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

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

množenje $n^3 \leadsto n^{\log_2 7} \leadsto n$

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

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

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

potem je

$\exists \underset{1}{\underset{\parallel}{\varepsilon}} > 0.\ \exists N > 0.\ \forall n > N.\ f(n) \leqslant \varepsilon g(n) \Rightarrow f \in O(g)$

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

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

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

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

kolena

- že celo poljedje (predvsem po kolesijenju)
- zdej se po koleni jemlje na fotelj
- sedaj ima obrabljen hrustanec

$$O(1) \subset O(\log(\log n)) \subset O(\log_2(n)) = O(\log_{10}(n))$$

$$\subset O(\sqrt{n}) \subseteq O(\log(n!)) \subseteq O(n \log n) \subseteq O(3^{\ln n})$$

$$\boxed{O(\log_a(n)) \subset O(n^{\varepsilon})}$$

$$\subseteq O(n^2) \subseteq O(2^n)$$

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

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

$$O(\log(\log(n))) \subseteq O(\log n)$$

$\log(n) < n$ in log inj naraščajoča

$$\log(\log(n)) \leq \log(n)$$

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

$$\lim_{n\to\infty} \frac{\log n}{\sqrt{n}} \overset{LH}{=} \frac{\frac{1}{\sqrt{n}}}{n\,\frac{1}{2}} = \frac{1}{\ln 2\sqrt{n^3}} = 0 \Rightarrow \log(n) \in O(\sqrt{n})$$

$$3^{\ln(n)} = 3^{\frac{\log_3(n)}{\log_3 e}} = = n^{\frac{1}{\log_3 e}} = n^{\varepsilon > 1}$$

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

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

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

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

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

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

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

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

seznam: n

append: $O(1)^*$

delite(i) $O(n)$ $(n-i)$

copy : $O(n)$

uppdete(i) : $O(1)$

get(i) : $O(1)$

find(a) : $O(n)$

add : $O(n)$

ko nes tipična zenim

add(i)	$O(n)$
add(0,n)	$O(n)O(1)$
delete(i)	$O(n)$
delet(0,1)	$O(n), O(1)$
get(i)	$O(1)$
search(x)	$O(n)$

ko nes tipična zenim	čas	prostor
add(i)	$O(n)$	$O(1)$
add(0,n)	$O(n)O(1)$	$O(1)$
delete(i)	$O(n)$	$O(1)$
delet(0,1)	$O(n), O(1)$	$O(1)$
get(i)	$O(1)$	$O(1)$
search(x)	$O(n)$	$O(1)$

Slovar

$O(1)$ vse

algoritem za iskanje maximuma

```
m= seznam[0]
for c in seznam[1:]
    if c>m => m=c
return m
```

$O(n), O(1)$

---

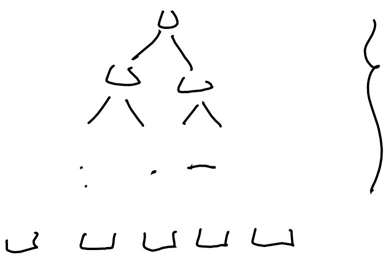
seznam n

seznam.sort $O(n \log n)$ $O(1)$

seznam[-1]

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

Rekurzivno: seznam na dva dela



log n globina

čas je lahko $O(\log n)$ z procesorjem n (procesorjev)

lahko pa je tud $O(\frac{n}{\log n})$ procesorja in je

še vedno hitreje

Bitne operacije

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

v jezikih
&&
||
!=
not

z: 111
g: 1001

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

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

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

$$!\;\begin{array}{r} 111 \\ \hline 0 \end{array} \qquad !\;\begin{array}{r} 1001 \\ \hline 110 \end{array}$$

$$<<\;\begin{array}{r} 111 \\ \hline 1110 \end{array} \qquad <<\;\begin{array}{r} 1001 \\ \hline 10010 \end{array} \qquad g>>2\;\begin{array}{r} 1001 \\ \hline 10 \end{array}$$

$a \oplus b \oplus b = a$

a\b	0	1
0	$0 \oplus 0 = 0$	$1 \oplus 1 = 0$
1	$1 \oplus 0 = 1$	$0 \oplus 1 = 1$

a...bit

$$
\begin{array}{ll}
 & a_n \ldots\ldots a_2 a_1 a_0 \\
\& & [0\ldots 01\ 0\ldots 0.0.0] \leftarrow \text{maske} \\
\hline
 & 0\ldots\ldots a_i \ldots\ldots 0
\end{array}
$$

if $(\underbrace{a \,\&\, \text{mask}}_{>0 \text{ zero True}})$

maske $= 1 \ll i$

$$
\begin{array}{ll}
1.) & [0\ldots\ldots 0] \\
\hline
\sim & [1\ldots\ldots 1] \\
\hline
\gg n-i & [0\ldots\ldots 1\ldots 1]
\end{array}
$$

