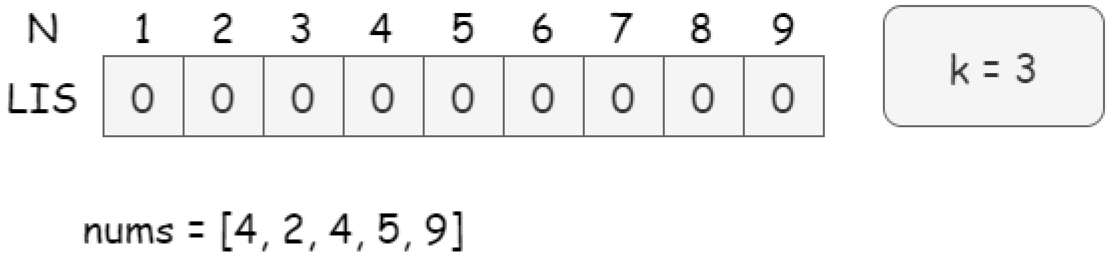
2407. Longest Increasing Subsequence II

Problem Statement: Longest Increasing Subsequence II - LeetCode

Disclaimer: Please try your best before jumping into the solution.

We store the Longest Increasing Subsequence ended by each number in array LIS (**1 – indexed**). Let's say the input **nums = [4, 2, 4, 5, 9].** In other words, indices of LIS represent actual number in input array.

Initially, **LIS[i] = 0** as we haven't added any number yet.



N	1	2	3	4	5	6	7	8	9
LIS	0	0	0	0	0	0	0	0	0

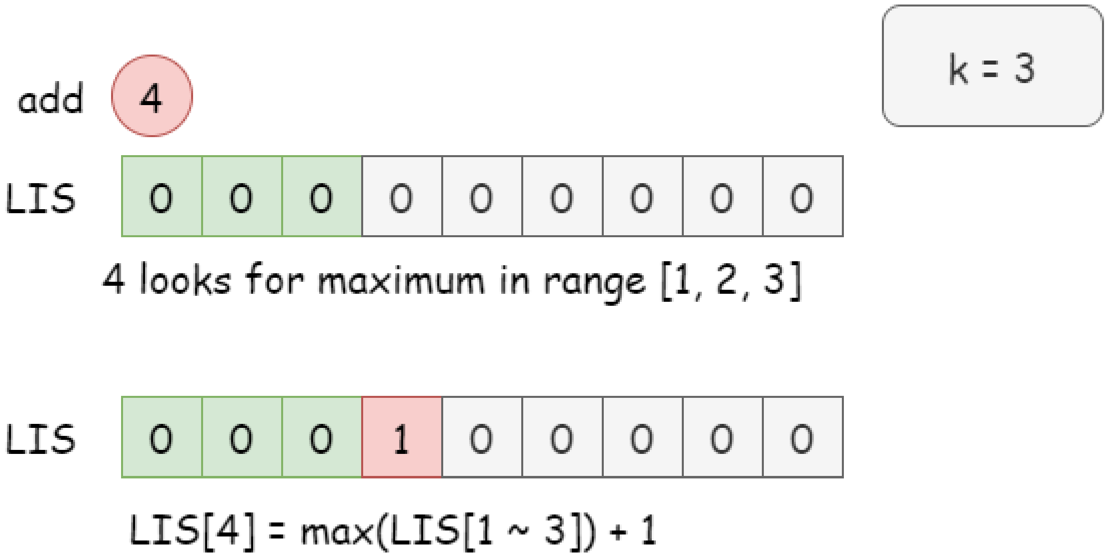
k = 3

nums = [4, 2, 4, 5, 9]

Now, the key is for a given value x, we should find the maximum value from **LIS[a – k…..x-1]**, then **LIS[x] = 1 + max(LIS[a-k….x-1])**. In other words, we will get maximum LIS from previous subsequence that ends at a number **y** which lies in the range **[x-k….x-1]** since it is given that **x – y <= k**

Let's take the above example and see how it works:

- For the first number **4**, the maximum length is the maximum of **LIS[1], LIS[2], LIS[3]** plus **1** (**4** itself). Thus we shall look for the **max(LIS[1…3])**. Apparently, **LIS[4] = 1** which stands for **4** itself.



