BINUS UNIVERSITY

BINUS INTERNATIONAL

Final Project Cover Letter

(Group Work)

Student Information:	**Surname:**	**Given Name:**	**Student ID Number:**
1.	Kusnadi	Clarissa Audrey Fabiola	2602118490
2.	-	Jeffrey	2602118484
3.	Munthe	Priscilla Abigail	2602109883

Course Code : COMP6048001 **Course Name :** Data Structures and Algorithm

Class : L2AC **Lecturer :** Nunung Nurul Qomariyah, S.Kom., M.T.I., Ph.D.

Type of Assignment : Final Project Report

Submission Pattern

Due Date : 13 January 2023 **Submission Date :** 10 January 2023

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating, and collusion as academic offences which may result in severe penalties, including loss/drop of marks, course/class discontinuity, and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept, and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

Clarissa Audrey Fabiola Kusnadi

Jeffrey

Priscilla Abigail Munthe

Table of Contents

Introduction **4**
- Background 4
- Problem description 4
- Objective 4

Project Specifications **6**
Theoretical Analysis of The Data Structures **8**
Proof of Working System **10**
- ArraylistDemo.java 10
- BinarysearchtreeDemo.java 12
- HashmapDemo.java 13
- LinkedlistDemo.java 14
- Benchmark.java 15
- BenchmarkSorted.java 15

Complexity Analysis **16**
- Add Course Method 16
- Remove Course method 17
- Edit Course Method 18
- Search Course Method 19
- Add Student Method 20
- Remove Student Method 21

Conclusion **26**
Appendices **27**
- Program Manual 27
- Link to the GIT Website 27
- Link to the Presentation File 27

References **28**

Introduction

Background

In light of the advancements in technology and the proliferation of the World Wide Web, educators are increasingly leveraging the internet as a medium for facilitating communication with their pupils. The utilization of online educational tools, also known as courseware, has resulted in educators being able to complete their routine tasks with greater efficiency and reduced frustration (*Moodle in English: Forums*, n.d.). Although numerous iterations of courseware are available, only a limited number of programs offer educators with comprehensive features that fulfill their requirements.

Educators face challenges when determining whether to adopt a pre-packaged courseware solution or a customized system. In addition, the level of computer proficiency among educators generally does not include expertise in programming. Therefore, there is a need for a software solution that is easy to install and adapts to meet their specific needs. Currently, the software is difficult and challenging to configure. The challenge of providing optimal levels of flexibility and user-friendliness has been a persistent problem for educators globally, since the emergence of online course management systems.

Problem description

During the course of this semester, we have studied various data structures and their implementations. Now, it is time to apply our knowledge and analyze the performance of these data structures in a practical scenario. The importance of selecting the appropriate data structure cannot be overstated, particularly when dealing with large volumes of data. Even the smallest difference in efficiency can have a significant impact on the overall performance of a system.

This course management system will encompass functions such as adding and removing courses, storing and retrieving student information, and managing course schedules. We will implement different data structures, such as array lists, linked lists, binary search trees, and hash maps to handle the data efficiently and effectively.

Through this analysis, we hope to gain insights into the performance of each data structure's time efficiency. By understanding this, we can make informed decisions about which data structure is most suitable for specific tasks and ensure optimal performance when working with substantial amounts of data.

Objective

Our main objective is to determine the most efficient data structure for implementing a Course Management System. The criteria of the evaluation will be based upon:

- Speed (time complexity)
- Memory use (space complexity)

These are several features in our system that are going to be analyzed:

1. Adding, removing, and modifying a course in the system.

2. Searching for a course based on the course' name.
3. Adding and removing a student from a particular course in the program.

Project Specifications

- Software and library used:
    - Jetbrains IDE
    - Java Development Kit (JDK)
    - java.util.Scanner
    - java.io.File
    - java.util.HashMap
    - java.util.ArrayList
    - java.util.Date
    - java.lang.System.nanoTime()
- Input:
    - ‘addCourse()’ method: prompts the user to enter the course details such as course name, course ID, day, start time, end time, and lecturer name.
    - ‘removeCourse()’ method: prompts the user to enter the course name to remove.
    - ‘editCourse()’ method: prompts the user to enter the course name to edit the course details.
    - ‘searchCourseByName()’ method: prompts the user to enter the course name to search for.
    - ‘addStudentToCourse()’ method: prompts the user to enter the student ID and name along with the course name that the student wants to be added into.
    - ‘removeStudentFromCourse()’ method: prompts the user to enter the student ID along with the course name that the student wants to be removed from.
- Output:
    - Add course:
        - If the course is added successfully: "✅The course has been successfully added."
    - Remove course:
        - If the course is removed successfully: "✅Successfully removed course!"
        - If no course is found with the given name: "❌No course found with that name!"
        - If there are no courses available: "❌No course available!"
    - Modify course:
        - If the course is modified successfully: "✅Successfully updated course!"
        - If no course is found with the given name: "❌No course found with that name!"
        - If there are no courses available: "❌No course available!"
    - View course:
        - If there are no courses available: "❌No courses available"
        - If there is a course:

            → Course details for each course: Course Name, Course ID, Day, Start Time, End Time, Lecturer Name.
    - Search course by name:

        - If the course is found

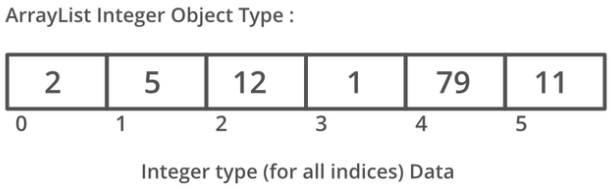
            → Course details for each course: Course Name, Course ID, Day, Start Time, End Time, Lecturer Name.

            → Enrolled students (if any)
        - If no course is found with the given name: "❌No course found with that name!"
    - Add student to a course:
        - If the student is added successfully: "✅Student added to the course successfully."
        - If no course is found with the given name: "❌Course not found."
        - If there are no courses available: "❌No course available!"
    - Remove student from a course:
        - If the student is removed successfully: "✅Successfully removed [student name] from course!"
        - If no course is found with the given name: "❌No course found with that name!"
        - If no student is found with the given ID: "❌No student found with that ID!"
        - If there are no courses available: "❌No course available!"
    - Exit: The program ends.
- Proposed alternative data structures:
    - Array List
    - Binary Search Tree
    - Hash Map
    - Linked List

Theoretical Analysis of The Data Structures

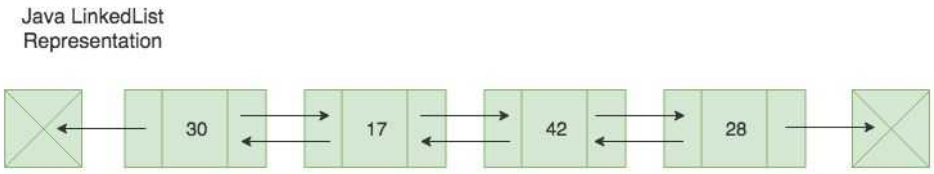
Array List

The first data structure that we decided to implement was Array List. Array List is a dynamic array that can grow or shrink as needed. It provides constant-time access to elements by their index, making it efficient for retrieving elements based on their position. However, inserting or deleting elements in the beginning or middle of the list requires shifting subsequent elements, resulting in a time complexity of O(n). With ArrayList, there are some space limitations, as it requires memory proportional to the number of courses in the system, regardless of whether the list is complete or not. O(n), where n is the number of elements contained in the list, is the space complexity of an ArrayList.



