General Trees

→ provide a natural organization for data and allow us to implement a host of algorithms much faster than a linear structure

→ relationships in a tree are hierarchical with some objects being "above" and some "below" others

Definitions & Properties

→ with exception of top element/root, each element in a tree has a parent and zero or more child elements

→ Formal Tree Definition

a tree T as a set of nodes storing elements such that the nodes have a parent-child relationship which satisfies:

- if T is nonempty, it has a root which has no parent
- each node v of T different from the root has a unique parent node w; every node with parent w is a child of w.

Edges and Paths

→ edge: pair of nodes (u,v) such that u is the parent of v (or vice-versa)

→ path: sequence of nodes such that any 2 consecutive nodes in the sequence form an edge

Tree Abstract Data Type: using concept of position as an abstraction for node in a tree

- T.root(): return position of root of T
- T.is_root(p): return true if p is root
- T.parent(p): return position of parent of position p
- T.num_children(p): return # of children
- T.children(p): generate iteration of children
- T.is_leaf(p): return true if no children
- len(T): return # of positions contained in T
- T.is_empty(): return true if T doesn't contain any positions
- T.positions(): generative iteration of positions in T
- iter(T): generate iteration of all elements stored in T