



Empowerment Through Quality Technical Education
AJEEENKYA DY PATIL SCHOOL OF ENGINEERING
Dr. D. Y. Patil Knowledge City, Charholi (Bk), Lohegaon, Pune – 412 105
Website: <https://dypsoe.in/>

LAB MANUAL

SOFTWARE LABORATORY-I Artificial Intelligence (317523)

TE (AI&DS) 2019 COURSE

Course Coordinator

Prof. Rupali N. Wagh

DEPARTMENT OF
ARTIFICIAL INTELLIGENCE & DATA SCIENCE

Department of Artificial Intelligence & Data Science

Vision:

Imparting quality education in the field of Artificial Intelligence and Data Science

Mission:

- To include the culture of R and D to meet the future challenges in AI and DS.
- To develop technical skills among students for building intelligent systems to solve problems.
- To develop entrepreneurship skills in various areas among the students.
- To include moral, social and ethical values to make students best citizens of country.

Program Educational Outcomes:

1. To prepare globally competent graduates having strong fundamentals, domain knowledge, updated with modern technology to provide the effective solutions for engineering problems.
2. To prepare the graduates to work as a committed professional with strong professional ethics and values, sense of responsibilities, understanding of legal, safety, health, societal, cultural and environmental issues.
3. To prepare committed and motivated graduates with research attitude, lifelong learning, investigative approach, and multidisciplinary thinking.
4. To prepare the graduates with strong managerial and communication skills to work effectively as individuals as well as in teams.

Program Specific Outcomes:

- 1. Professional Skills-** The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, networking, artificial intelligence and data science for efficient design of computer-based systems of varying complexities.
- 2. Problem-Solving Skills-** The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.
- 3. Successful Career and Entrepreneurship-** The ability to employ modern computer languages, environments and platforms in creating innovative career paths to be an entrepreneur and to have a zest for higher studies.

Table of Contents

Contents

1. Guidelines to manual usage.....	5
2. Laboratory Objective:.....	9
3. Laboratory Equipment/Software:.....	9
4. Laboratory Experiment list	10
Implement Greedy search algorithm for any ofthe following application:	10
4.1. Experiment No. 1.....	11
4.2. Experiment No. 2.....	18
A* Search.....	18
Disadvantages:	22
4.3. Experiment No. 3.....	24
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. .	24
Applying the branch and bound approach:	25
The indexes on these arrays would help us know which queen we are analysing.....	25
4.4. Experiment No. 4.....	30
Working of Alpha-Beta Pruning:.....	31
Move Ordering in Alpha-Beta pruning:.....	34
Rules to find good ordering:	34
Algorithm:.....	35
4.5. Experiment No. 5	37
5.1. Experiment No. 6.....	46
Aim: Implement Greedy search algorithm for any ofthe following application:	46
Theory: What is a Minimum Spanning Tree?	46
What is Prim's Algorithm?	46
Prim's Algorithm for Minimum Spanning Tree.....	46

Pseudocode.....	47
Example	47
Time Complexity:.....	51
1. How does Prim's Algorithm Work?	54
2. What Is a Minimum Spanning Tree?	54
3. What Is Prim's Minimum Spanning Tree Algorithm?.....	54
4. How to Implement the Prim's Algorithm	54
6. Appendix	55
Python Installation on Windows	55
Step 1: Select Python Version	55
Step 2: Download Python Executable Installer.....	55
Step 3: Run Executable Installer.....	56
Step 4: Add Python to Path (Optional)	57
Step 5: Verify Python Was Installed on Windows	58
Step 7: Install virtualenv (Optional)	59

1. Guidelines to manual usage

This manual assumes that the facilitators are aware of collaborative learning methodologies.

This manual will provide a tool to facilitate the session on Digital Communication modules in collaborative learning environment.

The facilitator is expected to refer this manual before the session.

Icon of Graduate Attributes

K Applying Knowledge	A Problem Analysis	D Design & Development	I Investigation of problems
M Modern Tool Usage	E Engineer & Society	E Environment Sustainability	T Ethics
T Individual & Team work	O Communication	M Project Management & Finance	I Life-Long Learning

Disk Approach- Digital Blooms Taxonomy



- 1: Remembering / Knowledge
- 2: Comprehension / Understanding
- 3: Applying
- 4: Analyzing
- 5: Evaluating
- 6: Creating / Design

Program Outcomes:

1. **Engineering knowledge:** An ability to apply knowledge of mathematics, including discrete mathematics, statistics, science, computer science and engineering fundamentals to model the software application.
2. **Problem analysis:** An ability to design and conduct an experiment as well as interpret data, analyze complex algorithms, to produce meaningful conclusions and recommendations.
3. **Design/development of solutions:** An ability to design and development of software system, component, or process to meet desired needs, within realistic constraints such as economic, environmental, social, political, health & safety, manufacturability, and sustainability.
4. **Conduct investigations of complex problems:** An ability to use research based knowledge including analysis, design and development of algorithms for the solution of complex problems interpretation of data and synthesis of information to provide valid conclusion.
5. **Modern tool usage:** An ability to adapt current technologies and use modern IT tools, to design, formulate, implement and evaluate computer based system, process, by considering the computing needs, limits and constraints.
6. **The engineer and society:** An ability of reasoning about the contextual knowledge of the societal, health, safety, legal and cultural issues, consequent responsibilities relevant to IT practices.
7. **Environment and sustainability:** An ability to understand the impact of engineering solutions in a societal context and demonstrate knowledge of and the need for sustainable development.
8. **Ethics:** An ability to understand and commit to professional ethics and responsibilities and norms of IT practice.
9. **Individual and team work :** An ability to apply managerial skills by working effectively as an individual, as a member of a team, or as a leader of a team in multidisciplinary projects.
10. **Communication:** An ability to communicate effectively technical information in speech, presentation, and in written form
11. **Project management and finance:** An ability to apply the knowledge of Information Technology and management principles and techniques to estimate time and resources needed to complete engineering project.
12. **Life-long learning:** An ability to recognize the need for, and have the ability to engage in independent and life-long learning.

Course Name: Software Laboratory I- Artificial Intelligence**Course Code: (317523)****Course Outcomes**

1. CO4: Design a system using different informed search/ uninformed search or heuristics approaches.
2. CO5: Apply basic principles of AI in solutions that require problem solving, inference, perception, knowledge representation, and learning.
3. CO6: Design and develop an interactive AI applications

CO to PO Mapping:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO4	1	2	2	1	2	-	-	-	1	1	-	-
CO5	3	2	2	1	-	-	-	-	1	-	-	-
CO6	2	2	2	2	2				1	-	-	1

CO to PSO Mapping:

	PSO1	PSO2	PSO3
CO4	2	1	-
CO5	2	2	-
CO6	2	2	1

2. Laboratory Objective:

To learn and apply various strategies for AI

To formalize and implement constraints in search problems

3. Laboratory Equipment/Software:

Open Source python Programming tool /java and Ubuntu.

4. Laboratory Experiment list

Sr. No	Title
	Prerequisite practical assignments or installation (if any)
1	Fundamentals of Data Structures (210242), Data Structures and Algorithms (210253)
	List of Assignments
1	Implement depth first search algorithm and Breadth First Search algorithm, Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure
2	Implement A* star Algorithm for any game search problem.
3	Implement Alpha-beta Tree search for any game search problem
4	Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem
5	Implement Greedy search algorithm for any of the following application: Prim's Algorithm Minimum Spanning Tree (Python Code)
6	Develop an elementary chatbot for any suitable customer interaction application.
	Content Beyond Syllabus
1	Chatbot Application in Different Domain
2	Miniproject: Implementation of various Expert System

4.1. Experiment No. 1

Aim:

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

Prerequisite: Basic knowledge of Arrays, Lists, Stack, Queue, Graph, Tree etc.

Objective:

In this experiment, we will be able to do the following:

- To understand Uninformed Search Strategies and to make use of Graph and Tree Data Structure for implementation of Uninformed Search strategies.
- Study the various Graph traversal algorithms and the difference between them.Understand the BFS Graph traversal using queues.
- Demonstrate knowledge of time complexity and space complexity of performing a BFS on a given graph.

Software and Hardware Requirement: Open Source python Programming tool /java and Ubuntu.

Theory:

Uninformed (Blind search) search

- Uninformed search is a class of general-purpose search algorithms which operates in brute force-way.
- It examines each node of the tree until it achieves the goal node.
- Uninformed search algorithms do not have additional information about state or search space other than how to traverse the tree and how to identify leaf and goal nodes, so it is also called blind search
- Eg. DFS, BFS Search algorithm Graphs Definition A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links.

The interconnected objects are represented by points termed as vertices, and the links that connect the vertices are called edges.

Types of Graphs

- Undirected Graph:- Directions are not given to edges

- Directed Graph:- Directions are given to edges with

Theory of Graph Traversal Techniques

In computer science, graph traversal (also known as graph search) refers to the process of visiting (checking and/or updating) each vertex in a graph. Such traversals are classified by the order in which the vertices are visited. Tree traversal is a special case of graph traversal. Techniques of Graph Traversal

- **DFS** - A depth-first search (DFS) is an algorithm for traversing a finite graph. DFS visits the child vertices before visiting the sibling vertices; that is, it traverses the depth of any particular path before exploring its breadth. A stack (often the program's call stack via recursion) is generally used when implementing the algorithm.
- **BFS** - A breadth-first search (BFS) is another technique for traversing a finite graph. BFS visits the neighbor vertices before visiting the child vertices, and a queue is used in the search process. This algorithm is often used to find the shortest path from one vertex to another

DFS

- Depth-first search is a recursive algorithm for traversing a tree or graph data structure.
- It is called the depth-first search because it starts from the root node and follows each path to its greatest depth node before moving to the next path.
- DFS uses a stack data structure for its implementation.
- The process of the DFS algorithm is similar to the BFS algorithm.
- Depth First Search algorithm is a recursive algorithm that uses the idea of backtracking.

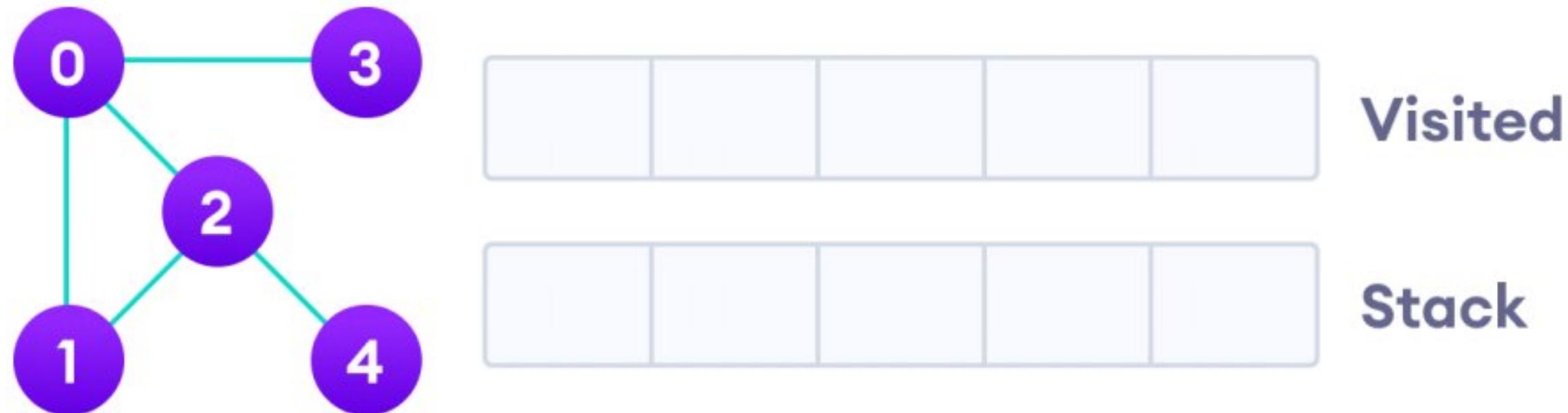
Depth First Search (DFS) Algorithm

Depth first search (DFS) algorithm starts with the initial node of the graph G, and then goes deeper and deeper until we find the goal node or the node which has no children. The algorithm then backtracks from the dead end towards the most recent node that is yet to be completely explored. The data structure which is being used in DFS is stack.

