

PB165 – Grafy a sítě

Kostry grafu

Obsah přednášky

- 1 Úvod
- 2 Budování stromu v grafu
- 3 Průchody grafem
 - Průchod do šířky
 - Průchod do hloubky
- 4 Minimální kostra grafu
 - Primův algoritmus
 - Kruskalův algoritmus

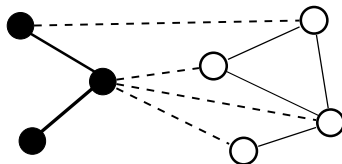
Terminologický úvod

Definice

Stromem v grafu G rozumíme podstrom grafu G , který je stromem. Hrany a vrcholy, které do tohoto stromu náležejí, nazýváme stromové. V opačném případě se hrana zve nestromová.

Hranu, jejíž jeden krajní vrchol je součástí stromu T v neorientovaném grafu, budeme značit jako okrajovou hranu stromu T . Je-li graf orientovaný, značíme hranu jako okrajovou pokud je součástí stromu T její počáteční vrchol.

Obrázek: Strom v grafu je vyznačen plnými vrcholy a tučnými hranami, okrajové hrany čárkovaně.



Růst stromu v grafu

Věta

Je-li G graf a T strom v G , potom graf vzniklý z T přidáním jeho libovolné okrajové hrany je také stromem.

Důkaz.

Jelikož hrana má alespoň jeden ze svých koncových vrcholů v T , existuje cesta z přidaného vrcholu do všech vrcholů T a graf zůstává spojitý.

Přidaná hrana má mezi vrcholy stromu T zároveň nejvýše jeden vrchol. Nemůže tedy žádným způsobem vzniknout cyklus a graf zůstává i acyklický, tedy strom. □

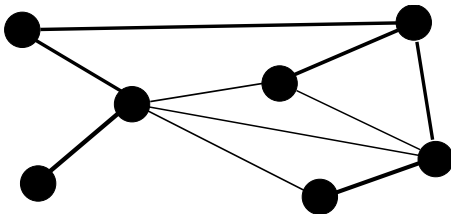
Kostra grafu

Definice

Kostra grafu G je takový strom T v grafu G , pro který platí $V(T) = V(G)$.

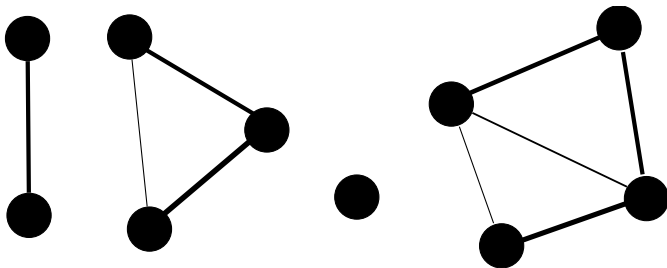
- Graf může mít více než jednu kostru.
- Každý acyklický podgraf grafu G je obsažen v alespoň jedné kostře grafu G .

Obrázek: Kostra je vyznačena tučnými hranami.



Kostra komponent grafu

Graf může mít kostru zřejmě jen v případě, že je souvislý. Pokud souvislý není, mohou ale mít kostru jeho komponenty souvislosti. Kostra nesouvislého grafu je tedy lesem, nikoliv stromem, přičemž každý jeho strom je kostrou jedné komponenty grafu.



Budování stromu v grafu

Vstupem algoritmu je graf G a jeho vrchol v . Výstupem je graf s očíslovanými (přirozenými čísly ohodnocenými) vrcholy $1, \dots, n$.

Inicializuj strom T jako vrchol v .

Nastav počítadlo vrcholů na 1 a označ vrchol v ,

Dokud strom T neobsahuje všechny vrcholy komponenty, již je podgrafem:

 Vyber okrajovou hranu e .

 Nechť u je její vrchol, který není součástí stromu.

 Přidej vrchol u a hranu e do stromu T .

