5. Minimální kostry

Tato kapitola uvede problém minimální kostry, základní věty o kostrách a klasické algoritmy na hledání minimálních koster. Budeme se inspirovat Tarjanovým přístupem z knihy [1]. Všechny grafy v této kapitole budou neorientované multigrafy a jejich hrany budou ohodnoceny vahami $w : E \to \mathbb{R}$.

Minimální kostry a jejich vlastnosti

Definice:

- *Podgrafem* budeme v této kapitole mínit libovolnou podmnožinu hran, vrcholy vždy zůstanou zachovány.
- *Přidání a odebrání hrany* budeme značit $T+e := T \cup \{e\}$, $T-e := T \setminus \{e\}$.
- *Kostra* (Spanning Tree) souvislého grafu G je libovolný jeho podgraf, který je strom. Kostru nesouvislého grafu definujeme jako sjednocení koster jednotlivých komponent. [Alternativně: kostra je minimální podgraf, který má komponenty s týmiž vrcholy jako komponenty G.]
- *Váha* podgrafu $F \subseteq E$ je $w(F) := \sum_{e \in F} w(e)$.
- *Minimální kostra* (Minimum Spanning Tree, mezi přáteli též MST) budeme říkat každé kostře, jejíž váha je mezi všemi kostrami daného grafu minimální.

Toto je sice standardní definice MST, ale jinak je dosti nešikovná, protože vyžaduje, aby bylo váhy možné sčítat. Ukážeme, že to není potřeba.

Definice: Buď $T \subseteq G$ nějaká kostra grafu G. Pak:

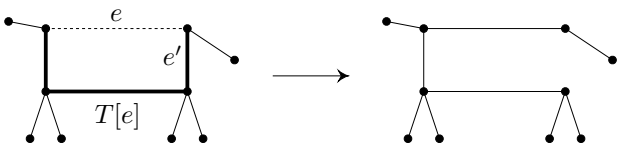
- $T[x, y]$ bude značit cestu v T, která spojuje x a y. (Cestou opět míníme množinu hran.)
- $T[e] := T[x, y]$ pro hranu $e = xy$. Této cestě budeme říkat *cesta pokrytá hranou* e.
- Hrana $e \in E \setminus T$ je *lehká vzhledem k* $T \equiv \exists e' \in T[e] : w(e) < w(e')$. Ostatním hranám neležícím v kostře budeme říkat *těžké.*

Věta: Kostra T je minimální $\Leftrightarrow$ neexistuje hrana lehká vzhledem k T.

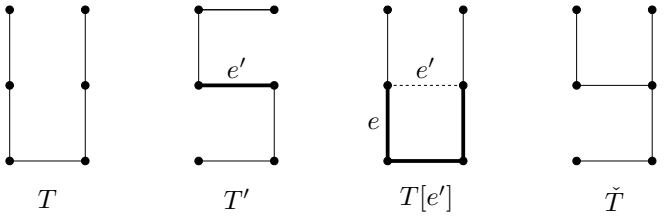
Tato věta nám dává pěknou alternativní definici MST, která místo sčítání vah váhy pouze porovnává, čili jí místo čísel stačí lineární (kvazi)uspořádání na hranách. Než se dostaneme k jejímu důkazu, prozkoumejme nejdříve, jak se dá mezi jednotlivými kostrami přecházet.

Definice: Pro kostru T a hrany e, e' zaveďme $swap(T, e, e') := T - e + e'$.

Pozorování: Pokud $e' \notin T$ a $e \in T[e']$, je $swap(T, e, e')$ opět kostra. Stačí si uvědomit, že přidáním e' do T vznikne kružnice (konkrétně $T[e'] + e'$) a vynecháním libovolné hrany z této kružnice získáme opět kostru.



Kostra T, cesta $T[e]$ a výsledek operace $swap(T, e', e)$



Jeden krok důkazu swapovacího lemmatu

Lemma o swapování: Máme-li libovolné kostry T a T', pak lze z T dostat T' konečným počtem operací *swap*.

