

Edited: 8:42am, April 19, 2012

---

DRAFT—DRAFT—DRAFT—DRAFT

---

# Discrete Mathematics<sup>1</sup>

W. Ted Mahavier < Clark, May, Shannon

Lamar University

---

<sup>1</sup>If each of  $X$  and  $Y$  is a person, then the relation  $X < Y$  indicates that the notes originated with  $Y$  and were subsequently modified by  $X$ , who takes full responsibility for the current version. While  $Y$  is credited with the genesis of the notes, s/he makes no claim to the accuracy of the current version which may or may not reflect her/his original version. For these notes I took authored notes from each of the authors and mangled them to meet my own prejudices. See introduction for further detail and credits.

# Contents

|   |           |
|---|-----------|
| <b>To the Student</b>   | <b>iv</b> |
| <b>1 Sets</b>   | <b>1</b>  |
| 1.1 Project: More on $\oplus$ and $\times$ . . . . .  | 5         |
| <b>2 Functions</b>  | <b>6</b>  |
| 2.1 Project: Composition . . . . .  | 11        |
| <b>3 Counting</b>   | <b>13</b> |
| 3.1 Project: Unordered Samples with Repetition . . . . .  | 21        |
| <b>4 Induction and Recursion</b>  | <b>23</b> |
| 4.1 Project: More Induction . . . . .   | 27        |
| <b>5 Equivalence Relations</b>  | <b>29</b> |
| 5.1 Project: More on Equivalence Classes . . . . .  | 32        |
| <b>6 Euler Paths and Circuits</b>   | <b>34</b> |
| <b>7 Traveling Salesman Problems</b>  | <b>48</b> |
| 7.1 Project: Greed Doesn't Pay . . . . .  | 53        |
| <b>8 Spanning Trees</b>   | <b>54</b> |
| 8.1 Project: Intuition for a Proof that Kruskal's Algorithm Works                               | 58        |
| <b>9 Finite State Machines</b>  | <b>60</b> |
| 9.1 Project: Machine Languages are closed under complement,<br>union and intersection . . . . . | 66        |

---

|  |           |
|--|-----------|
| <b>10 Languages and Machine Minimization</b>           | <b>67</b> |
| 10.1 Project: Languages that are not Regular . . . . . | 76        |

## **To the Student**

### **How this class works**

Every semester I collect written evaluations from my classes and the students who have taken this course have been overwhelmingly positive in their comments. While challenged by the material and the method, they enjoyed the class. If you'd like, I'll even read some random comments from the last semester I taught it. I'll print them out and let you randomly select a number and I'll read that comment.

The reason I open with a discussion of my evaluations, is because this class will be taught in a way that is (most likely) different from any mathematics classes you have encountered in the past. Most of the class will be devoted to students working problems at the board and half of your grade will be determined by the amount of mathematics that you produce in this class. I use the word produce because it is my belief that the best way to learn mathematics is by doing mathematics.

Therefore, just as I learned to ride a bike by getting on and falling off, I expect that you to learn mathematics by attempting it and (occasionally) falling off! You will have a set of notes (provided by me) that you will turn into a book by working through the problems. I urge you to seriously consider the value of becoming an independent thinker who tackles doing mathematics (and everything else in life) on your own, rather than waiting for someone else to show you how to do things. Jump in, it will be fun.

### **A common pitfall**

There are two ways in which students often approach my classes. The first is to say, "I'll wait and see how this works and then see if I like it and put some problems up later in the semester after I catch on." Think of it as a 40 yard dash. Do you really want to wait and see how fast the other runners are before you start running? If you try every night to do the problems, then either you will get a problem (YEA!) and be able to put it on the board with

pride and satisfaction or you will struggle with the problem and learn a lot in your struggle. If no one else has it, you can show your attempt which may give you further ideas on how to solve it. If someone else puts it on the board then you will be able to ask questions and help yourself and others understand it, as you say to yourself, “Ahhhh, now I see where I went wrong and now I can do this one and a few more for next class.” If you do not try problems each night, then you will watch the student put the problem on the board, but perhaps will not quite catch all the details and then when you study for the tests or try the next problems you will have only a loose idea of how to tackle such problems. Basically, you have seen it only once in this case. The first student saw it once when s/he tackled it on his own, again when either s/he put it on the board or another student presented it, and then a third time when s/he studies for the next test or quiz. Hence the difference between these two approaches is the difference between participating and watching a movie. I hope that each of you will tackle this course with the attitude that you will learn this material and thus will both enjoy and benefit from the class.

### **Boardwork**

Because the board work constitutes half of your grade, let's put your mind at ease regarding this part of the class. First, by coming to class today you have a 60% on board work. Every problem you present pushes that grade higher. You may come see me anytime for an indication of what grade your current level of participation will earn you at the end of the semester. Here are some rules and guidelines associated with the board work. I will call for volunteers every day and will pick the person with the least presentations to present a given problem. You may inform me that you have a problem in advance (which I appreciate), but the problem still goes to the person with the least presentations on the day I call for a solution. Ties are broken either randomly (at the beginning) or by test grades (lower test grades taking priority). A student who has not gone to the board on a given day will be given precedence over a student who has gone to the board that day. To “present” a problem at the board means to have written the problem statement up, to have written a correct solution using complete mathematical sentences, and to have answered all students' questions regarding the problem. Since you will be communicating with other students on a regular basis, here are several guidelines that will help you. The whole class is on your side because everyone wants to see the problem presented correctly.

When you speak, don't use the words “obvious,” “stupid,” or “trivial.”

Don't attack anyone personally or try to intimidate anyone. Don't get mad or upset at anyone (and if you do, try to get over it quickly). Don't be upset when you make a mistake - brush it off and learn from it. Don't let anything go on the board that you don't fully understand. Don't say to yourself, "I'll figure this out at home." Don't use concepts we have not defined. Don't use or get examples or solutions from other books. Don't work together without acknowledging it as the board. Don't try to put up a problem you have not written up.

Do prepare arguments in advance. Do be polite and respectful. Do learn from your mistakes. Do ask questions such as, "Can you tell me how you got the third line?" Do let people answer when they are asked a question. Do refer to earlier results and definitions by number when possible. "By Definition 29"

### **How to study**

1. Read over your notes from class that day.
2. Make a list of questions to ask at the beginning of the next class.
3. Review the recent problems.
4. Work on *several* new problems.
5. Write up as many solutions as you can so that you can simply copy your well-written solutions onto the board the next day.

Some problems are hard. If you don't get one, don't give up on it. Move on to another problem and come back to that problem later. The problems worth solving in life are not solved in five minutes.

### **What is discrete mathematics?**

Discrete mathematics is mathematics that is not indiscrete. Such mathematics would never participate in mosh pits, raves, pajama day, or rock concerts. LOL.

Seriously, the word discrete means finite in this setting. Thus, discrete mathematics means the study of the topics in mathematics that can be approached without using infinite sets. Still, you'll see a few infinite sets in the course.

# Chapter 1

## Sets

Every mathematics, science, and engineering course uses sets, the basic building blocks of mathematics, so we start here. One could make a good argument for starting even earlier, with numbers. Since numbers are usually treated in courses titled analysis and foundations, we'll start with sets.

The words **element**, **set**, and **universe** are undefined. By a **set** we mean a collection of **elements** taken from some fixed **universe** of elements that we will call **U**.

**Axiom 1.** *If **A** is a set, then for each element  $x$  in **U** either  $x$  is in **A** (written  $x \in \mathbf{A}$ ) or  $x$  is not in **A** (written  $x \notin \mathbf{A}$ ).*

**Definition 1.** *If **A** and **B** are sets and every element of **A** is also in **B** then we say that **A** is a subset of **B** and write  $\mathbf{A} \subseteq \mathbf{B}$ .*

**Axiom 2. Set Equality** *If **A** and **B** are sets that have the same elements (that is,  $\mathbf{A} \subseteq \mathbf{B}$  and  $\mathbf{B} \subseteq \mathbf{A}$ ), then we say that **A** and **B** are the same set and write  $\mathbf{A} = \mathbf{B}$ .*

**Axiom 3. Set Specification** *If **P** is a property that each element in **U** either has or does not have, then there is a set, denoted by*

$$\{x \in U \mid x \text{ has property } \mathbf{P}\},$$

*whose members are exactly those elements in **U** having property **P**.*

For example, the set

$$\mathbf{S} = \{(x, y) \mid x \text{ and } y \text{ are real numbers and } 2x + y^2 = 7\},$$

is a set of points in the plane, a parabola.

Using these two axioms the set we referred to earlier as the **universal set** may be written as

$$\mathbf{U} = \{x \in U \mid x = x\}$$

where we have taken the property  $\mathbf{P}$  to be “ $x = x$ ” which is true of every element.

**Definition 2.** The *empty set* is the set having no members,

$$\emptyset = \{x \in U \mid x \neq x\}.$$

We can construct new sets from old using the Set Specification Axiom.

**Definition 3.** Let  $A$  and  $B$  be sets. The *intersection* of  $\mathbf{A}$  and  $\mathbf{B}$  is the set

$$\mathbf{A} \cap \mathbf{B} = \{x \in U \mid x \in \mathbf{A} \text{ and } x \in \mathbf{B}\}.$$

**Definition 4.** Let  $A$  and  $B$  be sets. The *union* of  $\mathbf{A}$  and  $\mathbf{B}$  is the set

$$\mathbf{A} \cup \mathbf{B} = \{x \in U \mid x \in \mathbf{A} \text{ or } x \in \mathbf{B}\}.$$

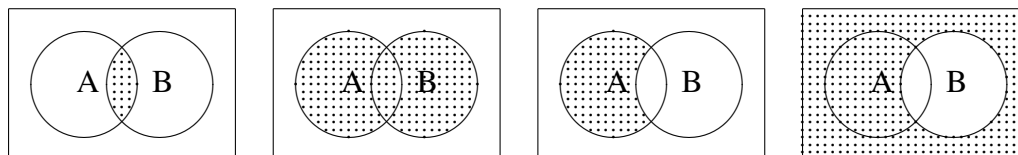
**Definition 5.** Let  $A$  be a set. The *complement* of  $\mathbf{A}$  is the set

$$\sim \mathbf{A} = \{x \in U \mid x \notin \mathbf{A}\}.$$

**Definition 6.** Let  $A$  and  $B$  be sets. The *difference* between  $\mathbf{A}$  and  $\mathbf{B}$  is the set

$$\mathbf{A} \sim \mathbf{B} = \mathbf{A} \cap (\sim \mathbf{B}).$$

The following illustrations are **Venn diagrams** for the sets just defined.



$\mathbf{A} \cap \mathbf{B}$

$\mathbf{A} \cup \mathbf{B}$

$\mathbf{A} \sim \mathbf{B}$

$\sim \mathbf{B}$

Different expressions might represent the same set as illustrated by the next example.

**Example 1.** For every choice of sets  $\mathbf{A}$ ,  $\mathbf{B}$  and  $\mathbf{C}$ ,

$$\mathbf{A} \cap (\mathbf{B} \cup \mathbf{C}) \subseteq (\mathbf{A} \cap \mathbf{B}) \cup (\mathbf{A} \cap \mathbf{C}).$$

Together, Example 1 and Problem 1 show that

$$\mathbf{A} \cap (\mathbf{B} \cup \mathbf{C}) = (\mathbf{A} \cap \mathbf{B}) \cup (\mathbf{A} \cap \mathbf{C}).$$

**Problem 1.** Show that  $(\mathbf{A} \cap \mathbf{B}) \cup (\mathbf{A} \cap \mathbf{C}) \subseteq \mathbf{A} \cap (\mathbf{B} \cup \mathbf{C})$ .

Once problem 1 is complete we may conclude by the Set Equality Axiom, that  $\mathbf{A} \cap (\mathbf{B} \cup \mathbf{C}) = (\mathbf{A} \cap \mathbf{B}) \cup (\mathbf{A} \cap \mathbf{C})$  and the proof is complete.

You are already familiar with operations on numbers such as addition, subtraction, multiplication, and division ( $+$ ,  $-$ ,  $*$ ,  $\div$ ). We have introduced the operations on sets such as intersection, union, and difference ( $\cap$ ,  $\cup$ , and  $\sim$ ). Soon, we will introduce more set operations, including  $\times$ ,  $\oplus$  and  $\circ$ . All of these number and set operations are referred to as **binary operations** because each one takes two inputs.

For the next few problems, illustrate each of the following identities with Venn Diagrams and then write down a proof using the Set Equality Axiom.

**Problem 2.** *The Commutative Laws*

$$\mathbf{A} \cap \mathbf{B} = \mathbf{B} \cap \mathbf{A}$$

$$\mathbf{A} \cup \mathbf{B} = \mathbf{B} \cup \mathbf{A}$$

**Problem 3.** *The Associative Laws:*

$$\mathbf{A} \cap (\mathbf{B} \cap \mathbf{C}) = (\mathbf{A} \cap \mathbf{B}) \cap \mathbf{C}$$

$$\mathbf{A} \cup (\mathbf{B} \cup \mathbf{C}) = (\mathbf{A} \cup \mathbf{B}) \cup \mathbf{C}$$

**Problem 4.** *The Distributive Laws*

$$\mathbf{A} \cup (\mathbf{B} \cap \mathbf{C}) = (\mathbf{A} \cup \mathbf{B}) \cap (\mathbf{A} \cup \mathbf{C})$$

$$\mathbf{A} \cap (\mathbf{B} \cup \mathbf{C}) = (\mathbf{A} \cap \mathbf{B}) \cup (\mathbf{A} \cap \mathbf{C})$$

**Problem 5.** *The Absorption Laws*

$$\mathbf{A} \cup (\mathbf{A} \cap \mathbf{B}) = \mathbf{A}$$

$$\mathbf{A} \cap (\mathbf{A} \cup \mathbf{B}) = \mathbf{A}$$

For the next problem, it is helpful to make the observation about the empty set that *any* statement starting with “If  $x \in \emptyset \dots$ ” is a true statement. For example, the statement “If  $x \in \emptyset$  then  $x$  is a seven-headed dog.” is a true statement. Why? Because for statement to be false, the hypothesis must be true and the conclusion false. For this statement there is no way for the hypothesis to be true since there are no elements in the empty set.

**Problem 6.** *The Identity Laws*

$$\mathbf{A} \cup \emptyset = \mathbf{A}$$

$$\mathbf{A} \cap \mathbf{U} = \mathbf{A}$$

**Problem 7.** *The Inverse Laws*

$$\mathbf{A} \cup \sim \mathbf{A} = \mathbf{U}$$

$$\mathbf{A} \cap \sim \mathbf{A} = \emptyset$$

**Problem 8.** *DeMorgan's Laws*

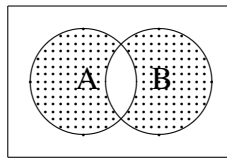
$$\sim (\mathbf{A} \cap \mathbf{B}) = \sim \mathbf{A} \cup \sim \mathbf{B}$$

$$\sim (\mathbf{A} \cup \mathbf{B}) = \sim \mathbf{A} \cap \sim \mathbf{B}$$

**Definition 7.** *The symmetric difference between sets  $\mathbf{A}$  and  $\mathbf{B}$  is defined by*

$$\mathbf{A} \oplus \mathbf{B} = (\mathbf{A} \cup \mathbf{B}) \sim (\mathbf{A} \cap \mathbf{B}).$$

Illustrated via a Venn diagram,



**Problem 9.**  $\mathbf{A} \oplus \mathbf{B} = (\mathbf{A} \sim \mathbf{B}) \cup (\mathbf{B} \sim \mathbf{A})$

**Problem 10.**  $\mathbf{A} \oplus (\mathbf{B} \oplus \mathbf{C}) = (\mathbf{A} \oplus \mathbf{B}) \oplus \mathbf{C}$

**Example 2.** *Does*

$$\mathbf{A} \oplus (\mathbf{B} \cup \mathbf{C}) = (\mathbf{A} \oplus \mathbf{B}) \cup (\mathbf{A} \oplus \mathbf{C})?$$

*Restated, does  $\oplus$  distribute over  $\cup$ ?*

**Definition 8.** *If  $\mathbf{A}$  and  $\mathbf{B}$  are sets, the **Cartesian product** of  $\mathbf{A}$  and  $\mathbf{B}$ , denoted by  $\mathbf{A} \times \mathbf{B}$ , consists of all ordered pairs  $(x, y)$  where  $x \in \mathbf{A}$  and  $y \in \mathbf{B}$ .*

*Restated,*

$$\mathbf{A} \times \mathbf{B} = \{(x, y) \mid x \in \mathbf{A} \text{ and } y \in \mathbf{B}\}.$$

**Problem 11.** *Prove or give a counter-example:*

$$\mathbf{A} \cap (\mathbf{B} \times \mathbf{C}) = (\mathbf{A} \cap \mathbf{B}) \times (\mathbf{A} \cap \mathbf{C})$$

### 1.1 Project: More on $\oplus$ and $\times$

For the following statements, either prove that it is true using the set equality axiom, or prove that it is false by exhibiting a counterexample.

**Problem 12.**  $A \oplus (B \cap C) = (A \oplus B) \cap (A \oplus C)$

**Problem 13.**  $A \cap (B \oplus C) = (A \cap B) \oplus (A \cap C)$

**Problem 14.**  $\sim (A \oplus B) = \sim A \oplus \sim B$

**Problem 15.**  $A \oplus B = \sim A \oplus \sim B$

**Problem 16.**  $A \cup (B \oplus C) = (A \cup B) \oplus (A \cup C)$

**Problem 17.**  $A \times (B \cup C) = (A \times B) \cup (A \times C)$

**Problem 18.**  $A \cup (B \times C) = (A \cup B) \times (A \cup C)$

## Chapter 2

### Functions

Every mathematics, science, and engineering course makes heavy use of functions. Having developed a deep understanding of sets, we now move to functions which are defined in terms of sets of ordered pairs. The first two problems make precise the notion of ordered pairs.

