Matematika III – 10. přednáška
Stromy a kostry

Michal Bulant

Masarykova univerzita
Fakulta informatiky

20. 11. 2007

Obsah přednášky

Doporučené zdroje

- Martin Panák, Jan Slovák, Drsná matematika, *e*-text.
- *Předmětové záložky v IS MU*

Doporučené zdroje

- Martin Panák, Jan Slovák, Drsná matematika, *e*-text.
- *Předmětové záložky v IS MU*
- Jiří Matoušek, Jaroslav Nešetřil, Kapitoly z diskrétní matematiky, Univerzita Karlova v Praze, Karolinum, Praha, 2000, 377 s.
- Petr Hliněný, Teorie grafů, studijní materiály, http://www.fi.muni.cz/~hlineny/Vyuka/GT/

Doporučené zdroje

- Martin Panák, Jan Slovák, Drsná matematika, *e*-text.
- *Předmětové záložky v IS MU*
- Jiří Matoušek, Jaroslav Nešetřil, Kapitoly z diskrétní matematiky, Univerzita Karlova v Praze, Karolinum, Praha, 2000, 377 s.
- Petr Hliněný, Teorie grafů, studijní materiály, http://www.fi.muni.cz/~hlineny/Vyuka/GT/
- Donald E. Knuth, The Stanford GraphBase, ACM, New York, 1993 (`http://www-cs-faculty.stanford.edu/~knuth/sgb.html`).

Plán přednášky

Stromy se často používají jako (acyklické) datové struktury, v praxi stromy procházíme v určitém pořadí vrcholů – narozdíl od grafů k jednoznačnému určení nějakého uspořádání vrcholů stačí vybrat jeden vrchol – *kořen* (root) v_r.
Ve stromu není žádná kružnice, proto volba jednoho vrcholu v_r zadává orientaci všech hran.

Stromy se často používají jako (acyklické) datové struktury, v praxi stromy procházíme v určitém pořadí vrcholů – narozdíl od grafů k jednoznačnému určení nějakého uspořádání vrcholů stačí vybrat jeden vrchol – *kořen* (root) v_r.
Ve stromu není žádná kružnice, proto volba jednoho vrcholu v_r zadává orientaci všech hran.
Po výběru **kořenu** se začíná graf více podobat skutečnému stromu v přírodě. Stromy s jedním vybraným kořenem nazýváme **kořenové stromy**.

Stromy se často používají jako (acyklické) datové struktury, v praxi stromy procházíme v určitém pořadí vrcholů – narozdíl od grafů k jednoznačnému určení nějakého uspořádání vrcholů stačí vybrat jeden vrchol – *kořen* (root) v_r.
Ve stromu není žádná kružnice, proto volba jednoho vrcholu v_r zadává orientaci všech hran.
Po výběru **kořenu** se začíná graf více podobat skutečnému stromu v přírodě. Stromy s jedním vybraným kořenem nazýváme **kořenové stromy**.

Definice

V kořenovém stromu $T = (V, E)$ je vrchol w je **následník vrcholu** v a naopak v je předchůdce vrcholu w právě tehdy, když existuje cesta z kořene stromu do w která prochází v a $v \neq w$. **Přímý následník** a **přímý předchůdce** vrcholu jsou pak následníci a předchůdci přímo spojení hranou. Mluvíme také o **synech** a **otcích** (patrně v narážce na genealogické stromy).

Definice

Binární stromy jsou speciálním případem kořenového stromu, kdy každý otec má nejvýše dva následníky (někdy se ale pod stejným označením binární strom předpokládá, že všechny vrcholy kromě listů mají právě dva následníky).

Definice

Binární stromy jsou speciálním případem kořenového stromu, kdy každý otec má nejvýše dva následníky (někdy se ale pod stejným označením binární strom předpokládá, že všechny vrcholy kromě listů mají právě dva následníky).

Často jsou vrcholy stromu spojeny s klíči v nějaké úplně uspořádané množině (např. reálná čísla) a slouží k hledání vrcholu s daným klíčem.
Je realizováno jako hledání cesty od kořene stromu a v každém vrcholu se podle velikosti rozhodujeme, do kterého ze synů budeme pokračovat (resp. zastavíme hledání, pokud jsme již ve hledaném vrcholu). Abychom mohli tuto cestu jednoznačně krok po kroku určovat, požadujeme aby jeden syn společně se všemi jeho následníky měli menší klíče než druhý syn a všichni jeho následníci.

Izomorfismus stromů

Již dříve jsme si říkali, že rozhodnout o izomorfismu dvou obecných grafů je velmi obtížný problém. U stromů je naštěstí, jak si ukážeme, situace podstatně jednodušší.

