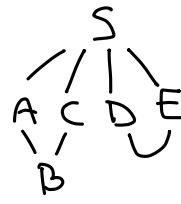
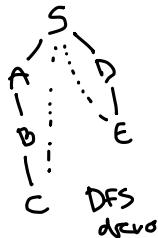
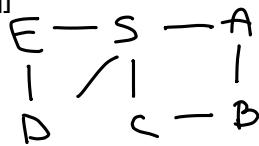


4. Poti v grafih

4.1. Razdalji

[[1, 2], [3], []]



← razdalja 1 od S
← razdalja 2 od S

$d_G(u, v)$ je dolžina najkrajše poti med u in v v G .
↑ razdalje

4.2. Iskanje v širimo

vhod graf G , $s \in V$

izhod $d_G(s, u)$ za vse $u \in V$

def razdalje (G, s):

za vsak $u \in V$:

$$\text{razdalja}_s[u] = \infty$$

$$\text{razdalja}_s[s] = 0$$

$$Q = [s]$$

(*) dokler Q ni prazna:

$$u = \text{odstraniPrvega}[Q]$$

za vsak $(u, v) \in E$:

če $\text{razdalja}_s[v] = \infty$:

vstavi (Q, v)

$$\text{razdalja}_s[v] = \text{razdalja}_s[u] + 1$$

① Za vsak $d \geq 0$ obstaja korak v zanki, da velja

a) v Q so natanko vozlišča na razdalji d

b) $\text{razdalja}_s[v] = \begin{cases} d_G(s, v) & \text{če je } d_G(s, v) \leq d \\ \infty & \text{sicer} \end{cases}$

z indukcijo na d .

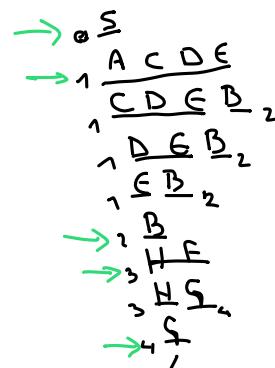
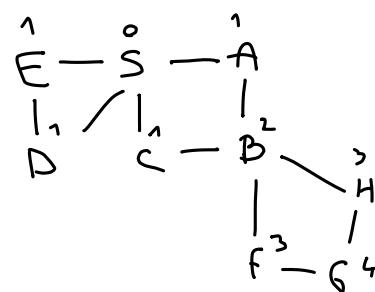
$d=0$) To velja ne točki (*)

$d \rightarrow d+1$)

Torej obstaja točka, ko je $Q = [v_1, \dots, v_k]$ sestavljene natanko iz vozlišč na razdalji d .

V naslednjih k korakih bomo odstranili v_i in dodali več njegove sosedje u , za katerih $d(s, u) = d+1$.

Zakaj: če je $d(s, u) < d+1$, je $\text{razdalja}_s[u] \neq \infty$, zato



m ne dodamo. Če je $d(s, m) > d+1$, potem pa velja
 $d(s, n_i) > d \Rightarrow$
 zakaj tako dodamo vsa vozlišča, za katera je $d(s, n) = d+1$?
 Če velja $d(s, m) = d+1$, obstaja pot $s - \dots - n - m$,
 kjer je $d(s, n) = d$, zato je $n \in Q$ in bomo obiskali m .

Ker velja $d(s, m) = d(s, n) + 1 = razdaljs[n] + 1$,
 je tudi razdaljs[m] pravilno nastavljeno.

Po k korakih se v Q torej na takško vsa vozlišča na razdalji
 $d+1$ in veljeta pogoj a&b. ✓

② $\mathcal{O}(|V| + |E|)$

③ Hitreje se ne da, ker potrebujemo toliko časa, da
 preberemo graf.



