Esercitazione 7
Algorithmi e Strutture Dati (Informatica)
A.A 2015/2016

Tong Liu

April 21, 2016

Elementi fondamentali

Notazioni Asintotiche

Definition 1.1. (Notazione O)
Sia $g(n)$ una funzione di costo; indichiamo con $O(g(n))$ l'insieme delle funzioni $f(n)$ tali per cui:

$$\exists c > 0, m \geq 0 : f(n) \leq cg(n), \forall n \geq m \tag{1.1}$$

$g(n)$ è un limite **asintotico** superiore per $f(n)$.

Definition 1.2. (Notazione Ω)
Sia $g(n)$ una funzione di costo; indichiamo con $\Omega(g(n))$ l'insieme delle funzioni $f(n)$ tali per cui:

$$\exists c > 0, m \geq 0 : cg(n) \leq f(n), \forall n \geq m \tag{1.2}$$

$g(n)$ è un limite **asintotico** inferiore per $f(n)$.

Definition 1.3. (Notazione Θ)
Sia $g(n)$ una funzione di costo; indichiamo con $\Theta(g(n))$ l'insieme delle funzioni $f(n)$ tali che per cui:

$$\exists c_1 > 0, c_2 > 0, m \geq 0 : c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq m \tag{1.3}$$

Tecniche su calcolo della complessità

Data un'equazione di ricorrenza

$$T(n) = aT(n/b) + cn^{\beta} \tag{1.4}$$

Metodo di sostituzione:
Sostituire $T(n)$ con $cg(n)$, si usa spesso in analisi per tentativi.

Metodo di iterazione:
Iterare sull'equazione di ricorrenza $T(n)$, e.g. $T(n) = T(n/2) + d = T(n/4) + 2d = ... = T(1) + \log nd$.

Analisi per tentativi:
Intuire il valore di $g(n)$ e sostituire il valore nella relazione di ricorrenza, dimostrare per l'induzione tale valore è una Soluzione. Si solito, si conclude definendo il valore di c.

Analisi per livelli di ricorsione:
Si espande $T(n)$ e si osserva l'andamento di cn^β alla fine si cerca di stimare il costo di intera procedura.

Teorema delle ricorrenze

Noto anche come teorema pricipale, teorema di esperto, master theorem.

Definition 1.4. Ricorrenze lineari con partizione bilanciata
Siano a e b costanti intere tali che $a \geq 1$ e $b \geq 2$, e c, d e β costanti reali tali che $c > 0, d \geq 0$, e $\beta \geq 0$. Sia $T(n)$ data dalla relazione di ricorrenza:

$$T(n) = \begin{cases} T(n) = d & \text{per } n = 1 \\ T(n) = aT(n/b) + cn^\beta & \text{per } n > 1 \end{cases} \tag{1.5}$$

Posto $\alpha = \frac{\log a}{\log b}$, allora:

$$T(n) = \begin{cases} O(n^\alpha) & \text{se } \alpha > \beta \\ O(n^\alpha \log n) & \text{se } \alpha = \beta \\ O(n^\beta) & \text{se } \alpha < \beta \end{cases} \tag{1.6}$$

Definition 1.5. Ricorrenze lineari di ordine costante
Siano $a_1, a_2, \ldots, a_n b$ costanti intere non negative, con h costante positiva, c e β costanti reali tali che $c > 0$ e $\beta \geq 0$, sia $T(n)$ definita dalla relazione di ricorrenza:

$$T(n) = \begin{cases} \text{costante} & \text{per } n \leq m \leq h \\ \sum_{1 \leq i \leq h} a_i T(n-i) + cn^\beta & \text{altrimenti} \end{cases} \tag{1.7}$$

Posto $a = \sum_{1 \leq i \leq h} a_i$ allora:

$$T(n) = \begin{cases} O(n^{\beta+1}) & \text{se } a = 1 \\ O(a^n n^\beta) & \text{se } a \geq 2 \end{cases} \tag{1.8}$$

Equazioni matematiche utili

$$\sum_{i=0}^{n-1} 2^i = 2^n - 1$$

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$\log xy = \log x + \log y,\ \log \tfrac{x}{y} = \log x - \log y$$

$$\log x^k = k \log x,\ a^{\log_a x} = \log_a a^x = x$$

$$\log_a a = 1,\ \log_a 1 = 0$$

Esercizi

Esercizio 1

Algorithm 1 FUN1

```
1: procedure FUN1(int n)
2:     if n ≤ 2 then
3:         Return n;
4:     else
5:         Return FUN1(n − 1) - 2 FUN1(n − 2);
6:     end if
7: end procedure
```

