

Chapter 5

Induction and Recursion

Objectives

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

MATHEMATICAL INDUCTION

Proofs

Basic proof methods: Direct, Indirect, Contradiction, By cases, Equivalences

Proof of quantified statements:

- **There exists x with some property $P(x)$**
 - It is sufficient to find one element for which the property holds.
- **For all x some property $P(x)$ holds**
 - Proofs of ‘For all x some property $P(x)$ holds’ must cover all x and can be harder.

Mathematical induction is a technique that can be applied to prove the universal statements for sets of positive integers or their associated sequences.

Mathematical induction

Used to prove statements of the form $\forall x P(x)$ where $x \in \mathbb{Z}^+$

PRINCIPLE OF MATHEMATICAL INDUCTION – To prove that $P(n)$ is true for all positive integers n , where $P(n)$ is a propositional function, we complete **two steps**:

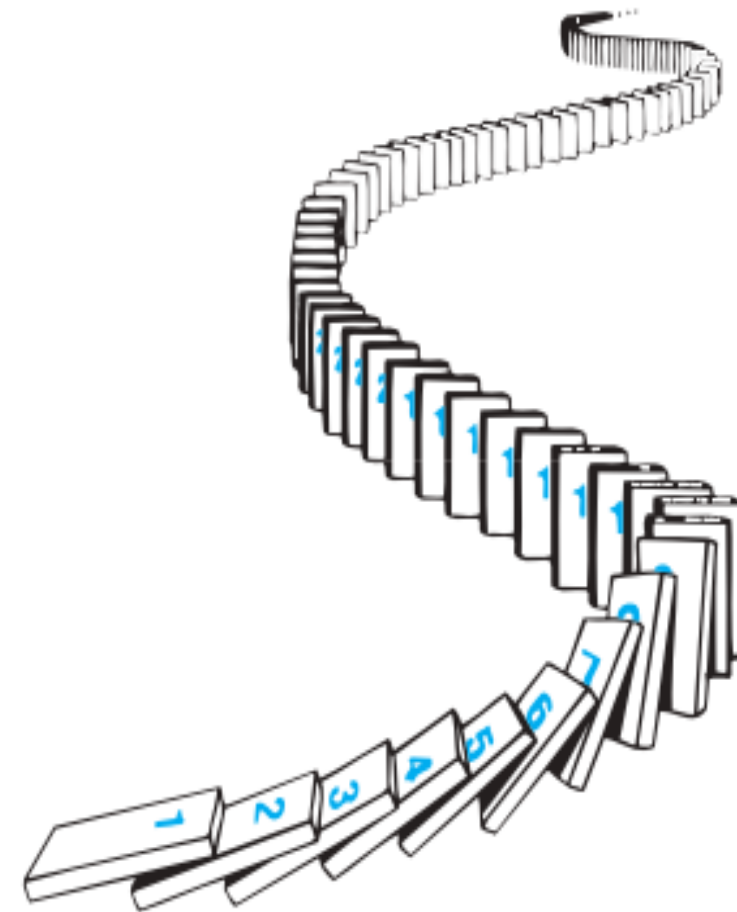
1. Basic step: Prove that $P(1)$ is true.

2. Inductive step: We show that the conditional statement $P(k) \rightarrow P(k + 1)$ is true for all positive integers k .

2.1. (**Inductive hypothesis**) Assume that $P(k)$ is true for some positive integer k .

2.2. Prove that $P(k + 1)$ is true.

Then we can conclude that $P(n)$ is true for all positive integer n .



Mathematical induction

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

Proof. Let $P(n): 1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2}$

Basis step: Show that $P(1)$ is true.

We have $1 = 1(1+1)/2$ (true)

Inductive step: Show if $P(k)$ is true then $P(k+1)$ is true for all k

Suppose $P(k)$ is true, that is $1 + 2 + \dots + k = \frac{k(k+1)}{2}$. Prove that $P(k+1): 1 + 2 + \dots +$

$$k + (k + 1) = \frac{(k+1)((k+1)+1)}{2}$$

$$1 + 2 + 3 + \dots + k + (k + 1) = \frac{k(k+1)}{2} + (k + 1)$$

$$= \frac{[k(k+1) + 2(k+1)]}{2} = \frac{(k+1)((k+1)+1)}{2}$$

Mathematical induction

Example. Prove $1 + 3 + 5 + \dots + (2n - 1) = n^2$ for $\forall n \in \mathbb{Z}^+$.

Proof

- What is $P(n)$? $P(n): 1 + 3 + 5 + 7 + \dots + (2n - 1) = n^2$

Basis Step Show $P(1)$ is true

- Trivial: $1 = 1^2$

Inductive Step Show if $P(n)$ is true then $P(n+1)$ is true for all n .

- Suppose $P(n)$ is true, that is $1 + 3 + 5 + 7 + \dots + (2n - 1) = n^2$
- Show $P(n+1): 1 + 3 + 5 + 7 + \dots + (2n - 1) + (2n + 1) = (n+1)^2$ follows:
- $$\underbrace{1 + 3 + 5 + 7 + \dots + (2n - 1)}_{n^2} + (2n + 1) = (n+1)^2$$

Mathematical induction

Example. Prove that $n < 2^n$ for $\forall n \in \mathbb{Z}^+$

Proof. $P(n): n < 2^n$

Basic step: $1 < 2^1$

Inductive step:

+ Suppose $P(k): k < 2^k$ is true

+ Show that $P(k + 1): k + 1 < 2^{k+1}$ is true

$$\begin{aligned} k + 1 &< 2^k + 1 \\ &< 2^k + 2^k \\ &= 2^k(1 + 1) \\ &= 2^{k+1} \end{aligned}$$

Therefore, $n < 2^n$ for $\forall n \in \mathbb{Z}^+$

Mathematical induction

Example: Prove $n^3 - n$ is divisible by 3 for all positive integers.

- $P(n)$: $n^3 - n$ is divisible by 3

Basis Step: $P(1)$: $1^3 - 1 = 0$ is divisible by 3 (obvious)

Inductive Step: If $P(n)$ is true then $P(n+1)$ is true for each positive integer.

- Suppose $P(n)$: $n^3 - n$ is divisible by 3 is true.
- Show $P(n+1)$: $(n+1)^3 - (n+1)$ is divisible by 3.

$$\begin{aligned}
 (n+1)^3 - (n+1) &= n^3 + 3n^2 + 3n + 1 - n - 1 \\
 &= (n^3 - n) + 3n^2 + 3n \\
 &= \underbrace{(n^3 - n)}_{\text{divisible by 3}} + \underbrace{3(n^2 + n)}_{\text{divisible by 3}}
 \end{aligned}$$

divisible by 3

divisible by 3

Exercises

1. Let $P(n)$ be the statement that $1^2 + 2^2 + \cdots + n^2 = n(n+1)(2n+1)/6$ for the positive integer n . Show that $P(n)$ is true for $\forall n \in \mathbb{Z}^+$.
2. Prove that 6 divides $n^3 - n$ whenever n is a nonnegative integer.
3. Let n be a positive integer. Prove that every checkerboard of size $2^n \times 2^n$ with one square removed can be tiled by triominoes.

