# Stacks and Queues

Hsuan-Tien Lin

Dept. of CSIE, NTU

March 21–22, 2011

# Stacks (Sec. 3.1): Abstract Data Type

**Stack**

- object: a container that holds some elements
- action: push (to the top), pop (from the top)

- last-in-first-out (LIFO): 擠電梯， 洗盤子
- very restricted data structure, but important for computers —will discuss some cases later

# A Simple Application: Parentheses Balancing

- in C, the following characters show up in pairs: (), [], {}, ""

```
good: {xxx(xxxxxx)xxxxx"xxxx"x}
bad:  {xxx(xxxxxx}xxxxx"xxxx"x}
```

- the LISP programming language

```
(append (pow (* (+ 3 5) 2) 4) 3)
```

how can we check parentheses balancing?

# Stack Solution to Parentheses Balancing

**inner-most parentheses pair $\Longrightarrow$ top-most plate**

'(': 堆盤子上去 ; ')': 拿盤子下來

**Parentheses Balancing Algorithm**

```
for each c in the input do
    if c is a left character
        push c to the stack
    else if c is a right character
        pop d from the stack and check if match
    end if
end for
```

many more sophisticated use in compiler design

# System Stack

- recall: function call $\Leftrightarrow$ 拿新的草稿紙來算
- old (original) scrap paper: temporarily not used, 可以壓在下面

**System Stack: 一疊草稿紙, each paper (stack frame) contains**

- return address: where to return to the previous scrap paper
- local variables (including parameters): to be used for calculating within this function
- previous frame pointer: to be used when escaping from this function

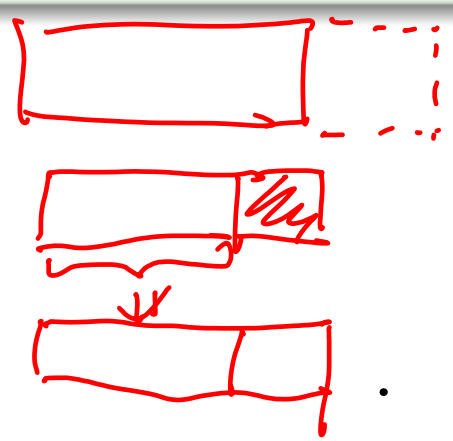
some related issues: stack overflow? security attack?

## Reading Assignment

be sure to go ask the TAs or me if you are still confused

# Stacks with Dynamically Growing Array (Sec. 3.2)

when stack full, grow array by size $M$



- successful (direct) growth: constant time
- if unlucky, growth by copying: $O(capacity)$
- $M = 1$ or any constant: very conservative
  —worst case, $O(n^2)$ for $n$ pushes (why?)
- $M = capacity$:
  —growth when exceeding 1, 2, 4, 8, 16, ...
  —each growth takes time around 1, 2, 4, 8, 16, ...
  —when $n$ pushes with $n = 13$?

push : 13

grow : 1 + 2 + 4 + 8 = 15

  —$2^k < n \leq 2^{k+1}$, time $2^{k+1} - 1$ on growth and $n$ on pushes
  —$O(n)$ for $n$ pushes

# Stack for Expression Evaluation (Sec. 3.6)

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

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

a b /
f c −  ⇒ a b / c −
d e *
g h +  ⇒ a b / c − d e * +
a c *
i j −  ⇒ a b / c − d e * + a c * −

## 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

3 4 / 5 − 6 7 * + 8 9 * −

- infix: ~~0.95~~ ~~−4.25~~ ~~42~~ ~~39.55~~ ~~92~~ ?

  3 / 4 − 5 + 6 * 7 − 8 * 9

- parenthesize:

  ((((3 / 4) − 5) + (6 * 7)) − (8 * 9))

- for every triple in parentheses, switch orders

  ((((3 4 /) 5 −) (6 7 *) +) (8 9 *) −)

- remove parentheses

  3 4 / 5 − 6 7 * + 8 9 * −

difficult to parenthesize efficiently