Appunti di Algoritmi e Strutture Dati

Algoritmi e Strutture Dati (prof. Pirola) - CdL Informatica Unimib - 23/24

Federico Zotti

14 Mar 2024

Indice

1 Introduzione **2**

1.1 Algoritmo: Definizione dell'ordinamento di un vettore 2

1.2 Scelta di un algoritmo 2

1.2.1 Tempo di esecuzione 2

1.3 Algoritmo: Ricerca sequenziale 3

2 Problema computazionale **5**

2.1 Esempio: Ricerca in un vettore 6

2.2 Esempio: Ricerca in un vettore ordinato 6

3 Trovare il miglior algoritmo **6**

3.1 Esempio 7

4 Notazioni asintotiche **7**

4.1 Limite asintotico superiore 7

4.2 Limite asintotico inferiore 9

4.3 Limite asintotico stretto 10

4.4 Scala degli asintoti 13

4.5 Esempi 14

5 Stimare il tempo di un algoritmo **14**

5.1 Problema: Trova minimo 14

1 Introduzione

Un'**algoritmo** è una sequenza di istruzioni **elementari** (devono essere comprese e eseguite dall'esecutore) che permettono di risolvere un problema computazionale (ovvero per ogni possibile input produce l'output corretto).

Per definire un **problema** è necessario specificare:

- Il tipo del parametro in input
- Il tipo del risultato in output
- Il legame tra input e output

Un'**istanza** di un problema si ottiene specificando uno dei possibili valori in input specifico per il problema.

1.1 Algoritmo: Definizione dell'ordinamento di un vettore

Sort:

- Input: `Array Int` (`Dim` n) $\rightarrow A = \langle a_1, a_2, \dots, a_n \rangle$
- Output: `Array Int` (`Dim` n) $\rightarrow A' = \langle a'_1, a'_2, \dots, a'_n \rangle$

A' è una permutazione di A, tale che $a'_1 \leq a'_{i+1} \quad \forall i \, . \, 1 \leq i \leq n-1$.

1.2 Scelta di un algoritmo

L'algoritmo migliore è quello che utilizza il minor numero di risorse.

Le risorse sono:

- Il tempo di esecuzione
- Lo spazio (memoria) utilizzato

1.2.1 Tempo di esecuzione

Per calcolare il tempo utilizziamo una funzione $T(n)$. n rappresenta la quantità di dati in input.

- $T_p(n)$ rappresenta il caso peggiore
- $T_n(n)$ rappresenta il caso "medio" (non è la media dei due)
- $T_m(n)$ rappresenta il caso migliore

1.2.1.1 Esempio

- Algoritmo 1: $T(n) = 100000 \cdot n$
- Algoritmo 2: $T(n) = 10 \cdot n^3$
- Algoritmo 3: $T(n) = 1 \cdot 2^n$

In questo caso il migliore dipende dal grado di n, dunque l'algoritmo 1 risulta quello più veloce. Per numeri di n molto piccoli invece è meglio calcolare caso per caso il tempo. Nel caso ci siano più n, si considera quello con il grado maggiore.

$$T(n) = 7n^3 + 2n + 10000 \sim n^3$$

1.3 Algoritmo: Ricerca sequenziale

- V: vettore di interi
- k: intero da cercare nel vettore
- p: posizione nel vettore

Ricerca sequenziale

```
function SEARCH(V[ ], k)
    p ← 1
    while (V[p] ≠ k) ∧ (p ≤ LENGTH(V) ) do
        p ← p + 1
    end while
    if p > LENGTH(V) then
        return −1
    else
        return p
    end if
end function
```

Analisi del tempo di esecuzione:

- Caso peggiore: $k \neq V[\,] \Rightarrow T(n) = 3 + 2 \cdot n + 1 \sim n$
- Caso migliore: $k = V[1] \Rightarrow T(n) = 4 \sim c$
- Caso medio: $k = V[\frac{n}{2}] \Rightarrow 3 + 2\frac{n}{2}(\pm 1) \sim n$

Se V è ordinato ci si può fermare appena trova un numero più grande di k.