Izomorfismus stromů

Již dříve jsme si říkali, že rozhodnout o izomorfismu dvou obecných grafů je velmi obtížný problém. U stromů je naštěstí, jak si ukážeme, situace podstatně jednodušší. Pro popis všech možných izomorfismů (kořenových) stromů je užitečné kromě vztahů otec–syn ještě užitečné mít syny uspořádány v pořadí (třeba v představě odleva doprava nebo podle postupného růstu atd.).

Izomorfismus stromů

Již dříve jsme si říkali, že rozhodnout o izomorfismu dvou obecných grafů je velmi obtížný problém. U stromů je naštěstí, jak si ukážeme, situace podstatně jednodušší. Pro popis všech možných izomorfismů (kořenových) stromů je užitečné kromě vztahů otec–syn ještě užitečné mít syny uspořádány v pořadí (třeba v představě odleva doprava nebo podle postupného růstu atd.).

Definice

Pěstěný strom $T = (V, E, v_r, \nu)$ je kořenový strom společně s částečným uspořádáním ν na hranách takovým, že srovnatelné jsou vždy právě hrany směřující od jednoho otce k synům.
Morfismem kořenových stromů $T = (V, E, v_r)$ a $T' = (V', E', v_r')$ rozumíme takový morfismus grafů $\varphi : T \to T'$, který převádí v_r na v_r'. Obdobně pro izomorfismy.
Pro pěstěné stromy navíc požadujeme aby zobrazení hran zachovávalo částečná uspořádání ν a ν'.

Kódy stromů

Pro pěstěné stromy $T = (V, E, v_r, \nu)$ zavedeme jejich (jak uvidíme) jednoznačný popis pomocí slov z nul a jedniček. Obrazně si můžeme představit, že strom kreslíme a každý přírůstek naznačíme dvěma tahy, které si označíme 0 (dolů) a 1 (nahoru). Začneme od listů (příp. mimo kořene), kterým všem přiřadíme slovo 01. Celý strom pak budeme popisovat zřetězováním částí slov tak, že má-li otec v syny uspořádány jako posloupnost $v_1, \ldots, v_\ell$, a jsou-li již jednotliví synové označeni slovy $W_1, \ldots W_\ell$, pak pro otce použijeme slovo

$$0W_1 \ldots W_\ell 1.$$

Hovoříme o **kódu pěstěného stromu**.

Kódy stromů

Pro pěstěné stromy $T = (V, E, v_r, \nu)$ zavedeme jejich (jak uvidíme) jednoznačný popis pomocí slov z nul a jedniček.
Obrazně si můžeme představit, že strom kreslíme a každý přírůstek naznačíme dvěma tahy, které si označíme 0 (dolů) a 1 (nahoru).
Začneme od listů (příp. mimo kořene), kterým všem přiřadíme slovo 01. Celý strom pak budeme popisovat zřetězováním částí slov tak, že má-li otec v syny uspořádány jako posloupnost $v_1, \ldots, v_\ell$, a jsou-li již jednotliví synové označeni slovy $W_1, \ldots W_\ell$, pak pro otce použijeme slovo

$$0W_1 \ldots W_\ell 1.$$

Hovoříme o **kódu pěstěného stromu**.
Skutečně, kreslením cest dolů a nahoru získáme skutečně původní strom s jednou hranou směřující shora do kořene navíc.

Věta

Dva pěstěné stromy jsou izomorfní právě, když mají stejný kód.

Důkaz.

Z konstrukce je zřejmé, že izomorfní stromy budou mít stejný kód, zbývá tedy pouze dokázat, že neizomorfní stromy vedou na různé kódy, jinými slovy, že z daného kódy jednoznačně zrekonstruujeme výchozí pěstovaný strom.
Dokážeme to indukcí podle délky kódu (tj. počtu nul a jedniček) tak, že využijeme jednoznačné kódy pro všechny podstromy vzniklé odjemutím kořene.

Věta

Dva pěstěné stromy jsou izomorfní právě, když mají stejný kód.

Důkaz.

Z konstrukce je zřejmé, že izomorfní stromy budou mít stejný kód, zbývá tedy pouze dokázat, že neizomorfní stromy vedou na různé kódy, jinými slovy, že z daného kódy jednoznačně zrekonstruujeme výchozí pěstovaný strom.
Dokážeme to indukcí podle délky kódu (tj. počtu nul a jedniček) tak, že využijeme jednoznačné kódy pro všechny podstromy vzniklé odjemutím kořene.
Pro nejkratší kód 01 je situace snadná.

Věta

Dva pěstěné stromy jsou izomorfní právě, když mají stejný kód.

Důkaz.

Z konstrukce je zřejmé, že izomorfní stromy budou mít stejný kód, zbývá tedy pouze dokázat, že neizomorfní stromy vedou na různé kódy, jinými slovy, že z daného kódy jednoznačně zrekonstruujeme výchozí pěstovaný strom.

