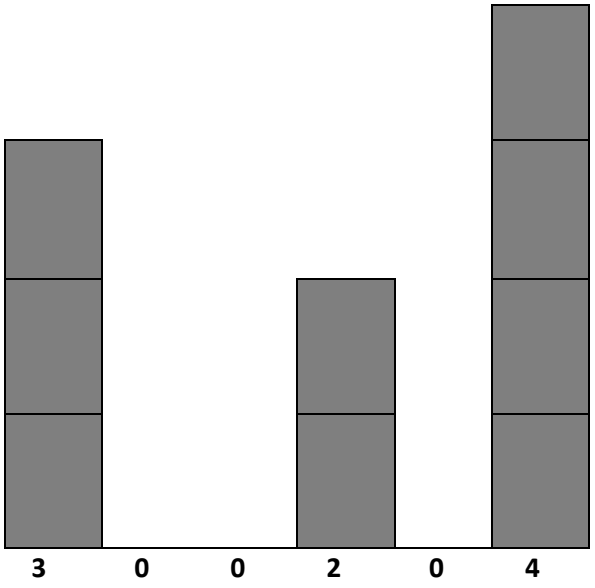
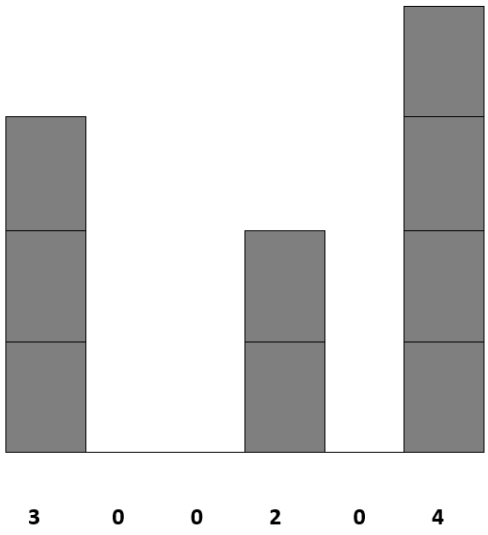
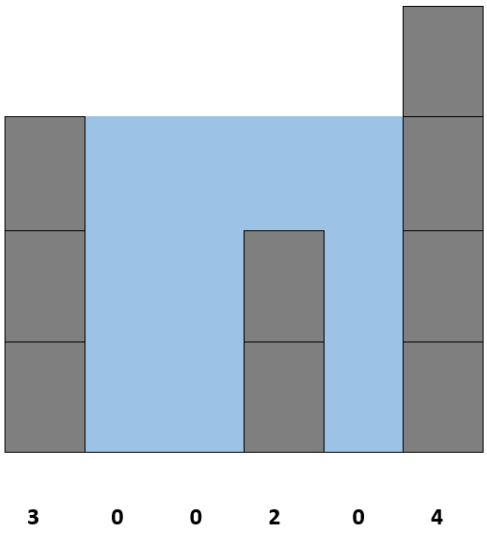
Trapping Rain Water (Water Area)

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much total water will be trapped after raining.

Input = [3, 0, 0, 2, 0, 4]







Total water trapped = 0 + 3 + 3 + 1 + 3 + 0 = 10

Approach:

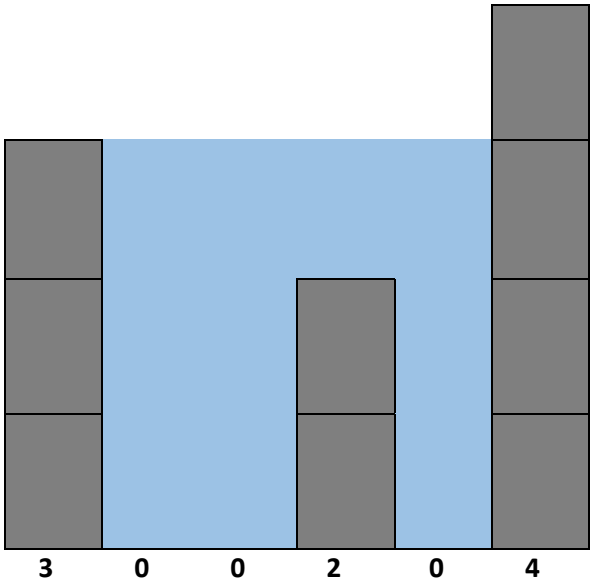
1. We will take an array (left) which will store the height of tallest bar at the left of i^{th} bar
2. We will also take an array (right) which will store the height of tallest bar at the right of i^{th} bar

After filling the values in left and right array, we are ready to calculate the water above i^{th} bar.

