import java.util.LinkedList;

import java.util.Queue;

import java.util.Scanner;

import java.util.Stack;

class graph {

final LinkedList<Integer> [] adjency;

final int vertices;

final int edges;

Scanner sc=new Scanner(System.in);

public graph()

{

System.out.println("Enter number of vertices");

vertices=sc.nextInt();

System.out.println("Enter number of edges");

edges=sc.nextInt();

adjency=new LinkedList[vertices+1];

for (int i = 0; i < vertices+1; i++) {

adjency[i]=new LinkedList<>();

}

}

public void insert\_edge()

{

for (int i = 1; i <=edges; i++) {

if (i==1) System.out.println("enter starting and destination of "+i+"st "+"edge");

else if (i==2||i==3)

System.out.println("enter starting and destination of "+i+"nd "+"edge");

else System.out.println("enter starting and destination of "+i+"th "+"edge");

int s=sc.nextInt();

int d=sc.nextInt();

adjency[s].add(d);

adjency[d].add(s);

}

}

public void bfs(int source)

{

boolean[]isvisited=new boolean[vertices+1];

int parent\_nodes[]=new int[vertices+1];

Queue<Integer> q=new LinkedList<Integer>();

q.add(source);

isvisited[source]=true;

parent\_nodes[source]=-1;

while (!q.isEmpty())

{

int p=q.poll();

System.out.print(p+"-->");

for (int i : adjency[p]) {

if (!isvisited[i])

{

isvisited[i]=true;

q.add(i);

parent\_nodes[i]=p;

}

}

}

System.out.println("END");

}

public void print()

{

System.out.println("======================================================================");

for (int i = 0; i < vertices ; i++) {

if (adjency[i].size()>0)

{

System.out.print("vertex "+i+" connected to-->");

for (int j = 0; j < adjency[i].size(); j++) {

System.out.print(adjency[i].get(j)+"-->");

}

System.out.println(" ");

}

}System.out.println("======================================================================");

}

}

public class BFS\_DFS {

public static void main(String[] args) {

graph g=new graph();

g.insert\_edge();

g.print();

System.out.println("Enter source for BFS travel");

Scanner sc= new Scanner(System.in);

int a=sc.nextInt();

g.bfs(a);

System.out.println("Enter source for DFS travel");

int b= sc.nextInt();

g.dfs(b);

sc.close();

}

}