STRONG INDUCTION AND WELL – ORDERING

STRONG INDUCTION

Regular induction

1. **Basic step:** Prove that $P(1)$ is true
 2. **Inductive step:**
 - 2.1. Suppose that $P(k)$ is true (induction hypothesis)
 - 2.2. Prove that $P(k + 1)$ is true
- Conclusion: $P(n)$ is true for all positive integers n

Strong induction

1. **Basic step:** Prove that $P(1)$ is true
 2. **Inductive step:**
 - 2.1. **Suppose that $P(j)$ is true for $j = 1, 2, \dots, k$**
 - 2.2. Prove that $P(k + 1)$ is true
- Conclusion: $P(n)$ is true for all positive integers n

STRONG INDUCTION

Strong induction

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

1. **Basic step:** Prove that $P(1)$ is true
2. **Inductive step:**
 - 2.1. Inductive hypothesis: $P(j)$ is true for $j = 1, 2, \dots, k$
 - 2.2. Prove that $P(k + 1)$ is true

Generalizing strong induction

$P(n)$ is true for $\forall n \geq b$ (b : fixed integer)

1. **Basic step:** Prove that $P(b), P(b + 1), \dots, P(b + j)$ are true (j : a fixed positive integer)
2. **Inductive step:**
 - 2.1. Inductive hypothesis: $P(j)$ is true for $j = b, b + 1, \dots, k$
 - 2.2. Prove that $P(k + 1)$ is true

STRONG INDUCTION

Example. Show that a positive integer greater than 1 can be written as a product of primes.

Assume $P(n)$: an integer n can be written as a product of primes.

Proof.

1. **Basis step:** $P(2)$ is true
2. **Inductive step:** Assume true for $P(2), P(3), \dots, P(k)$. Show that $P(k+1)$ is true as well.
 - 2.1. $k + 1$ is a prime then we immediately see that $P(k + 1)$ is true
 - 2.2. $k + 1$ is a composite then it can be written as a product of two integers $(n + 1) = a * b$ such that $1 < a, b < n+1$

From the assumption $P(a)$ and $P(b)$ holds.

Thus, $n+1$ can be written as a product of primes.

The Well-Ordering Property

THE WELL-ORDERING PROPERTY. Every nonempty set of nonnegative integers has a least element.

Correctness of the mathematical induction

Suppose $P(1)$ is true and $P(n) \rightarrow P(n+1)$ is true for all positive integers n . Want to show $\forall x P(x)$.

Assume there is at least one n such that $P(n)$ is false. Let S be the set of nonnegative integers where $P(n)$ is false. Thus $S \neq \emptyset$.

Well-Ordering Property: Every nonempty set of nonnegative integers has a least element.

By the Well-Ordering Property, S has a least member, say k . $k > 1$, since $P(1)$ is true. This implies $k - 1 > 0$ and $P(k-1)$ is true (since k is the smallest integer where $P(k)$ is false).

Now: $P(k-1) \rightarrow P(k)$ is true
 thus, $P(k)$ must be true (a contradiction).

- Therefore $\forall x P(x)$.

$\forall n, P(n)$ đúng

+ $P(1)$ đúng

+ $P(k) \rightarrow P(k+1)$ đúng, $\forall k > 0$

g/s ngược lại, tức là $\exists n_0 \in \mathbb{N}, P(n_0)$ sai

$$S = \{n \in \mathbb{N} \mid P(n) \text{ sai}\} \subseteq \mathbb{N}$$

$$\Rightarrow S \neq \emptyset \text{ vì } n_0 \in S$$

$\Rightarrow S$ có ptử' nhỏ nhất là $m \in S$ & $m > 1$

$\Rightarrow m-1 \notin S$

$\Rightarrow P(m-1)$ đúng

$\Rightarrow P(m)$ đúng, là điều vớ lý vì $P(m)$ sau

$P(n)$ đúng $\forall n \geq n_0 \geq 1$

+ $P(n_0)$ đúng

{ + g/s $P(n)$ đúng với $n = k > n_0$

thì là $P(k)$ đúng

+ Ta sẽ CM $P(n)$ đúng với $n = k+1$

thì là ta CM $P(k+1)$ đúng

STRONG INDUCTION

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

Proof.

$P(n)$: postage of n cents can be formed using 4-cent and 5-cent stamps.

1. Basic step: $P(12) = 3 \times 4$, $P(13) = 2 \times 4 + 5$, $P(14) = 1 \times 4 + 2 \times 5$, $P(15) = 3 \times 5$

2. Inductive step:

2.1. Inductive hypothesis: $P(j)$ is true for $12 \leq j \leq k$, where k is an integer with $k \geq 15$

2.2. Show that $P(k + 1)$ is true.

$k + 1 = (k - 3) + 4$, $k \geq 15$. $k \geq 15 \rightarrow k - 3 \geq 12$. Using the inductive hypothesis, we can say that $P(k - 3)$ is true, that is we can form postage of $k - 3$ cents using just 4-cent and 5-cent stamps. That is, we have shown that if the inductive hypothesis is true, then $P(k + 1)$ is also true. This completes the inductive step.

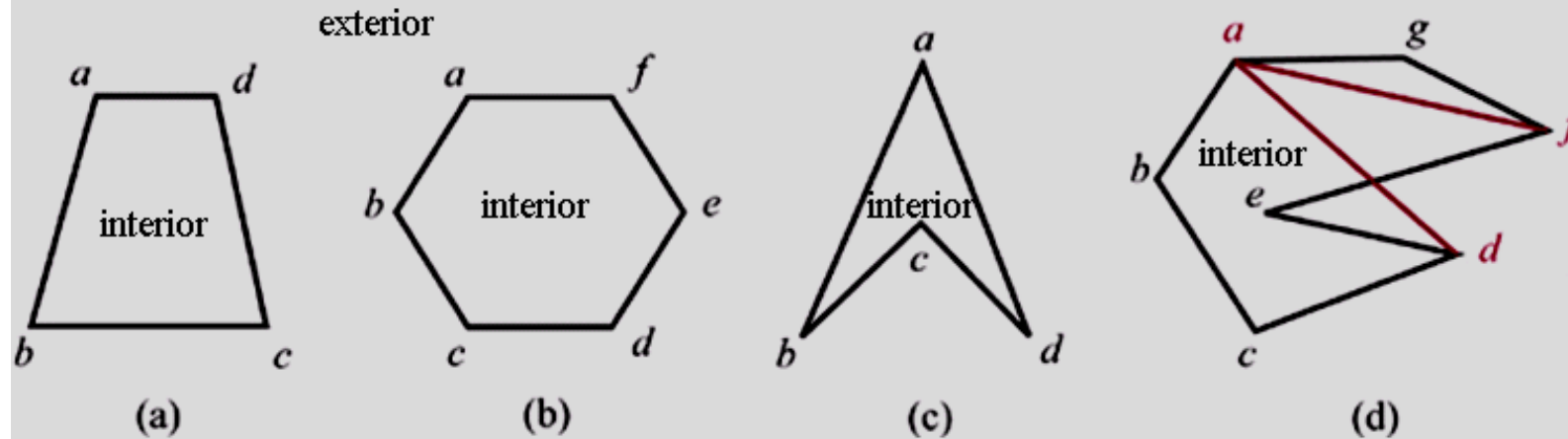
Using Strong Induction in Computational Geometry

A **polygon** is a closed geometric figure consisting of a sequence of line segments s_1, s_2, \dots, s_n , called **sides**. Each pair of consecutive sides, s_i and s_{i+1} , $i = 1, 2, \dots, n - 1$, as well as the last side s_n and the first side s_1 , of the polygon meet at a common endpoint, called a **vertex**. A polygon is called **simple** if no two nonconsecutive sides intersect. Every simple polygon divides the plane into two regions: its **interior**, consisting of the points inside the curve, and its **exterior**, consisting of the points outside the curve.

