Poradová štatistika

- i - tá poradová štatistika n prvkov (napr. kľúčov tabuľky) je i - tý najmenší prvok v tabuľke.
- i - tá poradová štatistika je prvok, ktorý je väčší ako práve i - 1 iných prvkov tabuľky

Predpokladajme jedinečnosť hodnôt kľúčov.
Špeciálne poradové štatistiky:
Minimum: i = 1
Maximum: i = n
Medián: i = (n + 1) / 2 ak n je nepárne,
i = n / 2 a i = n / 2 + 1 (dva mediány) ak n je párne.

Ukážeme si štyri algoritmy pre nájdenie i - tej poradovej štatistiky z n prvkov (prvé tri algoritmy predpokladajú uloženie prvkov v poli, štvrtý predpokladá uloženie prvkov vo vyváženom binárnom vyhľadávacom strome):

I. Poradová štatistika v očakávanom (priemerná zložitosť) aj najhoršom čase (najhoršia zložitosť) $O(n*\log_2 n)$
   1. utrieď prvky algoritmom s najhoršou zložitosťou $O(n*\log_2 n)$ (napr. Heapsort)
   2. v i - tom prvku je požadovaná štatistika

II. Poradová štatistika v očakávanom čase O(n), najhoršom $O(n^2)$
   Algoritmus je založený na tej istej myšlienke ako QuickSort (partitioning – oddeľovanie). Po oddelení sa však nevenuje obom častiam (ako QuickSort) ale iba tej, v ktorej sa nachádza index i.

Príklad:

Hľadáme 4-tú štatistiku

Pivot

Situácia po oddelení pivotom 8

1	2	3	4	5	6	7	8	9	10	11
7	**5**	**1**	**6**	**2**	**3**	**8**	**11**	**10**	**9**	**12**

V tejto časti pokračujeme voľbou pivota a jej rozdelením (prvky 1–6)

V tejto časti nemôže byť 4-tá štatistika, už sa k nej nevrátime (prvky 7–11)

Nepriaznivá najhoršia zložitosť je spôsobená <u>náhodným</u> výberom pivota (prípad, že za pivota sa vyberie vždy prvok blízky maximu alebo minimu v spracovávanej časti), tak ako je to aj v QuickSorte. Rovnako ako pri QuickSorte existuje viacero možných stratégií ako vybrať pivota. Jednou z nich výber mediánu z prvého, posledného a prostredného prvku. Inou možnosťou je náhodný výber pivota.

III. <u>Poradová štatistika v očakávanom aj najhoršom čase O(n):</u>

Princíp rozdeľovania zostáva podobný, ako v algoritme II ale deliaci prvok (pivot) sa vyberá deterministicky. Pivot sa vyberá ako medián z mediánov menších častí (o dĺžke 5 alebo iné nepárne číslo) a využíva sa skutočnosť, že ak tento medián mediánov častí postupnosti použijeme ako pivot pre rozdelenie zaručí nám to najhoršiu zložitosť O(n).

1. **Rozdeľ n prvkov uložených v poli na $\lfloor n/5 \rfloor$ skupín po 5 prvkov a najviac jednu obsahujúcu zvyšných n *mod* 5 prvkov.**
2. **Nájdi medián každej z $\lceil n/5 \rceil$ častí napr. tak, že sa utriedi InsertSortom a zoberie sa prostredný prvok (v prípade párneho počtu menší z nich).**
3. **Nasaď rekurzívne tento algoritmus na nájdenie mediánu *x* z $\lceil n/5 \rceil$ mediánov nájdených v kroku 2.**
4. **Teraz, po výstupe z rekurzie *x* síce nie je mediánom všetkých prvkov, ale je dokázané, že ak ho použijeme ako pivot, žiadna z častí (partition) nemôže obsahovať menej ako (3n/10)-6 prvkov (pre veľké n takmer 30%).**

5. Rozdeľ (ako v Quicksorte) všetky prvky na menšie ako *x* a väčšie ako *x*.
6. Nasaď rekurzívne tento algoritmus na tú časť poľa prvkov, v ktorej sa nachádza index *i*.

