1).Counting Elements

Given an integer array arr, count how many elements x there are, such that x + 1 is also in arr. If there are duplicates in arr, count them separately.

Example

Input: arr = [1,2,3]

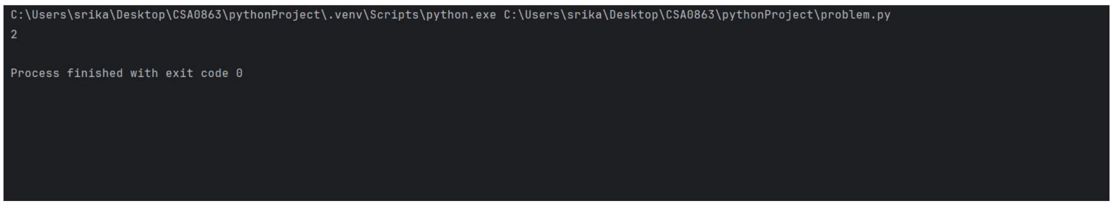
Output: 2

Explanation: 1 and 2 are counted cause 2 and 3 are in arr..

Program:

```
def count_elements(arr):
num_count = {}
for num in arr:
num_count[num] = num_count.get(num, 0)
+ 1
count = 0
for num in arr:
if num + 1 in num_count:
count += num_count[num]
return count
arr = [1, 2, 3]
print(count_elements(arr))
```

Output:



```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
2

Process finished with exit code 0
```

Time complexity:O(2n)

2).Counting Elements

Perform String Shifts

You are given a string s containing lowercase English letters, and a matrix shift, where shift[i] = [directioni, amounti]:

● directioni can be 0 (for left shift) or 1 (for right shift).

● amounti is the amount by which string s is to be shifted.

● A left shift by 1 means remove the first character of s and append it to the end.

● Similarly, a right shift by 1 means remove the last character of s and add it to the beginning.

Return the final string after all operations.

Example 1:

Input: s = "abc" , shift = [[0,1],[1,2]]

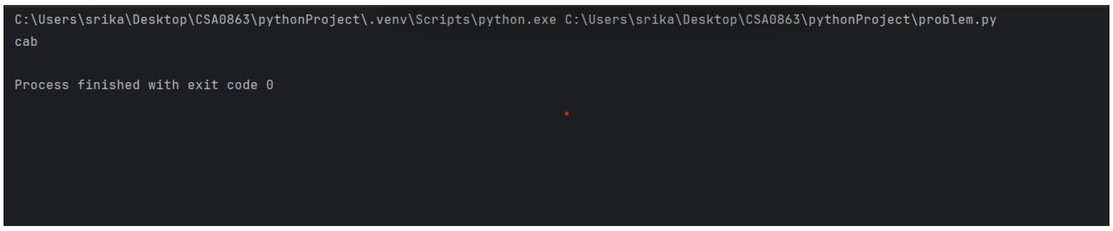
Output: "cab"

Program:

```
def string_shift(s, shift):
total_shift = 0
```

```
for direction, amount in shift:
if direction == 0:
total_shift -= amount
else:
total_shift += amount
total_shift %= len(s)if total_shift < 0:
s = s[-total_shift:] + s[:-total_shift]
elif total_shift > 0:
s = s[-total_shift:] + s[:-total_shift]
return s
s = "abc"
shift = [[0, 1], [1, 2]]
print(string_shift(s, shift))
```

Output:



```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
cab

Process finished with exit code 0
```

Time complexity:O(n)

3).. Leftmost Column with at Least a One

A row-sorted binary matrix means that all elements are 0 or 1 and each row of the matrix is sorted in non-decreasing order.

Given a row-sorted binary matrix binaryMatrix, return the index (0-indexed) of the leftmost column with a 1 in it. If such an index does not exist, return -1.

