References for Today’s Lecture

• Required reading
  – CLRS Chapter 3

• References
Some Notational Conventions

• Unless otherwise specified, we will assume all functions map $\mathbb{N}_+$ to $\mathbb{R}_+$.

• Our usual function names will be $f$, $g$, and $T$.

• We will also assume that $n$ is a variable denoting the input size that takes on values in $\mathbb{N}_+$.

• We will also use $m$ as a variable taking on values in $\mathbb{N}_+$.

• We will use $a$, $b$, and $c$ to denote constants.

• Generally, all variables and constants will take on values in $\mathbb{N}_+$.

• Although it is common practice, I will try not to refer to a function by the notation $f(n)$ because $f(n)$ is a value, not a function.

  – Correct: “$f$ is a polynomial function.”
  – Incorrect: “$f(n)$ is a polynomial function.”
Growth of Functions

• **Question**: Why are we *really* interested in the theoretical running times of algorithms?

• **Answer**: To compare different algorithm for solving the same problem.

• We are interested in performance for large input sizes.

• For this purpose, we need only compare the *asymptotic growth rates* of the running times.
  
  – Consider algorithm $A$ with running time given by $f$ and algorithm $B$ with running time given by $g$.
  
  – We are interested in knowing

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

  
  – What are the four possibilities?
We now define the set

$$\Theta(g) = \{f : \exists c_1, c_2, n_0 > 0 \text{ such that } c_1 g(n) \leq f(n) \leq c_2 g(n) \forall n \geq n_0\}$$

- If $f \in \Theta(g)$, then we say that $f$ and $g$ grow at the same rate or that they are of the same order.
- Note that

$$f \in \Theta(g) \iff g \in \Theta(f)$$

- We also know that if $\lim_{n \to \infty} \frac{f(n)}{g(n)} = c$ for some constant $c$, then $f \in \Theta(g)$.
- If the limit doesn’t exist, we don’t know.
**Big-\(O\) Notation**

- We now define the set
  \[
  O(g) = \{ f : \exists c, n_0 > 0 \text{ such that } 0 \leq f(n) \leq cg(n) \forall n \geq n_0 \}
  \]

- If \( f \in O(g) \), then we say that “\( f \) is big-\( O \) of \( g \)” or that \( g \) grows at least as fast as \( f \).

- Some other notation
  - \( f \in \Omega(g) \iff g \in O(f) \).
  - \( f \in o(g) \iff \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \).
  - \( f \in \omega(g) \iff g \in o(f) \iff \lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty \).

- Note that \( f \in o(g) \Rightarrow f \in O(g) \setminus \Theta(g) \).
Example

- Recall the polynomial evaluation example from last class.
- Let’s show that if $f(n) = \frac{1}{2}(n^2 + 3n)$, then $f \in \Theta(n^2)$. 

Comparing Functions

- The notation we have just defined gives us a way of ordering functions.

- We can interpret
  - \( f \in O(g) \) as "\( f \leq g \),"
  - \( f \in \Omega(g) \) as "\( f \geq g \),"
  - \( f \in o(g) \) as "\( f < g \),"
  - \( f \in \omega(g) \) as "\( f > g \)," and
  - \( f \in \Theta(g) \) as "\( f = g \)."

- This gives us a method for comparing algorithms based on their running times.

- Note that most of the relational properties of real numbers (transitivity, reflexivity, symmetry) work here also.

- However, not every pair of functions is comparable.
Commonly Occurring Functions

- **Polynomials**: All polynomials $f$ of degree $k$ are in $\Theta(n^k)$.

- **Exponentials**
  - A function in which $n$ appears as an exponent on a constant is an *exponential function*, i.e., $2^n$.
  - For all positive constants $a$ and $b$, $\lim_{n \to \infty} \frac{n^a}{b^n} = 0$.
  - This means that exponential functions always grow faster than polynomials.

- **Logarithms**
  - Logarithms of different bases differ only by a constant multiple, so they all grow at the same rate.
  - A *polylogarithmic* function is a function in $O(lg^k)$.
  - Polylogarithmic functions always grow more slowly than polynomials.

- **Factorials**: Factorial functions grow more quickly than exponentials, but are in $o(n^n)$. 
Problem Difficulty

• The **difficulty** of a problem can be judged by the (worst-case) running time of the **best-known algorithm**.

• Problems for which there is an algorithm with polynomial running time (or better) are called **polynomially solvable**.

• Generally, these problems are considered to be **easy**.

• There are many interesting problems for which it is not known if there is a polynomial-time algorithm.

• These problems are generally considered **difficult**.

• One of the great open questions in mathematics is whether these problems really are difficult or if we just haven’t discovered the right algorithm yet.

• If you answer this question, you can win a **million dollars**.

• In this course, we will stick mostly to the easy problems.