ICOM 4075:
Foundations of Computing

Lecture 7:
Functions (3)

Department of Electrical and Computer Engineering
University of Puerto Rico at Mayagüez
Summer 2005

Lecture Notes Originally Written By Prof. Yi Qian

Homework 6

- Due Tuesday, March 16, 2010
- **Section 2.2: (pp.98-100)**

  1. b. d. f.
  2. b.
  3. b. d.
  4. b.
  5.
  7. b.
  8. b.
  9. b. d. f. h.
  10.
  11. b.

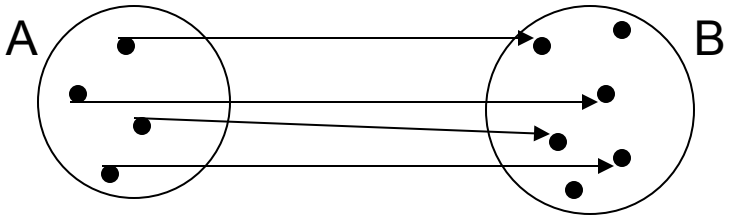
Reading

- Textbook: James L. Hein, *Discrete Structures, Logic, and Computability*, 2nd edition, Chapter 2. Section 2.3

Properties Of Functions

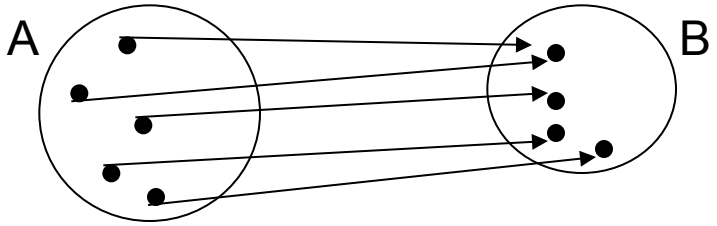
- Injections and Surjections
  - Injective Functions

    A function f: A→B is called *injective* (also *one-to-one*, or an *embedding*) if it maps distinct elements of A to distinct elements of B. Another way to say this is that f is injective if $x \neq y$ implies $f(x) \neq f(y)$. Yet another way to say this is that f is injective if $f(x) = f(y)$ implies $x = y$. An injective function is called an *injection*.
    - E.g., The figure in the following is an injection from a set A to a set B.



  - Surjective Functions

    A function f: A→B is called *surjective* (also *onto*) if the range of f is the codomain B. Another way to say this is that f is surjective if each element $b \in B$ can be written as $b = f(x)$ for some element $x \in A$. A surjective function is called a *surjection*.
    - E.g., The figure in the following is a surjection from A to B.



Injective or Surjective

- A few examples of functions that have one or the other of the injective and surjective properties:

1. The function f: R→Z defined by f(x) = $\lceil x + 1 \rceil$

2. The function f: $N_8 \rightarrow N_8$ defined by *f(x) = 2x mod 8*

3. Let g: N→N x N be defined by g(x) = (x, x).

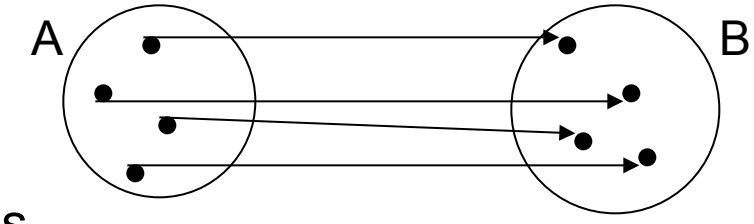
4. The function f: N x N → N defined by f(x, y) = 2x + y

Injective or Surjective

- A few examples of functions that have one or the other of the injective and surjective properties:
  1. The function f: R→Z defined by f(x) = $\lceil x + 1 \rceil$ is surjective because for any y∈Z there is a number in R, namely y – 1, such that f(y – 1) = y. But f is not injective because, for example, f(3.5) = f(3.7).
  2. The function f: $N_8 \rightarrow N_8$ defined by *f(x) = 2x mod 8* is not injective because, for example, f(0) = f(4). F is not surjective because the range of f is only the set {0, 2, 4, 6}.
  3. Let g: N→N x N be defined by g(x) = (x, x). Then g is injective because if x, y ∈N and x ≠ y, then g(x) = (x, x) ≠ (y, y) = g(y). But g is not surjective because, for example, nothing maps to (1, 2).
  4. The function f: N x N → N defined by f(x, y) = 2x + y is surjective. To see this, notice that any z ∈ N is either even or odd. If z is even, then z = 2k for some k ∈ N so f(k, 0) = z. If z is odd, then z = 2k + 1 for some k ∈ N, so f(k, 1) = z. Thus f is surjective. But f is not injective because, for example, f(0, 2) = f(1, 0).

