Sorting Assignment

Problem 1.Given an array of n numbers, give an algorithm which gives the element appearing m
number of times?

Problem 2 : We are given a list of n-1 integers and these integers are in the range of 1 to
duplicates in the list. One of the integers is missing in the list. Give an algorithm to fin
[1,2,4,6,3,7,8] 5 is the missing num.

Problem 3 : Given an array of n positive numbers. All numbers occurs even number of times ex
occurs odd number of times. Find that number in O(n) time and O(1) space. Ex: [1,2,3,2,3,1,3
times.

Problem 4 : Given an array of n elements. Find two elements in the array such that their sum
element K.

Problem 5 : Given an array of both positive and negative numbers, find two numbers such that
closest to 0. Ex: [1 ,60 ,-10, 70, -80,85]. Ans : -80,85.

Problem 6 : Given an array of n elements . Find three elements such that their sum is equal
number.

Problem 7 : Given an array of n elements . Find three elements i, j, k in the array such tha
i * i + j * j = k*k.

Problem 8 : An element is a majority if it appears more than n/2 times. Give an algorithm ta
element as argument and identifies a majority (if it exists).

Problem 9 : Given n × n matrix, and in each row all 1's are followed by 0's. Find the row wi
number of 0's.

Problem 10 : Sort an array of 0's, 1's and 2's [or R's, G's and B's]: Given an array A[] con
2's, give an algorithm for sorting A[].The algorithm should put all 0's first, then all 1's
end. Example Input = {0,1,1,0,1,2,1,2,0,0,0,1}, Output = {0,0,0,0,0,1,1,1,1,1,2,2}

```
''' Problem 1.Given an array of n numbers, give an algorithm which gives the element appear
number of times?'''
def most_freq_element(arr):
    freq={}
    max_count=0
    max_element =None

    for num in arr:
        freq[num] =freq.get(num,0) +1
        if freq[num]>max_count:
            max_count =freq[num]
            max_element =num
```

```
    return max_element
arr = [1, 3, 2, 3, 4, 3, 5, 3, 2, 2, 2]
print(most_freq_element(arr))
```

```
3
```

```
'''Problem 2 : We are given a list of n-1 integers and these integers are in the range of 1
duplicates in the list. One of the integers is missing in the list. Give an algorithm to fi
[1,2,4,6,3,7,8] 5 is the missing num.'''
def missing_number(arr,n):
    total_sum=n*(n+1)//2
    actual_sum=0
    for num in arr:
        actual_sum +=num
    return total_sum - actual_sum
arr = [1, 2, 4, 6, 3, 7, 8]
n = 8
print(missing_number(arr, n))
```

```
5
```

```
'''Problem 3 : Given an array of n positive numbers. All numbers occurs even number of times
occurs odd number of times. Find that number in O(n) time and O(1) space. Ex: [1,2,3,2,3,1,3
times.'''
def find_odd_occurence(arr):
    result=0
    for num in arr:
        result ^=num
    return result
arr=[1,2,3,2,3,1,3]
print(find_odd_occurence(arr))
```

```
3
```

```
'''
Problem 4 : Given an array of n elements. Find two elements in the array such that their sum
element K.'''
def two_sum(arr,k):
    seen={}
    for num in arr:
        if (k-num) in seen:
            return num,k-num
        seen[num]=True
    return None

arr =[1,4,7,10,2]
k=6
print(two_sum(arr,k))
```

```
(2, 4)
```

```
'''
Problem 5 : Given an array of both positive and negative numbers, find two numbers such that
closest to 0. Ex: [ 1 ,60 ,-10, 70, -80,85]. Ans : -80,85.
'''
def closest_to_zero(arr):
    arr.sort()
    left, right = 0, len(arr) - 1
    min_sum = float('inf')
    best_pair = None

    while left < right:
        current_sum = arr[left] + arr[right]
        if abs(current_sum) < abs(min_sum):
            min_sum = current_sum
            best_pair = (arr[left], arr[right])

        if current_sum < 0:
            left += 1
        else:
            right -= 1

    return best_pair

arr = [1, 60, -10, 70, -80, 85]
print(closest_to_zero(arr))  # Output: (-80, 85)
```

```
(-80, 85)
```

```
