6077. Sum of Total Strength of Wizards

My Submissions (/contest/weekly-contest-294/problems/sum-of-total-strength-of-wizards/submissions/) Back to Contest (/contest/weekly-contest-294/)

User Accepted:	1
User Tried:	2
Total Accepted:	1
Total Submissions:	3
Difficulty:	Hard

As the ruler of a kingdom, you have an army of wizards at your command.

You are given a **0-indexed** integer array `strength`, where `strength[i]` denotes the strength of the i^{th} wizard. For a **contiguous** group of wizards (i.e. the wizards' strengths form a **subarray** of `strength`), the **total strength** is defined as the **product** of the following two values:

- The strength of the **weakest** wizard in the group.
- The **total** of all the individual strengths of the wizards in the group.

Return *the **sum** of the total strengths of **all** contiguous groups of wizards*. Since the answer may be very large, return it **modulo** $10^9 + 7$.

A **subarray** is a contiguous **non-empty** sequence of elements within an array.

Example 1:

```
Input: strength = [1,3,1,2]
Output: 44
Explanation: The following are all the contiguous groups of wizards:
- [1] from [1,3,1,2] has a total strength of min([1]) * sum([1]) = 1 * 1 = 1
- [3] from [1,3,1,2] has a total strength of min([3]) * sum([3]) = 3 * 3 = 9
- [1] from [1,3,1,2] has a total strength of min([1]) * sum([1]) = 1 * 1 = 1
- [2] from [1,3,1,2] has a total strength of min([2]) * sum([2]) = 2 * 2 = 4
- [1,3] from [1,3,1,2] has a total strength of min([1,3]) * sum([1,3]) = 1 * 4 = 4
- [3,1] from [1,3,1,2] has a total strength of min([3,1]) * sum([3,1]) = 1 * 4 = 4
- [1,2] from [1,3,1,2] has a total strength of min([1,2]) * sum([1,2]) = 1 * 3 = 3
- [1,3,1] from [1,3,1,2] has a total strength of min([1,3,1]) * sum([1,3,1]) = 1 * 5 = 5
- [3,1,2] from [1,3,1,2] has a total strength of min([3,1,2]) * sum([3,1,2]) = 1 * 6 = 6
- [1,3,1,2] from [1,3,1,2] has a total strength of min([1,3,1,2]) * sum([1,3,1,2]) = 1 * 7 = 7
The sum of all the total strengths is 1 + 9 + 1 + 4 + 4 + 4 + 3 + 5 + 6 + 7 = 44.
```

Example 2:

```
Input: strength = [5,4,6]
Output: 213
Explanation: The following are all the contiguous groups of wizards:
- [5] from [5,4,6] has a total strength of min([5]) * sum([5]) = 5 * 5 = 25
- [4] from [5,4,6] has a total strength of min([4]) * sum([4]) = 4 * 4 = 16
- [6] from [5,4,6] has a total strength of min([6]) * sum([6]) = 6 * 6 = 36
- [5,4] from [5,4,6] has a total strength of min([5,4]) * sum([5,4]) = 4 * 9 = 36
- [4,6] from [5,4,6] has a total strength of min([4,6]) * sum([4,6]) = 4 * 10 = 40
- [5,4,6] from [5,4,6] has a total strength of min([5,4,6]) * sum([5,4,6]) = 4 * 15 = 60
The sum of all the total strengths is 25 + 16 + 36 + 36 + 40 + 60 = 213.
```

Constraints:

- `1 <= strength.length <= 10^5`
- `1 <= strength[i] <= 10^9`

JavaScript

```javascript
function DJSet(n) {
    let parent = [], min = [], max = [];
    for (let i = 0; i <= n; i++) {
        parent.push(i);
        min.push(i);
        max.push(i);
    }
    return { find, union, L, R }
    function find(x) {
        return parent[x] == x ? x : parent[x] = find(parent[x]);
    }
```

```
    function union(x, y) {
        x = find(x);
        y = find(y);
        if (x == y) return false;
        if (parent[x] < parent[y]) [x, y] = [y, x];
        parent[x] += parent[y];
        parent[x] = y;
        min[y] = Math.min(min[x], min[y]);
        max[y] = Math.max(max[x], max[y]);
        return true;
    }
    function L() {
        return min;
    }
    function R() {
        return max;
    }
}

const ll = BigInt, mod = 1e9 + 7, bmod = ll(mod);
const totalStrength = (a) => {
    let n = a.length, ds = new DJSet(n), id = Array(n).fill(0), sum = Array(n + 2).fill(0), vis = Array(n +
2).fill(false), res = 0n;
    for (let i = 0; i < n; i++) id[i] = i;
    id.sort((x, y) => a[y] - a[x]);
    for (let i = 2; i <= n + 1; i++) sum[i] = (sum[i - 1] + a[i - 2]) % mod;
    for (let i = 2; i <= n + 1; i++) sum[i] = (sum[i - 1] + sum[i]) % mod;
    for (let p of id) {
        p++;
        if (vis[p - 1]) ds.union(p, p - 1);
        if (vis[p + 1]) ds.union(p, p + 1);
        vis[p] = true;
        let pa = ds.find(p), l = ds.L()[pa], r = ds.R()[pa];
        let lcnt = ll((p - l + 1)) * ll((sum[r + 1] - sum[p]));
        let rcnt = ll((r - p + 1)) * ll((sum[p] - sum[l - 1]));
        lcnt %= bmod;
        rcnt %= bmod;
        res = (res + ll(a[p - 1]) * (lcnt - rcnt)) % bmod;
        res %= bmod;
    }
    res = (res + bmod) % bmod;
    return res;
};
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/704590857/) | More Details > (/submissions/detail/704590857/)

Share your acceptance!

1