Segment Tree

Concepts & Qns...

Facebook
Instagram → codestorywithMIK

(Twitter) → CSwithMIK

codestorywithMIK →

"No more fear of Segment Tree"

video - 13

Leetcode - 3477 & 3479

Try this channel to see my "Life behind the Scenes + Tech News"

Motivation :-

If you want to be known, you will have to put effort and go out of your comfort zone.

There is no shortcut to greatness.

If you want it → work for it.

MIK...

3479

3477. Fruits Into Baskets II

Topics | Companies | Hint

You are given two arrays of integers, `fruits` and `baskets`, each of length `n`, where `fruits[i]` represents the **quantity** of the `i^th` type of fruit, and `baskets[j]` represents the **capacity** of the `j^th` basket.

From left to right, place the fruits according to these rules:

- Each fruit type must be placed in the **leftmost available basket** with a capacity **greater than or equal** to the quantity of that fruit type.
- Each basket can hold **only one** type of fruit.
- If a fruit type **cannot be placed** in any basket, it remains **unplaced**.

Return the number of fruit types that remain unplaced after all possible allocations are made.

Example 1:

Input: `fruits = [4,2,5], baskets = [3,5,4]`

Output: 1

Explanation:

- `fruits[0] = 4` is placed in `baskets[1] = 5`.
- `fruits[1] = 2` is placed in `baskets[0] = 3`.
- `fruits[2] = 5` cannot be placed in `baskets[2] = 4`.

Since one fruit type remains unplaced, we return 1.

Example 2:

Input: `fruits = [3,6,1], baskets = [6,4,7]`

Output: 0

Explanation:

- `fruits[0] = 3` is placed in `baskets[0] = 6`.
- `fruits[1] = 6` cannot be placed in `baskets[1] = 4` (insufficient capacity) but can be placed in the next available basket, `baskets[2] = 7`.
- `fruits[2] = 1` is placed in `baskets[1] = 4`.

Since all fruits are successfully placed, we return 0.

Constraints:

- `n == fruits.length == baskets.length`
- `1 <= n <= 100`
- `1 <= fruits[i], baskets[i] <= 1000`

PART-II

Constraints:

- `n == fruits.length == baskets.length`
- `1 <= n <= 10^5`
- `1 <= fruits[i], baskets[i] <= 10^9`

PART-III

Thought Process

Part - II

fruits = [4, 2, 5] (0, 1, 2) i n = 3

baskets = [-1, -1, 4] (0, 1, 2) j

count = 1

T.C = O(n * n)

S.C = O(1)

PART - III

Constraints:

- `n == fruits.length == baskets.length`
- `1 <= n <= 10^5`
- `1 <= fruits[i], baskets[i] <= 10^9`

Optimal Approach...

fruits = [4, 2, 5] (0, 1, 2)

n = 3

baskets = [3, 5, 4] (0, 1, 2)

Why are we SLOW???

fruits = {3, 3, 4, 2, 1, 1} (0, 1, 2, 3, 4, 5)

baskets = {1, 2, 1, 3, 4, 5} (0, 1, 2, 3, 4, 5)

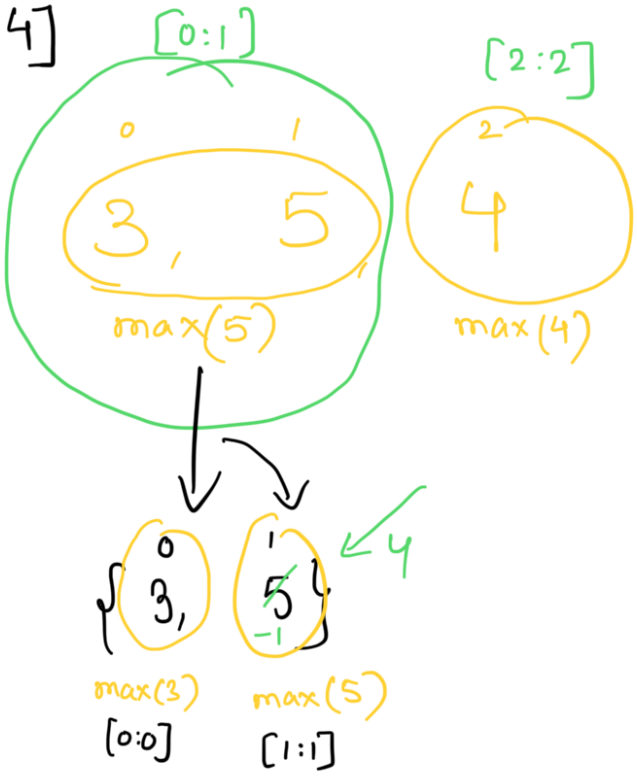
Baskets = [, 2 , ,

max(2) max = 5

Range 0-2 → max = 2 < 3

fruits = [4, 2, 5] n = 3

baskets = [3, 5, 4]



