

1. Perform String Shifts You are given a string *s* containing lowercase English letters, and a matrix *shift*, where *shift[i] = [direction_i, amount_i]*: • *direction_i* can be 0 (for left shift) or 1 (for right shift). • *amount_i* 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" Explanation: [0,1] means shift to left by 1. "abc" -> "bca" [1,2] means shift to right by 2. "bca" -> "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)
    return s[-total_shift:] + s[:-total_shift]

s = "abcdefg"
shift = [[1,1], [1,1], [0,2], [1,3]]
result = string_shift(s, shift)
print(result)
```

OUTPUT:

```
PS C:\Users\chall\OneDrive\Desktop\DAA> & C:/Users/chall/AppData/Local/Programs/Python/Python312/python.exe
"
efgabcd
PS C:\Users\chall\OneDrive\Desktop\DAA> █
```

2. 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

PROGRAM:

```
class BinaryMatrix:
```

```
    def init (self, matrix):
```

```
        self.matrix = matrix
```

```
    def get(self, row: int, col: int) -> int:
```

```
        return self.matrix[row][col]
```

```
    def dimensions(self) -> list:
```

```
        return [len(self.matrix), len(self.matrix[0])]
```

```
def leftMostColumnWithOne(binaryMatrix: 'BinaryMatrix') -> int:
```

```
    rows, cols = binaryMatrix.dimensions()
```

```
    current_row = 0
```

```
    current_col = cols - 1
```

```
    leftmost_col = -1
```

```
    while current_row < rows and current_col >= 0:
```

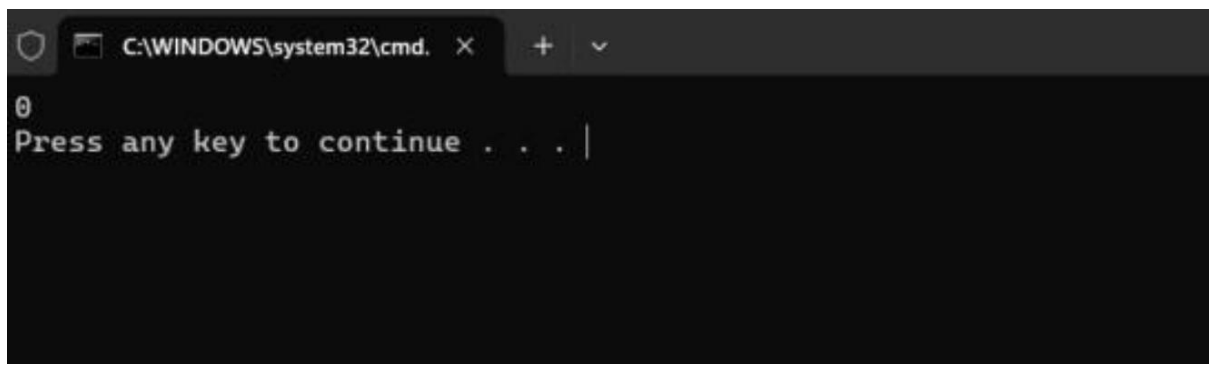
```
        if binaryMatrix.get(current_row, current_col) == 1:
```

```

        leftmost_col = current_col
        current_col -= 1
    else:
        current_row += 1
    return leftmost_col
mat = [[0, 0], [1,1]]
binaryMatrix = BinaryMatrix(mat)
result = leftMostColumnWithOne(binaryMatrix)
print(result)

```

OUTPUT:



```

C:\WINDOWS\system32\cmd. X + v
0
Press any key to continue . . . |

```

3. First Unique Number You have a queue of integers, you need to retrieve the first unique integer in the queue. Implement the **FirstUnique** class:

- **FirstUnique(int[] nums)** Initializes the object with the numbers in the queue.
- **int showFirstUnique()** returns the value of the first unique integer of the queue, and returns -1 if there is no such integer.
- **void add(int value)** insert value to the queue.

PROGRAM:

```

from collections import OrderedDict, deque

class FirstUnique:
    def init (self, nums):
        self.queue = deque()

```

```

self.counts = {}
for num in nums:
    self.add(num)
def showFirstUnique(self) -> int:
    while self.queue and self.counts[self.queue[0]] > 1:
        self.queue.popleft()
    return self.queue[0] if self.queue else -1
def add(self, value: int) -> None:
    self.counts[value] = self.counts.get(value, 0) + 1
    if self.counts[value] == 1:
        self.queue.append(value)
firstUnique = FirstUnique([2, 3, 5])
print(firstUnique.showFirstUnique())
firstUnique.add(5)
print(firstUnique.showFirstUnique())
firstUnique.add(2)
print(firstUnique.showFirstUnique())
firstUnique.add(3)
print(firstUnique.showFirstUnique())

```

OUTPUT:



```

C:\WINDOWS\system32\cmd. x + v
2
2
3
-1
Press any key to continue . . . |

```

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. Example 1: Input: root = [0,1,0,0,1,0,null,null,1,0,0], arr = [0,1,0,1] Output: true Explanation: The path 0 -> 1 -> 0 -> 1 is a valid sequence (green color in the figure). Other valid sequences are: 0 -> 1 -> 1 -> 0 0 -> 0 -> 0

PROGRAM:

class Node:

def init (self, val):

self.val = val

self.left = None

self.right = None

def exist_path(root):

if root is None:

return False

if root.left is None and root.right is None:

return root.val

return False

root = Node(5)

root.left = Node(2)

root.right = Node(7)

root.left.left = Node(1)

root.left.right = Node(4)

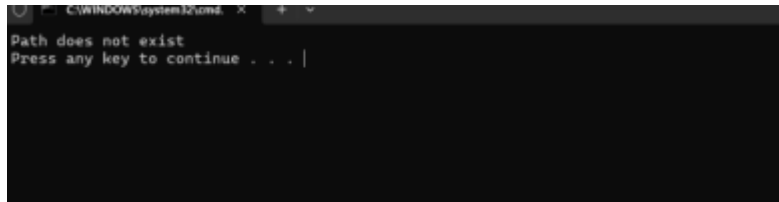
root.right.left = Node(6)

root.right.right = Node(8)

if exist_path(root):

```
print("flath exists")
else:
    print("flath does not exist")
```

OUTPUT:



5. 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 i th 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 i th 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 = [candy + extraCandies >= max_candies for candy in candies]
    return result

candies = [2, 3, 5, 1, 3]
extraCandies = 3
output = kidsWithCandies(candies, extraCandies)
print(output)
```

OUTPUT:

```
PS C:\Users\chall\OneDrive\Desktop\DAA> & C:/Users/chall/AppData/Local/Programs/Python/Python312/python.exe  
"  
[True, True, True, False, True]  
PS C:\Users\chall\OneDrive\Desktop\DAA>
```

6. 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 ($0 \leq x \leq 9$). • Pick another digit y ($0 \leq y \leq 9$). 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.

PROGRAM:

```
def maxDiff(num: int) -> int:  
    num_str = str(num)  
    max_diff = 0  
    for i, digit in enumerate(num_str):  
        if digit != '9':  
            new_num_str = num_str.replace(digit, '9')  
            max_diff = max(max_diff, int(new_num_str) - num)  
            break  
    if num_str[0] != '1':  
        new_num_str = num_str.replace(num_str[0], '1')  
        max_diff = max(max_diff, num - int(new_num_str))  
    return max_diff  
print(maxDiff(9))
```

OUTPUT:

```
PS C:\Users\chall\OneDrive\Desktop\DAA> & C:/Users/chall/AppData/Local/Programs/Python/Python312/python.exe  
"  
8  
PS C:\Users\chall\OneDrive\Desktop\DAA>
```

7. 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 $x[i] \geq y[i]$ (in alphabetical order) for all i between 0 and n-1. Example 1: Input: s1 = "abc", s2 = "xya" Output: true Explanation: "ayx" is a permutation of s2="xya" which can break to string "abc" which is a permutation of s1="abc".

PROGRAM:

```
def checkIfCanBreak(s1, s2):  
    s1_sorted = sorted(s1)  
    s2_sorted = sorted(s2)  
    if all(s1_char >= s2_char for s1_char, s2_char in zip(s1_sorted, s2_sorted))  
    or all(s2_char >= s1_char for s1_char, s2_char in zip(s1_sorted, s2_sorted)):  
        return True  
    else:  
        return False  
s1 = "abe"  
s2 = "acd"  
print(checkIfCanBreak(s1, s2))
```

OUTPUT:


```
PS C:\Users\chall\OneDrive\Desktop\DAA> & C:/Users/chall/AppData/Local/Programs/Python/Python312/python.exe
"
False
PS C:\Users\chall\OneDrive\Desktop\DAA>
```

