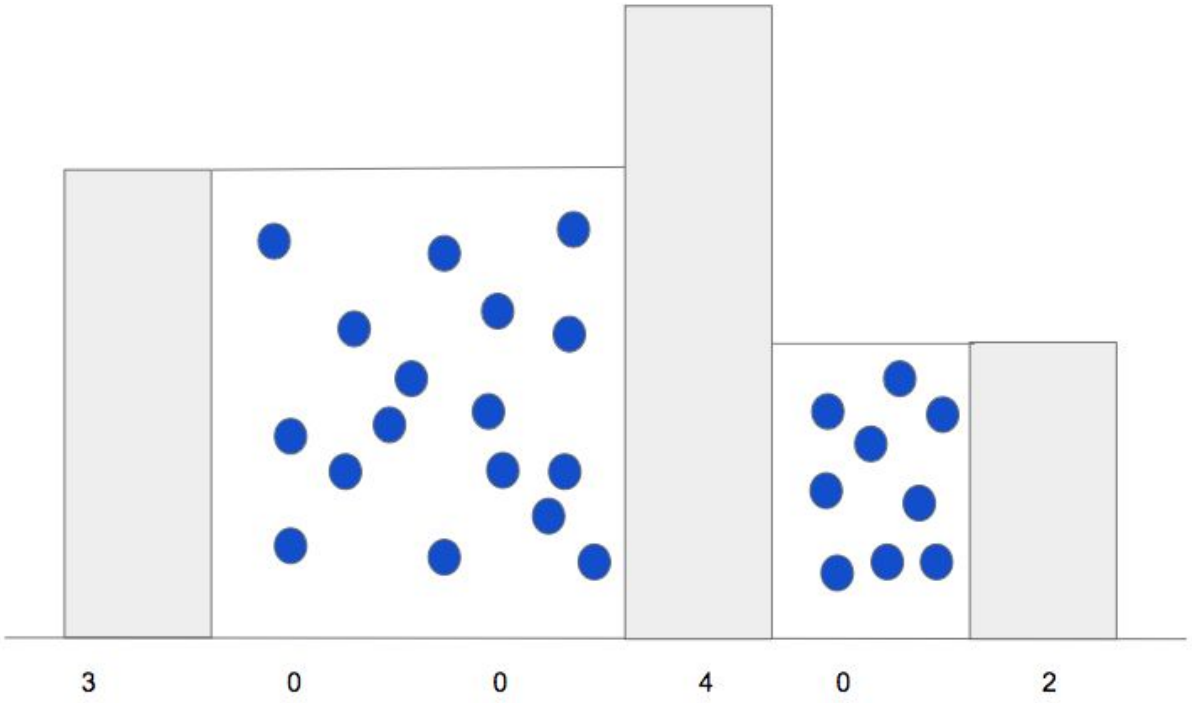
Gather Rain Water

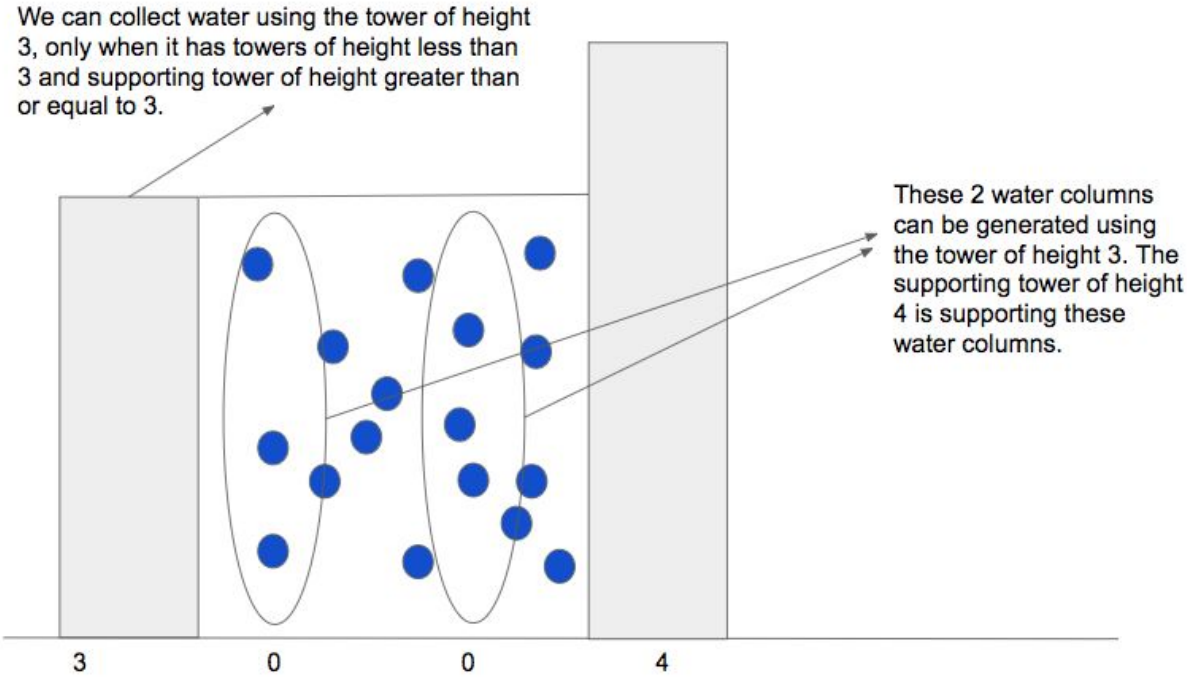
Problem Description: You are given an array of n integers representing heights of n towers. The width of all the towers is 1 unit. All the towers are standing close to each other such that they share an edge. The rain water can be collected, as described in the figure below:



