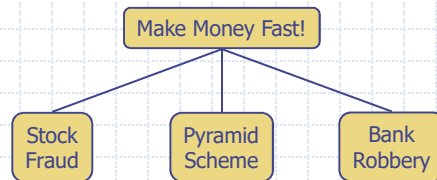


Trees



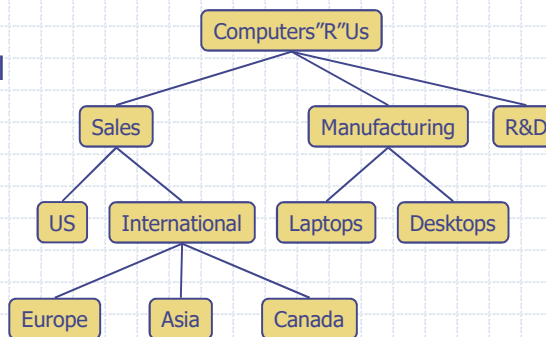
4/21/2017 12:47 PM

Trees

1

What is a Tree

- ◆ In computer science, a tree is an abstract model of a hierarchical structure
- ◆ Captures hierarchy
- ◆ Tree
 - A connected acyclic graph of nodes



4/21/2017 12:47 PM

Trees

2

What is a Tree

- ◆ Applications:
 - Organization charts
 - File systems
 - Genealogy (family tree)

4/21/2017 12:47 PM

Trees

3

Recursive Definition of a Tree

- ◆ A tree is a collection of nodes
 - One node is called the root
- ◆ A collection of nodes can be empty, otherwise a tree consists of a node called a root (R) and zero or more non-empty **subtrees** each of whose roots are connected by an edge to the root

4/21/2017 12:47 PM

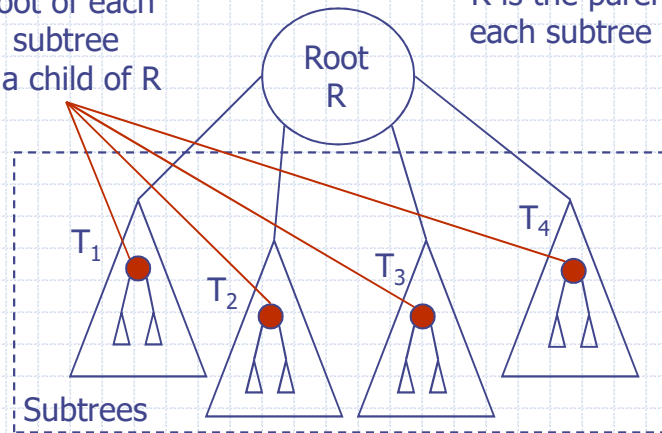
Trees

4

Recursive Definition of a Tree

Root of each subtree is a child of R

R is the parent of each subtree root



4/21/2017 12:47 PM

Trees

5

Tree - Nodes and Edges

- ◆ A tree has N nodes and $N-1$ edges because each edge connects some node to its parent and every node except the root has exactly one parent

4/21/2017 12:47 PM

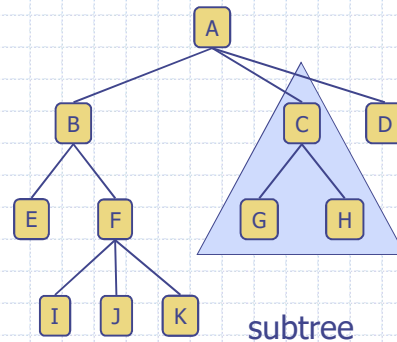
Trees

6

Tree Terminology

- ◆ **Root:** node without parent (A)
- ◆ **Internal node:**
 - node with at least one child
 - (A, B, C, F)
- ◆ **External node (or leaf):**
 - node without children
 - (E, I, J, K, G, H, D)
- ◆ **Ancestors of a node:**
 - parent,
 - grandparent,
 - grand-grandparent, etc.

