

# Analisi – Ordinamento funzioni

Ordinare le seguenti funzioni in accordo alla loro complessità asintotica. Si scriva  $f(n) < g(n)$  se  $O(f(n)) \subset O(g(n))$ . Si scriva  $f(n) = g(n)$  se  $O(f(n)) = O(g(n))$ , ovvero se  $f(n) = \Theta(g(n))$ .

$$f_1(n) = 2^{n+2}$$

$$f_2(n) = \log^2 n$$

$$f_3(n) = \log_n(n \cdot (\sqrt{n})^2) + \frac{1}{n^2}$$

$$f_4(n) = 3n^{0.5}$$

$$f_5(n) = 16^{n/4}$$

$$f_6(n) = 2\sqrt{n} + 4n^{1/4} + 8n^{1/8} + 16n^{1/16}$$

$$f_7(n) = \sqrt{(\log n)(\log n)}$$

$$f_8(n) = \frac{n^3}{(n+1)(n+3)}$$

$$f_9(n) = 2^n$$

## Analisi – MergeSortK

Si consideri una variante di MergeSort chiamata MergeSortK che, invece di suddividere l'array da ordinare in 2 parti, lo suddivide in  $k$  parti, ri-ordina ognuna di esse applicando ricorsivamente MergeSortK, e le riunifica usando un'opportuna variante MergeK di Merge, che fonde  $k$  sottoarray invece di 2.

- (1) Abbozzate il codice di MergeSortK e di MergeK (fatevi solo un'idea, ci sono molti dettagli nella gestione degli indici)
- (2) Scrivete la relazione di ricorrenza di MergeSortK

## Ripasso del metodo dell'albero di ricorsione

## Albero di ricorsione su MergeSortK

Calcolate la complessità computazionale delle seguenti funzioni, utilizzando il metodo dell'albero di ricorsione

$$T(n) = \begin{cases} 2T(n/2) + 2n & n > 1 \\ 1 & n = 1 \end{cases}$$

$$T(n) = \begin{cases} 3T(n/3) + 3n & n > 1 \\ 1 & n = 1 \end{cases}$$

$$T(n) = \begin{cases} kT(n/k) + kn & n > 1 \\ 1 & n = 1 \end{cases}$$

Spoiler alert!

# Analisi – Ordinamento funzioni

Le funzioni da ordinare:

$$f_1(n) = 2^{n+2} = 4 \cdot 2^n = \Theta(2^n)$$

$$f_2(n) = \log^2 n = \Theta(\log^2 n)$$

$$f_3(n) = \log_n(n \cdot (\sqrt{n})^2) + \frac{1}{n^2} = (\log_n n^2) + 1/n^2 = 2 + 1/n^2 = \Theta(1)$$

$$f_4(n) = 3n^{0.5} = \Theta(n^{1/2})$$

$$f_5(n) = 16^{n/4} = (2^4)^{n/4} = 2^{4n/4} = \Theta(2^n)$$

$$f_6(n) = 2\sqrt{n} + 4n^{1/4} + 8n^{1/8} + 16n^{1/16} = \Theta(n^{1/2})$$

$$f_7(n) = \sqrt{(\log n)(\log n)} = \Theta(\log n)$$

$$f_8(n) = \frac{n^3}{(n+1)(n+3)} = \Theta(n)$$

$$f_9(n) = 2^n = \Theta(2^n)$$

Una volta stabilito l'ordine  $\Theta$  delle funzioni, è abbastanza semplice stabilire l'ordine corretto:

$$f_3 < f_7 < f_2 < f_4 = f_6 < f_8 < f_1 = f_5 = f_9$$

# Analisi – MergeSortK

---

```
mergeK(ITEM A[], int[] start, int k)
```

---

```
int[] index = new int[1...k]    % Indici scansione sottovettori
```

```
for i = 1 to k do
```

```
    | index[i] = start[i]
```

```
int subvectors = k                % Numero di sottovettori non vuoti
```

```
int j = start[1]                  % Indice del vettore di appoggio
```

```
while subvectors > 0 do
```

```
    | int m = 1
```

```
    | for i = 2 to k do
```

```
        | if index[m] < index[m + 1] and A[index[i]] < A[index[m]]
```

```
            then
```

```
                | m = i
```

```
    | [...]
```

---

# Analisi – MergeSortK

---

`mergeK(ITEM  $A[]$ , int $[]$   $start$ , int  $k$ )`

---

**while**  $subvectors > 0$  **do**

$[\dots]$   
     $B[j] = A[index[m]]$   
     $j = j + 1$   
     $index[m] = index[m] + 1$   
    **if**  $index[m] == index[m + 1]$  **then**  
         $subvectors = subvectors - 1$

