UNIVERZITET U SARAJEVU

ELEKTROTEHNIČKI FAKULTET SARAJEVO

DOMAĆA ZADAĆA 4

OSNOVE OPERACIONIH ISTRAŽIVANJA

Zadatak 4 – Izvještaj, Zadatak 5/6

Student: Mašović Haris

Indeks: 17993

Odsjek: Računarstvo i Informatika

Datum:

29.12.2018

Potpis:

_____

Zadatak 1 (opis predstavljaju komentari u kodu)

```
function [T,V]=Kruskal(E)

% sortiramo nasu matricu tako da mozemo nalaziti uvijek najmanju granu bro
E = sortrows(E,3);
% kreiranje pomocnih matrica i vektora
M = max(E);
nasBrojCvorora = M(1);
if M(2)>M(1)
    nasBrojCvorova = M(2);
end
% kreiramo pocetnu T matricu
T = zeros(nasBrojCvorova - 1,2);
% kreiramo referentni vektor tako da svako referise prvo na sebe
referentniVektor = 1:nasBrojCvorova;
% kreiramo tezinski vektor jedinica
tezinaVektor = ones(1,nasBrojCvorova);
% postavljamo nase V na nulu na pocetku
V=0;
% brojimo grane radi T
grana = 0;
% tok postupka
for i=1:size(E,1) % ovim dobijamo nase m
   % uzimamo nasa 2 cvora najnormalnije
   prviCvor = E(i,1);
   drugiCvor = E(i,2);

   % trazimo nase referentne cvorove pri tome ovdje dobijamo nase log m
   refCvor1 = referentniVektor(prviCvor);
   pamti1 = prviCvor;
   while(refCvor1 ~= pamti1)
       pamti1 = refCvor1;
       refCvor1 = referentniVektor(pamti1);
   end
   refCvor2 = referentniVektor(drugiCvor);
   pamti2 = drugiCvor;
   while(refCvor2 ~= pamti2)
       pamti2 = refCvor2;
       refCvor2 = referentniVektor(pamti2);
   end
   % grana se moze uzeti akko referentni cvorovi nisu isti
   if(refCvor1 ~= refCvor2)
       % gledamo tezine pa zavisnosti od tezina jedan proglašavamo za
       % referentni, a drugi da prati
        cvorMinTezina = refCvor1;
        refCvor = refCvor2;
        if tezinaVektor(cvorMinTezina) < tezinaVektor(refCvor)
           pamti3 = cvorMinTezina;
           cvorMinTezina = refCvor;
           refCvor = pamti3;
        end
       % updateujemo tezinaVektor i referentniVektor
        tezinaVektor(cvorMinTezina) = tezinaVektor(cvorMinTezina) +
tezinaVektor(refCvor);
        referentniVektor(refCvor) = referentniVektor(cvorMinTezina);
       % upisujemo nase vrijednosti u T i V
        grana = grana + 1;
        T(grana,1) = prviCvor;
        T(grana,2) = drugiCvor;
        V = V + E(i,3);
```

```
   end
   % ukupna kompleknost je O(mlogn) odnosno O(mlogm)
end
end
```

Testirajmo nasu funkciju na nekoliko primjera.

Primjer #1:

```
E=[1,2,2;1,3,3;1,5,8;2,3,4;2,6,9;3,4,7;4,5,4;4,6,3;5,6,5;5,7,5;6,7,7;6,8,6;
7,8,1];

[T,V]=Kruskal(E)
```

Kao rezultat dobijamo:

```
T =


     7     8

     1     2

     1     3

     4     6

     4     5

     5     7

     3     4



V =

    25
```

Vidimo da je rješenje isto kao na postavci zadaće.

Primjer #2:

Potrebno je odrediti minimalno povezujuće stablo za graf čija je težinska matrica

$$W = \begin{pmatrix} 0 & 3 & 7 & 3 & 2 \\ 3 & 0 & 6 & 8 & 5 \\ 7 & 6 & 0 & 10 & \infty \\ 3 & 8 & 10 & 0 & 4 \\ 2 & 5 & \infty & 4 & 0 \end{pmatrix}$$

Rješenje:

Minimalno povezujuće stablo čine grane $1-2$, $1-4$, $1-5$ i $2-3$, a ukupna težina mu je 14.

```
E = [1 2 3; 1 3 7; 1 4 3; 1 5 2; 2 3 6; 2 4 8; 2 5 5; 3 4 10; 3 5 inf; 4 5
4];

[T,V]=Kruskal(E)
```

Kao rezultat dobijamo:

```
T =


     1     5

     1     2

     1     4

     2     3



V =


    14
```

Vidimo da se rješenja poklapaju.

