CSE-545 Artificial Intelligence

Hunter Ellenberger

Project 1

I first started by writing a function named “get\_coordinates.” This function’s primary purpose was to take the data from the .tsp file and place all the vertex data into a dictionary named “coordinates.” The keys for this dictionary are what would later be placed into the “permutations” function, which was loaded in from the itertools library. The permutation function then generates all paths which could be taken from point to point. These paths would then be run through a function I wrote called “permutations\_compute” which generates the total distance of the paths between every coordinate combination. These paths along with their distances are then added to a dictionary called “tspMap.” The keys for this map being the distance traveled along the path and the values being the path itself. The final TSP related part of the program is used to get the minimum value which is just acquired by using the built-in python “min” function on the keys of the “tspMap.” After writing all these functions and debugging them to make sure everything worked correctly, I then added and implemented the “time” library so that I could get an accurate measurement of the runtime of each TSP file.

The following TSP files took so little time that they could not be measured in milliseconds:

TSP 4:

The runtime of this TSP file was 0.0ms. There ended up being two optimal routes for this file that resulted in a total distance of 215.08553303209044. These two paths that achieved this minimal cost were 1 -> 4 -> 2 -> 3 -> 1 and 3 -> 2 -> 4 -> 1 -> 3.

TSP 6:

Surprisingly, this file reached a runtime of 0.0ms on my system when TSP 5, a file with less coordinates, did not. There ended up being nine optimal routes that resulted in a total distance of 118.96891407553862. Every other TSP problem ends up having more than a handful of solutions so going forward I will only note one of the many optimal solutions. For this file one of the paths that achieved minimal cost was 4 -> 5 -> 6 -> 1 -> 2 -> 3 -> 4.

The following TSP files now take anywhere from tens of milliseconds to hundreds of milliseconds. This increase can be seen getting exponentially larger as increased cost must be computed to find the shortest route.

TSP 5:

This was the first file to reach a runtime over 0.0ms. It clocked in at 6.975ms (rounded to the thousandth place). There were nine optimal paths for these coordinates and one of these paths were 1-> 2 -> 5 -> 3 -> 4 -> 1.

TSP 7:

The runtime of this file ended up reaching 31.856ms. There were fourteen optimal paths that minimized the cost at 63.863. One of these optimal paths was 1 -> 2 -> 7 -> 3 -> 6 -> 5 -> 4 -> 1.

TSP 8:

The runtime of this file clocked in at 192.172ms. There were just four optimal paths associated with this file and this is where we start to see a pattern of files having less and less optimal routes as the number of nodes gets larger. An optimal path associated with this file is 4 -> 5 -> 2 -> 3 -> 7 -> 1 -> 6 -> 8 -> 4.

The following TSP files runtimes are starting to become extenuatingly bad due to the exponential nature of generating more permutations. This computation simply starts to become too much for modern hardware.

TSP 9:

The runtime of this file is 1,950.387ms, as you can see the runtime has increased more than ten-fold compared to the previous file, this will only get worse. Following the pattern that was established by the previous file, this file only has one optimal path. Its cost is 131.028 and the path itself is 6 -> 7 -> 1 -> 8 -> 4 -> 9 -> 2 -> 5 -> 3 -> 6.

TSP 10:

The runtime of this file is 21,893.464ms (about 22 seconds). Although it has more than one optimal path it still has fewer than previously seen with the earlier TSP files. There are three optimal paths that have a cost of 106.786 and one of those paths is 1 -> 2 -> 7 -> 6 -> 8 -> 5 -> 9 -> 10 -> 4 -> 3 -> 1.

TSP 11:

This is the last final file that I was able to compute, and this can be seen by the runtime of this file reaching a total of 308,112.836ms (about 5 minutes). Surprisingly, this file also breaks my previous claim that as the node amount increases the permutations of optimal routes decreases. This file had fifteen total optimal paths, each with a cost of 252.684. One of these optimal paths was 1 -> 6 -> 10 -> 11 -> 8 -> 9 -> 7 -> 5 -> 3 -> 4 -> 2 -> 1.

TSP 12:

Not computable by my system.