CSCE 221 Cover Page
Program Assignment # 4

First Name: Rong Last Name: Xu UIN: 928009312

User Name: Abby-xu E-mail address: rongx0915@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 in the Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources					
People					
Web pages (provide URL)					
Printed material					
Other Sources					

I certify that I have listed all the sources that I used to develop the solutions/code 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 (signature) Rong Xu Date 2020/11/12 The Pro-

1. The description of an assignment problem.
   (a) This lab includes three parts.
   (b) Part 1: Students will represent the derected graph based on a input data file. Bascially, students will read each line of the input file, then build a graph based on the file. (the input file will present each nodes in the beginning of each line and the rest nodes are the adjacent nodes). After storing the nodes and build the graph, the program will also present the graph just built.
   (c) Part 2: Student will use the topological sorting to sort the graph just built and output the topological ordering. If the graph cannot be present as the topological order and the program will print the message "Cycle Detected."
   (d) Part 3: Students will use four test cases to test the program.
2. The description of data structures and algorithms used to solve the problem.
   (a) Provide definitions of data structures for graph
      i. For the private member, I used the unordered_map named node_set and vector named out_node. For the unordered _map I used the c++ standard library.
   (b) Provide definitions of data structures for topological sorting.
      i. For the topological sorting, I implement it based on the graph I just built. I also used the queue and vector to store the data I need.
   (c) Describe and analyze the algorithms used to solve the problem.
      i. graph
         A. build-graph(O(n*m)): for this function, we are going read the data from the input file and then store the data in our unordered_map(node_sets). For each line, we will first read the label and then build a vertex object based on that label. Then we keep reading the data and store them into a vector, at last we can make a pair of the label and the vector of adjcent nodes then insert the vertex into our unordered_map. The running time of this function is O(n*m) where the n is the number of vertex and the m is the number of the nodes in the adjcent vector.
         B. display-graph (O(n+m)): for this function, since I make a vector called input_order when I insert the pair. So for this function, I make a for loop to go over each pair in the unordered_map and then print them out. On the other hand, although I stored the label into the vector instead of the vertex, when I use the helper function ".at()". By using this function I will get the vertex in the unorder_map for running time of O(1), and if the program cannot find the vertex, it will throw an exception. Besides, for outputing each vertex, I override the "<<" sign for output the nodes in the adjcent list which will cost m times for each vertex.
         C. .at() (O(1)): for this function, since I used the unordered_map which is implemented by hasing table, tbe total running time will just O(1). However, if the hash code function is not good then, the time complexity will be O(n). [I think in my function, the code function is good so the time complexity will just O(1)...]
         D. size() (O(n)): for this function I used the code provided by the instruction. I think the time complexity is O(n) since we use the for loop to go over every vertex in the unordered_map node_sets.
      ii. topological sorting
         A. compute_indegree() (O(n+m)): for this function, we will goover each vertex and the adjcent list for each vertex. In thi scase, the n is the number of vertex in the unordered_map and the m is the number of nodes in each adjacent list. Remark: the m for each node is different, and I change the code in the struct of vertex from "int indegree;" to "int indegree = 0;". The outer for loop will be executed n number of times and the inner for loop will be executed m number of times, Thus overall time complexity is O(n+m).

B. topological_sort() (O(n+m)): for this function, as before, we will assume the graph is already read into an adjacency list and the indegrees are computed and stored by compute_indegree() funcion. The time to perform this algorithm is O(n+m). This apparent when one realized that the body of the for loop is executed at most once per edge.

iii. Explain why the algorithm detects cycles

A. For this function, we will use the queue data structure to store the vertex with indegree of 0. So at first we check every vertex and push the vertex with indegree of 0 to the queue. Then for every vertex in the queue, we first make a count of them (using the top_num to take the order, I didn't use the top_num for printing in the end, insted, I use the vector out_node as a private member of graph class to store the vertex I'm going to output.)

B. Then we go over every vertexs in the adjacent list of that vertex I mentioned above, for example vertex v, for those vertexs we will decrease their indegree by 1 since we are gonna delete the vertex v, and then if the indegree is 0 after we decrease 1, then we will push the vertex to the queue and keep looping.