Primjer #3:

Potrebno je odrediti minimalno povezujuće stablo za graf čija je težinska matrica

$$W = \begin{pmatrix} 0 & 5 & 30 & \infty & \infty & \infty \\ 5 & 0 & 20 & 10 & \infty & \infty \\ 30 & 20 & 0 & 10 & 15 & \infty \\ \infty & 10 & 10 & 0 & 5 & 20 \\ \infty & \infty & 15 & 5 & 0 & 15 \\ \infty & \infty & \infty & 20 & 15 & 0 \end{pmatrix}$$

Rješenje:

Minimalno povezujuće stablo čine grane $1-2$, $2-4$, $3-4$, $4-5$ i $5-6$, a ukupna težina mu je 45.

```
E = [ 1 2 5; 1 3 30; 1 4 inf; 1 5 inf; 1 6 inf; 2 3 20; 2 4 10; 2 5 inf; 2
6 inf; 3 4 10; 3 5 15; 3 6 inf; 4 5 5; 4 6 20; 5 6 15];

[T,V]=Kruskal(E)
```

Kao rezultat dobijamo:

```
T =

     1     2

     4     5

     2     4

     3     4

     5     6


V =

    45
```

Vidimo da su rješenja ista kao u postavci.

Primjer #4:

Potrebno je odrediti minimalno povezujuće stablo za graf čija je težinska matrica

$$W = \begin{pmatrix} 0 & 3 & 10 & 2 & 4 & \infty & \infty \\ 3 & 0 & \infty & \infty & 7 & \infty & \infty \\ 10 & \infty & 0 & 12 & \infty & 15 & \infty \\ 2 & \infty & 12 & 0 & 6 & 4 & \infty \\ 4 & 7 & \infty & 6 & 0 & 5 & 2 \\ \infty & \infty & 15 & 4 & 5 & 0 & 3 \\ \infty & \infty & \infty & \infty & 2 & 3 & 0 \end{pmatrix}$$

Rješenje:

Minimalno povezujuće stablo čine grane $1-2$, $1-3$, $1-4$, $1-5$ (ili $4-6$), $5-7$ i $6-7$, a ukupna težina mu je 24.

```
E = [1 2 3; 1 3 10; 1 4 2; 1 5 4; 1 6 inf; 1 7 inf; 2 3 inf; 2 4 inf; 2 5
7; 2 6 inf; 2 7 inf; 3 4 12; 3 5 inf; 3 6 15; 3 7 inf; 4 5 6; 4 6 4; 4 7
inf; 5 6 5; 5 7 2; 6 7 3];

[T,V]=Kruskal(E)
```

Kao rezultat dobijamo:

```
T =

     1     4

     5     7

     1     2
```

```
     6     7

     1     5

     1     3



V =


    24
```

Vidimo da su rješenja ista.

Zadatak 2 (opis predstavljaju komentari u kodu)

```
function [X,V] = protokLP(C)
% treba nam velicina nase matrice C kako bismo mogli novu matricu kreirati
[m,n] = size(C);
% kreiramo vektor nula za naše C
c = zeros(1, m*n);
% dodajemo jedinice za n-1 varijabli tj. od j=2 do n samo za prvi cvor jer
% je to zapravo cilj funkcije
c(2:n) = ones(1,n-1);
% kreiramo matricu Aeq tako da sadrži sve promijenljive
Aeq = zeros(n-2,m*n);
% definisemo vektor nula tj. za k=2 do n-1, velicina je jednaka broju
% redova u matrici A
beq = zeros(1,n-2);

% prolazimo kroz nasu matricu, sada moramo stavljati 1 i -1 u matrici A
% k krece do n-1
for i = 1:(n-1)
    % sume koje se krecu do n
    for j = 2:n
        % ako unutar velike C matrice tj. ako je razlicito od nula tada se
        % odlucujemo da stavljamo 1 odnosno -1 na odgovarajuce mjesto
        if C(i, j) ~= 0
            % idemo do n-1 odnosno gledamo sve osim n, jer prednacimo za 1
            if j ~= n
                Aeq(j-1, j + n*(i-1)) = 1;
            end
            % gleda se svaki i osim prvog
            if i ~= 1
                Aeq(i-1, j + n*(i-1)) = -1;
            end
        end
    end
end

% definisemo nule za nas broj promijenljivih
lb=zeros(1,m*n);
% invertujemo nasu matricu tako da donja ogranicenja mozemo lagano pokupiti
% vektor sa C(:)'
C=C';
% kupimo donje granice, na osnovu ovog rjesavamo zadnja ogranicenja tj.
```

```
% ogranicenja segmenta
ub=C(:)';

% definisemo opcije slicno kao u postavci, samo sam napravio da ne izbacuje
% error
opcije = optimoptions('linprog', 'Algorithm', 'dual-simplex', 'display',
'off');
% poziv linprog
[X, V] = linprog(-c, [], [], Aeq, beq, lb, ub, opcije);
% dobijamo rezultat kao kolonu, zbog toga moramo reshapati u novu matricu
% gdje cemo dobiti rjesenje nase
X = reshape(X,[m,n])';
% negiramo vrijednost
V = -V;
end
```

Testirajmo nasu funkciju na nekoliko primjera. Naravno rješenja tj. matrica X koju dobijemo kao rezultat ne mora biti ista kao ponuđeno rješenje. Tj. moguće je da ima i drugih raspodjela koje daju isti maksimalni protok.

