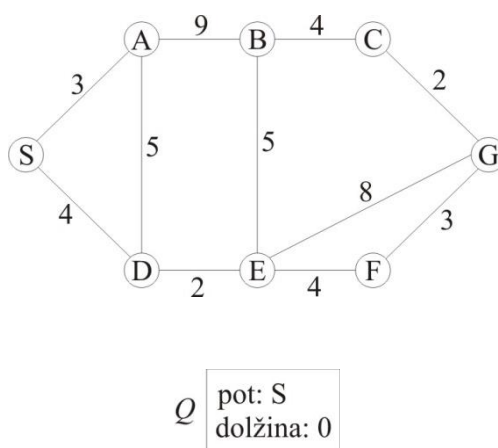


Vaja 6 – Iskanje najkrajše poti med vozlišči s pomočjo strategije razveji in omeji

1. Splošna predstavitev problema

Iskanje najkrajših poti med vozlišči znotraj grafa je pomemben problem znotraj teorije grafov, saj so se nanj navezovali že štiri predhodne vaje. V zadnji vaji se bomo lotili enakega problema, kot ga rešuje tudi Dijkstrov algoritem, to je iskanje najkrajše poti med dvema vozliščema grafa s tem, da bomo sedaj uporabili strategijo razveji in omeji. Strategija razveji in omeji je zelo priljubljena pri iskanjih znotraj grafa. Glavna ideja te metode je, da razvijamo vse potencialne rešitve, dokler ne dobimo prvega približka za njo. Nato dalje razvijamo samo tiste kandidate, ki imajo možnost to rešitev izboljšati. V našem primeru torej to pomeni, da nadaljujemo z razvojem samo tistih poti, ki so lahko teoretično ugodnejše od naše trenutne rešitve. V tistem trenutku, ko ugotovimo, da kandidat ne more izboljšati trenutne rešitve, le tega zavržemo. Iskanje poteka postopoma s pomočjo prioritete vrste Q . Iskanje začnemo tako, da v prioriteto vrsto Q vstavimo izhodiščno vozlišče z začetno ceno nič. Na vsakem koraku nato iz prioritete vrste Q vzamemo pot z najnižjo ceno in zadnjemu vozlišču v poti dodamo nove povezave. Z vsako dodano povezavo dobimo novo pot. Če povezava ne tvori cikla, pot sprejmemo in izračunamo njeno novo ceno. Če se pot konča v cilju, pot shranimo kot trenutno rešitev, ceno te poti pa kot trenutno najnižjo ceno. V nasprotnem primeru novo pot vstavimo v prioriteto vrsto, kjer čaka na nadaljnji razvoj. Pot zavržemo tudi v primeru, če je cena te poti višja od trenutne najnižje cene, saj je jasno, da takšna pot trenutne cene poti ne more izboljšati. Z iskanjem končamo, ko je vrsta Q prazna, ali pa so poti v vrsti vse dražje od trenutno najcenejše poti.

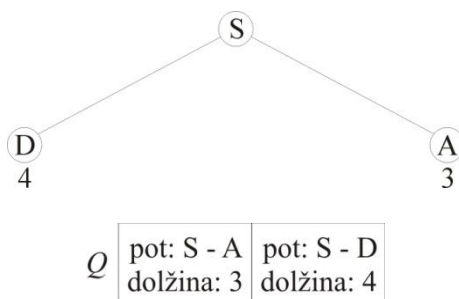
Delovanje strategije razveji in omeji si oglejmo na grafu s slike 1, kjer želimo poiskati najcenejšo pot med vozliščema S in G .



Slika 1: Osnovni graf

Najprej opravimo začetno inicializacijo, tako da v prioriteto vrsto Q vstavimo, pot z zgolj začetnim vozliščem, dolžine 0, ki jo prav tako vidimo na sliki 1.

Iskanje nadaljujemo tako, da iz vrste Q vzamemo to izhodiščno pot in zadnje vozlišče v poti podaljšamo za povezave, ki iz tega vozlišča izhajajo. Ker naša pot vsebuje zgolj vozlišče S , imamo na voljo dve povezavi, to je povezava $S - A$ in povezava $S - D$. Ker nobena od povezav ne tvori cikla z že obstoječimi sprejetimi povezavami, lahko obe poti sprejmemo, pred tem je treba izračunati le njuno ceno. Cena prve poti je tako 3 ($0 + 3$), cena druge pa je 4 ($0 + 4$). Ker nimamo še nobene trenutne rešitve obe poti sprejmemo, saj cene za primerjavo še ni. Dobimo situacijo, ki jo prikazuje slika 2.



