.Maximum XOR of Two Non-Overlapping Subtrees

There is an undirected tree with n nodes labeled from 0 to n - 1. You are given the integer n and a 2D integer array edges of length n - 1, where edges[i] = [ai, bi] indicates that there is an edge between nodes ai and bi in the tree. The root of the tree is the node labeled 0.Each node has an associated value. You are given an array values of length n, where values[i] is the value of the ith node.Select any two non-overlapping subtrees. Your score is the bitwise XOR of the sum of the values within those subtrees.Return the maximum possible score you can achieve. If it is impossible to find two nonoverlapping subtrees, return 0. Note that:

● The subtree of a node is the tree consisting of that node and all of its descendants. ● Two subtrees are non-overlapping if they do not share any common node. Example 1:

Input: n = 6, edges = [[0,1],[0,2],[1,3],[1,4],[2,5]], values = [2,8,3,6,2,5]

Output: 24

Program:

```
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.children = []
def max_xor_subtrees(n, edges, values):
    adjacency_list = [[] for _ in range(n)]
    for edge in edges:
        adjacency_list[edge[0]].append(edge[1])
        adjacency_list[edge[1]].append(edge[0])
    def dfs(node, parent):
        xor_sum = values[node]
        max_xor = 0
        for child in adjacency_list[node]:
            if child != parent:
                child_xor, child_max_xor = dfs(child, node)
                xor_sum ^= child_xor
                max_xor = max(max_xor, child_max_xor)
        max_xor = max(max_xor, xor_sum)
        return xor_sum, max_xor
    _, max_xor = dfs(0, -1)  # Start DFS from the root node (node 0)
    return max_xor
n = 6
edges = [[0, 1], [0, 2], [1, 3], [1, 4], [2, 5]]
values = [2, 8, 3, 6, 2, 5]
print(max_xor_subtrees(n, edges, values))
```

Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\assignment 6\program 1.py"
12

Process finished with exit code 0
```

2.Form a Chemical Bond

SQL Schema

Table: Elements

Column Name	Type
symbol	varchar
type	enum
electrons	int

symbol is the primary key for this table. Each row of this table contains information of one element. type is an ENUM of type ('Metal', 'Nonmetal', 'Noble')

- If type is Noble, electrons is 0. - If type is Metal, electrons is the number of electrons that one atom of this element can give. - If type is Nonmetal, electrons is the number of electrons that one atom of this element
needs. Two elements can form a bond if one of them is 'Metal' and the other is 'Nonmetal'.Write an SQL
query to find all the pairs of elements that can form a bond.Return the result table in any order.The query result format is in the following example.

Program:

```
import sqlite3
conn = sqlite3.connect(':memory:')
cursor = conn.cursor()
cursor.execute('''
CREATE TABLE Elements (
   symbol TEXT PRIMARY KEY,
   type TEXT CHECK(type IN ('Metal', 'Nonmetal', 'Noble')),
   electrons INTEGER
);
''')
elements_data = [
   ('H', 'Nonmetal', 1),
   ('He', 'Noble', 0),
   ('Li', 'Metal', 1),
   ('Be', 'Metal', 2),
   ('B', 'Nonmetal', 3),
   ('C', 'Nonmetal', 4),
   ('N', 'Nonmetal', 5),
   ('O', 'Nonmetal', 6),
   ('F', 'Nonmetal', 7),
   ('Ne', 'Noble', 0)
]

cursor.executemany('INSERT INTO Elements (symbol, type, electrons) VALUES (?, ?, ?)',
elements_data)

query = '''
SELECT e1.symbol AS metal, e2.symbol AS nonmetal
FROM Elements e1
JOIN Elements e2
ON e1.type = 'Metal' AND e2.type = 'Nonmetal';
'''
cursor.execute(query)
result = cursor.fetchall()
for row in result:
   print(f"Metal: {row[0]}, Nonmetal: {row[1]}")

conn.close()
```

Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\PROGRAM 69.PY"
Metal: Li, Nonmetal: B
Metal: Li, Nonmetal: C
Metal: Li, Nonmetal: F
Metal: Li, Nonmetal: H
Metal: Li, Nonmetal: N
Metal: Li, Nonmetal: O
Metal: Be, Nonmetal: B
Metal: Be, Nonmetal: C
Metal: Be, Nonmetal: F
Metal: Be, Nonmetal: H
```

Time complexity:

O(n^2)

3. Minimum Cuts to Divide a Circle

A valid cut in a circle can be:
A cut that is represented by a straight line that touches two points on the edge of the circle and passes through its center, or A cut that is represented by a straight line that touches one point on the edge of the circle and its center. Some valid and invalid cuts are shown in the figures below. Given the integer n, return the minimum number of cuts needed to divide a circle into n equal slices. Example 1:
Input: n = 4
Output: 2
Program:

```
def min_cuts_to_divide_circle(n):
    if n == 1:
        return 0
    elif n == 2:
        return 1
    elif n % 2 == 0:
        return n // 2
    else:
        return n
n = 4
print(min_cuts_to_divide_circle(n))
```

Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\PROGRAM 69.PY"
2

Process finished with exit code 0
```

Time complexity:O(1)
4. Difference Between Ones and Zeros in Row and Column
You are given the customer visit log of a shop represented by a 0-indexed string customers consisting only of characters 'N' and 'Y':
● if the ith character is 'Y', it means that customers come at the ith hour
● whereas 'N' indicates that no customers come at the ith hour. If the shop closes at the jth hour (0 <= j <= n), the penalty is calculated as follows:
● For every hour when the shop is open and no customers come, the penalty increases by 1. ● For every hour when the shop is closed and customers come, the penalty increases by 1. Return the earliest hour at which the shop must be closed to incur a minimum penalty. Note that if a shop closes at the jth hour, it means the shop is closed at the hour j. Example 1:
Input: customers = "YYNY" Output: 2
Program:

```
def min_penalty_closing_hour(customers):
    n = len(customers)

    no_customer_penalty_before = [0] * (n + 1)
    customer_penalty_after = [0] * (n + 1)

    for i in range(1, n + 1):
        no_customer_penalty_before[i] = no_customer_penalty_before[i - 1] + (1 if customers[i - 1] == 'N' else 0)

    for i in range(n - 1, -1, -1):
        customer_penalty_after[i] = customer_penalty_after[i + 1] + (1 if customers[i] == 'Y' else 0)
    min_penalty = float('inf')
    best_hour = 0
    for j in range(n + 1):
        current_penalty = no_customer_penalty_before[j] + customer_penalty_after[j]
```

```
        if current_penalty < min_penalty:
            min_penalty = current_penalty
            best_hour = j
    return best_hour
customers = "YYNY"
print(min_penalty_closing_hour(customers))
```

Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\PROGRAM 69.PY"
2

Process finished with exit code 0
```

Time complexity:O(n)

5. Minimum Penalty for a Shop

You are given the customer visit log of a shop represented by a 0-indexed string customers consisting only of characters 'N' and 'Y':

● if the ith character is 'Y', it means that customers come at the ith hour

● whereas 'N' indicates that no customers come at the ith hour. If the shop closes at the jth hour (0 <= j <= n), the penalty is calculated as follows:

● For every hour when the shop is open and no customers come, the penalty increases by 1. ● For every hour when the shop is closed and customers come, the penalty increases by 1. Return the earliest hour at which the shop must be closed to incur a minimum penalty. Note that if a shop closes at the jth hour, it means the shop is closed at the hour j. Example 1:

Input: customers = "YYNY" Output: 2

Program:

```
def min_penalty_closing_hour(customers):
    n = len(customers)

    open_no_customers = [0] * (n + 1)
    closed_with_customers = [0] * (n + 1)

    for i in range(1, n + 1):
        open_no_customers[i] = open_no_customers[i - 1] + (1 if customers[i - 1] == 'N' else 0)

    for i in range(n - 1, -1, -1):
        closed_with_customers[i] = closed_with_customers[i + 1] + (1 if customers[i] == 'Y' else 0)
    min_penalty = float('inf')
    best_hour = 0
    for j in range(n + 1):
        current_penalty = open_no_customers[j] + closed_with_customers[j]
        if current_penalty < min_penalty:
            min_penalty = current_penalty
            best_hour = j

    return best_hour

customers = "YYNY"
print(min_penalty_closing_hour(customers))
```

Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\PROGRAM 69.PY"
2

Process finished with exit code 0
```

Time complexity:

O(n)

6. Count Palindromic Subsequences

Given a string of digits s, return the number of palindromic subsequences of s having length 5. Since the answer may be very large, return it modulo 109 + 7. Note:

● A string is palindromic if it reads the same forward and backward.

● A subsequence is a string that can be derived from another string by deleting some or no characters without changing the order of the remaining characters. Example 1:

Input: s = "103301" Output: 2

Program:

```
MOD = 10 ** 9 + 7
def count_palindromic_subsequences(s):
    n = len(s)
    dp = [[[0] * (5 + 1) for _ in range(n)] for _ in range(n)]

    for i in range(n):
        dp[i][i][1] = 1

    # Fill the DP table
    for length in range(2, 6):
        for i in range(n - length + 1):
            j = i + length - 1
            for k in range(10):
                char = str(k)
                if s[i] == char and s[j] == char:
                    if length == 2:
                        dp[i][j][2] = 1
                    elif length == 3:
                        dp[i][j][3] = dp[i + 1][j - 1][1]
                    elif length == 4:
                        dp[i][j][4] = dp[i + 1][j - 1][2]
                    elif length == 5:
                        dp[i][j][5] = dp[i + 1][j - 1][3]
                if s[i] == char:
                    for l in range(1, length):
                        dp[i][j][l] = (dp[i][j][l] + dp[i + 1][j][l]) % MOD
                if s[j] == char:
                    for l in range(1, length):
                        dp[i][j][l] = (dp[i][j][l] + dp[i][j - 1][l]) % MOD
                for l in range(1, length):
                    dp[i][j][l] = (dp[i][j][l] - dp[i + 1][j - 1][l - 1] + MOD) % MOD

    result = 0
    for i in range(n):
        for j in range(i, n):
            result = (result + dp[i][j][5]) % MOD
    return result

s = "103301"
print(count_palindromic_subsequences(s))
```

Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\PROGRAM 69.PY"
0

Process finished with exit code 0
```

