Chapter 3, Algorithms

Young-Rae Cho
Associate Professor
Department of Computer Science
Baylor University

3.1. Algorithms

- **Definition**
  - A finite sequence of precise instructions for performing a specific task or for solving a problem

- **Examples**

```
BottlesOfBeer(x)
    for i ← x down to 1
        sing " i bottles of beer on the wall, i bottles of beer,"
        sing " take one down, pass it around, (i-1) bottles of beer on the wall."
        sing " No bottles of beer on the wall, no bottles of beer,"
        sing " Go to the store, buy some more, x bottles of beer on the wall."
```
Description of Algorithms

- **How to Express Algorithms?**
  - Natural languages (English)
  - Programming languages (Code)
  - Flow chart or Diagram
  - Pseudocode
    - Compact and informal high-level description of algorithms
    - Ignoring details in codes,
    - However, unambiguous for the task

```
FIBONACCI(n)
F_1 ← 1
F_2 ← 1
for i ← 3 to n
  F_i ← F_{i-1} + F_{i-2}
return F_n
```

Evaluation of Algorithms

- **How to Evaluate Algorithms?**
  - Correctness
    - An algorithm should produce the correct output values for each set of input values.
  - Finiteness
    - An algorithm should produce the desired output after a finite number of steps for any input.
  - Effectiveness
    - An algorithm should be performed in a finite amount of time.
    - Big-O notation is used.
  - Generality
    - An algorithm should be applicable for all problems of the desired form, not just for a particular set of input values.
Exhaustive Search Algorithms

- **Procedure of Exhaustive Search Algorithm**
  - Also called Brute force algorithm
  - Examine all possible output values (or cases) to find a solution.

- **Advantages, Disadvantages?**

- **Examples**
  - Finding the smallest number in a finite sequence of numbers
    - Pseudocode?
  - Sorting the numbers in a finite sequence from the smallest to the largest
    - Pseudocode?

Greedy Algorithms

- **Procedure of Greedy Algorithms**
  - Determine the optimal structure of a problem.
  - Find the local optimal solution at each step, instead of considering the entire sequence of steps.

- **Advantages, Disadvantages?**

- **Examples**
  - Counting a certain amount of money using the fewest bills and coins
    - Optimal structure?
  - Assigning $m$ jobs into $n$ processors to finish all the jobs as early as possible (where $m > n$)
    - Optimal structure?
Manhattan Tourist Problem

**Problem Definition**
- A tourist seeks a path to travel with the most attractions in Manhattan road map (grid structure)
- Restrictions
  - A path from a source to a sink
  - A path only eastward and southward

![Grid structure of Manhattan Tourist Problem]

Formulation of Manhattan Tourist Problem

**Goal**
- Finding the strongest path from a *source* to a *sink* in a weighted grid
  - The weight of an edge is defined as the number of attractions
  - The path strength is measured by summing the weights on the path

**Input**
- A weighted grid $G$ with two distinct vertices, *source* and *sink*

**Output**
- A strongest path in $G$ from the *source* to the *sink*
Example of Manhattan Tourist Problem

- Example

Solving Manhattan Tourist Problem

- Solving by Exhaustive Search Algorithm
  - Algorithm
    1. Enumerate all possible paths from the source to the sink
    2. Compute the path strength for all possible paths
    3. Find the strongest path
  - Problems?

- Solving by Greedy Algorithm
  - Algorithm
    1. Starting from the source, select the edge having the highest weight repeatedly until it reaches the sink
  - Problems?
3.2. The Growth of Functions

➢ Background
  • Analysis of an algorithm
    ▪ The running time to execute the algorithm
    ▪ The number of operations to execute the algorithm
    ▪ Called "time complexity" of the algorithm
    ▪ The asymptotic behavior of the function of the algorithm with respect to the size of input
    ▪ How fast the function of the algorithm grows as input size increases

➢ Examples
  • Suppose an algorithm $A$ takes $f_A(n) = 30n + 8$ microseconds to process $n$ records of input, while an algorithm $B$ takes $f_B(n) = n^2 + 1$ microseconds to process $n$ records of input

Big-O Notation

➢ Definition of Big-O Notation
  • Let $f$ and $g$ be functions $\mathbb{R}$ (or $\mathbb{Z}$) $\rightarrow \mathbb{R}$.
  • $f(x) = O(g(x))$ if there are constants $C$ and $k$ such that $|f(x)| \leq C |g(x)|$ for all $x > k$
  • It is read as "$f(x)$ is big-O of $g(x)$".
  • An asymptotic upper bound

➢ Examples?

➢ Theorem
  • Let $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ (i.e. a polynomial of degree $n$), where $a_n, a_{n-1}, \ldots, a_1, a_0$ are real numbers. Then, $f(x) = O(x^n)$. 
The Growth of Functions in Big-O Estimates

- The Growth of Common Functions
- The Growth of Combinations of Functions

Suppose that \( f_1(x) = O(g_1(x)) \) and \( f_2(x) = O(g_2(x)) \).
- \((f_1 + f_2)(x) = O(\max(|g_1(x)|, |g_2(x)|))\) → Proved using the triangle inequality.
- \((f_1 f_2)(x) = O(g_1(x) g_2(x))\).

Big-Ω Notation

- Definition of Big-Ω Notation
  - Let \( f \) and \( g \) be functions \( \mathbb{R} \) (or \( \mathbb{Z} \)) → \( \mathbb{R} \).
  - \( f(x) = \Omega(g(x)) \) if there are constants \( C \) and \( k \) such that \(|f(x)| \geq C |g(x)| \) for all \( x > k \).
  - It is read as "\( f(x) \) is big-Omega of \( g(x) \)"
  - An asymptotic lower bound

- Examples?
Big-Θ Notation

- **Definition of Big-Θ Notation**
  - Let \( f \) and \( g \) be functions \( \mathbb{R} \) (or \( \mathbb{Z} \)) \( \rightarrow \mathbb{R} \).
  - \( f(x) = \Theta(g(x)) \) if there are constants \( C_1, C_2 \) and \( k \) such that
    - \( C_1|g(x)| \leq |f(x)| \leq C_2|g(x)| \) for all \( x > k \)
  - It is read as “\( f(x) \) is big-Theta of \( g(x) \)”.
  - An asymptotic tight bound

- **Examples?**

- **Theorem**
  - \( f(x) = \Theta(g(x)) \) if \( f(x) = O(g(x)) \) and \( f(x) = \Omega(g(x)) \).

---

3.3. Complexity of Algorithms

- **Computational Complexity**
  - Effectiveness (or Efficiency) analysis of an algorithm
  - Time complexity
    - The running time (the number of operations) to execute the algorithm
  - Space complexity
    - The number of memory bits required to execute the algorithm

- **Time Complexity**
  - Worst-case complexity?
  - Average-case complexity?
Questions?

- Lecture Slides are found on the Course Website,
  web.ecs.baylor.edu/faculty/cho/2350