1. Pseudo-code – describe algorithms

2. Asymptotic notation – discuss efficiency

3. Design techniques – design algorithms

1. Concentrate on the worst case
2. Ignore constant factors/ lower-order terms
3. Asymptotic analysis – for large values of n

A FAST algorithm is one for which the worst-case running time grows slowly with input size
Asymptotic notation

1. Pseudo-code – describe algorithms
2. Asymptotic notation – discuss efficiency
3. Design techniques – design algorithms

- Describe growth of functions.
- Focus on what’s important by abstracting away low-order terms and constant factors.

How we indicate running times of algorithms.
A way to compare “sizes” of functions:

$O \leq \Omega \leq \Theta = \approx \leq \leq \leq < \omega$
Asymptotic notation

\[ O(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq c g(n) \text{ for all } n \geq n_0 \} . \]

\[ g(n) \text{ is an asymptotic upper bound for } f(n). \]

If \( f(n) \in O(g(n)) \), we write \( f(n) = O(g(n)) \).
Asymptotic notation

\[ \Omega(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0 \} . \]

\[ g(n) \text{ is an asymptotic lower bound for } f(n). \]
Asymptotic notation

\[ \Theta(g(n)) = \{ f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0 \} . \]

**Theorem 3.1**
For any two functions \( f(n) \) and \( g(n) \), we have \( f(n) = \Theta(g(n)) \) if and only if \( f(n) = O(g(n)) \) and \( f(n) = \Omega(g(n)) \).

\( g(n) \) is an **asymptotically tight bound** for \( f(n) \).
Asymptotic notation

\[ o(g(n)) = \{ f(n) : \text{for all constants } c > 0, \text{there exists a constant } n_0 > 0 \text{ such that } 0 \leq f(n) < cg(n) \text{ for all } n \geq n_0 \} \]

\[ \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \]

\[ \omega(g(n)) = \{ f(n) : \text{for all constants } c > 0, \text{there exists a constant } n_0 > 0 \text{ such that } 0 \leq cg(n) < f(n) \text{ for all } n \geq n_0 \} \]

\[ \lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty \]
Asymptotic notation in equations

When on right-hand side

$O(n^2)$ stands for some anonymous function in the set $O(n^2)$.

$2n^2 + 3n + 1 = 2n^2 + \Theta(n)$ means $2n^2 + 3n + 1 = 2n^2 + f(n)$ for some $f(n) \in \Theta(n)$. In particular, $f(n) = 3n + 1$.

When on left-hand side

No matter how the anonymous functions are chosen on the left-hand side, there is a way to choose the anonymous functions on the right-hand side to make the equation valid.

Interpret $2n^2 + \Theta(n) = \Theta(n^2)$ as meaning for all functions $f(n) \in \Theta(n)$, there exists a function $g(n) \in \Theta(n^2)$ such that $2n^2 + f(n) = g(n)$.

Can chain together:

\[ 2n^2 + 3n + 1 = 2n^2 + \Theta(n) = \Theta(n^2). \]
ASYMPTOTIC NOTATION: PROPERTIES

Transitivity:
\[ f(n) = \Theta(g(n)) \text{ and } g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n)). \]
Same for \( O, \Omega, o, \text{ and } \omega. \)

Reflexivity:
\[ f(n) = \Theta(f(n)). \]
Same for \( O \text{ and } \Omega. \)

Symmetry:
\[ f(n) = \Theta(g(n)) \text{ if and only if } g(n) = \Theta(f(n)). \]

Transpose symmetry:
\[ f(n) = O(g(n)) \text{ if and only if } g(n) = \Omega(f(n)). \]
\[ f(n) = o(g(n)) \text{ if and only if } g(n) = \omega(f(n)). \]
**COMPARISON OF FUNCTIONS**

$f(n)$ is *asymptotically smaller* than $g(n)$ if $f(n) = o(g(n))$.

$f(n)$ is *asymptotically larger* than $g(n)$ if $f(n) = \omega(g(n))$.

No trichotomy. Although intuitively, we can liken $O$ to $\leq$, $\Omega$ to $\geq$, etc., unlike real numbers, where $a < b$, $a = b$, or $a > b$, we might not be able to compare functions.

Example: $n^{1+\sin n}$ and $n$, since $1 + \sin n$ oscillates between 0 and 2.

Some problems from the text: 3.1-3, 3.1-4, 3-2

Let $f(n)$ and $g(n)$ denote *non-negative* functions of $n$. Prove that $\max(f(n), g(n)) = \Theta(f(n) + g(n))$.