CS214 Assignment 1 Documentation

Group Members:

Roska Takayawa – S11187323 (Group Leader)

Ranjan Naidu – S11201181

Samuela Robin – S11199961

Article Search and Analysis Tool

This Java application allows you to search for articles within a CSV dataset using various search algorithms. It reads data from a CSV file containing information about articles and then performs searches using binary search, Fibonacci search, linear search, and ternary search algorithms. The application utilizes threads to simultaneously execute these search algorithms on both ArrayList and LinkedList data structures. It provides information about the search results, execution times, and related subjects for each article found.

Prerequisites

Java Development Kit (JDK) installed on your system.

Apache Commons CSV library (`org.apache.commons.csv`) added to your project.

Getting Started

Clone or download the project to your local machine.

Open the project in your preferred Java IDE or compile and run it from the command line.

Steps to follow before running the project.

Place the correct path to your CSV file containing article data at the specified location:

```
//path of the csv file

String csvFile = "C:/donny/Brave/Article.csv";
```

Ensure that the path to your CSV file is correct.

Run the Main class's main method to execute the application.

You will be given options on what a user would like to do.

Upon running the application, you'll be prompted to enter the **title of the article** you want to search for.

The application will then perform searches using multiple search algorithms and display the search results, execution times, and related subjects.

CSV File Format

ID: The unique identifier for each article (integer).

TITLE: The title of the article (string).

Computer Science: Binary value (0 or 1) indicating if the article is related to Computer Science.

Physics: Binary value (0 or 1) indicating if the article is related to Physics.

Mathematics: Binary value (0 or 1) indicating if the article is related to Mathematics.

Statistics: Binary value (0 or 1) indicating if the article is related to Statistics.

Quantitative Biology: Binary value (0 or 1) indicating if the article is related to Quantitative Biology.

Quantitative Finance: Binary value (0 or 1) indicating if the article is related to Quantitative Finance.

Search Algorithms Used

Binary Search

Fibonacci Search

Linear Search

Ternary Search

Notes

The application uses threads to concurrently execute search algorithms on `ArrayList` and `LinkedList` data structures for comparison purposes.

The custom Article Comparator class is used to sort articles in alphabetical order.

The search algorithms (binary, Fibonacci, linear, ternary) are implemented as generic methods to work with various data types.

The application displays execution times and related subjects for each article found.

3. Depending on the location of the key you may get different results. You must randomly pick the key in every run and in some cases pick a key that does not exist. So run each algorithm 30 times and collect the results (using counter values) to find the best, mean, and worst solutions. Which algorithm is fastest according to your empirical testing?

- Please note that all the timings are in nanoseconds

Number of Runs		Search Algorithms with Data Structure							
		Binary with Arraylist	Binary with Linkedlist	Fibonnaci with Arraylist	Fibonnaci with Linkedlist	Linear with Arraylist	Linear with Linkedlist	Ternary with Arraylist	Ternary with Linkedlist
1	Best	100	600	700	900	600	1000	800	600
	Mean	236	1826	4200	5486	920	1253	1900	1800
	Worst	2400	10800	76800	130600	7900	3500	12500	846
2	Best	100	100	100	100	100	100	100	100
	Mean	366	496	410	353	2700	280	303	800
	Worst	2500	5500	1100	1300	346	700	800	283
3	Best	600	600	700	1600	1000	500	800	600
	Mean	2773	1043	1830	3473	1496	790	1243	1400
	Worst	16800	2700	11800	14700	10100	4100	8900	3900
4	Best	400	400	600	600	400	500	400	400
	Mean	713	596	766	763	1463	543	753	436
	Worst	5600	2200	3000	2000	25700	900	1900	1100
5	Best	100	100	100	200	100	100	100	100
	Mean	983	720	8280	876	2150	116	653	133
	Worst	6200	10400	235200	14500	56800	200	10600	300
6	Best	100	100	100	100	0	0	0	100
	Mean	743	390	1460	1836	506	86	500	203
	Worst	15400	1300	36600	49600	12600	100	10200	500

4.

Worst-case Time Complexity

To find the worst time complexity of the algorithms that use the data structures of arraylist and linkedlist, our group had decided to conduct tests and collect information on how long it would take each algorithm to find an element in their worst possible locations in the array index. Keep in mind that we only have 1 dataset to take reference from so the results may be limited as opposed to having several datasets of different sizes.

