1.TWO SUM

Given an array of integers nums and an integer target, return *indices of the two numbers such that they add up to target*.

You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

#CODE

class Solution(object):

def twoSum(self, nums, target):

"""

:type nums: List[int]

:type target: int

:rtype: List[int]

"""

num\_to\_index = {}

for i, n in enumerate(nums):

n2 = target - n

if n2 in num\_to\_index:

return [num\_to\_index[n2], i]

num\_to\_index[n] = i

return [] # Return an empty list if no solution is found

TESTCASE1:

nums =[2,7,11,15]

target =9

2.ADD TWO NUMBERS

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

#CODE

class Solution(object):

def addTwoNumbers(self, l1, l2):

sum = ListNode()

tail = sum

carry = 0

while l1 or l2 or carry:

val1 = l1.val if l1 else 0

val2 = l2.val if l2 else 0

total = val1 + val2 + carry

carry = total//10

tail.next = ListNode(total%10)

tail = tail.next

if l1: l1 = l1.next

if l2: l2 = l2.next

return sum.next

TESTCASE1

l1 =[2,4,3]

l2 =[5,6,4]

3.LONGEST SUBSTRING WITHOUT REPEATING CHARACTERS

Given a string s, find the length of the **longest** **substring** without duplicate characters.

 #CODE

class Solution:

# @return an integer

def lengthOfLongestSubstring(self, s):

start = maxLength = 0

usedChar = {}

for i in range(len(s)):

if s[i] in usedChar and start <= usedChar[s[i]]:

start = usedChar[s[i]] + 1

else:

maxLength = max(maxLength, i - start + 1)

usedChar[s[i]] = i

return maxLength

TESTCASE1:

S= ”abcabcbb”

4.MEDIAN OF TWO SORTED ARRAYS

Given two sorted arrays nums1 and nums2 of size m and n respectively, return **the median** of the two sorted arrays.

The overall run time complexity should be O(log (m+n)).

#CODE

 class Solution(object):

def findMedianSortedArrays(self, nums1, nums2):

"""

:type nums1: List[int]

:type nums2: List[int]

:rtype: float

"""

# Ensure nums1 is the smaller array

if len(nums1) > len(nums2):

nums1, nums2 = nums2, nums1

m, n = len(nums1), len(nums2)

total = m + n # Total number of elements

half = (total + 1) // 2 # Half length (for partitioning)

# Binary search boundaries on nums1

left, right = 0, m

while left <= right:

# Partition nums1

partition\_1 = (left + right) // 2

# Partition nums2 accordingly

partition\_2 = half - partition\_1

# Left side max and right side min for nums1

max\_left1 = float('-inf') if partition\_1 == 0 else nums1[partition\_1 - 1]

min\_right1 = float('inf') if partition\_1 == m else nums1[partition\_1]

# Left side max and right side min for nums2

max\_left2 = float('-inf') if partition\_2 == 0 else nums2[partition\_2 - 1]

min\_right2 = float('inf') if partition\_2 == n else nums2[partition\_2]

# ---------------------------------------------------------

# Check if partition is valid

# ---------------------------------------------------------

if max\_left1 <= min\_right2 and max\_left2 <= min\_right1:

# Odd total length → median is max of left halves

if total % 2 == 1:

return max(max\_left1, max\_left2)

# Even total length → average of max(left) & min(right)

else:

return (max(max\_left1, max\_left2) + min(min\_right1, min\_right2)) / 2.0

# If nums1's left partition is too big → move search left

elif max\_left1 > min\_right2:

right = partition\_1 - 1

# Else, move search right

else:

left = partition\_1 + 1

# Edge case (should never hit this if inputs are valid)

return 0

TESTCASE1:

nums1 =[1,3]

nums2 =[2]

5.LONGEST PALINDROMIC SUBSTRING

Given a string s, return *the longest* *palindromic* *substring* in s.

