

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

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

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

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

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

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

potem je

$$\underbrace{\exists \varepsilon > 0. \exists N > 0. \forall n > N. f(n) \leq \varepsilon g(n)}_1 \Rightarrow f \in O(g)$$

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

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

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

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

kolena

- že celo poljeje predstavi po kolonijemski
- zdej že po koloni jenz ne bdej
- seširne ima obnabn hrustace

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

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

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

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

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

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

$$3^{\ln(n)} = 3^{\frac{\log_2(n)}{\log_2 e}} = n^{\frac{1}{\log_2 e}} = n^\epsilon > 1$$

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

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

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

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

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

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

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

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

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: $O(n)$

le nes
tipično zenim

| | |
|-------------|--------------|
| 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) | $O(n)$ |

le nes
tipično zenim gas pro star

| | | |
|-------------|--------------|--------|
| 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) | $O(n)$ | $O(1)$ |

Slover

$O(1)$ vse

algoritem za iskanje maksimuma

$m = seznam[0]$

for c in $seznam[1:]$

if $c > m \Rightarrow m = c$

$O(n), O(1)$

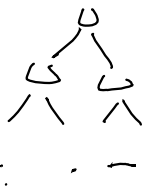
return m

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

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

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

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 pretekom n preizjav
lahko pa je tudi $O(\frac{n}{\log n})$, preizjav in je
se vedno hitreje

Bitwise operation

- bitand &
- bitor |
- bitxor ^
- bitnot ~, !
- bitshiftleft <<
- bitshiftright >>

✓ justoh
 &
 ||
 !=
 not

$$z: 111$$

$$g: 1001$$

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

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

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

$$! \begin{array}{r} 111 \\ 0 \end{array} , \begin{array}{r} 1001 \\ 110 \end{array}$$

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

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

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

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

| | | |
|-------|------------|------------|
| a \ b | 0 | 1 |
| 0 | 000 = 0 | 101 = 0 |
| 1 | 100 = 1 | 001 = 1 |

a...bit

$$a_n \dots a_2 a_1 a_0$$

$$\& \frac{[0 \dots 010 \dots 0.0.0.0]}{0 \dots a_i \dots 0} \leftarrow \text{mask}$$

if (a & mask)
 70 zero True

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

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

funkcija sprejme a, i

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

def $f(a, i)$:

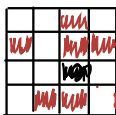
mask := $1 \ll i$

return $a | \text{mask}$

def $f'(a, i)$

mask := $\sim(1 \ll i)$

return $a \& \text{mask}$



$n = 0010; 1011; 000; 0110; 1010$

10 mask

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

def $\text{levo} \& \text{gor}(a)$

mask = $(1 \ll 4) - 1$

$p = a \& \text{mask}$

mask = $\sim(1 \ll 20 - p_{10} + 1)$

return $(a \& \text{mask}) - 1$

$\rightarrow a = a \& \text{mask}$
 $\rightarrow a = (a \gg u) \ll u$
 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

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

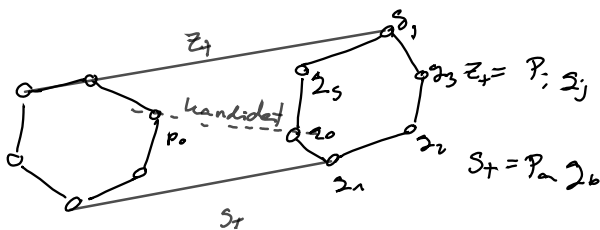
$$= O(i)$$

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

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

TRIE - prefix tree

Y España

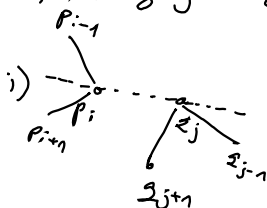


Zdravica avajnica ba neba traba

Kako do Z_f

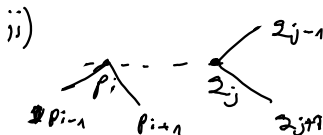
$$z_j = p_0 z_0$$

while z_i is always tangent

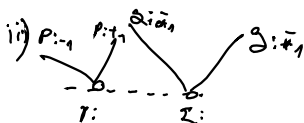


V tem primeru ne delimo

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



$$j = j + 1$$



to je opostrežena tangenta,
divergencija oba
 $i \neq 1$ $j \neq 1$

Be be be

če pozicija ni  naredis $+1$

Predstave grafu

- Matrike 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. iščemo sosedo, ki še ni bil obiskana. ozaremo obiskane
3. peneturamo se v sosedo

ideja BFS:

1. izberemo začetno vozlišče in ga damo v obiskane
2. obiščemo vse lege sosedo in jih damo v obiskane, ter naredimo seznam vseh sosedo v sosedov
3. obiščemo vse sosedo sosedov istokot 2.

