Program Analysis

Example: program analysis

Given the following program specification:

Input: $a \in \mathbf{Z}$, $b \in \mathbf{Z}^+$
Output: $a^b = a \cdot a \cdot \ldots \cdot a$ (i.e., a multiplied b times)

Consider the following implementations:

procedure *power*1(a, b)	procedure *power*3(a, b)
(1) if $b = 1$ then return a	(1) if $b = 1$ then return a
(2) else return $a \cdot$power1(a, b - 1)	(2) else r = power3(a, $\lfloor b/2 \rfloor$)
	(3) if b is odd then return $a \cdot r \cdot r$
	(4) else return $r \cdot r$

How many multiplications are done by each implementation when called with a, b?

Let M_K = # multiplications done by *power*K with input size n.

Consider power1

Recurrence relation: $m_1(1) = 0, m_1(n) = 1 + m_1(n-1)$ for $n > 1$

Solution guess: $m_1(n) = n - 1$

Prove this is the solution using induction:

P(n): $m_1(n) = n - 1$ show P(n) holds $\forall n \in \mathbb{N}^+$

Base case: when $n = b = 1$

$m_1(1) = 0$ from recurrence relation

$n - 1 = 1 - 1 = 0$ so $m_1(1) = 1 - 1$

Inductive step: $(m_1(k) = k - 1) \Rightarrow (m_1(k+1) = (k+1) - 1)$.

IH: $m_1(k) = k - 1$

Consider $m_1(k+1)$

$m_1(k+1) = 1 + m_1((k+1) - 1)$ from recurrence relation

$= 1 + m_1(k) = 1 + k - 1 = k = (k+1) - 1$

Consider power3

Best-case scenario: always do line (4), so consider $b = 2^k$ for $k \in \mathbb{N}$.

Worst-case scenario: always do line (3), want n odd, $\lfloor \frac{n}{2} \rfloor$ odd, $\lfloor \frac{n}{4} \rfloor$ odd.

Best-case scenario:

consider first only the case $b = 2^k$ for $k \in \mathbb{N}$

Then $m_3(1) = 0$, $m_3(n) = m_3(\frac{n}{2})$ for $n > 1$ (from line 4)

n	$m_3(n)$
1	0
2	$m_3(1)+1 = 1$
4	$m_3(2)+1 = 2$
8	$m_3(4)+1 = 3$
16	$m_3(8)+1 = 4$
⋮	
2^k	k

when $n = 2^k$, power3(a, n) does k multiplication, $n = 2^k \Rightarrow k = \log_2 n$,

so $m_3(n) = \log_2 n$ when n is power of 2 (proof by induction)

Worst-case scenario:

n is odd, $\lfloor \frac{n}{2} \rfloor$ is odd, $\lfloor \frac{n}{4} \rfloor$ is odd, ...

$m_3(1) = 0$, $m_3(n) = m_3(\lfloor \frac{n}{2} \rfloor) + 2$ for $n > 1$ (from line 3)

Suppose $n = 2^k - 1$ for $k \in \mathbb{N}^+$, n is odd.

Consider $\lfloor \frac{n}{2} \rfloor = \lfloor \frac{2^k - 1}{2} \rfloor = \lfloor 2^{k-1} - \frac{1}{2} \rfloor = 2^{k-1} - 1$

Now consider $m_3(n) = m_3(2^k - 1) = 2 + m_3(2^{k-1} - 1) = 2 + 2 + m_3(2^{k-2} - 1)$ ....

$= 2 \cdot (k-1)$

$= 2(\log_2(n+1) - 1)$ (since $n = 2^k - 1$, $n + 1 = 2^k$, $\log(n+1) = k$.)

So total # of multiplications in power3 is $\log_2 n \le m_3(n) \le 2(\log_2(n+1) - 1)$

Asymptotic Analysis

Goal: Capture "big picture" behavior of an algorithm so we can compare it to other algorithms.