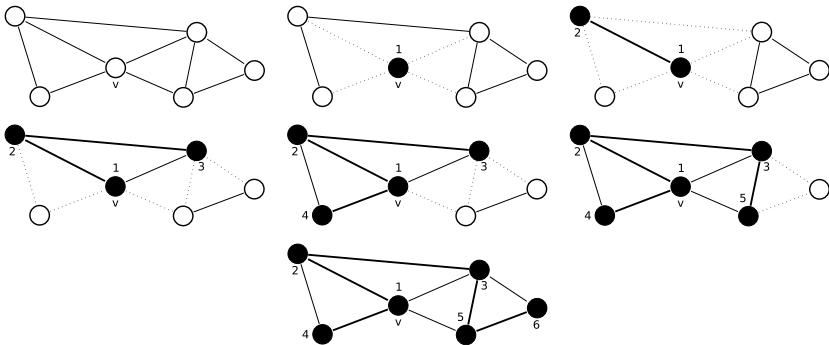
 Zvyš hodnotu počítadla vrcholů o 1.

 Očísluj vrchol u .

Vrať strom T .

Budování stromu v grafu – příklad

Obrázek: Růst stromu. Výstupní strom T je vyznačen tučně, okrajové hrany čárkovaně.

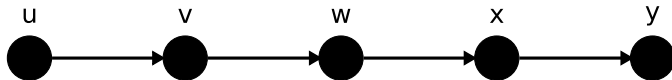


Budování stromu v grafu – vlastnosti

- Výběr okrajové hrany musí být proveden podle deterministického pravidla, aby výstup byl jednoznačný.
- Hranám je přidělena priorita a do grafu je přidána vždy ta s prioritou nejvyšší.
- Je-li algoritmus spuštěn z počátečního vrcholu v , strom T složený z očíslovaných vrcholů a stromových hran je kostrou komponenty grafu G , jíž je vrchol v součástí.
- Graf je spojitý právě když algoritmus budování stromu připojí všechny vrcholy tohoto grafu.

Budování stromu v orientovaném grafu

Algoritmus budování stromu v neorientovaném grafu je stejný jako v případě grafu orientovaného. Výstupní stromy se ovšem mohou lišit počtem vrcholů v závislosti na tom, který vrchol je vybrán jako počáteční.

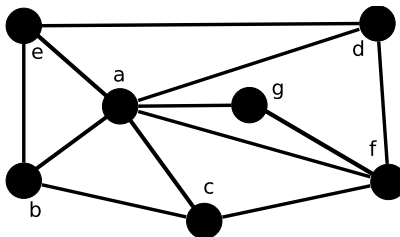


Výstup algoritmu budování stromu v grafu se bude lišit v závislosti na vybraném počátečním vrcholu.

Strom v grafu – cvičení

- 1 Nakreslete výstup algoritmu budování stromu v grafu, je-li vstupem graf na obrázku. Priorita hran je určena lexikografickým pořadím sestupně (lexikograficky menší hran má vyšší prioritu) a výpočet začíná ve vrcholu:

- 1 a
- 2 c



Průchod grafem do šířky

- BFS (Breadth-First Search)
- Předpokládáme, že vstupem je souvislý neorientovaný graf.
- Slouží k prohledání a navštívení všech vrcholů grafu.
- Vrcholy jsou navštěvovány v pořadí podle vzdálenosti od počátečního vrcholu.
- Nalezne nejkratší cestu z počátečního vrcholu do všech ostatních.
- Při průchodu grafem je budován strom cest do všech jeho vrcholů.
- Pro implementaci algoritmu se používá fronta.

Průchod grafem do šířky

Inicializuj strom T vrcholem v .

Nastav $\text{dist}[v] = 0$, $\text{dist}[x] = \text{nedosažitelný}$ pro $x \neq v$.

Inicializuj množinu okrajových hran jako prázdnou.

Inicializuj počítadlo vrcholů na 1 a označ jím vrchol v .

Dokud nejsou označeny všechny vrcholy, opakuj:

 Aktualizuj seznam okrajových hran.

 Vyber okrajovou hranu e , jež vychází z očíslovaného vrcholu w s nejnižším možným číslem.

 Přidej hranu e a její koncový vrchol u do stromu T .

