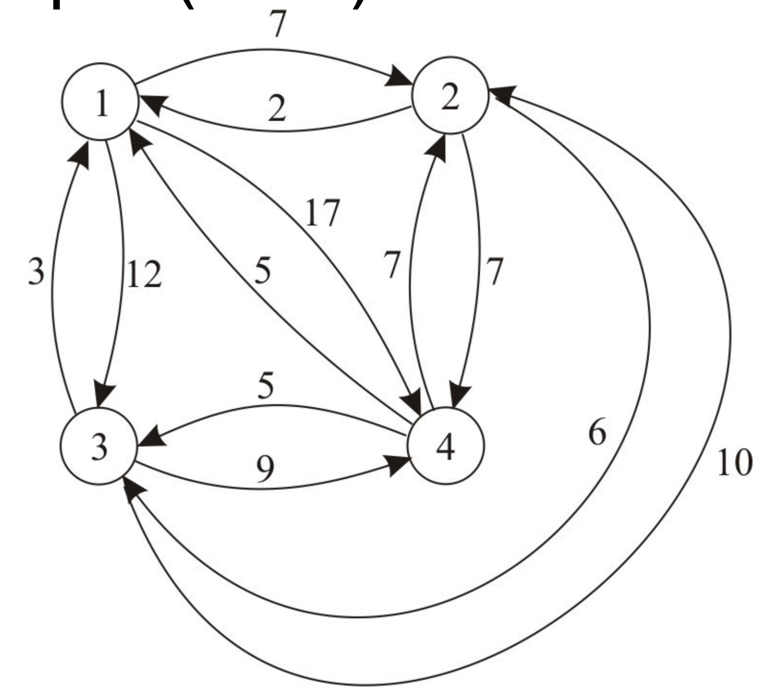
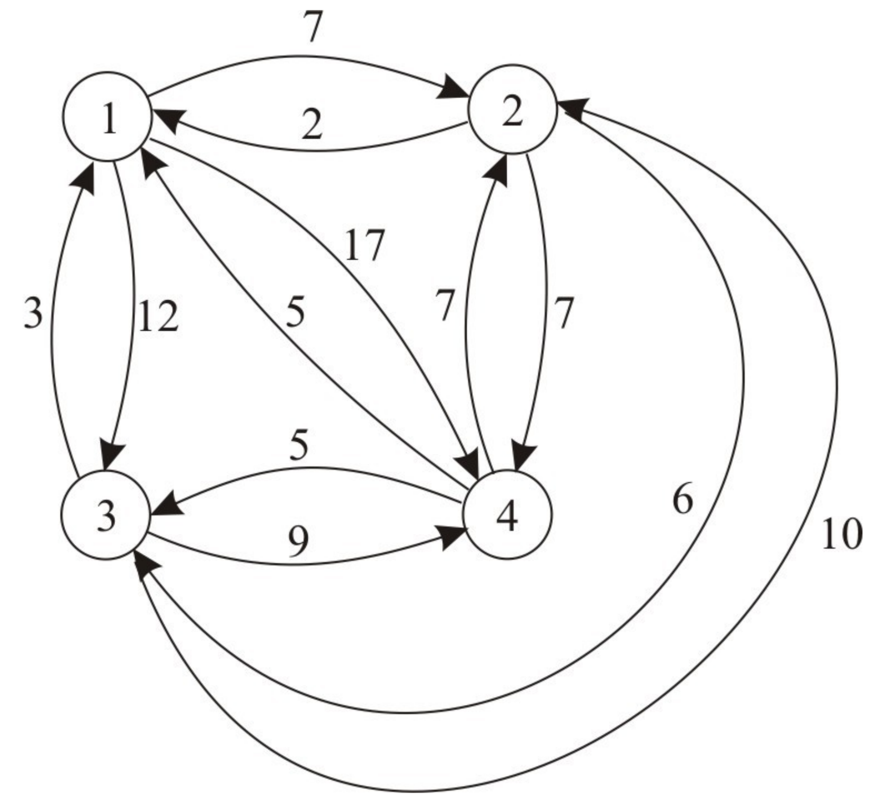
Trgovski potnik

- V vhodnem usmerjenem obtežen grafu želimo narediti najcenejši obhod skozi vsa vozlišča in se vrniti na začetno vozlišče
  - Vsa vozlišča obiščemo točno enkrat
  - Zanima nas najcenejša krožna pot (cikel)



Predstavitev vhodnega problema

$$C = \begin{bmatrix} 0 & 7 & 12 & 17 \\ 2 & 0 & 6 & 7 \\ 3 & 10 & 0 & 9 \\ 5 & 7 & 5 & 0 \end{bmatrix}$$

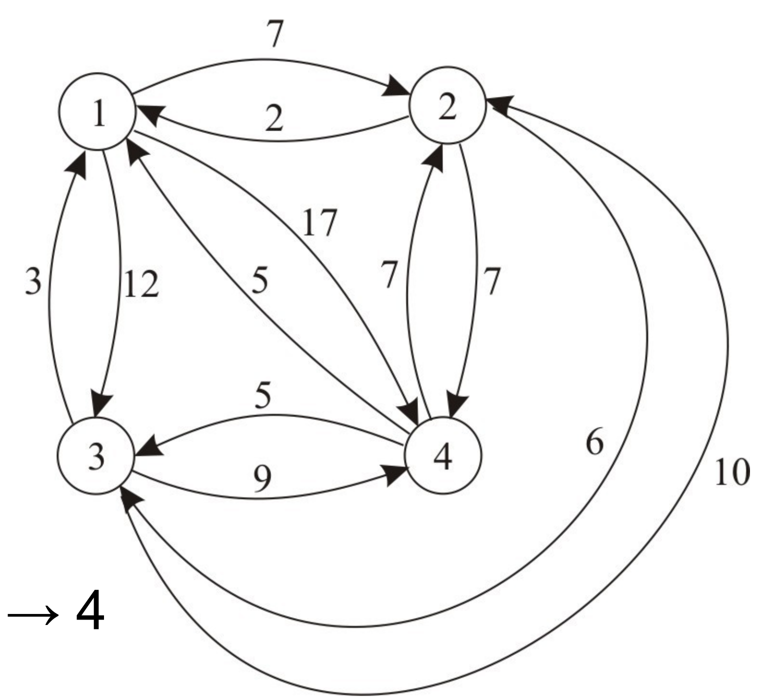


Naivna metoda iskanja najcenejšega obhoda

- Dobimo vse možne poti in izberemo najmanjšo?
- Obhod oz. cikel: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1 \;=\; 2 \rightarrow 3 \rightarrow 4 \rightarrow 1 \rightarrow 2$
  - Vseeno kje začnemo iskanje
  - Predpostavimo, da iskanje pričnemo v vozlišču 1
    - $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1$
    - $1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1$
    - $1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 1$
    - $1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 1$
    - $1 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 1$
    - $1 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1$

6 možnih poti: 3!

