

Chapter 4. Induction and Recursion

4.1 Mathematical Induction

Example: to show that we are able to *reach every step* on an infinite ladder

we do the proof by showing

(1) we can *reach the first step* of the ladder

(2) if we can *reach a particular step*, we can *reach the next step*

Rephrase it as

goal of the proof: $\forall n R(n)$

we just need to prove:

(1) $R(1)$

(2) $\forall k R(k) \rightarrow R(k + 1)$

Another example would be *Dominoes*.

Principle of Mathematical Induction:

To prove a certain property $P(n)$ holds for all positive integers $n \geq 1$, we need to do:

(1) **base step:** prove $P(1)$ to be true;

(2) **inductive step:** prove $\forall k(P(k) \rightarrow P(k+1))$ to be true.

This principle can be expressed as

$$[P(1) \wedge \forall k(P(k) \rightarrow P(k+1))] \rightarrow \forall nP(n)$$

Examples of proofs using math induction.

A. Prove that summation for all $n \geq 1$,

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

B. Prove that summation for all $n \geq 0$,

$$1 + 2 + 2^2 + \dots + 2^n = 2^{n+1} - 1$$

C. Prove that inequality, for all $n \geq 1$, (how about $n \geq 0$?)

$$n < 2^n$$

It suffices to show the basic step and the induction step.

More examples:

D. Prove that $2^n < n!$ for $n \geq 4$ (why $n \geq 4$?)

E. Prove that there are 2^n subsets of any finite set that contains n elements, for any $n \geq 0$.

[How to do induction?]

F. Prove that

$$\overline{\bigcap_{j=1}^n A_j} = \bigcup_{j=1}^n \overline{A_j}$$

G. Odd Pie Fights

Axiom: The well-ordering property of integers

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

Use this property to show that the math induction approach works correctly. I.e., after (1) base step and (2) inductive step are proved, $\forall n P(n)$.

Proof by contradiction:

Assume there are positive numbers whose property P is not proved true. Let S be the set of such numbers. Based on the well-ordering property, there is a least element $m \in S$. $P(m)$ is false.

$m \neq 1$ because $P(1)$ is proved true by (1).

$P(m - 1)$ is proved true since $m - 1 \notin S$. But by (2) $P(m - 1) \rightarrow P(m)$. So $P(m)$ is true, which contradicts to that $P(m)$ is false. So our assumption was wrong.

4.2 Strong Mathematical Induction

Example: For all $n \geq 2$, n can be expressed as the product of prime numbers.

Abbreviate n can be expressed as the product of prime numbers as $P(n)$

Proof:

(1) base step: $n = 2$, $P(n)$ is true.

(2) inductive step: assume $P(k)$ to be true

consider $k + 1$:

(a) if $k + 1$ is prime, then $P(k + 1)$ is true.

(b) if $k + 1$ is not prime, then $k + 1$ is a product of two integers. Let $k + 1 = a \times b$.

Now we are NOT able to use the inductive assumption $P(k)$ since a, b may be smaller than k .

So the inductive assumption needs to be stronger

Principle of Strong Mathematical Induction:

To prove a certain property $P(n)$ holds for all positive integers $n \geq 1$, we need to do:

(1) **base step:** prove $P(1)$ to be true;

(2) **inductive step:** prove

$$\forall k(P(1) \wedge P(2) \wedge \dots \wedge P(k) \rightarrow P(k + 1))$$

to be true.

This principle can be expressed as

$$[P(1) \wedge \forall k(P(1) \wedge P(2) \wedge \dots \wedge P(k) \rightarrow P(k + 1))] \rightarrow \forall n P(n)$$

Examples:

A. Previous example of expressing every $n \geq 2$ by product of primes.

B. Game of Two Piles of Matches

C. Every postage of 12 cents or more can be formed using just 4-cent and 5-cent stamps.

4.3 Recursive Definition and Structural Induction

Sometimes we have to describe a new concept or an object using the concept/object itself. This is a **recursive** process.

Recursively define sequences, sets, functions.

Recrusively defined sequences:

$$\text{e.g., } a_1 = 1, \quad a_n = 2a_{n-1}$$

defines sequence $\{1, 2, 4, 8, \dots\}$

We can use *the mathematical induction* to prove results about the recursively defined sequence. e.g., prove $a_n = 2^{n-1}$ using math induction.

Recrusively defined sets:

We can recursively define what elements should belong to a set. e.g.

To prove results about a recursively defined set, we can use a new method called *structural induction*.

- **A sequence is one-dimensional, characterized with positive integers**
- **A set is multi-dimensional, characterized with structures.**

We begin with

Recursively defined functions:

Two steps to recursively define a integer function

Basis Step: specify the value of the function at zero

Recursive Step: give a rule for finding its value at an integer from its values at smaller integers.

