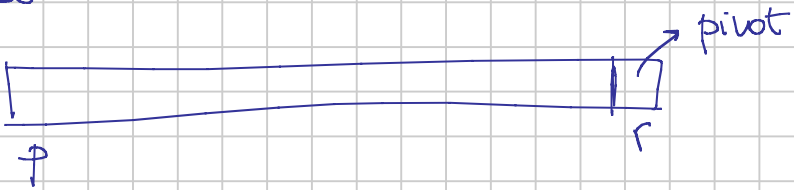
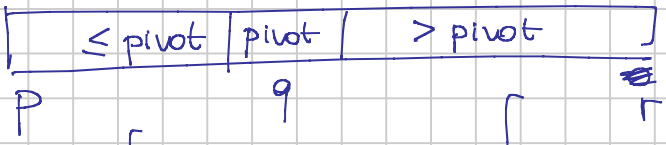
Titolo nota 16/03/2016

QuickSort (Hoare 1962)

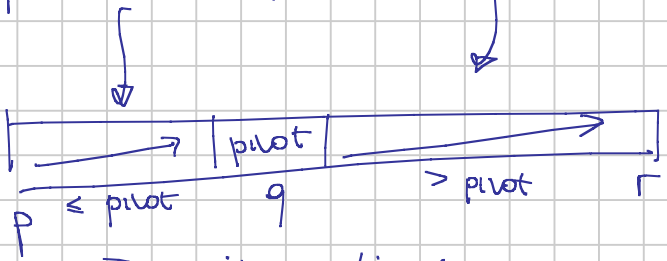
Divide et Impera

DIVIDE a

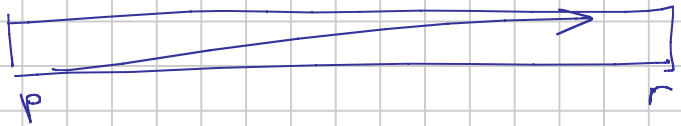




RISOLVI RICORSIVAMENTE



COMBINA: l'array è già ordinato



Titolo nota 16/03/2016

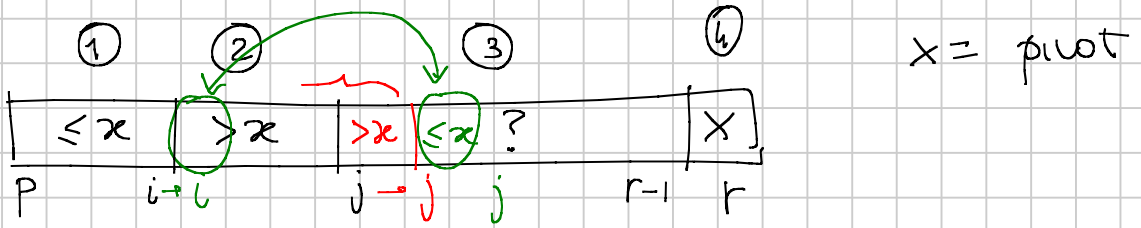
```
QuickSort(a, p, r)
if (p<r) {  // ci sono almeno due elementi in a[p..r]
    q = PARTITION(a, p, r)
    QuickSort(a, p, q-1);
    QuickSort(a, q+1, r);
}
```

CHIUSURA

DIVISIONE

RICORSIONE

Partition



x = pivot

All'inizio: ① e ② sono vuote, $i = p-1$, $j = p$

Titolo nota 16/03/2016

PARTITION (a, p, r)

```
Θ(1)  { x = a[r]          // x è il pivot
        i = p-1           // all'inizio la regione ① è vuota
        for j = p to r-1 {
Θ(1)      if (a[j] ≤ x) {
              i++;
              scambia (a[i], a[j]);
          }
        }
Θ(1)  { scambio (a[i+1], a[r])
        return i+1;
      }
```

