Induction and recursion

M,NAGA SRAVANI, ASSIST.PROF

Mathematical Induction

M,NAGA SRAVANI, ASSIST.PROF

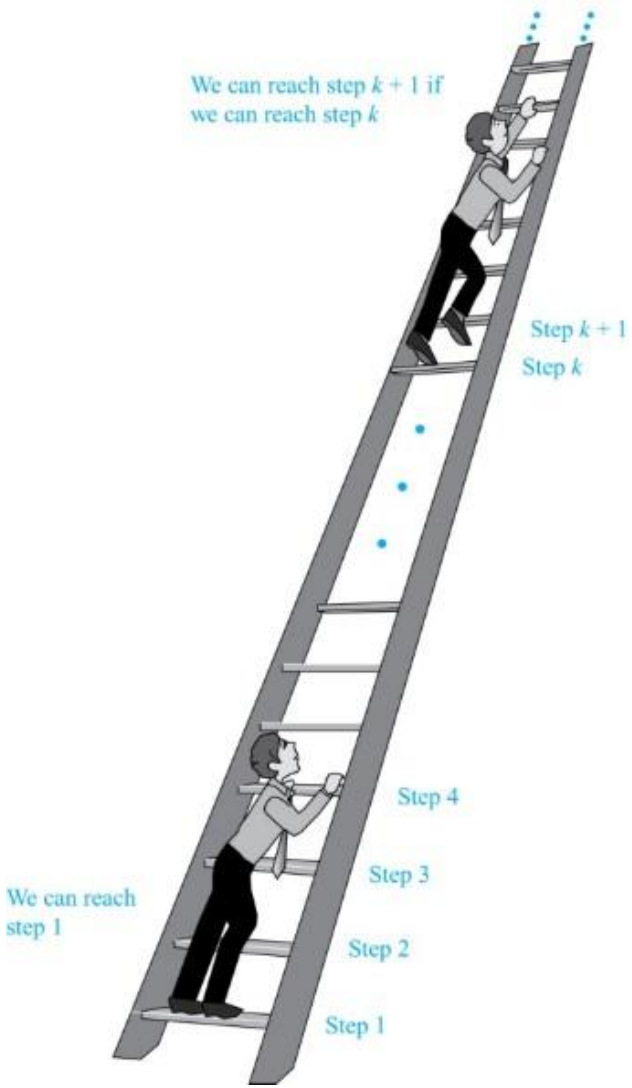
Climbing an Infinite Ladder

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.

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.

This example motivates proof by mathematical induction.



Principle of Mathematical Induction

Principle of Mathematical Induction: To prove that $P(n)$ is true for all positive integers n, we complete these steps:

- *Basis Step*: Show that $P(1)$ is true.
- *Inductive Step*: Show that $P(k) \rightarrow P(k + 1)$ is true for all positive integers k.

To complete the inductive step, assuming the *inductive hypothesis* that $P(k)$ holds for an arbitrary integer k, show that must $P(k + 1)$ be true.

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

M,NAGA SRAVANI, ASSIST.PROF

Important Points About Using Mathematical Induction

Mathematical induction can be expressed as the rule of inference

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

where the domain is the set of positive integers.

In a proof by mathematical induction, we don't assume that $P(k)$ is true for all positive integers! We show that if we assume that $P(k)$ is true, then $P(k + 1)$ must also be true.

Proofs by mathematical induction do not always start at the integer 1. In such a case, the basis step begins at a starting point b where b is an integer. We will see examples of this soon.

Validity of Mathematical Induction

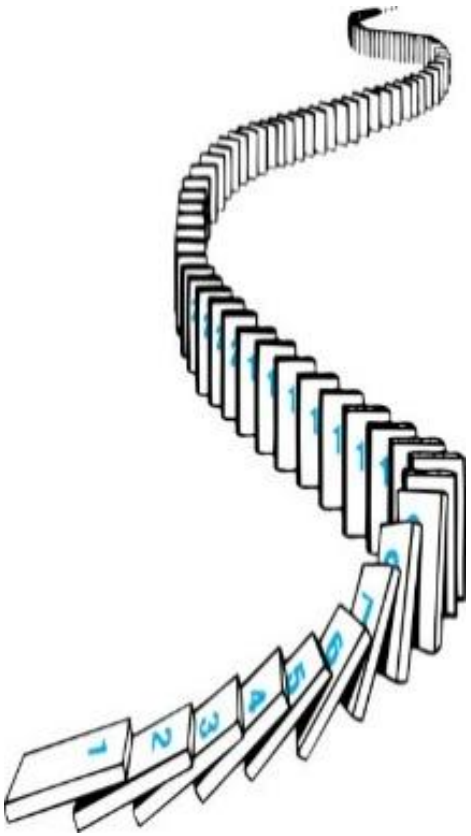
Mathematical induction is valid because of the well ordering property, which states that every nonempty subset of the set of positive integers has a least element (*see Section* 5.2 *and Appendix* 1). Here is the proof:

- Suppose that $P(1)$ holds and $P(k) \rightarrow P(k+1)$ is true for all positive integers k.
- Assume there is at least one positive integer n for which $P(n)$ is false. Then the set S of positive integers for which $P(n)$ is false is nonempty.
- By the well-ordering property, S has a least element, say m.
- We know that m can not be 1 since $P(1)$ holds.
- Since m is positive and greater than 1, $m - 1$ must be a positive integer. Since $m - 1 < m$, it is not in S, so $P(m - 1)$ must be true.
- But then, since the conditional $P(k) \rightarrow P(k+1)$ for every positive integer k holds, $P(m)$ must also be true. This contradicts $P(m)$ being false.
- Hence, $P(n)$ must be true for every positive integer n.

Remembering How Mathematical Induction Works

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

Let $P(n)$ be the proposition that the nth 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 kth domino is knocked over, it knocks over the $(k + 1)$st domino, i.e, $P(k) \rightarrow P(k + 1)$ is true for all positive integers k.

Hence, all dominos are knocked over.

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

M,NAGA SRAVANI, ASSIST.PROF

Proving a Summation Formula by Mathematical Induction

Example: Show that: $\sum_{i=1}^{n} = \frac{n(n+1)}{2}$

Solution:

Note: Once we have this conjecture, mathematical induction can be used to prove it correct.

- BASIS STEP: $P(1)$ is true since $1(1+1)/2 = 1$.
- INDUCTIVE STEP: Assume true for $P(k)$.

The inductive hypothesis is $\sum_{i=1}^{k} = \frac{k(k+1)}{2}$

Under this assumption,

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

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

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

Conjecturing and Proving Correct a Summation Formula

Example: Conjecture and prove correct a formula for the sum of the first n positive odd integers. Then prove your conjecture.