Primjer #1:

```
C=[0,3,0,3,0,0,0,0;0,0,4,0,0,0,0,0;0,0,0,1,2,0,0,0;0,0,0,0,2,6,0,0;
0,1,0,0,0,0,0,1;0,0,0,0,2,0,9,0;0,0,0,0,3,0,0,5;0,0,0,0,0,0,0,0];
[X,V]=protokLP(C)
```

Kao rezultat dobijamo:

```
X =


     0     2     0     3     0     0     0     0

     0     0     2     0     0     0     0     0

     0     0     0     1     1     0     0     0

     0     0     0     0     0     4     0     0

     0     0     0     0     0     0     0     1

     0     0     0     0     0     0     4     0

     0     0     0     0     0     0     0     4

     0     0     0     0     0     0     0     0



V =


     5
```

Vidimo da je drugačija raspodjela, ali ukupni protok je jednak.

Primjer #2:

Data je transportna mreža čija je matrica kapaciteta grana

$$C = \begin{pmatrix} 0 & 10 & 10 & 0 & 0 & 0 \\ 0 & 0 & 2 & 4 & 5 & 0 \\ 0 & 0 & 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 6 & 0 & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Potrebno je odrediti maksimalni protok od izvora do ponora i raspodjelu protoka po granama kojom se ostvaruje taj maksimalni protok

Rješenje:

Maksimalni protok iznosi 18. Jedna raspodjela protoka po granama koja daje takav protok je $f_{1,2} = 9$, $f_{1,3} = 9, f_{2,4} = 4, f_{2,5} = 5, f_{3,5} = 9, f_{4,6} = 8, f_{5,4} = 4, f_{5,6} = 10$. Moguće je da ima i drugih raspodjela koje daju isti maksimalni protok.

```
C=[0 10 10 0 0 0; 0 0 2 4 5 0; 0 0 0 0 9 0; 0 0 0 0 0 10; 0 0 0 6 0 10; 0 0
0 0 0 0];

[X,V]=protokLP(C)
```

Kao rezultat dobijamo:

```
X =


     0     9     9     0     0     0

     0     0     0     4     5     0

     0     0     0     0     9     0

     0     0     0     0     0     8

     0     0     0     4     0    10

     0     0     0     0     0     0



V =


    18
```

Vidimo da je raspodjela ista i da je ukupni protok jednak.

Primjer #3:

Data je transportna mreža čija je matrica kapaciteta grana

$$C = \begin{pmatrix} 0 & 5 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 2 & 0 \\ 0 & 1 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 1 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Potrebno je odrediti maksimalni protok od izvora do ponora i raspodjelu protoka po granama kojom se ostvaruje taj maksimalni protok

Rješenje:

Maksimalni protok iznosi 6. Jedna raspodjela protoka po granama koja daje takav protok je $f_{1,2} = 4, f_{1,3} = 2, f_{2,4} = 2, f_{2,5} = 2, f_{3,5} = 2, f_{4,6} = 2, f_{5,6} = 4$. Moguće je da ima i drugih raspodjela koje daju isti maksimalni protok.

```
C=[0 5 2 0 0 0; 0 0 0 3 2 0; 0 1 0 0 3 0; 0 0 0 0 0 2; 0 0 0 1 0 5; 0 0 0 0
0 0];

[X,V]=protokLP(C)

Kao rezultat dobijamo:

X =


     0     4     2     0     0     0

     0     0     0     2     2     0

     0     0     0     0     2     0

     0     0     0     0     0     2

     0     0     0     0     0     4

     0     0     0     0     0     0



V =


     6


Vidimo da je raspodjela ista i protok je isti.
```

Primjer #4:

Data je transportna mreža čija je matrica kapaciteta grana

$$C = \begin{pmatrix} 0 & 6 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 3 & 0 & 0 \\ 0 & 5 & 0 & 0 & 1 & 0 \\ 0 & 0 & 4 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Potrebno je odrediti maksimalni protok od izvora do ponora i raspodjelu protoka po granama kojom se ostvaruje taj maksimalni protok

Rješenje:

