**Tema Laborator 2**

**Ghimpu Lucian Eduard**

**Grupa 213**

**Documentatie**

1. **TEXTUL PROBLEMEI.**

Sa se elaboreze un algoritm care rezolva pentru un graf de forma G=(X,U), unde X este multimea varfurilor si U multimea arcelor, problema comis-voiajorului (metoda cel mai apropiat vecin).

1. **Dezvoltarea algoritmului**
2. Citirea matricii de costuri si generarea automata in memodire a unei matrici de adiacenta dupa regula:

Ai,j=

1. **Apelarea subalgoritmului de rezolvare**
2. **Afisarea drumului minim si a costului sau**
3. **Descrierea algoritmului in pseudocod**

Se folosesc 2 vectori ajutatori:

Parcurs -> cu o lungime egala cu numarul de varfuri, in care vom retine daca am parcurs un varf anume (Ex daca indicile 2 are valoare TRUE, inseamna ca varful 2 a fost parcurs).

Drum -> vector in care retinem ordinea prin care am trecut in drumul cu costul minim

In prima etapa se va selecta un varf sursa, apoi de la nodul sursa se va alege prin subalgorituml drumMinim, varful cel mai apropiat de varful sursa dat.

**Subalgoritm drumMinim(vs,matriceCosturi,parcurs) este:**

{vs –varful sursa, matriceCosturi- matrice cu costuri, parcurs-vectorul ajutator}

vu=-1 { vu reprezinta uramtorul varf, varful cu costul cel mai mic de la care putem ajunge din varful sursa dat ca parametru}

vectorCost = matriceCosturi[vs]

minim=99999

pentru I<-1,lungime(listaCost) executa:

daca listaCost[i]<minim si parcurs[i]==False atunci:

{daca costul e minim si nu am trecut prin varful respectiv, il alegem}

minim=listaCost[i]

vu=i

sfDaca

sfPentru

Daca vu != -1 atunci:

drumMinim = vu

altfel

drumMinim = -1

sfDaca

{returnam vu daca am gasit varful, returnam -1 cand au fost parcurse toate varfurile}

sfSubalgoritm

Algoritmul de determinare a drumului minim se va executa de atatea ori care varfuri are graful dat.

Iar la fiecare iteratie vom adauga la vectorul auxiliar drum, varful gasit in drumMinim.

Iar pentru cost, la fiecare iteratie vom adauga la o variabila, costul de la varful precedent la varful urmator dat de drumMinim.

In programul principal vom avea un ciclu infinit care se va intrerupe cand drum minim va returna -1, adica am trecut prin toate varfurile.

Cat timp 2>1 executa:

Daca drumMinim(vs,matriceCosturi,parcurs) == -1 atunci:

@iesi din cliclu (break din c/c++)

AdaugaLaDrum(drumMinim(vs,matriceCosturi,parcurs))

Cost = cost + matriceCosturi[vs][drumMinim(vs,matriceCosturi,parcurs)]

parcurs[drumMinim(vs,matriceCosturi,parcurs)]=True

vs=drumMinim(vs,matriceCosturi,parcurs)

In final afisam drumul, si costul aferent.

1. **Demonstrarea corectitudinii algoritmului**

Am folosit informatii gasite in bibliografia cursului si in resursele laboratorului.

1. **Date de Test**

PRECONDITIE:

Se presupune ca distanta de la varful X la varful X este infinita (desi teoretic este egala cu 0), pentru a facilita calcului drumului minim la urmatorul varf.

Matrice costuri:

inf 100 10 20 200

100 inf 20 70 40

10 20 inf 30 60

20 70 30 inf 40

200 40 60 40 inf

Varf sursa: 1

Drumul Minim 1->3->2->5->4->1

Cost: 130

Matrice costuri:

Inf 45 105 65

45 inf 55 35

105 55 inf 85

65 35 85 inf

Varf sursa: 1

Drumul minim 1->2->4->3

Cost: 270