CS7800: Advanced Algorithms. Fall 2017
Homework 2

Instructor: Jonathan Ullman TA: Albert Cheu

Due Friday, September 22 at 11:59pm
(Email to neu.cs7800@gmail.com)

Homework Guidelines

Collaboration Policy. Collaboration on homework problems is permitted, however it will serve you well on the exams if you solve the problems by yourself. If you choose to collaborate, you may discuss the homework with at most 2 other students currently enrolled in the class. Finding answers to problems on the Web or from other outside sources (these include anyone not enrolled in the class) is strictly forbidden.

You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem. You must also identify your collaborators. If you did not work with anyone, indicate that on your submission. If asked, you must be able to explain your solution to the instructors.

Preparing and Submitting Solutions. You must type your solutions using LaTeX. Please use an 11-pt or larger font. Please submit both the source and PDF files using the naming conventions `lastname_hw2.tex` and `lastname_hw2.pdf`. Your name must be on the first page of the PDF.

Solution guidelines For problems that require you to provide an algorithm, you must give the following: (1) a precise description of the algorithm in English and, if helpful, pseudocode, (2) a proof of correctness, (3) an analysis of running time. You may use any facts from class in your analysis and you may use any algorithms from class as subroutines in your solution.

You should be as clear and concise as possible in your write-up of solutions. Communication of technical material is an important skill, so clarity is as important as correctness. A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error and because it is easier to read and understand. Points might be subtracted for solutions that are too long.

Greedy Algorithms

Problem 1 (10 pts)**.**

The entire population of New York City has suddenly realized that New York City is a horrible place to live. There are trains lined up ready to take everyone out of the city. Every escapee has some amount of cargo c_i and every train has the same cargo capacity C. All the escapees are in a line ready to be loaded onto a train, and your job is decide when to send one train out of town and start loading the next one so as to use the fewest number of trains.

More formally, you are given an ordered list of cargo sizes $c_1, \ldots, c_n$ for the n passengers and a cargo capacity C. You must output a partition of the n passengers into intervals $T_1, \ldots, T_k$ containing sets of consecutive passengers such that every train contains at most C units of cargo total. Your objective is to make k as small as possible.[1]

Design a polynomial time greedy algorithm that solves this problem. Clearly describe your algorithm, prove that it is correct, and analyze its running time.

Problem 2 (10 pts)**.**

You are designing an MST algorithm that runs on a machine with a small workspace. The input to your algorithm is an undirected graph stored on a remote server and presented in the following form: an integer $n = |V|$, followed by a list of m edges along with their weights, that is, a list of triples (u, v, w_{uv}), where $u, v \in \{1, ..., n\}$ and $w_{uv} > 0$. For simplicity, you may assume that the graph is connected and that all edge weights are distinct.

Give an algorithm that makes a single pass through the list of edges, using only $O(n)$ space, and outputs a minimum spanning tree at the end of its pass. Your algorithm should work no matter what order the edges are listed in.[2] Clearly describe your algorithm, prove that it outputs a minimum spanning tree, and analyze is running time. Faster algorithms will receive slightly more credit than slower algorithms, but your algorithm need not run as fast as the ones we saw in class, as long as it uses polynomial time and the desired $O(n)$ space.

[1] **Example:** You have a list of cargo sizes $\{c_1 = 1, c_2 = 3, c_3 = 2, c_4 = 3, c_5 = 1\}$, and $C = 5$, one optimal solution is to use three trains and set $T_1 = \{1\}$, $T_2 = \{2, 3\}$, and $T_3 = \{4, 5\}$. You cannot use two trains by setting $T_1 = \{1, 2, 5\}$ and $T_2 = \{3, 4\}$, since T_1 would not be contiguous. You also cannot use two trains by setting $T_1 = \{1, 2, 3\}$ and $T_2 = \{4, 5\}$, since $c_1 + c_2 + c_3 = 6 > 5$, so T_1 would be overfilled.

[2] **Hint:** As each new edge is considered, decide whether or not to add it to your tree and, if necessary, which edges from the current tree to discard. Use the cut property and/or the cycle property to prove correctness.