Leetcode Problems

.10. Score of a String

```
class Solution:
    def scoreOfString(self, s: str) -> int:
        score = 0
        for i in range(len(s)-1):
            score += abs(ord(s[i]) - ord(s[i+1]))

        return score
```

.08. Defanging an IP Address

```
class Solution:
    def defangIPaddr(self, address: str) -> str:
        a=""
        for i in address:
            if i==".":
                a=a+"[.]"
            else:
                    a=a+i
        return a
```

)11. Final Value of Variable After Performaing Operations

```
class Solution:
    def finalValueAfterOperations(self, operations: List[str]) -
        x=0
        for i in operations:
            if i[0]=="+"or i[-1]=="+":
                x+=1
```

```
            else:
                x-=1
        return x
```

771. Jewels and Stones

```
class Solution:
    def numJewelsInStones(self, jewels: str, stones: str) -> int
        sum=0
        for i in jewels:
            for j in stones:
                if i==j:
                    sum=sum+1
        return sum
```

2042. Find Words Containing Character

```
class Solution:
    def findWordsContaining(self, words: List[str], x: str) -> L
        c=[]
        for i in range(0,len(words)):
            if x in words[i]:
                c.append(i)
        return c
```

1678. Goal parser Interpretation

```
class Solution:
    def interpret(self, command: str) -> str:
        a=""
        for i in range(len(command)):
            if command[i]=="G":
                a=a+"G"
            elif command[i]=="(" and command[i+1] ==")":
```

```
                a=a+"o"
            elif command[i]=="(" and command[i+1] =="a":
                a=a+"al"
        return a
```

1114. Maximum Number of Words Found in sentence

```
class Solution:
    def mostWordsFound(self, sentences: List[str]) -> int:
        n=[]
        for i in sentences:
            l=i.split(" ")
            n.append(len(l))
        return max(n)
```

1221. Split a String in Balanced String

```
class Solution:
    def balancedStringSplit(self, s: str) -> int:
        rno=0
        lno=0
        c=0
        for i in s:
            if i=="R":
                rno+=1
            else:
                lno+=1
            if rno==lno:
                c+=1
                lno=0
                rno=0
        return c
```

1662. Check If Two String Arrays are Equivalent

```
class Solution:
    def arrayStringsAreEqual(self, word1: List[str], word2: List
        if "".join(word1)=="".join(word2):
            return True
        else:
            return False
```

1773. Count Items Matching a Rule

```
class Solution:
    def countMatches(self, items: List[List[str]], ruleKey: str,
        c=0
        if ruleKey=="type":
            for i in items:
                if i[0]==ruleValue:
                    c+=1
        elif ruleKey=="color":
                    for i in items:
                        if i[1]==ruleValue:
                            c+=1
        elif ruleKey=="name":
                    for i in items:
                        if i[2]==ruleValue:
                            c+=1
        return c
```

1816. Truncate Sentence

```
class Solution:
    def truncateSentence(self, s: str, k: int) -> str:
        a=""
        l=s.split()
        for i in range(k):
            if i==k-1:
```

```
                a=a+l[i]
                return a
            a=a+l[i]+" "
```

528. Shuffle String

```
class Solution:
    def restoreString(self, s: str, indices: List[int]) -> str:
        l=[0]*len(s)
        for i in range(len(s)):
            l[indices[i]]=s[i]
        return "".join(l)
```

525. Decode the Message

```
class Solution:
    def decodeMessage(self, key: str, message: str) -> str:
        l1=""
        l2="abcdefghijklmnopqrstuvwxyz"
        for i in key:
            if i not in l1 and i!=" ":
               l1=l1+i
               ans=""
        for i in message:
            if i!=" ":
               a=l1.index(i)
               ans=ans+l2[a]
            else:
               ans=ans+" "
        return ans
```

2108. Find First Palindromic String in the Array

```
class Solution:
    def firstPalindrome(self, words: List[str]) -> str:
        for i in words:
            if i==i[::-1]:
                return i
        else:
            return ""
```

94. Cells in a Range on an Excel Sheet

```
class Solution:
    def cellsInRange(self, s: str) -> List[str]:
        i=s
        a=ord(i[0])
        b=ord(i[3])
        c=int(i[1])
        d=int(i[4])
        l=[]
        for i in range(a,b+1):
            for j in range(c,d+1):
                s=chr(i)+str(j)
                l.append(s)
        return l
```

