UNIVERSITY INSTITUTE OF ENGINEERING

Department of Computer Science & Engineering

Subject Name: Competitive Coding-II

Subject Code: 20CSP-351

Submitted to:

Mr. Arvind Gautam

Submitted by:

Name: Nabha Varshney

UID: 20BCS4995

Section: 20BCS_DM-704

Group: A

INDEX

Ex. No	List of Experiments	Conduct (MM: 12)	Viva (MM: 10)	Record (MM: 8)	Total (MM: 30)	Remarks/Signature
1.1						
1.2						
1.3						
2.1						
2.2						
2.3						
2.4						
3.1						
3.2						
3.3						

Experiment1.1

Student Name: Nabha Varshney **UID: 20BCS4995**

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

Semester: 6th **Date of Performance: 15th Feb 2023**

Subject Name: Competitive Coding II **Subject Code: 20CSP- 351**

Aim - To demonstrate the concept of string matching algorithm.

Objective-

- The objective is to build problem solving capability and to learn the basic concepts of data structures.
- The implementation of rotate string which shows and brushes up the concept of strings and can be solved through various approaches.
- The implementation of repeated string matching in which the concept of npos was introduced.

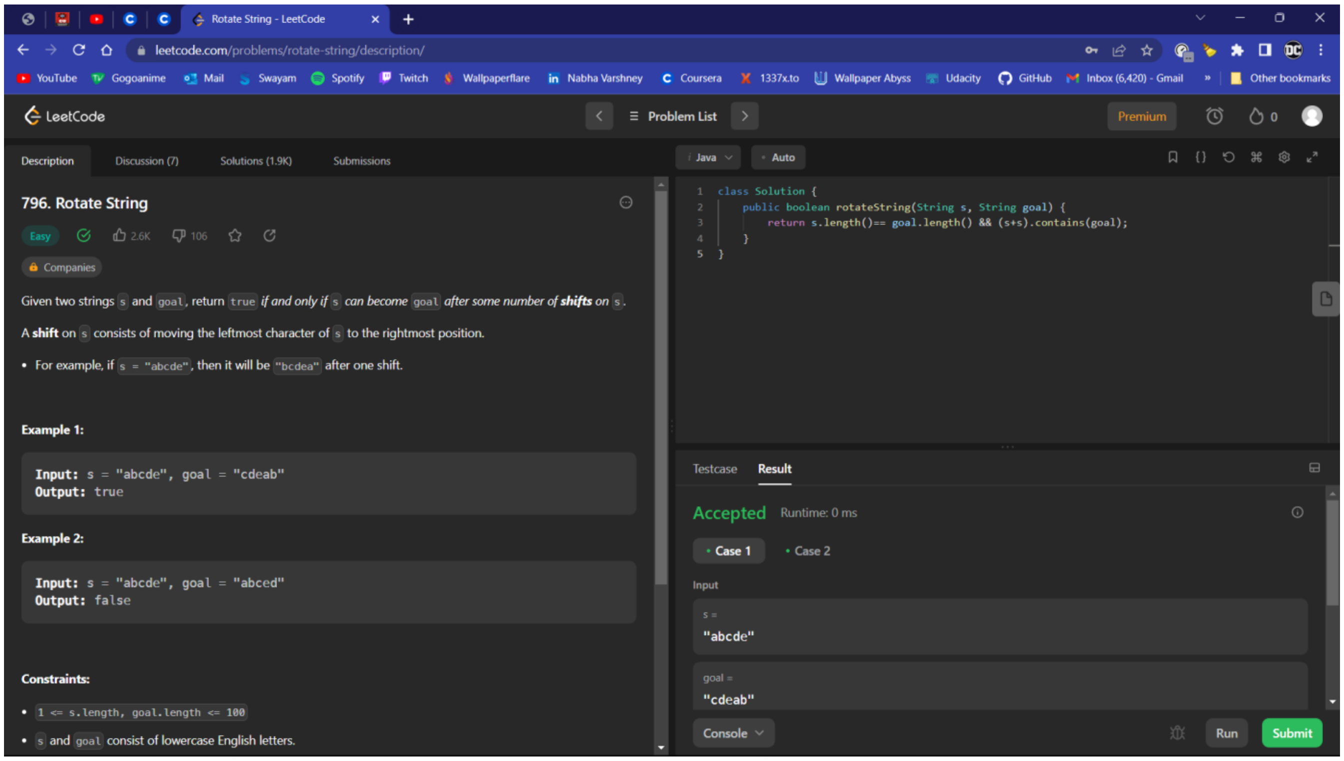
1) Rotate String

https://leetcode.com/problems/rotate-string/

Code –

```
class Solution {
    public boolean rotateString(String s, String goal) {
        return s.length()== goal.length() && (s+s).contains(goal);
    }
}
```

Output -



796. Rotate String

Easy

Given two strings `s` and `goal`, return `true` *if and only if* `s` *can become* `goal` *after some number of* ***shifts*** *on* `s`.

A **shift** on `s` consists of moving the leftmost character of `s` to the rightmost position.

- For example, if `s = "abcde"`, then it will be `"bcdea"` after one shift.