Solution: We have: $1=1,\ 1+3=4,\ 1+3+5=9,\ 1+3+5+7=16,\ 1+3+5+7+9=25.$

- We can conjecture that the sum of the first n positive odd integers is n^2,

$$1+3+5+\cdots+(2n-1)=n^2.$$

- We prove the conjecture is proved correct with mathematical induction.
- BASIS STEP: $P(1)$ is true since $1^2 = 1.$
- INDUCTIVE STEP: $P(k) \longrightarrow P(k+1)$ for every positive integer k.

  Assume the inductive hypothesis holds and then show that $P(k+1)$ holds has well.

  Inductive Hypothesis: $1+3+5+\cdots+(2k-1)=k^2$

- So, assuming $P(k)$, it follows that:

$$\begin{aligned}1+3+5+\cdots+(2k-1)+(2k+1) &= \left[1+3+5+\cdots+(2k-1)\right]+(2k+1)\\ &= k^2+(2k+1)\left(\textit{by the inductive hypothesis}\right)\\ &= k^2+2k+1\\ &= (k+1)^2\end{aligned}$$

- Hence, we have shown that $P(k+1)$ follows from $P(k)$. Therefore the sum of the first n positive odd integers is n^2.

Proving Inequalities[1]

Example: Use mathematical induction to prove that $n < 2^n$ for all positive integers n.

Solution: Let $P(n)$ be the proposition that $n < 2^n$.

- BASIS STEP: $P(1)$ is true since $1 < 2^1 = 2$.
- INDUCTIVE STEP: Assume $P(k)$ holds, i.e., $k < 2^k$, for an arbitrary positive integer k.
- Must show that $P(k + 1)$ holds. Since by the inductive hypothesis, $k < 2^k$, it follows that:

$$k + 1 < 2^k + 1 \leq 2^k + 2^k = 2 \cdot 2^k = 2^{k+1}$$

Therefore $n < 2^n$ holds for all positive integers n.

Proving Inequalities 2

Example: Use mathematical induction to prove that $2^n < n!$, for every integer $n \geq 4$.

Solution: Let $P(n)$ be the proposition that $2^n < n!$.

- BASIS STEP: $P(4)$ is true since $2^4 = 16 < 4! = 24$.
- INDUCTIVE STEP: Assume $P(k)$ holds, i.e., $2^k < k!$ for an arbitrary integer $k \geq 4$. To show that $P(k+1)$ holds:

$$\begin{aligned} 2^{k+1} &= 2 \cdot 2^k \\ &< 2 \cdot k! \qquad \left(\textit{by the inductive hypothesis}\right) \\ &< (k+1)k! \\ &= (k+1)! \end{aligned}$$

Therefore, $2^n < n!$ holds, for every integer $n \geq 4$.

Note that here the basis step is $P(4)$, since $P(0)$, $P(1)$, $P(2)$, and $P(3)$ are all false.

Proving Divisibility Results

Example: Use mathematical induction to prove that $n^3 - n$ is divisible by 3, for every positive integer n.

Solution: Let $P(n)$ be the proposition that $n^3 - n$ is divisible by 3.

- BASIS STEP: $P(1)$ is true since $1^3 - 1 = 0$, which is divisible by 3.
- INDUCTIVE STEP: Assume $P(k)$ holds, i.e., $k^3 - k$ is divisible by 3, for an arbitrary positive integer k. To show that $P(k+1)$ follows:

$$(k+1)^3 - (k+1) = (k^3 + 3k^2 + 3k + 1) - (k+1)$$

$$= (k^3 - k) + 3(k^2 + k)$$

By the inductive hypothesis, the first term $(k^3 - k)$ is divisible by 3 and the second term is divisible by 3 since it is an integer multiplied by 3. So by part (i) of Theorem 1 in Section 4.1 , $(k+1)^3 - (k+1)$ is divisible by 3.

Therefore, $n^3 - n$ is divisible by 3, for every integer positive integer n.

Number of Subsets of a Finite Set 1

Example: Use mathematical induction to show that if *S* is a finite set with n elements, where *n* is a nonnegative integer, then *S* has 2^n subsets.

(*Chapter* 6 *uses combinatorial methods to prove this result.*)

Solution: Let *P*(*n*) be the proposition that a set with *n* elements has 2^n subsets.

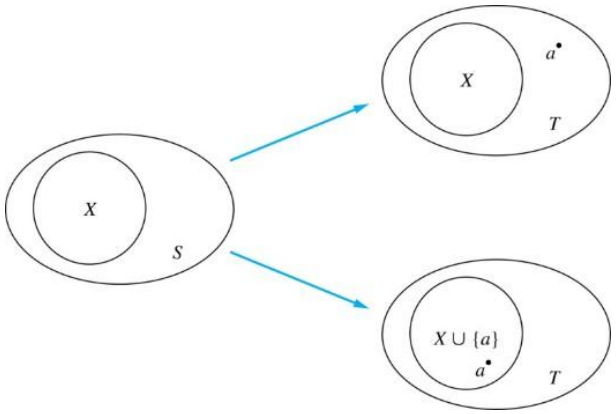
- Basis Step: *P*(0) is true, because the empty set has only itself as a subset and $2^0 = 1$.
- Inductive Step: Assume *P*(*k*) is true for an arbitrary nonnegative integer *k*.

Number of Subsets of a Finite Set 2

Inductive Hypothesis: For an arbitrary nonnegative integer k, every set with k elements has 2^k subsets.

Let T be a set with $k + 1$ elements. Then $T = S \cup \{a\}$, where $a \in T$ and $S = T - \{a\}$. Hence $|S| = k$.

For each subset X of S, there are exactly two subsets of T, i.e., X and $X \cup \{a\}$.



By the inductive hypothesis S has 2^k subsets. Since there are two subsets of T for each subset of S, the number of subsets of T is

$$2 \cdot 2^k = 2^{k+1}.$$

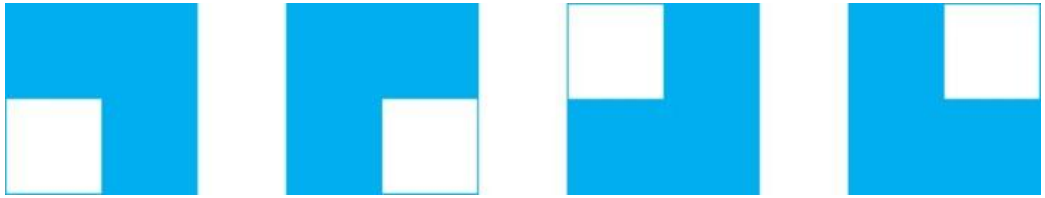
Tiling Checkerboards[1]

Example: Show that every $2^n \times 2^n$ checkerboard with one square removed can be tiled using right triominoes.

A right triomino is an L-shaped tile which covers three squares at a time.

Solution: Let $P(n)$ be the proposition that every $2^n \times 2^n$ checkerboard with one square removed can be tiled using right triominoes. Use mathematical induction to prove that $P(n)$ is true for all positive integers n.

