CS372 HW 5: CS3851 Final Project

Proposal: due Monday Nov 21st, at 11:59 PM

Video Presentation: due Thursday Dec 1st at 11:59PM (+2 for 24+ hour early submission)

Report and Code: due Tuesday Dec 6, at 11:59 PM (+2 for 24+ hour early submission)

You must email me saying the submission will not be edited for the early bonus points since I take unlimited submissions

+3 on the exam 3 if you fill out the class evaluation survey

Instructions

Proposal (due Monday Nov 21st, at 11:59PM)

Select an algorithm, application and language for your final project proposal. Your proposal may be very short (3-5 sentences is plenty). This triple may NOT match anyone else in the class, and there will be a running chart of everyone's selection on D2L. Read the report requirements before you make your final decision.

1. When choosing your algorithm, the following restrictions apply,
    a. You may propose an algorithm not discussed in class, an algorithm in the "Algorithm Suggestions" table in the appendix, or two algorithms discussed in class, but not done in homework from the "Algorithms from Class" table in the appendix. In the last case, the two algorithms must use each other in the same application (I should be able to test them both with a single "run"). Unpermitted algorithms are also in the appendix.
    b. If you propose an algorithm not in the tables, I reserve the right to veto or edit the proposal. This means if I think you chose something too small, I may expand it. If you chose something too large, I may contract it. It is unlikely that I will veto an algorithm if it has a run time greater then O(n) and it was not in Data Structures.
    c. Make sure you pick a specific algorithm and not a class of algorithms
2. When choosing your language, the language must be in the following list: C++, C, C#, Java, Python, PHP (you must give me you code and access to the server you tested it on), Javascript (embedded in a webpage), LISP (you must give me the variant used), Visual basic, Android (variant of Java), or Ruby.

3. When choosing your application, the following restrictions apply
    a. The input data must mean something. It cannot be a random string of numbers. For example, if merge sort was an option, the values would need to come from something other than a random generator. For example, a list of temperatures pulled from across the USA is acceptable.

You may ask for feedback on your selections at any time before the proposal deadline. I strongly advise this to confirm a unique triple, and for feedback on algorithms selected outside of the given tables. After the proposal due date, your project is set WITH any additional restrictions that may be placed after reviewing your submission.

Presentation (due Thursday Dec 1st at 11:59PM) (+2 for 24+ hour early submission)

- Create a short 60-120 second video briefly describing your algorithm. These *will be played* in class. Treat this as an "elevator pitch." It must include.
    a. Your name
    b. Algorithm name
    c. Application
    d. Other application that use this algorithm
    e. Other algorithms that your application could use.
    f. Basics of how the algorithm works/what it does
    g. Show algorithm running (and part of the run is fine)
- Other, optional, suggestions for your video are
    a. Run time
    b. Input/output
    c. Data source
    d. Why you picked what you did
- Your implementation does NOT need to be completed by the time the video is due.
- Sound is optional
- If it runs on a basic VLC install on Windows, the file type is acceptable.
- Good video making tools are:
    a. Camtasia (the free download should give you enough time to make your video)
    b. Powerpoint
    c. Jing
    d. Demo Builder (the free download should give you enough time to make your video)
    e. TinyTake
    f. I strongly prefer a video, but if you do this in person, you are saying "OK" to a video tape, and I still want all materials you use in the presentation submitted in the video dropbox. You also MUST arrange it with me prior to the due date.

Report and Code (due Tuesday Dec 6th , at 11:59 PM) (+2 for 24+ hour early submission)

Code

- Implement your algorithm for your application
    a. Validate the correctness empirically
    b. Determine your needed test cases (see the Department's Test Standard to confirm a sufficient number and range of tests)
        i. These must be listed in your report
    c. Include a test method (or integrative or unit test) in your code. Mark this with a "RUN THIS TO TEST" comment.

Report

Write a report on your final project. This will likely be 2-3 pages. This is the primary documentation for your project. Think of this like writing a "tutorial" for your project, and a template will be provided. Your written report, must include:

1. Your name
2. Your algorithm choice
3. Your application
4. Your language
5. The class of problems your algorithm(s) solves
   a. Also, you MUST name least 2 other applications.
6. Other possible algorithms for your application.
   a. Compare and contrast your selected algorithm with other algorithms you investigated that solve the same class of problem. Also, you need to answer, "Why did you choose what you chose?"
7. Validate the correctness theoretically
   a. One option is to justify the validity of your algorithm with a high-level proof. Proofs with loop invariants and proof of contradictions are acceptable, and expected. English and no pseudocode is acceptable. This is for any test set, not just the test sets you used for debugging.
   b. Another option is to "chunk" out you algorithm and explain what each block does, and why it works for any legal input.