Príklad:

Hľadáme 4-tú štatistiku

70 52 01 46 92 13 88 11 10 09 12 54 21 22 71 45 66 32 19 44 72 06 59 35 42 97 15 16 58 62 49 40 18

Utriedime po častiach

01 46 52 70 92 09 10 11 13 88 12 21 22 54 71 19 32 44 45 66 06 35 42 59 72 15 16 58 62 97 18 40 49

Vyberieme mediány častí

52 11 22 44 42 58 40

Utriedime po častiach

11 22 42 44 52 40 58

42 40

x

40

Medián mediánov použijeme ako pivot pre rozdelenie

i

1 2 3 4 5

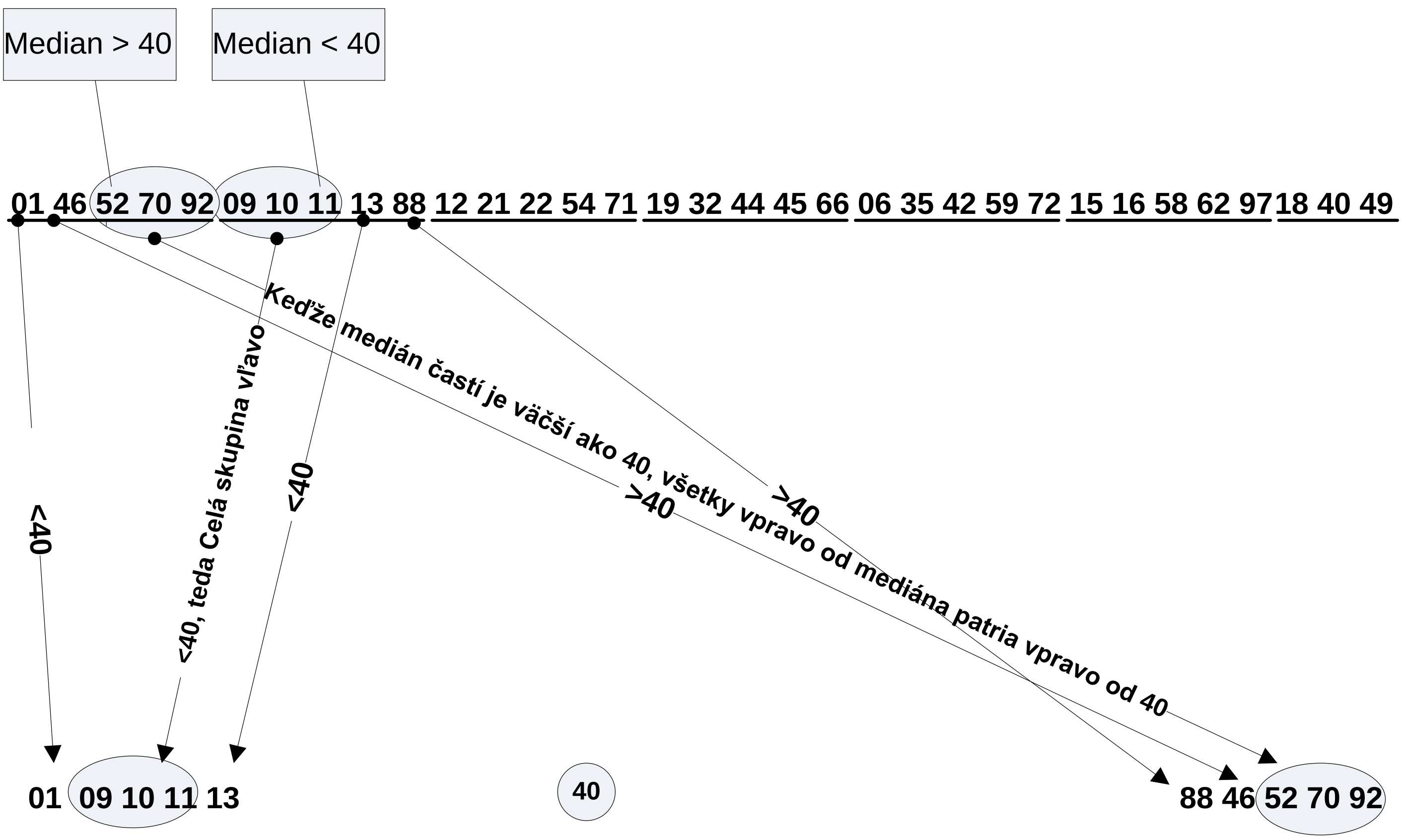
01 13 11 10 09 12 21 22 32 19 06 35 15 16 23 18 40 70 52 46 92 88 54 71 45 66 44 72 59 42 97 58 62 49