Analisi di complessità

$n = r - p + 1$

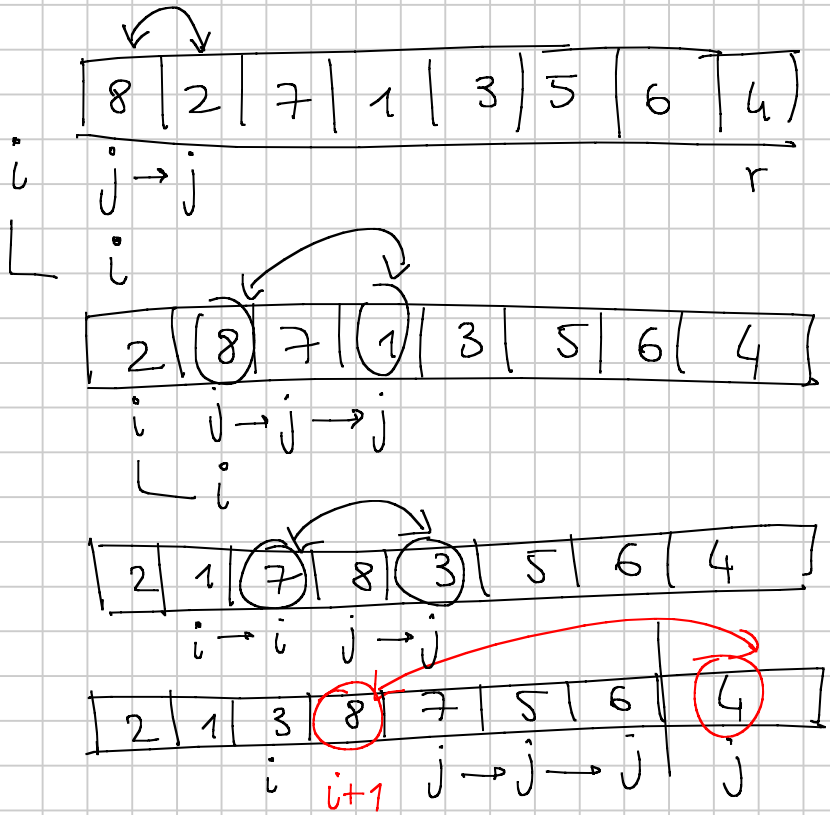
$T(n) = \Theta(n)$

$C(n) = \#$ confronti $= n - 1 = r - p.$

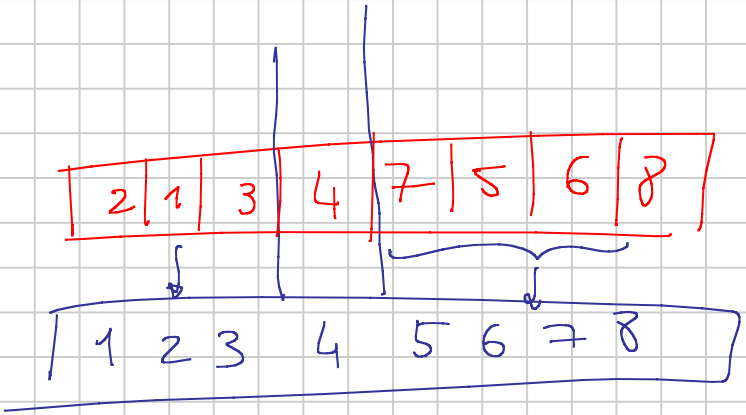
GLI ELEMENTI SI CONFRONTANO SOLO COL PIVOT (MAI TRA LORO)

