Design & Analysis of Algorithms Project - 1

Table of Contents:

DESCRIPTION

Insertion, Merge and Quicksort algorithms are implemented as part of this project. **Prudhvi Thumma** (Student ID: 1002033401) and **Rohith Kumar Boddu** (Student ID: 1002037081) are members of this team. We both contributed equally towards the implementation of algorithms, drawing comparisons, and documentation. File operations required for the project are mainly read and write operations. Parsing arrays are required while reading files or writing to files. Same text files consisting of auto-generated numbers are used for each sorting algorithm and time taken is noted and displayed as run. Comparisons of theoretical and obtained results are concluded and are detailed.

How to run files:

insertion_sort.py, merge_sort.py, quick_sort.py are files with code to sort numbers generated in text files.

arr_20.txt, arr_100.txt, arr_1000.txt, arr_4000.txt are files where randomly generated are written.

Step 1: Install python and set up a runtime environment for python (check system environment variables in the path)

Step 2: Copy files mentioned above to any directory

Step 3: Open command prompt, navigate to your directory with files

Step 4: Enter python “file_name.py” command compile and run given files. The output will be generated displaying sorted output and time taken to complete the given sort operation

Modules to import

- **random**: This module implements pseudo-random number generators for various distributions.
    - **sample()** function provides us randomly generated integers in specified range
- **time**: This module provides various time-related functions.
    - **time()** function returns us the current time based on which I could calculate the difference in time to perform my sort algorithms.

- **random_number_generator** is a method used to generate a required number of random numbers to run algorithms.
- **read_file** reads content from files, which are written after generating random numbers.
- **time()** gives current time and is used to calculate the time elapsed to run each file content on sortalgorithm.
- **open()** opens a file with the given mode.

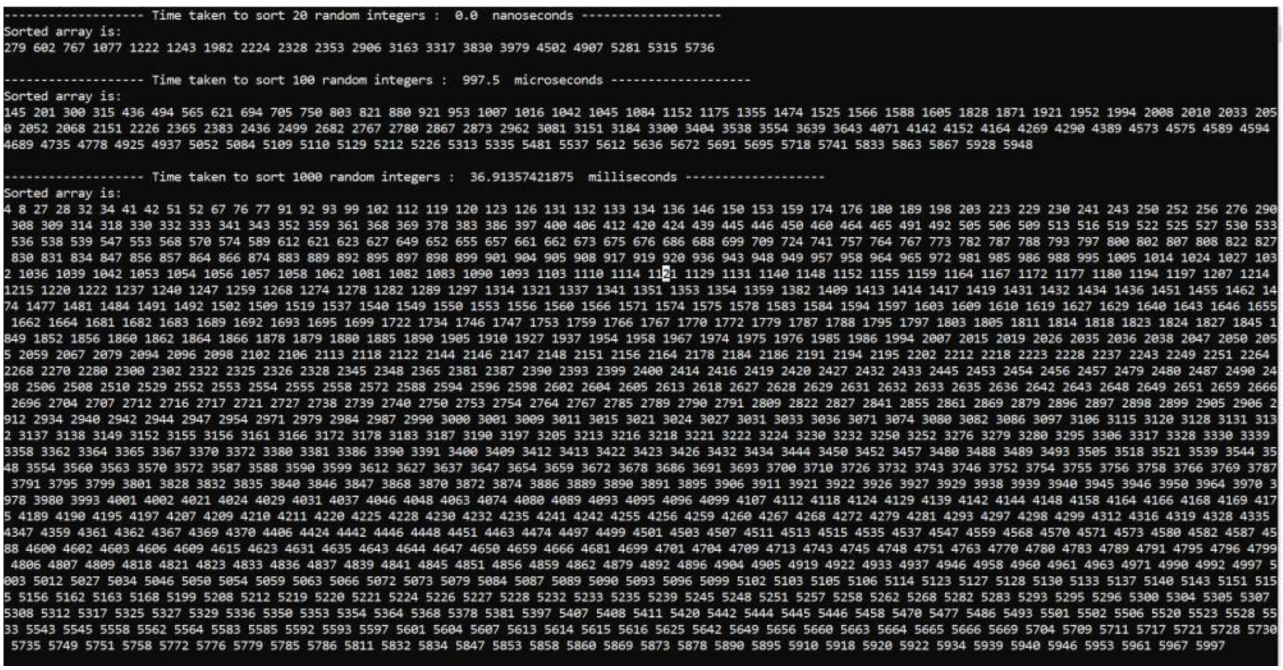
Contributions towards the project:

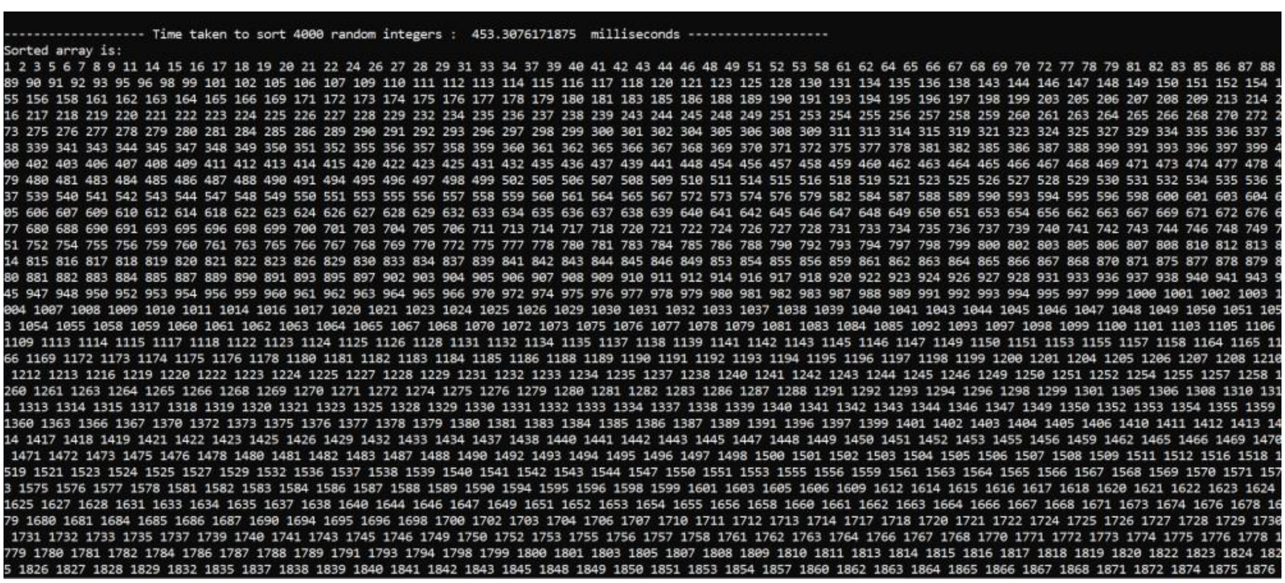
Random number generation, File read and write operations: Prudhvi ThummaSort

