Trees : Generic Trees

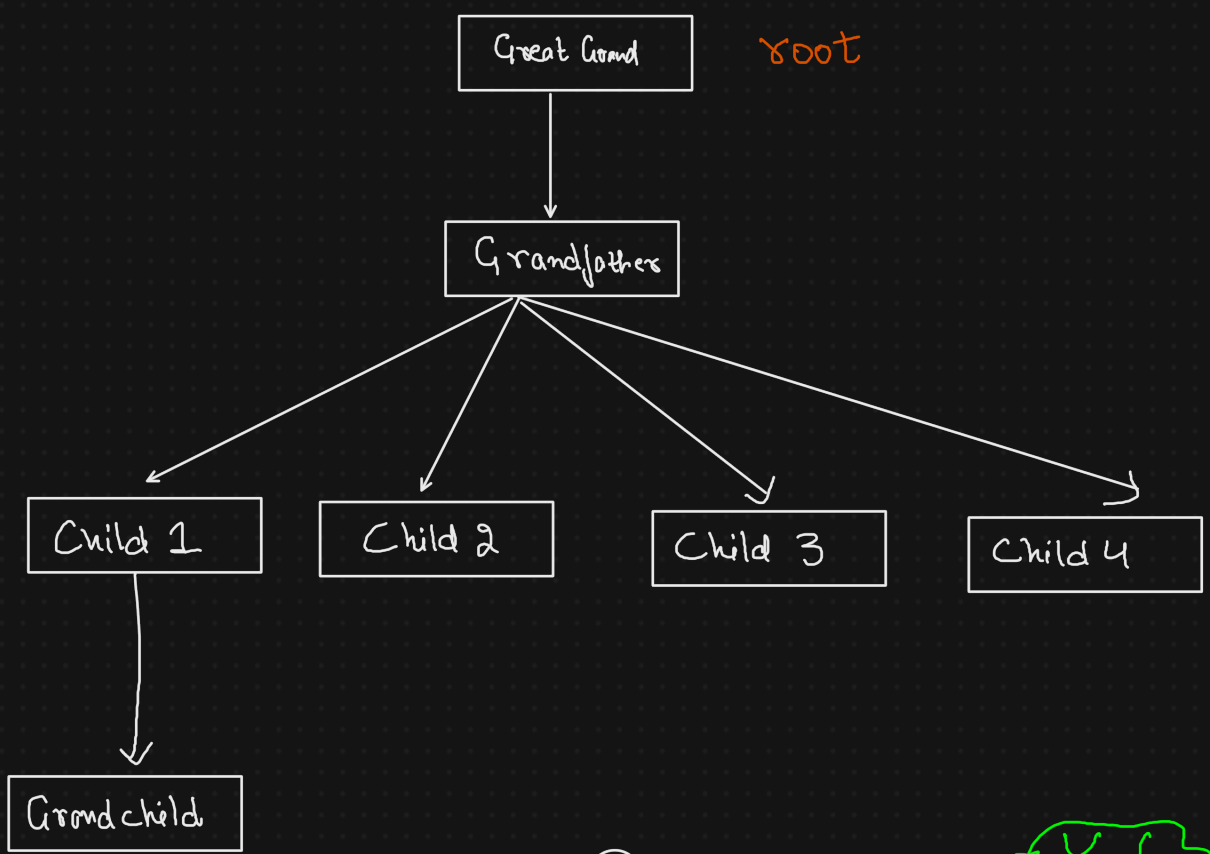
Primary
- Arrays
- Linked list
- Tree

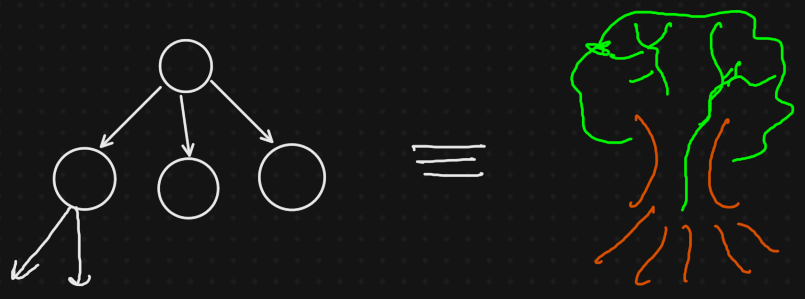
Derived
- Stack
- Queues

Tree

non-linear
heirarchial

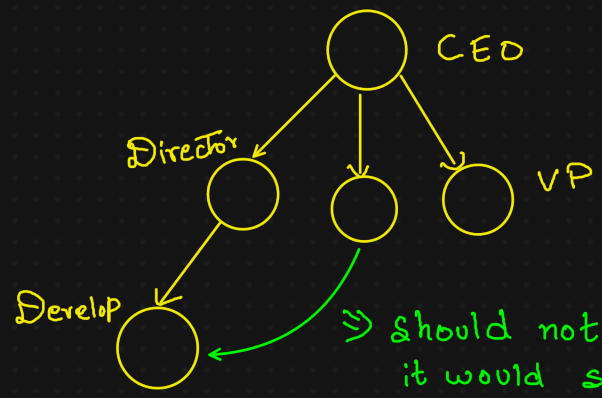
Trees is a heirarchial data structure consisting of nodes which are connected by edges.