Ricerca sequenziale

```
function SEARCH(V[ ], k)
    p ← 1
    while (V[p] < k) ∧ (p ≤ LENGTH(V) ) do
        p ← p + 1
    end while
    if V[p] ≠ k ∨ p > LENGTH(V) then
        return −1
    else
        return p
    end if
end function
```

Analisi del tempo di esecuzione:

- Caso migliore: $V[p] \geq k \Rightarrow 4 \sim c$
- Caso peggiore: $k > V[p] \Rightarrow 3 + 2n \sim n$
- Caso medio: $k = V\left[\frac{n}{2}\right] \Rightarrow 3 + 2\frac{n}{2} \cdot \frac{1}{2}$

Per avere un'ottimizzazione significativa si può sfruttare il fatto che il vettore è ordinato per implementare una semplice ricerca binaria (spezzare il vettore e guardare solo una metà).

Ricerca binaria

```
function SEARCH(V[ ], k)
    sx ← 1
    dx ← LENGTH(V)
    p ← dx + sx div 2
    while (V[p] ≠ k) ∧ (sx < dx) do
        if k > V[p] then
            sx ← p + 1
        else
            dx ← p − 1
        end if
        p ← (dx + sx) div 2
    end while
    if V[p] = k then
        return p
    else
        return −1
    end if
end function
```

Analisi del tempo di esecuzione:

- Caso migliore: $V\left[\frac{n}{2}\right] = k \Rightarrow t_m(n) = 6 \sim c$
- Caso peggiore: $k \notin V \Rightarrow T_p(n) = 5 + 4 \cdot (\log_2 n) + 1 \sim \log_2 n$
- Caso medio: $T(n) = \frac{T_p(n)}{2} \sim \log_2 n$

2 Problema computazionale

Un problema computazionale è una relazione tra input e output.

$$P \subseteq I \times O$$

2.1 Esempio: Ricerca in un vettore

Input:

- Un vettore V di n elementi
- Un valore k

Output:

- Un intero i, t.c. $i = -1$ se $k \notin V$ altrimenti $V[i] = k$

Ci sono tanti algoritmi per risolvere questo problema. Un esempio è la **ricerca sequenziale**.

> Aggiungere algo. #todo-uni

2.2 Esempio: Ricerca in un vettore ordinato

Input:

- Un vettore **ordinato** V di n elementi
- Un valore k

Output:

- Un intero i, t.c. $i = -1$ se $k \notin V$ altrimenti $V[i] = k$

Questo problema è diverso dal precedente, perché i dati in input sono diversi. Essendo che l'algoritmo della ricerca sequenziale è corretto per tutti i vettori in input, è corretto anche per i vettori ordinati. Esistono però algoritmi specifici per questo problema. Per esempio la **ricerca binaria** (o dicotomica).

> Inserire algo. #todo-uni

3 Trovare il miglior algoritmo

Generalmente il **tempo di esecuzione** e lo **spazio utilizzato** da un algoritmo sono legati alla grandezza dell'input. Dunque è possibile esprimere questi due dati come funzioni di n.

$$T(n)$$

$$S(n)$$

> In questo corso ci si concentrerà di più su $T(n)$.

Il tempo di esecuzione non può essere espresso in secondi, perché questi dipendono dalla velocità dell'elaboratore (da quanto tempo viene impiegato ad eseguire ogni singola istruzione). Dunque il tempo viene espresso in quante istruzioni devono essere eseguite.

Non sempre però il tempo dipende soltanto da n. Per esempio $300 + 200$ risulta molto più semplice di $345 + 783$, nonostante abbiano lo stesso numero di cifre. Dunque $T(n)$ oscilla tra il caso migliore $T_{migl}(n)$ e il caso peggiore $T_{pegg}(n)$.

3.1 Esempio

Riprendendo gli algoritmi di ricerca si possono definire i tempi di esecuzione.

