# Asymptotic Behavior of Functions

Textbook Reading:

Section 3.1, pp. 81-92.

# Asymptotic Behavior of Functions

$$f(n) = 100n + 50 \text{ and } g(n) = n^2 - n$$

Algorithm $A$ has worst-case complexity $f(n)$ and algorithm $B$ worst-case complexity $g(n)$ . Which is the faster algorithm?

# Answer

- Algorithm A.
- For all $n \geq 102$, $f(n)$ is smaller than $g(n)$ and for very large $n$, it is a lot smaller.
- Small values of $n < 102$, for which both $f(n)$ and $g(n)$ are at most 10150, are not important, since computers these days can perform billions of calculations per second.
- What matters is the number of computations that need to be performed as $n$ becomes large.

# Asymptotic Order

Let N denote the natural numbers,

$R^+$ the positive reals.

Our notion of order is restricted to the set $\mathscr{F}$ of functions

$$f(n)\colon \mathrm{N} \to \mathrm{R}^+$$

# Asymptotic Notation

Big Oh $O(g(n))$

Theta $\Theta(g(n))$

Omega $\Omega(g(n))$

Little oh $o(g(n))$

Asymptotically equivalent $\sim g(n)$

# Big Oh $f(n) \in O(g(n))$

Given a function $g(n) \in \mathscr{F}$, we define $O(g(n))$ to be the set of all functions $f(n) \in \mathscr{F}$ having the property that there exist positive constants $c$ and $n_0$ such that for all $n \geq n_0$,

$$f(n) \leq cg(n), \text{ for all } n \geq n_0,$$

PSN. For the example, $f(n) = 100n + 50$ and $g(n) = n^2 - n$, show that

$$f(n) \in O(g(n)).$$

Is this definition meaningful. For example is

$n^2 - n \in O(100n + 50)$.

Why not just chose a really, really huge constant $c$? Surely,

$$n^2 - n \leq 1000000000000000000000000000(100n + 50) \quad \text{for all } n \geq 1.$$

PSN. Show this isn’t true.

# Comparing orders

We have shown that $100n + 50 \in O(n^2 - n)$, but $n^2 - n \notin O(100n + 50)$.

We say that $100n + 50$ has **smaller order** than $n^2 - n$.

# Omega $f(n) \in \Omega(g(n))$

Given a function $g(n) \in \mathscr{F}$, we define $\Omega(g(n))$ to be the set of all functions $f(n) \in$ F having the property that there exist positive constants $c$ and $n_0$ such that for all $n \geq n_0$,

$$f(n) \geq cg(n), \text{ for all } n \geq n_0,$$

# Theta $f(n) \in \Theta(g(n))$

Given a function $g(n) \in \mathscr{F}$, we define $\Theta(g(n))$ to be the set of all functions $f(n) \in \mathscr{F}$ having the property that there exist positive constants $c_1$ and $c_2$ and $n_0$ such that for all $n \geq n_0$,

$$c_1 g(n) \leq f(n) \leq c_2 g(n), \text{ for all } n \geq n_0,$$

# Same order

$f(n)$ and $g(n)$ have the **same order** iff $f(n) \in \Theta(g(n))$.

"Same order" is an **equivalence relation**. For convenience denote the relation by $\Theta$, *i.e.*, $f \,\Theta\, g \Leftrightarrow f(n) \in \Theta(g(n))$.

This allows us to formally define the order of a function as the equivalence class it belongs to, where we choose the function in the class having the "simplest" form to represent the class.

# $\Theta$ is and equivalence relation

**Reflexive:** $\boldsymbol{f\ \Theta\ f}$

$c_1 f(n) \le f(n) \le c_2 f(n)$, for all $n \ge n_0$, for $c_1 = c_2 = n_0 = 1$

**Symmetric:** $\boldsymbol{f\ \Theta\ g \Rightarrow g\ \Theta\ f}$

$f\ \Theta\ g \Rightarrow \exists c_1, c_2, n_0$ such that $c_1 g(n) \le f(n) \le c_2 g(n)$, for all $n \ge n_0$

$\Rightarrow (1/c_2) f(n) \le g(n) \le (1/c_1) f(n)$, for all $n \ge n_0 \Rightarrow g\ \Theta\ f$

**Transitive:** $\boldsymbol{f\ \Theta\ g}$ **and** $\boldsymbol{g\ \Theta\ h \Rightarrow f\ \Theta\ h}$

$f\ \Theta\ g \Rightarrow \exists c_1, c_2, n_0$ such that $c_1 g(n) \le f(n) \le c_2 g(n)$, for all $n \ge n_0$

