4. Aplikace DFS *do použitelné podoby dokopal Jan „Moskyto“ Matějka*

Detekce cyklů v orientovaném grafu

Algoritmus na vypsání *všech* cyklů v grafu si neukážeme, neboť tento problém je těžší, než se zdá. Mimo jiné proto, že cyklů může být až exponenciálně mnoho. Zde probíráme nalezení *nějakého* cyklu v orientovaném grafu.

Definice: $DFS(G)$ značíme spuštění DFS tak, aby prošel všechny vrcholy.

Realizace předchozí definice je jednoduchá.

1. Dokud existuje nenavštívený vrchol v:
2. Spustíme $DFS(v)$.

Lemma: Cyklus je v grafu G právě tehdy, když $DFS(G)$ označí nějakou hranu jako zpětnou.

Důkaz: Buď $v_1, v_2, \ldots, v_k, v_1$ cyklus v grafu G a v_1 buď BÚNO vrchol s max. $out(v_i)$. Pak jistě $out(v_k) < out(v_1)$. Jakého typu je hrana (v_k, v_1)?

Uvažme pořadí opouštění vrcholů na různých typech hran (x, y). Na stromové, dopředné a příčné hraně platí $out(x) > out(y)$, na zpětné $out(x) < out(y)$. Tedy (v_k, v_1) musí být nutně zpětná.

V opačném směru nalezneme pro zpětnou hranu (x, y) cestu z y do x po stromových hranách. ♡

Algoritmus:

1. *Vstup:* Graf G
2. Spustíme $DFS(G)$ a při nalezení první zpětné hrany (u, v) zastavíme.
3. Pokud jsme nalezli zpětnou hranu:
4. Vypíšeme všechny vrcholy mezi (u, v) ze zásobníku onoho DFS v opačném pořadí.
5. jinak
6. Graf je acyklický.

Časová i paměťová složitost je $\mathcal{O}(m + n)$, neboť se jedná o triviálně upravené DFS.

Topologické uspořádání DAGu (grafu bez orientovaných cyklů)

Definice: *Topologické uspořádání* acyklického orientovaného grafu G o k vrcholech je ohodnocení vrcholů čísly $1 \ldots k$ tak, že $(u, v) \in E(G) \Rightarrow f(u) < f(v)$.

Pozorování: Vhodné topologické uspořádání nám dá například *out* z $DFS(G^R)$. G^R budiž graf G, ve kterém otočíme orientace všech hran.

Hledání mostů v neorientovaném grafu

Definice: Hrana $uv \in E(G)$ je *most*, pokud se jejím odebráním z grafu G zvýší počet komponent souvislosti tohoto grafu.

Pozorování: Hrana, která je v nějakém cyklu, nemůže být mostem. Z cyklu totiž lze libovolně odebrat jednu hranu a přesto zůstane souvislý. Všechny ostatní hrany naopak mosty jsou.

Může být mostem jiná hrana než stromová? Zpětné hrany nutně tvoří cyklus. Příčné a dopředné se v neorientovaném DFS neobjeví.

Hledáme tedy všechny stromové hrany, které nejsou v žádném cyklu. Projdeme graf zase pomocí upraveného DFS.

Kdy je hrana uv v nějakém cyklu? Když existuje cesta $v \dots w$ (a nebo $v = w$), zpětná hrana wx a cesta $x \dots u$ (a nebo $x = u$).

Algoritmus bude ještě trochu více upravené DFS. Pro každý vrchol v si budeme kromě hloubky ještě pamatovat $z(v)$. To bude nejmenší *in* vrcholu, do kterého se dá dostat po zpětných hranách z vrcholu v nebo podstromu pod ním.

1. Vstoupíme do vrcholu u jako DFS.
2. $x \leftarrow$ minimum z $in(v)$ přes všechny zpětné hrany uv
3. $y \leftarrow$ minimum ze $z(w)$ přes všechny stromové hrany uw
4. $z(u) \leftarrow \min xy$
5. Pokud $z(u) \geq in(u)$,
6. je jistě mostem hrana, po které jsme vstoupili do u.

Časová i paměťová složitost tohoto upraveného DFS jsou $\mathcal{O}(m + n)$.

♡

Nejdelší cesta v ohodnoceném DAGu

Definice: Pro $u, v \in V$ bude $D(u, v)$ délka nejdelší cesty z u do v. $D^R(u, v)$ bude délka nejdelší cesty v grafu s otočenými hranami.

Neexistuje-li cesta z u do v, nechť $D(u, v) = -\infty$.

Algoritmus:

1. *Vstup:* Graf G, v něm vrchol u.
2. Zvolíme topologické uspořádání $w_1 \dots w_n$ na G. Nechť $w_k = u$.
3. Předpočítáme si ke každému vrcholu všechny jeho předchůdce, tedy množinu $W_i = \{w_j \mid (w_j, w_i) \in E\}$.
4. Pro všechny vrcholy w_i před u ($\forall w_i : i < k$) nastavíme délku cesty $D(u, w_i) = -\infty$, pro u pak $D(u, u) = 0$.
5. Postupně procházíme vrcholy $w_i \in V(G)$ v topologickém pořadí a pro každý z nich spočítáme $D(u, w_i)$.

$$D(u, w_i) = \max_{w_j \mid (w_j, w_i) \in E} (D(u, w_j) + e(w_j, w_i))$$

6. *Výstup:* $\big\{D(u, w_i)\big\}$.

Časová složitost: Sestrojení topologického uspořádání a předpočet v $\mathcal{O}(n + m)$. Postupné počítání $D(u, w)$ také v $\mathcal{O}(n + m)$, celkem $\mathcal{O}(n + m)$.
Paměťová složitost: Předpočet předchůdců zabere $\mathcal{O}(n + m)$ paměti.