- BASIS STEP: P(1) is true, because each of the four 2 × 2 checkerboards with one square removed can be tiled using one right triomino.

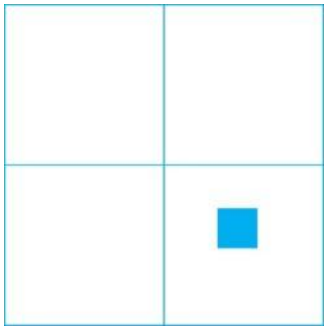


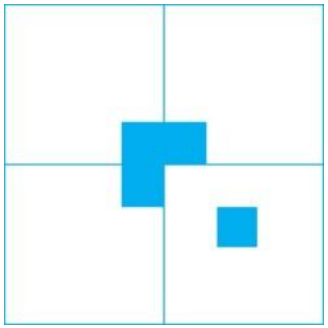
- INDUCTIVE STEP: Assume that $P(k)$ is true for every $2^k \times 2^k$ checkerboard, for some positive integer k.

Tiling Checkerboards 2

Inductive Hypothesis: Every $2^k \times 2^k$ checkerboard, for some positive integer k, with one square removed can be tiled using right trominoes.

Consider a $2^{k+1} \times 2^{k+1}$ checkerboard with one square removed. Split this checkerboard into four checkerboards of size $2^k \times 2^k$, by dividing it in half in both directions.





Remove a square from one of the four $2^k \times 2^k$ checkerboards. By the inductive hypothesis, this board can be tiled. Also by the inductive hypothesis, the other three boards can be tiled with the square from the corner of the center of the original board removed. We can then cover the three adjacent squares with a triomino.

Hence, the entire $2^{k+1} \times 2^{k+1}$ checkerboard with one square removed can be tiled using right trominoes.

An Incorrect "Proof" by Mathematical Induction[1]

Example: Let $P(n)$ be the statement that every set of n lines in the plane, no two of which are parallel, meet in a common point. Here is a "proof" that $P(n)$ is true for all positive integers $n \geq 2$.

- BASIS STEP: The statement $P(2)$ is true because any two lines in the plane that are not parallel meet in a common point.
- INDUCTIVE STEP: The inductive hypothesis is the statement that $P(k)$ is true for the positive integer $k \geq 2$, i.e., every set of k lines in the plane, no two of which are parallel, meet in a common point.
- We must show that if $P(k)$ holds, then $P(k + 1)$ holds, i.e., if every set of k lines in the plane, no two of which are parallel, $k \geq 2$, meet in a common point, then every set of k + 1 lines in the plane, no two of which are parallel, meet in a common point.

An Incorrect "Proof" by Mathematical Induction$_2$

Inductive Hypothesis: Every set of k lines in the plane, where $k \geq 2$, no two of which are parallel, meet in a common point.

Consider a set of $k + 1$ distinct lines in the plane, no two parallel. By the inductive hypothesis, the first k of these lines must meet in a common point p_1. By the inductive hypothesis, the last k of these lines meet in a common point p_2.

If p_1 and p_2 are different points, all lines containing both of them must be the same line since two points determine a line. This contradicts the assumption that the lines are distinct. Hence, $p_1 = p_2$ lies on all $k + 1$ distinct lines, and therefore $P(k + 1)$ holds. Assuming that $k \geq 2$, distinct lines meet in a common point, then every $k + 1$ lines meet in a common point.

There must be an error in this proof since the conclusion is absurd. But where is the error?

- **Answer**: $P(k) \rightarrow P(k + 1)$ only holds for $k \geq 3$. It is not the case that $P(2)$ implies $P(3)$. The first two lines must meet in a common point p_1 and the second two must meet in a common point p_2. They do not have to be the same point since only the second line is common to both sets of lines.

Guidelines: Mathematical Induction Proofs

Template for Proofs by Mathematical Induction

1. Express the statement that is to be proved in the form "for all $n \geq b$, $P(n)$" for a fixed integer b.
2. Write out the words "Basis Step." Then show that $P(b)$ is true, taking care that the correct value of b is used. This completes the first part of the proof.
3. Write out the words "Inductive Step".
4. State, and clearly identify, the inductive hypothesis, in the form "assume that $P(k)$ is true for an arbitrary fixed integer $k \geq b$."
5. State what needs to be proved under the assumption that the inductive hypothesis is true. That is, write out what $P(k + 1)$ says.
6. Prove the statement $P(k + 1)$ making use the assumption $P(k)$. Be sure that your proof is valid for all integers k with $k \geq b$, taking care that the proof works for small values of k, including $k = b$.
7. Clearly identify the conclusion of the inductive step, such as by saying "this completes the inductive step."
8. After completing the basis step and the inductive step, state the conclusion, namely, by mathematical induction, $P(n)$ is true for all integers n with $n \geq b$.

M,NAGA SRAVANI, ASSIST.PROF

Strong Induction and Well-Ordering

M,NAGA SRAVANI, ASSIST.PROF

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:

- *Basis Step*: Verify that the proposition $P(1)$ is true.
- *Inductive Step*: Show the conditional statement

$$\left[P(1) \wedge P(2) \wedge \boxtimes \wedge P(k)\right] \rightarrow P(k+1)$$

holds for all positive integers k.

Strong Induction is sometimes called the *second principle of mathematical induction* or *complete induction*.

Strong Induction and the Infinite Ladder

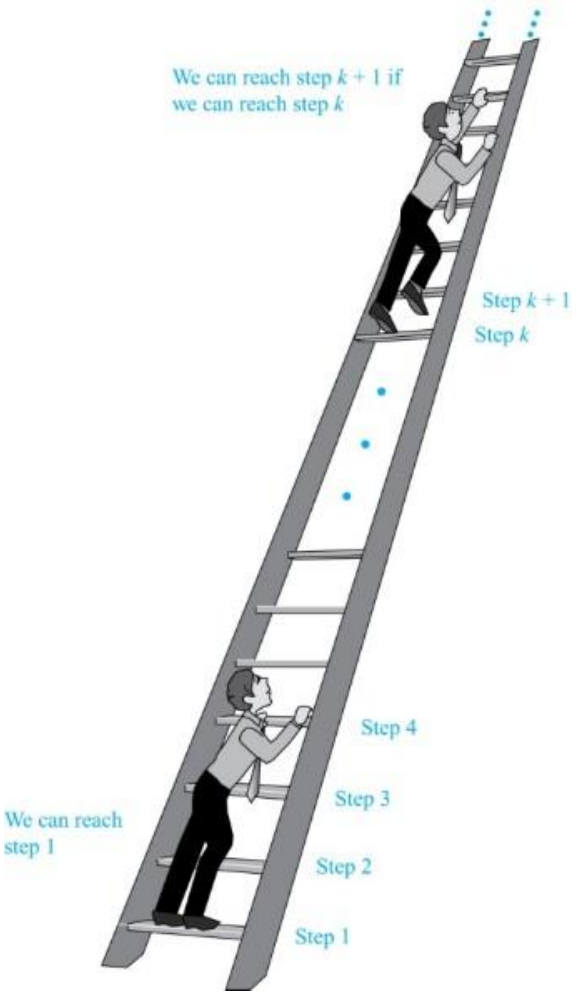
Strong induction tells us that we can reach all rungs if:

1. We can reach the first rung of the ladder.
2. For every integer k, if we can reach the first k rungs, then we can reach the $(k + 1)$st rung.

