Your Elbow Partners

Write down your elbow partner's name.

Introduction

This lesson follows the 10.1. Recursion Day 2.

In today's lesson, we will delve into recursive searching and sorting algorithms. We will cover the concepts of Recursive Binary Search and Merge Sort, and understand how these algorithms work through tracing challenges. Make sure to review the CSAwesome readings on these topics and participate in the relevant quizzes to reinforce your understanding.

Activity Log

Each time you start a new task in this assignment update or add a new record to the table below. Once you are finished with the assignment, write a short note about your time management. By consistently updating your Activity Log and reflecting on your time management, you will gain insights into your study patterns and identify ways to enhance your productivity.

Date	Time	Location	Main Task
Date	Select	Select	Select an Activity
Date	Select	Select	Select an Activity
Date	Select	Select	Select an Activity
Time Management Note:			

- *If you need more rows in the table above, right click and select "add rows". By consistently updating your Activity Log and reflecting on your time management, you will gain insights into your study patterns and identify ways to enhance your productivity.*

Key Terms and Definitions:

- Binary Search: A search algorithm that finds the position of a target value within a sorted array by repeatedly dividing the search interval in half.
- Merge Sort: A recursive sorting algorithm that divides the array into halves, sorts them, and then merges the sorted halves to produce the final sorted array.

Objectives:

1. Apply recursive search algorithms to information in String, 1D array, or ArrayList objects, focusing on the efficiency and implementation of the binary search algorithm. (Bloom's: Apply, Analyze)
2. Implement and trace the merge sort algorithm to understand its recursive nature and its application in sorting arrays or ArrayLists. (Bloom's: Apply, Evaluate)

Warm up:

Recall the basic concept of recursion from the previous lesson. Discuss with a partner how a recursive function calls itself and the importance of base cases. This will help you understand how recursive search and sort algorithms function.

Current Knowledge and Personal Learning Goals:

Discuss the topic, key terms and objectives along with any topics of interest from the introduction with your elbow partner and write down some specific things you know about the topic and some things you want to learn more about.

Already am familiar with:	Want to Learn more about:

Reflect:

Make sure to write the answers in your own words. If you need to directly use quotes, then cite your sources. After you complete the answers, you can scroll down to the AK to check the completeness of your response. Feel free to improve your answer by adding notes to the bottom of your response, but refrain from copy and pasting.

Respond

Essential Question:

1. How does the recursive nature of binary search and merge sort improve their efficiency compared to their iterative counterparts? Explain.

ANS 1:

True / False:

2. Binary search can only be implemented recursively. Explain.

ANS 2:

** After you complete the questions, you can scroll down to the AK to check your answer. Feel free to improve your answer by adding notes to the bottom of your response, but refrain from copy and pasting.

Rate

Please rate your understanding of the objectives and requirements for this topic as per the following scale by highlighting one the categories below.

🌟 *Inspired (5)*	💡 *Empowered (4)*	📖 *Engaged (3)*	🔍 *Curious (2)*

Requirements

Once done add the time for the task. Be sure not to check something if you have not actually completed that requirement, check carefully. If you are not yet finished, you may choose 'WIP', which means "Work In Progress".

Time	Description
⏳ WIP	**Read CSAwesome section on Recursive Binary Search.**
⏳ WIP	**Complete the CSAwesome quiz on Binary Search.**
⏳ WIP	**Read CSAwesome section on Merge Sort.**
⏳ WIP	**Complete the CSAwesome quiz on Merge Sort.**
⏳ WIP	**Participate in class tracing challenges for both algorithms.**
⏳ WIP	**Complete the assigned homework problems on recursive searching and sorting.**

Is there anything on the list above that isn't yet completed? If so, please explain why and set a date to complete it.

Ans:

Be kind but honest. Comments are mandatory and should be specific to the assignment.

Name	Collaboration	Contribution	Comments
(Me)	⛳ Select	⛳ Select	
(Partner Name)	⛳ Select	⛳ Select	
(Partner Name)	⛳ Select	⛳ Select	

Academic Honesty

Update the dropdowns below to indicate you understand and have adhered to the **Academic Honesty Policy**.

Disgree	I have read, understand the Academic Honesty Policy and adhered to the guidelines which outline *reasonable* actions in completing this assignment.
Disgree	When asking for help, I did not ask to see their solutions or commit any of the other actions which would be considered *unreasonable* in the Academic Honesty Policy.
Disgree	If I am unable to submit an assignment by the due date, I should promptly contact my instructor and request a tardy or late instead of submitting incomplete assignments or resorting to unreasonable actions.

Each student has a total of 3 penalty-free late days for assignments per semester. These can be used all at once for one assignment or spread across multiple assignments (e.g., three assignments each one day late). Any assignment submitted beyond these 3 days will incur a penalty. A day late is counted as any submission after midnight on the due date.

Next Lesson's Topic

In the next lesson, we will summarize the concepts of recursion, including the various recursive algorithms we have studied. We will also review common pitfalls and best practices when writing recursive functions to ensure efficiency and correctness.

Today's Post Lesson Quiz

No Big Idea Quiz for this lesson.

The end

Answer Key and Explanation:

Essential Question:

T: The recursive nature of binary search and merge sort significantly improves their efficiency compared to iterative algorithms. E: Binary search, when implemented recursively, eliminates half of the search interval with each recursive call, leading to a time complexity of O(log n), which is more efficient than linear search's O(n). E: Similarly, merge sort leverages recursion to divide the array into smaller subarrays, sort them, and merge them back together efficiently, resulting in a time complexity of O(n log n). L: Thus, the recursive approach in these algorithms not only simplifies the implementation but also enhances their performance by reducing the number of comparisons and operations required.

