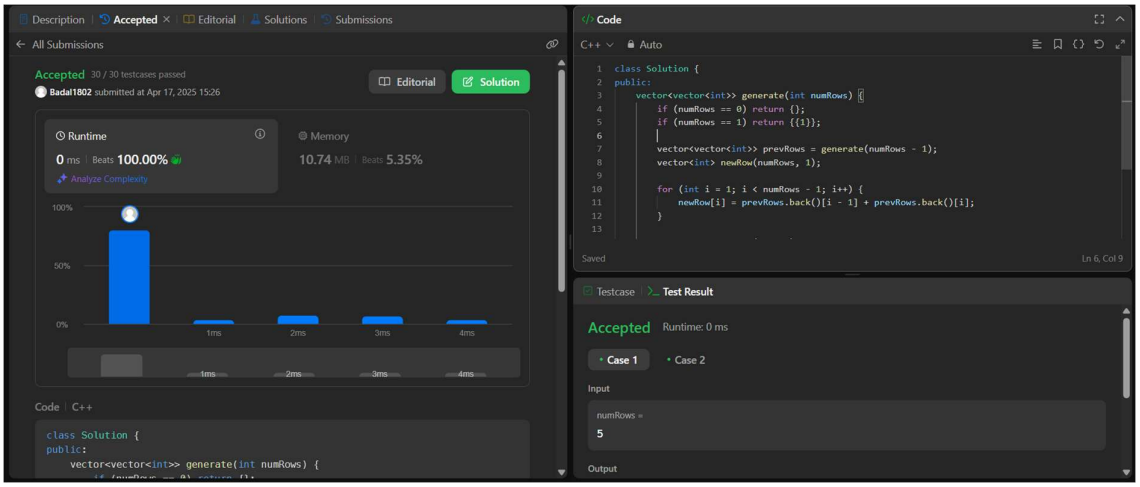
Experiment 10

Pascal's Triangle

Code:

```
class Solution {
public:
    vector<vector<int>> generate(int numRows) {
        if (numRows == 0) return {};
        if (numRows == 1) return {{1}};
        vector<vector<int>> prevRows = generate(numRows - 1);
        vector<int> newRow(numRows, 1);
        for (int i = 1; i < numRows - 1; i++) {
            newRow[i] = prevRows.back()[i - 1] + prevRows.back()[i];
        }
        prevRows.push_back(newRow);
        return prevRows;
    }
};
```

OUTPUT:

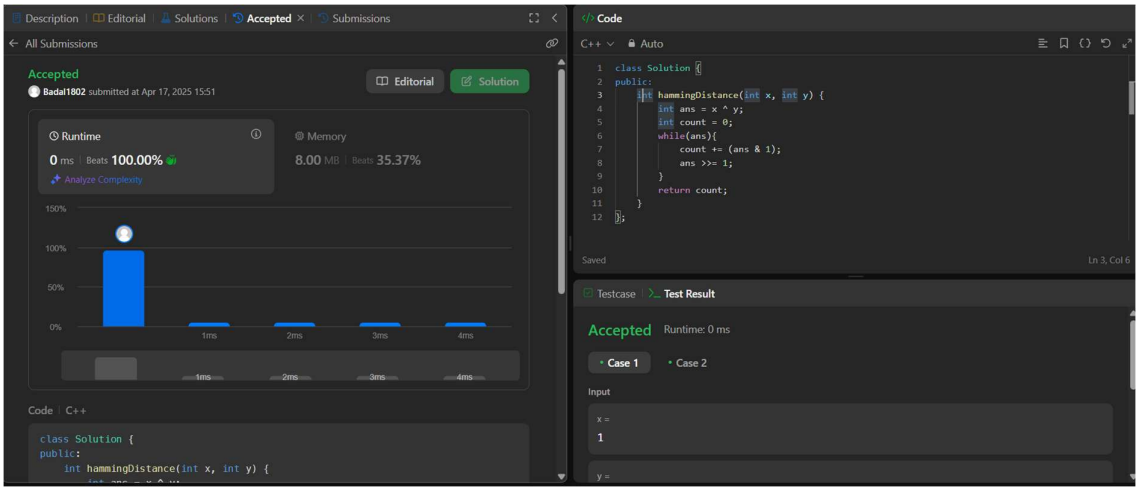


Hamming Distance

Code:

```
class Solution {
public:
    int hammingDistance(int x, int y) {
        int ans = x ^ y;
        int count = 0;
        while(ans){
            count += (ans & 1);
            ans >>= 1;
        }
        return count;
    }
};
```

