PB165 – Grafy a sítě

Kostry grafu

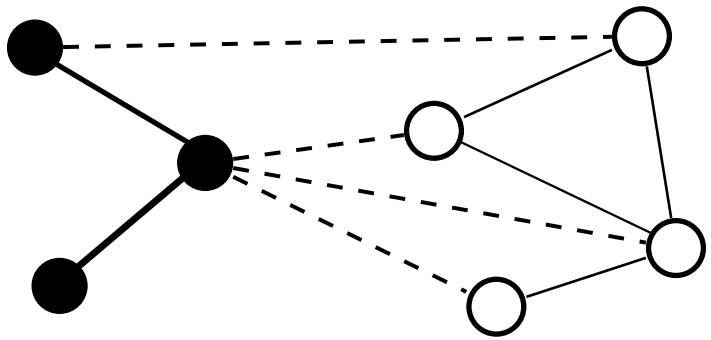
Obsah přednášky

- Úvod
- Budování stromu v grafu
- Průchod grafem
  - Průchod do šířky
  - Průchod do hloubky
- Minimální kostra grafu
  - Primův algoritmus
  - Kruskalův algoritmus

Definice

Stromem v grafu G rozumíme podgraf grafu G, který je stromem.
Hrany a vrcholy, které do tohoto stromu náleží, nazýváme stromové.
V opačném případě se hrana nazývá nestromová.
Hranu, jejíž jeden 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 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. □

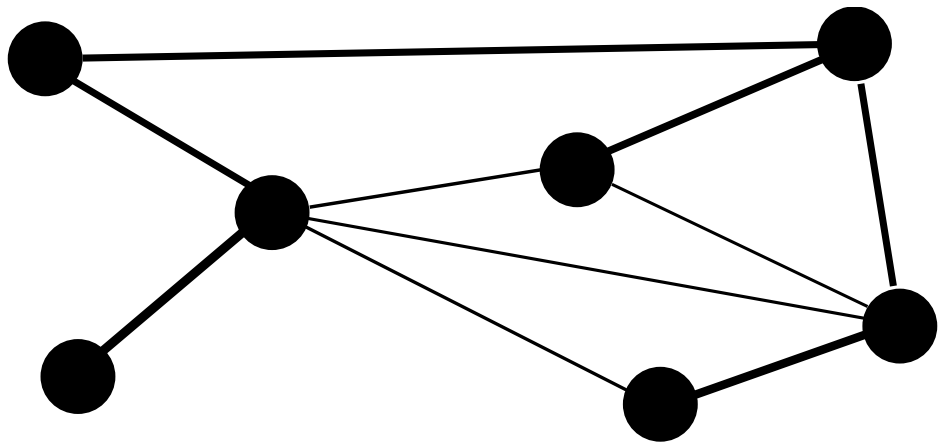
Kostra grafu

Definice

Kostra grafu G je takový strom T v grafu G, pro který platí $V(T) = V(G)$. *(vrcholy v T a v G jsou totožné)*

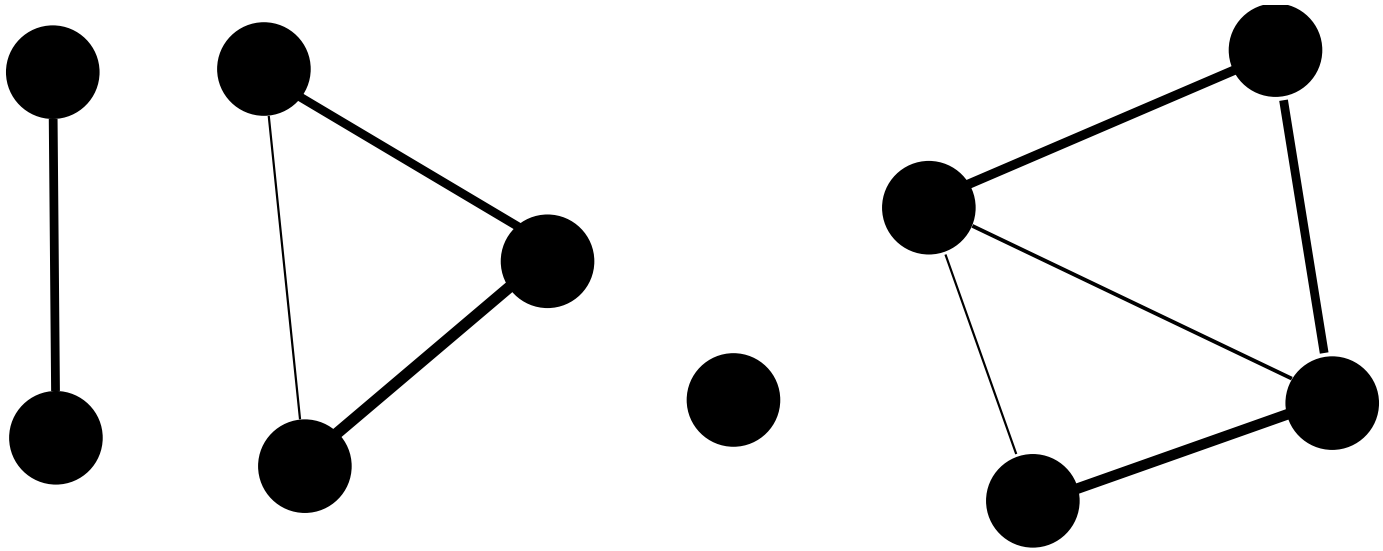
- Graf může mít více než jednu kostru.
- Každý acyklický podgraf grafu G je obsažen v alespoň jedné kostře grafu G.

