# Programming Project 1: Uninformed Search

1. **Your name and CSM Campus Wide ID (CWID).**

   **Name: Ping Zhang**
   **CWID: 10909957**

2. **What programming language and its version are used for developing your source codes?**

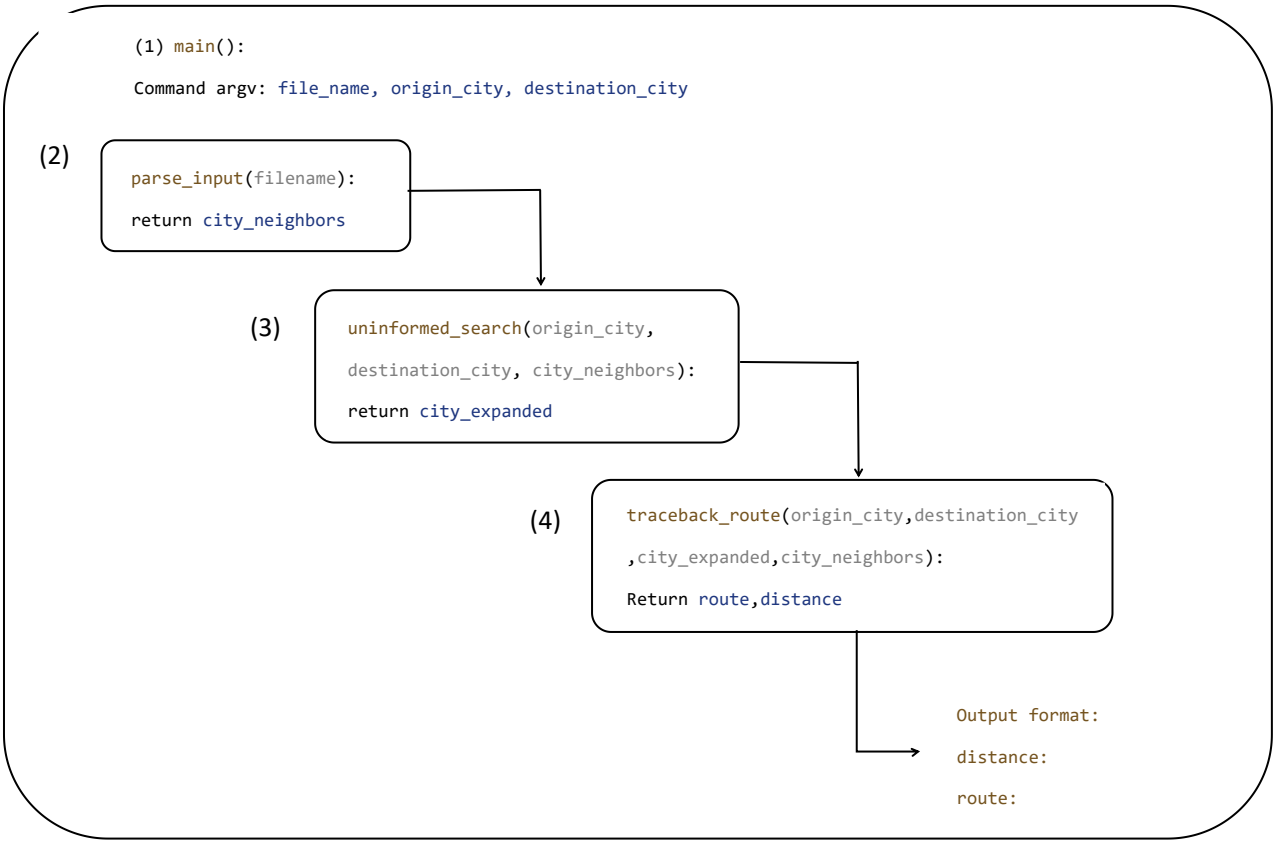
   **Programming language: Python**
   **Python version: Python 3.6**

3. **What OS and its version are used to compile and run the codes?**

   **OS version: Window 11**

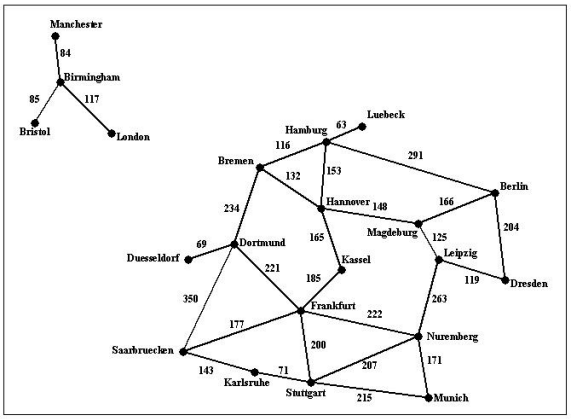
4. **How the code is structured.**

   **There are 4 functions in my code as below:**




(1) **main( )：** Parse Command argv: file_name, origin_city, destination_city

(2) **parse_input(filename):** Using a dictionary data structure named city_neighbors to store all the neighbors for each city. The key point is each city is the key of the dictionary, and the value of the dictionary is a data structure, which key is the city neighbor and the value is the cost between the two cities. The return of this function is a dictionary as below:




| Dictionary: Key | Dictionary: Value | |
|---|---|---|
| City 1 | Dictionary: Key (city neighbor) | Dictionary: Value (the cost between the two cities) |
| | City neighbor 1 | Cost value 1 |
| | City neighbor 2 | Cost value 2 |
| | City neighbor 3 | Cost value 3 |
| | ..... | ..... |
| | City neighbor n | Cost value n |

(3) **uninformed_search(origin_city, destination_city, city_neighbors):** Using dictionary data structure to build the tree search framework named city_expanded. There are several steps in this procedure.

- Using Priority Queue pop the smallest value（the smallest path cost）in the queue.
- Using a dictionary data structure named city_expanded to build the search by loops of the key of the city_neighbors.
- Updating the cost value in the search path, and putting the path cost value into the Priority Queue.
- Do the steps above by a while loop until the Priority Queue is empty.

(4) **traceback_route(origin_city,destination_city,city_expanded,city_neighbors):** Trace backing the search tree to find the shortest path from the origin_city to destination_city by a while loop in city_expanded and city_neighbors. The return of this function is the distance value and list of cities that build the shortest path.

## 5. How to run the code, including very specific compilation instructions, if compilation is needed. Instructions such as "compile using g++" are NOT considered specific.

**Environment : Window 11 + Window 11 + Python 3.6**
**Compilation ： No Need**

There are three steps for execute the code:

- Step 1: locate to the project folder path

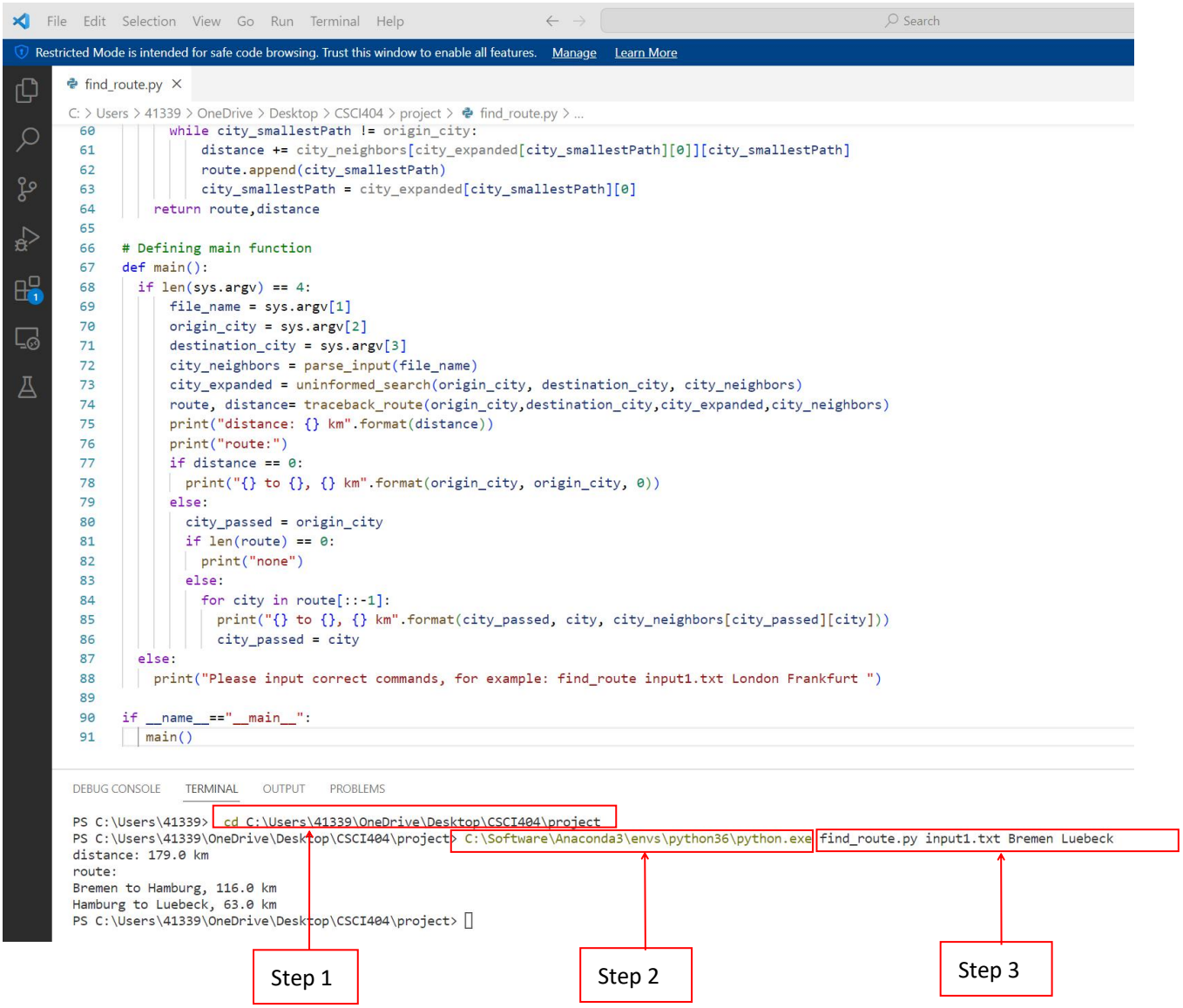
For Example: cd C:\Users\41339\OneDrive\Desktop\CSCI404\project

