Winner Stone

Problem Description

You are given an array of integers **A** of size **N**, where each element represents the weight of a stone. We are playing a game with the stones. On each turn, we choose the two heaviest stones and smash them together.
Suppose the stones have weights **x** and **y** with **x <= y**. The result of this smash is:

If **x == y**, both stones are totally destroyed.
If **x != y**, the stone of weight **x** is totally destroyed, and the stone of weight **y** has new weight **y - x**.

At the end, there is at most one stone left. Return the weight of this stone (or 0 if there are no stones left).

Problem Constraints

1 <= **N** <= 10^4

1 <= **A[i]** <= 10^6

$$A = [\overset{0}{\cancel{3}}\ \overset{1}{\cancel{5}}\ \overset{2}{\cancel{7}}\ \overset{3}{\cancel{1}}\ \overset{4}{\cancel{4}}\ \overset{5}{\cancel{2}}\ \overset{6}{\cancel{8}}\ \overset{7}{\cancel{6}}]$$

$\cancel{1}$ $\cancel{1}$ $\cancel{1}$ $\cancel{1}$

Ans = <u>0</u>

// max heap

```
for i → 0 to (N-1) {
    h.insert(A[i])        // max heap
}

while ( h.size() > 1 ) {
    y = h.getMax()
    x = h.getMax()
    if (x < y) h.insert(y - x)
}

if ( h.size() == 0) return 0
return h.getMax()
```

TC = <u>O(N log(N))</u>

SC = <u>O(N)</u>

House Color

Problem Description

In a small town, there are **N** shops lined up on a street. Each shop is painted with a certain color. You are given an array **A** of length **N**, where **A[i]** represents the color of the **i-th** shop. Return the maximum distance between two shops with different colors. The distance between the **i-th** and **j-th** shops is **abs(i - j)**, where **abs(x)** is the absolute value of **x**.

Note: If no **two** shops have different colours then return **-1**.

Problem Constraints

$2 <= A.length <= 10^6$
$1 <= A[i] <= 10^9$

A = [2 2 2]   Ans = -1

A = [4 4 4 5 8 4 5 6 4 4]   Ans = 7

```
if (A[0] != A[N-1]) return (N-1)
else {
    l = 0
    while (l < N && A[l] == A[N-1]) l++
    if (l == N) return -1
    ans = (N-1) - l
    r = N-1
    while (A[r] == A[0]) r--
    ans = max(ans, r)
}
return ans
```

TC = O(N)   SC = O(1)

Minimize Height

Problem Description

leaves

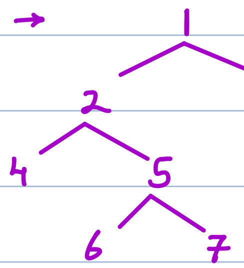
minimise height

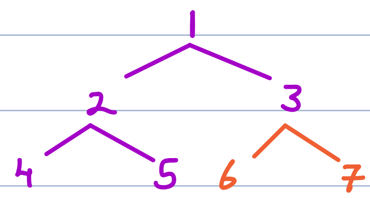
You are given a binary tree with **N** nodes.
You have to minimize the maximum distance between any node and the root of the tree.
In one operation, you can pick a leaf node, and attach it to any other node of the tree. The properties of a binary tree should not be broken at any point of time during the operations. The cost of one operation will be the value of the node being picked. Find the minimum total cost to perform this task.

Problem Constraints

$1 <= N <= 10^5$
$1 <=$ Value of Nodes $<= 10^3$

Eg →





Ans = 6 + 7 = $\underline{13}$

Binary tree with min height → <u>Complete Binary Tree</u>

$\boxed{\text{Height} = \log_2(N)}$

⇒ Any node below this height should move up. ✓

N = 10 (int) $\log_2(10) = \underline{3}$

Any node with level > 3 should move up. ✓

— 0
— 1
— 2
— 3

$\boxed{\log_a b = \frac{\log b}{\log a}}$

```
sum = 0
int minHeight ( root ) {
    N = count (root)
    H = (int) log (N) / log (2)
    travel (root, 0, H)
    return sum
}
```

```
void travel (root, level, H) {
    if (root == null) return
    if (level > H)    sum += root.val
    travel (root.left, level+1, H)
    travel (root.right, level+1, H)
}

int count (root) {
    if (root == null) return 0
    return 1 + count (root.left) + count (root.right)
}
```

TC = $\underline{O(N)}$ SC = $\underline{O(H)}$