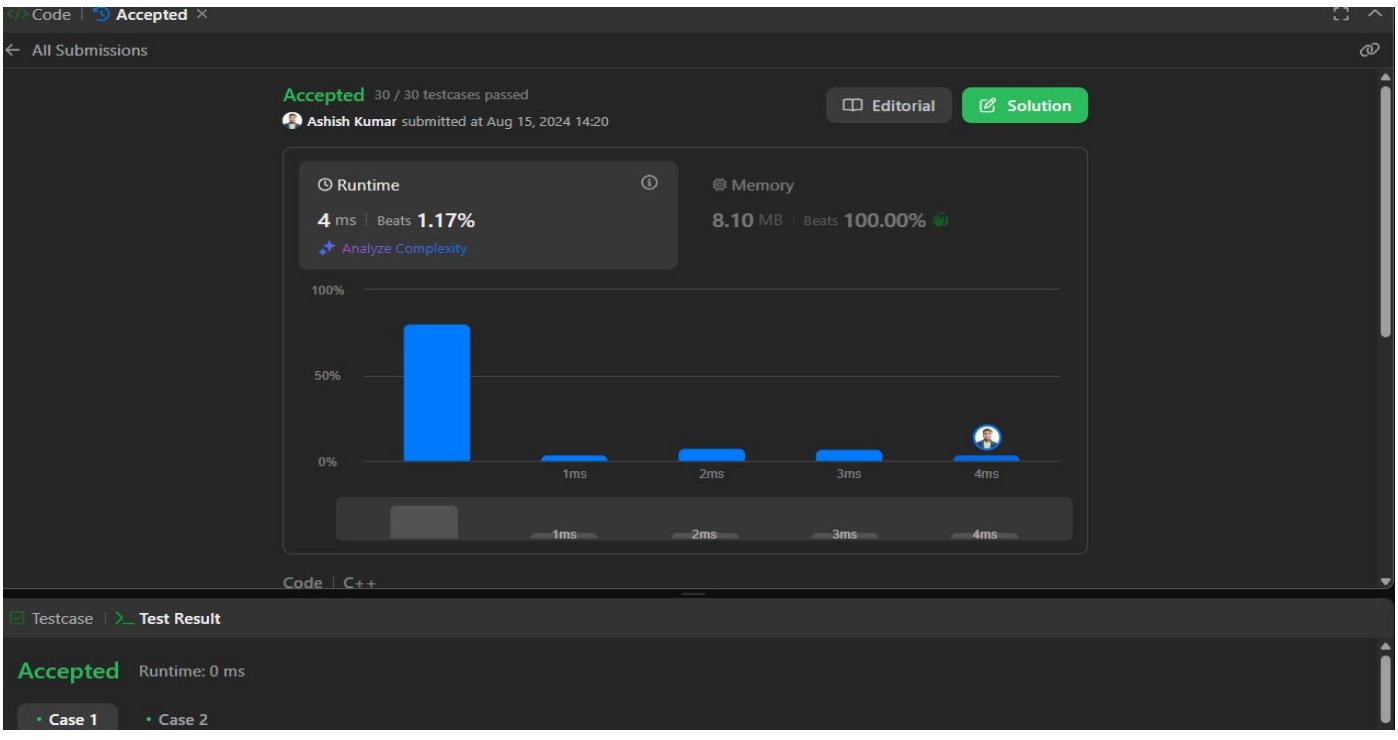
Experiment 10

Student Name: Ashish Kumar **UID:22bcs11958**
Branch: CSE **Section/Group:614(B)**
Semester: 6 **Date of Performance:17/04/25**
Subject Name: AP Lab **Subject Code: 22CSP-351**

