COMP20003 - Algorithms and Data Structures

Assignment 2- Experimentation

Introduction

This experimentation aims to test the complexity of the implemented algorithms which are constructed by K-Dimensional Tree (x & y coordinate). The three different provided datasets (sort x, median and random) would be the independent variables which would affect the searching complexity in this experimentation. Since the key comparisons is inserted at each searching stage, it would count the number of steps through the searching process. Therefore, the resulted key counts would testify the expectation of the time complexity based on theory in each case. Furthermore, the tree traversal reveals that there are 4181 nodes in the tree. This experimentation would take 1000 samples to do the time complexity analysis. The proportion of the testing size to the datasets is nearly 25%.

Stage 1

Method:

1. Create the testing queries (x & y coordinate) by choosing random datasets from the original data file in the size of 1500 and saved the queries.

   cat median.csv | awk -F "," '{printf ("%s %s\n",$9,$10)}'| shuf -n 1500|uniq >> testClose0.txt

2. Sort the queries file, then delete not available datasets. (The above terminal command line extracts the data depending on comma. However, characters before x and y coordinates might contain extra commas, hence the copied line would be unavailable.)

   sort testClose0.txt >>testClose1.txt

3. Execute the program with different csv data files and record the comparison keys.

   valgrind ./map1 sortx.csv output.txt < testClose1.txt

   valgrind ./map1 median.csv output.txt < testClose1.txt

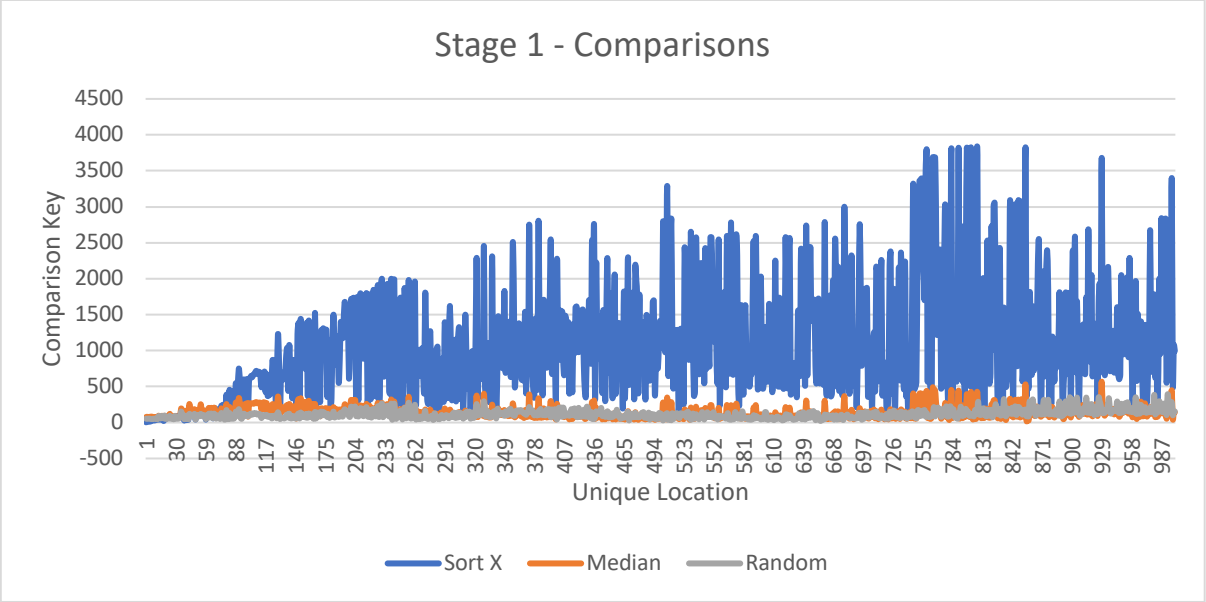
   valgrind ./map1 random.csv output.txt < testClose1.txt

4. Copy the comparison keys to excel.
5. Select 1000 datasets to calculate the average, median and stander deviation of the comparison keys.
6. Plot the counts of comparison keys with three different data files on each dataset.
7. Evaluate the counts of comparison keys to be cumulative counts and use it to plot the complexity graph.

