UMB/750 Diskrétní matematika

9. přednáška

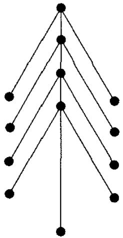
29. listopadu 2024

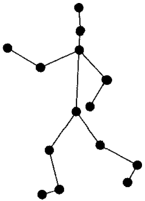
Stromy

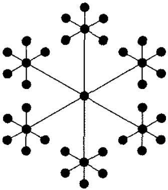
Strom je jeden z nejpřirozenějších útvarů jak v přírodě, tak v matematice. Stromy v přírodě jsou košaté, rozmanité a nesmírně složité.

Stromy v teorii grafů jsou téměř nejjednodušší ze všech grafů, ale přesto jejich studium tvoří bohatou a zajímavou oblast.

V teorii grafů se stromem rozumí konečný graf, který vypadá podobně jako na následujících obrázcích:







Jak tento pojem vystihnout?

Stromy

Definice

Strom je souvislý graf neobsahující kružnici. Les je graf, jehož komponenty jsou stromy.

$\rightarrow$ jak jednoduše zkontrolovat, že daný graf je strom?

souvislost ověříme snadno, s neexistencí kružnic je to však horší

Lemma

Každý strom s alespoň dvěma vrcholy obsahuje alespoň dva vrcholy stupně 1.
Vrchol stupně 1 *se nazývá koncový vrchol nebo list.*

Poznámka: Toto lemma neplatí pro nekonečné stromy.

značení: $T(V, E)$

Výstavba stromů

Značení: Je-li G graf a v jeho vrchol, potom $G - v$ označuje graf, který vznikne z G vynecháním v a všech hran, které vrchol v obsahují.

Pokud je v koncový vrchol (list) stromu T, vznikne $T - v$ odebráním v a jediné hrany jej obsahující.

Tvrzení

Pro daný graf G a jeho koncový vrchol v jsou následující dvě tvrzení ekvivalentní:

(i) *G je strom.*

(ii) *$G - v$ je strom.*

Toto tvrzení umožňuje postupně převádět daný strom na menší a menší stromy.

Ukazuje, že graf G je strom právě když ho lze převést na jeden vrchol postupným odebíráním koncových vrcholů (lze odebrat libovolný koncový vrchol).

Charakterizace stromů

Tvrzení

Pro graf $G = (V, E)$ jsou následující podmínky ekvivalentní:

(i) *G je strom.*

(ii) *(jednoznačnost cesty)*
Pro každé dva vrcholy $x, y \in V$ existuje právě jediná cesta z x do y.

(iii) *(minimální souvislost)*
Graf G je souvislý, a vynecháním libovolné hrany vznikne nesouvislý graf.

(iv) *(maximální graf bez kružnic)*
Graf G neobsahuje kružnici, a každý graf vzniklý z G přidáním hrany (tj. graf tvaru $G + e$, kde $e \in \binom{V}{2} \setminus E$) již kružnici obsahuje.

(v) *(Eulerův vzorec)*
G je souvislý a $|V| = |E| + 1$.

Cvičení

Kontrolní otázky:

1. Jak se jmenuje strom, který má n vrcholů, z nichž právě dva jsou listy?
2. Kolik musíme z grafu K_n vynechat hran, abychom dostali strom?

Příklad. Ukažte, že známe-li stupně $\deg(v_1), \deg(v_2), \ldots, \deg(v_k)$ všech nelistových vrcholů v nějakém stromu T, umíme také určit počet listů ve stromu T.

Isomorfismus stromů

Připomeňme: $f : V(T) \to V(T')$ je isomorfismus stromů T a T', jestliže f je bijekce splňující $\{x, y\} \in E(T)$, právě když $\{f(x), f(y)\} \in E(T')$, zapisujeme $T \cong T'$.

Pro úlohu rozhodnout, zda jsou dva obecné grafy isomorfní, není znám žádný rychlý algoritmus (s polynomiální složitostí). Pro stromy však takový algoritmus existuje!

Uvedeme postup, který rozhodne, zda T a T' jsou isomorfní nebo ne.