3. First we will calculate the, minimum value between left and right values for i^{th} bar
   minHeight = Math.min(left[i] , right[i])

Water above i^{th} bar =

- **minHeight - arr[i] , if minHeight > arr[i]**
- **0 , else**



Input = [3, 0, 0, 2, 0, 4]

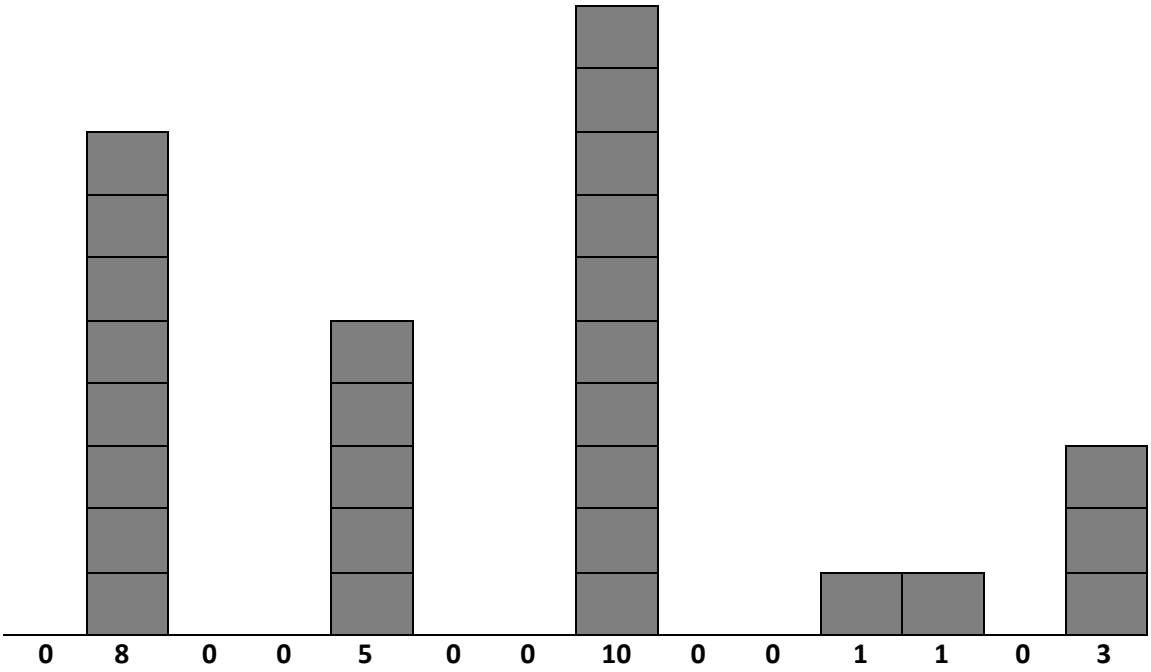
Left Max = [0, 3, 3, 3, 3, 3]

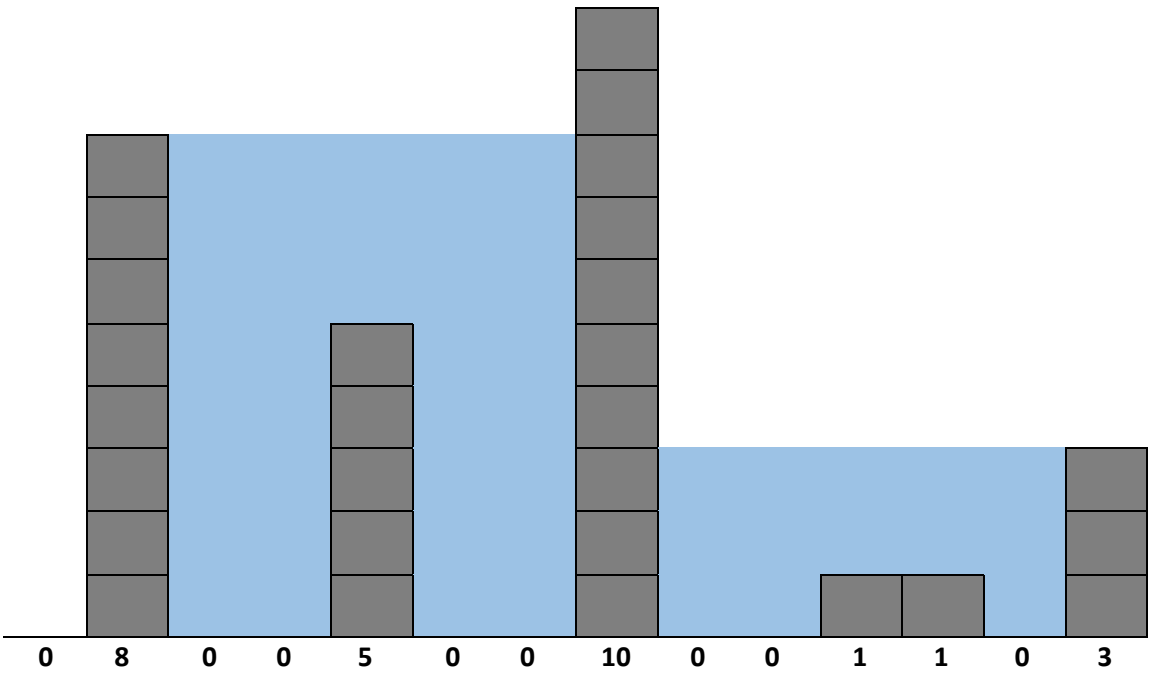
Right Max = [4, 4, 4, 4, 4, 0]