What kind of behavior? # elementary operations ≈ running time

- what happens as input size (n) gets "big".
- don't want to have to worry about exact operation.
  counts on every possible output

→ don't worry about "small" details = constants coefficients, lower-order terms.

Definitions

$f: \mathbb{N} \to \mathbb{R}_{>0}$ → positive real numbers.

Given two functions *f* and *g*, from **N** to $\mathbf{R}_{>0}$, representing the running times of programs (i.e., $f(n)$ represents how many elementary operations a particular program does on input of size n), we define the following:

asymptotically equivalent $f(n) \approx g(n)$.

$f(n)$ and $g(n)$ are *asymptotically equivalent* if $(\forall\ \varepsilon \in \mathbf{R}_{>0})(\exists\ N \in \mathbf{N})(\forall\ n \geq N)\ 1 - \varepsilon \leq f(n) / g(n) \leq 1 + \varepsilon$

$1-\varepsilon \leq \frac{f(n)}{g(n)} \leq 1+\varepsilon$ for all $\varepsilon > 0$ and for all $n \geq n_0$ for some $n_0 \in \mathbb{N}$.

i.e, $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$

big Theta $f(n) \in \Theta(g(n))$

$f(n)$ is *Theta* of $g(n)$ if $(\exists\ c, d \in \mathbf{R}_{>0})(\exists\ N \in \mathbf{N})(\forall\ n \geq N)\ c \leq f(n) / g(n) \leq d$

$\exists\ c, d > 0$, s.t. $c \cdot g(n) \leq f(n) \leq d \cdot g(n)$ $\forall n \geq n_0$ for some $n_0 \in \mathbb{N}$.

i.e. we can use $g(n)$ to "bound" $f(n)$ on both sides

big Oh $f(n) \in O(g(n))$

$f(n)$ is *Oh* of $g(n)$ if $(\exists\ d \in \mathbf{R}_{>0})(\exists\ N \in \mathbf{N})(\forall\ n \geq N)\ f(n) / g(n) \leq d$

$\exists\ d > 0$ such that $f(n) \leq d \cdot g(n)$ $\forall n \geq n_0$ for some $n_0 \in \mathbb{N}$

i.e. can use $g(n)$ to put an upper bound on $f(n)$.

big Omega $f(n) \in \Omega(g(n))$.

$f(n)$ is *Omega* of $g(n)$ if $(\exists\ c \in \mathbf{R}_{>0})(\exists\ N \in \mathbf{N})(\forall\ n \geq N)\ c \leq f(n) / g(n)$

$\exists\ C > 0$ s.t. $c \cdot g(n) \leq f(n)$ $\forall n \geq n_0$ for some $n_0 \in \mathbb{N}$.

i.e. can use $g(n)$ to put a lower bound on $f(n)$.

Examples

Given the following functions:

$f(n) = n$
$g(n) = n^2$
$h(n) = 3n + 7$
$k(n) = n^2 + 4n + 20$

Is $f(n) \sim g(n)$?

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n}{n^2} = \lim_{n\to\infty} \frac{1}{n} = 0.$

what about $g(n) \sim f(n)$? $\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{n^2}{n} = \lim_{n\to\infty} n = \infty$

Is $f(n) \sim h(n)$?

$\lim_{n\to\infty} \frac{f(n)}{h(n)} = \lim_{n\to\infty} \frac{n}{3n+7} = \frac{1}{3} \neq 1$ No

Is $g(n) \sim k(n)$?

$\lim_{n\to\infty} \frac{g(n)}{k(n)} = \lim_{n\to\infty} \frac{n^2}{n^2+4n+20} = 1$ ✓

Is $f(n) = \Theta(h(n))$? To show this, find c, d, n_0 so that.

$c \cdot h(n) \leq f(n) \leq d \cdot h(n)$ for all $n \geq n_0$.

pick $n_0 = 1$ set $c = 10$, $d = 3$.
$c = \frac{1}{10}$, $d = \frac{1}{3}$
(not unique, could pick, say, $d = 1$)

$c \cdot (3n+7) \leq n \leq d(3n+7)$.

