

$$f: \mathbb{N} \rightarrow \mathbb{R}$$

$$f \in O(g). \exists M > 0. \exists N \in \mathbb{N}. \forall n > N. f(n) \leq M g(n)$$

$$\text{mnazemij } n^3 \rightsquigarrow n \log_2 n^7 \rightsquigarrow n$$

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = 0 \Rightarrow f \in O(g)$$

$$\forall \varepsilon > 0. \exists N > 0. x > N. \Rightarrow \left| \frac{f(x)}{g(x)} \right| < \varepsilon$$

$$\Rightarrow |f(x)| < \varepsilon |g(x)|$$

potem je

$$\begin{array}{c} \exists \varepsilon > 0. \exists N > 0. \forall n > N. f(n) \leq \varepsilon g(n) \Rightarrow f \in O(g) \\ \parallel \\ 1 \end{array}$$

$$O(1) \subset O(\ln(\ln(n)))$$

$$\underline{O(\log_2(n)) = O(\log_{10}(n))}$$

$$\log_2(n) = \frac{\log_{10} n}{\log_{10} 2} \in O(\log_{10}(n))$$

$$\log_{10}(n) = \frac{\log_2 n}{\log_2 10} \in O(\log_2(n))$$

kolena

- že celo poljeje (predzsem po kolajenstih)
- zdej se po kolore jenje na fady
- seširde mra obratjeu hrustalee

$$\begin{aligned} O(1) &\subset O(\log(\log n)) \subset O(\log_2(n)) = O(\log_{10}(n)) \\ &\subset O(\sqrt{n}) \subset O(\log(n!)) \subset O(n \log n) \subset O(3^{1/n}) \\ &\boxed{O(\log_2(n)) \subset O(n^\epsilon)} \quad \begin{aligned} &\subset O(n^2) \subset O(2^n) \\ &\quad \quad \quad \uparrow \\ &\quad \quad \quad \subset (2^{2 \log n}) \end{aligned} \end{aligned}$$

$$\log(n!) \leq \log(n^n) = n \log n$$

$$O(\log(\log(n))) \subset O(\log n)$$

$$\log(\log(n)) \leq \log(n) \quad \begin{array}{l} \swarrow \log(n) < n \\ \text{in log} \\ \text{inj nerazsajena} \end{array}$$

$$\log(n) \in O(\sqrt{n})$$

$$\lim_{n \rightarrow \infty} \frac{\log n}{\sqrt{n}} \stackrel{\text{L'H}}{=} \frac{1}{n} \frac{1}{\frac{1}{2}} = \frac{1}{n} \frac{1}{2} = 0 \Rightarrow \log(n) \in O(\sqrt{n})$$

$$3^{1/n(n)} = 3^{\frac{\log_3(n)}{\log_3 e}} = n^{\frac{1}{\log_3 e}} = n^\epsilon > 1$$

$$\sqrt{n} < n < n \log n$$

$$O(n \log n) = O(\log n!)$$

$$\log n! = \sum_{i=1}^n \log i$$

$$\log n! > \log \left(\frac{n}{2}\right)^{\frac{n}{2}} = \frac{n}{2}(\log n - \log 2) \approx \frac{n}{2}$$

$$\underline{\underline{n \log n < n^2}}$$

$$n \log n \leq n \cdot n \leq n^2$$

$$O(n^2) \subseteq O(2^n) \subseteq O(2^{n \log n}) \subseteq O(n!)$$

$$2^{\log n^n} = 2^{\frac{\log_2 n^n}{\log_2 10}} = n^{\log_2 10 \cdot n}$$

seznam: n

append: $O(1)^*$
delete(i) $O(n)$ (n-i)
copy: $O(n)$
update(i): $O(1)$
get(i): $O(1)$
find(a): $O(n)$
add: $O(n)$

lo nes
tipična zanim

add(i)	$O(n)$
add(o,n)	$O(n), O(1)$
delete(i)	$O(n)$
delete(o,n)	$O(n), O(1)$
get(i)	$O(1)$
search(x)	$O(n)$

lo nes
tipična zanim
gas prastar

add(i)	$O(n)$	$O(1)$
add(o,n)	$O(n), O(1)$	$O(1)$
delete(i)	$O(n)$	$O(1)$
delete(o,n)	$O(n), O(1)$	$O(1)$
get(i)	$O(1)$	$O(1)$
search(x)	$O(n)$	$O(1)$

