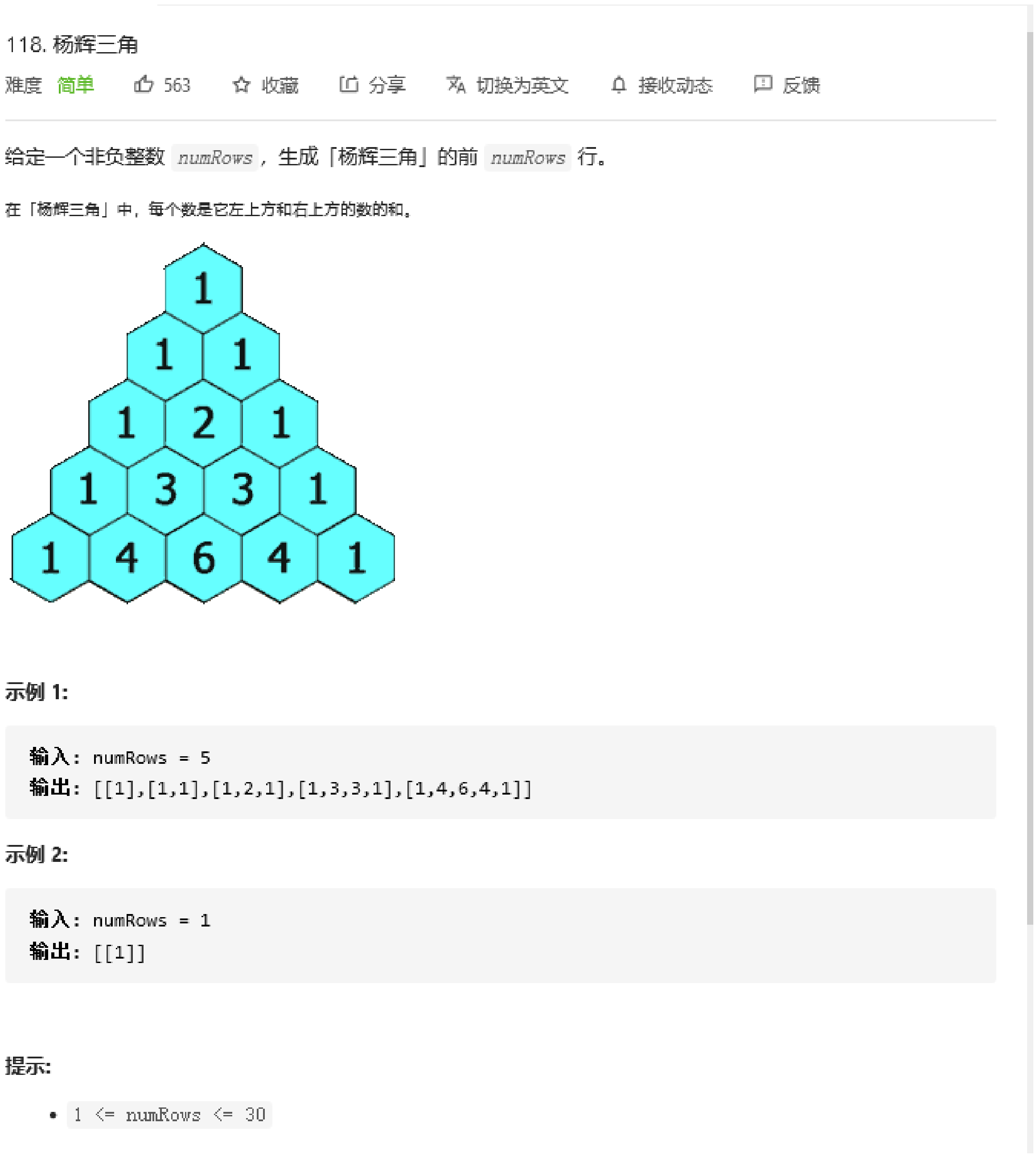
118-杨辉三角

题述

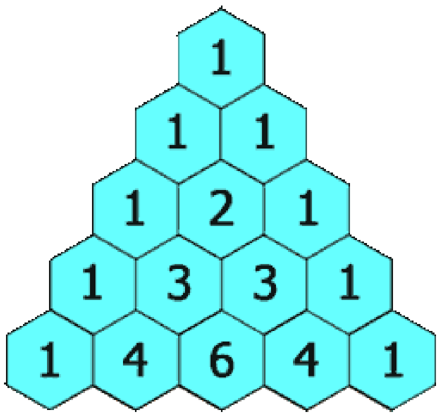


118. 杨辉三角

难度 简单 563 收藏 分享 切换为英文 接收动态 反馈

给定一个非负整数 `numRows`，生成「杨辉三角」的前 `numRows` 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 1:

```
输入: numRows = 5
输出: [[1],[1,1],[1,2,1],[1,3,3,1],[1,4,6,4,1]]
```