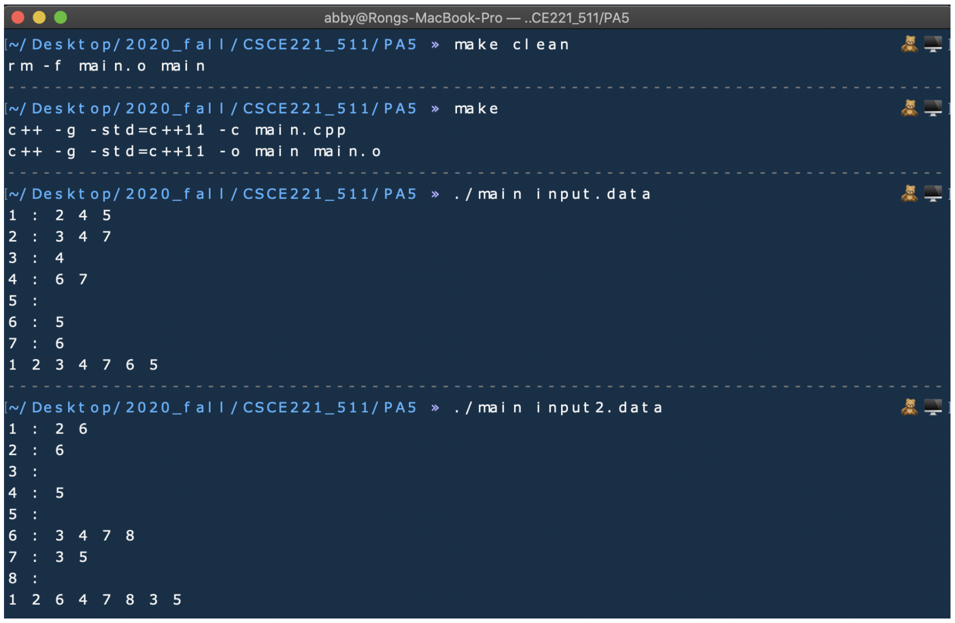
C. When the queue is empty, we will suppose that every vertex should be counted and stored in the vector. In this case, we will compare the counter(the number we counted in the wihle loop) and the size of our unordered_map.(the number of vertexs). If they are not equal to each other, then means there is a cycle.

iv. Why does the topological sort algorithm use a queue? Can we use a stack instead

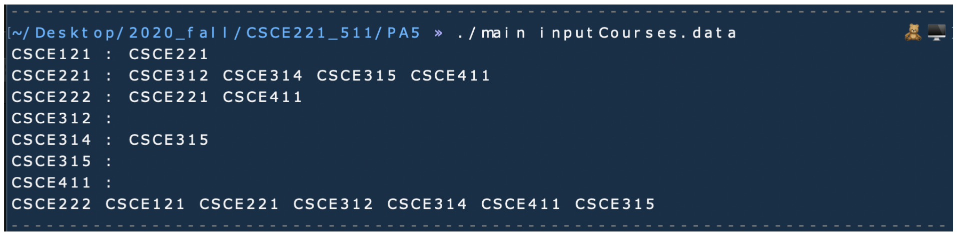
A. Because we are using a topological sort by performing a DFS-based implementation on the graph.

B. There is another way for topological sorting by using stack. However, the result would be different since a topological sort processes vertices is based on BFS.

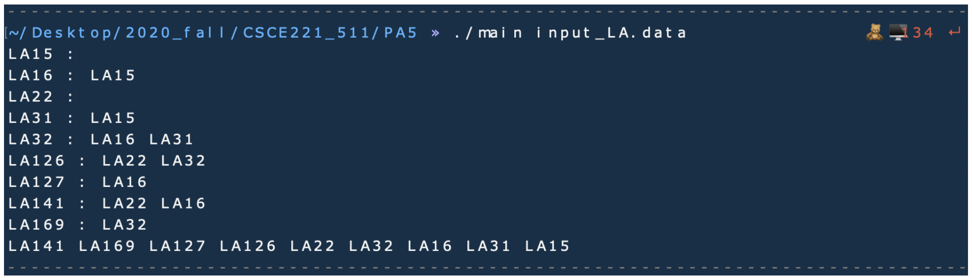
C. Both of these two ways avoid the sequential scan through all the vertices in the search for a vertex of indegree 0.