V tejto časti pokračujeme hľadaním mediána mediánov, ktorý použijeme ako ďalší pivot

V tejto časť nemôže byť 4-tá štatistika, už sa k nej nevrátime

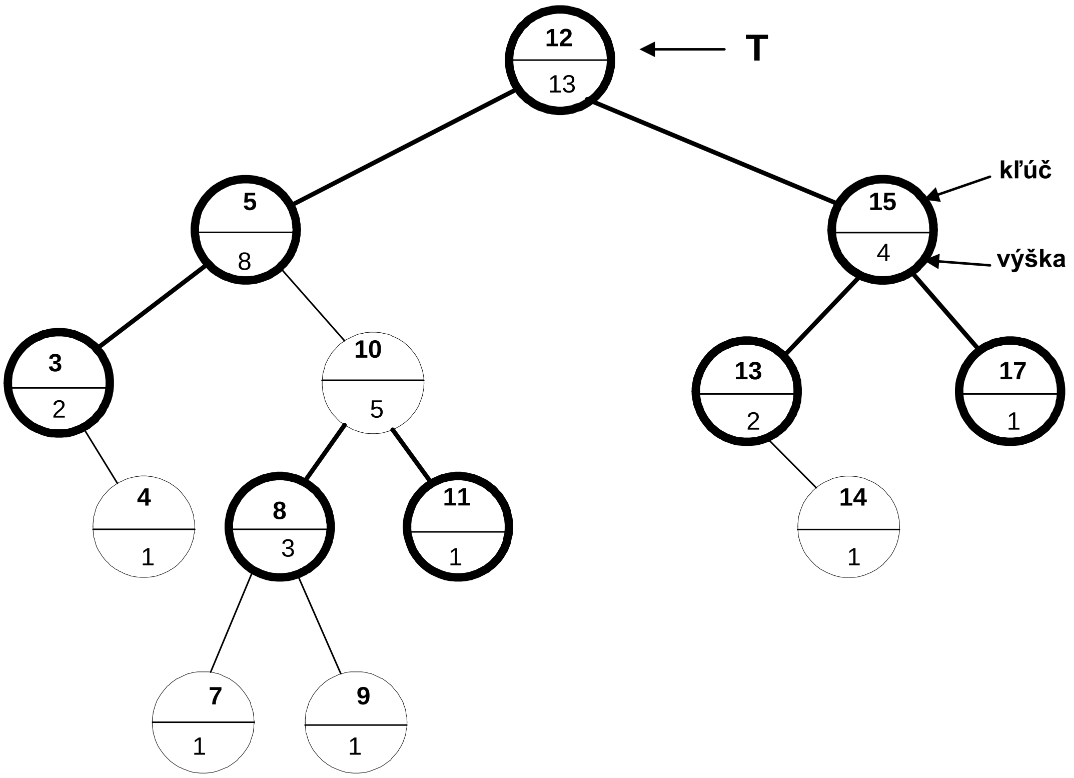
Zefektívnenie procesu rozdeľovania:

- **ak je medián skupiny menší ako medián mediánov, sú menšie aj všetky vľavo od neho v skupine a teda sa môžu presunúť do ľavej časti bez jednotlivého porovnávania**
- **analogicky, keď je medián skupiny väčší**



IV. Poradová štatistika v očakávanom aj najhoršom čase $O(\log_2 n)$

Predpokladá voľbu štruktúry dát vo forme ľubovoľného vyváženého binárneho vyhľadávacieho stromu (napr. RB strom).
Takýto strom doplníme tak, že každý vrchol *x* bude obsahovať informáciu *size(x)* o počte vrcholov v podstrome, ktorého je koreňom vrátane koreňa samého.



Ak dohodneme size(nil) = 0, tak:
size(x) = size(left(x))+ size(right(x)) + 1

Ukážeme si rekurzívnu formu algoritmus, ktorý nájde i - tú poradovú štatistiku v strome T:

funkcia Statistika(strom T, prvok i)

1. **r = size (left(x))**
2. **ak i = r + 1, vráť x (i – tý prvok nájdený), inak krok 3**
3. **ak i < r + 1, vráť Statistika(left(x), i),**
   inak vráť Statistika (right(x), i - r - 1)

Pri reorganizáciách spôsobených operáciami *Vlož, Odober* je nutné aktualizovať hodnoty *size*.