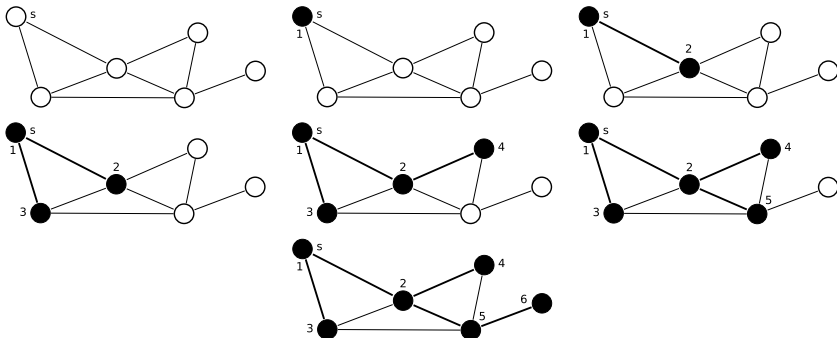
 Zvyš počítadlo vrcholů o 1.

 Očísluj přidaný vrchol u .

 Nastav $\text{dist}[u] = \text{dist}[w] + 1$.

Vrať strom T .

Průchod do šířky – příklad



Implementace průchodu do šířky

Inicializuj strom T vrcholem v .

Nastav $\text{dist}[v] = 0$, $\text{dist}[x] = -1$ pro $x \neq v$.

Inicializuj frontu vrcholů jako prázdnou a vlož v .

Inicializuj počítadlo vrcholů na 1 a označ jím vrchol v .

Dokud jsou ve frontě nějaké vrcholy, opakuj:

 Z počátku fronty odeber vrchol w .

 Dokud je e hrana vycházející z w do x :

 Je-li x neočíslovaný:

 Zvyš počítadlo vrcholů o 1.

 Očísluj x .

 Nastav $\text{dist}[x] = \text{dist}[w] + 1$.

 Přidej x na konec fronty.

 Přidej vrchol x a hranu e do T .

Vrať strom T .

Průchod do šířky – vlastnosti

Nechť u, v jsou vrcholy v grafu G . Vzdálenost vrcholů u, v (počet hran na nejkratší cestě mezi těmito vrcholy) budeme značit $\delta(u, v)$. Neexistuje-li cesta mezi těmito vrcholy, klademe $\delta(u, v) = \infty$.

Lemma 1

Nechť (u, v) je hrana v grafu G . Potom platí

$$\forall s \in V(G) : \delta(s, v) \leq \delta(s, u) + 1$$

Důkaz.

- Pokud do některého z vrcholů u, v neexistuje z s cesta, neexistuje cesta ani do druhého z nich a nerovnost platí.
- Nechť z vrcholu s do vrcholu u vede cesta p délky k . Potom existuje cesta z s do v složená z p a hrany (u, v) . Délka této cesty je právě $k + 1$ a nerovnost tedy platí. □

Průchod do šířky – vlastnosti

Lemma 2

Po skončení algoritmu BFS s počátečním vrcholem s platí

$$\forall v \in V(G) : \text{dist}[v] \geq \delta(s, v)$$

Důkaz.

Důkaz je veden indukcí k počtu vložení vrcholů do fronty:

- Tvrzení zřejmě platí po vložení počátečního vrcholu s do fronty.
- Uvažujme vkládání do fronty vrcholu x , do nějž vede hrana z w . Z indukčního předpokladu $\text{dist}[w] \geq \delta(s, w)$ a Lemmatu 1 plyne:

$$\text{dist}[x] = \text{dist}[w] + 1 \geq \delta(s, w) + 1 \geq \delta(s, x)$$



Průchod do šířky – vlastnosti

Lemma 3

Pro vrcholy $\langle v_1, \dots, v_k \rangle$ fronty v BFS platí

$$\text{dist}[v_k] \leq \text{dist}[v_1] \wedge \text{dist}[v_i] \leq \text{dist}[v_{i+1}]$$

Důkaz.

Indukcí k počtu operací vkládání a odebírání na frontě:

- Po vložení počátečního vrcholu zřejmě tvrzení platí.
- Indukční krok: odebráním vrcholu se platnost tvrzení změnit nemůže. Pokud $\text{dist}[v_1] = \text{dist}[v_k]$, pak po odebrání $\text{dist}[v_1]$ jsou přidány vrcholy o vzdálenosti rovné $\text{dist}[v_k] + 1$. V druhém případě, kdy $\text{dist}[v_1] + 1 = \text{dist}[v_k]$, jsou po odebrání $\text{dist}[v_1]$ přidány vrcholy o vzdálenosti rovné $\text{dist}[v_k]$.



