Algorithms Homework 9

Liam Dillingham

December 3, 2018

The algorithm I utilized for this assignment is a modified Depth-first search. I believe this algorithm is the best because it can halt its progress as soon as an inefficient cycle is found. So it has the potential to have a very fast run time if it finds an inefficient cycle on the first try. alternatively, since depth-first search runs in time $\mathcal{O}(V + E)$, it has worse-case runtime of $\mathcal{O}(n^2)$ if the graph is dense. However, my modified algorithm may have to run across all edges for each vertex, giving it a runtime of $\mathcal{O}(VE) = \mathcal{O}(n^3)$.

When the algorithm runs, it does a depth first search until it runs into the source vertex. if it cannot, then there is no cycle, and that's okay, we can't determine an inefficient system if we can't return to our source. However, if it does, it then computes the potential product and tests if it is greater than 1. If it is, the algorithm quits immediately and we are done. If it is not, it backs up one step (returns from the source) and attempts to find some other path that can also reach the vertex. If no possible inefficient cycle is found, it gets a new source and tries again. It does this until one is found (at that point it exists immediately) otherwise the system is deemed efficient.

When an inefficient system is found, it walks this cycle and pushes each vertex in the stack twice. This double-pushing is simply so I can match the given output format for the assignment. Since any inefficient path is sufficient for an inefficient system, once the inefficient path is found, I do this stack-pushing. Then I stop running the algorithm and move to print out my path.
during the reading of the input file, I save a "data" matrix which saves all the conversion so that when I am printing out the sequence, I can get all the necessary information in constant time. However, popping from the stack will take $\mathcal{O}(n)$ time.

Thus, this entire algorithm/assignment takes $\mathcal{O}(n^3 + n) = \mathcal{O}(n^3)$ time.