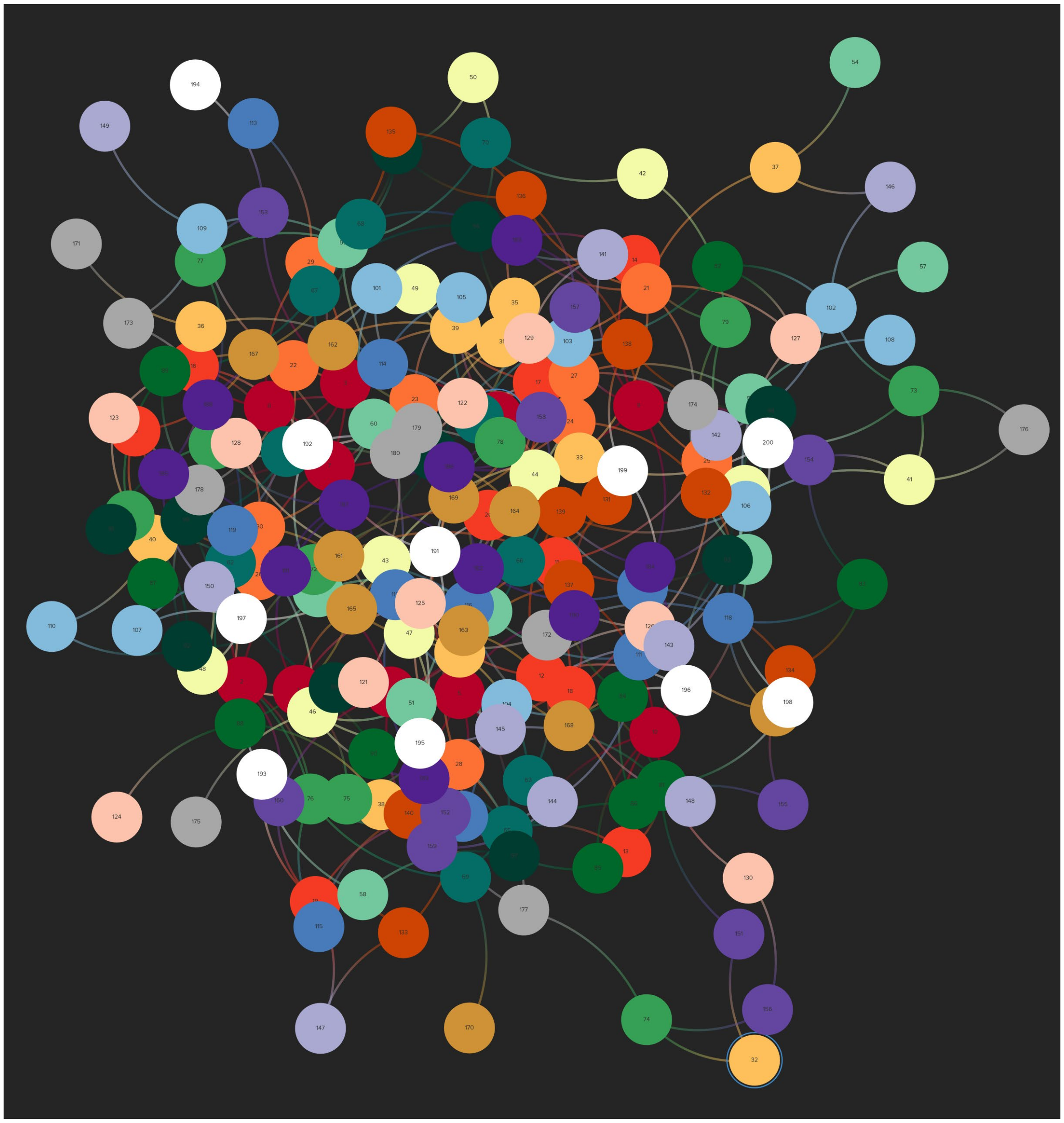
Homework Assignment #1: Breadth-First and Depth-First Pathfinding
CSC 462
Fall 2023



The image above is a visualization of the graph described in the file BFS_DFS.csv, a comma separated file that gives the node name in the first position, and a varying number of direct neighbors in the following positions.

In this assignment you will implement both Breadth-First and Depth-First path finding algorithms **<u>from scratch</u>**. This means that you should not rely on any packages for the implementation for either algorithm. You may use packages for a stack and a queue, but you should write your own node/edge implementation and you must code the mechanics of the algorithm by hand.

Be cautious when using a coding support tool such as CoPilot. Your code should be well tested before you tackle the problem with the full sample file.

Input
Your program should be able to read in "BFS_DFS.csv" (or any similarly formatted .csv file)

Your program should prompt the user for the name of the file, which you can assume lives in the local directory. Your program should also prompt the user to enter the starting node ID and the ending node ID. If the user enters invalid data (a bad file name or a node ID out of range), the program should exit gracefully or prompt the user again.

```
Example:
Please enter the file name and extension: BFS_DFS.csv
Loading file…
Start node (1 – 200): 123
End Node (1 – 200): 43
```

Output
For each algorithm, your program should print out the **first path** found by the algorithm or indicate that no path exists between the two nodes. Print your results to the console in the following format:

Sample Console Output [example only, not a verified path]

```
Breadth-first traversal
123 – 12 – 1 – 5 – 43

Depth-first Search
123 – 3 – 72 – 6 – 12 – 1 – 5 – 43
```

Implementation Requirements
You must implement the core search algorithms yourself. You may use Java or Python. Be sure to include a readme file so I can figure out how to run your code.

To avoid infinite runtimes, do not allow cycles in your search paths.
Remember that Homework Assignments are individual assignments.
You may
- talk through approaches to the problem with classmates
- ask someone to take a quick look at your *broken code* to help you debug
- use the internet and classmates to review the different search algorithms

You may NOT
- work together on coding your project
- show your *working code* to someone else as a way of helping them with their problem
- use someone else's code for credit (either a classmate's or an internet source). If you do use found code for part of your solution, **you must cite your source.** That portion of the code will not be graded. Failure to cite any outside sources will be considered a violation of the Academic Dishonesty policies.

How to Submit

You will submit your work on D2L. Make sure your full name is in the comments at the beginning of every source file.

Rubric:
1) Code compiles. /5
2) Code prompts for file name & search keys. /5
3) BFS runs correctly on instructor input. /15
4) DFS runs correctly on instructor input. /15
5) Algorithms check for cycles. /4
4) Code is well formatted and well documented. /10

Total: /54