Drsná matematika III — 10. přednáška
Stromy a kostry

Jan Slovák

Masarykova univerzita
Fakulta informatiky

21. 11. 2011

Obsah přednášky

1. Literatura
2. Užití stromů
3. Izomorfismy stromů
4. Kostra grafu
5. Minimální kostra
6. Obchodní cestující

Kde je dobré číst?

- 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/
- Bill Cherowitzo, Applied Graph Therory, Lecture Notes, http://www-math.cudenver.edu/~wcherowi/courses/m4408/m4408f.html

Stromy využíváme pro organizaci dat tak, abychom v datech uměli buď rychle vyhledávat nebo v nich udržovat pořádek, nejčastěji obojí.
Ve stromu není žádná kružnice, proto volba jednoho vrcholu v_r zadává orientaci všech hran.
Po výběru jednoho vrcholu – **kořenu** – se začíná graf více podobat skutečnému stromu v přírodě. Stromy s jedním vybraným „kořenem“ nazýváme **kořenové stromy**.

Definition

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).

Definition

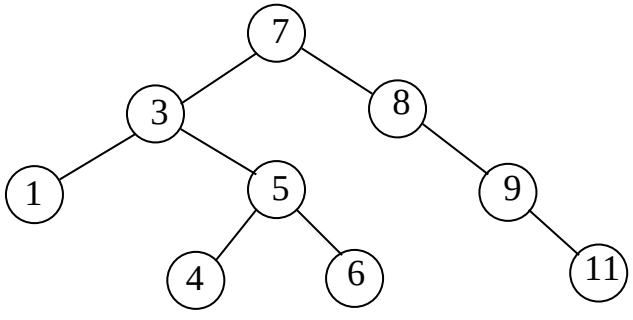
Binární stromy jsou speciálním případem kořenového stromu, kdy každý otec má nejvýše dva následníky (někdy se ale pod stejným označením binární strom předpokládá, že všechny vrcholy kromě listů mají právě dva následníky).

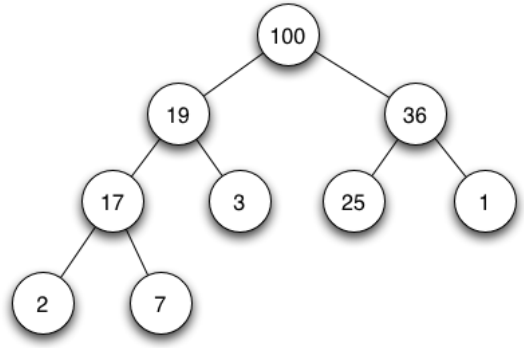
Často jsou vrcholy stromu spojeny s klíči v nějaké úplně uspořádané množině (např. reálná čísla) a slouží k hledání vrcholu s daným klíčem.
Je realizováno jako hledání cesty od kořene stromu a v každém vrcholu se podle velikosti rozhodujeme, do kterého ze synů budeme pokračovat (resp. zastavíme hledání, 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.

Pro efektivní vyhledávání užíváme **vyvážené binární stromy**, ve kterých se délky cest z kořene do listů liší maximálně o jedničku. Nejdále od vyváženého stromu na n vrcholech je tedy cesta P_n (která formálně může být považována za binární strom), zatímco dokonale vyvážený strom, kde kromě listů má každý otec právě dva syny je možné sestrojit pouze pro hodnoty $n = 2^k - 1$, $k = 1, 2, \ldots$. Ve vyvážených stromech dohledání vrcholu podle klíče bude vždy vyžadovat pouze $O(\log_2 n)$ kroků. Hovoříme v této souvislosti také často o **binárních vyhledávacích stromech**.

Jako cvičení si rozvažte, jak lze účinně vykonávat základní operace s grafy (přidávání a odebírání vrcholů se zadanými klíči, včetně vyvážení) nad binárními vyhledávacími stromy.

Příkladem využití struktury binárních stromů je datová struktura halda. Jde opět o vyvážené binární stromy s vrcholy opatřenými klíči a požadujeme aby podél všech cest od kořene k listům ve stromu klíče klesaly (tzv. maximální halda) nebo naopak stoupaly (tzv. minimální halda).





Na obrázku nalevo je binární vyhledávací strom (který ale není vyvážený), napravo je příklad maximální haldy.
Díky tomuto uspořádání umíme v konstatním čase odebírat z haldy podmnožiny buď maximálních nebo minimálních prvků a skutečné náklady na takovou operaci spočívají v obnovení struktury haldy po odebrání kořene. (Jako cvičení si ukažte, že je to možné zvládnout v logaritmickém čase.)

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.).

Definition

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, kterým takto 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 (třeba si můžeme představit že dolů malujeme levý obrubník cesty a nahoru pravý) získáme skutečně původní strom s jednou hranou směřující shora do kořene navíc.

