Algoritmi e Strutture Dati – Prova di Laboratorio

01/02/2018

Istruzioni

Risolvere il seguente esercizio implementando un programma in un singolo file .cpp, completo di funzione *main*. Si presti particolare attenzione alla formattazione dell'input e dell'output, e alla complessità target indicata per ciascuna funzionalità. Nel caso la complessità target non sia specificata, si richiede che sia la migliore possibile. La lettura dell'input e la scrittura dell'output **DEVONO** essere effettuate tramite gli stream **cin** e **cout** rispettivamente. La correzione avverrà prima in maniera automatica inviando il file .cpp al server indicato in aula. Quest'ultimo esegue dei test confrontando l'output prodotto dalla vostra soluzione con l'output atteso. In caso la verifica abbia esito positivo sarà possibile consegnare il compito, il quale verrà valuato dai docenti in termini di complessità. Si ricorda che è possibile testare la correttezza del vostro programma in locale su un sottoinsieme dei input/output utilizzati nella seguente maniera. I file di input e output per i test sono nominati secondo lo schema: `input0.txt output0.txt input1.txt output1.txt ...` Per effettuare le vostre prove potete utilizzare il comando del terminale per la redirezione dell'input. Ad esempio

```
./compilato < input0.txt
```

effettua il test del vostro codice sui dati contenuti nel primo file di input, assumendo che `compilato` contenga la compilazione della vostra soluzione e che si trovi nella vostra home directory. Dovete aspettarvi che l'output coincida con quello contenuto nel file `output0.txt`. Per effettuare un controllo automatico sul primo file input `input0.txt` potete eseguire la sequenza di comandi

```
./compilato < input0.txt | diff - output0.txt
```

Questa esegue la vostra soluzione e controlla le differenze fra l'output prodotto e quello corretto.

Esercizio

[**leggere il testo prestando particolare attenzione alle definizioni**]
Si consideri un sistema di memorizzazione che legga una sequenza di N interi unici e non negativi e li inserisca dentro un albero binario di ricerca (ABR). Siano date le seguenti definizioni:

- una foglia si dice *concorde* se ha etichetta *pari* (*dispari*) ed è figlia di un padre con etichetta *pari* (*dispari*);
- una foglia si dice *discorde* se ha etichetta *pari* (*dispari*) ed è figlia di un padre con etichetta *dispari* (*pari*);
- Per ciascun nodo x, si indica con $nC(x)$ e $nD(x)$ il numero di foglie rispettivamente concordi e discordi del sottoalbero radicato in x. Per una foglia tali valori sono entrambi nulli.

Scrivere un programma che:

- legga da tastiera N etichette e le inserisca all'interno dell'ABR. I valori devono essere inseriti nello stesso ordine con cui vengono letti;
- stampi le etichette dei nodi tali per cui $nC(x) - nD(x) \geq 0$, ordinati per etichetta non decrescente. (complessità al più $\mathcal{O}(n)$)

L'**input** è formattato nel seguente modo: la prima riga contiene l'intero N. Seguono N righe contenenti un'etichetta ciascuna.

L'**output** contiene gli elementi della soluzione, uno per riga.

Esempio

Input

```
6
6
3
2
5
24
1
```

Output

```
1
3
5
6
24
```