Q1:-Pascal's Triangle

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



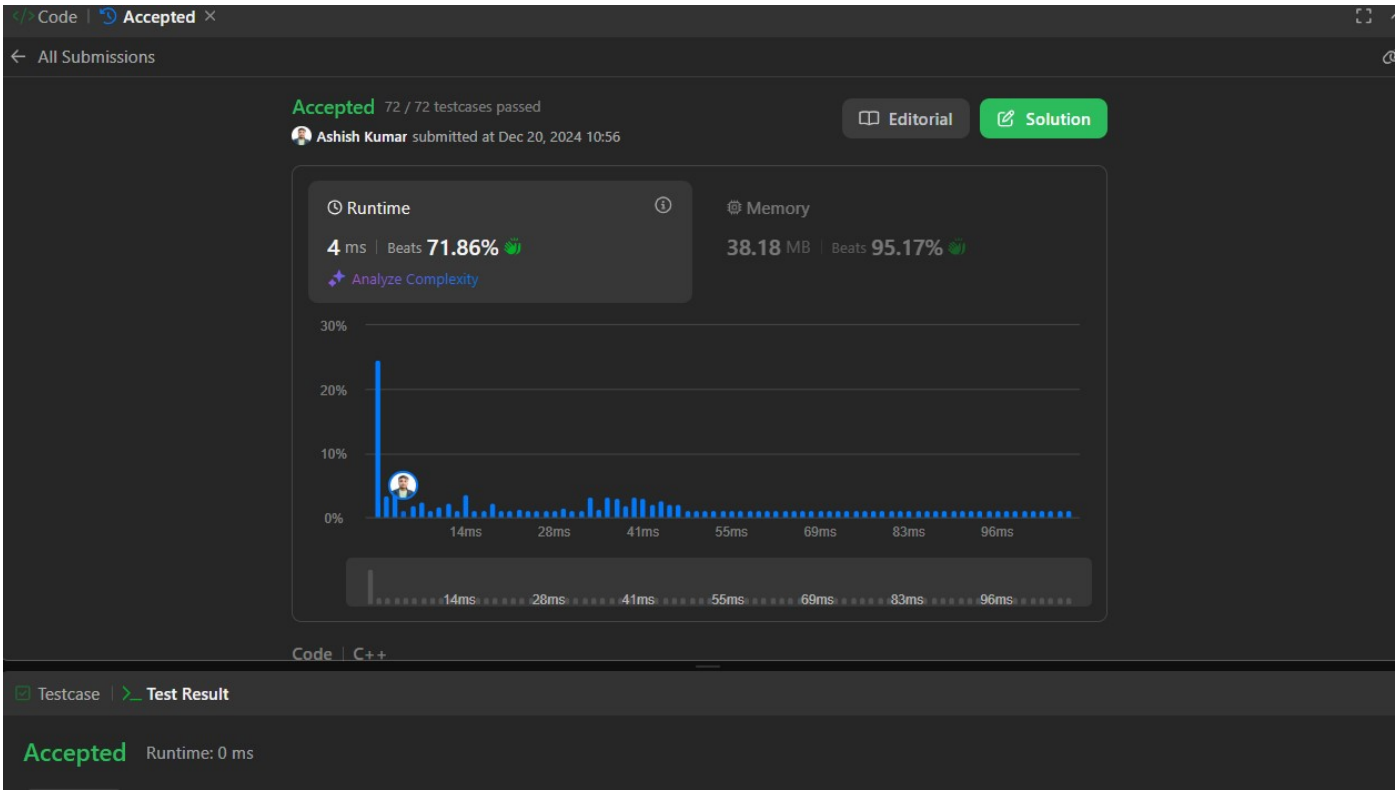
Q2:-Hamming Distance

```cpp
class Solution {
public:
    int hammingDistance(int x, int y) {
        int count=0;
        int n=x^y;
        for(int i=0;i<32;i++)
        {
            if(((n>>i)& 1)==1) count++;
        }
        return count;
    }
};
```

Q3:-Task Scheduler

```cpp
class Solution {
public:
    int leastInterval(vector<char>& tasks, int n) {
        if (n == 0) {
            return tasks.size();
        }

        vector<int> freq(26, 0); // Use int instead of char for the frequency array
```