Titolo nota 16/03/2016

Simulazione



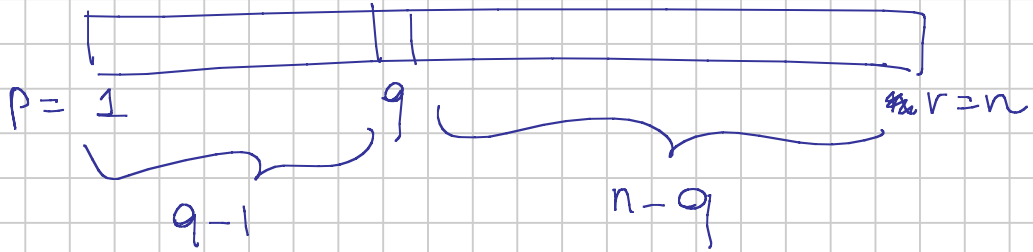
pivot = x = 4



Titolo nota 16/03/2016

Analisi di Complessità del QuickSort

$$T(n) = \begin{cases} \Theta(1) & n \leq 1 \\ T(q-1) + T(n-q) + \underbrace{\Theta(n)}_{\text{costo di Partition}} & n > 1 \end{cases}$$



Si può dimostrare che $T(n) = O(n^2)$

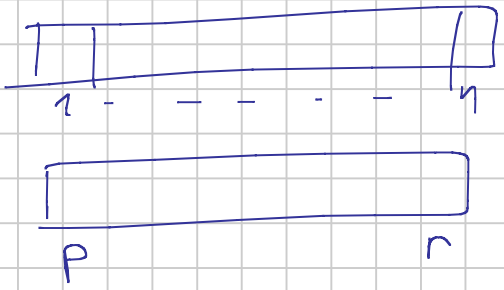
Titolo nota 16/03/2016

CASO PESSIMO

Partizionamento sbilanciato

$q = 1$, $q = n$

($q = p$, $q = r$)



$$T(n) = T(0) + T(n-1) + \Theta(n) = \Theta(1) + T(n-1) + \Theta(n) = T(n-1) + \Theta(n)$$

(pilot : max o min di $A(p \ldots r)$)

Supponiamo che il partizionamento sbilanciato accada ad ogni passo ricorsivo :

$$\boxed{T(n) = T(n-1) + c \cdot n}$$

c = costante

Titolo nota 16/03/2016

$$T(n) = T(n-1) + cn = T(n-2) + c\cdot(n-1) + cn =$$

$$= T(n-3) + c(n-2) + c(n-1) + cn = T(n-i) + \sum_{j=0}^{i-1} c(n-j)$$

$$\underset{i=n-1}{=} T(n-(n-1)) + \sum_{j=0}^{n-2} c\cdot(n-j) = T(1) + c\sum_{j=0}^{n-2}(n-j)$$

$$= \Theta(1) + c\cdot\left(n + (n-1) + (n-2) + \dots + 2\right) = \Theta(1) + c\cdot\left(\frac{n(n+1)}{2} - 1\right)$$

$$= \Theta(1) + \Theta(n^2) = \Theta(n^2)$$

$\Theta(n^2)$

Titolo nota 16/03/2016

Confronti al caso peggiore

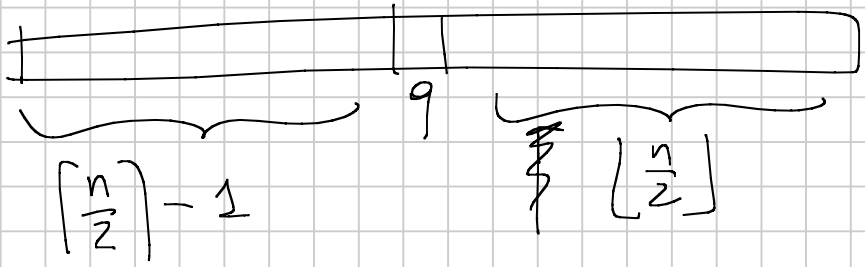
Dim. segmento di array	# confronti
n	$n-1$
$n-1$	$n-2$
$n-2$	$n-3$
⋮	
2	1

$$C(n) = \frac{n(n-1)}{2} = \binom{n}{2} = \#\text{coppie}!$$

Titolo nota 16/03/2016

QuickSort: Caso ottimo

Partition divide l'array in modo bilanciato



$$T(n) = T\left(\lceil \tfrac{n}{2} \rceil - 1\right) + T\left(\lfloor \tfrac{n}{2} \rfloor\right) + \Theta(n)$$

