1 Induction and Recursion

1.1 Mathematical Induction

To prove $P(n)$ is true for all integers n, where $P(n)$ is a propositional function, we complete two steps:

1. We verify that $P(1)$ is true.
2. We show that the conditional statement $P(k) \rightarrow P(k+1)$ is true for all positive integers k.

To complete the inductive step of a proof using the principle of mathematical induction, we assume that $P(k)$ is true for an arbitrary positive integer k and show that under this assumption, $P(k+1)$ must also be true. The assumption that $P(k)$ is true is called the inductive hypothesis. One we complete both steps in a proof by mathematical induction, we have shown that $P(n)$ is true for all positive integers n, that is, we have shown that $\forall n P(n)$ is true where the quantification is over the set of positive integers. In the inductive set, we show that $\forall k(P(k) \rightarrow P(k+1))$ is true, where again, the domain is set of all positive integers.

Written as a rule of inference this can be written as:

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

where the domain is the set of positive integers.

The first thing we do to prove that $P(n)$ is true for all positive integers n is to show that $P(1)$ is true. This amounts to showing that the particular statement obtained when n is replaced by 1 in $P(n)$ is true. Then we must show that $P(k) \rightarrow P(k+1)$ is true for every positive integer k. To prove that this conditional statement is true for every positive integer k, we need to show that $P(k+1)$ cannot be false when $P(k)$ is true. This can be accomplished by assuming $P(k)$ is true and showing that under this hypothesis $P(k+1)$ must also be true.

A guideline 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. For statements of the form "$P(n)$ for all positive integers n", let $b = 1$, and for all statements of the form "$P(n)$ for all nonnegative integers n", let $b = 0$.
2. Show that $P(b)$ is true, taking care that the correct value of b is used.
3. Identify the inductive hypothesis, in the form "Assume $P(k)$ is true for an arbitrary fixed integer $k \geq b$."
4. State what needs to be proved under the assumption that the inductive hypothesis is true. That is, write out what $P(k+1)$ says.
5. Prove the statement $P(k+1)$ making use of the assumption $P(k)$.
6. Clearly identify the conclusion of the inductive step.
7. State the conclusion.

1.2 Strong Induction and Well-Ordering

The basis step of a proof by strong induction is the same as a proof of the same result using mathematical induction. That is, in a strong induction proof $P(n)$ is true for all positive integers n, the basis step shows that $P(1)$ is true. However, the inductive steps in these two proof methods are different. In a proof by strong induction, the inductive step shows that $P(j)$ is true for all positive integers j not exceeding k, then $P(k+1)$ is true. That is, for the inductive hypothesis we assume $P(j)$ is true for $j = 1, 2, \ldots, k$.

Let's state this principle:

To prove that $P(n)$ is true for all positive integers n, where $P(n)$ is a propositional function, we complete two steps:

Basis Step:

We verify that the proposition $P(1)$ is true.

Inductive Step:

We show that the conditional statement $[P(1) \wedge P(2) \wedge \cdots \wedge P(k)] \rightarrow P(k+1)$ is true for all positive integers k.

Note that when we use strong induction to prove $P(n)$ is true for all positive integers n, our inductive hypothesis is the assumption that $P(j)$ is true for $j = 1, 2, \ldots, k$. That is, the inductive hypothesis includes all k statements $P(1)$, $P(2)$,..., $P(k)$ to prove $P(k+1)$, rather than just the statement $P(k)$ as in a proof by mathematical induction, strong induction is a more flexible proof technique.

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

Let b be a fixed integer and j a fixed positive integer. The form of a strong induction we need tells us that $P(n)$ is true for all integers n with $n \geq b$ if we can complete these two steps.

1. Basis Step: We verify that the propositions $P(b)$, $P(b+1)$,...,$P(b+j)$ are true.
2. Inductive Step: We show that $[P(b) \wedge P(b+1) \wedge \cdots \wedge P(k)] \rightarrow P(k+1)$ is true for every integer $k \geq b+j$.

Strong induction can also work in computational geometry.

A polygon is a closed geometric figure consisting of a sequence of line segments s_1, s_2, ..., s_n, called sides. Each pair of consecutive sides, s_i and s_{i+1}, $i = 1, 2, \ldots, n-1$, as well as the last side s_n and the first side s_1, of the polygon meet at a common endpoint, called a vertex. A polygon is called simple if no two nonconsecutive sides intersect. Each simple polygon divides the plane into two regions: its interior, consisting of the points inside the curve, and its exterior, consisting of the points outside the curve.

A polygon is called convex if every line segment connecting, two points in the interior of the polygon lies entirely inside the polygon. A diagonal of a simple polygon is a line segment connecting two nonconsecutive vertices of the polygon, and a diagonal is called an interior diagonal if it lies entirely inside the polygon, except for its endpoints.

One of the most basic operations of computational geometry involves dividing a simple polygon into triangles by adding nonintersecting diagonals. This process is called triangulation.

