DAY 18

1. BFS

import java.util.\*;

class Graph {

int[][] mat;

int num;

Graph(int num) {

this.num = num;

mat = new int[num][num];

}

void addedge(int source, int dest) {

mat[source][dest] = 1;

mat[dest][source] = 1;

}

void removeedge(int source, int dest) {

mat[source][dest] = 0;

mat[dest][source] = 0;

}

void printgraph() {

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

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

System.out.print(mat[i][j] + " ");

}

System.out.println();

}

}

void BFS(int startVertex) {

boolean[] visited = new boolean[num];

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

visited[startVertex] = true;

queue.offer(startVertex);

System.out.print("BFS from " + startVertex + ": ");

while (!queue.isEmpty()) {

int current = queue.poll();

System.out.print(current + " ");

for (int v = 0; v < num; v++) {

if (mat[current][v] == 1 && !visited[v]) {

visited[v] = true;

queue.offer(v);

}

}

}

System.out.println();

}

}

public class Main {

public static void main(String[] args) {

Graph g = new Graph(5);

g.addedge(0, 1);

g.addedge(1, 4);

g.addedge(4, 3);

g.addedge(3, 2);

g.addedge(2, 0);

g.printgraph();

g.BFS(0);

}

}

2)

import java.util.\*;

class Graph {

int[][] mat;

int num;

Graph(int num) {

this.num = num;

mat = new int[num][num];

}

void addedge(int source, int dest) {

mat[source][dest] = 1;

mat[dest][source] = 1;

}

void removeedge(int source, int dest) {

mat[source][dest] = 0;

mat[dest][source] = 0;

}

void printgraph() {

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

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

System.out.print(mat[i][j] + " ");

}

System.out.println();

}

}

void DFS(int startVertex) {

boolean[] visited = new boolean[num];

DFSRecursive(startVertex, visited);

System.out.println();

}

private void DFSRecursive(int vertex, boolean[] visited) {

visited[vertex] = true;

System.out.print(vertex + " ");

for (int v = 0; v < num; v++) {

if (mat[vertex][v] == 1 && !visited[v]) {

DFSRecursive(v, visited);

}

}

}

}

public class Main {

public static void main(String[] args) {

Graph g = new Graph(5);

g.addedge(0, 1);

g.addedge(1, 4);

g.addedge(4, 3);

g.addedge(3, 2);

g.addedge(2, 0);

g.printgraph();

g.DFS(0);

}

}