Zuse

*1910 Berlin, Nemčija
 †1995 Hünfeld, Nemčija

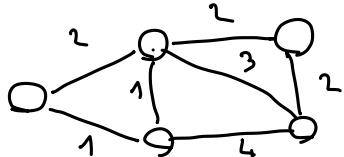


Moore

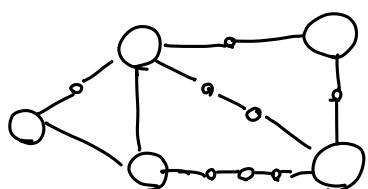
1925 Baltimore, MD, ZDA
 †2003 Madison, WI, ZDA

4.3. Razdalje na utreženih grafih

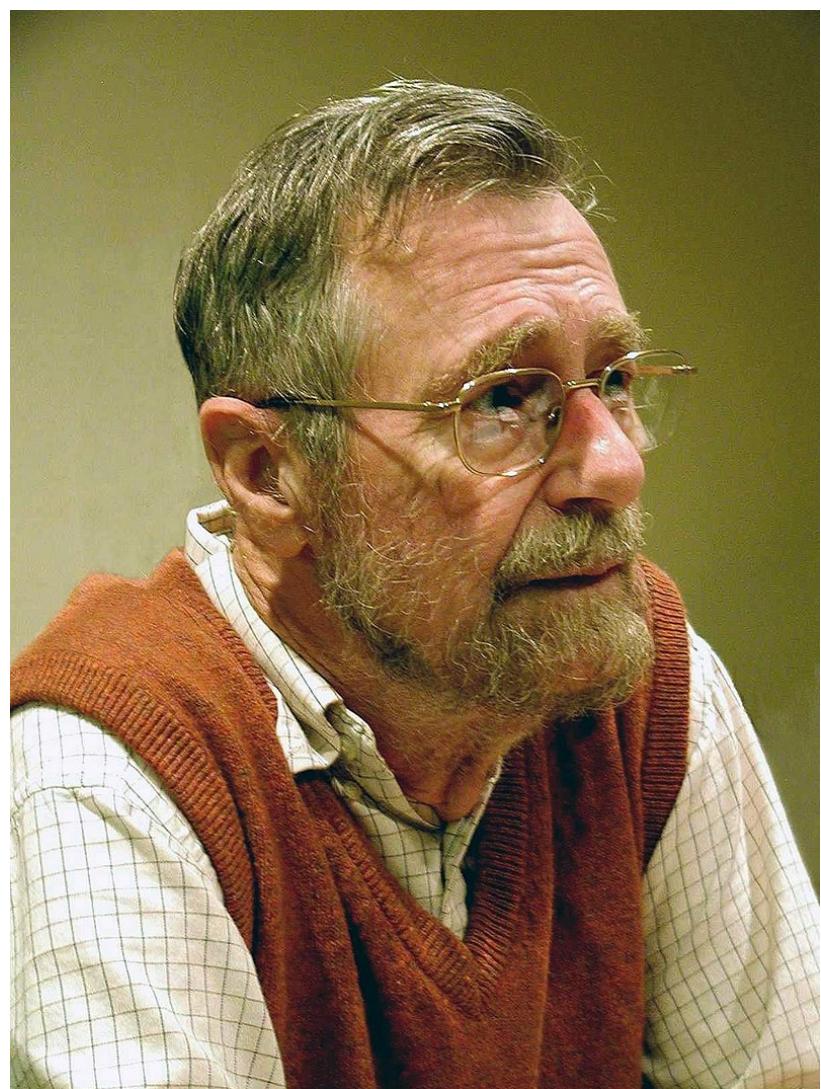
Na povezave obesimo uči, ki predstavlja razdaljo, ceno, trajanje
 potovanja čez povezano. Kako bi našli najcenejšo pot v tem primeru?



