Trees

Tuesday, February 20, 2018 1:15 PM

Tree Structure

- Nodes and edges
- Relationships: parent, children and siblings
  - Ancestor and descendants
  - Every node (except root) has one parent
  - Leaf node = no children (non-leaf = internal node)
- Root = level 1, height = max level
- Size = #nodes
- Recursive structure (branches are subtrees)

Applications

- Represent hierarchical data
- File systems, organisational charts
- arithmetic expressions (leaf nodes = operands, internal nodes = operators)
- XML documents: labelled trees

Binary Tree

- Each node has at most 2 ordered children
- Usually for linear data structures (e.g. ordered data)
- Logarithmic growth of height
- Full binary tree = all nodes (at level < height) have 2 children
  - Height H = 2^h - 1 nodes, N nodes = log (N + 1) height
- Complete binary tree: full until level h-1

Implementation

- Reference-based
  - TreeNode (similar to ListNode), root
- Array-based
  - Stores index of elements
  - Chain free space, store reference in left/ right (-1 = null)
  - Usually for complete trees (regular representation)

Tree Traversal

- How to iterate through every node?
- Post-order (root last)
- Pre-order (root first)
- In-order (root between)
  - Can be used for sorting
- *Given pre-order and post-order, need to state that nodes cannot have 1 child
- *From pre-order, insert in same way to get back BST (only for BST as it is sorted)
- Level-order (BFS)

Binary Search Tree

- Smaller = left, larger = right
- "sorted" binary tree
- No duplicate key values
- Insert, delete and search in O(H)

    - ~ O (log n) time (depends on balance/ skew)
    - Balanced tree: AVL

Delete

- 0, 1 or 2 children?
- In-order predecessor = value immediately before it (vs. successor)
    - Move to node position