CIT 596 Online **Spring 2020**

Module 7–8

Name: NAME HERE

Questions requiring written answers. Note that greedy algorithms are generally simple. It can be surprising that such simple algorithms can actually work for seemingly complicated problems. As a result of this, more emphasis will be placed on a rigorous proof of correctness when evaluating your solutions than usual; less credit will be apportioned to the algorithm itself.

1. The *fractional knapsack problem* is stated as follows. You are given a knapsack that can carry a weight of at most W. You are also given a collection of n items, each with an associated weight and value (both positive). Given that you are allowed to take a fraction of an item (with corresponding proportional value), design an algorithm determines which items to take (along with the weight of each item taken) that maximizes the value of the knapsack.

2. The Huffman encoding algorithm for generating optimal binary codes for an alphabet can easily be extended to generating any k-ary codes. (In a k-ary code each alphabet symbol is encoded as a string of 'digits' where each digit can take on values $0, 1, 2, \ldots, k-1$. Consider how this would be done and what minor modifications are necessary in the proof of correctness. Try to think about small examples to help you design to a general algorithm.

3. You are given a long, straight road with n houses located at points $x_1, x_2, \ldots, x_n$. The goal is to set up cell towers at points along the road such that each house is in range of at least one tower. If each tower has a range (radius) of r, give an algorithm to determine the minimum number of towers needed to cover all houses and the locations of these towers.

4. There are n jobs that need to be processed by computers. Each job i must first run for s_i minutes on a centralized computer before running for an additional t_i minutes on a personal computer (each job has its own personal computer). Design an algorithm to schedule the order in which jobs should be submitted to the centralized computer so as to minimize the time in which all jobs are completely done processing.

5. Let $G = (V, E, w)$ be a directed graph, where w is a weight function that assigns a positive weight to each edge. Let s be a node in G. Suppose we are given a program P that claims to implement Dijkstra's algorithm. For each vertex v in the graph this program produces $v.\pi$ which is supposedly the vertex that precedes v on the shortest path from s to v. It also produces $v.d$, which is supposedly the distance from s to v. Design and analyze an efficient algorithm that takes G and P's output as its inputs and checks if P's output is correct. Your algorithm should obviously be faster than rerunning Dijkstra's algorithm on G.

6. Suppose a graph G has two minimum spanning trees. Prove that in any cycle formed by the union of the edges in the two trees, at least two of the edges have the same weight. Is it also true that this duplicated weight is the largest weight on the cycle? Prove or give a counter-example.

7. You have to cut a log into several pieces. To do so, you bring it to a company that charges money according to the length of the log being cut. Their cutting method means that only one cut can be made at a time.
   For example, suppose that a log of length 10 meters has to be cut at positions 2, 4, and 7 from one end. There are several ways to do this. One way is to first cut at 2, then 4, and then at 7. This gives a price of $10 + 8 + 6 = 24$ because the first piece was 10 meters in length, the second was 8 meters, and the third was 6 meters in length. Another way is to cut at 4, then at 2, then at 7. This would

give a total price of $10 + 4 + 6 = 20$, which is a cheaper price.
Not all problem have greedy algorithms; when they do exist they are simpler to design but harder to prove correct since you have to prove that the first decision the algorithm makes is the right one.

One could consider several different greedy algorithms for this log cutting problem.

(a) Provide a counterexample to show that a greedy algorithm that always makes the median cut first is not correct. (If n cuts have to be made, the median cut is the $\lceil \frac{n}{2} \rceil^{th}$ cut, where ever it is located.

(b) Provide a counterexample to show that a greedy algorithm that always makes the cut closest to the center of the log first is not correct.

(c) Propose another simple, greedy way of choosing the first cut, and show that that does not work either.