Algoritmi e Strutture Dati – Prova di Laboratorio

24/06/2024

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 di memorizzazione che legga una sequenza di coppie $\{I, S\}$, con I intero non negativo e S stringa, e li inserisca dentro una particolare tabellahash, in cui a ciascun indirizzo è associato un albero binario di ricerca (ABR), i cui nodi conterranno le coppie $\{I, S\}$.
Sia l'inserimento nella tabella hash che l'inserimento in un ABR vengono effettuati usando l'intero I come etichetta.

Per ogni ABR, dato un nodo x si definiscono

- $MaxSx(x)$ come la lunghezza massima tra tutte le stringhe nel sottoalbero sinistro di x;
- $MaxDx(x)$ come la lunghezza massima tra tutte le stringhe nel sottoalbero destro di x;
- $lev(x)$ come il livello di x (per la radice si assume $lev(x) = 0$).

In caso un sottoalbero sia vuoto, il corrispondente valore di $MaxSx$ (o $MaxDx$) sará 0.
Un nodo x si dice *valido* se $MaxSx(x)$ e $MaxDx(x)$ differiscono al piú di $lev(x)$. Infine, per ogni ABR si indica con V il numero di nodi *validi* all'interno dell'albero.

Si scriva un programma che

- legga da tastiera N coppie $\{I, S\}$ e individui l'indirizzo corrispondente utilizzando la seguente funzione hash:

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

  dove p=999149 , a=1000, b=2000 e M intero positivo (specificato in seguito);
- inserisca detta coppia nell'ABR associato all'indirizzo calcolato al punto precedente. I valori devono essere inseriti nello stesso ordine con cui vengono letti (le etichette $\leq$ vanno inserite a sinistra);
- consideri gli indici della tabella hash in ordine crescente, e stampi quelli in cui il valore di V sia maggiore o uguale di K.

L'**input** è formattato nel seguente modo: la prima riga contiene gli interi N, K e M. Seguono N righe contenenti una coppia $intero, stringa$ ciascuna.

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