

Make All Equal (equal)

Als je de prehistorische steencirkel Stonehenge bezoekt op de langste dag van het jaar, zullen druïdes je uitdagen om mee te doen aan een eeuwenoud en heilig kiezel-spel. Allereerst word je geblinddoekt zodat je niks kan zien.

Daarna vertelt de hoofddruïde je dat er N stapels stenen op een rij liggen, **waarbij N een macht van twee is** ($N = 2^k$ voor een integer k).

Stapels zijn genummerd van 0 tot en met $N - 1$. Elke stapel heeft een hoogte H_i , een positief integer, en de stapels zijn geordend van klein naar groot ($H_0 \leq H_1 \leq \dots \leq H_{N-1}$). De hoofddruïde vertelt je de waarde N , maar niet de initiële hoogtes.


Je kan nu een actie kiezen van de volgende types:

1. Kies een positief getal X en een deelverzameling van de stapels S . De druïden zullen X stenen toevoegen aan elk van de stapels in S , waarna ze de stapels herordenen van klein naar groot.
2. Kies twee stapels i en j . De druïden vertellen je of deze twee stapels nu even hoog zijn.

Je doel is om een configuratie te bereiken waarin alle stapels even hoog zijn. Je mag hooguit Q_{add} acties van het eerste type, en hooguit Q_{compare} acties van het tweede type uitvoeren (zie onderdeel Scoring)

Implementatie

Je moet een enkel `.cpp` bestand inleveren.

 In de bijlagen van deze opgave vind je een template `equal.cpp` met een voorbeeld-implementatie.

Je moet de volgende functie implementeren:

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

- Integer N staat voor het aantal stapels.
- Integer Q_{add} staat voor het maximale aantal keer dat je `add` mag aanroepen.
- Integer Q_{compare} staat voor het maximale aantal keer dat je `compare` mag aanroepen.

Je kan de volgende functies aanroepen.

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

- De vector S moet verschillende integers tussen 0 en $N - 1$ (inclusief) bevatten.
- Integer X moet een getal tussen 0 en 10^{12} (inclusief) bevatten.
- Deze functie hoogt H_i op met X voor elke i in S . Daarna worden alle hoogtes gesorteerd in oplopende volgorde ($H_0 \leq \dots \leq H_{N-1}$).
- Deze functie mag hooguit Q_{add} keer worden aangeroepen.

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

- i en j moet tussen 0 en $N - 1$ (inclusief) zijn.

- De functie geeft `true` als de i -de kleinste stapel en de j -de kleinste stapel dezelfde huidige hoogte hebben (dus als $H_i = H_j$), en `false` anders.
- Deze functie mag hooguit Q_{compare} keer worden aangeroepen.

Voorbeeld Grader

De map van de opgave bevat een versimpelde versie van de jury's grader, die je kan gebruiken om je oplossing lokaal te testen. De versimpelde grader leest de invoer van `stdin`, roept de functies aan die je moet implementeren en schrijft uiteindelijk de uitvoer naar `stdout`.

De invoer bestaat uit twee regels:

- Regel 1: de integers N , Q_{add} en Q_{compare} , gescheiden door een spatie.
- Regel 2: de integers H_i , gescheiden door een spatie.

Als je programma wordt beoordeeld met **Accepted**, print de voorbeeldgrader `Accepted: add=U, compare=V` waarbij U en V het aantal keer dat je `add` en `compare` hebt aangeroepen.

Als je program wordt beoordeeld met **Wrong Answer**, print de voorbeeldgrader `Wrong Answer: MSG`, waarbij `MSG` een van de volgende is:

Bericht	Betekenis
too many calls to add	Je hebt <code>add</code> meer dan Q_{add} keer aangeroepen.
X out of range	Het integer X gegeven aan <code>add</code> is niet tussen 0 en 10^{12} (inclusief).
index in S out of range	Een element van de vector S gegeven aan <code>add</code> is niet tussen 0 en $N - 1$ (inclusief).
indices in S not distinct	Er zijn twee gelijke elementen in de vector S gegeven aan <code>add</code> .
too many calls to compare	Je hebt <code>compare</code> meer dan Q_{compare} keer aangeroepen.
i out of range	Het integer i gegeven aan <code>compare</code> is niet tussen 0 en $N - 1$ (inclusief).
j out of range	Het integer j gegeven aan <code>compare</code> is niet tussen 0 en $N - 1$ (inclusief).
heights are not equal	Na het aanroepen van <code>make_all_equal</code> , zijn er twee stapels met verschillende hoogtes.

