Induction and Recursion

Chapter 5

Chapter Summary

- ✓Mathematical Induction
- ✓Strong Induction
- ✓Well-Ordering
- ✓Recursive Definitions
- ✓Structural Induction
- ✓Recursive Algorithms
- ✓Program Correctness

Mathematical Induction

Section 5.1

Introduction

Mathematical induction is used to prove propositions of the form $\forall n P(n)$, where the universe of discourse is the set of positive integers.

We will discuss:

- **How mathematical induction can be used?**
- **Ways to remember how mathematical induction works**
- **<u>Why is mathematical induction valid?</u>**
- **The good and bad of mathematical induction**

Climbing an Infinite Ladder

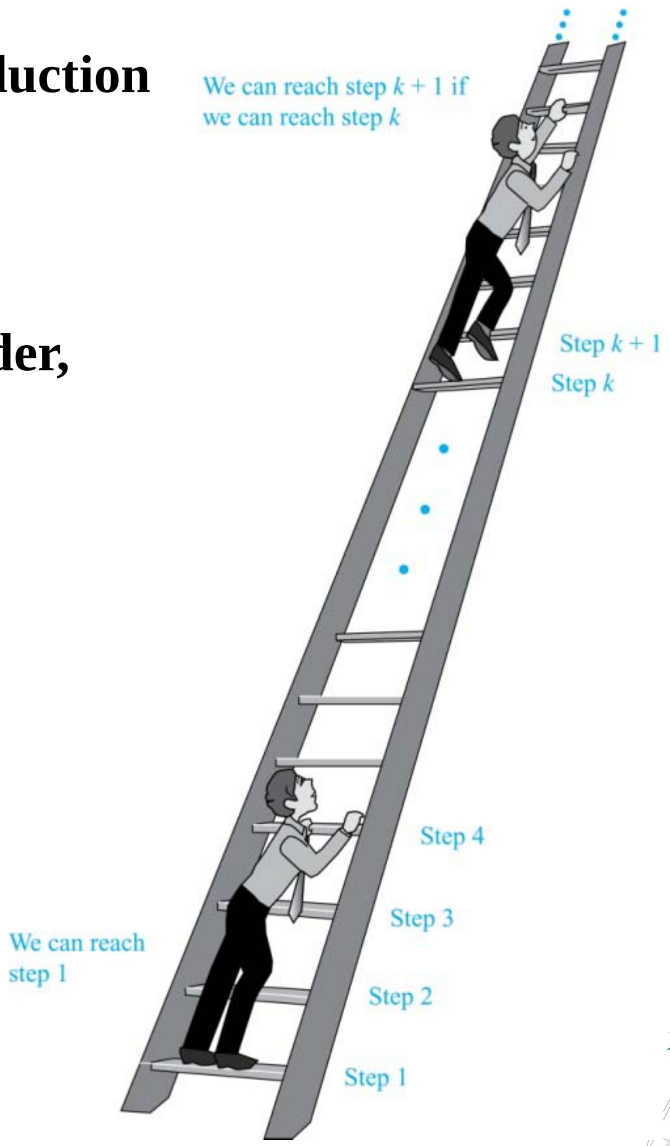
- An example motivates proof by mathematical induction

Suppose we have an infinite ladder:

1. **We can reach the first rung of the ladder.**
2. **If we can reach a particular rung of the ladder, then we can reach the next rung.**

We can reach every rung?

From (1), we can reach the first rung. Then by applying (2), we can reach the second rung. Applying (2) again, the third rung. And so on. We can apply (2) any number of times to reach any particular rung, no matter how high up.



Principle of Mathematical Induction

The (first) principle of Mathematical Induction

$$(P(1) \wedge \forall k(P(k) \rightarrow P(k+1))) \rightarrow \forall n\, P(n)$$

where the domain is the set of positive integers.

The principle has the following form.

$$\begin{array}{l} P(1) \\ \underline{P(k) \rightarrow P(k+1)} \\ \therefore \forall n P(n) \end{array}$$

The procedure for mathematical induction:

(1) Inductive base: Establish $P(1)$

(2) Inductive step: Prove that $P(k) \to P(k+1)$ for $k \geq 1$

Conclusion : The inductive base and the inductive step together imply $P(n)$ $\forall\ n \geq 1$.

