CS302 - Data Structures
using C++

Topic: Formal Definitions

Kostas Alexis

O, Ω, and Θ

- Let T, f be two monotone increasing functions that maps from $\mathbb{N}$ to $\mathbb{R}^+$
- **Asymptotic Upper Bounds**
  - We say $T(n) = O\big(f(n)\big)$ if there exists constants $c > 0$ and $n_0 \geq 0$, such that for all $n \geq n_0$, we have $T(n) \leq cf(n)$

O, Ω, and Θ

- Let T, f be two monotone increasing functions that maps from $\mathbb{N}$ to $\mathbb{R}^+$

- **Asymptotic Upper Bounds**
  - We say $T(n) = O\big(f(n)\big)$ if there exists constants $c > 0$ and $n_0 \geq 0$, such that for all $n \geq n_0$, we have $T(n) \leq cf(n)$

- **Examples**
  - $T(n) = 100n + 100 \Rightarrow T(n) = O(n)$
  - $T(n) = pn^2 + qn + r$ for constants $p, q, r, > 0 \Rightarrow T(n) = O(n^2)$

O, Ω, and Θ

- Let T, f be two monotone increasing functions that maps from $\mathbb{N}$ to $\mathbb{R}^+$

- **Asymptotic Upper Bounds**
  - We say $T(n) = O\big(f(n)\big)$ if there exists constants $c > 0$ and $n_0 \geq 0$, such that for all $n \geq n_0$, we have $T(n) \leq cf(n)$

- **Examples**
  - $T(n) = 100n + 100 \Rightarrow T(n) = O(n)$
    - Set $c = 1001$ and $n_0 = 100$
  - $T(n) = pn^2 + qn + r$ for constants $p, q, r, > 0 \Rightarrow T(n) = O(n^2)$

O, Ω, and Θ

- Let T, f be two monotone increasing functions that maps from $\mathbb{N}$ to $\mathbb{R}^+$
- **Asymptotic Upper Bounds**
  - We say $T(n) = O\big(f(n)\big)$ if there exists constants $c > 0$ and $n_0 \geq 0$, such that for all $n \geq n_0$, we have $T(n) \leq cf(n)$
- **Examples**
  - $T(n) = 100n + 100 \Rightarrow T(n) = O(n)$
    - Set $c = 1001$ and $n_0 = 100$
  - $T(n) = pn^2 + qn + r$ for constants $p, q, r, > 0 \Rightarrow T(n) = O(n^2)$
    - Set $c = p + q + r$ and $n_0 = 1$
    - Also correct to say $T(n) = O(n^3)$

Remarks

- **Equals sign.** $O(f(n))$ is a set of functions, but computer scientists often write $T(n) = O\big(f(n)\big)$ instead of $T(n) \epsilon\, O(f(n))$
  - Consider $f(n) = 5n^3$ and $g(n) = 3n^2$, we write $f(n) = O(n^3) = g(n)$, but it does not mean that $f(n) = g(n)$

Remarks

- **Equals sign.** $O(f(n))$ is a set of functions, but computer scientists often write $T(n) = O\big(f(n)\big)$ instead of $T(n) \epsilon O(f(n))$
  - Consider $f(n) = 5n^3$ and $g(n) = 3n^2$, we write $f(n) = O(n^3) = g(n)$, but it does not mean that $f(n) = g(n)$
- **Domain.** The domain of $f(n)$ is typically the natural numbers $\{0,1,2, \ldots\}$
  - Sometimes we restrict to a subset of the natural numbers. Other times we extend to the reals.

Remarks

- **Equals sign.** $O(f(n))$ is a set of functions, but computer scientists often write $T(n) = O\big(f(n)\big)$ instead of $T(n) \epsilon O(f(n))$
  - Consider $f(n) = 5n^3$ and $g(n) = 3n^2$, we write $f(n) = O(n^3) = g(n)$, but it does not mean that $f(n) = g(n)$
- **Domain.** The domain of $f(n)$ is typically the natural numbers $\{0,1,2, \dots\}$
  - Sometimes we restrict to a subset of the natural numbers. Other times we extend to the reals.
- **Nonnegative functions.** When using big-O notation, we assume the functions involved are (asymptotically) nonnegative.