3. C++ organization and implementation of the problem solution.
   (a) See the mimir submission.
4. test your program for correctness using the four cases below:
   (a) Case 1: Use the example (input.data)
      i. 

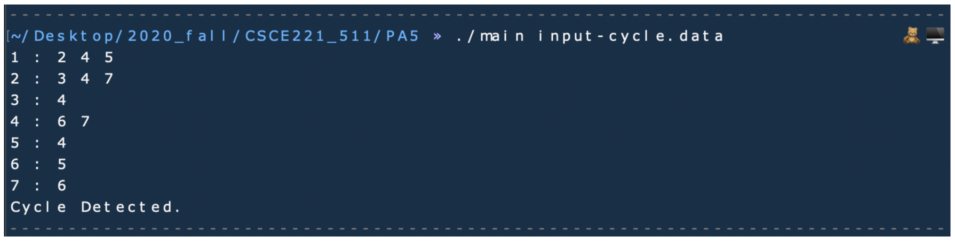
```
abby@Rongs-MacBook-Pro — ..CE221_511/PA5
~/Desktop/2020_fall/CSCE221_511/PA5 » make clean
rm -f main.o main
~/Desktop/2020_fall/CSCE221_511/PA5 » make
c++ -g -std=c++11 -c main.cpp
c++ -g -std=c++11 -o main main.o
~/Desktop/2020_fall/CSCE221_511/PA5 » ./main input.data
1 : 2 4 5
2 : 3 4 7
3 : 4
4 : 6 7
5 :
6 : 5
7 : 6
1 2 3 4 7 6 5
~/Desktop/2020_fall/CSCE221_511/PA5 » ./main input2.data
1 : 2 6
2 : 6
3 :
4 : 5
5 :
6 : 3 4 7 8
7 : 3 5
8 :
1 2 6 4 7 8 3 5
```

   (b) Case 2: Samantha's course schedule
      i. 

```
~/Desktop/2020_fall/CSCE221_511/PA5 » ./main inputCourses.data
CSCE121 : CSCE221
CSCE221 : CSCE312 CSCE314 CSCE315 CSCE411
CSCE222 : CSCE221 CSCE411
CSCE312 :
CSCE314 : CSCE315
CSCE315 :
CSCE411 :
CSCE222 CSCE121 CSCE221 CSCE312 CSCE314 CSCE411 CSCE315
```

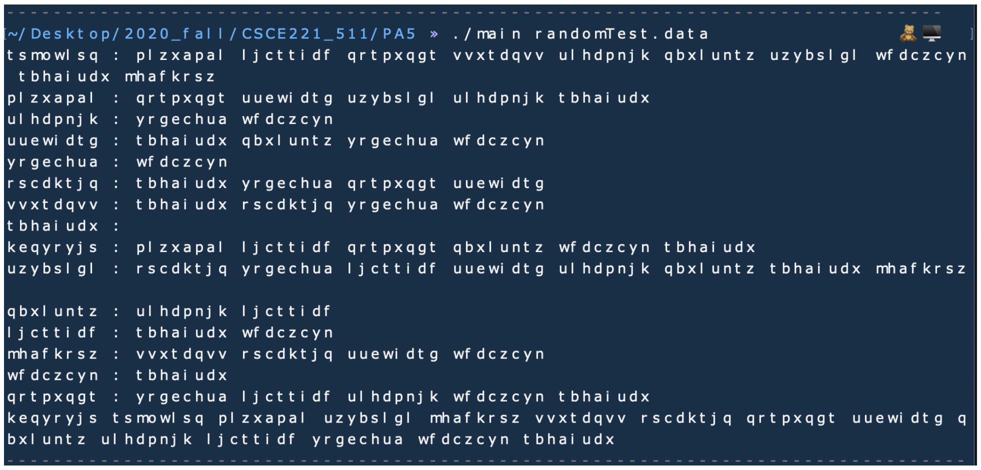
   (c) Case 3: Samantha's foreign languages
      i. 

```
~/Desktop/2020_fall/CSCE221_511/PA5 » ./main input_LA.data                     134 ↵
LA15 :
LA16 : LA15
LA22 :
LA31 : LA15
LA32 : LA16 LA31
LA126 : LA22 LA32
LA127 : LA16
LA141 : LA22 LA16
LA169 : LA32
LA141 LA169 LA127 LA126 LA22 LA32 LA16 LA31 LA15
```

