DAY54\_STRIVER\_STACK\_QUEUE\_30/10/2024 :

Striver :

# L7. Previous Smaller Element

// nearest smaller element

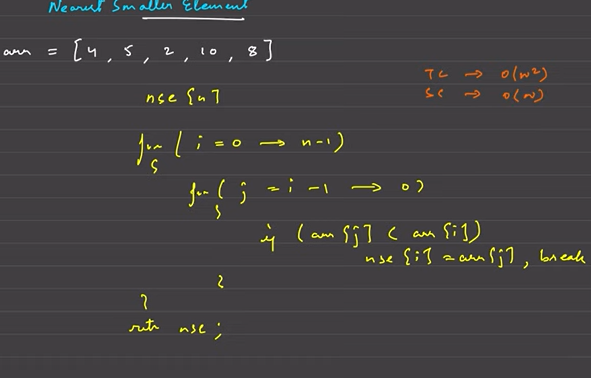
// here we need to find the nearest smaller elements on the left

//

Brutforce :

// start at an elements then u are going to look into the left

// that is it



Better one :

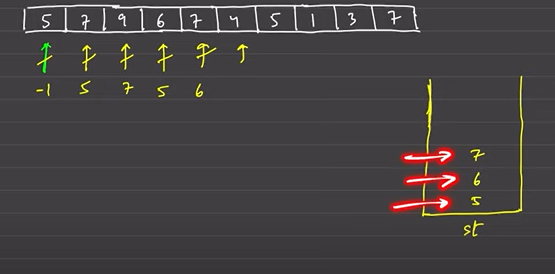
// start from the beginning

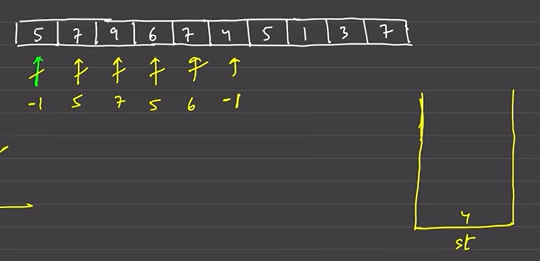
// then u are going to insert the things into the stack

// if u find the small on the stack , then om

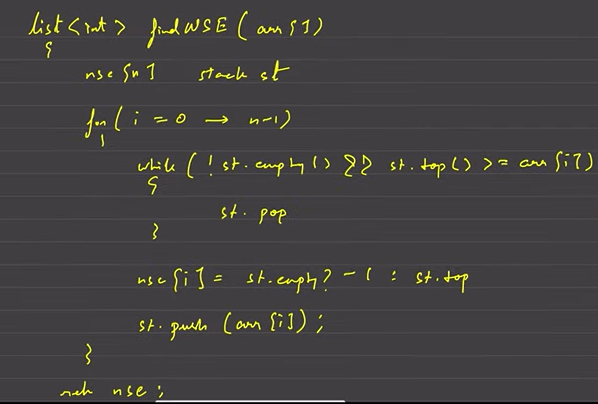
// if not u are going to pop until u find the small element

// do this until u find or until the stack is empty





—------------------



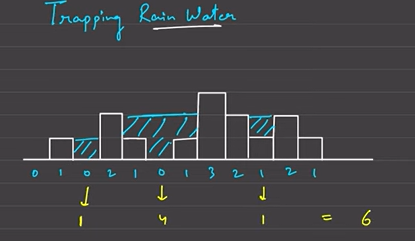
—----------------

# L8. Trapping Rainwater | 2 Approaches



Water will be locked here

// say total units of water that is stored



—--------------------

// we need taller building on left and the right to lock water

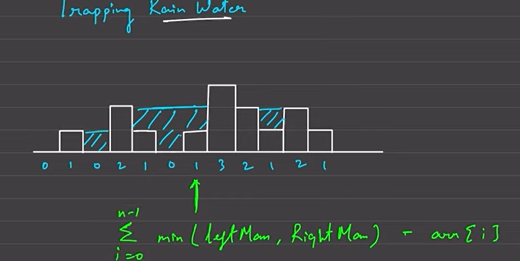
// how to find the water quantity

// on left we are finding the first greater and on the right also

//now the water is filled till the lower of the left one and the right one

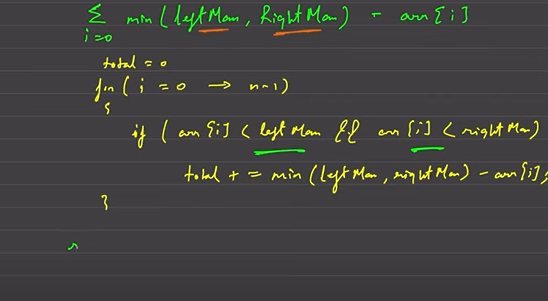
// current building and teh min(taler ones on left and the right)

