CS 577: Introduction to Algorithms — Homework 6

Out: 11/03/10 — **Due: 11/10/10**

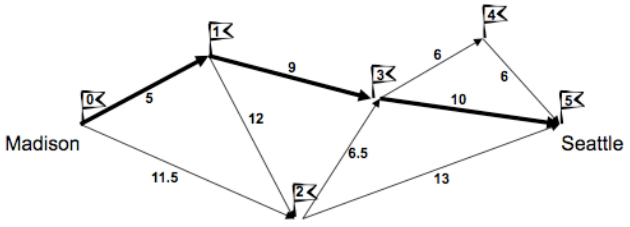
Ground Rules

- **Grading.** You will be graded on the correctness as well as clarity of your solutions. You are required to prove any claims that you make. In particular, when you are asked to design an algorithm, you must argue its correctness as well as running time. You may use without proof any theorems proven in class or in the textbook, as long as you state and cite them properly.
- **Collaboration.** You may work on and submit solutions in pairs.
- **Lateness.** Homework is due promptly at the start of class. Late homework will receive zero credit.
- **Extra credit questions.** There are no extra credit questions on this homework.
- Start working on your homework early. Plan your work in such a way that you have the opportunity to put some problems on the back burner for a while and revisit them later. Good luck!

Problems

1. **(5 points)** 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 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. **(5 points)** In a tribute to Dr. Seuss, here is a question based on his story "Yertle the turtle". You don't need to know the story to solve the question.

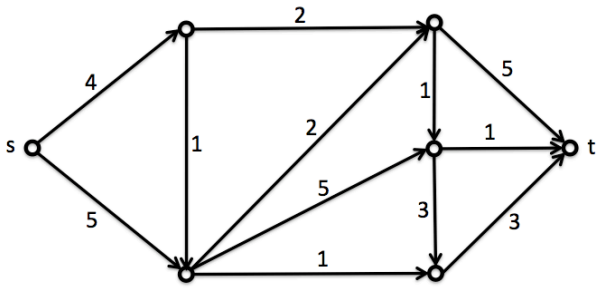
   Mack, in an effort to avoid being cracked, has enlisted your advice as to the order in which turtles should be dispatched to form Yertle's throne. Each of the turtles ordered by Yertle has a different weight and strength. Your task is to build the largest stack of turtles possible such that for every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

   You are given 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.

3. **(2+3 points)** For this question, you are given a directed network $G = (V, E)$ with capacities c_e on edges and a source-sink pair (s, t). In part (b) you are also given the max s-t flow in the graph, f^*. An edge in the network is called a *bottleneck* edge if increasing its capacity by one unit increases the max flow in the network.

   (a) For the network below determine the max s-t flow, f^*, a min s-t cut (note that this may not be unique), and the residual graph G_{f^*}. Also identify all of the bottleneck edges in the graph.



   (b) Develop an algorithm for determining all the bottleneck edges in the given graph. Your algorithm is allowed to use the max flow f^* and should run in linear time.