- **Ricerca sequenziale:**
  - $T_{migl}(n) = a_1$
  - $T_{pegg}(n) = a_2 + b_2 n$
- **Ricerca binaria:**
  - $T_{migl}(n) = a_3$
  - $T_{pegg}(n) = a_4 + b_4 \log_2 n$

4 Notazioni asintotiche

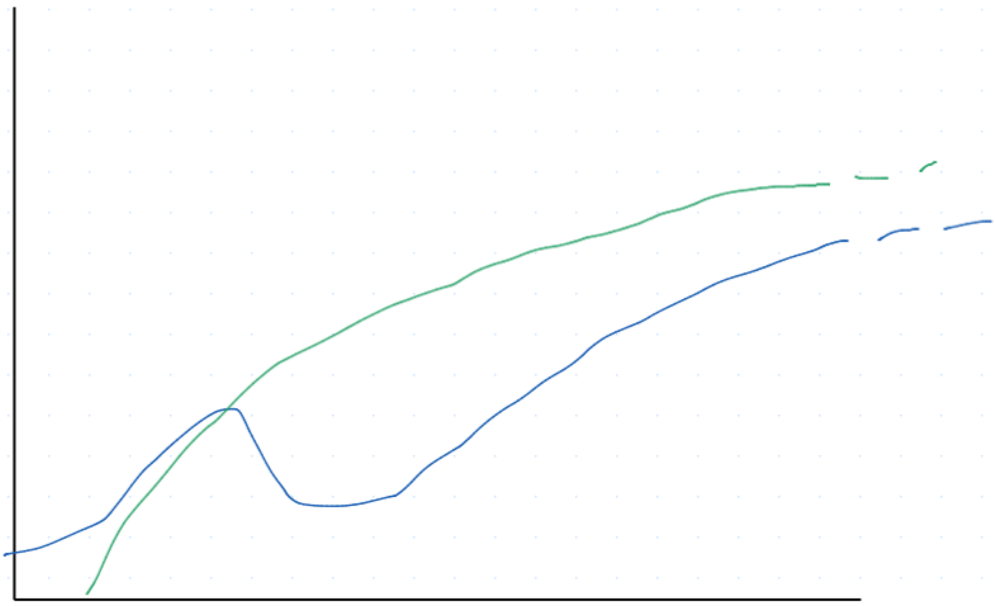
4.1 Limite asintotico superiore

Assunzioni:

- Le funzioni $T(n)$, $S(n)$ sono definite nei numeri naturali, ma i grafici vengono definiti nei reali per semplicità

- Le funzioni sono definitivamente positive

> Aggiungere grafico con una funzione casuale e una funzione def maggiore. #todo-uni



- Blu: $T_{pegg}(n) = f(n)$
- Verde: $O(g(n))$

$$O(g(n)) = \left\{ f(n) \mid \exists\, n_0 > 0, c > 0 \,:\, 0 \leq f(n) \leq c \cdot g(n) \forall\, n > n_0 \right\}$$

Ovvero $O(g(n))$ è l'insieme delle funzione superiormente limitate da $g(n)$ per una costante c $(g(n) \cdot c)$.

> **Esempio:**
>
> - $f(n) = 3n^2$
> - $3n^2 \in O(n^3)$? ovvero
> - $\exists\, n_0, c > 0 \,:\, 0 \leq 3n^2 \leq c \cdot n^3 \forall n > n_0$?
>
> $n \geq \frac{3}{c}$? Questo è verificato per esempio con $c = 3, n_0 = 1$.
> È possibile però che esista una funzione $g(n)$ "minore": $3n^2 = O(n^2)$?
> $c \geq 3$? Questo è verificato per esempio con $c = 4, n_0 = 1$.
> Dunque $3n^2 = O(n^2)$.

Si può provare con una funzione ancora più "bassa", ma facendo i calcoli la condizione non viene soddisfatta.

Esempio:

- $f(n) = 5n^3 + n^2$
- $f(n) \in O(n^3)$? ovvero
- $\exists\, n_0, c > 0 : 0 \leq 5n^3 + n^2 \leq c \cdot n^3 \forall n > n_0$?

$6n^3 \leq cn^3$? Questo è verificato per esempio con $c = 7, n_0 = 1$.

Si può provare con una funzione ancora più "bassa", ma facendo i calcoli la condizione non viene soddisfatta.

