КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: СИНТЕЗ И АНАЛИЗ НА АЛГОРИТМИ

# ЛАБОРАТОРНО УПРАЖНЕНИЕ № 11

## *ТЕМА: Алгоритъм за намиране на най – къси пътища в графи*

**ЦЕЛ:**

Целта на упражнението е студентите да се запознаят с алгоритми за намиране на най – къси пътища. След упражнението студентите би следвало да могат да прилагат тези алгоритми.

## I. ТЕОРЕТИЧНА ЧАСТ

Задачата за най- кратък път в граф се състои в намирането на най- кратък път от даден връх до друг даден връх, ако съществува път между тях. Алгоритмите на Прим и Краскал построяват МПД, но не винаги в него пътищата между два върха са най – кратките. За намирането на най- кратък път между два върха се използват следните алгоритми:

- Алгоритъм на Дейкстра;
- Алгоритъм на Белман-Форд.

### 1. Алгоритъм на Дейкстра

Основната идея на алгоритъма е, че се построява покриващо дърво по подобие на алгоритъма на Прим за МПД, като корен на това дърво е началният връх на пътя и добавя едно по едно ребрата и върховете на графа към вече построената част от дървото. Има обаче една съществена разлика. Докато при алгоритъма на Прим се избира ребро с най – малко тегло, което да се добави към покриващото дърво, то при алгоритъма на Дейкстра следващият връх, който се добавя, се избира така че гарантирано за всички вече добавени върхове да са построени най – кратките пътища от началния до всеки от тях.

Първоначално всички върхове се инициализират с етикет (*label*) за теглото на най-краткия път от началния връх до тях. Този етикет е число, по-голямо от сумата от всички тегла на ребрата на $G$. Обикновено за начална стойностна етикета се избира $+\infty$. За начален връх поставяме етикет 0 – най-краткият път от началния връх до самия него. На всяка стъпка се избира един текущ връх, като първоначално текущият връх се инициализира с началния връх. След това се прави преизчисляване на етикетите на неоцветените върхове, за които съществува ребро, инцидентно с тях и с текущия връх. Тези върхове. За които е правено преизчисление се съхраняват в множество, наречено фронт (*fringe*). Текущият връх се оцветява (*paint*) и се изключва от множеството на разглежданите върхове на следващите стъпки. При избора на текущ връх се избира от фронта този връх, който е с най-малък етикет и се продължава отново с оценяването, докато се оцветят всички върхове на графа.

Нека е даден свързан граф $G(V,E)$, $|V|=n$ $|E|=m$ и е зададена теглова функция по ребрата w: $E \rightarrow R^+$. Нека са зададени начален връх $s \in V$ и целеви връх $t \in V$. Дефинираме функция $l: V \rightarrow R^+$, която за всеки връх $x \in V$ съдържа стойността на най-краткия път, който е намерен до момента, от началния връх s до върха $x$.

***Процедурни стъпки:***

1) Дефинираме покриващо дърво $T(V',E')$ на $G$, като първоначално дървото е празно и $V'=\{s\}$, $E'=\emptyset$;
2) Дефинираме фронт $F=\emptyset$ и функция $D: F \rightarrow V'$;
3) Инициализираме стойностите на функцията $l$ за всички върхове от графа $G$, като полагаме $l(s):=0$ за началния връх и $l(x):=+\infty$ за всички останали $x \in V$;
4) Инициализираме текущия връх $p=s$ и фронтът $F=\{a\}$;
5) Ако $t \in V'$ отиваме на стъпка $9$;
6) Оцветяваме текущия връх $p$ и го изключваме от фронта $F=F\setminus\{p\}$, и включваме в покриващото дърво $V'=V' \cup \{p\}$;
7) За всеки връх $x \in V$, такъв че $x \notin V'$ и съществува ребро $\{p,x\} \in E$, го включваме във фронта $F:=F \cup \{x\}$ и преоценяваме стойността на $l(x)$.
   Ако $l(x)>l(x)+w(\{p,x\})$, то $l(x):=l(x)+w(\{p,x\})$ и $D(x)=p$;
8) Намира се връх $v \in F$ с минимална стойност на етикета от всички върхове във $F$. Полагаме $p=v$ и добавяме дъгата $\{p,x\}$ в покриващото дърво $T$, $E'=E' \cup \{p,x\}$. Извършва се връщане към стъпка $5$.
9) Край. Връща се стойността на етикета $l(t)$ - дължината на най-краткия път от началния връх $s$ до целевия връх $t$ в $G$.

