EE360C: Algorithms
University of Texas at Austin — Problem Set #9
Dr. Evdokia Nikolova & Dr. Pedro Santacruz — Due: November 20, 2014

Problem Set #9

You should try to solve these problems by yourself. I recommend that you start early and get help in office hours if needed. If you find it helpful to discuss problems with other students, go for it. **You do not need to turn these problems in. The goal is to be ready for the in class quiz that will cover the same or similar problems.**

Problem 1: Longest Paths

Let $G = (V, E)$ be a directed graph with nodes $v_1, v_2, \ldots, v_n$. We say that G is an ordered graph if it has the following properties.

- Each edge goes from a node with a lower index to a node with a higher index. That is, every directed edge has the form (v_i, v_j) with $i < j$.
- Each node except v_n has at least one edge leaving it. That is, for every node v_i, $i = 1, 2, \ldots, n-1$, there is at least one edge of the form (v_i, v_j).

The *length* of a path is the number of edges in the path. The goal in this question is to solve the following problem

> Given an ordered graph G, find the length of the longest path that begins at v_1 and ends at v_n.

Give an efficient algorithm using dynamic programming approach that takes an ordered graph G and returns the length of the longest path that begins at v_1 and ends at v_n.

Problem 2: Dynamic Programming

It's the end of the semester, and you're taking n courses, each with a final project. Each project will be graded on a scale of 1 to $g > 1$, where a higher number is a better grade. Your goal is to maximize your average grade on the n projects. (Note: this is equivalent to maximizing the *total* of all grades, since the difference between the total and the average is just the factor n.)

You have a finite number of hours $H > n$ to complete all of your course projects; you need to decide how to divide your time. H is a positive integer, and you can spend an integer number of hours on each project. Assume your grades are deterministically based on time spent; you have a set of functions $\{f_i : 1, 2, \ldots, n\}$ for your n courses; if you spend $h \leq H$ hours on the project for course i, you will get a grade of $f_i(h)$. The functions f_i are nondecreasing; spending more time on a course project will not *lower* your grade in the course.

To help get you started, think about the (i, h) subproblem. Let the (i, h) subproblem be the problem that maximizes your grade on the first i courses in at most h hours. Clearly, the complete solution is the solution to the (n, H) subproblem. Start by defining the values of the $(0, h)$ subproblems for all h and the $(i, 0)$ subproblems for all i (the latter corresponds to the grade you will get on a course project if you spend *no* time on it).

Give the dynamic programming equation to define the value of the optimal solution (i.e., the value of any (i, h) subproblem, where $0 \leq i \leq n$ and $0 \leq h \leq H$), describe an algorithm for iteratively computing the value of the optimal solution, and describe an algorithm for recreating the optimal solution (i.e., mapping your H hours to your n projects).

Problem 3: Network Flow Theory

Decide whether you think each of the following is true or false. If it is true, give a brief explanation. If it is false, give a counter example.

1. Let G be an arbitrary flow network with a source s, a sink t, and a positive integer capacity c_e on every edge e. If f is a maximum s-t flow in G, then f saturates every edge out of s with flow (i.e., for all edges e out of s, we have $f(e) = c_e$.

2. Let G be an arbitrary flow network with a source s, a sink t, and a positive integer capacity c_e on every edge e, and let (A, B) be a minimum s-t cut with respect to these capacities $\{c_e : e \in E\}$. Now suppose we add 1 to every capacity. Then (A, B) is still a minimum s-t cut with respect to these new capacities $\{1 + c_e : e \in E\}$.

Problem 4: Network Flow

Network flow issues come up in dealing with natural disasters and other crises, since major unexpected events often require the movement and evacuation of large numbers of people in a short amount of time.

Consider the following scenario. Due to large-scale flooding in a region, paramedics have identified a set of n injured people distributed across the region who need to be rushed to hospitals. There are k hospitals in the region, and each of the n people needs to be brought to a hospital that is within a half-hour's driving time of their current location(so different people will have different options for hospitals, depending on where they are right now).

At the same time, one doesn't want to overload any one of the hospitals by sending too many patients its way. The paramedics are in touch by cell phone, and they want to collectively workout whether they can choose a hospital for each of the injured people in such a way that the load on the hospital is balanced: Each hospital receives at most $\lceil n/k \rceil$ people.

Give a polynomial-time algorithm that takes the given information about the people's locations and determines whether this is possible