Esame scritto di programmazione 26/1/2015

Dato un albero binario A, si tratta di costruire un array di interi T che contiene una rappresentazione dei cammini completi di A. Diamo subito un esempio.

Esempio. Sia A il seguente albero binario: 1(2(0(_,_),4(_,_)),3(_,0(_,_))), allora vogliamo che T contenga i seguenti cammini:

0 0 -1 0 1 -1 1 1 -1 -2

infatti A contiene 3 cammini completi (cioè da una radice ad una foglia) che sono 0 0 -1, 0 1 -1 e 1 1 -1. Come sempre ogni cammino termina con -1. T contiene -2 dopo l'ultimo cammino, come sentinella che delimita la parte interessante di T. Si deve fare attenzione ai casi limite: se A ha solo la radice, allora T dovrà contenere -1 -2, mentre se A è vuoto, allora T dovrà contenere solo -2.

Parte ricorsiva:

a) scrivere una funzione ricorsiva che, dati 2 alberi binari, verifica che essi abbiano la stessa struttura. Quindi essa non guarda al campo info dei nodi, ma solo alla struttura dei 2 alberi. Per esempio, l'albero 1(2(0(_,_),4(_,_)),3(_,0(_,_))), è strutturalmente uguale a, -3(2(0(_,_),8(_,_)),9(_,0(_,_))).

La funzione ricorsiva richiesta deve soddisfare le seguenti specifiche:

PRE=(a e b sono 2 alberi binari corretti, ciascuno dei quali può essere vuoto)

bool conf(nodo*a, nodo*b)

POST=(risponde true sse a e b sono strutturalmente uguali)

Correttezza induttiva: dimostrare induttivamente che la vostra funzione conf è corretta rispetto a PRE e POST.

b) scrivere una funzione ricorsiva che, dato un albero binario A riempia un array T (fornito come parametro) con i cammini completi di A nel modo illustrato dall'esempio. La funzione deve soddisfare le seguenti specifiche.

PRE=(A è un albero binario, possibilmente vuoto, si deve assumere che T ed S siano lunghi a sufficienza[1], indS>=0, v_T=T)

void c_cam(nodo* A, int*&T, int* S, int indS)

POST=(a partire dalla posizione dell'array puntata da v_T, l'array contiene i cammini completi di A da sinistra a destra ciascuno preceduto da S[0..indS-1], T punta alla posizione dell'array che segue immediatamente i cammini inseriti a partire da v_T)

Attenzione: l'array S e la sua dimensione indS, servono per ricordare i cammini di A percorsi scendendo ricorsivamente verso le foglie di A. Una volta raggiunta una foglia, S[0..indS-1] andrà copiato in T. Osservare che, dopo averlo copiato su T, una parte di S potrà ancora servire perché essa fa parte del cammino per arrivare ad una successiva foglia. Nell'esempio precedente questo succede tra i primi 2

[1] Insomma non si devono inserire controlli sulla lunghezza di T ed S nella funzione c_cam.

cammini: quando si raggiunge la foglia più a sinistra, S conterrà: [0,0,-1], lo si copierà su T, e il primo [**0**] di S servirà per costruire il cammino verso la prossima, cioé, [**0**,1,-1].

E' possibile usare anche qualche (io direi una) funzione ausiliaria, che dovranno anch'esse essere ricorsive.

Parte iterativa: partendo da un array di cammini come quello costruito da c_cam del punto precedente, si deve scrivere una funzione iterativa che costruisca un albero contenente esattamente quei cammini completi. I campi info dei nodi dell'albero prodotto non hanno alcuna importanza e quindi potranno venire inizializzati tutti a 0. La funzione iterativa deve soddisfare le seguenti specifiche.

PRE=(T contiene cammini (possibilmente 0 cammini), ciascuno di essi termina con -1 e subito dopo i cammini, T contiene -2)

nodo* c_alb(int*T)

POST=(restituisce l'albero contenente esattamente i cammini completi contenuti in T, T non viene cambiata).

Correttezza iterativa: specificare un invariante per il ciclo principale della vostra funzione c_alb.

Viene fornito un main con delle funzioni per la costruzione di un albero e per la stampa lineare degli alberi (per fare debugging). L'albero costruito è uguale a quello dell'esempio precedente. Se le vostre c_cam e c_alb saranno giuste, il main, partendo dall'albero dato, dovrebbe arrivare a costruire un secondo albero strutturalmente uguale al primo e quindi la vostra funzione conf dovrebbe restituire true.

C'è un unico test nel quale non c'è alcun input (l'albero da usare è costruito dalla funzione costruz()) e l'output atteso è:

0 0 -1 0 1 -1 1 1 -1 -2

uguali