2817. Minimum Absolute Difference Between Elements With Constraint

My Submissions (/contest/weekly-contest-358/problems/minimum-absolute-difference-between-elements-with-constraint/submissions/)

Back to Contest (/contest/weekly-contest-358/)

You are given a **0-indexed** integer array `nums` and an integer `x`.

Find the **minimum absolute difference** between two elements in the array that are at least `x` indices apart.

In other words, find two indices `i` and `j` such that `abs(i - j) >= x` and `abs(nums[i] - nums[j])` is minimized.

Return *an integer denoting the **minimum** absolute difference between two elements that are at least `x` indices apart*.

User Accepted:	3735
User Tried:	9728
Total Accepted:	3901
Total Submissions:	23827
Difficulty:	Medium

Example 1:

```
Input: nums = [4,3,2,4], x = 2
Output: 0
Explanation: We can select nums[0] = 4 and nums[3] = 4.
They are at least 2 indices apart, and their absolute difference is the minimum, 0.
It can be shown that 0 is the optimal answer.
```

Example 2:

```
Input: nums = [5,3,2,10,15], x = 1
Output: 1
Explanation: We can select nums[1] = 3 and nums[2] = 2.
They are at least 1 index apart, and their absolute difference is the minimum, 1.
It can be shown that 1 is the optimal answer.
```

Example 3:

```
Input: nums = [1,2,3,4], x = 3
Output: 3
Explanation: We can select nums[0] = 1 and nums[3] = 4.
They are at least 3 indices apart, and their absolute difference is the minimum, 3.
It can be shown that 3 is the optimal answer.
```

Constraints:

- `1 <= nums.length <= 10^5`
- `1 <= nums[i] <= 10^9`
- `0 <= x < nums.length`

Discuss (https://leetcode.com/problems/minimum-absolute-difference-between-elements-with-constraint/discuss)

JavaScript

```
function SegmentTreeRMAXQ(n) {
    let h = Math.ceil(Math.log2(n)), len = 2 * 2 ** h, a = Array(len).fill(Number.MIN_SAFE_INTEGER);
    h = 2 ** h;
    return { update, maxx, firstle, tree }
    function update(pos, v) {
        a[h + pos] = v;
        for (let i = parent(h + pos); i >= 1; i = parent(i)) pushup(i);
    }
    function pushup(i) {
        a[i] = Math.max(a[left(i)], a[right(i)]);  // [max .... min]
    }
    function maxx(l, r) { // [L, R)
        let max = Number.MIN_SAFE_INTEGER;
        if (l >= r) return max;
        l += h;
        r += h;
```

```
        for (; l < r; l = parent(l), r = parent(r)) {
            if (l & 1) max = Math.max(max, a[l++]);
            if (r & 1) max = Math.max(max, a[--r]);
        }
        return max;
    }
    function firstle(l, v) {
        if (l >= h) return -1;
        let cur = h + l;
        while (1) {
            if (a[cur] <= v) {
                if (cur >= h) return cur - h;
                cur = left(cur);
            } else {
                cur++;
                if ((cur & cur - 1) == 0) return -1;
                if (cur % 2 == 0) cur = parent(cur);
            }
        }
    }
    function parent(i) {
        return i >> 1;
    }
    function left(i) {
        return 2 * i;
    }
    function right(i) {
        return 2 * i + 1;
    }
    function tree() {
        return a;
    }
}

const minAbsoluteDifference = (a, x) => {
    a = a.map((x, i) => [x, i]).sort((x, y) => x[0] - y[0] || x[1] - y[1]);
    let n = a.length, stmax = new SegmentTreeRMAXQ(n), res = Number.MAX_SAFE_INTEGER;
    for (const [v, i] of a) {
        stmax.update(i, v);
        let l = i - x, r = i + x, maxL, maxR;
        // query [0, l]  [r, n-1]
        if (l >= 0) maxL = stmax.maxx(0, l + 1);
        if (r < n) maxR = stmax.maxx(r, n);
        let d1 = v - maxL, d2 = v - maxR;
        if (!Number.isNaN(d1)) res = Math.min(res, d1);
        if (!Number.isNaN(d2)) res = Math.min(res, d2);
    }
    return res;
};
```