




CCDSALG Term 3, AY 2023 – 2024
Project 2 Documentation – Social Network (Undirected Graph Application)

DECLARATION OF INTELLECTUAL HONESTY / ORIGINAL WORK

We declare that the project that we are submitting is the product of our own work. No part of our work was copied from any source, and that no part was shared with another person outside of our group. We also declare that each member contributed to the project as indicated in the table below.

Section	Names and Signatures	Task 1	Task 2	Task 3	Task 4	Task 5
S17	Quijano, Jan Philip Alexandre 	X	X	X		
S17	Marcelo, Chrystel Anne 		X	X	X	
S17	Evangelista, Aiella Reigne 	X			X	X

Fill-up the table. For the tasks, put an X mark if you have performed the specified task. Please refer to the project specifications for the description of each task. Don't forget to affix your signature.

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 them verbatim. I will initially test your solution using 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 (for compiled language only):

```
C:\CCDSALG> gcc -Wall GROUP-49.c -o GROUP-49.exe
```

- How to run from the command line

```
C:\CCDSALG>GROUP-49
```

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 the Graph data structure using an adjacency matrix for simplicity and efficiency. Using an adjacency matrix allowed us to easily check vertex connections with direct indexing. For BFS and DFS, we used arrays to keep track of vertex visitation status and to hold the traversal results. While it uses more space compared to an adjacency list, its straightforward implementation and clear visualization of vertex connections provides better support for fast accessing and updating of data.

3. How did you implement the DFS and BFS traversals? Explain your algorithm succinctly.

a. DFS Traversal:

In implementing Depth-First Search, we begin by initializing an array to keep track of visited vertices. The algorithm starts by calling the recursive DFS function on the starting vertex, marking it as visited. The algorithm then collects all the adjacent vertices and arranges them according to vertex ID. The lowest vertex ID is then called in the recursive call. During each recursive call, the algorithm visits the current vertex and then recursively explores all its unvisited adjacent vertices. This process continues until all reachable vertices from the starting point have been visited, ensuring a complete exploration of the graph.

b. BFS Traversal:

To begin a breadth-first search, we initialize an array that keeps track of the visited vertices and an array that acts as a queue. We then enqueue the first vertex and mark it as having been visited. The algorithm then collects all the adjacent vertices and arranges them according to vertex ID. Then, we dequeue one vertex at a time, with the lower vertex ID marked as visited, and then enqueue all the unvisited neighbors that are adjacent to it. This process continues until the queue becomes empty; thus enabling all neighbors in the current distance or level to be visited before proceeding to the next level.

4. Disclose what is NOT working correctly in your solution. Be honest about this. Explain briefly the reason why your group was not able to make it work.

N/A

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 answered hard)? Type your answer individually for this question.

Quijano, Jan Philip Alexandre:

The project's difficulty was easy to medium at worst. The reason for this was the BFS and DFS algorithms followed almost the same process, making it easy to implement both processes. The difficulties we encountered however was in determining and implementing the best approach to storing the data. We had to choose from either an adjacency list or an adjacency matrix and we chose the latter due to its rather understandable complexity as compared to adjacency lists.

Marcelo, Chrystel Anne:

The difficulty of the project, I'd say, was medium. At first, we weren't sure what to use for the adjacency matrix, we at first used 3D arrays, but later found out that 2D arrays were easier to implement. Additionally, the complex part of the BFS algorithm implementation was managing the queue and making sure that all visited nodes were in the correct order. Also, the vertex input had to be case-insensitive so the process got a bit more complex as we had to make sure that the comparisons for each of the char arrays were right and consistent, though it was doable.

Evangelista, Aiella Reigne:

I would say the project's difficulty was relatively medium once we got started implementing the algorithms. We first encountered problems on whether to use adjacency list or adjacency matrix but ultimately decided to use an adjacency matrix for simpler data handling. The project got easier from there, as the implementation for both the BFS and DFS algorithms mostly followed the same process, and therefore only required a few changes from one to the other in order to make the program's algorithm work as expected.

6. Are you submitting solutions for:

- Bonus opportunity #1 (Yes/No): No
- Bonus opportunity #2 (Yes/No): No

Request: If you know in advance that your solution to the bonus portion is not working correctly, then please DO NOT submit it so as not to waste precious time for both of us.

Copy/paste below two screenshots of the drawings produced by your program. Crop out the unnecessary portions. Make sure that the images are large “enough” to see the details. I will test your project to verify that the images produced by your program are the same with the copy/pasted screenshots.

Screenshot of the original graph (corresponding to the data encoded in the sample input text file you submitted):

N/A

Screenshot of the graph due to the BFS traversal (corresponding to the data encoded in the sample input text file you submitted):

N/A

9. Fill-up the table below. Refer to the rubric in the project specs. It is suggested that you do first an individual self-assessment. Thereafter, compute the average evaluation for your group, and encode it below.

REQUIREMENT	AVE. OF SELF-ASSESSMENT
1. Correctly produced the list of vertices and their corresponding degrees	15 (max. 15 points)
2. Correctly produced the BFS traversal	35 (max. 35 points)
3. Correctly produced the DFS traversal	35 (max. 35 points)
4. Documentation	10 (max. 10 points)
5. Compliance (deduct 1 point for every instruction not complied with)	5 (max. 5 points)
Bonus opportunity #1: Drawing of the given graph	0 (0 or 10 points)
Bonus opportunity #2: Drawing of the BFS tree	0 (0 or 10 points)

TOTAL SCORE 100 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.