The Final Report

Team Name

"Swings"

Team Members (List your team members' name and Student ID)

202001368 박기윤 Giyoon Park

202001733 서정민 Jungmin Seo

202000947 김주연 Juyeon Kim

201602843 임유비 Yoobee Lim

A. Problem Definition

Shortest Path Problem *is an important issue in Industrial Engineering. Many algorithms have been developed such as* ***Dijkstra***, ***Bellman-ford Algorithm****. We want to optimize the algorithm operating time to solve the shortest path problem.*

Recently, near our school, there are some stations of ***Swing****. Swing is one of the companies that provide* ***electric scooter sharing services*** *such as "Lime". So, we will analyze the best* ***swing manager's route.***

B. Goal of the Project

Our team would like to compare the performance of Dijkstra and Bellman-ford Algorithm to solve the Shortest Path Problem.

Before finding the shortest path using Dijkstra and Bellman-ford algorithm, we want to sort the 'priority list of Swing station vertices'. We assume that the Swing Manager should go to the station first that has more swing numbers. Therefore we decided the priority with the number of swings and sorted them in descending order.

When we sort the priority, we used the three different sorting algorithms, Insertion sort, Quick sort, and Merge Sort.

Finally, we would like to compare the performance of Dijkstra and Bellman-ford algorithms according to each sorting algorithms.

For example, if the number of swings at Station-A is 30, Station-B:10, and Station-C: 20, the path will determine A->C->B by sorting algorithms. And then we will find the shortest path of A to C and C to B with the Dijkstra & Bellman-ford algorithm.

Dijkstra Algorithm: *This algorithm is kind of greedy algorithm to find shortest path between one vertex and all other vertices. For example, in a graph, if the vertices each represent a crossroad and the length of the road that the sides connect between the crossroad, the Dijkstra Algorithm can find the shortest path between the two crossroads.*

Bellman-ford Algorithm: *This algorithm is also used to find shortest path between one vertex and other vertices. It takes more times in same problem than Dijkstra. But, this algorithm allows the negative edges. And Dijkstra doesn't allow.*

Insertion Sort: *Insertion sort is an algorithm that completes an alignment by locating and inserting all elements of a data array to the parts of the array that have already been sorted from the front.*

Quick Sort*: Quick sorting sorts lists using a split and conquer method. Choose one element from the list. The element is called a pivot. The list is divided into two based on the pivot. The elements which are smaller than the pivot are placed in the front of the pivot. And the elements which are bigger than the pivot are placed after the pivot. This segmenting of the list into two is called 'division'. After segmenting the list, the pivot no longer moves. We iterate this process recursively for two small lists that are segmented. Recursion is repeated until the size of the list is 0 or 1.*

Merge Sort: *Split the unordered list into n partial lists, each containing only one element. (Because a list of only one element equals sorted). It repeatedly merges until only one partial list remains, creating an ordered partial list. The last remaining partial list is an ordered list.*

C. Simple example (Scenario)

- Our dataset

Column Explanation Of Excel Data

x, y = longitude, latitude of each vertex

degree of vertex =the number of intersection

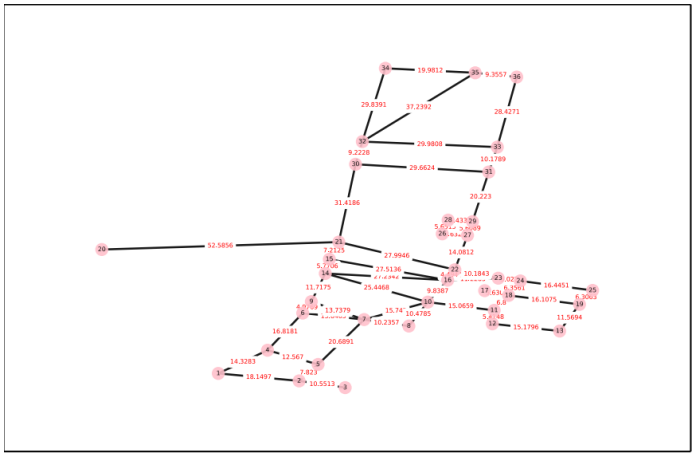
1,2,3,4,5 = opposite vertices of each vertex

pri = average number of E-scooter (period = 2days)

