Algoritmi e Strutture di Dati – Compito da 9 CFU – B

Testo e soluzioni dell'esame a distanza del 1° luglio 2020

Esercizio 1 (3 punti)

Discuti la complessità computazionale della seguente procedura nel caso peggiore fornendo O-grande, Omega e Theta in funzione del numero n di elementi dell'albero.

```
FUNZIONE(T)            /* T è un albero binario di interi */
L.root = NULL          /* L è una lista (vuota) di interi */
FUNZ-RIC(T.root,L,0) /* 0 è la profondità della radice */
return L

FUNZ-RIC(v,L,depth)
if(v==NULL) return
if(depth == 3)
     AGGIUNGI-IN-CODA(L,v.info)
else
     AGGIUNGI-IN-TESTA(L,v.info)
FUNZ-RIC(v.left,L,depth+1)
FUNZ-RIC(v.right,L,depth+1)
```

Assumi che AGGIUNGI-IN-TESTA faccia un numero di operazioni costante, mentre AGGIUNGI-IN-CODA faccia un numero di operazioni proporzionali alla lunghezza della lista corrente. Fai un esempio di tipologia di albero in cui si ha il caso peggiore del costo asintotico.

Soluzione esercizio 1

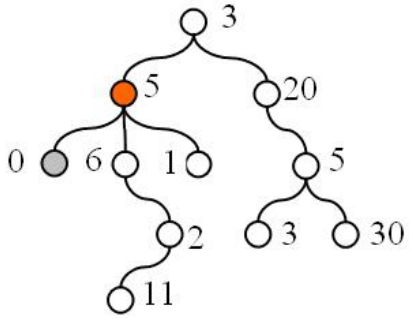
La funzione FUNZIONE(T), oltre a richiamare la funzione FUNZ-RIC(v,L), fa solamente operazioni O(1), quindi le due funzioni hanno la stessa complessità asintotica.

La funzione FUNZ-RIC(v,L) visita l'intero albero. A profondità 3, cioè per otto nodi al massimo, esegue un inserimento in coda. Per tutti gli altri nodi esegue un inserimento in testa. Anche se gli inserimenti in coda costassero Theta(n), poiché non ne vengono fatti in proporzione ad n, essi peserebbero complessivamente Theta(n). Il costo complessivo è dunque Theta(n).

Esercizio 2 (27 punti)

Scrivi in linguaggio C il codice della funzione int conta(nodo* a) che accetti in input un puntatore alla radice di un albero di grado arbitrario di interi. La funzione restituisce il numero dei nodi dell'albero che hanno un valore del campo info pari al numero dei discendenti del nodo stesso.

Per esempio per l'albero in figura la funzione conta() restituisce 2 perché ci sono due nodi (quello rosso e quello grigio) che hanno un valore del campo info (rappresentato affianco al nodo) esattamente uguale al numero dei loro discendenti.



L'albero di grado arbitrario è rappresentato con la struttura dati figlio-sinistro fratello-destro seguente:

```
typedef struct nodo_struct {
    struct nodo_struct* left_child;
    struct nodo_struct* right_sibling;
    int info;
} nodo;
```

Soluzione esercizio 2

```
int discendenti(nodo* a){
     if(a == NULL) return 0;
     int disc = 0;
     nodo* x = a->left_child;
     while(x != NULL){
          disc++;  // x è un discendente di a
          disc = disc + discendenti(x); // i discendenti
                                        // di x sono anche
                                        // discendenti di a
          x = x->right_sibling;
     }
     return disc;
}
```

```
/* Percorro l'albero di grado arbitrario come se fosse un
albero binario, tanto devo solo contare i nodi che
soddisfano la proprietà */

int conta(nodo* a){
     if(a == NULL) return 0;
     int cont = 0;
     if(discendenti(a)==a->info)
          cont = cont + 1;
     return cont + conta(a->left_child) +
                   conta(a->right_sibling);
}
```