To conclude that we can reach every rung by strong induction:

- BASIS STEP: $P(1)$ holds
- INDUCTIVE STEP: Assume $P(1) \wedge P(2) \wedge \cdots \wedge P(k)$ holds for an arbitrary integer k, and show that $P(k + 1)$ must also hold.

We will have then shown by strong induction that for every positive integer n, $P(n)$ holds, i.e., we can reach the nth rung of the ladder.



Proof using Strong Induction 1

Example: Suppose we can reach the first and second rungs of an infinite ladder, and we know that if we can reach a rung, then we can reach two rungs higher. Prove that we can reach every rung. (Try this with mathematical induction.)

Solution: Prove the result using strong induction.

- BASIS STEP: We can reach the first step.
- INDUCTIVE STEP: The inductive hypothesis is that we can reach the first k rungs, for any $k \geq 2$. We can reach the $(k + 1)$st rung since we can reach the $(k - 1)$st rung by the inductive hypothesis.
- Hence, we can reach all rungs of the ladder.

Which Form of Induction Should Be Used?

We can always use strong induction instead of mathematical induction. But there is no reason to use it if it is simpler to use mathematical induction. (*See page* 335 *of text*.)

In fact, the principles of mathematical induction, strong induction, and the well-ordering property are all equivalent. (*Exercises* 41-43)

Sometimes it is clear how to proceed using one of the three methods, but not the other two.

Completion of the proof of the Fundamental Theorem of Arithmetic

Example: Show that if n is an integer greater than 1, then n can be written as the product of primes.

Solution: Let $P(n)$ be the proposition that n can be written as a product of primes.

BASIS STEP: $P(2)$ is true since 2 itself is prime.

INDUCTIVE STEP: The inductive hypothesis is $P(j)$ is true for all integers j with $2 \leq j \leq k$. To show that $P(k + 1)$ must be true under this assumption, two cases need to be considered:

- If $k + 1$ is prime, then $P(k + 1)$ is true.
- Otherwise, $k + 1$ is composite and can be written as the product of two positive integers a and b with $2 \leq a \leq b < k + 1$. By the inductive hypothesis a and b can be written as the product of primes and therefore $k + 1$ can also be written as the product of those primes.

Hence, it has been shown that every integer greater than 1 can be written as the product of primes.

(*uniqueness proved in Section* 4.3)

Proof using Strong Induction 2

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

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

BASIS STEP: $P(12)$, $P(13)$, $P(14)$, and $P(15)$ hold.

- $P(12)$ uses three 4-cent stamps.
- $P(13)$ uses two 4-cent stamps and one 5-cent stamp.
- $P(14)$ uses one 4-cent stamp and two 5-cent stamps.
- $P(15)$ uses three 5-cent stamps.

INDUCTIVE STEP: The inductive hypothesis states that $P(j)$ holds for $12 \leq j \leq k$, where $k \geq 15$. Assuming the inductive hypothesis, it can be shown that $P(k + 1)$ holds.

Using the inductive hypothesis, $P(k - 3)$ holds since $k - 3 \geq 12$. To form postage of $k + 1$ cents, add a 4-cent stamp to the postage for $k - 3$ cents. Hence, $P(n)$ holds for all $n \geq 12$.

Proof of Same Example using Mathematical Induction

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

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

BASIS STEP: Postage of 12 cents can be formed using three 4-cent stamps.

INDUCTIVE STEP: The inductive hypothesis $P(k)$ for any positive integer k is that postage of k cents can be formed using 4-cent and 5-cent stamps. To show $P(k + 1)$ where $k \geq 12$, we consider two cases:

- If at least one 4-cent stamp has been used, then a 4-cent stamp can be replaced with a 5-cent stamp to yield a total of k + 1 cents.
- Otherwise, no 4-cent stamp have been used and at least three 5-cent stamps were used. Three 5-cent stamps can be replaced by four 4-cent stamps to yield a total of k + 1 cents.

Hence, $P(n)$ holds for all $n \geq 12$.

M,NAGA SRAVANI, ASSIST.PROF

Well-Ordering Property[1]

Well-ordering property: Every nonempty set of nonnegative integers has a least element.

The well-ordering property is one of the axioms of the positive integers listed in Appendix 1.

The well-ordering property can be used directly in proofs, as the next example illustrates.

The well-ordering property can be generalized.

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

- **N** is well ordered under ≤.
- The set of finite strings over an alphabet using lexicographic ordering is well ordered.

We will see a generalization of induction to sets other than the integers in the next section.

Well-Ordering Property 2

Example: Use the well-ordering property to prove the division algorithm, which states that if a is an integer and d is 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. The set is nonempty since $-dq$ can be made as large as needed.

- By the well-ordering property, S has a least element $r = a - dq_0$. The integer r is nonnegative. It also must be the case that $r < d$. If it were not, then there would be a smaller nonnegative element in S, namely, $a - d(q_0 + 1) = a - dq_0 - d = r - d > 0$.
- Therefore, there are integers q and r with $0 \leq r < d$.

(*uniqueness of q and r is Exercise* 37)

Recursive Definitions and Structural Induction

M,NAGA SRAVANI, ASSIST.PROF

Recursively Defined Functions[1]

Definition: A *recursive* or *inductive definition* of a function consists of two steps.

- 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.

A function $f(n)$ is the same as a sequence $a_0, a_1, \ldots,$ where a_i, where $f(i) = a_i$. This was done using recurrence relations in Section 2.4.

Recursively Defined Functions 2

Example: Suppose f is defined by:

$$f(0)=3,$$

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

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

Solution:

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

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

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

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

Example: Give a recursive definition of the factorial function $n!$:

Solution:

$$f(0)=1$$

$$f(n+1)=(n+1)\cdot f(n)$$

Recursively Defined Functions3

Example: Give a recursive definition of:

$$\sum_{k=0}^{n} a_k .$$

Solution: The first part of the definition is

$$\sum_{k=0}^{0} a_k = a_0 .$$

The second part is $$\sum_{k=0}^{n+1} a_k = \left(\sum_{k=0}^{n} a_k \right) + a_n + 1.$$

Fibonacci Numbers 1

Fibonacci
(1170- 1250)

Example : The Fibonacci numbers are defined as follows:

$$f_0 = 0$$

$$f_1 = 1$$

$$f_n = f_{n-1} + f_{n-2}$$

Find f_2, f_3, f_4, f_5.

$$f_2 = f_1 + f_0 = 1 + 0 = 1$$

$$f_3 = f_2 + f_1 = 1 + 1 = 2$$

$$f_4 = f_3 + f_2 = 2 + 1 = 3$$

$$f_5 = f_4 + f_3 = 3 + 2 = 5$$

Fibonacci Numbers 2

Example 4: Show that whenever $n \geq 3$, $f_n > \alpha^{n-2}$, where $\alpha = \left(1+\sqrt{5}\right)/2$.

