Ryan Kuhl

COP4531

Programming assignment 2

Finding Strongest and Weakest Connected Components

This programming assignment investigated finding the strongly and weakly connected components in a given list of directed graph edges. The directed edges correlate with the strongly connected components, where the weakly connected components are found with the given edges in an undirected manner. The algorithm used to find the strongly connected components of the node list was Depth First Search called recursively. The DFS (Depth First Search) is given a set of unique vertex nodes and an adjacency list, and outputs every strongly connected component set. The Weakly connected components are found using a depth first search and path compression on the nodes.

The DFS in the strongly connected components contains a nested for loop, lending itself to a $\Omega(n^2)$ and the additional for loop in the SCC function, $O(n)$. Moving to the weakly connected components algorithm, the nested for loop would be a $\Omega(n^2)$, and the various other for loops, $O(n)$. Both would be $O(n^3)$.

The difficulties that I experienced with this program were plentiful. The first problem was the complexity and container optimization for use in the SCC and WCC functions. The use of sets kept the running time down given the large size of the stanford input file. The SCC was difficult to conceptualize, as was the WCC and I am as of yet unable to create a working implementation of edge count for either. Python provided some issues while I was programming, notably issues with return types and functions not having iterable types. File input proved to be quite simple, despite my worries.