**Trees**

* **Linear Data Structure**: Data is arranged in an orderly sequential manner, where the elements are attached adjacently. There is a previous and next element.
* **Non-Linear Data Structure:** Data is arranged in a sorted order and there exists a relationship between the data elements.
* When choosing a data structure, consider the following:
  + What needs to be stored?
  + Cost of operations
  + Memory usage
  + Ease of implementation
* Trees are used commonly for hierarchical data. For example, employees in an organization. This is not the only use of Trees.
* Trees – Collection of nodes linked together to simulate a hierarchy. This is a non-linear structure. Top most node is called the “root.”

**Terms**

* In a tree, a node will have a link or reference to other nodes that are called its children.
* Root, Children, Parent, Sibling are self explanatory.
* **Leaf** – Node with no children.

**Properties**

* Trees are recursive in nature. You can reduce it in a self-similar manner, in that you can form sub-trees.
* If you have N nodes, then there are N – 1 edges in the tree. The only node that doesn’t have an incoming edge is the root.
* **Depth of Node –** Length of path from root to node. In other words, how many edges you need to traverse from root to get to the node. The depth of root is 0 since you don’t traverse any node from root to root.
* **Height of Node –** Number of edges in longest path from the node to a leaf.

**Applications**

* Hierarchical data – file system
* Organize data for quick search, insertion, deletion – Binary Search Tree
* Trie – Dictionary
* Network Routing algorithm

**Binary Tree**

* A tree in which each node has at most two children.
* **Strict/Proper Binary Tree –** Each node has 0 or 2 children.
* **Complete Binary Tree –** All levels except possibly the last are completely filled and all nodes are as left as possible.
* **Level –** Level is the dept of the nodes. Root starts of at 0, then root’s children are 1 and so on. The maximum number of nodes at a given level are 2i where i is the level you are at. So root is at 0, and 20 = 1. The children of root can at most be 2. The children of root are at level 1, so 21 = 2.
* **Perfect Binary Tree** – All nodes in the tree have exactly 2 children.
* **Maximum nodes with height h** = 2h + 1 - 1. You could also say 2(no. of levels) – 1.
* From the equation above, you can get log2(n + 1) – 1.
* **Minimum height of tree = log2n** – You want minimum height to minimuze search operations. If you have maximum height in the worst case, you’re basically looking at a linked list.
* **Balanced Binary Tree –** Difference between height of left and right subtree for every node is not more than k (mostly 1).

**Binary Search Tree**

* This is an attractive Data Structure, since Search, Insert, and remove will have a cost of O(log n) in the average case. In the worst case it is O(n), but we can avoid this by balancing the tree, as defined above.
* **Definition –** A binary tree in which for each node, value of all the nodes in left subtree is lesser or equal and value of all nodes in right subtree is greater.

**Binary Heap**