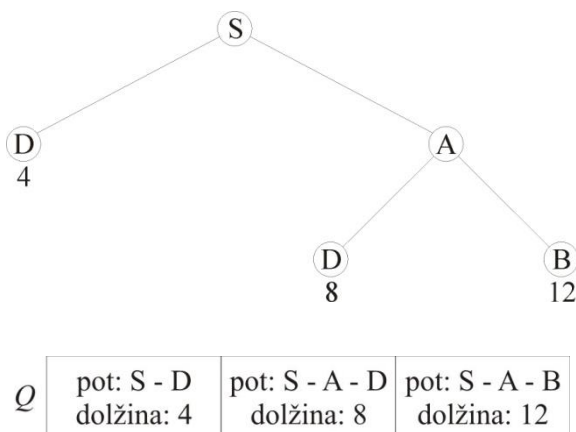
Slika 2: Dobljeno drevo stanj po prvem razvoju poti iz vozlišča S in prioriteta vrsta Q

Ob razvijanju različnih poti lahko le te najbolj nazorno prikažemo s pomočjo drevesa stanj, prikazanega na sliki 2. To drevo služi zgolj lažji predstavitvi in ga znotraj programa ne hranimo.

Iskanje nadaljujemo tako, da iz vrste Q vzamemo pot $S - A$. Pot razširimo tako, da zadnjemu vozlišču v poti, to je vozlišču A dodamo eno od treh povezav $A - S$, $A - B$ in $A - D$. Vsaka od teh treh povezav tvori novo pot in sicer:

- $S - A - S$: pot vsebuje cikel, zato jo zavržemo
- $S - A - B$: pot nima ciklov, zato ji določimo dolžino $3 + 9 = 12$
- $S - A - D$: pot nima ciklov, zato ji določimo dolžino $3 + 5 = 8$

Ker začasne rešitve še nimamo, tudi njene cene ne moremo prekoračiti, zato zadnji poti shranimo v vrsto Q . Tako dobimo:

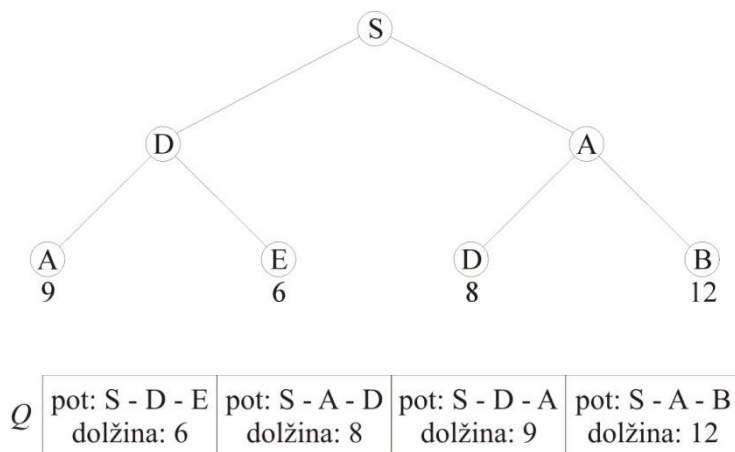


Slika 3: Drevo stanj in prioriteta vrsta Q po razvoju poti $S - A$

Iz vrste Q vzamemo naslednjo pot, to je pot $S - D$. Vozlišče D ima tri povezave, ki tvorijo tri možne poti in sicer:

- $S - D - S$: pot vsebuje cikel, zato jo zavržemo
- $S - D - A$: pot nima ciklov, zato ji določimo dolžino $4 + 5 = 9$
- $S - D - E$: pot nima ciklov, zato ji določimo dolžino $4 + 2 = 6$

Novo dobljeni poti shranimo nato v vrsto Q . Dobljeno situacijo prikazuje slika 4.



