Agenda of thursday.

1. Trapping Rain water →
   - O(n) space →
     - Array → lmax, rmax] height
     - Stack
   - O(1) space →
     - 2 pointer Approach] Stack
     - Dual Iteration Approach

2. Trapping Rain water -2 → Priority Queue

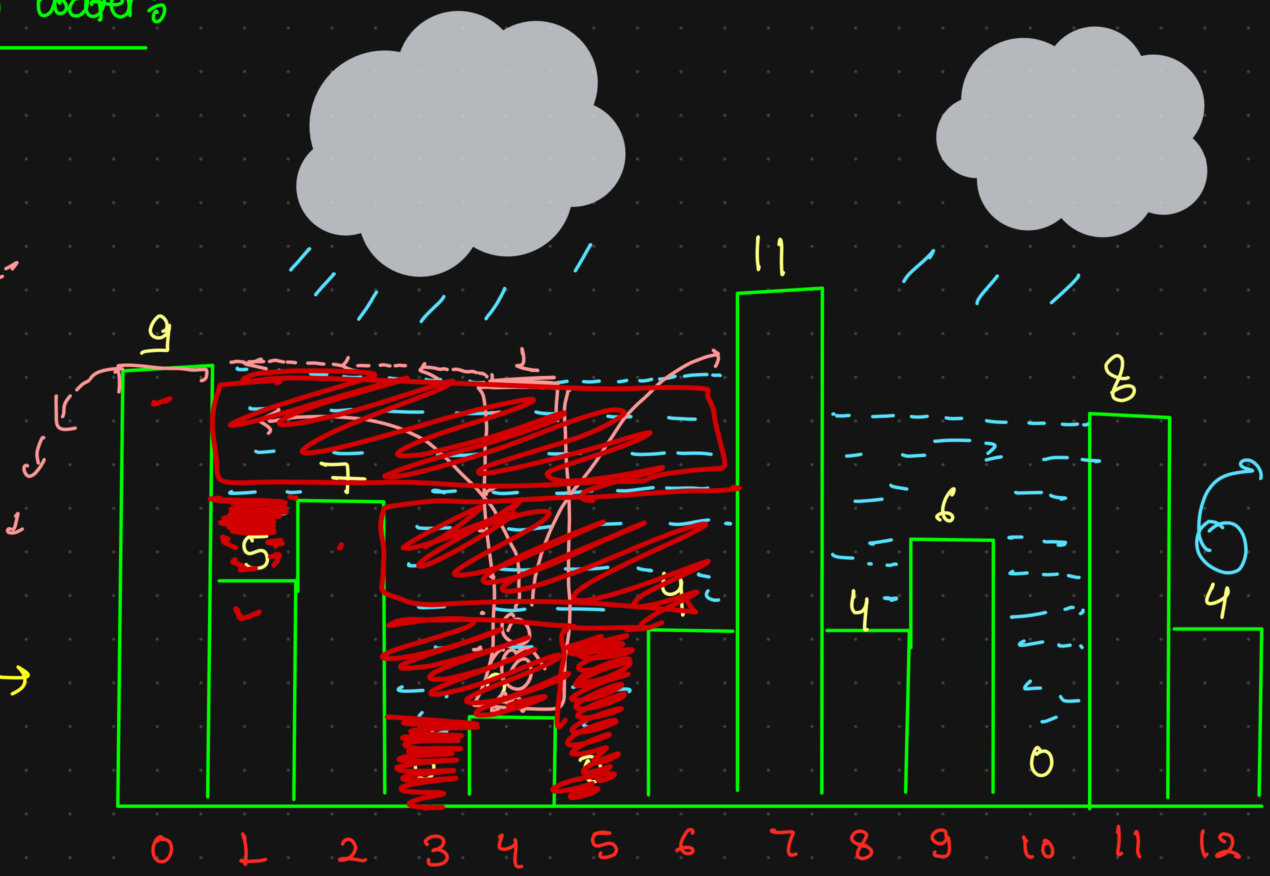
3. Basic Calculator - I   + I

4. Basic calculator - II] → Infix Evaluation $\underbrace{+,\ -,\ *,\ /}_{\text{Based on priority}}$, ()
   - → +ve Integers

Trapping Rain water:

$1 m^3$ = valum?
↳ 1000 Litre

Amount of water store b/w building ?



