Elementi di analisi degli algoritmi

Lezione 3 di Programmazione 2

Docente: Giuseppe Scollo

Università di Catania, sede di Comiso (RG)
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Studi in Informatica applicata, AA 2006-7

Indice

stima empirica delle prestazioni

valutazione empirica delle prestazioni: stima o misura?

si **misura**no le prestazioni di **di un'implementazione** dell'algoritmo

si procede dalla misura alla **stima** delle prestazioni **dell'algoritmo** astraendo (per quanto possibile) dall'influenza di caratteristiche specifiche dell'implementazione

a tale scopo è spesso necessaria un'analisi del codice, per:

- capire quali parti siano più critiche per le prestazioni (ad es., cicli più interni)
- capire se tali criticità siano dovute all'algoritmo o all'implementazione

problema inerente della stima empirica: dipendenza dall'implementazione

- per la correttezza metodologica del **confronto empirico** delle prestazioni di diversi algoritmi, si richiede che le implementazioni abbiano la stessa accuratezza

1° errore frequente nella scelta di un algoritmo:

- **sottovalutare** l'importanza delle prestazioni

2° errore frequente nella scelta di un algoritmo:

- **sopravvalutare** l'importanza delle prestazioni

analisi degli algoritmi: motivazioni e obiettivi

a che serve l'analisi delle prestazioni degli algoritmi:

- confrontare algoritmi diversi per uno stesso problema
- stimare le prestazioni di algoritmi indipendentemente dal linguaggio di programmazione e dall'ambiente operativo
- ottimizzare eventuali parametri degli algoritmi

cosa serve per l'analisi delle prestazioni degli algoritmi:

- alcuni (pochi) concetti matematici fondamentali (v. appresso)
- analisi disponibili in letteratura per algoritmi di ampia applicabilità

problemi inerenti dell'analisi degli algoritmi:

- stima approssimata (quella esatta può essere difficile)
- algoritmi diversi possono avere prestazioni incomparabili (ad es. con ampie e diverse fluttuazioni al variare dell'input)

quali analisi sono rilevanti:

- **caso medio:** si assume che l'input sia generato casualmente
- **caso peggiore:** analizzare algoritmo e **dati**, per determinare l'input appropriato

velocità di crescita delle funzioni

assunto: dipendenza delle prestazioni dell'algoritmo da un **parametro primario N**
(uno solo, per semplicità e senza perdita di generalità)

il tempo di esecuzione ("costo" più significativo degli algoritmi nella maggior parte dei casi) tipicamente cresce come una delle seguenti funzioni di **N**, e la rispettiva crescita si dice:

1 : costante (crescita zero)
log N : logaritmica
N : lineare
N log N : poco più che lineare ("linearitmica"?)
N^2 : quadratica
N^3 : cubica
2^N : esponenziale

valori di funzioni comuni

l'andamento, per un campione di valori, di funzioni comunemente incontrate mostra che il loro confronto per **piccoli** valori di N può essere anche molto diverso da quello per **grandi** valori di N, determinato dalla velocità di crescita **asintotica** delle funzioni considerate:

lg N	$N^{1/2}$	N	N lg N	$N(\lg N)^2$	$N^{3/2}$	N^2
3	3	10	33	110	32	10^2
7	10	10^2	664	4414	1000	10^4
10	32	10^3	9966	$\sim 10^5$	$\sim 3.2 \cdot 10^4$	10^6
13	100	10^4	$\sim 1.3 \cdot 10^5$	$\sim 1.7 \cdot 10^6$	10^6	10^8
17	316	10^5	$\sim 1.7 \cdot 10^6$	$\sim 2.8 \cdot 10^7$	$\sim 3.2 \cdot 10^7$	10^{10}
20	1000	10^6	$\sim 2 \cdot 10^7$	$\sim 4 \cdot 10^8$	10^9	10^{12}

N.B. la fattibilità pratica di un algoritmo è notevolmente sensibile all'**ordine di grandezza** del tempo di esecuzione: 10^5 secondi ~ 1 giorno, 10^8 secondi ~ **3 anni**

proprietà della funzione logaritmo

definizione: funzione inversa dell'esponenziale
ovvero, $\log_b x$ è l'esponente da dare a b per ottenere x :

$$b^{\log_b x} = x$$

$$\log_b b^x = x$$

dalla definizione discendono le proprietà:

reciprocità : $\log_b a = 1 / \log_a b$

additività : $\log_b xy = \log_b x + \log_b y$, $\log_b (x / y) = \log_b x - \log_b y$

notazione per **basi speciali:**

logaritmo naturale : $\ln x = \log_e x = \int_1^x dt/t$

logaritmo binario : $\lg x = \log_2 x$

un'applicazione delle approssimazioni intere del logaritmo binario:

per la codifica binaria di $N+1$ oggetti distinti occorrono $\lfloor \lg N \rfloor + 1 = \lceil \lg(N+1) \rceil$ bit

esercizi

1. dimostrare che $\lfloor \lg N \rfloor + 1 = \lceil \lg(N+1) \rceil$ per ogni intero positivo $N > 0$
2. il testo (Sedgewick, 2003, p. 41) propone due metodi simili, espressi da rispettive istruzioni C++, per il calcolo del più piccolo intero maggiore di $\lg N$:

```
for (lgN = 0; N > 0; lgN++, N /= 2) ;
```

```
for (lgN = 0, t = 1; t < N; lgN++, t += t) ;
```

dimostrare che i due metodi non sono equivalenti; più precisamente, trovare e correggere l'errore nel secondo metodo, posto che il valore della variabile intera `lgN` debba essere, all'uscita dal ciclo, il più piccolo intero maggiore di $\lg N$