

## Hledání k-tého nejmenšího prvku

1. Seřad' seznam/pole a vyber k-tý nejmenší, složitost  $\Theta(N \cdot \log(N))$ .

Nevýhodou je zbytečné řazení úplně všech dat.

2. Využij Randomized select založený na principu Quick sortu. Očekávaná složitost  $\Theta(N)$ , nejhorší možná  $\Theta(N^2)$ .

3. Využij Select založený na principu Quick sortu s výběrem pivota metodou medián z mediánů.

## Randomized select

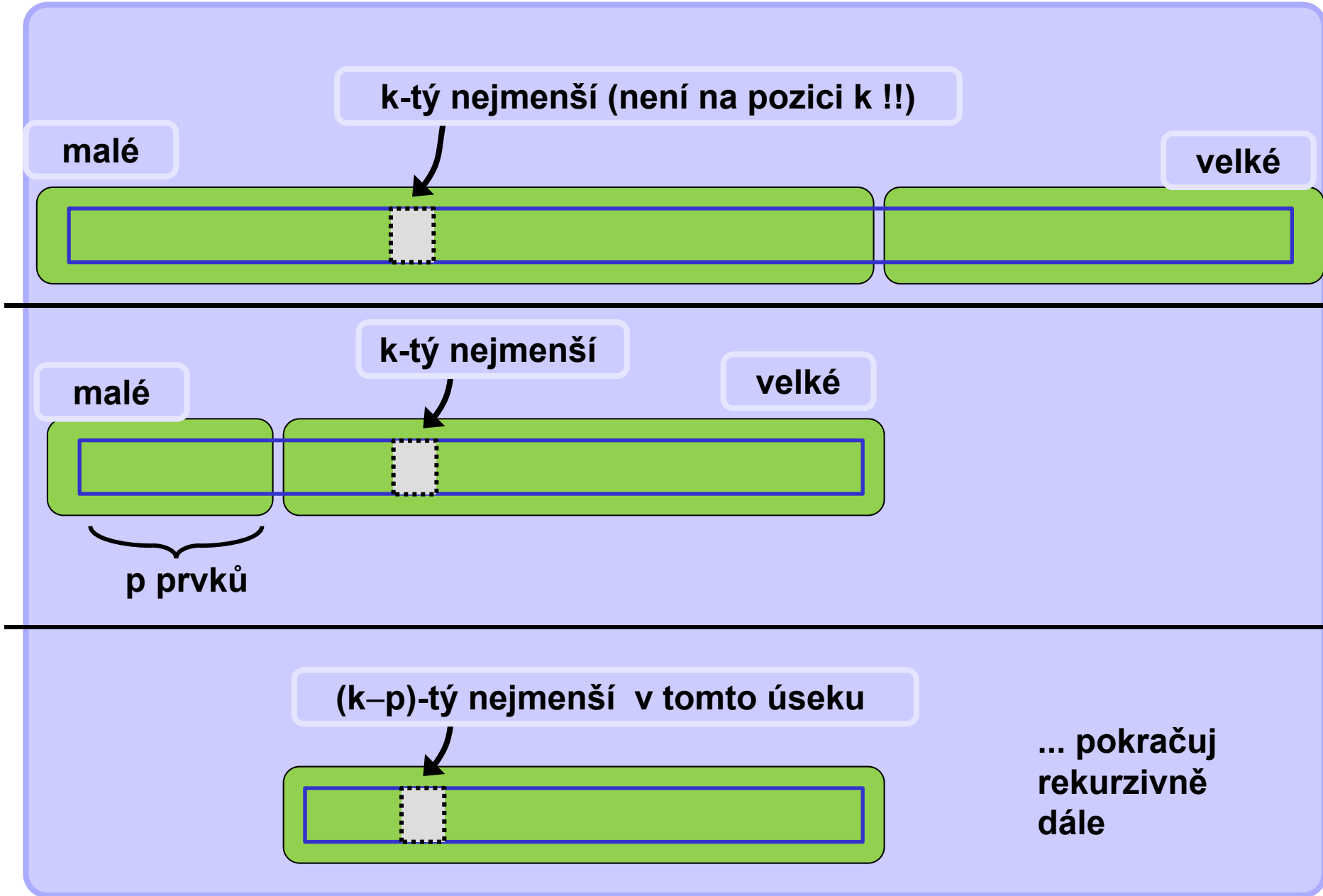
Partition -- dělení na "malé" a "velké" v Quicksortu

Randomized partition

- prohod' první prvek pole/úseku s náhodně zvoleným prvkem pole/úseku,
- proved' partition s tím, že pivotem je prvek na první pozici pole/úseku.

Randomized select

- když je délka úseku rovna 1, vrať jeho jediný prvek,
- proved' randomized partition tohoto úseku, vznikne levý a pravý podúsek,
- aplikuj randomized select rekurzivně buď na levý nebo pravý podúsek, podle toho, v kterém z nich může ležet k-tý nejmenší prvek celého pole.



