

## Discrete Mathematical Structures CS 3233 Lecture 14

Prof. William Winsborough  
October 23, 2007

## Business

- Recall: Homework 8 due Thursday 10/25
  - 3.1: 2, 4, 12, 18, 24
  - 3.2: 2, 4
- Next week class will be taught by Prof. Qi Tian (I will be at a conference)

23 October 2007

Winsborough CS 3233 Lecture 14

2

## Growth of Functions

- Big-O Notation
- Def: Let  $f, g : Z \rightarrow R$ . We say  $f(x)$  is  $O(g(x))$  if there are constants  $C$  and  $k$  such that  $|f(x)| \leq C|g(x)|$  for all  $x > k$ 
  - We say “ $f(x)$  is big-oh of  $g(x)$ ”
- When discussing positive-valued functions, we can drop the  $|\cdot|$

23 October 2007

Winsborough CS 3233 Lecture 14

3

## More Examples

- Sum of first  $n$  positive integers  
 $1+2+\dots+n \leq \underbrace{n+n+\dots+n}_n = n^2$   
So  $\sum_{i=1}^n i$  is  $O(n^2)$
- It follows that the complexity of both bubble sort and insertion sort is  $O(n^2)$

23 October 2007

Winsborough CS 3233 Lecture 14

4

## Still More Examples

- $f(n) = n!$  is the product of the first  $n$  positive integers  
 $1 \cdot 2 \cdot \dots \cdot n \leq \underbrace{n \cdot n \cdot \dots \cdot n}_n = n^n$   
So  $n!$  is  $O(n^n)$

23 October 2007

Winsborough CS 3233 Lecture 14

5

## 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  $O(g(x))$

23 October 2007

Winsborough CS 3233 Lecture 14

6

## Big-O and Products

- Def: If  $f_1, f_2$  are real-valued functions,  $(f_1 f_2)$  is the function such 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  $O(g_1(x) g_2(x))$

23 October 2007

Winsborough CS 3233 Lecture 14

7

## 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  $g(x)$

23 October 2007

Winsborough CS 3233 Lecture 14

8

## 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  $1+2+\dots+n \geq (n - \lceil n/2 \rceil + 1) \lceil n/2 \rceil \geq (n/2)(n/2) = n^2/4$

23 October 2007

Winsborough CS 3233 Lecture 14

9

## 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

23 October 2007

Winsborough CS 3233 Lecture 14

10

## Average-case Complexity

- While worst-case complexity analyzes the case in which the input causes the algorithm's cost to be maximal,
  - Average-case Complexity considers all input cases.
  - Also called Expected-case Complexity or, less formally, Expected Cost
- The expected cost of executing an algorithm is the sum over the different cases of the cost of each case times its probability
  - Can be difficult to calculate
  - Assumes you know the probability of each case

23 October 2007

Winsborough CS 3233 Lecture 14

11

## Linear Search Expected Cost

```

proc linear search(x:int; a1,a2,...,an: distinct ints)
  i := 1
  while (i ≤ n and x ≠ ai)
    i := i + 1
  if i ≤ n then location := i else location := 0
    
```

- Each case has  $x$  appearing in a different location
- When  $x$  is in position  $i$ ,  $1 \leq i \leq n$ ,  $2i + 1$  comparisons are performed
- If we assume  $x$  appears somewhere in the list and that each position is equally probable,
  - The probability of each case is  $1/n$
  - The expected cost is  $\sum_{1 \leq i \leq n} (2i+1)(1/n)$

23 October 2007

Winsborough CS 3233 Lecture 14

12

## Simplification of Sum

$$\begin{aligned}\sum_{1 \leq i \leq n} (2i+1)(1/n) &= (1/n) \sum_{1 \leq i \leq n} (2i+1) \\ &= (1/n) (n + \sum_{1 \leq i \leq n} 2i) \\ &= (1/n) (n + 2 \sum_{1 \leq i \leq n} i) \\ &= (1/n) (n + 2((n(n+1))/2)) \\ &= (1/n) (n + n^2 + n) \\ &= 2 + n\end{aligned}$$

Which is  $\Theta(n)$