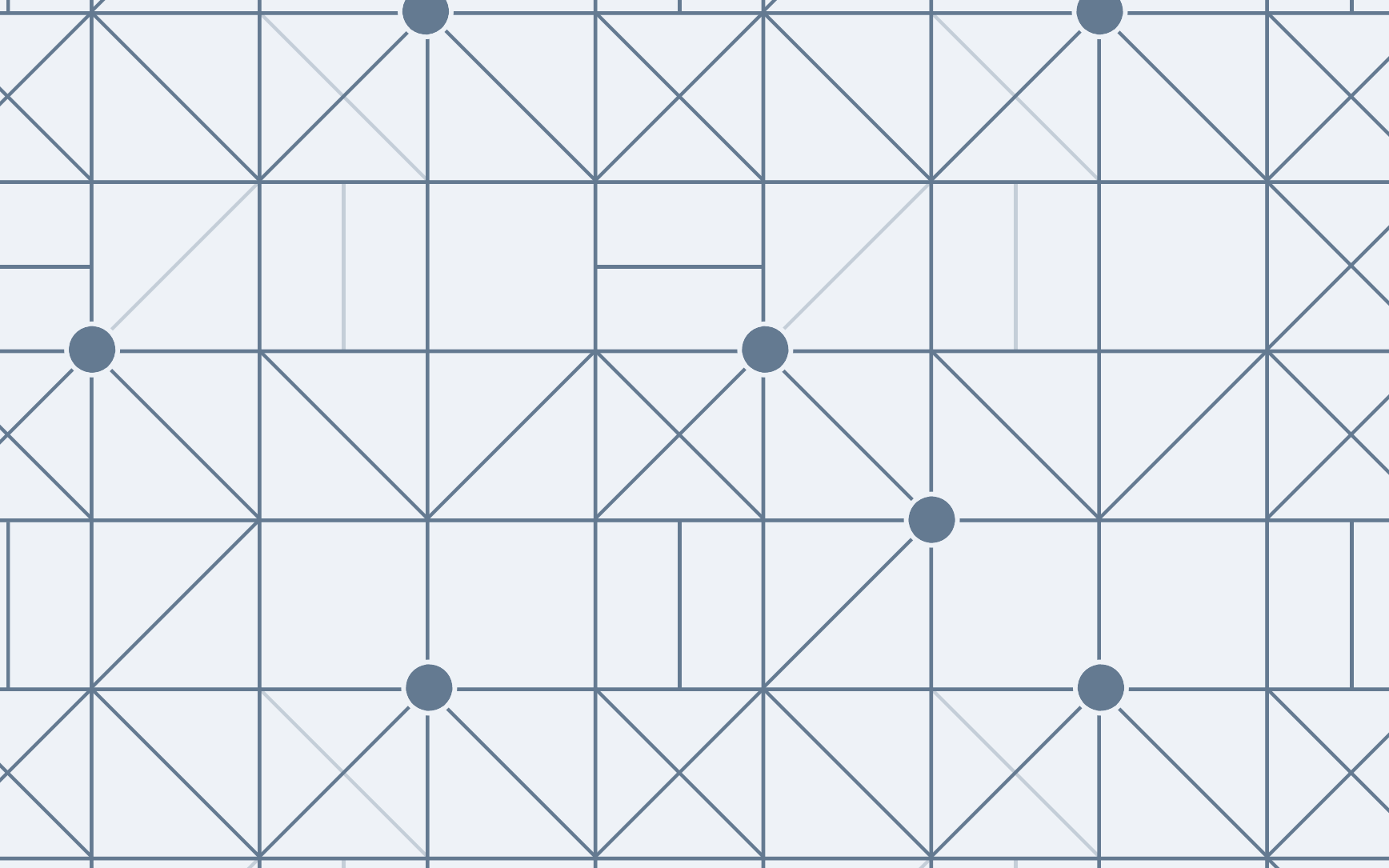


Last lect:

- Largest Area Histogram
- Celebrity Problem

→ Monotonic Stack

{ TC : O(N) SC : O(N) }

Agenda

- Sum of Subarray Minimums
- Trapping Rain water
- Minimum Stack

Sum of Subarray Minimums

```
int[] arr = { 3, 2, 4, 1, 5, 2 }
            0  1  2  3  4  5
```

Subarray?

Subarrays:

{3} {2} {4} {1} {5} {2}

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

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

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

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

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

Σ subarray min = 36

Brute force.

