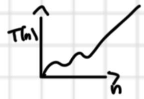
ANALISI DEL COSTO DI UN ALGORITMO

$T(n)$ Analisi asintotica
↳ il comportamento della funzione con valori molto grandi



$T(n)$ = tempo che dobbiamo aspettare per avere una risposta

la scomposizione in fattori primi molto più complesso della moltiplicazione

$50n$ crescita lineare

$(1,035)^n$ esponenziale

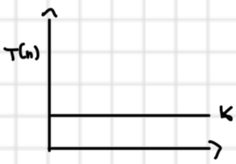
120 mesi

$O(n^2)$
algoritmi che per input piccoli → insertion il più veloce
quando input cresce → merge sort $O(n \log n)$

CONFRONTO TRA FUNZIONI
↳ risorse

Il tempo di esecuzione della macchina dipende da tanti fattori, noi parliamo del numero di passi elementari che l'algoritmo fa
somma dei passi = tempo computazionale

- $F(n) = K$ $0 < K < \infty$ → la cosa importante è che l'input non influisce sul tempo di esecuzione es. moltiplicazione



- $F(n) = n$ cresce linearmente rispetto al nostro input, cresce proporzionalmente
  $= 100n, \dots$

- $F(n) = \log_{(2)} n$ cresce meno rispetto alla lineare, $\forall\, n \to \infty$ cresce più lentamente

- $F(n) = n \log n$ FUNZIONE LINEARITMICA, quasi lineare, cresce più velocemente di n.
  es. merge sort

  - funzioni polinomiali

    - $F(n^2)$ quadratica
    - $F(n^3)$ cubica
      $n^{4,5,6\dots}$ = funzione polinomiale ($n^{\frac{1}{2}}$ cresce meno di n^2) il grado di crescita dipende dall'esponente

    - Funzione esponenziale:
      $F(n) = k^n$

    - Fattoriale più veloce dell'esponenziale
      $F(n) = n!$

da quella che cresce di meno a quella di più

es. $n = 1000$

$\log n$ → 10 (l'algoritmo avrà 10 passi) es. ricerca binaria

n → 1000

n^2 → 1000000 es. selection sort

2^n → 10^{300}

quale è la diff del numero dei passi di esecuzione?

NOTAZIONI ASINTOTICHE

- del di classi

classi che hanno elementi con stessa crescita

es.

$F(n) = n^3 + 7n^4 + 2n\log^3(n) + 3$ li confrontiamo $F(n) = 7n^4 \to$ classe

$\to$ le 7 posso eliminarlo

cresce in maniera simile a tutte le funzioni che appartengono alla classe n^4

- elimino fattori moltiplicativi ed elementi che danno contributo inutile

$7n^4 \quad \underbrace{n^3 \quad 2n\log^3(n) \quad 3}_{\text{contributo inutile perché piccoli quindi non le considero}}$ $\to$ dal più grande al più piccolo

Considero solo esponente delle n

- $F(n) = \cancel{\frac{7}{10}n} + \cancel{200\sqrt{n}} + \frac{1}{2}n^2 + 2000$ $\quad \sqrt{n} \to n^{\frac{1}{2}}$ $\qquad F(n) = n^2$ $\qquad$ due uguali n si sommano

- Notazione asintotica

si scrive:

$F(n) \in \Theta(n^{10}\log^7 n)$ $\qquad \Theta(g(n)) \to$ classe di tutte le funzioni che hanno comportamento asintotico simile a $g(n)$

limite stretto

es. $h(n) \in \Theta(g(n))$ $\qquad \lim_{x \to \infty} \frac{g(n)}{h(n)} \to k$

$F(n) \in \Theta(g(n)) \iff g(n) \in \Theta F(n)$ [lim sup e inf dell'altra, stessa classe perché si comportano allo stesso modo

$\Theta(g(n)) = \{F(n) : \exists\, c_1, c_2, n_0 \mid 0 \le c_1 g(n) \le F(n) \le c_2 g(n) \ \forall n \ge n_0\}$

$F(n)$ sta sempre in mezzo a c_1 e c_2; $g(n)$ accompagna sempre $F(n)$, si comportano allo stesso modo

- $O(n)$ (molto più ampia di Θ)

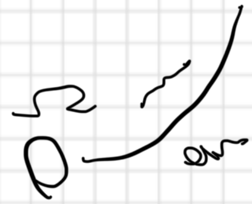
$F(n) \in O(g(n))$ (limitata superiormente)

voglio sapere se si comporta uguale o peggio a $g(n)$, allora $\in O(n)$

$O(g(n)) = \{F(n) : \exists\, c, n_0 \mid 0 \le F(n) \le c\, g(n),\ \forall n > n_0\}$

$\to$ Contiene sia funzioni che si comportano allo stesso modo, ma ci sono anche quelle che si comportano meglio

lim inf = $F(n) \in \Omega(g(n))$ $\qquad$ lim sup = $F(n) = O(g(n))$



$\Omega(g(n)) = \{F(n) : \exists\, c_1, n_0 \mid 0 \le c\,g(n) \le F(n) \ \forall n \ge n_0\}$ $\longrightarrow$ definizione di Ω

$F(n) \in \Theta(g(n))$ se e solo se $F(n) \in O(g(n))$ e $F(n) \in \Omega(g(n))$ (caso medio sta all'interno)

(O: posso andare sotto, garanzia maggiore; Ω: posso andare sopra, garanzia minore)

$\Theta(g(n)) \subset O(g(n)) \supset o(g(n))$

(Θ: come $g(n)$; O: come $g(n)$ o meglio, o più basse; o: $\to$ sottoinsieme di O)

$\Theta(g(n)) \subset \Omega(g(n))$ come o peggio

una classe che racchiude le funzioni che si comportano meglio di $g(n)$ ma mai come.

