CheggSolutions - Thegdp

Sub-subject: Computer Science

Topic: Array Partitioning

Given and Introduction

Given an array of integers, the task is to check if it is possible to partition the array into some number of subsequences of length `k` each, such that:

- Each element in the array occurs in exactly one subsequence.
- For each subsequence, all numbers are distinct. Elements having the same value must be in different subsequences.

Problem Constraint

If it is possible to partition the array into subsequences while satisfying the above conditions, return "Yes", else return "No".

Explanation and Approach

1. **Count Frequency**: First, count the frequency of each element in the array.
2. **Check Feasibility**: Ensure that the frequency of each element is less than or equal to (array length / `k`). This is essential because it is impossible to distribute the elements into subsequences of length `k` if an element repeats more times than the possible number of subsequences.

Detailed Step-by-Step Solution

Step 1: Initialization and Frequency Counting
Given:

```
k = 2
numbers = [1, 2, 3, 4]


from collections import Counter

def can_partition_k_length_subsequences(arr, k):
    # Counting the frequency of each element
    freq = Counter(arr)
    length = len(arr)

    # Checking if each element's frequency is <= array length divided by k
    for count in freq.values():
        if count > (length // k):
            return "No"
    return "Yes"

# Test example
k = 2
numbers = [1, 2, 3, 4]
print(can_partition_k_length_subsequences(numbers, k))
```

Explanation of Steps:

1. **Initialization and Frequency Counting**:

The `Counter` from the `collections` module is used to count the frequency of each element in the array. `freq` will be a dictionary where keys are elements and values are their respective counts.

```
# freq = {1: 1, 2: 1, 3: 1, 4: 1}
```

2. **Verification Loop:**

The code iterates through the frequency dictionary `freq`. In each iteration, it checks if the count of any element exceeds `(length of array / k)`. If any element's frequency exceeds this, it is not possible to split the array into subsequences of length `k` while satisfying the conditions, and hence the function returns "No". If no element exceeds this check, it returns "Yes".

In this test example:

```
# length = 4
# k = 2
# So, length // k = 2

# Verification:
# For element 1, count is 1, which is <= 2 (pass)
# For element 2, count is 1, which is <= 2 (pass)
# For element 3, count is 1, which is <= 2 (pass)
# For element 4, count is 1, which is <= 2 (pass)

return "Yes"
```

Final Solution

The array can be partitioned into valid subsequences for the given input. Hence, the function returns "Yes".

```
"Yes"
```