DT183G - Datastrukturer och Algoritmer

Laboration 2 - Q4

MITTUNIVERSITETET

Institutionen för kommunikation, kvalitetsteknik och informationssystem (KKI) Östersund

Author	Samuel Thand	sath2102@student.miun.se
Supervisor	Raja-Khurram Shahzad	Raja-Khurram.Shahzad@miun.se
Examiner	Raja-Khurram Shahzad	Raja-Khurram.Shahzad@miun.se
Program	Programvaruteknik, 180HP	
Course	DT183G, Datastrukturer och Algoritmer	
Field	Computer Engineering	
Semester	VT 2023	

Table of Contents

A. Which is faster? AVL Tree vs BST **4**
B. How much faster? **4**
C. Is there an input for which binary search tree is faster? If yes, what is the reason? **4**

A. Which is faster? AVL Tree vs BST

The AVL tree often has faster searching, insertion, and deletion due to the self-balancing via rotations. A tree that has the property of **|B(n)| <= 1** where **B(n) = H(TL) - H(TR)**, is balanced - and this property is guaranteed by continuous balancing. A balanced tree like this has a time complexity of **O(log n)** because as the tree nodes increase, the tree's height increases roughly with **Log2(n).** Insertion, searching and deletion is all done via traversing the tree from root to the left or right child, by comparing their values until reaching the desired point, and the maximum of these comparisons that can be made is the height, which is **log n**.

B. How much faster?

The worst-case scenario for a BST is when it has a maximum imbalance. This leads to the tree's height being equal to the number of nodes **n**, and therefore a linear time complexity of **O(n)**. **O(log n)** time complexity is much faster than **O(n)**, especially as the size of the tree grows.

C. Is there an input for which binary search tree is faster? If yes, what is the reason?

Yes. If the data is somehow sorted and inserted so that the BST is always balanced, the BST can achieve **O(log n)** time complexity in addition to a reduced overhead from not having to do balancing operations. This would mean faster speeds.