Titolo nota 03/03/2016

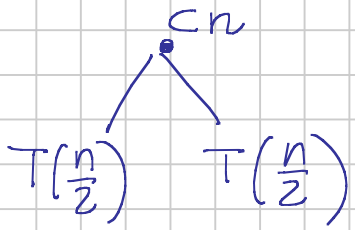
MergeSort : analisi di complessità

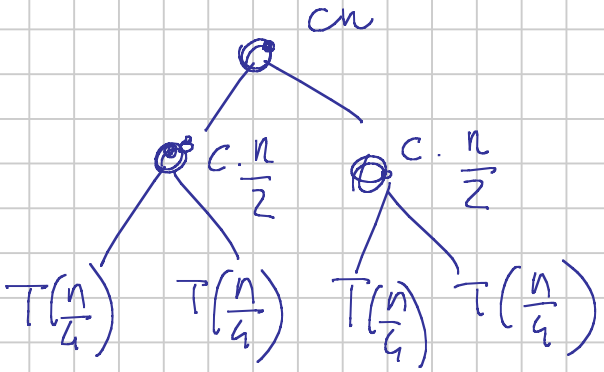
$$T(n) = \begin{cases} c_0 & n \leq 1 \\ 2T\left(\frac{n}{2}\right) + cn & n > 1 \end{cases}$$