'''Problem 6 : Given an array of n elements . Find three elements such that their sum is equ
number.'''
def three_sum(arr, k):
    arr.sort()
    n = len(arr)

    for i in range(n - 2):
        left, right = i + 1, n - 1
        while left < right:
            current_sum = arr[i] + arr[left] + arr[right]
            if current_sum == k:
                return arr[i], arr[left], arr[right]
            elif current_sum < k:
                left += 1
            else:
                right -= 1
    return None
```

```
arr = [1, 4, 6, 3, 9, 2]
k = 13
print(three_sum(arr, k))  # Output: (1, 4, 8) (if exists)
```

```
(1, 3, 9)
```

```
'''Problem 7 : Given an array of n elements . Find three elements i, j, k in the array such
i * i + j * j = k*k.
'''
def find_pythagorean_triplet(arr):
    arr = [x*x for x in arr]
    arr.sort()
    n = len(arr)

    for i in range(n-1, 1, -1):
        left, right = 0, i - 1
        while left < right:
            if arr[left] + arr[right] == arr[i]:
                return int(arr[left]**0.5), int(arr[right]**0.5), int(arr[i]**0.5)
            elif arr[left] + arr[right] < arr[i]:
                left += 1
            else:
                right -= 1
    return None

arr = [3, 1, 4, 6, 5]
print(find_pythagorean_triplet(arr))  # Output: (3, 4, 5)
```

```
(3, 4, 5)
```

```
'''Problem 8 : An element is a majority if it appears more than n/2 times. Give an algorithm
element as argument and identifies a majority (if it exists).

Problem 9 : Given n × n matrix, and in each row all 1's are followed by 0's. Find the row w
number of 0's.

Problem 10 : Sort an array of 0's, 1's and 2's [or R's, G's and B's]: Given an array A[] co
2's, give an algorithm for sorting A[].The algorithm should put all 0's first, then all 1's
end. Example Input = {0,1,1,0,1,2,1,2,0,0,0,1}, Output = {0,0,0,0,0,1,1,1,1,1,2,2}'''
def majority_element(arr):
    count, candidate = 0, None
    for num in arr:
        if count == 0:
            candidate = num
        count += (1 if num == candidate else -1)

    count = 0
```

```
    for num in arr:
        if num == candidate:
            count += 1

    return candidate if count > len(arr) // 2 else None

arr = [2, 2, 1, 1, 1, 2, 2]
print(majority_element(arr))  # Output: 2
```

```
2
```

```
'''
Problem 9 : Given n × n matrix, and in each row all 1's are followed by 0's. Find the row w
number of 0's.

Problem 10 : Sort an array of 0's, 1's and 2's [or R's, G's and B's]: Given an array A[] co
2's, give an algorithm for sorting A[].The algorithm should put all 0's first, then all 1's
end. Example Input = {0,1,1,0,1,2,1,2,0,0,0,1}, Output = {0,0,0,0,0,1,1,1,1,1,2,2}'''
def row_with_max_zeros(matrix):
    max_zeros, row_index = 0, -1
    for i, row in enumerate(matrix):
        count_zeros = len(row) - sum(row)
        if count_zeros > max_zeros:
            max_zeros, row_index = count_zeros, i
    return row_index

matrix = [[1, 1, 1, 0, 0], [1, 0, 0, 0, 0], [1, 1, 0, 0, 0]]
print(row_with_max_zeros(matrix))  # Output: 1
```

```
1
```

```
'''Problem 10 : Sort an array of 0's, 1's and 2's [or R's, G's and B's]: Given an array A[]
2's, give an algorithm for sorting A[].The algorithm should put all 0's first, then all 1's
end. Example Input = {0,1,1,0,1,2,1,2,0,0,0,1}, Output = {0,0,0,0,0,1,1,1,1,1,2,2}'''
def sort_colors(arr):
    low, mid, high = 0, 0, len(arr) - 1
    while mid <= high:
        if arr[mid] == 0:
            arr[low], arr[mid] = arr[mid], arr[low]
            low += 1
            mid += 1
        elif arr[mid] == 1:
            mid += 1
        else:
            arr[mid], arr[high] = arr[high], arr[mid]
            high -= 1
    return arr
```

```
arr = [0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0, 1]
print(sort_colors(arr))  # Output: [0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2]
```

```
[0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2]
```