Stack-3

Saturday, 27 July 2024
12:21PM

Sum of Subarray Minimums

$$\text{int[] arr} = \{ \underset{0}{3}, \underset{1}{4}, \underset{2}{2}, \underset{3}{5}, \underset{4}{6}, \underset{5}{7}, \underset{6}{1} \}$$

{3} → 3
{3,4} → 3
{3,4,2} → 2
{3,4,2,5} → 2
{3,4,2,5,6} → 2
{3,4,2,5,6,7} → 2
{3,4,2,5,6,7,1} → 1

{4} → 4
{4,2} → 2
{4,2,5} → 2
{4,2,5,6} → 2
{4,2,5,6,7} → 2
{4,2,5,6,7,1} → 1

Sum += 2

contest
{ 27/07 → 6PM
28/07 → 1PM }
28/07✓ → 2-5??
↪ ✓ Doubt

$$\sum_i \text{minVal}$$

Brute Force
↪ Using 2 loops generate all possible subarrays?

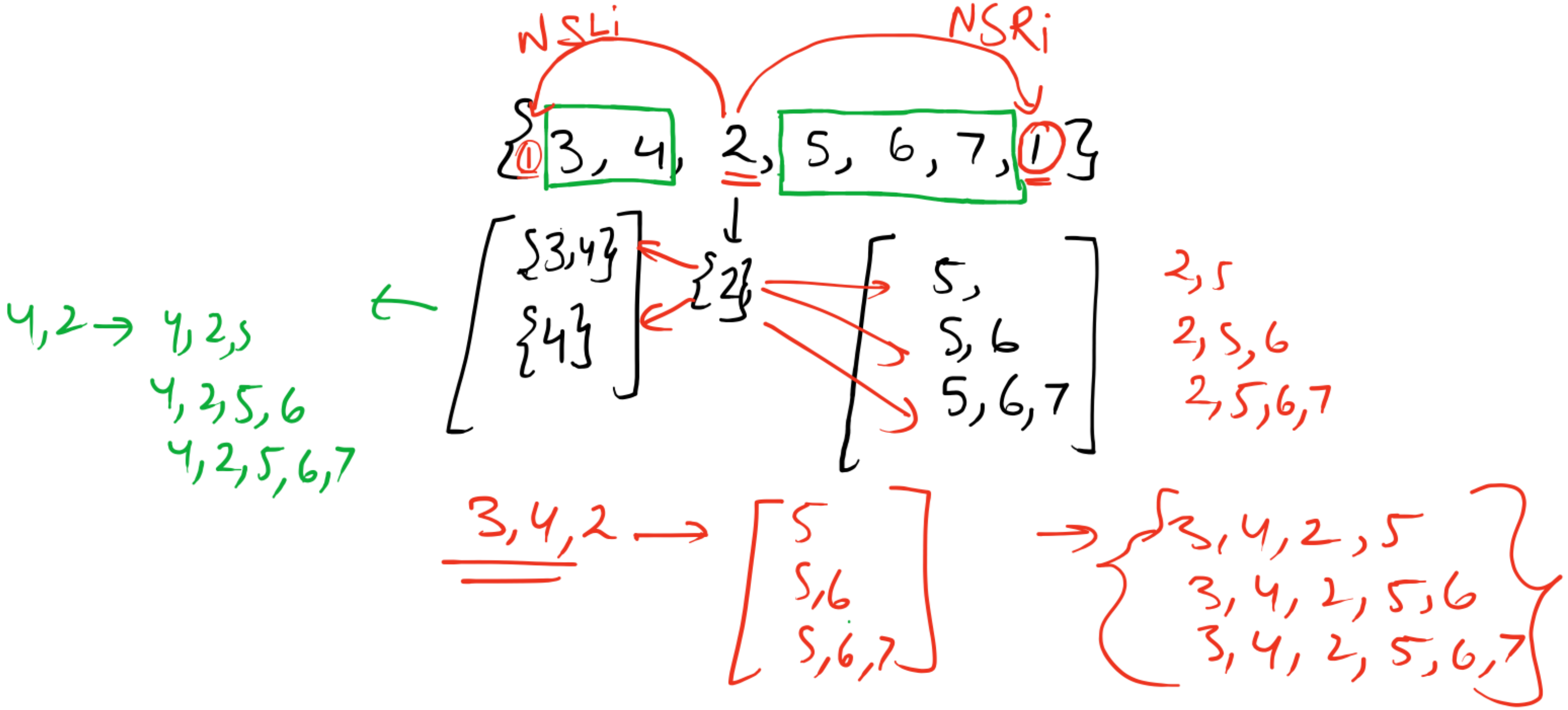
```
for(si){
    for(ei=si→n) ✓
        ↪ calculate the min
}
```

