1. Title of Assignment:

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. Prerequisite:

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

3. Objective:

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

- To understand Uninformed Search Strategies.
- 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.

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

5. Software and Hardware Requirement:

Open Source C++ Programming tool like G++/GCC, Python, Java and Ubuntu.

6. Relevant Theory / Literature Survey:

Uninformed (Blind search) search

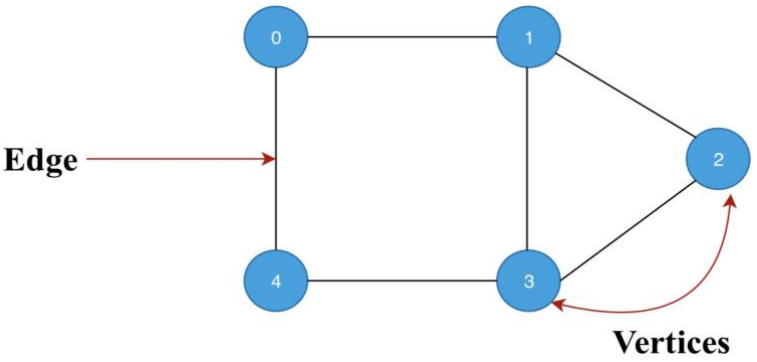
- Uninformed search is a class of general-purpose search algorithms which operates in brute force-way.
- It examines each node of the tree until it achieves the goal node.

- 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 thatconnect the vertices are called edges. Pictorial Representation of Graph.

Fig: Pictorial Representation of Graph



Types of Graphs

- Undirected Graph:- Directions are not given to edges
- Directed Graph:- Directions are given to edges .

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 thechild 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 visitsthe 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

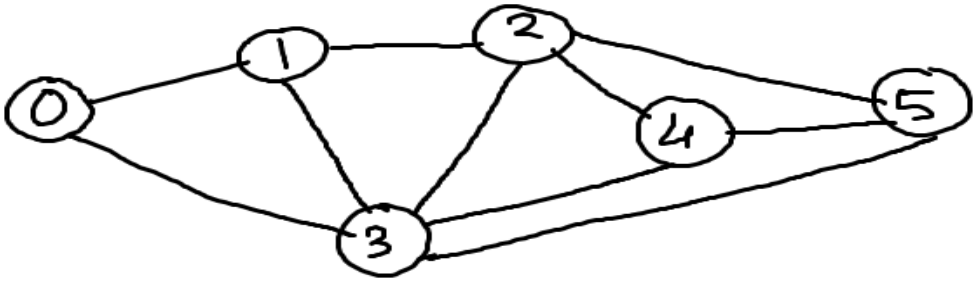
- Depth-first search isa recursive algorithm for traversing a tree or graph data structure.
- It is called the depth-first search because it starts from the root node and follows each pathto its greatest depth node before moving to the next path.
- DFS uses a stack data structure for its implementation.
- 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.

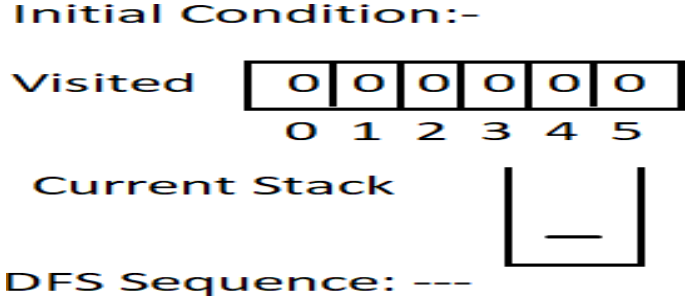
Depth First Search (DFS) Algorithm

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.

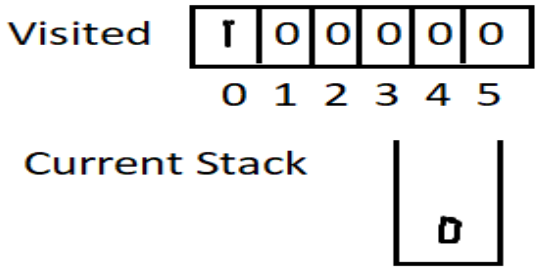
- STEP 1: Start by putting any one of the graph's vertices on top of a stack (acts as source node of DFS).
- STEP 2: Take the top item of the stack and set its visited as 1.
- STEP 3: Create a list of that vertex's adjacent nodes. Add the ones whose visited is 0 to the top of stack.
- STEP 4: Keep repeating steps 2 and 3 until the stack is empty.

