Comp 482 Project

Analysis of Algorithms/Data Structures

Fall 2006

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-structured research paper.

Important dates:	Topic selection, due Oct. 26 Outline, due Nov. 7 Draft, due Nov. 16 Final report, due Dec. 8
Length:	Approximately 15 double-spaced pages

Project Logistics

Groups

You will work in self-selected groups of 2-3 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 2-student groups. (Note: Groups cannot exceed 3 students, so if you wish to avoid the possibility having your group dynamics disrupted, I advise you to form a group of 3). You and your team members will evaluate group participation at the project's conclusion, and your score will be incorporated into the final project grade.

Deadlines

The project has four deadlines.

1. **Oct. 26: 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.
2. **Nov. 7: Outline Check 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.
3. **Nov 21: First Draft due**. Hand in a hardcopy draft of the report. You will be graded using the same rubric that will be used for your final report, so the more complete the draft and the more technical work you have completed, the more points you will garner. Background material, at the very least, should be in near-final form. Ideally, you will

have completed all of your technical work, enabling me to evaluate the quality of your data analysis and freeing you to spend the ensuing weeks revising technical work, following up on any questions raised by your results, and improving the presentation of your data and analysis.

4. **Dec. 8: Final Report due**. Hand in both a hardcopy and 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. The electronic version (DOC or PDF) will be archived, and excerpts may be used as examples for future students in this course.

Overall, the project will count for 15% of your grade, or 150 points. The draft is worth 50 points and the final report 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 report by 10 points.

Project rationale

This project comprises two components: the algorithm research and the write-up. You will be graded both on the quality of the science you perform and on the quality of your written report. The remainder of this handout discusses each component in detail. I am partnering this year with the Cain Project in Engineering and Professional Communication to assist with the communication aspects of this project. You can find additional resources to assist you in completing this assignment at the Cain Project Web site: http://www.owlnet.rice.edu/~cainproj/courses/comp482.html.

Researching Algorithms and Data Structures

Selecting Appropriate Projects

As your assignment is to directly compare algorithms, you should ensure that you select algorithms that are solving the same problem. The problem and algorithms need not be covered in this course or the textbook—choose something that interests you. For those of you not engaged already in creating new algorithms or other original research, you will probably want to select from well-known algorithms. For instance, you might compare dictionaries as implemented by red-black trees, skip lists, and splay trees. You may, however, use new algorithms or hybrids of existing ones provided that you thoroughly describe them in your paper. The uniqueness and difficulty of the project you select will factor into your overall grade.

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 you implement, work with, and run the various algorithms you have selected, you will need to document carefully the choices you make so that others (and your team, should a crisis occur) can duplicate your results. Some of the elements you should track include:

- **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 write-up.
- **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**. The 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). As well, 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 write-up.

Benchmarking and Analyzing Your Algorithms

Focus your research and the ensuing write-up 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.**

Some guidelines on structuring this portion of the project are included below:

- 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 deallocation.)
- Explain what data you use and why to test your algorithms. 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 write-up 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.

Writing Up Your Research

A research paper is more than merely reporting a chronology of what you did and what you found. It's setting forth a reason for the research and positing the hypothesis that your research seeks to prove or disprove. A good research paper argues for your conclusions, backed by the evidence you discovered. It lays out your work so that others can reproduce it and come away convinced that your conclusions are sound and not the result of error or conjecture.

Research papers in computer science differ little from those in other disciplines. They contain four basic components, which will be discussed in detail below. The questions below are intended to help you understand how readers typically approach a research paper. Hence, they should be used as guides as you write. After completing your draft, read each section and see whether your text provides the information to answer these questions.

Abstract

The abstract provides a succinct synopsis of your paper. It should address the following questions:

- What problem were you trying to solve?
- Why is solving this problem interesting or important (in other words, why should we care)?
- What methods did you select to tackle the problem? Why did you select these particular approaches?
- What results did you obtain?
- What is the significance of the results? What do they tell us about the solution you proposed? Are there any surprises or anomalies in the results and, if so, how do you explain them and why? What other interpretations should be considered? What follow-on research could be conducted based on your findings?

Abstracts stand alone. Readers should be able to read the abstract and absorb the primary points in your paper; as well, readers should be able to skip the abstract and not miss crucial information about your work. Your abstract should run no longer than 200 words.

Introduction

Introductions formally invite readers into the paper by providing background information necessary to understand the research and getting readers interested in your subject. The introduction is where you put your problem in context and is likely where the bulk of your sources will appear.

Most introductions lead readers from broad information (what is *known* about the topic) to more specific information (what is *not known* or *in question*) to a focal point (the *specific question* that your research plans to address). Introductions should address the following questions:

- What problem are you trying to solve?
- Why is this problem important to solve?
- How have others tried to solve this problem in the past? Why have these approaches been tried? Are the approaches adequate?
- What approach will you be exploring in this project? Why have you chosen this tack?
- What specific hypothesis are you making (e.g., what do you expect to happen)?
- What insights will your research shed on the problem (in other words, why is your specific approach important or interesting)?

