Stack

Hsuan-Tien Lin

Dept. of CSIE, NTU

March 31, 2020

Application: Expression Evaluation

Stack for Expression Evaluation

$$(a/b + c + b * e - a * c)$$

$$(a/b + c + b * e - a * c)$$

$$(a/b + c + b * e - a * c)$$

$$(a/b + c + b * e - a * c)$$

- precedence: $\{*,/\}$ first; $\{+,-\}$ later
- steps
 - f = a/b
 - g = f c
 - h = d * e

 - i = g + h
 - i = a * c
 - $\ell = i j$

ab/

Postfix Notation

same operand order, but put "operator" after needed operands

- —can "operate" immediately when seeing operator
- —no need to look beyond for precedence

Postfix from Infix (Usual) Notation

• infix:

parenthesize:

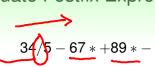
for every triple in parentheses, switch orders
$$\begin{pmatrix} (3/4)-5)+(6*7) \end{pmatrix}$$

$$(((34/)5-)(67*)+)(89*)-)$$

remove parentheses

difficult to parenthesize efficiently

Evaluate Postfix Expressions





- how to evaluate? left-to-right, "operate" when see operator
- 3, 4, $/ \Rightarrow 0.75$
- $0.75, 5, \Rightarrow -4.25$
- -4.25, 6, 7, $\star \Rightarrow$ -4.25, 42 (note: -4.25) stored for latter use)
- -4.25, 42, **←**)⇒ 37.75
- 37.75, 8, 9, * \Rightarrow 37.75, 72 (note: 37.75) stored for latter use)
- 37.75, 72, ⇒ ...

stored where?

stack so closest operands will be considered first!

14/19

Stack Solution to Postfix Evaluation

Postfix Evaluation

```
for each token in the input do

if token is a number

push token to the stack

else if token is an operator

sequentially pop operands a_{t-1}, \dots, a_0 from the stack

push token(a_0, a_1, a_{t-1}) to the stack

end if

end for

return the top of stack
```

matches closely with the definition of postfix notation



Application: Expression Parsing

at /, not sure of what to do (need later operands) so store

$$c + d * e - a * c$$

• at -, know that a / b can be a b / because - is of lower precedence

$$-c+d*e-a*c$$

 at +, know that ? - c can be ? c - because + is of same precedence but {-, +} is left-associative

$$a/b-c+d*e-a*c$$

at *, not sure of what to do (need later operands) so store

$$a/b-c+de=a*c$$

stored where? stack so closest operators will be considered first!

Stack Solution to Infix-Postfix Translation

```
for each token in the input do

if token is a number
output token
else if token is an operator
(while top of stack is of higher (or same) precedence do
pop and output top of stack
end while
push token to the stack
end if
end for
```

- here: infix to postfix with operator stack
 —closest operators will be considered first
- recall: postfix evaluation with operand stack
 —closest operands will be considered first
- mixing the two algorithms (say, use two stacks): simple calculator $2 \times 5 + 5566 (126)$

H.-T. Lin (NTU CSIE) Stack 18/19

Some More Hints on Infix-Postfix Translation

for each token in the input do
 if token is a number
 output token
 else if token is an operator
 while top of stack is of higher (or same) precedence do
 pop and output top of stack
 end while
 push token to the stack
 end if
end for

- for left associativity and binary operators
 - · right associativity? same precedence needs to wait
 - unary/trinary operator? same
- parentheses? higest priority
 - at '(', cannot pop anything from stack
 like seeing '*' while having '+' on the stack
 - at ')', can pop until '('—like parentheses matching

