Assignment 4

CSCI 3110: Design and Analysis of Algorithms

Due June 11, 2019

Banner ID: ____________________ Name: ________________________________

Banner ID: ____________________ Name: ________________________________

Banner ID: ____________________ Name: ________________________________

Assignments are due on the due date before class and have to include this cover page. Plagiarism in assignment answers will not be tolerated. By submitting their answers to this assignment, the authors named above declare that its content is their original work and that they did not use any sources for its preparation other than the class notes, the textbook, and ones explicitly acknowledged in the answers. Any suspected act of plagiarism will be reported to the Faculty's Academic Integrity Officer and possibly to the Senate Discipline Committee. The penalty for academic dishonesty may range from failing the course to expulsion from the university, in accordance with Dalhousie University's regulations regarding academic integrity.

1. (10 pts) You are given a strongly connected directed graph $G = (V, E)$ with positive edge weights along with a particular node $v_0 \in V$. Describe an $\mathrm{O}((n + m) \lg n)$ time algorithm for finding shortest paths between all pairs of nodes, with the one restriction that these paths must all pass through v_0.

2. (20 pts) Here's yet another scheduling problem, but one that cannot easily be solved exactly, particularly not using a greedy algorithm: You are given a set of tasks $t_1, \ldots, t_n$; task t_i has a length ℓ_i. More precisely, each task t_i is a job to be executed by a computer, and it takes ℓ_i time to complete the job. You are now given m computers $C_1, \ldots, C_m$ on which you can schedule these tasks. If your schedule assigns tasks $t_{i_1}, \ldots, t_{i_k}$ to computer C_j, with the first task starting at time 0 and without time intervals between two tasks when there is no task running, the last task assigned to C_j ends at time $f_j = \sum_{h=1}^{k} \ell_{i_h}$. The *make span* of your schedule is defined as $\max\{f_1, \ldots, f_m\}$. Your goal is to compute a schedule with minimum make span. This is a hard problem, though, which is why we relax things a little bit.

   a. Develop a simple greedy algorithm whose goal it is to compute a schedule—that is, an assignment of tasks to computers—whose make span is at most twice the minimum make span for the given set of tasks using m machines. The running time of your algorithm should be $\mathrm{O}(nm)$.

   b. Prove that the schedule produced by your algorithm has make span at most twice the minimum make span.

   c. Provide an example where your algorithm fails to compute a schedule with minimum make span. (This proves that your algorithm does not always produce an optimal answer.)