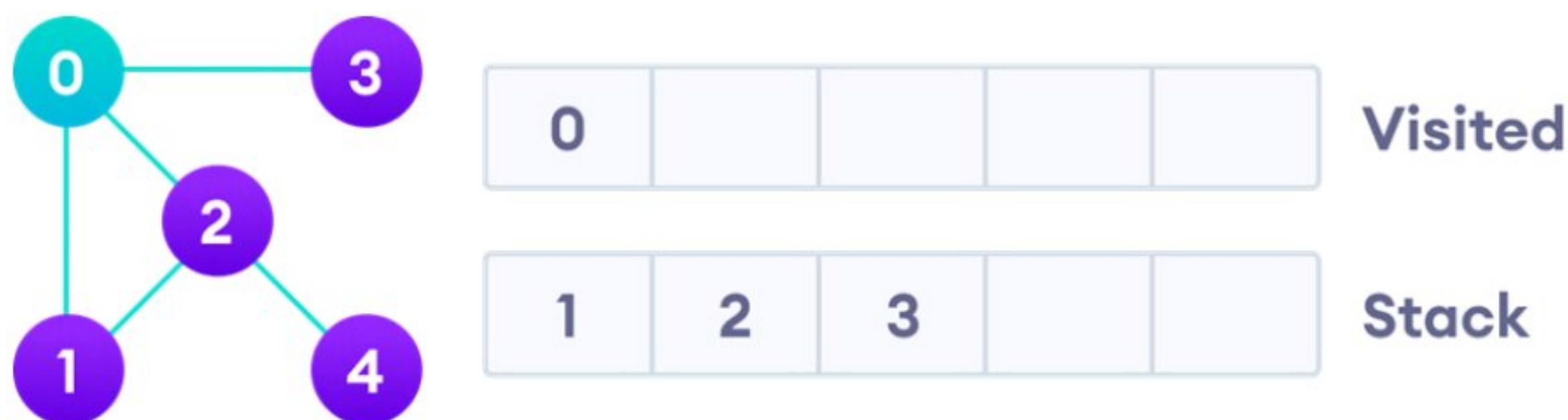
- STEP 1: Start by putting any one of the graph's vertices on top of a stack (acts as source node of DFS).

- STEP 2: Take the top item of the stack and set its visited as 1.
- STEP 3: Create a list of that vertex's adjacent nodes. Add the ones whose visited is 0 to the top of stack.
- STEP 4: Keep repeating steps 2 and 3 until the stack is empty.

An example which explains DFS Algorithm

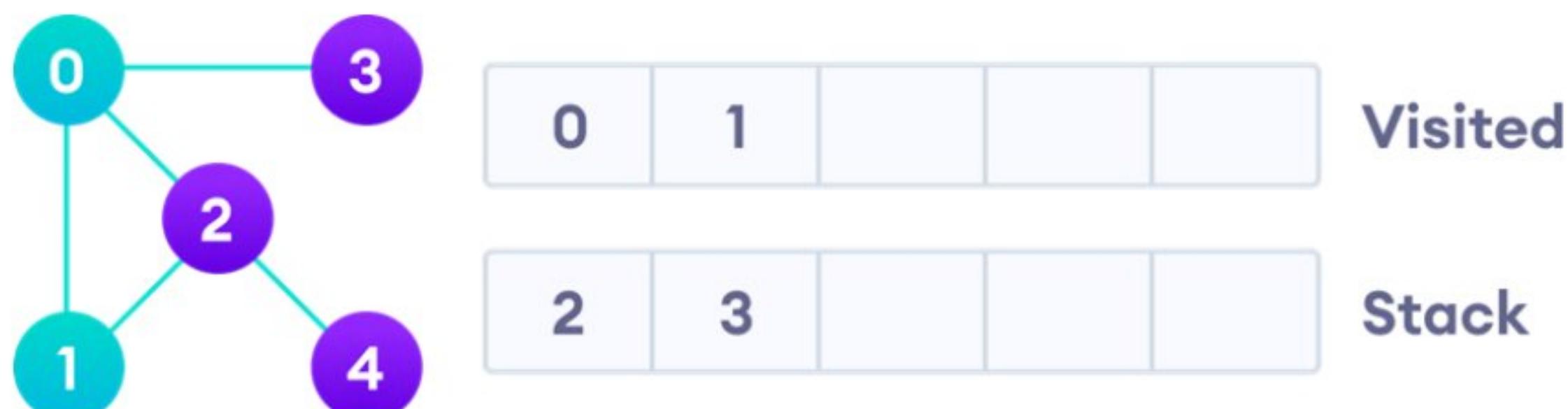


We start from vertex 0, the DFS algorithm starts by putting it in the Visited list and putting all its adjacent vertices in the stack.



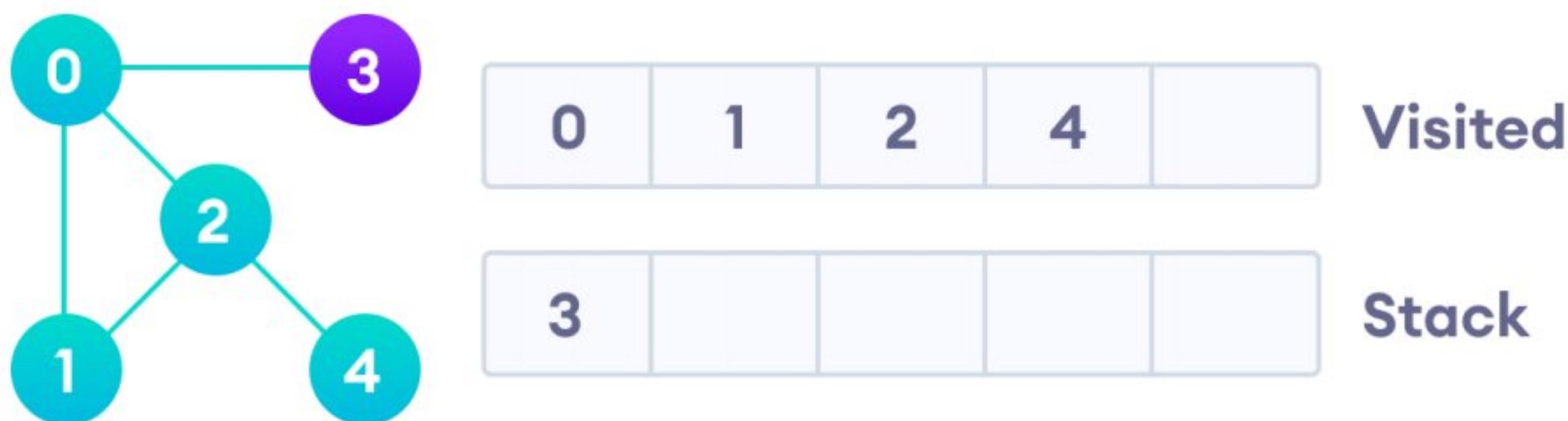
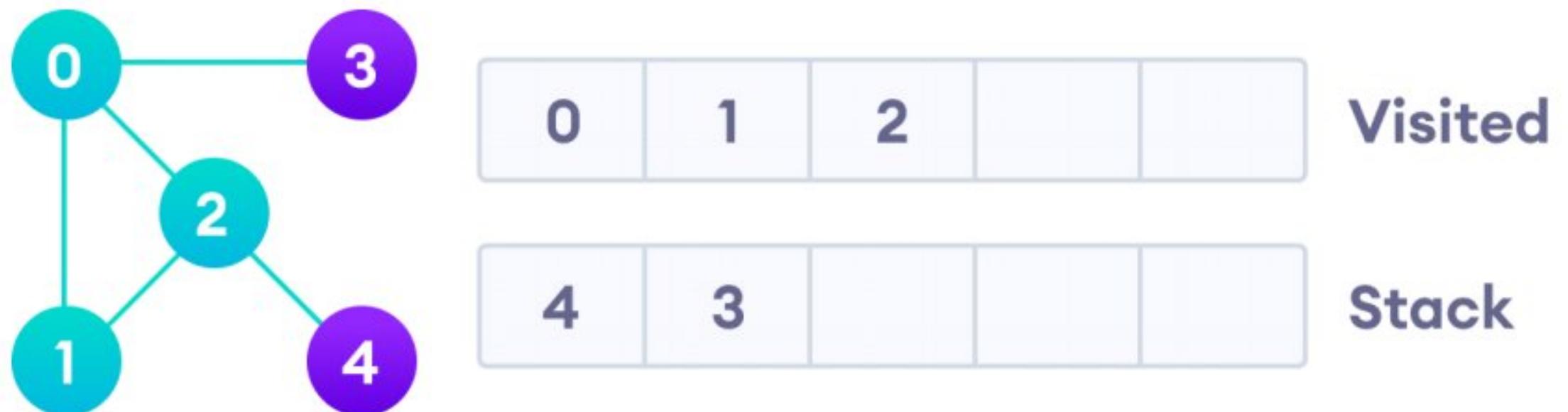
Visit the element and put it in the visited list

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.

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

For finding the path

To test if the graph is bipartite

For finding the strongly connected components of a graph

For detecting cycles in a graph

DFS Advantage:

- DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.
- It takes less time to reach to the goal node than BFS algorithm

DFS Disadvantage:

- There is the possibility that many states keep reoccurring, and there is no guarantee of finding the solution.
- DFS algorithm goes for deep down searching and sometimes it may go to the infinite loop.

BFS Queues Definition

A Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO).

- Breadth-first search is the most common search strategy for traversing a tree or graph.
- This algorithm searches breadthwise in a tree or graph, so it is called breadth-first search.
- BFS algorithm starts searching from the root node of the tree and expands all successor nodes at the current level before moving to nodes of the next level.
- The breadth-first search algorithm is an example of a general-graph search algorithm.
- Breadth-first search implemented using FIFO queue data structure

BFS Algorithm

The algorithm starts with examining the source node and all of its neighbors. In the next step, the neighbors of the nearest node of the source node are explored. The algorithm then explores all neighbors of all the nodes and ensures that each node is visited exactly once and no node is visited twice.

STEP 1: Set visited as 0 for all nodes in the Graph.

STEP 2: Enqueue the selected source node into the queue.

STEP 3: Dequeue a node N from the queue and update its visited as 1.

STEP 4: Enqueue all the neighbours of node N which are not present in the queue and whose visited

is 0.

STEP 5: Repeat steps 3 and 4 until the queue is empty.

