Random Quick Sort

Valore atteso del # di confronti

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

$p_{ij} = \text{Prob.}\{ s_i \text{ si confronta con } s_j \}$

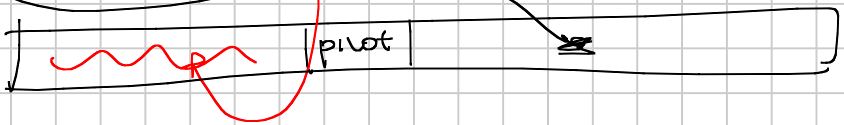
$i < j$

$$S_{ij} = \{ s_i, s_{i+1}, \dots, s_j \}$$

1° caso

pivot $\notin S_{ij}$

pivot $< s_i$ oppure pivot $> s_j$



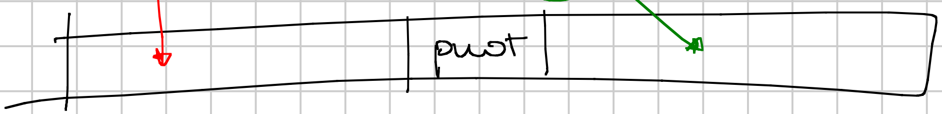
II° caso

$pivot \in S_{ij}$

Consideriamo il PRIMO pivot scelto in S_{ij}

$\boxed{2a}$ $pivot \in S_{ij}$, $pivot \neq S_i$ e $pivot \neq S_j$

$S_i < pivot < S_j$



S_i e S_j NON SI CONFRONTANO MAI

$\boxed{2b}$ $pivot = S_i$ oppure $pivot = S_j$

S_i e S_j si confrontano

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

(casi favorevoli)

(casi possibili, equiprobabili)

per $j = i+1$ $\qquad P_{i,i+1} = \frac{2}{i+1-i+1} = \frac{2}{2} = 1$

$$E(X) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P_{ij} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1}$$

$$< 2 \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{1}{k} = 2 \sum_{i=1}^{n-1} H_n = 2 \sum_{i=1}^{n-1} \Theta(\log n) = \Theta(n \log n)$$

$$H_n = \sum_{k=1}^{n} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$$

n_esimo numero armonico

$$H_n = \ln n + \Theta(1) = \Theta(\log n)$$

$$E[X] \leq \Theta(n \log n)$$

$$\Rightarrow \quad E[X] = O(n \log n)$$

R.QS è ottimo nel caso medio (in tempo)

<u>Complessità in spazio RQS</u> $= \begin{cases} \Theta(\log n) & \text{caso medio} \\ \Theta(n) & \text{caso peggiore} \end{cases}$

↳ gestione della ricorsione.

Selezione dell' elemento di rango i da un array non ordinato

IPOTESI: elementi distinti.

rango → i-esimo elemento più piccolo

[i-esima statistica d'ordine]

$i = 1$ → MINIMO

$i = n$ → MASSIMO

n dispari

$i = \lceil \frac{n}{2} \rceil$ → MEDIANA

n pari

$i = \frac{n}{2}$ MEDIANA INFERIORE

$i = \frac{n}{2} + 1$ " SUPERIORE

I° approccio (ineff.)

ordino a
return a[i]

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

È possibile selezionare l'elemento di rango i in tempo $\Theta(n)$

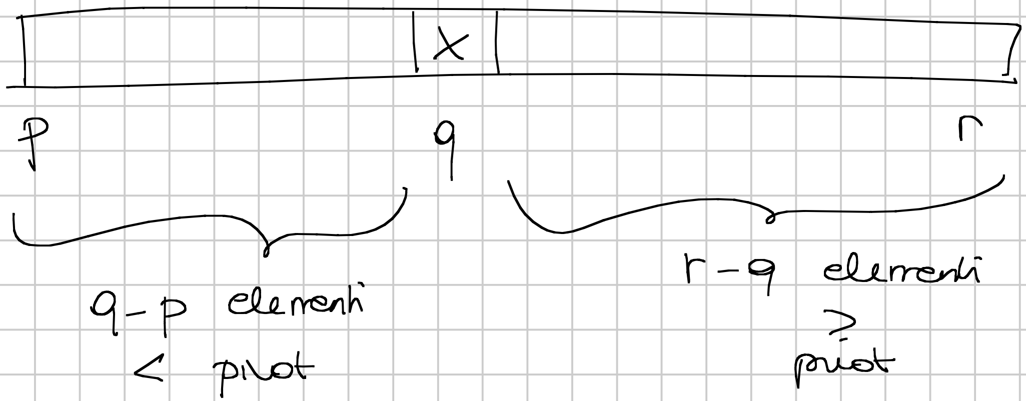
↳ RANDOMIZED_SELECT

$$T(n) = \begin{cases} \Theta(n^2) & \text{caso pessimo} \\ \Theta(n) & \text{caso medio} \end{cases}$$

