Chapter 4

Induction and Recursion

Objectives

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

Principle of Mathematical Induction

Principle of Mathematical Induction

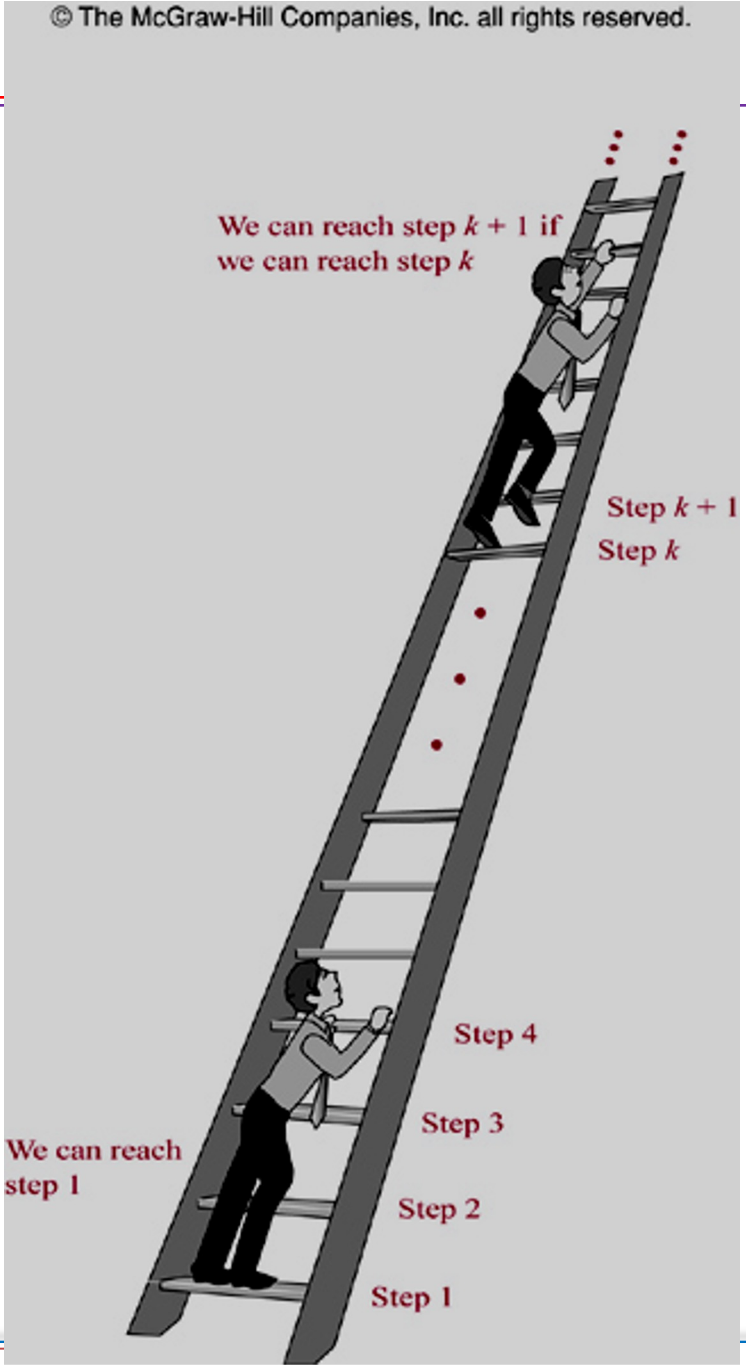
To prove P(n) is true for all possitive integers n, where P(n) is a propositional function, we complete two step:

