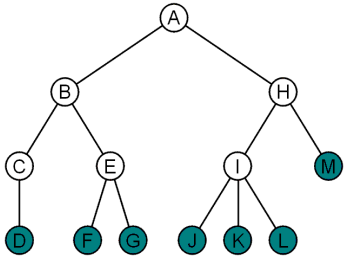
Discussion 4: Tree

CS101 Fall 2024

CS101 Course Team

Oct 2024

Tree



- root node: depth=0
- leaf node, internal node
- degree of a node: Num. of children
- depth of a node, height of a tree

Tree traversal

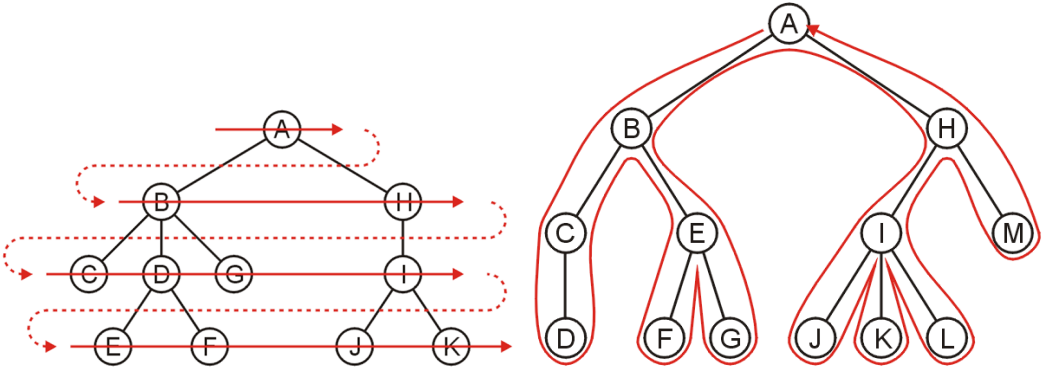
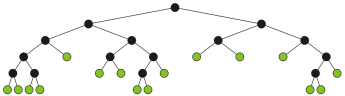
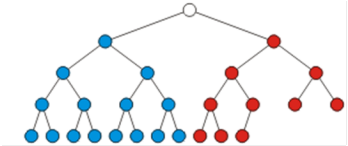
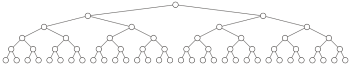


Figure: bfs v.s. dfs

Complexity: $\Theta(n)$ run time and O(n) memory

binary tree

- Binary Tree: at most two child. Array represention: for node i, left son$2i$, right son $2i+1$
- Full Binary Tree: Every node except the leaf nodes have two children.
- Complete Binary Tree: Every level except the last level is completely filled and all the nodes are left justified.
- Perfect Binary Tree: Every node except the leaf nodes have two children and every level (last level too) is completely filled.

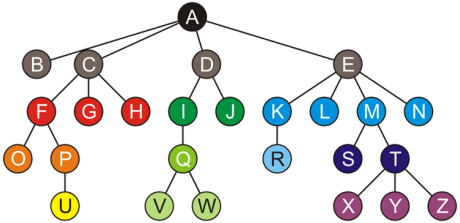




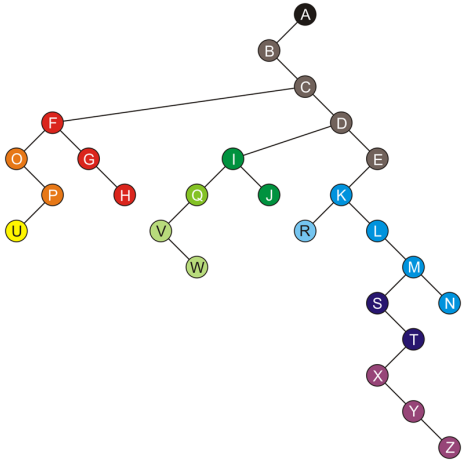
binary tree traversal

- in-order traversal (Left, Root, Right)
- pre-order traversal (Root, Left, Right)
- post-order traversal (Left, Right, Root)
- Breadth-First or Level Order Traversal

Left-child right-sibling

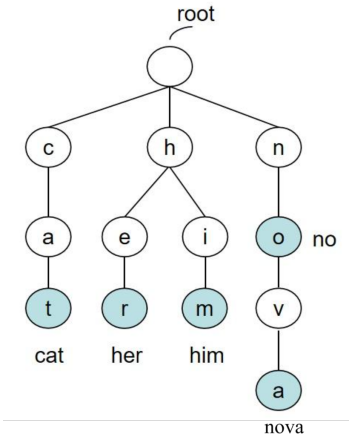
Left: first child; Right: other children





Trie*

re**trie**val / Prefix Tree



Trie*

- How to determine whether a word exists in the paragraph?
- As we learned: Hash.
- Basic thought: Space for Time
- insertion, delete, find operations
- advantages: no collision(compared with hash)
- disadvantages: the space consumption is relatively high

Trie*

insertion, delete, find operations

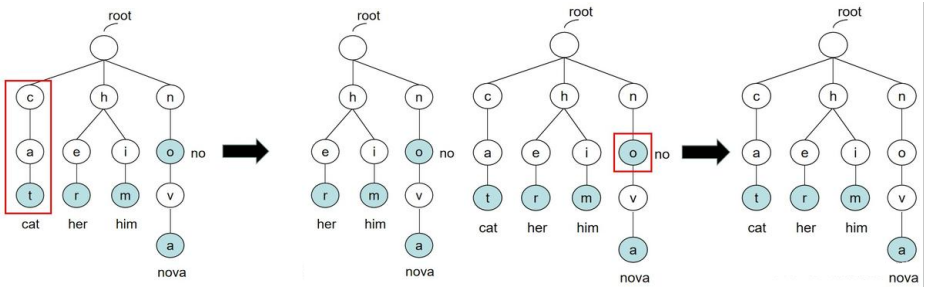


Figure: delete operations

Trie*

Usage of Trie:

- String retrieval
- word frequency count
- String sorting
- prefix match
- As an auxiliary structure for other data structures and algorithms
  e.g. Suffix tree, Aho-Corasick automaton