CCDSALG Term 3, AY 2024 – 2025

Project 2 Documentation – Social Network (Undirected Graph Application)

DECLARATION OF INTELLECTUAL HONESTY / ORIGINAL WORK

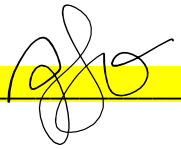
I declare that the project that I'm submitting (as part of a group) is the product of my own intellectual efforts. No part of the project was copied from any source, and no part was shared with another person.

NAME#1: Yuan Miguel A. Panlilio SIGNATURE#1:  SECTION: S13

The following are my specific contributions to MCO2:

1. BFS TRAVERSAL
2. DFS TRAVERSAL
3. Queue Data Structure
4. Documentation & Testing

Based on my honest estimate, and in consultation with my groupmates, I contributed approximately **28%** to the entire project.

NAME#2: Nigel Henry S. So SIGNATURE#2:  SECTION: S13

The following are my specific contributions to MCO2:

1. Graph Data Structure
2. Reading The Input File And Storing Its Information
3. Producing Output Files 1-4 & 7
4. Main
5. Bonus
6. Documentation and Testing

Based on my honest estimate, and in consultation with my groupmates, I contributed approximately **44%** to the entire project.

NAME#3: Princess Ayesa A. Tullao SIGNATURE#3:  SECTION:S12

The following are my specific contributions to MCO2:

1. DFS TRAVERSAL & its Sorting Function
2. BFS TRAVERSAL
3. Documentation & Testing

Based on my honest estimate, and in consultation with my groupmates, I contributed approximately **28%** to the entire project

1. Indicate how to compile (if it is a compiled language) your codes, and how RUN (execute) your program from the COMMAND LINE. Examples are shown below highlighted in yellow. Replace them accordingly. Make sure that all your group members test what you typed below because I will follow/copy-paste them verbatim. I will initially test your solution using some of the sample input text file that you submitted. Thereafter, I will run it again using my own test data:

- How to compile from the command line

  CCDSALG> **gcc -std=c99 -Wall graph.c files.c queue.c traversal.c main.c -o main.exe**

Next, answer the following questions:

a. Is there a compilation (syntax error) in your codes? (YES or NO). No
   WARNING: the project will automatically be graded with a score of **0** if there is syntax error in any of the submitted source code files. Please make sure that your submission does not have a syntax error.

b. Is there any compilation warning in your codes? (YES or NO) No
   WARNING: there will be a 1 point deduction for every unique compiler warning. Please make sure that your submission does not have a compiler warning.

- How to run from the command line

  CCDSALG>**main**

Did you do the BONUS part (YES or NO) Yes.

NOTE: Please do NOT submit a solution if it is not working correctly based on the exhaustive testing that you have done.

If you did the BONUS part, indicate below:

- How to compile from the command line

  CCDSALG> **gcc -std=c99 -Wall graph.c files.c queue.c traversal.c 02-BONUS.c -o BONUS.exe**

- How to run from the command line

  CCDSALG>**BONUS**

2. How did you implement your Graph data structure? Did you implement it using adjacency matrix? adjacency list? Why? Explain briefly (at most 5 sentences).

We implemented our Graph data structure using a 2x2 array-based adjacency matrix. An adjacency matrix would allow for a more simpler and familiar approach for us, having already pictured how indexing worked in CCPROG2. Moreover, an adjacency matrix would allow us to use it as some sort of boolean mechanism as well, which came in handy when we had to do the traversals.

3. How did you implement the DFS and BFS traversals? What other data structures (aside from the graph representation) did you use? Explain your algorithm succinctly.

a. DFS Traversal:

For the DFS Traversal, we used the iterative method instead of recursion to have more control over the process in case issues like *stack overflow* happens. We also made use of array-based stack to manually manage the traversal instead of relying on the system's call stack in recursion. Through this, we were able to track visited nodes efficiently and push adjacent vertices in the correct order. Lastly, we maximized the use of temporary arrays in collecting adjacent vertices, making sure they are sorted before pushing them onto the stack reversely, preserving the correct DFS tree structure.

b. BFS Traversal:

For the BFS traversal, we utilized an iterative approach to provide more simplicity to our benefit. Along with this comes with our implementation of an array-based queue data structure, in which we used to store unchecked vertices for the BFS traversal. Before the iteration itself starts, we store the vertices in a separate alphabetically sorted array, which uses the bubble sorting algorithm, to make sure that the output follows the "lowest value gets checked" rule when there are vertices of the same level. As the iteration progresses, the latest enqueued vertex' index in the adjacency matrix is identified. An inner loop then iterates through all of the vertices and enqueues them in the queue data structure as long as it is connected to the vertex in the outer loop and as long as the vertex has not been checked yet. This process iterates through until the queue is not empty yet, meaning that this process won't stop until all of the elements in the queue have been successfully dequeued into a separate array for printing purposes.

4. Disclose **IN DETAIL** what is/are NOT working correctly in your solution. Please be honest about this. NON-DISCLOSURE will result in severe point deduction. Explain briefly the reason why your group could not make it work.

For example:
The following are NOT working (buggy):
a.
b.

We were not able to make them work because:

a.
b

5. What do you think is the level of difficulty of the project (was it easy, medium or hard)? Which part is hard (if you answer was "hard") and why?

Yuan Miguel Panlilio: I cannot answer for the whole project as there was a specific task allotted for me. However, in my coding of the BFS traversal, I found that devising the code logic for the traversal was the most challenging part, as the use of queues was still a first for me. Comparatively, the first MCO was definitely more challenging, so I would say that the level of difficulty for the project would be at a medium.

Nigel Henry S. So: For me, I think the difficulty of the project is medium. The part I found most challenging was the reading input file where the contents of the file had to be read and stored in a specific way in order to match the desired outputs.

Princess Ayesa Tullao : Since I can only speak for the parts I worked on, I would rate it as medium difficulty. The most challenging part is trying to understand how my groupmates structured their portions of the code, especially that our modules are interconnected. This required extra time to coordinate and make sure my DFS traversal logic is integrated properly with the rest of the program. Another part that I consider as challenging is making sure the traversal output followed the correct alphabetical order of adjacent vertices because this added an extra layer of logic.

6. Fill out the table below.

REQUIREMENT	**AVE. OF SELF-ASSESSMENT**
1. Correctly produced Output Text File #1 (Set of Vertices & Edges)	10 (max. 10 points)
2. Correctly produced Output Text File #2 (Vertices With Degrees)	10 (max. 10 points)
3. Correctly produced Output Text File #3 (Adj. Matrix Visualization)	10 (max. 10 points)
4. Correctly produced Output Text File #4 (Adj. List Visualization)	10 (max. 10 points)
5. Correctly produced Output Text File #5 (BFS Traversal)	22.5 (max. 22.5 points)
6. Correctly produced Output Text File #6 (DFS Traversal)	22.5 (max. 22.5 points)
7. Documentation	10 (max. 10 points)

8. Compliance: deduct 1 point for every instruction not complied with (examples: incorrect output file names, extraneous contents in output files, etc.)	5 (max. 5 points)
9. Bonus :-)	10 (max. 10 points
TOTAL SCORE	110 over 100.

NOTE: The evaluation that the instructor will give is not necessarily going to be the same as what you indicated above. The self-assessment serves primarily as a guide on how your project will be evaluated.

***** THE END *****