Slavar

$O(1)$ vse

algoritem za iskanje maksimuma

$m = seznam[0]$

for c in $seznam[1:]$

if $c > m \Rightarrow m = c$

$O(n), O(1)$

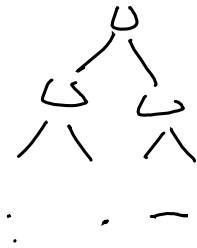
return m

$seznam.n$
 $seznam.sort$
 $seznam[-1]$

$O(n \log n)$ $O(1)$

$$\sum_{i \in \mathbb{N}}^n i = O(n^2)$$

Rekurzivno: seznam na dva dela



$\left\{ \begin{array}{l} \log n \text{ globine} \end{array} \right.$

U U U U U

čas je lahko $O(\log n)$ z pretecanjem n (precekanje)
lahko pa je tudi $O(\frac{n}{\log n})$, precekanje in je
se vedno hitreje

Bitwise operations

- bitand &
- bitor |
- bitxor ^
- bitnot ~, !
- bitshift left <<
- bitshift right >>

✓ jz, jch
 & &
 ||
 !=
 not

$$\begin{array}{r} z: 111 \\ g: 1001 \end{array}$$

$$\begin{array}{r} 111 \\ 1001 \\ \hline 1 \end{array} \&$$

$$\begin{array}{r} 0111 \\ 1001 \\ \hline 1111 \end{array} |$$

$$\wedge \begin{array}{r} 111 \\ 1001 \\ \hline 1110 \end{array}$$

$$! \begin{array}{r} 111 \\ 0 \\ \hline \end{array}$$

$$! \begin{array}{r} 1001 \\ 110 \end{array}$$

$$<< \begin{array}{r} 111 \\ 1110 \end{array}$$

$$<< \begin{array}{r} 1001 \\ 10010 \end{array}$$

$$g \gg 2 \begin{array}{r} 1001 \\ 10 \end{array}$$

$$a \oplus b \oplus b = a$$

a \ b	0	1
0	0 ⊕ 0 = 0	0 ⊕ 1 = 1
1	1 ⊕ 0 = 1	1 ⊕ 1 = 0

a...bit

$$a_n \dots a_2 a_1 a_0$$

$$\& \frac{[0 \dots 010 \dots 000]}{0 \dots a_i \dots 0} \leftarrow \text{mask}$$

if (a & mask)
 > 0 zero True

$$\text{mask} = 1 \ll i$$

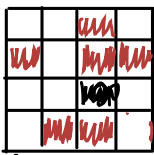
$$\begin{array}{l} 1.) [0 \dots 0] \\ \sim [1 \dots 1] \\ \hline \gg n-i [0 \dots 1 \dots 1] \end{array} \quad \begin{array}{l} n(n0) \ll i) \\ (1 \sim i) - 1 \end{array}$$

funkcija sprejme a, i

$$a_n \dots a_1 a_0 \rightarrow a_n \dots 1 \dots a_1 a_0$$

def $f(a, i)$:
 mask := $1 \ll i$
 return $a | \text{mask}$

def $f'(a, i)$
 mask := $\sim(1 \ll i)$
 return $a \& \text{mask}$



$$n = 0010; 1011; 0000; 0110; 1010$$

$\nwarrow$ 10 mesto $\searrow$

levo: $n-1$
 desno: $n+1$
 gor: $n-u$
 dol: $n+u$

def levoprij(a)
 mask = $(1 \ll 4) - 1$
 $p = a \& \text{mask}$
 mask = $\sim(1 \ll 20 - p_{10} + 1)$
 return $(a \& \text{mask}) - 1$

