University of Washington
Department of Computer Science and Engineering
CSE 417, Autumn 2020

December 1, 2020

Homework 8, Due Friday, December 4, 2020

On all problems provide justification of your answers. Provide a clear explanation of why your algorithm solves the problem, as well as a justification of the run time.

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) Word segmentation:

(This problem is based on problem 5 on Page 316 of the text without the excessive verbiage.) The word segmentation problem is: given a string of characters $Y = y_1y_2 \ldots y_n$, optimally divide the string into consecutive characters that form words. (The motivation is that you are given at text string without spaces and have to figure out what the words are. For example, *"meetateight"* could be *"meet ate ight"*, *"me et at eight"* or *"meet at eight"*.) The problem is to find best possible segmentation. We assume we have a function *Quality* which returns an integer value of the goodness of a word, with strings that correspond to words getting a high score and strings that do not correspond to words getting a low score. The overall quality of a segmentation is the sum of the qualities of the individual.

Give an dynamic programming algorithm to compute the optimal segmentation of a string. You can assume that calls to the function *Quality* take constant time and return an integer value.

Problem 3 (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.

Problem 4 (10 points):

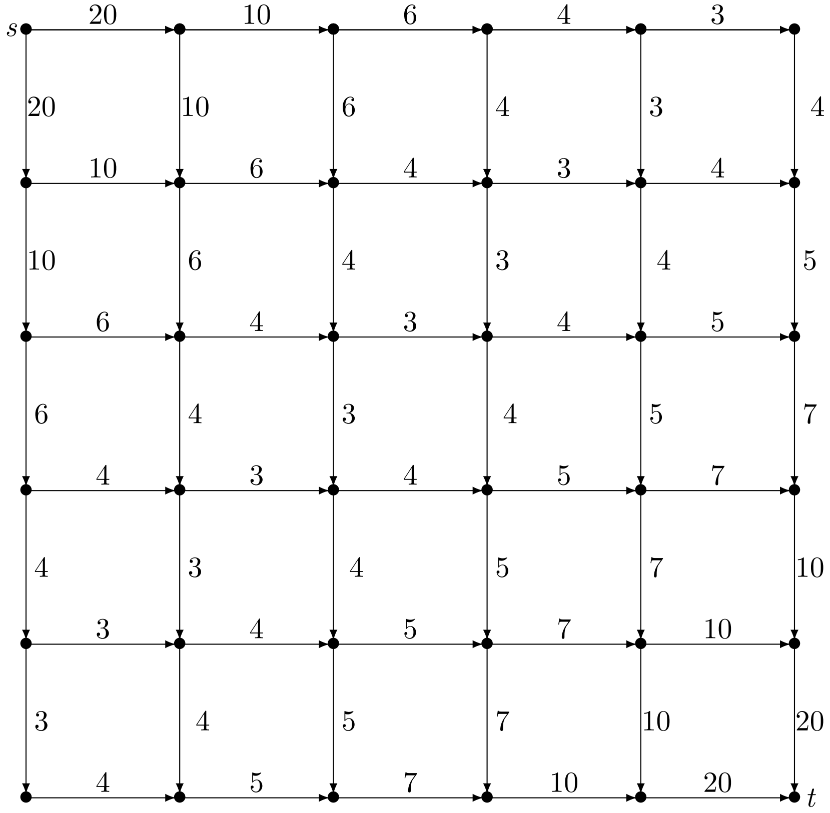
In a standard $s - t$ Maximum-Flow Problem, we assume edges have capacities, and there is no limit on how much flow is allowed to pass through a node. In this problem, we consider the variant of the Maximum-Flow problem with node capacities.

Let $G = (V, E)$ be a directed graph, with source $s \in V$, sink $t \in V$, and nonnegative node capacities $\{c_v \geq 0\}$ for each $v \in V$. Given a flow f in this graph, the flow through a node v is defined as $f^{\text{in}}(v)$, the sum of the flows on the incoming edges to v. We say that a flow is feasible if it satisfies the usual flow-conservation constraints and the node-capacity constraints: $f^{\text{in}}(v) \leq c_v$ for all nodes.

Give a polynomial-time algorithm to find an $s-t$ maximum flow in such a node-capacitated network. Justify the correctness of your algorithm. (Note and hint: you may call an $O(n^3)$ subroutine for the standard network flow problem

Problem 5 (10 points):

Consider the following flow graph. Find a maximum flow.



a) What is the value of the maximum flow? Indicate the value of flow on each edge.

b) Prove that your flow is maximum.

Problem 6 (10 points):

Decide whether you think the following statement is true or false. If it is true, give a short explanation. If it is false, give a counterexample.

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 $\{1 + c_e : e \in E\}$.