4.2 Limite asintotico inferiore

> Aggiungere grafico con funzione $T(n)$ e una funzione def inferiore.

$$\Omega(g(n)) = \{\, f(n) \mid \exists\, n_0 > 0, c > 0 \text{ t.c. } 0 \leq c \cdot g(n) \leq f(n) \,\forall\, n > n_0 \,\}$$

Es:

- $f(n) = 3n^2$
- $g(n) = n$
- $3n^2 \in \Omega(n)$?

$$\exists\, c > 0, n_0 > 0 \quad 0 \leq cn \leq 3n^2 \quad \forall\, n > n_0$$
$$3n^2 \geq cn$$
$$n \geq \frac{c}{3}$$
$$c = 3 \quad n_0 = 1$$

Dunque è verificato.

Si può dire che $3n^2 = \Omega(n)$ *(impropriamente)*.

Proseguendo, $3n^2 = \Omega(n^2)$?

$$\exists\, c > 0, n_0 > 0 \quad 0 \leq cn^2 \leq 3n^2 \quad \forall\, n > n_0$$

$$3n^2 \geq cn^2$$

$$3 \geq c$$

$$c = 3 \quad n_0 = 1$$

Dunque è verificato.

Continuando, $3n^2 = \Omega(n^3)$?

$$\exists\, c > 0, n_0 > 0 \quad 0 \leq cn^3 \leq 3n^2 \quad \forall\, n > n_0$$

$$3n^2 \geq cn^3$$

$$3 \geq cn$$

$$n \leq \frac{3}{c}$$

Questo non è verificato definitivamente. $3n^2 \notin \Omega(n^3)$. Lo stesso vale per tutti gli $\Omega(n^\varepsilon) \quad \forall\, \varepsilon > 2$.

4.3 Limite asintotico stretto

Devo trovare due costanti c_1, c_2 tale che la funzione $T(n)$ è compresa definitivamente tra $c_1 g(n)$ e $c_2 g(n)$.

> Disegnare grafico #todo-uni : funzione casuale $T(n)$ con due "rette"/"curve" che stanno def. sopra e def. sotto $T(n)$.

$$\Theta(g(n)) = \{\, f(n) \mid \exists\, n_0, c_1, c_2 > 0 \,:\, 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \forall\, n > n_0 \,\}$$

$$\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$$

Es 1: continuando gli esempi precedenti, sappiamo che

$$3n^2 = \Omega(n^2) \land 3n^2 = O(n^2) \implies 3n^2 = \Theta(n^2)$$

Es 2:

- $f(n) = 5n^3 + n^2$
- $g(n) = n^3$
- $5n^3 + n^2 = \Omega(n^3)$?

$$5n^3 + n^2 \geq 5n^3 = \Omega(n^3)$$
$$5n^3 + n^2 = O(n^3) \implies 5n^3 + n^2 = \Theta(n^3)$$

Es 3:

- $f(n) = 5n^3 - 10n^2 - 30n$
- $g(n) = n^3$
- $f(n) = \Omega(n^3)$?

$$\exists\, c, n_0 > 0 \quad 0 \leq cn^3 \leq 5n^3 - 10n^2 - 30n \quad \forall\, n > n_0$$
$$5n^3 - 10n^2 - 30n \geq cn^3$$
$$5n^3 - cn^3 \geq 10n^2 + 30n$$
$$(5 - c)n^3 \geq 10n^2 + 30n$$

Sapendo che

- $10n^2 \leq 10n^2$
- $30n \leq 30n^2$

possiamo scrivere $10n^2 + 30n \leq 10n^2 + 30n^2$. Adesso dobbiamo stabilire se $(5 - c)n^2 \geq 10n^2 + 30n^2$ (perché implica la disuguaglianza precedente).

$$(5 - c)n^2 \geq 10n^2 + 30n^2$$
$$n \geq \frac{40}{5 - c} \qquad \text{con } c < 5$$
$$c = 1 \quad n_0 = 9$$

Dunque è verificato.

Inoltre $f(n) = O(n^3) \implies f(n) = \Theta(n^3)$.

Dim: $\Omega()$ è uguale a n al grado massimo del polinomio?