Bijections and Inverses

- Bijections
  - A function is called *bijective* if it is both injective and surjective. Another term for bijective is "*one-to-one and onto*". A bijective function is called a *bijection* or a "*one-to-one correspondence*".
  - E.g., the following figure pictures a bijection from A to B.



- Inverse Functions
  - Bijections always come in pairs. If f: A→B is a bijection, then there is a function g: B→A, called the *inverse* of f, defined by g(b) = a if f(a) = b. Of course, the inverse of f is also a bijection and we have g(f(a)) = a for all a ∈ A and f(g(b)) = b for b ∈ B. In other words, $g \circ f = id_A$ and $f \circ g = id_B$.
  - There is exactly one inverse of any bijection f.
  - The inverse of f is often denoted by the symbol f^{-1}. So if f is a bijection and f(a) = b, then $f^{-1}(b) = a$. Notice the close relationship between the equation $f^{-1}(b) = a$ and the pre-image equation $f^{-1}(\{b\}) = \{a\}$.

A Bijection

- Let (0, 1) = {x ∈ R | 0 < x < 1} and let R^+ denote the set of positive real numbers. We'll show that the function f: (0, 1) → R^+ defined by f(x) = x/(1 – x) is a bijection.

  To show that f is an injection, let f(x) = f(y). Then x/(1 – x) = y/(1 – y), which can be cross multiplied to get x – xy = y – xy. Subtract –xy from both sides to get x = y. Thus f is injective.

  To show that f is surjective, let y > 0 and try to find x ∈ (0, 1) such that f(x) = y. Solve the equation: x/(1 – x) = y. Cross multiply and solve for x to obtain x = y/(y + 1).

  It follows that f(y/(y + 1)) = y, and since y > 0, it follows that

  0 < y/(y + 1) < 1. Thus f is surjective.

  Therefore, f is a bijection.

Inverses

- Let's look at two bijective functions together with their inverses:
  - 1. Let *Odd* and *Even* be the sets of odd and even natural numbers, respectively. The function f: *Odd*→*Even* defined by f(x) = x – 1 is a bijection. The inverse of f can be defined by $f^{-1}(x) = x + 1$.

    Notice that $f^{-1}(f(x)) = f^{-1}(x - 1) = (x - 1) + 1 = x$.
  - 2. The function f: $N_5 \rightarrow N_5$ defined by *f(x) = 2x mod 5* is bijective because, f(0) = 0, f(1) = 2, f(2) = 4, f(3) = 1, and f(4) = 3. The inverse of f can be defined by *$f^{-1}(x) = 3x \bmod 5$*. For example, $f^{-1}(f(4)) = 3f(4) \bmod 5 = 9 \text{ mode } 5 = 4$.

The Mod Function and Inverses

- The Mod Function and Inverses:
  - Let n > 1 and let f: $N_n \rightarrow N_n$ be defined as follows, where a and b are integers.

    $$f(x) = (ax + b) \bmod n.$$

    Then f is a bijection if and only if gcd(a, n) = 1. When this is the case, the inverse function f^{-1} is defined by

    $$f^{-1}(x) = (kx + c) \bmod n,$$

    where c is an integer such that f(c) = 0, and k is an integer such that 1 = ak + nm for some integer m.

  Proof:

The Mod Function and Inverses (Proof)

Proof:

We'll prove the iff part of the statement and leave the form of the inverse as an exercise.

Assume that f is a bijection and show that gcd(a, n) = 1. Then f is surjective, so there are numbers s, c $\in$ N_n such that f(s) = 1 and f(c) = 0. Using the definition of f, these equations become

(as + b) mod n = 1 and *(ac + b) mod n = 0*.

