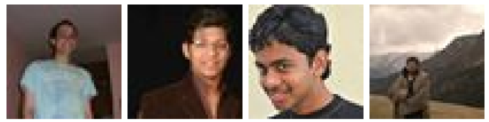
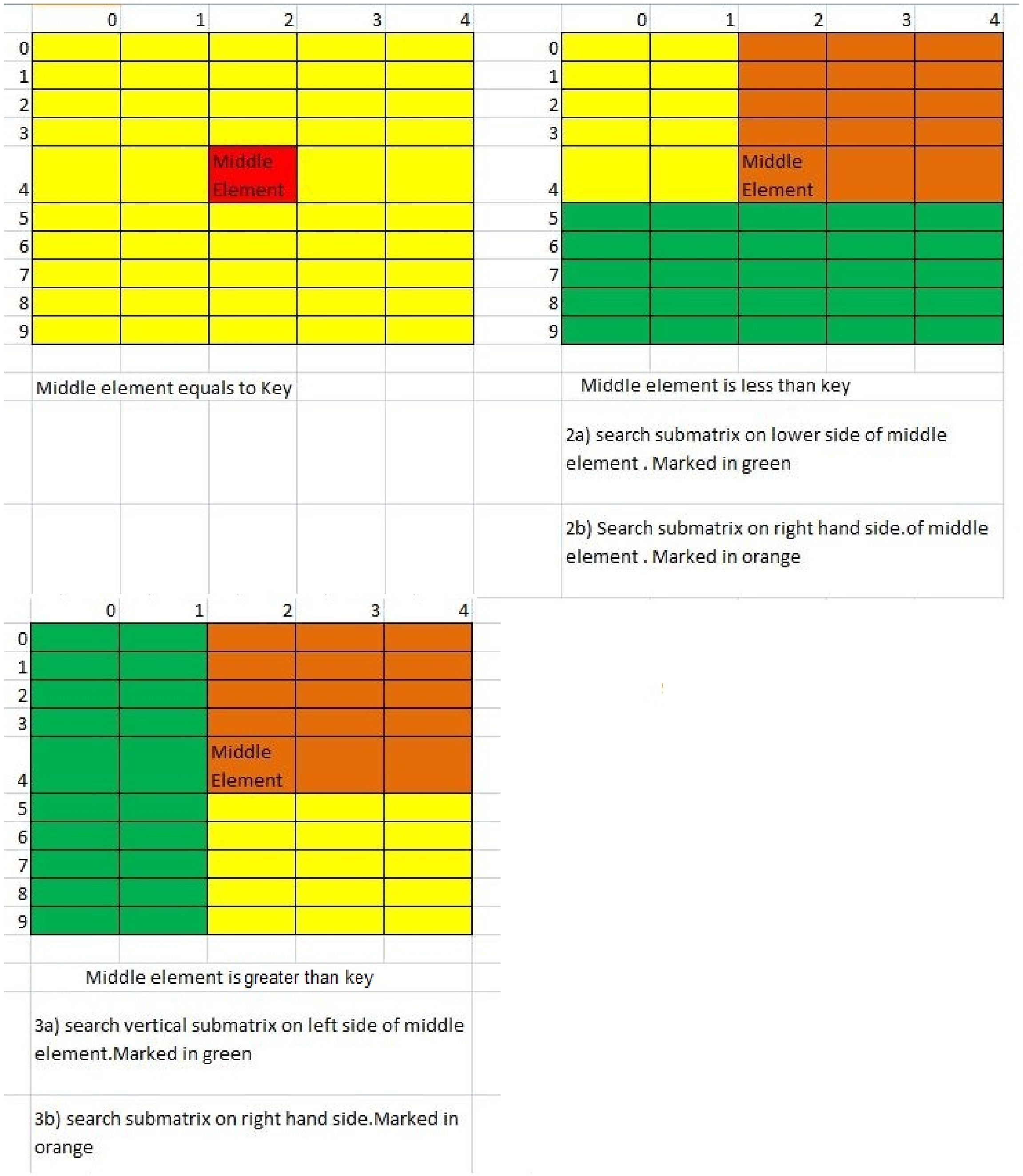
Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)

Given an n x n matrix, where every row and column is sorted in increasing order. Given a key, how to decide whether this key is in the matrix.

A linear time complexity is discussed in the previous post. This problem can also be a very good example for divide and conquer algorithms. Following is divide and conquer algorithm.

1) Find the middle element.
2) If middle element is same as key return.
3) If middle element is lesser than key then
….3a) search submatrix on lower side of middle element
….3b) Search submatrix on right hand side.of middle element
4) If middle element is greater than key then
….4a) search vertical submatrix on left side of middle element
….4b) search submatrix on right hand side.





