Algoritmi e Strutture Dati – Prova di Laboratorio

16/07/2024

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

Si consideri un sistema stradale in cui un veicolo, partendo da un luogo iniziale I, puó raggiungere piú destinazioni D possibili. Ogni coppia I, D é caratterizzata da un *percorso* composto da un certo numero di tappe intermedie. Una tappa intermedia puó appartenere a piú percorsi. Si definisce *lunghezza* del percorso, il numero di tappe intermedie che lo compongono. Il luogo iniziale, le destinazioni e le tappe intermedie sono dette *nodi* e sono identificati da un intero $ID \geq 0$ e univoco. Il sistema contiene un totale di N nodi.

Il veicolo considerato ha a disposizione un quantitativo di carburante iniziale C per completare un percorso. Ogni nodo attraversato (inclusi luogo iniziale e destinazioni) da un contributo positivo (rifornimento) o negativo (consumo) al carburante del veicolo: tale contributo prende il nome di costo W per quel nodo. La quantitá di carburante e il costo dei nodi sono espressi tramite valori interi. Un percorso si dice *percorribile* se il carburante del veicolo mantiene sempre un valore ≥ 0 su tutto il percorso.

Scrivere un programma che memorizzi il sistema sopra descritto tramite un albero binario di ricerca (ABR) in cui:

- la radice é il punto di partenza;
- ogni foglia é una destinazione;
- tutti gli altri nodi sono tappe intermedie.

Il programma dovrá identificare il percorso *percorribile* piú lungo. A paritá di lunghezza, considerare l'ID della destinazione in ordine non decrescente. Per detto percorso, stampare l'ID del nodo di destinazione e la lunghezza. In caso nessun percorso risulti percorribile, $ID = -1$ e $lunghezza = -1$.

L'**input** è formattato nel seguente modo: la prima riga contiene gli interi N e C. Seguono N righe contenenti le informazioni dei singoli nodi, rappresentati da una coppia ID, W ciascuna.

L'**output** contiene gli elementi della soluzione separati da uno spazio.