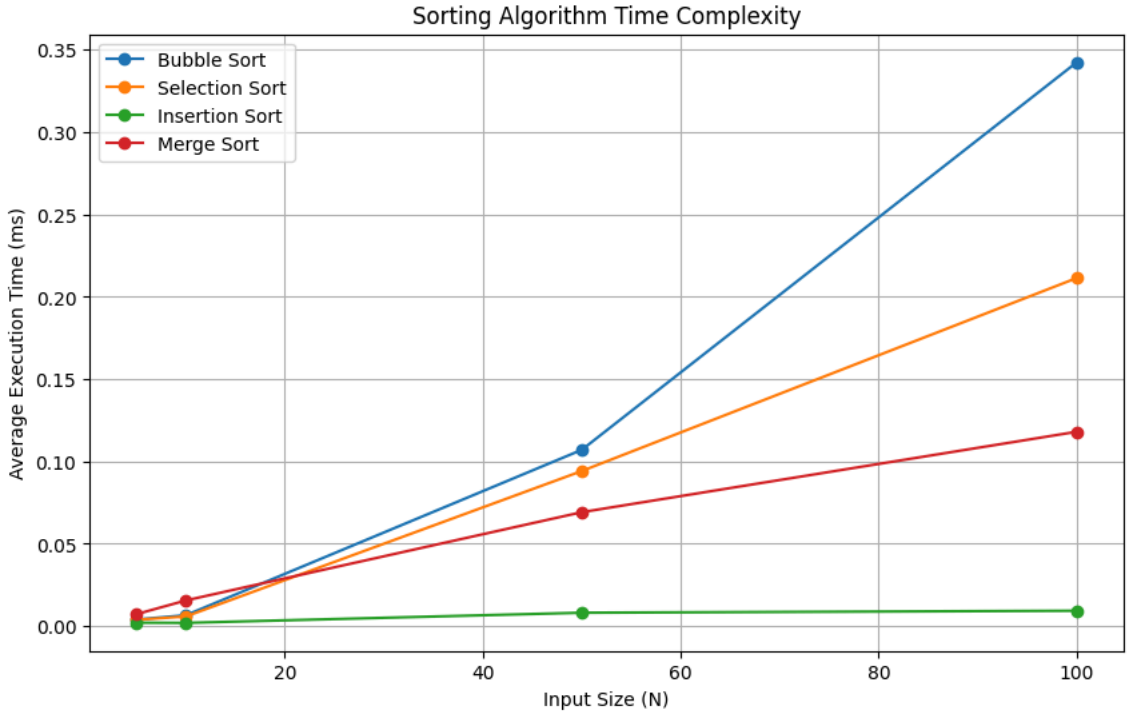
```
import time
import matplotlib.pyplot as plt
import numpy as np


# Arrays
arr1 = list(range(1, 6))
arr2 = list(range(1, 11))
arr3 = list(range(1, 51))
arr4 = list(range(1, 101))

arrays = [arr1, arr2, arr3, arr4]
array_sizes = [len(arr) for arr in arrays]

# Sorting Algorithms
def bubble_sort(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]

def selection_sort(arr):
    n = len(arr)
    for i in range(n):
        min_idx = i
        for j in range(i+1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]

def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = key

def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr)//2
        L = arr[:mid]
        R = arr[mid:]
        merge_sort(L)
        merge_sort(R)
        i = j = k = 0
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i += 1
            else:
                arr[k] = R[j]
                j += 1
            k += 1
        while i < len(L):
            arr[k] = L[i]
            i += 1
            k += 1
        while j < len(R):
            arr[k] = R[j]
            j += 1
            k += 1

# Measure Time Function
def measure_average_time(sort_func, arr, runs=5):
    times = []
    for _ in range(runs):
        arr_copy = arr.copy()
        start = time.perf_counter()
        sort_func(arr_copy)
        end = time.perf_counter()
        times.append((end - start) * 1000)  # convert to milliseconds
    return np.mean(times)

```

```
# Collect average times
times_bubble = []
times_selection = []
times_insertion = []
times_merge = []

for arr in arrays:
    times_bubble.append(measure_average_time(bubble_sort, arr))
    times_selection.append(measure_average_time(selection_sort, arr))
    times_insertion.append(measure_average_time(insertion_sort, arr))
    times_merge.append(measure_average_time(merge_sort, arr))

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(array_sizes, times_bubble, marker='o', label='Bubble Sort')
plt.plot(array_sizes, times_selection, marker='o', label='Selection Sort')
plt.plot(array_sizes, times_insertion, marker='o', label='Insertion Sort')
plt.plot(array_sizes, times_merge, marker='o', label='Merge Sort')
plt.xlabel('Input Size (N)')
plt.ylabel('Average Execution Time (ms)')
plt.title('Sorting Algorithm Time Complexity')
plt.legend()
plt.grid(True)
plt.show()

# Print Table
print("Input Size | Bubble Sort | Selection Sort | Insertion Sort | Merge Sort (ms)")
for i in range(len(array_sizes)):
    print(f"{array_sizes[i]:>10} | {times_bubble[i]:>11.5f} | {times_selection[i]:>14.5f} | {times_insertion[i]:>13.5f} | {times_merge[i
```




```
Input Size | Bubble Sort | Selection Sort | Insertion Sort | Merge Sort (ms)
         5 |     0.00398 |        0.00383 |        0.00206 |     0.00722
        10 |     0.00661 |        0.00584 |        0.00195 |     0.01559
        50 |     0.10707 |        0.09419 |        0.00811 |     0.06919
       100 |     0.34241 |        0.21140 |        0.00931 |     0.11804
```