Implementing Trees as Lists
The Tree is a Pervasive Information Structure

- Files & Directories
- Family Trees
- Management Hierarchies

In the current discussion:
- Trees are the abstraction
- Lists are the implementation
Files and Directories

/
  /dev
    /dev/console
    /dev/dsk
      /dev/dsk/dsk01
      /dev/dsk/dsk02
  /etc
    /etc/mail
  /usr
    /usr/bin/emacs
    /usr/bin/ls
    /usr/bin/more
Family Trees

Joseph Patrick Kennedy* - (m. Rose Elizabeth Fitzgerald*)

- Joseph Patrick Kennedy Jr.*
- Rosemary Kennedy

-Kathleen Kennedy* (m. William John Robert Cavendish)

-Eunice Mary Kennedy (m. Robert Sargent Shriver Jr.)

- Robert Sargent Shriver III
- Maria Owings Shriver (m. Arnold Schwarzenegger)
- Timothy Perry Shriver
- Mark Kennedy Shriver
- Anthony Paul Shriver

-Patricia Kennedy (m. Peter Lawford*, divorced)

- Christopher Kennedy Lawford
- Sydney Maleia Lawford McKelvey
- Victoria Francis Lawford Ponder
- Robin Elizabeth Lawford

-Robert Francis Kennedy* (m. Ethel Skakel)

- Kathleen Hartington Kennedy
- Joseph Patrick Kennedy II
- Robert Francis Kennedy Jr.
- David Anthony Kennedy*
- Mary Courteny Kennedy
- Michael LeMoyne Kennedy*
- Mary Kerry Kennedy
- Christopher George Kennedy
- Matthew Maxwell Taylor Kennedy
- Douglas Harriman Kennedy
- Rory Elizabeth Katherine Kennedy

-Jean Ann Kennedy (m. Stephen Edward Smith*)
- Stephen Edward Smith Jr.
- William Kennedy Smith
- Amanda Mary Smith
- Kym Marie Smith

-Edward Moore Kennedy (m. #1: Virginia Joan Bennet)
- (m. #2: Victoria Anne Reggie)

- Kara Ann Kennedy Allen
- Edward Moore Kennedy Jr.
- Patrick Joseph Kennedy

-John Fitzgerald Kennedy* (m. Jacqueline Lee Bouvier*)

- unnamed daughter* (still born)
- Caroline Bouvier Kennedy (m. Edwin A. Schlossberg)

- Rose Schlossberg
- Tatiana Schlossberg
- John Schlossberg

-John Fitzgerald Kennedy Jr.* (m. Carolyn Bessette*)

- Patrick Bouvier Kennedy*
Organization Chart
(Management Hierarchy)
Definition of Tree

- There are many different varieties of trees.
- We discuss only some of them.
- Use your knowledge of these to generalize to other varieties.
- We will base our definition on paths and related concepts.
A path in a graph $G$ is a list of nodes $n_0, n_1, \ldots, n_k$ such that each successive pair $(n_i, n_{i+1})$ is in the corresponding binary relation.

Some paths:
- $a, b, d$
- $c, e, a$
- $a, c, e, a, c, d$
A cycle is a path that starts and ends on the same node.

Examples:
- a, c, e, a
- e, a, c, e, a, c, e
A cyclic graph is one that has at least one cycle.

An acyclic graph is one that has no cycles.
DAGs

- DAG is an acronym for “Directed Acyclic Graph”
- DAG is mainly used because it is more pronounceable than ADG (“Acyclic Directed Graph”)
Target Set

- The target set of a node $n$ is the set of nodes to which there is an arc from $n$.

- $\text{targets}(a) = \{b, c\}$
- $\text{targets}(b) = \{d\}$
- $\text{targets}(c) = \{d, e\}$
- $\text{targets}(d) = \{\}\$}
- $\text{targets}(e) = \{a\}$
If a node’s target set is empty, that node is called a leaf.
Fan-In

- A directed graph is said to **fan-in at** node \( n \) if the node is in the target sets of two or more different nodes.
- A directed graph “**has fan-in**” if it fans in at least one node.
A root of a directed graph is a node that is not in any node's target set.

