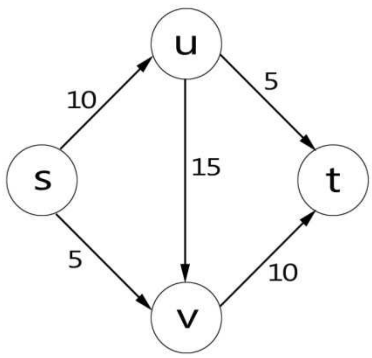
Vaja 5 – Mreže tokov

1. Splošna predstavitev problema

Problem mreže tokov je, podobno kot linearno programiranje, otrok druge svetovne vojne. Njegov tvorec je bil namreč ameriški matematik, Georg B. Dantzig, ki je bil med drugo svetovno vojno vodja oddelka za analize bojnih spopadov pri ameriškem vojnem letalstvu. Tam se je ukvarjal s predhodnikom problemov, ki ga danes poznamo pod imenom mreže tokov. Tega je v celoti izračunal ob začetku hladne vojne ob začetku vzpostavitve Berlinskega zračnega mostu, ko je Sovjetska zveza blokirala del Berlina, ki je bil pod zavezniško kontrolo ter ga odrezala od oskrbovalnih poti po cestah in železnicah, ki so bili pod sovjetsko kontrolo. Zaveznikom so tako za oskrbovanje Berlina ostale na voljo samo zračne poti, saj sovjeti niso verjeli, da bi bilo možno tako veliko mesto oskrbovati na ta način in so želeli zaveznike tako prisiliti, da bi prepustili celoten Berlin sovjetom. Tako je nastal problem, ki ga danes poznamo pod imenom Mreže tokov, ki ga lahko predstavimo z grafom, v katerem letališča predstavljajo vozlišča, letalske povezave med njimi pa povezave v grafu. Število pristajalnih stez in velikost letališča so predstavljala maksimalno kapaciteto prometa, ki se je lahko odvijal med letališči. Ciljno letališče v Berlinu je bilo ponorno vozlišče, eno od letališč v Angliji pa je bilo izvorno vozlišče. Promet z materialom je bilo potrebno po zavezniških letališčih v Evropi porazdeliti tako, da je bil skupni tok dobrin, ki so prispela v zahodni Berlin maksimalen. Pri tem ni bilo pomembno, kakšna je bila pot materiala, niti čas, ki je bil potreben, da je blago prispelo na svoj cilj. Pomembno je bilo le to, koliko materiala je bilo potrebno dnevno naložiti v Angliji in koliko materiala je prispelo dnevno v Berlin in seveda, da je ta količina maksimalna glede na razpoložljivo infrastrukturo.

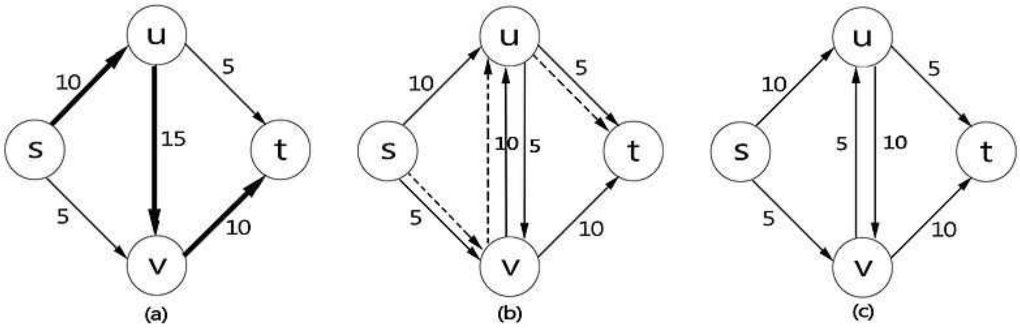
Z rešitvijo tega problema, objavljena je bila šele leta 1951, je zaveznikom uspelo v enem letu izvesti preko 200.000 letov in s tem v mesto pripeljati več materiala, kot ga je bilo pred tem pripeljanega s pomočjo železnice. Sovjeti so videli, da je njihova blokada neučinkovita, zato so jo maja leta 1949 sprostili, Nemčijo pa pozneje razdelili na dva dela in hladna vojna se je začela.

