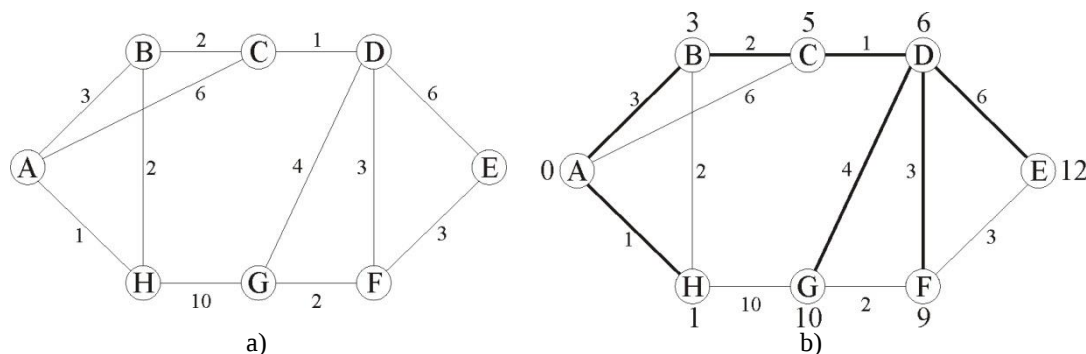


Vaja 4 – Iskanje poti iz enega vozlišča (Dijkstrov algoritem)

1. Splošna predstavitev problema

Problem, ki ga rešujemo v tej vaji je podoben iskanju v širino. Razlika je ta, da imajo povezave med vozlišči pri iskanju v širino povsod enako ceno, v tem primeru pa imajo povezave različno ceno. Se pravi, tudi v tem primeru bomo iskali poti med izbranim vozliščem in vsemi ostalimi, če le-te obstajajo, le da bomo sedaj upoštevali dejansko ceno povezave. Ta tip problema je veliko bližji realnemu svetu, kot iskanje v širino.

Kot primer denimo, da rešujemo problem distribucije iz centralnega skladišča do oddaljenih trgovin. Vozlišča v grafu naj predstavljajo različne trgovine, povezave v grafu pa ceste, ki do teh trgovin vodijo. Pri tem so nekatere ceste ugodnejše od drugih, saj je na njih manj prometa, so krajše, ali pa imajo kakšno drugo lastnost, ki jih loči od ostalih. Da to na nek način predstavimo, uporabljamo v teoriji grafov uteži, ki jih pripišemo posamezni povezavi. Kot rezultat dobimo utežen graf. Naša naloga je, da v takem grafu najdemo tisto pot, ki je najugodnejša, se pravi, da je vsota uteži povezav na njej najmanjša. To pomeni, da iščemo take poti, da bodo stroški distribucije minimalni. Kot primer pogledjmo sliko 1, kjer lahko vidimo začetni graf in minimalne poti, ki smo jih našli.



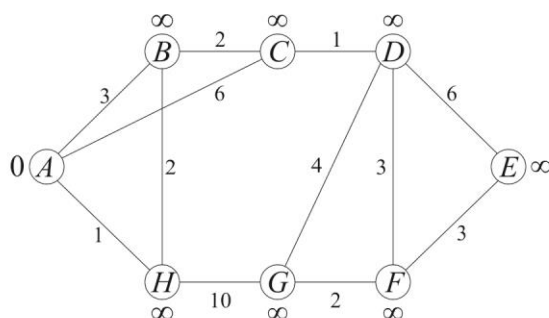
Slika 1: Dijkstrov algoritem a) Izhodiščni graf, b) Rešitev problema

Na sliki 1a vidimo prikaz problema distribucije, prikazan s pomočjo grafa. Centralno skladišče predstavlja vozlišče A, trgovine pa predstavljajo vozlišča B, C, D, E, F, G in H. Ceste med temi stavbami so predstavljene s povezavami, ki so utežene glede na stroške, ki jih imamo z njimi, recimo čas merjen v minutah, ki ga porabimo med vožnjo. Na sliki 1b vidimo rešitev tega problema. Optimalne poti so narisane z odebeljenimi črtami, nad vsakim vozliščem pa imamo zapisano ceno, čas, ki ga porabimo, da prevozimo to pot. Tako denimo za to, da pridemo iz vozlišča A do vozlišča D rabimo minimalno 6 minut in hitreje od tega med obema vozliščema ni možno priti. Optimalna pot vodi preko vozlišč B in C. Vsaka pot je merjena od izhodiščnega vozlišča A dalje, zato imamo pri vozlišču A vrednost 0, saj se v vozlišču A na začetku vsake poti tudi nahajamo. To dosežemo tako da najprej vozlišča opremimo s ceno poti od danega vozlišča do izhodišča, ki ga označujemo s črko s. Nato postavimo to vrednost v

