6206. Longest Increasing Subsequence II

My Submissions (/contest/weekly-contest-310/problems/longest-increasing-subsequence-ii/submissions/) Back to Contest (/contest/weekly-contest-310/)

User Accepted:	5
User Tried:	19
Total Accepted:	5
Total Submissions:	22
Difficulty:	Hard

You are given an integer array `nums` and an integer `k`.

Find the longest subsequence of `nums` that meets the following requirements:

- The subsequence is **strictly increasing** and
- The difference between adjacent elements in the subsequence is **at most** `k`.

Return *the length of the **longest subsequence** that meets the requirements.*

A **subsequence** is an array that can be derived from another array by deleting some or no elements without changing the order of the remaining elements.

Example 1:

```
Input: nums = [4,2,1,4,3,4,5,8,15], k = 3
Output: 5
Explanation:
The longest subsequence that meets the requirements is [1,3,4,5,8].
The subsequence has a length of 5, so we return 5.
Note that the subsequence [1,3,4,5,8,15] does not meet the requirements because 15 - 8 = 7 is larger than 3.
```

Example 2:

```
Input: nums = [7,4,5,1,8,12,4,7], k = 5
Output: 4
Explanation:
The longest subsequence that meets the requirements is [4,5,8,12].
The subsequence has a length of 4, so we return 4.
```

Example 3:

```
Input: nums = [1,5], k = 1
Output: 1
Explanation:
The longest subsequence that meets the requirements is [1].
The subsequence has a length of 1, so we return 1.
```

Constraints:

- $1 <= nums.length <= 10^5$
- $1 <= nums[i], k <= 10^5$

JavaScript

```
function SegmentTreeRMQ(n) {
    let h = Math.ceil(Math.log2(n)), len = 2 * 2 ** h, a = Array(len).fill(Number.MAX_SAFE_INTEGER);
    h = 2 ** h;
    return { update, minx, indexOf, tree }
    function update(pos, v) {
        a[h + pos] = v;
        for (let i = parent(h + pos); i >= 1; i = parent(i)) propagate(i);
    }
    function propagate(i) {
        a[i] = Math.min(a[left(i)], a[right(i)]);
    }
    function minx(l, r) {
        let min = Number.MAX_SAFE_INTEGER;
        if (l >= r) return min;
        l += h;
        r += h;
        for (; l < r; l = parent(l), r = parent(r)) {
            if (l & 1) min = Math.min(min, a[l++]);
```

```
            if (r & 1) min = Math.min(min, a[--r]);
        }
        return min;
    }
    function indexOf(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 lengthOfLIS = (a, k) => {
    let max = Math.max(...a), st = new SegmentTreeRMQ(max + 3), res = 0;
    for (const x of a) {
        let l =  Math.max(x-k, 0), r = x;
        let min = st.minx(l, r), maxL = min == Number.MAX_SAFE_INTEGER ? 0 : -min;
        maxL++;
        res = Math.max(res, maxL);
        st.update(x, -maxL);
    }
    return res;
};
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/796859601/) | More Details > (/submissions/detail/796859601/)

Share your acceptance!