**Laboratory Practice II Manual**

**Artificial IntelligenceGroup A**

**1. Implement depth first search algorithm and Breadth FirstSearch algorithm, Use an undirected graph and develop arecursive algorithm for searching all the vertices of a graph or tree data structure.**

BFS is one of the traversing algorithms used in graphs. This algorithm is implementedusing a queue data structure. In this algorithm, the main focus is on the vertices of the graph. Select a starting node or vertex at first, mark the starting node or vertex as visited and store it in a queue. Then visit the vertices or nodes which are adjacentto the starting node, mark them as visited and store these vertices or nodes in a queue.Repeat this process until all the nodes or vertices are completely visited.

**Advantages of BFS**

1. It can be useful in order to find whether the graph has connectedcomponents or not.
2. It always finds or returns the shortest path if there is more than one pathbetween two vertices.

**Disadvantages of BFS**

1. The execution time of this algorithm is very slow because the timecomplexity of this algorithm is exponential.
2. This algorithm is not useful when large graphs are used.

Explanation:

1. Create a graph.
2. Initialize a starting node.
3. Send the graph and initial node as parameters to the bfs function.
4. Mark the initial node as visited and push it into the queue.
5. Explore the initial node and add its neighbours to the queue and removethe initial node from the queue.
6. Check if the neighbours node of a neighbouring node is already visited.
7. If not, visit the neighbouring node neighbours and mark them as visited.
8. Repeat this process until all the nodes in a graph are visited and the queuebecomes empty.

Output:

```
['A', 'B', 'C', 'E', 'D', 'F', 'G']
```

**Implementation of BFS in Python ( Breadth First Search )**

**Source Code I: BFS in Python**

```
graph = {'A': ['B', 'C', 'E'],
'B': ['A','D', 'E'],
'C': ['A', 'F', 'G'],
'D': ['B'],
'E': ['A', 'B','D'],
'F': ['C'],
'G': ['C']}
def bfs(graph, initial):
visited = []
queue = [initial]
while queue:
node  =  queue.pop(0)
if node not in visited:
visited.append(node)
neighbours = graph[node]
for neighbour in neighbours:
queue.append(neighbour)
return visited
print(bfs(graph,'A'))
```

**Source Code II # BFS algorithm in Python**

```
import collections

# BFS  algorithm
def bfs(graph, root):

    visited, queue = set(), collections.deque([root])
    visited.add(root)

    while queue:

        # Dequeue a vertex from queue
```

```
        vertex = queue.popleft()
        print(str(vertex) + " ", end="")

        # If not visited, mark it as visited, and
        # enqueue it
        for neighbour in graph[vertex]:
            if neighbour not in visited:
                visited.add(neighbour)
                queue.append(neighbour)

if__name__== '__main__':
    graph = {0: [1, 2], 1: [2], 2: [3], 3: [1, 2]}
    print
```

**Depth First Search (DFS)**

Depth first Search or Depth first traversal is a recursive algorithm for searching all the vertices of a graph or tree data structure. Traversal means visiting all the nodes of a graph.

### Depth First Search Algorithm

A standard DFS implementation puts each vertex of the graph into one of two categories:

1. Visited
2. Not Visited

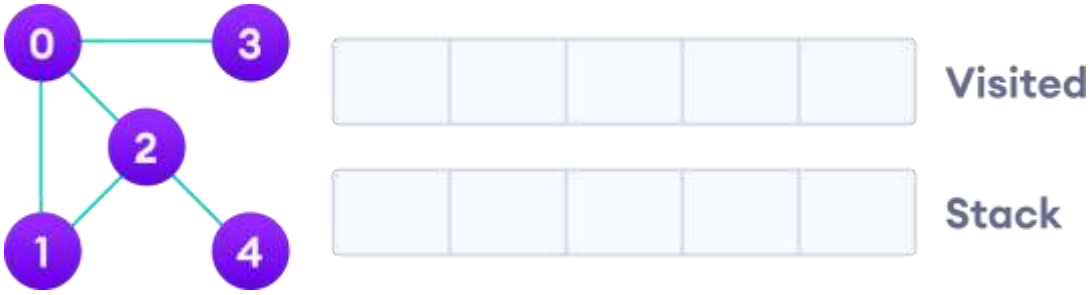
The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

The DFS algorithm works as follows:

1. Start by putting any one of the graph's vertices on top of a stack.
2. Take the top item of the stack and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of the stack.
4. Keep repeating steps 2 and 3 until the stack is empty.

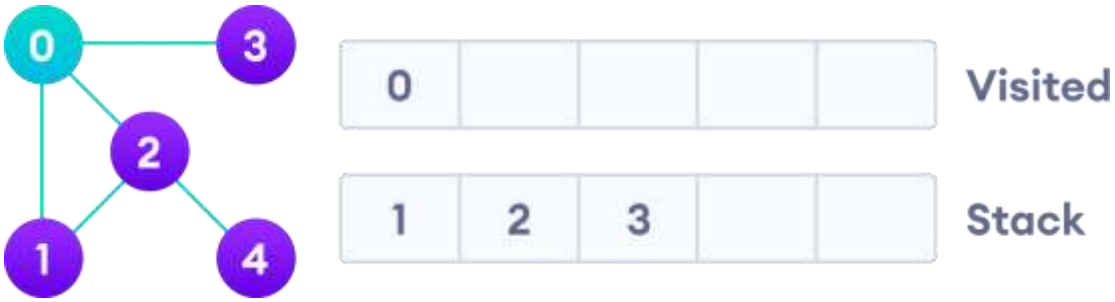
### Depth First Search Example

Let's see how the Depth First Search algorithm works with an example. We use an undirected graph with 5 vertices.



