Algoritmi e Strutture di Dati – A.A. 2019-2020
Esame scritto del 28/01/20 – D.M. 270-9CFU
Libri e appunti chiusi – Tempo = 2:00h

9

☐ Note (vincoli, indisponibilità, preferenze, ecc.) ……

……

……

N.B.: gli esami orali si volgeranno dal 28/1 al 14/2 e dal 24/2 al 28/2

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

DOMANDA SULLA COMPLESSITA' ASINTOTICA (3 punti su 30)

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,L1,L2)      /* T è un albero binario di interi */
                       /* L1 ed L2 sono due liste (vuote) di interi */
FUNZ-RIC(T.root,L1,L2)
```

$\Theta(F\_Ric)$

```
FUNZ-RIC(v,L1,L2)
if(v==NULL) return
if(v.left == NULL and v.right == NULL)
      AGGIUNGI-IN-CODA(L1,v.info)
else
      AGGIUNGI-IN-TESTA(L2,v.info)
FUNZ-RIC(v.left,L1,L2)
FUNZ-RIC(v.right,L1,L2)
```

$\lceil \frac{n}{2} \rceil$ $\quad \Theta(n^2)$

$1 \quad m \quad 1 \quad 2^h - 1 \quad m \quad m$

$(n+n)+(n(\frac{n}{2}))$

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.

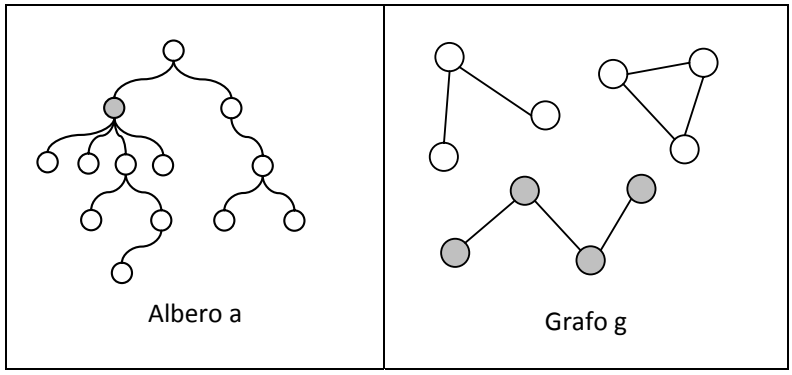
Descrivi anche un albero su cui si verifica il caso peggiore delle due funzioni e un albero su cui non si verifica il caso peggiore.

ALGORITMO IN LINGUAGGIO C (27 punti su 30)

Scrivi in linguaggio C il codice della funzione

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

che accetti in input un puntatore **a** alla radice di un albero di grado arbitrario di interi e un puntatore **g** ad grafo non orientato rappresentato tramite oggetti e riferimenti. La funzione restituisce 1 se esiste nodo dell'albero che ha tanti figli quanti sono i nodi di una componente connessa del grafo **g**, altrimenti la funzione restituisce 0. Se uno (o entrambi) tra grafo e albero è vuoto (cioè uguale a NULL) la funzione ritorna 0.



Per esempio l'albero **a** in figura ha un nodo (quello grigio) con quattro figli e una componente connessa (quella grigia) con 4 nodi, dunque **verifica**(a,g) ritorna 1 (true).

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;

/* struttura per l'albero di grado arbitrario */

typedef struct nodo_albero_struct {
        struct nodo_albero_struct* left;
        struct nodo_albero_struct* right;
        int info;
} nodo_albero;
```

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