Analysis of time complexity

Biman Tang

UM-SJTU Joint Institute

September 27, 2012

CONTENT

1. Big-O Notation
   - Defination
   - Rules
   - Why disregard the constant?
2. Other Notations
3. Exercise

1. Big-O Notation
   - Defination
   - Rules
   - Why disregard the constant?
2. Other Notations
3. Exercise

Defination

- $f(n)$ and $g(n)$ are the running times of 2 algorithms on inputs of size n.
- If there is a constant $c > 0$ such that

$$f(n) \leq c \cdot g(n)$$

- We say

$$f = O(g)$$

Rules

- Multiplicative constants can be omitted,

$$14n^2 \to n^2$$

- n^a dominates n^b if $a > b$,

$$n^2 + n \to n^2$$

- Any exponential dominates any polynomial, 3^n dominates n^5
- Any polynomial dominates any logarithm, n dominates $(\log n)^3$

Why disregard the constant?

- $f_1(n) = n^2$, $f_2(n) = 2n + 20$,
- Which is better?
- It depends on n.
- When it comes to Big-O, it just depends on

$$\lim_{n \to \infty} \frac{f_2(n)}{f_1(n)}$$

  - 0, $f_2(n) = O(f_1(n))$, but $f_1(n) \neq O(f_1(n))$
  - constant, $f_2(n) = O(f_1(n))$, as well as $f_1(n) = O(f_1(n))$
  - ∞, $f_1(n) = O(f_2(n))$, but $f_2(n) \neq O(f_1(n))$

1 Big-O Notation

2 Other Notations

3 Exercise

Other Notations

- $f = \Omega(g) \to g = O(f)$
- $f = \Theta(g) \to g = O(f),\ f = O(g)$
- eg. $f_1 = n^2 + 2,\ f_2 = 2n + 1,\ f_3 = n + 1$
  - $f_2 = O(f_1),\ f_3 = O(f_1),\ f_2 = O(f_3),\ f_3 = O(f_2)$
  - $f_1 = \Omega(f_2),\ f_1 = \Omega(f_3),\ f_2 = \Omega(f_3),\ f_3 = \Omega(f_2)$
  - $f_2 = \Theta(f_3),\ f_3 = \Theta(f_2)$

Examples

- Polynomial, $O(1)$

```
    a = b * c + 5 + d * d;
```

- *if* statament, $O(1)$

```
if (a > 0)
    b++;
```

- forloop, $O(n)$

```
for (i = 0; i < n; i++)
    a += i;
```

Examples (cont'd)

- Nested forloop, $O(n^2)$

```
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
        a[i][j] = i * j;
```

```
for (i = 0; i < n; i++)
    for (j = 0; j < i; j++)
        a[i][j] = i * j;
```

- Nested forloop, $O(n^3)$

```
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
        for (k = 0; k < n; k++)
            a[i][j][k] = i * j * k;
```

Examples (cont'd)

- forloop + *if*, $O(n)$

```
for (i = 0; i < n; i++)
    if (i / 3 == 0)
        a += i;
```

```
for (i = 0; i < n; i++)
    if (i / 3 == 0)
        a += i;
    else
        a += i + 1;
```

- 2 forloops, $O(n)$

```
for (i = 0; i < n; i++)
    a[i] = 0;
for (i = 0; i < n; i++)
    b[i] = 0;
```

1 Big-O Notation

2 Other Notations

3 Exercise

Exercise

In each of the following situations, indicate whether $f = O(g)$, or $f = \Omega(g)$, or both (in which case $f = \Theta(g)$).

PROB.

$$f(n) = n - 100$$

$$g(n) = n - 200$$

SOL.

$$f = \Theta(g)$$

Exercise (cont'd)

PROB.

$$f(n) = n^{\frac{1}{2}}$$

$$g(n) = n^{\frac{2}{3}}$$

SOL.

$$f = O(g)$$

Exercise (cont'd)

PROB.

$$f(n) = 100n + \log n$$

$$g(n) = n + (\log n)^2$$

SOL.

$$f = \Theta(g)$$

Exercise (cont'd)

PROB.

$$f(n) = n \log n$$

$$g(n) = 10n \log(10n)$$

SOL.

$$f = \Theta(g)$$

Exercise (cont'd)

PROB.

$$f(n) = n^{1.01}$$

$$g(n) = n \log^2 n$$

SOL.

$$f = \Omega(g)$$

Exercise (cont'd)

PROB.

$$f(n) = n^{\frac{1}{2}}$$

$$g(n) = 5^{\log_2 n}$$

SOL.

$$f = O(g)$$

Exercise (cont'd)

PROB.

$$f(n) = n2^n$$

$$g(n) = 3^n$$

SOL.

$$f = O(g)$$

Exercise (cont'd)

PROB.

$$f(n) = n!$$

$$g(n) = 2^n$$

SOL.

$$f = \Omega(g)$$

Exercise (cont'd)

Consider that

$$g(n) = 1 + c + c^2 + \cdots + c^n$$

when $\Theta(1)$, $\Theta(n)$, $\Theta(c^n)$?

Since we have

$$g(n) = 1 + c + c^2 + \cdots + c^n = \frac{1 - c^n}{1 - c}$$

for $c < 1$,

$$\lim_{n \to \infty} \frac{g(n)}{1} = \lim_{n \to \infty} \frac{1 - c^n}{1 - c} = \frac{1}{1 - c}$$

so

$$g(n) = \Theta(1)$$

Exercise (cont'd)

Consider that

$$g(n) = 1 + c + c^2 + \cdots + c^n$$

when $\Theta(1)$, $\Theta(n)$, $\Theta(c^n)$?

Since we have

$$g(n) = 1 + c + c^2 + \cdots + c^n = \frac{1 - c^n}{1 - c}$$

for $c = 1$,

$$g(n) = n$$

so

$$g(n) = \Theta(n)$$

Exercise (cont'd)

Consider that

$$g(n) = 1 + c + c^2 + \cdots + c^n$$

when $\Theta(1)$, $\Theta(n)$, $\Theta(c^n)$?

Since we have

$$g(n) = 1 + c + c^2 + \cdots + c^n = \frac{1 - c^n}{1 - c}$$

For $c > 1$,

$$\lim_{n\to\infty} \frac{g(n)}{c^n} = \lim_{n\to\infty} \frac{1 - c^n}{(1 - c)c^n} = \frac{1}{c - 1}$$

so

$$g(n) = \Theta(c^n)$$