Climbing an Infinite Ladder Example:

•BASIS STEP: By (1), we can reach rung 1.

•INDUCTIVE STEP: Assume the inductive hypothesis that we can reach rung k. Then by (2), we can reach rung $k + 1$.

Hence, $P(k) \to P(k + 1)$ is true for all positive integers k. We can reach every rung on the ladder.

More general form

$$\forall n[n \geq \kappa \rightarrow P(n)]$$

The procedure :

(1) Inductive base: Establish $P(k)$

(2) Inductive step: Prove that $P(n) \rightarrow P(n+1)$ for $n \geq k$

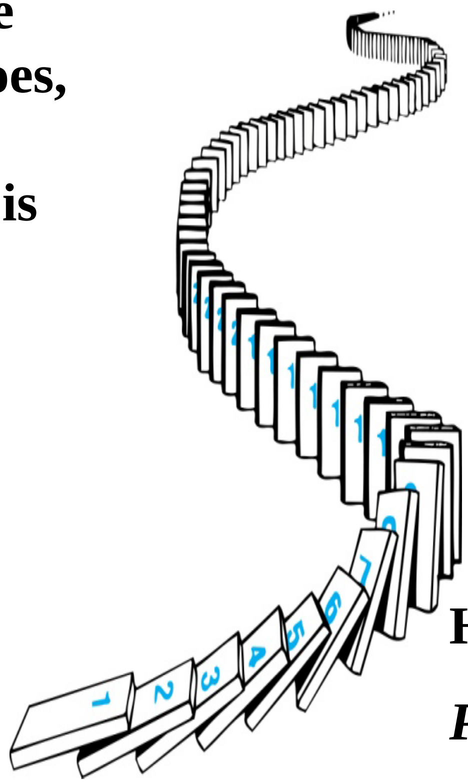
Conclusion: The inductive base and the inductive step together imply $P(n)$ $\forall$ $n \geq k$.

How Mathematical Induction Works?

◆ ***If the nth domino falls over the (n+1)st must fall over so pushing the first one down means all must fall down.***

Consider an infinite sequence of dominoes, labeled 1,2,3, …, where each domino is standing.

Let *P*(*n*) be the proposition that the *n*th domino is knocked over.



We know that the first domino is knocked down, i.e., *P*(1) is true .

We also know that if whenever the *k*th domino is knocked over, it knocks over the (*k* + 1)st domino, i.e, *P*(*k*) → *P*(*k* + 1) is true for all positive integers *k*.

Hence, all dominos are knocked over.

***P*(*n*) is true for all positive integers *n*.**

Why Is Mathematical Induction Valid?

The validity of mathematical induction follows from the *well-ordering property* .

A set S is *well ordered* if every subset has a least element.

For example,

1) **N is well ordered (under the ≤ relation)**
2) **Z is not well ordered under the ≤ relation (Z has no smallest element).**
3) **[0, 1] is not well ordered since (0,1) does not have a least element.**

> ***The well-ordering property***
>
> **Every nonempty set of nonnegative integers has a least element.**

Why mathematical induction is valid?

$$\left.\begin{array}{l} P(1) \\ \forall k(P(k)\rightarrow P(k+1)) \end{array}\right\} \Longrightarrow \forall n P(n)$$

Proof:

Assume that there is at least one positive integer for which $P(n)$ is false.

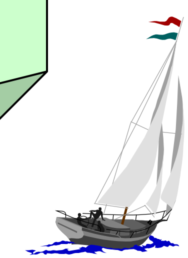
S: the set of positive integer for which $P(n)$ is false.

Then S is nonempty.

By the well-ordering property, S has a least element, which will be denoted by m.

Then $m \neq 1$, m -1 is a positive integer. m-1 is not in S. So $P(m$ - 1) is true.

Since the implication $P(k) \rightarrow P(k+1)$ is also true, $P(m)$ must be true.



