```
/*****************************************************************
 * Project Report Template
 * PA04 (Map Routing), ECE368
 *****************************************************************/

Name:  Yi_li     Shutao_Wang
Login: li1652     wang2590

/*****************************************************************
 *  Explain your overall approach to the problem and a short
 *  general summary of your solution and code.
 *****************************************************************/
Overall we used an weighted Adjacency list representation to build this
graph. From the begining of the loading process, we used an array to load
each vertices and the x, y coordinates. Then we use linked-list to connect
each edges with the first element been the parent, the second been the child.
Later on we used dijkstra's algorithm to sort and assign the shortest distance.
Meanwhile, we used a similar structure which performs the function as a priority
queue to keep the dijkstra's sorted results.
assume we have V vertexes and E edges.
Space complexity: O(V) for holding vertexes, O(E) for holding Edges. All recursions are O(V)[Free
function, Track back function]. So, total space complexity is O(V + E).
Time complexity: O(V + E) for load function. Priority queue takes O(V) * search each Node takes O(V) = O
(V^2). There's nothing would takes more time than O(V^2). So, time complexity should be O(V^2).

/*****************************************************************
 *  Known bugs / limitations of your program / assumptions made.
 *****************************************************************/
We assume that the adjacency list representation for building the graph
to have the best time performace. We think that the space complexity will
O(n), since we used one array, each and individual array[] set have s list
And, we used a priority queue to sorted the sorted results, we also assume
 a queue will have the best performance.

/*****************************************************************
 *  List whatever help (if any) that you received.
 *****************************************************************/
During building the graph and laoding, we got the help from lecture notes
22 on the weighted graphs structure. Between adjacency matrix and adjacency list
we picked the list for a better time performance. Also, we got the ideas about
dijkstra's algorithm from the lectures notes as well.

/*****************************************************************
 *  Describe any serious problems you encountered.
 *****************************************************************/
Soooooooo many 0 distance in this PA which is really annoying.



/*****************************************************************
 *  List any other comments/feedback here (e.g., whether you
 *  enjoyed doing the exercise, it was too easy/tough, etc.).
 *****************************************************************/
There are two hard aspects about this coding, first we facing the trouble
of choosing the better structure to load and build this graph. Second, we
were picking between wheather to use min_heap for an priority queue structure
to store the sorted results. Overall, it was a fast pace and well challenged
coding experience.
```