CMPSCI 311: Introduction to Algorithms **Spring 2017**

Discussion 4

Your Name: ____________________ Collaborators: ____

You will be randomly assigned groups to work on these problems in discussion section. List your group members on your worksheet and turn it in at the end of class.

Problem 1. Class Selection. You are helping a freshman entering the architecture major plan out their academic career starting this fall. The freshman can take up to one core class per semester. Each core class is consistently offered in either the fall or the spring semester, but never both. A fall core class will only have other fall core classes as prerequisites and a spring core class will only have other spring core classes as prerequisites. There are no circular course requirements.

The freshman must take all core classes and wants to finish taking core classes as soon as possible. Note that there might be semesters where the freshman takes no core classes.

(a) Find the best ordering for the student to take the classes if the set of core classes that the student needs are:

Class	Semester	Prerequisites
2D Design	Spring	
3D Design	Spring	2D Design
Analysis I	Fall	
Analysis II	Fall	Analysis I, Basic Drawing
Basic Drawing	Fall	
Basic Studio	Fall	
Composition	Fall	Analysis I

* This list is fictional.

(b) Consider the general problem where want to produce the order that the student should take core classes given a list of the core classes, their prerequisites, and if they are in the fall or the spring. We can model this problem by a graph: Create a node for every class and draw an arrow from class A to class B if A is a prerequisite of B.

(1) How can you model a feasible ordering in the primary problem in this graph?

(2) Design an algorithm to find such an ordering.

(3) Show correctness of your algorithm.

(4) What is the run time of your algorithm?

Problem 2. Hiking.

Suppose that three of your friends, inspired by repeated viewings of the horror-movie phenomenon The Blair Witch Project, have decided to hike the Appalachian Trail this summer. They want to hike as much as possible per day but, for obvious reasons, not after dark. On a map theyve identified a large set of good stopping points for camping, and theyre considering the following system for deciding when to stop for the day. Each time they come to a potential stopping point, they determine whether they can make it to the next one before nightfall. If they can make it, then they keep hiking; otherwise, they stop.

Despite many significant drawbacks, they claim this system does have one good feature. "Given that were only hiking in the daylight," they claim, "it minimizes the number of camping stops we have to make."

To make this question precise, lets make the following set of simplifying assumptions. Well model the Appalachian Trail as a long line segment of length L, and assume that your friends can hike d miles per day (independent of terrain, weather conditions, and so forth). Well assume that the potential stopping points are located at distances $x_l, x_2, \cdots, x_n$ from the start of the trail. Well also assume (very generously) that your friends are always correct when they estimate whether they can make it to the next stopping point before nightfall.

Well say that a set of stopping points is valid if the distance between each adjacent pair is at most d, the first is at distance at most d from the start of the trail, and the last is at distance at most d from the end of the trail!. Thus a set of stopping points is valid if one could camp only at these places and still! make it across the whole trail. Well assume, naturally, that the full set of n stopping points is valid; otherwise, there would be no way to make it the whole way.

We can now state the question as follows. Is your friends greedy algorithm - hiking as long as possible each day - optimal, in the sense that it finds a valid set whose size is as small as possible?