- Calc. all subarrays, get min^m in them
- Add all minimums

```
int sum = 0;
for (int i = 0 ⟶ n)
{
    int min;
    for (int j = i ⟶ n)
    {
        min = Math.min(min, arr[j]);
        sum += min
    }
}
```

TC : O(N²)
SC : O(1)

int[] arr = { 3, 2, 4, 1, 5, 2 }

0 1 2 3 4 5

left

right

{5, 3, 9, 2, 4, 10, 1}

principle of counting.

(pivot)

{2}

left side

{9}

{3, 9}

{5, 3, 9}

right side

{4}

{4, 10}

{2}

3 { {9, 2} → {2, 4}, {2, 4, 10}

{3, 9, 2} → {2, 4}, {2, 4, 10}

{5, 3, 9, 2} → {2, 4}, {2, 4, 10} } 2

(3 x 2) {

{5, 3, 9, 2, 4}

{5, 3, 9, 2, 4, 10}

{9, 2, 4} {9, 2, 4, 10}

{3, 9, 2, 4} {3, 9, 2, 4, 10}

total = 1 + 1x1 + 1x2 + 1x1x2 ✓

1 4 1 12

int[] arr = {3, 2, 4, 1, 5, 2}

0 1 2 3 4 5

✓

nseli = {-1, -1, 1, -1, 3, 3}

nsri = {1, 3, 3, 6, 5, 6}

→ l = (x - nseli - 1)

r = (nsri - x - 1)

3 - (-1) - 1 = 3

6 - (3) - 1 = 2

sum

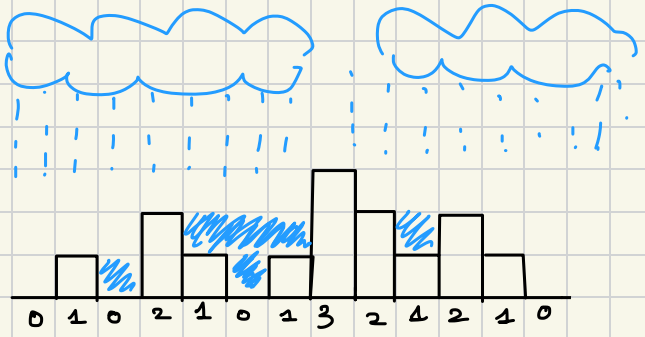
1x3 + 4x2 + 1x4 + 12x1 - -

1 + 3x1 + 2x1 + 3x1x2

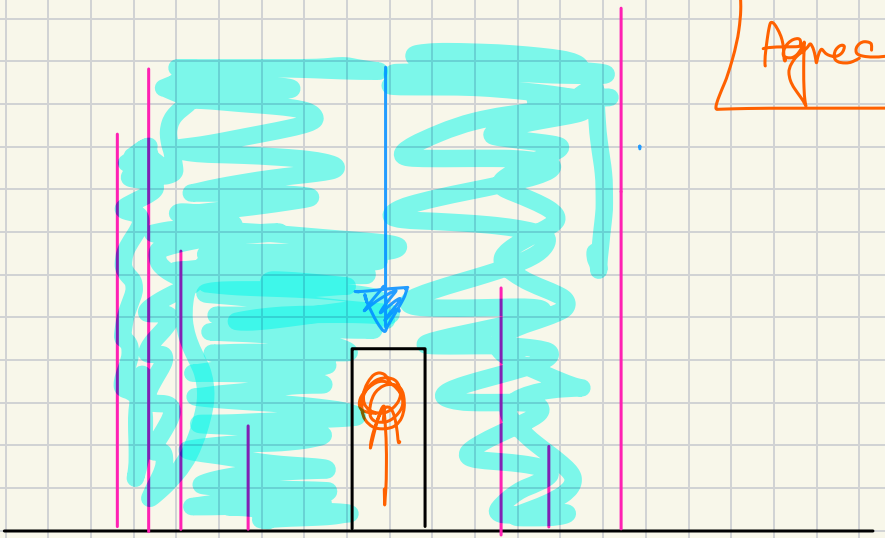
= 12

Trapping rain water.

list[] = {0, 1, 0, 2, 1, 0, 3, 2, 1, 2, 1, 0}



amt of rain trapped = 6 sq unit of rain

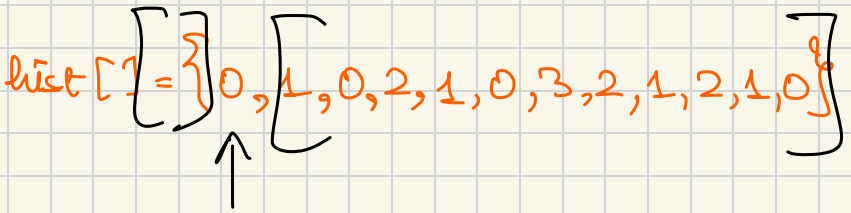


Agree?

