Search in a row wise and column wise sorted matrix

Given an n x n matrix, where every row and column is sorted in increasing order. Given a number x, how to decide whether this x is in the matrix. The designed algorithm should have linear time complexity.

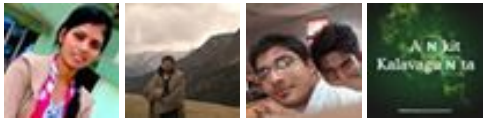
Thanks to devendraiiit for suggesting below approach.

1) Start with top right element
2) Loop: compare this element e with x
….i) if they are equal then return its position
…ii) e < x then move it to down (if out of bound of matrix then break return false)
..iii) e > x then move it to left (if out of bound of matrix then break return false)
3) repeat the i), ii) and iii) till you find element or returned false

Implementation:

```
#include<stdio.h>

/* Searches the element x in mat[][]. If the element is found,
    then prints its position and returns true, otherwise prints
    "not found" and returns false */
int search(int mat[4][4], int n, int x)
{
   int i = 0, j = n-1;  //set indexes for top right element
   while ( i < n && j >= 0 )
   {
      if ( mat[i][j] == x )
      {
         printf("\n Found at %d, %d", i, j);
         return 1;
      }
```

```
        if ( mat[i][j] > x )
          j--;
        else //  if mat[i][j] < x
          i++;
    }

    printf("\n Element not found");
    return 0;  // if ( i==n || j== -1 )
}

// driver program to test above function
int main()
{
  int mat[4][4] = { {10, 20, 30, 40},
                    {15, 25, 35, 45},
                    {27, 29, 37, 48},
                    {32, 33, 39, 50},
                  };
  search(mat, 4, 29);
  getchar();
  return 0;
}
```

Time Complexity: O(n)

The above approach will also work for m x n matrix (not only for n x n). Complexity would be O(m + n).

Please write comments if you find the above codes/algorithms incorrect, or find other ways to solve the same problem.

Geometric Algorithms

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays

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

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

41 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

kaushik Lele • 22 days ago

I have written divide-and conquer method to search key in n*n matrix where ro
ascending.
Algorithm is :-
Go for middle element.
1) If middle element is same as key return.
2) If middle element is lesser than key then

2a) search submatrix on lower side of middle element (half of matrix)
2b) Search submatrix on right hand side.of middle element (up-left quarter)
3) If middle element is greater than key then
3a) search vertical submatrix on left side of middle element (left vertical half)
3b) search submatrix on right hand side. (up-left quarter)

A picture could help it explain better. But code is also easy to understand. Finc
http://ideone.com/zJ29vW

Can anyone comment is this algo is any better than above simple search; esp

kaushik Lele • 23 days ago

What is the correct code for binary search method ?

newCoder • 2 months ago

```
/**
* Given an n x n matrix, where every row and column is sorted in increasing
* order. Given a number x, how to decide whether this x is in the matrix.
* The designed algorithm should have linear time complexity.
*
* @param mat
* @param x
* @return
*/
public static boolean find(int[][] mat, int x) {
```


Recent Comments

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 26 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 30 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 55 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 57 minutes ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it...

Find subarray with given sum · 1 hour ago

```
int j = mat[0].length - 1;

while (i < mat.length && j >= 0) {
if (mat[i][j] == x) {
System.out.println(""+ i + " " + j);
return true;
} else if (mat[i][j] > x) {
j--;
} else {
i++;
}
}
return false;
}
```


numid • 4 months ago

The solution should be updated as it can be done in O(logM +logN) by applyin little modifications:
1. search for the middle element of the matrix
2.if the middle elemnt is less than the missing number,search the lower half else
search the upper half
3. when u get a single row or two rows in which the search is to be performed do 1 comparison with the last arr[i][n] to get the row in which the normal binary the following link have not been commented though,shows the code.
http://ideone.com/s2WvGR

1

Kartik ↗ numid • a month ago

This may not work for many examples. Here the assumption is wrong.

middle element, then we can discard half of the matrix, we can only di

Jiten numid • a month ago

Will this work if the rows are not ordered according to their first elemer

Lokesh numid • 2 months ago

Is there a middle element if this is not a square matrix ?

