Unit 4

Induction and Recursion

4.1 Induction

A proof technique used to prove a property P(n) for integers n greater than or equal to a fixed integer n0

4.1.1 Proof by mathematical induction

Suppose we have to show that $P(n)$ is true for all integers n greater that equal to n_0 i.e., we have to show that $\forall n P(n)$ is true where the universe of discourse of n is the set of all integers greater than or equal to n_0. We can use the technique of mathematical induction for proving such assertions.

The proof by mathematical induction proceeds in two steps:

Basis step: Show that $P(n_0)$ is true.

Inductive step: Show that $P(k) \rightarrow P(k+1)$ is true for any integer $k \geq n_0$.

The truth of basis and inductive steps implies the truthness of $P(n_0)$, $P(n_0+1)$, $P(n_0+2), \cdots$ as follows. The truthness of $P(n_0)$ is shown in the basis step. The truthness of inductive step for $k = n_0$ implies that $P(n_0) \rightarrow P(n_0+1)$ is also true and hence by modus ponens, $P(n_0+1)$ must be true. Again taking $k = n_0 + 1$ in the inductive step we get the truthness of $P(n_0+1) \rightarrow P(n_0+2)$ and together with the truthness of $P(n_0+1)$ we conclude that $P(n_0+2)$ is also true. Continuing in this fashion, we can show that $P(k)$ is true for all $k \geq n_0$.

Example:

1. Show that $1 + 2 + \cdots + n = \dfrac{n(n+1)}{2}$ for all $n \geq 1$.

   Proof: We have to prove that

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

   for all $n \geq 1$.
   Basis step: We have to show that the equality is true for $n = 1$. When $n = 1$, we have

   $$LHS = 1$$

and

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

so that $LHS = RHS$. Hence the equality is true for $n = 1$.
Inductive step: Suppose that the equality is true when $n = k$ i.e.,

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

We now need to prove that the equality is true when $n = k + 1$ i.e.,

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

Now,

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

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

Therefore inductive step is true as well.
Hence $1 + 2 + \cdots + n = \frac{n(n+1)}{2}$ for all $n \geq 1$. □

2. Prove that $n^3 - n$ is divisible by 3 for all $n \geq 1$.

Proof: We need to prove that $n^3 - n$ is divisible by 3 for all $n \geq 1$ i.e., $n^3 - n = 3m$ for some integer m.
Basis Step: We show that $n^3 - n = 3m$ for some integer m when $n = 1$. Now when $n = 1$, we have

$$n^3 - n = 1^3 - 1 = 1 - 1 = 0 = 3 \times 0.$$

Hence $n^3 - n = 3m$ when $n = 1$.
Inductive Step: Suppose that $n^3 - n = 3m$ for some integer m when $n = k$ i.e., $k^3 - k = 3m$. We now need to prove that $n^3 - n$ is again a multiple of 3 when $n = k + 1$ i.e., $(k+1)^3 - (k+1)$ is a multiple of 3. Now

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

$$= 3m + 3(k^2 + k) = 3(m + k^2 + k).$$

So $(k+1)^3 - (k+1)$ is a multiple of 3. Therefore inductive step is true as well.
Hence $n^3 - n$ is divisible by 3 for all $n \geq 1$. □

4.1.2 Proof by strong/complete induction

To prove that $P(n)$ is true for all integers $n \geq n_0$, we can use the technique of proof by strong induction or complete induction that proceeds in two steps as follows:

Basis step: Show that $P(n_0)$ is true.

Inductive step: Show that $[P(n_0) \wedge P(n_0+1) \wedge \cdots \wedge P(k)] \rightarrow P(k+1)$ is true for any integer $k \geq n_0$.

Comparing this with the proof by (normal) induction, we can see that the only difference is in the inductive step: unlike normal induction, in strong induction one assumes the truthness of not only $P(n_0)$ but also of $P(n_0+1), \cdots, P(k)$ and uses these to prove the truthness of $P(k+1)$. However, it can be proved that both the proof techniques are equivalent i.e., anything that can be proved using strong induction can also be proved using normal induction and vice-versa.

