Peter Anteater panteate Seat: TBD

CompSci 260P Exam 1 Diagnostic Exam 1 Spring 2023

DO NOT OPEN EXAM UNTIL INSTRUCTED TO DO SO

SILENCE AND STORE ALL ELECTRONICS

This is a diagnostic exam intended to help you evaluate your readiness for the real exam.

The following rules apply to you, whether you think they do or not. Read and understand them; failure to abide by these rules, or directions given by course staff during the exam, may result in disciplinary action, including but not limited to a failing grade in the class.

- This exam is solely for students enrolled in this lecture. Anyone not enrolled in this lecture may not take an exam.
- You may not open the exam or begin writing until the instructor has explicitly given you permission to do so.
- Keep your UCI ID readily accessible during the test. Proctors may request to see it.
- This exam is closed book, closed notes, and is individual effort. Once course staff begin passing out exams, you may not communicate with anyone other than proctors for any reason, nor may you have electronics, including calculators watches and phones, available to you during the test for any reason. **YOU DO NOT NEED A CALCULATOR!**
- If you leave your seat during the test for any reason, your instructor may collect it and deem you to have turned it in. Do not ask proctors for an exemption to this, they are not authorized to grant such.
- If you are still seated at 7:35, you may not leave your seat until explicitly dismissed by the instructor. Leaving after 7:35 and before being dismissed may result in a grade penalty.
- If you believe a question is ambiguous, write at least two reasonable interpretations and indicate clearly which one you will be using. Then answer your question with that assumption. Unless your interpretation makes the problem much more trivial than intended, we will grade your response as if one of us had made that clarification.
- The purpose of the real exam is to evaluate how well *you* understand the material presented in the course. It is an academic integrity violation to do anything that subverts the goals of this assessment including, but not limited to, not doing your own work or submitting that of anyone else.
- We will only grade responses marked in the space provided for each question.

Nothing you write on this page will be graded. The next page in this booklet contains a spot to answer these questions. You may use this page as scratch paper if you would like, and room to do so exists.

A number of languages are written without spaces between the words. Consequently, software that works with text written in these languages must address the *word segmentation problem* – inferring likely boundaries between consecutive words in the text. If English were written without spaces, the analogous problem would consist of taking a string like "meetateight" and deciding that the best segmentation is "meet at eight" and not "me et at eight" or "meet ate ight" or any of a huge number of even less plausible alternatives.

One approach to processing this is to find a segmentation that maximizes the total "quality" of the individual constituent words. Suppose you are given a function `quality(S)` that takes, as input, a string and returns a number, which may be positive or negative, where larger numbers correspond to more plausible English words. The *total quality* of a segmentation is determined by adding up the qualities of each of its blocks.

Use dynamic programming to design an efficient algorithm that takes as input a string s and determines the segmentation of maximum total quality. For simplicity in analyzing the running time, you may assume that the substring operation takes $\mathcal{O}(1)$, that the `quality` function runs in $\mathcal{O}(1)$, and that the `quality` function can (if you prefer) be `quality(`S, i, j`)` which returns the quality of $S[i \ldots j]$, also in $\mathcal{O}(1)$.

Remember to write your answer on the answer page →, not here, although you may use this space for scratch work.

To complete these instructions, do the following.

- Give a clear and precise English definition that describes the function you are implementing. Not *how* it works (yet), but rather *what* it solves.
- Give that function a meaningful variable name. This is not [just] me being pedantic; I have found it helps students with this topic if they do this. "OPT" is not a meaningful variable name, nor is "table." Single letters are not meaningful variable names.
- Give a clear recursive formula or algorithm in terms of smaller instances of *exactly* the same problem.
- Describe the iterative running time correctly. This can either be by writing the iterative algorithm (in which case, you can point out where the previous part is within the solution), or by taking your recursive solution, counting the cases, describing the order in which the table would be filled in, and analyzing the time accordingly.

Write your answer to question 1 on this page

Nothing you write on this page will be graded. The next page in this booklet contains a spot to answer these questions. You may use this page as scratch paper if you would like, and room to do so exists.

Read the following prompt. There are two problems described. One of them can be solved in polynomial time. The other is known to be $\mathcal{NP}$-complete .

You will need to identify which one is which, indicate why one can be solved in polynomial time, and indicate why the other one is likely to be computationally difficult via reduction. For the last part, you may assume any problem shown in class (lecture or lab) to be $\mathcal{NP}$-complete is. The following are a few such problems:

- Independent Set: Given a graph G and an integer k, determine if there is some subset of the vertices V', $|V'| \geq k$ such that no two vertices in V' share an edge.
- 3-Color: Given an undirected graph G, determine whether there is a mapping of vertices in G to three distinct colors such that no edge is monochromatic.
- Hamiltonian Path: does this graph contain a simple path that includes every vertex? This is $\mathcal{NP}$-complete for both directed and undirected graphs. It is also $\mathcal{NP}$-complete to check for Hamiltonian Cycles (cycles that include every vertex) in either directed or undirected graphs.
- Subset Sum: given a list of numbers S and a target number T, is there a subset $S' \subseteq S$ whose elements add to T?

Your reduction may not be based on a problem not shown to be $\mathcal{NP}$-complete in lecture or lab.

.....

Consider the following two problems that may arise with respect to loading potentially hazardous cargo into trucks. We have a warehouse with n canisters, each of which have some form of hazardous material. We also have m trucks, each of which can hold up to k containers.

- Version One : For each canister, there is a specified subset of trucks that may carry it. Is there a way to load all n canisters into m trucks such that no truck is overloaded and each container goes into a truck that can carry it?
- Version Two : In this version, any canister can be placed on any truck. However, there are some pairs of containers that cannot be placed on the same truck. Is there a way to load all n containers onto the m trucks such that no truck is overloaded and no two containers are placed on the same truck when they cannot be?

Identify which version of the problem is polynomial time solvable, explain why it is, and indicate why the other one is likely to be computationally difficult via reduction.

Remember to write your answer on the answer page →, not here, although you may use this space for scratch work.

Which version is polynomial-time solvable? Circle exactly one :

VERSION ONE VERSION TWO

Briefly justify your answer:

.....

Which version is $\mathcal{NP}$-complete ? Circle exactly one :

VERSION ONE VERSION TWO

Provide a reduction.