REPORT

****BFS Algorithm****

State Space - All cities reachable from the current city by road
Successor function- First element from the fringe
Goal State- Destination city
The program explores all neighboring cities using the concept of queue. The cost function has no effect on the output of the BFS.

****DFS Algorithm****

State Space -All cities reachable from the current city by road
Successor function- Last element from the fringe
Goal State- Destination City
The program explores routes using the concept of stack. The cost function has no effect on the output of the BFS. This is the most inefficient algorithm as it explores a node till it doesn't find the goal city. So even if there is a direct route from the source to destination, the DFS algorithm never reaches till there.

****Uniform Cost Search****

State Space - All cities reachable from the current city by road
Successor function- City with lowest distance to the next city
Goal State- Destination city
The algorithm determines the path by using a cost function. Priority Queue has been used to implement this algorithm.

1.Segments:
Edge weights- All edge weights are set to 1.
In this case the priority is given to node with minimum segments. The cost of each segment is set to be 1. Therefore a node with minimum number of segments is explored.

2.Distance:
Edge weights- Distance between cities
In this case the priority is given to the node with minimum value of distance. Therefore a node with minimum distance is explored. This gives the shortest distance to reach to the goal.

3.Time:
Edge weights- Time required to reach from one city to another. Calculated by distance(city,next city)/speedlimit(city,next city)
In this case the priority is given to the node with minimum time required to reach the next node. The time is calculated dynamically by dividing distance with speed limit of the segment.
Note - If the speed is unknown or equal to zero, the corresponding path is ignored by the algorithm

4.Longtour:

REPORT

Edge weights- Distance between cities

In this case the priority is given to the node with maximum value of distance. Therefore a node with maximum distance is explored. This is done by negating the priority.This gives the longest distance to reach to the goal.

****PERFORMANCE ANALYSIS:****

a.Routing Options:

1.Distance- Astar seems to work the best, as it gives almost optimal result in shortest time.

2.Time- Astar seems to work the best, as it gives almost fastest route in shortest time.

3.Segments- In this case, BFS, Astar and Uniform have similar performances. There is no drastic difference observed.

b.Computation time:

DFS gives the fastest results compared to other algorithms.

For eg:-

time python route.py Benicia,_California Boston,_Massachusetts bfs distance

Computation time = 8.347s

time python route.py Benicia,_California Boston,_Massachusetts uniform distance

Computation time = 8.42Sec

time python route.py Benicia,_California Boston,_Massachusetts dfs distance

Computation time = 3.8 sec only.

time python route.py Benicia,_California Boston,_Massachusetts astar distance

Computation time = 5.035 sec only.

This proves that dfs has least computation time than all others.

c.Memory Requirement:

DFS has least memory requirement. This is because of reduction of the explored state space.

time python route.py Benicia,_California Boston,_Massachusetts bfs distance

Total states explored = 5330

time python route.py Benicia,_California Boston,_Massachusetts uniform distance

Total states explored = 11244

time python route.py Benicia,_California Boston,_Massachusetts astar distance

Total states explored = 4878

time python route.py Benicia,_California Boston,_Massachusetts dfs distance

Total states explored = 4132

This proves that dfs has lowest memory requirement than all others.

NOTE: Though DFS has least time and memory requirements, it gives the worst results amongst all the algorithms.

****Astar HEURISTIC FUNCTION:****

REPORT

1.Distance:
State Space - All cities reachable from the current city by road
Successor function- Element with lowest cost(n)+heuristic(n) value
Goal State- Destination city
Edge weights- Distance between cities

Heuristic function: Distance of a city from the goal city. This distance has been calculated using the great circle distance between the latitude & longitu de of cities.
The function is admissible because it would always calculate the shortest distance of reaching from a city to the goal city. Hence it will always direct us towards the shortest route for reaching the goal city.
Cost function: Total distance travelled till now.
Using the above-mentioned h(n) and c(n), the state space was reduced to more than half and hence results in better performance.

2.Segments:
State Space - All cities reachable from the current city by road
Successor function- Element with lowest cost(n)+heuristic(n) value
Goal State- Destination city
Edge weights- All edge weights are set to 1.

Heuristic function: Least number of edge required to reach from a city to the goal city. This is set to 1.
The function is admissible because it would always have least number of segment for reaching from a city to the goal city i.e 1. Hence it will always direc t us towards the route with least edges for reaching the goal city.
Cost function: Total number of edges travelled till now.

3.Time:
State Space - All cities reachable from the current city by road
Successor function- Element with lowest cost(n)+heuristic(n) value
Goal State- Destination city
Edge weights- Time required to reach from one city to another. Calculated by distance(city,next city)/speedlimit(city,next city)
Heuristic function: Least amount of time required to reach to the goal city. The time has been calculated by dividing the distance(city,goal)/speed(city,g oal)
Distance(city,goal)= Calculated using the great circle distance between the latitude & longitude of cities.
Speed(city,goal) is assumed to be same as speed(city,next_city)
The function is admissible because it would always calculate the least time for reaching from a city to the goal city. Hence it will always direct us towar ds the route for goal city which requires minimal time.
Cost function . Total time travelled till now.
Using the above-mentioned h(n) and c(n), the state space was reduced to more than half and hence results in better performance.

REPORT

4.Longtour:
State Space - All cities reachable from the current city by road
Successor function- Element with lowest cost(n)+heuristic(n) value
Goal State- Destination city
Edge weights- Distance between cities

In this case the priority is given to the node with maximum value of distance. Therefore a node with maximum distance is explored. This is done by negating the value of h(n)+c(n). h(n) & c(n) are calculate as similar to the distance cost function of astar. Howver they are negated later, in order to fetch the one with highest value using priority queue.This gives the longest distance to reach to the goal.

****ASSUMPTIONS:****

1. If the speed of a segment is unknown or equal to zero, the path is ignored.
2. In case of highway intersections, we do not have their corresponding latitudes and longitudes available. Therefore for heuristic calculation, we assume that the highway intersection always lies in the path towards the goal.

Therefore,Dist(highway_intersection,goal)=dist(previous_city,goal)-dist(previous_city,highway_intersection)

****IMPROVEMENT:****

We observe that the result obtained by the astar is close to optimal but not equal. This is because using this heuristic we are estimating the distance of the highway intersections from the goal. This results in such tradeoff.

****CITATIONS:****

Great Circle Distance - https://en.wikipedia.org/wiki/Great-circle_distance