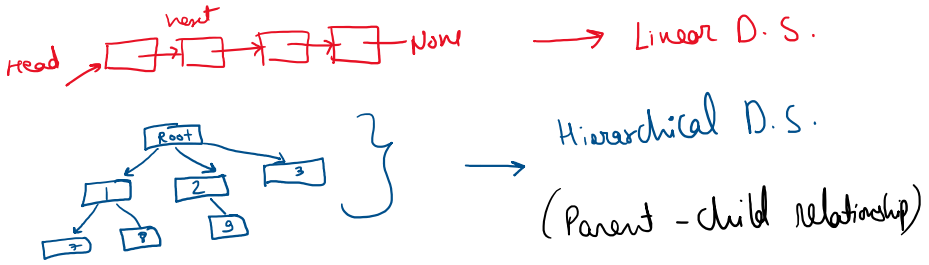
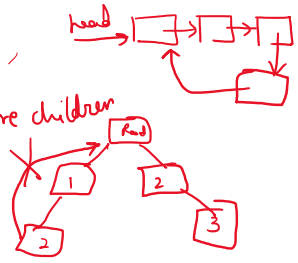
TREES

A tree is a data structure consisting of nodes, where each node has a value and references to child nodes.



Properties:

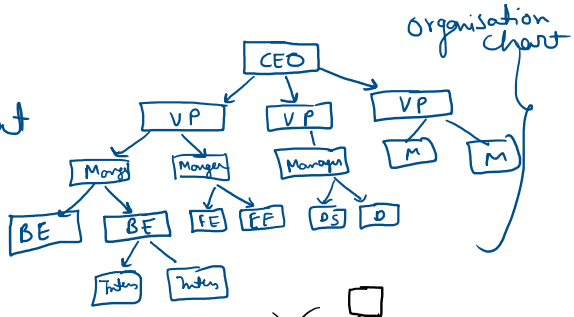
- 1. → Trees are Acyclic
- 2. → Each node can have 0 or more children
- 3. → Root node
  - ↳ Topmost node



Example:

① Family tree

② Organisation chart



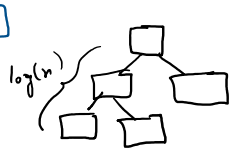
Why do we Need Trees ?

→ Hierarchical Data Storing
  ↳ XML, HTML

→ Efficient Searching / Retrieval

→ Compression of Data → Huffman encoding

→ Routing → shortest path.



Advantages of Trees :

① Dynamic Size → Add elements / Remove

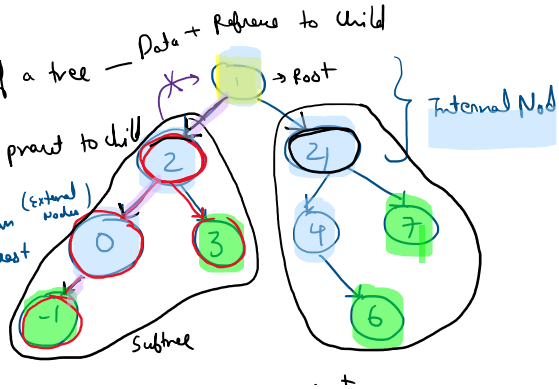
② Efficient Insertion & Deletion

③ Efficient searching.

④ Hierarchical order.

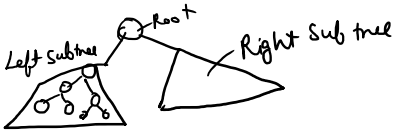
Terminologies:

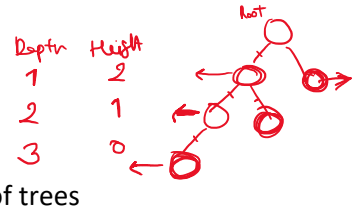
- Node → Fundamental unit of a tree — Data + Reference to child
- Root → Topmost Node
- Edge → Reference / connect for parent to child
- Parent and Child
- Leaf → Nodes with no children (External nodes)
- Internal Node → Node with atleast 1 child
- Subtree →
- Height → length of longest path from root to leaf
- Depth



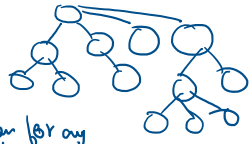
- Subtree → 

- Height → length of Longest path from root to leaf
- Depth
- Path → Set of nodes connecting a Node with descendon
- Ancestor and Descendant
- Degree → No of child





Types of trees



General trees →

Binary tree → 0, 1, 2 children for any node

Binary Search tree → Left side < Right

AVL tree → Balanced → (Height) difference in size of Left subtree & R.S.T ≤ 1

B-Tree

Trie → Multiple child nodes (Self Balanced tree)

- Dictionary, Phone directory
- self balancing tree