Home Algorithms DS GATE Interview Corner Q&A C C++ Java Books Contribute Ask a Q About
Array Bit Magic C/C++ Articles GFacts Linked List MCQ Misc Output String Tree Graph

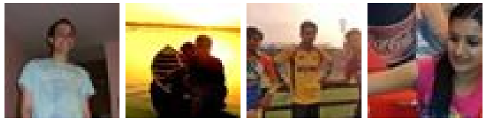
Inplace M x N size matrix transpose | Updated

About four months of gap (missing GFG), a new post. Given an M x N matrix, transpose the matrix without auxiliary memory.It is easy to transpose matrix using an auxiliary array. If the matrix is symmetric in size, we can transpose the matrix inplace by mirroring the 2D array across it's diagonal (try yourself). How to transpose an arbitrary size matrix inplace? See the following matrix,

```
a b c       a d g j
d e f  ==>  b e h k
g h i       c f i l
j k l
```

As per 2D numbering in C/C++, corresponding location mapping looks like,

```
Org element New
 0     a     0
 1     b     4
 2     c     8
 3     d     1
 4     e     5
 5     f     9
 6     g     2
 7     h     6
 8     i    10
 9     j     3
 10    k     7
 11    l    11
```



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

Note that the first and last elements stay in their original location. We can easily see the transformation forms few permutation cycles. 1->4->5->9->3->1 - Total 5 elements form the cycle 2->8->10->7->6->2 – Another 5 elements form the cycle 0 - Self cycle 11 – Self cycle From the above example, we can easily devise an algorithm to move the elements along these cycles. *How can we generate permutation cycles?* Number of elements in both the matrices are constant, given by N = R * C, where R is row count and C is column count. An element at location *ol* (old location in R x C matrix), moved to *nl* (new location in C x R matrix). We need to establish relation between *ol, nl, R* and *C*. Assume *ol = A[or][oc]*. In C/C++ we can calculate the element address as,

```
ol = or x C + oc (ignore base reference for simplicity)
```

It is to be moved to new location *nl* in the transposed matrix, say *nl = A[nr][nc]*, or in C/C++ terms

```
nl = nr x R + nc (R - column count, C is row count as the matrix is transposed)
```

Observe, *nr* = *oc* and *nc* = *or*, so replacing these for *nl*,

```
nl = oc x R + or -----> [eq 1]
```

after solving for relation between *ol* and *nl*, we get

```
ol     = or x C     + oc
ol x R = or x C x R + oc x R
       = or x N     + oc x R    (from the fact R * C = N)
       = or x N     + (nl - or) --- from [eq 1]
       = or x (N-1) + nl
```

OR,

```
nl = ol x R - or x (N-1)
```

Note that the values of *nl* and *ol* never go beyond *N-1*, so considering modulo division on both the sides by (*N-1*), we get the following based on properties of congruence,

```
nl mod (N-1) = (ol x R - or x (N-1)) mod (N-1)
             = (ol x R) mod (N-1) - or x (N-1) mod(N-1)
```

