

Count all possible paths from top left to bottom right of a mXn matrix

The problem is to count all the possible paths from top left to bottom right of a mXn matrix with the constraints that **from each cell you can either move only to right or down**

We have discussed a [solution to print all possible paths](#), counting all paths is easier. Let NumberOfPaths(m, n) be the count of paths to reach row number m and column number n in the matrix, NumberOfPaths(m, n) can be recursively written as following.

```
#include <iostream>
using namespace std;

// Returns count of possible paths to reach cell at row number m and c
// number n from the topmost leftmost cell (cell at 1, 1)
int numberOfPaths(int m, int n)
{
    // If either given row number is first or given column number is fi
    if (m == 1 || n == 1)
        return 1;

    // If diagonal movements are allowed then the last addition
    // is required.
    return numberOfPaths(m-1, n) + numberOfPaths(m, n-1);
    // + numberOfPaths(m-1,n-1);
}

int main()
{
    cout << numberOfPaths(3, 3);
    return 0;
}
```

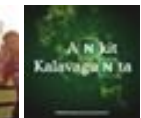
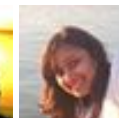
Google™ Custom Search



GeeksforGeeks



53,519 people like [GeeksforGeeks](#).



Interview Experiences

Advanced Data Structures

Dynamic Programming

Greedy Algorithms

Backtracking

Pattern Searching

Divide & Conquer

Mathematical Algorithms

Recursion

Output:

6

The time complexity of above recursive solution is exponential. There are many overlapping subproblems. We can draw a recursion tree for `numberOfPaths(3, 3)` and see many overlapping subproblems. The recursion tree would be similar to [Recursion tree for Longest Common Subsequence problem](#).

So this problem has both properties (see [this](#) and [this](#)) of a dynamic programming problem. Like other typical [Dynamic Programming\(DP\) problems](#), recomputations of same subproblems can be avoided by constructing a temporary array `count[][]` in bottom up manner using the above recursive formula.

```
#include <iostream>
using namespace std;

// Returns count of possible paths to reach cell at row number m and c
// number n from the topmost leftmost cell (cell at 1, 1)
int numberOfPaths(int m, int n)
{
    // Create a 2D table to store results of subproblems
    int count[m][n];

    // Count of paths to reach any cell in first column is 1
    for (int i = 0; i < m; i++)
        count[i][0] = 1;

    // Count of paths to reach any cell in first column is 1
    for (int j = 0; j < n; j++)
        count[0][j] = 1;

    // Calculate count of paths for other cells in bottom-up manner us
    // the recursive solution
    for (int i = 1; i < m; i++)
    {
        for (int j = 1; j < n; j++)

            // By uncommenting the last part the code calculatest he t
            // possible paths if the diagonal Movements are allowed
            count[i][j] = count[i-1][j] + count[i][j-1]; //+ count[i-1

    }
    return count[m-1][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](#)

```
// Driver program to test above functions
int main()
{
    cout << numberOfPaths(3, 3);
    return 0;
}
```

Output:

6

Time complexity of the above dynamic programming solution is $O(mn)$.

Note the count can also be calculated using the formula $(m-1 + n-1)!/(m-1)!(n-1)!$ as mentioned in the comments of [this](#) article.

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

HP Chromebook 11

google.com/chromebook

Everything you need in one laptop. Made with Google. Learn more.



HIGH-PERFORMANCE COMPUTING ON A UNIVERSITY BUDGET

Define your ideal server

Download the infographic

SERVERSDIRECT

Related Topics:

- 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



19



Tweet

3



1

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

18 Comments

GeeksforGeeks

Sort by Newest ▼



Join the discussion...



Vinod Prabhu · a month ago

assuming that I have a 2 rows and 3 columns matrix.

then the number of paths according to this program is 3. but if ai draw paths it

[0,0]->[0,1]->[0,2] ->[1,2]

[0,0]->[0,1]->[1,1] ->[1,2]

[0,0]->[1,0]->[1,1] ->[1,2]

[0,0]->[1,0]->[1,1] ->[0,1]->[0,2]->[1,2]

^ | v · Reply · Share ›



sujeet singh · a month ago

#define ROW 5

705



Subscribe

Recent Comments

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 3 minutes ago

Sanjay Agarwal bool

tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 28

minutes ago

GOPI GOPINATH @admin Highlight this

sentence "We can easily...

Count trailing zeroes in factorial of a number · 30

minutes ago

newCoder3006 If the array contains negative

numbers also. We...

Find subarray with given sum · 54 minutes ago

newCoder3006 Code without using while

loop. We can do it...

Find subarray with given sum · 1 hour ago

Sanjay Agarwal You can also use the this

method:...

Count trailing zeroes in factorial of a number · 1

hour ago

AdChoices ▶

▶ [Matrix in Java](#)

▶ [C++ Vector](#)

▶ [Matrix Code](#)



#define ROW 5

#define COLUMN 5

using namespace std;

int get_count_paths(int* matrix,int m,int n)

```
{
return *(matrix+((m-1)*COLUMN)+n-1);
}
```

void set_count_paths(int* matrix)