The Good and Bad of Mathematical Induction

- **The good**
  - can be used to prove a conjecture once it is has been made and is true.

- **The bad**
  - Proofs do not provide insights as to why theorems are true
  - Cannot be used to find new theorems

You can prove a theorem by M.I. even if you do not have the slightest idea why it is true!

Examples of Proofs by Mathematical Induction

- Proving summation formular
- Proving inequalities holding for all positive integers greater than a particular positive integer
- Proving results about algorithms

For example,

M.I. can be used to prove that some greedy algorithms always yields an optimal solution.

Strong Induction and Well-ordering

Section 5.2

Strong Induction

The Second Principle of Mathematical Induction (Strong Induction, complete induction)

$$(P(n_0) \wedge \forall k\ (\ k >= n_0 \wedge P(n_0) \wedge P(n_0+1) \wedge \ldots \wedge P(k) \rightarrow \mathrm{P}(k+1))) \rightarrow \forall n\ P(n)$$

The procedure :

(1) Inductive base: Establish $P(n_0)$

(3) Inductive step: Prove $P(n_0) \wedge P(n_0+1) \wedge \ldots \wedge P(k) \rightarrow \mathrm{P}(k+1)$

Conclusion: The inductive base and the inductive step allow one to conclude that $P(n)\ \forall n \geq n_0$

〖Example 1〗 Show that if n is an integer greater than 1, then n can be written uniquely as a prime or the product of two or more primes.

Solution:

Let $P(n)$ be n can be written uniquely as a prime or the product of two or more primes

(1) Inductive base

$P(2)$ is true.

(2) Inductive step

Assume that $P(k)$ is true for all positive integers k with $k \le n$. Show that $P(k+1)$ is true under the assumption.

Proof by cases.

- **$k+1$ is prime**

  $P(k+1)$ is true.

- **$k+1$ is composite**

  $k+1=a \times b,\ 2 \le a, b < n+1$

Note:

1. The validity of both mathematical induction and strong induction follow from the well-ordering property.

2. In fact, **mathematical induction**, **strong induction**, and **well-ordering** are all equivalent principles.

Questions:

How to establish?

Another Strong Induction Proof Example

Some terms:

- **polygon**
- **side, vertex**
- **a polygon is simple**
- **Every simple polygon divides the plane into two regions: its interior, its exterior.**
- **convex, nonconvex**
- **diagonal, interior diagonal**
- **triangulation**

【LEMMA 1】Every simple polygon with at least four sides has an interior diagonal.

【Theorem 1】A simple polygon with *n* sides, where *n* is an integer with $n \geq 3$, can be triangulated into *n*-2 triangles.

Proof:

Let *T(n)* be the statement that simple polygon with *n* sides can be triangulated into *n*-2 triangles

(1) Inductive base

$T(3)$ is true.

(2) Inductive step

Assume that $T(j)$ is true for all integers *j* with $3 \leq j \leq k$. We must show $T(k+1)$ is true, that is that every simple polygon with *k*+1 sides can be triangulated into *k*-1 triangles.

Suppose that we have a simple polygon P with *k*+1 sides. By Lemma 1, P has an interior diagonal *ab*. *ab* splits P into two simple polygon Q, with *s* $(3 \leq s \leq k)$ sides, and R, with *t* $(3 \leq t \leq k)$ sides.

Proofs by the Well-ordering property

〖Example 2〗 Use the well-ordering property to prove the division algorithm. Recall that the division algorithm states that if *a* be an integer and *d* a positive integer, Then there are unique integers *q* and *r*, with $0 \leq r < d$ such that $a = dq + r$.

Solution:

Let *S* be the set of nonnegative integers of the form $a-dq$, where *q* is an integer. This set in nonempty.

By the well-ordering property, S has a least element $r=a-dq_0$.

The integer *r* is nonnegative. It is also the case that $r<d$. If it were not , then there would be a smaller nonnegative element in *S*, namely, $r=a-d(q_0+1)$.

Recursive Definition and Structural Induction

Section 5.3

Why Recursive definition?

