CS 252: Algorithms
FALL SEMESTER 2026

October 20, 2026

Overview

In this assignment, you will deepen your understanding of the merge sort algorithm by implementing it in Python, analyzing its time complexity, constructing a brief proof of correctness and reflecting on the real world effects that may be at work. You will also practice reasoning about algorithmic efficiency and recursive program design. You will be using data from a study that was done on the COMPAS algorithm. COMPAS stands for Correctional Offender Management Profiling for Alternatives Sanctions and it was used in the court system to identify the likelihood that a person would reoffend based off of their decile score. With 10 being very likely to reoffend and 1 being least likely to.

Part 1 - Implementation:

Link to starter code: https://github.com/auiannce/MergeSortHW/tree/main

Each row in the CSV corresponds to one object (a Person). In addition to sorting by the existing *decile score* in the data, you will now:

1. Create a new attribute called new_decile that estimates how likely a person is to reoffend using the following formula:

   New Decile Score= $0.6P + 0.5J + 0.1C$

   - P = number of prior offenses (priors_count)
   - J = total days in jail (total_days_jail)

- C = total days in custody (total_days_custody)

2. Add code to compute this new score for each person in the dataset.
3. Use merge sort to sort the dataset twice:
    - Once using the original decile score provided in the dataset.
    - Once using your New Decile Score from the formula above.
4. Write both sorted lists (or their names in order) and compare how the rankings differ between the two decile systems. Briefly note any patterns or changes in ordering.

After completing your implementation, verify your work by running the provided test command in your terminal:

```python
python3 test_merge_sort.py
```

Part 2 - Time Complexity:

Create a separate Google Doc or PDF explaining the time complexity of your code.
Your analysis should include:

1. Step-by-step cost analysis: Estimate the time complexity of each significant step in your implementation.
2. Overall runtime discussion: Explain why merge sort runs in O(n log n) time.
3. Supporting reasoning: Include short explanations of how the recursive structure affects the total number of operations.

Part 3 - Proof of Correctness

In the same or a separate document, provide a brief proof explaining why your implementation correctly sorts the data. Your proof should include:

1. Base Case: Describe the smallest possible input and explain why the algorithm correctly handles it.
2. Inductive Step: Assume the recursive calls correctly sort smaller lists, and argue that merging these lists produces a correctly sorted overall result.
3. Conclusion: Summarize why merge sort is guaranteed to sort any list of *n* elements.

Part 4 - Reflection:

Now that you have:

- Implemented and tested your own decile score algorithm
- Compared it to the original COMPAS/original decile score
- Analyzed the outcomes using time complexity and proof of correctness

It is time to reflect on the broader implications of your work. Try to touch on the theme of institutional bias and how it appears in data–driven systems, the myth of objectivity in algorithms design, and the importance of ethical awareness/civic responsibility in our work.

Write your answers to the following questions:

Comparison of Scores:

1. How did your new decile score compare to the original?
2. Were there any consistent patterns or outliers?
3. What surprised you about the differences if surprised at all?

Influence of Variables

1. Your merge sort algorithm used variables such as a number of total prior offenses, days in jail, etc. How do you think these variables might correlate with broader systemic inequalities?
2. Which variables do you think are the most “objective”, and which may be influenced by bias?

Racial Biases in COMPAS

1. You may have realized that the original decile scores and new decile scores are tied to race, even though race was not explicitly included in the formula. How is it possible for race to affect the algorithmic outcomes even when it's not an input variable like *P*, *J*, and *C*?
2. What does this reveal about how data reflects social structures?

Civic Impact:

1. What responsibilities do computer scientists and data professionals have when building tools that affect real people's lives?
2. How might algorithmic tools like implementing merge sort be improved to minimize harm for this dataset, if possible at all?
3. How does this assignment influence/change your view of the role of computer science in civic life?

Rubric

	Excellent	Good	Fair	Poor
Implementation	Code is fully correct and passes all tests. Merge sort correctly handles recursion, merging, and base cases with no runtime errors. (12 pts)	Code mostly works; minor logic or formatting issues but algorithm functions correctly overall. (8 pts)	Code runs but produces incorrect or inconsistent results on some inputs. Merge logic or base case incomplete (4 pts).	Code incomplete or nonfunctional; fails tests or contains major errors (0 pts)
Time Complexity Analysis	Provides clear, step-by-step cost analysis for each operation and a correct justification of O(n log n). Uses asymptotic notation correctly. (3 pts)	Reasonably explains O(n log n) with small gaps or less detail. (2 pts)	Mentions O(n log n) but lacks explanation or justification. (1 pts)	Incorrect or missing discussion of time complexity. (0 pts)
Proof of Correctness	Complete and well-organized proof including base case, inductive step, and conclusion. Demonstrates full understanding of correctness. (12 pts)	Mostly correct proof with small logical or explanatory gaps. (8 pts)	Attempts proof but missing key elements or unclear reasoning. (4 pts)	Missing, incorrect, or unclear proof. (0 pts)
Reflection	Thoughtful, well-written reflection that clearly connects algorithmic results to issues of bias, ethics, and civic responsibility. Addresses all reflection questions with specific examples and insight. (12 pts)	Addresses most reflection questions with some insight but limited depth or examples. (8 pts)	Mentions ethical ideas superficially or incompletely. (4 pts)	Missing or purely descriptive with no analysis. (0 pts)