Pascal's Triangle

Pascal's triangle is a triangular array of the binomial coefficients. Write a function that takes an integer value n as input and prints first n lines of the Pascal's triangle. Following are the first 6 rows of Pascal's Triangle.

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

Method 1 (O(n^3) time complexity)
Number of entries in every line is equal to line number. For example, the first line has "1", the second line has "1 1", the third line has "1 2 1",.. and so on. Every entry in a line is value of a Binomial Coefficient. The value of *i*th entry in line number *line* is *C(line, i)*. The value can be calculated using following formula.

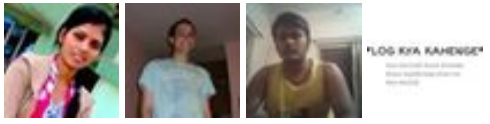
```
C(line, i)   = line! / ( (line-i)! * i! )
```

A simple method is to run two loops and calculate the value of Binomial Coefficient in inner loop.

```
// A simple O(n^3) program for Pascal's Triangle
#include <stdio.h>

// See http://www.geeksforgeeks.org/archives/25621 for details of this
int binomialCoeff(int n, int k);

// Function to print first n lines of Pascal's Triangle
void printPascal(int n)
```

```
{
  // Iterate through every line and print entries in it
  for (int line = 0; line < n; line++)
  {
    // Every line has number of integers equal to line number
    for (int i = 0; i <= line; i++)
      printf("%d ", binomialCoeff(line, i));
    printf("\n");
  }
}
```

```
// See http://www.geeksforgeeks.org/archives/25621 for details of this
int binomialCoeff(int n, int k)
{
    int res = 1;
    if (k > n - k)
       k = n - k;
    for (int i = 0; i < k; ++i)
    {
        res *= (n - i);
        res /= (i + 1);
    }
    return res;
}

// Driver program to test above function
int main()
{
  int n = 7;
  printPascal(n);
  return 0;
}
```

Time complexity of this method is O(n^3). Following are optimized methods.

Method 2(O(n^2) time and O(n^2) extra space)

If we take a closer at the triangle, we observe that every entry is sum of the two values above it. So we can create a 2D array that stores previously generated values. To generate a value in a line, we can use the previously stored values from array.

Popular Posts

All permutations of a given string

Memory Layout of C Programs

Understanding “extern” keyword in C

Median of two sorted arrays

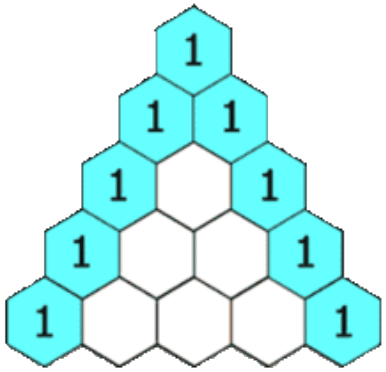
Tree traversal without recursion and without stack!

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




```
// A O(n^2) time and O(n^2) extra space method for Pascal's Triangle
void printPascal(int n)
{
  int arr[n][n]; // An auxiliary array to store generated pscal triang

  // Iterate through every line and print integer(s) in it
  for (int line = 0; line < n; line++)
  {
    // Every line has number of integers equal to line number
    for (int i = 0; i <= line; i++)
    {
      // First and last values in every row are 1
      if (line == i || i == 0)
           arr[line][i] = 1;
      else // Other values are sum of values just above and left of ab
           arr[line][i] = arr[line-1][i-1] + arr[line-1][i];
      printf("%d ", arr[line][i]);
    }
    printf("\n");
  }
}
```

This method can be optimized to use O(n) extra space as we need values only from previous row. So we can create an auxiliary array of size n and overwrite values. Following is another method uses only O(1) extra space.