Dokážeme to indukcí podle délky kódu (tj. počtu nul a jedniček) tak, že využijeme jednoznačné kódy pro všechny podstromy vzniklé odjemutím kořene.

Pro nejkratší kód 01 je situace snadná. V indukčním kroku je dán kód délky $2(n+1)$ tvaru $0A1$, přičemž $A = A_1 A_2 \dots A_t$ je zřetězením několika kódů pěstovaných stromů. Část A_1 je tvořena nejkratším prexifex A, který má stejný počet 0 a 1, dále A_2, atd. Podle indukčního předpokladu každé A_i jednoznačně odpovídá pěstěnému stromu, z čehož zřejmě dostáváme jediný pěstěný strom odpovídající kódu $0A1$. □

Nyní převedeme testování izomorfismu kořenových stromů, resp. stromů na testování pěstěných stromů.

Nyní převedeme testování izomorfismu kořenových stromů, resp. stromů na testování pěstěných stromů.
U kořenových stromů lze využít kódy, pokud se podaří určit pořadí jejich synů jednoznačně až na izomorfismus. Na pořadí synů ovšem nezáleží právě tehdy, když jsou podgrafy určené jejich následníky izomorfní.

Nyní převedeme testování izomorfismu kořenových stromů, resp. stromů na testování pěstěných stromů.
U kořenových stromů lze využít kódy, pokud se podaří určit pořadí jejich synů jednoznačně až na izomorfismus. Na pořadí synů ovšem nezáleží právě tehdy, když jsou podgrafy určené jejich následníky izomorfní.
Využijeme proto obdobu (rekurzivní) konstrukce kódu pro pěstěné stromy – budeme postupovat obdobně s využitím lexikografického (slovníkového) uspořádní synů podle jejich kódů. Kořenový strom budeme tedy popisovat zřetězováním částí slov tak, že má-li otec v syny již označeny kódy $W_1, \ldots W_\ell$, pak pro otce použijeme slovo

$$0W_1 \ldots W_\ell 1$$

kde pořadí $W_1, \ldots, W_\ell$ je zvoleno tak aby $W_1 \leq W_2 \leq \cdots \leq W_\ell$.

Pokud není určen kořen ve stromě, můžeme jej určit tak, aby byl „přibližně uprostřed stromu“.

Pokud není určen kořen ve stromě, můžeme jej určit tak, aby byl „přibližně uprostřed stromu“.
Každému vrcholu stromu T přiřadíme hodnotou $\mathrm{ex}_G(v)$ tzv. **výstřednosti** (excentricity), kterou definujeme pro každý vrchol v jako maximální vzdálenost z v do jiného vrcholu w v T.

Tvrzení

Buď $C(T)$ množina vrcholů stromu T, jejichž výstřednost nabývá minimální hodnoty ($C(T)$ se nazývá střed/centrum grafu, minimální hodnota pak poloměr grafu). Pak $C(T)$ má jeden vrchol nebo dva vrcholy spojené hranou v T.

Pokud není určen kořen ve stromě, můžeme jej určit tak, aby byl „přibližně uprostřed stromu“.
Každému vrcholu stromu T přiřadíme hodnotou $\mathrm{ex}_G(v)$ tzv. **výstřednosti** (excentricity), kterou definujeme pro každý vrchol v jako maximální vzdálenost z v do jiného vrcholu w v T.

Tvrzení

Buď $C(T)$ množina vrcholů stromu T, jejichž výstřednost nabývá minimální hodnoty ($C(T)$ se nazývá střed/centrum grafu, minimální hodnota pak poloměr grafu). Pak $C(T)$ má jeden vrchol nebo dva vrcholy spojené hranou v T.

Důkaz.

Snadno indukcí s využitím triviálního faktu, že nejvzdálenějším vrcholem od každého vrcholu v je nutně list. Centrum T tedy splývá s centrem stromu T', který vznikne z T vypuštěním listů a příslušných hran. □

Libovolnému stromu přiřadíme jednoznačný kód, až na izomorfismus takto: Pokud je v centru T jediný vrchol, použijeme jej jako kořene; v opačném případě vytvoříme stejným způsobem kód pro dva stromy vzniklé z T odebráním hrany (bez vrcholů) spojující vrcholy x_1, x_2 v $C(T)$ a kód vznikne zřetězením kódů obou kořenových stromů $(T_1, x_1), (T_2, x_2)$ v pořadí podle lexikografického uspořádání těchto kódů.

Libovolnému stromu přiřadíme jednoznačný kód, až na izomorfismus takto: Pokud je v centru T jediný vrchol, použijeme jej jako kořene; v opačném případě vytvoříme stejným způsobem kód pro dva stromy vzniklé z T odebráním hrany (bez vrcholů) spojující vrcholy x_1, x_2 v $C(T)$ a kód vznikne zřetězením kódů obou kořenových stromů $(T_1, x_1), (T_2, x_2)$ v pořadí podle lexikografického uspořádání těchto kódů.

