KD-Trees for Storage and Search

- Sanjog Gururaj (1099103)

Abstract

Implementation of a KD-Tree Data Structure for storage, traversal and searching. The Data Structure stores the location of different Business Entities around Melbourne. The program allows two types of search;

1. To find the closest Business Entity for a given location.
2. To find all the Entities within a radius for a given location.

The KD-Tree implemented is an unbalanced tree, thus results in varying complexity for different datasets:

1. For datasets which distributed on their median, insertion and search occur with a complexity of $O(\log n)$.
2. For datasets that are randomly distributed, insertion and search occur with a complexity of O(n), but with lower number of comparisons when compared to median distributed datasets.
3. For datasets that are sorted, we have a complexity of O(n), since the KD-Tree is not a balanced one. Thus, it results in a stick (singly linked list) with complexity of O(n).

Introduction

Binary Search Tree is a node-based binary tree data structure which has the following properties:

- The left subtree of a node contains only nodes with keys lesser than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.
- The left and right subtree each must also be a binary search tree.

A **KD-Tree** (also called as K-Dimensional Tree) is a binary search tree where data in each node is a K-Dimensional point in space. In short, it is a space partitioning data structure for organizing points in a K-Dimensional space.
K-D Trees are widely used in scenarios like Nearest Neighbour Search (Machine Learning), which makes them a very important Data Structure.

The objective is to evaluate the performance of KD-Trees on different types of data sets and assess its dexterity in real world cases.

The data set used is provided by City of Melbourne Open Data. The dataset used in this project is a subset of the Business establishment trading name and industry classification 2018 dataset, accessed from:
https://data.melbourne.vic.gov.au/Business/Business-establishment-trading-name-and-industry-c/vesm-c7r2

Performance Evaluation

1. Randomly Distributed Data sets:

After testing the program on 5 different datasets with, 2000, 5600, 9000, 15000, and 19117 records we find the number of comparisons (on average) to be:

- 12 for 2000 records
- 200 For 5600 records
- 574 for 9000 records
- 580 for 15000 records
- 584 for 19117 records

This displays O(logn) characteristics, but with some constants which can be ignored in large cases. Which is lower than the Median Distributed, this is consistent

2. Sorted Data sets:

After testing the program on 5 different datasets with, 2000, 5600, 9000, 15000, and 19117 records we find the number of comparisons (on average) to be:

- 700 for 1000 records
- 729 for 1300 records
- 2260 for 5000 records
- 5500 for 15000 records

It can be interpreted that this is has a complexity of O(n) since the tree is not balanced, thus the root always has only one child to the right or left which implies that the tree is a stick (singly linked list).

RESULTS:

Dataset	Average Complexity	Upper Limit
Random	Theta(logn)	O(logn)
Median	Theta(logn)	O(logn)
Sorted	Theta(n)	O(logn)

The second type of search has the same complexity as the first, since this search is a subset of the first.

CONCLUSION:

KD-Trees, if balanced, are the best Data Structures to store and search for higher dimensional data. They can achieve searching in O(logn) time. It is better to use a randomly sorted dataset if an unbalanced KD-Tree is used.