1 Heuristic ( 15 pts)

Design a novel heuristic function other than euclidean distance or spherical distance. Describe its idea

and calculation. Show in detail whether your heuristic function is consistent or not.

Idea:

The heuristic can be the angle between the two cities scaled by a constant factor

The angle subtended by 2 cities at the center of the earth can be measured and used as a heuristic to calculate the approximate distance between 2 cities. The angle can be calculated from the latitude and longitude information given in the usaroads.pl file.

The sector length is a good heuristic since, even if the earth is spherical in 3 dimensions and not just a circle of two dimensions. The distance can be calculated by considering a ring in form of a circle on the sphere, which passes through the two cities under consideration. The ring is so formed such that it passes through the 2 cities and the angle can be calculated by connecting the 2 cities to the center of the earth.

Approximation:

Since heuristic serves as an approximation to the expected final value, the shape of the ring can be seen as a circle, instead of an ellipsoid which the earth actually is. The ring can be formed such that it covers the 2 cities on the surface of earth, and passes through the other side of the earth. Covering 360 degrees.

Scaling:

Since the backward cost is measured in terms of distance for f value to give proper estimate, the heuristic value should also be in similar measure as the backward cost. Hence the angle calculated in radians has to be converted to degrees and have to be scaled to a particular measure to convert this value to distance heuristic.

Consistency:

The heuristic of angle is not consistent since due to the unique shape of earth,

To see if the heuristic is consistent, it needs to be converted into a distance heuristic to calculate match with the already given value! We know that the circle has in total 360 degree and further, the circumference can be calculated using the radius of earth which is also known. Thus the degree is converted to distance .But the heuristic is not consistent due to the shape of earth not being spherical.

As seen in the above diagram, the orange circle corresponds to the heuristic calculated assuming earth to be a sphere, However, the actual distance between two cities is given by the blue line.due to the unique shape of earth

Heuristic calculation (reference: <https://www.igismap.com/formula-to-find-bearing-or-heading-angle-between-two-points-latitude-longitude/>)

Let the 2 cities be X and Y with latitudes and longitudes, X(xlat,xlong) and Y(ylat,ylong)

\* Angle= atan2( a,b)

Here a and b can be calculated as follows

* a= cos (ylat) \*sin(|ylong-xlong|)
* b= cos(xlat)\*sin(ylat) – sin(xlat)\*cos(ylat)\*cos(|ylong-xlong|)

converting angle from radians to degrees,

angle\_in\_degree = angle \*180/ π

scaling to distance heuristic to calculate f value. Since heuristic and backward cost has to be given equal weightage. Scaling angle to circumference:

scaled distance = angle\_in\_degree /360\* 24901 mi

3. Analysis:

3.1. Experiment on your heuristic function and spherical distance using A\*.

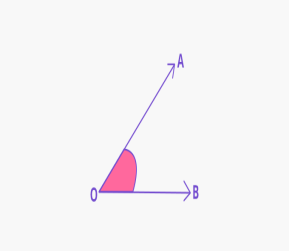
Function:

analysis1\_for\_A\_Star()

If your heuristic function is proven previously to be consistent, try to find a pair of <source, destination> that has a different number of expanded cities and/or maximum size of the queue (any of them that are applicable) during search in experiment results from your heuristic function and spherical distance.Explain the phenomena. If you cannot find one, justify the reason why you think such a pair doesn’t exist.

If your heuristic function is proven to be not consistent, find a pair of <source, destination> where your heuristic function produces a longer path than spherical distance. Explain in detail how such a path is found by A\* algorithm with your heuristic function.

Here the heuristic is not consistent in cases where the shape of the earth is such that the actual distance is greater than the once we get by considering earth to be of spherical shape. Considering the rings that was mentioned in the first part, in my own heuristic, the ring is considered to be in round shape while the earth itself is not round in shape its ellipsoid when projected in 2 dimension



In the above figure the black ring represents the actual earth shape and the grey represents the heuristic that was explained in the first part. The actual distance between 2 cities as estimated by the heuristic is greater than that of actual distance in areas circled in green. Hence in those regions the heuristic is inconsistent.

Result:

('albanyGA', 'albuquerque')

Citites on spherical heuristic path:

Cities in parent list: final : ['albanyGA', 'tallahassee', 'pensacola', 'newOrleans', 'batonRouge', 'lafayette', 'beaumont', 'houston', 'austin', 'sanAntonio', 'elPaso', 'albuquerque']

Cities expanded : ['albanyGA', 'macon', 'atlanta', 'tallahassee', 'pensacola', 'chattanooga', 'newOrleans', 'batonRouge', 'lafayette', 'nashville', 'beaumont', 'houston', 'memphis', 'littleRock', 'mexia', 'austin', 'tulsa', 'dallas', 'lakeCity', 'oklahomaCity', 'sanAntonio', 'jacksonville', 'elPaso', 'tampa', 'laredo', 'ftWorth']

Cities on my heuristic path:

Cities in parent list : : ['albanyGA', 'macon', 'atlanta', 'chattanooga', 'nashville', 'memphis', 'littleRock', 'tulsa', 'kansasCity', 'wichita', 'denver', 'coloradoSprings', 'santaFe', 'albuquerque']

Cities expanded : ['albanyGA', 'tallahassee', 'pensacola', 'newOrleans', 'batonRouge', 'lafayette', 'lakeCity', 'tampa', 'orlando', 'daytonaBeach', 'westPalmBeach', 'miami', 'macon', 'atlanta', 'chattanooga', 'nashville', 'memphis', 'littleRock', 'tulsa', 'oklahomaCity', 'ftWorth', 'kansasCity', 'wichita', 'denver', 'coloradoSprings', 'santaFe', 'grandJunction']

3.2.