A polygon is called **convex** if every line segment connecting, two points in the interior of the polygon lies entirely inside the polygon. (A polygon that is not convex is said to be **nonconvex**.)

Using Strong Induction in Computational Geometry

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

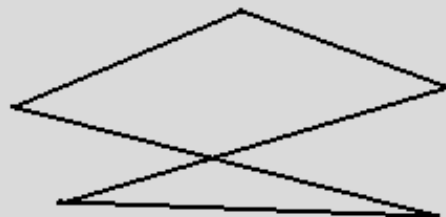


Convex Polygons

Nonconvex Polygons

af is an interior diagonal
ad is not an interior diagonal

Simple Polygons

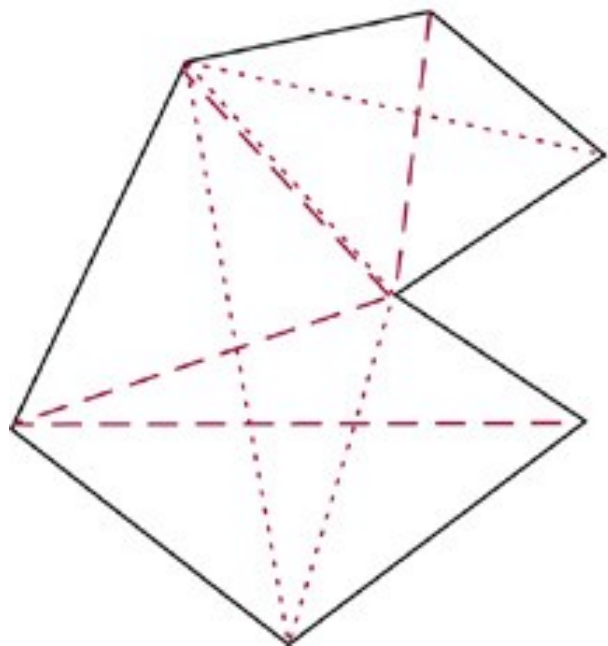


Non-simple polygon

Using Strong Induction in Computational Geometry

Theorem. 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.

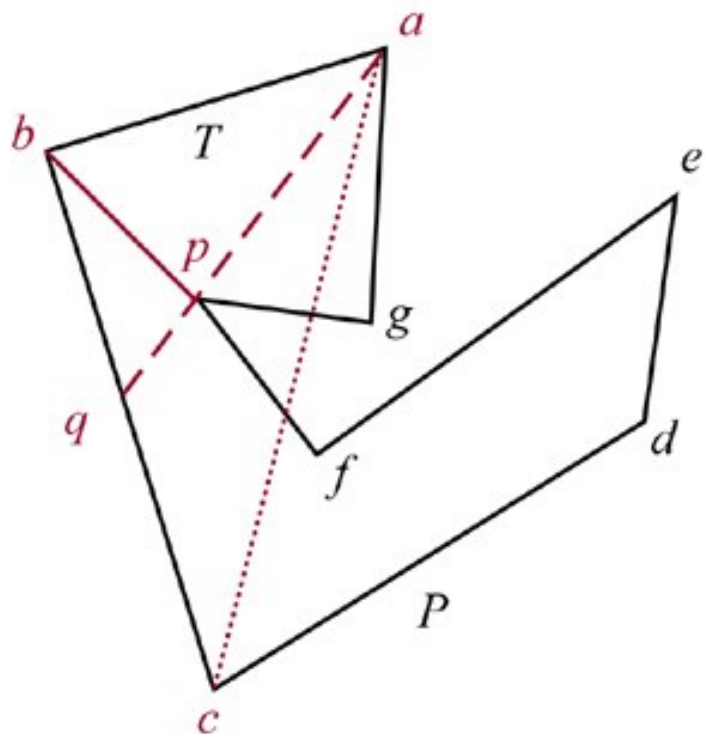


Two different triangulations of a simple polygon with seven sides into five triangles, shown with dotted lines and with dashed lines, respectively

Using Strong Induction in Computational Geometry

Lemma. 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

Exercises.

1. Let $P(n)$ be the statement that a postage of n cents can be formed using just 3-cent stamps and 5-cent stamps. The parts of this exercise outline a strong induction proof that $P(n)$ is true for all integers $n \geq 8$.
 - a) Show that the statements $P(8)$, $P(9)$, and $P(10)$ are true, completing the basis step of a proof by strong induction that $P(n)$ is true for all integers $n \geq 8$.
 - b) What is the inductive hypothesis of a proof by strong induction that $P(n)$ is true for all integers $n \geq 8$?
 - c) What do you need to prove in the inductive step of a proof by strong induction that $P(n)$ is true for all integers $n \geq 8$?
 - d) Complete the inductive step for $k \geq 10$.
 - e) Explain why these steps show that $P(n)$ is true whenever $n \geq 8$.

Examples.

2. Let $P(n)$ be the statement that a postage of n cents can be formed using just 4-cent stamps and 7-cent stamps. The parts of this exercise outline a strong induction proof that $P(n)$ is true for all integers $n \geq 18$.
- a) Show that the statements $P(18)$, $P(19)$, $P(20)$, and $P(21)$ are true, completing the basis step of a proof by strong induction that $P(n)$ is true for all integers $n \geq 18$.
 - b) What is the inductive hypothesis of a proof by strong induction that $P(n)$ is true for all integers $n \geq 18$?
 - c) What do you need to prove in the inductive step of a proof by strong induction that $P(n)$ is true for all integers $n \geq 18$?
 - d) Complete the inductive step for $k \geq 21$.
 - e) Explain why these steps show that $P(n)$ is true whenever $n \geq 18$.

Exercises

3. Which amounts of money can be formed using just two-dollar bills and five-dollar bills? Prove your answer using strong induction
4. Suppose that a store offers gift certificates in denominations of 25 dollars and 40 dollars. Determine the possible total amounts you can form using these gift certificates. Prove your answer using strong induction.

Exercises

3. Which amounts of money can be formed using just two-dollar bills and five-dollar bills? Prove your answer using strong induction
4. Suppose that a store offers gift certificates in denominations of 25 dollars and 40 dollars. Determine the possible total amounts you can form using these gift certificates. Prove your answer using strong induction.

RECURSIVE DEFINITIONS AND STRUCTURAL INDUCTION

Introduction

Sometimes it is possible to define an object (function, sequence, algorithm, structure) in terms of itself. This process is called **recursion**.