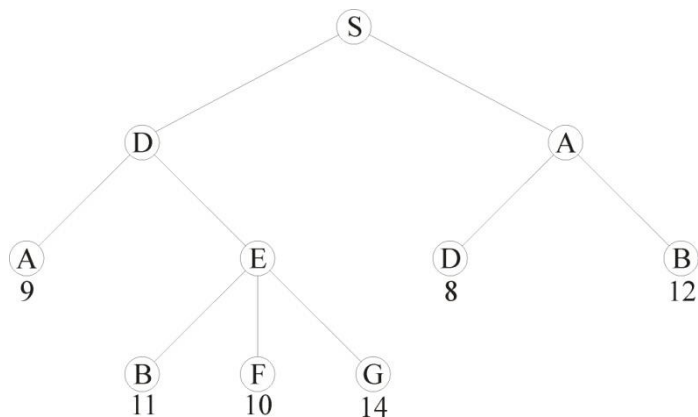
Slika 4: Drevo stanj in prioriteta vrsta Q po razvoju poti $S - D$

Sedaj na začetku prioriteta vrste Q čaka pot $S - D - E$. Zato jo vzamemo iz vrste in razvijemo.

Možne so naslednje poti:

- $S - D - E - D$: pot vsebuje cikel, zato jo zavržemo
- $S - D - E - B$: pot nima ciklov, zato ji določimo dolžino $6 + 5 = 11$
- $S - D - E - G$: pot nima ciklov, zato ji določimo dolžino $6 + 8 = 14$
- $S - D - E - F$: pot nima ciklov, zato ji določimo dolžino $6 + 4 = 10$

Pri razvoju zadnjih treh poti smo razvili tudi pot $S - D - E - G$, ki se zaključi v ciljnem vozlišču G .



Slika 5: Drevo stanj po razvoju poti $S - D - E$

Ta pot postane naša trenutna rešitev, 14 pa trenutna cena najkrajše poti. Sedaj iščemo samo še tiste poti, ki so cenejše od te vrednosti. Trenutne rešitve ne vpišemo v prioriteto vrsto Q, ampak jo hranimo posebej, glej sliko 6.

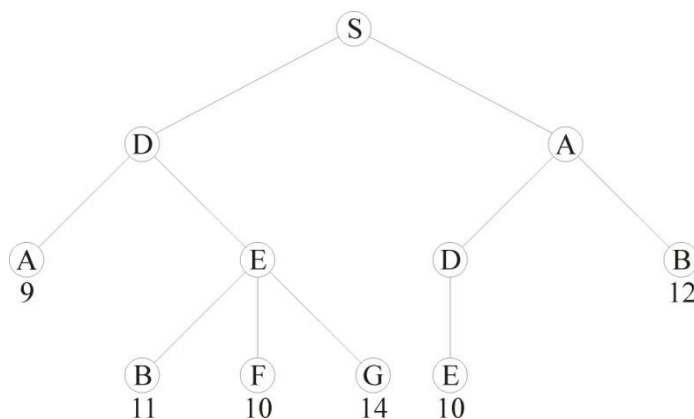
Q	pot: S - A - D dolžina: 8	pot: S - D - A dolžina: 9	pot: S - D - E - F dolžina: 10	pot: S - D - E - B dolžina: 11	pot: S - A - B dolžina: 12
R	pot: S - D - E - G dolžina: 14				

Slika 6: Prioritetna vrsta Q in začasna rešitev R

Ker imamo v prioritetni vrsti še poti, ki so cenejše od trenutne rešitve je še vedno mogoče, da obstaja cenejša rešitev, zato iz vrste q vzamemo pot S - A - D. Zato jo vzamemo iz vrste in razvijemo. Možne so naslednje poti:

- S - A - D - S: pot vsebuje cikel, zato jo zavržemo
- S - A - D - A: pot vsebuje cikel, zato jo zavržemo
- S - A - D - E: pot nima ciklov, zato ji določimo dolžino $8 + 2 = 10$

Od vseh možnih poti lahko sprejmemo samo zadnjo. Drevo stanj se temu ustrezno poveča, poveča pa se tudi prioriteta vrsta Q. Situacija je prikazana na sliki 7.



