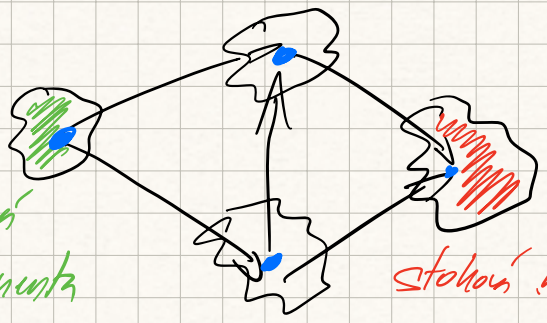
Silná souvislost: Existuje cesta mezi všemi vrcholy v orientovaném grafu

Graf komponent $C(G)$:

- vrcholy jsou komponenty
- je to DAG (nejsou tam orientované cykly)



Zdrojová komponenta
- nic do ní

Stohová komp.
- nic z ní vyvede

Jednoduše se hledá vrchol ve zdrojové komponentě. Opakovaným DFS prohledáním a vrchol s největším číslem out je vždycky ve zdrojové komp.

(vrcholy z jejíž nenavštívíme vrcholy)

Při vyhledávání se nikdy nedostaneme z této komponenty.

↳ Poslední vrchol, který jsme opustili, musí ležet ve zdrojové komponentě, jinak bychom ho už navštívili.

Našli jsme zdrojovou. Jak ale najít stohovou?

$G^T := (V(G), \{vu \mid uv \in E(G)\})$ → tedy jsme doslova otočili orientaci šipek.

G je DAG $\iff$ G^T je DAG. $\Rightarrow$ mají stejné silně souvislé třídy.

Zdroj $\xleftarrow{G^T}$ Stoh; protože jsme otočili šipky v orientovaném grafu, takže nyní hledám opět zdrojovou komponentu, která je ve skutečnosti stohová.

↳ Jakožto je to horečnatý, ale pomalý. Protože prohledávám znovu a znovu zbylé komponenty.

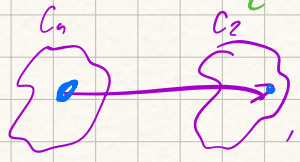
Projdeme tedy vrcholy v pořadí klesajících outů v G^T, pokud jsme nenašli přiřazenou komponentu, spouštím na nich DFS.

- Musím ale dokázat, že po ukončení prohledávání v jedné komponentě je další nenavštívený vrchol s nejvyšším outem další složka.

Lemma: Pokud C_1, C_2 jsou komponenty t.ž. $C_1C_2 \in E(C(G))$, pak:

$\max_{u \in C_1} out(u) > \max_{v \in C_2} out(v)$ — v orientovaném grafu! — v DFS na H

Vždy mám dvě komponenty



, tak z té první odejdu později.

Případy: → DFS vstoupí do C_1 před C_2.

- pak to platí. Nejdřív prohledám C_1, pak skočím do C_2, tu prohledám a do C_1 se vrátím až později.

DFS vstoupí do C_2 první

- zase první musím prohledat celou C_2 a do C_1 nevstoupím, jinak by to byla stejná komponenta. Následně do C_1 vstoupím až později po uzavření C_2, tedy zase bude out u vrcholu v C_1 mít větší out.

Algoritmus:

1. Sestrojíme G^T $O(n+m)$
2. $Z \leftarrow$ prázdný zásobník — konst $\rightarrow O(n+m)$
3. Opakování DFS v G^T, při opouštění vrcholu přidávám do Z. (v pořadí rostoucích outů)
4. $\forall v$ $comp(v) \leftarrow \emptyset$ $O(n)$
5. Postupně odebírám vrcholy ze Z, pro vrchol v: pokud $comp(v) = \emptyset$ $O(n)$

6

⑦ Spustíme DFS v G z vrcholu v, chodíme jen do vrcholů s $comp = \emptyset$ a nastavujeme $comp \Leftarrow v$ — $O(n+m)$

Celkově tedy lineární časová i paměťová složitost.

