CSC 8301- Design and Analysis of Algorithms

Lecture 2
Principles of the Analysis of Algorithms

Reading: Sections 2.1 and 2.2

Exercises:
2.1: 1, 2, 3, 4, 6, 8, 10
2.2: 1, 2, 3, 5, 7

Next time: Analysis of nonrecursive algorithms (2.3)

Analysis of Algorithms

- **Issues:**
  - Termination
  - Correctness
  - Time efficiency
  - Space efficiency
  - Optimality
- **Approaches:**
  - Theoretical analysis
  - Empirical analysis

Theoretical analysis of time efficiency

Time efficiency is analyzed by determining the number of *basic operations* as a function of *input size*

- ***Input size:*** **amount of memory needed to store input**
- ***Basic operation:*** **proportional to amount of time taken by algorithm.**

input size

$$T(n) = c_{op} C(n)$$

running time — execution time for basic operation — Number of basic operations

Input size and basic operation examples

Problem	*Input size measure*	*Basic operation*
Search for key in list of n items	Number of items in list n	Key comparison
Multiply two matrices of floating point numbers	Dimensions of matrices	Floating point multiplication
Compute a^n	n	Floating point multiplication
Solve graph problem	#vertices and/or edges	Visiting a vertex or traversing an edge

Empirical analysis of time efficiency

- Also measured in terms of input size
- Use physical unit of time (e.g., milliseconds)

  OR

- Count actual number of basic operations

Best-case, average-case, worst-case

For some algorithms efficiency depends on form of input:

- **Worst case:** W(n) – maximum over inputs of size n
- **Best case:** B(n) – minimum over inputs of size n
- **Average case:** A(n) – "average" over inputs of size n
  - Number of times the basic operation will be executed on typical input
  - NOT the average of worst and best case
  - Expected number of basic operations considered as a random variable under some assumption about the probability distribution of all possible inputs

Example: Sequential search

- *Problem:* Given a list of n elements and a search key K, find an element equal to K, if any.
- *Algorithm:* Scan the list and compare its successive elements with K until either a matching element is found (*successful search*) of the list is exhausted (*unsuccessful search*)
- Worst case
- Best case
- Average case

Types of formulas for basic operation count

- Exact formula
  e.g., $C(n) = n(n-1)/2$
- Formula indicating order of growth with specific multiplicative constant
  e.g., $C(n) \approx 0.5\, n^2$
- Formula indicating order of growth with unknown multiplicative constant
  e.g., $C(n) \approx cn^2$

Order of growth

- Most important: Order of growth within a constant multiple as $n \to \infty$
- Example:
  - How much faster will algorithm run on computer that is twice as fast?
  - How much longer does it take to solve problem of double input size?
- See tables 2.1 and 2.2

Asymptotic growth rate

- A way of comparing functions that ignores constant factors and small input sizes
- $O(g(n))$: class of functions $f(n)$ that grow <u>*no faster*</u> than $g(n)$
- $\Theta(g(n))$: class of functions $f(n)$ that grow <u>*at same rate*</u> as $g(n)$
- $\Omega(g(n))$: class of functions $f(n)$ that grow <u>*at least as fast*</u> as $g(n)$

Establishing rate of growth: Method 1 – using limits

$$\lim_{n\to\infty} T(n)/g(n) = \begin{cases} 0 & \text{order of growth of } T(n) \text{ \_\_\_ order of growth of } g(n) \\ c>0 & \text{order of growth of } T(n) \text{ \_\_\_ order of growth of } g(n) \\ \infty & \text{order of growth of } T(n) \text{ \_\_\_ order of growth of } g(n) \end{cases}$$

Examples:

- $10n$ vs. $2n^2$
- $n(n+1)/2$ vs. n^2
- $\log_b n$ vs. $\log_c n$

L'Hôpital's rule

If

- $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = \infty$
- The derivatives f', g' exist,

Then

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

- Example: $\log n$ vs. n

Establishing rate of growth: Method 2 – using definition

- $f(n)$ is $O(g(n))$ if order of growth of $f(n) \leq$ order of growth of $g(n)$ (within constant multiple)
- There exist positive constant c and non-negative integer n_0 such that

$$f(n) \leq c\, g(n) \text{ for every } n \geq n_0$$

Examples:

- $10n$ is $O(2n^2)$
- $5n+20$ is $O(10n)$