Če imamo uči $\in \mathbb{N}$, lahko graf pretvorimo v neutreženega

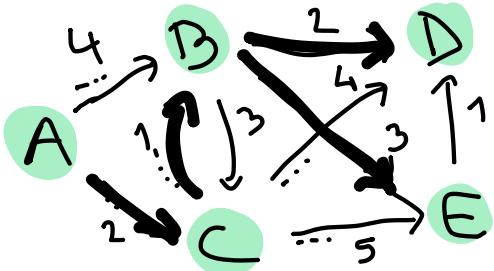


Videti je potencialno, zagotovo pa ne dela za cene, ki niso naravna števila.



Dijkstra

* 1930 Rotterdam, Nizozemska
† 2002 Nuenen, Nizozemska



A	0	0	0	0	0
B	∞	4	3	3	3
C	∞	∞	2	2	2
D	∞	∞	6	5	5
E	∞	∞	7	6	6

vhod graf G , $l: E(G) \rightarrow \mathbb{R}^+$, $s \in V(G)$
izhod $d_{G,l}(s, v)$ za vsak $v \in V(G)$
 ↑ najkrajša razdalja
 glede na l

def dijkstra(G, l, s):

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

razdalja[v] = ∞

predhodnik[v] = ?

razdalja[s] = 0

H = vrsta-s-prednostje ($V(G)$, razdalja)

dokler H ni prazna:

$u = \text{odstrani-min}(H)$

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

če $l(u, v) + \text{razdalja}[u] < \text{razdalja}[v]$:

$\text{razdalja}[v] = l(u, v) + \text{razdalja}[u]$

posodobi(H, v)

predhodnik[v] = u

① Ideja (več v 4.4.2)

$R = V \setminus H$ je del grafa, v katerem že poznamo najkrajšo pot.
Na vsakem koraku dodamo najbližje vozlišče m , ki še ni v R .

Recimo, da je $s - \dots - m - n$ najkrajša pot v G .

Potem je $w \in R$. Zakaj? Ker je $d(w, n) > 0$, je
 $d_{G, L}(s, w) < d_{G, L}(s, n)$. Če $w \notin R$, bi ga izbrali pred n .

Z indukcijo na št. obhodov zanke pokazemo, da
za vsak obhod obstaja $d \in R$, da velja

- 1) za vse $v \in R$ velja $d_{G, L}(s, v) \leq d$
- 2) za vse $v \notin R$ velja $d_{G, L}(s, v) > d$
- 3) za vse $v \in V$ velja

$$\text{razdalje}[v] = d_{\text{G} \setminus \{R \cup \{v\}\}, L}(s, v) \geq d_{G, L}(s, v)$$

inducirani graf
 G omejen na vozlišča $R \cup \{v\}$
in povezave med njimi

② $\mathcal{O}(|V|) \cdot T_{\text{odstrani-min}}(|V|) + \mathcal{O}(|E|) \cdot T_{\text{posodi}}(|V|) + T_{\text{vrsta-s-pred}}^*(|V|)$

Casi so odvisni od implementacije

4.5. Implementacije vrste s prednostjo

Pri implementaciji nas bodo zanimali samo prednosti, pri padajoči
vrednosti lahko ignoriramo.

4.5.1 S seznamom

Uname seznam nevegrnih vrednosti

$T_{\text{odstrani-min}}$	$\mathcal{O}(m)$
T_{posodi}	$\mathcal{O}(1)$
$T_{\text{vrsta-s-pred}}$	$\mathcal{O}(1)$
Dijkstra	$\mathcal{O}(V ^2)$

4.5.1 Z večnim seznamom

T_{odstrani}	$\mathcal{O}(1)$
T_{posodi}	$\mathcal{O}(n)$
$T_{\text{vrsta-s-pred}}$	$\mathcal{O}(n \cdot \log n)$
$T_{\text{vrsta-s-pred}}^*$	$\mathcal{O}(n)$
Dijkstra	$\mathcal{O}(E \cdot V)$

4. 5. 2. S kopicami (heap)