Věta

Dva stromy T a T' jsou izomorfní právě, když mají společný kód.

Libovolnému stromu přiřadíme jednoznačný kód, až na izomorfismus takto: Pokud je v centru T jediný vrchol, použijeme jej jako kořene; v opačném případě vytvoříme stejným způsobem kód pro dva stromy vzniklé z T odebráním hrany (bez vrcholů) spojující vrcholy x_1, x_2 v $C(T)$ a kód vznikne zřetězením kódů obou kořenových stromů $(T_1, x_1), (T_2, x_2)$ v pořadí podle lexikografického uspořádání těchto kódů.

Věta

Dva stromy T a T' jsou izomorfní právě, když mají společný kód.

Poznámka

Z uvedených úvah lze snadno nahlédnout, že algoritmus na testování izomorfismu stromů lze implementovat v lineárním čase vzhledem k počtu vrcholů.

Plán přednášky

V praktických aplikacích často zadává graf všechny možnosti propojení mezi objekty, příkladem může být třeba silniční nebo vodovodní nebo elektrická síť. Pokud nám stačí zajistit propojitelnost každých dvou vrcholů při minimálním počtu hran, hledáme vlastně v grafu G faktor T, který je stromem.

V praktických aplikacích často zadává graf všechny možnosti propojení mezi objekty, příkladem může být třeba silniční nebo vodovodní nebo elektrická síť. Pokud nám stačí zajistit propojitelnost každých dvou vrcholů při minimálním počtu hran, hledáme vlastně v grafu G faktor T, který je stromem.

Definice

Libovolný strom $T = (V, E')$ v grafu $G = (V, E)$, $E' \subseteq E$ se nazývá **kostra** (*spanning tree*) grafu G (tj. faktor grafu, který neobsahuje kružnice).

V praktických aplikacích často zadává graf všechny možnosti propojení mezi objekty, příkladem může být třeba silniční nebo vodovodní nebo elektrická síť. Pokud nám stačí zajistit propojitelnost každých dvou vrcholů při minimálním počtu hran, hledáme vlastně v grafu G faktor T, který je stromem.

Definice

Libovolný strom $T = (V, E')$ v grafu $G = (V, E)$, $E' \subseteq E$ se nazývá **kostra** (*spanning tree*) grafu G (tj. faktor grafu, který neobsahuje kružnice).

Evidentně může kostra v grafu existovat pouze pokud je graf G souvislý. Místo formálního důkazu, že platí i opak uvedeme přímo algoritmus, jak kostru grafu sestrojit.

Počet koster grafu K_n

Věta (Cayleyho formule)

Pro $n \geq 2$ je počet koster $\kappa(K_n)$ na K_n (tj. počet stromů na daných n vrcholech) roven n^{n-2}.

Poznámka

Počet koster je významný pojem používaný v mnoha aplikacích. Např. v elektrotechnice, při hypotetickém předpokladu jednotkového odporu mezi každými dvěma vrcholy spojenými hranou, naměříme mezi 2 vrcholy spojenými hranou (vodičem) odpor, který je roven počtu koster obsahujících tuto hranu lomeno celkový počet koster v grafu

Důkaz: Není znám žádný přímočarý způsob, jak dokázat platnost této jednoduché formule, lze ji ale dokázat mnoha různými způsoby (např. pomocí skóre, kódování koster, determinantů, či počítání *povykosů* – viz [MN]).

Důkaz: Není znám žádný přímočarý způsob, jak dokázat platnost této jednoduché formule, lze ji ale dokázat mnoha různými způsoby (např. pomocí skóre, kódování koster, determinantů, či počítání *povykosů* – viz [MN]).
Spočítáme dvěma způsoby *povykosy* (povykos = postup výroby kořenového stromu). Povykos je definován jako trojice (T, r, ν), kde T je strom na n vrcholech, r jeho kořen a ν očíslování hran, neboli bijekce $\nu : E(T) \to \{1, 2, \ldots, n-1\}$ (začínáme s prázdnou množinou hran a postupně přidáváme hranyv pořadí podle ν).

Důkaz: Není znám žádný přímočarý způsob, jak dokázat platnost této jednoduché formule, lze ji ale dokázat mnoha různými způsoby (např. pomocí skóre, kódování koster, determinantů, či počítání *povykosů* – viz [MN]).
Spočítáme dvěma způsoby *povykosy* (povykos = postup výroby kořenového stromu). Povykos je definován jako trojice (T, r, ν), kde T je strom na n vrcholech, r jeho kořen a ν očíslování hran, neboli bijekce $\nu : E(T) \to \{1, 2, \ldots, n-1\}$ (začínáme s prázdnou množinou hran a postupně přidáváme hranyv pořadí podle ν).
Pro každý strom T můžeme kořen r zvolit n způsoby a očíslování hran $(n-1)!$ způsoby, proto je počet povykosů $n(n-1)! \cdot \kappa(K_n)$.

