CIS 561 Project #1
Due Date: Oct 15, 2014
Demo by Oct 24th, 2014

Please submit the printout project report in class and upload source code to MyCourse.

Team Project: you may work as a team of 2 persons for this project.

(**Modified** Russel&Norvig, Page 119, 3.30) The traveling salesperson problem (TSP) can be solved with the minimum- spanning-tree (MST) heuristic, which estimates the cost of completing a tour, given that a partial tour has already been constructed. The MST cost of a set of cities is the smallest sum of the link costs of any tree that connects all the cities.

1 Model this problem as a search problem, including, how to define the state space, start states, operators, path cost and goal state.

2 Show that how the MST heuristic can be derived from a relaxed version of the TSP, and it is admissible.

3 Write a problem generator for instances of the TSP where cities are represented by random points in the unit square. (You may use existing code with reference)

4 Find an efficient algorithm for constructing the MST, and use it with the A * graph search to solve instances of the TSP. (You may use existing code with reference)

5 Implement an A* search algorithm to solve the problem. How far can you go with the A* approach, as the problem size (#cities increases) increases, record the time (measured by the number of nodes explored) and space (measured by the maximum number of nodes stored in the open list during the program execution) needed for a certain size of problem.

6 When the problem size increase, A* runs out of memory. So implement the SMA* to solve the memory limit problem. Put a fixed limit on the number of nodes that you can store in open list, show that with the same amount of memory available, SMA* is able to solve larger size of problem while A* cannot solve. What is the solution found by SMA*, is it optimal?

7 Sometimes you need find a solution within limited amount of time, you need trade off the computation time with the solution quality. Modify the A* to an anytime A* and try it with different w values, record the results you get. Show the multiple different solutions found by the anytime A* as search time increases.

8 Implement a GUI that allows the choices of #cities, memory size (#nodes) and the algorithm to run, and demonstrate the solution both graphically and numerically.

What to handled in:

1. A report that describes the design to solve the problem (Question 1). The experimental results and answers to questions 4, 5, 6 and 7. Give reference to any type of help you have used. (Hard copy of the report submitted by Due date)
2. The source code with documentation and instruction of how to run it. (Uploaded to mycourse by Due date).
3. Demo your program to the instructor by Demo Due date.

Grading Policy:

1. Basic documentation on how to solve this problem using search. (20pt)
2. Implementation of the problem instance generator. (5pt)
3. Implementation of an efficient algorithm for constructing the MST. (5pt)
4. Working program with at least A* search and experiment results (25 pt)
5. SMA* works and show experiment results (20pt)
6. Anytime A* works and show experiment results (25pt)
7. A GUI Implementation (Extra 20 pt)

Search Code Available

Download the Search package (in Java) from MyCourse, which includes the implementation of a set of general search framework. You can use this package as the foundation of your coding, or you can choose to start from scratch if you prefer another programming language.

To compile: javac search/*.java
To execute: java search/SearchApp

There are also search code available from the textbook website:
http://aima.cs.berkeley.edu/code.html

Note: these packages are complimentary and you are solely responsible for your own code to work correctly and in time.