Chapter 5: Induction and Recursive Definitions

Kenneth Rosen, Discrete Mathematics, 8th Edition

August 27, 2025

What is Mathematical Induction?

- A proof technique to establish that a statement $P(n)$ is true for all positive integers n.
- Based on the principle: if a statement holds for a base case and can be shown to hold for the next case, it holds for all cases.
- Two key steps:
  1. **Base Case**: Prove the statement for the smallest value of n (usually $n = 1$ or $n = 0$).
  2. **Inductive Step**: Assume the statement holds for $n = k$ (inductive hypothesis), and prove it for $n = k + 1$.

Principle of Mathematical Induction

Formal Statement

To prove $P(n)$ is true for all $n \geq n_0$:

- **Base Case**: Show $P(n_0)$ is true.
- **Inductive Step**: Assume $P(k)$ is true for some $k \geq n_0$ (inductive hypothesis). Prove $P(k+1)$ is true.

Principle of Mathematical Induction

Formal Statement

To prove $P(n)$ is true for all $n \geq n_0$:

- **Base Case**: Show $P(n_0)$ is true.
- **Inductive Step**: Assume $P(k)$ is true for some $k \geq n_0$ (inductive hypothesis). Prove $P(k+1)$ is true.

- If both steps are proven, then $P(n)$ is true for all $n \geq n_0$.
- Analogy: Like climbing a ladder—prove you can start (base case) and move to the next rung (inductive step).

Example: Sum of First n Positive Integers

Statement

Prove: $1 + 2 + \cdots + n = \frac{n(n+1)}{2}$ for all $n \geq 1$.

Example: Sum of First n Positive Integers

Statement

Prove: $1 + 2 + \cdots + n = \frac{n(n+1)}{2}$ for all $n \geq 1$.

- **Base Case** $(n = 1)$:

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

  Holds true.

- **Inductive Step**: Assume true for $n = k$:

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

  Prove for $n = k + 1$:

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

  Holds true.

Strong Induction

- A variation of mathematical induction.
- **Base Case**: Same as standard induction (prove for smallest n).
- **Inductive Step**: Assume $P(m)$ is true for all $m \leq k$, and prove $P(k+1)$.
- Useful when proving $P(k+1)$ requires more than just $P(k)$.

Strong Induction

- A variation of mathematical induction.
- **Base Case**: Same as standard induction (prove for smallest n).
- **Inductive Step**: Assume $P(m)$ is true for all $m \leq k$, and prove $P(k+1)$.
- Useful when proving $P(k+1)$ requires more than just $P(k)$.

Example Use Case

May sometimes make proofs easier.

Summary

- Mathematical induction proves statements for all positive integers.
- Requires a base case and an inductive step.
- Strong induction allows assuming all previous cases in the inductive step.
- Applications: Summations, inequalities, divisibility, and recursive definitions.

Key Takeaway

Induction is a powerful tool for proving statements about infinite sets by reducing them to finite steps.

Why Multiple Base Cases?

- Some proofs require more than one base case, especially in:
  - **Strong Induction**: Inductive step may depend on multiple previous cases.
  - **Recursive Definitions**: Statements defined using several prior terms (e.g., Fibonacci: $F(n) = F(n-1) + F(n-2)$).
  - Problems where the smallest n needs extra cases to establish the pattern.
- Multiple base cases ensure the inductive step can "reach back" to valid cases.

Why Multiple Base Cases?

- Some proofs require more than one base case, especially in:
  - **Strong Induction**: Inductive step may depend on multiple previous cases.
  - **Recursive Definitions**: Statements defined using several prior terms (e.g., Fibonacci: $F(n) = F(n-1) + F(n-2)$).
  - Problems where the smallest n needs extra cases to establish the pattern.
- Multiple base cases ensure the inductive step can "reach back" to valid cases.

Key Idea

The number of base cases depends on how many prior values the inductive step needs.

Example: Fibonacci Numbers Are Positive

Statement

Prove: Fibonacci numbers $F(n)$, defined by $F(1) = 1$, $F(2) = 1$, $F(n) = F(n-1) + F(n-2)$ for $n \geq 3$, are positive for all $n \geq 1$.

Example: Fibonacci Numbers Are Positive

Statement

Prove: Fibonacci numbers $F(n)$, defined by $F(1) = 1$, $F(2) = 1$, $F(n) = F(n-1) + F(n-2)$ for $n \geq 3$, are positive for all $n \geq 1$.

