

```
/******  
* Project Report Template  
* Project 3 (Map Routing), ECE368  
******/
```

Name: Wen-Hsiang Shih  
Login: wshih

```
/******  
* Explain your overall approach to the problem and a short  
* general summary of your solution and code.  
******/
```

My implementation is based on Dijkstra Algorithm with priority queue (min heap). There are two functions, read\_map and read\_query, that read the given map and find shortest path between two nodes. Since I am given a map with nodes containing only coordinates, I calculate the distance between two nodes and store the data while I am adding edges in the map. The map in my implementation is constructed in the form of Adjacency List which allocates less memory space in contrast to Adjacency Matrix.

```
/******  
* Known bugs / limitations of your program / assumptions made.  
******/
```

The assumptions of my program are:

- The given maps are undirected graph.
- There isn't any negative edges in the map.

```
/******  
* List whatever help (if any) that you received.  
******/
```

References:

- [Binary Heap from Geeks for Geeks](#)
- [Dijkstra's algorithm from Wiki](#)
- [Finding The Shortest Path, With A Little Help From Dijkstra](#)

```
/******  
* Describe any serious problems you encountered.  
******/
```

There were two problems, execution time and memory usage. Originally, I wanted to use cache to store the computation result for every starting point. So I can quickly

lookup the shortest path between two nodes in cache, if I already have the computation result kept in cache. However, this approach allocates too much memory space if the graph is large. Therefore, I decided not to use cache but to compute the result for each query.

I tried to reduce some nested for loops in order to have better performance in time consumption. Eventually, my improved program runs 8% faster than original implementation.

```
real    0m4.241s
user    0m4.216s
sys     0m0.017s
```

```
real    0m3.862s
user    0m3.836s
sys     0m0.020s
```

/\*\*\*\*\*

- \* List any other comments/feedback here (e.g., whether you
- \* enjoyed doing the exercise, it was too easy/tough, etc.).

\*\*\*\*\*/

This project is pretty interesting and I enjoyed a lot while I was programming. It would be much fun if we can decide what algorithm we want to use to solve the given problem, for example Bellman-Ford Algorithm or Floyd-Warshall Algorithm.