Divide-et-impera

Rango i

x = pivot



$k = q-p+1$

⇒ il pivot è l'elemento di rango k

$\boxed{k=i}$ il pivot è l'elemento di rango i

$\boxed{i<k}$ l'elemento di rango i si trova in $A[p \dots q-1]$

$\boxed{i>k}$ l'elemento di rango i si trova in $A[q+1 \dots r]$, ed è l'elemento di rango $i-k$ in $A[q+1 \dots r]$

Titolo nota 26/02/2016

```
RANDOMIZED-SELECT ( A, p, r, i )   //   p ≤ i ≤ r
  if (p == r) return A[p];
  else
        q = RANDOMIZED-PARTITION (A, p, r )
        k = q - p + 1
        if (i == k) return A[q]   // il pivot è l'elemento di rango i
        else if (i < k) return RANDOMIZED-SELECT (A, p, q-1, i)
             else return RANDOMIZED-SELECT (A, q+1, r, i-k);
```

Caso pessimo

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

Caso ottimo

$$T(n) = \underset{1}{} T\left(\frac{n}{2}\right) + \Theta(n)$$

$a = 1$
$b = 2$ $\qquad n^{\log_b a} = n^{\log_2 1} = 1$

$f(n) = \Theta(n)$

$f(n) = \Omega\left(n^{\log_2 1 + \varepsilon}\right) \qquad 0 < \varepsilon \leq 1$

$\Rightarrow$ III° caso $\boxed{T(n) = \Theta(n)}$

Caso medio

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

NON LO DIMOSTRIAMO

$\frac{k-1}{k} n$ $\qquad$ $\frac{1}{k} n$

intuizione

k = costante

$$T(n) = T\left(\frac{k-1}{k} \cdot n\right) + \Theta(n)$$

$a = 1$

$b = \frac{k}{k-1}$

$$n^{\log_{\frac{k}{k-1}} 1} = n^0 = 1$$

III° caso

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

ESERCITAZIONE

(1) $T(n) = 3T\left(\frac{n}{4}\right) + n \log n$

$a = 3 \quad b = 4 \quad n^{\log_4 3} \qquad \log_4 3 < 1$

$f(n) = n^{1} \log n = \Omega\left(n^{\log_4 3 + \varepsilon}\right) \qquad 0 < \varepsilon \leq 1 - \log_4 3$

$1 \geq \log_4 3 + \varepsilon$

Cond. di regolarità

$a f\left(\frac{n}{b}\right) \leq c f(n) \qquad c < 1$

$3\left(\frac{n}{4}\right) \log\left(\frac{n}{4}\right) = \frac{3}{4} \cdot n \log \frac{n}{4} < \frac{3}{4} \cdot \underbrace{n \log n}_{f(n)} = \frac{3}{4} \cdot f(n) \qquad c = \frac{3}{4}$

