CS 225 Final Project Results

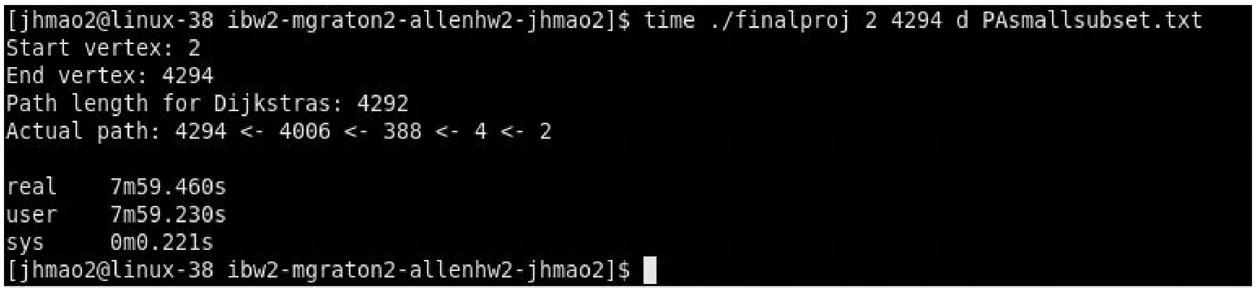
Molly Graton, Jonathan Mao, Bella Watters, Allen Wang

Preface

Our group's original goals were to use the Pennsylvania Road Network dataset (found on the Stanford Dataset Network) to find the shortest path between nodes (ends or intersections of roads). The algorithms we implemented were Djikstra's algorithm and A* algorithm, along with a general BFS iterator.

Outcomes

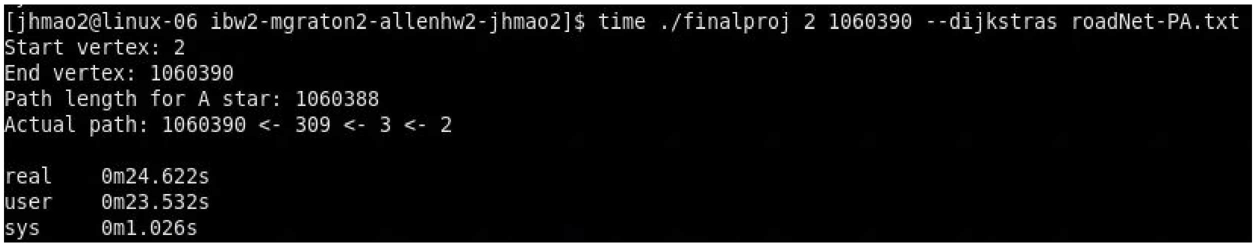
We were successfully able to implement Dijkstra's algorithm. This algorithm, based on the weights of the edges in the graph, finds the shortest distance from the inputted starting node to the inputted ending node. It returns both the path of edges taken in the shortest path, along with the optimal weighted numerical distance between the two nodes.



```
[jhmao2@linux-38 ibw2-mgraton2-allenhw2-jhmao2]$ time ./finalproj 2 4294 d PAsmallsubset.txt
Start vertex: 2
End vertex: 4294
Path length for Dijkstras: 4292
Actual path: 4294 <- 4006 <- 388 <- 4 <- 2

real    7m59.460s
user    7m59.230s
sys     0m0.221s
[jhmao2@linux-38 ibw2-mgraton2-allenhw2-jhmao2]$
```

We also successfully implemented A* algorithm. This algorithm also aims to find the shortest path between two inputted nodes. When running on large datasets, this algorithm definitely showed to be more efficient and faster.



```
[jhmao2@linux-06 ibw2-mgraton2-allenhw2-jhmao2]$ time ./finalproj 2 1060390 --dijkstras roadNet-PA.txt
Start vertex: 2
End vertex: 1060390
Path length for A star: 1060388
Actual path: 1060390 <- 309 <- 3 <- 2

real    0m24.622s
user    0m23.532s
sys     0m1.026s
```

The implementation of our breadth first search algorithm went very smoothly considering that we became familiar with it over the semester. We decided to use this iterator in the print function so that we could utilize its capabilities in a helpful way (both for visualization and debugging).

Discoveries

In regards to the coding aspect of our project, we made a few discoveries in our algorithm implementations. In Djikstra's algorithm, using INT_MAX will not work for setting initial distance to infinity since the data type of weights are integers. Thus, whenever we add the weight of an edge to INT_MAX, it causes overflow and makes the resultant value to be a number close to INT_MIN. This causes the if statement to be always true and update the distance vector with wrong values. We instead found it useful to use a large number that will not overflow.

Additionally, we use a priority queue to store unvisited vertices, with the order based on distances. Since we want the vertex with the shortest distance to be at the top of the priority queue, we need a min heap in this case. However, the default priority queue provided by C++ is a max heap, and in order for it to be a min heap, we have to declare the priority queue with comparison type as an additional input argument.

In regards to the actual project discoveries, something interesting we discovered was the time needed for the two algorithms to run. Dijkstra's Algorithm took much longer than A* algorithm when run on the same dataset. As our original idea was to compare the speed of the two, it was interesting to see the stark difference in timings. This is likely attributed to the fact that A* algorithm stops once the ending node is found, while Djikstra's goes through the entire graph regardless. We did end up altering Djikstra's to stop at the end node, but the development of it was still interesting.

Additionally, we were able to find the shortest path between two nodes. However, since the data did not provide weights, we had to add our own randomly computed weights (using a Python script). While it was interesting to see the algorithm working based on this data, it would be beneficial to move forward utilizing real, weighted data on the algorithm to see how it lines up with actual GPA systems.

The last discovery we made was about the A* search and its heuristic value. The algorithm only produces the shortest path if the heuristic is "admissible", meaning that it never over-estimates the remaining distance. We chose the difference in weights to be our value, but if given more time, we would've liked to look into optimizing this value to provide the best result possible.