Esercitazione per il corso di Elementi di Algoritmi

Dott. Valerio Capozio

28 Aprile 2008

1 Notazioni asintotiche

1.1 Esercizio 1

Siano $f(n)$ e $g(n)$ due funzioni sempre positive. Dimostrare o confutare le seguenti affermazioni.

1. se $f(n) \leqslant g(n) \forall n \geqslant 0$ allora $g(n) - f(n) = \Omega(g(n))$.
2. se $f(n) = O(g(n))$ allora $2^{(f(n))} = O(2^{(g(n))})$.
3. $f(n) = \Theta(f(n)/3)$
4. $f(n) = \Theta(f(n/3))$
5. $max\{f(n), g(n)\} = \Theta(f(n) + g(n))$.

Soluzioni

1. FALSO. Un controesempio è: $f(n) = n$ e $g(n) = n + 1$. Chiaramente vale $n \leqslant n + 1 \forall n$, ma $1 \neq \Omega(g(n))$.
2. FALSO. Un controesempio è: $f(n) = n$ e $g(n) = \frac{n}{2}$. Chiaramente vale $n = O(\frac{n}{2})$ ma $2^n \neq O(2^{\frac{n}{2}})$. Infatti

$$\lim_{n\to\infty} \frac{2^n}{2^{\frac{n}{2}}} = \lim_{n\to\infty} \frac{2^{\frac{n}{2}} \cdot 2^{\frac{n}{2}}}{2^{\frac{n}{2}}} = \infty$$

3. VERO. Infatti esistono tre costanti c_1, c_2, n_0 t.c. $c_1 \frac{1}{3} f(n) \leqslant f(n) \leqslant c_2 \frac{1}{3} f(n) \forall n \geqslant n_0$. Per esempio $c_1 = c_2 = 3$ e $n_0 = 0$
4. FALSO. Un controesempio è: $f(n) = 2^n$ e quindi $f(\frac{n}{3}) = 2^{\frac{n}{3}}$. Infatti $2^n \neq \Theta(2^{\frac{n}{3}})$ perchè

$$\lim_{n\to\infty} \frac{2^n}{2^{\frac{n}{3}}} = \lim_{n\to\infty} \frac{2^{\frac{n}{3}} \cdot 2^{\frac{n}{3}} \cdot 2^{\frac{n}{3}}}{2^{\frac{n}{3}}} = \infty$$

5. VERO. Applicando la definizione di Θ, poichè $\frac{1}{2}(f(n)) + g(n)) \leqslant \max\{f(n), g(n)\} \leqslant f(n) + g(n)$
   $\forall n$ e $c_1 = \frac{1}{2}, c_2 = 1, n_0 = 0$

1.2 Esercizio 2

Per ogni coppia di funzioni f(n) e g(n), si dica se $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$ o $f(n) = \Theta(g(n))$.

Soluzioni

f(n)	$9^{\log_3 n}$	$n \log\log n$	$2^{n/2}$	1	$n^2 + 8n$	n^n	2^n	$4^{\log_2 n}$
g(n)	$n\sqrt[2]{n}$	$n^{1+\epsilon}, \epsilon > 0$	$n^{\log\log n}$	2^9	$(n \log n)^2$	n!	$2^{n/4}$	$n^{\log_2 4}$
O		X		X	X			X
Ω	X		X	X		X	X	X
Θ				X				X

Per quanto riguarda le funzioni della sesta colonna si può notare che:

$$\lim_{n\to\infty} \frac{n^2 + 8n}{(n \log n)^2} = \lim_{n\to\infty} \frac{n(n+8)}{n^2 \log^2 n} = \lim_{n\to\infty} \frac{n(1+8/n)}{n \log^2 n} = \lim_{n\to\infty} \frac{1}{\log^2 n}$$

Infine si osserva che il limite $\lim_{n\to\infty} \frac{1}{\log^2 n}$ tende a 0 e quindi $f(n) \in O(g(n))$.

Per quanto riguarda le funzioni dell'ultima colonna si può osservare quanto segue: $4^{\log_2 n} = 2^{\log_2 n} \cdot 2^{\log_2 n} = n \cdot n = n^2$.
$n^{\log_2 4} = n^2$.
Avendo ottenuto, dopo le semplificazioni la medesima funzione n^2 è ovvio che questa sia Θ di sé stessa.

1.3 Esercizio 3 Ripreso dall'appello del 25-09-07

Sia k una costante intera e $\forall i \in \{1, ..., k\}$ sia $f_i(n) : \mathbb{N} \to \mathbb{R}^+$ una funzione. Inoltre sia $g(n) : \mathbb{N} \to \mathbb{R}^+$ un'altra funzione. Dimostrare o confutare la seguente relazione

$$\sum_{i=1}^{k} f_i(n) = O(g(n))$$

quando

1. $f_{i-1}(n) = o(f_i(n)) \forall i = 2, ..., k$ e $f_1(n) = o(g(n))$
2. $f_i(n) = O(g(n)) \forall i = 1, ..., k$

Soluzioni
Si vedano le soluzioni dello scritto.

2 Ordinamenti

2.1 Definizioni

Un heap associato ad un insieme di elementi è un albero binario radicato che rispetta le seguenti proprietà.

