

AP-10

Name: Harmandeep Singh

Section:614-B

UID:22BCS14975

1.Pascal's Triangle

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

    void printTriangle(const vector<vector<int>>>& triangle) {
        for (const auto& row : triangle) {
            for (int num : row) {
                cout << num << " ";
            }
            cout << endl;
        }
    }
};
```

The screenshot shows a C++ solution for a problem. The left sidebar displays the problem status as "Accepted" with 30/30 testcases passed. The runtime is 0 ms, beating 100.00% of other solutions. The memory usage is 9.78 MB, beating 33.62% of other solutions. A bar chart shows the runtime distribution across different time intervals. The main code editor shows a C++ solution for generating a Pascal's triangle.

```

class Solution {
public:
    vector<vector<int>> generate(int numRows) {
        vector<vector<int>> triangle;
        for (int i = 0; i < numRows; i++) {
            vector<int> row(i + 1, 1);
            for (int j = 1; j < i; j++) {
                row[j] = triangle[i - 1][j - 1] + triangle[i - 1][j];
            }
            triangle.push_back(row);
        }
        return triangle;
    }
};

void printTriangle(const vector<vector<int>>& triangle) {
    for (const auto& row : triangle) {
        for (int num : row) {
            cout << num << " ";
        }
    }
}

```

The test result shows the solution is "Accepted" with a runtime of 0 ms. The input is numRows = 5.

2.Hamming Distance

```

class Solution {
public:
    int hammingDistance(int x, int y) {
        int xorResult = x ^ y;
        return __builtin_popcount(xorResult);
    }
};

```

The screenshot shows the 461. Hamming Distance problem page on LeetCode. The problem description states: "The Hamming distance between two integers is the number of positions at which the corresponding bits are different. Given two integers x and y, return the Hamming distance between them." Example 1: Input: x = 1, y = 4, Output: 2. Example 2: Input: x = 3, y = 1, Output: 1. The constraints are 0 ≤ x, y ≤ 2³¹ - 1. The solution code is shown in the main editor, and the test result shows the solution is "Accepted" with a runtime of 0 ms.

```

class Solution {
public:
    int hammingDistance(int x, int y) {
        int xorResult = x ^ y;
        return __builtin_popcount(xorResult);
    }
};

```

The test result shows the solution is "Accepted" with a runtime of 0 ms. The input is x = 1, y = 4, and the output is 2.

3.Task Scheduler

```
class Solution {
public:
    int leastInterval(vector<char>& tasks, int n) {
        unordered_map<char, int> freq;
        for (char task : tasks) {
            freq[task]++;
        }
        priority_queue<int> pq;
        for (auto it : freq) {
            pq.push(it.second);
        }

        int totalTime = 0;

        while (!pq.empty()) {
            int time = 0;
            vector<int> temp;
            for (int i = 0; i <= n; i++) {
                if (!pq.empty()) {
                    temp.push_back(pq.top() - 1);
                    pq.pop();
                    time++;
                }
            }

            for (int count : temp) {
                if (count > 0) pq.push(count);
            }

            totalTime += pq.empty() ? time : (n + 1);
        }

        return totalTime;
    }
};
```

The screenshot shows a C++ solution for a problem. The left sidebar displays the submission status: "Accepted 72 / 72 testcases passed". The runtime is 35 ms, which is 39.05% better than the fastest solution. The memory usage is 47.72 MB, which is 19.34% better than the fastest solution. A bar chart shows the runtime distribution across different test cases. The main code editor shows the following C++ code:

```

1 class Solution {
2 public:
3     int leastInterval(vector<char>& tasks, int n) {
4         unordered_map<char, int> freq;
5         for (char task : tasks) {
6             freq[task]++;
7         }
8         priority_queue<int> pq;
9         for (auto it : freq) {
10             pq.push(it.second);
11         }
12
13         int totalTime = 0;
14
15         while (!pq.empty()) {
16             int time = 0;
17             vector<int> temp;
18             for (int i = 0; i <= n; i++) {
19                 if (!pq.empty()) {
20                     temp.push_back(pq.top() - 1);
21                     pq.pop();
22                     time++;

```

The test result section shows "Accepted" with a runtime of 0 ms. The input field is empty.

4. Number of 1's bits

```

class Solution {
public:
    int hammingWeight(int n) {
        int count = 0;
        while (n) {
            count++;
            n &= (n - 1);
        }
        return count;
    }
};

```

The screenshot shows a C++ solution for the "Number of 1's bits" problem. The left sidebar displays the submission status: "Accepted 598 / 598 testcases passed". The runtime is 0 ms, which is 100.00% better than the fastest solution. The memory usage is 8.36 MB, which is 11.84% better than the fastest solution. A bar chart shows the runtime distribution across different test cases. The main code editor shows the following C++ code:

```

1 class Solution {
2 public:
3     int hammingWeight(int n) {
4         int count = 0;
5         while (n) {
6             count++;
7             n &= (n - 1);
8         }
9         return count;
10    }
11 };

```

The test result section shows "Accepted" with a runtime of 0 ms. The input field is empty.

5. Valid Parenthesis

```

public:

```

```

bool isValid(string s) {
    std::unordered_map<char, char> bracketPairs = {
        {'}', '{'},
        {'}', '{'},
        {'}', '{'}
    };
    std::stack<char> openBrackets;

    for (char c : s) {
        if (bracketPairs.count(c)) {
            if (!openBrackets.empty() && openBrackets.top() == bracketPairs[c]) {
                openBrackets.pop();
            } else {
                return false;
            }
        } else {
            openBrackets.push(c);
        }
    }

    return openBrackets.empty();
}

```

Problem List < > >

Description Accepted Editorial Solutions S

All Submissions

Accepted 100 / 100 testcases passed
Harman... submitted at Apr 17, 2025 23:46

Runtime
0 ms | Beats 100.00%
Analyze Complexity

Memory
9.04 MB | Beats 15.78%

Code C++

```

class Solution {
public:
    bool isValid(string s) {

```

```

2 public:
3 bool isValid(string s) {
4     std::unordered_map<char, char> bracketPairs = {
5         {'}', '{'},
6         {'}', '{'},
7         {'}', '{'}
8     };
9     std::stack<char> openBrackets;
10
11     for (char c : s) {
12
13         if (bracketPairs.count(c)) {
14
15             if (!openBrackets.empty() && openBrackets.top() == bracketPairs[c]) {
16                 openBrackets.pop();
17             } else {
18                 return false;
19             }
20         } else {
21             openBrackets.push(c);
22         }
23     }
24
25     return openBrackets.empty();
26 }
27
28 }
29

```

Saved

Ln 16, Col 37

Testcase > Test Result

6.Divide Two Integer

```

class Solution {

```

```

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

long long a = abs((long long)dividend);
long long b = abs((long long)divisor);
int result = 0;

while (a >= b) {
long long temp = b, multiple = 1;

while (a >= (temp << 1)) {
temp <<= 1;
multiple <<= 1;
}

a -= temp;
result += multiple;
}

if ((dividend > 0) ^ (divisor > 0))
result = -result;

return result;
}
};

```

The screenshot displays a C++ IDE with the following components:

- Problem List:** Shows the problem is 'Accepted' with 30/30 testcases passed. The user 'Harman...' submitted it at Apr 17, 2025 23:30.
- Runtime:** Shows a runtime of 0 ms, which is 100.00% faster than other solutions. Memory usage is 9.78 MB, which is 33.62% less than other solutions.
- Code:** The code defines a class 'Solution' with a method 'generate(int numRows)' that returns a vector of vectors of integers representing a Pascal's triangle. It also includes a 'printTriangle' method for visualization.
- Testcase:** The input is 'numRows = 5'. The output is '5'.
- Test Result:** The solution is 'Accepted' with a runtime of 0 ms.

7.Trapping Rain Water

```

class Solution {
public:
int trap(vector<int>& height) {
int left = 0, right = height.size() - 1;

```

```

int leftMax = 0, rightMax = 0;
int 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;
}
};

```

The screenshot displays a LeetCode solution for the "Trapping Water" problem. The interface includes a top navigation bar with options like "Problem List", "Run", "Submit", and "Premium". The main content area is divided into several sections:

- Problem List:** Shows the problem name "Trapping Water" and its status "Accepted". It also displays the user's submission details: "Harma..." submitted at Apr 18, 2025 00:00.
- Runtime:** Shows a bar chart indicating the execution time. The runtime is 0 ms, which is 100.00% faster than other solutions. The memory usage is 25.88 MB, which is 93.52% better than other solutions.
- Code:** Displays the C++ code solution. The code defines a class "Solution" with a method "trap" that calculates the total water trapped. The code uses a two-pointer approach to find the maximum height on the left and right sides of the current element and calculates the water trapped as the difference between the maximum height and the current element's height.
- Testcase:** Shows the test results for the solution. The test case is "Case 1" and the result is "Accepted". The input is "height = [0,1,0,2,1,0,1,3,2,1,2,1]" and the output is "6".