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.



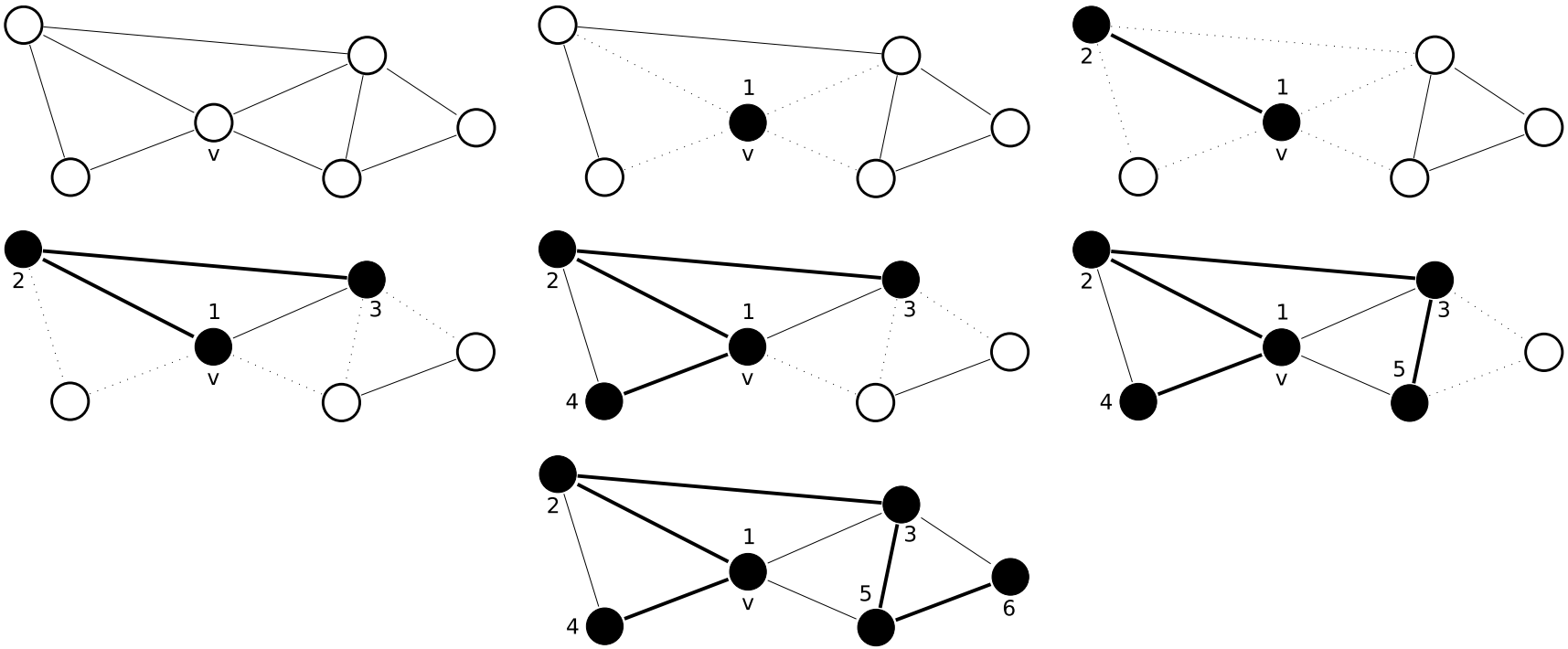
Budování stromu v grafu

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

```
inicializuj strom T jako vrchol v.
Nastav počítadlo vrcholů na 1 a očísluj vrchol v,
Dokud strom T neobsahuje všechny vrcholy komponenty,
  které 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.

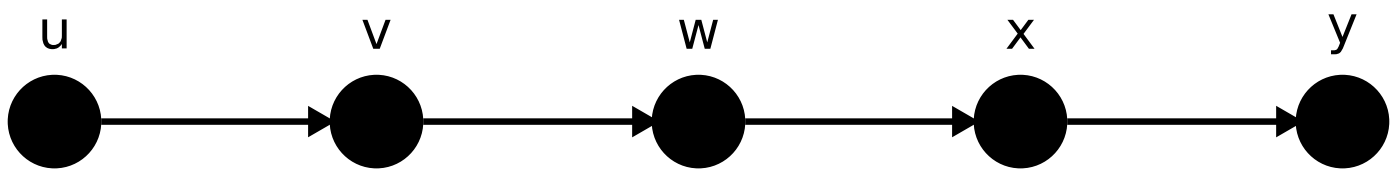


Budování stromu v grafu – vlastnosti

- 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

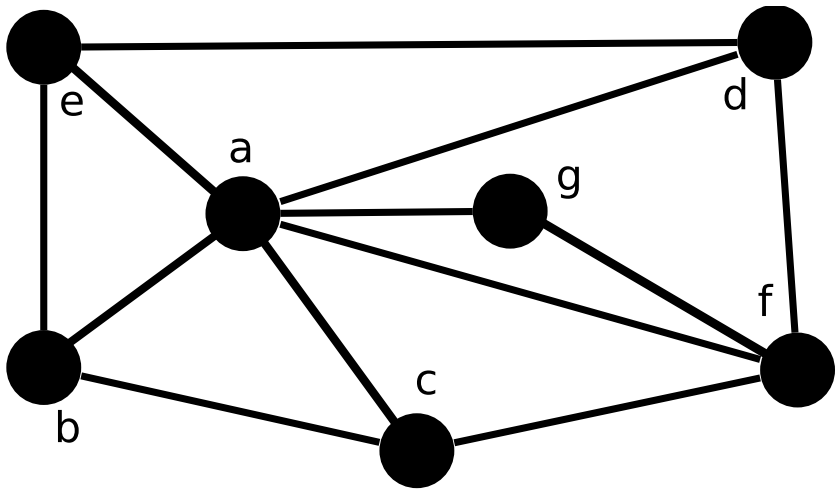
