CS 686 Assignment 1

Student name: Zefeng Qiu
Student number: 20550621
UW email: z9qiu@uwaterloo.ca

Question 1:

a.

s -> h -> k -> c -> a -> b -> d -> m -> e -> n -> g

b.

i. The heuristic function h is admissible. (proof by contradiction)

Suppose that h is not admissible. There is a h(s') > h*(n), which for some initial state s', h*(n) is the shortest path from n to the goal state.

For goal can be reached in practical h*(n) steps, and every action move one square, performing an auction at most reduce by one, so h(G) = h(s') - h*(n) > 0, which there is a contradiction for h(G) should be zero.

ii. s -> h -> k -> c -> a -> b -> d -> m -> g

iii. s , h, k , c, a, f, p, q, r , t , g

Question 2:

a. When all steps cost are equal, so in this case that uniform search reproduces breath-first search.

b. DFS is best-first search when f(n) = - depth(n), which depth(n) means deep level of the tree.

c. Uniform-cost search is A* search when heuristic function which is h(n) = 0.

Question 3:

a.

States: Agent in one city and is going to the unvisited cites, if all city have been visited, go back to the start city; each city represent nodes, and distance of cities are vertices that connect two nodes;

Initial State: Agent in the start city "A" and has not visited any cities;

Goal State: Agent has visited all the other cities and back to city "A" (the start city);

Successors: Unvisited cities(nodes) in alphabetical order

Operators: Agent go to another city through vertices, their cost is the Euclidean distance between two cities.

The problem then can be abstract as a weighted undirected graph, and firstly all the nodes in this graph is connected to each other. The aim is to find a path in the graph that start from A, connecting all the nodes and at last back to A where it started.

b. Heuristic function:

h(n) = distance of current node to the nearest unvisited cities
+ estimated distance of travel to unvisited cities (Minimum Spanning Tree heuristic used)
+ nearest distance from an unvisited city to the start city(A)

proofing h(n) is admissible (proof by contradiction)

Suppose h(n) is not admissible, according the definition, there is a h(s') > h*(n), which for some state s', h*(n) is actual the shortest path from n to the goal state.

Reaching goal can cost h*(n), so h(G) = h(s') - h*(n) > 0 which contradicts the fact that h(G) = 0, for when reaching the goal city A, there is no unvisited city. Hence, the heuristic function is admissible.

c.

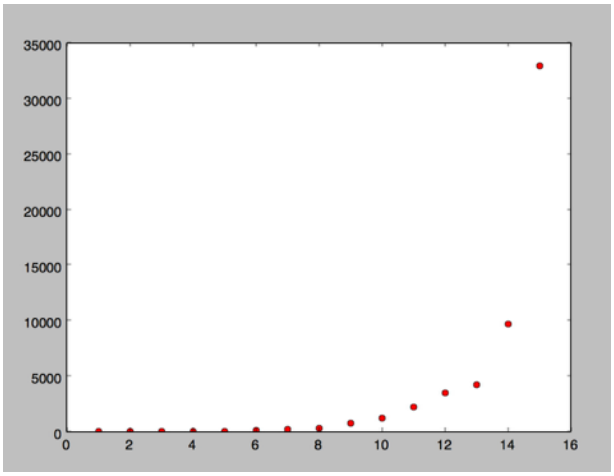


figure 1-1

As figure 1-1 show, the number of node that generated by A*Search grow exponentially as the increase of number of cities to be visited.

instance number	1	2	3	4	5	6	7	8	9	10	average
1	0	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1	1
3	4	4	4	6	4	4	4	6	6	4	4.6
4	11	9	17	9	13	15	9	9	11	11	11.4
5	32	21	26	44	36	34	39	32	18	11	29.3
6	78	55	72	59	46	79	38	30	56	65	57.8
7	363	69	233	103	174	123	250	66	191	22	159.4
8	210	364	373	380	211	327	51	350	74	655	299.5
9	1269	1365	766	309	926	502	202	281	880	387	688.7
10	1123	939	1398	1293	663	73	432	464	383	4939	1170.7
11	485	8915	1902	635	3537	150	2835	341	576	2741	2211.7
12	5885	1263	998	346	3453	5660	9803	646	3677	3084	3481.5
13	1188	10180	2990	4436	652	10356	2551	6058	486	2648	4154.5
14	3242	1392	2462	40861	3016	6222	1020	15189	18534	4608	9654.6
15	100861	2214	6121	1221	120861	1760	1060	3383	90861	699	32904.1
16		1862		6571	8611				11754		

sheet 1-1

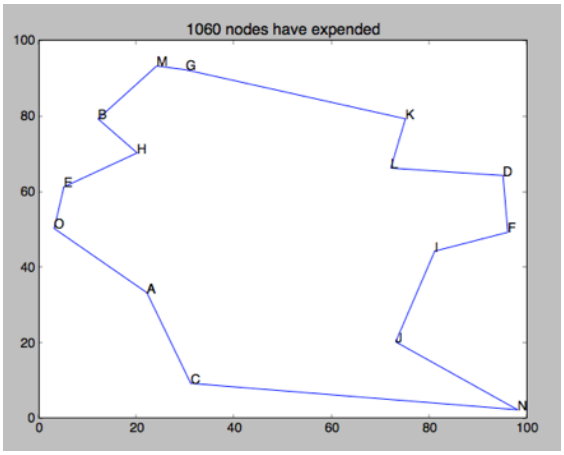
The data showed in sheet 1-1 is the result after running all the test cases. Noticing that the number of nodes expanded have huge gap among the instances.