Zelo poenostavljen problem z letališči lahko prikažemo na sliki 1, na grafu s štirimi vozlišči, kjer je letališče *s* izvorno letališče, letališče *t* pa ponorno vozlišče. Vozlišči *u* in *v* pa sta vmesni vozlišči.



Slika 1: Poenostavljen problem mreže letališč

Iskanje optimalne kapacitete začnemo z izbiro ene izmed poti med vozliščema *s* in *t*. Denimo, da je to pot *s*–*u*–*v*–*t*. Po tej poti lahko prenesemo 10 enot materiala, saj je 10 ozko grlo na tej poti. S prometom po tej poti smo zapolnili kapacitete povezav *su* in *st*, medtem, ko lahko po povezavi *vu* prenesemo še pet enot materiala. Ker so kapacitete povezav *su* in *st* že zasedene, pri iskanju alternativnih poti ne pridejo več v poštev. S tako spremenjeno situacijo se zopet lotimo iskanja poti po grafu in edina pot, ki nam je ostala je pot *s–v–u–t*. Ozko grlo po tej poti je 5. Ker je to tudi kapaciteta povezav *sv*, *vu* in *ut* tudi te poti pri iskanju alternativne poti niso več uporabne in jih iz nadaljnjega iskanja izločimo. Ker smo s tem izkoristili že vso razpoložljivo kapaciteto, smo z iskanjem končali in naš maksimalni tok skozi mrežo je enak 15. Postopek iskanja je prikazan na sliki 2.



Slika 2: Iskanje maksimalnega toka v mreži

Če opisani postopek formalno zapišemo, dobimo Ford–Fulkersonov algoritem, ki se imenuje po svojih avtorjih Lester R. Fordu in Delbert R. Fulkersonu, ki sta ga predstavila leta 1955.

Najprej je potrebno podati definicijo mreže tokov. Mreža tokov je usmerjen graf $G(V, E)$, v katerem ima vsaka povezava (u, v), sami jo bomo kratko označili z *uv*, nenegativno kapaciteto, ki jo označimo s $c(u, v)$. V mreži tokov lahko identificiramo dve posebni vozlišči izvor *s* in ponor *t*. Tok mreže označujemo s $f(u,v)$ in je tok, ki se pošilja preko povezave *uv*, pri čemer za vse povezave velja, da je $f(u, v) \le c(u, v)$. Ker se tok v vozliščih ne sme nabirati, mora veljati naslednje:

$$f(u,v) = -f(v,u) \text{ in}$$

$$\sum_{v \in V} f(u,v) = 0.$$

Definirajmo sedaj še mrežo preostankov kot graf $G_f(V, E_f)$, ki je mreža s kapaciteto $c_f = c(u,v) - f(u,v)$ in je brez toka. Pri tem se lahko zgodi, da je v mreži preostankov dovoljen tok iz vozlišča *u* v vozlišče *v*, čeprav v originalni mreži ta tok ni bil dovoljen. Do tega pride, ko je $f(u,v) > 0$ in $c(v,u) = 0$. Takrat velja, da je $c_f(v,u) = c(v,u) - f(v,u) = f(u,v)$.

Reševanje maksimalnega toka poteka s pomočjo iskanja poti povečanja. Pot povečanja, *p*, je preprosta pot med vozlišči s in t v mreži preostankov G_f. Kapaciteto poti povečanja označujemo s $c_f(p)$ in je enaka minimalni kapaciteti povezav $c_f(u,v)$ v poti *p*.

Za vsako povezavo v poti *p* je sedaj potrebno izračunati novi povečan tok, to storimo z uporabo naslednjih dveh enačb:

$$f(u,v) = f(u,v) + c_f(p) \text{ in}$$

$$f(v,u) = f(v,u) - c_f(p).$$

Jasno je, da se graf G_f zaradi teh operacij spremeni, mi pa nadaljujemo z iskanjem maksimalnega toka z iskanjem nove poti povečanja p. Iskanje se konča, ko ni več moč najti poti med vozliščema s in t v novo nastalem grafu G_f.

