Chapter 1: Logic and Proofs

**1.1 Propositional Logic**

Negation¬ Conjunction ∧

|  |  |
| --- | --- |
| **p** | **¬p** |
| T | F |
| F | T |

|  |  |  |
| --- | --- | --- |
| **p** | **q** | **p∧q** |
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | F |

Disjunction∨ Exclusive Or (Xor) ⊕

|  |  |  |
| --- | --- | --- |
| **p** | **q** | **p∨q** |
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |

|  |  |  |
| --- | --- | --- |
| **p** | **q** | **p⊕q** |
| T | T | F |
| T | F | T |
| F | T | T |
| F | F | F |

Implication → Biconditional ↔

|  |  |  |
| --- | --- | --- |
| **p** | **q** | **p→q** |
| T | T | T |
| T | F | F |
| F | T | T |
| F | F | T |

|  |  |  |
| --- | --- | --- |
| **p** | **q** | **p↔q** |
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | T |

From p →q:

* converse q →p
* contrapositive ¬q → ¬ p
* inverse ¬ p → ¬ q

**1.2 Applications of Logic Propositions**

* A list of propositions is **consistent** if it is possible to assign truth values to the proposition variables so that each proposition is true.

**1.3 Propositional Equivalences**

* A **tautology** is a proposition which is always true.
* A **contradiction** is a proposition which is always false.
* A **contingency** is a proposition which is neither a tautology nor a contradiction, such as p

De Morgan’s Laws

¬(p∧q) ≡ ¬p∨¬q

¬(p∨q) ≡ ¬p∧¬q

Identity Laws: p∧T ≡ p, p∨F ≡ p Domination Laws: p∨T ≡ T, p∧F ≡ F

Idempotent laws: p∨p ≡ p, p∧p ≡ p Double Negation Law: ¬(¬p) ≡ p

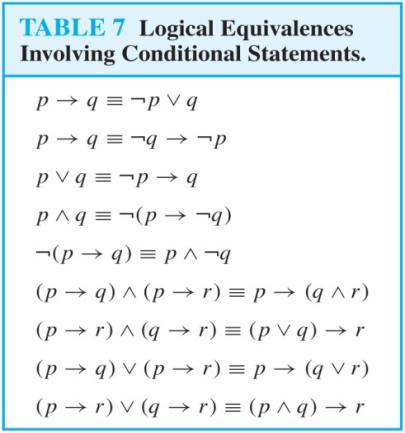
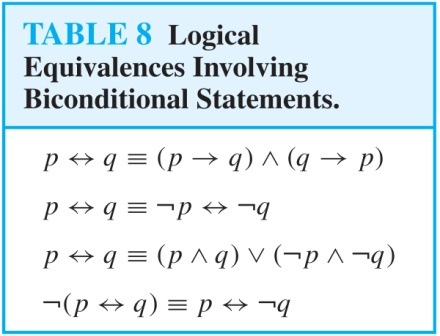
Negation Laws: p∨¬p ≡ T, p∧¬p ≡ F

Commutative Laws: p∨q ≡ q∨p, p∧q ≡ q∧p

Associative Laws: (p∧q)∧r ≡ p∧(q∧r), (p∨q)∨r ≡ p∨(q∨r)

Distributive Laws: (p∨(q∧r)) ≡ (p∨q)∧(p∨r), (p∧(q∨r)) ≡ (p∧q)∨(p∧r)

Absorption Laws: p∨(p∧q) ≡ p, p∧(p∨q) ≡ p

Propositional Satisfiability

* A compound proposition is satisfiable if there is an assignment of truth values to its variables that make it true. When no such assignments exist, the compound proposition is unsatisfiable.
* A compound proposition is unsatisfiable if and only if its negation is a tautology.

**1.4 Predicates and Quantifiers**

Quantifiers

* Universal Quantifier, “For all,” symbol: ∀
* Existential Quantifier, “There exists,” symbol: ∃

Uniqueness Quantifier

* ∃!x P(x) means that P(x) is true for one and only one x in the universe of discourse.

The quantifiers ∀ and ∃ have higher precedence than all the logical operators.

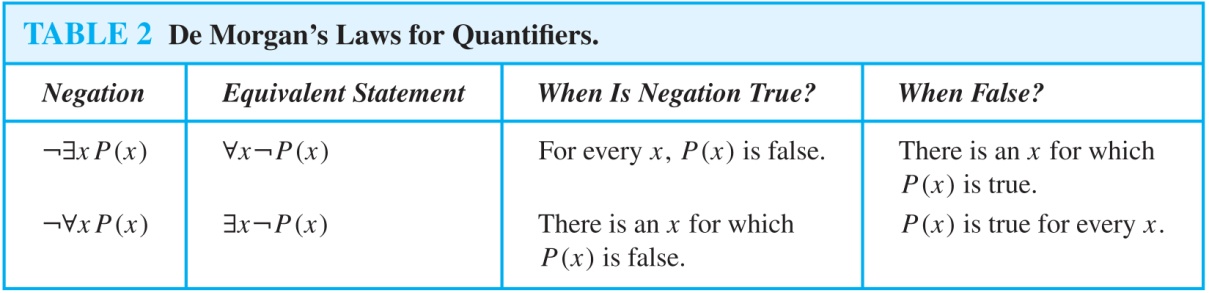
Quantifiers as Conjunctions and Disjunctions

