CS 240 - Data Structures and Data Management **Spring 2017**

Lecture 2: May 4th, 2017

Lecturer: Taylor Smith *Notes By: Harsh Mistry*

2.1 Algorithms and Programs

2.1.1 Terminology

Definition 2.1 *Problem : Given a problem instance, carry out a particular computational task*

Definition 2.2 *Problem Instance : Input for the specified problem*

Definition 2.3 *Problem Solution : Output for the specified problem*

Definition 2.4 *Size of a problem instance : Size(I) is a positive integer which is measure of the size of the instance I*

Definition 2.5 *Algorithm : An algorithm is a step-by step process for carrying out a series of computations, given an arbitrary problem instance I.*

Definition 2.6 *Algorithm solving a problem : An algorithm A solves a problem (x) is for every instance I of x, A finds a valid solution for the instance I in finite time*

Definition 2.7 *Program : A program is an implementation of an algorithm using a specified computer language*

2.1.2 Order Notation

1. O-notation ($\leq$) : $f(n) \in O(g(n))$ if there exists constants $c > 0$ and $n_0 > 0$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$
2. Ω-notation ($\geq$) : $f(n)\Omega(g(n))$ if there exists constants $c > 0$ and $n_0 > 0$ such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$
3. θ-notation ($=$) : $f(n) \in \theta(g(n))$ if there exists constants $c_1, c_2 > 0$ and $n_0 > 0$ such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$
4. o-notation ($<$): $f(n) \in o(g(n))$ if for all constants $c > 0$ and $n_0 > 0$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$
5. ω-notation ($>$) : $f(n) \in \omega(g(n))$ if for all constants $c > 0$ and $n_0 > 0$ such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$

2.1.3 Complexity of algorithms

1. Average-case complexity of an algorithm :

$$T_A^{avg}(n) = \frac{1}{\mid \{I : Size(I) = n\} \mid} \sum_{\{I:Size(I)=n\}} T_A(I)$$

2. Worst-case complexity of an algorithm :

$$T_A(n) = max\{T_A(I) : Size(I) = n\}$$

2.1.4 Growth Rates

- If $f(n) \in \theta(g(n))$, then growth rates are the same
- If $f(n) \in o(g(n))$, then growth rate of f(n) is less than g(n)
- If $f(n) \in \omega(g(n))$, then the growth rate for f(n) is greater than the growth rate of g(n).
- Typically f(n) may be ”complicated” and g(n) is chosen to be a very simple function.

2.1.5 Common Growth Rates

- $\theta(1)$ (Constant Complexity)
- $\theta(logn)$ (Logarithmic Complexity)
- $\theta(n)$ (Linear Complexity)
- $\theta(nlogn)$ (linearithmic)
- $\theta(nlog^k n)$, for some constant k (quasi-linear)
- $\theta(n^2)$ (Quadratic Complexity)
- $\theta(n^3)$ (Cubic Complexity)
- $\theta(2^n)$ (Exponential Complexity)