

# 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  $f(n) = \Theta(g(n))$ .

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

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

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

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

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

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

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

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

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

## Ricorrenza $2T(n/8) + 2T(n/4) + n$

Trovare un limite asintotico superiore e un limite asintotico inferiore alla seguente ricorrenza, facendo uso del metodo di sostituzione:

$$T(n) = \begin{cases} 1 & n \leq 1 \\ 2T(n/8) + 2T(n/4) + n & n > 1 \end{cases}$$

# Ricorrenza

Trovare i limiti superiore e inferiori più stretti possibili per la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 2T(\lfloor n/2 \rfloor) + 4T(\lfloor n/4 \rfloor) + 15T(\lfloor n/8 \rfloor) + n^2 & n > 8 \\ 1 & n \leq 8 \end{cases}$$

## Analisi – Algoritmo di selezione deterministico

Si consideri la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} T(\lfloor n/5 \rfloor) + T(\lfloor 7n/10 \rfloor) + \frac{11}{5}n & n > 1 \\ 1 & n \leq 1 \end{cases}$$

Individuare limiti inferiori e superiori tramite il metodo di sostituzione.

## Analisi – MergeSortK

Si supponga di scrivere 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. Come cambia, se cambia, la complessità temporale di MergeSortK rispetto a quella di MergeSort?

# SortinoSort

Il professor Sortino ha inventato un nuovo algoritmo di ordinamento.

- Il vettore di input viene diviso in tre parti, di dimensioni circa  $n/3$ .
- Vengono ordinati ricorsivamente i primi due terzi, i secondi due terzi, e infine di nuovo i primi due terzi.

---

```
SortinoSort(int[] A, int i, int j)
```

---

```
if  $j - i + 1 \leq 6$  then
```

```
    InsertionSort(A, i, j)
```

```
else
```

```
    int  $s = \lceil (j - i + 1) / 3 \rceil$ 
```

```
    SortinoSort(A, i, i + 2s - 1)
```

```
    SortinoSort(A, i + s, j)
```

```
    SortinoSort(A, i, i + 2s - 1)
```

---

# SortinoSort

- ① Qual è la complessità di questo algoritmo? Il Prof. Sortino finirà nella prossima edizione del mio libro?
- ② (Difficile, Opzionale) Dimostrare per induzione che questo algoritmo è corretto. Per comodità, assumete pure che tutti i valori siano distinti.

# Famiglia

Trovare i limiti superiori e inferiori più stretti possibili per la seguente famiglia di equazioni di ricorrenza, per valori di  $a$  interi positivi.

$$T(n) = \begin{cases} aT(\lfloor n/2 \rfloor) + n^{a-1} & n \geq 2 \\ 1 & n < 2 \end{cases}$$



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:

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

## Ricorrenza $2T(n/8) + 2T(n/4) + n$

$$T(n) = \begin{cases} 1 & n \leq 1 \\ 2T(n/8) + 2T(n/4) + n & n > 1 \end{cases}$$

È facile dimostrare che ovviamente, la funzione  $T(n)$  è  $\Omega(n)$ ; infatti,  
$$T(n) = 2T(n/8) + 2T(n/4) + n \geq n \geq c_1 n$$

per  $c_1 \leq 1$ . Non è necessario dimostrare il caso base, perché abbiamo rimosso gli elementi ricorsivi e quindi non abbiamo bisogno di induzione.

## Ricorrenza $2T(n/8) + 2T(n/4) + n$

$$T(n) = \begin{cases} 1 & n \leq 1 \\ 2T(n/8) + 2T(n/4) + n & n > 1 \end{cases}$$

Una prima osservazione che si potrebbe fare per “indovinare” un limite superiore è la seguente:

$$\begin{aligned} T(n) &= 2T(n/8) + 2T(n/4) + n \\ &\leq 2T(n/4) + 2T(n/4) + n \\ &\leq 4T(n/4) + n \end{aligned}$$

In base a questo risultato, il master theorem dice che  $T(n) = O(n \log n)$ .

## Ricorrenza $2T(n/8) + 2T(n/4) + n$

$$T(n) = \begin{cases} 1 & n \leq 1 \\ 2T(n/8) + 2T(n/4) + n & n > 1 \end{cases}$$

A questo punto, non ci resta che vedere quale dei due risultati ( $\Omega(n)$  e  $O(n \log n)$ ) è stretto. Proviamo con  $O(n)$ .

Ipotesi induttiva:  $\forall k < n : T(k) \leq ck$ . Proviamo che il risultato è valido anche per  $n$ .

$$\begin{aligned} T(n) &= 2T(n/8) + 2T(n/4) + n \\ &\leq 2cn/8 + 2cn/4 + n \\ &= 3/4cn + n \\ &\leq cn \end{aligned}$$

L'ultima disequazione è vera se  $3/4c + 1 \leq c$ , ovvero se  $c \geq 4$ .

