Algoritmi e Strutture Dati – Prova di Laboratorio

24/01/2025

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 complessit'a 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 verr'a 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 a docenti ed ai loro corsi. Ciascun docente è caratterizzato da un intero *matricola*, e da una stringa *cognome*. Ciascun corso invece è caratterizzato da un intero *codice* e da un intero *iscritti*. Un corso è associato ad un docente. Corsi di docenti diversi possono avere lo stesso *codice*.

Il sistema memorizza i docenti in una tabella hash con liste di trabocco, utilizzando la *matricola* come chiave. Per ciascun docente nella tabella hash, i suoi corsi sono memorizzati all'interno di un ABR, utilizzando il *codice* come etichetta.

Una volta inseriti tutti i docenti, il sistema permette di inserire delle coppie $\langle matricola, codice \rangle$, ciascuna rappresentante un'iscrizione al corso identificato da *codice*, del docente identificato da *matricola*. Nel caso il corso non sia ancora memorizzato, la prima iscrizione allo stesso ne causa la creazione, inizializzando *iscritti* a 1. Ogni successiva iscrizione incrementa il valore di *iscritti*.

Si scriva un programma che:

- Legga da tastiera N coppie $\langle intero, stringa \rangle$, ciascuna rappresentante rispettivamente la *matricola* e il *cognome* di un docente, e le inserisca all'interno della tabella hash all'indirizzo ottenuto usando la seguente funzione:

$$h(x) = [(a \times x + b) \mod p] \mod (N \times 2)$$

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

- Legga da tastiera M iscrizioni, specificate da coppie $\langle intero, intero \rangle$, ciascuna rappresentante rispettivamente la *matricola* di un docente e il *codice* del corso a cui l'iscrizione si riferisce;

- Stampi il *cognome* del docente del corso con più iscritti. In caso di parimerito, si considerino i docenti in ordine lessicografico per *cognome*.

L'**input** è formattato nel seguente modo: la prima riga contiene gli interi N e M. Seguono N righe contenenti una coppia $\langle intero, stringa \rangle$ ciascuna. Seguono infine M righe contenenti una coppia $\langle intero, intero \rangle$ ciascuna.
L'**output** contiene gli elementi della soluzione, uno per riga.

Esempio

Input

```
3 14
1 Virdis
2 Lemmi
3 Puliafito
1 5
2 10
1 4
1 2
1 2
1 2
3 6
3 1
3 1
2 7
2 9
1 6
1 6
3 1
```

Output

```
Puliafito
```