
LEETCODE
DAY 1 - GETTING STARTED

Problem List

1768. Merge Strings Alternately - Strings

389. Find the Difference - Strings

28. Find the Index of the First Occurrence in a String - Strings

242. Valid Anagram - Strings

13. Roman to Integer - Strings

412. Fizz Buzz - Strings

459. Repeated Substring Pattern - Strings

1502. Can Make Arithmetic Progression From Sequence - Arrays

283. Move Zeroes - Arrays

66. Plus One - Arrays

Problem 1: 1768. Merge Strings Alternately

You are given two strings word1 and word2. Merge the strings by adding letters in alternating order, starting with word1. If a string is longer than the other, append the additional letters onto the end of the merged string. Return the merged string.

Solution:

```python
class Solution:
    def mergeAlternately(self, word1: str, word2: str) -> str:
        i = 0
        j = 0
        res = ""
        while i < len(word1) and j < len(word2):
            res += word1[i]
            res += word2[j]
            i = i + 1
            j = j + 1
        res += word1[i:] + word2[j:]
        return res
```

Problem 2: 389. Find the Difference - Strings

You are given two strings s and t. String t is generated by random shuffling string s and then add one more letter at a random position. Return the letter that was added to t.

Solution:

```python
class Solution:
    def findTheDifference(self, s: str, t: str) -> str:
        ss,tt=0,0
        for i in s:
            ss+=ord(i)
        for j in t:
            tt+=ord(j)
        return chr(tt-ss)
```

Problem 3: 28. Find the Index of the First Occurrence in a String

Given two strings needle and haystack, return the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.

Solution:

```python
class Solution:
    def strStr(self, haystack: str, needle: str) -> int:
        return haystack.find(needle)
```

Problem 4: 242. Valid Anagram - Strings

Given two strings s and t, return true if t is an anagram of s, and false otherwise. An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

Solution:

```
class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        list1=[]
        list2=[]
        for i in s:
            list1.append(i)
        for j in t:
            list2.append(j)
        list1.sort()
        list2.sort()
        if(list1==list2):
            return True
        return False
```

Problem 5: 13. Roman to Integer - Strings

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX.Given a roman numeral, convert it to an integer.

Solution:

```
class Solution:
  def romanToInt(self, s: str) -> int:
    ans = 0
    roman = {'I': 1, 'V': 5, 'X': 10, 'L': 50,
             'C': 100, 'D': 500, 'M': 1000}
    for a, b in zip(s, s[1:]):
      if roman[a] < roman[b]:
        ans -= roman[a]
      else:
        ans += roman[a]
    return ans + roman[s[-1]]
```

Problem 6: 412. Fizz Buzz - Strings

Given an integer n, return a string array answer (1-indexed) where:

- answer[i] == "FizzBuzz" if i is divisible by 3 and 5.
- answer[i] == "Fizz" if i is divisible by 3.
- answer[i] == "Buzz" if i is divisible by 5.
- answer[i] == i (as a string) if none of the above conditions are true.

Solution:

```python
class Solution:
    def fizzBuzz(self, n: int) -> List[str]:
        list=[]
        for i in range(1,n+1):
            if(i%3==0 and i%5==0):
                list.append("FizzBuzz")
            elif(i%3==0):
                list.append("Fizz")
            elif(i%5==0):
                list.append("Buzz")
            else:
                list.append(str(i))
        return list
```

Problem 7: 459. Repeated Substring Pattern - Strings

Given a string s, check if it can be constructed by taking a substring of it and appending multiple copies of the substring together.

Solution:

```python
class Solution:
    def repeatedSubstringPattern(self, s: str) -> bool:
        return s in (s+s)[1:-1]
```

Problem 8 : 1502. Can Make Arithmetic Progression From Sequence - Arrays

A sequence of numbers is called an arithmetic progression if the difference between any two consecutive elements is the same. Given an array of numbers arr, return true if the array can be rearranged to form an arithmetic progression. Otherwise, return false.

Solution:

```
class Solution:
    def canMakeArithmeticProgression(self, arr: List[int]) -> bool:
        arr.sort()
        comm=arr[1]-arr[0]
        for i in range(1,len(arr)-1):
            if(arr[i+1]-arr[i]!=comm):
                return False
        return True
```

Problem 9: 283. Move Zeroes - Arrays

Given an integer array nums, move all 0's to the end of it while maintaining the relative order of the non-zero elements. **Note** that you must do this in-place without making a copy of the array.

Solution:

```python
class Solution:
    def moveZeroes(self, nums: List[int]) -> None:
        listt=[]
        count=0
        for i in nums:
            if i==0:
                count=count+1
                continue
            else:
                listt.append(i)
        for j in range(count):
            listt.append(0)
        nums.clear()
        for p in listt:
            nums.append(p)
```

Problem 10: 66. Plus One

You are given a **large integer** represented as an integer array digits, where each digits[i] is the i^{th} digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's.Increment the large integer by one and return *the resulting array of digits*.

Solution:

```
class Solution:
    def plusOne(self, digits: List[int]) -> List[int]:
        res=""
        for i in digits:
            a=str(i)
            res+=a
        b=int(res)
        b=b+1
        c=str(b)
        listt=[]
        for i in c:
            l=int(i)
            listt.append(l)
```