izhodiščnem vozlišču na 0, v vseh ostalih pa na vrednost ∞ . Nato tako opremljena vozlišča postavimo v vrsto, ki je urejena glede na te razdalje. Iskanje rešitve nato poteka tako, da jemljemo vozlišče za vozliščem iz te vrste in za sosede tega vozlišča preverimo, ali lahko pridemo do njih preko trenutnega vozlišča ceneje, kot je trenutno shranjena cena. Če lahko, potem v soseda vpišemo novo, nižjo, ceno, trenutno obravnavano vozlišče pa označimo kot njegovega starša. Ko smo pregledali vse sosede, vrsto ponovno uredimo. Nova vrsta je seveda za eno vozlišče krajša kot prej. Postopek končamo, ko je vrsta prazna.

V nadaljevanju si oglejmo, kako na grafu s slike 1a poiščemo rešitev.

Najprej je potrebno vozlišča v grafu opremiti z začetnimi cenami poti do izhodiščnega vozlišča s , ki je v našem primeru vozlišče A . Dobimo situacijo, ki jo prikazuje slika 2.

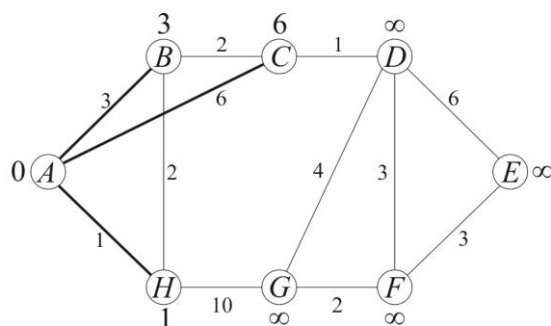


Q	vozl: A	B	C	D	E	F	G	H
	razdalja: 0	∞	∞	∞	∞	∞	∞	∞

Slika 2: Graf z označenimi razdaljami pri vozliščih

Nato je potrebno vozlišča vpisati v urejeno vrsto, Q , ki ji iz tega razloga pravimo tudi prioriteta vrsta, ki je prav tako prikazana na sliki 2. Iz vrste Q sedaj vzamemo prvo vozlišče, to je vozlišče A . Sosedi vozlišča A so vozlišča B , C in H , s katerimi je vozlišče A neposredno povezano. Sledi izračun dolžine poti, do izhodiščnega vozlišča A , ki je enak dolžini poti svojega predhodnika, povečani za ceno povezave med predhodnikom in obravnavanim vozliščem. Ker v tem primeru velja, da je predhodnik vozlišč B , C in H tudi izhodiščno vozlišče, je cena poti do teh vozlišč enaka kar ceni povezave; za vozlišče B torej 3, za vozlišče C , 6 in za vozlišče H , 1. Situacijo, ki jo dobimo, prikazuje slika 3.

Ker vrsta Q ni prazna iz nje vzamemo prvo vozlišče, to je vozlišče H , ter poiščemo njegove sosede, ki sta v tem primeru vozlišči B in G . Izračunamo dolžine poti iz obe vozlišč do vozlišča A , pri čemer dobimo pri vozlišču B vrednost 3 ($1+2$), pri vozlišču G pa vrednost 11 ($1+10$). Ker je v primeru vozlišča G nova vrednost manjša od trenutne, shranimo novo vrednost, v vozlišču G pa označimo, da je njegov predhodnik vozlišče H . Ker pri vozlišču B nova cena ni manjša od trenutno shranjene, v tem primeru ne naredimo nič.

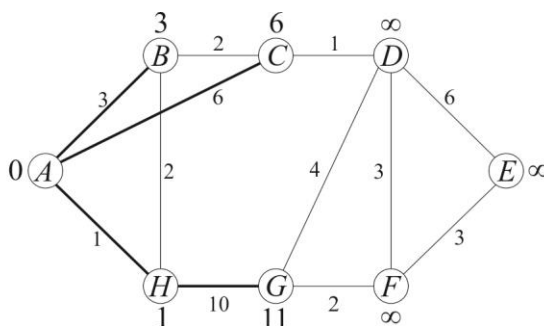


Q

H	B	C	D	E	F	G
1	3	6	∞	∞	∞	∞

Slika 3: Situacija po obravnavi vozlišča A

Situacijo, ki jo dobimo, prikazuje slika 4.



