CS4/531
Analysis of Algorithms
(Spring 1996)

Prof. Russ Miller
217 Bell Hall
645-3180 × 113
miller@cs.buffalo.edu
http://www.cs.buffalo.edu/pub/WWW/faculty/miller/

1 Administrivia

Lecture: TTh, 12:30-1:50, NatSci 220

Recitation: F, 12:00-12:50, Norton 213 **or** M, 2:00-2:50, Norton 210 **or** W, 11-11:50, Capen 262

Dr. Russ Miller's Office Hrs.: Tues. 10:30-11:30

T.A.s: Ajay P Nerurkar, Office Hrs.: M, 3:00-4:00, F, 1:00-2:00, Bell 329; Yingzhong Wen, WTh, 3:00-4:00, Trailer B.

Preamble: CS531 is a graduate-level course in Analysis of Algorithms that is required of all graduate students in the Department of Computer Science. CS431 is an undergraduate-level course in Algorithms that serves as an elective for undergraduate students at the University at Buffalo.

Reading Material:

- *Introduction to Algorithms*, Cormen, Leiserson, and Rivest, McGraw-Hill, 1990. Available in the bookstore.
- Miscellaneous handouts that will be distributed during the semester.

Grading Policy:

1. Midterm exam will account for 25% of your grade.
2. Final exam will account for 40% of your grade.

3. Homeworks will account for 35% of your grade.

4. If you **pass** both the Exam Portion **and** the Non-exam Portion of the course, then your course grade will be the (weighted) average of these two letter grades. If you fail *either* portion of the course, you will receive an F in the course.

Assignments must be submitted *before* the beginning of class on the day they are due.

Recitation is required. New and supplementary material will be presented in recitation sections.

Plagiarism:

- Any plagiarism, be it on homeworks/programs or exams, will earn you an F in the course. Further, you should be aware that the Department of Computer Science has recently taken a serious stance in such instances and may bring you up on charges of academic dishonesty and attempt to have you expelled from the University.

Disclaimer: *I reserve the right to change any part of this tentative syllabus at any time.*

2 Tentative Syllabus

1. Prerequisite: A course in *Advanced Data Structures*

2. Students should know, and are responsible for, the material in chapters 1-13. In addition, students are also responsible for material on balanced trees (e.g., AVL, Red/Black, B-trees).

3. Topics to be covered:

   - Brief review of asymptotic notation
   - Brief review of Divide-and-Conquer
   - Dynamic Programming (chapt. 16)
   - Greedy Algorithms (chapt. 17)
   - Graph Algorithms (chapts. 23-26)
   - Computational Geometry (chapt. 35)
   - Parallel Algorithms (Chapt. 30)

3 Introduction to Asymptotic Notation

We use Θ to mean 'order exactly', O to mean 'order at most', and Ω to mean 'order at least'.

Definition of Θ: Given nonnegative functions f and g defined on the positive integers, we write $f = \Theta(g)$ if and only if there are positive constants C_1, C_2, and a positive integer N such that

$$C_1 * g(n) \leq f(n) \leq C_2 * g(n), \text{ whenever } n > N.$$

Definition of O: Given nonnegative functions f and g defined on the positive integers, we write $f = \mathrm{O}(g)$ if and only if there is a positive constant C and an integer N such that

$$f(n) \leq C * g(n), \text{ for all } n > N.$$

Definition of Ω: Given nonnegative functions f and g defined on the positive integers, we write $f = \Omega(g)$ if and only if there is a positive constant C and an integer N such that

$$C * g(n) \leq f(n), \text{ for all } n > N.$$

When using order notation, one wants to use the simplest function possible within the Θ, O, or Ω. The idea is to use the notation to reduce complicated functions to simpler ones whose behavior is easier to understand.

Proposition 1: Suppose f and g are positive functions defined on the positive integers, and that $\lim_{n\to\infty} \frac{f(n)}{g(n)} = a$, for some nonzero a. Then $f = \Theta(g)$.

Proof: $lim_{n\to\infty}\frac{f(n)}{g(n)} = a$ means that for any $c > 0$ there is an integer N such that $\left|\frac{f(n)}{g(n)} - a\right| \leq c$, for all $n \geq N$. If we chose $c = \frac{a}{2}$ then there is an N such that $\frac{a}{2} \leq \frac{f(n)}{g(n)} \leq \frac{3a}{2}$ for all $n \geq N$. So, $\left(\frac{a}{2}\right) * g(n) \leq f(n) \leq \left(\frac{3a}{2}\right) * g(n)$, for all $n \geq N$. $\square$

Proposition 1 is useful because there are many techniques, such as L'Hopital's rule, for evaluating limits. However, there are some instances where Proposition 1 cannot be used.

For example, $2 + sin(n) = \Theta(1)$, even though $\lim_{n\to\infty} \frac{[2+sin(n)]}{1}$ is undefined.

Examples of Θ: If $f(n) = 3n + 6$ and $g(n) = n$, then $f(n) = \Theta(g(n))$, or $f = \Theta(g)$, or $3n + 6 = \Theta(n)$, as can be seen by using $C_1 = 3, C_2 = 4$, and $N = 5$.

Notice that if $f = \Theta(g)$ then $g = \Theta(f)$, since if there are positive constants C_1, C_2, and N such that $C_1 * g(n) \le f(n) \le C_2 * g(n)$ for all $n > N$, then $\left(\frac{1}{C_2}\right) * f(n) \le g(n) \le \left(\frac{1}{C_1}\right) * f(n)$, for all $n > N$.

