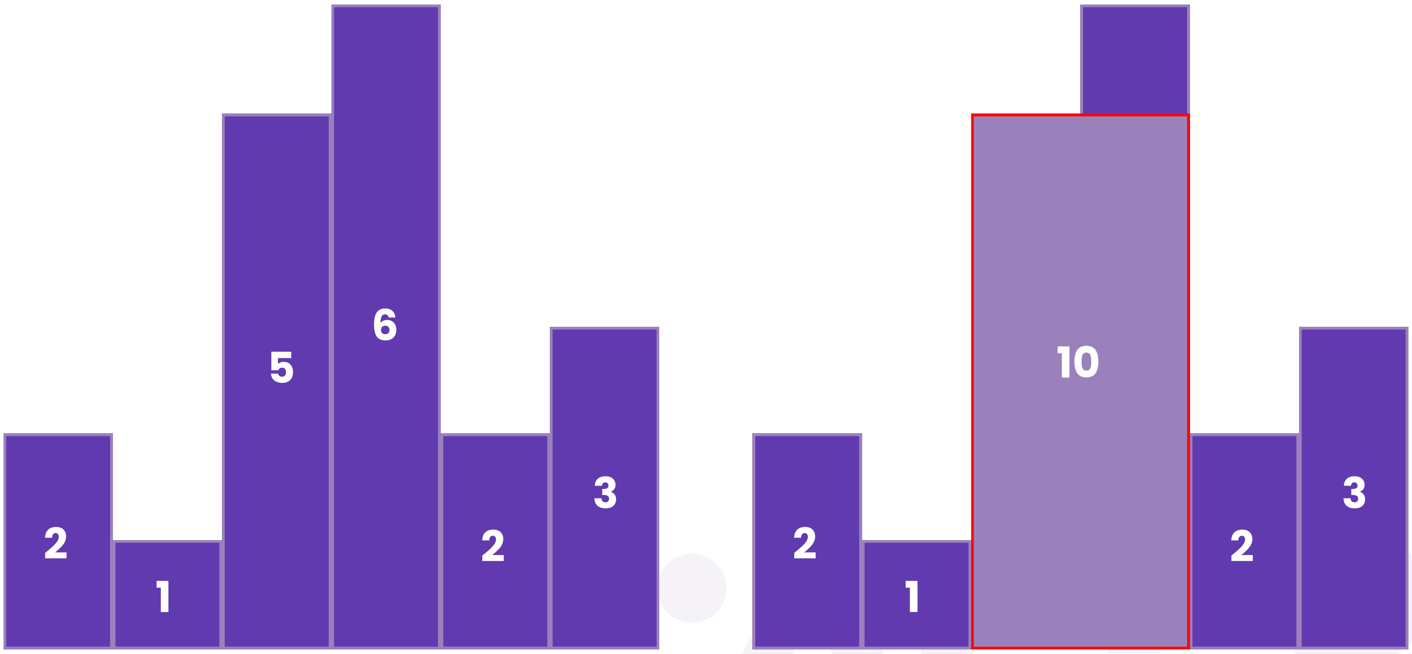
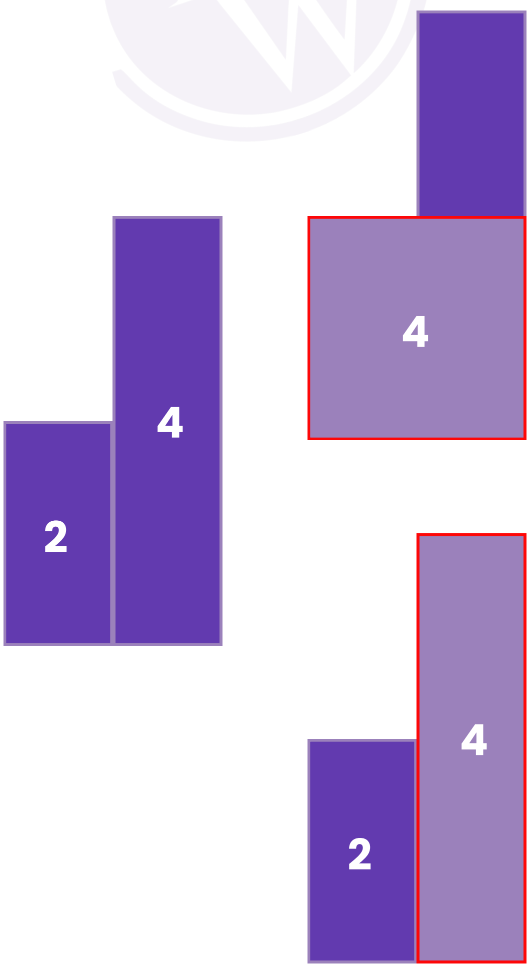
Q3. Given an array of integer heights representing the histogram's bar height where the width of each bar is 1, return the area of the largest rectangle in the histogram.