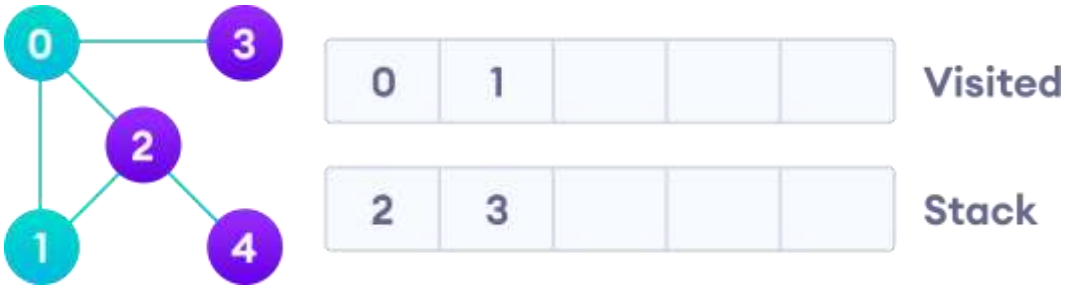

Undirected graph with 5 vertices

We start from vertex 0, the DFS algorithm starts by putting it in the Visited list and putting all its adjacent vertices in the stack.



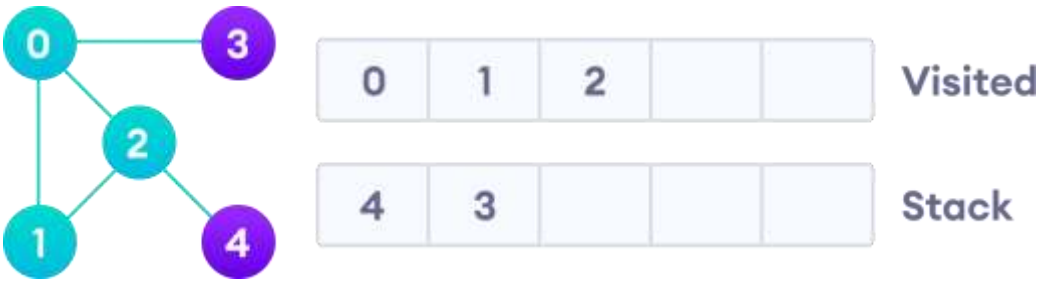

Visit the element and put it in the visited list

Next, we visit the element at the top of stack i.e. 1 and go to its adjacent nodes. Since 0 has already been visited, we visit 2 instead.

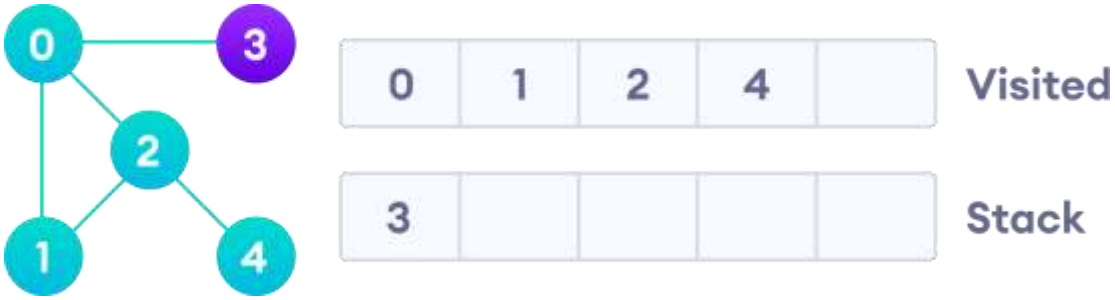



Visit the element at the top of stack

Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.

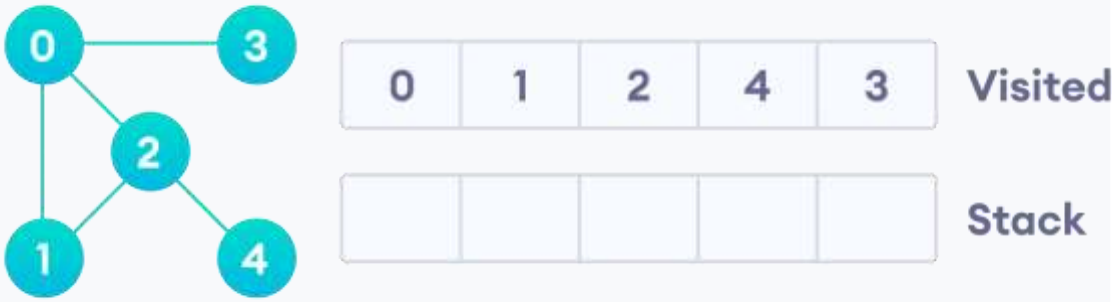



Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.




Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.

After we visit the last element 3, it doesn't have any unvisited adjacent nodes, so we have completed the Depth First Traversal of the graph.




After we visit the last element 3, it doesn't have any unvisited adjacent nodes, so we have completed the Depth First Traversal of the graph.

## Complexity of Depth First Search

The time complexity of the DFS algorithm is represented in the form of $O(V + E)$, where $V$ is the number of nodes and $E$ is the number of edges.

The space complexity of the algorithm is $O(V)$.

## Application of DFS Algorithm

1. For finding the path
2. To test if the graph is bipartite
3. For finding the strongly connected components of a graph
4. For detecting cycles in a graph

```
// DFS algorithm in Java

import java.util.*;

class Graph {
  private LinkedList<Integer> adjLists[];
  private boolean visited[];

  // Graph creation
  Graph(int vertices) {
    adjLists = new LinkedList[vertices];
    visited = new boolean[vertices];

    for (int i = 0; i < vertices; i++)
```

```
    adjLists[i] = new LinkedList<Integer>();
}

// Add edges
void addEdge(int src, int dest) {
  adjLists[src].add(dest);
}

// DFS algorithm
void DFS(int vertex) {
  visited[vertex] = true;
  System.out.print(vertex + " ");

  Iterator<Integer> ite = adjLists[vertex].listIterator();
  while (ite.hasNext()) {
    int adj = ite.next();
    if (!visited[adj])
    DFS(adj);
  }
}

public static void main(String args[]) {
  Graph g = new Graph(4);

  g.addEdge(0, 1);
  g.addEdge(0, 2);
  g.addEdge(1, 2);
  g.addEdge(2, 3);

  System.out.println("Following is Depth First Traversal");

  g.DFS(2);
```

```
  }
}
```

