COMP20003 Algorithms and Data Structures
Assignment 2 – Experimentation
Aoi Fujii

Introduction

Varieties of datasets and key files were generated and number of comparisons were recorded to analyse the complexity of the 2d-tree algorithms for nearest neighbour search and radius search. There were datasets stored in sorted order, random order, median order, and datasets in different sizes from 1000 to 19000. These key-value pairs were created using UNIX commands and run with the program, ./map1 and ./map2. From each dataset, 100 randomly chosen keys were searched. The number of key comparisons were recorded for further analysis.

Big O notation shows the algorithmic behaviour for n close to infinity. Thus, algorithmic complexity should be analysed from the algorithm rather than a graph.

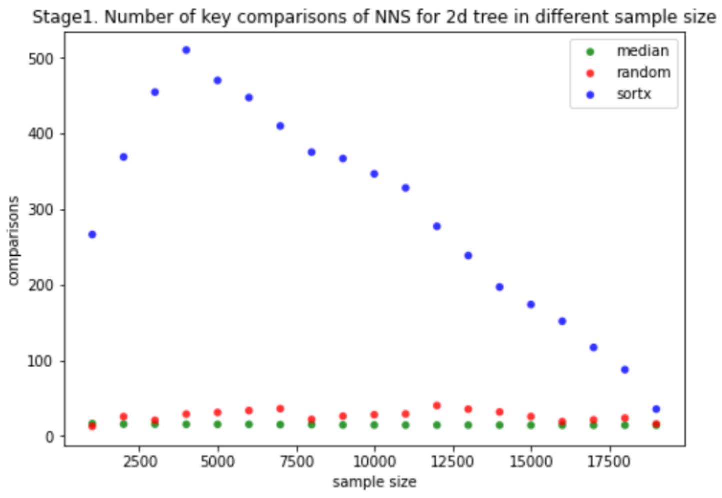
In addition, we must take into consideration that the sample size does not necessarily corresponds to the number of nodes as there are some duplicate values.

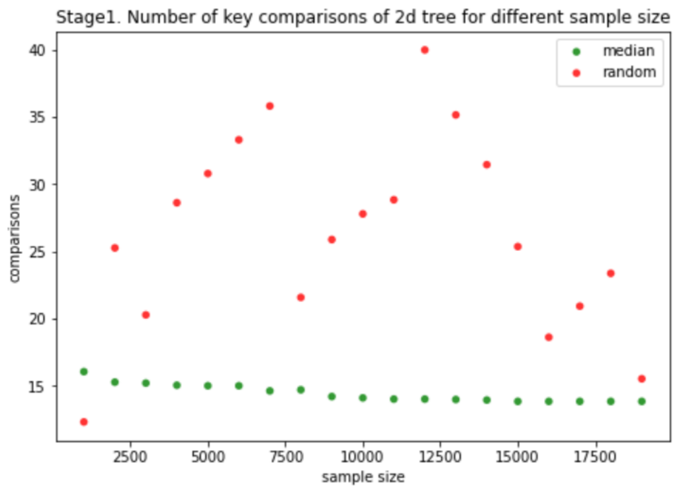
Experiment Method

1. For each of the datasets with different orders (sorted, median, random), create datasets of 19 varying sizes where n = [1000, 19000] with 1000 jumps. 100 keys are randomly chosen from the original datasets.
2. For map2, radius was set to constant value of 0.005 to minimise the effect of different factors other than sample size on the time complexity of the algorithm.
3. For each of the key value pairs, program was run and corresponding number of comparisons were recorded.
4. The average number of comparisons were calculated for each sample size for 3 different data types (sorted, median, random).
5. The number of comparisons was analysed and compared for each program, map1 and map2. The order of datasets and sample size were also analysed.

Stage1: Nearest Neighbour Search

This algorithm keeps track of the closest match from the query point. Each step compares a value in one dimension so the algorithm does not necessarily return the best match. If the distance between a node and a query node along axes is smaller than the current minimum distance, both branch are checked, otherwise it will check only one side. Nearest neighbour search has O(log(n)) average case time complexity and O(n) worst case even in balanced tree.