- **Base Cases**:
  - $n = 1$: $F(1) = 1 > 0$.
  - $n = 2$: $F(2) = 1 > 0$.
- **Inductive Step (Strong Induction)**:
  - Assume $F(m) > 0$ for all $m \leq k$, where $k \geq 2$.
  - Prove $F(k+1)$: Since $F(k+1) = F(k) + F(k-1)$, and $F(k) > 0$, $F(k-1) > 0$ by hypothesis, so $F(k+1) > 0$.
- Two base cases needed because $F(k+1)$ depends on $F(k)$ and $F(k-1)$.

Example: All Horses Are the Same Color

Claim (Flawed)

All horses in any set of n horses are the same color.

Example: All Horses Are the Same Color

Claim (Flawed)

All horses in any set of n horses are the same color.

- **Base Case**: For $n = 1$, one horse has its own color. True.
- **Inductive Step**: Assume true for $n = k$: any set of k horses is the same color.
- Prove for $n = k + 1$:
  - Take a set of $k + 1$ horses: $\{h_1, h_2, \ldots, h_{k+1}\}$.
  - Subset $\{h_1, \ldots, h_k\}$ has k horses, so all are the same color (by hypothesis).
  - Subset $\{h_2, \ldots, h_{k+1}\}$ has k horses, so all are the same color.
  - Since h_2 is in both subsets, all $k + 1$ horses are the same color.

Why the Horse Proof Fails

- The inductive step fails for $n = 2$ (base case is $n = 1$):
  - For $k = 1$, consider $\{h_1, h_2\}$.
  - Subset $\{h_1\}$ has one horse, $\{h_2\}$ has one horse—both trivially true.
  - But no overlap exists (no common horse), so we cannot conclude h_1 and h_2 are the same color.
- **Lesson**: The inductive step must hold for all cases, including the transition from base case to the next step.
- **Fix**: Adding a base case for $n = 2$ reveals the flaw, as two horses may differ in color.

What is Strong Induction?

- A variation of mathematical induction for proving statements $P(n)$ for all positive integers n.
- Differs from standard induction in the inductive step:
  - **Base Case**: Prove $P(n)$ for the smallest value(s) (e.g., $n = 1$ or multiple base cases).
  - **Inductive Step**: Assume $P(m)$ is true for all $m \leq k$, and prove $P(k + 1)$.
- Useful when $P(k + 1)$ depends on multiple previous cases, not just $P(k)$.

Principle of Strong Induction

Formal Statement

To prove $P(n)$ is true for all $n \geq n_0$:

- **Base Case**: Show $P(n_0), P(n_0+1), \ldots$ for necessary starting values.
- **Inductive Step**: Assume $P(m)$ is true for all m where $n_0 \leq m \leq k$. Prove $P(k+1)$.

Principle of Strong Induction

Formal Statement

To prove $P(n)$ is true for all $n \geq n_0$:

- **Base Case**: Show $P(n_0), P(n_0 + 1), \ldots$ for necessary starting values.
- **Inductive Step**: Assume $P(m)$ is true for all m where $n_0 \leq m \leq k$. Prove $P(k + 1)$.

- If both steps hold, $P(n)$ is true for all $n \geq n_0$.
- Analogy: Like climbing a ladder, but you can use all previous rungs to reach the next one.

Example: Divisibility of Numbers

Statement

Prove: Every integer $n \geq 12$ can be written as $n = 3a + 7b$ for non-negative integers a, b.

Example: Divisibility of Numbers

Statement

Prove: Every integer $n \geq 12$ can be written as $n = 3a + 7b$ for non-negative integers a, b.

- **Base Cases**:
  - $n = 12$: $12 = 3 \cdot 4 + 7 \cdot 0$.
  - $n = 13$: $13 = 3 \cdot 2 + 7 \cdot 1$.
  - $n = 14$: $14 = 3 \cdot 0 + 7 \cdot 2$.
- **Inductive Step**: Assume true for all m where $12 \leq m \leq k$. Prove for $k + 1$:
  - Since $k + 1 \geq 15$, consider $m = k + 1 - 3 = k - 2$. Since $k - 2 \geq 12$, $k - 2 = 3a + 7b$.
  - Then, $k + 1 = (k - 2) + 3 = 3a + 7b + 3 = 3(a + 1) + 7b$.

Well-Ordering Property

Definition

Every non-empty set of positive integers has a least element.

