Algoritmi e Strutture di Dati – A.A. 2018-2019
Simulazione d'esame del 23/01/19 – D.M. 270-9CFU
Libri e appunti chiusi – Tempo = 2:00h

9

☐ Note (es: correzione veloce, eventuali indisponibilità, ecc.) ……
……
……

Cognome: _____ **Nome:** _____ **Matricola:** _____

DOMANDA SULLA COMPLESSITA' ASINTOTICA (3 punti su 30)

Discuti la complessità computazionale nel caso peggiore (in termini di O-grande, Omega e Theta) delle seguenti procedure in funzione del numero n di elementi dell'albero. Assumi che

- AGGIUNGI_IN_TESTA faccia un numero di operazioni costante
- AGGIUNGI_IN_CODA faccia un numero di operazioni proporzionali alla lunghezza della lista passata in input

```
CONTIENE_DOPPIONE(T)       /* T è un albero binario di interi */

L.head = NULL    /* L è una nuova lista (vuota) di interi */
FUNZ_RIC(T.root,L,0)
return L

FUNZ_RIC(v,L,p)  // p è la profondità del nodo

 if(v==NULL) return
 if(p < 10)
     AGGIUNGI_IN_CODA(L,v.info)
 else
     AGGIUNGI_IN_TESTA(L,v.info)
 FUNZ_RIC(v.left,L,p+1)
 FUNZ_RIC(v.right,L,p+1)
```

ALGORITMO IN LINGUAGGIO C (27 punti su 30)

Scrivi in linguaggio C il codice della funzione

```
int verifica_figli_componente(grafo* g, nodo_albero* a)
```

che accetti in input un puntatore ad grafo non orientato **g** rappresentato tramite oggetti e un puntatore **a** alla radice di un albero di grado arbitrario di interi rapresentato tramite la struttura figlio-sinistro (`left_child`) e fratello destro (`right_sibling`). La funzione restituisce 1 se esiste un nodo dell'albero **a** che ha tanti figli quanti sono i nodi della componente connessa più grande del grafo **g**, altrimenti la funzione restituisce 0. Se grafo e albero sono entrambi vuoti (cioè uguali a NULL) la funzione ritorna true. Se uno è vuoto e uno no, allora ritorna false.

Usa le seguenti strutture (che si suppone siano contenute nel file "`strutture.h`"):

```
typedef struct nodo_struct {
        elem_nodi* pos;  /* posizione nodo nella
                            lista del grafo */
        elem_archi* archi;   // lista archi incidenti
        int color;
} nodo;

typedef struct arco_struct {
        elem_archi* pos;  // pos. arco lista grafo
        nodo* from;
        nodo* to;
        elem_archi* frompos;  // pos. arco nodo from
        elem_archi* topos;      // pos. arco nodo to
} arco;

typedef struct elem_lista_nodi {
        struct elem_lista_nodi* prev;
        struct elem_lista_nodi* next;
        nodo* info;
} elem_nodi;   // elemento di una lista di nodi

typedef struct elem_lista_archi {
        struct elem_lista_archi* prev;
        struct elem_lista_archi* next;
        arco* info;
} elem_archi;  // elemento di una lista di archi

typedef struct {
        int numero_nodi;
        int numero_archi;
        elem_archi* archi;  // lista degli archi
        elem_nodi* nodi;    // lista dei nodi
} grafo;
```

```
typedef struct nodo_albero_struct {
        struct nodo_albero_struct* left_child;
        struct nodo_albero_struct* right_sibling;
        int info;
} nodo_albero;
```

È possibile utilizzare qualsiasi libreria nota e implementare qualsiasi funzione di supporto a quella richiesta.