

## Make All Equal (equal)

Se visiti il cerchio di pietre preistorico di Stonehenge all'alba del giorno più lungo dell'anno, i druidi ti sfideranno a partecipare all'antico e sacro *gioco dei sassi*. Prima, ti bendano gli occhi, in modo che tu non possa vedere nulla.

Dopo di che, il capo druido ti dirà che ha  $N$  pile di pietre in fila, **dove  $N$  è una potenza di due** ( $N = 2^k$  per qualche intero  $k$ ).

Le pile sono indicizzate da 0 a  $N - 1$ . Ogni pila ha un'altezza  $H_i$ , che è un intero positivo, e le pile sono ordinate da quella più bassa a quella più alta ( $H_0 \leq H_1 \leq \dots \leq H_{N-1}$ ). Il capo druido ti dice il valore di  $N$ , ma non le altezze iniziali.


Puoi decidere di fare delle azioni tra quelle dei seguenti tipi:

1. Scegliere un numero positivo  $X$  e un sottoinsieme delle pile  $S$ . I druidi aggiungeranno  $X$  pietre a ciascuna delle pile in  $S$ , e poi riordineranno le pile da quella più piccola a quella più grande.
2. Scegliere due pile  $i$  e  $j$ . I druidi ti diranno se queste due pile hanno attualmente la stessa altezza.

Il tuo obiettivo è raggiungere una configurazione in cui tutte le pile hanno la stessa altezza. Puoi fare al massimo  $Q_{\text{add}}$  azioni del primo tipo e al massimo  $Q_{\text{compare}}$  azioni del secondo tipo (vedi la sezione di assegnazione del punteggio).

## Implementazione

Dovrai inviare un singolo file sorgente `.cpp`.

 Tra gli allegati di questo problema troverai un template `equal.cpp` con un esempio di implementazione.

Dovrai implementare la seguente funzione:

```
C++ | void make_all_equal(int N, int Q_add, int Q_compare);
```

- L'intero  $N$  rappresenta il numero di pile.
- L'intero  $Q_{\text{add}}$  rappresenta il numero massimo di volte che puoi chiamare `add`.
- L'intero  $Q_{\text{compare}}$  rappresenta il numero massimo di volte che puoi chiamare `compare`.

Puoi chiamare le seguenti funzioni:

```
C++ | void add(vector<int> S, long long X);
```

- Il vettore  $S$  deve contenere interi **distinti** tra 0 e  $N - 1$ , inclusi.
- L'intero  $X$  deve essere compreso tra 0 e  $10^{12}$ , inclusi.
- Questa funzione incrementerà  $H_i$  di  $X$  per ogni  $i$  in  $S$ . Poi, ordinerà le altezze in ordine crescente ( $H_0 \leq \dots \leq H_{N-1}$ ).
- Questa funzione può essere chiamata al massimo  $Q_{\text{add}}$  volte.

```
C++ | bool compare(int i, int j);
```

- $i$  e  $j$  devono essere compresi tra 0 e  $N - 1$ , inclusi.
- La funzione restituirà **true** se  $i$ -esima pila più piccola e la  $j$ -esima pila più piccola hanno la stessa altezza in quel momento (cioè se  $H_i = H_j$ ), e **false** altrimenti.
- Questa funzione può essere chiamata al massimo  $Q_{\text{compare}}$  volte.

## Grader di prova

La cartella del problema contiene una versione semplificata del grader, che puoi usare per testare la tua soluzione in locale. Il grader semplificato legge i dati di input da **stdin**, chiama le funzioni che devi implementare, e scrive l'output su **stdout**.

L'input è formato da due righe, contenenti:

- Riga 1: gli interi  $N$ ,  $Q_{\text{add}}$  e  $Q_{\text{compare}}$ , separati da spazio.
- Riga 2: gli interi  $H_i$ , separati da spazio.

Se il tuo programma è giudicato **Accepted**, il grader di esempio stampa **Accepted: add=U, compare=V** dove  $U$  e  $V$  sono il numero di volte in cui hai chiamato **add** e **compare**.