Example 1:

```
Input: s = "abcde", goal = "cdeab"
Output: true
```

Example 2:

```
Input: s = "abcde", goal = "abced"
Output: false
```

Constraints:

- `1 <= s.length, goal.length <= 100`
- `s` and `goal` consist of lowercase English letters.

```java
class Solution {
    public boolean rotateString(String s, String goal) {
        return s.length()== goal.length() && (s+s).contains(goal);
    }
}
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

```
s =
"abcde"
goal =
"cdeab"
```

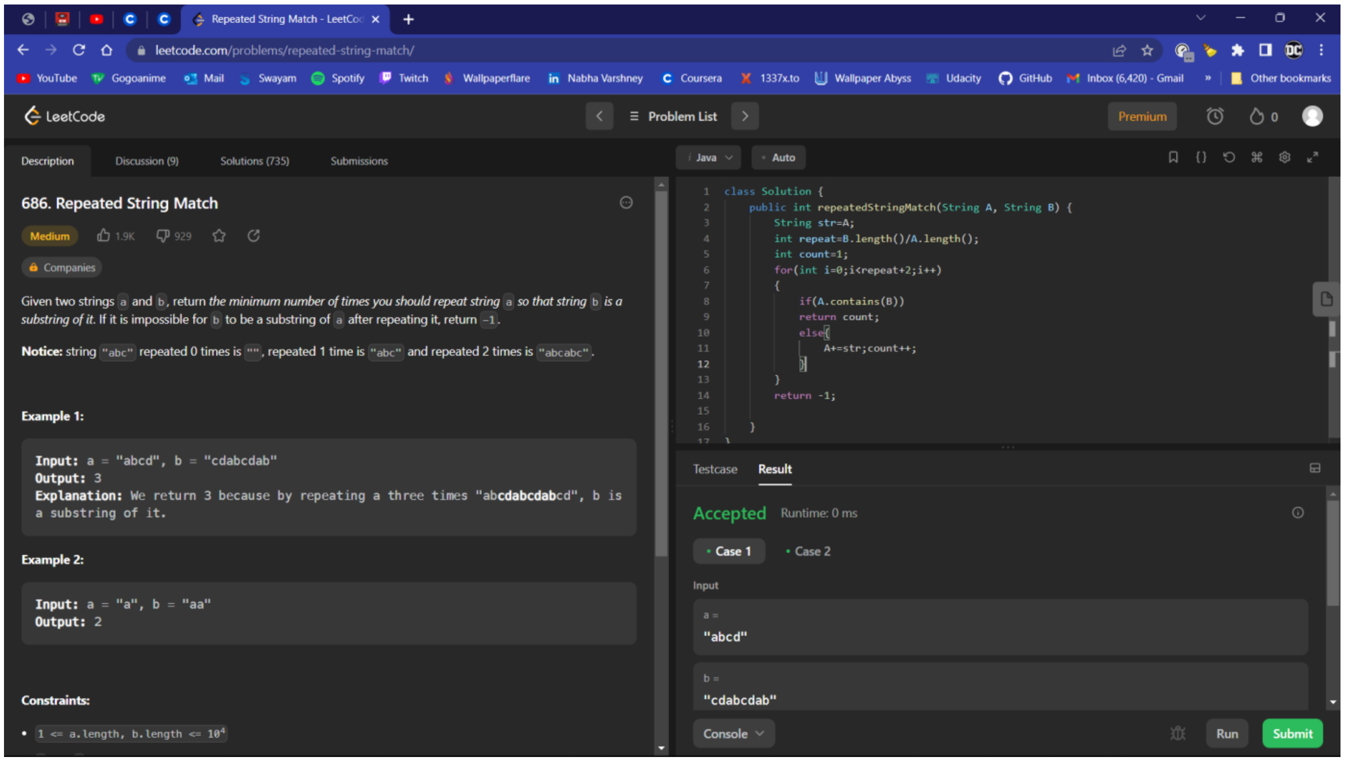
2) Repeated String

https://leetcode.com/problems/repeated-string-match/

Code -

```java
class Solution {
    public int repeatedStringMatch(String A, String B) {
        String str=A;
        int repeat=B.length()/A.length();
        int count=1;
        for(int i=0;i<repeat+2;i++)
        {
            if(A.contains(B))
            return count;
            else{
                A+=str;count++;
            }
        }
        return -1;
    }
}
```

Output -



686. Repeated String Match

Medium 1.9K 929

Given two strings `a` and `b`, return *the minimum number of times you should repeat string* `a` *so that string* `b` *is a substring of it*. If it is impossible for `b` to be a substring of `a` after repeating it, return `-1`.

Notice: string `"abc"` repeated 0 times is `""`, repeated 1 time is `"abc"` and repeated 2 times is `"abcabc"`.

Example 1:

```
Input: a = "abcd", b = "cdabcdab"
Output: 3
Explanation: We return 3 because by repeating a three times "abcdabcdabcd", b is
a substring of it.
```

Example 2:

```
Input: a = "a", b = "aa"
Output: 2
```

Constraints:

- `1 <= a.length, b.length <= 10^4`

```java
class Solution {
    public int repeatedStringMatch(String A, String B) {
        String str=A;
        int repeat=B.length()/A.length();
        int count=1;
        for(int i=0;i<repeat+2;i++)
        {
            if(A.contains(B))
            return count;
            else{
                A+=str;count++;
            }
        }
        return -1;

    }
}
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

a =
"abcd"

b =
"cdabcdab"