minHeight = {0, 3, 3, 3, 3, 0}

water = {0, 3, 3, 1, 3, 0}

```
Input = [0, 8, 0, 0, 5, 0, 0, 10, 0, 0, 1, 1, 0, 3]
```





Total water trapped = 0 + 0 + 8 + 8 + 3 + 8 + 8 + 0 + 3 + 3 + 2 + 2 + 3 + 0 = 48

Implementation

```
class Water {

	public static void main(String[] args) {
		int arr[] = {3, 0, 0, 2, 0, 4};
		System.out.println("Total water that can be trapped is " +
							waterArea(arr));
	}

	public static int waterArea(int arr[]) {

		if (arr.length == 0)
			return 0;

		// left[i] stores the height of tallest bar to the left of i'th bar
		int left[] = new int[arr.length];

		// right[i] stores the height of tallest bar to the right of i'th bar
		int right[] = new int[arr.length];

		// Fill left array
		left[0] = 0;
		int leftMax = arr[0];
		for (int i = 1; i < arr.length; i++) {
			left[i] = leftMax;
			leftMax = Math.max(leftMax, arr[i]);
		}

		// Fill right array
		right[arr.length - 1] = 0;
		int rightMax = arr[arr.length - 1];
		for (int i = arr.length - 2; i >= 0; i--) {
			right[i] = rightMax;
			rightMax = Math.max(rightMax, arr[i]);

		}

		int minHeight = 0;
		int water = 0;
		int totalWater = 0;

		for (int i = 0; i < arr.length; i++) {
			minHeight = Math.min(left[i], right[i]);
			if (minHeight > arr[i]) {
				water = minHeight - arr[i];
			} else {
				water = 0;
			}
			totalWater += water;
		}
		return totalWater;
	}
}
```

Time Complexity = O(n) { 3 for loops , O(n) + O(n) + O(n) }

Space Complexity = O(n) { 2 arrays (left and right) each of size n}

Optimization

```
class Water {

	public static void main(String[] args) {
		int arr[] = {3, 0, 0, 2, 0, 4};
		System.out.println("Total water that can be trapped is " +
					waterArea(arr));
	}

	public static int waterArea(int arr[]) {
		if (arr.length == 0)
			return 0;
		// left[i] stores the height of tallest bar to the left of i'th bar
		int left[] = new int[arr.length];


		// Fill left array
		left[0] = 0;
		int leftMax = arr[0];
		for (int i = 1; i < arr.length; i++) {
			left[i] = leftMax;
			leftMax = Math.max(leftMax, arr[i]);
		}

		int minHeight = 0;
		int water = 0;
		int totalWater = 0;

		int rightMax = 0;
		for (int i = arr.length-1; i >= 0; i--) {
			minHeight = Math.min(left[i], rightMax);
			if (minHeight > arr[i]) {
				water = minHeight - arr[i];
			} else {
				water = 0;
			}
			totalWater += water;

			rightMax = Math.max(rightMax, arr[i]);
		}

		return totalWater;
	}

}
```

Time Complexity = O(n) { 2 for loops , O(n) + O(n) }

Space Complexity = O(n) { 1 array (left) of size n }