The shaded portion with blue dots represents the collected water and the numbers below represent the heights of the tower. The water collected will be calculated in units square, as we are provided with only two dimensions (height and width). The amount of water collected, in the above figure, is 8 units square.

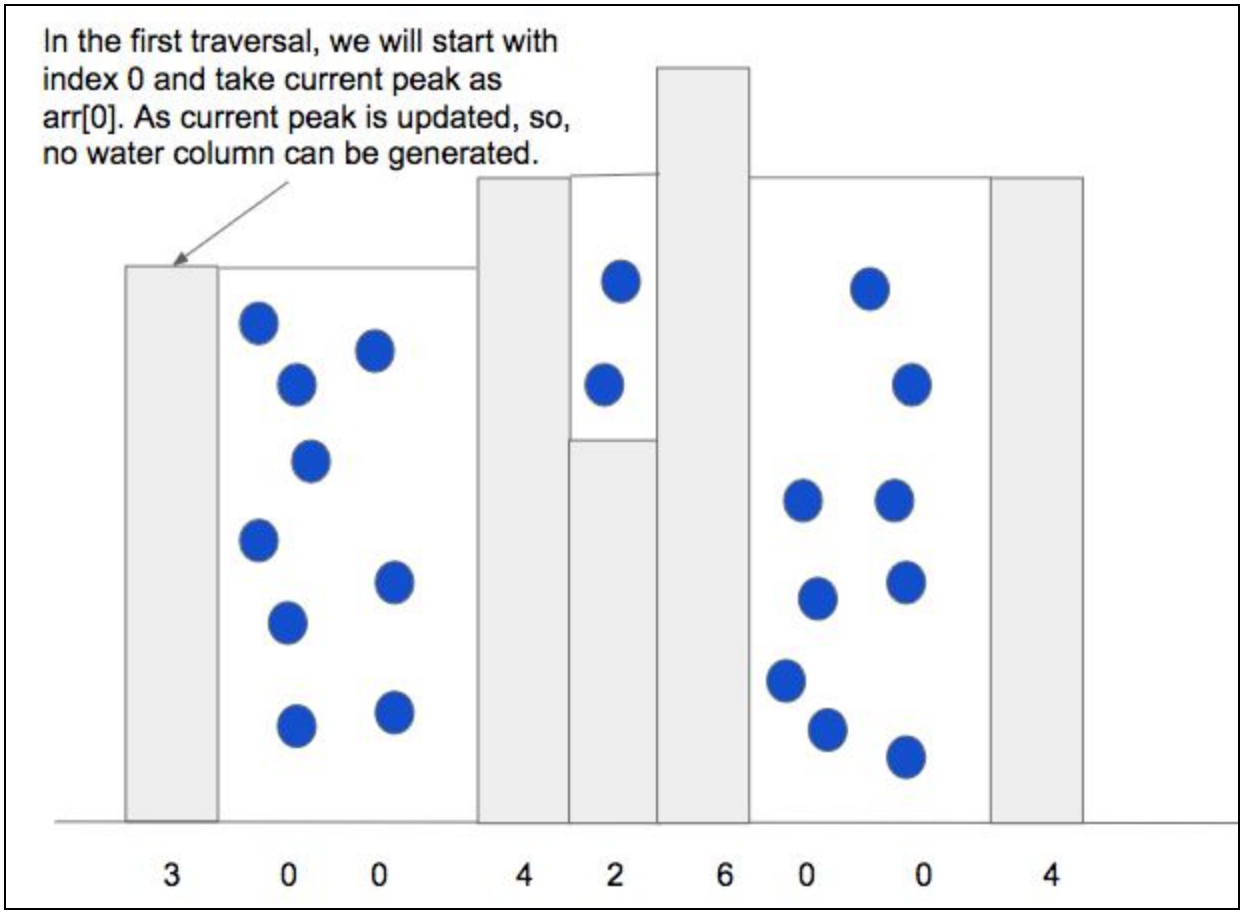
Solution:
There are some insights which lead us to the solution.