24 izračunov cen povezav (6x4)



  - Redundantni izračun cene poti
    - $1 \rightarrow 4$, $2 \rightarrow 1$, …
    - Pri večjih grafih še bolj očitno $1 \rightarrow x \rightarrow y \rightarrow z \rightarrow 4$

Predstavitev problema za dinamično programiranje

- V – množica vseh vozlišč
- Osnovna formulacija krožne poti:
  - Krožna pot do k preko ostalih vozlišč: $1 \rightarrow k \rightarrow \ldots \rightarrow 1$; $k \in V\text{-}\{1\}$
    - Preglednejši zapis: $1 \rightarrow k \rightarrow S \rightarrow 1$, kjer je S – **množica** ostalih vozlišč: $S = V\text{-}\{1, k\} \rightarrow S \subset V$
  - Najkrajšo krožno pot dobimo tako,da izberemo takšen k, da je cena poti $(1 \rightarrow k) + (k \rightarrow \ldots V\text{-}\{1,k\} \ldots \rightarrow 1)$ minimalna

Formalni zapis najkrajše poti z Bellmanovo enačbo

- Splošen rekurziven zapis najkrajše krožne poti s funkcijo *g*(*i*, S):
  - Cena najkrajše poti iz *i* preko vozlišč v množici S do začetnega vozlišča

$$g(i, \mathrm{S}) = \min_{j \in \mathrm{S}} \left\{ \mathrm{C}[i, j] + g\left(j, \mathrm{S} - \{j\}\right) \right\}$$

Učinkovito reševanje Bellmanove enačbe

- Z Bellmanovo enačbo iščemo najcenejši obhod od vozlišča 1 do vozlišča 1 preko vseh ostalih vozlišč

$$g\left(1, V-\{1\}\right) = \min_{2 \le k \le n} \left\{ C[1,k] + g\left(k, V-\{1,k\}\right) \right\}$$

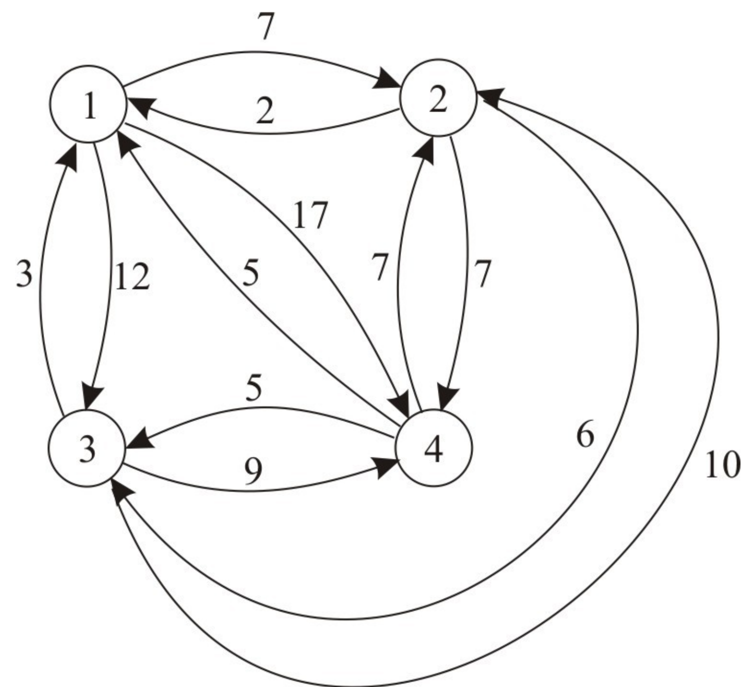
$$g\left(i, S\right) = \min_{j \in S} \left\{ C[i,j] + g\left(j, S-\{j\}\right) \right\}$$

- Reševanje Bellmanove enačbe brez podvajanja izračuna z iskanjem poti od konca proti začetku:
  1. Izvedba Bellmanove enačbe za direktne povezave do 1
  2. Izvedba Bellmanove enačbe za povezave do 1 preko dveh povezav
  3. Izvedba Bellmanove enačbe za povezave do 1 preko treh povezav
  4. ....
- V vsakem koraku uporabimo rešitev iz prejšnjega koraka: ne podvajamo izračuna
- Med potmi, ki gredo skozi ista vozlišča (množica S), izberemo najcenejšo in tako sproti izločamo slabše dele rešitve

Primer

1. Poišči neposredne poti od vseh vozlišč do vozlišča 1
2. Poišči poti iz vseh vozlišč preko enega vozlišča do vozlišča 1
3. Poišči poti iz vseh vozlišč preko dveh vozlišč do vozlišča 1
4. Poišči poti iz vseh vozlišč preko treh vozlišč do vozlišča 1

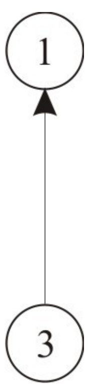
- V vsakem koraku uporabimo rešitev iz prejšnjega koraka



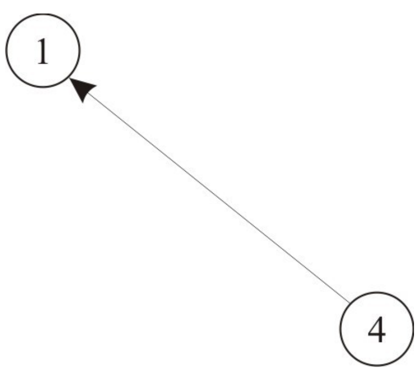
Primer – 1. nivo

Iskanje poti z brez vmesnih vozlišč

$$g(2,[\,])=C[2,1]=2$$

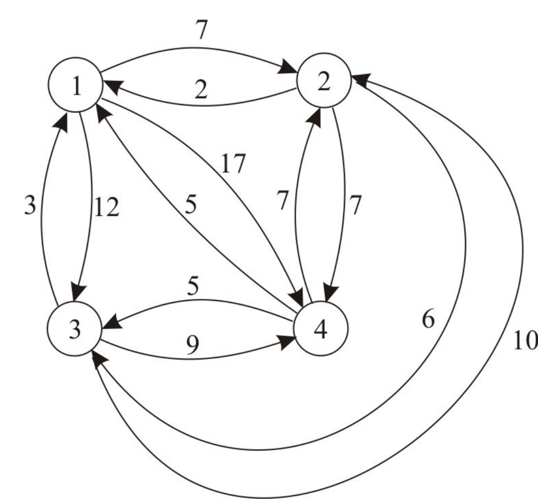


$$g(3,[\,])=C[3,1]=3$$



$$g(4,[\,])=C[4,1]=5$$

$$g(i,S)=\min_{j\in S}\left\{C[i,j]+g\left(j,S-\{j\}\right)\right\}$$



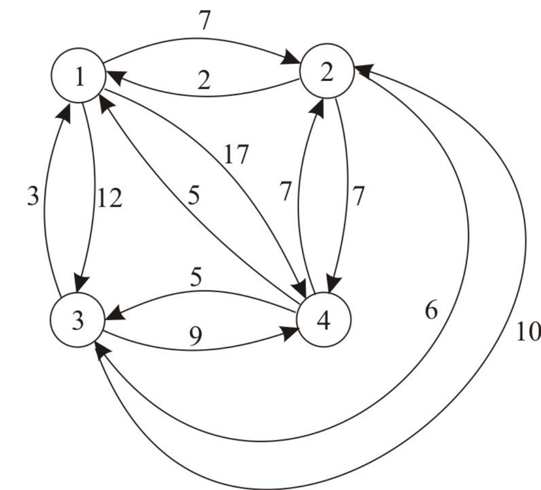
Primer – 1. nivo

Rešitev (brez vmesnih vozlišč)

$$g(2,[\,])=2$$

$$g(3,[\,])=3$$

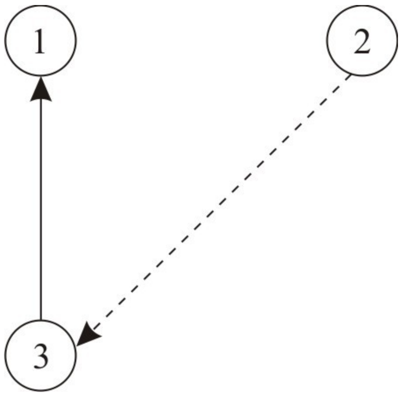
$$g(4,[\,])=5$$



$$g(i,S)=\min_{j\in S}\left\{C[i,j]+g\left(j,S-\{j\}\right)\right\}$$

Primer – 2. nivo

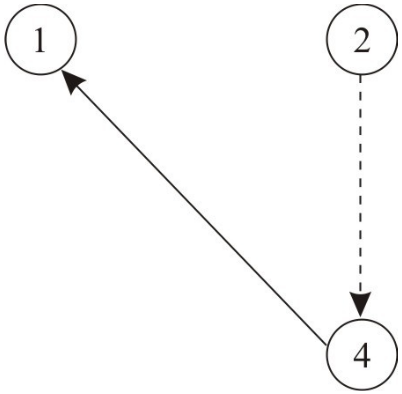
Iskanje poti iz vozlišča 2 z enim vmesnim vozliščem



