/\*Title :- To implement bfs and dfs

\* Roll number :3946

\*/

package adjacency;

import java.util.\*;

class node {

node next;

int data;

public node() {

next = null;

data = 0;

}

public node(int d) {

data = d;

next = null;

}

}

class graph {

Scanner sc = new Scanner(System.in);

node head[] = new node[20];

int v;// /no of vertices

int e;// no of edges

int a[][] = new int[10][10];

public graph() {

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

a[i][j] = -1;

}

}

}

void Bfs(int n) {

System.out.println("\n BFS Traversal:");

Queue<Integer> q = new LinkedList();

int v[] = new int[n];

for (int i = 0; i < n; i++)

v[i] = 0;

System.out

.println("\n Enter the home no. which you want to start with:");

int k = sc.nextInt();

v[k] = 1;

q.add(k);

while (!q.isEmpty()) {

int d = q.remove();

System.out.println(" House no." + (d) + "\t");

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

if (a[d][i] == 1 && v[i] == 0) {

q.add(i);

v[i] = 1;

}

}

}

}

/\*public void RecursiveDFS(node temp1,int visited[])

{

if(temp1==null)

return;

else

{

while(temp1!=null)

{

if(visited[temp1.data]==0)

{

visited[temp1.data]=1;

System.out.println(temp1.data);

RecursiveDFS(head[temp1.data],visited);

}

else

temp1=temp1.next;

}

}

}

public void callRDFS(int n)

{

System.out.println("Enter starting vertex: ");

int t=sc.nextInt();

node ptr=new node();

ptr=head[t];

int[] visited= new int[n];

for(int i=0;i<n;i++)

visited[i]=0;

visited[t]=1;

System.out.println(t);

RecursiveDFS(ptr,visited);

}

void Dfs(int n) {

Stack<Integer> s = new Stack();

int v[] = new int[n];

for (int i = 0; i < n; i++)

v[i] = 0;

System.out.println("\n Enter the house you want to start at:");

int k = sc.nextInt();

if (k < 0 && k > n) {

System.out.println("\n Re-Enter a node between 0 and " + n);

k = sc.nextInt();

}

s.push(k);

System.out.println("House no." + k);

v[k] = 1;

while (!s.isEmpty()) {

int flag = 0;

node ptr = new node();

int p = s.peek();

ptr = head[p];

while (ptr != null) {

if (v[ptr.data] != 1) {

s.push(ptr.data);

System.out.println("House no." + ptr.data);

v[ptr.data] = 1;

flag = 1;

break;

} else {

ptr = ptr.next;

}

}

if (flag == 0) {

s.pop();

}

}

}\*/

void creatematrix(int n) {

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

head[i] = null;

}

node temp = new node();

node temp1 = new node();

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

node ptr = new node();

ptr = head[i];

for (int j = 0; j < n; j++) {

if (i == j)

a[i][j] = 0;

if (a[i][j] == -1 && i != j) {

System.out

.println("\n Enter 1 if there is a route between house "

+ (i)

+ " and house "

+ (j)

+ " else Enter 0:");

a[i][j] = sc.nextInt();

a[j][i] = a[i][j];

}

if (a[i][j] == 1) {

temp = new node(j);

if (head[i] == null)

head[i] = temp;

else {

ptr = head[i];

while (ptr.next != null)

ptr = ptr.next;

ptr.next = temp;

}

}

}

}

}

void displaylist(int n) {

System.out.println("\n Displaying the adjacency list:");

node ptr = new node();

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

System.out.println("\n The houses connected to node " + (i + 1));

ptr = head[i];

if (head[i] == null)

System.out

.println("\n There is no lane connecting this node to other nodes");

while (ptr != null) {

System.out.print(" House no. " + (ptr.data + 1));

ptr = ptr.next;

}

System.out.print("\n");

}

}

void displaymatrix(int n) {

System.out.println("\n Displaying the adjacency Matrix:");

System.out.print("\t");

for (int i = 0; i < n; i++)

System.out.print("\tH" + (i + 1));

System.out.println();

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

System.out.print("\tH" + (i + 1));

for (int j = 0; j < n; j++) {

System.out.print("\t" + a[i][j]);

}

System.out.print("\n");

}

}

}

public class graph4 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n1, n2 = 0;

System.out

.println("\n Enter the number of houses where the man drops the newspaper:");

int n = sc.nextInt();

graph m = new graph();

do {

System.out

.println("\n1.Create Matrix\n2.Create List\n3.BFS Traversal\n4.DFS Traversal\n5.Recursive DFS:");

n1 = sc.nextInt();

switch (n1) {

case 1:

m.creatematrix(n);

m.displaymatrix(n);

break;

case 2:

m.creatematrix(n);

m.displaylist(n);

break;

case 3:

m.Bfs(n);

break;