Solution: Let $P(n)$ be the statement $f_n > \alpha^{n-2}$.

Use strong induction to show that $P(n)$ is true whenever $n \geq 3$.

BASIS STEP: $P(3)$ holds since $\alpha < 2 = f_3$

$P(4)$ holds since $\alpha^2 = \left(3+\sqrt{5}\right)/2 < 3 = f_4$.

INDUCTIVE STEP: Assume that $P(j)$ holds, i.e., $f_j > \alpha^{j-2}$ for all integers j with $3 \leq j \leq k$, where $k \geq 4$. Show that $P(k+1)$ holds, i.e., $f_{k+1} > \alpha^{k-1}$.

- Since $\alpha^2 = \alpha + 1$ (because α is a solution of $x^2 - x - 1 = 0$),

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

- By the inductive hypothesis, because $k \geq 4$ we have

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

- Therefore, it follows that

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

Why does this equality hold?

- Hence, $P(k+1)$ is true.

Lamé's Theorem [1]

Gabriel Lamé
(1795-1870)

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

Proof: When we use the Euclidian algorithm to find gcd(a,b) with $a \geq b$,

- n divisions are used to obtain (with $a = r_0, b = r_1$):

$$r_0 = r_1 q_1 + r_2 \qquad 0 \leq r_2 < r_1,$$
$$r_1 = r_2 q_2 + r_3 \qquad 0 \leq r_3 < r_2,$$
$$\vdots$$
$$r_{n-2} = r_{n-1} q_{n-1} + r_n \qquad 0 \leq r_n < r_{n-1},$$
$$r_{n-1} = r_n q_n.$$

- Since each quotient $q_1, q_2, \ldots, q_{n-1}$ is at least 1 and $q_n \geq 2$:

$$r_n \geq 1 = f_2,$$
$$r_{n-1} \geq 2 r_n \geq 2 f_2 = f_3,$$
$$r_{n-2} \geq r_{n-1} + r_n \geq f_3 + f_2 = f_4,$$
$$\vdots$$
$$r_2 \geq r_3 + r_4 \geq f_{n-1} + f_{n-2} = f_n,$$
$$b = r_1 \geq r_2 + r_3 \geq f_n + f_{n-1} = f_{n+1}.$$

Lamé's Theorem 2

It follows that if n divisions are used by the Euclidian algorithm to find gcd(a,b) with $a \geq b$, then $b \geq f_{n+1}$. By Example 4, $f_{n+1} > \alpha^{n-1}$, for $n > 2$, where $\alpha = \left(1 + \sqrt{5}\right)/2$. Therefore, $b > \alpha^{n-1}$.

Because $\log_{10} 0.208 \; 1/5$, $\log_{10} b > (n-1)\log_{10} \alpha > (n-1)/5$. Hence,

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

Suppose that b has k decimal digits. Then $b < 10^k$ and $\log_{10} b < k$. It follows that $n - 1 < 5k$ and since k is an integer, $n \leq 5k$.

As a consequence of Lamé's Theorem, $O(\log b)$ divisions are used by the Euclidian algorithm to find gcd(a,b) whenever $a > b$.

- By Lamé's Theorem, the number of divisions needed to find gcd(a,b) with $a > b$ is less than or equal to $5\left(\log_{10} b + 1\right)$ since the number of decimal digits in b (which equals $\lfloor \log_{10} b \rfloor + 1$ is less than or equal to $\log_{10} b + 1$.

Lamé's Theorem was the first result in computational complexity.

Recursively Defined Sets and Structures 1

Recursive definitions of sets have two parts:

- The *basis step* specifies an initial collection of elements.
- The *recursive step* gives the rules for forming new elements in the set from those already known to be in the set.

Sometimes the recursive definition has an *exclusion rule*, which specifies that the set contains nothing other than those elements specified in the basis step and generated by applications of the rules in the recursive step.

We will always assume that the exclusion rule holds, even if it is not explicitly mentioned.

We will later develop a form of induction, called *structural induction*, to prove results about recursively defined sets.

Recursively Defined Sets and Structures 2

Example : Subset of Integers S:

BASIS STEP: $3 \in S$.

RECURSIVE STEP: If $x \in S$ and $y \in S$, then $x + y$ is in S.

Initially 3 is in S, then 3 + 3 = 6, then 3 + 6 = 9, etc.

Example: The natural numbers **N**.

BASIS STEP: $0 \in \mathbf{N}$.

RECURSIVE STEP: If n is in **N**, then $n + 1$ is in **N**.

Initially 0 is in S, then 0 + 1 = 1, then 1 + 1 = 2, etc.

Strings

Definition: The set Σ^* of *strings* over the alphabet Σ:

BASIS STEP: $\lambda \in \Sigma^*$ (λ is the empty string)

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

Example: If $\Sigma = \{0,1\}$, the strings in in Σ^* are the set of all bit strings, λ,0,1, 00,01,10, 11, etc.

Example: If $\Sigma = \{a,b\}$, show that aab is in Σ^*.

- Since $\lambda \in \Sigma^*$ and $a \in \Sigma$, $a \in \Sigma^*$.
- Since $a \in \Sigma^*$ and $a \in \Sigma$, $aa \in \Sigma^*$.
- Since $aa \in \Sigma^*$ and $b \in \Sigma$, $aab \in \Sigma^*$.

String Concatenation

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

BASIS STEP: If $w \in \Sigma^*$, then $w \cdot \lambda = w$.

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

Often $w_1 \cdot w_2$ is written as $w_1 w_2$.

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

Length of a String

Example: Give a recursive definition of $l(w)$, the length of the string w.

Solution: The length of a string can be recursively defined by:

$$l(\lambda) = 0;$$

$$l(wx) = l(w) + 1 \text{ if } w \in \Sigma^* \text{ and } x \in \Sigma.$$

Balanced Parentheses

Example: Give a recursive definition of the set of balanced parentheses P.

Solution:

BASIS STEP: $() \in P$

RECURSIVE STEP: If $w \in P$, then $()w \in P$, $(w) \in P$ and $w() \in P$.

Show that $((())())$ is in P.

Why is $))(()$ not in P?

Well-Formed Formulae in Propositional Logic

Definition: The set of *well-formed formulae* in propositional logic involving **T**, **F**, propositional variables, and operators from the set $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$.

BASIS STEP: **T**,**F**, and *s*, where *s* is a propositional variable, are well-formed formulae.

