Assignment 10

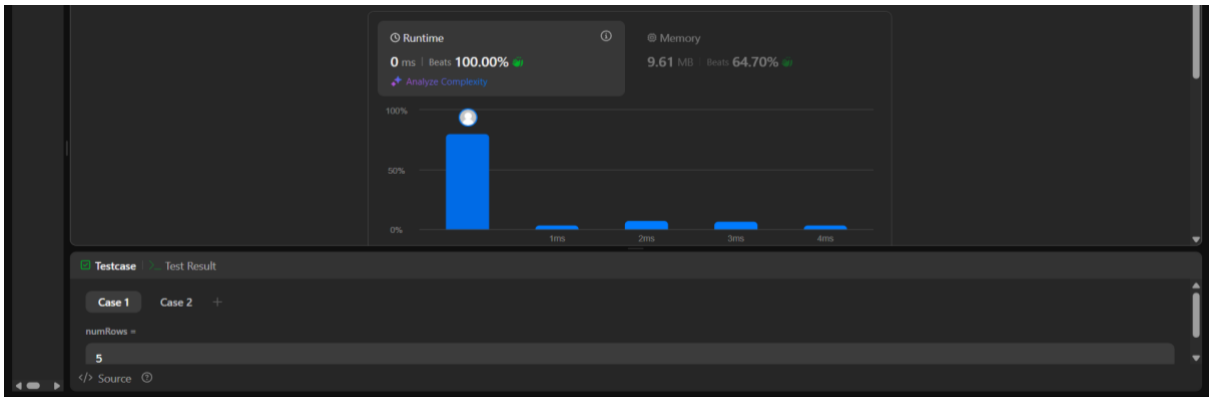
Name – Nikhil Kumar Tiwari **Uid – 22BCS10471**

Problem 1: Pascals Triangle (https://leetcode.com/problems/pascals-triangle/ **)**

Code:

```
class Solution {

public:

  vector<vector<int>> generate(int numRows) {

    vector<vector<int>> triangle(numRows);

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

      triangle[i].resize(i + 1);

      triangle[i][0] = triangle[i][i] = 1;


      for (int j = 1; j < i; ++j) {

        triangle[i][j] = triangle[i - 1][j - 1] + triangle[i - 1][j];

      }

    }

    return triangle;

  }

};
```

Screenshot:

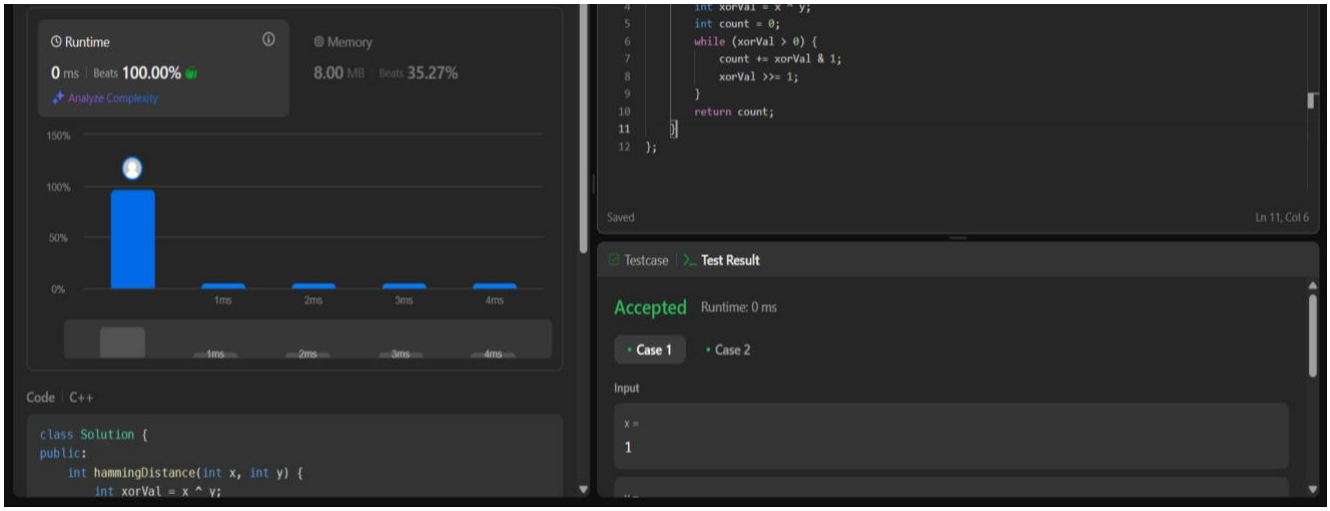


Problem 2: Hamming Distance (https://leetcode.com/problems/hamming-distance/ **)**