Q

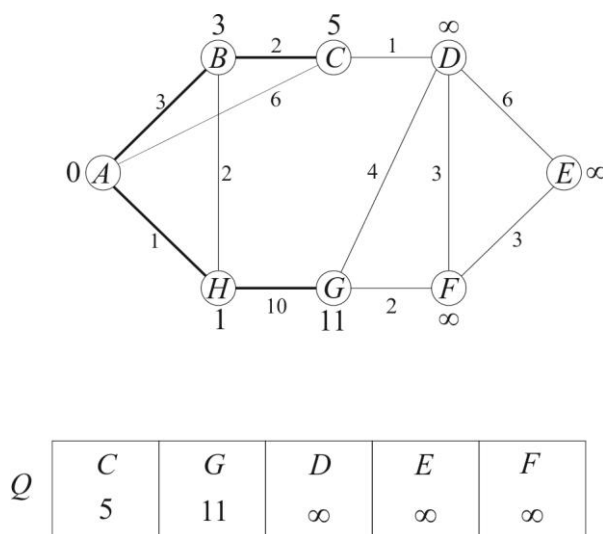
B	C	G	D	E	F
3	6	11	∞	∞	∞

Slika 4: Situacija po obravnavi vozlišča H

Vrsta Q še vedno ni prazna, zato iz nje vzamemo naslednjo vrednost, to je vozlišče B . Vozlišče B ima dva soseda in sicer vozlišči H in C . Izračunajmo najprej ceno poti do vozlišča H , ki znaša 5 ($3+2$), kar je več od cene shranjene v vozlišču, zato ne naredimo ničesar. Cena poti do vozlišča C je enaka 5 ($3+2$), kar je manj od trenutno shranjene, zato novi predhodnik vozlišča C postane vozlišče B , shrani pa se tudi nova cena. Situacijo, ki jo dobimo, prikazuje slika 5.

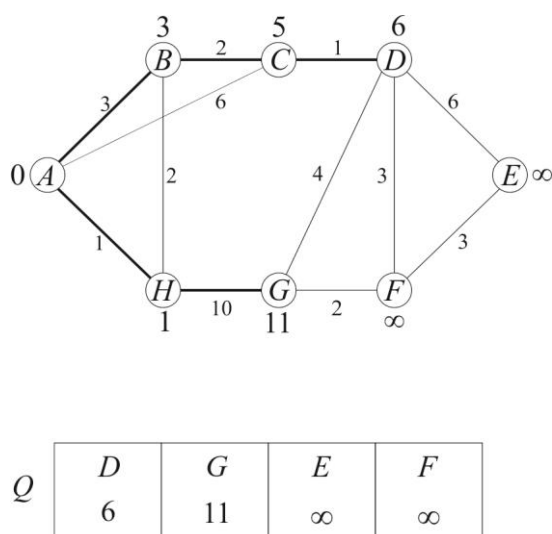
Nadaljujemo z iskanjem in iz vrste Q vzamemo naslednje vozlišče, to je vozlišče C , katerega sosedi so vozlišča A , B in D do katerih je potrebno izračunati ceno poti. Pri tem ni pomembno, da je vozlišče A izhodiščno vozlišče, vozlišče B pa predhodnik obravnavanega vozlišča C . Cena poti do vozlišča A tako znaša 11 ($5+6$), do vozlišča B , 7 ($5+2$) in do vozlišča D je cena enaka 6 ($5+1$). ker je cena v primeru vozlišč A in B višja od tiste, shranjene v vozliščih, ne naredimo ničesar, medtem, ko je cena do

vozlišča D nižja od trenutno shranjene, zato v vozlišče D shranimo novo ceno, vozlišče C pa postane njegov predhodnik.



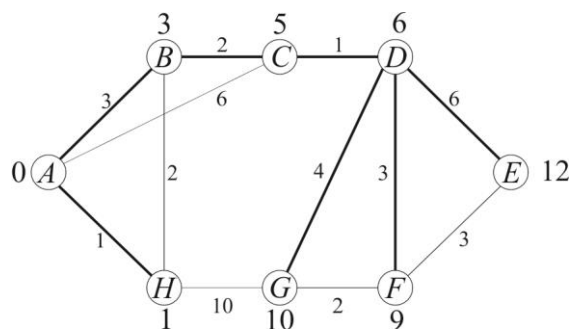
Slika 5: Situacija po obravnavi vozlišča B

Situacija, ki jo dobimo, je prikazana na sliki 6.



