



Dash



All



Articles



Videos



Problems



Quiz

<< Prev

Next >>

Largest Rectangle with all 1's

Given a binary matrix, find the maximum size rectangle binary-sub-matrix with all 1's.

Example:

Input :

```
0 1 1 0
1 1 1 1
1 1 1 1
1 1 0 0
```

Output :

8

Explanation :

The largest rectangle with only 1's is from
(1, 0) to (2, 3) which is

```
1 1 1 1
1 1 1 1
```

Input:

```
0 1 1
1 1 1
0 1 1
```

Output:

6

Explanation :





Dash



All



Articles



Videos



Problems



Quiz



The largest rectangle with only 1's is from

(0, 1) to (2, 2) which is

1 1

1 1

1 1

Approach:

In this article, an interesting method is discussed that uses largest rectangle under histogram as a subroutine.

If the height of bars of the histogram is given then the largest area of the histogram can be found. This way in each row, the largest area of bars of the histogram can be found. To get the largest rectangle full of 1's, update the next row with the previous row and find the largest area under the histogram, i.e. consider each 1's as filled squares and 0's with an empty square and consider each row as the base.

Illustration:

Input :

0 1 1 0

1 1 1 1

1 1 1 1

1 1 0 0

Step 1:

0 1 1 0 maximum area = 2

Step 2:

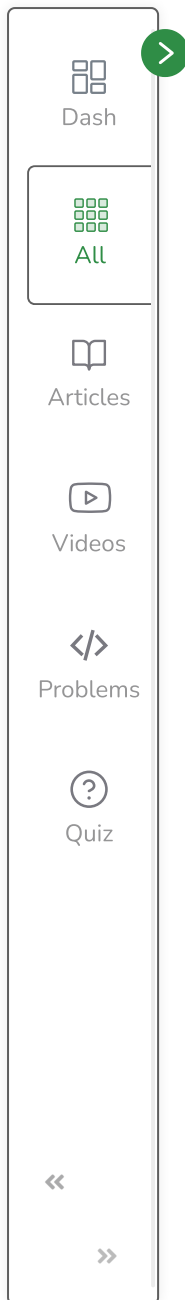
row 1 1 2 2 1 area = 4, maximum area becomes 4

row 2 2 3 3 2 area = 8, maximum area becomes 8

row 3 3 4 0 0 area = 6, maximum area remains 8

Algorithm:





1. Run a loop to traverse through the rows.
2. Now If the current row is not the first row then update the row as follows, if $\text{matrix}[i][j]$ is not zero then $\text{matrix}[i][j] = \text{matrix}[i-1][j] + \text{matrix}[i][j]$.
3. Find the maximum rectangular area under the histogram, consider the i th row as heights of bars of a histogram. This can be calculated as given in this article Largest Rectangular Area in a Histogram
4. Do the previous two steps for all rows and print the maximum area of all the rows.

Note: It is strongly recommended to refer to Largest Area in Histogram post first as most of the code is taken from there.