$\nwarrow$ $a = a \& \text{mask}$
 $\nwarrow$ $a = (a \gg 4) \ll 4$
 return $a | p$

def fib(n):

prv = 1

drv = 1

for i in range(n)

drv = drv'

drv += prv

prv = drv'

return drv

$$O(drv) = 1,6^i = 1,6^n$$

^
value

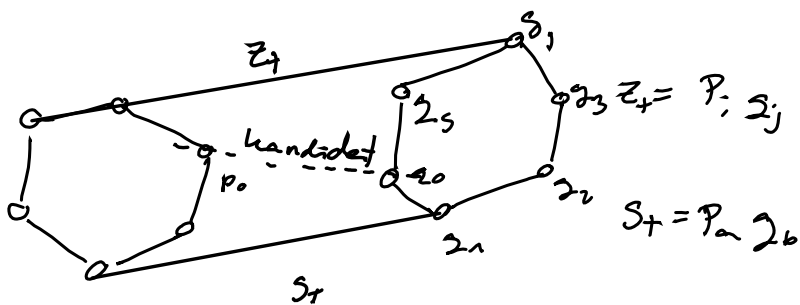
$$\text{31 bitov za } drv = \log_2 1,6^i$$
$$= O(i)$$

$$\text{assign} = O(n)$$

$$\sum v_1 + v_2 + v_3 = \sum_{i=1}^n 3O(i) = O(n^2)$$

TRIE - prefix tree

Y España

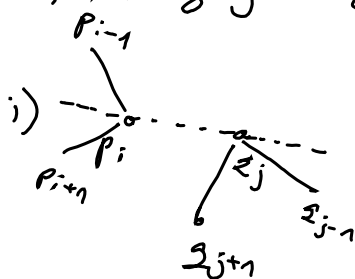


Združena dvojica bo naša nova

Kako do Z_+

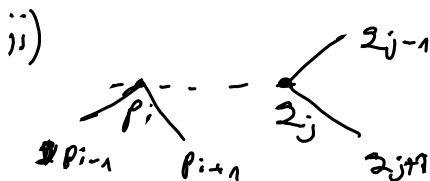
$$Z_+ = p_0 z_0$$

while z_+ ni razgraja tangente

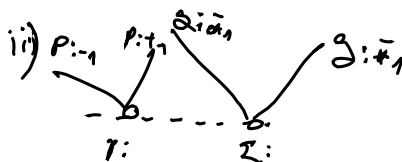


V tem primeru nastavimo

$$p_i = p_{i-1} \quad (i = i-1)$$



$$j = j+1$$



to je opozarja tangente
divnjeno obe
 $i = i+1 \quad j = j+1$

Bla bla bla

če pozicija ni - - - - - naslednji + = 1

Predstave grafu

- Matrika sosednjosti $A = [a_{ij}]$; $a_{ij} = \begin{cases} 1; & i \text{ in } j \text{ povezana} \\ 0; & \text{sicer} \end{cases}$
- slovar $\{ \text{vozlišče} : [\text{povezave do vozlišča}] \}$

$\{ v : [u \text{ for } u \text{ in next}(v)] \text{ for } v \text{ in } G \}$

opomba: če vozlišča označimo od 0 do $n-1$, namesto slovarja uporabimo seznam

ideja DFS:

1. izberemo začetno vozlišče in ga shranimo v obiskane
2. izberemo sosedo, ki še ni bil obiskana. odpremo obiskane
3. preiskujemo se v sosedo

ideja BFS:

1. izberemo začetno vozlišče in ga damo v obiskane
2. obiščemo vse kake sosedo in jih damo v obiskane, ter naredimo seznam vseh sosedov v sosedov
3. obiščemo vse sosedbe sosedov isto kot 2.

from collections import deque ← double ended que

def DFS(G, s):

n = len(G)

obiskeni = [False]*n

sklad = deque([s])

while sklad:

v = sklad.pop()

if not obiskeni[v] ← standardna struktura

for u in G[v]:

if obiskeni[u] == False:

sklad.add(u)
obiskeni[v] = True

def BFS(G, s):

n = len(G)

obiskeni = [False]*n

vrsta = deque([s])

while vrsta:

v = vrsta.popleft()

if not obiskeni[v]