STEP 6: EXIT

BFS Applications

- Path and Minimum Spanning Tree for unweighted graph
- In Peer to Peer Networks like BitTorrent, Breadth First Search is used to find all neighbor nodes.
- GPS Navigation systems Breadth First Search is used to find all neighboring locations.
- Cycle Detection in Undirected Graph In undirected graphs, either Breadth First Search or Depth First Search can be used to detect cycle. In directed graphs, only depth first search can be used.
- Finding all nodes within one connected component We can either use Breadth First or Depth First Traversal to find all nodes reachable from a given node.

Differences between BFS and DFS

Key Differences Between BFS and DFS:

- BFS is a vertex-based algorithm while DFS is an edge-based algorithm.
- Queue data structure is used in BFS. On the other hand, DFS uses stack or recursion.
- Memory space is efficiently utilized in DFS while space utilization in BFS is not effective.
- BFS is an optimal algorithm while DFS is not optimal.
- DFS constructs narrow and long trees whereas BFS constructs wide and short tree.

Conclusion: In This way we have studied uninformed search strategy, how to implement depth first search algorithm and Breadth First Search algorithm, Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.

Outcome: Successfully able to find depth first search algorithm and Breadth First Search algorithm.

Questions:

Q 1: Differentiate between DFS and BFS Algorithms.

Q 2: Write down the Time and Space Complexity of DFS and BFS.

Q 3: Which data structure is used by DFS and BFS?

4.2. Experiment No. 2

Aim:

Implement A* star Algorithm for any game search problem.

Objective:

In this experiment, we will be able to do the following:

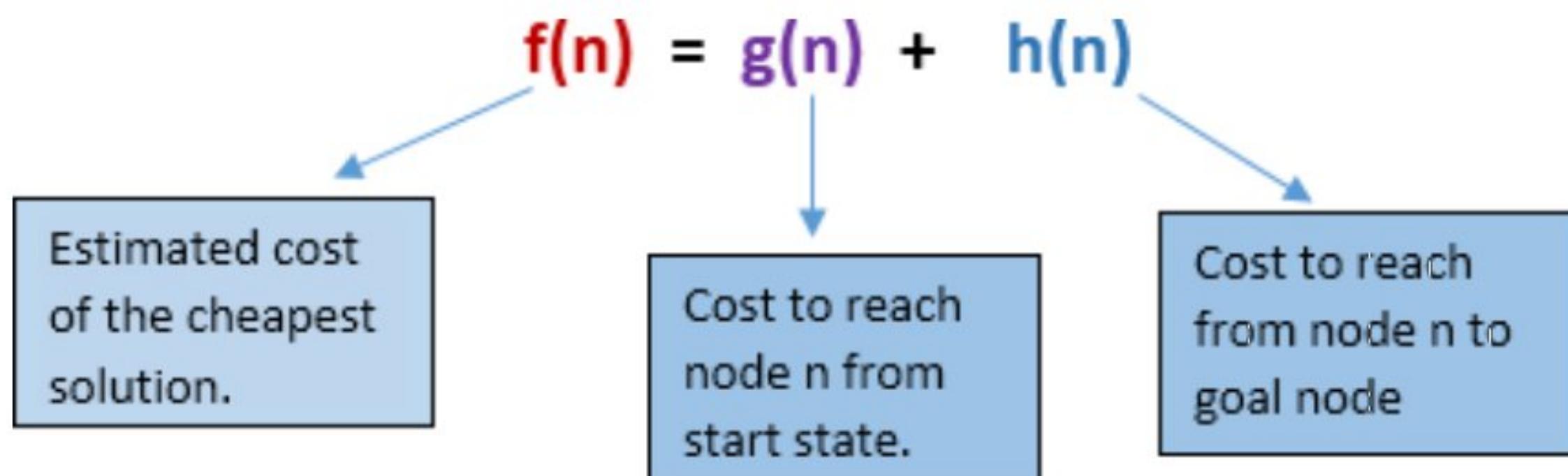
- To understand informed Search Strategies.
- To make use of Graph and Tree Data Structure for implementation of informed Search strategies.
- Study the A* algorithms for various graph traversal problem.

Theory:
A* Search

A* search is the most commonly known form of best-first search. It uses 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 solves 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:

Step 1: 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 return success and stop, otherwise

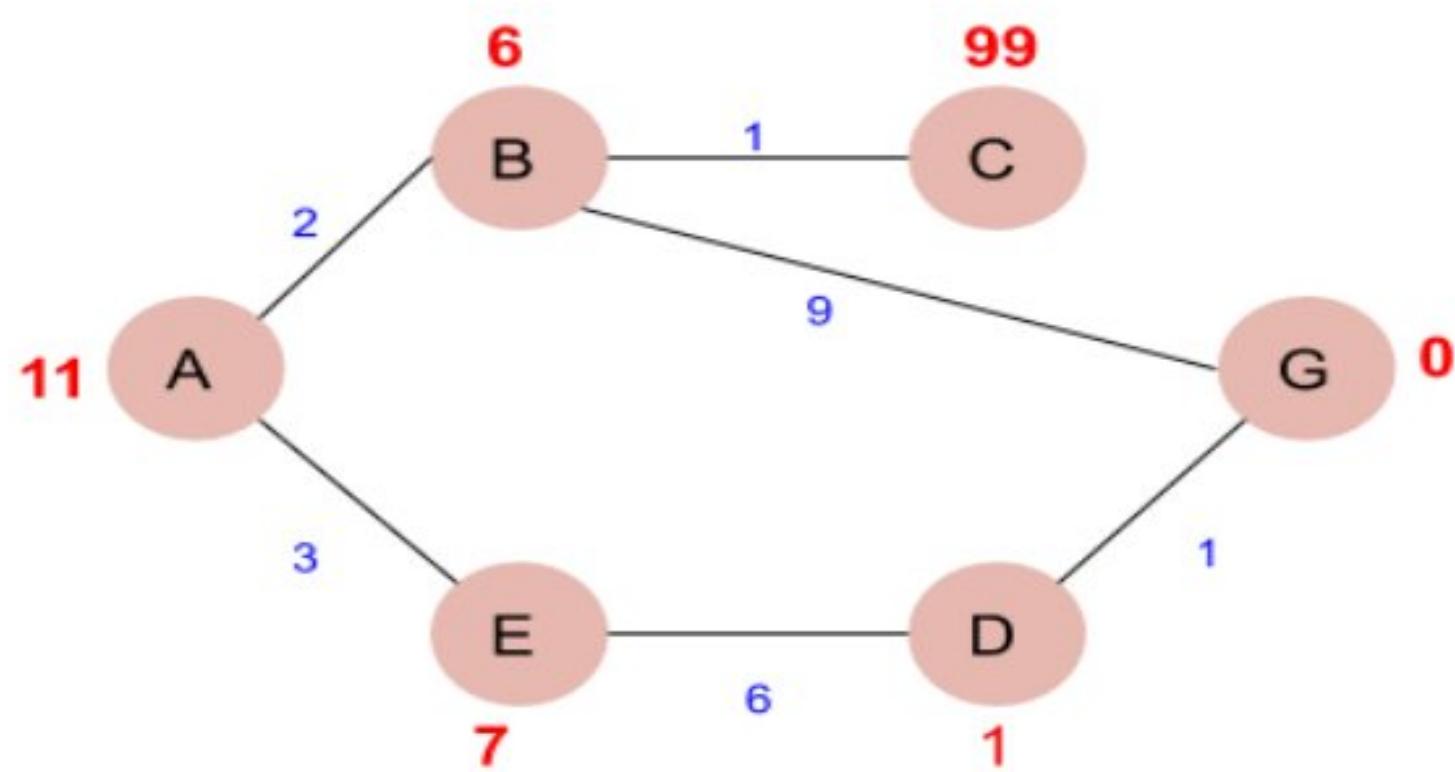
Step 4: Expand node n and generate all of its successors, and put n into the closed list. For each successor n', check whether n' is already in the OPEN or CLOSED list, if not then compute evaluation function for n' and place into Open list.

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.

Implementation with Python

In this section, we are going to find out how the A* search algorithm can be used to find the most cost-effective path in a graph. Consider the following graph below.



The numbers written on edges represent the distance between the nodes, while the numbers written on nodes represent the heuristic values. Let us find the most cost-effective path to reach from start state A to final state G using the A* Algorithm.

Let's start with node A. Since A is a starting node, therefore, the value of $g(x)$ for A is zero, and from the graph, we get the heuristic value of A is 11, therefore

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

$$0+11=11$$

Thus for A, we can write $A=11$

Now from A, we can go to point B or point E, so we compute $f(x)$ for each of them

$$A \rightarrow B = 2 + 6 = 8$$

$$A \rightarrow E = 3 + 6 = 9$$

Since the cost for $A \rightarrow B$ is less, we move forward with this path and compute the $f(x)$ for the children nodes of B

Since there is no path between C and G, the heuristic cost is set to infinity or a very high value

$$A \rightarrow B \rightarrow C = (2 + 1) + 99 = 102$$

$$A \rightarrow B \rightarrow G = (2 + 9) + 0 = 11$$

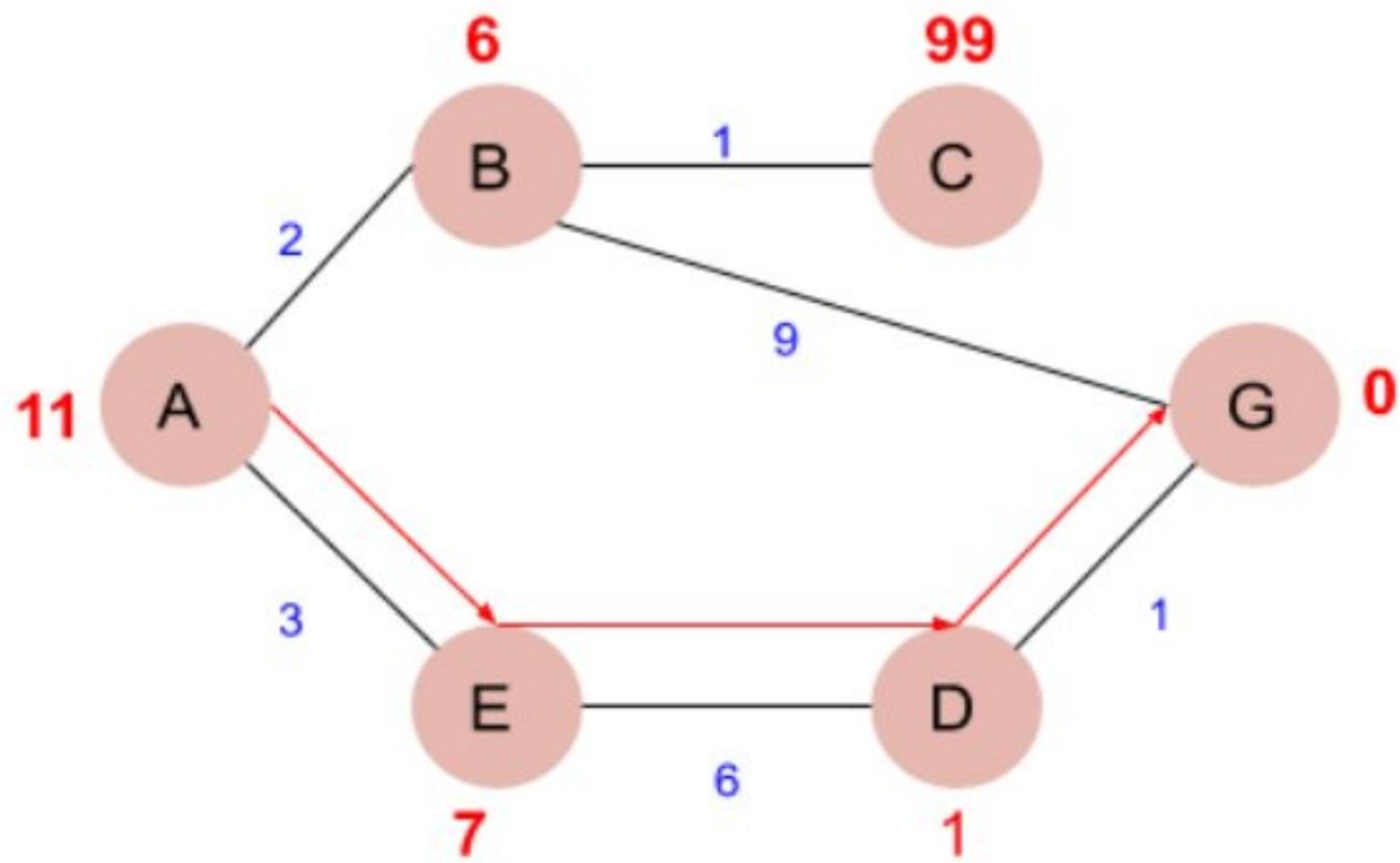
Here the path $A \rightarrow B \rightarrow G$ has the least cost but it is still more than the cost of $A \rightarrow E$, thus we explore this path further

