IE170: Algorithms in Systems Engineering: Lecture 2

Jeff Linderoth

Department of Industrial and Systems Engineering
Lehigh University

January 17, 2007

Sums

Arithmetic Series

$$1 + 2 + \cdots + n = \sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

Sum Of Squares

$$\sum_{k=0}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

- Often, such formulae can be proved via *mathematical induction*

Induction

- A way to prove that every statement in a (countably) infinite sequence of statements is true.

How to do Induction

1. Prove that the first statement in the infinite sequence of statements is true: The base case.
2. Prove that if any one statement in the infinite sequence of statements is true, then so is the next one: The induction .

More Sums

Geometric Series

$$\sum_{k=0}^{n} x^k = \frac{1 - x^{n+1}}{1 - x}$$

If $|x| < 1$, then the series converges to

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1 - x}.$$

Harmonic Series

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{k} = \sum_{k=1}^{n} \frac{1}{k} \approx \ln(n)$$

Bounding Sums By Integrals

- When f is a (monotonically) increasing function, then we can approximate the sum $\sum_{k=m}^{n} f(k)$ by the integrals:

$$\int_{m-1}^{n} f(x)dx \leq \sum_{k=m}^{n} f(k) \leq \int_{m}^{n+1} f(x)dx.$$

  and a decreasing function can be approximated by

$$\int_{m}^{n+1} f(x)dx \leq \sum_{k=m}^{n} f(k) \leq \int_{m-1}^{n}$$

- For example, the harmonic series ($\sum_{k=1}^{n} k^{-1}$).

$$\int_{1}^{n+1} x^{-1}dx \quad \leq \quad \sum_{k=1}^{n} k^{-1} \leq \int_{0}^{n} x^{-1}dx$$

$$\ln(n+1) \quad \leq \quad \sum_{k=1}^{n} k^{-1} \leq \ln(n) + 1$$

The Joy of Sets

- You are also responsible for knowing the definitions and notation of sets given in Appendix B
- $\emptyset$: Empty Set
- $\mathbb{Z}$: The set of integers: $\{-2, -1, 0, 1, 2\}$
- $\mathbb{R}$: The set of real numbers
- $\mathbb{R}_+$: The set of non-nonnegative real numbers: $\{x \in \mathbb{R} \mid x \geq 0\}$
- $A \subseteq B \Rightarrow x \in A \Rightarrow x \in B$
- $A \not\subseteq B \Rightarrow \exists x \in A$ such that $x \notin B$
- $|A|$ denotes the cardinality, or number of elements, of the set A.
  - Note that $|A|$ is not finite for all sets

The Joy of Sets

- $A \cap B = \{x \mid x \in A$ and $x \in B\}$
- $A \cup B = \{x \mid x \in A$ or $x \in B\}$
- $A \setminus B = \{x \mid x \in A$ and $x \notin B\}$
- For any two sets A and B, we have the identity

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

- This is a specialization of the general principle of inclusion and exclusion

You're On Your Own

- Be sure to read and understand the sections on bounding summations and splitting summations (Appendix A.2)
- Be sure to read sections on relations, functions, graphs (B.2, B.3, and B.4)
- This course is fairly mathematical, so you need to know this stuff. :-(
- I will try and (re)-introduce the mathematics we need as we go, but if you are ever confused by my jibberish and jargon in class, please feel free to stop me and ask a question.

Some Notational Conventions for Today

- Unless otherwise specified, we will assume all functions map $\mathbb{N}$ to $\mathbb{R}_+$
- The symbols f, g, and T will typically denote such functions
- The variable n will typically be used to denote the *input size* for an algorithm
- We will use a, b, and c to denote constants.
- In an abuse of notation, I *may* refer to $f(n)$ as a function, but in reality it is simply a value.
  - 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?

Answers

1. To get to the other side
2. To get a reasonable grade in this course
3. To compare different algorithms 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.

Comparing Algorithms

- 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?
  - $L = 0$: g grows faster than f
  - $L = \infty$: f grows faster than g
  - $L = c$: f and g grow at the same rate.
  - The limit doesn't exist.

Θ Notation

- We now define the set

$$\Theta(g) = \{f : \exists\, c_1, c_2, n_0 > 0 \text{ such that } c_1 g(n) \leq f(n) \leq c_2 g(n)\ \forall n \geq n_0\} \quad (1)$$

- 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

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

- We also know that if

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

for some constant c, then $f \in \Theta(g)$.

Big-O Notation

- We now define the set of functions

$$O(g) = \{f : \exists c, n_0 > 0 \text{ such that } 0 \leq f(n) \leq cg(n)\ \forall n \geq n_0\}$$

- 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 facts and notation:
  - $f \in \Omega(g) \Leftrightarrow g \in O(f)$.
  - $f \in o(g) \Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.
  - $f \in \omega(g) \Leftrightarrow g \in o(f) \Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$.
- Note that $f \in o(g) \Rightarrow f \in O(g) \setminus \Theta(g)$.

Comparing Functions

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

Commonly Occurring Functions

Polynomials

- $f(n) = \sum_{i=0}^{k} a_i n^i$ is a polynomial of degree k
- 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

More Functions

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

- $n! = n(n-1)(n-2)\cdots(1)$
- $n! = o(n^n)$
- $n! = \omega(2^n)$
- $\lg(n!) = \Theta(n \lg n)$

Logs

- $a^n a^m = a^{n+m}$
- We use the notation
  - $\lg n = \log_2 n$
  - $\ln n = \log_e n$
  - $\lg^k n = (\lg n)^k$
- Changing the base of a logarithm changes its value by a constant factor

Log Rules

- $a = b^{\log_b a}$
- $\lg \left(\prod_{k=1}^n a_k\right) = \sum_{k=1}^n \lg a_k$
- $\log_b a^n = n \log_b a$
- $\log_b a = (\log_c a)/(\log_b a)$
- $\log_b a = 1/(\log_a b)$
- $a^{\log_b n} = n^{\log_b a}$

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.
  - Formally, they are in the complexity class $\mathcal{P}$
- There are many interesting problems for which it is not known if there is a polynomial-time algorithm.
- These problems are generally considered difficult.
  - This is known as the complexity class $\mathcal{NP}$.

A++++++++++++++++++++++++

- You will get a very good grade in this class if you prove $\mathcal{P} = \mathcal{NP}$
- It is open of the great open questions in mathematics: Are these truly difficult problems, or have we not yet discovered the right algorithm?
- If you answer this question, you can win a million dollars: `http://www.claymath.org/millennium/P_vs_NP/`
- Most important, you can get the jokes from the Simpsons: `www.mathsci.appstate.edu/~sjg/simpsonsmath/`
- In this course, we will stick mostly to the easy problems, for which a polynomial time algorithm is known.

Next Time

- A short amount of time to address homework questions
- Recurrences and the Master Method