*//*

*Un drum de valoare maxima de la p la q folosind algoritmul lui Bellman - Kalaba.*

*//*

1. Formularea corecta completa si concreta a problemei

Sa se determine drumul de valoare maxima intre varfurile p si q, ale unui graf, folosind algoritmul lui Bellman-Kalaba.

Se da :

Varfurile matricea valorilor v a unui graf precum si varfurile p si q.

Se cere:

Drumul de valoare maxima intre varfurile p si q.

2. Modelarea problemei (mod de abordare)

Descrierea algoritmului

Algoritmul lui Bellman-Kalaba determina valorile maxime ale drumurilor de la un varf fixat la orice alt varf intr-un graf oarecare sau detecteaza existenta unui circuit de valoare pozitiva.

Observatie:

In elaborarea algoritmului lui Bellman-Kalba pentru a determina elementul ℷik+1

din tablou se procedeaza astfel: se aduna linia k din tablou cu coloana i din matricea valorilor arcelor apoi se determina valoarea maxima a rezultatelor. Valoare maxima se trece in tablou pe pozitia ℷik+1.

Metoda de rezolvare

Se observa ca numarul ℷik reprezinta valoarea maxima a drumurilor de la varful 1 la varful i ∈ X care au lungimea cel mult k, adica valoarea maxima a drumurilor care trec prin cel mult k + 1 varfuri. Daca nu exista circuite de valoare pozitiva atunci cum drumul de valoare maxima este elementar inseamna ca el trece prin cel mult n varfuri de valoare maxima este cel mult n - 1 iteratii.

3. Descrierea algoritmului Pseudocod

Date: n, p, q, (Vij, i=1,..,n, j=1,..,n)

Rezultate: u (p, q) - drumul

Algoritmul transform este:

(a) initializari

ℷ11 :=0

ℷ1j := v1j, j ≥ 2

k := 1

(b) iteratia de baza

ℷ1k+1 := 0

ℷjk+1 := max { ℷ1k + vij | i = 1,n }, j ≥ 2

k := k + 1

(c) criteriul de stop

**daca** ( ℷjk+1 = ℷjk **pentru** j=1,n )

**atunci** stop **sdf**;

**daca** ( k >m )

**atunci** goto (b);

**altfel** tipareste “exista circuit de valoare pozitiva”

stop;

**sdf**.

Sfarsit algoritm.

4. Date de test

|  |  |
| --- | --- |
| Date de intrare | Date de iesire |
| n = 5  p = 2  q = 4  0 5 6 8 0  0 0 1 0 0  0 0 0 5 9  0 0 0 0 0  0 0 0 6 0 | 2,3,4,5 |

5. Codul Sursa

/\*

\*\*

Drumul de valoare maxima

\*\*/

public class lab02 {

private static int n;

private static int m = 0;

private static int p;

private static int q;

private static int[][] valueMatrix = new int[20][20];

private static int[][] transValueMatrix = new int[20][20];

private static int[][] lambda = new int[20][20];

private static int[] road = new int[40];

private static int k = 1;

private static int cod;

private static final int INFINIT = Integer.MAX\_VALUE / 2;

public static void main(String[] args) {

read();

trans();

road();

print();

}

private static void print() {

File file = new File("output.txt");

BufferedWriter writer;

try {

writer = new BufferedWriter(new FileWriter(file));

if (cod == 1) {

writer.write("No road between " + p + " and " + q);

writer.close();

return;

}

if (lambda[k][q] >= INFINIT) {

writer.write("No road between " + p + " and " + q);

writer.close();

return;

}

writer.write("The road between " + p + " and " + q

+ " has the length " + lambda[k][q]);

writer.newLine();

writer.write("The road is: ");

writer.newLine();

printRoad(road, q, writer);

writer.close();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

private static void printRoad(int[] road2, int q2, BufferedWriter writer)

throws IOException {

if (road[q2] != p) {

printRoad(road, road[q2], writer);

writer.write(road[q2] + " ");

} else {

writer.write(road[q2] + " ");

}

}

private static void road() {

int max, poz = 0;

boolean e;

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

lambda[k][i] = transValueMatrix[p][i];

lambda[k + 1][i] = transValueMatrix[p][i];

road[i] = p;

}

do {

for (int j = 1; j < n; j++) {

max = maxim(j, poz);

if (max > lambda[k + 1][j]) {

lambda[k + 1][j] = max;

road[j] = poz;

}

}

e = eq();

k++;

if (!e) {

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

lambda[k + 1][i] = lambda[k][i];

}

}

} while (e || k > m);

if (k > m) {

cod = 1;

} else {

cod = 0;

}

}

private static boolean eq() {

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

if (lambda[k][i] != lambda[k + 1][i]) {

return false;

}

}

return true;

}

private static int maxim(int j, int poz) {

int inf = INFINIT;

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

int s = lambda[k][i] + transValueMatrix[i][j];

if (inf > s) {

inf = s;

poz = i;

}

}

return inf;

}

private static void trans() {

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

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

if (valueMatrix[i][j] != 0) {

transValueMatrix[i][j] = 1;

m++;

} else {

transValueMatrix[i][j] = INFINIT;

}

}

}

}

private static void read() {

File file = new File("input.txt");

String[] elements;

try {

BufferedReader reader = new BufferedReader(new FileReader(file));

n = Integer.parseInt(reader.readLine());

elements = reader.readLine().split(" ");

p = Integer.parseInt(elements[0]);

q = Integer.parseInt(elements[1]);

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

elements = reader.readLine().split(" ");

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

valueMatrix[i][j] = Integer.parseInt(elements[j]);

}

}

reader.close();

} catch (FileNotFoundException e) {

System.out.println("Error reading the file");

e.printStackTrace();

} catch (IOException e) {

System.out.println("Error reading the file");

e.printStackTrace();

}

}

}