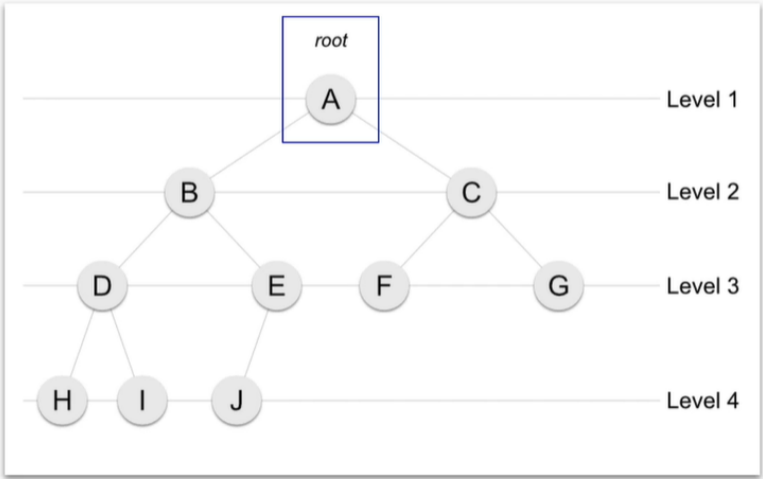
Root

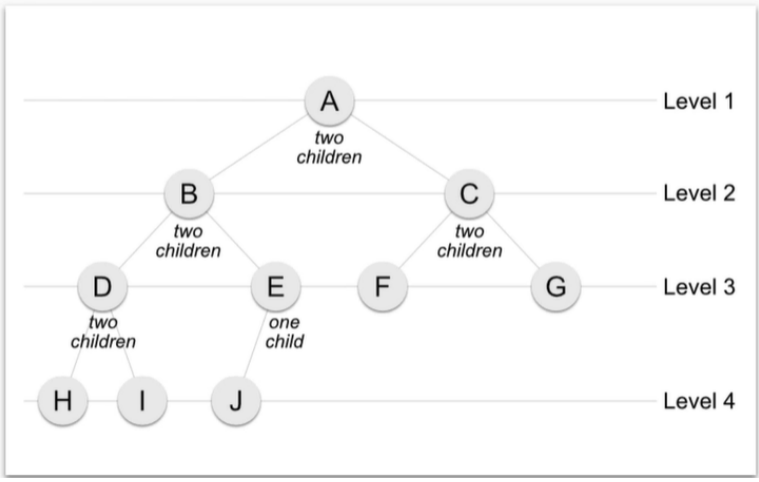
The "first" node in the tree

The node at the top



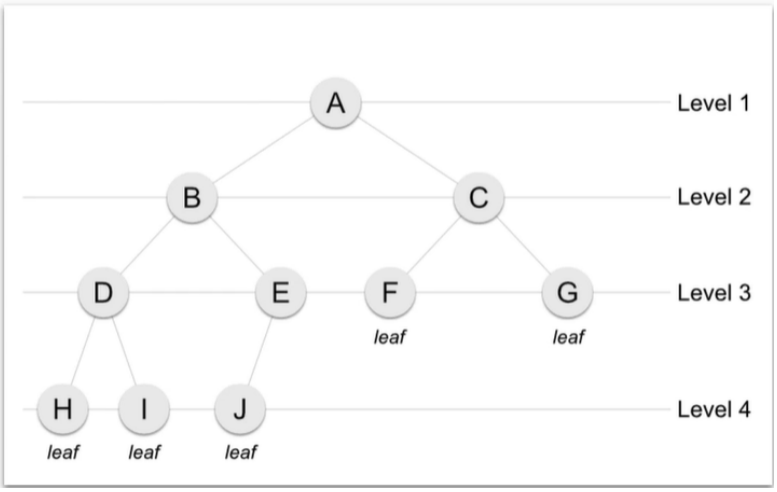
Children

A node has "children" if it has nodes in lower levels associated with it



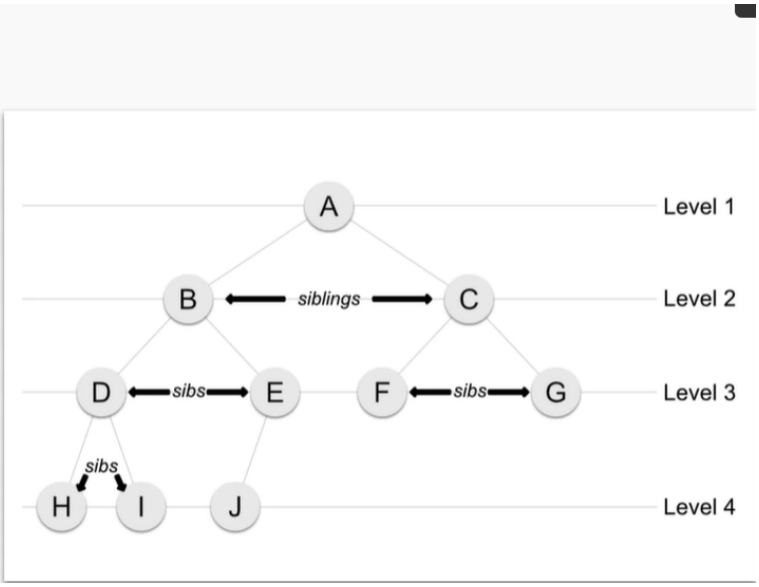
Leaf

And node with not children is called a "leaf node"



