



## WORKSHEET 9

**Student Name:** Parth Arora

**UID:** 22BCS16661

**Branch:** BE-CSE

**Section/Group:** 22BCS\_NTPP-602-A

**Semester:** 6<sup>th</sup>

**Date of Performance:** 3/04/2025

**Subject Name:** AP LAB - II

**Subject Code:** 22CSP-351

1. **Aim:** Given an integer numRows, return the first numRows of **Pascal's triangle**.

2. **Source Code:**

```
class Solution:
    def generate(self, numRows: int) -> List[List[int]]:
        res = [[1]]
        for _ in range(numRows - 1):
            dummy_row = [0] + res[-1] + [0]
            row = []
            for i in range(len(res[-1]) + 1):
                row.append(dummy_row[i] + dummy_row[i+1])
            res.append(row)

        return res
```

3. **Screenshots of outputs:**

The screenshot displays a coding platform interface. On the left, the submission status is 'Accepted' with 30/30 testcases passed. The runtime is 0 ms and memory usage is 17.71 MB. The right panel shows the Python code for generating Pascal's triangle. The test result for Case 1 is 'Accepted' with a runtime of 0 ms. The input for Case 1 is numRows = 5.

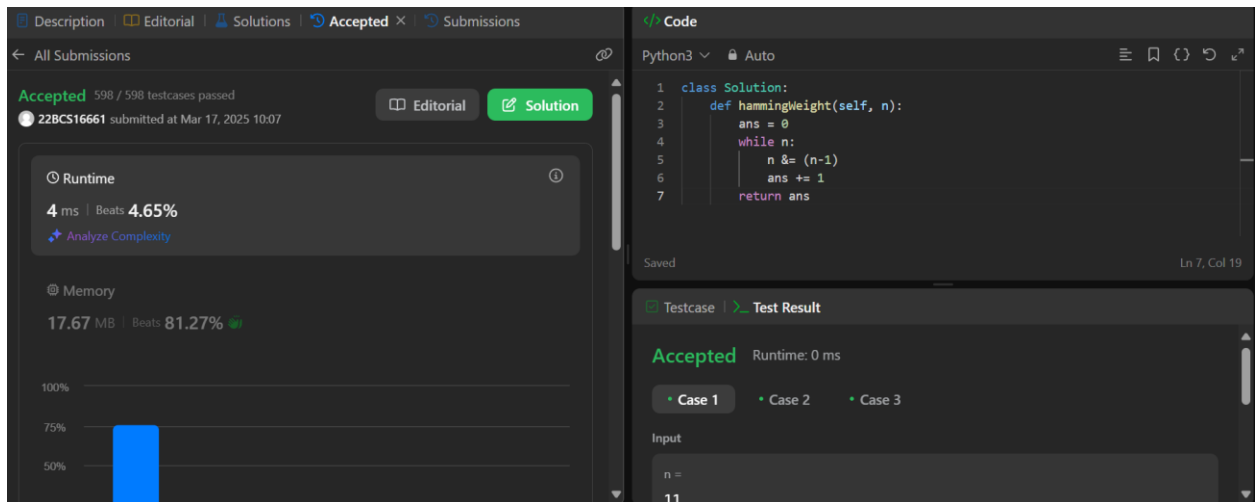
2.

**Aim:** Given a positive integer  $n$ , write a function that returns the number of set bits in its binary representation (also known as the [Hamming weight](#)).

## Source Code:

```
class TreeNode:
class Solution:
    def hammingWeight(self, n):
        ans = 0
        while n:
            n &= (n-1)
            ans += 1
        return ans
```

## Screenshots of outputs:



The screenshot displays a code editor interface for a problem solution. The top navigation bar includes tabs for Description, Editorial, Solutions, Accepted (598 / 598 testcases passed), and Submissions. The main content area shows the solution code in Python3, which defines a class Solution with a method hammingWeight. The code is as follows:

```
1 class Solution:
2     def hammingWeight(self, n):
3         ans = 0
4         while n:
5             n &= (n-1)
6             ans += 1
7         return ans
```

Below the code editor, the Testcase and Test Result section shows the solution is Accepted with a Runtime of 0 ms. The input for the test case is n = 11.

3.

**Aim:** Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

## Source Code:

```
class Solution:
    def trap(self, height: List[int]) -> int:
        n = len(height)
        result = 0
        maxI, waterBlock = 0, 0
        for i in range(1, n):
            if height[i] >= height[maxI]:
                result += waterBlock
                waterBlock = 0
                maxI = i
            waterBlock += (height[maxI] - height[i])
        end = maxI - 1
        maxI, waterBlock = n - 1, 0
        for i in range(n - 2, end, -1):
            if height[i] >= height[maxI]:
                result += waterBlock
                waterBlock = 0
                maxI = i
            waterBlock += (height[maxI] - height[i])
        return result
```

## 4. Screenshots of outputs:

