

# **Chapter 4**

# **Induction and Recursion**

# Objectives

- Mathematical Induction
- Strong Induction and Well-Ordering
- Recursive Definitions and Structural Induction
- Recursive Algorithms

## 4.1- Mathematical Induction

- Introduction
- Mathematical Induction
- Examples of Proofs by Mathematical Induction

# Principle of Mathematical Induction

## Principle of Mathematical Induction

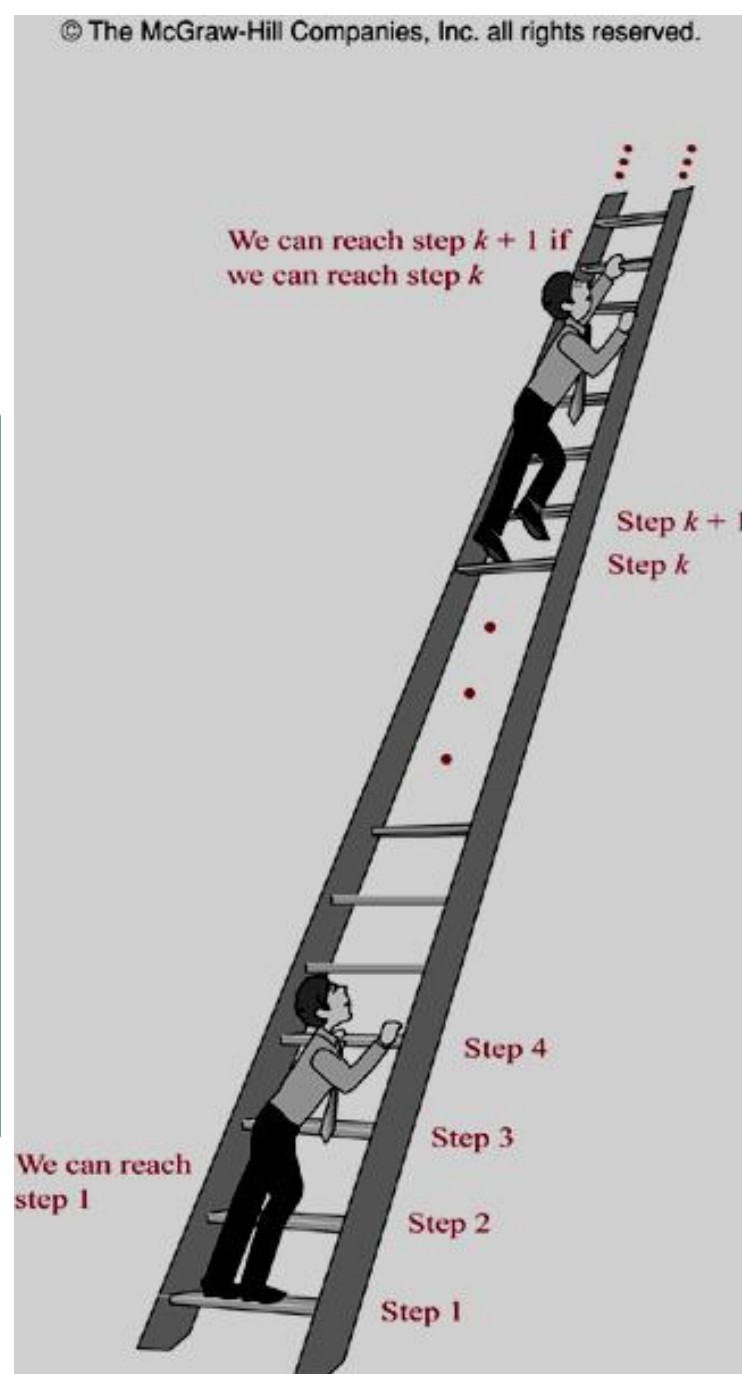
To prove  $P(n)$  is true for all possible integers  $n$ , where  $P(n)$  is a propositional function, we complete two step:

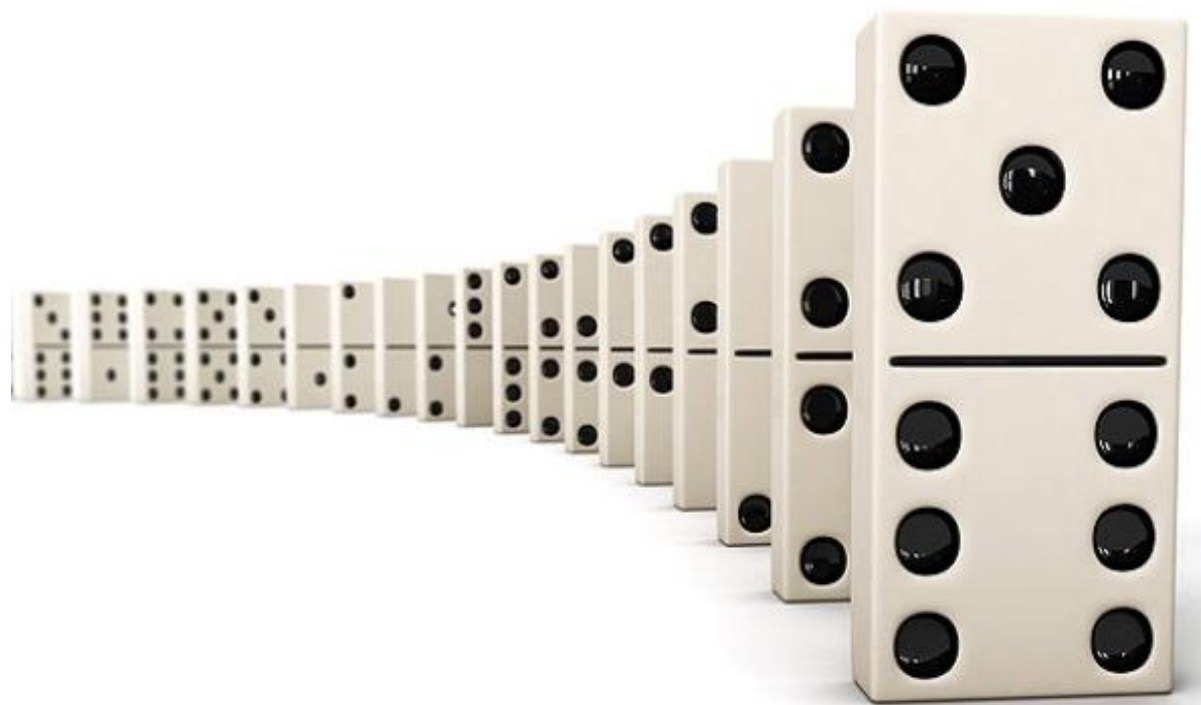
### **Basis step:**

Verifying  $P(1)$  is true

### **Inductive step:**

Show  $P(k) \rightarrow P(k+1)$  is true for all  $k > 0$





# Induction: Example 1

Prove that  $1 + 2 + 3 + \dots + n = n(n+1)/2$  for all integers  $n > 0$

*Solution.*

Let  $P(n) = "1 + 2 + 3 + \dots + n = n(n+1)/2"$ .

- **Basis step:**  $P(1) = "1 = 1(1+1)/2" \rightarrow \text{true}$
- **Inductive step:** With arbitrary  $k > 0$ ,  
 $P(k) = "1 + 2 + \dots + k = k(k+1)/2"$  is true.

We have

$$\begin{aligned}
 \underline{1+2+3+\dots+k} + (k+1) &= k(k+1)/2 + (k+1) \\
 &= [k(k+1) + 2(k+1)]/2 \\
 &= (k+1)(k+2)/2 \\
 &= (k+1)((k+1)+1)/2
 \end{aligned}$$

$P(k+1) = "1 + 2 + 3 + \dots + k + 1 = (k+1)(k+2)/2"$  is true.

$P(k) \rightarrow P(k+1)$ : true

Proved.

## Example 2 p.268

- Conjecture a formula for the sum of the first  $n$  positive odd integers. Then prove your conjecture using mathematical induction.

- *Solution.*

The sum of the first  $n$  positive odd integers for  $n=1, 2, 3, 4, 5$  are:

$$1=1, \quad 1+3=4, \quad 1+3+5=9,$$

$$1+3+5+7=16, \quad 1+3+5+7+9=25.$$

- *Conjecture:*  $1+3+5+\dots+(2n-1)=n^2$ .
- *Proof.* Let  $P(n) = "1+3+5+\dots+(2n-1)=n^2."$

- Basis step.  $P(1) = "1=1"$  is true.

- Inductive step.  $(P(k) \rightarrow P(k+1))$  is true.

Suppose  $P(k)$  is true for arbitrary  $k > 0$ . That is, " $1+3+5+\dots+(2k-1)=k^2$ "

We have,  $\underline{1+3+5+\dots+(2k-1)} + (2k+1) = \underline{k^2} + 2k+1 = (k+1)^2$ .

So,  $P(k+1)$  is true.

Proved.

## Induction: Examples 2..13 – pages: 268..278

- $1+3+5+\dots+(2n-1) = n^2$
- $2^0+2^1+2^2+2^3+\dots+2^n = \sum 2^n = 2^{n+1}-1$
- $\sum ar^j = a + ar + ar^2 + \dots + ar^n = (ar^{n+1}-a)/(r-1)$
- $n < 2^n$
- $2^n < n! , n > 3$
- $n^3-n$  is divisible by 3,  $n$  is positive integer
- The number of subsets of a finite set: a set with  $n$  elements has  $2^n$  subsets.
- ....
- Let  $H(j) = 1/1 + 1/2 + 1/3 + \dots + 1/j$   
Prove that  $H(2^n) \geq 1 + n/2$  for all  $n \geq 0$



## 4.2- Strong Induction and Well-Ordering

### Principle of Strong Induction

To prove  $P(n)$  is true for all positive integers  $n$ , where  $P(n)$  is a propositional function, two steps are performed:

#### **Basis step:**

Verifying  $P(1)$  is true

#### **Inductive step:**

Show  $[P(1) \wedge P(2) \wedge \dots \wedge P(k)] \rightarrow P(k+1)$  is true for all  $k > 0$

# Strong Induction: Example 1

**Prove that if  $n$  is an integer greater than 1, then  $n$  can be written as the product of primes**

$P(n)$  :  $n$  can be written as the product of primes

Basis steps:  $P(2) = \text{true}$  //  $2=2$  , product of 1 primes

$P(4) = \text{true}$  //  $4=2.2$

Inductive step:

Assumption:  $P(j)=\text{true}$  for all positive  $j \leq k$

- Case  $k+1$  is a prime  $\rightarrow P(k+1) = \text{true}$
- Case  $k+1$  is a composite  $\rightarrow k+1 = ab$ ,  $2 \leq a \leq b < k+1$   
 $\rightarrow P(k)$  is true

## Strong Induction: Example 2

Prove that every amount of postage of 12 cents or more can be formed using just 4-cents and 5-cents stamps

$P(n)$  : “  $n$  cents can be formed using just 4-cent and 5-cent stamps”

$P(12)$  is true :  $12 \text{ cents} = 3 \cdot 4 \text{ cents}$

$P(13)$  is true :  $13 = 2 \cdot 4 + 1 \cdot 5$

$P(14)$  is true:  $14 = 1 \cdot 4 + 2 \cdot 5$

$P(15)$  is true:  $15 = 3 \cdot 5$

Assumption:  $P(j)$  is true with  $12 \leq j \leq k$ ,  $k > 15 \Rightarrow P(k-3)$  is true

$k+1 = (k-3) + 4$ ,  $k > 12$

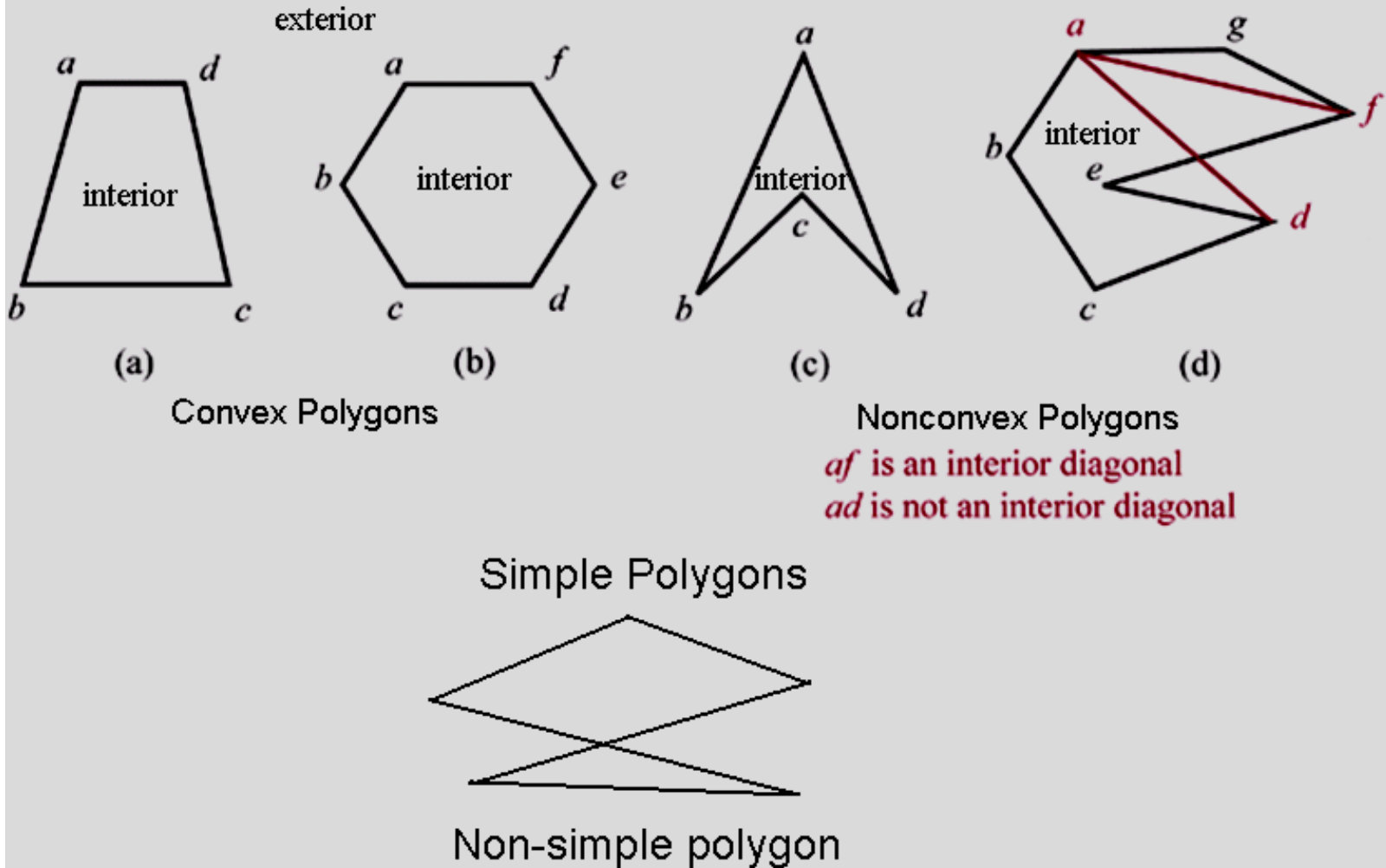
$\rightarrow P(k+1)$  is true because  $k+1$  is the result of adding a 4-cent stamp to the amount  $k-3$

