROBLEM STATEMENT:

To implement RSA (Rivest–Shamir–Adleman) algorithm by using HTML and Javascript.

THEORY:

RSA

In 1978, Rivest, Shamir and Adleman of MIT proposed a number-theoretic way of implementing a Public Key Cryptosystem. Their method has been widely adopted. The basic technique is:

- Choose two large prime numbers, *p* and *q*.
- compute $n = p * q$ and $x = (p-1)*(q-1)$
- Choose a number relatively prime to *x* and call it *d*. This means that *d* is not a prime factor of *x* or a multiple of it.
- Find *e* such that $e * d = 1 \bmod x$.

To use this technique, divide the plaintext (regarded as a bit string) into blocks so that each plaintext message *P* falls into the interval $0 <= P < n$. This can be done by dividing it into blocks of *k* bits where *k* is the largest integer for which $2^k < n$ is true.

To encrypt: $C = P^e \pmod n$

To decrypt: $P = C^d \pmod n$

The public key, used to encrypt, is thus: (*e*, *n*) and the private key, used to decrypt, is (*d*, *n*))

RSA in Practice

RSA works because knowledge of the public key does not reveal the private key. Note that both the public and private keys contain the important number $n = p * q$. The security of the system relies on the fact that *n* is **hard to factor** -- that is, given a large number (even one which is known to have only two prime factors) there is no easy way to discover what they are. This is a well known mathematical fact. If a fast method of factorisation *is* ever discovered then RSA will cease to be useful.

It **is** obviously possible to break RSA with a **brute force** attack -- simply factorise *n*. To make this difficult, it's usually recommended that *p* and *q* be chosen so that *n* is (in 2002 numbers) at least 1024 bits.

One **excellent** feature of RSA is that it is **symmetrical**. We already know that if you encrypt a message with my public key then only I can decrypt that ciphertext, using my secret key. However, it

also turns out that a message encrypted with my secret key **can only be decrypted with my public key**

.

The RSA algorithm operates with *huge* numbers, and involves lots of exponentiation (ie, repeated multiplication) and modulus arithmetic. Such operations are **computationally expensive** (ie, they take a long time!) and so RSA encryption and decryption are **incredibly slow**, even on fast computers. Compare this to the operations involved in **DES** (and other single-key systems) which consist of repeated simple **XOR**s and transpositions. Typical numbers are that DES is 100 times faster than RSA on equivalent hardware. Furthermore, DES can be easily implemented in dedicated hardware (RSA is, generally speaking, a software-only technology) giving a speed improvement of up to 10,000 times.

Advantages of RSA algorithm

- **No Key Sharing**: RSA encryption is performed using the public key of the receiver, therefore there is no need to share the private key.
- **Proof of Authenticity**: Since the key pairs are related to each other, only the receiver can decrypt the message using its private key.
- **Faster Encryption**: The encryption process is fast.
- **Ensures Integrity**: Data will be tamper-proof in transit since meddling with the data will alter the usage of the keys. And the private key won't be able to decrypt the information, hence alerting the receiver of manipulation.

ALGORITHM:

1. Choose two prime number p and q
2. Compute the value of n and **p**
3. Find the value of ***e*** (public key)
4. Compute the value of ***d*** (private key) using gcd()
5. Do the encryption and decryption
    a. Encryption is given as,

$$c = t^e \bmod n$$

    b. Decryption is given as,

$$t = c^d \bmod n$$

PROGRAM:

rsa.html

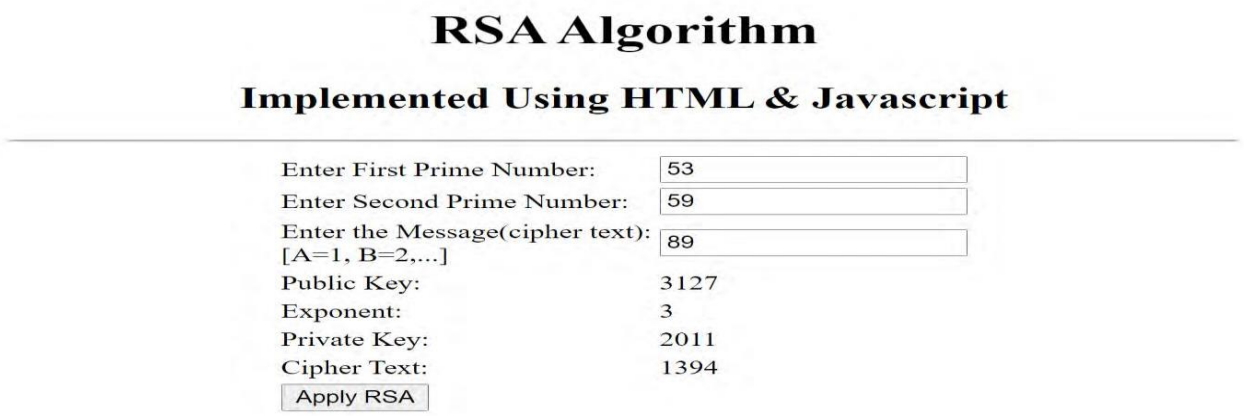
```
<html>

<head>
    <title>RSA Encryption</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>

<body>
    <center>
        <h1>RSA Algorithm</h1>
        <h2>Implemented Using HTML & Javascript</h2>
        <hr>
        <table>
            <tr>
                <td>Enter First Prime Number:</td>
                <td><input type="number" value="53" id="p"></td>
            </tr>
            <tr>
                <td>Enter Second Prime Number:</td>
                <td><input type="number" value="59" id="q"></p>
                </td>
            </tr>
            <tr>
                <td>Enter the Message(cipher text):<br>[A=1, B=2,...]</td>
                <td><input type="number" value="89" id="msg"></p>
```

```
            </td>
        </tr>
        <tr>
            <td>Public Key:</td>
            <td>
                <p id="publickey"></p>
            </td>
        </tr>
        <tr>
            <td>Exponent:</td>
            <td>
                <p id="exponent"></p>
            </td>
        </tr>
        <tr>
            <td>Private Key:</td>
            <td>
                <p id="privatekey"></p>
            </td>
        </tr>
        <tr>
            <td>Cipher Text:</td>
            <td>
                <p id="ciphertext"></p>
            </td>
        </tr>
        <tr>
            <td><button onclick="RSA();">Apply RSA</button></td>
         </tr>
      </table>
</center>
```

```
</body>
<script type="text/javascript">
    function RSA() {
        var gcd, p, q, no, n, t, e, i, x;
        gcd = function (a, b) { return (!b) ? a : gcd(b, a % b); };p =
        document.getElementById('p').value;
        q = document.getElementById('q').value;
        no = document.getElementById('msg').value;n = p *
        q;
        t = (p - 1) * (q - 1);

        for (e = 2; e < t; e++) {if
            (gcd(e, t) == 1) {
                break;
            }
        }




        for (i = 0; i < 10; i++) {x =
            1 + i * t
            if (x % e == 0) {d
                = x / e; break;
            }
        }

        ctt = Math.pow(no, e).toFixed(0);ct =
        ctt % n;
```

```
        dtt = Math.pow(ct, d).toFixed(0);dt =
        dtt % n;


        document.getElementById('publickey').innerHTML = n;
        document.getElementById('exponent').innerHTML = e;
        document.getElementById('privatekey').innerHTML = d;
        document.getElementById('ciphertext').innerHTML =  ct;
    }
</script>
</html>
```

OUTPUT:



RESULT:

Thus the RSA algorithm has been implemented using HTML & CSS and the output has been verified successfully.

11	Diffie-Hellman key exchange algorithm

PROBLEM STATEMENT:

To implement the Diffie-Hellman Key Exchange algorithm for a given problem

THEORY:

The problem here is to exchange a secret key (think of it as a password) without actually exchanging it. Because if you do the exchange, there is a chance that a third person, who is listening, would then know your secret.

But how is that even possible? How can two people who have never met agree on a secret key (number) with a guarantee that no third person can guess or find that out?

The way out? Diffie-Hellman Key Exchange Technique

Diffie-Hellman is now used almost everywhere every time to exchange keys

Diffie-Hellman Key Exchange:

a cryptographic algorithm requires a shared key between the sender and the receiver. The sender uses the key to encrypt the message, while the receiver needs the key to decrypt it. Without the shared key, they wouldn't be able to understand each other's messages.

Diffie-Hellman key exchange can be used together to securely exchange keys over an insecure network. It works using the concepts of prime numbers and modulo operations.

Here comes the genius of Diffie and Hellman. To golden rules:

Diffie, Hellman

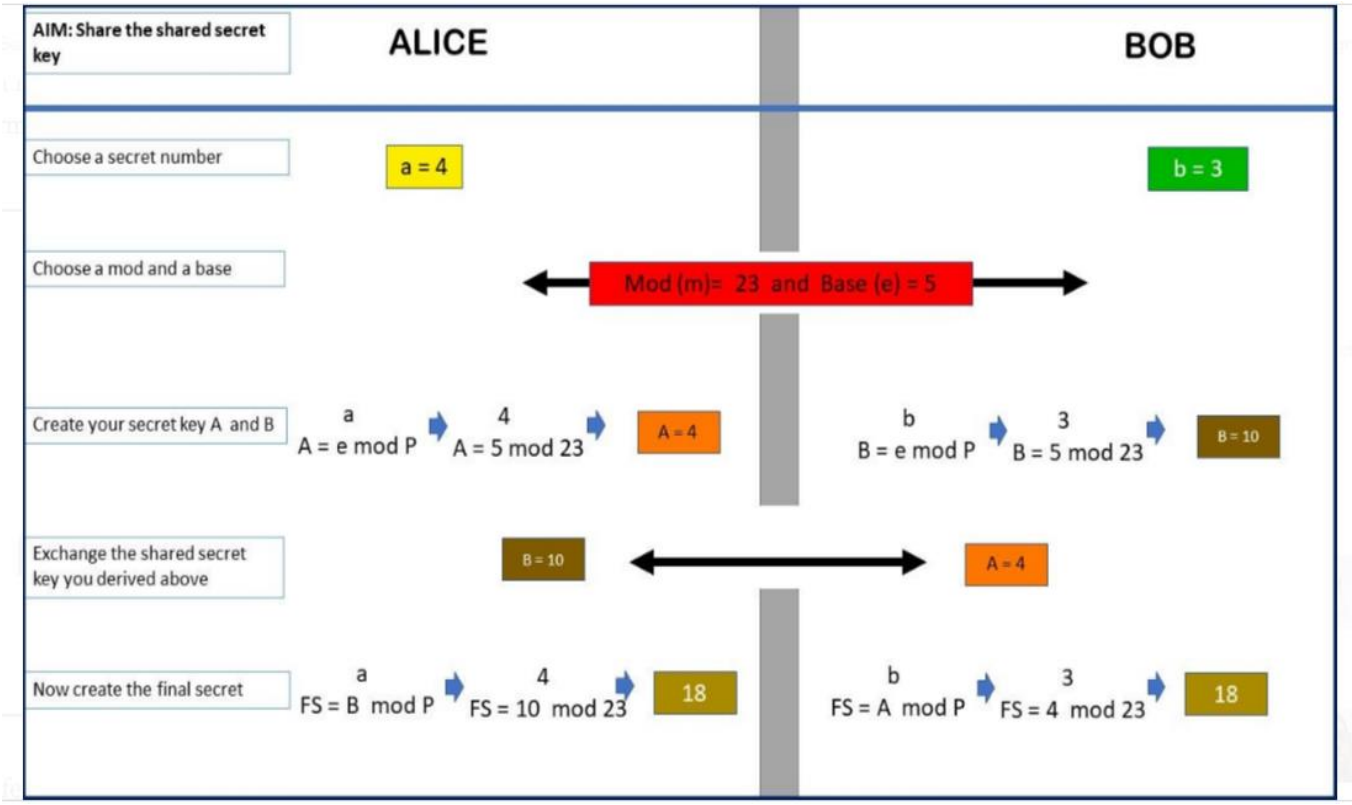
In order to agree on something without sharing, you need to have:

1. something which is easy to do one way, impossibly to reverse it,
2. and can be done in more than one way.

It means easy to 'do' it, hard to 'undo' it.

Example

- Making omelet from egg is easy. Making egg from omelet is impossible.
- Or mixing two colors is easy. Unmixing them is likely impossible.
- Mixing three colors is as easy. Unmixing them isn't.



ALGORITHM:

1. Alice and Bob publicly agree to use a modulus $p = 23$ and base $g = 5$ (which is a primitive root modulo 23).
2. Alice chooses a secret integer $a = 4$, then sends Bob $A = g^a \bmod p$
   - $A = 5^4 \bmod 23 = 4$
3. Bob chooses a secret integer $b = 3$, then sends Alice $B = g^b \bmod p$
   - $B = 5^3 \bmod 23 = 10$
4. Alice computes $s = B^a \bmod p$
   - $s = 10^4 \bmod 23 = 18$
5. Bob computes $s = A^b \bmod p$
   - $s = 4^3 \bmod 23 = 18$
6. Alice and Bob now share a secret (the number 18).

PROGRAM:

DiffieHellman.java

```
class DiffieHellman {
   public static void main(String args[]) {
      int p = 23; /* publicly known (prime number) */
      int g = 5; /* publicly known (primitive root) */
      int x = 4; /* only Alice knows this secret */
      int y = 3; /* only Bob knows this secret */
      double aliceSends = (Math.pow(g, x)) % p;
      double bobComputes = (Math.pow(aliceSends, y)) % p;
      double bobSends = (Math.pow(g, y)) % p;
      double aliceComputes = (Math.pow(bobSends, x)) % p;
      double sharedSecret = (Math.pow(g, (x * y))) % p;
      System.out.println("simulation of Diffie-Hellman key exchange algorithm\n--
_____");
      System.out.println("Alice Sends : " + aliceSends);
      System.out.println("Bob Computes : " + bobComputes);
      System.out.println("Bob Sends : " + bobSends);
      System.out.println("Alice Computes : " + aliceComputes);
      System.out.println("Shared Secret : " + sharedSecret);
```

```
        /* shared secrets should match and equality is transitive */
        if ((aliceComputes == sharedSecret) && (aliceComputes == bobComputes))
            System.out.println("Success: Shared Secrets Matches! " + sharedSecret);
        else
            System.out.println("Error: Shared Secrets does not Match");
    }
}
```

OUTPUT:

```
simulation of Diffie-Hellman key exchange algorithm
---------------------------------------------------------------------
Alice Sends : 4.0
Bob Computes : 18.0
Bob Sends : 10.0
Alice Computes : 18.0
Shared Secret : 18.0
Success: Shared Secrets Matches! 18.0
```

RESULT:

Thus the *Diffie-Hellman key exchange algorithm* has been implemented using Java Program and the output has been verified successfully.