Důkaz: Pokud $T \neq T'$, musí existovat hrana $e' \in T' \setminus T$, protože $|T| = |T'|$. Kružnice $T[e'] + e'$ nemůže být celá obsažena v T', takže existuje hrana $e \in T[e'] \setminus T'$ a $\check{T} := swap(T, e, e')$ je kostra, pro kterou $|\check{T} \,\Delta\, T'| = |T \,\Delta\, T'| - 2$. Po konečném počtu těchto kroků tedy musíme dojít k T'. ♡

Monotónní lemma o swapování: Je-li T kostra, k níž neexistují žádné lehké hrany, a T' libovolná kostra, pak lze od T k T' přejít posloupností swapů, při které váha kostry neklesá.

Důkaz: Podobně jako u předchozího lemmatu budeme postupovat indukcí podle $|T \,\Delta\, T'|$. Pokud zvolíme libovolně hranu $e' \in T' \setminus T$ a k ní $e \in T[e'] \setminus T'$, musí $\check{T} := swap(T, e, e')$ být kostra bližší k T' a $w(\check{T}) \geq w(T)$, jelikož e' nemůže být lehká vzhledem k T, takže speciálně $w(e') \geq w(e)$.

Aby mohla indukce pokračovat, potřebujeme ještě dokázat, že ani k nové kostře neexistují lehké hrany v $T' \setminus \check{T}$. K tomu nám pomůže zvolit si ze všech možných hran e' tu s nejmenší vahou. Uvažme nyní hranu $f \in T' \setminus \check{T}$. Cesta $\check{T}[f]$ pokrytá touto hranou v nové kostře je buďto původní $T[f]$ (to pokud $e \notin T[f]$) nebo $T[f] \,\Delta\, C$, kde C je kružnice $T[e'] + e'$. První případ je triviální, ve druhém si stačí uvědomit, že $w(f) \geq w(e')$ a ostatní hrany na C jsou lehčí než e'. ♡

Důkaz věty:

- $\exists$ lehká hrana $\Rightarrow$ T není minimální.
  Nechť $\exists e$ lehká. Najdeme $e' \in T[e] : w(e) < w(e')$ (ta musí existovat z definice lehké hrany). Kostra $T' := swap(T, e, e')$ je lehčí než T.

- K T neexistuje lehká hrana $\Rightarrow$ T je minimální.

Uvažme nějakou minimální kostru T_{min} a použijme monotónní swapovací lemma na T a T_{min}. Z něj plyne $w(T) \leq w(T_{min})$, a tedy $w(T) = w(T_{min})$. ♡

Věta: Jsou-li všechny váhy hran navzájem různé, je MST určena jednoznačně.

Důkaz: Máme-li dvě MST T_1 a T_2, neobsahují podle předchozí věty lehké hrany, takže podle monotónního lemmatu mezi nimi lze přeswapovat bez poklesu váhy. Pokud jsou ale váhy různé, musí každé swapnutí ostře zvýšit váhu, a proto k žádnému nemohlo dojít. ♡

Poznámka: Často se nám bude hodit, aby kostra, kterou hledáme, byla určena jednoznačně. Tehdy můžeme využít předchozí věty a váhy změnit o vhodné epsilony, respektive kvaziuspořádání rozšířit na lineární uspořádání.

Červenomodrý meta-algoritmus

Všechny tradiční algoritmy na hledání MST lze popsat jako speciální případy následujícího meta-algoritmu. Rozeberme si tedy rovnou ten. Formulujeme ho pro případ, kdy jsou všechny váhy hran navzájem různé.

Meta-algoritmus:

1. Na počátku jsou všechny hrany bezbarvé.
2. Dokud to lze, použijeme jedno z následujících pravidel:
3. *Modré pravidlo:* Vyber řez takový, že jeho nejlehčí hrana není modrá,[⟨1⟩] a obarvi ji na modro.
4. *Červené pravidlo:* Vyber cyklus takový, že jeho nejtěžší hrana není červená, a obarvi ji na červeno.