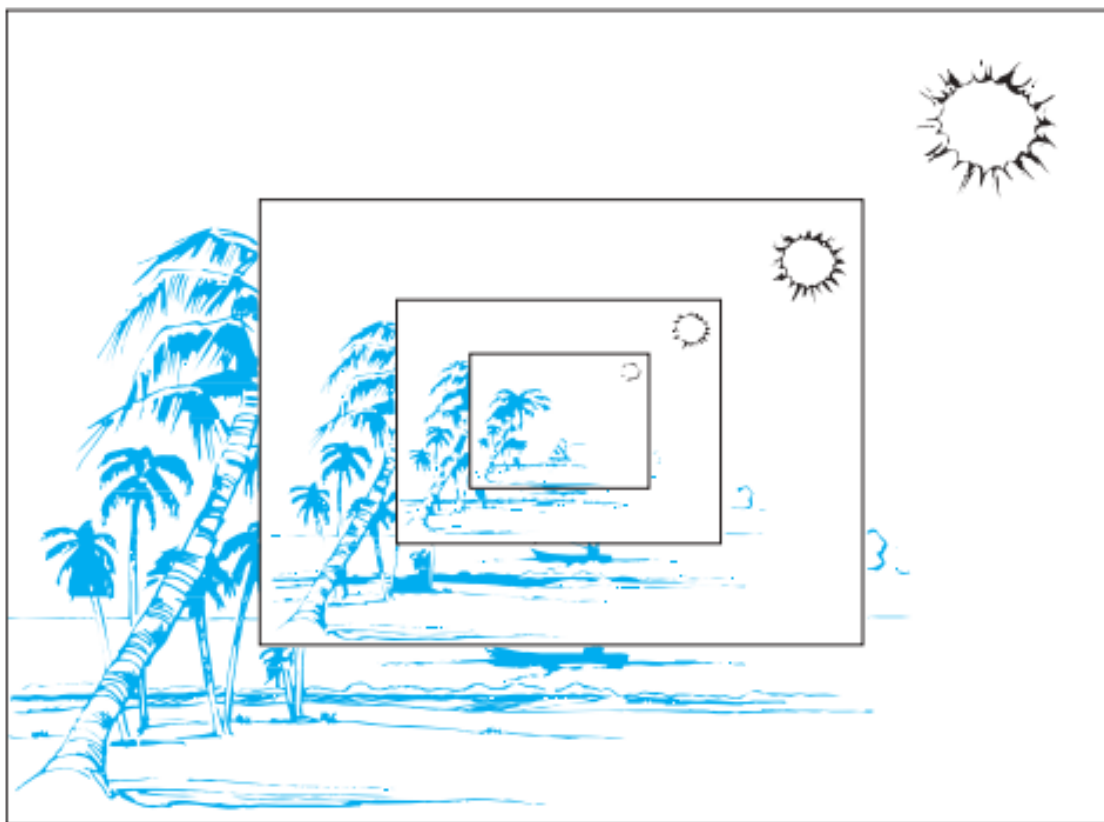


FIGURE 1 A recursively defined picture.

Introduction

Examples.

- Recursive definition of an arithmetic sequence:

$$a_n = a + nd$$

$$a_0 = a, a_n = a_{n-1} + d$$

- Recursive definition of a geometric sequence:

$$x_n = ar^n$$

$$x_0 = a, x_n = rx_{n-1}$$

Recursively defined functions

We use two steps to define a function with the set of nonnegative integers as its domain:

1. **BASIS STEP:** Specify the value of the function at zero.
2. **RECURSIVE STEP:** Give a rule for finding its value at an integer from its values at smaller integers.

Such a definition is called a **recursive definition**.

To define a function on the set of nonnegative integers

1. Basic step: Specify the value of the function at 0.
2. Recursive step: Give a rule for finding the function's value at $n+1$ in terms of the function's value at integers $i \leq n$.

Recursively defined functions

Example. Suppose that f is defined recursively by

- $f(0) = 3$
- $f(n + 1) = 2f(n) + 3$

Find $f(1)$, $f(2)$, $f(3)$ and $f(4)$.

Solution. From the recursive definition it follows that

$$f(1) = 2f(0) + 3 = 2 \cdot 3 + 3 = 9,$$

$$f(3) = 2f(2) + 3 = 2 \cdot 21 + 3 = 45,$$

$$f(2) = 2f(1) + 3 = 2 \cdot 9 + 3 = 21,$$

$$f(4) = 2f(3) + 3 = 2 \cdot 45 + 3 = 93,$$

Recursively defined functions

Example. Give a recursive definition of a^n , where a is a nonzero real number and n is a nonnegative integer.

Solution.

Basic step: $a^0 = 1$

Recursive step: $a^{n+1} = a \cdot a^n$, for $n = 0, 1, 2, \dots$

Example. Give a recursive definition of $S(k) = \sum_{k=0}^n a_k$.

Solution.

Basic step: $S(0) = \sum_{k=0}^0 a_0$

Recursive step: $S(k+1) = \sum_{k=0}^{n+1} a_k = (\sum_{k=0}^n a_k) + a_{k+1} = S(k) + a_{k+1}$

Recursively defined functions

Some important functions or sequences in mathematics are defined recursively.

□ **The Fibonacci numbers**, f_0, f_1, f_2, \dots , are defined by

- $f_0 = 0, f_1 = 1$
- $f_n = f_{n-1} + f_{n-2}$ for $n = 2, 3, 4, \dots$

Factorials

$$0! = 1$$

$$n! = n \cdot (n - 1)! \text{ for } n \geq 1$$

Lame's theorem

Theorem. Let $a, b \in \mathbb{N}$, $a \geq b$ and n be the number of steps in the Euclidean algorithm needs to compute $\gcd(a, b)$. Then $n \leq 5k$, where $k = \lfloor \log_{10} b \rfloor + 1$ is the number of decimal digits in b .