514. Maximum Nesting Depth of the Parentheses

```
class Solution:
    def maxDepth(self, s: str) -> int:
        c=0
        max=0
        for i in s:
            if i=="(":
               c+=1
            if max<c:
```

```
                max=c
            if i==")":
                c-=1
        return max
```

709. To Lower Case

```
class Solution:
    def toLowerCase(self, s: str) -> str:
        for i in range(len(s)):
            if 65 <= ord(s[i]) <= 90:
                s = s[:i] + chr(ord(s[i]) + 32) + s[i+1:]
        return s
```

810. Faulty Keyboard

```
class Solution:
    def finalString(self, s: str) -> str:
        result = []
        reverse = False
        for char in s:
            if char == 'i':
                reverse = not reverse
            else:
                if reverse:
                    result.insert(0, char)
                else:
                    result.append(char)
        if reverse:
            result.reverse()
        return ''.join(result)
```

1832. Check if the Sentence Is Pangram

```
class Solution:
    def checkIfPangram(self, sentence: str) -> bool:
        s = sorted(list(set(sentence)))
        alpha = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u',
'v', 'w', 'x', 'y', 'z']
        if alpha == s:
            return True
        return False
```

557. Reserve Words in a String III

```
class Solution:
    def reverseWords(self, s: str) -> str:
        a=""
        l=s.split(" ")
        for i in range(len(l)):
            if i==len(l)-1:
                a=a+l[i][::-1]
            else:
                a=a+l[i][::-1]+" "
        return a
```

1684. Count the Number of Consistent Strings

```
class Solution:
    def countConsistentStrings(self, allowed: str, words: List[s
            allowed = set(allowed)
            count = 0
            for word in words:
                for letter in word:
                    if letter not in allowed:
                       count += 1
```

```
                        break
            return len(words) - count
```

359. Sorting the Sentence

```
class Solution:
    def sortSentence(self, s: str) -> str:
        l = s.split()
        d = {}
        for i in l:
            d[i[-1]]=i[:-1]
        s=""
        for i in range(1,len(l)+1):
            if i==len(l):
                s=s+d[str(i)]
            else:
                s=s+d[str(i)]+" "
        return s
```

328. Check if a String Is an Acronym of Words

```
class Solution:
    def isAcronym(self, words: List[str], s: str) -> bool:
        if len(words) != len(s):
            return False
        else:
            for i in range(len(s)):
                if s[i] != words[i][0]:
                    return False
        return True
```

804. Unique Morse Code Words

```
class Solution:
    def uniqueMorseRepresentations(self, words: List[str]) -> in
        mor=[".-","-...","-.-.","-..",".","..-.","--.","....","
        l=[]
        for j in words:
            ans=""
            for i in j:
                ans=ans+mor[ord(i)-97]
            l.append(ans)
        q=set(l)
        return len(q)
```

1021. Remove Outermost Parenthesis

```
class Solution:
    def removeOuterParentheses(self, s: str) -> str:
        out  = ''
        count = -1
        f = True
        for i in s:
            if f:
                f = False
                count +=1
                continue
            if i=='(':
                out+=i
                count +=1
            if i ==')':
                count -=1
                if count == -1:
                    f = not f
                    continue
```

```
                out +=i
        return out
```

315. Count Asterisks

```
class Solution:
    def countAsterisks(self, s: str) -> int:
        ans,t = 0,0
        for i in s:
            if i == "|":
                t += 1
            elif t % 2 ==0:
                ans += i=="*"

        return ans
```

344. Replace All Digits With Characters

```
class Solution:
    def replaceDigits(self, s: str) -> str:
        l=list(s)
        for i,j in enumerate(l):
            if j.isdigit():
                l[i]=chr(ord(l[i-1])+int(j))
        return ''.join(l)
```

2185. Counting Words With a Given Prefix

```
class Solution:
    def prefixCount(self, words: List[str], pref: str) -> int:
        ans = 0
        for i in words:
            if i[:len(pref)] == pref:
                ans += 1
```

```
        return ans
```

344. Reserve String

```
class Solution:
    def reverseString(self, s: List[str]) -> None:
        s[:]=s[::-1]
```

2. Add Two Numbers

```
class Solution:
    def addTwoNumbers(self, l1: Optional[ListNode], l2: Optional
        s=""
        s2=""
        while l1!=None:
            s=s+str(l1.val)
            l1=l1.next
        while l2!=None:
            s2=s2+str(l2.val)
            l2=l2.next
        s=s[::-1]
        s2=s2[::-1]
        a=int(s)+int(s2)
        a=str(a)
        a=a[::-1]
        q=str(a)
        z=ListNode()
        curr=z
        for i in q:
            new=ListNode(int(i))
            curr.next=new
            curr=curr.next
        return z.next
```