Věta: Pro Červenomodrý meta-algoritmus spuštěný na libovolném grafu s hranami lineárně uspořádanými podle vah platí:

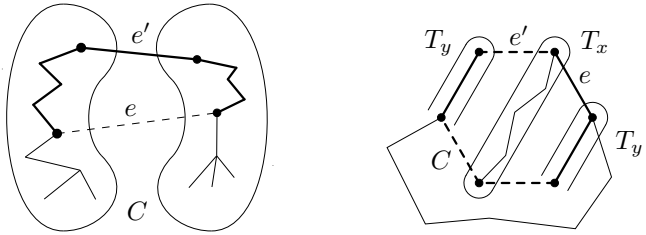
1. Vždy se zastaví.
2. Po zastavení jsou všechny hrany obarvené.
3. Modře obarvené hrany tvoří minimální kostru.

Důkaz: Nejdříve si dokážeme několik lemmat. Jelikož hrany mají navzájem různé váhy, můžeme předpokládat, že algoritmus má sestrojit jednu konkrétní minimální kostru T_{min}.

Modré lemma: Je-li libovolná hrana e algoritmem kdykoliv obarvena na modro, pak $e \in T_{min}$.

Důkaz: Sporem: Hrana e byla omodřena jako nejlehčí hrana nějakého řezu C. Pokud $e \notin T_{min}$, musí cesta $T_{min}[e]$ obsahovat nějakou jinou hranu e' řezu C. Jenže e' je těžší než e, takže operací $swap(T_{min}, e', e)$ získáme ještě lehčí kostru, což není možné. ♡

[⟨1⟩] Za touto podmínkou nehledejte žádná kouzla, je tu pouze proto, aby se algoritmus nemohl zacyklit neustálým prováděním pravidel, která nic nezmění.



Situace v důkazu Modrého a Červeného lemmatu

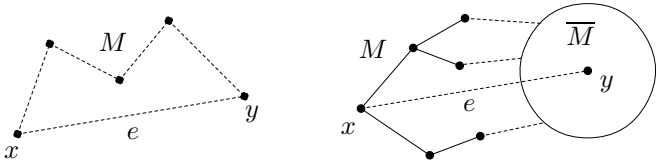
Červené lemma: Je-li libovolná hrana e algoritmem kdykoliv obarvena na červeno, pak $e \notin T_{min}$.

Důkaz: Opět sporem: Předpokládejme, že e byla obarvena červeně jako nejtěžší na nějaké kružnici C a že $e \in T_{min}$. Odebráním e se nám T_{min} rozpadne na dvě komponenty T_x a T_y. Některé vrcholy kružnice připadnou do komponenty T_x, ostatní do T_y. Na C ale musí existovat nějaká hrana $e' \neq e$, jejíž krajní vrcholy leží v různých komponentách, a jelikož hrana e byla na kružnici nejtěžší, je $w(e') < w(e)$. Pomocí $swap(T_{min}, e, e')$ proto získáme lehčí kostru, a to je spor. ♡

Bezbarvé lemma: Pokud existuje nějaká neobarvená hrana, lze ještě použít některé z pravidel.

Důkaz: Nechť existuje hrana $e = xy$, která je stále bezbarvá. Označíme si M množinu vrcholů, do nichž se lze z x dostat po modrých hranách. Nyní mohou nastat dvě možnosti:

- $y \in M$ (tj. existuje modrá cesta z x do y): Modrá cesta je v minimální kostře a k minimální kostře neexistují žádné lehké hrany, takže hrana e je nejdražší na cyklu tvořeném modrou cestou a touto hranou a mohu na ni použít červené pravidlo.



Situace v důkazu Bezbarvého lemmatu

- $y \notin M$: Tehdy řez $\delta(M)$ neobsahuje žádné modré hrany, takže na tento řez můžeme použít modré pravidlo. ♡

Důkaz věty:

- *Zastaví se:* Z červeného a modrého lemmatu plyne, že žádnou hranu nikdy nepřebarvíme. Každým krokem přibude alespoň jedna obarvená hrana, takže se algoritmus po nejvýše m krocích zastaví.

