Trees!

- non-linear, hierarchical data structures
- lots of associated vocabulary
- node
- edge
- root
- siblings
- descendants, height 5
- ancestors, depth 5
- internal nodes
- Path BACDG
- E’s parent
- E’s child
- D’s subtree
- leaves/external nodes
Abstract Tree

• `tree.py`
• main job – provide interface for concrete trees
• Recursion used extensively
  • Base case: a particula node
  • Otherwise recurse on children or parent, depending on particular algorithm
• `depth()`
• `_height1()` vs `_height2()`
Binary Trees

• 0, 1, or 2 children/node only
• *proper/full* – every parent has 2 children
• *improper* – not every parent has 2 children
• Recursive definition:
  • base case: tree is empty
  • otherwise: node $r$ has left and right binary subtrees
    • (note: these are possibly empty subtrees)
Arithmetic

- Infix binary tree

\[((5 + 9) \times 7) / ((13 - 7) + 2)) - ((8 \times (2 - 7)) + 6)\]

\[5 + 7 \times 13 - 2 + (2 - 8) \times 6 + -\]
Binary Tree Properties

- For both proper and improper (h = height)
  - $h + 1 \leq n \leq 2^{h+1} - 1$
  - $1 \leq \text{leaves} \leq 2^h$
  - $h \leq \text{non-leaves} \leq 2^h - 1$
  - $\log(n + 1) - 1 \leq h \leq n - 1$

- For just proper
  - $2^h + 1 \leq n \leq 2^{h+1} - 1$
  - $h + 1 \leq \text{leaves} \leq 2^h$
  - $h \leq \text{non-leaves} \leq 2^h - 1$
  - $\log(n+1) - 1 \leq \text{height} \leq (n-1)/2$
  - $\text{non-leaves} + 1 = \text{leaves}$
Binary tree implementations

• `simpletree.py`
  • just to get the idea down

• `binary_tree.py`
  • more complete, uses `Tree.Position` objects

• `linked_binary_tree.py`
  • see complexity chart in notes

• see array-based binary tree example in notes