Algoritmus budování stromu v orientovaném grafu je stejný jako v případě grafu neorientovaného. Výstupní stromy se ovšem mohou lišit počtem vrcholů v závislosti na tom, který vrchol je vybrán jako počáteční.



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

Strom v grafu – cvičení

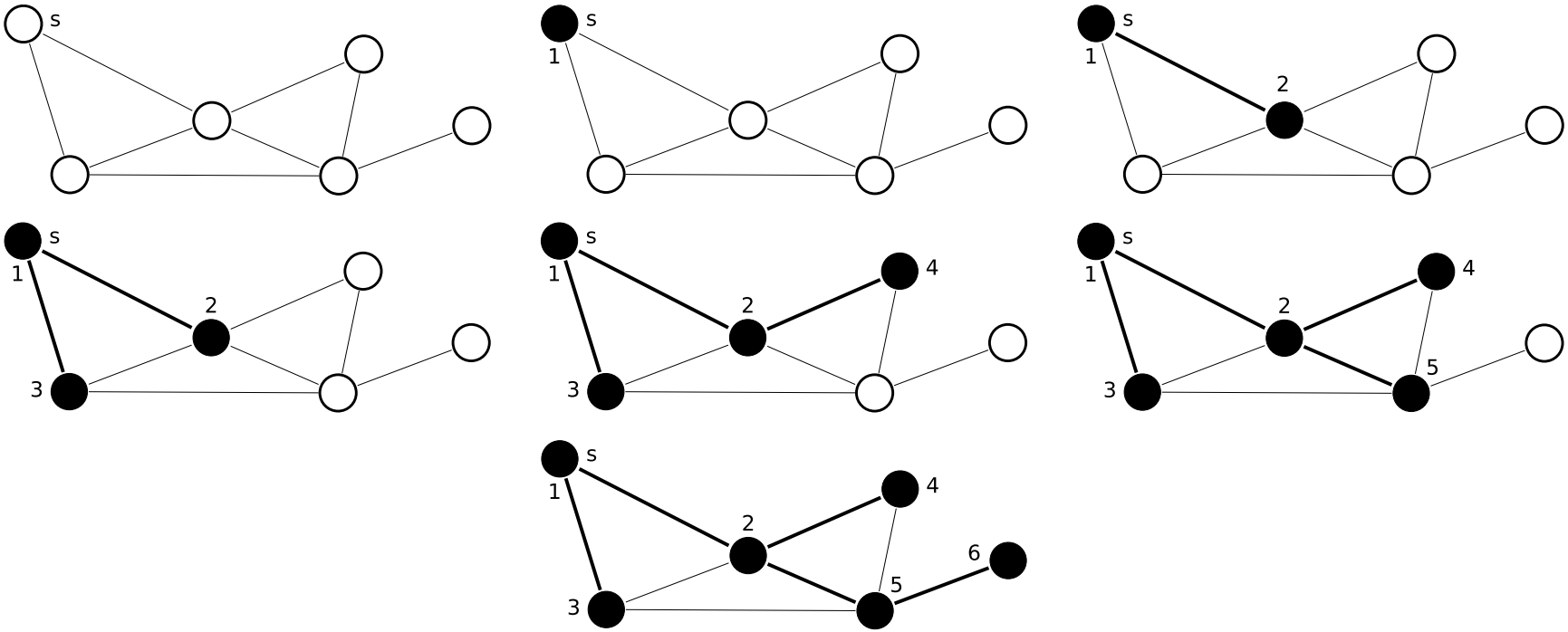
1. Nakreslete výstup algoritmu budování stromu v grafu, je-li vstupem graf na obrázku. Priorita hran je určena lexikografickým pořadím sestupně (lexikograficky menší hrana má vyšší prioritu) a výpočet začíná ve vrcholu:
   1. *a*
   2. *c*



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 do šířky – příklad



Algoritmus průchodu do šířky