**# DFS algorithm in Python**

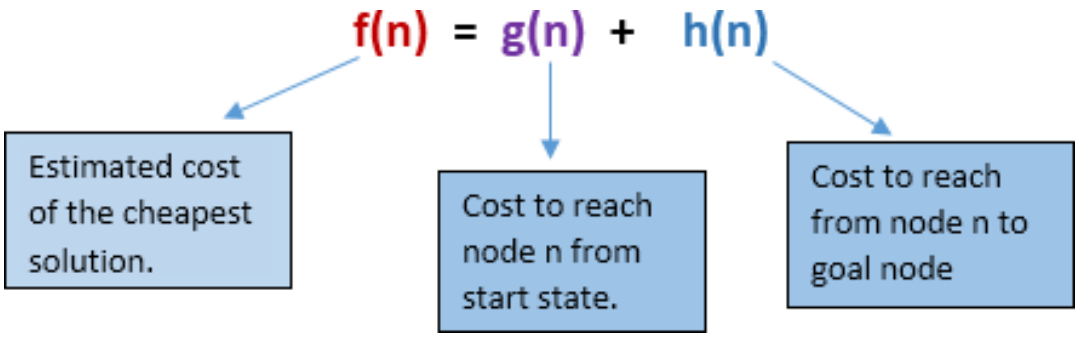
```
# DFS algorithm
def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)

    print(start)

    for next in graph[start] - visited:
        dfs(graph, next, visited)
    return visited


graph = {'0': set(['1', '2']),
         '1': set(['0', '3', '4']),
         '2': set(['0']),
         '3': set(['1']),
         '4': set(['2', '3'])}

dfs(graph, '0')
```

**2.Implement A star Algorithm for any game searchproblem.**

**A* Search**

A* search is the most commonly known form of best-first search. Ituses heuristic function h(n), and cost to reach the node n from the start state g(n). It has combined features of UCS and greedy best- first search, by which it solve the problem efficiently. A* search algorithm finds the shortest path through the search space using the heuristic function. This search algorithm expands less search tree and provides optimal result faster. A* algorithm is similar to UCS except that it uses g(n)+h(n) instead of g(n).

In A* search algorithm, we use search heuristic as well as the cost to reach the node. Hence we can combine both costs as following, and this sum is called as a **fitness number**.




**Algorithm of A* search:**

**Step1:** Place the starting node in the OPEN list.

**Step 2:** Check if the OPEN list is empty or not, if the list is empty then return failure and stops.

**Step 3:** Select the node from the OPEN list which has the smallest value of evaluation function (g+h), if node n is goal node then returnsuccess and stop, otherwise

**Step 4:** Expand node n and generate all of its successors, and put n into the closed list. For each successor n', check whether n' is already in the OPEN or CLOSED list, if not then compute evaluationfunction for n' and place into Open list.

**Step 5:** Else if node n' is already in OPEN and CLOSED, then it should be attached to the back pointer which reflects the lowest g(n')value.

**Step 6:** Return to **Step 2**.

**Advantages:**

1. A* search algorithm is the best algorithm than other searchalgorithms.
2. A* search algorithm is optimal and complete.
3. This algorithm can solve very complex problems.

**Disadvantages:**

1. It does not always produce the shortest path as it mostlybased on heuristics and approximation.
2. A* search algorithm has some complexity issues.
3. The main drawback of A* is memory requirement as it keepsall generated nodes in the memory, so it is not practical for various large-scale problems.

**#AIM: Implement A* search.**

```
dict_hn={'Arad':336,'Bucharest':0,'Craiova':160,'Drobeta':242,'Eforie':161,
    'Fagaras':176,'Giurgiu':77,'Hirsova':151,'Iasi':226,'Lugoj':244,
```

```
        'Mehadia':241,'Neamt':234,'Oradea':380,'Pitesti':100,'Rimnicu':193,
        'Sibiu':253,'Timisoara':329,'Urziceni':80,'Vaslui':199,'Zerind':374}

dict_gn=dict(
Arad=dict(Zerind=75,Timisoara=118,Sibiu=140),
Bucharest=dict(Urziceni=85,Giurgiu=90,Pitesti=101,Fagaras=211),
Craiova=dict(Drobeta=120,Pitesti=138,Rimnicu=146),
Drobeta=dict(Mehadia=75,Craiova=120),
Eforie=dict(Hirsova=86),
Fagaras=dict(Sibiu=99,Bucharest=211),
Giurgiu=dict(Bucharest=90),
Hirsova=dict(Eforie=86,Urziceni=98),
Iasi=dict(Neamt=87,Vaslui=92),
Lugoj=dict(Mehadia=70,Timisoara=111),
Mehadia=dict(Lugoj=70,Drobeta=75),
Neamt=dict(Iasi=87),
Oradea=dict(Zerind=71,Sibiu=151),
Pitesti=dict(Rimnicu=97,Bucharest=101,Craiova=138),
Rimnicu=dict(Sibiu=80,Pitesti=97,Craiova=146),
Sibiu=dict(Rimnicu=80,Fagaras=99,Arad=140,Oradea=151),
Timisoara=dict(Lugoj=111,Arad=118),
Urziceni=dict(Bucharest=85,Hirsova=98,Vaslui=142),
Vaslui=dict(Iasi=92,Urziceni=142),
Zerind=dict(Oradea=71,Arad=75)
)
import queue as Q
#from RMP import dict_gn
#from RMP import dict_hn

start='Arad'
goal='Bucharest'
result=''

def get_fn(citystr):
    cities=citystr.split(" , ")
```

