

AP-10

Name: Aman Raj

Section:614-B

UID:22BCS10078

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 in a code editor. The left sidebar displays the 'Runtime' and 'Memory' sections. The 'Runtime' section shows '0 ms' and 'Beats 100.00%'. The 'Memory' section shows '9.78 MB' and 'Beats 33.62%'. A bar chart below these sections shows the runtime distribution. The code in the editor is 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 << " ";
        }
    }
}
```

2.Hamming Distance

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

The screenshot shows the LeetCode problem page for '461. Hamming Distance'. 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.' The code in the editor is a C++ solution for calculating the Hamming distance.

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

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;
}
};
```

Accepted 72 / 72 testcases passed
Harman... submitted at Apr 17, 2025 23:39

Runtime
35 ms | Beats 39.05%

Memory
47.72 MB | Beats 19.34%

Code | C++

```
class Solution {
public:
    int leastInterval(vector<char>& tasks, int n) {
```

Testcase Test Result
Accepted Runtime: 0 ms
Case 1 Case 2 Case 3

4. Number of 1's bits

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

Accepted 598 / 598 testcases passed
Harman... submitted at Apr 17, 2025 23:41

Runtime
0 ms | Beats 100.00%

Memory
8.36 MB | Beats 11.84%

Code | C++

```
class Solution {
public:
    int hammingWeight(int n) {
```

Testcase Test Result
Accepted Runtime: 0 ms
Case 1 Case 2 Case 3

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();
}
};

```

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++ code editor with the following code:

```

1 class Solution {
2 public:
3     vector<vector<int>> generate(int numRows) {
4         vector<vector<int>> triangle;
5         for (int i = 0; i < numRows; i++) {
6             vector<int> row(i + 1, 1);
7             for (int j = 1; j < i; j++) {
8                 row[j] = triangle[i - 1][j - 1] + triangle[i - 1][j];
9             }
10            triangle.push_back(row);
11        }
12        return triangle;
13    }
14    void printTriangle(const vector<vector<int>>& triangle) {
15        for (const auto& row : triangle) {
16            for (int num : row) {
17                cout << num << " ";
18            }
19        }
20    }
21 }

```

The editor shows the code is accepted with a runtime of 0 ms and memory of 9.78 MB. The test result for Case 1 is 'Accepted'.

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 coding platform interface for a C++ solution. The top navigation bar includes 'Problem List', 'Run', 'Submit', and 'Premium' options. The main area is divided into several sections:

- Problem List:** Shows 'Accepted' status with 324/324 testcases passed. The user 'Harma...' submitted the solution on Apr 18, 2025 at 00:00. A 'Solution' button is visible.
- Runtime:** Displays '0 ms | Beats 100.00%'. A link to 'Analyze Complexity' is provided.
- Memory:** Shows '25.88 MB | Beats 93.52%'.
- Bar Chart:** A chart showing the distribution of runtime performance across different percentiles, with a blue bar indicating the current solution's performance.
- Code Editor:** Contains the following C++ code:


```

1 class Solution {
2 public:
3     int trap(vector<int>& height) {
4         int left = 0, right = height.size() - 1;
5         int leftMax = 0, rightMax = 0;
6         int waterTrapped = 0;
7
8         while (left < right) {
9             if (height[left] < height[right]) {
10                 if (height[left] >= leftMax)
11                     leftMax = height[left];
12                 else

```
- Testcase / Test Result:** Shows 'Accepted' status with 'Runtime: 0 ms'. It includes a 'Case 1' tab with the following input and output:

Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]

Output: 6

Expected: (empty field)