示例 2:

```
输入: numRows = 1
输出: [[1]]
```

提示:

- `1 <= numRows <= 30`

浅析

把握理解杨辉三角的特性即可迅速求解得出答案

在杨辉三角中，每个数是它左上方和右上方的数的和。

索引在编码时可能会比较绕，拿草稿纸写几行就会了。

在翻评论区的时候，看到了一种"暴力"解法，有点意思哈哈哈哈。

代码

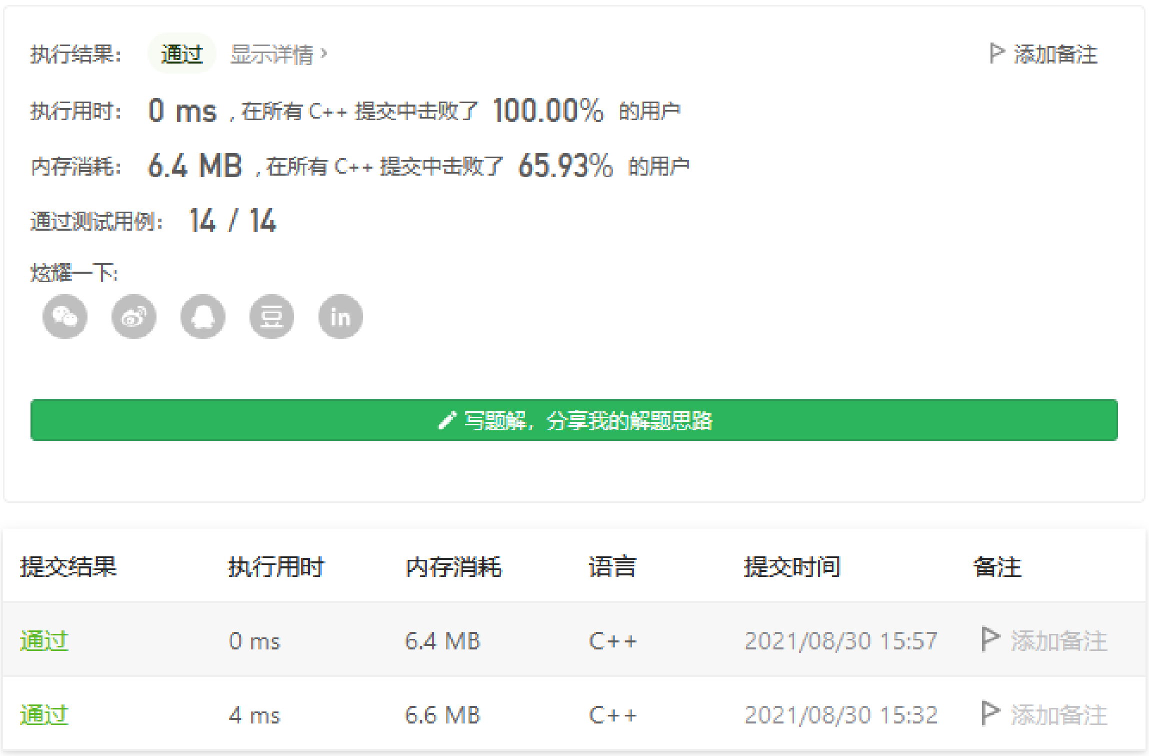
常规

```
class Solution {
public:
    vector<vector<int>> generate(int numRows)
    {
        //在杨辉三角中，每个数是它左上方和右上方的数字之和
        vector<vector<int>> ans(numRows);
        for(int i=0;i<numRows;++i)
        {
            ans[i].resize(i+1);    //重塑那一行的元素个数
            ans[i][0]=ans[i][i]=1; //每一行的左右边界值都为1
            for(int j=1;j<i;++j)
            {
                ans[i][j]=ans[i-1][j]+ans[i-1][j-1];
            }
        }
        return ans;
    }
};
```