```
Inicializuj strom T vrcholem s.
Nastav dist[s] = 0, dist[x] = -1 pro x != s.
Inicializuj frontu vrcholů jako prázdnou a vlož s.
Inicializuj počítadlo vrcholů na 1 a označ jím vrchol s.
Dokud jsou ve frontě nějaké vrcholy, opakuj:
 Z počátku fronty odeber vrchol w.
 Dokud neprojdeme všechny hrany e z vrcholu w do x, opakuj:
    Je-li x neočíslovaný:
      Zvyš počítadlo vrcholů o 1.
      Očísluj x.
      Nastav dist[x] = dist[w] + 1.
      Přidej x na konec fronty.
      Přidej vrchol x a hranu e do T.
Vrať strom T.
```

Průchod do šířky – vlastnosti

Definice

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 $\delta(u, v) = \infty$.

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 $\delta(s, v)$.*

Intuitivně:

- nejprve projdeme všechny vrcholy, které mají od s vzdálenost 1
- pak postupně procházíme vrcholy se vzdáleností 2, 3, ...
- a odpovídajícím způsobem se nastavuje `dist`

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 $O(|V| + |E|)$.

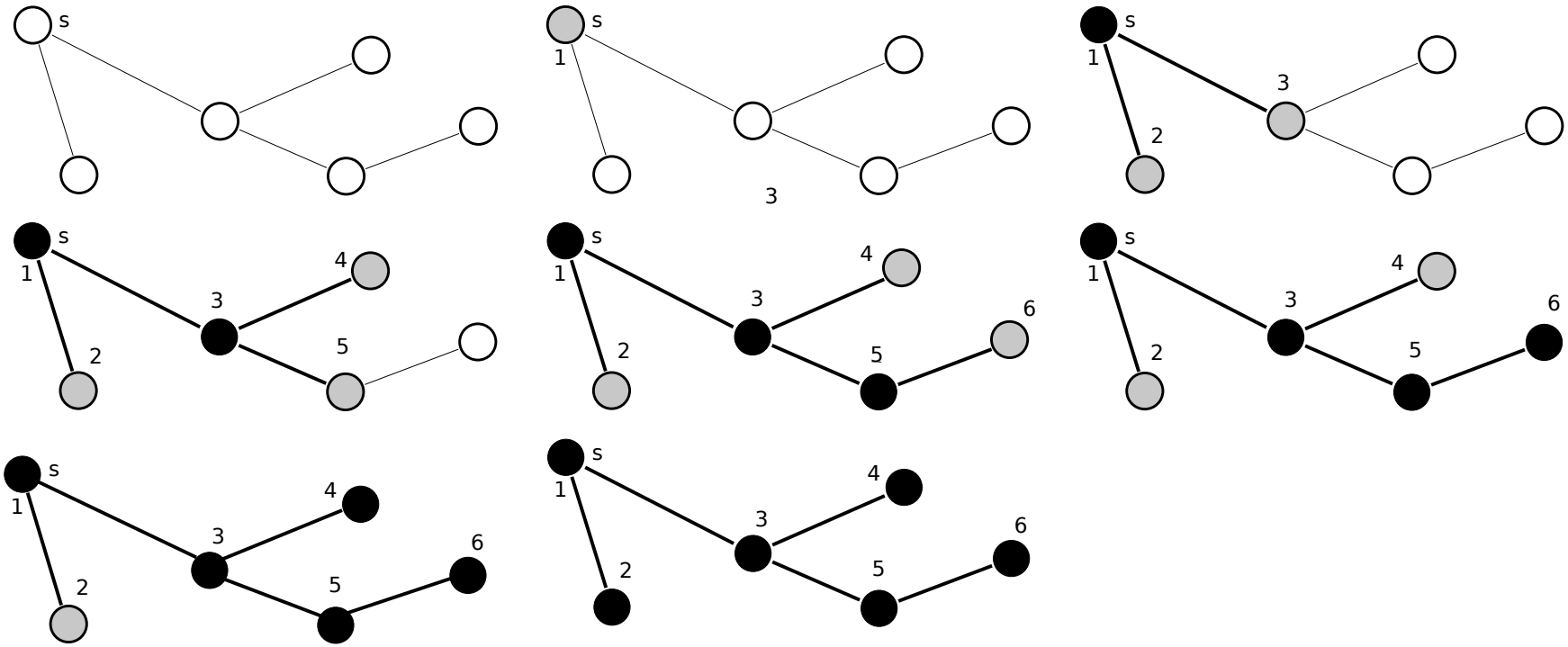
Poznámka

Prezentovaný algoritmus lze snadno upravit tak, aby kromě 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
- Algoritmus ale nenalezne nejkratší cesty do vrcholů.
- Algoritmus je vhodnější pro prohledávání stavových prostorů a heuristiky.
- K implementaci se používá zásobník.

Průchod do hloubky – příklad



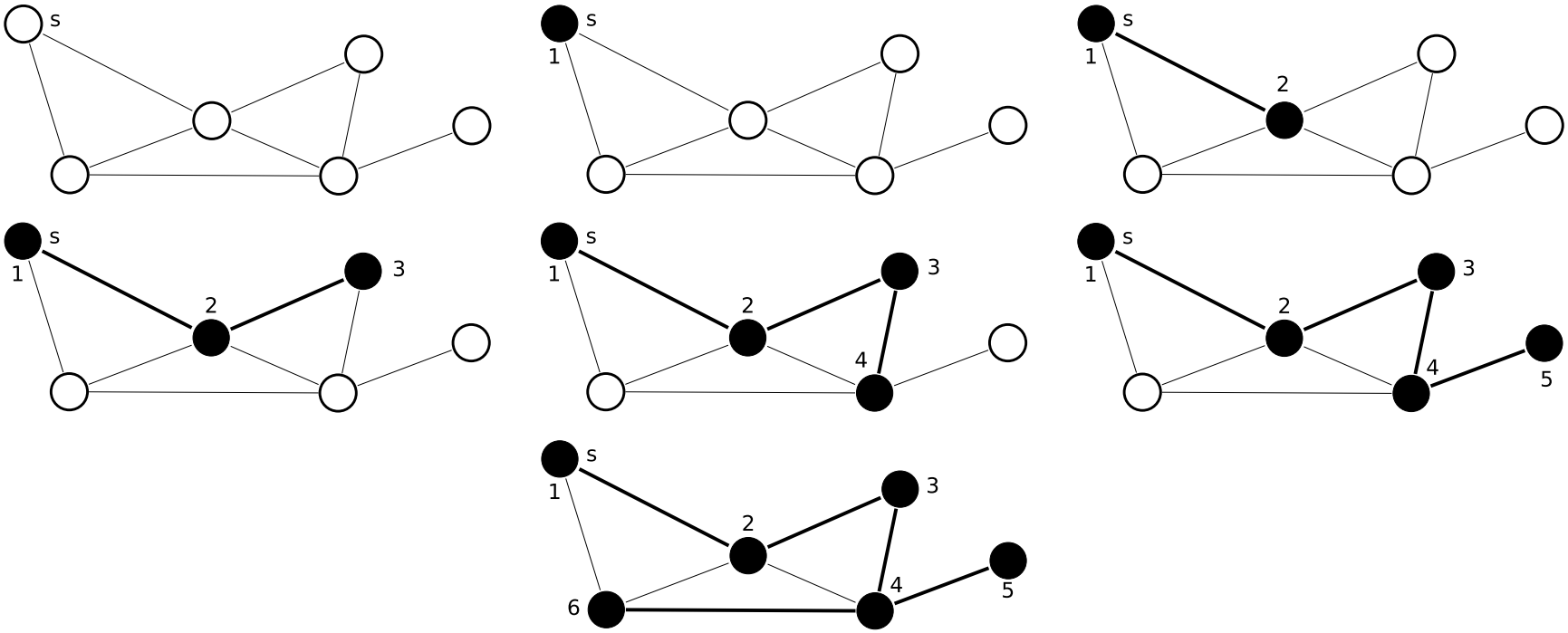
- vrchol zešedne, když je očíslován a přidán na zásobník
- vrchol zčerná, když je vybrán ze zásobníku

Algoritmus průchodu stromu do hloubky

Zjednodušená verze algoritmu pro průchod stromem

