Coding Problems Solutions

Problem 1

Given an array of n numbers, give an algorithm which gives the element appearing maximum number of times.

Solution: You can use a hash map to count occurrences of each element.

```
def max_occurrence(arr):
    count = {}
    max_count = 0
    max_elem = None

    for num in arr:
        count[num] = count.get(num, 0) + 1
        if count[num] > max_count:
            max_count = count[num]
            max_elem = num

    return max_elem

# Example usage
arr = [1, 3, 2, 1, 2, 1]
print(max_occurrence(arr))  # Output: 1
```

Problem 2

We are given a list of n-1 integers in the range of 1 to n. One integer is missing. Find that element.

Solution: You can use the formula for the sum of the first n natural numbers.

```
def find_missing_number(arr, n):
    total = n * (n + 1) // 2
    arr_sum = sum(arr)
    return total - arr_sum

# Example usage
arr = [1, 2, 4, 6, 3, 7, 8]
n = 8
print(find_missing_number(arr, n))  # Output: 5
```

Problem 3

Given an array of n positive numbers, find the number that occurs an odd number of times.

Solution: You can use the XOR operation.

```
def find_odd_occurrence(arr):
    result = 0
    for num in arr:
        result ^= num
    return result

# Example usage
arr = [1, 2, 3, 2, 3, 1, 3]
```

```
print(find_odd_occurrence(arr))  # Output: 3
```

Problem 4

Find two elements in the array such that their sum equals a given element K.

Solution: Use a hash set to track complements.

```
def find_pair_with_sum(arr, K):
    seen = set()
    for num in arr:
        if K - num in seen:
            return (num, K - num)
        seen.add(num)
    return None

# Example usage
arr = [10, 15, 3, 7]
K = 17
print(find_pair_with_sum(arr, K))  # Output: (10, 7)
```

Problem 5

Find two numbers in the array such that their sum is closest to 0.

Solution: Sort the array and use two pointers.

```
def closest_to_zero(arr):
    arr.sort()
    left, right = 0, len(arr) - 1
    closest_sum = float('inf')
    result = (0, 0)

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

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

    return result

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

Problem 6

Find three elements such that their sum equals a given number.

Solution: Use a nested loop approach.

```
def three_sum(arr, target):
```

```
    arr.sort()
    result = []

    for i in range(len(arr) - 2):
        left, right = i + 1, len(arr) - 1
        while left < right:
            current_sum = arr[i] + arr[left] + arr[right]
            if current_sum == target:
                result.append((arr[i], arr[left], arr[right]))
                left += 1
                right -= 1
            elif current_sum < target:
                left += 1
            else:
                right -= 1

    return result

# Example usage
arr = [1, 2, -2, -1, 0, 3]
target = 0
print(three_sum(arr, target))  # Output: [(1, -1, 0), (2, -2, 0)]
```

Problem 7

Find three elements i, j, k such that $i^2 + j^2 = k^2$.

Solution: Use a hash set to store squares.

```
def find_pythagorean_triplet(arr):
    squares = {x * x for x in arr}
    for i in range(len(arr)):
        for j in range(i + 1, len(arr)):
            if (arr[i] ** 2 + arr[j] ** 2) in squares:
                return (arr[i], arr[j], (arr[i] ** 2 + arr[j] ** 2) ** 0.5)
    return None

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

Problem 8

Identify a majority element that appears more than n/2 times.

Solution: Use the Boyer-Moore Voting Algorithm.

```
def majority_element(arr):
    candidate, count = None, 0
    for num in arr:
        if count == 0:
            candidate, count = num, 1
        elif num == candidate:
            count += 1
        else:
            count -= 1

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

# Example usage
arr = [2, 2, 1, 1, 1, 2, 2]
```

```
print(majority_element(arr))  # Output: 2
```

Problem 9

Find the row with the maximum number of 0's in a binary matrix.

Solution: Iterate through each row and count zeros.

```
def row_with_max_zeros(matrix):
    max_zeros = -1
    row_index = -1

    for i in range(len(matrix)):
        count_zeros = matrix[i].count(0)
        if count_zeros > max_zeros:
            max_zeros = count_zeros
            row_index = i

    return row_index

# Example usage
matrix = [
    [0, 0, 1, 1],
    [0, 1, 1, 1],
    [0, 0, 0, 0],
    [1, 1, 1, 1]
]
print(row_with_max_zeros(matrix))  # Output: 2
```

Problem 10

Sort an array of 0's, 1's, and 2's.

Solution: Use the Dutch National Flag algorithm.

```
def sort_012(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

# Example usage
arr = [0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0, 1]
print(sort_012(arr))  # Output: [0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2]
```