- ◆ **Subtree:** tree consisting of a node and its descendants



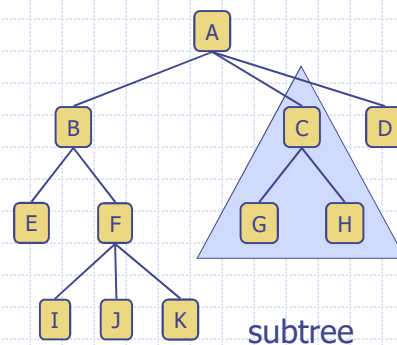
4/21/2017 12:47 PM

Trees

7

Tree Terminology

- ◆ **Descendant of a node:**
 - child,
 - grandchild,
 - grand-grandchild, etc.
- ◆ **Siblings:**
 - nodes with the same parent



4/21/2017 12:47 PM

Trees

8

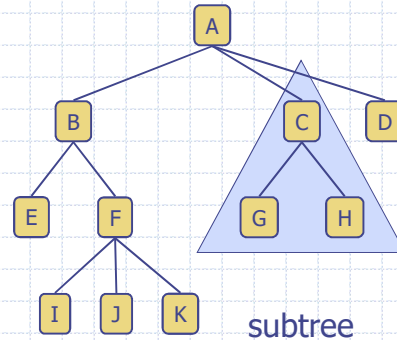
Tree Terminology

◆ Depth of a node:

- number of ancestors

◆ Height of root:

- maximum depth of any node.
- The height of a tree is the number of edges on the longest path from the root to a leaf
- Can similarly find the height of any node



4/21/2017 12:47 PM

Trees

9

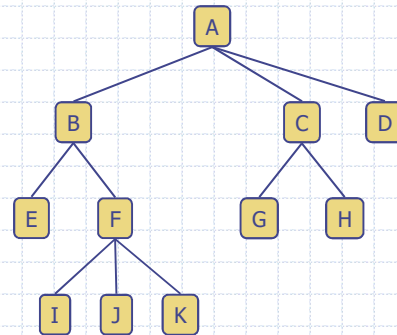
Tree Terminology

◆ Path - a sequence of nodes n_1, n_2, \dots, n_k such that n_i is the parent of n_{i+1} for $1 \leq i \leq k$

◆ Path length - the number of edges on the path ($k-1$)

◆ There is a path of length zero from every node to itself, i.e. shortest path.

◆ There is exactly one path from the root to each node (acyclic)



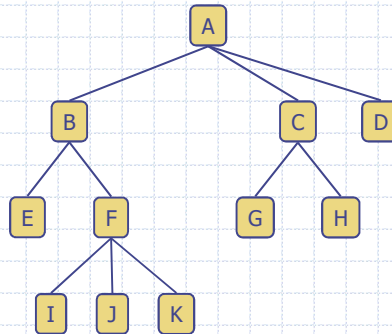
4/21/2017 12:47 PM

Trees

10

Tree Terminology

- ◆ If there is a path from n_1 to n_2 then n_1 is an ancestor of n_2 and n_2 is a descendant of n_1



4/21/2017 12:47 PM

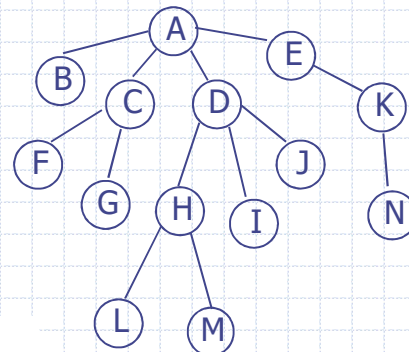
Trees

11

Tree Terminology

- ◆ In class exercise - what is/are the...

- Root -
- Leaves -
- Height of H -
- Depth of H -
- Ancestors of H -
- Descendants of H -
- Path from A to M -
- Length of path from A to M -
- Internal nodes -

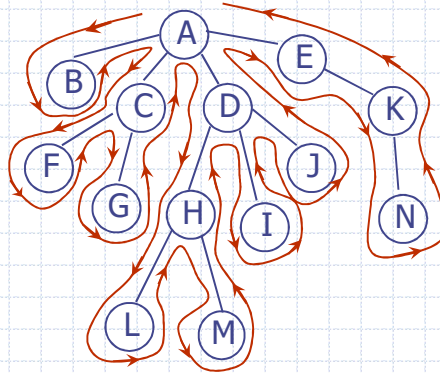


4/21/2017 12:47 PM

Trees

12

Tree Traversals



4/21/2017 12:47 PM

Trees

13

Tree Traversal

◆ **Definition:** a **traversal** of a tree T is a systematic way of accessing, or "visiting", all of the nodes in T .

