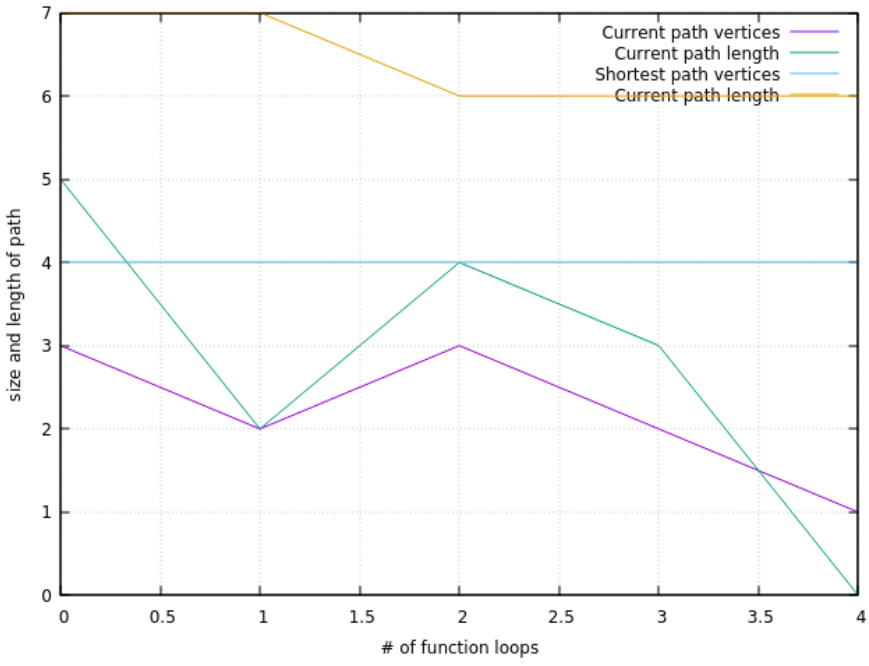
WRITEUP - Assignment 4 - The Perambulations of Denver Long

In this writeup, the nature of the Travelling Salesman Problem will be discussed, and the method for finding the Hamiltonian cycle from a particular point will be described, in regards to the most recent assignment. However, before any of that is further discussed, it's important to first understand what's happening:

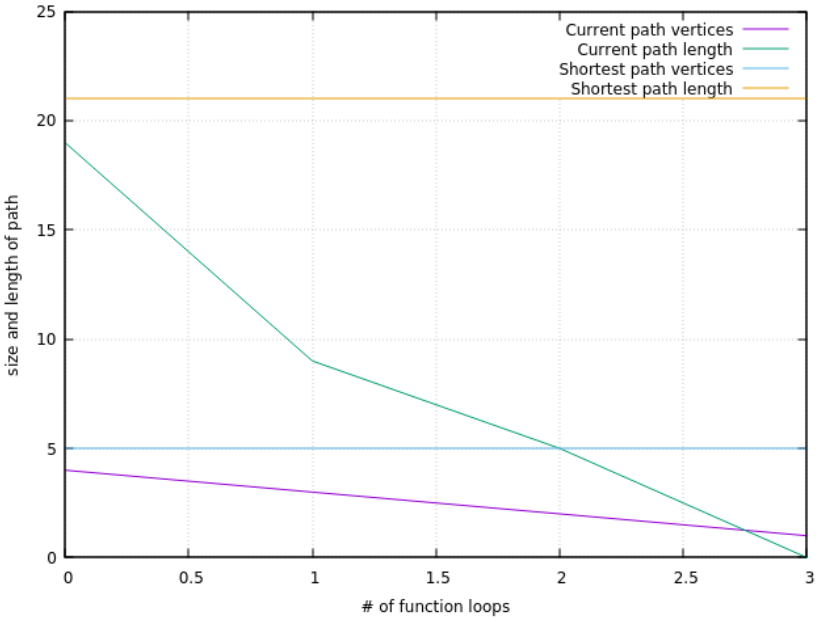
```
salamander1223@Vivobook1223:~/cse13s/asgn4$ ./tsp -i texas.graph
Path length: 61
Path: San-Antonio -> Del-Rio -> Lubbock -> Amarillo -> Bandera -> El-Paso -> Abilene -> Houston
 -> Waco -> Luckenbach -> Pecos -> Austin -> Midland -> Galveston -> Dallas -> Corpus-Christi -
> Odessa -> San-Antonio
Total recursive calls: 2290792
salamander1223@Vivobook1223:~/cse13s/asgn4$
```