Každému stromu T na n vrcholech přiřadíme posloupnost 0 a 1 délky $2n$, zvanou *kód* stromu T.

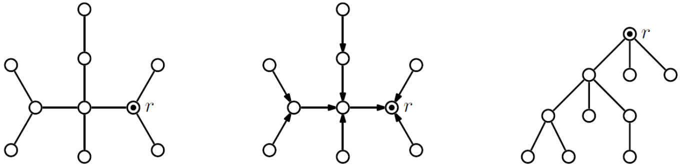
$\to$ isomorfní stromy budou mít stejný kód, neisomorfní různé kódy

Kořenové stromy

Definice

Kořenový strom je dvojice (T, r), kde $r \in V(T)$ je jeden zvolený vrchol T, zvaný *kořen*.

Je-li $\{x, y\} \in E(T)$ hrana, a leží-li x na (jediné) cestě z y do kořene, říkáme, že x je *rodič* y a y je *potomek* x.



Definice

Pěstovaný strom je nějaký kořenový strom (T, r), plus jeho pevně zvolené rovinné nakreslení.

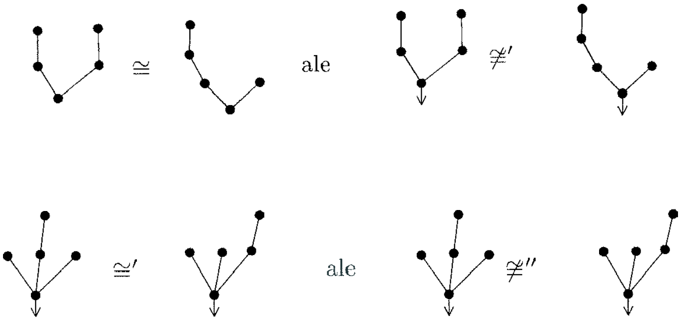
(pěstovaný strom je tedy kořenový strom, kde pro každý vrchol v je dáno pořadí jeho potomků)

Isomorfismus stromů

Isomorfismus kořenových stromů (T, r) a (T', r') je takový isomorfismus f stromů T a T', pro nějž navíc $f(r) = r'$. Značíme $(T, r) \cong' (T', r')$.

Isomorfismus pěstovaných stromů zachovává navíc uspořádání potomků každého vrcholu, $(T, r, v) \cong'' (T', r', v')$.

(definice $\cong$, $\cong'$ a $\cong''$ se postupně zesilují)



(kořen je vyznačen šipkou směřující dolů)

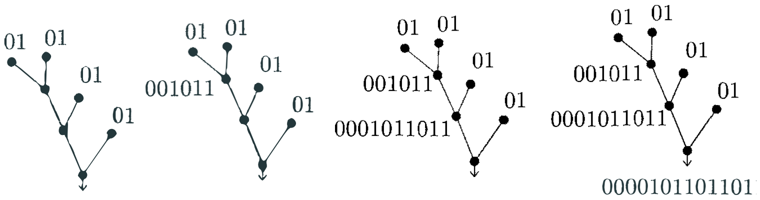
Kódování pěstovaných stromů

Definice pěstovaného stromu (a jeho isomorfismus) je nejvíce omezující, a proto je kódování těchto stromů nejsnazší.

Následující metoda přiřadí jistý kód každému vrcholu pěstovaného stromu; kód celého stromu bude pak totožný s kódem kořene.

Kl. Koncovým vrcholům (různým od kořene) přiřadíme 01.

K2. Buď v nějaký vrchol, $v_1, \ldots, v_t$ jeho potomci v pořadí zleva doprava. Má-li potomek v_i kód A_i, potom vrcholu v přiřadíme kód $0A_1A_2\ldots A_t1$.



Obrázek 5.2: Kódování pěstovaného stromu.

Kódování pěstovaných stromů

Jak z kódu jednoznačně *rekonstruovat* pěstovaný strom? (tím ukážeme, že neisomorfním pěstovaným stromům je přiřazen různý kód)

Indukcí podle délky kódové posloupnosti:

Nejkratší možný kód 01 odpovídá jedinému pěstovanému stromu s jedním vrcholem.