k = 3

add 4

LIS	0	0	0	0	0	0	0	0	0

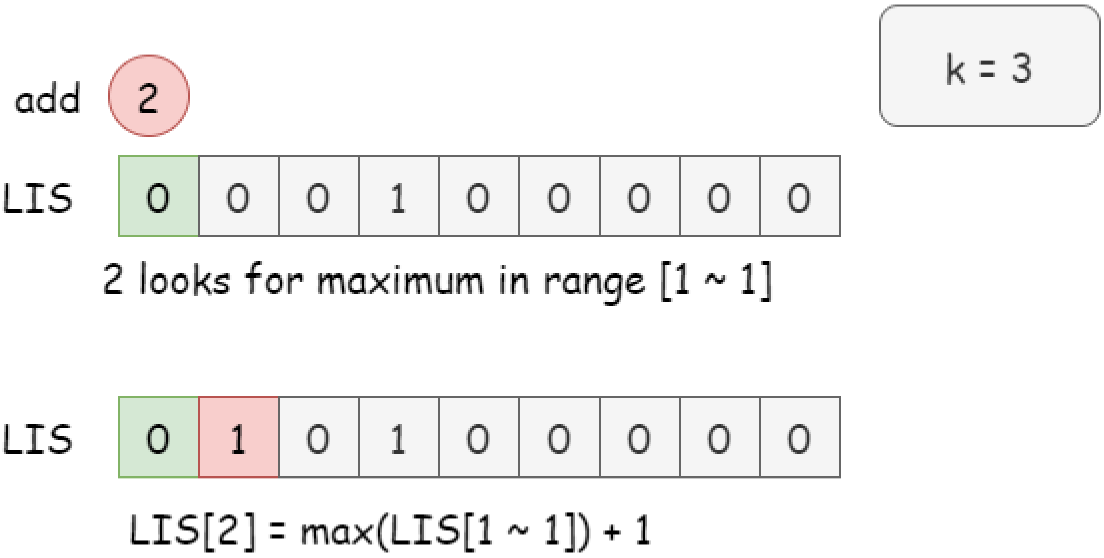
4 looks for maximum in range [1, 2, 3]

