Algoritmi e Strutture Dati – Prova di Laboratorio

10/01/2019

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 di memorizzazione dei dati relativi agli utenti. Ogni utente è rappresentato da un ID univoco, intero e positivo, e da un'età. Il sistema legge i dati relativi a N utenti e li inserisce dentro una *tabella hash*, utilizzando l'ID come etichetta. La tabella hash è realizzata con il metdodo di concatenazione.

Si definisce I come l'insieme dei primi K indirizzi della tabella hash che hanno generato più collisioni. A parità di numero di collisioni, si considerino gli indirizzi in ordine crescente. Si definisce U come l'insieme degli utenti memorizzati negli indirizzi appartenenti all'insieme I.
Scrivere un programma che:

- legga da tastiera una sequenza di N coppie di interi $\{ID$,età$\}$ e le inserisca nella tabella hash all'indirizzo dato dalla seguente funzione hash:
$$h(x) = \{[(\ a \times x\ ) +\ b\ ]\ \%\ p\ \}\ \%\ (S)$$
dove p=999149 , a=1000 e b=2000;

- Individui l'insieme I. (complessità al più $\mathcal{O}(nlogn)$).

- Stampi a video ID ed età dell'utente più anziano tra quelli appartenenti all'insieme U. A parità di età, si considerino gli utenti in ordine di ID crescente. (complessità al più $\mathcal{O}(n)$).

L'**input** è formattato nel seguente modo: la prima riga contiene gli interi N, K e S separati da uno spazio. Seguono N righe contenenti una coppia di interi ciascuna.

L'**output** contiene gli elementi della soluzione, uno per riga.

Esempio

Input	**Output**

`11 3 7`
`170 88`
`96 68`
`577 79`
`692 99`
`873 75`
`446 88`
`732 79`
`394 78`
`845 58`
`715 9`
`455 11`
	`170`
	`88`