**Problem 19.** *Suppose that each of  $a$ ,  $b$ ,  $c$  and  $d$  is an element. For each statement, determine if it is true or false.*

1. *If  $\{a, b\} = \{c, d\}$  then  $a = c$  and  $b = d$*
2. *If  $a = c$  and  $b = d$  then  $\{a, b\} = \{c, d\}$*

**Definition 9.** *If each of  $a$  and  $b$  is an element, then by the **ordered pair**  $(a, b)$  we mean the set  $\{\{a\}, \{a, b\}\}$ . The elements  $a$  and  $b$  are known as the first and second coordinates, respectively.*

Problem 19 hinted at the property that we want from our new definition for ordered pairs. We want that  $(a, b) = (c, d)$  if and only if  $a = c$  and  $b = d$ . The next problem allows you to show that the definition for ordered pairs,  $(a, b) = \{a, \{a, b\}\}$ , satisfies this property and therefore is a valid definition.

**Problem 20.** *Suppose each of  $a$ ,  $b$ ,  $c$  and  $d$  is an element. Use Definition 9 to prove the following two statements:*

1. *If  $(a, b) = (c, d)$  then  $a = c$  and  $b = d$ .*
2. *If  $a = c$  and  $b = d$  then  $(a, b) = (c, d)$ .*

**Problem 21.** *Killed this problem – spacer to keep students numbers right...*

Recall from Definition 8 that the Cartesian Product of  $X$  and  $Y$  is  $X \times Y = \{(x, y) : x \in X, y \in Y\}$ .

**Definition 10.** Let  $X$  and  $Y$  be sets. A **relation** on  $X \times Y$  is any subset of  $X \times Y$ . A **function** on  $X \times Y$  is a relation on  $X \times Y$  where no two elements have the same first coordinates. The set of all first coordinates is called the **domain** of the function and the set of all second coordinates is called the **range** of the function.

**Example 3.** Let  $A = \{x \in \mathbb{R} : -3 \leq x \leq 3\}$ . The set  $C = \{(x, y) : x^2 + y^2 = 9\}$  is a relation on  $A \times A$  but not a function on  $A \times A$ . Do the words “domain” and “range” make sense in this context? Why or why not?

**Example 4.** The set  $f = \{(x, y) : x \in \mathbb{R} \text{ and } y = 2x - 3\}$  is a function on  $\mathbb{R} \times \mathbb{R}$ . How do we prove this?

**Example 5.** An archer has 10 arrows and a target, divided into 4 concentric circles. The arrows form one set  $A$ , and the areas between the circles along with the area inside the smallest circle (the bulls-eye) form another set  $B$ . Discuss possible functions, one-to-one and onto.

**Problem 22.** Let  $A = \{1, 2, 3\}$  and  $B = \{\square, \diamond, \triangle\}$ . Which of the following are relations on  $A \times B$ ? Which are functions on  $A \times B$ ?

1.  $\{(1, \square), (1, \triangle), (2, \diamond)\}$
2.  $\{(3, \square), (1, \triangle), (2, \triangle)\}$
3.  $\{((1, 1), \square), ((1, 2), \triangle), ((2, 1), \diamond), ((2, 2), \diamond)\}$

**Problem 23.** Consider the function  $f$  defined by

$$f = \{(x, y) : x \in \mathbb{R} \sim \{1\} \text{ and } y = \frac{2x}{x-1}\}.$$

State the domain and the range of this function.

**Problem 24.** Consider

$$f = \{(x, y) : x \in \mathbb{R} \text{ and } y = \sin(2x) + \sin^2(x) + \cos^2(x)\} \text{ and}$$

$$g = \{(x, y) : x \in \mathbb{R} \text{ and } y = 2 \sin(x) \cos(x) + 1\}.$$

Are these the same function?

Since we often think of a function  $f$  on  $X \times Y$  as a rule assigning elements of  $X$  to elements of  $Y$ , we often write  $f : X \rightarrow Y$ . When  $(x, y)$  is an element of  $f$ , we write,  $f(x) = y$  and say that  $f$  **maps**  $x$  to  $y$ . When we use this notation, it is common for  $Y$  to contain the range of  $f$ . In such cases, we call  $Y$  the **codomain** of  $f$ .

**Example 6.** Let  $f = \{(x, y) : y = x^2 + 3\}$ . We might write the same function as  $f : \mathbb{R} \rightarrow \mathbb{R}$  where  $f(x) = x^2 + 3$ . Is  $f$  onto  $\mathbb{R}$ ? Is  $f$  onto  $\{y : y \geq 3\}$ ? We would call  $\mathbb{R}$  a codomain of  $f$ .

**Problem 25.** Suppose  $f(x) = \frac{x^2 - 5x + 6}{x - 3}$  and  $g(x) = x - 2$ . Does  $f = g$ ? What is the largest allowable domain for  $f$ ? For  $g$ ?

**Problem 26.** Let  $S$  denote the set of all subsets of a set  $U$ , and choose two particular sets  $A, B \in S$ . Define functions  $f : S \rightarrow S$  and  $g : S \rightarrow S$  so that for every subset  $X \in S$ ,  
 $f(X) = X \cap (A \sim B)$  and  $g(X) = (X \cap A) \sim (X \cap B)$ . What is the domain of  $f$ ? Of  $g$ ? Does  $f = g$ ?

**Definition 11.** For real valued functions  $f : \mathbb{R} \rightarrow \mathbb{R}$  and  $g : \mathbb{R} \rightarrow \mathbb{R}$  define a new function  $f + g : \mathbb{R} \rightarrow \mathbb{R}$ , called the **sum** of  $f$  and  $g$ , by the rule  
 $(f + g)(x) = f(x) + g(x)$  for  $x \in \mathbb{R}$ .

**Problem 27.** Suppose each of  $f : \mathbb{R} \rightarrow \mathbb{R}$  and  $g : \mathbb{R} \rightarrow \mathbb{R}$  are functions and prove that  $f + g = g + f$  by showing that  $(f + g)(x) = (g + f)(x)$  for all  $x \in \mathbb{R}$ . This shows that addition of functions is commutative.

**Definition 12.** If  $f : X \rightarrow Y$  is a function, then  $f$  is **one-to-one** if no two elements of  $f$  have the same second coordinate and different first coordinates. Restated, no two elements of  $X$  can map to the same element of  $Y$ . We say that  $f$  is **onto the set**  $Y$  if for each element  $y \in Y$  there is some element  $x \in X$  such that  $f(x) = y$ . A function that is both one-to-one and onto is **bijective**.

**Problem 28.**  $A = B = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ . Table 2.1 defines a function  $f : A \rightarrow B$ . For each  $x \in A$ , the value of  $f(x)$  is written below  $x$ . Is  $f$  one-to-one? Is  $f$  onto  $A$ ? Is  $f$  a bijection?

|      |   |   |   |   |   |   |   |   |   |
|------|---|---|---|---|---|---|---|---|---|
| x    | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| f(x) | 5 | 7 | 9 | 3 | 1 | 2 | 6 | 4 | 8 |

Table 2.1: A function,  $f$

**Problem 29.** Let  $f = \{(x, y) : x \in \mathbb{R} \sim \{2\} \text{ and } y = \frac{x}{x - 2}\}$ .

1. Is  $f$  a one-to-one function?
2. Is  $f$  onto the set  $\mathbb{R}$ ?

**Problem 30.** Let  $f : \mathbb{R} \rightarrow \mathbb{R}$  defined by  $f(x) = \sqrt[3]{x - 1}$ .

1. Is  $f$  a one-to-one function?
2. Is  $f$  onto  $\mathbb{R}$ ?

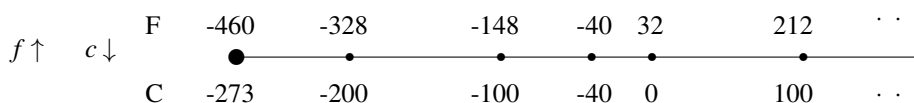


Figure 2.1: Thermometer showing both Fahrenheit and Celcius temperature scales

By Definition 10, every function  $f : X \rightarrow Y$  is onto its range since the range of  $f$  is the set of all  $y$  such that  $(x, y) \in f$ . Restated, the range of  $f$  is  $\{y : (x, y) \in f\} = \{f(x) : x \in X\}$

When you board an airplane (a function) that “maps” you from Houston to Chicago, you will definitely want to board another airplane (the inverse function) that “maps” you back home! Countless people have been lost because they built a time travel machine but forgot to build the inverse time machine!

**Definition 13.** Suppose that  $g : \mathbf{A} \rightarrow \mathbf{B}$  and  $f : \mathbf{B} \rightarrow \mathbf{C}$ . We define the **composition** of  $f$  and  $g$  (denoted by  $f \circ g$ ) to be the function from  $\mathbf{A}$  to  $\mathbf{C}$  satisfying:

$$(f \circ g)(x) = f(g(x))$$

**Problem 31.** Figure 2.1 shows the relationship between the Fahrenheit and Celcius temperature scales. Write down a function  $f$  that converts Celcius to Fahrenheit, and a function  $c$  that converts Fahrenheit back to Celcius. Verify that  $f(c(F)) = F$  for every  $F \in \mathbb{R}$  and that  $c(f(C)) = C$  for every  $C \in \mathbb{R}$ .

**Definition 14.** Given a function  $f : X \rightarrow Y$ , the relation  $f^{-1}$  is defined by  $f^{-1} = \{(y, x) : (x, y) \in f\}$ .

The set  $f^{-1}$  might not be a function. Problems 32 and 33 tell us exactly when  $f^{-1}$  is a function.

**Problem 32.** Let  $f : X \rightarrow Y$  be onto  $Y$ . Show that if  $f$  is one-to-one then there is a function  $g : Y \rightarrow X$  so that  $g(f(x)) = x$  for all  $x \in X$ .

**Problem 33.** Let  $f : X \rightarrow Y$  be onto  $Y$ . Show that if there is a function  $g : Y \rightarrow X$  so that  $g(f(x)) = x$  for all  $x \in X$  then  $f$  is one-to-one.

**Problem 34.** For each function  $f$  determine if  $f^{-1}$  is a function.

1.  $f = \{(x, y) : x \in \mathbb{R} \text{ and } y = x(x - 1)\}$
2. The archer function in Example 5 a few pages back

3.  $f : \mathbb{R} \rightarrow \mathbb{R}$  defined by  $f(x) = \sqrt[3]{x-1}$

**Problem 35.** Let  $f : \mathbb{R} \rightarrow \mathbb{R}$  with  $f(x) = 3x - 7$ .

- (i) Prove that  $f$  is onto  $\mathbb{R}$ .
- (ii) Prove that  $f$  is one-to-one.
- (iii) Find a formula for  $f^{-1}(y)$ .
- (iv) Verify that for all  $x \in \mathbb{R}$  we have,  $f^{-1}(f(x)) = x$  and  $f(f^{-1}(x)) = x$ .

**Problem 36.** Show that every non-constant linear function  $l(x) = mx + b$ , with  $m \neq 0$ , is a bijection  $l : \mathbb{R} \rightarrow \mathbb{R}$ . Find a formula for  $l^{-1}(y)$ .

The binary operations of addition and multiplication can be used to combine any two numbers in order to get a new number. The binary operations of intersection, union, difference and symmetric difference can be used to combine two sets in order to form a new set. Addition, subtraction, multiplication and division of functions are also binary operations. Another binary operation on functions is called composition.

In the Table 2.2 below, we have four bijections from  $\{1, 2, 3, 4\}$  onto  $\{1, 2, 3, 4\}$ . The first bijection,  $i$ , is the **identity map** since it maps  $1 \rightarrow 1$ ,  $2 \rightarrow 2$ ,  $3 \rightarrow 3$  and  $4 \rightarrow 4$ .

| i    | p    | q    | r    |
|------|------|------|------|
| 1234 | 1234 | 1234 | 1234 |
| 1234 | 2134 | 1243 | 2143 |

Table 2.2: Bijections

Let's compute  $q \circ r$ .

- 1.  $(q \circ r)(1) = q(r(1)) = q(2) = 2 = p(1)$ ,
- 2.  $(q \circ r)(2) = q(r(2)) = q(1) = 1 = p(2)$ ,
- 3.  $(q \circ r)(3) = q(r(3)) = q(4) = 3 = p(3)$ , and
- 4.  $(q \circ r)(4) = q(r(4)) = q(3) = 4 = p(4)$ .

Therefore,  $q \circ r = p$ .

**Problem 37.** Use the example above and Table 2.2 to fill in the rest of Table 2.3.

|   |   |   |   |   |
|---|---|---|---|---|
| ◦ | i | p | q | r |
| i |   |   |   |   |
| p |   |   |   |   |
| q |   |   |   | p |
| r |   |   |   |   |

Table 2.3: Composition

**Problem 38.** What is this inverse of each of these bijections?

**Problem 39.** Addition is **commutative** because for any two numbers  $a$  and  $b$  we have  $a + b = b + a$ . If for every  $f : \mathbf{A} \rightarrow \mathbf{A}$  and  $g : \mathbf{A} \rightarrow \mathbf{A}$  it is true that

$$f \circ g = g \circ f$$

then we will say that composition of functions is also **commutative**. Either show that composition is commutative or give a counter example by finding two functions for which the statement is not true.

**Problem 40.** Intersection of sets is **associative** since for any sets  $A, B,$  and  $C$  we have  $\mathbf{A} \cap (\mathbf{B} \cap \mathbf{C}) = (\mathbf{A} \cap \mathbf{B}) \cap \mathbf{C}$ . Is composition of functions associative? Restated, is it true that if  $f$  and  $g$  are functions then  $(f \circ g) \circ h = f \circ (g \circ h)$ ?

## 2.1 Project: Composition

**Problem 41.** Show that the composition of injections is an injection by letting  $g : \mathbf{B} \rightarrow \mathbf{C}$  and  $h : \mathbf{A} \rightarrow \mathbf{B}$  be injective functions and showing that  $g \circ h : \mathbf{A} \rightarrow \mathbf{C}$  is injective as well.

**Problem 42.** Show that the composition of surjective functions is surjective.

The word *identity* is used in a number of different contexts in mathematics. The function  $f(x) = x$  is called the *identity* function because it identifies every number with itself. Similarly, 0 is called an *additive identity* because adding 0 to a number does not change the number, so  $0 + x$  is identified with  $x$ . By the same reasoning, 1 is called the multiplicative identity.

**Problem 43.** Let  $\mathbf{S}$  denote the set of all bijections from  $\mathbf{A}$  to itself. According to Problems 41 and 42,  $\circ$  is a binary operation on  $\mathbf{S}$ , that is,  $f \circ g \in \mathbf{S}$  whenever  $f$  and  $g$  are in  $\mathbf{S}$ . We define a special bijection  $i : \mathbf{A} \rightarrow \mathbf{A}$  called the **identity function** on  $\mathbf{A}$  as

$$i(x) = x \text{ for all } x \in \mathbf{A}.$$

Show that  $i$  is an identity for  $\mathbf{S}$ , that is,

(i)  $f \circ i = f = i \circ f$  and

(ii)  $f \circ f^{-1} = i = f^{-1} \circ f$

for all  $f \in \mathbf{S}$ . This shows that  $i$  serves as an identity for composition in the same way that  $0, 1, \emptyset$  work as identities for  $+, \times$  and  $\oplus$ , respectively.

**Problem 44.** Consider the list below of all 6 different bijections from  $\{0, 1, 2\}$  onto itself. For example  $p$  maps  $0 \rightarrow 1, 1 \rightarrow 2$  and  $2 \rightarrow 0$ .

| i   | p   | q   | f   | g   | h   |
|-----|-----|-----|-----|-----|-----|
| 012 | 012 | 012 | 012 | 012 | 012 |
| 012 | 120 | 201 | 021 | 210 | 102 |

Make a composition table for these 6 bijections, illustrating Problems 41, 42 and 43 above.

**Problem 45.**  $\circ$  is said to have the **Left Cancellation Property** if, for any functions  $f : \mathbf{B} \rightarrow \mathbf{C}, g : \mathbf{A} \rightarrow \mathbf{B}$  and  $h : \mathbf{A} \rightarrow \mathbf{B}$ ,

$$f \circ g = f \circ h, \quad \text{implies} \quad g = h.$$

Find a counterexample to show that this is not always true. Then prove that the Left Cancellation Property does hold whenever  $f$  is one-to-one.

## Chapter 3

### Counting

A more formal title for this chapter would be *combinatorics*. Many problems involving probability and statistics require knowing how many elements are in a particular set. Consider poker hands. What is the probability that a hand of five cards has two aces? To answer this, we would divide the number of possible hands with two aces by the total number of five card hands. Of course, we could list all possible five card hands and then count the number of these hands that have two aces. To be more efficient, we need to know how to count the number of hands without listing all of them. So, we will develop strategies for determining the number of elements of certain sets without listing all the elements. Still, I use lists all the time to answer counting questions, so don't hesitate to use lists to help obtain or verify answers.

