

5. Požreška metoda

greedy

5.1. Najcenejša vpeta drevesa

Umašo neusmerjen graf G z ustrežnimi povezavami; vr: $E(G) \rightarrow \mathbb{R}^+$
 Iščemo tak podgraf G' , da bo G' povezan, $V(G) = V(G')$ in
 da bo $W(G') = \sum_{e \in E(G')} w(e)$ minimalna.

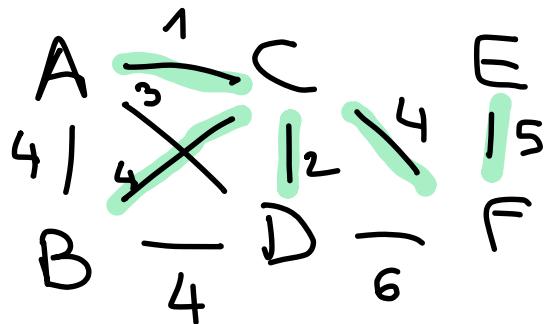
vpet
Podgraf

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

Povezan graf + brez ciklov = drevo

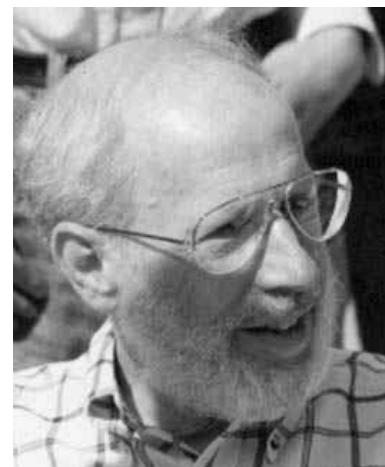
Iščemo torej najcenejše vpeto drevo. (NVD / MST)

minimal spanning tree



5.1.1. Kruskalov algoritmom

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



Kruskal

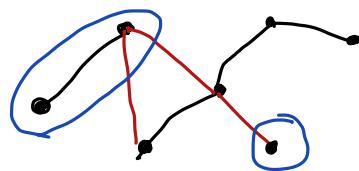
*1928 NY, NY, ZDA

†2010 Maplewood, NJ, ZDA

5.1.2. Izrek o robu

Naj bo $S \subseteq V(G)$

Rob množice S je



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

Izrek Naj bo G povezan graf in $w: E(G) \rightarrow \mathbb{R}^+$

Naj bo $X \subseteq E(G)$ podmnožica povezav nekega NVD.

Naj bo $S \subseteq V(G)$ tak, da je $X \cap JS = \emptyset$

do sedaj
zgrajeno
drvo

kandidati
za naslednjo
povezano NVD

Naj bo e najcenejša povezava v JS \leftarrow to implicira $JS \neq \emptyset$

Tedaj je $X \cup \{e\}$ tudi vsebovana v nekem NVD.

Dokaz

Naj bo T najcenejše vpeto drvo, 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 $e' \in JS$.

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 JS , 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 $e' \in JS$ ter velja $JS \cap X = \emptyset$,

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 , $w: E(G) \rightarrow \mathbb{R}^+$, vrednost 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 poisci-razred (u) \neq poisci-razred (v):

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

zdrži-razreda (u, v)

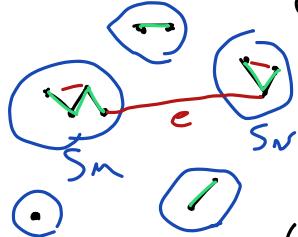
Potrebuješ strukturo, ki beleži disjunktna množice vozlišč.

① Na vsakem koraku začneš, če je $X \subseteq$ nekega NVD.

$$n=0 \quad \checkmark$$

$$n \rightsquigarrow n+1$$

- če na tem koraku nismo dodali novc povezave \checkmark
- če smo dodali (u, v) , ki poveže prej nepravilni množici S_u in S_v . Tedaj je (u, v) najcenejša povezava na $\bigcup S_u$. Ko velja $\bigcup S_u \cap X = \emptyset$,



lahko uporabimo
tukaj o robu.

