Todays Content

Strength: 22

1. Max histogram area
2. Max rectangular area

Histogram Area:

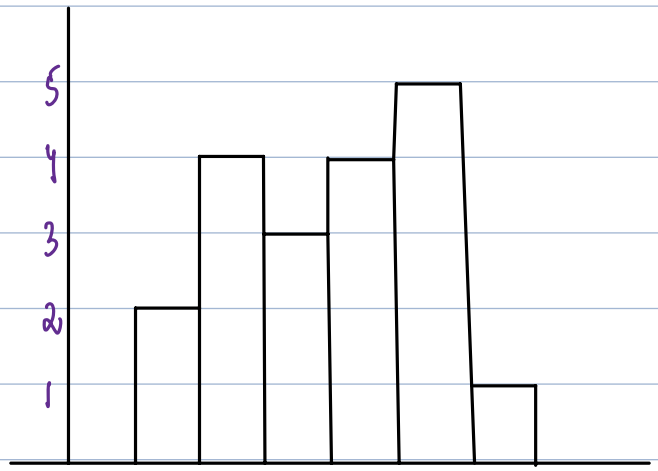
Given continous block of histogram find max Rectangle Area.

Note1: Rectangular area, should be with in histogram

Note2: Width of each histogram is 1.

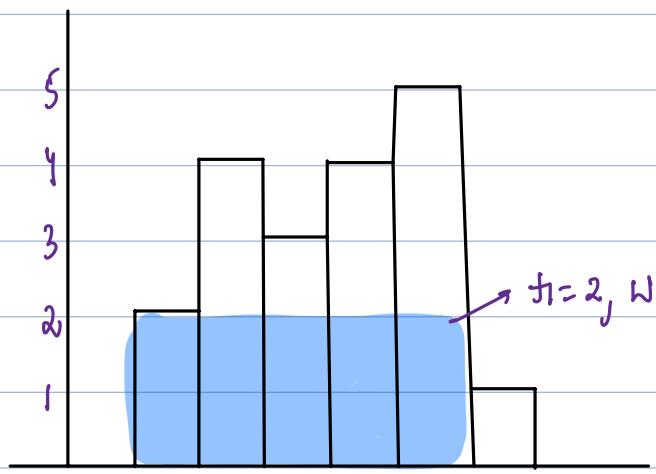
Eg1:

	0	1	2	3	4	5
ar[6] = {	2	4	3	4	5	1 }



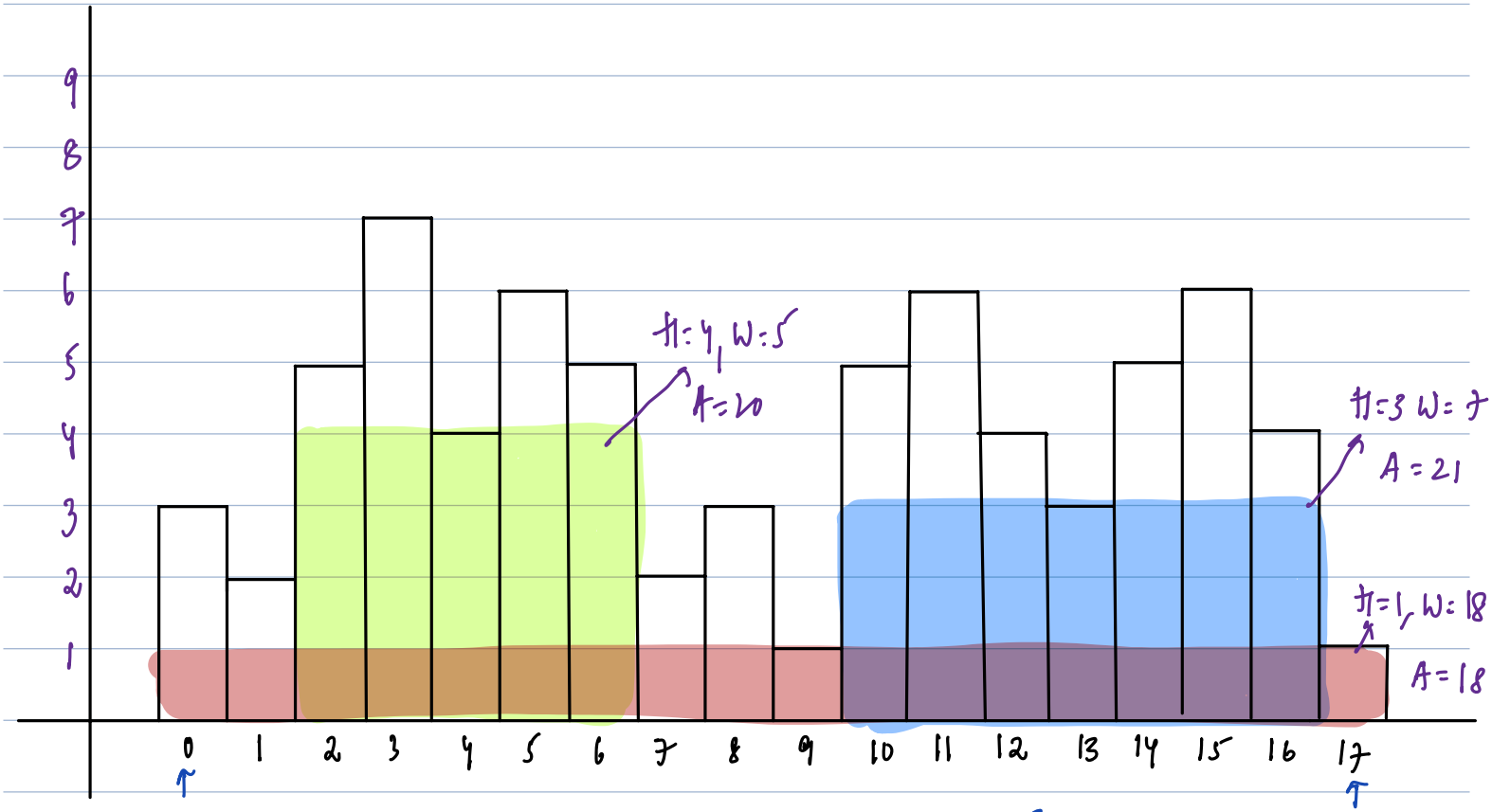
Rect1:

Rect2:



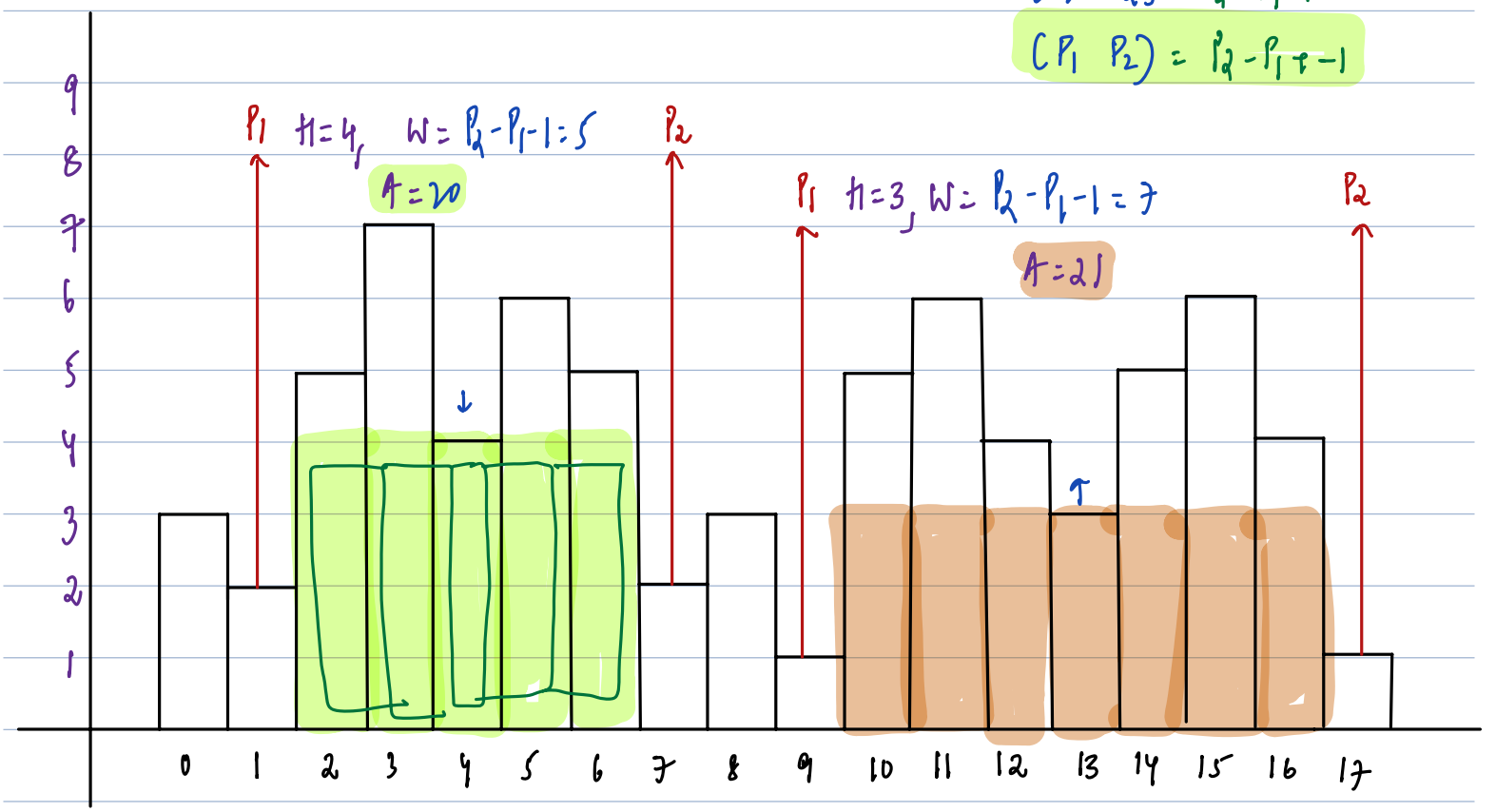
Eg2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
arr[] =	3	2	5	7	4	6	5	2	3	1	5	6	4	3	5	6	4	1