Example:

1. A Fibonacci sequence $\{F_n\}_{n=0}^{\infty}$ is a sequence defined recursively as follows:

$$F_0 = 0,\ F_1 = 1 \text{ and } F_{n+1} = F_n + F_{n-1} \text{ for } n \geq 2.$$

So the Fibonacci sequence is $0, 1, 1, 2, 3, 5, 8, 13, \cdots$.
Show that for each integer $n \geq 0$,

$$F_n < \left(\frac{7}{4}\right)^n.$$

Proof: We need to show that

$$F_n < \left(\frac{7}{4}\right)^n$$

for all $n \geq 0$.
Basis step: We need to show that the above inequality is true when $n = 0$ and $n = 1$ i.e.,

$$F_0 < \left(\frac{7}{4}\right)^0 \text{ and } F_1 < \left(\frac{7}{4}\right)^1.$$

Since $F_0 = 0$ and $\left(\frac{7}{4}\right)^0 = 1$, so it is true when $n = 0$. Since $F_1 = 1$ and $1 < \left(\frac{7}{4}\right)^1$, so it is true when $n = 1$ as well.

Inductive step: We now show that if the inequality is true when $n = 0, 1, \cdots, k$, then it is true for $n = k+1$ as well i.e., if $F_0 < \left(\frac{7}{4}\right)^0$, $F_1 < \left(\frac{7}{4}\right)^1$, $\cdots$, $F_k < \left(\frac{7}{4}\right)^k$, then $F_{k+1} < \left(\frac{7}{4}\right)^{k+1}$. Now,

$$F_{k+1} = F_k + F_{k-1} < \left(\frac{7}{4}\right)^k + \left(\frac{7}{4}\right)^{k-1} = \left(\frac{7}{4}\right)^{k-1}\left(\frac{7}{4}+1\right)$$

$$= \left(\frac{7}{4}\right)^{k-1}\left(\frac{11}{4}\right) < \left(\frac{7}{4}\right)^{k-1}\left(\frac{7}{4}\right)^2 = \left(\frac{7}{4}\right)^{k+1},$$

i.e., $F_{k+1} < \left(\frac{7}{4}\right)^{k+1}$.
Therefore inductive step is true as well. Hence $F_n < \left(\frac{7}{4}\right)^n$ for any integer $n \geq 0$. □

2. Show that if $n \geq 2$ is an integer, then n can be written as the product of primes.

   Proof: We need to prove that any integer $n \geq 2$ can be written as a product of prime numbers.

   Basis step: We shall show that $n = 2$ can be written as a product of prime numbers. But this is obviously true because 2 is itself a prime number.

   Inductive step: Suppose that all the integers $2, 3, \ldots, k$ where $k \geq 2$, can be written as a product of prime numbers. Then we need to show that $k + 1$ can also be written as a product of prime numbers.
   Consider the case when $k + 1$ is itself a prime number. Then our assertion is obviously true. However if $k + 1$ is a composite number then $k + 1$ can be written as a product of two integers a and b such that $2 \leq a \leq b \leq k + 1$. By inductive hypothesis, both a and b can be written as a product of primes. Therefore $k + 1$ itself can be written as a product of primes, namely those primes in the factorization of a and those in the factorization of b.
   □

4.1.3 Well-Ordering Property

Well-ordering property is one of the fundamental axioms of the set of integers. It states that

> "Every nonempty set of nonnegative integers has a least element."

i.e., if $A \subseteq \{0, 1, 2, 3, \cdots\}$ and $A \neq \emptyset$, then A must have a least element. This property can be used to prove the validity of the principle of mathematical induction as well as the principle of strong induction. In fact, one can show that the well-ordering property is equivalent to both these principles of induction.

