

```

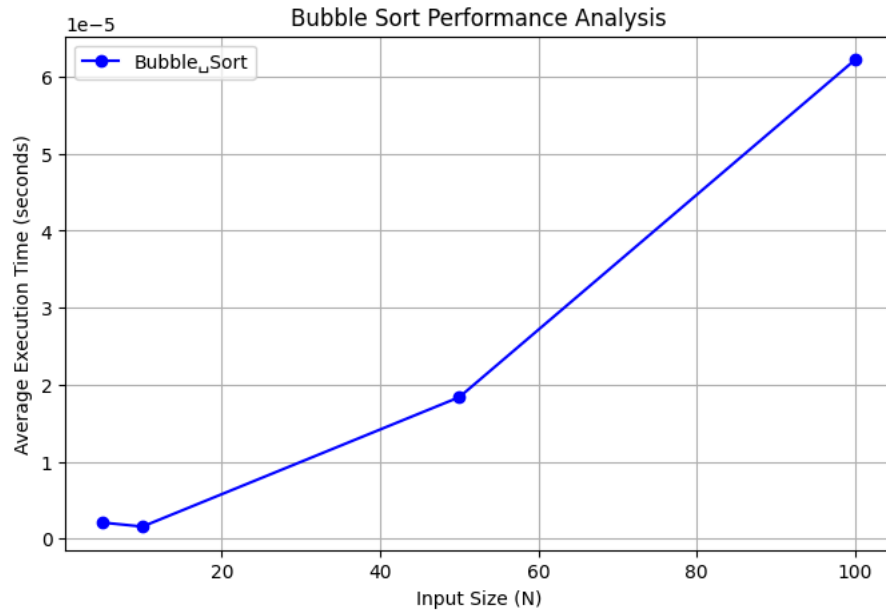
1 import time
2 import matplotlib.pyplot as plt
3
4 def bubble_sort(arr):
5     n = len(arr)
6     for i in range(n):
7         for j in range(0, n-i-1):
8             if arr[j] > arr[j+1]:
9                 arr[j], arr[j+1] = arr[j+1], arr[j]
10
11 Arr1 = list(range(1, 6))
12 Arr2 = list(range(1, 11))
13 Arr3 = list(range(1, 51))
14 Arr4 = list(range(1, 101))
15 arrays = [Arr1, Arr2, Arr3, Arr4]
16 input_sizes = [len(arr) for arr in arrays]
17
18 def measure_time(sort_function, arr):
19     runs = 5
20     total_time = 0
21     for _ in range(runs):
22         copy_arr = arr.copy()
23         start = time.perf_counter()
24         sort_function(copy_arr)
25         end = time.perf_counter()
26         total_time += (end - start)
27     return total_time / runs
28
29
30 bubble_times = []
31 for arr in arrays:
32     avg_time = measure_time(bubble_sort, arr)
33     bubble_times.append(avg_time)
34 copy_arr = arr.copy()
35 bubble_sort(copy_arr)
36 print(f"Sorted array of size {len(arr)}: {copy_arr}")
37 print("\nAverage Execution Times:")
38 for size, time_taken in zip(input_sizes, bubble_times):
39     print(f"Input size {size}: {time_taken:.8f} seconds")
40 # Plotting
41 plt.figure(figsize=(8, 5))
42 plt.plot(input_sizes, bubble_times, marker='o', color='blue', label="Bubble_Sort")
43 plt.title("Bubble Sort Performance Analysis")
44 plt.xlabel("Input Size (N)")
45 plt.ylabel("Average Execution Time (seconds)")
46 plt.legend()
47 plt.grid(True)
48 plt.show()
49

