



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment-2.3

Student Name: Nabha Varshney

UID: 20BCS4995

Branch: CSE

Section/Group: 20BCS-DM-704 (A)

Semester: 6th

Date of Performance: 12th Apr 2023

Subject Name: Competitive Coding II

Subject Code: 20CSP- 351

Aim – To demonstrate the concept of divide and conquer.

Objective-

- ♦ The objective is to build problem solving capability and to learn the basic concepts of data structures.
- ♦ The implementation of Count and say which shows and brushes up the concept of divide and conquer and can be solved through various approaches.
- ♦ The implementation of water and jug problem in C++.

1) Count and Say

<https://leetcode.com/problems/count-and-say/description/>

Code –

```
class Solution {
public:
    string countAndSay(int n) {
        if(n == 1) {
            string str = "";
            str += '1';
            return str;
        }
        string ans = countAndSay(n-1);
        string str = "";
        for(int i=0; i<ans.size(); i++) {
            int count = 1;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
while(i != ans.size()-1 && ans[i] == ans[i+1]) {  
    count++;  
    i++;  
}  
str += (count + '0');  
str += ans[i];  
}  
return str;  
}  
};
```

Output -

38. Count and Say

Medium 3.1K 6.5K

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 *n*th term of the **count-and-say** sequence.

Example 1:

Input: `n = 1`
Output: `"1"`
Explanation: This is the base case.

Example 2:

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"`

Constraints:

- `1 <= n <= 30`

Accepted 777.8K | Submissions 1.5M | Acceptance Rate 52.1%

Seen this question in a real interview before? 1/4

```
1 class Solution {  
2 public:  
3     string countAndSay(int n) {  
4         if(n == 1) {  
5             string str = "1";  
6             return str;  
7         }  
8         string ans = countAndSay(n-1);  
9         string str = "";  
10        for(int i=0; i<ans.size(); i++) {  
11            int count = 1;  
12            while(i != ans.size()-1 && ans[i] == ans[i+1]) {  
13                count++;  
14            }
```

38. Count and Say

Medium 3.1K 6.5K

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 *n*th term of the **count-and-say** sequence.

Example 1:

Input: `n = 1`
Output: `"1"`
Explanation: This is the base case.

Example 2:

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"`

Constraints:

- `1 <= n <= 30`

Accepted 777.8K | Submissions 1.5M | Acceptance Rate 52.1%

Seen this question in a real interview before? 1/4

```
1 class Solution {  
2 public:  
3     string countAndSay(int n) {  
4         if(n == 1) {  
5             string str = "1";  
6             return str;  
7         }  
8         string ans = countAndSay(n-1);  
9         string str = "";  
10        for(int i=0; i<ans.size(); i++) {  
11            int count = 1;  
12            while(i != ans.size()-1 && ans[i] == ans[i+1]) {  
13                count++;  
14            }
```