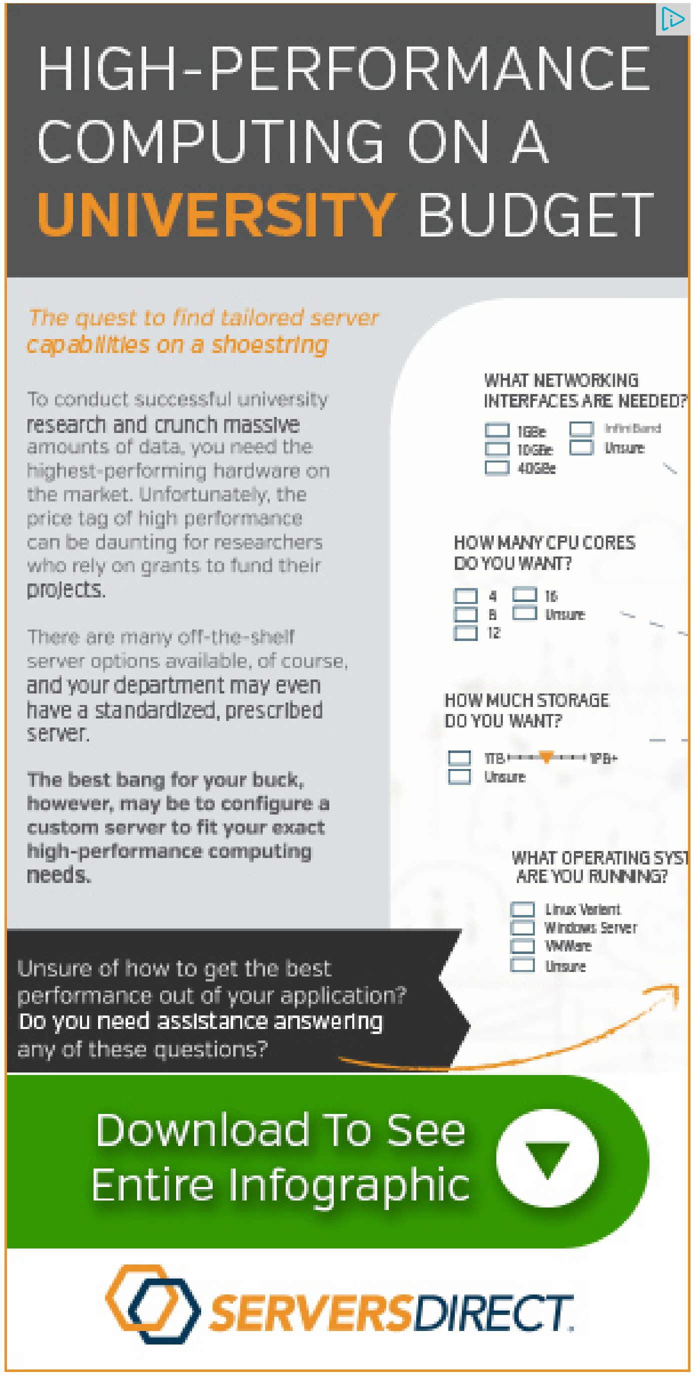
Popular Posts

- 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

```
           = ol x R mod (N-1), since second term evaluates to zero
nl = (ol x R) mod (N-1), since nl is always less than N-1
```

A curious reader might have observed the significance of above relation. Every location is scaled by a factor of R (row size). It is obvious from the matrix that every location is displaced by scaled factor of R. The actual multiplier depends on congruence class of (N-1), i.e. the multiplier can be both -ve and +ve value of the congruent class. Hence every location transformation is simple modulo division. These modulo divisions form cyclic permutations. We need some book keeping information to keep track of already moved elements. Here is code for inplace matrix transformation,

```
// Program for in-place matrix transpose
#include <stdio.h>
#include <iostream>
#include <bitset>
#define HASH_SIZE 128

using namespace std;

// A utility function to print a 2D array of size nr x nc and base add
void Print2DArray(int *A, int nr, int nc)
{
    for(int r = 0; r < nr; r++)
    {
        for(int c = 0; c < nc; c++)
            printf("%4d", *(A + r*nc + c));

        printf("\n");
    }

    printf("\n\n");
}

// Non-square matrix transpose of matrix of size r x c and base addres
void MatrixInplaceTranspose(int *A, int r, int c)
{
    int size = r*c - 1;
    int t; // holds element to be replaced, eventually becomes next el
    int next; // location of 't' to be moved
    int cycleBegin; // holds start of cycle
    int i; // iterator
    bitset<HASH_SIZE> b; // hash to mark moved elements

    b.reset();
```



```
    b[0] = b[size] = 1;
    i = 1; // Note that A[0] and A[size-1] won't move
    while (i < size)
    {
        cycleBegin = i;
        t = A[i];
        do
        {
            // Input matrix [r x c]
            // Output matrix 1
            // i_new = (i*r)%(N-1)
            next = (i*r)%size;
            swap(A[next], t);
            b[i] = 1;
            i = next;
        }
        while (i != cycleBegin);

        // Get Next Move (what about querying random location?)
        for (i = 1; i < size && b[i]; i++)
            ;
        cout << endl;
    }
}

// Driver program to test above function
int main(void)
{
    int r = 5, c = 6;
    int size = r*c;
    int *A = new int[size];

    for(int i = 0; i < size; i++)
        A[i] = i+1;

    Print2DArray(A, r, c);
    MatrixInplaceTranspose(A, r, c);
    Print2DArray(A, c, r);

    delete[] A;

    return 0;
}
```

Output:

Recent Comments

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

Write a C program to Delete a Tree. · 9 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 12 minutes ago

