Amazon | MS | Twitter | Google Code Jam | FB | LinkedIn | Uber | Snapchat | Airbnb | VMWare

Q. Nearest Smallest on left

Given an array of positive integers
For every index i, find the nearest ele on left side of i which is smaller than A[i].

A[] = { 4 5 2 10 8 2 }
-1 4 -1 2 2 -1

A[] = { 4 6 10 11 7 8 3 5 }
-1 4 6 10 6 7 -1 3

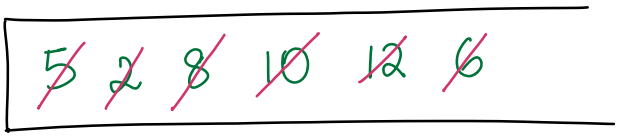
A[] = { 5 2 8 10 12 6 1 }
-1 -1 2 8 10 2 -1

Brute Force :

```
ans[N] = {-1}
for(i=0; i<N; i++){
    for(j=i-1; j>=0; j--){
        if(A[j] < A[i]){
            ans[i] = A[j]
            break
        }
    }
}
return ans
```

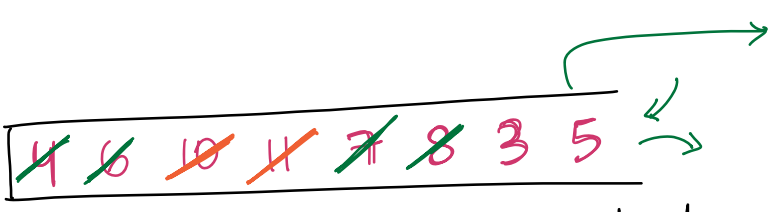
TC: $O(N^2)$
SC: $O(1)$

{ 5, 2, 8, 10, 12, 6, 1 }
-1 -1 2 8 10 2 -1



Any ele greater than 5 will obviously be greater than 2 thence, need not keep 5 in the box

{4, 6, 10, 11, 7, 8, 3, 5}
-1 4 6 10 6 7 -1 3



stack

(A[i] <= st.top())
st.pop()

(A[i] > st.top())
ans[i] = st.top()

Pseudocode

```
ans[N] = {-1}
stack<int> s;
for(i=0; i<N; i++) {
    While(s.size()>0 && A[i] <= st.top()){
        st.pop()
    }
    if(s.size()>0){ ans[i] = st.top()}
    s.push(A[i])
}
return ans
```

TC: O(N)
SC: O(N)

Q. Find the index of nearest smaller on left side

-1 ↳ NO answer present

Code

```
ans[N] = {-1}
stack<int> s;
for(i=0; i<N; i++) {
    While(s.size()>0 && A[i] <= A[s.top()] ){
        s.pop()
    }
    if (s.size()>0 ){ans[i] = s.top()}
    s.push( i )
}
return ans
```

0	1	2	3	4	5	6	7	8	9
4,	7,	-9,	5,	-6,	9,	7,	5,	2,	10
-1	0	-1	2	2	4	4	4	4	8

Storing the indices will also work for negative elements.

Que. Get distance of nearest smaller ele on left side

Que. Find nearest greater on left side

2	9	7	1	4	7	2	6	5
-1	-1	9	7	7	9	7	7	6

~~2~~ 9 ~~7~~ ~~1~~ ~~4~~ 7 ~~2~~ 6 5

A[i] < st.top()
ans[i] = st.top()

A[i] >= st.top()
st.pop()

Q. Find nearest smaller on right

2	9	7	1	4	7	2	6	5
1	7	1	-1	2	2	-1	5	-1

// Iterate from right

Q. Find nearest greater on right

for(i = n-1; i >= 0; i--)