Example 1:



Input: heights = [2,1,5,6,2,3]
Output: 10
Explanation: The above is a histogram where the width of each bar is 1.
The largest rectangle is shown in the red area, which has an area = 10 units.

Example 2:



Input: heights = [2,4]
Output: 4

Solution :

Code : LP_Code3.java

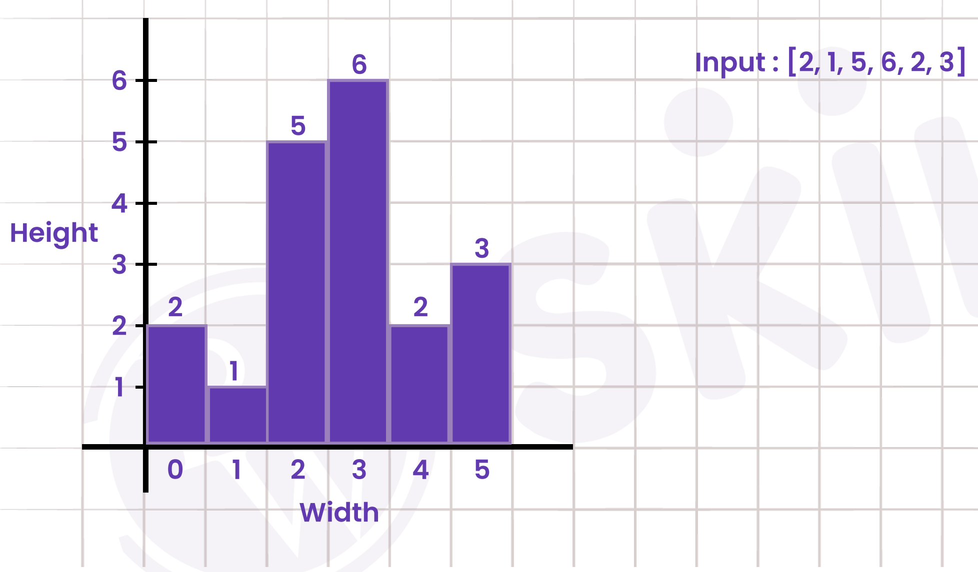
Output :

```
The maximum area is : 10
```

Approach :

The main concept here is that we need to find the minimum element to the right and to the left for any particular element so that we can use this difference as the width of the rectangle.

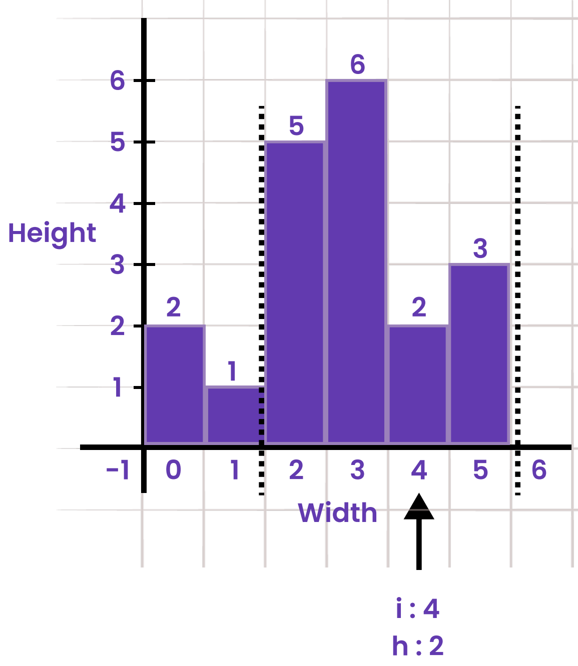
- Let's understand this problem with an example;
- Input: heights = [2,1,5,6,2,3]



- Let's understand how we calculate this, let's take index 4 height is 2. If we go to the left and see where the height is less then 2 and i.e. at index 1. So, that becomes the left Boundary. Similarly if we go to the right and see where the height is less then 2 & i.e. at index 6 [which is end of the array].

- Now we have left & right boundary the area could be find out by height * width. And width will get by (right - left - 1) in case of 4 that will be 6 - 1 - 1 = 4. And we know the height already i.e. 2.

- The Area we get is 2 * 4 i.e. 8



Basic Idea:

At each index find a left & right Boundary; between which the height of rectangle is atleast equal to current height

For index: 4
height=2

Left boundary= 1; right boundary = 6
width (r-1-1)=6-1-1=4
area = height *width = 2*4=8

- The main idea here would be how to find the left boundary & right boundary for every index. The brute way is using an ARRAY
- **For left boundary array:**

- Start with index 1, (left[0] = -1)
- For each index -> go to left & find the nearest index where height [index] < height[curr]
- Start with index 1,
- Eg : Index 1 : left[1] = -1

- **For right boundary array:**

- Start with index n - 2 right[n - 1] = n
- For each index -> go to right & find the nearest index where height[index] < height[curr]
- Start with index n - 2
- Eg : Index 2 : right[2] = 4