Todstran: $O(\log n)$

Tposodbi: $O(\log n)$

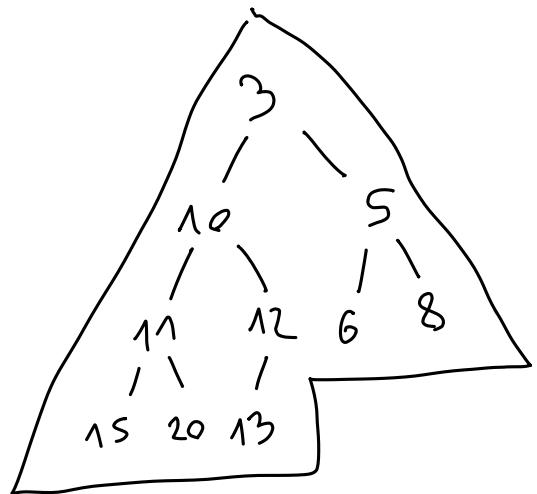
Tvrst-s-pred: $O(n)$

Dijkstra: $O(|E| \log |V|)$

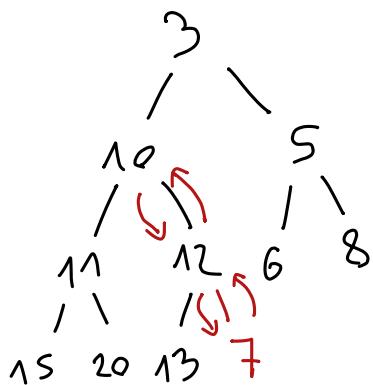
Kopica je dvojško drevo, ki:

- je levo poravnano

- ima vsakega starša manjšega od otrok



Vstavljanje



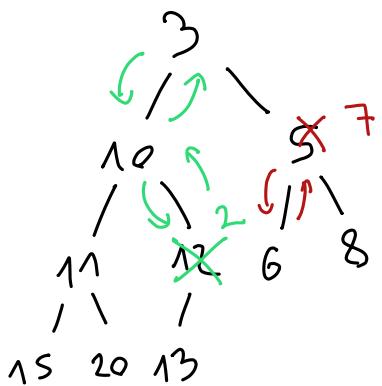
1. Novo vrednost damo na zadnje mesto



2. Vrednost pomikamo navzgor, dokler ni manjša od svojih otrok



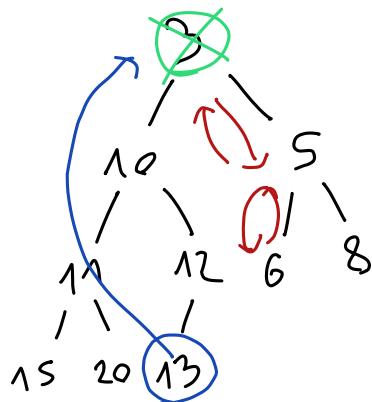
Posodabljanje



- Če posodobimo na manjšo vrednost, ravnamo kot pri v 2. koraku

- Če posodobimo na večjo vrednost, pa element potapljamemo tako, da ga zamenjujemo z manjšim od otrok

Odstanjevanje minimum



1. Minimum je na vrhu



2. Namesto njega vstavimo zadnji element



3. Novi koren poprimo na pravo mesto.



Zaradi kompaktne oblike kopic jih lahko učinkovito predstavimo s tabelami

3	10	5	11	12	6	8	15	20	13
---	----	---	----	----	---	---	----	----	----



Uporaba kopic:

