Analysis Tools

Hsuan-Tien Lin

Dept. of CSIE, NTU

March 12, 2013

Properties of Good Programs

- meet requirements, correctness: basic
- clear usage document (external), readability (internal), etc.

Resource Usage (Performance)

- efficient use of computation resources (CPU, FPU, etc.)?
  time complexity
- efficient use of storage resources (memory, disk, etc.)?
  space complexity

Space Complexity of List Summing

List-Sum(float array *list*, integer length *n*)

$tempsum \leftarrow 0$
for $i \leftarrow 0$ to $n-1$ **do**
  $tempsum \leftarrow tempsum + list[i]$
end for
return *tempsum*

- array *list*: size of pointer, commonly 4
- integer *n*: commonly 4
- float *tempsum*: 4
- integer *i*: commonly 4
- float return place: 4

total space 20 (constant), does not depend on *n*

Space Complexity of Recursive List Summing

RECURSIVE-LIST-SUM(float array *list*, integer length *n*)

if $n = 0$
  return 0
else
  return $list[n] +$ RECURSIVE-LIST-SUM($list, n - 1$)
end if

- array *list*: size of pointer, commonly 4
- integer *n*: commonly 4
- float return place: 4

only 12, better than previous one? (NO, why?)

12n

Time Complexity of Matrix Addition

MATRIX-ADD
(integer matrix *a*, *b*, result integer matrix *c*, integer *rows*, *cols*)

for $i \leftarrow 0$ to $rows - 1$ **do**
  for $j \leftarrow 0$ to $cols - 1$ **do**
    $c[i][j] \leftarrow a[i][j] + b[i][j]$
  end for
end for

- inner for: $R = P \cdot cols + Q$
- total: $(S + R) \cdot rows + T$

$$P \cdot rows \cdot cols + (Q + S) \cdot rows + T$$

Rough Time Complexity of Matrix Addition

$P \cdot rows \cdot cols + (Q + S) \cdot rows + T$

P, Q, R, S, T hard to keep track and not matter much

MATRIX-ADD
(integer matrix a, b, result integer matrix c, integer $rows$, $cols$)

for $i \leftarrow 0$ to $rows - 1$ **do**
  for $j \leftarrow 0$ to $cols - 1$ **do**
    $c[i][j] \leftarrow a[i][j] + b[i][j]$
  end for
end for

- inner for: $R = P \cdot cols + Q = \Theta(cols)$
- total: $(S + R) \cdot rows + T = \Theta(\Theta(cols) \cdot rows)$

rough total: $\Theta(rows \cdot cols)$

Asymptotic Notations: One Way for Rough Total

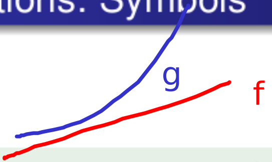
- goal: rough total rather than exact steps when input size large
- why rough total? constant not matter much

compare two complexity functions $f(n)$ and $g(n)$ when n large

growth of functions matters
—n^3 would eventually be bigger than $1000n$

- n^2 grows much faster than n
- n grows much slower than n^2, which grows much slower than 2^n
- $3n$ grows "slightly faster" than n
  —when constant not matter, $3n$ grows similarly to n

Asymptotic Notations: Symbols



- $f(n)$ grows slower than or similar to $g(n)$: $f(n) = O(g(n))$
- $f(n)$ grows faster than or similar to $g(n)$: $f(n) = \Omega(g(n))$
- $f(n)$ grows similar to $g(n)$: $f(n) = \Theta(g(n))$

- $n = O(n)$; $n = O(10n)$; $n = O(0.3n)$; $n = O(n^2)$; $n = O(n^5)$; $\cdots$ (note: $=$ more like "$\in$")
- $n = \Omega(n)$; $n = \Omega(0.2n)$; $n = \Omega(5n)$; $n = \Omega(\log n)$; $n = \Omega(\sqrt{n})$; $\cdots$
- $n = \Theta(n)$; $n = \Theta(0.1n + 4)$; $n = \Theta(7n)$; $n \neq \Theta(5^n)$

Asymptotic Notations: Definitions

- $f(n)$ grows slower than or similar to $g(n)$:

  $f(n) = O(g(n))$, iff exist c, n_0 such that $f(n) \le c \cdot g(n)$ for all $n \ge n_0$

- $f(n)$ grows faster than or similar to $g(n)$:

  $f(n) = \Omega(g(n))$, iff exist c, n_0 such that $f(n) \ge c \cdot g(n)$ for all $n \ge n_0$

- $f(n)$ grows similar to $g(n)$:

  $$f(n) = \Theta(g(n)), \text{ iff } f(n) = O(g(n)) \text{ and } f(n) = \Omega(g(n))$$