 #CODE

class Solution(object):

def longestPalindrome(self, s):

res = ""

for i in range(len(s)):

l, r = i, i

while l >= 0 and r < len(s) and s[l] == s[r]:

if len(s[l:r+1]) > len(res):

res = s[l:r+1]

l -= 1

r += 1

l, r = i, i + 1

while l >= 0 and r < len(s) and s[l] == s[r]:

if len(s[l:r+1]) > len(res):

res = s[l:r+1]

l -= 1

r += 1

return res

TESTCASE1:

S= “babad”

6.ZIGZAG CONVERSION

The string "PAYPALISHIRING" is written in a zigzag pattern on a given number of rows like this: (you may want to display this pattern in a fixed font for better legibility)

P A H N

A P L S I I G

Y I R

And then read line by line: "PAHNAPLSIIGYIR"

Write the code that will take a string and make this conversion given a number of rows:

string convert(string s, int numRows);

#CODE

class Solution(object):

def convert(self, s, numRows):

if numRows == 1: return s

a=""

for i in range(numRows):

for j in range(i,len(s),2\*(numRows-1)):

a+=s[j]

if(i>0 and i<numRows-1 and j+2\*(numRows-1)-2\*i < len(s)):

a+=s[j+2\*(numRows-1)-2\*i]

return a

"""

:type s: str

:type numRows: int

:rtype: str

"""

TESTCASE1:

s ="PAYPALISHIRING"

numRows =3

7.REVERSE INTEGER

Given a signed 32-bit integer x, return x*with its digits reversed*. If reversing x causes the value to go outside the signed 32-bit integer range [-231, 231 - 1], then return 0.

#CODE

class Solution(object):

def reverse(self, x):

min,max=-2\*\*31,2\*\*31 -1

if(x<max and x>min):

if(x!=0):

if(x<0):

x=x\*-1

sx=str(x)

sx=-int(sx[::-1])

if(sx<max and sx>min):

return sx

else:

return 0

else:

sx=str(x)

sx=int(sx[::-1])

if(sx<max and sx>min):

return sx

else:

return 0

else:

return 0

else:

return 0

TESTCASE1: X=123

8.STRING TO INTEGER(atoi)

Implement the myAtoi(string s) function, which converts a string to a 32-bit signed integer.

#CODE

class Solution(object):

def myAtoi(self, s):

num = '0123456789'

res = ''

for x in s:

if x == ' ' and len(res) == 0:

continue

if x != ' ' and (x in '-+' or x in num) and len(res) == 0:

res += x

elif x in num:

res += x

else:

break

if res == '' or res in '-+':

return 0

else:

# to avoid int casting simply run a loop and use ord(char) - ord('0')

if int(res) < -(2\*\*31):

return -(2\*\*31)

elif int(res) > (2\*\*31 - 1):

return (2\*\*31 - 1)

else:

return int(res)

TESTCASE1:

S=”42”

9.PALINDROME NUMBER

Given an integer x, return true*if*x*is a****palindrome****, and*false*otherwise*.

#CODE

class Solution(object):

def isPalindrome(self, x):

"""

:type x: int

:rtype: bool

"""

if x<0:

return False

else:

r=0

num=x

while x!=0:

r=r\*10+x%10

x=x//10

return r==num

TESTCASE1:

X=121

10.REGULAR EXPRESSION MATCHING

Given an input string s and a pattern p, implement regular expression matching with support for '.' and '\*' where:

* '.' Matches any single character.​​​​
* '\*' Matches zero or more of the preceding element.

The matching should cover the **entire** input string (not partial).

#CODE

class Solution:

def isMatch(self, s, p):

mat = [[False] \* (len(p) + 1) for \_ in range(len(s) + 1)]

mat[0][0] = True

for i in range(1, len(p) + 1):

if p[i - 1] == '\*':

mat[0][i] = mat[0][i - 2]

for i in range(1, len(s) + 1):

for j in range(1, len(p) + 1):

if p[j - 1] == '.' or p[j - 1] == s[i - 1]:

mat[i][j] = mat[i - 1][j - 1]

elif p[j - 1] == '\*':

mat[i][j] = mat[i][j - 2]

if p[j - 2] == '.' or p[j - 2] == s[i - 1]:

mat[i][j] = mat[i][j] or mat[i - 1][j]

else:

mat[i][j] = False

return mat[len(s)][len(p)]

TESTCASE1:

s ="aa"

p =”a”