

---

**Experiment No. - 7**

**Student Name:** Vivek Kumar  
**Branch:** BE-CSE(LEET)  
**Semester:** 6<sup>th</sup>  
**Subject Name:** Competitive coding - II

**UID:** 21BCS8129  
**Section/Group:** 20BCS-ST-801/B  
**Date of Performance:** 25/04/2023  
**Subject Code:** 20CSP-351

**1. Aim/Overview of the practical:****Q.1 Water and Jug Problem.**

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

**2. Apparatus / Simulator Used:**

- Windows 7 or above
- Google Chrome

**3. Objective:**

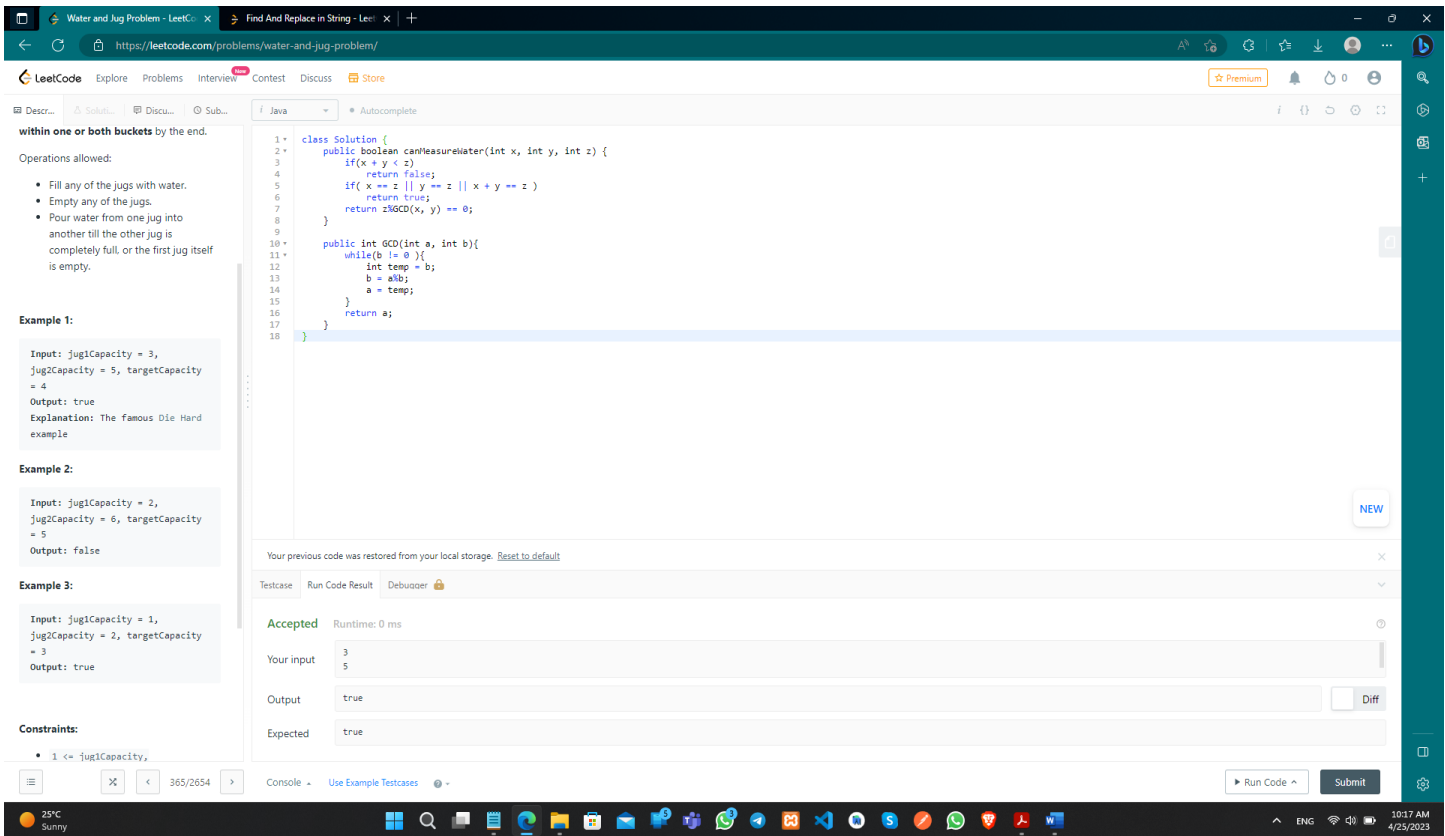
- To understand the concept of Divide and Conquer.
- To implement the concept of Water and Jug Problem.