**Problem 46.** *The standard license plate for a non-commercial vehicle titled in the State of Maryland consists of six characters. The first character must be one of the nine digits 1, 2, ..., 9. Each of the next three characters must be a letter of the alphabet other than i, o, q, or u. Each of the last two characters must be one of the ten digits 0, 1, 2, ..., 9. How many standard license plates for non-commercial vehicles can be issued by the State of Maryland?*

**Problem 47.** *There are six area codes used for Maryland telephone numbers: 227, 240, 301, 410, 443, and 667. Following each area code are a three-digit "prefix" and a four-digit "exchange." The prefix may not be the number 555, or begin with the number 0. How many telephone numbers can be issued for the State of Maryland?*

**Problem 48.** *Andrew, Bob, Carly, and Diane are the only entrants in a prize giveaway. Both prizes are the same model of Play Station. In how many ways can two winners be chosen from them?*

**Problem 49.** *Andrew, Bob, Carly, and Diane are the only entrants in a prize giveaway. The first prize is an Audi TT, and the second is a Ford Focus. No one person is allowed to win both prizes. In how many ways can the prizes be awarded?*

**Problem 50.** *Andrew, Bob, Carly, and Diane are the only entrants in a prize giveaway. The first prize is an Audi TT, and the second is a Ford Focus. It is allowable for the same person to win both prizes. In how many ways can the prizes be awarded?*

The next two theorems simply give names to the tools you probably used to solve the last few problems.

**Theorem 1. Multiplication Principle:** *If each of  $m$  and  $n$  is a positive integer,  $A$  is an  $m$ -element set, and  $B$  is an  $n$ -element set, then  $|A \times B| = mn$  or restated,  $|A \times B| = |A||B|$ . This is sometimes called the Fundamental Principle of Counting.*

**Theorem 2. Generalized Multiplication Principle:** *Suppose that  $n$  is a positive integer and each of  $A_1, A_2, \dots, A_n$  is a finite set. Then  $|A_1 \times A_2 \times A_3 \times \dots \times A_n| = |A_1||A_2| \dots |A_n|$ . This is sometimes called or the Generalized Fundamental Principle of Counting.*

**Example 7.** *This example will illustrate both Multiplication Principles. Suppose you have 4 different pairs of pants, 3 different shirts, and 5 different hats. How many outfits can you make? (Remember, you are a mathematician, so you have no worries about how what you wear actually looks to the rest of the world.) Since each of the 4 pairs of pants can be matched with any of the 3 shirts, you have  $4 \cdot 3 = 12$  ways to pick a pants/shirt combination. Once you pick one of the 12 combinations of pants and shirts, you can pick any of the hats, so you have  $12 \cdot 5 = 60$  choices. We used the Multiplication Principle twice to do this, but of course  $4 \cdot 3 \cdot 5 = 60$  so we could have used the Generalized Multiplication Principle.*

**Problem 51.** *There are 13 cards of each of the four suits in a 52-card deck. How many four-card hands are there containing a card from each of the four suits?*

**Problem 52.** *Sherwoodn't Cars sells five models of car in three colors. How many different cars could you see in Sherwoodn't's lot, ignoring accessories and options?*

**Problem 53.** *A program to produce greeting cards has 100 pictures to choose from and 25 sayings. How many different cards can it produce, assuming one picture and one saying per card? How many assuming two pictures and one saying per card?*

**Problem 54.** Suppose there are four 400-level mathematics courses: MATH 406, 402, 465, and 482, offered in a particular semester, and you want to take two of them. How many options do you have?

**Problem 55.** Consider the algorithm below.

```

Let i = 1
While i < 7 do
  Let j=2
  While j < 6
    Print "Here is an ordered pair", (i,j)
    j = j + 1
  End j While Loop
  i = i + 1
End i While Loop

```

How many ordered pairs would this program write out?

**Problem 56.** Suppose that a certain web site has you choose a username and a password. The username must consist of 10 alphanumeric characters; the password must consist of seven alphanumeric characters, the last of which must be numeric and the first of which must be alphabetical.

1. How many usernames are possible if they are not case-sensitive?
2. How many passwords are possible if they are not case-sensitive?
3. How many usernames are possible if they are case-sensitive?
4. How many passwords are possible if they are case-sensitive?

**Definition 15.** A set  $M$  is **finite** if there is a positive integer  $n$  so that  $M$  has  $n$  elements and does not have  $n + 1$  elements. If the set  $M$  has  $n$  elements but does not have  $n + 1$  elements, then we write  $|M| = n$ . A set is **infinite** if it is not finite.

In counting the number of subsets of a particular set, two questions that may help are: Does order matter? Is repetition (replacement) allowed? Because “subset” implies that order should not matter ( $\{1, 2, 3\} = \{2, 3, 1\}$ ) we sometimes refer to the set we are drawing from as the *population* and the sets we are picking as *samples*. Using this language, we can restate Problem 48 as “How many unordered samples of size two can be chosen without replacement (or “without repetition”) from the four-element “population” (Andrew, Bob, Carly, Diane)?” In Problem 49, we sought an ordered sample of size two without replacement (repetition) from the same population.

Finally, in Problem 50 we desired an ordered sample of size two “with replacement” (or “with repetition”).

### Ordered Samples with Repetition Allowed (n-tuples)

The expression (3,-5) is an example of an ordered pair;  $(-1, 0, \frac{1}{2})$  is an ordered triple; and  $(7, \frac{1}{4}, \pi, z)$  is an ordered quadruple. If  $n \in \text{nat}$  and  $A$  is a set and  $a_i \in A$  for each integer  $i \in \{1, \dots, n\}$ , then  $(a_1, a_2, \dots, a_n)$ , where is an ordered  $n$ -tuple.

**Theorem 3.** *If each of  $n$  and  $k$  is a positive integer and  $A$  is an  $n$ -element set, then the number of ordered  $k$ -tuples that can be selected from  $A$  is  $n^k$ .*

Another way in which to state Theorem 3, and one that is useful for solving problems of the type that arise in situations involving sophisticated counting, is:

**Theorem 4.** *If each of  $n$  and  $k$  is a positive integer and  $P$  is an  $n$ -element set (population), then the number of ordered samples of size  $k$  that can be drawn with replacement (repetition) from  $P$  is  $n^k$ .*

### Ordered Samples without Repetition (permutations)

**Theorem 5.** *If each of  $n$  and  $k$  is a positive integer and  $P$  is an  $n$ -element set (population), then the number of ordered samples of size  $k$  that can be drawn without replacement from  $P$ , is  $n(n-1)(n-2)\cdots(n-k+1)$ .*

**Definition 16.** *If  $n$  is a non-negative integer then  $n$  factorial is denoted by  $n!$  and defined by*

$$n! = \begin{cases} 1 & \text{if } n = 0 \text{ or } n = 1 \\ n(n-1)(n-2)\cdots 1 & \text{if } n > 1 \end{cases}$$

**Problem 57.** *Let each of  $n$  and  $k$  represent positive integers with  $k \leq n$  and show that  $n(n-1)(n-2)\cdots(n-k+1) = n!/(n-k)!$ .*

The next theorem is merely a restatement of Theorem 5 when  $k = n$ .

**Theorem 6.** *If  $P$  is an  $n$ -element population, then the number of ordered samples of size  $n$  that can be drawn without replacement from  $P$  is  $n!$ .*

**Definition 17.** *If  $n$  is a positive integer and  $S$  is an  $n$ -element set, then a permutation of  $S$  is a bijection on  $S$ .*

**Example 8.** *Let  $S = \{1, 2, 3\}$ . One permutation of  $S$  would be the bijection  $f : S \rightarrow S$  defined by  $f(1) = 2, f(2) = 3$ , and  $f(3) = 1$ . Typically, we omit all the function notation and just write the range of  $f$  as  $(2, 3, 1)$ . Notice that of course, the order matters when we use  $()$  but not when we use  $\{\}$ .*

**Problem 58.** List all the permutations of  $S = \{1, 2, 3\}$ .

**Problem 59.** How many seven-letter strings can be formed using the letters from the word TUESDAY, where no letter may be used twice? How many five-letter strings?

**Problem 60.** Let  $D = \{a, b, c, d, e\}$  and  $R = \{1, 2, 3\}$ . How many functions are there with domain all of  $D$  and range a subset of  $R$ ? How many are one-to-one? How many are surjective?

**Problem 61.** Let  $D = \{a, b, c\}$  and  $R = \{1, 2, 3, 4, 5\}$ . How many functions are there with domain all of  $D$  and with range a subset of  $R$ ? How many are one-to-one? How many are surjective?

### Unordered Samples without Repetition (subsets)

**Definition 18.** If  $n$  is a positive integer and  $k$  is a nonnegative integer not larger than  $n$ , then  **$n$  choose  $k$**  is the number denoted by

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}.$$

**Problem 62.** Compute:

1.  $\binom{10}{3}$

2.  $\binom{10}{7}$

3.  $\binom{5}{0}$

4.  $\binom{5}{5}$

**Problem 63.** Show that

$$\binom{20}{8} = \binom{20}{12}. \text{ Show that if } n \text{ and } k \text{ are positive integers and } n \geq k \text{ then}$$
$$\binom{n}{k} = \binom{n}{n-k}$$

**Theorem 7.** If  $n$  is a positive integer and  $k$  is a nonnegative integer not larger than  $n$ , then the number of  $k$ -element subsets of an  $n$ -element set is  $\binom{n}{k}$ .

**Problem 64.** *Dr. Shannon has a total of six nieces and nephews. She has just won a set of eight different CD's. In how many different ways can she*

1. *Give each child exactly one CD?*
2. *Give away all the CD's so that each child gets at least one without giving any child more than two?*
3. *Give away all the CD's so that each child gets at least one?*

**Problem 65.** *Suppose that each of  $m$  and  $n$  is a positive integer and each of  $A$  and  $B$  is a set such that  $|A| = n$  and  $|B| = m$ . How many functions are there from  $A$  to  $B$ ? How many are one-to-one? How many are surjective?*

**Problem 66.** *Six cards are to be drawn from a standard deck and laid on a table in the order in which they were drawn. How many outcomes are possible to this "experiment"?*

**Problem 67.** *You and 13 of your closest friends have decided to form a club.*

1. *If you decide to have four officers – a president, a vice president, a secretary, and a treasurer - how many possible slates of officers are there?*
2. *If you decide that the job of secretary is too much for one person and elect a president, a vice president, a treasurer, and two secretaries, how many slates are there? (Assume that the secretaries will divide up the load as they see fit, that is, they will not be elected separately to do specific tasks.)*

**Problem 68.** *To avoid the diplomatic quagmire of deciding who will sit at the head of the table and who at the foot, a group planning peace talks with the single heads of state of four nations decides to seat all four at a round table, where all spots are equally prestigious and powerful.*

1. *How many possible seating arrangements are there?*
2. *How many arrangements are there if you care only who sits next to whom but not on which side of the person each neighbor sits?*
3. *Define an equivalence relation on the set of possibilities for Part a the equivalence classes of which represent the possibilities for Part b.*

**Problem 69.** *An elementary-school teacher, Mr. Linksweller, is directing an after-school parade with 12 of his students. Three of them will be twirling batons, five will be playing cymbals, and four will be doing somersaults. They will be parading in single file.*

1. How many different parades are possible if he wants to have the twirlers followed by the cymbal players, with the tumblers at the rear?
2. How many are possible if he simply keeps together the children who are doing the same thing?
3. How many would be possible in a free-for-all, where the kids are in any order?
4. How many are possible if the twirlers stay together but the others can be in any order?

**Problem 70.** A coin is tossed six times and the results, heads or tails on each toss, are recorded in order.

1. How many outcomes are possible?
2. How many of these have exactly one head?
3. How many have at least one head?
4. How many have at least one head and at least one tail?

**Definition 19.** Suppose that  $A$  is a set and  $f : A \rightarrow A$  is a function. If  $x \in A$  satisfies  $f(x) = x$  then we say that  $x$  is a **fixed point of  $A$  with respect to  $f$** .

**Problem 71.** Suppose that  $A = \{!, ", \#, \$\}$ .

1. How many functions are there on  $A$  for which  $!$  is a fixed point?
2. How many for which  $!$  and  $"$  are fixed points?
3. How many for which  $!$ ,  $"$  and  $\#$  are fixed points?
4. How many functions are there that fix every point of  $A$ ?

**Problem 72.** Suppose  $A = \{!, ", \#, \$\}$ . Let  $F1$  be the set of functions  $f : A \rightarrow A$  for which  $!$  is a fixed point. Let  $F2$  be the set of functions  $f : A \rightarrow A$  for which  $!$  and  $"$  are fixed points. Let  $F3$  be the set of functions  $f : A \rightarrow A$  for which  $!$ ,  $"$  and  $\#$  are fixed points. Let  $F4$  be the set of functions for which all four elements are fixed points. Let  $F5$  be the set of functions  $f : A \rightarrow A$  for which  $"$  is a fixed point. What are  $|F1 \cap F2|$ ,  $|F3 \cap F2|$ ,  $|F3 \cap F4|$  and  $|F1 \cap F2 \cap F3|$ ? How about  $|F1 \cap F5|$ ?

**Problem 73.** If six cards are chosen without replacement from a standard deck, how many hands are possible if

1. The six cards can be anything?

2. At least one card is an ace?
3. At least four are clubs?
4. Exactly three of the cards are hearts?

**Problem 74.** The five-member math club at Salisbury Steak University decided to hold a raffle, with each member being responsible for selling ten tickets (for a total of 50). Each club member bought one ticket and sold nine to non-members. The stubs were then to be thrown into a fish bowl and three winning tickets were to be chosen at random.

1. Of the possible outcomes, in how many would at least one math-club member win?
2. How many outcomes involve no math club member winning?
3. How many outcomes involve exactly two math club members winning?
4. How many outcomes involve all three winners being club members?

**Theorem 8. Binomial Theorem:**

If each of  $x$  and  $y$  represents a real number and  $n$  is a non-negative integer, then

$$(x+y)^n = \binom{n}{0}x^n + \binom{n}{1}x^{n-1}y + \binom{n}{2}x^{n-2}y^2 + \cdots + \binom{n}{n-1}xy^{n-1} + \binom{n}{n}y^n.$$

**Problem 75.** 1. Expand  $(x+y)^3$  by hand and via the theorem.

2. Expand  $(x-y)^5$  via the theorem.
3. Use the Binomial Theorem to expand  $(x-1)^{10}$ . When  $x = 2$ , what happens?

**Problem 76.** What is the coefficient of  $x^7$  in the expansion of  $(1+x)^{23}$ ?

**Problem 77.** Suppose that  $A$ ,  $B$ , and  $C$  are sets such that  $B \subseteq A$ ,  $C \subseteq A$ ,  $B \cap C = \emptyset$ ,  $|A| = 50$ ,  $|B| = 20$ , and  $|C| = 15$ .

1. How many four-element subsets does  $A$  have?
2. How many five-element subsets of  $A$  are disjoint from  $B$ ? How many are disjoint from  $C$ ? How many are disjoint from  $B$  and  $C$ ?
3. How many six-element subsets of  $A$  intersect  $B$ ? How many intersect at least one of  $B$  and  $C$ ?

4. How many seven-element subsets of  $A$  contain at least three elements of  $B$ ?
5. Consider the set of all functions on  $A$ . How many of them map  $B$  to  $B$ ? (That is, for how many of them is it true that, if  $x$  is in  $B$ , then  $f(x)$  is in  $B$ ?) How many of them map  $C$  to  $C$ ? How many map  $B$  to  $B$  and  $C$  to  $C$ ?

**Problem 78.** If  $n$  is a positive integer and  $k$  is a positive integer no larger than  $n$ , then

$$\binom{n+k-1}{n-1} = \binom{n+k-1}{k}.$$

### 3.1 Project: Unordered Samples with Repetition

**Problem 79.** Chocolate M&Ms come in six colors: red, blue, green, yellow, orange, and brown. Considering a five-M&M bag containing three red and two green candies to be of a different type from a bag containing one red, one green, two blue, and one yellow, how many (different) types of five-candy bags can Mars produce?

**Problem 80.** One nonnegative-integer solution to the equation  $x_1 + x_2 + x_3 + x_4 + x_5 = 5$  is the quintuple  $(1, 1, 1, 1, 1)$  since  $1 + 1 + 1 + 1 + 1 = 5$ . Two other solutions are:  $(4, 0, 0, 0, 0)$  and  $(0, 4, 0, 0, 0)$ . How many nonnegative-integer solutions are there?

**Problem 81.** A boy at a candy store is told that he may have 25 candies and that he may choose from 7 types of fudge squares, a fancy chocolate rabbit, coconut macaroons, and 10 types of hard candy. How many different choices may he make?

**Problem 82.** In the immediately-preceding problem, suppose that he is additionally told that he cannot have more than one fudge square or more than one macaroon. How many choices may he make?

**Problem 83.** A philanthropist has invited 25 unknown kids to an art exhibit and she wants to give each of them one of seven books the gallery gift shop carries. How many different purchases might she make at the gift shop for this?

**Problem 84.** Repeat Problem 80 for the equation  $x_1 + x_2 + x_3 + x_4 + \cdots + x_n = k$  where each of  $n$  and  $k$  is a positive integer.

### The Principle of Inclusion and Exclusion

**Theorem 9.** *If  $A$  and  $B$  are intersecting finite sets neither of which is a subset of the other, then  $|A \cup B| = |A| + |B| - |A \cap B|$ .*

**Theorem 10.** *If  $A$ ,  $B$ , and  $C$  are finite sets such that each of the appropriate intersections exists, then  $|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$ .*

**Problem 85.** *Generalize Theorem 10 to four sets.*

# Chapter 4

## Induction and Recursion

In this chapter we will study a method for proving that a given statement is true for every natural number.