* If U consists of the integers 1,2, and 3:

∀xP(x) ≡ P(1)∧P(2)∧P(3)

∃xP(x) ≡ P(1)∨P(2)∨P(3)

De Morgan’s Laws for Quantifiers



**1.6 Rules of Inference**

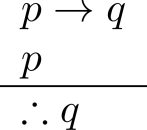
Arguments:

1. Propositional Logic

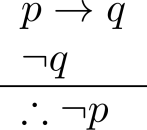
*Inference Rules*

* the final proposition are called premises. The last statement is the conclusion.

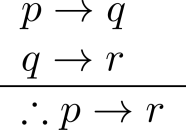
Modus Ponens (MP)

 Corresponding Tautology: (p ∧ (p →q)) → q

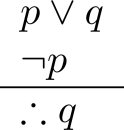
Modus Tollens (MT)

 Corresponding Tautology: (¬q∧(p →q))→¬p

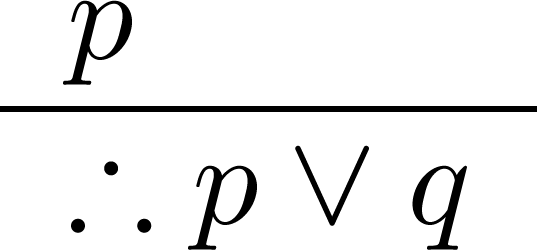
Hypothetical Syllogism (HS)

 Corresponding Tautology: ((p →q) ∧ (q→r))→(p→ r)

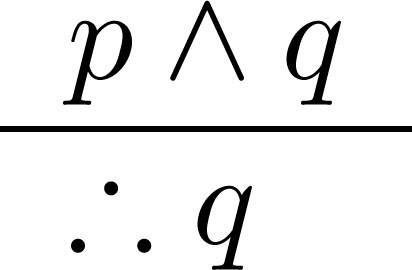
Disjunctive Syllogism (DS)

 Corresponding Tautology: (¬p∧(p ∨q))→q

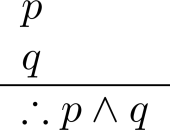
Addition

 Corresponding Tautology: p →(p ∨q)

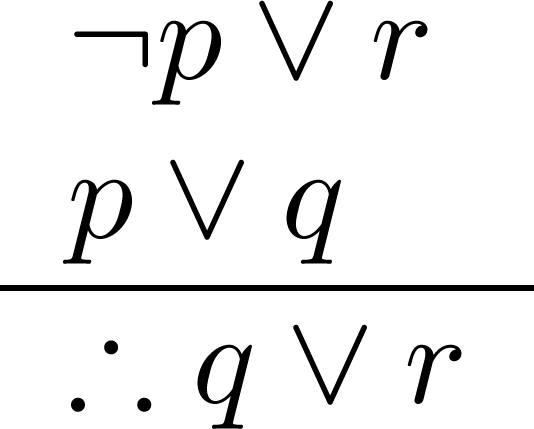
Simplification

 Corresponding Tautology: (p∧q) →p

Conjunction

 Corresponding Tautology: ((p) ∧ (q)) →(p ∧ q)

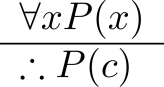
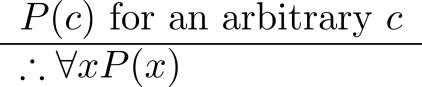
Resolution

 Corresponding Tautology: ((¬p ∨ r ) ∧ (p ∨ q)) →(q ∨ r)

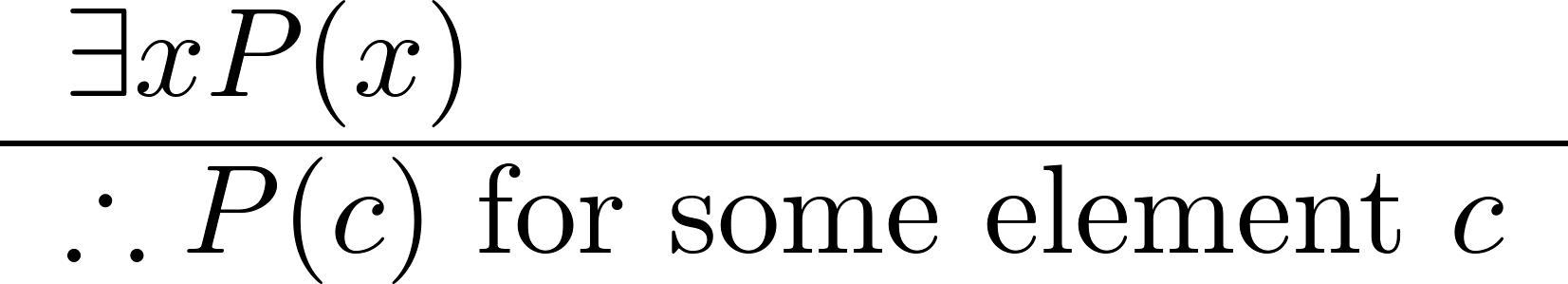
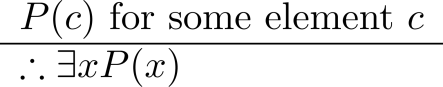
1. Predicate Logic