Linked List

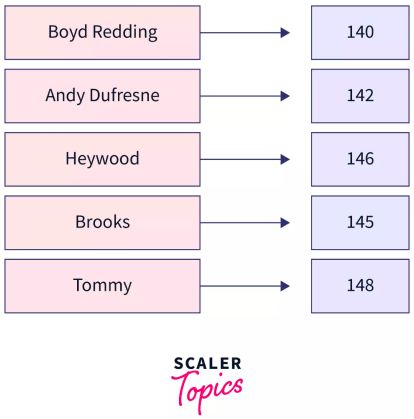
The second data structure that we implemented was Linked List. It is a data structure composed of nodes, where each node contains a value and a reference to the previous and next node. Linked Lists are particularly useful when dynamic resizing and frequent insertion or deletion of elements are required. LinkedList allows for efficient insertion and deletion of elements at arbitrary positions, including the head and tail, with constant-time complexity (O(1)). It eliminates the need for shifting subsequent elements when adding or removing elements. However, accessing elements requires traversing the list from the beginning, resulting in a time complexity of O(n). In contrast to ArrayList, LinkedList can require slightly more memory (O(n)) due to the overhead of maintaining the node structure (reference and data).



Hash Map

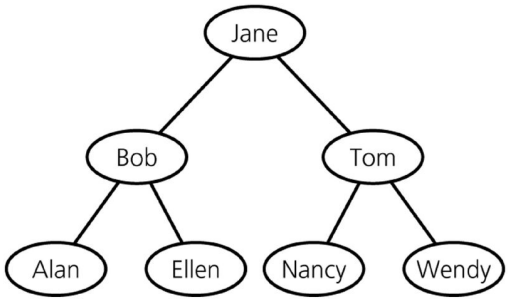
The next data structure that we use is Hash Map. Hash map is a data structure that allows for efficient insertion, deletion, and retrieval of key-value pairs. Hash Maps provide constant-time complexity (O(1)) for average-case operations, such as retrieving elements.

However, the time complexity can degrade to O(n) in the worst case due to collisions. In terms of memory used, the Hash Map would require memory to store key-value pairs and bucket arrays which results in a O(n) space complexity.



Binary Search Tree

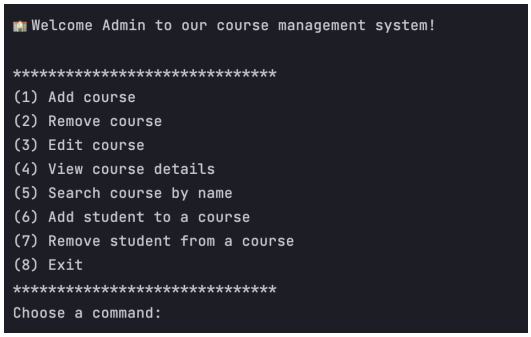
The last data structure that we implement is a Binary Search Tree (BST), which is a tree-based data structure where each node has at most two child nodes. It follows the property that the value of the left child is less than the parent, and the value of the right child is greater. This attribute enables efficient search, insertion, and deletion operations with an average time complexity of O(log n). However, the time complexity would be O(n) in the worst case scenario if the tree is highly unbalanced. The space complexity of a BST is also O(n), where n is the number of nodes in the tree. Each node requires memory to store its data and references to its children. Similar with LinkedList, BST can require more memory compared to other data structures due to the additional overhead of maintaining the tree structure.



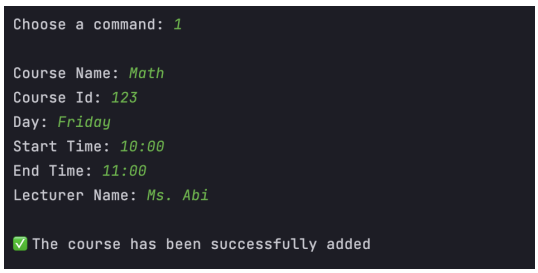
Proof of Working System

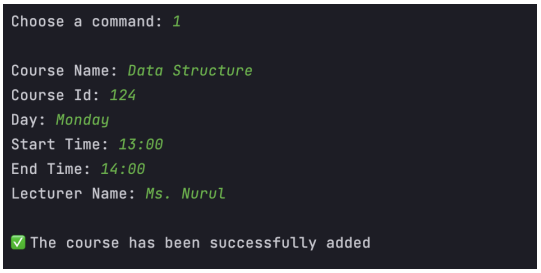
A rray ListArraylistDemo.java

- Display menu

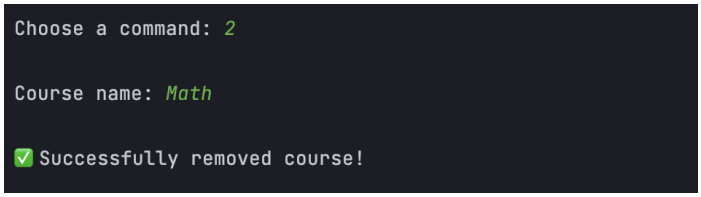


- Add course

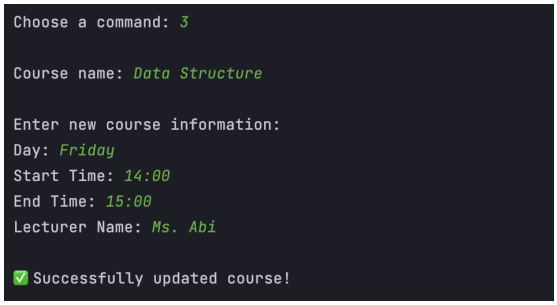




- Remove course



- Edit course



- View course details

```
Choose a command: 4

-Course Details-
Course Name: Data Structure
Course Id: 124
Day: Friday
Start Time: 14:00
End Time: 15:00
Lecturer Name: Ms. Abi
```

- Search course by name

```
Choose a command: 5

Course Name: Data Structure

-Course Details-
Course Name: Data Structure
Course ID: 124
Day: Friday
Start Time: 14:00
End Time: 15:00
Lecturer Name: Ms. Abi

-Enrolled Students-
Name (ID): Jeffrey (000)
```

- Add student to a course

```
Choose a command: 6

Enter student name: Jeffrey
Enter student ID: 000
Enter course name to add the student: Data Structure

✅ Student added to the course successfully.
```

- Remove student from a course

```
Choose a command: 7

Course Name: Data Structure
Student ID: 000

✅ Successfully removed Jeffrey from course!
```