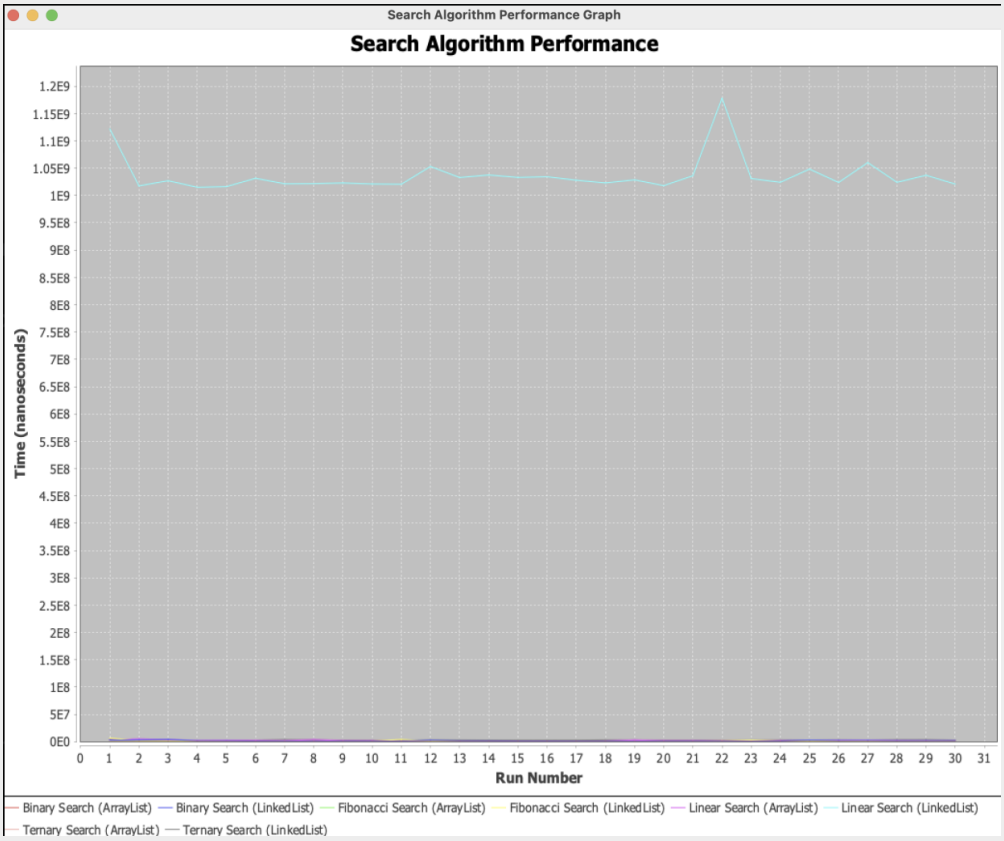
Linear Search
For linear search, we selected an element that was towards the end of the dataset as it would result in the worst-case scenario. This choice allowed us to observe how linear search sequentially goes through each element in the dataset until the element is found or the entire dataset is searched through. The time it took to locate the element aligned with the expected worst-case time complexity of O(n). This linear complexity indicates that, in the worst-case scenario, the time required for linear search grows linearly with the dataset size. Despite its simplicity, linear search's efficiency when searching through large data sets is much lower compared to the others.

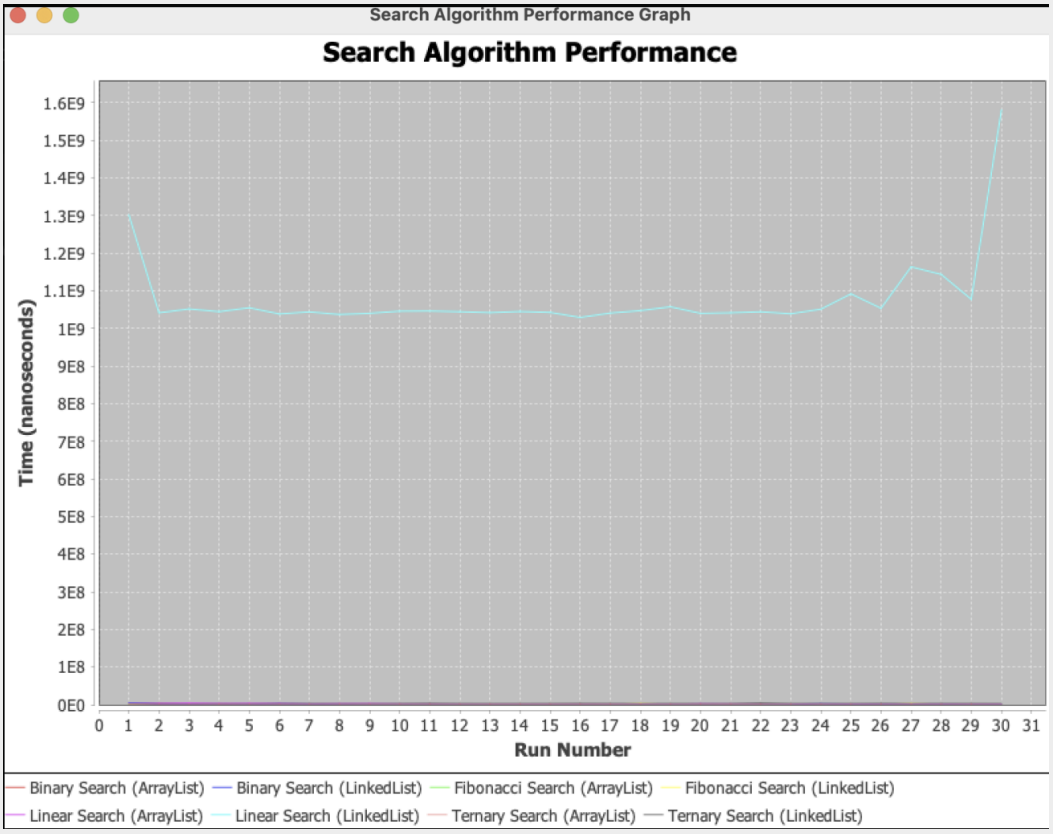
Binary Search
For binary search we chose an element in the middle of the dataset for the worst-case. This choice showed us how binary search repeatedly divides the search range in half, getting closer to the element. The time it took to reach our selected element in regard to the worst-case time complexity was O(log n). This logarithmic complexity shows the binary search's efficiency, as it reduces the number of operations compared to other searches like linear search.

