CS 161 Winter 2021 Section 7

February 2021

Warmup

For each of the following shortest path problems, choose which algorithm you will use.

1. All-Pairs Shortest Path on a graph with (possibly negative) edge weights
2. All-Pairs Shortest Path on a graph with non-negative edge weights
3. Single Source Shortest Path on a graph with non-negative edges weights
4. Single Source Shortest Path on a graph with (possibly negative) edge weights

Edsger's Apfelstrudel

You are eating at a cozy little restaurant which serves a prix fixe menu of $k + 1$ courses, with several available choices for each course. Each dish belongs to exactly one course (e.g., risotto can only be ordered as an appetizer, not a main), and you are effectively indifferent between most of the items on the menu (because they are all so tasty), but the main draw of this particular restaurant is that they serve a delicious 'bottomless' dessert: their world-famous Viennese-style apple strudel. They have an unlimited supply of this apple strudel, but each serving will still cost you $1. The restaurant also has a few interesting rules:

1. You must finish your current dish before ordering another.
2. Each dish after the first course depends on what you ordered in the previous course, e.g., you can only order salmon for your main if you ordered a Caesar salad or chicken noodle soup for the previous course. You are told on the menu exactly what these restrictions are before you order anything.
3. Most importantly, you are not allowed to have their unlimited dessert unless you finish one dish from each of the first k courses!

You are told the cost of each item in each course on the menu, and you plan your meal with a twofold goal: to be able to order the strudel, but also to save as much money as possible throughout the first k courses so that you have more money to spend on the unlimited dessert. Design an algorithm to find the smallest amount of money you can spend on the first k courses and still order the 'bottomless' strudel. If you would like, you may assume the very first course has exactly one choice (e.g., a single complimentary leaf of spinach that costs 0 dollars).

Currency Exchange

Suppose the various economies of the world use a set of currencies $C_1, \ldots, C_n$ — think of these as dollars, pounds, bitcoins, etc. Your bank allows you to trade each currency C_i for any other currency C_j at an exchange rate r_{ij}, that is, you can exchange each unit of C_i for $r_{ij} > 0$ units of C_j. Due to fluctuations in the markets, it is occasionally possible to find a sequence of exchanges that lets you start with currency A,

change into currencies, B, C, D, etc., and then end up changing back to currency A in such a way that you end up with more money than you started with. That is, there are currencies $C_{i_1}, \ldots, C_{i_k}$ such that

$$r_{i_1 i_2} \times r_{i_2 i_3} \times \cdots \times r_{i_{k-1} i_k} \times r_{i_k i_1} > 1.$$

This is called an arbitrage opportunity, but to take advantage of it you need to be able to identify it quickly (before other investors leverage it and the exchange rates balance out again)! Devise an efficient algorithm to determine whether an arbitrage opportunity exists. Justify the correctness of your algorithm and its runtime.

Rod Cutting

Suppose we have a rod of length k, where k is a positive integer. We would like to cut the rod into integer-length segments such that we maximize the *product* of the resulting segments' lengths. Multiple cuts may be made. For example, if $k = 8$, the maximum product is 18 from cutting the rod into three pieces of length $3, 3$, and 2. Write an algorithm to determine the maximum product for a rod of length k.