Hledání kritických hran v ohodnoceném DAG-u

Definice: Hrana je kritická právě tehdy, když leží na některé z nejdelších cest.
Pozorování: Hrana (x, y) je kritická právě tehdy, když $D(u, x) + e(x, y) = D(u, y)$.
Algoritmus:

1. *Vstup:* Graf G, v něm vrchol u.

2. Nalezneme v grafu G nejdelší cesty z u předchozím algoritmem
3. Vybereme ty hrany, které splňují rovnost $D(u,x) + e(x,y) = D(u,y)$ – kritické
4. *Výstup:* Seznam kritických hran.

Časová a paměťová složitost: $\mathcal{O}(n+m)$

Rozkládání orientovaných grafů na komponenty silné souvislosti

Definice: R bude relace na $V(G)$ taková, že uRv právě tehdy, když existuje orientovaná cesta v G z u do v a současně z v do u.
Definice: G je *silně souvislý* právě tehdy, když $\forall(u,v) \in V(G) : uRv$.
Definice: *Komponenty silné souvislosti* grafu G jsou ekvivalenční třídy relace R.
Poznámka: Je R vůbec ekvivalence? Ano, zkuste si v definici prohodit u a v.
Definice: *Graf komponent* $C(G)$
$V(C(G))$: Komponenty silné souvislosti grafu G
$(C_1, C_2) \in E(C(G)) \Leftrightarrow \exists v_1 \in C_1, v_2 \in C_2 : (v_1, v_2) \in E(G)$

Lemma: Graf komponent $C(G)$ každého grafu G je DAG.
Důkaz: Sporem: Nechť $C_1, C_2, \ldots C_k$ tvoří cyklus v $C(G)$. Podle definice grafu komponent tedy musí existovat vrcholy $x_1 \ldots x_k \in C_i$ a $y_1 \ldots y_k \in C_{i+1}$ takové, že (x_i, y_i) jsou hrany grafu G.

Všechny C_i jsou silně souvislé, tedy existuje cesta z $y_{i-1} \pmod k$ do x_i v C_i. Slepíme nyní všechny tyhle cesty a hrany za sebe.

$$x_1 \to y_1 \to \ldots \to x_2 \to y_2 \to \ldots \to x_3 \to y_3 \to \ldots\ldots\ldots \to x_k \to y_k \to \ldots \to x_1$$

Slepením vznikne cyklus v G, což je spor, neboť všechny vrcholy v cyklu musí ležet v jedné komponentě souvislosti. ♡

Definice: *Zdroj* v grafu G je takový vrchol, do kterého nevedou žádné hrany
Definice: *Zdrojová komponenta* grafu G je taková komponenta silné souvislosti, která tvoří zdroj v $C(G)$.
Trik: Uvažujme graf G^R (graf G, ve kterém otočíme orientace všech hran). Pokud v leží ve zdrojové komponentě grafu G, pak $DFS(v)$ v G^R projde právě komponenty G.

Pustíme-li $DFS(G)$, pak vrchol s maximálním $out(v)$ leží nutně ve zdrojové komponentě (laskavý čtenář dokáže samostatně).
Tvrzeníčko: Pokud $(C_1, C_2) \in E(C(G))$, pak

$$\max_{x \in C_1} out(x) > \max_{x \in C_2} out(x).$$

Důkaz:

- Buďto DFS vstoupí nejdříve do C_1 – někdy odtamtud dojde do C_2 a zase se někdy vrátí, rozhodně ale dříve než z C_1. Pro tento případ tvrzeníčko platí.

- Nebo vstoupí nejdříve do C_2. Odtud nemůže nikdy dojít do C_1. Vrátí se tedy rozhodně dříve z celé C_2, než kdy vůbec vstoupí do C_1, tedy pro tento případ také tvrzeníčko platí.

♡

Vybereme si tedy vrchol v_1 ve zdrojové komponentě grafu G (ten s maximálním $out(v_1)$), spustíme $DFS(v)$ v G^R a všechny dosažené vrcholy w označkujeme – $komp(w) \leftarrow v$.

Nyní si vybereme vrchol v_2 ve zdrojové komponentě neoznačkované části G' grafu G (ten s maximálním $out(v_2)$) ... a algoritmus analogicky opakujeme, dokud existují neoznačkované vrcholy.

Pozorování: Neoznačkovaná část G' grafu G je prázdná, nebo má zdrojovou komponentu. Kdyby zdrojovou komponentu neměla, tak nemá zdroj ani $C(G')$, tedy $C(G')$ buďto obsahuje cyklus (spor s definicí), nebo je prázdný.

Algoritmus:

1. *Vstup:* Graf G
2. Sestrojíme G^R
3. $Z \leftarrow$ prázdný zásobník, $komp(*) \leftarrow ?$
4. Spustíme $DFS(G)$, při opuštení vrcholu jej vložíme do Z. Máme tedy vrcholy v zásobníku setříděné podle $out(v)$.
5. Postupně pro $v \in Z$:
6. Pokud $komp(v) = ?$
7. pustíme $DFS(v)$ v G^R s omezením jen na vrcholy w, pro které $komp(w) = ?$, všem navštíveným vrcholům w nastavíme $komp(w) \leftarrow v$.
8. *Výstup:* Pro každý vrchol v vrátíme jeho komponentu $komp(v)$.

Časová a paměťová složitost bude $\mathcal{O}(m+n)$, neboť první DFS má $\mathcal{O}(m+n)$, každé další DFS se omezí jen na svoji komponentu silné souvislosti a součet velikostí všech komponent souvislosti je $\mathcal{O}(m+n)$.