Tutorial 9: Red/Black Trees

ELEC 278: Fundamentals of Information Structures

The learning goals for Tutorial 9 are:

- Gain familiarity with another type of self-balancing trees: red/black trees.

Red-black trees are another type of binary search trees. Like the AVL tree, a red-black tree has self-balancing properties. The structure of a red-black tree follows certain rules to ensure that the tree is always balanced.

Properties of a red-black tree:

1. Each tree node is colored either red or black.
2. The **root node** of the tree is **always black**.
3. **Every path** from the root to any of the leaf nodes must have the **same number of black nodes**.
4. **Red property:** No two red nodes can be adjacent, i.e., a red node cannot be the parent or the child of another red node.

Red/black trees have less strict balancing rules than AVL trees, i.e., the balance factor at the nodes can exceed 1. As such, they are less costly than AVL trees if you are doing frequent insertions or deletions from the BST as less rotations will be required overall. Hence, red/black trees might be better than AVL trees for frequent insertions/deletions, while AVL trees might be better for frequent searches.

The rotations in red/black trees are not determined by the node's balance factor, but rather its color.

Problem 1. Visit `https://www.cs.usfca.edu/~galles/visualization/RedBlack.html` which visualizes operations on red/black trees. Read the insertion and deletion steps below and validate your understanding using the visualization. Compare to AVL trees.

Insertion

- **Step 1:** Insert the new node as a red node.
- **Step 2:** If the new node is the **root**, recolor it to black.

- **Step 3:** If the **parent** of the new node is black, the tree remains valid.
- **Step 4:** If the **parent** of the new node is red, we have a violation:
  - If the **uncle** is red, recolor the **parent**, **uncle**, and **grandparent**.
  - If the **uncle** is black, rotate and/or recolor to fix the tree.

Deletion

- **Step 1:** Find the node to be deleted.
- **Step 2:** Replace the node with its **successor** (if two children).
- **Step 3:** If the **deleted node** or its **successor** is red, just remove it.
- **Step 4:** If the **deleted node** and its **successor** are black, fix the double black issue:
  - If the **sibling** is black and has a red child, rotate and recolor.
  - If the **sibling** is black and has two black children, recolor and move the problem up the tree.
  - If the **sibling** is red, rotate and recolor to make the sibling black.