# Ricorrenza

E' facile vedere che la ricorrenza è  $\Omega(n^2)$ , per via della sua componente non ricorsiva. Proviamo quindi a dimostrare che  $T(n) = O(n^2)$ .

- Caso base:  $T(n) = 1 \leq cn^2$ , per tutti i valori di  $n$  compresi fra 1 e 8. Tutte queste disequazioni sono soddisfatte da  $c \geq 1$ .
- Ipotesi induttiva:  $T(k) \leq ck^2$ , per  $k < n$
- Passo induttivo:

$$\begin{aligned} T(n) &= 2T(\lfloor n/2 \rfloor) + 4T(\lfloor n/4 \rfloor) + 15T(\lfloor n/8 \rfloor) + n^2 \\ &\leq 2c\lfloor n/2 \rfloor^2 + 4c\lfloor n/4 \rfloor^2 + 15\lfloor n/8 \rfloor^2 + n^2 \\ &\leq 2cn^2/4 + 4cn^2/16 + 15cn^2/64 + n^2 \\ &\leq 63/64cn^2 + n^2 \leq cn^2 \end{aligned}$$

L'ultima disequazione è rispettata per  $c \geq 64$ .

Abbiamo quindi dimostrato che  $T(n) = \Theta(n^2)$ .

# Analisi – Algoritmo di selezione deterministico

- Funzione:  $T(n) = T(\lfloor n/5 \rfloor) + T(\lfloor 7n/10 \rfloor) + 11/5 n$
- Ipotesi:  $T(n) = \Theta(n)$
- Limite superiore:  $\exists c > 0, \exists m > 0 : T(n) \leq cn, \forall n \geq m$

## Caso base

$$T(0) = 1 \leq c \cdot 0 \Leftrightarrow 1 \leq 0 \quad \text{Falso!}$$

$$T(1) = 1 \leq c \cdot 1 \Leftrightarrow c \geq 1$$

$$\begin{aligned} T(2) &= 22/5 + T(\lfloor 2/5 \rfloor) + T(\lfloor 14/10 \rfloor) \\ &= 22/5 + T(0) + T(1) = 32/5 \leq c \cdot 2 \Leftrightarrow c \geq 32/10 \end{aligned}$$

$$\begin{aligned} T(3) &= 33/5 + T(\lfloor 3/5 \rfloor) + T(\lfloor 21/10 \rfloor) \\ &= 33/5 + T(0) + T(2) = 70/5 \leq c \cdot 3 \Leftrightarrow c \geq 70/15 \end{aligned}$$

$$\begin{aligned} T(4) &= 44/5 + T(\lfloor 4/5 \rfloor) + T(\lfloor 28/10 \rfloor) \\ &= 44/5 + T(0) + T(2) = 81/5 \leq c \cdot 4 \Leftrightarrow c \geq 81/20 \end{aligned}$$

$$T(5) = 55/5 + T(\lfloor 5/5 \rfloor) + T(\lfloor 35/10 \rfloor) = 55/5 + T(1) + T(3)$$

# Analisi – Algoritmo di selezione deterministico

- Funzione:  $T(n) = T(\lfloor n/5 \rfloor) + T(\lfloor 7n/10 \rfloor) + 11/5 n$
- Ipotesi:  $T(n) = \Theta(n)$
- Limite superiore:  $\exists c > 0, \exists m \geq 0 : T(n) \leq cn, \forall n \geq m$

## Passo ricorsivo

$$\begin{aligned} T(n) &\leq c \lfloor n/5 \rfloor + c \lfloor 7n/10 \rfloor + 11/5 n \\ &\leq 1/5 cn + 7/10 cn + 11/5 n \\ &= 9/10 cn + 22/10 n \leq cn \end{aligned}$$

- L'ultima disequazione è vera per  $c \geq 22$ , quindi  $T(n) = O(n)$
- $T(n) = \Omega(n)$  per la componente non ricorsiva



## Analisi – MergeSortK

La funzione di ricorrenza per MergeSortK è la seguente:

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

Un modo per implementare MergeK consiste nel trovare il minimo dei  $k$  valori, presenti, operazione che ha complessità  $O(k)$ . Ripetendo l'operazione  $n$  volte, la complessità di MergeK è pari a  $O(kn)$ .

Calcolando  $\alpha = \log_k k = 1$  e confrontandolo con  $n^1$ , è possibile vedere che il costo di MergeSortK è  $O(kn \log n)$ . Per valori costanti di  $k$ , questo corrisponde a  $O(n \log n)$ .