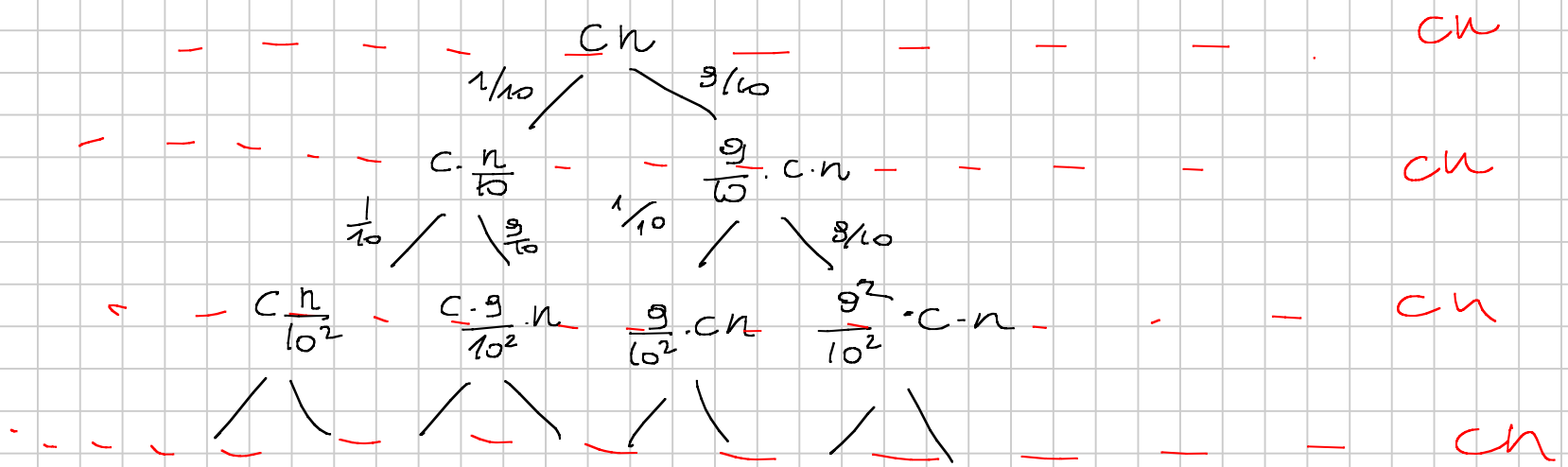
$$\leq 2T\left(\tfrac{n}{2}\right) + \Theta(n) = \Theta(n \log n)$$