LIS	0	0	0	1	0	0	0	0	0

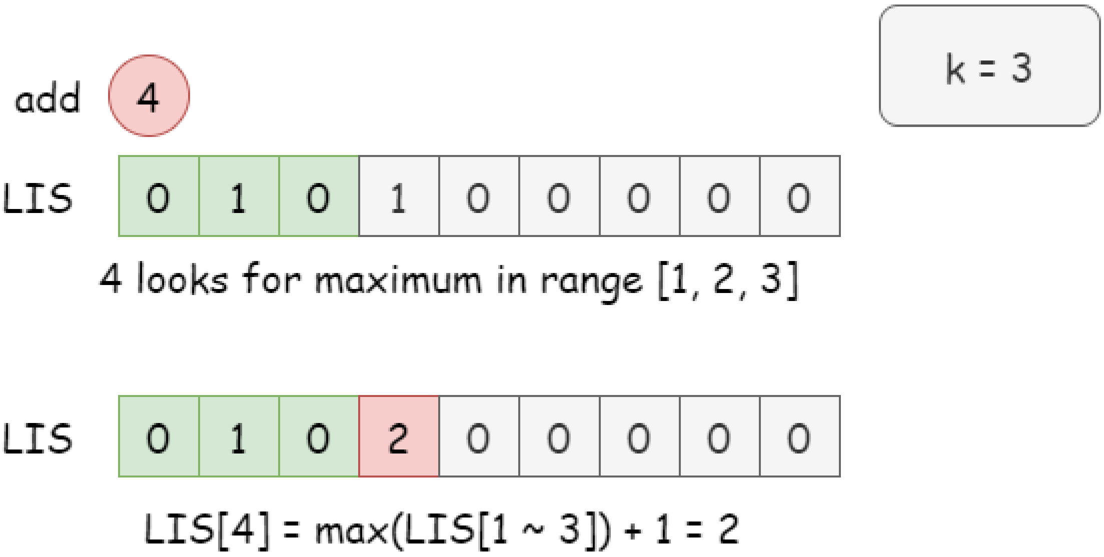
LIS[4] = max(LIS[1 ~ 3]) + 1

Created By: **Saquib Ahsan**

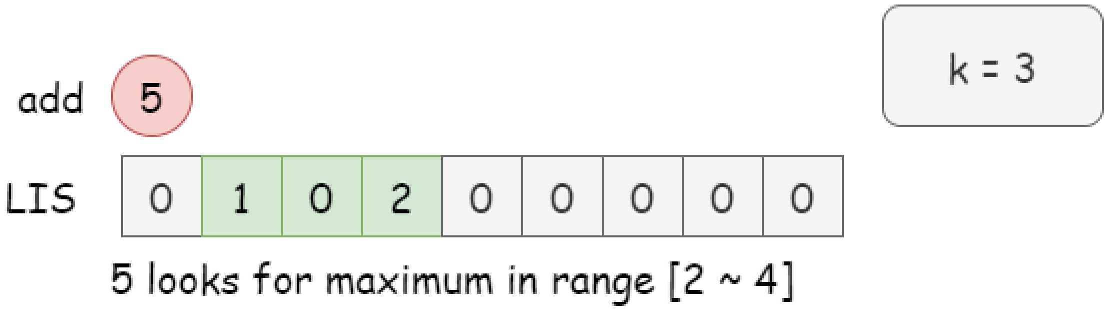
- Then update LIS for **2**, we shall look for **max(LIS[1…1])**, ➔ **LIS[2] = 1 + max(LIS[1…1])**


k = 3
add 2
LIS 0 0 0 1 0 0 0 0 0
2 looks for maximum in range [1 ~ 1]
LIS 0 1 0 1 0 0 0 0 0
LIS[2] = max(LIS[1 ~ 1]) + 1

- Then update LIS for **4**, we look for **max(LIS[1…3])** ➔ **LIS[4] = 1 + max(LIS[1…3]).** Since there is an **2** updated in LIS, thus the maximum value from the same range **LIS[1…3]** gives us **1**. Then we can update **LIS[4] = 2**, implying a subsequence of **2, 4**.

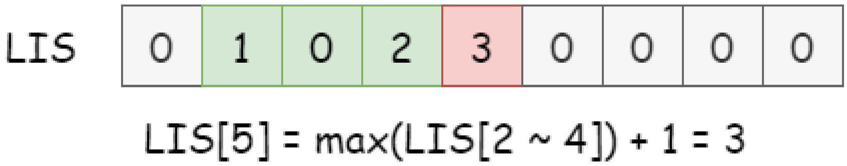

k = 3
add 4
LIS 0 1 0 1 0 0 0 0 0
4 looks for maximum in range [1, 2, 3]
LIS 0 1 0 2 0 0 0 0 0
LIS[4] = max(LIS[1 ~ 3]) + 1 = 2

- Update LIS for **5**, ➔ **LIS[5] = 1 + max(LIS[2…4])** as **k = 3**, implying a subsequence of **2, 4, 5**.



LIS									
LIS	0	1	0	2	0	0	0	0	0

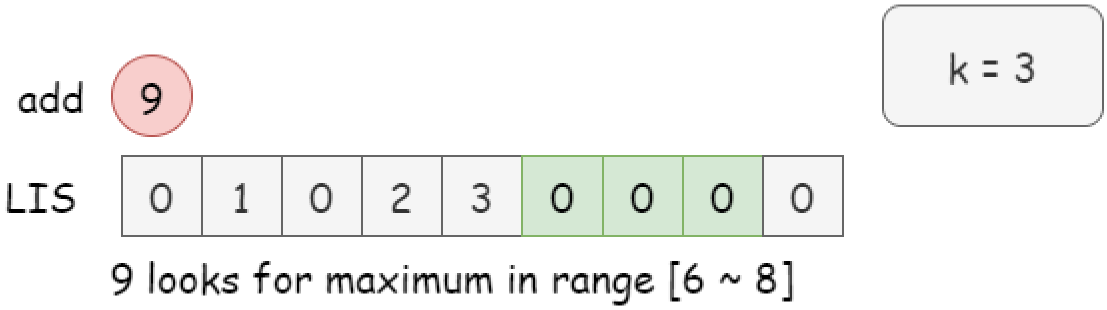
5 looks for maximum in range [2 ~ 4]