Maksimalni protok iznosi 4. Jedna raspodjela protoka po granama koja daje takav protok je $f_{1,2} = 3, f_{1,3} = 1, f_{2,3} = 5, f_{2,4} = 3, f_{3,2} = 3, f_{3,5} = 1, f_{4,6} = 3, f_{5,6} = 1$. Moguće je da ima i drugih raspodjela koje daju isti maksimalni protok.

```
C=[0 6 2 0 0 0; 0 0 3 3 0 0; 0 5 0 0 1 0; 0 0 4 0 0 3; 0 1 0 0 0 2; 0 0 0 0
0 0]; [X,V]=protokLP(C)
```

Kao rezultat dobijamo:

```
X =

     0     2     2     0     0     0

     0     0     3     3     0     0

     0     4     0     0     1     0

     0     0     0     0     0     3

     0     0     0     0     0     1

     0     0     0     0     0     0


V =

     4
```

Vidimo da raspodjela nije ista, ali protok jeste.

Zadatak 3 (opis predstavljaju komentari u kodu)

```
function [X,V] = protokEK(C)
% kupimo velicine matrice
[~,n] = size(C);
% definisemo kontrolnu varijablu
iteracijaGotovo = 0;
while ~iteracijaGotovo
    % definisemo 2 kolone matrice da bi pratili nas BFS
    redovi = zeros(n, 2);
    % krecemo od prvog reda, zato je pocetniIndex = 1
    pocetniIndex = 1;
    % definisemo prvi red sa n+1, jer je lakse vrsiti provjere
    redovi(1, 2) = 1;
    % uzimamo drugi red kao finalni, tj. ukoliko ne prosirimo ovaj raspon
    % više i ako ne dodjemo do zadnjeg red algoritam završava
    krajIndex = 2;
    while pocetniIndex < krajIndex
        % ovaj dio odgovara nasem BFS, ovdje vrsimo provjere i prosiravanje
        % indeksa tako da pokupimo sve redove, tj. dok ne dodjemo do
        % redovi(n,1) tj. zapravo zadnjeg reda koji odgovara cvoru do kojeg
        % ide protok
        i = redovi(pocetniIndex, 2);
        % prolazimo kroz nas red, da nadjemo nove redove, ovaj dio odgovara
        % ubacivanju grana
        for j = 1:n
            % samo gledamo one koje su >0
            if C(i, j) > 0
                % ukoliko ima takvih, gledamo jel vec uzeta taj red, tj. da
                % li je prvi element jednak nuli, ukoliko nije, uzimamo taj
                % red i povecavamo raspon povecavanjem krajIndex
                if redovi(j, 1) == 0
                    redovi(j, 1) = i;
                    redovi(krajIndex, 2) = j;
                    krajIndex = krajIndex + 1;
                end
            end
        end
        % povecavamo uvijek pocetniIndex u iteraciji
        pocetniIndex = pocetniIndex + 1;
        % provjeravamo u svakoj iteraciji jesmo li pokupili posljednji red,
        % ukoliko nismo nastavljamo sve dok ne dobijemo zadnji red
        if redovi(n, 1) ~= 0
            break;
        end
    end
    % ukoliko zadnji red nije jednak 0, tj. ukoliko smo pronasli
    % povecavajuci lanac tada moramo povecati tj. smanjtiti odredene
    % vrijednosti u matrici za nase delta, koje je jednako minimalnoj
    % vrijednosti u svakoj iteraciji
    if redovi(n, 1) ~= 0
        % kada imamo povecavajuci lanac, idemo od kraja matrice ka pocetka
        j = n;
        % kupimo naredni red, tj. naredni red ce biti prva vrijednost u
        % nase 2 kolone
        i = redovi(j, 1);
        % moramo jos naci minimalno delta
        delta = C(i, j);
        % tako vrtimo u krug sve dok ne dodjemo do prvog red i svaki put
        % provjeravamo je li i jednak 1, tj. jesmo li dosli do prvog reda
        while i ~= 1
```

```
            j = i;
            i = redovi(j, 1);
            % uvijek trazimo minimalno delta
            if C(i, j) < delta
                delta = C(i, j);
            end
        end
        % kada smo nasli minimalno delta, jos samo moramo ponoviti isti put
        % tako da na mjestima u gornjoj matrici oduzmemo vrijednosti tj. po
        % kojim smo dosli, i u inverznoj matrici dodamo vrijednosti, na
        % isti nacin idemo od kraja sve do prvog reda
        j = n;
        i = redovi(j, 1);
        C(i, j) = C(i, j) - delta;
        C(j, i) = C(j, i) + delta;
        % sve dok ne dodjemo do prvog reda
        while i ~= 1
            j = i;
            i = redovi(j, 1);
            % gornjoj matrici oduzimamo
            C(i, j) = C(i, j) - delta;
            % donjoj dodajemo
            C(j, i) = C(j, i) + delta;
        end
    else
        % ovaj slucaj govori samo da je gornji algoritam zavrsio, a da pri
        % tome nije moguce doci u zadnji red, ukoliko imamo taj slucaj
        % algoritam se zavrsava i zavrsen je postupak
        iteracijaGotovo = 1;
    end
end

% formiramo finalni X, tj. nase rjesenje formiramo tako što uzimamo
% vrijednosti donje matrice, posto imamo n vrijednosti znamo da na glavnoj
% matrici nece biti vrijednosti zato druga petlja krece od i+1, i kupimo
% vrijednosti (j,i)
X = zeros(n);
for i = 1:n
    for j = (i+1):n
        X(i, j) = C(j, i);
    end
end

% zadnji red u matrici predstavlja ukupni protok, moglo se isto pamtiti u
% svakoj iteraciji, medutim to isti mozemo procitati sumiranjem prvog
% reda u matrici, isto smo mogli uraditi i sumiranjem zadnjeg reda, ali sam
% samo zapisivao u gornjoj matrici da bude kao u postavci
prviRed = X(1,1:end);
V = sum(prviRed);

end
```