1, Determinare un limite inferiore sufficientemente accurato del tempo di esecuzione $T(n)$
2, Determinare un limite superiore sufficientemente accurato del tempo di esecuzione $T(n)$

Soluzione

Il tempo di esecuzione T(n) può essere calcolato come Soluzione della seguente equazione di ricorrenza:

$$T(n) = \begin{cases} c_1 & \text{se } n \leq 2 \\ T(n-1) + T(n-2) + c_2 & \text{altrimenti} \end{cases} \tag{1.9}$$

con c_1 e c_2 opportune costanti positive. Per studiare il valore di $T(n)$ è importante osservare che la funzione $T(b)$ è monotona crescente, ossia risulta $T(n) \geq T(n-1)$, da questa osservazione possiamo scrivere

$T(n) = T(n-1) + T(n-2) + c_2$
$\leq 2T(n-1) + c_2$
$\leq 4T(n-2) + 2c_2 + c_2$
$\leq 8T(n-3) + 4c_2 + 2c_2 + c_2$
$\dots$
$\leq 2^k T(n-k) + \sum_{i=0}^{k-1} 2^i c_2$
$\dots$
$\leq 2^n T(0) + \sum_{i=0}^{n-1} 2^i c_2$
$= 2^n c_1 + (2^n - 1)c_2$
$= 2^n (c_1 + c_2) - c_2$

Da cui possiamo concludere che $T(n) = O(2^n)$. In realtà, quando arriviamo al $2T(n-1) + c_2$ possiamo già applicare la teorema delle ricorrenze lineare per concludere il risultato. Per la seconda domanda, ragionando in maniera simile, avremmo:

$T(n) = T(n-1) + T(n-2) + c_2$
$\geq 2T(n-2) + c_2$
$\geq 4T(n-4) + 2c_2 + c_2$
$\geq 8T(n-6) + 4c_2 + 2c_2 + c_2$

$$
\begin{aligned}
&\dots \\
&\geq 2^k T(n-2k) + \sum_{i=0}^{k-1} 2^i c_2 \\
&\dots \\
&\geq 2^{\lfloor \frac{n}{2} \rfloor} T(n - 2\lfloor \frac{n}{2} \rfloor) + \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor - 1} 2^i c_2 \\
&= 2^{\lfloor \frac{n}{2} \rfloor} c_1 + (2^{\lfloor \frac{n}{2} \rfloor} - 1) c_2 \\
&= 2^{\lfloor \frac{n}{2} \rfloor} (c_1 + c_2) - c_2
\end{aligned}
$$

Possiamo concludere che $T(n) = \Omega(2^{\lfloor \frac{n}{2} \rfloor})$.

Esercizio 2

[15/09/2015] Si calcoli la complessità $T(n)$ della seguente procedura (Pseudocodice 2), chiamata con *crazy(A,1,n)*

Algorithm 2 L'algoritmo da calcolare la cui complessià

```
1:  procedure CRAZY(integer[] A, integer i, integer j)
2:      if i ≥ j then
3:          return 2
4:      end if
5:      Integer h ← ⌊(i+j)/2⌋
6:      for Integer k ← i + 1to j − 1 do
7:          h ← h + 1
8:      end for
9:      return ⌊h/2⌋ crazy(A, i + 1, ⌊(i+j)/2⌋) + crazy(A, ⌊(i+j)/2 + 1⌋, j-1)
10: end procedure
```

Soluzione

Ad ogni chiamata della funzione crazy(), il ciclio della linea 6 comporta un costo di $O(n)$, per la procedura ricorsiva della linea 9, infatti comporta un costo di $T(\frac{n}{2}) + T(\frac{n}{2})$. Quindi il costo della procedura è il seguente:

$$T(n) = 2T(n/2) + n \tag{1.10}$$

Applichiamo il teorema delle ricorrenze, otteniamo $T(n) = O(n \log n)$

Esercizio 3

[Libro 2.13] Scrivere un algoritmo ricorsivo che riceve come parametro un vettore di n elementi tale che il suo tempo di esecuzione $T(n)$ verifichi la relazione

$$T(n) = \begin{cases} d & \text{per } n \leq k \\ 5T(n/2) + 2T(n-1) + cn^2 & \text{per } n > k \end{cases} \tag{1.11}$$

con c, d e k costanti opportune. L'ordine di grandezza di $T(n)$ è polimoniale oppure no?