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

Root to leaf path sum equal to a given number · 37 minutes ago

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

Count trailing zeroes in factorial of a number · 39 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

```
  1   2   3   4   5   6
  7   8   9  10  11  12
 13  14  15  16  17  18
 19  20  21  22  23  24
 25  26  27  28  29  30

  1   7  13  19  25
  2   8  14  20  26
  3   9  15  21  27
  4  10  16  22  28
  5  11  17  23  29
  6  12  18  24  30
```

Extension: 17 – March – 2013 Some readers identified similarity between the matrix transpose and string transformation. Without much theory I am presenting the problem and solution. In given array of elements like [a1b2c3d4e5f6g7h8i9j1k2l3m4]. Convert it to [abcdefghijklm1234567891234]. The program should run inplace. What we need is an inplace transpose. Given below is code.

```
#include <stdio.h>
#include <iostream>
#include <bitset>
#define HASH_SIZE 128

using namespace std;

typedef char data_t;

void Print2DArray(char A[], int nr, int nc) {
   int size = nr*nc;
   for(int i = 0; i < size; i++)
      printf("%4c", *(A + i));

   printf("\n");
}

void MatrixTransposeInplaceArrangement(data_t A[], int r, int c) {
   int size = r*c - 1;
   data_t t; // holds element to be replaced, eventually becomes next
   int next; // location of 't' to be moved
   int cycleBegin; // holds start of cycle
   int i; // iterator
   bitset<HASH_SIZE> b; // hash to mark moved elements

   b.reset();
   b[0] = b[size] = 1;
   i = 1; // Note that A[0] and A[size-1] won't move
   while( i < size ) {
      cycleBegin = i;
      t = A[i];
      do {
         // Input matrix [r x c]
         // Output matrix 1
         // i_new = (i*r)%size
         next = (i*r)%size;
         swap(A[next], t);
         b[i] = 1;
         i = next;
      } while( i != cycleBegin );

      // Get Next Move (what about querying random location?)
      for(i = 1; i < size && b[i]; i++)
         ;
      cout << endl;
   }
}

void Fill(data_t buf[], int size) {
   // Fill abcd ...
   for(int i = 0; i < size; i++)
   buf[i] = 'a'+i;

   // Fill 0123 ...
   buf += size;
   for(int i = 0; i < size; i++)
      buf[i] = '0'+i;
}

void TestCase_01(void) {
   int r = 2, c = 10;
   int size = r*c;
   data_t *A = new data_t[size];

   Fill(A, c);

   Print2DArray(A, r, c), cout << endl;
   MatrixTransposeInplaceArrangement(A, r, c);
   Print2DArray(A, c, r), cout << endl;
```

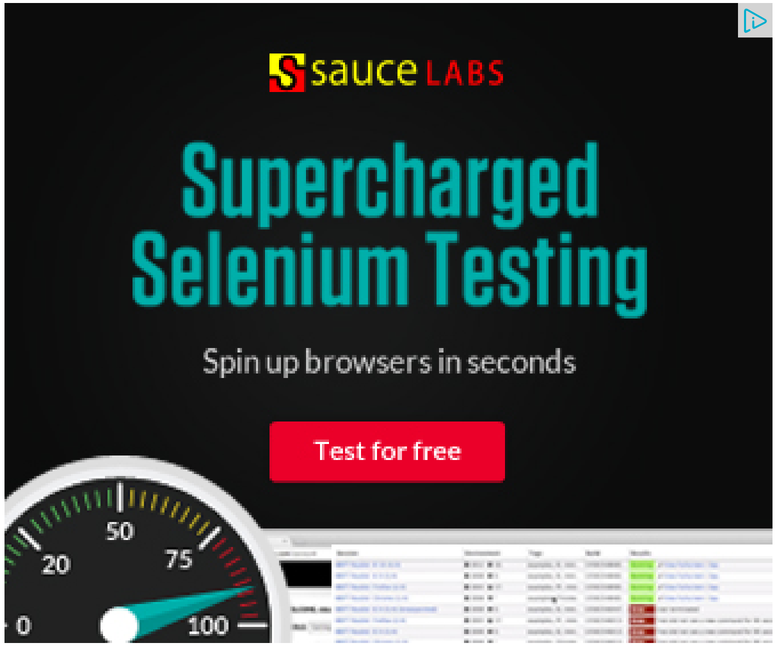
```
    delete[] A;
}

int main() {
    TestCase_01();

    return 0;
}
```

The post is incomplete without mentioning two links.