V indukčním kroku nechť je dán kód k délky $2(n+1)$. Tento kód je tvaru $0A1$, přičemž $A = A_1A_2\dots A_t$ je zřetězení kódů několika pěstovaných stromů. Část A_1 identifikujeme jako nejkratší počáteční úsek posloupnosti A, který obsahuje stejný počet nul a jedniček. Podobně A_2 je nejkratší následující úsek posloupnosti obsahující stejně nul a jedniček, atd.

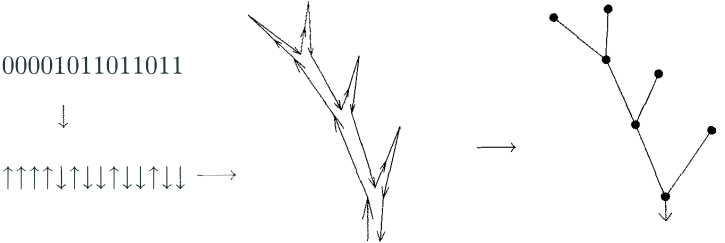
Podle indukčního předpokladu každému A_i odpovídá právě jediný pěstovaný strom.

Pěstovaný strom kódovaný kódem k bude mít jeden kořen r, a k němu budou v pořadí $r_1, \dots, r_t$ připojeny hranou kořeny stromů kódovaných $A_1, A_2, \dots$. Tedy kód určuje jednoznačně pěstovaný strom.

Dekódování metodou šipek

Názorný – obrázkový – postup, jak pěstovaný strom z daného kódu zrekonstruovat:

V daném kódu nahradíme 0 šipkou “nahoru” a 1 šipkou “dolů” a kreslíme graf podle pravidla, že šipka nahoru nikdy nevede po jiné šipce, zatímco šipka dolů vždy sleduje šipku v protisměru. Celý postup je patrný z obrázku.



Obrázek 5.3: Dekódování metodou šipek.

Isomorfismus stromů

Isomorfismus kořenových i obecných stromů se dá převést na předchozí případ.

$\rightarrow$ musíme umět jednoznačně zvolit kořen a musíme umět jednoznačně seřadit potomky každého vrcholu

Centrem (středem) stromu rozumíme vrchol nebo hranu v daném stromu T, které je určeno následujícím postupem:

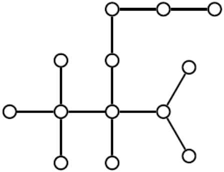
1. Jestliže má strom T jediný vrchol v, je v centrum stromu T. Pokud má strom T dva vrcholy, je centrem stromu hrana spojující tyto dva vrcholy. Jinak přejdeme k bodu 2.
2. Vytvoříme (menší) strom $T' \subset T$ oholením všech listů stromu T a vracíme se na předchozí bod 1.

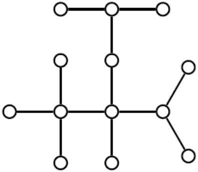
Rekurzivním postupem získané centrum stromu T' bude zároveň centrem stromu T.

Poznámka: Obecně se střed grafu G zavádí jako množina vrcholů, jejichž výstřednost nabývá (excentricita) nabývá nejmenší hodnoty. Excentricitou vrcholu v přitom rozumíme maximální vzdálenost vrcholu v od jiného vrcholu grafu G. Střed grafu může splývat s množinou všech vrcholů (např. kružnice). Pro stromy však platí, že střed je tvořen nejvýše dvěma vrcholy. Je-li tvořen dvěma vrcholy x a y, potom $\{x, y\}$ tvoří hranu.

Příklad

Určete centrum následujících stromů:





Poznámka o minimálním kódu stromu

Kódy můžeme chápat jako řetězce a ty umíme seřadit jednoznačně, například *lexikograficky* ("slovníkové" uspořádání).

$\rightarrow$ pro každou dvojici kódů umíme rozhodnout, který kód by byl ve slovníku napsán dříve

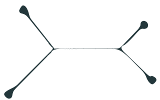
například řetězec 000111 by byl ve slovníku před řetězci 001011, 0011 i 01

Jestliže při sestavování kódu kořenového stromu pro každý vrchol nejprve seřadíme kódy jeho potomků lexikograficky, dostaneme tzv. *minimální kód*.