Overall greatest on left

Overall greatest on right.

Brute force

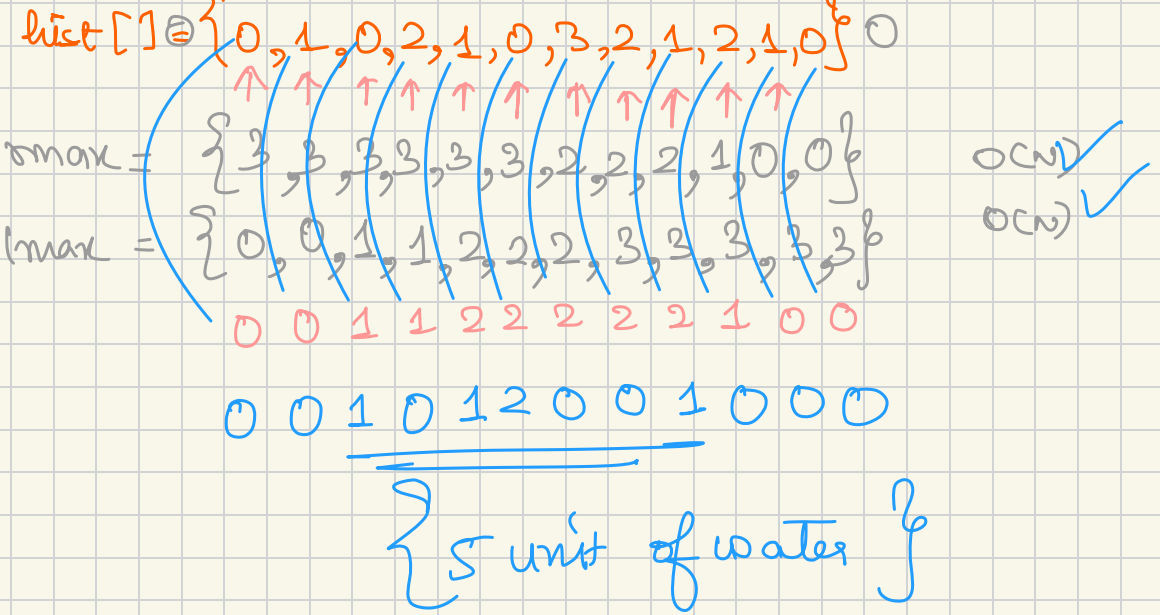


TC : O(N²)
SC : O(1)

- go at each building
- scan right side and get max.
- scan left side and get max.
- Calc water above you {if any}

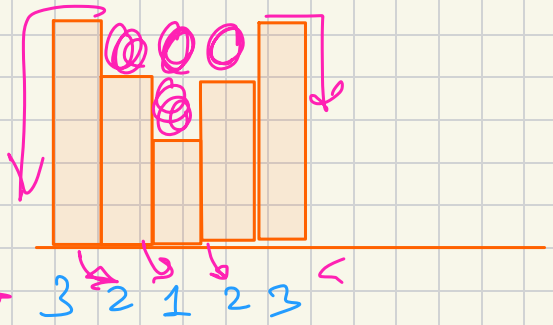
✓ Can we do better?

prefix Sum.