Algoritmus najde komponenty silné souvislosti v čase a prostoru $\Theta(n+m)$.

Kde používáme v praxi?

- ověření silné souvislosti (pokud existuje jenom jedna či více komp.)
- často se používá hlavně jako nástroj do mezivýpočtu

Nejkratší cesty v ohodnoceném grafu:

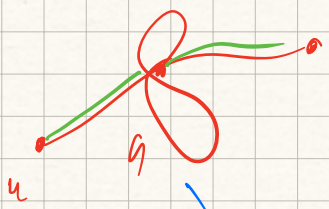
$\ell : E \to \mathbb{R}_0^+$: délka hrany

Vzdálenost je délka nejkratší cesty.

- délka cesty P: $\ell(P) := \sum_{p \in P} \ell(p)$

- vzdálenost $d(u,v) := \min \{ \ell(P) \mid P \text{ je } u,v\text{-cesta} \}$
  - pokud neexistuje cesta, $d := +\infty$

Pokud s je u,v-sled, pak $\exists P$ u,v-cesta t.ž. $\ell(p) \leq \ell(s)$



- pokud je sled cesta, hotovo.
- jinak se mi nějaký vrchol opakuje.

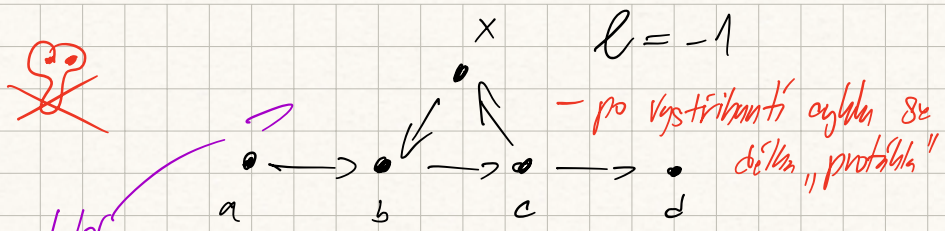
vybrat část před prvním a po posledním výskytu q.

Takováto cesta musí být nejvýš stejně dlouhá jako s.

$d(u,v) = \min \{ \ell(s) \mid s \text{ je } u,v \text{ sled} \}$

△ nerovnost: $\forall u,v,w \quad d(u,v) \leq d(u,v) + d(v,w)$

Záporné hrany



- pro vystřižení cyklu se délka „protáhla"

Každým dalším průchodem smyčkou by se úlet „zhoršil", takže to jde do nekonečna a nejde vzdálenost určit

$d(a,d) = -3$

$d(a,x) = -3$

$d(x,d) = -3$

$\Delta \neq: \; -3 \leq -3 + -3$

$-3 \not\leq -6$

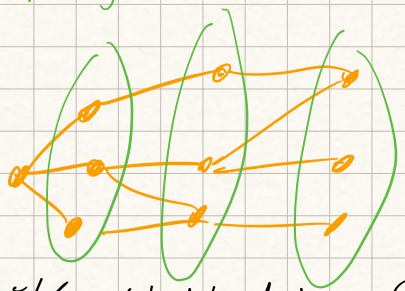
Najít nejkratší cestu je těžký.

Nic z toho se neděje, pokud jsou zakázány záporné cykly.

Důležité případy:

- konstantní vzdálenost všech hran:
  - klasický BFS

vrstvy ~ vzdálenosti od startu

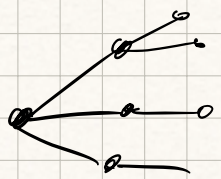


- každý vrchol tak dostane číslo vrstvy, tedy délku cesty.
- pro rekonstrukci cesty si pamatuji i předchůdce.