Използването на този алгоритъм има ограничение че теглата трябва да имат положителни стойности. Той не може да се използва ако има отрицателни тегла на ребрата.

2. Алгоритъм на Белман-Форд

В много приложни задачи освен положителни тегла на ребрата на графа се налага да има и отрицателни такива. Това прави предходния алгоритъм неизползваем за съответния граф. В този случай може да се приложи алгоритъма на Белман-Форд. В задачата за намиране на най-кратък път чрез този алгоритъм се налага единствено ограничение в графа $G$ да няма цикли със сумарно отрицателно тегло. Наличието на такива цикли прави задачата несъдържателна. В този случай цикълът ще се обхожда многократно и пътят между два върха става безкрайно малък с общо тегло, което ще клони към $-\infty$.

Основната разлика между двата алгоритъма е, че при алгоритъма на Дейкстра търсим пътища с най – малък брой ребра и с най – малко сумарно тегло. Докато при алгоритъма на Белман-Форд, ако има ребра с отрицателни тегла, при по-голям брой на ребрата намереният път е възможно да се получи по-малко сумарно тегло.

Основната идея на този алгоритъм е да се започне от начален връх $1$ и нека за два върха $i$ и $j$ да съществува ребро $(i, j)$. Тогава, ако $d_i$ на $d_j$ са съответно теглата на текущия най-кратък път от върха $i$ до $j$, то на всяка стъпка се взема по-малкото от разстоянията $d_j$ за пътя от $1$ до $j$ или $d_i+w(i,j)$ за най – краткия път от $1$ до $i$ и реброто $(i, j)$.

Първоначално търсим всички пътища от началния връх, които съдържат едно ребро(дъга). След това изследваме пътищата, които съдържат $2$ ребра, след което пътищата с $3$ и т.н. Като при разглеждането на следващата стъпка се построява само продълженията с още $1$ ребро, само на онези пътища от предходната стъпка, за която е

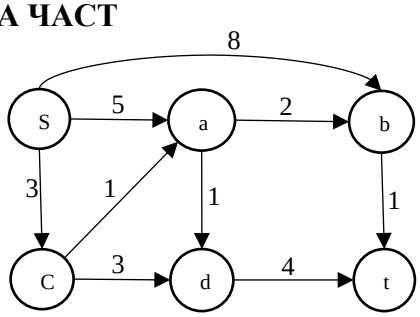
приложена релаксация. Понеже графът не съдържа цикли със сумарно отрицателно тегло, то максималното количество ребра, които може да съдържа даден път, е колкото общият брой на върховете в графа минус ***1***.

Нека е даден свързан граф без цикли със сумарно отрицателно тегло ***G(V, E)***, ***|V|=n,*** ***|E|=m*** и е зададена теглова функция по ребрата ***w: E→R***. Нека е зададен начален връх ***s∈V***. Дефинираме функция ***d:V→R***, която за всеки връх ***x∈V*** съдържа стойността на най-краткия път, който е намерен до момента, от началния връх ***s*** до върха ***x***.

***<u>Процедурни стъпки:</u>***

1) Инициализират се стойностите на функцията ***d*** за всички върхове от графа ***G***, като полагаме ***d(s):=0*** за началния връх и ***d(x):=***+∞ за всички останали ***x∈V***.;
2) Търси се ребро ***(i, j)∈E***, такова че ***d(j)-d(i)>w(i, j)***;
3) Ако съществува такова ребро се преминава към ***стъпка 5***;
4) Ако не съществува такова ребро се полага ***d(j):=d(i)+w(i, j)*** и прави връщане към ***стъпка 2***.
5) Край.

## II. ПРАКТИЧЕСКА ЧАСТ



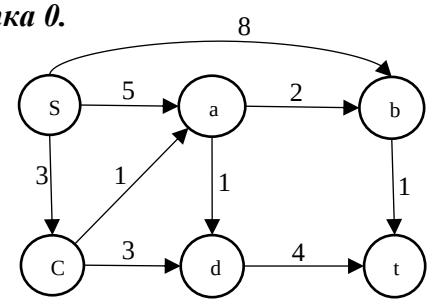
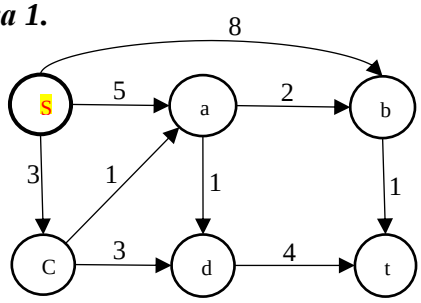