② $|V| \cdot T_{\text{singleton}} + 2 \cdot (|E| \cdot T_{\text{poisci}} + (|V|-1) \cdot T_{\text{zdrži}}$

5.1.4. Implementacija disjunktivnih množic (union-find)

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

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

To lahko učinkovito predstavimo s tabelo

	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):

$$\Pi(x) = x$$

$$\text{rank}(x) = 0$$

def poišči(x):

dokler $x \neq \Pi(x)$:

$$x = \Pi(x)$$

vrni x

def poišči(x):

če $x \neq \Pi(x)$:

$$\text{poišči}(\Pi(x))$$

sicer:

vrni x

def združi(x, y):

$$p_x = \text{poišči}(x)$$

$$p_y = \text{poišči}(y)$$

če $x \neq y$:

če $\text{rank}(p_x) < \text{rank}(p_y)$:

$$\Pi(p_x) = p_y$$

če $\text{rank}(p_x) > \text{rank}(p_y)$:

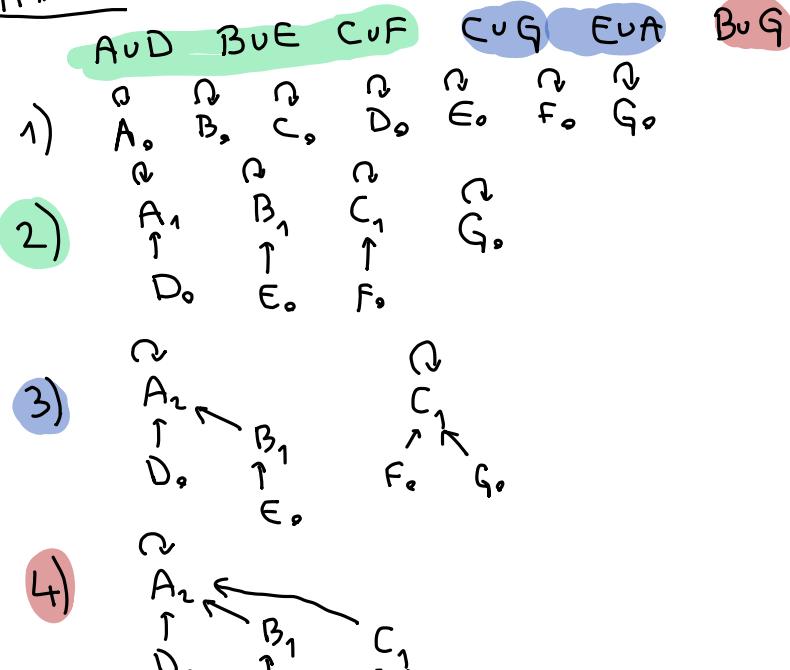
$$\Pi(p_y) = p_x$$

če $\text{rank}(p_x) = \text{rank}(p_y)$:

$$\Pi(p_y) = p_x$$

$$\text{rank}(p_x) += 1$$

Primer



Eo Fo Go

Trditev

- rank(x) < rank($\pi(x)$) Če $x \neq \pi(x)$ ✓
- Vsek koren ranga k ima vsaj 2^k potomcev vključno s sabo. ✓
- Vozlišč ranga k je krečjemu $\frac{n}{2^k}$, ker je n št. vseh vozlišč ✓
- maksimalni rang je $\log n$ ✓

Časovne zahtevnosti

singleton : $O(1)$

poisci : $O(\log n)$

unija : $O(\log n)$

Kruskal: $O(|E| \cdot \log |V|)$

Komprezija poti

Ko najdemo pot do korenja, napiši preusmerimo vse povezave na pot.

def poisci(x):

Če $x \neq \pi(x)$:

$\pi(x) = \text{poisci}(\pi(x))$
vrni $\pi(x)$

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 poisci ne 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 poisci ne velja več, da je $\text{rank}(x)$ globina drevesa pod x, a trditve a)-d) še vedno veljajo.

Umejmo intervale

$$\begin{aligned}
 I_0 &= [1, 1] \\
 I_1 &= [2, 2] \\
 I_2 &= [3, 4] \\
 I_3 &= [5, 16] \\
 I_4 &= [17, 2^{16} = 65536] \\
 I_5 &= [2^{16} + 1, 2^{2 \cdot 16}]
 \end{aligned}$$

$$x \in I_k \Leftrightarrow \log^* x = k$$

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

$$\frac{m}{2^{l+1}} + \frac{m}{2^{l+2}} + \frac{m}{2^{l+3}} + \dots + \frac{m}{2^{2l}} < \frac{m}{2^l}$$

Zato na tem intervalu podelimo največ m žepnine.

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

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

$$x \rightarrow y_1 \rightarrow y_2 \rightarrow \dots \rightarrow y_t$$

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

- na različnih intervalih, vendar to se zgoditi največ $\log^* n$ -krat, torej je strošek teh korakov $O(\log^* n)$
- na istem intervalu, pri čemer bomo strošek placali it žepnine y_i . Ker bomo y_i prevezali največ 2^l -krat, ima dolgi žepniti.