Example:

A.

$$f(0) = 3, \quad f(n + 1) = 2f(n) + 3$$

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

B.

$$f(0) = 1, \quad f(n + 1) = (n + 1)f(n) \text{ defines } n!.$$

C. Recursive definition for $f(n) = a^n$

D. Fibonacci number $f(n)$, $f(1) = f(2) = 1$

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

Use math induction to show: $f(n) > \alpha^{n-2}$
where $\alpha = (1 + \sqrt{5})/2$ for $n \geq 3$.

(note that $P(4)$ is also the base case.)

Recursively defined sets and structures

Examples:

Consider S defined as:

(1) basis step: $3 \in S$

(2) recursive step: if $x \in S$ and $y \in S$, then $x + y \in S$.

What does the set S contain?

Example:

define *well-formed boolean formulae* (WFF)

(1) T , F , and any proposition s are WFF;

(2) if A and B are WFF, then $(\neg A)$, $(A \vee B)$, $(A \wedge B)$, $(A \rightarrow B)$, and $(A \leftrightarrow B)$ are WFF.

Example, defining the set CUBE that contains all n -cubes, $n \geq 0$.

(1) basis step: a single point, labeled with λ , belongs to CUBE;

(2) recursive step: if $x \in \text{CUBE}$, then let x_0 and x_1 are copies of x . Add prefix 0 to the labels of all points in x_0 and add prefix 1 to the labels of all points in x_1 . Then the resulting object by linking every two points of the same label except the first bit in x_0 and x_1 belongs to CUBE.

Some properties of the CUBE that can be proved by structural induction:

- the number of points for any object in CUBE is a power of 2.
- each point in any object in CUBE is linked to exactly $\log_2 m$ other points in the object, where m is the total number of points in the object.
- for any object in CUBE, if it contains m points, then the points are uniquely labeled with different $\log_2 m$ -bit binary numbers.

Example, defining set E of arithmetic expressions

(1) basis step:

– (a) an integer belongs to E ;

– (b) a variable belongs to E ;

(2) recursive step:

– (a) if $e_1, e_2 \in E$, then $e_1 + e_2 \in E$;

– (b) if $e_1, e_2 \in E$, then $e_1 - e_2 \in E$;

– (c) if $e_1, e_2 \in E$, then $e_1 * e_2 \in E$;

how to add parentheses (precedence)?

Example, defining set St that contains all legal statements in certain language (e.g., PASCAL);

(1) basis step: string ' $v = e$ ' $\in St$, where v is a name, e is an arithmetic expression;

(2) recursive step:

(a) if string $s_1 \in St$, string $s_2 \in St$, the string ' $s_1 ; s_2$ ' $\in St$;

(b) if string $s_1 \in St$, string $s_2 \in St$, and b is a boolean expression, then

string '**if b then s_1 else s_2** ' $\in St$;

(c) if string $s_1 \in St$ and b is an boolean expression, then '**while b do s** ' $\in St$;

Example, can you define a set of recursive objects?

Example, defining binary trees:

Structural Induction

When a set or structure is *recursively* defined, we can prove certain properties of the set or the structure by induction that follows the two steps of the recursive definition.

Similar to the mathematical and strong mathematical inductions, *structural induction* uses the same pattern of two steps.

(1) base step: show the property holds for the scenario defined in the basis step of the recursive definition;

(2) inductive step: assume the property holds for all the substructures used to define the structure in the recursive step of the recursive definition, show the property holds for the defined structure.

example, recursive definition for set S :

(1) basis step: $3 \in S$;

(2) recursive step: if $x, y \in S$, then $x + y \in S$.

Using *structural induction* to prove that for any $z \in S$, $3|z$.

proof:

(1) basis step: $z = 3$, $3|3$, so

(2) inductive step: $z = x + y$.

Assume $3|x$ and $3|y$, then $x = 3a$ and $y = 3b$ for some integers a, b . So $z = x + y = 3a + 3b = 3(a + b) = 3c$ where $c = a + b$ integer. That is $3|z$.

Consider the following definition of the set B of boolean expressions. Here is the recursive definition for B :

(1) basis step: $T, F \in B$;

(2) recursive step: if $b_1, b_2 \in B$, then $\neg b_1, (b_1 \wedge b_2), (b_1 \vee b_2) \in B$.

Show that, using structural induction, every expression has value either T or F .

(1) base step: $b = T$ or $b = F$, in either case, the claim is true.

(2) inductive step:

(a) $b = \neg b_1$. Assume b_1 has value T , then b has value F and vice versa.

(b) $b = (b_1 \vee b_2)$. Assume b_1, b_2 both have value either T or F , then we can construct a truth table to show b has value either T or F .

(c) $b = (b_1 \wedge b_2)$. Assume b_1, b_2 both have value either T or F , then we can construct a truth table to show b has value either T or F .

4.4 Recursive Algorithms

Recursive algorithms are not most efficient;

Recursive algorithms can be easier to design;