06/10/2023 - Algoritmi LAB

- Strumenti per l'analisi

Analizzare un algoritmo vuol dire predire e quantificare le risorse utilizzate dallo stesso.

- Spazio
- Memoria
- Energia
- Larghezza di banda
- Tempo Computazionale (RUNTIME)

- Adotteremo il modello RAM (Random-Access Machine)

Caratteristiche:

- Operazioni eseguite in modo sequenziale senza operazioni contemporanee
- Ogni istruzione e accesso alla memoria avviene in tempo costante.

Istruzioni:

- Aritmetiche ($+, -, *, /, \%, \lfloor\rfloor, \lceil\rceil$)
- Spostamento dei dati
- Controllo (Branching condizionali e non, subroutines, return)

Data Types:

- Interi
- Floating Point
- Caratteri

- Ogni stringa di bit ha un numero limitato di bit.

Per rappresentare n servono $\log n$ bits.

- Non si tiene conto delle gerarchie di memoria.

Vogliamo espressioni che descrivano l'andamento dell'algoritmo che possono essere formule che dipendono dall'input.

Es. Quanto impiega l'insertion sort?

Esaminiamo il numero di volte che l'algoritmo esegue ogni riga di pseudocodice e quanto tempo richiede ciascuna riga per essere eseguita.

Il **TEMPO COMPUTAZIONALE** di un algoritmo su un input di fissata misura n è il numero di istruzioni e accessi ai dati eseguiti. Nel modello RAM assumiamo che sia necessaria una quantità costante di tempo per eseguire ogni riga.

La misura dell'input dipende dal problema da risolvere.

```
1. for i = 2 to n
2.     key = A[i]
3.     j = i - 1
4.     while j > 0 AND A[j] > key
5.         A[j+1] = A[j]
6.         j = j - 1
7.     A[j+1] = key
```

COSTO	RIPETIZIONI
c_1	n*
c_2	$n-1$
c_3	$n-1$
c_4	$\sum_{i=2}^{n} t_i$
c_5	$\sum_{i=2}^{n} (t_i - 1)$
c_6	$\sum_{i=2}^{n} (t_i - 1)$
c_7	$n-1$

*] cicli for vengono ripetuti una volta in più rispetto alle istruzioni all'interno

t_i: numero di volte che il ciclo while a riga 6 viene eseguito, per i.

$$T(n) = c_1 \cdot n + c_2(n-1) + c_3(n-1) + c_4 \sum_{i=2}^{n} t_i + c_5 \sum_{i=2}^{n}(t_i - 1) + c_6 \cdot \sum_{i=2}^{n}(t_i - 1) + c_7(n-1)$$

Best Case: $\forall i,\ t_i = 1$

$$T(n) = c_1 \cdot n + c_2(n-1) + c_3(n-1) + c_4 \sum_{i=2}^{n} t_i + c_5 \sum_{i=2}^{n}(t_i - 1) + c_6 \cdot \sum_{i=2}^{n}(t_i - 1) + c_7(n-1) =$$

($c_2(n-1) = c_2 n - c_2$; $\sum_{i=2}^{n} t_i$ con $t_i = 1$ vale n; $t_i - 1 = 0$)

$$= \underbrace{(c_1 + c_2 + c_3 + c_4 + c_5 + c_7)}_{a} \cdot n + \underbrace{[-(c_2 + c_3 + c_4 + c_7)]}_{b}$$

$$T(n) = an + b \quad \rightarrow \quad T(n) = O(n)$$

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \qquad \sum_{i=2}^{n} i = \sum_{i=1}^{n} i - 1$$

Worst Case: $\forall i \in \{2, 3, \dots, n\},\ t_i = i$

$$T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4\left(\frac{n(n+1)}{2} - 1\right) + c_5\left(\frac{n(n-1)}{2}\right) + c_6\left(\frac{n(n-1)}{2}\right) + c_7(n-1) =$$

($\frac{c_4}{2}n^2 + \frac{c_4}{2}n - c_4$)

$$= \underbrace{\left(\frac{c_5}{2} + \frac{c_5}{2} + \frac{c_6}{2}\right)}_{a} n^2 + \underbrace{\left(c_1 + c_2 + c_3 + \frac{c_4}{2} - \frac{c_5}{2} - \frac{c_6}{2} + c_7\right)}_{b} n + \underbrace{[-(c_2 + c_3 + c_4 + c_7)]}_{c}$$

$$T(n) = an^2 + bn + c \quad \rightarrow \quad T(n) = O(n^2)$$

Un algoritmo è più efficiente di un altro se il suo tempo nel caso peggiore è di un ordine di grandezza

inferiore, ovvero cresce più lentamente al crescere della misura dell'input.

