**Traveling Salesperson Problem – Using Brute Force**

Will B. Crawford

CECS

Speed School of Engineering

University of Louisville, USA

[wbcraw02@louisville.edu](mailto:wbcraw02@louisville.edu)

1. **Introduction**

In order to study the Traveling Salesperson Problem, or T.S.P., the brute force method was used in order to show that more optimized algorithms need to be used. The reason for this is because using the brute force method will generate every possible permutation for the tour you could complete. Using this method, the amount of calculations needed grows exponentially. This is due to the nature of performing all the calculations you need at the beginning. The space complexity then rises as well, because you need a way to store all the possible combinations of tours and their respective weights.

1. **Approach**

The algorithm used in this problem utilized a lookup-table to store all the distances between cities and recursion to solve the problem. First the algorithm iterates over all cities in a array and computes the distances between each city. You can refer to the pseudo-code below for reference.

Then, using the starting city of 1, it iterates over cities and recursively calls a function who permutes the combinations of tours. The algorithm used is assuming that the first city will always be the starting city and will keep a current tally of the tour. Then, once the tour has been completed, it will compute weight of the current tour using the lookup table. Then, if that weight is better than the current best tour, then it will update the best weight and tour. You can refer to the pseudo-code below for reference.

This algorithm will always compute the best path until it gets to the end, there can be some optimization done there so that it knows when to stop computing that path and continue with the algorithm. More optimization was done during the computation of permutations by spawning threads.

1. **Results**

The algorithm at first, ran very well. Then, as the dimension of cities increased, the running time slowed down very quickly. This is because the complexity of the algorithm is and space is . Therefore, at very small inputs, the algorithm is very effective, but as soon as any kind of data approaches to production level, the algorithm would never be able to perform well. The program could not compute the problem when the input reached to 12 cities.

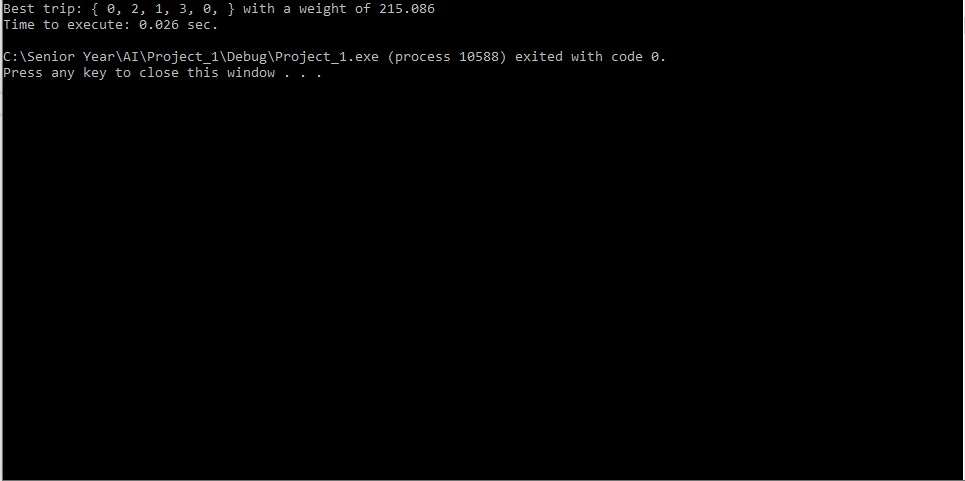
* 1. **Data** (Describe the data you used.)

The data used was generated by a tool that randomizes Travelling Salesperson Problem nodes, the tool can be found in the references. In the following section, there are the outputs to all the given TSP files.