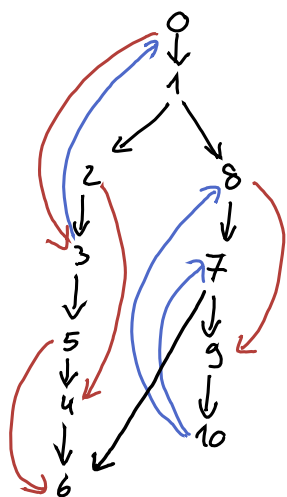
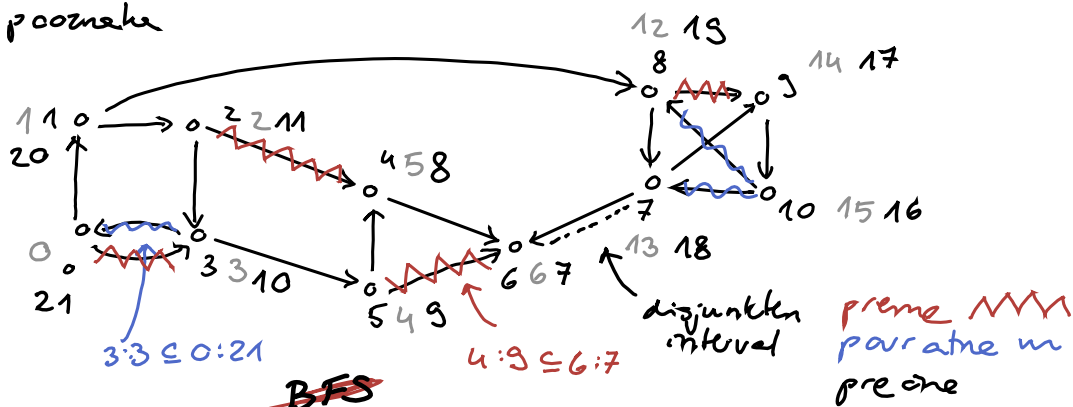
for u in G[v]:

if obiskeni[u] == False:

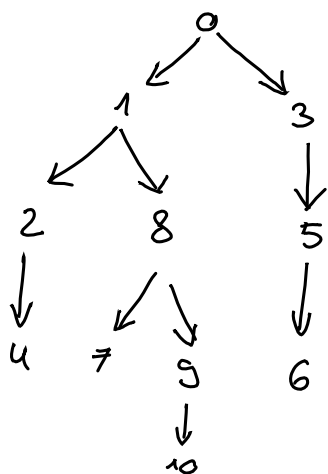
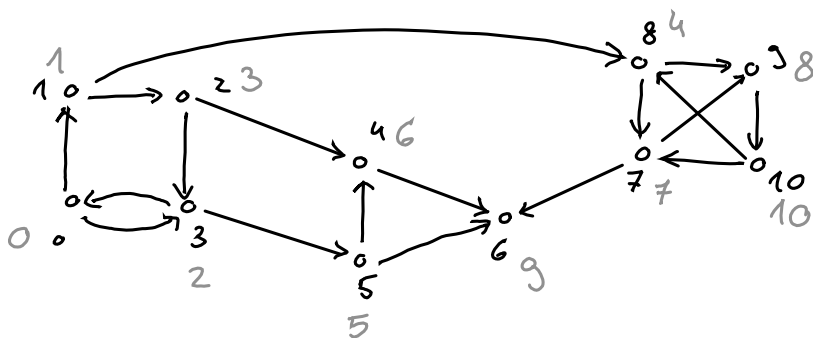
vrsta.add(u)
obiskeni[v] = True

-predoznaka
-poznahe

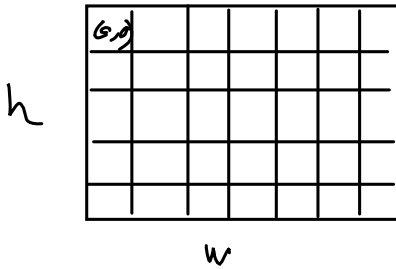
DFS



BFS



Razvij pristop za delo z grafi, ki so mreže



vozlišč = $h \cdot w$ (celice)

4-sosednost

$(i,j) \sim (i \pm 1, j)$

$(i,j) \sim (i, j \pm 1)$

smeri = $[(1,0), (-1,0), (0,1), (0,-1)]$

def get-sosed(i,j):

return $[(i+a, j+b)$ for (a,b) in smeri]

