**LAB DAY-4 (07-06-2024)**

**1. Counting Elements**

We need to count how many elements xxx in the array have x+1x + 1x+1 also present in the array.

PROGRAM:

def count (arr):

set = set(arr)

n= sum(1 for x in arr if x + 1 in element\_set)

return n

print(count([1, 2, 3]))

OUTPUT: 2

**2. Perform String Shifts**

We need to perform a series of left and right shifts on a string.

PROGRAM:

def shift(s, shift):

total = sum(amount if direction == 1 else -amount for direction, amount in shift)

total %= len(s)

return s[-total:] + s[:-total]

print(shift("abc", [[0, 1], [1, 2]]))

OUTPUT: "cab"

**3. Leftmost Column with at Least a One**

We need to find the leftmost column with at least one 1 in a binary matrix.

PROGRAM:

def lcol(binaryMatrix):

rows, cols = binaryMatrix.dimensions()

currentrow, currentcol = 0, cols - 1

leftcol = -1

while current\_row < rows and current\_col >= 0:

if binaryMatrix.get(currentrow, currentcol) == 1:

leftcol = currentcol

currentcol -= 1

else:

currentrow += 1

return lcol

**4. First Unique Number**

We need to find and manage the first unique number in a queue.

from collections import OrderedDict, deque

class FirstUnique:

def \_\_init\_\_(self, nums):

self.count = {}

self.queue = deque()

for num in nums:

self.add(num)

def showFirstUnique(self):

while self.queue and self.count[self.queue[0]] > 1:

self.queue.popleft()

return self.queue[0] if self.queue else -1

def add(self, value):

if value in self.count:

self.count[value] += 1

else:

self.count[value] = 1

self.queue.append(value)

firstUnique = FirstUnique([2, 3, 5])

print(firstUnique.showFirstUnique())

OUTPUT: 2

**5. Check If a String Is a Valid Sequence from Root to Leaves Path in a Binary Tree**

We need to determine if a given array represents a valid sequence from root to leaf in a binary tree.

def Valid(root, arr):

def dfs(node, index):

if not node or index >= len(arr) or node.val != arr[index]:

return False

if index == len(arr) - 1:

return not node.left and not node.right

return dfs(node.left, index + 1) or dfs(node.right, index + 1)

return dfs(root, 0)

**6. Kids With the Greatest Number of Candies**

We need to determine which kids can have the greatest number of candies after receiving extra candies.

def kids (candies,y):

max\_candies = max(candies)

return [candy + y >= max\_candies for candy in candies]

print(kids([2, 3, 5, 1, 3], 3))

OUTPUT: [True, True, True, False, True]

**7. Max Difference You Can Get From Changing an Integer**

We need to find the maximum difference between two integers obtained by replacing digits in the given integer.

**PROGRAM:**

def max (num):

num= str(num)

a = int(num.replace(num\_str[0], '9'))

b = int(num.replace(num\_str[0], '1'))

return a - b

print(max(555))

**OUTPUT: 888**

**8. Check If a String Can Break Another String**

We need to determine if one string can break another or vice versa.

PROGRAM:

Def break(s1, s2):

Str1, Str2= sorted(s1), sorted(s2)

return all(a >= b for a, b in zip(Str1,Str2)) or all(a <= b for a, b in zip(Str1,Str2))

print(break("abc", "xya"))

**OUTPUT:** True

**9. Number of Ways to Wear Different Hats to Each Other**

We need to find the number of ways to assign hats to people such that each person gets a different hat.

def Noofhats(hats):

MOD = 10\*\*9 + 7

n = len(hats)

dp = [0] \* (1 << n)

dp[0] = 1

hattopeople = {}

for i, personhats in enumerate(hats):

for hat in personhats:

if hat not in hattopeople:

hattopeople[hat] = []

hattopeople[hat].append(i)

for hat in range(1, 41):

if hat in hattopeople:

for mask in range((1 << n) - 1, -1, -1):

for person in hattopeople[hat]:

if mask & (1 << person) == 0:

dp[mask | (1 << person)] = (dp[mask | (1 << person)] + dp[mask]) % MOD

return dp[(1 << n) - 1]

print(noofways([[3, 4], [4, 5], [5]]))

**Output: 1**

**10. Destination City**

We need to find the destination city in a list of paths.

def dest(paths):

outgoing = {start for start, end in paths}

for start, end in paths:

if end not in outgoing:

return end

print(dest([["a", "b"], ["b", "c"], ["c", "a"]]))

OUTPUT: "a"