Functions:

analysis\_2\_for\_dfs

analysis2\_for\_a\_star

analysis2\_for\_rbfs

3.2.2. Analysis:

DFS:

smallest\_cities\_exp\_pair : ['albanyGA', 'tallahassee'] of value 2

smallest\_size\_of\_queue : ['albanyGA', 'tallahassee'] of value 2

largest\_cities\_exp : ['albanyGA', 'uk2'] of value 112

largest\_size\_q\_pair : ['salem', 'chicago'] of value 71

average\_cities\_exp 57.0

average\_cities\_exp 39.14792471042471

A-star:

smallest\_cities\_exp\_pair : ['albanyGA', 'macon'] of value 1

smallest\_size\_of\_queue : ['europe', 'philadelphia'] of value 1

largest\_cities\_exp : ['raleigh', 'boise'] of value 105

largest\_size\_q\_pair : ['phoenix', 'newHaven'] of value 22

average\_cities\_exp 29.552043114543114

average\_queue\_size 8.358590733590734

RBFS:

RBFS is working for most pairs the ones that I found by plugging the values and testing are:

smallest\_cities\_exp\_pair : ['albanyGA', 'macon']

-The number of cities expanded : 1

● The maximum size of the queue during search : 2

● The final path length : 2

● The final path represented as a sequence of cities: ['albanyGA', 'macon']

Possible largest pairs:

The number of cities expanded : 33

● The maximum size of the queue during search : 43

● The final path length : 33

● The final path represented as a sequence of cities: ['raleigh', 'greensboro', 'charlotte', 'augusta', 'savannah', 'jacksonville', 'lakeCity', 'tallahassee', 'albanyGA', 'macon', 'atlanta', 'chattanooga', 'nashville', 'memphis', 'littleRock', 'tulsa', 'kansasCity', 'stLouis', 'indianapolis', 'cincinnati', 'dayton', 'columbus', 'cleveland', 'buffalo', 'toronto', 'saultSteMarie', 'thunderBay', 'winnipeg', 'calgary', 'vancouver', 'seattle', 'portland', 'boise']

3.

From the values of the analysis part, it is clear that the average number of citites expanded and the average queue size is the least for A-star algorithm. Therefore, the best algorithm to solve the path finding problem on US roads is A-star.

The DFS search might go in a wrong path all the way before identifying that the path was wrong , this is comparatively time consuming although memory requirement is not that high. Since DFS is an uninformed search, the unnecessary traversal of wrong sub trees sometimes is unavoidable. This implies unnecessary nodes are stored and removed from memory

A-star does a better job by keeping in memory only the nodes that have the highest potential to lead us to the right node directly. It ensures that we go to the least expensive node first.

RBFS on the other hand, only keeps track of 2 nodes whose f cost is the lowest. This is far too less use of memory which might appear to save space in the beginning, but finally proves to be worse since it ignores the other states and this might lead us to continually traverse the already traversed path, which leads us to gather more states and consumes much more space even in straight forward but longer paths in the roads between 2 cities.

4.

analysis4\_for\_A\_star

analysis4\_for\_dfs

Analysis:

The hardest pair was calculated based on the below criteria:

* the pair with largest length of expanded cities is chosen
* The pair with largest queue size maintained is also taken

For these two pairs A and B. The sum of largest length of expanded cities and largest queue size maintained is computed and compared. The larger one corresponds to the hardest pair.

DFS:

The hardest pair for DFS is

Src: Salem

Destn: Chicago

With queue size: 71

And

Maximum len of cities expanded:105

AStar:

largest\_cities\_exp : ['raleigh', 'boise'] of value exp: 105 and queue size 15

The hardest pair for DFS is

Src: 'raleigh'

Destn: 'boise'

With queue size: 15

And

Maximum len of cities expanded:105

RBFS:

RBFS is working for most of the pairs. I’ve worked really hard to fix as much as I could. The hardest pair I found by plugging in the values are:

The number of cities expanded : 33

● The maximum size of the queue during search : 43

● The final path length : 33

● The final path represented as a sequence of cities: ['raleigh', 'greensboro', 'charlotte', 'augusta', 'savannah', 'jacksonville', 'lakeCity', 'tallahassee', 'albanyGA', 'macon', 'atlanta', 'chattanooga', 'nashville', 'memphis', 'littleRock', 'tulsa', 'kansasCity', 'stLouis', 'indianapolis', 'cincinnati', 'dayton', 'columbus', 'cleveland', 'buffalo', 'toronto', 'saultSteMarie', 'thunderBay', 'winnipeg', 'calgary', 'vancouver', 'seattle', 'portland', 'boise']