Well-Ordering Property

Definition

Every non-empty set of positive integers has a least element.

- Key principle in proofs, especially for strong induction and contradiction.
- Example use: Prove a property by assuming a counterexample exists, then showing the smallest counterexample leads to a contradiction.
- Connection to induction: Well-ordering ensures the "smallest" case exists, supporting the base case and inductive reasoning.

Example: Fundamental Theorem of Arithmetic

Statement

Every integer $n > 1$ can be expressed as a product of primes (unique up to order).

Example: Fundamental Theorem of Arithmetic

Statement

Every integer $n > 1$ can be expressed as a product of primes (unique up to order).

- Proof by well-ordering (sketch):
  - Assume the set S of integers $n > 1$ with no prime factorization is non-empty.
  - By well-ordering, S has a least element m.
  - If m is prime, it is its own factorization (contradiction).
  - If m is composite, $m = a \cdot b$ where $1 < a, b < m$. Since $a, b \notin S$, they have prime factorizations, so m does (contradiction).
  - Thus, S is empty, and every $n > 1$ has a prime factorization.

Summary

- **Strong Induction**: Assumes all cases $n_0 \leq m \leq k$ to prove $P(k+1)$.
- **Well-Ordering Property**: Every non-empty set of positive integers has a least element.
- Applications: Recursive sequences, divisibility, prime factorization, and algorithm correctness.
- Strong induction is more flexible than standard induction for complex dependencies.

Key Takeaway

Strong induction and well-ordering provide powerful tools for proving statements about integers, especially when multiple prior cases are needed.

Definitions

- **Weak Induction (Standard Induction)**:
  - Base Case: Prove $P(n_0)$ is true.
  - Inductive Step: Assume $P(k)$ is true, prove $P(k+1)$.
- **Strong Induction**:
  - Base Case: Prove $P(n_0), P(n_0+1), \ldots$ for necessary starting values.
  - Inductive Step: Assume $P(m)$ for all $m \leq k$, prove $P(k+1)$.
- **Well-Ordering Principle**:
  - Every non-empty set of positive integers has a least element.

Equivalence Overview

- Weak induction, strong induction, and well-ordering are logically equivalent.
- Proof strategy:
  1. Show weak induction implies strong induction.
  2. Show strong induction implies well-ordering.
  3. Show well-ordering implies weak induction.
- Equivalence means any one can be used as a foundation for proofs about integers.

Equivalence Overview

- Weak induction, strong induction, and well-ordering are logically equivalent.
- Proof strategy:
  1. Show weak induction implies strong induction.
  2. Show strong induction implies well-ordering.
  3. Show well-ordering implies weak induction.
- Equivalence means any one can be used as a foundation for proofs about integers.

Key Idea

Each principle allows us to prove statements $P(n)$ for all positive integers n, but they approach it differently.

Recall

Weak Induction:

- Premise, $Weak_p$: $P(n_0)$ and For any k, $P(k) \to P(k+1)$
- Conclusion, $Weak_c$: $\forall n : (n > n_0) \to P(n)$.

Strong Induction:

- Premise, $Strong_p$: $P(n_0)$ and For any k, $P(1) \wedge \ldots \wedge \P(k) \to P(k+1)$
- Conclusion, $Strong_c$: $\forall n : (n > n_0) \to P(n)$.

Well Ordering Principle

- Statement, *WOP*: Under the usual meaning $\leq$. Any subset $\mathbb{N}$ has a least element (with respect to $\leq$).

Weak Induction $\implies$ Strong Induction

Proof Sketch

We now prove the rule of inference corresponding to strong induction using the principle of weak induction. That is, $Strong_p \rightarrow Strong_c$ is a tautology. In other words, if we assume $Strong_p$, $Weak_p \rightarrow Weak_c$ is true, then $Strong_c$ is true.

- We now construct a proposition $P'(n)$ which is true if and only if $n \geq n_0$ and $P(n_0), P(n_0+1), \ldots P(n)$ is true
- $P'(n_0)$ is true as $P(n_0)$ is true (by $Strong_p$).
- Suppose $P'(k)$ is true for some arbitrary k. This implies that (A) $P(n_0) \wedge P(n_0+1) \wedge \ldots \wedge P(n)$ is true .
- By $Strong_p$ (A) implies (B) $P(k+1)$ is true.
- By weak induction, the above implies $\forall(n > n_0) : P(n)$, which is the same as $Strong_c$.

Weak Induction $\implies$ Strong Induction

Proof Sketch

We now prove the rule of inference corresponding to strong induction using the principle of weak induction. That is, $Strong_p \rightarrow Strong_c$ is a tautology. In other words, if we assume $Strong_p$, $Weak_p \rightarrow Weak_c$ is true, then $Strong_c$ is true.

- We now construct a proposition $P'(n)$ which is true if and only if $n \geq n_0$ and $P(n_0), P(n_0+1), \ldots P(n)$ is true
- $P'(n_0)$ is true as $P(n_0)$ is true (by $Strong_p$).
- Suppose $P'(k)$ is true for some arbitrary k. This implies that (A) $P(n_0) \wedge P(n_0+1) \wedge \ldots \wedge P(n)$ is true .
- By $Strong_p$ (A) implies (B) $P(k+1)$ is true.
- By weak induction, the above implies $\forall (n > n_0) : P(n)$, which is the same as $Strong_c$.

-

Strong Induction $\implies$ Well-Ordering

We need to prove that $(Strong_p \to Strong_c) \to WOP$. We prove this by contradiction. That is, we show that is, if WOP is false and $(Strong_p \to Strong_c)$ is true, then we can derive a contradiction.

Proof Sketch (by Contradiction)

Assume well-ordering is false: there exists a non-empty set S of positive integers with no least element.

- Define $P(n)$: "No integer $m \leq n$ is in S."
- **Base Case**: Prove $P(1)$ (1 is not in S, as S has no least element).
- **Inductive Step**: Assume $P(m)$ for all $m \leq k$ (no integer $\leq k$ is in S). Prove $P(k+1)$ (i.e., $k+1 \notin S$).
- If $k+1 \in S$, it would be the least element (since no $m \leq k$ is in S), a contradiction.

Strong Induction $\implies$ Well-Ordering

We need to prove that $(Strong_p \rightarrow Strong_c) \rightarrow WOP$. We prove this by contradiction. That is, we show that is, if WOP is false and $(Strong_p \rightarrow Strong_c)$ is true, then we can derive a contradiction.

Proof Sketch (by Contradiction)

Assume well-ordering is false: there exists a non-empty set S of positive integers with no least element.

- Define $P(n)$: "No integer $m \leq n$ is in S."
- **Base Case**: Prove $P(1)$ (1 is not in S, as S has no least element).
- **Inductive Step**: Assume $P(m)$ for all $m \leq k$ (no integer $\leq k$ is in S). Prove $P(k+1)$ (i.e., $k+1 \notin S$).
- If $k+1 \in S$, it would be the least element (since no $m \leq k$ is in S), a contradiction.

- $P(n)$ true for all n implies S is empty, proving well-ordering.

Well-Ordering $\implies$ Weak Induction

Given $Weak_p$ and WOP, we show that $Weak_c$ is true.

Proof Sketch

To prove $Weak_p : P(n)$ for all $n \geq n_0$ using well-ordering:

- Suppose $Weak_p$ is false. That is, $P(n)$ is false for some $n \geq n_0$. Let S be the set of all $n \geq n_0$ where $P(n)$ is false.
- By well-ordering, S has a least element m.
- **Base Case**: $m \neq n_0$, since $P(n_0)$ is true.
- Since m is the least element, $P(m-1)$ is true (for $m - 1 \geq n_0$).
- **Inductive Step**: Use $P(m-1)$ to prove $P(m)$, contradicting $m \in S$.

Well-Ordering $\implies$ Weak Induction

Given $Weak_p$ and WOP, we show that $Weak_c$ is true.

Proof Sketch

To prove $Weak_p : P(n)$ for all $n \geq n_0$ using well-ordering:

- Suppose $Weak_p$ is false. That is, $P(n)$ is false for some $n \geq n_0$. Let S be the set of all $n \geq n_0$ where $P(n)$ is false.
- By well-ordering, S has a least element m.
- **Base Case**: $m \neq n_0$, since $P(n_0)$ is true.
- Since m is the least element, $P(m-1)$ is true (for $m - 1 \geq n_0$).
- **Inductive Step**: Use $P(m-1)$ to prove $P(m)$, contradicting $m \in S$.

- Thus, S is empty, so $P(n)$ is true for all $n \geq n_0$.