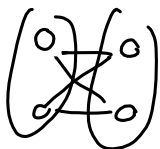
G ne ustrejen graf

Algoritem Da če graf dvodelen, ne sicer

$V(G) = A \cup B$ -biparticijska nedwodela, tako da
med

$\Leftrightarrow$

kekolahko pobarvamo $V(G)$ tako, da sosedstva vedno
dobijo drugo barvo.



```
from collections import deque
```

```
def je_dvodelen(G):
```

```
    n = len(G)
```

```
    barve = [None] * 2  ← = obiskani
```

```
    vrsta = deque([0])
```

```
    barve[0] = 0
```

```
    while not vrsta.empty():
```

```
        u = vrsta.popleft()
```

```
        trb = barve[u]
```

```
        for v in G[u]:
```

```
            if barve[v] == None
```

```
                vrsta.push(v)
```

```
                barve[v] = trb % 2
```

```
            elif barve[v] == trb
```

```
                return False
```

```
    return True
```

```
def pot_do_najk(G):  
    n = len(G)  
    razdelje = [None]*n  
    pred = [None]*n  
    razdelje[0] = 0  
    vrsta.add(0)  
    while not vrsta.empty:
```

3 Mreža 4×4 7 rdeči 8 modri polji 1 prazen

	W	B	B
W	W	B	B
W	W	B	B
W	W	B	B

W	W	B	B
W	W	B	B
W		B	B
W	W	B	B

Zanima nas najmanjše število poljev ^{iz A v B} ~~3~~
pojudni pozicija

Vsaka konfiguracija je svoje vodilo

pozicija bele 4 bit
rdeče in modre 16 bitov

Sosed:

pozicija bele $(\pm 1 \vee \pm 4)$ ^{razen na robu} $\wedge$ pa tudi ista barva
razen v pozicijah bele, tam pa mora biti ista barva

V

28.11

Wen: pamemboro

Dijkstrov algoritem

5. 12

Vhod: G usmerjen/neusmerjen graf,

povezavno utežen, uteži so nenegativne,
 $w: E(G) \rightarrow \mathbb{R}_+$

za dano vozlišče s (opazite še konano vozlišče)

izhod $\Rightarrow$ razdelje najkrajše poti od s do vseh ostalih vozlišč

- za dano v s $d(s) = 0$
- Pregledamo sosedo od s in izberemo tistega, ki je "najbližji" $d(u) = u(w)$
- Ostale sosedo damo na prioriteto vrsto v obliki $(d(s) + w(v), v)$
- nato iz vrste

za dvojisko kopico

$O(V) + O(E \log E)$ (zahtevnost dodajanja + zahtevnost brisanja)

$O(1)$ $O(E)$

koliko jih je ne prioriteten vrst

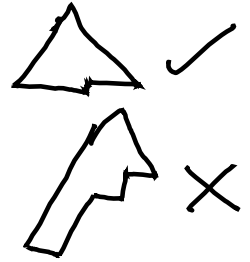
PQ seznam

$$= O(V + E \log E) = O(E \log E + V)$$

Lahko tudi $O(V^2)$

Dvojiška kopica

- levo poravnano dvojiško drevo
- lastnost kopice: vrednost vsakega starša $\leq$ min vrednosti obeh otrok



V korenu je najmanjši element, ampak to ni iskalno drevo

Osnovne operacije

dodaj (X)

brisi (izbriše minimum)

dodaj:

dodamo ga na zadnje mesto in dvigujemo gor, dokler ne zadošča lastnosti kopice

brisi:

vzamemo koren in ga izbrišemo.

Najbol spodnji desni element prestavimo v koren in potem ga bublamo dol



V 12.12

Bellman-ford

vhod: graf G (usmerjen, utežen), zač. vozlišče

izhod: seznam d_z
dolžinam: najkrajših
poti od s do ostalih
vozlišč.

↳ dovolimo
negativne povezave
↳ ne dovolimo
negativnih ciklov

Ideja: $(n-1)$ -krat relaksiramo vsako povezavo

def $f(G, s)$:

