Esame di Programmazione del 3/9/2019 Parte iterativa

L'esercizio riguarda l'attraversamento di un albero binario in larghezza. Si parte del livello 0 che contiene solo la radice dell'albero, poi il livello 1 che contiene i figli della radice, il livello 2 che contiene i figli dei nodi del livello 1 e così via. Ogni livello è attraversato da sinistra a destra e il nodo più a sinistra è il nodo 1 di quel livello, subito più a destra c'è il nodo 2 e così via.

Per eseguire in modo iterativo l'attraversamento di un albero in larghezza, conviene usare una lista concatenata i cui nodi puntino ai nodi dell'albero. Inizialmente la lista contiene un puntatore alla radice dell'albero. Ad ogni iterazione si deve estrarre dalla lista il primo nodo e aggiungere in fondo alla lista i puntatori ai suoi figli (quelli che ci sono). Viste le operazioni da fare, conviene gestire la lista con una coda. La struct coda, le funzioni di pop e push_end sulla coda e la struct nodoEx dei puntatori ai nodi che sono le componenti della lista gestita dalla coda, sono tutti nel programma dato. Si osservi che la struttura nodoEx contiene anche un campo liv di tipo int che serve a contenere il livello del nodo puntato. Si osservi anche che i figli di un nodo di livello k hanno ovviamente livello k+1.

Ogni nodo di un albero binario è individuato da 2 interi: il livello dell'albero in cui si trova e il suo indice, 1,2,… nel livello stesso.

Esercizio da fare: si chiede di realizzare la funzione **iterativa** perlar (percorso in larghezza) che riceve un albero R e un intero x, e percorre R in larghezza cercando i nodi con info=x e per ciascuno di essi scrive in un array T il livello e l'indice del nodo. Restituisce anche il numero di nodi trovati.

Esempio 1: sia R=[20]([10]([5]([0](_,_),[10](_,_)),[15]([10](_,_),_)),[30]([25]([0](_,_),_),[35]([0](_,_),_))) e sia x= 10. In R ci sono 3 nodi con info=10: il primo è nel livello 1 in posizione 1, mentre gli altri 2 sono entrambi nel livello 3 in posizione 2 e 3.

La funzione iterativa perlar deve soddisfare le seguenti specifiche:

PRE=(albero(R) è un albero binario ben formato, x è definito, T contiene un numero di posizioni pari al doppio del numero dei nodi di albero(R))

void perlar(nodo*R, int x, int*T, int& nt)

POST=(nt è è il numero di nodi con info=x in albero(R) e T contiene il livello e la posizione di ciascuno degli nt nodi nell'ordine del percorso in larghezza)

Esempio 2: riprendendo l'Esempio 1, perlar deve restituire nt=3 e T=[1,1,3,2,3,3]. Se con lo stesso albero R cercassimo x=0, allora la risposta sarebbe: nt=3 e i 3 nodi sarebbero tutti nel livello 3 nelle posizioni 1, 4 e 5. Quindi T=[3,1,3,4,3,5]. Se invece x=4, ovviamente nt=0 e T non conterrebbe nulla. Se x=20, allora nt=1 e T=[0,1].

Correttezza:

i) scrivere un invariante significativo per il ciclo principale di perlar
ii) scrivere la correttezza del ciclo principale di perlar che usi l'invariante di (i)