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  $\neq$  0
begin
  r := x mod y
  x:=y
  y:= r
end {gcd(a,b) is x}

```

Recursively Defined Sets

Definition. The set S is defined recursively by

- BASIS STEP: specifies one or more initial members of S .
- RECURSIVE STEP: specifies the rule(s) for constructing new elements of from the existing elements.

Exclusion (or closure) rule states that every element in follows from the basis step and a finite number of recursive steps. (The exclusion rule is assumed, so no need to state it explicitly.)

Recursively Defined Sets

Examples.

- Natural numbers

Basis: $0 \in S$

Recursive: if $n \in S$, then $n + 1 \in S$

- Odd natural numbers

Basis: $1 \in S$

Recursive: if $x \in S$, then $x + 2 \in S$

Recursively Defined Sets

Example. Let $3 \in S$ and let $x + y \in S$ if $x, y \in S$. What is S ?

Solution.

- Basis step: $3 \in S$
- $6 \in S$ (first application of recursive step)
- $9(= 3 + 6) \in S$ and $12(= 6 + 6) \in S$ (second application of recursive step)
- ...

Therefore, $S = \{3, 6, 9, 12, 15, 18, 21, 24, \dots\}$

Sets of strings

Definition. An alphabet Σ is any finite set of characters. The set Σ^* of strings over the alphabet Σ is defined recursively by

BASIS STEP: $\lambda \in \Sigma^*$ (where λ is the empty string containing no symbols).

RECURSIVE STEP: If $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma^*$.

Example. $\Sigma = \{0, 1\}$

- Basis step: $\lambda \in \Sigma^*$
- 0 and 1 are in Σ^* (first application of recursive step)
- 00, 01, 10 and 11 are in Σ^* (2nd application of recursive step)
- ...

Therefore, $\Sigma^* = \{\lambda, 0, 1, 00, 01, 10, 11, 000, 001, 010, 100, \dots\}$.

Example. Show that if $\Sigma = \{a, b\}$ then aab is in Σ^* .

Concatenation of two strings

Definition. Two strings can be combined via the operation of concatenation. Let Σ be a set of symbols and Σ^* the set of strings formed from symbols in Σ . We can define the concatenation of two strings, denoted by \cdot , recursively as follows.

BASIS STEP: If $w \in \Sigma^*$, then $w \cdot \lambda = w$ where λ : the empty string.

RECURSIVE STEP: If $w_1 \in \Sigma^* \wedge w_2 \in \Sigma^* \wedge x \in \Sigma^*$, then $w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x$.

Example. If $w_1 = abra$ and $w_2 = cadabra$, the concatenation $w_1 w_2 = abracadabra$

Length of a string

Let Σ be a set of symbols and Σ^* the set of strings formed from symbols in Σ .

The **length of a string** can be recursively defined by

- $l(\lambda) = 0$;
- $l(wx) = l(w) + 1$ if $w \in \Sigma^*$ and $x \in \Sigma$.

Rooted trees

A tree is a special type of a graph; a graph is made up of vertices and edges connecting some pairs of vertices. (we will study trees and graphs in Chapter 10, 11.)

Definition. The set of rooted trees, where a rooted tree consists of a set of vertices containing a distinguished vertex called the root, and edges connecting these vertices, can be defined recursively by these steps:

BASIS STEP: A single vertex r is a rooted tree.

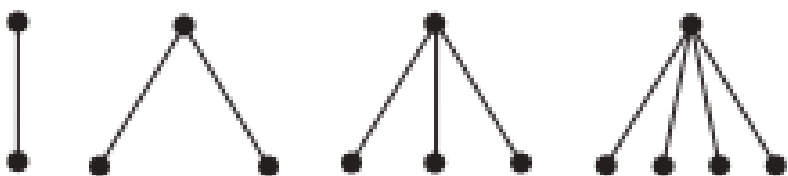
RECURSIVE STEP: Suppose that T_1, T_2, \dots, T_n are disjoint rooted trees with roots r_1, r_2, \dots, r_n , respectively. Then the graph formed by starting with a root r , which is not in any of the rooted trees T_1, T_2, \dots, T_n , and adding an edge from r to each of the vertices r_1, r_2, \dots, r_n , is also a rooted tree.

Building up rooted trees

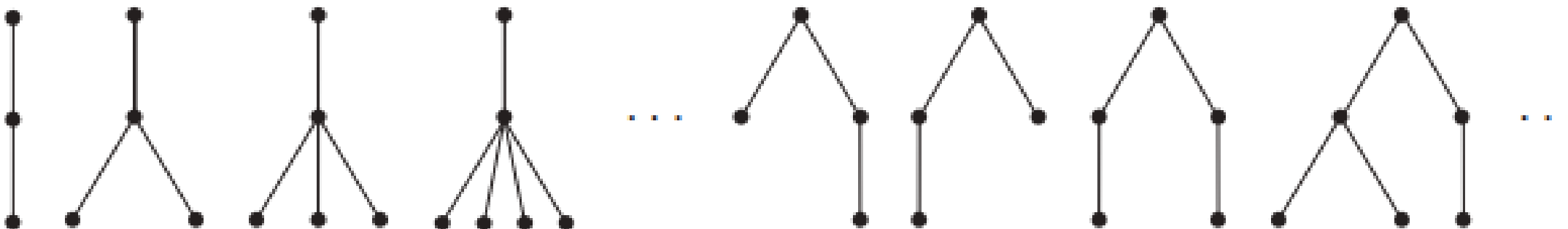
Basis step



Step 1



Step 2



Extended binary trees

Definition. The set of extended binary trees can be defined recursively by these steps:

BASIS STEP: The empty set is an extended binary tree.


RECURSIVE STEP: If T_1 and T_2 are disjoint extended binary trees, there is an extended binary tree, denoted by $T_1 \cdot T_2$, consisting of a root r together with edges connecting the root to each of the roots of the left subtree T_1 and the right subtree T_2 when these trees are nonempty

Building up extended binary trees

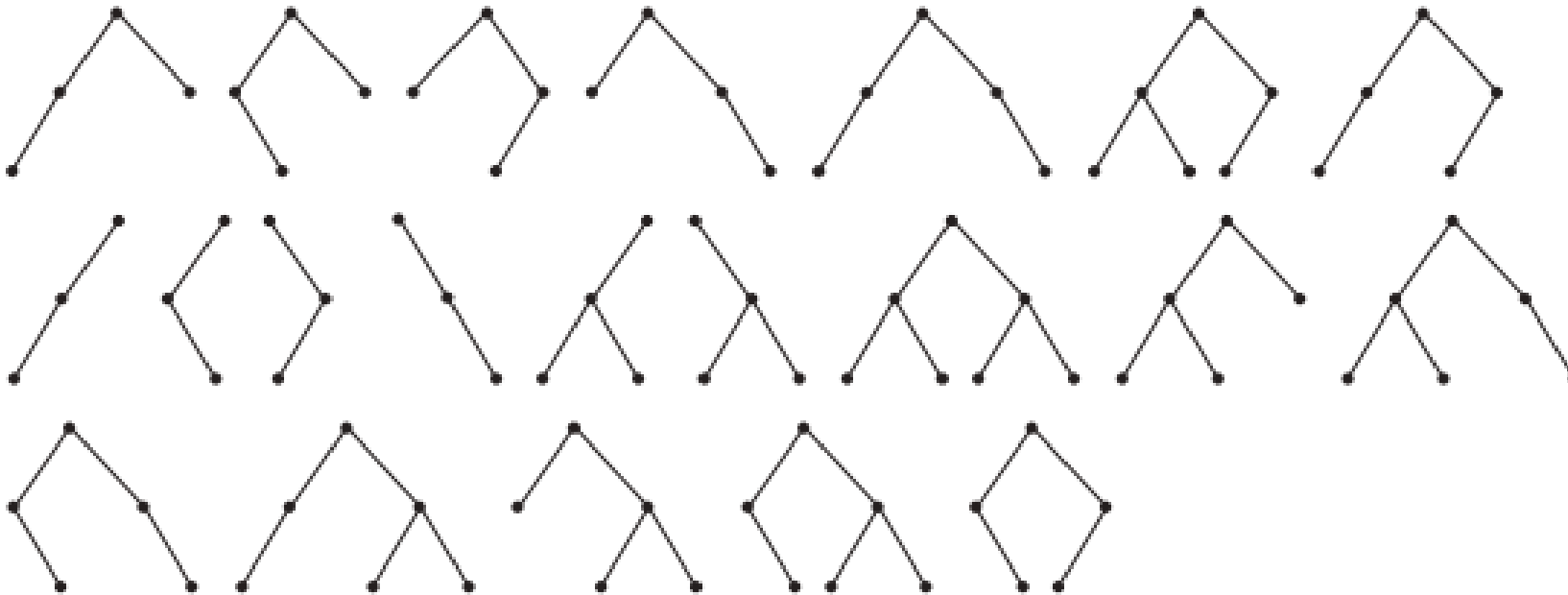
Basis step \emptyset

Step 1 •

Step 2



Step 3



Full binary trees

Definition. The set of full binary trees can be defined recursively by these steps:

BASIS STEP: There is a full binary tree consisting only of a single vertex r .

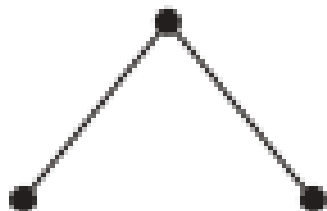
RECURSIVE STEP: If T_1 and T_2 are disjoint full binary trees, there is a full binary tree, denoted by $T_1 \cdot T_2$, consisting of a root r together with edges connecting the root to each of the roots of the left subtree T_1 and the right subtree T_2 .

Building up full binary trees

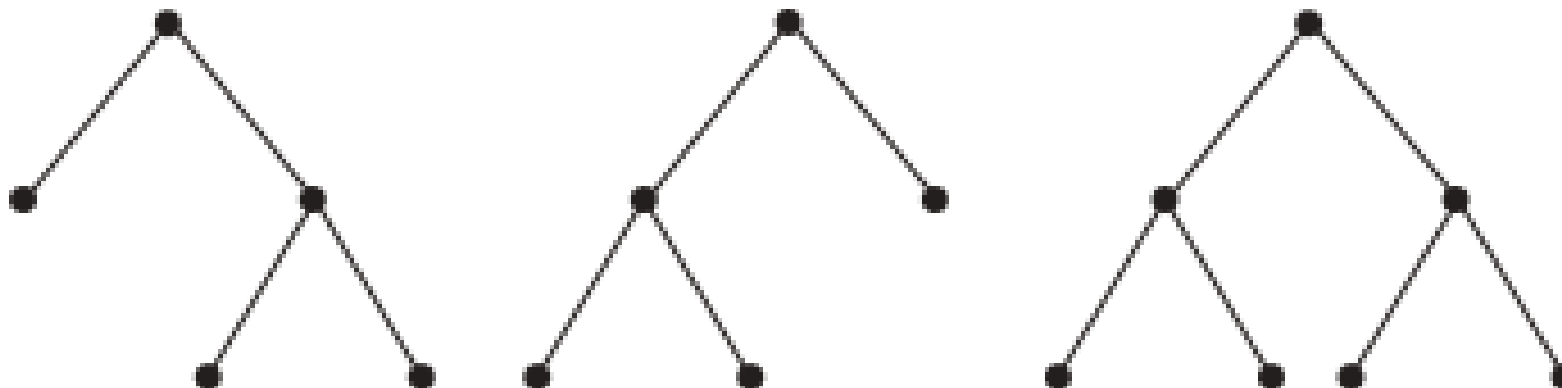
Basis step



Step 1



Step 2



Structural induction

Definition. To prove a property of the elements of a recursively defined set, we use structural induction. A proof by structural induction consists of two parts:

BASIS STEP: Show that the result holds for all elements specified in the basis step of the recursive definition to be in the set.

RECURSIVE STEP: Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the result holds for these new elements.


The validity of structural induction follows from the principle of mathematical induction for the nonnegative integers.

Structural induction

Example. Show that the set S defined by $3 \in S$ and if $x \in S$ and $y \in S$, then $x + y \in S$, is the set of all positive integers that are multiples of 3.

Solution: Let A be the set of all positive integers divisible by 3. To prove that $A = S$, we must show that A is a subset of S and that S is a subset of A . To prove that A is a subset of S , we must show that every positive integer divisible by 3 is in S . We will use mathematical induction to prove this.

Let $P(n)$ be the statement that $3n$ belongs to S . The basis step holds because by the first part of the recursive definition of S , $3 \cdot 1 = 3$ is in S . To establish the inductive step, assume that $P(k)$ is true, namely, that $3k$ is in S . Because $3k$ is in S and because 3 is in S , it follows from the second part of the recursive definition of S that $3k + 3 = 3(k + 1)$ is also in S .

To prove that S is a subset of A , we use the recursive definition of S . First, the basis step of the definition specifies that 3 is in S . Because $3 = 3 \cdot 1$, all elements specified to be in S in this step are divisible by 3 and are therefore in A . To finish the proof, we must show that all integers in S generated using the second part of the recursive definition are in A . This consists of showing that $x + y$ is in A whenever x and y are elements of S also assumed to be in A . Now if x and y are both in A , it follows that $3 \mid x$ and $3 \mid y$. By part (i) of Theorem 1 of Section 4.1, it follows that $3 \mid (x + y)$ completing the proof. 

Full binary trees

Definition. We define the height $h(T)$ of a full binary tree T recursively.

BASIS STEP: The height of the full binary tree T consisting of only a root r is $h(T) = 0$.

RECURSIVE STEP: If T_1 and T_2 are full binary trees, then the full binary tree $T = T_1 \cdot T_2$ has height $h(T) = 1 + \max(h(T_1), h(T_2))$.

- If we let $n(T)$ denote the number of vertices in a full binary tree, we observe that $n(T)$ satisfies the following recursive formula:

BASIS STEP: The number of vertices $n(T)$ of the full binary tree T consisting of only a root r is $n(T) = 1$.

RECURSIVE STEP: If T_1 and T_2 are full binary trees, then the number of vertices of the full binary tree $T = T_1 \cdot T_2$ is $n(T) = 1 + n(T_1) + n(T_2)$.

Full binary trees

Theorem. If T is a full binary tree T , then $n(T) \leq 2^{h(T)+1} - 1$.

Proof.

- **BASIS STEP:** The result holds for a full binary tree consisting only of a root, $n(T) = 1$ and $h(T) = 0$. Hence, $n(T) = 1 \leq 2^{0+1} - 1 = 1$.
- **RECURSIVE STEP:** Assume $n(T_1) \leq 2^{h(T_1)+1} - 1$ and also

$n(T_2) \leq 2^{h(T_2)+1} - 1$ whenever T_1 and T_2 are full binary trees.

$$\begin{aligned}
 n(T) &= 1 + n(T_1) + n(T_2) && \text{(by recursive formula of } n(T)) \\
 &\leq 1 + \left(2^{h(T_1)+1} - 1\right) + \left(2^{h(T_2)+1} - 1\right) && \text{(by inductive hypothesis)} \\
 &\leq 2 \cdot \max\left(2^{h(T_1)+1}, 2^{h(T_2)+1}\right) - 1 \\
 &= 2 \cdot 2^{\max(h(T_1), h(T_2)+1)} - 1 && \left(\max(2^x, 2^y) = 2^{\max(x, y)}\right) \\
 &= 2 \cdot 2^{h(T)} - 1 && \text{(by recursive definition of } h(T)) \\
 &= 2^{h(T)+1} - 1
 \end{aligned}$$

Exercises

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

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

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

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

d) $f(n+1) = f(n)^2 + f(n) + 1$

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

a) $f(n+1) = f(n) + 3f(n-1)$.

c) $f(n+1) = 3f(n)^2 - 4f(n-1)^2$.

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

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

Exercises

3. Determine whether each of these proposed definitions is a valid recursive definition of a function f from the set of nonnegative integers to the set of integers. If f is well defined, find a formula for $f(n)$ when n is a nonnegative integer and prove that your formula is valid.

a) $f(0) = 0$, $f(n) = 2f(n - 2)$ for $n \geq 1$

b) $f(0) = 1$, $f(n) = f(n - 1) - 1$ for $n \geq 1$

c) $f(0) = 2$, $f(1) = 3$, $f(n) = f(n - 1) - 1$ for $n \geq 2$

d) $f(0) = 1$, $f(1) = 2$, $f(n) = 2f(n - 2)$ for $n \geq 2$

e) $f(0) = 1$, $f(n) = 3f(n - 1)$ if n is odd and $n \geq 1$ and $f(n) = 9f(n - 2)$ if n is even and $n \geq 2$

Exercises.

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

a) $a_n = 6n$. b) $a_n = 2n + 1$. c) $a_n = 10^n$. d) $a_n = 5$

5. Let F be the function such that $F(n)$ is the sum of the first n positive integers. Give a recursive definition of $F(n)$.

6. Give a recursive definition of:

a) the set of positive integers that are multiples of 5.

b) the set of even positive integers.

c) the set of positive integer powers of 4.

7. Find a recursive definition for each sequence:

a) 2, 6, 18, 54, 162, ...

b) 20, 17, 14, 11, 8, ...

c) 1, 3, 6, 10, 15, ...

Exercises.

8. Give a recursive definition of the following sets:

a) $\{0.1, 0.01, 0.001, 0.0001, \dots\}$

b) $\{\dots, -6, -4, -2, 0, 2, 4, 6, \dots\}$

c) $\{0, 101, 11011, \dots\}$

9. The reversal of a string is the string consisting of the symbols of the string in reverse order. The reversal of the string w is denoted by w^R . Find the reversal of the following bit strings.

a) 0101 b) 1 1011 c) 1000 1001 0111

10. When does a string belong to the set A of bit strings defined recursively by

$$\lambda \in A$$

$$0x1 \in A \text{ if } x \in A,$$

where λ is the empty string?

Exercises.

11. Let S be the subset of the set of ordered pairs of integers defined recursively by

Basic step: $(0,0) \in S$

Recursive step: If $(a, b) \in S$, then $(a + 2, b + 3) \in S$ and $(a + 3, b + 2) \in S$

Which element is in S ?

- A. $(8, 8)$
- B. $(10, 10)$
- C. $(9, 9)$
- D. $(6, 6)$

RECURSIVE ALGORITHMS

Recursive algorithm

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

For the algorithm to terminate, the instance of the problem must eventually be reduced to some initial case for which the solution is known.

Some Recursive Algorithm

ALGORITHM 1. A Recursive Algorithm for Computing $n!$

procedure *factorial*(n : nonnegative integer)

if $n = 0$ **then return** 1

else return $n \cdot \textit{factorial}(n - 1)$

{output is $n!$ }

Some Recursive Algorithm

ALGORITHM 2. A Recursive Algorithm for Computing a^n .

procedure *power*(*a*: nonzero real number, *n*: nonnegative integer)

if $n = 0$ **then return** 1

else return $a \cdot \textit{power}(a, n - 1)$

{output is a^n }

Some Recursive Algorithm

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 $\gcd(b \bmod a, a)$

{output is $\gcd(a, b)$ }

Some Recursive Algorithm

ALGORITHM 4. A Recursive Linear Search Algorithm

procedure search(i, j, x : integers, $1 \leq i \leq j \leq n$)

if $a_i = x$ **then**

return i

else if $i = j$ **then**

return 0

else

return search($i + 1, j, x$)

{output is the location of x in a_1, a_2, \dots, a_n if it appears; otherwise it is 0}

Some Recursive Algorithm

ALGORITHM 5. A Recursive Binary Search Algorithm.