$O(n+m)$

→ Strom nejkratších cest:

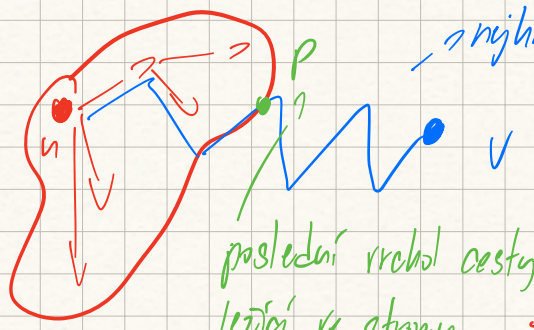
- cestu lze reprezentovat stromem:
  - strom na V
  - podgraf G
  - orientovaný od kořene, kterým je start hledání
  - $\forall v \in V$: cesta ve stromu (u,v) je jedna z nejkratších v G.



Kompaktní reprezentace z u do kamkoliv.

- 2 kružby vzdálenosti mezi (u,x) bude zobrazena pravě jedna cesta, přestože v G jich může být nekonečno.

Strom nejkratších cest existuje i v orientovaných grafech:



→ nejkratší (u,v) cesta

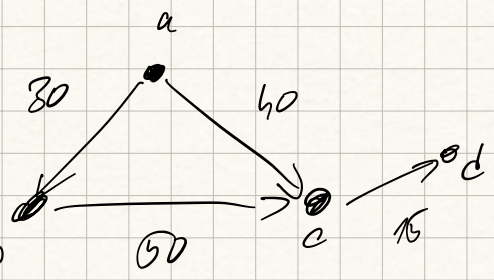
poslední vrchol cesty ležící ve stromu.

→ Pak vše mezi (p,v) je větev stromu, protože to je část cesty.

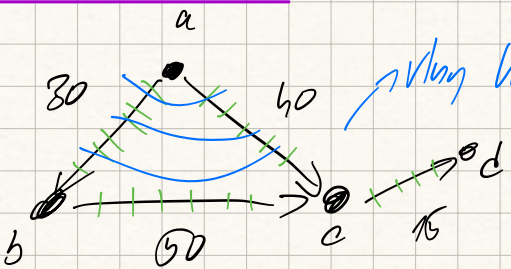
Prefix nejkratší cesty je zase nejkratší cesta.

Pokud (u,v) je nejkratší cesta, pak (u,w) má nejkratší cesta stejnou. Jinak by se dalo z u do w dojít jinudy a šlo by se tak i z u do v.

$\ell(e) \in \mathbb{N}$ a nemusí být stejná pro všechny hrany:



Složité řešení:



→ vlny kolem

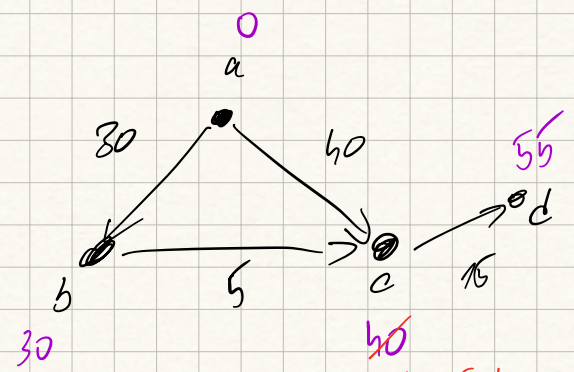
- rozdělím si hrany na jednotkové hrany... Pak spustím BFS.

→ maximální délka.

↳ Nový graf $O(m \cdot L)$ hran

Lepší řešení:

- Budeme mít „budíky" pro jednotlivé vrcholy.
  - ten znamená, kdy se poprvé vlna dostane do vrcholu. (např. b má budík 30).



→ Pokud se dostanu do vrcholu, kde už je nastaven budík a jeho hodnota je vyšší než bych nastavil já, přepíšu jeho budík, protože moje cesta reprezentuje nejlevnější cestu.

Dijkstrův als.