Method 3 (O(n^2) time and O(1) extra space)

This method is based on method 1. We know that ***i***th entry in a line number *line* is Binomial Coefficient *C(line, i)* and all lines start with value 1. The idea is to calculate *C(line, i)* using *C(line, i-1)*. It can be calculated in O(1) time using the following.

```
C(line, i)   = line! / ( (line-i)! * i! )
C(line, i-1) = line! / ( (line - i + 1)! * (i-1)! )
We can derive following expression from above two expressions.
C(line, i) = C(line, i-1) * (line - i + 1) / i

So C(line, i) can be calculated from C(line, i-1) in O(1) time
```

```
// A O(n^2) time and O(1) extra space function for Pascal's Triangle
void printPascal(int n)
{
  for (int line = 1; line <= n; line++)
  {
    int C = 1;  // used to represent C(line, i)
    for (int i = 1; i <= line; i++)
    {
      printf("%d ", C);  // The first value in a line is always 1
      C = C * (line - i) / i;
    }
    printf("\n");
  }
}
```

So method 3 is the best method among all, but it may cause integer overflow for large values of n as it multiplies two integers to obtain values.

This article is compiled by **Rahul** and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recent Comments

Related Tpoics:

- Backtracking | Set 8 (Solving Cryptarithmetic Puzzles)
- Tail Recursion
- Find if two rectangles overlap
- Analysis of Algorithm | Set 4 (Solving Recurrences)
- Print all possible paths from top left to bottom right of a mXn matrix
- Generate all unique partitions of an integer
- Russian Peasant Multiplication
- Closest Pair of Points | O(nlogn) Implementation

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

25 Comments GeeksforGeeks

Sort by Newest

Count trailing zeroes in factorial of a number · 1 hour ago

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

Find subarray with given sum · 1 hour ago

GREAT ANDY • a month ago

#include<stdio.h>
long factorial(int);

int main()

{

int i,n, c;

printf("ENTER THE NUMBER OF ROWS YOU WISH TO SEE IN PASCAL TF

scanf("%d",&n);

for(i=0;i<n;i++) {="" for(c="0;c<=(n-i-2);c++)" printf("="" ");="" for(c="0;c<=
(factorial(c)*factorial(i-c)));="" printf("\n");="" }="" return="" 0;="" }="" long="" fa
long="" result="1;" for(c="1;c<=n;c++)" result="result*c;" return(result);="" }=

Subrahmanyan Sankar • 3 months ago

```
// PascalsTriangleRecursive.cpp : Defines the entry point for the cons
#include "stdafx.h"
#include <iostream>
using namespace std;
void PascalsTriangle(int a[], int size, int levels)
{
   if (levels > size)
   return;
   if (levels == 0)
    return;
   else
```

```
{
        for (int i = 0; i < size - 1; i++)
        {
          if (a[i] > 0)
          std::cout << "[" << a[i] << "]";
        }
         std::cout << std::endl;
```

see more

preethi • 6 months ago

Please explain why C = C * (line - i) / i is working but not
C = C * (line - i+1) / i.
Thanks.

raghson • 10 months ago

Can anybody explain me that in the third code why we multiply C by (line - i) ar

1

Hailu Kebede • a year ago

it very nice keep it up.

Hailu Kebede • a year ago

it very nice keep it up.

Vikash Verma • a year ago

Try BigInteger class for third approach in java for large values... :D

Vikash Verma • a year ago

Try BigInteger class for third approach in java for large values... :D

Anand Vemprala • a year ago

None of the codes work for very large integers! What should one do in case of

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

Ranveer Singh • a year ago

The implementation with O(n^2) is better and easier too!

Ranveer Singh • a year ago

The implementation with O(n^2) is better and easier too!

