Bisekcija

Naloga: V dani tabeli

$$a = [a_0, a_1, \ldots, a_{n-1}]$$

poišči indeks danega elementa x.

Postopek (algoritem): po vrsti pregledamo tabelo in iščemo x.

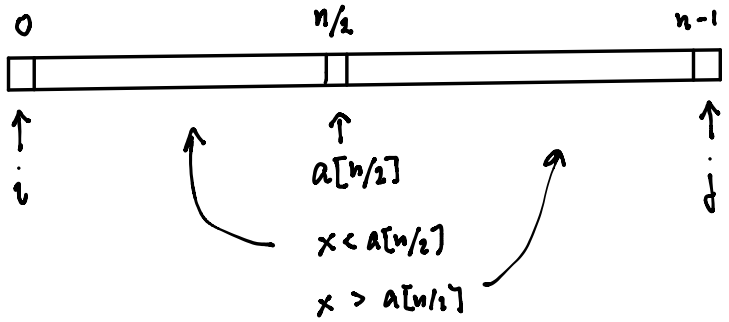
Časovna zahtevnost n

Naloga: V dani urejeni tabeli a poišči indeks elementa x.
(Dodatno: elementi v a se ne ponavljajo.)

Postopek:

$$a_0 < a_1 < a_2 < \cdots < a_{n-1}$$

$n = \text{len}(a)$



Časovna zahtevnost bisekcije:

- Širina iskalnega območja se razpolavlja:

n

$\frac{n}{2}$

$\frac{n}{4}$

$\frac{n}{8}$

$\vdots$

$\frac{n}{2^k}$ po k korakih

Ustavi se, ko je $\frac{n}{2^k} = 1$

$n = 2^k$

$k = \log_2 n$

① Odgovor: $O(\log_2 n)$

Časovna zahtevnost rekurzivne funkcije (glej kodo spodaj):

$T(k)$ = časovna zahtevnost funkcije isci, če je iskalno območje velikosti d

$(d = j - i)$

$$T(0) = 1$$

$$T(d) = 1 + T(d/2)$$

↑ konstantno število korakov

$$T(n) = ?$$

$$T(n) = 1 + T\left(\frac{n}{2}\right)$$

$$= 1 + 1 + T\left(\frac{n}{4}\right)$$

$$= 1 + 1 + 1 + T\left(\frac{n}{8}\right)$$

$$\vdots$$

$$= \underbrace{1 + \dots + 1}_{k} + T\left(\frac{n}{2^k}\right)$$

ustavimo se, ko je $k = \log_2 n$

$$= \underbrace{1 + \dots + 1}_{\log_2 n} + T(1)$$

$$= \log_2 n$$

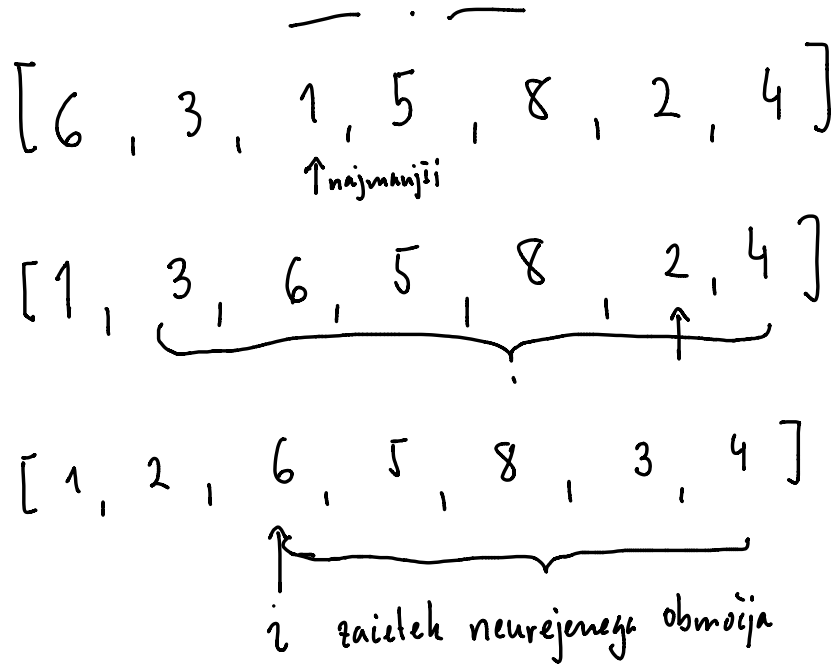
Odgovor $O(\log_2 n)$

```
def bisekcija1(a, x):
    """V urejeni tabeli a poišči indeks elementa x z bisekcijo, rekruzivna verzija."""

    # definiramo pomožno funkcijo, ki išče v podtabeli a[i], a[i+1], ..., a[j],
    # pri čemer se izognemu temu, da bi dejansko zgradili podtabelo
    def isci(i, j):
        if j < i:
            return None
        else:
            k = (i + j) // 2
            if x == a[k]:
                return k
            elif x < a[k]:
                return isci(i, k-1)
            else: # a[k] < k
                return isci(k+1, j)

    # dejansko pokličemo pomožno funkcijo

    return isci(0, len(a)-1)
```

Urejanje tabel

Urejanje na mestu:
- vhod: tabela a
  preuredi elemente a, da bo urejena

Urejanje:
- vhod: tabela a
- izhod: urejena tabela elementov iz a
  (prvotne tabele a ne smemo spremeniti)



```
def urejanje_z_izbiranjem(a):
    """Uredi tabelo a na mestu."""
    n = len(a)
    for i in range(0, n-1):
        # poiščemo indeks j najmanjšega elementa v podtabeli a[i:]
        j = i # kandidat za indeks najmanjševa v podatebeli a[i:]
        for k in range(i, n):
            if a[k] < a[j]: j = k # popravi kandidata, če smo našli boljšega
        # zamenjamo a[i] in a[j]
        (a[i], a[j]) = (a[j], a[i])
```

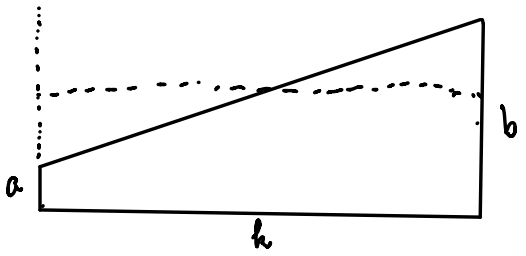
notranja zanka se izvede $(n-i)$-krat

zunanja	notranja	
$i=0$	$n-0$	n
$i=1$	$n-1$	$\vdots$
$\vdots$		$\vdots$
i	$n-i$	$\vdots$
$\vdots$		3
$i=n-2$	$n-(n-2)$	2

to seštejemo

To je manj kot

$$1+2+3+\cdots+n = \frac{1+n}{2}\cdot n \in \mathcal{O}(n^2) \quad \underline{\text{slabo}}$$



$\frac{a+b}{2}\cdot h$