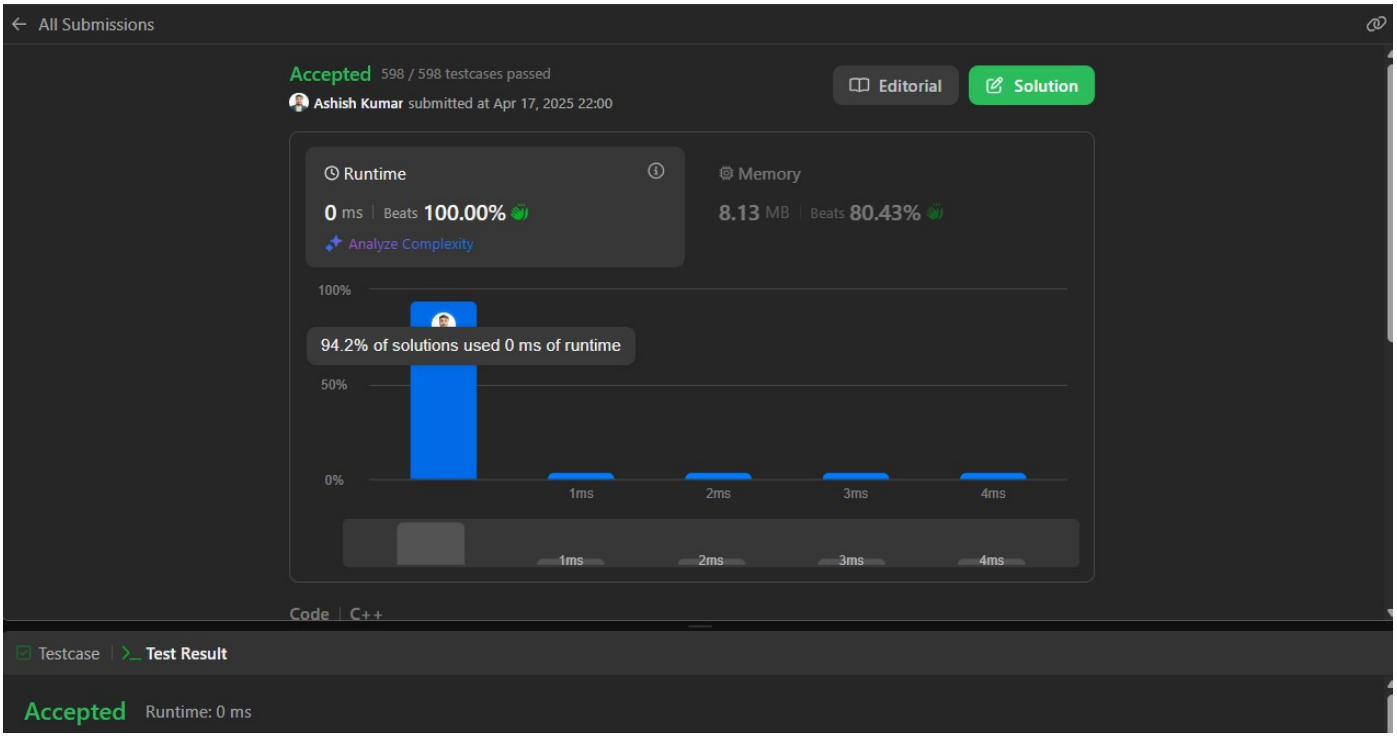
```
        for (char& ch : tasks) {
            freq[ch - 'A']++;
        }

        sort(freq.begin(), freq.end());
        int maxFreq = freq[25];
        int groupCount = maxFreq - 1;
        int idleSlots = n * groupCount;

        for (int i = 24; i >= 0; i--) {
            idleSlots -= min(freq[i], groupCount);
        }

        if (idleSlots > 0) {
            return tasks.size() + idleSlots;
        }
        return tasks.size();
    }
};
```