- Exit

```
Choose a command: 8

Process finished with exit code 0
```

BinarysearchtreeDemo.java

File - BinarysearchtreeDemo

```
/Users/clarissaaudrey/Library/Java/JavaVirtualMachines/openjdk-19.0.2/Contents/Home/
bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=60036
:/Applications/IntelliJ IDEA CE.app/Contents/bin -Dfile.encoding=UTF-8 -Dsun.stdout.
encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath /Users/clarissaaudrey/Downloads/
course_management_system-main/out/production/course_management_system-main
BinarysearchtreeDemo

   Welcome Admin to our course management system!
*****************************************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
*****************************************************
Choose a command: 1

Course Name: English
Course Id: 125
Day: Monday
Start Time: 12:00
End Time: 15:00
Lecturer Name: Ms. Abi

   The course has been successfully added

   Welcome Admin to our course management system!
*****************************************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
*****************************************************
Choose a command: 3

Course Name: English

Enter new course information:
Day: Friday
Start Time: 13:00
End Time: 15:00
Lecturer Name: Ms. Audrey

  Successfully updated course!

   Welcome Admin to our course management system!
*****************************************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
*****************************************************
Choose a command: 4

-Course Details-
Course Name: English
Course ID: 125
Day: Friday
Start Time: 13:00
End Time: 15:00
```

File - BinarysearchtreeDemo

```
Lecturer Name: Ms. Audrey

   Welcome Admin to our course management system!
*****************************************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
*****************************************************
Choose a command: 6

Student Name: Jeffrey
Student ID: 000
Enter course name to add the student: English

   Student added to the course successfully.

   Welcome Admin to our course management system!
*****************************************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
*****************************************************
Choose a command: 5

Course Name: English

-Course Details-
Course Name: English
Course ID: 125
Day: Friday
Start Time: 13:00
End Time: 15:00
Lecturer Name: Ms. Audrey

-Enrolled Students-
Name (ID): Jeffrey (000)


   Welcome Admin to our course management system!
*****************************************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
*****************************************************
Choose a command: 7

Course Name: English
Student ID: 000
  Student removed from the course successfully.

   Welcome Admin to our course management system!
*****************************************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
```

File - BinarysearchtreeDemo

```
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
*****************************************************
Choose a command: 2

Course Name: English
Course successfully removed.

   Welcome Admin to our course management system!
*****************************************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
*****************************************************
Choose a command: 8

Process finished with exit code 0
```

HashmapDemo.java

File - HashmapDemo

```
/Users/clarissaaudrey/Library/Java/JavaVirtualMachines/openjdk-19.0.2/Contents/Home/
bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=59418
:/Applications/IntelliJ IDEA CE.app/Contents/bin -Dfile.encoding=UTF-8 -Dsun.stdout.
encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath /Users/clarissaaudrey/Downloads/
course_management_system-main/out/production/course_management_system-main HashmapDemo

  Welcome Admin to our course management system!

*******************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
*******************************
Choose a command: 1

Course Name: Data Structure
Course Id: 123
Day: Monday
Start Time: 13:00
End Time: 14:00
Lecturer Name: Ms. Nurul

  The course has been successfully added

  Welcome Admin to our course management system!

*******************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
*******************************
Choose a command: 3

Course Name: Data Structure

Enter new course information:
Day: Friday
Start Time: 10:00
End Time: 11:00
Lecturer Name: Mr. Jeffrey

  Successfully updated course!

  Welcome Admin to our course management system!

*******************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
*******************************
Choose a command: 6

Enter student name: Abigail
Enter student ID: 000
Enter course name to add the student: Data Structure
```

File - HashmapDemo

```
  Student added to the course successfully.

  Welcome Admin to our course management system!

*******************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
*******************************
Choose a command: 5

Course Name: Data Structure

-Course Details-
Course Name: Data Structure
Course ID: 123
Day: Friday
Start Time: 10:00
End Time: 11:00
Lecturer Name: Mr. Jeffrey

-Enrolled Students-
Name (ID): Abigail (000)

  Welcome Admin to our course management system!

*******************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
*******************************
Choose a command: 7

Course Name: Data Structure
Student ID: 000

  Successfully removed Abigail from course!

  Welcome Admin to our course management system!

*******************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
*******************************
Choose a command: 2

Course Name: Data Structure

  Successfully removed course!

  Welcome Admin to our course management system!

*******************************
(1) Add course
(2) Remove course
```

File - HashmapDemo

```
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
*******************************
Choose a command: 8

Process finished with exit code 0
```

LinkedlistDemo.java

File - LinkedlistDemo

```
/Users/clarissaaudrey/Library/Java/JavaVirtualMachines/openjdk-19.0.2/Contents/Home/
bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=59540
:/Applications/IntelliJ IDEA CE.app/Contents/bin -Dfile.encoding=UTF-8 -Dsun.stdout.
encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath /Users/clarissaaudrey/Downloads/
course_management_system-main/out/production/course_management_system-main
LinkedlistDemo

  Welcome Admin to our course management system!

******************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
******************************
Choose a command: 1

Course Name: Calculus
Course Id: 124
Day: Monday
Start Time: 12:00
End Time: 15:00
Lecturer Name: Ms. Abi

  The course has been successfully added

  Welcome Admin to our course management system!

******************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
******************************
Choose a command: 3

Course Name: Calculus

Enter new course information:
Day: Friday
Start Time: 14:00
End Time: 16:00
Lecturer Name: Mr. Jeffrey

Current Course Information:
Course Name: Calculus
Course ID: 124
Day: Monday
Start Time: 12:00
End Time: 15:00
Lecturer Name: Ms.

  Successfully updated course!

  Welcome Admin to our course management system!

******************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
```

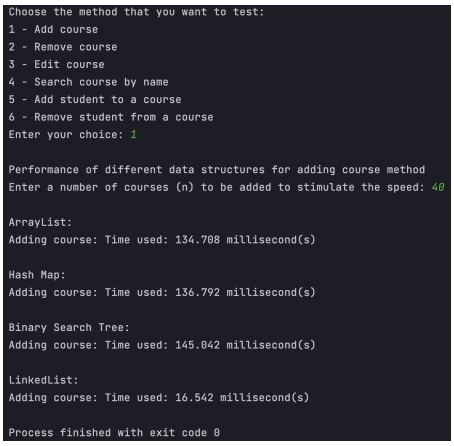
File - LinkedlistDemo

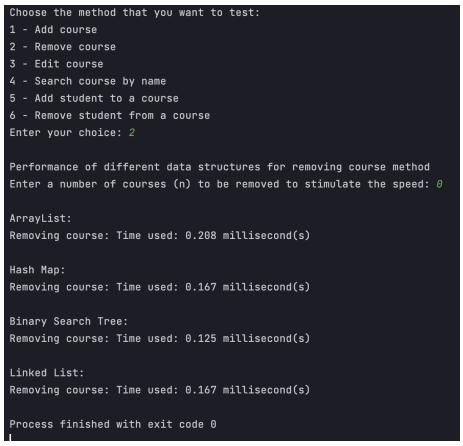
```
******************************
Choose a command: 7

Course Name: Calculus
Student ID: 000

  Successfully removed Clarissa from course!

  Welcome Admin to our course management system!

******************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
******************************
Choose a command: 2

Course Name: Calculus

  Successfully removed course!

  Welcome Admin to our course management system!

******************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
******************************
Choose a command: 8

Process finished with exit code 0
```