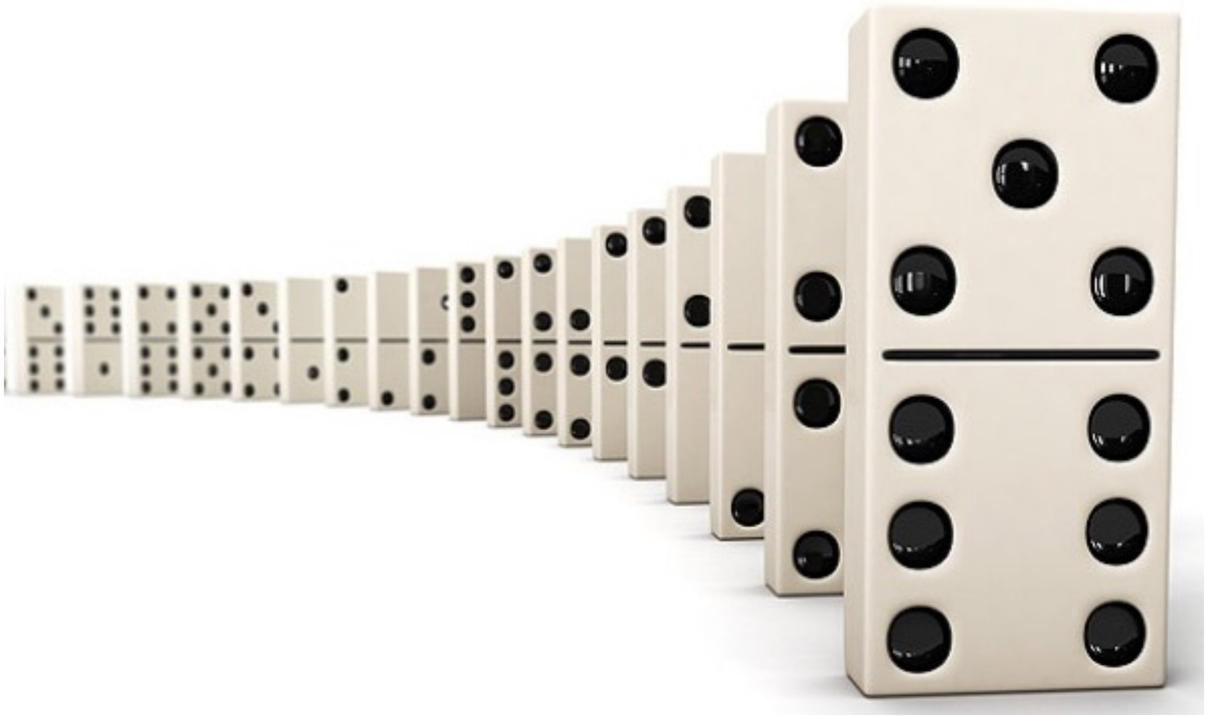
Basis step:

Verifying P(1) is true

Inductive step:

Show P(k) → P(k+1) is true for all k>0





Induction: Example 1

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

Solution.

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

- Basis step: P(1) = "$1 = 1(1+1)/2$" ➔ true
- Inductive step: With arbitrary $k>0$,

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

We have

$$\underline{1+2+3+\ldots+k}+(k+1)= k(k+1)/2 + (k+1)$$
$$= [k(k+1)+ 2(k+1)]/2$$
$$= (k+1)(k+2)/2$$
$$= (k+1)((k+1)+1)/2$$

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

P(k) → 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, 1+3=4, 1+3+5=9,

  1+3+5+7=16, 1+3+5+7+9=25.
- *Conjecture*: **1+3+5+…+(2n-1)=n^2.**
- *Proof*. Let P(n)="**1+3+5+…+(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+…+(2k-1)=k^2"

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

  So, P(k+1) is true.

Proved.

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

- $1+3+5+\ldots+(2n-1) = n^2$
- $2^0+2^1+2^2+2^3+\ldots+2^n = \sum 2^n = 2^{n+1}-1$
- $\sum ar^j = a + ar + ar^2 + \ldots + ar^n = (ar^{n+1}-a)/(r-1)$
- $n < 2^n$
- $2^n < n!\,,\ n>3$
- 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 $H(j)= 1/1 + ½ + 1/3 + \ldots + 1/j$

  Prove that $H(2^n)$ ⍰ $1 + n/2$ for all n ⍰ 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 $[P(1) \wedge P(2) \wedge \ldots \wedge P(k)] \rightarrow P(k+1)$ 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) = true // 2=2 , product of 1 prime

P(4) = true // 4=2.2

Inductive step:

Assumption: P(j)=true for all positive $j \le k$

- Case k+1 is a prime ➔ P(k+1) =true
- Case k+1 is a composite ➔ k+1= ab, $2 \le a \le b < k+1$

➔ 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 cents = 3. 4 cents

P(13) is true : 13= 2.4 + 1.5

P(14) is true: 14= 1.4 + 2.5

P(15) is true: 15= 3.5

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

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

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

→ **Proved**

10. a) Find a formula for

$$\frac{1}{1\cdot 2}+\frac{1}{2\cdot 3}+\cdots+\frac{1}{n(n+1)}$$

by examining the values of this expression for small values of n.

b) Prove the formula you conjectured in part (a).