```

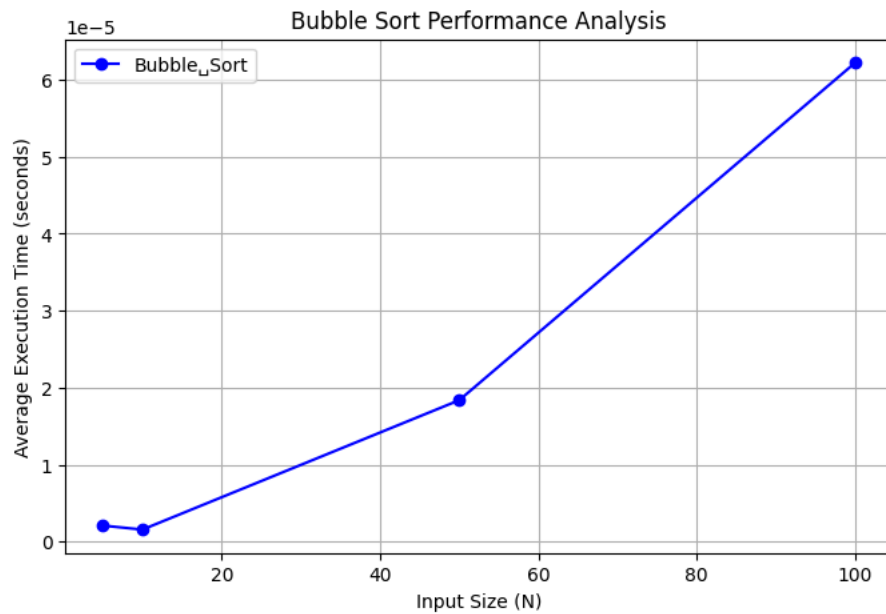
Sorted array of size 100: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,

Average Execution Times:

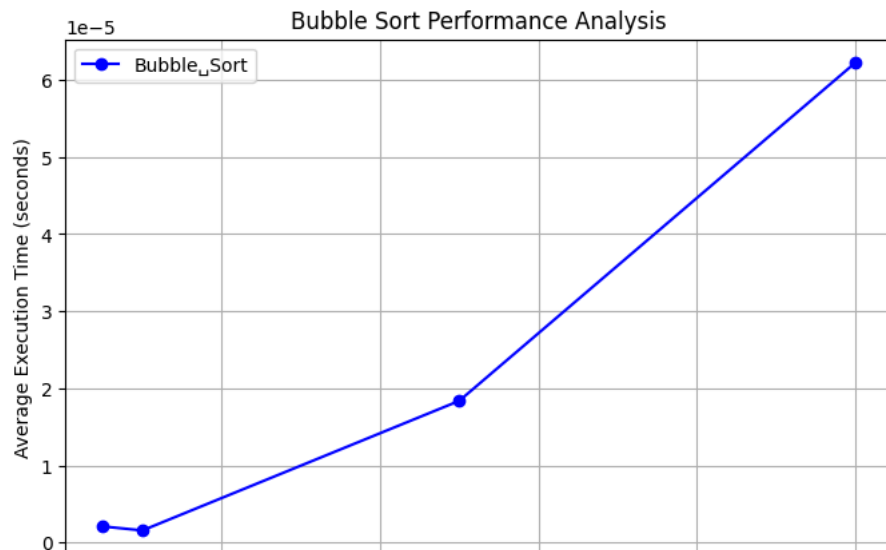
Input size 5: 0.00000208 seconds

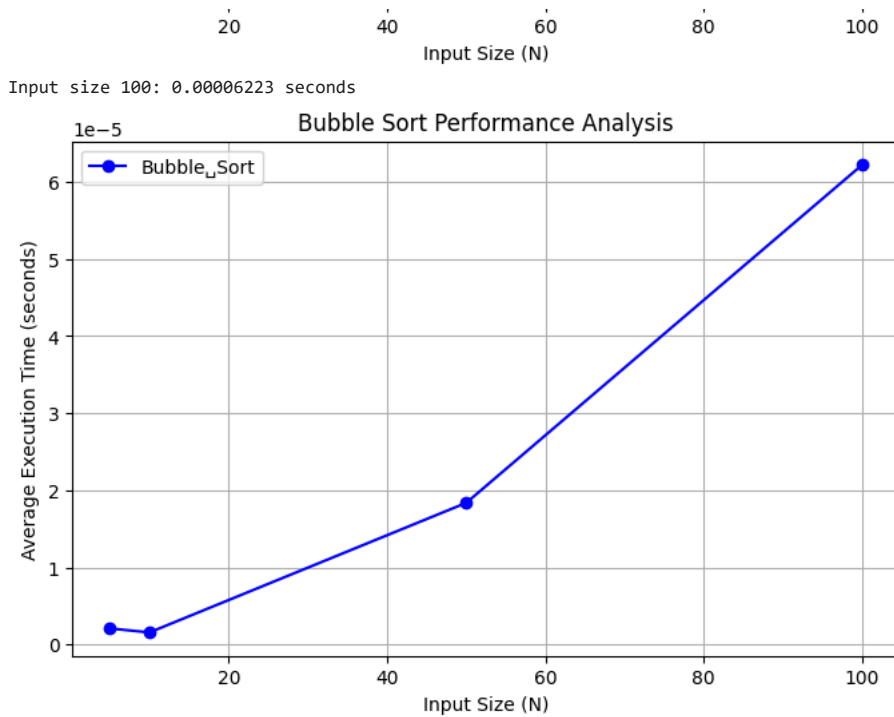


Input size 10: 0.00000157 seconds



Input size 50: 0.00001838 seconds





```

1 import time
2 import matplotlib.pyplot as plt
3
4 def selection_sort(arr):
5     n = len(arr)
6     for i in range(n):
7         min_idx = i
8         for j in range(i+1, n):
9             if arr[j] < arr[min_idx]:
10                 min_idx = j
11     arr[i], arr[min_idx] = arr[min_idx], arr[i]
12
13 Arr1 = list(range(1, 6))
14 Arr2 = list(range(1, 11))
15 Arr3 = list(range(1, 51))
16 Arr4 = list(range(1, 101))
17 arrays = [Arr1, Arr2, Arr3, Arr4]
18 input_sizes = [len(arr) for arr in arrays]
19
20 def measure_time(sort_function, arr):
21     runs = 5
22     total_time = 0
23     for _ in range(runs):
24         copy_arr = arr.copy()
25         start = time.perf_counter()
26         sort_function(copy_arr)
27         end = time.perf_counter()
28         total_time += (end - start)
29     return total_time / runs
30
31
32 selection_times = []
33 for arr in arrays:
34     avg_time = measure_time(selection_sort, arr)
35     selection_times.append(avg_time)
36     copy_arr = arr.copy()
37     selection_sort(copy_arr)
38     print(f"Sorted array of size {len(arr)}: {copy_arr}")
39     print("\nAverage Execution Times for Selection Sort:")
40 for size, time_taken in zip(input_sizes, selection_times):
41     print(f"Input size {size}: {time_taken:.8f} seconds")
42 # Plotting
43 plt.figure(figsize=(8, 5))
44 plt.plot(input_sizes, selection_times, marker='o', color='green', label="Selection Sort")
45 plt.title("Selection Sort Performance Analysis")
46 plt.xlabel("Input Size (N)")

```

```

47 plt.ylabel("Average Execution Time (seconds)")
48 plt.legend()
49 plt.grid(True)
50 plt.show()
51

