5. Požrešna metoda

greedy

5.1. Najcenejša vpeta drevesa

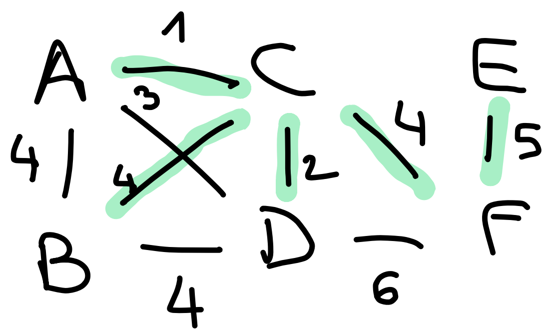
Imamo neusmerjen graf G z uteženimi povezavami; $w: E(G) \to \mathbb{R}^+$
Iščemo tak podgraf G', da bo G' povezan, $\underbrace{V(G) = V(G')}_{\text{vpet podgraf}}$ in
da bo $W(G') = \sum_{e \in E(G')} w(e)$ minimalna.

Hitro vidimo, da ta graf nima ciklov, saj sicer vedno lahko odstranimo povezave.

povezan graf + brez ciklov = drevo

Iščemo torej najcenejše vpeto drevo. (NVD / MST)

minimal spanning tree



5.1.1. Kruskalov algoritem

Ideja: zaporedoma jemlji najcenejšo povezavo, ki ne naredi cikla.

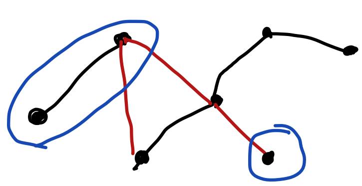
Kruskal

*1928 NY, NY, ZDA

† 2010 Maplewood, NJ, ZDA

5.1.2. Izrek o robu

Naj bo $S \subseteq V(G)$



Rob množice S je

$$\partial S = \{(u,v) \in E(G) \mid u \in S,\ v \notin S\}$$

Izrek Naj bo G povezan graf in $w: E(G) \to \mathbb{R}^+$
Naj bo $X \subseteq E(G)$ podmnožica povezav nekega NVD.
Naj bo $S \subseteq V(G)$ tak, da je $X \cap \partial S = \emptyset$

do sedaj zgrajeno drevo

kandidati za naslednjo povezavo NVD

Naj bo e najcenejša povezava v ∂S ← to implicira $\partial S \neq \emptyset$
Tedaj je $X \cup \{e\}$ tudi vsebovana v nekem NVD.

Dokaz

Naj bo T najcenejše vpeto drevo, ki razširja X.
Imamo dve možnosti:

- $e \in T$ ✓
- $e \notin T$

V tem primeru $T \cup \{e\}$ vsebuje cikel C.
Želimo pokazati, da C vsebuje še neko povezavo $e' \in \partial S$.
To je res, saj sicer C lahko razdelimo na del, ki je v celoti v S in del, ki je v celoti izven S.
Ker je e najcenejša povezava v ∂S, je $w(e') \geq w(e)$
in $W(T \setminus \{e'\} \cup \{e\}) = W(T) - w(e') + w(e) \leq W(T)$.
Ker je T NVD in je $W(T \setminus \{e'\} \cup \{e\}) \leq W(T)$, je tudi $T \setminus \{e'\} \cup \{e\}$ NVD.
Ali je $X \cup \{e\} \subseteq T \setminus \{e'\} \cup \{e\}$?
Ker je $X \subseteq T$ in je $e' \in \partial S$ ter velja $\partial S \cap X = \emptyset$, je $X \subseteq T \setminus \{e'\}$, zato je $X \cup \{e\} \subseteq T \setminus \{e'\} \cup \{e\}$ ∎

5.1.3. Algoritem

vhod povezan graf G, $w: E(G) \to \mathbb{R}^+$, ureditev E glede na w

izhod NVD $X \subseteq E(G)$

def kruskal (G, w):

za vse $v \in V(G)$:

naredi_singleton (v)

$X = \{\}$

za vse $(u,v) \in E$, urejene glede na w:

če poišči_razred $(u) \neq$ poišči_razred (v):

$X = X \cup \{(u,v)\}$

združi_razreda (u,v)

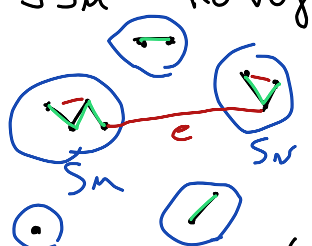
Potrebujemo strukturo, ki beleži disjunktne množice vozlišč.

① Na vsakem koraku zanke je $X \subseteq$ nekega NVD.

$n = 0$ ✓

$n \rightsquigarrow n+1$

- če na tem koraku nismo dodali nove povezave ✓
- če smo dodali (u,v), ki poveže prej nepovezani množici S_u in S_v. Tedaj je (u,v) najcenejša povezava na ∂S_u. Ker velja $\partial S_u \cap X = \emptyset$, lahko uporabimo izrek o robu.

