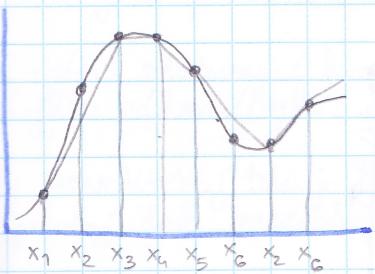


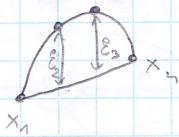
Příklad: Approximace funkce $f(x)$ - máme body x_i a x_j a chtěme vybrat několik množství bodů a položit je přímkami

Úvěření
21.3.2019



Pokud spojíme body x_i a x_j , počítáme chybou:

$$c_{ij} = \sum_{k=i+1}^{j-1} \epsilon_k^2$$



Konec pro chybu upravíme, abychom penalizovali model, když ponecháme všechny vrcholy:

$$c_{ij} = \alpha + \beta \sum_{k=i+1}^{j-1} \epsilon_k^2 \quad \begin{cases} \text{pro } \alpha \gg \beta & \text{vybereme co nejméně hrany, ideálně jedinou přímku} \\ \text{pro } \beta \gg \alpha & \text{vybereme všechny hrany.} \end{cases}$$

Když máme c_{ij} , můžeme ulohu řešit pomocí nejkratších cest.

Důležité: Máme body na obvodu ČR a chceme některé odstranit. Chyba ϵ_x se počítá jako vzdálenost bodu od přímky. Na implementaci.

Příklad: Kamarádi spolu žili na nýlet, každý nás plátil. Jak je vyrovnat?

$P = \{ \text{Adam, Bedřich, Cecília, David} \}$

$c = \{ 0, 590, 110, 300 \} \leftarrow \text{kolik kdo plátil}$

x_{ij} : Kolik zaplatil člověk i člověku j ? $x_{ij} \in \mathbb{N}$

$$c_i + \sum_{j=1}^n x_{ij} - \sum_{j=1}^n x_{ji} = s, \quad s = \frac{\sum c_i}{4} = 250$$

\rightarrow kolik ostatní poslou mne } r. pohledu člověka i
 → kolik pořlu jai ostatním }
 → kolik jsem ubratil na nýleti }

Jak ho udělat tak, aby byly co nejméně transakce?

$$p_{ij} \in \{0, 1\} \quad \begin{cases} 0: \text{neproběhla platba } i \rightarrow j \\ 1: \text{proběhla platba } i \rightarrow j \end{cases}$$

$$\min \sum_{i=1}^n \sum_{j=1}^n p_{ij}$$

$$\text{s.t.} \quad \begin{aligned} x_{ij} &\geq p_{ij} && \text{pokud transakce neproběhne, bude } p_{ij} \text{ ukládáno na 0.} \\ x_{ij} &\leq p_{ij} \cdot c_{\max} && \text{pokud proběhne, musí se } p_{ij} \text{ uvevnout, a to jde jen na 1.} \\ c_{\max} &= 590 \end{aligned}$$

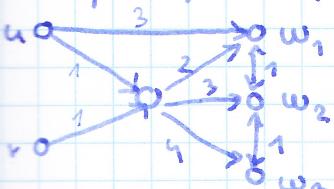
Příští test: algoritmus na nejkratší cesty, když

Přednáška
26.3.2019

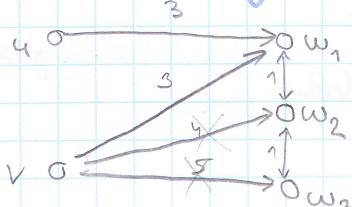
Contraction hierarchies (CH)

- mapa (graf) má několik vrátnic (ulice, ulice, hlavní ulice, dálnice)
- při cestě z vesnice do vesnice, dojdeme na nějakou dopravní lypu, po ní dojdeme daleko a pak k ní sjedeme až dojdeme do cíle