TC: $O(N^2)$
SC: $O(1)$

{3, 4, 2, 5, 6, 7, 1}

$$\text{Sum} = \underbrace{3 \times a}_{\{\text{No. of subarrays where 3 is minimum}\}} + \underbrace{4 \times b}_{\text{No SA where 4 is min}} + 2 \times c + 5 \times d + 6 \times e + 7 \times f + 1 \times g$$

Sum = element × { No of sub arrays where element will be minimum }



Total no. of sub: 2 + 1 + 3 + 2×1×3

3,4,2 2 2,5,6

4,2 2,5,6,7

$$2+1+3+6 = \underline{\underline{12}}$$

Total no. of subarray where ele is min $= 1 + l + r + l \times 1 \times r$ $= (1+l)(1+r)$

$l + (r+1) + lr$

$= l(\cancel{(1+r)}+(1+r)$

$= (1+l)(1+r)$

↳ $l \Rightarrow$ no. of greater elements on left

↳ $r \Rightarrow$ no. of greater elements on right

$$\left\{\begin{array}{l} \underline{\underline{l}} = idx - \underline{NSEL_i} - 1 \\ r = \underline{NSER_i} - idx - 1 \end{array}\right\} \checkmark$$

TC: O(n)

SC: O(n)

$$\{3, 4, 2, 5, \underline{6}, \boxed{7}, 1\}$$

6

6,7

$l=0$

$r=1$

$= (1+0)(1+1)$ ✓

$=2$

?? { NSELi ⇒ this is needed for
{ NSERi   every element.

⇓

Stack can be used here.

$\{\underset{0}{3}, \underset{1}{4}, \underset{2}{2}, \underset{3}{5}, \underset{4}{6}, \underset{5}{7}, \underset{6}{1}\}$ ①

↳ Stack will store smaller elements only

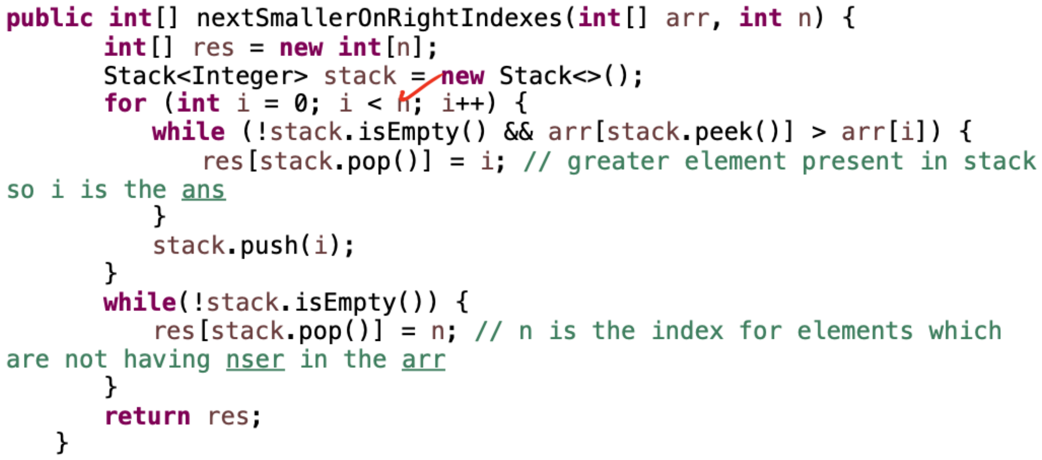
NSEL = $\{-1, \underset{0}{3}, -1, \underset{2}{2}, \underset{3}{5}, \underset{4}{6}, -1\}$

