6064. Maximum Consecutive Floors Without Special Floors

My Submissions (/contest/weekly-contest-293/problems/maximum-consecutive-floors-without-special-floors/submissions/)

Back to Contest (/contest/weekly-contest-293/)

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

Alice manages a company and has rented some floors of a building as office space. Alice has decided some of these floors should be **special floors**, used for relaxation only.

You are given two integers `bottom` and `top`, which denote that Alice has rented all the floors from `bottom` to `top` (**inclusive**). You are also given the integer array `special`, where `special[i]` denotes a special floor that Alice has designated for relaxation.

Return *the **maximum** number of consecutive floors without a special floor*.

Example 1:

```
Input: bottom = 2, top = 9, special = [4,6]
Output: 3
Explanation: The following are the ranges (inclusive) of consecutive floors without a special floor:
- (2, 3) with a total amount of 2 floors.
- (5, 5) with a total amount of 1 floor.
- (7, 9) with a total amount of 3 floors.
Therefore, we return the maximum number which is 3 floors.
```

Example 2:

```
Input: bottom = 6, top = 8, special = [7,6,8]
Output: 0
Explanation: Every floor rented is a special floor, so we return 0.
```

Constraints:

- `1 <= special.length <= ` 10^5
- `1 <= bottom <= special[i] <= top <= ` 10^9
- All the values of `special` are **unique**.

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);
            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 maxDis = (a) => {
    let max = Number.MIN_SAFE_INTEGER;
    for (let i = 1; i < a.length; i++) max = Math.max(max, a[i] - a[i - 1] - 1);
    return max;
};

const maxConsecutive = (bottom, top, special) => {
    special.sort((x, y) => x - y);
    let bi = new Bisect(), start = bi.bisect_left(special, bottom), end = bi.bisect_right(special, top) - 1;
    let a = special.slice(start, end + 1);
    a.unshift(bottom - 1);
    a.push(top + 1);
    // pr(start, end, a);
    let res = maxDis(a);
    return res;
};
```

Custom Testcase Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/699704064/) More Details (/submissions/detail/699704064/)

Share your acceptance!

2