*Rules of Inference for Propositional Logic + Rules of Inference for Quantified Statements*

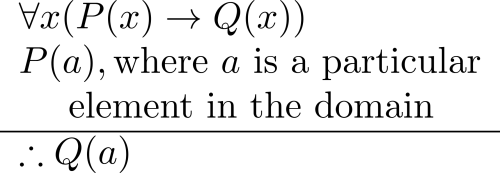
Universal Instantiation (UI) Universal Generalization (UG)

Existential Instantiation (EI) Existential Generalization (EG)

Universal Modus Ponens



.

Chapter 2: Basic Structures

**2.1 Sets**

* A set is an unordered collection of objects.
* The objects in a set are called the elements, or members of the set. A set is said to contain its elements.
* The notation a ∈ A denotes that a is an element of the set A.
* If a is not a member of A, write a ∉ A
* S={a,b,c,d}
* Each distinct object is either a member or not; listing more than once does not change the set.

N = natural numbers = {0,1,2,3….}

Z = integers = {…,-3,-2,-1,0,1,2,3,…}

Z⁺ = positive integers = {1,2,3,…..}

R = set of real numbers

R+ = set of positive real numbers

C = set of complex numbers.

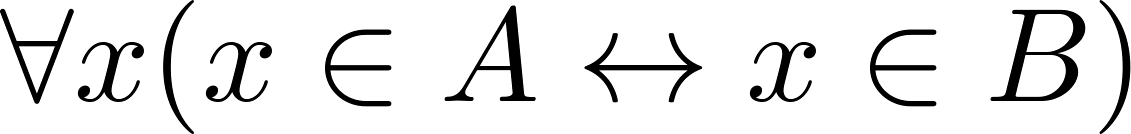
Q = set of rational numbers

S = {x | P(x)}

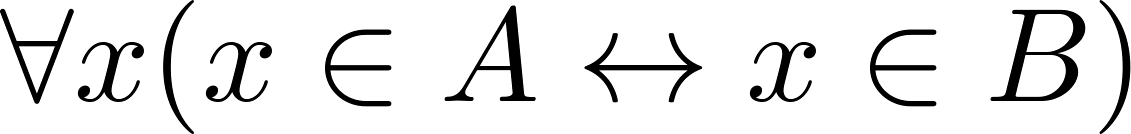
1. closed interval [a,b] 2. open interval (a,b)

* The universal set U is the set containing everything currently under consideration.
* The empty set is the set with no elements. Symbolized ∅, but {} also used.
* Sets can be elements of sets. Ex. {{1,2,3},a, {b,c}}, {N,Z,Q,R}
* The empty set is different from a set containing the empty set. ∅ ≠ { ∅ }

Set Equality

* Two sets are equal if and only if they have the same elements. 

Subsets

* The set A is a subset of B, if and only if every element of A is also an element of B.
* A ⊆ B holds if and only if  is true.

1. Because a ∈ ∅ is always false, ∅ ⊆ S ,for every set S.
2. Because a ∈ S → a ∈ S, S ⊆ S, for every set S.

Proper Subsets

* If A ⊆ B, but A≠B, then we say A is a proper subset of B, denoted by A ⊂ B. If A ⊂ B, then addin_tmp.png

Set Cardinality

* The cardinality of a finite set A, denoted by |A|, is the number of (distinct) elements of A.

Power Sets

* The set of all subsets of a set A, denoted P(A), is called the power set of A.
* If a set has n elements, then the cardinality of the power set is 2ⁿ.

Tuples

* The ordered n-tuple (a1,a2,…..,an) is the ordered collection that has a1 as its first element and a2 as its second element and so on until an as its last element.
* Two n-tuples are equal if and only if their corresponding elements are equal.
* 2-tuples are called ordered pairs.

Cartesian Product

* The Cartesian Product of two sets A and B, denoted by A × B is the set of ordered pairs (a,b) where a ∈ A and b ∈ B .

A × B = {(a,b)|a∈A ∧ b∈B}

* A subset R of the Cartesian product A × B is called a relation from the set A to the set B.

Truth Sets of Quantifiers

* Given a predicate P and a domain D, we define the truth set of P to be the set of elements in D for which P(x) is true. The truth set of P(x) is denoted by {x∈D|P(x)}

**2.2 Set Operations**

Boolean Algebra

* Propositional calculus and set theory are both instances of an algebraic system called a Boolean Algebra.
* The operators in set theory are analogous to the corresponding operator in propositional calculus.