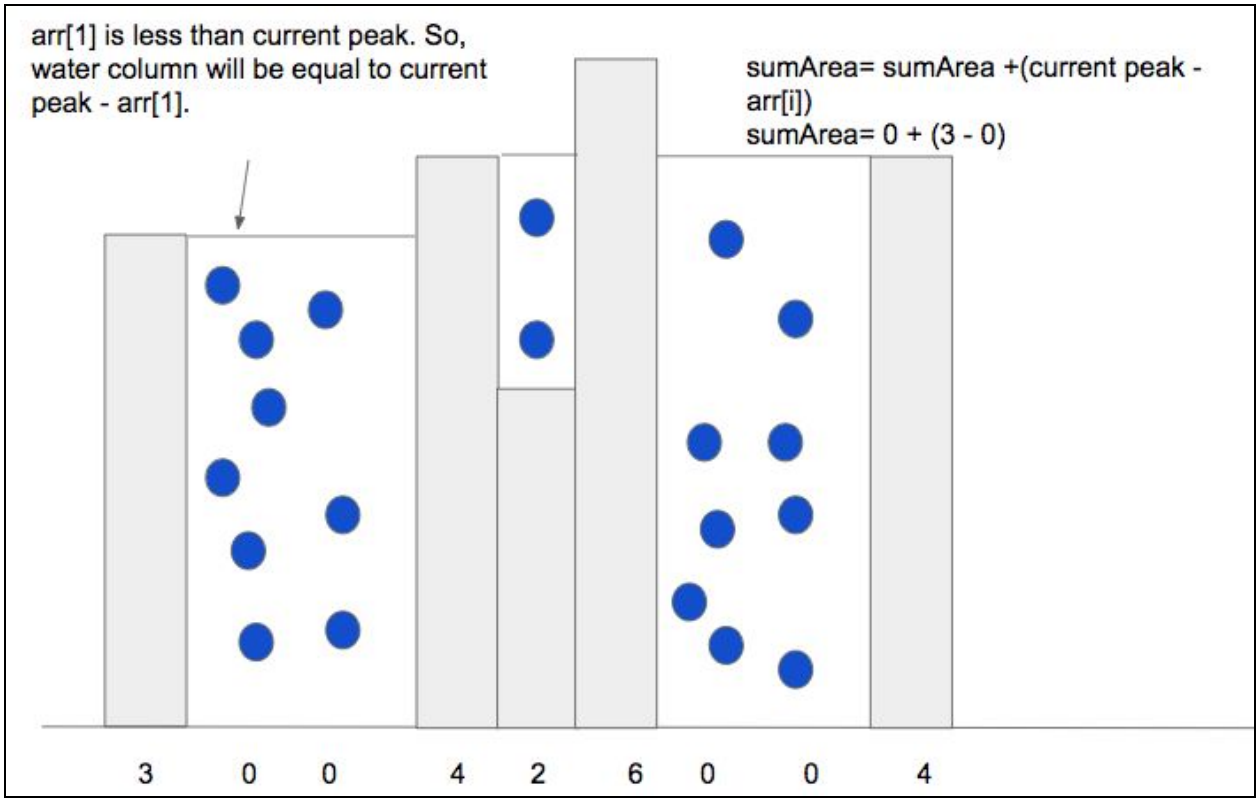
1. With the help of a tower of height h, we can store water only when there are few towers with heights less than h and a supporting tower of height greater than or equal to h.

