6068. Maximum White Tiles Covered by a Carpet

My Submissions (/contest/biweekly-contest-78/problems/maximum-white-tiles-covered-by-a-carpet/submissions/)

Back to Contest (/contest/biweekly-contest-78/)

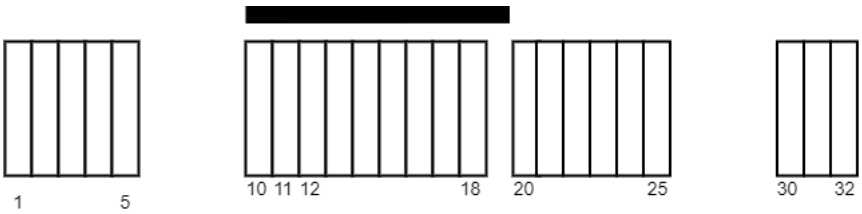
You are given a 2D integer array `tiles` where `tiles[i] = [l_i, r_i]` represents that every tile `j` in the range `l_i <= j <= r_i` is colored white.

You are also given an integer `carpetLen`, the length of a single carpet that can be placed **anywhere**.

Return *the **maximum** number of white tiles that can be covered by the carpet*.

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

Example 1:




```
Input: tiles = [[1,5],[10,11],[12,18],[20,25],[30,32]], carpetLen = 10
Output: 9
Explanation: Place the carpet starting on tile 10.
It covers 9 white tiles, so we return 9.
Note that there may be other places where the carpet covers 9 white tiles.
It can be shown that the carpet cannot cover more than 9 white tiles.
```

Example 2:


```
Input: tiles = [[10,11],[1,1]], carpetLen = 2
Output: 2
Explanation: Place the carpet starting on tile 10.
It covers 2 white tiles, so we return 2.
```

Constraints:

- `1 <= tiles.length <= 5 * 10^4`
- `tiles[i].length == 2`
- `1 <= l_i <= r_i <= 10^9`
- `1 <= carpetLen <= 10^9`
- The `tiles` are **non-overlapping**.

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 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];

const maximumWhiteTiles = (tiles, carpetLen) => {
    tiles.sort((x, y) => x[0] - y[0]);
    let n = tiles.length, a = tiles.map(x => x[0]), dis = tiles.map(e => e[1] - e[0] + 1), bi = new Bisect();
    let pre = preSum(dis), res = Number.MIN_SAFE_INTEGER;
    // pr(tiles)
    // pr(dis)
    for (let i = 0; i < n; i++) {
        let l = tiles[i][0];
        let end = l + carpetLen - 1;
        let idx = bi.bisect_right(a, end) - 1;
        let stop = tiles[idx];
        let lastLen = Math.min(end, stop[1]) - stop[0] + 1;
        // pr("\n", end, stop, lastLen)
        let rangeSum = subArraySum(pre, i, idx - 1);
        let use = rangeSum + lastLen;
        // pr("l", l, "end", end, "r", stop[1], "range", i, idx, "rangeSum", rangeSum, "lastLen", lastLen, "use", use);
        res = Math.max(res, use);
    }
    return res;
};
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/699374347/) | More Details > (/submissions/detail/699374347/)

Share your acceptance!