Time complexity:O(n^3)

7.Find the Pivot Integer
Given a positive integer n, find the pivot integer x such that:
● The sum of all elements between 1 and x inclusively equals the sum of all elements between x and n inclusively. Return the pivot integer x. If no such integer exists, return -1. It is guaranteed that there will be at
most one pivot index for the given input. Example 1:
Input: n = 8
Output: 6
Program:

```
def find_pivot_integer(n):
    total_sum = n * (n + 1) // 2
    partial_sum = 0
    for x in range(1, n + 1):
        partial_sum += x
        if partial_sum == total_sum - partial_sum + x:
            return x
    return -1

n = 8
print(find_pivot_integer(n))
```

Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\PROGRAM 69.PY"
6

Process finished with exit code 0
```

Time complexity:
O(n)
8.. Append Characters to String to Make Subsequene
You are given two strings s and t consisting of only lowercase English letters. Return the minimum number of characters that need to be appended to the end of s so that t
becomes a subsequence of s. A subsequence is a string that can be derived from another string by deleting some or no
characters without changing the order of the remaining characters. Example 1:
Input: s = "coaching", t = "coding" Output: 4
Program:

```
def append_chars_to_make_subsequence(s, t):
    i, j = 0, 0
    while i < len(s) and j < len(t):
        if s[i] == t[j]:
            j += 1
        i += 1
    return len(t) - j

s = "coaching"
t = "coding"
print(append_chars_to_make_subsequence(s, t))
```

Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\PROGRAM 69.PY"
4

Process finished with exit code 0
```

Time complexity;
O(n+m)
9.Remove Nodes From Linked List

You are given the head of a linked list.Remove every node which has a node with a strictly greater value anywhere to the right side of it.Return the head of the modified linked list. Example 1:
Input: head = [5,2,13,3,8]
Output: [13,8]
Program:

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
def reverse_linked_list(head):
    prev = None
    current = head
    while current:
        next_node = current.next
        current.next = prev
        prev = current
        current = next_node
    return prev
def remove_nodes(head):
    head = reverse_linked_list(head)

    max_val = float('-inf')
    dummy = ListNode(0)
    current = head
    new_list_tail = dummy

    while current:
        if current.val >= max_val:
            max_val = current.val
            new_list_tail.next = current
            new_list_tail = new_list_tail.next
        current = current.next

    new_list_tail.next = None

    head = reverse_linked_list(dummy.next)

    return head

def create_linked_list(lst):
    if not lst:
        return None
    head = ListNode(lst[0])
    current = head
    for val in lst[1:]:
        current.next = ListNode(val)
        current = current.next
    return head

def print_linked_list(head):
    current = head
    while current:
        print(current.val, end=" -> ")
        current = current.next
    print("None")
head = create_linked_list([5, 2, 13, 3, 8])
print("Original list:")
```

```
print_linked_list(head)

modified_head = remove_nodes(head)
print("Modified list:")
print_linked_list(modified_head)
```

Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\PROGRAM 69.PY"
4

Process finished with exit code 0
```

Time complexity:
O(n)

10.Count Subarrays With Median K

You are given an array nums of size n consisting of distinct integers from 1 to n and a positive integer k. Return the number of non-empty subarrays in nums that have a median equal to k. Note:
● The median of an array is the middle element after sorting the array in ascending order. If the array is of even length, the median is the left middle element. ○ For example, the median of [2,3,1,4] is 2, and the median of [8,4,3,5,1] is 4. ● A subarray is a contiguous part of an array. Example 1:
Input: nums = [3,2,1,4,5], k = 4
Output: 3
Program:

```
def countSubarraysWithMedianK(nums, k):
    n = len(nums)
    transformed = []
    k_index = -1
    for i in range(n):
        if nums[i] < k:
            transformed.append(-1)
        elif nums[i] == k:
            transformed.append(0)
            k_index = i
        else:
            transformed.append(1)
    prefix_count = {0: 1}
    balance = 0
    result = 0
    for i in range(n):
        balance += transformed[i]
        if i >= k_index:
            result += prefix_count.get(balance, 0)
            result += prefix_count.get(balance - 1, 0)
        if i < k_index:
            prefix_count[balance] = prefix_count.get(balance, 0) + 1
    return result
nums = [3, 2, 1, 4, 5]
k = 4
print(countSubarraysWithMedianK(nums, k))
```

Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\PROGRAM 69.PY"
3

Process finished with exit code 0
```

Time complexity;O(n)