MIT WORLD PEACE UNIVERSITY

Advanced Data Structures
Second Year B. Tech, Semester 4

THEORY ASSIGNMENT

CCA

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

April 25, 2023

Contents

1 AA Trees

AA Trees are a type of self-balancing binary search tree that are similar to red-black trees. They were invented by Arne Andersson in 1993, and their name stands for "Arne Andersson" tree. Like red-black trees, AA trees are designed to maintain balance by adjusting the structure of the tree as nodes are inserted or deleted.

The key difference between AA trees and red-black trees is the way that they maintain balance. In a red-black tree, nodes are colored red or black to ensure that the tree is balanced, while in an AA tree, nodes are given a level number. The levels of the tree are used to ensure that the tree is always balanced in a specific way, which makes the AA tree simpler to implement and easier to understand.

AA trees have the following properties:

1. Every node has a level number, which is a non-negative integer.
2. The level number of the right child of a node is always one less than the level number of the node.
3. A node can have at most one child with the same level number.
4. All leaves have a level number of 0. The path from the root to any leaf has the same number of nodes with level number 1.

Here's an example of an AA tree with the following keys: 10, 5, 12, 3, 7, 1, 9, 14, 11, 13, 15.

```
     7
   /   \
  3     12
 / \   / \
1   5 11 14
   / \   / \
  9   10 13 15
```

In this tree, each node has a level number assigned to it. The root node has level number 2, its children have level number 1, and their children have level number 0 (which makes them leaves).

The tree is balanced according to the AA tree properties, with all nodes having the correct level numbers and the path from the root to any leaf having the same number of nodes with level number 1. This ensures that the tree is always balanced in a specific way, making it easier to maintain and understand compared to other types of self-balancing binary search trees.

2 B+ Trees

B+ Trees are a type of self-balancing tree that are commonly used for indexing and file systems. They were invented by Rudolf Bayer and Edward McCreight in 1972, and their name stands for "Bayer-McCreight" tree. B+ Trees are similar to other types of self-balancing trees, but they have some unique properties that make them particularly useful for certain applications.

The key difference between B+ Trees and other types of trees is the way that they store data. In a B+ Tree, all data is stored in the leaf nodes of the tree, while the internal nodes of the tree only contain keys that are used to navigate the tree. This means that the leaf nodes are linked together in a linked list, which makes it easy to iterate over all the data in the tree in order.

B+ Trees have the following properties:

1. Every node has at most m children, where m is the order of the tree.
2. Every non-leaf node (except the root) has at least ceil(m/2) children.
3. The root has at least two children if it is not a leaf node.
4. All leaves appear in a linked list, sorted by their keys.
5. All internal nodes contain only keys, not data.

Here's an example of a B+ Tree with order 3 and the following keys: 10, 20, 30, 40, 50, 60, 70, 80, 90, 100.

```
         [40]
       /      \
   [20,30]   [60,70,80]
   /   |   \        |    \
[10] [20] [30]  [50] [60] [70,80] [90,100]
```

In this tree, the order is 3, which means that each node can have at most 3 children. The internal nodes of the tree contain only keys, while the data is stored in the leaf nodes, which are linked together in a linked list.

The tree is balanced according to the B+ Tree properties, with each non-leaf node having at least 2 children (except the root) and at most 3 children, and all the leaves appearing in a linked list sorted by their keys.

B+ Trees are particularly useful for indexing and file systems because they allow for efficient range queries and sequential access to data. The linked list of leaf nodes makes it easy to iterate over all the data in the tree in order, while the keys in the internal nodes allow for efficient navigation to specific ranges of data.