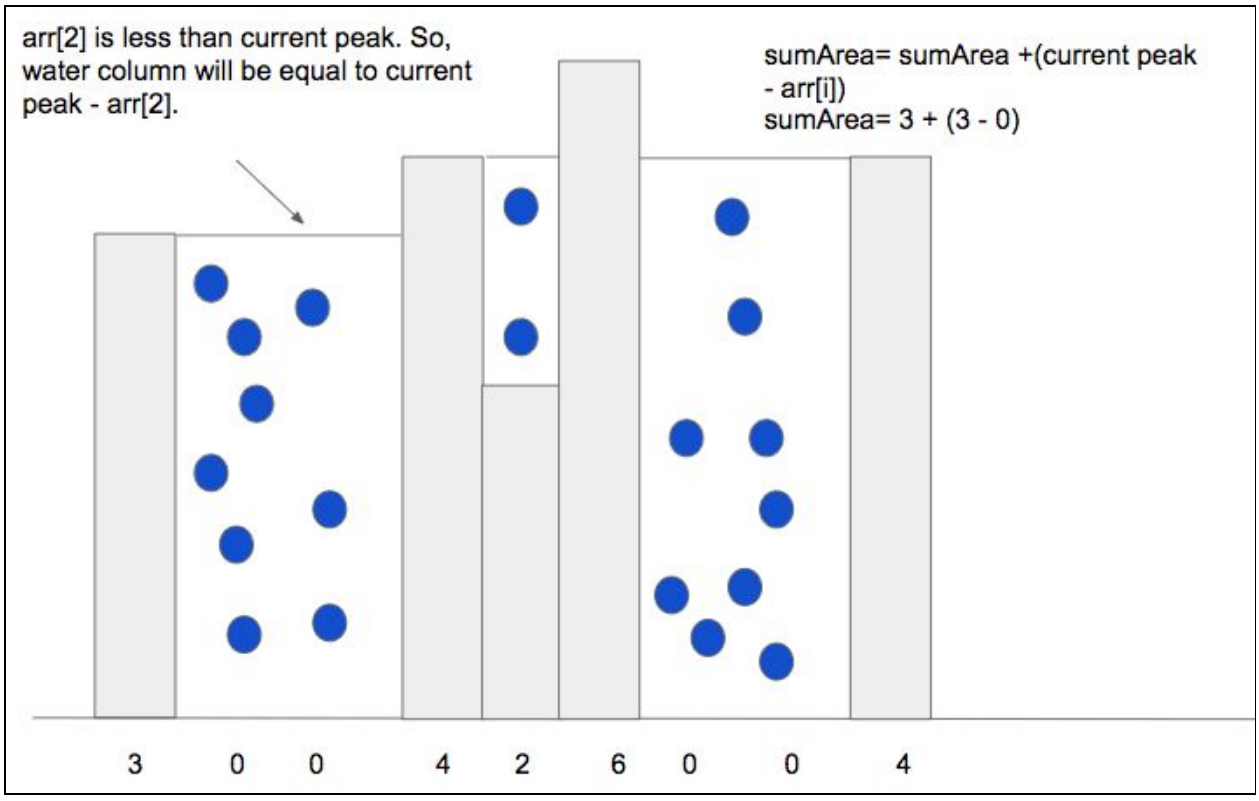


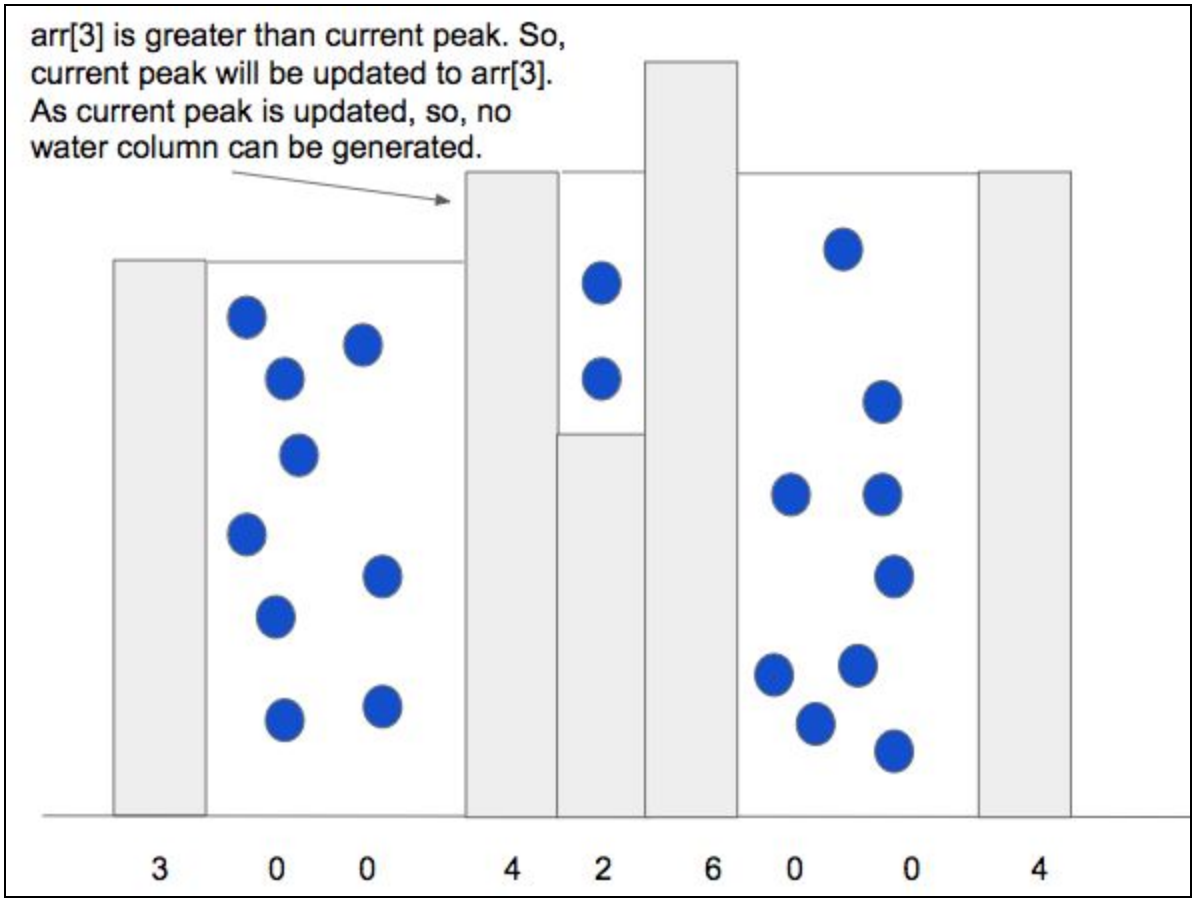
2. Since, we have established that we need a supporting tower of height greater than or equal to height h, therefore, we will need two traversals. First traversal will be left to right, from the 0th index to index of tower of maximum height. We will traverse till tower of maximum height because there is no supporting tower for tower with maximum height, in left to right direction.
   Second traversal will be from rightmost index to left, from array.length-1 to index of tower of maximum height. We will traverse till tower of maximum height because there is no supporting tower for tower with maximum height, in right to left direction.
3. We will sum up water collected by each column to get the total collected rain water.

