CS372 HW 5: CS3851 Final Project

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

Video Checkpoint: due Tuesday Dec 5th at 11:59PM (+2 for 24+ hour early submission, +4 for 48+ hour early)

Final Video Report and Code: Monday Dec 11th at class time (+2 for 24+ hour early submission, +4 for 48+ hour early)

Instructions

The purpose of this assignment is to demonstrate the ability to learn algorithms on your own and integrate them into an application. To that end, you also must understand the consequences of your choice (such as the run time) and can show correctness in your program. The applications are permitted to be "alpha" versions such that they are "prototype" systems and you can expect clean input.

Proposal (8 points)

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. Please, submit to D2L regardless of approval via email (…bookkeeping on my end).

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 second case, the two algorithms must be used in the same application. This should NOT be two separate applications. In other words, 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. For example, "Boyer-Morre" is acceptable, "string matching" is not.
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 (VBA specifically), Unity, Android (variant of Java), UNITY, or Ruby. I will

need the full project and make files this time. If you do not have one, I'll still need the README for compiling it on Windows 10.

3. When choosing your application, the following restrictions apply
    a. The input data must mean something. Part of the assignment is to show you can integrate an algorithm into an "legitimate" application. Data 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.

Checkpoint Video (14 points)

A near complete video presentation doubles as a checkpoint for the project, assuring you have gotten your project to compile. Video tutorials have become more popular then text tutorials. Therefore, you will make a short video presentation with a "elevator" speech for your algorithm. As I know this will be the first time making a video presentation for many of you, treat this as a "hello world" of video presentations.

- The video must be 2-4 minutes, and these *will be played* in class. It must include the following components. I'll be *literally* checking off if these exist in your video for the checkpoint:
    1. Your name and your algorithm's name
    2. Your Application
    3. Other applications that use this algorithm
    4. Other algorithms that your application could use.
    5. Basics of how the algorithm works/what it does
    6. Show algorithm running (any part of the run is fine)
- Other, optional, suggestions for your video are
    7. Run time
    8. Input/output
    9. Data source
    10. Why you picked what you did
- Your implementation does NOT need to be completed by the time the video is due.
- Sound is optional
- The file MUST runs on a basic VLC or Firefox install on Windows.
- Good video making tools are:
    a. Ice Cream Screen Recorder (the free version should be sufficient)
    b. Windows 10 (the x-box app has a hidden screen capture)
    c. Powerpoint (record your slideshow and then export it to a video. You may still need to embed a video of your code running in the power point)
    d. Demo Builder (the free download should give you enough time to make your video)

Final Report, Code, and Video (remaining points)

Video

Update your video to reflect any changes in the project, and have it **clearly** show the code running **with a completed project.**

Code

- Implement your algorithm for your application
    a. It is *very* good practice to include built-in tests that check the accuracy of you code. Determine your needed test cases, and implement **at least 3.** Language built-in testing are also fine.
        i. These must be listed in your report
    b. Mark the tests with a "LOOK HERE TO TEST" comment, and then any additional commenting on how to test. *If you don't have this exact comment*, I'll assume you don't have the built-in tests.
    c. You are permitted to do some tests manually, but fundamental requirements of the algorithm (such as "does it work with a small case" or the "low edge/boundary" case) should be in code.
- Make sure I can run your code. This means including any special libraries or dlls in your submission, and have them referenced properly. If I can't get it running in less than 5 minutes, you will have a 20% deduction on the code part. This means if I have to install anything, you are going go past **the 5 minute limit**. You may check with me ahead of time if I have a particular library installed.

Report

Write a report on your final project. This will likely be 2-5 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, which MUST be used. Not using the template will result in a **25% deduction** on your report specific score. You will be graded on clarity and succinctness, as well as completeness. 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.
    b. You MUST name least 2 other algorithms
7. Using the answer to 6, also answer, "Why did you choose what you chose?"
8. Validate the correctness theoretically. This is more so, why/how does it work? This is for any test set, not just the test sets you used for debugging.

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.
b. Another option is to "chunk" out you algorithm and explain what each block does, and why it works for any legal input.
c. Be sure this is not just what the input and output is. I need what happens in between for your algorithm (not the entire application).

> 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 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"), you will have to explain this in probabilistic terms.

9. List your test cases for your correctness tests
10. List the output for each of your test cases
11. The algorithm's theoretical run time for your implementation
    a. Include **why** it has this run time. The may be done with instruction counting, probability, or recursion analysis depending on your problem. You will likely need to use psudeocode to prove this.
12. 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
    b. There should be a minimum of ten points
13. 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

Submit your written proposal (pdf, doc, txt, or D2L comment) on D2L week 13.

Video Checkpoint Presentation

Submit your checkpoint video by on D2L. The file MUST run on a basic VLC install or Firefox on Windows. You code does NOT need to be finalized. Since I will be playing these randomly, the naming is less important, but it should still identify the author.

Final Video, Report, and Code:

1. I will be playing your video presentation in class.
   Zip up your video, code, AND PDF with the file with your starting project or file with main (depending on you project structure) at the **root** level. I want **everything** but the

compiled files this time! If it exists before you compile, I need it. If it exists afterwards, I do not. If you are using Visual Studio you may, and should, delete the intellisence files as well (the sd and sdf extensions). You MUST use the report template. Not using the template will be a 25% deduction on your report specific score.

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. 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	6 points
Presentation	15 points
Code	25 points
Report	28 points
Empirical tests	14 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
Random number generator	Fibonacci Heaps	Almost any dynamic algorithm
Knuth-Morris-Pratt string matching	Ford-Fulkerson (or other maximum flow algorithm)	Variation of the dynamic knapsack problem (you must give me your cost function)
Strongly connected components	NP-complete (no approximation versions)	Suffix tree
Fenwick Tree	Segment tree	Finite automaton string matching
NP-complete algorithm	AES encryption	Aho-Corasick string matching
Ford-Fulkerson	Markov chains	Fortune's algorithm
DBSCAN	edit distance/ Levenshtein	k-means
Page rank	Non uniform bucket sort	Anything else from the book
O(N) search	O(n) convex hull	Hungarian algorithm
Rabin Karp string matching	O(n) suffix tree	Boyer-Moore string match
Chinese remainder theorem		Variant of FFT

Algorithms from Class

Any segment pair intersect	Closest pair of points	Uniform bucket sort
Rod cutting problem	Dynamic knapsack	Matrix multiplication
1D FFT	Graham scan	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	Jarvis's march

Power function in a modulus base at the bit-level	2D or variant of Kadane	You may use Djikstra's algorithm if it is a comparison to another search.
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)
Huffman encoding	Original A* (variants are permitted)	O(n) application of formula to data (if you are planning a matrix of these, please ask)
Breadth/depth first search	Greedy smallest number of coins for change	Original FFT (variants are permitted)
Basic permutation algorithms	Euclid's algorithms	Polygon area
Capital budgeting	1D kadane	Horners calculation
Greedy activity selection		