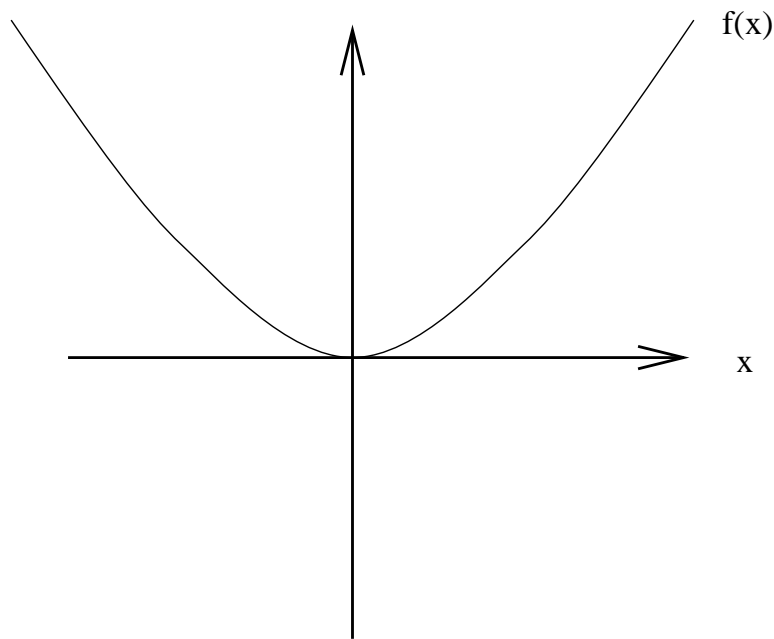


Functions - Mathematical View

- A function is a *mapping* that assigns to each element from a given domain some value from another or the same domain.
- Mathematical functions are often drawn as graphs, e.g., the function that maps each real number x to x^2 , or the function that maps x to $x^3 + x$, etc. The values along the y -coordinate are defined by an expression in terms of x , e.g., $y = x^2$ or $y = x^3 + x$.



- Note that not all identities in variables x and y define a function. For instance, the formula for a circle, $x^2 + y^2 = 1$, describes no function.

Functions - Computational View

- A function also describes the input/output behavior of a computational device or program: for each suitable input value a corresponding output value is computed.
- For instance, a grading program takes as input a list of relevant scores (exams, home works, etc.) and produces as output a normalized number, say, between 0 and 100, that can then be converted into a letter grade.
- A sorting program takes as input a list of integers and produces as output a sorted version of the input list. There are many different sorting algorithms that compute the same function, but differ in their efficiency.
- It should be noted that many (actually, most) functions cannot be computed algorithmically.
- We first formalize the mathematical definition of functions in terms of set theory, and discuss computational aspects of functions later.

Basic Terminology

- We use the notation $f : A \rightarrow B$ to indicate that f is a function from A to B and denote by $f(a)$ the value assigned to the argument a by f .
- For example, the *squaring function* is a function $f : \mathbf{R} \rightarrow \mathbf{R}$ such that $f(x) = x^2$, for all real numbers x .
- A *constant function* (on the integers) is a function $f : \mathbf{Z} \rightarrow \mathbf{Z}$ such that $f(n) = k$, for all integers n , where k is a fixed value, e.g., $k = 2$.
- We call A the *domain* of the function f , and B the *codomain*. We also speak of “a function from A to B ” or, if $A = B$, “a function on A .”
- The *range* of a function f is defined as the collection of all values y in B , such that $y = f(x)$, for some x in A .

In the examples above range and codomain of the function are different.

Set-Theoretic Definition

- Functions are mathematical objects and can be defined in terms of sets.
- Specifically, a function $f : A \rightarrow B$ is a subset of the Cartesian product $A \times B$ that satisfies the following two properties:

Uniqueness

The set f does not contain two pairs (x, y) and (x, z) , where y and z are different.

Completeness

For each element x of the set A , there exists an element y of B , such that the pair (x, y) is in f .

- For example, the set

$$\{(a, 1), (b, 2), (c, 1)\}$$

describes a function from $\{a, b, c\}$ to $\{1, 2\}$, that maps a to 1, b to 2, and c to 1.

Partial Functions

- Is the (multiplicative) *inverse mapping*, which assigns to each rational number m/n the number n/m a function on the rational numbers?

No, because n/m is not defined if $m = 0$.

But it can be thought of as a function on the non-zero rational numbers.

- In computer science one also often encounters “partial” functions, such as division, which are undefined for certain arguments.
- In set-theoretic terms, a partial function is a subset of $A \times B$ that satisfies the uniqueness property, but not the completeness property.

