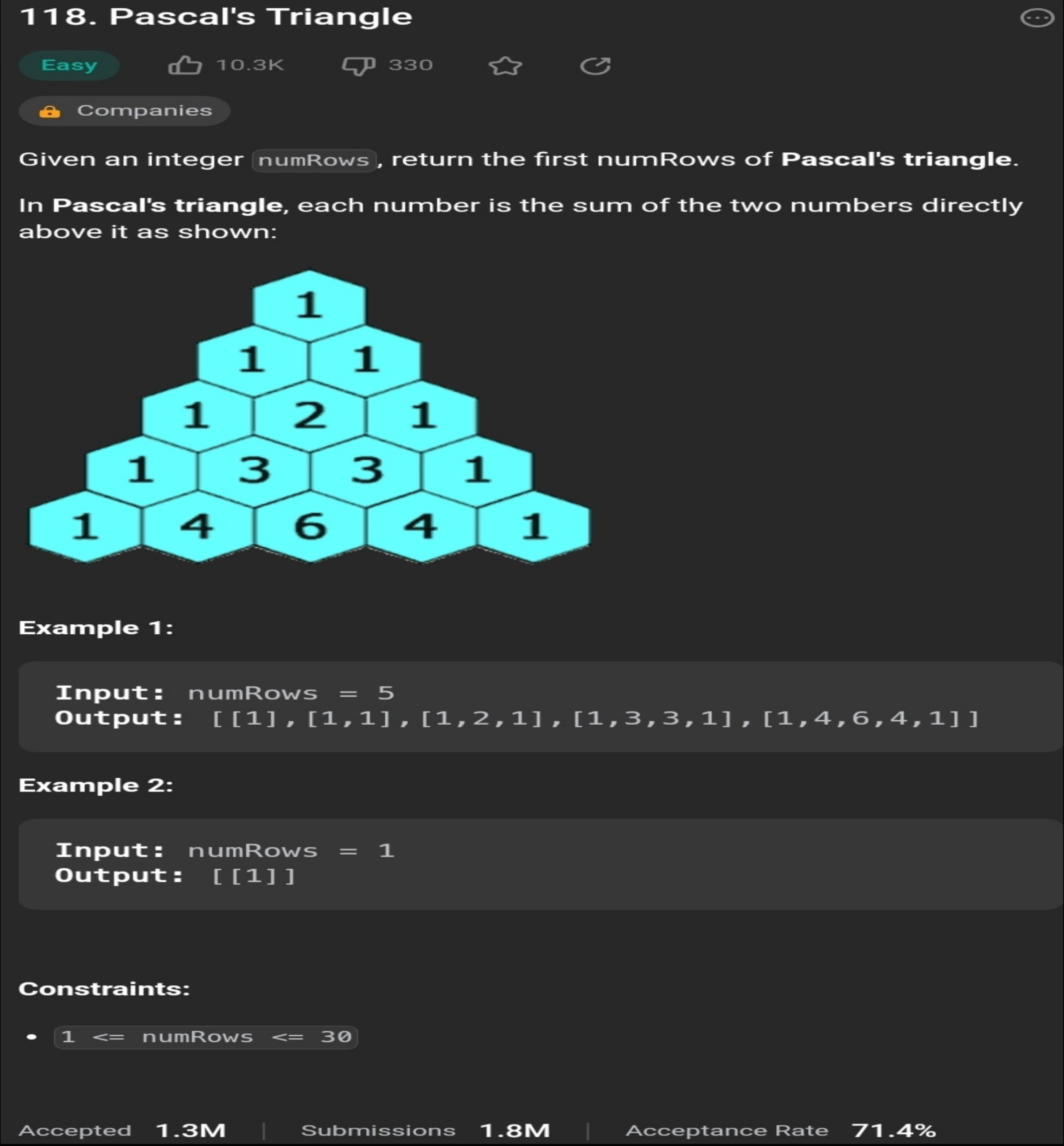


118. Pascal's Triangle

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]]
```

Example 2:

```
Input: numRows = 1
Output: [[1]]
```

Constraints:

- `1 <= numRows <= 30`

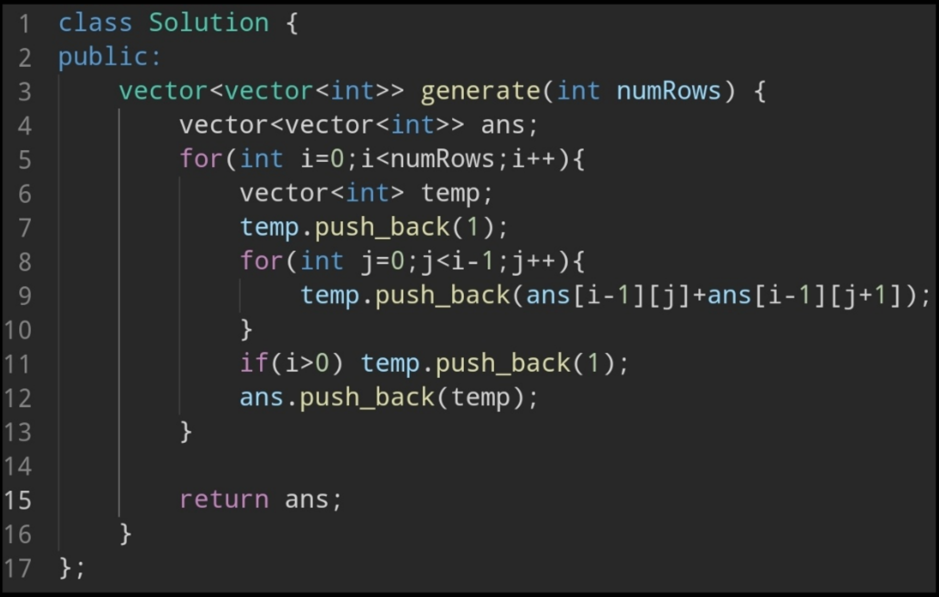
Accepted 1.3M | Submissions 1.8M | Acceptance Rate 71.4%

numRows = 5

	$\overset{0}{i}$	Size	$\overset{0}{j}$
{ 1 }	0	1	-1
{ 1 , 1 }	1	2	0
{ 1 , 2 , 1 }	2	3	1

{1, 3, 3, 1}   3   4   2

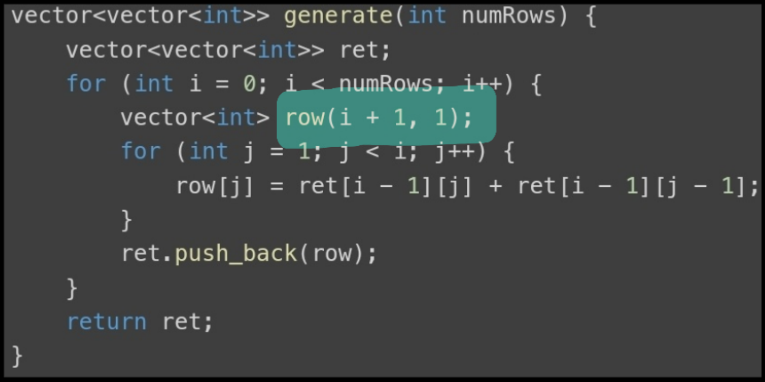
{1, 4, 6, 4, 1}   4   5   3



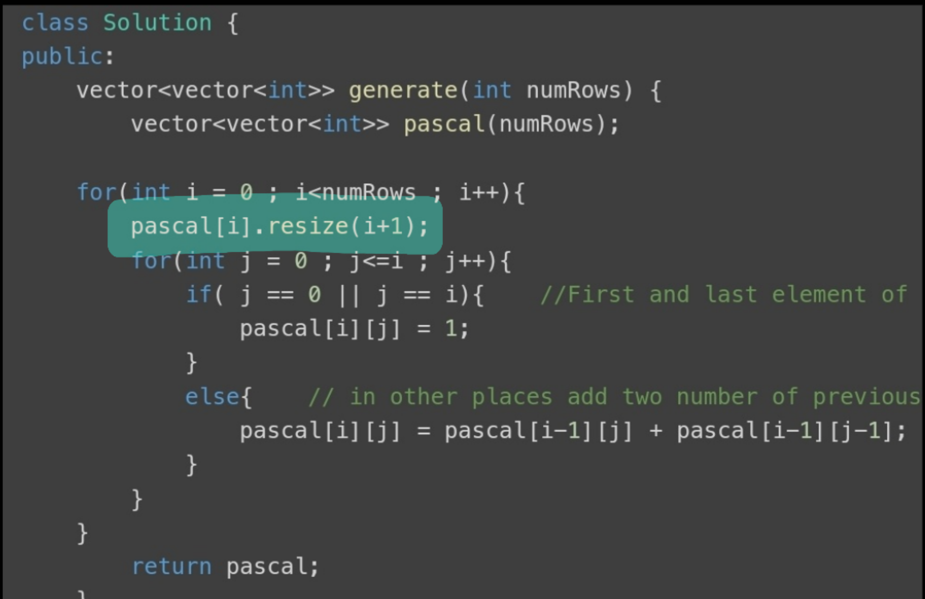
```
class Solution {
public:
    vector<vector<int>> generate(int numRows) {
        vector<vector<int>> ans;
        for(int i=0;i<numRows;i++){
            vector<int> temp;
            temp.push_back(1);
            for(int j=0;j<i-1;j++){
                temp.push_back(ans[i-1][j]+ans[i-1][j+1]);
            }
            if(i>0) temp.push_back(1);
            ans.push_back(temp);
        }