Union

* Let A and B be sets. The union of the sets A and B, denoted by A ∪ B, is the set:

addin_tmp.png

Intersection

* The intersection of sets A and B, denoted by A ∩ B, is

addin_tmp.png

* Note if the intersection is empty, then A and B are said to be disjoint.

Complement

* If A is a set, then the complement of the A (with respect to U), denoted by Ā is the set U - A

Ā = {x ∈ U | x ∉ A} (The complement of A is sometimes denoted by Ac .)

Difference

* Let A and B be sets. The difference of A and B, denoted by A – B, is the set containing the elements of A that are not in B. The difference of A and B is also called the complement of B with respect to A.

A – B = {x | x ∈ A  x ∉ B} = A ∩ B-

The Cardinality of the Union of Two Sets

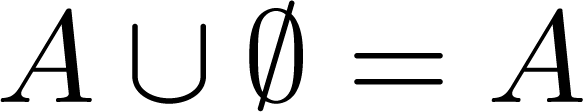
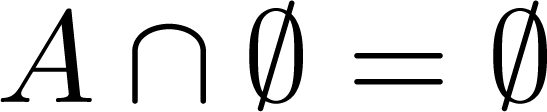
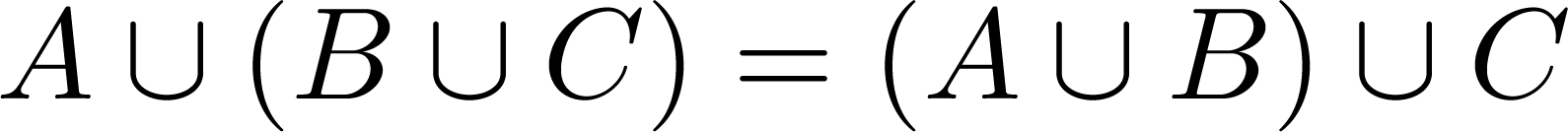
* |A ∪ B| = |A| + | B| − |A ∩ B|

Symmetric Difference

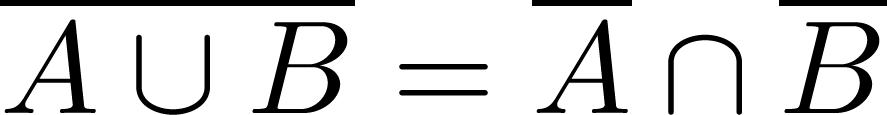
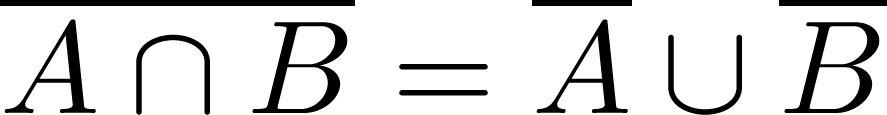
* The symmetric difference of A and B, denoted by addin_tmp.png is the set

addin_tmp.png

Set Identities

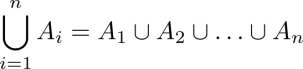
* Identity laws  addin_tmp.png
* Domination laws addin_tmp.png 
* Idempotent laws addin_tmp.png addin_tmp.png
* Complementation law addin_tmp.png
* Commutative laws addin_tmp.png addin_tmp.png
* Associative laws  addin_tmp.png
* Distributive laws

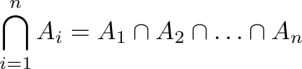
addin_tmp.png 

* De Morgan’s laws  
* Absorption laws addin_tmp.png addin_tmp.png
* Complement laws addin_tmp.png addin_tmp.png

Generalized Unions and Intersections

* Let A1, A2 ,…, An be an indexed collection of sets.





**2.3 Functions**

* Let A and B be nonempty sets. A function f from A to B, denoted f: A → B is an assignment of each element of A to exactly one element of B. We write f(a) = b if b is the unique element of B assigned by the function f to the element a of A.
* Functions are sometimes called mappings or transformations.
* A function f: A → B can also be defined as a subset of A×B (a relation). This subset is restricted to be a relation where no two elements of the relation have the same first element.
* Specifically, a function f from A to B contains one, and only one ordered pair (a, b) for every element a∈A.

addin_tmp.pngand

f: A → B:

* We say f maps A to B or f is a mapping from A to B.
* A is called the domain of f.
* B is called the codomain of f.
* If f(a) = b, 1)then b is called the image of a under f. 2)a is called the preimage of b.
* The range of f is the set of all images of points in A under f. We denote it by f(A).
* Two functions are equal when they have the same domain, the same codomain and map each element of the domain to the same element of the codomain.

Injections

* A function is said to be an injection if it is one-to-one.

Surjections

* A function f is called a surjection if it is onto.