```
procedure binary search( $i, j, x$ : integers,  $1 \leq i \leq j \leq n$ )  
   $m := \lfloor (i + j)/2 \rfloor$   
  if  $x = a_m$  then  
    return  $m$   
  else if ( $x < a_m$  and  $i < m$ ) then  
    return binary search( $i, m - 1, x$ )  
  else if ( $x > a_m$  and  $j > m$ ) then  
    return binary search( $m + 1, j, x$ )  
  else return 0  
{output is location of  $x$  in  $a_1, a_2, \dots, a_n$  if it appears; otherwise it is 0}
```


Proving Recursive Algorithms Correct

Both mathematical and strong induction are useful techniques to show that recursive algorithms always produce the correct output.

Example. Prove that the algorithm 2 for computing the powers of real numbers is correct

Solution. Use mathematical induction on the exponent n

Basic step: $a^0 = 1$ for every nonzero real number a , and $\text{power}(a, 0) = 1$

Inductive step:

Inductive hypothesis: $\text{power}(a, k) = a^k$, for all $a \neq 0$ is true

Since $\text{power}(a, k + 1) = a \cdot \text{power}(a, k) = a \cdot a^k = a^{k+1}$, the algorithm correctly computes a^{k+1}

Recursive and Iteration

This reading examines recursion more closely by comparing and contrasting it with iteration. Both approaches create repeated patterns of computation. Recursion produces repeated computation by calling the same function recursively, on a simpler or smaller subproblem. Iteration produces repeated computation using for loops or while loops.

If we can come up with an iterative version, do we need recursion at all? In one sense, no, we don't need recursion -- any function we can write recursively could also be written iteratively. But some problems lend themselves naturally to a recursive solution. When we try to solve those kinds of problems iteratively instead, we find ourselves simulating recursion anyway, using an explicit agenda to keep track of what we need to do next.

The converse is also true: any function we can write iteratively, with a loop, could also be written recursively. So it's important to be able to think in both ways and choose the one that is most natural and appropriate to the problem we are trying to solve.

Recursive and Iteration

Recursive

procedure *factorial*(*n*: nonnegative integer)

if $n = 0$ **then return** 1

else return $n \cdot \textit{factorial}(n - 1)$

{output is $n!$ }

Iteration

procedure *factorial*(*n*: nonnegative integer)

result = 1

for *i* **in** range(1, $n + 1$)

 result = result * *i* {output is $n!$ }

return result

Recursive and Iteration

The iterative version might be more efficient, because it doesn't need to create new frames for the recursive calls.

The recursive version might feel simpler, a better match for the mathematical definition of factorial.

The two versions might behave differently with illegal inputs, like $n < 0$.

Fibonacci Numbers Algorithm

A Recursive Algorithm