Testirajmo nasu funkciju na iste primjere ko što smo testirali zadatak 2. Naravno rješenja tj. matrica X koju dobijemo kao rezultat ne mora biti ista kao ponuđeno rješenje. Tj. moguće je da ima i drugih raspodjela koje daju isti maksimalni protok.

Primjer #1:

```
C=[0,3,0,3,0,0,0,0;0,0,4,0,0,0,0,0;0,0,0,1,2,0,0,0;0,0,0,0,2,6,0,0;
0,1,0,0,0,0,0,1;0,0,0,0,2,0,9,0;0,0,0,0,3,0,0,5;0,0,0,0,0,0,0,0];
[X,V]=protokEK(C)
```

Kao rezultat dobijamo:

```
X =


     0     2     0     3     0     0     0     0

     0     0     2     0     1     0     0     0

     0     0     0     1     1     0     0     0

     0     0     0     0     0     4     0     0

     0     0     0     0     0     2     3     1

     0     0     0     0     0     0     4     0

     0     0     0     0     0     0     0     4

     0     0     0     0     0     0     0     0



V =


     5
```

Vidimo da je raspodjela drugačija nego u zadatku 2, ali vidimo da je ukupni protok jednak kao u zadatku 2.

Primjer #2:

```
C=[0 10 10 0 0 0; 0 0 2 4 5 0; 0 0 0 0 9 0; 0 0 0 0 0 10; 0 0 0 6 0 10; 0 0
0 0 0 0];

[X,V]=protokEK(C)
```

Kao rezultat dobijamo:

```
X =


     0     9     9     0     0     0

     0     0     0     4     5     0

     0     0     0     0     9     0
```

```
     0     0     0     0     2     8
     0     0     0     0     0    10
     0     0     0     0     0     0


V =


    18
```

Vidimo da je raspodjela ista kao u zadatku 2, ujedno i ukupni protok.

Primjer #3:

```
C=[0 5 2 0 0 0; 0 0 0 3 2 0; 0 1 0 0 3 0; 0 0 0 0 0 2; 0 0 0 1 0 5; 0 0 0 0
0 0];

[X,V]=protokEK(C)
```

Kao rezultat dobijamo:

```
X =


     0     4     2     0     0     0
     0     0     1     2     2     0
     0     0     0     0     2     0
     0     0     0     0     1     2
     0     0     0     0     0     4
     0     0     0     0     0     0


V =


     6
```

Vidimo da raspodjela nije ista, ali zato ukupni protok jeste.

Primjer #4:

```
C=[0 6 2 0 0 0; 0 0 3 3 0 0; 0 5 0 0 1 0; 0 0 4 0 0 3; 0 1 0 0 0 2; 0 0 0 0
0 0];

[X,V]=protokEK(C)
```

Kao rezultat dobijamo:

```
X =


     0     3     1     0     0     0

     0     0     5     3     1     0

     0     0     0     4     1     0

     0     0     0     0     0     3

     0     0     0     0     0     1

     0     0     0     0     0     0



V =


     4
```

Vidimo da je ukupni protok jednak kao u zadatku 2, dok raspodjela nije.