```
Inicializuj strom T vrcholem s.
Inicializuj zásobník vrcholů jako prázdný a vlož s.
Inicializuj počítadlo vrcholů na 1 a označ jím vrchol s.
Dokud jsou v zásobníku nějaké vrcholy, opakuj:
 Z vrcholu zásobníku odeber vrchol w.
 Dokud neprojdeme všechny hrany e z vrcholu w do x, opakuj:
    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



- vrchol zčerná, když je poprvé vybrán ze zásobníku a očíslován

Algoritmus průchodu grafu do hloubky

Obecná verze algoritmu pro průchod grafem

- do zásobníku vkládáme kromě vrcholu i jeho předchůdce

```
Inicializuj počítadlo vrcholů na 0.
Inicializuj zásobník vrcholů jako prázdný a vlož (s,nil).
Dokud jsou v zásobníku nějaké vrcholy, opakuj:
   Z vrcholu zásobníku odeber (w,pw).
   Je-li w neočíslovaný:
      Přidej vrchol w do stromu T.
      Je-li pw != nil:  (platí pro všechny vrcholy kromě s)
         Přidej hranu z vrcholu pw do w do stromu T.
      Zvyš počítadlo vrcholů o 1.
      Očísluj w.
      Dokud neprojdeme všechny hrany z w do x, opakuj:
         Je-li x neočíslovaný:
            Přidej (x,w) na vrchol zásobníku.
Vrať strom T.
```

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 (DFS), aby u každého vrcholu uložil i jeho předchůdce na cestě z počátečního vrcholu.

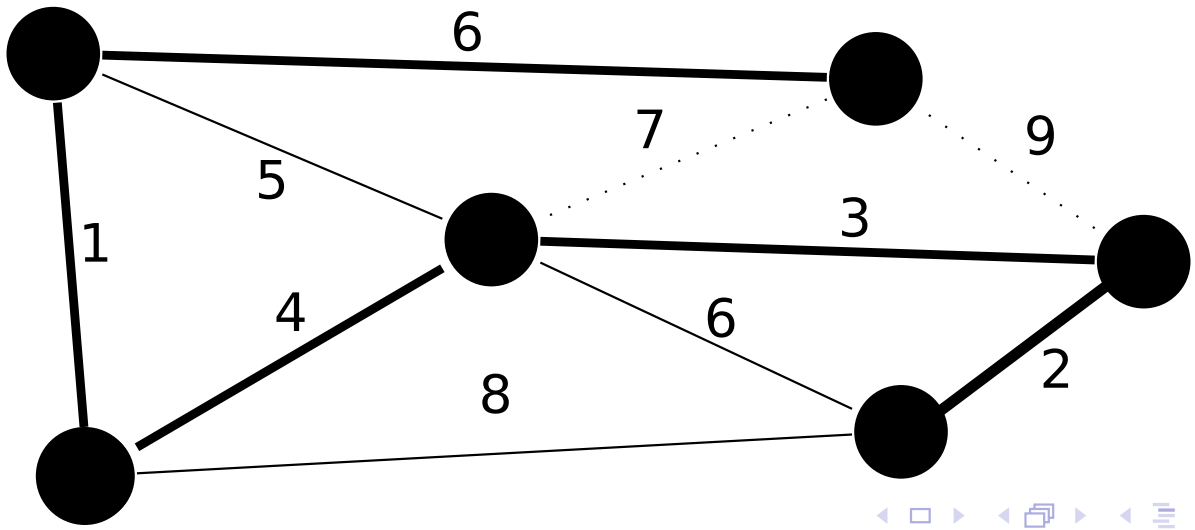
Minimální kostra grafu

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.
- Stále existuje jen jedna komponenta minimální kostry, která postupně roste.
- Složitost závisí na datové struktuře ukládající okrajové hrany:

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

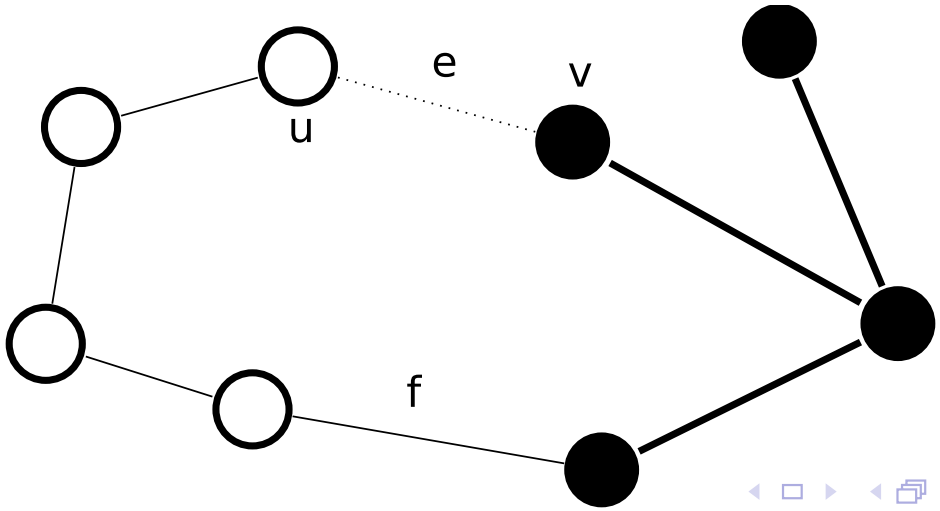
Indukcí přes k:

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 v patří do T_k a u nikoliv.