	B	C	D	E	F	G	H	I	J
1	x	y	degree_of_vertex	1	2	3	4	5	Pri
2	127.24712	37.33225	2	2	4				
3	127.24891	37.33195	3	1	3	5			0.5
4	127.24993	37.33168	1	2					
5	127.24821	37.33318	3	1	5	6			
6	127.24933	37.33261	3	2	4	7			
7	127.24899	37.33467	3	4	7	9			
8	127.25035	37.33441	5	5	6	8	9	10	
9	127.25134	37.33415	2	7	10				
10	127.24918	37.33513	3	6	7	14			2.5
11	127.25176	37.33511	5	7	8	11	14	16	
12	127.25323	37.33478	3	10	12	18			
13	127.25319	37.33424	2	11	13				
14	127.25468	37.33395	2	12	19				1.5
15	127.24949	37.33626	4	9	10	15	16		
16	127.24958	37.33683	3	14	16	21			
17	127.2522	37.33599	5	10	14	15	22	23	
18	127.25302	37.33557	1	18					
19	127.25355	37.33538	4	11	17	19	24		
20	127.25512	37.33502	3	13	18	25			
21	127.24454	37.33722	1	21					0
22	127.24979	37.33752	4	20	15	22	30		3
23	127.25236	37.33641	4	16	21	23	27		
24	127.25332	37.33607	3	16	22	24			6
25	127.25381	37.33596	3	18	23	25			
26	127.25541	37.33558	2	19	24				4
27	127.25208	37.33785	2	27	28				
28	127.25264	37.33779	3	22	26	29			
29	127.25221	37.3384	2	26	29				
30	127.25275	37.33834	3	27	28	31			
31	127.25016	37.34064	3	21	31	32			1
32	127.25311	37.34033	3	29	30	33			3.5
33	127.25031	37.34155	4	30	33	34	35		
34	127.2533	37.34133	3	31	32	36			
35	127.25082	37.34449	2	32	35				
36	127.25281	37.34431	3	32	34	36			2
37	127.25373	37.34414	2	33	35				

- Our process

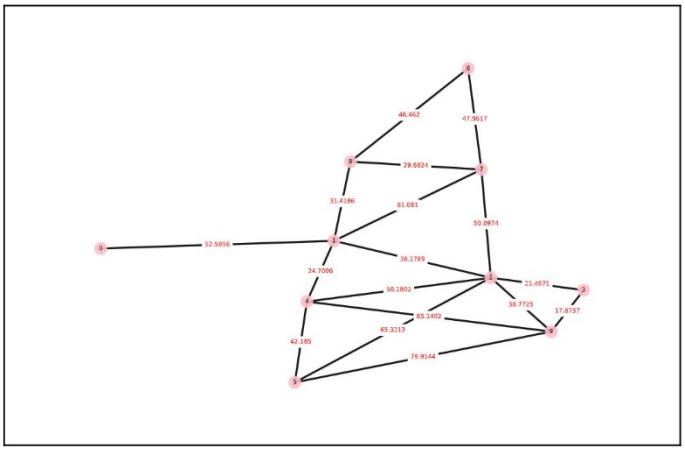
-> Mohyun map graph



in our data, there are some data which don't have priority value(average number of E-scooter).

So, we rebuild the Graph with the vertices which have priority values

-> Swing Station graph



- Our scenario

1. *We sorted out the vertices that managers have to go to in descending order from the large priority.*

(Sorting algorithm)

2. *Find shortest path between the vertices that managers have to go*

ex) If, managers have to go vertex #1(pri=3) and #2(pri=4)

1. route will be #0(start point) -> #2 -> #1

2. Find the shortest path between (#0, #2) &(#2, #1)

(Shortest Path algorithm)

D. Summary of the algorithms

D1. Data structure used in the project

- Graph
- Merge Sort
- Quick Sort
- Insertion Sort
- Dijkstra Algorthm
- Bellman Ford Algorithm

D2. Unique data structure used in the project

- *pandas*
- *networkx*
- *pyplot*

E. Algorithm Analysis

E1. Time Complexity with Big Oh Analysis

If we have V vertices and E edges. Our algorithms' time complexity are as follows.