**Example 9.** *We begin with three statements that we might try to prove are true for every natural number. Which do you think are true for every natural number?*

1. *The sum of the first  $n$  positive integers is  $n(n+1)/2$ .*
2. *My car will start on the  $n^{\text{th}}$  day after yesterday.*
3. *The number  $p(n) = n^2 - n + 41$  prime for every  $n \in \mathbb{Z}^+$ .*

Suppose I wish to prove that the first statement above is true for every positive integer. Let's check the first few. If  $n = 1$  then this says  $1 = \frac{1(1+1)}{2}$  which is true. If  $n = 2$  then this says that  $1 + 2 = \frac{2(2+1)}{2}$  which is true. If  $n = 3$  then this says that  $1 + 2 + 3 = \frac{3(3+1)}{2}$  which is true. Now, even if we proved this was true for  $n = 1, 2, 3, \dots, 1,000,000$  we would not know if it was true for  $n = 1,000,001$ ! How do we *prove* that it is true for all natural numbers? Let's proceed by contradiction and suppose that it is *not* true for every positive integer. Since it is not true for every positive integer, then let  $n$  be the *last* positive integer for which it is true. Since it is true for  $n$ , we know that

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

and adding  $n+1$  to both sides we have

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

and using common denominators to combine the terms on the right

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

which leaves us with

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

We assumed that  $n$  was the last number for which the statement was true and we showed that the statement is true for  $n + 1$ . Thus, we have a contradiction. This means that our assumption that it was not true for all positive integers is false. Hence, the statement is true for all positive integers.

**Problem 86.** Show that the sum of the first  $n$  cubes is  $\frac{n^2(n+1)^2}{4}$ .

**Definition 20.** A set of numbers is **bounded** if there are two numbers  $m$  and  $M$  so that for each number  $x$  in the set,  $m \leq x \leq M$ .

**Problem 87.** Show that for each positive integer  $n$ , a set with  $n$  elements is bounded.

**Problem 88.** Show that the sum of the squares of the first  $n$  positive integers is  $n(n + 1)(2n + 1)/6$ .

The technique we have just used is called *induction*. Here is how most books would state it.

**Theorem 11. Principle of Mathematical Induction** Let  $S(n)$  be a statement about an arbitrary positive integer  $n$ . If

1.  $S(1)$  is true and
2.  $S(k + 1)$  is true whenever  $S(k)$  is true,

then  $S(n)$  is true for all positive integers  $n$ .

If you want to prove Theorem 11, you'll likely use exactly the technique we have used in these first three problems. If you don't want to prove it, skip the next problem!

**Problem 89.** Prove Theorem 11.

**Problem 90.** The sum of the first  $n$  odd positive integers is  $n^2$ .

**Problem 91.** Assume that Theorem 1 (The Multiplication Principle for Counting) is true and prove that Theorem 2 (The Generalized Multiplication Principle for Counting) is true for any integer  $n$ .

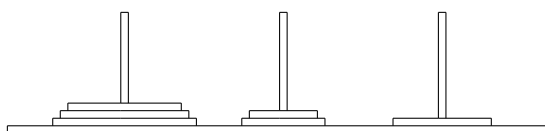
**Problem 92.** From your Calculus I course, you know that if each of  $f$  and  $g$  is a differentiable functions, then  $(fg)' = f'g + fg'$ . Use induction to show that for every natural number  $n$ , if each of  $f_1, f_2, f_3, \dots, f_n$  is a differentiable function then

$$(f_1 f_2 f_3 \cdots f_n)' = f_1' f_2 f_3 \cdots f_n + f_1 f_2' f_3 \cdots f_n + f_1 f_2 f_3' \cdots f_n + \cdots + f_1 f_2 f_3 \cdots f_n'.$$

**Problem 93.** Show that  $1/2 + 1/4 + 1/8 + 1/16 + \dots + 1/2^n = 1 - 1/2^n$ .

**Problem 94.** Show that  $3^0 + 3^1 + 3^2 + \dots + 3^n = (3^{n+1} - 1)/2$  is valid for all non-negative integers,  $n = 0, 1, 2, \dots$

**Problem 95.** Show that  $1 \cdot 2 + 2 \cdot 3 + 3 \cdot 4 + 4 \cdot 5 + \dots + n \cdot (n + 1) = n(n + 1)(n + 2)/3$ .



### Towers of Hanoi

The Towers of Hanoi is a mathematical puzzle consisting of three rods with  $n$  disks of different radii which can slide onto any rod. Start with the disks stacked all on one rod, the largest on the bottom, the smallest on the top and in order of size. The object of the puzzle is to move the stack to another rod, obeying the following rules:

1. Only one disk may be moved at a time.
2. Each move consists of taking the upper disk from one of the rods and sliding it onto another rod, on top of the other disks that may already be present on that rod.
3. No disk may be placed on top of a smaller disk.

**Problem 96.** In the Tower of Hanoi puzzle, it is possible to move  $n$  discs in  $2^n - 1$  moves.

**Problem 97.** In the Tower of Hanoi puzzle, if  $n$  discs are moved from one peg to another, then at least  $2^n - 1$  moves must be made.

In many cases a stronger version of induction is useful.

**Theorem 12. PRINCIPLE OF STRONG INDUCTION:**

Let  $S(n)$  be a statement about an arbitrary positive integer  $n$ . If

1.  $S(1)$  is true and
2.  $S(k+1)$  is true whenever  $S(m)$  is true for  $m = 1, 2, 3, \dots, k$ ,

then  $S(n)$  is true for all positive integers  $n$ .

**Example 10.** Use both standard induction and strong induction to show that every amount of postage of 12 cents or more can be formed using just 4-cent and 5-cent stamps.

**Problem 98.** Use Strong Induction to prove that every positive integer is either 1, a prime, or a product of primes.

There are many commonly used functions  $f$  with domain  $\mathbb{Z}^+$  that are not easily defined explicitly. Instead, we tell how to compute  $f$  by what we call a recursive process. To do this, we choose a positive integer  $b$  (often  $b = 1$ ), give

1. the values of  $f(1), f(2), \dots, f(b)$  explicitly and then
2. tell how to compute, for any  $k > b$ , the value of  $f(k)$  using the previous values  $f(1), f(2), f(3), \dots, f(k-1)$ .

**Problem 99.** Use Strong Induction to prove that (1.) and (2.) can be used to compute  $f(n)$  for every positive integer  $n$ .

This is exactly what happens when you ask a program to do a recursive call. An advantage of recursively defined functions is that their properties can often be verified by using Mathematical Induction. For example, suppose that we define

1.  $a(1) = 2$
2.  $a(k+1) = 2a(k) - 1$

Here we have taken  $b = 1$ . At the beginning of this chapter we proved by Mathematical Induction that  $a(n)$  is given explicitly by the equation  $a(n) = 2^{n-1} + 1$ .

**Problem 100.** The *Fibonacci Sequence* is obtained by taking  $b = 2$  and defining

1.  $F(1) = 1, F(2) = 1$  and
2.  $F(k+2) = F(k) + F(k+1)$  for  $k \geq 1$ .

Compute  $F(3)$  through  $F(12)$ . Use induction to prove that for all  $n \geq 1$ ,

$$F(1) + F(2) + F(3) + \dots + F(n) + 1 = F(n+2)$$

**Problem 101.** Check that  $F(3), F(6), F(9)$  and  $F(12)$  are even. Then use induction to prove that  $F(3n)$  is even for all  $n \geq 1$ .

## 4.1 Project: More Induction

Use induction to prove that the statement,  $a(n) = 2^{n-1} + 1$  is true for every natural number.

**Problem 102.** Use induction to prove that if  $n \geq 5$ , then  $n!$  is a multiple of 5.

**Problem 103.** Define a function  $g$  recursively by

1.  $g(1) = 2$  and
2.  $g(k+1) = 4g(k) - 5$ .

Compute  $g(7)$ . Use induction to prove that  $g(n) > 2^n$  for all  $n \geq 4$ .

Recursively defined functions can grow surprisingly fast. As a result, one must be cautious about asking a computer to do recursions since both time and memory are limited. When we were students, if you divided by zero or asked a computer to compute a number that was too large for its memory, it exploded, immediately killing the student. That is why everyone over the age of 30 is smart. ; >

**Problem 104.** Define a series of functions  $A_1, A_2, A_3, \dots$  as follows:

1. Let  $A_1(n) = 2n$  for all  $n$ .
2. For  $k = 1, 2, 3, \dots$  let
  - (a)  $A_{k+1}(0) = 1$  and
  - (b)  $A_{k+1}(m+1) = A_k(A_{k+1}(m))$ .
1. Use induction to show that  $A_2(n) = 2^n$  for all  $n$ .
2. Use induction to show that  $A_n(1) = 2$  for all  $n$ .
3. Write down  $A_3(5)$ .
4. Describe  $A_3(n)$ .

5. The number  $A_4(4)$  is so large that I will give an “A” in the course to anyone who can bring me a printout of its value (in normal decimal notation) by the end of the term.

**Problem 105.** *In one of our recent wars, an army captain received orders to deploy his soldiers over a designated enemy area and to instruct them to shoot any resident who came into view.*

*Now the captain talked with other captains and heard that among the new recruits there were soldiers who for some strange reason didn't buy into this mission. They were even known, in the heat of battle, to shoot their own captains and go AWOL. This worried him, so he devised the following plan.*

*He would deploy his soldiers, as ordered, but instruct them to situate themselves so that each one was nearest to exactly one other soldier. Each soldier was be ordered to watch that nearest soldier to him, and report any suspicious behavior to the captain.*

*Assuming the number of soldiers was odd, prove that there was at least one soldier who wasn't being watched.*

# Chapter 5

## Equivalence Relations

In this chapter we will examine what it means for two things to be, “equivalent,” a notion that turns out to have important applications in mathematics and science. Suppose you write a piece of software to check student algebra homework. You might need to determine when two equations in variables  $x$  and  $y$  are equivalent, meaning that they have the same sets of solutions. Thus  $y = 3x + 4$  and  $6x = 2y - 8$  are equivalent, so both are valid answers to a the same problem. Two computer programs might be thought of as “equivalent” if they always produce the same output from the same input.

Let’s recall what we know about relations in general before we move to *equivalence* relations.

Recall from Definitions 8 and 10 that if  $\mathbf{A}$  and  $\mathbf{B}$  are sets, then the **Cartesian product** of  $\mathbf{A}$  and  $\mathbf{B}$  is the set of all ordered pairs  $(x, y)$  where  $x \in \mathbf{A}$  and  $y \in \mathbf{B}$ ,

$$\mathbf{A} \times \mathbf{B} = \{(x, y) \mid x \in \mathbf{A} \text{ and } y \in \mathbf{B}\},$$

and a **relation** from  $A$  to  $B$  is any subset of this set.

**Problem 106.** Suppose we have the relation  $R = \{(x, y) : x \in \{a, b, c, d, e\} \text{ and } y \in \{\text{earth, moon, sun}\}\}$  then  $\text{Dom}(R) = \{a, b, c, d, e\}$  and  $\text{Rng}(R) = \{\text{earth, moon, sun}\}$ . Graph  $R$ . Is  $R$  a function?

**Problem 107.** Let  $A = \{1, 2, 3\}$ . Write out all elements of  $A \times A$ . List two examples of relations on the set  $A = \{1, 2, 3\}$ , one which is not a function and one which is a function.

Functions are one example of relations and equivalence relations are a second example of relations. If  $R$  is any relation, then  $xRy$  means  $(x, y) \in R$ .

**Definition 21.** Suppose  $A$  is a set and  $R \subseteq A \times A$  is a relation on  $A$ .

1. If for every  $x \in A$  we have  $xRx$  then  $R$  is said to be **reflexive**.

2. If for every  $x, y \in A$  satisfying  $xRy$  we have  $yRx$  then  $R$  is said to be **symmetric**.
3. If for every  $x, y, z \in A$  satisfying  $xRy$  and  $yRz$  we have  $xRz$  then  $R$  is said to be **transitive**.
4. If  $R$  is reflexive, symmetric and transitive, then  $R$  is said to be an **equivalence relation**.

Note that for equivalence relations, the domain and range must be the same.

**Example 11.** Suppose  $X = \{ \text{students in our class} \}$  and  $x$  is related to  $y$  means  $x$  and  $y$  have the same classification ( $Fr$ =freshman,  $So$ =sophomore,  $Jr$ =junior,  $Sr$ =senior). Discuss whether the reflexive, symmetric and transitive properties hold.

**Example 12.** Suppose  $X = \{ \text{all people in the world} \}$  and  $x$  is related to  $y$  means  $x$  is a friend of  $y$ . Discuss whether the reflexive, symmetric and transitive properties hold.

**Definition 22.** Given two integers  $x$  and  $y$  we say  $x$  **divides**  $y$  if there is an integer  $k$  so that  $y = kx$ .

**Example 13.** Thus 3 divides 39 (with  $k = 13$ ), but 8 does not divide 4 (because  $k = \frac{1}{2}$  is not an integer).

**Problem 108.** Let  $A$  be the set of all positive integers. Define a relation  $R$  by  $xRy$  if  $x$  divides  $y$ . Which of these properties hold for the relation  $R$ ?

1. reflexive (does  $x$  divide  $x$ ?)
2. symmetric (does  $x$  divides  $y$  imply that  $y$  divides  $x$ ?)
3. transitive (if  $x$  divides  $y$  and  $y$  divides  $z$  does  $x$  divide  $z$ ?)

**Problem 109.** Let  $A$  be the set of all real numbers. Let  $xRy$  if  $x = y$ . Is  $R$  an equivalence relation?

**Problem 110.** Let  $A$  be the set of all real numbers. Let  $xRy$  if  $x \leq y$ . Is  $R$  an equivalence relation?

**Problem 111.** Let  $A$  be the set of all integers. Suppose that  $xRy$  if  $x$  and  $y$  have the same remainder when divided by 5. Is  $R$  an equivalence relation?

Consider the set  $A$  of children in elementary school. For certain purposes it is helpful to think of two children as being equivalent if they were born in the same year. We then have one grade for each *equivalence class*. If

Charlie is in the fourth grade, his equivalence class is the fourth grade class. Or we might think of two children as equivalent if they are working at the same grade level in mathematics. We could then partition the children into different math classes, each working at a different level. Alternately, we could think of two children as equivalent if they have the same favorite hobby, and then partition them into clubs that each consist of children with a particular hobby. But sometimes we just need to realize that each child is unique, and focus on the partition with a single child in each equivalence class. These are all different partitions, or equivalence relations, on the same set  $A$ . There is no “right” meaning of equivalence; rather, each notion of equivalence is useful in a different context.

The next definition is a precise way of saying that you may break the set  $A$  down into the subsets of elements that are related to one another then the union of all these subset will be all of  $A$  and we'll denote the subset of all elements related to  $a$  by  $[a]$ .

**Definition 23.** For an element  $a \in A$ , the **equivalence class of  $a$**  is the set  $\{y : y \in A \text{ and } aRy\}$  and is denoted by  $[a]$ .

**Definition 24.** Given a set  $A$  and an equivalence relation  $R$  on  $A$ , the set of all the equivalence classes is called  **$A$  modulo  $R$**  and is denoted  $A/R$ . That is,  $A/R = \{[a] : a \in A\}$ .

To determine what the equivalence classes are, just pick an element and ask yourself, “What other elements are related to this element?” Once you’ve done this for a few elements, you’ll understand all the equivalence classes for that particular relation.

**Problem 112.** For each problems 108 through 111 where  $R$  turned out to be an equivalence relation, define the equivalence classes.

**Definition 25.** A collection  $P$  of sets is said to be **pairwise disjoint** if the intersection of any two sets in the collection is empty.

**Definition 26.** A collection  $P$  of set is said to be a **partition** of the set  $A$  if the collection is pairwise disjoint and the union of the sets in  $P$  is  $A$ .

The next problem gives us a new way to define equivalence relations. Previously, if we had a relation, we checked to see if it was reflexive, symmetric, and transitive. If so, then we could list all the equivalence classes. After the next problem, if we have a set and we can partition it into a collection of subsets which are pairwise disjoint and whose union is our set, then we can define a relation by saying that  $x$  and  $y$  are related if they are in the same subset in the partition. This relation will be an equivalence relation, so we now have a way to create equivalence relations.

**Problem 113.** Suppose  $P$  is a partition of the set  $A$ . Define the relation  $R$  on  $A$  by  $xRy$  if there is  $B \in P$  such that  $x$  and  $y$  are both in  $B$ . Prove that  $R$  is an equivalence relation on  $A$ .

**Problem 114.** Let  $A$  be the set of points  $(x, y)$  in the plane. Suppose  $(x, y)R(x', y')$  if  $(x, y)$  and  $(x', y')$  are the same distance from the origin. Describe the equivalence class for  $(0, 2)$ , that is describe  $[(0, 2)] = \{(x, y) \in \mathbb{R}^2 : (x, y)R(0, 2)\}$ . Determine if  $R$  is an equivalence relation and if so, describe all the equivalence classes.

**Problem 115.** Let  $A$  be the set of all integers. Suppose  $xRy$  if  $x$  and  $y$  are integers and  $x - y$  is a multiple of 3. Determine if  $R$  is an equivalence relation and if so, describe the equivalence classes.

**Problem 116.** Let  $A$  be the set of real numbers. Suppose  $xRy$  if  $x$  and  $y$  are integers and  $x + y$  is even. Determine if  $R$  is an equivalence relation and if so, describe the equivalence classes.

**Problem 117.** There is a standard and important notion of equivalence between sets. Let  $\mathcal{S}$  denote the collection of all sets. For  $X, Y \in \mathcal{S}$ , we say that  $XRY$  if there is a bijection  $f : X \rightarrow Y$  from  $X$  to  $Y$ . Thus finite sets  $X$  and  $Y$  are equivalent if and only if they have the same number of elements, but in general they need not be finite. Show that  $R$  is reflexive, symmetric and transitive.

## 5.1 Project: More on Equivalence Classes

**Problem 118.** Suppose  $R$  is an equivalence relation on the nonempty set  $A$ . Prove that

1.  $A = \cup A/R$
2. two equivalence classes are distinct iff they have no points in common.

