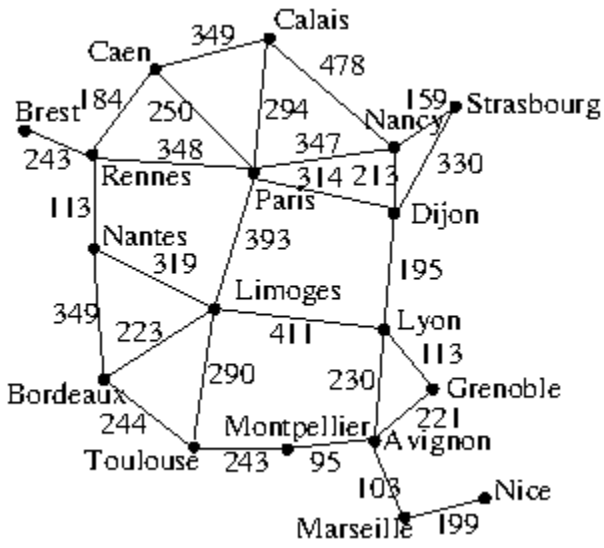
CS331
Spring 2024
Programming Assignment 1 – Search
Due April 22nd at 11:59pm PST
PROCEDURES AND LATE POLICY REMINDER

- **Turn-in:** Please commit your final working code to GitHub classroom (https://classroom.github.com/a/JQF7KaHD) by the deadline.
- **Deadline**: The on-time deadline for all students is 11:59pm PST on the due date.
- **Late policy**: Assignments turned in after the deadline will be accepted but penalized 20% per day. Once the solutions have been given, a late assignment will not be accepted.

Problem Statement
In this assignment, we are going on vacation to France. Bon voyage, je vous attends!
We need to know the best ways to get around, and we only know coordinates and relative distances between certain cities. Your job is to create an agent that uses uninformed and informed search algorithms to find a path from any given city to any other, or return null if there is no path. For this and every assignment, you must implement your solution using only the standard libraries that Python3 provides. If you want to use external libraries, please ask me first. You may use any version of Python3 that suits you, but I will use Python 3.11 for grading.

Your solution must work for any state-space search problem, so be sure to create classes in your code that are abstract enough to use for other problems (for instance, you will at least need a class that keeps track of the map, a class that keeps track of each city, and a class that defines the actions that an agent can take).

The map that you will use for this assignment is pictured below and takes the form of a weighted graph:



Your agent must use multiple different types of search:

1. Breadth-first search (bfs)
2. Iterative deepening depth-limited search (dls)
3. Uniform-cost search (ucs) (you can find this algorithm in your book, or in the Search module)
4. A* search (astar)

The main function of your code should allow for the type of search to be passed as a parameter from the command line (the name of each option is in parentheses above), and the default value should be bfs. The main function should also take as an argument the start and end cities (-A and -B respectively). Your main function should also take as a required argument the name of a file that defines the map. You will be loading in the data needed to create your map from a file called `france.txt`. Each line contains a city and its coordinates, followed by edges that connect that city to another, and the distance from the first city to the second. You can assume that any map file that your agent is tested on will be of the same structure as the `france.txt` file.

In your main function, if no initial and goal city parameters (-A and -B) are passed in, you should compute the path between following initial and goal cities, **using all search methods**:

1. Brest -> Nice
2. Montpellier -> Calais
3. Strasbourg -> Bordeaux
4. Paris -> Grenoble
5. Grenoble -> Paris
6. Brest -> Grenoble
7. Grenoble -> Brest
8. Nice -> Nantes
9. Caen -> Strasbourg

For any initial and goal city, the agent should compute the solution (the path, as a list of cities), the cost of the path, and:

1. The number of nodes explored, entered, or visited (i.e., the number of nodes removed from the frontier)
2. The number of nodes expanded (i.e., the total number of successors)
3. The number of nodes maintained (i.e., stored in the frontier)

For the nine city pairs defined above, your main function should write the path, the cost, and the other three metrics above to a plain text file called `solutions.txt`, and you should include that file in your submission.

For each search method over the nine city pairs, your main function should compute the following:

1. The average number of nodes explored or entered (i.e., the number of nodes removed from the frontier)
2. The average number of nodes expanded (i.e., the total number of successors)
3. The average number of nodes maintained (i.e., stored in the frontier)
4. The number of times it found the optimal solution (optimal here is measured as “found the best solution out of the four search algorithms)

Write these averages to a plain text file named README. Write a paragraph comparing the search algorithms in terms of these metrics. What are the pros and cons of each algorithm? Include this file in your submission.

Deliverables:

You should submit everything below to your GitHub classroom repository: https://classroom.github.com/a/JQF7KaHD

1. README File
2. Solutions file
3. Python code (multiple files)
4. france.txt file