CS 375 Design and Analysis of Algorithms **Fall 2021**

PROJECT

I. INTRODUCTION

For this project, you will work in a Project Team with *exactly three members*, all of whom must be enrolled in our class.[1] You will select one or more algorithms, data structures, or both, to implement and test thoroughly, using your language of choice (C, C++, or Java). Importantly, you will design one or more questions that you hope to answer about your algorithm, and you will use your implementation and a well-designed set of experiments to answer your question(s). You will explain the problem your algorithm addresses, how your algorithms and data structures work, and what you learned about them from your experimentation. You will provide this explanation in a 10-minute Oral Presentation of your work to your classmates, during the last week of classes. Your Oral Presentation will be supported by well-designed and informative Powerpoint slides,[2] whose design and content will serve as a component of your grade. All three team members must contribute equitably to the project design, programming, and presentation. You will evaluate one another's contribution to the different aspects of the project, and you will demonstrate your program to the class as part of your presentation, taking no more than one or two of your ten minutes to do so. There will be 2-3 minutes of QA for each presentation beyond the 10 minutes allocated for the presentation.

This assignment is designed to enable you to:

- Study and implement a new algorithm, or implement a familiar algorithm, to gain depth of knowledge in an area of the course that interests you most;
- Broaden the base of algorithms with which you are familiar, by listening to other Project Team presentations;
- Practice the important skills of designing and delivering an Oral Presentation of your work; and
- Acquire experience working within a Project Team to complete a software development project.

II. PROJECT TEAM AND TOPIC SELECTION

Project Teams will form in one of two ways:

(1) You may select your own Project Team of three students from your section of the course.
(2) You may ask us to place you on a team, which we will do as soon as possible after you make your request.

Therefore, as soon as possible, but sometime *before* ***Midnight Wednesday November 3rd***, please send an email to aandhar2@binghamton.edu, wdai@binghamton.edu with either (i) the names of the three members of your Project Team, or (ii) a request for us to place you on a team. We will fulfill those requests "first come first served" and post official Project Teams in Brightspace. Send this email using a subject of "**CS375 Project Team**" or "**CS375 Request Project Team**", as appropriate.

Once your Project Team is established, you should write a short email with your proposed topic, including the data structure(s) and/or algorithm(s) you will implement. You should also describe what you hope to investigate about the algorithm and how you intend to do so. (More details on this below.) Send this email to aandhar2@binghamton.edu, wdai@binghamton.edu with subject "**CS375 Project Proposal**", sometime *before* ***Midnight Friday November 5th***, so that we can approve your idea for scope and feasibility, and make suggestions to improve your chances of success. We will send feedback quickly and approve your topic as soon as possible. All Project Topics must be approved and finalized by ***Monday November 8th***.

Please note that although the deadlines for establishing teams and topics are November 3rd, 5th, and 8th, there is no reason to wait until these dates. The sooner you get started, the more time you will have to complete a successful project. Teams and project ideas can begin coming together immediately!

---

[1] It is possible that *exactly one group* could have 2 or 4 members. Please assume this is *not* your group unless *we contact you*. (Don't request a group of 2 or 4; we'll decide that.)

[2] You may use other presentation software, such as Keynote or Prezi.

III. PROJECTS

Each project should implement an algorithm and investigate certain aspects and behavior of the algorithm. The algorithm could already have been covered in class, or it could be an algorithm that we did not cover, but that is described in the textbook (including in the Problems or Exercises sections). You may *not* choose an algorithm that you implemented for one of the three programming assignments, obviously; there are no other restrictions on your selection of a project topic.

Some algorithms may have various parameters, alternative components, and implementation strategies. For example, a digit in RADIXSORT may contain some number of bits, and there are different ways to perform PARTITION in QUICKSORT. You may decide to test an algorithm with different parameter values or components to see how the algorithm behaves in terms of running time. You may also try different inputs (such as arrays with different lengths, different types, and different input permutations) to see how your algorithm behaves. Testing and experimenting with your algorithm are important requirements of this project.

Some algorithms can be fine-tuned to make them more efficient. For example, for dynamic programming algorithms, some entries in the table(s) need not be computed and sometimes the procedures for the 3rd and 4th steps can be combined. You can also consider improving an algorithm as part of your investigation; in this case you should design experiments to test the extent of your improvement.

