(/) Explore(/explore/) Problems(/problemset/all/) Interview New (/contest/) Contest Discuss(/discuss/) Store ☆ Premium (/subscribe?ref=nb_npl) 0 (/problems/longest-cycle-in-a-graph/)

6357. Minimum Operations to Make All Array Elements Equal

My Submissions (/contest/weekly-contest-338/problems/minimum-operations-to-make-all-array-elements-equal/submissions/)

Back to Contest (/contest/weekly-contest-338/)

User Accepted:	30
User Tried:	84
Total Accepted:	33
Total Submissions:	97
Difficulty:	Medium

You are given an array `nums` consisting of positive integers.

You are also given an integer array `queries` of size `m`. For the `i`th query, you want to make all of the elements of `nums` equal to `queries[i]`. You can perform the following operation on the array **any** number of times:

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

Return *an array* `answer` *of size* `m` *where* `answer[i]` *is the* ***minimum*** *number of operations to make all elements of* `nums` *equal to* `queries[i]`.

Note that after each query the array is reset to its original state.

Example 1:

```
Input: nums = [3,1,6,8], queries = [1,5]
Output: [14,10]
Explanation: For the first query we can do the following operations:
- Decrease nums[0] 2 times, so that nums = [1,1,6,8].
- Decrease nums[2] 5 times, so that nums = [1,1,1,8].
- Decrease nums[3] 7 times, so that nums = [1,1,1,1].
So the total number of operations for the first query is 2 + 5 + 7 = 14.
For the second query we can do the following operations:
- Increase nums[0] 2 times, so that nums = [5,1,6,8].
- Increase nums[1] 4 times, so that nums = [5,5,6,8].
- Decrease nums[2] 1 time, so that nums = [5,5,5,8].
- Decrease nums[3] 3 times, so that nums = [5,5,5,5].
So the total number of operations for the second query is 2 + 4 + 1 + 3 = 10.
```

Example 2:

```
Input: nums = [2,9,6,3], queries = [10]
Output: [20]
Explanation: We can increase each value in the array to 10. The total number of operations will be 8 + 1 + 4 + 7 = 20.
```

Constraints:

- `n == nums.length`
- `m == queries.length`
- $1 <= n, m <= 10^5$
- $1 <= nums[i], queries[i] <= 10^9$

JavaScript

```javascript
const preSum = (a) => { let pre = [0]; for (let i = 0; i < a.length; i++) { pre.push(pre[i] + a[i]); } return pre; };
const subArraySum = (a, l, r) => a[r + 1] - a[l];

function Bisect() {
    return { insort_right, insort_left, bisect_left, bisect_right }
    function insort_right(a, x, lo = 0, hi = null) {
        lo = bisect_right(a, x, lo, hi);
        a.splice(lo, 0, x);
    }
    function bisect_right(a, x, lo = 0, hi = null) { // > upper_bound
        if (lo < 0) throw new Error('lo must be non-negative');
        if (hi == null) hi = a.length;
        while (lo < hi) {
            let mid = parseInt((lo + hi) / 2);
            a[mid] > x ? hi = mid : lo = mid + 1;
        }
        return lo;
    }
    function insort_left(a, x, lo = 0, hi = null) {
        lo = bisect_left(a, x, lo, hi);
        a.splice(lo, 0, x);
    }
    function bisect_left(a, x, lo = 0, hi = null) { // >= lower_bound
        if (lo < 0) throw new Error('lo must be non-negative');
        if (hi == null) hi = a.length;
        while (lo < hi) {
            let mid = parseInt((lo + hi) / 2);
            a[mid] < x ? lo = mid + 1 : hi = mid;
        }
        return lo;
    }
}

const minOperations = (a, b) => {
    a.sort((x, y) => x - y);
    let res = [], bi = new Bisect(), n = a.length, pre = preSum(a);
    for (const v of b) {
        let l = bi.bisect_right(a, v), cnt = n - l, sum = subArraySum(pre, l, n - 1), moveDown = sum - cnt * v;
        let r = l - 1, cnt2 = r + 1, sum2 = subArraySum(pre, 0, r), moveUp = cnt2 * v - sum2;
        let use = moveDown + moveUp;
        res.push(use);
    }
    return res;
};
```

□ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/922160989/) More Details (/submissions/detail/922160989/)

Share your acceptance!

Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Online Interview (/interview/) | Students (/student) | Terms (/terms) | Privacy Policy (/privacy)

United States (/region)