University of Washington
Department of Computer Science and Engineering
CSE 421, Autumn 2019

November 18, 2019

Homework 8, Due Wednesday, November 20, 2019

Problem 1 (10 points):

(Page 324, Exercise 13.) The problem of searching for cycles in graphs arises naturally in financial trading applications. Consider a firm that trades shares in n different companies. For each pair $i \neq j$, they maintain a trade ratio r_{ij}, meaning that one share of i trades for r_{ij} shares of j. Here we allow the rate r to be fractional; that is, $r_{ij} = \frac{2}{3}$ means that you can trade three shares of i to get two shares of j.

A *trading cycle* for a sequence of shares $i_1, i_2, \ldots, i_k$ consists of successively trading shares in company i_1 for shares in company i_2, then shares in company i_2 for shares i_3, and so on, finally trading shares in i_k back to shares in company i_1. After such a sequence of trades, one ends up with shares in the same company i_1 that one starts with. Trading around a cycle is usually a bad idea, as you tend to end up with fewer shares than you started with. But occasionally, for short periods of time, there are opportunities to increase shares. We will call such a cycle an *opportunity* cycle, if trading along the cycle increases the number of shares. This happens exactly if the product of the ratios along the cycle is above 1. In analyzing the state of the market, a firm engaged in trading would like to know if there are any opportunity cycles.

Give a polynomial-time algorithm that finds such an opportunity cycle, if one exists.

Problem 2 (10 points):

Give an algorithm, which given a directed graph $G = (V, E)$, with vertices $s, t \in V$ and an integer k, determines the number of paths from s to t of length k. Your algorithm should be polynomial in k, $|V|$ and $|E|$.

Problem 3 (10 points):

Give an $O(n^2)$ algorithm for finding the longest strictly increasing subsequence of a sequence of n integers. (Note that this problem can be solved in $O(n \log n)$ time by a non-dynamic programming style algorithm, but you do not need find it. Use of dynamic programming for this problem is recommended, but not required.)

Problem 4 (10 points):

The Chvatal-Sankoff constants are mathematical constants that describe the length of the longest common subsequences of random strings. Given parameters n and k, choose two length n strings A and B from the same k-symbol alphabet, with each character chosen uniformly at random. Let

$\lambda_{n,k}$ be the random variable whose value is the length of the longest common subsequence of A and B. Let $E[\lambda_{n,k}]$ denote the expectation of $\lambda_{n,k}$. The Chvatal-Sankoff constant γ_k is defined at

$$\gamma_k = \lim_{n \to \infty} \frac{E[\lambda_{n,k}]}{n}.$$

Experimentally determine (by implementing an LCS algorithm), the smallest value of k, such that $\gamma_k < \frac{2}{5}$. In other words determine how large an alphabet needs to be so that the expected length of the LCS of two random strings is less than 40% the length of the strings.