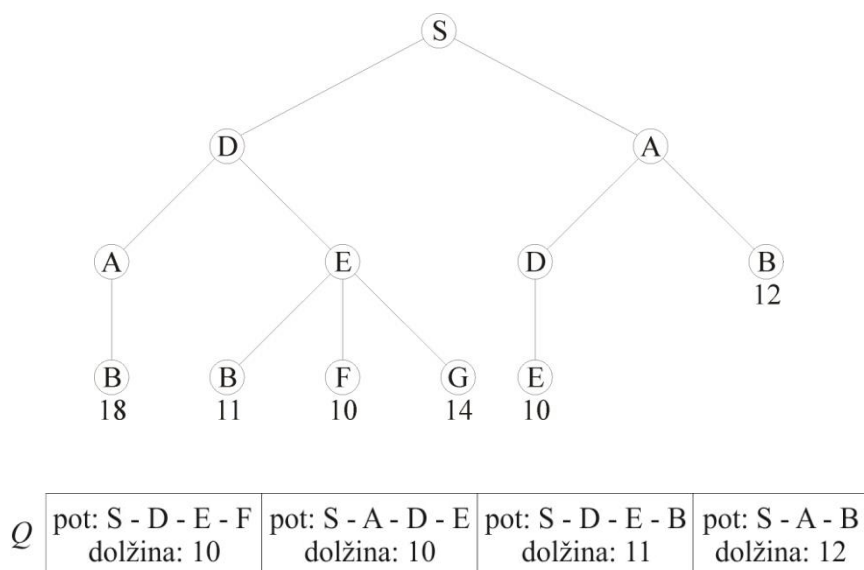
Q	pot: S - D - A dolžina: 9	pot: S - D - E - F dolžina: 10	pot: S - A - D - E dolžina: 10	pot: S - D - E - B dolžina: 11	pot: S - A - B dolžina: 12

Slika 7: Drevo stanj in prioriteta vrsta Q po razvoju poti S - A - D

V vrsti Q imamo še vedno poti, ki so cenejše od trenutne rešitve, zato iz vrste vzamemo naslednjo pot, S - D - A. Pri razvoju te poti imamo naslednje možnosti:

- S - D - A - S: pot vsebuje cikel, zato jo zavržemo
- S - D - A - D: pot vsebuje cikel, zato jo zavržemo
- S - D - A - B: pot nima ciklov, zato ji določimo dolžino $9 + 9 = 18$

Pri razvoju poti $S - D - A$ smo našli samo eno kandidatko, ki pa jo prav tako zavržemo, saj je dražja od trenutne rešitve, tako se spremeni samo drevo stanj, medtem, ko se vrsta Q skrajša, glej sliko 8.

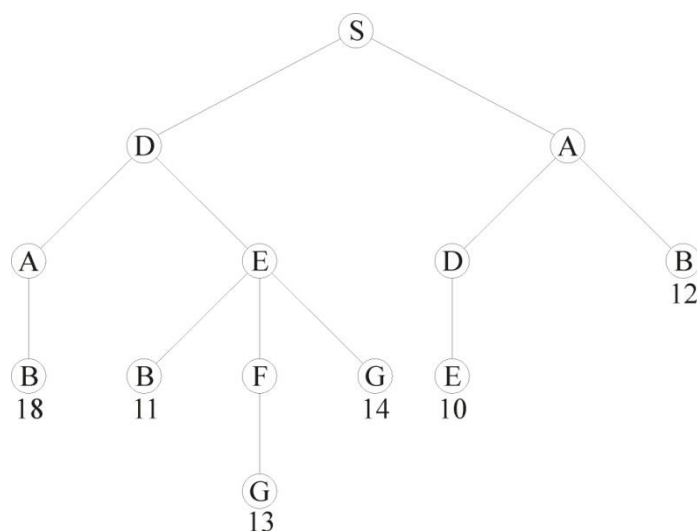


Slika 8: Drevo stanj in prioriteta vrsta Q po razvoju poti $S - D - A$

V vrsti Q imamo še vedno poti, ki so cenejše od trenutne rešitve, zato iz vrste vzamemo prvo izmed njih, to je pot $S - D - E - F$. Zato jo vzamemo iz vrste in razvijemo. pri tem dobimo naslednji poti:

- $S - D - E - F - E$: pot vsebuje cikel, zato jo zavržemo
- $S - D - E - F - G$: pot nima ciklov, zato ji določimo dolžino $10 + 3 = 13$

Kot lahko vidimo, smo našlo novo pot do ciljnega vozlišča, ki je cenejša od trenutne rešitve, zato ta pot postane naša nova začasna rešitev, razširili pa smo tudi drevo stanj, glej sliko 9.