## Select metodou medián z mediánů

Select se od Quicksortu liší hlavně tím, že nevolá rekurzi na úsek s "malými" a "velkými" hodnotami, ale jen na jeden z nich, podle toho, v kterém z nich lze očekávat výskyt hledaného k-tého nejmenšího prvku.

Jak si pamatujeme z Quicksortu, pokud je úsek "malých" nebo "velkých" hodnot opakovaně extrémně krátký, může to vést až na kvadratickou složitost Quicksortu.

Stejná potíž by mohla nastat i metodě select.

Je nutno zajistit, aby v metodě partition byl vybrán pokaždé dostatečně vhodný pivot, tj. takový, který dobře rozdělí daný úsek na "malé" a "velké", tj. takový, který je pokud možno co nejblíže mediánu hodnot tohoto úseku.

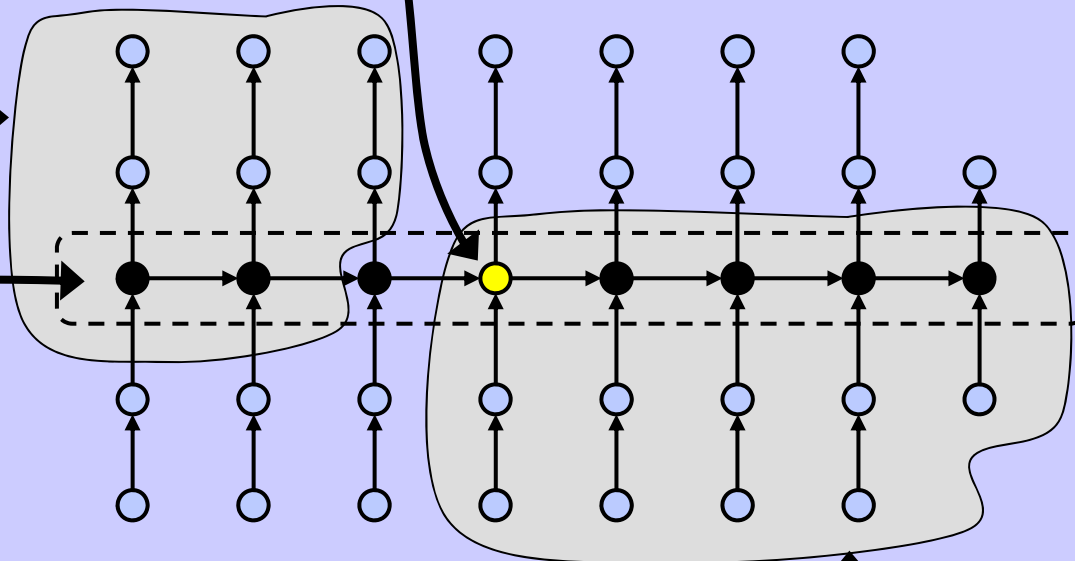
## Select metodou medián z mediánů

0. Pokud má aktuální úsek pole délku 1, vrať jeho jediný prvek.
1. Rozděl pole nebo úsek pole délky  $N$  na  $\lfloor N/5 \rfloor$  skupin po pěti prvcích a nejvýše jednu zbylou s méně než 5 prvky.
2. Každou skupinu seřaď (např. Insert sort) a vyber z ní její medián. Všechny tyto mediány přesuň na začátek pole/úseku.
3. Aplikuj 1. až 5. rekurzivně na úsek obsahující právě nalezené mediány, dokud nezískáš medián původních  $\lceil N/5 \rceil$  mediánů z bodu 2. Tento medián  $M$  bude pivotem.
4. Rozděl pole/úsek na "malé" a "velké" pomocí metody partition s pivotem  $M$ .
5. Podle velikosti úseků "malých" a "velkých" rozhodni, v kterém z nich se nachází  $k$ -tý nejmenší prvek celého pole a na něj aplikuj rekurzivně postup počínaje bodem 0.

Prvky zaručeně menší než  $M$

$M = \text{medián } \lceil N/5 \rceil \text{ mediánů}$

$\lceil N/5 \rceil$  mediánů



Prvky zaručeně větší než  $M$

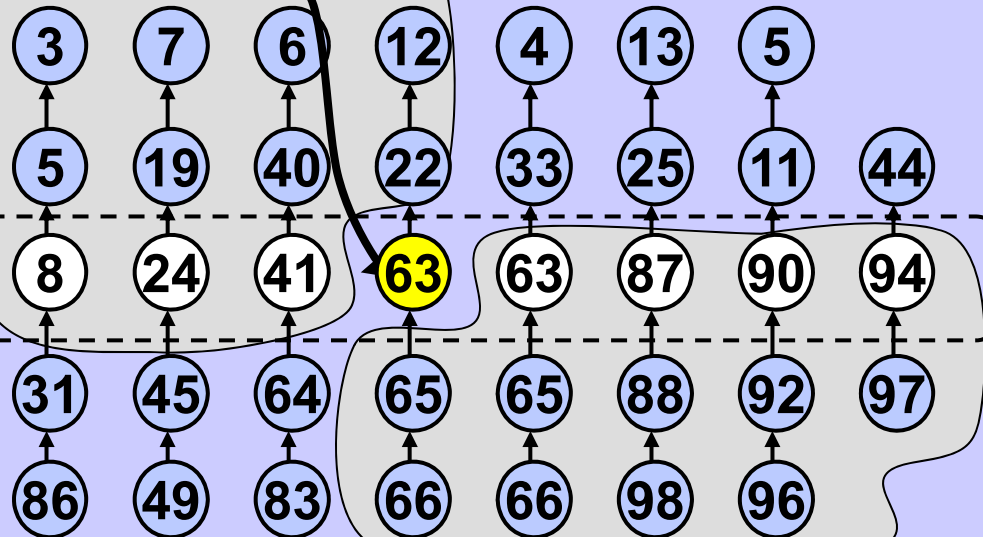
Prvků zaručeně větších než  $M$  je alespoň  $3N/10-6$ .  
Analogicky, prvků menších než  $M$  je také alespoň  $3N/10-6$ .