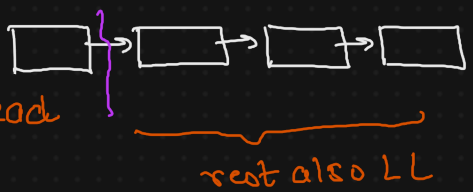


Examples of Trees

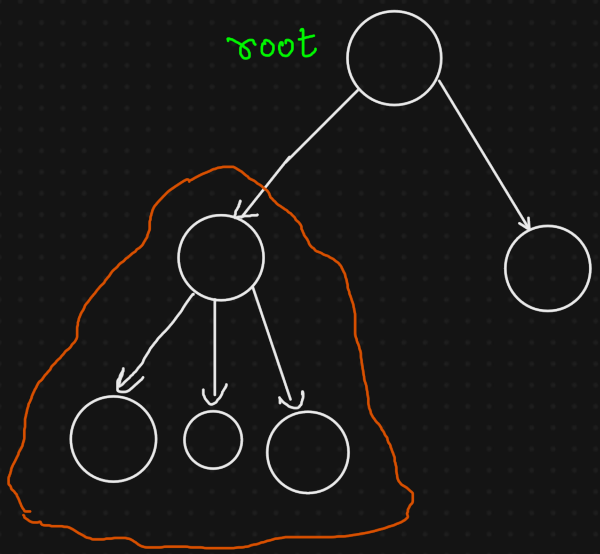
- Family tree
- Organizational chart



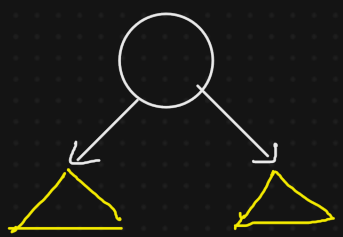
⇒ should not be a edge as it would start a cycle