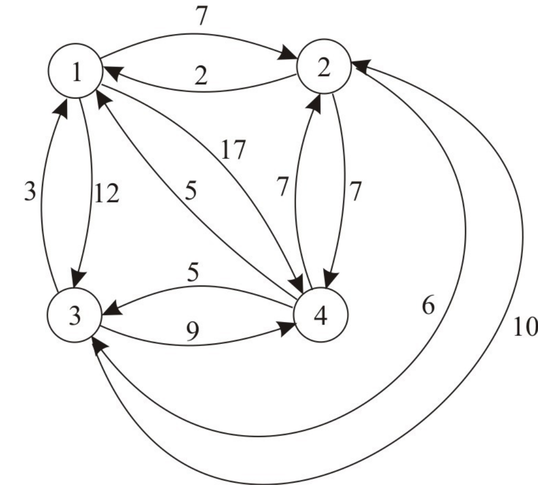
$$g(2,[3]) = C[2,3] + g(3,[\,]) = 6 + 3 = 9$$

Rešitev iz prejšnjega koraka

$$g(2,[\,]) = 2$$
$$g(3,[\,]) = 3$$
$$g(4,[\,]) = 5$$



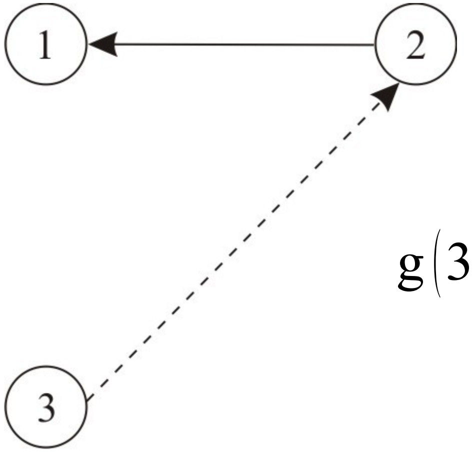
$$g(2,[4]) = C[2,4] + g(4,[\,]) = 7 + 5 = 12$$



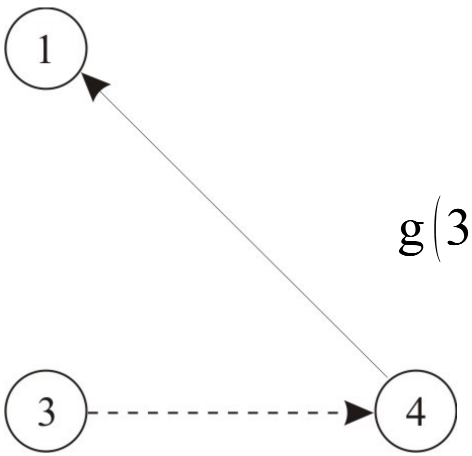
$$g(i,S) = \min_{j \in S} \left\{ C[i,j] + g\left(j, S - \{j\}\right) \right\}$$

Primer – 2. nivo

Iskanje poti iz vozlišča 3 z enim vmesnim vozliščem



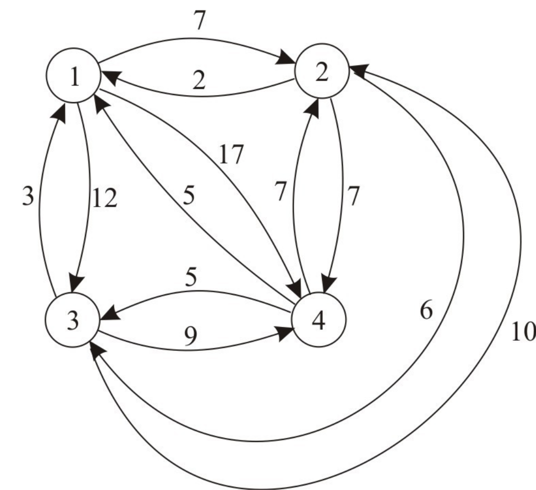
$$g(3,[2]) = C[3,2] + g(2,[]) = 10 + 2 = 12$$



$$g(3,[4]) = C[3,4] + g(4,[]) = 9 + 5 = 14$$

Rešitev iz prejšnjega koraka

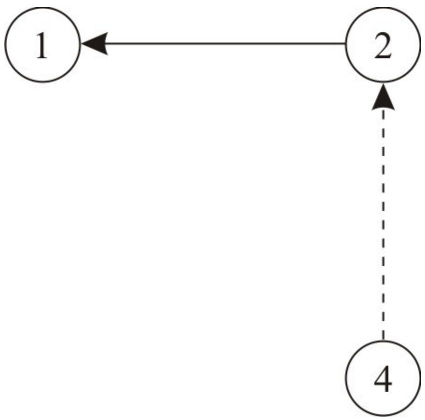
$$g(2,[]) = 2$$
$$g(3,[]) = 3$$
$$g(4,[]) = 5$$



$$g(i,S) = \min_{j \in S} \{ C[i,j] + g(j, S - \{j\}) \}$$

Primer – 2. nivo

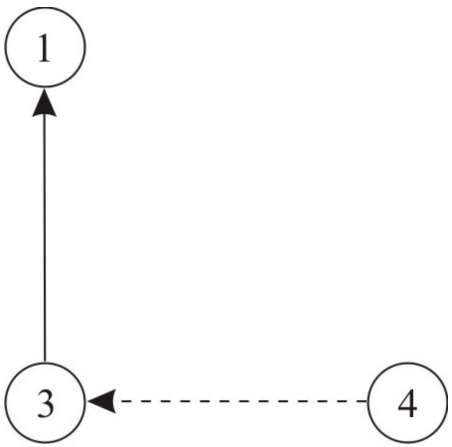
Iskanje poti iz vozlišča 4 z enim vmesnim vozliščem



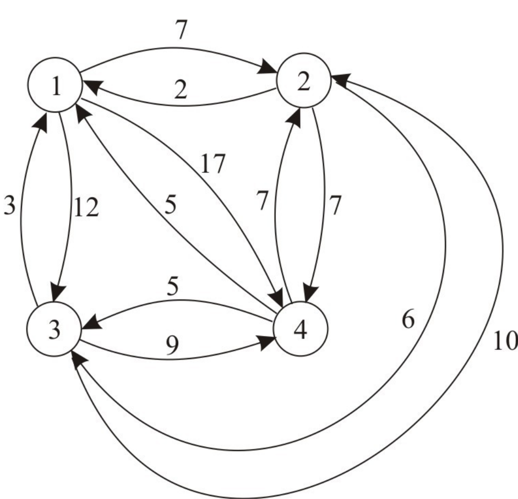
$$g(4,[2]) = C[4,2] + g(2,[\,]) = 7 + 2 = 9$$

Rešitev iz prejšnjega koraka

$$g(2,[\,]) = 2$$
$$g(3,[\,]) = 3$$
$$g(4,[\,]) = 5$$



$$g(4,[3]) = C[4,3] + g(3,[\,]) = 5 + 3 = 8$$

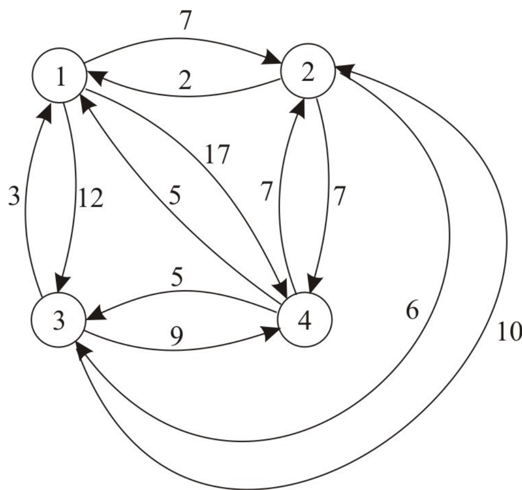


$$g(i,S) = \min_{j \in S} \{ C[i,j] + g(j, S - \{j\}) \}$$

Primer – 2. nivo

Rešitev

$$g(2,[3])=9$$
$$g(2,[4])=12$$
$$g(3,[2])=12$$
$$g(3,[4])=14$$
$$g(4,[2])=9$$
$$g(4,[3])=8$$



