Traveling Salesman Problem (TSP)

The Traveling Salesman problem is a classic computer problem that has received a great deal of attention in the computer science literature. It is one example of an NP-Complete problem, which means that there is no known solution technique that performs substantially better than simply just searching through all possible alternatives. Given a group of cities with a fixed travel distance between each pair, the problem is to find the shortest tour that visits each city exactly once.

The Traveling Salesman problem can be solved through a combinatorial search of all possible tours. Therefore, it is well suited for a Replicated Worker solution. As partially completed tours are generated during the search, they can be stored in the Work Pool. Each Worker process removes a partial tour from the Work Pool, computes all possible extensions to the tour to one more city, and then writes all these new partial tours back into the Work Pool.

The cities are numbered from 0 to n, and the travel distances are given by a two-dimensional array distance, such that distance[i,j] gives the travel distance from city i to city j. For simplicity, assume that all tours start at city 0, visit all the other cities exactly once, and return to city 0 at the end. The Traveling Salesman problem is to find the tour with the minimum total travel distance.

A partial tour is just a sequence of city numbers beginning with city 0. Associated with each partial tour is the travel distance for that tour. To extend a partial tour, one must consider all the remaining cities not yet in the tour: each of these is a potential candidate for the next city to be visited in the partial tour. Each Worker process reads a partial tour from the Work Pool, computes all possible one city extensions to the tour, and then writes them all back to the Work Pool.

A global minimum is maintained for the shortest complete tour found so far. Whenever any Worker finds a complete tour, it is compared with this global minimum, and the global minimum is replaced if necessary. After the Replicated Worker program terminates, this global minimum is the final answer.

The algorithm described above is suitable for writing a Replicated Worker program for the Traveling Salesman problem. However, the performance can be improved through a minor modification. In the above algorithm, each tour is extended by one city and then returned to the
Work Pool. Thus, all the partial tours grow gradually, one city at a time. As the algorithm progresses, the number of such partial tours grows explosively requiring a large amount of computation, and possibly even overwhelming the storage system.

The combinatorial explosion can be reduced by using a heuristic search technique, which attempts to find good tours first, and then uses these to eliminate bad tours at early stages of their creation. When a Worker process reads a partial tour from the Work Pool, it selects the next city for the tour as the nearest of the remaining cities. Partial tours for the other cities are also computed, but these are immediately written back into the Work Pool. Then the worker extends its partial tour by one more city, by finding the nearest of the cities that still remain unvisited. The Worker keeps pushing forward with its partial tour in this way until all the cities are visited, and a complete tour is generated.

This algorithm is really almost identical to the one described previously. The Worker still computes all possible one city extensions to each partial tour. However, instead of immediately writing all of them back to the Work Pool, the shortest of these partial tours is retained and extended, while the others are written back to the Work Pool. In this way, each Worker pushes forward quickly to find some reasonably good tours, which are recorded in the global minimum.

Then the global minimum can be used to eliminate many partial tours before they are completed. As each new partial tour is generated, its travel distance is compared to the global minimum. In this way, many bad tours that jump long distances between cities are eliminated in their early stages. This simple heuristic greatly reduces the number of partial tours generated. Also, it reduces storage requirements for the Work Pool, because the Workers continue to push partial tours quickly towards complete tours, and thus eliminate them from the Work Pool.

The data set given with this assignment is named graph.txt and it is for the following graph.
You are to use OpenMP to identify parallelism in this program.

Use this data set to make sure your algorithm is working.

When your algorithm is working, there are large data sets available at http://www.tsp.gatech.edu/. These data sets list cities by their latitude and longitude. There are many sites on the web that show how to calculate the distance between two cities specified by latitude and longitude.

Notes:

- Make sure your program conforms to the Programming Style document on the course home page.
- Write a report that shows your results in a way that will impress the grader. Your report must be in one of the following formats: .doc, .docx, .pdf, .odt.
- Submit your finished program to the Eagle system, including your makefile, as a compressed tar file.