Find the row with maximum number of 1s

Given a boolean 2D array, where each row is sorted. Find the row with the maximum number of 1s.

```
Example
Input matrix
0 1 1 1
0 0 1 1
1 1 1 1  // this row has maximum 1s
0 0 0 0

Output: 2
```

A simple method is to do a row wise traversal of the matrix, count the number of 1s in each row and compare the count with max. Finally, return the index of row with maximum 1s. The time complexity of this method is O(m*n) where m is number of rows and n is number of columns in matrix.

We can do better. Since each row is sorted, we can **use Binary Search** to count of 1s in each row. We find the index of first instance of 1 in each row. The count of 1s will be equal to total number of columns minus the index of first 1.

See the following code for implementation of the above approach.

```
#include <stdio.h>
#define R 4
#define C 4

/* A function to find the index of first index of 1 in a boolean array
```

Google™ Custom Search

GeeksforGeeks

53,520 people like GeeksforGeeks.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

```
int first(bool arr[], int low, int high)
{
  if(high >= low)
  {
    // get the middle index
    int mid = low + (high - low)/2;

    // check if the element at middle index is first 1
    if ( ( mid == 0 || arr[mid-1] == 0) && arr[mid] == 1)
      return mid;

    // if the element is 0, recur for right side
    else if (arr[mid] == 0)
      return first(arr, (mid + 1), high);

    else // If element is not first 1, recur for left side
      return first(arr, low, (mid -1));
  }
  return -1;
}

// The main function that returns index of row with maximum number of
int rowWithMax1s(bool mat[R][C])
{
    int max_row_index = 0, max = -1; // Initialize max values

    // Traverse for each row and count number of 1s by finding the ind
    // of first 1
    int i, index;
    for (i = 0; i < R; i++)
    {
       index = first (mat[i], 0, C-1);
       if (index != -1 && C-index > max)
       {
           max = C - index;
           max_row_index = i;
       }
    }

    return max_row_index;
}

/* Driver program to test above functions */
int main()
{
    bool mat[R][C] = { {0, 0, 0, 1},
        {0, 1, 1, 1},
```

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
        {1, 1, 1, 1},
        {0, 0, 0, 0}
    };

    printf("Index of row with maximum 1s is %d \n", rowWithMax1s(mat))

    return 0;
}
```

Output:

```
Index of row with maximum 1s is 2
```

Time Complexity: O(mLogn) where m is number of rows and n is number of columns in matrix.

The above solution **can be optimized further**. Instead of doing binary search in every row, we first check whether the row has more 1s than max so far. If the row has more 1s, then only count 1s in the row. Also, to count 1s in a row, we don't do binary search in complete row, we do search in before the index of last max.

Following is an optimized version of the above solution.

```
// The main function that returns index of row with maximum number of
int rowWithMax1s(bool mat[R][C])
{
    int i, index;

    // Initialize max using values from first row.
    int max_row_index = 0;
    int max = C - first(mat[0], 0, C-1);

    // Traverse for each row and count number of 1s by finding the ind
    // of first 1
    for (i = 1; i < R; i++)
    {
        // Count 1s in this row only if this row has more 1s than
        // max so far
        if (mat[i][C-max-1] == 1)
        {
            // Note the optimization here also
            index = first (mat[i], 0, C-max);

            if (index != -1 && C-index > max)
            {
                max = C - index;
                max_row_index = i;
            }
        }
    }
    return max_row_index;
}
```

The worst case time complexity of the above optimized version is also O(mLogn), the will solution work better on average. Thanks to Naveen Kumar Singh for suggesting the above solution.

Sources: this and this

The worst case of the above solution occurs for a matrix like following.
0 0 0 … 0 1
0 0 0 ..0 1 1
0 … 0 1 1 1
….0 1 1 1 1

Following method works in O(m+n) time complexity in worst case.

Step1: Get the index of first (or leftmost) 1 in the first row.

Step2: Do following for every row after the first row
…IF the element on left of previous leftmost 1 is 0, ignore this row.
…ELSE Move left until a 0 is found. Update the leftmost index to this index and max_row_index to be the current row.

The time complexity is O(m+n) because we can possibly go as far left as we came ahead in the first step.

Following is C++ implementation of this method.