Průchod do šířky – důkaz

Věta

Po skončení algoritmu BFS s počátečním vrcholem s na grafu G jsou očíslovány všechny vrcholy dosažitelné z s a v poli `dist` jsou hodnoty $\delta(s, v)$.

Důkaz.

Pro nedosažitelné vrcholy tvrzení zřejmě platí. Označme V_k množinu vrcholů, pro něž $\delta(s, v) = k$. Dokážeme, že pro každý vrchol v jsou následující kroky provedeny právě jednou:

- 1 Očíslování vrcholu.
- 2 Nastavení `dist[v]`
- 3 Vložení v do fronty.

Důkaz bude veden indukcí ke k .

Průchod do šířky – důkaz

Důkaz - pokračování.

- Pro $k = 0$, tedy počáteční vrchol, jsou tyto kroky provedeny jedinkrát při inicializaci.
- Indukční krok: Fronta se před skončením výpočtu nikdy nevyprázdní a poté, co je do ní vrchol vložen, se jeho vypočtená vzdálenost ani očíslování nemění. Uvažujme libovolný vrchol $v \in V_k, k \geq 1$. Z lemat 2 a 3 plyne, že vrchol v může být navštíven až poté, co jsou do fronty přidány všechny vrcholy z množiny V_{k-1} . Jelikož $\delta(s, v) = k$, existuje na cestě z s do v vrchol $u \in V_{k-1}$ takový, že existuje hrana (u, v) . Nechť u je první takový vrchol přidáný do fronty. Potom je vrchol v objeven právě z vrcholu u a platí $\text{dist}[v] = \text{dist}[u] + 1$.

Průchod do šířky – složitost

Každou hranu „projdeme“ právě jednou a všechny vrcholy také navštívíme právě jednou. Při vhodné implementaci fronty, která umožňuje přidávání a odebírání vrcholů v konstantním čase, je tedy časová složitost BFS $O(|V| + |E|)$.

Poznámka

Prezentovaný algoritmus lze snadno upravit tak, aby krom výpočtu vzdálenosti od počátečního vrcholu s vypočítal i jeho předchůdce na nejkratší cestě z s .

Průchod grafem do hloubky

- DFS – Depth First Search
- Namísto postupného procházení vrcholů od nejbližších ke kořeni postupuje algoritmus do hloubky – dokud je to možné, vybere vždy hranu vedoucí dále z vrcholu, do kterého právě vstoupil. Poté se vrací stromem ke kořenu – „backtrackuje“.
- Algoritmus i jeho implementace velice podobné BFS – stejná časová složitost.
- Projde všemi vrcholy grafu.
- Vstupem je rovněž neorientovaný souvislý graf
- Nenalezne nejkratší cesty do vrcholů.
- DFS vhodnější pro prohledávání stavových prostorů a heuristiky.
- K implementaci se používá zásobník.

Průchod grafem do hloubky

Inicializuj strom T vrcholem v .

Inicializuj množinu okrajových hran jako prázdnou.

Inicializuj počítadlo vrcholů na 1 a označ jím vrchol v .

Dokud nejsou označeny všechny vrcholy, opakuj:

 Aktualizuj seznam okrajových hran.

 Vyber okrajovou hranu e , jež vychází z očíslovaného vrcholu w s nejvyšším možným číslem.

 Přidej hranu e a její koncový vrchol u do stromu T .

 Zvyš počítadlo vrcholů o 1.

 Očísluj přidaný vrchol u .

Vrať strom T .

Průchod do hloubky – implementace

Inicializuj strom T vrcholem v .

Inicializuj zásobník vrcholů jako prázdný a vlož v .

Inicializuj počítadlo vrcholů na 1 a označ jím vrchol v .

Dokud jsou v zásobníku nějaké vrcholy, opakuj:

 Z vrcholu zásobníku odeber vrchol w .

 Dokud je e hrana vycházející z w do x :

 Je-li x neočíslovaný:

 Zvyš počítadlo vrcholů o 1.

