Algoritmi e Strutture Dati – Prova di Laboratorio

09/06/2025

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 che memorizza informazioni relative alle missioni (viaggi) del personale universitario. Ogni missione é caratterizzata da un intero *matricola*, da un valore intero *categoria* compreso tra 0 e $C-1$ e un float *spesa*. La *matricola* rappresenta l'identificativo dell'utente; la categoria rappresenta il ruolo dell'utente all'interno dell'università; e la spesa rappresenta il costo totale della missione.

Il sistema memorizza le missioni in una tabella hash, utilizzando la *matricola* come chiave della tabella. Per ogni entrata della tabella, le collisioni sono gestite tramite un Albero Binario di Ricerca (ABR). Gli inserimenti all'interno degli ABR vengono effettuati utilizzando la *matricola* come etichetta.

Per ogni posizione i della tabella hash, il sistema identifica la categoria $MaxC(i)$ che ha la spesa media $s_c(i)$ maggiore. Si definisce spesa media per la categoria c, alla posizione i della tabella hash come segue:

$$s_c(i) = \frac{\text{somma spesa delle missioni con } categoria = c \text{ alla posizione } i}{\text{numero missioni con } categoria = c \text{ alla posizione } i}$$

A parità di $s_c(i)$ si scelga la categoria con valore minore.

Si scriva un programma che

- legga da tastiera N triple $\{intero, intero, float\}$, ciascuna rappresentante rispettivamente, la *matricola*, la *categoria* e la *spesa* per una missione, e le inserisca all'interno della tabella hash all'indirizzo ottenuto usando la seguente funzione

  $$h(x) = \{[(\ a \times x\ ) +\ b\ ]\ \%\ p\ \}\ \%\ (C)$$

  dove p=999149 , a=1000 e b=2000;

- Per ogni posizione i della tabella hash, identifichi la categoria MaxC(i) e stampi le matricole degli utenti appartenenti a tale categoria in ordine di *matricola* non decrescente e separate da uno spazio. Le informazioni relative a diversi ABR vengono stampate su righe diverse.

L'**input** è formattato nel seguente modo: la prima riga contiene gli interi N e C. Seguono N righe contenenti una tripla $\{intero, intero, float\}$ ciascuna.

L'**output** contiene gli elementi della soluzione, una riga diversa per ogni indice della tabella hash.

Esempio

Input

```
10 5
671170 1 2000
681190 1 6000
098765 3 2785
681290 2 30
892348 1 100
896712 0 10
786523 1 1090
623590 1 2000
676170 0 5000
651190 0 3000
```

Output

```
651190 676170 681190
786523
98765
623590
```