Breadth-First Search Algorithm

Objectives:

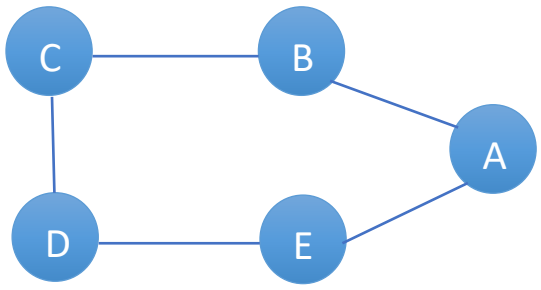
Graph traversal is a search technique to find a vertex in a graph. The breadth-first search or BFS algorithm is used to search a tree or graph data structure for a node that meets a set of criteria. It begins at the root of the tree or graph and investigates all nodes at the current depth level before moving on to nodes at the next depth level. You can solve many problems in graph theory via the breadth-first search. For example, finding the shortest path between two vertices a and b is determined by the number of edges. In a flow network, the Ford–Fulkerson method is used to calculate the maximum flow and when a binary tree is serialized/deserialized instead of serialized in sorted order, the tree can be reconstructed quickly.

Example of Breadth-First Search Algorithm:

In a tree-like structure, graph traversal requires the algorithm to visit, check, and update every single un-visited node. The sequence in which graph traversals visit the nodes on the graph categorizes them.

The BFS algorithm starts at the first starting node in a graph and travels it entirely. After traversing the first node successfully, it visits and marks the next non-traversed vertex in the graph.

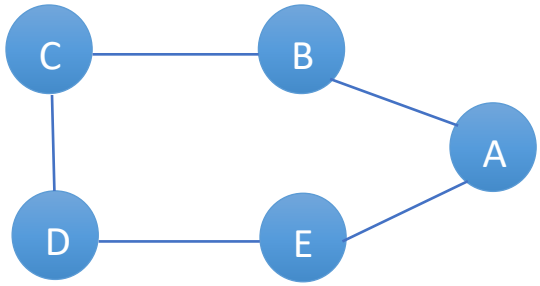
Step 1: In the graph, every vertex or node is known. First, initialize a queue.



Source Node

Queue->

Step 2: In the graph, start from source node A and mark it as visited.

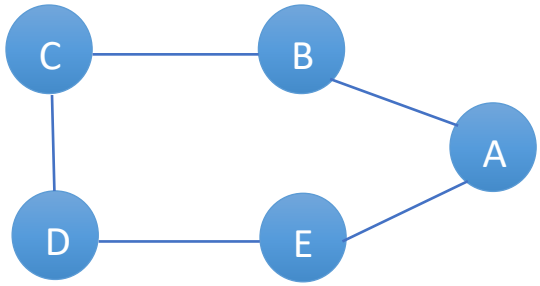


Source Node

Queue->

A					

Step 3: Then you can observe B and E, which are unvisited nearby nodes from A. You have two nodes in this example, but here choose B, mark it as visited, and enqueue it alphabetically.

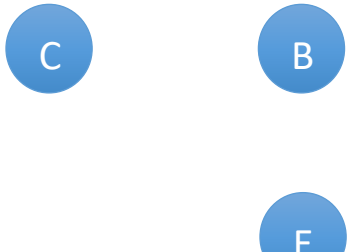


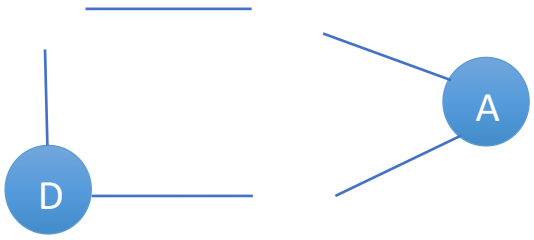
Source Node

Queue->

B	A				

Step 4: Node E is the next unvisited neighboring node from A. You enqueue it after marking it as visited.

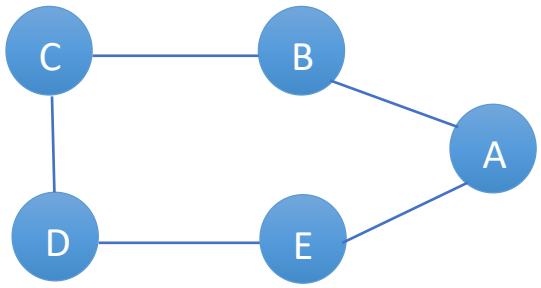




Queue->

E	B	A			

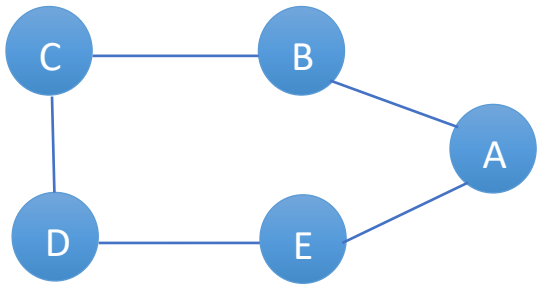
Step 5: A now has no unvisited nodes in its immediate vicinity. As a result, you dequeue and locate A.



Queue->

E	B				

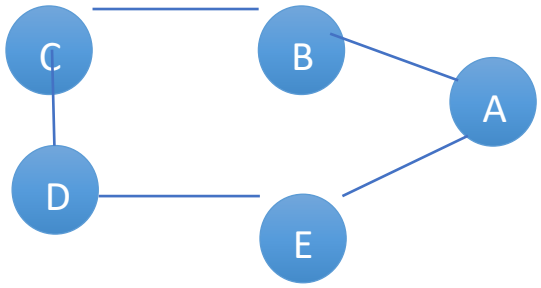
Step 6: Node C is an unvisited neighboring node from B. You enqueue it after marking it as visited.