11. a) Find a formula for

$$\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\cdots+\frac{1}{2^n}$$

by examining the values of this expression for small values of n.

b) Prove the formula you conjectured in part (a).

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:

$$2^n = \begin{cases} 1, n = 0 \\ 2.2^{n-1}, n > 0 \end{cases}$$

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

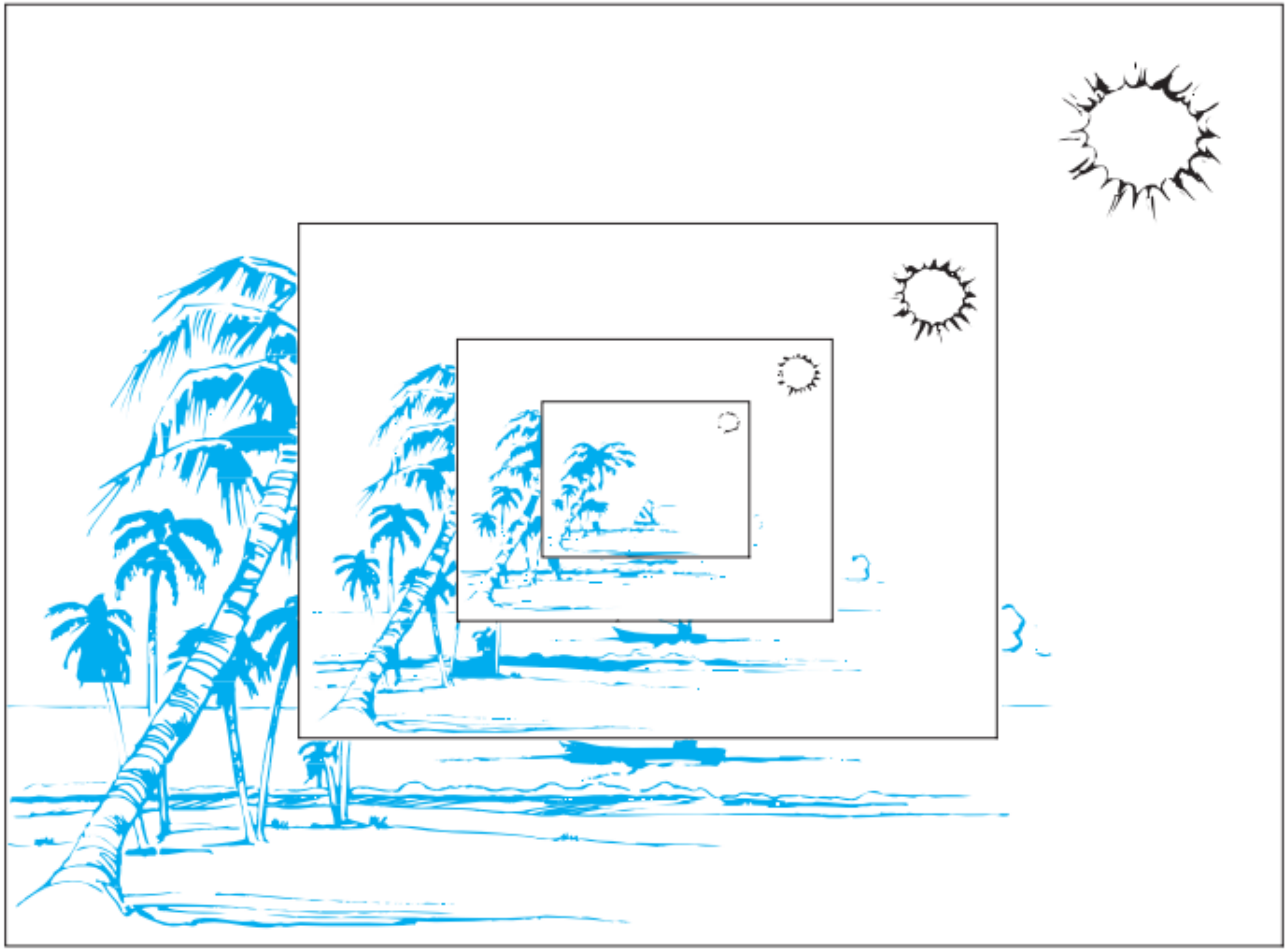


FIGURE 1 **A Recursively Defined Picture.**

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:

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

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

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

Recursively Defined Functions

- Definition 1: Fibonacci numbers

Recursively Defined Functions

Theorem 1: Lamé's theorem: Let a,b be integers, a ≥ 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.

