Reference of complexity

VE281 - Data Structures and Algorithms, Xiaofeng Gao, TA: Qingmin Liu, Autumn 2019

1 The O-Notation

The O-notation provides an *upper bound* of the running time; it may not be indicative of the actual running time of an algorithm.

Definition 1 (O-Notation)**.** *Let $f(n)$ and $g(n)$ be functions from the set of natural numbers to the set of nonnegative real numbers. $f(n)$ is said to be $O(g(n))$, written $f(n) = O(g(n))$, if*

$$\exists c.\exists n_0.\forall n \geq n_0.f(n) \leq cg(n)$$

Intuitively, f grows no faster than some constant times g.

2 The Ω-Notation

The Ω-notation provides a *lower bound* of the running time; it may not be indicative of the actual running time of an algorithm.

Definition 2 (Ω-Notation)**.** *Let $f(n)$ and $g(n)$ be functions from the set of natural numbers to the set of nonnegative real numbers. $f(n)$ is said to be $\Omega(g(n))$, written $f(n) = \Omega(g(n))$, if*

$$\exists c.\exists n_0.\forall n \geq n_0.f(n) \geq cg(n)$$

Clearly $f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$.

3 The Θ-Notation

The Θ-notation provides an exact picture of the growth rate of the running time of an algorithm.

Definition 3 (Θ-Notation)**.** *Let $f(n)$ and $g(n)$ be functions from the set of natural numbers to the set of nonnegative real numbers. $f(n)$ is said to be $\Theta(g(n))$, written $f(n) = \Theta(g(n))$, if both $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.*

Clearly $f(n) = \Theta(g(n))$ if and only if $g(n) = \Theta(f(n))$.

4 The o-Notation

Definition 4 (o-Notation)**.** *Let $f(n)$ and $g(n)$ be functions from the set of natural numbers to the set of nonnegative real numbers. $f(n)$ is said to be $o(g(n))$, written $f(n) = o(g(n))$, if*

$$\forall c.\exists n_0.\forall n \geq n_0.f(n) < cg(n)$$

5 The ω-Notation

Definition 5 (ω-Notation)**.** *Let $f(n)$ and $g(n)$ be functions from the set of natural numbers to the set of nonnegative real numbers. $f(n)$ is said to be $\omega(g(n))$, written $f(n) = \omega(g(n))$, if*

$$\forall c.\exists n_0.\forall n \geq n_0.f(n) > cg(n)$$

6 Definition in Terms of Limits

Suppose $\lim_{n\to\infty} f(n)/g(n)$ **exists**.

- $\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq \infty$ implies $f(n) = O(g(n))$.
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq 0$ implies $f(n) = \Omega(g(n))$.
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$ implies $f(n) = \Theta(g(n))$.
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ implies $f(n) = o(g(n))$.
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$ implies $f(n) = \omega(g(n))$.

7 A Helpful Analogy

- $f(n) = O(g(n))$ is similar to $f(n) \leq g(n)$.
- $f(n) = o(g(n))$ is similar to $f(n) < g(n)$.
- $f(n) = \Theta(g(n))$ is similar to $f(n) = g(n)$.
- $f(n) = \Omega(g(n))$ is similar to $f(n) \geq g(n)$.
- $f(n) = \omega(g(n))$ is similar to $f(n) > g(n)$.

8 Complexity Classes

An equivalence relation $\mathcal{R}$ on the set of complexity functions is defined as follows: $f\mathcal{R}g$ if and only if $f(n) = \Theta(g(n))$.

A complexity class is an equivalence class of $\mathcal{R}$.

The equivalence classes can be ordered by $\prec$ defined as follows: $f \prec g$ iff $f(n) = o(g(n))$.

$1 \prec \log\log n \prec \log n \prec \sqrt{n} \prec n^{\frac{3}{4}} \prec n \prec n\log n \prec n^2 \prec 2^n \prec n! \prec 2^{n^2}$