O(N) ✓

O(N) ✓



3 2 1 2 3

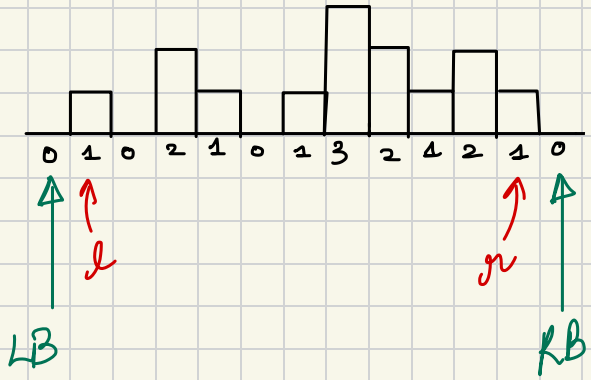
0 3 3 3 3

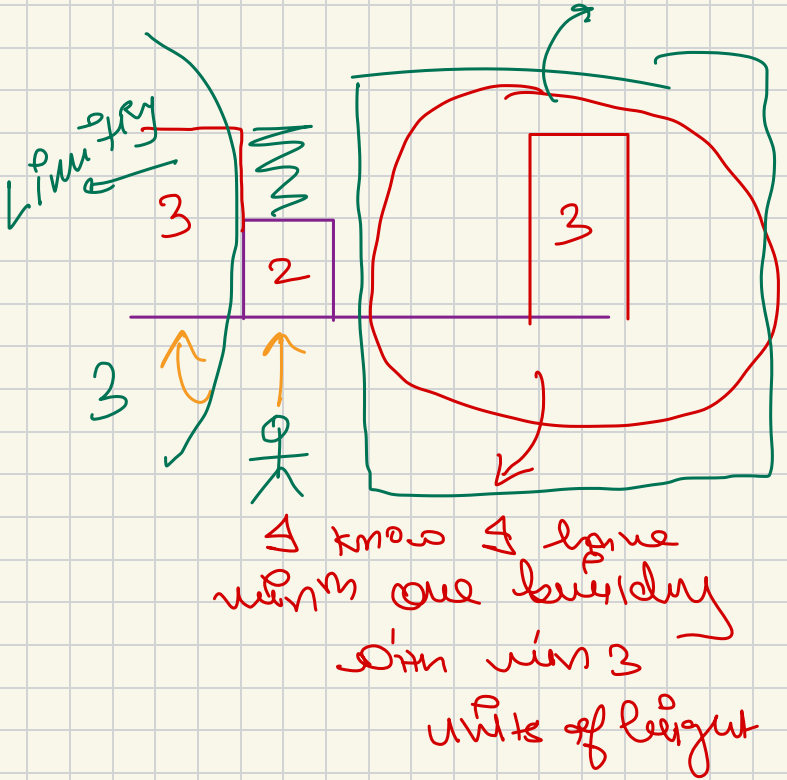
3 3 3 3 0

0 3 3 3 0

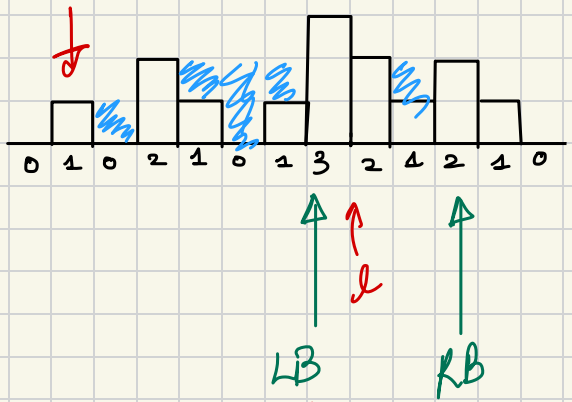
1 2 1

= 4





I know I have
with one building
of min 3
units of height



✓ LB → On max height Building for l

RB → On max height Building for r

LB == RB

{ LB is limiting,