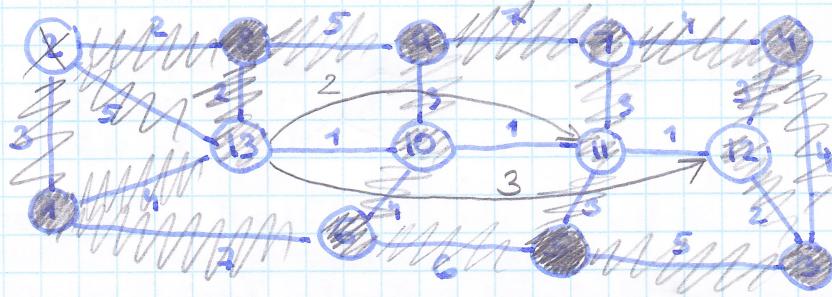
Fáz 1: precomputation, výpočet vzdálostí ahož



\Rightarrow



G_0 je výchozí graf
 G_i je graf, který málo a G_{i-1} , odebíráme vrchol u_i a měkceji už za jeho hrany.



Vrcholy:

1 - můžu odstranit, protože ke každé cestě přes 1 existuje alternativa za lepší cenu

2 - 11

3 - 11

4 - 11

5 - 11

6 - 11

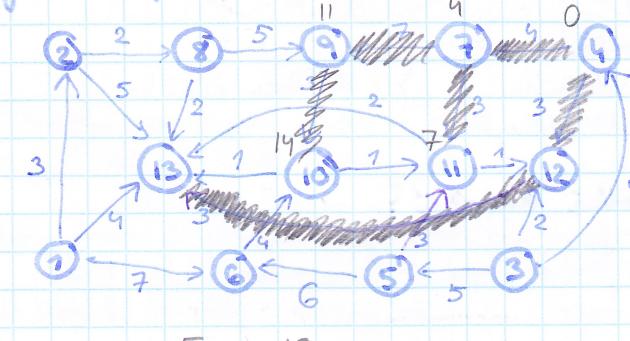
10: nedostanu se $13 \rightarrow 11$, nejdřív kroužek 2

11 - 11 - 12 \rightarrow 13 ravedu kroužek 3

Petr říká, že sem napsal, že tento algoritmus nemusíme umět.

Následující graf má hrany stejně jako G_0 , ale k tomu navíc má hrany $13 \leftrightarrow 11$ o ceně 2, a $12 \leftrightarrow 13$ o ceně 3.

Fáze 2: prohledávání: z 1 do 4, hrany uděláme z méně hodnotných vrcholů do výšších



z vrcholu 4 se dostanu do $\{4, 7, 9, 10, 11, 12, 13\}$

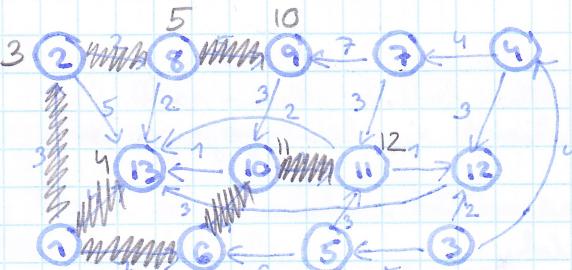
z vrcholu 1 se dostanu do $\{1, 2, 6, 8, 9, 10, 11, 13\}$

vrchol	cena z 1	cena z 4	celkem
9	10	11	21
10	11	14	25
11	12	7	19
13	4	6	10

Nejrychlejší cesta vede přes 13:

$1 - 13 - 10 - 11 - 12 - 4$, což je v pravidlu

graf: $1 - 13 - 10 - 11 - 12 - 4$



funguje to stejně jako bidirectional Dijkstra

hodně se liší na číslování vrcholů, ale to je jiná úloha

Taky v grafech (v záti)

- maximální hory

- max. cardinality v bipartitních grafech

- multisegmentní hory

Síť je pátečice (G, l, u, s, t)

- u, l je maximální a minimální průtok hrany
- s, t jsou počáteční a cílový vrchol

Tok v síti: ohodnocení hrany $f(e)$, takže $\sum_{e \in \delta^+(v)} f(e) = \sum_{e \in \delta^-(v)} f(e)$, tj. platí Kirchhoffův zákon. Neplatí pro s a t .