- Dijkstra Algorithm + Insertion sort: O(ElogV)n^2*

*- Dijkstra Algorithm + Merge Sort: O(ElogV)*nlogn*

*- Dijkstra Algorithm + Quick Sort: O(ElogV)*nlogn*

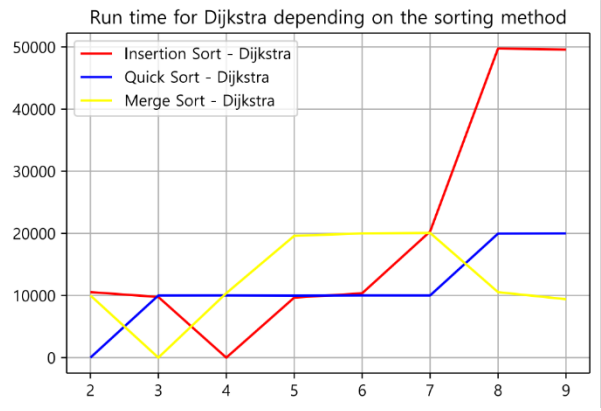
- Bellman Ford Algorithm + Insertion Sort: O(VE)n^2*

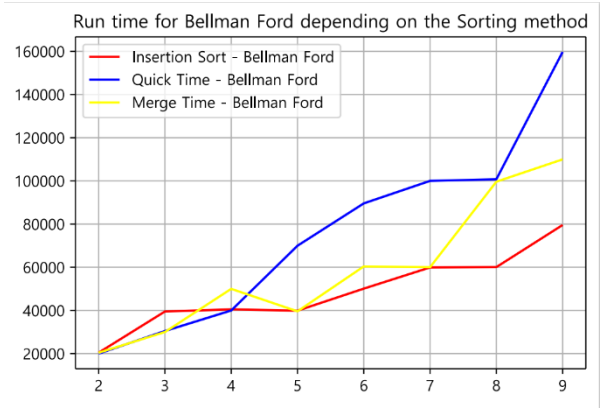
- Bellman Ford Algorithm + Merge Sort: O(VE) nlogn*

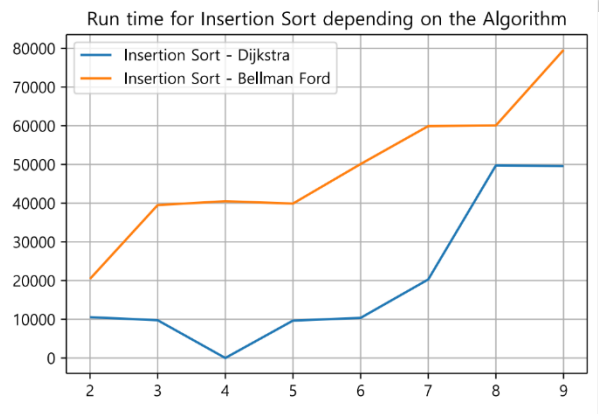
- Bellman Ford Algorithm + Quick Sort: O(VE) nlogn*

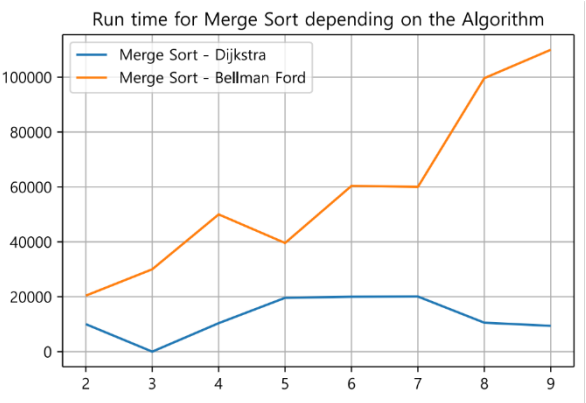
E2. Time Complexity from the experiments

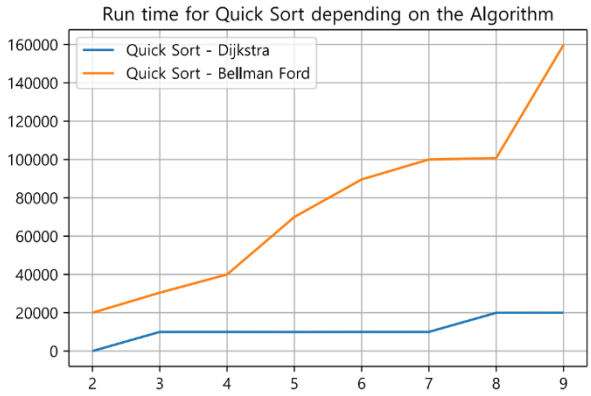
Our final result is as follows.