Theorem

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

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 a postupovat obdobně s využitím lexikografického uspořádní synů podle jejich kódů. Kořenový strom budeme tedy popisovat zřetězováním částí slov tak, že má-li otec v syny již označeny kódy $W_1, \dots W_\ell$, pak pro otce použijeme slovo

$$0W_1 \dots W_\ell 1$$

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

Pokud není určen kořen ve stromě, můžeme se jej určit tak, aby byl „přibližně uprostřed stromu“.
Všechny jednotlivé vrcholy stromu označíme hodnotou tzv. **výstřednosti**, kterou definujeme pro každý vrcholu v v grafu T jako největší možnou vzdálenost z v do nějakého vrcholu w v T, kterou lze dosáhnout.
U stromu díky nepřítomnosti kružnic platí, že maximální hodnoty excentricity vždy dosahuje buď právě jeden vrchol nebo právě dva vrcholy. Skutečně, nejdelší možná cesta z kteréhokoliv vrcholu stromu nutně končí v některém z jeho listů.

Libovolnému stromu přiřadíme jednoznačný kód, až na izomorfismus takto: Pokud existuje jediný vrchol s maximální excentricitou, použijeme jej jako kořene, v opačném případě postupujeme stejně pro dva stromy vzniklé z T odebráním hrany spojující vrcholy s maximální excentricitou a kód vznikne zřetězením kódů obou stromů v pořadí podle lexikografického uspořádání.

Theorem

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

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 podgraf T na všech vrcholech grafu G, který je stromem.

Definition

Libovolný strom $T = (V, E')$ v grafu $G = (V, E)$, $E' \subset E$ se nazývá **kostra** grafu G.

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.

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 tak, že v $(i+1)$–vém kroku přidáme hranu e_i k E'_i jestliže tím nevznikne v grafu $G_i = (V, E_i \cup \{e_i\})$ kružnice, a ponecháme E_i 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.

Theorem

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.

Poznámka

Jako vždy bychom se měli zabývat otázkou, jak složitý je uvedený algoritmus. 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 tříd 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 můžme 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. Budeme vždy přeznačovat tu menší z nich a pak celkový počet operací potřebných v našem algoritmu bude $O(n \log n + m)$.

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 hedáme mezi hranami, které dosud nejsou v T_{i-1}, mají v T_{i-1} jeden koncový vrchol, ale druhý koncový vrchol fo 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 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.

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.

Definition

Nechť $G = (V, E, w)$ je souvislý graf s ohodnocenými hranami s nezápornými vahami $w(e)$ pro všechny hrany. Jeho **minimální kostra** 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.

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 „hladoveckého přístupu“, kdy se k maximalizaci zisku (nebo minimalizaci nákladů) snažíme dostat výběrem momentálně (snad) nejvýhodnějšího kroku. Často tento přístup zklame, protože nízké náklady na začátku procesu mohou zavinit vysoké na jeho konci.

Theorem

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.

Spočívá v analýze běhu Algoritmu 1 ve vztahu k velikosti 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**, je z r. 1957. Byl ale popsán Jarníkem již v roce 1930. Říkáme mu tedy **Jarníkův algoritmus**. Jarník vycházel z algoritmu O. Borůvky z r. 1928.

Theorem

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.

Dosud jsme vybírali pouze jednoduché grafové problémy. V drtivé většině případů je tomu naopak a teoretické výsledky pouze ukazují, že algoritmus fungující alespoň v polynomiálním čase neexistuje. Velice studovaná je úloha, kdy máme najít v grafu s ohodnocenými hranami minimální hamiltonovskou kružnici, tzn. kružnici s minimálním součtem vah použitých hran mezi všemi možnými hamiltonovskými kružnicemi.

S problémem se setkáváme např. při

- plánování dodávek zboží nebo služeb
- organizaci poštovní služby (rozvoz pošty, výběr pošty ze schránek)
- plánování údržby sítí (např. bankomatů)
- obsluha požadavků z fronty (např. při paralelních požadavcích na čtení z hard disku)
- plánování postupného měření jednotlivých částí celku (např. při studiu struktury krystalu proteinu pomocí X-ray, kdy náklady jsou soustředěny zejména na posuvy a zaostření pro jednotlivá měření)
- plánování dělení materiálů (např. při kladení tapet jejich dělení na použité pásy tak, aby navazoval vzorek a došlo k co nejmenším ztrátám)

Theorem

Polynomiálně složitý algoritmus pro problém obchodního cestujícího neexistuje.

I v případě hledání minimální hamiltonovské kružnice můžeme uplatnit hladovecký (anglicky „greedy“) přístup.
Algoritmus začne v libovolném vrcholu v_1, postupně v krocích najde ten dosud neumístěný vrchol ze spících, do kterého vede z aktivního vrcholu nejméně ohodnocená hrana, aktivní vrchol označí jako zpracovaný, tento nový vrchol se stane aktivním. Algoritmus skončí buď neúspěchem nebo získáme hamiltonovskou kružnici.
Je zjevné, že tento algoritmus jen velice zřídka vyprodukuje skutečně minimální hamiltonovskou kružnici. Na úplném grafu zato vždy alespoň nějakou najde. Je dokázáno, že se dokonce polynomiálně rychlými algoritmy nelze ani libovolně přibližovat k optimálnímu řešení.