- vrsta s prednostjo
- urejanje s kopicami / Heapsort



Williams

* 1929, Chippenham, ZK
† 2012 Ottawa, Kanada

4.5.3. Variante kopic

- kopice z d otroci

odstran-min : $\mathcal{O}(d \cdot \log_d n)$

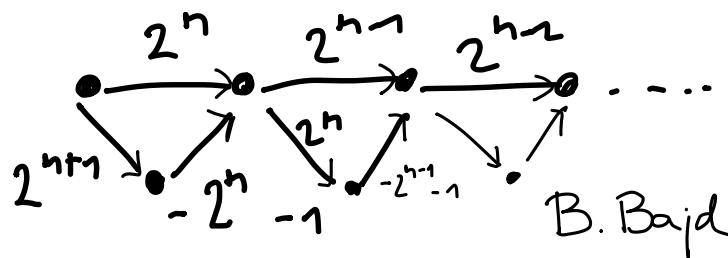
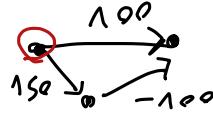
vstavi : $\mathcal{O}(\log_d n)$

Dijkstra : $\mathcal{O}(|V| + |E|) \cdot \log_d |V| + d \cdot |V| \cdot \log_d (|V|)$

- Fibonaccijeve kopice

4.6. Najcenejše poti v grafih z negativnimi udežmi

Če imamo neg. udeži, Dijkstrov alg. ne deluje pravilno.



če prilagodimo
Dijkstra, da gremo ponovno
 $O(2^n)$ raziskovati
izboljšana
vozljivca

4.6.1. Bellman-Fordov algoritem

def posodobi (razdalja, μ , v):

$$\text{razdalja}[v] = \min(\text{razdalja}[v], \text{razdalja}[\mu] + l_{\mu, v})$$

brez neg. ciklov

vhod

graf G , $s \in V(G)$, $l: E(G) \rightarrow \mathbb{R}$

izhod

$d_{G, l}(s, v)$ za vse $v \in V(G)$

def bellman-ford (G, s, l):

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

$\text{razdalja}[v] = \infty$

$\text{razdalja}[s] = 0$

ponovi $|V(G)| - 1$ krat:

za vse $(\mu, v) \in E(G)$:

posodobi (razdalja, μ, v)

vrni razdalja

① Recimo, da je najkrajsa pot med s in v enaka

$$s \xrightarrow{e_2} M_1 \xrightarrow{e_1} M_2 \xrightarrow{e_5} \dots \xrightarrow{e_7} M_k \xrightarrow{e_3} v$$

Če je $\text{razdalja}[M_k]$ pravilno nastavljena, posodobi (raz., M_k, v) pravilno nastavi tudi $\text{razdalja}[v]$. Če vmes poklicemo ka druge se ta lastnosti nespremeni.

$$\left. \begin{array}{c} s \checkmark \\ e_1 \quad e_2 \quad e_3 \quad \dots \quad e_m \\ M_1 \checkmark \quad e_1 \quad e_2 \quad e_3 \quad e_4 \quad \dots \quad e_m \end{array} \right\} |V| - 1$$

$e_1 e_2 e_3 \dots e_m$

Ker je najkrajša pot čez največ $|V|-1$ povezav (sicer bi imeli neg. cikel) po $|V|-1$ obhodih čez vsa povezave pravilno posodobimo vse razdalje.

Če želimo odkriti ali obstaja neg. cikel, naredimo še en obhod. Če x ni nač spremenjilo, cikla ni, sicer je in najkrajše poti ni.

2. $\mathcal{O}(|V| \cdot |E|)$

3. ?

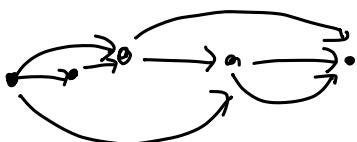


Bellman
*1920 NY, NY, ZDA
†1984 LA, CA, ZDA



Ford
*1927 Houston, TX, ZDA
†2017 St. Barbara, CA, ZDA

4.6.2. Najkrajše poti v DAG



za vsak $v \in V(G)$ v top. ureditvi:

$\mathcal{O}(|V| + |E|)$

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

posodobi (razdalje, u, v)

To pravilno nastavi vse razdalje, ker na vsaki najkrajši poti povezave obiščemo v pravem vrstnem redn.