Zadatak 5 [0.8 poena]

U sljedećoj tablici dati su podaci o svim aktivnostima nekog projekta, njihovim logičkim međuzavisnostima, kao i procjene trajanja pojedinih aktivnosti:

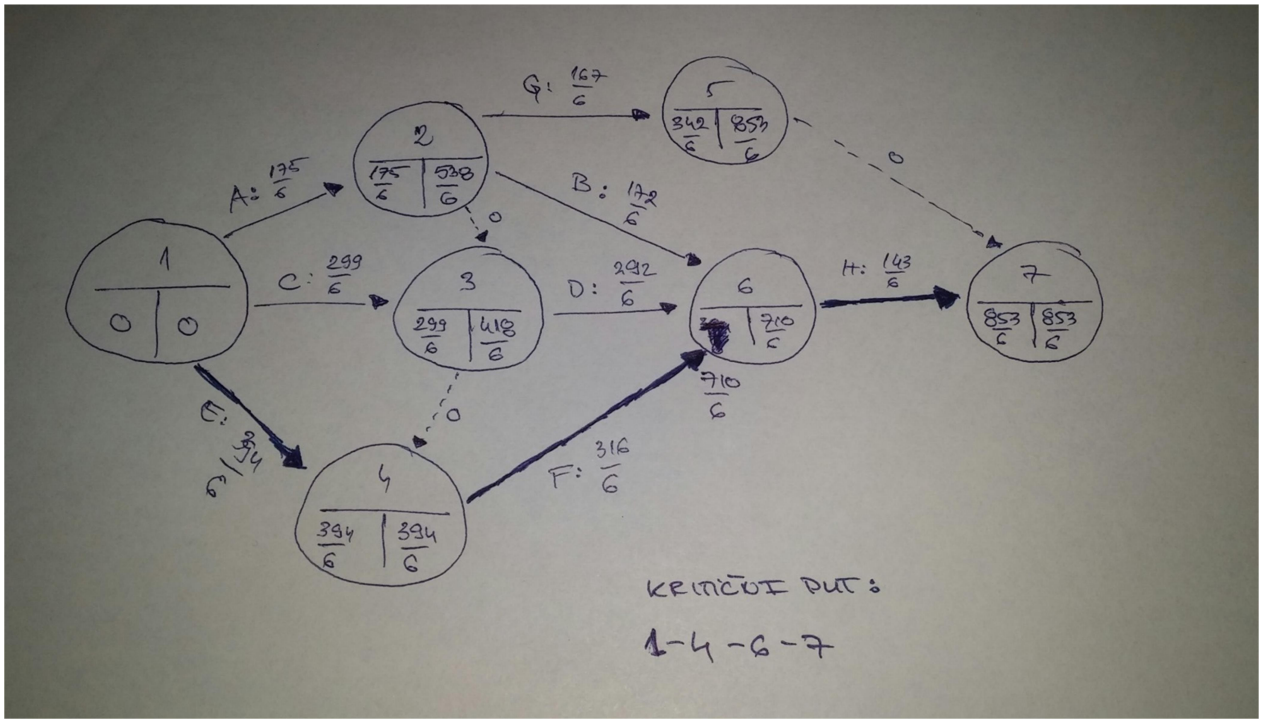
Aktivnost	Preduvjeti	Optimističko trajanje	Najvjerovatnije trajanje	Pesimističko trajanje
A	-	27	29	32
B	A	25	29	31
C	-	43	51	52
D	A, C	37	51	51
E	-	63	66	67
F	C, E	49	53	55
G	A	18	29	33
H	B, D, F	21	24	26

Formirajte mrežni dijagram projekta i pomoću PERT /TIME metoda odredite očekivana vremena početka i završetka svakog od događaja, najranije i nakasnije početke i završetke za sve aktivnosti, očekivane vremenske rezerve za sve aktivnosti, te vrijeme u kojem se projekat može završiti sa vjerovatnoćama od 25 %, zatim 75 % i skoro sigurno. Rezultate provedene analize prikažite u tabelarnoj formi. U svim slučajevima pretpostavite da će očekivano kritični put zaista biti kritičan. Potrebno je da predate izvještaj koji sadrži sve što se od Vas traži u `.pdf` formatu.

Odredimo očekivano vrijeme i varijansu:

Aktivnost	Preduvjeti	Očekivano vrijeme	Varijansa
A	-	175/6	25/36
B	A	172/6	36/36
C	-	299/6	81/36
D	A, C	292/6	196/36
E	-	394/6	16/36
F	C, E	316/6	36/36
G	A	167/6	225/36
H	B, D, F	143/6	25/36

Nacrtajmo sada mrežni dijagram:



Kritični put je označen podebljanom linijom. Kritični put je 1-4-6-7.

