Search Trees & Graphs
Recursion on Trees
Tree Dichotomy

- We often want a different dichotomy than for lists: either
  - The tree has a single node (and is thus a leaf), or
  - The tree is a node with offspring (and is thus **not a leaf**).
Model Independence

- We can abstract away the specific representation being used:
  - **isLeaf**(T) 1 when T is leaf, 0 otherwise.
  - **offspring**(T) the list of offspring of a non-leaf, undefined otherwise.

- The implementation depends on which tree model we are using.
Two Possible Implementations

- Unlabeled-Tree Implementation:
  - `makeTree(ListSubTrees) = ListSubTrees;
  - `isLeaf(T) = atomic(T);
  - `offspring(T) = T;

- Labeled-Tree Implementation:
  - `makeTree(Root, ListSubTrees) = [Root | ListSubTrees];
  - `isLeaf(T) = null(rest(T));
  - `offspring(T) = rest(T);
  - `root(T) = first(T);
Recursion on Trees

- **Basis**: What happens on a single leaf.

- **Induction step**: What happens on a non-leaf.
Example: Height of a Tree

- The height of a tree is the length of the longest path from the root.

- \( \text{height}(T) \Rightarrow \text{isLeaf}(T) \ ? 0; \)

- \( \text{height}(T) \Rightarrow 1 + \maxList(\text{map(height, offspring(T))}); \)

- Let recursion do the work for you.
Searching a Tree

- Suppose we want to find all nodes with labels having a property P.

- Here we need to say whether the interior nodes are labeled or not.
Searching a Labeled Tree

- \( \text{findInTree}(P, T) = \)
  
  \[
  \text{Root} = \text{root}(T), \\
  \text{foundInRest} = \text{mappend}((S) \mapsto \text{findInTree}(P, S), \text{offspring}(T)), \\
  \]

  \[
  P(\text{Root}) \text{ ? } [\text{Root} \mid \text{foundInRest}] : \text{foundInRest};
  \]
Depth-First Search

- The preceding expresses only one form of search:

  depth-first search

  The pattern is to search “deeper” before “broader”.
Example: Find evens
Wavefront Analogy
Advantage of Breadth-First
Breadth-First Searching a Labeled Tree

- \( \text{findInTreeBF}(P, T) = \text{findInForest}(P, [T]); \)

- \( \text{findInForest}(P, [\ ]) \Rightarrow [\ ]; \)
  \[ \text{findInForest}(P, [\Tree | Trees]) \Rightarrow \]
  \[
  \text{Root} = \text{root}(\Tree), \]
  \[
  \text{foundInRest} = \text{findInForest}(P, \text{append}(Trees, \text{offspring}(\Tree)), P(\text{Root}) ? [\text{Root} | \text{foundInRest}] : \text{foundInRest}; \]
Searching Graphs vs. Trees

- Basic ideas still apply
- In graph, avoid re-searching same nodes due to fanin-in
  - In graph, avoid infinite loops
Depth- vs. Breadth- First in Graph

Example: Find evens
Searching Without Recursion

- **Depth-First**: Use Stack
- **Breadth-First**: Use Queue
- **Avoid Fan-in and Cycles**: 
  - “Mark” nodes as encountered
    - Refuse to re-search from a marked node
  - Marking can be virtual, e.g. by membership on a list, or
  - The node itself can be marked (non-functional programming).
Searching a Maze

- A maze is an implicit graph
- Nodes are identifiable by position
- The arcs are implicit
- Marking can be done in a “parallel array”