CS 577: Introduction to Algorithms — Homework 6

Out: 03/02/16 — **Due: 03/09/16**

Ground Rules

- See HW1.
- Both problems will be graded. Please submit each problem on a different sheet of paper.
- For each DP-based algorithm, clearly provide: (1) the definition of the recursive function you are computing; (2) the recurrence or Bellman equation for this function; (3) a proof of correctness for the equation; (4) an algorithm based on the equation; (5) an analysis for the running time of the algorithm.

Problems

1. During spring break you are planning to make a road trip from Madison to Miami, and want to stop at various tourist locations along the way for sight-seeing. You have a map (in the form of a graph) of all the locations showing the time it takes to go from any one location to another. You have a limited amount of time for the trip, so you want to spend no more than x hours driving. Furthermore, you want to plan your trip in such a way that each location you visit is closer to Miami than the previous one.

   Design a DP-based algorithm that returns a route from Madison to Miami with total driving time at most 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. A pseudo-polynomial time algorithm running in time polynomial in the size of the graph as well as x will receive partial credit.

2. In a tribute to Dr. Seuss[1], here is a problem based on his story "Yertle the Turtle". You don't need to know the story to solve the problem.

   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 available 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 of all the turtles it supports, including itself (you may ignore Yertle's own weight).

   Formally, you are given the weight and the strength for each of n turtles. 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. (You need not output the subset of turtles that form the optimal stack.) Develop a DP-based algorithm for this problem that runs in time $O(n^2)$, where n is the total 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] Today, March 2, is celebrated as National Read Across America Day in honor of Dr. Seuss' birthday.