ARTIFICIAL INTELLIGENCE AND PROGRAMMING ROBOT

PRACTICAL NO 7

027_Abhishek_Ojha

Practical No. 7

Aim:

Write a program to implement bfs algorithm for a given standard problem.

Theory:

Breadth First Traversal or Breadth First Search is a recursive algorithm for searching all the vertices of a graph or tree data structure.

A standard BFS implementation puts each vertex of the graph into one of two categories:

1. Visited
2. Not Visited

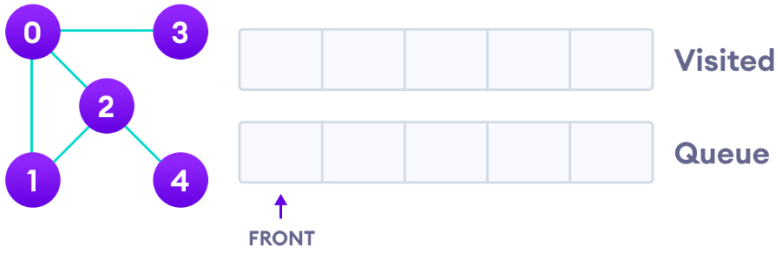
The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

The algorithm works as follows:

1. Start by putting any one of the graph's vertices at the back of a queue.
2. Take the front item of the queue and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.
4. Keep repeating steps 2 and 3 until the queue is empty.

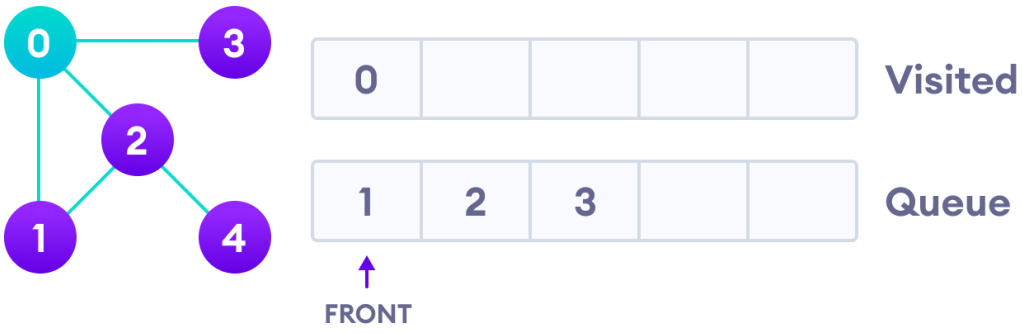
The graph might have two different disconnected parts so to make sure that we cover every vertex, we can also run the BFS algorithm on every node.

Let's see how the Breadth First Search algorithm works with an example. We use an undirected graph with 5 vertices.



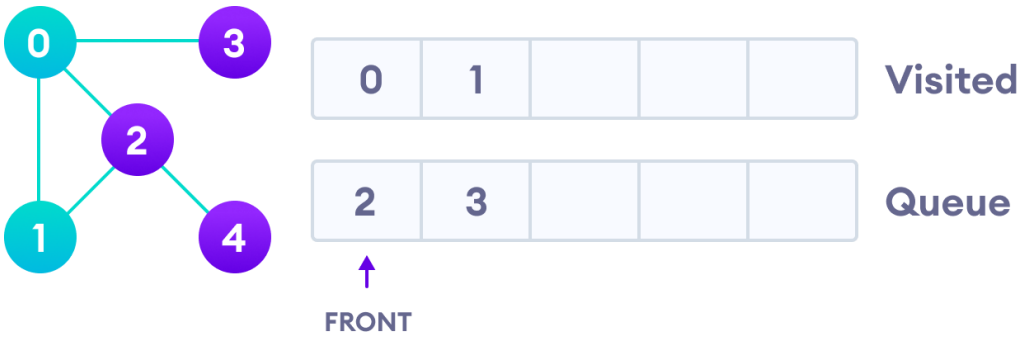
Undirected graph with 5 vertices

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



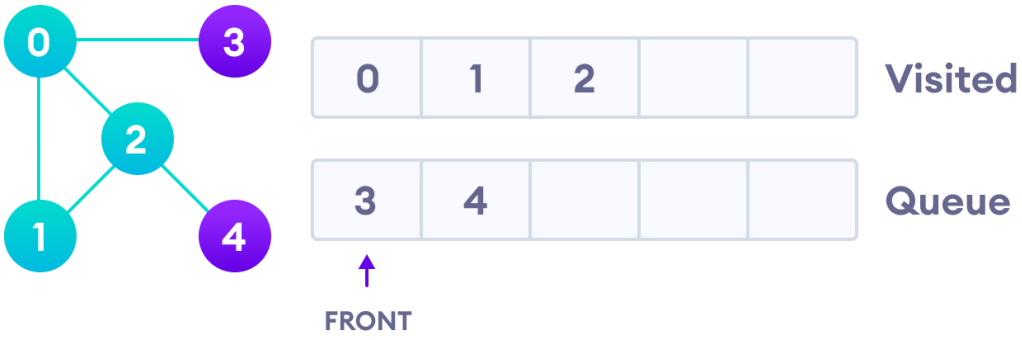
Visit start vertex and add its adjacent vertices to queue

Next, we visit the element at the front of queue i.e. 1 and go to its adjacent nodes. Since 0 has already been visited, we visit 2 instead.