$d = [\infty]^* n$ $n, \dots$ št vozlišč

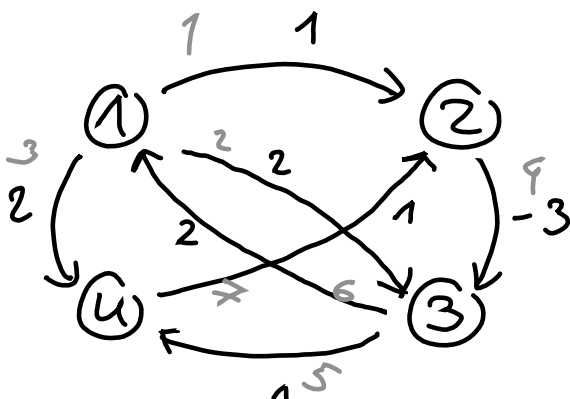
$d[s] = 0$

for $_$ in range($n-1$):

for (u, v, w) in $E(G)$:

if $d[v] > d[u] + w$:
 $d[v] = d[u] + w$ } relaksacija
povezave

return d



	0	1	2	3	4
1	0	0	0	0	0
2	∞	1	1	1	0
3	∞	2	-2	-2	-2
4	∞	2	2	-1	-1

↑ kapa tem vemo, da
 je nekje negativen cikel
 Tu bi se
 moralo končati

2) Floyd Warshallov algoritem

a) $D_k(i, j)$ dolžina najkrajše poti od i do j ,
ki vsebuje le vozlišča $1, \dots, k$
notranja

$$D_0(i, j) = \begin{cases} w(i, j) & ; \exists (i, j, w) \\ \infty & ; \text{sicer} \end{cases}$$

$$D_k(i, j) = \min \{ D_{k-1}(i, j), D_{k-1}(i, k) + D_{k-1}(k, j) \}$$

$D_n(i, j)$ je najkrajša pot

če je $D_n(i, j) < 0$ potem Graf vsebuje
negativen cikel

čas je $O(n^3)$ n...štev vozlišč

Velika igra z odprtim svetom ima na različnih lokacijah tržnice, kjer lahko kupujemo in prodajamo surovine
 npr.: 10 železa za 1000 leze

Ker se tržnice nahajajo na različnih delih sveta, kjer so precejšnje razdalje, različne so lahko menjalni tečaji surovine.

Surovine x bi radi zamenjali za oim več surovine y . Kateri: $\rightarrow$ vredni?

Vozlišča so vse možne surovine

$$x_1 \rightarrow x_2 \Leftrightarrow \exists T. T(x_1, x_2) \quad \begin{array}{l} T_0 \dots \text{tržnica} \\ T_1 \dots \text{mrežica tržnic} \end{array}$$

$$w(x_1, x_2) = \max_{T \in T} \sum_i s(x_1, x_2) \text{ menjalni tečaj}$$

izračunajmo pot od x_1 v y kjer je produkt uteži na povezavi čim večji

$$w(x_1, x_2) \rightarrow -\log(w(x_1, x_2))$$

Problem minimalnega vpetege drevesa

$G(V, E)$ utežen graf (neusmerjen)

w_e utež

Iščemo vpeto drevo T , da je vsota $\sum_{e \in T} w_e$ najmanjša možna

Kruskalov algoritem

$[w_{e_1} \dots w_{e_n}] \dots$ sortirane povezave

Začenemo z gozdom kjer je vsaka krajšica svoj drevo

Česta krajšici povezave e_i iz različnih dreves to povezavo dodamo v gozd in s tem združimo te dve drevesi; če sta iz istega drevesa preoblikujemo povezavo

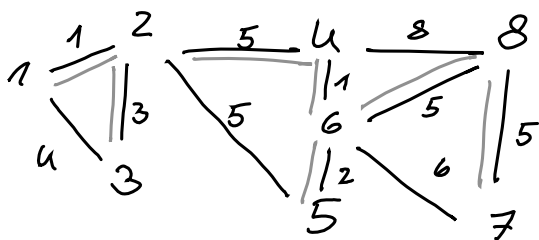
Unija z rangem ... podobna struktura, ki

učinkovito opravlja te operacije

$\text{naredi}(x) \dots$ naredi novo množico z elementom x

$\text{najdi}(x) \dots$ vrne množico v kateri se nahaja x
Vsako množico predstavi z enim predstavnikom

$\text{združi}(x, y) \dots$ naredi disjunktno unijo $\text{najdi}(x)$ in $\text{najdi}(y)$



$(1, (1, 2)), (1, (4, 6)), (2, (5, 6)), (3, (2, 3)), (4, (1, 3)),$
 $(5, (2, 4)), (5, (6, 8)), (5, (7, 8)), (5, (2, 5)), (6, (6, 7))$

ČZ Kruskal: $O(m \log m) + m O(\log n) + n O(\log n)$
 m najdi

m ... povezeve $|E|$

n ... vozlišča $|V|$

Kompresija poti

ko na koncu pokličemo najdi 5 ugotovimo
 da $\text{najdi}(5) \neq \text{starš}(5)$

Potem se splača za $\text{starš}(5)$ nastaviti $\text{najdi}(5)$

1. izp-it 2324

Razdelimo seznam na 3 dele.