   I'm looking for something very high level and anything more that 1-2 paragraphs is probably more than necessary. The main reason for this is to be sure you understand what you coded! Pretend you are explaining how it works to your grandma. If that doesn't work, pretend you are talking to a freshman CSC major who just coding for the first time 8 weeks ago. If you are unsure what this would entail for your algorithm please ask. If you pick something with randomness (such as "random gradient descent"), this may be a bit different.

8. List your test cases for your empirical tests
9. List the output for each of your test cases
10. The algorithm's theoretical run time
    a. Include why it has this run time. The may be done with instruction counting, probability, or recursion analysis depending on your problem
11. Create a runtime graphs for varying n (if you have more than 1 value that affects input, just vary one for the graph, but be clear which you used)
    a. Compare and contrast this to the theoretical run time
12. References –important to provide proper attribution. Please include your source(s) for your psudeocode or other code base if applicable. For example, if you used a tutorial, I want to know where you found it.

Submission Instructions

Proposal due Monday, Nov 21st, at 11:59 PM

Submit your written proposal (pdf or doc) by Friday Nov 18th, at 11:59 PM on D2L week 13. We will be taking part of Friday Nov 18th to select options in class if you are still looking for a unique triple. WARNING: I will be traveling Wednesday the 16th right after class to Friday, but you will have a guest lecturer to ask questions, and I will be checking my email.

Video Presentation due Thursday Dec 1st at 11:59PM (+2 for 24+ hour early submission)

Submit your video by Thursday Dec 3rd , at 11:59 PM on D2L week 15. The videos will be shown starting Friday, Dec 4th after the exam review. Any file that can be run on a basic VLC install on Windows is acceptable. You code does NOT need to be finalized.

Report and Code: due Tuesday Dec 6th , at 11:59 PM (+2 for 24+ hour early submission)

1. Zip up your code (.cpp and .h files only) AND PDF with the file with your "main" function at the **root** level. Name this **lastnameFirstInital** (so mine would be rebenitschL). If you know there is a repeat name, please use the first two letters of your first name.
2. Submit your zip folder to D2L (you should be submitting only 1 file). Any archive file that can be opened with 7-Zip is acceptable. The dropbox will be under the associated week's content (Week 16 in this case). CHECK that it uploaded correctly. I need that D2L email to confirm submissions!
    a. *Alternative:* zip up just your code in a zip folder and submit to D2L, and place your written submission in my mailbox, slide it under my door, or give it to me in person.

Rubric

Task	Points
Correct Submission	6 points (2 each)
Coding standard	6 points
Proposal	8 points
Presentation	15 points
Code	25 points
Report	28 points
Empirical tests	12 points
Total	100 points

Appendix

Algorithm Suggestions

Bipartite numbers	Minimum Edit Distance	Efficient String search
Graph analysis – PageRank, HITS	Clustering –(MANY variations)	Gradient descent search
Convex hull	Fibonacci Heaps	Almost any dynamic algorithm
Random number generator	Polygon area	Variation of the dynamic knapsack problem (you must give me your cost function)
Knuth-Morris-Pratt string matching	Ford-Fulkerson (or other maximum flow algorithm)	Suffix tree
Strongly connected components	NP-complete (no approximation versions)	Finite automaton string matching
Fenwick Tree	Segment tree	Aho-Corasick string matching
NP-complete algorithm		Anything else from the book

Algorithms from Class

Any segment pair intersect	Closest pair of points	Guaranteed O(n) selection
Rod cutting problem	Capital budgeting	Radix sort (must use a radix other than 2 or 10)
Variant of FFT	Topological sort	Bucket sort
Hiring problem	Variant of Longest Common Subsequence (original is not acceptable)	Optimal binary search tree
Dynamic version of activity selection	Point inside polygon	Rabin Karp string matching
0-1 Knapsack	Boyer-Moore string match	Power function in a modulus base at the bit-level
Chinese remainder theorem		All pairs shortest path

Unpermitted Algorithms

Anything from data structures	Anything done in homework (pseudo)code	O(n^2) sorts
O(nlgn) sorts	Topological sort	Radix sort (with a radix of 2 or 10)
Breadth/depth first search	Original A* (variants are permitted)	O(n) application of formula to data (if you are planning a matrix of these, please ask)
Basic permutation algorithms	Greedy smallest number of coins for change	Original FFT (variants are permitted)
	Euclid's algorithms	