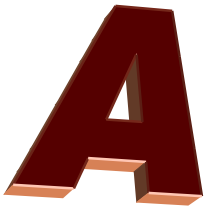
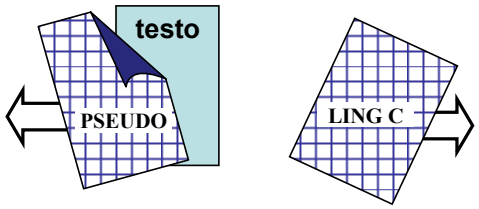


Algoritmi e Strutture di Dati – A.A. 2013-2014
Prova scritta del 31 gennaio 2014
Libri e appunti chiusi
Tempo: D.M. 270 = 2:15h; D.M. 509 = 2:00h

☐ Ho bisogno di una correzione veloce in quanto _____

_____

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



- **CONSEGNA PSEUDOCODIFICA E LINGUAGGIO C SU DUE FOGLI PROTOCOLLO SEPARATI**
- **METTI IL TESTO DENTRO LA PARTE DI PSEUDOCODIFICA**
- **PUOI SCRIVERE (E CONSEGNARE) A MATITA**

PSEUDOCODIFICA

Negli esercizi seguenti un grafo non orientato è rappresentato con un array A in cui ogni elemento A[*u*] è un riferimento al primo elemento della lista di adiacenza doppiamente concatenata del nodo *u* (con i campi `prev`, `info` e `next`). Essendo il grafo non orientato esiste un arco (*u*,*v*) per ogni arco (*v*,*u*).

Un albero di grado arbitrario `T` nella rappresentazione figlio-sinistro fratello-destro è un riferimento ad un oggetto che ha il solo campo `T.root`, che è un riferimento al nodo radice dell'albero. Ogni nodo `n` ha i campi `n.parent`, `n.left`, `n.right`, ed `n.info`, dove i primi tre campi sono riferimenti ad altri nodi (oppure NULL), mentre il quarto campo è un numero intero.

Esercizio 1 (tutti gli studenti)

Scrivi lo pseudocodice della procedura COPIA-ALBERO(T) che accetti in input un albero di grado arbitrario T e, senza modificare T, ritorni una copia di T (l'ordine dei figli di ogni nodo non è significativo). Puoi assumere che, se T ha n nodi, questi abbiano nei valori del campo `info` tutti gli interi da 0 a n-1.

Esercizio 2 (solo studenti D.M. 270/99, 6 CFU)

Scrivi lo pseudocodice della procedura ORDINE-BFS(A,u) che accetti in input un grafo non orientato A e l'indice di un nodo *u* e restituisca una lista L di interi in cui gli indici dei nodi compaiono nell'ordine con cui sono stati raggiunti nella visita in ampiezza di A a partire da *u*. Assumi che il grafo sia connesso. Scrivi anche la procedura che aggiunge un elemento alla lista.

Esercizio 3 (solo studenti D.M. 509/99, 5 CFU)

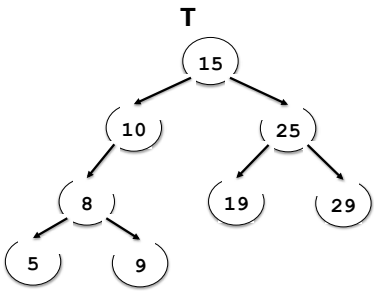
Scrivi lo pseudocodice della procedura RIMOZIONE(L,A) che accetti in input una lista di interi doppiamente concatenata L e un array di interi A e rimuova da L (se esistono) tutti gli elementi di A.

Esercizio 4

Discuti la complessità computazionale nel caso peggiore (in termini di O-grande, Omega e Theta) delle procedure che hai proposto per gli esercizi precedenti.

LINGUAGGIO C

Si consideri un albero binario di ricerca i cui nodi contengono un numero intero maggiore di tutti i valori contenuti nel sottoalbero sinistro e minore di tutti i valori contenuti nel sottoalbero destro. La **dimensione** dell'albero (numero di nodi) **non è nota a priori**. Un livello dell'albero può essere memorizzato in una **lista** ordinata in modo strettamente crescente. Si consideri l'esempio di seguito



Livello 2

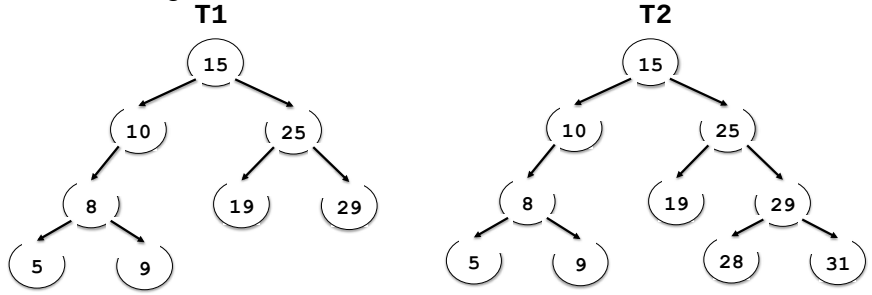
L = 8 ➔ 19 ➔ 29

Si definisca un progetto in linguaggio C in cui siano implementate le seguenti funzionalità:

1) le strutture dati più adeguate per rappresentare l'albero binario di ricerca e una lista, e i file **header** del progetto (**.h**);

2) la funzione **`allocaLista`** che dato l'albero binario di ricerca **`T`** e un valore intero **`v`** allochi dinamicamente e restituisca una lista **`L`** contenete i valori dei nodi posizionati al livello **`v`** di **`T`**. Gli elementi in **`L`** devono essere ordinati in modo strettamente crescente. Se l'albero **`T`** non ha nessun nodo al livello **`v`**, bisogna restituire una lista vuota.

3) la funzione **`verificaLivelli`** che dati due alberi binari di ricerca **`T1`** e **`T2`** e un valore intero **`v`** restituisca il numero di nodi che il livello **`v`** di **`T1`** e il livello **`v`** di **`T2`** hanno in comune. Si considerino i seguenti due alberi **`T1`** e **`T2`** e il livello **`v`**=3.



la funzione **`verificaLivelli`** restituirà il numero 2.

4) la funzione **`ModificaFile`** che accetti come parametri l'albero binario di ricerca **`T`** e il nome di un file di testo "**`livello.txt`**", contenente in ogni riga due numeri interi **`x`** e **`y`**, separati da spazio. La funzione **`ModificaFile`** per ogni riga del file, deve verificare che il valore **`y`** sia esattamente il numero di nodi a livello **`x`** di **`T`** e, se ciò non è vero, deve sovrascrivere **`y`** con il valore corretto. Si consideri ad esempio l'albero binario **`T`** della prima figura e il seguente file:

```
3 2
0 1
2 4
```

Il file verrà modificato come segue

```
3 2
0 1
2 3
```

E' possibile definire metodi di supporto e utilizzare ogni libreria conosciuta.