Dinitzov algoritem

Projektna naloga pri predmetu Osnove programiranja v diskretni matematiki

Jimmy Zakeršnik

3. 1. 2024

Sledeče definicije in opis algoritma so povzete po [1].

Definicija 1:

1. Naj bo $G = (V, E)$ usmerjen povezan graf brez vzporednih povezav in brez povezav tipa $vv; v \in V$ ter $s, t \in V$ poljubni različni vozlišči na njem. Vozlišču s pravimo *vir*, vozlišču t pa *ponor*. Dodatno, naj bo $c : E \to \mathbb{R}^+$ funkcija, ki vsaki povezavi priredi nenegativno realno število. Tej funkciji pravimo *kapaciteta*, peterici $N = (V, E, s, t, c)$ pa *omrežje*.

2. Funkciji $f : E \to \mathbb{R}^+$ pravimo *pretok*, če zadošča naslednjima pogojema:
   - $\forall e \in E : 0 \leq f(e) \leq c(e)$
   - $\forall v \in V \setminus \{s, t\}$:
   $$\sum_{\substack{u \in V \\ uv \in E}} f(uv) = \sum_{\substack{u \in V \\ vu \in E}} f(vu)$$

3. Količino $F = \sum_{\substack{u \in V \\ ut \in E}} f(ut) - \sum_{\substack{u \in V \\ tu \in E}} f(tu)$ imenujemo *skupni pretok* f v omrežju N.

Mnogo problemov iz prakse (npr. sestavljanje prometnih omrežjih) lahko prevedemo na iskanje maksimalnega pretoka skozi neko omrežje. Eden izmed algoritmov, kako poiščemo maksimalni pretok, bo obravnavan v tej nalogi - Dinitzov Algoritem. Tukaj bo algoritem opisan, v priloženi datoteki *Dinitz.py*, pa bo tudi implementiran.
Dinitzov algoritem v prvi fazi iz danega omrežja $N = (V, E, s, t, c)$ in nekega začetnega pretoka f sestavi t. i. »rezidualno omrežje« $N' = (V, E', s, t, c')$, kjer je $E' = E \cup \{vu; uv \in E \land f(uv) > 0\}$in $c'(e) = \begin{cases} c(e) - f(e) & ; e \in E \\ f(e) & ; e \in E' \setminus E \end{cases}$ t. i. »rezidualna kapaciteta«. V drugi fazi algoritem s pomočjo BFS algoritma sestavi t. i. razslojeno omrežje $N'' = (V'', E'', s, t, c'')$ na naslednji način:

1. Poženi BFS algoritem z vrhom v s. Če algoritem ne doseže ponorja t, je dan pretok f že maksimalen.

2. Če BFS algoritem doseže t: $V'' = \bigcup_{i=0}^{l} V_i$; V_i so sloji, ki jih določi BFS algoritem v s in $V_l = \{t\}$.

3. $E''_i = \{uv \in E'; u \in V_i \land v \in V_{i+1}\}$

4. $E'' = \bigcup_{i=0}^{l} E''_i$

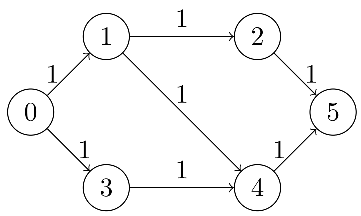
5. c'' je skrčitev c' na E''

V tretji fazi Dinitzov algoritem iz f sestavi maksimalen pretok f'' v N'', ki ga tudi imenujemo zaporni pretok. To naredi tako, da v N'' poišče neko (s, t)-pot po povezavah (s pomočjo DFS algoritma), ki še nimajo popolnoma zapolnjene kapacitete, in pretok f'' na tej poti poveča, kolikor lahko. To algoritem ponavlja, dokler lahko. Če pri iskanju poti algoritem zaide v »slepo ulico«, se vrne do vozlišča na katerem je bil, preden je zašel v njo in slepo ulico blokira, da vanjo ne zaide še enkrat.