RECURSIVE STEP: If *E* and *F* are well formed formulae, then $(\neg E)$, $(E \wedge F)$, $(E \vee F)$, $(E \rightarrow F)$, $(E \leftrightarrow F)$, are well-formed formulae.

Examples: $\big((p \vee q) \rightarrow (q \wedge \mathbf{F})\big)$ is a well-formed formula.

$pq \wedge$ is not a well formed formula.

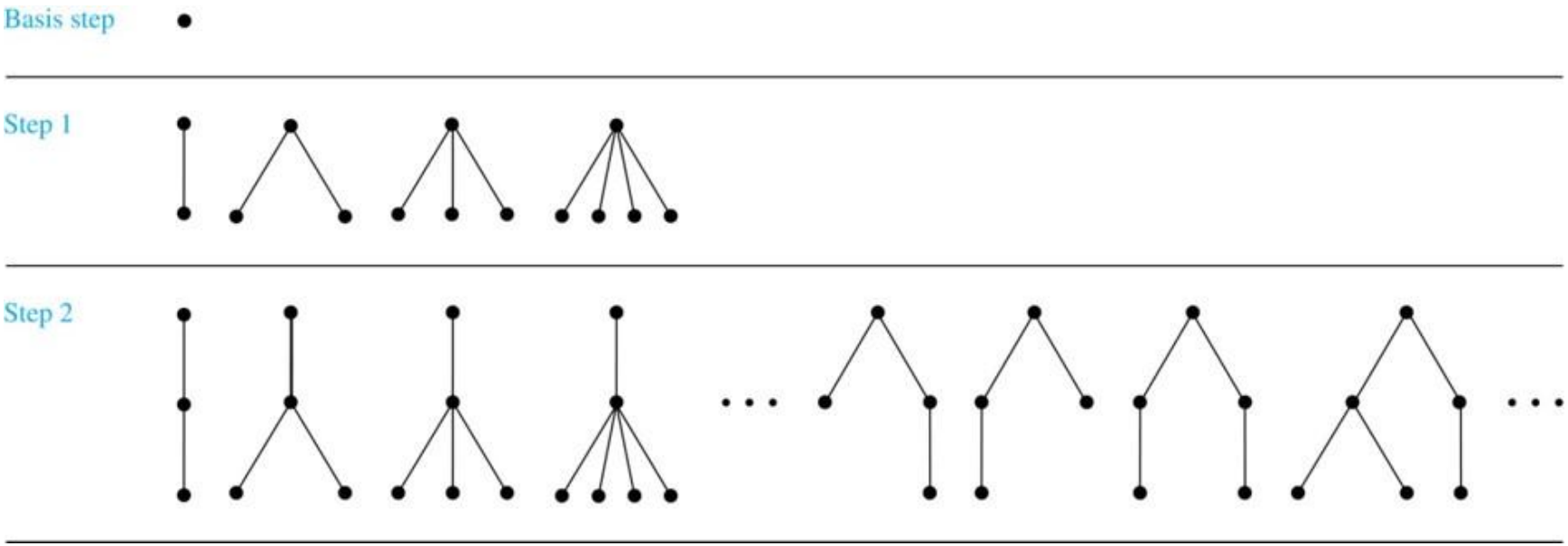
Rooted Trees

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

Building Up Rooted Trees



Trees are studied extensively in Chapter 11.

Next we look at a special type of tree, the full binary tree.

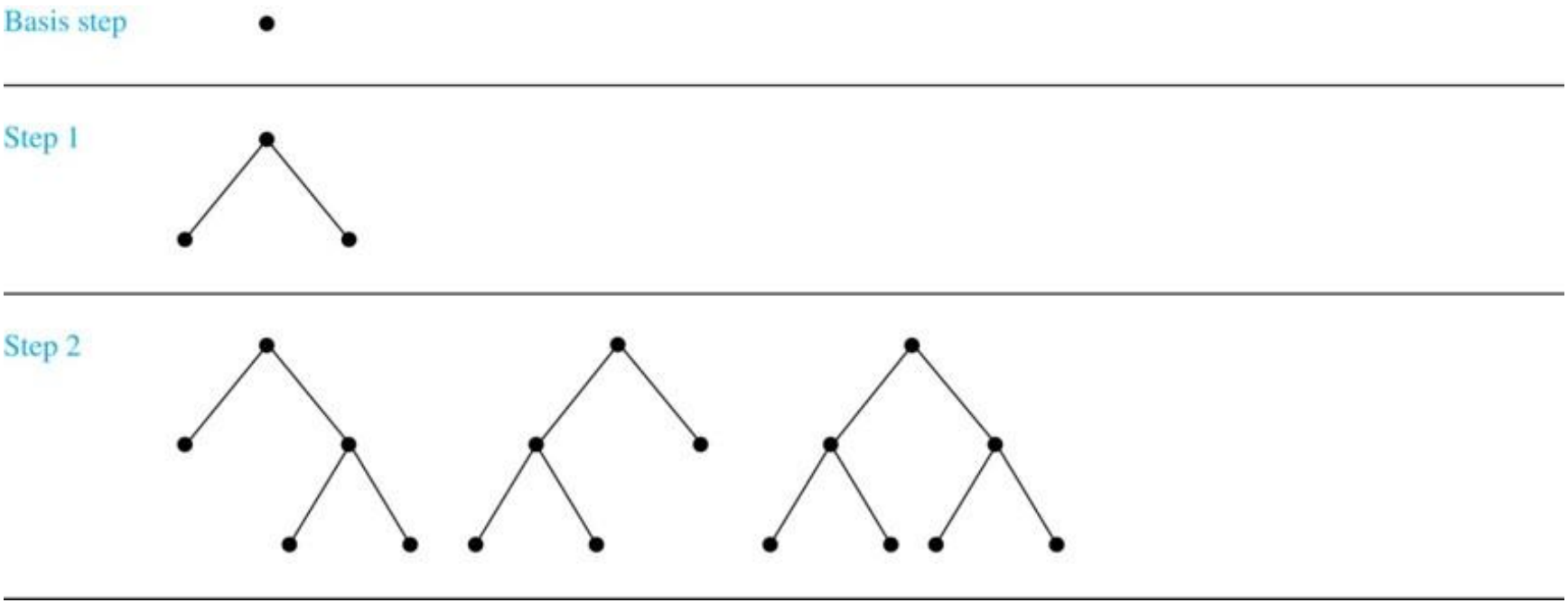
Full Binary Trees 1

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

BASIS STEP: There is a full binary tree consisting of only 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



Induction and Recursively Defined Sets

Example: Show that the set S defined by specifying that $3 \in S$ and that if $x \in S$ and $y \in S$, then $x + y$ is 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 $A = S$, show that A is a subset of S and S is a subset of A.

- $A \subset S$: Let P(n) be the statement that $3n$ belongs to S.

  BASIS STEP: $3 \cdot 1 = 3 \in S$, by the first part of recursive definition.

