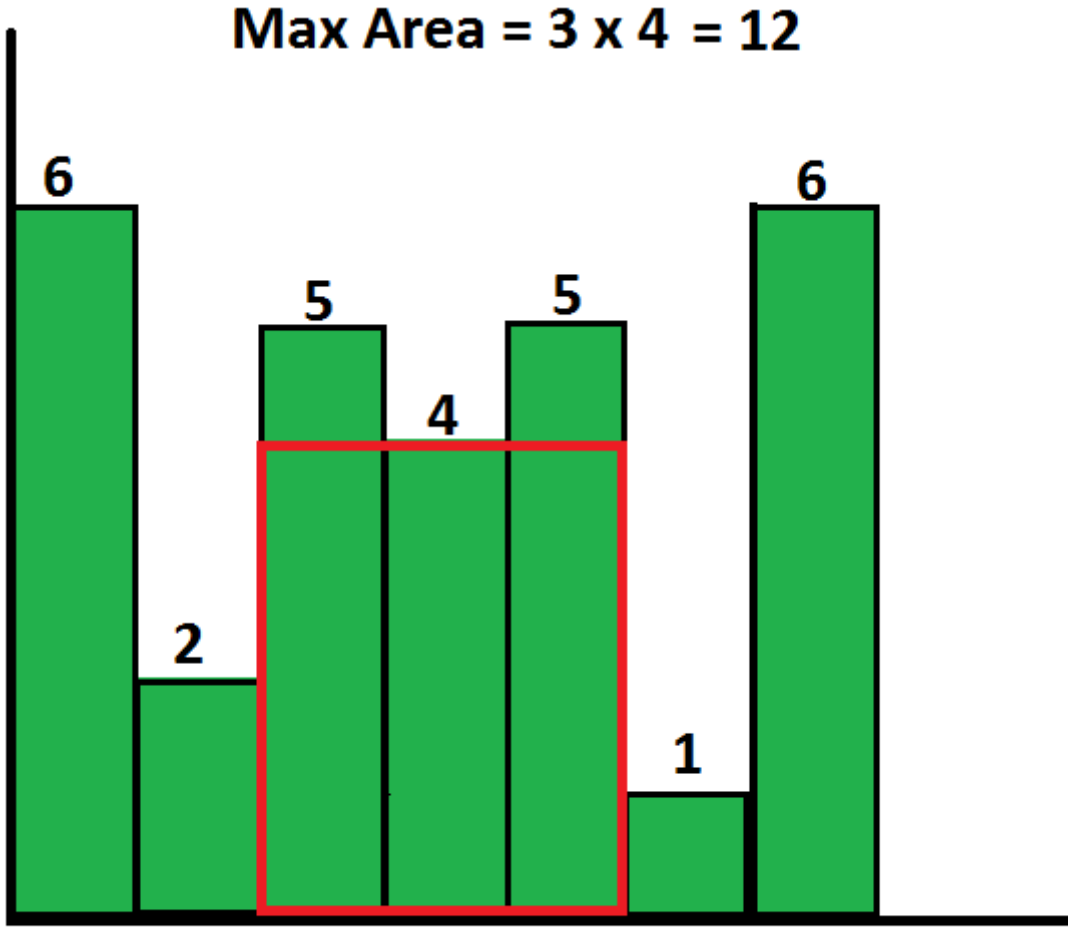
Maximum Rectangular Area in a Histogram

Find the largest rectangular area possible in a given histogram where the largest rectangle can be made of a number of contiguous bars. For simplicity, assume that all bars have the same width and the width is **1 unit**.

Example 1:

Input: N = 7
arr[] = {6,2,5,4,5,1,6}
Output: 12 **Explanation:**



Example 2:

Input: N = 8
arr[] = {7 2 8 9 1 3 6 5}
Output: 16 **Explanation:** Maximum size of the histogram will be 8 and there will be 2 consecutive histogram. And hence the area of the histogram will be 8x2 = 16.

Your Task:
The task is to complete the function **getMaxArea**() which takes the array arr[] and its size N as inputs and finds the largest rectangular area possible and **returns** the answer.

Expected Time Complxity : O(N)
Expected Auxilliary Space : O(N)

Constraints:
1 <= N <= 106
1 <= arr[i] <= 1012

```
def getMaxArea(self,histogram):
        #code here
        left = self.nextSmallerLeft(histogram)
        right = self.nextSmallerRight(histogram)
        maxArea = -1
        for i in range(len(histogram)):
            width = right[i]-left[i]-1
            area = histogram[i]*width
            maxArea = max(maxArea,area)
        return maxArea

    def nextSmallerRight(self,arr):
        stack = [len(arr)-1]
        ans = [0]*len(arr)
        ans[len(arr)-1] = len(arr)
        for i in range(len(arr)-2,-1,-1):
            while len(stack) and arr[i]<=arr[stack[-1]]:
                stack.pop()
            if len(stack)==0:
                ans[i] = len(arr)
            else:
                ans[i] = stack[-1]
            stack.append(i)
        return ans
    def nextSmallerLeft(self,arr):
        stack = [0]
        ans = [0]*len(arr)
        ans[0] = -1
        for i in range(1,len(arr)):
            while len(stack) and arr[i]<=arr[stack[-1]]:
                stack.pop()
            if len(stack)==0:
```

```
                ans[i] = -1
            else:
                ans[i] = stack[-1]
            stack.append(i)
        return ans
```