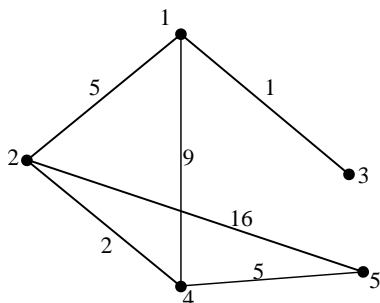


Seminar 12

Algoritmul Dijkstra – descriere, implementare, exemple

Descriere etape – algoritmul Dijkstra

1. Fie graful ponderat,



Considerînd $u_0=1$, scrieți toate etapele în aplicarea algoritmului Dijkstra.

Rezolvare:

P1: $i=0$; $S_0=\{1\}$; $L(1)=0$, $L(i)=\infty$ pentru toți $i = \overline{2,5}$.

P2: $\bar{S}_0=\{2,3,4,5\}$, $u_0=1$

$L(2)=\infty > L(1)+5=5 \Rightarrow L(2)=5$, etichetează 2 cu 1

$L(3)=\infty > L(1)+1=1 \Rightarrow L(3)=1$, etichetează 3 cu 1

$L(4)=\infty > L(1)+9=9 \Rightarrow L(4)=9$, etichetează 4 cu 1

$L(5)=\infty$, $w(1,5)=\infty$, deci $L(5)$ nu se modifică

$$E = \begin{pmatrix} 0 & 5 & \textcolor{red}{1} & 9 & \infty \\ -1 & 1 & 1 & 1 & \end{pmatrix}$$

P3: selectează $u_1=3$, $L(3)=1$, cea mai mică dintre w-distanțele calculate la P2

P4: $S_1=\{1,3\}$

P5: $i=i+1=1 \neq 4$, reia P2

P2: $\bar{S}_1=\{2,4,5\}$, $u_1=3$

Nu se modifică nici o etichetă și nici o w-distanță ($w(3,i)=\infty$, pentru toți i din \bar{S}_1)

P3: selectează $u_2=2$, $L(2)=5$, cea mai mică dintre w-distanțele calculate la P2

P4: $S_2=\{1,3,2\}$

P5: $i=i+1=2 \neq 4$, reia P2

P2: $\bar{S}_2=\{4,5\}$, $u_2=2$

$L(4)=9 > L(2)+2=7 \Rightarrow L(4)=7$, etichetează 4 cu 2

$L(5)=\infty > L(2)+16=21$, etichetează 5 cu 2

Vîrful v pînă la care este calculată w-distanța	1	2	3	4	5
D(1,v), eticheta lui v	0, -1	5, 1	1, 1	7, 2	21, 2

P3: selectează $u_3=4$, $L(4)=7$, cea mai mică dintre w-distanțele calculate la P2

P4: $S_3=\{1,3,2,4\}$

P5: $i=i+1=3 \neq 4$, reia P2

P2: $\bar{S}_3=\{5\}$, $u_3=4$

$L(5)=21 > L(4)+5=12$, etichetează 5 cu 4

Vîrful v pînă la care este calculată w-distanța	1	2	3	4	5
D(1,v), eticheta lui v	0, -1	5, 1	1, 1	7, 2	12, 4

P3: selectează $u_4=5$, $L(5)=12$, cea mai mică dintre w-distanțele calculate la P2

P4: $S_3=\{1,3,2,4,5\}$

P5: $i=i+1=4$, stop.

Algoritmul calculează următoarele rezultate:

$$E = \begin{pmatrix} L & 0 & 5 & 1 & 7 & 12 \\ p & -1 & 1 & 1 & 2 & 4 \end{pmatrix}$$

Drumurile de cost minim de la vîrful 1 la fiecare dintre vîrfurile grafului se stabilesc pe baza sistemului de etichete astfel: drumul de la 1 la un vîrf v este dat de: v_1 , eticheta lui v, v_2 eticheta lui v_1 ..., pînă se ajunge la eticheta 1. Astfel, v_0 -drumurile de cost minim sunt:

pînă la 2: 2,1;

pînă la 3: 3,1;

pînă la 4: 4,2,1;

pînă la 5: 5,4,2,1.

Implementarea algoritmului Dijkstra

```
//Implementarea algoritmului Dijkstra
#include <stdio.h>
#include <malloc.h>

//l -> distanta de la vf. de START(v0) pana la varful curent
//vf -> varful anterior prin care s-a ajuns la varful curent.
typedef struct {
    float l;
    int vf;
}eticheta;

//subprogram
//preia datele unui graf in forma tabelara din fisierul "nume" si obtine
//matricea ponderilor, unde MAX este valoarea cu semnificatie de infinit
//ultima linie a fisierului contine virful initial din algoritmul Dijkstra
void preia_graf(char *nume, float ***w, int *n, int *v0, float MAX)
{
    int i, j, m, u, v;
```

```

float p;

FILE *f = fopen(ume, "rt");
fscanf(f, "%i", n); //citire nr. de varfuri

float **mw = (float **)malloc(*n * sizeof(float*)); //alocare memorie pentru matricea
ponderilor
for (i = 0; i < *n; i++)
    mw[i] = (float *)malloc(*n * sizeof(float));

fscanf(f, "%i", &m); //citire nr. de muchii
for (i = 0; i < *n; i++) //toate elementele din mat.ponderilor sunt initializate cu MAX
    for (j = 0; j < *n; j++)
        mw[i][j] = MAX;

for (i = 0; i < m; i++) //se citesc muchiile dintre varfuri si ponderea unei muchii
{
    fscanf(f, "%i", &u);
    fscanf(f, "%i", &v);
    fscanf(f, "%f", &p);
    mw[u - 1][v - 1] = mw[v - 1][u - 1] = p;
}

fscanf(f, "%i", v0); //pe ultima linie din fisier se afla varful initial.
fclose(f);

*w = mw; //pt a transmite in main() modificarile din aceasta functie
}

//subprogram
//functia pentru implementarea algoritmului Dijkstra - returneaza vectorul de etichete
eticheta *Dijkstra(float **w, int n, int v0, float MAX)
{
    int i, *prel, nrit, ui, v, vmin;
    float lmin;

    //vector de etichete ce reprezinta cea mai scurta distanta de la varful de START(v0)
    pana la un varf oarecare
    //si varful PRECEDENT prin care s a ajuns la varful CURENT
    eticheta *v_et = (eticheta *)malloc(n * sizeof(eticheta));

    //initial cea mai scurta distanta de la vf. de START pana la orice alt varf = INFINIT
    for (i = 0; i < n; i++)
        v_et[i].l = MAX;

    v_et[v0 - 1].l = 0; //distanța de la vf. de START pana la el este logic = 0

    prel = (int*)malloc(n * sizeof(int));
    //vector cu proprietatea prel[i]=1 daca pentru virful i+1 a
    //fost deja determinat un drum de cost minim prel[i]=0 in caz contrar
    //(adica daca varful a fost vizitat)

    for (i = 0; i < n; i++)
        prel[i] = 0;
    prel[v0 - 1] = 1; //varful v0 este primul varf vizitat..

    ui = v0;
    for (nrit = 0; nrit < n - 1; nrit++)
    {
        lmin = MAX;
        //sunt recalculate w-distantele de la virfurile v cu prel[v-1]=0 si este
        determinat vmin,
        //urmatorul virf cu proprietatea ca pentru acesta a fost determinat un drum de
        cost minim - lmin
        for (v = 1; v <= n; v++)
        {

```

```

        //daca nodul CURENT (v-1) este nevizitat si daca distanta de la nodul de
START(v0) pana la nodul CURENT(v-1) este mai mare decat
        //distanța de la nodul de START(v0) pana la varful precedent vizitat(ui-
1) +
        //distanța de la nodul CURENT(v-1) pana la nodul precedent vizitat(ui-1)
        if ((prel[v - 1] == 0) && (v_et[v - 1].l > v_et[ui - 1].l + w[v - 1][ui -
1]))
        {
            v_et[v - 1].l = v_et[ui - 1].l + w[v - 1][ui - 1];
            v_et[v - 1].vf = ui;
        }

        //determina nodul care are cea mai mica distanta fata de nodul de START si
este si nevizitat
        if ((prel[v - 1] == 0) && v_et[v - 1].l < lmin)
        {
            lmin = v_et[v - 1].l; vmin = v;
        }
    }
    //nodul cu cea mai mica distanta fata de nodul de START este adaugat in vectorul
de noduri vizitate
    ui = vmin; prel[ui - 1] = 1;
}
free(prel);
return v_et;
}

//subprogram
//functia principala
void main()
{
    float **w, MAX = 1000000; int n, v0, v, u, i;
    char numef[20];

    printf("Introduceti numele fisierului care contine graful:");
    scanf("%s", numef);

    //citire graf din fisier
    preia_graf(numef, &w, &n, &v0, MAX);

    //calculeaza distanta minima de la v0 pana la orice alt varf
    eticheta *rez = Dijkstra(w, n, v0, MAX);

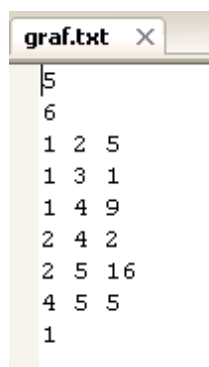
    //afisare
    for (v = 1; v <= n; v++)
    {
        if (v != v0)
        {
            printf("Costul celui mai ieftin drum de la %i la %i este %8.3f", v, v0,
rez[v - 1].l);
            printf("\n Un drum de cost minim: %i ", v);
            u = v;
            while (rez[u - 1].vf != v0)
            {
                printf("%i ", rez[u - 1].vf);
                u = rez[u - 1].vf;
            }
            printf("%i \n\n", v0);
        }
    }

    //eliberare memorie
    free(rez);

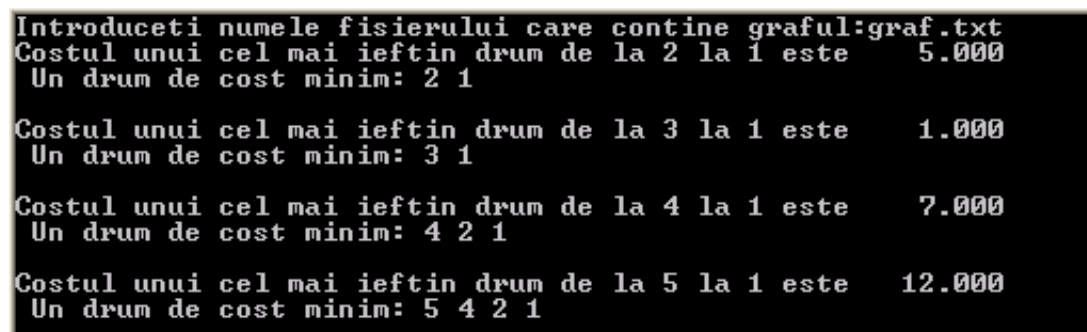
    for (i = 0; i < n; i++) free(w[i]);
}