Notazioni Asintotiche

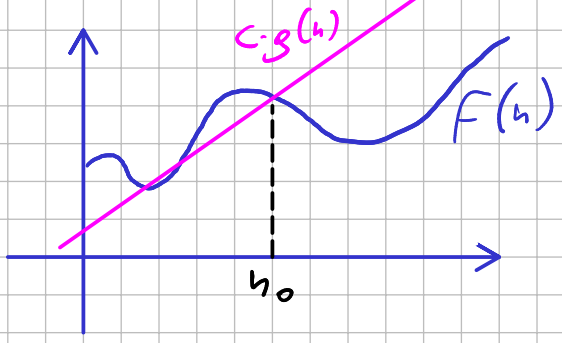
Convenzione: $T(n) = f(N)$

(big-Oh) O-NOTATION

Abuso di notazione (dovrebbe essere $\in$)

es. $f(n) = 7n^3 + 10n^2 - 20n + 6 \rightsquigarrow f(n) = O(n^3), O(n^4), O(n^5)$

upper bound

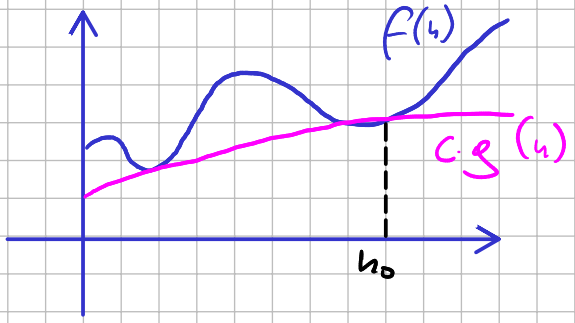


$$O(g(n)) = \left\{ f(n) \;\middle|\; \begin{array}{l} \exists\ c>0 \text{ costante e } n_0 \in \mathbb{N} \\ \text{Tale che } 0 \le f(n) \le c \cdot g(n) \quad \forall n \ge n_0 \end{array} \right\}$$

(big-omega) Ω-NOTATION

es. $f(n) = 7n^3 + 10n^2 - 20n + 6 \rightsquigarrow f(n) = \Omega(n^3), \Omega(n^2), \Omega(n)$

lower bound



$$\Omega(g(n)) = \left\{ f(n) \;\middle|\; \begin{array}{l} \exists\ c>0 \text{ costante e } n_0 \in \mathbb{N} \\ \text{Tale che } 0 \le c \cdot g(n) \le f(n) \quad \forall n \ge n_0 \end{array} \right\}$$

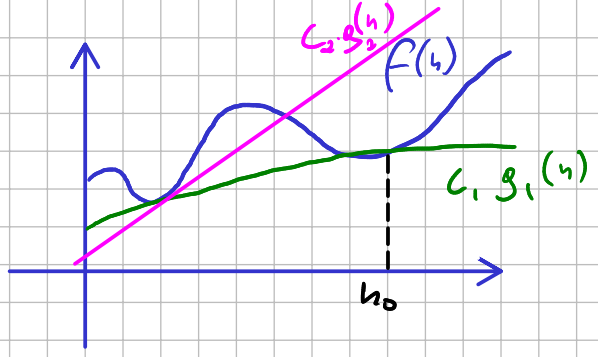
(big-Theta) Θ-NOTATION

es. $7n^3 + 10n^2 - 20n + 6 = \Theta(n^3)$

$\neq \Theta(n^2)$

$\neq \Theta(n^4)$

tight bound



$$\Theta(g(n)) = \left\{ f(n) \;\middle|\; \begin{array}{l} \exists\, c_1, c_2 > 0 \text{ costanti e } n_0 \in \mathbb{N} \\ \text{tale che } 0 \le c_1 \cdot g_2(n) \le f(n) \le c_2 \cdot g_2(n) \quad \forall n \ge n_0 \end{array} \right\}$$

TEO.

$\forall f(n), g(n)$

$$f(n) = \Theta(g(n)) \iff \begin{cases} f(n) = O(g(n)) \\ f(n) = \Omega(g(n)) \end{cases}$$

(little-oh) o-NOTATION

Per notazioni non particolarmente precise

es. $2n^2 = O(n^2)$ → asintoticamente precisa (asymptotically tight)

$2n = O(n^2)$ → non asintoticamente precisa

$2n = o(n^2)$, mentre $2n^2 \neq o(n^2)$

$$o(g(n)) = \left\{ f(n) \;\middle|\; \begin{array}{l} \forall\, c > 0 \text{ costante}, \exists\, n_0 \in \mathbb{N} \\ \text{tale che } 0 \le f(n) < c \cdot g(n) \end{array} \right\}$$

* $f(n)$ è asintoticamente più piccola di $g(n)$

(little-omega) ω-NOTATION

$$\omega(n) = \left\{ f(n) \;\middle|\; \begin{array}{l} \forall c > 0 \text{ costante}, \exists n_0 \in \mathbb{N} \\ \text{tale che } 0 \le c \cdot g(n) < f(n) \end{array} \right\}$$

oss. 1: $f(n) = \omega(g(n)) \iff \lim_{n \to +\infty} \frac{f(n)}{g(n)} = +\infty$ (se esiste)

oss. 2: $f(n) = \omega(g(n)) \iff g(n) = o(f(n))$

Proprietà:

- Transitiva: $f(n) = \Theta(g(n))$ e $g(n) = \Theta(h(n))$ $\Downarrow$ $f(n) = \Theta(h(n))$

  (vale per $\Theta, O, \Omega, o, \omega$)

- Riflessiva: $f(n) = \Theta(f(n))$

  (vale per Θ, O, Ω)

- Simmetrica: $f(n) = O(g(n)) \iff g(n) = \Omega(f(n))$

- Simmetrica Trasposta: $f(n) = O(g(n)) \iff g(n) = \Omega(f(n))$

  $f(n) = o(g(n)) \iff g(n) = \omega(f(n))$

Non vale l'ordinamento totale:

$\exists\ f(n) \neq O(g(n)),\ f(n) \neq \Omega(g(n))$

(non possiamo confrontare due potenze di n)