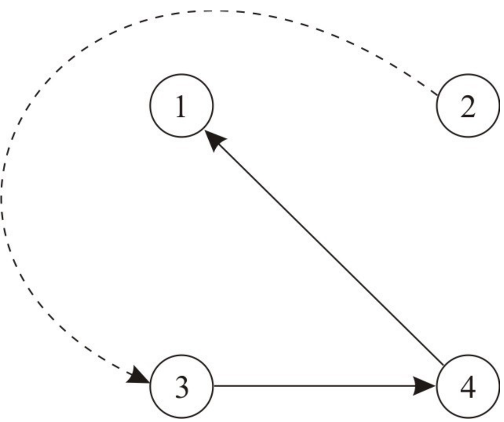
$$g(i,S)=\min_{j\in S}\{C[i,j]+g(j,S-\{j\})\}$$

Primer – 3. nivo

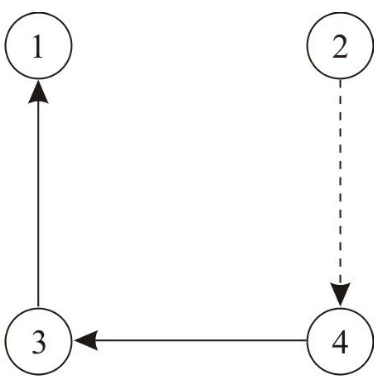
Iskanje poti iz vozlišča 2 z dvema vmesnima vozliščema

Rešitev iz prejšnjega koraka

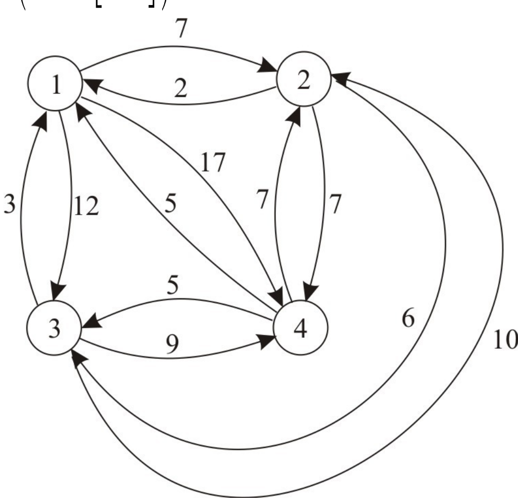
$$g(2,[3])=9$$
$$g(2,[4])=12$$
$$g(3,[2])=12$$
$$g(3,[4])=14$$
$$g(4,[2])=9$$
$$g(4,[3])=8$$



$$g(2,[3,4])=C[2,3]+g(3,[4])=6+14=20$$



$$g(2,[3,4])=C[2,4]+g(4,[3])=7+8=15$$



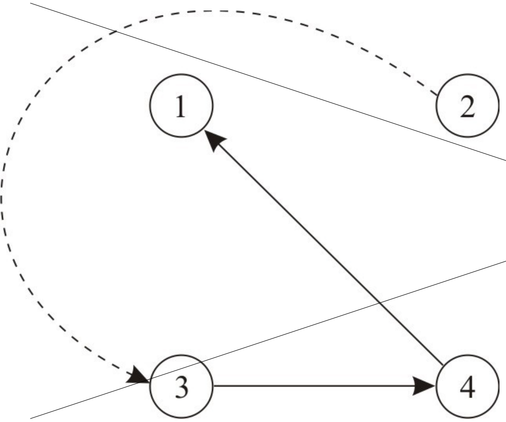
$$g(i,S)=\min_{j\in S}\{C[i,j]+g(j,S-\{j\})\}$$

Primer – 3. nivo

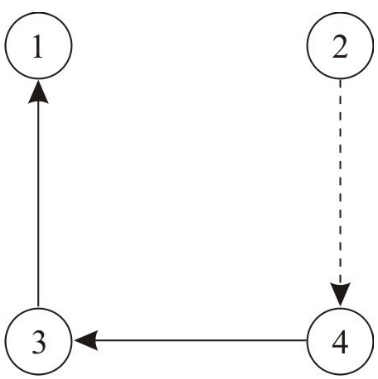
Iskanje poti iz vozlišča 2 z dvema vmesnima vozliščema

Rešitev iz prejšnjega koraka

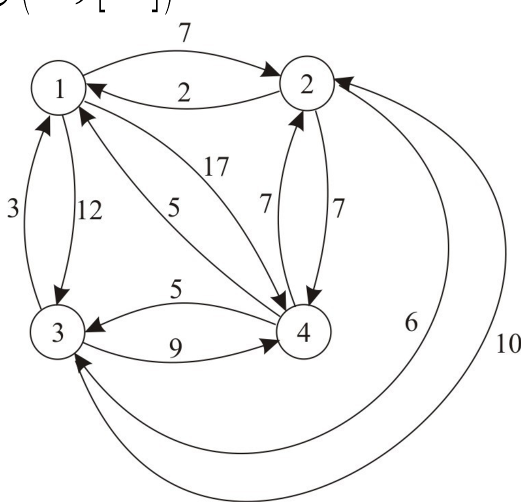
$$g(2,[3])=9$$
$$g(2,[4])=12$$
$$g(3,[2])=12$$
$$g(3,[4])=14$$
$$g(4,[2])=9$$
$$g(4,[3])=8$$



$$\cancel{g(2,[3,4])=C[2,3]+g(3,[4])=6+14=20}$$



$$g(2,[3,4])=C[2,4]+g(4,[3])=7+8=15$$

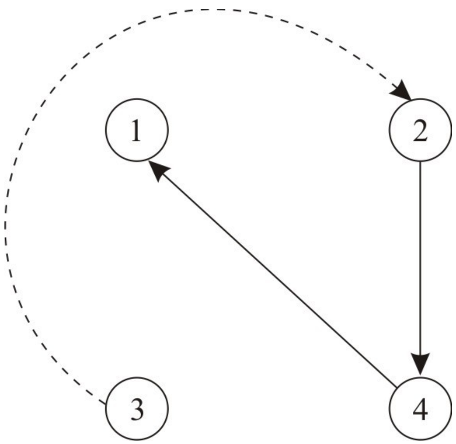


$$g(i,S)=\min_{j\in S}\{C[i,j]+g(j,S-\{j\})\}$$

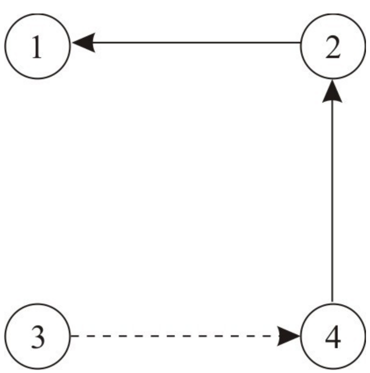
$$g(2,[3,4])=15$$

Primer – 3. nivo

Iskanje poti iz vozlišča 3 z dvema vmesnima vozliščema



$$g\left(3,\left[2,4\right]\right)=C\left[3,2\right]+g\left(2,\left[4\right]\right)=10+12=22$$

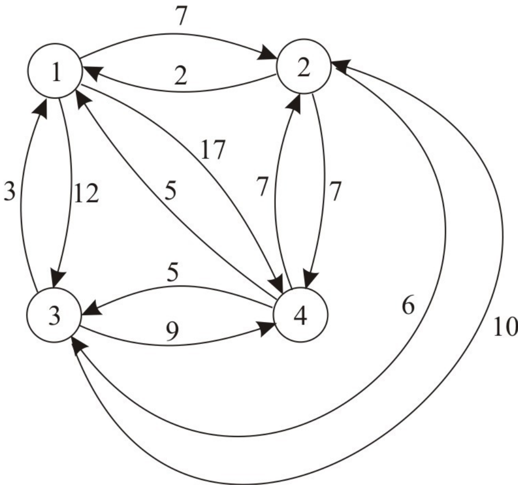


$$g\left(3,\left[2,4\right]\right)=C\left[3,4\right]+g\left(4,\left[2\right]\right)=9+9=18$$

$$g\left(i,S\right)=\min_{j\in S}\left\{C\left[i,j\right]+g\left(j,S-\left\{j\right\}\right)\right\}$$

