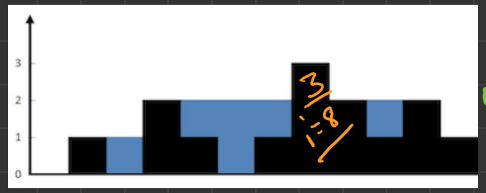
Trapping Rain Water



minheight = min(Leftmax, Right max)

if(minheight > height)

[0,1,0,2,1,0,1,3,2,1,2,1] → height

0 1 2 3 4 5 6 7 8 9 10 11

0 0 1 1 2 2 2 2 3 3 3 3 ← max left (N)

3 3 3 3 3 3 3 2 2 2 1 0 ← max right (N)

O(N)

0 1 2 3 4 5 6 7 8 9 10 11

0 0 1 0 1 2 1 0 0 1 0 0 ← ans (N)

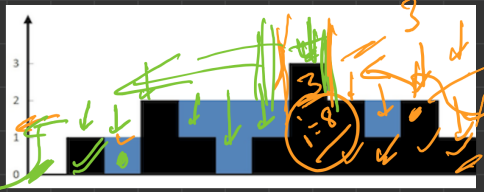
O(N)

```
vector<int> height ;
leftmax[n] , rightmax[n];
leftmax [0] = 0
for( i=1; i<n; i++)
leftmax[i] = max( leftmax[i-1], height[i-1]);
Rightmax [n-1] = 0;
for (i= n-2 ; i >=0 ; i++) {
Rightmax [i] =  max (Right max[i+1] , height[i+1]
int water =0;
for (i=0; i< n ; i++)
{
min height= min (left max[i] , right max[i])
```

n for
n
O(n)
n
3(n)
4n)

```
int water = 0;
for (i=0; i<n; i++)
{
    min height = min (left max[i], right max [i]);
    if (min height - height [i] >=0)
        water += minheight - height [i];
}
return water;
```

n



→ min (max left, max right)

Right

[0,1,0,2,1,0,1,3,2,1,2,1] → height

left max = 0 1

water = 0 +1 +1

1-0 = 1

O(n) O(1)

Right = 3

i = 8

2-1 = 1

1

max Right = 0 1 2

0 - 1 = -1

+ve

1 - 2 = -1

-ve

```
maxleft = 0, maxright = 0, water = 0;
maxheight = Height[0], index = 0;
for (i = 1; i < n; i++) {
    if (height[i] > maxheight)
    { maxheight = height[i];
        index = i;
    }
}
```

```
// left
for (i=0 ; i<index ; i++){
  if (leftmax > height[i])
     water += leftmax - height[i]
  else
     leftmax = height[i];
```

```
// Right part
for(i= n-1 ; i > index ; i--) {
if (Rightmax > height[i]) {
    water += Rightmax - height[i]
else
    Rightmax = height[i]
}
return water;
```

Triplet Sum in Array

arr[] = [1, 4, 45, 6, 10, 8], target = 13

x = target

[1, 4, 6, 8, 10, 45] target = 24

$O(n^2)$

$O(1)$

n →

```
for (i=0; i<n-2; i++)
    ans = x - arr[i];
    start = i+1, end = n-1;
    while (start < end)
      if (arr[start] + arr[end] == ans)
          return 1;
      else if (arr[start] + arr[end] > ans)
          end--
      else
          start++
return 0;
```