Slika 9: Drevo stanj po razvoju poti $S - D - E - F$

Spremembe v vrstah Q in R so prikazane na sliki 10.

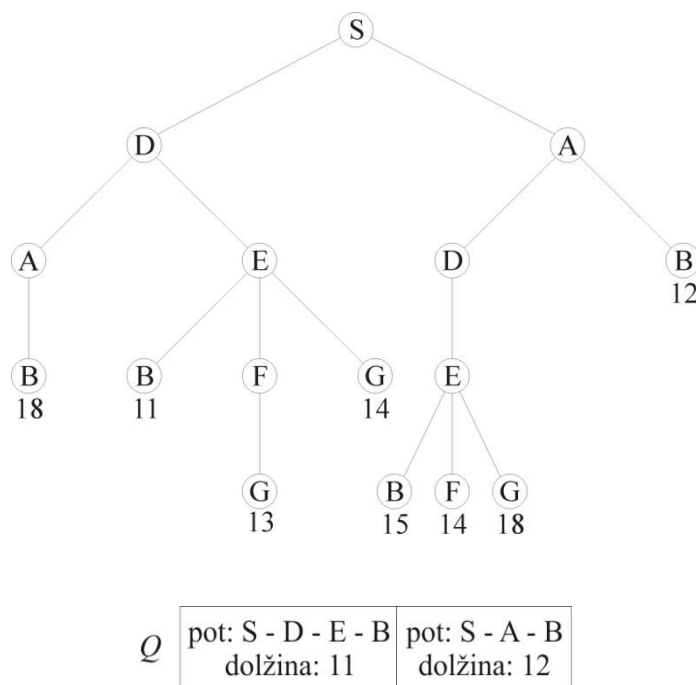
Q	pot: S - A - D - E dolžina: 10	pot: S - D - E - B dolžina: 11	pot: S - A - B dolžina: 12
R	pot: S - D - E - F - G dolžina: 13		

Slika 10: Prioritetna vrsta Q in vrsta rešitve R po razvoju poti S - D - E - F

Z novo rešitvijo razvijamo samo tiste poti, katerih cena je nižja od trinajst, ker pot na čelu vrste Q ta pogoj izpolnjuje, jo vzamemo iz vrste in razširimo. Pri tem dobimo naslednje možne poti:

- S - A - D - E - D: pot vsebuje cikel, zato jo zavržemo
- S - A - D - E - B: pot nima ciklov, zato ji določimo dolžino $10 + 5 = 15$
- S - A - D - E - G: pot nima ciklov, zato ji določimo dolžino $10 + 8 = 18$
- S - A - D - E - F: pot nima ciklov, zato ji določimo dolžino $10 + 4 = 14$

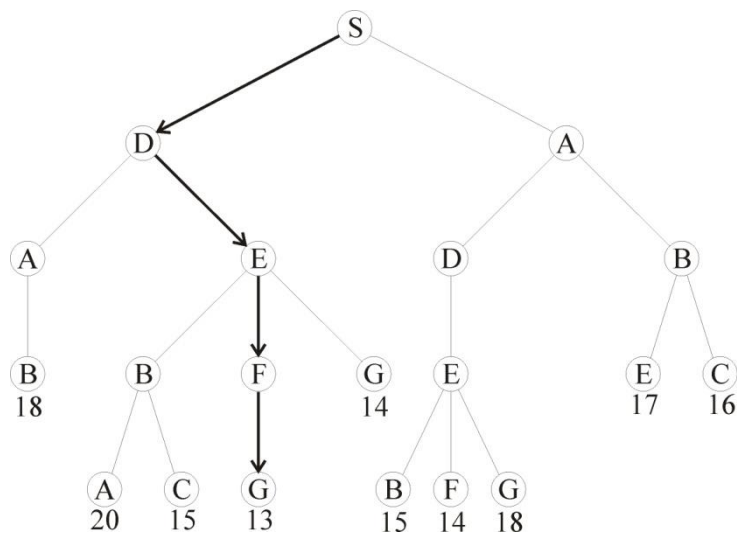
Nobena od dopustnih razvitih poti ni cenejša od trenutne rešitve, zato vse poti zavržemo. Drevo stanj se kljub temu razširi in je prikazano na sliki 11.