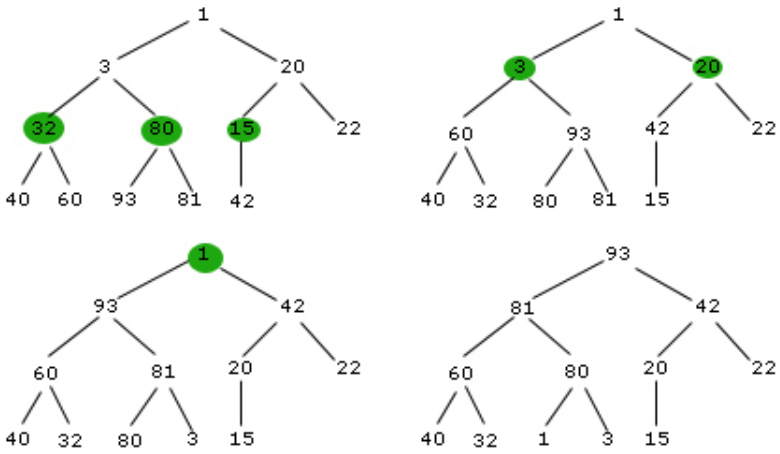
1. Struttura: l'albero è completo almeno fino al penultimo livello.
2. Contenuto: gli elementi dell'insieme sono memorizzati nei nodi dell'albero. Ogni nodo v memorizza uno ed un solo elemento, che denoteremo con chiave(v) e ogni elemento è memorizzato in un solo nodo.
3. Ordinamento heap: il valore dell'elemento in un nodo è sempre maggiore o uguale al valore degli elementi figli.

2.2 Esercizio 1

Considerare la seguente sequenza di numeri:
1 3 20 32 80 15 22 40 60 93 81 42
Assumendo che siano memorizzati in un array secondo la rappresentazione posizionale, discutere, motivando la risposta, se la sequenza rappresenta un heap. In caso la sequenza non sia un heap, applicare la funzione heapfy per renderla tale.

Soluzione

La struttura dati non è un heap poiché alcuni nodi dell'albero hanno valori maggiori rispetto a quelli dei nodi padre(esempio nodo 22).

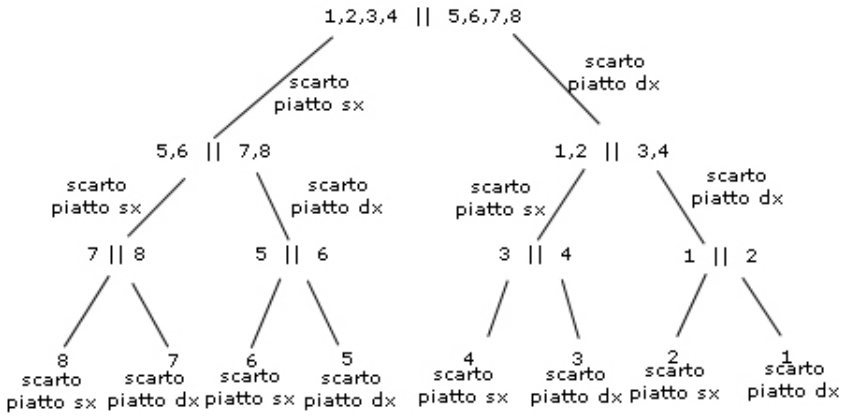


2.3 Esercizio 2

Siano date n monete d'oro (si assuma pure che n sia una potenza di 2) tutte dello stesso peso, eccetto una più leggera delle altre, e una bilancia con due piatti, su ciascuno dei quali è possibile mettere un numero qualunque di monete e sapere se i piatti hanno o no lo stesso peso, o quale dei due è più leggero. Progettare un algoritmo per trovare la moneta leggera che richieda $O(\log n)$ pesate nel caso peggiore. Disegnare poi l'albero di decisione corrispondente nel caso $n = 8$, assumendo che ogni nodo dell'albero corrisponda alla pesatura di due sottoinsiemi di monete.

Soluzione

Un possibile algoritmo può essere il seguente. All'istante iniziale si pongono sul piatto di sinistra gli elementi che vanno da 1 a $n/2$ e sul piatto di destra quelli che vanno da $(n/2) + 1$ a n. Si effettua il confronto scartando la metà più pesante. Quindi si ridivide l'insieme restante nuovamente a metà e si itera il procedimento fino a quando non si individua la moneta più leggera che sarà l'unica a rimanere sul piatto. Dopo l'i-esima pesata l'insieme di monete nel quale si sta cercando quella falsa contiene $\frac{n}{2^i}$ monete. Da questo segue che dopo $\log_2 n$ pesate si è individuata la moneta falsa.



2.4 Esercizio 3 Ripreso dall'appello del 28-01-08

Siano $A = \{a_1, ..., a_n\}$ e $B = \{b_1, ..., b_n\}$ due insiemi di interi. Si realizzi un algoritmo che, presi in input i due insiemi memorizzati in due array, restituisca la loro intersezione. L'algoritmo deve avere complessità temporale $o(n^2)$.
Soluzione
Si vedano le soluzioni dello scritto.