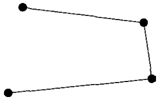
Problém minimální kostry

Představme si mapu jižní Moravy. Máme na ní vyznačeno 30–40 měst a vesniček. Naším úkolem bude tato města navzájem propojit (třeba elektrickým vedením) tak, aby délka spojení nebyla příliš velká. Řekněme dokonce, že bychom chtěli, aby celková délka propojení byla co nejkratší.

Navíc se rozhodneme, že naše síť se nebude větvit nikde mimo spojované obce, tj. nebudou v ní situace typu



takže tato 4 místa by se musela propojit takto:



Vynecháme-li právě zavedené omezení, dostaneme problém minimálního Steinerova stromu. To je zcela jiná úloha, pro niž asi neexistuje efektivní algoritmus, v praxi se řeší pouze přibližně.

– další ze základních problémů teorie grafů, který hraje důležitou roli v řadě praktických aplikací při hledání optimálního řešení

– objevuje se tak jako část jiných algoritmů, např. v Christofidově algoritmu pro heuristické řešení úlohy obchodního cestujícího se nejprve hledá minimální kostra daného grafu

Budeme uvažovat grafy $G = (V, E)$ spolu s *ohodnocenými hranami*, tj. pro každou hranu $e \in E$ je dáno číslo $w(e)$, nazývané *váha* hrany e (váha je zpravidla kladné číslo.)

Graf spolu s ohodnocením $w : E \to \mathbb{R}$, se nazývá *síť*.

Formulace problému

Pro souvislý graf $G = (V, E)$ s nezáporným ohodnocením hran w nalezněte souvislý podgraf (V, E') takový, že výraz

$$w(E') = \sum_{e \in E'} w(e)$$

nabývá minimální hodnoty.

Mezi řešeními této úlohy je vždy nějaký strom (je-li ohodnocení všech hran kladné, je řešením jenom strom).

Například, jestliže $w(e) = 1$ pro každou hranu e, úlohu řeší libovolný podgraf obsahující všechny vrcholy a který je strom (tedy má $|V| - 1$ hran). Pro takové podgrafy se zavádí zvláštní název:

Definice

Nechť $G = (V, E)$ je graf. Libovolný strom tvaru (V, E'), kde $E' \subset E$, nazveme *kostra* grafu G.

Problém minimální kostry

Pro souvislý graf $G = (V, E)$ s nezáporným ohodnocením hran w nalezněte kostru $T = (V, E')$ grafu G s nejmenší možnou hodnotou $w(E')$.

Kostra existuje, pouze je-li graf G souvislý.

Každý souvislý graf má kostru.

Daný graf může mít velmi mnoho koster. Nalézt tu nejlepší z nich není obtížné.

Kruskalův neboli "hladový" algoritmus

Je dán souvislý graf $G = (V, E)$ s n vrcholy, m hranami a s ohodnocením w. Očíslujme hrany $e_1, e_2, \dots, e_m$ tak, aby

$$w(e_1) \le w(e_2) \le \dots \le w(e_m).$$

(Tento krok tedy vyžaduje setřídění vah.)

Nyní budeme postupně konstruovat množiny hran $E_0, E_1, E_2, \dots, E_m \subset E$.

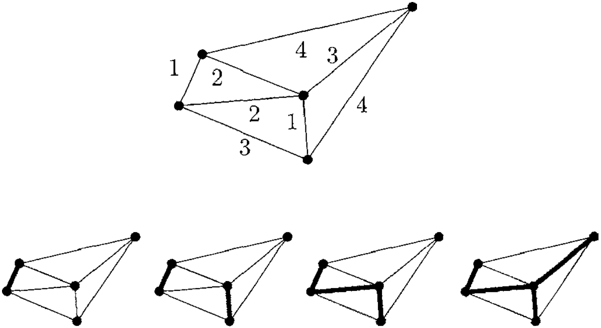
Položme $E_0 = \emptyset$. Byla-li již nalezena množina E_{i-1}, spočítáme množinu E_i, následovně:

$$E_i = \begin{cases} E_{i-1} \cup \{e_i\} & \text{neobsahuje-li graf } (V, E_{i-1} \cup \{e_i\}) \text{ kružnici} \\ E_{i-1} & \text{jinak.} \end{cases}$$

Algoritmus se zastaví po m-tém kroku.

