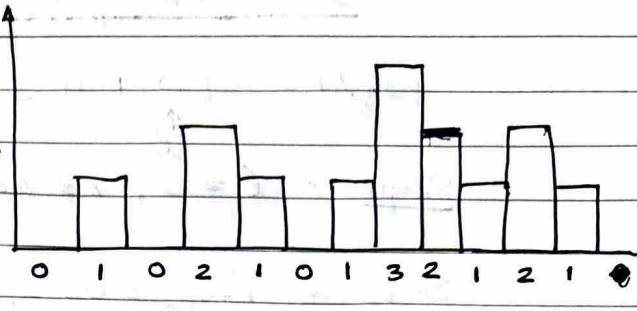
DATE

LEETCODE | 42|

Trapping Rain Water

Input

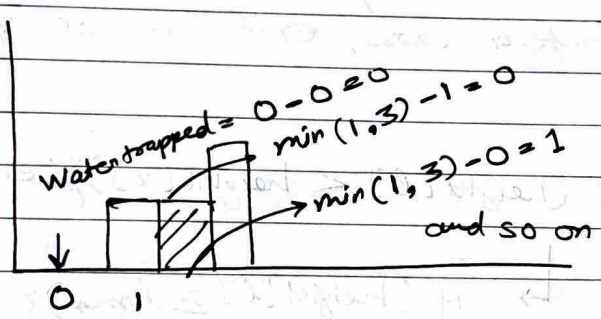
height = [0,1,0,2,1,0,1,3,2,1,2,1]



Output = 6

Intuition

For at any index, water trapped = min(l_{max}, r_{max}) − current height



Better solution Using prefixSum & suffixSum which gives height max (left) & height max (right) respectively

Step1 → for loop from i=0 to height.length

Step2 → prefix = [0,1,1,2,2,2,2,3,3,3,3,3]

Step3 → suffix = [3 3 3 3 3 3 3 3 2 2 2 1]

Step4 → for loop again from i=0 to height.length

```
int trapped = 0;
for (i=0 → height length)
    trapped += min(prefix[i], suffix[i]) - height[i]
```

Step5) → Return trapped

DATE

Optimal Approach

Using Two pointers

height = [0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1]
↑ l ↑ r

Step1 → $l = 0$, r = height.length − 1 ← two pointers

Step2 → $l_{max} = 0$, $r_{max} = 0$ ← to store max in both directions

Step3 → trapped = 0

Step4 – In all two pointers ($l \leq r$) ← once pointers cross, exit so use while ($l \leq r$)

Step5

```
if (height[l] ≤ height[r]){//answer right has some big building
    ↳ if (height[l] ≥ lmax){
        lmax = height[l] ; //store left max till pointer
      }
      else {
        trapped += lmax - height[l];
      }
    l++;
}
```

Step6

```
else {
    if ( height[r] ≥ rmax ) {  //left has some big building
        rmax = height[r]
    }
    else {
        trapped += rmax - height[r]
    }
    r--;
```

Step7 Return trapped;

classmate

PAGE