Důkaz – pokr.:

Druhý způsob: kořenový strom budeme uvažovat jako orientovaný strom se šipkami směřujícími ke kořeni. Začneme s prázdným grafem a budeme přidávat šipky v $n-1$ krocích.

1. šipka: $n(n-1)$ možností.

Důkaz – pokr.:

Druhý způsob: kořenový strom budeme uvažovat jako orientovaný strom se šipkami směřujícími ke kořeni. Začneme s prázdným grafem a budeme přidávat šipky v $n-1$ krocích.

1. šipka: $n(n-1)$ možností.

další šipka:

- nesmí vytvořit kružnici;

Důkaz – pokr.:

Druhý způsob: kořenový strom budeme uvažovat jako orientovaný strom se šipkami směřujícími ke kořeni. Začneme s prázdným grafem a budeme přidávat šipky v $n-1$ krocích.

1. šipka: $n(n-1)$ možností.

další šipka:

- nesmí vytvořit kružnici;
- na konci musí z každého vrcholu kromě jediného vycházet nějaká šipka, proto musí každá nová šipka vycházet z vrcholu, z něhož ještě žádná nevychází.

Důkaz – pokr.:

Druhý způsob: kořenový strom budeme uvažovat jako orientovaný strom se šipkami směřujícími ke kořeni. Začneme s prázdným grafem a budeme přidávat šipky v $n-1$ krocích.

1. šipka: $n(n-1)$ možností.

další šipka:

- nesmí vytvořit kružnici;
- na konci musí z každého vrcholu kromě jediného vycházet nějaká šipka, proto musí každá nová šipka vycházet z vrcholu, z něhož ještě žádná nevychází.

Důkaz – pokr.:

Druhý způsob: kořenový strom budeme uvažovat jako orientovaný strom se šipkami směřujícími ke kořeni. Začneme s prázdným grafem a budeme přidávat šipky v $n-1$ krocích.

1. šipka: $n(n-1)$ možností.

další šipka:

- nesmí vytvořit kružnici;
- na konci musí z každého vrcholu kromě jediného vycházet nějaká šipka, proto musí každá nová šipka vycházet z vrcholu, z něhož ještě žádná nevychází.

V každé komponentě již vytvořeného grafu je právě jeden vrchol, z něhož nevychází šipka. Šipka číslo $k+1$ musí vést do některého vrcholu a vycházet z kořene některé z ostatních komponent – $n(n-k-1)$ možností.

Důkaz – pokr.:

Druhý způsob: kořenový strom budeme uvažovat jako orientovaný strom se šipkami směřujícími ke kořeni. Začneme s prázdným grafem a budeme přidávat šipky v $n-1$ krocích.

1. šipka: $n(n-1)$ možností.

další šipka:

- nesmí vytvořit kružnici;
- na konci musí z každého vrcholu kromě jediného vycházet nějaká šipka, proto musí každá nová šipka vycházet z vrcholu, z něhož ještě žádná nevychází.

V každé komponentě již vytvořeného grafu je právě jeden vrchol, z něhož nevychází šipka. Šipka číslo $k+1$ musí vést do některého vrcholu a vycházet z kořene některé z ostatních komponent – $n(n-k-1)$ možností. Celkem máme

$$\prod_{k=0}^{n-2} n(n-k-1) = (n-1)! \cdot n^{n-1}$$

způsobů a porovnáním s prvním výpočtem dostaneme tvrzení.

Algoritmus pro nalezení kostry 1

Seřadíme zcela libovolně všechny hrany $e_1, \ldots, e_m$ v E do pořadí a postupně budujeme množiny hran $E_i (i = 0, \ldots, m)$ tak, že $E_0 = \emptyset$ v t-tém kroku přidáme hranu e_i k E_{i-1} (tj. $E_i = E_{i-1} \cup \{e_i\}$), jestliže tím nevznikne v grafu $G_i = (V, E_i)$ kružnice, a ponecháme $E_i = E_{i-1}$ beze změny v případě opačném. Algoritmus skončí pokud buď má již graf G_i pro nějaké i právě $n - 1$ hran nebo je již $i = m$. Pokud zastavujeme z druhého důvodu, byl původní graf nesouvislý a kostra neexistuje.

