CSCI 1900
Discrete Structures

Searching Trees
Reading: Kolman, Section 7.3

Tree Searching

Trees can represent the organization of elements, but they can also represent a process with each vertex specifying a task.

– For-loop example
```
for i = 1 to 3
    for j = 1 to 5
        array[i,j] = 10*i + j
    next j
next i
```
– Mathematical expression from section 7.2 – each vertex represents a computation that combines its offspring.

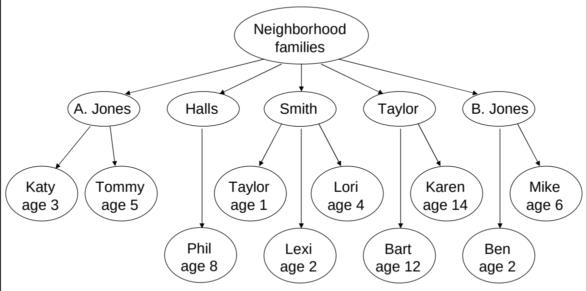
Terminology

- “Visiting” a vertex – the act of performing a task at a vertex, e.g., perform a computation or make a decision.
- “Searching” the tree – the process of visiting each vertex in a specific order or path. The term “searching” can be misleading. Just think of it as “traversing” the tree.

Tree Search

- The application of some trees involves traversing the tree in a methodical pattern so as to address every vertex.
- Our book uses the term “search”, but sometimes search can imply we’re looking for a particular vertex. This is not the case.
- Example:
  Assume we want to compute the average age, maximum age, and minimum age of all of the children from five families. (Tree is on next slide.)

Search Example



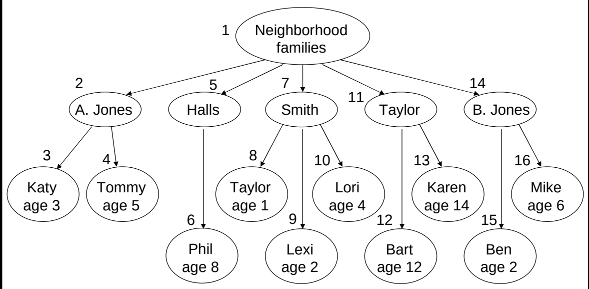
Search Example (continued)

- To calculate the average, max, and min ages for all of the children, we need to have a method for going through the tree so that we don’t miss a child.
- By defining a rigorous process, not only can a human be sure not to miss a vertex, but also an algorithm can be defined for a computer

A Suggested Process for Searching a Tree

1. Starting at the root, repeatedly take the leftmost "untraveled" edge until you arrive at a leaf which should be a child.
2. Include this child in the average, max, and min calculations.
3. One at a time, go back up the edges until you reach a vertex that hasn't had all of its outgoing edges traveled.
4. If you get back to the root and cannot find an untraveled edge, you are done. Otherwise, return to step 1.

Vertices Numbered in Order of Visits



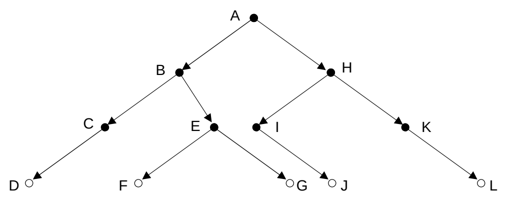
Preorder Search

- This methodical pattern of traversing a tree is called a ***preorder search***.
- Assume v is the root of a binary positional tree T.
  - Each vertex of this tree has at most a left vertex, v_L, and a right vertex, v_R.
  - If either v_L or v_R have offspring, then they are subtrees of T, and a search can be performed of them too.
  - By viewing a tree this way, then the search method we described in earlier slides can be performed using a recursive algorithm applied to each vertex.
  - The recursive algorithm is repeatedly applied until every leaf has been reached.

Preorder Search Algorithm

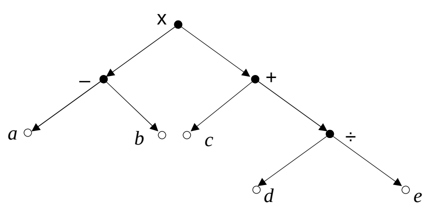
- A preorder search of a tree has the following three steps:
  1. Visit the root
  2. Search the left subtree if it exists
  3. Search the right subtree if it exists
- The term "search" in steps 2 and 3 implies that we apply all three steps to the subtree beginning with step 1.

Vertices Visited in Alphabetical Order Using Preorder Search



Prefix or Polish Form

Binary tree representing: $(a - b) \times (c + (d \div e))$



Preorder search produces: $\times - a\ b + c \div d\ e$

Polish Form (continued)

- Allows us to write complex arithmetic expressions without using parenthesis
- Expression is evaluated by performing following steps:
  - Move left to right until you find a string of the form *Fxy*, where *F* is the symbol for a binary operation and *x* and *y* are numbers.
  - Evaluate *x F y* and substitute answer for string *Fxy*.
  - Repeat starting at beginning of string again.