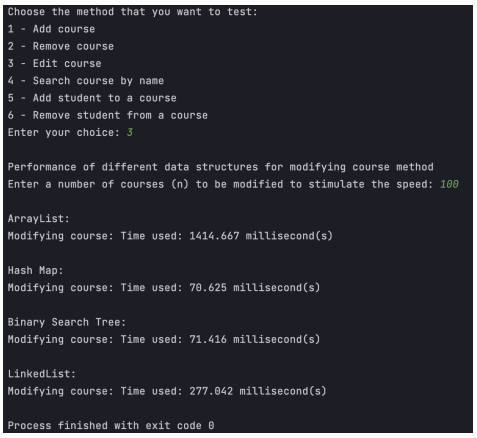
File - LinkedlistDemo

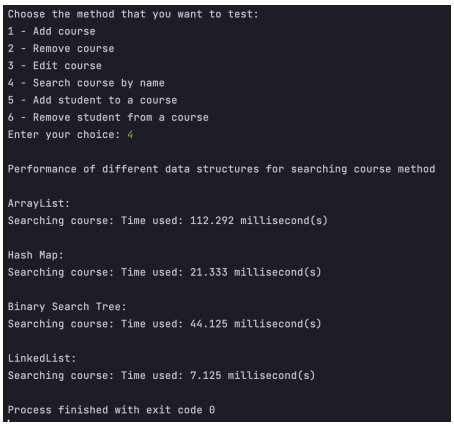
```
(7) Remove student from a course
(8) Exit
******************************
Choose a command: 4

-Course Details-
Course Name: Calculus
Course ID: 124
Day: Friday
Start Time: 14:00
End Time: 16:00
Lecturer Name: Mr. Jeffrey

  Welcome Admin to our course management system!

******************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
******************************
Choose a command: 6

Enter student name: Clarissa
Enter student ID: 000
Enter course name to add the student: Calculus

  Student added to the course successfully.

  Welcome Admin to our course management system!

******************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
******************************
Choose a command: 5

Course Name: Calculus

-Course Details-
Course Name: Calculus
Course ID: 124
Day: Friday
Start Time: 14:00
End Time: 16:00
Lecturer Name: Mr. Jeffrey

-Enrolled Students-
Name (ID): Clarissa (000)

  Welcome Admin to our course management system!

******************************
(1) Add course
(2) Remove course
(3) Edit course
(4) View course details
(5) Search course by name
(6) Add student to a course
(7) Remove student from a course
(8) Exit
```

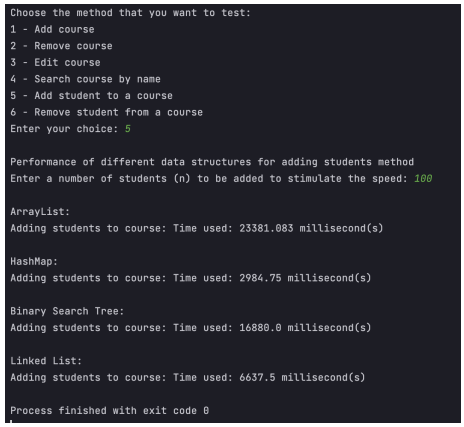
Benchmark.java

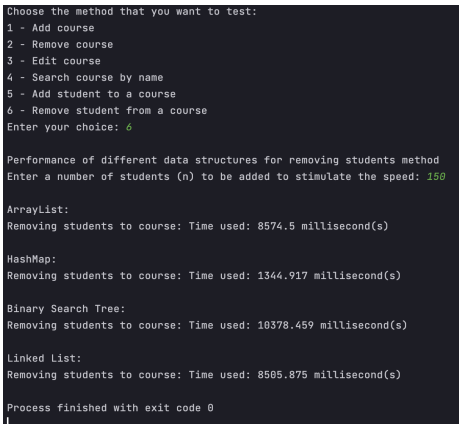




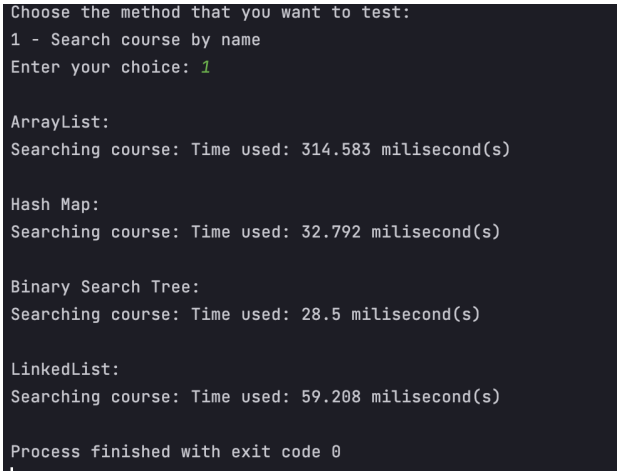








BenchmarkSorted.java



Complexity Analysis

To ensure accurate and reliable results, we tested each method with each data structure 10 times, calculating the average time, while examining datasets of 0, 50, 100, 150 data points.

Add Course Method

```
case "1":
    Scanner read1 = new Scanner(System.in);
    System.out.println("\nPerformance of different data structures for adding course method");
    System.out.print("Enter a number of courses (n) to be added to stimulate the speed: ");
    n = read1.nextInt();
    //ArrayList
    System.out.println("\nArrayList: ");
    System.out.print("Adding course: ");
    startTime1 = System.nanoTime();
    for (int i = 0; i < n && n <= courseCount; i++) {
        arrayList.addCourse(courseName[i], courseId[i], courseDay[i], courseStartTime[i], courseEndTime[i], lecturerName[i]);
    }
    endTime1 = System.nanoTime();
    getTime(startTime1, endTime1);

    //Hashmap
    System.out.println("\nHash Map: ");
    System.out.print("Adding course: ");
    startTime2 = System.nanoTime();
    for (int i = 0; i < n && n <= courseCount; i++) {
        hashMap.addCourse(courseName[i], courseId[i], courseDay[i], courseStartTime[i], courseEndTime[i], lecturerName[i]);
    }
    endTime2 = System.nanoTime();
    getTime(startTime2, endTime2);

    //Binary Search Tree
    System.out.println("\nBinary Search Tree: ");
    System.out.print("Adding course: ");
    startTime3 = System.nanoTime();
    for (int i = 0; i < n && n <= courseCount; i++) {
        binarySearchTree.addCourse(courseName[i], courseId[i], courseDay[i], courseStartTime[i], courseEndTime[i], lecturerName[i]);
    }
    endTime3 = System.nanoTime();
    getTime(startTime3, endTime3);
```

Add course - 0 course

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	0.357	0.299	0.33	0.288	0.31	0.315	0.28	0.434	0.341	0.331	0.3285
LinkedList	0.176	0.235	0.234	0.192	0.199	0.209	0.304	0.244	0.19	0.197	0.218
Hash Map	0.224	0.338	0.268	0.268	0.323	0.326	0.244	0.253	0.26	0.252	0.2756
Binary Search Tree	0.227	0.323	0.291	0.186	0.232	0.326	0.297	0.28	0.201	0.201	0.2564