*Фиг.1. Граф за обхождане*

***ЗАДАЧА1:*** *За графа на фиг.1. да се намери най краткия път от връх s до връх t.*

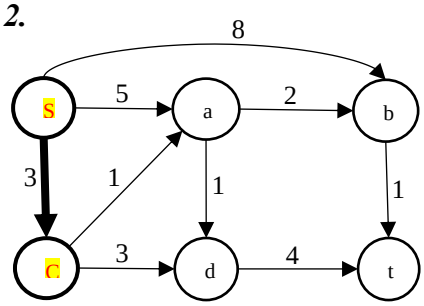
***РЕШЕНИЕ:***

*В таблица 1 е описано стъпка по стъпка намирането на най краткия път между двата върха, а в таблица 2 оцветяването на върховете в отделните стъпки при алгоритъма на Дейкстра.*

*Табл.1.Намиране на най – кратък път по алгоритъма на Дейкстра*

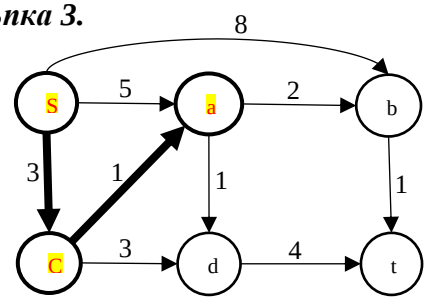
| ***Стъпка 0.*** | ***Стъпка 1.*** |
|---|---|
|  |  |
| *V'={s}, F={s}, E'=∅;*<br>*l(s)=0*<br>*l(a)=l(b)=l(c)=l(d)=l(t)=+∞* | *V'={s}, F={a,b,c}, E'=∅;*<br>*l(a)=l(s)+w(s, a)=0+5=5*<br>*l(b)=l(s)+w(s, b)=0+8=8*<br>*l(c)=l(s)+w(s, c)=0+3=3 -> min* |

***Стъпка 2.***



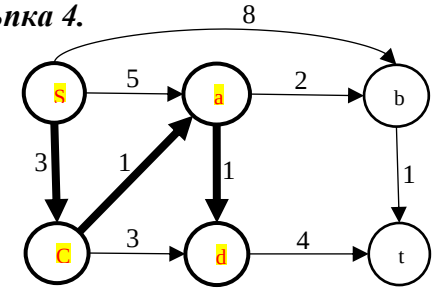

*V‘={s,c}, F={a,b,d}, E’={(s, c)};*
*l(a)=l(c)+w(c, a)=3+1=4 -> min*
*l(b)=l(c)+w(c, d)=3+3=6*

***Стъпка 3.***



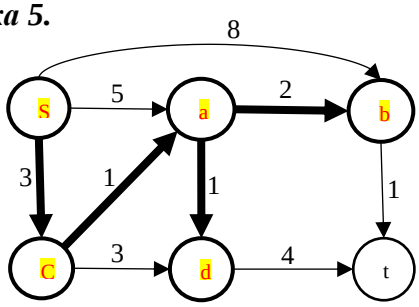

*V‘={s, c, a}, F={b, d}, E’={(s, c),(c,a)};*
*l(b)=l(a)+w(a, b)=4+2=6*
*l(d)=l(a)+w(a, d)=4+1=5 -> min*

***Стъпка 4.***



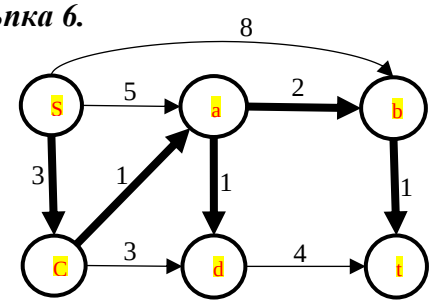

*V‘={s, c, a, d}, F={b, t},*
*E’={(s, c),(c, a),(a, d)};*
*l(b)=6 – не се променя -> min*
*l(t)=l(d)+w(d, t)=5+4=9*

***Стъпка 5.***




*V‘={s, c, a, d, b}, F={t},*
*E’={(s, c),(c, a),(a, d),(a, b)};*
*l(t)=l(b)+w(b, t)=6+1=7 -> min*