2. Pomoč pri implementaciji

Ford–Fulkersonov algoritem lahko zapišemo v obliki psevdokoda, prikazanega v izpisu 1.

```
function EDMONDS_KARP(G, s, t)
 begin
   maxFlow := 0;
   for vsako povezavo uv ∈ E  do
     begin
       f[u][v] := 0;
       f[v][u] := 0;
     end

   while BFS(s, t) do
     begin
      cf(p):= min(cf(u,v), za vse povezave v poti p);

      for vse povezave uv v poti p do
        begin
          f[u][v]:= f[u][v] + cf(p);
          f[v][u]:= f[v][u] - cf(p);
        end
        maxFlow := maxFlow + cf(p);
     end

  return maxFlow;
end
```

Izpis 1: Iskanje maksimalnega pretoka

Prva naloga Ford–Fulkersonovega algoritma je iskanje poti povečanja, za kar uporabljamo algoritem iskanja v širino, ki je bil predmet obravnave pri predmetu Algoritmi in podatkovne strukture, zato ga tukaj ne bomo posebej obravnavali. Ford–Fulkersonov algoritem, kjer iskanje poti izvedemo z iskanjem v širino, imenujemo tudi Edmonds-Karpov algoritem.

Osnovni algoritem za iskanje v širino je potrebno vseeno nekoliko modificirati, namreč pri iskanju sinov trenutnega vozlišča u je potrebno dodatno preverjati, ali ima povezava med vozliščema še kaj proste kapacitete. Tako je pogoj za izbiro sinov pri iskanju poti povečanja naslednji:

```
function BFS(s, t)
 begin
                                ...
  if color[v]=WHITE and c[u][v]-f[u][v] > 0 then
                                ...
 end
```

Izpis 2: Ogrodje funkcije za iskanje v širino

Algoritem za iskanje v širino vedno najde eno samo obstoječo pot iz vozlišča s v vozlišče t, v kolikor pa takšna pot ne obstaja, nam funkcija `BFS` vrne vrednost `FALSE`. Pot se shranjuje tako, da vsako vozlišče v poti pozna svojega predhodnika. Pot pregledamo tako, da se postavimo v vozlišče *t* in se pomikamo nazaj po poti, dokler ne dosežemo vozlišča *s*.

3. Zahteve naloge

Izdelati je potrebno aplikacijo, ki bo poiskala maksimalni tok skozi poljubno mrežo. Mreža je podana s tekstovno datoteko, kjer je na začetku podano število vozlišč in število povezav v grafu, nato so naštete povezave s trojico *u v c*, kjer sta *u* in *v* krajiščni vozlišči, *c* pa je kapaciteta povezave. Primer vhodnega grafa je prikazan v izpisu 3.

```
6 9
0 1 16
0 2 13
1 2 10
1 3 12
2 4 14
3 2 9
3 5 20
4 3 7
4 5 4
```

Izpis 3: Primer datoteke z vhodno mrežo

Kot izhod bo program vrnil vrednost tokov preko posameznih povezav, skupaj z njihovimi kapacitetami in maksimalno vrednostjo toka. Primer izhoda je prikazan v izpisu 4.

```
(0, 1) [12/16]
(0, 2) [11/13]
(1, 2) [0/10]
(1, 3) [12/12]
(2, 4) [11/14]
(3, 2) [0/9]
(3, 5) [19/20]
(4, 3) [7/7]
(4, 5) [4/4]

maks. tok: 23
```

Izpis 4: Pričakovani primer izhoda

Privzeto ime datoteke z vhodnim grafom naj bo *graf.txt*, aplikacija pa naj preko dialoga za nalaganje datotek omogoča nalaganje tudi ostalih datotek z mrežami. Vozlišča so lahko tudi podana s tekstovnimi imeni. Določanje izvornega in ponornega vozlišča naj bo prepuščeno uporabniku.

Za lažjo testiranje je še potrebno implementirati grafičen izris grafa (tako vhodnega problema kot tudi rešitve). Za pomoč lahko uporabite poljubno knjižnico, kot je na primer: http://www.graphviz.org/ .