Rešitev iz prejšnjega koraka

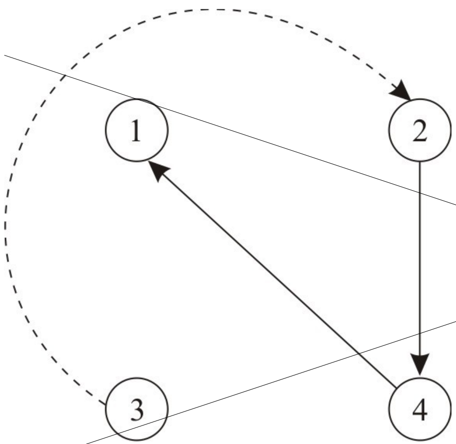
$$g\left(2,\left[3\right]\right)=9$$
$$g\left(2,\left[4\right]\right)=12$$
$$g\left(3,\left[2\right]\right)=12$$
$$g\left(3,\left[4\right]\right)=14$$
$$g\left(4,\left[2\right]\right)=9$$
$$g\left(4,\left[3\right]\right)=8$$



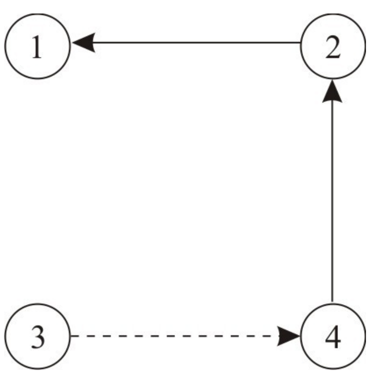
Primer – 3. nivo

Iskanje poti iz vozlišča 3 z dvema vmesnima vozliščema

Rešitev iz prejšnjega koraka

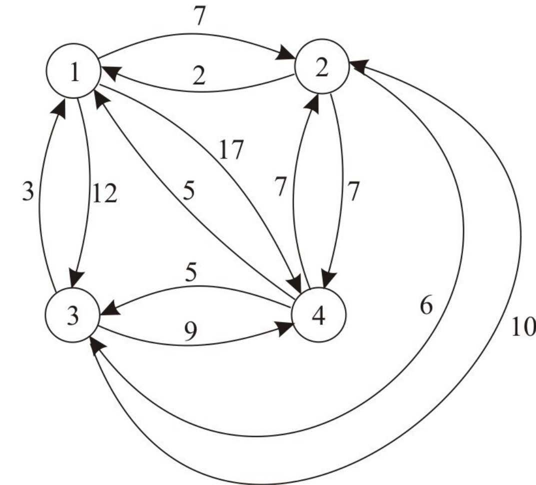


$$g(3,[2,4]) = C[3,2] + g(2,[4]) = 10 + 12 = 22$$



$$g(3,[2,4]) = C[3,4] + g(4,[2]) = 9 + 9 = 18$$

$$g(2,[3]) = 9$$
$$g(2,[4]) = 12$$
$$g(3,[2]) = 12$$
$$g(3,[4]) = 14$$
$$g(4,[2]) = 9$$
$$g(4,[3]) = 8$$



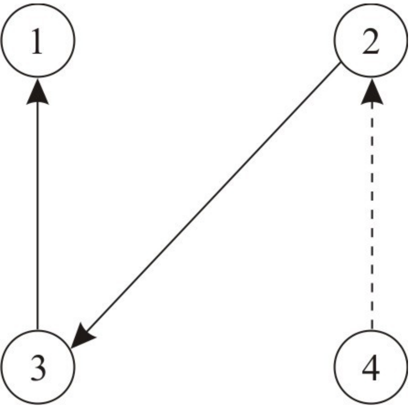
$$g(i,S) = \min_{j \in S} \{ C[i,j] + g(j, S - \{j\}) \}$$

$$g(3,[2,4]) = 18$$

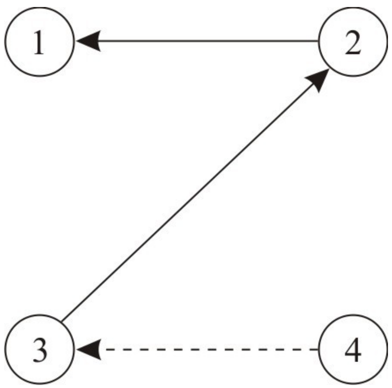
Primer – 3. nivo

Iskanje poti iz vozlišča 4 z dvema vmesnima vozliščema

Rešitev iz prejšnjega koraka

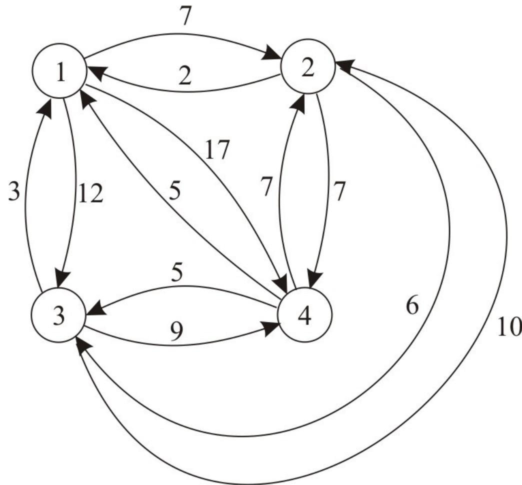


$$g(4,[2,3]) = C[4,2] + g(2,[3]) = 7 + 9 = 16$$



$$g(4,[2,3]) = C[4,3] + g(3,[2]) = 5 + 12 = 17$$

$$g(2,[3]) = 9$$
$$g(2,[4]) = 12$$
$$g(3,[2]) = 12$$
$$g(3,[4]) = 14$$
$$g(4,[2]) = 9$$
$$g(4,[3]) = 8$$

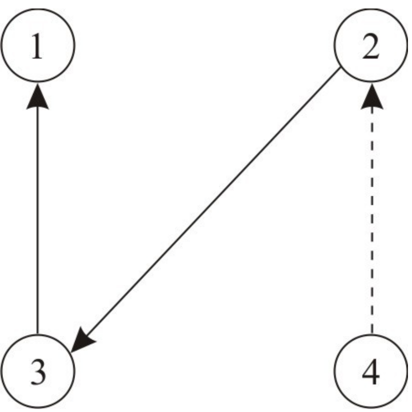


$$g(i,S) = \min_{j \in S} \{C[i,j] + g(j, S - \{j\})\}$$

Primer – 3. nivo

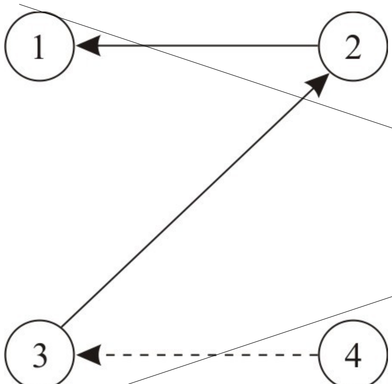
Iskanje poti iz vozlišča 4 z dvema vmesnima vozliščema