***Стъпка 6.***




*V‘={s, c, a, d, b}, F={t},*
*E’={(s, c),(c, a),(a, d),(a, b),(b, t)};*

***Стъпка 7.***

*КРАЙ*

*Най – кратък път: s-c-a-b-t*
*l(t)=7*

*Табл.2. Оцветяване на върховете стъпка по стъпка при алгоритъма на Дейкстра*

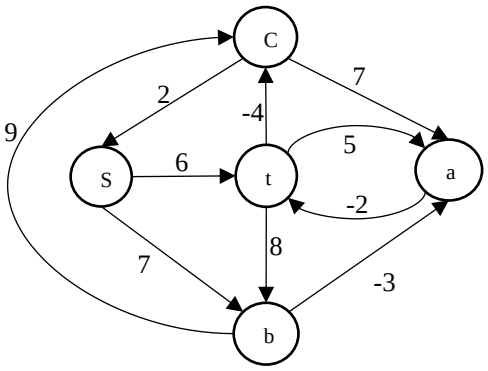
| ***Стъпка*** | ***p*** | ***l(s)*** | ***l(a)*** | ***l(b)*** | ***l(c)*** | ***l(d)*** | ***l(t)*** |
|---|---|---|---|---|---|---|---|
| *0* | *s* | ***0*** | *+ ∞* | *+ ∞* | *+ ∞* | *+ ∞* | *+ ∞* |
| *1* | *s* | - | *5* | *8* | ***3*** | *+ ∞* | *+ ∞* |
| *2* | *c* | - | ***4*** | *8* | - | *6* | *+ ∞* |
| *3* | *a* | - | - | *6* | - | ***5*** | *+ ∞* |
| *4* | *d* | - | - | ***6*** | - | - | *9* |
| *5* | *b* | - | - | - | - | - | ***7*** |
| *6* | *t* | - | - | - | - | - | - |

***ЗАДАЧА2:*** *Да се напише функция която реализира алгоритъма на Дейкстра*

***РЕШЕНИЕ:***

```
void dijkstra()
{
// c[N][N] – ценова матрица на разстоянията;
/* S[N] – булев масив определящ дали даден връх е посетен или не първоначално всички
трябва да са не посетени */
// d[N] – масив съдържащ най късите разстояния от началния до всеки един връх на графа
// p[N] – множество реализирано като масив за списъка с избраните върхове.

    int w, i;
    s[node] = true;     // определяме началния връх за посетен. Началния връх е node
    p[0] = node;        // добавя този връх в масива p.
    for (i = 1; i < n; i++)
    {   // смятаме разстоянията от началния връх до i-тия;
        d[i] = c[node][i]; p[i] = 0;
    }

    for (i=1; i<n; i++)
    {
        w = minimum(d, n); /* функция минимум намира най-краткото разстояние до даден
връх от началния */

        // избиране на връх w, за който d[w] е min;
        s[w] = true;//добавяне на w към маркираните (като посетен)
        p[i] = w;   // и добавяме върхът като следващ избран в масива p
        for (int j=1; j<n; j++)
           if (s[j] == false)  //проверяваме дали текущия връх j е посетен
             if (c[w][j] != 0)//проверяваме дали има директен път от връх w до връх j.
                { /* ако до този момент е липсвало разстоянието до j-тия връх от началния
или то е по голямо от новото пресметнато, то то се добавя като ново разстояние на мястото
на старото.*/
                    if (d[j] == 0)
                        d[j] = (d[w] + c[w][j]);
                    if (d[j] > (d[w] + c[w][j]))
                        d[j] = d[w] + c[w][j];
                }
    }
}
```

***ЗАДАЧА2:*** *За графа на фиг.2. да се намери най кратките пътища от начален връх s до всички останали върхове.*




*Фиг.2. Граф за задача 3*

***РЕШЕНИЕ:***

*Табл.3.Алгоритъм на Белмон-Форд*

| ***Брой дъги*** | ***Път*** | ***Текущи най кратки пътища*** |
|---|---|---|
| *1* | *s – t*<br>*s - b* | *d(t)=d(s)+w(s, t) = 0 + 6 = 6*<br>*d(b)=d(s)+w(s, b)= 0 +7 = 7* |
| | *s – t – c*<br>*s – t – a*<br>*s – t – b*<br>*s – b – c*<br>*s – b – a* | *d(c)=d(t)+w(t, c) = 6 + (-4) = 2*<br>*d(a)=d(t)+w(t, a)= 6 +5 = 11*<br>*7=d(b)!≥ d(t)+w(t, b)=6+8=14 (изключва се)*<br>*2=d(c) !≥d(b)+w(b, c)=7+9=16 (изключва се)*<br>*d(a)=d(b)+w(b, a)= 7 + (-3) = 4* |
| | *s – t – c – a*<br>*s – t – c – s*<br>*s – b – a – t* | *4=d(a) !≥d(c)+w(c, a)=2 + 7=9 (изключва се)*<br>*Не го разглеждаме понеже е цикличен*<br>*d(t)=d(a)+w(a, t) = 4 + (-2) = 2* |
| | *s – b – a – t – b*<br>*s – b – a – t – c*<br>*s – b – a – t – a* | *Не го разглеждаме понеже е цикличен*<br>*d(c)=d(t)+w(t, c) = 2 + (-4) = -2*<br>*Не го разглеждаме понеже е цикличен* |