```
{

for(int i =0 ;i < ROW;i++)
for(int j=0;j< COLUMN;j++)
{
if(i==0 ||j ==0)
*(matrix+(i*COLUMN)+j)=1;
else
*(matrix+(i*COLUMN)+j)= *(matrix+((i-1)*COLUMN)+j)+ *(matrix+(i*COLUMN
}
}
```

^ | v • Reply • Share ›



proton • a month ago

// Count of paths to reach any cell in first column is 1

for (int i = 0; i < m; i++)

count[i][0] = 1;

We're starting from top-left with one column at a time...How then to reach any

^ | v • Reply • Share ›



Gnanodharan Madhavan • a month ago

Simple recursion to print all the paths.

AdChoices ▶

▶ [Matrix for All](#)

▶ [Can Matrix](#)

▶ [Part Matrix](#)

AdChoices ▶

▶ [Matrix Total](#)

▶ [Matrix Questions](#)

▶ [Time Matrix](#)

```

import java.awt.Point;

import java.util.List;

import java.util.ArrayList;

public class printpathofmxmatrix{

private static int TARGET = 100;

private void printPathABofMatrix(int arr[],int m, int n, List<point> list){

if(m>=arr.length || n>=arr[0].length)

return;

Point point = new Point(m,n);

list.add(point):

```

see more

^ | v • Reply • Share ›



Alok Kumar • a month ago

The time complexity of $O(m*n)$ is OK, but we can improve the space complex

```

#include<stdio.h>
#include<stdlib.h>
int ans(int m,int n)
{
if(m<=0||n<=0) return 0;
if(m>n) return ans(n,m);
int dp1[m];
int loop1,loop2;
for(loop1=0;loop1<m;loop1++) dp1[loop1]="1:" for(loop1="0;loop1<n-1;loop1

```

```

for(loop1=0;loop1<n;loop1++) dp1[loop1]=1; for(loop1=0;loop1<n-1;loop1+
2;loop2">=0;loop2--)
{
dp1[loop2]=dp1[loop2]+dp1[loop2+1];
}
}
return dp1[0];
}
int main()
{
printf("%d\n",ans(5,8));
return 0;
}

```

^ | v • Reply • Share ›



Ritesh Mahato • 3 months ago

@GeeksForGeeks : In second example, the comment should be 'row' and not
// Count of paths to reach any cell in first 'column' is 1

```

for (int j = 0; j < n; j++)
count[0][j] = 1;

```

3 ^ | v • Reply • Share ›



Sekhar • 3 months ago

```

static int printAllPaths(int[][] a, int rowCount, int colCount, int currX, int currY) {

```

```

if (currX == rowCount - 1) {
return 1;
}

```

```

if (currY == colCount - 1) {
return 1;
}

```

```
return printAllPaths(a, rowCount, colCount, currX + 1, currY)
+ printAllPaths(a, rowCount, colCount, currX, currY + 1);
}
```

^ | v • Reply • Share ›



trojansmith1990 • 3 months ago

Hi,

Have written here

<http://ideone.com/qrYpmc>

^ | v • Reply • Share ›



Ram • 4 months ago

There are several variations of this problem which are exhaustively covered at

5 ^ | v • Reply • Share ›



Subrahmanyam Sankar → Ram • 3 months ago

Thank you for sharing this

1 ^ | v • Reply • Share ›



Hari → Ram • 4 months ago

Nice blog thanks for sharing...

^ | v • Reply • Share ›



Rohit Mitra • 4 months ago

It can be written as $(m + n - 2) C (n - 1)$

1 ^ | v • Reply • Share ›



LK → Rohit Mitra • 4 months ago

Could you please explain?

1 ^ | v • Reply • Share ›



gourav pathak → LK • 4 months ago

Are you a developer? Try out the [HTML to PDF API](#)



to reach the final cell you have to take $(m-1)$ steps to the right and $(n-1)$ steps to the down. So total steps are $(m-1)+(n-1)=m+n-2$. Out of these $(m+n-2)$ steps any $(n-1)$ step of ways is $(m+n-2)C(n-1)$ or $(m+n-2)C(m-1)$.

6 ^ | v • Reply • Share ›



h@kumar • 4 months ago

A Short and sweet soln->
 $(2n-2) C (n-1)$

3 ^ | v • Reply • Share ›



sudhakar ➔ h@kumar • 4 months ago

this won't work for large matrix like 1000 x 1000

^ | v • Reply • Share ›



gourav pathak ➔ h@kumar • 4 months ago

Even that would take an $O(mn)$ if m was large

1 ^ | v • Reply • Share ›



Vinod ➔ gourav pathak • a month ago

assuming that I have a 2 rows and 3 columns matrix.

then the number of paths according to this program is 3. but if a

$[0,0] \rightarrow [0,1] \rightarrow [0,2] \rightarrow [1,2]$

$[0,0] \rightarrow [0,1] \rightarrow [1,1] \rightarrow [1,2]$

$[0,0] \rightarrow [1,0] \rightarrow [1,1] \rightarrow [1,2]$

$[0,0] \rightarrow [1,0] \rightarrow [1,1] \rightarrow [0,1] \rightarrow [0,2] \rightarrow [1,2]$

1 ^ | v • Reply • Share ›

 [Subscribe](#)

 [Add Disqus to your site](#)

@geeksforgeeks, **Some rights reserved**

[Contact Us!](#)

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team