$3n+2 = \Theta(n)$. $10n^2+4n+2 = \Theta(n^2)$. $10n^2+4n+2 \ne \Theta(n)$. $10*\log n+4 = \Theta(\log n)$.□

Useful Properties of Θ:

1. If $f(n) = \Theta(h(n))$ and $g(n) = \Theta(h(n))$, then $(f + g)(n) = \Theta(h(n))$.
2. If $f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n))$, then $f(n) = \Theta(h(n))$.
3. $a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0 = \Theta(n^k), a_k > 0$.
4. $\log_a(n) = \Theta(\log_b(n))$ for any $a, b > 1$.
5. $\sum_{i=1}^{n} i = \Theta(n^2)$
6. $\sum_{i=1}^{n} \log_a(i) = \Theta(n \log n), a > 1$.
7. $\sum_{i=1}^{n} \frac{1}{i} = \Theta(\log(n))$.
8. $\sum_{i=1}^{n} \frac{1}{i^2} = \Theta(1)$.
9. If $f(n) = \Theta(h(n)), g(n) = \Theta(h(n))$, and $f(n) \le e(n) \le g(n)$, then $e(n) = \Theta(h(n))$.

O and Ω Notation

It is not always possible to determine the behavior of an algorithm using Θ-notation. For example, given a problem with n inputs, we may have an algorithm to solve the problem which takes Dn^2 time when n is even and Cn time when n is odd. Or, we may only be able to prove that an algorithm never uses more than En^2 time and never less than Fn time. In either circumstance we can claim neither $\Theta(n)$ nor $\Theta(n^2)$ to be the order of the time usage of the algorithm. O and Ω notation will allow us to give at least partial information.

Proposition 2: Suppose f and g are positive functions defined on the positive integers, and that $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, then $f =$O(g). □

Examples of O: $3n + 2 = \mathrm{O}(n)$, since $3n + 2 \leq 4n$, for all $n \geq 2$. $100n + 6 = \mathrm{O}(n)$, since $100n + 6 \leq 101n$, for all $n \geq 10$. $10n^2 + 4n + 2 = \mathrm{O}(n^2)$, since $10n^2 + 4n + 2 \leq 11n^2$, for all $n \geq 5$. $3n + 2 \neq \mathrm{O}(1)$, since $3n + 2$ is not less than or equal to c for any constant c and all $n, n \geq N$. $10n^2 + 4n + 2 \neq \mathrm{O}(n)$. $10n^2 + 4n + 2 = \mathrm{O}(n^4)$. $\square$

Proposition 3: Suppose f and g are positive functions defined on the positive integers, and that $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, then $f = \Omega(g)$. $\square$

Examples of Ω: $3n + 2 = \Omega(n)$, since $3n + 2 \geq 3n$ for all $n \geq 1$. $100n + 6 = \Omega(n)$, since $100n + 6 \geq 100n$ for all $n \geq 1$. $10n^2 + 4n + 2 = \Omega(n^2)$ since $10n^2 + 4n + 2 \geq n^2$ for all $n \geq 1$. $10n^2 + 4n + 2 = \Omega(n)$. $10n^2 + 4n + 2 = \Omega(1)$. $\square$

Useful Properties of O and Ω:

1. $f = \Theta(g)$ if and only if $f = \mathrm{O}(g)$ and $f = \Omega(g)$.
2. If $f = \mathrm{O}(h)$ and $g = \mathrm{O}(h)$ then $f + g = \mathrm{O}(h)$.
3. If $f = \Omega(h)$ and $g = \Omega(h)$ then $f + g = \Omega(h)$.
4. If $f = \mathrm{O}(g)$ and $g = \mathrm{O}(h)$ then $f = \mathrm{O}(h)$.
5. If $f = \Omega(g)$ and $g = \Omega(h)$ then $f = \Omega(h)$.
6. Suppose $0 < a < b$. Then $n^a = \mathrm{O}(n^b)$ and $n^b = \Omega(n^a)$.
7. Suppose $0 < a$ and $1 < b$, then $\log_b n = \mathrm{O}(n^a)$ and $n^a = \Omega(\log_b n)$.
8. Suppose $1 < c$ and $0 < a$, then $n^a = \mathrm{O}(c^n)$ and $c^n = \Omega(n^a)$.

4 Homework

Unless otherwise stated, you should *prove* all answers. In addition, unsupported solutions will receive no credit.

These are *not* joint assignments. That is, you are to do the homeworks on your own.

4.1 Assignment #1 due Feb. 6

1. Rank the following functions by order of growth; that is, find an arrangement of $g_1, g_2, \ldots, g_{11}$ of the functions satisfying $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, ..., $g_{10} = \Omega(g_{11})$.

   n, $n^{1/2}$, $\log n$, $\log(\log n)$, $\log^2 n$, $n/\log n$, $n^{1/2}\log^2 n$, $(1/3)^n$, 4, $(3/2)^n$, $n!$

2. CLR 1-2

3. CLR 2.1-1

4. CLR 2.2-4

5. CLR 8.2-1

6. CLR 8.2-4

4.2 Assignment #2 due Feb. 20

1. A sorting algorithm is defined as *stable* if equal elements are in the same relative order in the sorted sequence as in the original sequence. Which of insertion sort, quicksort, selection sort, and mergesort are stable, and which are unstable. Give a simple fix to make the unstable sorts stable.
2. CLR 8.3-4
3. CLR 8.4-1
4. CLR 8-2
5. CLR 8-4
6. CLR 16.3-4
7. CLR 16-1
8. CLR 16-2

4.3 Assignment #3 due ???

4.4 Assignment #4 due ???

4.5 Assignment #5 due ???

4.6 Assignment #6 due ???

4.7 Assignment #7 due ???