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
```

We strongly recommend that you click here and practice it, before moving on to the solution.

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 https://www.geeksforgeeks.org/archives/25621 for details of this function
```

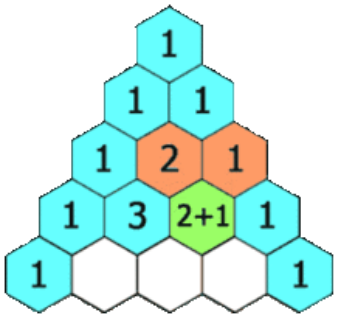
```
int binomialCoeff(int n, int k);

// 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 https://www.geeksforgeeks.org/archives/25621 for details of this function
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;
}
```

Run on IDE

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.




```
// 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 triangle values
```

```
  // 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 above
           arr[line][i] = arr[line-1][i-1] + arr[line-1][i];
      printf("%d ", arr[line][i]);
    }
    printf("\n");
  }
}
```

Run on IDE

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");
  }
}
```

Run on IDE

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.

GATE CS Corner **Company Wise Coding Practice**

Mathematical binomial coefficient pattern-printing

Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

Select a random number from stream, with O(1) space
Program to find amount of water in a given glass
Space and time efficient Binomial Coefficient
Dynamic Programming | Set 9 (Binomial Coefficient)
Find the largest multiple of 2, 3 and 5
Check whether a number is semiprime or not
Sum of series (n/1) + (n/2) + (n/3) + (n/4) +…….+ (n/n)
Sum of all the factors of a number
Sum of the series 2 + (2+4) + (2+4+6) + (2+4+6+8) + …… + (2+4+6+8+….+2n)

Sum of squares of binomial coefficients

Logged in as **himanshudce**(Logout)

3 Average Difficulty : **3/5.0**
Based on **43** vote(s)

Basic Easy Medium Hard Expert

Add to TODO List

Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments Share this post!