Equality of Functions

- Two functions f and g from A to B are said to be *equal*, written $f = g$, if they agree on all arguments, i.e., $f(x) = g(x)$ for all $x \in A$.
- For example, let f and g be functions on the integers such that $f(n) = n^2 - 1$ and $g(n) = (n + 1)(n - 1)$. Then $f = g$.
- The *identity function* on a set A is the function $id : A \rightarrow A$ such that $id(x) = x$, for all $x \in A$.
- Is the function f on the natural numbers, such that $f(n) = n \bmod n$, equal to the identity function on \mathbf{N} ?

Multiple-Argument Functions

- Functions of two or more arguments may be viewed as functions where the domain is a set of tuples (of fixed length).
- For example, a *binary* function is a function f of type $f : A_1 \times A_2 \rightarrow B$.
- The addition function on the integers is a binary function that maps each pair of integers (m, n) to their sum $m + n$.
- In general, by an *n-ary function* we mean a function of type

$$f : (A_1 \times A_2 \times \cdots \times A_n) \rightarrow B$$

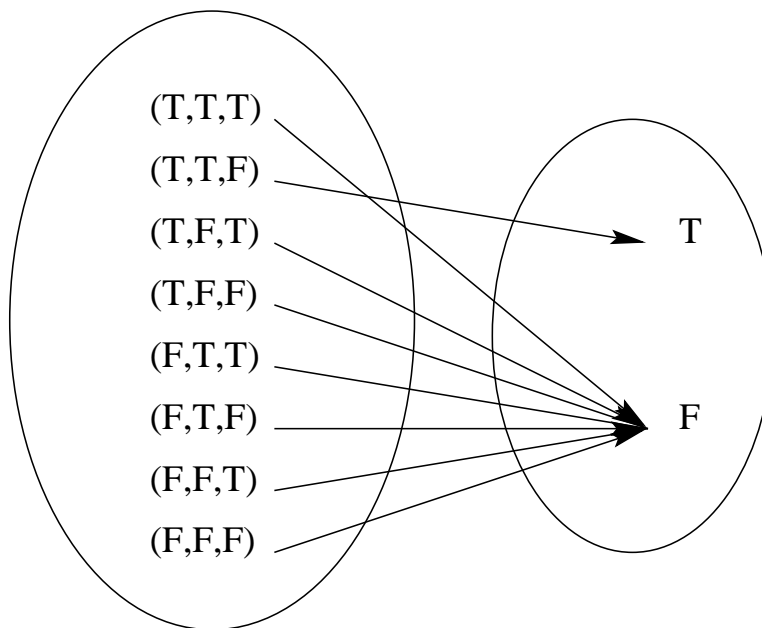
the domain of which is a set of n -tuples.

- It is of course also possible for the codomain of a function to be a set of pairs or tuples.

For example, we may define a function $f : \mathbf{Z} \times \mathbf{Z} \rightarrow \mathbf{Z} \times \mathbf{Z}$ such that $f(m, n) = (q, r)$, where q and r are the quotient and remainder, respectively, of the integer division of m by n .

Boolean Functions

- Truth tables describe functions, called *Boolean functions*, that map n -tuples of truth values to single truth values.
- On the other hand, every propositional formula A defines a truth table, and hence a (unique) Boolean function.
- For example, the formula $p \wedge q \wedge \sim r$ defines the following Boolean function.



One-to-One and Onto Functions

- A function $f : X \rightarrow Y$ is said to be *one-to-one* (or *injective*) if, and only if, for all elements x and y in X , we have $x = y$ whenever $f(x) = f(y)$.

- **Example.** Let f and g be functions with domain $\{1, 2\}$ and co-domain $\{1, 2, 3\}$, defined by:

$$\begin{aligned} f(1) &= 1 & \text{and} & & f(2) &= 1 \\ g(1) &= 2 & \text{and} & & g(2) &= 3 \end{aligned}$$

Then g is one-to-one, but f is not.

- A function $f : X \rightarrow Y$ is said to be *onto* (or *surjective*) if, and only if, for every element y in Y , there exists an element x in X , such that $f(x) = y$.

A function is onto iff its range equals its codomain.

- For example, the function f from $\{1, 2, 3\}$ to $\{1, 2\}$, defined by:

$$f(1) = 1, f(2) = 1, f(3) = 2$$

is surjective, whereas no function from $\{1, 2\}$ to $\{1, 2, 3\}$ can possibly be onto.

- A function that is one-to-one and onto is said to be *bijective*.

Composition of Functions

- If the range of a function $f : X \rightarrow Y'$ is a subset of the domain Y of a function $g : Y \rightarrow Z$, we can *compose* the two functions f and g to obtain a function

$$g \circ f : X \rightarrow Z$$

defined by: $(g \circ f)(x) = g(f(x))$, for all $x \in X$.

