

```

1  /*
2  给定一个非负索引 k，其中 k ≤ 33，返回杨辉三角的第 k 行。
3  在杨辉三角中，每个数是它左上方和右上方的数的和。
4
5  示例：
6
7  输入：3
8  输出：[1,3,3,1]
9
10 进阶：
11
12 你可以优化你的算法到 O(k) 空间复杂度吗？
13
14 来源：力扣（LeetCode）
15 链接：https://leetcode-cn.com/problems/pascals-triangle-ii
16 著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
17 */

```

分析:

- 根据二项式展开定理,第 $n, n \geq 0$ 层杨辉三角系数有 $n + 1$ 个,且第 i 个系数的通用表达式为

$$C_i = C(n, i) = \frac{n!}{i!(n-i)!} \quad (1)$$

- 由于等式中的阶乘操作的复杂度为 $O(n)$,且容易溢出,所以不能直接根据二项式定理直接展开做,否则复杂度为 $O(n^2)$,且会有溢出错误.
- 找迭代关系

$$C_0 = \frac{n!}{0!(n-0)!} \quad (2)$$

$$C_1 = \frac{n!}{1!(n-1)!} = \frac{C_0 * n}{1} \quad (3)$$

$$C_2 = \frac{n!}{2!(n-2)!} = \frac{C_1 * (n-1)}{2} \quad (4)$$

$$\dots \quad (5)$$

$$C_i = \frac{n!}{i!(n-i)!} = \frac{C_{i-1} * (n-i+1)}{i} \quad (6)$$

- 我们只要知道 C_0 的值,即可根据迭代公式进行逐一计算.

方法一:C++_迭代法

```

1  class Solution
2  {
3
4      public:
5          vector<int> getRow(int rowIndex)
6          {
7              int i = 0 ;
8              vector<int> ret_val ;
9              unsigned long temp = 1 ;
10

```

```

11         ret_val.push_back(1);
12         for(i = 1; i <= rowIndex ; i++)
13         {
14             temp = temp * (rowIndex-i+1);
15             temp = temp / i ;
16             ret_val.push_back(temp);
17         }
18         return ret_val;
19     }
20 };
21
22 /*
23 执行结果:
24 通过
25 显示详情
26 执行用时 :0 ms, 在所有 C++ 提交中击败了100.00% 的用户
27 内存消耗 :8 MB, 在所有 C++ 提交中击败了99.84%的用户
28 */

```

方法二:C_迭代法

```

1  /**
2   * Note: The returned array must be malloced, assume caller calls free().
3   */
4  int* getRow(    int    rowIndex    ,
5               int*    returnSize
6               )
7  {
8      int*    ret_val = NULL ;
9      unsigned long    temp    = 1    ;
10     int        i        = 0    ;
11
12     *returnSize = rowIndex+1;
13     ret_val     = (int*)malloc(sizeof(int)*(rowIndex+1));
14     ret_val[0]  = 1 ;
15     for(i=1;i<=rowIndex;i++)
16     {
17         temp = temp *(rowIndex-i+1);
18         temp = temp / i ;
19         ret_val[i] = temp;
20     }
21     return ret_val;
22 }
23
24 /**
25 执行结果:
26 通过
27 显示详情
28 执行用时 :8 ms, 在所有 C 提交中击败了25.00% 的用户
29 内存消耗 :7.1 MB, 在所有 C 提交中击败了6.10%的用户
30 */
31
32 /**
33 * Note: The returned array must be malloced, assume caller calls free().
34 */
35 int* getRow(    int    rowIndex    ,

```

```

36         int*    returnSize
37     )
38 {
39     int*    ret_val = NULL ;
40     int     i      = 0     ;
41     unsigned long    temp    = 1     ;
42
43     *returnSize = ++rowIndex;
44     ret_val     = (int*)malloc(sizeof(int)*(rowIndex));
45     ret_val[0]  = temp ;
46     for(i=1;i<rowIndex;i++)
47     {
48         temp *= (rowIndex-i);
49         temp /= i;
50         ret_val[i] = temp;
51     }
52     return ret_val;
53 }
54
55 /*
56 执行结果:
57 通过
58 显示详情
59 执行用时 :0 ms, 在所有 C 提交中击败了100.00% 的用户
60 内存消耗 :7 MB, 在所有 C 提交中击败了7.32%的用户
61 */

```

AlimyBreak
2019.09.12