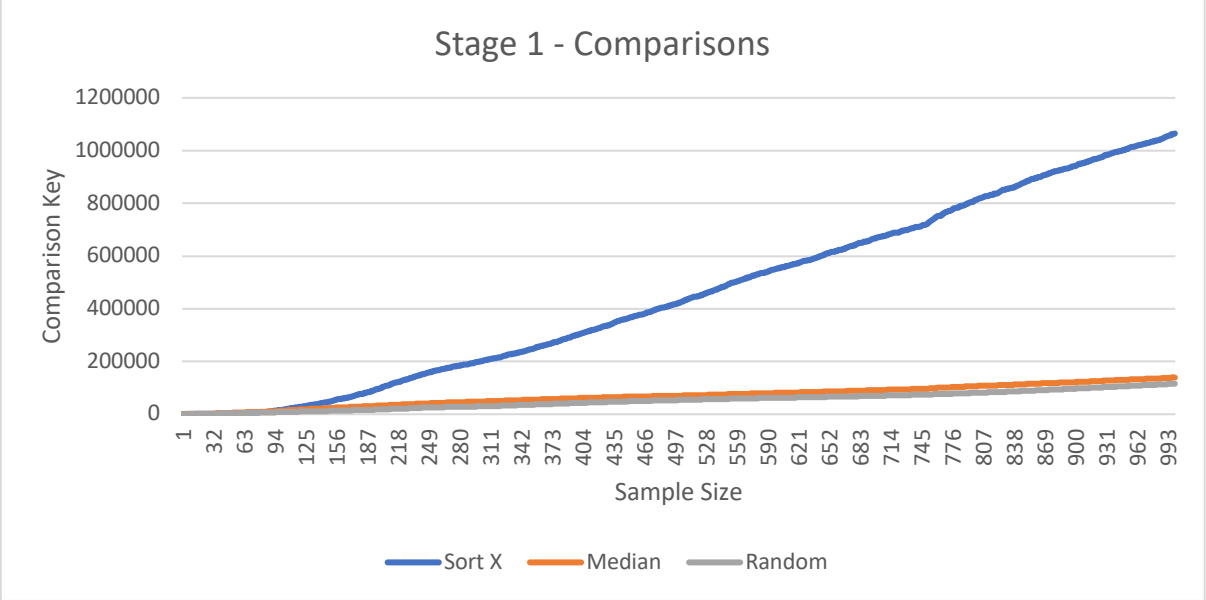
Data & Comparisons to Theory:

Stage1	Sort X	Median	Random
Mean	1064.931	139.133	115.828
Median	871.5	112	105
StDev	775.779568	79.5314942	57.3854817

(table 1)



(figure 1.1)



(figure 1.2)

Certainly, the “sort x” data file is the worst-case scenario. Since it is pre-sorted by x coordinate, the x coordinate of the input data would always be larger and larger, the tree would be unbalanced due to no node being inserted on the left side of the tree. It is expected even when sorted data is inserted into a binary search tree, a stick is created, and the binary search tree becomes essentially a one-dimensional linked list with O(n) searches since each child only has one child. Hence, the worst case of the single search would be m*O(n) (= 1000 *O(4181)). However, from figure 1.2, the sort x shows the time complexity of m samples is smaller than to m*O(4181). It is because the right side of the tree is not a complete stick, it still has some nodes which possess two branches due to the unsorted y-coordinates. On the other hand, median and random datasets are expected to have better performances since their trees are more balance. Based on theory, the average case of searching node would be $O(\log_2 n)$ which is $1000*O(\log_2 4181)$. Yet, in figure 1.2, the experimentation shows that the comparison keys of the median and random datasets are 10 times higher than expectation. The reason could be that the searching process not only proceeds down to one child, but also proceeds down to both branches sometimes, since the program aims to find the closest node. To some extent, it increases the time complexity of searching. The interesting finding is that using random datasets has less average time complexity comparing to using median datasets. It might be caused by the testing size, so if the sample size be increased to 50% of the datasets, the median might have a better time complexity as expected. Overall, the searching time complexity of sort x datasets is the worst and it has large deviation to its average complexity. The random and median datasets have the similar average time complexity for searching which is expected.

