CS483-02 Asymptotic Notations for Algorithm Analysis

Instructor: Fei Li

Room 443 ST II

Office hours: Tue. & Thur. 4:30pm - 5:30pm or by appointments

`lifei@cs.gmu.edu` with subject: CS483

`http://www.cs.gmu.edu/~ lifei/teaching/cs483_fall07/`

This lecture note is based on Dr. J.M. Lien's lecture notes.

Outline

- Review
- Time efficiency
- Worst-case, best-case, and average case
- Order of growth
- O, Ω and Θ
- Examples and exercises

Review

In last class

- What is an algorithm?
  - Definition and properties
- Why do we study algorithms?
  - Theoretical importance and practical importance
- There may exist multiple algorithms for the same problem. (Refer to "Greatest Common Divisor" Problem.)

Review

In last class

- ➢ What is an algorithm?
  - Its definition and properties
- ➢ Why do we study algorithms?
  - Theoretical importance and practical importance
- ➢ There may exist multiple algorithms for the same problem. (Refer to "Greatest Common Divisor" Problem.)

In this class

- ➢ ➔ Given an algorithm, how do we analyze it?
  - Is the algorithm correct?
  - What is its running time?

    (similar for space efficiency analysis)

Example: Greatest Common Divisor $gcd(m, n)$

3 algorithms calculating $gcd(m, n)$. Assume $m > n > 0$.

1. From $gcd(m, n)$'s definition: check each number in decreasing order by 1 starting from $\min\{m, n\}$.

   $n, n-1, n-2, \ldots$

2. From $gcd(m, n)$'s definition and prime numbers definition: $gcd(m, n)$ is the product of all common factors of m and n.
   - List all prime numbers $\leq n$. (Sieve method)
   - Prime number factorization of m and n.
   - Multiply all the common factors.

3. Euclid's algorithm: $gcd(m, n) = gcd(n, m \text{ mod } n)$

Example: Greatest Common Divisor $gcd(m, n)$

3 algorithms calculating $gcd(m, n)$. Assume $m > n > 0$.

1. From $gcd(m, n)$'s definition: check each number in decreasing order by 1 since $\min\{m, n\}$.

   $n, n-1, n-2, \ldots$

2. From $gcd(m, n)$'s definition and prime numbers definition: $gcd(m, n)$ is the product of all common factors of m and n.

   - List all prime numbers $\leq n$. (Sieve method)
   - Prime number factorization of m and n.
   - Multiply all the common factors.

3. Euclid's algorithm: $gcd(m, n) = gcd(n, m \bmod n)$

➔ Question: Which one runs the fastest?

Outline

- Review
- Time efficiency
- Worst-case, best-case, and average case
- Order of growth
- O, Ω and Θ
- Examples and exercises

Empirical Analysis vs. Theoretical Analysis of Time Efficiency

- Approach of estimating the running time

1. Select a typical sample of inputs
2. Count actual number of basic executions or record the running time
3. Analyze the collected data

Empirical Analysis vs. Theoretical Analysis of Time Efficiency

- Approach of estimating the running time
  1. Select a typical sample of inputs
  2. Count actual number of basic executions or record the running time
  3. Analyze the collected data
- Drawbacks
  - Difficult to decide on how many samples/tests are needed to be done
  - Hardware/environmental dependent
  - Implementation dependent

Theoretical Analysis of Time Efficiency

- **Goal**

  Provide a *machine independent* measurement

- **Common sense**

  The size of input is increased $\longrightarrow$ algorithms run longer

Theoretical Analysis of Time Efficiency

➢ **Goal**

Provide a *machine independent* measurement

We are satisfied with this goal.

➢ **Common sense**

The size of input is increased $\longrightarrow$ algorithms run longer

➢ **Measurement is based on**

- input size
- basic operation
- order of growth of running time

Theoretical Analysis of Time Efficiency

- **Input size**

1. sort $\{a_1, a_2, \cdots, a_n\}$

2. $\begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix} \begin{bmatrix} b_{11} & \cdots & b_{1k} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mk} \end{bmatrix}$

3. $prime(n)$

4. $\cdots$

- **Basic operation**

The operation that contributes most towards the running time

Theoretical Analysis of Time Efficiency

- **Input size**
- **Basic operation**
- **Running time $T(n)$**

$$T(n) \approx c_{op} \cdot C(n),$$

where

- n is the input size
- $C(n)$ is the total number of basic operations for input of size n.
- c_{op} is the time needed to execute one single basic operation.

