CSE 332: Data Structures and Parallelism

Section 4: Balanced Trees

0. The ABC's of AVL Trees

What are the constraints on the data types you can store in an AVL tree? When is an AVL tree preferred over another dictionary implementation, such as a HashMap?

1. Let's Plant an AVL Tree.

Insert 10, 4, 5, 8, 9, 6, 11, 3, 2, 1, 14 into an initially empty AVL Tree.

2. The ABC's of B-Trees

(a) What properties must a B-tree of n values have with given values for M and L?

(b) Give an example of a situation that would be a good job for a B-tree. Furthermore, are there any constraints on the data that B-trees can store?

3. Implement a B-Tree? Nah, Let's Analyze!

Given the following parameters for a B-Tree with a page size of 256 bytes:

- Key Size $= 8$ bytes
- Pointer Size $= 2$ bytes
- Data Size $= 14$ bytes per record (includes the key)

Assuming that M and L were chosen appropriately, what are M and L? Recall that M is defined as the maximum number of pointers in an internal node, and L is defined as the maximum number of values in a leaf node. Give a numeric answer and a short justification based on two equations using the parameter values above.

4. Oh, B-Trees

Find a tight upper bound on the *worst case runtime* of these operations on a B-tree. Your answers should be in terms of L, M, and n.

(a) Insert a key-value pair

(b) Look up the value of a key

(c) Delete a key-value pair

5. It's Fun to B-Trees!

(a) Insert the following into an empty B-Tree with $M = 3$ and $L = 3$: $3, 18, 14, 30, 32, 36, 15, 16, 12, 40, 45, 38$.

(b) Delete $45, 14, 15, 36, 32, 18, 38, 40, 12$