Code:

```
class Solution {

public:

   int hammingDistance(int x, int y) {

      int xorVal = x ^ y;

      int count = 0;

      while (xorVal > 0) {

         count += xorVal & 1;

         xorVal >>= 1;

      }

      return count;

   }

};
```

Screenshot:

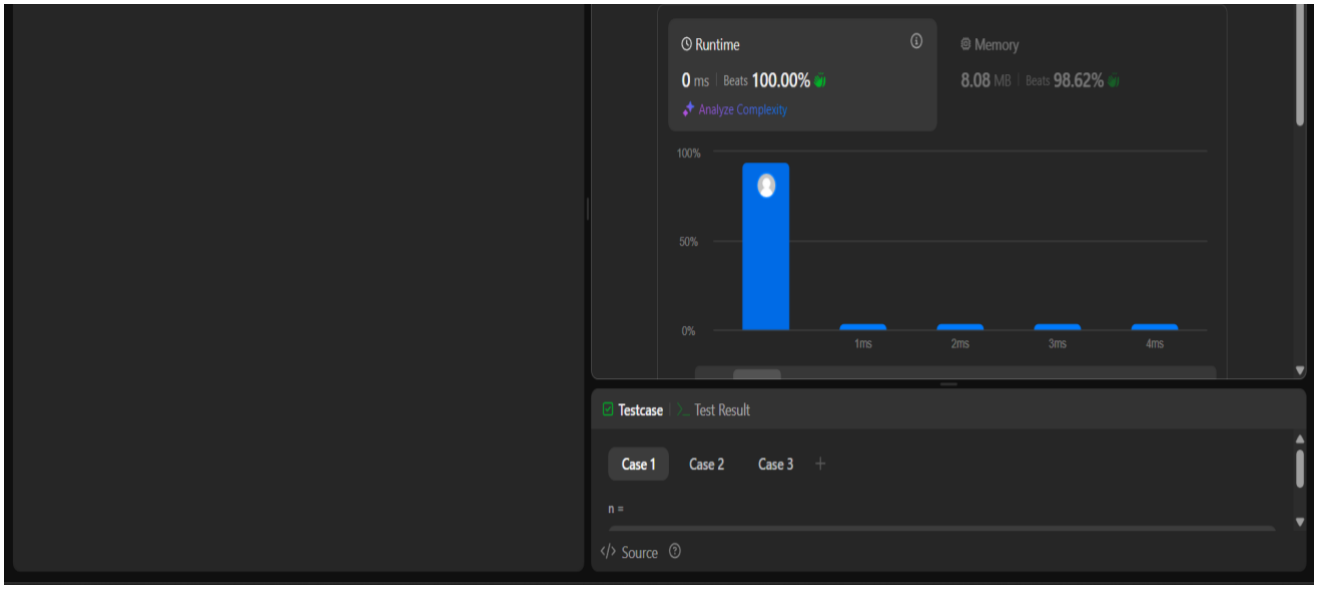


Problem 3: Number of 1 bits (https://leetcode.com/problems/number-of-1-bits/ **)**

Code:

```
class Solution {

public:

  int hammingWeight(int n) {

    int count = 0;

    while(n) {

      n = n & (n-1);

      count++;

    }

    return count;

  }

};
```

Screenshot:

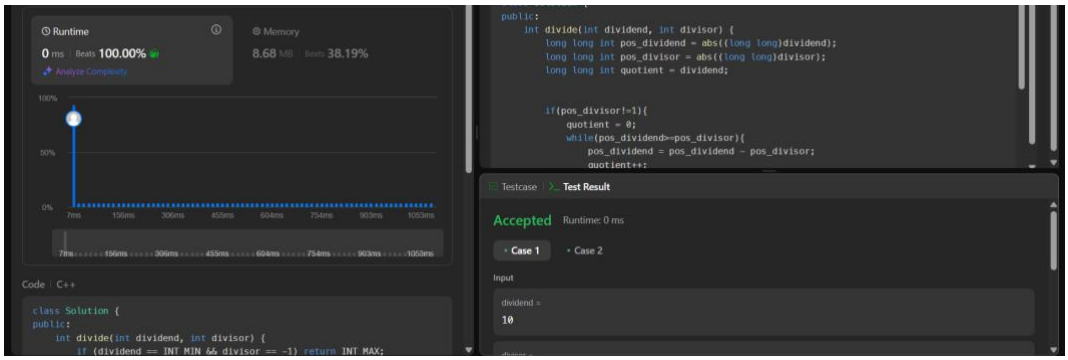


Problem 4: Divide Two Integers (https://leetcode.com/problems/divide-two-integers/ **)**