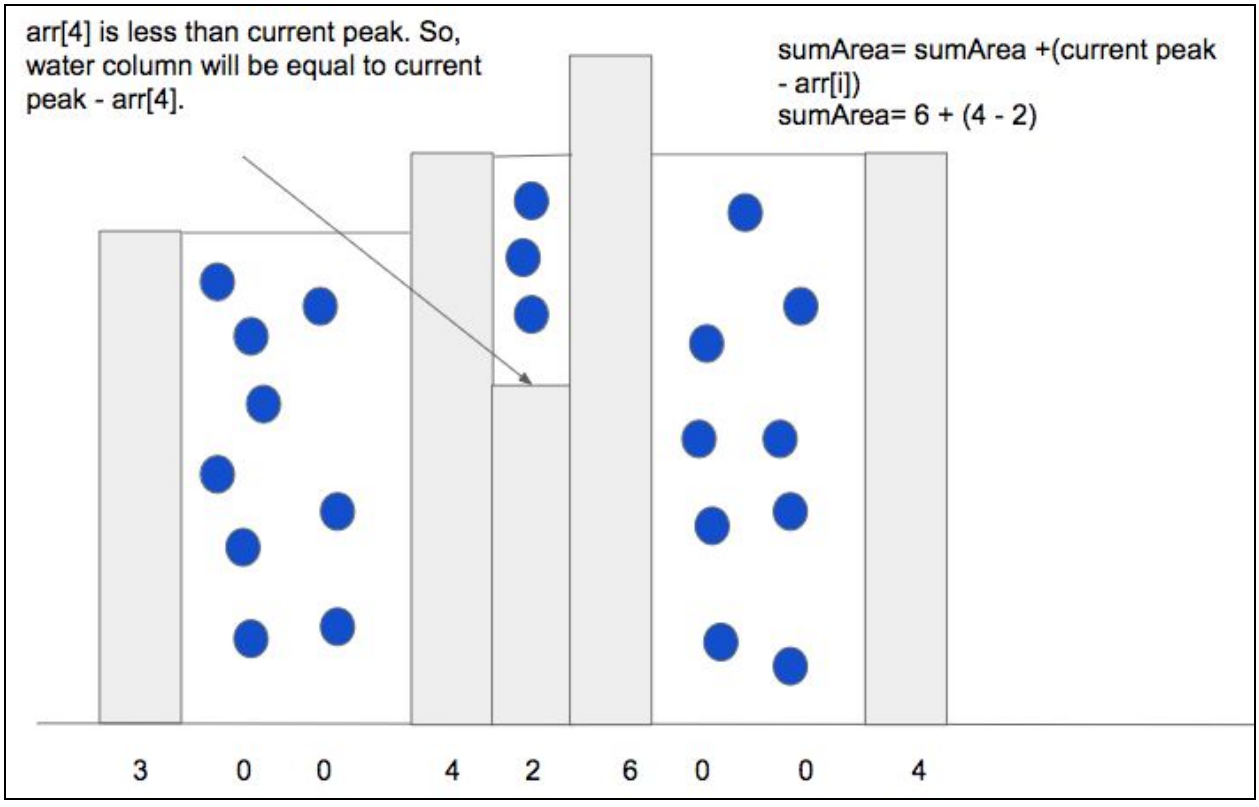
Let us apply these insights to a test case:
Let the array of heights be: 3 0 0 4 2 6 0 0 4

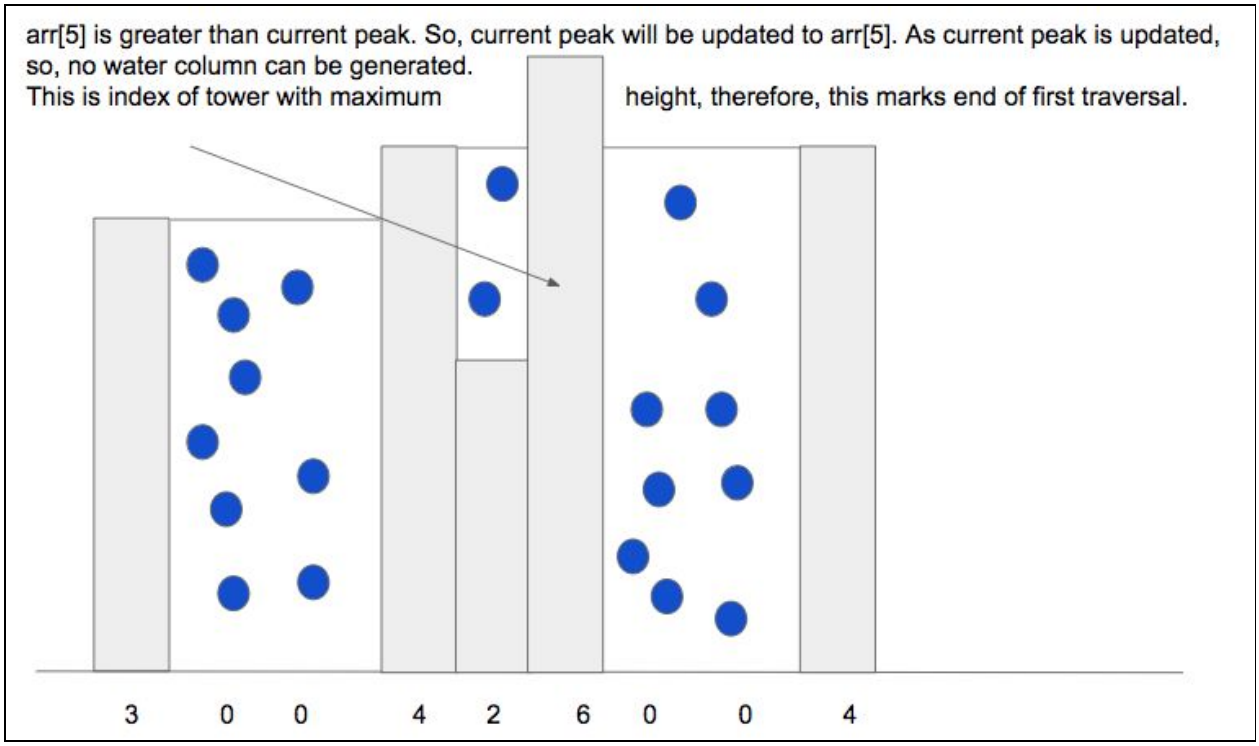

In the first traversal, we will start with index 0 and take current peak as arr[0]. As current peak is updated, so, no water column can be generated.
3
0
0
4
2
6
0
0
4

