Republic of the Philippines
POLYTECHNIC UNIVERSITY OF THE PHILIPPINES
College of Engineering
Sta. Mesa, Manila

Data Structures and Algorithm
Group Laboratory Exercise - Search Algorithms
Group 1

In partial fulfillment of the requirements in
CMPE 201 Data Structures and Algorithms
Academic Year 2023–2024

Submitted by:
Baay, Frances Leigh G.
Legaspi, Ken Andrea G.
Mallari, Ella Marie A.
Membrere, Zemerelin Iris M.
Pabuna, Katrina B.
Sajo, Jhon Kaiser R.
Sevial, Rebekah Joy E.

Submitted to:
Prof. Godofredo T. Avena

November 16, 2023

Data Structures and Algorithm

Group Laboratory Exercise - Search Algorithms

Group 1

Objective:

To implement and compare the performance of various search algorithms on different data sets.

Materials:

- Computer with Python installed (https://www.python.org/downloads/)
- Integrated development environment (IDE) for writing Python code (https://code.visualstudio.com/download)
- Download code: “git clone https://github.com/gavena/PUPSearchAlgo.git” or https://github.com/gavena/PUPSearchAlgo/archive/refs/heads/main.zip

Instructions:

Part 1: Implementing Search Algorithms

Implement all search algorithms in Python (code already provided):

- Linear search
- Binary search
- Jump search
- Exponential search
- Interpolation search
- Ternary Search

Note: You can refer to the earlier explanations and the code provided for each algorithm to help you with the implementation.

Part 2: Generating Test Data

Create python code three different data sets for testing the search algorithms:

a. Small data set: Generate a sorted list of 100 integers.
b. Medium data set: Generate a sorted list of 1,000 integers.
c. Large data set: Generate a sorted list of 10,000 integers.

Choose a random target element from each data set to search for.

Part 3: Testing and Comparing Search Algorithms

For each search algorithm, create a code to measure the time taken to find the target element in each data set. A code has been given but you can change it to get a more efficient answer. Record the execution times for each algorithm and data set combination.

Part 4: Analysis and Conclusion

Create a table or graph to visualize the performance of each search algorithm on the different data sets.

Analyze the results and answer the following questions:

a. Which search algorithm performed the best overall?
b. Did any search algorithms perform better on specific data sets?
c. How did the size of the data set affect the performance of the search algorithms?
d. Write a brief conclusion summarizing your findings?

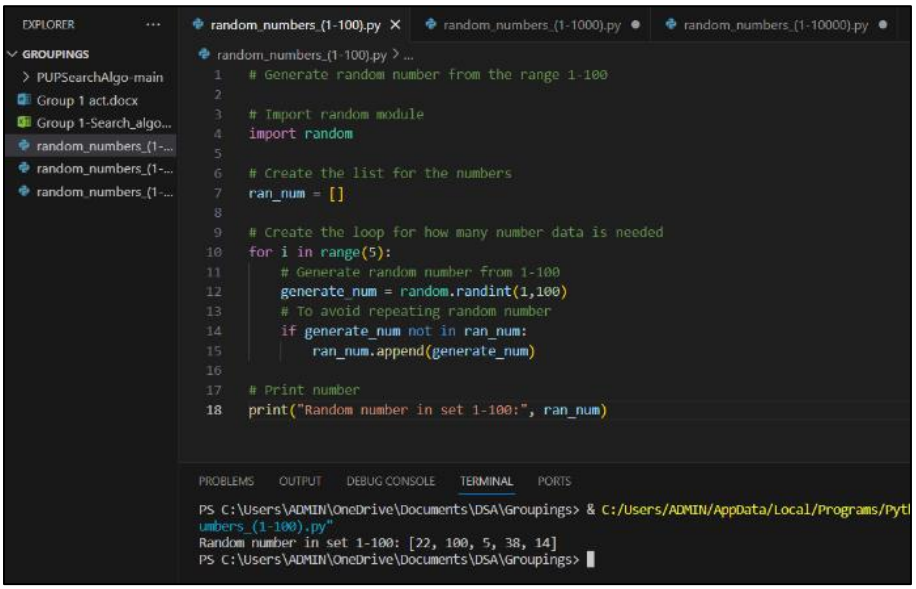
Deliverables:

- Python code for each search algorithm (already provided) and the python code on how to generated dataset and executing the data set.
- Table and graph comparing the performance of the search algorithms on the different data sets (see sample)
- Analysis and conclusion of the exercise

Generating Random Number

Figure 1

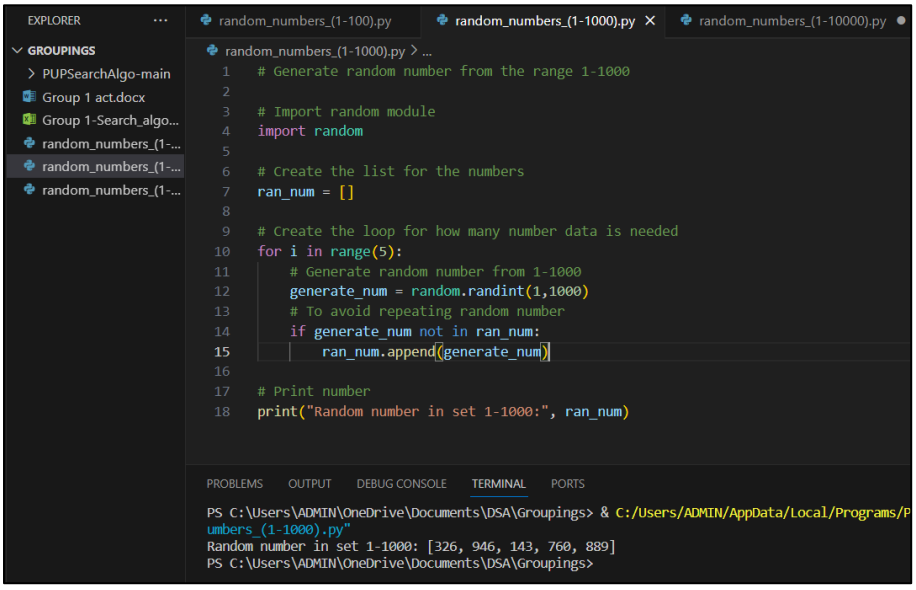
Set 1-100



Note. The number was listed in ascending order on the data sheet. The code can be found on the repository of where this file is located.

Figure 2

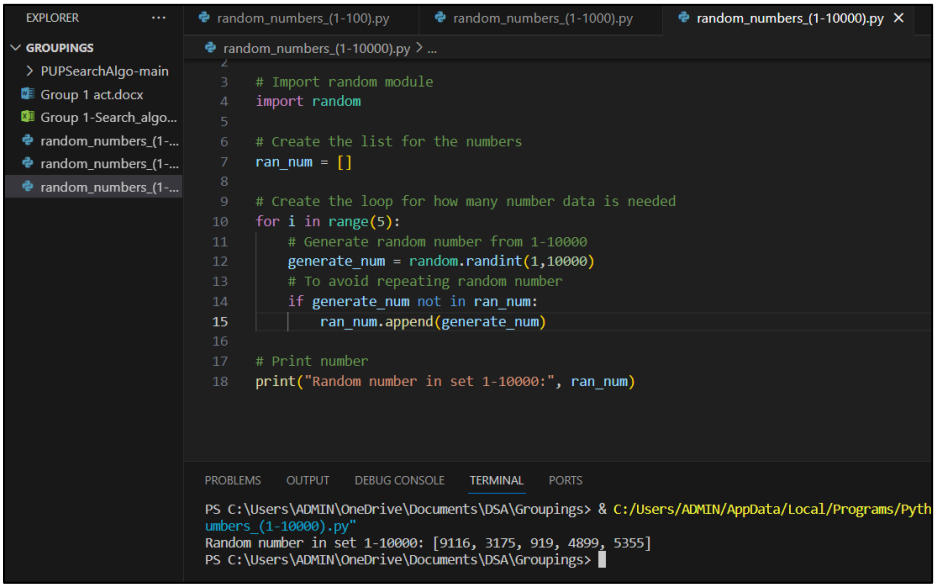
Set 1-1000



Note. The number was listed in ascending order on the data sheet. The code can be found on the repository of where this file is located.