no water

water in terms of volume

water on head of ith building

Bruteforce - ⓪ →

* Reach at Every building & find leftmax & Right max
* find water on head of ith building

Bruteforce ① →

Manage lmax & rmax from an array,

← left → right

highest building in left

highest building in Right

$$water += \underbrace{\min(l_{max}, r_{max})}_{height} \times \overset{width}{1} \overset{volume}{-} \underbrace{ht[i] \times 1}_{occupied\ by\ building}$$

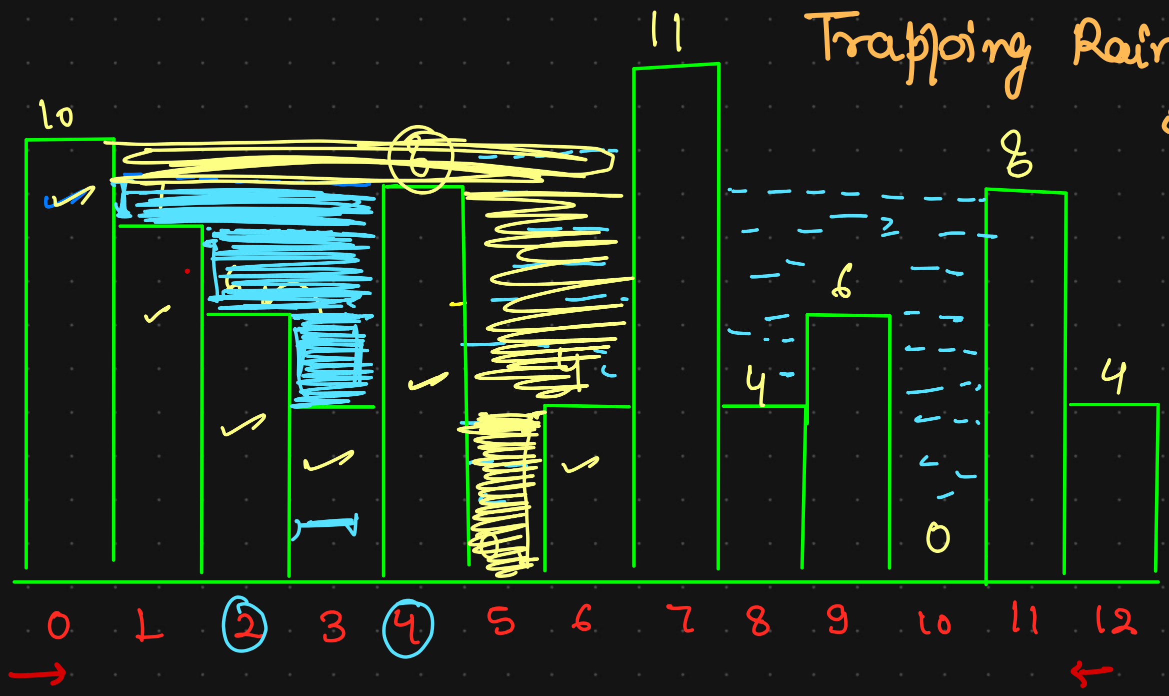
How we get lmax & rmax. ??

Trapping Rain water using STACK

next greater Element

Stack order

↳ Decreasing from Bottom to top

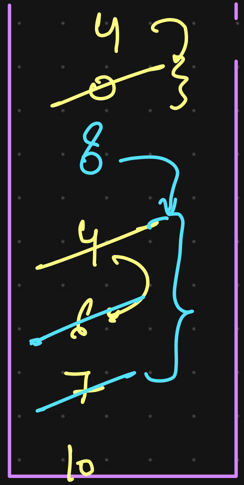


rmax = 4 , rmaxgndex = 6

current height = 0

lmax = 8, lmax gndx = 4.

water = + ht * wd



rmax = ~~8~~ , rmax indx = 4.

current height = ~~4~~, ~~6~~ 7

size = = peek(hm

lmax = ~~6~~ ~~7~~ 10 , lmax gndx = ~~2~~ - ~~1~~ 0

width = rmax indx - lmax gndx - 1;

$$water \mathrel{+}= \underbrace{[min(lmax, rmax) - current hight]}_{height} * \underbrace{width}_{width}$$

rmax = 2
indx = st.pop(); = 5
curr building = height[index]

