Trees defined and some treeingo and measures

Explicit versus Implicit trees.

Other Tree Applications.

Expression Trees.

Special features about postix.

Infix, Prefix, Postfix notations for expressions.

CSI 310: Lecture 22
Parentheses!

Prefix and postfix notation can express every expression without any parentheses.

Fact: Supposing each operator has its own (fixed) number of operands, the expression

Prefix: Each operator is written after its operand subexpressions.
Infix: Each operator is written between its operand subexpressions.
Postfix notation: Each operator is written before its operand subexpressions.

Expressions (as strings) are very smart.

Computer Scientists know two alternatives to infix notation for writing
expression in postfix form.

are used by recursive evaluation of an expression tree, you will write the
3. If you write the constants and variables, and the operations, in the order they

successes to store and retrieve subexpression values.

A stack of subexpression values (store by push, access and removed by pop)

expression.

exactly matches the order of data access when the computer evaluates the

I. The left-to-right order of atoms (constants and variables) and operators

What's cool about Postfix:

What kind of tree traversal is used to EVALUATE an expression tree?

University at Albany Computer Science Dept.
Suppose we write down the constants and operations in the order that we use them when we evaluate this expression. First we take 6, then add them, and then 9, and then add them. Next we write 3, then divide it by 6, then add 9. So, we first write: 6 + 9. Next we write 6 + 9 / 3 + 6 * 9 - 4. Here's the result: The expression tree:  

\[( (4 - 6) \times (3 / (6 + 9)) ) \]
We can verify the result is 10 by calculating directly from the expression tree.

We can also verify this from the expression by using elementary school methods:
Example of an expression and its Parse Tree

(A = (B = ((C - (D * (E++))) + (F * G))))

Details continued on the next 2 frames.
The top level operator is multiplication (*).

Diagram:

```
  (F*G)
   ^
   *

F

G
```

Identifier

Identifier
The top-level operator in the expression $(C-(D*(E++)))$ is subtraction. The identifier $E$ is incremented at the next level, $(E++)$. The operator $*$ is a multiplication operator at the next level, $D*(E++)$. The operator $-$ is a subtraction operator at the lowest level, $(C-(D*(E++)))$. The diagram visually represents the structure and precedence of the operators in the expression.
The following C/C++ expression has an expression tree that is binary but not complete binary.
while (*(S++) = *(D++)) { }

Another C++ expression expresses a loop which copies C-strings:
question. (M/S confuse this with taxonomy trees).

(Binary) Decision Trees: Each leaf is an answer, each non-leaf is a yes-no

(it is like a telephone book).

Other Name Space Trees: EC, the Domain Name System of the Internet

directory names, plus a file name.

File Name Trees: Express a system to identify files using a sequence of

search for "human":


expression string, web document, program, etc.

Expression Trees: express the structure of the computation expressed by an

Tree Examples/Applications
O (heigh) operations.

number can be moved to the root with the tree remaining heap-ordered, using

Heap qualities: (1) The largest number is in the root. (2) The next-largest

For the tree and each subtree £, the root contains the largest of the numbers in

Heap ordered £ (Heap): A tree (or numbers) with the heap property:

su btree of £ is < the number in the root.

the left subtree of £ is > the number in the root and every number in the left

every number in the right subtree of £ is > the number in the root and every number in the right.

Binary Search Tree Property: For the tree and each subtree £, every number in £, each subtree of £, the largest number in £, not the question. (Only each u in stored in each node, not the question.

Is it greater? "If number is in a finite set, by using questions of the form: "Is it to 0."

and

Binary Search Tree: A decision tree for answering whether or not a given

Trees used for searching and sorting (different problems, different arrangements

University at Albany Computer Science Dept.
courses prove theoretical results about sorting algorithms.

This tree has \( N \) leaves; it is used in Graduate CS

combination of outcomes of comparisons possibly made by that algorithm

\begin{itemize}
\item Sorting-
\item State tree of a sorting algorithm: One node for EVERY
\item Game Trees: One node for EVERY legal combination of moves by the player(s)
\item More Conceptual Kinds of trees:
\end{itemize}
Implicit, helpful way for people to understand the structure: Each node

Explicit, implemented by a data structure: Each node of the tree is an

In various applications, the tree might occur in two ways:

knowledge you get from looking at questions of the form (C) processes in a search, etc.
An expression

Each arc expresses the structural relation between the root node and the
subtrees.

