Understanding the problem statement

Given n non negative intergers representing elevation map,where width of each bar is 1, we have to find the amount of water trapped in between these bars.

Input : [1,0,2,0,1,0,3,1,0,2]

In the image we can see that each integer represents height, and we have to find amount of water trapped on the top of each building. Sum of all these gives total water trapped.

Output: Either total sum or water trapped on each building i.e [0,1,0,2,1,2,0,1,2,0]

Approach 1

1. From the image we can see that water will be trapped on the building only if there are taller buildings on both sides.
2. So we have find the greater number to the right side, and also greater number on the left side.
3. Take minimum of these two greater values and substract it with the current number.
4. If one of the number either on the left or on the right is lower than current number then no water is stored.
5. For finding greater number to right or smaller number to left we have to scan all elements. O(n).
6. We have to repeat the above process for n elements.

   So total time complexity is O(n) * n = O(n^2)

Time Complexity : O(n^2)

Space Complexity: O(1)

Approach 2

1. Maintain two arrays instead of scanning the array everytime to find max to right and max to left.
2. Build an array left[] of size n, where left[i] represents maximum height bar that is left to it including ith bar
3. Build an array right[] of size n, where right[i] represents maximum height bar that is right to it including ith bar
4. sum = 0

   repeat for every ith bar :

```
    sum = sum + minimum(left[i],right[i])-height[i]
```

5. return sum

6. We scan once to build left array --> O(n)

   We scan once to build right array --> O(n)

   We scan once to build total sum --> O(n)

   We maintain two lists left and right of size n --> 2n --> O(n)

Time Complexity : O(n)

Space Complexity: O(n)

Input : [1,0,2,0,1,0,3,1,0,2] Left : [1,1,2,2,2,2,3,3,3,3] Right : [3,3,3,3,3,3,3,2,2,2] Output: sum([0,1,0,2,1,2,0,1,2,0])

Implementation

In [13]:
```
def trappedRain(arr,size):
    left = []
    right= [0]*size
    max_left = 0
    max_right = 0
    for i in range(size):
        max_left = max(max_left,arr[i])
        left.append(max_left)
    for i in range(size-1,-1,-1):
        max_right = max(max_right,arr[i])
        right[i] = max_right
    trapped_rain = 0
    for i in range(size):
        trapped_rain = trapped_rain + (min(left[i],right[i])-arr[i])
    return trapped_rain
```

In [14]:
```
arr = [1,0,2,0,1,0,3,1,0,2]
size = len(arr)
print(trappedRain(arr,size))
# ans
# 12
```

```
9
```

In [ ]:
```
```