CSCE 221 Cover Page
PA5 Report
Due April 27 at midnight to eCampus

First Name: Cristian Last Name: Avalos UIN: 627003137

Username: avalos672918 E-mail address: avalos672918@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more: Aggie Honor System Office

Type of Sources			
People	Friends (discussion)		
Web pages (provide URL)	http://www.cplusplus.com/reference/list/list/		
Printed material			
Other Sources			

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.

"On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work."

Cristian Avalos April 27, 2020

Introduction: This programming assignment reads data from a csv file, creates a graph from that data, find the topological sorting, and then displays the results. The data structures utilized here are:

1. Graph
   A Graph consists of a finite set of vertices and set of edges which connect a pair of vertices. For this assignment, a graph is created by using two vector: "node_list" and "adj_list". node_list consist of type "Vertex" (from the struct Vertex) and adj_list consist of pointer lists of integers. The data is read from a csv file and is held in a class called "Graph".

2. Queue
   A queue is a linear structure which follows a particular order in which the operations are performed. A queue of type Vertex is utilized in the function topological_sort() in order to complete the top sort for the graph.

1. Why does the algorithm use a queue? Can we use a stack instead?
   This algorithm uses a queue because of its FIFO (first in first out) working principle. This works great with the way that the graph was built and because of how my top sort function is written. It is possible to use a stack to compute the topological sorting, however, for this assignment it will not produce the proper answer. This is because it has LIFO (last in first out) principal and because of how my top sort function is written.

2. Can you explain why the algorithm detects cycles?
   A cycle occurs when there is a back edge in the graph. The algorithm checks for cycles by keeping track of the amount of vertices that the algorithm has gone through . Every time a vertex is passed once the "indegree" is decremented because it is going through its adjacent list. If the count of iterations matches the total size of the tree, the no cycles are present. However, if there are cycles then it will iterate less times than the total number of vertexes, causing an exception to be thrown.

3. What is the running time for each function? Use the Big-O notation asymptotic notation and justify your answer.
   Each of the following functions are considered algorithms because they all perform a certain task.

   (a) buildGraph() - builds a graph using the data from the csv file
   This function reads in data from a csv file. It consist of two while loops. The first loop gets each line of the file and the second splits the lines by ",". The Big-O of this function is $O(V + E)$.

   (b) displayGraph() - displays the graph that was created
   This function prints the graph that I created using the inputted data. It consist of a while loop inside of a for loop. The for loop iterates V number of times where V is the number of vertices. The while loop runs E where E are the adjacent vertices. Combined, this function has a Big-O of $O(V + E)$.

(c) compute_indegree() - computes the indegree in every vertex
This function consists of a nested triple for loop that iterates through node_list.size(), node_list.size(), and adj_list() respectively. This has a Big-O of $O(V^3)$.

(d) topological_sort() - finds the topological sorting for the graph
This function consists of a while loop that runs V number of times, V being the number of vertices that the graph has. Inside of that while loop is a for loop that runs E number of times, E being the amount of adjacent vertices. Combined this function has a Big-O of $O(V + E)$.

(e) print_top_sort() - prints the results of the top sort
This function prints the items inside of a vector using a for loop. This function has a Big-O of $O(V)$.

7. Conclusion.

```
[avalos672918]@compute ~/spring2020/csce221/PA5> (14:20:06 04/27/20)
:: ./main input.data
1 : 2 4 5
2 : 3 4 7
3 : 4
4 : 6 7
5 :
6 : 5
7 : 6
Top Sort: 1 2 3 4 7 6 5

[avalos672918]@compute ~/spring2020/csce221/PA5> (14:20:12 04/27/20)
:: ./main input2.data
1 : 3
2 : 3 8
3 : 4 5 6 8
4 : 7
5 : 7
6 :
7 :
8 :
Top Sort: 1 2 3 4 5 6 8 7

[avalos672918]@compute ~/spring2020/csce221/PA5> (14:20:19 04/27/20)
:: ./main input3.data
1 : 2 4
2 : 5 7 8
3 : 6 8
4 : 5
5 : 6 9
6 :
7 :
8 :
9 :
Top Sort: 1 3 2 4 7 8 5 6 9

[avalos672918]@compute ~/spring2020/csce221/PA5> (14:20:26 04/27/20)
:: ./main input-cycle.data
1 : 2 4 5
2 : 3 4 7
3 : 4
4 : 6 7
5 : 4
6 : 5
7 : 6
terminate called after throwing an instance of 'GraphCycle'
  what():  There is a cycle in the graph. Top sort cannot work.
Aborted

[avalos672918]@compute ~/spring2020/csce221/PA5> (14:20:37 04/27/20)
::
```

Figure 1: Results from my code and input files