Determining Big O Complexity and Validating with Experiments

Objective:

Analyze and validate the Big O complexity of the given algorithms. Implement the code, test it for different input sizes, and observe how execution time scales.

Task Instructions

1. **Analyze the Code:**
   - For each algorithm provided, identify its Big O complexity.
   - Write a detailed explanation for your conclusion based on loops, iterations, and operations.
2. **Run the Code:**
   - Test the algorithms with different input sizes.
   - Record the execution times.
3. **Plot the Results:**
   - Create graphs showing input size vs. time taken for each algorithm.
4. **Deliverables:**
   - Your analysis of the Big O complexity.
   - Graphs showing the relationship between input size and execution time.
   - A brief summary comparing theoretical and observed results.

1. Simple Arithmetic Operation

```
def constant_operation_1(n):
    return n * 10
```

2. Dictionary Access

```
def constant_operation_2(dictionary, key):
    return dictionary[key]
```

---

3. Summing Elements in a List

```
def linear_operation_1(arr):
    total = 0
    for num in arr:
        total += num
    return total
```

4. **Counting Even Numbers**

```
def linear_operation_2(arr):
    count = 0
    for num in arr:
        if num % 2 == 0:
            count += 1
    return count
```

5. **Copying a List**

```
def linear_operation_3(arr):
    return [x for x in arr]
```

---

6. **Nested Loops for Pairwise Addition**

```
def quadratic_operation_1(arr):
    result = 0
    for i in arr:
        for j in arr:
            result += i + j
    return result
```

7. **Matrix Multiplication (Simplified)**

```
def quadratic_operation_2(matrix):
    n = len(matrix)
    result = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            for k in range(n):
                result[i][j] += matrix[i][k] * matrix[k][j]
    return result
```

8. **Bubble Sort**

```
def quadratic_operation_3(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr
```