Contents

Complessità computazionale

Possiamo calcolare quanti passi una funzione svolge e quanta memoria usa.

Inoltre, è molto comodo descrivere queste informazioni attraverso delle funzioni matematiche così da confrontare facilmente due o più algoritmi.

Questo si usa perchè il tempo di calcolo dipende dalla macchina su cui si esegue l'algoritmo. Allora assumiamo un tempo fisso per tutti i passi elementari che l'algortmi esegue.

In questo modo il tempo è proporzionale al numero di istruzioni dell'algoritmo.

Esempio:

```
int maximum(const int[] a, size t n) {
    int max = a[0];
    // I.C.: max e' il massimo in a[0..i-1]
    for (size t i = 1; i <= n-1; i++) {
        if (max < a[i]) max = a[i];
    }
    return max;
}
```

Richiede al massimo 2 variabili(max e i) e la dimensione dell'input è al massimo n.

Adesso calcoliamo i passi fondamentali:

```
int maximum(const int[] a, size t n) {
    int max = a[0]; // Costa 3 perchè crea una variabile, legge una cella e
     ↪  l'assegna
    // I.C.: max e' il massimo in a[0..i-1]
    for (size t i = 1; i <= n-1; i++) { // Costa 4*n perchè crea i,
     ↪  l'assegna, la controlla e poi l'aumenta
        if (max < a[i]) max = a[i]; // Tra 4*n e 9*m
    }
    return max; // Costa 2
}

// TOTALE: tra (5 + 9*n) cioè il best case e (5 + 13*n) cioè il worst case
```

Questa è una stima imprecisa

Tra il best case e il worst case notiamo che cambiano solo delle costanti che però non sono rilevanti in quanto potrebbero essere ottimizzate su altre macchine.

A questo punto vogliamo considerare solo l'andamento asintotico di queste funzione f(n).

Esempio:

Dato un algoritmo:

- Best case: f(n) = 9
- Worst case: f(n) = 8*n

Allora in questo caso il worst case è molto peggio del best case.

Distinguiamo le crescite delle funzioni:

- f(x)=x cresce linearmente
- f(x)=x^2 cresce quadraticamente

Quindi un algoritmo con complessità lineare è molto più efficiente di un algoritmo con complessità quadratica.

Notiamo che per un algoritmo f(x) ax^2+bx+c, la parte bx+c diventa osservabile per costanti molto grosse. Quindi se le costanti di una funzione lineare sono molto più grandi di una funzione quadratica, potrebbe essere peggio usare una funzione lineare.

Crescite logaritmiche

se x si moltiplica per m allora y si incrementa di $c * log_b(m)$.

Questa funzione cresce molto lentamente ed è anche meglio quindi della lineare.