Algoritmica – Prova di Laboratorio

Corso A e B

Appello del 25/07/2016

Istruzioni

Risolvete il seguente esercizio prestando particolare attenzione alla formattazione dell'input e dell'output. La correzione avverrà in maniera automatica eseguendo dei test 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 accettato una soluzione come corretta.

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 $\Theta\left(n^3\right)$ 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 **tutti** gli input/output contenuti nel TestSet. Al fine di agevolare la verifica, si consiglia di posizionare la soluzione compilata e i file appartenenti al TestSet nella stessa directory e lanciare sotto tale directory lo script seguente:

```
for i in input* ; do
  ./soluzione < ${i} | diff -q - output${i##input}
done
```

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`:

```
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

Dato un insieme di interi $\mathcal{S}$, un intero $i \in \mathcal{S}$ ha *rango* k se esistono esattamente k interi più piccoli di i in $\mathcal{S}$. Ad esempio, nell'insieme $\mathcal{I} = \{1, 5, 9, 13, 19\}$, l'elemento 9 ha rango 2 (1 e 5 sono più piccoli di 9 in $\mathcal{I}$), mentre l'elemento 1 ha rango 0 (nessun elemento è più piccolo di 1 in $\mathcal{I}$). La *mediana* di un insieme $\mathcal{S}$ con n elementi è dunque definita come l'elemento di S con rango $\lfloor n/2 \rfloor$. Intuitivamente, la mediana è l'elemento "centrale" della lista ottenuta ordinando gli elementi nell'insieme: nell'esempio precedente, la mediana dell'insieme $\mathcal{I}$ è l'elemento 9 in quanto ha rango $\lfloor 5/2 \rfloor = 2$.

L'esercizio richiede di implementare una funzione `mediana` che, dato un albero di ricerca binario, restituisca la mediana dell'insieme degli elementi ivi contenuti.
In particolare, il programma deve leggere una sequenza di $n \geq 1$ interi distinti ed inserirli in un albero binario di ricerca **non bilanciato**.
Al termine dell'inserimento, il programma deve invocare la funzione `mediana` sull'albero costruito al punto precedente e stampare in output il risultato ottenuto.

Nota: Al fine del superamento della prova, la procedura `mediana` dovrà richiedere tempo $\mathcal{O}(n)$, **lineare** rispetto al numero di elementi contenuti nell'albero.

L'input è formattato nel seguente modo. La prima riga contiene il numero n di interi da inserire nell'albero. Seguono poi n righe, una per ogni intero nella sequenza.

L'output è costituito da una singola riga contenente l'intero identificato dalla funzione `mediana`.

Esempi

Nota:

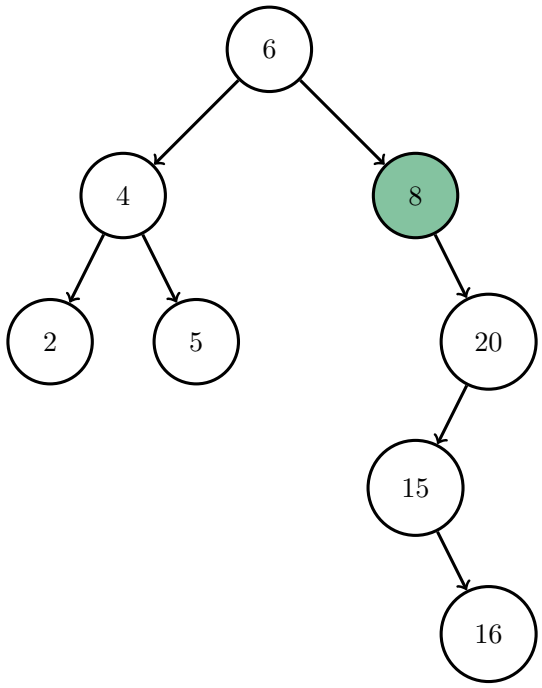
- il testo in verde è da intendersi come *commento* e dunque non fa parte dell'input;
- nell'albero finale, il nodo relativo all'elemento mediano è evidenziato in verde.

Esempio 1

Input

```
8       # N
6       # Primo valore
4       # Secondo valore
2       # etc.
5
8
20
15
16
```

Albero finale



Output

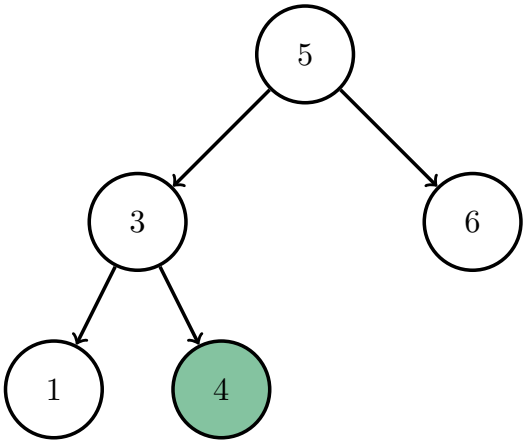
```
8  # 8 ha 4 elementi più piccoli nell'albero
```

Esempio 2

Input

```
5       # N
5       # Primo valore
6       # Secondo valore
3
1
4
```

Albero finale



Output

```
4   # 4 ha 2 elementi più piccoli
```