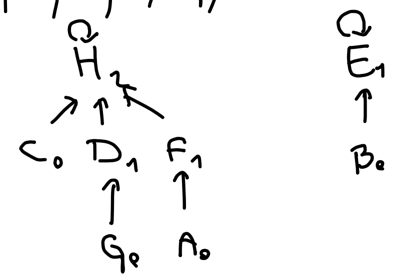


② $|V| \cdot T_{singleton} + 2 \cdot |E| \cdot T_{poišči} + (|V|-1) \cdot T_{združi}$

5.1.4. Implementacija disjunktnih množic (union-find)

Množice bomo predstavili z drevesi z rangom, ki (zaenkrat) predstavlja globino drevesa pod elementom

$\{A, C, D, F, G, H\}$ $\{B, E\}$



To lahko učinkovito predstavimo s tabelo

x	A	B	C	D	E	F	G	H
$\pi(x)$	F	E	H	H	E	H	D	H
rank(x)	0	0	0	1	1	1	0	2

```
def singleton(x):
    π(x) = x
    rank(x) = 0

def poišči(x):
    dokler x ≠ π(x):
        x = π(x)
    vrni x

def poišči(x):
    če x ≠ π(x):
        poišči(π(x))
    sicer:
        vrni x

def združi(x, y):
    px = poišči(x)
    py = poišči(y)
    če x ≠ y:
        če rank(px) < rank(py):
            π(px) = py
        če rank(px) > rank(py):
            π(py) = px
        če rank(px) = rank(py):
            π(py) = px
            rank(px) += 1
```

Primer

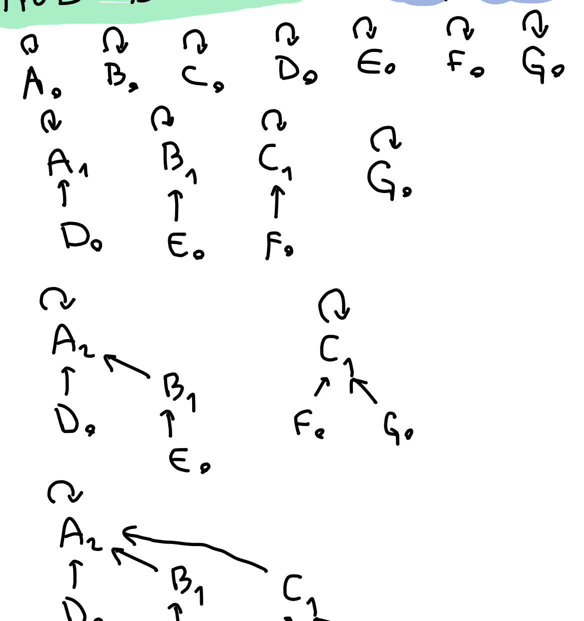
$A \cup D$ $B \cup E$ $C \cup F$ $C \cup G$ $E \cup A$ $B \cup G$

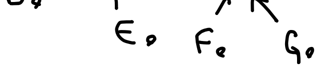
1)

2)

3)

4)





Trditev

a) $\text{rank}(x) < \text{rank}(\pi(x))$ če $x \neq \pi(x)$ ✓

b) Vsak koren ranga k ima vsaj 2^k potomcev vključno s sabo. ✓

c) Vozlišč ranga k je kvečjemu $n/2^k$, kjer je n št. vseh vozlišč ✓

d) maksimalni rang je $\log n$ ✓

Časovne zahtevnosti

singleton : $O(1)$
poišči : $O(\log n)$
unija : $O(\log n)$
Kruskal: $O(|E| \cdot \log |V|)$

Kompresija poti

Ko najdemo pot do korena, nanj preusmerimo vse povezave na poti

```
def poišči(x):
    če x ≠ π(x):
        π(x) = poišči(π(x))
    vrni π(x)
```

časovna zahtevnost je $O(\log^* n)$ (amortizirano)

$$\log^* n = \begin{cases} 0 & n \leq 1 \\ 1 + \log^*(\log n) & \text{sicer} \end{cases}$$

Dokaz

Ocenili bomo, da m klicev poišči na n vozliščih porabi $O(m \cdot \log^* n) + O(n \cdot \log^* n)$ časa.
Pri Kruskalu velja $n \leq m \leq n^2$, torej $O(m \cdot \log^* n)$ časa.
oz. $O(\log^* n)$ na operacijo.
Opazimo, da zaradi spremenjene funkcije poišči ne velja več, da je $\text{rank}(x)$ globina drevesa pod x, a trditve a)–d) še vedno veljajo.

Imejmo intervale

$I_0 = [1,1]$

$I_1 = [2,2]$

$I_2 = [3,4]$ $\qquad x \in I_k \Leftrightarrow \log^* x = k$

$I_3 = [5,16]$

$I_4 = [17, 2^{16} = 65536]$

$I_5 = [2^{16}+1, 2^{2^{16}}]$

Ko vozlišče z rangom $\in [\ell+1, 2^\ell]$ preneha biti koren, mu damo 2^ℓ žepnine. Ker je korenov z rangom iz tega intervala največ

$$\frac{n}{2^{\ell+1}} + \frac{n}{2^{\ell+2}} + \frac{n}{2^{\ell+3}} + \dots + \frac{n}{2^{2^\ell}} < \frac{n}{2^\ell}$$

zato na tem intervalu podelimo največ n žepnine

Intervalov je največ $\log^* n$, torej smo podelili $O(n \log^* n)$ žepnine.

Vsaka operacija poišči(x) porabi toliko časa, kot je dolga pot od x do korena.

$$x \to y_1 \to y_2 \to \dots \to y_t$$

Vrednosti rang(y_i) in rang$(\pi(y_i))$ sta lahko:

- na različnih intervalih, vendar to se zgodi največ $\log^* n$-krat, torej je strošek teh korakov $O(\log^* n)$
- na istem intervalu $[\ell+1, 2^\ell]$, pri čemer bomo strošek plačali iz žepnine y_i. Ker bomo y_i prevezali največ 2^ℓ-krat, ima dovolj žepnine.