Add course - 50 courses

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	34.41	28.059	38.038	38.87	28.768	28.645	42.059	25.905	26.431	26.524	31.7709
LinkedList	17.261	17.491	16.702	16.233	17.255	16.949	17.166	16.902	18.777	16.739	17.1475
Hash Map	107.84	88.507	79.802	98.633	117.362	96.856	84.78	79.617	84.838	94.189	93.2424
Binary Search Tree	131.265	122.835	118.214	121.012	108.05	121.624	124.758	119.62	138.829	125.908	123.2115

Add course - 100 courses

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	63.007	43.108	43.442	39.351	40.492	45.056	40.399	50.599	41.898	41.115	44.8467
LinkedList	28.239	30.706	27.224	26.699	28.717	26.6	27.36	27.196	33.103	26.989	28.2833
Hash Map	130.617	151.654	155.955	120.715	207.595	126.053	133.236	209.71	134.125	121.261	149.0921
Binary Search Tree	214.85	189.54	202.78	203.283	185.967	185.081	203.286	193.272	207.032	205.605	199.0696

Add course - 100 courses

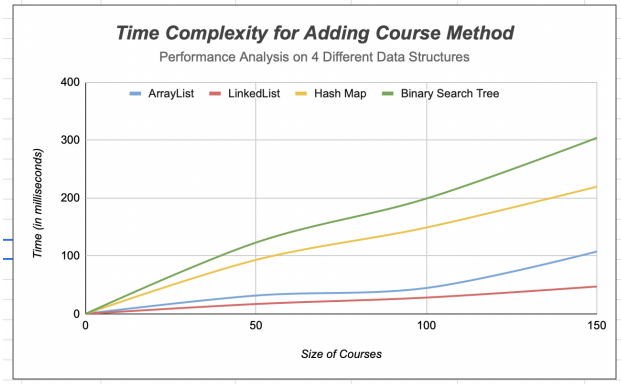
Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	63.007	43.108	43.442	39.351	40.492	45.056	40.399	50.599	41.898	41.115	44.8467
LinkedList	28.239	30.706	27.224	26.699	28.717	26.6	27.36	27.196	33.103	26.989	28.2833
Hash Map	130.617	151.654	155.955	120.715	207.595	126.053	133.236	209.71	134.125	121.261	149.0921
Binary Search Tree	214.85	189.54	202.78	203.283	185.967	185.081	203.286	193.272	207.032	205.605	199.0696

Add course - 150 courses

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	78.826	120.85	98.039	189.277	136.319	116.004	75.711	79.027	94.772	88.13	107.6955
LinkedList	41.88	46.881	60.913	51.123	41.126	47.326	46.973	47.343	47.363	42.525	47.3453
Hash Map	154.228	200.452	211.304	270.89	242.54	192.846	252.989	235.385	272.269	162.465	219.5368
Binary Search Tree	308.466	299.646	305.639	317.46	291.769	301.499	306.348	307.373	312.718	286.092	303.701

Add Course Method

Data Structures	Average Time (in milliseconds)			
	0	50	100	150
ArrayList	0.3285	31.7709	44.8467	107.6955
LinkedList	0.218	17.1475	28.2833	47.3453
Hash Map	0.2756	93.2424	149.0921	219.5368
Binary Search Tree	0.2564	123.2115	199.0696	303.701



Remove Course method

Remove course - 0 course

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	0.26	0.28	0.337	0.299	0.302	0.341	0.317	0.294	0.341	0.313	0.3084
LinkedList	0.196	0.218	0.196	0.211	0.192	0.262	0.195	0.198	0.205	0.199	0.2072
Hash Map	0.247	0.263	0.233	0.262	0.216	0.279	0.271	0.278	0.249	0.25	0.2548
Binary Search Tree	0.233	0.207	0.194	0.189	0.206	0.266	0.208	0.232	0.21	0.186	0.2131

Remove course - 50 courses

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	704.488	706.277	700.624	800.454	699.852	706.894	724.613	702.348	724.326	739.837	720.9713
LinkedList	15.154	17.033	20.847	15.374	16.269	15.896	21.88	15.254	15.783	16.365	16.9855
Hash Map	36.466	44.114	36.778	36.608	48.243	35.772	40.354	36.47	36.427	35.966	38.7198
Binary Search Tree	92.232	81.919	122.989	78.526	79.888	76.385	102.714	90.474	91.552	71.674	88.8353

Remove course - 100 courses

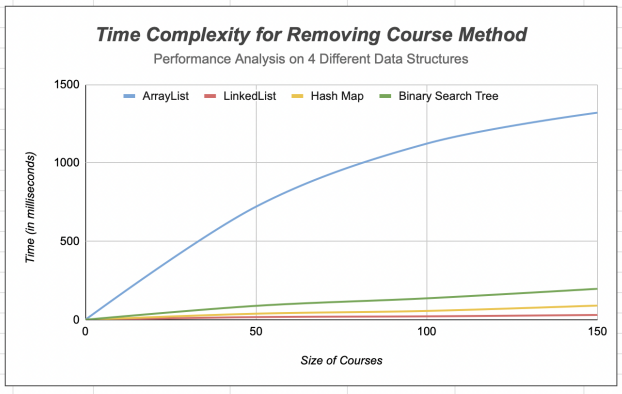
Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	1332.867	1098.247	1099.029	1084.018	1092.979	1166.847	1091.988	1086.359	1085.474	1088.021	1122.5829
LinkedList	24.357	20.905	21.242	22.226	21.427	21.258	21.748	22.417	20.96	21.623	21.8163
Hash Map	60.405	60.473	61.215	60.329	58.382	57.761	60.294	57.997	56.901	29.882	56.3639
Binary Search Tree	125.934	141.108	126.836	146.409	129.663	143.026	130.185	148.203	125.206	150.172	136.6742

Remove course - 150 courses

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	1266.011	1248.466	1249.162	1263.565	1265.246	1262.029	1264.17	1297.245	1607.308	1474.547	1319.7749
LinkedList	28.947	32.568	30.084	28.396	30.603	33.038	30.015	29.746	29.441	34.223	30.7061
Hash Map	79.179	96.742	81.466	102.868	84.548	80.875	85.461	90.249	89.321	115.123	90.5832
Binary Search Tree	163.635	187.383	187.806	170.149	183.854	314.958	178.178	180.058	188.673	219.234	197.3928

Remove Course Method

Data Structures	Average Time (in milliseconds)			
	0	50	100	150
ArrayList	0.3084	720.9713	1122.5829	1319.7749
LinkedList	0.2072	16.9855	21.8163	30.7061
Hash Map	0.2548	38.7198	56.3639	90.5832
Binary Search Tree	0.2131	88.8353	136.6742	197.3928



Edit Course Method

Modify course - 0 course

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	0.667	0.583	0.542	0.417	0.333	0.291	0.75	0.709	0.75	0.25	0.5292
LinkedList	0.5	0.542	0.416	0.416	0.375	0.333	0.458	0.375	0.375	0.209	0.3999
Hash Map	0.333	0.334	0.25	0.292	0.125	0.25	0.375	0.292	0.458	0.125	0.2834
Binary Search Tree	0.583	0.417	0.459	0.5	0.166	0.291	0.5	0.25	0.542	0.208	0.3916