$\rightarrow$  Proved

# Using Strong Induction in Computational Geometry

## • Definitions

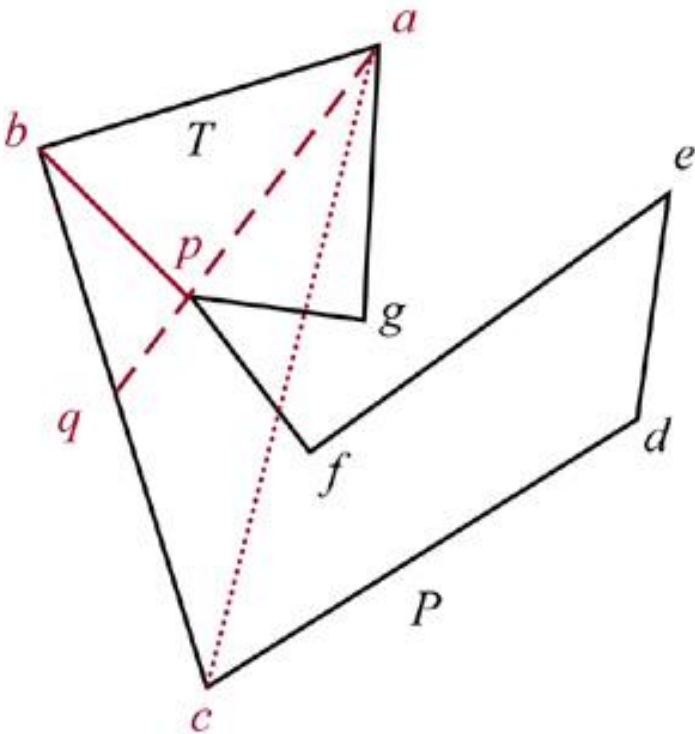
© The McGraw-Hill Companies, Inc. all rights reserved.



# Strong Induction in Computational Geometry

**Lemma 1:** Every simple polygon has an interior diagonal.

© The McGraw-Hill Companies, Inc. all rights reserved.



$T$  is the triangle  $abc$

$p$  is the vertex of  $P$  inside  $T$  such that the  $\angle bap$  is smallest

$bp$  must be an interior diagonal of  $P$

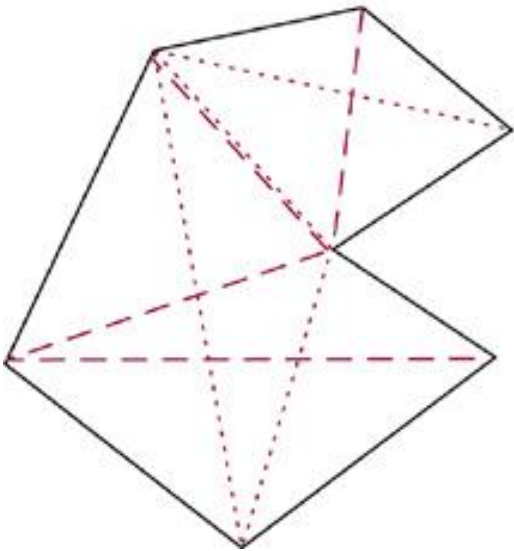
Proof: page 290

# Strong Induction in Computational Geometry

## Theorem 1:

A simple polygon with  $n$  sides, where  $n$  is integer with  $n \geq 3$ , can be triangulated into  $n-2$  triangles

© The McGraw-Hill Companies, Inc. all rights reserved.



Two different triangulations of a simple polygon with seven sides into five triangles, shown with dotted lines and with dashed lines, respectively

**Proof: page 289**

## 4.3- Recursive Definition and Structural Induction

- Introduction
- Recursively Defined Functions
- Recursively Defined Sets and Structures
- Structural Induction
- Generalized Induction
- Recursive Algorithms

