1.Height of Binary Tree After Subtree Removal Queries

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

File Edit Format Run Options Window Help

```python
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
def calculate_heights_and_depths(node, height_map, depth_map, depth=0):
    if not node:
        return 0
    depth_map[node.val] = depth
    left_height = calculate_heights_and_depths(node.left, height_map, depth_r
    right_height = calculate_heights_and_depths(node.right, height_map, depth
    current_height = max(left_height, right_height) + 1
    height_map[node.val] = current_height
    return current_height
def calculate_tree_height_excluding_node(root, node_val, height_map, depth_ma
    if not root:
        return 0
    if root.val == node_val:
        return 0
    left_height = height_map[root.left.val] if root.left and root.left.val !=
    right_height = height_map[root.right.val] if root.right and root.right.va
    return max(left_height, right_height) + 1
def tree_heights_after_queries(root, queries):
    if not root:
        return []
    height_map = {}
    depth_map = {}
    initial_height = calculate_heights_and_depths(root, height_map, depth_map
    results = []
    for q in queries:
        if q not in height_map:
            results.append(initial_height - 1)
        else:
            new_height = max(initial_height - height_map[q], depth_map[q] - 1
            results.append(new_height)
    return results
root = TreeNode(5)
root.left = TreeNode(8)
root.right = TreeNode(9)
root.left.left = TreeNode(2)
root.left.right = TreeNode(1)
```

Ln: 36 Col: 18

IDLE Shell 3.12.1

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\sleva\Desktop\Duplicate.py
[3, 2, 3, 1]
>>>
```

Ln: 6 Col: 0

2. Sort Array by Moving Items to Empty Space

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

File Edit Format Run Options Window Help

```python
def min_operations_to_sort(nums):
    n = len(nums)
    visited = [False] * n
    pos = nums.index(0)  # Find the position of the empty space
    moves = 0

    for i in range(n):
        if not visited[i]:
            cycle_length = 0
            x = i

            while not visited[x]:
                visited[x] = True
                x = nums[x]
                cycle_length += 1

            if cycle_length > 0:
                moves += (cycle_length - 1)

    return moves

# Example usage
nums = [4, 2, 0, 3, 1]
print(min_operations_to_sort(nums))  # Output: 3
```

IDLE Shell 3.12.1

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\sleva\Desktop\Duplicate.py
3
>>>
```

3. Apply Operations to an Array

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

File Edit Format Run Options Window Help

```python
def process_array(nums):
    n = len(nums)

    # Step 1: Apply the sequential operations
    for i in range(n - 1):
        if nums[i] == nums[i + 1]:
            nums[i] *= 2
            nums[i + 1] = 0

    # Step 2: Shift all zeros to the end
    result = []
    zero_count = 0

    for num in nums:
        if num != 0:
            result.append(num)
        else:
            zero_count += 1

    # Append the zeros to the result
    result.extend([0] * zero_count)

    return result

# Example usage
nums = [1, 2, 2, 1, 1, 0]
print(process_array(nums))  # Output: [1, 4, 2, 0, 0, 0]
```

IDLE Shell 3.12.1

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\sleva\Desktop\Duplicate.py
[1, 4, 2, 0, 0, 0]
>>>
```

4. Maximum Sum of Distinct Subarrays With Length K

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def max_subarray_sum(nums, k):
    n = len(nums)
    if n < k:
        return 0

    # Initialize variables
    current_sum = 0
    max_sum = 0
    start = 0
    num_set = set()

    for end in range(n):
        # If we encounter a duplicate, move the start pointer
        while nums[end] in num_set:
            num_set.remove(nums[start])
            current_sum -= nums[start]
            start += 1

        # Add the current element to the set and update the current sum
        num_set.add(nums[end])
        current_sum += nums[end]

        # Check if we have a valid window of size k
        if end - start + 1 == k:
            max_sum = max(max_sum, current_sum)

            # Slide the window
            num_set.remove(nums[start])
            current_sum -= nums[start]
            start += 1

    return max_sum
nums = [1, 5, 4, 2, 9, 9, 9]
k = 3
print("Sum :",max_subarray_sum(nums, k))
```

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\sleva\Desktop\Duplicate.py
Sum : 15
>>>
```

5. Total Cost to Hire K Workers

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
import heapq

def total_cost_to_hire_workers(costs, k, candidates):
    n = len(costs)
    left_heap = []
    right_heap = []
    for i in range(candidates):
        if i < n:
            heapq.heappush(left_heap, (costs[i], i))
        if n - 1 - i >= 0:
            heapq.heappush(right_heap, (costs[n - 1 - i], n - 1 - i))
    total_cost = 0
    hired = set()
    for _ in range(k):
        while left_heap and left_heap[0][1] in hired:
            heapq.heappop(left_heap)
        while right_heap and right_heap[0][1] in hired:
            heapq.heappop(right_heap)

        if left_heap and (not right_heap or left_heap[0] <= right_heap[0]):
            cost, index = heapq.heappop(left_heap)
        else:
            cost, index = heapq.heappop(right_heap)

        total_cost += cost
        hired.add(index)

        if index < n - 1 and index + 1 not in hired:
            heapq.heappush(left_heap, (costs[index + 1], index + 1))
        if index > 0 and index - 1 not in hired:
            heapq.heappush(right_heap, (costs[index - 1], index - 1))

    return total_cost

# Example usage
costs = [17, 12, 10, 2, 7, 2, 11, 20, 8]
k = 3
candidates = 4
print(total_cost_to_hire_workers(costs, k, candidates))  # Output: 11
```