$$A \rightarrow E \rightarrow D = (3 + 6) + 1 = 10$$

Comparing the cost of $A \rightarrow E \rightarrow D$ with all the paths we got so far and as this cost is least of all we move forward with this path. And compute the $f(x)$ for the children of D

$$A \rightarrow E \rightarrow D \rightarrow G = (3 + 6 + 1) + 0 = 10$$

Now comparing all the paths that lead us to the goal, we conclude that $A \rightarrow E \rightarrow D \rightarrow G$ is the most cost-effective path to get from A to G.



Input: # heuristic distance for all nodes

```
def heuristic(n):
```

```
    H_dist = {
        'A': 11,
        'B': 6,
        'C': 99,
        'D': 1,
        'E': 7,
        'G': 0,
```

```
}
```

```
    return H_dist[n]
```

#Describe your graph here

```
Graph_nodes = {
```

```
    'A': [('B', 2), ('E', 3)],
    'B': [('C', 1), ('G', 9)],
    'C': None,
    'E': [('D', 6)],
    'D': [('G', 1)],
```

```
}
```

Output:

Path Found: ['A', 'E', 'D', 'G']

Advantages:

1. A* search algorithm is the best algorithm than other search algorithms.
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 mostly based on heuristics and approximation.
2. A* search algorithm has some complexity issues.
3. The main drawback of A* is memory requirement as it keeps all generated nodes in the memory, so it is not practical for various large-scale problems.

Applications:

A-Star algorithm is used to determine the closest path the enemy can go to the tree.

Furthermore, the player must try to protect the tree from the enemy by touching the enemy or using a barrier to keep the tree alive.

Conclusion:

Successfully able to implement graph traversal problem using A* Algorithm.

Questions:

1. How is the heuristic function used in the A* algorithm, and what is its significance?

2. What is the role of a priority queue in the A* algorithm?
3. How does A* guarantee finding the optimal solution, and under what conditions does it hold true?
4. What is an admissible heuristic, and why is it important in the context of the A* algorithm?
5. Explain the similarities and differences between A* and the Best-First Search algorithm.

4.3. Experiment No. 3

Aim:

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

Objective:

Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and

Backtracking for n-queens problem or a graph coloring problem

Theory:**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 diagonal.

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 diagonal
4. no two queens share the same right diagonal

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 suggest 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 diagonals 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 diagonal, and other for top-right to bottom-left diagonal. We need to fill these in such a way that two queens sharing same top-left bottom-right diagonal will have same value in slash Diagonal and two queens sharing same top-right bottom-left diagonal will have same value in back Slash Diagonal.

$$\text{slashDiagnol}(\text{row})(\text{col}) = \text{row} + \text{col}$$

$$\text{backSlashDiagnol}(\text{row})(\text{col}) = \text{row} - \text{col}$$

$$+ (N-1) \quad \{ N = 8 \}$$

{ we added (N-1) as we do not need negative values in backSlashDiagnol }

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 diagonals; and recur for queen **i+1**.

Applications:

Constraint satisfaction problem in AI has a wide range of applications, including scheduling, resource allocation, and automated reasoning.

Branch and bound is an algorithm used to solve combinatorial optimization problems

Input: n=8

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 can be 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 be easily transformed into a graph. Each region of the map becomes the node, and if two 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

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 diagonal[row][col] = row + col

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.

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 diagonal[row][col] = row - col + (N-1)

The colors are represented by the integers $1, 2, \dots, m$ and the solutions are given by the n-tuple $(x_1, x_2, x_3, \dots, x_n)$, where x_i is the color of node i .

Algorithm for finding the m - colorings of

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

graph**Conclusion:**

Thus we have implemented pre-processing of a text document such as stop word removal, stemming.

Outcome:

Upon completion of this experiment, students will be able to: Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem

Questions:

- Q.1) Which are the constraints required to solve N Queen problem?
- Q.2) Compare backtracking and branch and bound method
- Q.3) What do mean by constraints satisfaction problem.

4.4. Experiment No. 4

Aim:

Implement Alpha-beta Tree search for any game search problem

Objective:

Students will be able to understand how Alpha-beta Tree search is used in any game search problem

Theory:

- Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.
- As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called **pruning**. This involves two threshold parameter Alpha and beta for future expansion, so it is called **alpha-beta pruning**. It is also called as **Alpha-Beta Algorithm**.
- Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.
- The two-parameter can be defined as:
 - a. **Alpha:** The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is $-\infty$.
 - b. **Beta:** The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is $+\infty$.
- The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.
- The main condition which required for alpha-beta pruning is:

$$\alpha >= \beta$$

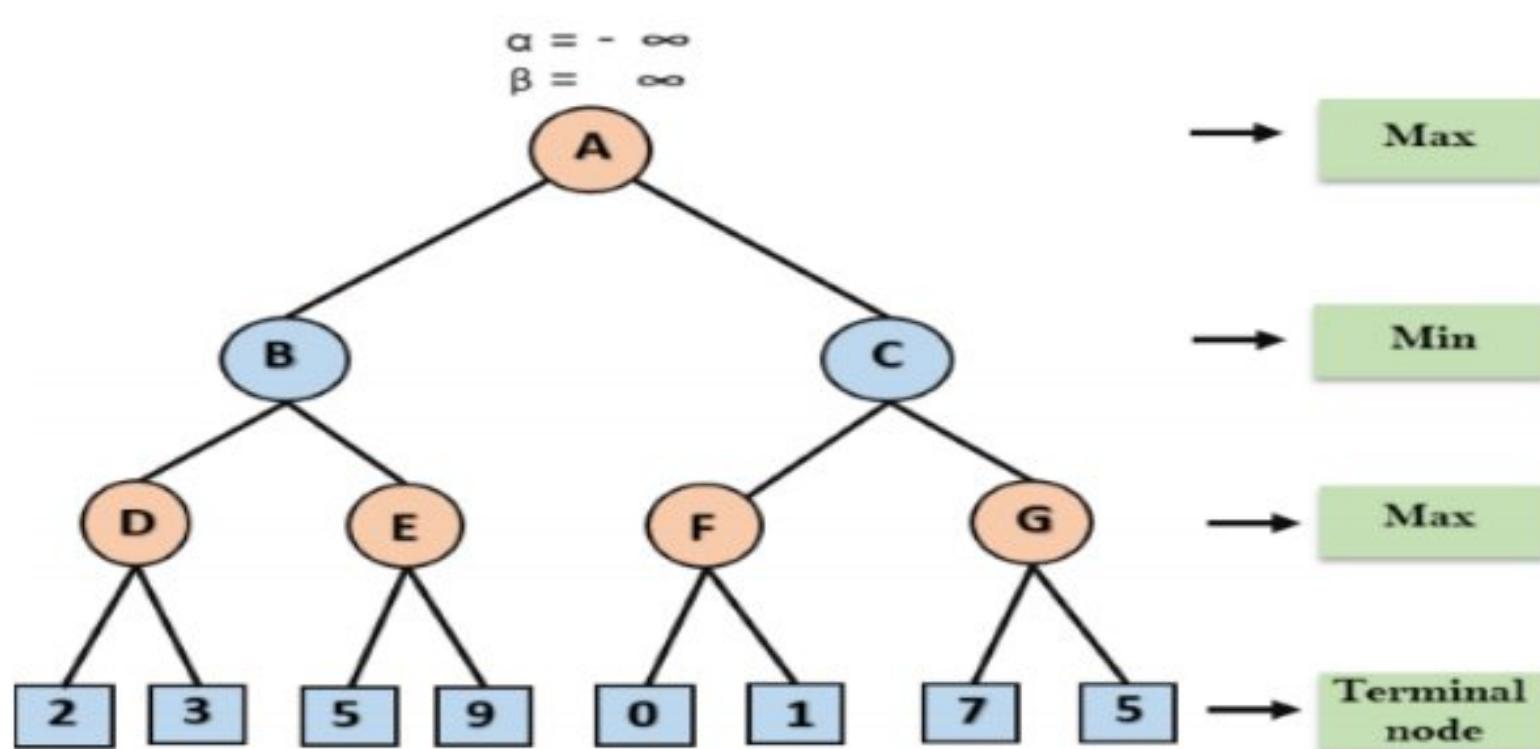
- The Max player will only update the value of alpha.
- The Min player will only update the value of beta.

- While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
- We will only pass the alpha, beta values to the child nodes.

