Algoritmi e Strutture di Dati – A.A. 2011-2012
Seconda Prova dell'appello del 14 e 15 febbraio 2012
Libri e appunti chiusi

☐ Studente 270/04 (tempo prova = 2:15 h) ☐ Studente 509/99 (tempo prova = 2:00 h)

☐ Ho bisogno di una correzione veloce in quanto _____

_____

Cognome: _____ Nome: _____ Matricola:_____

PSEUDOCODIFICA

Negli esercizi seguenti supponi che un grafo diretto sia 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*. Un albero binario T ha invece il solo riferimento alla radice T.root (ogni nodo dell'albero ha i campi parent, left, right e key).

Esercizio 1

Scrivi lo pseudocodice della procedura SOLO-RADICE(T) che prende in input un albero T e restituisce **true** se l'albero ha solo la radice, **false** altrimenti.

Esercizio 2

Scrivi lo pseudocodice della procedura CONTA-NODI(T) che prende in input l'albero T e restituisce il numero dei nodi dell'albero T.

Esercizio 3

Scrivi lo pseudocodice della procedura GRAFO-DA-ALBERO(T) che prende in input un albero T (che contiene le chiavi 0, 1, 2,…, n) e restituisce un grafo A. Il grafo A contiene un nodo con indice *k* per ogni nodo dell'albero con chiave *k* e contiene un arco diretto dal nodo *u* al nodo *v* se nell'albero T *u* è parent di *v*. [Se ti serve, quando scrivi questa procedura puoi utilizzare la procedura CONTA-NODI(T) dell'Esercizio 2 o altre procedure di supporto (che però devi descrivere in dettaglio)].

Esercizio 4 (solo studenti D.M. 270/04)

Scrivi lo pseudocodice della procedura VERIFICA-ALBERO(A, *u*) che prende in input il grafo A e verifica che A rappresenti un albero con radice *u* e con gli archi diretti dai nodi genitori verso i nodi figli. [Suggerimento: non ci devono essere archi diretti verso nodi marcati come "raggiunti" durante una visita del grafo].

Esercizio 5

Discuti la complessità computazionale (nel solo caso peggiore) delle procedure che hai proposto per gli esercizi precedenti, utilizzando *n* per denotare il numero totale dei nodi dell'albero o del grafo.

CODIFICA C

Si consideri un insieme di caratteri compresi nel vocabolario di 21 lettere seguenti {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'z'}. Tale insieme viene chiamato *paroliere*. Il paroliere può contenere un numero imprecisato di caratteri e il singolo elemento del vocabolario può ricorrere più di una volta, come ad esempio nel paroliere seguente:

{'a', 'a', 'g', 'o', 'o', 'l', 'l', 'g', 'r', 'r', 'i', 'i', 'i', 't', 't', 'i', 'm', 'm', 'm', 'h', 'h', 'z', 'a'}

Si definisca in linguaggio C:

Esercizio 1

Le strutture dati più adeguate a realizzare le funzionalità degli esercizi seguenti, motivandone anche la scelta.

Esercizio 2

Una funzione che dato il paroliere, conti quante volte posso comporre la parola "algoritmi" con le lettere a disposizione. Nell'esempio del paroliere qui sopra è possibile comporre la parola "algoritmi" due volte.

Esercizio 3

Una funzione ricorsiva che dato il paroliere, crei il file binario “parole.dat” in cui scriva tanti record contenenti ognuno la stringa "algoritmi" quante sono le volte che possiamo comporre la parola "algoritmi" con le lettere a disposizione nel paroliere. E' ammesso scrivere anche una versione non ricorsiva ma in quel caso sarà valutata al massimo metà del punteggio a disposizione per questo punto.