Example: Use the well-ordering property to prove the division algorithm: "If a is an integer and d is a positive integer, then there are unique integers q and r with $0 \leq r < d$ and $a = dq+r$."

Proof: Let

$$S = \{a - dm : \ a - dm \geq 0 \text{ and } m \text{ is an integer}\}.$$

Then S is a set of nonnegative integers. Also, S is nonempty because by taking m a negative integer with large absolute value, we can make $-dm$ as large as we like thus making $a-dm \geq 0$. Therefore by the well-ordering property, S has a least element, say $r = a - dq$. Since $r \in S$, so $r \geq 0$. Also, $r < d$ because if $r \geq d$ then $a - d(q + 1) = a - dq - d = r - d \geq 0$ so $a - d(q + 1) \in S$. But $a - d(q + 1) = r - d < r$ which contradicts that r is the least element in S. So $0 \leq r < d$.

To prove the uniqueness of q and r, suppose that there is another pair q_1 and r_1 with $0 \leq r_1 < d$ and $a = dq_1 + r_1$. First we prove that $r_1 = r$. For if not, let $r < r_1$ so that $0 < r - r_1 < d$ because $0 \leq r < d$ and $0 \leq r_1 < d$. But

$$r_1 - r = a - dq_1 - (a - dq) = d(q - q_1)$$

i.e., $d \mid (r_1 - r)$ which is impossible because $0 < r - r_1 < d$. So $r \geq r_1$. Similarly, we can show that $r_1 \geq r$ and therefore $r = r_1$. Then $q = q_1$ as well.
□

4.2 Recursion

The process of defining objects such as functions, sets or sequences in terms of itself is called recursion. When such objects are defined using recursion, we say that they are recursively defined.

4.2.1 Recursively-Defined Functions

If a function f is defined for all integers $n \geq n_0$ in terms of itself, then it is called a recursively-defined function and such definition of f is called a recursive definition or inductive definition.

We use the following two steps to give a recursive definition of function f:

> **Basis Step:** Specify the value of f at n_0.
> **Recursive Step:** Give a rule for finding its value at an integer $n+1$ from its values at smaller integer.

Examples:

1. Let f be a function defined for all integers $n \geq 0$ as follows:

   > **Basis step:** $f(0) = 3$
   > **Recursive step:** $f(n+1) = 2f(n) + 3$.

   Then f is a recursively defined function. Its functional values for integers can be calculated as follows:

   $$f(1) = 2f(0) + 3 = 2 \times 3 + 3 = 9$$
   $$f(2) = 2f(1) + 3 = 2 \times 9 + 3 = 21$$
   $$f(3) = 2f(2) + 3 = 2 \times 21 + 3 = 45$$

   and so on.

2. Let f be defined for all integers $n \geq 0$ as follows:

   > **Basis step:** $f(0) = 1$
   > **Recursive step:** $f(n+1) = (n+1)f(n)$.

   Then f is a recursive definition of the factorial function $f(n) = n!$:

   $$f(1) = 1 \times f(0) = 1 \times 1 = 1 = 1!$$
   $$f(2) = 2 \times f(1) = 2 \times 1 = 2!$$
   $$f(3) = 3 \times f(2) = 3 \times 2 \times 1 = 3!$$
   $$f(4) = 4 \times f(3) = 4 \times 3 \times 2 \times 1 = 4!$$

   and so on.

3. While defining a recursive function, basis step may require declaration of functional values at more than one initial point as in example below:

   Basis step: $f(0) = 0, f(1) = 1$
   Recursive step: $f(n+1) = f(n) + f(n-1)$.

   Then

$$f(2) = f(1) + f(0) = 1 + 0 = 1$$
$$f(3) = f(2) + f(1) = 1 + 1 = 2$$
$$f(4) = f(3) + f(2) = 2 + 1 = 3$$
$$f(5) = f(4) + f(3) = 3 + 2 = 5$$

   and so on.