2	9	7	1	4	7	2	6	5
9	-1	-1	4	7	-1	6	-1	-1

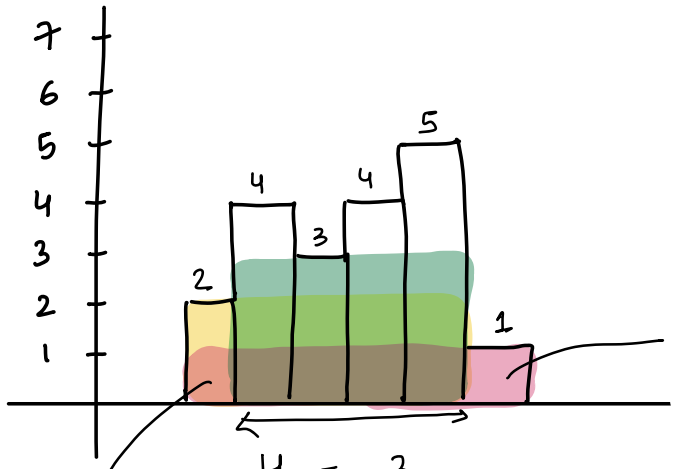
// Iterate from the right

Break → 10 : 40

10 : 33
↳ Black out.
↓

Adobe

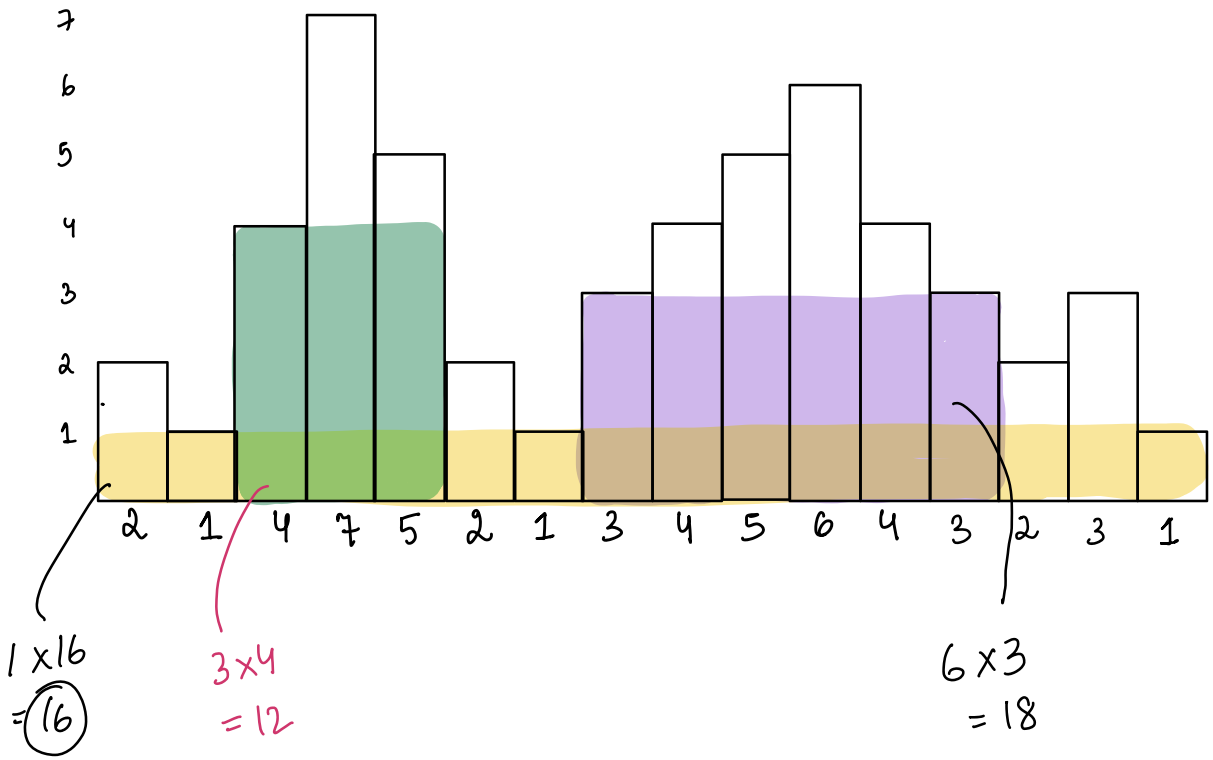
Qu. Given continuous blocks of histogram, find max rectangular area
↳ not exceeding histogram



H = 1
W = 6
area = 6

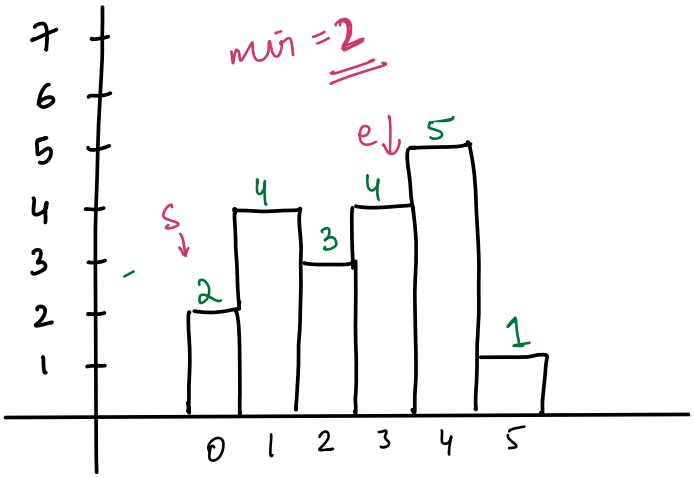
H = 3
W = 4
area = 12

H = 2
W = 5
area = 10



Brute Force

Given any two bars, can we find area b/w them?