↳ remove bigger elements from stack
→ Peek

$\{\underset{0}{3}, \underset{1}{4}, \underset{2}{2}, \underset{3}{5}, \underset{4}{6}, \underset{5}{7}, \underset{6}{1}\}$ n

NSER

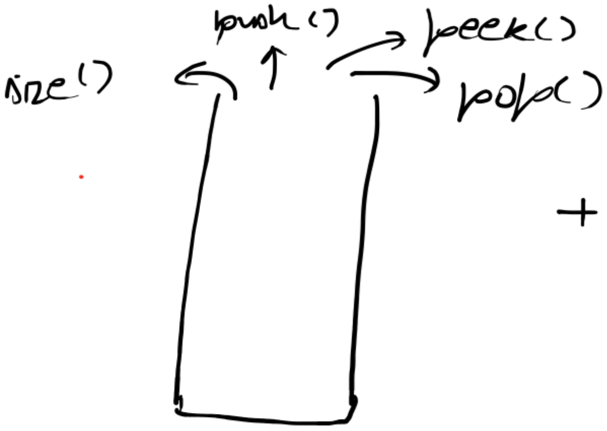
2	2	6	6	6	6	n
0	1	2	3	4	5	6



```
public int[] nextSmallerOnRightIndexes(int[] arr, int n) {
        int[] res = new int[n];
        Stack<Integer> stack = new Stack<>();
        for (int i = 0; i < n; i++) {
            while (!stack.isEmpty() && arr[stack.peek()] > arr[i]) {
                res[stack.pop()] = i; // greater element present in stack
so i is the ans
            }
            stack.push(i);
        }
        while(!stack.isEmpty()) {
            res[stack.pop()] = n; // n is the index for elements which
are not having nser in the arr
        }
        return res;
    }
```

↳ Stack will store the elements for which ele is the NSER

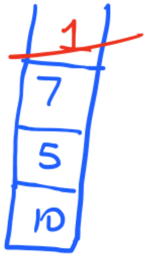
Minimum Stack



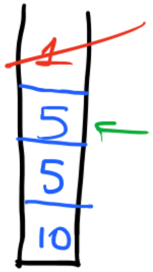
+ getMin()
↳ returns the min present in the stack
TC: O(1)

push(10)

1) Use 2 Stacks



Stack



Aux Stack
(Store Min)

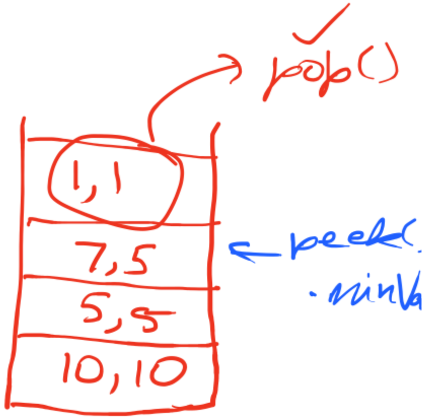
getMin ← Aux.peek() ✓

TC for each op: O(1)