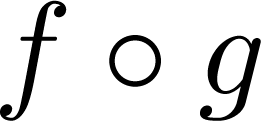
Bijections

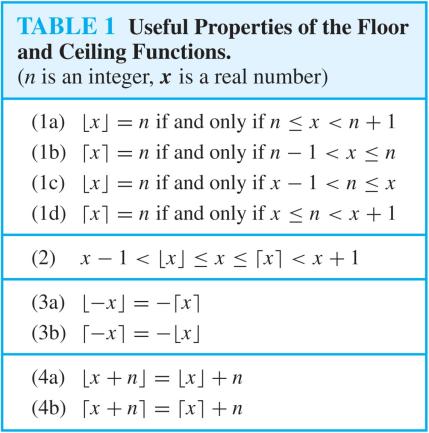
* A function f is a one-to-one correspondence, or a bijection, if it is both one-to-one and onto (surjective and injective).

Inverse Functions

* Let f be a bijection from A to B. Then the inverse of f, denotedaddin_tmp.png , is the function from B to A defined as addin_tmp.png

Composition

* Let f: B → C, g: A → B. The composition of f with g, denotedis the function from A to C defined by addin_tmp.png
* The floor function, denoted addin_tmp.png is the largest integer less than or equal to x.
* The ceiling function, denoted addin_tmp.png is the smallest integer greater than or equal to x



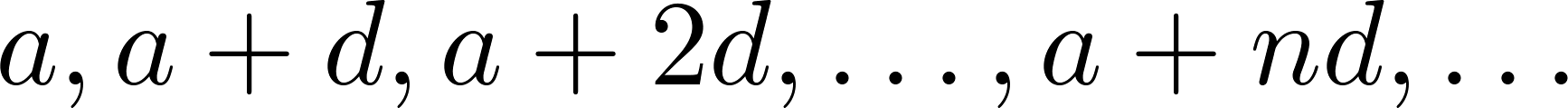
**2.4 Sequences and Summations**

* Sequences are ordered lists of elements.
* A sequence is a function from a subset of the integers (usually either the set {0, 1, 2, 3, 4, …..} or {1, 2, 3, 4, ….} ) to a set S.
* The notation an is used to denote the image of the integer n. We can think of an as the equivalent of f(n) where f is a function from {0,1,2,…..} to S.

Geometric Progression

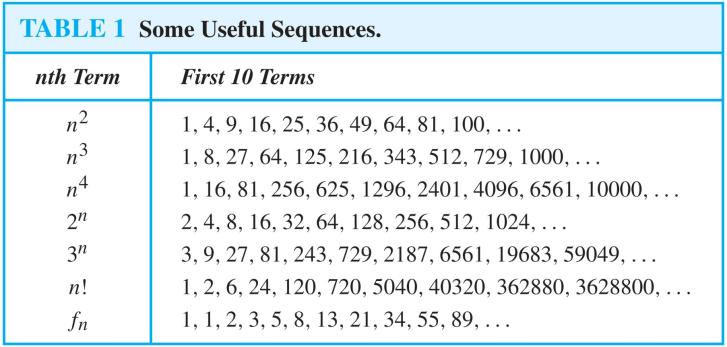
* A geometric progression is a sequence of the form: addin_tmp.png

Arithmetic Progression

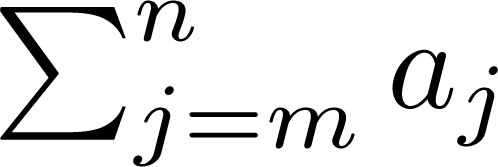
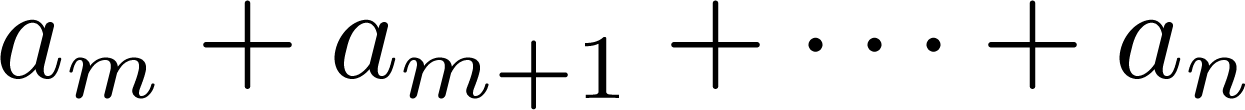
* A arithmetic progression is a sequence of the form: 

Strings

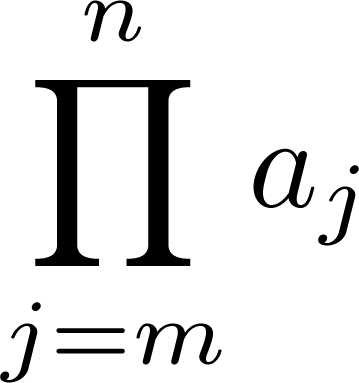
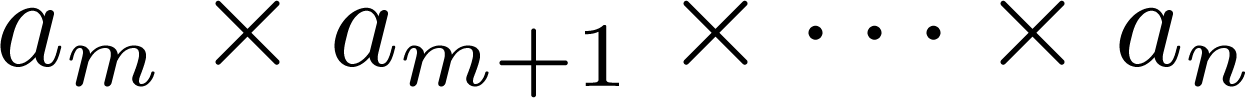
* A string is a finite sequence of characters from a finite set (an alphabet).
* The empty string is represented by λ.