Working of Alpha-Beta Pruning:

Let's take an example of two-player search tree to understand the working of Alpha-beta pruning

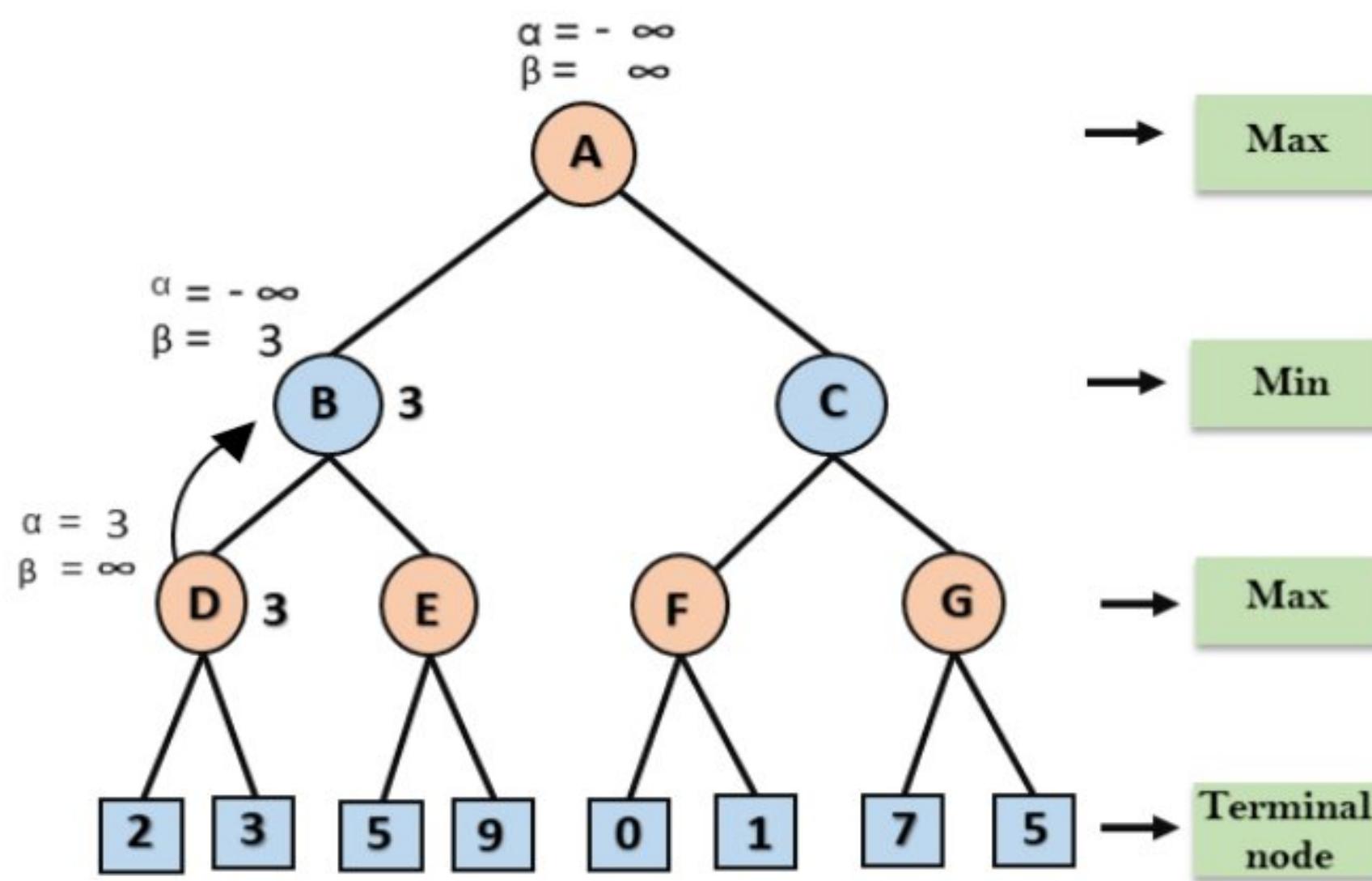
Step 1: At the first step the, Max player will start first move from node A where $\alpha = -\infty$ and $\beta = +\infty$, these value of alpha and beta passed down to node B where again $\alpha = -\infty$ and $\beta = +\infty$, and Node B passes the same value to its child D.



Step 2: At Node D, the value of α will be calculated as its turn for Max. The value of α is compared with firstly 2 and then 3, and the max (2, 3) = 3 will be the value of α at node D and node value will also 3.

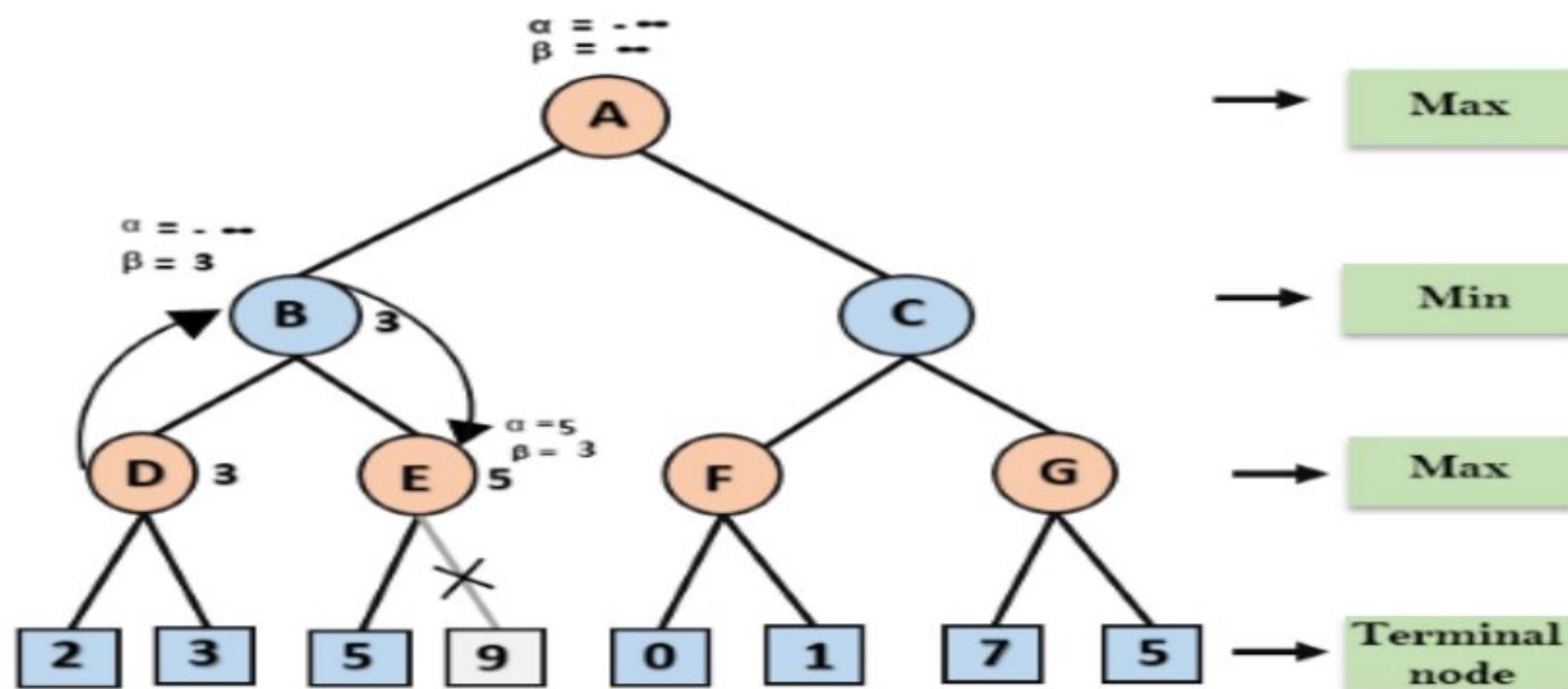
Step 3: Now algorithm backtrack to node B, where the value of β will change as this is a turn of Min, Now $\beta = +\infty$, will compare with the available subsequent

nodes value, i.e. $\min(\infty, 3) = 3$, hence at node B now $\alpha = -\infty$, and $\beta = 3$.



In the next step, algorithm traverse the next successor of Node B which is node E, and the values of $\alpha = -\infty$, and $\beta = 3$ will also be passed.

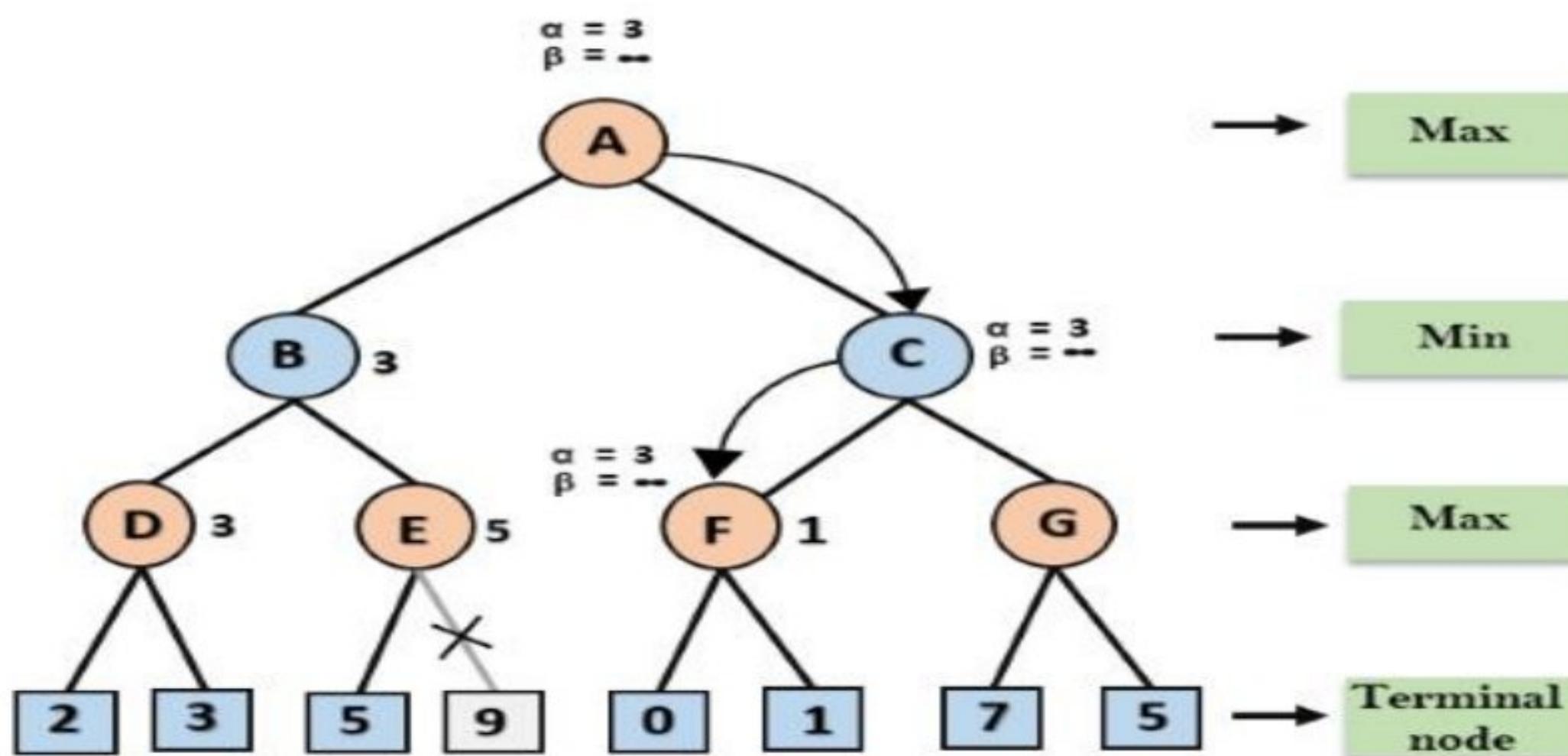
Step 4: At node E, Max will take its turn, and the value of alpha will change. The current value of alpha will be compared with 5, so $\max(-\infty, 5) = 5$, hence at node E $\alpha = 5$ and $\beta = 3$, where $\alpha >= \beta$, so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be 5.