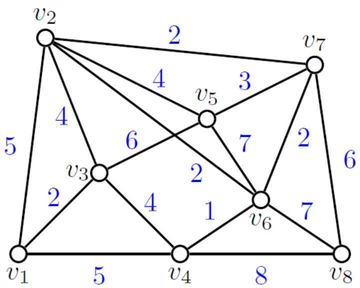
Kostra $T = (V, E_m)$.

Příklad



Cvičení

Najděte minimální kostru grafu:



Borůvkův algoritmus

historicky první postup pro řešení problému minimální kostry
(O. Borůvka, 1928 - příklad z jižní Moravy)

Je dán graf $G = (V, E)$ s ohodnocením hran w. Předpokládáme navíc, že *různým hranám jsou přiřazena různá čísla*, tj. funkce w je prostá.

Algoritmus postupně vytváří množiny hran $E_0, E_1, \dots \subseteq E$, $E_0 = \emptyset$.

Předpokládejme, že jsme již spočítali množinu E_{i-1} a nechť $(V_1, V_2, \dots, V_t)$ je rozklad množiny V podle komponent souvislosti grafu (V, E_{i-1}).

Pro každou třídu V_j tohoto rozkladu vyhledáme hranu $e_j = \{x_j, y_j\}$ (kde $x_j \in V_j$, $y_j \notin V_j$), jejíž váha je minimální mezi hranami tvaru $\{x, y\}$, $x \in V_j$, $y \in V \setminus V_j$ (přitom se může stát $e_j = e_{j'}$ pro $j \neq j'$).

Definujeme $E_i = E_{i-1} \cup \{e_1, \dots, e_t\}$.

Algoritmus končí, má-li graf (V, E_i) jedinou komponentu.

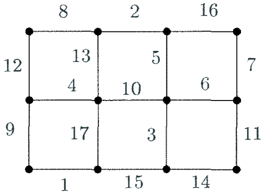
Jarníkův (Primův) algoritmus

Je dán souvislý graf $G = (V, E)$ s n vrcholy, m hranami a nezáporným ohodnocením hran w.

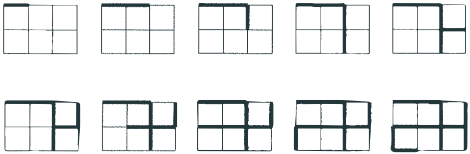
Budeme postupně vytvářet množiny vrcholů $V_0, V_1, \dots \subseteq V$ a množiny hran $E_0, E_1, \dots \subseteq E$, přičemž $E_0 = \emptyset$ a $V_0 = \{v\}$ je libovolně zvolený vrchol.

Jsou-li již V_{i-1} a E_{i-1} vytvořeny, uvážíme množinu F_i všech hran $\{x_i, y_i\} \in E(G)$, kde $x_i \in V_{i-1}$ a $y_i \in V \setminus V_{i-1}$. Pokud $F_i = \emptyset$, algoritmus končí (grafem $(V_t, E_t) = T$). Jinak zvolíme hranu $e_i \in F_i$, jejíž váha je nejmenší mezi všemi hranami F_i, a položíme $V_i = V_{i-1} \cup \{y_i\}$, $E_i = E_{i-1} \cup \{e_i\}$.

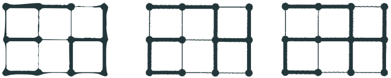
Uvažme následující síť (ohodnocení jsou připsána k příslušným hranám):



Jarníkův algoritmus použitý na tuto síť proběhne takto (začínáme v levém horním rohu):



Kruskalův algoritmus by probíhal v 17 krocích (ale jenom v 10 z nich by přibyly hrany kostry). Borůvkův algoritmus je oproti tomu krátký:



V každém kroku však musíme vykonat více práce.