Assignment 2 – Experimentation Report

Introduction:

At first, A KD Tree is built, which is useful for two-dimensional location points. This KD Tree is the primary data structure I use for stage 1 and stage 2. Moreover, when it comes to a repeated location (the same site contains different companies), a linked list structure is used. In this way, there is no need to append the same place to the next node of the tree and potentially reduce its size.

In this experimentation report, we need to compare the number of key comparisons based on a query (redirecting a .txt file from the stdin in the command line) and test its accuracy. The instructors have provided us with different CSV files, including a sorted dataset, a random dataset, and a median dataset. We need to use these datasets to evaluate the effects (number of comparisons) on searching queried points using different datasets.

Stage one: (Nearest neighbour search)

A table shown below includes the input data used for the query when evaluating the effects on different datasets and their corresponding number of comparisons.

	X coordinate	Y coordinate	Random	Median	Sorted
1	144.854857	-37.8943775	**80**	**84**	**104**
2	144.863985	-37.8848754	**51**	**56**	**55**
3	144.873985	-37.8739754	**39**	**35**	**24**
4	144.880287	-37.8634987	**26**	**29**	**16**
5	144.893498	-37.8534897	**15**	**15**	**6**
6	144.903479	-37.8432701	**10**	**11**	**6**
7	144.911693	-37.8320826	**18**	**16**	**15**
8	144.923774	-37.8223973	**14**	**15**	**34**
9	144.933897	-37.8147437	**28**	**25**	**29**
10	144.943876	-37.8038744	**30**	**21**	**61**
11	144.951931	-37.7941038	**11**	**25**	**277**
12	144.964762	-37.7801873	**39**	**94**	**59**
13	144.973817	-37.7710835	**88**	**117**	**61**
14	144.983749	-37.7630847	**117**	**136**	**166**
15	144.993764	-37.7510932	**178**	**197**	**359**
16	145.001083	-37.7438616	**465**	**406**	**644**
17	145.013082	-37.7303764	**1112**	**1016**	**1224**
18	145.023498	-37.7238618	**1827**	**1809**	**2034**
19	145.033976	-37.7134093	**2627**	**2623**	**2791**
20	145.043982	-37.7030873	**3403**	**3421**	**3470**
21	145.053027	-37.6939873	**3760**	**3784**	**3750**

Table 1: different x, y coordinate used and its related No. of Comparisons on different datasets

Explanations of the data point:

- As the X coordinate range is from 144.900409 to 144.990552, and the range of Y coordinate is from -37.776234 to -37.849719, the range of the above data increases from the original coordinate range by 0.01 on both X and Y coordinate.
- The first five significant figures are manually set (e.g., X coordinate for the first five significant figures are from 144.85 to 145.05), and the rest are randomly assigned.

