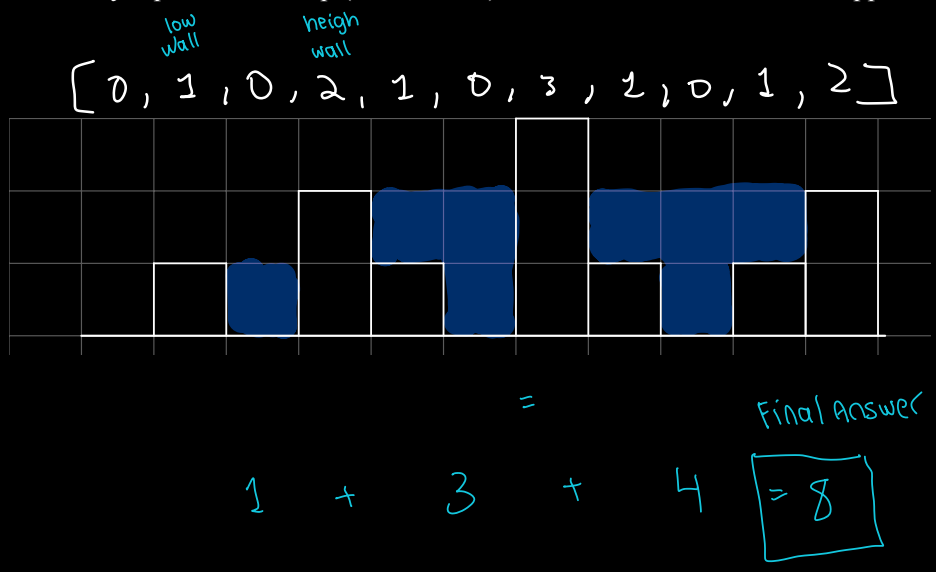
Problem: Given int array rep elevation map (bar width: 1), return amount of rainwater trapped

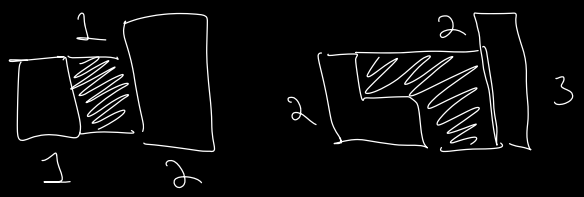


Intro:

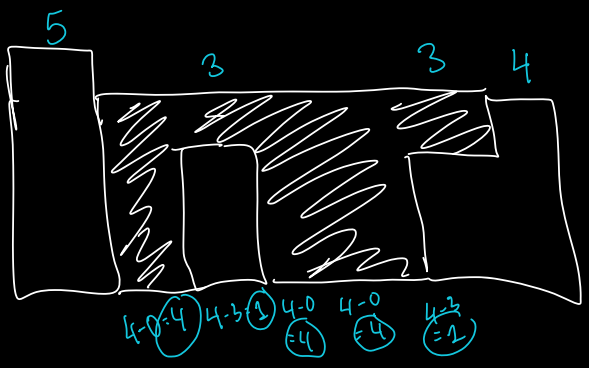
- Verify Constraints
  - Left & right sides count as walls? No
  - Negative ints? No
- Create Testcases
  - [0, 1, 0, 2, 1, 0, 3, 1, 0, 1, 2] -> 8 sqs of water
  - [] -> 0
  - [3] -> 0
  - [3, 4, 3] -> 0

Brute Force:

- Brainstorming & Pattern Observations



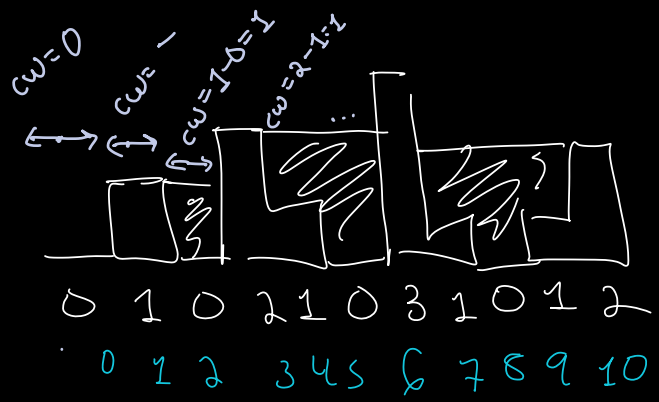
smaller of the 2 walls determines the max height of water