```

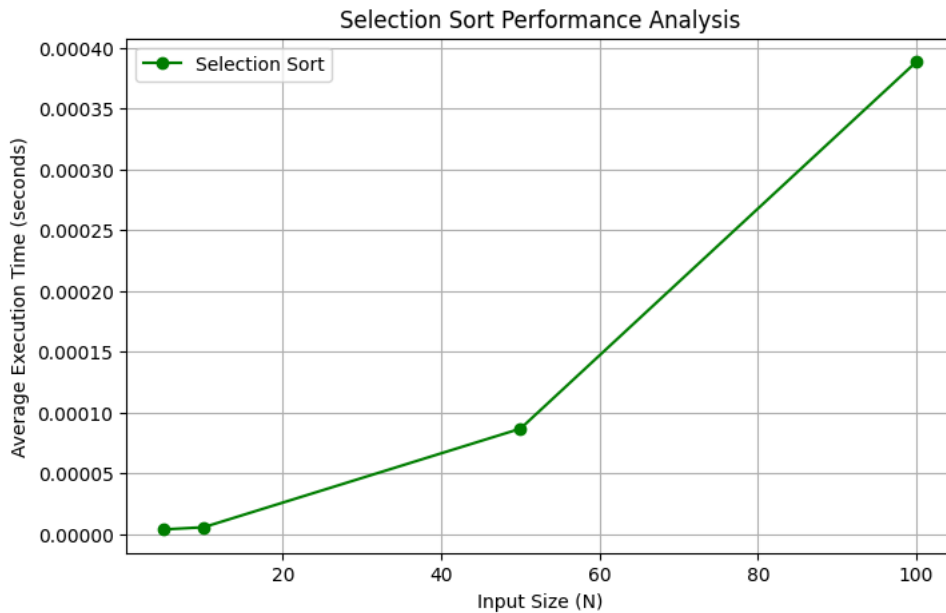
Sorted array of size 5: [1, 2, 3, 4, 5]

Average Execution Times for Selection Sort:  
Sorted array of size 10: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Average Execution Times for Selection Sort:  
Sorted array of size 50: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,

Average Execution Times for Selection Sort:  
Sorted array of size 100: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,

Average Execution Times for Selection Sort:  
Input size 5: 0.00000391 seconds  
Input size 10: 0.00000572 seconds  
Input size 50: 0.00008679 seconds  
Input size 100: 0.00038847 seconds



```

1 import time
2 import matplotlib.pyplot as plt
3
4 def insertion_sort(arr):
5     for i in range(1, len(arr)):
6         key = arr[i]
7         j = i - 1
8         while j >= 0 and arr[j] > key:
9             arr[j + 1] = arr[j]
10            j -= 1
11            arr[j + 1] = key
12
13 Arr1 = list(range(1, 6))
14 Arr2 = list(range(1, 11))
15 Arr3 = list(range(1, 51))
16 Arr4 = list(range(1, 101))
17 arrays = [Arr1, Arr2, Arr3, Arr4]
18 input_sizes = [len(arr) for arr in arrays]
19
20
21 def measure_time(sort_function, arr):
22     runs = 5
23     total_time = 0
24     for _ in range(runs):
25         copy_arr = arr.copy()
26         start = time.perf_counter()
27         sort_function(copy_arr)
28         end = time.perf_counter()

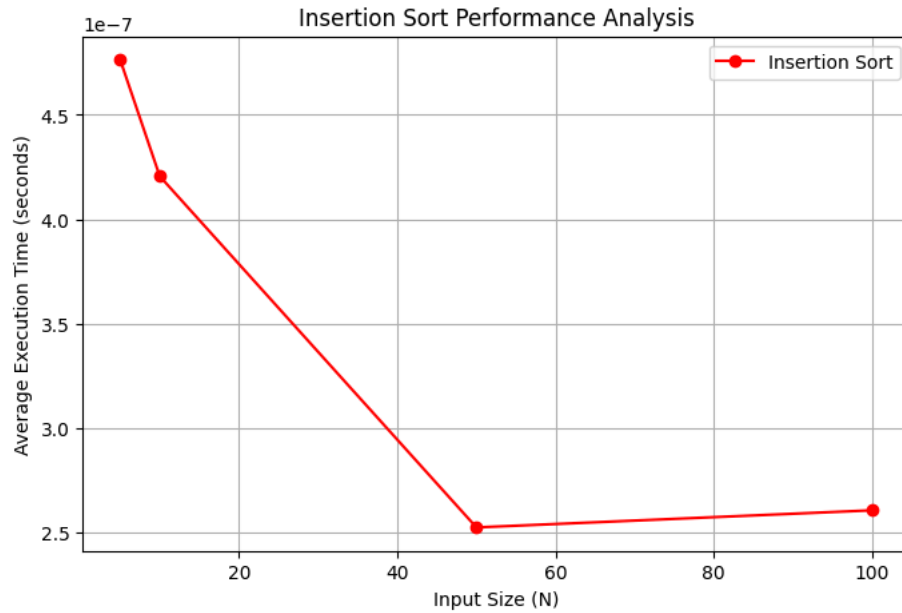
```

```
29 total_time += (end - start)
30 return total_time / runs
31
32 insertion_times = []
33 for arr in arrays:
34     avg_time = measure_time(insertion_sort, arr)
35     insertion_times.append(avg_time)
36
37 copy_arr = arr.copy()
38 insertion_sort(copy_arr)
39 print(f"Sorted array of size {len(arr)}: {copy_arr}")
40 print("\nAverage Execution Times for Insertion Sort:")
41 for size, time_taken in zip(input_sizes, insertion_times):
42     print(f"Input size {size}: {time_taken:.8f} seconds")
43 plt.figure(figsize=(8, 5))
44 plt.plot(input_sizes, insertion_times, marker='o', color='red', label="Insertion Sort")
45 plt.title("Insertion Sort Performance Analysis")
46 plt.xlabel("Input Size (N)")
47 plt.ylabel("Average Execution Time (seconds)")
48 plt.legend()
49 plt.grid(True)
50 plt.show()
51
```