**Problem 119.** Is Problem 20 valid if we define  $(a, b) = \{\{a, 1\}, \{2, b\}\}$ ?

**Problem 120.** How would one formally define an ordered triple? an ordered quartuple?, an ordered quintuple?, an ordered  $n$  tuple?

In a different context, we might think of two compound sentences in propositional logic to be “equivalent” if they have the same truth tables. For example, the truth table below shows that every implication  $p \rightarrow q$  is equivalent to its contrapositive  $\neg q \rightarrow \neg p$ .

| $p$ | $q$ | $p \rightarrow q$ | $\neg q \rightarrow \neg p$ |
|-----|-----|-------------------|-----------------------------|
| 0   | 0   | 1                 | 1                           |
| 0   | 1   | 1                 | 1                           |
| 1   | 0   | 0                 | 0                           |
| 1   | 1   | 1                 | 1                           |

Table 5.1: 1 is True; 0 is False

**Problem 121.** Let  $\mathbf{A}$  be the (infinite) set of all compound sentences in propositional logic that can be constructed using the sentence variables  $p$  and  $q$  and the connectives  $\neg$ ,  $\wedge$ ,  $\vee$  and  $\rightarrow$ . For  $a, b \in \mathbf{A}$  we say that  $a \equiv b$  if  $a$  and  $b$  have the same truth table.

1. Show that  $\equiv$  is an equivalence relation.
2. List as many non-equivalent compound sentences as you can.
3. How many different equivalence classes are there?

**Problem 122.** Suppose each of  $R$ ,  $S$  and  $T$  is a relation. Prove that  $T \circ (S \circ R) = (T \circ S) \circ R$ .

**Problem 123.** Suppose  $R$  is a relation. Prove that  $I_{\text{Rng}(R)} \circ R = R$  and  $R \circ I^{\text{Dom}(R)} = R$  where  $IM = \{(x, x) : x \in M\}$ .

**Problem 124.** Suppose each of  $R$  and  $S$  is a relation. Prove that  $(S \circ R)^{-1} = R^{-1} \circ S^{-1}$ .

**Problem 125.** Let  $\mathbf{S} = \{a, b, c, d, e, f, g, h\}$ , let  $\mathbf{T} = \{a, b, c\}$  and let  $\mathbf{A}$  be the set of all  $(2^8)$  subsets of  $\mathbf{S}$ . For  $X, Y \in \mathbf{A}$ , we define  $XRY$  if  $X$  and  $Y$  have the same intersection with  $\mathbf{T}$ . Determine if  $R$  is an equivalence relation and if so, describe the equivalence classes.

**Problem 126.** Let  $\mathbf{A} = \{0, 1\}^5$  be the set of 32 different five-tuples of 0s and 1s, that is, all sequences  $(a, b, c, d, e)$  where  $a, b, c, d, e \in \{0, 1\}$ . For  $x, y \in \mathbf{A}$ , define  $xRy$  to mean that  $x$  and  $y$  have the same number of 1s. Show that  $R$  is an equivalence relation and write down all the members of each of the different equivalence classes.

**Problem 127.** Suppose  $\mathbf{A} = \{0, 1\}^n$  is the set of all  $n$ -tuples of 0s and 1s. For  $x, y \in \mathbf{A}$ , again define  $xRy$  to mean that  $x$  and  $y$  have the same number of 1s. How many different equivalence classes does  $\mathbf{A}$  have?

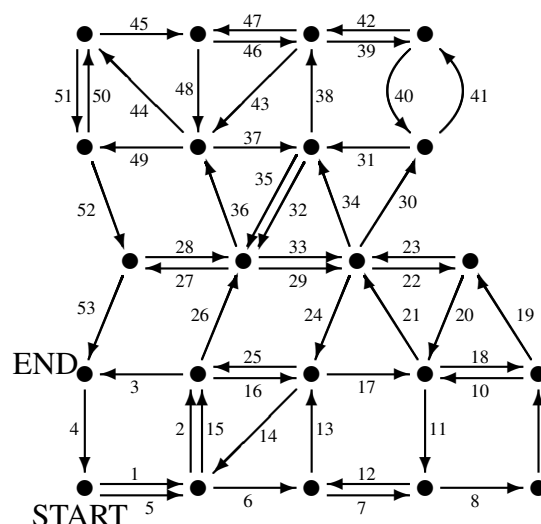
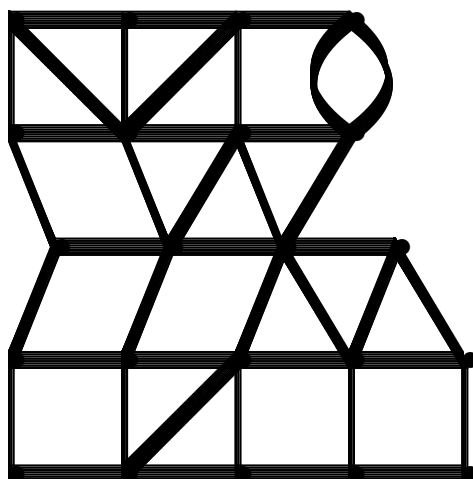
# Chapter 6

## Euler Paths and Circuits

We begin this chapter with a practical problem. You are offered a job delivering papers in the neighborhood shown below to the left. Your boss suggests that you use the route he used before he passed it on to you, shown below to the right. The rules are simple:

1. It takes 10 minutes to walk each block.
2. You must deliver papers on each block, so must walk each block.
3. You may start and end anywhere because your mom drops you off and picks you up.

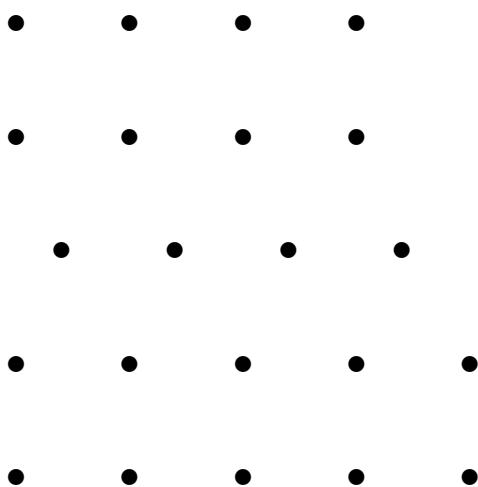
Use the patterns on the next page to determine if you can find a better path than the one your boss suggests. Write out the best path you find in Problem 128.





**Problem 128.** Write down your best solution to the Mail Carrier Problem, labeling the *START* and *END* and numbering the blocks as you go.

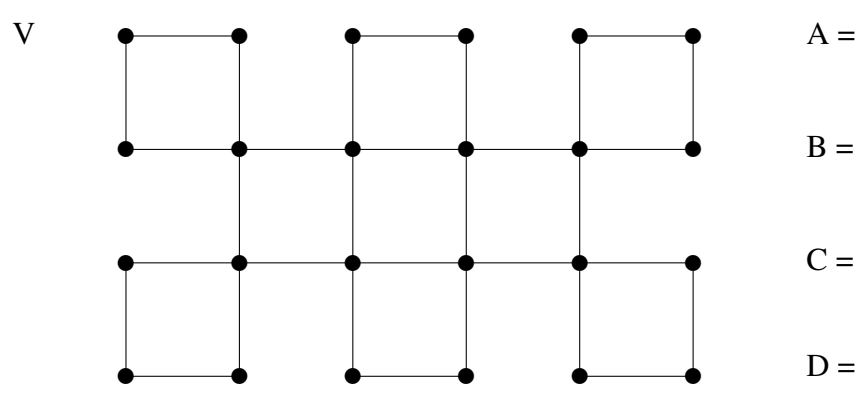
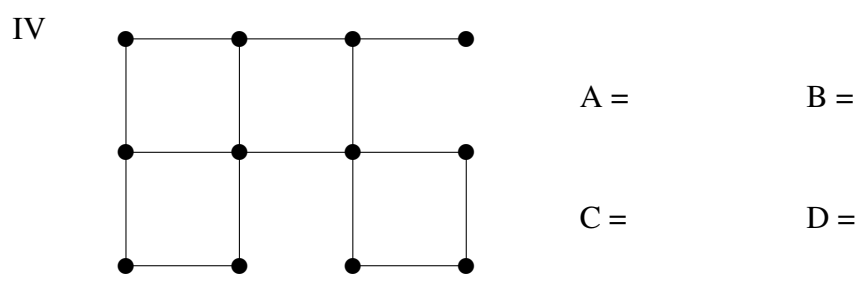
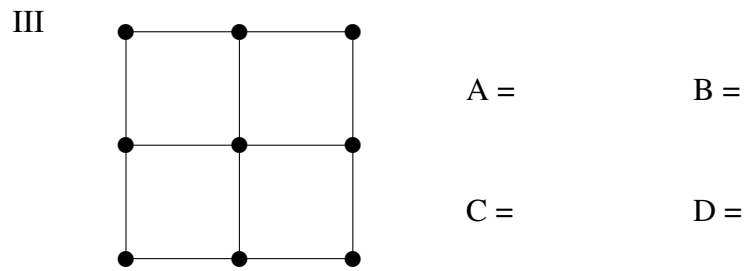
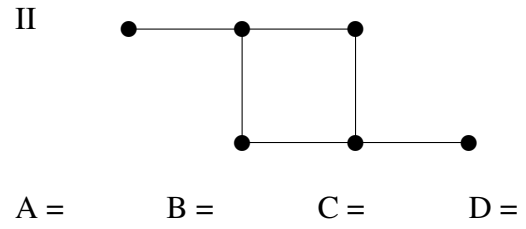
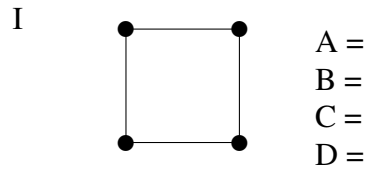
1. How many blocks are in the neighborhood?
2. How long will it take if you walk each block once?
3. How many blocks must you walk in your best solution?
4. How long will it take you?



Clearly it is difficult to determine the shortest path, so let's examine the mini-neighborhoods illustrated on the next page by a schematic diagram showing streets as lines and intersections as solid circles.

**Problem 129.** For each graph on the next page write down:

1. the total number of blocks (edges) in the neighborhood,
2. the time required, at ten minutes per block, if each block were traveled exactly once,
3. the smallest number of blocks that you can find to deliver mail along each block (show your path) and
4. the time required for your solution to Item 3 at ten minutes per block.



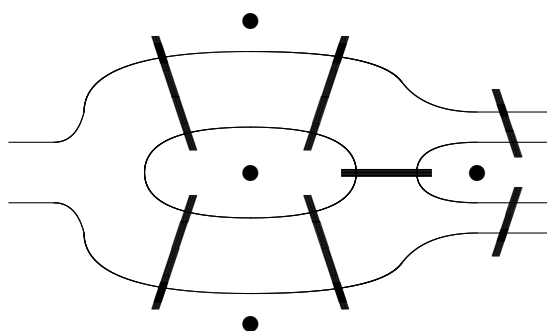


Figure 6.1: Königsberg Bridges

The mathematics of Mail Carrier Problems date to eighteenth century east Prussia. The historic city of Königsberg (now Kaliningrad) is broken into four regions by the forking Pregel River. Seven bridges connect the parts of the city as shown in Figure 6.1. Strolling the city, the residents of Königsberg found a challenge. Is there a path that would take them over each bridge exactly once?

**Problem 130.** Find a path connecting all four regions of Königsberg (Figure 6.1) that takes you over each bridge exactly once.

Suppose we make each region of Königsberg a dot and each bridge a curved line connecting the respective regions. The result is shown in Figure 6.2. Now this problem looks like the Mail Carrier problem except that we want to traverse each block *exactly once*.

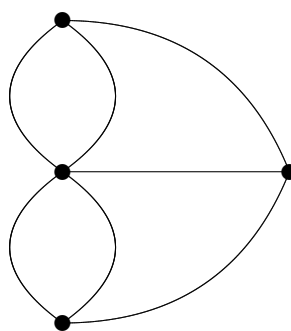


Figure 6.2: Mathematical Representation of Königsberg Bridges

The mathematical model Figure 6.2 of the Königsberg problem in Figure 6.1 is called a *graph* and was invented (discovered?) by Leonard Euler, an eighteenth century resident of Königsberg, in order to solve the Königsberg Bridge Problem.

To solve Mail Carrier Problems and the Königsberg Bridge Problem, we will first try to decide when it is possible to find a path going over each block or each bridge *exactly once*. These problems can both be rephrased as the problem of determining whether or not the associated graph has an Euler path.

**Definition 27.** A **graph** consists of a set of points, called its **vertices**, and a set of line segments connecting them, called its **edges**.

**Definition 28.** A **loop** is an edge connecting a vertex to itself. Two or more edges between the same two vertices are called **multiple edges**.

**Problem 131.** Can you draw a graph which has 6 vertices, 4 loops and 2 multiple edges? If so, do so. If not, why not?

**Definition 29.** The **degree** of a vertex in a graph is the number of edges coming into it. (A loop counts as two edges.) A vertex is **even** if its degree is an even number, and it is **odd** if its degree is an odd number.

**Problem 132.** Can you draw a graph with 5 vertices, each of degree 4, which has no loops or multiple edges? If so, do so. If not, why not?

**Problem 133.** Can you draw a graph with 3 vertices of degree 2 and 2 vertices of degree 3? If so, do so. If not, why not?

**Problem 134.** Can you draw a graph with 2 vertices of degree 2 and 3 vertices of degree 3? If so, do so. If not, why not?

**Problem 135.** Can you draw a graph with 1 vertex of degree 1, 2 vertices of degree 2, 3 vertices of degree 3 and 4 vertices of degree 4? If so, do so. If not, why not?

**Definition 30.** Two vertices are **adjacent** if an edge connects them.

**Definition 31.** A **path** is a sequence of adjacent vertices with a connecting edge between each pair of adjacent vertices.

**Definition 32.** A **circuit** is a path which starts and ends at the same vertex without repeating an edge.

**Definition 33.** An **Euler path** is a path that passes over every edge of the graph exactly once.

**Definition 34.** An *Euler circuit* is a circuit that passes over every edge of the entire graph.

**Problem 136.** Can you draw a graph with 8 vertices and 6 edges which contains no circuit at all? If so, do so. If not, why not?

**Definition 35.** A graph is *connected* if every pair of vertices is connected by some path. A *component* of a graph is a maximal connected piece of the graph.

**Problem 137.** Can you draw a connected graph with exactly 3 edges that does not have an Euler path? If so, do so. If not, why not?

The next theorems tell us which graphs have an Euler path or an Euler circuit and how, if there is one, to find it.

**Theorem 13. Odd Vertex Theorem** Suppose  $A$  is an odd vertex of a graph.

1. An Euler path that starts at  $A$  cannot end at  $A$ .
2. An Euler path that does not start at  $A$  must end at  $A$ .
3. Every Euler path either starts or ends at  $A$ .

**Theorem 14. Even Vertex Theorem** Suppose  $B$  is an even vertex of a graph. Then every Euler path that starts at  $B$  must also end at  $B$  (and is therefore an Euler circuit).

From these two observations we can establish the following necessary conditions for a graph to have an Euler path or an Euler circuit.

**Theorem 15. First Euler Path Theorem** If a graph has an Euler path, then

1. it must be connected and
2. it must have either 0 or 2 odd vertices.

**Theorem 16. First Euler Circuit Theorem** If a graph has an Euler circuit, then

1. it must be connected and
2. it must have no odd vertices.

The two theorems above tell us which graphs do *not* have an Euler path or circuit. We would like to know that all remaining graphs (connected graphs with either 0 or 2 odd vertices) do have Euler paths. To do so requires a few theorems. Try to prove the next three problems, but don't let them slow you down. We can assume them and move on to the rest of the chapter and show them later when you get them.

**Problem 138.** *The sum of an odd number of odd numbers is odd.*

**Problem 139.** *The sum of the degrees of the vertices of a graph is even.*

**Problem 140.** *Every graph has an even number of odd vertices.*

**Definition 36.** *A **subgraph** of a graph is any subset of the graph.*

**Definition 37.** *A **mouse** is a subgraph of a graph which is connected to the rest of the graph exactly one edge called the (**the tail**). The (**the body**) of the mouse is the portion of the mouse with the tail removed.*

**Problem 141.** *Show that the body of every mouse contains an odd vertex.*

**Definition 38.** *An edge of a connected graph is called a **bridge** if removal of that edge causes the graph to become disconnected.*

**Theorem 17. Fleury's Theorem** *If a graph is connected and has either 0 or 2 odd vertices, then an Euler path can be found by the following algorithm.*

1. *Start at one of the odd vertices if there are any; otherwise start where ever you choose.*
2. *At each vertex in the path you build, traverse and remove any untraversed edge that is not a bridge in the graph that remains. Traverse a bridge if there is no other choice.*
3. *Stop where you reach a vertex which is not adjacent to any untraversed edge, that is, when you can't go any further without repeating an edge.*

*At that point you will have completed an Euler path.*

**Proof:** Assume there are two odd vertices,  $A$  and  $B$ . (A very slight variation of this argument works if there are no odd vertices.) Start Fleury's Algorithm at  $A$  and observe each of the following.