Def: proveditelný tok je takový tok, kde $f(e) \in \langle l(e), u(e) \rangle$. Může se dát, že přípustný tok neexistuje.

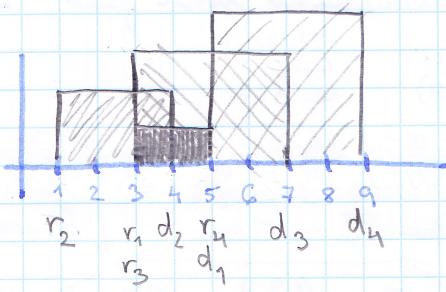
Jedna: Máme zadanou síť, chceme najít takový tok $f(e)$, abychom maximizovali bilanci ($\sum_{e \in \delta^+(s)} f(e) - \sum_{e \in \delta^-(s)} f(e)$).

Např: chceme poslat co nejvíce vagónů po řeleznicí, nebo co nejvíce kopii do rozhovodu, nebo nejvíce aut stát město.

Zpracování jednoho toku v síti:

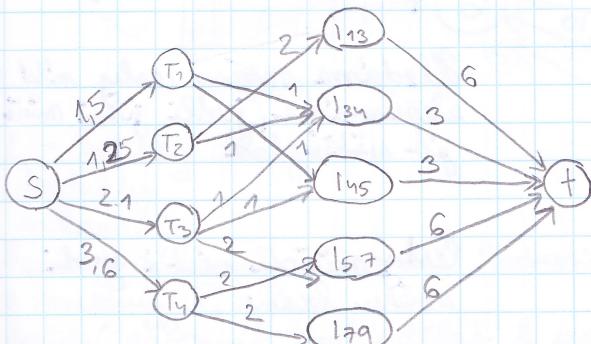
- máme 3 identické procesory
- úloha lze řešit a parallelně v mí, ale v jednu chvíli ji můžete dělat jen jeden procesor (nebo parallelizovat).

task	T_1	T_2	T_3	T_4
processing time	1.5	1.25	2.1	3.6
release date	3	1	3	5
deadline	5	4	7	9



kolik tasků můžu v daném intervalu počítat?

0	1	1	3	2	2	2	1	1	0
13	34	45	57	79					



- hrana $s \rightarrow T_i$ má takový maximální průtok, jaký je processing time úlohy T_i
- hrana $s \rightarrow T_i$ do T_i má takový maximální průtok, kolik dluhohlede v daném intervalu slíhnout
- hrana $s \rightarrow +$ do $+$ má takový maximální průtok, jaké jsou vypočtené možnosti v T_i .

Úloha lze řešit dvěma způsoby:

- 1) nastavíme minimální průtok do všech hrani na 0, pak hledáme maximální tok $f(e)$. Tento tok je řešením, pokud splňuje hrany $s \rightarrow T_i$
- 2) nastavíme minimální průtok v hraniach $s \rightarrow T_i$ na $l(e) = u(e)$, pak hledáme, jestli existuje přípustný tok.

Cvičení
28.3.2019

Naučníkách předmětu jsou nové odkazy

- jak správně svolit big M do ILP

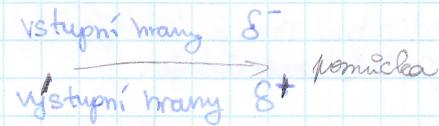
- formulace Rubikovy kostky na co nejméně tahů pomocí ILP

