Informatica Generale
Secondo Esonero

docenti:
Giancarlo Bongiovanni, Tiziana Calamoneri, Ivano Salvo
Sapienza Università di Roma

11 giugno 2015

Esercizio 1 (10 punti) Siano dati due alberi binari di ricerca T_1 con n_1 nodi, e T_2 con n_2 nodi. T_1 e T_2 sono memorizzati tramite record e puntatori; inoltre T_i, $i = 1, 2$, è costituito da una radice r_i, dal suo sottoalbero sinistro T_i^{sx} e dal suo sottoalbero destro T_i^{dx} con la proprietà che per ogni quaterna di chiavi k_1^{sx} in T_1^{sx}, k_1^{dx} in T_1^{dx}, k_2^{sx} in T_2^{sx}, k_2^{dx} in T_2^{dx} vale che $k_1^{sx} < k_2^{sx} < k_1^{dx} < k_2^{dx}$. Si progetti un algoritmo con costo lineare nell'altezza dei due alberi che fonda T_1 e T_2 in un unico albero binario di ricerca con $n_1 + n_2$ nodi (senza creare nuovi nodi).

Dell'algoritmo proposto:

1. **(2 punti)** Si dia la descrizione a parole;
2. **(4 punti)** Si scriva lo pseudocodice;
3. **(2 punti)** Si calcoli il costo computazionale;
4. **(2 punti)** Si discuta la possibilità di utilizzare lo stesso algoritmo nel caso in cui T_1 e T_2 siano alberi Red & Black.

Esercizio 2 (10 punti) Considerare il problema di sostituire tutte le etichette di un albero binario di interi con il prodotto tra il contenuto dell'etichetta e il livello del nodo. Ad esempio, l'albero 3(4,5(6,1(_,2))) viene trasformato nell'albero 0(4,5(12,2(_,6))). Infatti la radice si trova al livello 0, i suoi figli al livello 1, e cosi' via.

- **(5 punti)** Senza scrivere nessun ciclo, scrivere una funzione C
  `void trasformaAlberoRec(tree T)`

che trasforma l'albero `T` come descritto sopra. [SUGG: usare una funzione ausiliaria ricorsiva con parametri ausiliari]

- **(5 punti)** Supponendo di avere a disposizione di un tipo di dato `Queue`, che implementa le code con le usuali operazioni di inserzione in testa, estrazione in coda e test di coda vuota, senza mai scrivere chiamate ricorsive, scrivere una funzione C `void trasformaAlberoIt(tree T)` che trasforma l'albero `T` come descritto sopra.

Esercizio 3 (10 punti) Sia dato il grafo G definito dalle seguenti liste di adiacenza:

1	$\longrightarrow 2 \rightarrow 3 \rightarrow 4$
2	$\longrightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 3$
3	$\longrightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 2$
4	$\longrightarrow 6 \rightarrow 2 \rightarrow 1 \rightarrow 3$
5	$\longrightarrow 2 \rightarrow 3 \rightarrow 6$
6	$\longrightarrow 5 \rightarrow 4 \rightarrow 3$

Utilizzando questa struttura dati:

1. **(2 punti)** Si esegua l'algoritmo di visita in profondità a partire dal nodo 3; si disegni l'albero di visita e si evidenzino gli archi non dell'albero;
2. **(2 punti)** Si esegua l'algoritmo di visita in ampiezza a partire dal nodo 5; si disegni l'albero di visita e si evidenzino gli archi non dell'albero;
3. **(6 punti: 1.5 a domanda)** Si risponda alle seguenti domande sulle proprietà dei due alberi, eventualmente riportando le proposizioni generali dai quali esse si possono dedurre, giustificando le proprie affermazioni:
   - quanti archi ha ciascuno dei due alberi?
   - quanti archi di attraversamento ha l'albero di visita in profondità?
   - quanti archi all'indietro ha l'albero di visita in ampiezza?
   - è possibile dedurre da uno dei due alberi la distanza sul grafo tra due nodi qualunque? se si, da quale?

Nota: Nel risolvere i primi due punti non occorre elencare tutti i passi degli algoritmi di visita, ma gli alberi di visita devono risultare coerenti con l'ordine dei nodi nelle liste di adiacenza.