744. Find Maximum number of String pairs

```
class Solution:
    def maximumNumberOfStringPairs(self, words: List[str]) -> in
        ans = 0
        for i in range(len(words)):
            for j in range(i+1, len(words)):
                if words[i][0] == words[j][1] and words[i][1] ==
                    ans += 1
        return ans
```

1470. Shuffle the array

```
 i = 0
        j = n
        ans = []

        while i <= n-1:
            ans.append(nums[i])
            ans.append(nums[j])
            i += 1
            j += 1

        return ans
```

1774. Minimum Number Game

```
nums.sort()
        i=0
        while(i<len(nums)):
            nums[i],nums[i+1]=nums[i+1],nums[i]
            i+=2
        return nums
```

1356. Design an ordered stream

```
class OrderedStream:

    def __init__(self, n: int):
        self.ptr = 1
        self.hashmap = dict()


    def insert(self, idKey: int, value: str) -> List[str]:
        self.hashmap[idKey] = value
        output = []
        if idKey > self.ptr:
            return output

        while idKey in self.hashmap:
            output.append(self.hashmap[idKey])
            idKey += 1
            self.ptr = idKey

        return output



# Your OrderedStream object will be instantiated and called as s
# obj = OrderedStream(n)
# param_1 = obj.insert(idKey,value)
```

572. Richest Customer Wealth

```
class Solution:
    def maximumWealth(self, accounts: List[List[int]]) -> int:
        larg=0
        for i in range(len(accounts)):
            summ=0
            for j in range(len(accounts[i])):
                summ+=accounts[i][j]
```

```
                if summ>larg:
                    larg=summ
        return larg
```

341. Maximum Number of Pairs in Array

```
class Solution:
    def numberOfPairs(self, nums: List[int]) -> List[int]:
        pair = 0
        count = 0
        nums_set = set(nums)
        for i in nums_set:
            temp = nums.count(i)
            pair += temp // 2
            count += temp % 2
        return [pair, count]
```

136. Single Number

```
class Solution:
    def singleNumber(self, nums: List[int]) -> int:
        hash = {}

        for i in nums:
            hash[i] = hash.get(i,0)+1

        for j in hash:
            if hash[j] == 1:
                return j
```

231. Power of Two

```
class Solution:
    def isPowerOfTwo(self, n: int) -> bool:
        for i in range(31):
            ans = 2 ** i
            if ans == n:
                return True
        return False
```

202. Happy Number

```
class Solution:
    def isHappy(self, n: int) -> bool:
        if n == 1 or n == 7:
            return True
        if n < 10:
            return False
        lib = set()
        while n != 1:
            n = sum(int(i)**2 for i in str(n))
            if n in lib:
                return False
            lib.add(n)
        return True
```

268. Missing Number

```
class Solution:
    def missingNumber(self, nums: List[int]) -> int:
        n = len(nums)
        v = [-1] * (n + 1)
        for num in nums:
            v[num] = num
        for i in range(len(v)):
            if v[i] == -1:
```

```
                return i
        return 0
```

58. Length of Last Word

```
class Solution:
    def lengthOfLastWord(self, s: str) -> int:
        words = s.strip().split()

        if not words:
            return 0

        return len(words[-1])
```

66. Plus One

```
class Solution:
    def plusOne(self, digits: List[int]) -> List[int]:
        for i in reversed(range(len(digits))):
            if digits[i] != 9:
                digits[i] += 1
                return digits
            digits[i] = 0

        return [1] + digits
```

35.Search Insert Position

```
class Solution:
    def searchInsert(self, nums: List[int], target: int) -> int
        left, right = 0, len(nums) - 1

        while left <= right:
            mid = left + (right - left) // 2
```

```
            if nums[mid] == target:
                return mid
            elif nums[mid] < target:
                left = mid + 1
            else:
                right = mid - 1

        return left
```

69. Sqrt(x)

```
class Solution:
    def mySqrt(self, x: int) -> int:
        b = x

        while b*b > x: b=(b+x//b)//2

        return b
```