Događaj	$(Te)_i$	$(Tl)_i$	Rezerve
1	0	0	0
2	175/6	538/6	363/6
3	299/6	418/6	199/6
4	394/6	394/6	0
5	342/6	853/6	511/6
6	710/6	710/6	0
7	853/6	853/6	0

Vidimo da je kritični put 1-4-6-7. Očekivano trajanje projekta je: 142.167 ~ 142 sedmice. Na kritičnom putu imamo aktivnosti E, F, H, te na osnovu toga imamo varijansu:

$$s^2 = e^2 + f^2 + h^2 = \frac{16}{36} + 1 + \frac{25}{36} = \frac{77}{36} = 2.1389$$

Onda imamo standardnu devijaciju:

$$d = \sqrt{2.1389} = 1.4624$$

Odredimo sada vrijeme u kojem se projekat može završiti sa vjerovatnoćama od 25 %, zatim 75 % i skoro sigurno vrijeme 99.83%.

$$(Ts)_7 = (Te)_7 + d * fi^{-1}(0.25) = 142.167 - 1.4624 * 0.6745 = \ 141.1806112 \sim 141\ sedmicu$$

$$(Ts)_7 = (Te)_7 + d * fi^{-1}(0.75) = 142.167 + 1.4624 * 0.6745 = \ 143.1533888 \sim 143\ sedmice$$

$$(Ts)_7 = (Te)_7 + d * fi^{-1}(0.9983) = 142.167 + 1.4624 * 3 = \ 146.5542 \sim 146\ sedmica$$

Vidimo da se projekat, sa vjerovatnoćama 25%,75%,99.83% može završiti završniti respektivno za 141,143,146 sedmica.

Zadatak 6 [1.1 poen]

U sljedećoj tablici dati su podaci o svim aktivnostima nekog projekta, njihovim logičkim međuzavisnostima, kao i procjene trajanja i troškova pojedinih aktivnosti (za svaku aktivnost data su normalne i usiljene vrijednosti trajanja i troškova).

Aktivnost	Preduvjeti	$t^{(n)}$	$t^{(u)}$	$c^{(n)}$	$c^{(u)}$
A	–	3	3	7	7
B	A	2	2	12	12
C	A	7	7	13	13
D	B	11	6	3	28
E	C	9	7	10	22
F	C, D	5	3	6	20
G	E	10	7	7	25

Vaš zadatak je da:

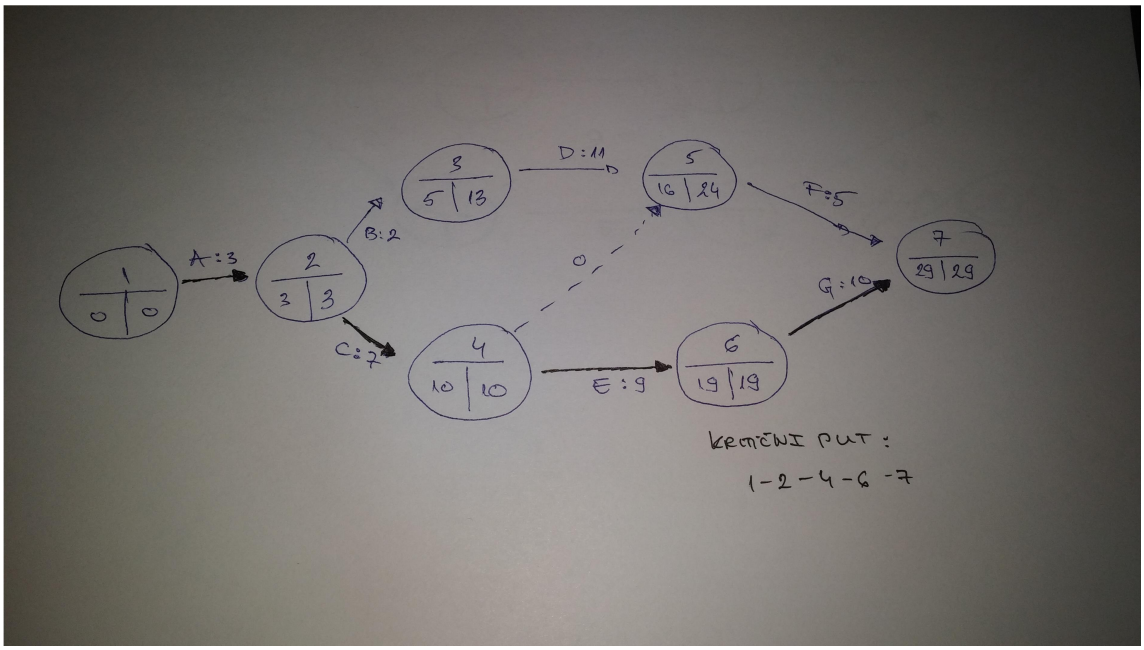
a. Izvršite analizu strukture i analizu vremena prema CPM metodi za ovaj projekat uz najekonomičniju realizaciju. Na mrežnom dijagramu treba da se vide najranija i najkasnija vremena započinjanja svih događaja, te kritični put. Potrebno je također izračunati i sve tipove vremenskih rezervi za sve aktivnosti (rezultate prikazati tabelarno), kao i cijenu projekta pri najekonomičnijoj realizaciji. **[0.4 poena]**
b. Odredite plan realizacije projekta sa minimalnim trajanjem, uz najmanje moguće poskupljenje projekta. Obavezno naglasite koliko će iznositi novi troškovi projekta. **[0.4 poena]**
c. Provjerite rezultat pod b. tako što ćete problem modelirati koristeći modele linearnog programiranja i rješavanjem odgovarajućih problema uz pomoć `linprog` funkcije u MATLAB-u. Obavezno naznačite šta su bili ulazni a šta izlazni podaci prilikom korištenja ove funkcije. **[0.3 poena]**