$g\ \Theta\ h \Rightarrow \exists d_1, d_2, n_0$ such that $d_1 h(n) \le g(n) \le d_2 h(n)$, for all $n \ge n_1$

$\Rightarrow c_1 d_1 h(n) \le f(n) \le c_2 d_2 h(n)$, for all $n \ge \max\{n_0, n_1\}$

$\Rightarrow f\ \Theta\ h$

# logarithms

PSN. Show that $\log_a n$ *and* $\log_b n$ *have the same order independent of the choice of bases a and b.*

Sometimes = is used instead of $\in$. For example, instead of saying $f(n)$ belongs to $O$(g(n)), denoted $f(n) \in O(g(n))$, we can say $f(n)$ is $O(g$(n)), denoted $f(n) = O(g(n))$.

# Comparing orders

**Theorem 3.1.2 The Ratio Limit Theorem**

Let $f(n)$, $g(n) \in$ F. If the limit of the ratio $f(n)/g(n)$ exists as $n$ tends to infinity, then $f$ and $g$ are comparable. Moreover, assuming $L = \lim_{n\to\infty} f(n)/g(n)$ exists, then the following results hold.

| | | |
|---|---|---|
| 1. | $0 < L < \infty \Rightarrow f(n) \in \Theta(g(n))$. | $f$ and $g$ have the same order. |
| 2. | $L = 0 \Rightarrow O(f(n)) \subset O(g(n))$ | $f$ has a *smaller* order than $g$. |
| 3. | $L = \infty \Rightarrow O(g(n)) \subset O(f(n))$ | $f$ has a *larger* order than $g$. |

PSN. Using the Ratio Limit Theorem, show that worst-case complexity of Binary Search has smaller order than the worst-case complexity of Linear Search, i.e.,

$$O(\log_2 n) \subset O(n).$$

# Hint: Apply L'Hôpital's Rule

**L'Hôpital's Rule:** Suppose $\lim_{n\to\infty} f(n)$ and $\lim_{n\to\infty} g(n)$ are both 0 or both ∞. Then

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

# little oh $f(n) \in o(g(n))$

We say that $f(n)$ is "little oh" of $g(n)$, denoted $f(n) \in o(g(n))$, if

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0.$$

By the Ratio Limit Theorem

$$f(n) \in o(g(n)) \Rightarrow f(n) \in O(g(n))$$

# Strongly Asymptotic Behavior

We say that $f(n)$ is **strongly asymptotic** to $g$(n), denoted $f(n) \sim g(n)$, if

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1.$$

By the Ratio Limit Theorem

$$f(n) \sim g(n) \Rightarrow f(n) \in \Theta(g(n))$$

# *Insertionsort* Complexities

**Proposition.** The complexities of *Insertionsort*, assuming a uniform distribution for the average complexity, satisfy

$$B(n) \sim n \qquad W(n) \sim \frac{n^2}{2} \qquad A(n) \sim \frac{n^2}{4}$$

# Using formula for *B*(*n*), *W*(*n*), *A*(*n*) for Insertion Sort

Best-Case: $B(n) = n - 1$

$$\lim_{n\to\infty} \frac{n-1}{n} = \lim_{n\to\infty} \left(1 - \frac{1}{n}\right) = 1 + 0 = 1$$

Worst-Case: $W(n) = \frac{n^2}{2} - \frac{n}{2}$

$$\lim_{n\to\infty} \frac{\frac{n^2}{2} - \frac{n}{2}}{\frac{n^2}{2}} = \lim_{n\to\infty} \left(1 - \frac{1}{n}\right) = 1 + 0 = 1$$

Average: $A(n) = \frac{n^2}{4} + \frac{3n}{4} - \frac{1}{2}$

$$\lim_{n\to\infty} \frac{\frac{n^2}{4} + \frac{3n}{4} - \frac{1}{2}}{\frac{n^2}{4}} = \lim_{n\to\infty} \left(1 + \frac{3}{n} - \frac{2}{n^2}\right) = 1 + 0 + 0 = 1$$

# Use of Big Oh vs. Theta

- In the literature big oh is used more extensively then other asymptotic notation. This is because we can say that algorithms like Linear Search and Insertion Sort have computing times $O(n)$ and $O(n^2)$, respectively, because they have these computing times for all inputs of size $n$.
- Linear Search has computing times ranging from 1 to $n$, which is $O(n)$ for any input of size $n$, so it would be incorrect to say it has computing time $\Theta(n)$. We would need to say that it has computing time $\Theta(n)$ in the worst-case.
- The assumption when saying Linear Search has computing time $O(n)$ is that this is sharp, i.e., it has worst-case complexity $\Theta(n)$.

# $O(1)$ in real life