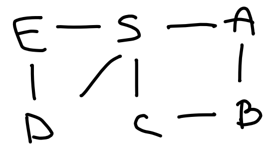
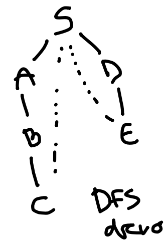
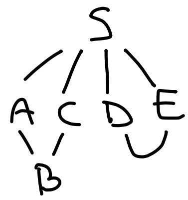
4. Poti v grafih

4.1. Razdalja





DFS drevo



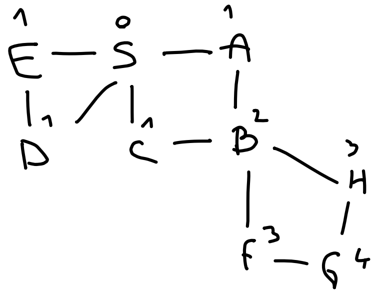
← razdalja 1 od S

← razdalja 2 od S

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

↑ razdalja

4.2. Iskanje v širino



vhod graf G, $s \in V$

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

def razdalja (G, s):

  za vsak $v \in V$:

    razdalja$_s[v] = \infty$

  razdalja$_s[s] = 0$

  $Q = [s]$

(*) dokler Q ni prazna:

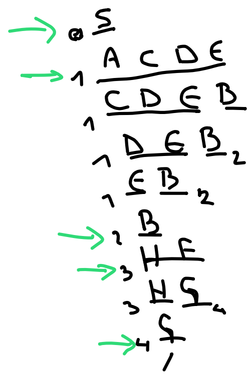
    u = odstraniPrvega $[Q]$

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

      če razdalja$_s[v] = \infty$:

        vstavi (Q, v)

        razdalja$_s[v]$ = 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) 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 na točki (*)

$d \rightsquigarrow d+1$)

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

V naslednjih k korakih bomo odstranili v_i in dodali vse njegove sosede u, za katere velja $d(s,u) = d+1$.

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

u ne dodamo. Če je $d(s,u) > d+1$, potem pa velja $d(s, v_i) > d$ ↯

Zakaj tako dodamo vsa vozlišča, za katera je $d(s,u) = d+1$?
Če velja $d(s,u) = d+1$, obstaja pot $s - \dots - v - u$,
kjer je $d(s,v) = d$, zato je $v \in Q$ in bomo obiskali u.

Ker velja $d(s,u) = d(s,v) + 1 = \text{razdalja}_s[v] + 1$,
je tudi $\text{razdalja}_s[u]$ pravilno nastavljena.
Po k korakih so v Q torej natanko vsa vozlišča na razdalji $d+1$ in veljata pogoja a & b. ✓

(2) $O(|V| + |E|)$

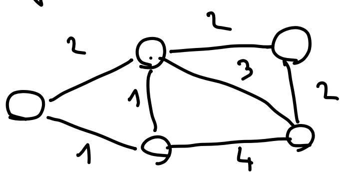
(3) Hitreje se ne da, ker potrebujemo toliko časa, da preberemo graf.

Zuse
$*$1910 Berlin, Nemčija
† 1995 Hünfeld, Nemčija

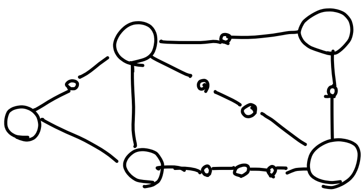
Moore
$*$1925 Baltimore, MD, ZDA
† 2003 Madison, WI, ZDA

4.3. Razdalje na uteženih grafih

Na povezave obesimo uteži, ki predstavljajo razdaljo, ceno, trajanje potovanja čez povezavo. Kako bi našli najcenejšo pot v tem primeru?



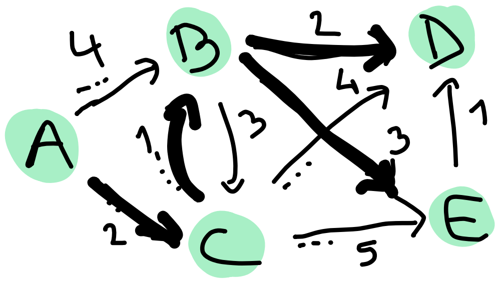
Če imamo uteži iz $\mathbb{N}$, lahko graf pretvorimo v neuteženega



