Uniform Cost Route Finder

By Eric McKevitt

Overview:

Uniform Cost Route Finder is a Python-based tool designed for pathfinding in graphs, specifically between city nodes. At its core, the application employs Uniform Cost Search (UCS), a prominent pathfinding algorithm widely used in the field of Artificial Intelligence. UCS is part of a family of search algorithms that are foundational in AI for solving problems that involve navigating through complex spaces or networks. By applying UCS, Uniform Cost Route Finder efficiently calculates the shortest route between cities, demonstrating a practical implementation of these algorithms in a real-world scenario. This project not only showcases the use of UCS in pathfinding but also serves as an illustrative example of how search algorithms are vital in AI for solving optimization and navigation problems.

Features:

- Uniform Cost Search for efficient pathfinding.
- Graphical visualization of city paths.
- Two operational modes:
    1. Standard mode, which utilizes a force-directed layout for node positioning, providing a visually balanced graph based on the network's structure.
    2. Geography mode using real-world city coordinates via `geopy`.

Prerequisites:

- Python 3.9.13 or higher
- Libraries: `matplotlib`, `networkx`, `numpy`, `heapq`
- For geography mode: `geopy` library

Installation:

1. Clone the repository.
2. Install the required libraries:

```
pip install matplotlib networkx numpy heapq
```

3. For geography mode:

```
pip install geopy
```

Usage:

Navigate to the project directory and run the following command in the terminal:

```
python find_route.py [input_file] [source_city] [destination_city]
```

For geography mode:

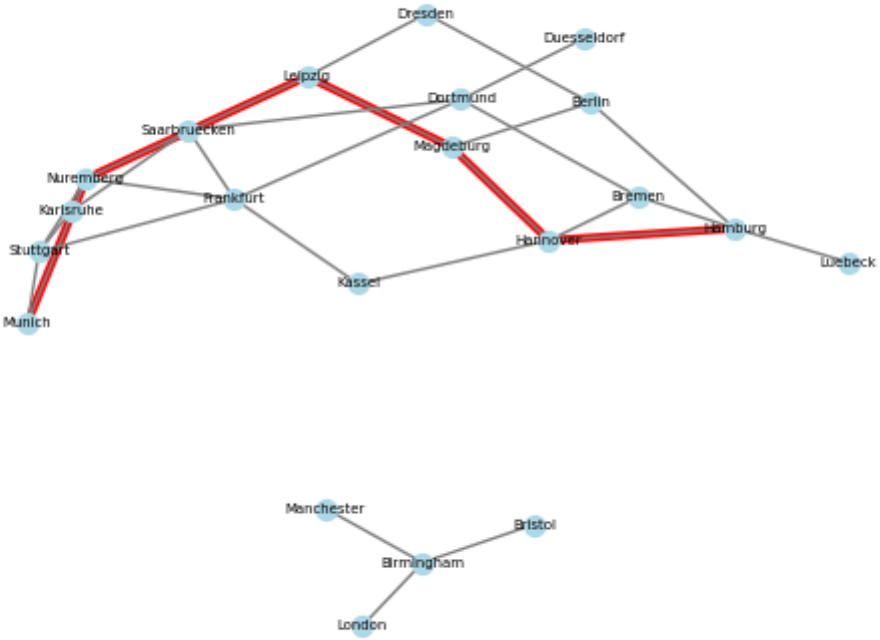
```
python find_route_geography.py [input_file] [source_city]
[destination_city]
```

Example (Standard Mode):

```
python find_route.py inputs/germany.txt Munich Hamburg
```

Output:

```
distance: 860 km
route:
Munich to Nuremberg, 171 km
Nuremberg to Leipzig, 263 km
Leipzig to Magdeburg, 125 km
Magdeburg to Hannover, 148 km
Hannover to Hamburg, 153 km
```

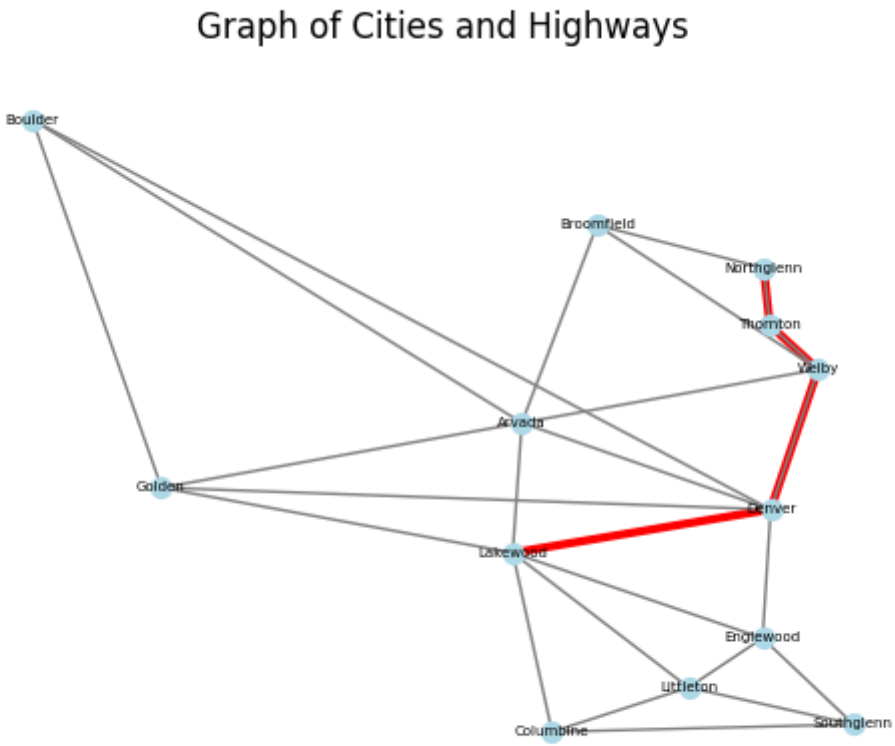


Example (Geography Mode):

```
python find_route_geography.py inputs/colorado.txt Northglenn Lakewood
```

Output:

```
distance: 33 km
route:
Northglenn to Thornton, 4 km
Thornton to Welby, 4 km
Welby to Denver, 12 km
Denver to Lakewood, 13 km
```



Notice how the cities are positioned similarly to their actual geographical locations on a map, offering a realistic depiction of the path's layout. This is because the `geopy` module is able to look up the coordinates of each city at runtime and `networkx` is able to create the graph accordingly.

Input File Format:

The input file should contain city pairs with the distance between them in kilometers, ending with 'END OF INPUT'. For example:

```
CityA CityB 100
CityB CityC 150
END OF INPUT
```