CX1107 Data Structures and Algorithms

Advanced Data Structure

College of Engineering

School of Computer Science and Engineering

Dr. Loke Yuan Ren
Lecturer
yrloke@ntu.edu.sg

Algorithm Design Strategies

A general approach to solving problems algorithmically that is applicable to a variety of problems from different areas of computing

- Brute Force and Exhaustive Search
- Divide-and-Conquer
- Greedy Strategy
- …etc.
- Decrease-and-Conquer
- Transform-and-Conquer
- Iterative Improvement

Transform-and-Conquer: Algebraic Expressions

$$a + b \times c - d \times e \div f \; =?$$

- $+, -, \times, \div$ are known as binary operator
- This expression is an infix expression which the operator is written between its operands.
  - Precedence rules: $\times, \div$ have higher precedence than $+, -$
  - Left-to-right association: Evaluate from left to right
- Without using parentheses, the evaluation is ambiguous
  - $((((a + b) \times c) - d) \times e) \div f$ or
  - $a + (b \times c) - ((d \times e) \div f)$
- Evaluation is tedious by using the infix expression
  - Multiple scanning is required to find the next operation
- How do our calculators work?

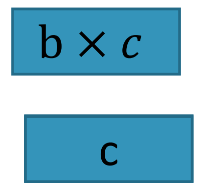
The expression is stored as a string "a+b*c"
How does computer interpret the string?

Transform-and-Conquer: Algebraic Expressions

$$a + b \times c - d \times e \div f$$

$$\Downarrow$$

$$abc \times +de \times f \div -$$



- Use a stack
- When the character is an operand, push it to the stack

- When the character is an operator, ‘×’, pop two operands from the stack
- Evaluate $\mathrm{b} \times c$
- Push the result of $\mathrm{b} \times c$ back to the stack etc.

Transform-and-Conquer: Algebraic Expressions

$$a + b \times c - d \times e \div f$$

$$\Downarrow$$

$$abc \times +de \times f \div -$$

- The expression is known as postfix expression a.k.a reverse Polish notation
- Reduce memory access and improve computational efficiency
- Under this convention, operators appears after its operands

  <operand> <operand> <operator>

Transform-and-Conquer: Algebraic Expressions

$$a + b \times c - d \times e \div f$$

$$\Downarrow$$

$$bc \times \quad de \times$$

- The expression is known as postfix expression a.k.a reverse Polish notation
- Reduce memory access and improve computational efficiency
- Under this convention, operators appears after its operands

  <operand> <operand> <operator>

Transform-and-Conquer: Algebraic Expressions

$$a + b \times c - \boxed{d \times e \div f}$$

$$\Big\downarrow$$

$$bc \times \quad \boxed{de \times f \div}$$

- The expression is known as postfix expression a.k.a reverse Polish notation
- Reduce memory access and improve computational efficiency
- Under this convention, operators appears after its operands

  <operand> <operand> <operator>

Transform-and-Conquer: Algebraic Expressions

$$\begin{tikzcd} \boxed{a + b \times c} - \boxed{d \times e \div f} \arrow[d] \\ \boxed{abc \times +}\ \boxed{de \times f \div}\ - \end{tikzcd}$$

- The expression is known as postfix expression a.k.a reverse Polish notation
- Reduce memory access and improve computational efficiency
- Under this convention, operators appears after its operands

  <operand> <operand> <operator>

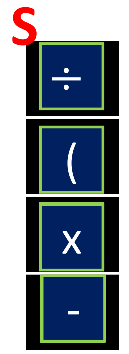
Transform-and-Conquer: Algebraic Expressions

Algorithm 1 Infix Expression to Postfix Expression

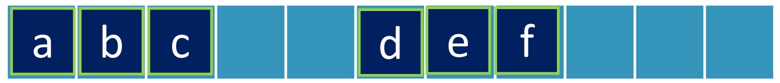
```
function IN2POST(String $infix$, String $postfix$)
    create a Stack $S$
    for each character $c$ in $infix$ do
        if $c$ is an operand then
            $postfix \leftarrow c$
        else if $c$ = ‘)’ then
            while peek($S$) $\neq$ ‘(’ do
                $postfix \leftarrow$ pop($S$)
            pop($S$)
        else if $c$ = ‘(’ then
            push($c$,$S$)
        else                                    ▷ $c$ is an operator or left parenthesis
            while $S \neq$ empty && peek($S$) != ‘(’ && precedence of peek($S$) $\geq$ precedence of $c$ do
                $postfix \leftarrow$ pop($S$)
            push($c$,$S$)
    while $S$ is not empty do
        $postfix \leftarrow$ pop($S$)
```