4/21/2017 12:47 PM

Trees

14

Preorder Traversal

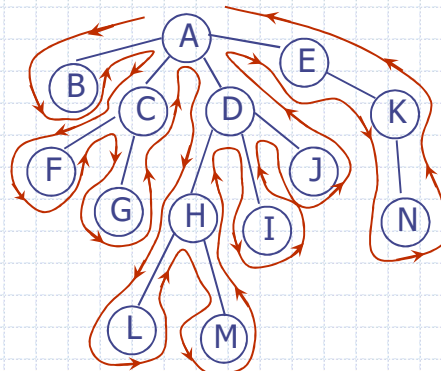
- ◆ In a **preorder traversal**, a node is visited before its descendants
- ◆ The **preorder listing** of the nodes of T is the root n of T followed by the nodes of T_1 in **preorder**, then the nodes of T_2 in **preorder**, and so on up to the nodes of T_k in **preorder**.
- ◆ **Intuition**: List the node the first time it is passed

4/21/2017 12:47 PM

Trees

15

Preorder Traversal



ABCFGDHLMIJEKN

4/21/2017 12:47 PM

Trees

16

Inorder Traversal

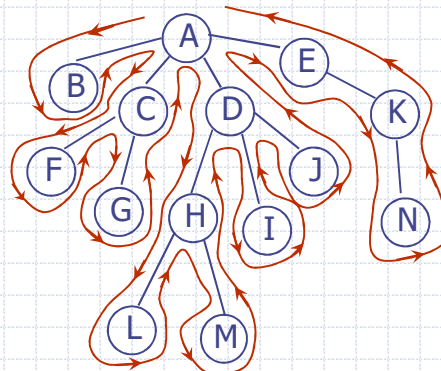
- ◆ The **inorder** listing of the nodes of T are the nodes of T_1 **inorder**, followed by the node n , followed by the nodes of T_2, \dots, T_k each group of nodes in **inorder**.
- ◆ **Intuition:** List the node the second time it is passed

4/21/2017 12:47 PM

Trees

17

Inorder Traversal



BAF CGLHMDI JNKE

4/21/2017 12:47 PM

Trees

18

Postorder Traversal

- ◆ The **postorder** listing of the nodes of **T** is the nodes of T_1 in **postorder** then the nodes of T_2 in **postorder** and so on up to T_k all followed by the node **n**
- ◆ **Intuition:** List the node the last time it is passed

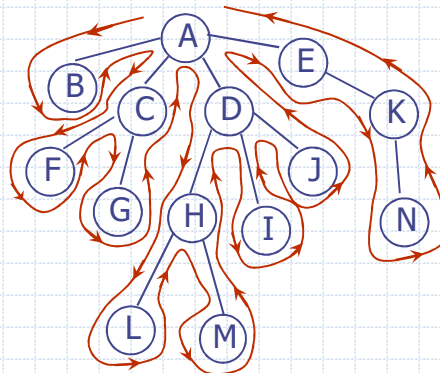
4/21/2017 12:47 PM

Trees

19

Postorder Traversal

In class
exercise



4/21/2017 12:47 PM

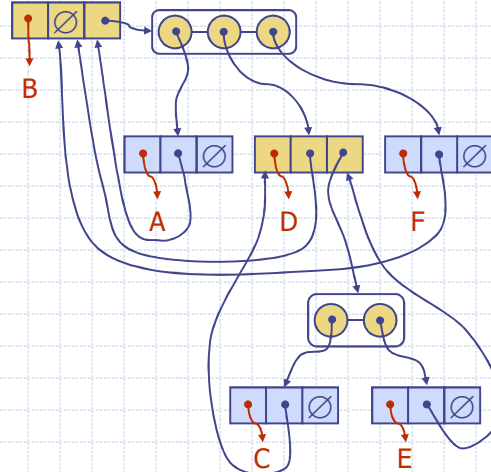
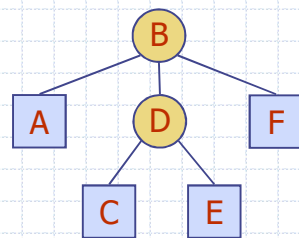
Trees