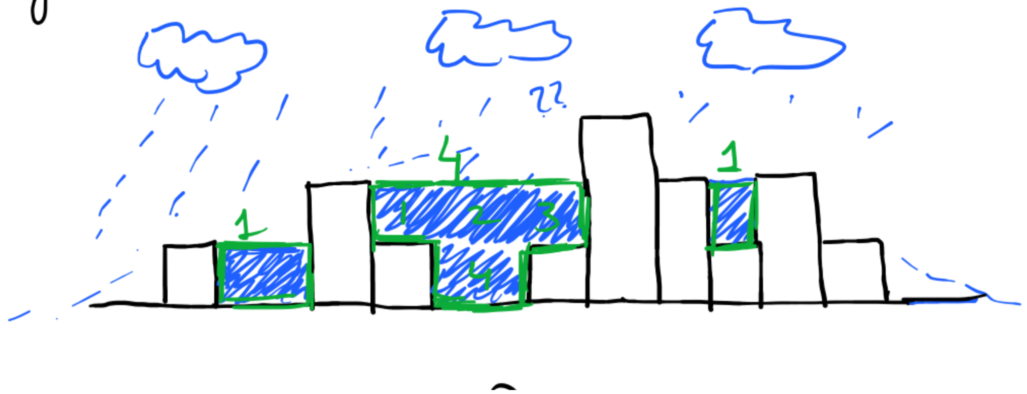
SC: O(n) → extra space

push(10)
push(5)
push(7)
getMin() → 5
push(1).
pop(~~1~~) → 1

2) Use 1 Stack of Pair → val, MinVal



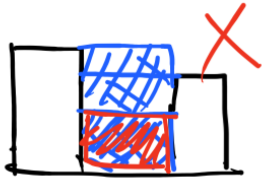
Trapping Rainwater Problem

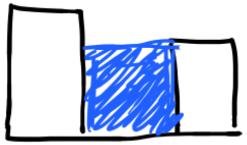


1+4+1 = 6

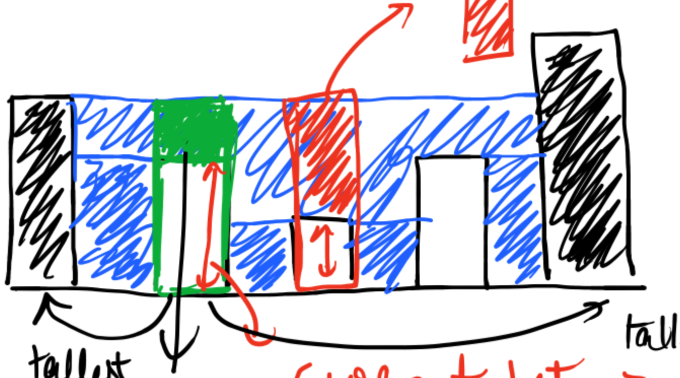
0 1 0 2 1 0 1 3 2 1 2 1

010210132121





🟥 = blocking ht - ↕

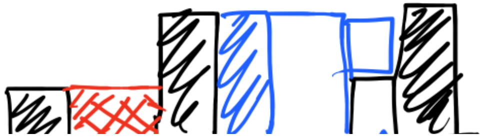


min: blocking ht
⇓
largest tower on left
largest tower on right

current ht = ht of building

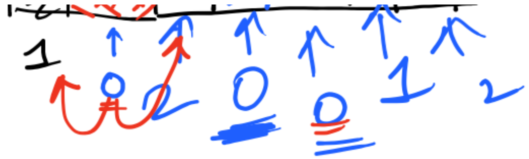
🟩 = blocking ht - current ht

min (tallest left, tallest right) ✓



min(2, 2) - 0 = 2

every element



will have diff tallest on left and right.

Brute Force

↳ Go on each building

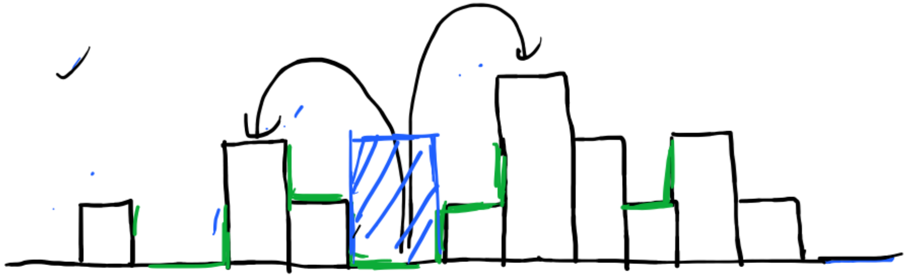
↳ Look on left and right and find the height of tallest building in each dirn

↳ choose the min of 2 tallest building

↳ subtract current ht.

TC: $O(N^2)$
SC: $O(1)$

```
for(i -> n){
    for(0 -> i-1) -> left
    for(i+1 -> n) -> right
}
```



0 1 0 2 1 $\boxed{0}$ 1 3 2 1 2 1

lmax

TC: O(N)

Prefix Max : -1 0 1 1 2 2 2 2 3 3 3 3 PM + SM + forloo

r max

Suffix Max : 3 3 3 3 3 3 3 2 2 2 1 -1

↳ Math.min(2,3) - ch

2 - 0 = 2