You can't access the Binary Matrix directly. You may only access the matrix using a BinaryMatrix interface:

- BinaryMatrix.get(row, col) returns the element of the matrix at index (row, col) (0-indexed).
- BinaryMatrix.dimensions() returns the dimensions of the matrix as a list of 2 elements [rows, cols], which means the matrix is rows x cols.

Submissions making more than 1000 calls to BinaryMatrix.get will be judged Wrong Answer. Also, any solutions that attempt to circumvent the judge will result in disqualification.

For custom testing purposes, the input will be the entire binary matrix mat. You will not have access to the binary matrix directly.

Example 1:Input: mat = [[0,0],[1,1]]

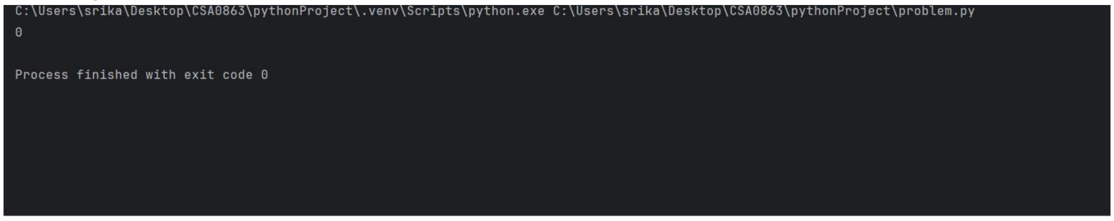
Output: 0

Program:

```
class BinaryMatrix:
def __init__(self, matrix):
self.matrix = matrix
def get(self, row, col):
return self.matrix[row][col]
def dimensions(self):
```

```
return [len(self.matrix), len(self.matrix[0])]
def leftmost_column_with_one(binaryMatrix):
rows, cols = binaryMatrix.dimensions()
row = 0
col = cols - 1
leftmost_col = -1
while row < rows and col >= 0:
if binaryMatrix.get(row, col) == 1:
leftmost_col = col
col -= 1
else:
row += 1
return leftmost_col
mat = [[0, 0], [1, 1]]
binaryMatrix = BinaryMatrix(mat)
print(leftmost_column_with_one(binaryMatrix))
```

Output:



```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
0

Process finished with exit code 0
```

Time complexity:O(n+m)

4. Search in Rotated Sorted Array

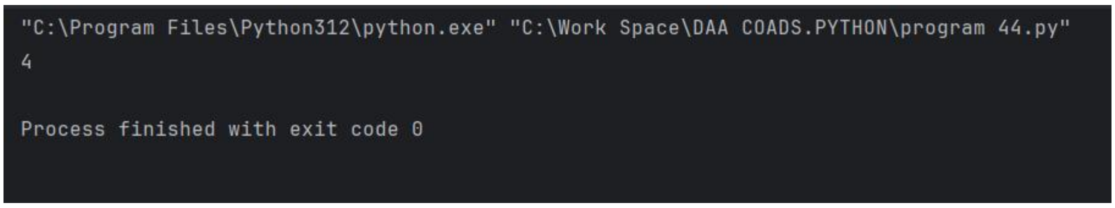
There is an integer array nums sorted in ascending order (with distinct values).

Prior to being passed to your function, nums is possibly rotated at an unknown pivot

index
k (1 <= k < nums.length) such that the resulting array is [nums[k], nums[k+1], ...,
nums[n 1], nums[0], nums[1], ..., nums[k-1]] (0-indexed). For example, [0,1,2,4,5,6,7]
might be
rotated at pivot index 3 and become
[4,5,6,7,0,1,2].
Given the array nums after the possible rotation and an integer target, return the
index of
target if it is in nums, or -1 if it is not in nums.
PROGRAM:

```
ddef search(nums, target):
left, right = 0, len(nums) - 1
while left <= right:
mid = (left + right) // 2
if nums[mid] == target:
return mid
if nums[left] <= nums[mid]:
if nums[left] <= target < nums[mid]:
right = mid - 1
else:
left = mid + 1
else:
if nums[mid] < target <= nums[right]:
left = mid + 1
```