- Step 2: locate to the Python.exe path

For Example: C:\Software\Anaconda3\envs\python36\python.exe

- Step 3: type the requested command

For Example: find_route.py input1.txt Bremen Luebeck



# Programming Project 1: Uninformed Search

### Design two sets of test cases as described below.

A. 10 points: 5 test cases designed for using the provided map.

B. 10 points: a new map file (visualization like the above figure is not required) and 5 test cases designed for using the new map.

## A: 10 points: 5 test cases designed for using the provided map.

### (1) find_route.py input1.txt Bremen Luebeck

```
PS C:\Users\41339\OneDrive\Desktop\CSCI404\project> C:\Software\Anaconda3\envs\python36\python.exe find_route.py input1.txt Bremen Luebeck
distance: 179.0 km
route:
Bremen to Hamburg, 116.0 km
Hamburg to Luebeck, 63.0 km
```

### (2) find_route.py input1.txt Bremen Berlin

```
distance: 407.0 km
route:
Bremen to Hamburg, 116.0 km
Hamburg to Berlin, 291.0 km
```

### (3) find_route.py input1.txt Hannover Munich

```
PS C:\Users\41339\OneDrive\Desktop\CSCI404\project> C:\Software\Anaconda3\envs\python36\python.exe find_route.py input1.txt Hannover Munich
distance: 707.0 km
route:
Hannover to Magdeburg, 148.0 km
Magdeburg to Leipzig, 125.0 km
Leipzig to Nuremberg, 263.0 km
Nuremberg to Munich, 171.0 km
```

