CSC520: Spring 2020 Assignment 2

Question 1 - 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.

Answer – We are given with longitude and latitude of the cities, and assuming the earth is a sphere. We can assume that we will create a sphere using the two points for which we need heuristic function. Once we have two points on a sphere, we can find out the distance between them.

Heuristic function used – Formula to find out distance between 2 points:

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi 1 \cdot \cos\varphi 2 \cdot \sin^2(\Delta\lambda/2)$$

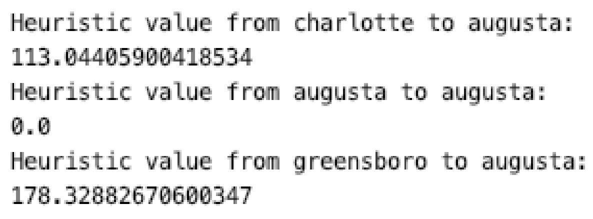
$$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R \cdot c$$

Where φ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km)

Admissible - This method will determine the great circle distance between two points on a sphere given. The great – circle distance will give us the shortest path between two points, which will make heuristic admissible, because it will always try to get the actual/optimal distance between those two points on sphere and will never overestimate the values. The distance between two points is Euclidean distance, this is the length of a straight line between them, but on the sphere, there are no straight lines. So, for the points that are not directly opposite to each other, we can create virtual circle and two points will divide the circle in two arcs, smallest arc would be the shortest distance between those points.
*To make the distance admissible and consistent, I have divided the final heuristic value with 2.

To prove – Lets check for Augusta to Augusta, Charlotte to Augusta, and Augusta to Greensboro.
Start – Augusta **Goal** – Augusta **Cost** – 0 **Heuristic** – 0 (Same city)
Start – Charlotte **Goal** – Augusta **Cost** – 161 **Heuristic** – 113 (Neighboring city)
Start – Augusta **Goal** – Greensboro **Cost** – 252 **Heuristic** – 178 (City with one city in between)
Heuristic value would have to be less than the actual cost, which is the case in all the three scenarios.



```
Heuristic value from charlotte to augusta:
113.04405900418534
Heuristic value from augusta to augusta:
0.0
Heuristic value from greensboro to augusta:
178.32882670600347
```

Above is the screenshot of output from program

Consistency – Heuristic is always said to be consistent or monotone, when its estimate is always less than or equal to the estimated distance from any neighbouring vertex to the goal, plus the cost of reaching that neighbour.

For every node N and each successor P of N, the estimated cost of reaching the goal from N is no greater than the step cost of getting to P plus the estimated cost of reaching the goal from P. That is:

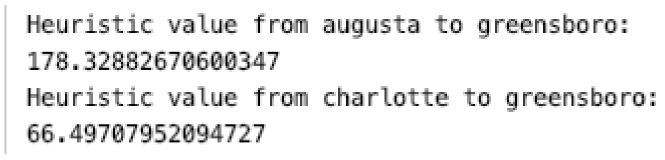
$$H(N) <= C(N-P) + H(P) \text{ and } H(Goal) = 0$$

To Prove – Lets check for consistency now by taking example from Vancouver to Thunder Bay. For heuristic to be consistent

Heuristic(Augusta – Greensboro) <= Heuristic(Charlotte – Greensboro) + Cost(Augusta – Charlotte)

$$178.32 < 66.49 + 161$$

$$178.32 < 227.49$$



```
Heuristic value from augusta to greensboro:
178.32882670600347
Heuristic value from charlotte to greensboro:
66.49707952094727
```

Above is the screenshot from Program.

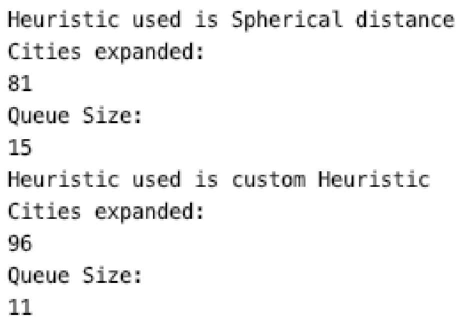
Hence proved that Heuristic is both Admissible and Consistent. This formula is also known as Haversine formula.

Reference - https://www.movable-type.co.uk/scripts/latlong.html

Question 3 - Experiment on your heuristic function and spherical distance using A*. 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.

Answer – My custom heuristic is consistent and when I try to run the program from "Vancouver" to "Miami" with given spherical distance heuristic and my custom heuristic, there was difference in number of cities expanded and maximum size of the queue.



```
Heuristic used is Spherical distance
Cities expanded:
81
Queue Size:
15
Heuristic used is custom Heuristic
Cities expanded:
96
Queue Size:
11
```

Above is the screenshot from program when ran from Vancouver to Miami. Actual path came was same as it was by spherical distance heuristic but there was a difference in number of nodes expanded and in queue. This is because number of nodes to be expanded will depend f(n) i.e sum of node value from cost to start and heuristic its value. Astar will choose the node having least value of f(n) and expand it. Astar will always look for the leaf nodes having least value of f(n) so we cannot control which direction the algorithm will go and how many nodes it will expand but it will give the optimal path, which mean path having the least cost which was the case in our scenario also.

2. Experiment on DFS, A*, Recursive Best-First Search using spherical distance if a heuristic function was needed. For each algorithm, run experiments for all pairs of <source, destination> and calculate following criteria. Report their maximum, average and minimum values across all pairs. (You don't need to report these statistics for each individual pair).

- The number of cities expanded during a single path search.

- The maximum size of the queue during a single path search.

Answers: Below are the stats that I found after running the programs using spherical distance heuristic

	DFS	**A***	**RBFS**
Maximum nodes expanded	111	98	164379
Minimum nodes expanded	0	0	0
Average nodes expanded	56	29	1896
Maximum Queue Size	66741	25	33

3. Consider the relative performance of the algorithms and state which algorithm is the best for this problem. Justify your answer.

Answer – Astar algorithm works the best for this problem in my case. It expanded least number of nodes and have maintained very less queue size in the execution. Astar have also shown fast execution as compared to DFS and have taken less memory as compared to RBFS.

4. Using the criteria above, identify the hardest pair of <source, destination> for each algorithm.

Answer –

DFS – Charlotte - Greensboro

Astar – Miami - Seattle

RBFS – Sanfrancisco – Uk2