Stage2

Method:

1. Create the testing queries (x & y coordinate and radius) by choosing random datasets from the original data file in the size of 1500 and saved the queries.

   cat median.csv | awk -F "," '{printf ("%s %s %s\n",$9,$10,0.005)}'| shuf -n 1500|uniq >> testRadius0.txt

2. Sort the queries file, then delete not available datasets.

   sort testRadius0.txt >>testRadius1.txt

3. Execute the program with different csv data files and record the comparison keys.

   valgrind ./map2 sortx.csv output.txt < testRadius1.txt

   valgrind ./map2 median.csv output.txt < testRadius1.txt

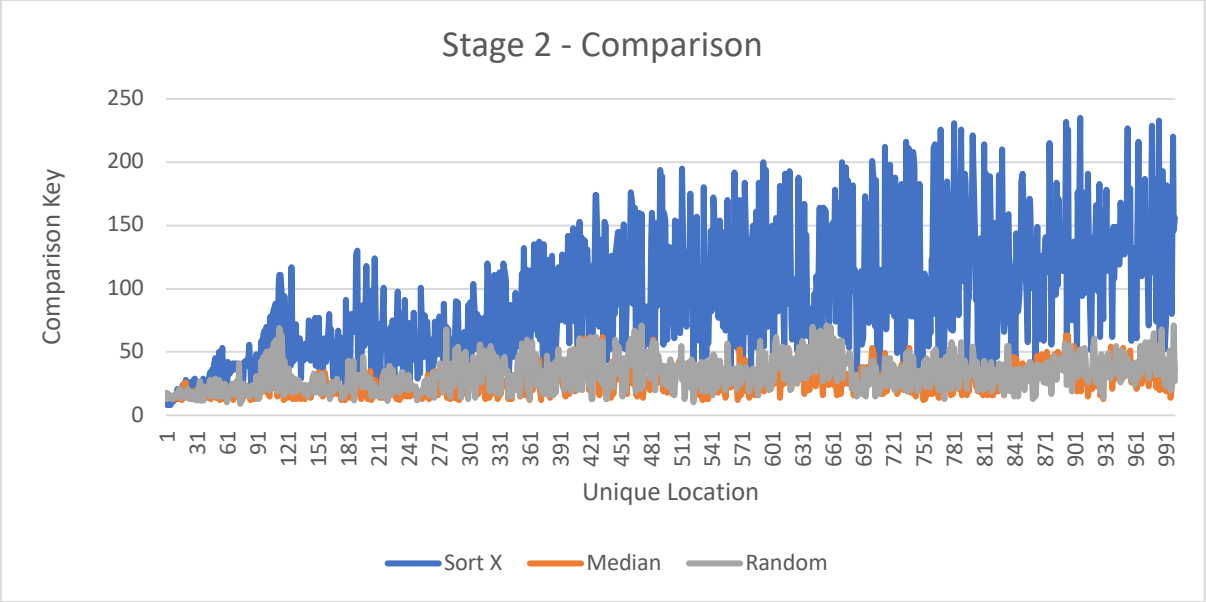
   valgrind ./map2 random.csv output.txt < testRadius1.txt

4. Copy the comparison keys to excel.
5. Select 1000 datasets to calculate the average, median and stander deviation of the comparison keys.
6. Plot the counts of comparison keys with three different data files on each dataset.
7. Evaluate the counts of comparison keys to be cumulative counts and use it to plot the time complexity graph.