8. 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 i th 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 $10^9 + 7$. Example 1: Input: `hats = [[3,4],[4,5],[5]]` Output: 1 Explanation: There is only one way to choose hats given the conditions. First person choose hat 3, Second person choose hat 4 and last one hat 5. Example 2: Input: `hats = [[3,5,1],[3,5]]` Output: 4 Explanation: There are 4 ways to choose hats: (3,5), (5,3), (1,3) and (1,5)

PROGRAM:

```
def countWaysToWearHats(hats):
```

```
    MOD = 10**9 + 7
```

```
    n = len(hats)
```

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

```
    dp[0] = 1
```

```
    hat_to_people = [[] for _ in range(41)]
```

```
    for i in range(n):
```

```
        for hat in hats[i]:
```

```
            hat_to_people[hat].append(i)
```

```
    for hat in range(1, 41):
```

```
        for state in range((1 << n) - 1, -1, -1):
```

```
            for person in hat_to_people[hat]:
```

```
                if state (1 << person):
```

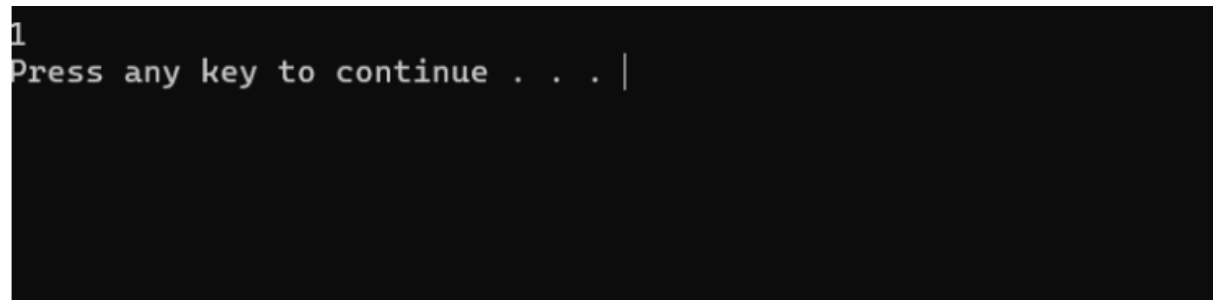
```
                    continue
```

```

        dp[state | (1 << person)] += dp[state]
        dp[state | (1 << person)] %= MOD
    return dp[(1 << n) - 1]
hats1 = [[3, 4], [4, 5], [5]]
print(countWaysToWearHats(hats1))

```

OUTPUT:



```

1
Press any key to continue . . . |

```

9. 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"` Explanation: Starting at `"London"` city you will reach `"Sao Paulo"` city which is the destination city. Your trip consist of: `"London" -> "New York" -> "Lima" -> "Sao Paulo"`.

PROGRAM:

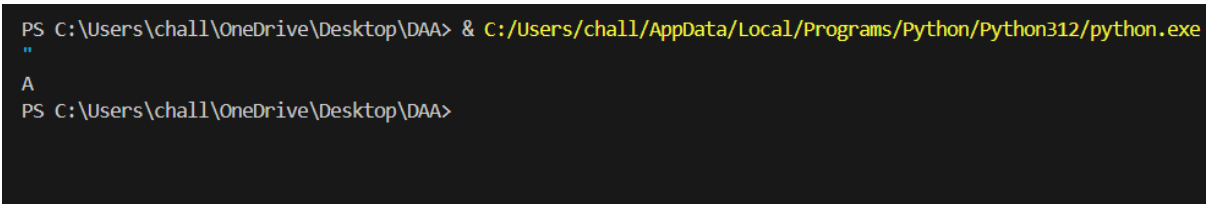
```

def destCity(paths):
    start_cities = set()
    dest_cities = set()
    for path in paths:
        start_cities.add(path[0])

```

```
    dest_cities.add(path[1])
    return (dest_cities - start_cities).pop()
paths = [["B","C"],["D","B"],["C","A"]]
destination = destCity(paths)
print(destination)
```

OUTPUT:



```
PS C:\Users\chall\OneDrive\Desktop\DAA> & C:/Users/chall/AppData/Local/Programs/Python/Python312/python.exe "
A
PS C:\Users\chall\OneDrive\Desktop\DAA>
```

10. 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):
    count = 0
    num_set = set(arr)
    for x in arr:
        if x + 1 in num_set:
            count += 1
    return count
arr1 = [1, 2, 3]
print(count_elements(arr1))
```

OUTPUT:

```
PS C:\Users\chall\OneDrive\Desktop\DAA> & C:/Users/chall/AppData/Local/Programs/Python/Python312/python.exe  
"  
2  
PS C:\Users\chall\OneDrive\Desktop\DAA>
```