V bodě 5 se rekurze zavolá na úsek obsahující nejvýše  
 $N-(3N/10-6) = 7N/10+6$  prvků.

Prvky zaručeně menší než M

M = medián  $\lceil N/5 \rceil$  mediánů

$\lceil N/5 \rceil$  mediánů



Prvky zaručeně větší než M

Prvků zaručeně větších než M je alespoň  $3N/10-6$ .  
Analogicky, prvků menších než M je také alespoň  $3N/10-6$ .

V bodě 5 se rekurze zavolá na úsek obsahující nejvýše  $N-(3N/10-6) = 7N/10+6$  prvků.

## Select metodou medián z mediánů

Metoda subSort je obyčejný Insert Sort,  
rychlý na malých úsecích pole.

Zde slouží k nalezení mediánu hodnot v úseku délky 5 nebo menší.

```
void subSort(int [] a, int L, int R) {  
    int insVal, j;  
    for(int i = L+1; i <= R; i++) {  
        insVal = a[i];  
        j = i-1;  
        while ((j >= L) && (a[j] > insVal)) {  
            a[j+1] = a[j];  
            j--;  
        }  
        a[j+1] = insVal;  
    }  
}
```



## Select metodou medián z mediánů

Dělení úseku pole od indexu L, až po index R včetně na "malé" a "velké" hodnoty podle předem zvolené pivotní hodnoty.

Analogicky jako v Quick sortu.

```
int partition( int [] a, int iL, int iR, int piv) {  
    do {  
        while (a[iL] < piv) iL++;  
        while (a[iR] > piv) iR--;  
        if (iL < iR)  
            swap(a, iL++, iR--);  
    } while( iL < iR);  
    if (iR > iL) return iR;  
    if (a[iR] < piv) iR++;  
    return iR;  
}
```

## Select metodou medián z mediánů

Metoda select volá sama sebe dvakrát.  
Poprvé pro nalezení mediánu z mediánů úseků délky 5,  
podruhé pro hledání k-tého prvku celého pole  
mezi buďto "malými nebo "velkými" hodnotami v aktuálním úseku.

```
int select(int [] a, int L, int R, int k) {  
    // 0. if the group is short terminate the recursion  
  
    // 1. move medians of groups of 5 to the left  
  
    // 2. also the median of the last possibly smaller group  
  
    // 3. find median M of medians of 5  
  
    // 4. use M for partition  
  
    // 5. recursively call select on the appropriate part  
  
}
```

## Select metodou medián z mediánů

```

int select(int [] a, int L, int R, int k) {

    // 0. if the group is short terminate the recursion
    if (L == R) return a[L];
    if (L+1 == R) return (a[L] < a[R]) ? a[L] : a[R];

    // 1. move medians of groups of 5 to the left
    int lastM = L; // place for medians
    int i5; // 5-elem groups index
    for (i5 = L; i5 <= R-4; i5 += 5) {
        subSort(a, i5, i5+4); // find median of 5
        swap(a, lastM++, i5+2); // put it to the left
    }

    // 2. also the median of the last possibly smaller group
    if (i5 <= R) {
        subSort(a, i5, R);
        swap(a, lastM, i5+(R-i5)/2 ); // lastM -> last of medians
    }

    ... // continue
}

```

## Select metodou medián z mediánů

```
... // continued:

// 3 .find median (medmed5) of medians of 5
int medmed5 = select(a, L, lastM, L+(lastM-L)/2);

// 4. use medmed5 for partition, iBig -> first "big" value
int iBig = partition(a, L, R, medmed5);

// 5. recursively call select on the appropriate part
if (iBig > k)
    // the k-th elem is among the "small"
    return select(a, L, iBig-1, k);
else
    // the k-th elem is among the "big"
    return select(a, iBig, R, k);
} // end of select
```

## Odvození složitosti

Rekurence:

$$T(N) \leq \begin{cases} O(1) & N < 140 \\ T(\lceil N/5 \rceil) + T(7N/10+6) + O(N) & N \geq 140 \end{cases}$$

**bod 3**      **bod 5**      **body 1, 2, 4**

Řešení substituční metodou, ukážeme

$$T(N) \leq cN,$$

pro dostatečně velké  $c$  a pro  $N > 0$ .