pratheba • a year ago

```
 #include<iostream>
#include<queue>
#include<stack>
#include <limits.h>


void pascalTriangle(int n)
{
        if (n == 0)
                return;
```

```
        if(n == 1)
        {
                std::cout<< "1" << std::endl;
                return;
        }

        if(n == 2)
```

see more

bartender • a year ago

```
#include<stdio.h>

int main()
{
   int* a = malloc(100*sizeof(int)); // we can use n instead of
   int* b = malloc(100*sizeof(int));
   int* temp;           //used to swap a and b
   int i=7,k,j;         // i is the number of levels of pascal //tree
   a[1]=1;              //we use arrays starting from 1, ignore //0th
   printf("1\n");        //print 1 statically
   for(k=2;k<=i;k++)     // loop through the levels
   {
     b[1]=1;             //first and last at each level are  //always
     for(j=2;j<=k-1;j++)
      {
         b[j] = a[j-1] + a[j];  // previous level in a
       }
     b[j]=1;                   //ending value at each level is //1
```

see more

Nikhil • 2 years ago

[sourcecode language="C++"]
#include<iostream>
using namespace std;
long fact(int n)
{
if(n==1)
return 1;
else
return (n*fact(n-1));
}
int main()
{
int n;
cout<<"no of lines of code?";
cin>>n;

for(int i=0;i<n;i++)
{

see more

GeeksFollower • 2 years ago

In Method 3:
formula should be:
(line-i+1) // not (line-i)

so it is:
C(line, i) = C(line, i-1) * (line - i+1) / i

Kartik ➚ GeeksFollower • 2 years ago

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

a2 • 2 years ago

Is the Method 1 code correct ?
The algo is correct but I think the code is slightly wrong ..
The correct code is -

```
void printPascal(int n)
{
  // Iterate through every line and print entries in it
  for (int line = 0; line < n; line++)
  {
    // Every line has number of integers equal to line number
    for (int i = line; i >= 0; i--)
      printf("%d ", binomialCoeff(line, i));
    printf("\n");
  }
}
```

Kartik ➚ a2 • 2 years ago

The code given in post and your code, both are correct.

C(n, r) = C(n, n-r)

a2 ➚ Kartik • 2 years ago

But I do not get the correct output from the code given

aishs8 a2 • 2 years ago

You may try to initialize line=0 and i=0 in the above code expect.

```
void printPascal(int n)
{
  // Iterate through every line and print entries
for (int line = 0; line < n; line++)
  {
    // Every line has number of integers equal to

  for (int i = 0; i <= line; i++)
      printf("%d ", binomialCoeff(line, i));
    printf("\n");
  }
}
```

a2 aishs8 • 2 years ago

Thanks !!! :)

siva • 2 years ago

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

```
public void printpascal(int n)
{
int[] a=new int[n];
a[0]=1;
int k=1;
while(k<n)
a[k++]=0;
for(int i=0;i=1;j--)
a[j]+=a[j-1];
k=0;
while(k<n){
if(a[k]!=0)
System.out.print(" "+a[k]);
k++;
}
System.out.println();
}
}
```

1 ^ | v •

sreeram • 2 years ago

code for second method ..

```
/*void printPascal(int n)
{
int arr[n]; // An auxiliary array to store generated pscal triangle values

// Iterate through every line and print integer(s) in it
for (int line = 0; line = 0; i--)
{
// First and last values in every row are 1
```

```
if (line == i || i == 0)
arr[i] = 1;
else // Other values are sum of values just above and left of above
arr[i] = arr[i-1] + arr[i];
printf("%d ", arr[i]);
}
printf("\n");
}
} */
```

rogueh ➦ sreeram • a year ago

This will not work because you are overwriting the previous values. So previous line then arr[i] will have correct value but arr[i-1] will have the v Here is a way to do it :

```
void printPascal(int n)
{
  int arr[n]; // An auxiliary array to store generated pscal tr
  int tmp1,tmp2;
  // Iterate through every line and print integer(s) in it
  for (int line = 0; line < n; line++)
  {
    // Every line has number of integers equal to line number
    for (int i = 0; i <= line; i++)
    {
          tmp2=arr[i];
      // First and last values in every row are 1
      if (line == i || i == 0)
```

1 ^ | v •

suresh • 2 years ago

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

```
#include
long factorial(int k);
long comb(int n,int r)
{
long c;
c=factorial(n)/(factorial(r)*factorial(n-r));
return c;
}
long factorial(int k)
{
long fact = 1;
while(k>0)
{
```

see more

^ | v •

Subscribe Add Disqus to your site