1. *At each step in the path, either we are at  $B$  and there are no odd vertices in the remaining graph or we are at a vertex  $X$  other than  $B$  and  $B$  and  $X$  are the only two odd vertices in the remaining graph.*
2. *If we go from a vertex  $X$  to a vertex  $Y$ , erasing the connecting edge, then  $Y$  and the remaining edges still form a connected graph.*
3. *When we reach a vertex with no remaining edges (degree 0), that vertex must be  $B$  and all of the edges must have been erased. #*

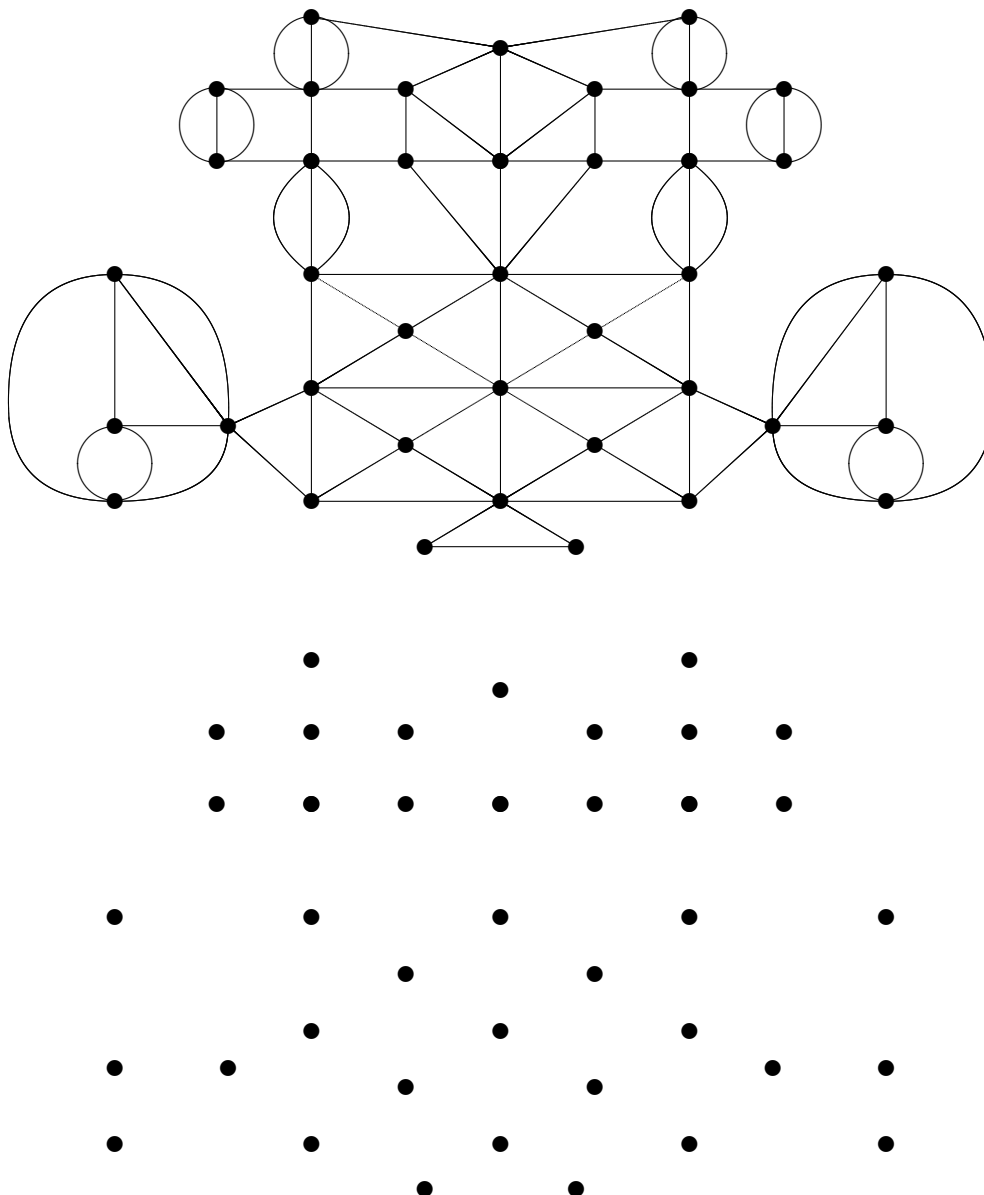
Fleury's Algorithm will be demonstrated in class using the graph in Figure 6.4. Applying Fleury's Algorithm and Fleury's Theorem, we see that the following two conclusions following immediately.

**Theorem 18. Second Euler Path Theorem** *If a graph is connected and has exactly 2 odd vertices, then it has an Euler path.*

**Theorem 19. Second Euler Circuit Theorem** *If a graph is connected and has no odd vertices, then it has an Euler circuit (which is also an Euler path).*

**Problem 142.** *Decide whether or not each of the three graphs in Figure 6.3 has an Euler path or an Euler circuit. If it has an Euler path or Euler circuit, trace it on the graph by marking the START and END and number the edges. If it does not, then write a complete sentence explaining how you know it does not.*

**Problem 143.** Use Fleury's Algorithm to find an Euler path for the graph below. A good strategy is to draw the graph in pencil at the bottom, number the edges of the Euler Path on the top graph, and erase each edge on the bottom as you traverse it. Label the "START" and "END" of your path on the upper graph.



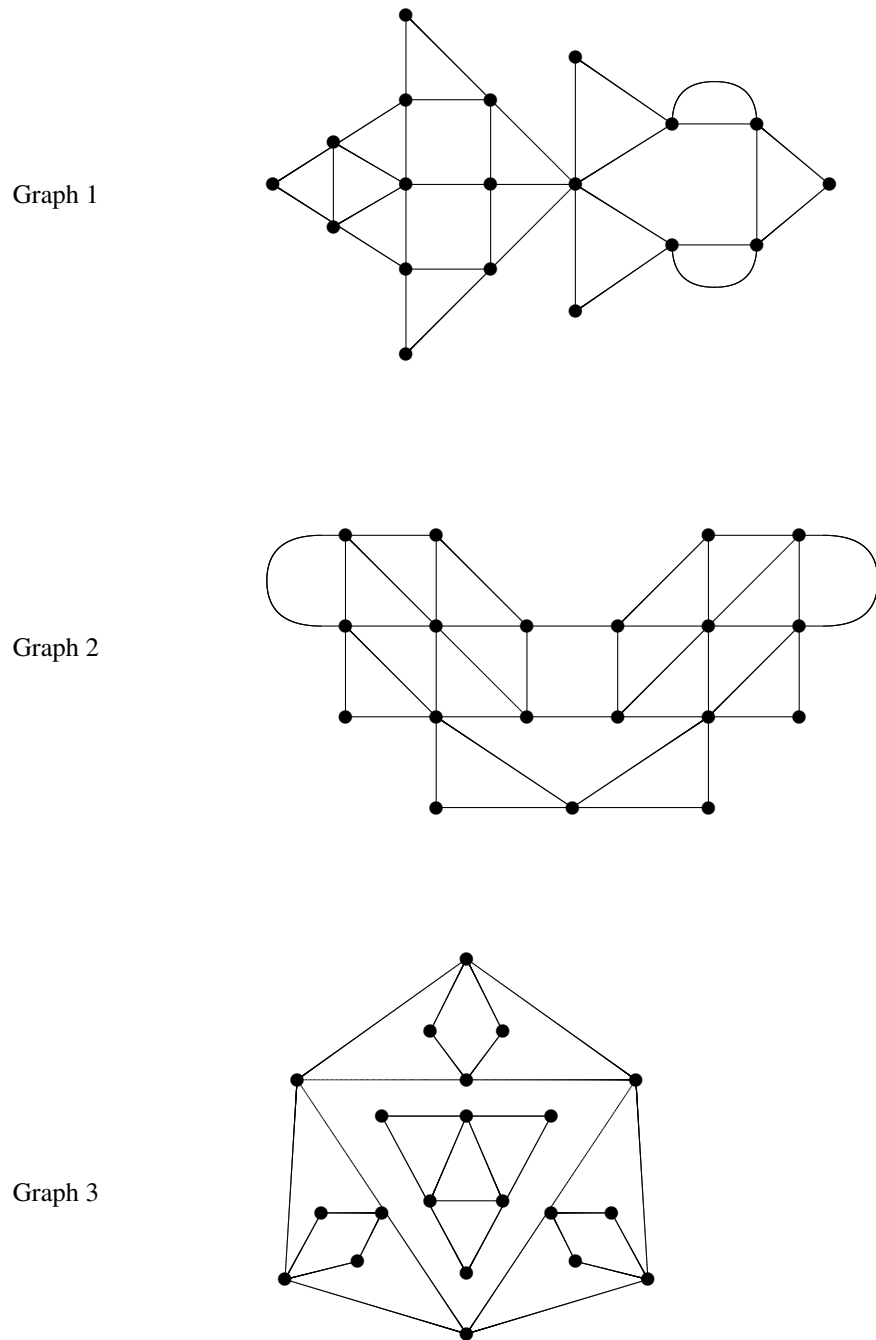


Figure 6.3: Three Graphs for Problem 142

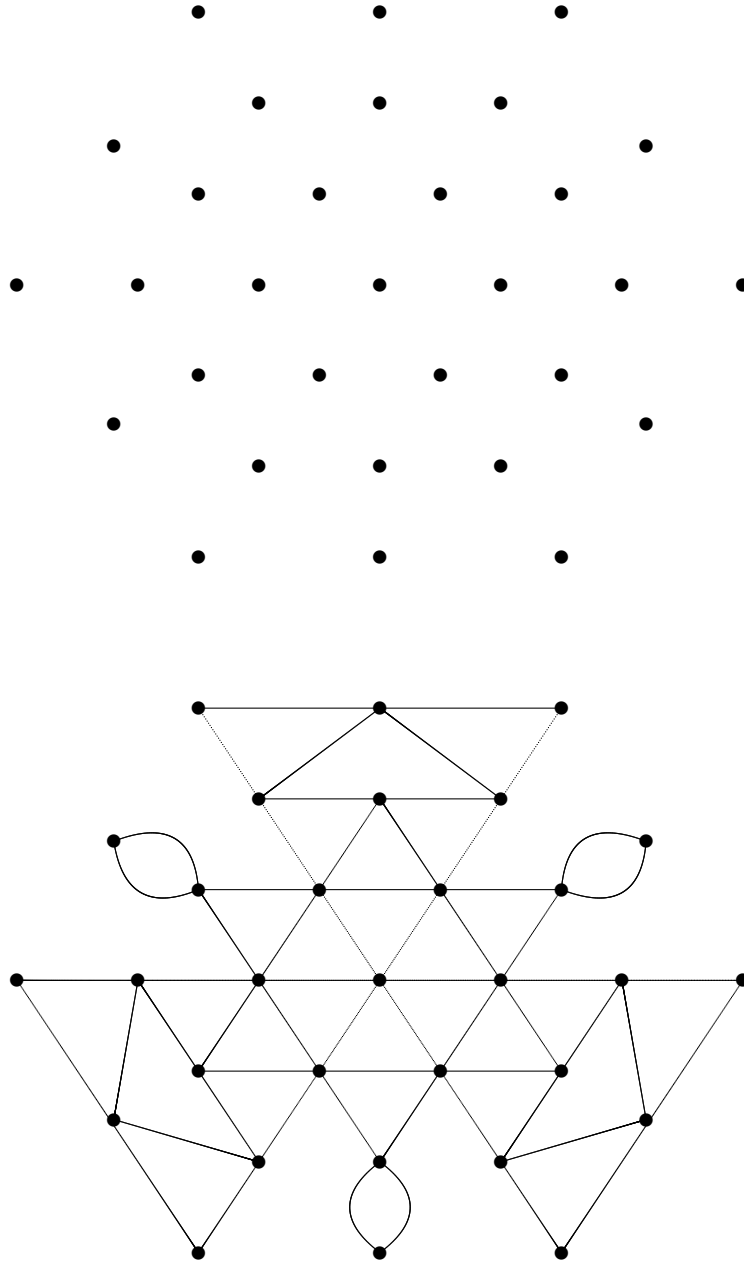


Figure 6.4: Fleury's Algorithm

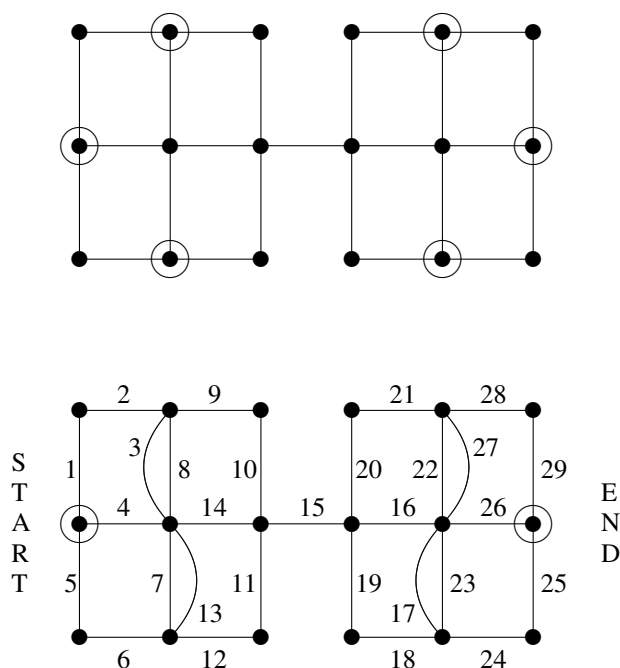


Figure 6.5: Eulerization of the Mail Carrier Problem

Returning to our mail carrier problem, consider the problem of delivering mail in the top neighborhood illustrated in Figure 6.5. Because it has six odd vertices, we know that we can't find an Euler Circuit and we can't travel down each road exactly once. What is the smallest number of blocks that we can walk and still deliver all the mail? To answer this we apply a process called the **Eulerization** of the graph. In the bottom copy of the graph in Figure 6.5, we have inserted the minimum possible number of edges that will reduce the number of odd vertices to 2. The minimum number of edges that will reduce the number of odd vertices to 2 is 4. Because we can't build new roads (we are mail carriers, not city planners), the added edges only represent the streets we walk over twice, not new streets. The new graph is connected with exactly 2 odd vertices, so has an Euler Path of length  $25(\text{original edges}) + 4(\text{added edges}) = 29(\text{total edges})$  blocks. This is the shortest possible path that will cover each block in the neighborhood.

**Problem 144.** Apply the Eulerization process to the original Mail Carrier Problem in Problem 128 at the beginning of the chapter. How many blocks are in the neighborhood and what is the minimum number of blocks that must be walked in order to deliver all the mail?



## Chapter 7

### Traveling Salesman Problems

Most GPS units have the capability to allow you to input multiple locations and have the unit compute a path that visits all locations while minimizing your choice of time or distance. Does yours always find the best path? Mine doesn't. The unit most likely uses one of the algorithms in this chapter. The Traveling Salesman Problem (TSP) models a variety of different real world problems:

**work orders**, where vertices represent repair jobs and weights represent times required to re-tool for the next job;

**jobs on a machine**, where vertices represent tasks and weights represent times to reconfigure the machine for the next task;

**circuit boards**, where vertices represent holes to be drilled in the board and weights represent times to rotate the board into the new position.

Returning to your employment status... Having determined that the shortest mail route required nine-hour days, you've decided to look for a job as a traveling salesman. A health food supplier is looking for someone to make deliveries to five stores in the region. Figure 7.1 shows the five stores and the distances between them in miles. You will be paid for each trip that visits all five stores.

This looks good since you can travel at your own pace and won't have to worry about graph theory! Since you live near  $A$ , you decide on the loop,

$$A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow A$$

which is

$$60 + 70 + 60 + 100 + 90 = 380 \text{ miles.}$$

On second thought, you observe that you *could* travel the loop,

$$A \rightarrow B \rightarrow D \rightarrow E \rightarrow C \rightarrow A$$

which is

$$60 + 40 + 100 + 60 + 100 = 360 \text{ miles.}$$

Your brain starts to hurt when you wonder, “What is the shortest route?” and “Just how many routes are there?” Back to graph theory and counting...

**Problem 147.** *List all the possible routes starting and ending at A. List the total distance traveled for each route. What is a (the?) shortest route? Could there be a shorter route starting and ending somewhere else?*

Let’s rephrase this problem in the language of graph theory.

**Definition 39.** A **weighted graph** is a graph in which each edge has been assigned a positive number called its **weight**.

**Definition 40.** A circuit which passes through every vertex exactly once is called a **Hamilton circuit**.

**Definition 41.** A **minimum weight Hamilton circuit** is a Hamilton circuit that has the smallest possible weight of all Hamilton circuits.

In graph theory terms, the **TSP** is the problem of finding a minimum weight Hamilton circuit. Notice that there was an edge between every pair of stores in Figure 7.1. Such a graph is called *complete*.

**Definition 42.** A **complete graph** is a graph in which every pair of edges is connected by an edge.

**Problem 148.** *How many Hamilton circuits are there in a complete graph on three vertices? On four vertices? On five vertices? On  $N$  vertices?*

We will study three different algorithms for solving the TSP, each of which is unsatisfactory for a different reason. The first method we study is

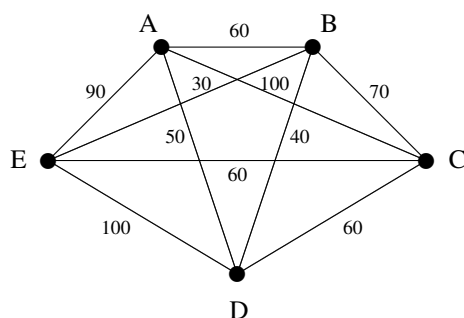


Figure 7.1: Health Food Stores

the one you applied above.

### **Brute Force TSP Algorithm**

1. *List all Hamilton circuits.*
2. *Compute the total weight of each circuit in the list.*
3. *Pick out the minimum weight circuit.*

This algorithm finds all optimal solutions, since we check every possible solution. However, the time required to execute the algorithm grows rapidly as we add new vertices. Suppose we have a weighted graph with 25 vertices. To decide if this graph has an Euler path, and then to use Fleury's Algorithm to find that Euler Path, might take 20 minutes. How long would it take to check all  $24!$  different Hamilton paths? If a computer can check 1 million circuits per second, then this would take approximately 8 billion years.