O, Ω, and Θ

- Let T, f be two monotone increasing functions that maps from $\mathbb{N}$ to $\mathbb{R}^+$
- **Asymptotic Lower Bounds**
  - We say $T(n) = \Omega\big(f(n)\big)$ if there exists constants $\epsilon > 0$ and $n_0 \geq 0$, such that for all $n \geq n_0$, we have $T(n) \geq \epsilon f(n)$
- **Examples**
  - $T(n) = pn^2 + qn + r$ for constants $p, q, r, > 0 \Rightarrow T(n) = \Omega(n^2)$
    - Set $\epsilon = p$ and $n_0 = 1$
    - Also correct to say $T(n) = \Omega(n)$
- **Meaningful Statement.** Any compare-based sorting algorithm requires $\Omega(\mathrm{nlogn})$ compares in the worst case.
- **Meaningless Statement.** Any compare-based sorting algorithm requires $O(nlogn)$ compares in the worst case

O, Ω, and Θ

- Let T, f be two monotone increasing functions that maps from $\mathbb{N}$ to $\mathbb{R}^+$
- **Asymptotic Tight Bounds**
  - We say $T(n) = \Theta(f(n))$ if there exists constants $c_1, c_2 > 0$ and $n_0 \geq 0$, such that for all $n \geq n_0$, we have $c_1 \leq T(n) \leq c_2$
- **Alternative definition**
  - $T(n) = \Theta(\mathrm{f(n)})$ if $T(n)$ is both $O(f(n))$ and also $\Omega(\mathrm{f(n)})$
- **Examples**
  - $T(n) = pn^2 + qn + r$ for constants $p, q, r > 0 \Longrightarrow T(n) = \Theta(n^2)$
    - Set $c_1 = p, c_2 = p + q + r$ and $n_0 = 1$
    - $T(n)$ is neither $\Theta(\mathrm{n})$ nor $\Theta(n^3)$

Properties of Asymptotic Growth Rates

- If $f = O(g)$ and $g = O(h)$, then $f = O(h)$
- If $f = \Omega(g)$ and $g = \Omega(h)$, then $f = \Omega(h)$
- If $f = \Theta(g)$ and $g = \Theta(h)$, then $f = \Theta(h)$
- If $f = O(\eta)$ and $g = O(h)$, then $f + g = O(h)$
- If $\mathrm{g} = O(f)$, then $f + g = \Theta(\varphi)$
- If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c > 0$, then $f = \Theta(\mathrm{g})$

Some Common Functions

- **Polynomials.** Let f be a polynomial of degree d, in which the coefficient a_d is positive. Then $f = O(n^d)$.
  - Asymptotic rate of growth is determined by their "higher-order term".
  - Polynomial-Time Algorithm: A polynomial-time algorithm is one with running time $O(n^d)$ for some constant d.
- **Logarithms.** For every $a, b > 1$ and every $\chi > 0$, we have $log_a n = \Theta(log_b n) = O(n^\chi)$
  - No need to specify base (assuming it is a constant).
  - Logarithms are always better than polynomials.
- **Exponentials.** For every $r > 1$ and every $d > 0$, we have $n^d = O(r^n)$.
  - Polynomials are always better than exponentials.

More Notations

- **Asymptotic Smaller**
  - We say $T(n) = o\big(f(n)\big)$ if $\lim_{n\to\infty} \frac{T(n)}{f(n)} = 0$
- **Asymptotic Larger**
  - We say $T(n) = \omega\big(f(n)\big)$ if $\lim_{n\to\infty} \frac{f(n)}{T(n)} = 0$

O, Ω, and Θ with multiple variables

- **Asymptotic Upper Bounds**
  - We say $T(m,n) = \mathrm{O}\big(f(m,n)\big)$ if there exists constants $\mathrm{c} > 0$ and $m_0 \geq 0$ and $n_0 \geq 0$, such that for all $\mathrm{m} \geq m_0$ and $n \geq n_0$, we have $T(m,n) \leq cf(m,n)$.
- Similar definitions hold for Ω and Θ

- **Examples**
  - $T(m,n) = 32mn^2 + 17mn + 32n^3$
    - $T(m,n)$ is both $O(mn^2 + n^3)$ and $O(mn^3)$
    - $T(m,n)$ is neither $O(n^3)$ nor $O(mn^2)$

Thank you