Therefore, there are integers q_1 and q_2 such that the two equations become

$as + b + nq_1 = 1$ and $ac + b + nq_2 = 0$.

Solve the second equation for b to get $b = -ac - nq_2$, and substitute for b in the first equation to get

$1 = a(s - c) + n(q_1 - nq_2)$.

Since gcd(a, n) divides both a and n, it divides the right side of the above equation, and therefore must also divides 1. Therefore, gcd(a, n) = 1.

Now assume that gcd(a, n) = 1 and show that f is a bijection. Since N_n is finite, we need only show that f is an injection to conclude that it is a bijection. So let x, y $\in$ N_n and let f(x) = f(y). Then

(ax + b) mod n = (ay + b) mod n,

which by (2.4a) implies that n divides (ax + b) – (ay + b). Therefore, n divides a(x – y), and since gcd(a, n) = 1, we conclude from (2.2d) that n divides x – y. But the only way for n to divide x – y is for x – y = 0 because both x, y $\in$ N_n. Thus x = y, and it follows that f is injective, hence also surjective, and therefore bijective.

QED.

Injective and Surjective Relationships

- Injective and Surjective Relationships:
  - a. If f and g are injective, then g○f is injective.
  - b. If f and g are surjective, then g○f is surjective.
  - c. If f and g are bijective, then g○f is bijective.
  - d. There is an injection from A to B if and only if there is a surjection from B to A.

  Proof:

Injective and Surjective Relationships (Proof)

– **Proof**:

We will prove part (d) and leave the others as exercise.

Suppose that f is an injection from A to B. We will define a function g from B to A. Since f is an injection, it follows that for each $b \in$ range(f) there is exactly one $a \in A$ such that b = f(a). In this case, we define g(b) = a. For each $b \in B$ – range(f) we have the freedom to let g map b to any element of A that we like. So g is a function from B to A and we defined g so that range(g) = A. Thus g is surjective.

For the other direction, assume that g is a surjection from B to A. We'll define a function f from A to B. Since g is a surjection, it follows that for each $a \in A$, the pre-image $g^{-1}(\{a\}) \neq \Phi$. So we can pick an element $b \in g^{-1}(\{a\})$ and define f(a) = b. Thus f is a function from A to B. Now if $x, y \in A$ and $x \neq y$,

then $g^{-1}(\{x\}) \cap g^{-1}(\{y\}) = \Phi$. Since $f(x) \in g^{-1}(\{x\})$ and $f(y) \in g^{-1}(\{y\})$, it follows that $f(x) \neq f(y)$. Thus f is injective.

QED.

The Pigeonhole Principle

- Pigeonhole Principle:
  - If m pigeons fly into n pigeonholes where m > n, then one pigeonhole will have two or more pigeons.
- We can describe the pigeonhole principle in more formal terms:
  - If A and B are finite sets with |A| > |B|, then every function from A to B maps at least two elements of A to a single element of B. This is the same as saying that no function from A to B is an injection.

Pigeonhole Examples

- 1. The “musical chairs” game is played with n people and n – 1 chairs for them to sit on when the music stops.
- 2. In a group of eight people, two were born on the same day of the week.
- 3. If a six-sided die is tossed seven times, one side will come up twice.
- 4. If a directed graph with n vertices has a path of length n or longer, then the path must pass through some vertex at least twice. This implies that the graph contains a cycle.
- 5. In any set of n + 1 integers, there are two numbers that have the same remainder on division by n. This follows because there are only n remainders possible on division by n.
- 6. The decimal expansion of any rational number contains a repeating sequence of digits (they might be all zeros). For example, 359/495 = 0.7252525…, 7/3 = 2.333…, and 2/5 = 0.4000…. To see this, let m/n be a rational number. Divide m by n until all the digits of m are used up. This gets us to the decimal point. Now continue the division by n for n + 1 more steps. This gives us n + 1 remainders. Since there are only n remainders possible on division by n, the pigeonhole principle tells us that one of remainders will be repeated. So the sequence of remainders between the repeated remainders will be repeated forever. This cause the corresponding sequence of digits in the decimal expansion to be repeated forever.