- For example, let f be the successor function on the integers, i.e., $f(n) = n + 1$, and g be the squaring function, $g(n) = n^2$. Then

$$(f \circ g)(n) = n^2 + 1$$

whereas

$$(g \circ f)(n) = (n + 1)^2.$$

- If domain and codomain of a function are identical, it can be composed with itself:

$$(f \circ f)(n) = n + 2$$

and

$$(g \circ g)(n) = n^4.$$

Properties of Composition

- Recall that the *identity function* i_A on a domain A is defined to be a function from A to A with $i_A(x) = x$, for all elements x of the domain A .

- **Theorem**

If f is a function from A to B , then

$$f \circ i_A = f$$

and

$$i_B \circ f = f.$$

- Certain properties of functions carry over to their composition.

- **Theorem**

If f is a one-to-one function from A to B and g a one-to-one function from B to C , then $g \circ f$ is a one-to-one function from A to C .

If f is an onto function from A to B and g an onto function from B to C , then $g \circ f$ is an onto function from A to C .

Permutations

- *Permutations* are bijective functions that rearrange a (finite) sequence of the numbers $1, \dots, n$.

- For example, reversing a sequence is a permutation:

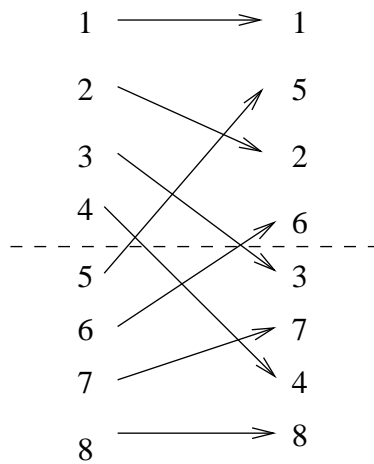
$$(1, 2, 3, 4, 5, 6) \mapsto (6, 5, 4, 3, 2, 1)$$

or

$$(3, 2, 6, 4, 1, 5) \mapsto (5, 1, 4, 6, 2, 3).$$

- Another examples is a *card shuffle*. We can represent the 52 cards in a deck by the numbers $1, 2, \dots, 52$. Shuffling corresponds to rearranging a given sequence of these numbers.
- The reason we use shuffles on a deck of cards is to mix them up. A *perfect shuffle* of a deck of cards splits the deck in half and then merges the two halves so that no two adjacent cards come from the same half.

- Here is an example of a perfect shuffle on an eight card deck.



Perfect Shuffles

- What happens if we do a series of eight consecutive perfect shuffles on a 52 card deck?
- The example shows that we end up with the cards in the same order as we started!!!
- What this shows is that the composition of eight shuffles gives the identity function:

$$x = s \circ s \circ s \circ s \circ s \circ s \circ s \circ s(x)$$

- Why do eight suffice? Observe that (except for the last card), the shuffle takes position x to position $2x \bmod 51$. Further, $2^8 \bmod 51 = 1$, so multiplying it eight times is like multiplying by 1, i.e. the identity function.

The Inverse of a Function

- If f is a bijective function from A to B , its *inverse* is a function f^{-1} from B to A , defined by:

$$f^{-1}(y) = x \text{ iff } f(x) = y.$$

- For example, if f is the function on the real numbers, defined by $f(x) = 4x - 1$, then $f^{-1}(y) = (y + 1)/4$, for all real numbers y .
- Note that the inverse of a bijective function is also bijective.
- The above definition is valid since for a bijective function f the value $f(x)$ is uniquely determined by x . But this is also the case for one-to-one functions. Can we define an inverse for any one-to-one function f , even if f is not onto?
Yes, but in that case the inverse is a function from the range of f to the domain of f .

Application: Codes

- Let Σ be the set $\{a, b, c, d, e, f\}$ and Σ^* be the set of all finite sequences, or *strings*, of elements of Σ .
- Let us map characters in Σ to binary strings as follows.

x	a	b	c	d	e	f
$C(x)$	000	001	010	011	100	101

- We can extend this mapping to a function on Σ^* by defining:

$$C(x_1x_2 \dots x_n) = C(x_1)C(x_2) \dots C(x_n).$$

- For example, the code $C(eaabb)$ is the binary string
100000000001001.

- Is the function C one-to-one? Onto?
- In general, encoding functions need to be one-to-one, for otherwise decoding may be difficult, or impossible.

Character codes like this (ascii) is how text strings are represented in computers.

Encoding – Decoding

- Now consider a different, variable-length, code:

x	a	b	c	d	e	f
D(x)	0	1	10	11	100	101

- With this function we can encode *eabbe* by a shorter binary string

100011100.

- But can we also decode strings?

No, the code could also represent the string *baaabbaa*. In other words, the function is not one-to-one.