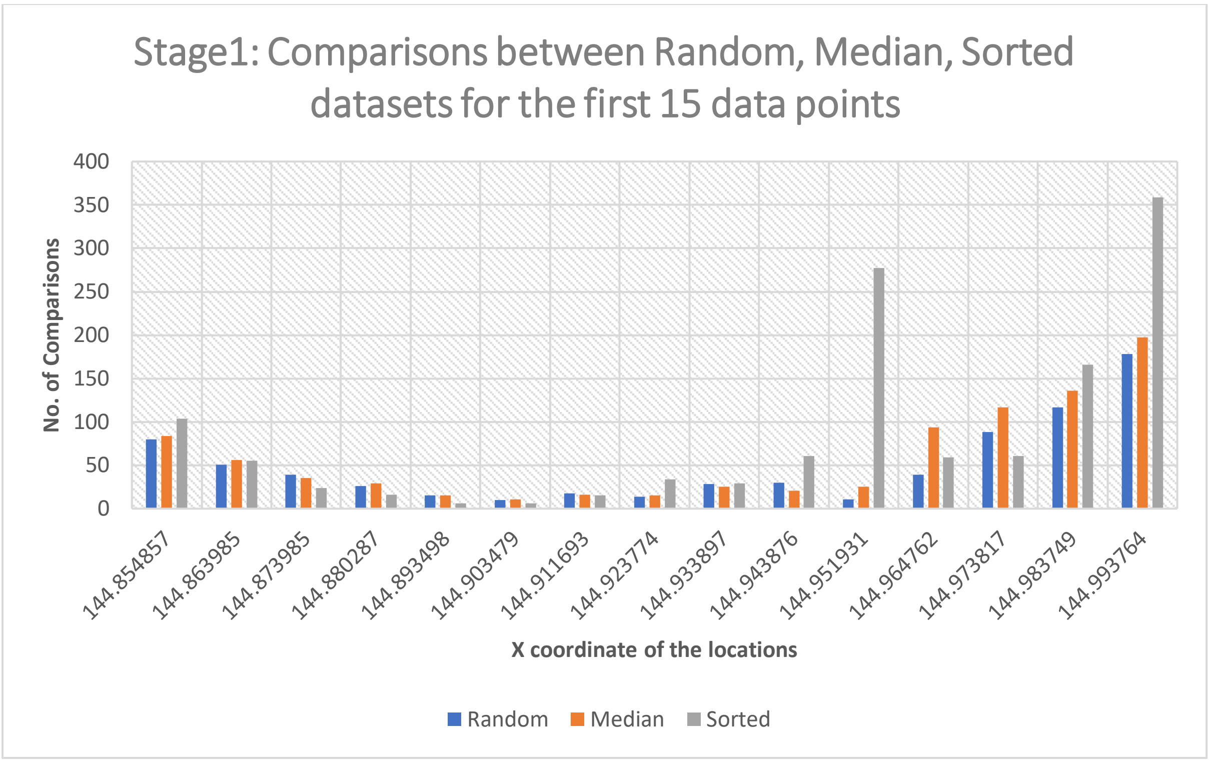

Figure 1: Comparisons between Random, Median, Sorted datasets for the first 15 data points

Similarities:

The theoretical average time complexity of the KD Tree is O($\log_2$n). There are 19117 records in the dataset, and the theoretical average comparisons are k*$\log_2$19117(where k is a constant), which is approximately 14*k for each search. These three datasets all perform well when the query data locations are within the range (the shaded green area in Table 1). However, when the query data locations are away from the range, the comparisons increase rapidly. The reason for this is that these queried data points can be considered as worst-case scenarios. The theoretical worst-case time complexity is O(n) because we have to traverse all the nodes in the KD Tree. In the worst-case scenario, When the queried points are more and more away from the dataset, it has to compare with more points so that the closest point could be found.

Differences:

Typically, a relatively balanced tree can be built using the median dataset because the first inserted record into the KD Tree is the median value of the whole dataset. When we found the median value, we can construct a balanced KD Tree since the tree's root node is the median value. Every data gets less than the root node will go left, and the other will go right. The KD Tree does well when median dataset's data points are within the bounds of x, y coordinate (shaded with green in the table). The comparisons are almost all less than 100.

A right skewed KD tree can be constructed using the sorted dataset. However, although it is right skewed, there may have some nodes on the left of the subtree because it compares with x or y coordinate only for each insertion. Thus, it is similar to using the median dataset and the number of comparisons shows that they are almost the same when searching for queried points if the query point is within the range (the green area). However, as soon as the point is outside the range, the number of comparisons starts increasing quickly.

Moreover, when comparing line 14, 15, and 16, the sorted dataset rises much faster than the random.csv and median.csv.

When using the "random.csv", the data have no order and are all randomly place in the KD Tree. It may have some chance that the root node can be 'median' enough, which results in O($\log_2 n$) theoretically as the "median.csv". The comparisons it needs to search have almost no difference from the median.csv. It also performs fewer comparisons than "sorted.csv" compared to line 14, 15 and 16.

Varying the size of the input csv:

Using UNIX "head -n" to create 1000, 2000, 3000, and 4000 rows from different csv files and compares with the ANC (Average Number of Comparisons).