Modify course - 50 courses

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	960.333	761.583	863.541	1045.042	1011.791	816.625	418.583	924.334	881.291	795.75	847.8873
LinkedList	46.916	61.791	55.167	56.625	76.75	62.791	40.416	53	63.834	48.5	56.579
Hash Map	50.542	58.959	49.625	55.083	75	62.084	36.333	51.5	70.833	40.541	55.05
Binary Search Tree	100.791	97.083	96.5	96.208	91.375	89.334	53	114.083	102.416	88.208	92.8998

Modify course - 100 courses

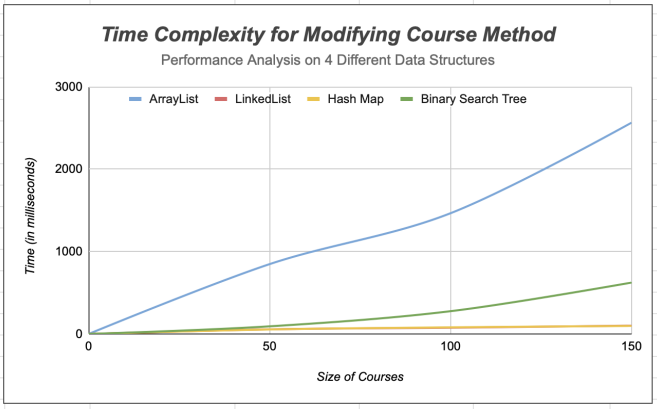
Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	1641.542	1476.416	1590.875	1561.208	2059.667	1631.417	927.958	877.333	1273.417	1618.167	1465.8
LinkedList	72.5	81.167	81.584	91.292	90.125	79.458	63.459	41.042	73.792	89	76.3419
Hash Map	82.541	76.542	94.833	81.208	91.875	85.041	67.459	48.833	75.083	101.917	80.5332
Binary Search Tree	289.084	284.875	288.042	292.375	349.75	284.959	245.791	206.167	215.25	311.084	276.7377

Modify course - 150 courses

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	2771.709	2677.334	2391.959	2643.958	2390.458	2735.291	2564.708	1621.917	2727.208	3123.25	2564.7792
LinkedList	112.25	101.208	96	100.209	103.917	90.792	92.5	110.792	99.291	106.792	101.3751
Hash Map	102.625	94.375	94.041	96.292	95	98.334	89.666	114.417	97.584	95.125	97.7459
Binary Search Tree	603.708	609.625	607.833	618.791	615.958	628.334	624.25	705.542	610.084	614.666	623.8791

Modify Course Method

Data Structures	Average Time (in milliseconds)			
	0	50	100	150
ArrayList	0.5292	847.8873	1465.8	2564.7792
LinkedList	0.3999	56.579	76.3419	101.3751
Hash Map	0.2834	55.05	80.5332	97.7459
Binary Search Tree	0.3916	92.8998	276.7377	623.8791



Search Course Method

In assessing the performance of the 'searchCourse()' method, we focused exclusively on the 150-data point dataset to comprehensively analyze its impact on search efficiency. Furthermore, we conducted evaluations using both sorted and unsorted data to gain insights into the algorithm's effectiveness in different scenarios.

Search course - Unsorted

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	206.167	234.333	136.209	229.958	426.875	287.25	115.209	139.791	138.333	210.291	212.4416
LinkedList	37.542	49.25	30.083	47.792	36.208	31.875	30.791	30	22	48.125	36.3666
Hash Map	58.791	68.375	36.042	71.625	31.208	53.5	40.666	41.917	38	76.542	51.6666
Binary Search Tree	23.834	19.833	12.209	19.792	18.375	16.25	11.667	12.333	12.625	21.583	16.8501

Search Course Method - Unsorted

Data Structures	Average Time (in milliseconds)
	150
ArrayList	212.4416
LinkedList	36.3666
Hash Map	51.6666
Binary Search Tree	16.8501

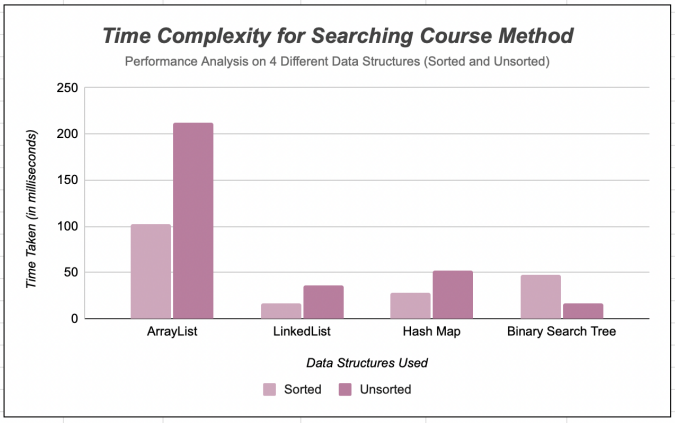
Search Course (150 Courses) - Sorted

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	77.42	81.447	83.222	114.908	125.158	92.292	88.186	119.11	116.593	128.623	102.6959
LinkedList	14.676	15.145	13.314	18.969	14.286	16.321	17.516	15.662	22.695	15.933	16.4517
Hash Map	26.702	26.757	23.55	24.586	24.531	24.837	26.349	24.702	45.988	30.14	27.8142
Binary Search Tree	51.699	51.84	39.854	52.497	60.906	39.95	58.182	51.972	14.661	58.065	47.9626

Search Course Method - Sorted

Data Structures	Average Time (in milliseconds)
	150
ArrayList	102.6959
LinkedList	16.4517
Hash Map	27.8142
Binary Search Tree	47.9626

	ArrayList	LinkedList	Hash Map	Binary Search Tree
Sorted	102.6959	16.4517	27.8142	47.9626
Unsorted	212.4416	36.3666	51.6666	16.8501



Add Student Method

Add student - 0 students

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	3.708	4.583	4.708	4.75	4.084	4.709	4.75	4.875	4.625	4.75	4.5542
LinkedList	3.875	3.959	3.208	3.167	3.792	3.167	2.834	2.667	3.834	2.75	3.3253
Hash Map	3.917	3.5	3.708	3.667	3.625	3.459	2.875	2.916	3.458	2.917	3.4042
Binary Search Tree	5.417	4.75	5.292	4.709	6.791	5.209	4.166	4.125	4.5	4.041	4.9

Add student - 50 students

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	18339.583	20708	16742.25	16628.834	16757.459	16309.708	29948.041	15748.583	18221.208	14756.5	18416.0166
LinkedList	4216.5	3986.916	4054.375	4214.583	4652.625	4254.375	4433.542	4397.709	3950	4019.875	4218.05
Hash Map	2089.625	1778.416	4703.625	1989.875	4030.333	1980.666	4083.5	1979.542	2244.875	2576.5	2745.6957
Binary Search Tree	13677.292	12431.625	13124	13105	15327.625	13598.833	11676.916	14220.75	13322.75	13037.792	13352.2583