Slika 11: Drevo stanj in prioriteta vrsta Q po razvoju poti S - A - D - E

V vrsti Q sta ostali še dve poti S - D - E - B, S - A - B, ki sta obe cenejši od trenutne rešitve, zato najprej razvijemo prvo od obeh, pri čemer so vse nove poti dražje in jih zavržemo. Enako se zgodi pri razvoju zadnje poti v vrsti, to je S - A - B.

Ker je prioriteta vrsta Q pri tem ostala prazna, je zmanjkalo kandidatov za nove poti in z iskanjem zaključimo. Tako naša trenutna rešitev $S - D - E - F - G$ s ceno 13 postane končna. Končno drevo stanj, ki smo ga pri iskanju razvili je prikazano na sliki 12.



Slika 12: Končno drevo stanj s posebej označeno najdeno najkrajšo potjo

2. Pomoč pri implementaciji

Za predstavitev grafa bomo tudi tukaj uporabili matriko sosednosti **C**. Psevdokod samega algoritma je naslednji:

```
function RAZVEJI_IN_OMEJI(G, s, g, R, cena_resitve)  
begin  
  pot.vozlisca := {s}  
  pot.dolzina := 0  
  R := ∅  
  cena_resitve := ∞  
  VSTAVI_V_VRSTO(Q, pot)  
  
  while not (PRAZNA(Q))  
  begin  
    p := VZAMI_IZ_VRSTE(Q)  
    u := zadnje vozlišče v poti p  
    for each vozlišče v ∈ G.Sosed(u) do  
      if not (v ∈ p.vozlisca) then  
        begin  
          p1 := DODAJ_VOZLISCE(p, v)  
          p1.dolzina := p.dolzina + C[u,v]  
          if v = g then  
            begin  
              if p1.dolzina < cena_resitve then  
                VSTAVI_NOVO_RESITEV(R, p1)  
              else if p1.dolzina = cena_resitve then  
                DODAJ_RESITEV(R, p1)  
            end  
          else if p.dolzina < cena_resitve then  
            VSTAVI_V_VRSTO(Q, p1)  
          end  
        end  
      end  
    end  
  end  
end
```

Izpis 1: Psevdokod funkcije RAZVEJI_IN_OMEJI

Algoritem v izpisu 1 ima pet parametrov, tri vhodne in sicer *G* predstavlja podatke o grafu, *s* izhodiščno vozlišče, *g* pa ciljno vozlišče. Zadnja dva parametra sta izhodna parametra. V spremenljivki *R* vrnemo najdeno najkrajšo pot, v spremenljivki *cena_resitve* pa ceno te najkrajše poti. Predstavljeni algoritem RAZVEJI_IN_OMEJI najde vse poti z isto ceno. V ta namen služi funkcija DODAJ_RESITEV, ki k trenutno poti doda novo pot z isto ceno. Funkcija VSTAVI_NOVO_RESITEV počisti vrsto z rešitvijo in vstavi novo pot. Funkcija DODAJ_VOZLISCE zadnjemu vozlišču *v* poti doda novo vozlišče. Pot hranimo v statičnem polju. Če določenih povezav v grafu ni, to označimo z vrednostjo 9999.

3. Zahteve naloge

Za uspešno opravljeno 6. vajo mora študent v programskem jeziku C++ implementirati algoritem za iskanje poti med dvema mestoma s pomočjo strategije razveji in omeji. Po zagonu programa se mora na zaslon izpisati meni, prikazan na sliki 13.

Razveji in omeji - izbira: 1 Preberi podatke iz datoteke 2 Iskanje poti med vozliščema s in g 3 Konec Vaša izbira:
--

Slika 13: Začetni meni, ob zagonu aplikacije

Ob izbiri postavke 1, aplikacija prebere matriko sosednosti s pomočjo rutine priložene na portalu. Ob izbiri menijske postavke 2 mora uporabnik najprej vnesti začetno in končno vozlišče s in g, nato pa program izpiše najkrajšo pot in njeno ceno, oziroma najkrajše poti, če je teh poti več. Ob izbiri menijske postavke 3 se izvajanje programa konča.