

Discrete Mathematical Structures CS 2233 Lecture Nineteen

Prof. William Winsborough
April 14, 2009

Business

- Recall: Homework 9 due Thursday 4/16
 - 4.3: 4, 6, 12, 22
- Read
 - 4.4, 7.3

14 April 2009

Winsborough CS 2233 Lecture 19

2

Recursive Programming

- Can write the same programs using iteration
 - Often algorithms are easier to design and understand using a recursive formulation
- Classic programming paradigm: Divide and Conquer
 - Decompose problem into subproblems
 - Solve each subproblem
 - Combine the results

14 April 2009

Winsborough CS 2233 Lecture 19

3

A Simple Iterative Program

- Consider the following algorithm:
 - $\text{power}(n)$ takes a non-negative integer and returns 3^n

```
power(n)
power = 1
for i = 1 to n
    power = power * 3
return power
```

- Cost (time complexity): $O(n)$

14 April 2009

Winsborough CS 2233 Lecture 19

4

Example Recursive Program

- Consider the following algorithm:
 - $\text{power}(n)$ takes a non-negative integer and returns 3^n

```
power(n)
if (n ≤ 0) return 1
return 3*power(n-1)
```

- Cost (time complexity): $T(n)$
 - Defined by the following *recurrence* (basically a recursive definition):
 $T(0) = 1$
 $T(n) = T(n-1) + 3$ (depending on which ops you count)
- Closed form solution: $T(n) = 3n+1 = O(n)$

14 April 2009

Winsborough CS 2233 Lecture 19

5

Improving the Example Program

- A faster version of $\text{power}(n)$:

```
power(n)
if (n ≤ 0) return 1
temp = ⌊n/2⌋
if 2*temp = n return (power(temp))^2
return 3*(power(temp))^2
```

- Cost:
 $T(0) = 1$
 $T(n) = T(\lfloor n/2 \rfloor) + d$, in which d is a small constant
- Claim: closed form solution is $T(n) = O(\lg n)$

14 April 2009

Winsborough CS 2233 Lecture 19

6

Solving Recurrences

- Two steps
 - Find a good guess
 - Prove it is correct
- Guess
 - Expansion method

$$T(n) = T(n-1) + 3 = T(n-2) + 3 + 3 = \dots = 1+3+\dots+3 \text{ (with } n \text{ copies of "+ 3")}$$
 - Expand each occurrence of $T(n)$ until you get to $T(0) = 1$

14 April 2009

Winsborough CS 2233 Lecture 19

7

Proving Recurrence

- Recall 1st recurrence:
 $T(0) = 1$ and $T(n) = T(n-1) + d$
- Want to prove $T(n) = O(n)$
 - We're not so worried about exactly which function in $O(n)$ $T(n)$ may happen to be
- Suppose $T(n) \leq en + h$, for some constants e and h
 - That would prove $T(n) = O(n)$, since $en + h = O(n)$
 - Use induction to prove that for some e and h , $T(n) \leq en + h$ satisfies the definition of $T(n)$

14 April 2009

Winsborough CS 2233 Lecture 19

8

Induction Proof

- Show $T(n) \leq en + h$ in which $T(n)$ is given by
 $T(0) = 1$ and $T(n) = T(n-1) + d$
- Basis: $T(0) \leq h$ if $h \geq 1$
- Step: $T(n) = T(n-1) + d$, by recurrence
 $\leq e(n-1) + h + d$, by induction assumption
 $= en + h - (e - d)$
 $\leq en + h$, as long as $e \geq d$
- It now follows that $T(n) = O(n)$

14 April 2009

Winsborough CS 2233 Lecture 19

9

Proving Recurrence

- Recall 2nd recurrence:
 $T(0) = 1$ and $T(n) = T(\lfloor n/2 \rfloor) + d$
- Guess: How many times can you divide n in half before reaching 1? Answer: $\lg n$
- Want to prove $T(n) = O(\lg n)$
- Suppose $T(n) \leq e \cdot \lg n + h$, for some constants e and h

14 April 2009

Winsborough CS 2233 Lecture 19

10

Induction Proof

- Show $T(n) \leq e \cdot \lg n + h$ in which $T(n)$ is given by
 $T(0) = 1$ and $T(n) = T(\lfloor n/2 \rfloor) + d$
- Basis: $T(0) \leq h$ if $h \geq 1$
- Step: $T(n) = T(\lfloor n/2 \rfloor) + d$, by recurrence
 $\leq e \cdot \lg(n/2) + h + d$, by induction assumption
 $= e \cdot (\lg n - \lg 2) + h + d$
 $= e \cdot \lg n + h - (e - d)$
 $\leq e \cdot \lg n + h$, as long as $d \leq e$

14 April 2009

Winsborough CS 2233 Lecture 19

11