Figure 3

Set 1-10000



Note. The number was listed in ascending order on the data sheet. The code can be found on the repository of where this file is located.

PERFORMANCE OF THE SEARCH ALGORITHMS ON THE DIFFERENT DATA SETS

Table 1

Search Algorithms Analysis

		Linear	Binary	Ternary	Exponential	Interpolation	Jump
Target Set	Search data	Time in Milliseconds					
100	5	0.0164	0.0456	0.0328	0.0407	0.027	0.0539
	14	0.0234	0.0332	0.0291	0.0251	0.0261	0.0373
	22	0.1871	0.0396	0.03	0.0392	0.0393	0.0738
	38	0.0396	0.025	0.0254	0.0187	0.0242	0.042
	100	0.1735	0.0273	0.0245	0.2993	0.0061	0.0436
1000	143	0.0715	0.044	0.358	0.0638	0.0252	0.0475
	326	0.1725	0.0355	0.0249	0.0625	0.0289	0.0633
	760	0.3621	0.0573	0.0805	0.0596	0.0532	0.061
	889	0.7934	0.0307	0.0434	0.0665	0.034	0.0829
	946	1.0081	0.0303	0.0437	0.0833	0.1162	0.0674
10000	919	0.1819	0.0408	0.0568	0.0495	0.0185	0.0388
	3175	0.9177	0.0478	0.0713	0.0455	0.1896	0.1259
	4899	3.1565	0.0319	0.0541	0.0891	0.0211	0.064
	5355	4.204	0.9451	0.3986	0.1603	0.0563	0.033
	9116	2.3695	0.0729	0.0638	0.0462	0.0222	0.1569