```
// The main function that returns index of row with maximum number of
int rowWithMax1s(bool mat[R][C])
{
    // Initialize first row as row with max 1s
    int max_row_index = 0;

    // The function first() returns index of first 1 in row 0.
    // Use this index to initialize the index of leftmost 1 seen so fa
    int j = first(mat[0], 0, C-1) - 1;
    if (j == -1) // if 1 is not present in first row
```

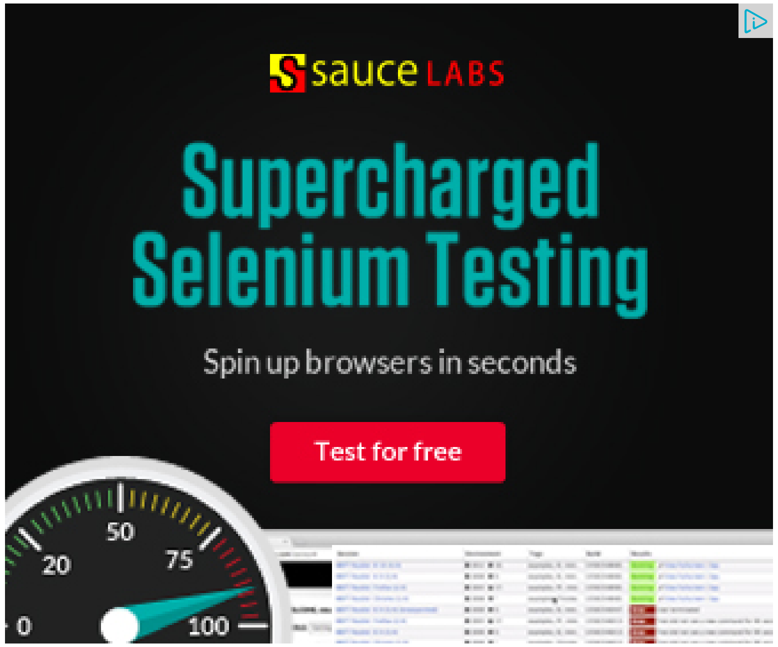
Recent Comments

```
      j = C - 1;

    for (int i = 1; i < R; i++)
    {
        // Move left until a 0 is found
        while (j >= 0 && mat[i][j] == 1)
        {
           j = j-1;  // Update the index of leftmost 1 seen so far
           max_row_index = i;  // Update max_row_index
        }
    }
    return max_row_index;
}
```

Thanks to Tylor, Ankan and Palash for their inputs.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)

- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes
- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen’s Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3

5 Tweet 0 0

Writing code in comment? Please use **ideone.com** and share the link here.

40 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Guest · 6 months ago

Can we directly add the number of 1s in each row and whichever row has gre
number of 1s ?

1 · Reply · Share ›

Ramesh Jothimani · 6 months ago

Can we directly add the numbers in each row and which ever row has greater
of 1s ?

· Reply · Share ›

laxman · 7 months ago

This quesion was asked in my amazon interview.
unfortunately i couldn't make it :(

· Reply · Share ›

trying · 10 months ago

last method won't work for a matrix where there is no 1 at all. The code will ret

4 ^ | ⌄ • Reply • Share ›

anjaneya2 • 10 months ago

what it means row is sorted?

^ | ⌄ • Reply • Share ›

anjaneya2 ➜ anjaneya2 • 10 months ago

got it

^ | ⌄ • Reply • Share ›

raghson • 11 months ago

The third method which has complexity of O(m+n) fails if the very first row has has the flaw. The statement (j==-1) will be true if the first row has all 1s.

The following lines of code should be replaced -

```
 // Old
int j = first(mat[0], 0, C-1) - 1;
    if (j == -1) // if 1 is not present in first row
      j = C - 1;

// Corrected
    int j = first(mat[0], 0, C-1);
    if (j == -1) // if 1 is not present in first row
      j = C - 1;
    else if(j==0) // first row has all 1s and hence we are done
       return max_row_index;
    else
       j--; // go to one position left from leftmost 1
```

1 ^ | ⌄ • Reply • Share ›

shek8034 ➦ raghson • 11 months ago

Nice correction. :)

^ | ˅ • Reply • Share ›

khurshid ➦ shek8034 • 11 months ago

a simple and bugfree approach..
comment if anything is wrong .

```
 int rowReturn(int M[][] , int row, int col )
{
   int j = col - 1 ;
   int ans = j ;
   for ( int i = 0 ; i < row ; i++ )
   {
       while ( j >= 0 && M[i][j] )
       {
          j-- ; ans = i ;
       }
       if ( j == 0) break ;
   }
   if ( j==col-1 ) return -1 ;
   return ans ;
}
```

^ | ˅ • Reply • Share ›

Prem Kamal • 11 months ago

"each row is sorted" according to the question !! your second row isn&#039t.

^ | ˅ • Reply • Share ›

manish • a year ago

