CSCE 221 Cover Page

Homework Assignment # PA-5

First Name: Justin Last Name: Lee UIN: 727000824

Username: 727000824 E-mail address: jlee232435@gmail.com

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 a get a lower number of points or even zero, read more in the Aggie Honor System O- e http://aggiehonor.tamu.edu/

Type of sources	Textbook	Lecture Slides	Website	
People				
Web pages (provide URL)		Dr. Leyk's Diagraph Slides	https://www.geeksforgeeks.org/topological-sorting-indegree-based-solution/	
Printed material	Data Structures and Algorithm Analysis in C++ by Mark A. Weiss			
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.*

Your Name: Justin Lee Date: 4/27/20

Description of your implementation, C++ features used, assumptions on input data.

The purpose of this assignment was to implement a graph data structure using Kahn's algorithm for topological sorting as shown in the textbook. More specifically, after the DAG was generated from reading in the input data given, the indegree was computed for every vertex. Then the vertices with indegree of zero were enqueued and then dequeued until the queue was empty. While it is dequeuing each time, the counter for visited vertices will be increased by one and the adjacent node indegrees will be decreased by one. If the indegree ever becomes zero for the adjacent node then it will be added to the queue. The graph itself was a Directed Acyclic Graph, which is a necessary requirement to even implementing topological sorting. The C++ features used were the STL queue, vector, and I/O streams to allow for storing, parsing, and accessing the input data. Some important assumptions on the input data is that the graph is sparse and unweighted, graph nodes are numbered consecutively starting from one and there are no gaps in the node numbering, and that -1 is used as a line terminator. Lastly, each row's first number indicates the label of the starting vertex of a directed edge, while the numbers following in that row are the end vertices accessed from the start vertex.

Why does the algorithm use a queue? Can we use a stack instead?

The algorithm uses a First In First Out (FIFO) queue mainly because it guarantees that each set operation takes O(1) time, in which all the vertices must be enqueued and dequeued until the queue is empty and the nodes have been visited. This same implementation can also be done with a stack list but stacks push and pop data following a Last In First Out (LIFO) pattern which would change the way it implements the algorithm.

Can you explain why the algorithm detects cycles?

Any vertex on a cycle will never be decremented to an indegree of 0 and never be added to the list. In addition, any vertices reachable from cycles won't be able to get an indegree of 0 either as well. Thus, the algorithm detects cycles as DAGs should not contain cycles, meaning that all paths have a finite length. A cycle would cause the orientation of the edge into the earliest vertex to be in the wrong direction. This would jeopardize the purpose of the program since a cycle would prevent a graph from being topologically sorted. This was accounted for in the program by checking if the counter for the visited vertices is ever not equal to the size of the node_list then it would throw an exception.

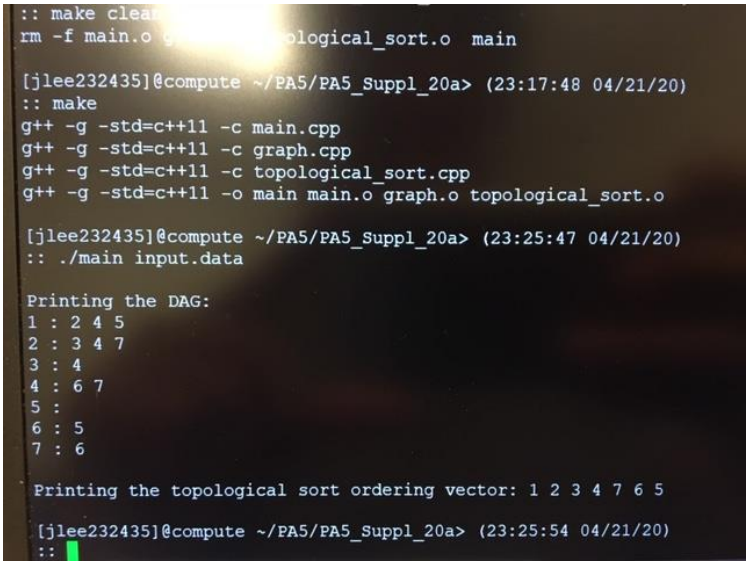
What is the running time for each function? Use Big-O notation asymptotic notation and justify your answer.

The running time for the topological_sort() function would have a run time of $O(|V| + |E|)$ since adjacency lists are used and the for loop operates at most once per edge. The running time for the compute_indegree() helper function would also have the same run time as topological_sort() since the outer for loop executes V number of times and the inner loop will go E number of times.