Implementation

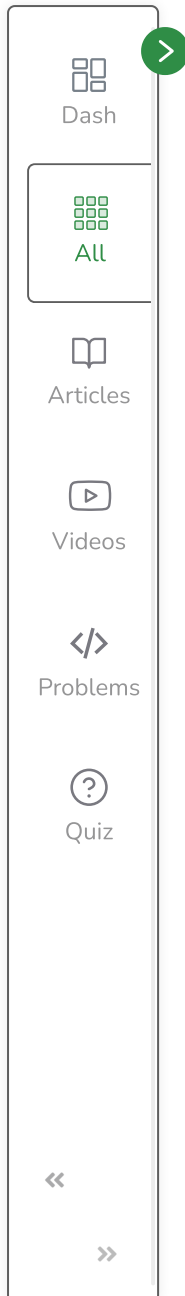
C++**Java**

```
// Java program to find largest rectangle with all 1s
// in a binary matrix
import java.io.*;
import java.util.*;

class GFG {
    // Finds the maximum area under the histogram
    // represented by histogram. See below article for

    static int maxHist(int R, int C, int row[])
    {
        // Create an empty stack. The stack holds indexes of
        // hist[] array/ The bars stored in stack are always
```





```
// in increasing order of their heights.
Stack<Integer> result = new Stack<Integer>();

int top_val; // Top of stack

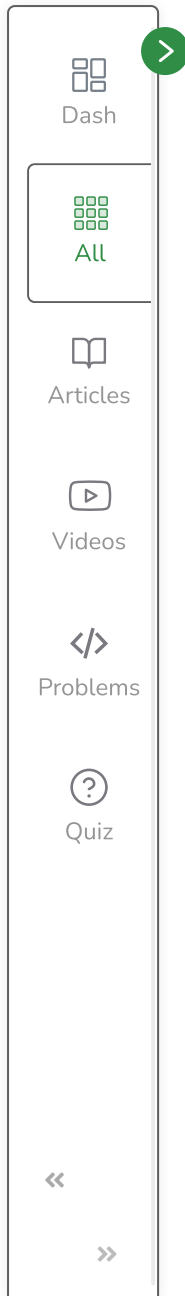
int max_area = 0; // Initialize max area in current
// row (or histogram)

int area = 0; // Initialize area with current top

// Run through all bars of given histogram (or row)
int i = 0;
while (i < C) {
    // If this bar is higher than the bar on top
    // stack, push it to stack
    if (result.empty()
        || row[result.peek()] <= row[i])
        result.push(i++);

    else {
        // If this bar is lower than top of stack,
        // then calculate area of rectangle with
        // stack top as the smallest (or minimum
        // height) bar. 'i' is 'right index' for the
        // top and element before top in stack is
        // 'left index'
        top_val = row[result.peek()];
        result.pop();
        area = top_val * i;
    }
}
```





```

        if (!result.empty())
            area
                = top_val * (i - result.peek() - 1);
            max_area = Math.max(area, max_area);
        }
    }

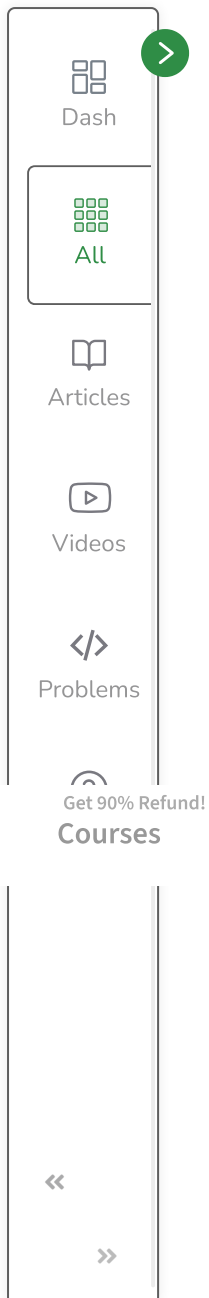
    // Now pop the remaining bars from stack and
    // calculate area with every popped bar as the
    // smallest bar
    while (!result.empty()) {
        top_val = row[result.peek()];
        result.pop();
        area = top_val * i;
        if (!result.empty())
            area = top_val * (i - result.peek() - 1);

        max_area = Math.max(area, max_area);
    }
    return max_area;
}

// Returns area of the largest rectangle with all 1s in
// A[][]
static int maxRectangle(int R, int C, int A[][])
{
    // Calculate area for first row and initialize it as
    // result

```





```
int result = maxHist(R, C, A[0]);

// iterate over row to find maximum rectangular area
// considering each row as histogram
for (int i = 1; i < R; i++) {

    for (int j = 0; j < C; j++)

        // if A[i][j] is 1 then add A[i - 1][j]
        if (A[i][j] == 1)
            A[i][j] += A[i - 1][j];

    // Update result if area with current row (as
    // last row of rectangle) is more
    result = Math.max(result, maxHist(R, C, A[i]));
}

return result;
}
```

}

Tutorials

Jobs

Practice

Contests



```
public static void main(String[] args)
{
    int R = 4;
    int C = 4;

    int A[][] = {
        { 0, 1, 1, 0 },
        { 1, 1, 1, 1 },
    }
```



P

```
        { 1, 1, 1, 1 },
        { 1, 1, 0, 0 },
    };
    System.out.print("Area of maximum rectangle is "
        + maxRectangle(R, C, A));
}
```

Output

Area of maximum rectangle is 8

Complexity Analysis:

- **Time Complexity:** $O(R \times C)$.
Only one traversal of the matrix is required, so the time complexity is $O(R \times C)$
- **Space Complexity:** $O(C)$.
Stack is required to store the columns, so space complexity is $O(C)$

Mark as Read

 Report An Issue

If you are facing any issue on this page. Please let us know.



Dash



All



Articles



Videos



Problems



Quiz