# Recursion: Introduction

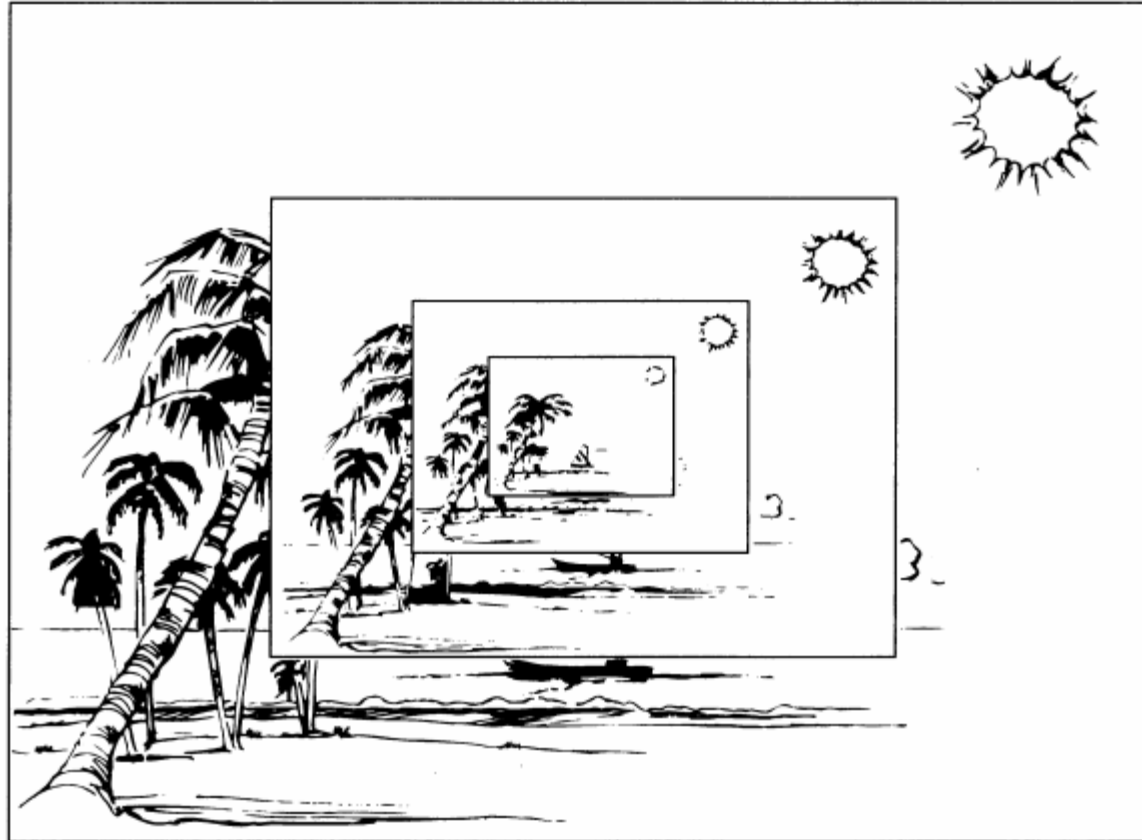
- Objects/ functions may be difficultly defined.
- Define an object/function in terms of itself
- Examples:

$$2^n = \begin{cases} 1, n = 0 \\ 2 \cdot 2^{n-1}, n > 0 \end{cases}$$

$$\sum_{i=0}^n i = \begin{cases} 0, n = 0 \\ n + \sum_{i=0}^{n-1} i, n > 0 \end{cases}$$

```
#include <stdio.h>
#include <conio.h>
double compute2n( int n)
{ if (n==0) return 1;
  return 2*compute2n(n-1);
}
void main()
{ clrscr();
  int n;
  scanf("%d",&n);
  double r=compute2n(n);
  printf("%lf",r);
  getch();
}
```





**FIGURE 1** A Recursively Defined Picture.

# Recursively Defined Functions

- Recursive ( inductive) function

Two steps to define a function with the set of nonnegative integers as its domain:

- Basis step: Specify the value of the function at zero.
- Recursive step: Give a rule for finding its value at an integer from its values at smaller integers
- Example: Find  $f(1)$ ,  $f(2)$ ,  $f(4)$ ,  $f(6)$  of the following function:

$$f(n) = \begin{cases} n, & n < 3 \\ 3n + f(n-1), & n \geq 3 \end{cases}$$

# Recursively Defined Functions

- Example: Give the recursive definition of  $\sum a_i$ ,  $i=0..k$
- Basis step:  $\sum a_i = a_0$ ,  $i=0$
- Inductive step:

$$\boxed{a_0 + a_1 + \dots + a_{k-1}} + a_k$$

$$(\sum a_i, i=0..k-1)$$

$$\sum a_i = a_k + (\sum a_i, i=0..k-1)$$

# Recursively Defined Functions

**Definition 1: Fibonacci numbers**  $f(n) = \begin{cases} 1, n = 0, 1 \\ f(n-1) + f(n-2), n > 1 \end{cases}$

**Theorem 1: Lamé's theorem:** Let  $a, b$  be integers,  $a \geq b$ .  
Then the number of divisions used by the Euclidean algorithm to find  $\gcd(a, b)$  is less than or equal to five times the number of decimal digits in  $b$ .