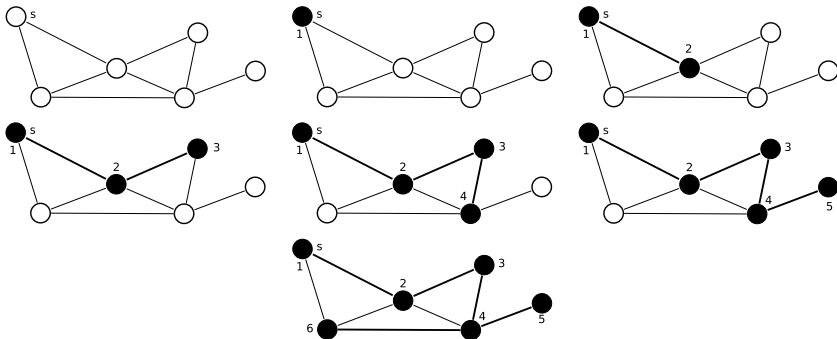
 Očísluj x .

 Přidej x na vrchol zásobníku.

 Přidej vrchol x a hranu e do T .

Vrať strom T .

Průchod do hloubky – příklad



Průchod do hloubky – vlastnosti

Věta

Nechť T je strom, jenž je výstupem běhu DFS na grafu G , a e hrana z G , která nepatří do stromu T . Nechť x, y jsou koncové vrcholy hrany e a platí, že x je algoritmem označen nižším číslem než y . Potom y je následníkem vrcholu x ve stromu T .

Důkaz.

Vrchol y byl algoritmem zřejmě navštíven později než x . Při vstupu algoritmu do x je hrana (x, y) označena jako okrajová. Vrchol y je však očíslován dříve než algoritmus vrchol x definitivně „opustí“ – y je tedy následníkem vrcholu x . □

Průchod grafem – cvičení

- 1 Pro graf z předchozího cvičení a počáteční vrchol b nakreslete výstup včetně očíslování
 - 1 průchodu do šířky.
 - 2 průchodu do hloubky.
- 2 Charakterizujte grafy, jejichž výstupní strom včetně očíslování je shodný v případě průchodu do šířky i do hloubky.
- 3 Upravte algoritmus BFS, aby u každého vrcholu uložil i jeho předchůdce na cestě z počátečního vrcholu.

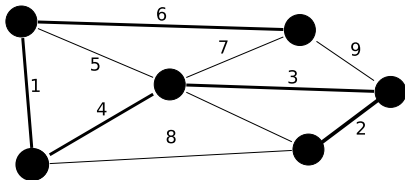
Minimální kostra grafu

Definice

Nechť G je souvislý graf s ohodnocenými hranami. Kostra grafu G , jejíž součet ohodnocení všech hran je nejnižší, se nazývá minimální kostra grafu G .

- Nalezení nejlevnější, ale neredundantní, počítačové či např. elektrické sítě spojující všechny koncové a aktivní prvky, resp. přípojná místa.

Obrázek: Minimální kostra je vyznačena tučně.



Primův algoritmus

- Hledání minimální kostry.
- Neprohledává systematicky všechny kostry grafu.
- Začíná v libovolném vrcholu a buduje strom.
- Nejvyšší prioritu mají hrany s nejnižším ohodnocením.
- Složitost závisí na datové struktuře ukládající okrajové hrany:

Matice susednosti $\mathcal{O}(V^2)$

Binární halda $\mathcal{O}(E \log(V))$

Fibonacciho halda $\mathcal{O}(E + V \log(V))$

Primův algoritmus

Vyber libovolný vrchol s vstupního grafu.

Inicializuj výstupní strom T vrcholem s .

Inicializuj množinu okrajových hran jako prázdnou.

Dokud T neobsahuje všechny vrcholy:

 Aktualizuj množinu okrajových hran.

 Nechť e je okrajová hrana s nejnižším ohodnocením
 a její koncový vrchol v nepatřící do T .

 Přidej vrchol v a hranu e do stromu T .

Vrať strom T .

Primův algoritmus – důkaz

Věta

Výstupní strom T_k vytvořený k iteracemi Primova algoritmu je podstromem minimální kostry grafu.

Důkaz.