25. Valid Palindrome

```
class Solution:
    def isPalindrome(self, s: str) -> bool:
        l = 0
        r = len(s) - 1
        while l < r:
            if not s[l].isalnum():
                l += 1
            elif not s[r].isalnum():
                r -= 1
            elif s[l].lower() == s[r].lower():
                l += 1
                r -= 1
            else:
```

```
                return False

        return True
```

190. Reverse Bits

```
class Solution:
    def reverseBits(self, n: int) -> int:
        def f(n,r,count):
            if n<1:
                return r<<(32-count)
            return f(n>>1,(r<<1)|(n&1),count+1)
        return f(n,0,0)
```

121. Best Time to Buy and Sell Stock

```
class Solution:
    def maxProfit(self, prices: List[int]) -> int:
        min_price = prices[0]
        max_profit = 0

        for price in prices[1:]:
            max_profit = max(max_profit, price - min_price)
            min_price = min(min_price, price)

        return max_profit
```

1941. Check if All Characters Have Equal Number of Occurrences

```
class Solution:
    def areOccurrencesEqual(self, s: str) -> bool:
        d = defaultdict(int)
        for i in s: d[i] += 1
        c = d[s[0]]
        for i in d:
```

```
            if d[i] != c: return 0
        return 1
```

570. Increasing Decreasing String

```
class Solution:
    def sortString(self, s: str) -> str:
        s = list(s)
        result = ''
        while s:
            for letter in sorted(set(s)):
                s.remove(letter)
                result += letter
            for letter in sorted(set(s), reverse=True):
                s.remove(letter)
                result += letter
        return result
```

716. Minimize String Length

```
class Solution:
    def minimizedStringLength(self, s: str) -> int:
        res = []
        for i in s:
            res.append(i)
        news=set(res)
        return len(news)
```

944. Delete Columns To Make Sorted

```
class Solution:
    def minDeletionSize(self, strs: List[str]) -> int:
        c = []
        for i in range(len(strs)-1):
            for j in range(len(strs[0])):
```

```
                if strs[i][j] > strs[i+1][j]:
                    c.append(j)
        return len(set(c)
```

2778. Percentage of Letter in String

```
class Solution:
    def percentageLetter(self, s: str, letter: str) -> int:
        l=len(s)
        count=0
        for i in s:
            if i==letter:
                count+=1
            else:
                pass
        return int((count/l)*100)
```

2351. First Letter to Appear Twice

```
class Solution:
    def repeatedCharacter(self, s: str) -> str:
        setS = set()

        for x in s:
            if x in setS:
                return x
            else:
                setS.add(x)
```

2255. Count Prefixes of a Given String

```
class Solution:
    def countPrefixes(self, words: List[str], s: str) -> int:
        count=0
```

```
        for word in words:
            n=len(word)
            if s[0:n]==word:
                count+=1
        return count
```

500. keyboard Row

```
class Solution:
    def findWords(self, words: List[str]) -> List[str]:
        l1="qwertyuiop"
        l2="asdfghjkl"
        l3="zxcvbnm"
        res=[]
        for word in words:
            w=word.lower()
            if len(set(l1+w))==len(l1) or len(set(l2+w))==len(l2
                res.append(word)
        return res
```

13.Roman to Integer

```
class Solution(object):
    def romanToInt(self, s):
        sum=0
        i=0
        while i<len(s):
            if s[i]=="I":
                if i<=len(s)-2 and s[i+1]=="V":
                    sum=sum+4
                    i=i+2
                elif i<=len(s)-2 and s[i+1]=="X":
                    sum=sum+9
```

```
                i=i+2
            else:
                sum=sum+1
                i=i+1
        elif s[i]=="V":
            sum=sum+5
            i=i+1
        elif s[i]=="X":
            if i<=len(s)-2 and s[i+1]=="L":
                sum=sum+40
                i=i+2
            elif i<=len(s)-2 and s[i+1]=="C":
                sum=sum+90
                i=i+2
            else:
                sum=sum+10
                i=i+1
        elif s[i]=="L":
            sum=sum+50
            i=i+1
        elif s[i]=="C":
            if i<=len(s)-2 and s[i+1]=="D":
                sum=sum+400
                i=i+2
            elif i<=len(s)-2 and s[i+1]=="M":
                sum=sum+900
                i=i+2
            else:
                sum=sum+100
                i=i+1
        elif s[i]=="D":
            sum=sum+500
            i=i+1
        elif s[i]=="M":
            sum=sum+1000
```

```
                i=i+1
        return sum
```

