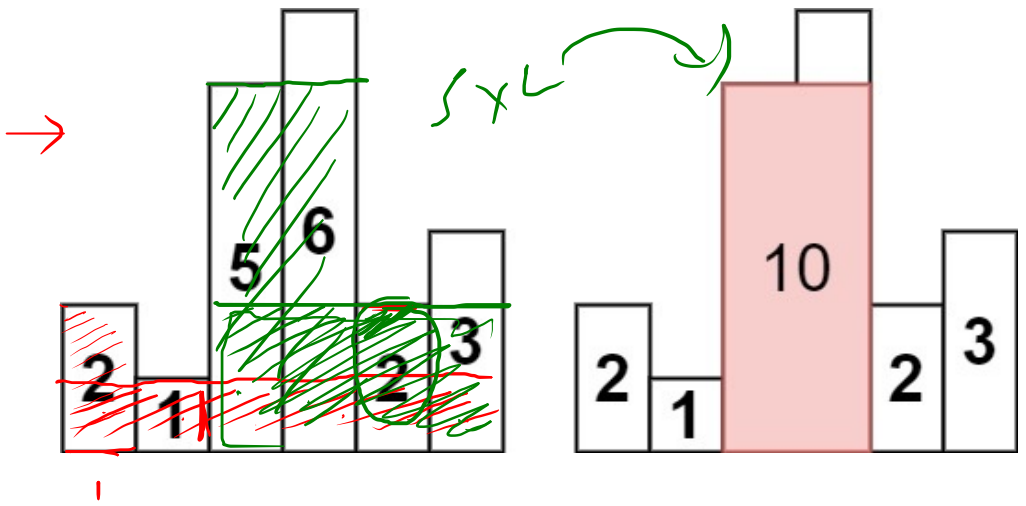
Largest Rectangle in Histogram

⇒ Find the largest Area

I/p :



HARD

Code

2 x 1 = 2

1 x 6 = 6

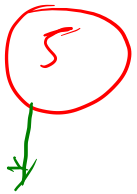
2 x 4 = 8

I/p

data = [2, 5, 1]

1 x 3 = 3

o/p



2 x 2 = 4

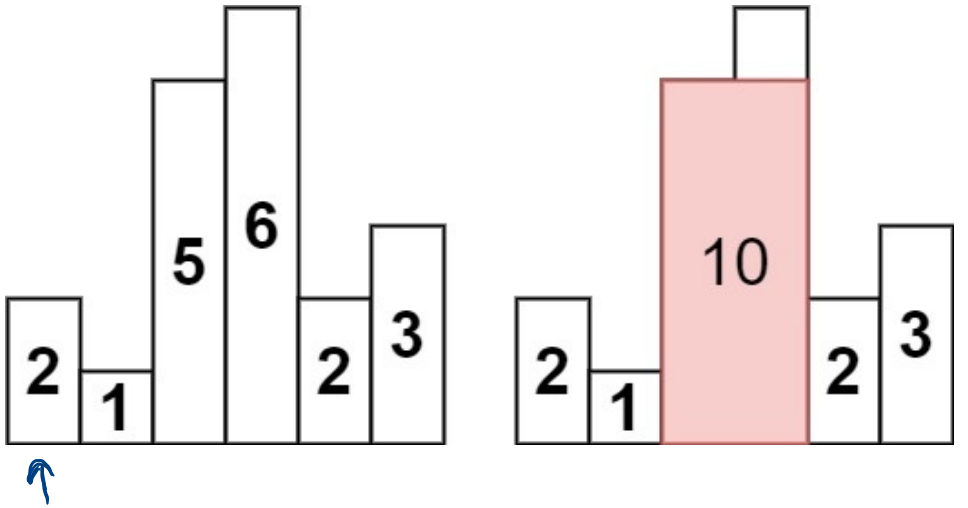
Soln

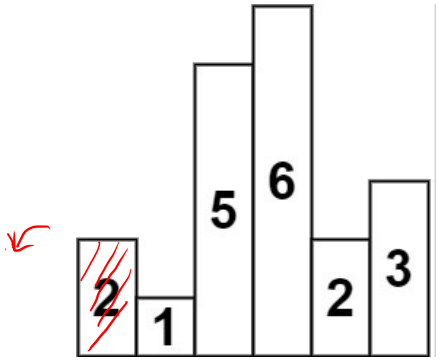
Key point

=> (we consider every bar is smaller)

