

**Algoritmi e Strutture di Dati – A.A. 2022-2023**  
**Esame scritto del 16/02/23 – 9CFU**  
**Libri e appunti chiusi – Tempo = 2:00h**

9-B

☐ Preferenze o indisponibilità per la data dell'orale (entro il 28/02).....

Cognome: \_\_\_\_\_ Nome: \_\_\_\_\_

**DOMANDA SULLA COMPLESSITA' ASINTOTICA (3 punti su 30)**

Discuti la complessità computazionale delle seguenti procedure 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.head = NULL      /* L è una nuova lista (vuota) di interi */
    FUNZ_RIC(T.root, L)
    return L

FUNZ_RIC(v, L)
    if( v==NULL ) return
    if( LUNGHEZZA(L) <= 10 )
        AGGIUNGI_IN_TESTA(L,v.info)
        FUNZ_RIC(v.left, L)
    else
        AGGIUNGI_IN_TESTA(L,v.info)
        FUNZ_RIC(v.right, L)
```

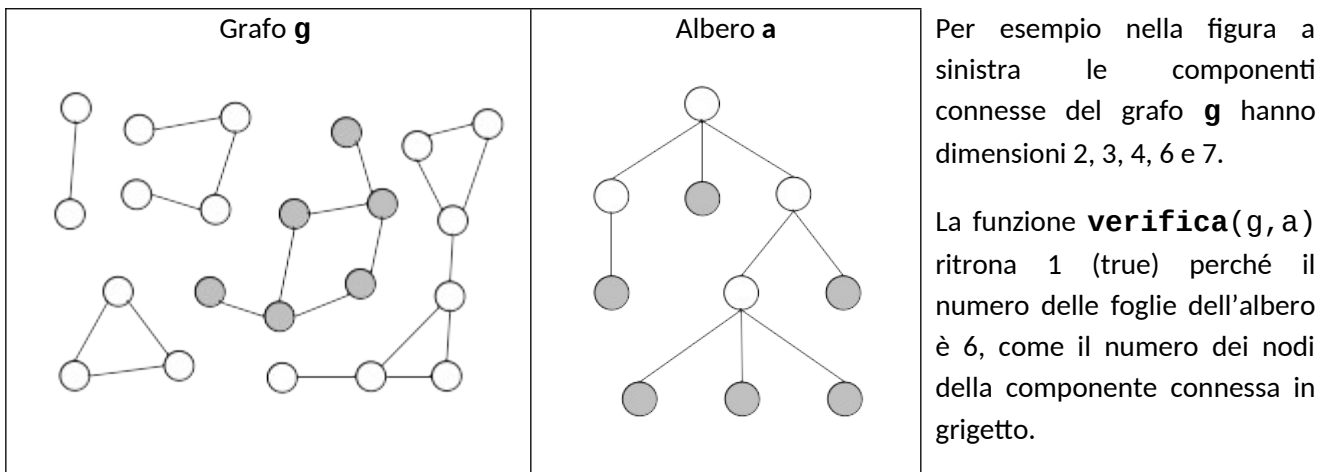
Assumi che sia **LUNGHEZZA** che **AGGIUNGI\_IN\_CODA** facciano un numero di operazioni proporzionale al numero degli elementi della lista mentre **AGGIUNGI\_IN\_TESTA** faccia un numero di operazioni costante. Descrivi una tipologia di alberi in cui si verifica il caso peggiore (ed una in cui non si verifica il caso peggiore).

## ALGORITMO IN LINGUAGGIO C (27 punti su 30)

Scrivi in linguaggio C il codice della funzione

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

che accetti in input un puntatore **g** ad un grafo non orientato rappresentato tramite oggetti e riferimenti e un puntatore **a** alla radice di un albero di grado arbitrario. La funzione restituisce in output 1 (true) se esiste una componente connessa di **g** che ha tanti nodi quante sono le foglie dell'albero **a**. La funzione ritorna 0 (false) altrimenti. Se il grafo o l'albero (o entrambi) sono NULL la funzione ritorna 0 (false).



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

<pre>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</pre>	<pre>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_child;     struct nodo_albero_struct* right_sibling;     int info; } nodo_albero;</pre>
--	--

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