Informal Definition: Tree

Mathematical abstraction that plays a central role in the design and analysis of algorithms
- build and use explicit data structures that are concrete realizations of trees
- describe the dynamic properties of algorithms

Formal Definition: Tree

Tree: set of nodes storing elements in a parent-child relationship with the following properties:
- \( T \) has a special node \( r \), called the root of \( T \), with no parent node;
- Each node \( v \) of \( T \), such that \( v \neq r \), has a unique parent node \( u \)

*Note a tree cannot be empty (must have root)*
- Just a convention
Formal Definition (Alternative)

Tree: Nonempty collection of nodes (vertices) and edges (links) in which there exists exactly one path connecting any two nodes.

Some Tree Terminology

- **Root**: "top-most" vertex in the tree
  - the initial call
- **Parent/Child**: direct links in tree
- **Siblings**: children of the same parent
- **Ancestor**: predecessor in tree
  - closer to root along path
- **Descendent**: successor in tree
  - further from root along path

Some Tree Terminology

- **Internal node**: a node with children
- **Leaf/External node**: a node without children
- **Ordered Tree**: linear ordering for the children of each node
- **Binary Tree**: ordered tree in which every node has at most two children
- **Proper Binary Tree**: binary tree in which every node has exactly zero or two children
Tree Terminology

- Root: node without parent (A)
- Internal node: node with at least one child (A, B, C, F)
- External node (a.k.a. leaf): node without children (E, I, J, K, G, H, D)
- Ancestors of a node: parent, grandparent, grand-grandparent, etc.
- Depth of a node: number of ancestors
- Height of a tree: maximum depth of any node (3)
- Descendant of a node: child, grandchild, grand-grandchild, etc.

Subtree: tree consisting of a node and its descendants

Properties of Binary Trees

- Level 0 has 1 node (the root)
- Level 1 has at most 2 nodes
- Level 2 has at most 4 nodes
- ...
- Level d has at most $2^d$ nodes

Properties of Binary Trees

Let $T$ be a (proper) binary tree with $n$ nodes, and let $h$ denote the height of $T$:

1. The number of external nodes in $T$ is at least $h+1$ and at most $2^h$.
2. The number of internal nodes in $T$ is at least $h$ and at most $2^h - 1$.
3. The total number of nodes in $T$ is at least $2h+1$ and at most $2^h - 1$.
4. The height of $T$ is at least $\log(n+1) - 1$ and at most $(n-1)/2$, that is $\log(n+1) - 1 \leq h \leq (n-1)/2$. 

Tree ADT: Functions

Query Functions
- `isInternal(v)`, `isExternal(v)`, `isRoot(v)`: test whether tree is ..., return Boolean

Generic Functions
- `size()`: return number of nodes in tree
- `elements()`, `positions()`: iterate on elements/positions of tree
- `swapElements(v, w)`: swap elements stored at nodes v and w
- `replaceElement(v, e)`: replace element stored at node v with e

Trees: Data Structures

Vector Implementation
- root at index 1
- left child of node i at 2*i
- right child of node i at 2*i + 1
- some indices may be skipped
- can be space prohibitive for sparse trees

Trees: Data Structures

List Implementation

```c
struct node
{
    Item item;
    node *left, *right, *parent;
};
```
- if node is root, then *parent is null
- if node is external, then *left and *right are null
Translating General Trees into Binary Trees

T: General tree
T': Binary tree

Intuition:
- take set of siblings \( v_1, v_2, \ldots, v_k \) in \( T \) that are children of \( v \)
- \( v_1 \) becomes left child of \( v \) in \( T' \)
- \( v_2, \ldots, v_k \) become chain of right children of \( v_1 \) in \( T' \)
- recurse

Translating General Trees into Binary Trees

Algorithm
1) \( u \in T \Rightarrow u' \in T' \)
2) ??
3) if \( u \in T \) is internal, and \( v \) is leftmost child of \( u \), then \( v' \) is left child of \( u' \in T' \)
4) if \( v \) has sibling \( w \), then \( w' \) is right child of \( v' \in T' \)

An Example

![Diagram of tree transformation]
Summary

- Trees have intuitive definitions
  - think family tree
- Tree ADTs have specific functions
  - root(), children(v), isExternal(v), swap(v,w),…
- Trees can be implemented
  - as array (vector)
  - as linked structure
- General trees can be converted to binary trees