**4. Code:**

```
class Solution {
    public boolean canMeasureWater(int x, int y, int z) {
        if(x + y < z)
            return false;
        if( x == z || y == z || x + y == z )
            return true;
        return z%GCD(x, y) == 0;
    }

    public int GCD(int a, int b){
        while(b != 0 ){
            int temp = b;
            b = a%b;
            a = temp;
        }
        return a;
    }
}
```

## 5. Result/Output/Writing Summary:



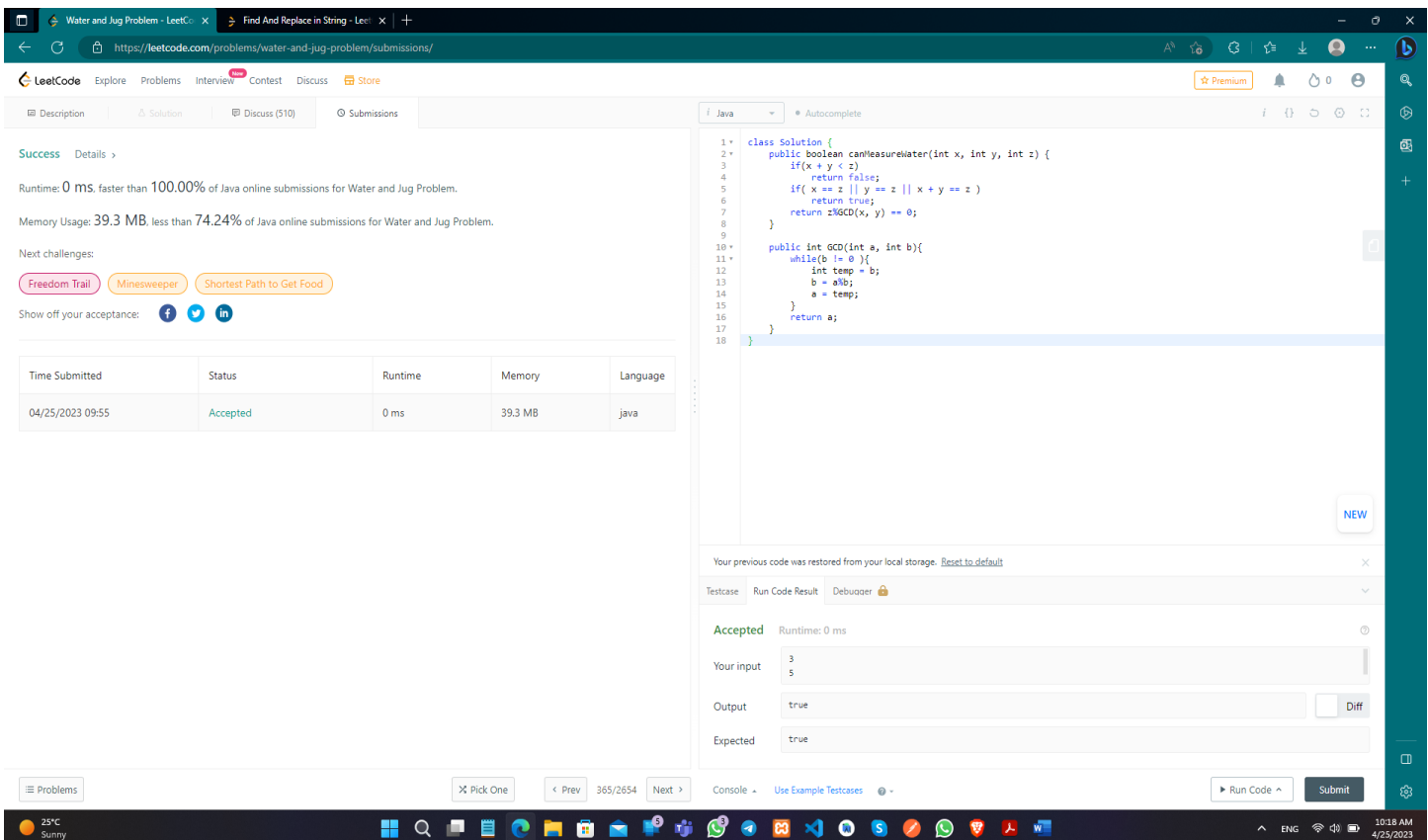
The screenshot shows the LeetCode interface for the 'Water and Jug Problem'. The problem description states: "within one or both buckets by the end. Operations allowed: Fill any of the jugs with water, Empty any of the jugs, Pour water from one jug into another till the other jug is completely full, or the first jug itself is empty." Example 1: Input: jug1Capacity = 3, jug2Capacity = 5, targetCapacity = 4. Output: true. Example 2: Input: jug1Capacity = 2, jug2Capacity = 6, targetCapacity = 5. Output: false. Example 3: Input: jug1Capacity = 1, jug2Capacity = 2, targetCapacity = 3. Output: true. Constraints: 1 <= jug1Capacity, jug2Capacity, targetCapacity <= 100.

```

1 class Solution {
2     public boolean canMeasureWater(int x, int y, int z) {
3         if(x + y < z)
4             return false;
5         if(x == z || y == z || x + y == z)
6             return true;
7         return z % GCD(x, y) == 0;
8     }
9
10    public int GCD(int a, int b){
11        while(b != 0){
12            int temp = b;
13            b = a % b;
14            a = temp;
15        }
16        return a;
17    }
18 }