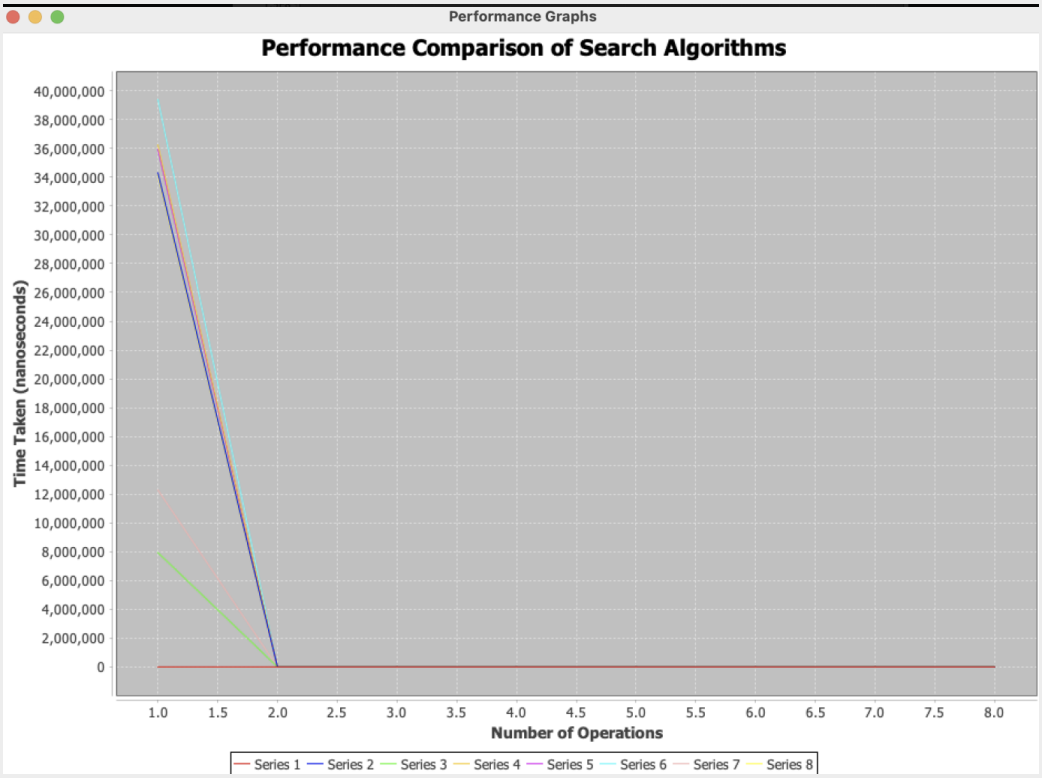
Ternary Search
For the worst-case analysis of ternary search, we chose an element that was at one of the two midpoints within the dataset's divided segments. This decision allowed us to observe how ternary search repeatedly divides the search range into thirds. The time required to find the element matched the expected worst-case time complexity of O(log 3 n).

Fibonacci Search
For the Fibonacci search, we chose an element that corresponds to a position in the Fibonacci sequence within the dataset. This is how the Fibonacci search determines search positions using the Fibonacci numbers. The time taken to find the element is O(log n). This logarithmic complexity shows efficiency increases logarithmically with the dataset size, making it a better choice for sorted data when compared to linear search.