$\to o(g(n))\ \{F(n) : \exists\, c, n_0 \mid 0 \le F(n) < c\,g(n),\ \forall n \ge n_0\}$

$\omega(g(n))$ peggio ma mai come

$\omega(g(n))\ \{F(n) : \exists\, c, n_0 \mid 0 \le c\,g(n) < F(n),\ \forall n \ge n_0\}$

$\Theta,\ O,\ \Omega,\ o$ $\to$ Tutte le notazioni

(ascoltare registrazione x esempi ordinamento) a partire dalla procedura posso individuare la classe?

Mergesort

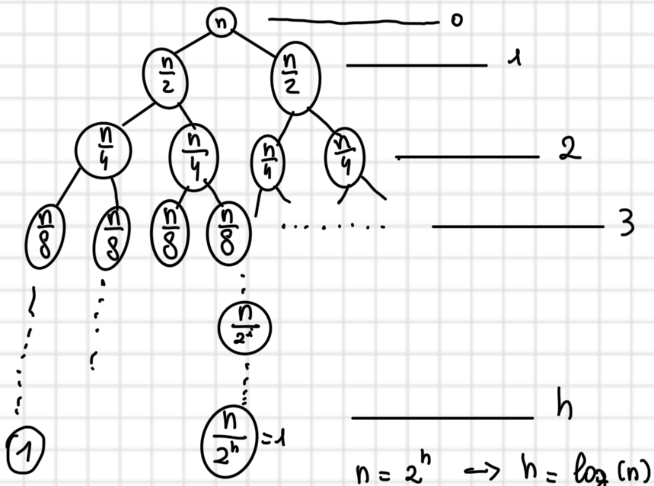
```
MS (A, n)
  IF n = 0 return
  m = n/2
  MS (A [0... m], n/2)
  MS (A [m.... n], n/2)
  MERGE (A, n)
```

equaz. di ricorrenza mi aiuta

$T(n) = \underbrace{2\,T(\tfrac{n}{2})}_{\text{numero chiamate ricorsive}} + \underbrace{n}_{\text{merge lineare}}$

$T(\tfrac{n}{2}) = 2T(\tfrac{n}{4}) + \tfrac{n}{2}$

1° tecnica / ALBERO DI RICORSIONE



LAVORO

n

n (se li sommo fa n)

n

n

$n = 2^h \rightarrow h = \log_2(n)$ i livelli dell'albero vanno da 0 a $\log_2(n)$

altezza = $\log_2(n)$

alla fine:

$T(n) = \sum_{i=0}^{\log(n)} n \in \Theta(n \log(n))$

↳ n perché è per ogni livello

↳ seguire questi passaggi per trovare la complessità

esempi

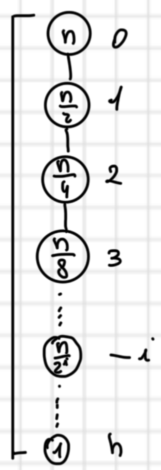
$T(n) = \underbrace{T(\tfrac{n}{2})}_{\text{1 chiamata ricorsiva dove dimezzo}} + 1 \rightarrow 1$ costante $\Rightarrow$ ricerca binaria

albero binario di ricerca

||

$T(n) = \log n$

log n



LAVORO

1

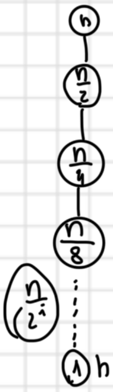
1

1

1

$T(n) = \sum_{i=0}^{\log(n)} 1 = \Theta(\log n)$

in array disordinato, trovare il K-esimo basato su quicksort



LAVORO

n

$\frac{n}{2}$

$\frac{n}{4}$

$\frac{n}{8}$

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

1

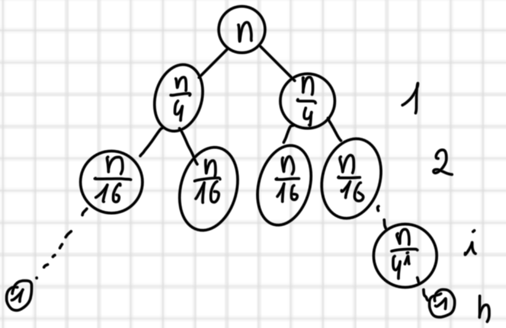
$$T(n) = \sum_{i=0}^{\log n} \frac{n}{2^i} = n \sum_{i=0}^{\log n} \left(\frac{1}{2}\right)^i \quad n\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = n\left(\frac{1}{1-\frac{1}{2}}\right) = 2n \in \Theta(n)$$

$$\left(\sum_{i=0}^{\infty} r^i = \frac{1}{1-r} \quad r<1\right)$$

~

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

2 sottoprob di dim $\frac{n}{4}$ e impiega tempo lineare



n

$\frac{n}{2}$

$\frac{n}{4}$

$\frac{n}{2^i}$ → regolarita'

