CompSci 261P
Data Structures

Data Retrieval Efficiency Comparison among Binary Search Tree, AVL Tree and B- Tree

Name: Xingyou Ji
Date: 9 November 2019

1 Summary

This project will focus on the efficiency to retrieve records from binary search tree, AVL tree and B- tree. Specifically, I will implement binary search tree, AVL tree and B- tree, and perform data retrieval tests for all these tree trees to validate B- tree has the best performance.

2 Underlying Problem

Binary tree is a common and fundamental data structure to store data, and it supports $O(\log n)$ data retrieval in average case. However, if the binary search tree is unbalanced as shown in Fig 1, the worst case running time of data retrieval can go up to $O(n)$. Therefore, if the data size is sufficiently large, users cannot afford the worst case running time.

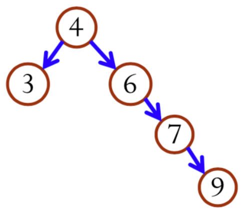

Figure 1: Unblanced binary search tree

AVL tree is designed to fix the above issue by enforcing the maximum difference between left and right subtrees to be no more than 1.

Modern databases, such as SQL, use B- tree to store data. The advantage of a B- tree is to reduce the number of nodes, so that fewer I/O processes are performed.

In this project, I will implement binary search tree, AVL tree as well as B- tree, and test their efficiency on data retrieval to see whether B- tree has the best performance.

3 Evaluation Method

I propose to use a counter to record how many nodes are visited when doing a retrieval operation. The reason why I choose to record the number of visited nodes rather than operation time is that the runnimg time depends on the complexity of the data structure. So, as the B- tree has a more complex structure, it could happen that it takes longer time to do the opertion. However, in real-life appications, the most time-consuming step is I/O, which is equivalent to the number of visited nodes in this project. Therefore, in order to validate that B- tree is indeed the best option for databases to use, it is wise to compare the number of visited nodes.

4 Proposed Test

I propose to generate 8 files, each containing $10, 50, 100, 500, 1000, 5000, 10000, 50000$ random data. Denote the file with size n as F_n. I will first store these n records into binary search tree, AVL tree and B- tree. Then, I will randomly generate an index i, and do a retrieval of record $F_n[i]$ for all three trees, and record the number of visited nodes in each tree.

In order to show the worst case of binary search tree, I will perform one additional test, that is, store 100 sorted elements in these three trees, and record their corresponding number of visited nodes.