fruit = 4

Range [l : r] → max

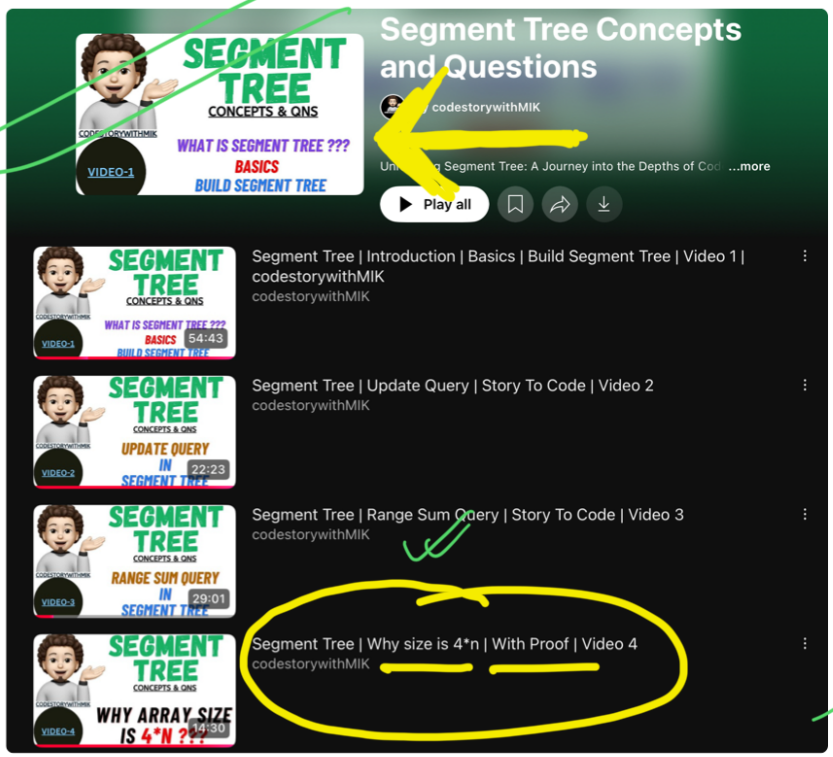
Segment Tree :-

max ↳ segm

Min

Sum

Medium.



fruits [i]

baskets [0 , 1 , 2]

Segment Tree

(Basket) → n

(Range max query).

(•) Build Segment Tree.

(•) vector<int> segTree(4*n, -1);

fruits = [4, 2, 5] n = 3

baskets = [3, 5, 4]

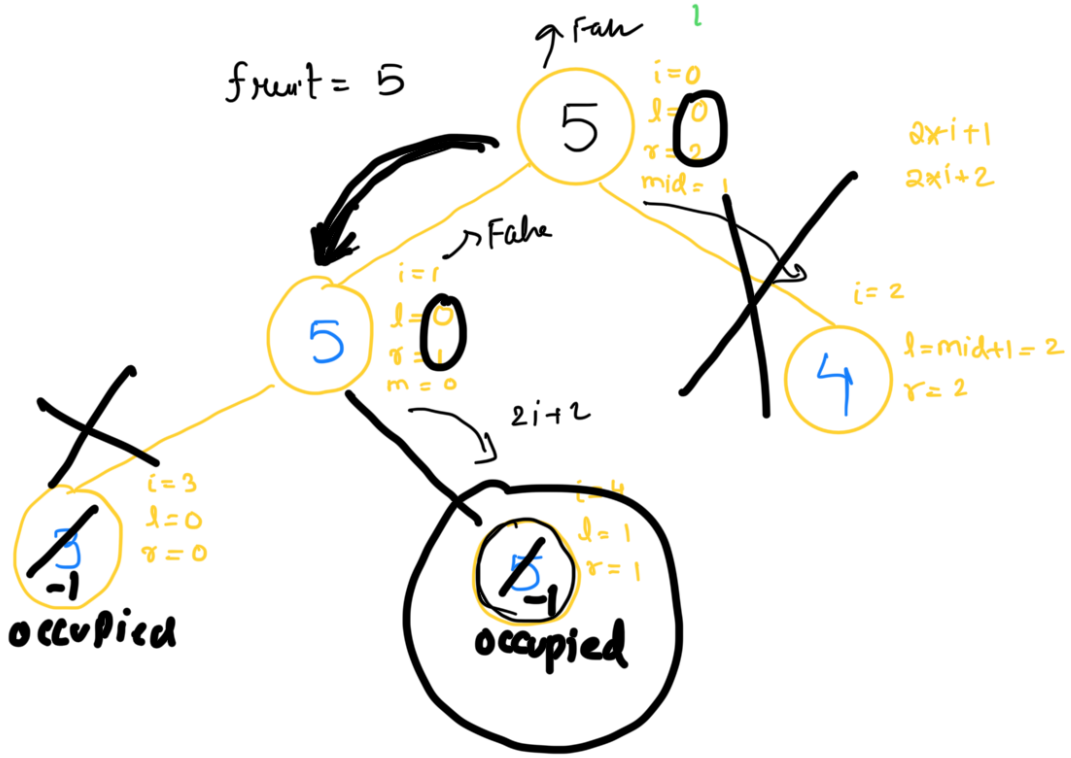
unplaced fruit = 1

segmentTree = | 5 | 5 | 4 | 3 | 5 | .. | .. | .. | .. | .. | .. |

0 1 2 3 4 ... 11

i

fruit = 5



Segment Tree:-

TC = n