Naj pogostejši element mera Liti najpogostejši
v vsaj dveh od treh delov. Če je tudi
tudi polovica v enem, vedno v enem
bo gotovo najpogostejši

del seznama (A)

$$\left. \begin{aligned} A_1 &= A[0 : \frac{n}{3}] \\ A_2 &= A[\frac{n}{3} : \frac{2n}{3}] \\ A_3 &= A[\frac{2n}{3} :] \end{aligned} \right\} O(1)$$

3. $T(\frac{n}{3})$ $mb_i = [\text{seznamaj}(A_i) \text{ for } i \in \{1, 2, 3\} \text{ if not false}]$

if $mb_i = []$: return False

$a = mb_i[0]$

if $a \in mb_i[1:2]$: return a

$O(\frac{2n}{3})$ if $(A_2 \cup A_3).count(a) \geq \frac{n}{6}$: return True
 $= O(n)$

$a=3$

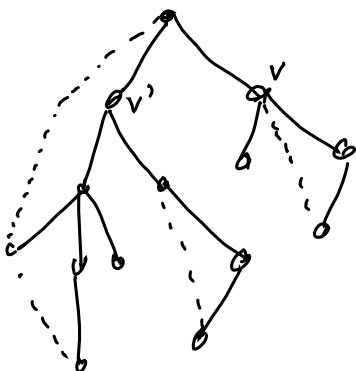
$b=3$

$d=1$

$$b^d = 3 \Rightarrow O(n^1 \log(n))$$

po krovnem izreku

$$\text{iz potenca} = 3 \quad d=1 \quad n^{\log_3(3)} = O(n)$$



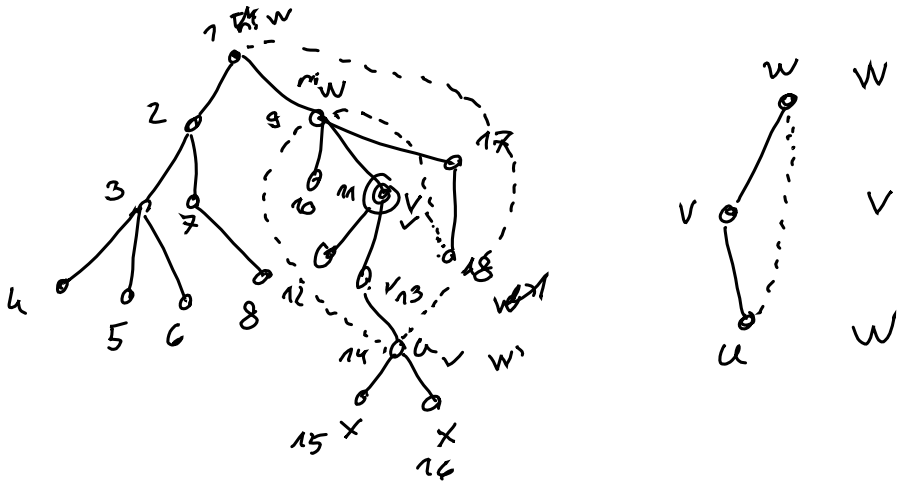
⇒ Rečimo da je v prerezno

Rečimo da za vsakega drugega človeka
pomeni, ki je pa era na prednikih od v

Potem $\exists$ pot iz korena do vsakega pomena

v in ne gre čez v , ker lahko preidemo
do vsakega drugega od v če v ni prerezno

⇐ Rečimo da $\exists$ drake katerega pomeni n. k. k. c.
pomeni nekaj potan je d. m. n. o. d. n. l. e.
pridemo da s t. l. da gre čez v to
je prerezno

