Breadth first search(BFS) 1

Write a Java Program to print all nodes reachable from a given starting node in a digraph using BFS method

Input Format

4 5 1 3 0 2 2 3 3 0 0 3 2

Constraints

positive vertex values only

Output Format

Enter the number of vertices : 4 Enter the number of edges : 5 Enter source : 1 Enter destination : 3 Enter source : 0 Enter destination : 2 Enter source : 2 Enter destination : 3 Enter source : 3 Enter destination : 0 Enter source : 0 Enter destination : 3 Enter Start Vertex for BFS : 2 BFS of graph : 2 3 0

Sample Input 0

```
4
5
1
3
0
2
2
3
3
0
0
3
2
```

Sample Output 0

```
BFS of graph : 2 3 0
```



```
1 import java.io.*;
2 import java.util.*;
```

```java
import java.text.*;
import java.math.*;
import java.util.regex.*;
class Gnode {
    Gnode next;
    int vertex;
}
public class Solution {
    Gnode graph[ ];
    boolean visited[ ];
    int queue[];
    int numVertices;
    int front;
    int rear;
    public Solution(int n) {
        graph = new Gnode[n];
        visited = new boolean[n];
        queue = new int[n];
        numVertices = n;
        front = -1;
        rear = -1;

    }
    void insertQueue(int vertex) {
        if(rear == numVertices - 1){}
        //System.out.printf("Queue Overflow.\n");
        else {
            if(front == -1)
                front = 0;
            rear = rear + 1;
            queue[rear] = vertex ;
        }
    }

    boolean isEmptyQueue() {
        if(front == -1 || front > rear)
            return true;
        else
            return false;
    }

    int deleteQueue() {
        int deleteItem;
        if(isEmptyQueue()) {
            //System.out.printf("Queue Underflow\n");
            return -1;
        }
        deleteItem = queue[front];
        front = front + 1;
        return deleteItem;
    }
    void Bfs(int v) {
        int w;
        insertQueue(v);
        while(!isEmptyQueue()) {
            v = deleteQueue( );
            System.out.printf(" %d",v);
            visited[v] = true;
            Gnode g = graph[v];
            for( ; g != null; g = g.next) {
                w = g.vertex;
                if(visited[w] == false) {
                    insertQueue(w);
                    visited[w] = true;
                }
            }
        }
    }
    public static void main(String []args) {
        int n, e, i, s, d, v;
        Gnode q, p;
        Scanner sc = new Scanner(System.in);

```

```
        //System.out.printf("Enter the number of vertices : ");
        n = sc.nextInt();
        //System.out.printf("Enter the number of edges : ");
        e = sc.nextInt();
        Solution g = new Solution(n);
        for(i=1;i<=e;i++) {
            //System.out.printf("Enter source : ");
            s = sc.nextInt();
            //System.out.printf("Enter destination : ");
            d = sc.nextInt();
            q = new Gnode();
            q.vertex = d;
            q.next = null;
            if(g.graph[s] == null)
                g.graph[s]=q;
            else {
                p=g.graph[s];
                while(p.next != null)
                    p = p.next;
                p.next = q;
            }
        }
        for(i = 0;i < n;i++)
            g.visited[i] = false;
        //System.out.printf("Enter Start Vertex for BFS : ");
        v = sc.nextInt();
        System.out.printf("BFS of graph :");
        g.Bfs(v);
        System.out.printf("\n");
    }
}
```

Line: 1 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
4
5
1
3
0
2
2
3
3
0
0
3
2
```

Your Output (stdout)

```
BFS of graph : 2 3 0
```

Expected Output

```
BFS of graph : 2 3 0
```