Algoritmica – Prova di Laboratorio

Corso A e B

Appello del 13/02/2020

Istruzioni

Risolvete il seguente esercizio prestando particolare attenzione alla formattazione dell'input e dell'output. La correzione avverrà in maniera automatica eseguendo dei tests e confrontando l'output prodotto dalla vostra soluzione con l'output atteso. Si ricorda che è possibile verificare la correttezza del vostro programma su un sottoinsieme dei input/output utilizzati. 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.

Una volta consegnata, la vostra soluzione verrà valutata nel server di consegna utilizzando altri file di test non accessibili. Si ricorda di avvisare i docenti una volta che il server ha verificato la correttezza dell'esecuzione del vostro codice sui dataset forniti, affinché questi possano effettuare il controllo finale sull'aderenza "algoritmica" della soluzione proposta al testo d'esame.

Suggerimenti

Progettare una soluzione efficiente. Prestare attenzione ad eventuali requisiti in tempo e spazio richiesti dall'esercizio. In ogni caso, valutare la complessità della soluzione proposta e accertarsi che sia *ragionevole*: difficilmente una soluzione con complessità $\Theta(n^3)$ sarà accettata se esiste una soluzione semplice ed efficiente in tempo $\mathcal{O}(n)$.

Abilitare i messaggi di diagnostica del compilatore. Compilare il codice usando le opzioni `-g -Wall` di gcc:

```
gcc -Wall -g soluzione.c -o soluzione
```

risolvere *tutti* gli eventuali *warnings* restituiti dal compilatore, in particolar modo quelli relativi alle funzioni che non restituiscono un valore e ad assegnamenti tra puntatori di tipo diverso.

Provare la propria soluzione in locale. Valutare la correttezza della soluzione sulla propria macchina accertandosi che rispetti gli input/output contenuti nel TestSet. In particolare, si consiglia di provare **tutti** gli input/output contenuti nel TestSet usando le istruzioni nella pagina precedente.

Usare `valgrind`. Nel caso in cui il programma termini in modo anomalo o non calcoli la soluzione corretta, è utile accertarsi che non acceda in modo scorretto alla memoria utilizzando `valgrind` (accertarsi di aver compilato il codice con il flag `-g`):

```
valgrind ./soluzione < input0.txt
```

`valgrind` eseguirà il vostro codice sull'input specificato (in questo caso, il file `input0.txt`), mostrando in output dei messaggi di diagnostica nei casi seguenti:

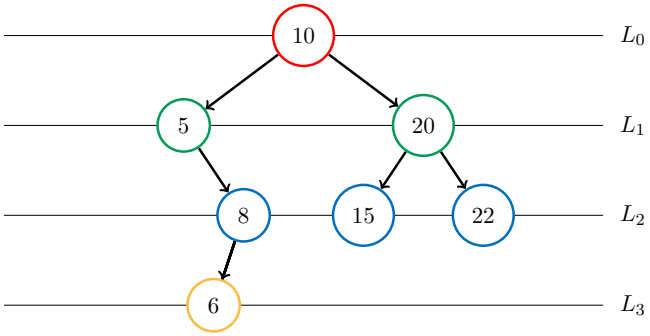
1. accesso (in lettura o scrittura) ad una zona di memoria non precedente allocata;
2. utilizzo di una variabile non inizializzata precedentemente;
3. presenza al termine dell'esecuzione del programma di zone di memoria allocate con `malloc` ma non liberate con `free` (*memory leak*).

Risolvere *tutti* i problemi ai punti 1. e 2. prima di sottoporre la soluzione al server.

Esercizio

Il *livello* k di un albero è dato dall'insieme di tutti i nodi dell'albero ad altezza k. Si ricorda che l'altezza di un nodo u in un albero T radicato in r è definita come la lunghezza del cammino in T da r ad u.

Si consideri ad esempio il seguente albero binario di ricerca (**ABR**, d'ora in poi):



La radice, in colore rosso, è al livello 0, mentre i nodi verdi sono al livello 1, i blu al livello 2 ed i gialli al livello 3.

Il programma, data una sequenza di interi *distinti* ed un livello k, dovrà costruire l'ABR contenente la sequenza di interi in input e stampare i nodi al livello k *in ordine crescente*, una linea per ogni intero.

L'**input** è formattato nel seguente modo. La prima riga contiene l'intero n. Seguono n righe, una per ciascun intero della sequenza S da includere nell'ABR T. Gli interi in S sono distinti. Segue infine una riga contente l'intero k, il livello da stampare. L'intero k può essere maggiore dell'altezza dell'albero.

L'**output** è costituito dagli interi nell'ABR T letto in input al livello k, in ordine crescente, un intero per riga. L'output deve essere vuoto nel caso in cui k sia maggiore dell'altezza dell'albero T.

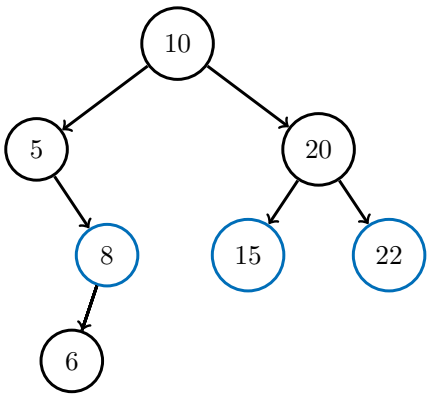
Esempio

Nota: il testo in verde è da intendersi come *commento* e dunque non fa parte dell'input.

Input

```
7  # Interi da inserire nell'ABR
10
5
8
6
20
15
22
2  # Livello k
```

ABR (nodi da stampare in blu)



Output

```
8
15
22
```