CSE 421 Winter 2021
Homework 6

Due: Friday, February 26, 2021, 6:00 pm

Problem 1:
When people type, they are much more likely to transpose two adjacent characters typed with different hands rather than insert, delete, or change them. Suppose that there is a partition of the alphabet into two disjoint parts L and R with a cost 1 per transposition if the letters are from opposite parts of the alphabet. Other transpositions are not allowed. Suppose that it is cost 2 for either an insertion or deletion and cost e_{ab} with $1 < e_{ab}$ for typing an a instead of a b. (The likelihood of a mis-typed character depends on how far apart the characters are on the keyboard.) Also assume that the alphabet is of a fixed constant size and that a single letter cannot be counted in more than one transposition so that you can transpose anm to nam but not to nma.

Clarification: Characters that have been transposed can also be involved in mismatches, but only if each mismatch is in the same portion of the alphabet, L or R, as the character that it replaced.

For example, if we assume that a and b are in L and j and k are in R and we start with ak:

- at cost 1 (transposition) we can get ka
- at cost $1 + e_{kj}$ (transposition + mismatch) we can get ja
- at cost $1 + e_{kj} + e_{ab}$ (transposition + two mismatches) we can get jb.

However, we would not be allowed to use a transposition+mismatch to get ba or kj because these would involve mismatches that cross between L and R. (Of course, these could also be produced other ways via insertions and deletions and other mismatches, as can all changes.)

Design an efficient algorithm to compute the edit distance between two strings under this measure and determine the sequence of errors needed to achieve this distance.

Problem 2:
Using *arbitrage* one can take advantage of differences in the prices of items in different markets. That is, one can purchase units of a good in one market A at a low price p and sell them in another market B at a higher selling price s and then pocket the difference.

In the case when there are no other transaction costs, we can represent a single instance of such arbitrage as a weighted directed graph with 3 nodes: One node u represents a unit of money (in some fixed currency, say dollars), another node v represents a unit of the good in market A, and the third node w represents a unit of the good in market B. We put an edge from u to v with weight equal to the inverse $1/p$ of the purchase price p in market A, which indicates that a single dollar gives you $1/p$ of the item in market A. We include an edge from v to w with weight 1 representing that a unit of the good in market A becomes a unit of the same good in market B. Finally, since the sales price in market B is s there would be an edge of weight s indicating that 1 item in market B becomes s dollars. Assuming that $s > p$, by following the transactions given by the edges of this cycle once, starting with dollars, we can increase our dollars by an s/p factor, which is the product of the edge weights in the cycle.

In general, there can be much more complicated kinds of arbitrage possible that don't involve immediately converting back to money in the same currency. One could trade goods in one market for a different goods/currency in a different market and repeat this several times before getting a profit (and that profit doesn't have to be realized in any particular currency or location). Each time, it is merely the opportunity for a sequence of trades that leaves one with more of some good than one started with.

Suppose that you are given a directed graph $G = (V, E)$ with n vertices and m edges, with each edge (u, v) labeled by a weight $s_{uv} > 0$ representing the amount of the good at v you could get by trading a unit of the good at u for it. (Don't worry that you might only be able to buy an integer amount of the good at v; this is the kind of thing that hedge funds do, and you should imagine that the starting stake is so high that any rounding errors in the number of goods at v you can actually purchase don't actually matter.) An *arbitrage opportunity* in such an input graph is any cycle such that the product of the weights in the cycle is > 1. (By trading around such a cycle one would end up with more than one started with.)

Give a polynomial-time algorithm that, given such a weighted directed graph as input, can find an arbitrage opportunity if one exists.

Problem 3:
Suppose that you have a flow network G with source s and sink t.

This problem is about understanding something more that will help to narrow down the nodes/edges that might be potentially involved in bottlenecks in such a network.

We say that a node v is:

- *upstream* of any st-bottleneck iff for every minimum st-cut of G, v is on the side of the cut that includes S,
- *downstream* of any st-bottleneck iff for every minimum st-cut of G, v is on the side of the cut that includes t, and
- *risky* otherwise.

Only the edges that touch risky nodes are of potential concern with respect to bottlenecks.

Give an algorithm that runs in time within a constant factor of a single a maxflow computation that classifies every node of G as either upstream, downstream, or risky.

Problem 4 (Extra credit):
Suppose that you have a set S of possible labels and are given a directed graph $G = (V, E)$ with a designated start node s with each edge (u, v) having a label $L(u, v)$ from S. (Note that multiple edges out of a node may have the same label.) In addition, each edge has a probability $p(u, v) \geq 0$ of being taken when at u. In other words, for every $u \in V$,

$$\sum_{v:\ (u,v)\in E} p(u, v) = 1.$$

The probability of taking a path beginning at the start node s is the product of the probabilities labeling its edges. Produce an efficient algorithm that takes as input the graph G with its edge labels and edge probabilities and a sequence of labels $a_1, \ldots, a_t$ and determines the most likely path beginning at node s that is consistent with the sequence of labels (and determine the probability of that path). You can assume that arithmetic operations on real numbers have cost 1.