4. Recursive definition of the power function $f(n) = a^n$ where $a \neq 0$ and $n \geq 0$:

   Basis step: $f(0) = 1$
   Recursive step: $f(n+1) = af(n)$.

5. Recursive definition of the summation function $f(m) = \sum_{k=0}^{m} a_k$:

   Basis step: $f(0) = a_0$
   Recursive step: $f(n+1) = f(n) + a_{n+1}$.

4.2.2 Recursively Defined Sets

If a set S is defined in terms of itself, then S is called a recursively defined set and such definition of S is called recursive definition or inductive definition.

We use the following two steps to give a recursive definition of set S:

Basis Step: Specify the elements that initially belongs to the set S.
Recursive Step: Give rules for forming new elements in the set S from those already known to be in S.

Examples:

1. Let S be the set defined recursively as follows:

   Basis Step: $3 \in S$
   Recursive Step: If $x, y \in S$ then $x + y \in S$.

   Then S is the set of all the positive multiples of 3.

2. Let S be the set defined recursively as

Basis Step: $5 \in S$
Recursive Step: If $x, y \in S$, then $x + y \in S$ and $x - y \in S$.

Then S is the set of all the possible integer multiples of 5.

3. Let S be the set of all well-formed formulae for compound propositions formed from T, F, propositional variables p, q, r, s etc and logical operators $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$. Then S can be defined recursively as follows:

   Basis Step: T, F and propositional variables such as p, q, r, s etc are well-defined formulae.
   Recursive Step: If U and V are well-formed formulae then $(\neg U), (U \wedge V), (U \vee V), (U \rightarrow V)$ and $(U \leftrightarrow V)$ are also well-formed formulae.

4.2.3 Structural Induction

To prove results about recursively defined sets, we use a form of mathematical induction which is called structural induction. A proof by structural induction consists of two steps:

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

Example:

1. Show that the set S defined recursively in previous example 1. is the set of all the positive integers that are multiples of 3.

   Proof: We have to prove that if x is any element of the set S, then $x = 3m$ for some positive integer m. We proceed by structural induction.

   Basis step: Since $3 = 3 \times 1$ is a positive integer multiple of 3, so the result holds for the element 3 specified in the basis step of recursive definition.

   Recursive step: Now suppose x, y are some positive integer multiples of 3 i.e., $x = 3m$ and $y = 3n$ for some positive integers m and n. Then

   $$x + y = 3m + 3n = 3(m + n)$$

   is also a positive integer multiple of 3 and so the result holds.

2. Show that every well-formed formulae for compound propositions contains an equal number of left and right parentheses.

   Proof:
   Basis step: Since the formulae T, F, p, q, r, s etc do not have any parentheses, so they have an equal number (i.e. zero number) of left and right parentheses.

Recursive step: Now suppose U and V are well-formed formulae each containing an equal number of left and right parentheses. That is, if L_U, R_U, L_V, R_V are the number of left and right parentheses in U and V respectively then $L_U = R_U$ and $L_V = R_V$. To complete the induction step, we need to show that $(\neg U)$, $(U \wedge V)$, $(U \vee V)$, $(U \rightarrow V)$, $(U \leftrightarrow V)$ also contain an equal number of left and right parentheses. The number of left parentheses in $(\neg U)$ is $L_U + 1$ and in others there are $L_U + L_V + 1$. The number of right parentheses in $(\neg U)$ is $R_U + 1$ and in others there are $R_U + R_V + 1$. Since $L_U = R_U$ and $L_V = R_V$, so $L_U + 1 = R_U + 1$ and $L_U + L_V + 1 = R_U + R_V + 1$. Therefore each of these compound propositions also contain the same number of left and right parentheses.

4.2.4 Recursive Algorithms

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

1. **Recursive algorithm for computing $n!$, $n \geq 0$:**

   The factorial of an integer n, denoted by $n!$, is defined as $n! = 1$ if $n = 0$ and $n! = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n$ if $n \geq 1$. The following is a recursive algorithm to compute $n!$.

