Elementi di Algoritmi e Strutture Dati

Testo della prova scritta del 28 gennaio 2008

docente: Luciano Gualà

Cognome:.................... Nome:....................... Matr.:....... Corso di Laurea:...............

Esercizio 1 [8 punti] Sia $k > 1$ una costante reale, e siano $f(n) : \mathbb{N} \to \mathbb{R}^+$ e $g(n) : \mathbb{N} \to \mathbb{R}^+$ due funzioni. Dimostrare o confutare la seguente relazione

$$\log\left(n^{f(n)}\right) = O\left(\log^k\left(n^{g(n)}\right)\right)$$

quando

(a) $f(n) = o\left(g(n)^{2k}\right)$.

(b) $f(n) = \Theta(\log n)$.

Soluzione esercizio 1 Poiché $\log^k(n^{g(n)}) = (\log\left(n^{g(n)}\right)^k = (g(n)\log n)^k = g(n)^k \log^k n$, possiamo riscrivere la relazione nel seguente modo:

$$f(n)\log n = O(g(n)^k \log^k n),$$

da cui segue:

(a) Falsa. Infatti, si consideri il seguente controesempio: $k = 2$, $g(n) = n$, $f(n) = n^3$. Si ha $f(n) = n^3 = o(n^4)$ $(g(n)^{2k} = n^4)$, ma $f(n)\log n = n^3 \log n = \omega(n^2 \log^2 n)$ $(g(n)^k \log^k n = n^2 \log^2 n)$.

(b) Falsa. Un controesempio è il seguente: $f(n) = \log n$, $g(n) = 1$, $k = 3/2$. Sostituendo si ottiene:

$$\log^2 n = O(\log^{3/2} n),$$

che è chiaramente falsa.

Esercizio 2 [8 punti] Siano $A = \{a_1, \ldots, a_n\}$ e $B = \{b_1, \ldots, b_n\}$ due insiemi di interi. Si realizzi un algoritmo che, preso in input i due insiemi memorizzati in due array, restituisce la loro intersezione, cioè l'insieme composto da tutti gli elementi che sono sia in A che in B. L'algoritmo deve avere complessità temporale $o(n^2)$. *Attenzione*: l'esercizio sarà valutato solo se corredato da adeguata descrizione del funzionamento dell'algoritmo, in base ai seguenti parametri: correttezza, efficienza e analisi di complessità.

Soluzione esercizio 2 Di seguito si discuteranno due diverse soluzioni. Entrambe avranno complessità temporale pari a $O(n \log n)$. L'intersezione di A e B è un insieme C che può essere mantenuto tramite un array (di al più n elementi), o una lista. Lo pseudocodice dei due algoritmi è lasciato come esercizio agli studenti.

Prima soluzione. Si ordinano entrambi i vettori A e B in ordine crescente. Poi si confrontano A e B alla maniera, per capirsi, dell'operazione Merge del Mergesort (si pongono un indice i all'inizio di A e un indice j all'inizio di B; si confronta $A[i]$ con $B[j]$: se essi sono uguali l'elemento si inserisce in C e si incrementano sia i che j, se $A[i]$ è minore si incrementa i, se $B[j]$ è minore si incrementa j e si ripete il confronto finché non si raggiunge la fine di uno dei due vettori). Ordinare i due vettori costa $O(n \log n)$ (se si usa per esempio l'Heap-sort). L'operazione di costruzione di C, invece, prende tempo $O(n)$, perché ad ogni confronto viene incrementato almeno un indice di uno dei due vettori. Quindi la complessità temporale dell'algoritmo è $O(n \log n)$.

Seconda soluzione Si ordina uno solo dei due vettori, per esempio B. Poi si scorre A dall'iniziono alla fine con un indice i che parte da 1 fino a n. Per ogni i si controlla se l'elemento $A[i]$ è presente in B tramite una ricerca binaria. Se è presente si inserisce $A[i]$ in C, altrimenti si passa a considerare il prossimo elemento di A. Per quanto riguarda la complessità, B può essere ordinato in tempo $O(n \log n)$. Si effettuano n ricerche binarie, ognuna delle quali ha costo temporale $O(\log n)$. Quindi, la complessità di questo secondo algoritmo è ancora $O(n \log n)$.