**for**  $j = start[j]$  **to**  $start[k + 1]$  **do**

$A[j] = B[j]$

---



## Analisi – MergeSortK

La funzione di ricorrenza per MergeSortK è la seguente:

$$T(n) = \begin{cases} k \cdot T(n/k) + kn & n > 1 \\ 1 & n = 1 \end{cases}$$

$$T(n) = 2T(n/2) + 2n$$

Livello	Dimensione	Costo chiamata	N. chiamate	Costo livello
0	$n$	$2 \cdot (n)$	1	$2 \cdot n$
1	$n/2$	$2 \cdot (n/2)$	2	$2 \cdot 2 \cdot (n/2)$
2	$n/2^2$	$2 \cdot (n/2^2)$	$2^2$	$2 \cdot 2^2 \cdot (n/2^2)$
...	...	...	...	...
$i$	$n/2^i$	$2 \cdot (n/2^i)$	$2^i$	$2 \cdot 2^i \cdot (n/2^i)$
...	...	...	...	...
$\ell - 1$	$n/2^{\ell-1}$	$2 \cdot (n/2^{\ell-1})$	$2^{\ell-1}$	$2 \cdot 2^{\ell-1} \cdot (n/2^{\ell-1})$
$\ell = \log n$	1	$T(1) = 1$	$2^{\log n}$	$2^{\log n}$

$$\begin{aligned}
 T(n) &= \left( \sum_{i=0}^{\log n - 1} 2 \cdot 2^i \cdot (n/2^i) \right) + 2^{\log n} = \left( 2n \sum_{i=0}^{\log n - 1} \frac{2^i}{2^i} \right) + n \\
 &= 2n \left( \sum_{i=0}^{\log n - 1} 1 \right) + n = 2n \log n + n = \Theta(n \log n)
 \end{aligned}$$

$$T(n) = 3T(n/3) + 3n$$

Livello	Dimensione	Costo chiamata	N. chiamate	Costo livello
0	$n$	$3 \cdot (n)$	1	$3 \cdot n$
1	$n/3$	$3 \cdot (n/3)$	3	$3 \cdot 3 \cdot (n/3)$
2	$n/3^2$	$3 \cdot (n/3^2)$	$3^2$	$3 \cdot 3^2 \cdot (n/3^2)$
...	...	...	...	...
$i$	$n/3^i$	$3 \cdot (n/3^i)$	$3^i$	$3 \cdot 3^i \cdot (n/3^i)$
...	...	...	...	...
$\ell - 1$	$n/3^{\ell-1}$	$3 \cdot (n/3^{\ell-1})$	$3^{\ell-1}$	$3 \cdot 3^{\ell-1} \cdot (n/3^{\ell-1})$
$\ell = \log_3 n$	1	$T(1) = 1$	$3^{\log_3 n}$	$3^{\log_3 n}$

$$\begin{aligned}
 T(n) &= \left( \sum_{i=0}^{\log_3 n - 1} 3 \cdot 3^i \cdot (n/3^i) \right) + 3^{\log_3 n} = \left( 3n \sum_{i=0}^{\log_3 n - 1} \frac{3^i}{3^i} \right) + n \\
 &= 3n \left( \sum_{i=0}^{\log_3 n - 1} 1 \right) + n = 3n \log_3 n + n = \Theta(n \log n)
 \end{aligned}$$

$$T(n) = kT(n/k) + kn$$

Livello	Dimensione	Costo chiamata	N. chiamate	Costo livello
0	$n$	$k \cdot (n)$	1	$k \cdot n$
1	$n/k$	$k \cdot (n/k)$	$k$	$k \cdot k \cdot (n/k)$
2	$n/k^2$	$k \cdot (n/k^2)$	$k^2$	$k \cdot k^2 \cdot (n/k^2)$
...	...	...	...	...
$i$	$n/k^i$	$k \cdot (n/k^i)$	$k^i$	$k \cdot k^i \cdot (n/k^i)$
...	...	...	...	...
$\ell - 1$	$n/k^{\ell-1}$	$k \cdot (n/k^{\ell-1})$	$k^{\ell-1}$	$k \cdot k^{\ell-1} \cdot (n/k^{\ell-1})$
$\ell = \log_k n$	1	$T(1) = 1$	$k^{\log_k n}$	$k^{\log_k n}$

$$\begin{aligned}
 T(n) &= \left( \sum_{i=0}^{\log_k n - 1} k \cdot k^i \cdot (n/k^i) \right) + k^{\log_k n} = \left( kn \sum_{i=0}^{\log_k n - 1} \frac{k^i}{k^i} \right) + n \\
 &= kn \left( \sum_{i=0}^{\log_k n - 1} 1 \right) + n = kn \log_k n + n = \Theta(n \log n)
 \end{aligned}$$