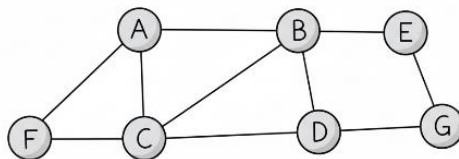


Lab Report -1

BFS: BFS stands for **Breadth-First Search**.

It's an algorithm used for **traversing** (visiting all the nodes) or **searching** a **tree** or **graph** data structure. The key idea of BFS is that it explores all the nodes at the current depth level before moving on to the nodes at the next depth level. It works on only unweighted graph.

Graph Search Example:



Find the path from A to G, where A is the source node.

1. Color

- **White (w)** → not visited
- **Gray (g)** → discovered, waiting in queue
- **Black (b)** → fully processed

2. Distance

How far a node is from the source (in number of edges).

3. Queue

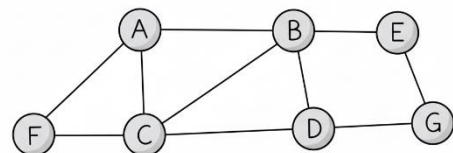
Shows the real BFS process.

4. Parent

Which node discovered this node.

Initialize:

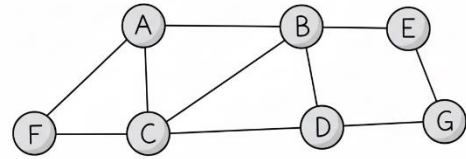
| | A | B | C | D | E | F | G |
|----------|---|---|---|---|---|---|---|
| Color | w | w | w | w | w | w | w |
| Distance | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Queue | | | | | | | |
| Parent | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



Step-1:

| | A | B | C | D | E | F | G |
|-----------------|----|---|---|---|---|---|---|
| Color | g | w | w | w | w | w | w |
| Distance | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Queue | A | | | | | | |
| Parent | -1 | 0 | 0 | 0 | 0 | 0 | 0 |

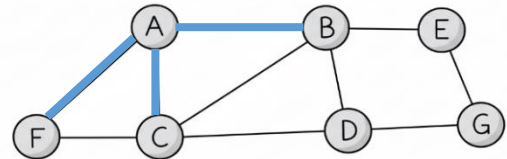
Source Node=A



Step-2:

| | A | B | C | D | E | F | G |
|-----------------|----|---|---|---|---|---|---|
| Color | b | g | g | w | w | g | w |
| Distance | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| Queue | A | B | C | F | | | |
| Parent | -1 | A | A | 0 | 0 | A | 0 |

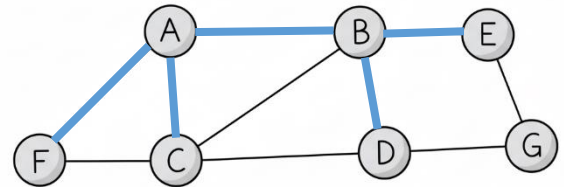
Source Node=A & Selected Vertex=A



Step-3:

| | A | B | C | D | E | F | G |
|-----------------|----|---|---|---|---|---|---|
| Color | b | b | g | g | g | g | w |
| Distance | 0 | 1 | 1 | 2 | 2 | 1 | 0 |
| Queue | A | B | C | F | D | E | |
| Parent | -1 | A | A | B | B | A | 0 |

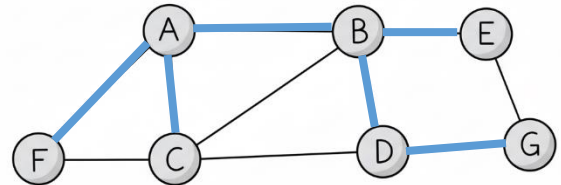
Source Node=A & Selected Vertex=B



Step-4:

| | A | B | C | D | E | F | G |
|-----------------|----|---|---|---|---|---|---|
| Color | b | b | b | b | w | b | g |
| Distance | 0 | 1 | 1 | 2 | 2 | 1 | 3 |
| Queue | A | B | C | F | D | E | G |
| Parent | -1 | A | A | B | B | A | D |

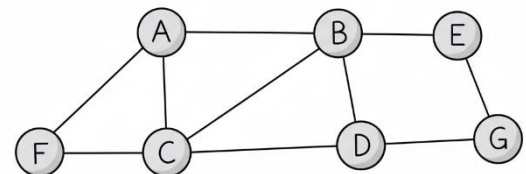
Source Node=A & Selected Vertex=D



Step-5:

| | A | B | C | D | E | F | G |
|-----------------|----|---|---|---|---|---|---|
| Color | b | b | b | b | b | b | b |
| Distance | 0 | 1 | 1 | 2 | 2 | 1 | 3 |
| Queue | A | B | C | F | D | E | G |
| Parent | -1 | A | A | B | B | A | D |

Source Node=A



➤ The Shortest path from A to G Node:

A → B → D → G

Pseudocode:

```
BFS(G, s) {  
    for each u in V {           // initialization  
        color[u] = white  
        d[u] = infinity  
        pred[u] = null  
    }  
    color[s] = gray             // initialize source s  
    d[s] = 0  
    Q = {s}                    // put s in the queue  
    while (Q is nonempty) {  
        u = Q.Dequeue()        // u is the next to visit  
        for each v in Adj[u] {  // explore all neighbors of u  
            if (color[v] == white) { // if neighbor v undiscovered  
                color[v] = gray    // ...mark it discovered  
                d[v] = d[u] + 1    // ...set its distance  
                pred[v] = u        // ...and its predecessor  
                Q.Enqueue(v)       // ...put it in the queue  
            }  
        }  
        color[u] = black         // we are done with u  
    }  
}
```