1) Vylepšení naivní verze F-F alg.:

Kde to nefunguje:

Vylepšení —> hledáme vždy jednosměrné nejkratší

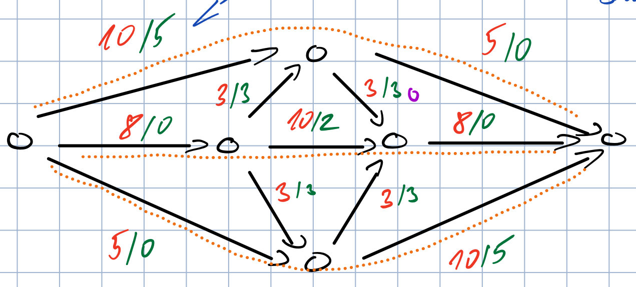
? Dokažte že nemusí najít maximální ?

Kapacity

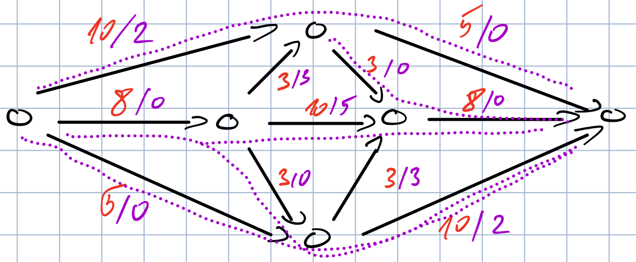
Kapacity při nejkratším hledání

odtud se nelze dostat do stoku.

Nejkratší cesty, co zablokují další růst.



Tento protipříklad je důkazem, že i zde takový alg. selže, protože max. tok by našel jako 16, zatímco FF by našel 23.



Pozn.: Stejná situace nastane u každého grafu, který vyžaduje poslání části toku po delší cestě do větve s vyšší kapacitou (která se ale zúží dále od zdroje).

=> Tudíž bych na její plnou kapacitu nedosáhl při primárním naplňování nejkratšími cestami.

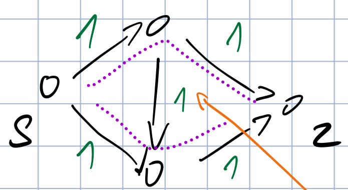
2) Jednosměrný s orákulem

A) Existuje vůbec nějaká posloupnost zlepšujících cest po směru, co najde maximální tok?

B) Stačí nám takové orákulum vždy?

C) Platilo by to pro orákulum napovídající pouze nejkratší cesty?

A) Ano, například v síti: bychom pomocí orákula max. tok našli.



Maximální tok

– Orákulum nám jen musí poradit, ať nevybereme tuto vertikální hranu.

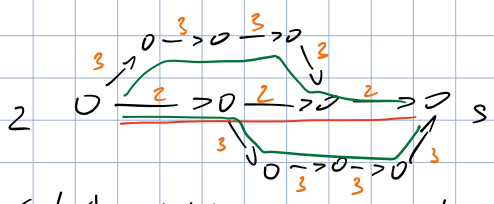
– Obecně lze tvrdit, že maximální tok je součet série zlepšujících cest. Tudíž platí, že pokud nám orákulum pošeptá tu správnou (takovou, že nezablokuje to, co nemá) cestu, najdeme nakonec maximální tok.

b) Takovéto orákulum obecně stačí vždy. Jelikož FF pro nalezení max. toku bere pro hledání cesty rezervní hrany, aby započítal i cesty „proti" hraně, tak to za nás převezme nyní orákulum. FF algo. si pomocí rezervy hrany zajistí, že nezáleží na pořadí, ve kterém cesty najdeme (jakmile bych narazil do saturovaného místa, tak mohu jít „proti proudu" a jen to zaznamenám změnou rezervy hrany), abychom dosáhli maximálního toku. Pomocí orákula vím ale zajištěno, že budu dostávat správné pořadí.

c) Pro „krátké" orákulum platí stále stejný problém co byl zmíněn výše, tedy že existují nejkratší cesty, které však zároveň při saturaci zablokují jedinečné cesty jiné.

Mějme pro příklad síť:

kapacity



$f =$ 4 vs. 2

Zde bych si pomocí kterékoli (jediné) nejkratší cesty zablokoval ostatní cesty a již bych nemohl výsledný tok zvýšit.

3) Dinic s omezenými celými čísly

Ostatní metody alg. běží v $O(n)$ čase.

Chceme ukázat, že hledání blokujícího toku bude s-m složité pro nějakou konstantu c, tedy $O(m)$ místo $O(mn)$ v originálním algoritmu.

Zde je předpis pro hledání blokujícího toku:

Blokující tok:

1. $g = 0$
2. Dokud $\exists P$ cesta (z, s):
3. $\quad \varepsilon = \min_{e \in P} (c(e) - g(e))$
4. $\quad \forall e \in P: g(e) \mathrel{+}= \varepsilon$
5. $\quad$ Odstraním $g(e) = c(e)$: zablokování
   $\quad$ snížíme ε.
6. Očistím síť ⊛

Pro jednotlivé kapacity platilo, že každé nalezení cesty vedlo k blokaci celé cesty.