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

x	y	r
25	7	25 mod 7=4
7	4	7 mod 4=3
4	3	4 mod 3=1
3	1	3 mod 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,…}

Step 1: $3 \in S$

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

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

Basis step: $\lambda \in \Sigma^*$, λ 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 strings 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 (.)

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

Recursive step: If $w_1 \in \Sigma^*$ and $w_2 \in \Sigma^*$ and $x \in \Sigma$ then $w_1.(w_2x) = (w_1.w_2)x$

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

2. Find $f(1)$, $f(2)$, $f(3)$, $f(4)$, and $f(5)$ if $f(n)$ is defined recursively by $f(0) = 3$ and for $n = 0, 1, 2, \ldots$

a) $f(n+1) = -2f(n)$.

b) $f(n+1) = 3f(n) + 7$.

c) $f(n+1) = f(n)^2 - 2f(n) - 2$.

d) $f(n+1) = 3^{f(n)/3}$.

4. Find $f(2)$, $f(3)$, $f(4)$, and $f(5)$ if f is defined recursively by $f(0) = f(1) = 1$ and for $n = 1, 2, \ldots$

a) $f(n+1) = f(n) - f(n-1)$.

b) $f(n+1) = f(n)f(n-1)$.

c) $f(n+1) = f(n)^2 + f(n-1)^3$.

d) $f(n+1) = f(n)/f(n-1)$.

8. Give a recursive definition of the sequence $\{a_n\}$, $n = 1, 2, 3, \ldots$ if

a) $a_n = 4n - 2$.

b) $a_n = 1 + (-1)^n$.

c) $a_n = n(n+1)$.

d) $a_n = n^2$.

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
$a^n = a.a.a...a = a.a^{n-1}$, n>0

Recursive Algorithms…

Example: Recursive algorithm for computing gcd(a,b)

a,b: non negative integer, a < b

If a>b then swap a,b
gcd(a,b)=b , a=0
gcd (a,b) = gcd(b mod a, a)

ALGORITHM 3 A Recursive Algorithm for Computing gcd(a, b).

procedure $gcd(a, b$: nonnegative integers with $a < b$)
if $a = 0$ **then return** b
else return $gcd(b$ **mod** $a, a)$
{output is gcd(a, b)}

Recursive Algorithms…

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

a_i, a_{i+1},…, a_j, sub-sequence of a_n.

$1 \le i \le n$, $1 \le j \le n$

$i>j \rightarrow$ location =0
$a_i=x \rightarrow$ location = i
location (i, j, x) = 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 $a_i, a_{i+1}, \ldots, a_{j-1}$, sub-sequence of a_n. $1 \le i \le n$, $1 \le j \le n$

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.

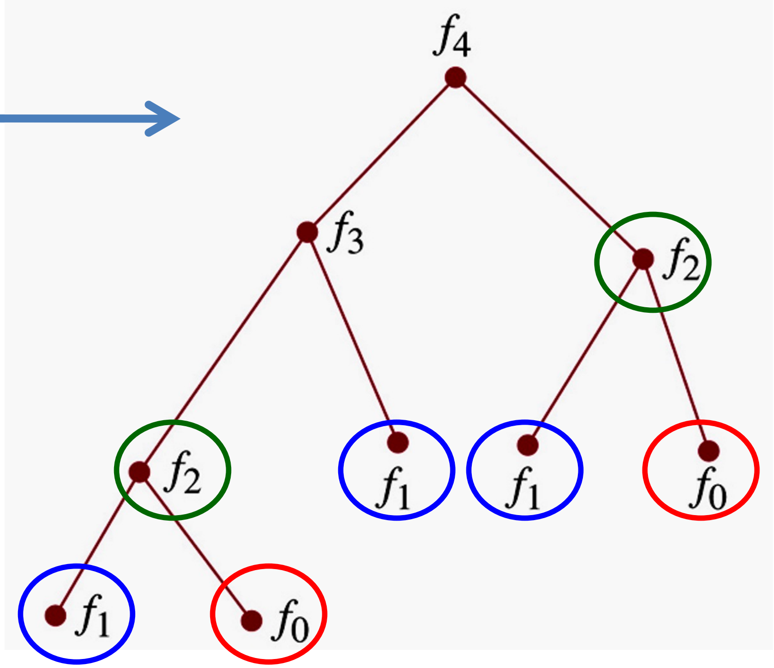
procedure f (n: nonnegative integer)
  if n=0 **then** f(n) :=1
  else f(n) = n.f(n-1)

