# Algoritmi in podatkovne strukture – 2

## Mediana

## $k$-ti element po velikosti

# Iskanje $k$-tega elementa (*Select*)

Imamo vhodni niz predmetov: $X = [x_1, x_2, ...x_n]$ in številko $k <= n$.

Za poljubna predmeta velja, da sta urejena, kar pomeni, da je eden večji ali enak kot drugi: $x_i \leq x_j$ ali $x_i \leq x_j$ in da je ta relacija tranzitivna.

Poiščemo $k$-ti element po velikosti v nizu.

Posebni primeri:

- najmanjši (minimalni) element – $k = 1, \min X$ $O(n)$
- največji (maksimalni) element – $k = n, \max X$ $O(n)$
- mediana – $k = \lfloor \frac{n}{2} \rfloor$ $O(?)$

Preprosta rešitev:

- uredimo elemente po velikosti
- pogledamo $k$-ti element
- čas: $O(n \log n)$ – se dâ kaj bolje?

# Hitro urejanje – *quick sort*

```
public void QuickSort(int[] X, int l, int r) {
  if (l >= r) return;
  int p= X[l];
  int ix_p;
  ix_p= Partition(X, l, r);
  QuickSort(X, l, ix_p-1);  QuickSort(X, ix_p+1, r);
}
```

# Hitro iskanje – *quick select*

```
public int QuickSelect(int[] X, int l, int r, int k) {
  if (l >= r) return X[l];
  int p= X[l];
  int ix_p;
  ix_p= Partition(X, l, r);
  if (k < ix_p) return QuickSelect(X, l, ix_p-1, k);
  else          return QuickSelect(X, ix_p+1, r, k-ix_p);
}
```

Podobno kot pri hitrem urejanju, le da sedaj nam ni potrebno iskati v obeh poddelih polja, ampak samo v enem – v tistem, ki je pravi.

In časovna zahtevnost: v najboljšem (in pričakovanem) primeru $O(n)$. Kaj pa najslabši primer?

# Iskanje v najslabšem primeru

Težava s `QuickSelect` algoritmom je, da:

- slabo razdeli niz
- in za razdelitev porabi čas $O(r - l) = O(n)$

Druge pomanjkljivosti ne moremo odpraviti (zakaj?), zato se lotimo prve.

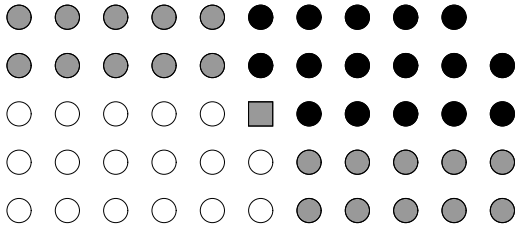
# Boljša delitev

- razdelilni element moramo izbrati tako, da bomo v vsakem koraku izločili *ne samo konstanto* število elementov, ampak nekako $\frac{n}{p}$ za neko konstanto $p > 1$.

Iskanje razdelilnega elementa:

1. razdelimo $n$ elementov na $\frac{n}{5}$ peterk (v zadnji $\gg$peterki$\ll$ je lahko manj elementov)
2. v vsaki peterki poiščemo mediano (v zadnji je lahko malce drugače) in dobimo $\frac{n}{5}$ elementov
3. med temi rekurzivno poiščemo mediano

# Razdelilni element



Velja

- število elementov, ki so *za gotovo* večji od razdelilnega elementa, je *vsaj*

$$3\left(\left\lceil\frac{1}{2}\left\lceil\frac{n}{5}\right\rceil\right\rceil-2\right)\geq\frac{3n}{10}-6$$

  kjer smo zanemarili zadnji stolpec, ki je morda nepopoln, in stolpec, v katerem je razdelilni element. Podobno, je število elementov, ki so *za gotovo* manjši od razdelilnega elementa je *vsaj* $\frac{3n}{10}-6$;
- torej je število elementov, ki so manjši manjši od razdelilnega elementa *največ*

$$n-\left(\frac{3n}{10}-6\right)=\frac{7n}{10}+6$$

# Iskanje

Psevdokoda – podrobnosti opuščene:

```
int Select(int[] X, int l, int r, int k) {
  if (X.length <= 1) return X[0];
  int i, groups= X.length / 5;
  int[] Y= new int[groups];
  for (i= 0; i < groups; i++) {
    Sort(X, l+5*i, l+5*i+5);
    y[i]= X[l+5*i + 3];
  }
  p= Select(Y, 0, groups, groups/2);
  ix_p= Partition(X, l, r, p);
  if (k < ix_p) return Select(X, l, ix_p-1, k);
  else          return Select(X, ix_p+1, k-ix_p);
}
```

# Časovna zahtevnost

$$T(n) \leq T(\left\lceil \frac{n}{5} \right\rceil) + T(\frac{7n}{10} + 6) + O(n)$$

Recimo, da velja (hipoteza): $T(n') \leq cn'$ za neko konstanto $c$ in vse $n' < n$. Potem:

$$\begin{aligned} T(n) &\leq T(\left\lceil \frac{n}{5} \right\rceil) + T(\frac{7n}{10} + 6) + O(n) \\ &\leq c \lceil n \rceil 5 + \frac{7cn}{10} + 6c + O(n) \\ &\leq \frac{9cn}{10} + 6c + O(n) \\ &\leq cn \end{aligned}$$

# Zahtevnost

| rešitev | zahtevnost |
|---|---|
| uravnotežena drevesa | $O(\log n)$ |
| `QuickSelect` | $O(n^2)$ |
| `Select` | $O(n)$ |

Komentarji:

- So rešitve primerljive?
- Kaj pa prostorska zahtevnost `Select`? Upoštevati moramo še polje `Y`. Je nujno?