CS 170

1. Study Group

None

2. Arbitrage

(a) **Algorithm Description:**
Create a weighted directed graph G where each vertex v_i represents one kind of currency c_i, and for any pair i, j create an edge from v_i to v_j with weight $log(1/r_{ij})$.
Then run Bellman-Ford algorithm from s on G to find the shortest path from s to t. This path is the most advantageous sequence of currency exchanges for converting currency s into currency t.
Correctness:
Let P be the set of all the possible ways to exchange from s to t. To find the most advantageous sequence of currency exchange, we need to maximize the exchange rate $rate_p$, where $p \in P$. For a particular exchange path $a_0 = t, a_1, a_2, ..., a_k = t$, the exchange rate is $\Pi_{i=0}^{k-1} C_i^p$, where C_i^p is the exchange rate between a_i and a_{i+1} in path p.
Since maximize $\Pi_{i=0}^{k-1} C_i^p$ is equivalent to maximize $log(\Pi_{i=0}^{k-1} C_i^p) = \sum_{i=0}^{k-1} log(C_i^p) = -\sum_{i=0}^{k-1} log(1/C_i^p)$, which is also equivalent to minimize $\sum_{i=0}^{k-1} log(1/C_i^p)$. So if we let $log(1/C_i^p)$ be the weight between a_i and a_{i+1}, this is exactly a shortest-path problem on a weighted-directed graph which may have negative weight. So the Bellman-Ford algorithm is applied.
Runtime:
We can create the graph in O(n^2), and run Bellman-Ford in O(n^3). So the total runtime is O(n^3).

(b) **Algorithm Description:**
Use the same graph representation as for part (a). Run Bellman-Ford algorithm to find if there exists a negative cycle in the graph. If so, there is possibility of arbitrage, and vice versa.
Correctness:
If there exists a negative cycle in the graph, then there is a path $p \in P$, so that $\sum_{i=0}^{k-1} log(1/C_i^p) < 0$, which is equivalent to $\Pi_{i=0}^{k-1} C_i^p > 1$, then $a_0, a_1, ..., a_k$ is a exchange path to arbitrage.
Runtime:
The same as part (a).

2. Money Changing

(a) Let $D_n = \{x_i | i = 1, 2, ..., n\}$, you are able to do this decomposition for all integers $A > 0$ iff $1 \in D_n$. The proof is straightforward. To represent 1, you must have $1 \in D_n$. If you have $1 \in D_n$ and WLOG $x_1 = 1$, then for any A, you can express $A = \sum_{i=1}^{n} a_i x_i$ with $a_1 = A, a_i = 0, 2 \leq i \leq n$.

(b) My *greedy algorithm* is as follows: always choose the bigger denomination if possible.

(c) It is easy to prove by induction.

(d) If the denomination is $\{1, 6, 7\}$, then for $A = 26$, the *greedy algorithm* gives 8, while the optimal is 6.

3. Bounded Bellman-Ford

Algorithm description:
Define dist$[u][i]$ as the shortest path from s to u with at most i path, then use dynamic programming.
t **Pseudocode:**
for all $u \in V$:
    dist$[u]$ = $[\infty]$ * k
dist$[s][0]$ = 0
for j = 1 to k:
    for all $(u, v) \in E$:
      dist$[v][j]$ = min(dist$[v][j]$, dist$[u][j-1]$ + $l(u, v)$)
return dist$[t][k]$
Runtime:
O($k|E|$).
The Official-solution uses rolling-array to reduce the space-complexity.