<u>Sorting Algorithms Performance Report</u>

- **Advanced Algorithm Analysis And Design ICTS6305-99207**
- **Dr. Ashraf Y. Maghari**
- **Student name :** Mohanad Abdalwahab - 120240145

1. Introduction

In this project, we study and compare the performance of three advanced sorting algorithms: **Mergesort**, **Quicksort**, and **Heapsort**. These algorithms are well-known for having good performance with large datasets and a time complexity of O(n log n) in most cases.

We run them on random arrays of sizes:
N = 10^3 , 10^5, 10^7 and 10^8

I also tried to run the algorithms on an array of size 10^9, but my system showed a Memory Error because this size requires a very large amount of RAM. Therefore, the largest size tested in this project is 10^8.

We measure:

- Execution time (seconds)
- Number of comparisons
- Number of moves/swaps

2. Algorithms Description:

- **Mergesort**

- **Idea**: Split the array into two halves, sort each half, then merge them.
- **Time Complexity**: O(n log n) (best, average, worst)
- **Space Complexity**: O(n)
- **Pros**: Stable, same speed even in worst case.
- **Cons**: Needs extra memory.

- **Quicksort**

- **Idea**: Choose a pivot, put smaller numbers on the left and bigger on the right, then sort each side.
- **Time Complexity**:
    - Best & Average: O(n log n)
    - Worst: O(n^2)

- **Space Complexity**: O(log n)
- **Pros**: Very fast in practice.
- **Cons**: Can be slow if pivot choice is bad.

- **Heapsort**

- **Idea**: Make a heap structure, then remove the biggest element and put it at the end until sorted.
- **Time Complexity**: O(n log n) (all cases)
- **Space Complexity**: O(1)
- **Pros**: No extra memory needed.
- **Cons**: Usually slower than Quicksort and Mergesort.

The input files can be found on :

https://drive.google.com/drive/folders/1J22putB0dAYKUkAWYDiYP23ttXJERnQi?usp=sharing

3. Results Table:

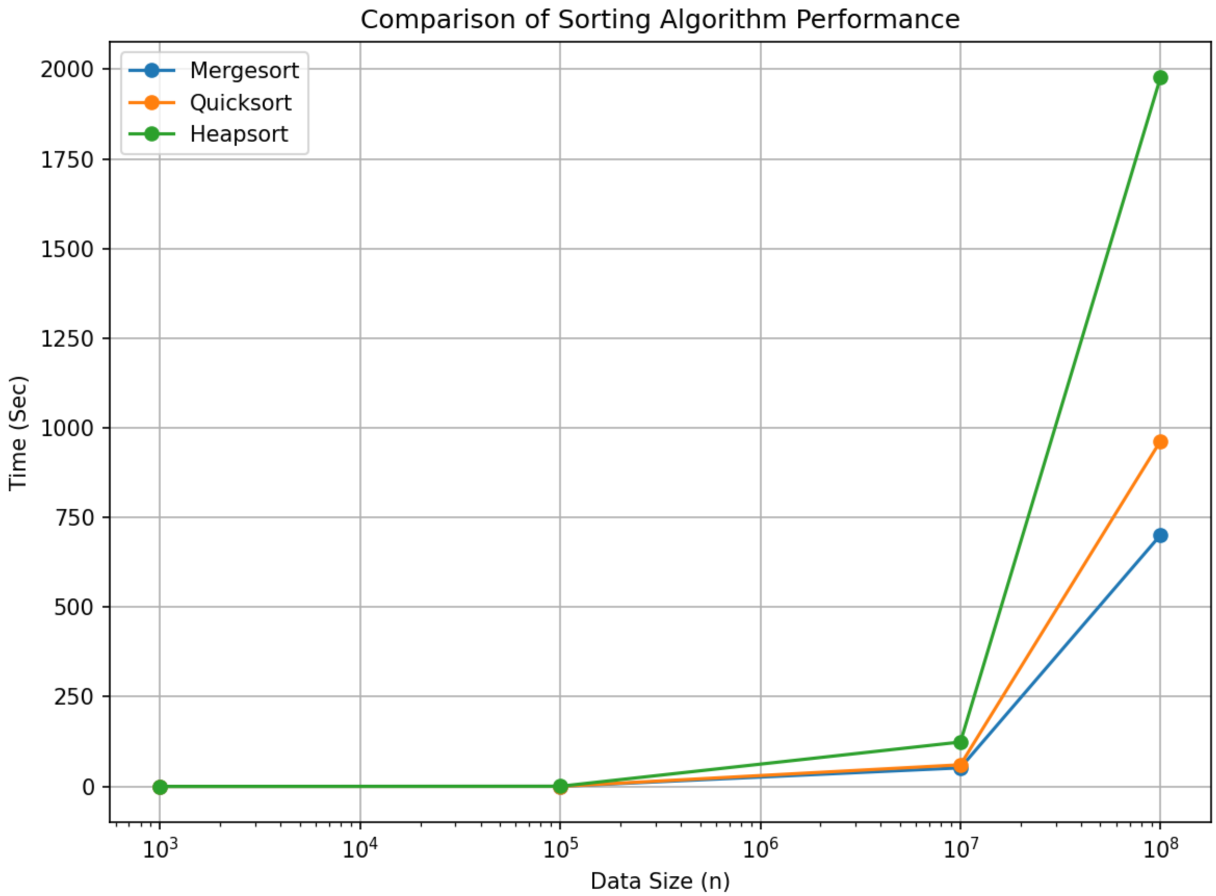
Arry size (n)	Algorithm	execution time (Sec)	number of comparisons	number of swaps
1000	Mergesort	0.001725435	8694	9976
1000	Quicksort	0.001503706	10938	15038
1000	Heapsort	0.002178907	16867	18194
100000	Mergesort	0.277723312	1536031	1668928
100000	Quicksort	0.23032403	1958604	2282338
100000	Heapsort	0.421770334	3019230	3149386
10000000	Mergesort	51.70041227	220102476	233222784
10000000	Quicksort	60.23544621	293349183	342899452
10000000	Heapsort	123.7161436	434642641	447670724
100000000	Mergesort	701.0768421	2532921464	2665782272
100000000	Quicksort	963.4685433	3385640743	3834016248
100000000	Heapsort	1977.557832	5012881237	5143147780

3. **Performance Graph:**

x-axis : Array size (n)

Y-axis : execution time/sec

Lines : Sorting Algorithms (one line per algorithm)



5. Analysis:

- **Fastest Algorithm:**
  - For small and medium sizes, Quicksort was a bit faster.
  - For very large sizes (10^8), Mergesort was better than Heapsort.

- **Data Size Effect**:
  - Time grows when size increases.

    - Growth matches O(n log n)

- **Differences**:
    - Heapsort was slower for huge data.
    - Quicksort is good in average but needs good pivot choice.
    - Mergesort is stable and works well for big data.

6. Conclusion

- **Quicksort** is best if pivot is chosen well.
- **Mergesort** is stable and good for very big data.
- **Heapsort** is useful when memory is limited.