Assignment 4

Due Date: Thursday, April 20th 2017, 11:59pm

Objectives

The purpose of this assignment is to expose you to a graduate course style assignment. As this is not a graduate level class, the assignment grading will be extremely lenient.

Graduate course assignments differ from their undergraduate counterparts in their scope and open-endedness. Some students do not like this level of freedom. It's impossible to know if you like this sort of assignment unless you try it. I'm asking you to give this a chance to see if you like this kind of open-ended project. If you end up hating this sort of project, you can now be sure that graduate school is not for you. This is perfectly OK! I'd rather you realize that during your undergrad degree. If you end up enjoying this style project, you may enjoy graduate school. This is also something I want you to realize before graduating.

This project goes a little beyond the scope of the usual CS1 class but, being an honors class, I think you all should be up to the task. I'm interested in seeing what each team discovers. Don't worry about the grades on this assignment. Have fun and make something cool! Take risks and try stuff out. If something you tried didn't work, explain this in your final presentation.

Project Details

You will form teams of three. With your team, you will decide on which of the three problem types you like most. Your task is to design a unique (or as close to unique as possible) algorithm or data structure that solves the problem.

Try to brainstorm amongst your team. Think about different approaches to the problem. Implement approaches in C and analyze empirically how fast the algorithm is. Give Big-O analysis for best-, worst-, and average-case runtimes. (Try doing average case, even if you don't have all the tools. It is OK if you are incorrect.)

Your group will give a 10-minute presentation on your project on the last day of class.

Types of Problems

sorting – In class, we covered different methods of sorting data: bubble sort, selection sort, insertion sort, lassie sort, radix sort, quicksort, mergesort, zigzag sort, etc. Many more sorting algorithms exist! The first option is to create your own sorting algorithm. Try to make the sorting algorithm efficient or have interesting behavior on certain classes of data. The algorithm doesn't need to be long or complex, just interesting.

balanced binary search trees – In class, we covered one method for maintaining a balanced binary search tree, AVL trees. There are many other approaches to maintaining a balanced tree. This includes: treaps, splay trees, red-black trees, scapegoat tree, etc. With this option you will come up with your own balancing scheme. I'm also open to techniques that don't use binary trees but instead have a related, but different, structure.

heaps – In class, we covered binary heaps as a way to solve the min-heap problem (and obtain a priority queue). There are, of course, other techniques for maintaining the minimum value in the heap. These include: leftist trees, binomial heaps, Fibonacci heaps, soft heaps, weak heap, radix heap, skew heap, etc. With this option you will come up with your own data structure for maintaining a heap.

Advice: Read about other techniques used for solving this same problem. This will let you know what to avoid but also help you think more creatively.

Team Proposal (Due 03/31)

One member of your team needs to send me an email with your teammate's names and a name for your team. Feel free to be creative with the team name!

Idea Proposal (Due 04/07)

Meet with me during office hours or a scheduled meeting time to discuss your team's initial ideas.

Final Presentation (Due 04/20)

Give a 10-minute presentation on the final day of class. Explain your approach to the problem and give Big-O analysis of space and time complexity. Share how you came up with the idea and problems you had with your approach and how you solved them. Discuss weaknesses to this approach.

Deliverables (Due 04/20)

Submit the source code for your algorithm or data structure and a PDF write-up. View the write-up as a written version of your presentation.