20

Data Structure for Trees

- ◆ A node is represented by an object storing

- Element
- Parent node
- Sequence of children nodes



4/21/2017 12:47 PM

Trees

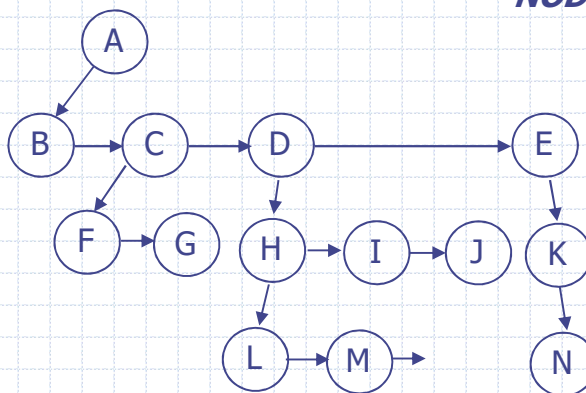
21

Data Structure for Trees

- ◆ First child/next sibling implementation

NODE class

itemtype item
Node firstChild*
Node nextSibling*



4/21/2017 12:47 PM

Trees

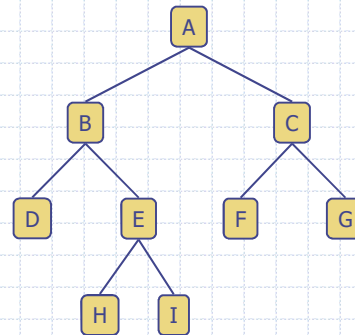
22

Binary Tree

- ◆ A **binary tree** is a tree with the following properties:
 - Each internal node has at most two children
- ◆ We call the children of an internal node left child and right child
- ◆ Alternative recursive definition: a binary tree is either
 - a tree consisting of a single node, or
 - a tree whose root has an ordered pair of children, each of which is a binary tree

Applications:

- arithmetic expressions
- decision processes
- searching



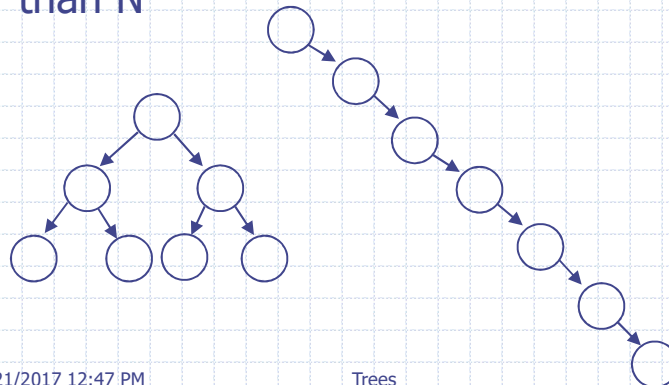
4/21/2017 12:47 PM

Trees

23

Binary Tree

- ◆ **Property** - the depth of an average binary tree is usually considered smaller than N



4/21/2017 12:47 PM

Trees

24

Binary Tree ADT

- ◆ Can keep direct pointers to children since we know how many there will be

```
class Node
    itemtype item
    Node* left
    Node* right
    Node* parent
```

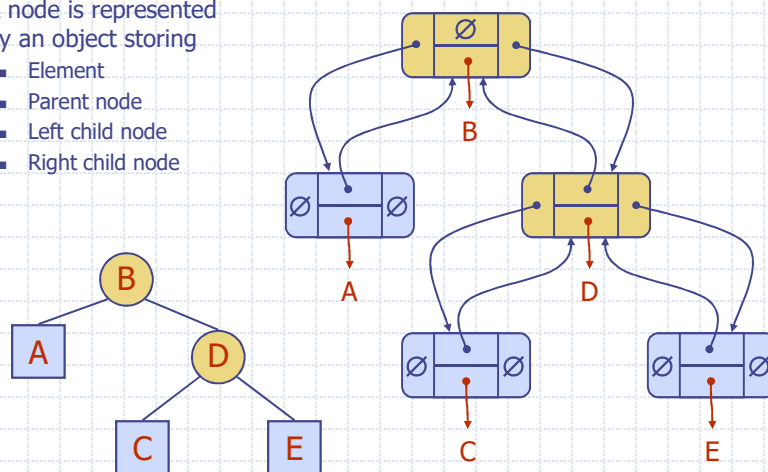
4/21/2017 12:47 PM

Trees

25

Data Structure for Binary Trees

- ◆ A node is represented by an object storing
 - Element
 - Parent node
 - Left child node
 - Right child node