Sometimes it is difficult to express the members of an object or numerical sequence explicitly.

For example,

The Fibonacci sequence:

$\{f_n\} = 0,1,1,2,3,5,8,13,21,34,55,\ldots$

***Recursion* is a principle closely related to mathematical induction.**

In a *recursive definition*, an object is defined in terms of itself.

We can recursively define *sequences, functions* and *sets*.

Recursively defined functions

Recursively defined functions, **with <u>the set of nonnegative integers as its domain</u>:**

- **Basis Step: Specify the value of the function at zero.**
- **Recursive Step: Give the rules for finding its value at an integer from its value at smaller integers.**

〖Example 1〗 Give a recursive definition of factorial function.

Solution:

$$f(n) = n! = 1 \times 2 \times \ldots \times (n-1) \times n$$

$$f(0) = 1$$

$$f(n) = nf(n-1)$$

Are recursively defined functions well-defined?

Let *P*(*n*) be the statement “*f* is well-defined at *n* .

(1) *P*(0) is true.

(2) Assume that *P*(*n-1*) is true. Then *f* is well-defined at *n*.

Since *f*(*n*) is given in terms of some *f*(*n-1*) .

The Recursive definition of the Fibonacci numbers and relative proofs

〖Example 2〗 Give a recursive definition of the Fibonacci numbers

$$\{f_n\} = 0,1,1,2,3,5,8,13,21,34,55,\ldots$$

Note:

The Fibonacci number f_n can be thought either as the nth term of the sequence of Fibonacci numbers f_0, f_1,…or as the value at the integer n of a function $f(n)$.

Solution:

The Fibonacci numbers can be defined

$$f_0 = 0$$

$$f_1 = 1$$

$$f_n = f_{n-1} + f_{n-2} \text{ for } n=2,3,4,\ldots$$

〖Example 3〗 **Show that**

$$f_n > \alpha^{n-2} \quad where \quad \alpha = \frac{1+\sqrt{5}}{2} \quad whenever \quad n \geq 3.$$

Proof:

(1) Inductive base

$$f_3 = 2 > \alpha = \frac{1+\sqrt{5}}{2} \qquad f_4 = 3 > \alpha^2 = (\frac{1+\sqrt{5}}{2})^2$$

(2) Inductive step

Assume that *P*(*k*) is true, namely, that $f_k > \alpha^{k-2}$ **for all integers *k* with** $3 \leq k \leq n$ **.**

We must show that *P*(*k*+1) is true, that is $f_{k+1} > \alpha^{k-1}$ **.**

Since α **is the solution of** $x^2 - x - 1 = 0$ **, it follows that** $\alpha^2 = \alpha + 1$

Therefore,

$$\alpha^{k-1} = \alpha^2 \cdot \alpha^{k-3} = (\alpha+1) \cdot \alpha^{k-3} = \alpha^{k-2} + \alpha^{k-3}$$

By the inductive hypothesis, if $n \geq 5$ **, it follows that**

$$f_{k-1} > \alpha^{k-3}, f_k > \alpha^{k-2}$$

Therefore, $f_{k+1} = f_{k-1} + f_k > \alpha^{k-3} + \alpha^{k-2} = \alpha^{k-1}$

The Complexity of Euclidean algorithm

Euclidean algorithm can used to find the greatest common divisor of the positive integers a and b, where a≥b.

Let $a=bq+r$, where a, b, q, and r are integers.

Then gcd (a, b)=gcd (r, b)

$a = r_0, b = r_1$

$r_0 = r_1 q_1 + r_2 \qquad 0 \le r_2 < r_1$

$r_1 = r_2 q_2 + r_3 \qquad 0 \le r_3 < r_2$

…

$r_{n-2} = r_{n-1} q_{n-1} + r_n \qquad 0 \le r_n < r_{n-1}$

$r_{n-1} = r_n q_n$

gcd (a, b)=gcd (r_{n-1}, r_n)= r_n

For example, find gcd (662,414)

662=414×1+248

414=248×1+166

248=166×1+82

166=82×2+2

82=2×41

hence, gcd (662,414)=2

The complexity of Euclidean algorithm?

