

Grande Muraglia (muraglia)

Come tutti sanno, la *Grande Muraglia* è un'imponente struttura fortificata formata da N bastioni presidiati da guardie, numerati da 0 a $N - 1$. Poiché si trova in un territorio fortemente montuoso, vi sono grandi differenze di altitudine tra i bastioni e ogni bastione i è caratterizzato dalla sua altitudine H_i , un numero intero positivo.

L'avanzare della tecnologia medioevale nella regione del Catai ha permesso alle guardie che sorvegliano i bastioni di munirsi di droni automatizzati, usati per spiare l'attività di altri bastioni. Infatti, tra guardie di bastioni diversi è presente una grande rivalità, conseguenza del regime altamente meritocratico cui sono sottoposte.

Quando le guardie del bastione x decidono di spiare gli altri bastioni, queste fanno partire due droni dalla sommità del bastione. I droni sono automatizzati e trasmettono immagini di ciò che sorvolano, ma non sono abbastanza sofisticati da permettere alle guardie di controllarli da remoto. In particolare, uno dei due droni viaggia verso sinistra (in direzione del bastione 0) e l'altro verso destra (in direzione del bastione $N - 1$). I droni si mantengono sempre alla stessa altezza, pari a quella del bastione x .


Ovviamente, nel momento in cui un drone raggiunge un bastione y di altezza **strettamente** maggiore di quella di x , il drone si schianta e smette di trasmettere immagini. Pertanto, le guardie del bastione x riusciranno a spiare un intervallo di bastioni (che include x stesso), quello compreso tra i due bastioni più vicini a x che sono più alti di x . Si noti che **anche questi due bastioni vengono spiati**. Inoltre, se tutti i bastioni a sinistra di x sono alti al più quanto x , il drone di sinistra li spierà tutti senza schiantarsi; la stessa cosa vale per i bastioni a destra di x .

La situazione è complicata dalla natura fortemente tellurica della zona: infatti le altitudini a cui i bastioni sono situati cambiano frequentemente. Scrivi dunque un programma che permetta di:

- gestire le variazioni di altitudine dei bastioni;
- rispondere a interrogazioni per sapere quali sono gli estremi dell'intervallo di bastioni che verranno spiati dalle guardie di un bastione x .

Implementazione

Dovrai sottoporre esattamente un file con estensione `.cpp`.

 Tra gli allegati a questo task troverai un template `muraglia.cpp` con un esempio di implementazione.

Dovrai implementare le seguenti funzioni:

C++ | `void inizializza(int N, vector<int> H);`

- L'intero N rappresenta il numero di bastioni.
- L'array H , indicizzato da 0 a $N - 1$, contiene le altezze iniziali dei bastioni.

C++ | `void cambia(int x, int h);`

La funzione viene chiamata quando la nuova altezza del bastione in posizione x diventa h .

```
C++ | pair<int, int> chiedi(int x);
```

La funzione deve restituire il primo e ultimo bastione (in quest'ordine) che vengono spiati se le guardie del bastione x fanno partire dei droni.

Il grader chiamerà prima la funzione `inizializza`, poi chiamerà le funzioni `chiedi` o `cambia` per M volte in totale, stampando in output i valori restituiti dalla funzione `chiedi`.

Grader di prova

Nella directory relativa a questo problema è presente una versione semplificata del grader usato durante la correzione, che potete usare per testare le vostre soluzioni in locale. Il grader di esempio legge i dati da `stdin`, chiama le funzioni che dovete implementare e scrive su `stdout`, secondo il seguente formato.

Il file di input è composto da $M + 2$ righe, contenenti:

- Riga 1: gli interi N e M che rappresentano rispettivamente il numero di bastioni e numero di operazioni.
- Riga 2: N interi che rappresentano le altitudini iniziali H_i dei bastioni della muraglia.
- Righe $3 + i$ ($0 \leq i < M$): il primo carattere determina il tipo di operazione: se è “Q” è seguito da un intero x tra 0 e $N - 1$ che rappresenta il bastione a cui si applica la query; “S” è invece seguito da due interi x e h che rappresentano il bastione e la sua nuova altitudine.

Il file di output è composto da tante righe quante sono le operazioni di tipo “Q”, contenente i valori restituiti dalla funzione `chiedi`.

Assunzioni

- $1 \leq N, M \leq 10^6$.
- $1 \leq H_i \leq 10^9$ per $i = 0, \dots, N - 1$.
- $0 \leq x < N$ per ogni chiamata alla funzione `chiedi`.
- $0 \leq x < N$ e $1 \leq h \leq 10^9$ per ogni chiamata alla funzione `cambia`.

Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [0 punti]**: Casi d'esempio.
- **Subtask 2 [10 punti]**: $N \leq 100$, $M \leq 100\,000$.
- **Subtask 3 [20 punti]**: $H_i \leq 5$ per ogni $i = 0, \dots, N - 1$ e $h \leq 5$ per ogni chiamata alla funzione `cambia`.
- **Subtask 4 [25 punti]**: La funzione `cambia` non viene mai chiamata.
- **Subtask 5 [25 punti]**: $M \leq 100\,000$.
- **Subtask 6 [20 punti]**: Nessuna limitazione aggiuntiva.

Esempi di input/output

stdin	stdout
14 6 7 4 78 2 6 4 1 9 10 7 100 56 8 1 Q 0 Q 4 S 7 3 Q 4 S 2 100 Q 2	0 2 2 7 2 8 0 13
5 3 3 2 4 2 4 Q 1 S 2 2 Q 1	0 2 0 4