Algoritmus pro nalezení kostry 1

Seřadíme zcela libovolně všechny hrany $e_1, \ldots, e_m$ v E do pořadí a postupně budujeme množiny hran $E_i (i = 0, \ldots, m)$ tak, že $E_0 = \emptyset$ v t-tém kroku přidáme hranu e_i k E_{i-1} (tj. $E_i = E_{i-1} \cup \{e_i\}$), jestliže tím nevznikne v grafu $G_i = (V, E_i)$ kružnice, a ponecháme $E_i = E_{i-1}$ beze změny v případě opačném. Algoritmus skončí pokud buď má již graf G_i pro nějaké i právě $n - 1$ hran nebo je již $i = m$. Pokud zastavujeme z druhého důvodu, byl původní graf nesouvislý a kostra neexistuje.

Věta

Výsledkem předchozího algoritmu je vždy les T. Jestliže algoritmus skončí s $k \leq n - 1$ hranami, má původní graf $n - k$ komponent. Zejména je tedy T kostrou právě, když algoritmus skončí pro dosažení $n - 1$ hran.

Důkaz.

Tvrzení, že výsledný graf T je lesem, je zřejmé z postupu konstrukce. Je-li $k = n - 1$, je navíc T strom podle charakterizační věty o stromech. Je-li $k < n - 1$, je T lesem, s $n - k$ stromovými komponentami, neboť každá další komponenta přispívá jedničkou k hodnotě $(n - 1) - k$ (rozdíl počtu hran ve stromu a počtu hran v grafu T). □

Poznámka (složitost algoritmu)

Kružnice přidáním nové hrany vznikne tehdy a jen tehdy, jestli její koncové vrcholy leží ve stejné souvislé komponentě budovaného lesu T. Stačí nám proto průběžně udržovat znalost souvislých komponent.

Poznámka (složitost algoritmu)

Kružnice přidáním nové hrany vznikne tehdy a jen tehdy, jestli její koncové vrcholy leží ve stejné souvislé komponentě budovaného lesu T. Stačí nám proto průběžně udržovat znalost souvislých komponent.

V abstraktní podobě nám stačí umět pro již zadané třídy ekvivalence na dané množině (v našem případě jsou to vrcholy) slučovat dvě třídy ekvivalence do jedné a nalézat pro daný prvek, do které třídy patří. Pro sjednocení jistě potřebujeme $O(k)$ času, kde k je počet prvků slučovaných tříd a jistě můžeme použít ohraničení počtu k celkovým počtem vrcholů n. Se třídami si můžeme pamatovat i počty jejich prvků a průběžně pro každý vrchol uchovávat informaci do které třídy patří. Sjednocení dvou tříd tedy představuje přeznačení jména u všech prvků jedné z nich. Máme tedy $n-1$ operací sjednocení a m operací testování ekvivalence vrcholů, proto lze složitost ohraničit $O(n^2+m)$.

Poznámka (složitost algoritmu)

Kružnice přidáním nové hrany vznikne tehdy a jen tehdy, jestli její koncové vrcholy leží ve stejné souvislé komponentě budovaného lesu T. Stačí nám proto průběžně udržovat znalost souvislých komponent.

V abstraktní podobě nám stačí umět pro již zadané třídy ekvivalence na dané množině (v našem případě jsou to vrcholy) slučovat dvě třídy ekvivalence do jedné a nalézat pro daný prvek, do které třídy patří. Pro sjednocení jistě potřebujeme $O(k)$ času, kde k je počet prvků slučovaných tříd a jistě můžeme použít ohraničení počtu k celkovým počtem vrcholů n. Se třídami si můžeme pamatovat i počty jejich prvků a průběžně pro každý vrchol uchovávat informaci do které třídy patří. Sjednocení dvou tříd tedy představuje přeznačení jména u všech prvků jedné z nich. Máme tedy $n-1$ operací sjednocení a m operací testování ekvivalence vrcholů, proto lze složitost ohraničit $O(n^2+m)$. Budeme-li vždy přeznačovat menší ze slučovaných tříd, pak celkový počet operací v našem algoritmu lze ohraničit $O(n \log n + m)$.

Algoritmus pro nalezení kostry 2

Jiný postup: Budeme v grafu $G = (V, E)$ s n vrcholy a m hranami postupně budovat strom T. Začneme v libovolně zvoleném vrcholu v a prázdnou množinou hran, tj. $T_0 = (\{v\}, \emptyset)$. V i–tém kroku hledáme mezi hranami, které dosud nejsou v T_{i-1}, mají v T_{i-1} jeden koncový vrchol, ale druhý koncový vrchol do T_{i-1} nepatří. První takovou hranu přidáme i s druhým koncovým vrcholem a získáme tak T_i. Algoritmus skončí, až taková hrana neexistuje.

Algoritmus pro nalezení kostry 2

Jiný postup: Budeme v grafu $G = (V, E)$ s n vrcholy a m hranami postupně budovat strom T. Začneme v libovolně zvoleném vrcholu v a prázdnou množinou hran, tj. $T_0 = (\{v\}, \emptyset)$. V i–tém kroku hledáme mezi hranami, které dosud nejsou v T_{i-1}, mají v T_{i-1} jeden koncový vrchol, ale druhý koncový vrchol do T_{i-1} nepatří. První takovou hranu přidáme i s druhým koncovým vrcholem a získáme tak T_i. Algoritmus skončí, až taková hrana neexistuje.