Source Node

Queue-> E | B | C | | | |

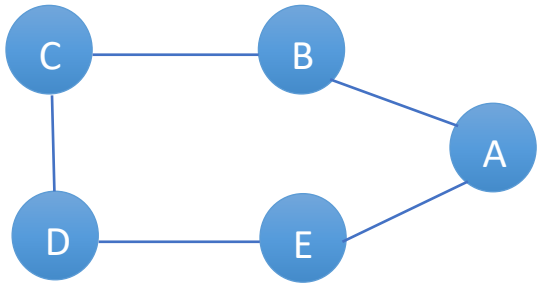
Step 7: Node D is an unvisited neighboring node from C. You enqueue it after marking it as visited.



Source Node

Queue-> E | B | C | D | | |

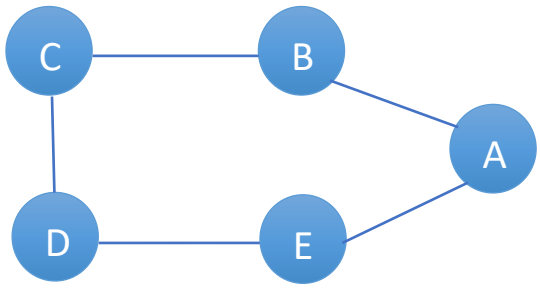
Step 8: If all of D's adjacent nodes have already been visited, remove D from the queue.



Source Node

Queue-> E | B | C | | | |

Step 9: Similarly, all nodes near E, B, and C nodes have already been visited; therefore, you must remove them from the queue.



Source Node

Queue->

Step 10: Because the queue is now empty, the BFS traversal has ended.

Code in C:

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

int twodimarray[10][10],queue[10],visited[10],n,i,j,front=0,rear=-1;

void breadthfirstsearch(int vertex) // breadth first search function

{

   for (i=1; i<=n; i++)
```

```
        if(twodimarray[vertex][i] && !visited[i])

            queue[++rear]=i;

    if(front<=rear)

    {

        visited[queue[front]]=1;

        breadthfirstsearch(queue[front++]);

    }

}

int main()

{

    int x;

    printf("\n Enter the number of vertices:");

    scanf("%d",&n);

    for (i=1; i<=n; i++)

    {
```

```
    queue[i]=0;
    visited[i]=0;
}
printf("\n Enter graph value in form of matrix:\n");
for (i=1; i<=n; i++)
    for (j=1; j<=n; j++)
        scanf("%d",&twodimarray[i][j]);
printf("\n Enter the source node:");
scanf("%d",&x);
breadthfirstsearch(x);
printf("\n The nodes which are reachable are:\n");
for (i=1; i<=n; i++)
    if(visited[i])
        printf("%d\t",i);
```

```
    else

        printf("\n Breadth first search is not possible");

  getch();

}
```

Complexity Analysis:

In this technique, each neighboring vertex is inserted into the queue if it is not visited. This is done by looking at the edges of the vertex. Each visited vertex is marked visited once we visit them hence, each vertex is visited exactly once, and all edges of each vertex are checked. So the complexity of BFS is V + E

Pseudocode for BFS:

```
create a queue Q
v.visited = true
Q.push(v)
while Q is non-empty
  remove the head u of Q
  mark and enqueue all (unvisited) neighbours of u
```

Since we are only iterating over the graph's edges and vertices only once, hence the time complexity for the algorithms is linear O(V+E).

Auxiliary Space Taken is O(V) at worst case.

Advantages of BFS:

1. The solution will definitely found out by BFS If there is some solution.
2. BFS will never get trapped in a blind alley, which means unwanted nodes.
3. If there is more than one solution then it will find a solution with minimal steps.

Disadvantages Of BFS:

1. Memory Constraints As it stores all the nodes of the present level to go for the next level.
2. If a solution is far away then it consumes time.

Application Of Breadth-First Search Algorithm:

The breadth-first search algorithm has the following applications:

For Unweighted Graphs, You Must Use the Shortest Path and Minimum Spacing Tree.

The shortest path in an unweighted graph is the one with the fewest edges. You always reach a vertex from a given source using the fewest amount of edges when utilizing breadth-first. In unweighted graphs, any spanning tree is the Minimum Spanning Tree, and you can identify a spanning tree using either depth or breadth-first traversal.

Peer to Peer Network

Breadth-First Search is used to discover all neighbor nodes in peer-to-peer networks like BitTorrent.

Crawlers in Search Engine

Crawlers create indexes based on breadth-first. The goal is to start at the original page and follow all of the links there, then repeat. Crawlers can also employ Depth First Traversal. However, the benefit of breadth-first traversal is that the depth or layers of the created tree can be limited.

Social Networking Websites

You can use a breadth-first search to identify persons within a certain distance 'd' from a person in social networks up to 'd's levels.

GPS Navigation System

To find all nearby locations, utilize the breadth-first search method.

Broadcasting Network

A broadcast packet in a network uses breadth-first search to reach all nodes.

Garbage Collection

Cheney's technique uses the breadth-first search for duplicating garbage collection. Because of the better locality of reference, breadth-first search is favored over the Depth First Search algorithm.

Cycle Detection in Graph

Cycle detection in undirected graphs can be done using either Breadth-First Search or Depth First Search. BFS can also be used to find cycles in a directed graph.

Identifying Routes

To see if there is a path between two vertices, you can use either Breadth-First or Depth First Traversal.

Finding All Nodes Within One Connected Component

To locate all nodes reachable from a particular node, you can use either Breadth-First or Depth First Traversal.