algorithms: Rohith Kumar Boddu

Report Documentation and Time complexity comparison: Team

INSERTION SORT



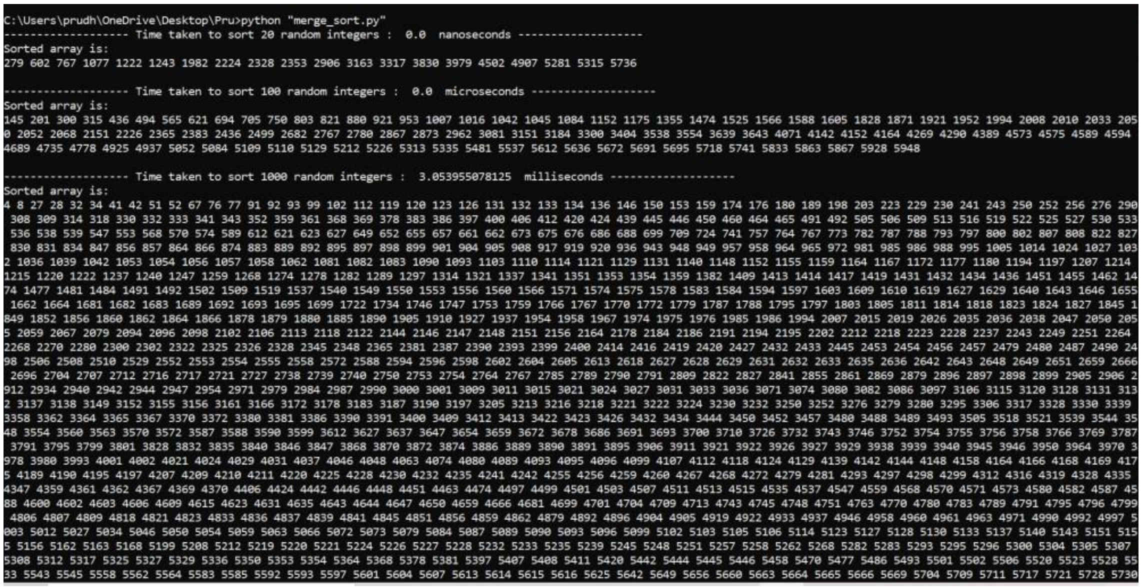


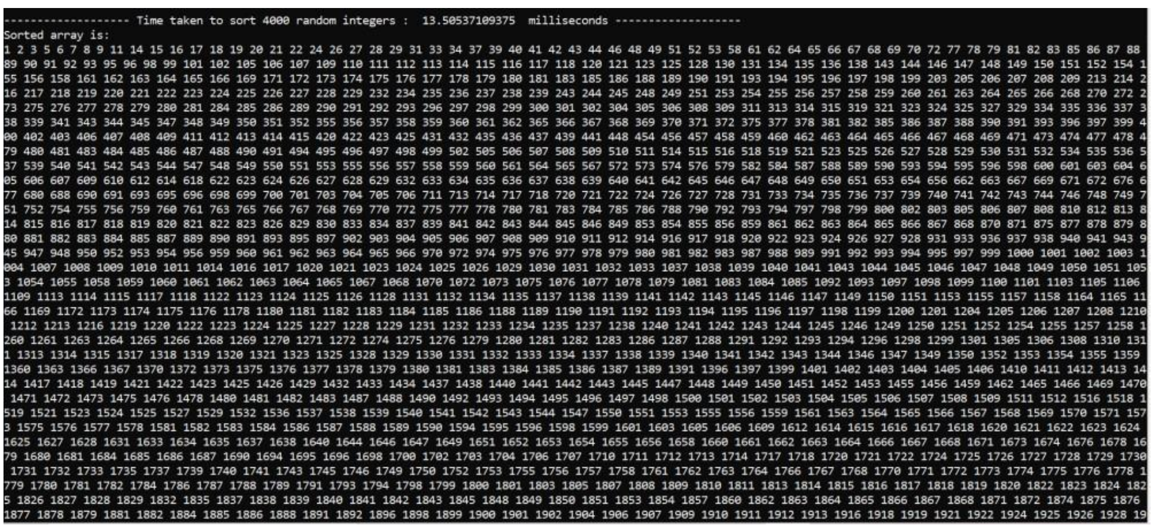
Practical time taken:

20 random integers:	0 nanoseconds
100 random integers:	997.5 microseconds ≈ 1 millisecond
1000 random integers:	36.91357421875 milliseconds
4000 random integers:	453.3076171875 milliseconds

MERGE SORT

Run merge_sort.py file in command prompt as directed, which provides us output with the time taken as below



