

Krovni izrek: $T(n) = aT(\frac{n}{b}) + O(n^d)$

$$T(n) = \begin{cases} O(n^d) & ; a < b^d \\ O(n^d \ln(n)) & ; a = b^d \\ O(n^{\log_b(a)}) & ; a > b^d \end{cases}$$

Iskanje z bisekcijo $O(\log(n))$

Vhod: urejena tabela a , x

Izhod: ali je x v tabeli

```
def bisekcija(a, x)
```

```
    l, r := 0, len(a)-1
```

```
    while l < r:
```

```
        m = (l+r)//2
```

```
        if a[m] == x: return m
```

```
        if a[m] < x: l = m+1
```

```
        else: r = m-1
```

```
    return False
```

Urejanje z zlivanjem $O(n \log(n))$

```
def merge(a, b):
```

```
    if len(a) == 0:
```

```
        return b
```

```
    if len(b) == 0:
```

```
        return a
```

```
    if a[1] < b[1]:
```

```
        return a[1] + merge(a[1:], b)
```

```
    else: return b[1] + merge(a, b[1:])
```

```
def merge_sort(a):
```

```
    if n > 1
```

```
        return merge(ms(a[:n/2]), a[n/2:])
```

```
    else: return a
```

Izbira pivota

```
def select(k, a)  $O(n)$ 
    if len(a) == 1:
        return S[1]

    izberi i  $\in [n]$ 
    p = S[i]
    SL = a[:l]
    SV = a[l+1:m]
    SR = a[m+1:]

    if l+1 ≤ k ≤ m:
        return i

    if k ≤ l:
        return select(k, SL)
    if k ≥ m+1:
        return select(k-m, SR)
```

```
def drug_select(a):
    izberi i  $\in [n]$ 
    SL, SV, SR

    if |SL|, |SR| ≤  $\frac{3}{4}n$ 
        return i
```

Hitro urejanje

```
def quick_sort(a)
    if len(a) ≤ 1: return a
    p = random.choice(a)
    SL, SV, SR
    return quick_sort(SL)
        + SV
        + quick_sort(SR)
```

DFS

↙ lahko usmerjen $O(|V|^2)$ ($V^2 \geq E$)

def dfs(G, v) $O(E)$ • povezana komponenta od v

visited[V] := True • pre-post

$s := \text{sklad}$

• usmerjen cikel $\Leftrightarrow$ povratna povezava

$s.push(v)$

while s :

$u := s.pop()$

if not visited[u]

for w in sosedu(u)

$s.push(w)$

visited[u] = True

$u \rightarrow v$
prema $[a, b] \supset [c, d]$
povratna $[a, b] \subset [c, d]$
prečna $[a, b] \cap [c, d] = \emptyset$

• Topološka urejenost
(če je DAG)
(uredimo po
največjem post)

krepro povezane komponente $u \rightsquigarrow v$

• $C(G) = C(G^R)$

• kvocientigraf je DAG

• $C \rightarrow C' \Rightarrow \max \text{post}(C) > \max \text{pos}(C')$

• točka z največjim post pri DFS je v ponoru

Tarjanov algoritem (za krepro povezane komponente)
konstruiraj G^R ($O(V+E)$)

Razvrstimo z DFS po padajočem post

Izvedemo DFS za G ,

iz ponora naprej

BFS $O(V+E)$

def bfs(G,s)

raz = $[\infty] * n$

raz[s] = 0

Q = vrsta

Q.push(s)

while Q:

u = Q.pop()

for u.sosed(v)

if raz[v] = ∞ :

raz[v] = raz[u] + 1

Q.push(v)

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

def Dijkstra(G,l,s)

dist = $[\infty] * n$

prev = [None] * n

dist[s] = 0

Q = prioritetna vrsta

u = Q.getmin()

for v.sosed(u):

if dist(v) > dist(u) + w(u,v):

dist(v) = dist(u) + w(u,v)

prev(v) = u

Q.decrease_key(v, dist(v))

prioritetna vrsta:

insert $O(\log(V))$

extract_min $O(\log V)$

decrease_key $O(\log V)$

(Pravilna kop: ∞)