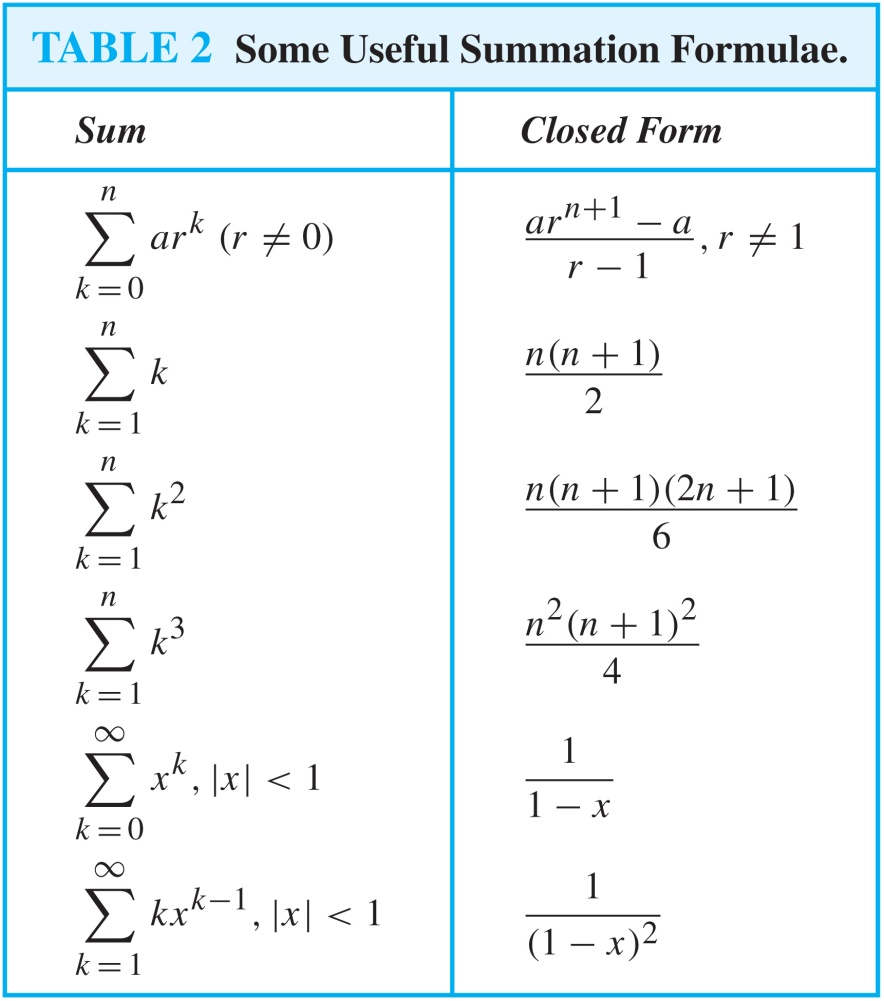


Summations

*   addin_tmp.png represents 
* The variable j is called the index of summation. It runs through all the integers starting with its lower limit m and ending with its upper limit n.
* More generally for a set S: 

Product Notation

*  addin_tmp.png addin_tmp.png represents 



Chapter 3: Algorithms

**3.1**

* An algorithm is a finite set of precise instructions for performing a computation or for solving a problem.

Linear Search Algorithm

* The linear search algorithm locates an item in a list by examining elements in the sequence one at a time, starting at the beginning.

Binary Search

* The algorithm begins by comparing the element to be found with the middle element.

Sorting

* To sort the elements of a list is to put them in increasing order (numerical order, alphabetic, and so on).
* A variety of sorting algorithms are studied in this book; binary, insertion, bubble, selection, merge, quick, and tournament.

Bubble

* Bubble sort makes multiple passes through a list. Every pair of elements that are found to be out of order are interchanged.

Insertion

* Insertion sort begins with the 2nd element. It compares the 2nd element with the 1st and puts it before the first if it is not larger.

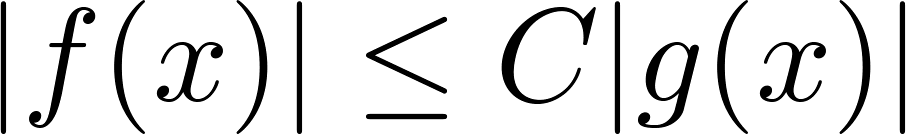
Greedy Algorithms

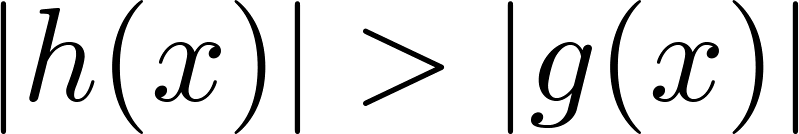
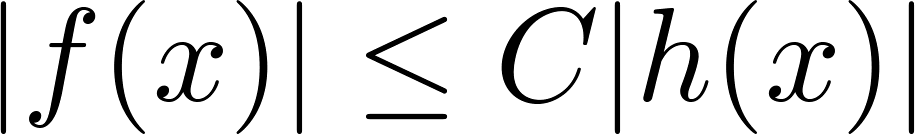
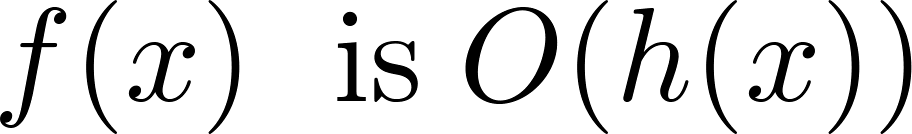
* Optimization problems minimize or maximize some parameter over all possible inputs.

