Quick Sort

mercoledì 1 giugno 2022 17:22

$\theta(n \log n)$
$O(n^2)$

- Algoritmo in loco -> Non abbiamo bisogno di array di supporto
- Algoritmo non stabile
- E' molto difficile che raggiunga n^2

Divide:

- Diviso in 2 parti non necessariamente uguali
  1) Numeri "piccolo"
  2) Numeri "grandi"

  E decide se piccolo/grande prendendo elemento di riferimento: PIVOT
  Metto tutti i valori minori a sinistra, e tutti maggiori a destra (Partition)

Impera:

- Ordina la prima parte, e la seconda parte

Combina:

- Affianca i due array -> Non fa niente

```
QuickSort(A[], Inizio, Fine):
    If inizio < fine:
        Q = Partizione(A, inizio, fine)
        QuickSort(A, inizio, Q)
        QuickSort(A, Q+1, fine)

Int Partizione(A[], I, F):
    Pivot = A[I]
    SX = I-1, DX = F+1
    While sx < dx:
        Do:
            Sx++
        While A[sx] < pivot

        Do:
            Dx--
        While A[dx] < pivot

        If sx < dx:
            App = a[sx]
            A[sx] = a[dx]
            A[dx] = app
    Return dx
```

$T_{partizione}(n) = \theta(n)$
$T(n) = \begin{cases} \theta(1) \\ \theta(n) + T(q) + T(n-q) \end{cases}$
$1 \leq q \leq n-1$

Quick sort randomizzato:

- Come quick sort, solo che il pivot non è il primo valore
  Ma uno casuale dell'array.

Quick sort lomuto:

- Il pivot è l'ultimo elemento dell'array
- Per la decisione dello scambio il procedimento è il seguente:
  - Sx e Dx partono a sinistra
  - Dx controlla se quello che stà controllando è più piccolo del pivot
    se si lo porta ad Sx, e se c'è stato uno scambio Sx si ingrandisce.