$\sim((\sim 0) \ll i)$

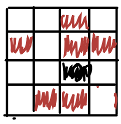
$(1 \sim i) - 1$

funkcija sprejme a, i

$a_n \dots a_1 0 \rightarrow a_n \dots 1 \dots a_1 a_0$

```
def f(a,i):
    mask := 1<<i
    return a | mask

def f'(a,i)
    mask := ~(1<<i)
    return a & mask
```



10 mesta

n: 0010 : 1011 : 0000 : 0110 : 1010

levo: n-1
desno: n+1
gor: n-4
dol: n+4

```
def levopogoj(a)
    maskp = (1 << 4) - 1
    p = a & maskp
    mask = ~(1 << 20 - p_10 + 1)
    return (a & mask) - 1
```

```
a = a & mask
a = (a>>4)<<4
return a | p
```

```
def fib(n):
   prv = 1
   drv = 1
   for i in range(n)
       drv' = drv
       drv += prv
       prv = drv'
   return drv
```

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

veikost

št bitov za drv $= \log_2 1{,}6^i$

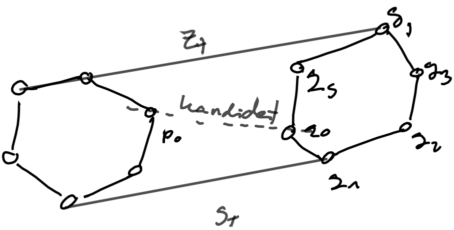
$= O(i)$

assign $= O(n)$

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

TRIE - prefix tree

V españa



$z_t = p_i\, q_j$

$s_t = p_a\, q_b$

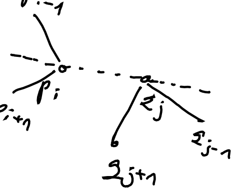
Združena ovojnica bo nek tisto

Kako do z_t

$z_t = p_0 q_0$

while z_t ni zgornja tangenta

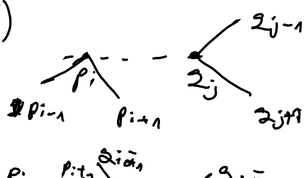
i)



V tem primeru nastavimo

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

ii)



$j = j+1$

iii)

Če je spodnja tangenta, dvignemo oba

$i += 1$ $j += 1$

Bla bla bla

Če pozicija ni

naredi $t += 1$

Predstave grafov

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

{ v: [u for u in next(v)] for v 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 soseda, ki še ni bil obiskan. [illegible]
3. ponovimo se v sosedih

ideja BFS:

1. izberemo začetno vozlišče in ga damo v obiskane
2. obiščemo vsakega soseda in jih damo v obiskane,
   ter naredimo seznam vseh sosedov sosedov
3. obiščemo vse sosede sosedov ...... isto kot 2.

```
from collections impart deque
```

dubble ended que

$O(1)$ add(x) (da na desno)

$O(1)$ pop (x) (vzemi desno)

$O(1)$ popleft(x) (vzemi levo)

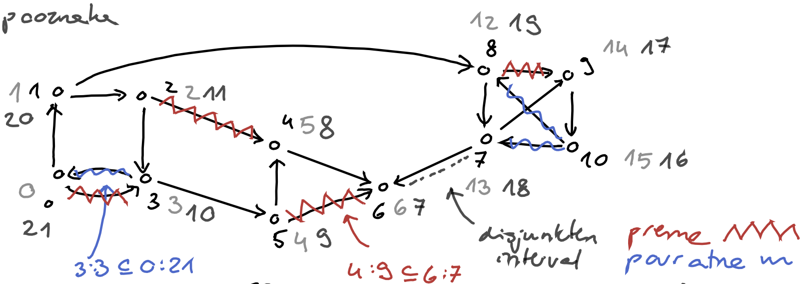
```
def DFS(G,s):
    n = len(G)
    obiskani = [False]*n
    sklad = deque([s])
    while sklad:
        v = sklad.pop()
        if not obiskani[v]
            for u in G[v]:
                if obiskani[u] == False:
                    sklad.add(u)
            obiskani[v] = True
```

standardna struktura

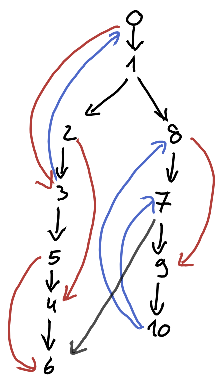
```
def BFS(G,s):
    n = len(G)
    obiskani = [False]*n
    vrsta = deque([s])
    while vrsta:
        v = vrsta.popleft()
        if not obiskani[v]
            for u in G[v]:
                if obiskani[u] == False:
                    vrsta.add(u)
            obiskani[v] = True
```

