Tutorato 1

1 Esercizi

Es. 1. Computer B è 1000 volte più veloce di Computer A. Computer A usa un algoritmo di costo $O(n \log n)$ mentre Computer B usa un algoritmo di costo $O(n^2)$. Quale computer impiega meno tempo su un problema di dimensione $n = 10^7$? Che succede se cambio n?
Es. 2. Se $f(n) = O(g(n))$ è sempre vero che $g(n) = \Omega(f(n))$?
Es. 3. Dire se le seguenti uguaglianze sono vere o false e perchè:

- $2^{n+1} = O(2^n)$
- $2^n = O(3^n)$
- $3^n = O(2^n)$
- $n^k = O(c^n), k > 0, c > 1$

Es. 4. Calcolare il costo dei seguenti algoritmi:

Algorithm 1

```
for i in range(n) do
  for j in range(n) do
    for k in range(n) do
      c[i,j] += a[i,k]*b[k,j]
    end for
  end for
end for
```

Algorithm 2

```
for i in range(n) do
  for j in range(n) do
    sum[i,j] = 0
    for k in range(i+j) do
      c[i,j] += a[i,k]*b[j,k]
    end for
  end for
end for
```

Algorithm 3

```
for i in range(n) do
  for j in range(i) do
    for k in range(j) do
      c[i,j] += a[i,k]*b[j,k]
    end for
  end for
end for
```

Algorithm 4

```
for i in range(n) do
  for j in range(i) do
    sum += 1
  end for
  for j in range(n-i) do
    sum -= 1
  end for
end for
```

2 Soluzioni

Es. 1.

Chiamiamo t_A e t_B i tempi impiegati da Computer A e Computer B per svolgere un'operazione elementare. Dal momento che Computer B è 1000 volte più veloce di Computer A, abbiamo $t_A = 1000 t_B$. Se chiamiamo T_A e T_B il tempo totale per risolvere il problema possiamo calcolare i due tempi totali moltiplicando il numero di operazioni necessarie a ciascun computer per il tempo necessario a svolgere una singola operazione. Dunque abbiamo:

$$T_A = \underbrace{10^7 \log 10^7}_{\text{num. op. A}} \cdot \underbrace{1000 t_B}_{t_A} = 7 \cdot 10^{10} t_B$$

$$T_B = \underbrace{10^{14}}_{\text{num. op. B}} \cdot t_B = 10^{14}$$

Dunque, Computer A impiegherà meno tempo: anche se è più lento, l'algoritmo più efficiente lo rende più performante.

Esistono n per cui performerà meglio Computer B? Per calcolarlo dobbiamo calcolare per quali n si ha $T_B < T_A$, ovvero

$$n^2 t_B < 10^3 n \log n \, t_B.$$

Risolvendo per n si ottiene $\frac{n}{\log n} < 1000$. Dunque, per valori piccoli di n, Computer B impiegherà meno tempo di Computer A, ma da un certo valore in poi sarà sempre più veloce Computer A.

Es. 2.

Ricordiamo che $f(n) = O(g(n))$ vuol dire che

$$\exists c > 0, \exists n_0 \in \mathbb{N} : \forall n \geq n_0 \quad f(n) \leq c g(n), \tag{1}$$

mentre $g(n) = \Omega(f(n))$ vuol dire che

$$\exists c' > 0, \exists m_0 \in \mathbb{N} : \forall n \geq m_0 \quad g(n) \geq c' f(n). \tag{2}$$

Supponiamo che $f(n) = O(g(n))$, allora esistono c e n_0 per cui (1) è verificato. Posso dividere entrambi i membri della disuguaglianza per c e ottengo:

$$g(n) \geq \frac{1}{c} f(n) \quad \forall n \geq n_0$$

che, scegliendo $c' = \frac{1}{c}$ e $m_0 = n_0$, è esattamente (2). Allo stesso modo si prova che se $f(n) = \Omega(g(n))$ allora $g(n) = O(f(n))$.

Es. 3.

- $2^{n+1} = O(2^n)$ è vero, infatti $2^{n+1} = 2 \cdot 2^n = O(2^n)$;

- $2^n = O(3^n)$ è vero, infatti:

$$2^n \leq c3^n \iff \left(\frac{2}{3}\right)^n \leq c.$$

  Dal momento che $\left(\frac{2}{3}\right)^n$ è una funzone decrescente e per $n = 1$ e $c = 1$ la disuguaglianza è verificata, in (1) posso scegliere $c = 1, n_0 = 1$ per cui la disuguaglianza è sempre vera.

- $3^n = O(2^n)$ è falso. Per dimostrarlo supponiamo che sia vero, e facciamo vedere che arriviamo a una contraddizione. Se fosse vero esisterebbero c e n_0 per cui:

$$3^n \leq c2^n \quad \forall\, n \geq n_0 \text{ ovvero } \left(\frac{3}{2}\right)^n \leq c \quad \forall\, n \geq n_0.$$

  Dal momento che $\left(\frac{3}{2}\right)^n$ è una funzione crescente che tende a infinito, per qualunque scelta di c, ci sarà sempre un n molto grande per cui la disuguaglianza è falsa.

- $n^k = O(c^n)$ è vero per ogni $k > 0$ e $c > 1$, ovvero i polinomi crescono sempre più lentamente di un esponenziale. Per provare questa relazione usiamo il fatto che

  - $f(n) = O(g(n))$ se e solo se $\lim_n \frac{f(n)}{g(n)} \in \mathbb{R}_{\geq 0}$ (tende a 0 o a un numero positivo);
  - $f(n) = \Omega(g(n))$ se e solo se $\lim_n \frac{f(n)}{g(n)} \in \{\mathbb{R}_{>0}, +\infty\}$ (tende a un numero positivo o a $+\infty$).

  Applicando ripetutamente la regola di de l'Hopital, possiamo verificare che $\lim_n \frac{n^k}{c^n} = 0$, e quindi $n^k = O(c^n)$.

Es. 4

Tutti i costi si calcolano in maniera simile, considerando il massimo numero di iterazioni che può essere svolto da ciascun ciclo (caso peggiore). In particolare Algorithm 1, 2, 3 hanno costo $O(n^3)$ mentre Algorithm 4 ha costo $O(n^2)$.