LAME'S Theorem Let a , b be positive integers with $a \geq b$.
Then the number of d
to find gcd (a, b) is les
of decimal digits in b.

$$r_0 = r_1 q_1 + r_2 \qquad 0 \leq r_2 < r_1$$

$$r_1 = r_2 q_2 + r_3 \qquad 0 \leq r_3 < r_2$$

$$\ldots$$

$$r_{n-2} = r_{n-1} q_{n-1} + r_n \qquad 0 \leq r_n < r_{n-1}$$

$$r_{n-1} = r_n q_n$$

Proof:

$n \leq 5k$

From the above equations,

$q_1, q_2, \ldots, q_{n-1} \geq 1$ $\qquad$ $q_n \geq 2 \quad \because r_n < r_{n-1}$

This implies

$$r_n \geq 1 = f_2$$

$$r_{n-1} \geq 2r_n \geq 2 f_2 = f_3$$

$$r_{n-2} \geq r_{n-1} + r_n \geq f_3 + f_2 = f_4$$

$$r_{n-3} \geq r_{n-2} + r_{n-1} \geq f_3 + f_4 = f_5$$

…

$$r_2 \geq r_3 + r_4 \geq f_{n-1} + f_{n-2} = f_n$$

$$b = r_1 \geq r_2 + r_3 \geq f_n + f_{n-1} = f_{n+1}$$

$$\square \; f_{n+1} > \alpha^{n-1} \text{ for } n > 2$$

$$\therefore b > \alpha^{n-1}$$

$$\log_{10} b > (n-1)\log_{10}\alpha > \frac{n-1}{5}$$

$$n - 1 < 5 \cdot \log_{10} b < 5k$$

$$\therefore n \leq 5k$$

$\boldsymbol{r_0 = r_1\, q_1 + r_2}$ $\quad$ $\boldsymbol{0 \leq r_2 < r_1}$

$\boldsymbol{r_1 = r_2\, q_2 + r_3}$ $\quad$ $\boldsymbol{0 \leq r_3 < r_2}$

…

$\boldsymbol{r_{n-2} = r_{n-1}\, q_{n-1} + r_n}$ $\quad$ $\boldsymbol{0 \leq r_n < r_{n-1}}$

$\boldsymbol{r_{n-1} = r_n\, q_n}$

Recursively defined sets

- Sets can be defined recursively.
  - **Basis Step**: Specify an initial collection of elements.
  - **Recursive Step**: Give the rules for constructing elements of the set from other elements already in the set.
- Sets described in this way are well-defined.

〖Example 4〗 Consider the subset of the set of integers defined by

Basis Step: $3 \in S$

Recursive Step: if $x \in S$ and $y \in S$, then $x+y \in S$

◆ **Recursive definitions play an important role in the study of strings.**

〖Example 6〗 The set Σ^* of strings over the alphabet Σ.

Solution:

Basis Step: **$\lambda \in \Sigma^*$, where λ is the empty string containing no symbols.**

Recursive Step: **$\omega x \in \Sigma^*$ whenever $\omega \in \Sigma^*$ and $x \in \Sigma$.**

Questions:

- **How to define the set of all bit strings?**
- **The concatenation of two strings?**
- **The length of a string, $l(\omega)$?**

◆ Another important use of Recursive definitions to define well-formed formulae of various type.

〖Example 5〗 **Well-formed formulae for compound proposition.**

Solution:

Basis Step: ***T***, ***F***, **and *p*, where *p* is a propositional variable, are well-formed formulae.**

Recursive Step: $(\neg p)$, $(p \vee q)$, $(p \wedge q)$, $(p \rightarrow q)$, $(p \leftrightarrow q)$ **are well-formed formulae if *p* and *q* are well-formed formulae.**

The set of rooted trees, extended binary trees and full binary trees can be defined recursively.

How?

Structural Induction

To prove results about recursively defined sets, we can use

- **Mathematical induction**
- **Structural induction**

A proof by structural induction:

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

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

$p(n)$: the result is true for all elements of the set that are generated by n or fewer applications of the rules in the recursive step of a recursive definition.

