🌟 LeetCode 118 – Pascal's Triangle

🧩 Problem Overview

Given an integer numRows, generate the first numRows of **Pascal's Triangle**.

Each number in Pascal's Triangle is the sum of the two numbers directly above it. The triangle starts with 1 at the top and continues with each row having one more element than the previous.

💡 Intuition

Pascal's Triangle has the following properties:

- The **first and last** elements of each row are always **1**.
- Every **inner element** at position j in row i is the **sum** of the elements at positions j-1 and j from the previous row (i - 1).

🧠 Solution Explanation (C++)

🔹 1. Initialization

```
vector<vector<int>> v;

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

   vector<int> a(i + 1);    // Create a row with (i+1) elements

   v.push_back(a);          // Add it to the triangle

}
```

✅ Creates a 2D vector v, where each row has i + 1 elements
✅ This sets up the structure of the triangle

🔹 2. Filling the Triangle

```
for(int i = 0; i < numRows; i++) {

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

      if(j == 0 || j == i) {

         v[i][j] = 1; // First and last elements are always 1

      } else {

         v[i][j] = v[i-1][j-1] + v[i-1][j]; // Inner elements

      }
```

```
    }
}
```

- **Edge cases** (j == 0 || j == i) are filled with 1
- **Other elements** are calculated by summing two numbers from the row above

---

🖨️ 3. Example Output for numRows = 5

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

Each row has one more element than the previous and follows the Pascal's Triangle rule.

⏱️ Time & Space Complexity

Complexity	Explanation
Time	$O(n^2)$ – Each element is computed once
Space	$O(n^2)$ – All rows are stored in a 2D vector

🔑 Key Takeaways

- 🟧 Pascal's Triangle builds naturally from top to bottom
- 🟦 First and last elements of each row are **always 1**
- 🟥 Inner elements are the **sum of two above elements**
- 📈 Very efficient solution using dynamic programming principles

---

Would you like a **Python version** or a **visual diagram** to go along with it too?