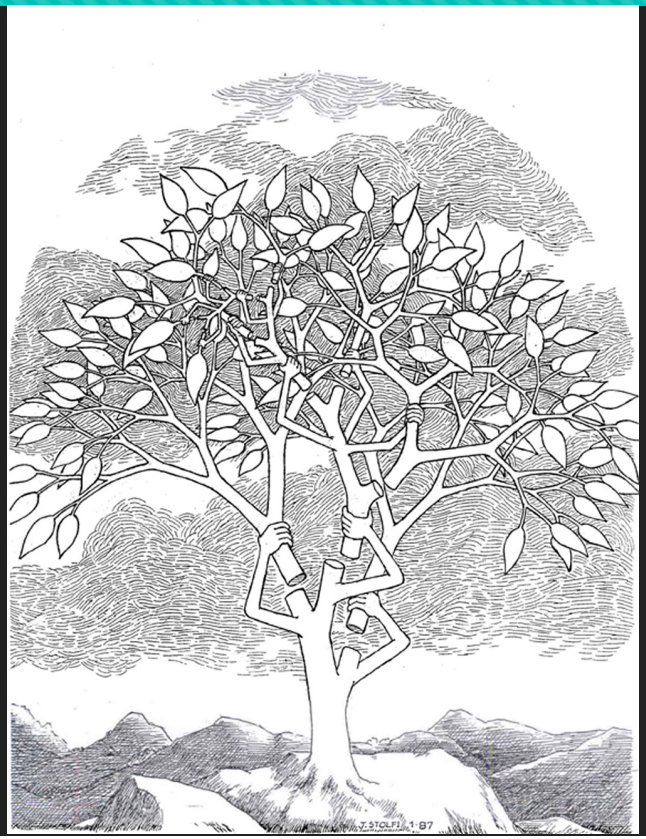
Balance

cs136

Review

- A **balanced** binary search tree implements all core operations in O(lg n)
- Binary search trees are typically used as **maps** (Dictionaries)
  - Ordered (comparable) **key**
  - Used to reference a **value**
- BST insert, contains, get, and remove have common code…**locate**() helper
- Depth-first in-order iteration gives the values in sorted order!
- **AVL** and **Red-Black Trees** auto-balance
- **Splay Trees** don't maintain balance…but still give expected, amortized O(lg n)

Splay Tree Demonstration



Self-adjusting binary search trees
Sleator and Tarjan
Journal of the ACM, 1985

https://github.com/morgan3d/misc/tree/master/splaytree

Implementing Balance()

- Depth-first in-order iteration gives the values in sorted order
- Inserting in "binary search order" from a sorted array gives a balanced BST
- Strategy:
  - Extract sorted array of elements from an arbitrary BST
  - Clear the tree
  - Insert via "depth first" iteration on the *array*
- Warmup:
  - Look at print()
  - Implement printSorted()

Summary

- Easy to implement code to balance a BST!
- Recursion is beautiful
- Exciting conclusion on Monday: BST.remove()