CMPSCI 311: Introduction to Algorithms **Spring 2018**

Homework 2

Released 2/07/2018 Due 11:59pm 2/21/2018 in Gradescope

Instructions. You may work in groups, but you must individually write your solutions yourself. List your collaborators on your submission.

If you are asked to design an algorithm as part of a homework problem, please provide: (a) the pseudocode for the algorithm, (b) an explanation of the intuition for the algorithm, (c) a proof of correctness, (d) the running time of your algorithm and (e) justification for your running time analysis.

There are four questions, each worth 25 points total.

Submission instructions. This assignment is by 11:59pm on 2/21/2018 in Gradescope. Please submit a pdf file. You may submit a scanned handwritten document, but a typed submission is preferred. It will be extremely helpful if in your submission, you start each question on a new page.

1. **Graph short answer.**

   (a) **K&T Ch3.Ex7.** Claim: Let G be a graph on n nodes, where n is an even number. If every node of G has degree at least $n/2$, then G is connected. Decide whether you think the claim is true or false, and either give a proof of the claim or give a counterexample.

   (b) **K&T Ch3.Ex9.** Let $G = (V, E)$ be an n node undirected graph containing two nodes s and t, such that the distance between s and t is strictly greater than $n/2$. Show that there must be some node v, not equal to either s or t such that deleting v from G destroys all $s - t$ paths. In other words, the graph G' obtained by deleting v contains no paths from s to t.

   (c) **Weighted Graphs.** Suppose we have two weighted graphs $G_1 = (V, E, w)$ and $G_2 = (V, E, w')$ where $w'(e) = w(e)+1$ and w, w' are the *edge weights* in the two graphs. The graphs are identical, except that all the edges weights in G_2 are one larger than the corresponding edge in G_1. Suppose that in G_1, we have computed the shortest weighted path from some fixed node $s \in V$ to some other node $t \in V$. Call this path p. Is it always the case that p is the shortest $s \to t$ path in G_2? If it is, then prove it, otherwise give a counterexample.

2. **Directed Graphs.** Given a directed acyclic graph G, give a linear time algorithm to determine if the graph has a directed path that visits every vertex.

3. **K&T Ch4.Ex3.** You are consulting for a trucking company that does a large amount of business shipping packages between New York and Boston. The volume is high enough that they have to send a number of trucks each day between the two locations. Trucks have a fixed limit W on the maximum amount of weight they are allowed to carry. Boxes arrive at the New York station one by one, and each package i has a weight w_i. The trucking station is quite small, so at most one truck can be at the station at any time. Company policy requires that boxes are shipped in the order they arrive; otherwise, a customer might get upset upon seeing a box that arrived after his make it to Boston faster. At the moment, the company is using a simple greedy algorithm for packing: they pack boxes in the order they arrive, and whenever the next box does not fit, they send the truck on its way.

   But they wonder if they might be using too many trucks, and they want your opinion on whether the situation can be improved. Here is how they are thinking. Maybe one could decrease the number of trucks needed by sometimes sending off a truck that was less full, and in this way allow the next few trucks to be better packed.

   Prove that, for a given set of boxes with specified weights, the greedy algorithm currently in use actually minimizes the number of trucks that are needed. Your proof should follow the type of analysis used in

the book for the Interval Scheduling Problem: it should establish the optimality of this greedy packing algorithm by identifying a measure under which it stays ahead of all other solutions.

4. **K&T Ch4.Ex14.** You're working with a group of security consultants who are helping to monitor a large computer system. There's particular interest in keeping track of processes that are labeled "sensitive". Each such process P has a designated start time and finish time, and it runs continuously between these times; the consultants have a list of the planned start and finish times of all sensitive processes that will be run that day.
   As a simple first step, they've written a program called `statusCheck` that, when invoked, runs for a few seconds and records various pieces of logging information about all sensitive processes running on the system at that moment. (We'll model each invocation of `statusCheck` as lasting for only this single point in time.) What they'd like to do is to run `statusCheck` as few times as possible during the day, but enough that for each sensitive process P, `statusCheck` is invoked at least once during the execution of process P.

   (a) Give an efficient algorithm that, given the start and finish times of all the sensitive processes, finds a small a set of times as possible at which to invoke `statusCheck`, subject to the requirement that `statusCheck` is invoked at least once during each sensitive process P.

   (b) While you were designing your algorithm, the security consultants were engaging in a little back-of-the-envelope reasoning. "Suppose we can find a set of k sensitive processes with the property that no two are ever running at the same time. Then clearly your algorithm will need to invoke `statusCheck` at least k times: no one invocation of `statusCheck` can handle more than one of these processes."
   This is true, of course, and after some further discussion you all begin wondering whether something stronger is true as well, a kind of converse to the above argument. Suppose that k^* is the largest value of k such that one can find a set of k sensitive processes with no two ever running at the same time. Is it the case that there must be a set of k^* times at which you can run `statusCheck` so that some invocation occurs during the execution of each sensitive process? (In other words, the kind of argument in the previous paragraph is really the only things forcing you to need a lot of invocations of `statusCheck`.) Decide whether you think this claim is true or false, and give a proof or counterexample.

5. **(0 points).** How long did it take you to complete this assignment?