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\sleva\Desktop\Duplicate.py
11
>>>
```

6. Minimum Total Distance Traveled

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def min_total_distance(robot, factory):
    robot.sort()
    factory.sort()

    n = len(robot)
    m = len(factory)
    total_distance = 0

    # Pointers for robots and factories
    robot_idx = 0
    factory_idx = 0

    while robot_idx < n:
        pos_robot = robot[robot_idx]
        pos_factory, limit = factory[factory_idx]

        if limit > 0:
            # Calculate distance from current robot to current factory
            distance = abs(pos_robot - pos_factory)
            total_distance += distance

            # Reduce the limit of the factory by 1 since it repairs one robot
            factory[factory_idx][1] -= 1

        # Move to the next robot
        robot_idx += 1

        # Move to the next factory if the current one has reached its limit
        if factory_idx + 1 < m and factory[factory_idx][1] == 0:
            factory_idx += 1

    return total_distance

# Example usage
robot = [0, 4, 6]
factory = [[2, 2], [6, 2]]
print(min_total_distance(robot, factory))  # Output: 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 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
4
>>>
```

7. Minimum Subarrays in a Valid Split

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

File Edit Format Run Options Window Help

```python
import math

def findMinSubarrays(nums):
    n = len(nums)

    if n == 0:
        return -1

    # Precompute gcd values
    gcd = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(i, n):
            gcd[i][j] = math.gcd(nums[i], nums[j])
            gcd[j][i] = gcd[i][j]

    # Initialize dp array
    dp = [float('inf')] * n
    dp[0] = 1

    # Fill dp array
    for i in range(1, n):
        for j in range(i):
            if gcd[0][j] > 1 and gcd[j][i] > 1:
                dp[i] = min(dp[i], dp[j] + 1)

    # If dp[n-1] is still inf, no valid splitting found
    if dp[n-1] == float('inf'):
        return -1
    else:
        return dp[n-1]

# Example usage:
nums = [2, 3, 6, 9]
print(findMinSubarrays(nums))
```

IDLE Shell 3.12.1

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\sleva\Desktop\Duplicate.py
3
>>>
```

8. Number of Distinct Averages

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

File Edit Format Run Options Window Help

```python
def countDistinctAverages(nums):
    nums.sort()
    n = len(nums)
    left, right = 0, n - 1
    distinct_averages = set()

    while left < right:
        avg = (nums[left] + nums[right]) / 2.0
        distinct_averages.add(avg)
        left += 1
        right -= 1

    return len(distinct_averages)

# Example usage:
nums = [4, 1, 4, 0, 3, 5]
print(countDistinctAverages(nums))  # Output: 2
```

IDLE Shell 3.12.1

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\sleva\Desktop\Duplicate.py
2
>>>
```

9. Count Ways To Build Good Strings

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

File Edit Format Run Options Window Help

```python
def countGoodStrings(zero, one, low, high):
    MOD = 10**9 + 7

    # Initialize dp array
    dp = [0] * (high + 1)
    dp[0] = 1  # There's 1 way to make an empty string

    # Build dp array up to high
    for i in range(1, high + 1):
        if i >= 1:
            dp[i] += dp[i - 1]  # Append '0' to dp[i-1] length strings
        if i >= 2:
            dp[i] += dp[i - 2]  # Append '1' to dp[i-2] length strings
        dp[i] %= MOD

    # Sum up dp values from low to high
    result = 0
    for i in range(low, high + 1):
        result += dp[i]
        result %= MOD

    return result

# Example usage:
zero = 1
one = 1
low = 2
high = 4
print(countGoodStrings(zero, one, low, high))
```

IDLE Shell 3.12.1

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\sleva\Desktop\Duplicate.py
10
>>>
```

10. Most Profitable Path in a Tree

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

File Edit Format Run Options Window Help

```python
    for a, b in edges:
        tree[a].append(b)
        tree[b].append(a)

    # dp[node] will store the maximum net income Alice can achieve starting f
    dp = [-float('inf')] * n
    sum = [0] * n  # sum[node] will store the total price/reward at `node` ar

    def dfs(node, parent):
        nonlocal dp, sum, tree, amount

        # Calculate sum[node] including the node itself
        sum[node] = amount[node]
        for neighbor in tree[node]:
            if neighbor == parent:
                continue
            dfs(neighbor, node)
            sum[node] += sum[neighbor]

        # Calculate dp[node]
        dp[node] = amount[node]
        max_child_income = -float('inf')
        for neighbor in tree[node]:
            if neighbor == parent:
                continue
            max_child_income = max(max_child_income, dp[neighbor])

        if max_child_income != -float('inf'):
            dp[node] = max(dp[node], max_child_income - amount[node] / 2)

    # Start DFS from root node 0
    dfs(0, -1)

    return dp[0]

# Example usage:
n = 7
edges = [[0,1],[0,2],[1,4],[1,5],[2,3],[2,6]]
amount = [-10, 5, 6, -8, -7, 9, -8]
print(maximumNetIncome(n, edges, amount))
```

IDLE Shell 3.12.1

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\sleva\Desktop\Duplicate.py
11.5
>>>
```