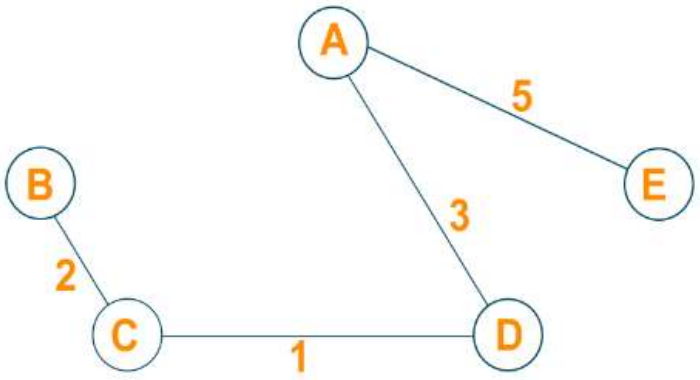
```
    hn=gn=0
    for ctr in range(0, len(cities)-1):
        gn=gn+dict_gn[cities[ctr]][cities[ctr+1]]
    hn=dict_hn[cities[len(cities)-1]]
    return(hn+gn)

def expand(cityq):
    global result
    tot, citystr, thiscity=cityq.get()
    if thiscity==goal:
        result=citystr+" : : "+str(tot)
        return
    for cty in dict_gn[thiscity]:
        cityq.put((get_fn(citystr+" , "+cty), citystr+" , "+cty, cty))
    expand(cityq)

def main():
    cityq=Q.PriorityQueue()
    thiscity=start
    cityq.put((get_fn(start),start,thiscity))
    expand(cityq)
    print("The A* path with the total is: ")
    print(result)

main()

"""
OUTPUT:
The A* path with the total is:
Arad , Sibiu , Rimnicu , Pitesti , Bucharest : : 418
"""
```

## 3. Implement Greedy search algorithm for any ofthe following application:

**Prim's Minimal Spanning Tree Algorithm**

**Prim's Algorithm | Minimum Spanning Tree (Python Code)**

# Prim's Algorithm
## (Minimum Spanning Tree)




We will study what is the minimum spanning tree and how to convert a graph into a minimum spanning tree using Prim's Algorithm. We will learn the algorithm and python code for prim's algorithm and an example for better understanding. Lastly, we will study the running time complexity and applications of prim's algorithm in real life.

### What is a Minimum Spanning Tree?

As we all know, the graph which does not have edges pointing to any direction in a graph is called an undirected graph and the graph always has a path from a vertex to any other vertex. A spanning tree is a subgraph of the undirected connected graph where it includes all the nodes of the graph with the minimum possible number of edges. Remember, the subgraph should contain each and every node of the original graph. If any node is missed out then it is not a spanning tree and also, the spanning tree doesn't contain cycles. If the graph has n number of nodes, then the total number of spanning trees created from a complete graph is equal to n^(n-2). In a spanning tree, the edges may or may not have weights associated with them. Therefore,the spanning tree in which the sum of edges is minimum as possible then that spanning tree is called the minimum spanning tree. One graph can have multiple spanning-tree but it can haveonly one unique minimum spanning tree. There are two different ways to find out the minimum spanning tree from the complete graph i.e Kruskal's algorithm and Prim's algorithm. Let us study prim's algorithm in detail below:

### What is Prim's Algorithm?

Prim's algorithm is a minimum spanning tree algorithm which helps to find out the edges of the graph to form the tree including every node with the minimum sum of weights to form the minimum spanning tree. Prim's algorithm starts with the single source node and later explore all the adjacent nodes of the source node with all the connecting edges. While we are exploringthe graphs, we will choose the edges with the minimum weight and those which cannot cause the cycles in the graph.

### Prim's Algorithm for Minimum Spanning Tree

Prim's algorithm basically follows the greedy algorithm approach to find the optimal solution.To find the minimum spanning tree using prim's algorithm, we will choose a source node andkeep adding the edges with the lowest weight.

The algorithm is as given below:

- Initialize the algorithm by choosing the source vertex
- Find the minimum weight edge connected to the source node and another node and add it to the tree
- Keep repeating this process until we find the minimum spanning tree

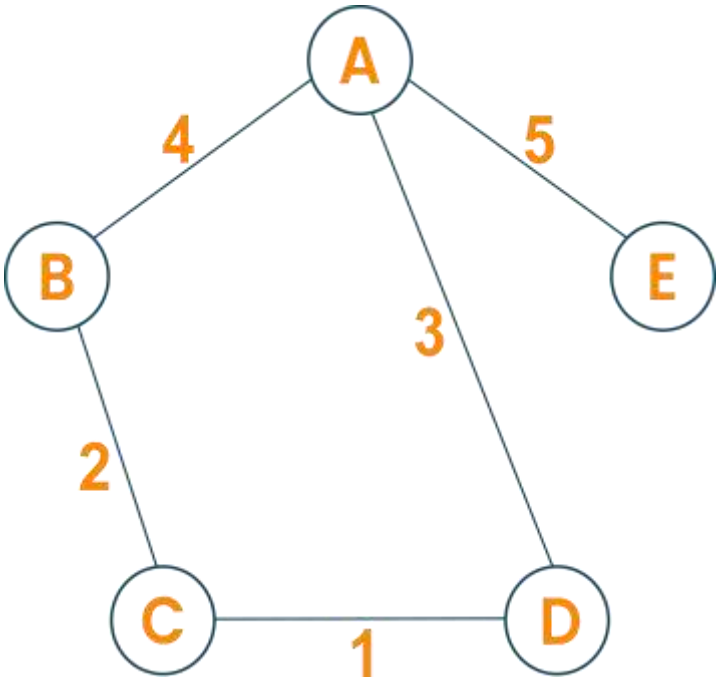
### Pseudocode

```
T = ∅;
M = { 1 };
while (M ≠ N)
   let (m, n) be the lowest cost edge such that m ∈ M and n ∈ N -
   M;T = T U {(m, n)}
   M = M U {n}
```

Here we create two sets of nodes i.e M and M-N. M set contains the list of nodes that have been visited and the M-N set contains the nodes that haven't been visited. Later, we will move each node from M to M-N after each step by connecting the least weight edge.