4/21/2017 12:47 PM

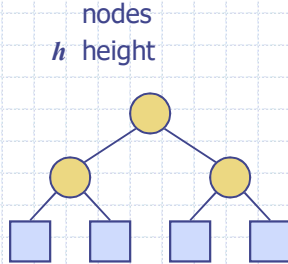
Trees

26

Properties of Proper Binary Trees

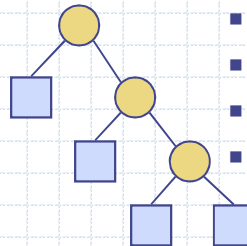
◆ Notation

- n number of nodes
- e number of external nodes
- i number of internal nodes
- h height



◆ Properties:

- $e = i + 1$
- $n = 2e - 1$
- $h \leq i$
- $h \leq (n - 1)/2$
- $e \leq 2^h$
- $h \geq \log_2 e$
- $h \geq \log_2 (n + 1) - 1$



4/21/2017 12:47 PM

Trees

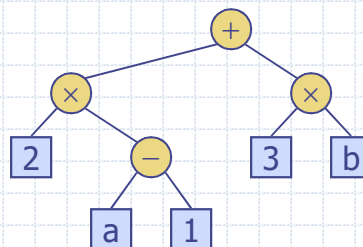
27

Arithmetic Expression Tree

◆ Binary tree associated with an arithmetic expression

- internal nodes: operators
- external nodes: operands

◆ Example: arithmetic expression tree for the expression $(2 \times (a - 1) + (3 \times b))$



4/21/2017 12:47 PM

Trees

28

Inorder Traversal

Recursive implementation of inorder traversal

```
void inorder(Node* nodePtr)
    if ( nodePtr )
        inorder (nodePtr->left)
        print node
        inorder (nodePtr->right)
```

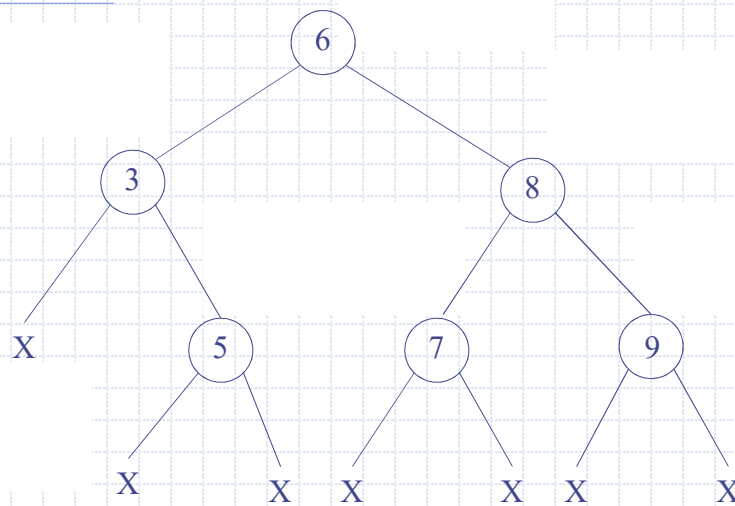
4/21/2017 12:47 PM

Trees

29

Inorder Traversal

3 5 6 7 8 9



4/21/2017 12:47 PM

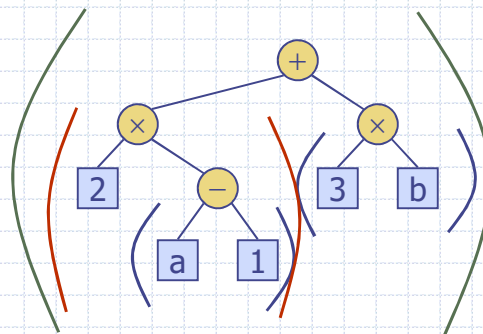
Trees

30

Arithmetic Expression Tree

- ◆ **Infix notation** - Inorder traversal and parenthesize each subtree

■ $((2 \times (a - 1)) + (3 \times b))$



```
void infix(Node* subRoot) {
    if ( subRoot ) {
        print "("
        infix ( subRoot->left )
        print subRoot->item
        infix ( subRoot->right )
        print ")"
    }
}
```

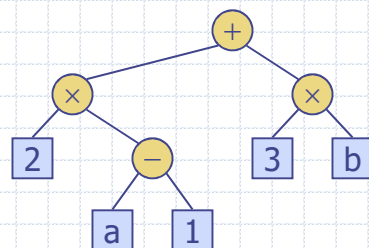
4/21/2017 12:47 PM

Trees

31

Arithmetic Expression Tree

- ◆ In class exercise - give **postfix notation** by doing **postorder traversal**