  INDUCTIVE STEP: Assume $P(k)$ is true. By the second part of the recursive definition, if $3k \in S$, then since $3 \in S$, $3k + 3 = 3(k + 1) \in S$. Hence, $P(k + 1)$ is true.

- $S \subset A$:

  BASIS STEP: $3 \in S$ by the first part of recursive definition, and $3 = 3 \cdot 1$.

  INDUCTIVE STEP: The second part of the recursive definition adds $x + y$ to S, if both x and y are in S. If x and y are both in A, then both x and y are divisible by 3. By part (i) of Theorem 1 of Section 4.1, it follows that $x + y$ is divisible by 3.

We used mathematical induction to prove a result about a recursively defined set. Next we study a more direct form induction for proving results about recursively defined sets.

Structural Induction

Definition: To prove a property of the elements of a recursively defined set, we use *structural induction*.

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

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 can be shown to follow from the principle of mathematical induction.

Full Binary Trees 2

Definition: The *height h(T)* of a full binary tree *T* is defined recursively as follows:

- BASIS STEP: The height of a 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 $h(T) = 1 + \max(h(T_1), h(T_2))$.

The number of vertices *n*(*T*) of a full binary tree *T* satisfies the following recursive formula:

- **BASIS STEP**: The number of vertices of a 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 full binary tree $T = T_1 \cdot T_2$ has the number of vertices $n(T) = 1 + n(T_1) + n(T_2)$.

Structural Induction and Binary Trees

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

Proof: Use structural induction.

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

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

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

Generalized Induction 1

Generalized induction is used to prove results about sets other than the integers that have the well-ordering property. (*explored in more detail in Chapter* 9)

For example, consider an ordering on **N × N**, ordered pairs of nonnegative integers. Specify that (x_1, y_1) is less than or equal to (x_2, y_2) if either $x_1 < x_2$, or $x_1 = x_2$ and $y_1 < y_2$. This is called the *lexicographic ordering*.

Strings are also commonly ordered by a *lexicographic ordering*.

The next example uses generalized induction to prove a result about ordered pairs from **N × N**.

Generalized Induction 2

Example: Suppose that $a_{m,n}$ is defined for $(m,n) \in \mathbf{N} \times \mathbf{N}$ by $a_{0,0} = 0$ and

$$a_{m,n} = \begin{cases} a_{m-1,n} + 1 & \text{if } n = 0 \text{ and } m > 0 \\ a_{m,n-1} + n & \text{if } n > 0 \end{cases}.$$

Show that $a_{m,n} = m + n(n+1)/2$ is defined for all $(m,n) \in \mathbf{N} \times \mathbf{N}$.

Solution: Use generalized induction.

BASIS STEP: $a_{0,0} = 0 + (0.1)/2$

INDUCTIVE STEP: Assume that $a_{m',n'} = m' + n'(n'+1)/2$

whenever(m',n') is less than (m,n) in the lexicographic ordering of $\mathbf{N} \times \mathbf{N}$.

- If $n = 0$, by the inductive hypothesis we can conclude

$$a_{m,n} = a_{m-1,n} + 1 = m - 1 + n(n+1)/2 + 1 = m + n(n+1)/2.$$

- If n > 0, by the inductive hypothesis we can conclude

$$a_{m,n} = a_{m-1,n} + 1 = m + n(n-1)/2 + n = m + n(n+1)/2.$$

Recursive Algorithms

M,NAGA SRAVANI, ASSIST.PROF

Recursive Algorithms

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.

Recursive Factorial Algorithm

Example: Give a recursive algorithm for computing $n!$, where n is a nonnegative integer.

Solution: Use the recursive definition of the factorial function.

procedure *factorial*(n: nonnegative integer)

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

else return $n \cdot factorial(n-1)$

{output is $n!$}

Recursive Exponentiation Algorithm

Example: Give a recursive algorithm for computing a^n, where a is a nonzero real number and n is a nonnegative integer.

Solution: Use the recursive definition of a^n.

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

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

else return $a \cdot power\left(a, n-1\right)$

$\{$output is $a^n\}$

Recursive GCD Algorithm

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

Solution: Use the reduction

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

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

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 Modular Exponentiation Algorithm

Example: Devise a a recursive algorithm for computing b^n **mod** m, where b, n, *and* m are integers with $m \geq 2$, $n \geq 0$, and $1 \leq b \leq m$.

Solution: (*see text for full explanation*)

procedure *mpower*(b,m,n: integers with $b > 0$ and $m \geq 2$, $n \geq 0$)
if $n = 0$ **then**
    return 1
else if *n is even* **then**
    return *mpower*

$$(b, n/2, m)^2 \textbf{ mod } m$$

else
    return $\left(mpower\left(b, \lfloor n/2 \rfloor, m\right)^2 \textbf{ mod } m \cdot b \textbf{ mod } m\right) \textbf{ mod } m$

{output is b^n **mod** m}

Recursive Binary Search Algorithm

Example: Construct a recursive version of a binary search algorithm.

Solution: Assume we have $a_1, a_2, \ldots, a_n$, an increasing sequence of integers. Initially *i* is 1 and *j* is *n*. We are searching for *x*.

procedure *binary search*(*i, j, x :* integers, $1 \le i \le j \le n$)
$m := \lfloor (i + j)/2 \rfloor$
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 $a_1, a_2, \ldots, a_n$ if it appears, otherwise 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 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

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

Merge Sort 1

Merge Sort works by iteratively splitting a list (with an even number of elements) into two sublists of equal length until each sublist has one element.

Each sublist is represented by a balanced binary tree.

At each step a pair of sublists is successively merged into a list with the elements in increasing order. The process ends when all the sublists have been merged.

The succession of merged lists is represented by a binary tree.