Proof: page 298

Example  $\gcd(25, 7)$ ,  $b = 7$ , 1 digit

$x$	$y$	$r$
25	7	$25 \bmod 7 = 4$
7	4	$7 \bmod 4 = 3$
4	3	$4 \bmod 3 = 1$
3	1	$3 \bmod 1 = 0$ ( 4 divisions)
1	0	Stop

```

procedure gcd(a,b)
x:=a; y:=b
while y ≠ 0
begin
  r := x mod y
  x:=y
  y:= r
end { gcd(a,b) is x}

```

# Recursively Defined Sets and Structures

- Example  $S = \{3, 6, 9, 12, 15, 18, 21, \dots\}$

Step 1:  $3 \in S$

Step 2: If  $x \in S$  and  $y \in S$  then  $x + y \in S$

- **Definition 2:** The set  $\Sigma^*$  of string over alphabet  $\Sigma$  can be defined recursively by:

**Basis step:**  $\lambda \in \Sigma^*$ ,  $\lambda$  is the empty string with no symbols

**Recursive step:** If  $w \in \Sigma^*$  and  $x \in \Sigma$  then  $wx \in \Sigma^*$

**Example:**  $\Sigma = \{0, 1\} \rightarrow \Sigma^*$  is the set of string made by 0 and 1 with arbitrary length and arbitrary order of symbols 0 and 1

# Recursively Defined Sets and Structures

- **Definition 3:** String Concatenation ( $\cdot$ )

**Basis step:** If  $w \in \Sigma^*$  then  $w \cdot \lambda = w$ ,  $\lambda$  is the empty string

**Recursive step:** If  $w_1 \in \Sigma^*$  and  $w_2 \in \Sigma^*$  and  $x \in \Sigma$   
then  $w_1 \cdot (w_2 x) = (w_1 \cdot w_2) x$

**Example:**  $\Sigma = \{0, 1\} \rightarrow \Sigma^*$  is the set of string made by 0 and 1 with arbitrary length and arbitrary order of symbols 0 and 1

## 4.4- Recursive Algorithms

- **Definition 1:** An algorithm is called recursive if it solves a problem by reducing it to an instance of the same problem with smaller input.

**Example: Recursive algorithm for computing  $n!$**

```
procedure factorial (n: nonnegative integer)
  if n=0 then factorial(n) := 1
  else factorial(n) = n.factorial(n-1)
```

$n! = 1, n=0$

$n! = 1.2.3.4...n = n.(n-1)!, n>0$

# Recursive Algorithms...

**Example: Recursive algorithm for computing  $a^n$**

```
procedure power (a: nonzero real number  
                n: nonnegative integer)  
  if n=0 then power(a,n) :=1  
  else power(a,n)=a.power(a,n-1)
```

$a^n = 1, n=0$

$a^n = a \cdot a \cdot \dots \cdot a = a \cdot a^{n-1}, n>0$



# Recursive Algorithms...

**Example: Recursive algorithm for computing  $b^n \bmod m$**   
 $m \geq 2, n \geq 0, 1 \leq b < m.$

$$b^n \bmod m = (b \cdot (b^{n-1} \bmod m)) \bmod m$$

$$b^0 \bmod m = 1$$

Using division to improve performance: (  $n$  steps backward to 0 faster)

If  $n$  is even  $\rightarrow b^n = b^{n/2} \cdot b^{n/2}$

$$\rightarrow b^n \bmod m = ((b^{n/2} \bmod m) \cdot (b^{n/2} \bmod m)) \bmod m$$

$$\rightarrow b^n \bmod m = (b^{n/2} \bmod m)^2 \bmod m$$

If  $n$  is odd  $\rightarrow b^n = b \cdot b^{\lfloor n/2 \rfloor} \cdot b^{\lfloor n/2 \rfloor}$

$$\rightarrow b^n \bmod m = (((b^{\lfloor n/2 \rfloor} \bmod m)^2 \bmod m) \cdot (b \bmod m)) \bmod m$$

Algorithm: page 313

# Recursive Algorithms...

**Example: Recursive algorithm for computing  $\text{gcd}(a,b)$**   
 **$a,b$ : non negative integer,  $a < b$**

If  $a > b$  then swap  $a, b$   
 $\text{gcd}(a, b) = b$ ,  $a = 0$   
 $\text{gcd}(a, b) = \text{gcd}(b \bmod a, a)$

Algorithm: page 313

# Recursive Algorithms...

**Example: Recursive algorithm for linear searching the value  $x$  in the sequence**

**$a_i, a_{i+1}, \dots, a_j$ , sub-sequence of  $a_n$ .**

$$1 \leq i \leq n, 1 \leq j \leq n$$

$i > j \rightarrow \text{location} = 0$

$a_i = x \rightarrow \text{location} = i$

$\text{location}(i, j, x) = \text{location}(i+1, j, x)$

Algorithm: page 314 – You should modify it.

