1) Vylepšení naivní verze F-F alg.:

Kde to nefunguje:

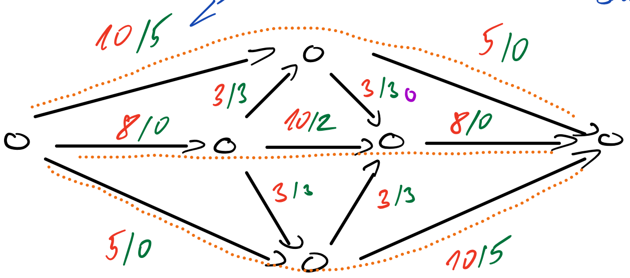
Vylepšení —> hledáme vždy jednosměrné nejhutší

? Dokážte že nemusí najít maximální ?

Kapacity

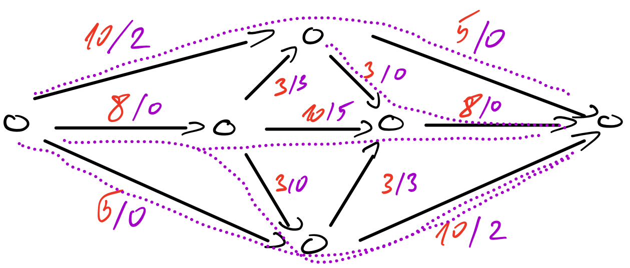
Kapacity po nejhutším hledání

odtud se nelze dostat do stoku.



Nejhutší 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 přelití části toku po delší cestě do větve s vyšší kapacitou (která se ale užší dále od zdroje).

=> Tudíž bych ani její plnou kapacitu nedosáhl při přímočarém naplňování nejhutšími cestami.

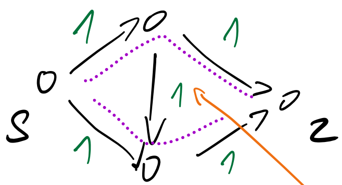
2) Jednostranný s orákulem

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 nejhutší cesty?

A) Ano, například v síti:



Maximální tok

bychom pomocí orákula max. tok našli.

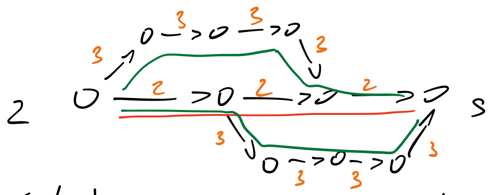
– Orákulum nám jen musí poradit, ať nevybereme tuto vertikální hranu.

– Obecně lze tvrdit, že maximální tok je součet série zlepšujících cest. Tudíž platí, že pokud nám orákulum pošeptá ty správné (takové, že nezablokuje to, co nemá) cesty, najdeme nakonec maximální tok.

b) Takový algoritmus obecně stačí vždy. Jelikož FF pro nalezení max. toku bere pro hledání cesty rezervu hrany, aby započítal i cesty „proti" hraně, tak to za nás převezme nyní algoritmus. FF alg. si pomocí rezervy hrany zajistí, že nezáleží na pořadí, ve kterém cesty nasytíme (jakmile bych nasytil do saturovaného místa, tak mohu jít „proti proudu" a jen to zaznamenám změnou rezervy hrany), abychom dosáhli maximálního toku. Pomocí algoritmu nám ale zajistíme, že budu dostávat správné pořadí.

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

Mějme pro příklad síť:



kapacity

$f = \boxed{4 \text{ vs. } 2}$

Zde bych si pomocí kteréhokoli (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 $c \cdot m$ 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. $\varepsilon = \min_{e \in P} \left(c(e) - g(e) \right)$
4. $\forall e \in P$: $g(e) \mathrel{+}= \varepsilon$
5. Kdykoliv $g(e) = c(e)$: smažeme e. zablokování
6. Dočistím síť ⊛

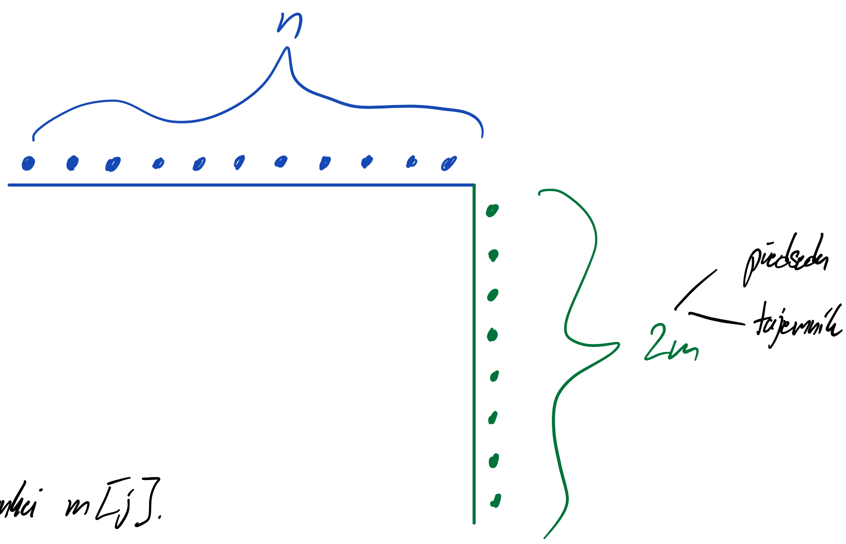
Pro jednotkové kapacity platilo, že každé nalezení cesty vedlo k blokaci celé cesty. Nyní tvrdím, že než zablokuji konkrétní hranu na nějaké blokující cestě, tak ji navštívím nejvýše c-krát pro nějakou konstantu c. To proto, že kapacity jsou celočíselné a omezené, tudíž ε bude vždy alespoň 1. V nejhorším případě bude vždy právě 1 a já budu muset $g(e)$ zvedat právě c-krát.

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

4) Parlamentní kluby:

n poslanců

m klubů



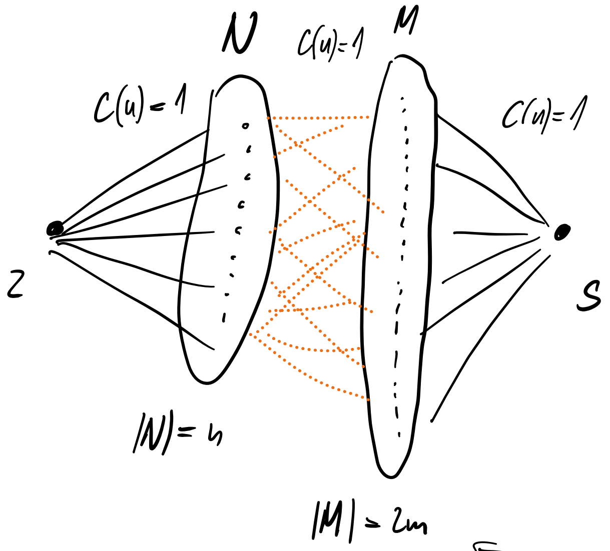
Opět hledám párování v grafu

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

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

Kdyby jeden poslanec měl hranu do dvou vrcholů partity m, znamenalo by to, že sedí na dvou křeslech. To ale v max. párování nenastane. Stejně tak dva poslanci do jednoho křesla sednout nemohou.

<u>Sestrojení grafu pro nalezení největšího párování:</u>



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 například FF.

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

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

$\Leftrightarrow$ Tedy že vede hrana do pozice v klubu a z ní do stoku.

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.