Expressions and Expression Trees

CS2: Data Structures and Algorithms
Colorado State University

Russ Wakefield, Sudipto Ghosh and Wim Bohm

Infix Expressions

- **Infix** notation places each operator between two operands for binary operators:

```
A * x * x + B * x + C; // quadratic equation
```

- This is the customary way we write math formulas in most programming languages.
- However, we need to specify an order of evaluation in order to get the correct answer.

Evaluation Order

- The evaluation order you may have learned in math class is named PEMDAS:

Parentheses →
Exponentiation →
Multiplication, Division →
Addition, Subtraction

- Also need to account for unary, logical and relational operators, pre/post increment, etc.
- Java has a similar order of evaluation.

Reminder: Java Precedence

parentheses	( )
unary	++ -- + - ~ !
multiplicative	* / %
additive	+ -
shift	<< >>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary conditional	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Associativity

Operators with same precedence:

* /

and

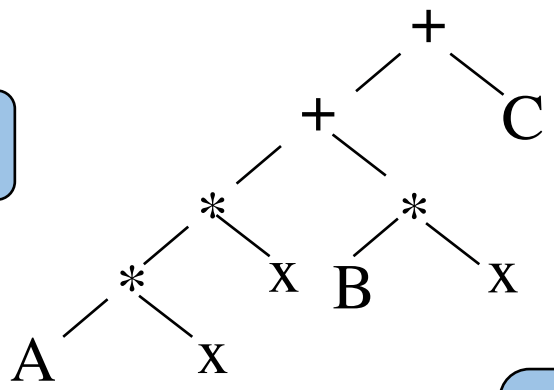
\+ -

are evaluated left to right: 2-3-4 = (2-3)-4

Expression Trees

- Parsing decomposes source code and builds a representation that represents its structure.
- Parsing generally results in a data structure such as a tree:

A * x * x + B * x + C

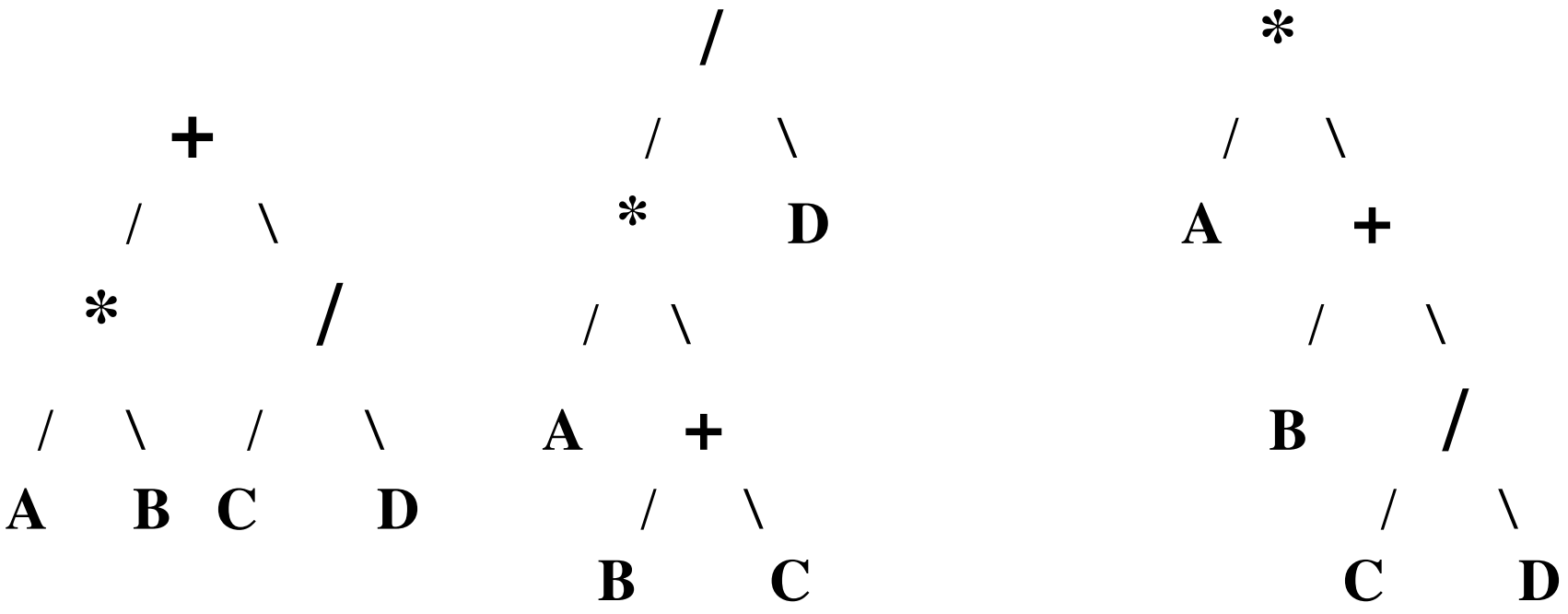


A x * x * B x * + C +
// postfix version

Infix, Postfix, Prefix Conversion

Infix	Postfix	Prefix	Notes
A * B + C / D	A B * C D / +	+ * A B / C D	multiply A and B, divide C by D, add the results
A * (B + C) / D	A B C + * D /	/ * A + B C D	add B and C, multiply by A, divide by D
A * (B + C / D)	A B C D / + *	* A + B / C D	divide C by D, add B, multiply by A

Expression Trees



Infix: **A*B + C/D** **A*(B+C)/D** **A*(B+C/D)**

Postfix: **A B * C D / +** **A B C + * D /** **A B C D / + ***

Evaluate Post Order Left to Right

Notice: the deeper in the tree, the higher the precedence

Evaluating expression trees

- By postfix traversal:
  - Internal node: operator
    - first evaluate children sub-trees
    - then evaluate the operator and return result
  - Leaf: operand
    - either identifier: produce current value
    - or constant: produce value
    - return result

Java support for trees?

- **Question:** Does the Java Collection framework have support for binary trees?

- **Answer:** No, you have to build your own trees using the same techniques as with linked lists.

Post Order Evaluation of an Integer Expression Tree

```
private Integer evalBin(String op, Integer left, Integer right){
        if(op.equals("+"))  return left + right;
        if(op.equals("-"))   return left - right;
        if(op.equals("*"))  return left * right;
        if(op.equals("/"))   return left / right;
        else return null;
 }
public int postorderEval(TreeNode node){
       String token = node.getItem();
       if( isOperator(token)){  //internal node
         Integer left  = postorderEval(node.getLeft());
         Integer right = postorderEval(node.getRight());
         return evalBin(token, left, right);
       } else  // leafs are int literals here
            return Integer.parseInt(token);
 }
```

Post order evaluation of 3*4+10-2