- Given: a matrix size m x n
- You can shift columns only
- Find largest sub-matrix in which all elements are 1

Thoughts:

If all 1's were on bottom or top of columns, would be trivial:

0	0	0	1
0	1	1	1
1	1	1	1
1	1	1	1

Assuming sorted and

top = m[0][-1]

bottom = m[-1][-1]

Continue to move bottom ← and top ↓ checking if maximum area after each iteration.

Problem: NOT SORTED; Have to find sub-matrix that could be ANYWHERE

- Create a container that tracks current height of matrix.
- If next column value is 1 increment height by 1, otherwise assign to 0.
- Create a copy of height container representing sorted heights
- Iterate over each item in height calculate area by

```
height[i] * (height.size() - i)
```

0	0	1
0	1	1
1	1	1

1. height[i] = 1 ; i=0; 1*(3-0) = 3 (answer)
2. height[i] = 2 ; i = 1 2*(3-1) = 4
3. height[i] = 3 ; i = 2 3*(3-2) = 3

Time Complexity: $O(m \cdot n \log n)$

Space Complexity: $O(n)$

```
f(m: [][int]) -> int

    ans: int => 0
    height: [int]  / of size m[0].size
    for row -> m
        for j=0 -> row.size()
            height[j] = 0 if row[j]==0 else height[j]+1
        s_height: [] => height
        sort(s_height)

        for j=0 -> row.size()
            ans = max(ans, s_height[j]*(row.size()-j)
    return ans
```