Kostry

Na vstupu je graf s hranami ohodnocenými vahami

Předpoklady:

- Zadaný graf je souvislý, v důsledku $n = O(m)$
- Váhy hran jsou navzájem různé

Unikátnost: Za předpokladu různosti vah hran je minimální kostra unikátní.

Modré lemma: Nejlehčí hrana libovolného řezu patří do minimální kostry. Ekvivalentně, pro každou množinu vrcholů $X \subset V$, nejlehčí hrana mezi X a $V \backslash X$ patří do minimální kostry.

Červené lemma: Nejtěžší hrana libovolného cyklu nepatří do žádné minimální kostry.

Lemma: Každá hrana je "modrá" nebo "červená"

Algoritmy - hlavní myšlenky a porovnání

Kruskal
Hrany se setřídí podle váhy. Kostra se hladově vybuduje z co nejlevnějších hran.
Rychlý pokud hrany na vstupu jsou již setříděné

Kruskal Jarník
Začneme v libovolném vrcholu, pěstujeme strom rozšiřováním do okolí nejlevnější hranou.
S použitím vhodné haldy dobrá závislost složitosti na počtu hran
Rychlý pro husté grafy (při použití dobré haldy)

Borůvka
Paralelně pěstujeme mnoho stromů / začnem s triviálními kusy, ty pak srůstáme dohromady.
Paralelizace umožňuje jednodušší implementaci (založená pouze na BFS/DFS).
V konstrahující verzi rychle redukuje počet vrcholů, rychlý pro různé speciální třídy grafů, velmi rychlý pro rovinné grafy.

Kruskal

```
Function Kruskal(G, w váhy hran)
    setřiď G(E) dle váhy
    T ← ∅
    inicializuj UnionFind na T
    for e = (u, v) ∈ E dle rostoucí váhy do
        if Find(u) ≠ Find(v) then
            T ← T + e
            Union(u,v)
        end
    end
    return T
end
```

Korektnost

- **Souvislost** - Pokud je výsledná T nesouvislá, Kruskal nemohl vidět žádnou hranu spojující komponenty protože první takovou hranu vždy přijme.
- **Minimalita** - V momentě přijetí je každá hrana minimální na řezu mezi komponentami, které spojuje. Korektnost plyne z modrého lemma.

Složitost
$\mathcal{O}(m \log n)$ obecně, $\mathcal{O}(n * Union + m * Find)$ pro předtříděné hrany

Union-Find

Začínáme s prvky $1, \ldots, n$, každý je reprezentant vlastní jednoprvkové skupiny.
Operace Union(x,y) - sloučí skupiny prvků x,y pod jediného reprezentanta
Dotaz Find(x) - najde reprezentanta skupiny prvku x
Implementace: Skupiny jsou modelovány pomocí zakořeněných stromů. Každý prvek je buďto reprezentant (kořen), nebo si pamatuje nějakého předchůdce. Find postupuje po předchůdcích, až najde reprezentanta (kořen). Union se provede tak, že reprezentantovi skupiny prvku x se nastaví jako předchůdce y. Reprezentantem obou skupin se tak stává reprezentant skupiny prvku y.
Zlepšení: U každého reprezentanta si pamatujeme hloubku jeho stromu. Union pak připojí reprezetanta nižšího stromu pod reprezentanta hlubšího stromu. Pokud se hloubky rovnají, spojený strom bude mít hloubku o 1 vyšší. Získáváme garanci houbky $\mathcal{O}(\log n)$ a tedy Union i Find běží v čase $\mathcal{O}(\log n)$.
Amortizace: Pokaždé když procházíme strom (při Find/Union) po nalezení kořene projdeme cestu ještě jednou a všem prvkům nastavíme jako předchůdce přímo kořen stromu. Při kombinaci s předchozím lze dokázat (obtížné), že operace trvají amortizovaně $\mathcal{O}(\log^* n)$ kde $\log^*$ je inverz věžové funkce. Dokonce lze dokázat (velmi obtížné), že amortizovaná složitost obou operací je asymptoticky inverzní Ackermannova funkce z n (v praxi neodlišitelné od konstnaty).

Jarník

```
Function RelaxAlg(G, w váhy hran)
    stav[*] ← mimo
    v_0 ← libovolný vrchol
    stav[v_0] ← soused
    T ← ({v_0}, ∅)
    H = ∅ - halda vrcholů sousedících s T
    H.Insert(v,0)
    while H ≠ ∅ do
        v ← H.ExtractMin()
        e ← hrana zodpovědná za váhu v
        T ← T + e
        stav[v] ← v kostře
        for x soused v do
            if stav[x] = soused then
                H.Decrease(x,w(vx))
            end
            if stav[x] = mimo then
                H.Insert(x,w(vx))
            end
        end
    end
    return T
end
```

Korektnost

- **Souvislost -** Každý vrchol sousedící se stromem, je ke stromu připojen.
- **Minimalita -** V momentě přidání je každá hrana minimální na řezu mezi T a zbytkem grafu. Korektnost plyne z modrého lemma.

Složitost
$\mathcal{O}(n * Insert + n * Extract + m * Decrease)$
$\mathcal{O}(m \log n)$ s binární haldou, $\mathcal{O}(m + n \log n)$ s Fib. haldou

Borůvka

Function *Borůvka(G, w váhy hran)*
  $F \leftarrow (V, \emptyset)$
  while F *nesouvislý* **do**
    detekuj komponenty F
    pro každou komponentu najdi nejlehčí incidentní hranu
    přidej nejlehčí hrany do F
  end
  return F
end

Korektnost

- **Souvislost** - Algoritmus neskončí dokud les není souvislý
- **Minimalita** - Každá vybraná hrana je minimální na řezu mezi komponentami, které spojuje. Korektnost plyne z modrého lemma.

Složitost
Počet komponent ve fázi i je nejvýše $n/2^i$, počet fází je $\mathcal{O}(\log n)$
Složitost fáze je ekvivalentní složitosti BFS/DFS
Celková složitost $\mathcal{O}(m \log n)$

Kontrahující Borůvka

Function *Borůvka(G, w váhy hran)*
  $T \leftarrow \emptyset$
  while G *má více než jeden vrchol* **do**
    $F \leftarrow \emptyset$
    for v *vrchol* G **do**
      $e \leftarrow$ nejlehčí hrana incidentní s v
      $F \leftarrow F + e$
    end
    $T \leftarrow T + F$
    $G \leftarrow G/F$ - kontrakce hran v F
  end
  return T
end

Složitost
Počet vrcholů ve fázi i je nejvýše $n/2^i$, počet hran může klesat jen pomalu.
Složitost fáze je ekv. BFS/DFS na aktuálním počtu vrcholů a hran.
Celková složitost $\mathcal{O}(m \log n)$

Pro řídké grafové třídy uzavřené na kontrakce (třeba rovinné grafy) bude složitost $\mathcal{O}(n + m)$ protože počet hran bude v každé fázi klesat geometricky.