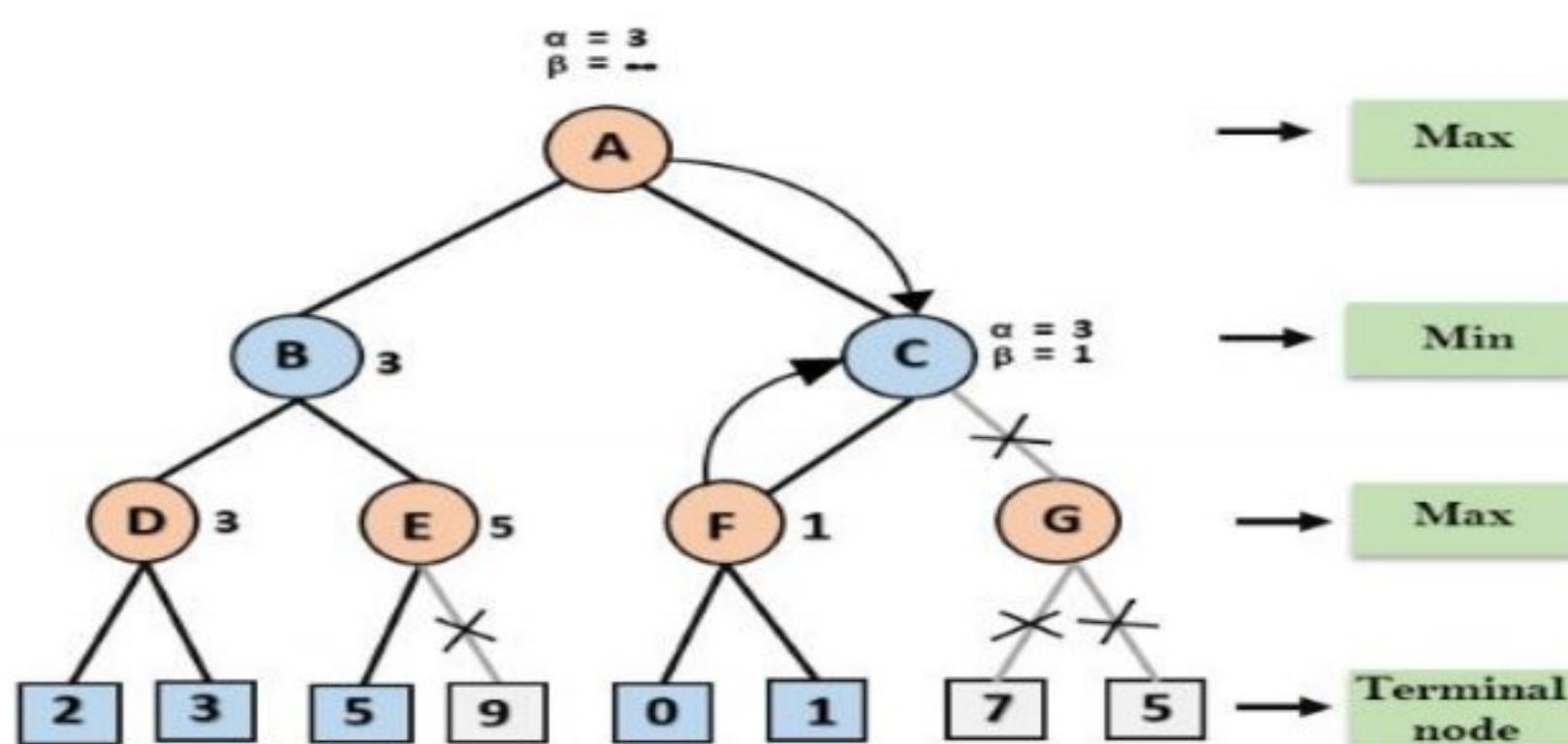
Step 5: At next step, algorithm again backtrack the tree, from node B to node A. At node A, the value of alpha will be changed the maximum available value is 3 as max $(-\infty, 3) = 3$, and $\beta = +\infty$, these two values now passes to right successor of A which is Node C.

At node C, $\alpha=3$ and $\beta= +\infty$, and the same values will be passed on to node F.

Step 6: At node F, again the value of a will be compared with left child which is 0, and $\max(3,0) = 3$, and then compared with right child which is 1, and $\max(3,1) = 3$ still a remains 3, but the node value of F will become 1.

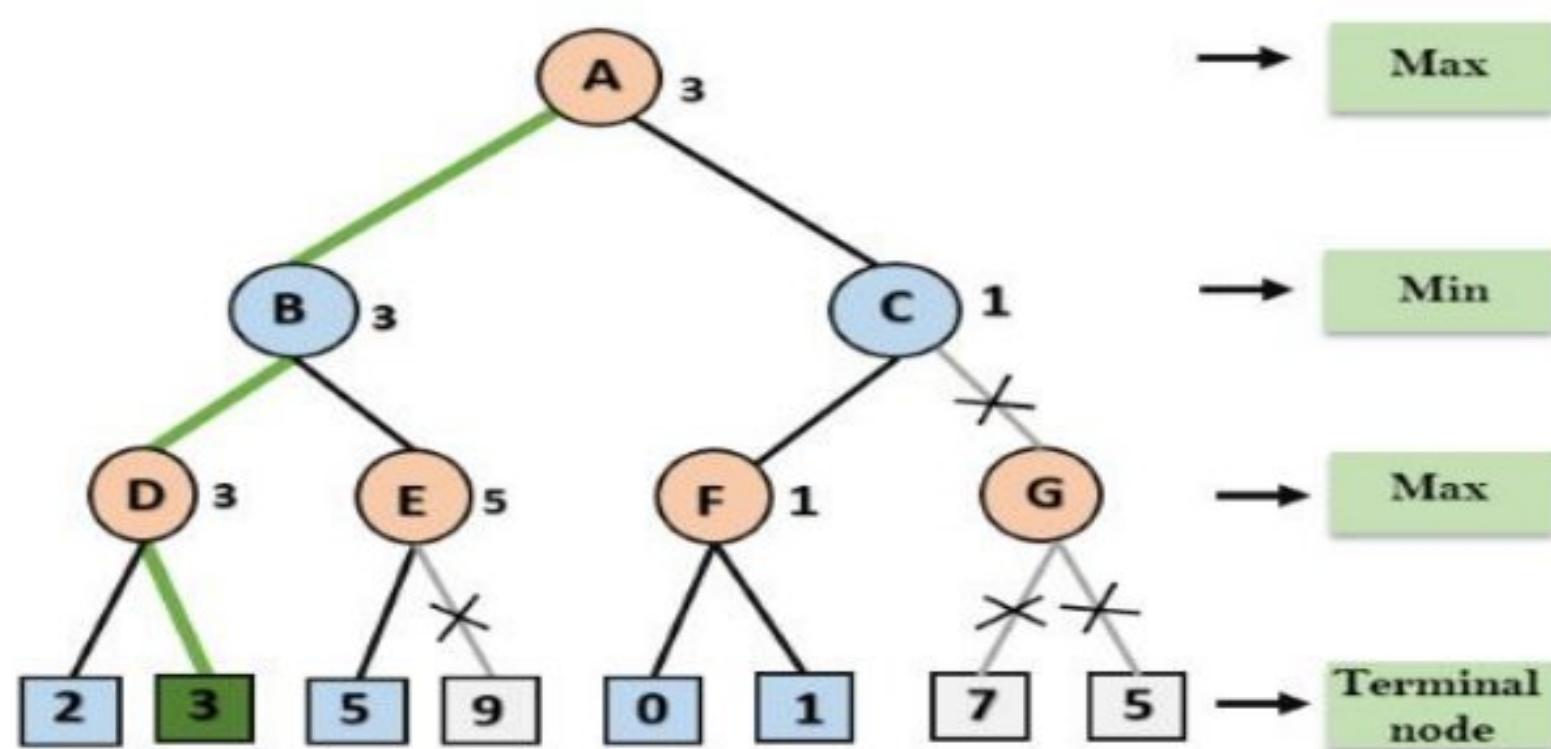


Step 7: Node F returns the node value 1 to node C, at C $\alpha = 3$ and $\beta = +\infty$, here the value of beta will be changed, it will compare with 1 so $\min(\infty, 1) = 1$. Now at C, $\alpha = 3$ and $\beta = 1$, and again it satisfies the condition $\alpha >= \beta$, so the next child of C which is G will be pruned, and the algorithm will not compute the entire sub-tree G.



Step 8: C now returns the value of 1 to A here the best value for A is $\max(3, 1) = 3$. Following is the final game tree which is showing the nodes which are computed

and nodes which has never computed. Hence the optimal value for the maximizer is 3 for this example.



Move Ordering in Alpha-Beta pruning:

The effectiveness of alpha-beta pruning is highly dependent on the order in which each node is examined. Move order is an important aspect of alpha-beta pruning.

It can be of two types:

- **Worst ordering:** In some cases, alpha-beta pruning algorithm does not prune any of the leaves of the tree, and works exactly as minimax algorithm. In this case, it also consumes more time because of alpha-beta factors, such a move of pruning is called worst ordering. In this case, the best move occurs on the right side of the tree. The time complexity for such an order is $O(b^m)$.
- **Ideal ordering:** The ideal ordering for alpha-beta pruning occurs when lots of pruning happens in the tree, and best moves occur at the left side of the tree. We apply DFS hence it first search left of the tree and go deep twice as minimax algorithm in the same amount of time. Complexity in ideal ordering is $O(b^{m/2})$.

Rules to find good ordering:

Following are some rules to find good ordering in alpha-beta pruning:

- Occur the best move from the shallowest node.
- Order the nodes in the tree such that the best nodes are checked first.
- Use domain knowledge while finding the best move. Ex: for Chess, try order: captures first, then threats, then forward moves, backward moves.
- We can bookkeep the states, as there is a possibility that states may repeat.

Algorithm:

1. First, generate the entire game tree starting with the current position of the game all the way up to the terminal states.
2. Apply the utility function to get the utility values for all the terminal states.
3. Determine the utilities of the higher nodes with the help of the utilities of the terminal nodes. For instance, in the diagram below, we have the utilities for the terminal states written in the squares.
4. Calculate the utility values with the help of leaves considering one layer at a time until the root of the tree.
5. Eventually, all the backed-up values reach to the root of the tree, i.e., the topmost point. At that point, MAX has to choose the highest value.

Applications:

Alpha Beta Pruning is an optimization technique of the Minimax algorithm. This algorithm solves the limitation of exponential time and space complexity in the case of the Minimax algorithm by pruning redundant branches of a game tree using its parameters Alpha(α) and Beta(β).

Input: { 3, 5, 6, 9, 1, 2, 0, -1 }

Output: The optimal value is : 5

Conclusion:

Thus we have implemented pre-processing of a text document such as stop word removal, stemming.

Outcome:

Upon completion of this experiment, students will be able to:

Experiment level outcome (SL-1): Implementation of alpha beta tree search for any game search problem.

Questions:

1. What is the Minimax algorithm?

2. Why is the time complexity of the Minimax algorithm so high?
3. What are the best case and worst case time complexity of alpha-beta pruning?
4. Can alpha-beta pruning be used for applications apart from games?

4.5.Experiment No. 5

1. Title of Assignment: Develop an elementary chatbot for any suitable customer interaction application.

2. Prerequisite: Basic knowledge of AI and customer interaction etc.

3. Objective: To create a chatbot to interact with customers.

4. Outcome: Successfully able to implement Chatbot for different applications

5. Software and Hardware Requirement: Open Source C++ Programming tool like G++/GCC, python,java and Ubuntu.

5. Relevant 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 they are 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.

some of the benefits that chatbots provide:

1. Available 24*7: 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

Applications:

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.
- b. **Customer Engagement Clearance Sale:** Notify users about on-going clearance sale of products relevant to the users at their nearest outlets.
- c. **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

d. **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. 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

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.

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.

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 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 into the Finance Industry. Chatbots can be programmed to assist 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?
- Improve conversions from your existing SMS campaigns.
- Talk to your customers when they want to use “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.

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.

Input:

"hello":

"how are you":

"name":

"bye":

Output:

"Hello! How can I assist you today?"

"I'm just a computer program, but I'm doing well. How can I help you?"

"I don't have a name, you can call me Chatbot."

