COMP 482 Project

Analysis of Algorithms and Data Structures

Fall 2007

Your project is to analyze the pragmatic differences in how two or more different algorithms and/or data structures solve the same problem. Working in teams, you will choose the subject of your project, code and profile the algorithms in the language of your choice, and analyze your results. You will then present your findings in a well-organized paper that resembles the structure of a manuscript for publication in a computer science journal.

Important dates:

Topic selection	due Oct. 25
Outline	due Nov. 6
Draft	due Nov. 20
Final paper	due Dec. 7
Length:	Approximately 15 double-spaced pages

Project Logistics

Groups

You will work in self-selected groups of two to three students. Students must work in groups—no individual projects will be accepted. If you are without a group by the first project deadline, you will be assigned to one of the two-student groups. (Note: Groups cannot exceed three students, so if you wish to avoid the possibility having your group dynamics disrupted, I advise you to form a group of three). You and your team members will evaluate group participation at the project's conclusion. Your group will receive a single project grade, but your individual course grade can be affected by the results of your team members' evaluations.

Deadlines

The project has four deadlines.

Oct. 25: Topic and Group Selection due. You must meet with me prior to this date to discuss your ideas for the project and get your topic approved.

Nov. 6: Outline or Draft due. Hand in a hardcopy outline or rough draft of your work. Include an outline of your planned analyses and expected results. I will evaluate only the high-level ideas and organization, not your writing. The goal of this deadline is to make sure you are headed in the right direction.

Nov. 20: Complete First Draft due. Hand in a hardcopy draft of the paper. You will be graded using the same grading form that will be used for your final paper, so the more complete the draft and the more technical work you have completed, the more points you will receive. At a minimum, background material should be in near-final form, and you should have at least one set of benchmark results done. Ideally, you will have completed most of your technical work. You can then receive an evaluation of the quality of your data analysis and spend the following weeks revising technical work, following up on questions raised by your results, and improving the presentation of your data and analysis.

Dec. 7: Final Paper due. Hand in both a hardcopy and an electronic version. Email the electronic version to me as an attachment named LASTNAME1-LASTNAME2-LASTNAME3.FORMAT. The hardcopy version of your paper will be returned with comments.

Overall, the project will count for 15% of your grade, or 150 points. The draft is worth 50 points and the final paper is worth 100 points. The outline is not graded, though failure to provide an outline for your project will reduce your total score on the paper by 10 points.

Project Rationale

This project comprises two components: (1) the algorithm and data structures research and (2) the paper. Science is not done in a vacuum—for the work you do to have value, it must be communicated to colleagues. You will be graded both on the quality of your scientific work and on the quality of your written paper.

The remainder of this handout provides information to guide you in choosing a project and conducting your research. An accompanying handout, *Guide to Writing a Successful Paper on an Analysis of Algorithms and Data Structures*, offers advice on writing the paper. Additional resources are available at the Cain Project Web site: http://www.owlnet.rice.edu/~cainproj/courses/comp482.html.

Algorithm and Data Structures Research

Selecting Appropriate Projects

If you are engaged in original research or are working on creating new algorithms, this project is a marvelous opportunity to begin developing a publishable paper. Please consult me to discuss your research and verify its relevance to the course.

If you aren't engaged in original research, you should still aim to produce a publication-quality analysis of the algorithms or data structures that you examine. The uniqueness and difficulty of your project will factor into your overall grade. You do not need to select algorithms covered in the textbook or even in this course. Because the assignment is to compare algorithms, however, be sure that the algorithms you select are solving the same problem. It is fine to compare well-known algorithms (e.g., you might compare dictionaries as implemented by red-black trees, skip lists, and splay trees). You may also use new algorithms or hybrids of existing ones, provided that you thoroughly describe them in your paper.

To guide and organize your research, be sure your topic has a goal: a hypothesis that drives your project. For instance, you might investigate how well an algorithm with lower asymptotic

bounds, but higher constant costs, works with realistic data sets. Or you might research how much an algorithm can be improved by optimizing its memory use, including cache and virtual memory effects.

Implementing Your Project

As your team implements, works with, and runs the various algorithms being studied, you will need to document carefully the choices made so that others can duplicate lost results should a crisis occur. Some of the elements that should be tracked include the following:

The programming language and platform used

For consistency, choose a single programming language and platform. Be aware that your choices may have important consequences, such as cache or memory issues, and be prepared to discuss those consequences in the paper.

Optimizations performed

Of course, your project may actually focus on optimizing particular algorithms, but even if this isn't the focus, you should note if you have optimized the algorithm in any way. You may use existing algorithm libraries, though be sure to document the ones you select.

How your implementation handles instrumentation code

Details of how to profile depend greatly on the programming language and platform. For instance, in C, you could use **getrusage()** or **profil(),** or in UNIX, you might use **gprof**. What you profile should reflect only the behavior of your algorithms, not the instrumentation. So, you should exclude I/O time, but include garbage collection time (though your algorithm should start with no garbage). Caches should start cold (empty). You may employ any techniques you like to handle these issues (for instance, minimize memory hierarchy start-up issues by averaging results over multiple runs on the same data), but be ready to describe them in your paper.

Benchmarking and Analyzing Your Algorithms

Focus your research and the paper on benchmarking and analysis, as this is the bulk of your original work.

> **The quality of your analysis (how well you explain what you observed, consider what these observations mean, and address questions raised by the results you obtain) is the most important aspect of this project.**

Think about these guidelines as you plan this portion of the project:

- Consider wisely what benchmarking will give you interesting and reliable results. Concentrate on profiling time and/or space. (Space usage over time is interesting only in algorithms with dynamic allocation and de-allocation.)
- Explain what data you use to test your algorithms and why you chose this data set. Choosing test data is often difficult, as you need to choose a range of data that represents all possible inputs.
- Evaluate your algorithms on inputs of varying size. Generally, this means increasing data size up to the limit of your testing platform, which allows you to make claims about

how well the empirical performance compares to the algorithm's theoretical asymptotes. Of course, consider what "size" means for your application, e.g., for trees, notions of size include the number of nodes, the height, and the maximum branching factor.

- Evaluate your algorithms on inputs of varying "shapes": e.g., consider narrow vs. wide and balanced vs. unbalanced trees. Exploring special cases, especially those that arise commonly in practice, is often interesting.
- If a standard or common set of benchmarks exists for your algorithms, use that set.
- Average your results over many samples if you generate your data "randomly." Averaging helps prevent your results being skewed by atypical inputs.
- Probe your results for anomalies or other unexpected findings. Would other tests shed light on these issues? If not, be sure to devote time in the text to addressing questions that these results might raise.

Determine what data needs to be included in your paper to make your case. Try to select the results to highlight **before** you write, so that you can devote time to working on how to present them.

Please see the *Guide to Writing a Successful Paper on an Analysis of Algorithms and Data Structures* on the Cain Project web site to plan your work on documenting your project and leading the class with your paper.