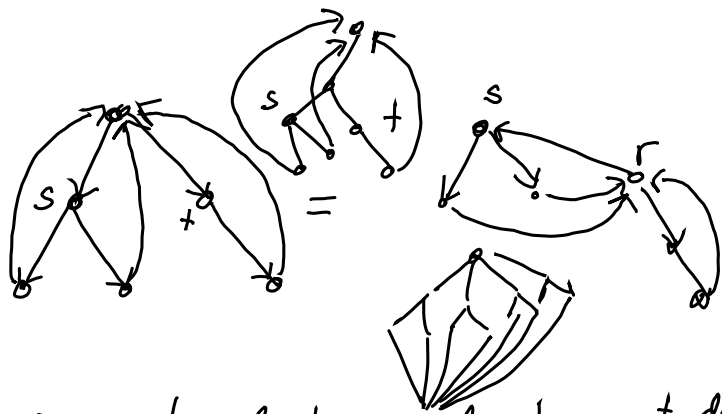


Vozlišča so urejena po pred od največ. do
najmanjšega

Gremo po povezavi največjega vozlišča

$$low(u) = \min(\text{preu}(\text{parčne povezave}))$$

for



Nejkrajša pot od korena do t = pot do t

t potomec $s \Rightarrow$ pot do t - pot do s

t ne drugem drevesu $\Rightarrow$ pot do s + pot do t

Če vsako vozlišče izračunamo najkrajšo razdaljo od korena. Ker je to drevo je to lahko narediti v $O(V)$

Zraven se gledamo če t potomec s .

če je, lahko zapišemo kje najdemo in vremo $w(t) - w(s)$

Drugače bomo morali pot do korena
biti in potem od s
do t

to lahko naredimo z dts di to b

v $O(V)$

potem moramo še upoštevati da je pot do t

Spet lahko uporabimo # parov je največ

$2 \cdot (n-2) + (n-1) \in O(n)$

potem pot od r do t

je $O(n)$ kaj lahko bolj

Majhne uteži?

če je najmanjša pot pot

možna je

možna je

dijkstra dela vedno v

$O(V \log V)$ $\log(V)$ t. v. de priredila niste
ker je $O(n \log n)$

če ni najmanjša celotna vedno da
boste vedno morali dati.

če sta t potomec s je enačba

če pa ni, pa niste ito pot do korena
teraj r

druga kot vprašanje

že no afekto

R \ C	1	3	0	2	1
1	1	0	0	0	0
2	0	1	0	1	0
3	0	1	0	1	1
1	0	1	0	0	0
0	0	0	0	0	0

	0	0	0	1	0
0	1	1	1	2	1
1					1
0					
0					
0					
5	1	1	1	1	1

for r in R
for c

	3	1	1	1	0
3	1	1	1		
2	1	1	1	1	
1	1	1	1		
0					

medans R in C (zganime si
permute) to je bolje v $O(n \log n)$
če dajemo permutirane matrice
količke (vse iz algebre 1)
počrtno polno matriko pti km
unverzalno kake je stopenj stolpca
in pride pri dajanju re, gledamo najprej
zame se nato ho $\begin{bmatrix} i \\ c \\ i \end{bmatrix}$ c ne zide se
če pride ne ≤ -1

$$\text{cost} = 2 \cdot n \log n + O(n \cdot n) = n^2 + \text{agstrukcija}$$

prilagodit

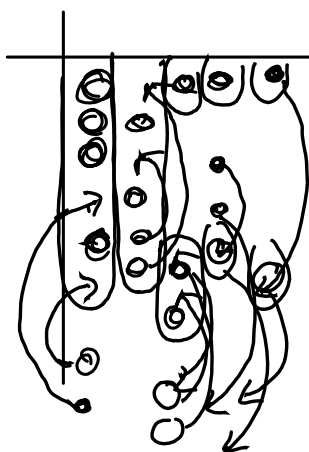
pravilnost? npr me $[i, j]$ je to $w(\pi(i) \pi(j))$

če nepravilno gleda dajajo

tu hako vire false tudi

Take me se na neki vrstici pride
smeri bre zel bi videl vse sve

brezno da oddeja kake pritrzja
tu jo ter meče
plem jo kake breake
p. & kake jira me



tu najprej
odpre
vrstica.

