• Which of the following are not guaranteed to be “balanced”?
  A. AVL Tree
  B. Red-black Tree
  C. Heap
  D. They are all balanced
  E. Whatever
Administrative Details

• Lab 9 Today: Gardner’s Hex-a-Pawn
  • Another partner lab!
  • Challenging to design & debug
  • Design doc is worth 2 points of your lab grade
Today’s Outline

• Balanced Binary Search Trees
  • AVL Tree
  • Red-Black Tree
  • (Splay Tree)
• Game Trees
Splay Trees

Splay trees are self-adjusting binary trees

• Each time a node is accessed, it is moved to root position via rotations
• No guarantee of balance (or shallow height)
• But good *amortized* performance

Theorem: Any set of m operations (add, remove, contains, get) on an n-node splay tree take at most $O(m \log n)$ time.
Splay Tree Rotations

Right Zig Rotation (left version too)

Right Zig-Zig Rotation (left version too)

Right Zig-Zag Rotation (left version too)
AVL Tree Facts

- A tree that is AVL except at root, where root balance factor equals $\pm 2$ can be rebalanced with at most 2 rotations.
- `add(v)` requires at most $O(\log n)$ balance factor changes and one (single or double) rotation to restore AVL structure.
- `remove(v)` requires at most $O(\log n)$ balance factor changes and $O(\log n)$ (single or double) rotations to restore AVL structure.
- An AVL tree on $n$ nodes has height $O(\log n)$.
Red-Black Tree Facts

- The coloring rules lead to the following result:
  - No leaf has depth more than twice that of any other leaf.
  - A Red-Black tree with $n$ nodes has height $O(\log n)$.
Single Rotation (Right)
Single Rotation (Right)

// pre: this has a left subtree
// post: rotates local portion of tree so left child is root
protected void rotateRight() {
    BinaryTree<E> p = parent;
    boolean wasLeftChild = isLeftChild();
    
    BinaryTree<E> newRoot = left;
    
    setLeft(newRoot.right); // 0
    newRoot.setRight(this);  // 0

    if (parent != null) {
        if (isLeftChild())
            parent.setLeft(newRoot); // 2
        else
            parent.setRight(newRoot);
    }

    newRoot.setParent(p);
    p.setLeft(newRoot);
    left.setParent(this);
Today’s Outline

- Balanced Binary Search Trees
  - AVL Tree
  - Red-Black Tree
  - (Splay Tree)
- Game Trees
Game Trees
Game Trees

- Nodes are positions in a game (game state)
- Edges are moves (transition from one game state to another)
  - All nodes at a given level represent moves by the same player
- Leaf nodes represent ending board states (winner or tie)
  - # of leaf nodes = # of ways a game can be played
In AI, often search the game tree and use an algorithm like **minimax** to choose the next “best move”

- Chess, checkers, Go, etc.
Minimax Example

The diagram depicts a minimax tree with decision points labeled as 'max' (for max player) and 'min' (for min player). The tree structure shows the game's decision points and the corresponding outcomes, with values indicating the utility or score for each decision path.
Modern Game AI
Hexapawn