hw = LB,

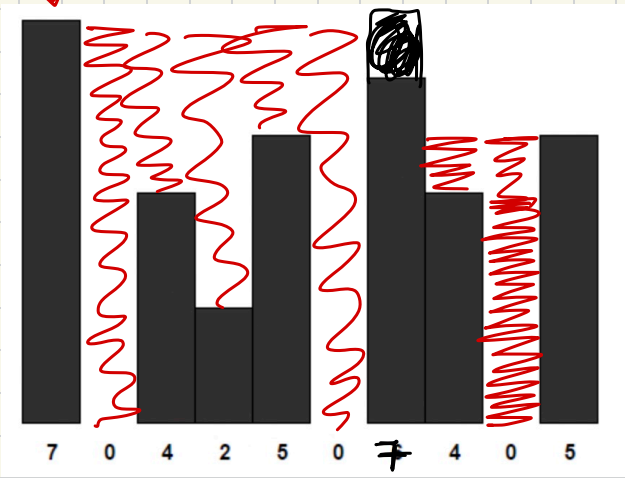
l++

LB > RB

{ RB is limiting

hw = RB

r--



$(RB >= LB)$

LB is limiting

$h_w = LB$

$(RB < LB)$

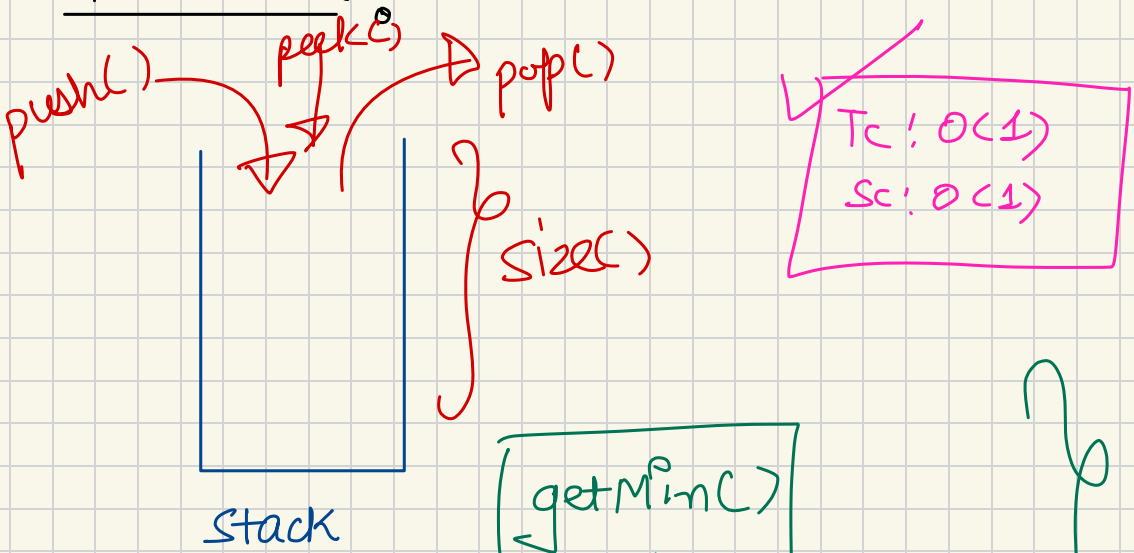
RB is limiting

$h_w = RB$

h

l

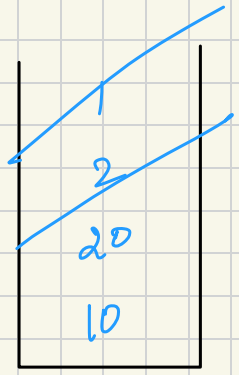
MinStack



Tc : O(1)
Sc : O(1)

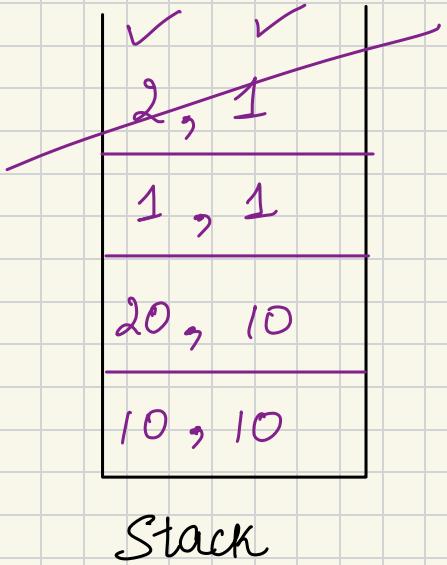
getMin()

Tc : O(1)



