Maximum area histogram

We need to find a histogram with max area. Left me agr koi chota bar ho to aage ni bdh skte similarly for right.

Lkn agr greater h to we can extend isko use krke width find krni h and jis hist k liye find kri h width uski height se multiply krke ans aajayega.

To approach this problem we make use of next smaller left and next smaller right. width find krne k liye ns[i]-ps[i]-1 krenge then multiply by height usme se max choose krlenge.

```
class Solution {
public:
    void nextSmallerLeft(vector<int>& ps,vector<int>& heights)
    {
        stack<pair<int,int>> s;
        for(int i=0;i<heights.size();i++)
        {
            if(s.empty())
            {
                ps.push_back(-1);
            }
            else
            {
                if(s.top().first<heights[i])
                {
                    ps.push_back(s.top().second);
                }
                else
                {
                    while(!s.empty() && s.top().first >= heights[i])
                    {
                        s.pop();
                    }
                    if(s.empty())
                    {
                        ps.push_back(-1);
                    }
                    else
                    {
                        ps.push_back(s.top().second);
                    }
                }
            }
            s.push({heights[i],i});
        }
```

```
    }
    void nextSmallerRight(vector<int>& ns,vector<int>& heights)
    {
        stack<pair<int,int>> s;
        for(int i=heights.size()-1;i>=0;i--)
        {
            if(s.empty())
            {
                ns.push_back(heights.size());
            }
            else
            {
                if(s.top().first<heights[i])
                {
                    ns.push_back(s.top().second);
                }
                else
                {
                    while(!s.empty()&&s.top().first>=heights[i])
                    {
                        s.pop();
                    }
                    if(s.empty())
                    {
                        ns.push_back(heights.size());
                    }
                    else
                    {
                        ns.push_back(s.top().second);
                    }
                }
            }
            s.push({heights[i],i});
        }

        reverse(ns.begin(),ns.end());
    }
    int largestRectangleArea(vector<int>& heights) {
       int n=heights.size();
        vector<int> ns;
        vector<int> ps;
        nextSmallerLeft(ps,heights);
        nextSmallerRight(ns,heights);
        int maxm=INT_MIN;

        for(int i=0;i<n;i++)
        {
            maxm=max(maxm,(ns[i]-ps[i]-1)*heights[i]);
        }
        return maxm;
    }
};
```