A

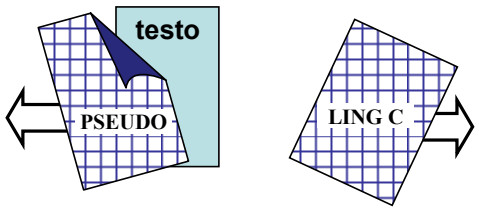
Algoritmi e Strutture di Dati – A.A. 2013-2014
Prova scritta del 18 Giugno 2014
Tempo: D.M. 270 = 2:15h; D.M. 509 = 2:00h

☐ Sono uno studente D.M. 270/04 ☐ Sono uno studente D.M. 509/99

☐ 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 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`**).

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. Assumiamo che se l'albero ha k nodi, questi usino i numeri da 0 a k-1 (in un ordine qualsiasi).

Esercizio 1 (tutti gli studenti)

Scrivi lo pseudocodice della procedura **DEPTH**(**`T`**) che accetti in input un albero di grado arbitrario **`T`** e, senza modificare **`T`**, ritorni un array di interi in cui per ogni nodo (identificato dal suo indice) sia specificata la sua profondità nell'albero. La radice ha profondità zero. *Suggerimento: devi prima contare i nodi dell'albero per poter definire un array della dimensione giusta.*

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

Scrivi lo pseudocodice della procedura **SAME**(**`A`**,**`T`**,***`u`***) che accetti in input un grafo orientato **`A`**, un albero **`T`** e l'indice di un nodo *u* e verifichi che le distanze minime dei nodi di **`A`** da ***`u`*** siano uguali alla profondità in **`T`** dei nodi con lo stesso indice. *Suggerimento: usa **DEPTH**(**`T`**) e scrivi lo pseudocodice della procedura **DISTANCE**(**`A`**,u) che restituisca un array di interi in cui per ogni nodo sia specificata la sua distanza da u.*

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

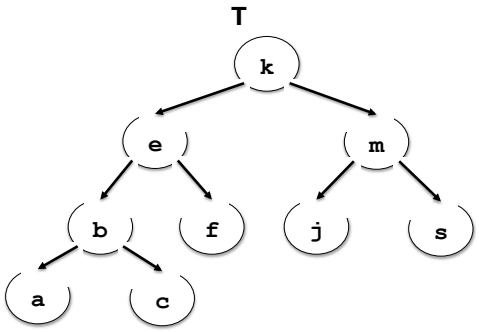
Scrivi lo pseudocodice della procedura **SAME**(**`A`**,**`T`**) che accetti in input un grafo orientato **`A`** e un albero **`T`** e verifichi che il numero degli archi uscenti di ogni nodo di **`A`** sia uguale alla sua profondità in **`T`**. *Suggerimento: usa **DEPTH**(**`T`**) e scrivi lo pseudocodice della procedura **DEGREE**(**`A`**) che restituisca un array di interi in cui per ogni nodo sia specificato il numero degli archi uscenti.*

Esercizio 4

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

LINGUAGGIO C

Si consideri un albero binario di ricerca i cui nodi contengono un carattere (si prendano in considerazione solo le lettere minuscole dell'alfabeto inglese); rispetto ad ogni nodo, il figlio sinistro avrà un valore minore rispetto al padre (è precedente in ordine alfabetico), il figlio destro avrà un valore maggiore rispetto al padre (è successivo in ordine alfabetico). La **dimensione** dell'albero (numero di nodi) **non è nota a priori**. La **distanza** tra due nodi nell'albero è il numero di archi del cammino diretto che connette i due nodi (ogni arco è diretto dal genitore al figlio). Se il cammino non esiste, oppure se uno dei due nodi non è presente nell'albero oppure se entrambi i nodi non sono presenti nell'albero, allora la distanza è 0. Si consideri l'esempio di seguito



la distanza tra il nodo contenente il carattere 'e' e il nodo contenente il carattere 'c' è 2. La distanza tra il nodo contenente il carattere 'e' e il nodo contenente il carattere 'j' è 0.

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 ogni altra struttura necessaria richiesta nei punti successivi, e i file **header** del progetto (**.h**);

2) la funzione **`distanza`** che dato l'albero binario di ricerca **`T`** e due caratteri **`v1`** e **`v2`** calcoli la distanza tra i nodi dell'albero contenenti rispettivamente i caratteri **`v1`** e **`v2`**. Si precisa che bisogna verificare se esiste il cammino dal nodo contenete **`v1`** al nodo contenente **`v2`** oppure il cammino dal nodo contenete **`v2`** al nodo contenente **`v1`**. Ad esempio nell'albero T, **`distanza(T,'c','e')`** restituirà il valore 2.

3) la funzione **`allocaLista`** che dato un albero binario di ricerca **`T`** e un carattere **`v`** ritorni una lista **`L`** di interi (non necessariamente ordinata) con i valori di tutte le distanze tra il nodo contente il carattere **`v`** e le foglie raggiungibili dal nodo. Si consideri l'albero di esempio: **`allocaLista(T,'e')`**ritornerà la lista **`L`** = 2 ➔ 1 ➔ 2

4) la funzione **`ModificaFile`** che accetti come parametri l'albero binario di ricerca **`T`** e il nome di un file binario "**`distanze.dat`**". I record di tale file presentano tre campi **`x, y`** e **`z`**: il campo **`x`** e il campo **`y`** contengono due caratteri mentre il campo **`z`** contiene un valore intero. La funzione **`ModificaFile`** per ogni record del file, deve verificare se il valore del campo **`z`** corrisponda alla distanza tra i nodi dell'albero **`T`** che contengono i caratteri **`x`** e **`y`**. Nel caso in cui il campo **`z`** non contenga il valore corretto, la funzione deve sovrascrivere **`z`** con il valore corretto. Si consideri ad esempio l'albero binario **`T`** sopra riportato e il seguente file:

a	**e**	**2**	**k**	**z**	**1**	**j**	**k**	**1**	**m**	**c**	**0**

Il file verrà modificato come segue

a	**e**	**2**	**k**	**z**	**0**	**j**	**k**	**2**	**m**	**c**	**0**

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