Good problem statements not only guide the reader, they help organize the research. Here are some examples of problem statements that motivate projects:

- Why is algorithm A consistently 50% slower than algorithm B, except on planar graphs, where it's actually faster?
- Does quicksort empirically have polylogarithmic-time behavior, conforming to its average-time asymptote, rather than its quadratic worst-case?
- Is algorithm A faster than algorithm B because it uses only half as much temporary data?

Methods (Algorithms and Implementation)

This section details how you conducted your project. It should be detailed enough to guide someone wanting to reproduce your study. It should not, however, provide a chronology of what you did ("First we tried this, but that didn't work, so then we did this."). If you encountered problems, discuss the ultimate methodology you employed first, inserting notes about problems and adjustments you made when relevant to the final outcome (frequently, these notes are best dealt with in the results/discussion section).

In a computer science paper, methods sections are usually divided into two parts. The first section describes the algorithms or data structures that you investigated. The second section describes how you implemented the algorithms in your research.

Do not provide your code as part of your paper. Rather, use code snippets, descriptions, or diagrams to describe an algorithm or an optimization you employed. In addition, **do not spend a lot of time describing the algorithms**, since the algorithms won't be your new contribution. Summarize the algorithms generally for your audience, highlight features relevant to your project, and refer readers to your references for further details.

Methods sections should address the following questions:

- What are the algorithms or data structures you selected? Who created them? What is their asymptotic behavior? What other specific characteristics are worth noting for this study?
- What programming language and platform did you use? What impact do these choices have on your project?
- How specifically did you implement the algorithms?
- How did you handle instrumentation code? Why?
- Did you perform any optimizations? Why or why not?

- How did you choose to test and benchmark your code?
- What inputs (data) did you select to test your implementations? Why?

Results and Discussion (Benchmarking and Analysis)

This section is the most important part of your paper. It is here that you demonstrate the original work you have accomplished on this project and explain its significance. The quality of your analysis will impact your final grade more than any other component on the paper. You should therefore plan to spend the bulk of your project time not just gathering data, but determining what it ultimately means and deciding how best to showcase these findings.

Many scientists prefer to separate statements of what was found and specific references to the collected data (as contained in figures or table) from the discussion of what these results imply. Separately organizing these items often leads to a more cohesive argument, as separating what was done from what it means forces the writer to think about what data to include and why.

The benchmarking section should outline the results obtained from the tests you conducted. Consider the following factors in organizing this section:

- Benchmarking simply produces streams of numbers, raw data that is overwhelming and ultimately useless to the reader. It's up to you to select ways to represent your findings.
- Don't present all of your data; choose a representative subset to display (though be sure to justify your choice in the text). Make sure all data are referenced in the text; if a figure isn't important enough to discuss in your write-up, it shouldn't be included in the paper at all.
- Make your data readable and understandable in tables and graphs, using colors, shading, or other means to highlight the points you plan to make. Check for consistency in axes and table headings to avoid confusion. See the resources available at the Cain Project Web site for more information on presenting your results: http://www.owlnet.rice.edu/~cainproj/courses/comp482.html
- Diagrams and figures should be **in-lined in the body of the paper**, not grouped at the end.
- Include informative legends so that readers can tell, without reading the text, what the figure is about.
- Remember that savvy scientists often read an article by skimming the abstract and studying the figures; make sure your figures hit on your key findings and conclusions.
- **You will be graded on the quality of your figures.**

The analysis is ultimately the main point of your paper. This is where you interpret the results and demonstrate how they answer or elucidate the questions you posited in the introduction. The numerical results should back your conclusions.

Try to think about what your reader wants to know. Here are some questions to guide you:

- What do your results say about the problem you are investigating?
- Was your hypothesis confirmed or disproved?

- Are the results what you expected? Why or why not?
- If you obtained unexpected results, what caused them? If you don't know, how could you set about identifying what caused them?
- What anomalies did you find? Are these expected? Why are they there?
- How do your results compare to past findings? Are they consistent? Different? Why?

Conclusion

The conclusion should give your reader the points to "take home" from your paper. It should state clearly what your results demonstrate about the problem you were tackling in the paper. It should also generalize your findings, putting them into a useful context that can be built upon. All generalizations should be supported by your data, however; the discussion should prove these points, so that when the reader gets to the conclusion, the statements are logical and seem self-evident.

Bibliography

The sources you cite give your research credibility and help put it in context with a greater effort. While you need not explore related work thoroughly, your paper should at a minimum cite the sources of your algorithms. **The textbook used in this course is not a sufficient reference.** Citations in the body of the paper should refer to a bibliography at the end of the paper. Please use a standard scientific citation format for all references.