```

Testcase: Run Code Result: Debbuger. Accepted. Runtime: 0 ms. Your input: 3, 5. Output: true. Expected: true.



The screenshot shows the LeetCode submission page for the 'Water and Jug Problem'. It displays the success message: "Success Details". Runtime: 0 ms, faster than 100.00% of Java online submissions for Water and Jug Problem. Memory Usage: 39.3 MB, less than 74.24% of Java online submissions for Water and Jug Problem. Next challenges: Freedom Trail, Minesweeper, Shortest Path to Get Food. Show off your acceptance: Facebook, Twitter, LinkedIn.

Time Submitted	Status	Runtime	Memory	Language
04/25/2023 09:55	Accepted	0 ms	39.3 MB	java

The solution code is the same as in the previous screenshot.

**1. Aim/Overview of the practical:****Q.2 Find and Replacement in String**<https://leetcode.com/problems/find-and-replace-in-string/>**2. Apparatus / Simulator Used:**

- Windows 7 or above
- Google Chrome

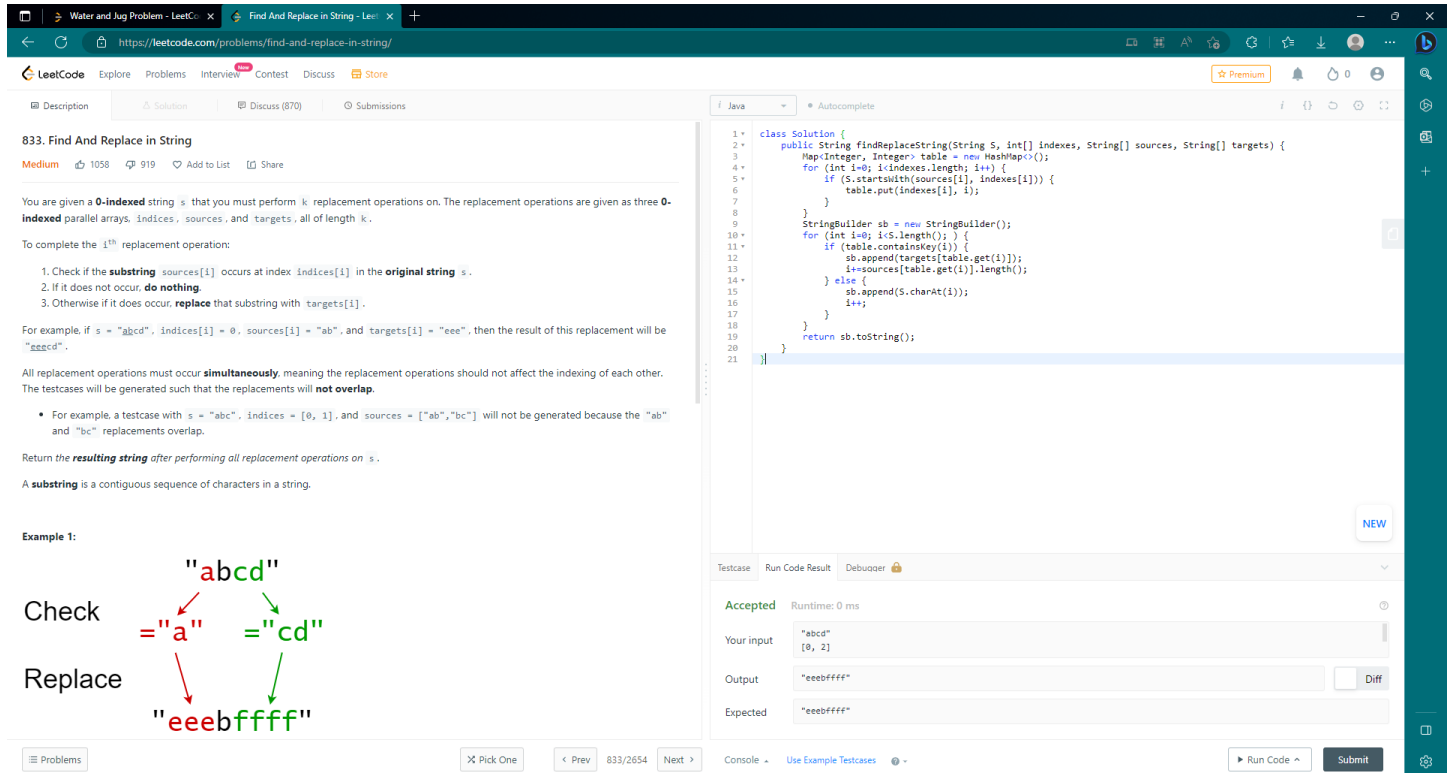
**3. Objective:**

- To understand the concept of String.
- To implement the concept of Divide and Conquer.

**4. Code:**

```
class Solution {
    public String findReplaceString(String S, int[] indexes, String[] sources, String[] targets) {
        Map<Integer, Integer> table = new HashMap<>();
        for (int i=0; i<indexes.length; i++) {
            if (S.startsWith(sources[i], indexes[i])) {
                table.put(indexes[i], i);
            }
        }
        StringBuilder sb = new StringBuilder();
        for (int i=0; i<S.length(); ) {
            if (table.containsKey(i)) {
                sb.append(targets[table.get(i)]);
                i+=sources[table.get(i)].length();
            } else {
                sb.append(S.charAt(i));
                i++;
            }
        }
        return sb.toString();
    }
}
```

## 5. Result/Output/Writing Summary:



**833. Find And Replace in String**  
Medium 1058 919 Add to List Share

