Esercizio 1 1/6/2016

Si tratta dello stesso esercizio della lezione 17 discusso in classe il 31/5. Abbiamo una sequenza di cammini in un array X (sequenze di 0/1 con un -1 finale) e un albero binario e vogliamo aggiungere all'albero dei nodi che corrispondono ai cammini di X. Non tutti i cammini portano all'aggiunta di un nodo, ma solo quelli per cui l'albero non contiene già un nodo alla fine del cammino, ma contiene un nodo che possa fare da padre al nuovo nodo. Un cammino è troppo corto quando il corrispondente nodo è già presente nell'albero, ed è troppo lungo quando il nodo che dovrebbe fungere da padre del nuovo nodo, non c'è nell'albero.

Si tratta di scrivere un programma **iterativo** che consideri tutti i cammini di X e che per ciascun cammino verifichi quale caso si applichi tra i 3 possibili elencati qui di seguito:

i) se il cammino è tale che il nodo non è già presente nell'albero, ma suo padre è presente, allora il nodo viene aggiunto e il suo campo info è il prossimo valore letto da cin;

ii) se il cammino è troppo corto l'albero non cambia e viene stampato su cout:

"cammino troppo corto, non usato:"<<b<<endl; dove b è il prossimo valore letto da cin

iii) se il cammino è troppo lungo l'albero non cambia e viene stampato su cout:

"cammino troppo lungo, non usato:"<<b<<endl; dove b è il prossimo valore letto da cin.

Cosa fare: è richiesto di risolvere il problema con 2 funzioni (entrambe iterative): lavora_iter e cam_iter, dove la prima considera tutti i cammini su X (usa la funzione S dell'esercizio_1_22_3_2016 per conoscere l'indice di inizio in X di un qualsiasi cammino di X) e per ciascun cammino, invoca cam_iter che percorre con un while il cammino e l'albero contemporaneamente in modo da verificare se si è nel caso (i), (ii) o (iii) e fare le azioni richieste in ciascun caso. Le segnature delle 2 funzioni sono:

nodo* lavora_iter(nodo*r, int*X) e nodo* cam_iter(nodo*r, int*X)

Importante: si deve osservare che nel caso (i) sarà necessario avere un puntatore al nodo corrente che diventa 0, ma anche un puntatore al padre, visto che il padre dovrà puntare al nuovo nodo che stiamo per aggiungere, ma non basta, dovremo anche sapere se il nuovo nodo sarà il figlio sinistro o destro di quel padre.

Correttezza:

i) scrivere PRE e POST di lavora_iter e cam_iter. Per questo compito è certamente d'aiuto osservare le PRE e POST delle 2 funzioni ricorsive lavora e cura_cam date nella Lezione 17. E' interessante osservare la differenza tra le PRE e POST delle funzioni e ricorsive e di quelle iterative.

ii) scrivere l'invariante del ciclo della funzione cam_iter

iii) dimostrare la correttezza della funzione cam_iter

Osservare: l'esercizo_1_1_6_2016 del moodle può servire per testare sia la soluzione ricorsiva (vista in classe) che quella iterativa. Basta modificare il main in modo che invochi lavora (ricorsiva) o lavora_iter (iterativa). Attenzione che hanno parametri leggermente divesri. I test sono identici per le 2 soluzioni.