

1. 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 i th 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.

```

Assignment 5.1.py - C:/Users/balas/OneDrive/Documents/Assignment 5.1.py (3.12.1)
File Edit Format Run Options Window Help
def xorfun(n, edges, values):
    tree = defaultdict(list)
    for a, b in edges:
        tree[a].append(b)
        tree[b].append(a)
    subtree_sums = [0] * n
    visited = [False] * n
    def sumofsub(node):
        visited[node] = True
        subtree_sum = values[node]
        for neighbor in tree[node]:
            if not visited[neighbor]:
                subtree_sum += sumofsub(neighbor)
        subtree_sums[node] = subtree_sum
        return subtree_sum
    sumofsub(0)
    max_xor = 0
    visited = [False] * n
    def dfs(node):
        nonlocal max_xor
        visited[node] = True
        curr = subtree_sums[node]
        xor_candidates = []
        for neighbor in tree[node]:
            if not visited[neighbor]:
                dfs(neighbor)
                xor_candidates.append(subtree_sums[neighbor])
        for xor_candidate in xor_candidates:
            max_xor = max(max_xor, curr ^ xor_candidate)
        return curr
    dfs(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(xorfun(n, edges, values))

```

```

IDLE Shell 3.12.1
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/balas/OneDrive/Documents/Assignment 5.1.py
22
>>>

```

2. Form a Chemical Bond

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

```

main.py
1 import pandas as pd
2 data = {
3     'symbol': ['He', 'Na', 'Ca', 'La', 'Cl', 'O', 'N'],
4     'type': ['Noble', 'Metal', 'Metal', 'Metal', 'Nonmetal', 'Nonmetal', 'Nonmetal'],
5     'electrons': [0, 1, 2, 3, 1, 2, 3]
6 }
7 elements = pd.DataFrame(data)
8 metals = elements[elements['type'] == 'Metal']
9 nonmetals = elements[elements['type'] == 'Nonmetal']
10 bonds = pd.merge(metals, nonmetals, how='cross', suffixes=('_metal', '_nonmetal'))
11 result = bonds[['symbol_metal', 'symbol_nonmetal']]
12 print(result)
13

```

```

symbol_metal symbol_nonmetal
0      Na      Cl
1      Na      O
2      Na      N
3      Ca      Cl
4      Ca      O
5      Ca      N
6      La      Cl
7      La      O
8      La      N

```

=== Code Execution Successful ===

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.

```

Assignment 5.3.py - C:\Users\balas\OneDrive\Documents\Assignment 5.3.py (3.12.1)
File Edit Format Run Options Window Help
def mincuts(n):
    if n == 1:
        return 0
    elif n == 2:
        return 1
    elif n==3:
        return 1
    else:
        return n//2
print(mincuts(4))

IDLE Shell 3.12.1
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bi
AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\balas\OneDrive\Documents\Assignment 5.3.py
2
>>>
>>>

```

4. 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 i th character is 'Y', it means that customers come at the i th hour
- whereas 'N' indicates that no customers come at the i th hour. If the shop closes at the j th hour ($0 \leq j \leq n$), 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

```

File Edit Format Run Options Window Help
def best(cust: str) -> int:
    n = len(cust)
    open_penalty = [0] * (n + 1)
    close_penalty = [0] * (n + 1)
    for i in range(1, n + 1):
        open_penalty[i] = open_penalty[i - 1] + (1 if cust[i - 1] == 'N' else 0)
    for i in range(n - 1, -1, -1):
        close_penalty[i] = close_penalty[i + 1] + (1 if cust[i] == 'Y' else 0)

    min_penalty = float('inf')
    best_hour = 0
    for j in range(n + 1):
        total_penalty = open_penalty[j] + close_penalty[j]
        if total_penalty < min_penalty:
            min_penalty = total_penalty
            best_hour = j

    return best_hour
print(best("YYNY"))
print(best("NNNN"))
print(best("YYYY"))

IDLE Shell 3.12.1
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bi
AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\balas\OneDrive\Documents\Assignment 5.3.py
2
0
4
>>>
>>>

```

5. . 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 i th character is 'Y', it means that customers come at the i th hour
- whereas 'N' indicates that no customers come at the i th hour. If the shop closes at the j th hour ($0 \leq j \leq n$), 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

```

File Edit Format Run Options Window Help
def best(cust: str) -> int:
    n = len(cust)
    open_penalty = [0] * (n + 1)
    close_penalty = [0] * (n + 1)
    for i in range(1, n + 1):
        open_penalty[i] = open_penalty[i - 1] + (1 if cust[i - 1] == 'N' else 0)
    for i in range(n - 1, -1, -1):
        close_penalty[i] = close_penalty[i + 1] + (1 if cust[i] == 'Y' else 0)

    min_penalty = float('inf')
    best_hour = 0
    for j in range(n + 1):
        total_penalty = open_penalty[j] + close_penalty[j]
        if total_penalty < min_penalty:
            min_penalty = total_penalty
            best_hour = j

    return best_hour
print(best("YYNY"))
print(best("NNNN"))
print(best("YYYY"))

IDLE Shell 3.12.1
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bi
AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\balas\OneDrive\Documents\Assignment 5.3.py
2
0
4
>>>
>>>

```

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 $10^9 + 7$.