- *Obarví vše:* Pokud existuje alespoň jedna neobarvená hrana, pak podle bezbarvého lemmatu algoritmus pokračuje.
- *Najde modrou MST:* Podle červeného a modrého lemmatu leží v T_{min} právě modré hrany. ♡

Poznámka: Červené a modré pravidlo jsou v jistém smyslu duální. Pro rovinné grafy je na sebe převede obyčejná rovinná dualita (stačí si uvědomit, že kostra duálního grafu je komplement duálu kostry primárního grafu), obecněji je to dualita mezi matroidy, která prohazuje řezy a cykly.

Klasické algoritmy na hledání MST

Kruskalův neboli Hladový:[2]

1. Setřídíme hrany podle vah vzestupně.
2. Začneme s prázdnou kostrou (každý vrchol je v samostatné komponentě souvislosti).
3. Bereme hrany ve vzestupném pořadí.
4. Pro každou hranu e se podíváme, zda spojuje dvě různé komponenty – pokud ano, přidáme ji do kostry, jinak ji zahodíme.

Červenomodrý pohled: pěstujeme modrý les. Pokud hrana spojuje dva stromečky, je určitě minimální v řezu mezi jedním ze stromečků a zbytkem grafu (ostatní hrany téhož řezu jsme ještě nezpracovali). Pokud nespojuje, je maximální na nějakém cyklu tvořeném touto hranou a nějakými dříve přidanými.

Potřebujeme čas $\mathcal{O}(m \log n)$ na setřídění hran a dále datovou strukturu pro udržování komponent souvislosti (Union-Find Problem), se kterou provedeme m operací *Find* a n operací *Union*. Nejlepší známá implementace této struktury dává složitost obou operací $\mathcal{O}(\alpha(n))$ amortizovaně, takže celkově hladový algoritmus doběhne v čase $\mathcal{O}(m \log n + m\alpha(n))$.

Borůvkův:

Opět si budeme pěstovat modrý les, avšak tentokrát jej budeme rozšiřovat ve fázích. V jedné fázi nalezneme ke každému stromečku nejlevnější incidentní hranu a všechny tyto nalezené hrany naráz přidáme (aplikujeme několik modrých pravidel najednou). Pokud jsou všechny váhy různé, cyklus tím nevznikne.

Počet stromečků klesá exponenciálně $\Rightarrow$ fází je celkem $\log n$. Pokud každou fázi implementujeme lineárním průchodem celého grafu, dostaneme složitost $\mathcal{O}(m \log n)$. Mimo to lze každou fázi výtečně paralelizovat.

Jarníkův:

Jarníkův algoritmus je podobný Borůvkovi, ale s tím rozdílem, že nenecháme růst celý les, ale jen jeden modrý strom. V každém okamžiku nalezneme nejlevnější hranu vedoucí mezi stromem a zbytkem grafu a přidáme ji ke stromu (modré pravidlo); hrany vedoucí uvnitř stromu průběžně zahazujeme (červené pravidlo). Kroky

[2] Možná hladový s malým ‘h’, ale tento algoritmus je pradědečkem všech ostatních hladových algoritmů, tak mu tu čest přejme.

opakujeme, dokud se strom nerozroste přes všechny vrcholy. Při šikovné implementaci pomocí haldy dosáhneme časové složitosti $\mathcal{O}(m \log n)$, v příští kapitole ukážeme implementaci ještě šikovnější.

Cvičení: Nalezněte jednoduchý algoritmus pro výpočet MST v grafech ohodnocených vahami $\{1, \ldots k\}$ se složitostí $\mathcal{O}(mk)$ nebo dokonce $\mathcal{O}(m + nk)$.

Literatura

[1] R. E. Tarjan. *Data structures and network algorithms*, volume 44 of *CMBS-NSF Regional Conf. Series in Appl. Math.* SIAM, 1983.