If n=0, first step of the algorithm tells us f(0)=1 → true
Assuming f(n) is true for all n≥0
f(n)= 1.2.3....(n)
(n+1).f(n)= 1.2.3...n.(n+1) = (n+1)!
f(n+1) = (n+1)!
Conclusion: f(n) is true for all integer n, n≥0

More examples: Page 315

Recursion and Iteration

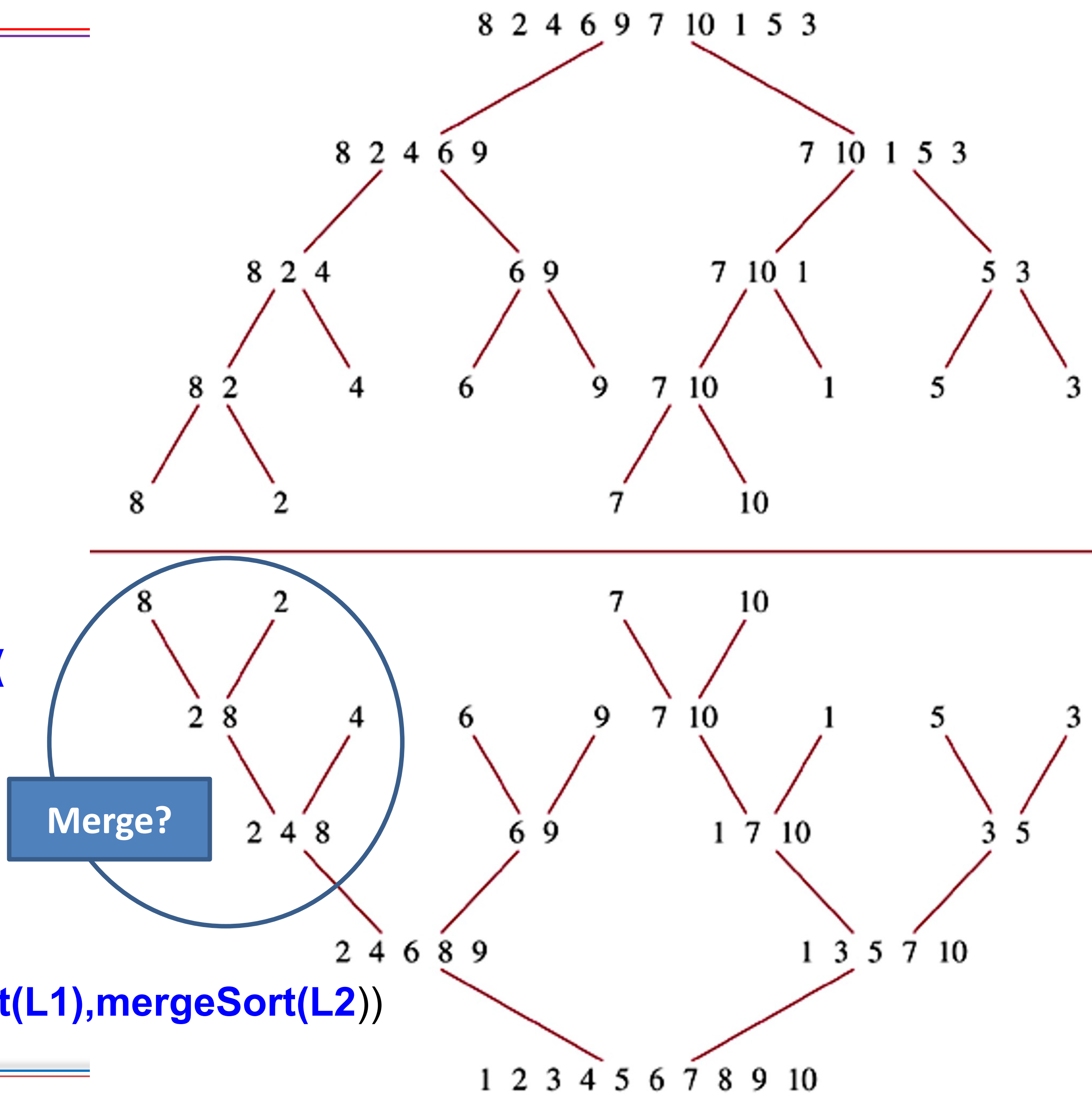
procedure rfibo (n: nonnegative integer)
If n=0 **then** rFibo(0)=0
Else if n=1 **then** rFibo(1)=1
Else rFibo(n) := rFibo(n-2) + rFibo(n-1)



