Typografie a publikování – 5.projekt

Kruskalův algoritmus

Roman Machala (xmacha86)

Vysoké učení technické v Brně
Fakulta informačních technologií

6. září 2023

Obsah prezentace

Myšlenka Kruskalova algoritmu

- Slouží k nalezení minimální kostry grafu (kostry takové, že součet vah jejich hran je minimální)
- Hrany grafu musejí mít nezáporné ohodnocení (délku)
- Všechny vrcholy původního grafu jsou i vrcholy minimální kostry

Algebraický zápis

$G(V, H)$, kde:

- G je graf
- V je neprázdná množina vrcholů grafu G
- H je neprázdná množina kladně ohodnocených hran grafu G

Minimální kostra grafu

Definice

Podgraf $T \subseteq G$ souvislého grafu G se nazývá kostrou, pokud:

- T je stromem a
- $V(T) = V(G)$, neboli T propojuje všechny vrcholy G a
- Součet vah $T \subseteq G$ jejích hran je minimální

Minimální kostra grafu

Váhou (délkou) minimální kostry $T \subseteq G$ pak rozumíme:

$$c^w(T) = \sum_{h \in H(T)} w(h)$$

kde:

- $c^w(T)$ je minimální ohodnocení grafu $T \subseteq G$
- h je hrana grafu $\in H(T)$
- $w(h)$ je ohodnocení hrany

Algebraické značení minimální kostry

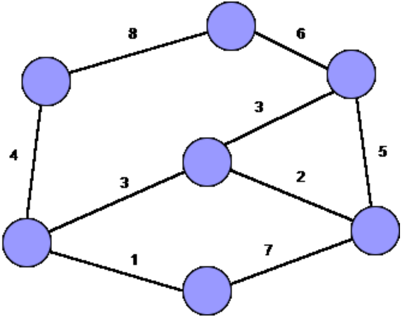
(T, w), kde:

- T je minimální kostra
- w je ohodnocení minimální kostry

Princip Kruskalova algoritmu

- Kruskalův algoritmus nejprve setřídí hrany dle jejich ohodnocení od nejmenších po největší
- Dále přidává hrany tak, aby nevznikl cyklus mezi vrcholy
- Všechny vrcholy musí být propojeny (do každého vrcholu musí existovat cesta)

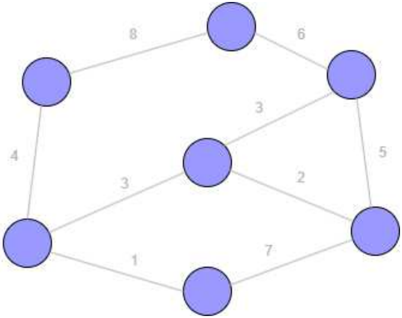
Příklad Kruskalova algoritmu



Vypíšeme jednotlivé ohodnocení hran a seřadíme je od nejmenších po největší:

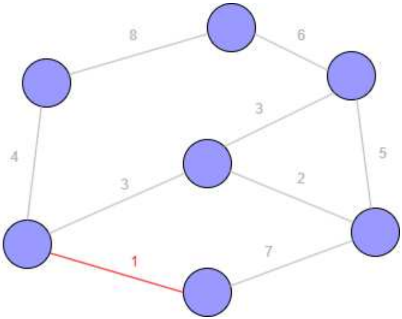
- 4,3,1,7,2,3,5,6,8
- 1,2,3,3,4,5,6,7,8

Příklad Kruskalova algoritmu



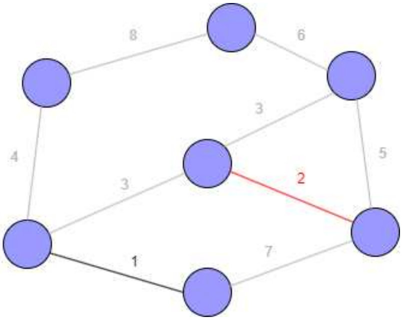
- Odstraníme všechny hrany, ale musíme si pamatovat jejich původní pozici a ohodnocení

Příklad Kruskalova algoritmu



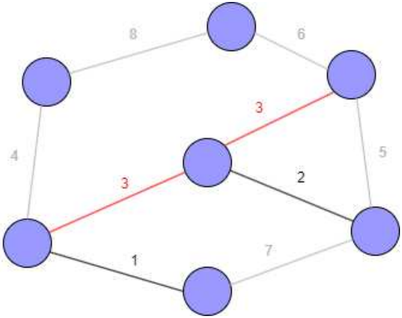
- Vybereme hranu s nejmenším ohodnocením (1) a přidáme ji do grafu, pokud netvoří cyklus

Příklad Kruskalova algoritmu



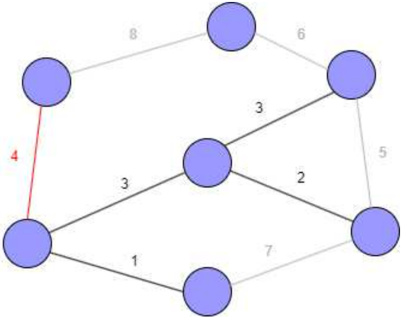
- Pokračujeme s další hranou s nejmenším ohodnocením a za stejných podmínek ji přidáme do grafu

