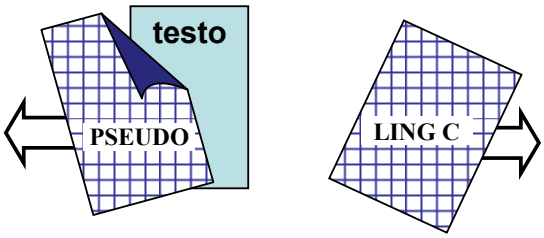
Algoritmi e Strutture di Dati – D.M. 270/04
Appello dell'11 giugno 2013 – 2 ore e 15 minuti
Libri e appunti chiusi

☐ Ho bisogno di una correzione veloce in quanto _____

_____

Cognome: _____ Nome: _____Matricola:_____



CONSEGNA PSEUDOCODIFICA E LINGUAGGIO C SU DUE FOGLI PROTOCOLLO SEPARATI

PUOI SCRIVERE (E CONSEGNARE) A MATITA

LA BRUTTA LA PUOI TENERE

PSEUDOCODIFICA

Esercizio 1

Un albero binario `T` 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.

Scrivi lo pseudocodice della procedura CAMMINO_PIU_CORTO(T) che prende in input un albero binario T e produce in output la lunghezza (numero di archi) del cammino più corto dalla radice ad una foglia dell'albero T.

Esercizio 2

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*).

La distanza di un nodo *u* da un nodo *v* è il numero degli archi del cammino più breve da *u* a *v*.

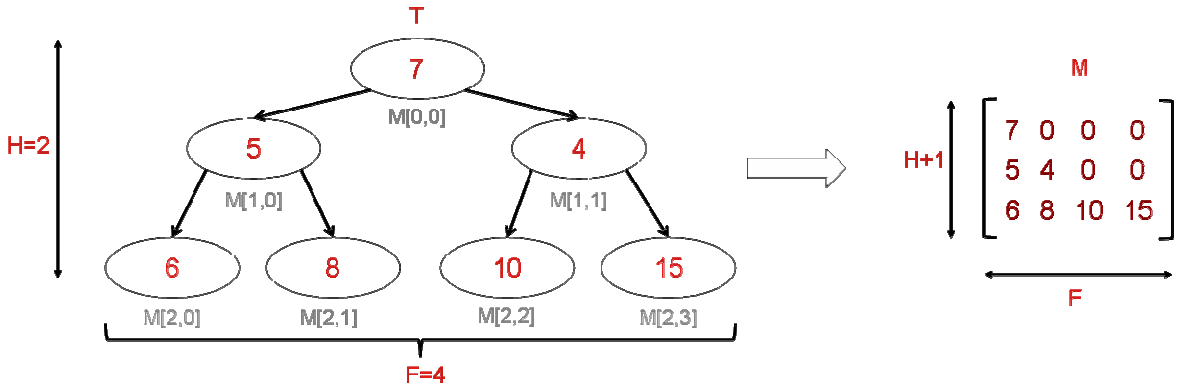
Scrivi lo pseudocodice della procedura NODI_LONTANI(A,*u*,*d*) che prende in input un grafo non orientato A, l'indice *u* di un nodo e una distanza *d* e produce in output il numero dei nodi che sono a distanza maggiore di *d* dal nodo *u*.

Esercizio 3

Discuti la complessità computazionale nel caso peggiore (in termini di O-grande, Omega e Theta) delle procedure che hai proposto per gli esercizi precedenti, utilizzando *v* per denotare il numero dei nodi dell'albero, *n* per denotare il numero dei nodi del grafo ed *m* per gli archi.

CODIFICA C

Si consideri un albero binario i cui nodi contengono un numero intero. Tale albero è completo: ha tutti i nodi possibili fino ad un livello `H`. La **dimensione** dell'albero (il numero di nodi e l'altezza `H`) **non è nota a priori**. A partire dall'albero, si vuole costruire una **matrice rettangolare** (`H+1`) × (`F`) dove `H+1` (il numero di righe) corrisponde all'altezza dell'albero+1, mentre `F` (il numero di colonne) corrisponde al numero di foglie dell'albero. Si consideri l'esempio di seguito



Ogni livello dell'albero corrisponde ad una riga della matrice, considerando i nodi da sinistra verso destra. Gli elementi della matrice che non corrispondono a nodi dell'albero avranno valore `0`. Considerando l'esempio, otterremo dall'albero una matrice (`3 × 4`), in cui la radice `7` corrisponde all'elemento in posizione `[0,0]` nella matrice, l'elemento `5` (livello 1 e primo nodo da sinistra) corrisponde all'elemento in posizione `[1,0]` nella matrice, e così via. L'elemento in posizione `[0,1]` nella matrice ha valore `0`, poiché non corrisponde a nessun nodo nell'albero (non esiste un secondo nodo da sinistra sul livello zero).

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 e la matrice rettangolare, e i file **header** del progetto (**.h**);
2) le funzioni `altezza` e `foglie` che dato un albero binario `T` di interi restituiscano rispettivamente l'altezza `H` di `T` e il numero di foglie `F` di `T`;
3) la funzione `creaMatrice` che dato un albero binario `T` di interi restituisca la matrice rettangolare corrispondente all'albero `T`. Poiché le dimensioni di `T` non sono note a priori, anche le dimensioni della matrice non possono essere definite a priori ma calcolate visitando `T`; si consiglia:
    a) di calcolare prima `H` e `F` e costruire dinamicamente una matrice rettangolare (`H+1`)×(`F`) inizializzando tutti i suoi elementi a `0`, poi
    b) di eseguire una visita di `T` per aggiornare, in base agli elementi di `T`, la matrice rettangolare sopra creata;
4) la funzione `contaDiagonale` che dato un albero binario `T` di interi, e una lista `L` di interi distinti, conti quanti elementi della lista sono presenti nella diagonale principale (caselle con indice di riga e colonna uguali) della matrice rettangolare corrispondente a `T`. Ad esempio dato l'albero binario di esempio e la lista `L: 5 -> 4 -> 15 -> 10`, la funzione restituirà **2**;
5) la funzione `createTextFile` che dato un albero binario `T` di interi, stampi sulla singola riga di un file di testo `"colonne.txt"` tutti gli elementi di ogni colonna della matrice rettangolare corrispondente a `T`. Considerando l'albero binario dell'esempio, sarà creato il file di testo `"colonne.txt"` con il seguente contenuto

```
7 5 6
0 4 8
0 0 10
0 0 15
```

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