Titolo nota 26/02/2016

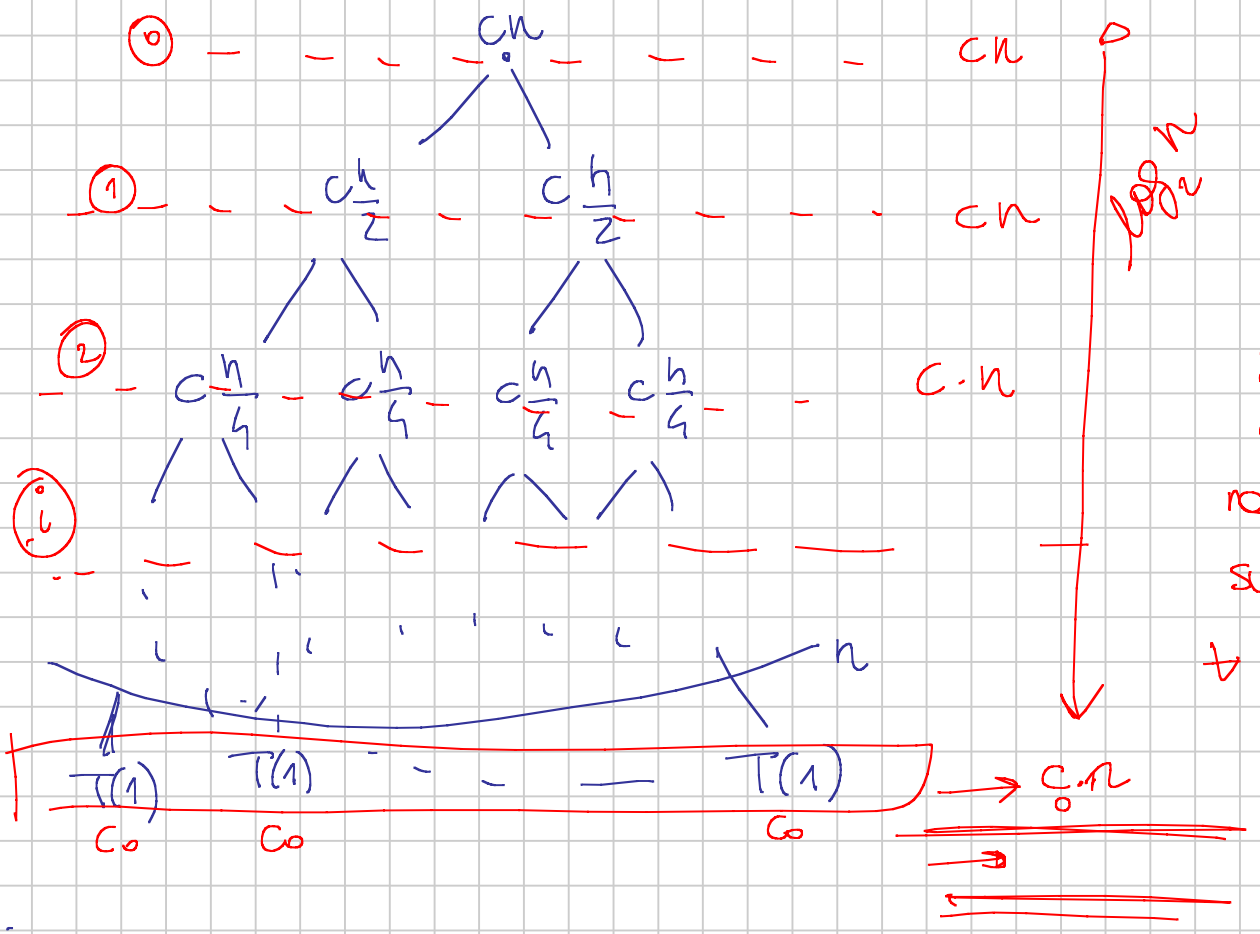
Albero di ricorrenza

$T(n)$





Titolo nota 26/02/2016



$\log_2 n$

livelli = $\log_2 n + 1$

Sul livello i dell'albero ci sono 2^i nodi, che rappresentano le 2^i chiamate su partizioni dell'array di dim. $\frac{n}{2^i}$.

$\forall$ nodo sul livello $\rightsquigarrow c \cdot \frac{n}{2^i}$

$2^i * c\frac{n}{2^i} = cn$

Titolo nota 26/02/2016

$$\overline{T}(n) = c \cdot n \cdot \log_2 n + n \cdot c_0 = \Theta(n \log n)$$

$(c \cdot n \times \#\text{livelli albero})$

$$\frac{n}{2^i} = 1 \qquad n = 2^i$$

$$2^i = n \qquad \log_2 2^i = \log_2 n \Rightarrow \boxed{i = \log_2 n}$$

Titolo nota 26/02/2016

Metodo iterativo

$$T(n) = 2T\left(\frac{n}{2}\right) + cn = 2\cdot\left[2T\left(\frac{n}{4}\right) + c\cdot\frac{n}{2}\right] + cn =$$

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

$$= 2^2\cdot\left[2T\left(\frac{n}{2^3}\right) + c\cdot\frac{n}{2^2}\right] + 2cn = 2^3\cdot T\left(\frac{n}{2^3}\right) + 3cn =$$

$$= \dots = 2^i T\left(\frac{n}{2^i}\right) + icn \overset{i=\log n}{=\!=} 2^{\log_2 n} T\left(\frac{n}{2^{\log_2 n}}\right) + c\cdot n\cdot\log_2 n$$

$$= n\cdot T(1) + cn\log_2 n = c_0 n + cn\log n = \Theta(n\log n)$$

Titolo nota 26/02/2016

Merge Sort ← Commodor 64 1982

0.5 MIPS (milioni di istruzioni al secondo)

Insertion Sort ← Intel 7 (2013) 190000 MIPS (380000 volte più veloce del Commodor 64)

$n = 10^7$

I.S. n^2 operazioni

$$\text{tempo} = \frac{n^2}{\#\text{op. al sec}} = \frac{(10^7)^2}{190000 \cdot 10^6} = 526 \text{ sec} \simeq 9 \text{ minuti}$$

Titolo nota 26/02/2016

Merge Sort

$$\text{tempo} = \frac{(n \cdot \log_2 n)}{\#\text{op. al sec}} = \frac{(10^7 \cdot 23)}{0.5 \cdot 10^6} \simeq 460 \text{ sec} \simeq 8 \text{ minuti}$$

$n = 10^8$

Intel 7
Ins. Sort

$$t = \frac{(10^8)^2}{150000 \cdot 10^6} \simeq 15 \text{ ore}$$

Commodore 64
Merge Sort

$$t = \frac{(10^8) * 27}{0.5 \cdot 10^6} \simeq 1,5 \text{ ore}$$

→ 10 volte più veloce!

