Valutazione prestazioni algoritmi di sorting

Diego Biagini

July 8, 2019

1 Introduzione

Una delle classi di problemi più studiate nel campo dell'informatica sono i problemi di ordinamento. Essi possono essere descritti nel seguente modo:
a partire da una sequenza di dati $S = \{x_1, ..., x_n\}$ in ingresso trovare una permutazione $S' = \{x'_1, ..., x'_n\}$ di essa tale che $x'_1 \leq x'_2 \leq ... \leq x_n\prime$.
Per la loro risoluzione sono stati pensati numerosi algoritmi di ordinamento,questi differiscono soprattutto per i campi di applicazione e per il comportamento rispetto a certe sequenze di ingresso.
Per valutare quale sia meglio usare è quindi necessario vedere quanto sono efficienti, ovvero quanto tempo impiegano ad essere eseguiti, rispetto a insiemi di dati particolari e con dimensione sempre crescente.
Gli algoritmi che saranno presi in esame sono **Insertion sort** e **Quicksort**.

2 Cenni teorici

2.1 Tempo di esecuzione di un algoritmo

Nell'analizzare un qualsiasi algoritmo emerge la necessità di definire in modo chiaro le sue prestazioni, senza essere vincolati a fattori come la macchina su cui viene eseguito.
Per questo motivo supponiamo sempre che essi siano eseguiti su una **RAM(Random access machine)**, una macchina astratta con le seguenti caratteristiche:

- può essere eseguita solo un'operazione alla volta(macchina monoprocessore)
- operazioni elementari eseguite in tempo costante
- infinite celle di memoria di dimensione finita

I parametri che di norma vengono presi in considerazione per valutare l'efficienza sono il tempo di esecuzione(misurato in numero di operazioni elementari effettuate) e la quantità di memoria impiegata.

Nel caso di algoritmi il cui scopo è analizzare e modificare grandi quantità di dati, come quelli di ordinamento, c'è bisogno di vedere come questi parametri si comportano all'aumentare della quantità di dati presi in esame.
Per far questo viene usata la **notazione asintotica**, attraverso essa siamo in grado di descrivere un limite stretto(Θ),superiore(O) o inferiore(Ω) di una funzione al tendere di uno dei suoi parametri all'infinito.
Per esempio $f(n) = \Theta(n^2)$ indica che f si comporta come una parabola quando n(il numero di elementi da ordinare nel nostro caso) tende ad infinito, non ci preoccupiamo dei termini di primo o secondo grado.

2.2 Insertion sort

Insertion sort è un algoritmo di ordinamento iterativo utile soprattutto per ordinare insiemi con un numero ridotto di elementi.

Lo pseudocodice è il seguente:

1 Insertion-Sort(A)

```
1: for j ← 2 to A.lenght do
2:     key ← A[j]
3:     i ← j − 1
4:     while  i > 0 and A[i] > key do
5:         A[i + 1] ← A[i]
6:         i ← i − 1
7:     end while
8:     A[i + 1] ← key
9: end for
```

La correttezza dell'algoritmo può essere dimostrata con la sua invariante di ciclo:
prima della j-esima iterazione il sotto-array $A[1...j-1]$ è ordinato e contiene tutti i valori che vi erano presenti inizialmente.
E' facile verificare che il caso migliore di questo algoritmo si ha quando l'array dato è già ordinato, infatti in questo caso non entreremo mai nel ciclo while interno.
Da questo possiamo dire che nel **caso migliore** insertion sort è un $\Theta(n)$.
Il **caso peggiore** si ha invece quando l'array è ordinato in senso decrescente, in questo caso il ciclo while controllerà tutti i valori precedenti a j e il tempo di esecuzione asintotico è $\Theta(n^2)$.
Il caso medio ha un comportamento simile al caso peggiore,infatti mediamente metà elementi di $A[1...j-1]$ sono più piccoli di $A[j]$, nonostante questo abbiamo sempre un comportamento quadratico e possiamo dire che anche nel **caso medio** insertion sort è un $\Theta(n^2)$

2.3 Quicksort

3 Esperimenti svolti

4 Documentazione del codice

5 Risultati sperimentali

6 Analisi e conclusioni