LIS									
LIS	0	1	0	2	3	0	0	0	0

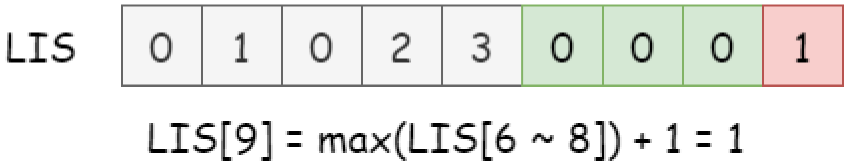
LIS[5] = max(LIS[2 ~ 4]) + 1 = 3

- Then finally, update LIS for **9**, ➔ **LIS[9] = 1 + max(LIS[6…8])**



LIS									
LIS	0	1	0	2	3	0	0	0	0

9 looks for maximum in range [6 ~ 8]



LIS									
LIS	0	1	0	2	3	0	0	0	1

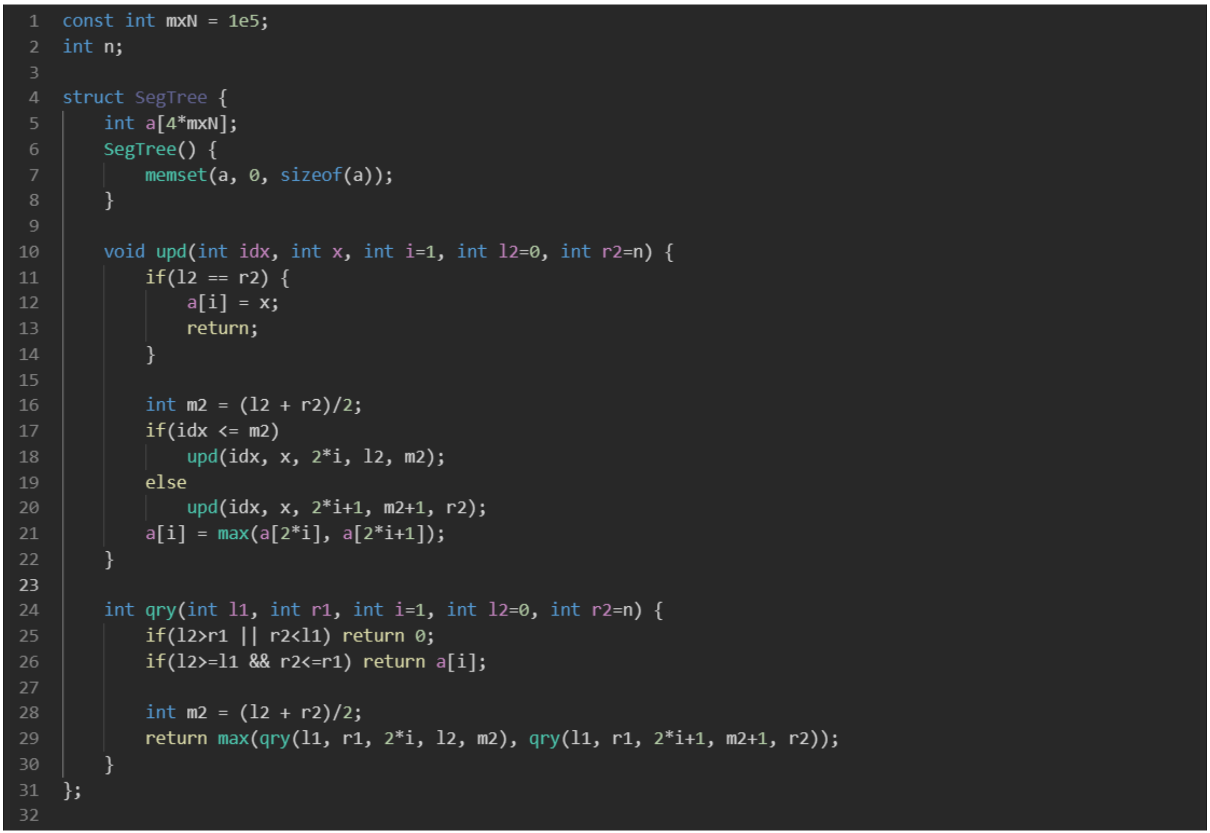
LIS[9] = max(LIS[6 ~ 8]) + 1 = 1

Note: Since we are querying the max value in a range **LIS[x-k…x-1]**, we can use Segment Tree which gives us better complexity.