        return ans;
    }
};
```

Some more cleaner and simple codes:



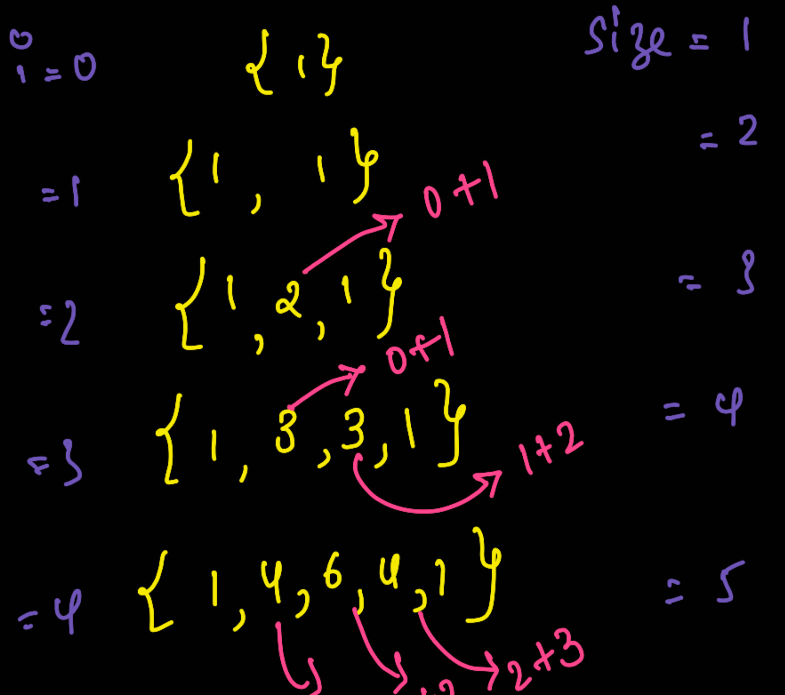
```
vector<vector<int>> generate(int numRows) {
    vector<vector<int>> ret;
    for (int i = 0; i < numRows; i++) {
        vector<int> row(i + 1, 1);
        for (int j = 1; j < i; j++) {
            row[j] = ret[i - 1][j] + ret[i - 1][j - 1];
        }
        ret.push_back(row);
    }
    return ret;
}
```



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

    for(int i = 0 ; i<numRows ; i++){
        pascal[i].resize(i+1);
        for(int j = 0 ; j<=i ; j++){
            if( j == 0 || j == i){    //First and last element of
                pascal[i][j] = 1;
            }
            else{    // in other places add two number of previous
                pascal[i][j] = pascal[i-1][j] + pascal[i-1][j-1];
            }
        }
    }
        return pascal;
    }
```

```
    }
};
```



i = 0 {1} size = 1
= 1 {1, 1} = 2
= 2 {1, 2, 1} = 3
0+1
= 3 {1, 3, 3, 1} = 4
0+1
1+2
= 4 {1, 4, 6, 4, 1} = 5
2+3

0+1 1+2

numRows = 5