Titolo nota 26/02/2016

$n = 10^9$

Ins. Sort su Intel 7 → 61 giorni

Merge Sort sul Commodor 64 → 16 ore

Merge Sort su intel 7 < 1 sec.

Titolo nota 26/02/2016

Merge Sort

$T(n) = \Theta(n \log n)$ caso ottimo, medio, peggiore

$S(n) = \Theta(n)$ → array di appoggio

Titolo nota 26/02/2016

ESERCIZI

Ricerca Sequenziale (a, k) // n = dim. di a

$\Theta(1)$
```
i = 1;
posizione = -1;
while (i <= n && posizione == -1) {        Θ(1)
    if (a[i] == k) posizione = i;          Θ(1)
    else i++;
}
```
$\Theta(1)$
```
return posizione;
```

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

t_w = # volte che si esegue il while

Titolo nota 26/02/2016

$$T(n) = \Theta(t_w) = \begin{cases} \Theta(1) \\ \Theta(n) \end{cases}$$

caso ottimo $\dfrac{t_w = 1}{a[1] == k}$

caso pessimo $\dfrac{t_w = n}{k \notin a}$

$$C(n) = \begin{cases} 1 \\ n \end{cases}$$

$C(n)$ = # confronti

caso ottimo

caso pessimo

Titolo nota 26/02/2016

Confronti al caso medio

k occorre in pos. 1 → 1 conf.

k è in pos. 2 → 2 conf.

k è in pos. i → i

k è in pos. n → n

k non è in Q → n

n + 1 casi

$$C(n) = \underset{\text{medio}}{} \frac{1 + 2 + 3 + \dots + i + \dots + n + n}{n+1} =$$

$$= \frac{1}{n+1}\left[\sum_{i=1}^{n} i + n\right] = \frac{1}{n+1}\left(\frac{n(n+1)}{2} + n\right) = \frac{n}{2} + \frac{n}{n+1} \simeq \frac{n}{2} + 1$$

Titolo nota 26/02/2016

RicercaBinaria (a, sx, dx, k)

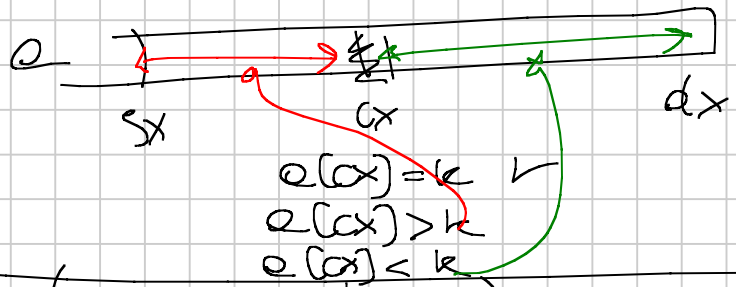
```
if (sx > dx) return -1;
if (sx == dx) {
        if (a[sx] == k) return sx;
        else return -1;
}
cx = (sx + dx) / 2
if (a[cx] == k) return cx;
if (a[cx] < k) return RicercaBinaria(a, cx+1, dx, k);
else return RicercaBinaria(a, sx, cx-1, k);
```

$\Theta(1)$ ←
$\Theta(1)$ ←

$T(n/2)$
$T(n/2)$

RicercaBinaria (a, 1, n, k)

a è ORDINATO



Titolo nota 26/02/2016

14	43	76	100	115	290	500	571
1	2	3	4	5	6	7	8

$k = 76$

sx	dx	cx	confronto	
1	8	4	76 : 100	$k <$
1	3	2	76 : 43	$k >$
3	3		76	= trovato (return 3)