$[P_1\ P_2] = P_2 - P_1 + 1$

$(P_1\ P_2) = P_2 - P_1 + 1$



Rectangle:

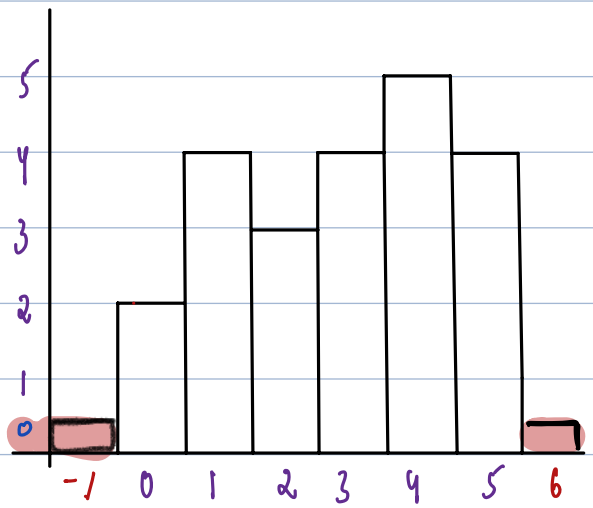
#Hint1: Rectangle height should match histogram height
Which?

#Hint2: Take every histogram height h[i] as rectangle height:
Iterate & calculate P_2 # P_2 is smaller index on right for h[i]
Iterate & calculate P_1 # P_2 is smaller index on left for h[i]
Area = h[i] * ($P_2 - P_1 - 1$)
ans = max(ans, Area);
TC: O(N*N) = O(N^2) SC: O(1)

#Idea2: Optimization
In above for every h[i] we need 1st smaller index on left & right.
Precompute 1st smaller index on left & right for all elements at once

Dry Run:

	0	1	2	3	4	5
ar[] =	2	4	3	4	5	1



sl[] P_1 :	-1	0	0	2	3	2	# Note: Default value for P_1 = -1
sr[] P_2 :	-1	2	6	6	5	6	# Note: Default value for P_2 = N
Width $P_2 - P_1 - 1$:	6	1	5	3	1	3	
Area h*W :	12	4	15	12	5	12	

```
int RectangularArea(vector<int> &ar) { TC: O(N+N+N) = O(N)  SC: O(N+N) = O(N)
    int N = ar.size();                                          10:27 pm
    # 1st smaller index in left
    stack<int> st1;
    vector<int> P1(N, -1);
    for(int i=0; i<N; i++) {
        while(st1.size()>0 && st1.top()>=ar[i]) {
            st1.pop();
        }
        if(st1.size()>0) {
            P1[i] = st1.top();
        }
        st1.push(ar[i])
    }
    # 1st smaller index in right;
    stack<int> st2;
    vector<int> P2(N, N);
    for(int i=N-1; i>=0; i--) {
        while(st2.size()>0 && st2.top()>=ar[i]) {
            st2.pop();
        }
        if(st2.size()>0) {
            P2[i] = st2.top();
        }
        st2.push(ar[i])
    }
    # For each histogram calculate rectangular area
    int ans=0;
    for(int i=0; i<N; i++) {
        ans = max(ans, ar[i] * (P2[i] - P1[i] - 1))
                          Height   Width
    }
    return ans;
}
```