Esercizio 3 [8 punti] Sia $A = [2, 7, 5, 15, 12, 6, 10]$ un vettore posizionale rappresentante un albero binario completo.

(a) Si mostri l'applicazione dell'algoritmo `Heapify`(A) che trasforma A in un heap.

(b) Sia T l'heap risultante dal punto (a). Si mostri l'operazione di estrazione del massimo da T.

(c) Sia T' l'heap risultante dal punto (b). Si mostri l'ordine di visita dei nodi, se si visita T' mediante una visita in profondità simmetrica e mediante una visita in postordine.

Soluzione esercizio 3 Si veda la figura.

Esercizio 4 [8 punti] Si descriva in modo sintetito e preciso l'algoritmo `RadixSort`, analizzandone in particolare la complessità temporale nel caso peggiore.

Soluzione esercizio 4 Si consulti il libro di testo.

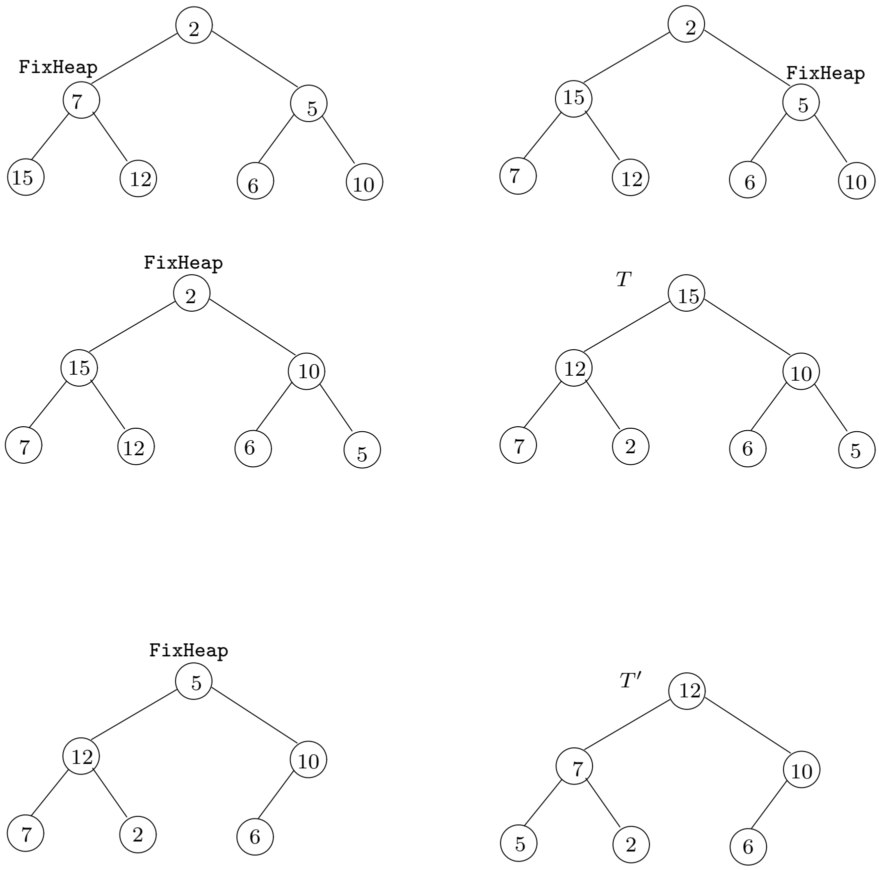

Figura 1: I primi quattro alberi mostrano l'esecuzione di `Heapify`(A), mentre gli altri due l'estrazione del massimo da T. L'ordine di visita dei nodi di T' è: (i) profondità simmetrica: 5,7,2,12,6,10; (ii) postordine: 5,2,7,6,10,12.