# MAXIMUM AREA OF 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.

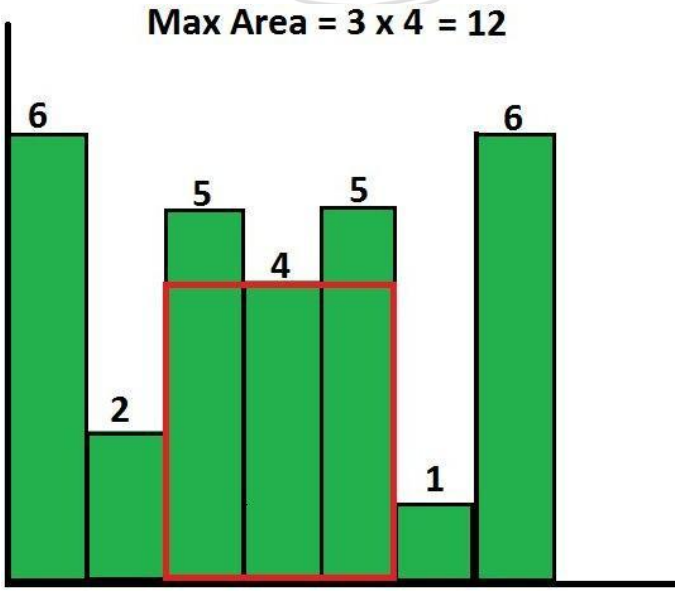
**Input:**

The first line contains an integer 'T' denoting the total number of test cases. T test-cases follow. In each test case, the first line contains an integer 'N' denoting the size of the array. The second line contains N space-separated integers $A_1$, $A_2$, ..., $A_N$ denoting the elements of the array. The elements of the array represent the height of the bars.

**Output:**

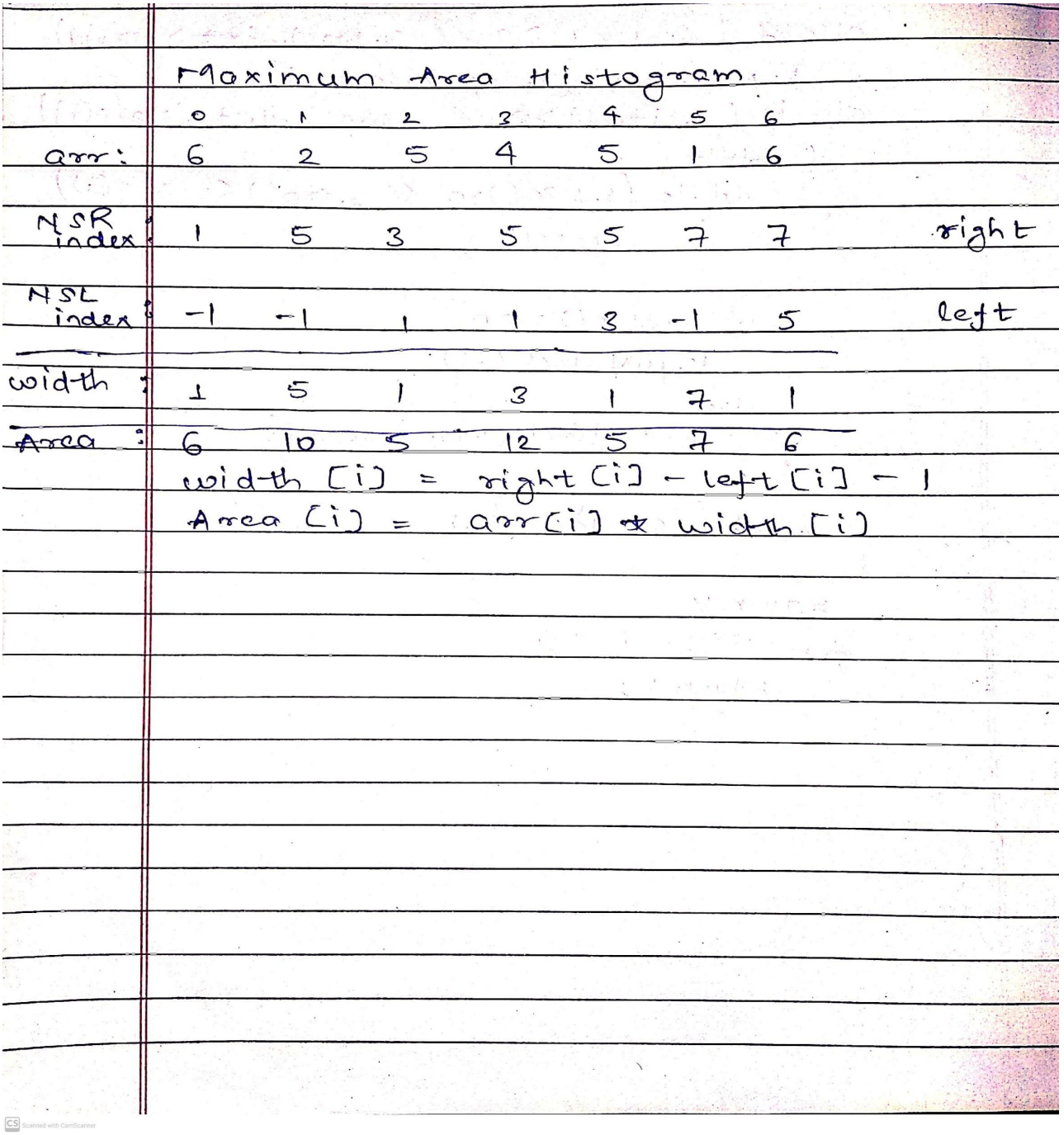
For each test-case, in a separate line, the maximum rectangular area possible from the contiguous bars.