Rešitev iz prejšnjega koraka

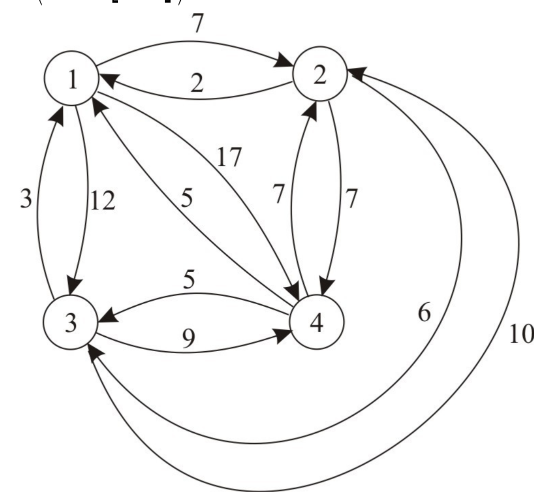


$$g(4,[2,3]) = C[4,2] + g(2,[3]) = 7 + 9 = 16$$

$$g(2,[3]) = 9$$
$$g(2,[4]) = 12$$
$$g(3,[2]) = 12$$
$$g(3,[4]) = 14$$
$$g(4,[2]) = 9$$
$$g(4,[3]) = 8$$



$$\cancel{g(4,[2,3]) = C[4,3] + g(3,[2]) = 5 + 12 = 17}$$



$$g(i,S) = \min_{j \in S} \{ C[i,j] + g(j, S - \{j\}) \}$$

$$g(4,[2,3]) = 16$$

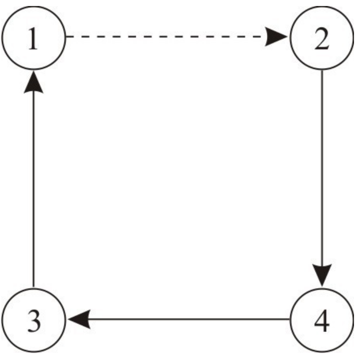
Primer – 3. nivo

Rešitev

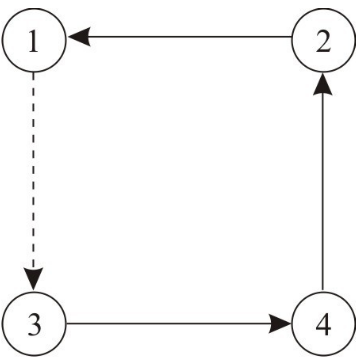
$$g\left(2,\left[3,4\right]\right)=15$$
$$g\left(3,\left[2,4\right]\right)=18$$
$$g\left(4,\left[2,3\right]\right)=16$$

Primer – 4. nivo

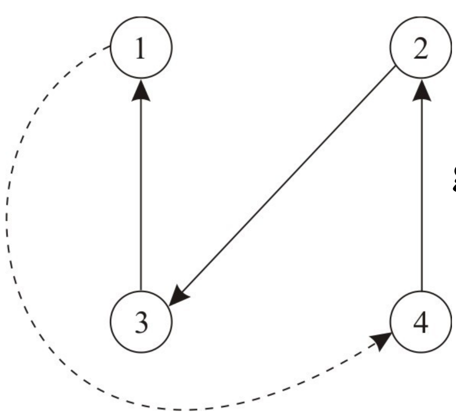
Zaključevanje krožne poti



$$g(1,[2,3,4]) = C[1,2] + g(2,[3,4]) = 7 + 15 = 22$$



$$g(1,[2,3,4]) = C[1,3] + g(3,[2,4]) = 12 + 18 = 30$$

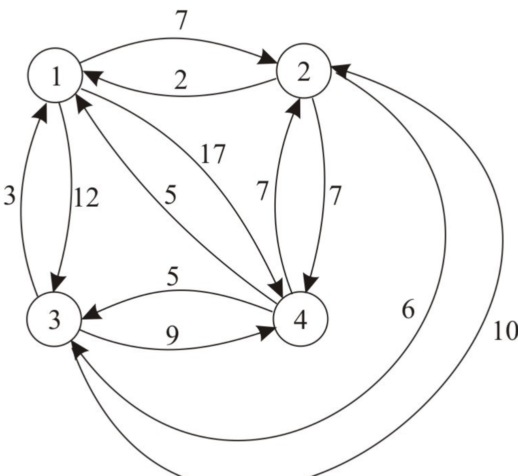


$$g(1,[2,3,4]) = C[1,4] + g(4,[2,3]) = 17 + 16 = 33$$

Rešitev iz prejšnjega koraka

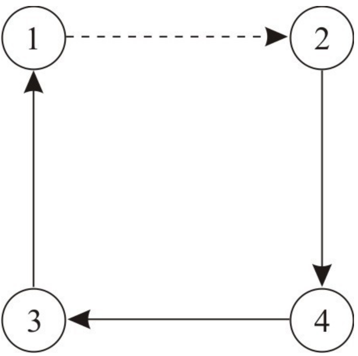
$$g(2,[3,4]) = 15$$
$$g(3,[2,4]) = 18$$
$$g(4,[2,3]) = 16$$

$$g(i,S) = \min_{j \in S} \{C[i,j] + g(j, S - \{j\})\}$$

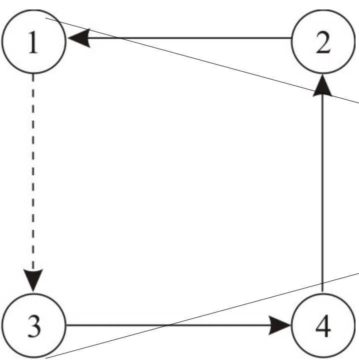


Primer – 4. nivo

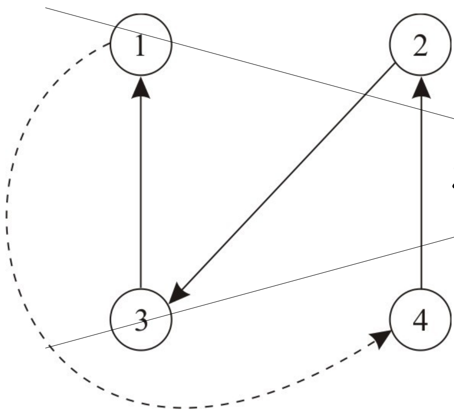
Zaključevanje krožne poti



$$g\left(1,\left[2,3,4\right]\right)=C\left[1,2\right]+g\left(2,\left[3,4\right]\right)=7+15=22$$



$$g\left(1,\left[2,3,4\right]\right)=C\left[1,3\right]+g\left(3,\left[2,4\right]\right)=12+18=30$$



$$g\left(1,\left[2,3,4\right]\right)=C\left[1,4\right]+g\left(4,\left[2,3\right]\right)=17+16=33$$

$$g\left(1,\left[2,3,4\right]\right)=22$$

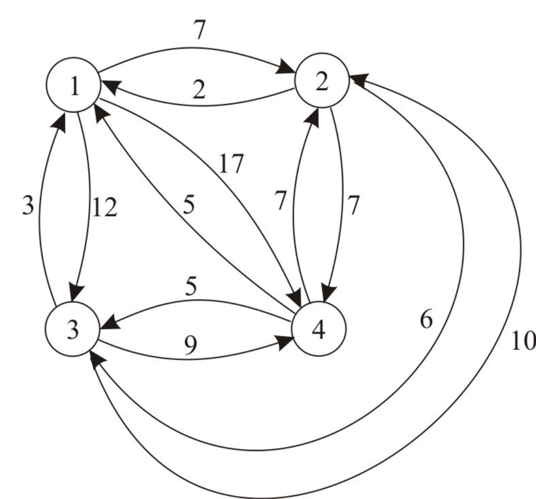
Rešitev iz prejšnjega koraka

$$g\left(2,\left[3,4\right]\right)=15$$

$$g\left(3,\left[2,4\right]\right)=18$$

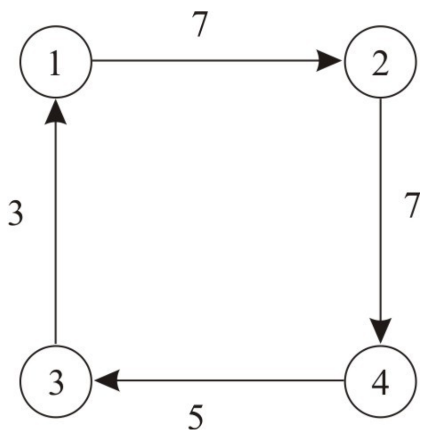
$$g\left(4,\left[2,3\right]\right)=16$$

$$g\left(i,S\right)=\min_{j\in S}\left\{C\left[i,j\right]+g\left(j,S-\left\{j\right\}\right)\right\}$$



Primer – 4. nivo

Rešitev



$$g\left(1,\left[2,3,4\right]\right)=22$$

Implementacija - predstavitev Bellmanove enačbe

- Predstavitev ene poti oziroma g(*i*,S):

iz_vozlišča:
v_vozlišče:
cena:
množica:

$$g(i,S)=\min_{j\in S}\{C[i,j]+g(j,S-\{j\})\}$$

```
struct pot {
   int iz_vozlisca;
   int v_vozlisce;
   int cena;
   bool mnozica[ST_VOZLISC]; //S
};
```

