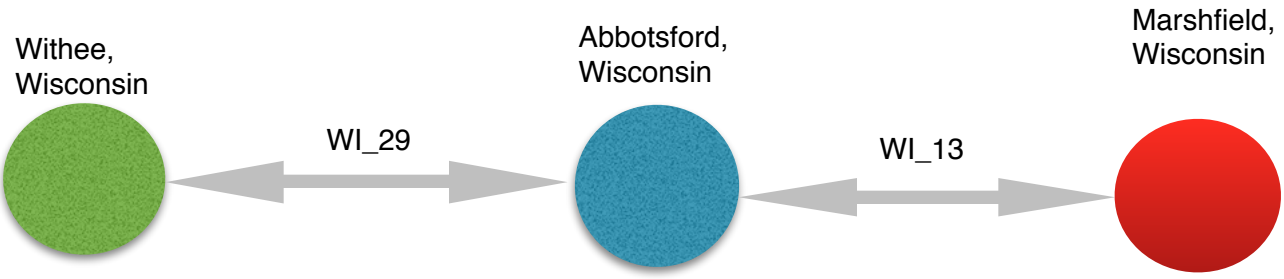
Programming Problem 2:

We tried to represent the Map as a graph with Cities as nodes and roads between the cities as edges

We implemented the graph by using a class. The road between the two cities were added as tuples and since the roads are considered to be double way, we added the reverse of the cities as well.



We read the roads and cities from the files and passed it to generate the graph that we required.

The user enters the input in the format of

[Starting City]_[Starting State] [Destination City]_[Destination State] [Routing Option] [Routing Algorithm]

We intend to validate the Cities to check if it matches any of the vertices in the graph before we proceed with the search.

Then based on the Routing algorithm the user selects, we go ahead and do the following searches:

1 Breadth First Search

We tried to implement it using queue data structure. The node that is input First is traversed to First and when a node is traversed to, we add it to the queue. This goes on till we find the Destination.

2 Depth First Search

We tried to implement DFS using Stack data structure. The node that is inserted First will be evaluated Last and whenever a node is evaluated, we add it to the stack. This goes on till we find the Destination City.

3 A* Search

For A* search the heuristic function that we agreed to use was the euclidean distance between the current city and the destination. For calculating the distance based on latitude and longitude, we use the following formula

```
    Vertical      = Longitude2- Longitude1
    Horizontal    = Latitude2 - Latitude1
 RawDistance= (sin(Horizontal/2))^2 + cos(Latitude1) * cos(Latitude2) * (sin(Vertical/2))^2
    Temp          = 2 * Raw_Distance * tan2(sqrt(a), sqrt(1-a) )
    Distance      = 3961 * Temp [3961 is the Radius of the earth in miles]
```

We use the Distance found above as the Heuristic and we navigate to the Destination by the following algorithm:

We implement A* search using Priority Queue. When we encounter a node with a less value for the f(n), we add it to the queue with a high priority. At each step, the node with the lowest value for f(n) is evaluated and its f(n) for it's adjacent nodes are found and are added to the Priority Queue. This goes on till the solution is found.

Difficulties faced while solving this problem:

1) The representation of the roads and cities as a graph was difficult to implement while coding.
2) We couldn't associate the distance between the cities as the cost for the node. We assumed it to be equal to 1 and implemented the same. The code has to be further refactored to incorporate distance.
3) We have handled bad data in the output to the maximum extent, but this causes a break in some of the routes between some of the cities.

References:

[1] Formula for calculating distance based on Latitude and Longitude

http://andrew.hedges.name/experiments/haversine/

[2] Python Documentation

https://docs.python.org/2/library/index.html

[3] Python Reference

http://www.python-course.eu/