"Goodbye! If you have more questions, feel free to ask."

Outcome: Students will be successfully able to build chatbot for various application

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?

5.1.Experiment No. 6

Aim: **Implement Greedy search algorithm for any ofthe following application:**

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

Objective:

Theory: **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 have only 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 exploring the 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 and keep 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

Step 1: Determine an arbitrary vertex as the starting vertex of the MST.

Step 2: Follow steps 3 to 5 till there are vertices that are not included in the MST (known as fringe vertex).

Step 3: Find edges connecting any tree vertex with the fringe vertices.

Step 4: Find the minimum among these edges.

Step 5: Add the chosen edge to the MST if it does not form any cycle.

Step 6: Return the MST and exit

Pseudocode

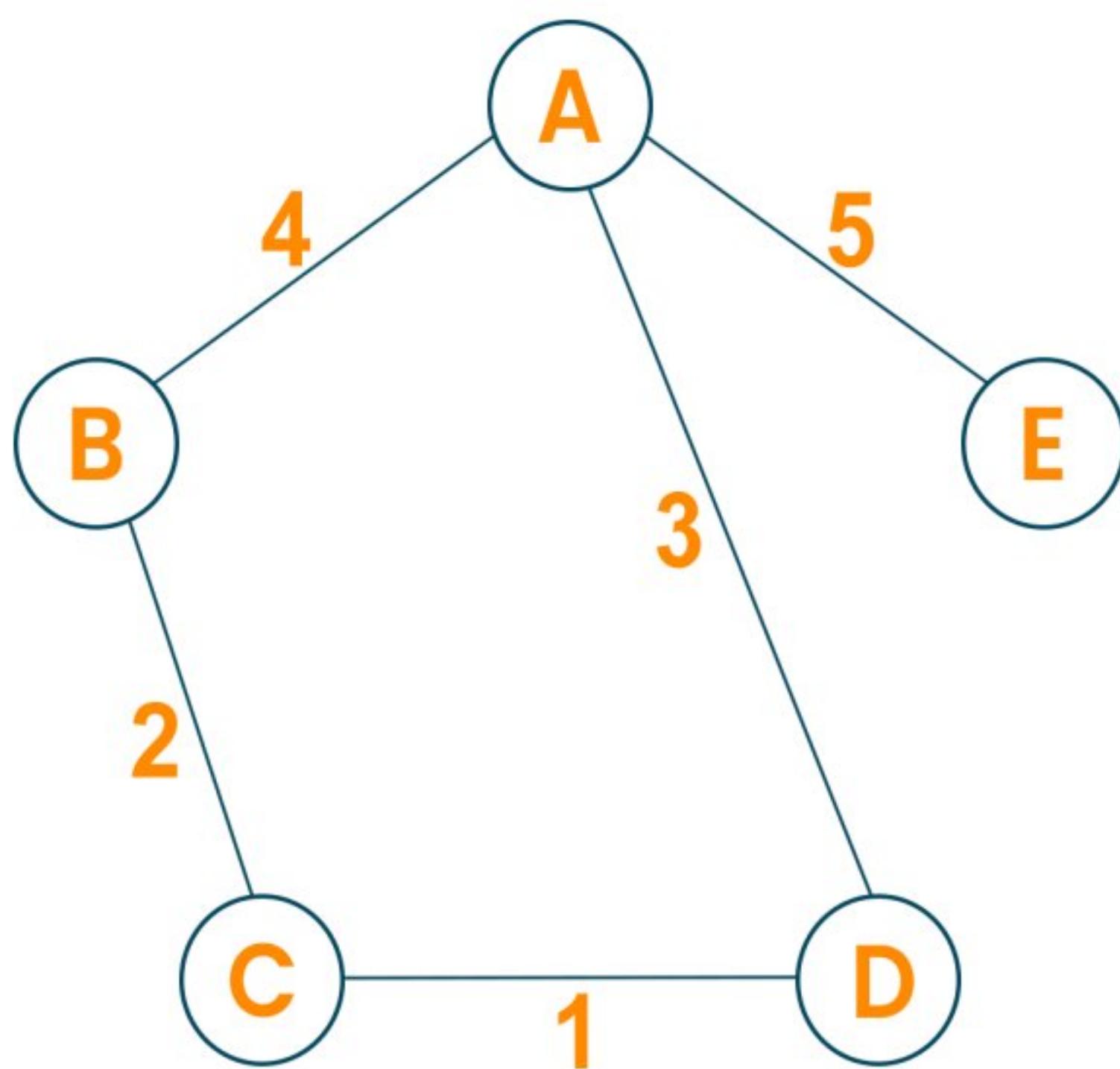
```

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

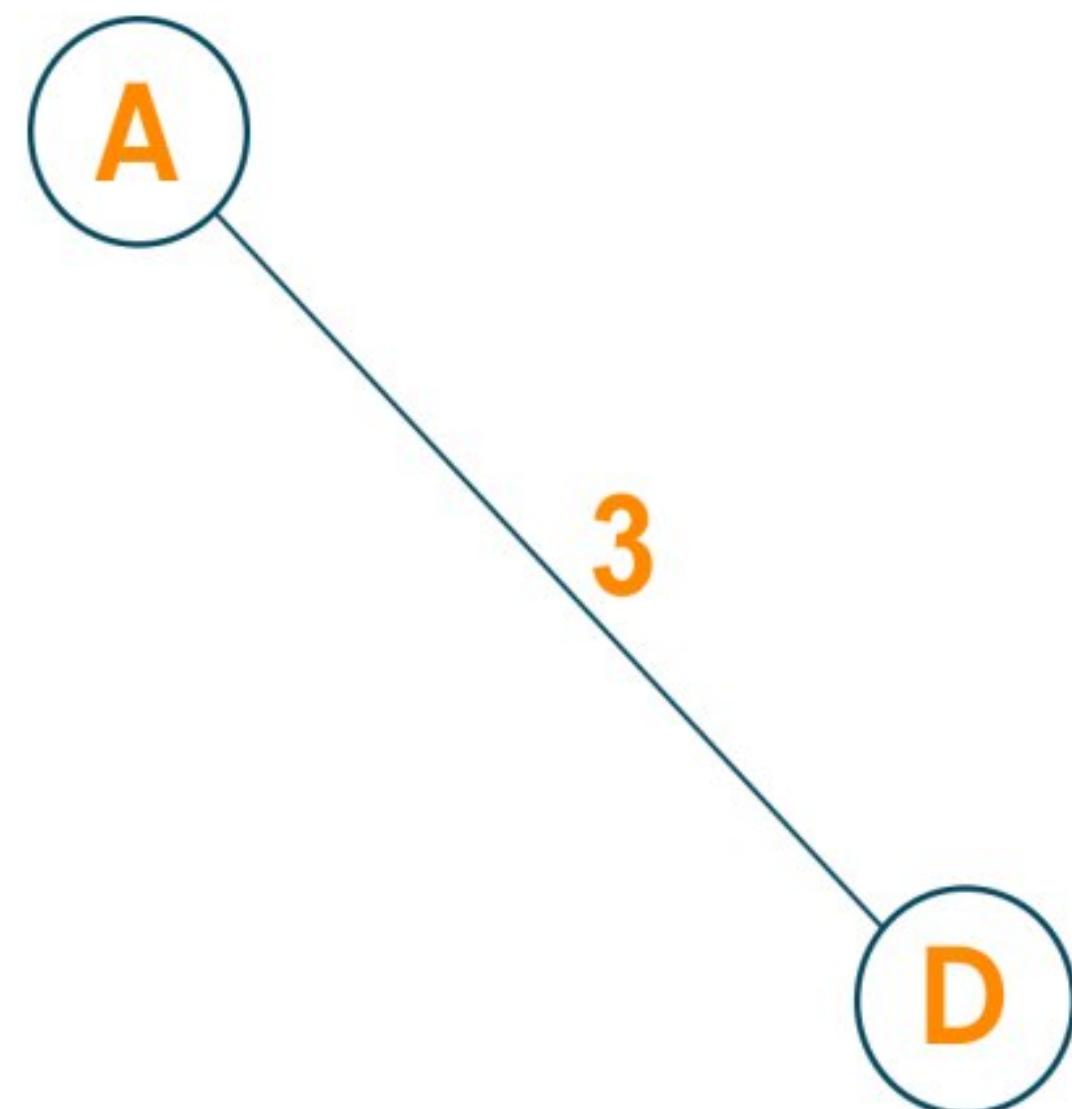
Let us consider the below-weighted graph



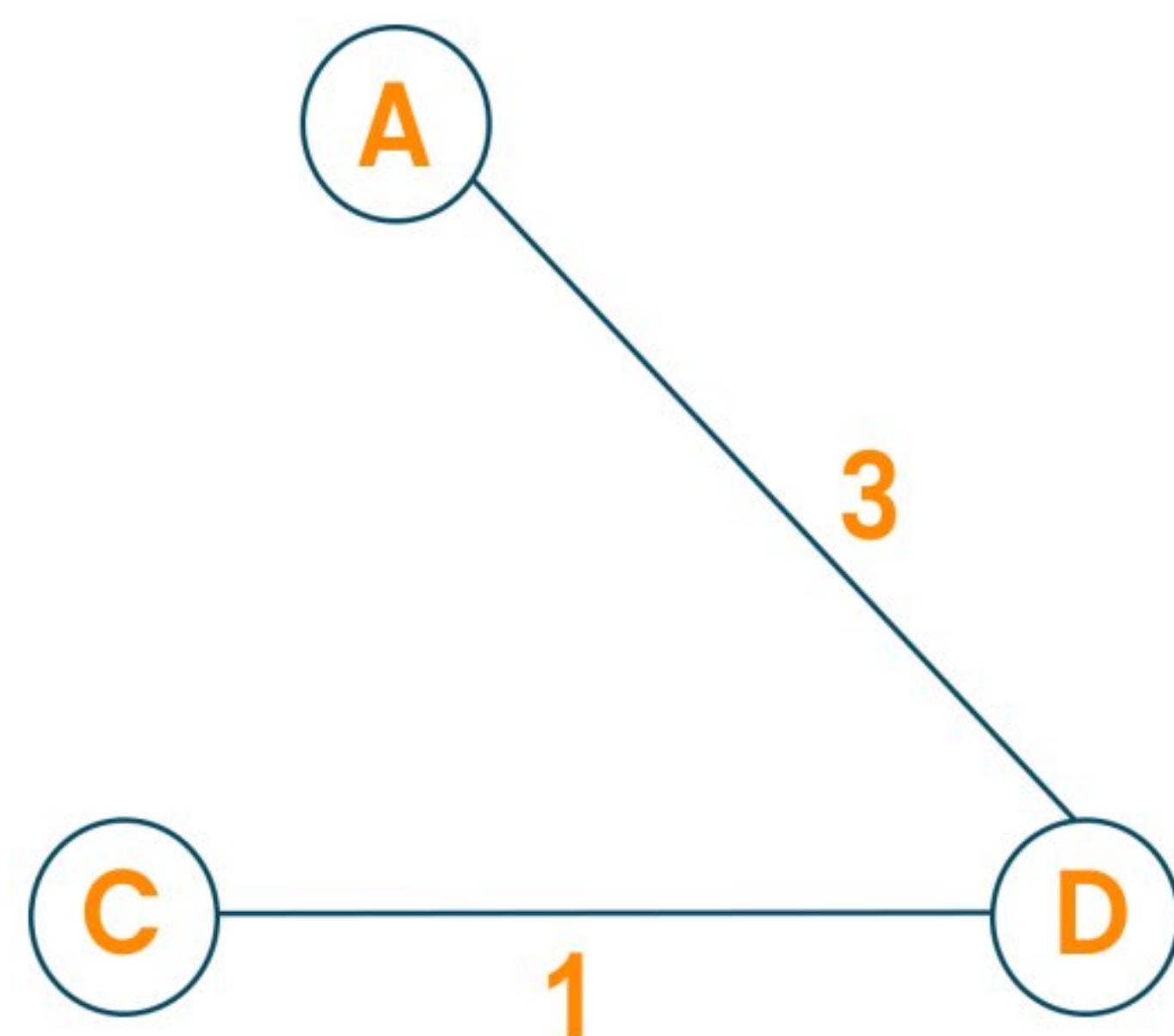
Later we will consider the source vertex to initialize the algorithm



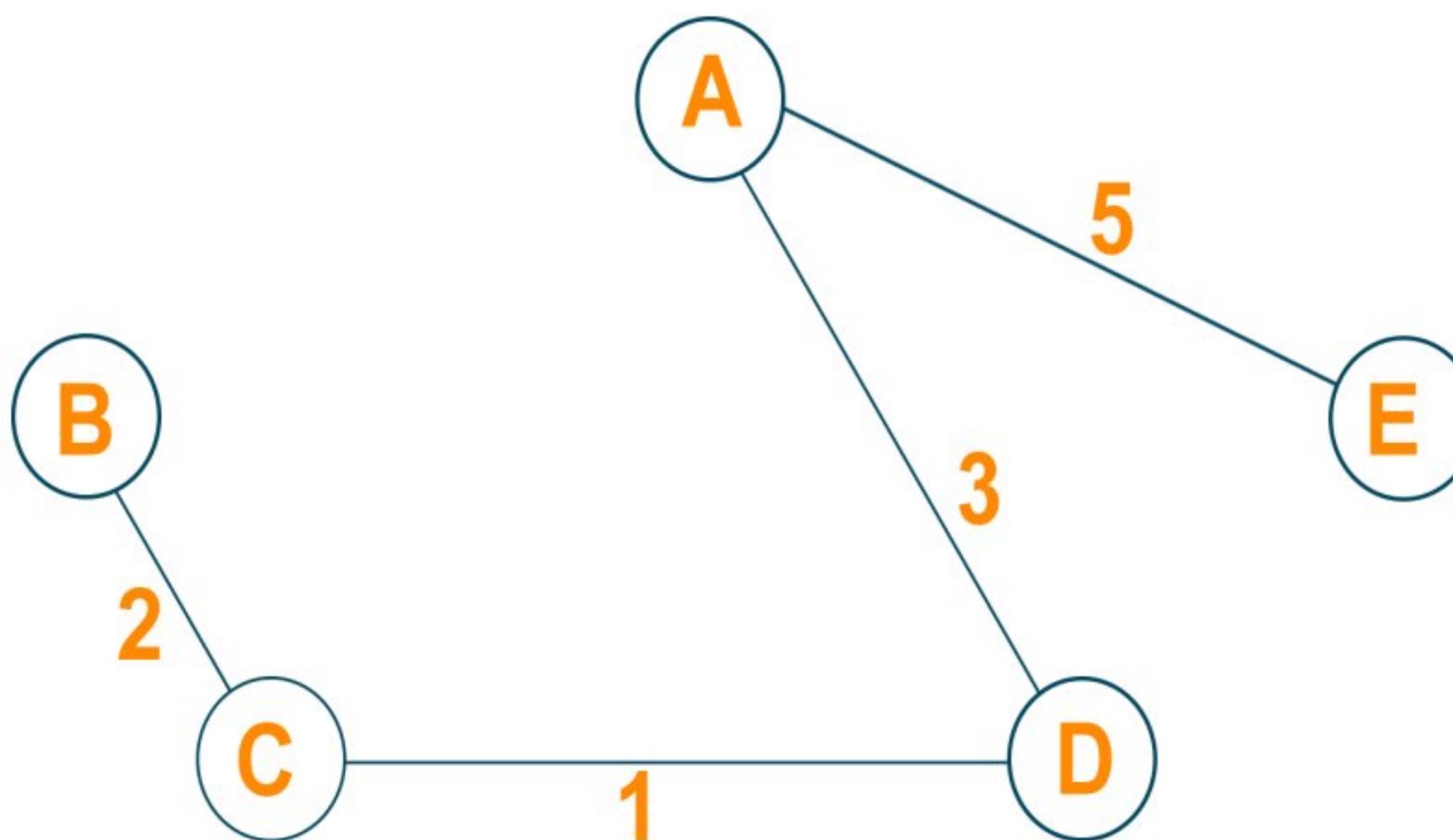
Now, we will choose the shortest weight edge from the source vertex and add it to finding the spanning tree.



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.



Time Complexity:

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