Toky: Tok je (G, s, t, l, u) , G je orientovaný graf

$$l, u: E(G) \rightarrow \mathbb{R}^+$$

$$\text{tok: } f(e): E(G) \rightarrow \mathbb{R}^+$$

$$\forall v \in G, v \neq s, t: \sum_{e \in S^+(v)} f(e) = \sum_{e \in S^-(v)} f(e)$$



$$\text{připustný: } \forall e: f(e) \leq u(e), f(e) \geq l(e)$$

$$\text{maximální tok: } \max \sum_{e \in S^+(s)} f(e) - \sum_{e \in S^-(s)} f(e)$$

Hledání maximálního toku jako LP

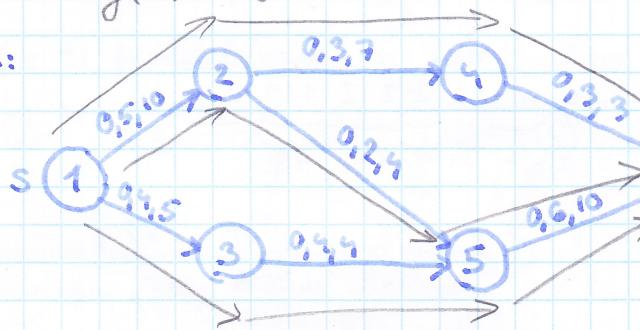
$$\max \sum_{e \in S^+(s)} f(e) - \sum_{e \in S^-(s)} f(e)$$

$$\text{s.t.: } \forall e \in E \quad l(e) \leq f(e) \leq u(e)$$

$$\forall v \in V - \{s, t\} \quad \sum_{e \in S^+(v)} f(e) = \sum_{e \in S^-(v)} f(e)$$

$$f(e) \in \mathbb{R}^+$$

Př:



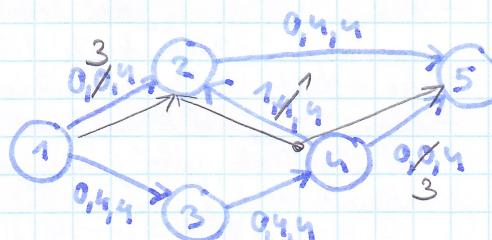
Není to ILP, ale LP, takže
to můžeme řešit polynomálně

Máme zadanou síť a následující
připustný tok, který chceme algoritmem

$$(l(e), f(e), u(e))$$

Hledáme možné cesty alespoň a
zkontrolujeme, jestli jimi může
jít větší tok.

Př:



Cesty můžou větši proti
směru toku

Jakkoliv na cestách 125, 1345 a 13425 můžeme nic přidat, pojďme
na cestě obrácenou hranu. 1245

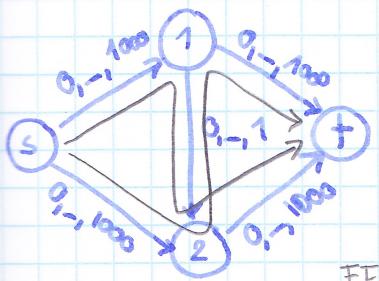
$$s \xrightarrow{+S} +$$

$$+ \xrightarrow{-S} s$$

U cest po směru hran zkontrolujeme, kolik můžeme přidat do
toku. U obrácených hran zkontrolujeme, kolik můžeme odebrat.

$$\text{Kapacita cesty} = \min \left[\begin{array}{l} \{u(e) - f(e) \mid e \text{ je dopředná hraha}\} \cup \\ \{f(e) - l(e) \mid e \text{ je hraha zadní}\} \end{array} \right]$$

Tahle se nazývá Ford-Fulkerson alg., hledá cesty a zkoumá, jestli je lepší.



Ford-Fulkerson selře, protože má náročnost $O(|E| \cdot \max(u))$ (kdeždou cestu, nejlepší)

Variace FF algoritmu pro hledání nejkratší cesty, kdeždou slouží jen $O(V^2 \cdot |E|)$. Ríká se tomu Edmonds-Karp algoritmus.