Příklad Kruskalova algoritmu



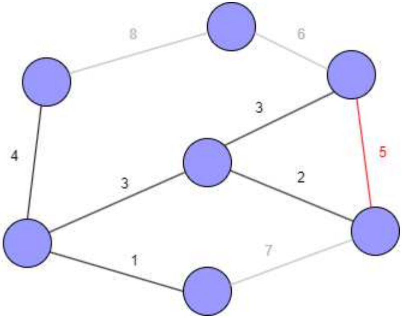
- Pokud máme více hran se stejným ohodnocením, můžeme si vybrat, kterou z nich přidáme
- V tomto případě, pokud přidáme obě hrany, tak ani jedna netvoří cyklus

Příklad Kruskalova algoritmu



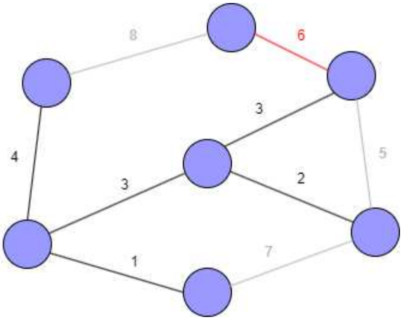
- Pokračujeme další hranou s nejmenším ohodnocením z hran, které jsme ještě neprošli

Příklad Kruskalova algoritmu



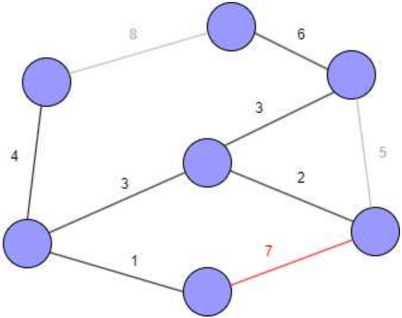
- V tomto případě nám další hrana tvoří cyklus (kružnici), do kostry ji tedy přidat nemůžeme

Příklad Kruskalova algoritmu



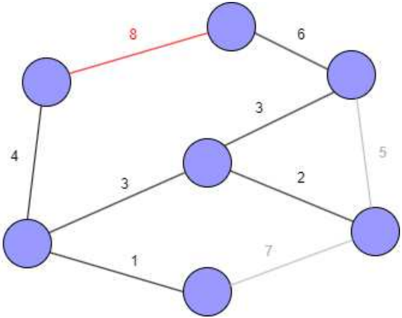
- Dle postupu pokračujeme dále a volíme další hranu, která ještě nebyla zkontrolována

Příklad Kruskalova algoritmu



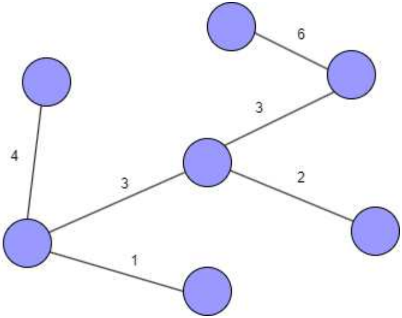
- V tomto případě hrana opět tvoří cyklus, proto ji do kostry nemůžeme použít

Příklad Kruskalova algoritmu



- Poslední hrana nám opět tvoří cyklus, proto ji do kostry nezahrneme

Příklad Kruskalova algoritmu



- Takto potom vypadá výsledná minimální kostra našeho grafu

Obecný postup Kruskalova algoritmu

Algoritmus 1: Kraskalův algoritmus

Input: Graph *graph, Edge *edges, Vertex *vertexes

```
Graph min_graph;                    //creates new empty graph
for all edges do
    getweight(edge);                //gets weights for all edges
    getvertexes(edge);              //gets vertexes for each edge
end for
sort(edges);                        //sorts all weights from lowest to highest
for all edges do
    //checks if edge creates a cycle, if not adds edge to min_graph
    if (edge != edgecycle(edge, min_graph)) then
        add_edge_to_graph(edge, min_graph);
    end if
end for
return min_graph;                   //returns minimal graph
```

Časová složitost algoritmu

Pokud předpokládáme, že algoritmus ještě bude zjišťovat, zda-li poskytnuté vrcholy náleží danému grafu, časová složitost alogritmu bude vypadat takto:

$$O(H\ log\ V\ +\ H \times T_{find}(V)\ +\ V \times T_{union}(V))$$

Kde:

- $T_{find}(V)$ a $T_{union}(V)$ jsou časové složitosti operací Find a Union na grafech s vrcholy V a hranami H, keré spadají do struktury Union-Find
- Union-Find je algoritmus zaměřený na zjišťování konektivity mezi jednotlivými prvky (v našem případě se jedná o zjištění cyklu mezi jednotlivými vrcholy)

Časová složitost algoritmu

Pokud předpokládáme, že v nejhorším případě je počet vrcholů $H = V^2$ a v souladu s pravidly asymptotické složitosti můžeme dále zanedbat aditivní konstanty, dostaneme:

$$O(H \times log(H))$$

Pokud jsou hrany již seřazeny nebo je možno k jejich seřazení použít řadící algoritmus s lineární složitostí (např. counting sort), tak je složitost rovna:

$$O(H \times \alpha(H))$$

Kde:

- α je inverzní Ackermanova funkce (odpovídá složitosti Union-Find)

Použité zdroje

- algoritmy.net
  algoritmy.net/Kruskaluv-algoritmus
- blog.kostecky.cz
  blog.kostecky.cz/Kruskaluv-algoritmus
- hackearth.com
  hackerearth.com/Find-Union
- referaty-seminarky.cz
  referaty-seminarky.cz/Ackermannova-funkce
- teorie-grafu.cz
  teorie-grafu.cz