Nyní tvrdím, že **buď zablokuji konkrétní hranu na nějaké blokující cestě, tak ji maximálně nejvýše c-krát** pro malou konstantu c.

To proto, že kapacity jsou celočíselné a omezené, takže ε bude vždy alespoň 1. V nejhorším případě bude vždy právě 1 a já budu muset $g(e)$ **zvednout právě c-krát.**

Amortizovaně můžeme nahlížet na situaci následovně: Každou hranu smažeme nejvíce c-krát, jelikož pak musí být určitě zablokována. Při každém průchodu cestou vždy všechny hrany na cestě alespoň o 1 přiblížím blokaci. Zároveň očišťování smaže každou hranu/vrchol nejvíce jednou a pokud si do mazané fronty budu přidávat vrcholy už při blokování cesty, tak celkové složitost hledání, jelikož celkem přispěje jen m hrany. Celková složitost je pak $O(m)$.

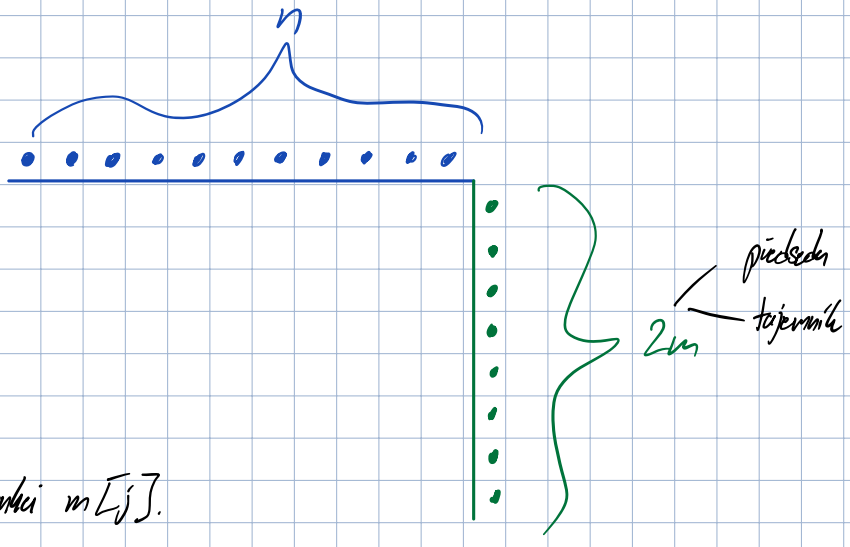
4) Parlamentní lobby:

n poslanců

m lobbů

Opět hledám párování v grafu

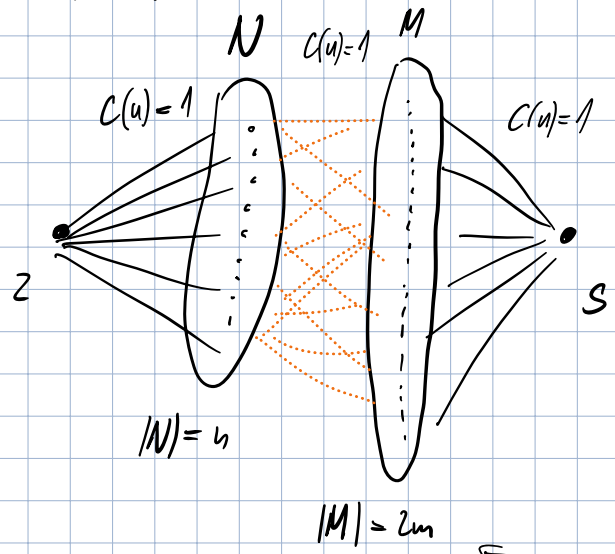
↳ hrana reprezentuje situaci, že poslanec $n[i]$ zastává funkci $m[j]$.



Pozorování: Lze hledat opět v bipartitním grafu, jen místo m vrcholů v partitě lobby bude $2m$. Platí taktéž, že v rámci jedné partity nejsou žádné vrcholy spojené.

Uvažby jedna pozice měl hrany do dvou různých partity N, znamenalo by to, že sedí na dvou křeslech. To však v max. párování nenastane. Stejně tak dva pozice do jednoho křesla sednout nemohou.

Sestrojení grafu pro nalezení největšího párování:



Pro každý klub dvě pozice.

Tzn. že všechny hrany mají kapacitu $c(u)=1$.
Pak stačí pustit alg. na nalezení největšího toku. Já si volím implementaci FF.

Takový algoritmus vždy možné uskutečnit.
Řešení nalezne tehdy, pokud velikost maximálního toku je rovna počtu $2m$.

Pokud $|f| < 2m$, některá pozice není obsazena.

↳ Tedy že vede hrana do pozice v klubu a z ní do studia.

Složitost algoritmu bude $O(m \cdot n)$, jelikož taková je složitost pro FF alg. s jednotkovou kapacitou. To, že ve skutečnosti je to $n \cdot 2m$ se schová do asymptotického O.