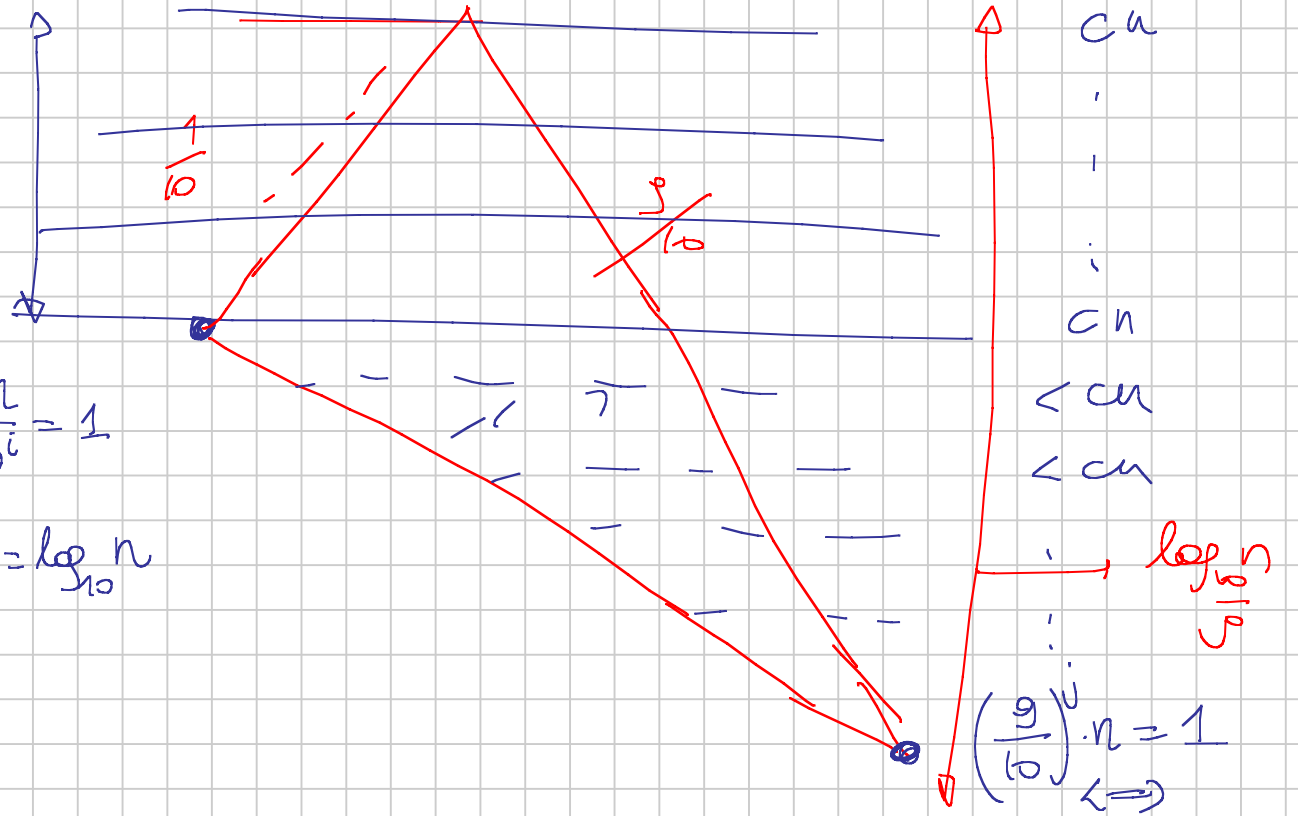
(come il merge sort)

Titolo nota 16/03/2016

Analisi al caso medio

$$T(n) = T\left(\frac{n}{10}\right) + T\left(\frac{9}{10}n\right) + c \cdot n$$





cn
$\vdots$
cn
$< cn$
$< cn$
$\vdots$ → $\log_{\frac{10}{9}} n$
$\vdots$
$\left(\frac{9}{10}\right)^i \cdot n = 1$
$\Leftrightarrow$
$i = \log_{\frac{10}{9}} n$

$T(n) \leq$ # complessità di livelli $\cdot$ costo max dei livelli

$= \left(\log_{\frac{10}{9}} n\right) \cdot (cn)$

$= \Theta(n \log n)$

$\Rightarrow$

$T(n) \leq \Theta(n \log n)$

$< cn$ $\boxed{T(n) = O(n \log n)}$

$\forall$ partizione di "proporzionalità costante" 1 a k

$$\left(\frac{1}{k} \cdot n \qquad n \cdot \frac{k-1}{k} = \left(1 - \frac{1}{k}\right) \cdot n \right)$$

il costo è $T(n) = O(n \log n)$

RANDOM QUICK-SORT

idea: scelgo casualmente il pivot $a[p \dots r]$

```
RANDOMIZED-QS (a, p, r)
  if (p<r) {
      q = RANDOMIZED-PARTITION (a, p, r)
      RANDOMIZED-QS (a, p, q-1)
      RANDOMIZED-QS (a, q+1, r)
  }

RANDOMIZED-PARTITION (a, p, r)
    i = RANDOM [p, r]
    Scambia (a[i] ⇄ a[r])
    Partition (a, p, r)
```

} $\Theta(n)$

Analisi del caso medio (Random QS)

$$T(n) = \Theta(X)$$

X = # complessivo di confronti eseguiti da tutte le chiamate di Partition durante l'esecuzione di QS.