   Procedure factorial(n : nonnegative integer)

   IF $n = 0$ THEN factorial(n) $= 1$
   ELSE factorial(n) $= n \cdot$ factorial($n - 1$)

2. **Recursive algorithm for computing a^n, $a \neq 0, n \geq 0$:**

   For a number $a \neq 0$ and an integer $n \geq 0$, we define a^n as $a^n = 0$ if $n = 0$ and $a^{n+1} = a \cdot a^n$ if $n \geq 1$. So the following gives the recursive algorithm to compute a^n.

   Procedure power(a: nonzero real number; n : nonnegative integer)

   IF $n = 0$ THEN power(a, n) $= 1$
   ELSE power(a, n) $= a \cdot$ power(a, $n - 1$)

3. **Recursive algorithm for computing $\gcd(a, b)$:**

   While computing $gcd(a, b)$ where $a < b$ are nonnegative integers, if the integer a is zero, then $gcd(0, b) = b$. Otherwise if $a \neq 0$, then $gcd(a, b)$ is equal to the gcd of a and the remainder obtained when b is divided by a i.e. $gcd(a, b) = gcd(b \mod a, a)$. So we can form a recursive algorithm for computing $gcd(a, b)$ as follows:

   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)

4. **Recursive algorithm for linear search:**

   This algorithm searches for x in the sequence $a_i, a_{i+1}, \ldots, a_j$.

Procedure search(i, j, x : i, j are integers with $1 \leq i, j \leq n$ and x a number)

IF $a_i = x$ THEN location $= i$
ELSE IF $i = j$ THEN location $= 0$
ELSE search($i + 1, j, x$)

5. **Recursive algorithm for modular exponentiation:**

   Given integers b, n and m where $n \geq 0$ and $m \geq 2$, this algorithm computes b^n mod m i.e. the remainder when b^n is divided by m.

   Procedure mpower(b, n, m : integers with $m \geq 2$, $n \geq 0$)

   IF $n = 0$ THEN mpower$(b, n, m) = 1$
   ELSE IF n is even THEN mpower(b, n, m) = mpower$(b, n/2, m)^2$ **mod** m
   ELSE mpower(b, n, m) = (mpower$(b, \lfloor n/2 \rfloor, m)^2$ **mod** $m \cdot b$ **mod** m) **mod** m

Proving correctness of recursive algorithm

1. Prove that the algorithm to compute a^n for $a \neq 0$ and $n \geq 0$ an integer, is correct.

   Solution: We have to prove that power$(a, n) = a^n$ for all $n \geq 0$. For this, we use the process of mathematical induction on the exponent n.

   Basis Step: We prove that power$(a, n) = a^n$ when $n = 0$. Now when $n = 0$, we have power$(a, 0) = 1$ from the algorithm and also $a^0 = 1$ for nonzero a. Therefore

   $$LHS = RHS$$

   when $n = 0$.

   Inductive Step: Suppose that the equality is true when $n = k$ i.e.,

   $$power(a, k) = a^k.$$

   We have to prove that the equality is true when $n = k + 1$ i.e.,

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

   Now since $n = k + 1$ is greater than 0, so by the algorithm, we have

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

   But by the inductive hypothesis, we have power$(a, k) = a^k$ and so

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

   Hence the inductive step is also true and therefore, power$(a, n) = a^n$ for all $n \geq 0$.

2. Prove that the algorithm to compute $n!$ for $n \geq 0$ is correct.

   Solution: We have to prove that factorial$(n) = n!$ for all $n \geq 0$. For this, we use the process of mathematical induction on the integer n.

   Basis Step: We prove that factorial$(n) = n!$ when $n = 0$. Now when $n = 0$, we have factorial$(0) = 1$ from the algorithm and also $0! = 1$ by the definition of the factorial of an integer. Therefore

   $$LHS = RHS$$

   when $n = 0$.