# Recursive Algorithms...

**Example: Recursive algorithm for binary searching the value  $x$  in the increasingly ordered sequence  $a_i, a_{i+1}, \dots, a_{j-1}$ , subsequence of  $a_n$ .  $1 \leq i \leq n, 1 \leq j \leq n$**

```

procedure binary-search( $x, i, j$ )
if  $i > j$  then location=0
 $m = \lfloor (i+j)/2 \rfloor$ 
if  $x = a_m$  then location =  $m$ 
else if  $x < a_m$  then location= binary-search( $x, i, m-1$ )
else location= binary-search( $x, m+1, j$ )
    
```

**Algorithm: page 314 – You should modify it.**

# Proving Recursive Algorithms Correct

- Using mathematical induction.
- Example: prove the algorithm that computes  $n!$  is correct.

```
procedure f (n: nonnegative integer)
  if n=0 then f(n) :=1
  else f(n) = n.f(n-1)
```

If  $n=0$ , first step of the algorithm tells us  $f(0)=1 \rightarrow \text{true}$

Assuming  $f(n)$  is true for all  $n \geq 0$

$f(n) = 1.2.3 \dots (n)$

$(n+1).f(n) = 1.2.3 \dots n.(n+1) = (n+1)!$

$f(n+1) = (n+1)!$

Conclusion:  $f(n)$  is true for all integer  $n$ ,  $n \geq 0$

More examples: Page 315

# Recursion and Iteration

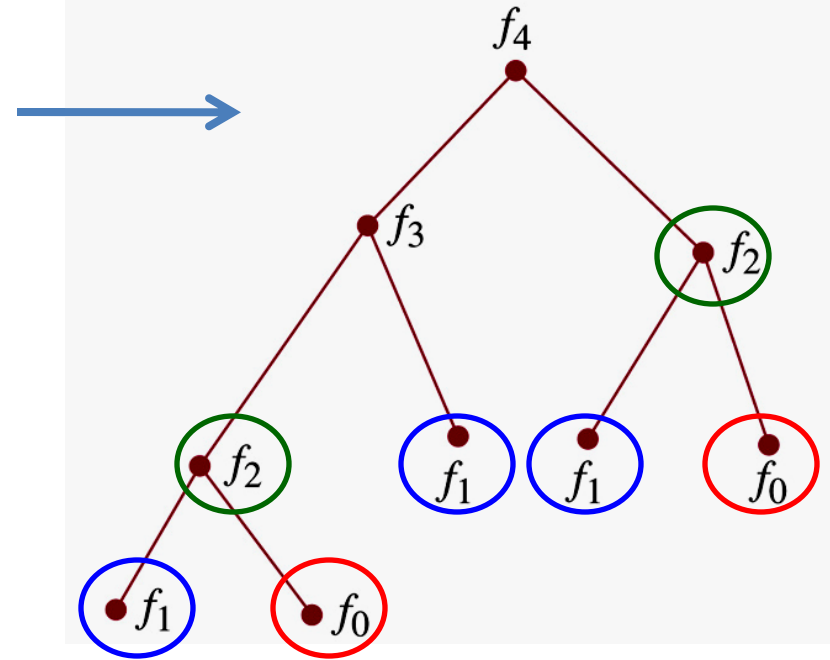
```

procedure rfibo (n: nonnegative integer)
If n=0 then rFibo(0)=0
Else if n=1 then rFibo(1)=1
Else rFibo(n) := rFibo(n-2) + rFibo(n-1)
    
```

```

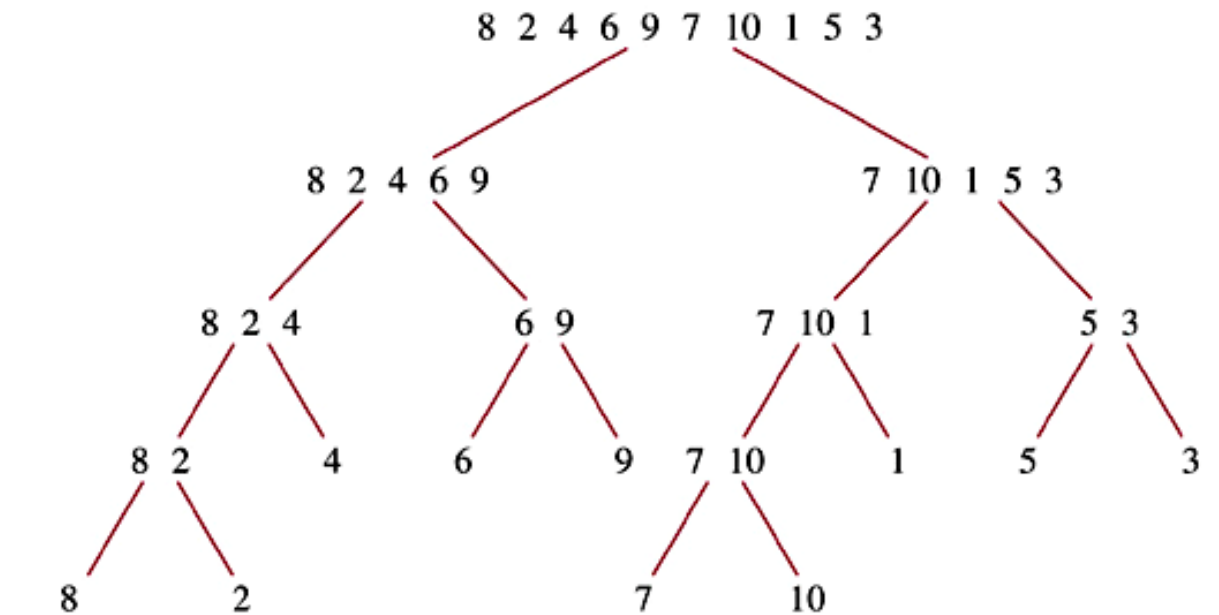
procedure iFibo (n: nonnegative integer)
If n=0 then y:=0
Else if n=1 then y:=1
Else Begin
    x:=0 ; y:=1
    for i:= 2 to n
        Begin
            z:= x+y; x:= y; y:=z
        End
    End { iFibo(n) = z }
    
```