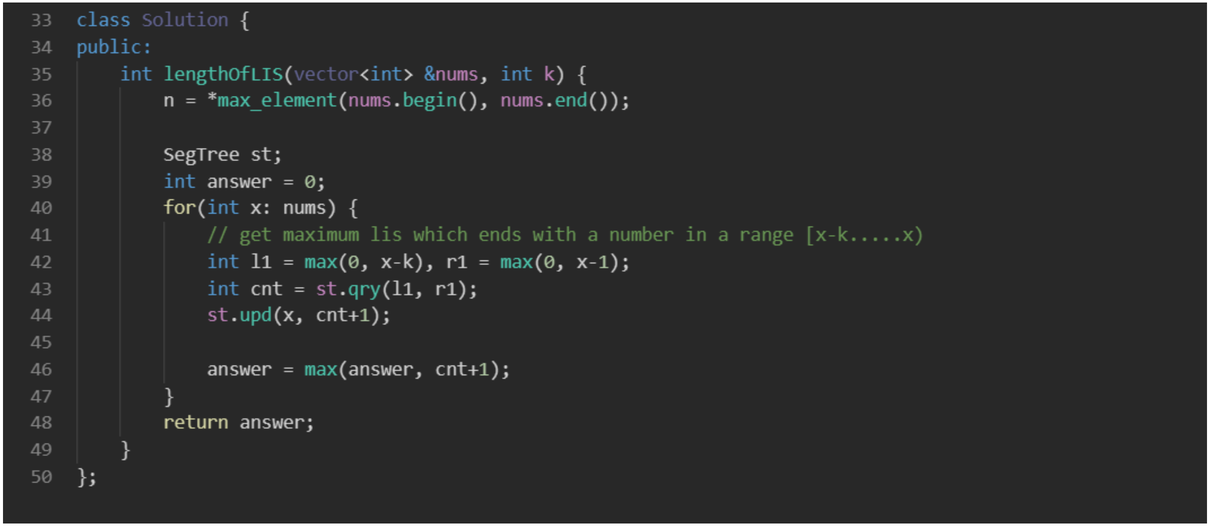
2407. Longest Increasing Subsequence II

Problem Statement: Longest Increasing Subsequence II - LeetCode

Implementation in C++



```cpp
const int mxN = 1e5;
int n;

struct SegTree {
    int a[4*mxN];
    SegTree() {
        memset(a, 0, sizeof(a));
    }

    void upd(int idx, int x, int i=1, int l2=0, int r2=n) {
        if(l2 == r2) {
            a[i] = x;
            return;
        }

        int m2 = (l2 + r2)/2;
        if(idx <= m2)
            upd(idx, x, 2*i, l2, m2);
        else
            upd(idx, x, 2*i+1, m2+1, r2);
        a[i] = max(a[2*i], a[2*i+1]);
    }

    int qry(int l1, int r1, int i=1, int l2=0, int r2=n) {
        if(l2>r1 || r2<l1) return 0;
        if(l2>=l1 && r2<=r1) return a[i];

        int m2 = (l2 + r2)/2;
        return max(qry(l1, r1, 2*i, l2, m2), qry(l1, r1, 2*i+1, m2+1, r2));
    }
};

```



```cpp
class Solution {
public:
    int lengthOfLIS(vector<int> &nums, int k) {
        n = *max_element(nums.begin(), nums.end());

        SegTree st;
        int answer = 0;
        for(int x: nums) {
            // get maximum lis which ends with a number in a range [x-k.....x)
            int l1 = max(0, x-k), r1 = max(0, x-1);
            int cnt = st.qry(l1, r1);
            st.upd(x, cnt+1);

            answer = max(answer, cnt+1);
        }
        return answer;
    }
};
```

Credit: LeetCode

Created By: **Saquib Ahsan**