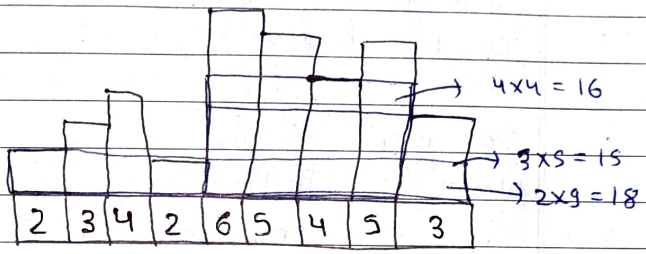
Date _____ Page _____

Day - 136

Stack - 4

* **Largest Rectangle in Histogram:**



⇒ We have to return the largest rectangle i.e. the rectangle that have largest area.

⇒ So, we will find area due to every rectangle and store that.

⇒ Area due to every rectangle:

2 X 9 = 18
3 x 2 = 6
4 x 1 = 4
2 x 9 = 18
6 x 1 = 6
5 x 2 = 10
4 x 4 = 16
5 x 1 = 5
3 x 5 = 15

Date _____ Page _____

⇒ So, here, we can observe a pattern that we are going until we don't get smaller rectangle than itself on both sides.

⇒ That means, we have find next smallest on left & right.

⇒ So, if we calculate area for 4 height rectangle then —

⇒ It's next smallest left is on 3rd index & next smallest right is on 8th index.

⇒ So, width = (8 − 3) − 1 = 5 − 1 = 4
& height = 4

⇒ So, Area = 4 × 4 = 16

⇒ If any rectangle don't have NSR & NSL then for NSR we will return ~~+~~ size of array & for NSL, we will return -1.

	0	1	2	3	4	5	6	7	8
arr	2	3	4	2	6	5	4	5	3
NSR	9	2	3	9	5	6	8	8	9
NSL	-1	0	1	-1	2	3	3	6	3
	18	6	4	18	6	10	16	5	15

Date _____ Page _____

Code

```
vector<int> right(n);
vector<int> left(n);
stack<int> st;

for(int i=0; i<n; i++){
    while(!st.empty() && height[st.top()]
                        > height[i]){
        right[st.top()] = i;
        st.pop();
    }
    st.push(i);
}
while(!st.empty()){
    right[st.top()] = n;
    st.pop();
}

for(i=n-1; i>=0; i--){
    while(!st.empty() && height[st.top()] >
                        height[i]){
        left[st.top()] = i;
        st.pop();
    }
    st.push(i);
}
```

Date _____ Page _____

```
while(!st.empty()){
    left[st.top()] = -1;
    st.pop();
}

int ans = 0;
for(i=0; i<n; i++){
  ans=max(ans, height[i] * (right[i] -
                ~~right[~~ left[i] - 1);
}
return ans;
```

⇒ So, we have solve this question in three pass.

⇒ So, how can we solve this problem in one pass?

⇒ So, when we are finding NSR then the values are stored in increasing order in stack.

⇒ So, we can find NSR and for NSL, the element after popping is NSL for that element.

⇒ This method is only apply when we want overall answer, not apply for ~~every~~ finding every area of the rectangle.

Date _____ Page _____

Code

```
stack<int> st; int ans = 0;
for(int i=0; i<n;i++){ :
        while(!st.empty() && height[st.top()]
                                >= height[i]){
            index = st.top();
            st.pop();
            if(!st.empty()){
                ans = max(ans, height[index]
                        *(i-st.top()-1);
            else
                ans = max(ans, height[index]*i);
        }
        st.push(i);
}
while(!st.empty()){
    index = st.top();
    st.pop;
    if(!st.empty()){
        ans = max(ans, height[index] *
                (n - st.top()-1);
    else
        ans = max(ans, height[index] *(n));
}
return ans;
```

Date _____ Page _____

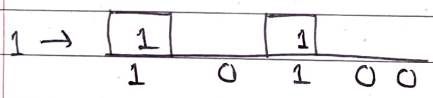
* Maximal Rectangle:

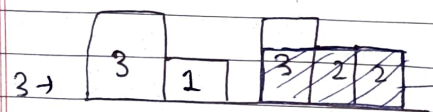
⇒

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

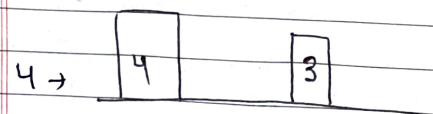
⇒ We have to find the max. area of that rectangle that made up of 1.

⇒ So, we can take every row as base & convert it into as previous problem.





4 ans



⇒ So, as we can see, we have converted this problem into previous problem.

Date Page

⇒ We will create a height array that contains the height of rectangle.

Code

```
int ans = 0;
int row = matrix.size();
int col = matrix[0].size();
vector<int> height(col, 0);
for(i=0; i<row; i++){
    for(j=0; j<=<col; j++){
        if(matrix[i][j] == '0')
            height[j] = 0;
        else
            height[j] += 1;
    }
    ans = max(ans, rectangle(height));
}
return ans;
```