$c \leq \frac{n}{3n+7} \leq d$.

$\frac{1}{c} \geq \frac{3n+7}{n} \geq \frac{1}{d}$. (if $n \geq 1$).

$\frac{1}{c} \geq 3 + \frac{7}{n} \geq \frac{1}{d}$

Is $g(n) = \Theta(f(n))$? If so, then $\exists c, d > 0$ and $n_0 \in \mathbb{N}$ s.t. $c \cdot n \leq n^2 \leq d \cdot n$. $\forall n \geq n_0$.

So that means $n^2 \leq d \cdot n$ $\forall n \geq n_0$

In other words $n \leq d$ $\forall n \geq \max(n_0, 1)$, but $\nexists d$ s.t. $d \geq \forall n \in \mathbb{N}$, $n \geq \max(n_0, 1)$

Relationships between the definitions

$f(n) = \Theta(g(n)) \iff f(n) \in O(g(n)) \wedge f(n) \in \Omega(g(n))$.

$f(n) = \Theta(g(n)) \iff g(n) \in \Theta(f(n))$

$f(n) \sim g(n) \implies f(n) \in \Theta(g(n))$

but converse is not necessarily true. The contrapositive is true: $f(n) \notin \Theta(g(n)) \Rightarrow f(n) \not\sim g(n)$.

Asymptotic Analysis (continued)

Another look at asymptotic analysis

Given two functions f and g, from $\mathbf{N}$ to $\mathbf{R}_{>0}$, representing the running times of programs (i.e., $f(n)$ represents how many elementary operations a particular program does on input of size n), we define the following:

big Theta : $f(n)$ is $\Theta(g(n))$ if $(\exists\ c, d \in \mathbf{R}_{>0})(\exists\ N \in \mathbf{N})(\forall\ n \geq N)\ c \leq f(n) / g(n) \leq d$

$c \leq \lim_{n\to\infty} \frac{f(n)}{g(n)} \leq d$ where $c, d \in \mathbb{R}_{>0} \iff f(n) = \Theta(g(n))$

big Oh : $f(n)$ is $O(g(n))$ if $(\exists\ d \in \mathbf{R}_{>0})(\exists\ N \in \mathbf{N})(\forall\ n \geq N)\ f(n) / g(n) \leq d$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} \leq d$ where $d \in \mathbb{R}_{>0} \Rightarrow f(n) = O(g(n))$

big Omega : $f(n)$ is $\Omega(g(n))$ if $(\exists\ c \in \mathbf{R}_{>0})(\exists\ N \in \mathbf{N})(\forall\ n \geq N)\ c \leq f(n) / g(n)$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} \geq c$ where $c \in \mathbb{R}_{>0} \Rightarrow f(n) = \Omega(g(n))$

Additionally:

$\lim_{n\to\infty} f(n) / g(n) = 0 \implies$ $f(n) = O(g(n)) \wedge f(n) \neq \Omega(g(n))$

$\lim_{n\to\infty} f(n) / g(n) = \infty \implies$ $f(n) = \Omega(g(n)) \wedge f(n) \neq O(g(n))$

Examples

1. Show that $3n + 7$ is $\Omega(n)$ $\lim_{n\to\infty} \frac{3n+7}{n} = 3 \geq 3$ so $3n+7 = \Omega(n)$
2. Show that n is $\Omega(3n + 7)$ $\lim_{n\to\infty} \frac{n}{3n+7} = \frac{1}{3} \geq \frac{1}{3}$ so $n = \Omega(3n+7)$
3. Show that $3n^2 + 7$ is $\Omega(n)$ $\lim_{n\to\infty} \frac{3n^2+7}{n} = \infty$ since we found a lower bound > 0, $3n^2+7 = \Omega(n)$
4. Show that $15n^2 + 8n\log_2(n^2) + 10^9 n + 7$ is $O(n^2)$

$= 15n^2 + 16 n\log_2 n + 10^9 n + 7 = f(n)$

$\lim_{n\to\infty} \frac{f(n)}{n^2}$

$= \lim_{n\to\infty} \left(15 + 16\frac{\log_2 n}{n} + \frac{10^9}{n} + \frac{7}{n}\right)$ — just need to find an upper bound that is in $\mathbb{R}_{>0}$.

