CSE 373: Data Structures and Algorithms

Lecture 4: Math Review/Asymptotic Analysis II

Functions in Algorithm Analysis

- $f(n) : \{0, 1, \ldots\} \rightarrow \Re^+$
  - domain of *f* is the nonnegative integers
  - range of *f* is the nonnegative reals

- Unless otherwise indicated, the symbols *f*, *g*, *h*, and *T* refer to functions with this domain and range.

- We use many functions with other domains and ranges.
  - Example: $f(n) = 5\, n \log_2 (n/3)$
    - Although the domain of *f* is nonnegative integers, the domain of $\log_2$ is all positive reals.

Efficiency examples 5

```
sum = 0;
for (int i = 1; i <= N; i *= c) {
   sum++;
}
```

$\log_c N$

$\log_c N + 1$

Math background: Logarithms

- Logarithms
  - *definition*: $X^A = B$ if and only if $\log_X B = A$
  - *intuition*: $\log_X B$ means:
    "the power X must be raised to, to get B"

  - In this course, a logarithm with no base implies base 2.
    $\log B$ means $\log_2 B$

- Examples
  - $\log_2 16 = 4$ (because $2^4 = 16$)
  - $\log_{10} 1000 = 3$ (because $10^3 = 1000$)

Logarithm identities

Identities for logs with addition, multiplication, powers:

- log (AB) = log A + log B
- log (A/B) = log A – log B
- log (A^B) = B log A

Identity for converting bases of a logarithm:

- $$\log_A B = \frac{\log_C B}{\log_C A} \quad A, B, C > 0, A \neq 1$$
  - example:
    $\log_4 32 = (\log_2 32) / (\log_2 4)$
    $= 5 / 2$

Techniques: Logarithm problem solving

- When presented with an expression of the form:
  - $\log_a X = Y$

  and trying to solve for X, raise both sides to the a power.
  - $X = a^Y$

- When presented with an expression of the form:
  - $\log_a X = \log_b Y$

  and trying to solve for X, find a common base between the logarithms using the identity on the last slide.
  - $\log_a X = \log_a Y / \log_a b$

Logarithm practice problems

- Determine the value of *x* in the following equation.
  - $\log_7 x + \log_7 13 = 3$

- Determine the value of *x* in the following equation.
  - $\log_8 4 - \log_8 x = \log_8 5 + \log_{16} 6$

Prove identity for converting bases

Prove $\log_a b = \log_c b / \log_c a$.

A log is a log…

- We will assume all logs are to base 2

- Fine for Big Oh analysis because the log to one base is equivalent to the log of another base within a constant factor
  - E.g., $\log_{10} x$ is equivalent to $\log_2 x$ within what constant factor?

Efficiency examples 6

```
int sum = 0;
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= i / 2; j += 2) {
        sum++;
    }
}
```

Math background: Arithmetic series

- Series

$$\sum_{i=j}^{k} Expr$$

  - for some expression *Expr* (possibly containing *i*), means the sum of all values of *Expr* with each value of *i* between *j* and *k* inclusive

Example:

$$\sum_{i=0}^{4} 2i+1$$

= (2(0) + 1) + (2(1) + 1) + (2(2) + 1)
+ (2(3) + 1) + (2(4) + 1)
= 1 + 3 + 5 + 7 + 9
= 25

Series identities

- sum from 1 through N inclusive

$$\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$$

- is there an intuition for this identity?
  - sum of all numbers from 1 to N

    1 + 2 + 3 + ... + (N-2) + (N-1) + N

  - how many terms are in this sum? Can we rearrange them?

More series identities

- sum from *a* through N inclusive
(when the series doesn't start at 1)

$$\sum_{i=a}^{N} i = \sum_{i=1}^{N} i - \sum_{i=1}^{a-1} i$$

- is there an intuition for this identity?

Series of constants

- sum of constants
  (when the body of the series doesn't contain the counter variable such as *i*)

$$\sum_{i=a}^{b} k = k\sum_{i=a}^{b} 1 = k(b-a+1)$$

- example:

$$\sum_{i=4}^{10} 5 = 5\sum_{i=4}^{10} 1 = 5(10-4+1) = 35$$

Splitting series

for any constant *k*,

- splitting a sum with addition

$$\sum_{i=a}^{b}(i+k) = \sum_{i=a}^{b} i + \sum_{i=a}^{b} k$$

- moving out a constant multiple

$$\sum_{i=a}^{b} ki = k\sum_{i=a}^{b} i$$

Series of powers

- sum of powers of 2

$$\sum_{i=0}^{N} 2^i = 2^{N+1} - 1$$

  – 1 + 2 + 4 + 8 + 16 + 32 = 64 - 1 = 63
  – think about binary representation of numbers...

```
  111111 (63)
 +     1 (1)
 -------
 1000000 (64)
```

- when the series doesn't start at 0:

$$\sum_{i=a}^{N} 2^i = \sum_{i=0}^{N} 2^i - \sum_{i=0}^{a-1} 2^i$$

Series practice problems

- Give a closed form expression for the following summation.
  - A closed form expression is one without the Σ or "…".

$$\sum_{i=0}^{N-2} 2i$$

- Give a closed form expression for the following summation.

$$\sum_{i=10}^{N-1} (i-5)$$

Efficiency examples 6 (revisited)

```
int sum = 0;
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= i / 2; j += 2) {
        sum++;
    }
}
```

- Compute the value of the variable sum after the following code fragment, as a closed-form expression in terms of input size n.
  - Ignore small errors caused by i not being evenly divisible by 2 and 4.

Big omega, theta

- **big-Oh Defn**: T(N) = O(g(N)) if there exist positive constants c , n_0 such that:
  $T(N) \leq c \cdot g(N)$ for all $N \geq n_0$

- **big-Omega Defn**: T(N) = Ω(g(N)) if there are positive constants c and n_0 such that $T(N) \geq c\, g(N)$ for all $N \geq n_0$
  - Lingo: "T(N) grows no slower than g(N)."

- **big-Theta Defn**: T(N) = Θ(g(N)) if and only if T(N) = O(g(N)) and T(N) = Ω(g(N)).
  - Big-Oh, Omega, and Theta establish a *relative ordering* among all functions of N

- **little-oh Defn**: T(N) = o(g(N)) if and only if T(N) = O(g(N)) and T(N) $\neq$ Ω(g(N)).

Intuition about the notations

notation	intuition
O (Big-Oh)	T(N) ≤ g(n)
Ω (Big-Omega)	T(N) ≥ g(n)
Θ (Theta)	T(N) = g(n)
o (little-Oh)	T(N) < g(n)