1. Aashish covered good theory behind cycle leader algorithm. See his post on string transformation.

2. As usual, Sambasiva demonstrated his exceptional skills in recursion to the problem. Ensure to understand his solution.

— Venki. 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

7 Tweet 0 0

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

26 Comments GeeksforGeeks

Sort by Newest ▾

Join the discussion…

Lohith Ravi • 17 days ago

array[i][j] = (array[i][j]%MaxOfArray) + (array[j][i]*MaxOfArray);

This technique can also be used rit

• Reply • Share ›

Venu Gopal • 21 days ago

http://ideone.com/hPygvY

you may see in the output how elements are swapped, and what happens at t

• Reply • Share ›

謝侑驊 • 2 months ago

I was wandering the time complexity of " Matrix Inplace Transpose" this functic especially here,

```
do
{
// Input matrix [r x c]
// Output matrix 1
// i_new = (i*r)%(N-1)
next = (i*r)%size;
swap(A[next], t);
b[i] = 1;
i = next;
}
```

how can we judge the big O of this code.???

and I don't know the

^ | ˅ • Reply • Share ›

Anil • 7 months ago

The mentioned program for string transformation is giving wrong answer. Che

```
void Fill(data_t buf[], int size) {
// buf[]="a0b1c2d3e4f5g6h7i8j9";
for(int i = 0; i < size; i++) {
i%2 == 0 ? buf[i] = 'a'+i/2 : buf[i] = '0'+i/2;
}
}
```

^ | ˅ • Reply • Share ›

venkat • 7 months ago

In the question " Given an M x N matrix, transpose the matrix without auxiliary ı
used O(N) memory for bitset.
Can anyone explain

• Reply • Share ›

Rohit • 10 months ago

@Venki: Code given for string transformation takes input as abcdefghij012345
and produces output as a0b1c2d3e4f5g6h7i8j9 instead of abcdefghij01234567
statement). I have changed only the Fill() function to make the correct input as
code doesn't produce the correct output. Can you please tell me what is the is
something?

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

```
#include
#include
#include
#define HASH_SIZE 128

using namespace std;

typedef char data_t;
```

see more

• Reply • Share ›

Rohit • 10 months ago

@Venki: Code given for string transformation takes input as abcdefghij012345
and produces output as a0b1c2d3e4f5g6h7i8j9 instead of abcdefghij01234567
statement). I have changed only the Fill() function to make the correct input as
code doesn't produce the correct output. Can you please tell me what is the is
something?

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

#include
#include
#include
#define HASH_SIZE 128

using namespace std;

typedef char data_t;
```

see more

⌃ | ⌄ • Reply • Share ›

Rohit • 10 months ago

@Venki: The code provided for string transformation takes the input arrays as a0b1c2d3e4f5g6h7i8j9 and produces output as a0b1c2d3e4f5g6h7i8j9 insteac what mentioned in the problem statement). I have changed only the Fill() funct a0b1c2d3e4f5g6h7i8j9, but then given code does not produce the correct outp is the issue? Is it the code or I am missing something?

```
/* Paste your code here (You may delete these lines if not writing code) */
#include
#include
#include
#define HASH_SIZE 128

using namespace std;
```

typedef char data_t;

see more

Reply • Share ›

srinivas • 10 months ago

Can u explain why bitset b will use only 128 bits? If it is used to mark the move
required for bitset itself. Then it cannot be considered as inplace transpose, si

4 • Reply • Share ›

cooldude • a year ago

We can also do inplace image rotation by 90 degrees by first inplace transpos
and last columns and so on

Reply • Share ›

abhishek08aug • a year ago

Intelligent :D

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

Reply • Share ›

rahul • a year ago

@venki:-Nice post.plz tell me how this statement

next = (i*r)%size;
ensures that in one cycle no two elements map to same location..i mean say
x->y->z->a->b
how this code ensures that in one cycle no elemt is repeated

like we cant have x y z a y x

as it is not happening but i want to know how it is working.
And how swap is implemented..Is it swapping two variables?We are just pass
And y we need to swap?We can simple put value and new index and save old

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

^ | ⌄ • Reply • Share ›

Venu Gopal ➦ rahul • 21 days ago
http://ideone.com/hPygvY

you may see in the output how elements are swapped, and what happ

^ | ⌄ • Reply • Share ›

Aishwarya • a year ago
Now I can answer a question which was asked in an interview. Here is the que

In given array of elements like [a1,a2,a3,..an,b1,b2,b3,..bn,c1,c2,c3,...cn] Write
[a1,b1,c1,a2,b2,c2,...an,bn,cn].

PS: Do it without using extra memory
Sample Testcases:
Input #00:
{1,2,3,4,5,6,7,8,9,10,11,12}
Output #00:
{1,5,9,2,6,10,3,7,11,4,8,12}
Explanation:
Here as you can notice, the array is of the form
{a1,a2,a3,a4,b1,b2,b3,b4,c1,c2,c3,c4}

^ | ⌄ • Reply • Share ›

Venki ➦ Aishwarya • a year ago

@Aishwarya, I have extended the post to cover inplace merging of strii

1 ⌃ | ⌄ • Reply • Share ›

kiran • 2 years ago

@venki:
Great post.
For selecting the next move. starting point in the loop can be cycleBegin