Summary

- **Weak Induction**: Uses $P(k)$ to prove $P(k+1)$.
- **Strong Induction**: Uses $P(m)$ for all $m \leq k$ to prove $P(k+1)$.
- **Well-Ordering**: Every non-empty set of positive integers has a least element.
- All three are equivalent: proving one implies the others.
- Practical use: Choose the principle that best fits the proof structure (e.g., strong induction for recursive cases, well-ordering for contradiction).

Key Takeaway

These principles form the foundation for proving statements about integers, offering flexible approaches to the same logical truth.

On which sets can you apply Induction

What were the properties of positive integers that we used for proving induction works?

On which sets can you apply Induction

What were the properties of positive integers that we used for proving induction works?

- Positive integers are totally ordered. There exists some order $\preceq$ amongst the integers such that for any two distinct integers n_1 and n_2, we can always decide whether $n_1 \preceq n_2$ or not.

On which sets can you apply Induction

What were the properties of positive integers that we used for proving induction works?

- Positive integers are totally ordered. There exists some order $\preceq$ amongst the integers such that for any two distinct integers n_1 and n_2, we can always decide whether $n_1 \preceq n_2$ or not.
- Well-Ordering Property - Every non-empty subset of S has a **least element**.

On which sets can you apply Induction

What were the properties of positive integers that we used for proving induction works?

- Positive integers are totally ordered. There exists some order $\preceq$ amongst the integers such that for any two distinct integers n_1 and n_2, we can always decide whether $n_1 \preceq n_2$ or not.
- Well-Ordering Property - Every non-empty subset of S has a **least element**.
- Notice that we any set with the above proof of equivalence between strong induction, weak induction, and well-ordering principle will work for any set which satisfies the above properties.

Well-Ordered Sets.

- What is a Well-Ordered Set?
- Given some ordering $\prec$ on S, a set S along with a total order $\prec$ $(S; \prec)$ is **well-ordered** if every non-empty subset of S has a **least element**.
- **Total Order**: Given some ordering $\prec$ For any $a, b \in S$, exactly one of $a \prec b$, $a = b$, or $b \prec a$ holds.
- **Least Element**: An element $m \in S$ such that $m = x$ or $m \prec x$ for all x in a subset of S.

Well-Ordered Sets.

- What is a Well-Ordered Set?
- Given some ordering $\prec$ on S, a set S along with a total order $\prec$ $(S; \prec)$ is **well-ordered** if every non-empty subset of S has a **least element**.
- **Total Order**: Given some ordering $\prec$ For any $a, b \in S$, exactly one of $a \prec b$, $a = b$, or $b \prec a$ holds.
- **Least Element**: An element $m \in S$ such that $m = x$ or $m \prec x$ for all x in a subset of S.

Example

The positive integers $\mathbb{N} = \{1, 2, 3, \dots\}$ with the usual order $\leq$ are well-ordered (every non-empty subset has a smallest element).

Well-Ordered Sets.

- What is a Well-Ordered Set?
- Given some ordering $\prec$ on S, a set S along with a total order $\prec$ $(S; \prec)$ is **well-ordered** if every non-empty subset of S has a **least element**.
- **Total Order**: Given some ordering $\prec$ For any $a, b \in S$, exactly one of $a \prec b$, $a = b$, or $b \prec a$ holds.
- **Least Element**: An element $m \in S$ such that $m = x$ or $m \prec x$ for all x in a subset of S.

Example

The positive integers $\mathbb{N} = \{1, 2, 3, \dots\}$ with the usual order $\leq$ are well-ordered (every non-empty subset has a smallest element).
What about $(\mathbb{N}; \geq)$.

Well-Ordered Sets.

- What is a Well-Ordered Set?
- Given some ordering $\prec$ on S, a set S along with a total order $\prec$ $(S;\prec)$ is **well-ordered** if every non-empty subset of S has a **least element**.
- **Total Order**: Given some ordering $\prec$ For any $a, b \in S$, exactly one of $a \prec b$, $a = b$, or $b \prec a$ holds.
- **Least Element**: An element $m \in S$ such that $m = x$ or $m \prec x$ for all x in a subset of S.

Example

The positive integers $\mathbb{N} = \{1, 2, 3, \dots\}$ with the usual order $\leq$ are well-ordered (every non-empty subset has a smallest element). What about $(\mathbb{N}; \geq)$. What about negative integers?

Well-Ordered Sets.