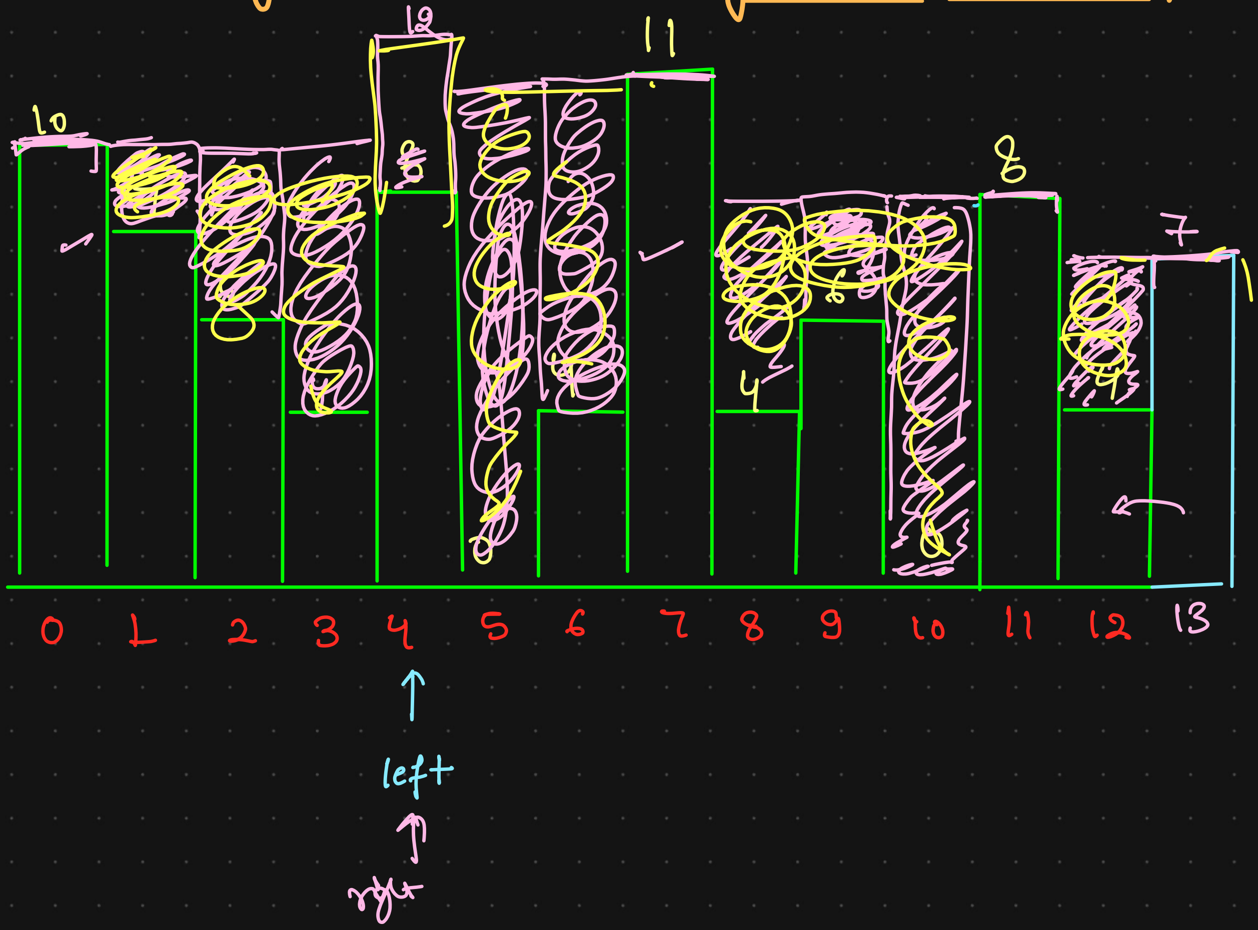
lmax gndx = 0

water += .