Simple Ciphers

- Bijections and inverse functions play an important role when working with systems (called ciphers) to encipher and decipher information. In the following, for ease of discussion, we'll use the 26 letters of the lowercase alphabet by the set $N_{26} = \{0, 1, 2, \ldots, 25\}$, where we identify a with 0, b with 1, and so on.
- A simple cipher to transfer a string of text by means of a simple translation of the characters:
  - E.g., the message 'abcd' translated by 5 letters becomes 'fghi', … …

    $f(x) = (x + 5) \bmod 26$

    $f^{-1}(x) = (x - 5) \bmod 26$
- The cipher above is called an *additive* cipher. An additive cipher is an example of a *monoalphabetic* cipher, which is a cipher that always replaces any character of the alphabet by the same character from the cipher alphabet.
- A *multiplicative* cipher is a monoalphabetic cipher that translate each letter by using a multiplier.
  - E.g., $g(x) = 3x \bmod 26$, we can check that g is a bijection by exhaustive checking. Or, it's more easier to use (2.6) on page 103: since gcd(3, 26) = 1 it follows that g is a bijection.

    What is deciphering? : using (2.6) again, since we can write gcd(3, 26) = 1 = 3(9) + 26(-1), since g(0) = 0, so we can define g^{-1} as

    $g^{-1}(x) = 9x \bmod 26$

The Mod Function and Fixed Points

- An *affine* cipher is a monoalphabetic cipher that translates each letter by using two kinds of translation.
  - E.g., we can start with a pair of keys (M, A) and transform a letter by first applying the additive cipher with key A to get an intermediate letter. Then apply the multiplicative cipher with key M to that letter to obtain the desired letter.
  - E.g., we might use the pair of keys (5, 3) and define f as

    $f(x) = 3((x + 5) \bmod 26) \bmod 26 = (3x + 15) \bmod 26$.

    By using (2.6) on page 103, f is a bijection because gcd(3, 26) = 1. So we can also decipher messages with f^{-1}, which we can construct using (2.6) as

    $f^{-1}(x) = (9x + 7) \bmod 26$.

- Some ciphers leave one or more letters fixed.
  - E.g., an additive cipher that translates by a multiple of 26 will leave all letters fixed.
  - A multiplicative cipher always sends 0 to 0, so one letter is fixed.
  - What about an affine cipher of the form $f(x) = (ax + b) \bmod 26$?
  - When can we be sure that no letters are fixed? i.e., when can we be sure that $f(x) \neq x$ for all $x \in N_{26}$?
- The Mod Function and Fixed Points
  - Let $n > 1$ and let $f: N_n \rightarrow N_n$ be defined as follows, where a and b are integers.

    $$f(x) = (ax + b) \bmod n.$$

    Then f has no fixed points (i.e., f changes every letter of an alphabet) if and only if gcd(a – 1, n) does not divide b.

Simple Ciphers

- The Mod Function and Fixed Points
  - Let n > 1 and let f: Nn → Nn be defined as follows, where a and b are integers.

    $$f(x) = (ax + b) \bmod n.$$

    Then f has no fixed points (i.e., f changes every letter of an alphabet) if and only if gcd(a – 1, n) does not divide b.

- Simple Ciphers
  - The function f(x) = (3x + 5) mod 26 does not have any fixed points because gcd(3 – 1, 26) = gcd(2, 26) = 2, and 2 does not divide 5. It's nice to know that we don't have to check all 26 values of f.

    On the other hand, the function f(x) = (3x + 4) mod 26 has fixed points because gcd(3 – 1, 26) = 2, and 2 divides 4. Here we can find that f(11) = 11 and f(24) = 24. So in terms of our association of letters with numbers we would have f(l) = l and f(y) = y.

- For any cipher we use, we always ask:
  - Is it a bijection?
  - What is the range of values for the keys?
  - Is it hard to decipher an intercepted message?

Hash Functions

- Given a key, find the table entry containing the key without searching.

  This may seem impossible at first glance. But let's consider a way to use a function to map each key directly to its table location.

- Definition of Hash Function
  - A *hash function* is a function that maps a set S of keys to a finite set of table indexes, which we'll assume are 0, 1, …, n – 1. A table whose information is found by a hash function is called a *hash table*.