$\Rightarrow \{2, 4, 3, 4, 5, 1\}$

s	e	H	W	area
0	0	2	1	2
0	1	2	2	4
0	2	2	3	6
0	3	2	4	8
⋮				

```
for ( s = 0 ; s < N ; s++ ) {
    m_e = A[s]
    for (e = s ; e < N ; e++ ) {
        m_e = min(m_e, A[e])
        // [s e]
        W = e - s + 1
        H = get min from [s e]   m_e
        area = W * H
        ans = max(ans, area)
    }
}
return ans
```

Carry Forward

$x = \min[i \quad j]$

$\min[i \quad j+1] = \min(x, j+1)$

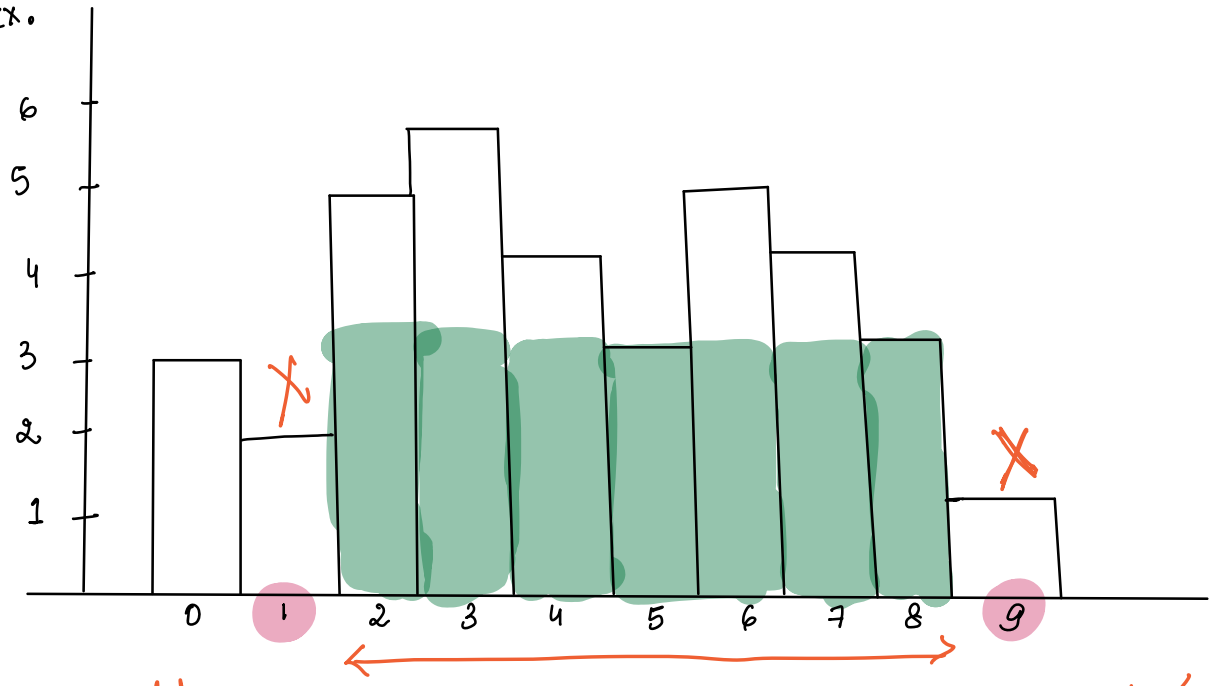
TC: $O(N^2)$

SC: $O(1)$

TC: $O(N^3)$

SC: $O(1)$

Ex.



height < A[i] on left

width = 9 - 1 - 1

(7)