*Така окончателно намерените най кратки пътища с начален връх s са:*
*s – b – a : d(a)=4;*
*s – b : d(b)=7;*
*s – b – a – t – c : d(c)= -2;*
*s – s : d(s)=0;*
*s – b – a – t : d(t) = 2.*

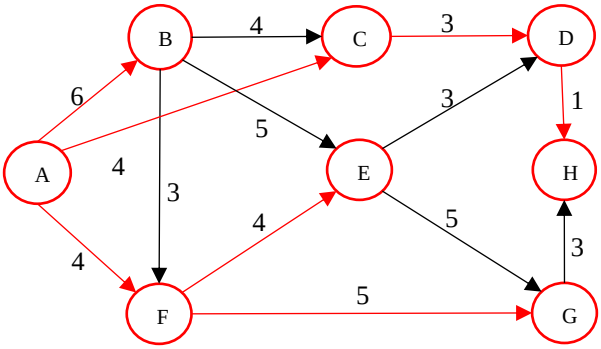
***ЗАДАЧА4:*** *Да се напише функция която реализира алгоритъма на Белмант Форд*

***РЕШЕНИЕ:***

```
void BellmanFord(int graph[][3], int V, int E, int src)
{
    //Първоначална инициализация на разстоянията до всички върхо на графа като 0.
    int dis[N];
    for (int i = 0; i < V; i++)
        dis[i] = INT_MAX;

    //Инициализира разстоянията до началния връх като 0.
    dis[src] = 0;

    //Намира най късите разстояния до всички останали върхове на графа
    for (int i = 0; i < V - 1; i++) {

        for (int j = 0; j < E; j++) {
            if (dis[graph[j][0]] != INT_MAX && dis[graph[j][0]] + graph[j][2] <
                dis[graph[j][1]])
                dis[graph[j][1]] =
                dis[graph[j][0]] + graph[j][2];
        }
    }


 //Проверява дали графа съдържа цикли със отрицателно сумарно тегло на ребрата в него.
 // Ако съдържа такъв цикъл приключва програмата
    for (int i = 0; i < E; i++) {
        int x = graph[i][0];
```

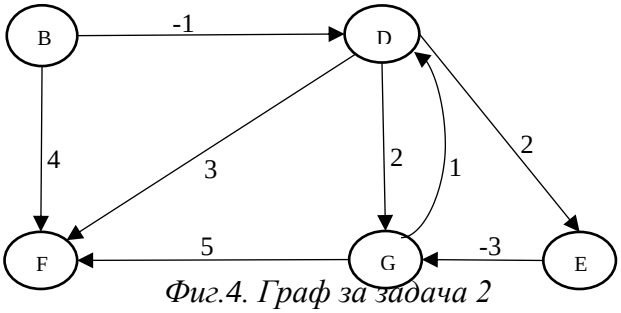
```
        int y = graph[i][1];
        int weight = graph[i][2];
        if (dis[x] != INT_MAX && dis[x] + weight < dis[y])
        {
            cout << "Grafa sadarja cikli sas sumarno otricatelno teglo na rebrata w
cikala" << endl;
            return;
        }
    }

    cout << "Razstoqniqta do varhovete ot na4alniq vrah sa slednite: " << endl;
    for (int i = 0; i < V; i++)
        cout << i << "\t" << dis[i] << endl;
}
```

## III. Задача за семинарни упражнения.




*Фиг.3. Граф за задача 1*

1. Да се намери най-краткият път от върха A до връх H за графа на фиг.3, като се приложи алгоритъмът на Дейкстра.
2. Да се намери най-краткият път от връх B до всеки един от върховете за графа на фиг.4. като се приложи алгоритъмът на Белман-Форд.




*Фиг.4. Граф за задача 2*

## IV. Задача за лабораторни упражнения.

1. Довършете задача 2 от практическата част така че да работи за графа на фиг. 1., като се добавят липсващите фрагменти и функции от кода. Приложете написания код и за графа на фиг. 3.
2. Довършете задача 4 от практическата част така че да работи за графа на фиг.2., като се добавят липсващите фрагменти и функции от кода. Приложете написания код и за графа на фиг. 4.