599. Minimum Index Sum of Two Lists

```
class Solution:
    def findRestaurant(self, list1: List[str], list2: List[str]
        sum=[]
        ans=[]
        for i in range(0,len(list1)):
            for j in range(0,len(list2)):
                if list1[i]==list2[j]:
                    sum.append(i+j)
        m=min(sum)
        for i in range(0,len(list1)):
            for j in range(0,len(list2)):
                if list1[i]==list2[j] and i+j==m:
                    ans.append(list1[i])
        return ans
```

58. Length of Last Word

```
class Solution:
    def lengthOfLastWord(self, s: str) -> int:
        words = s.strip().split()

        if not words:
            return 0

        return len(words[-1])
```

14. Longest Common Prefix

```
class Solution:
    def longestCommonPrefix(self, strs: List[str]) -> str:
```

```
        strs=sorted(strs)
        if len(strs)==1:
           return strs[0]
        ans=""
        s=strs[0]
        q=0
        for i in s:
           for j in strs:
             if i!=j[q]:
                    return ans
           else:
              ans=ans+i
              q+=1
        return ans
```

290. Word Pattern

```
class Solution:
    def wordPattern(self, pattern: str, s: str) -> bool:
        s1=s.split(" ")
        mapChar={}
        mapWord={}
        if len(pattern)!=len(s1):
            return False

        for char,word in zip(pattern,s1):
            if char not in mapChar:
                if word in mapWord:
                    return False
                else:
                    mapChar[char]=word
                    mapWord[word]=char
            else:
                if mapChar[char]!=word:
```

20. Valid Parentheses

```
class Solution:
    def isValid(self, s: str) -> bool:
        a=[]
        for i in s:
            if i=="(" or i=="{" or i=="[":
                a.append(i)
            elif (i==")" or i=="}" or i=="]")and len(a)==0:
                return False
            else:
                if i==")" and a[-1]=="(":
                    a.pop()
                elif i=="}"and a[-1]=="{":
                    a.pop()
                elif i=="]" and a[-1]=="[":
                    a.pop()
                else:
                    a.append(i)
        if len(a)==0:
            return True
        else:
            return False
```

45. Binary Tree Postorder Traversal

```
class Solution:
    def postorderTraversal(self, root: Optional[TreeNode]) -> Li
        s=[]
        def order(root,s):
            if root:
                order(root.left,s)
                order(root.right,s)
                s.append(root.val)
```

```
        order(root,s)
        return s
```

94. Binary Tree Inorder Traversal

```
class Solution:
    def inorderTraversal(self, root: Optional[TreeNode]) -> List
        s=[]
        def order(root,s):
            if root:
                order(root.left,s)
                s.append(root.val)
                order(root.right,s)
        order(root,s)
        return s
```

.04. Maximum Depth of Binary Tree

```
class Solution:
    def maxDepth(self, root: Optional[TreeNode]) -> int:
        def height(root):
            if root:
                leftnode=height(root.left)
                rightnode=height(root.right)
                return max(leftnode,rightnode)+1
            else:
                return 0
        a=height(root)
        return a
```

.11. Minimum Depth of binary Tree

```
class Solution:
    def minDepth(self, root: Optional[TreeNode]) -> int:
        def height(root):
```

```
            if root:
                if root.left is None:
                    return height(root.right)+1
                if root.right is None:
                    return height(root.left)+1
                leftnode=height(root.left)
                rightnode=height(root.right)
                return min(leftnode,rightnode)+1
            else:
                return 0
        a=height(root)
        return a
```

00. Same Tree

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def isSameTree(self, p: Optional[TreeNode], q: Optional[Tree
        def same(p,q):
            if p is None and q is None:
                return True
            if p is None or q is None:
                return False
            if p.val!=q.val:
                return False
            return same(p.left,q.left) and same(p.right,q.right)
        return same(p,q)
```

101.Symmetric Tree

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def isSymmetric(self, root: Optional[TreeNode]) -> bool:
        def same(p,q):
            if p is None and q is None:
                return True
            if p is None or q is None:
                return False
            return (p.val==q.val) and same(p.left,q.right) and s
        return same(root.left,root.right)
```

222. Count Complete Tree Nodes

```
# Definition for a binary tree node.
# class TreeNode:
#     def _init_(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def countNodes(self, root: Optional[TreeNode]) -> int:
        s=[]
        def order(root,s):
            if root:
                order(root.left,s)
                s.append(root.val)
                order(root.right,s)
        order(root,s)
```

```
        return len(s)
```