1.Maximum XOR of Two Non-Overlapping Subtrees

**Code with output:**

def m (n, edges, values):

from collections import defaultdict

tree = defaultdict(list)

for u, v in edges:

tree[u].append(v)

tree[v].append(u)

subtree\_sum = [0] \* n

visited = [False] \* n

def dfs(node):

visited[node] = True

total = values[node]

for i in tree[node]:

if not visited[i]:

total += dfs(i)

subtree\_sum[node] = total

return total

dfs(0)

max\_xor = 0

def find\_max\_xor(node):

nonlocal max\_xor

visited[node] = True

for i in tree[node]:

if not visited[i]:

max\_xor = max(max\_xor, subtree\_sum[0] ^ subtree\_sum[i])

find\_max\_xor(i)

visited = [False] \* n

find\_max\_xor(0)

return max\_xor

print(m (6, [[0, 1], [0, 2], [1, 3], [1, 4], [2, 5]], [2, 8, 3, 6, 2, 5])) # Output: 24

print(m (3, [[0, 1], [1, 2]], [4, 6, 1])) # Output: 0

2.Form a Chemical Bond

**Code with output:**

SELECT

m.symbol AS metal,

n.symbol AS nonmetal

FROM

Elements m

JOIN

Elements n

ON

m.type = 'Metal'

AND n.type = 'Nonmetal';

3. Minimum Cuts to Divide a Circle

**Code with output:**

def m(n):

if n == 1:

return 0

elif n % 2 == 0:

return n // 2

else:

return n

print(m(4)) # Output: 2

print(m(3)) # Output: 3

4. Difference Between Ones and Zeros in Row and Column

**Code with output:**

def b (customers):

n = len(customers)

total\_y = customers.count('Y')

total\_n = customers.count('N')

min\_penalty = total\_y # Penalty if we close at hour 0

min\_hour = 0

current\_penalty = total\_y

for i in range(n):

if customers[i] == 'Y':

current\_penalty -= 1

else: # customers[i] == 'N'

current\_penalty += 1

if current\_penalty < min\_penalty:

min\_penalty = current\_penalty

min\_hour = i + 1

return min\_hour

print(b ("YYNY")) # Output: 2

print(b ("NNNNN")) # Output: 0

print(b ("YYYY")) # Output: 4

5. Minimum Penalty for a Shop

**Code with output:**

def bestClosingTime(customers):

n = len(customers)

current\_penalty = customers.count('Y')

min\_penalty = current\_penalty

min\_hour = 0

for i in range(n):

if customers[i] == 'Y':

current\_penalty -= 1

else:

current\_penalty += 1

if current\_penalty < min\_penalty:

min\_penalty = current\_penalty

min\_hour = i + 1

return min\_hour

print(bestClosingTime("YYNY")) # Output: 2

print(bestClosingTime("NNNNN")) # Output: 0

print(bestClosingTime("YYYY")) # Output: 4

**6. Count Palindromic Subsequence:**

**Code with output:**

MOD = 10\*\*9 + 7

def counts):

n = len(s)

if n < 5:

return 0

def p(subseq):

return subseq == subseq[::-1]

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):

subseq = s[i] + s[j] + s[k] + s[l] + s[m]

if p(subseq):

count = (count + 1) % MOD

return count

print(count ("103301")) # Output: 2

print(count("0000000")) # Output: 21

print(count ("9999900000")) # Output: 2

7. Find the Pivot Integer

**Code with output:**

def pn):

s= n \* (n + 1) // 2

for x in range(1, n + 1):

l= x \* (x + 1) // 2

r = s - l+ x

if l == r:

return x

return -1

print(p(8)) # Output: 6

print(p(1)) # Output: 1

print(p(4)) # Output: -1

8. Append Characters to String to Make Subsequence

**Code with output:**

def append(s, t):

s\_len, t\_len = len(s), len(t)

i, j = 0, 0

while i < s\_len and j < t\_len:

if s[i] == t[j]:

j += 1

i += 1

return t\_len - j

print(append("coaching", "coding")) # Output: 4

print(append("abcde", "a")) # Output: 0

print(append("z", "abcde")) # Output: 5

9. Remove Nodes From Linked List

**Code with output:**

class ListNode:

def \_\_init\_\_(self, val=0, next=None):

self.val = val

self.next = next

def remove(head):

def reverse(head):

prev = None

while head:

next\_node = head.next

head.next = prev

prev = head

head = next\_node

return prev

head = reverse(head)

stack = []

while head:

while stack and stack[-1] < head.val:

stack.pop()

stack.append(head.val)

head = head.next

new\_head = ListNode(stack.pop()) if stack else None

current = new\_head

while stack:

current.next = ListNode(stack.pop())

current = current.next

return new\_head

def l (lst):

dummy = ListNode(0)

current = dummy

for value in lst:

current.next = ListNode(value)

current = current.next

return dummy.next

def l (head):

lst = []

while head:

lst.append(head.val)

head = head.next

return lst

head = l([5, 2, 13, 3, 8])

new\_head = removeNodes(head)

print(l (new\_head)) # Output: [13, 8]

head =l([1, 1, 1, 1])

new\_head = removeNodes(head)

print(l (new\_head)) # Output: [1, 1, 1, 1]

10. Count Subarrays With Median K

**Code with output:**

def count(nums, k):

k\_index = nums.index(k)

prefix\_sums = {0: 1}

balance = 0

count = 0

for i in range(k\_index, -1, -1):

balance += 1 if nums[i] > k else -1 if nums[i] < k else 0

prefix\_sums[balance] = prefix\_sums.get(balance, 0) + 1

balance = 0

for i in range(k\_index, len(nums)):

balance += 1 if nums[i] > k else -1 if nums[i] < k else 0

count += prefix\_sums.get(-balance, 0) + prefix\_sums.get(-balance + 1, 0)

return count

print(count ([3, 2, 1, 4, 5], 4)) # Output: 3

print(count ([2, 3, 1], 3)) # Output: 1