4+1+4+4+1=14

current_water = min(maxL, maxR) − current_height

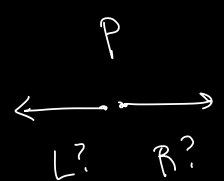
formula to calculate waterheight at every index



total_water

maxL = 0

maxR = 0



p	water height
1	min(0, 3) - 1 = -1
2	min(1, 3) - 0 = 1
3	min(2, 3) - 2 = 0
4	min(2, 3) - 1 = 1
5	min(2, 3) - 0 = 2
6	min(3, 2) - 3 = 2
7	min(3, 2) - 3 = -1
8	min(3, 2) - 0 = 2
9	min(3, 2) - 1 = 1
10	min(3, 2) - 2 = 0

iterate thru each index:

at each index:

find the highest left wall & highest right wall

calculate the water height & add it to the total water amt

- Pseudocode

```
iterate thru each index:
    at each index:
        find:
            highest left wall
            highgest right wall

        calculate the water height
        add to total water amt
```

- Write code
- Run through testcases
- Analyze time and space complexity

    Time: O(n^2): at each index, max left wall and max right could be at ends of array, so outer loop iterates thru n items and inner loop could iterate thru n times: n*n = n^2

    Space: O(1)

<u>Optimal:</u>

- Brainstorming & Pattern Observations

    Technique check:

        tradeoff between space and time? no

        shifting 2 pointers? maybe

        need pointers to move inwards

current water = min(maxL, maxR) − curr height

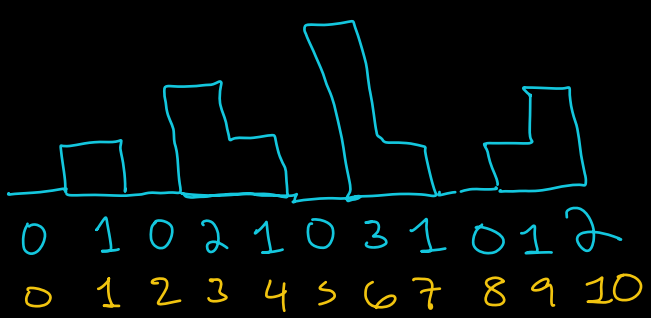
conditionally moving 2 pointers inward

move smaller pointer

maxL or maxR

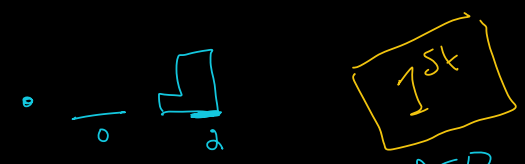
b/c we want the walls w/ the largest height

move smallest height pointer inward



maxL = 0

maxR = 0



1st

p = 0

0 < 2

maxL < maxR ⇒ True

Use maxL

smaller

↓

① calculate water : if maxL > curr_val

0 > 0

false

or

update maxL

maxL = (heights[p], heights[maxL]) = 0

② move pointer right

2nd

p = 1

0 < 2

maxL < maxR ⇒ True

Use maxL

maxL = 1

↓

smaller

↓

① calculate water : if maxL > curr_val

0 1

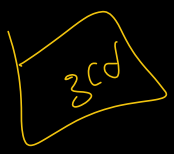
false

or

update maxL

maxL = (heights[p], heights[maxL]) = 1

② move pointer right



p = 2

1 < 2

maxL < maxR ⇒ True

use maxL

0 or 2 greater

less

① calculate water : if maxL > curr_val

1 0

or

update max

↓

t = maxL − curr_val

② move pointer right

---

2 or 2 equal

left

p = 3

2 ≤ 2

maxL ≤ maxR

use maxL

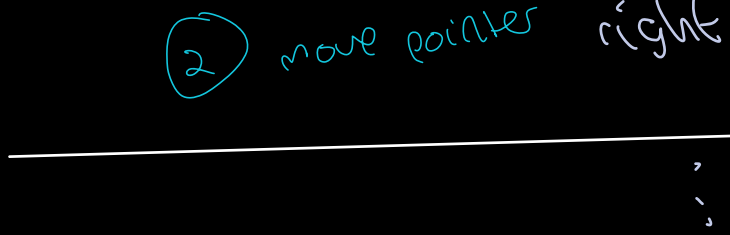
① calculate water : if maxL > curr_val

1 > 2

or

update maxL?

↳ maxL = max(maxL, curr_val)



- Pseudocode

```
iterate thru each index:
    determine which pointer has the min height
    use the pointer with the min height
    if max > curr_val:
        total_water += max - curr_val
    else:
        max = curr_val

    move pointer inwards
```

- Write code
- Run through testcases
- Analyze time and space complexity:
  Time: O(n): the 2 shifting pointers traverse thru the whole array once bc of moving the lowest height pointer
  Space: O(1)