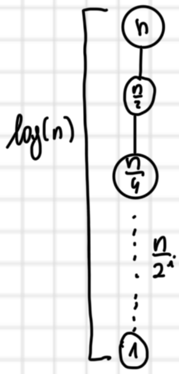
$$T(n) = \sum_{i=0}^{\log_4 n} \frac{n}{2^i} = n \sum_{i=0}^{\log_4 n} \left(\frac{1}{2}\right)^i \leq$$

$$n\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = n\left(\frac{1}{1-\frac{1}{2}}\right) = 2n \in \Theta(n)$$

altezza: $\frac{n}{4^h} = 1 \quad 4^h = n \quad h = \log_4 n$

~

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



LAVORO

$\log n$

$\log \frac{n}{2} = \log n - \log 2 = \log n - 1$

$\log \frac{n}{4} = \log n - \log \overset{(2^2)}{4} = \log n - 2$

$\log\left(\frac{n}{2^i}\right) = \log n - i$ → ad ogni passo

$$T(n) = \sum_{i=0}^{\log n} (\log n - i) = \sum_{i=0}^{\log n} \log n - \sum_{i=0}^{\log n} i$$

$$\left(\sum_{i=0}^{n} i = \frac{n(n+1)}{2}\right) \text{ formula}$$

$$\log^2(n) - \frac{\log n(\log n + 1)}{2}$$

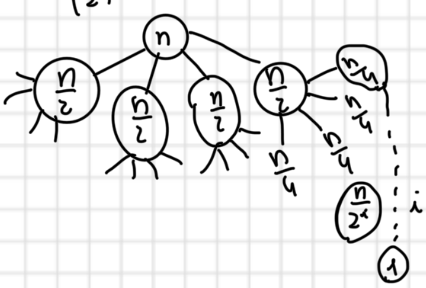
$$\log^2(n) - \cancel{\frac{\log^2(n)}{2}}$$ più piccolo si toglie

$\log^2(n) \in O(\log^2(n))$

↳ quadratica

~

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



n

$2n$

$4n$

$2^i n$

$$T(n) = \sum_{i=0}^{\log n} 2^i n = n \sum_{i=0}^{\log n} 2i \quad > 1$$

$$\left(\sum_{i=0}^{n} r^i \overset{r>1}{=} \frac{r^{n+1}-1}{r-1}\right) \text{ formula}$$

$$= n\left(\frac{2^{\log n + 1} - 1}{2-1}\right) = n(2^{\log n + 1} - n) = 2n\,2^{\log n} - n = 2n \cdot n^{\log 2}$$

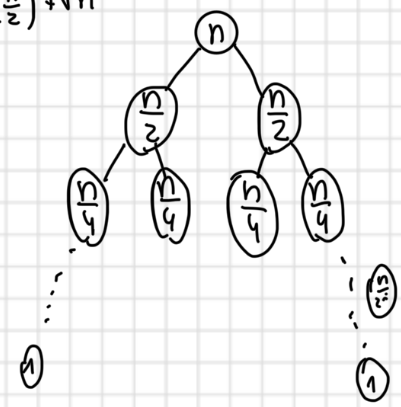
$$= 2n^2 - n$$

→ moltiplicato ×2 messo davanti

→ posso scambiare $2^{\log n}$

se vado a dimezzare altezza dell'albero è $\log n$

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



i

LAVORO

$\sqrt{n}$

$2\sqrt{\frac{n}{2}} = \sqrt{2n} = \sqrt{2}\sqrt{n}$

$4\sqrt{\frac{n}{4}} = \sqrt{4n} = \sqrt{4}\sqrt{n}$

$\sqrt{2i}\,\sqrt{n}$

$$T(n) = \sum_{i=0}^{\log n} \sqrt{2}^i \sqrt{n} = \sqrt{n} \sum_{i=0}^{\log n} (\sqrt{2})^i = \sqrt{n} \cdot \left(\frac{\sqrt{2}^{\log n + 1} - 1}{\sqrt{2} - 1} \right) = \sqrt{n} \left(\frac{\sqrt{2}\sqrt{n} - 1}{\sqrt{2} - 1} \right) = \sqrt{n}\sqrt{n} = n$$

$\downarrow_i$ livelli

(nel caso $\sum_{i=0}^{n} r^i$)

$$\sqrt{2}^{\log n} = 2^{\frac{1}{2}\log n} = 2^{(\log n)\frac{1}{2}} = n^{(\log 2)\frac{1}{2}} = n^{\frac{1}{2}} = \sqrt{n}$$