Indukcí přes k :

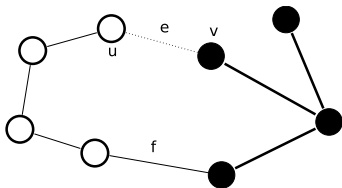
- 1 Pro $k = 0$ patří do grafu jen vrchol s .
- 2 Nechť T_k je podstromem minimální kostry T a strom T_{k+1} vznikne přidáním hrany e s minimálním ohodnocením, jejíž vrchol u patří do T_k a v nikoliv.



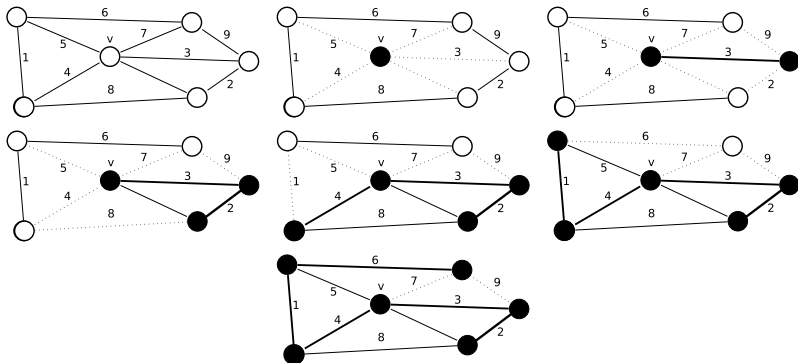
Primův algoritmus – důkaz

Důkaz – pokračování

Pokud T obsahuje hranu e , je T_{k+1} podstromem minimální kostry. V případě, že hrana e do minimální kostry nepatří, existuje v grafu $T + e$ (minimální kostra s přidanou hranou e) cyklus hranou e procházející. Nechť f je první hrana na „delší“ cestě mezi vrcholy u, v taková, že nepatří do T_k . Potom je i f okrajová hrana stromu T_k , ale má nižší prioritu (tudíž vyšší ohodnocení) než hrana e . Nahradíme-li tedy v T hranu f hranou e , celková váha se nezvýší a vzniklá kostra bude minimální.



Primův algoritmus – příklad



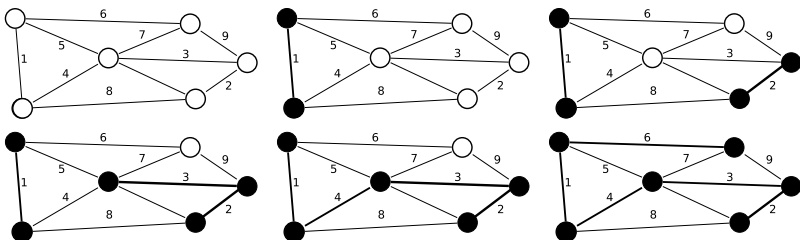
Kruskalův algoritmus

- Druhý algoritmus pro hledání minimální kostry grafu.
- Nepostupuje cestou budování stromu.
- Přidává hrany seřazené vzestupně podle jejich ohodnocení.
- Při použití vhodných datových struktur časová složitost $\mathcal{O}(E \log(V))$.

Kruskalův algoritmus

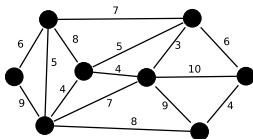
Setříd' hrany grafu G vzestupně podle ohodnocení.
Inicializuj seznam komponent souvislosti všemi vrcholy.
Dokud je ve výstupním stromu T více než 1 komponenta:
 Vyber hranu s nejnižším ohodnocením, která spojuje
 vrcholy ležící v různých komponentách.
 Přidej tuto hranu do výstupního stromu T .
 Aktualizuj seznam komponent.
Vrať strom T .

Kruskalův algoritmus – příklad



Minimální kostra – cvičení

- Na graf na obrázku aplikujte některý z algoritmů hledání minimální kostry.



- Graf na obrázku představuje komunikační síť, kde ohodnocení hran udává pravděpodobnost nechybovosti linky. Pravděpodobnost nechybové cesty v grafu je součinem pravděpodobností všech linek na trase. Najděte nejspolehlivější cestu z s do t .

