Synthesis Assignment #3
CS 5800, Fall 2022
Dr. Lindsay Jamieson

DUE: December 5th, 11:59pm pacific via Canvas

Tl;dr version - create a textbook collecting your explanations of 11 different topics including problems and solutions for each. Put it in a single pdf/docx file and submit on Canvas.

Your task is to demonstrate your understanding of the material by writing a mini-textbook. For each topic we've discussed in class (up to the week before this assignment is due) you will write an explanation of the topic based on the way that you understand the concept. A complete topic list is provided at the end of the document, divided into "chapters" based on the modules as we presented them in class.

Each topic will be complemented by at least one problem and solution of your own creation (no taking problems from other sources). This means that you will (in addition to the write up) be also responsible for creating 8 original problems and their solutions (with some noted exceptions). **NOTE:** these are problems like homework problems NOT problems that you are using to explain the concept. Yes, this means that there are 11 sections that you will be creating as part of this assignment. **ANOTHER NOTE:** implementation exercises (i.e. "Implement this algorithm in Python") are not acceptable for the synthesis.

Please clearly number and space out your sections so that I know exactly where one section ends and the other begins. If helpful, a format I would suggest is:

1 **Chapter Name** (like Asymptotic Analysis)

1.1 **Section Name** (like Big O)
<your write up goes here>
Exercise: <problem write up>
Solution: <solution write up>

1.2 **Section Name** ...

Note: these sections don't have to be terribly long or involved, **but they have to demonstrate that you do understand the concepts enough to explain them and come up with exercises**. There is no page limit, or page minimum, so use your best judgment. An example of the kind of thing that I'm looking for will be provided in the assignment area on Canvas.

Special topic note: The topics labeled "XXXX design" are topics that are meant to distill your knowledge about the topic, and as such do not have a particular linkage to a specific thing talked about in class. These are designed to explain to the reader how you, personally, would

design an algorithm for a problem using that technique. The original exercise for these sections should show your steps and thought process behind designing a solution for a particular problem, utilizing this technique. **DO NOT GIVE ME JUST CODE**. You must explain every part of your solution.

Teamwork note: this is an assignment where it is important that you are the sole author of your work. You are allowed to discuss concepts with your classmates, but at no time are you allowed to discuss or share problems, writeups, or your overall plan for this assignment with anyone other than the professor or the TAs.

Due: December 5th, 11:59pm pacific, via Canvas
Format: a single pdf or docx file. Note that you can create this in Google docs, just please export to pdf or docx before submitting.

Topics to be covered (organizational section in bold, actual topics listed under each "chapter"):

Chapter 6: Greedy Algorithms Continued -

- encodings & Huffman's Encoding Algorithm
    - Original problem should be a step-by-step execution of Huffman's Encoding Algorithm with a list of frequencies of length at least 6.
- greedy algorithm design
    - Problem should be to pose a problem and solve that problem with a greedy algorithm, discussing your thought process and explaining correctness and time complexity. This problem does not need to be original, but your solution and explanation should be.

Chapter 7: Dynamic Programming -

- technique definition (no problem/solution required)
- Edit-distance
    - Original problem should be a step-by-step execution of the edit-distance algorithm with two words of length at least 6 characters.
- knapsack (both 0-1 and unrestrained)
    - Original problem should be a step-by-step execution of a solution to the 0-1 and unrestrained knapsack problem using the same weight/value list for both executions with at least 5 items in the list.
- dynamic programming algorithm design
    - Problem should be to pose a problem and solve that problem with a dynamic programming algorithm, discussing your thought process and explaining correctness and time complexity. This problem does not need to be original, but your solution and explanation should be.

Chapter 8: Linear Programming -

- technique definition and problem specification
    - Original problem should be taking a description like the example exercise in class and converting it into a linear programming problem.

Chapter 9: NP-Completeness -

- definition (no problem/solution required)
- SAT and 3SAT
    - Original problem should be to create a Boolean equation and prove that it is satisfiable (or not). This requires a formal proof.
- Proving NP-Completeness
    - Problem should be to take a problem and prove that it is NP-complete. This problem does not need to be original, but your explanation should be. This requires a formal proof.
- getting around NP-Completeness (no problem/solution required)