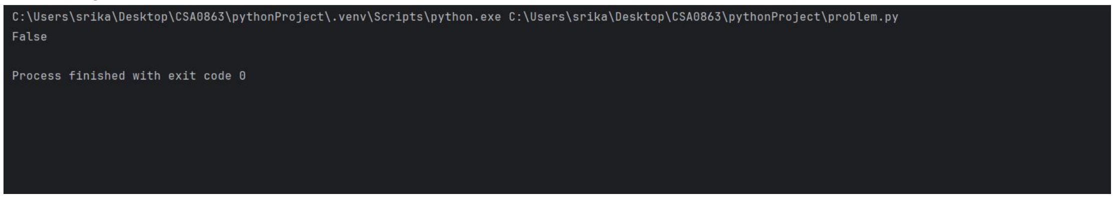
else:
right = mid - 1
return -1
# Example usage
nums = [4, 5, 6, 7, 0, 1, 2]
target = 0
print(search(nums, target)) # Output: 4
Output:



```
"C:\Program Files\Python312\python.exe" "C:\Work Space\DAA COADS.PYTHON\program 44.py"
4

Process finished with exit code 0
```

Time complexity:
O(log n)
4.)Check If a String Is a Valid Sequence from Root to Leaves Path in a Binary Tree
Given a binary tree where each path going from the root to any leaf form a valid
sequence, check if a given string is a valid sequence in such binary tree.
We get the given string from the concatenation of an array of integers arr and the
concatenation of all values of the nodes along a path results in a sequence in the given
binary tree.
Program:
class TreeNode:
def __init__(self, val=0, left=None,

```
right=None):
self.val = val
self.left = left
self.right = right
def isValidSequence(root, arr):
def dfs(node, index):
if not node:
return False
if index == len(arr) - 1:
return node.val == arr[index] and not
node.left and not node.right
if node.val != arr[index]:
return Falsereturn dfs(node.left, index + 1) or
dfs(node.right, index + 1)
return dfs(root, 0)
root = TreeNode(0)
root.left = TreeNode(1)
root.right = TreeNode(0)
root.left.left = TreeNode(0)
root.left.right = TreeNode(1)
root.right.left = TreeNode(0)
root.right.right = None
root.left.left.left = None
root.left.left.right = None
root.left.right.left = TreeNode(1)
root.left.right.right = TreeNode(0)
arr = [0, 1, 0, 1]
```

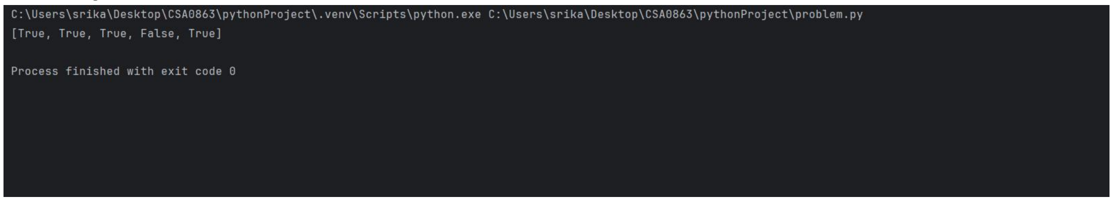
print(isValidSequence(root, arr))
Output:


```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
False

Process finished with exit code 0
```

Time complexity:O(n)
Q).. Kids With the Greatest Number of Candies
There are n kids with candies. You are given an integer array candies, where each
candies[i] represents the number of candies the ith kid has, and an integer extraCandies,
denoting the number of extra candies that you have.
Return a boolean array result of length n, where result[i] is true if, after giving the ith kid
all the extraCandies, they will have the greatest number of candies among all the kids, or
false otherwise.
Note that multiple kids can have the greatest number of candies.Program:

```
def kidsWithCandies(candies, extraCandies):
max_candies = max(candies)
result = []
for candy in candies:
result.append(candy + extraCandies >=
max_candies)
return result
```

candies = [2, 3, 5, 1, 3]
extraCandies = 3
print(kidsWithCandies(candies, extraCandies))
Output:

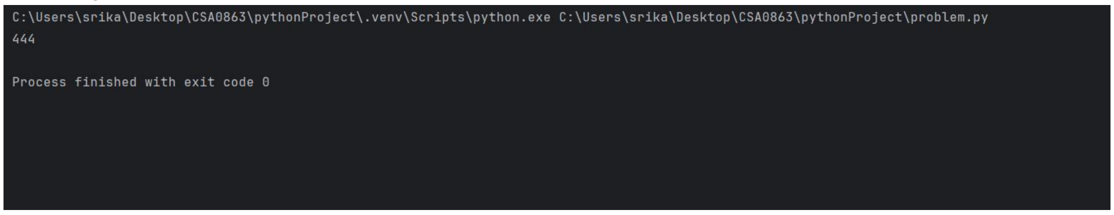


```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
[True, True, True, False, True]

Process finished with exit code 0
```

Time complexity:O(n)
7)..Max Difference You Can Get From Changing an Integer
You are given an integer num. You will apply the following steps exactly two times:
● Pick a digit x (0 <= x <= 9).
● Pick another digit y (0 <= y <= 9). The digit y can be equal to x.
● Replace all the occurrences of x in the decimal representation of num by y.
● The new integer cannot have any leading zeros, also the new integer cannot be 0.
Let a and b be the results of applying the operations to num the first and second times, respectively.
Return the max difference between a and b
Program:
def maxDiff(num):
num_str = str(num)
max_diff = 0

```
for x in range(10):
for y in range(10):
a_str = num_str.replace(str(x), str(y))
if a_str != "0" and not
a_str.startswith("0"):
a = int(a_str)
max_diff = max(max_diff, abs(a - num))
return max_diffnum = 555
print(maxDiff(num))
```

Output:



```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
444

Process finished with exit code 0
```

Time complexity:O(1)

Q)..Number of Ways to Wear Different Hats to Each Other

There are n people and 40 types of hats labeled from 1 to 40.

Given a 2D integer array hats, where hats[i] is a list of all hats preferred by the ith person.

Return the number of ways that the n people wear different hats to each other.

Since the answer may be too large, return it modulo 109 + 7.

Example 1:

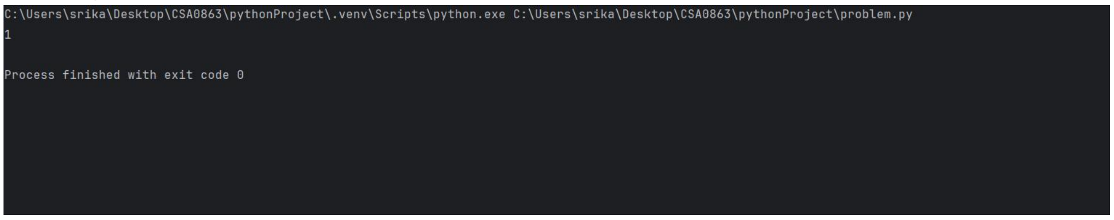
Input: hats = [[3,4],[4,5],[5]]

Output: 1

Program:

```
MOD = 10**9 + 7
def numberWays(hats):
preferences = {}
for i, person_hats in enumerate(hats):
for hat in person_hats:
if hat not in preferences:
preferences[hat] = []
preferences[hat].append(i)
n = len(hats)
dp = [0] * (1 << n)
dp[0] = 1
for hat in range(1, 41):if hat in preferences:
for i in range(len(dp) - 1, -1, -1):
for person in preferences[hat]:
if not (i & (1 << person)):
dp[i | (1 << person)] += dp[i]
dp[i | (1 << person)] %= MOD
return dp[(1 << n) - 1]
hats = [[3, 4], [4, 5], [5]]
print(numberWays(hats))
```