IV. PRESENTATION

Each Project Team will give a 10-minute In-Class Presentation during the final week of the semester. Your presentation should cover the following aspects:

1. ***Algorithm Purpose***: Clearly describe the *problem* that your algorithm is designed to solve. What are the inputs and outputs of your algorithm. Use an example to illustrate the problem, if possible.
2. ***Algorithm***: Describe the basic idea of the algorithm; show and describe pseudo-code. Your pseudo-code should be in the style of the text book, should be extremely neat and clear, and should use enough abstraction so that it fits on one or two slides.
3. ***Algorithm Analysis***: What are the interesting features of the algorithm? What is its time complexity, and why?
4. ***Problem Statement***: What question did you set out to answer about your algorithm? Why is this an interesting and important question? (Note that you may choose to include this part of your presentation earlier or later in your talk, depending on your project.)
5. ***Implementation***: What language did you use for your implementation? What data structure(s) did you implement?
6. ***Demo***: A short demonstration of your program. Have this ready to go; do not waste time "setting up." You may decide to provide static screen shots and sample output instead of running your program "live", to ensure a smooth presentation.
7. ***Experimental Plan***: Describe the dataset that you used to test the algorithm. How did you generate it? What characteristics does it have, and why? What did you decide to vary in the input set, and why?
8. ***Results***: What did you learn from testing your algorithm?
9. ***Limitations and Future Work***: What limitations does your project currently exhibit? If you had another month to work on the project, how would you improve your project? What additional tests would you run?
10. Concluding remarks.

Managing your time to fit all of this within 10 minutes will be difficult. Limit the scope of detail that you present, and practice your presentation! We strongly suggest that you address each of these topics explicitly in the slides, in the order provided. In other words, you should have a slide titled "Problem Statement", another titled "Experimental Plan", etc. Note that an average of 1 minute per topic above would produce a talk of appropriate length.

V. DUE DATES

The following due dates, introduced above, should represent *worst-case deadlines*. Ideally, you would work well ahead of the "schedule" that these deadlines imply!

1. **Wednesday November 3rd**: Send a **CS375 Project Team** or **CS375 Request Project Team** email to aandhar2@binghamton.edu, wdai@binghamton.edu, as described above.
2. **Friday November 5th**: Send a **CS375 Project Proposal** email to aandhar2@binghamton.edu, wdai@binghamton.edu, as described above. A "Sample Project Proposal email" appears at the end of this assignment.
3. **Monday November 8th**: Make sure you have an approved Project Team and Topic, listed in Brightspace.
4. **Midnight Tuesday November 30th**: One team member should submit to Brightspace:
    a. A serious draft of your **Presentation Slides**. Your draft should address all of the components of the Presentation, listed above. You may tweak and polish the slides before delivering your presentation to the class.
    b. A **Working Program**, including all *test programs* and *data sets* that you used for your project. These items should be tarred and zipped in the format we use for Programs.
    c. Please include a **README.TXT** file to describe how to run your program. We may or may not have you demonstrate your program to an instructor or TA; the README.TXT file, along with the materials you include in your submission, should enable us to reproduce the results that you include in your presentation.
    d. Please include the following statement in your submission, in a file named **STATEMENT.TXT**.
    "Our project group has done this assignment completely on our own. We have not copied it, nor have we given our solution to anyone else. We understand that if we are involved in plagiarism we will have to sign an official form stating that we have cheated, and this form will be stored in our student records. We also understand that we will receive a grade of 0 for this project."
5. **Midnight Tuesday November 30th**: Each student *individually* should submit to Brightspace:
    a. A single file explaining how your team divided the work (if at all), and which group members completed which aspects of the project. You should also evaluate your own contribution and the contribution of your two teammates to the success of the project. Was the work equitable? Did everyone contribute significantly? Etc. This information should take at most one page, probably much less. We will treat it as confidential. We hope to assign the same (excellent!) grade to all Project Team members, but we may not be able to do so if your evaluations of one another raise red flags. Please work hard in your group, resolve problems early in the project, come to us for help with your Project Team if absolutely necessary, and then be honest in your assessment of yourself and one another.

VI. EVALUATION

Project Evaluation (5% of your overall course grade) will be based in equal parts on:

1. **Project Selection**: How well you *defined the question* that your project intended to answer.
2. **Experimental Design**: How you approached answering the question – the selection of your datasets and the design of your experiments.
3. **Degree of Difficulty**: The difficulty of your algorithm's implementation, and the sophistication of the experimentation with the algorithm that you carried out. Please note that we are not looking for overly difficult and ambitious projects, but at the same time your algorithm cannot be trivial; you have three capable CS Majors working on it together! In assessing your Project Proposal, we will help ensure that the scope and difficulty of your project are appropriate, before you start working on it in earnest.
4. **Team Management**: Your plan for working in a Project Team, and how well you carried it out.
5. **Overall Success**: How well the code works, addresses your question, and draws a correct and useful Conclusion about your algorithm.