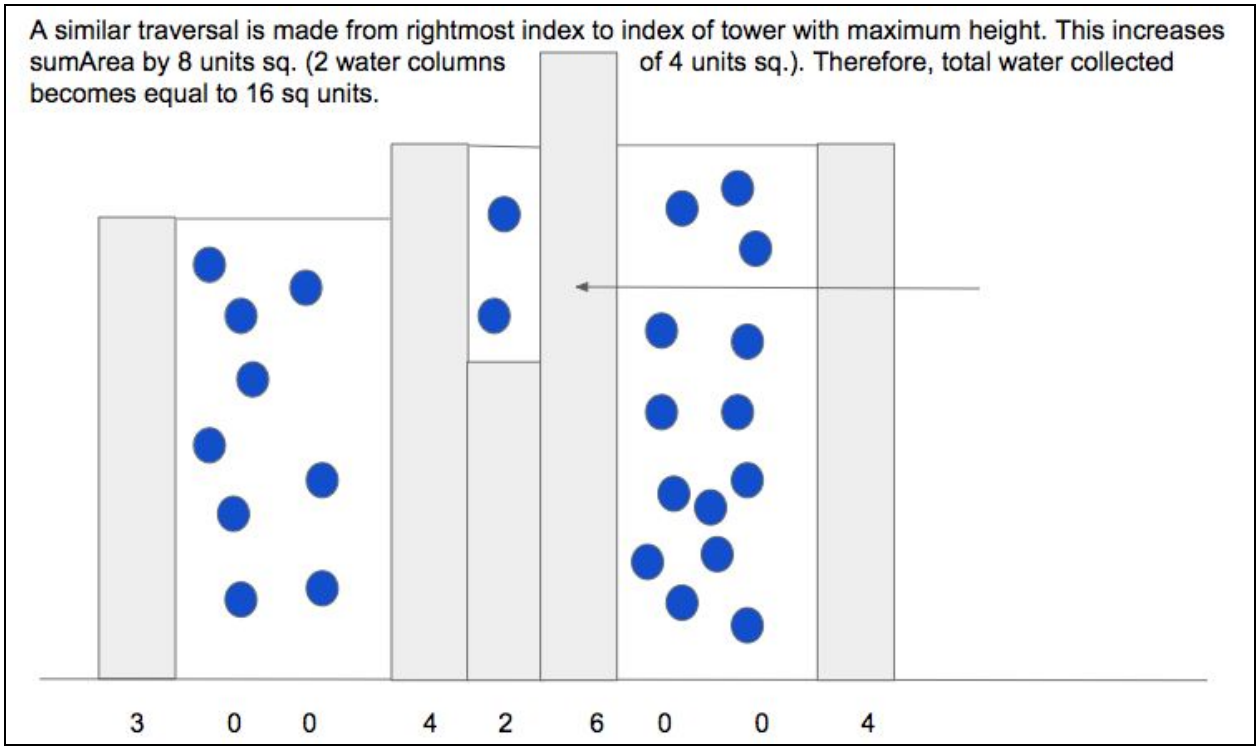

arr[1] is less than current peak. So, water column will be equal to current peak - arr[1].
sumArea= sumArea +(current peak - arr[i])
sumArea= 0 + (3 - 0)
3 0 0 4 2 6 0 0 4


arr[2] is less than current peak. So, water column will be equal to current peak - arr[2].
sumArea= sumArea +(current peak - arr[i])
sumArea= 3 + (3 - 0)
3 0 0 4 2 6 0 0 4


arr[3] is greater than current peak. So, current peak will be updated to arr[3]. As current peak is updated, so, no water column can be generated.
3 0 0 4 2 6 0 0 4


arr[4] is less than current peak. So, water column will be equal to current peak - arr[4].
sumArea= sumArea +(current peak - arr[i])
sumArea= 6 + (4 - 2)
3 0 0 4 2 6 0 0 4





Pseudocode of the solution:

```
// Loop to find index of tower with maximum height
peakValue = Integer.MIN_VALUE
```

```
peakValueIdx = -1
Loop (i = 0; i < arr.length; i++)
        if (arr[i] > peakValue)
                peakValue = arr[i];
                peakValueIdx = i;


// Traversal from leftmost index to index of tower with maximum height
sumArea = 0
peakSoFar = Integer.MIN_VALUE
Loop (i = 0; i <= peakValueIdx; i++)
        if (arr[i] >= peakSoFar)
                peakSoFar = arr[i]  // updating the peak so far
         else
                sumArea+=(peakSoFar-arr[i])

// Traversal from rightmost index to index of tower with maximum height
peakSoFar = Integer.MIN_VALUE
Loop (i = arr.length - 1; i >= peakValueIdx; i--)
        if (arr[i] >= peakSoFar)
                peakSoFar = arr[i];
        else
                sumArea+=(peakSoFar-arr[i])


Print sumArea
```