

Algoritmica – Prova di Laboratorio

Corso A e B

Appello del 01/09/2020

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 redirectione 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 $\Theta(n^3)$ 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 particolare 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

Si implementi un programma che legga da input un grafo indiretto $G = (V, E)$ e determini se G è un albero. Il grafo è rappresentato nel seguente formato: la prima riga contiene il numero $n = |V|$ di nodi, le successive n righe contengono, per ciascun nodo i , il numero n_i di archi uscenti da i seguito da una lista di n_i nodi destinazione, rappresentati con i numeri $[0, n)$. Si assuma che l'input contenga un grafo indiretto, e quindi che per ciascun arco da i a j esista anche l'arco da j a i .

Un albero è un grafo indiretto connesso aciclico. Il programma dovrà stampare 1 se il grafo è un albero, 0 altrimenti.

Input.

- una riga contenente il numero n ;
- n righe, una per ciascun nodo i , con $i \in [0, n)$, nel seguente formato:
 - numero n_i di archi uscenti da i ;
 - lista di n_i nodi di destinazione, rappresentati con i numeri $[0, n)$.

Output.

1 se il grafo è un albero, 0 altrimenti.

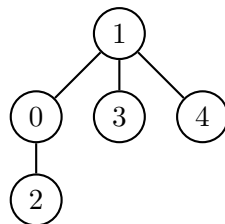
Vincolo.

La soluzione proposta deve avere complessità in tempo di $O(V)$, indipendentemente da $|E|$.

Esempio

Input

```
5           // n
2 1 2
3 0 3 4
1 0
1 1
1 1
```



Output

1