Presentation Evaluation (5% of your overall course grade) will be based on:

1. An evaluation by your fellow students in your class. Each student will fill out an evaluation form to ensure that feedback addresses appropriate aspects of your presentation. In particular, important issues will include:
   a. [20%] ***Goals* and *Results***: Were the goals and results of the project clearly defined and communicated? After your talk, does your audience know what question you set out to answer about your algorithm, and what the results of your investigation were?
   b. [20%] **Algorithm**: Was the main idea of the Algorithm clearly explained? After your talk, does your audience understand what problem your algorithm addresses (including inputs and outputs), and how the algorithm works? Does your audience know the runtime complexity of the algorithm, and roughly why the runtime complexity is what it is?
   c. [10%] **Slides:** Were the slides professional, helpful, and complete, according to project specs?
2. An evaluation by your instructor and TA:
   a. [20%] **Completeness**: How well did the presentation address all of the aspects it was supposed to?
   b. [20%] **Delivery and QA**: How clearly and effectively did you communicate during your presentation and how well did you answer the questions from the audience?
   c. [10%] **Slides**: How well-organized and effective were your presentation materials?

VII. SAMPLE 375 PROJECT PROPOSAL EMAIL

Subject: CS375 Project Proposal
Date: By 11:59pm Friday November 5th

Our group would like to implement Djikstra's single source shortest path algorithm, along with three different implementations of a min-priority queue, to investigate the effect of the min-priority queue's implementation on Djikstra's running time. We have code *that one of our group members wrote herself*,[3] from CS 240, for Djikstra's algorithm. We will convert that code to accept very large graphs in a format that is conducive to testing. Our current implementation of Djikstra's algorithm uses an inefficient min-priority queue in a simple singly linked list. On page 662, Cormen describes three different implementations of a min-priority queue, which exhibit three different running times for Extract-Min, namely $O(V)$, $O(\lg V)$, and $O(1)$. These runtime complexities correspond to a simple Queue, a Min-Heap, and a Fibonacci-Heap (Chapter 19), respectively. We plan to implement all three and to test the conditions under which the min-priority queue implementation impacts the overall runtime. We will measure the impact by timing the program for a variety of inputs, and will report the extent of that impact in our presentation. In particular, we would like to at least find out (a) *how large* a graph needs to be to illustrate significant differences in running time, for the different queue implementations, and (b) the effect of *graph density* on the impact of the different queues.

Our group will proceed, and divide responsibilities, as follows:

- First, we will extract our existing code for Djikstra's algorithm from the CS 240 program, removing the unnecessary aspects of the 240 assignment that the program solved. (In particular, that assignment involved airline flight schedules; we will change the code to operate on simple integer costs between graph nodes, rather than calculating flight times, etc.) We will perform this step together as a group, so that everyone understands the algorithm and our initial code base as well as possible.
- Next, we will decide on the exact interface of the min-priority queue, so that we can implement the Min-Heap and Fibonacci-Heap separately. We plan to create a C++ virtual base class called Queue, which defines all the operations necessary for Djikstra's Algorithm, and derive SimpleQueue, Min-Heap, and Fibonacci-Heap from it. With the interface set in the base class, Harriett and Cody will implement Min-Heap and Fibonacci-Heap independently. In the meantime, Devon will design a program that will generate large graph input test cases, and will convert the Djikstra code to be able to test multiple different Queue implementations for the same input cases.
- With at least one week to go until the project is due, we will integrate the working parts together, and design and run test cases to answer our questions.

Sincerely,
Harrett Hacker, Cody Coder, and Devon Developer (Section A)

[Note that Harriett, Cody, and Devon have not described the design of their dataset or experiments, which will be important for a successful project. That's OK. This sample email is intended to approximate the level of detail you should include in your Project Proposal, and the extent to which you should have thought about your project at the time you propose your topic. This would be an appropriately ambitious project, and an appropriately detailed Project Proposal.]

[3] This is important. You may use existing code, but you must have written it yourself. We expect most projects will start from scratch and will *not* use existing code.