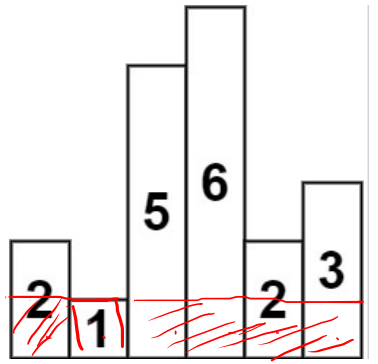
Solve

I/p:

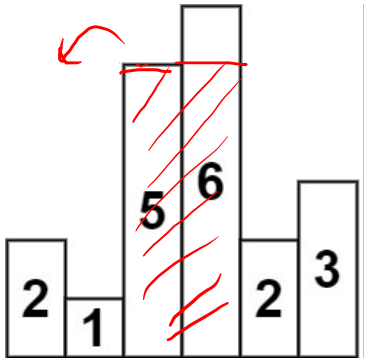




2

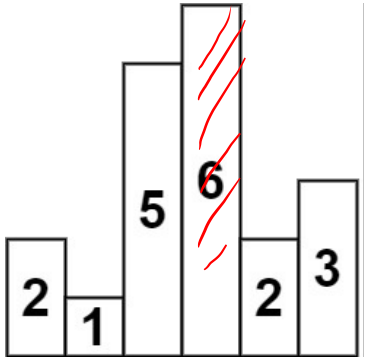


1 x 6 = 6

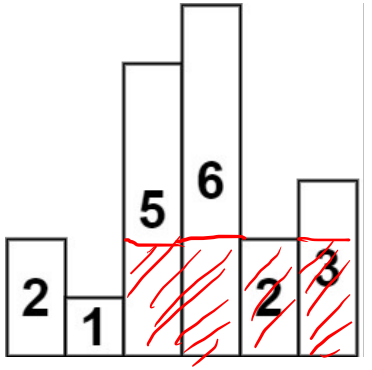


5 x 2 = 10

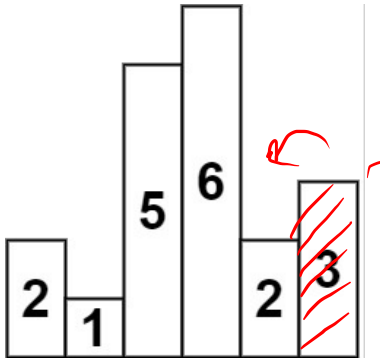
Ans



6 x 1 = 6



2 x 4 = 8



3 x 1 = 5

I/p: { 6, 2, 5, 4, 1, 5, 6 }

Initially ans = 0

i = 0 ; curr = 6 ans = 6

i = 1 ; curr = 8 , ans = 8

i = 2 ; curr = 5

i = 3 ; curr = 8

i = 4 ; curr = 7

i = 5 ; curr = 10 , ans = 10

i = 6 ; curr = 6

```
int Area( int arr[], int n )
{
    int ans = 0;
    for(int i=0 ; i<n ; i++)
    {
        int curr = arr[i];
        for(int j= i-1 ; j>=0 ; j--)
        {   if( arr[j] >= arr[i])
                curr += arr[i];
            else
                break;
        }
        for(int j= i+1 ; j<n; j++)
        {   if( arr[j] >= arr[i])
                curr += arr[i];
            else
                break;
        }
        ans = max(ans, curr);
    }
    return ans;
}
```

NAIVE

left

right

Dry Run

I/p: { 6, 2, 5, 4, 1, 5, 6 }

Initially ru = 0

i = 0 ; curr = 6 ; ru = 6

i = 1 ; curr = 8 ; ru = 8

i = 2 ; curr = 5

i = 3 ; curr = 8

i = 4 ; curr = 7

i = 5 ; curr = 10 ; ru = 10

i = 6 ; curr = 6

Ans = ru = 10

2222

44

1111011

```
int Area( int arr[], int n )
{
    int ru = 0;
    for(int i=0 ; i<n ; i++)
    {
        int curr = arr[i];
        for(int j=i-1; j>=0 ; j--)
        {
            if( arr[j] >= arr[i] )
                curr += arr[i];
            else
                break;
        }
        for(int j=i+1; j<n; j++)
        {
            if( arr[j] >= arr[i] )
                curr += arr[i];
            else
                break;
        }
        ru = max(ru, curr);
    }
    return ru;
}
```