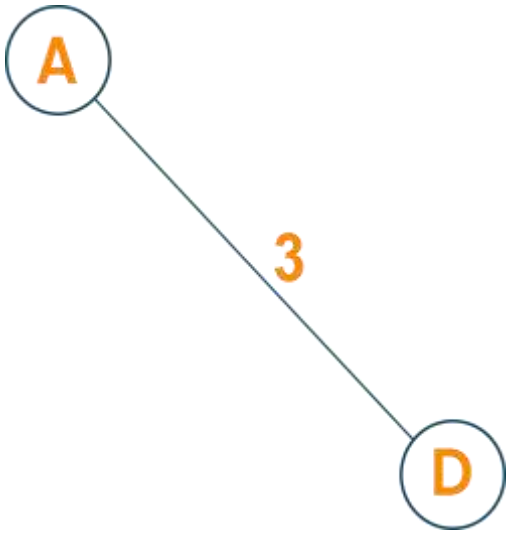
### Example

Let us consider the below-weighted graph

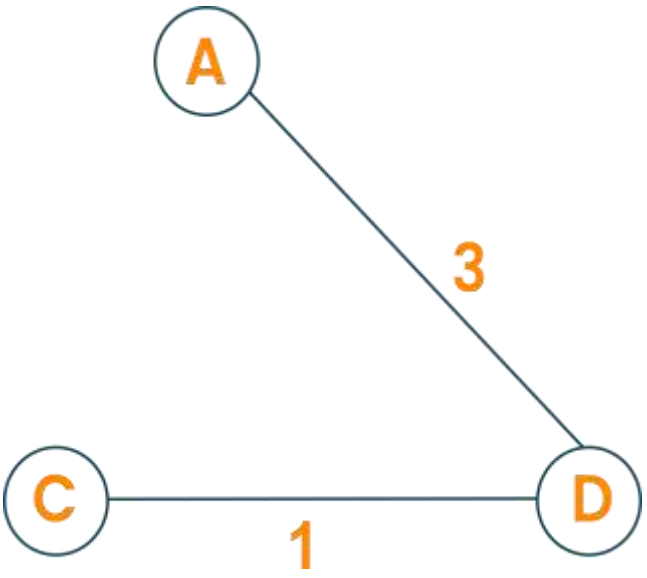


Later we will consider the source vertex to initialize the algorithm

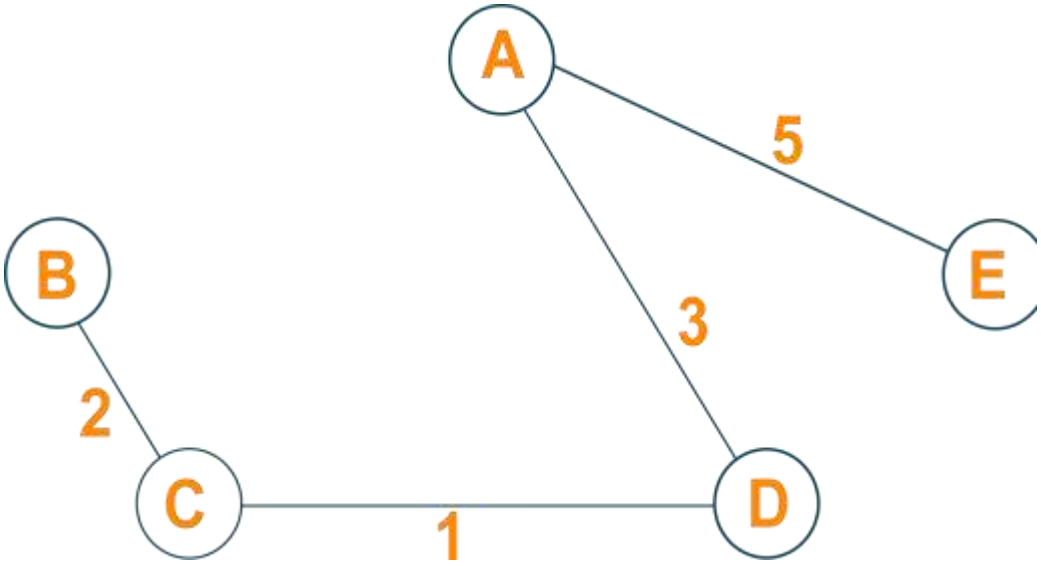

Now, we will choose the shortest weight edge from the source vertex and add it to finding the spanning tree.




Then, choose the next nearest node connected with the minimum edge and add it to the solution. If there are multiple choices then choose anyone.




Continue the steps until all nodes are included and we find the minimum spanning tree.




```
# Prim's Algorithm in Python

INF = 9999999
# number of vertices in graphN
= 5
#creating graph by adjacency matrix methodG =
[[0, 19, 5, 0, 0],
     [19, 0, 5, 9, 2],
     [5, 5, 0, 1, 6],
     [0, 9, 1, 0, 1],
     [0, 2, 6, 1, 0]]

selected_node = [0, 0, 0, 0, 0]

no_edge = 0

selected_node[0] = True

# printing for edge and weight
print("Edge : Weight\n") while
(no_edge < N - 1):

    minimum = INFa
    = 0
    b = 0
    for m in range(N):
          if selected_node[m]:
             for n in range(N):
              if ((not selected_node[n]) and G[m][n]):#
                 not in selected and there is an edge if
                 minimum > G[m][n]:
                    minimum = G[m][n]
```

```
                a = m
                b = n
    print(str(a) + "-" + str(b) + ":" + str(G[a][b]))
    selected_node[b] = True
    no_edge += 1
```

**Time Complexity:**

The running time for prim's algorithm is O(VlogV + ElogV) which is equal to O(ElogV) because every insertion of a node in the solution takes logarithmic time. Here, E is the numberof edges and V is the number of vertices/nodes. However, we can improve the running time complexity to O(E + logV) of prim's algorithm using Fibonacci Heaps.

**Applications**

- Prim's algorithm is used in network design
- It is used in network cycles and rail tracks connecting all the cities
- Prim's algorithm is used in laying cables of electrical wiring
- Prim's algorithm is used in irrigation channels and placing microwave towers
- It is used in cluster analysis
- Prim's algorithm is used in gaming development and cognitive science
- Pathfinding algorithms in artificial intelligence and traveling salesman problems make use of prim's algorithm.