- What is a Well-Ordered Set?
- Given some ordering $\prec$ on S, a set S along with a total order $\prec$ $(S; \prec)$ is **well-ordered** if every non-empty subset of S has a **least element**.
- **Total Order**: Given some ordering $\prec$ For any $a, b \in S$, exactly one of $a \prec b$, $a = b$, or $b \prec a$ holds.
- **Least Element**: An element $m \in S$ such that $m = x$ or $m \prec x$ for all x in a subset of S.

Example

The positive integers $\mathbb{N} = \{1, 2, 3, \ldots\}$ with the usual order $\leq$ are well-ordered (every non-empty subset has a smallest element). What about $(\mathbb{N}; \geq)$. What about negative integers? Is it possible to "make" them "Well-Ordered"?

Well-Ordered Sets.

- What is a Well-Ordered Set?
- Given some ordering $\prec$ on S, a set S along with a total order $\prec$ $(S; \prec)$ is **well-ordered** if every non-empty subset of S has a **least element**.
- **Total Order**: Given some ordering $\prec$ For any $a, b \in S$, exactly one of $a \prec b$, $a = b$, or $b \prec a$ holds.
- **Least Element**: An element $m \in S$ such that $m = x$ or $m \prec x$ for all x in a subset of S.

Example

The positive integers $\mathbb{N} = \{1, 2, 3, \dots\}$ with the usual order $\leq$ are well-ordered (every non-empty subset has a smallest element). What about $(\mathbb{N}; \geq)$. What about negative integers? Is it possible to "make" them "Well-Ordered"?What about rational numbers with $\leq$?

Well-Ordered Sets.

- What is a Well-Ordered Set?
- Given some ordering $\prec$ on S, a set S along with a total order $\prec$ $(S; \prec)$ is **well-ordered** if every non-empty subset of S has a **least element**.
- **Total Order**: Given some ordering $\prec$ For any $a, b \in S$, exactly one of $a \prec b$, $a = b$, or $b \prec a$ holds.
- **Least Element**: An element $m \in S$ such that $m = x$ or $m \prec x$ for all x in a subset of S.

Example

The positive integers $\mathbb{N} = \{1, 2, 3, \dots\}$ with the usual order $\leq$ are well-ordered (every non-empty subset has a smallest element). What about $(\mathbb{N}; \geq)$. What about negative integers? Is it possible to "make" them "Well-Ordered"?What about rational numbers with $\leq$? What about real numbers $\leq$?

Section 5.3: What are Recursive Definitions?

- A **recursive definition** defines an object in terms of itself.
- Two parts:
  - **Base Case**: Initial elements or values.
  - **Recursive Step**: Rules to construct new elements from existing ones.
- Used for:
  - Sequences (e.g., Fibonacci numbers).
  - Sets (e.g., well-formed formulas).
  - Functions (e.g., factorial).

Section 5.3: What are Recursive Definitions?

- A **recursive definition** defines an object in terms of itself.
- Two parts:
  - **Base Case**: Initial elements or values.
  - **Recursive Step**: Rules to construct new elements from existing ones.
- Used for:
  - Sequences (e.g., Fibonacci numbers).
  - Sets (e.g., well-formed formulas).
  - Functions (e.g., factorial).

Example

Factorial: $n! = \begin{cases} 1 & \text{if } n = 0, \\ n \cdot (n-1)! & \text{if } n \geq 1. \end{cases}$

Examples of Recursive Definitions

- **Sequence**: Fibonacci numbers
  - Base: $F(0) = 0$, $F(1) = 1$.
  - Recursive: $F(n) = F(n-1) + F(n-2)$ for $n \geq 2$.

Examples of Recursive Definitions

- **Sequence**: Fibonacci numbers
    - Base: $F(0) = 0$, $F(1) = 1$.
    - Recursive: $F(n) = F(n-1) + F(n-2)$ for $n \geq 2$.
- **Set**: Strings over alphabet $\{0, 1\}$
    - Base: Empty string ϵ is a string.
    - Recursive: If w is a string, then $w0$ and $w1$ are strings.

Examples of Recursive Definitions

- **Sequence**: Fibonacci numbers
  - Base: $F(0) = 0$, $F(1) = 1$.
  - Recursive: $F(n) = F(n-1) + F(n-2)$ for $n \geq 2$.
- **Set**: Strings over alphabet $\{0, 1\}$
  - Base: Empty string ϵ is a string.
  - Recursive: If w is a string, then $w0$ and $w1$ are strings.
- **Function**: Ackermann's function
  - $A(m, n) = \begin{cases} n+1 & \text{if } m = 0, \\ A(m-1, 1) & \text{if } m > 0, n = 0, \\ A(m-1, A(m, n-1)) & \text{if } m > 0, n > 0. \end{cases}$

What is Structural Induction?