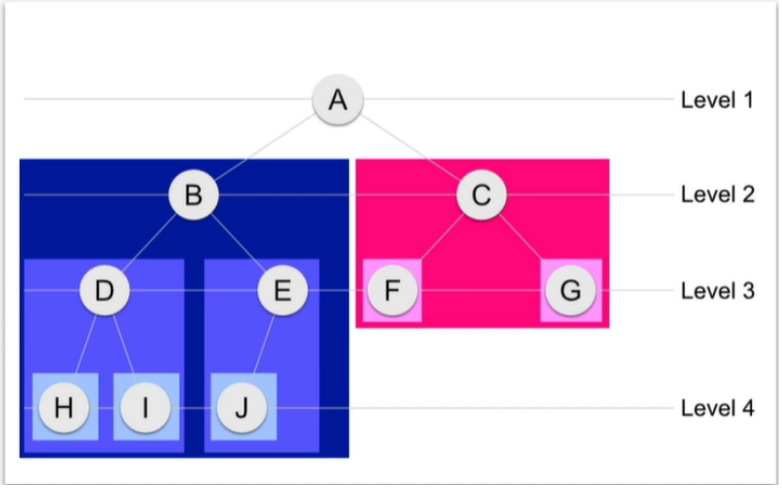
Siblings

Nodes with the same parent are “sibling nodes”



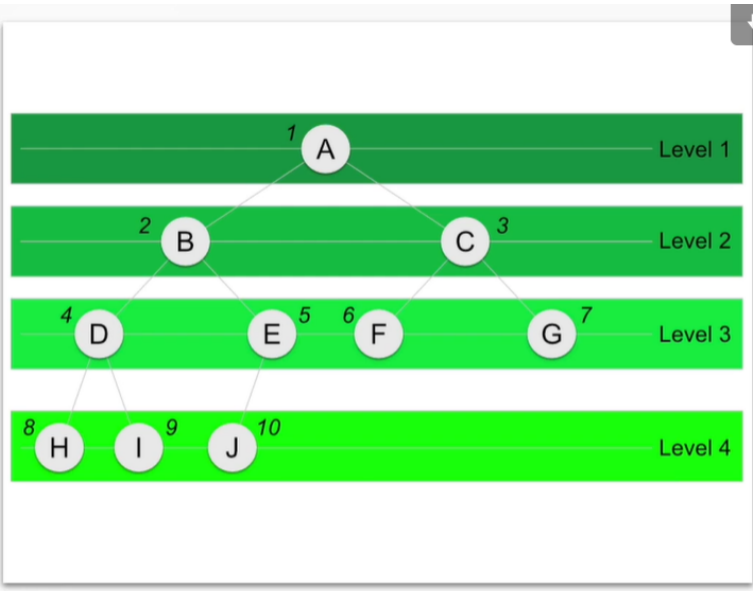
Subtree

Any non-root node and all of its children are a subgraph



Breadth First

Visit by each level, left-to-right

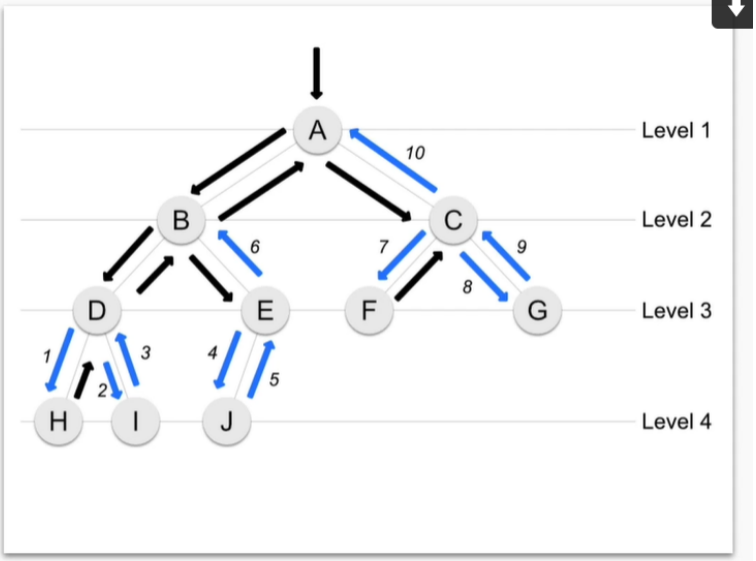


A, B, C, D, E, F, G, H, I, J

Depth First

"Post-order"