```

```
    free(w);  
}  
Exemplu de execuție
```



```
graf.txt X  
5  
6  
1 2 5  
1 3 1  
1 4 9  
2 4 2  
2 5 16  
4 5 5  
1
```

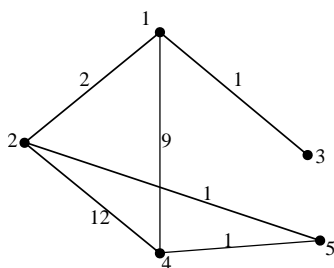


```
Introduceti numele fisierului care contine graful:graf.txt  
Costul unui cel mai ieftin drum de la 2 la 1 este      5.000  
Un drum de cost minim: 2 1  
  
Costul unui cel mai ieftin drum de la 3 la 1 este      1.000  
Un drum de cost minim: 3 1  
  
Costul unui cel mai ieftin drum de la 4 la 1 este      7.000  
Un drum de cost minim: 4 2 1  
  
Costul unui cel mai ieftin drum de la 5 la 1 este     12.000  
Un drum de cost minim: 5 4 2 1
```

În anumite aplicații este necesară exclusiv determinarea w -distanțelor $D(v_0, v)$, pentru toți $v \in V$. În acest caz algoritmul Roy-Floyd permite o rezolvare a acestei probleme mai simplu de implementat decât algoritmul Dijkstra.

Teste de autoevaluare

2. Fie graful ponderat,



Considerînd $u_0=1$, descrieți rezultatele aplicării algoritmului Dijkstra acestui graf.

Raspuns:

Se vor parcurge toți pașii, la fel ca la problema 1. Rezultatul se va compara cu rezultatul execuției programului (pentru verificare).

Algoritmul calculează următoarele rezultate:

Vîrful v pînă la care este calculată w -distanța	1	2	3	4	5
$D(1,v)$, eticheta lui v	0, 1	2, 1	1, 1	4, 5	3, 2

Drumurile de cost minim de la vîrful 1 la fiecare dintre vîrfurile grafului se stabilesc pe baza sistemului de etichete. Astfel, v_0 -drumurile de cost minim sînt:

pînă la 2: 2,1;

pînă la 3: 3,1;

pînă la 4: 4,5,2,1;

pînă la 5: 5,2,1.

Fisierul graf1.txt trebuie sa se regaseasca in directorul proiectului.

```
graf1.txt X
5
6
1 2 2
1 3 1
1 4 9
2 4 12
2 5 1
4 5 1
1
```

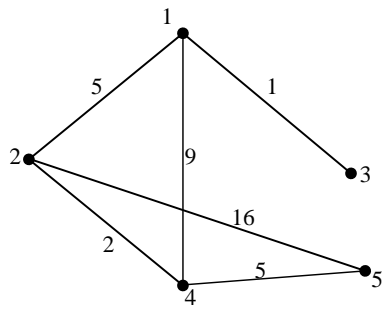
```
Introduceti numele fisierului care contine graful:graf1.txt
Costul unui cel mai ieftin drum de la 2 la 1 este      2.000
Un drum de cost minim: 2 1

Costul unui cel mai ieftin drum de la 3 la 1 este      1.000
Un drum de cost minim: 3 1

Costul unui cel mai ieftin drum de la 4 la 1 este      4.000
Un drum de cost minim: 4 5 2 1

Costul unui cel mai ieftin drum de la 5 la 1 este      3.000
Un drum de cost minim: 5 2 1
```

3. Fie graful,



Considerînd $u_0=4$, descrieți rezultatele aplicării algoritmului Dijkstra acestui graf.