- A proof technique for recursively defined objects (sets, sequences, trees, etc.).
- Similar to strong induction, but tailored to the structure of the definition.
- Steps:
  - **Base Case**: Prove the property holds for base elements.
  - **Inductive Step**: Assume the property holds for all elements used in the recursive step, then prove it for the new element.
- Useful for proving properties of recursive sets or structures.

Key Idea

Follow the recursive definition to ensure the property holds for all elements.

Example: Full Binary Trees

- A **full binary tree** is a tree where every node is either a leaf or has exactly two children (left and right).
- **Recursive Definition**:
  - **Base Case**: A single node (leaf) is a full binary tree.
  - **Recursive Step**: If T_1 and T_2 are full binary trees, a new tree T can be formed with a root node and T_1 as left child, T_2 as right child.
- Notation:
  - $L(T)$: Number of leaves in tree T.
  - $N(T)$: Total number of nodes in tree T.

Example: Theorem - Number of Nodes in a Full Binary Tree

Statement

For any full binary tree T, the number of nodes $N(T)$ satisfies:

$$N(T) = 2L(T) - 1,$$

where $L(T)$ is the number of leaves in T.

Example: Theorem - Number of Nodes in a Full Binary Tree

Statement

For any full binary tree T, the number of nodes $N(T)$ satisfies:

$$N(T) = 2L(T) - 1,$$

where $L(T)$ is the number of leaves in T.

- **Why Interesting?**
  - Relates leaves to total nodes in a recursive structure.
  - Useful in computer science (e.g., analyzing binary tree algorithms).

Structural Induction for the Proof

- We use **structural induction** to prove the theorem.
- Follows the recursive definition of full binary trees:
    - **Base Case**: Prove for the simplest tree (single node).
    - **Inductive Step**: Assume the property holds for subtrees T_1 and T_2, prove for a tree T with T_1 and T_2 as children.
- Goal: Show $N(T) = 2L(T) - 1$ for all full binary trees.

Base Case

Single Node (Leaf)

Consider a full binary tree T with a single node.

Base Case

Single Node (Leaf)

Consider a full binary tree T with a single node.

- Number of leaves: $L(T) = 1$ (the node is a leaf).
- Number of nodes: $N(T) = 1$ (only one node).
- Check:
$$2L(T) - 1 = 2 \cdot 1 - 1 = 1 = N(T).$$
- The property holds for the base case.

Inductive Step

- **Inductive Hypothesis**: Assume for full binary trees T_1 and T_2:
$$N(T_1) = 2L(T_1) - 1, \quad N(T_2) = 2L(T_2) - 1.$$
- Consider a tree T with root and children T_1 (left) and T_2 (right).
- Compute:
  - Leaves: $L(T) = L(T_1) + L(T_2)$.
  - Nodes: $N(T) = 1 + N(T_1) + N(T_2)$ (1 for the root).

Inductive Step

- **Inductive Hypothesis**: Assume for full binary trees T_1 and T_2:
$$N(T_1) = 2L(T_1) - 1, \quad N(T_2) = 2L(T_2) - 1.$$
- Consider a tree T with root and children T_1 (left) and T_2 (right).
- Compute:
  - Leaves: $L(T) = L(T_1) + L(T_2)$.
  - Nodes: $N(T) = 1 + N(T_1) + N(T_2)$ (1 for the root).
- Substitute hypothesis:
$$N(T) = 1 + (2L(T_1) - 1) + (2L(T_2) - 1).$$
$$= 1+2L(T_1)-1+2L(T_2)-1 = 2(L(T_1)+L(T_2))-1 = 2L(T)-1.$$
- The property holds for T.

Conclusion and Applications

- **Conclusion**: By structural induction, $N(T) = 2L(T) - 1$ holds for all full binary trees.
- **Applications**:
  - Analyzing binary tree structures in computer science (e.g., binary search trees, expression trees).
  - Understanding node-leaf relationships in algorithms and data structures.
- **Why Interesting?**:
  - Demonstrates structural induction on a recursive, hierarchical structure.
  - Connects mathematical proof to practical applications in computing.