Evidentně je výsledný graf T (v případě, že má n vrcholů) souvislý a podle počtu vrcholů a hran je to strom. Vrcholy T splývají s vrcholy souvislé komponenty G.

Algoritmus pro nalezení kostry 2

Jiný postup: Budeme v grafu $G = (V, E)$ s n vrcholy a m hranami postupně budovat strom T. Začneme v libovolně zvoleném vrcholu v a prázdnou množinou hran, tj. $T_0 = (\{v\}, \emptyset)$. V i–tém kroku hledáme mezi hranami, které dosud nejsou v T_{i-1}, mají v T_{i-1} jeden koncový vrchol, ale druhý koncový vrchol do T_{i-1} nepatří. První takovou hranu přidáme i s druhým koncovým vrcholem a získáme tak T_i. Algoritmus skončí, až taková hrana neexistuje.

Evidentně je výsledný graf T (v případě, že má n vrcholů) souvislý a podle počtu vrcholů a hran je to strom. Vrcholy T splývají s vrcholy souvislé komponenty G.

Algoritmus v čase $O(n + m)$ nalezne kostru souvislé komponenty zvoleného počátečního vrcholu v.

Plán přednášky

Kromě nalezení kostry je často účelné znát nejlepší možnou kostru vzhledem k nějakému ohodnocení hran. Protože je to obecnou vlastností stromů, každá kostra grafu G má stejný počet hran. V grafech s ohodnocenými hranami, budeme hledat kostry s minimálním součtem ohodnocení použitých hran.

Definice

Nechť $G = (V, E, w)$ je souvislý graf s ohodnocenými hranami s nezápornými vahami $w(e)$ pro všechny hrany. Jeho **minimální kostra** (*minimum spanning tree*) T je taková kostra grafu G, která má mezi všemi jeho kostrami minimální součet ohodnocení všech hran.

Kromě nalezení kostry je často účelné znát nejlepší možnou kostru vzhledem k nějakému ohodnocení hran. Protože je to obecnou vlastností stromů, každá kostra grafu G má stejný počet hran. V grafech s ohodnocenými hranami, budeme hledat kostry s minimálním součtem ohodnocení použitých hran.

Definice

Nechť $G = (V, E, w)$ je souvislý graf s ohodnocenými hranami s nezápornými vahami $w(e)$ pro všechny hrany. Jeho **minimální kostra** (*minimum spanning tree*) T je taková kostra grafu G, která má mezi všemi jeho kostrami minimální součet ohodnocení všech hran.

O praktičnosti takové úlohy můžete přemýšlet třeba v souvislosti s rozvodnými sítěmi elektřiny, plynu, vody apod (viz např. problém elektrifikace části jižní Moravy, který vyřešil Otakar Borůvka v roce 1926 pomocí algoritmu – v dnešní terminologii – minimální kostry, přestože obor teorie grafů, čekal ještě cca 10 let na svůj oficiální vznik).

Kruskalův algoritmus

Předpokládejme, že jsou všechna ohodnocení $w(e)$ hran v grafu G nezáporná. Následujícímu postupu se říká **Kruskalův algoritmus**:

- Setřídíme všech m hran v E tak, aby $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m)$.
- v tomto pořadí aplikujeme na hrany postup z Algoritmu 1 pro kostru.

Kruskalův algoritmus

Předpokládejme, že jsou všechna ohodnocení $w(e)$ hran v grafu G nezáporná. Následujícímu postupu se říká **Kruskalův algoritmus**:

- Setřídíme všech m hran v E tak, aby $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m)$.
- v tomto pořadí aplikujeme na hrany postup z Algoritmu 1 pro kostru.

Jde o typický příklad takzvaného „hladového (greedy) přístupu“, kdy se k maximalizaci zisku (nebo minimalizaci nákladů) snažíme dostat výběrem momentálně nejvýhodnějšího kroku. Často tento přístup zklame, protože nizké náklady na začátku procesu mohou zavinit vysoké na jeho konci.
V tomto případě (naštěstí) hladový přístup funguje, nemusíme tedy prohledávat a porovnávat až n^{n-2} koster na daném grafu.

Věta

Kruskalův algoritmus správně řeší problém minimální kostry pro každý souvislý graf G s nezáporným ohodnocením hran. Algoritmus pracuje v čase $O(m \log m)$, kde m je počet hran v G.

Věta

Kruskalův algoritmus správně řeší problém minimální kostry pro každý souvislý graf G s nezáporným ohodnocením hran. Algoritmus pracuje v čase $O(m \log m)$, kde m je počet hran v G.

Důkaz.