Potrebno je da predate izvještaj koji sadrži sve što se od Vas traži u `.pdf` formatu.

a) Cijena projekta pri najekonomičnijoj realizaciji je očigledno ona suma koja je jednaka sumi troškova prilikom normalnih trajanja aktivnosti. To predstavlja suma u našoj trećoj koloni u tabeli i onda iznosi (pod predpostavkom da je jedinica n.j.):

S = 7 + 12 + 13 + 3 + 10 + 6 + 7 = 58 n.j.

Nacrtajmo sada mrežni dijagram:



Vidimo da je kritični put 1-2-4-6-7.

Prikažimo još sve tipove vremenskih rezervni tabelarno:

Aktivnosti	[i-j]	$t_{i,j}$	$t_i^{(0)}$	$t_j^{(0)}$	$t_i^{(1)}$	$t_j^{(1)}$	$R_{t_{i,j}}$	$R_{s_{i,j}}$	$R_{n_{i,j}}$
A	1-2	3	0	3	0	3	0	0	0
B	2-3	2	3	5	3	13	8	0	0
C	2-4	7	3	10	3	10	0	0	0
D	3-5	11	5	16	13	24	8	0	-8
E	4-6	9	10	19	10	19	0	0	0
fiktivna	4-5	0	10	16	10	24	14	6	6
F	5-7	5	16	29	24	29	8	8	0
G	6-7	10	19	29	19	29	0	0	0

b) Odrediti ćemo plan realizacije projekta sa minimalnim trajanjem, uz najmanje moguće poskupljenje projekta tabelarno.

Aktivnost	$t^{(n)}$	$t^{(u)}$	alfa	Iteracija		
				I	II	III
A	3	3	-	3	3	3
B	2	2	-	2	2	2
C	7	7	-	7	7	7
D	11	6	5	11	11	11
E	9	7	6	9	7	7
F	5	3	7	5	5	5
G	10	7	6	10	10	7

Put	Iteracija		
	I	II	III
P1: A->B->D->F	21	21	21
P2: A->C->E->G	29	27	24
Troškovi:	58	70	88
Krati se:	E	G	-

Vidimo da se više ne može poboljšati kritični put, shodno tome on ostaje i jedini kritični put.

Također vidimo da smo dobili minimalno trajanje koje iznosi 24 sedmice (pod pretpostavkom da ovo predstavljaju sedmice), a da smo cifru koju smo morali platiti za to je 88 n.j tj. novi troškovi projekta će iznositi 88 n.j. dok za 29 sedmica smo imali troškove projekta 58 n.j.

Ovo predstavlja minimalno trajanje projekta.

c)

Provjerimo sada naše rezultate sa linprog funkcijom. Naravno samo ćemo smanjivati broj sedmica po kritičnom putu. Pozovimo sljedeći kod:

```
A = [1 0 0 0 0 0 0;
0 1 0 0 0 0 0;
0 0 1 0 0 0 0;
0 0 0 1 0 0 0;
0 0 0 0 1 0 0;
0 0 0 0 0 1 0;
0 0 0 0 0 0 1];
Tn = [3 2 7 11 9 5 10];
Tu = [3 2 7 6 7 3 7];
Cn = [7 12 13 3 10 6 7];
Cu = [7 12 13 28 22 20 25];
Calfa = (Cu - Cn) ./ (Tn - Tu);
% zanemarujemo NaN koje nastanu radi T-ova koji su jednaki
for i=1:size(Calfa,2)
 if isnan(Calfa(i))
  Calfa(i) = 0;
 end
end

B = [3;2;7;11;9;5;10];
% iz linproga cemo dobiti nase sedmice, tj. koliko traje projekat, ali tako
da gleda samo kriticni put
[x, y] = linprog(Calfa,A,B,[], [],[3 2 7 11 7 5 7],[3 2 7 11 9 5 10]);
value = sum(Cn + Calfa .* Tn) - y;
% prikaz nasih vrijednosti tj. najekonomicnije realizacije projekta
x
value
```

Kao rezultat poziva ovog koda dobijamo sljedeće ispise:

```
x =

     3

     2

     7

    11

     7

     5

     7

value =


    88
```

Vidimo da su x-ovi trajanja aktivnosti, dok value predstavlja ukupni trošak. Vidimo da je isto kao pod b.