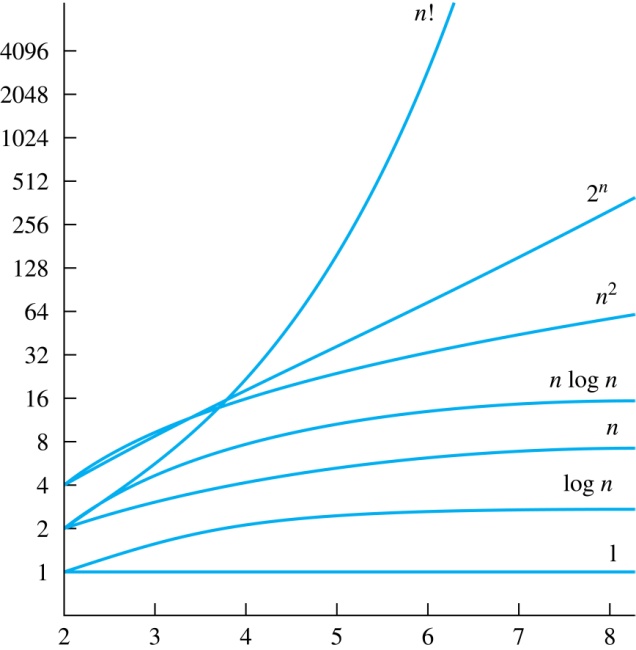
**3.2 The Growth of Functions**

Big-O Notation

* Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that f(x) is O(g(x)) if there are constants C and k such that

 whenever x > k

* This is read as “f(x) is big-O of g(x)” or “g asymptotically dominates f.”
* The constants C and k are called witnesses to the relationship f(x) is O(g(x)). Only one pair of witnesses is needed.
* Note that ifaddin_tmp.pngfor x > k and iffor all x, then  if x > k. Hence,  .
* For many applications, the goal is to select the function g(x) in O(g(x)) as small as possible (up to multiplication by a constant, of course).



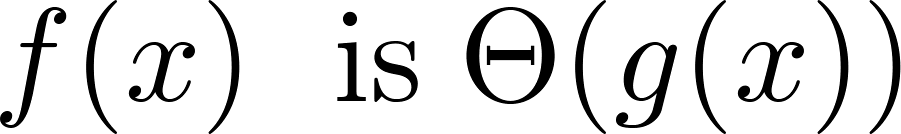
Combinations of Functions

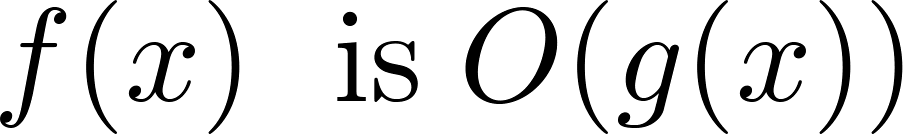
* If f1 (x) is O(g1(x)) and f2 (x) is O(g2(x)) then ( f1 + f2 )(x) is O(max(|g1(x) |,|g2(x) |)).
* If f1 (x) and f2 (x) are both O(g(x)) then ( f1 + f2 )(x) is O(g(x)).
* If f1 (x) is O(g1(x)) and f2 (x) is O(g2(x)) then ( f1 f2 )(x) is O(g1(x)g2(x)).

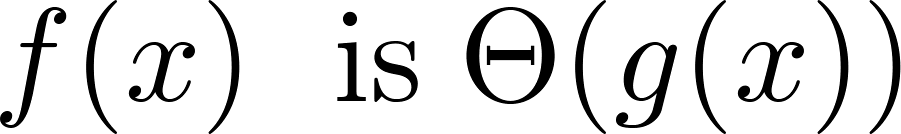
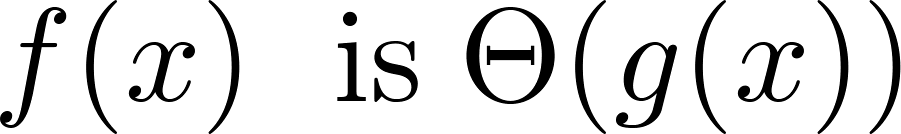
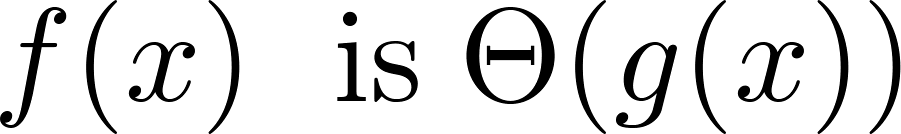
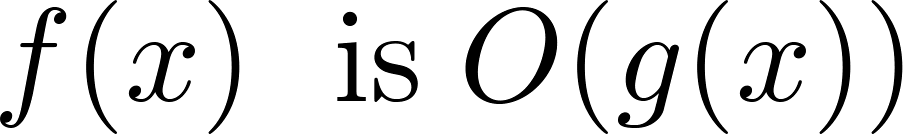
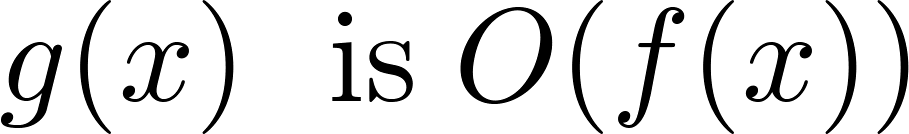
Big-Omega Notation

* Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that  if there are constants C and k such that addin_tmp.png when x>k
* We say that “f(x) is big-Omega of g(x).”
* Big-Omega tells us that a function grows at least as fast as another.

Big-Theta Notation

* Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. The function 

and 

* We say that “f is big-Theta of g(x)” and also that “f(x) is of order g(x)” and also that “f(x) and g(x) are of the same order.”
*  if and only if there exists constants C1 , C2 and k such that C1g(x)<f(x)<C2g(x) if x > k. This follows from the definitions of big-O and big-Omega.
* When  it must also be the case that 
* Note thatif and only if it is the case that  and 

Chapter 4: Number Theory and Cryptography

**4.1 Divisibility and Modular Arithmetic**

Division

* If a and b are integers with a ≠ 0, then a divides b if there exists an integer c such that b = ac.
* When a divides b we say that a is a factor or divisor of b and that b is a multiple of a.
* The notation a | b denotes that a divides b.
* If a | b, then b/a is an integer.
* If a does not divide b, we write a ∤ b.

1. If a | b and a | c, then a | (b + c);
2. If a | b, then a | bc for all integers c;
3. If a | b and b | c, then a | c.

Division Algorithm

* If a is an integer and d a positive integer, then there are unique integers q and r, with 0 ≤ r < d, such that a = dq + r

- d is called the divisor. 除数

- a is called the dividend. 被除数

- q is called the quotient. 商

- r is called the remainder. 余数

* Definitions of Functions

**div** and **mod**

q = a **div** d

r = a **mod** d

Congruence Relation

* If a and b are integers and m is a positive integer, then a is congruent to b modulo m if m divides a – b.
* a ≡ b
* Two integers are congruent mod m if and only if they have the same remainder when divided by m.
* If a is not congruent to b modulo m, we write a ≢ b (mod m)
* a = b + km.

(mod m) and **mod** m

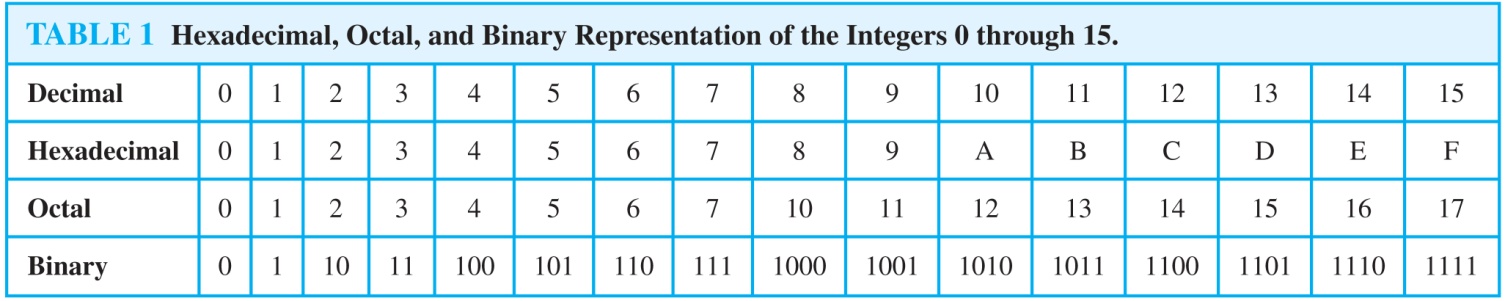
* Let a and b be integers, and let m be a positive integer. Then a ≡ b (mod m) if and only if a **mod** m = b **mod** m.

+ and \*

* a + c ≡ b + d (mod m) and ac ≡ bd (mod m)
* (a + b) (**mod** m) = ((a **mod** m) + (b **mod** m)) **mod** m and ab **mod** m = ((a **mod** m) (b **mod** m)) **mod** m.

**4.2 Integer Representations and Algorithms**

* b = 2 (binary), b = 8 (octal) , and b= 16 (hexadecimal)



**4.3 Primes and Greatest Common Divisors**

Prime

* A positive integer p greater than 1 is called prime if the only positive factors of p are 1 and p. A positive integer that is greater than 1 and is not prime is called composite.

The Sieve of Erastosthenes

* If an integer n is a composite integer, then it has a prime divisor less than or equal to √n.
* if n = ab, then  a ≤ √n or b ≤√n.

Infinitude of Primes

* There are infinitely many primes. (Euclid)

Mersene Primes

* Prime numbers of the form 2^p − 1 , where p is prime, are called Mersene primes.