Prefix Codes

- A *prefix code function* is a function such that no codeword $C(x)$ is a prefix of another codeword $C(y)$, where x and y denote different symbols.
- Prefix code functions are one-to-one.
- Let's look at another variable-length code.

x	a	b	c	d	e	f
$D'(x)$	0	101	100	111	1101	1100

The encoding of *eabbe* by D' yields

110101011011101.

Is this function one-to-one?

How are binary strings decoded in this case?

Application: Hashing

You are given a set of student records, each of which includes the student's ID-number and which have to be stored in a table. For any given ID-number one has to be able to determine whether there is a record with that number and, if so, to retrieve the record. It may also be necessary to add new records and delete existing ones.

It is fairly easy to come up with some method for solving this problem:

Store the first record in the first table slot, the second in the second slot, and so on. To search for a record, simply scan all entries from the beginning (to the end if necessary).

How are records added or deleted?

It is much harder to design a method for which the search, add, and delete operations are efficient *and* at the same time storage space is used economically (i.e., no huge table for just a few records).

One solution is to use a *hash table* based on a well-chosen *hash function*.

A Hash Table

For simplicity let us first assume that the number of records is small, say no more than 7.

We will use a table with seven entries, numbered 0, 1, 2, ..., 6.

- Given a (9-digit) ID-number x , compute its *hash value* $h(x)$ using the hash function h , where

$$h(x) = x \bmod 7.$$

- Store the student's record in table entry $h(x)$.

Example.

0	356-63-3102
1	
2	513-40-8716
3	223-79-9061
4	
5	328-34-3419
6	

The elements x for which one computes hash values are also called *keys*. In this example the keys are ID-numbers.

Collisions

Typically, the domain X of a hash function is much larger than its co-domain Y , though the subset X' of those elements of X for which hash values need to be computed is usually about the same size as Y .

If the function f , when restricted to X' as a domain, is one-to-one, then hashing works fine. If it is not one-to-one, there may be *collisions*.

Where do we store the record of a student with ID-number 223 – 79 – 9068?

Collisions can be resolved in two ways:

- store the record in the “next available” slot, or
- store (a pointer to) a list of record in each slot.

Hash functions are typically onto – why is this good?

In the example, if the number of student records is reasonably large, say around 8,000, the function h above, with $h(x) = x \bmod 7$, is not suitable. A more reasonable function might be

$$h'(x) = x \bmod 10,000.$$

The Pigeonhole Principle

- If A and B are finite domains and B has fewer elements than A , then there is no one-to-one function from A to B .

This observation is also known as the *Pigeonhole Principle*.

- For example, let A be the set $\{1, 2, 3, 4, 5, 6, 7, 8\}$. How many integers from A need to be selected so that, regardless of the choice of selection, there is at least one pair with a sum of 9?

Four is not enough, as we may select 1, 2, 3, 4 where no pair yields a sum larger than 7.

But any selection of five integers from A must contain a pair whose sum is 9. To see why, observe that A can be partitioned into four different subsets $A_1 = \{1, 8\}$, $A_2 = \{2, 7\}$, $A_3 = \{3, 6\}$, and $A_4 = \{4, 5\}$, defining pairs whose sum is 9.

Now if a_1, a_2, a_3, a_4 , and a_5 are the selected integers from A , we define a function f , by setting $f(a_i)$ to be the set A_j that contains a_i .

By the pigeonhole principle, the function f is not one-to-one, so that there exists two integers a_i and a_j with $f(a_i) = f(a_j)$. In other words, there must be one subset A_k , both of whose elements are selected. The corresponding sum is 9.

A Bald Statement

- Despite its simplicity, the pigeonhole principle can be used to solve an amazing variety of problems.
- *Claim:* There must be at least two non-bald New Yorkers who have exactly the same number of hairs on their heads!
- *Proof:* The maximum number of hairs on a human head is 1,000,000, and there are greater than 1,000,000 non-bald New Yorkers. ■
- Note that this proof, although completely rigorous, is not constructive. We don't figure out which two people share the same hair count, or what the hair count is – only that the given pair must exist.

A Subset of Divisors

- Suppose you are given an arbitrary subset of 101 distinct integers from the set $S = \{1, 2, 3, \dots, 200\}$. There must be two integers x, y in S such that x divides y .
- *Proof:* Every positive integer n can be written as $2^k \times m$, for $k \geq 0$ and m odd. (Why? Factoring all the twos from n leaves an odd number.)

Thus every number in S can be mapped to an odd number from 1 to 199. There are exactly 100 such numbers. (Why? These are the integers $2i - 1$ for $1 \leq i \leq 100$)

Thus at least two of the 101 distinct integers must be mapped to the same odd number m , say $x = 2^k m$ and $y = 2^{k+c} m$. Then x must divide y . ■

This result can be generalized to state that any subset S of $n+1$ integers from 1 to $2n$ must contain a pair x, y in S such that x divides y .