Height = 3

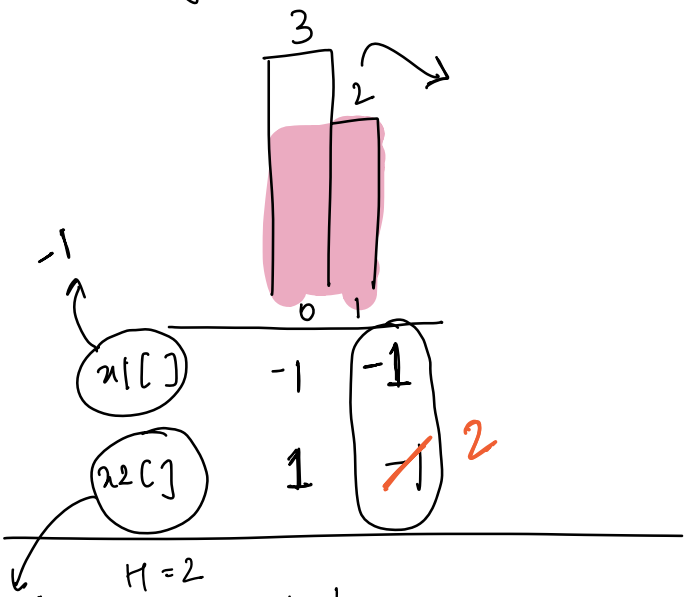
area = 21

height < A[i] right

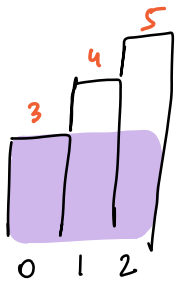
Code

```
for (i=0; i<N; i++){
    H = A[i]
    x1 = get the nearest smaller to A[i] on left
    x2 = get the nearest smaller to A[i] on right
    W = x2 - x1 - 1
    ans = max(ans, H * W)
}
```

Edge Case



H = 2
W = nr - nl - 1
= -1 - (-1) - 1
= -1 + 1 - 1
= -1



-1
3

H: 3
W: 3 - (-1) - 1
: 3 + 1 - 1 = 3
area = 9

H = 2
W = 2 - (-1) - 1
= 2 + 1 - 1
= 2

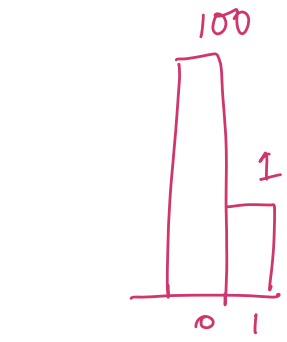
area = 2 x 2
= 4

nl[ ] = {-1} : // find nearest smaller index on left // O(N)
nr[ ] = {N} : // find " " index " right // O(N)

// O(N)

```
for (i = 0; i < N; i++){
    H = A[i]
    W = nr[i] - nl[i] - 1
    area = H * W
    ans = max(ans, area
}
return ans
```

TC : O(N)
SC : O(N)



nl[ ] -1 -1

nr[ ] 1 2

1 - (-1) - 1 = (1)

1 × 100 = 100

2 - (-1) - 1 = (2)

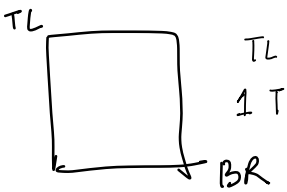
2 × 1 = (2)

Que: Given a matrix which contains 1 & 0, find max rectangular area which contains all 1s.

	0	1	2	3	4	5
0	1	1	0	1	0	1
1	0	1	1	1	1	1
2	1	1	1	1	1	0
3	1	0	1	0	0	1
4	1	1	0	0	1	1

Brute Force

consider all the submatrices



TL → N*M

1 TL → O(N*M) BR

Total submatrices

↳ $N^2 M^2$

1 TL → N * M BR

N * M TL → $N^2 * M^2$ submatrices.

If size == sum for a submatrix
Then could be a possible answer.

1 → O(1)

TC: $O(N^2M^2)$

	0	1	2	3	4	5
0	1	1	0	1	0	1
1	0	1	1	1	1	1
2	1	1	1	1	1	0
3	1	0	1	0	0	1
4	1	1	0	0	1	1

1	1	0	1	0	1
0	2	1	2	1	2
1	3	2	3	2	0
2	0	3	0	0	1
3	1	0	0	1	2

TC: O(N * M) + O(N * M)

SC: O(M * N) + O(M)

if 1D array SC: O(M) + O(M)

Problem Solving
↳ Double character Trouble's extended version

```
mat[N][M] = {0}
for(i=0; i<M; i++){
|  mat[0][i] = A[0][i]
}

for(i=1; i<N; i++){
|    for(j=0; j<M; j++){
|    |  if(A[i][j] != 0) mat[i][j] += mat[i-1][j]
|    |  else mat[i][j] = 0
|    }
}
for(i=0; i<N; i++){
|    maxRectangleArea ( mat [i])
}
```

TC: O(N*M)
SC: O(N*M)

```
ar[M] = {0}
for(i=0; i<M; i++){
    ar[i] = A[0][i]
}
ans = max(ans, maxRectangleArea(ar))
for(i=1; i<N; i++){
    for(j=0; j<M; j++){
        if(A[i][j] != 0) ar[j] += A[i][j]
        else { ar[j] = 0 }
    }
    ans = max(ans, maxRectangleArea(ar))
}
return ans
```

TC: O(N * M)
SC: O(M)