An example which explains DFS Algorithm





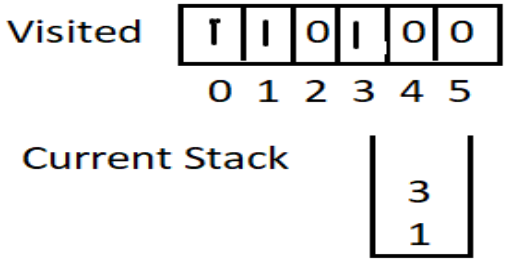
Step 1:
Start with Node 0
Node 0 is pushed into stack and its visited value set to 1



DFS Sequence: ---

Step 2:

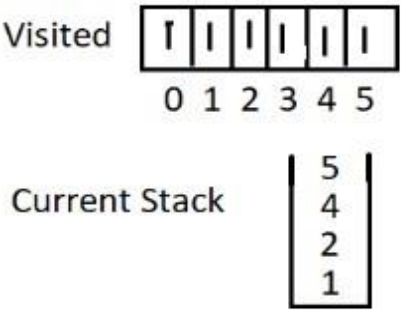
Node 0 is popped from stack and its adjacent nodes 1 and 3 are pushed into stack and visited value set to 1



DFS Sequence: 0

Step 3:

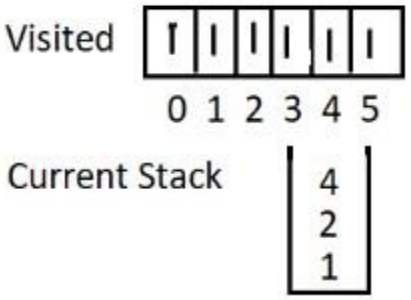
Node 3 is popped from stack and its adjacent nodes 2,4 and 5 are pushed into stack because their visited value is 0,
Set visited value to 1.



DFS Sequence: 0 , 3

Step 4:

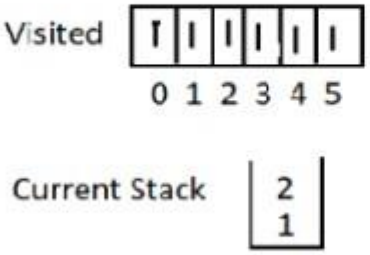
Node 5 is popped from stack. No adjacent node to add in stack



DFS Sequence: 0, 3, 5

Step 5:

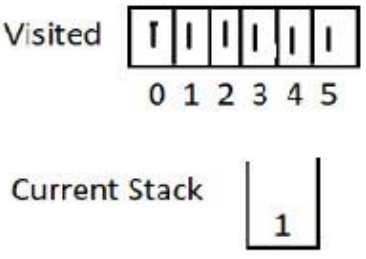
Node 4 is popped from stack.
Adjacent Nodes visited value is already 1 so no need to push it into stack.



DFS Sequence: 0,3,5,4

Step 6:

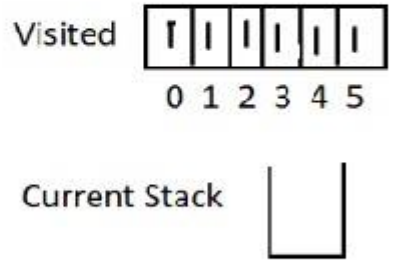
Node 2 is pooped from Stack, Adjacent Nodes Visited Value is Already 1 , So No Need to Push it into Stack



DFS Sequence: 0,3,5,4,2

Step 7:

Node 1 is pooped from stack as no adjacent nodes are remain unvisited.
No Nodes are pushed into stack.
As stack became empty We can stop.



DFS Sequence: 0,3,5,4 ,2 ,1

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.

Completeness: DFS search algorithm is complete within finite state space as it will expand every node within a limited search tree.