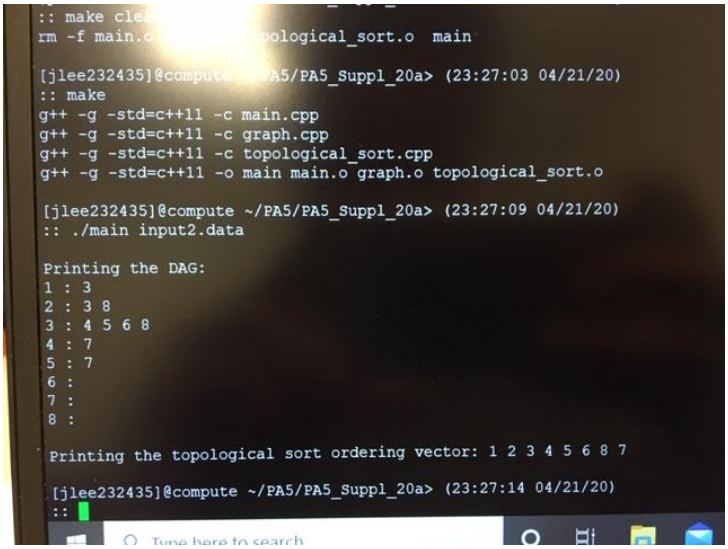
Testing program for correctness.

All test cases passed with the correct outputs.

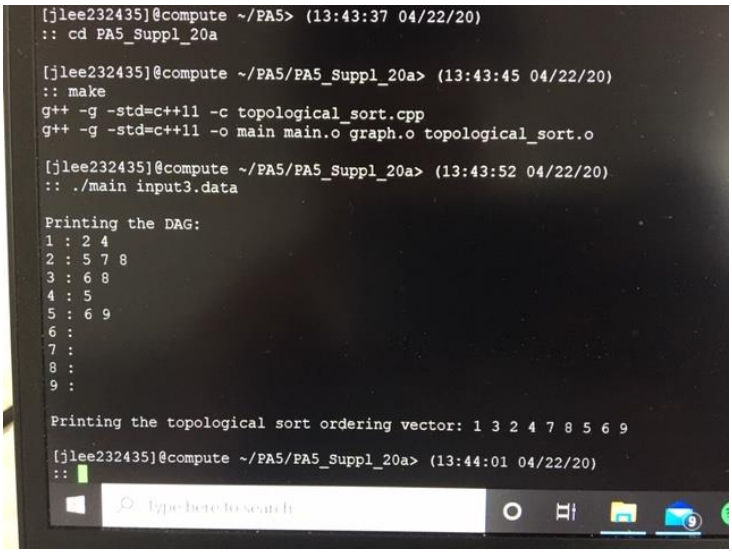
Case 1:



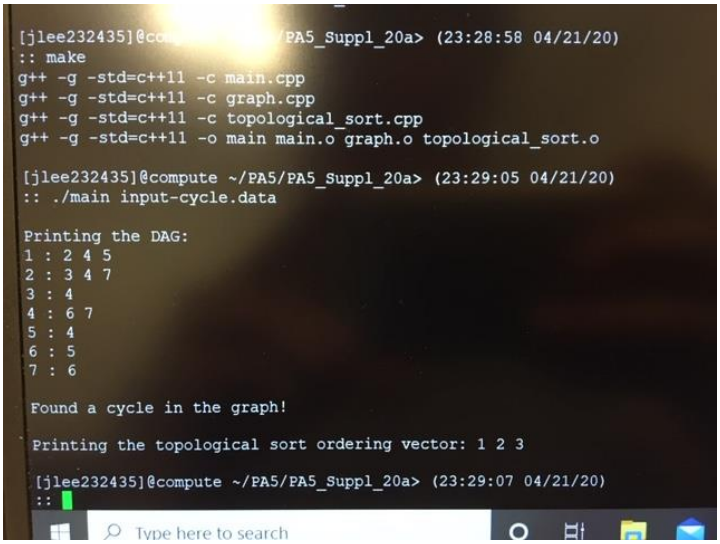
Case 2:



Case 3:


[jlee232435]@compute ~/PA5> (13:43:37 04/22/20)
:: cd PA5_Suppl_20a
[jlee232435]@compute ~/PA5/PA5_Suppl_20a> (13:43:45 04/22/20)
:: make
g++ -g -std=c++11 -c topological_sort.cpp
g++ -g -std=c++11 -o main main.o graph.o topological_sort.o
[jlee232435]@compute ~/PA5/PA5_Suppl_20a> (13:43:52 04/22/20)
:: ./main input3.data
Printing the DAG:
1 : 2 4
2 : 5 7 8
3 : 6 8
4 : 5
5 : 6 9
6 :
7 :
8 :
9 :
Printing the topological sort ordering vector: 1 3 2 4 7 8 5 6 9
[jlee232435]@compute ~/PA5/PA5_Suppl_20a> (13:44:01 04/22/20)
::

Case 4:


[jlee232435]@co /PA5_Suppl_20a> (23:28:58 04/21/20)
:: make
g++ -g -std=c++11 -c main.cpp
g++ -g -std=c++11 -c graph.cpp
g++ -g -std=c++11 -c topological_sort.cpp
g++ -g -std=c++11 -o main main.o graph.o topological_sort.o
[jlee232435]@compute ~/PA5/PA5_Suppl_20a> (23:29:05 04/21/20)
:: ./main input-cycle.data
Printing the DAG:
1 : 2 4 5
2 : 3 4 7
3 : 4
4 : 6 7
5 : 4
6 : 5
7 : 6
Found a cycle in the graph!
Printing the topological sort ordering vector: 1 2 3
[jlee232435]@compute ~/PA5/PA5_Suppl_20a> (23:29:07 04/21/20)
::
Type here to search