Soluzione

Una possibile funzione (Pseudocodice 3) può essere il seguente:

Algorithm 3 una possibile Soluzione per l'esercizio 3

```
1:  procedure CRAZY(integer[] A,integer i,integer j)
2:      integer c ← 0
3:      if j − i > 10 then
4:          for k ← 1 to 5 do
5:              c ← c+ crazy(A, i + k, ⌊(i + j)/2⌋)
6:          end for
7:          c ← c+ crazy(A, i, j − 1)
8:          c ← c+ crazy(A, i + 1, j)
9:          for k1 ← i to j do
10:             for k2 ← k1 + 1 to j do
11:                 if A[k1] = A[k2] then
12:                     c ← c + 1
13:                 end if
14:             end for
15:         end for
16:     end if
17:     return c
18: end procedure
```

Nota: funzioni in riga 7 e riga 8 hanno lo stesso costo ($T(n-1)$), il ciclio annidato in riga 9 ha il costo $O(n^2)$. Il costo della funzione è polimoniale, si può dimostrare con il metodo di sostituzione.

Esercizio 4 Analisi con scambio di variabile

[Libro 2.14] Si dimostri che la ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq k \\ 2T(\sqrt{n}) + \log n & \text{se } n > k \end{cases} \tag{1.12}$$

con d e k costanti opportune, ha Soluzione $O(\log n \log\log n)$.

Soluzione

Per ridurre l'equazione verso il formato di quello di teorema ddelle ricorrenze, dobbiamo eliminare il $\log n$ Quindi, poniamo $n = 2^m$. La ricorrenza viene riscritta nel modo seguente:

$$T(2^m) = 2T(2^{m/2}) + \log 2^m \tag{1.13}$$

Ora poniamo $S(m) = T(2^m)$, sostituiamo e semplifichiamo:

$$S(m) = 2S(m/2) + m \tag{1.14}$$

la cui Soluzione sappiamo essere $S(m) = \Theta(m \log m)$. Sapendo che $m = \log n$, otteniamo:

$$T(n) = \Theta(\log n \log\log n) \tag{1.15}$$

Esercizio 5.1

[3/06/2015,Libro 2.2] Per dimostrare che una funzione $g(n) = O(f(n))$ si può calcolare il limite per n che tende all'infinito del rapporto $\frac{g(n)}{f(n)}$. Se tale limite esiste ed è uguale ad una costante allora $g(n) = O(f(n))$, mentre se il limite è infinito allora $g(n)$ cresce più di $f(n)$. Si dimostri che $O(\log n) = O(n^a)$ per ogni $a > 0$

Soluzione

$$\lim_{n\to\infty} \frac{n \log n}{n^a} = \lim_{n\to\infty} \frac{\log n}{n^{a-1}}$$

calcolare la derivata applicando la regola de L'Hôpital

$$= \lim_{n\to\infty} \frac{n^{-1}}{(a-1)n^{a-2}}$$

$$= \lim_{n\to\infty} \frac{1}{(a-1)n^{a-1}}$$

$$= 0$$

Esercizio 5.2

Dimostrare $\log n! = \theta(n \log n)$

Soluzione

$\log n! = O(n \log n)$ perchè:
$n! = n \cdot (n-1) \cdot \cdots \cdot 1 \leq nn \ldots n = n^n$
$\log n! \leq \log n^n$
$\log n! = O(n \log n)$

$\log n! = \Omega(n \log n)$ perchè:
Metà dei fattori in $n!$ superano $n/2$, l'altra metà supera 1. Quindi:

$$n! \geq \prod_{i=1}^{\frac{n}{2}} \frac{n}{2} \prod_{i=1}^{\frac{n}{2}} \frac{1}{2}$$

$$= (\tfrac{n}{2})^{\frac{n}{2}}$$

Applicando il logaritmo, otteniamo:
$\log n! \geq \log(\frac{n}{2})^{\frac{n}{2}} = \frac{n}{2}\log\frac{n}{2} = \frac{n}{2}\log n - \frac{n}{2}\log 2 = \Omega(n\log n)$

Esercizio 5.3

Dimostrare $\sum_{i=1}^{n} i^h$ è $O(n^{h+1})$

Soluzione

$\sum_{i=1}^{n} i^h \leq \sum_{i=1}^{n} n^h = nn^h = n^{h+1} = O(n^{h+1})$