```
#include<stdio.h>
int main()
{
        int a[5][5];
        int st=4,j,i,ind=-1;
        for(i=0;i<5;i++)
                for(j=0;j<5;j++)
                        scanf("%d",&a[i][j]);
        for(i=0;i<5;i++)
        {

                if(a[i][st] && st >-1)
                {
                        while(a[i][st] && st+1)
                                st--;
                        ind=i;
                }
        }
        printf("\nThe row containing maximum 1's is at index %d\n",i
}
```

^ | ⌄ • Reply • Share ›

darkpassenger • a year ago

plz explain the complexity of the last method .
how it can be O(m+n) in worst case. consider the case when only last row co

^ | ⌄ • Reply • Share ›

darkpassenger ➦ darkpassenger • a year ago

okk i got it........

^ | ⌄ • Reply • Share ›

Mallikarjun Banda • a year ago

Hey! the method 2 which finds max 1&#039s by first occurrence is failing for {

{1, 1, 1, 0},

{1, 1, 1, 1},

{0, 0, 0, 0}.

};.

Its giving row 1 as max instead of 2 please check.

^ | ⌄ • Reply • Share ›

Marek Černý ↪ Mallikarjun Banda • 13 days ago

Numbers in row must be ordered ;)

{0, 0, 0, 1},

{0, 1, 1, 1},

{1, 1, 1, 1},

{0, 0, 0, 0}.

^ | ⌄ • Reply • Share ›

abhishek08aug • a year ago

Intelligent :D

O(m+n) solution is really cool !

1 ^ | ⌄ • Reply • Share ›

Ganesh • a year ago

[sourcecode language="JAVA"]

/**
* Given a boolean 2D array, where each row is sorted. Find the row with the m

Example
Input matrix
0 1 1 1
0 0 1 1
1 1 1 1 // this row has maximum 1s
0 0 0 0

Output: 2

* @author GAPIITD
*
*/
public class FindTheRowWithMaximumNumberOf1s {

static int mat[][] = new int[][]{ {0, 0, 0, 1},

see more

^ | ˅ • Reply • Share ›

ranga • a year ago

O(m+n) is a very good approach...Thanks alot.

```
/* Paste your code here (You may delete these lines if not writing c
```

^ | ˅ • Reply • Share ›

Piyush • a year ago

```
 #include<iostream>
using namespace std;
#define r 4
```

```
#define c 4

int first(bool arr[r][c])
{
    for(int i=c-1;i>=0;i--)
        if(!arr[0][i])
            return i;
}
int max_1(bool arr[r][c])
{
    int max_row_index,max;
    max_row_index=-1;
    int  i=0;
    int j=first(arr);
    while(i<r && j>=0)
```

see more

Reply • Share ›

hary • 2 years ago

I am writing the code and the strategy is very much what everyone has talked solution to be failing for some cases.

```
 int FindRowWithMaxOnes(const int arr[], unsigned int r, unsigned int
{
    unsigned int row = 0, col = c-1;
    int maxRow = -1;
    while ((row < r) && (col >= 0))
    {
        if ( arr[row][col] == 0 )
            row++;
        else
        {
            maxRow = row;
            col--;
        }
    }
    return maxRow;
}

// maxRow = -1 is an indicative to the fact that you never encountere
```

^ | ⌄ • Reply • Share ›

VtotheIzzay • 2 years ago

Here's a simpler O(n+m) solution that uses the fact that each row is sorted in
Simply traverse in column-major order starting from [0][0] and stop when the f

```
 #include <stdio.h>
#define R 4
#define C 4

// The main function that returns index of row with maximum number of
int rowWithMax1s_new(bool mat[R][C])
{
    //Traverse in column-major-order from [0][0] and stop when the fi

        int i=0, j=0;
        bool found = false;

        for(;!found && j<C; j++)
        {
                for(i=0;i<R; i++)
```

see more

⌃ | ⌄ • Reply • Share ›

JOBBINE JOSEPH ➦ VtotheIzzay • 9 months ago
Even though the method is correct the analysis you have given is incor
i will give you one example ,suppose all the elements are zero except t
program will run for n square times. so the complexity of your algorithn
i am wrong.
⌃ | ⌄ • Reply • Share ›

Aashish ➦ VtotheIzzay • 2 years ago
Processing in column major is usually slower because it doesn't make

Follow this link for further details:
http://stackoverflow.com/quest...

It is always advisable to process a 2D matrix in a row - major fashion.
⌃ | ⌄ • Reply • Share ›

VtotheIzzay ➦ Aashish • 2 years ago
Sure. But row-major traversal wouldn't solve the given problem
⌃ | ⌄ • Reply • Share ›

Anand ➦ VtotheIzzay • 2 years ago
But how is this solution O(n+m)?

What will be the complexity when you have following ma

0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0

Wouldn't your above solution go through each matrix ele

O(m*n)? Plus, as Aashish mentioned above, using colu

Let me know if I am overlooking something obvious.

• Reply • Share ›

Rohit Anand • 10 months ago

@Anand: in your example, j will be -1 and hence j=c-1, a
the rows, but does not enter the while loop as it encoun
sorted), and hence complexity is rather less i.e O(m). H