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}

Recursive algorithm uses far more computation than iterative one

Merge Sort



Procedure mergeSort ($L=a_1 \ldots a_n$)
If $n>1$ **then**
Begin
  $m := \lfloor n/2 \rfloor$
  $L1 := a_1 \ldots a_m$
  $L2 := a_{m+1} \ldots a_n$
  L:=merge(mergeSort(L1),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 no 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(nlog n)

ALGORITHM 10 Merging Two Lists.

procedure *merge*(L_1, L_2 : sorted lists)
L := empty list
while L_1 and L_2 are both nonempty
    remove smaller of first elements of L_1 and L_2 from its list; put it at the right end of L
    if this removal makes one list empty **then** remove all elements from the other list and
        append them to L
return L\{L is the merged list with elements in increasing order\}

Two sorted lists with m elements and n elements can be merged into a sorted list using no more than $m + n - 1$ comparisons.

How many comparisons are required to merge these pairs of lists using Algorithm 10?

1, 3, 5, 7, 9; 2, 4, 6, 8, 10

Which of the following procedures are recursive algorithms?

```
(A)Procedure ORD(x: int, n: positive int)
       d:=0
       for i = 1 to n do
              d:=d+x
      print(d)
(B) Procedure ORD(x: int, n: positive int)
       If n =1 then ORD(x,n):=x
       else ORD(x,n):=ORD(x,n-1)+x
(C) Procedure ORD(n: positive int)
 d:=0
       for i = 1 to n do
              d:=d+n
      print(d)
```

Given the function f defined recursively as follows:

$$f(n) = 1 + f(n - 1)\, f(n - 2) \; - 2\, f(n - 1)\;,\; n = 3, 4, \ldots$$

with $f(1) = 1$, $f(2) = -1$. Find $f(5)$.

Find a recursive definition for the set of all positive integers NOT divisible by 3.

(i) $1 \in S$, if $a \in S$ then $a+3 \in S$ and $a-3 \in S$

(ii) $1 \in S$, if $a \in S$ then $a+3 \in S$

(iii) $1, 2 \in S$, if $a \in S$ then $a+3 \in S$ and $a-3 \in S$

(iv) $1, 2 \in S$, if $a \in S$ then $a+3 \in S$

Example 2.

(a) Give a recursive definition for the set of positive integers that are not divisible by 3

(b) Give a recursive definition for the set of integers that are not divisible by 3

S is the set of all positive even integers

$2 \in S$,
If $x \in S$ then $x + 2 \in S$

S is the set of all positive integers that are divisible by 5

$2 \in S$,
If $x \in S$ then $x + 5 \in S$

$$S = \{0,5,10\}$$

S is the set of all integers that are not divisible by 5

$1 \in S; 2 \in S; 3 \in S; 4 \in S,$
If $x \in S$ then $x + 5 \in S$ and $x - 5 \in S$

$$S = \{0,3, \ldots\}$$

Which of the following procedures are recursive algorithms?

(i) procedure tt1(x: int, n: positive int)

```
d:=0
for i:=1 to n do
  d:=d+x
print(d)
```

(ii) procedure tt2(x: int, n: positive int)

```
If n=1 then tt2(x,n):=x
else tt2(x,n):=tt2(x,n-1)+x
```

(iii) procedure tt3(n: positive int)

```
s:=0
For i:=1 to n do
  s:=s+n
Print(s)
```

Summary

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

Thanks