(c) One arc from this tree's root to
the root of each of the trees specified

and

(d) One arc from other root to the root
common with each other root.

its subtrees, with no nodes or arcs in
or more rooted trees called

(b) Either is an identifier or constant,
or has a top level operator, exclusive
or operator and operands under

(c) (and

or more expressions as operands.)

and

(b) Either is an identifier or constant,
or has a top level operator, exclusive
or operator and operands under

(c) (and

or more expressions as operands.)

and

(b) Either is an identifier or constant,
or has a top level operator, exclusive
or operator and operands under

(c) (and

or more expressions as operands.)

and

(b) Either is an identifier or constant,
the subexpression values until they are used by the recursive evaluator.

3. Solution to expression problem (2): The stack of activation records stores order.

Recursive evaluation easily finds and executes all the operations in the right

2. The top level operation is in the root of the tree.

1. The expression tree directly reveals the order of operations.

Remember about expression trees:
root.

and keep moving upward to each node's parent, you will eventually reach the

one), then move again to that node's parent's parent (provided there is one).

If you start at any node and move to the node's parent (provided there is

Each node except for the root has exactly one parent; the root has no parent.

Each node may be [is] associated with [zero] or more different nodes, called

say that „d is c's parent.”

is that node's children. If a node c is the child of another node d, then we

There is one special node, called the root.

If the [the set of] nodes is empty, then it is called the empty tree. But logically, “or” means “and”, “and” means “or”.

A tree is a finite set of „nodes”. The set might be empty (no nodes, which is

Main/Secondary definition:
{ 
    halt();
    count >>= count(temp >> is the root
    !
    
    while (temp IS NOT the root
    
    temp = u;
    node temp;
    }

find-root(node u) {

The following algorithm, started at any node, always halts, eventually:

A more formal description of property 4:

University at Albany Computer Science Dept.
The no-cycle property of trees: For every node \( n \), the \( \text{find-root}(n) \) algorithm halts when it is started on \( n \).

```c

return \text{find-root}(\text{parent}(n))
else
{
return n;
return cout >> n >> " is the root." >> end;
if (n is the root)
}
node \text{find-root}(node n)

Recursive root-finder
```
The purpose of clause 4. of the definition and not clause 4. of the relationship which satisfies clauses 1–3.

This is an example of a parent–child relationship where nodes 3, 2, 1, 4, 3, etc., are the unique root of a tree to exclude situations like nodes 1, 2, 3, 4, etc., where 2 is the parent of 3, 1 is the parent of 2, 4 is the parent of 3, and 3 is the parent of 1, and 3 is the parent of 4.

The unique root is 5.
is c’s parent.

$d$ is left child and its right child. If a node $c$ is the child of node $d$, we say “$c$ is the child of node $d$.”

2. Each node may be associated with up to two other different nodes, called its

Replace M/S tree definition clause (2) by:

(Left and Right)

Binary Tree (in brief, each node’s children are distinguished from one another as
Complete Binary Tree: Every non-leaf has exactly 2 children.

Full Binary Tree: Every leaf has the same depth.

Leaves:

Depth (aka Height) of a WHOLE TREE: Maximum depth of any of its leaves.

Depth (root) = zero.

Depth of a node: Number of parent-to-child steps from the root to this node.

Left and Right subtrees of a node:

(descendants of this new root's descendants).

Subtree (we can view any non-root node as the root of a new, smaller tree, Parent, Sibling, Ancestor, Descendant.

NO CHILDREN.

Computer Sci. examinations and job placements interviews): leaf: A node with
Recursive algorithm to compute the depth of a node:

```c
int depth(node n) {
    if (n is the root) return 0;
    else return 1 + depth(parent(n));
}
```
{ 
    { 
        count step from u to its child
        return mxer + 1;
    }

    mxer = max(mxer, height(c));
}

for each of the children nodes c of u
    if mxer = 0;
else
    if u is a leaf (return 0;

    int height (node u)
Worst case.

Use an amount of computer work proportional to the height of the tree, in the
insertion into a heap,
and
search and insertion in a binary search tree,
Main conclusion:
into a heap ordered tree.
We then viewed Main/Savitch's page that described how to insert a new number
search tree.
example of how a dictionary of states (of the USA) is implemented by a binary
We viewed Main's slide set 10b on the dictionary abstract data type and the
search and insert a new number into it.
We viewed Main/Savitch's binary search tree (Ch. 10) and dissected how to
decision tree (Ch. 10).

University at Albany Computer Science Dept.