Code:

```
if (dividend == INT_MIN && divisor == -1) return INT_MAX;
    bool negative = (dividend < 0) ^ (divisor < 0);
    long long a = abs((long long)dividend);
    long long b = abs((long long)divisor);
    long long result = 0;

    while (a >= b) {
      long long temp = b, multiple = 1;
      while (a >= (temp << 1)) {
        temp <<= 1;
        multiple <<= 1;
      }
      a -= temp;
      result += multiple;
    }

    if (negative) result = -result;
    return result;
}
};
```

Screenshot:

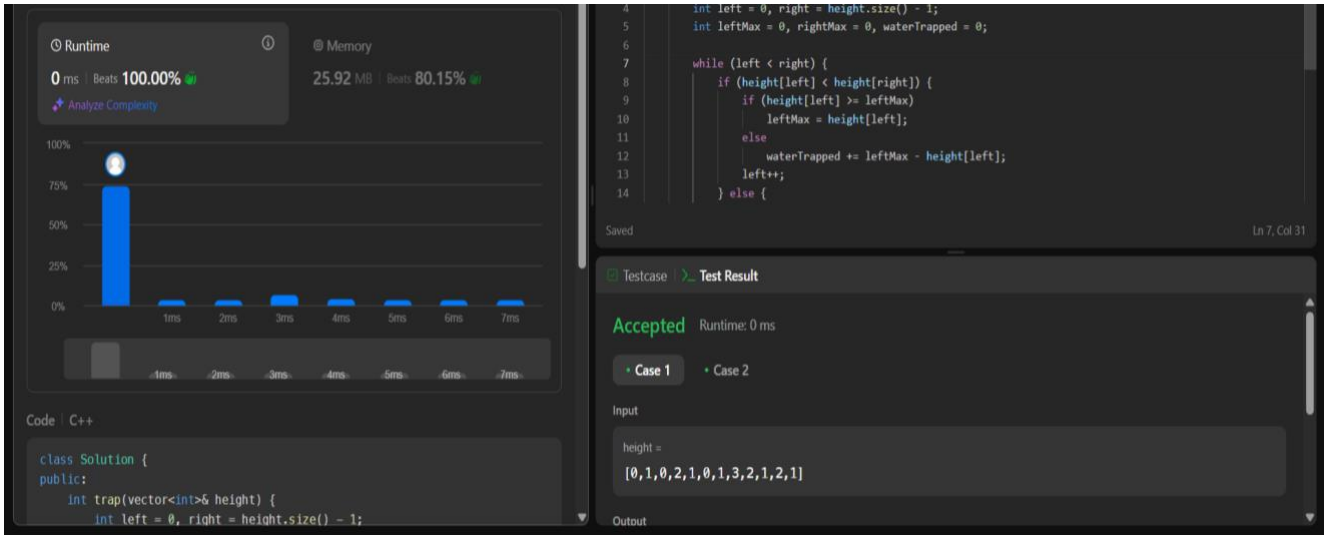


Problem 5: Trapping Rain Water (https://leetcode.com/problems/trapping-rain-water/**)**

Code:

```
class Solution {

public:

  int trap(vector<int>& height) {

    int left = 0, right = height.size() - 1;

    int leftMax = 0, rightMax = 0, waterTrapped = 0;


    while (left < right) {

      if (height[left] < height[right]) {

        if (height[left] >= leftMax)

          leftMax = height[left];

        else

          waterTrapped += leftMax - height[left];

        left++;

      } else {

        if (height[right] >= rightMax)

          rightMax = height[right];

        else

          waterTrapped += rightMax - height[right];

        right--;

      }

    }

    return waterTrapped;

  }

};
```

Screenshot:


Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
25.92 MB | Beats 80.15%
100%
75%
50%
25%
0%
1ms
2ms
3ms
4ms
5ms
6ms
7ms
Code | C++
class Solution {
public:
int trap(vector<int>& height) {
int left = 0, right = height.size() - 1;
int leftMax = 0, rightMax = 0, waterTrapped = 0;
while (left < right) {
if (height[left] < height[right]) {
if (height[left] >= leftMax)
leftMax = height[left];
else
waterTrapped += leftMax - height[left];
left++;
} else {
Saved
Ln 7, Col 31
Testcase | Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
height =
[0,1,0,2,1,0,1,3,2,1,2,1]
Outout