Ryan Russell

Dr. Leyk

CSCE.221.511

27 April 2020

Programming Assignment Five Report

The program for this assignment reads in an input file with data and creates a graph from that data. The buildGraph function reads in a file line by line, creates a Vertex from the first number, puts that vertex into a vector called the node_list, and then puts the other numbers in that line in a list of lists called the adj_list. The displayGraph function prints out the graph by printing out a vertex from the node_list and the other vertices that that vertex connects to for each vertex. The compute_indegree function computes how many vertices point to a certain vertex for each vertex in the graph. The topological_sort function uses the indegrees of each vertex to sort them out and the print_top_sort function prints out this topological sort. The C++ features that I used were the STL list, vector, and queue data structures for holding either the vertices for the node_list,the list of lists of integers for the adj_list, or aiding in the process of finding the topological sort. Also, I used the STL string class in order to read in the strings of data from input files and convert them to integers, the std exception class to deal with the case of a cycle found in a graph, the iostream class in order to print out the data, and the fstream class in order to read in the data from files. The assumptions on the input data include that each line of data ends with a -1 in order for the program to know where to terminate the line. The program

also assumes that the beginning number of each line (the vertex) increments by 1 after every line. The algorithm uses a queue because it is necessary that you pop out the first element and the queue data structure uses FIFO (first in first out). This algorithm can use a stack, however, would need to be recursive and would require to have a helper function in order to acquire the same results. Also, the stack data structure can not use indegree to find the topological sort like the algorithm does with a queue. The algorithm is able to detect cycles because it shows that the while loop in the topological_sort function executes more times than the amount of vertices the graph has which means that the function is trying to sort one of the vertices even though the graph is already sorted. The algorithm in the topological_sort function will only be able to work if the graph is directed acyclic graph. For the compute_indegree function, the running time is O(|E| + |V|). This is because the first for each loop goes through each one of the vertices in node_list and assigns that node's indegree to zero, meaning that the for each loop will execute the number of vertices in the graph giving us O(|V|). The nested for loop goes through each adjacency list for each node and increments that node's indegree, meaning that the nest for loop will execute the number of edges in the graph giving us O(|E|). Combining these two parts of code together results in a running time of O(|E| + |V|) for the algorithm. For the topological_sort function, the running time is O(|E| + |V|). The for loop above the while loop (where every node that has an indegree of zero will be added into the queue) will execute at the most the number of vertices in the graph giving us O(|V|). The for loop inside of the while loop (where each edge towards a node is visited and that node's indegree is decremented) will execute at most the number of edges in the graph giving us O(|E|). Combining these two parts of code together results in a running time of O(|E| + |V|) for the algorithm.

Test Cases:

Case 1:

```
[rhrussell]@compute ~/CSCE_221/Assignment_5> (23:00:37 04/27/20)
:: ./main input.data
Graph:
1:2 4 5
2:3 4 7
3:4
4:6 7
5:
6:5
7:6

Topological Sort: 1 2 3 4 7 6 5
```

Case 2:

```
[rhrussell]@compute ~/CSCE_221/Assignment_5> (23:02:24 04/27/20)
:: ./main input2.data
Graph:
1:3
2:3 8
3:4 5 6 8
4:7
5:7
6:
7:
8:

Topological Sort: 1 2 3 4 5 6 8 7
```

Case 3:

```
[rhrussell]@compute ~/CSCE_221/Assignment_5> (23:03:14 04/27/20)
:: ./main input3.data
Graph:
1:2 4
2:5 7 8
3:6 8
4:5
5:6 9
6:
7:
8:
9:

Topological Sort: 1 3 2 4 7 8 5 6 9
```

Case 4:

```
[rhrussell]@compute ~/CSCE_221/Assignment_5> (23:03:53 04/27/20)
:: ./main input-cycle.data
Graph:
1:2 4 5
2:3 4 7
3:4
4:6 7
5:4
6:5
7:6

Topological Sort: Cycle Found in Graph
```