You are given a 0-indexed string *s* that you must perform *k* replacement operations on. The replacement operations are given as three 0-indexed parallel arrays, *indices*, *sources*, and *targets*, all of length *k*.

To complete the *i*<sup>th</sup> replacement operation:

- Check if the **substring** *sources[i]* occurs at index *indices[i]* in the **original string** *s*.
- If it does not occur, **do nothing**.
- Otherwise if it does occur, **replace** that substring with *targets[i]*.

For example, if *s* = "abcd", *indices* = [0], *sources* = ["ab"], and *targets* = ["ee"], then the result of this replacement will be "eebcd".

All replacement operations must occur **simultaneously**, meaning the replacement operations should not affect the indexing of each other. The testcases will be generated such that the replacements will **not overlap**.

- For example, a testcase with *s* = "abc", *indices* = [0, 1], and *sources* = ["ab","bc"] will not be generated because the "ab" and "bc" replacements overlap.

Return the **resulting string** after performing all replacement operations on *s*.

A **substring** is a contiguous sequence of characters in a string.

**Example 1:**

Check "abcd"  
Replace "a" "cd"  
"eeebffff"

```

1 class Solution {
2     public String findReplaceString(String S, int[] indexes, String[] sources, String[] targets) {
3         Map<Integer, Integer> table = new HashMap<>();
4         for (int i=0; i<indexes.length; i++) {
5             if (S.startsWith(sources[i], indexes[i])) {
6                 table.put(indexes[i], i);
7             }
8         }
9         StringBuilder sb = new StringBuilder();
10        for (int i=0; i<S.length(); i++) {
11            if (table.containsKey(i)) {
12                sb.append(targets[table.get(i)]);
13                i+=sources[table.get(i)].length();
14            } else {
15                sb.append(S.charAt(i));
16                i++;
17            }
18        }
19        return sb.toString();
20    }
21 }