4/21/2017 12:47 PM

Trees

32

Evaluate Arithmetic Expressions

◆ Specialization of a postorder traversal

- recursive method returning the value of a subtree
- when visiting an internal node, combine the values of the subtrees

Algorithm *evalExpr(v)*

if *isExternal* (*v*)

return *v.element* ()

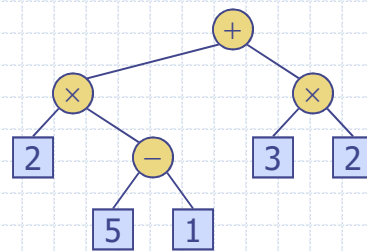
else

x \leftarrow *evalExpr*(*leftChild* (*v*))

y \leftarrow *evalExpr*(*rightChild* (*v*))

\diamond \leftarrow operator stored at *v*

return *x* \diamond *y*



4/21/2017 12:47 PM

Trees

33

Creating an Expression Tree

- ◆ Given an infix expression, use the stack based algorithm to convert infix to postfix
- ◆ Convert postfix expression to a tree

4/21/2017 12:47 PM

Trees

34

Creating an Expression Tree

◆ Algorithm

- Make a stack of node pointers
- Operands - push a new tree onto the stack
- Operators - pop two trees from the stack. Use the operator as the root of a new tree with the popped trees as children. Push a new tree onto the stack

4/21/2017 12:47 PM

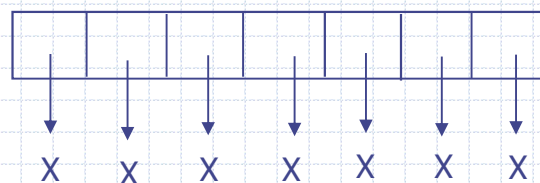
Trees

35

Creating an Expression Tree

$$(a + b) * (c * (d + e)) \rightarrow ab+cde+**$$

Stack of node pointers:



4/21/2017 12:47 PM

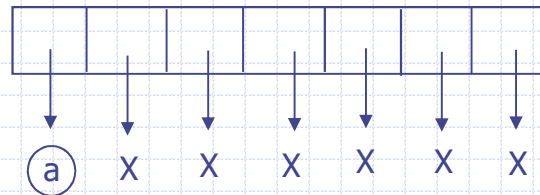
Trees

36

Creating an Expression Tree

$(a + b) * (c * (d + e)) \rightarrow ab+cde+**$

Stack of node pointers:



4/21/2017 12:47 PM

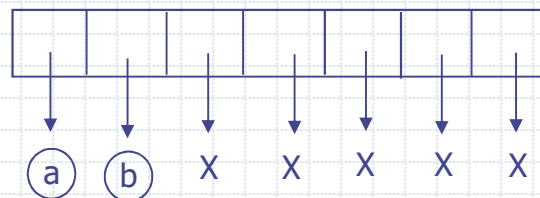
Trees

37

Creating an Expression Tree

$(a + b) * (c * (d + e)) \rightarrow ab+cde+**$

Stack of node pointers:



4/21/2017 12:47 PM

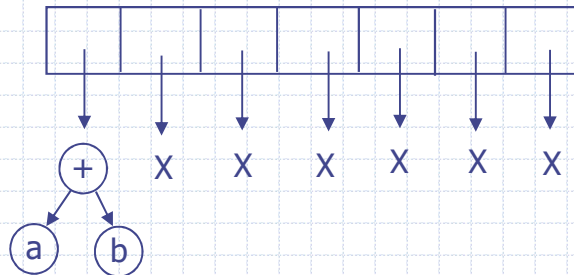
Trees

38

Creating an Expression Tree

$(a + b) * (c * (d + e)) \rightarrow ab+cde+**$

Stack of node pointers:



4/21/2017 12:47 PM

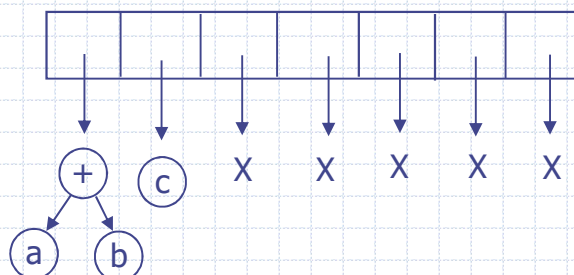
Trees

39

Creating an Expression Tree

$(a + b) * (c * (d + e)) \rightarrow ab+cde+**$

Stack of node pointers:



4/21/2017 12:47 PM

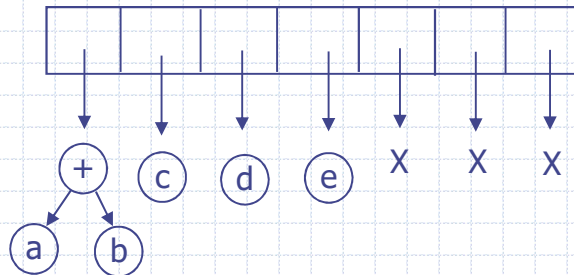
Trees

40

Creating an Expression Tree

$(a + b) * (c * (d + e)) \rightarrow ab+cde+**$

Stack of node pointers:



4/21/2017 12:47 PM

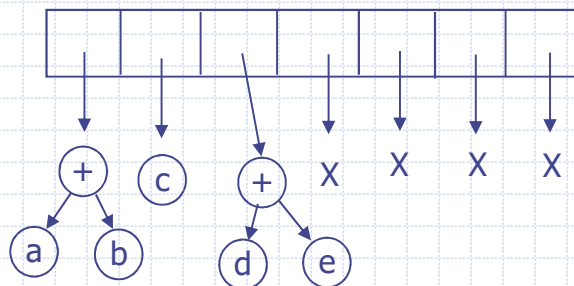
Trees

41

Creating an Expression Tree

$(a + b) * (c * (d + e)) \rightarrow ab+cde+**$

Stack of node pointers:



4/21/2017 12:47 PM

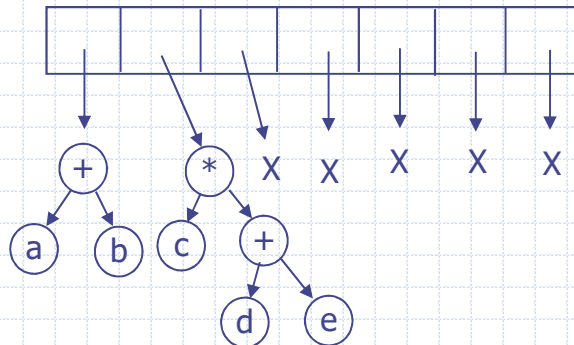
Trees

42

Creating an Expression Tree

$(a + b) * (c * (d + e)) \rightarrow ab+cde+**$

Stack of node pointers:



4/21/2017 12:47 PM

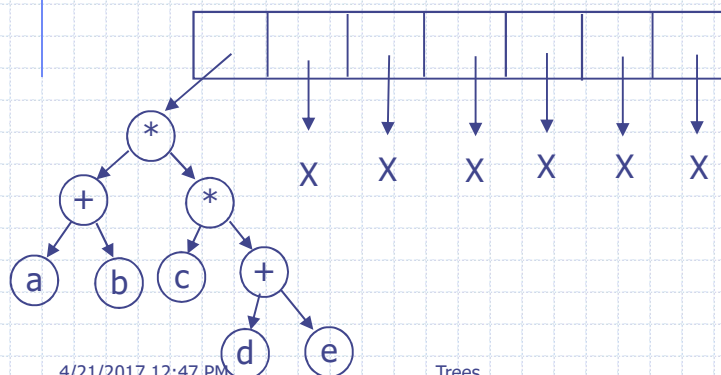
Trees

43

Creating an Expression Tree

$(a + b) * (c * (d + e)) \rightarrow ab+cde+**$

Stack of node pointers:



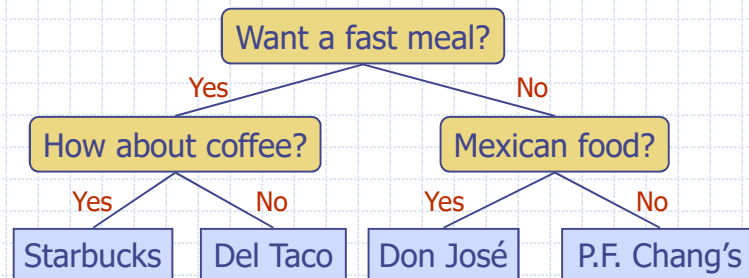
4/21/2017 12:47 PM

Trees

44

Decision Tree

- ◆ Binary tree associated with a decision process
 - internal nodes: questions with yes/no answer
 - external nodes: decisions
- ◆ Example: dining decision



4/21/2017 12:47 PM

Trees

45