CSE 60111: Complexity and Algorithms
Homework 5

Reminder: For any algorithms question (unless otherwise specified), you must provide the following:

- an algorithm, meaning clear pseudocode, as well as perhaps a brief explanation of the pseudocode if you believe it helpful to clarify the code;
- an analysis of the runtime and space complexity of your algorithm;
- and a proof of correctness for your algorithm.

Each component is required and graded, so be sure you include all of them!

---

1. Revisiting NP-Hard problems (or why DAGs matter!): Let G be a directed acyclic graph.

   (a) A Hamiltonian path is a directed path in G that visits every vertex once. Describe an algorithm to determine whether G has a Hamiltonian path.

   (b) Now given an algorithm to count the number of s to t paths in the graph. (Assume that you can add any two numbers is $O(1)$ time.). [Note: this problem in general graphs is actually #P-complete, so even harder than NP-Hard!).

2. You find yourself in a new city with two overlapping public transit systems, one using trams and the other buses. Both transit modes use the same payment system. Whenever you board a bus or tram, you scan your transit card; whenever you leave a bus or tram, you scan your card again. A centralized computer system charges your card based on the distance you travel. (Passengers who "forget" to scan their cards are subject to heavy fines, so don't do that.)

   To attract riders, each transit system offers a discount for multiple consecutive rides. Any ride on the same type of vehicle (bus or tram) as the previous ride is automatically given a 50% discount. Your goal is to find the cheapest trip from your home to your work, strictly on public transit, taking advantage of these discounts.

   More formally, you are given a directed graph G (whose vertices represent bus/tram stops), where every edge is colored either Red or Blue (indicating whether the edge represents a tRam or Bus ride), and every edge $u \to v$ has a non-negative cost $\$(u \to v)$. You are also given two nodes s and t, representing your home and your work. The discounted cost of a walk $W = v_0 \to v_1 \to v_2 \to \cdots \to v_k$ in G is

   $$\sum_{i=1}^{k} \text{discount}(W, i) \cdot \$(v_{i-1} \to v_i)$$

   where

   $$\text{discount}(W, i) = \begin{cases} \frac{1}{2} & \text{if } i > 1 \text{ and } \mathit{color}(v_{i-1} \to v_i) = \mathit{color}(v_{i-2} \to v_{i-1} \\ 1 & \text{otherwise} \end{cases}$$

   denotes the discount factor for the i^{th} edge $v_{i-1} \to v_i$ of W.

(a) Give an example where the walk in G from s to t with minimum discounted cost is not a simple path. In other words, show that visiting the same bus/tram stop more than once can actually save you money.

(b) Describe and analyze an algorithm to compute minimum discounted cost of a walk in G from s to t. You may assume that G contains at least one walk from s to t. (Your algorithm does not need to return the optimal walk itself, only its discounted cost.) [Hint: I strongly suggest that you look for ways to modify the input graph and use an algorithm we studied as a black box.]

3. In this problem we will discover how you, yes you, can be employed by Wall Street and cause a major economic collapse! The arbitrage business is a money-making scheme that takes advantage of differences in currency exchange. In particular, suppose 1 US dollar buys 120 Japanese yen, 1 yen buys .01 Euros, and 1 euro buys 1.2 US dollars. Then, a trader starting with $1 can convert their money from dollars to yen, then from yen to euros, and finally from euros back to dollars, ending with $1.44! The cycle of currencies (dollar to yen to Euro to dollar) is called an arbitrage cycle. Of course, finding and exploiting arbitrage cycles before the prices are corrected requires extremely fast algorithms.

   Suppose n different currencies are traded in your currency market. You are given the matrix $Exch[1..n, 1..n]$ of exchange rates between every pair of currencies; for each i and j, one unit of currency i can be traded for $Exch[i, j]$ units of currency j. (Do not assume that $Exch[i, j] \cdot Exch[j, i] = 1$, because money is not logical!)

   (a) Describe an algorithm that returns an array $MaxAmt[1..n]$, where $MaxAmt[i]$ is the maximum amount of currency i that you can obtain by trading, starting with one unit of currency 1, assuming there are no arbitrage cycles.

   (b) Describe an algorithm to determine whether the given matrix of currency exchange rates creates an arbitrage cycle.

   (c) Modify your algorithm from part (b) to actually return an arbitrage cycle, if it exists.