   (d) Case 4: input-cycle.data
      i. 

```
~/Desktop/2020_fall/CSCE221_511/PA5 » ./main input-cycle.data
1 : 2 4 5
2 : 3 4 7
3 : 4
4 : 6 7
5 : 4
6 : 5
7 : 6
Cycle Detected.
```

(e) Case 5: random data

i. 

```
~/Desktop/2020_fall/CSCE221_511/PA5 » ./main randomTest.data
tsmowlsq : plzxapal ljcttidf qrtpxqgt vvxtdqvv ulhdpnjk qbxluntz uzybslgl wfdczcyn
 tbhaiudx mhafkrsz
plzxapal : qrtpxqgt uuewidtg uzybslgl ulhdpnjk tbhaiudx
ulhdpnjk : yrgechua wfdczcyn
uuewidtg : tbhaiudx qbxluntz yrgechua wfdczcyn
yrgechua : wfdczcyn
rscdktjq : tbhaiudx yrgechua qrtpxqgt uuewidtg
vvxtdqvv : tbhaiudx rscdktjq yrgechua wfdczcyn
tbhaiudx :
keqyryjs : plzxapal ljcttidf qrtpxqgt qbxluntz wfdczcyn tbhaiudx
uzybslgl : rscdktjq yrgechua ljcttidf uuewidtg ulhdpnjk qbxluntz tbhaiudx mhafkrsz

qbxluntz : ulhdpnjk ljcttidf
ljcttidf : tbhaiudx wfdczcyn
mhafkrsz : vvxtdqvv rscdktjq uuewidtg wfdczcyn
wfdczcyn : tbhaiudx
qrtpxqgt : yrgechua ljcttidf ulhdpnjk wfdczcyn tbhaiudx
keqyryjs tsmowlsq plzxapal uzybslgl mhafkrsz vvxtdqvv rscdktjq qrtpxqgt uuewidtg q
bxluntz ulhdpnjk ljcttidf yrgechua wfdczcyn tbhaiudx
```

5. A user guide description how to navigate your program with the instructions how to:

(a) compile the program: makefile file

```
#OBJ = main.o topological_sort.o # Part 2
OBJ = main.o
MAIN = main
$(MAIN): $(OBJ)
        $(CXX) -g -std=c++11 -o $(MAIN) $(OBJ)
main.o: main.cpp graph.h
        $(CXX) -g -std=c++11 -c main.cpp
topological_sort.o: topological_sort.cpp graph.h
        $(CXX) -g -std=c++11 -c topological_sort.cpp
clean:
        rm -f $(OBJ) $(MAIN)
```

(b) compile the program: specify the directory and file names, etc.

```
» cd ./PA_5
» make clean
» make
» ./main input.data
```

6. Specifications and description of input and output formats and files
   (a) The type of files: keyboard, text files, etc (if applicable).
      i. Input file: a datafile of adjacent vertex list
   (b) A file input format: when a program requires a sequence of input items, specify the number of items per line or a line termination. Provide a sample of a required input format.
      i. Input file: input file should be an .data file which have the format like this:
         A. Test file for "input.data"
         B.
```
1 2 4 5
2 3 4 7
3 4
4 6 7
5
6 5
7 6
```
   (c) Discuss possible cases when your program could crash because of incorrect input (a wrong file name, strings instead of a number, or such cases when the program expects 10 items to read and it finds only 9.)
      i. none.
      ii. no special case.

7. Provide types of exceptions and their purpose in your program.
   (a) logical exceptions (such as deletion of an item from an empty container, etc.).
      i. No logical error has been found in testing
   (b) runtime exception (such as division by 0, etc.)
      i. When the program cannot find the vertex, it will throw an exception.

8. Test your program for correctness using valid, invalid, and random inputs (e.g., insertion of an item at the beginning, at the end, or at a random place into a sorted vector). Include evidence of your testing, such as an output file or screen shots with an input and the corresponding output
   (a) no implementation of invalid input.
   (b) All tests passes.