Polish Form Example

1. × – 6 4 + 5 ÷ 2 2	1st pattern: – 6 4
2. × 2 + 5 ÷ 2 2	2nd pattern: ÷ 2 2
3. × 2 + 5 1	3rd pattern: + 5 1
4. × 2 6	4th pattern: × 2 6
5. 12	

(6 – 4) × (5 + (2 ÷ 2)) = 12

Inorder and Postorder Searches

- Preorder search gets its name from the fact that the operator that joins two items is evaluated first, e.g., the binary operation 6 – 4 is visited in the order – 6 4.
- Inorder search evaluates the expression as it is written, e.g., the binary operation 6 – 4 is visited in the order 6 – 4.
- Postorder search evaluates the operator after the elements are read, e.g., the binary operation 6 – 4 is visited in the order 6 4 –.

Inorder Search Algorithm

An inorder search of a tree has the following three steps:

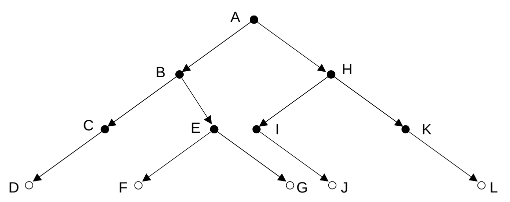
1. Search the left subtree if it exists
2. Visit the root
3. Search the right subtree if it exists

Postorder Search Algorithm

A postorder search of a tree has the following three steps:

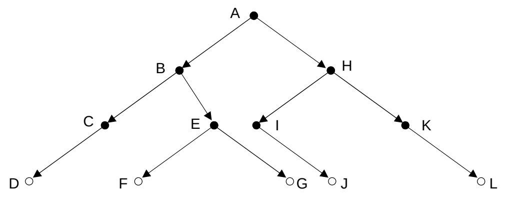
1. Search the left subtree if it exists
2. Search the right subtree if it exists
3. Visit the root

Evaluation of Tree Using Inorder Search



Resulting string: DCBFEGAIJHKL

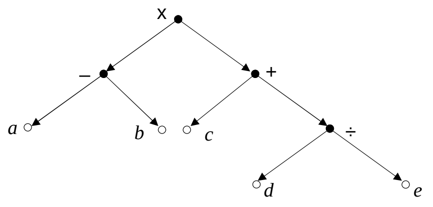
Evaluation of Tree Using Postorder Search



Resulting string: DCFGEBJILKHA

Infix Form

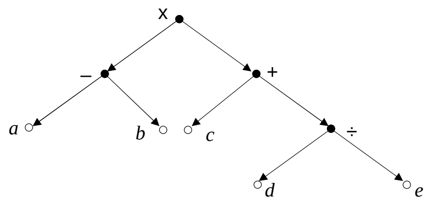
Binary tree representing: $(a - b) \times (c + (d \div e))$



Inorder search produces: $a - b \times c + d \div e$
Unfortunately, without parenthesis, we can't do anything with this expression.

Postfix or Reverse Polish Form

Binary tree representing: $(a - b) \times (c + (d \div e))$



Inorder search produces: $a\ b - c\ d\ e \div + \times$

Reverse Polish Form (continued)

- Allows us to write complex arithmetic expressions without using parenthesis
- Expression is evaluated by performing following steps:
  - Move left to right until you find a string of the form *xyF*, where *F* is the symbol for a binary operation and *x* and *y* are numbers.
  - Evaluate *x F y* and substitute answer for string *xyF*.
  - Repeat starting at beginning of string again.

Reverse Polish Form Example

From left-to-right evaluate *xyF* first.

1. 2 1 – 3 4 2 ÷ + ×	1st pattern: 2 1 –
2. 1 3 4 2 ÷ + ×	2nd pattern: 4 2 ÷
3. 1 3 2 + ×	3rd pattern: 3 2 +
4. 1 5 ×	4th pattern: 1 5 ×
5. 5	

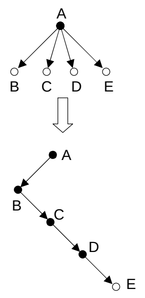
$(2 - 1) \times (3 + (4 \div 2)) = 5$

Converting an Ordered n-tree to a Positional Binary Tree

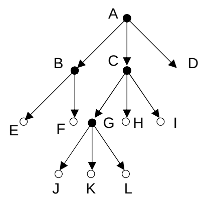
- An ordered n-tree where some vertices have more than two offspring can be converted to a positional binary tree.
- This allows easier computer representation with methods such as linked lists.
- A process exists for this conversion that works on any finite tree.

Process to Convert Ordered n-tree to Positional Binary Tree

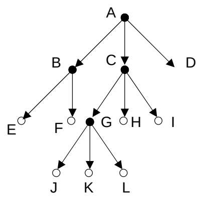
- Starting at the root, the first or leftmost offspring of a vertex remains the leftmost vertex in the binary tree
- The first sibling to the right of the leftmost vertex becomes the right offspring of the leftmost vertex
- Subsequent siblings become the right offspring in succession until last sibling is converted.

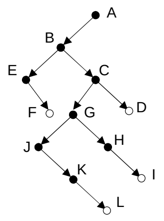


Conversion Example



Conversion Example





Preorder search:
  Left tree – ABEFCGJKLHID
  Right tree – ABEFCGJKLHID