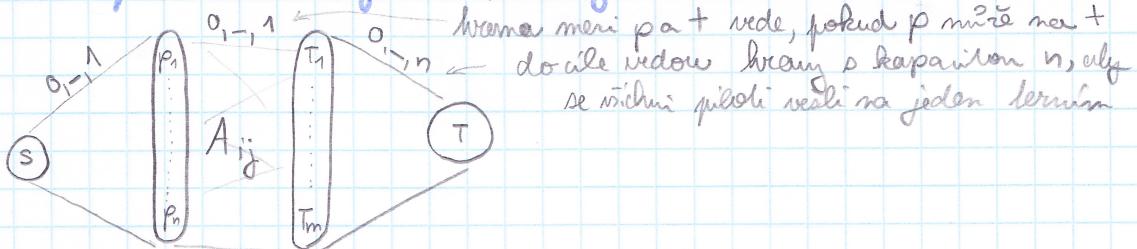
FF algoritmus má náročnost pseudopolynomialní, protože $\max(u)$ je konstanta, ale závisí na rozsahu.

Příklad:

$P = \{P_1, \dots, P_n\}$ sestava pilotů, kterí mají přijít na skoušku

$T = \{T_1, \dots, T_m\}$ sestava terminálních skoušek

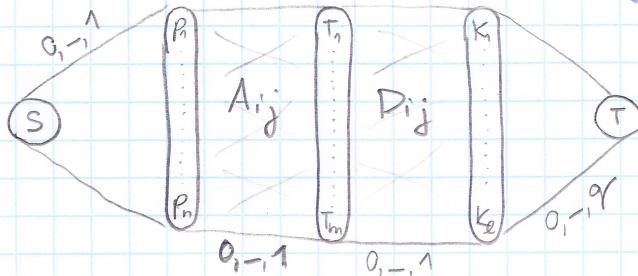
$A_{ij} = \begin{cases} 1 & \text{pilot } i \text{ může přijít na termín } j \\ 0 & \text{pilot } i \text{ nemůže jít na termín } j \end{cases}$



hledáme meni p a t vede, pokud p může na t
do alespoň dvou lince s kapacitou n, aby
se všechni piloti vydali na jeden termin

přidáme komisaře. U každého skoušeného pilota bude vždy jeden pilot.

$D_{ij} = \begin{cases} 1 & \text{komisař může na termín } j \\ 0 & \text{komisař nemůže na termín } j \end{cases}$ Komisař nemůže oddělat více
než q skoušek.



přidáme hrany D_{ij} , kdeždou ukazuje, jestli může komisař k na termín t, stejně jako A_{ij} pro piloty.

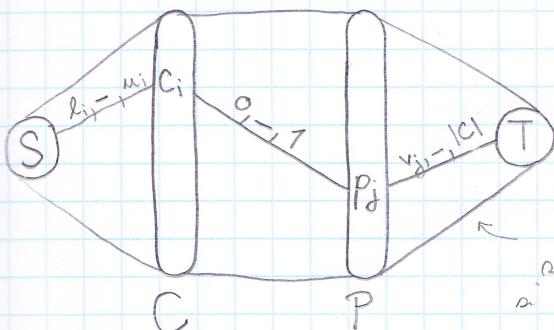
Hrany z k do t mají horní omezení q, aby každý komisař
dělal jen q skoušek.

HW 2 - Survey design

P: produkty

C: zákazníci

- každý zákazník C; může hodnotit jen obor, který si koupil, tj. složku z množiny $P_i \in SP$
- každý zákazník smí udělat k produktu p_j pouze jednu recenci.
- počet recenzí od jednoho zákazníka lze v intervalu (l_i, u_i)
- každý produkt p_j dostane alespoň vj recenci



Hledáme maximální tok, tj. chceme
mít co nejméně vyplňující recenze

K produktu p_j dle může alespoň vj recenci, a jako horní mezi můžu svolit alkdy počet zákazníků, nebo ještě lepe počet zákazníků, kteří si produkt p_j koupili.

$e = (c_i, p_j) \in E \iff p_j \in P_i \Rightarrow$ zákazník c_i může mapat recenci k p_j , jen pokud si produkt p_j koupil