# 题目

给定一个非负索引 k，其中k ≤ 33，返回杨辉三角的第k行。

在杨辉三角中，每个数是它左上方和右上方的数的和。

**示例:**

输入: 3

输出: [1,3,3,1]

**进阶：**

你可以优化你的算法到 O(k) 空间复杂度吗？

# 分析

## 方法一：动态规划

思路：

这里采用统一的方法去做：

两步：

1、先一个循环构造杨辉三角的边角值

2、再利用动态方程dp[i][j] = dp[i-1][j-1]+dp[i-1][j];

只是在返回值上增加一下对应的行号索引即可。

代码：

class Solution {

public:

vector<int> getRow(int rowIndex) {

vector<vector<int>> dp(rowIndex + 1, {{0}});

if (rowIndex < 0) return dp[0];

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

dp[i].resize(i + 1);

dp[i][0] = dp[i][i] = 1;

for (int j = 1; j < i; ++j) {

dp[i][j] = dp[i - 1][j - 1] + dp[i - 1][j];

}

}

return dp[rowIndex];

}

};

或：

class Solution {

public:

vector<int> getRow(int rowIndex) {

vector<int> ret(rowIndex+1);

for(int i=0;i<=rowIndex;i++)

{

ret[i] = 1; //初始值

for(int j=i;j>1;j--)

{

ret[j-1] = ret[j-2] + ret[j-1]; //状态转移方程

}

}

return ret;

}

};

或：

#include <vector>

class Solution {

public:

vector<int> getRow(int rowIndex) {

vector<int> row(rowIndex + 1, 0); // 初始化一个大小为 rowIndex + 1 的向量，所有元素为 0

row[0] = 1; // 杨辉三角每一行的第一个元素总是 1

for (int i = 1; i <= rowIndex; ++i) {

for (int j = i; j > 0; --j) {

row[j] += row[j - 1]; // 当前元素等于它左边的元素加上上一行对应位置的元素

}

}

return row;

}

};