**Conclusion**

As we studied, the minimum spanning tree has its own importance in the real world, it is important to learn the prim's algorithm which leads us to find the solution to many problems. When it comes to finding the minimum spanning tree for the dense graphs, prim's algorithm isthe first choice.

## Group-B

## 4. Implement a solution for a Constraint Satisfaction Problemusing Branch and Bound and Backtracking for n-queens problem or a graph coloring problem

### 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 ordiagnol.

**The branch and bound solution is somehow different, it generates a partial solution until it figures that there's no point going deeperas 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 theway we fill out our auxiliary

matrix (column by column). Hence, onlythe 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 checkfor 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 samevalue in backSlashDiagnol.

slashDiagnol(row)(col) = row + col

backSlashDiagnol(row)(col) = row - col + (N-1) { N = 8 }

{ we added (N-1) as we do not need negative values 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 locationfor queen **i** on row **j**, mark the row and diagnols; and recur for queen **i+1**.

```
#include<bits/stdc++.h>

using namespace std; int

board[8][8]

int n

// function to print solution

void printSolution(int board[n][n])

{

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

```
    {
        for (int j = 0; j < n; j++)
         { cout<<board[i][j]<<" ";
         }
        cout<<endl;
    }
}

//function to check if queen can
// be placed on board[row][col]

bool isPossible(int row, int col, int slashDiagnol[n][n],int
            backSlashDiagnol[n][n], bool rowLook[n],
            bool slashDiagnolLook[], bool backSlashDiagnolLook[])
{
    if (slashDiagnolLook(slashDiagnol[row][col] || backSlashDiagnol[row][col]
        || rowLook[row] )
    return false;

    return true;
}

//A recursive utility function to solve N Queen problembool
solveNQueensUtil(int board[n][n], int col,
    int slashDiagnol[n][n],int backSlashDiagnol[n][n],bool
    rowLook[n], bool slashDiagnolLook[],
    bool backSlashDiagnolLook[] )
{
    //base case: If all queens are placedif
    (col >= N)
        return true;
```

```
    //Consider this column and try placing
    // queen in all rows one by one
    for (int i = 0; i < n; i++)
    {

        if ( isPossible(i, col, slashDiagnol, backSlashDiagnol,
             rowLook, slashDiagnolLook, backSlashDiagnolLook)
             )
        {
            board[i][col] = 1;
            rowLookup[i] = true;
            slashDiagnolLook[slashDiagnol[i][col]] = true;
            backSlashDiagnolLook[backSlashDiagnol[i][col] = true;

            //recur to place rest of the queens
            if ( solveNQueensUtil(board, col + 1, slashCode, backslashCode,
                    rowLookup, slashCodeLookup, backslashCodeLookup) )
                return true;

            // placing queen in board[i][col]
            // dosen't yield a solution, backtrack

            board[i][col] = 0; rowLook[i]
            = false;
            slashDiagolLook[slashDiagnol[i][col]] = false;
            backSlashDiagnolLook[backSlashDiagnol[i][col]] = false;
        }
    }

    //If queen can not be place in any row in
   //this colum col then return false
    return false;
}

/* This function solves the N Queen problem using Branch and Bound. It mainly uses
solveNQueensUtil() to solve the problem. It returns false if queens cannot be placed, otherwise
```

```
return true and prints placement of queens in the form of 1s. Please note that there may be
more than one solutions, this function prints one of the feasible solutions.*/
bool solveNQueens(n)
{

   memset(board, 0, sizeof(board));

   // helper matrices
   int slashDiagnol[n][n];
   int backSlashDiagnol[n][n];

   // arrays to tell us which rows are occupied
   bool rowLook[n] = {false};

   //keep two arrays to tell us which diagonals are occupiedbool
   slashDiagnolLook[2*n-1] = {false};
   bool backSlashDiagnolLook[2*n-1] = {false};

   // initialize helper matrices
   for (int r = 0; r < n; r++)
      for (int c = 0; c < n; c++)
      {
         slashDiagnol[r][c] = r+c;
         backSlashDiagnol[r][c] = (r+c-7);
       }

   if (solveNQueensUtil(board, 0, slashDiagnol, backSlashDiagnol,rowLook,
     slashDiagnolLook, backSlashDiagnolLook) == false)
   {
      cout<<"No solution"<<endl;
      return false;
   }

   // solution found
   printSolution(board);
   return true;
}
```

```
// main function
int main()
{
    cin>>n; // can take any size from 0 to 8
    solveNQueens(n);
    return 0;
}
```

**Output- for (n = 8)**

```
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
```

**Graph coloring problem's solution using backtracking algorithmGraph coloring**

The **graph coloring problem** is to discover whether the nodes of the graph G canbe covered in such a way, that no two adjacent nodes have the same color yet only m colors are used. This graph coloring problem is also known as M- colorability decision problem.

The M – colorability optimization problem deals with the smallest integer m for which the graph G can be colored. The integer is known as a chromatic number of the graph.

Here, it can also be noticed that if d is the degree of the given graph, then it can be colored with d+ 1 color.

A graph is also known to be planar if and only if it can be drawn in a planar in such a way that no two edges cross each other. A special case is the 4 - colors problem for planar graphs. The problem is to color the region in a map in such away that no two adjacent regions have the same color. Yet only four colors are needed. This is a problem for which graphs are very useful because a map can beeasily transformed into a graph. Each region of the map becomes the node, and iftwo regions are adjacent, they are joined by an edge.

**Graph coloring problem** can also be solved using a state space tree, whereby applying a backtracking method required results are obtained.

For solving the **graph coloring problem**, we suppose that the graph is represented by its adjacency matrix G[ 1:n, 1:n], where, G[ i, j]= 1 if (i, j) is an edge of G, and G[i, j] = 0 otherwise.

The colors are represented by the integers 1, 2, ..., m and the solutions are given by the n-tuple (x1, x2, x3, ..., xn), where x1 is the color of node i.

**Algorithm for finding the m - colorings of a graph**

