Discrete Mathematical Structures
CS 3233 Lecture 15

Prof. William Winsborough
October 19, 2006

Business

- Read section 3.3 for Tuesday
- Recall: Homework 6
  Due Tuesday 24 October
  - 3.2: 6, 8, 10, 14, 20, 22
- Reminder: Winsborough will be out of town 10/30-11/2
  - Lecture will be covered by Prof. Tom Bylander
  - Recitation will be covered by Catherine

What We Did in Recitation Tuesday

- Worked problem 23 from section 3.1
- Proved:
  $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$ is $O(x^n)$
- Worked problems 3 and 5 in section 3.2

Still More Examples

- f(n) = n! is the product of the first n positive integers
  $1 \cdot 2 \cdot \ldots \cdot n \le \underbrace{n \cdot n \cdot \ldots \cdot n}_{n} = n^n$

  So n! is $O(n^n)$

Big-O and Sums

- Def: If f_1, f_2 are real-valued functions, (f_1+f_2) is the function such that that $(f_1+f_2)(x) = f_1(x)+f_2(x)$ for all x
- Theorem:
  If $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$, then $(f_1+f_2)(x)$ is $O(\max(|g_1(x)|, |g_2(x)|)$
- Corollary:
  If $f_1(x)$ and $f_2(x)$ are each $O(g(x))$, then $(f_1+f_2)(x)$ is $O(g(x))$

Big-O and Products

- Def: If f_1, f_2 are real-valued functions, $(f_1 f_2)$ is the function such that that $(f_1 f_2)(x) = f_1(x) \cdot f_2(x)$ for all x
- Theorem:
  If $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$, then $(f_1 f_2)(x)$ is $O(g_1(x)\, g_2(x))$

Big-Omega and Big-Theta

- Big-O provides an upper bound on function growth
- Big-Omega gives a lower bound
  - Def: $f(x)$ is $\Omega(g(x))$ if there are positive constants C and k such that $|f(x)| \geq C|g(x)|$ whenever $x>k$
- Big-Theta gives both
  - Def: $f(x)$ is $\Theta(g(x))$ if $f(x)$ is $O(g(x))$ and $f(x)$ is $\Omega(g(x))$
  - In this case we say $f(x)$ is *of order* $g(x)$

Example

- So $\sum_{i=1}^{n} i$ is $\Theta(n^2)$
  - We already shown it is $O(n^2)$, so we just have to show it is $\Omega(n^2)$
  - Summing only the terms greater than or equal to $\lceil n/2 \rceil$, we have $n - \lceil n/2 \rceil +1$ such terms
  - So $1+2+\ldots+n \geq (n - \lceil n/2 \rceil +1) \lceil n/2 \rceil$ $\geq (n/2)(n/2) = n^2/4$

Computational Complexity of Algorithms

- This is where Big-O, Big-Omega, and Big-Theta get used
- Complexity is measured principally in terms of two resources
  - Time Complexity
  - Space Complexity
    - Discussed more in course on data structures
- Worst-case complexity vs. Average-case

Linear Search

proc linear search(x:int; $a_1,a_2,\ldots,a_n$: distinct ints)
i := 1
while ($i \leq n$ and $x \neq a_i$)
  i := i + 1
if $i \leq n$ **then** location := i **else** location := 0

- How many comparisons are performed?

Binary Search

proc binary search(x: int; $a_1, a_2, \ldots, a_n$: <u>increasing</u> ints)
i:=1 {left end of search interval}
j:=n {right end of search interval}
while i<j **begin**
  m:= $\lfloor (i+j)/2 \rfloor$
  if $x > a_m$ **then** i:=m+1 **else** j:= m
  end
if $x=a_i$ **then** location := i **else** location := 0

- How many comparisons are performed?