Following Java implementation of above algorithm.

```
// Java program for implementation of divide and conquer algorithm
// to find a given key in a row-wise and column-wise sorted 2D array
```

Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
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
Tree traversal without recursion and without

```
class SearchInMatrix
{
    public static void main(String[] args)
    {
        int[][] mat = new int[][] { {10, 20, 30, 40},
                                    {15, 25, 35, 45},
                                    {27, 29, 37, 48},
                                    {32, 33, 39, 50}};
        int rowcount = 4,colCount=4,key=50;
        for (int i=0; i<rowcount; i++)
          for (int j=0; j<colCount; j++)
             search(mat, 0, rowcount-1, 0, colCount-1, mat[i][j]);
    }

    // A divide and conquer method to search a given key in mat[]
    // in rows from fromRow to toRow and columns from fromCol to
    // toCol
    public static void search(int[][] mat, int fromRow, int toRow,
                              int fromCol, int toCol, int key)
    {
        // Find middle and compare with middle
        int i = fromRow + (toRow-fromRow )/2;
        int j = fromCol + (toCol-fromCol )/2;
        if (mat[i][j] == key) // If key is present at middle
          System.out.println("Found "+ key + " at "+ i +
                             " " + j);
        else
        {
            // right-up quarter of matrix is searched in all cases.
            // Provided it is different from current call
            if (i!=toRow || j!=fromCol)
             search(mat,fromRow,i,j,toCol,key);

            // Special case for iteration with 1*2 matrix
            // mat[i][j] and mat[i][j+1] are only two elements.
            // So just check second element
            if (fromRow == toRow && fromCol + 1 == toCol)
              if (mat[fromRow][toCol] == key)
                System.out.println("Found "+ key+ " at "+
                                   fromRow + " " + toCol);

            // If middle key is lesser then search lower horizontal
            // matrix and right hand side matrix
            if (mat[i][j] < key)
            {
                // search lower horizontal if such matrix exists
                if (i+1<=toRow)
```

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

```
                search(mat, i+1, toRow, fromCol, toCol, key);
            }

            // If middle key is greater then search left vertical
            // matrix and right hand side matrix
            else
            {
                // search left vertical if such matrix exists
                if (j-1>=fromCol)
                  search(mat, fromRow, toRow, fromCol, j-1, key);
            }
        }
    }
}
```

Time complexity:
We are given a n*n matrix, the algorithm can be seen as recurring for 3 matrices of size n/2 x n/2. Following is recurrence for time complexity

```
T(n) = 3T(n/2) + O(1)
```

The solution of recurrence is O($n^{1.58}$) using Master Method.
But the actual implementation calls for one submatrix of size n x n/2 or n/2 x n, and other submatrix of size n/2 x n/2.

This article is contributed by **Kaushik Lele**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Recent Comments

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

Write a C program to Delete a Tree. · 2 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 5 minutes ago

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

Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- 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
- Sort n numbers in range from 0 to n^2 – 1 in linear time

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

10 Comments GeeksforGeeks

Sort by Newest ▾

Join the discussion

Root to leaf path sum equal to a given number · 30 minutes ago

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

Count trailing zeroes in factorial of a number · 32 minutes ago

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

Find subarray with given sum · 57 minutes ago

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

Find subarray with given sum · 1 hour ago

Join the discussion…

Aman · 13 days ago

In main function why is it there :

for (int i=0; i<rowcount; i++)="" for="" (int="" j="0;" j<colcount;="" j++)="" searc colcount-1,="" mat[i][j]);="">

Reply · Share ›

kaushik Lele → Aman · 11 days ago

It is for testing purpose. Every element is searched one by one.

Reply · Share ›

pd9009 · 14 days ago

One more solution@ http://ideone.com/NR0i8H

Reply · Share ›

kaushik Lele → pd9009 · 13 days ago

Can you explain your strategy using a simple psuedocode

Reply · Share ›

Bhagwat kumar Singh → pd9009 · 13 days ago

int mat[][4] = {{10, 20, 23, 28},

{15, 25, 35, 45},

{27, 29, 37, 48},

{32, 33, 39, 50}};

int flag = search(mat,0,3,0,3,23);

when i do the above changes to your code the segmentation fault com http://www.compileonline.com/c...

and on ideone its run time error.

1 • Reply • Share ›

pd9009 Bhagwat kumar Singh • 11 days ago

Thanks for pointing that out! I made the changes hoping its errc

• Reply • Share ›

omar salem • 15 days ago

third picture should have the caption (middle element is greater than key)

1 • Reply • Share ›

GeeksforGeeks Mod omar salem • 15 days ago

Thanks for pointing this out. We have updated the caption.

1 • Reply • Share ›

ravi • 15 days ago

worst case time complexity of the implementation can be written as

T(m, n) = MAX(T(m/2, n), T(m, n/2)) + T(m/2, n/2)

• Reply • Share ›

ravi ravi • 15 days ago

m is number of rows and n is number of columns in input matrix

• Reply • Share ›

Subscribe Add Disqus to your site