14. Complessità Computazionale

Corso di Informatica

Corso di Laurea in Matematica (D.M. 270/04) - A.A. 2020/2021
Angelo Cardellicchio
angelo.cardellicchio@uniba.it
04/11/2020

Outline

- Analisi degli Algoritmi
- Complessità degli Algoritmi
    - Complessità Spaziale
    - Complessità Temporale
- Complessità di caso peggiore

Analisi degli Algoritmi

- Due fasi
- Analisi **a priori**
  - *Teorica*
  - *Assunzioni forti*
  - *Deterministica*
- Analisi **a posteriori**
  - *Empirica*
  - *Dipende dal contesto*
  - *Beneficia di un'analisi statistica*

Complessità degli Algoritmi

- Algoritmo X, un insieme di dati di ingresso di cardinalità N
- Complessità legata a:
  - *Tempo richiesto per l'esecuzione*
  - *Dipende dal numero di operazioni effettuate*
  - *Spazio occupato in memoria durante l'esecuzione*
  - *Dipende dalle variabili presenti in memoria durante l'esecuzione del programma*
  - *Supponiamo implementazione ottimale!*
- Entrambi influenzano la risolvibilità dell'algoritmo.

Complessità Spaziale (1)

- Algoritmo X
- La **complessità spaziale** $S(X)$ è data da due contributi:
  - *una parte fissa S_F…*
  - *…ed una parte variabile $S_V(C)$*
  - *C è un contributo associato alle caratteristiche dell'algoritmo*
  - ***Domanda:** qual è la natura di C? Es. funzione, scalare, etc.*
- *Risulta quindi:*

$$S(X) = S_F + S_V(C)$$

Complessità Spaziale (2)

- Ad esempio

```
SUM_ONE(P, Q)
Step 1 -> START
Step 2 -> R = P + Q + 1
Step 3 -> STOP
```

- Analisi a priori

$$S_{pr}(X) = S_F + S_V(C) = 1 + 3$$

- Analisi a posteriori
  - *Ipotesi: un intero pesa m_{int} bit, con $m_{int} = 32$*

$$S_{po}(X) = S_F + S_V(C) = 4 \cdot m_{int} = 128\ bit$$

Complessità Temporale (1)

- Algoritmo X, un insieme di dati di ingresso di cardinalità N
- La **complessità temporale** $T(I)$ tiene conto del numero di operazioni effettuate.
- Nel caso precedente:
  - *A quanto è pari la complessità temporale a priori?*
  - *A quanto è pari la complessità temporale a posteriori?*

Complessità Temporale (2)

```
SUM_ONE(P, Q)
Step 1 -> START
Step 2 -> R = P + Q + 1
Step 3 -> STOP
```

- Analisi a priori

$$T_{pr}(I) = 3$$

- Analisi a posteriori
  - *Ipotesi: un'addizione richiede un millisecondo, mentre un'assegnazione 2 millisecondi*

$$T_{po}(I) = 4\ ms$$

Complessità di caso peggiore (1)

- Vogliamo conoscere il *limite massimo* di operazioni o spazio necessario
- Usiamo la *notazione O-grande*, o *Big-O notation*, che descrive il limite asintotico superiore di una funzione rispetto ad un'altra
- Ad esempio:

$$T(X) = O(N)$$

- indica che la complessità temporale dell'algoritmo X è nell'ordine di N
  - *Ciò implica che, nel **caso peggiore**, saranno necessarie N operazioni per risolvere l'algoritmo X.*
  - *Per esempio, se $f(x) = 3x^2 + 2x \Rightarrow f(x) = O(x^2)$*

Complessità di caso peggiore (2)

- **Ci sono due regole per determinare la complessità di caso peggiore per una funzione** $f(x)$
- Se $f(x)$ è una somma di più termini, si **considera** solo quello con il **tasso di crescita maggiore**
- Se $f(x)$ è un prodotto di più fattori, si possono **omettere** quelli **costanti**
- Per esempio, se $f(x) = 3x^2 + 2x \Rightarrow f(x) = O(x^2)$

Complessità di caso peggiore (3)

- Ad esempio, per l'algoritmo X:

```
int n = 10;
for(int i = 0; i < n; i++) {
    printf("%d", i);
    i++;
}
```

$$T(X) = \frac{n}{2} \Rightarrow$$
$$\Rightarrow T(X) \in O(n)$$

Domande?