Output:



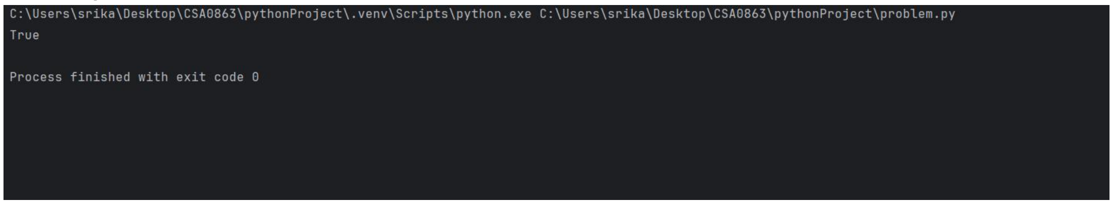
```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
1

Process finished with exit code 0
```

Time complexity:O(1)

Q)..Check If a String Can Break Another String

Given two strings: s1 and s2 with the same size, check if some permutation of string s1
can break some permutation of string s2 or vice-versa. In other words s2 can break s1 or
vice-versa.
A string x can break string y (both of size n) if x[i] >= y[i] (in alphabetical order) for all i between 0 and n-1.
Example 1:
Input: s1 = "abc" , s2 = "xya"
Output: true
Program:

```
def canBreak(s1, s2):
s1_sorted = sorted(s1)
s2_sorted = sorted(s2)
s1_can_break_s2 = all(s1_char >= s2_char for
s1_char, s2_char in zip(s1_sorted, s2_sorted))
s2_can_break_s1 = all(s2_char >= s1_char for
s1_char, s2_char in zip(s1_sorted, s2_sorted))
return s1_can_break_s2 or s2_can_break_s1
s1 = "abc"
s2 = "xya"
print(canBreak(s1, s2))
```

Output:


```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
True

Process finished with exit code 0
```

Time complexity:O(nlogn)

Q)..Destination City

You are given the array paths, where paths[i] = [cityAi, cityBi] means there exists a direct path going from cityAi to cityBi. Return the destination city, that is, the city without any path outgoing to another city.

It is guaranteed that the graph of paths forms a line without any loop, therefore, there will be exactly one destination city.

Example 1:

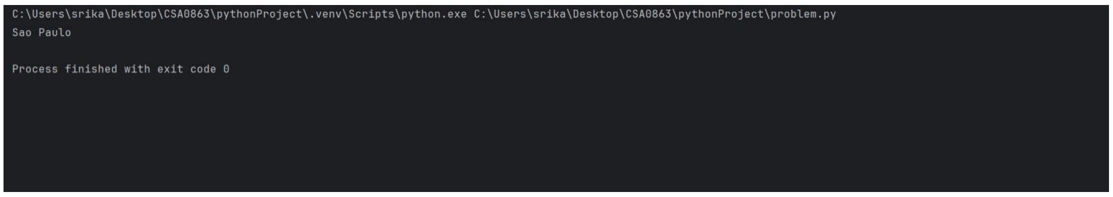
Input: paths = [["London" , "New York"],["New York" , "Lima"],["Lima" , "Sao Paulo"]]

Output: "Sao Paulo"

Program:

```
def destCity(paths):
outgoing_cities = set()
for path in paths:
outgoing_cities.add(path[0])
for path in paths:
if path[1] not in outgoing_cities:
return path[1]
paths = [["London" , "New York"], ["New York" ,
"Lima"], ["Lima" , "Sao Paulo"]]
print(destCity(paths))
```

Output:



```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
Sao Paulo

Process finished with exit code 0
```

Time complexity:O(n)