Costo del caso medio $\longrightarrow$ valor medio di X

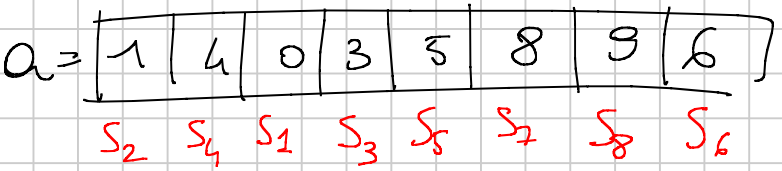
$$E[X]$$

Ipotesi: elementi in a DISTINTI

NOTAZIONE

$1 \le i \le n$ $\quad S_i$ = l'i-esimo elemento più piccolo di Q
[elemento di rango i]

S_1 = MINIMO dell'array
S_2 = secondo minimo
⋮
$S_{n/2}$ = mediano
⋮
S_n = massimo



quando A sarà ordinato

$\forall i \quad A[i] = S_i$

P_{ij} = Probabilità che S_i e S_j confrontino tra loro in una esecuzione di QuickSort

$$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P_{ij}$$

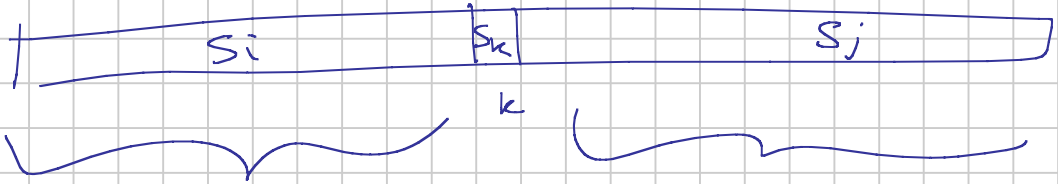
Sommo su tutte le coppie (S_i, S_j) con $i < j$

- Gli elementi si confrontano al massimo una volta (quando uno dei due è il pivot)
- due elementi non si confrontano se sono mandati se una chiamata di Partition li manda in due partizioni diverse dell'array

$\forall i,j \quad S_{ij} = \{ s_i, s_{i+1}, \ldots, s_j \}$

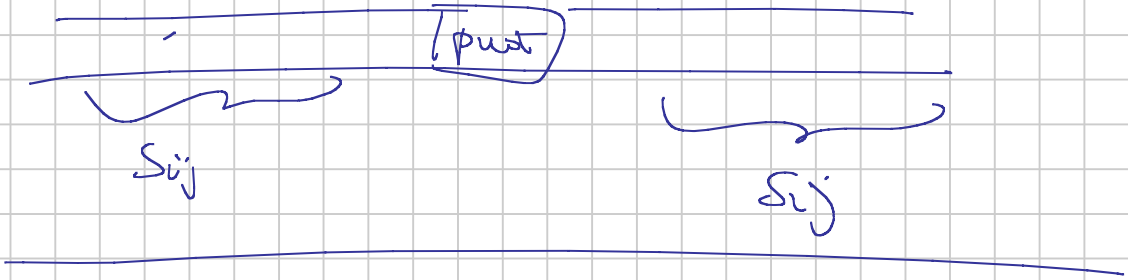
① se il primo elemento scelto come pivot in S_{ij} è s_i oppure s_j sicuramente s_i e s_j si confrontano

② se il primo elemento scelto come pivot in S_{ij} è s_k ~~[illegible]~~ $i < k < j$, s_i e s_j NON SI CONFRONTERANNO MAI PIÙ



OSSERVAZIONE

Se pivot $\notin S_{ij}$



Conclusione

S_i e S_j si confrontano $\iff$ il primo pivot scelto nell'insieme $S_{ij} = \{S_i, S_{i+1}, \dots, S_j\}$ è S_i o S_j

$$P_{ij} = \frac{2}{j - i + 1}$$

$// \; |S_{ij}| = j - i + 1$

$$P = \frac{\text{casi favorevoli}}{\text{casi possibili}}$$