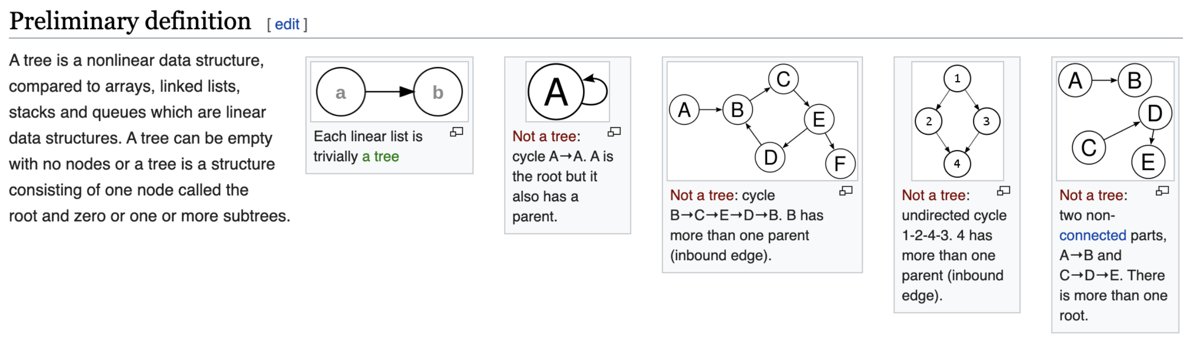
Trees

Intro



Preliminary definition [edit]

A tree is a nonlinear data structure, compared to arrays, linked lists, stacks and queues which are linear data structures. A tree can be empty with no nodes or a tree is a structure consisting of one node called the root and zero or one or more subtrees.

Each linear list is trivially a tree

Not a tree: cycle A→A. A is the root but it also has a parent.

Not a tree: cycle B→C→E→D→B. B has more than one parent (inbound edge).

Not a tree: undirected cycle 1-2-4-3. 4 has more than one parent (inbound edge).

Not a tree: two non-connected parts, A→B and C→D→E. There is more than one root.

Basically, trees are a name for data collections with a descending hierarchy. Parent nodes have child nodes, and descendants can never be the parents of their ancestors. Data always flows from the root through the branches to the leaves.

Types:

- Document Object Model in [X|HT]ML
- Files and Folders
- A To Do List with subtasks
- Your family tree
- Earth, which contains continents, which contain countries, which contain states, which contain suburbs, etc.
- Binary Search Trees

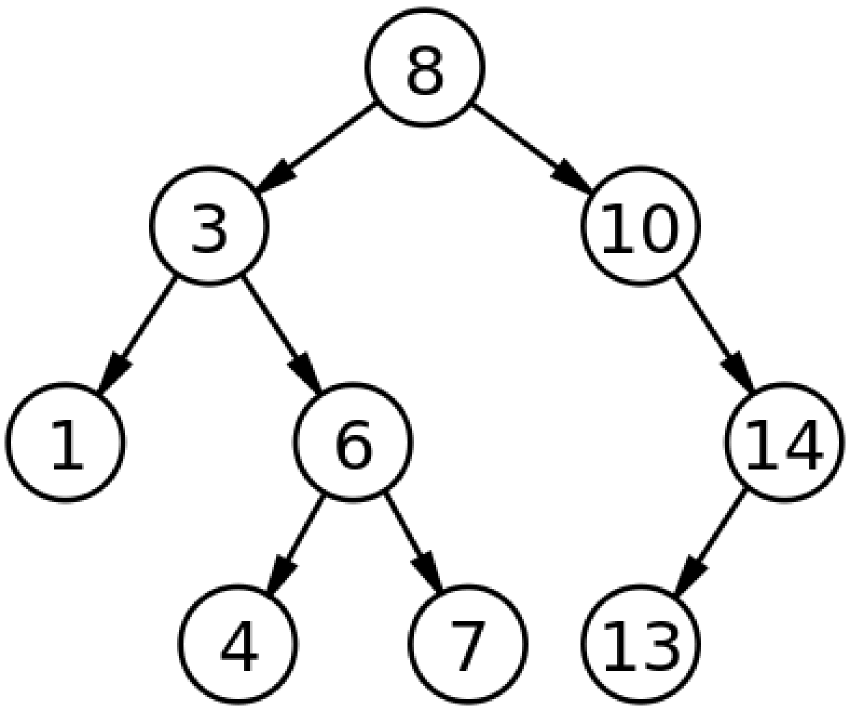
Binary Search Trees

Binary search trees are a specific type of tree with the following properties.

- Each node in the tree has 0, 1, or 2 child nodes.
- The data in the tree is always ordered.
- Trees don't contain cycles. The data flows from top to bottom.

- The order that elements are inserted matters.
- Every new child node that is inserted starts it's life as a leaf node (this is surprisingly important).

This is an example of a BST:



Rules for inserting new nodes:

> Start at the top, if the new value is less than the current node value, branch left. If the new value is greater than the current node value, branch right. Repeat this until you reach a leaf.

Advantages

Inserting and retrieving nodes is generally faster than with linked lists. Finding an element in a randomly ordered linked list with 100 elements is at worst 100. However, finding an element in a BST is usually log_2(100) = 6. Getting used to log_2 notation is important and we'll use it more and more to describe efficiency of our algorithms.

Questions:

1. Construct a BST from the following arrays:
    - `[5, 3, 4, 1, 9, 6]`
    - `[9, 4, 1, 3, 5, 6]`
    - `[1, 2, 3, 4, 5]`
    - `[V, A, P, O, R, W, A, V, E]`
    - `[10, 1, 9, 2, 8, 3]`
2. What order works best for inserting data into a BST?
3. How might you 'visit' all the nodes in a tree?

BST Patterns:

Traverse the whole tree:

```
const traverse = (node) => {
  if(!node){ return; } // Base Case
  traverse(node.left); // Recursive Case
  doSomething(node.val); // Read/Modify the tree in someway
  traverse(node.right); // Recursive Case
}

traverse(root); //
```

Find a node in a BST:

```
const find = (node, val) => {
  if(!node){ return null; }
  if(val < node.val){ return find(node.left, val); }
  if(val === node.val){ doSomething(node.val); }
  if(val > node.val){ return find(node.right, val); }
}
```