# CSC 8301- Design and Analysis of Algorithms

Lecture 3

## Techniques for efficiency analysis of recursive algorithms

## Time efficiency of recursive algorithms General Plan

- Decide on parameter *n* indicating *input size*
- Identify algorithm's *basic operation*
- Determine *worst*, *average*, and *best* case for inputs of size *n* to analyze them separately if needed
- Set up a *recurrence relation* and *initial condition(s)* for *C*(*n*) – the number of times the basic operation will be executed for an input of size *n*
- Solve the recurrence to obtain a closed form or estimate the order of growth of the solution (see Appendix B)

# Example 1: Recursive evaluation of *n*!

Definition: $n! = 1 * 2 * \ldots *(n-1) * n$ for $n \geq 1$ and $0! = 1$

Recursive definition of $n$!: $F(n) = F(n-1) * n$ for $n \geq 1$ and
$F(0) = 1$

```
ALGORITHM F(n)
    //Computes n! recursively
    //Input: A nonnegative integer n
    //Output: The value of n!
    if n = 0 return 1
    else return F(n − 1) ∗ n
```

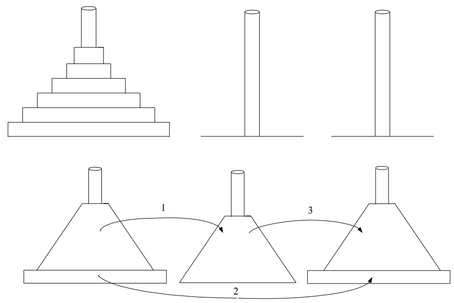
Input size:
Basic operation:
Recurrence for time complexity:

## Solving the recurrence (backward substitution)

$M(n) = M(n-1) + 1$

$M(0) = 0$

## Example 2: Tower of Hanoi Puzzle




Goal: move all *n* disks to peg 3
- can use peg 2 in the process

Restriction:
- cannot place a disk on top of a smaller one

**Recursive solution:**

Recurrence for the number of moves:

## Solving the recurrence (backward substitution)

$M(n) = 2\,M(n\text{-}1) + 1$

$M(1) = 1$

## Example 3: Counting binary digits

**ALGORITHM** *BinRec*($n$)
    //Input: A positive decimal integer $n$
    //Output: The number of binary digits in $n$'s binary representation
    **if** $n = 1$ **return** 1
    **else return** $BinRec(\lfloor n/2 \rfloor) + 1$

Recurrence for time complexity:

## Fibonacci numbers

The *Fibonacci numbers*:

0, 1, 1, 2, 3, 5, 8, 13, 21, …

Fibonacci recurrence:

$F(n) = F(n\text{-}1) + F(n\text{-}2)$

$F(0) = 0$

$F(1) = 1$

**2nd *order linear homogeneous recurrence relation with constant coefficients* (2nd order LHRRCC)**

# Solving 2nd order LHRRCC

<u>Definition</u> *2nd order linear homogeneous recurrence with constant coefficients* is a recurrence of the form:

$$ax_n + bx_{n-1} + cx_{n-2} = 0$$

where $a$, $b$, $c$ are real numbers (called the coefficients), $a \neq 0$.

Unless $b = c = 0$, this equation has infinitely many solutions called the *general solution*. A formula expressing this solution depends on the root of the quadratic equation called the *characteristic equation* for the above recurrence:

$$ar^2 + br + c = 0$$

<u>Theorem</u> If the characteristic equation has two real roots $r_1$, $r_2$
then $x_n = c_1 r_1^n + c_2 r_2^n$ ($c_1$, $c_2$ derived from initial conditions)

If the characteristic equation has one root $r$,
then $x_n = c_1 r^n + c_2 n r^n$

# LHRRCC Example

Find the general solution for

$x(n) = 5\,x(n\text{-}1) - 6x(n\text{-}2)$
$x(0) = 9$
$x(1) = 20$

# Application to the Fibonacci numbers

The Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, …

The Fibonacci recurrence: $F(n) - F(n\text{-}1) - F(n\text{-}2) = 0$

The characteristic equation: $r^2 - r - 1 = 0$

The roots: $r_{1,2} = (1\pm\sqrt{5})/2$

The general solution: $F(n) = c_1((1+\sqrt{5})/2)^n + c_2((1-\sqrt{5})/2)^n$

The particular solution – use initial conditions $F(0) = 0$, $F(1) = 1$ to obtain $c_1$ and $c_2$ after solving a system of two linear equations in two unknowns.

$$F(n) = (\phi^n - \phi_1{}^n)/\sqrt{5}$$

where $\phi = (1+\sqrt{5})/2) \approx 1.618$ (*golden ratio*),
$\phi_1 = (1-\sqrt{5})/2) \approx -0.618$.

# Computing Fibonacci numbers

Definition-based recursive algorithm

**ALGORITHM** $F(n)$
//Computes the $n$th Fibonacci number recursively by using its definition
//Input: A nonnegative integer $n$
//Output: The $n$th Fibonacci number
**if** $n \leq 1$ **return** $n$
**else return** $F(n-1) + F(n-2)$

**Recurrence for time complexity:**

## Computing Fibonacci numbers (cont.)

Nonrecursive brute-force algorithm

**ALGORITHM** *Fib*($n$)

//Computes the $n$th Fibonacci number iteratively by using its definition
//Input: A nonnegative integer $n$
//Output: The $n$th Fibonacci number
$F[0] \leftarrow 0$; $F[1] \leftarrow 1$
**for** $i \leftarrow 2$ **to** $n$ **do**
    $F[i] \leftarrow F[i-1] + F[i-2]$
**return** $F[n]$

**Summation for time complexity:**

## Computing Fibonacci numbers (cont.)

Explicit formula algorithm based on

$$F(n) = \varphi^n / \sqrt{5} \text{ rounded to the nearest integer}$$

Logarithmic algorithm based on formula:

$$\begin{pmatrix} F(n-1) & F(n) \\ F(n) & F(n+1) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n$$

**with an efficient way of computing matrix powers**

# Homework

Exercises

- 2.4: 1, 3, 4, 8, 9, 12
- 2.5: 3, 7, 8

Reading:

- Sections 2.4 and 2.5
- pp. 479–485 in Appendix B

Next: Chapter 3