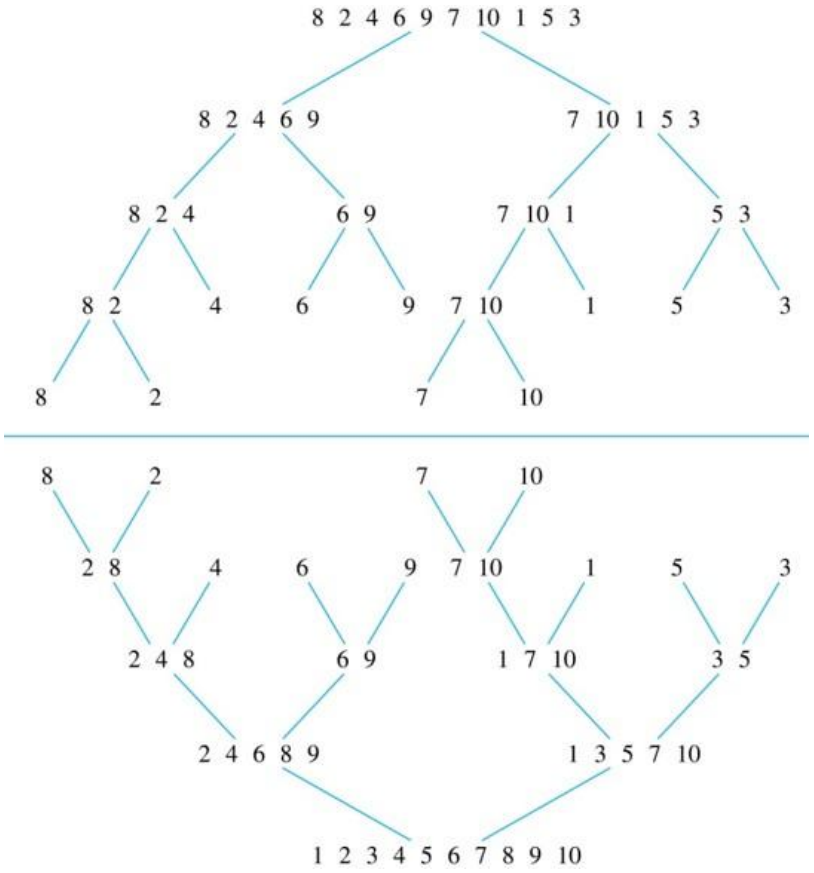
Merge Sort 2

Example: Use merge sort to put the list

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

into increasing order.

Solution:



Recursive Merge Sort 1

Example: Construct a recursive merge sort algorithm.

Solution: Begin with the list of n elements L.

procedure $mergesort(L = a_1, a_2, \ldots, a_n)$

if $n > 1$ **then**

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

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

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

$L := merge(mergesort(L_1), mergesort(L_2))$

{L is now sorted into elements in increasing order}

Recursive Merge Sort 2

Subroutine *merge*, which merges two sorted 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 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 the elements in increasing order}

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

Merging Two Lists

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.

First List	*Second List*	*Merged List*	*Comparison*
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	

Complexity of Merge Sort 1

Complexity of Merge Sort: The number of comparisons needed to merge a list with n elements is $O(n \log n)$.

For simplicity, assume that n is a power of 2, say 2^m.

At the end of the splitting process, we have a binary tree with m levels, and 2^m lists with one element at level m.

The merging process begins at level m with the pairs of 2^m lists with one element combined into 2^{m-1} lists of two elements. Each merger takes two one comparison.

The procedure continues, at each level ($k = m,\ m-1,\ m-1,...,3,2,1$) 2^k lists with 2^{m-k} elements are merged into 2^{k-1} lists, with 2^{m-k+1} elements at level $k-1$.

- We know (by the complexity of the merge subroutine) that each merger takes at most $2^{m-k} + 2^{m-k} - 1 = 2^{m-k+1} - 1$ comparisons.

Complexity of Merge Sort 2

Summing over the number of comparisons at each level, shows that

$$\sum_{k=1}^{m} 2^{k-1}\left(2^{m-k+1}-1\right)=\sum_{k=1}^{m} 2^{m}-\sum_{k=1}^{m} 2^{k-1}=m2^{m}-\left(2^{m}-1\right)=n\log n-n+1,$$

because $m = \log n$ and $n = 2^m$.

(The expression $\sum_{k=1}^{m} 2^{k-1}$ in the formula above is evaluated as $2^m - 1$ using the formula for the sum of the terms of a geometric progression, from Section 2.4.)

In Chapter 11, we'll see that the fastest comparison-based sorting algorithms have $O(n \log n)$ time complexity. So, merge sort achieves the best possible big-O estimate of time complexity.

Accessibility Content: Text Alternatives for Images

M,NAGA SRAVANI, ASSIST.PROF

Climbing an Infinite Ladder – Text Alternative

There is a man climbing an infinite ladder, the steps of which are numbered with natural numbers from the bottom. The man can reach step k plus one if he can reach step k.

Remembering How Mathematical Induction Works – Text Alternative

There are dominoes numbered with natural numbers. The domino with number one falls on the domino with number two, and the domino with number two falls on the domino with number three, etc.

Number of Subsets of a Finite Set $_2$ – Text Alternative

There is field S with circle X inside. S has two arrows. The first one is from S to field T that has circle X and element A inside, the second arrow is from S to field T that has a circle named X union left brace A right brace inside, which has element A inside.

Tiling Checkerboards 1 – Text Alternative

There are four checkerboards of the size 2 times 2 with one square removed each. The first checkerboard does not have the left bottom square. The second checkerboard does not have the right bottom square. The third checkerboard does not have the left top square, and the fourth one does not have the right top square. In each case, the remaining squares form right triomino.

Tiling Checkerboards $_2$ – Text Alternative

There are four squares forming together a large square. The right bottom square has a small shaded square inside.

There are four squares forming together a large square, each one has small shaded square inside. In the left top square, the small square is in the bottom right corner. In the right top square, the small square is in the bottom left corner. In the left bottom square, the small square is in the top right corner. Thus, these three small squares form a right triomino.

Building Up Rooted Trees – Text Alternative

There are 3 steps of building up rooted trees shown. Basic step contains one vertex which is a root. The first step contains several vertices that are added to the next level. All these vertices are connected to the root. At the second step, the vertices of the previous level are the roots for the added vertices of the next level.

Building Up Full Binary Trees – Text Alternative

There are three steps of building up full binary trees shown. The basic step contains one vertex on the first level which is the root. Each next level is located below the previous one. At the first step, two vertices are added to the next level. They are connected to the root forming the right and left branches. At the second step, the vertices of the second level are the roots for the added vertices of the third level. The right and the left branches can be formed in several ways: only from the left vertex, only from the right, or from both vertices.

Merge Sort 2 – Text Alternative

There is a binary balanced tree at the top. Its root consists of numbers 8, 2, 4, 6, 9, 7, 10, 1, 5, and 3. At the first step, there are two branches, the left leads to the vertex containing elements 8, 2, 4, 6, and 9. The right one leads to the vertex containing elements 7, 10, 1, 5, and 3. Two branches lead from the vertex 8, 2, 4, 6, 9 . The left one leads to the vertex with the elements 8, 2, 4. The right one leads to the vertex with the elements 6 and 9. Two branches also lead from the vertex 7, 10, 1, 5, and 3. The left one leads to the vertex with the elements 7, 10, 1. The right one leads to the vertex with the elements 5 and 3. Each of the four vertices of the previous level has two branches leading to the vertices of the next level: from 8, 2, 4 to 8, 2 and 4, from 6, 9 to 6 and 9, from 7, 10, 1 to 7, 10 and 1, from 5, 3 to 5 and 3. At the next level there are branches from 8, 2 to 8 and 2, and from 7, 10 to 7 and 10. At the bottom of the picture there is a similar tree, but it is turned upside down. In such vertices where there is more than one element, elements are written in the increasing order from left to right.

M,NAGA SRAVANI, ASSIST.PROF