The number of key comparisons for datasets with different orders (sorted, median, random) are compared.

Sorted datasets

Sorted dataset is expected to create a linked list structure for the 2-d tree which will result in O(n) complexity (worst case). However, decreasing number of comparisons were observed after n > 4000. This could be caused by the key sampling method for the experiment. Since 100 keys were randomly chosen from the original datasets, the number of comparisons tend to decrease as the number of keys contained in the tree structures increase. They are more likely to be included in the tree and found much quicker than the non-existing keys. For further analysis, key values could be chosen differently to get more accurate results. For example, half of the key values could be taken from the sample and the other value could be randomly generated so that the values will not be biased. In addition, duplicate values might have also affected the lower number of comparisons relative to the sample size as same values are stored in the same nodes.

Median datasets

This dataset generates balanced tree since the values are inserted from median. The depth of the tree will be log(n). Thus, the time complexity for searching will be O(log(n)) complexity on average as it should traverse at least one leaf node. This 2-d tree gave relatively smaller number of comparisons compared to the one with sorted dataset.

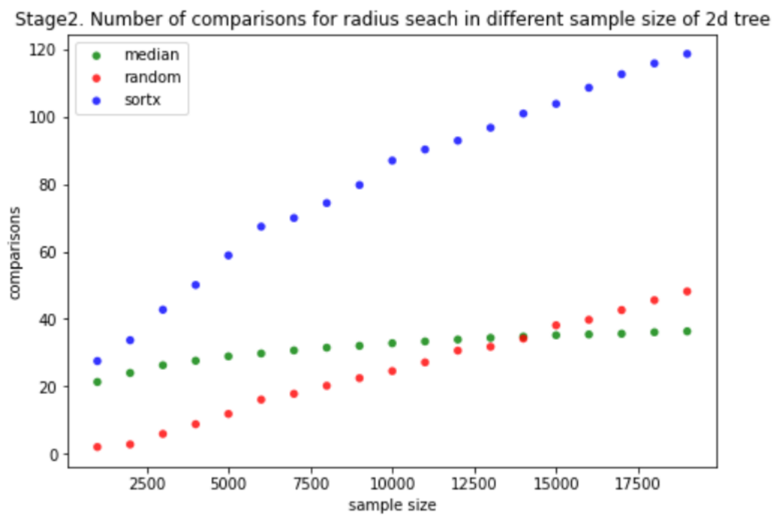
Random datasets

Similarly, random datasets created more balanced tree compared to the sorted datasets. However, the number of comparison does not necessarily depend on the sample size as the structure varies depending on the samples. The graph shows the randomness in the number of key comparisons for different sample size. This indicates that the complexity does not necessarily follow the certain trends but O(log(n)) on average. If the samples are close to the ordered form, it will create an unbalanced stick-like structure. In that case, the complexity will be closer to O(n).

For the median and random datasets, the number of comparison is more flat compared to log(n). This is because the sample size n is not necessarily equals to the number of nodes in the trees since duplicates exist in the tree.

Stage2: Radius Search

Radius search for this 2d tree algorithm check all the records within a given radius. If the distance between the key of the node and query along axes is larger than the radius, it will compare the right or left branch. Otherwise, it will search both branches until it reaches a leaf node. This algorithm has O(log(n)) best case and O(n) worst case.



Sorted datasets

This dataset creates a stick therefore gives the worst-case time complexity of O(n). From the graph, the number of comparisons for the tree created with the sorted dataset are increasing linearly (O(n)) with higher rate compared to other two tree structures.

