Prof. L. Margara **Scritto del 12/7/2021**

Algoritmi e Strutture Dati

Esercizio 1.[11 punti]
Fornire un algoritmo in pseudocodice (quello con il costo computazionale più basso possibile nel caso pessimo) che dati in input un albero binario di ricerca T e due numeri interi positivi a e b con $a \leq b$ ritorni il numero di elementi compresi strettamente tra a e b.
Discutere le correttezza e il costo computazionale dell'algoritmo proposto.

Traccia della soluzione dell'esercizio .
Qual è il caso pessimo?
Se ogni nodo avesse solo il figlio sinistro (tranne l'unica foglia che non ha figli) avremmo il massimo memorizzato nella radice e il minimo memorizzato nell'unica foglia. Se il massimo fosse b e il minimo fosse a, qualunque algoritmo per calcolare la soluzione dovrebbe (partendo dalla radice) scorrere tutti gli elementi fino ad arrivare alla foglia. Quindi il tempo necessario sarebbe almeno pari al numero dei nodi dell'albero.
L'algoritmo che ordina il BST (costo lineare) e che poi calcola gli elementi compresi tra a e b (costo lineare) è quindi ottimo nel caso pessimo.

Esercizio 2.[11 punti]
Risolvere in ordine di grandezza la seguente equazione ricorsiva.

$$T(n) = \begin{cases} 1 & \text{if } n \leq 2, \\ T(n/2) + log_2(n) & \text{if } n > 2. \end{cases}$$

Traccia Soluzione esercizio

$$\begin{aligned} T(n) &= T(n/2) + log_2(n) \\ &= T(n/4) + log_2(n/2) + log_2(n) \\ &\vdots \\ &= log_2(n) + log_2(n/2) + log_2(n/4) + log_2(n/8) + log_2(n/16) + \cdots + 1 \\ &= 1 + 2 + 3 + \cdots + log_2(n) \\ &= \Theta(log^2(n)) \end{aligned}$$

Esercizio 3.[11 punti]
Sia $P = \{p_1, p_2, \ldots, p_n\}$ un insieme di $n \geq 2$ numeri interi positivi. Dare un algoritmo (in pseudocodice) Greedy polinomiale per partizionare P in due insiemi P_1 e P_2 in modo che la quantità

$$|Somma(P_1) - Somma(P_2)|$$

sia minimizzata. Determinare se l'algoritmo proposto è ottimo oppure no. Calcolare il costo computazionale dell'algoritmo proposto.

Traccia Soluzione esercizio.

INSIEMI-BILANCIATI(P)

1 $P_1 = \emptyset$
2 $P_2 = \emptyset$
3 **for** $i = 1$ **to** n
4 **if** $Somma(P_1) \leq Somma(P_2)$
5 Aggiungi p_i a P_1
6 **else** Aggiungi p_i a P_2
7 **return** P_1 e P_2

Controesempio: $P = \{1, 1, 1, 3\}$
Risultato dell'algoritmo: $P_1 = \{1, 1\}, P_2 = \{1, 3\}$
Risultato ottimo: $P_1 = \{1, 1, 1\}, P_2 = \{3\}$
Costo computazionale: $\Theta(n)$.