

Analysis of Algorithms  
 CS 3343 Lecture Four  
 Prof. William Winsborough  
 September 11, 2008

## Business

- Recall: Homework 1
  - Exercise 2.3-3 is due today
  - Exercises 2.1-3, 2.2-1, 2.2-2, 2.3-1, 2.3-3, 2.3-5 are due Tuesday September 16
- Recall: In the future, we will reverse the order of lecture and recitation on Tuesdays  
**The rooms in which we meet will not change**
  - The 3:30 – 4:45 period will be used for a combination of recitation and lecture
  - The 5:30 – 6:20 period will be used for lecture
- Read section 4.2
- Homework 2 Due Thursday September 18:
  - Exercises: 3.1-2, 3.1-3, 3.1-4, 3.2-1, 3.2-2, 4.1-5, 4.1-6

## How to analyze the time-efficiency of a recursive algorithm?

- Express the running time on input of size  $n$  as a function of the running time on smaller problems
  - Such equalities are called *recurrences*

## Analyzing merge sort

$T(n)$	<b>MERGE-SORT</b> $A[1 \dots n]$
$\Theta(1)$	1. If $n = 1$ , done.
$2T(n/2)$	2. Recursively sort $A[1 \dots \lceil n/2 \rceil]$ and $A[\lceil n/2 \rceil + 1 \dots n]$ .
$C(n)$	3. <b>“Merge”</b> the 2 sorted lists

Should be  $T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor)$ , but it turns out not to matter asymptotically.

## Analyzing merge sort

1. **Divide:** Trivial.
2. **Conquer:** Recursively sort 2 subarrays.
3. **Combine:** Merge two sorted subarrays

$$T(n) = 2T(n/2) + D(n) + C(n)$$

$\swarrow$  # subproblems       $\swarrow$  subproblem size       $\swarrow$  work dividing       $\swarrow$  work combining

1. What is the time for the base case? Constant
2. What are  $D(n)$  and  $C(n)$ ?
3. What is the growth order of  $T(n)$ ?

## Recurrence for merge sort

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1; \\ 2T(n/2) + \Theta(n) & \text{if } n > 1. \end{cases}$$

- **Note:**  $\Theta(n)$  stands for an arbitrary *anonymous* function belonging to  $\Theta(n)$
- The base case is often not mentioned when  $T(n) = \Theta(1)$  for values of  $n$  that are bounded by a constant

- What function  $T(n)$  satisfies this equation?
  - This actually depends on the anonymous  $\Theta(n)$  function
- And what is its asymptotic complexity?

## The Substitution Method for Solving Recurrences

- Guess the form of the solution
- Use mathematical induction to find the constants and show that the solution works
- Example:  
 $T(n) = 2T(\lfloor n/2 \rfloor) + n$   
 Guess  $T(n) = O(n \lg n)$   
 Requires us to prove there exists a constant  $c$  such that for sufficiently large  $n$ ,  $T(n) \leq cn \lg n$

11 September 2008

Winsborough CS 3343 Lecture 4

7

## Proof of Induction Step

- Induction assumption:  $T(\lfloor n/2 \rfloor) \leq c \lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor$
- Proof of step:  

$$\begin{aligned} T(n) &\leq 2(c \lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor) + n \\ &\leq cn \lg(n/2) + n \\ &= cn \lg n - cn \lg 2 + n \\ &= cn \lg n - cn + n \\ &\leq cn \lg n \end{aligned}$$
 provided  $c \geq 1$

11 September 2008

Winsborough CS 3343 Lecture 4

8

## Proof of Induction Basis

- When  $n = 1$ , we don't actually have  $T(n) \leq cn \lg n$  because  $T(n) = 1$  and  $\lg 1 = 0$
- But if we take  $n_0 = 2$  and any constant  $c \geq 2$ , then we have for all  $n \geq n_0$ ,  $T(n) \leq cn \lg n$ , as required
- The base case of our induction shows:  
 $T(2) = 2T(1) + 2 \leq 2c \lg 2$  and  
 $T(3) = 2T(1) + 3 \leq 3c \lg 3$   
 – Note that the basis of our induction is different from the basis of the recurrence

11 September 2008

Winsborough CS 3343 Lecture 4

9

## A Curious Phenomenon

- Consider  $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$
- Guess that the solution is  $O(n)$  and try to show  $T(n) \leq cn$  for some constant  $c$
- Induction step:  

$$\begin{aligned} T(n) &\leq c \lfloor n/2 \rfloor + c \lceil n/2 \rceil + 1 \\ &= cn + 1 \end{aligned}$$
 Fails to show the induction hypothesis holds for  $n$
- Trick: *subtract* a lower order term: try to show  $T(n) \leq cn - b$  for some constants  $c$  and  $b \geq 0$   
 – Note:  $cn - b = O(n)$   

$$\begin{aligned} T(n) &\leq (c \lfloor n/2 \rfloor - b) + (c \lceil n/2 \rceil - b) + 1 \\ &= cn - 2b + 1 \\ &\leq cn - b \end{aligned}$$
 provided  $b \geq 1$
- This works because we have *strengthened* the induction hypothesis:  $T(n) \leq cn - b \rightarrow T(n) \leq cn$

11 September 2008

Winsborough CS 3343 Lecture 4

10

## Avoid Pitfall with Misusing Asymptotic Notation

- An incorrect proof that  $T(n) = 2T(\lfloor n/2 \rfloor) + n$  is  $O(n)$ :  
 – Guess  $T(n) \leq cn$  and argue as follows  

$$\begin{aligned} T(n) &\leq 2(c \lfloor n/2 \rfloor) + n \\ &\leq cn + n \\ &= O(n) \quad \text{Wrong!} \end{aligned}$$
- This is incorrect because we have not proven the induction hypothesis,  $T(n) \leq cn$

11 September 2008

Winsborough CS 3343 Lecture 4

11

## A Cool Trick: Changing Variables

- Scary recurrence:  $T(n) = 2T(\lfloor \sqrt{n} \rfloor) + \lg n$
- Trick 1: reason in terms of  $m = \lg n$ , which yields  $T(2^m) = 2T(2^{m/2}) + m$
- Trick 2: reason in terms of  $S(m) = T(2^m)$ :  
 $S(m) = 2S(m/2) + m$
- This now looks familiar, and indeed has the solution  $S(m) = O(m \lg m)$
- We now have  
 $T(n) = T(2^m) = S(m) = O(m \lg m) = O(\lg n \lg \lg n)$

11 September 2008

Winsborough CS 3343 Lecture 4

12