```
/* Paste your code here (You may delete these li
```

• Reply • Share ›

Shashank • 2 years ago

For each row you are making a linear search for the first '1' by moving left of th
largest exists).But if the number of elements are very large then binary search
skip some number of 1s for counting.

0 0 0 … 0 1
0 0 ..0 1 1
0 … 0 1 1 1
….0 1 1 1 1
when the number of 1s increases linearly by 1 binary search would be more e
after logof(elements in a row) which is more than number of extra 1s found ea
For cases where the matrix is densely populated with 1s binary search would
of ones in each row than linear search

```
/* Paste your code here (You may delete these lines if not writing co
```

• Reply • Share ›

Srini • 2 years ago

The first(...) function return an incorrect index if C = 2 and the first row has or

00
01
10

In this case, first(...) will return 0 as the first index with a '1' which is incorrect.

The condition should be:
if (mid != 0 && a[mid-1] == 0 && a[mid] == 1)
return mid;
// rest of the code is the same

Reply • Share ›

Prashant • 2 years ago

```
public static int getRowForMaxOnes(int [,]array)
{
int i, j,count=0;
int[] result=new int[5];
for (j = 0; j < 5; j++)
{
for (i = 0; i < 5; i++)
{
if (array[i, j] == 1)
result[count] = result[count] + 1;
}
count++;
}

count = result.Max();
for (i = 0; i < 5; i++)
{
```

```
if (result[i] == count)
break;
}
return i;
}
```

1 ^ | ⌄ • Reply • Share ›

Rama • 2 years ago

if the first row doesn't have any 1's then the O(m+n) implementation will returr
To solve this, if j is -1 then move to another row to find a 1. the worst case wou

```
/* Paste your code here (You may delete these lines if not writing co
```

^ | ⌄ • Reply • Share ›

kartik → Rama • 2 years ago

@Rama: Thanks for pointing out this case. Adding a if condition before
Following is the modified code.

```
// The main function that returns index of row with maximum nur
int rowWithMax1s(bool mat[R][C])
{
    // Initialize first row as row with max 1s
    int max_row_index = 0;

    // The function first() returns index of first 1 in row 0.
    // Use this index to initialize the index of leftmost 1 see
    int j = first(mat[0], 0, C-1) - 1;

    if (j == -1)
      j = C-1;
    for (int i = 1; i < R; i++)
```

see more

Reply • Share ›

GeeksforGeeks • 2 years ago

@Tylor, @Ankan and @Palash:

Thanks for your inputs. We have added the O(m+n) solution to this post. Keep

Reply • Share ›

Vineet • 2 years ago

can we do one more method. since all the rows have only ones and zeroes. fi
with the highest sum has the largest no. of 1's. ??

```
/* Paste your code here (You may delete these lines if not writing co
```

Reply • Share ›

Alexey ➦ Vineet • 2 years ago

This method would be o (n * m), because we must sum every elemen

@Tylor & @Ankan 's solution is the first that comes up on my mind an

Reply • Share ›

Kartik • 2 years ago

@Tylor & @Ankan: Thanks for suggesting a new approach.

Please let us know how the time complexity of this approach is O(m+n)

Reply • Share ›

Palash ➦ Kartik • 2 years ago

Palash ↪ Kartik • 2 years ago

It's very straightforward to see that the time complexity of the algorithm
O(m+n).

Step 1 - find 1st 1 in row 1 - O(n)
Step 2 - Go down in each row and move left - Now clearly you can only
ahead in the first step, i.e - O(n).

Reply • Share ›

Ankit ↪ Palash • a year ago

@Palash

Cant step 1 be done in O(logn) time, if we still use binary searc
first row ?

This way the complexity would be O(logn + m)

```
/* Paste your code here (You may delete these lines if r
```

1 • Reply • Share ›

Tylor • 2 years ago

This can be done in O(n+m).

[sourcecode language="C#"]
static int RowWithMax1s(int[,] mat)
{
int result = 0;
int R = mat.GetLength(0);
int C = mat.GetLength(1);

int j = C - 1;
for (int i = 0; i < R; ++i)

```
{
while (j >= 0 && mat[i,j] == 1)
{
j -= 1;
result = i;
}
}
return result;
}
```

• Reply • Share ›

Piyush Kumar • 2 years ago

can v traverse along coloumn n find the first of occurance of 1 in a row

• Reply • Share ›

Ankan • 2 years ago

Approach:

Step1: We get the no of ones in the first row.[We have the index of the first ins
row.

Step2: Now Start moving downwards.

IF the element just below the 1 is 0 ignore it
ELSE IF the element just below 1 is 1 . move left until 0 is found. Update the m

• Reply • Share ›