Time Complexity: Time complexity of DFS will be equivalent to the node traversed by the algorithm. It is given by:

$T(n)= 1+ n^2+ n^3 +.....+ n^m=O(n^m)$

Where, m= maximum depth of any node and this can be much larger than d (Shallowest solution depth)

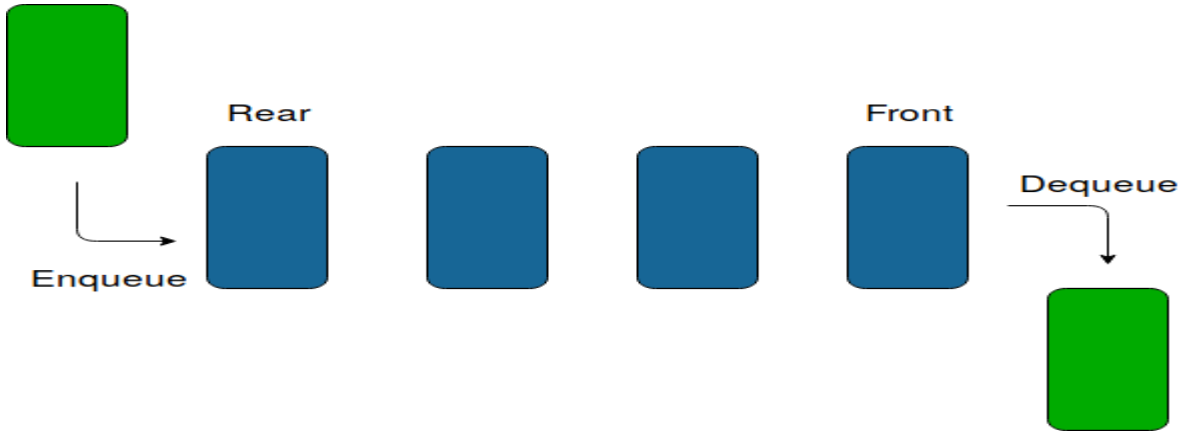
Space Complexity: DFS algorithm needs to store only single path from the root node, hence space complexity of DFS is equivalent to the size of the fringe set, which is **O(bm)**.

Optimal: DFS search algorithm is non-optimal, as it may generate a large number of steps or high cost to reach to the goal node.

BFS

Queues Definition

A Queue is a linear structure which follows a particular order in which the operations areperformed. 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 nodesat 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.

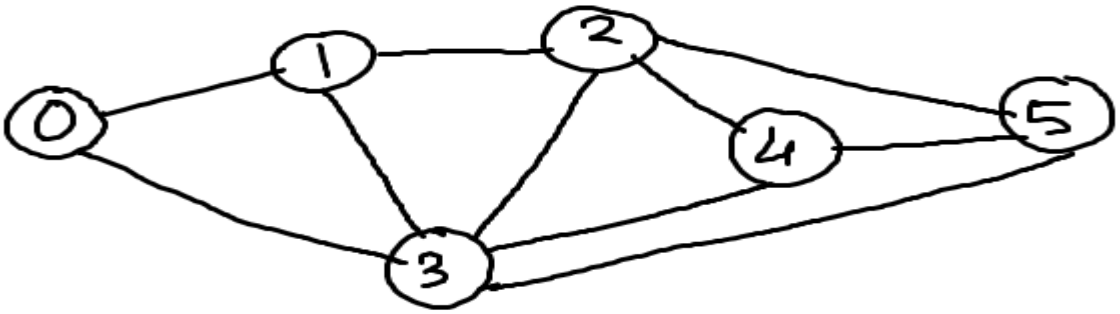
STEP 3: Dequeue a node N from the queue and update its visited as 1.

STEP 4: Enqueue all the neighbours of node N which are not present in the queue and whose visited is 0.