This Approach is Application of next greater on right

Trapping Rain water using two pointer Approach:

Think About head of building.



left = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

leftmax = ~~10~~ 12

right = ~~13~~ ~~12~~ ~~11~~ ~~10~~ ~~9~~ ~~8~~ ~~7~~ ~~6~~ ~~5~~ 4

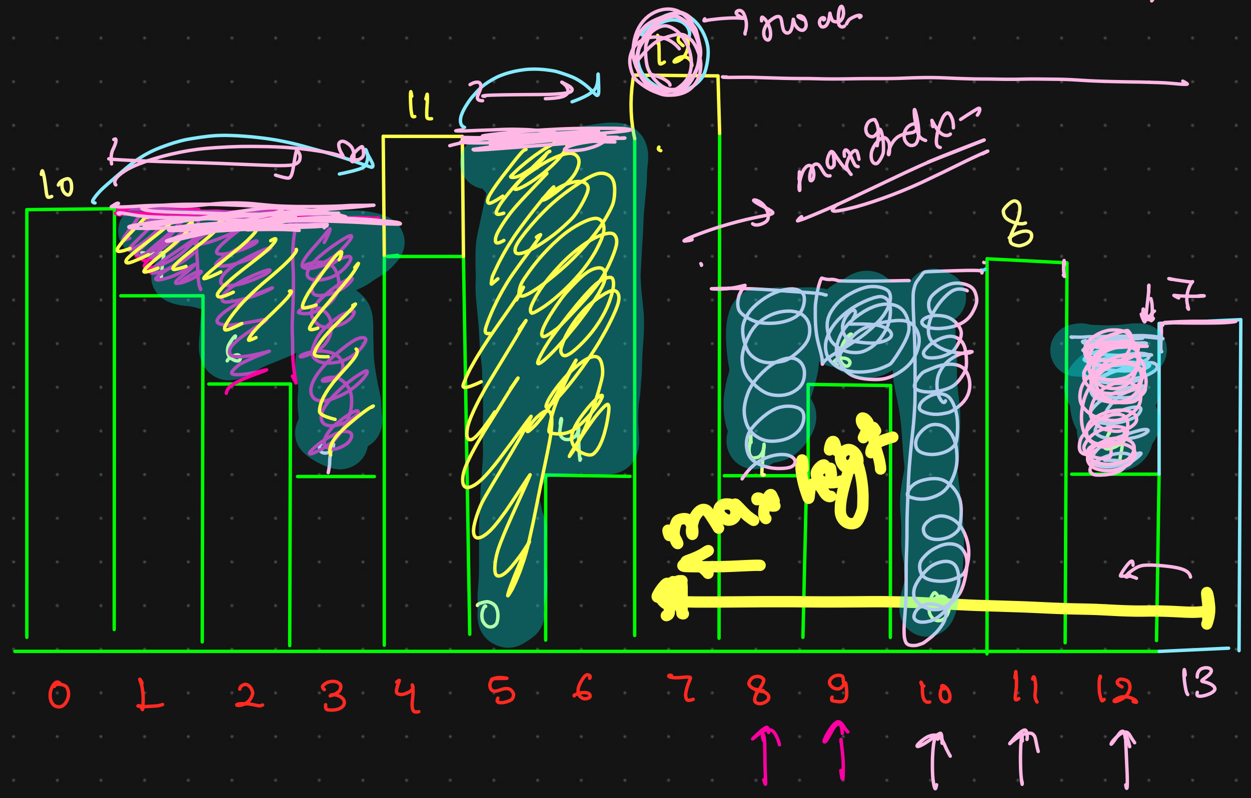
rightmax = ~~7~~ ~~8~~ 11

```
while( left < right) {
    leftmax = max(leftmax, ht[left]);
    rightmax = max(rightmax , ht[right]);
    if(leftmax < rightmax) {
        water += leftmax - ht[left];
        left ++;
    }else {
        water += rightmax - ht[right]!
        right - -,
    }
}
```

water = 0 + 3 + 8 + 2 + 4 + 3 + 4 + 6 + 7 + 11 = 48

Result

overall res = 0+3+4+6 + 11+7 +3 +8+2 +4



```
if( max <=ht[i]){
  max = ht[i];
  overall res += water;
  water = 0;

}

water = max - currentht
```

water = 0 + 3 + 4 + 6 + 0 + 11 + 7 + 0 + 8 + 6 + 12 + 4 + 8 + 5 +

max = ~~10~~ ~~11~~ 12

maxIndex

water = 0+3 + 0 +8 +2 +4
max = ~~7~~ 8 → Iteration from Right to left
↳ stop at maxIndex (old)

j
"35 + 27 - 8"
i

"35" ← String to Ints