Several different header files that contain the abstract data types necessary for this assignment are being loaded into tsp.c, which is then executing a particular function called tsp(), which prints to the screen the shortest path in which every vertex is reached, and the path ends where it started. The weight of each edge from vertex to vertex is stored and represented by an adjacency matrix. This is known as the Travelling Salesman Problem.

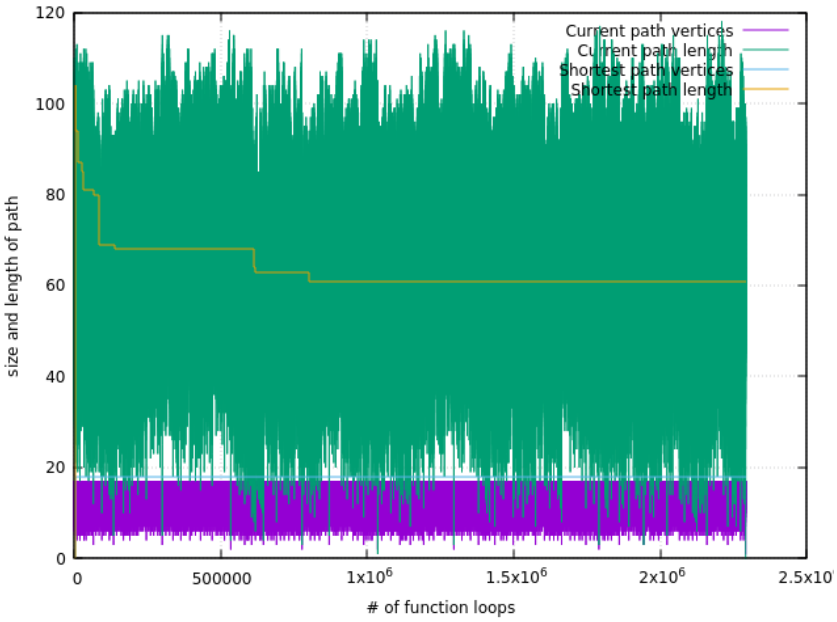
So what makes this problem so well known? It has to do with the computational complexity of finding the shortest path in an adjacency matrix. What first must be realized is that not all points are equidistant, or have the same weighted edge. What's also important to note is that not all vertices are connected, which prioritize certain routes over others. It's as though you're attempting to navigate through a maze, and you have next to no idea the proper route to take, because you're completely unaware of the possible paths ahead. The way tsp.c has tackled this problem is



through a depth-first search. In the image shown above, the current path being traversed is constantly changing, and the shortest path recorded only ever decreases, in vertices and length. If the function finds, in its recursive search, a Hamiltonian cycle that's shorter than the shortest path recorded, then it replaces it, as seen at x=2.



With smaller graphs, the amount of recursive calls remains relatively constant, but this is only with smaller graphs, erring on the side of 10 or less vertices. The depth-first search solution to this problem has a computational complexity of *O(n!)*, which can yield an absurd number of computations. In the graphs shown above, the number of vertices included in the corresponding adjacency matrices, was no more than four, but in the graph shown below, the vertices number 17, which generates an obscene number of possible paths, only a handful of which could be Hamiltonian. As is shown by the line describing the shortest path length, the number of computations, ranging in the millions, to find an otherwise small improvement, ranging in the tens, to the smallest path recorded, demonstrates the poor algorithmic approach to this problem, and hopes to demonstrate the beauty of the Travelling Salesman Problem.



Note: Credit for any pseudocode and any other separate lines of code go to Professor Long and the staff of CSE13S Fall 2021, as is provided in the assignment documentation.