*1928 Chicago, IL, USA
† 2006 Ann Arbor, MI, USA

Galler



*1942 Ann Arbor, MI, USA
Fischer

5.1.5. Primov algortem



Jarník
* 1897, Praha, A-O
† 1970, Praha, CS

Prim
* 1921, Sweetwater, TX, ZDA

Ideja Imamo drevo, ki ga postopoma širimo z najcenejšo povezavo iz obresa (na robu $\Delta V(x)$)

vhod graf G , $w: E(G) \rightarrow \mathbb{R}^+$, $s \in V(G)$
izhod najcenejše vpeto drevo na G

def prim (G, w, s) :

za vsa $v \in V(G)$:
razdalja $[v] = \infty$
predhodnik $[v] = ?$

razdalja $[s] = 0$

$H = \text{vrsta_s_prednostja}(V(G), \text{razdalja})$

dokler H ni prazna:

$u = \text{odstrani_min}(H)$

za vsa $(u, v) \in E(G)$:

če $w(u, v)$

razdalja $[v] = w(u, v)$

posodobi (H, v)

predhodnik $[v] = u$

< razdalja $[v]$:

- ① Po izreku o robu
- ② $O(|V|) \cdot O(\log |V|) + O(|E|) \cdot O(\log |V|) = O(|E| \cdot \log |V|)$

③ Ja, Kruskal je $O(|E| \cdot \alpha(|V|))$

5.2. Huffmanova kodiranje

A	70 mil	00	0
G	20 mil	01	110
C	37 mil	10	10
T	3 mil	11	111
	130 mil	260 Mb	$\leq 13 \text{ Mb}$ oz. 18% manj

Huffmanov alg. izračuna optimalno brezpredponsko kodiranje glede na dane pojavitve znakov.

nabena koda ni
predpona kakšne druge
kode

Vseko brezpredponsko kodiranje lahko predstavljamo z drevesom, npr.



za dervo H in števila pojavitv f lahko izračunamo skupno ceno kot

$$E(f, H) = \sum_{x \in \Gamma} f(x) \cdot (\text{globina } x \text{ v } H)$$

in to je vrednost, ki jo želimo minimizirati za dan f.

Trditve Če sta x in y najglobiji vozlišči v optimalnem drevesu H , sta $f(x)$ in $f(y)$ najmanjši vrednosti v sliki f.

Dokaz V nasprotnem primor bi X in Y lahko zamenjena s še bolj redzima vozliščema in dobili cončnejše dervo.

Označimo vsa vozlišča drevsa H z njihovo pogostostjo

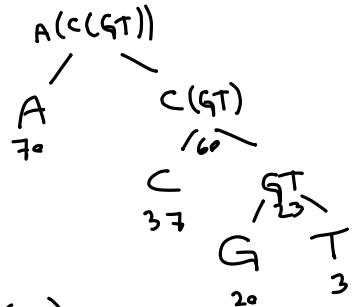
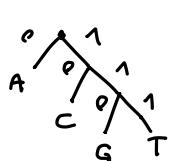
- vozlišče z otrokoma X in Y

izračemo z XY ter

mn dodelimo št. poj.

$$\hat{f}(XY) = \hat{f}(X) + \hat{f}(Y)$$

- za listo je $\hat{f}(X) = f(X)$



Tedaj velja:

$$E(f, H) = \sum_{n \in V(H)} \hat{f}(n)$$

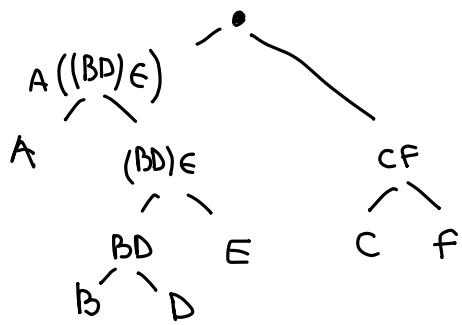
Za najveću vrednost x i y vrijedi

$$E(f, H) = \hat{f}(x) + \hat{f}(y) + E(f', H')$$
, kjer je H' drugo bilo x i y ,
zato tako problem prevedemo na manje gena

f' pa razširjen s simbolom XY
in $f'(XY) = f(x) + f(y)$

Primjer

A ↗
B ↘
C ↙ ↘
D ↗
E ↙ ↗
F ↘
BD ↗ ↘
 $(BD)E$ ↗ ↘
CF ↗ ↗



$A((BD)E)$ 1111