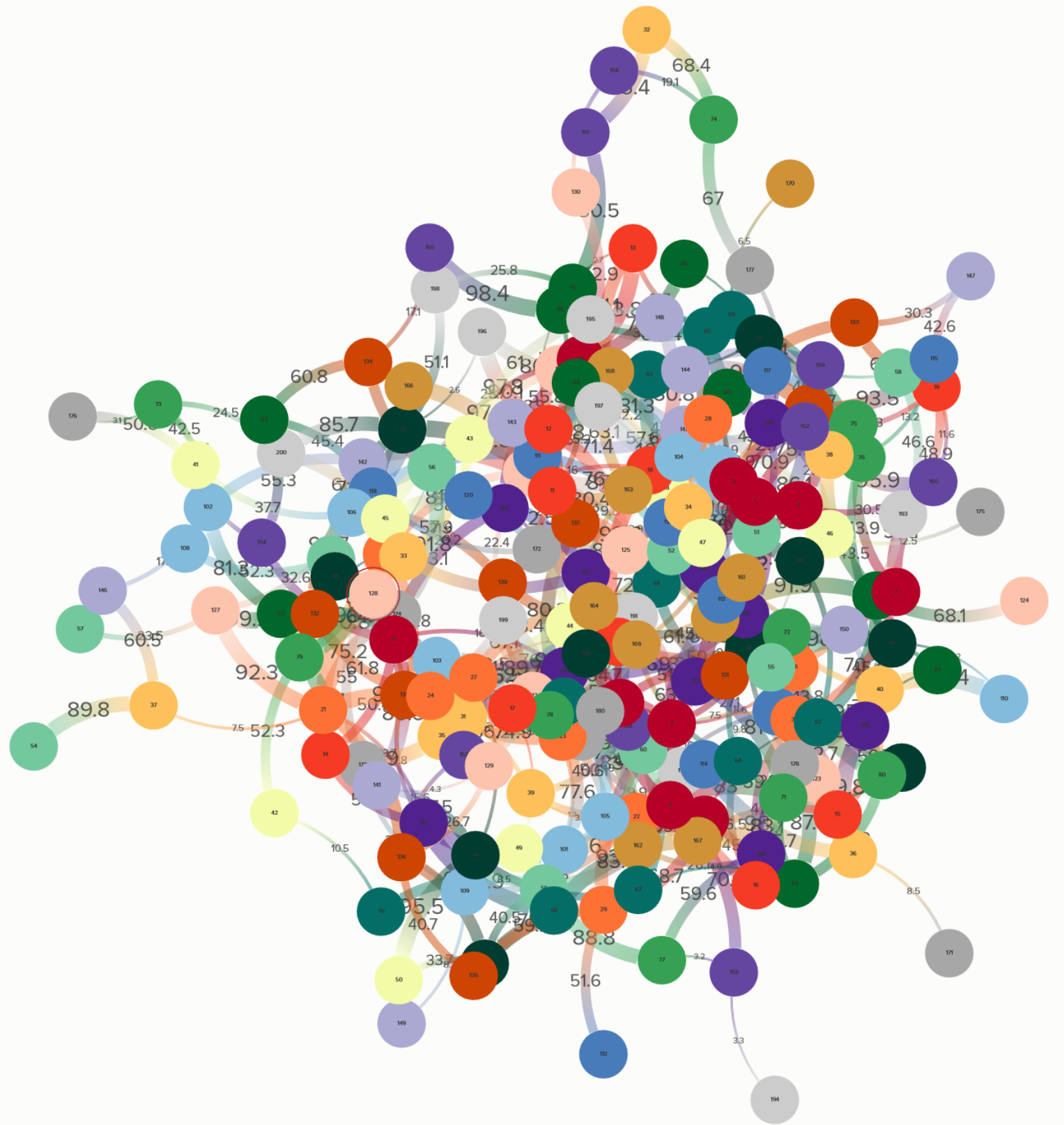
**Homework Assignment #2: A\* Pathfinding**

**CSC 462**

**Fall 2023**



The image above is a visualization of the graph described in the file EdgeWeights.csv, a comma separated file that gives the departing node ID in the first column, the arriving node in the second column, and the edge weight in the third column. In the above image, the thicker edges represent edges with larger weights.

In this assignment you will implement the A\* minimum cost path finding (and optionally, IDA\*) algorithms **from scratch**. This means that you should not rely on any packages for the implementation for either algorithm. You may use packages for a priority queue or sorted list, but you should write your own node/edge implementation and you must code the mechanics of the algorithm by hand.

**Input**

Your program should be able to read in “EdgeWeights.csv” (or any similarly formatted .csv file) which contains the list of all edges and their weights. You should be able to infer the structure of the graph from this file. Your program should also read in a file of heuristic values, “minCosts.csv”.

I have also generated a file of heuristics for all nodes to all other nodes. The heuristic for this problem is the smallest edge cost found in the least expensive path from *i* to *j*. Notice that the heuristic for node1 to node2 is included, but not node2 to node1. That is because they will have the same value.

You 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 edge weight file name and extension: *EdgeWeights.csv*

Loading file…

Please enter the heuristic file name and extension:

minCosts.csv

Start node (1 – 200): *123*

End Node (1 – 200): *43*

**Output**

For each algorithm, your program should print out the **first, least expensive 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:

[<minimum path cost>] <path>

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

A\* minimum cost path

[167.4] 123 – 12 – 1 – 5 – 43

**Implementation Requirements**

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 extended runtimes, do not allow cycles in your search paths.

**Bonus Challenge**

In addition to A\*, you may also implement IDA\*. The output for IDA\* should show when the limit cost increases to demonstrate the deepening process.

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 (either a classmate’s or an internet source) 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) Edge file reads correctly. /5

4) Heuristic file reads correctly. /5

5) A\* runs correctly on instructor input. /15

6) Algorithms check for cycles. /2

7) Code is well formatted and well documented. /10

8) IDA\* runs correctly on instructor input. /10 **bonus**

**Total: /47**