Buď T výsledná kostra z algoritmu, T_0 taková minimální kostra na G, která se s T shoduje na co nejvíce (seřazených) hranách.
Sporem dokážeme, že $T_0 = T$.

Věta

Kruskalův algoritmus správně řeší problém minimální kostry pro každý souvislý graf G s nezáporným ohodnocením hran. Algoritmus pracuje v čase $O(m \log m)$, kde m je počet hran v G.

Důkaz.

Buď T výsledná kostra z algoritmu, T_0 taková minimální kostra na G, která se s T shoduje na co nejvíce (seřazených) hranách.
Sporem dokážeme, že $T_0 = T$.
Předpokládejme $T_0 \neq T$ a buď j nejmenší index, takový, že se T a T_0 liší v hraně e_j (zřejmě $e_j \in T \setminus T_0$).

Věta

Kruskalův algoritmus správně řeší problém minimální kostry pro každý souvislý graf G s nezáporným ohodnocením hran. Algoritmus pracuje v čase $O(m \log m)$, kde m je počet hran v G.

Důkaz.

Buď T výsledná kostra z algoritmu, T_0 taková minimální kostra na G, která se s T shoduje na co nejvíce (seřazených) hranách.
Sporem dokážeme, že $T_0 = T$.
Předpokládejme $T_0 \neq T$ a buď j nejmenší index, takový, že se T a T_0 liší v hraně e_j (zřejmě $e_j \in T \setminus T_0$). Pak $T_0 \cup \{e_j\}$ obsahuje právě jednu kružnici C.

Věta

Kruskalův algoritmus správně řeší problém minimální kostry pro každý souvislý graf G s nezáporným ohodnocením hran. Algoritmus pracuje v čase $O(m \log m)$, kde m je počet hran v G.

Důkaz.

Buď T výsledná kostra z algoritmu, T_0 taková minimální kostra na G, která se s T shoduje na co nejvíce (seřazených) hranách.
Sporem dokážeme, že $T_0 = T$.
Předpokládejme $T_0 \neq T$ a buď j nejmenší index, takový, že se T a T_0 liší v hraně e_j (zřejmě $e_j \in T \setminus T_0$). Pak $T_0 \cup \{e_j\}$ obsahuje právě jednu kružnici C. Na této kružnici tedy existuje hrana $e_k (k > j)$, která není v T. Pak ale $w(e_k) \geq w(e_j)$ a kostra s hranami $T_0 \setminus \{e_k\} \cup \{e_j\}$ není horší než T_0 a protože se od T liší „později“, měli jsme ji na začátku vybrat místo T_0. Spor. □

I druhý z našich algoritmů pro kostru grafu v předchozím odstavci vede na minimální kostru, když v každém okamžiku volíme ze všech možných hran $e_i = \{v_i, v_{i1+}\}$, $v_i \in V_i$, $v_{i+1} \in V \setminus v_i$ tu, která má minimální ohodnocení.

I druhý z našich algoritmů pro kostru grafu v předchozím odstavci vede na minimální kostru, když v každém okamžiku volíme ze všech možných hran $e_i = \{v_i, v_{i1+}\}$, $v_i \in V_i$, $v_{i+1} \in V \setminus v_i$ tu, která má minimální ohodnocení.
Výsledný postup je **Primův algoritmus** z roku 1957. Byl ale popsán Jarníkem již v roce 1930. Říkáme mu tedy **Jarníkův algoritmus**. Jarník vycházel z algoritmu O. Borůvky z r. 1926.

Věta

Jarníkův algoritmus najde minimální kostru pro každý souvislý graf s libovolným ohodnocením hran.

I druhý z našich algoritmů pro kostru grafu v předchozím odstavci vede na minimální kostru, když v každém okamžiku volíme ze všech možných hran $e_i = \{v_i, v_{i1+}\}$, $v_i \in V_i$, $v_{i+1} \in V \setminus v_i$ tu, která má minimální ohodnocení.
Výsledný postup je **Primův algoritmus** z roku 1957. Byl ale popsán Jarníkem již v roce 1930. Říkáme mu tedy **Jarníkův algoritmus**. Jarník vycházel z algoritmu O. Borůvky z r. 1926.

Věta

Jarníkův algoritmus najde minimální kostru pro každý souvislý graf s libovolným ohodnocením hran.

Poznámka

Borůvkův algoritmus tvoří stále co nejvíce souvislých komponent zaráz. Začneme s jednoprvkovými komponentami v grafu $T_0 = (V, \emptyset)$ a pak postupně každou komponentu propojíme nejkratší možnou hranou s komponentou jinou. Opět takto obdržíme minimální kostru. Tento algoritmu je základem nejrychlejšího známého algoritmu, běžícího v čase $O(n + m)$.

Příklad

Určete pomocí uvedených algoritmů minimální kostru grafu