

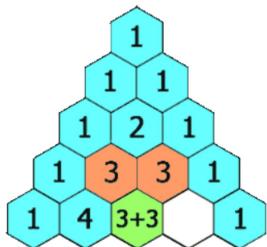
32. Pascal Triangle | Finding nCr in minimal time

118. Pascal's Triangle

Easy 9639 314 Add to List Share

Given an integer `numRows`, return the first `numRows` of **Pascal's triangle**.

In **Pascal's triangle**, each number is the sum of the two numbers directly above it as shown:



Example 1:

```
Input: numRows = 5
Output: [[1],[1,1],[1,2,1],[1,3,3,1],[1,4,6,4,1]]
```

$$\begin{array}{ccccccc} & & & & & & \\ & & & & & & \\ & & 1 & & 1 & & \\ & & & & & & \\ & 1 & & 2 & & 1 & \\ & & & & & & \\ & & 1 & & 3 & & 1 & \\ & & & & & & & \\ & & & 1 & & 4 & & 1 & \\ & & & & & & & & \\ & & & & 1 & & 5 & & 1 & \\ & & & & & & & & & \end{array}$$

3 type of question

1) Given Row and Column, tell the element in that place

$$R = 5, \quad C = 3, \quad \text{Op} \Rightarrow 6$$

2) Print any n^{th} row of pascal triangle

$$N=5 \quad o/p \Rightarrow 1\ 4 \quad 6\ 4\ 1$$

3) Given n , print the entire triangle. $N=6$

Solution for 1st

$$C_{c-1}^{r-1}$$

$${}^n C_r = \frac{n!}{r! \times (n-r)!}$$

$$R=5 \quad C=3$$

$${}^4 C_2 = \frac{4!}{2! (2!)} = \frac{4 \times 3 \times 2 \times 1}{2 \times 1 \times 2 \times 1}$$

$$\boxed{{}^4 C_2 = 6 \text{ o/p}}$$

$${}^7 C_2 = \frac{7!}{2! \times (5!)} = \frac{7 \times 6 \times (5 \times 4 \times 3 \times 2 \times 1)}{2 \times 1 \times 5 \times 4 \times 3 \times 2 \times 1}$$

\searrow 2 place you go before

$$= \frac{7 \times 6}{2 \times 1}$$

$${}^{10} C_3 = \frac{10 \times 9 \times 8}{3 \times 2 \times 1} = \frac{10}{1} \times \frac{9}{2} \times \frac{8}{3}$$

$$C_3 = \frac{3 \times 2 \times 1}{3 \times 2 \times 1} = \frac{1}{1} \times \frac{1}{2} \times \frac{1}{3}$$

3 places you go divide

```
fun NCR (n, r)
{
    long long res = 1
    for (i=0; i<r; i++)

```

```
main()
{
    fun NCR (n-1, r-1)
}
```

$$res = res \times (n-i) \quad // \quad \begin{array}{l} 10-1=10 \\ 10-2=9 \\ 10-3=8 \end{array} \quad \text{numerator}$$

$$res = res / (i+1) \quad (1 \frac{10}{1} \quad \frac{9}{2} \quad \frac{8}{3})$$

}

return res;

}

T.C $\Rightarrow O(r)$ S.C $\Rightarrow O(1)$

Solution for 2nd

1	1	1		
2	1	2	1	
3				1

4	1	3	3	1		
5	1	4	6	4	1	
6	1	5	10	10	5	1

n^{th} row $\rightarrow n^{th}$ element

Brute Force

```

r-1           for (c = 1; c <=n; c++)
c             {
c-1           print (m[n][c] (n-1, c-1))
}

```

T.C $O(n \times r)$ SC $\Rightarrow O(1)$

Optimal Solution

```

r-1
c
c-1

```

$$\begin{array}{cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
 6 & 1 & 5 & 10 & 10 & 5 & 1 \\
 . & 1 & \frac{5}{1} & R-1 C_{c-1} & R-1 C_{c-1} & & \\
 & \frac{5}{1} & S_{C_2} & S_{C_3} & S_{C_4} & & \\
 & \frac{5}{1} & \frac{5 \times 4}{2 \times 1} & \frac{5 \times 4 \times 3}{3 \times 2 \times 1} & \frac{5 \times 4 \times 3 \times 2}{4 \times 3 \times 2 \times 1} & & \\
 & \boxed{\frac{5}{1}} & \boxed{\frac{5}{1} \times \frac{4}{2}} & \boxed{\frac{5}{1} \times \frac{4}{2} \times \frac{3}{3}} & \boxed{\frac{5}{1} \times \frac{4}{2} \times \frac{3}{3} \times \frac{2}{4}} & & \\
 \text{ans} = 1 & C=1 & C=2 & & & &
 \end{array}$$

first & last element = 1 (always)

$\text{ans} = 1$

now if column is zero based indexing,

$$\begin{array}{cccccc}
 6 & 1 & 5 & 10 & 10 & 5 & 1 \\
 0 & 0 & 1 & 2 & 3 & 4 & 5 \\
 -1 & & & & & & \\
 -2 & & & & & & \\
 \text{ans} \times \frac{(\text{row} - \text{col})}{\text{col}}
 \end{array}$$

$\frac{5 \times 4 \times 3 \times 2 \times 1}{(x 2 \times x) \times 4 \times 5}$

whatever is "ans" we are dividing it column

now for multiplying $\Rightarrow (row - col)$

Pseudo Code

ans = 1

print (ans) // 0th col

for (i=1; i < n; i++)

{
 ans = ans * (n - i)
 $\begin{cases} \nearrow \text{row} \\ \searrow \text{col} \end{cases}$

 ans = ans / i

 print (ans);

}

T.C $\Rightarrow O(n)$ S.C $\Rightarrow O(1)$

Solution for 3rd type :

i) Brute Force

$r-1$
 c_{c-1}

ans = []

for (row = 1 \rightarrow n)

{
 tempList = []

```

for (col = 1 → row)
{
    templist.add (nCr (row-1, col-1)),
}

ans.add (templist);

print (ans)

```

$$TC = \approx O(n \times n \times r) \approx O(r^3)$$

Optimize Approach

→ Using 2nd type problem.

CodeStudio

```

1 vector<int> generateRow(int row){
2     long long ans = 1;
3     vector<int> ansRow;
4     ansRow.push_back(1);
5     for(int col = 1; col < row; col++){
6         ans = ans * (row - col); //numerator
7         ans = ans / col; // denominator
8         ansRow.push_back(ans);
9     }
10    return ansRow;
11 }
12 vector<vector<int>> pascalTriangle(int N) {
13     vector<vector<int>> ans;
14     for(int i = 1; i <= N; i++){
15         ans.push_back(generateRow(i));
16     }
17     return ans;
18 }
```

LeetCode

```

1 class Solution {
2 public:
3     vector<int> generateRow(int row){
4         long long ans = 1;
5         vector<int> ansRow;
6         ansRow.push_back(1);
7         for(int col = 1; col < row; col++){
8             ans = ans * (row - col); //numerator
9             ans = ans / col; // denominator
10            ansRow.push_back(ans);
11        }
12        return ansRow;
13    }
14 public:
15     vector<vector<int>> generate(int N) {
16         vector<vector<int>> ans;
17         for(int i = 1; i <= N; i++){
18             ans.push_back(generateRow(i));
19         }
20         return ans;
21     }
22 }
23 }
```