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

Name: Adil Zhakypbek
Login: azhakypb

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

I implemented the default Dijkstra's algorithm using standard queue. Where I would check threw the whole array to find minimum index which resulted in a O(n^2) time complexity. Therefore, when I was done, I realized that my program takes a lot of time for big data.

After that, I thought about implementing priority queue to decrease the time complexity to O(log(n)). This time I was satisfied with the performance.

I also worked on memory allocation to ensure that all memory is freed at the end.

I split the program into three main .c files.

1. Main.c. Which just called for other functions and operated the whole process.
2. edge_map.c. Which read the map, allocated memory for each vertex, operated with edges so they would be correctly connected before Djikastra's algorithm.
3. Query_path.c This is the heart of the program. It traversed the query, called for Djikastra for each separate query. Run MinHeap to get the correct answers.

Finally, important to mention that I call my Dijkstra algorithm for every query, which implies that the total time complexity is O(queryNum * log(n))

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

I made an assumption that the input files will always be in the correct form. The x and y coordinates are always bigger than or equal to 0. The limit of cords do not exceed 10,000. The total amount of vertices does not exceed 100000. When using valgrind I ignored error messages as long as the memory was freed successfully.

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

I have read the articles from geeksforgeeks.org to better understand priority queue and how to correctly implement it in the program.

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

It was a challenge for me to improve the time complexity and correctly
deallocate allocated memory.

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

It was fun to solve the problem with path direction, I actually had to use
some recursion skills to make it work.
It also took me some time to understand how to stop the Dijkstra’s algorithm
once the needed answer was found.
Overall the project was challenging and interesting, I feel like it is worth
adding to my resume. Thank you