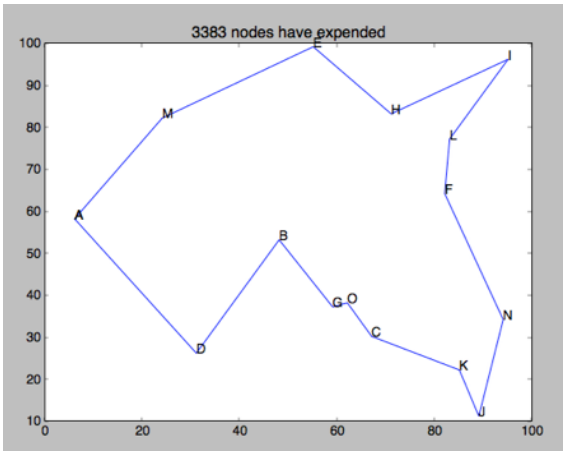
Reason for this phenomenon is owing to complexity of cities' location and the heuristic function, sometimes it is easier to find optimal path which passed all cities and back to city A. The purpose of heuristic function is that, in this TSP problem, simulating and forecasting the following path that agent may travelling to, and the path is minimum.

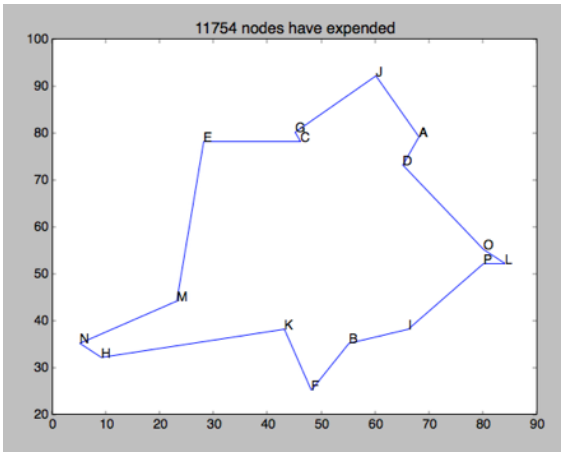
When heuristic function is better fitting and simulating the problem, nodes generated in A* search algorithm are less, and this algorithm seems more smart.

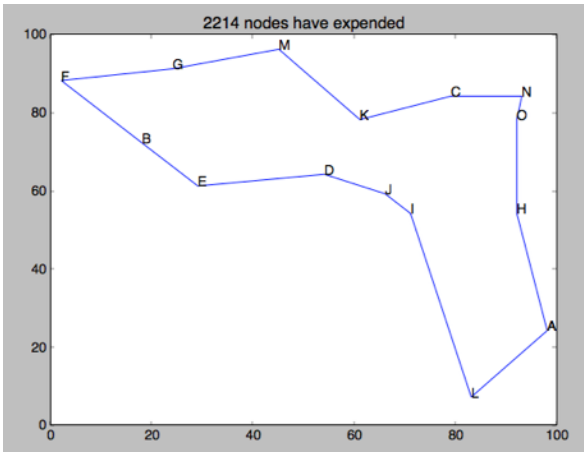
When running instances in 16 nodes/ cities, it is possible to give the optimal solution very quickly due to the heuristic function is suitable to the instance and its complexity. However, honestly in other cases it cannot produce the result in a reasonable time.

Here is some examples of optimal solution of instance 15 and 16:









Explain the Code:

The code consist 6 .py files, which are a1_main, ai, node_fringe, node, city, min_spanning_tree;

1. ai_main.py is the main function: open the file and read data from it, and then store the data into City distance and call method in AI class to perform the A* search, after algorithm complete, plot the result using method from matplotlib.pyplot library;

2. city.py is the data model to store basic “city” information includes city name and its coordinate — x, y value, the method in it are to calculate distance to A, next node, and unvisited;

3. mini_spanning_tree.py is for calculating minimum spanning tree using prim algorithm of unvisited cities, return the length of minimum spanning tree;

4. node.py is a data model for performing A* search algorithm, containing methods —heuristic function, g(cost) function and their sum — f function;

5. node_fringe is also a data model for performing A* search, mainly has a list called queue which provides functionality such as adding node or a list of nodes to list, pop the minimum node which has minimum f value;

6. ai.py is for executing the algorithm, firstly adding A city as start node to queue, generating fringe from A’s neighbour and calculating their f value respectively; then pick the minimum value node in fringe and add it to current node, expanding this current node according to its unvisited cities, calculating their f value and adding to fringe again; loop this process until there is no unvisited node in the city.