$\left(15 + 16\frac{\log_2 n}{n} + \frac{7}{n^2}\right) + \frac{10^9}{n}$

$\leq \frac{16^9}{2} + \frac{16^9}{2}$ for $n \geq 2$.

Properties of log (where $x, y > 0$ and $a, b > 1$)
$\log_b(x \cdot y) = \log_b x + \log_b y$
$\log_b(x^y) = y \cdot \log_b x$
$\log_a x = (\log_b x) / (\log_b a)$

5. Show that $15n^2 + 8n\log_2(n^2) + 10^9 n + 7$ is $\Omega(n^2)$

$\lim_{n\to\infty} 15 + 16\frac{\log_2 n}{n} + \frac{10^9}{n} - \frac{7}{n^2} \geq 15$. — just need to find a lower bound that a positive number

6. Show that $15n^2 + 8n\log_2(n^2) + 10^9 n + 7$ is $\Theta(n^2)$

since this is $O(n^2)$ and $\Omega(n^2)$, it is $\Theta(n^2)$

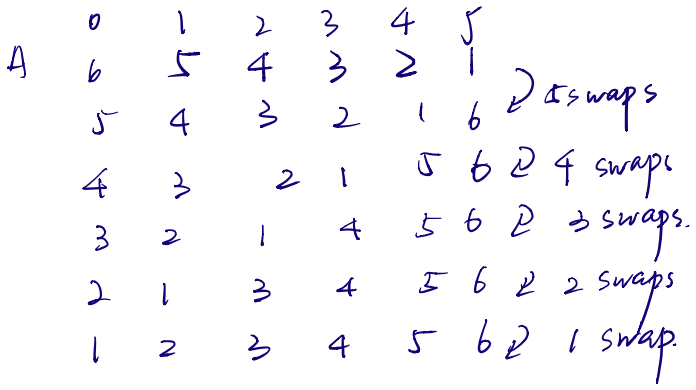
Program Analysis Revisited

Program analysis example

Consider the following program specification and the implementation known as bubblesort:

Input: An integer $n \geq 0$ and an array $A[0 .. n - 1]$ of integers.
Output: The array A where $A[0 .. n - 1]$ is sorted from smallest to largest.

```
    BubbleSort(n, A)
(1)   m ← n
(2)   while m > 1
(3)       k ← 0
(4)       i ← 0
(5)       while i < m - 1
(6)           if A[i] > A[i + 1]
(7)               swap A[i] and A[i + 1]
(8)               k ← i + 1
(9)           end (if)
(10)          i ← i + 1
(11)      end (inner while)
(12)      m ← k
(13)  end (outer while)
```



What is the input size of this program? n = # elts to sort.

What is the exact number of swaps done by the bubblesort implementation in the worst-case?

worst case: as many swaps as possible. → reverse order.

n elts to sort $\quad (n-1) + (n-2) + \cdots + 1 = \frac{n-1}{2} \cdot n$

proof of $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$ by induction.

Show that the worst-case number of swaps is $\Theta(n^2)$

$\lim_{n\to\infty} \frac{\frac{1}{2}(n^2-n)}{n^2} = \lim_{n\to\infty} \left(\frac{1}{2} - \frac{1}{2n}\right) = \frac{1}{2}$

Since $\frac{1}{2} \leq \frac{1}{2} \leq \frac{1}{2}$, $\frac{1}{2}(n^2-n) = \Theta(n^2)$