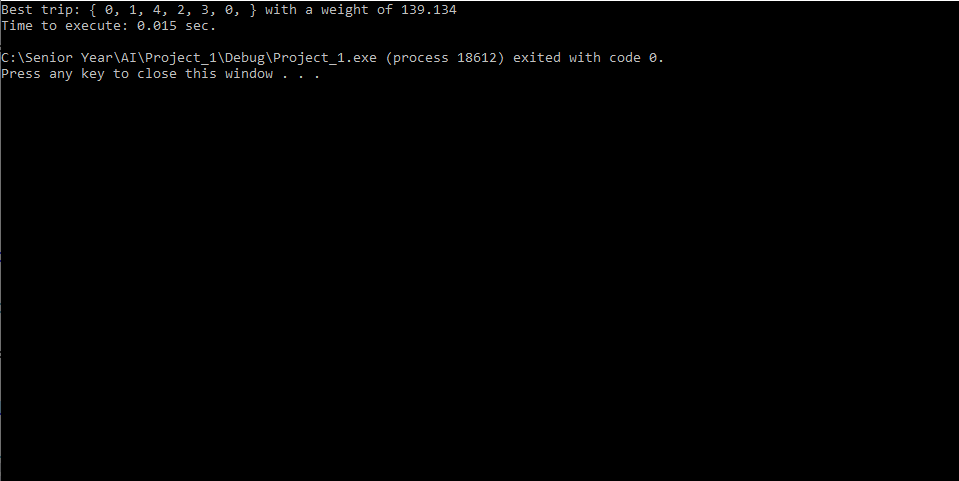
* 1. **Results** (Numerical results and any figures or tables.)