from collections import deque ← double ended queue

def DFS(G, s):

n = len(G)

obiskani = [False]*n

sklad = deque([s])

while sklad:

v = sklad.pop()

if not obiskani[v] ← standardna struktura

for u in G[v]:

if obiskani[u] == False:

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

def BFS(G, s):

n = len(G)

obiskani = [False]*n

rrsta = deque([s])

while rrsta:

v = rrsta.popleft()

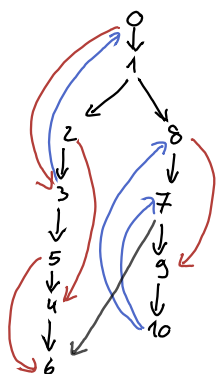
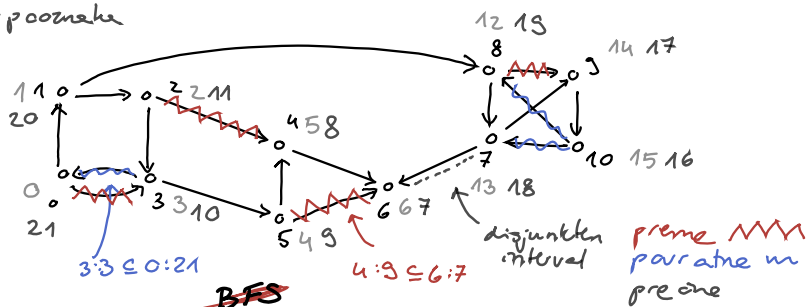
if not obiskani[v]

for u in G[v]:

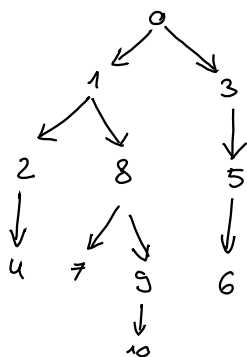
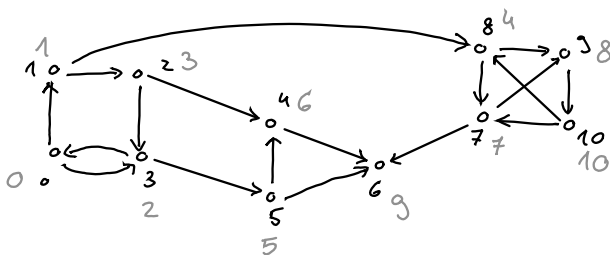
if obiskani[u] == False:

rrsta.add(u)
obiskani[v] = True

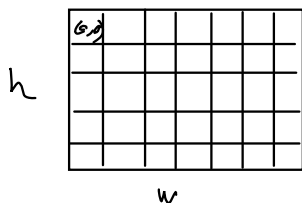
DFS



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]

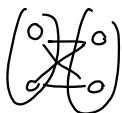
G ne ustreja graf

Algoritem Da če graf dvodelen, ne sicer

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

\Leftrightarrow

kakor koli pobarvamo $V(G)$ tako, da sosednja 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)
    razdalje = [None]*n
    pred = [None]*n
    razdalje[0] = 0
    vrsta.add(0)
    while not vrsta.empty
```

3. Mreža 4×4 rdeči in 8 modri poziciji 1 peon

| | | | |
|---|---|---|---|
| | 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 potez ^{iz A v B} ~~iz A v B~~
pojdni pozicija

Vsaka konfiguracija je svojo vozlišče

pozicija bele rdeči in modre
 6 bit 10 bitov

Sosed:

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

V

28.11

Wen: pamembao

Dijkstra algoritem

5. 12

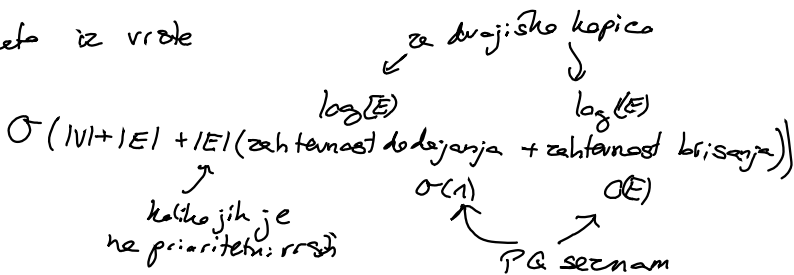
vhod: G usmerjen/neusmerjen graf,

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

za določeno vozlišče s (opažajo še končno vozlišče)

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

- začnemo 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



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

Lahko tudi $O(V^2)$

Dvojiška kopica

- levo poravnano dvojiško drevo



- lastnost kopice: vrednost

vsakega starša \leq min vrednosti
oben otrok



V korenu je najmanjši element, ampak
to ni iskalno drevo

Osnovne operacije

dodaj (X)

brši (izbriše minimum)

dodaj:

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

brši:

Vzamemo koren in ga izbrisemo.

Najbol spodnji desni element prestavimo

v koren in potem ga bublamo dol