Topološka ureditev

```
def toposort(G)  $O(V+E)$ 
    indeg = [0]*n
    for i in range(n)
        for j in sasedi(i)
            indeg[j] += 1
    vrsta = tisti, ki imajo 0 staršev
    topo = []
    while vrsta:
        c = vrsta.poplett()
        topo.append(c)
        for j in sasedi(c)
            indeg[j] -= 1
            if indeg[j] == 0
                vrsta.push(j)
    return topo
```

```
def deg-najkrajša_pot(G, s)  $O(V \cdot \deg(G))$ 
    dist = [∞]*|V|
    prev = [None]*|V|
    dist[s] = 0
    topološko_uredi(G)
    for u ∈ V (topološko urejeni)
        for u, w sased(v)
            if dist(v) > dist(u) + w:
                dist(v) = dist(u) + w
                prev(v) = u
```

Bellman-Ford $O(V \cdot E)$

vhod: usmerjen graf, brez negativnih ciklov, s
izhodi razdelja do s za vsako vozlišče

def Bellman_ford(G, s)

dist = $[\infty] * |V|$

prev = $[None] * |V|$

dist[s] = 0

for _ in range($|V|-1$)

for $(u, v, w) \in E$

if dist(v) > dist(u) + w:

dist(v) = dist(u) + w

prev(v) = u

Floyd-Warshall $O(V^3)$

$D_k(i, j)$... najkrajša pot od i do j, z vozlišči $[k]$

$D_0(i, j) = \begin{cases} w(i, j) \\ \infty \end{cases}$

$D_k(i, j) = \begin{cases} D_{k-1}(i, k) + D_{k-1}(k, j), & D_{k-1}(i, j) \end{cases}$

$D_n(i, j)$ najkrajša razdelja

$D_n(i, i) < 0 \Rightarrow$ negativen cikel

Požrešna metode

def Huffmanova_kopice (S, f) ^{malik} ^{frekvence znakov}

H... prioriteta vrsta urejena po f

for k in rang $(len(f), 2len(f))$

i, j = dva najmanjša iz H

$f[k] = f[i] + f[j]$

ustvarimo novo z i, j otrokoma

H.insert $(f[k], k)$

Drveca

$T(G) = T(G - e) + T(G/e)$

def kruskov_algorithm (G)

$T = \emptyset$
for $v \in V$

Make-Set (v)

sort (E)

for $(w, u, v) \in E$

if Find-set $(u) \neq$ Find-set (v)

$T.append((w, u, v))$

return T Union (u, v)

Unija z rangom

def MakeSet (x) $O(1)$

$\pi(x) = x$

rank = 0

def Find (x) $O(\log n)$

while $x \neq \pi(x)$

$x = \pi(x)$

def Union (x, y) $O(\log n)$

$rx = \text{Find}(x)$

$ry = \text{Find}(y)$

if $rx == ry$

return

if rank $(rx) >$ rank (ry)

$\pi(ry) = rx$

if rank $(rx) ==$ rank (ry)

rank $(ry) =$ rank (ry)

+ 1

else: $\pi(rx) = ry$

def kompresija-poti-find (x)

if $x \neq \pi(x)$:

$\pi(x) = \text{komp-po-find}(\pi(x))$

return $\pi(x)$

$O(m \log^*(n))$

//

$\log(\log(\dots(\log(n)) \leq 1$

?

def primov-algorithm(G)

$Q = V$

$kljuc = [\infty] \cdot \text{len } Q$

$kljuc(r) = 0$

$\pi(r) = \text{None}$

while $Q \neq \emptyset$:

$u = Q.\text{Extract-min}$

for $v \in \text{sosed}_i(u)$

if $v \in Q \wedge w(u, v) < kljuc[v]$

$\pi[v] = u$

$kljuc[v] = w(u, v)$

Q kopica
Binomska:

$O(E \cdot \log V)$

Fibon:

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