© The McGraw-Hill Companies, Inc. all rights reserved.



**Recursive algorithm uses far more computation than iterative one**

# Merge Sort



Procedure **mergeSort** (  
     $L = a_1 \dots a_n$ )

If  $n > 1$  then

Begin

$m := \lfloor n/2 \rfloor$

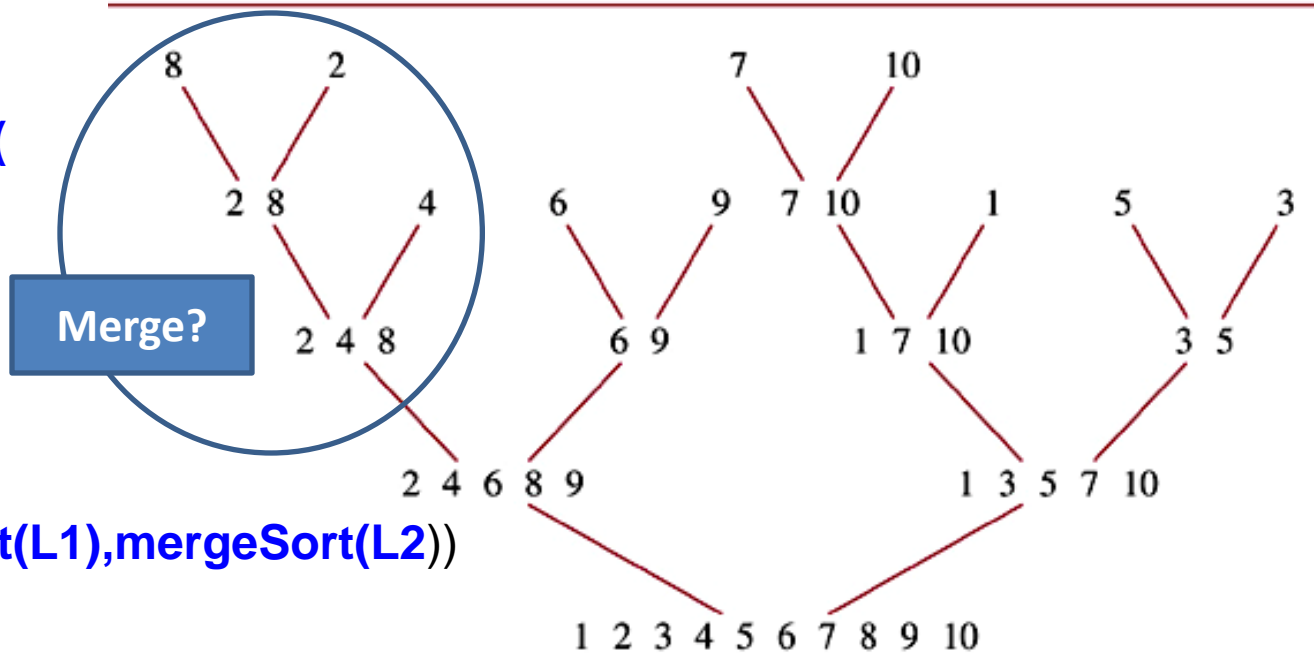
$L1 := a_1 \dots a_m$

$L2 := a_{m+1} \dots a_n$

$L := \text{merge}(\text{mergeSort}(L1), \text{mergeSort}(L2))$

End

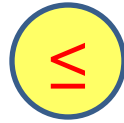
{L is sorted}



L1: 1 2 2 5 7 9 12 15 17 19

L2: 3 5 8 9 11 15

# Merge Sort



L : 1 2 2 3 5 5 7 8 9 9 11 12 15 15 17 19

- Merge two sorted lists L1, L2 to list L, an increasing ordered list.

procedure Merge (L1, L2: sorted list)

L:= empty list

While L1 and L2 are both not empty

Begin

**remove smaller of first element of L1 and L2**

**and put it to the right end of L**

**if removal of this element makes one list empty  
then remove all elements from the other list and  
append them to L**

End { L has increasing order }

**Theorem 1: The number of comparisons needed to merge sort a list with n elements is  $O(n \log n)$**



# Summary

- Mathematical Induction
- Strong Induction and Well-Ordering
- Recursive Definitions and Structural Induction
- Recursive Algorithms

**Thanks**