$O(N^2)$ ✗

$O(N^2)$

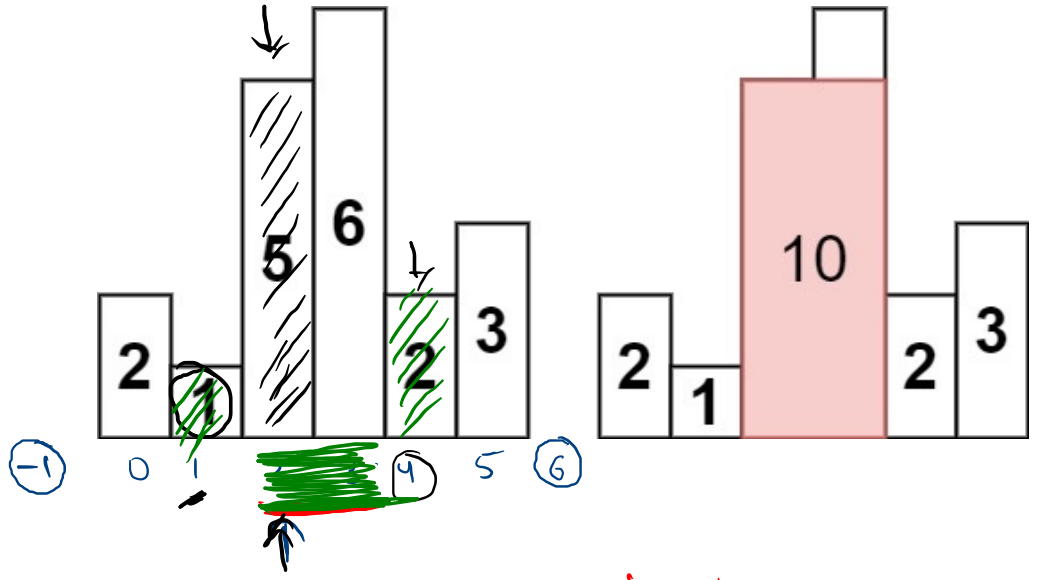
curr area

$O(N)$ ✗

$O(N)$

Efficient solution

O(N)



PS = Previous Smaller

NS = next Smaller

index

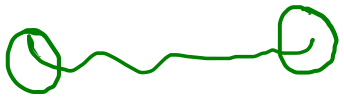
PS = 1

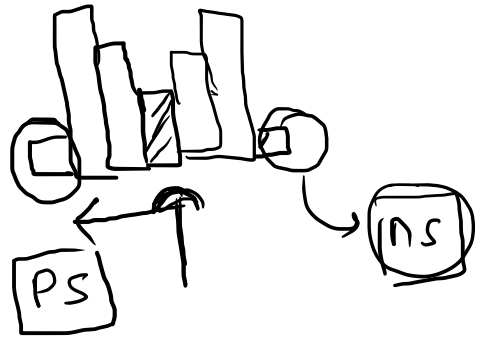
NS = 4

$\{4-1\} - 1$

$\Rightarrow 3-1 = 2 \times 5$

$= 10$



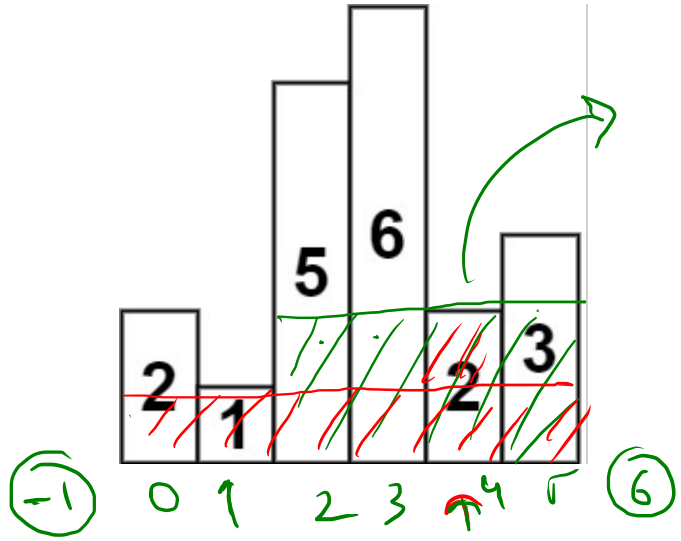


↳ Height $\times$ (Total no. of bars)

$\Rightarrow (4-1) - 1$

$\Rightarrow (NS - PS - 1)$] $\Rightarrow 2 \times$ height

$= 2 \times 5 = 10$



ns - ps - 1 × height

ps = 1

ns = 6

6 - 1 - 1

⇒ 4 × 2 = 8

ps = -1

ns = 6

6 - (-1) - 1

⇒ 6 × 1

Index 0 1 2 3 4 5 6

I/P: { 6, 2, 5, 4, 1, 5, 6 }

PS[]: (DO it yourself)

NS[]:

6

① Initilise `ru = 0`

② Find previous smaller and next smaller of every element

③ Now, for every element `height[i]`

{

int `area` = (ns[i] - ps[i] - 1) × height[i]

`ru` = max(ru, area);

}

↳ Total no of bars

④ return `ru`;

O(N) solution

Index: 0 1 2 3 4 5 6

I/P: { 6, 2, 5, 4, 1, 5, 6 } 7

-1

⇒ ps[]: -1 -1 1 1 -1 4 5

⇒ ns[]: 1 4 3 4 7 7 7

index

match

① Initilise ra = 0;

② Find previous smaller and next smaller of every element

③ Now, for every element height[i]

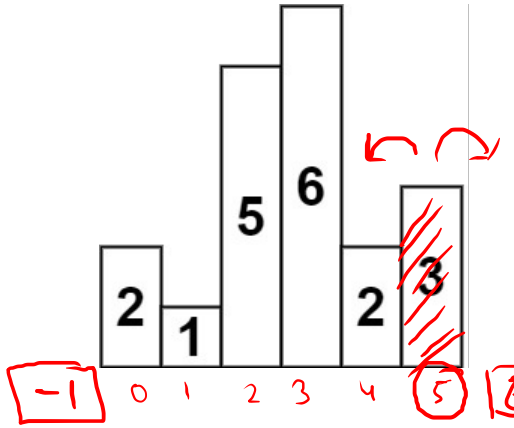
{

int area = (ns[i] - ps[i] - 1) * height[i];

ra = max(ra, area);

}

④ return ra;



6-4-1

⇒ 1

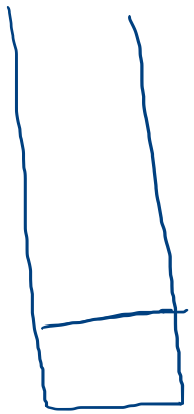
ps: 4

ns: 6

Code of Previous Smaller

I/p: { 6, 2, 5, 4, 1, 5, 6 }
0 1 2 3 4 5 6

o/p:



our mission is to find smaller element

index

dt

2o-W

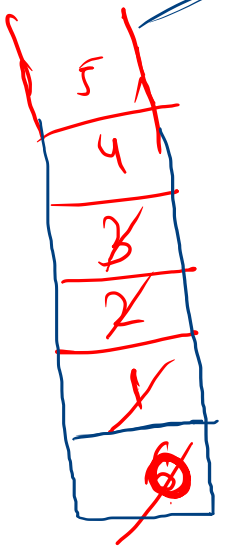
n

Code C++

```
vector<int> PS ( int arr[], int n)
{
    vector<int> ans;
    stack<int> st;

    return ans;
}
```

Code of Previous Smaller



I/p: { 6, 2, 5, 4, 1, 5, 6 }

PS[] O/p: -1 -1 1 1 -1 4 5

Ans

PS()

PS()

arr[1] >= arr[2]

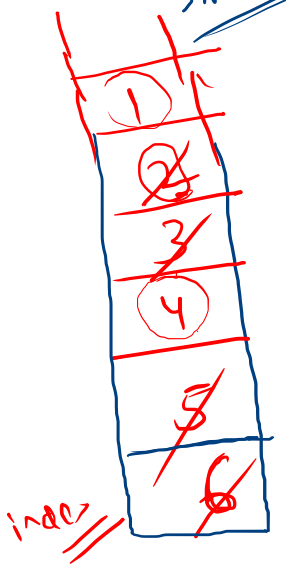
2 >= 5 X

our mission is to find smaller element

```
vector<int> PS ( int arr[], int n )
{
    vector<int> ans;
    stack<int> st;

    for( int i=0; i<n; i++ )
    {
        while( st.empty() == false and
               arr[st.top()] >= arr[i] )
        {
            st.pop();
        }
        int ele = ( st.empty() ) ? -1 : (st.top())
        ans.push_back(ele);
        st.push(i);
    }
    return ans;
}
```

Code of next Smaller



I/p: { 6, 2, 5, 4, 1, 5, 6 } 7

0 1 2 3 4 5 6

o/p: 1 4 3 4 7 7 7

Ans

PS
NS

```
vector<int> ns ( int arr[], int n)
{
    vector<int> ans;
    stack<int> st;

    for( int i = n-1 ; i>=0 ; i--)
    {
        while( st.empty() == false and
               arr[st.top()] >= arr[i] )
        {
            st.pop();
        }
        int ele = ( st.empty() ) ? n : (st.top())
        ans.push_back(ele);
        st.push(i);
    }
    return ans;
}
```