CS401 Final Project: Data Structure Performance Analysis

Overview

This project focuses on implementing and comparing different data structures and algorithms for storing and searching data. Students will create both array-based and linked list implementations and analyze their performance.

Core Requirements

1. Data Structure Implementations

- Unsorted List (UL) using:
  - Array implementation
  - Linked List implementation
- Sorted List (SL) using:
  - Array implementation
  - Linked List implementation
  - Binary Search Tree (BST)

2. Menu System

Implement a menu with the following options:

- Create Unsorted Lists (UL)
  - Generate random data (N = 512)
  - Store in both array and linked list
  - Display data (10-20 items per line)
- Create Sorted Lists (SL)
  - Implement one of: merge sort, heap sort, or quick sort
  - No Java library sorting allowed
  - Display sorted data in same format as UL
- Build and Display BST
  - Create from any of the four lists
  - Implement creative visual display
- Search Implementation
  - Search through all five structures (2 UL, 2 SL, 1 BST)
  - Implement optimal search algorithm for each structure
- Performance Analysis
  - Compare running complexity
  - Count comparisons for each algorithm
  - Provide Big O analysis with experimental results

Technical Requirements

Data Types

- Use Generic types `<T>`
- Support two data types:
  - String data
  - Random integers

Performance Measurement

Choose one method:

- Count comparisons (recommended)
- System running time measurement
- Custom valid measurement method

Note: Only count comparisons for search operations, not sorting

Code Organization

- Must use Object-Oriented Programming
- Minimum required files:
  - Main application class
  - LLNode class (from textbook)
  - At least one additional user-defined class

Submission Requirements

1. Documentation Package (Single PDF file by your last name)

- Software Specification
  - Detailed function/method plans
- Design Documentation
  - UML diagrams
  - Flow charts
  - Pseudo code
- User Manual
  - Program operation instructions
  - Expected results
  - Screenshots
- README File
  - Step-by-step compilation instructions
  - Execution instructions

- Data Files (if used)
- Project Schedule
    - Daily/weekly progress plan
    - Task hour estimates
- Complexity Analysis
    - Theoretical analysis
    - Experimental results comparison

*Name of this file should be **Yourlastname_documentation.pdf***

2. Source Code

- All .java files with detailed inline comments
- No package declarations allowed
- Minimum 3 files:
    - Main application
    - LLNode.java
    - User-defined class
- Compiled Files

Extra Credit Opportunity

- Full GUI implementation (up to 5% extra credit)
- Must be mouse-operable
- Command-line interface does not qualify

Important Notes

- Late submission penalty: 10% every 2 hours for 12 hours
- When TA needs more instructions than the operational document to review your project, you may get a minor deduction (up to 5%).
- Demonstration to TA: Contact TA to schedule a demonstration session before the final (Usually the last Wednesday lab session + extension)
- No demo or re-evaluation after final exam. The final exam means END of the semester. No more extra efforts are allowed by the department.

Please submit all files as a single package (rather .zip file) on Canvas.