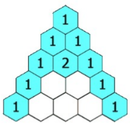
118. Pascal's Triangle

Dec. 13, 2017 | 140.4K views

Average Rating: 4.63 (35 votes)

Given a non-negative integer *numRows*, generate the first *numRows* of Pascal's triangle.



In Pascal's triangle, each number is the sum of the two numbers directly above it.

Example:

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

Approach 1: Dynamic Programming

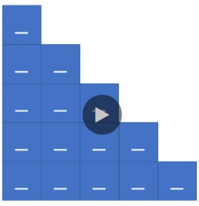
Intuition

If we have the a row of Pascal's triangle, we can easily compute the next row by each pair of adjacent values.

Algorithm

Although the algorithm is very simple, the iterative approach to constructing Pascal's triangle can be classified as dynamic programming because we construct each row based on the previous row.

First, we generate the overall `triangle` list, which will store each row as a sublist. Then, we check for the special case of 0, as we would otherwise return $[1]$. If $numRows > 0$, then we initialize `triangle` with $[1]$ as its first row, and proceed to fill the rows as follows:



1 / 12

Java Python3 Copy

```
class Solution:
    def generate(self, num_rows):
        triangle = []

        for row_num in range(num_rows):
            # The first and last row elements are always 1.
            row = [None for _ in range(row_num+1)]
            row[0], row[-1] = 1, 1

            # Each triangle element is equal to the sum of the elements
            # above-and-to-the-left and above-and-to-the-right.
            for j in range(1, len(row)-1):
                row[j] = triangle[row_num-1][j-1] + triangle[row_num-1][j]

            triangle.append(row)

        return triangle
```

Complexity Analysis

- Time complexity : $O(numRows^2)$

  Although updating each value of `triangle` happens in constant time, it is performed $O(numRows^2)$ times. To see why, consider how many overall loop iterations there are. The outer loop obviously runs $numRows$ times, but for each iteration of the outer loop, the inner loop runs $rowNum$ times. Therefore, the overall number of `triangle` updates that occur is $1 + 2 + 3 + \ldots + numRows$, which, according to Gauss' formula, is

$$\begin{aligned}\frac{numRows(numRows+1)}{2} &= \frac{numRows^2 + numRows}{2} \\ &= \frac{numRows^2}{2} + \frac{numRows}{2} \\ &= O(numRows^2)\end{aligned}$$

- Space complexity : $O(numRows^2)$

  Because we need to store each number that we update in `triangle`, the space requirement is the same as the time complexity.

Rate this article:

Previous Next

Comments: 48

Sort By

Type comment here... (Markdown is supported)

Preview Post

leetcodefan ★ 1909 January 3, 2019 10:23 AM — Report

Man, you make simple stuff complicated, in a horrible way.

And what is this?

```
!?!../Documents/118_Pascals_Triangle.json:1280,720!?!
```

47 Share Reply

SHOW 4 REPLIES

Dr_Sean ★ 535 November 29, 2019 4:02 AM

Easy to understand Python3 code using append:

```
    def generate(self, numRows: int) -> List[List[int]]:
        if numRows  == 0: return []
        elif numRows == 1: return [[1]]
```

Read More

21 Share Reply

SHOW 1 REPLY

terrible_whiteboard ★ 627 May 19, 2020 6:12 PM

I made a video if anyone is having trouble understanding the solution (clickable link)
https://youtu.be/7pOzP9m_bX8

Read More

20 Share Reply

willye ★ 861 May 25, 2018 10:56 PM — Report

This question is so funny, made me realize Pascal's Triangle is the embodiment of DP! Will use this as an example when explaining DP next time :^)

25 Share Reply

SHOW 3 REPLIES

sofs1 ★ 723 February 21, 2019 5:11 AM

This solution was written by https://leetcode.com/problems/pascals-triangle/discuss/38141/My-concise-solution-in-Java :

```
public List<List<Integer>> generate(int numRows)
```

Read More

12 Share Reply

SHOW 2 REPLIES

fuckyoudeletemyaccount ★ 36 September 1, 2018 11:17 AM

Two-liner Python solution (well, three, if you must include the import line):

```
from math import factorial as f

class Solution:
```

Read More

8 Share Reply

SHOW 3 REPLIES

powergascan ★ 2 February 28, 2020 9:23 AM — Report

Python 3: Recursion

```
def gen(numRows):
    if numRows==0:
        return []
```

Read More

2 Share Reply

muhammad2 ★ 3 May 23, 2018 12:02 PM

C# Code that beats more than 96% submissions of C#.

```
public class Solution {
    public IList<IList<int>> Generate(int numRows) {
        IList<IList<int>> pascal = new List<IList<int>>();
```

Read More

3 Share Reply

sacerdoti ★ 16 March 14, 2019 3:56 AM — Report

C++ combinatorics answer

```
class Solution {
public:
    vector<vector<int>> generate(int numRows) {
```

Read More

3 Share Reply

evseenko ★ 2 May 29, 2018 4:06 PM

Recursive Scala solution (beats 85% of other submissions in Scala):

```
object Solution {
    def generate(numRows: Int): List[List[Int]] = {
        val result = new collection.mutable.ListBuffer[List[Int]]()
```

Read More

2 Share Reply