```
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
using namespace std;

#define mx              100010
#define inf              0x3f3f3f3f
#define mod             1000000007
#define PI              2*acos(0.0)
#define E               2.71828182845904523536
#define ll              long long int
#define ull             unsigned long long int
#define pii             pair<int,int>
#define pll             pair<ll,ll>
#define valid(tx,ty)    tx>=0&&tx<r&&ty>=0&&ty<c
#define mem(arr,val)    memset(arr,val,sizeof(arr))
#define fast            ios_base::sync_with_stdio(false),cin.tie(NULL)
string tostr(int n)     {stringstream rr;rr<<n;return rr.str();}
const int fx[]={+0,+0,+1,-1,-1,+1,-1,+1};
const int fy[]={-1,+1,+0,+0,+1,+1,-1,-1};
bool bitcheck(ll p,ll pos){return (bool)(p&(1<<pos));}
template <typename T> T biton(T p,T pos){return p=p|(1<<pos);}
template <typename T> T bitoff(T p,T pos){return p=p&~(1<<pos);}
template <typename T> T POW(T b,T p) {T Ans=1; while(p){if(p&1)Ans=(Ans*b);b=(b*b);p>>=1;}return Ans;}
template <typename T> T BigMod(T b,T p,T Mod) {T Ans=1; while(p){if(p&1)Ans=(Ans*b)%Mod;b=(b*b)%Mod;p>>=1;}return Ans;}
template <typename T> T ModInverse(T p,T Mod) {return BigMod(p,Mod-2,Mod);}
template <typename T> using ordered_set = tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update>;
//**************************************************************************************************************************//

int main(){
//    freopen("Input.txt","r",stdin); freopen("Output.txt","w",stdout);

    return 0;
}
```

```
int main(){                   **HISTOGRAM**
//    freopen("Input.txt","r",stdin); freopen("Output.txt","w",stdout);
    int t;
    scanf("%d",&t);
    for(int cs=1;cs<=t;cs++){
        int n;
        scanf("%d",&n);
        for(int i=0;i<n;i++) scanf("%d",&arr[i]);
        int top,i=0,sum=0,max_area=0;
        stack<int>st;
        st.push(0);
        for(i=0;i<n;){
            if(st.empty()||arr[st.top()]<=arr[i]) st.push(i++);
            else{
                top=st.top();
                st.pop();
                if(st.empty()){
                    sum=arr[top]*i;
                }
                else{
                    sum=arr[top]*(i-st.top()-1);
                }
                max_area=max(sum,max_area);
            }
        }
        while(!st.empty()){
            top=st.top();
            st.pop();
            if(st.empty()) sum=arr[top]*i;
            else sum=arr[top]*(i-st.top()-1);
            max_area=max(sum,max_area);
        }
        printf("Case %d: %d\n",cs,max_area);
    }
    return 0;
}
```