Outline

- Review
- Time efficiency
- Worst-case, best-case, and average case
- Order of growth
- O, Ω and Θ
- Examples and exercises

Worst-case, Best-case, Average-case

- **Input size**
- **Basic operation**
- **Running time** $T(n)$

$$T(n) \approx c_{op} \cdot C(n),$$

➢ For some algorithms efficiency depends on form of input:

- Worst case: $C_{worst}(n) \longrightarrow$ maximum over inputs of size n
- Best case: $C_{best}(n) \longrightarrow$ minimum over inputs of size n
- Average case: $C_{avg}(n) \longrightarrow$ "average" over inputs of size n

Example: Greatest Common Divisor

Algorithm 0.1: $\mathrm{gcd}(a, b)$

for $i = \{\min(a,b), \cdots, 1\}$

do $\begin{cases} \textbf{if } a \bmod i = 0 \text{ and } b \bmod i = 0 \\ \quad \textbf{then return } (i) \end{cases}$

- Input size: $n = \min(a, b)$
- Basic operation: a mod i

Example: Greatest Common Divisor

Algorithm 0.2: $\mathrm{gcd}(a, b)$

$$
\textbf{for } i = \{\min(a,b), \cdots, 1\} \\
\textbf{do } \begin{cases} \textbf{if } a \bmod i = 0 \text{ and } b \bmod i = 0 \\ \quad \textbf{then return } (i) \end{cases}
$$

- Input size: $n = \min(a, b)$
- Basic operation: a mod i
- Worst case (worst case analysis provides an upper bound):
  - When does the worst case happen? (a and b are relatively prime)
  - $C_{worst}(n) = n$

Example: Greatest Common Divisor

Algorithm 0.3: $\mathrm{gcd}(a, b)$

for $i = \{\min(a, b), \cdots, 1\}$

do $\begin{cases} \textbf{if } a \bmod i = 0 \text{ and } b \bmod i = 0 \\ \quad \textbf{then return } (i) \end{cases}$

- Input size: $n = \min(a, b)$
- Basic operation: a mod i
- Best case
  - When does the best case happen? ($\mathrm{gcd}(a, b) = \min(a, b)$)
  - $C_{best}(n) = 1$

Example: Greatest Common Divisor

Algorithm 0.4: $\text{gcd}(a, b)$

for $i = \{\min(a, b), \cdots, 1\}$

do $\begin{cases} \textbf{if } a \text{ mod } i = 0 \text{ and } b \text{ mod } i = 0 \\ \quad \textbf{then return } (i) \end{cases}$

- Input size: $n = \min(a, b)$
- Basic operation: a mod i
- Average case
  - Things need to be pay attention to
    - ∗ Number of times of the basic operation will be executed on typical instances

* NOT the average of worst and best cases
* Expected number of basic operations considered as a random variable under some assumption about the probability distribution of all possible inputs

– Assumptions
  * Assume that a and b are two randomly chosen integers
  * Assume that all integers have the same probability of being chosen
  * **hint**: The probability that an integer i is a and b's greatest common divisor is $P_{a,b}(i) = \frac{6}{\pi^2 i^2}$
    · $gcd(a, b)$ as the integer i such that ($i|a, b$) and $x := a/i$ and $y := b/i$ are co-prime.
    · The probability of two integers sharing a factor i is i^{-2}. The probability that two integers are co-prime is $6/\pi^2$.)

Example: Greatest Common Divisor

Algorithm 0.5: $\text{gcd}(a, b)$

for $i = \{\min(a, b), \cdots, 1\}$
do $\begin{cases} \textbf{if } a \text{ mod } i = 0 \text{ and } b \text{ mod } i = 0 \\ \quad \textbf{then return } (i) \end{cases}$

- Average case

  Denote $n = \min(a, b)$

$$\begin{aligned} C_{avg}(n) &= 1 \cdot P_{a,b}(n) + 2 \cdot P_{a,b}(n-1) + \cdots + n \cdot P_{a,b}(1) \\ &= \frac{6}{\pi^2}(\frac{1}{n^2} + \frac{2}{(n-1)^2} + \cdots + \frac{n}{1^2}) \end{aligned}$$

When $n = 10$, $C_{avg}(10) = 8.583$.

Outline

- Review
- Time efficiency
- Worst-case, best-case, and average case
- Order of growth
- O, Ω and Θ
- Examples and exercises

Orders of Growth

- **Input size**
- **Basic operation**
- **Running time** $T(n)$

$$T(n) \approx c_{op} \cdot C(n),$$

where n is the input size, $C(n)$ is the total number of basic operations for input of size n, and c_{op} is the time needed to execute one single basic operation.

- **Examples**

Given that $C(n) = \frac{1}{2}n(n-1)$, how much the performance will be affected if the input size n is doubled?

$$growth = \frac{T(2n)}{T(n)} \approx \frac{c_{op} \cdot C(2n)}{c_{op} \cdot C(n)} = \frac{4n-2}{n-1} \approx 4$$