```
1.      Algorithm mcoloring ( k )
2.      // this algorithm is formed using the recursive backtracking
3.      // schema. The graph is represented by its Boolean adjacency
4.      // matrix G [1: n, 1: n]. All assignments of 1, 2, …, m to the
5.      // vertices of the graph such that adjacent vertices are
6.      // assigned distinct are printed. K is the index
7.      // of the next vertex to color.
```

```
8.        {
9.        Repeat
10.       {
11.       // generate all legal assignments for x[k],
12.       Next value (k); // assign to x[k] a legal color.
13.        If ( x[k] = 0 ) then return; // no new color possible
14.        If (k = n) then // at most m colors have been used to color the n
vertices.
15.         Write (x[1 : n ]);
16.        Else mcoloring (k + 1);
17.       }
18.       Until (false);
19.       }
```

This algorithm uses the recursive backtracking schema. In this algorithm colorsto be assigned are to determine from the range (0, m), i.e., m colors are available.

The total time required by the above algorithm is **O (nm^n)**.

**Implementation of Backtracking solution**

C/C++

```
/* C/C++ program to solve N Queen Problem usingbacktracking */
#define N 4
#include <stdbool.h>
#include <stdio.h>

/* A utility function to print solution */void
printSolution(int board[N][N])
{
    for (int i = 0; i < N; i++) { for (int
        j = 0; j < N; j++)
            printf(" %d ", board[i][j]);
        printf("\n");
    }
}
/* A utility function to check if a queen canbe placed on board[row][col]. Note that this function is called when "col" queens arealready placed in columns from0 to col -1. So we need to check only left side for attacking queens */

bool isSafe(int board[N][N], int row, int col)
{
    int i, j;

    /* Check this row on left side */for (i =
    0; i < col; i++)
        if (board[row][i])
            return false;

    /* Check upper diagonal on left side */
    for (i = row, j = col; i >= 0 && j >= 0; i--, j--)if
        (board[i][j])
```

```
            return false;

    /* Check lower diagonal on left side */
    for (i = row, j = col; j >= 0 && i < N; i++, j--)if
        (board[i][j])
            return false;


    return true;
}

/* A recursive utility function to solve NQueen
   problem */
bool solveNQUtil(int board[N][N], int col)
{
    /* base case: If all queens are placedthen
      return true */
    if (col >= N)
        return true;

    /* Consider this column and try placingthis queen
       in all rows one by one */
    for (int i = 0; i < N; i++) {
        /* Check if the queen can be placed on
          board[i][col] */
        if (isSafe(board, i, col)) {
            /* Place this queen in board[i][col] */
            board[i][col] = 1;

            /* recur to place rest of the queens */if
            (solveNQUtil(board, col + 1))
                return true;


            /* If placing queen in board[i][col] doesn't lead
               to a solution, then remove queen from
               board[i][col] */
            board[i][col] = 0; // BACKTRACK
        }
    }


    /* If the queen cannot be placed in any row inthis column
        col then return false */
    return false;
}

/* This function solves the N Queen problem using   Backtracking. It mainly uses solveNQUtil() to solve the problem. It returns false if queens cannot be placed, otherwise, return true and prints placement of queens in the form of 1s. Please note that there may be more than one solutions, this function prints oneof the feasible solutions.*/
bool solveNQ()
{
    int board[N][N] = { { 0, 0, 0, 0 },
                        { 0, 0, 0, 0 },
                        { 0, 0, 0, 0 },
                        { 0, 0, 0, 0 } };
```

```
    if (solveNQUtil(board, 0) == false) {
        printf("Solution does not exist"); return false;
    }

    printSolution(board);return true;
}

// driver program to test above functionint main()
{
    solveNQ();return
    0;
}
```

Output:The 1 values indicate placements of queens