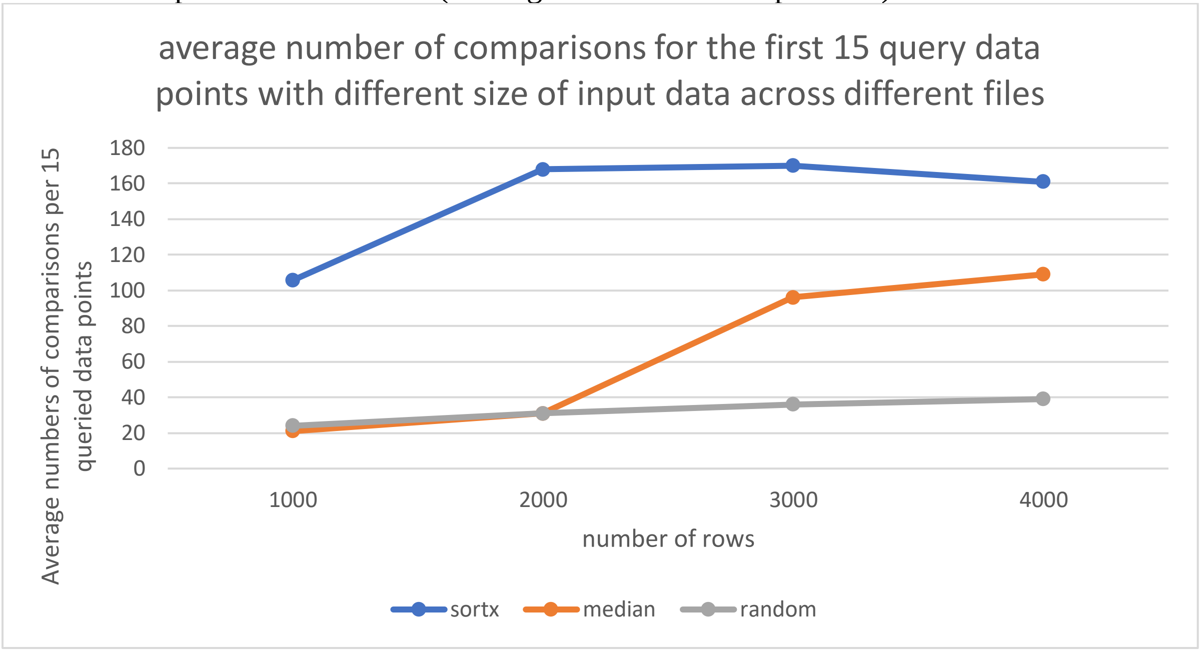

Figure 2:Average number of comparisons with different size of input data across different files

When the size of the input data file increases, it makes almost no difference to the number of comparisons when using the random csv file. The reason is that the points are randomly located in KD Tree, and although the size increases, some of the added locations may be useless. Thus, the ANC almost has no change.

However, when increasing the size of the median csv file, the ANC increase respectively. The reason is that when the size increases, there may have more points take into account, and the closest points to the queried points will change.

As for increasing the size of the sorted csv, the ANC first increase then decrease slightly. This reason for increase is the same as the median csv file. However, the decrease is not what I expected. I think one reason maybe that some of the query points may be founded first on the other end of the subtree as the size of the input csv increases, which results in less comparisons.

Stage two: (Radius search)

X coordinate	Y coordinate	radius	Random	Median	Sorted
144.951931	-37.794104	**0.0005**	**9**	**19**	**130**
144.951931	-37.794104	**0.005**	**129**	**82**	**516**
144.951931	-37.794104	**0.015**	**1665**	**1640**	**2025**
144.943876	-37.803874	**0.0005**	**41**	**21**	**59**
144.943876	-37.803874	**0.005**	**407**	**377**	**621**
144.943876	-37.803874	**0.015**	**1779**	**1697**	**2358**

Table 2: stage 2 radius search

When the searched radius is small, the time complexity of the radius search is $O(\log_2 n)$. As the data are shown above (green area), the time for searching one point with a small radius only takes a few comparisons, which matches the theoretical time complexity. Moreover, when it comes to comparing different datasets, the “median.csv” can have the best performance on average, among other datasets.

When the radius gets larger, the time complexity changes from $O(\log_2 n)$ to $O(n)$. As the data are shown above, when the radius increase, the number of comparisons increase rapidly. This behaviour matches the theoretical time complexity. One reason why the time complexity changes is that when the radius increases, more points are included, and we have to traverse more nodes in the KD Tree to find all the matched data.

Discussion and conclusion:

On average, the “median.csv” can do the best job among other datasets in both stages. The reason is that it can create a better KD Tree than others (a more balanced tree). When the tree gets more and more balance, our number of comparisons will decrease, and the time complexity will get closer to $O(\log_2 n)$.