- Predoznaka
- poznaka

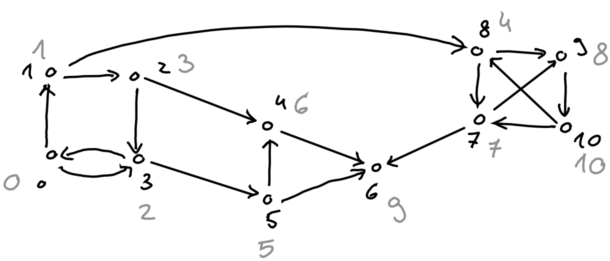
DFS

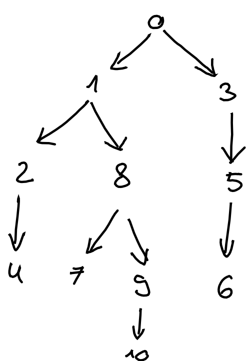


~~BFS~~



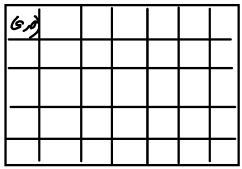
BFS





Razvij pristop za delo z grafi, ki so mreže

h



w

vozlišč = $h \cdot w$ (celice)

4-sosednost

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

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

```
smeri = [(1,0), (-1,0), (0,1), (0,-1)]
def get-sosedi(i,j):
    return [(i+a, j+b)    for (a,b) in smeri]
```

G neusmerjen graf

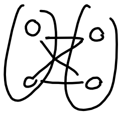
Algoritem ~~Da~~ če graf dvodeln, ne sicer

$V(G) = A \sqcup B$ -bipartcija na dva dela, tako da

med

$\Longleftrightarrow$

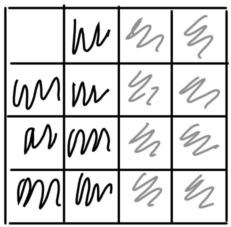
kako lahko pobarvamo $V(G)$ tako, da vozlišča vedno dobijo drugo barvo.

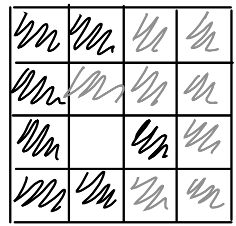


```
from collections import deque
def je_dvodeln(G)
    n = len(G)
    bare = [None]*2        = obiskani
    vrsta = deque([S])
    barva[0] = 0
    while not vrsta.empty:
        u = vrsta.popleft()
        trb = barva[u]
        for v in G[u]
            if barva[v] == None
                vrsta.push(v)
                barva[v] = trb  %2
            elif barva[v] == trb
                return False
    return True
```

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

3 Mreža 4x4 7 rdečih 8 modrimi polji 1 prazna





Zanima nas najmanjše število potez iz A v B

poljubni pozicija

Vsaka konfiguracija je svoje vozlišče

$\underbrace{}_{\text{pozicija bele}}$ $\underbrace{}_{\text{rdeče in modre}}$

4 biti 16 bitov

Sosedi:

razen na robu

pozicija bele $(\pm 1 \lor \pm 4) \land$ povsod ista kot

razen v pozicijah bele, tam pa mora biti ista barva

Vf 28.11

Zveni pomembno

5. 12

Dijkstrov algoritem

vhod: G usmerjen/neusmerjen graf,
povezavno utežen, uteži so nenegativne,
$w: E(G) \to \mathbb{R}_+$
začetno vozlišče s (lahko še končno vozlišče)

izhod ⇒ razdalja najkrajše poti od s do vseh ostalih vozlišč

- začnemo v s $d(s)=0$
- Pogledamo sosede od s in izberemo tistega, ki je „najbližji" $d(u)=w(u)$
- ostale sosede damo na prioritetno vrsto v obliki $(d(s)+w(v), v)$
- vzamemo iz vrste

za dvojiško kopico

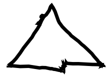
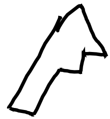
$$O(|V|+|E| + \underbrace{|E|}_{\text{koliko jih je na prioritetni vrsti}}(\overbrace{\text{zahtevnost dodajanja}}^{\log(E)} + \overbrace{\text{zahtevnost brisanja}}^{\log(E)}))$$