Median datasets

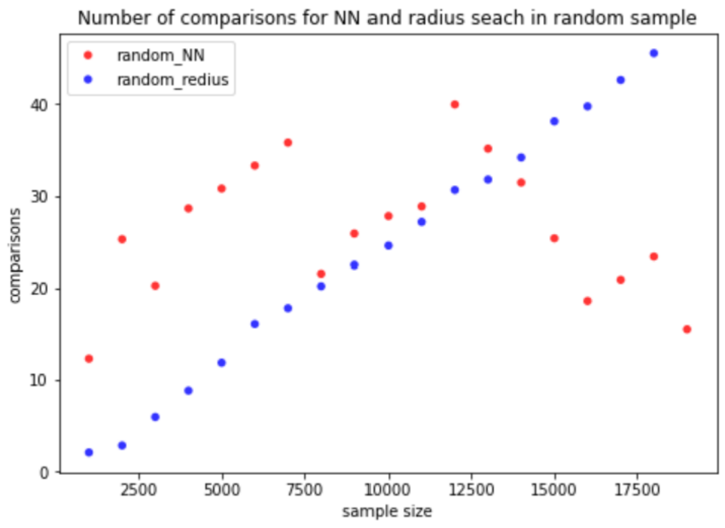
The complexity of this datasets is closer to log(n) best-case as seen from the graph. The number of comparisons does not increase significantly compared to other datasets even the sample size increases.

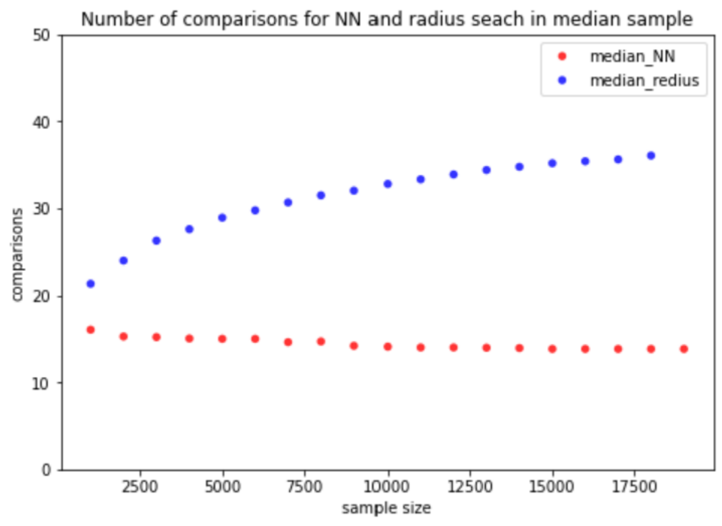
Random datasets

The number of comparisons of this dataset started off with small number which was more efficient than the trees with median datasets. However, number of comparisons increased linearly and exceeded the values of median datasets after n = 15000.

Since radius search looks for all the nodes within radius, balanced tree has significantly better time complexity as the sample size increase.

Comparison of NN and Radius Search





From the experiment, it was found that the time complexity for the radius search increases at a higher rate than the nearest neighbour search as the sample size n increased. A straight line could be fit to radius search indicating that it has a time complexity of O(n) for searching. On the other hand, nearest neighbour search is more fit to log(n) curve which is indicating O(log(n)) complexity. This is expected because radius search must find all the data within the radius while nearest neighbour should only find the nearest point. As a sample size increases, the number of nodes that must be searched for radius search will increase significantly compared to nearest neighbour search. For the best-case scenario, although both search algorithms have O(log(n)) complexity, nearest neighbour search has fewer number of comparisons as it needs to traverse at least 1 leaf node (O(log(n))) while radius search needs to traverse all the nodes (O(k*log(n)) where k=number of results).

Conclusion

It was found that radius search match with the theoretical complexities for searching. Its search on sorted datasets were completed in O(n) and search on median datasets were completed in O(log(n)). The nearest neighbour search did not match the theoretical model which has the average time complexity of O(log(n)) and O(n) for worst-case. This could be caused by the choice of query keys and duplicate values. Comparing two searching algorithms, radius search is more affected by the sample size than nearest neighbour search in terms of increasing time complexity.

Reference

Sorting and Assignment 2, Semester 2, 2020, Algorithms and Data Structures lecture slides