Stephen Chambers
CS858
April 11th, 2016

Assignment 10 Writeup

1. Exercise 24.2–4 in CLRS.

```
for vertex v in V
   v.pathCount = 0
V = toplogicalSort(V)
for vertex v in V
    for edge e in v.edges()
       v.pathCount += e.pathCount + 1
retval = 0
for vertex v in V
   retval += v.pathCount
return retval
```

Because the input is topologically sorted, given an edge {v,u}, the total number of paths to u is the total number of paths to v + 1 for the edge.

Topological sorting takes O(V+E), initializing the path counts takes O(V) and, updating the path counts takes O(E). Therefore, the total running time for this algorithm is O(V+E).

2. Do the first part of exercise 15.3-6 (the $c_k = 0$ case where you show optimal substructure) and part a of problem 24-3 in CLRS.

15.3-6:

This problem is essentially the longest path problem, which has been proven to have optimal substructure.

Proof by contradiction:

Consider a solution *S* with an optimal dollar amount *D*. Let *i* and *j* be two vertices in *S*. Let d_{ij} be the product of every exchange rate between vertexes *i* and *j*. Suppose there is a vertex *k* that exists between *i* and *j* that will increase d_{ij}. If vertex *k* was included in *S*, *D* would have to increase. This makes our assumption that *D* was optimal false, and proves that this problem exhibits optimal substructure.

24-3 part a:
Consider the representation of this problem as a directed acyclic graph (DAG). Each vertex will have *n* edges with the weights being the exchange rate for each currency 1..2..*n*. We need an algorithm that detects whether or not $r_{01} * r_{12} \ldots * r_{k0} > 1$. Looking at section 24.1 in CLRS, the Bellman-Ford algorithm returns TRUE if there exists no negative cycles, and false if there is a negative cycle. If we could somehow make our cost function negative, we could simply run the Bellman-Ford algorithm and swap the boolean value that we return.

$$r_{01} * r_{12} \ldots * r_{k0} > 1$$

First, let's take the log of both sides to convert the cost function to be strictly positive.

$$\log(r_{01}) * \log(r_{12}) \ldots * \log(r_{k0}) > 0$$

Now, let's negate both sides to get our final cost function.

$$-\log(r_{01}) * -\log(r_{12}) \ldots * -\log(r_{k0}) < 0$$

Run the Bellman-Ford algorithm exactly as it is in section 24.1, but replace FALSE with TRUE and TRUE with FALSE. This will return whether or not Arbitrage can occur.

The algorithm runs in O(VE) time (the time complexity of Bellman-Ford).

3. (Those in 858 only) Do part b of problem 24-3 and the second part of exercise 15.3-6(the c_k != 0 case).

15.3-6:

Proof by contradiction:

Consider a solution *S* with an optimal dollar amount *D*. Let d_{ij} be the product of every exchange rate between vertexes *i* and *j*. For each pair [*i,j*], let d_{ij} represent the maximum, optimal value for that pair. Let *i* and *j* be two vertices in *S*. Assume *x* and *y* are two other vertices in *S* that can be chosen alternatively to *i* and *j*, and that $d_{ij} > d_{xy}$. Let Let c_k be an arbitrary constant such that $d_{xy} * c_k > d_{ij}$. If we replaced vertices *i* and *j* with *x* and *y*, *D* would increase. Therefore, our initial assumption was false, and introducing arbitrary constants violates the optimal substructure property.

24-3 part b:

After running the Bellman-Ford algorithm, go through each vertex and relax each edge again. If there were no cycles, the value of each vertex would be converged and would not change.

However, since there is a negative cycle, the value of a vertex *v* will change. Simply follow the parent pointers from *v* until *v* is reached again (the start of the next cycle).

The time complexity for printing out the cycle is O(V).

4. Exercise 25.2-4 from CLRS.

In the Floyd-Warshall algorithm, d_{ij}^{k} relies *only* on the information in the d_{ij}^{k-1} 2D array. Therefore, we can reduce a dimension of space by overwriting the previous 2D array's element when computing dij^{k}. Additionally, the information in d_{ij}^{k-1} is not used by any other iteration other than d_{ij}^{k}.

5. What suggestions do you have for improving this assignment in the future?

Put more weight on the written questions, they were tricky!