root also LL

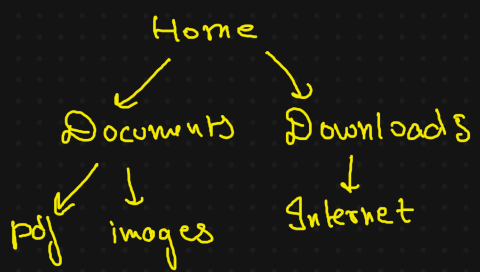


another tree



Practical Examples

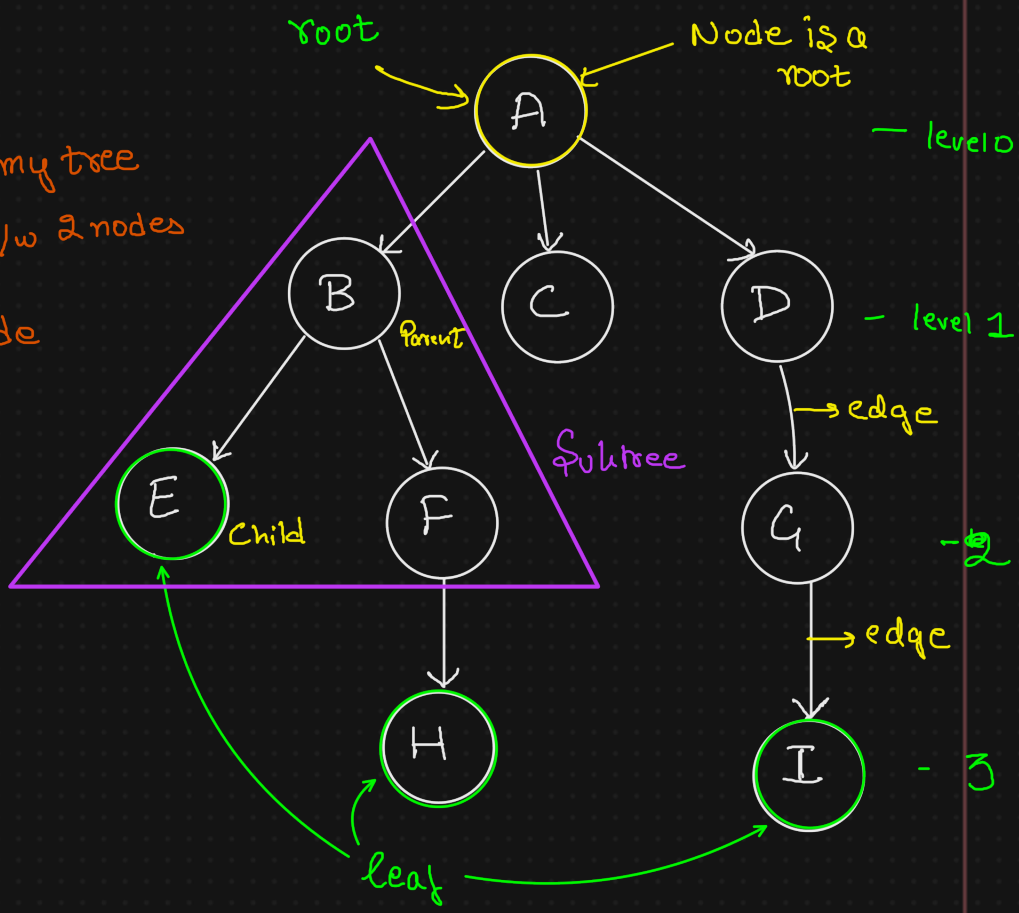
- File structure
- Decision tree



Industry use case

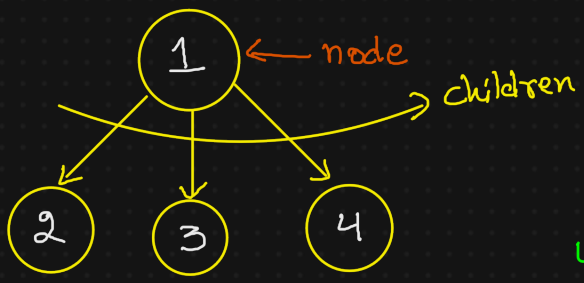
- Indexing in Database
- Network routing
- HTML DOM

Terminology in trees



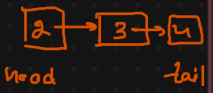
1. Node → each item of my tree
2. Edge → Connection b/w 2 nodes
3. Root → Topmost node
4. Leaf → Node with no children
5. Parent
6. Child
7. Ancestors
   E : A, B
   H : F, B, A
8. Descendants
   B : E, F, H
   D : G, I
9. Subtree → marked in violet
10. Level / Depth
11. Path – sequence of nodes and edges connecting one node to another (we can only move downwards)

Tree Implementation

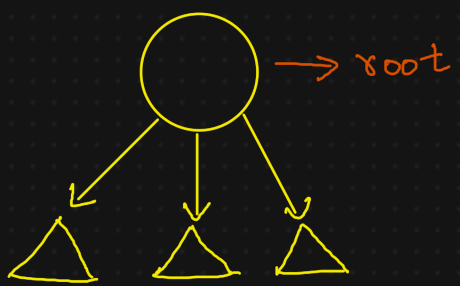


1. Data
2. Children

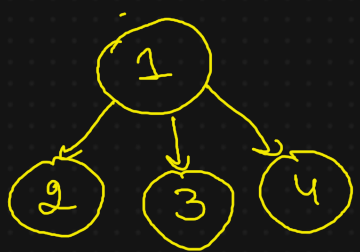
- List
  1. indexing
  2. dynamic nature
- Linked list



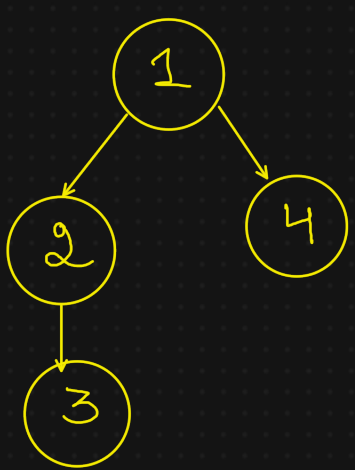
Print Tree



⇒ we handle root and call recursion on the rest

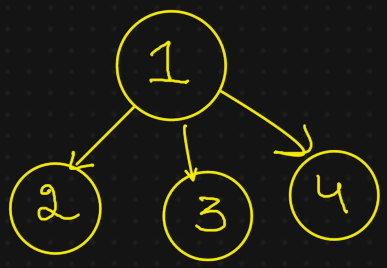


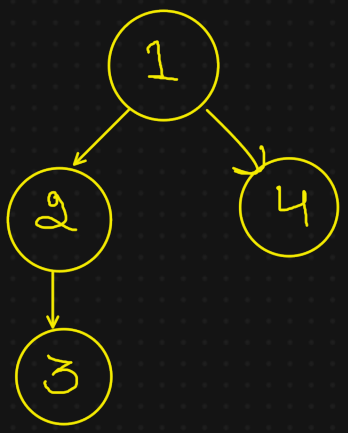
1: 2, 3, 4
2:
3:
4:

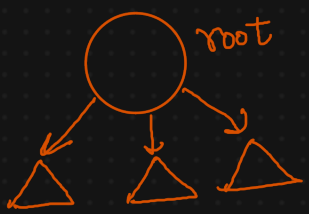


1: 2, 3
2: 3
3:
4:

Take Input - Tree



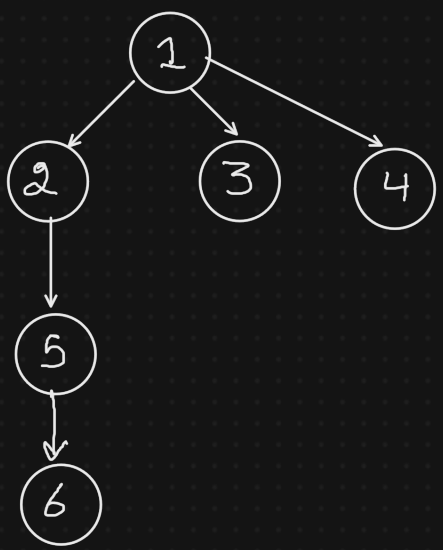




Take Input - Better way

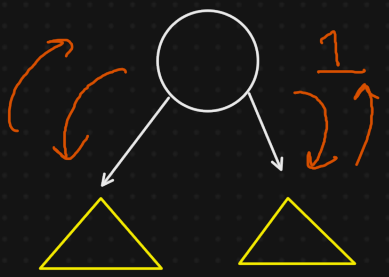
→ taking input level wise

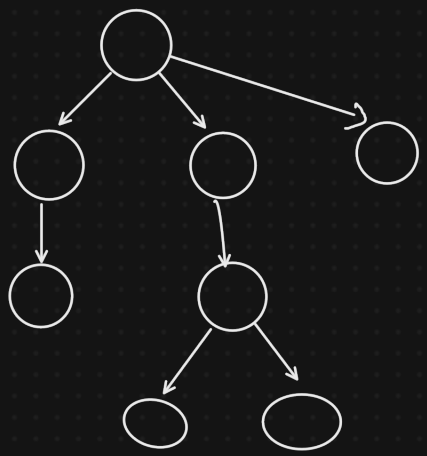
→ we will process our nodes the first to come will be the first to get process.



Queue

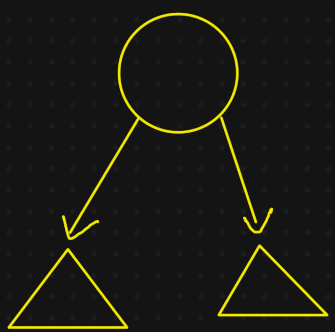
Count Nodes

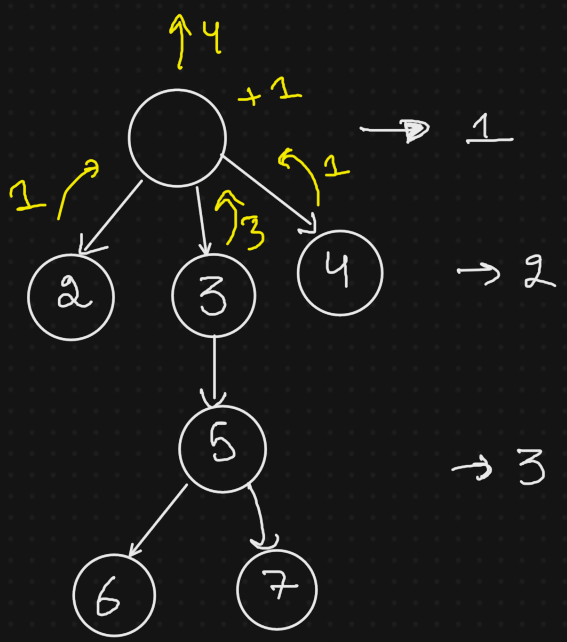




number of nodes += count nodes(each child)

Height of a Tree

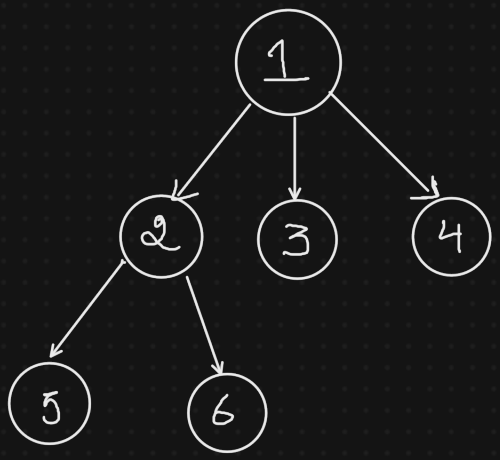




Traversal of a Tree

Preorder: Parent → child

1 2 5 6 3 4



Post order: Child → Parent

5 6 2 3 4 1