Theorem 1.1

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

The validity of both the principle of mathematical induction and strong induction follows from a fundamental axiom of the set of integers, the well-ordering property. The well-ordering property states that every nonempty set of nonnegative integers has a least element.

1.3 Recursive Definitions and Structural Induction

We can prove results about recursively defined sets using structural induction.

We can use two steps to define a function with the set of nonnegative integers as its domain:

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.

Such a definition is called a recursive or inductive definition. note that a function $f(n)$ from the set of nonnegative integers to the set of a real numbers is the same as a sequence a_0, a_1,..., where a_i is a real number for every nonnegative integer i¿

Recursively defined functions are well-defined. That is, for every positive integer, the value of the function at this integer is determined in an unambiguous way.

We can show that the Euclidean algorithm uses $O(\log b)$ divisions to find the greatest common divisor of the positive integers a and b, where $a \geq b$.

Theorem 1.2

Lame's Theorem.

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

Just as in recursive definition of the functions, recursive definitions of sets have two parts, a basis step and a recursive step. In the basis step, an initial collection of elements is specified. In the recursive step, rules for forming new elements in the set from those already known to be in the set are provided. Recursive definitions may also include an exclusion rule, which specifies that a recursively defined set contains nothing other than those elements specified in the basis step or generated by applications of the recursive step.

We can define $\sum^*$, the set of strings over $\sum$ recursively.

Definition

The set $\sum^*$ of strings over the alphabet $\sum$ is defined recursively by

Basis Step: $\lambda \in \sum^*$ (where λ is the empty string containing no symbols).

Recursive Step: If $w \in \sum^*$ and $x \in \sum$, then $wx \in \sum^*$.

Definition

Two strings can be combined via the operation of concatenation. Let $\sum$ be a set of symbols and $\sum^*$ the set of strings formed from symbols in $\sum$. We can define the concatenation of two strings, denoted by $\cdot$, recurisvely as follows.

Basis Step: If $w \in \sum^*$, then $w \cdot \lambda = w$, where λ is the empty string.

Recursive Step: If $w_1 \in \sum^*$ and $w_2 \in \sum^*$ and $x \in \sum$, then $w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x$.

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 recurisvely by these steps:

Basis Step: A single vertex r is a rooted tree.

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

Definition

The set of extended binary trees can be defined recursively by these steps:

Basis Step: The empty set is an extended binary tree.

Recursive Step: If T_1 and T_2 are disjoint extended binary trees, there is an extended 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 when these trees are nonempty.

Definition

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

Basis Step: There is a full binary tree consisting only of 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.

Instead of using mathematical induction to directly prove results about recurisvely defined sets, we can use a more convenient form of induction known as structural induction.

Basis Step: Show that the results holds for all elements specified in the basis step of the recursive definition to be in the set.

Recursive Step: Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the results holds for these new elements.

The validity of structural induction follows from the principle of mathematical induction for nonnegative integers. To see this, let $P(n)$ state that the claim is true for all elements of the set that are generated by n or fewer applications of the rules in the recursive step of a recursive definition. We will have established that the principle of mathematical induction implies the principle of structural induction if we can show that $P(n)$ is true whenever n is a positive integer. In the basis step of a proof by structural induction we show that $P(0)$ is true. That is, we show that the result is true of all elements specified to be in the set in the basis step of the definition. A consequence of the recursive step is that if we assume $P(k)$ is true, it follows that $P(k+1)$ is true. When we have completed a proof using structural induction, we have shown that $P(0)$ is true and that $P(k)$ implies $P(k+1)$. By mathematical induction it follows that $P(n)$ is true for all nonnegative integers n. This also shows that the result is true for all elements generated by the recursive definition, and shows that structural induction is a valid proof technique.

Definition

We define the height $h(T)$ of a full binary tree T recursively.

Basis Step: The height of the 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 $h(T) = 1 + \max(h(T_1), h(T_2))$.

Theorem 1.3

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

1.4 Recursive Algorithms

Sometimes we can reduce the solution to a problem with a particular set of input values to the solution of the same problem with smaller input values.

When such a reduction can be done, the solution to the original problem can be found with a sequence of reductions, until the problem has been reduced to some initial case for which the solution is known. For instance, for finding the greatest common divisor, the reduction continues until the smaller of the two numbers of zero, because $\gcd(a, 0) = a$ when $a > 0$.

Definition

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

A recursive defintion expresses the value of a function at a positive integer in terms of the values of the function at smaller integers. This means that we can devise a recursive algorithm to evaluate a recursively defined function at a positive integer. Instead of successively reducing the computation to the evaluation of the function at smaller integers, we can start with the value of the function at one or more integers, the base cases, and successively apply the recursive definition to find the values of the function at successive larger integers. Such a procedure is called iterative. Often an iterative approach for the evaluation of a recursively defined sequence requires much less computation than a procedure using recursion.

Lemma 1.4

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

Theorem 1.5

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