

#### 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.

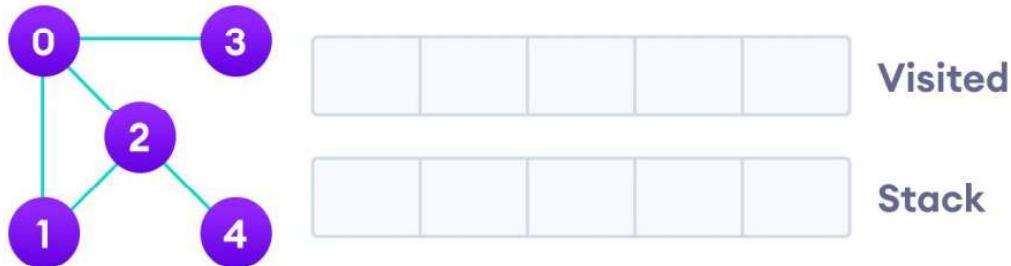
### **Depth First Search (DFS) Algorithm**

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.

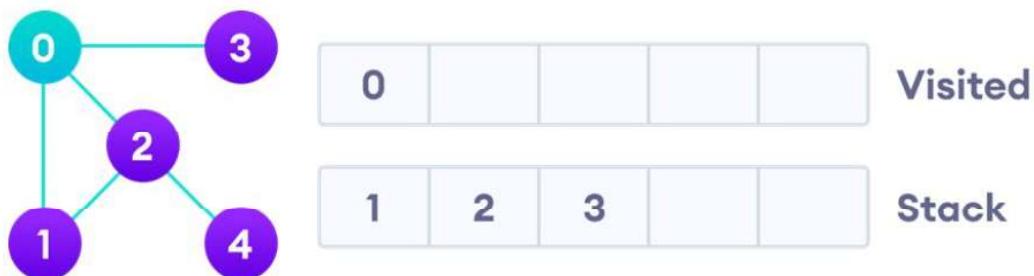
- STEP 1: Start by putting any one of the graph's vertices on top of a stack (acts as source node of DFS).

- 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



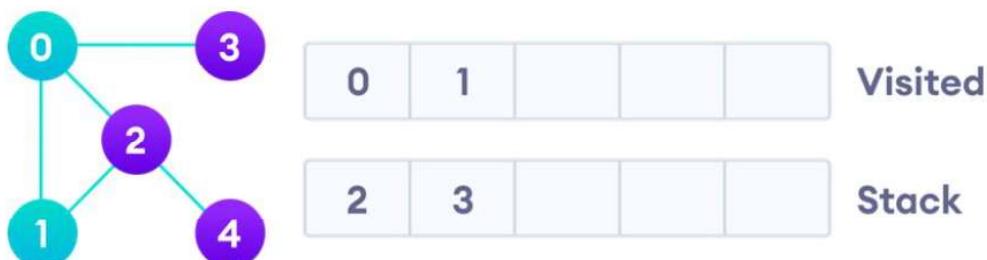
We start from vertex 0, the DFS algorithm starts by putting it in the Visited list and putting all its adjacent vertices in the stack.



Visit the element and put it in the visited list

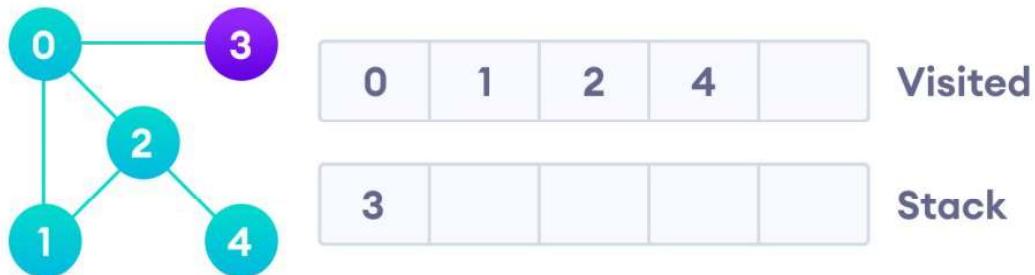
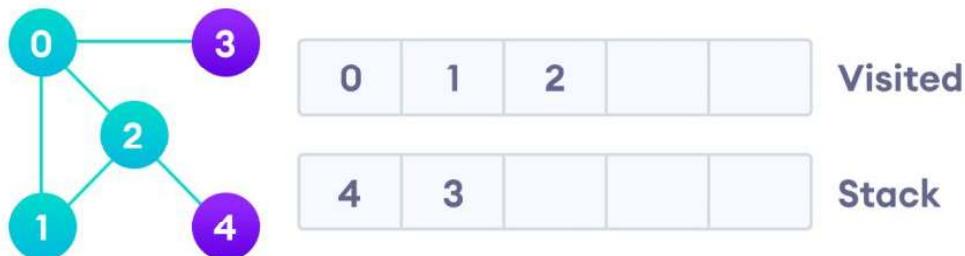
Next, we visit the element at the top of stack i.e. 1 and go to its adjacent nodes.

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?