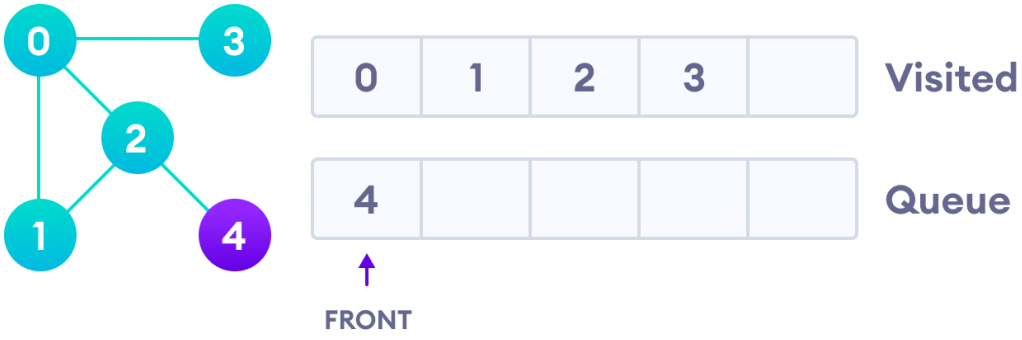


Visit the first neighbour of start node 0, which is 1

Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the back of the queue and visit 3, which is at the front of the queue.



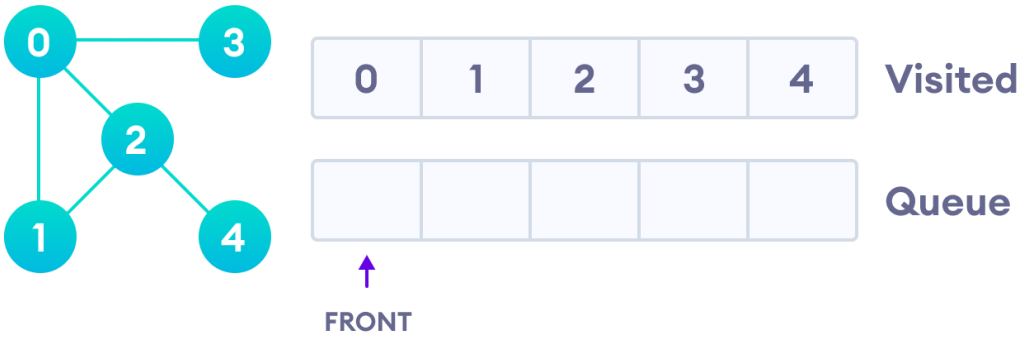
Visit 2 which was added to queue earlier to add its neighbours



4 remains in the queue

Only 4 remains in the queue since the only adjacent node of 3 i.e. 0 is already visited.

We visit it.



Visit last remaining item in the stack to check if it has unvisited neighbors

Since the queue is empty, we have completed the Breadth First Traversal of the graph.

Code:

```
package bfs;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.Queue;

public class BFS {

   private final Queue<Node> queue;
   static ArrayList<Node> nodes = new ArrayList<Node>();

   static class Node {

      int data;
      boolean visited;
      List<Node> neighbours;

      Node(int data) {
         this.data = data;
         this.neighbours = new ArrayList<>();
      }

      public void addneighbours(Node neighbourNode) {
         this.neighbours.add(neighbourNode);
      }

      public List<Node> getNeighbours() {
         return neighbours;
      }

      public void setNeighbours(List<Node> neighbours) {
         this.neighbours = neighbours;
      }
   }

//    constructor
   public BFS() {
      queue = new LinkedList<>();
   }

   public void bfs(Node node) {
      queue.add(node);
      node.visited = true;
```

```
        while (!queue.isEmpty()) {
            Node element = queue.remove();
            System.out.print(element.data + "\t");
            List<Node> neighbours = element.getNeighbours();
            for (int i = 0; i < neighbours.size(); i++) {
                Node n = neighbours.get(i);
                if (n != null && !n.visited) {
                    queue.add(n);
                    n.visited = true;

                }
            }

        }
    }

    public static void main(String[] args) {
        Node node40 = new Node(40);
        Node node10 = new Node(10);
        Node node20 = new Node(20);
        Node node30 = new Node(30);
        Node node60 = new Node(60);
        Node node50 = new Node(50);
        Node node70 = new Node(70);

        node40.addneighbours(node10);
        node40.addneighbours(node20);
        node10.addneighbours(node30);
        node20.addneighbours(node10);
        node20.addneighbours(node30);
        node20.addneighbours(node60);
        node20.addneighbours(node50);
        node30.addneighbours(node60);
        node60.addneighbours(node70);
        node50.addneighbours(node70);
        System.out.println("The BFS traversal of the graph is ");
        BFS bfsExample = new BFS();
        bfsExample.bfs(node40);
    }

}
```

Output:

```
The BFS traversal of the graph is
40      10      20      30      60      50      70
```

Conclusion: We successfully implemented Breath first search.