Randvoorwaarden

- $2 \leq N \leq 2048$, en N is een macht van twee.
- De initiële hoogtes voldoen aan $1 \leq H_0 \leq \dots \leq H_{N-1} \leq 1\,000\,000$.

Scoring

Je programma wordt getest op een set testgevallen gegroepeerd per deelopgave. Om de score te krijgen die hoort bij een deelopgave, moet je alle testgevallen oplossen die daarbij horen.

- **Subtask 1** [0 punten]: Voorbeeld testgevallen.
- **Subtask 2** [5 punten]: $N = 2$, $H_i \leq 4$, $Q_{\text{add}} \geq 3000$ en $Q_{\text{compare}} \geq 4000$.
- **Subtask 3** [16 punten]: $N = 2$, $H_i \leq 1\,000\,000$, $Q_{\text{add}} \geq 22$ en $Q_{\text{compare}} \geq 1$.
- **Subtask 4** [15 punten]: $N = 256$, $H_i \leq 10$, $Q_{\text{add}} \geq 3000$ en $Q_{\text{compare}} \geq 255$.
- **Subtask 5** [18 punten]: $N = 4$, $H_i \leq 1\,000\,000$, $Q_{\text{add}} \geq 45$ en $Q_{\text{compare}} \geq 3$.

- **Subtask 6** [22 punten]: $N = 2048$, $H_i \leq 1\,000\,000$, $Q_{\text{add}} \geq 298$ en $Q_{\text{compare}} \geq 4000$ en **initieel zijn er maar twee verschillende hoogtes**. Oftewel, voor elke i , of $H_0 = H_i$ of $H_i = H_{N-1}$.
- **Subtask 7** [24 punten]: $N = 2048$, $H_i \leq 1\,000\,000$, $Q_{\text{add}} \geq 298$ en $Q_{\text{compare}} \geq 2047$.

Merk op dat geen van deze deelopgaven alle testgevallen bevat.

Voorbeelden

Input	Sample Communication			
	Calls	Return	Heights	Explanation
4 45 3 1 4 5 5	compare(1, 2)	false		De oplossing vraagt of $H_1 = H_2$. Omdat $H_1 = 4$ en $H_2 = 5$ is dit false.
	compare(2, 3)	true		De oplossing vraagt of $H_2 = H_3$. Omdat $H_2 = H_3 = 5$ is dit true.
	add({0, 1}, 2)		3, 5, 5, 6	Na het toevoegen van de stenen zijn de nieuwe hoogtes [3, 6, 5, 5]. Na sorteren wordt dit $H = [3, 5, 5, 6]$.
	compare(1, 2)	true		De oplossing vraagt of $H_1 = H_2$. Omdat $H_1 = H_2 = 5$ is dit true.
	add({0, 3}, 3)		5, 5, 6, 9	Na het toevoegen van de stenen zijn de nieuwe hoogtes [6, 5, 5, 9]. Na sorteren wordt dit $H = [5, 5, 6, 9]$.
	add({0, 1}, 4)		6, 9, 9, 9	Na het toevoegen van de stenen zijn de nieuwe hoogtes [9, 9, 6, 9]. Na sorteren wordt dit $H = [6, 9, 9, 9]$.
	add({0}, 3)		9, 9, 9, 9	Na het toevoegen van de stenen zijn de nieuwe hoogtes [9, 9, 9, 9]. Na sorteren wordt dit $H = [9, 9, 9, 9]$.