- For example, let S be the set of three-letter abbreviations for the months of the year. We might define a hash function f: S → {0, 1, … 11} in the following way.

  f(XYZ) = (ord(X) + ord(Y) + ord(Z)) mod 12.

  where ord(X) denotes the integer value of the ASCII code for X. (The ASCII values for A to Z and a to z are 65 to 90 and 97 to 122, respectively.) For example, we'll compute the value for the key Jan.

  f(Jan) = (ord(J) + ord(a) + ord(n)) mod 12
  = (74 + 97 + 110) mod 12
  = 5.

  Most programming languages have efficient implementations of the ord and mod functions, so hash functions constructed from them are quite fast. Here is the listing of all the values of f.

Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
5	5	0	3	7	1	11	9	8	6	7	4

  Notice the function f is not injective. For example, f(Jan) = f(Feb) = 5. So if we use f to construct a hash table, we can't put the information for January and February at the same address.

Hash Functions (Cont.)

- Collisons

  If a hash function is injective, then it maps every key to the index of the hash table where the information is stored and no searching is involved. Often this is not possible. When two keys map to the same table index, the result is called a *collision*. So if a hash function is not injective, it has collisions. The previous example hash function has collisions f(Jan) = f(Feb) and f(May) = f(Nov).

  When collisions occur, we store the information for one of the keys in the common table location and must find some other location for the other keys. There are many ways to find the location for a key that has collided with another key. One technique is called *linear probing*. With this technique the program searches the remaining locations in a "linear" manner.

  For example, if location k is the collision index, then the following sequence of table locations is searched

  $$(k + 1) \bmod n,\ \ (k + 2) \bmod n,\ \ldots,\ (k + n) \bmod n.$$

  In constructing the table in the first place, these locations would be searched to find the first open table entry. Then the key would be placed in that location.

Hash Functions (Cont.)

- A Hash Table

  We'll use the sample hash function f to construct a hash table for the months of the year by placing the three-letter abbreviations in the table one by one, starting with Jan and continuing to Dec. We'll use linear probing to resolve collisions that occur in the process. For example, since f(Jan) = 5, we place Jan in position 5 of the table. Next, since f(Feb) = 5 and since position 5 is full, we look for the next available position and place Feb in position 6. Continuing in this way, we eventually construct the following hash table, where entries in parentheses need some searching to be found.

0	1	2	3	4	5	6	7	8	9	10	11
Mar	Jun	(Nov)	Apr	Dec	Jan	(Feb)	May	Sep	Aug	(Oct)	Jul

  There are many questions. Can we find an injection so there are no collisions? If we increased the size of the table, would it give us a better chance of finding an injection? If table size is increased, can we scatter the elements so that collisions can be searched for in less time?

Hash Functions (Cont.)

- Probe Sequences

  Linear probing that looks at locations one step at a time may not be the best way to resolve collisions for some kinds of keys. An alternative is to try linear probing with a "gap" between table locations in order to "scatter" or "hash" the information to different parts of the table. The idea is to keep the number of searches to a minimum. Let g be a gap, where $1 \leq g < n$. Then the following sequence of table locations is searched in case a collision occurs at location k:

  $$(k + g) \bmod n,\ (k + 2g) \bmod n, \ldots, (k + ng) \bmod n.$$

  Some problems can occur if we're not careful with our choice of g. For example, suppose n = 12 and g = 4. Then the probe sequence can skip some table entries. For example, if k = 7, the above sequence becomes

  11, 3, 7, 11, 3, 7, 11, 3, 7, 11, 3, 7.

  So we would miss table entries 0, 1, 2, 4, 5, 6, 8, 9, and 10. Let's try another value for g. Suppose we try g = 5, Then we obtain the following probe sequence starting at k = 7:

  0, 5, 10, 3, 8, 1, 6, 11, 4, 9, 2, 7.

  In this case we cover the entire set {0, 1, …, 11}. In other words, we're defined a bijection $f: N_{12} \rightarrow N_{12}$ by $f(x) = 5x \bmod 12$. Can we always find a probe sequence that hits all the elements of {0, 1, …, n – 1}? Happily, the answer is yes. Just pick g and n so that they are relatively prime, gcd(g, n) = 1. For example, if we pick n to be a prime number, then (g, n) = 1 for any g in the interval $1 \leq g < n$. That's why table sizes are often prime numbers, even though the data set may have fewer entries than the table size.