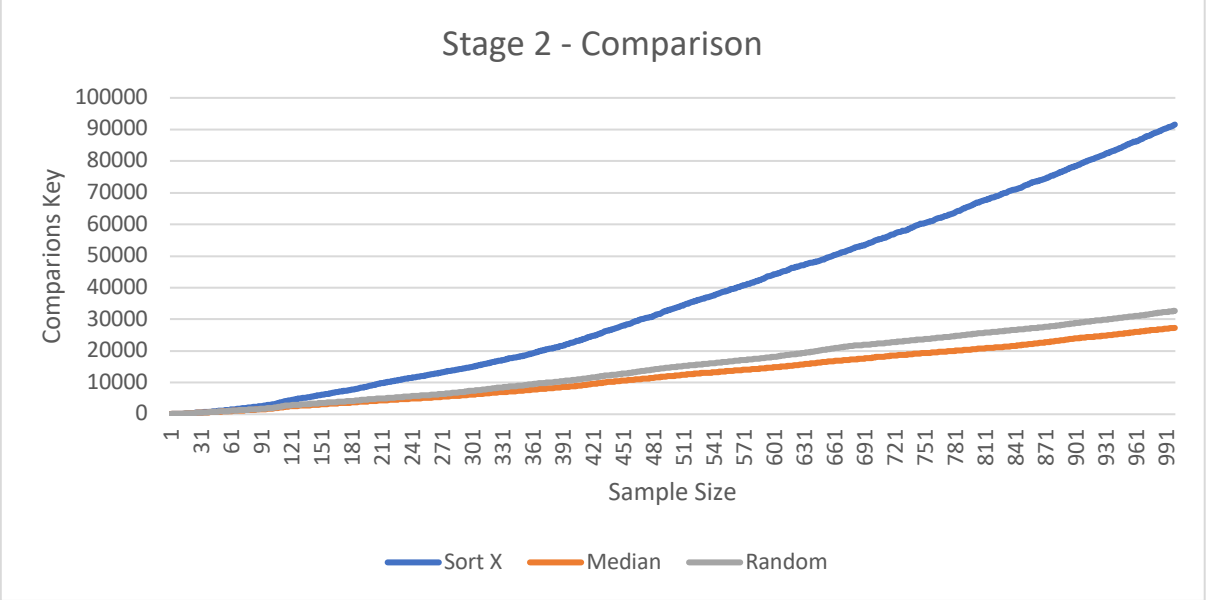
Data & Comparisons to Theory:

Stage2	Sort X	Median	Random
Mean	91.572	27.299	32.672
Median	82	26	31
StDev	49.61009956	10.75673307	13.02636181

(table 2)



(figure 2.1)



(figure 2.2)

In Stage 2 of the experimentation, the radius is set to be very small to result rational comparison keys. Otherwise, the searching would be meaningless, since more nodes would be searched, the comparison keys would not be able to show the difference between each dataset. The expected time complexity of Stage 2 is much less than Stage 1's, since the radius can lead the proceeding direction, and reduce the chances of proceeding down to both branches. Therefore, all comparison keys resulted by the three different datasets are much less than Stage 1's results. Also, it indicates that the results of comparisons keys are closer to the expectation based on theory comparing to Stage 1. Interestingly, the median has less time complexity than random in Stage 2.

Conclusion

Overall, the experimentation shows that the searching comparison keys are not entirely match to the expectation referred to theory. Since there are additional proceeding (both sides searching), it increases the count of comparison keys. The effect of the additional proceeding could be tested in Stage 2. When it is provided by a precise target (small radius), the performance of searching time complexity could be improved and be closer to the expectation. Generally, the sort x dataset has the worst-case scenario. Instead of the median dataset having less time complexity than the random dataset, those two datasets have similar time complexity and increasing the sample size could be a method to do the further clarification. Conclusively, the two stages of experimentation reveal that the more precise searching results closer outcome to the theoretical expectation and a sorted dataset would not perform well in a binary tree.