CSCI 3901

Software Development Concepts

Assignment 2

Problem 2

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

Submitted by:

Kshitijkumar Paraskumar Patel(B00942090)

October 5th, 2023

kd-Tree

Overview

We implement a data structure that can store the data in multidimensional order. For instance we can store the coordinates of a graph. The main purpose of choosing kd-Tree over BST is that it increases the randomness while dealing with multidimensional data.

In this implementations there are 2 ways to add a node to the tree. First is to add using *add()* method in which you can add a single point at a time. Second is to add using *build()* method where you can pass a Set of points to create a tree. Both of them have a major difference. The *build()* method will create the tree in a balanced manner whereas the tree built by the *add()* method will be unbalanced.

Files and External Data

The implementation divides into 3 classes and 2 interface:

- kdTree – Manages the overall tree and the actions supposed to be performed on the tree.
- Nodes – Hold the constructor for creating nodes of the tree.
- Points – All the services of the points that the nodes hold can be found in this class.
- Searchable – Interface which holds behaviour of all the tree related methods.
- TreeDebug – Interface that extends searchable and used for visualizing the tree.

Data structures and their relations to each other

Neither time nor space complexity was an issue but constraint was to not use any Data Structure from Java Collections Frameworks.

kdTree

- Stores the root node of the tree in form of a Node object.
- Dimension that all Points objects should have.

Node

- Stores the Point object
- Left child as a Node object and right child as a Node object

Point

- Stores the coordinates in a generic manner.

Assumptions

- When a build function is called another time we will overwrite the previous tree.

Choices

- Whenever we have an even number of points to choose from we will consider the smaller median.
- When there is a same x-coordinate and y-coordinate to compare, we will move to the left of the node.

Key algorithms and design elements

Sorting the points array

While building the tree we needed to sort the tree to find the median of that tree. We used bubble sort to deal with it as the time and space complexity were not a concern, bubble sort was easy to implement[1].

Find in range

We need to find all the points lying between minimum point and maximum point. We first find out the trueMin point and trueMax point as the minimum point and maximum point is not guaranteed to possess all the minimum coordinates.

We then traverse the tree and compare if the dimension across which the tree is split lies in between the trueMin and trueMax's corresponding coordinates. If the it is lower than trueMin's coordinate then we move to the left node and leave the right node as it is and vice versa.

Limitations

- The size method returns the size considering that there is no way to delete the entire sub-branch the function. If a method to delete or remove a branch is introduced then the size function won't work.

References

[1] GeeksforGeeks, "Bubble Sort - GeeksforGeeks," *GeeksforGeeks*, Oct 05, 2023. https://www.geeksforgeeks.org/bubble-sort/.
[2] "Level Order Binary Tree Traversal," *GeeksforGeeks*, Oct. 05, 2023. https://www.geeksforgeeks.org/level-order-tree-traversal/

[3] “Arrays.sort() in Java with examples,” *GeeksforGeeks*, Oct. 4, 2023. https://www.geeksforgeeks.org/arrays-sort-in-java-with-examples/.