```
 0  0  1  0
 1  0  0  0
 0  0  0  1
 0  1  0  0
```

**5. Develop an elementary chatbot for any suitable customerinteraction application.**

**What is a chatbot?**

A chatbot is a computer program designed to have a conversation with human beingsover 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 **ArtificialIntelligence**, 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 ofadvantages, one of which is that, because Chatbots communicate using a natural language, users don't need to learn yet another new website interface, to getcomfortable 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, 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 canalso save money for companies and are easy to set up.

Chatbots are relatively new and most companies haven't implemented them yet, it'sonly 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 24apps a month, wherein 80% of the time would be in just 5 apps. This means you canhardly shoot ahead with an app, but you still have high chances to integrate your chatbot with one of these platforms.

**Now lets 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 availableoperative. Bots are replacing live chat and other forms of contact such as emails andphone calls.

Since chat bots are basically virtual robots they never get tired and continue to obeyyour 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 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 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. 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:

Lets 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. 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.

**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. 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 readyto 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 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.

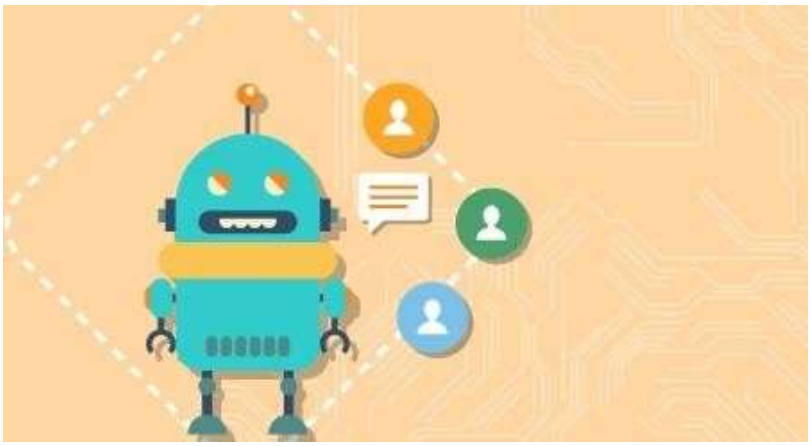
**e. Customer Service**

- 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.
- 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 andunderstanding 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 areasof 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 customersquickly. 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 works to accept table reservations, take special requests and go take 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 canprovide 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 toproductivity 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. Throughthese 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 successof 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 in Fashion Industry**

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 in Finance Industry. Chatbots can be programmed to assists the customers as Financial Advisor, Expense Saving Bot, Banking Bots, Taxbots, 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 andexpenses on particular things. Further about Finance and Chatbots have beendiscussed 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 personalised health and fitness tips. Tech based fitness companies can have a huge opportunity by developing their own Chatbots offering huge customer base with personalised services. Engage with your fans like never before with news, highlights,game-day info, roster and more.

Chatbots and Service Industry together have a wide range of opportunities and smallto 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 thelatest news on the go.

**Chatbot in Celebrity:**

With a chatbot you can now have one-on-one conversation with millions of fans.

**Chatbot in MarketingSMS**

**Marketing**

- Why promote just a coupon code that the customer does not know how touse?

- 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 wayto 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 brandon social media?
- Addressing queries instantly and effectively can convert even an angrycustomer into a loyal fan.
- Leverage a chatbot as your first response strategy and comfort that customer.

*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 applya 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 theproject, 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.

**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

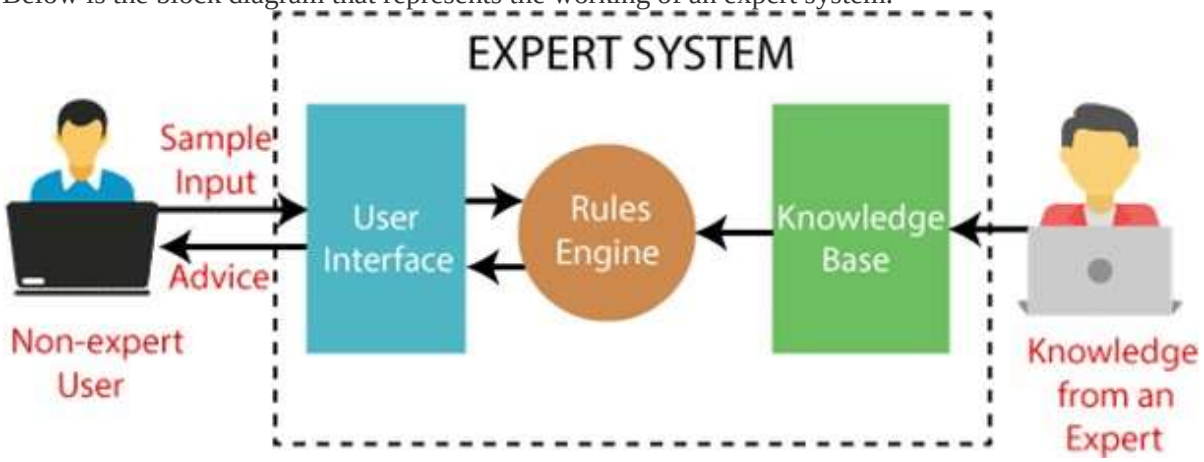
## What is an Expert System?

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 wasthe 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 itsknowledge 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 designedto 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 cancerat early stages.

**Characteristics of Expert System**

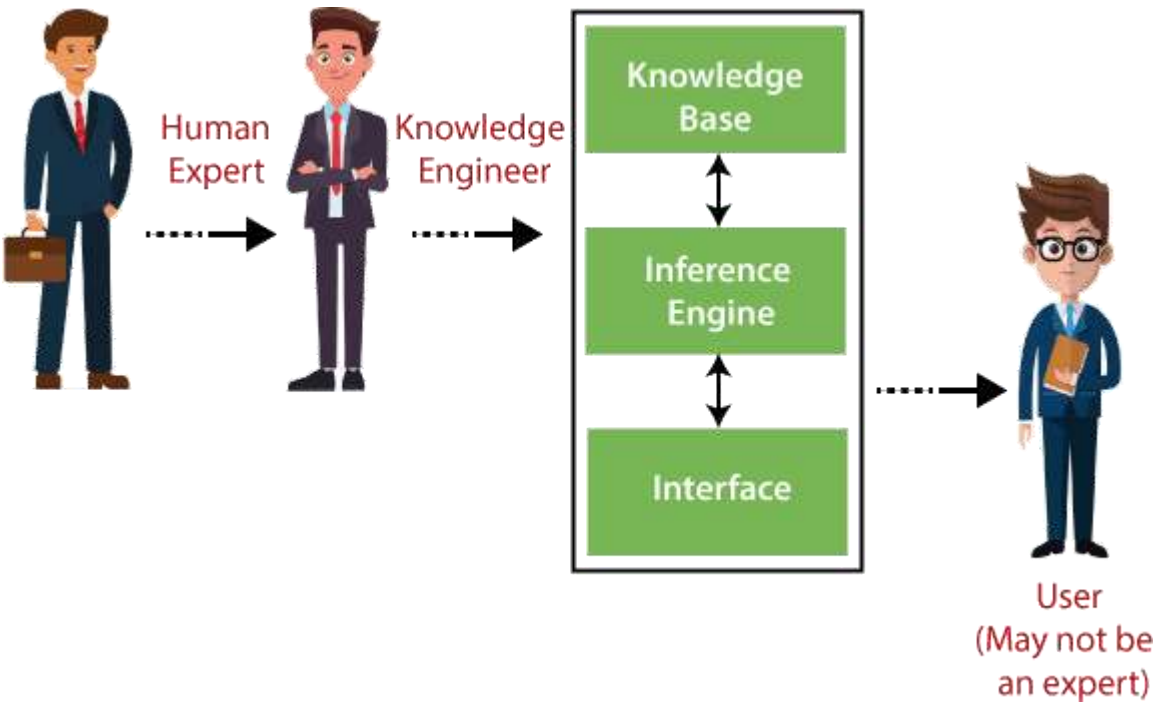
- **High Performance:** The expert system provides high performance for solving any type of complex problem of a specific domain with high efficiency and accuracy.

- **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

- These systems are highly reproducible.
- They can be used for risky places where the human presence is not safe.
- Error possibilities are less if the KB contains correct knowledge.
- The performance of these systems remains steady as it is not affected by emotions, tension, or fatigue.
- They provide a very high speed to respond to a particular query.

## 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.