- $f(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_0 \quad a_k > 0$
- $g(n) = n^k$
- $f(n) = \Omega(n^k)$?

$$\exists\, c, n_0 > 0 \quad 0 \leq cn^k \leq f(n) \quad \forall\, n > n_0$$
$$a_k n^k + a_{k-1} n^{k-1} + \ldots + a_0 \geq cn^k$$
$$(a_k - c)n^k \geq -a_{k-1} n^{k-1} - \ldots - a_0$$

$$-a_{k-1} n^{k-1} - \ldots - a_0 \leq |a_{k-1}| n^{k-1} + |a_{k-2}| n^{k-1} + \ldots + |a_0| n^{k-1}$$

$$(a_k - c)n^k \geq \sum_{i=0}^{k-1} |a_i| n^{k-1}$$
$$(a_k - c)n^k \geq n^{k-1} \sum_{i=0}^{k-1} |a_i|$$
$$(a_k - c)n \geq \sum_{i=0}^{k-1} |a_i|$$
$$n \geq \frac{\sum_{i=0}^{k-1} |a_i|}{a_k - c} \qquad \text{con } c < a_k$$

Dimostrato che un polinomio è un omega di n al grado massimo.

Dim: un'esponenziale è limitato inferiormente da un polinomio.

- $f(n) = 2^n$
- $2^n = O(2^n)$
- $2^n = \Omega(n^b) \quad \forall\, b > 0$?

$$\lim_{n \to +\infty} \frac{n^b}{2^n} = 0$$
$$\implies \exists\, n_0 \quad \forall\, n > n_0 \quad \frac{n^b}{2^n} < 1$$
$$2^n > 1 n^b$$

Quindi tutti i polinomi limitano inferiormente un esponenziale. E sapendo che $2^n =$

$\Omega(2^n)$, si considera questo perché è "il più grande". Ciò implica $2^n = \Theta(2^n)$.

Dim: lo stesso per i logaritmi.

- $f(n) = \log_2^a n$
- $\log_2^a n = O(n^b) \quad \forall\, b > 0\,?$

$$\lim_{n \to +\infty} \frac{\log_2^a n}{n^b} = 0$$
$$\implies \exists\, n_0 \quad \forall\, n > n_0 \quad \frac{\log_2^a n}{n^b} < 1$$
$$\log_2^a = O(n^b)$$

Quindi tutti i polinomi limitano inferiormente un logaritmo. E sapendo che $\log_2^a = O(\log_2^a)$, si considera questo perché è "il più piccolo". Ciò implica $\log_2^a = \Theta(\log_2^a)$.

4.4 Scala degli asintoti

- 1
- $\log n$
- $n^b \quad \forall\, 0 < b < 1$
- n
- $n \cdot \log n$
- $n^{1+\varepsilon} \quad \forall\, 0 < \varepsilon < 1$
- n^2
- $n^2 \cdot \log n$
- $n^{2+\varepsilon} \quad \forall\, 0 < \varepsilon < 1$
- n^3
- $n^3 \cdot \log n$
- $n^{2+\varepsilon} \quad \forall\, 0 < \varepsilon < 1$
- $\dots$
- $a^n \quad \forall\, a > 1$ *(ogni a è una classe a se)*
- $n!$
- n^n

4.5 Esempi

Es 1: ricerca in un vettore ordinato.

Algoritmo	Tempo caso migliore	Tempo caso peggiore
Ricerca sequenziale	$a_1 = \Omega(1)$	$a_2 + b_2 n = O(n)$
Ricerca dicotomica	$a_3 = \Omega(1)$	$a_4 + b_4 \log n = O(\log n)$

5 Stimare il tempo di un algoritmo

5.1 Problema: Trova minimo

- **Input:** un vettore V di n interi
- **Output:** una posizione i t.c. $V[i] \leq V[j] \quad \forall\, 1 \leq j \leq n$

```
function TrovaMinimo(V[])
    posmin ← 1
    for i = 2 to length(V) do
        if V[i] < V[posmin] then
            posmin ← i
        end if
    end for
    return posmin
end function
```

- **Caso migliore:** $1 + n + (n-1) + 0 + 1 = 2n + 1 = \Omega(n)$
- **Caso peggiore:** $1 + n + (n-1) + (n-1) + 1 = 3n = O(n)$

Dunque il tempo di calcolo di questo algoritmo è $\Theta(n)$.