The next two algorithms are called **greedy algorithms** because at each step we do whatever is most advantageous to us at that moment, with planning ahead.

### **Nearest Neighbor Greedy Algorithm**

1. *Choose any vertex as a starting point.*
2. *At each step, go to any one of the closest remaining vertices.*
3. *Once every vertex has been visited, return home.*

*Now, repeat this process starting at another vertex. Once you have created circuits using each vertex as a starting point, compare the lengths of all the circuits and choose the one of minimum length.*

**Problem 149.** *Apply the nearest neighbor greedy algorithm to the Health Food Store problem in Figure 7.1.*

**Cheapest Link Greedy Algorithm** *Add edges, one at a time, to form a Hamilton circuit as follows.*

1. *Choose any edge with the least weight.*
2. *At each step, add the least weight remaining edge so that*
  - (a) *no degree three vertex is formed and*
  - (b) *a circuit is not closed if unvisited vertices remain.*

3. Stop when step 2 cannot be repeated.

**Problem 150.** Apply the cheapest link greedy algorithm to Figure 7.1.

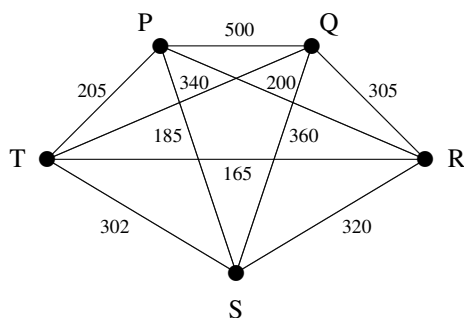


Figure 7.2: Weighted Graph

**Problem 151.** Solve the TSP for Figure 7.2 using the Brute Force Algorithm.

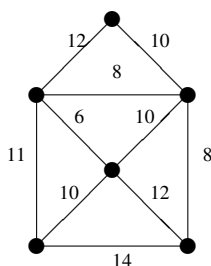


Figure 7.3: Cheapest Link Greedy Algorithm

**Problem 152.** Apply the cheapest link greedy algorithm to Figure 7.3.

**Problem 153.** Suppose you want to use the Brute Force Algorithm to solve the TSP for a graph with 12 vertices, that you can compute the length of one Hamilton circuit in 10 minutes and that the fate of the world rests on your results. How long will it take you to check all of the different circuits?

**Problem 154.** Use the Nearest Neighbor Greedy Algorithm to solve the TSP in Figure 7.2. You need to start by drawing 5 copies of the vertices of the above graph, one for each starting place. Circle the starting vertex of each one, and then insert the edges that you use for your circuit numbering them as you add them. Write under each one the length of the path you get. Finally, tell which path is the shortest. That is your solution to the TSP.

**Problem 155.** Use the Cheapest Link Greedy Algorithm to solve the TSP in Figure 7.2. Draw the 5 vertices, then add the edges one at a time, numbering them as you go, until you have a circuit. What is the length of this circuit?

**Problem 156.** The Greedy Algorithms (NN and CL), like Fleury's Algorithm but unlike the Brute Force Algorithm, are very quick and efficient to apply. The problem with them is that, unlike Fleury's Algorithm, they don't always give us the shortest path! Find a (small) example of a weighted graph in which neither of the Greedy Algorithms produce a correct answer to the TSP. (This proves mathematically that greed does not always pay!)

Since 1971, when it first gained public attention, considerable effort has gone into finding a good solution to the Traveling Salesman Problem, but no solution has been found that is quick, efficient, easy to apply (like Greedy algorithms) and guaranteed to work (like the Brute Force Algorithm). Considerable evidence suggests that no such solution exists at all.

## 7.1 Project: Greed Doesn't Pay

**Problem 157.** *Suppose that you need to make a delivery to each of 11 friends and then return home. What's more, your car is in very bad shape and you don't want to drive it any further than absolutely necessary. Call your house A and the houses of your friends B, C, D, E, F, G, H, I, J, K and L. Table 7.1 below indicates the distances between each pair of houses. You need to find a Hamilton circuit through the complete graph on 12 vertices, A, B, C, D, E, F, G, H, I, J, K and L, which has as small a total distance as possible. According to what you found in doing Problem 2.2 above, you are quite certain that you don't want to use the Brute Force Algorithm to find this circuit.*

|   | A | B | C  | D  | E  | F  | G  | H  | I  | J  | K  | L  |
|---|---|---|----|----|----|----|----|----|----|----|----|----|
| A |   | 8 | 15 | 7  | 6  | 9  | 10 | 16 | 11 | 7  | 9  | 10 |
| B |   |   | 9  | 12 | 6  | 9  | 11 | 15 | 12 | 8  | 9  | 6  |
| C |   |   |    | 7  | 10 | 11 | 9  | 5  | 6  | 9  | 10 | 12 |
| D |   |   |    |    | 8  | 6  | 13 | 10 | 9  | 5  | 11 | 7  |
| E |   |   |    |    |    | 10 | 7  | 6  | 9  | 10 | 8  | 9  |
| F |   |   |    |    |    |    | 9  | 8  | 10 | 6  | 5  | 16 |
| G |   |   |    |    |    |    |    | 7  | 15 | 7  | 6  | 8  |
| H |   |   |    |    |    |    |    |    | 5  | 9  | 14 | 10 |
| I |   |   |    |    |    |    |    |    |    | 8  | 19 | 7  |
| J |   |   |    |    |    |    |    |    |    |    | 12 | 5  |
| K |   |   |    |    |    |    |    |    |    |    |    | 6  |

Table 7.1: table of distances between 12 houses

1. Pick a random Hamilton circuit and compute its total distance.
2. Apply the Nearest Neighbor Greedy Algorithm, starting from A (only), to find a Hamilton circuit. What is its total length?
3. Apply the Nearest Neighbor Greedy Algorithm, starting from D (only), to find a Hamilton circuit. What is its total length?
4. Apply the Cheapest Link Greedy Algorithm to find a Hamilton circuit. What is the length of this circuit?

This example shows how the Greedy Algorithms can be efficient to use and can give a good solution, though they do not always give the best possible solution.

## Chapter 8

### Spanning Trees

The left graph in Figure 8.1 shows 17 towns and the roads connecting the towns. The town with the circle has a power plant. Your job as a consultant to the power company is to decide which roads to place power lines down so that every town receives power and you put line down on as few roads as possible. You don't need a power line constructed down every single road since you only need enough to ensure that there is a path from the power station to each town.

**Problem 158.** *Using the graph at the right of Figure 8.1, sketch a possible way to connect all the towns to the power station. Look for one that places power lines on the minimal number of roads.*

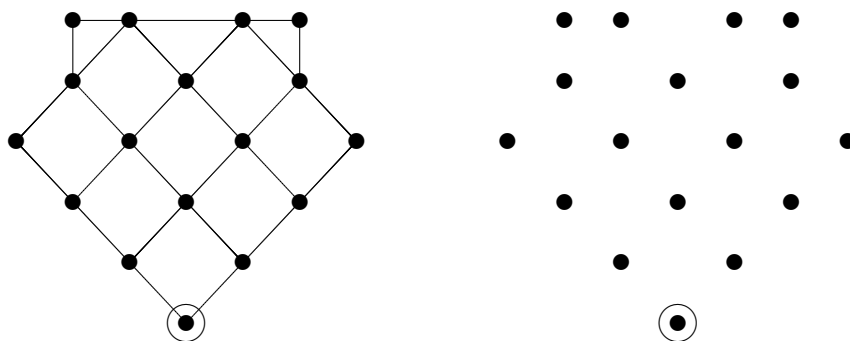


Figure 8.1: Power Grid Spanning Tree

Placing this in the language of graph theory requires a few definitions. Recall from Definition 35 that a **connected graph** is one with a path be-

tween every pair of vertices.

**Definition 43.** A *tree* is a connected graph containing no circuits.

**Definition 44.** A *spanning tree* of a graph  $G$  is a tree whose vertices are the same as the vertices of  $G$  and whose edges are among the edges of  $G$ .

**Problem 159.** Is the power grid you created in the last problem a spanning tree?

There are several equivalent definitions for trees:

- a **tree** is a connected graph with no circuits;
- a **tree** is a graph in which there is exactly one path from each vertex to each other vertex with no repeated edge;
- a **tree** is connected graph in which every edge is a cut edge.

Another real-world application for spanning trees might be to find the best possible collection of roads to plow when the city is snowed in. If we plow out a spanning tree then people may travel from any intersection (vertex) to any other intersection in the entire city by traversing only plowed streets (edges in the spanning tree). For a large city, computing the spanning tree by hand would be impractical. We need an algorithm that a computer can implement.

**Theorem 20. Spanning Tree Algorithm.** To construct a spanning tree for a connected graph  $G$ , make another copy of the vertices of  $G$ . Then

1. add edges from  $G$  between these vertices one at a time making sure to never close a circuit, and
2. stop when no additional edge from  $G$  can be added without closing a circuit.

**Problem 160.** Prove that for any connected graph  $G$ , the Spanning Tree Algorithm always produces a spanning tree for  $G$ .

**Problem 161.** Draw five different trees. Count the number of edges and the number of vertices. Make a conjecture for the relationship between the number of vertices and the number of edges in a tree?

**Problem 162.** Prove your conjecture from the previous problem.

Applying the Spanning Tree Algorithm to the power grid problem illustrated in Figure 8.1, you could find many different spanning trees to build

your power lines. Of course, in the real world, some would cost more to construct depending on both the distance between towns and the terrain between towns. Just as with the shortest path problems, this problem may be modeled mathematically using a **weighted graph** where the weights represent quantities like distances or costs.

**Minimum Weight Spanning Tree Problem.** *Given a weighted graph, find a spanning tree whose total weight is less than or equal to that of any other spanning tree.*

In Figure 8.2, we have added weights to represent either distances (in miles) or road construction costs (in millions of dollars) between our 17 towns.

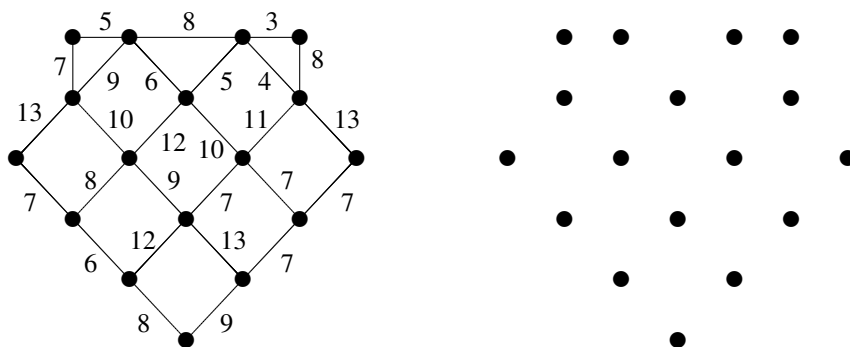


Figure 8.2: Weighted Graph

**Problem 163.** *Consider the weighted graph in Figure 8.2.*

1. *Using this weighted graph, calculate the total weight of the spanning tree in your solution to Problem 158.*
2. *Is your solution a minimum weight spanning tree?*
3. *Find a spanning tree for the weighted graph in Figure 8.2 that has weight less than your solution to Problem 163.*
4. *What is the least weighted spanning tree you can find?*

We can now ask if your solution has minimal total weight among all possible spanning trees. How might we do this without looking at all possible spanning trees? The answer was discovered by Joseph Kruskal in 1956

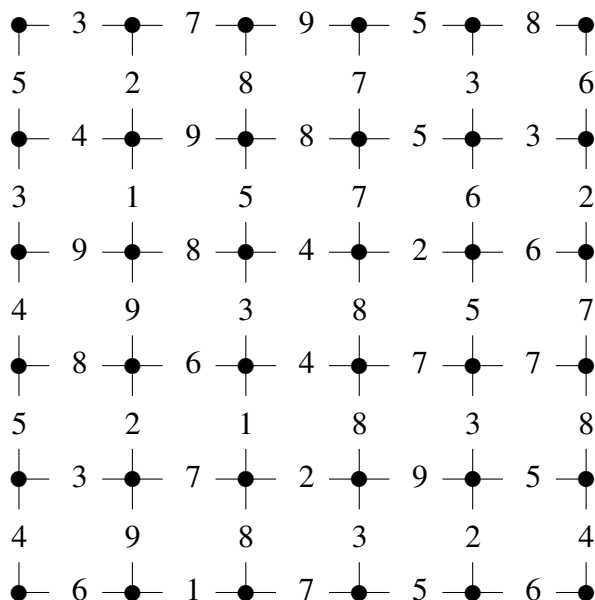
and is simply the Spanning Tree Algorithm enhanced by the Cheapest Link Greedy Algorithm. Instead of choosing an arbitrary edge that doesn't complete a circuit, we choose the cheapest (least weight) one. Since Kruskal's Algorithm is also a greedy algorithm, it is practical and efficient to use. But unlike our other greedy algorithms, it *always works!*

**Theorem 21. *Kruskal's Greedy Algorithm.*** *To construct a minimum weight spanning tree for a connected weighted graph  $G$ , make another copy of the vertices of  $G$ . Then*

1. *add edges from  $G$  between these vertices one at a time, each time choosing from among the edges that do not close a circuit one of minimal weight, and*
2. *stop when no additional edge from  $G$  can be added without closing a circuit.*

You can check that the 108-weight spanning tree above is obtained using Kruskal's Greedy Algorithm. Kruskal's important contribution was to show that his algorithm always produces a spanning tree of minimal possible total weight.

**Theorem 22. *Kruskal's Theorem:*** *Kruskal's Greedy Algorithm always produces a minimum weight spanning tree  $T$  from a connected weighted graph  $G$ .*

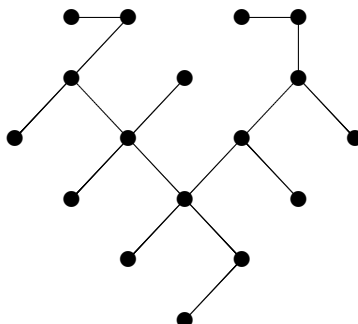


**Problem 164.** Use the Spanning Tree Algorithm to construct an arbitrary spanning tree for the graph. What is the total weight of this arbitrary spanning tree?

**Problem 165.** Apply Kruskal's Greedy Algorithm to find a minimum weight spanning tree. What is the total weight of this spanning tree?

## 8.1 Project: Intuition for a Proof that Kruskal's Algorithm Works

In the next two problems we will look at Kruskal's Theorem to see why the spanning tree obtained from Kruskal's Algorithm always has the smallest total weight of any spanning tree. In the figure below is an arbitrary spanning tree  $S$  for the graph in Figure 8.1. We would like to see why its total weight must be no less than that of the spanning tree obtained from Kruskal's Algorithm.



Spanning Tree  $S$

**Problem 166.** Apply Kruskal's Algorithm to the graph in Figure 8.2 to obtain a minimum weight spanning tree  $T$ . As you do it, number the edges in the order you add them. Find the total weight of  $T$ .

**Problem 167.** What is the total weight of the spanning tree  $S$ ? Make a copy of  $S$  in pencil. Apply the proof of Kruskal's Theorem to the spanning tree  $S$  to transform it into  $T$  by adding the edges of  $T$  to it, in the order you numbered them, each time removing an edge not in  $T$  that closes a circuit. As you do so, fill out a table like this:

---

| Weight of<br>edge added<br>to $S$ from $T$ | Weight of<br>edge not in $T$<br>removed from $S$ | Amount the<br>weight of $S$<br>is reduced |
|--|--|---|
| ...  | ...  | ...                                       |
| ...  | ...  | ...                                       |

When you have finished transforming  $S$  into  $T$ , add up the reductions you made in the weight of  $S$ . Now subtract this amount from the original weight of  $S$  to see if you get the weight of  $T$ . Write this out *neatly* so that it is clear what you have done. (If you can reduce the weight of  $S$  to get the weight of  $T$ , then you know that  $T$  must have smaller weight than  $S$ !)

**Problem 168.** *Prove that Kruskal's Algorithm always produces a minimal spanning tree.*

## Chapter 9

### Finite State Machines

In this chapter we'll study Finite State Transducers and Finite State Acceptors. Each is an example of a Finite State Machine. Finite State Machines (FSMs) are the simplest class of computing machines. They model the logic of machines which require logic circuitry but are not usually thought of as computers, such as vending machines, washing machines and toasters. FSMs are characterized by the fact that their action, at any point in their operation, is completely determined by their current state and their current input. They have no long term memory and are therefore unable to carry out any logical process of reviewing the past or anticipating the future.

#### Finite State Transducers

Suppose the directions on a vending machine ask the user to:

1. Choose either
  - (W) Pure Spring Water (65 cents) or
  - (C) Coca-Cola (45 cents).
2. Insert either quarters (q) or half dollars (h).
3. Press "RETURN" (R) for your drink and change.

In order to execute instructions, the machine needs to be able to remember, at any time, the user's selection and the amount of money it has received. Memory will take the form of **internal states**, one for each possible data set. The machine logic is represented by a directed labeled graph. The internal states form the vertices of the graph. Each arrow tells us two things:

1. for a particular input, what the output should be, and

2. what new internal state the machine should go to.

The start state is “ $s$ ”, and each other state has a name that indicates what it is to remember. The symbol “ $-$ ” means no output.