$\Rightarrow T(n) = \Theta(f(n)) = \Theta(n \log n)$

(2) $T(n) = T\left(\frac{2}{3}n\right) + 1$

$T(n) = T\left(\frac{n}{b}\right) + 1 \quad \forall\, b > 1$ costante

$a = 1 \qquad b = \frac{3}{2}$

$n^{\log_b a} = n^{\log_{3/2} 1} = 1$

Th. principale II° caso $\quad T(n) = \Theta(\log n)$

(3) $T(n) = 2T\left(\frac{n}{2}\right) + n \log n$

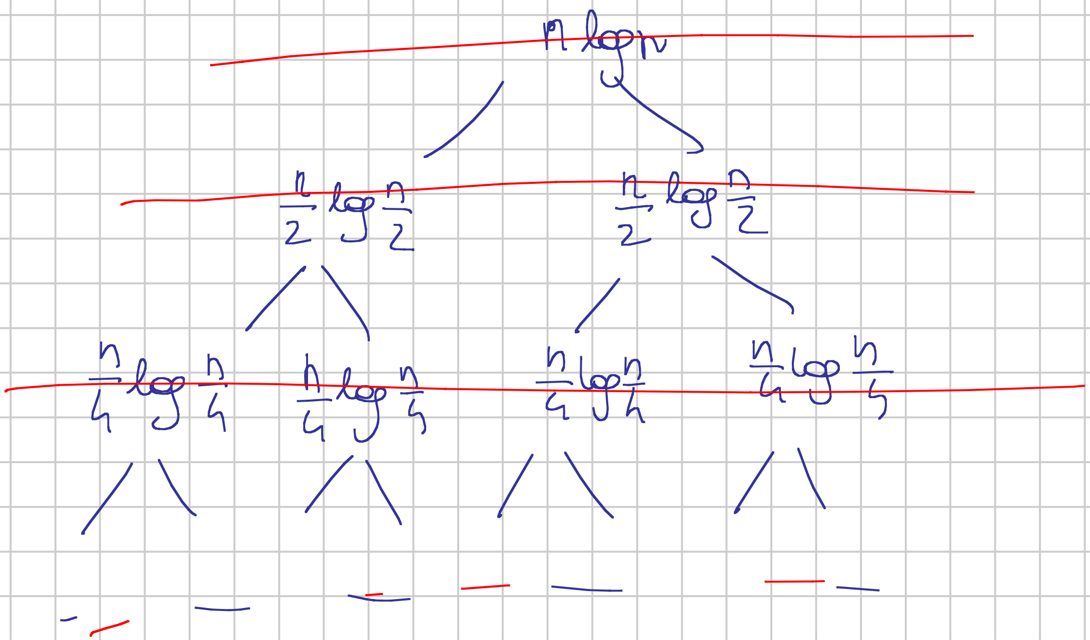
$a = 2 \quad b = 2 \qquad n^{\log_b a} = n$

$f(n)$ cresce più velocemente di $n^{\log_b a}$ solo per un fattore logaritmico

il th. principale NON si applica

Titolo nota 26/02/2016

$$T(n) = 2T\left(\frac{n}{2}\right) + n\log n$$



$n\log n \leq n\log n$

$n\log\frac{n}{2} \leq n\log n$

$n\log\frac{n}{4} \leq n\log n$

$n\log\frac{n}{2^i} \leq n\log n$

$T(n) \leq \#$ livelli $\cdot$ costo max di ciascun livello

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

$T(n) = O(n\log^2 n)$

$\frac{n}{2^i} = 1$ $\quad i = \log_2 n$

l'albero contiene $\log n$ livelli

ESERCIZIO 2 (Più 0 che 1)

0 1 1 0 | 1 1 1 0

1 -1 -1 1 | -1 -1 -1 1

0 0 | -2 0

0 -2

-2

diff == 0	#0 = #1
diff < 0	#0 < #1
diff > 0	#0 > #1

D&I → calcolo diff tra #0 e #1

diff = #0 − #1

```
PiùOche1 (a, sx, dx)   // sx ≤ dx
```

Ch.
```
if (sx==dx) {
      if (a[sx]==0) return 1;
      else return -1;
}
```

DIV $cx = \frac{sx+dx}{2}$

RIC diffSx = PiùOche1 (a, sx, cx)

RIC diffDx = PiùOche1 (a, cx+1, dx).

RIC return $\boxed{\text{diffSx + diffDx;}}$

$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(1) = \Theta(n)$

Main (a)

```
diff = PiùOccl(a, 1, n);
if (diff > 0) return "Più 0"
else if (diff < 0) return "Più 1"
else return "0 e 1 in uguel numero"
```

Dim. di correttezza

Per induzione su n

BASE $n=1$ OK ✓

PASSO INDUTTIVO

i.i. algoritmo corretto su istanze di dim $< n$

$$\text{diff} = \text{diff}\,Sx + \text{diff}\,Dx \underset{i.i}{=} \left(\#0_{Sx} - \#1_{Sx}\right) + \left(\#0_{Dx} - \#1_{Dx}\right)$$

$$= \underbrace{\#0_{Sx} + \#0_{Dx}} - \underbrace{\#1_{Sx} - \#1_{Dx}} = \#0 - \#1$$

✓

ESERCIZIO 4 (MergeSort3)

MergeSort3 (a, sx, dx)

if (sx == dx - 1) { // ci sono 2 elementi in a[sx, dx]

if (a[sx] > a[dx]) scambio (a[sx], a[dx]);

}

else if (sx < dx - 1) // ci sono almeno 3 elementi {

$cx_1 = \frac{2sx + dx}{3}$ // $sx + \frac{dx - sx}{3}$

$cx_2 = \frac{sx + 2dx}{3}$ // $sx + \frac{2}{3}(dx - sx)$

MergeSort3 (a, sx, cx1)
MergeSort3 (a, cx1+1, cx2)
MergeSort3 (a, cx2+1, dx)
Merge3 (a, sx, cx1, cx2, dx)

}

$$T(n) = 3T\left(\frac{n}{3}\right) + \Theta(n)$$

$$= \Theta(n \log n)$$