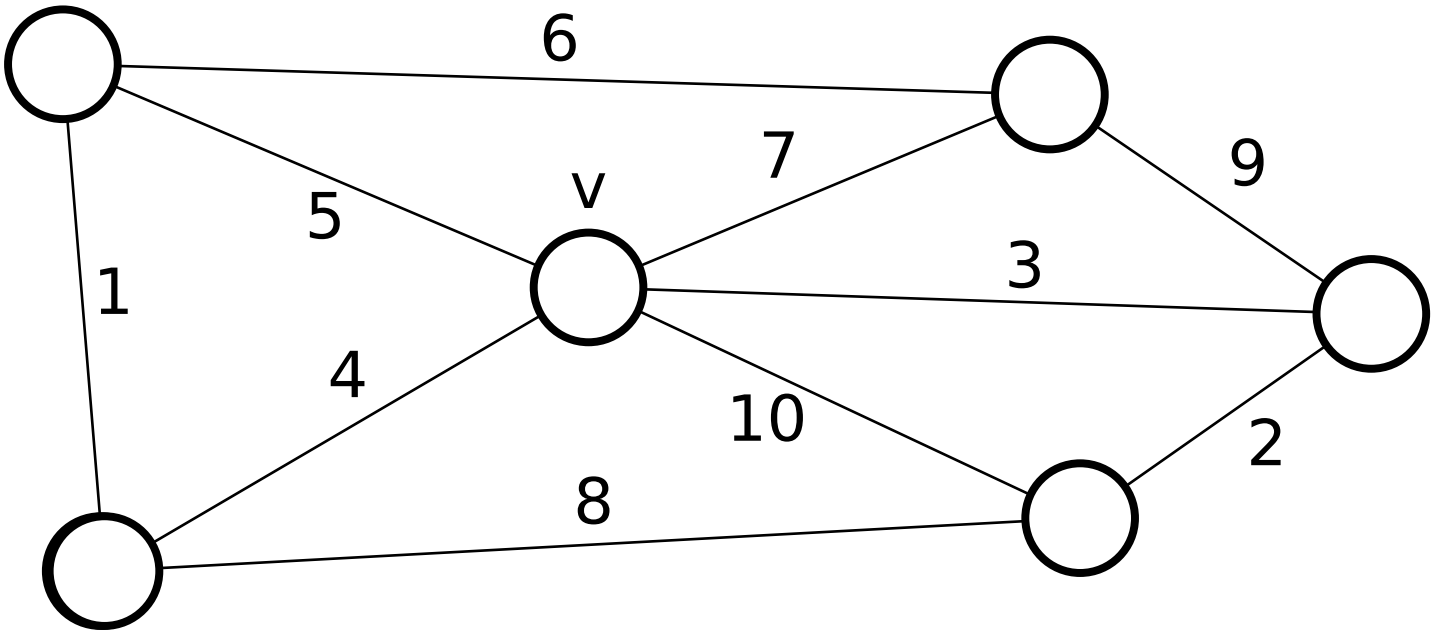
Primův algoritmus – důkaz

Důkaz – pokračování

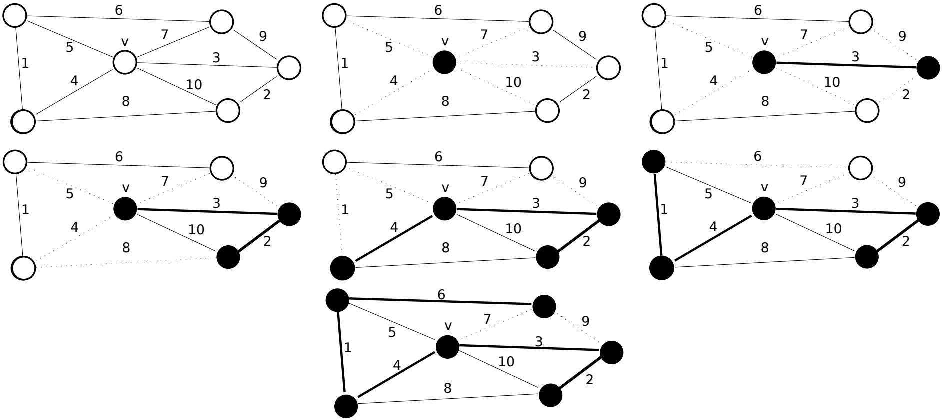
Pokud T obsahuje hranu e, je T_{k+1} podstromem minimální kostry. V případě, že hrana e do minimální kostry nepatří, existuje v grafu $T + e$ (minimální kostra s přidanou hranou e) cyklus hranou e procházející. Nechť f je první hrana na "delší"cestě mezi vrcholy u, v taková, že nepatří do T_k. Potom 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