Varsha Anandani • 6 months ago

can someone tell the space complexity also..???

mahesh Varsha Anandani • 5 months ago

space complexity is O(1). We didn't use any auxiliary space.

Pranav • a year ago

A modified approach, such that moving to down/left occurs in binary fassion.

1) Start with top right element
2) Loop: compare this element e with x
….i) if they are equal then return its position
…ii) e x then move it to left by going to the middle (if out of bound of matrix the
3) repeat the i), ii) and iii) till you find element or returned false

Time Complexity:O(logn + logm)

6

aks Pranav • 4 months ago

element to find is the first col last element)

2

S.m. Amran • a year ago

complete code for 2d array sorting and searching.

```
/*
* File: main.cpp.
* Author: lm.
*
* Created on March 31, 2013, 8:25 PM.
*/
// C library headers.

#include <cassert>
#include <cctype>
#include <cerrno>
#include <cfloat>
#include <ciso646>
#include <climits>
#include <clocale>
#include <cmath>
#include <csetjmp>
```

see more

mrn • a year ago

```
while(l<r)
        {
                m=(l+r)/2;
                if(a[m][0]>x)
                        r=m-1;
```

```
            else
                    if(a[m][col]>x)
                    l=m+1;
                    else
                    break;
        }

        for(int i=0;i<=col;i++)
                if(a[m][i]==x)
                        cout<<m<<" "<<i<<endl;
```

1 ^ | v •

algobard • 2 years ago

Can you please post the recursive solution (O(n^1.53)) to this problem too?

^ | v •

Jay • 2 years ago

Use binary search,

O(log(column))*O(log(row))

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

^ | v •

pavansrinivas ➦ Jay • 6 months ago

actually it will be O(log(row*column)) which is nothing but(O(log(row)+
because the above 2D matrix can be considered as a single sorted arr
complexity of binary search in a sorted array with n elements is O(log(

^ | v •

Ashish Ranjan Singh ➚ Jay • 9 months ago

It should be in order of O(log(row))+O(log(column)) rather O(log(colum
As time for searching for Correct row will take time in O(log(rows)) and
row will take take time O(log(column)).

laddoo • 2 years ago

One Logic can be :

Start searching/comparing with Diagonal Elements of the matrix:
if(arr[i][i] == item)
{ //found the item }
if(arr[i][i] < item && arr[i+1][i+1] > item)
{ //find the element in rest of the ith row and rest of the ith column }
else
{
//increment "i" in loop
}

All above things can be done with binary search as well.
So,Time Complexity can be reduced to : O(log(m+n)).

gaurav1424 • 2 years ago

Is there any way to do this in O(log n) ?

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

rajat rastogi • 2 years ago

Use Improved binary partition method given in leetcode.com

rajat rastogi rajat rastogi • 2 years ago

In this case complexity will be
O(n) constant factor is very less.

Shobhit • 3 years ago

You can check the number with the middle element of matrix. Depending on w
we are searching or greater, we can eliminate 1/4th of the array(either top left
left with 3 smaller matrices of size 1/4th of the original matrix. Continue searcl
the next step you will get 9 matrices of size 1/16 of the original. This approach
as 4/3.

1 

Venki • 3 years ago

Another Approach:

We can cut down the search space by examining the diagonal elements.

Trace the floor (ceil) value of x on the diagonal using binary search. The eleme
floor (ceil) of x.

As an example x = 29 in the above matrix, and x floor value is 25 (ceil value is
then search in the first row and first column (Again we can use binary search

There will be maximum of n elements on diagonal, and less than n elements i
n). The method works only on symmetric matrices.

Can we generalize the method for asymmetric matrices? Yes, I guess.