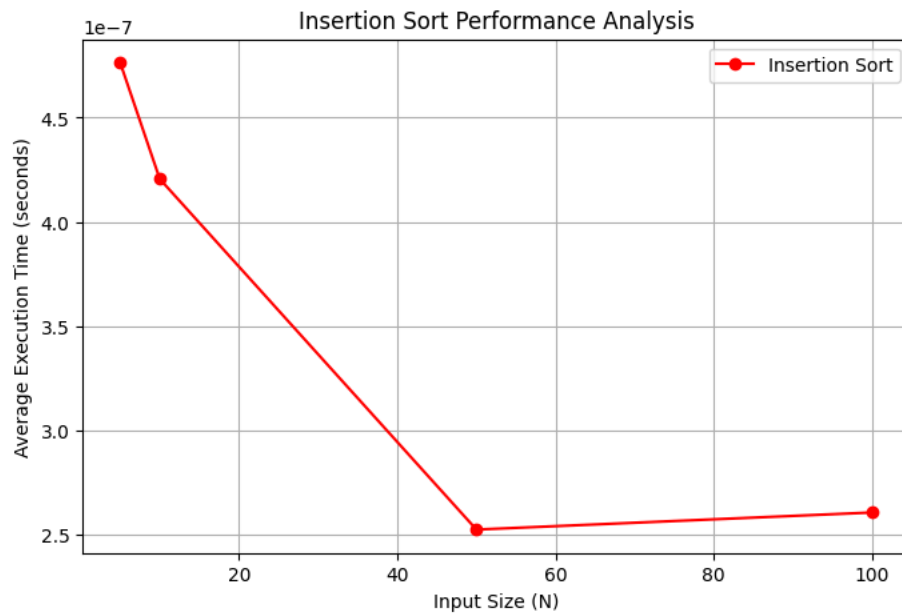
Sorted array of size 100: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,

Average Execution Times for Insertion Sort:

Input size 5: 0.00000048 seconds



Input size 10: 0.00000042 seconds



Input size 50: 0.00000025 seconds