Add student - 100 students

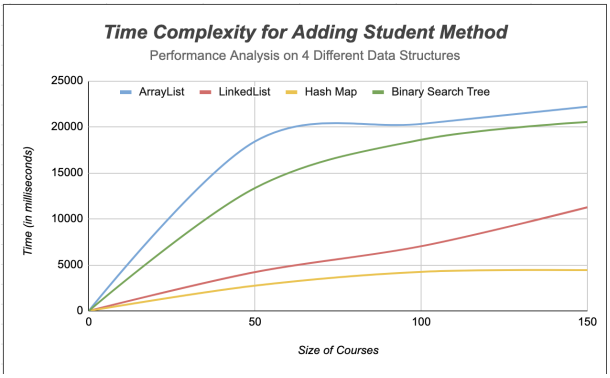
Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	18216.125	19661.125	18712	28119.459	21004.875	20056.459	18788.375	21926.667	17430.708	19246.584	20316.2377
LinkedList	7602.875	6963.916	6903.209	6908.334	6913.334	6229.417	6556.792	7025.75	8050.417	7124.042	7027.8086
Hash Map	5358.125	3004.417	3191.584	3425.917	5972.208	5432.041	4962.375	2766.583	4991.5	3341.042	4244.5792
Binary Search Tree	18373.125	17923.333	19042.542	17716.584	18000.708	22232.375	17260.709	18154.333	18828.959	18411.625	18594.4293

Add student - 150 students

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	25081	23250.208	22456.083	21998.208	23430.958	23828.041	22241.584	19727.042	19541.75	20402.708	22195.7582
LinkedList	10857.5	11787.208	10949.75	10940.917	11133.959	11690.375	10931.708	11149.5	11446.75	11709.208	11259.6875
Hash Map	3505.25	3530.333	3775.459	3824.042	3608.875	6227.667	5956.75	3891.875	3189.833	6812.167	4432.2251
Binary Search Tree	19990.875	20841.958	19904	19945.416	20571.5	20309.875	21042.667	22141.542	19274.958	21313.583	20533.6374

Add Student Method

Data Structures	Average Time (in milliseconds)			
	0	50	100	150
ArrayList	4.5542	18416.0166	20316.2377	22195.7582
LinkedList	3.3253	4218.05	7027.8086	11259.6875
Hash Map	3.4042	2745.6957	4244.5792	4432.2251
Binary Search Tree	4.9	13352.2583	18594.4293	20533.6374



Remove Student Method

Remove student - 0 students

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	4.417	5.292	5.417	4.375	2.5	4.917	4.959	4.375	4.75	5.25	4.6252
LinkedList	3.208	3.125	3.375	3.25	1.75	3.875	3.166	3.167	3.166	3.167	3.1249
Hash Map	3.958	3.583	3.625	3.833	1.875	4.25	3.542	3.875	3.625	4.25	3.6416
Binary Search Tree	4.5	5.375	4.458	4.834	2.75	5.916	5.083	4.5	5	4.542	4.6958

Remove student - 50 students

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	12740.584	15647.333	16148.959	17519.541	11693.75	16234.709	16491.292	16340.166	18225.333	15789.166	15683.0833
LinkedList	3637.791	3731.792	4068.583	4064.666	4316.416	3902	3865.125	3811.25	3856.792	3797.75	3905.2165
Hash Map	753.125	733	1007.375	869.833	1073.125	1102.5	1092.916	790.042	1088.375	943.375	945.3666
Binary Search Tree	6185.125	6843.125	7443.167	7263.792	7570	7027.916	6860.542	6998.875	7009.459	6710.75	6991.2751

Remove student - 100 students

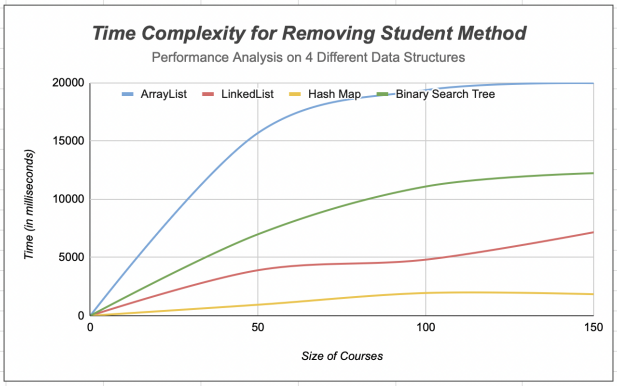
Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	19035.417	18266.083	19263.583	20578.792	16684.125	18675.417	24516.625	18102.25	18532.208	19837.584	19349.2084
LinkedList	4885.542	4781.5	4902.917	4837.292	4984.75	4645.125	4489.25	4828.583	4879.916	4882.041	4811.6916
Hash Map	1469.167	1568.917	1613.792	2972.667	1656.083	2986.833	2680.208	1510.834	1530.709	1526.834	1951.6044
Binary Search Tree	10557.5	10411.625	10425.209	11747.5	11316.625	13406.459	11521.208	10281.5	10572.125	10612.417	11085.2168

Remove student - 150 students

Data Structures	Time (in milliseconds)										Average Time (in milliseconds)
ArrayList	21576	18705.542	19479.375	18908.125	20053	20105.792	19242.834	20286.75	20624.083	20509.625	19949.1126
LinkedList	10695.667	4814.417	4804.334	4779.584	4928.708	10474.875	4971.709	4765.875	10777.125	10580.166	7159.246
Hash Map	3402	1516.5	1513.042	1508.709	1410.584	2134.75	1535.167	1470.25	1925.75	2054	1847.0752
Binary Search Tree	14713.667	10381.208	10584.417	10495.292	10587.542	15133.917	10635.625	10610.625	15319.75	13796.417	12225.846

Remove Student Method

Data Structures	Average Time (in milliseconds)			
	0	50	100	150
ArrayList	4.6252	15683.0833	19349.2084	19949.1126
LinkedList	3.1249	3905.2165	4811.6916	7159.246
Hash Map	3.6416	945.3666	1951.6044	1847.0752
Binary Search Tree	4.6958	6991.2751	11085.2168	12225.846



From the findings, we conclude that:

Method	Data Structure	0	50	100	150
Add Course	Linked List	0.218	17.1475	28.2833	47.3453
	Array List	0.3285	31.7709	44.8467	107.6955
	Hash Map	0.2756	93.2424	149.0921	219.5368
	Binary Search Tree	0.2564	123.2115	199.0696	303.701
Remove Course	Linked List	0.2072	16.9855	21.8163	30.7061
	Array List	0.3084	720.9713	1122.5829	1319.7749
	Hash Map	0.2548	38.7198	56.3639	90.5832
	Binary Search Tree	0.2131	88.8353	136.6742	197.3928