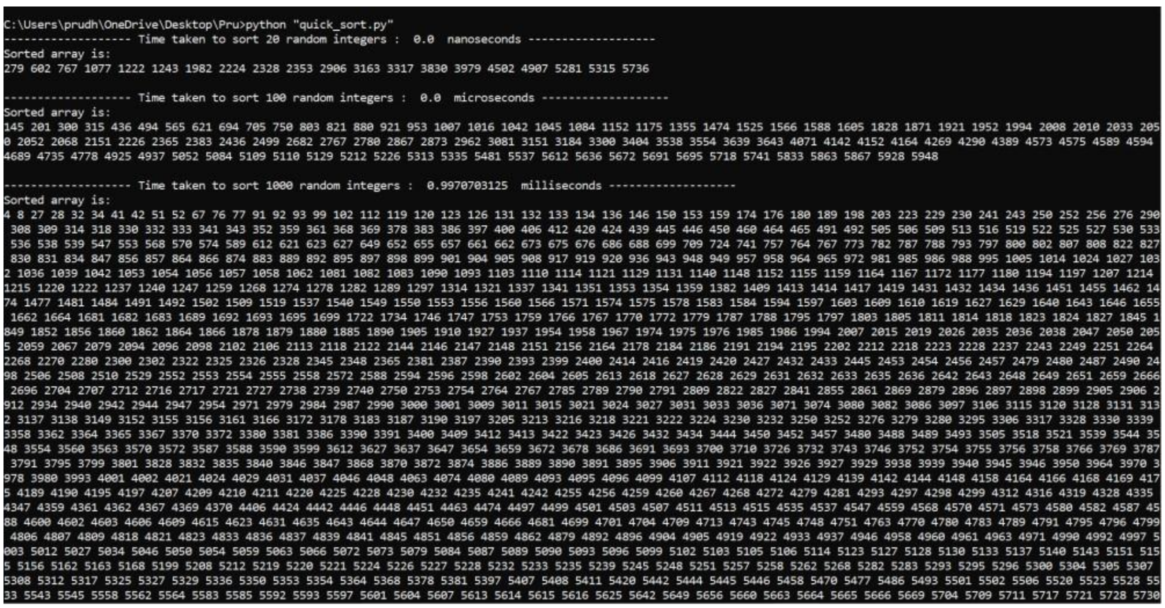


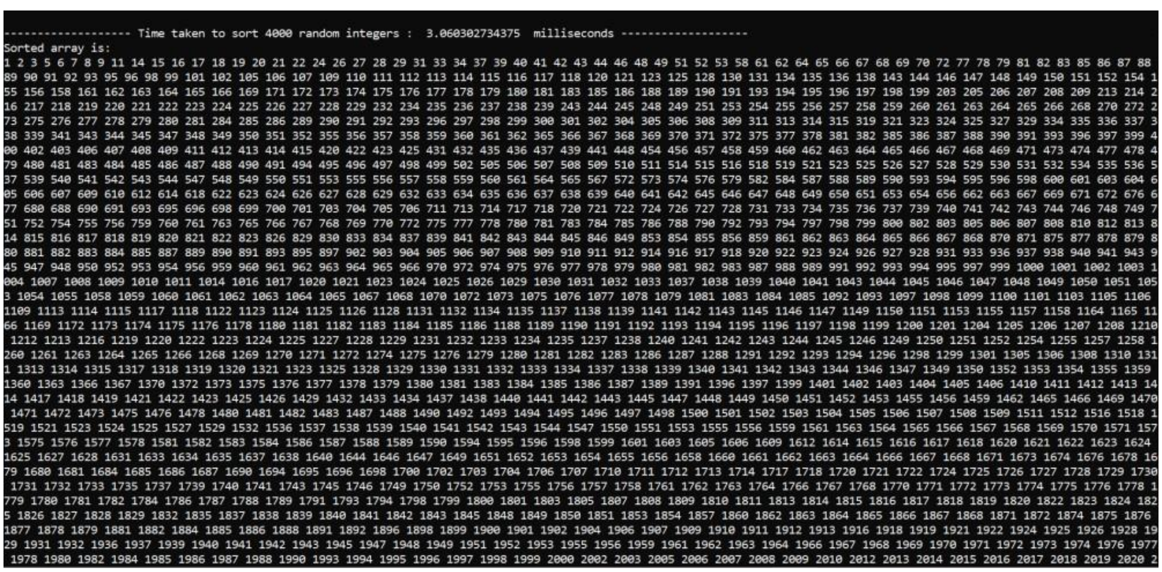
Practical time taken:

20 random integers:	0 nanoseconds
100 random integers:	0 microseconds
1000 random integers:	3.053955078125 milliseconds
4000 random integers:	13.50537109375 milliseconds

QUICK SORT

Run quick_sort.py file in command prompt as directed, which provides us output with the time taken as below





Practical time taken:

20 random integers:	0 nanoseconds
100 random integers:	0 microseconds
1000 random integers:	0.9970703125 milliseconds
4000 random integers:	3.060302734375 milliseconds

Theoretical and Practical Time Complexity Comparison

We will analyze differences in theoretical time complexity and achieve experimental time complexity. Let's check the time complexity of individual sort algorithms

Insertion Sort Theoretical:

- Average Time Complexity: $O(n^2)$
- Worst-case Time Complexity: $O(n^2)$
- Best case Time Complexity: $O(n)$

Theoretically, time taken to sort “n” numbers will increase by order n^2

Time taken to sort 100 numbers is approximately 1 millisecond and for 1000 numbers is 37 milliseconds and to sort 4000 is 453.3 milliseconds.

100numbers	1000numbers
1ms	37ms

The difference is 37 times, but as per average case difference can be in the range of

1000numbers	4000numbers
37ms	453.3ms

The difference is approx. 13 times, but average case expects difference to be in the range of 16times

Insertion Sort justified with average case time complexity but not accurate as we are not considering lower order and coefficients.

Merge Sort Theoretical:

- Average Time Complexity: $O(n \log n)$
- Worst-case Time Complexity: $O(n \log n)$
- Best case Time Complexity: $O(n \log n)$

Theoretically, time taken to sort “n” numbers will increase by order n*(log n)

Time taken to sort 1000 numbers is approximately 3.05ms and for 4000 numbers is 13.5 milliseconds.

1000numbers	4000numbers
3.05 ms	13.5 ms

The difference is approx. 4.4 times, but the average case expects difference to be in the range of 4.8times

Merge Sort average case time complexity is approximated correctly

Quick Sort Theoretical:

- Average Time Complexity: *O (n log n)*
- Worst-case Time Complexity: $O(n^2)$
- Best case Time Complexity*: O (n log n)*

Theoretically, time taken to sort "n" numbers will increase by order n*(log n)

Time taken to sort 1000 numbers is approximately 1 millisecond and for 4000 numbers is 3.06 milliseconds.

1000numbers	4000numbers
1ms	3.06ms

The difference is approx. 3.06 times, but the average case expects difference to be in the range of 4.8times

Quick Sort average case time complexity is justified by our experiment.

REFERENCES

[1] https://docs.python.org/3/library/random.html

[2] https://docs.python.org/3/library/time.html

HONOR CODE

HONOR CODE

I pledge, on my honor, to uphold UT Arlington's tradition of academic integrity, a tradition that values hard work and honest effort in the pursuit of academic excellence.

I promise that I will Submit only work that I personally Create or that I Contribute to group Collaborations, and I will appropriately reference any work from other Sources. I will follow the highest standards of integrity and uphold the Spirit of honor Code.

I will not participate in any form of Cheating/Sharing the questions/Solutions

PRUDHVI THUMMA
1002033401

ROHITH KUMAR BODDU.
1002037081

5th March 2022