/\*case 4:

m.Dfs(n);

break;

case 5:m.callRDFS(n);

break;\*/

}

System.out.println("\n Enter 1 to continue:");

n2 = sc.nextInt();

} while (n2 == 1);

}

}

/\*OUTPUT :-

\*

Enter the number of houses where the man drops the newspaper:

8

1.Create Matrix

2.Create List

3.BFS Traversal

4.DFS Traversal

5.Recursive DFS:

1

Enter 1 if there is a route between house 0 and house 1 else Enter 0:

1

Enter 1 if there is a route between house 0 and house 2 else Enter 0:

0

Enter 1 if there is a route between house 0 and house 3 else Enter 0:

1

Enter 1 if there is a route between house 0 and house 4 else Enter 0:

0

Enter 1 if there is a route between house 0 and house 5 else Enter 0:

0

Enter 1 if there is a route between house 0 and house 6 else Enter 0:

1

Enter 1 if there is a route between house 0 and house 7 else Enter 0:

0

Enter 1 if there is a route between house 1 and house 2 else Enter 0:

0

Enter 1 if there is a route between house 1 and house 3 else Enter 0:

0

Enter 1 if there is a route between house 1 and house 4 else Enter 0:

1

Enter 1 if there is a route between house 1 and house 5 else Enter 0:

1

Enter 1 if there is a route between house 1 and house 6 else Enter 0:

0

Enter 1 if there is a route between house 1 and house 7 else Enter 0:

0

Enter 1 if there is a route between house 2 and house 3 else Enter 0:

0

Enter 1 if there is a route between house 2 and house 4 else Enter 0:

0

Enter 1 if there is a route between house 2 and house 5 else Enter 0:

1

Enter 1 if there is a route between house 2 and house 6 else Enter 0:

0

Enter 1 if there is a route between house 2 and house 7 else Enter 0:

1

Enter 1 if there is a route between house 3 and house 4 else Enter 0:

0

Enter 1 if there is a route between house 3 and house 5 else Enter 0:

1

Enter 1 if there is a route between house 3 and house 6 else Enter 0:

0

Enter 1 if there is a route between house 3 and house 7 else Enter 0:

0

Enter 1 if there is a route between house 4 and house 5 else Enter 0:

0

Enter 1 if there is a route between house 4 and house 6 else Enter 0:

1

Enter 1 if there is a route between house 4 and house 7 else Enter 0:

0

Enter 1 if there is a route between house 5 and house 6 else Enter 0:

0

Enter 1 if there is a route between house 5 and house 7 else Enter 0:

0

Enter 1 if there is a route between house 6 and house 7 else Enter 0:

0

Displaying the adjacency Matrix:

H1 H2 H3 H4 H5 H6 H7 H8

H1 0 1 0 1 0 0 1 0

H2 1 0 0 0 1 1 0 0

H3 0 0 0 0 0 1 0 1

H4 1 0 0 0 0 1 0 0

H5 0 1 0 0 0 0 1 0

H6 0 1 1 1 0 0 0 0

H7 1 0 0 0 1 0 0 0

H8 0 0 1 0 0 0 0 0

Enter 1 to continue:

1

1.Create Matrix

2.Create List

3.BFS Traversal

4.DFS Traversal

5.Recursive DFS:

2

Displaying the adjacency list:

The houses connected to node 1

House no. 2 House no. 4 House no. 7

The houses connected to node 2

House no. 1 House no. 5 House no. 6

The houses connected to node 3

House no. 6 House no. 8

The houses connected to node 4

House no. 1 House no. 6

The houses connected to node 5

House no. 2 House no. 7

The houses connected to node 6

House no. 2 House no. 3 House no. 4

The houses connected to node 7

House no. 1 House no. 5

The houses connected to node 8

House no. 3

Enter 1 to continue:

1

1.Create Matrix

2.Create List

3.BFS Traversal

4.DFS Traversal

5.Recursive DFS:

3

BFS Traversal:

Enter the home no. which you want to start with:

0

House no.0

House no.1

House no.3

House no.6

House no.4

House no.5

House no.2

House no.7

Enter 1 to continue:

1

1.Create Matrix

2.Create List

3.BFS Traversal

4.DFS Traversal

5.Recursive DFS:

4

Enter the house you want to start at:

0

House no.0

House no.1

House no.4

House no.6

House no.5

House no.2

House no.7

House no.3

Enter 1 to continue:

1

1.Create Matrix

2.Create List

3.BFS Traversal

4.DFS Traversal

5.Recursive DFS:

5

Enter starting vertex:

0

0

1

4

6

5

2

7

3

Enter 1 to continue:

0

\*/