```
// Get Next Move (what about querying random location?)
        for (i = cycleBegin; i < size && b[i]; i++)
```

Please correct me if I am missing something

⌃ | ⌄ • Reply • Share ›

MJ ➦ kiran • 2 years ago

Can you plz put the code for choosing next move?

⌃ | ⌄ • Reply • Share ›

Venki ➦ kiran • 2 years ago

Yes, thanks. This can be one way. But I suspect that there might be sc
say, there are some uncovered elements between two cycleBegins. P
covered as move forward. Overall I see it as an improvement.

A deterministic random number generator can do well. As long as ther
query a random location repeatedly till we find hash location with zero
probability distribution of random number generator.

⌃ | ⌄ • Reply • Share ›

Nicam • 2 years ago

I believe a simple two for loops will do, no need to do such math conversions. sure the source matrix has the size of [max(row, col)][max(row, col)].

```
public void transpose(int[][] matrix) {
                int min = Math.min(matrix.length, matrix[0].length);
                int max = Math.max(matrix.length, matrix[0].length);

                for (int i = 0; i < min; ++i) {
                        for (int j = i+1; j < max; ++j) {
                                int tmp = matrix[i][j];
                                matrix[i][j] = matrix[j][i];
                                matrix[j][i] = tmp;
                        }
                }
        }
```

^ | ˅ • Reply • Share ›

Venki ➦ Nicam • 2 years ago

@Jayaprakash, note that @Nicam's approach is nothing special. I hav an exercise. It is an implementation of square matrix transpose, which here is complex, infact it was a fun to me while deriving these equatior

An example output makes Nicam's method more clear.

Consider 2 x 3 size matrix. It appears as (row major),

1 2 3 | 4 5 6 | 0 0 0

Note that the zeros are excess room for making the matrix a perfect s

1 4 0 | 2 5 0 | 3 6 0

We have to omit last element (in general |M-N| elements) to treat the m
need to move these zero to right end to make the matrix as per C/C++

The whole intention behind the algorithm is transpose of non-square m

^ | ⌄ • Reply • Share ›

Jayaprakash S ➦ Venki • a year ago

@venki agree.. Thank you

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

^ | ⌄ • Reply • Share ›

Jayaprakash Sundararaj ➦ Nicam • 2 years ago

@Venki, @admin : Nicam's solution seems to ok and easy enough. W
method to do the transpose? I dont see any memory improvement -OF
solution.

^ | ⌄ • Reply • Share ›

Aashish ➦ Nicam • 2 years ago

This will consume space. e.g. For transposing a matrix of size 4000 x

^ | ⌄ • Reply • Share ›

a2 • 2 years ago

Can the following code be improved any further ?

```
#include <stdio.h>
#include <stdlib.h>

void printm(int* A,int r,int c)
```

```
{
        for(int i=0;i<r;i++)
        {
                for(int j=0;j<c;j++)
                        printf(" %d ",*(A+(i*c)+j));
                printf("\n");
        }
}

void transpose(int* mat,int row ,int col)
{
```

see more

^ | ⌄ • Reply • Share ›

a2 ➦ a2 • 2 years ago

Obviously I have tried the naive approach !

^ | ⌄ • Reply • Share ›

sreeram • 2 years ago

awesome ...every time i see this question i redirect to wiki for that ccycles and venki ...

^ | ⌄ • Reply • Share ›

✉ Subscribe   Add Disqus to your site