Problem 4-1. Graph Coloring [15 points]

Let a graph $G = (V, E)$ be given. If $C$ is a set of colors, a coloring function for $G$, $c_G : V \rightarrow C$, maps every vertex to a color such that adjacent vertices do not have the same color. That is to say, for all $uv \in E$, $c_G(u) \neq c_G(v)$. A graph is said to be $k$-colorable if there exists a coloring function for that graph whose range has size at most $k$. For example, every graph is trivially $|V|$-colorable by mapping each vertex to its own color; a star graph is 2-colorable by picking one color for the fringe vertices and another for the central vertex. The graph coloring problem is to produce a coloring function that uses the smallest possible number of colors. Producing a minimum coloring, i.e. one that uses the fewest possible number of colors, is known to be \textbf{NP}-complete in general, so we’ll be looking for an approximation.

Let $\Delta$ denote the maximum degree in a graph. Develop an algorithm that reduces coloring to MIS to produce a $(\Delta + 1)$-coloring. Prove that your algorithm will in fact generate a $\Delta + 1$-coloring. Also analyze the running time.

Problem 4-2. Cycle Graphs [40 points]

In class, we saw a graph contraction algorithm for the cycle graphs. It flips a coin for each edge and selects an edge to contract if it gets a heads and its two neighbors both come up tails. In class, we analyzed the variant when the probability of heads is 0.5. Now you will analyze some variants of that algorithm. For both of these, on an $n$-vertex cycle $C_n$, compute the probability that an edge $e$ is contracted. In expectation, how many edges are contracted in terms of $n$? Show your work.

(a) [15 points]
We use the same algorithm as in class, but a coin comes up heads with probability $p$ (instead of 0.5). What is the probability of contraction of each edge? What is the value of $p$ to get the maximum probability of contraction?

(b) [25 points]
Now, we’ll analyze a slight variant of the algorithm: the modified algorithm chooses a random number $x_e \in_R [0, 1]$ for each edge $e$ independently—and it selects an edge to contract if that edge gets a larger number than those of its two neighbors.

Problem 4-3. Currency Exchange [15 points]
(a) [10 points]
You are planning to go to Japan, and want to buy Japanese yen. However, you want to get the maximum number of yen for your dollar. Ah! Your knowledge of algorithms is finally going to come in handy. More formally, you want to solve the following problem: Given the a set currencies, a set of exchange rates between them, and a source currency $s$, find for each other currency $v$ the best sequence of exchanges to get from $s$ to $v$.

(b) [5 points]
Now that you are familiar with the currency market, you decide to get rich by trying your hand at arbitrage. You want to see if there is a way to game the market so that you start with $x$ amount of a currency $s$, and after a series of exchanges, you end up with $y$ amount of $s$ where $y > x$. Again, given a set of currencies and a set of exchange rates, give an algorithm to find out if there is a way to commit arbitrage.

Problem 4-4. Simple Graphs (2 parts) [30 points]
You are given a directed graph $G = (V, E)$ with $|V| = n$ and $|E| = m$. The vertices are labeled $1..n$. The graph is represented as an adjacency array $A$. Recall that the adjacency array consists of a array where element $A[i]$ is a pointer to another array that contains the neighbors of vertex $i$.

(a) [15 points] Given a directed graph $G = (V, E)$, you want to compute reachability matrix $R$. A vertex $j$ is reachable from vertex $i$ if and only if there is a path from $i$ to $j$ in $G$. A reachability matrix is an $n \times n$ matrix where element $R[i, j]$ is 1 if $j$ is reachable from $i$ and 0 otherwise. Given an adjacency array for graph $G$, give an efficient algorithm to compute the reachability matrix.

(b) [15 points] Given a directed graph $G = (V, E)$ its transpose is $G^T = (V, E')$ where

$$E' = \{(b, a) | (a, b) \in E\}.$$

Informally, it is another directed graph on the same vertices with every edge flipped. Given an adjacency array representation of a graph $G$, give an efficient algorithm to calculate the adjacency array representation of its transpose $G^T$. 

2