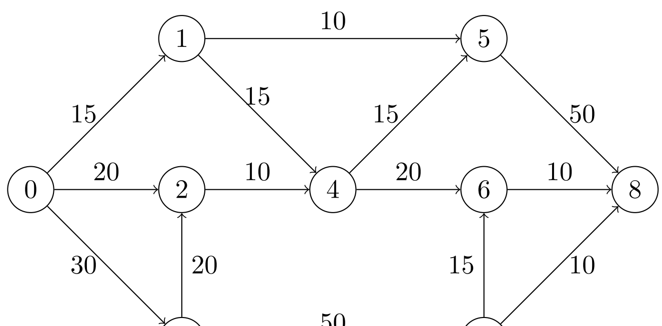
Na koncu se algoritem vrne v N in pretok f »poveča«:

$$f(e) = \begin{cases} f(uv) + f''(uv) & ;\ uv \in E \\ f(uv) - f''(vu) & ;\ vu \in E'' \setminus E \end{cases}$$

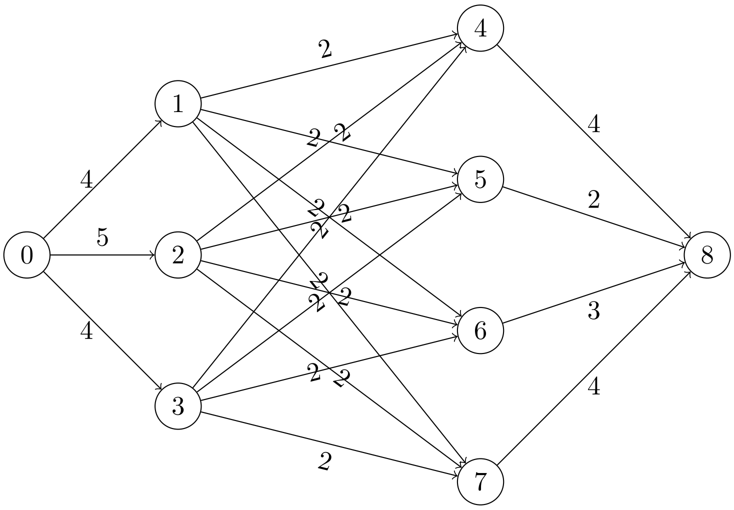
S tako popravljenim f se algoritem vrne v prvo fazo. Ta cikel se ponavlja, dokler ne dobi pretoka f, ki je maksimalen.
Za dano omrežje $N = (V, E, s, t, c)$ ima zgoraj opisani Dinitzov algoritem časovno zahtevnost $O(|V|^2|E|)$. To časovno zahtevnost ima tudi implementiran algoritem DinitzAlg v *Dinitz.py*, torej $O(|V|^2|E|)$. Da je to res je treba preveriti samo, da je časovna zahtevnost vsakega koraka implementiranega algoritma enaka $O(|V||E|)$. Vidimo, da za sestavo stratificiranega omrežja N'' iz N potrebujemo $O(|E|)$ in ker so vse ostale operacije v algoritmu, razen klic funkcije *maksimalenpretok*, tudi opravljene v $O(|E|)$, bo časovna zahtevnost omenjene funkcije edino, kar bi lahko vplivalo na časovno zahtevnost koraka celega algoritma. Funkcija *maksimalenpretok* pa ima časovno zahtevnost $O(|V||E|)$. Sledi torej, da ima ena izvedba zanke v funkciji *DinitzAlg* časovno zahtevnost $O(|V||E|)$. Preostanek razmisleka sledi na enak način, kot je podan v viru [1].
V *Dinitz.py* se koda izvede na treh testnih primerih: *primer.txt*, *primer2.txt* in *primer3.txt*. Omenjena omrežja so narisana spodaj, skupaj z navedbo maksimalnega pretoka, kot ga izračuna *DinitzAlg*.



Slika 1: *Omrežje iz primer.txt z maksimalnim pretokom* 2.



Slika 2: *Omrežje iz primer2.txt z maksimalnim pretokom* 45.



Slika 3: *Omrežje iz primer3.txt z maksimalnim pretokom* 13.

Literatura

[1] S. Even in G. Even, *Graph algorithms, 2nd edition*, Cambridge University Press, 2012, str. 85–102.

[2] *Dinic's algorithm*, v: Wikipedia, the free encyclopedia, [ogled 3. 1. 2024], dostopno na `https://en.wikipedia.org/wiki/Dinic%27s_algorithm`.