Primův algoritmus – příklad



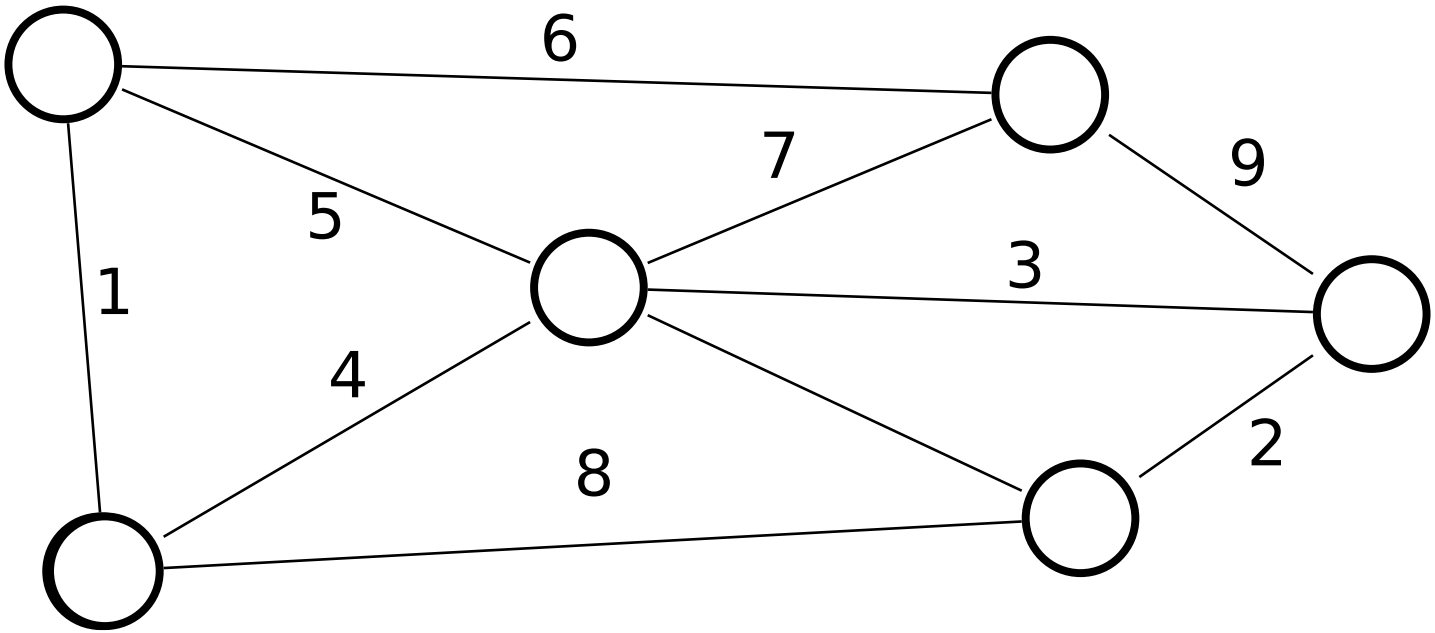
Kruskalův algoritmus

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

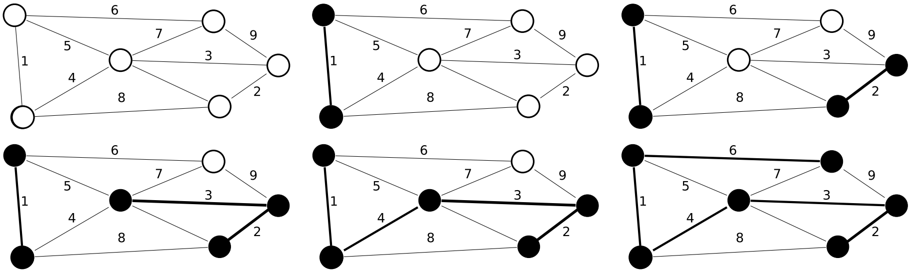
Kruskalův algoritmus

```
Setřiď 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

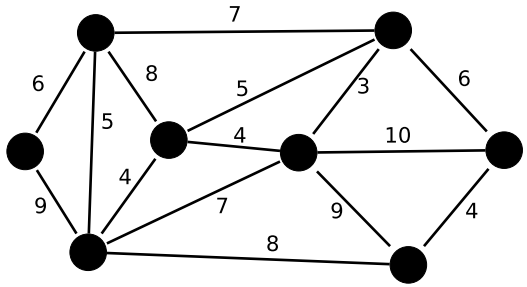


Kruskalův algoritmus – příklad



Minimální kostra – cvičení

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



2. 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.