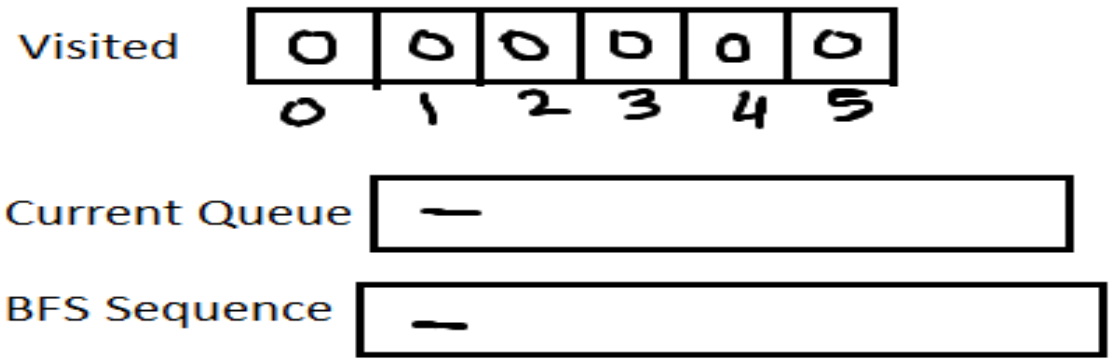
STEP 5: Repeat steps 3 and 4 until the queue is empty.

STEP 6: EXIT

An example which explains BFS Algorithm

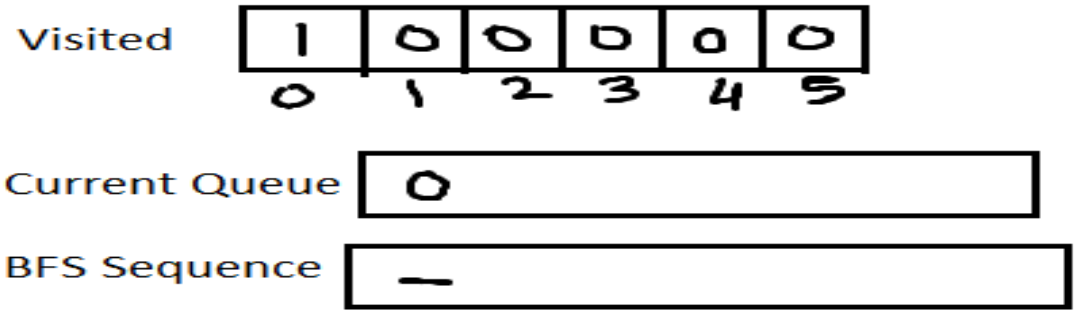


Initial Condition:-



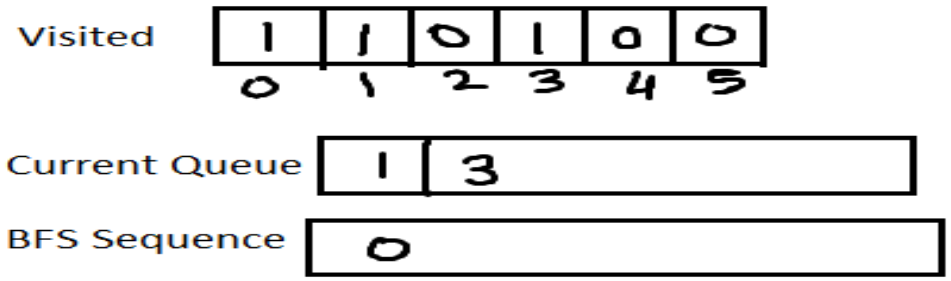
Step 1:-

Node '0' is selcted as source node of BFS So Node '0' is enqueued to Queue



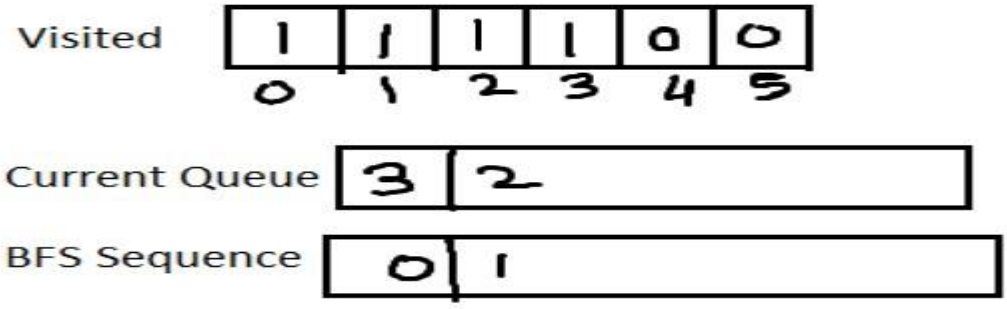
Step 2:-

Node '0' is dequeued from Queue and it's adjacent nodes are enqueued into queue i.e. 1,3 by changing visited value to '1'



