5186. Range Frequency Queries

My Submissions (/contest/weekly-contest-268/problems/range-frequency-queries/submissions/) Back to Contest (/contest/weekly-contest-268/)

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

Design a data structure to find the **frequency** of a given value in a given subarray.

The **frequency** of a value in a subarray is the number of occurrences of that value in the subarray.

Implement the `RangeFreqQuery` class:

- `RangeFreqQuery(int[] arr)` Constructs an instance of the class with the given **0-indexed** integer array `arr`.
- `int query(int left, int right, int value)` Returns the **frequency** of `value` in the subarray `arr[left...right]`.

A **subarray** is a contiguous sequence of elements within an array. `arr[left...right]` denotes the subarray that contains the elements of `nums` between indices `left` and `right` (**inclusive**).

Example 1:

```
Input
["RangeFreqQuery", "query", "query"]
[[[12, 33, 4, 56, 22, 2, 34, 33, 22, 12, 34, 56]], [1, 2, 4], [0, 11, 33]]
Output
[null, 1, 2]

Explanation
RangeFreqQuery rangeFreqQuery = new RangeFreqQuery([12, 33, 4, 56, 22, 2, 34, 33, 22, 12, 34, 56]);
rangeFreqQuery.query(1, 2, 4); // return 1. The value 4 occurs 1 time in the subarray [33, 4]
rangeFreqQuery.query(0, 11, 33); // return 2. The value 33 occurs 2 times in the whole array.
```

Constraints:

- `1 <= arr.length <= 10^5`
- `1 <= arr[i], value <= 10^4`
- `0 <= left <= right < arr.length`
- At most `10^5` calls will be made to `query`

JavaScript

```
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);
            x < a[mid] ? 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 counter_value_in_indexA_in = (a_or_s) => { let m = new Map(); let n = a_or_s.length; for (let i = 0; i < n; i++) { if (!m.has(a_or_s[i])) m.set(a_or_s[i], []); m.get(a_or_s[i]).push(i); } return m; };

function RangeFreqQuery(a) {
    let m = counter_value_in_indexA_in(a), memo = new Map(), bi = new Bisect();
    // pr(m);
    return { query }
    function query(l, r, x) {
        if (memo.has(x)) return memo.get(x);
        if (!m.has(x)) return 0;
        let a = m.get(x), len = a.length;
        let min = a[0], max = a[len - 1];
        if (l <= min && r >= max) return len;
        if (r < min || l > max) return 0;
        let lbs = bi.bisect_left(a, l); // lbs >= l
        let ubs = bi.bisect_right(a, r); // ubs <= r
        ubs--;
        // pr(x, a, lbs, ubs);
        return ubs - lbs + 1;
    }
}
```

☐ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/590300431/) More Details > (/submissions/detail/590300431/)

Share your acceptance!