

Dato un vettore A trova l'elemento che finirebbe in posizione i-esima se A fosse ordinato

Ordinando il vettore la complessità sarebbe  $\Omega(n \log n)$  Usando Partition, senza ordinare,  $O(n)$

Alcuni casi particolari: -  $i=1 \rightarrow$  trova il minimo -  $i=n \rightarrow$  trova il massimo -  $i=5 \rightarrow$  trova il minimo per 5 volte

### Prima soluzione: quasiSelect

Sfruttando partition, si costruisce questa soluzione:

```
1 quasiSelect(A, p, q){
2     if(p=q): return A[p]
3     else{
4         r = partition(A, p, q)
5         if (i=r): return A[r]
6         if (i<r): return quasiSelect(A, p, r-1, i)
7         if (i>r): return quasiSelect(A, r+1, q, i)
8     }
9 }
```

Il problema è che partition usa come pivot l'ultimo elemento del vettore passatogli come argomento. Quindi non è detto che sia il mediano, la chiamata successiva potrebbe avvenire al massimo su  $n-1$  elementi

### Ottimizzazione di quasiSelect

1. Si divide A in blocchetti da 5 elementi
2. Per ogni blocchetto si trova il mediano e si copia in B, array ausiliario
3. Si trova il mediano di B (il mediano dei mediani)  $\rightarrow$  M
4. Si usa M come pivot per il partition

```
1 Select(A, p, q){
2     if(p=q): return A[p]
3     else{
4         M = findMedian(A, p, q) // And copy in in B[i]
5         swap(A, M, q)
6         r = partition(A, p, q)
7         if (i=r): return A[r]
8         if (i<r): return quasiSelect(A, p, r-1, i)
9         if (i>r): return quasiSelect(A, r+1, q, i)
10    }
11 }
```

## Complessità

$$f_X(x; x_1, \dots, x_n) = \begin{cases} \frac{1}{n} & \text{se } x = x_1, \dots, x_n \\ 0 & \text{altrimenti} \end{cases}$$