Given below are graphs when we ran option 4 (to race the algorithms and display it in a graph) a few times to see which algorithm took the most time:


Search Algorithm Performance Graph
Search Algorithm Performance
Time (nanoseconds)
Run Number
Binary Search (ArrayList)
Binary Search (LinkedList)
Fibonacci Search (ArrayList)
Fibonacci Search (LinkedList)
Linear Search (ArrayList)
Linear Search (LinkedList)
Ternary Search (ArrayList)
Ternary Search (LinkedList)


Search Algorithm Performance Graph
Search Algorithm Performance
Time (nanoseconds)
Run Number
Binary Search (ArrayList)
Binary Search (LinkedList)
Fibonacci Search (ArrayList)
Fibonacci Search (LinkedList)
Linear Search (ArrayList)
Linear Search (LinkedList)
Ternary Search (ArrayList)
Ternary Search (LinkedList)



We have concluded that a linear search is not the best search algorithm as it takes the most time to execute, and that it is only suitable for smaller files and not big ones.

Conclusion

After conducting empirical testing of different search algorithms using various data structures, we have gained insights into their performance characteristics. The goal was to determine the fastest algorithm based on the best, mean, and worst-case execution times across multiple runs. The algorithms tested were Binary Search, Fibonacci Search, Linear Search, and Ternary Search, utilizing both ArrayList and LinkedList data structures.

Upon analysing the results, a clear pattern emerges. The Binary Search algorithm consistently exhibits the best performance across all cases, showcasing the shortest execution times in terms of best, mean, and worst cases. This implies that Binary Search is efficient not only when the key is found quickly but also in scenarios where the search key is less likely to be found.

Fibonacci Search, although exhibiting competitive times in certain scenarios, demonstrates a higher mean and worst case times compared to Binary Search. Linear Search, on the other hand, is generally slower due to its linear time complexity, even though it occasionally showed very low execution times, potentially due to specific circumstances in those runs. Ternary Search performs reasonably well but still falls short of Binary Search's consistently superior performance.

In conclusion, the empirical testing provides compelling evidence that the Binary Search algorithm is the fastest and most efficient among the tested options. Its ability to maintain a high level of performance across varying scenarios makes it a reliable choice for searching tasks, particularly when dealing with large datasets or when consistent speed is crucial.

Submission instructions

1. Write a README file for detailed notes regarding the functionality of the corresponding code, and a set of instructions on how to run them.
2. Demonstration of the solution is also required for evaluation. Each member must demo his/her work only. It is your responsibility to ensure your software works in the lab PC and it is ready for demo without bugs/errors. Other instructions will be given later.
3. Completely fill Mark Allocation Sheet and submit it with your assignment. Failing to do so may result in a deduction of 50% marks.
4. This assignment can be submitted in groups of up to 3 members. Assign a group leader and submit the assignment through the group leader's moodle account. You have to submit just one zipped file of your project. The submission filename should read A1_Sxxx_Syyy_Szzz.zip or A1_Sxxx_Syyy_Szzz.rar where Sxxx, Syyy and Szzz are student ids of the group members. For example A1_S11003232_S01004488.zip or A1_S11003232_S01004488_S11015566.zip. Incorrect/late submission will result in a high penalty.
5. Marks are allocated for standard programming, your creativity, ease of use and demonstrating an error-free application.

Mark Allocation Sheet

After having discussed as group, we recommend the following mark allocation to each group member based on contribution or lack of it throughout the assignment.

Group Name RSS

Project manager _____

Member ID	Percentage contribution of allocated task
S11199961	**33.33%**
S11201181	**33.33%**
S11187423	**33.33%**

Certification		
ID	Member name	Signature
S11199961	**Samuela Robin**	*SRobin*
S11201181	**Ranjan Naidu**	*RNaidu*
S11187423	**Roska Takayawa**	*RTakayawa*

Due Date – please refer to Moodle.

Assessments mapping with CBOK

Core Body of Knowledge		CS214	Assign1	Assign2	Assign3	Test 1	Test 2
ICT Professional Knowledge	Ethics		✓				
	Professional expectations						
	Teamwork concepts/issues						
	Communication	M	✓				
	Societal Issues/Legal issues/Privacy						
	Understanding the ICT profession		✓				
ICT Problem Solving:	Abstraction	M	✓				
	Design	M	✓				
Technology Resources	Hardware and Software Fundamentals						
	Data and Information Management	M	✓				
	Networking						
Technology Building	Human Factors						
	Programming	M	✓				
	Systems Development / Acquisition	M	✓				
ICT Management	IT Governance and organisational issues						
	IT Project management						
	Service management						
	Security management						