These five graphs show six algorithms' time performance with our own dataset.

[we multiplied the time with 10 power of 7. to make easy to see.]

< INPUT > We use data to calculate time analysis as shown in the table below.

#Data	Value (#Vertex)
n=2	(2,7)
n=3	(2,6,9)
n=4	(2,7,8,9)
n=5	(1,3,4,7,8)
n=6	(1,2,4,5,7,9)
n=7	(1,2,3,5,7,8,9)
n=8	(1,2,3,4,5,6,8,9)
n=9	(1,2,3,4,5,6,7,8,9)

└ The '#Data' in the table above referes to the number of nodes a manager must visit.

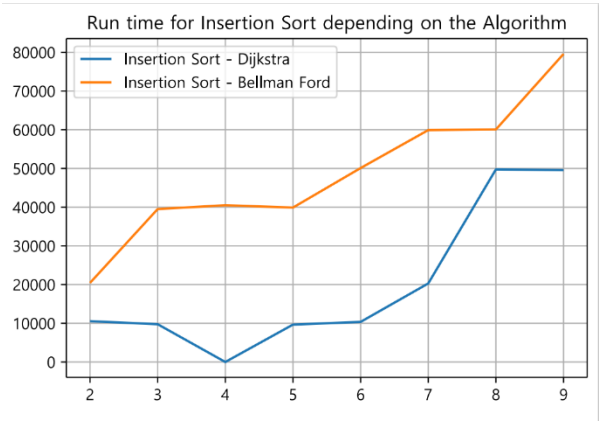
The reason why we showed the Excel graph instead of the Jupyter notebook is because of our scenario. We tried to use the Jupyter notebook and we used the random function to draw the graph because we received the deficient items as input function. But the graph's variation differs case by case, so we entered the value directly to represent the graph in Excel.

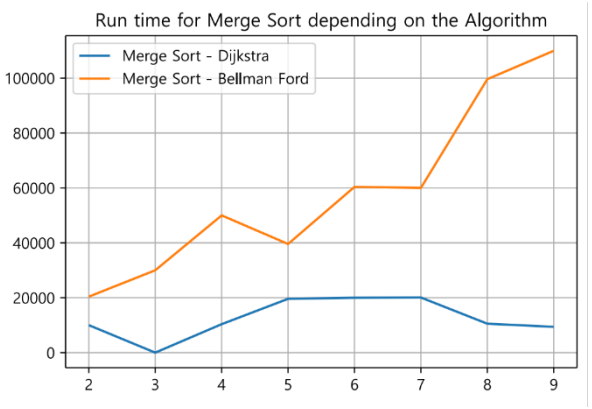
< OUTPUT >

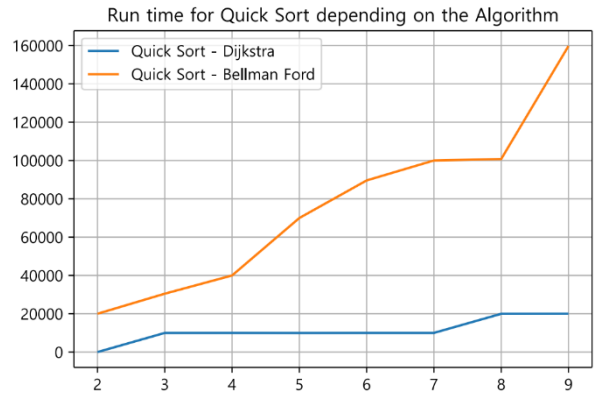
	2	3	4	5	6	7	8	9
distance	140.8619	325.6552	338.1018	325.6969	371.6981	528.5027	642.6652	715.6647

As a result of our process using the test data above, we obtain the shortest distances.

