Before spring break notes

Prof Bill, Feb 2020

Before we head to the beach, I want to introduce balanced trees.

One page per tree. We'll go *deep* after Spring break.

The goal of balanced trees is to eliminate the destructive case where a BST turns into a linked list. In other words, we want the worst case performance of our balanced trees to be O(log n).

Each page starts with a Wikipedia link as an introduction. Then, I add some Bill-blather. Pages end with an animation link, so we can play. (yeah!)

Pages:

1. B-trees, 2-3-4 trees
2. Red-black trees
3. AVL trees
4. Scapegoat trees

thanks...yow, bill

1. B-trees, 2-3-4 trees

I like Wikipedia here, actually:

- ➔ B-tree, en.wikipedia.org/wiki/B-tree
- ➔ 2-3-4 tree, en.wikipedia.org/wiki/2%E2%80%933%E2%80%934_tree

B-trees

Important: “Not to be confused with Binary tree”

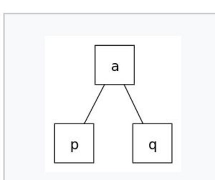
Definition:

> In computer science, a B-tree is **a self-balancing tree** data structure that maintains sorted data and allows searches, sequential access, insertions, and deletions in **logarithmic time**. The B-tree generalizes the binary search tree, allowing for **nodes with more than two children**. Unlike other self-balancing binary search trees, the B-tree is well suited for storage systems that read and write relatively large blocks of data, such as discs. It is commonly used in databases and file systems.

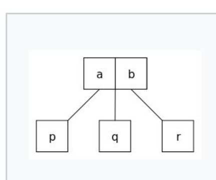
2-3-4 trees

We’ll focus on 2-3-4 trees, which are one flavor of B-tree. Definition:

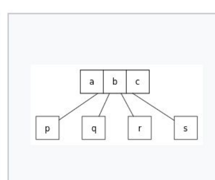
> In computer science, a 2–3–4 tree (also called a 2–4 tree) is a self-balancing data structure...where every node with children (internal node) has either two, three, or four child nodes:
>
> - a 2-node has one data element, and if internal has two child nodes
> - a 3-node has two data elements, and if internal has three child nodes
> - a 4-node has three data elements, and if internal has four child nodes



2-node



3-node



4-node

Properties:

- ➢ Every node (leaf or internal) is a 2-node, 3-node or a 4-node, and holds one, two, or three data elements, respectively.
- ➢ All leaves are at the same depth (the bottom level).
- ➢ All data is kept in sorted order.

Since our tree is always balanced, then search, insert, and delete are all **O(log n)**!

2–3–4 trees are an isometry of red–black trees, meaning that they are equivalent data structures. Keep reading...

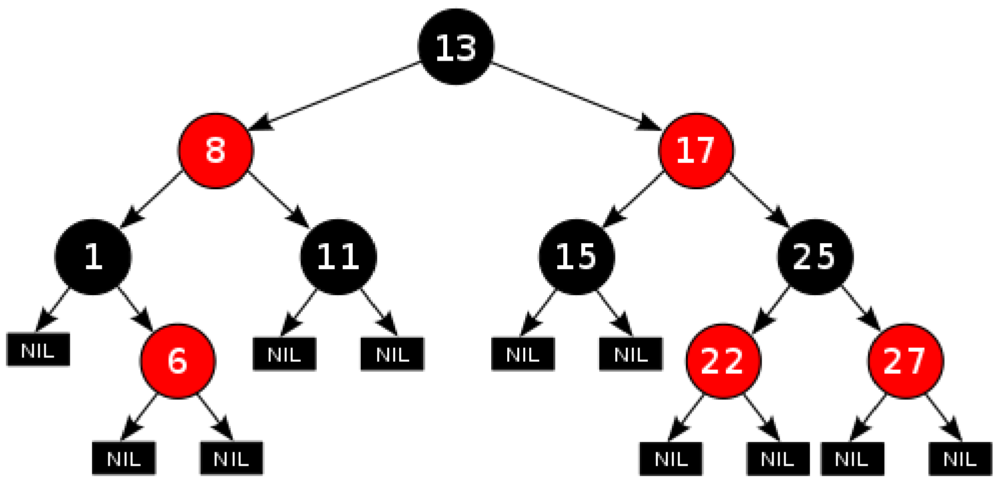
Maybe the best animation of all (select “B Tree”):
www.cs.usfca.edu/~galles/visualization/Algorithms.html

2. Red-black trees

Wikipedia goodness: en.wikipedia.org/wiki/Red%E2%80%93black_tree

The red-black rules are:

- ❏ Each node is either red or black.
- ❏ The root is black.
- ❏ All leaves (NIL) are black.
- ❏ If a node is red, then both its children are black.
- ❏ Every path from a given node to any of its descendant NIL nodes contains the same number of black nodes.



Great animation (select "Red-Black Trees"):
www.cs.usfca.edu/~galles/visualization/Algorithms.html

3. AVL trees

Wikipedia: https://en.wikipedia.org/wiki/AVL_tree

Definition.

> In computer science, an AVL tree (named after inventors Adelson-Velsky and Landis) is a self-balancing binary search tree. It was the first such data structure to be invented.[2] In an AVL tree, the heights of the two child subtrees of any node differ by at most one; if at any time they differ by more than one, rebalancing is done to restore this property.

AVL is notoriously complex/difficult to code. You can see...Wikipedia hides the huge code chunks.

We'll learn the various rotations to make AVL work.

This is an OK start. (hint: we will do better)
www.tutorialspoint.com/data_structures_algorithms/avl_tree_algorithm.htm

4. Scapegoat trees

Wikipedia: en.wikipedia.org/wiki/Scapegoat_tree

Read Morin 8 Scapegoat trees: opendatastructures.org/ods-java/8_Scapegoat_Trees.html

Quick facts: 1) The Wikipedia page for scapegoat trees is weak(ipedia), 2) There is no animation for scapegoat trees, and 3) I've never used them.

/* What do these facts tell you? */