```

Testcase Run Code Result Debuzzer

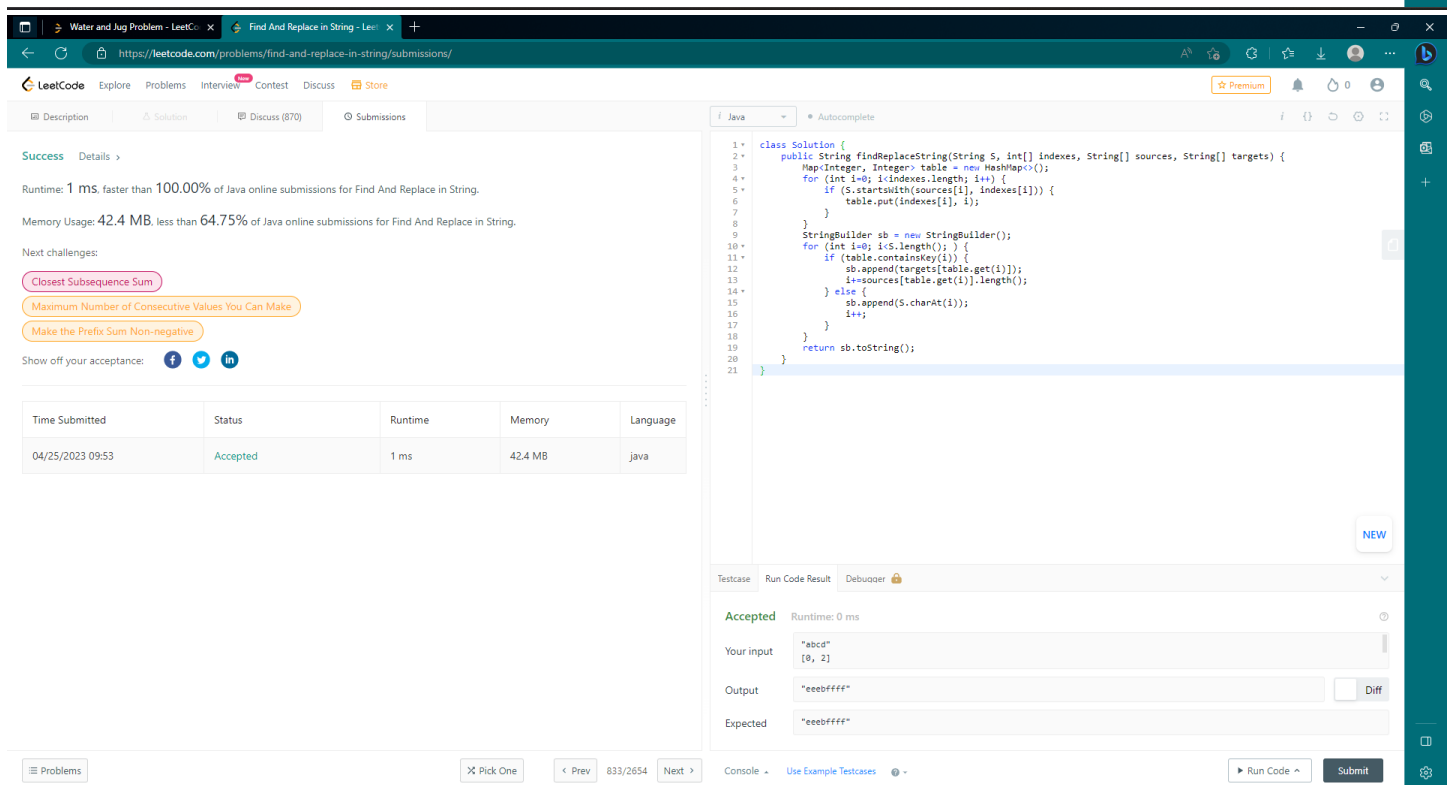
Accepted Runtime: 0 ms

Your input "abcd" [0, 2]

Output "eeebffff" Diff

Expected "eeebffff"

Console Use Example Testcases Run Code Submit






**Success** Details

Runtime: 1 ms, faster than 100.00% of Java online submissions for Find And Replace in String.

Memory Usage: 42.4 MB, less than 64.75% of Java online submissions for Find And Replace in String.

Next challenges:

- Closest Subsequence Sum
- Maximum Number of Consecutive Values You Can Make
- Make the Prefix Sum Non-negative

Show off your acceptance:   

Time Submitted	Status	Runtime	Memory	Language
04/25/2023 09:53	Accepted	1 ms	42.4 MB	java

```

1 class Solution {
2     public String findReplaceString(String S, int[] indexes, String[] sources, String[] targets) {
3         Map<Integer, Integer> table = new HashMap<>();
4         for (int i=0; i<indexes.length; i++) {
5             if (S.startsWith(sources[i], indexes[i])) {
6                 table.put(indexes[i], i);
7             }
8         }
9         StringBuilder sb = new StringBuilder();
10        for (int i=0; i<S.length(); i++) {
11            if (table.containsKey(i)) {
12                sb.append(targets[table.get(i)]);
13                i+=sources[table.get(i)].length();
14            } else {
15                sb.append(S.charAt(i));
16                i++;
17            }
18        }
19        return sb.toString();
20    }
21 }

```

Testcase Run Code Result Debuzzer

Accepted Runtime: 0 ms

Your input "abcd" [0, 2]

Output "eeebffff" Diff

Expected "eeebffff"

Console Use Example Testcases Run Code Submit

## Learning outcomes (What I have learnt):

- Learned the concept of Divide and Conquer.
- Learnt about Water and Jug Problem & Find and Replacement in String.