// that is it



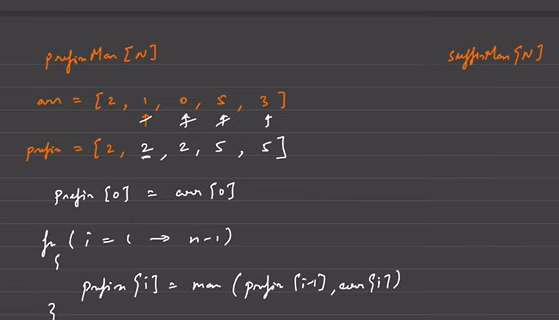
The left and the right should be both greater than current element

Pesudocode :

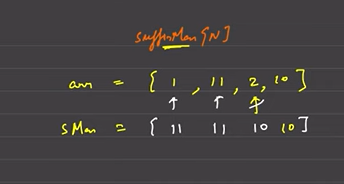




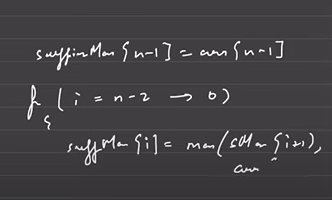
At any position we know the max at the entire left

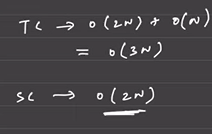


Right max : suffix mam



—--------------





Better one :

U can traverse in a single loop in both directions

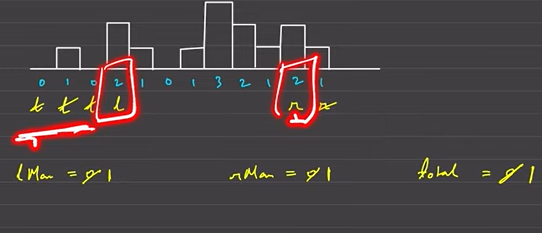
Better one 👍

We need to have the smaller one , either left , or right , therefore we are going to use that

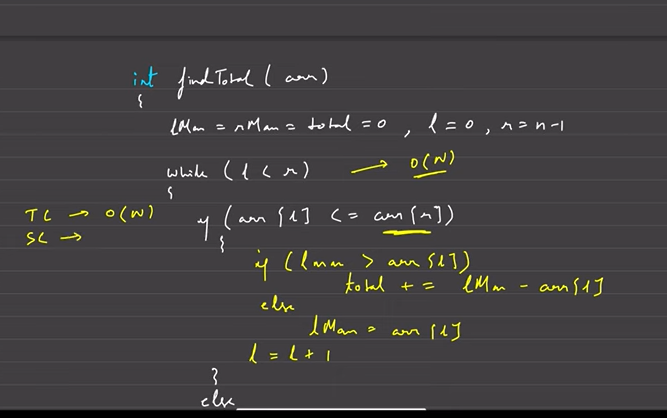
How to find the smaller one without finding both

We need to do 2 way traversal and then compare finding the things

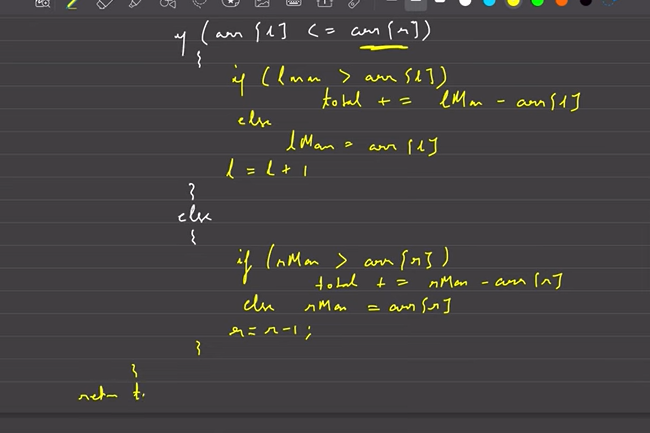
// we are going to move the min pointer



—---------



—------------



CODE :

class Solution {

public:

int trap(vector<int>& height) {

int n = height.size();

if (n == 0) return 0; // Edge case: no bars

int l = 0, r = n - 1;

int lmax = 0, rmax = 0;

int total = 0;

while (l <= r) {

if (height[l] <= height[r]) {

if (height[l] >= lmax) {

lmax = height[l];

} else {

total += lmax - height[l];

}

l++;

} else {

if (height[r] >= rmax) {

rmax = height[r];

} else {

total += rmax - height[r];

}

r--;

}

}

return total;

}

};