OUTPUT:

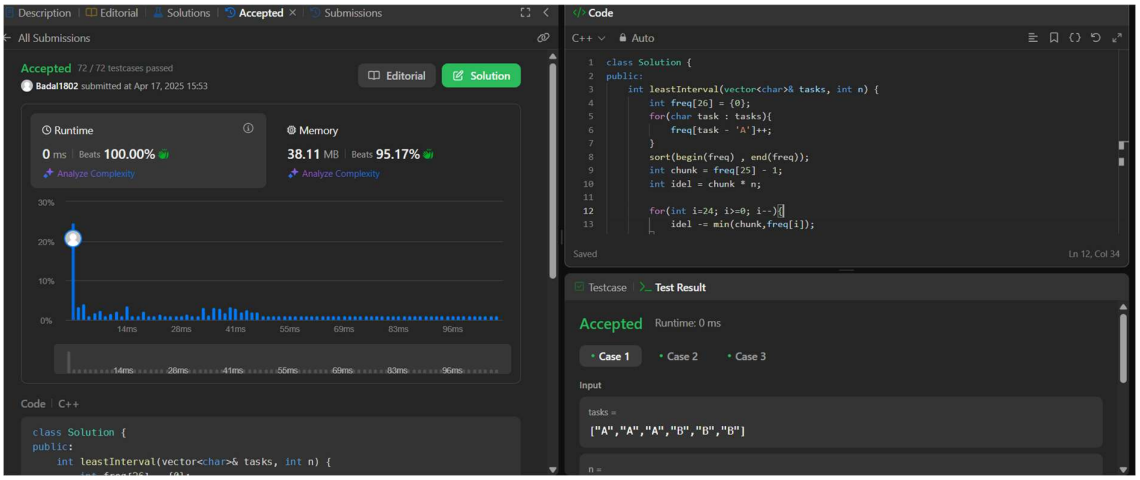


Task Scheduler

CODE:

```
class Solution {
public:
   int leastInterval(vector<char>& tasks, int n) {
      int freq[26] = {0};
      for(char task : tasks){
         freq[task - 'A']++;
      }
      sort(begin(freq) , end(freq));
      int chunk = freq[25] - 1;
      int idel = chunk * n;

      for(int i=24; i>=0; i--){
         idel -= min(chunk,freq[i]);
      }
      return idel < 0 ? tasks.size() : tasks.size() + idel;
   }
};
```

OUTPUT:

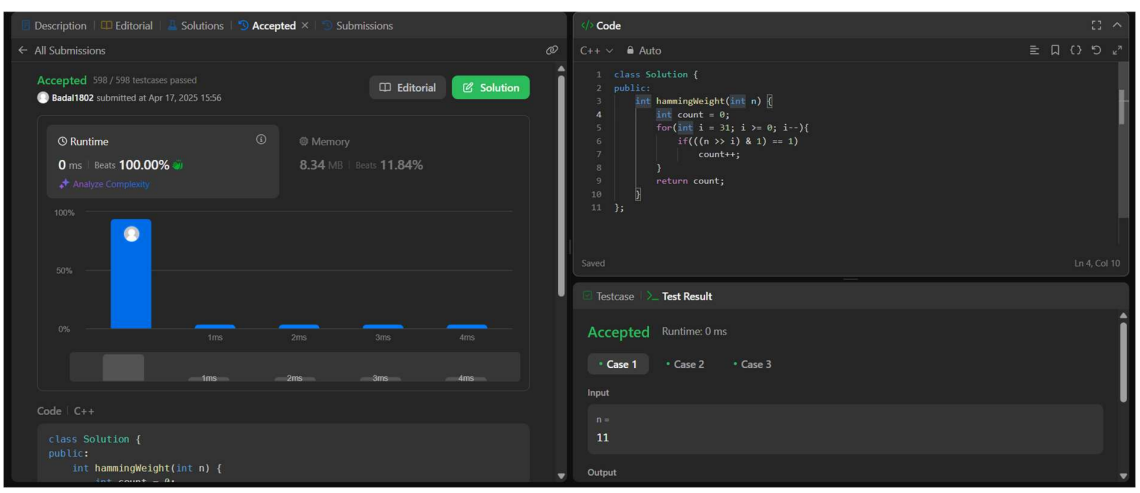


Number of 1 Bits

CODE:

```
class Solution {
public:
    int hammingWeight(int n) {
        int count = 0;
        for(int i = 31; i >= 0; i--){
            if(((n >> i) & 1) == 1)
                count++;
        }
        return count;
    }
};
```

OUTPUT:

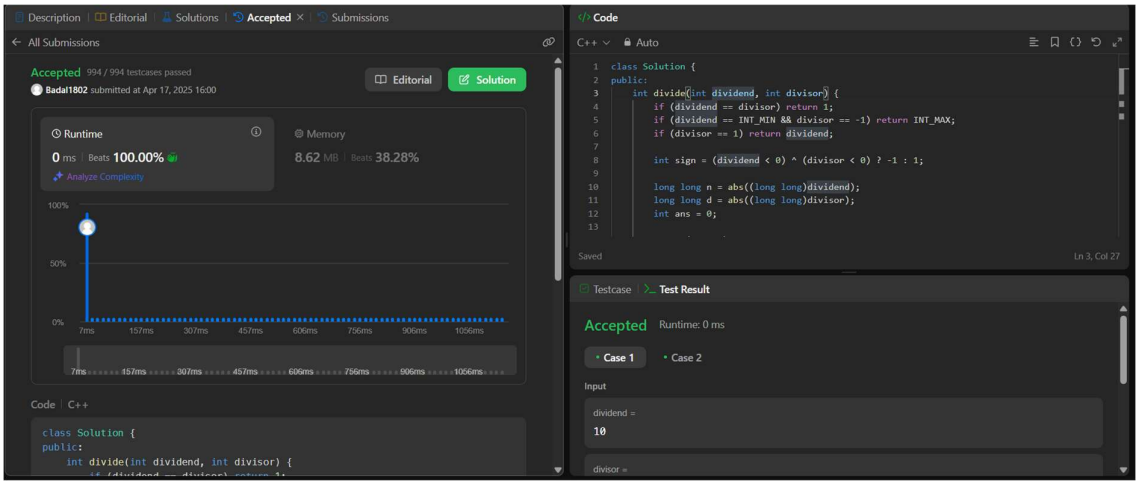


Divide Two Integers

CODE:

```
class Solution {
public:
    int divide(int dividend, int divisor) {
        if (dividend == divisor) return 1;
        if (dividend == INT_MIN && divisor == -1) return INT_MAX;
        if (divisor == 1) return dividend;

        int sign = (dividend < 0) ^ (divisor < 0) ? -1 : 1;

        long long n = abs((long long)dividend);
        long long d = abs((long long)divisor);
        int ans = 0;

        while (n >= d) {
            int p = 0;
            while (n >= (d << p)) p++;
            p--;
            n -= (d << p);
            ans += (1 << p);
        }

        return sign * ans;
    }
};
```

OUTPUT:


Description | Editorial | Solutions | Accepted | Submissions
All Submissions
Accepted 994 / 994 testcases passed
Badal1802 submitted at Apr 17, 2025 16:00
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
8.62 MB | Beats 38.28%
100%
50%
0%
7ms 157ms 307ms 457ms 606ms 756ms 906ms 1056ms
Code | C++
class Solution {
public:
int divide(int dividend, int divisor) {
Code
C++
Auto
class Solution {
public:
int divide(int dividend, int divisor) {
if (dividend == divisor) return 1;
if (dividend == INT_MIN && divisor == -1) return INT_MAX;
if (divisor == 1) return dividend;
int sign = (dividend < 0) ^ (divisor < 0) ? -1 : 1;
long long n = abs((long long)dividend);
long long d = abs((long long)divisor);
int ans = 0;
Saved
Ln 3, Col 27
Testcase | Test Result
Accepted Runtime: 0 ms
Case 1
Case 2
Input
dividend =
10
divisor =