- Predstavitev množice (S) s poljem (med iz_vozlisca in 1 imamo vozlišča v množici S)

Vozlišče	1	2	3	4	5	6	7	8..
Ali se nahaja znotraj poti	0	0	0	0	0	0	0	0

Implementacija – predstavitev celotnega problema

- Poti na enem nivoju shranjujemo v enojno-povezan seznam poti

```
struct pot {
   int iz_vozlisca;
   int v_vozlisce;
   int cena;
   bool mnozica[ST_VOZLISC]; //S
   pot *naslednji;
};
```

- Nivoje shranjujemo na sklad ali seznam nivojev

Implementacija

graf: matrika, število vozlišč

```
procedure POTNIK(C, N)
 begin
   sklad={} // sklad nivojev
   nivo=VSTAVI_PRVA_VOZLIŠČA(C,N);
   NOV_ELEMENT_SKLADA(sklad, seznam_poti);
   for st_nivoja := 2 to N - 1 do
     begin
       nivo={}
       for vozlišče := 2 to N do
         begin
           NAPRAVI_SEZNAM(C,N, vozlišče, sklad, nivo);
         end;
       NOV_ELEMENT_SKLADA(sklad, nivo);
     end;
   nivo=KONČAJ_SEZNAM(C, N, sklad);
   NOV_ELEMENT_SKLADA(sklad, nivo);
   return sklad;
 end
```

dodaj direktne poti: k → 1 (1. nivo) na prvi nivo

dodaj poti vozlišče → … → 1 v seznam (nivo)

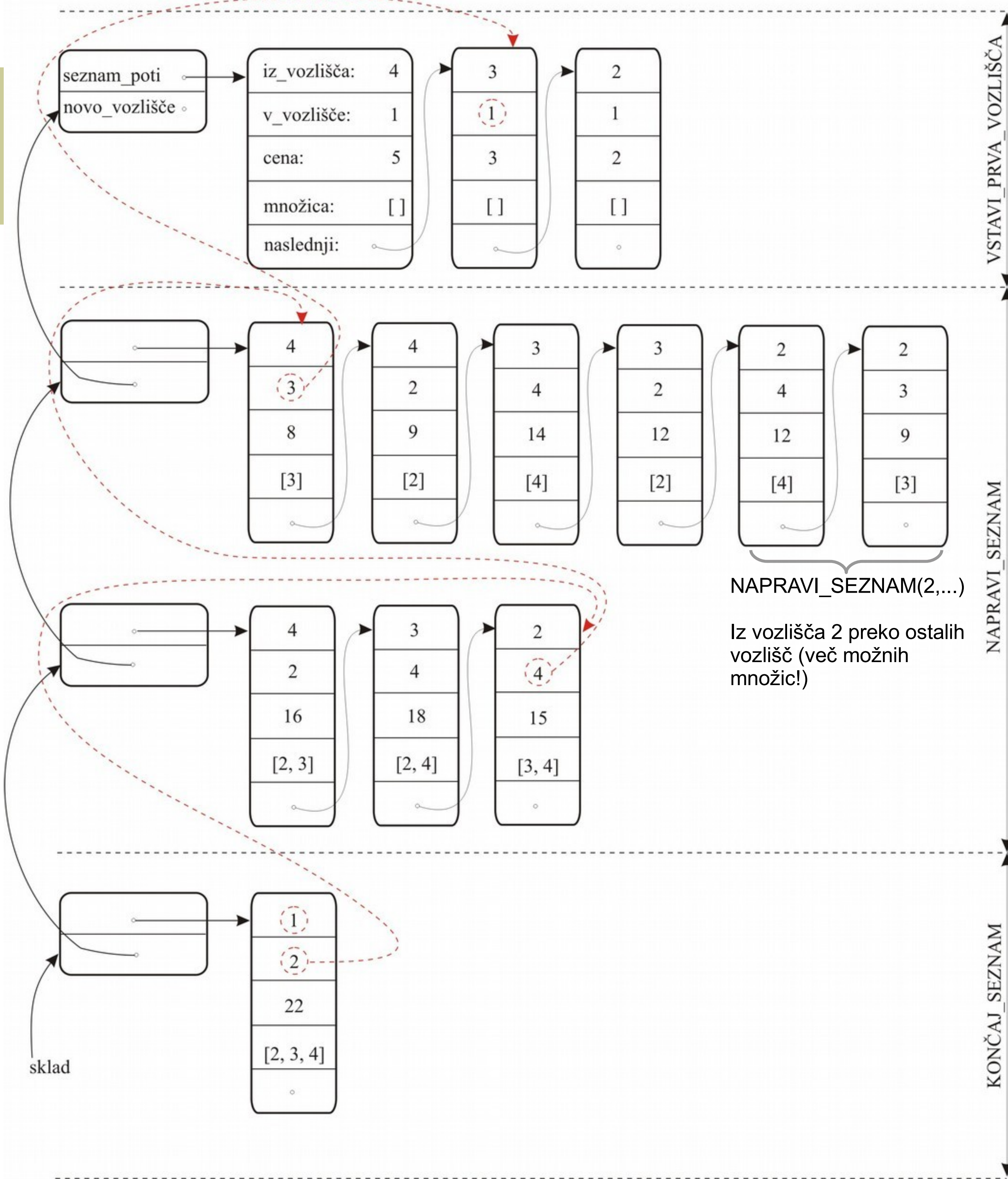
zaključi pot: 1 → k → ... → 1 (zadnji nivo)

1. nivo

2. nivo
NOV_ELEMENT_SKLADA

3. nivo
NOV_ELEMENT_SKLADA

4. nivo



Implementacija

- VSTAVI_PRVA_VOZLIŠČA(C,N)
  - Ustvari seznam poti prvega nivoja
- NOV_ELEMENT_SKLADA (sklad, seznam_poti)
  - Doda seznam_poti na vrh sklada
- NAPRAVI_SEZNAM(C,N, **vozlišče**, sklad, nivo)
  - Algoritem: **vozlišče** dodamo na poti prejšnjega nivoja in dobimo nove poti
    - **Vozlišče** dodamo samo na tiste poti prejšnjega nivoja, kjer ga še ni na poti
  - Izločanje dražjih poti:
    - Če dobimo več poti z enakima iz_vozlišča in množica, ohranimo zgolj najcenejšo pot
- KONČAJ_SEZNAM(C, N, sklad)
  - Na poti prejšnjega nivoja dodamo vozlišče 1 in dobimo nove poti
  - Med dobljenimi potmi vrnemo najcenejšo