Table 2

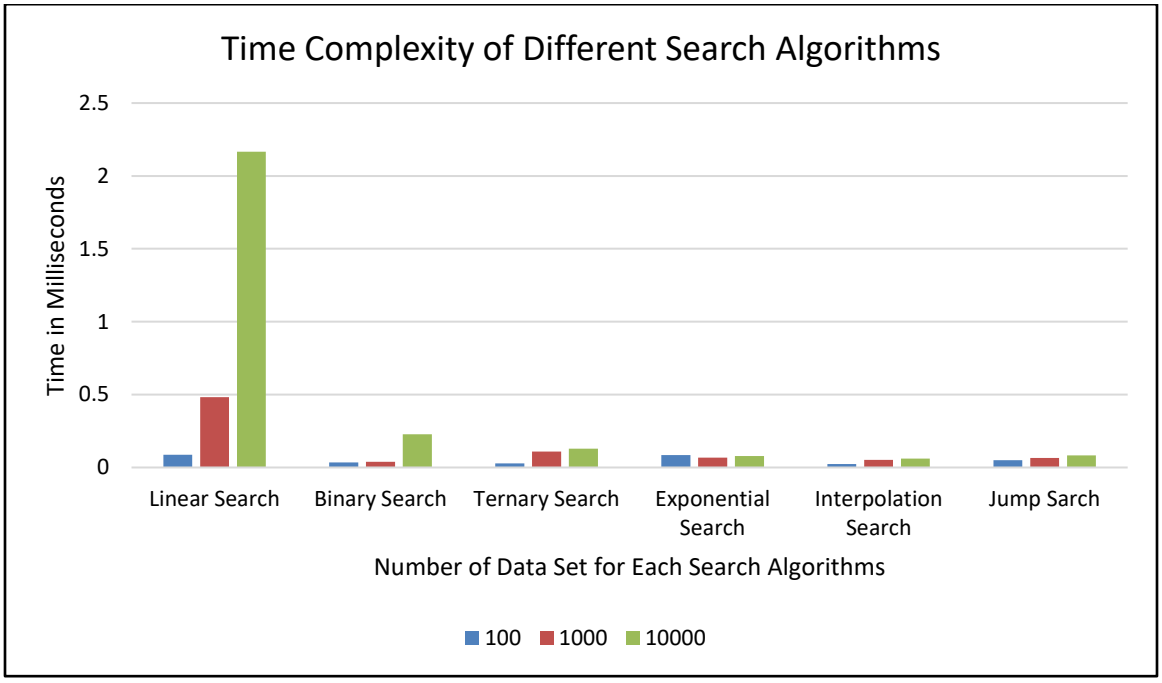
Average Time Complexity

Average	Linear	Binary	Ternary	Exponential	Interpolation	Jump
100	0.088	0.03414	0.02836	0.0846	0.02454	0.05012
1000	0.48152	0.03956	0.1101	0.06714	0.0515	0.06442
10000	2.16592	0.2277	0.12892	0.07812	0.06154	0.08372
TOTAL	**0.911813**	**0.100467**	**0.089127**	**0.07662**	**0.04586**	**0.066087**

RESULTS AND DISCUSSION

Figure 4

Time Complexity of Different Search Algorithms Graph



Note. Analysis can be found below.

A. Which search algorithm performed the best overall?

The data provided in table 2 contains the average time values in milliseconds for different search algorithms. In the context of algorithmic performance, lower time values are indicative of faster execution. Analyzing the given data, the "Interpolation" search algorithm stands out as the top performer with a remarkably low time of 0.04585999995 milliseconds. Following closely is the "Jump" algorithm at 0.06608666663 milliseconds, demonstrating efficient performance. The "Exponential" search algorithm is next in line, taking 0.07662000001 milliseconds. While slightly slower, it outpaces the subsequent algorithms. The "Ternary" search algorithm takes 0.08912666657 milliseconds, and the "Binary" algorithm follows with 0.1004666668 milliseconds. Lastly, the "Linear" search algorithm has the highest time value among the listed algorithms at 0.9118133332

milliseconds. Therefore, based on these time measurements, the ranking from fastest to slowest is: Interpolation, Jump, Exponential, Ternary, Binary, and Linear.

B. Did any search algorithms perform better on specific data sets?

Figure 4 illustrates the performance trends of the six implemented algorithms across datasets of varying sizes. Notably, within the 100-data-point range, interpolation demonstrated superior performance, followed by ternary and binary algorithms. Within the 1000 elements, the binary algorithm emerged as the top performer, followed by jump and exponential algorithms. This trend persisted in the 10000-data-point dataset, where interpolation once again outperformed the other algorithms, with exponential and jump algorithms following respectively.

C. How did the size of the data set affect the performance of the search algorithms?

As the data set gets bigger and gets a wider range, the performance of search algorithms tends to become slower, particularly Linear, Binary, Ternary, Interpolation, and Jump search algorithms. In Exponential algorithm, however, when the data set was 1-10000, the performance was slower compared to when the data set was 1-1000. It became much slower when the data set was 1-100. In conclusion, the performances in Linear, Binary, Ternary, Interpolation, and Jump search algorithms become slower as the data set gets bigger, while Exponential search algorithm has the condition where the data set with 1-1000 performed the fastest, slower in a 1-10000 data set, and the slowest in a 1-100 data set.

D. Write a brief conclusion summarizing your finding

According to the simulation using the given search algorithms, it was found that among the algorithms, interpolation search performed best overall, as it took the shortest amount of time on average. While, in the process of analyzing the specific data sets, it was found that the interpolation, binary, and interpolation algorithms—in ascending order—performed the best on the range of 100, 1000, and 10000 data points. It was also found that the smallest data set (1–100) varied in different algorithms but showed to be the

fastest using the interpolation algorithm, while on the medium-sized data set (1–1000), it showed that binary turned out to be the fastest among the algorithms, and on the largest data set (1–10,000), interpolation turned out to be the fastest. In terms of the slowest algorithm in the overall average of the data sets, it was found that the linear search algorithm took its time to function.

The results, therefore, shows that among the six algorithms (linear, binary, ternary, exponential, interpolation, and jump) the most efficient search algorithm is interpolation search algorithm.