```

# Prim's Algorithm in Python

INF = 9999999
# number of vertices in graph
N = 5
#creating graph by adjacency matrix method
G = [[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 = INF
a = 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

```

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.

Input: g.graph = [[0, 2, 0, 6, 0],

```

[2, 0, 3, 8, 5],
[0, 3, 0, 0, 7],
[6, 8, 0, 0, 9],
[0, 5, 7, 9, 0]]

```

Or

```
G = [[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]]
```

Output:

Edge	Weight
0 - 1	2
1 - 2	3
0 - 3	6
1 - 4	5

Or

Edge : Weight

0-2:5
2-3:1
3-4:1
4-1:2

Conclusion: Successfully able to implement Greedy search Algorithm for Prims Minimum Spanning tree.

Outcome: The minimum spanning tree has its own importance to learn the prim's algorithm which leads to find the solution to many problems. When it comes to finding the minimum spanning tree for the dense graphs, prim's algorithm is the first choice.

Questions:

1. How does Prim's Algorithm Work?
2. What Is a Minimum Spanning Tree?
3. What Is Prim's Minimum Spanning Tree Algorithm?
4. How to Implement the Prim's Algorithm
5. Does Prim's algorithm run on a negative edge weight graph?
6. Can there be multiple MST for a graph?

6. Appendix

Introduction

[Python](#) is a widely used high-level [programming language](#). It is one of the most popular and flexible server-side programming languages.

Windows does not have the Python programming language installed by default. However, you can install Python on Windows in just a few easy steps.

This guide provides step-by-step instructions to install and set up Python on Windows.

Prerequisites

- A system running Windows 10 with administrator access.
- Access to the [command prompt](#).
- Access to a [web browser](#).

Python Installation on Windows

The installation requires downloading the official Python .exe installer and running it on your system. The sections below will explain several options and details during the installation process.

Step 1: Select Python Version

Deciding on a version depends on what you want to do in Python. The two major versions are Python 2 and Python 3. Choosing one over the other might be better depending on your project details. If there are no constraints, choose whichever one you prefer.

We recommend Python 3, as Python 2 reached its end of life in 2020. Download Python 2 only if you work with legacy scripts and older projects. Also, choose a stable release over the newest since the newest release may have bugs and issues.

Step 2: Download Python Executable Installer

Start by downloading the Python executable installer for Windows:

1. Open a web browser and navigate to the [Downloads for Windows section](#) of the official Python website.
2. Locate the desired Python version.
3. Click the link to download the file. Choose either the Windows [32-bit or 64-bit](#) installer.

The download is approximately 25MB.

Step 3: Run Executable Installer

The steps below guide you through the installation process:

1. Run the downloaded **Python Installer**.
2. The installation window shows two checkboxes:
 - **Admin privileges**. The parameter controls whether to install Python for the current or all system users. This option allows you to change the installation [folder](#) for Python.
 - **Add Python to PATH**. The second option places the executable in the PATH variable after installation. You can also add Python to the PATH environment variable manually later.

For the most straightforward installation, we recommend ticking both checkboxes.

3. Select the **Install Now** option for the recommended installation (in that case, skip the next two steps).

To adjust the default installation options, choose **Customize installation** instead and proceed to the following step.

The default installation installs Python to `C:\Users\[user]\AppData\Local\Programs\Python\Python[version]` for the current user. It includes IDLE (the default [Python editor](#)), the PIP package manager, and additional documentation. The installer also creates necessary shortcuts and file associations.

Customizing the installation allows changing these installation options and parameters.

4. Choose the optional installation features. Python works without these features, but adding them improves the program's usability.

Click **Next** to proceed to the Advanced Options screen.

5. The second part of customizing the installation includes advanced options.

Choose whether to install Python for all users. The option changes the install location to `C:\Program Files\Python[version]`. If selecting the location manually, a common choice is `C:\Python[version]` because it avoids spaces in the path, and all users can access it. Due to administrative rights, both paths may cause issues during package installation.

Other advanced options include creating shortcuts, file associations, and adding Python to PATH.

After picking the appropriate options, click **Install** to start the installation.

6. Select whether to disable the path length limit. Choosing this option will allow Python to bypass the 260-character **MAX_PATH** limit.

The option will not affect any other system settings, and disabling it resolves potential name-length issues. We recommend selecting the option and closing the setup.

Step 4: Add Python to Path (Optional)

If the Python installer does not include the **Add Python to PATH** checkbox or you have not selected that option, continue in this step. Otherwise, skip to the next step.

Adding the Python path to the PATH variable alleviates the need to use the full path to access the Python program in the command line. It instructs Windows to review all the folders added to the PATH environment variable and to look for the `python.exe` program in those folders.

To add Python to PATH, do the following:

1. In the **Start** menu, search for **Environment Variables** and press **Enter**.

2. Click **Environment Variables** to open the overview screen.

3. Double-click **Path** on the list to edit it.

Alternatively, select the variable and click the **Edit** button.

4. Double-click the first empty field and paste the Python installation folder path.

Alternatively, click the **New** button instead and paste the path.

5. Click **OK** to save the changes. If the command prompt is open, restart it for the following step.

Step 5: Verify Python Was Installed on Windows

The first way to verify that Python was installed successfully is through the command line. Open the command prompt and run the following command:

```
python --version
```

The output shows the installed Python version.

The second way is to use the [GUI](#) to verify the Python installation. Follow the steps below to run the Python interpreter or IDLE:

1. Navigate to the directory where Python was installed on the system.

2. Double-click *python.exe* (the Python interpreter) or IDLE.

3. The interpreter opens the command prompt and shows the following window:

Running IDLE opens Python's built-in IDE:

In both cases, the installed Python version shows on the screen, and the editor is ready for use.

Step 6: Verify PIP Was Installed

To verify whether PIP was installed, enter the following command in the command prompt:

```
pip --version
```

If it was installed successfully, you should see the PIP version number, the executable path, and the Python version:

PIP has not been installed yet if you get the following output:

```
'pip' is not recognized as an internal or external command,  
Operable program or batch file.
```

If an older version of Python is installed or the PIP installation option is disabled during installation, PIP will not be available. To install PIP, see our article [How to Install PIP on Windows](#).

Step 7: Install virtualenv (Optional)

Python software packages install system-wide by default. Consequently, whenever a single project-specific package is changed, it changes for all your Python projects.

The **virtualenv** package enables making isolated local virtual environments for Python projects. Virtual environments help avoid package conflicts and enable choosing specific package versions per project.

To install **virtualenv**, run the following command in the command prompt:

```
pip install virtualenv
```

Wait for the installation to complete. Once done, it is installed on the system and available for use.