Experiment 7: Divide and Conquer

Student Name: Nikhil Kumar UID: 20BCS1817
Branch: CSE Section/Group: 907/B
Semester: 6 Date of Performance: 12/04/2023
Subject Name: Competitive Coding II Subject Code: 20-CSP-351

7.1 Count and Say

1. Aim:

The count-and-say sequence is a sequence of digit strings defined by the recursive formula:

countAndSay(1) = "1" countAndSay(n) is the way you would "say" the digit string from countAndSay(n-1), which is then converted into a different digit string.
Given a positive integer n, return the nth term of the count-and-say sequence.

Input: n = 4
Output: "1211"
Explanation:
countAndSay(1) = "1" countAndSay(2) = say "1" = one 1 = "11" countAndSay(3) = say "11" = two 1's = "21"
countAndSay(4) = say "21" = one 2 + one 1 = "12" + "11" = "1211"

2. Objective:

Return nth term of count and say sequence.

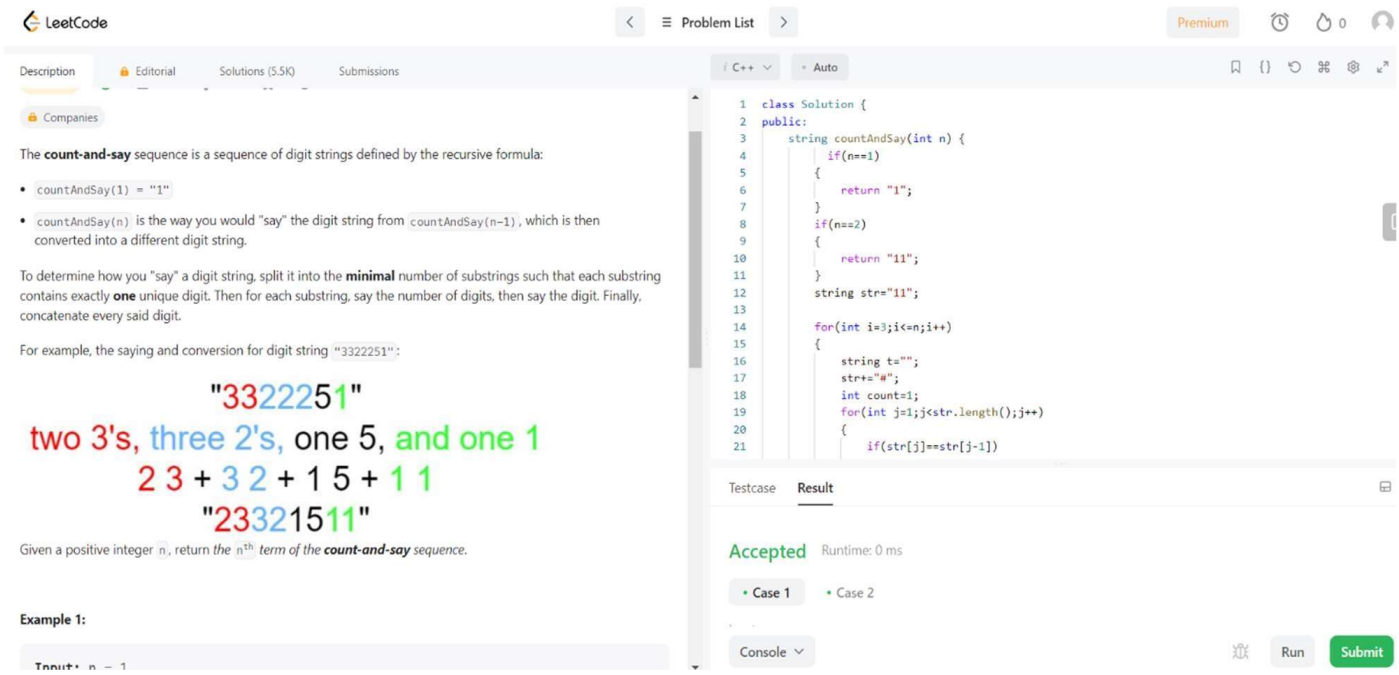
3. Script and output:

a) Code

```
class Solution { public:
   string countAndSay(int n) {
if(n==1)        {            return
"1";
       }        if(n==2)
{            return "11";
}         string str="11";
       for(int i=3;i<=n;i++)
       {            string t="";
str+="#";            int count=1;
for(int j=1;j<str.length();j++)
          {
             if(str[j]==str[j-1])              {
count++;               }               else
{                  t=t+to_string(count);
t=t+str[j-1];                  count=1;
             }
}            str=t;
}         return str;


   }
};
```

b) Output


LeetCode
Problem List
Premium
Description
Editorial
Solutions (5.5K)
Submissions
C++
Auto
Companies
The count-and-say sequence is a sequence of digit strings defined by the recursive formula:
countAndSay(1) = "1"
countAndSay(n) is the way you would "say" the digit string from countAndSay(n-1), which is then converted into a different digit string.
To determine how you "say" a digit string, split it into the minimal number of substrings such that each substring contains exactly one unique digit. Then for each substring, say the number of digits, then say the digit. Finally, concatenate every said digit.
For example, the saying and conversion for digit string "3322251":
"3322251"
two 3's, three 2's, one 5, and one 1
2 3 + 3 2 + 1 5 + 1 1
"23321511"
Given a positive integer n, return the nth term of the count-and-say sequence.
Example 1:
Input: n = 1
class Solution {
public:
string countAndSay(int n) {
if(n==1)
{
return "1";
}
if(n==2)
{
return "11";
}
string str="11";
for(int i=3;i<=n;i++)
{
string t="";
str+="#";
int count=1;
for(int j=1;j<str.length();j++)
{
if(str[j]==str[j-1])
Testcase
Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Console
Run
Submit

7.2 Water Jug Problem

1. Aim:

You are given two jugs with capacities jug1Capacity and jug2Capacity liters. There is an infinite amount of water supply available. Determine whether it is possible to measure exactly targetCapacity liters using these two jugs.

If targetCapacity liters of water are measurable, you must have targetCapacity liters of water contained within one or both buckets by the end.
Input: jug1Capacity = 2, jug2Capacity = 6, targetCapacity = 5
Output: false

2. Objective:

Find whether we can measure target capacity using jug 1 and jug 2.

3. Script and output:

c) Code

```
class Solution { public:
    bool canMeasureWater(int jug1Capacity, int jug2Capacity, int
targetCapacity) {

        int x=jug1Capacity,y=jug2Capacity,z=x+y;
        int steps[]={x,-x,-y,y}; //STEPS THAT CAN BE PERFORMED

        queue<int> q; // QUEUE TO STORE ALL POSSIBLE STATES OF
CAPACITY vector<int> vis(z+1,0); // VISITED ARRAY TO KEEP TRACK OF
VISITED NODES.
        q.push(0); vis[0]=1;
        while(q.size())
        { int node=q.front();
            q.pop();

            if(node==targetCapacity) {
                return true; // WHEN WE FIND THE TARGET CAPACITY
ACHIEVED  }
```

```
                for(int i=0;i<4;i++)
                { int newNode=node+steps[i];
                    //BOUNDARY CHECKS if(newNode>=0 &&
                    newNode<=z && vis[newNode]==0)
                    {
                        q.push(newNode); vis[newNode]=1;
                    }
                } } return false; // IF TARGET CAPACITY CAN NEVER BE
        ACHIVED }
};
```

d) Output