Explore each branch as far as you can before backtracking



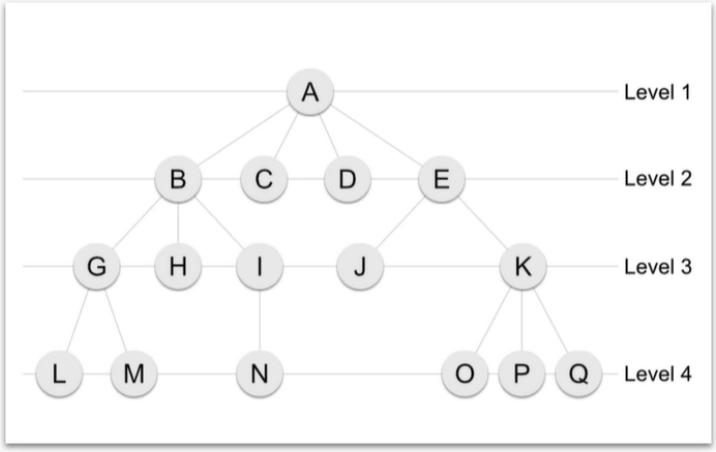
H, I, D, J, E, B, F, G, C, A

Generic (N-ary) Tree

Useful for organizing hierarchical data

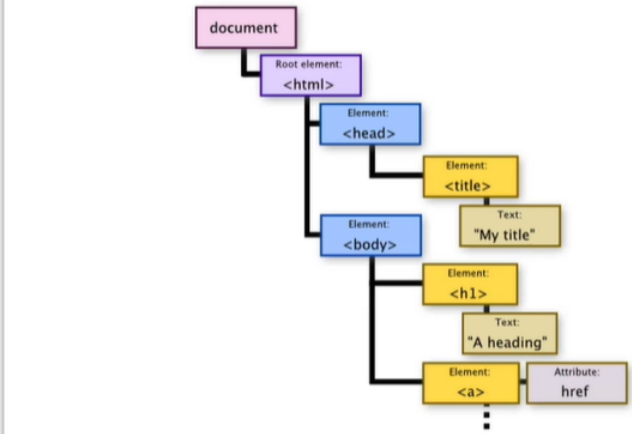
Four operations

- Insert child of
- Remove
- Has?
- Get subtree for



Generic (N-ary) Tree

The DOM!



Generic (N-ary) Tree

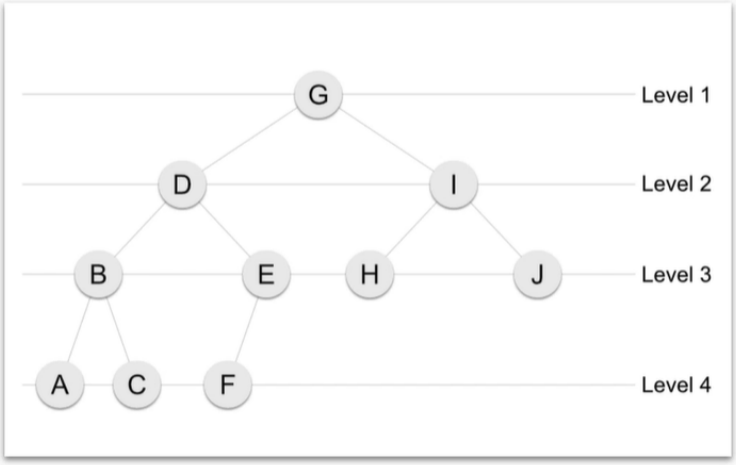
Your computer's file system!

```
.
├── channels
│   └── user_socket.ex
├── controllers
│   ├── account_conrtoller.ex
│   ├── catalog_controller.ex
│   ├── email_registration_controller.ex
│   └── page_controller.ex
├── endpoint.ex
├── gettext.ex
├── plugs
│   └── auth.ex
├── router.ex
├── subdomain.ex
├── templates
│   ├── account
│   │   └── details.html.eex
│   ├── catalog
│   │   └── index.html.eex
│   ├── email_registration
│   │   └── create.html.eex
│   ├── layout
│   │   ├── app.html.eex
│   │   └── catalog.html.eex
```

Binary Tree

Nodes to the left are less than the current node

Nodes to the right are greater than the current node



nodes on right side have to have a greater value than their lefter sibling

when we call add method it figures where to put it...keeps data sorted in such a way as to be easy to search