*Figure 1:*

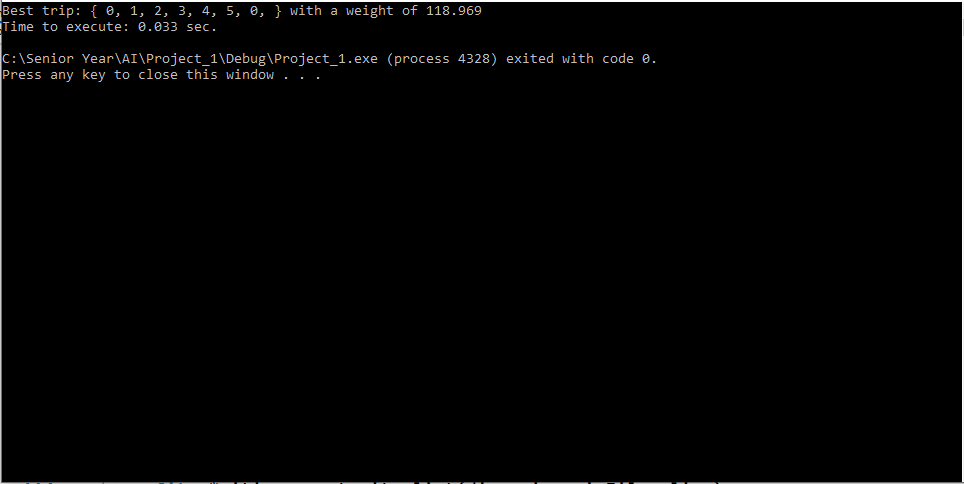
*Figure 2:*

**

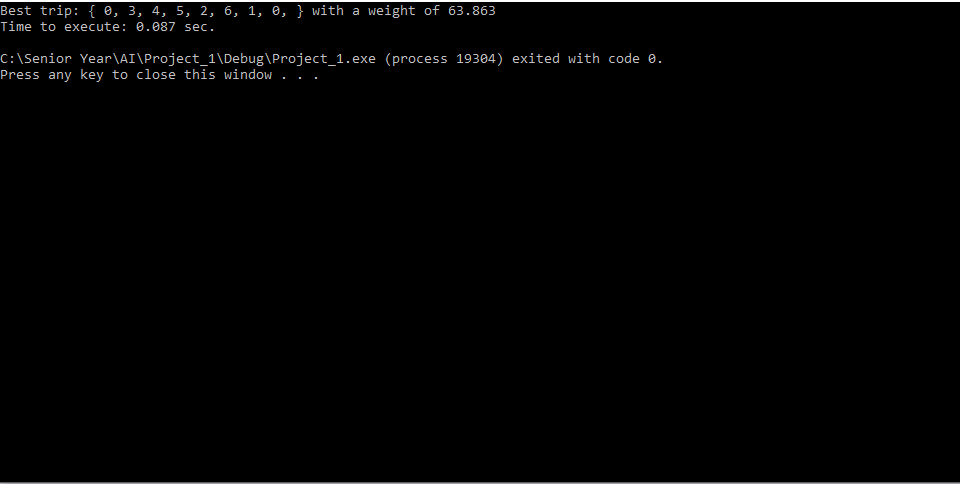
*Figure 3:*

**

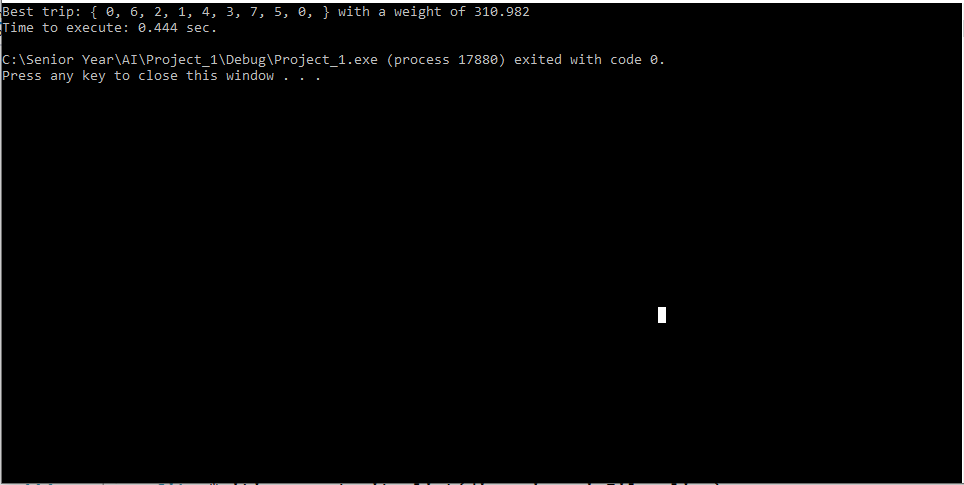
*Figure 4:*

**

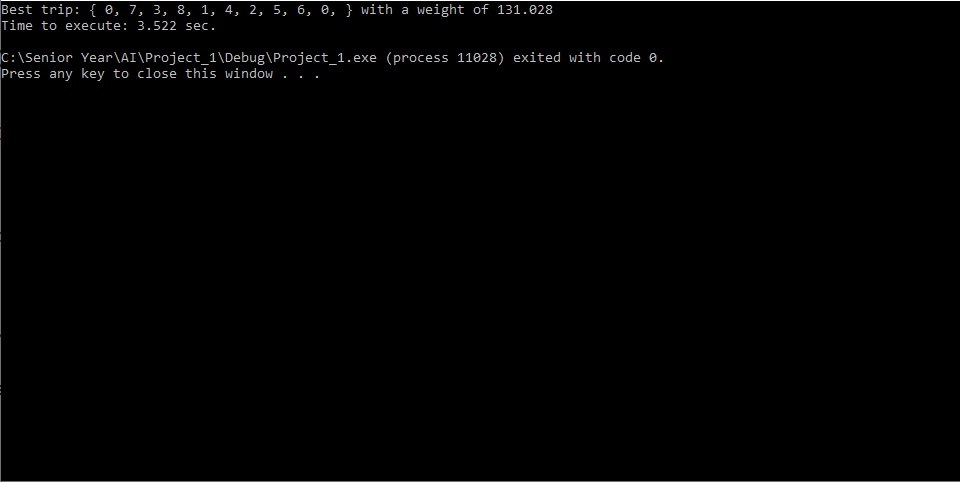
*Figure 5:*

**

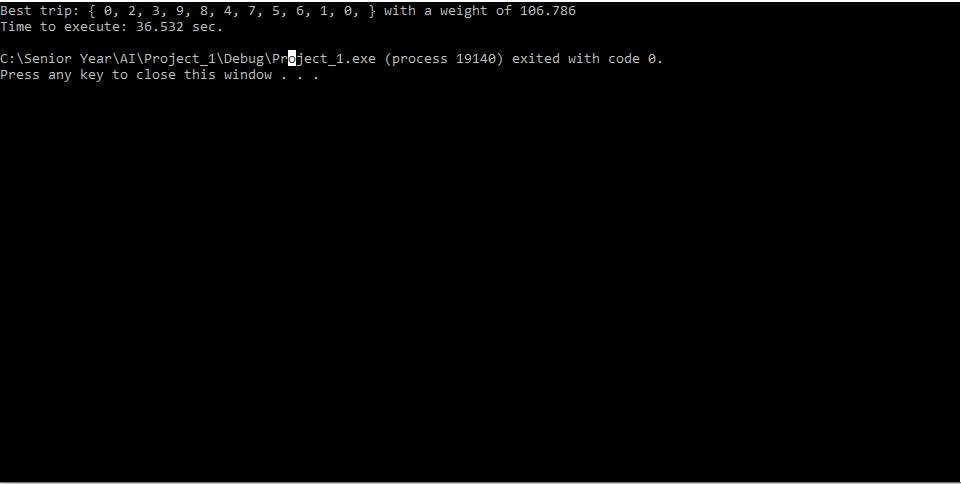
*Figure 6:*

**

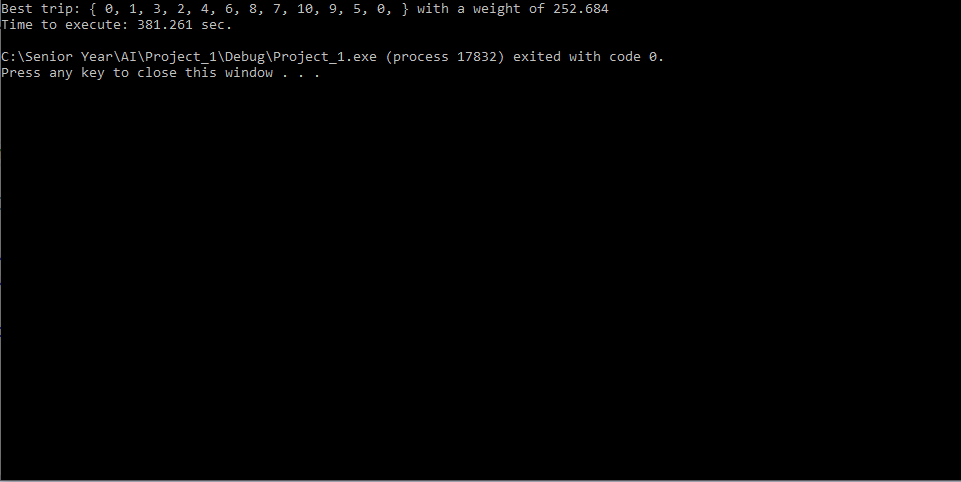
*Figure 7:*

**

*Figure 8:*

**

*Figure 9:*

**

1. **Discussion**

As you can see from *Figure 1* above, the running time increases exponentially to city size of 11 and no longer runs at city size of 12. This is due to the amount of data being generated. In order to increase the efficiency of the algorithm, it would be better to not use brute force method. Algorithms that do not generate every combination would be significantly better to use because those algorithms can find the global best solution without needing to generate every possibility.

1. **References**

- http://www.math.uwaterloo.ca/tsp/