infix

$$a + b \times c - d \times (e \div f)$$



postfix

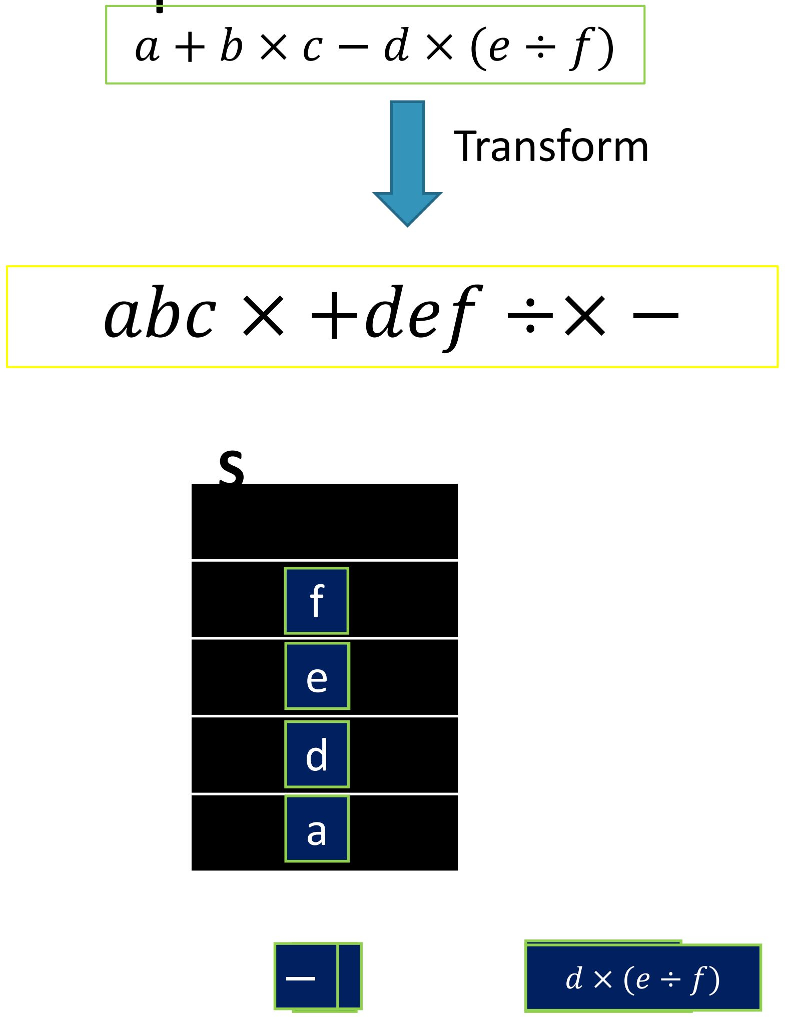


$$abc \times + def \div \times -$$

Transform-and-Conquer: Algebraic Expressions

Algorithm 2 Evaluation Postfix Expression

```
function ExePost(String $postfix$)
    create a Stack $S$
    for each character $c$ in $postfix$ do
        if $c$ is an operand then
            push($c$, $S$)
        else
            $operand1 \leftarrow$ pop($S$)
            $operand2 \leftarrow$ pop($S$)
            $result \leftarrow$ Evaluate($operand2$, $c$, $operand1$)
            push($result$, $S$)
```


$a + b \times c - d \times (e \div f)$
Transform
$abc \times + def \div \times -$
S
f
e
d
a
$-$
$d \times (e \div f)$

Transform-and-Conquer: Algebraic Expressions

$$\boxed{a + b \times c} - \boxed{d \times (e \div f)}$$

$$\Downarrow$$

$$- + a \times bc \times d \div ef$$

- The expression is known as prefix expression a.k.a Polish notation
- Under this convention, operators appears before its operands

  <operator> <operand> <operand>

Hint: Its algorithm is similar to postfix expression’s.

Summary

- An algorithm is not simply a computer program
- Algorithm Design Strategies
    - Transform-and-Conquer
        - Infix expression to Postfix expression
        - Tree Balancing

- Lectures focus on introduction to concepts
- Lab Sessions focus on practice and realization
- Assignments and Lab Tests are assessments