stack

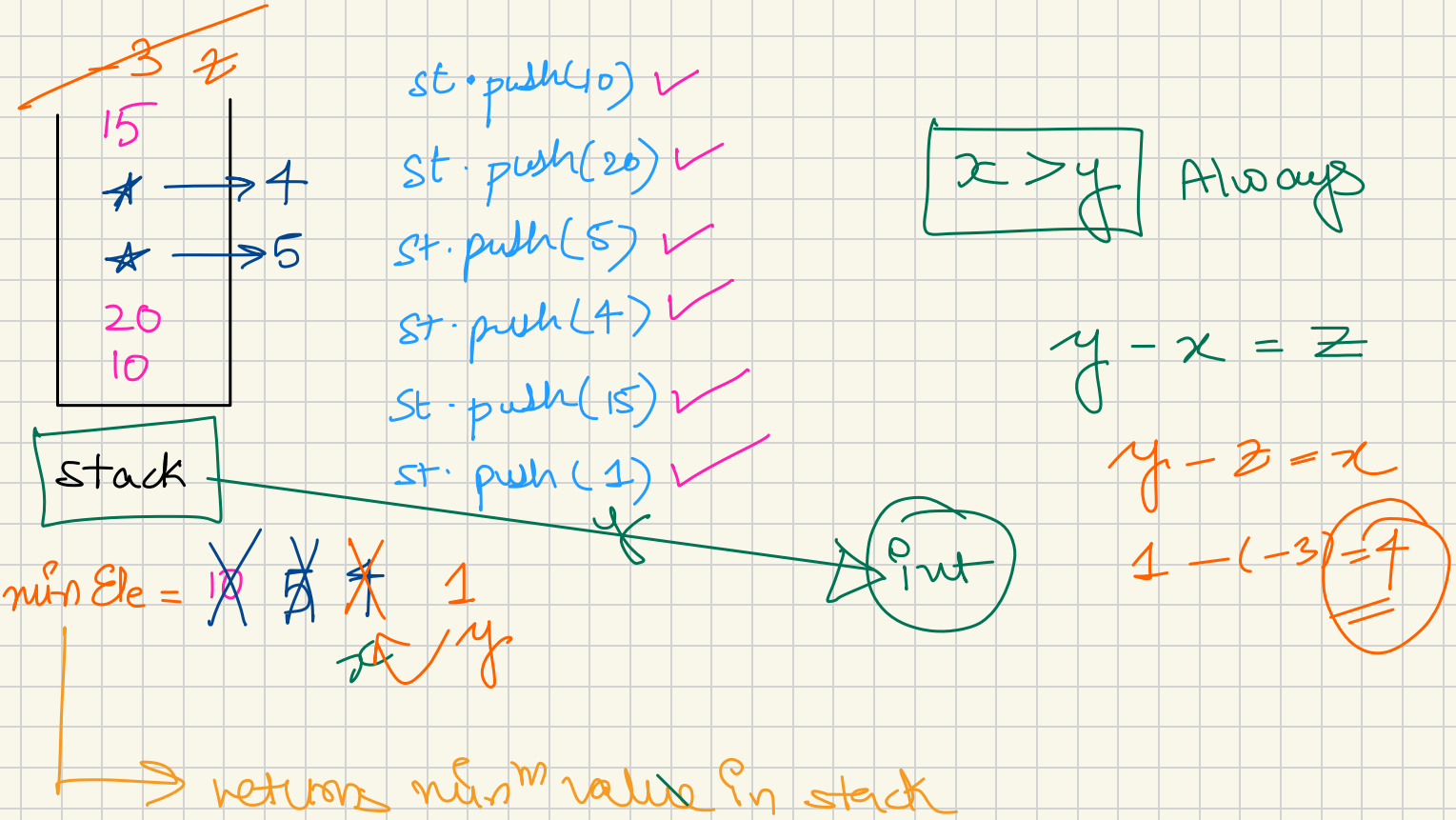
st.push(10)
st.push(20)
st.push(2)
st.push(1)
st.peek() ⇝ 1
st.size() ⇝ 4
st.getMin() ⇝ 1
st.pop() ⇝ 1
st.getMin() ⇝ 2
st.pop() ⇝ 2
st.getMin() ⇝ 10

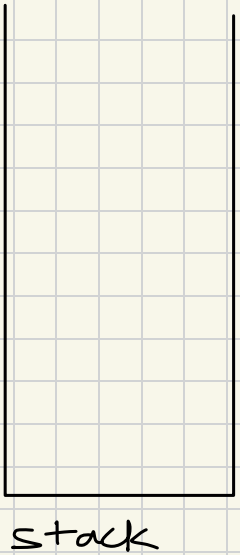



```
class Node
{
    int val;
    int minVal;
}
```

10,

20


-3 z
15
4
5
20
10
stack
st.push(40)
st.push(20)
st.push(5)
st.push(4)
st.push(15)
st.push(1)
int
minEle = 10 5 1
x y
x > y Always
y - x = z
y - z = x
1 - (-3) = 4
return minm value in stack



st.push(10)
st.push(20)
st.push(5)
st.push(4)
st.push(15)
st.push(1)

minEle =

5 - 10 = -5
4 - 5 = -1
1 - 4 = -3

minEle - (peek)