No roots:

\begin{figure}[h]
\centering
\includegraphics[width=0.4\textwidth]{no_roots.png}
\end{figure}

B and c are roots:

\begin{figure}[h]
\centering
\includegraphics[width=0.4\textwidth]{roots.png}
\end{figure}
Tree at Last

- A tree is a directed graph such that:
  - The graph is acyclic.
  - There is exactly one root.
  - It has no fan-in.
A tree:

Not a tree:
Classify these for Tree-dom
More Graphs to Classify
Reverse Graphs

- Some graphs that may look tree-like aren’t technically trees unless we consider the reverse graph (one with all of the arcs of the original reversed).

![Diagram of a graph and its reverse graph]
Reconvergence, an Alternative

- A reconvergence is a pair of different paths that start and end, respectively, on the same nodes.
- Therefore, a tree can also be characterized as a directed graph that:
  - has one root
  - has no cycles
  - has no reconvergences

Reconvergence below:
- $c, d, e$
- $c, e$

\[
\begin{align*}
\text{d} & \quad \text{c} \\
\text{e} & \quad \text{c} \\
\end{align*}
\]
Subsets of Three Properties

- **DAG**: acyclic, but
  - may have multiple roots,
  - may have fan-in
- **Forest**: acyclic, and no fan-in but
  - may have multiple roots
- A forest can also be characterized as a **collection of disjoint trees**.
  Each tree could be identified with its root.
Adding/Removing Arcs

- Adding arcs to a ______ that is not a tree may make it into a tree.
- Adding arcs to a ______ that is not a tree will never make it into a tree.
- Removing arcs from a ______ that is not a tree may make it into a tree.
- Removing arcs from a ______ that is not a tree will never make it into a tree.
Ordered Directed Graphs

- We use the adjective *ordered* to indicate that the order of targets of a node matters.
- This property is *implicit* with trees much of the time.
- Because we are going to represent trees by lists, we can have ordering for free if we want it.
Representing/Implementing Trees as Lists

- Every tree can be represented as a list.
- Obvious:
  - Tree is a special kind of directed graph.
  - Every directed graph can be represented as a list of pairs.
- But we want a representation that makes it clear that we have a tree.
First Try: Target Sets

- We know that sets can be represented as lists.
- Use a list to represent the target set of each node.
- Associate each node with its list of targets.
Target-Lists Representation

- \([ [a, [b, c]], [b, []], [c, [d, e]], [d, []], [e, []]]\)

- This works for graphs in general; is not limited to trees.
- Doesn’t directly show tree-dom.
Nested Target-Lists Representation

- List the root, followed by the representation of each sub-tree:

- $[a, \_\_, \_\_]$
- $[a, [b], [c, \_\_, \_\_]]$
- $[a, [b], [c, [d], [e]]]$
A leaf is a node with no targets.

When a sub-tree is a leaf, omit the brackets around it.

- \([a, ____, ____]\)
- \([a, b, [c, ____, ____]]\)
- \([a, b, [c, d, e]]\)

Less-cluttered appearance but also less uniform.
Representation of “Unlabelled” Trees

- In this model, only *leaves* have labels.
- A leaf is represented by its label
- A non-leaf tree is represented by a list of the representations of the targets of the root.
  - \([a, \_\_\_\_\_\_\_\_\_]\)
  - \([a, [b, c]]\)
Representing Lists by Ordered Trees

- (This may look “backward” at first.)
- Every list can be represented as an ordered **binary tree** (tree in which each node has at most two targets).
- This corresponds to the “box” storage abstraction, where the data items may themselves be lists.
Representing Lists as Trees

- An **atomic** item (non-list) is represented by itself.
- The **null list** is represented as a leaf `[]`.
- A list `[First | Rest]` is represented by a node with two targets:
  - The **left** target is the representation of First.
  - The **right** target is the representation of Rest.
- Note that ordering of targets is essential.
Representing Lists as Trees

Atom a: \( a \)

Empty list \([\ ]\): \( [\ ] \)

Non-empty list \([F \mid R]\):

- Rep. of F
- Rep. of R
Representing Lists as Trees

- Matters are actually simpler if we rotate the tree 45°, so that “right” is horizontally right and “left” is down.

Tree representing the first element

Tree representing the rest of the list

Tree representing the list
Example: Binary Tree

- Represent as a binary tree: [1, 2, 3]
Example: Binary Tree

- Represent as a binary tree:
  \([1, [2, 3], [4]]\)
Corresponding Box Diagram

\[ [1, [2, 3], [4]] \]