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

Dijkstra

* 1930 Rotterdam, Nizozemska
† 2002 Neunen, Nizozemska



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

vhod graf G, $\ell: E(G) \to \mathbb{R}^+$, $s \in V(G)$

izhod $d_{G,\ell}(s,v)$ za vsak $v \in V(G)$

najkrajša razdalja glede na ℓ

```
def dijkstra(G, ℓ, s):
    za vse v ∈ V(G):
        razdalja[v] = ∞
        predhodnik[v] = ?
    razdalja[s] = 0
    H = vrsta_s_prednostjo(V(G), razdalja)
    dokler H ni prazna:
        u = odstrani_min(H)
        za vse (u,v) ∈ E(G):
            če ℓ(u,v) + razdalja[u] < razdalja[v]:
                razdalja[v] = ℓ(u,v) + 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 najcenejše poti.

Na vsakem koraku dodamo najbližje vozlišče u, ki še ni v R.

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

Potem je $v \in R$. Zakaj? Ker je $\ell(v,u) > 0$, je $d_{G,\ell}(s,v) < d_{G,\ell}(s,u)$. Če $v \notin R$, bi ga izbrali pred v.

Z indukcijo na št. obhodov zanke pokažemo, da za vsak obhod obstaja $d \in \mathbb{R}$, da velja

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

$$\text{razdalja}[v] = d_{G|_{R \cup \{v\}}, \ell}(s,v) \geq d_{G,\ell}(s,v)$$

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

② $O(|V|) \cdot T_{odstrani\_min}(|V|) + O(|E|) \cdot T_{posodobi}(|V|) + T^*_{vrsta\_s\_pred}(|V|)$

Časi so odvisni od implementacije

4.5. Implementacije vrste s prednostjo

Pri implementaciji nas bodo zanimale samo prednosti, pripadajoče vrednosti lahko ignoriramo.

4.5.1 S seznamom

Imamo seznam neurejenih vrednosti

$T_{odstrani\_min}$	$O(n)$
$T_{posodobi}$	$O(1)$
$T_{vrsta\_s\_pred}$	$O(1)$
Dijkstra	$O(\|V\|^2)$

4.5.2 Z urejenim seznamom

$T_{odstrani}$	$O(1)$
$T_{posodobi}$	$O(n)$
$T_{vrsta\_s\_pred}$	$O(n \cdot \log n)$
$T^*_{vrsta\_s\_pred}$	$O(n)$
Dijkstra	$O(\|E\| \cdot \|V\|)$

4.5.2. S kopicami (Heap)

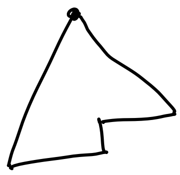
$T_{odstrani}$ $O(\log n)$

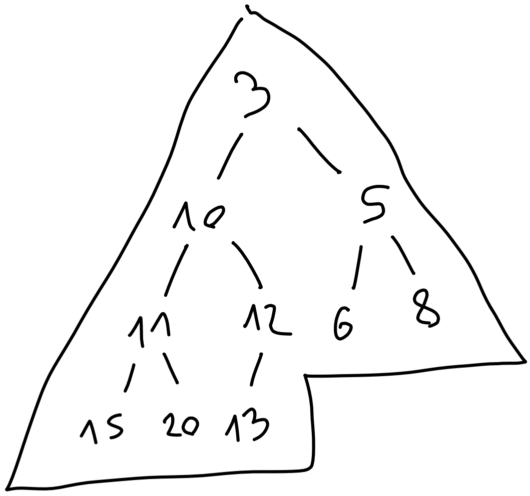
$T_{posodobi}$ $O(\log n)$

$T_{vrsta\_s\_pred}$ $O(n)$

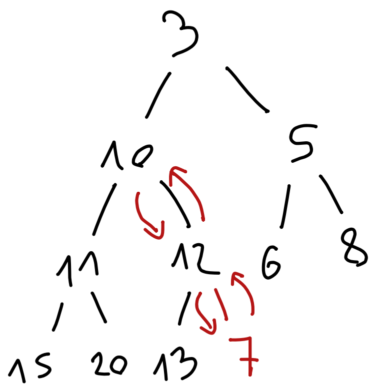
Dijkstra $O(|E| \cdot \log |V|)$

Kopica je dvojiško drevo, ki:
- je levo poravnano
- ima vsakega starša manjšega od otrok

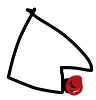




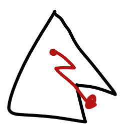
Vstavljanje



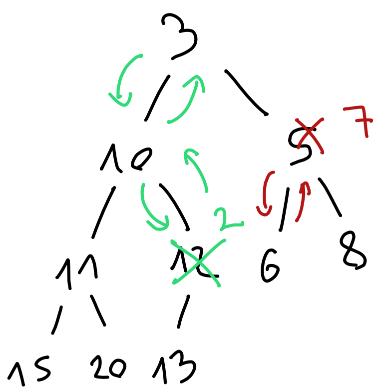
1. Novo vrednost damo na zadnje mesto



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



Posodabljanje



- Če posodobimo na manjšo vrednost, ravnamo kot prej v 2. koraku
- Če posodobimo na večjo vrednost, pa element potapljamo tako, da ga zamenjujemo z manjšim od otrok