> 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 -> trova il minimo - i=n -> trova il massimo - i = 5 -> trova il minimo per 5 volte

Prima soluzione: quasiSelect

Sfruttando partition, si costruisce questa soluzione:

```
quasiSelect(A, p, q){
    if(p=q): return A[p]
    else{
        r = partition(A, p, q)
        if (i=r): return A[r]
        if (i<r): return quasiSelect(A, p, r-1, i)
        if (i>r): return quasiSelect(A, r+1, q, i)
    }
}
```

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 il 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) -> M
4. Si usa M come pivot per il partition

```
Select(A, p, q){
    if(p=q): return A[p]
    else{
        M = findMedian(A, p, q) // And copy in in B[i]
        swap(A, M, q)
        r = partition(A, p, q)
        if (i=r): return A[r]
        if (i<r): return quasiSelect(A, p, r-1, i)
        if (i>r): return quasiSelect(A, r+1, q, i)
    }
}
```

Complessità

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