Question link :
https://practice.geeksforgeeks.org/problems/maximum-rectangular-area-in histogram/0



**ALGORITHM :**

1. Create a stack and push all the elements into the stack
2. For each element , find the index of nearest smaller to left , and nearest smaller to right .
3. Find the area considering the length of the rectangle as ngr - ngl -1 and width as the element(height of that bar).
4. Find the max area.



CODE :

```
class Solution
{
  public:
  long long getMaxArea(long long arr[], int n)
    {
    vector<int> left,right;
    stack<pair<int,int>> s1,s2;
    int pseudo_index =-1;
    int pseudo_index1 =n;
    for (int i=0;i<n;i++)
    {
      if (s1.size()==0)
      {
        left.push_back(pseudo_index);
      }
      else if (s1.size()>0 && s1.top().first<arr[i])
      {
        left.push_back(s1.top().second);
      }
      else if (s1.size()>0 && s1.top().first>=arr[i])
      {
         while(s1.size()>0 && s1.top().first>=arr[i])
         {
           s1.pop();
         }
        if (s1.size()==0)
        {
          left.push_back(pseudo_index);
        }
        else
        {
          left.push_back(s1.top().second);
        }
      }
      s1.push({arr[i],i});
    }
```

```
        for (int i=n-1;i>=0;i--)
        {
          if (s2.size()==0)
          {
            right.push_back(pseudo_index1);
          }
          else if (s2.size()>0 && s2.top().first<arr[i])
          {
            right.push_back(s2.top().second);
          }
          else if (s2.size()>0 && s2.top().first >= arr[i])
          {
            while(s2.size()>0 && s2.top().first >= arr[i])
            {
              s2.pop();
            }
            if (s2.size()==0)
            {
              right.push_back(pseudo_index1);
            }
            else
            {
              right.push_back(s2.top().second);
            }
          }
          s2.push({arr[i],i});
        }
```

```
      reverse(right.begin(),right.end());
      long long m=INT_MIN;
      for (long long i=0;i<n;i++)
      {
        m=max(m,(right[i]-left[i]-1)*arr[i]); // taking max after finding area
      }
      return m;
    }
};
```

Question for practice :

1. https://practice.geeksforgeeks.org/problems/stock-span-problem/0
2. https://practice.geeksforgeeks.org/problems/smallest-number-on-left/0
3. https://practice.geeksforgeeks.org/problems/save-gotham/0
4. https://practice.geeksforgeeks.org/problems/next-larger-element/0
5. https://practice.geeksforgeeks.org/problems/delete-array-elements - which-are-smaller-than-next-or-become-smaller/0