$k = 300$

sx	dx	cx	confronto	
1	8	4	300 : 100	$k >$
5	8	6	300 : 290	$k >$
7	8	7	300 : 500	$k <$
7	6		=	NON TROVATO (return −1)

Titolo nota 26/02/2016

$$T(n) = \begin{cases} c_0 & n \leq 1 \\ T\left(\frac{n}{2}\right) + c_1 & n > 1 \end{cases}$$

Analisi del caso peggiore

↓

quando $k \neq a$

$$T(n) = T\left(\frac{n}{2}\right) + c_1 = \left[T\left(\frac{n}{4}\right) + c_1\right] + c_1 = T\left(\frac{n}{4}\right) + 2c_1 =$$

$$= \left[T\left(\frac{n}{8}\right) + c_1\right] + 2c_1 = T\left(\frac{n}{8}\right) + 3c_1 = T\left(\frac{n}{2^3}\right) + 3c_1 = \dots$$

$$\dots = T\left(\frac{n}{2^i}\right) + i c_1 = \quad i = \log_2 n$$

$$T\left(\frac{n}{2^{\log_2 n}}\right) + c_1 \cdot \log_2 n = T(1) + c_1 \log_2 n =$$

$$= c_0 + c_1 \log_2 n = \Theta(\log n)$$

Titolo nota 26/02/2016

$$T(n) = \begin{cases} \Theta(1) & \text{caso ottimo} \\ \Theta(\log n) & \text{caso peggiore} \end{cases} = O(\log n)$$

Ricerca Binaria

$$C(n) = \begin{cases} 1 & \text{caso ottimo} \\ \log_2 n + 1 & \text{caso peggiore} \end{cases}$$

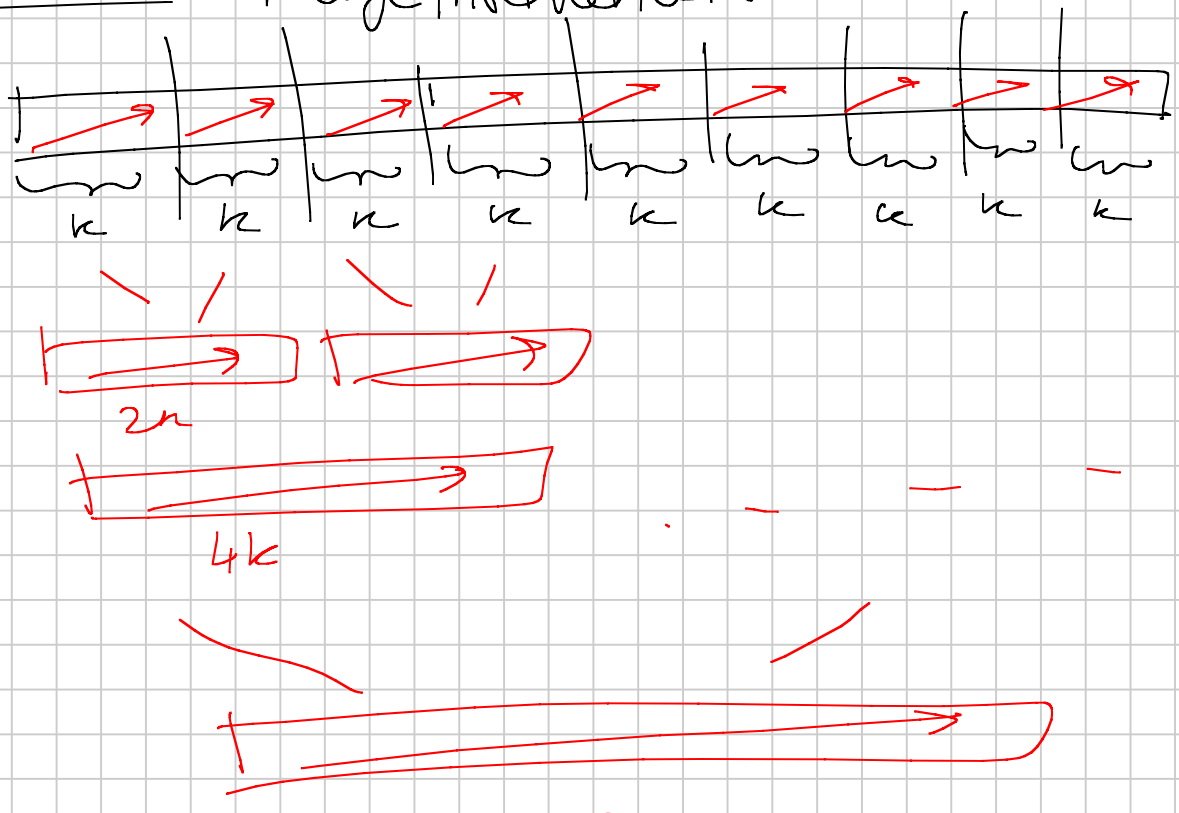
$\frac{\#\text{confronti}}{\#\text{elementi dell'array con cui si confronta } k}$

