Mazin Haider <- replace with your name

CS 480 Spring 2025 Programming Assignment #01

Due: **Sunday, February 23, 2025, 11:59 PM CST**
Points: **100**

Instructions:

1. Place **all your deliverables (as described below) into a single ZIP** file named:

```
LastName_FirstName_CS480_Programming01.zip
```

2. Submit it to Canvas Assignments section before the due date. **No late submissions will be accepted**.

Objectives:

1. (100 points) Implement and evaluate two informed search algorithms.

Input data files:

You are provided two CSV (comma separated values) files (see Programming Assignment #01 section in Canvas):

- `driving.csv` - with **driving distances** between state capitals.
- `straightline.csv` - with **straight line distances** between state capitals.

You **CANNOT** modify nor rename input data files. Rows and columns in those files represent individual state data (state labels/names are in the first row and column). Numerical data in both files is either:

- a non-negative integer corresponding to the distance between two state capitals,
- negative integer -1 indicating that there is no direct “road” (no edge on the graph below) between two state capitals.

Deliverables:

Your submission should include:

- Python code file(s). Your py file should be named:

  ```
  cs480_P01_AXXXXXXXX.py
  ```

  where `AXXXXXXXX` is your IIT A number (this is REQUIRED!). If your solution uses multiple files, makes sure that the main (the one that will be run to solve the problem) is named that way and others include your IIT A number in their names as well.

- this document with your results and conclusions. You should rename it to:

  ```
  LastName_FirstName_CS480_Programming01.doc
  ```

Problem description:

Consider the graph presented below (fig. 1). Each node represents a single state (or the District of Columbia (DC)). If two states are neighbors, there is an edge between them.

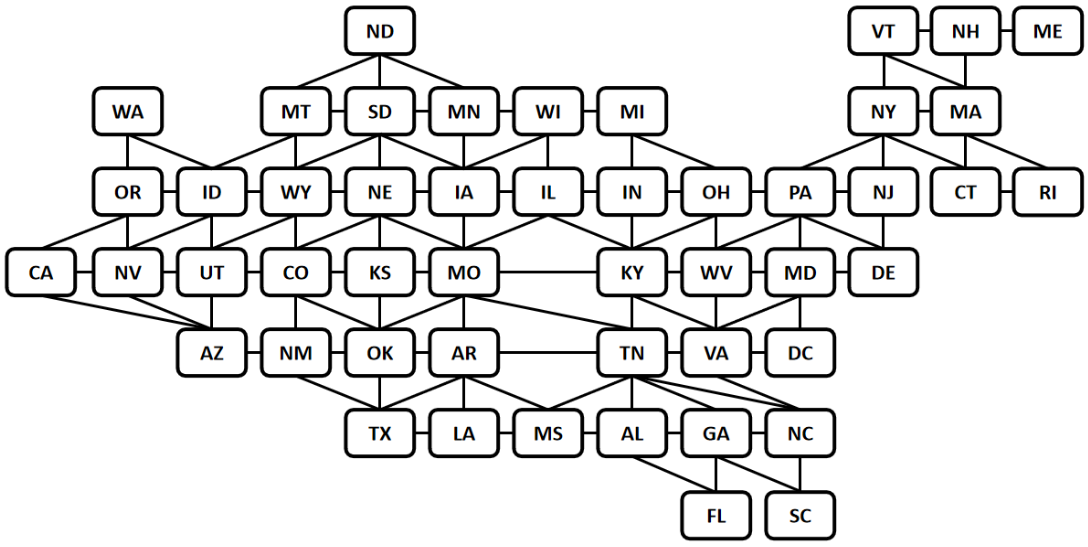

Figure 1: A graph representing all 48 contiguous US states and District of Columbia.

Assume that edge weights represent **driving distances between state capitals**.

Your task is to implement in Python two informed search algorithms:

- Greedy Best First Search algorithm, and
- A* algorithm,

and apply them to find a path between two state capitals using provided data.

Your program should:

- Accept two (2) command line arguments corresponding to two states / state capitals (initial and goal states) so your code could be executed with

  ```
  python cs480_P01_AXXXXXXXX.py GOAL INITIAL
  ```

  where:

  - `cs480_P01_AXXXXXXXX.py` is your python code file name,
  - `GOAL` is the label/name of the initial state.
  - `INITIAL` is the label/name of the initial state,

  Example:

  ```
  python cs480_P01_A11111111.py WA TX
  ```

If the number of arguments provided is NOT two (none, one, or more than two), your program should display the following error message:

```
ERROR: Not enough or too many input arguments.
```

and exit.

- Load and process both input data files provided (assume that input data files are ALWAYS in the same folder as your code - this is REQUIRED!). Make sure your program is **flexible enough to accommodate different input data sets** (with a different graph of states and distances). Your submission will be tested using a different set of files!
- Run Greedy Best First Search and A* algorithms searches to find a path between `INITIAL` and `GOAL` states and measure execution time (in seconds) for both methods.
- Report results on screen in the following format:

```
Last Name, First Name, AXXXXXXXX solution:
Initial state: INITIAL
Goal state: GOAL

Greedy Best First Search:
Solution: STATE1, STATE2, STATE3, …, STATEN-1, STATEN
Number of expanded nodes: AAAA
Number of stops on a path: X1
Execution time: T1 seconds
Complete path cost: Y1


A* Search:
Solution: STATE1, STATE2, STATE3, …, STATEN-1, STATEN
Number of expanded nodes: AAAA
Number of stops on a path: X2
Execution time: T2 seconds
Complete path cost: Y2
```

where:

- `AXXXXXXXX` is your IIT A number,
- `INITIAL` is the label/name of the initial state,
- `GOAL` is the label/name of the initial state,
- `AAAA` is the number of expanded nodes (including the root node),
- `STATE1, STATE2, STATE3, …, STATEN-1, STATEN` is a solution represented as a list of visited states (including `INITIAL` and `GOAL` states), for example: `IL, IA, NE,`

If no path is found replace appropriate information with:

```
Solution: NO SOLUTION FOUND
Number of stops on a path: 0
Execution time: T3 seconds
Complete path cost: 0
```

Pick `INITIAL` / `GOAL` state pair (with at least 5 states between them) and run both Greedy Best First and A* algorithms to find the path between them. Repeat this search ten (10) times for each algorithm and calculate corresponding averages. Report your findings in the Table A below.

NH -> AL	**TABLE A: Results comparison**				
Algorithm	Visited states	Number of visited states	Number of expanded nodes	Path cost	Average search time in seconds
Greedy Best First Search		22	8	1584	0.0102
A*		24	13	1352	0.0145

What are your conclusions? Which algorithm performed better? Was the optimal path found? Write a summary below

Conclusions
Overall, it seems as if the A* algorithms performed better. There was a search in which the A* executed quicker than the GBFS. I'm not sure why this is. For the duration of correcting my program, I found that A* was performing better than GBFS. There may be other factors that are causing this difference as I was working on my program as well. Also, considering A* is to be more optimal and cheaper.

External sources used:

https://www.geeksforgeeks.org/dealing-with-rows-and-columns-in-pandas-dataframe/?ref=lbp

https://www.geeksforgeeks.org/python-read-csv-using-pandas-read_csv/

https://stackoverflow.com/questions/21800169/python-pandas-get-index-of-rows-where-column-matches-certain-value

https://blog.ehoneahobed.com/how-to-remove-the-first-item-in-a-python-dictionary

https://www.geeksforgeeks.org/python-next-method/

https://www.freecodecamp.org/news/sort-dictionary-by-value-in-python/