Venki ➚ Venki • 3 years ago

Sorry, it can only bring down the search space. We can create counte

ddfd • 3 years ago

This question was asked at Microsoft Internship interview 2011.

ddfd • 3 years ago

I think after we find that the last element in the row is greater than the required search on that row . Hence Complexity would be O (m + logn) if matrix of size complexity would be O(n + logn) i.e O(n)

Venki ➚ ddfd • 3 years ago

@ddfd, the method fails as the 2D array is not strictly sorted. For exam 29, the end elements of first two rows are greater than 29, but they dor

If it is like apply binary search on every row, we end up with O(m log n)

shanky • 3 years ago

too gooood n simple solution man.....god knows what what type of algo i was t

shrikant • 3 years ago

can we tackle this problem like a binary search, only that at each point of time mat[(n-1)/2][(n-1)/2], now the number can be in any of the four quadrants.

```
int midi = (n-1)/2, midj=(n-1)/2;
```

```
if(x == mat[midi][midj])
   return true;
else if(x < mat[midi][midj-1] && x < mat[midi-1][midj])
   p = midi-1, q = midj-1;
else if(x > mat[midi+1][midj] && x > mat[midi][midj+1])
   p = midi+1, q = midj+1;
else if(x > mat[midi][midj-1] && x < mat[midi+1][midj])
   p = midi+1, q = midj;
else if(x >mat[midi-1][midj] && x < mat[midi][midj+1])
   p = midi-1, q = midj;
```

please let me know if this should work.

Sandeep shrikant • 3 years ago

@shrikant: The approach should work, but the time complexity of this a
See the below comments from @Kartik and @Vamshi.

shiv • 3 years ago

superb

Vamshi • 3 years ago

One algorithm that i could come up with is as below.

1. Let A be the n*n array.
2. Report the element doesn't exist in the array, if the search element e doesn
A[n,n]
3. If n = 1, return true if the element in the array is same as search element.
3. Consider A[n,n] as a set of 4 subarrays of sizes n/2*n/2 and recursively sea

At any given time, out of the four subarrays the recursive search will exit at the
So, in the worst case, the complexity is like T(n) = 3T(n/4) + constant time wh

Any thoughts?

kartik Vamshi • 3 years ago

@Vamsi: The algo proposed by you is simple and good!

I think the time complexity should be T(n) = 3T(n/**2**) + C. The subproble
O(n^2) though.

Vamshi kartik • 3 years ago

@kartik: I realised my mistake now.

The complexity of this algorithm will be now n^3/2. But the algor
than this one as it is of linear time.

Satyanarayana Batchu Vamshi • a year ago

I think time complexity is T(n) = 3 T(n/4) + C.

Then time complexity should be T(n) = n Pow log 3 bas

This is far less than O(n), so this algorithm is better

Kartik Satyanarayana Batchu • a month ago

No, it is n/2.

Original matrix size was n*n. The reduced matrix size is

reg_frenzy • 3 years ago

Also, another optimization could be applied, extending the idea of girish. We c
searched is lesser or greater than Matrix[n/2][n/2]. Depending on that we coul
the bottom right element as the start element as the start element. This way, v
the matrix.

This could also be extended recursively, searching half the arrays, every time.

ynnus4u • 3 years ago

another approach: given n x m matrix A[][],find x, complexity O(log (min(m,n))

Assume n < m, do binary search on 1st column & find the smallest element th
Say it&#039s gonna be A[i,0]. Now throw all rows after and including i (since X
Repeat for a smaller matrix of size (i-1, m - 1) by proceeding to next column.

Proof of complexity : effectively we are searching along an array of size m, tim

Venki • 3 years ago

Given below are related posts,

http://geeksforgeeks.org/forum...

http://geeksforgeeks.org/forum...

http://geeksforgeeks.org/forum...

@Vick's method seems to be feasible (I haven't tried). On big 2D arrays it may

I closed all of them to consolidate any further comments here.

girish.khadke ➚ Venki • 3 years ago

Two conditions are missing and code can be optimized to handle thos

```
 int search(int mat[4][4], int n, int x)
{
   if(x <a> a[n][n]) return -1;  //Or return some exception coc

   int i = 0, j = n-1;  //set indexes for top right element
   while ( i = 0 )
   {
      if ( mat[i][j] == x )
      {
         printf("\n Found at %d, %d", i, j);
         return 1;
      }
      if ( mat[i][j] > x )
        j--;
      else //  if mat[i][j] < x
        i++;
   }

   printf("\n Element not found");
   return 0;  // if ( i==n || j== -1 )
}
```

girish.khadke ➚ girish.khadke • 3 years ago

Sorry can not get code part properly.

Check if x is less than a[0][0] or greater than a[n][n] and return
have to search such number in array anymore due to sorted or