6216. Minimum Cost to Make Array Equal

My Submissions (/contest/weekly-contest-316/problems/minimum-cost-to-make-array-equal/submissions/) Back to Contest (/contest/weekly-contest-316/)

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Hard

You are given two **0-indexed** arrays `nums` and `cost` consisting each of `n` **positive** integers.

You can do the following operation **any** number of times:

- Increase or decrease **any** element of the array `nums` by `1`.

The cost of doing one operation on the ith element is `cost[i]`.

Return *the **minimum** total cost such that all the elements of the array* `nums` *become **equal**.*

Example 1:

```
Input: nums = [1,3,5,2], cost = [2,3,1,14]
Output: 8
Explanation: We can make all the elements equal to 2 in the following way:
- Increase the 0th element one time. The cost is 2.
- Decrease the 1st element one time. The cost is 3.
- Decrease the 2nd element three times. The cost is 1 + 1 + 1 = 3.
The total cost is 2 + 3 + 3 = 8.
It can be shown that we cannot make the array equal with a smaller cost.
```

Example 2:

```
Input: nums = [2,2,2,2,2], cost = [4,2,8,1,3]
Output: 0
Explanation: All the elements are already equal, so no operations are needed.
```

Constraints:

- `n == nums.length == cost.length`
- `1 <= n <= 10^5`
- `1 <= nums[i], cost[i] <= 10^6`

Java

```java
class Solution {
        int[] a, b;
    int n;

    public long minCost(int[] A, int[] B) {
        a = A;
        b = B;
        n = a.length;
        int low = Integer.MAX_VALUE, high = Integer.MIN_VALUE;
        for (int x : a) {
            low = Math.min(low, x);
            high = Math.max(high, x);
        }
        while ((high - low) > 2) {
            int mid1 = low + (high - low) / 3;
            int mid2 = high - (high - low) / 3;
            long cost1 = computeCost(mid1), cost2 = computeCost(mid2);
            if (cost1 < cost2) {
                high = mid2;
            } else {
                low = mid1;
            }
        }
        long res = Long.MAX_VALUE;
        for (int v = low; v <= high; v++) res = Math.min(res, computeCost(v));
        return res;
    }

```

```
    long computeCost(int v) {
        long res = 0;
        for (int i = 0; i < n; i++) res += (long) Math.abs(a[i] - v) * b[i];
        return res;
    }
}
```

Custom Testcase Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/828374894/) More Details (/submissions/detail/828374894/)

Share your acceptance!