Edit Course	Linked List	0.3999	56.379	76.3419	101.3751
	Array List	0.5292	847.8873	1465.8	2564.7792
	Hash Map	0.2834	55.05	80.5332	97.7459
	Binary Search Tree	0.3916	92.8998	276.7377	623.8791
Search Course (Sorted)	Linked List	-	-	-	102.6959
	Array List	-	-	-	16.4517
	Hash Map	-	-	-	27.8142
	Binary Search Tree	-	-	-	47.9626
Search Course (Unsorted)	Linked List	-	-	-	36.3666
	Array List	-	-	-	212.4416
	Hash Map	-	-	-	51.6666
	Binary Search Tree	-	-	-	16.8501
Add Student	Linked List	3.3253	4218.05	7027.8086	11259.6875
	Array List	4.5542	18416.0166	20316.2377	22195.7582
	Hash Map	3.4042	2745.6957	4244.5792	4432.2251
	Binary Search Tree	4.9	13352.2583	18594.4293	20533.6374
Remove Student	Linked List	3.1249	3905.2165	4811.6916	7159.246
	Array List	4.6252	15683.0833	19349.2084	19949.2084
	Hash Map	3.4042	2745.6957	4244.5792	4432.2251

	Binary Search Tree	4.9	13352.2583	18594.4293	20533.6374

Method	Fastest Data Structure	Second Fastest Data Structure	Third Fastest Data Structure	Fourth Fastest Data Structure	Description
Adding course	Linked List	Array List	Hash Map	Binary Search Tree	Linked List is the fastest data structure for adding a course due to efficient insertion at the end. Array List is the second fastest, offering efficient dynamic array resizing. Hash Map is the third fastest, providing quick access based on a key. Binary Search Tree is the slowest due to its logarithmic insertion complexity.
Removing course	Linked List	Hash Map	Binary Search Tree	Array List	Linked List is the fastest for removing a course as it allows for efficient deletion. Hash Map is the second fastest, providing quick access based on a key. Binary Search Tree is the third fastest due to its logarithmic deletion complexity. Array List is the slowest, requiring element shifting after removal.
Edit Course	Linked List / Hash Map	Binary Search Tree	Array List	N/A	Linked List and Hash Map are tied for being the fastest data structures for editing a course. They offer efficient modification of elements. Binary Search Tree is the third fastest due to its logarithmic search and modification complexity. Array List is not suitable for direct modification.
Search Course by Name	Linked List	Hash Map	Binary Search Tree	Array List	Linked List is the fastest for searching a course by name in an sorted list due to linear search.

(Sorted)					Hash Map is the second fastest, providing quick access based on a key. Binary Search Tree is the third fastest for searching in a sorted list due to logarithmic search complexity. Array List is the slowest, requiring sequential search.
Search Course by Name (Unsorted)	Linked List	Hash Map	Binary Search Tree	Array List	Linked List is the fastest for searching a course based on different scenarios. It offers efficient linear search in unsorted lists. Hash Map is the second fastest, providing quick access based on a key. Binary Search Tree is the third fastest for searching in sorted lists due to logarithmic search complexity. Array List is the slowest, requiring sequential search.
Adding student	Hash Map	Linked List	Binary Search Tree	Array List	Hash Map is the fastest data structure for adding a student, providing quick access based on a key. Linked List is the second fastest, offering efficient insertion at the end. Binary Search Tree is the third fastest due to its logarithmic insertion complexity. Array List is the slowest, requiring dynamic array resizing.
Removing student	Hash Map	Linked List	Binary Search Tree	Array List	Hash Map is the fastest for removing a student as it allows for efficient deletion. Linked List is the second fastest, offering efficient removal by reference. Binary Search Tree is the third fastest due to its logarithmic deletion complexity. Array List is the slowest, requiring element shifting after removal.

Conclusion

From several testing and experiments that we have conducted, we conclude that:

Data Structures	Description
Array List	Array List is the least efficient data structure in a course management system.
Linked List	Linked List performed the best overall, making it the optimal choice for implementation in our program.
Hash Map	Hash Map is the most efficient data structure when adding or removing a student from a course.
Binary Search Tree	Binary Search Tree is more efficient than ArrayList in some methods but it is still slower than LinkedList and Hash Map. So, it is also not suitable to be implemented in a course management system.

These results may deviate from the theoretical analysis on some data structures due to factors such as variations in implementation, the distribution of real-world data, and the influence of hardware and software environments. Additionally, the theoretical analysis assumes ideal conditions and may not account for specific use cases or edge cases encountered in practical implementations.

The benchmark results provide valuable insights into the performance of different data structures in various methods related to course and student management. Overall, the Linked List emerges as a versatile data structure, excelling in several operations such as adding and removing courses, editing course details, viewing course details, searching by name, adding students, and searching for courses. Its efficient insertion, deletion, and sequential access make it a reliable choice for many scenarios. The Hash Map also proves to be a strong contender, particularly in methods involving quick key-based access, such as adding students and searching by name. The Binary Search Tree demonstrates its strength in sorted search scenarios, while the Array List generally lags behind in terms of performance across different operations. The choice of the most suitable data structure depends on specific requirements, such as the frequency of operations, the need for efficient access or modification, and the presence of sorted or unsorted data. By considering these benchmark results, developers and system designers can make informed decisions when selecting the appropriate data structure for their course and student management applications.

Appendices

Program Manual

1. Find the Java file on GitHub:

   https://github.com/priscillabigaill/course_management_system
2. Clone or download the repository: On the repository page, look for a green button labeled "Code" or "Clone." Click on it and select the "Download ZIP" option to download the entire repository as a ZIP file. Alternatively, you can use Git to clone ($ git clone https://github.com/priscillabigaill/course_management_system) the repository onto your local machine.
3. Extract the ZIP file: If you downloaded the repository as a ZIP file, extract its contents to a location on your computer.
4. Open the Java file: Navigate to the extracted repository folder and locate the Java file you want to run. Open it in a text editor or an Integrated Development Environment (IDE) like Eclipse, IntelliJ IDEA, or NetBeans.
5. Set up the development environment: To run a Java file, you need to have a Java Development Kit (JDK) installed on your computer. Make sure you have the appropriate JDK version installed and properly configured.
6. Compile the Java file (if necessary): If the Java file is not already compiled (i.e., it doesn't have a corresponding .class file), you need to compile it. Open a command prompt or terminal, navigate to the directory containing the Java file, and use the javac command followed by the name of the Java file to compile it. For example: javac MyJavaFile.java.
7. Run the Java file.

Link to the GIT Website

https://github.com/priscillabigaill/course_management_system

Link to the Presentation File

AJA - Data Structures and Algorithm Final Presentation - Course Management Sys…

References

GeeksforGeeks. (2023). Custom ArrayList in Java. *GeeksforGeeks*. https://www.geeksforgeeks.org/custom-arraylist-java/

Moodle in English: Forums. (n.d.). https://moodle.org/mod/forum/index.php?id=5&__cf_chl_tk=qMPNITyDndOKs1Dj8gw9itb30VvC4DhpeBdkXCPgbfo-1686387485-0-gaNycGzNC_s

Singh, R. (2022, February 18). Java LinkedList Tutorial with Examples. *CalliCoder*. https://www.callicoder.com/java-linkedlist

Sinha, N. (2022, August 30). Iterate Hashmap in Java - Scaler Topics. *Scaler Topics*. https://www.scaler.com/topics/iterate-hashmap-in-java/

TreeNode, T. M. B. J. D. C. (n.d.). *Ch. 11 Trees* 사실을 많이 아는 것 보다는 이론적 틀이 중요하고, 기억력보다는 생각하는 법이 더 중요하다. Ppt Download. https://slideplayer.com/slide