# CS 483 - Data Structures and Algorithm Analysis

## A Short Word on Recurrences

R. Paul Wiegand

George Mason University, Department of Computer Science

February 22, 2006

# Outline

# What Is a Recurrence Relation?

**Definition (Coren *et al.* 2001)**

A *recurrence [relation]* is an equation or inequality that describes a function in terms of its values on smaller inputs.

**Examples:**

- $x(n) = x\left(\frac{n}{2}\right) + 5$ for $n > 0$, $x(1) = 0$
- $T(n) = \begin{cases} 9 & \text{if } n = 1 \\ 2T(n-2) + 2n & \text{if } n > 1 \end{cases}$
- Etc.

# Recurrences And Sequences

It is also useful to think in terms of *sequences*:

- A *sequence* is an ordered list of numbers
- E.g., $2, 4, 6, 8, 10, 12, \ldots$ (positive even numbers)
- We often refer to a sequence using a variable, say $x$, and we often indicate an element of the sequence with an index, $x_i$
- We might also use something called the *generic term*, $x(n)$ — where $x(n)$ represents the $n^{th}$ number in the $x$ sequence
- We can then use the generic term as a *function* to help define the sequence: $x(n) = 2n$ for $n \geq 0$
- Alternatively, we could define the sequence by showing how to step from one element to another: $x(n) = x(n-1) + n$ for $n > 0, \quad x(0) = 0$
- It is clear now why an initial condition is needed ... there can be many sequences defined by a recurrence, the initial condition tells you which one by specifying the starting position of the sequence

# Why and What Now?

"This is complicated. Why would I express a sequence or a function in this way? What do I do with it now?"

- Sometimes it is the most natural way to so
- For example: When analyzing recursive functions, it is typically very natural to express the running time as a recurrence
- On the other hand, it is a lot easier to deal with the *closed form* (an algebraic form where the function appears only on the left-hand-side of the [in]equality, and where more complicated notational elements such as summations are resolved)
- Moreover, we need the closed form to express the *order of growth* of an algorithm's efficiency properly

# What Is *Solving* A Recurrence?

- Simply, solving a recurrence is to find the closed form of the relation
- An *exact* solution will be the fully specified algebraic closed form of the recurrence
  - For example: Find the exact solution of $x(n) = x(n-1) + n$ for $n > 0$ subject to initial condition $x(0) = 0$
  - Answer: $x(n) = \frac{n(n+1)}{2}$ for $n \geq 0$
- But typically, we are interested in asymptotic bounds on the solution
  - For example: Find the asymptotic solution of
    $$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 \end{cases}$$
  - Answer: $\Theta(n \lg n)$

# Forward Substitution

- We start with initial term(s) of a sequence given by initial conditions
- We use the recurrence equation itself to generate several terms
- We look for a pattern that can be expressed in closed form

**Example:** $x(n) = 2x(n-1) + 1$ for $n > 1$, $x(1) = 1$

$$\begin{aligned} x(1) &= 1 \\ x(2) &= 2 \cdot x(1) + 1 = 2 \cdot 1 + 1 = 3 \\ x(3) &= 2 \cdot x(2) + 1 = 2 \cdot 3 + 1 = 7 \\ x(4) &= 2 \cdot x(3) + 1 = 2 \cdot 7 + 1 = 15 \end{aligned}$$

Each number is one less than consecutive powers of two (2, 4, 8, 16, ...), so the solution is probably something like $x(n) = 2^n - 1$.

# Backward Substitition

- We start at the penultimate step of the sequence (e.g., $x(n-1)$)
- We express the final step in terms of the recurrence relation
- We repeat this process for the ante-penultimate step, etc.

**Example:** $x(n) = x(n-1) + n$ for $n > 1$, $\mathrm{x}(1) = 1$

$$
\begin{aligned}
x(n) &= x(n-1) + n \\
&= [x(n-2) + n - 1] + n = x(n-2) + (n-1) + n \\
&= [x(n-3) + n - 2] + (n-1) + n = x(n-3) + (n-2) + (n-1) + n \\
& \quad \text{after } i \text{ substitutions ...} \\
&\rightsquigarrow x(n-i) + (n-i+1) + (n-i+2) + \cdots + n \\
& \quad \text{...to the initial condition} \\
&\rightsquigarrow x(0) + 1 + 2 + \cdots + n = n(n+1)/2
\end{aligned}
$$

# Solving versus Proving

- Technically, to "solve" a recurrence is just to elicit its closed form solution
- When someone else looks at your solution (or you 15 minutes later), you'd like to have a way to convince him or her that it is correct
- To do that, you must *prove* it is true
- Substitution and recurrence trees are *not* proofs, they merely help with intuition ... they help you *guess* the solution
- Typically, we prove that a closed form solution is (asymptotically) correct by *induction*...

# Some Preliminaries

To obtain closed form asymptotic bounds on a recurrence, we use induction and the definitions for Big-O and Big-$\Omega$.

**Definition (MathWorld)**

The truth of an infinite sequence of propositions $P_i$ for $i = \{1, \ldots, \infty\}$ is established if (1) $P_1$ is true, and (2) $P_k \Rightarrow P_{(k+1)}$ for all $k$. This principle is sometimes also known as the method of induction.

**Definition**

$O\left(g(n)\right) = \{t(n) : \exists c, n_0 > 0 \text{ such that } 0 \leq t(n) \leq c \cdot g(n) \; \forall n \geq n_0\}$.

**Definition**

$\Omega\left(g(n)\right) = \{t(n) : \exists c, n_0 > 0 \text{ such that } 0 \leq c \cdot g(n) \leq t(n) \; \forall n \geq n_0\}$.

# Proof By Induction

- Consider the recurrence relation
- Posit a guess for the asymptotic closed form solution
- Write down the inequality from the Big-O/$\Omega$ definition(s)
- Use the definition and substitution to show that the definition holds after a step of the recurrence
- Indicate the constant values for which the definition holds

# Proof By Induction

- Consider the recurrence relation
- Posit a guess for the asymptotic closed form solution
- Write down the inequality from the Big-O/Ω definition(s)
- Use the definition and substitution to show that the definition holds after a step of the recurrence
- Indicate the constant values for which the definition holds

**Example:**

- Recurrence: $T(n) = 2T(\lfloor n/2 \rfloor) + n$
- Asymptotic solution: $T(n) \in O(n \lg n)$
- Big-O Definition: $T(n) \leq cn \lg n$
- Given it holds for $n$, assume it holds for $\lfloor n/2 \rfloor$ : $T(\lfloor n/2 \rfloor) \leq c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$

# Proof By Induction

- Consider the recurrence relation
- Posit a guess for the asymptotic closed form solution
- Write down the inequality from the Big-O/Ω definition(s)
- Use the definition and substitution to show that the definition holds after a step of the recurrence
- Indicate the constant values for which the definition holds

**Example:**

- Recurrence: $T(n) = 2T(\lfloor n/2 \rfloor) + n$
- Asymptotic solution: $T(n) \in O(n \lg n)$
- Big-O Definition: $T(n) \leq cn \lg n$
- Given it holds for $n$, assume it holds for $\lfloor n/2 \rfloor$ : $T(\lfloor n/2 \rfloor) \leq c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$

- Substituting into the recurrence:

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