$n = 2^t$

Titolo nota 26/02/2016

Esercizio 3

Merge Insertion Sort



Titolo nota 26/02/2016

MergeInsSort (a, sx, dx, k)

Base

if ((dx − sx + 1) ≤ k) InsertionSort (a, sx, dx)

Divisione

else { cx = (sx + dx) / 2;

Ricorsione

MergeInsSort (a, sx, cx, k);
MergeInsSort (a, cx+1, dx, k);

Combinazione

Merge (a, sx, cx, dx);

}

Titolo nota 26/02/2016

$$T(n) = \begin{cases} \Theta(n^2) & n \leq k \\ 2\,T\left(\frac{n}{2}\right) + \Theta(n) & n > k \end{cases}$$

$$T(n) = \begin{cases} c_0 n^2 & n \leq k \\ 2T\left(\frac{n}{2}\right) + cn & n > k \end{cases}$$

Titolo nota 26/02/2016

$$T(n) = 2T\left(\frac{n}{2}\right) + cn = 2\left[2T\left(\frac{n}{4}\right) + c\cdot\frac{n}{2}\right] + cn = 4T\left(\frac{n}{4}\right) + 2cn =$$

$$= 8T\left(\frac{n}{8}\right) + 3cn = \dots = 2^i T\left(\frac{n}{2^i}\right) + i\,cn \underset{i=\log_2\frac{n}{k}}{=} 2^{\log_2\frac{n}{k}} \cdot T\left(\frac{n}{2^{\log_2\frac{n}{k}}}\right) + cn\log_2\frac{n}{\quad}$$

quando $\frac{n}{2^i}$ diventa $\leq k$ arresto le chiamate ricorsive e ordino con Ins. Sort.

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

$$\Rightarrow \boxed{i = \log_2 \frac{n}{k}}$$

$$= \underset{i=\log_2 \frac{n}{k}}{=} 2^{\log_2 \frac{n}{k}} \cdot T\left(\frac{n}{2^{\log_2 \frac{n}{k}}}\right) + c_1 \cdot n \cdot \log_2 \frac{n}{k} =$$

$$= \frac{n}{k} \cdot T\left(\frac{n}{n/k}\right) + cn \log_2 \frac{n}{k} = \frac{n}{k} \cdot T(k) + cn \log \frac{n}{k} =$$

$$= \frac{n}{k} \cdot c_0 \cdot k^2 + c \cdot n \log_2 \frac{n}{k} = \Theta\left(nk + n \log \frac{n}{k}\right)$$