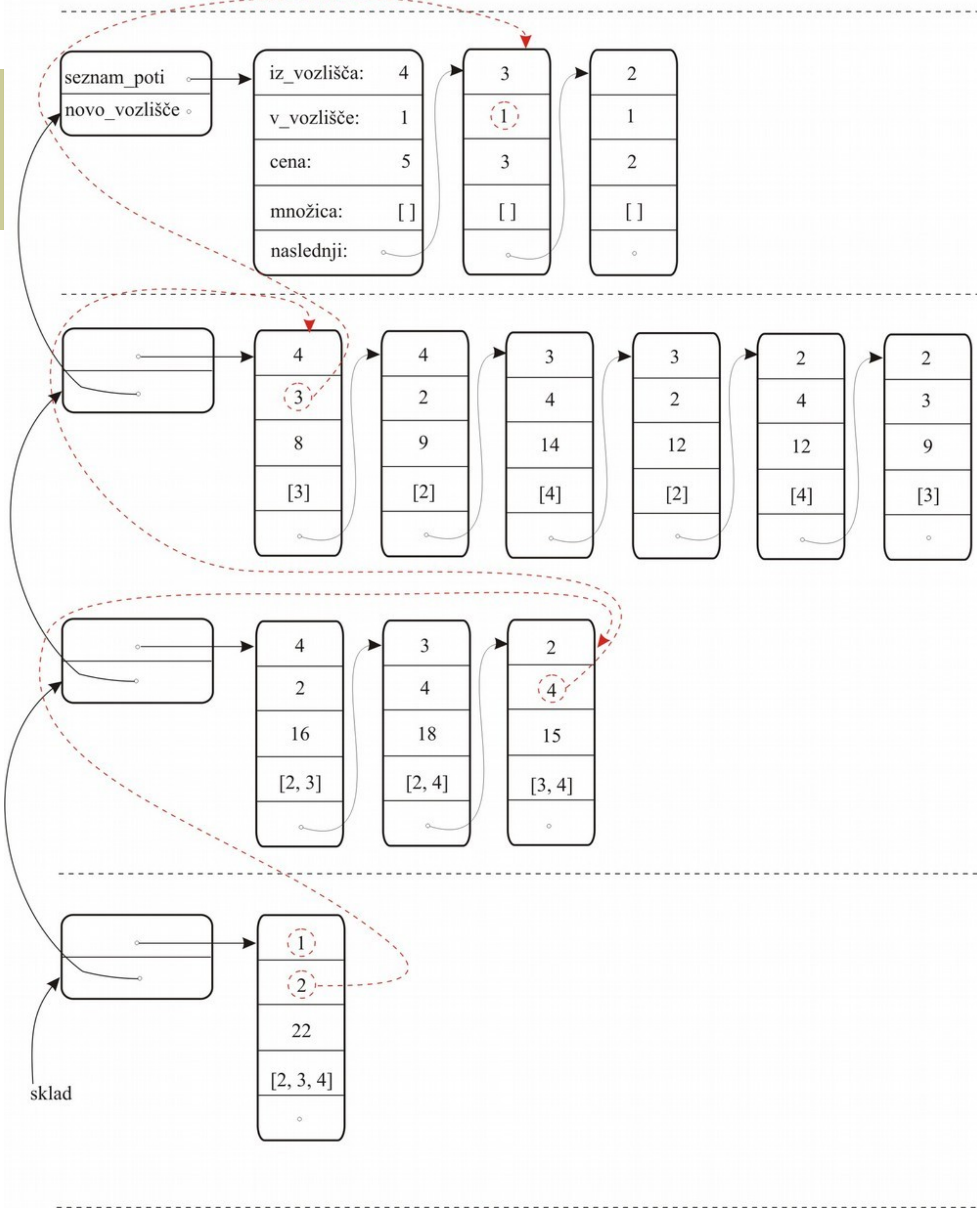
Rekonstrukcija poti

- Bellmanova enačba nam izračuna ceno najkrajše poti (shranjen v poti iz zadnjega nivoja)
- Za rekonstrukcijo poti moramo potovati od zadnjega nivoja do prvega
  - Potrebujemo sklad nivojev (z operacijo POP potujemo do prvega nivoja)
  - Pri potovanju med nivoji poiščemo ustrezno pot, pri tem si pomagamo z informacijo v_vozlišče in množico

Iščemo pot z:
- iz vozlišča 3,
- množica S-3 : {}

Iščemo pot z:
- iz vozlišča 4,
- množica S-4 : {3}

Iščemo pot z:
- iz vozlišča 2,
- množica S-2 : {3,4}



Implementacija

```
procedure REKONSTRUCKIJA_POTI(sklad)
 begin
   trenutni_nivo=POP(sklad)
   pot_od=PRVA_POT(trenutni_nivo)
   print pot_od.iz_vozlisca

   while not SKLAD_PRAZEN(sklad)
     trenutni_nivo=POP(sklad)
     pot_do=NAJDI_PRAVO_POT(pot_od, trenutni_nivo)
       // NAJDI_PRAVO_POT naj vrne pot_do iz trenutni_nivo, kjer je:
       //   pot_od.v_vozlisce=pot_do.iz_vozlisca in
       //   pot_od.S - pot_od.v_vozlisce = pot_do.S
       //

     print pot_do.iz_vozlisca

     pot_od=pot_do
   end
 end
```

Zahteve naloge

- Aplikacija, ki bo poiskala najcenejši obhod v vhodnem grafu z do 12 vozlišči s pomočjo trgovskega potnika z <u>dinamičnim programiranjem (potrebno je delati s prej prikazano podatkovno strukturo, ostale rešitve se ne upoštevajo)</u>
- 1) Potrebno se je striktno držati formata vhodne matrike *graf.txt*
- 2) Privzeto izhodiščno vozlišče: 1
  - Izpišite ceno najkrajše krožne poti in čas trajanja algoritma
- 3) Izpišite nivoje iz sklada nivojev. Pri vsakem nivoju izpišite seznam poti (vse podatke strukture pot)
- 4) Izpišite krožno pot in ceno na podlagi rezultata operacije 2
  - Pazite na vrstni red vozlišč!

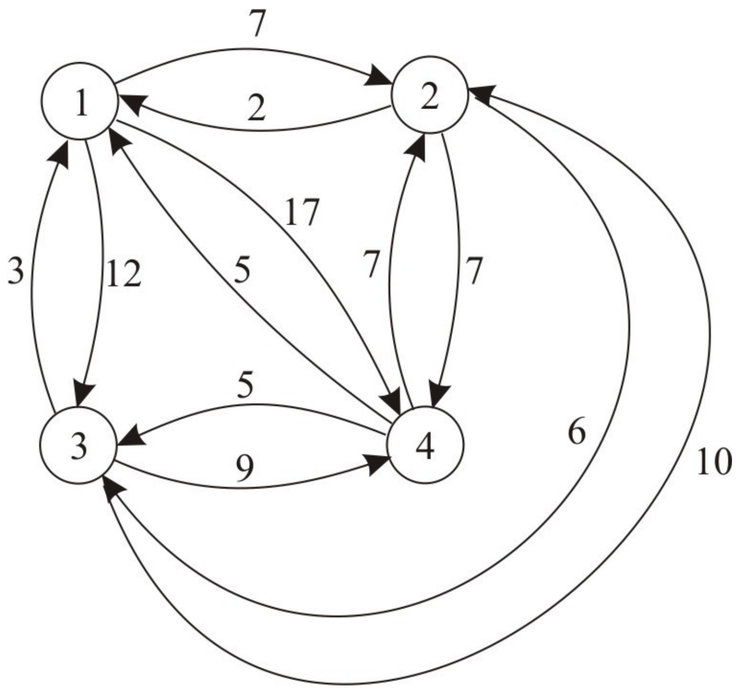
```
Trgovski potnik – izbira:

1) Preberi matriko
2) Reši problem trgovskega potnika
3) Izpiši dobljen seznam nivojev
4) Rekonstrukcija poti
5) Izhod
Izbira:
```

Implementirati je potrebno problem trgovskega potnika z dinamičnim programiranjem. Implementacija mora uporabljati prej definirano podatkovno strukturo. Kot vhod naj služi matrika sosednosti, shranjena v datoteki *graf.txt*, ki jo preberete s priloženo proceduro. Graf ima lahko do vključno 12 vozlišč.
Ob zagonu programa naj se izpiše meni, prikazan na prejšnji strani. Graf se prebere ob izbiri menijske postavke *Preberi matriko*. Ob izbiri menijske postavke *Reši problem trgovskega potnika* naj se zažene procedura *POTNIK* s privzetim izhodiščnim vozliščem 1! Na ekran se naj izpiše cena najkrajše poti tudi čas, ki ga je algoritem potreboval za računanje. Ob izbiri menijske postavke *Izpiši dobljen seznam nivojev* iz seznama nivojev preberite sezname poti in pri vsaki poti izpišite vse parametre strukture pot. Ob izbiri menijske postavke *Rekonstrukcija poti* izpišite najkrajšo krožno pot in ceno te najkrajše poti. Program se zaključi ob izbiri menijske postavke *Konec*.

Vhodna datoteka

```
4
0 7  12 17
2 0  6  7
3 10 0  9
5 7  5  0
```



Za ta testni primer je potrebno dobiti enak sklad nivojev, kot je v primeru!

Vrednost naloge 6 točk:

- Vse do vključno prvega nivoja: 2 točki
- Ostali nivoji: 2 točki
- Rekonstrukcija poti: 2 točki