Documentation for Pascal's Triangle II Problem

Problem Statement

- 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 1

Input: rowIndex = 3

Output: [1, 3, 3, 1]

Example 2

Input: rowIndex = 0

Output: [1]

Example 3

Input: rowIndex = 1

Output: [1, 1]

Constraints

(0 leq text{rowIndex} leq 33)

Follow Up

- Could you optimize your algorithm to use only (O(text{rowIndex})) extra space?

Solution

The solution involves generating the rowIndex-th row of Pascal's triangle using an iterative approach. The algorithm uses a list to store the current row of the triangle and updates it in place to save space.

Explanation

1. **Initialization:**

- Create a list row of length rowIndex + 1 with all elements initialized to 1. This represents the row of Pascal's triangle we want to generate.

2. **Iterative Update:**

- For each index i from 1 to rowIndex 1 (inclusive), update the row from right to left:
- For each index j from i to 1 (inclusive), update row[j] by adding the value of row[j 1] to it. This ensures that the values are correctly updated based on the previous row of Pascal's triangle.

3. **Return the Result:**

- After the iterations are complete, return the row list, which now contains the rowIndex-th row of Pascal's triangle.

This approach ensures that we use only (O(text{rowIndex})) extra space, as the list row is updated in place.