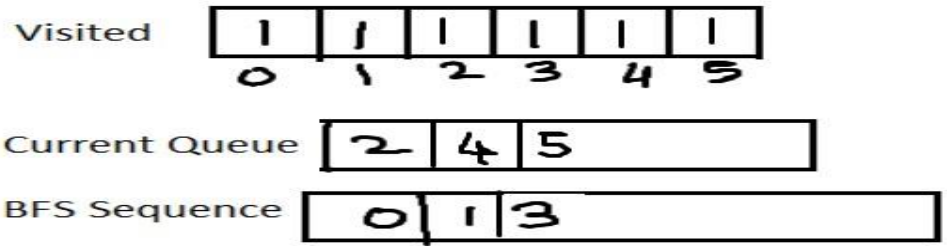
Step 3:-

Node '1' is dequeued from Queue and it's adjacent nodes are enqueued into queue i.e.2,3 but only node 2 is having visited value 0 but node 3 is already having visited value 1. we only enqueued node 2 in queue.



Step 4:-

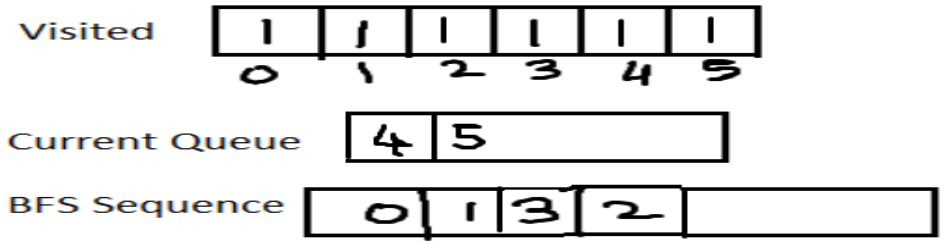
Node '3 ' is dequeued from Queue and it's adjacent nodes are enqueued into queue i.e. 4,5 and set their visited value to 1



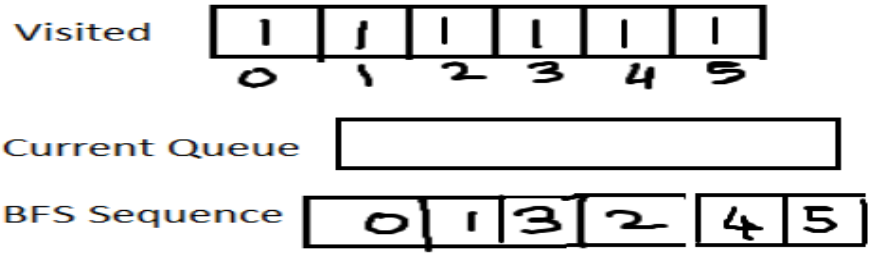
Step 5:-

Step 6:-

Node 2 is dequeued from Queue and it's adjacent nodes are already having visited value 1 so No enqueued node 3,4,5 in Queue.



Simillarly Node 4,5 are dequeued from Queue , No Nodes are enqueued in Queue.
Once Queue is empty u can stop.



As queue is empty , BFS is done on node '0'
Nodes visited in BFS sequence : 0, 1, 3, 2, 4, 5

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 neighbornodes.

- 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 orDepth 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 DepthFirst Traversal to find all nodes reachable from a given node.

Time Complexity: Time Complexity of BFS algorithm can be obtained by the number of nodes traversed in BFS until the shallowest Node. Where the d= depth of shallowest solution and b is a node at every state.

$$T(b) = 1+b^2+b^3+.....+ b^d= O(b^d)$$

Space Complexity: Space complexity of BFS algorithm is given by the Memory size of frontier which is $O(b^d)$.

Completeness: BFS is complete, which means if the shallowest goal node is at some finite depth, then BFS will find a solution.

Optimality: BFS is optimal if path cost is a non-decreasing function of the depth of the 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.

7. 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?

8. 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.