3.Hard\01.Pascal_triangle.cpp

```cpp
/*
**Question:**

Given an integer `rowIndex`, return the `rowIndex`th (0-indexed) row of Pascal's triangle.

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

Example:

Input: `rowIndex = 3`

Output: `[1, 3, 3, 1]`
*/

/*
**APPROACH:**
To generate the `rowIndex`th row of Pascal's triangle, we can use the property that each number is the sum of the two numbers directly above it. We start with the base case of the first row, which is `[1]`. Then, for each subsequent row, we calculate the elements using the formula `C(n, k) = C(n-1, k-1) * (n-k+1) / k`, where `C(n, k)` represents the binomial coefficient.

**CODE:**
*/

vector<int> getRow(int rowIndex) {
    vector<int> row(rowIndex + 1, 1); // Initialize the row with 1s
    long long coefficient = 1;

    for (int col = 1; col <= rowIndex; col++) {
        coefficient = coefficient * (rowIndex - col + 1) / col;
        row[col] = coefficient;
    }

    return row;
}

/*
**COMPLEXITY ANALYSIS:**
- Time Complexity: O(rowIndex)
  - We iterate over each element in the row and calculate its value using the binomial coefficient formula.
- Space Complexity: O(rowIndex)
  - We use additional space to store the row of Pascal's triangle.

Overall, the algorithm has a linear time complexity and linear space complexity.
*/

```