Final project notes

16 April 2019

This document lays out the general structure of the final project, what you'll have to do, and how it will be graded.

Topic choice and partner(s)

The groups are assigned as follows (with presentation dates):

- Wed, 24 April
  - Jason and Katie — quicksort
  - Grant and Peter — heapsort
  - Ben, Christopher, and Grady — parallel merge sort
- Fri, 26 April
  - Dean and Garrett — shellsort
  - George and Matt — circular arrays
  - Christian, Dashawn, and Trenton — AVL trees
- Mon, 29 April
  - Jennifer, Jonathan, and Michael — linear probing

Research

A big part of what you're doing in this project is learning about a data structure or algorithm on your own. Many of the topics have at least partial coverage in the textbook, and they all have relevant Wikipedia pages, but you really need to look beyond those sources to help you understand how your data structure or algorithm works.

Note that it's not against the rules to look at code, even C++ code, as part of learning how your data structure/algorithm works, although see below under Implementation.

Implementation

We'll talk on Thursday about just what I expect you to implement for the project. Whatever we decide, you'll work on it together with your partner(s) and document it together; both/all partners need to participate substantially in the implementation, but the work does not have to be a 50-50 split. One of you will submit it electronically as `proj` by the start of class on the 29th (the last day of class). By the evening of the 29th, each of you should email me separately with a short summary of what you did and what your partner did.

As you do your research, you'll find existing implementations, including some in C++, of your assigned data structure or algorithm. Any that you spend significant time reading (including those in other languages or in pseudocode) should probably be cited in a comment, although whether you cite your code as "inspired by" or "patterned after" or "adapted from" the other algorithm, or whether you use some other citation, will depend on just how you're using it.

The implementation is worth 15 points in the Lab grade (i.e. it counts as a lab and a half), which amounts to roughly 5.5% of the final course grade. I will normally assign both/all partners the same grade, but I may make adjustments if circumstances warrant.

Presentation

Your presentation should be:

- about 12 minutes, allowing 2–3 minutes for questions after (for a two-person group), or
- about 18 minutes, allowing 2–3 minutes for questions after (for a three-person group).

During the actual talk, I'll raise a sign at 5 minutes left (that is, after 7 or 11 minutes), at 1 minute left, and at zero minutes left, to help you keep track of your pacing. Both/all partners need to participate substantially in the presentation, but the time does not have to be a 50-50 split.

As described in the syllabus, your presentation should include:

Accurate example diagrams. The exact form this will take will depend on what you're presenting (is it a tree-based data structure? A linear one based on arrays?), but it should have real, concrete data in it. The data should be arranged correctly for your data structure and set up so that it will be illustrative when you run algorithms on it. Probably at least one of your diagrams should have "typical" data, or at least a region with typical data; but make sure that your examples include data that show unusual or interesting cases. For instance, if adding to the end of your data structure has to be handled specially, you should be sure to have data where an insertion would be done at the end.

Pseudocode and tracing using the example. You should display on the screen or write on the board actual pseudocode. The "pseudo" means it doesn't have to be pure C++, but it should still be clear and step-by-step. You'll trace an algorithm using the example you presented, showing us the updates to any counters or accumulators as the algorithm runs (as well as updates to the main data structure itself, of course). You'll narrate what you're doing—don't just change a lot of numbers and assume the audience will know what they all mean—but make sure that you don't *only* narrate step-by-step: you should also be explaining the big-picture effects of what the code accomplishes. For instance, what does *each iteration* of a loop do? What parts of the data structure are "done" vs. "still need to be modified"?

A demonstration of either correctness or efficiency. Anyone thinking critically about your data structure will be inclined to ask questions like, "Why should I believe that the result of following these steps will give me a correct answer?", or, "Why should I believe that this algorithm really runs in $O(\lg n)$ time?" For at least one of the algorithms you present, you should put yourself in their shoes, identify a situation that one of these skeptics might reasonably expect to "break" the algorithm (making it incorrect or slowing it down), and show that it really is correct or that it really has the stated efficiency even in the worst case.

You should *not* be presenting a thinly-veiled reading of the Wikipedia page, or any other source. Synthesise your understanding of the topic into a presentation that's really your own.

You may use whatever combination of whiteboard and technology (e.g. Powerpoint, program demos, your own laptop) that you find most effective, but if you are using technology make sure it runs in our classroom *before* you come to class on the day of your presentation. Whichever tech you are using, be sure neither to err on the side of too little text (e.g. no labels at all) nor to err on the side of too much (e.g. full sentences on your slides).

Below is the 30-point rubric that I'll be grading your presentation on.

RUBRIC

Overview	
1	Explains what we might use this data structure or algorithm for
1	Connects this DS/Alg with related topics, i.e. other DS/Algs we've covered
1	Gives high-level overview description of DS/Alg, how it works
Example diagrams	
1	Presents some sort of example diagram with concrete data
1	. . . that is generally of the right shape or structure
1	One example is correct example of appropriate "typical" data
1	Multiple data examples, and all are correct examples of appropriate data
1	Examples illustrate interesting, nonobvious, or boundary cases
1	Presenter clearly explains what we are seeing in the diagram
Pseudocode	
1	Presents some (pseudo)code
1	. . . that does at least roughly the right thing
1	The algorithm(s) shown are correct
1	Pseudocode is legible and clear
1	Presenter clearly explains the code at a high level (not just narrating trace)
Demonstration of the algorithm (tracing/visualisation)	
1	Example on screen or whiteboard is changed according to algorithm
1	. . . and the changes are explicitly linked to specific lines in the code/pseudocode
1	. . . and local variables (pointers, counts, indices) are updated appropriately
1	One complete run of an algorithm is correct
1	All demos, including edge case demo(s), are correct
1	Presenter clearly narrates the changes as they occur (not just high-level explanation)
1	The demonstration is effective at showing how the algorithm works
Demonstration of correctness or efficiency	
1	Presenter explicitly addresses either algorithm correctness or efficiency
1	. . . and presents legitimate evidence to support their claim
1	. . . and the evidence is both correct and convincing
Presentation mechanics	
1	First partner talks (yes, this point is just about free)
1	Both/all partners talk
1	Talk is well-planned for 12 (18) minutes without ending way early or rushing
1	Presenters are prepared and know what they need to say and do
1	Presenters speak clearly and fluently without just reading script or slides
1	Presenters make effective use of chosen technology (whiteboard, slides, whatever)

The presentation is worth 10% of the final course grade. I will normally assign both partners the same grade, but I may make adjustments if circumstances warrant.