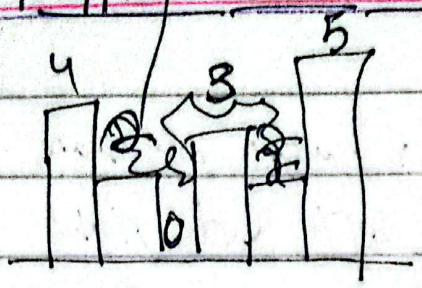
Trapping Rain water



Trapped water will be decided over the boundaries

① Naive ② Better ③ Optimal

① water trapped by each bar.

• boundaried of each bar

lmax[i] i rmax[i]

min (lmax, rmax)

① Naive (Brute force)

min(lmax, rmax) - ht[i]

② Better

Approach

vector<int> lmax(n,0)

vector<int> rmax(n,0)

lmax[0] = ht[0]

rmax[n-1] = ht[n-1]

Prefix Array

TC = O(n)

SC = O(n)

① for(i=1; i<n; i++){

lmax[i] = max(lmax[i-1], ht[i])

② for(i=n-2; i>=0; i--){

rmax[i] = max(rmax[i+1], ht[i])

for(i=0; i<n; i++){

ans += (min(lmax[i], rmax[i]) - ht[i])

2

3

lmax | 4 | 4 | 4 | 4 | 4 | 5 | — Individual Boundaries Tracking

rmax | 5 | 5 | 5 | 5 | 5 | 5 |

0 + 2 + 4 + 1 + 2 + 0 = 9

③ Optimal (2pointer Approach)

We will not use 2 arrays

ans = 0, l = 0, r = n-1

lmax = 0 rmax = 0

```
while(l < r){
lmax = max(lmax, ht[l]);
rmax = max(rmax, ht[r]);
if(lmax < rmax) — left
  ans += (lmax - ht[l]);
  l++
else — Right
ans += (rmax - ht[r]);
  r--
```

T.C = O(n)

S.C = O(1)