Se il tuo programma è giudicato **Wrong Answer**, il grader di esempio stampa **Wrong Answer: MSG**, dove **MSG** è uno dei seguenti messaggi:

Messaggio	Significato
too many calls to add	Hai chiamato <b>add</b> più di $Q_{\text{add}}$ volte.
X out of range	L'intero $X$ passato a <b>add</b> non è compreso tra 0 e $10^{12}$ , inclusi.
index in S out of range	Un elemento del vettore $S$ passato a <b>add</b> non è compreso tra 0 e $N - 1$ , inclusi.
indices in S not distinct	Ci sono due elementi uguali nel vettore $S$ passato a <b>add</b> .
too many calls to compare	Hai chiamato <b>compare</b> più di $Q_{\text{compare}}$ volte.
i out of range	L'intero $i$ passato a <b>compare</b> non è compreso tra 0 e $N - 1$ , inclusi.
j out of range	L'intero $j$ passato a <b>compare</b> non è compreso tra 0 e $N - 1$ , inclusi.
heights are not equal	Dopo aver chiamato <b>make_all_equal</b> , ci sono due pile con altezze diverse.

## Assunzioni

- $2 \leq N \leq 2048$ , e  $N$  è una potenza di due.
- Le altezze iniziali soddisfano  $1 \leq H_0 \leq \dots \leq H_{N-1} \leq 1\,000\,000$ .

## Assegnazione del punteggio

Il tuo programma verrà testato su più test case raggruppati in subtask. Per ottenere il punteggio associato a un subtask, devi risolvere correttamente tutti i casi di test che contiene.

- **Subtask 1** [ 0 punti]: Casi d'esempio.
- **Subtask 2** [ 5 punti]:  $N = 2$ ,  $H_i \leq 4$ ,  $Q_{\text{add}} \geq 3000$  e  $Q_{\text{compare}} \geq 4000$ .
- **Subtask 3** [16 punti]:  $N = 2$ ,  $H_i \leq 1\,000\,000$ ,  $Q_{\text{add}} \geq 22$  e  $Q_{\text{compare}} \geq 1$ .
- **Subtask 4** [15 punti]:  $N = 256$ ,  $H_i \leq 10$ ,  $Q_{\text{add}} \geq 3000$  e  $Q_{\text{compare}} \geq 255$ .
- **Subtask 5** [18 punti]:  $N = 4$ ,  $H_i \leq 1\,000\,000$ ,  $Q_{\text{add}} \geq 45$  e  $Q_{\text{compare}} \geq 3$ .

- **Subtask 6** [22 punti]:  $N = 2048$ ,  $H_i \leq 1\,000\,000$ ,  $Q_{\text{add}} \geq 298$  e  $Q_{\text{compare}} \geq 4000$  e **inizialmente, tutte le pile hanno la stessa altezza**. In altre parole,  $H_0 = H_1 = \dots = H_{N-1}$ .
- **Subtask 7** [24 punti]:  $N = 2048$ ,  $H_i \leq 1\,000\,000$ ,  $Q_{\text{add}} \geq 298$  e  $Q_{\text{compare}} \geq 2047$ .

Nota che nessuno di questi subtask contiene tutti i casi di test.

## Esempi di input/output

Input	Esempio di comunicazione		
	Chiamate	Valore	Altezze
4 45 3 1 4 5 5	compare(1, 2)	false	La soluzione chiede se $H_1 = H_2$ . Poiché $H_1 = 4$ e $H_2 = 5$ , la risposta è falsa.
	compare(2, 3)	true	La soluzione chiede se $H_2 = H_3$ . Poiché $H_2 = H_3 = 5$ , la risposta è vera.
	add({0, 1}, 2)		Dopo aver aggiunto le pietre, le nuove altezze sono [3, 6, 5, 5]. Dopo averle riordinate, diventano $H = [3, 5, 5, 6]$ .
	compare(1, 2)	true	La soluzione chiede se $H_1 = H_2$ . Poiché $H_1 = H_2 = 5$ , la risposta è vera.
	add({0, 3}, 3)		Dopo aver aggiunto le pietre, le nuove altezze sono [6, 5, 5, 9]. Dopo averle riordinate, diventano $H = [5, 5, 6, 9]$ .
	add({0, 1}, 4)		Dopo aver aggiunto le pietre, le nuove altezze sono [9, 9, 6, 9]. Dopo averle riordinate, diventano $H = [6, 9, 9, 9]$ .
	add({0}, 3)		Dopo aver aggiunto le pietre, le nuove altezze sono [9, 9, 9, 9]. Dopo averle riordinate, diventano $H = [9, 9, 9, 9]$ .