Basis Step: **Show that $p(0)$ is true.**

Recursive Step: **if we assume $p(k)$ is true, it follows that $p(k+1)$ is true.**

Examples of Proofs Using Structural Induction

〖Example 7〗Show that every well-formed formula for compound propositions, as defined in Example 5, contains an equal number of left and right parentheses.

Proof:

Basis Step: Show that the result is true for *T*, *F*, and *p*, whenever *p* is a propositional variable.

Recursive Step: Show that if the result is true for the compound propositions *p* and *q* , it is also true for $(\neg p)$, $(p \vee q)$, $(p \wedge q)$, $(p \rightarrow q)$, $(p \leftrightarrow q)$.

Recursive Algorithms

Section 5.4

Section Summary

- ✓Recursive Algorithms
- ✓Proving Recursive Algorithms Correct
- ✓Recursion and Iteration

Recursive Algorithms

- 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.**

Recursive Factorial Algorithm

〖Example 1〗 Give a recursive algorithm for computing the factorial function *n*! .

$$n! = \text{factorial}(n) = \begin{cases} 1, & \text{if } \; n = 0 \\ n \cdot \text{factorial}(n-1), & \text{if } \; n > 0 \end{cases}$$

Algorithm 1 A Recursive Procedure for Factorials.

procedure factorial(*n*:nonnegative integer)

{

if *n*=0 then

factorial(*n*):=1

else

factorial(*n*) := *n*×factorial(*n*-1)

}

How this algorithm works?

Recursive GCD Algorithm

〖Example 2〗 Give a recursive algorithm for computing the greatest common divisor of two nonnegative integers a and b with $a<b$.

> 1. **How this algorithm works?**
> 2. **Is this recursive algorithms correct?**

Solution: Use the reduction

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

and the condition $\gcd(0,b) = b$ when $b > 0$.

Algorithm 2 A Recursive Algorithm for Computing gcd(a,b).

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

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 for computing the powers of real numbers is correct.

> **procedure** *power*(a: nonzero real number, n: nonnegative integer)
> **if** $n = 0$ **then return** 1
> **else return** $a \cdot$ *power* $(a, n - 1)$
> {output is a^n}

Solution: Use mathematical induction on the exponent n.

BASIS STEP: $a^0 = 1$ for every nonzero real number a, and *power*$(a,0) = 1$

INDUCTIVE STEP: The inductive hypothesis is that *power*$(a,k) = a^k$, for all $a \neq 0$. Assuming the inductive hypothesis, the algorithm correctly computes a^{k+1}, since

$$power(a,k+1) = a \cdot power(a,k) = a \cdot a^k = a^{k+1}.$$

Recursion and Iteration

- **Recursion**
  - Successively reducing the computation to the evaluation of the function an smaller integers
- **Iteration**
  - Start with the value of the function at one or more integers, the base cases, and successively apply the recursive definition to find the value of the function at successive large integers.

Recursion and Iteration

Algorithm 3 A Recursive Algorithm for Fibonacci Numbers.

procedure fibo(n: nonnegative integer)
if $n = 0$ **then** fibo(0) := 0
else if $n = 1$ **then** fibo(1) := 1
else fibo(n) := fibo($n - 1$) + fibo($n - 2$)

Algorithm 4 An Iterative Algorithm for Fibonacci Numbers.

```
procedure iterative_fibo(n: nonnegative integer)
if n = 0 then y := 0
else
begin
        x := 0
        y := 1
        for i := 1 to n-1
        begin
                z := x + y
                x : = y
                y := z
        end
end   {y is the n-th Fibonacci number}
```

Note:

- **For every recursive algorithm, there is an equivalent iterative algorithm.**
- **Recursive algorithms are often shorter, more elegant, and easier to understand than their iterative counterparts.**
- **However, iterative algorithms are usually more efficient in their use of space and time.**

Homework:

Seventh Edition: P.331 43,61

P.344 31,41

P.358 10,12,31,50,59 (b)(d)