Slika 6: Situacija po obravnavi vozlišča C

Vrsta Q še vedno ni prazna, zato vzamemo iz nje naslednje vozlišče, ki je v tem primeru vozlišče D , poiščemo njegove sosedne, ki so vozlišča C , E , F in G , ter izračunamo nove cene poti do njih. Pri vozliščih C ta znaša 7 ($6 + 1$), pri vozlišču E 12 ($6 + 6$), pri vozlišču F 9 ($6 + 3$) in pri vozlišču G 10 ($6 + 4$). V zadnjih treh primerih je nova cena nižja od obstoječe, zato se v vozlišča vpiše nova cena, vsa tri vozlišča pa dobijo za svojega predhodnika vozlišče D . Rezultat opisanih operacij vidimo na sliki 7.



Q

F	G	E
9	10	12

Slika 7: Situacija po obravnavi vozlišča D

Ker vrsta Q še vedno ni prazna, iz nje vzamemo najprej vozlišče F , s sosedi D , G in E , cena poti pa je v vseh treh primerih višja od že shranjene, zato iz vrste Q vzamemo naslednje vozlišče, to je vozlišče G . To ima sosede D , F in H , cena poti pa se tudi tokrat poveča, zato vzamemo iz vrste Q še zadnje vozlišče, to pa je vozlišče E s sosedom F . Tudi v tem primeru se cene poti povečajo, zato ne naredimo ničesar. Ker je vrsta sedaj prazna z iskanjem končamo.

2. Pomoč pri implementaciji

Podobno kot pri prejšnjih vajah imamo tudi tukaj graf predstavljen z matriko sosednosti. Ob tako dani predstavitvi, si pogledjmo psevdokod algoritma:

```

procedure DIJKSTROV_ALGORITEM( $G, s$ )
begin
  for each vozlišče  $v \in G.V$  do
    begin
       $d[v] := \infty$ 
       $predhodnik[v] := NIL$ 
    end
   $d[s] := 0$ 
  VSTAVI_VSA_VOZLIŠČA( $Q, G.V$ )
  while not(PRAZNA( $Q$ )) do
    begin
       $u := IZLOČI\_NAJMANJŠEGA(Q)$ 
      for each vozlišče  $v \in G.Sosedi(u)$  do
        if  $d[v] > d[u] + G.P[(u, v)].utež$  then
          begin
             $d[v] := d[u] + G.P[(u, v)].utež$ 
             $predhodnik[v] := u$ 
          end
        end
      end
    end
  end

```

Izpis 1: Psevdokod procedure DIJKSTROV_ALGORITEM

Psevdokod za izpis poti:

```
procedure IZPIS_POTI(G, s, v)
begin
  if v = s then
    izpiši s + ": dolzina: " + G.d[s]
  else if G.predhodnik[v] = NIL then
    izpiši "med " + s + " in " + v + " ni poti"
  else
    begin
      IZPIS_POTI(G, s, G.predhodnik[v])
      izpiši v + ": dolzina: " + G.d[v]
    end
end
```

Izpis 2: Psevdokod procedure IZPIS_POTI

Za izdelavo prioritete vrste Q bomo uporabili polje števil, ki ga bomo po vsakem koraku uredili s pomočjo algoritma za hitro urejanje.

3. Zahteve naloge

Za uspešno opravljeno 4. vajo mora študent v programskem jeziku C++ implementirati Dijkstrov algoritem za iskanje najkrajše poti iz enega vozlišča v vsa ostala vozlišča v grafu. Po zagonu programa se mora na ekranu izpisati meni na sliki 9.

Dijkstrov algoritem - izbira: 1 Naloži graf 2 Zagon algoritma 3 Izpis najkrajše poti 4 Konec Vaša izbira:

Slika 9: Začetni meni, ob zagonu aplikacije

Uporabnik mora nato vpisati vrednost od 1 do 4, kar sproži ustrezno akcijo. Graf je podan na enak način kot pri drugi vaji. Ob kliku na menijsko postavko 2 mora uporabnik določiti izhodiščno vozlišče, nato pa aplikacija poišče vse poti do ostalih vozlišč. V kolikor uporabnik ta ukaz uporabi še enkrat, se ponovno poiščemo nove poti iz na novo izbranega izhodišča. Pri menijski postavki 3 uporabnik vpiše ciljno vozlišče, nakar dobi izpisano iskano pot, vključno z njeno ceno. Program se zaključi, ko izberemo menijsko postavko 4.