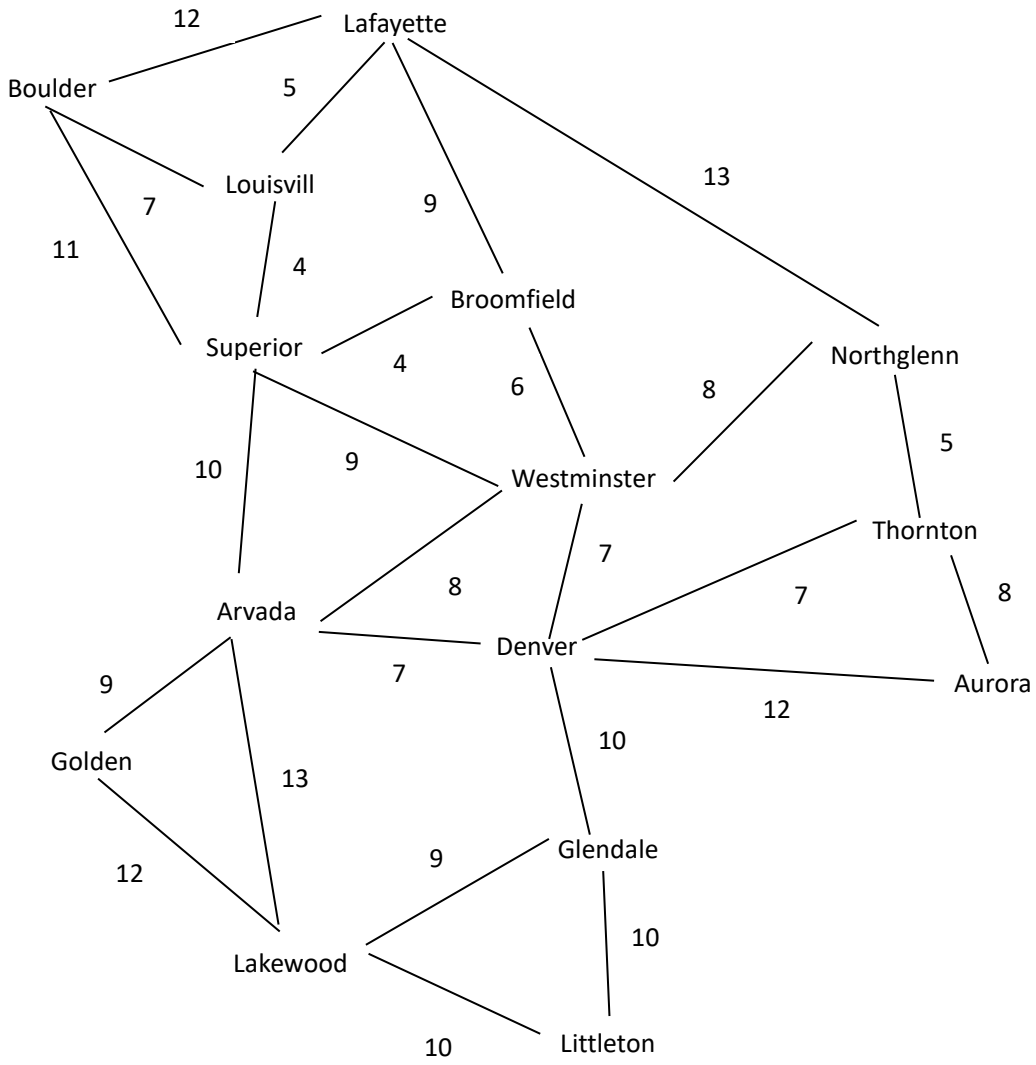
### (4) find_route.py input1.txt Birmingham Duesseldorf

```
PS C:\Users\41339\OneDrive\Desktop\CSCI404\project> C:\Software\Anaconda3\envs\python36\python.exe find_route.py input1.txt Birmingham Duesseldorf
distance: infinity km
route:
none
```

### (5) find_route.py input1.txt Karlsruhe Hamburg

```
PS C:\Users\41339\OneDrive\Desktop\CSCI404\project> C:\Software\Anaconda3\envs\python36\python.exe find_route.py input1.txt Karlsruhe Hamburg
distance: 774.0 km
route:
Karlsruhe to Stuttgart, 71.0 km
Stuttgart to Frankfurt, 200.0 km
Frankfurt to Kassel, 185.0 km
Kassel to Hannover, 165.0 km
Hannover to Hamburg, 153.0 km
```

**B：10 points: a new map file (visualization like the above figure is not required) and 5 test cases designed for using the new map.**




## (1) find_route.py a.txt Boulder Golden

```
PS C:\Users\41339\OneDrive\Desktop\CSCI404\project> C:\Software\Anaconda3\envs\python36\python.exe find_route.py a.txt Boulder Golden
distance: 30.0 km
route:
Boulder to Superior, 11.0 km
Superior to Arvada, 10.0 km
Arvada to Golden, 9.0 km
```

## (2) find_route.py a.txt Superior Denver

```
PS C:\Users\41339\OneDrive\Desktop\CSCI404\project>  C:\Software\Anaconda3\envs\python36\python.exe find_route.py a.txt Superior Denver
distance: 16.0 km
route:
Superior to Westminster, 9.0 km
Westminster to Denver, 7.0 km
```

## (3) find_route.py a.txt Lafayette Littleton

```
PS C:\Users\41339\OneDrive\Desktop\CSCI404\project> C:\Software\Anaconda3\envs\python36\python.exe find_route.py a.txt Lafayette Littleton
distance: 42.0 km
route:
Lafayette to Broomfield, 9.0 km
Broomfield to Westminster, 6.0 km
Westminster to Denver, 7.0 km
Denver to Glendale, 10.0 km
Glendale to Littleton, 10.0 km
```

## (4) find_route.py a.txt Golden Aurora

```
PS C:\Users\41339\OneDrive\Desktop\CSCI404\project>  C:\Software\Anaconda3\envs\python36\python.exe find_route.py a.txt Golden Aurora
distance: 28.0 km
route:
Golden to Arvada, 9.0 km
Arvada to Denver, 7.0 km
Denver to Aurora, 12.0 km
```

## (5) find_route.py a.txt Arvada Thornton

```
PS C:\Users\41339\OneDrive\Desktop\CSCI404\project>  C:\Software\Anaconda3\envs\python36\python.exe find_route.py a.txt Arvada Thornton
distance: 14.0 km
route:
Arvada to Denver, 7.0 km
Denver to Thornton, 7.0 km
```