CS 577: Introduction to Algorithms — Homework 6

Out: 03/08/18 — **Due: 03/15/18**

Ground Rules

- The homework is worth 5 points (out of a total of 100 points you can accumulate in this course).
- The homework is to be done and submitted in pairs. You can partner with someone from either section.
- The homework is due at the beginning of either lecture on the due date.
- No extensions to the due date will be given under any circumstances.
- Turn in your solution to each problem on a **separate sheet** of paper (or sheets stapled together), with your names clearly written on top.

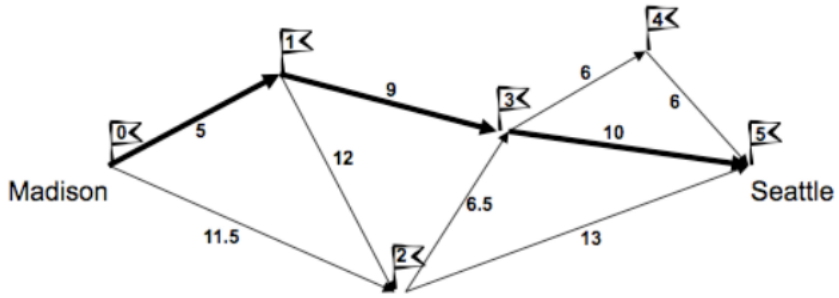
Note

- When we ask for an efficient algorithm, it means that the running time for the algorithm should be some polynomial in the size of the problem. For graph problems, the size is $|V| + |E| = n + m = O(n^2)$, so we require an algorithm with running time $O(n^c)$ for some constant c.
- When we ask you to analyze an algorithm, provide a proof of correctness as well as an analysis of running time.

Problems

1. **(Worth: 2 points. Page limit: 1 sheet; 2 sides)** During thanksgiving break Alice is planning to make a road trip from Madison to Seattle, and wants to stop at various tourist locations along the way for sight-seeing. The locations are numbered from 1 through n with n being Seattle, and this is the order that Alice wants to visit them in. Furthermore, since she has a limited amount of time for the trip, she wants to spend no more than x hours driving.

   Your goal is to help Alice plan her trip. You are given a directed graph over locations with each edge between two locations specifying the amount of time it takes to drive from one to the other. Design an algorithm that returns a path from Madison to Seattle with total driving time $\leq x$ hours, and that visits the maximum possible number of locations enroute. Your algorithm should run in time polynomial in the size of the graph (number of vertices and edges), independent of x. An algorithm running in time polynomial in the size of the graph as well as x will receive partial credit.



Example for problem 1: The optimal path for x=25 is shown in bold.

2. **(Worth: 3 points. Page limit: 1 sheets; 2 sides)** In a tribute to Dr. Seuss,[1] here is a question based on his story "Yertle the turtle". You don't need to know the story to solve the question.

   Mack, in an effort to avoid being cracked, has enlisted your advice as to the order in which turtles should be dispatched to form Yertle's throne. Each of the turtles ordered by Yertle has a different weight and strength. Your task is to build the largest stack of turtles possible such that for every turtle in the stack, its strength is no less than the total weight that it carries above it.

   You are given as input the weight and the strength of each turtle. The strength is the turtle's overall carrying capacity, including its own weight. That is, a turtle weighing 300 units with a strength of 1000 units could carry other turtles weighing a total of 700 units on its back.

   Your algorithm should output the maximum number of turtles that can be stacked without exceeding the strength of any one. It should run in time $O(n^2)$, where n denotes the number of turtles given. You will be given partial credit for an algorithm that runs in time polynomial in n and the weights or strengths of the turtles.

   Hint: Think about the order in which turtles should be stacked. Then try to come up with a recurrence that leads to an O(nW) time algorithm, where W is the total weight of all the turtles. Then think about how to change your recurrence to get a better running time.

[1] Last Friday, March 2, was celebrated as "Read Across America" day in honor of Dr. Seuss' birthday.