```
procedure fibonacci(n: nonnegative integer)
if n = 0 then return 0
else if n = 1 then return 1
else return fibonacci(n - 1) + fibonacci(n - 2)
{output is fibonacci(n)}
```

Recursive algorithm uses far more computation than iterative one

An Iterative Algorithm

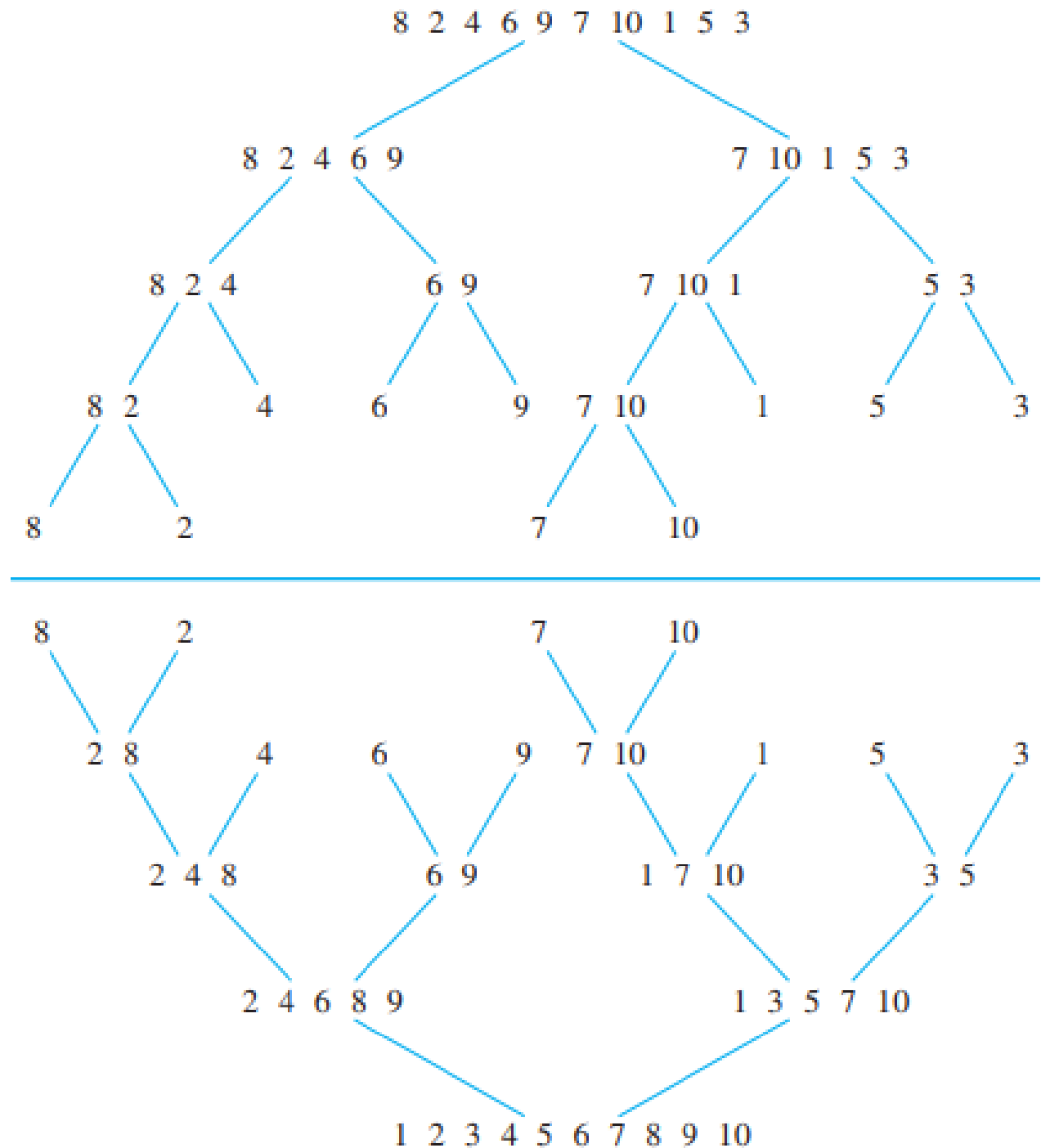
```
procedure iterative fibonacci(n: nonnegative integer)
if n = 0 then return 0
else
  x := 0
  y := 1
  for i := 1 to n - 1
    z := x + y
    x := y
    y := z
  return y
{output is the nth Fibonacci number}
```

The Merge Sort

A merge sort proceeds by iteratively splitting lists into two sublists of equal length (or where one sublist has one more element than the other) until each sublist contains one element. This succession of sublists can be represented by a balanced binary tree. The procedure continues by successively merging pairs of lists, where both lists are in increasing order, into a larger list with elements in increasing order, until the original list is put into increasing order.

The Merge Sort

Example. Use the merge sort to put the terms of the list 8, 2, 4, 6, 9, 7, 10, 1, 5, 3 in increasing order.



The Merge Sort

ALGORITHM. A Recursive Merge Sort.

procedure *mergesort*($L = a_1, \dots, a_n$)

if $n > 1$ **then**

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

$L_1 := a_1, a_2, \dots, a_m$

$L_2 := a_{m+1}, a_{m+2}, \dots, a_n$

$L := \text{merge}(\text{mergesort}(L_1), \text{mergesort}(L_2))$

{ L is now sorted into elements in nondecreasing order}

The Merge Sort

Example. Merge the two lists 2, 3, 5, 6 and 1, 4

Solution.

TABLE 1 Merging the Two Sorted Lists 2, 3, 5, 6 and 1, 4.			
<i>First List</i>	<i>Second List</i>	<i>Merged List</i>	<i>Comparison</i>
2 3 5 6	1 4		$1 < 2$
2 3 5 6	4	1	$2 < 4$
3 5 6	4	1 2	$3 < 4$
5 6	4	1 2 3	$4 < 5$
5 6		1 2 3 4	
		1 2 3 4 5 6	

The Merge Sort

ALGORITHM. 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}

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

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

Exercises

1. Give a recursive algorithm for computing n^x whenever n is a positive integer and x is an integer, using just addition.

2. Consider an recursive algorithm to compute the n th Fibonacci number:

```
procedure Fibo( $n$  : positive integer)
```

```
  if  $n = 1$  return 1
```

```
  else if  $n = 2$  return 1
```

```
    else return  $\text{Fibo}(n - 1) + \text{Fibo}(n - 2)$ 
```

How many additions (+) are used to find $\text{Fibo}(6)$ by the algorithm above?

3. Give a recursive algorithm for finding the sum of the first n odd positive integers.

Exercises

4. Consider the following algorithm:

```
procedure tinh(a: real number; n: positive integer)
```

```
if  $n = 1$  return a
```

```
else return  $a \cdot \text{tinh}(a, n-1)$ .
```

a) What is the output if inputs are: $n = 4$, $a = 2.5$? Explain your answer.

b) Show that the algorithm computes $n \cdot a$ using Mathematical Induction.

5. Consider the following algorithm:

```
procedure F( $a_1, a_2, a_3, \dots, a_n$  : integers)
```

```
if  $n = 0$  return 0
```

```
else return  $a_n + F(a_1, a_2, a_3, \dots, a_{n-1})$ 
```

Find $F(1,3,5)$, $F(1,3,4,7)$, $F(1,2,3,5,9)$

Exercises

6. Which of the following algorithms are recursive?

i. procedure A(n: nonnegative even integer)

if $n = 0$ then $A(n) := 1$;

else $A(n) := A(n - 2) * 3$

ii. procedure A(n: nonnegative even integer)

if $n = 0$ then $y := 1$;

else

begin

$y := 1$; $m = n \text{ div } 2$;

for $i := 1$ to m

$y := y * 3$;

end

A. only i B. both of them C. only ii D. none of them

Summary

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

Thanks