**Stock span problem:**

**int**[] arr = { 5, 2, 1, 6, 3, 4, 8, 7 };

array is given we have to find out nearest consecutive smaller or equal to element count for each element.

For

5 …….nearest smallest element is 1

2……. Nearest smallest element is 1

1 …… Nearest smallest element is 1

6……. Nearest smallest element is 6, 1, 2, 5 so count will be 4(index of 6 is 3 and index of nearest greatest element is -1 so count of element will 3 – ( -1) = 4

3…….nearest smallest element is 1

4 …… nearest smallest element is 3 so count will be 2(index of 4 is 5 and index of nearest greatest element is 6 and its index is 3 so count of element is 5 – 3 = 2

8 ……nearest smallest element is 8, 4, 3, 6, 3, 1, 2, 5 so count will be 7(same logic like above

7……nearest smallest element is 1

As per the above result we can see that first we need to find out nearest greater element to left. Once we get the nearest greatest element to left we stop there.

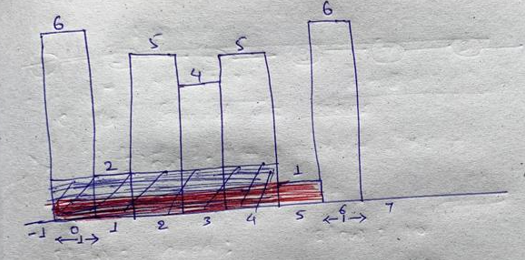
We need to return the count of the element which is consecutive smaller or equal to.

So we need to find out the index of the nearest greatest element to left and take its index. Index of each element – index of ngl will give the count .

**Maximum area in histogram:**

**int**[] arr = { 6, 2, 5, 4, 5, 1, 6 };

Array element is nothing but height of the building. Width will be same 1 unit for all building.



We need to find out maximum area by the rectangle.

Solution:

We can expand the rectangle iff height of the next smaller building height is greater than current element. Once we get the next smaller left and next smaller right we stop expanding the rectangle.

For NSL if we don’t find any smaller element in left then we can think of that at -1 index there is building with height 0. So we will put index of this building that is -1

For NSR if we don’t find any smaller element in right then we can assume at input.length index there is a building with height 0. So we will put index of this building that is input.length;

In the above example 2nd element is 2 so we check left and right of the 2 and find NSL, NSR

NSL = -1

NSR = 1 ( index of 1 is 5)

Width of the rectangle will be ( as we can see in the above image) this 5 will come by doing

width[i] = NSR[i] - NSL[i] - 1;

width of the 2 will 5 – ( -1 ) – 1 = 5

So area of element 2 histogram will be 5 \* 2 = 10

Similarly do the same operation for each element and find the max area.

**Max Rectangle area in binary matrix:**

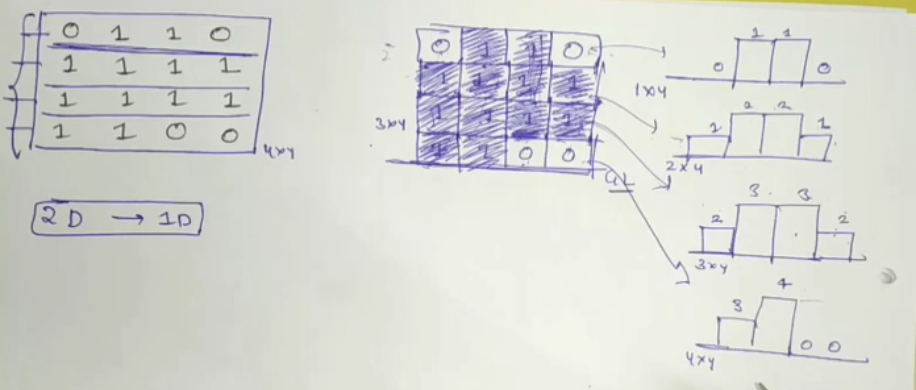
**char**[][] matrix = { { '0', '1', '1', '0' },

{ '1', '1', '1', '1' },

{ '1', '1', '1', '1' },

{ '1', '1', '0', '0' }

};



So now this binary matrix is converted into 4 histograms. Max area of binary matrix will be the max area from all these 4 histograms.

For this first we find out the histogram for the 1st row.

**int** result = maxHistogram(matrix[0]);

Then we iterate from the 2nd row and add the element if element == 1 and after adding the elements we need to find out MAH on this 2nd row elements…and so on

**for** (**int** i = 1; i < row; i++) {

**for** (**int** j = 0; j < column; j++) {

**if** (matrix[i][j] == 1)

matrix[i][j] += matrix[i - 1][j];

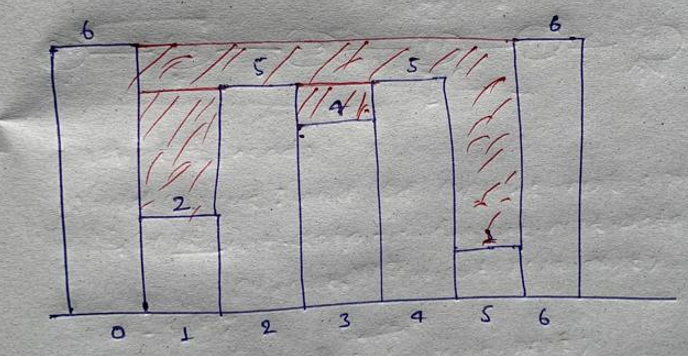
}

result = Math.*max*(result, maxHistogram(matrix[i]));

}

Rain Trapping problem:

**int**[] arr = { 6, 2, 5, 4, 5, 1, 6 };



Here we need to find out the total amount of water stored in between buildings.

Here we need to find out max value in left and right for each building.

Once we find max from left and right we need to find min from this left and right because water amount will depend on min building height only. and amount of water for each building will be total amount of water on that building – building height

Eg:

For building 2, max value in left = 6 and max value in right = 6

So min of(6, 6) is 6

So height of water is 6 – 2(height of building)

And in last sum up all the amount of water and return the result.