Q4:-Number of 1 Bits

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

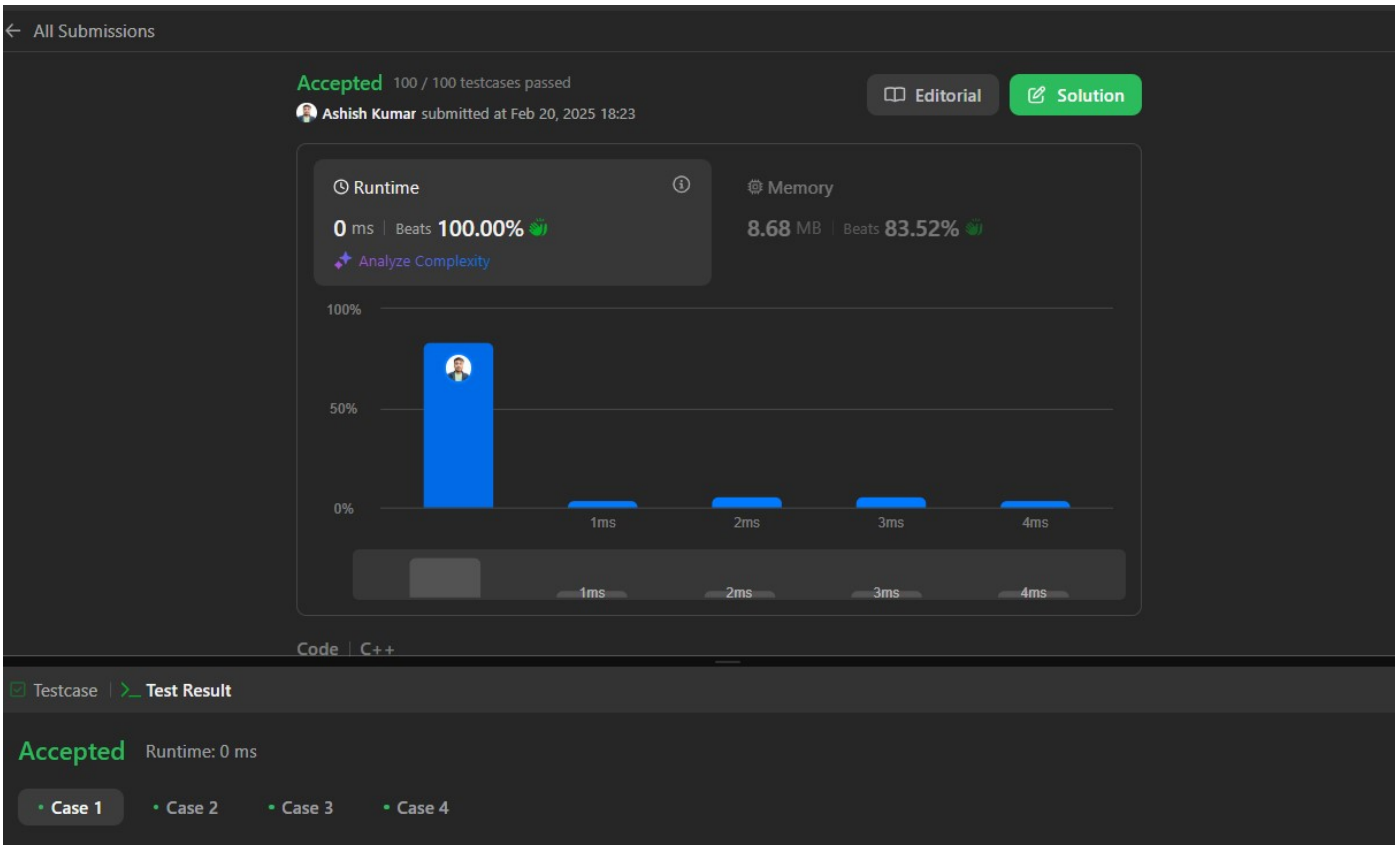
```
        int count=0;
        for(int i=0;i<32;i++)
        {
            if(((n>>i)&1)==1) count++;
        }
        return count;
    }
};
```



Q5:-Valid Parenthesis

```
class Solution {
public:
    bool isValid(string s) {
        stack<char>st;
        for(auto ch:s)
        {
            if(ch=='['|| ch=='{'|| ch=='(')
            {
                st.push(ch);
            }
            else
            {
                if(ch==')' && !st.empty() && st.top()=='(')
                {
                    st.pop();
                }
                else if(ch=='}'&& !st.empty()&& st.top()=='{')
```

```
                {
                    st.pop();
                }
                else if(ch==']' && !st.empty() && st.top()=='[')
                {
                    st.pop();
                }
                else
                {
                    return false;
                }
            }
        }
        return st.empty();
    }
};
```



Q6:-Trapping Rain Water

```
class Solution {
public:
    int trap(vector<int>& height) {
        int n=height.size();
        int sum=0;
        vector<int>leftMaxheight(n),rightMaxheight(n);
```

```
        if(n<=2) return 0;
        leftMaxheight[0]=height[0];
        for(int i=1;i<n;i++)
        {
            leftMaxheight[i]=max(leftMaxheight[i-1],height[i]);
        }
        rightMaxheight[n-1]=height[n-1];
        for(int i=n-2;i>=0;i--)
        {
            rightMaxheight[i]=max(rightMaxheight[i+1],height[i]);
        }
        for(int i=0;i<n;i++)
        {
        sum+=max(0,min(leftMaxheight[i],rightMaxheight[i])-height[i]);
        }
        return sum;
    }
};
```