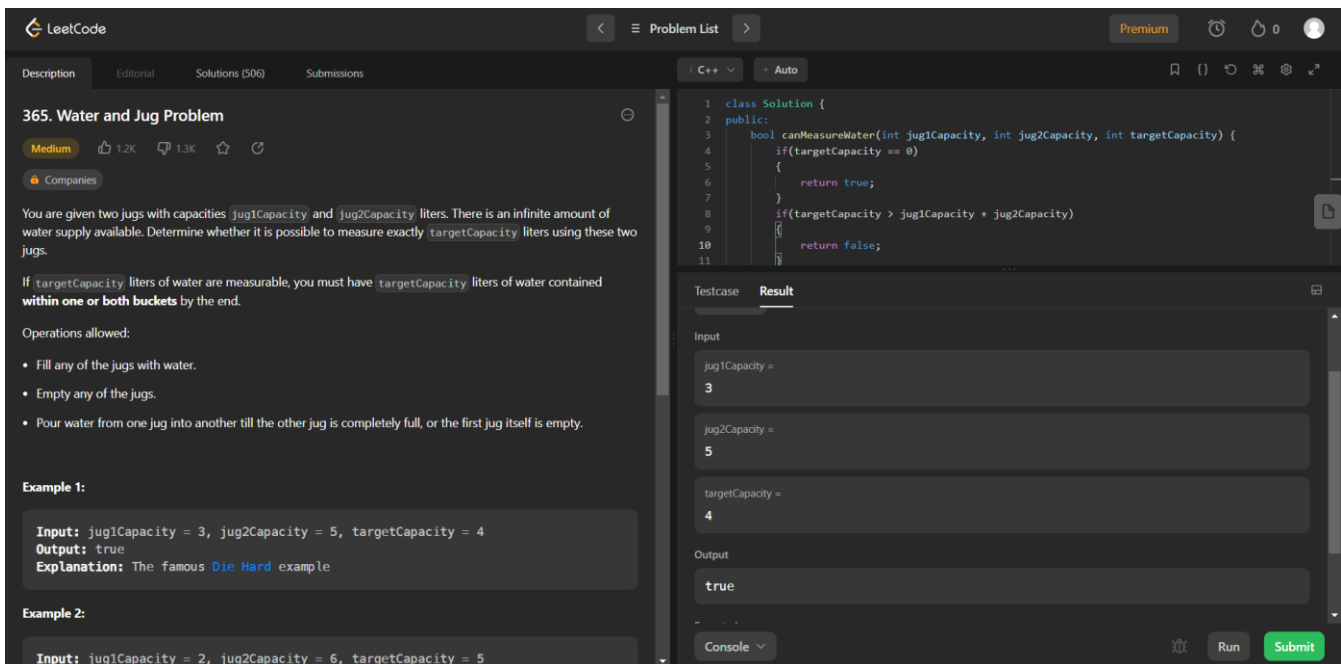
2) Water and Jug Problem

<https://leetcode.com/problems/water-and-jug-problem/>

Code -

```
class Solution {
public:
    bool canMeasureWater(int jug1Capacity, int jug2Capacity, int targetCapacity) {
        if(targetCapacity == 0)
        {
            return true;
        }
        if(targetCapacity > jug1Capacity + jug2Capacity)
        {
            return false;
        }
        else
        {
            int g = gcd(jug1Capacity, jug2Capacity);
            return targetCapacity % g == 0;
        }
    }
};
```

Output –



The screenshot shows the LeetCode interface for problem 365, "Water and Jug Problem". The problem description states: "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." It also lists allowed operations: filling, emptying, and pouring between jugs. Two examples are provided: Example 1 with input (3, 5, 4) and output true; Example 2 with input (2, 6, 5). The code editor on the right shows the C++ solution, and the test case section shows the input values (3, 5, 4) and the resulting output "true".

365. Water and Jug Problem
Medium | 1.2K | 1.3K | 0 | 0

Companies

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.

Operations allowed:

- Fill any of the jugs with water.
- Empty any of the jugs.
- Pour water from one jug into another till the other jug is completely full, or the first jug itself is empty.

Example 1:

Input: `jug1Capacity = 3, jug2Capacity = 5, targetCapacity = 4`
Output: true
Explanation: The famous Die Hard example

Example 2:

Input: `jug1Capacity = 2, jug2Capacity = 6, targetCapacity = 5`

```
1 class Solution {
2 public:
3     bool canMeasureWater(int jug1Capacity, int jug2Capacity, int targetCapacity) {
4         if(targetCapacity == 0)
5         {
6             return true;
7         }
8         if(targetCapacity > jug1Capacity + jug2Capacity)
9         {
10             return false;
11         }
12     }
13 }
```

Testcase Result

Input

`jug1Capacity =`
3

`jug2Capacity =`
5

`targetCapacity =`
4

Output

true

Console Run Submit



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

LeetCode

< Problem List >

Premium

0

Description

Editorial

Solutions (506)

Submissions

• Empty any of the jugs.

• Pour water from one jug into another till the other jug is completely full, or the first jug itself is empty.

Example 1:

Input: jug1Capacity = 3, jug2Capacity = 5, targetCapacity = 4

Output: true

Explanation: The famous [Die Hard](#) example

Example 2:

Input: jug1Capacity = 2, jug2Capacity = 6, targetCapacity = 5

Output: false

Example 3:

Input: jug1Capacity = 1, jug2Capacity = 2, targetCapacity = 3

Output: true

Constraints:

• $1 \leq \text{jug1Capacity}, \text{jug2Capacity}, \text{targetCapacity} \leq 10^6$

C++

Auto

```
1 class Solution {
2 public:
3     bool canMeasureWater(int jug1Capacity, int jug2Capacity, int targetCapacity) {
4         if(targetCapacity == 0)
5         {
6             return true;
7         }
8         if(targetCapacity > jug1Capacity + jug2Capacity)
9         {
10             return false;
11         }
12     }
13 }
```

Testcase

Result

jug1Capacity =

1

jug2Capacity =

2

targetCapacity =

3

Output

true

Expected

true

Console

Run

Submit