**Experiment-No.4**

**Objective: Write a program to implement breadth first search**

| **Scheduled Date:** | **Compiled Date:** | **Submitted Date:** |
| --- | --- | --- |
| 24 Sep 2024 | 24 Sep 2024 | 25- Sep 2024 |

**Description of BFS (BreaPdth-First Search)**

Breadth-First Search (BFS) is a fundamental graph traversal algorithm that explores vertices in a breadthward motion, meaning it systematically visits all the neighboring nodes at the present depth before moving on to nodes at the next depth level.

### 1. General Overview:

* **Type**: BFS is a *graph traversal* algorithm, and it works on both trees and graphs.
* **Purpose**: BFS is primarily used to explore a graph's nodes in layers, starting from a given source node.
* **Strategy**: It explores all neighbors (i.e., nodes directly connected by an edge) of a node before moving on to the neighbors of those neighbors.
* **Queue-based**: BFS uses a queue data structure to keep track of which node to explore next, ensuring a first-in-first-out (FIFO) order.

### 2. How BFS Works:

1. **Start**: The algorithm starts from a source node and adds it to the queue.
2. **Visit Neighbors**: It dequeues the current node, visits its unvisited neighbors (i.e., nodes that are directly connected), and adds them to the queue.
3. **Mark as Visited**: Once a node is visited, it is marked as "visited" to avoid revisiting.
4. **Repeat**: The process repeats until all nodes at the current depth level are explored, then it moves on to the next depth level, continuing until the queue is empty.

### 

### 3. Key Features:

* **Layer-by-Layer Exploration**: Nodes are visited in increasing order of their distance from the source node.
* **Shortest Path Guarantee**: In an unweighted graph, BFS guarantees the shortest path (in terms of the number of edges) from the source node to any other node.
* **Graph Types**: BFS works for both directed and undirected graphs, and can be adapted for various scenarios, such as finding connected components in an undirected graph.

### 4. Time and Space Complexity:

* **Time Complexity**: O(V + E), where *V* is the number of vertices and *E* is the number of edges.
* **Space Complexity**: O(V), as it needs space for the queue and the visited list.

**Algorithm of BFS (Breadth-First Search)**

1. **Initialize:**

* Create an empty list visited to store the nodes that have been visited.
* Create an empty queue queue to keep track of nodes to be processed.

1. **Start BFS:**

* Add the starting node to the visited list.
* Add the starting node to the queue.

1. **Process the Queue:**

* While the queue is not empty:
  1. Remove (dequeue) the first node from the queue.
  2. Print or process the current node.
  3. For each neighbor of the current node:
     + If the neighbor has not been visited:
       - Add the neighbor to the visited list.
       - Add the neighbor to the queue.

**4. End:**

* When the queue becomes empty, the BFS traversal is complete.

**Python Code for BFS (Breadth-First Search)**

**graph={**

**'A':['B','C'],**

**'B':['D','E'],**

**'C':['F'],**

**'D':[],**

**'E':['F'],**

**'F':[]**

**}**

**visited=[]**

**queue=[]**

**def bfs(visited,graph,node):**

**visited.append(node)**

**queue.append(node)**

**while queue:**

**m=queue.pop(0)**

**print(m,end=" ")**

**for n in graph[m]:**

**if n not in visited:**

**visited.append(n)**

**queue.append(n)**

**bfs(visited,graph,'A')**