   Inductive Step: Suppose that the equality is true when $n = k$ i.e.,

   $$factorial(k) = k!.$$

   We have to prove that the equality is true when $n = k + 1$ i.e.,

   $$factorial(k + 1) = (k + 1)!.$$

   Now since $n = k + 1$ is greater than 0, so by the algorithm, we have

   $$factorial(k + 1) = (k + 1) \cdot factorial(k).$$

   But by the inductive hypothesis, we have factorial$(k) = k!$ and so

   $$factorial(k + 1) = (k + 1) \cdot factorial(k) = (k + 1) \cdot k! = (k + 1)!.$$

   Hence the inductive step is also true and therefore, factorial$(n) = n!$ for all $n \geq 0$.

4.3 Exercise

4.3.1 Induction

1. Use mathematical induction to prove the following:

   (a) $1^2 + 2^2 + \cdots n^2 = \dfrac{n(n+1)(2n+1)}{6}$ for $n \geq 1$.

   (b) $1.2 + 2.3 + ... + n(n+1) = \dfrac{n(n+1)(n+2)}{3}$ for $n \geq 1$.

   (c) $1^3 + 2^3 + ... + n^3 = \left[\dfrac{n(n+1)}{2}\right]^2$ for $n \geq 1$.

   (d) $1^2 + 3^2 + \cdots + (2n-1)^2 = \dfrac{n(2n-1)(2n+1)}{3}$ for $n \geq 1$.

   (e) $\dfrac{1}{1 \cdot 2} + \dfrac{1}{2 \cdot 3} + \cdots + \dfrac{1}{n(n+1)} = \dfrac{n}{n+1}$ for $n \geq 1$.

   (f) $1 + 2 + 2^2 + \cdots + 2^n = 2^{n+1} - 1$ for all $n \geq 0$.

   (g) $1.1! + 2.2! + \cdots + n.n! = (n+1)! - 1$ for all $n \geq 1$.

   (h) The sum of the first n odd positive integers is n^2 i.e. $1+3+5+\cdots+(2n-1) = n^2$ for all $n \geq 1$.

   (i) If $0 \leq a < 1$ then show that $(1-a)^n \geq 1 - na$ for all $n \geq 1$.

   (j) $f_0^2 + f_1^2 + \cdots + f_n^2 = f_n f_{n+1}$ for all $n \geq 0$ where f_i's are the i^{th} Fibonacci numbers.

   (k) Prove that $n^2 + n$ is divisible by 2 for all $n \geq 1$.

   (l) Prove that 3 divides $n^3 + 2n$ whenever n is a positive integer.

   (m) Prove that 5 divides $n^5 - n$ for all $n \geq 0$.

   (n) Prove that $n^3 - n$ is divisible by 6 whenever $n \geq 0$.

   (o) $2^n < n!$ for all $n \geq 4$.

   (p) $n^2 < 2^n$ for all $n \geq 5$.

2. Use strong induction to prove that the n^{th} Fibonacci number F_n satisfies the inequality $F_n < \left(\dfrac{13}{8}\right)^n$ for $n \geq 1$.

4.3.2 Recursion

1. Find a recursive definition of the functions

   (a) $f(n) = n^2, n \geq 0$

   (b) $f(n) = 2n + 1, n \geq 1$

   (c) $f(n) = \dfrac{1}{n}, n \geq 1$

2. Give a recursive definition of the set

   (a) of all odd positive integers if x belongs to S then x+2 belongs S

   (b) of all even integers 0 belon S, if x belon S, then x+2 belong S and x-2 belong S

   (c) of all positive integer powers of 3 3 belong S, if x belong S, then 3x belong S

   (d) of all positive integers not divisible by 5

   (e) of all positive integers congruent to 2 modulo 3

3. Trace the recursive factorial algorithm for $n = 7$.
4. Trace the recursive power algorithm for $a = 3$ and $n = 6$.
5. Trace the recursive gcd algorithm for $a = 272$, $b = 1479$.