Q

Given a binary matrix [N][M]

Return max rectangular area containing only 1's.

Ex1:

mat[4][5]

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

ans = 6

Ex2:

mat[8][5]

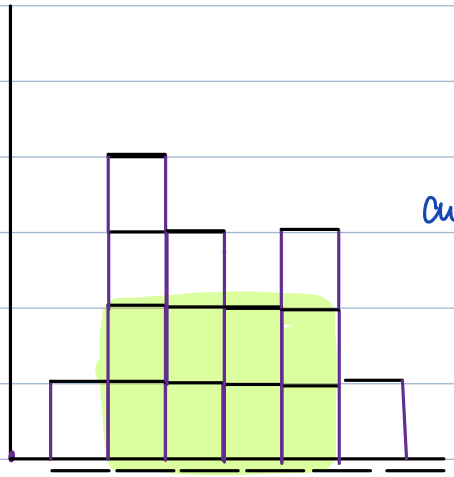
0	1	1	1	1	1	1
1	0	1	1	1	0	1
2	1	1	1	1	1	1
3	1	1	1	1	1	0
4	0	0	1	1	1	0
5	0	1	0	1	1	1
6	1	1	1	1	1	0
7	1	1	1	1	0	1

ans = 12

#Hint: Max rectangular area, whose base at last row.

Ex1: mat[4][6]:

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



#Tracing Ideas:

Take every row base:

For each column in a row, calculate no: of consecutive is along Top.

For each row, apply previous rectangle area logic.

mat[8][5]

	0	1	2	3	4	5
0	1	1	1	1	1	1
1	0	1	1	1	0	1
2	1	1	1	1	1	1
3	1	1	1	1	1	0
4	0	0	1	1	1	0
5	0	1	0	1	1	1
6	1	1	1	1	1	0
7	1	1	1	1	0	1

For every Row: #N

For each col: #M

Iterate along col, to get consective is #N

Apply Above rectangular area #M

Total TC: O(N*M*N + N*M) = O(N²M)

0	1	2	3	4	5	
0	1	0	6	4	1	Ans: 8
1	2	1	7	5	0	Ans: 10
2	3	2	8	0	1	Ans: 8

Optimize Idea:

For each column: Iterate from top to down & calculate consecutive 1s.

mat[8][5]

	0	1	2	3	4	5
0	1↓	1↓	1	1	1	1
1	0↓	1↓	1	1	0	1
2	1↓	1↓	1	1	1	1
3	1↓	1↓	1	1	1	0
4	0↓	0↓	1	1	1	0
5	0↓	1↓	0	1	1	1
6	1↓	1↓	1	1	1	0
7	1↓	1↓	1	1	0	1

mat[8][5]

	0	1	2	3	4	5
0	1	1	1	1	1	1
1	0	2	2	2	0	2
2	1	3	3	3	1	3
3	2	4	4	4	2	0
4	0	0	5	5	3	0
5	0	1	0	6	4	1
6	1	2	1	7	5	0
7	2	3	2	8	0	1

Approach: TC: O(N*M)

For each col:

Iterate & store consecutive 1s information at each cell

For every row:

Apply rectangular area in histogram logic.

```
int maxRectangle(vector<vector<int>>& mat){ TC: O(N*M)  SC: O(M)
    int N = mat.size(), M = mat[0].size();
    for(int j=0; j<M; j++){
        # Iterate on jth col & get consecutive 1s count & store it.
        int c=0;
        for(int i=0; i<N; i++){
            if(mat[i][j]==1){
                c++;
            }else{
                c=0;
            }
            mat[i][j]=c;
        }
    }
    int A=0;
    for(int i=0; i<N; i++){
        A = max(A, RectangleArea(mat[i]));
    }

    return A;
}
```

mat[i] : ith row of ith row