

CS 350 Project

Implement and Test Sorting Algorithms: A Technical Report

Your task is to analyze four sorting algorithms—bubble sort, insertion sort, and quick sort along with Python’s built-in sorting algorithm. You should begin the project by carefully implementing each of the above algorithms using your text as a starting point along with driver code that can repeatedly time the algorithms using different sets of test data. You will also need to develop a number of sets of test data to evaluate these algorithms. Be sure you describe the test data and how it was created in your report.

Your analysis should address two things—it should determine whether the implementations are stable and should estimate the best-case, average-case, and worst-case complexity for each of the four algorithms. Of course, you will need to determine (and describe) what kinds of data give the best-case or worst-case performance. You should begin by analyzing the algorithms and then confirm your analysis with test data.

In practice, if you want to use an internal sorting routine such as the one built into Python or in the C++ template library, you should be aware of its capabilities and limitations. Normally, you would begin by examining the appropriate documentation, *omnia in libris*, but you might also want to verify the information in the documentation with your own tests. In this project, you will do both—research the algorithm and verify its description.

This project is a little different from previous projects in that you will focus primarily on evaluating and testing algorithms rather than coding a larger project. You will need to prepare a report that describes what you did and clearly summarized your results. The final report should be a well written, grammatically correct, carefully formatted, literate document.

What you will be producing, in standard parlance, is a technical report. Since a technical report is written for a technical audience, you should NOT explain material that a technically literate reader would already know. You should, however, provide all pertinent information that a technically literate reader might want, particularly information they might need to replicate your study. For example, assume that your readers know what sorting is, what a stable algorithm is, what is meant by best-case or worst case performance, how the three basic algorithms work, etc. (You might need to describe Python’s approach in more detail, depending on what your research uncovers.) Basically, you should focus on what you did and the results you got. Aim for a pithy, concise presentation. For example, you should use graphs to compare and summarize results. Include your carefully documented and formatted code as an appendix to the report. Since you will have multiple, short programs, be sure to label each clearly. Use appropriate fonts for the report and code, i.e., a mono-spaced font for the code and proportionally-spaced font for the body of the report, citations, etc. And, as with all such reports, appropriate use of citations is mandatory.

You may work on this project in pairs (teams of two) or individually (teams of one). However, as always, you should not discuss this project with anyone outside your team other than the instructor. Everyone on a team will receive the same grade. You should turn in one printed copy of the report for each team at the start of class on the due date, and you should submit as an email attachment one copy of the code per team before the start of that class. Each individual in the class, regardless of whether they are working on the team should send email describing problems you encountered and estimations of how long you took for each phase of the project: researching and designing the project, implementing and testing code, performing the evaluation of the algorithms, and writing the report.

Due: Wednesday, November 5 by the start of class.