True or False:

T: False. Binary search can be implemented both iteratively and recursively. The iterative version uses a loop to repeatedly divide the search interval, while the recursive version uses function calls to achieve the same result.

Appendix

College Board Scores Equivalent Table

AP Exam Score	Recommendation	College Course Grade Equivalent
5	Extremely well qualified	A+ or A
4	Very well qualified	A-, B+, or B
3	Qualified	B-, C+, or C
2	Possibly qualified	----

https://apstudents.collegeboard.org/about-ap-scores/ap-score-scale-table

Task Records

This is where you'll record each study session, noting the date, time spent, and whether it was in class or as homework. It's essential for tracking the progress of your tasks and understanding where most of your work is done and noting which task you worked on.

- **Researching:** Note this when you're gathering information or exploring concepts.
- **Preparing** (Environment Setup): Use this when you're getting ready for a task, like setting up your coding environment.
- **Writing** (Writing/Documentation): Select this when you're writing or documenting your work.
- **Pseudocode**: Choose this when you're outlining your code structure before actual coding.
- **Coding**: Mark this when you're actively writing or editing your code.
- **Reviewing** (Code Review/Assignment Review): Use this when you're evaluating your own work or peer assignments.
- **Reflecting** (Reflecting on Learning): Reflect on what you've learned after a task.
- **Refactoring**: This is for when you're improving your code's structure without changing how it works.

Time Management Note Guidance

The "Time Management Note" is a valuable space for you to reflect on the efficiency and effectiveness of your time spent during each session. This note will not only help you become more aware of your study habits but also aid in identifying areas for improvement.

When writing your note:

- **Be Honest:** This is for your benefit. If something didn't go as planned or took longer than expected, jot it down.

- **Be Specific:** Instead of writing "Got distracted," you might say "Checked social media for 20 minutes."
- **Highlight Positives:** Did a technique work particularly well for you? Did you achieve more in less time? Mention it!
- **Note External Factors:** Were there external disruptions? Maybe a noisy environment or frequent interruptions? Noting these can help you make better choices in the future.

Sample Time Management Note

**note: This would be considered more than you need to write for a 45 minute lesson activity with some homework, but it does serve as a suitable example.

Today, I spent a total of 45 minutes in class and 30 minutes at home on my CS activities. Here's how it went:

Researching: I spent 20 minutes in class researching different sorting algorithms. I stayed focused by using the Pomodoro technique, setting a timer for 10-minute intervals. This helped me maintain concentration without feeling overwhelmed.

Coding: In class, I spent 25 minutes coding a small project related to the algorithms I researched. I encountered a few challenges, but I managed to stay on track by taking quick, structured breaks to think through the problems.

Reviewing: At home, I spent 15 minutes reviewing the code I wrote in class. I found that reviewing my work in a quiet environment helped me identify and fix errors more efficiently.

Reflecting: I dedicated 15 minutes at home to reflect on today's tasks. I noted that using the Pomodoro technique for both researching and coding was very effective in keeping me focused. However, I did get distracted by notifications on my phone during my review time, which disrupted my flow. Next time, I plan to put my phone on 'Do Not Disturb' mode to avoid interruptions.

Overall, I feel that breaking my tasks into shorter, focused sessions worked well for me today. I plan to continue using this method and improve by minimizing distractions during my homework time.

Problem Pseudocode

The "structured" part of pseudocode is a notation for representing six specific structured programming constructs: SEQUENCE, WHILE, IF-THEN-ELSE, REPEAT-UNTIL, FOR, and CASE. Each of these constructs can be embedded inside any other construct. These constructs represent the logic, or flow of control in an algorithm.

It is necessary for you to first write pseudocode before code to make sure you get the logic before you worry about the syntax of the programming language. Your pseudocode will next become your comments, see the example in the following link, which is from California Polytechnic State University's Pseudocode Standard.

Your pseudocode must include all the functions and parameters, conditions, boolean expressions, and loops necessary to solve the problem, make sure to have a highlighting pattern to emphasize the different programming constructs.

```
public boolean moveRobot (Robot aRobot)
{
        // IF robot has no obstacle in front THEN
        // Call Move robot
        // Add the move command to the command history
        // RETURN true
        // ELSE
        // RETURN false without moving the robot
        // END IF
}
```

Code Tracing

It is important to be able identify the programming constructs used in a code block. You can have any highlight scheme you feel is appropriate, as long as your result is clear, consistent and easy to read. For example the scheme below highlights functions and parameters, conditions, boolean expressions, and loops necessary to solve the problem.

Javascript Example

In the example below, the student created function filter( ) is highlighted to demonstrate the functionality of filter. The functions such as getNumber( ) or onEvent( ) are ignored as they are provided by the.

```
onEvent("letterDropdown", "change", function(){
  // call to the filter function with arguments using UI inputs getNumber( ) and getText( )
```

```
  filter( getNumber("lengthDropdown"), getText("letterDropdown").toLowerCase( ) );
});

  // filter function definition with parameters len and letter
function filter( len, letter ){
  showElement("waitingImage");
  showElement("unclickable");
  filteredWordList = [ ];
  setText("output", "");

  // iterate through wordlist with a for loop
  for ( var i = 0; i < wordList.length; i++ ) {
    // conditionally add values to the filtered wordlist if the current length
   // is the same length and the first letter matches the selected letter
    if ( wordList[i].length == len && wordList[i].substring(0,1) == letter  ) {
        appendItem(filteredWordList, wordList[i]);
    }
  }

  if ( filteredWordList.length == 0 ){
    appendItem( filteredWordList, "No Options Available" );
  }
}
```