PQ seznam: $O(1)$, $O(E)$

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

Lahko tudi: $O(V^2)$

Dvojiška kopica

- levo poravnano dvojiško drevo  ✓
- lastnost kopice: vrednost vsakega starša ≤ min vrednosti oben otrok  ✗

V korenu je najmanjši element, ampak to ni iskalno drevo

Osnovne operacije

dodaj(x)
briši (izbriše minimum)

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

briši:
vzamemo koren in ga izbrišemo.
Najbol spodnji desni element prestavimo v koren in potem ga bublamo dol

12.12

Bellman-ford

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

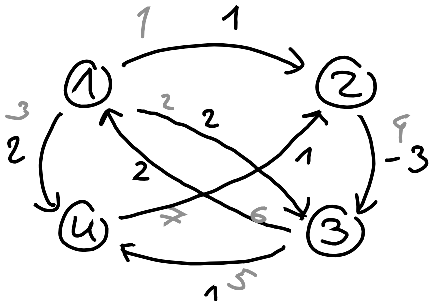
↳ dovolimo negativne povezave

↳ ne dovolimo negativnih ciklov

izhod: seznam d z dolžinami najkrajših poti od s do ostalih vozlišč.

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

```
def f(G, s):
    d = [∞]*n          n ... št vozlišč
    d[s] = 0
    for _ in range(n-1):
        for (u, v, w) in E(G):
            if d[v] > d[u] + w:      } relaksacija
                d[v] = d[u] + w      } 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

potem vemo, da je nekje negativni cikel

Tu bi se morala končati

2) Floyd Warshallov algoritem

a) $D_k(i,j)$ ..... dolžina najkrajše poti od i do j,

ki vsebuje le vozlišča $1, \ldots, k$

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

čas je $O(n^3)$ n...št vozlišč

Velika igra z odprtim svetom ima na različnih lokacijah tržnice, kjer lahko kupujemo in prodajamo surovine

npr: 10 železa za 1000 lesa

Ker se tržnice nahajajo na različnih delih sveta, kjer so [illegible] razlike so lahko menjalni tečaji različni

Surovine X bi radi zamenjali za čim več surovine Y. Kako bi to rešil/

Vozlišča so vse možne surovine

T ... tržnica

$\mathcal{T}$ ... množica tržnic

$$X_1 \to X_2 \Leftrightarrow \exists T.\ T(X_1, X_2)$$

$$w(X_1, X_2) = \max_{T \in \mathcal{T}} \begin{cases} r(X_1, X_2) \text{ menjalni tečaj} \\ r(X_1, X_2) \end{cases}$$

iščemo pot od X v Y kjer je produkt uteži na povezavah čim večji

$$w(X_1, X_2) \Rightarrow -\log(w(X_1, X_2))$$

Problem minimalnega vpetega drevesa

$G(V,E)$ utežen graf (neusmerjen)

w_e uteži

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

Kruskalov algoritem

$[w_{e_1} \dots w_{e_m}]$ ... sortirane povezave

Začnemo z gozdom kjer je vsako krajišče svoje drevo

Če sta krajišči povezave e_i iz različnih dreves to povezavo dodamo v gozd in s tem združimo te dve drevesi

če sta iz istega drevesa preskočimo povezavo

Unija z rangom

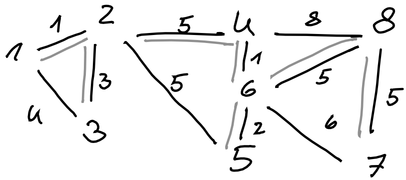
... podatkovna struktura, ki učinkovito opravlja te operacije

naredi(x) .... naredi novo množico z elementom x

najdi (x) .... vrne množico v kateri se nahaja x

Vsako množico predstavi z enim predstav.

združi(x,y) .... naredi disjunktno unijo najdi(x) in 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 Kruske: $O(m \log m) + m\,O(\log n) + n\,O(\log n)$

m× najdi

m ... povezave $|E|$
n ... vozlišča $|V|$

Kompresija poti

Ko na koncu pokličemo najdi 5 ugotovimo
da najdi(5) ≠ starš(5)
Potem se splača za starša(5) nastavit naji(5)