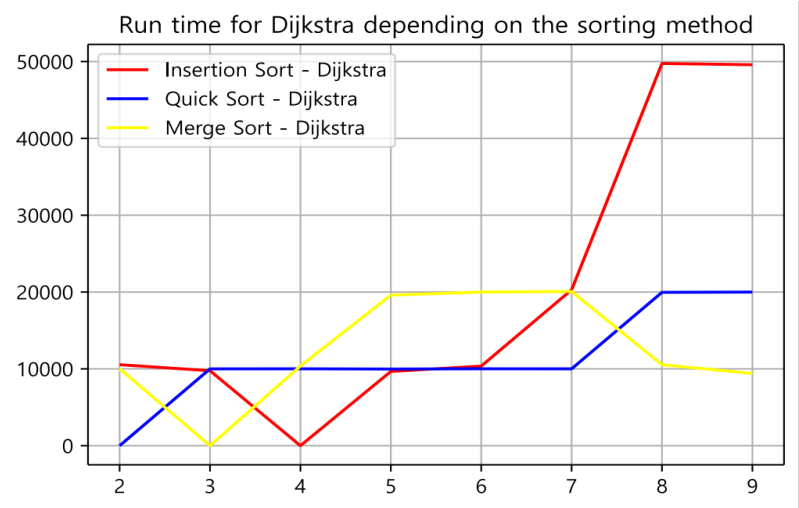
F. Conclusion







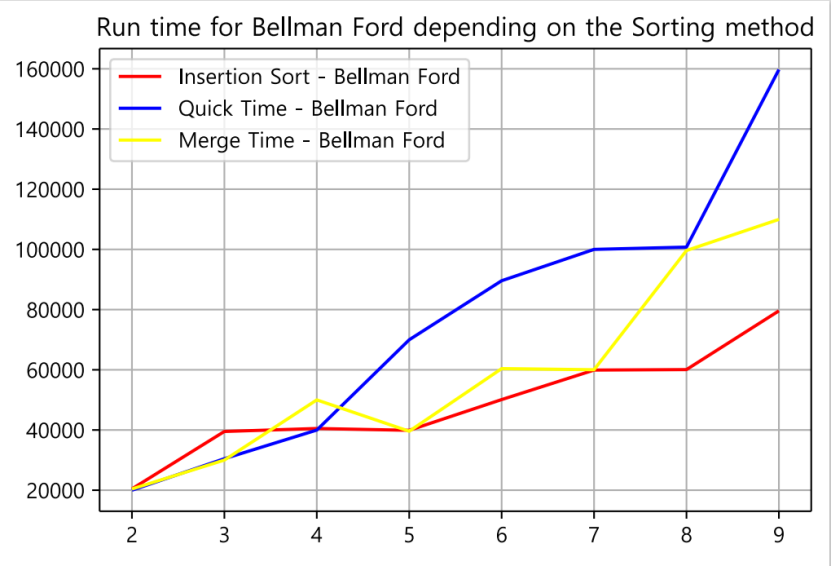
All these three graphs show that Dijkstra algorithm has a short running time.



When the number of data is small, the run time of insertion sort is equally small. However, when it is large, it can be seen to increase exponentially.

In our process, there was no significant difference in running time between Quick sort and Merge sort. However, when the number of data was nine, Merge sort was the fastest.

└Dijkstra + Merge sort was best in our project .



└In addition, we obtain the speed difference according to the Sorting algorithm in the Bellman ford algorithm as shown above

G. Dataset Explanation (if you use any specific dataset from internet)

We used our own dataset that we get the value from google map and Application "Swing"

google map -> longitude, latitude, number of

Swing -> number of intersection

H. Role of each members

202001368 박기윤 Do the code design, Make the code, Data Search/Preprocessing

202001733 서정민 Do the code design, Modify the code, Modeling

202000947 김주연 Do the code design, Modify the code, Modeling

201602843 임유비 Do the code design, Make the code, Data Search/Preprocessing

I. References (참고문헌)

https://ordo.tistory.com/87

https://github.com/burakguneli/sorting-algorithms-with-python/blob/master/allsorts.py

https://velog.io/@adorno10/%EC%B5%9C%EB%8B%A8%EA%B2%BD%EB%A1%9C-2-%EB%B2%A8%EB%A7%8C-%ED%8F%AC%EB%93%9CBellman-

Ford-%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98

https://miin-z.tistory.com/40

https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/