暴力解法

```
class Solution {
public:
    vector<vector<int>> generate(int numRows) {
        vector<vector<int>> res={
        {1},
        {1,1},
        {1,2,1},
        {1,3,3,1},
        {1,4,6,4,1},
        {1,5,10,10,5,1},
        {1,6,15,20,15,6,1},
        {1,7,21,35,35,21,7,1},
        {1,8,28,56,70,56,28,8,1},
        {1,9,36,84,126,126,84,36,9,1},
        {1,10,45,120,210,252,210,120,45,10,1},
        {1,11,55,165,330,462,462,330,165,55,11,1},
        {1,12,66,220,495,792,924,792,495,220,66,12,1},
        {1,13,78,286,715,1287,1716,1716,1287,715,286,78,13,1},
        {1,14,91,364,1001,2002,3003,3432,3003,2002,1001,364,91,14,1},
        {1,15,105,455,1365,3003,5005,6435,6435,5005,3003,1365,455,105,15,1},

{1,16,120,560,1820,4368,8008,11440,12870,11440,8008,4368,1820,560,120,16,1},

{1,17,136,680,2380,6188,12376,19448,24310,24310,19448,12376,6188,2380,680,136,17
,1},

{1,18,153,816,3060,8568,18564,31824,43758,48620,43758,31824,18564,8568,3060,816,
153,18,1},

{1,19,171,969,3876,11628,27132,50388,75582,92378,92378,75582,50388,27132,11628,3
876,969,171,19,1},
{1,20,190,1140,4845,15504,38760,77520,125970,167960,184756,167960,125970,77520,38760,15504,4845,1140,190,20,1},

{1,21,210,1330,5985,20349,54264,116280,203490,293930,352716,352716,293930,203490,116280,54264,20349,5985,1330,210,21,1},

{1,22,231,1540,7315,26334,74613,170544,319770,497420,646646,705432,646646,497420,319770,170544,74613,26334,7315,1540,231,22,1},

{1,23,253,1771,8855,33649,100947,245157,490314,817190,1144066,1352078,1352078,1144066,817190,490314,245157,100947,33649,8855,1771,253,23,1},

{1,24,276,2024,10626,42504,134596,346104,735471,1307504,1961256,2496144,2704156,2496144,1961256,1307504,735471,346104,134596,42504,10626,2024,276,24,1},

{1,25,300,2300,12650,53130,177100,480700,1081575,2042975,3268760,4457400,5200300,5200300,4457400,3268760,2042975,1081575,480700,177100,53130,12650,2300,300,25,1},

{1,26,325,2600,14950,65780,230230,657800,1562275,3124550,5311735,7726160,9657700,10400600,9657700,7726160,5311735,3124550,1562275,657800,230230,65780,14950,2600,325,26,1},

{1,27,351,2925,17550,80730,296010,888030,2220075,4686825,8436285,13037895,17383860,20058300,20058300,17383860,13037895,8436285,4686825,2220075,888030,296010,80730,17550,2925,351,27,1},

{1,28,378,3276,20475,98280,376740,1184040,3108105,6906900,13123110,21474180,30421755,37442160,40116600,37442160,30421755,21474180,13123110,6906900,3108105,1184040,376740,98280,20475,3276,378,28,1},

{1,29,406,3654,23751,118755,475020,1560780,4292145,10015005,20030010,34597290,51895935,67863915,77558760,77558760,67863915,51895935,34597290,20030010,10015005,4292145,1560780,475020,118755,23751,3654,406,29,1}};
        res.resize(numRows);
        return res;
    }
};
```

AC


执行结果： 通过 显示详情 ›

添加备注

执行用时： 0 ms ，在所有 C++ 提交中击败了 100.00% 的用户

内存消耗： 6.4 MB ，在所有 C++ 提交中击败了 65.93% 的用户

通过测试用例： 14 / 14

炫耀一下:

写题解，分享我的解题思路

提交结果	执行用时	内存消耗	语言	提交时间	备注
通过	0 ms	6.4 MB	C++	2021/08/30 15:57	添加备注
通过	4 ms	6.6 MB	C++	2021/08/30 15:32	添加备注