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.

```

File Edit Format Run Options Window Help
def count(s):
    MOD = 10**9 + 7
    n = len(s)
    count = 0

    for i in range(n):
        for j in range(i + 1, n):
            for k in range(j + 1, n):
                for l in range(k + 1, n):
                    for m in range(l + 1, n):
                        if s[i] == s[m] and s[j] == s[l]:
                            count = (count + 1) % MOD

    return count
s = "103301"
print(count(s))

```

```

File Edit Shell Debug Options Window Help
Python 3.12.1 (tags/v3.12.1:230 AMD64) on win32
Type "help", "copyright", "cred
>>>
= RESTART: C:/Users/balas/OneDr
2
>>>

```

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.

```

Assignment 5.6.py - C:/Users/balas/OneDrive/Documents/Assignment 5.6.py (3.12.1)
File Edit Format Run Options Window Help
def findpivot(n):
    total_sum = n * (n + 1) // 2

    left_sum = 0
    for x in range(1, n + 1):
        left_sum += x
        right_sum = total_sum - left_sum

        if left_sum == right_sum:
            return x

    return -1
n = 8
print(findpivot(n))

```

```

IDLE Shell 3.12.1
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/balas/OneDrive/Documents/Assignment 5.6.py =
-1
>>>

```

8. Append Characters to String to Make Subsequence 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.

```

File Edit Format Run Options Window Help
def char(s, t):
    m, n = len(s), len(t)
    dp = [[0] * (n + 1) for _ in range(m + 1)]
    for i in range(1, m + 1):
        for j in range(1, n + 1):
            if s[i - 1] == t[j - 1]:
                dp[i][j] = dp[i - 1][j - 1] + 1
            else:
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])
    append_count = n - dp[m][n]

    return append_count
s = "abcde"
t = "a"
print(char(s, t))

```

```

File Edit Shell Debug Options Window Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
- RESTART: C:/Users/balas/OneDrive/Documents/Assignment 5.8.py
0
>>>

```

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.

The screenshot shows an IDE with two windows. The left window, titled 'Assignment 5.9.py - C:/Users/balas/OneDrive/Documents/Assignment 5.9.py (3.12.1)', contains the following Python code:

```

class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def removeNodes(head):
    if not head:
        return None

    dummy = ListNode(-1)
    dummy.next = head
    current = dummy
    max_val = float('-inf')

    while current.next:
        if current.next.val < max_val:
            current.next = current.next.next
        else:
            max_val = current.next.val
            current = current.next

    return dummy.next

head = ListNode(8)
head.next = ListNode(3)
head.next.next = ListNode(13)
head.next.next.next = ListNode(2)
head.next.next.next.next = ListNode(5)

result = removeNodes(head)
while result:
    print(result.val, end=" ")
    result = result.next

```

The right window, titled 'IDLE Shell 3.12.1', shows the execution output:

```

Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.
AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more informa
>>>
= RESTART: C:/Users/balas/OneDrive/Documents/Assignment 5.9.py
8 13
>>>
>>>

```

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.

The screenshot shows an IDE with two windows. The left window, titled 'Assignment 5.10.py - C:/Users/balas/OneDrive/Documents/Assignment 5.10.py (3.12.1)', contains the following Python code:

```

def count(nums, k):
    def median(arr):
        n = len(arr)
        sorted_arr = sorted(arr)
        if n % 2 == 0:
            return sorted_arr[n // 2 - 1]
        else:
            return sorted_arr[n // 2]

    count = 0
    n = len(nums)

    for i in range(n):
        for j in range(i, n):
            subarray = nums[i:j + 1]
            if median(subarray) == k:
                count += 1

    return count

nums = [3, 2, 1, 4, 5]
k = 4
print(count(nums, k))

```

The right window, titled 'IDLE Shell 3.12.1', shows the execution output:

```

Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25)
AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more
>>>
===== RESTART: C:/Users/balas/OneDrive/Documents/Assign
3
>>>

```