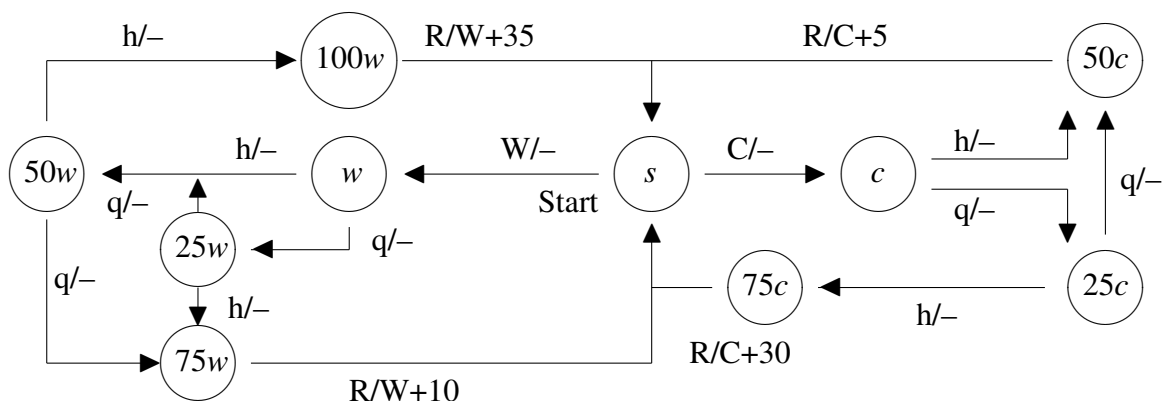


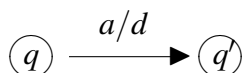
Figure 9.1: Transition Diagram for Finite State Transducer for Vending Machine

For example, if the user asks for a Water and puts in a quarter and a half dollar and then presses “RETURN”, the machine will go from state  $s$  to  $w$  to  $25w$  to  $75w$ , and then will output a Water and a dime.

**Definition 45.** A *finite state transducer* is a quintuple  $M = \langle \Sigma_1, \Sigma_2, Q, s, \delta \rangle$  where

1.  $\Sigma_1$  is a finite set of symbols, called the **input alphabet**;
2.  $\Sigma_2$  is a finite set of symbols, called the **output alphabet**;
3.  $Q$  is a finite set, the **internal states** of  $M$ ;
4.  $s \in Q$  is called the **initial state** of  $M$ ;
5.  $\delta : Q \times \Sigma_1 \rightarrow Q \times \Sigma_2$  is called the **state transition function**.

A finite state transducer is illustrated with a *transition diagram* such as the one in Figure 9.1. A transition diagram is a labeled directed graph whose vertices are the states of  $Q$ , and we label the arrow



from state  $q$  to state  $q'$  with  $a/d$  if  $\delta(q, a) = (q', d)$ , that is, in state  $q \in Q$  reading input symbol  $a \in \Sigma_1$  the machine is to go to state  $q' \in Q$  and output symbol  $d \in \Sigma_2$ .

**Problem 169.** Construct a transition diagram for a finite state transducer that will read a sequence of letters from the set  $\{a, b, c\}$  starting from the left (for example “caabbaccbcb”). As output, it will change each letter to an “e” until it has read three “a”s. After that, it will change each “a” to “b”, each “b” to “c” and each “c” to “a”.

**Problem 170.** Construct a transition diagram for a stamp vending machine that asks the user to

1. choose either a 10 stamp book (\$3.40) or a 20 stamp book (\$6.80);
2. insert either \$1, \$5 or \$10 bills;
3. press “RETURN” for stamps and change.

You may assume that the user never inserts a bill that will need to be returned as change; for example, a \$1 followed by a \$10.

**Problem 171.** Construct a transition diagram for a finite state transducer which will take as input a sequence of letters from  $\{a, b, c, x\}$ . It will output a blank symbol – for each input of  $a, b$  or  $c$ . When “ $x$ ” is the input, it will terminate and output the remainder of the number of “ $a$ ”s read divided by 3. (Thus its output will be 0, 1 or 2.)

Consider a vending machine that only accepts dollar bills and has six snacks, each costing one dollar. This machine has only eight buttons:  $A$ ,  $B$ , 1, 2, 3,  $D$  and  $C$ . This machine has two rows labeled  $A$  and  $B$  and three columns labeled 1, 2, and 3. Each combination  $A1, B1, A2, B2, A3, B3$  identifies a snack such as M&Ms, Cheetos, or Doritos. The button labeled  $D$  is activated when a dollar bill goes into the slot. The button  $C$  is the coin-return button. This machine is modeled in Figure 9.2. In our diagram,  $s$  represents the initial state (the state the machine is in when you walk up to it),  $x$  represents the accepting state (meaning you get your snack), the circles represent intermediate states, and the arrows tell you which state the machine will move to if you push a given button.

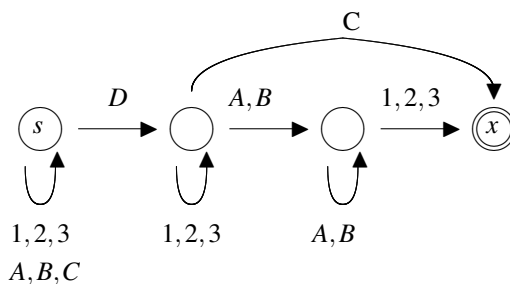


Figure 9.2: Vending Machine

**Problem 172.** Assume you are a graphic designer and sketch a picture of our vending machine. Assuming the machine is in state  $s$ , give three examples of strings consisting of letters from our input alphabet,  $\{A, B, C, D, 1, 2, 3\}$  that will result in a snack. Each of these three strings should result in the machine ending up in state  $x$ .

**Problem 173.** Assuming the machine is in state  $s$ , give three examples of strings consisting of letters from our input alphabet,  $\{A, B, C, D, 1, 2, 3\}$  that will not result in a snack. I.e. these three strings will not result in the machine ending up in state  $x$ . (FAIL!)

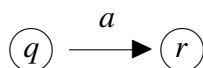
**Problem 174.** I designed this machine for Tater-Head-Ted's SnackCo Inc. which promptly went out of business. Describe some of the flaws that the machine has and draw a transition diagram for an improved machine.

Machines such as the one just described are called a **Finite State Acceptors**. For this finite state acceptor, there are four states: the initial state  $s$ , the accepting state  $x$  and two unlabeled states. And we have an input alphabet,  $\{A, B, C, D, 1, 2, 3\}$ . We also have a state transition function that tells you what state to move to depending on the input. This function is illustrated in Figure 9.2. This machine reads a finite string of letters and numbers and either accepts it, in which case you receive a snack, or rejects it, in which case you don't!

**Definition 46.** A *finite state acceptor* (FSA) is a quintuple  $M = (\Sigma, Q, s, Y, \delta)$  where

- $\Sigma$  is a finite set of symbols, called the **input alphabet** of  $M$ ,
- $Q$  is a finite set, the **internal states** of  $M$ ,
- $s \in Q$  is called the **initial state** of  $M$ ,
- $Y$  is a subset of  $Q$  called the **accepting states** of  $M$ , and
- $\delta : Q \times \Sigma \rightarrow Q$  is called the **state transition function** of  $M$ .

A FSA is illustrated by a labeled directed graph. The vertices of the graph are the states of  $Q$ , and there is a labeled arrow for each state  $q \in Q$  and each symbol  $a \in \Sigma$ . The mathematical expression,  $\delta(q, a) = r$  means that when our FSA  $M$  is already in state  $q$  and reads the input symbol  $a$ , it will go into state  $r$ . We illustrate this using this diagram.



Because an input to  $M$  will be a string of symbols (think of a series of button pushes on your car stereo) we define  $\Sigma^*$  as the set of all finite strings of symbols from  $\Sigma$ . The **empty string**, is denoted by  $\lambda \in \Sigma^*$ . Since the transition function  $\delta$  can only handle one input from our alphabet at a time, we extend this recursively to a function

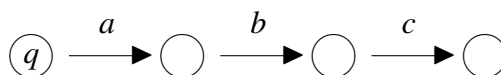
$$\delta^* : Q \times \Sigma^* \rightarrow Q$$

by defining

1.  $\delta^*(q, \lambda) = q$  and  $\delta^*(q, a) = \delta(q, a)$  for  $a \in \Sigma$ ,
2.  $\delta^*(q, au) = \delta^*(\delta(q, a), u)$  for  $a \in \Sigma, u \in \Sigma^*$ .

For example, if  $q \in Q$  and  $a, b, c \in \Sigma$ , then we have

$$\delta^*(q, abc) = \delta^*(\delta(q, a), bc) = \delta^*(\delta(\delta(q, a), b), c) = \delta(\delta(\delta(q, a), b), c).$$



The job of  $M$  will be to read a string  $w$  of symbols from  $\Sigma$  and either accept or reject it. If  $M$  begins in its initial state  $s$  reading a string  $w \in \Sigma^*$ , then  $\delta^*(s, w)$  is the state it will be in when it reaches the end of  $w$  and halts. We say that  $M$  **accepts** the string  $w$  if  $\delta^*(s, w) \in Y$ ; otherwise it **rejects**  $w$ . The set

$$L(M) = \{w \in \Sigma^* \mid \delta^*(s, w) \in Y\}$$

consists of all strings accepted by  $M$  and is called **the language recognized by  $M$** . We think of each string in  $\Sigma^*$  as expressing a question (namely, “Am I in  $L(M)$ ?”). The finite state acceptor answers YES if, reading  $w$ , it ends in a state of  $Y$ ; otherwise it answers NO. In the transition diagram we mark the start state with “START” and indicate the accepting states of  $Y$  with a double circle.

In practice we often find that after reading only part of a string we can be certain, regardless of what follows, the answer will be NO. In these cases

it is useful to specify a “dead state” which is non-accepting and cannot be exited. We frequently denote the dead state by  $Z$  and, since many arrow go to  $Z$ , follow the convention that undrawn arrows are intended to go to  $Z$ .

We denote the collection of languages recognizable by some FSA by

$$\mathbb{FSA} = \{L(M) : M \text{ is a finite state acceptor}\}.$$

An understanding of what languages are in  $\mathbb{FSA}$  will give us an indication of the capabilities of finite state machines.

**Problem 175.** Consider the FSA  $M$  illustrated in Figure 9.3. Here  $s$  is the start state,  $x$  (indicated with a double circle) is the only accepting state, and all undrawn arrows go to the ‘dead’ state  $z$ . Let  $\Sigma = \{0, 1, 2\}$  and list three strings in  $\Sigma^*$  that are accepted by  $M$  and three strings that are not accepted (rejected) by  $M$ .

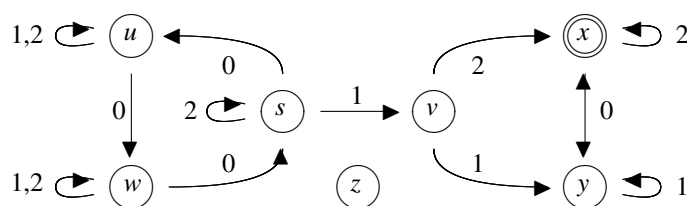


Figure 9.3: A Finite State Machine

**Problem 176.** Let  $\Sigma = \{a, b, c, d\}$ . Construct an FSA  $M_1$  that will answer the question, “Does  $w \in \Sigma^*$  begin with  $ab$ , contain exactly four  $c$ ’s, and end with  $d$ ?”.

**Problem 177.** Are the same strings accepted in Problem 178 if the two commas are removed?

**Problem 178.** Construct an FSA  $M_2$  that recognizes the language consisting of all strings that are either

1. a positive multiple of three 2s, followed by a possibly empty string of 0s and 1s or
2. a positive multiple of two 3s, followed by a possibly empty string of 1s and 2s.

**Problem 179.** Construct an FSA that recognizes the union of the languages recognized by  $M_1$  and  $M_2$  in Problems 176 and 178 above.

**Definition 47.** The *concatenation* of  $L(M_1)$  and  $L(M_2)$  consists of all strings  $uv$  (a string  $u$  followed by another string  $v$ ) where  $M_1$  accepts  $u$  and  $M_2$  accepts  $v$ .

**Problem 180.** Construct an FSA that recognizes the concatenation of the languages  $L(M_1)$  and  $L(M_2)$  in Problems 176 and 178 above.

## 9.1 Project: Machine Languages are closed under complement, union and intersection

Let  $M_1$  and  $M_2$  be the finite state acceptors,  $M_1 = \langle \Sigma, Q_1, s_1, Y_1, \delta_1 \rangle$  and  $M_2 = \langle \Sigma, Q_2, s_2, Y_2, \delta_2 \rangle$ . Let  $L_1 = L(M_1)$  and  $L_2 = L(M_2)$  be the languages associated with  $M_1$  and  $M_2$ . Recall that  $\mathbb{FSA}$  is the set of all languages from all finite state acceptors.

**Problem 181.** Demonstrate that  $\mathbb{FSA}$  is *closed under complement*, that is, if  $L_1 \in \mathbb{FSA}$  then  $\Sigma^* \sim L_1 \in \mathbb{FSA}$ . (Tell how to use  $M_1$  to design a FSA  $M$  that will recognize  $\Sigma^* \sim L_1$ .)

**Problem 182.** Demonstrate that  $\mathbb{FSA}$  is *closed under intersection*, that is, if  $L_1, L_2 \in \mathbb{FSA}$  then  $L_1 \cap L_2 \in \mathbb{FSA}$ . (Tell how to use  $M_1$  and  $M_2$  to design a FSA  $M$  that will recognize  $L_1 \cap L_2$ . HINT: Try using  $Q = Q_1 \times Q_2$  for the states of  $M$ .)

**Problem 183.** Demonstrate that  $\mathbb{FSA}$  is *closed under union*, that is, if  $L_1, L_2 \in \mathbb{FSA}$  then  $L_1 \cup L_2 \in \mathbb{FSA}$ . (Tell how to use  $M_1$  and  $M_2$  to design a FSA  $M$  that will recognize  $L_1 \cup L_2$ . See Problem 182.)

**Problem 184.** Do you think there is a FSA for  $L = \{a^n b^n \mid n = 1, 2, 3, \dots\}$ ?

## Chapter 10

### Languages and Machine Minimization

When writing computer programs or mathematical proofs, the first goal is to create one that works. The second goal is to make it as concise as possible. If we could write everything minimally, then the goal of writing programs to check the validity of proofs or programs would be easier.

Finite state acceptor models for simple computing machines give us a context in which this minimization issue can be fully resolved. In this chapter we will see how when given a finite state acceptor,  $M$  we can construct a minimal finite state acceptor  $M'$  so that  $L(M) = L(M')$ .

#### Alphabets and Languages

Let's begin with a motivating example, foregoing all formal definitions and relying only on our intuition for the moment. Suppose  $[\ ]$  represents a blank space,  $[a - z]$  means pick any letter between  $a$  and  $z$ , and  $*$  means you can repeat that you can repeat the last step as many times as you like before moving to the next step. So,  $[a - c]^*$  means you can pick as many letters from the set  $\{a, b, c\}$  as you like and string them together.

What might this expression represent?  $[A - Z][a - z]^*[\ ] [A - Z][A - Z]$

How about this one?  $[A - Z][a - z]^*([\ ] [A - Z][a - z]^*)^*[\ ] [A - Z][A - Z]$

How about this one?  $([A - Z][a - z]^*)^*[\ ] [A - Z][A - Z]$

How about this one?  $([\ ] [A - Z][a - z]^*)^* [A - Z][A - Z]$

Are the second and fourth expressions the same?

For a deeper understanding, we need to make the process of interpreting such expressions formal through precise definitions.

**Definition 48.** *Ok, it's really several definitions....*

1. An **alphabet**  $\Sigma$  is a finite set of **symbols**.
2. A **string** is a finite sequence of symbols from  $\Sigma$  written in a particular order.
3.  $\Sigma^*$  is the set of all strings that may be created from elements of  $\Sigma$  along with the **empty string** which will be represented by  $\lambda$  (as it is otherwise very hard to see).
4. If  $w \in \Sigma^*$  is a string, we denote the **length** of  $w$  by  $|w|$ .
5. Given strings  $u, v \in \Sigma^*$ , the **concatenation** of  $u$  and  $v$  is the string  $w = uv$  formed by tacking  $v$  onto the end of  $u$ . We say that  $u$  is a **prefix** of  $w$  and that  $v$  is a **suffix** of  $w$ .
6. If  $x, y, z \in \Sigma^*$  and  $w = xyz$ , we say that  $y$  is a **substring** of  $w$ .
7. If  $w \in \Sigma^*$  then the **powers** of  $w$  are also elements of  $\Sigma^*$  and are:
  - (a)  $w^0 = \lambda$  and
  - (b)  $w^{n+1} = w^n w$  where  $n \in \mathbb{N}$

**Problem 185.** Consider the alphabet  $\Sigma = \{a, b, c, \dots, x, y, z\}$ .

1. Give five examples of elements in  $\Sigma^*$ .
2. Are the words in the dictionary members of  $\Sigma^*$ ?
3. Is *quizzottlmf* in  $\Sigma^*$ ?
4.  $(\text{watchdog})^3 = \text{-----}$
5. Why do you think we call  $\lambda$  the **identity** string?

**Definition 49.** If  $\Sigma$  is an alphabet, then by a **language over**  $\Sigma$  we mean any subset of  $\Sigma^*$ .

You've studied operations on numbers,  $+$ ,  $-$ ,  $*$ ,  $/$ , and operations on sets,  $\cap$ ,  $\cup$ ,  $\oplus$ ,  $\times$ , now we study the operations on languages of concatenation, "star," and union. From here forward, we'll assume  $\Sigma$  is always an alphabet.

**Definition 50.** If each of  $L_1$  and  $L_2$  are languages, then the **concatenation** of  $L_1$  and  $L_2$  is

$$L_1 L_2 = \{w \in \Sigma^* \mid w = uv \text{ for some } u \in L_1, v \in L_2\}$$

**Problem 186.** Let  $L = \{a, ba, cat\}$  and compute  $LL$ , the concatenation of  $L$  with itself.

**Definition 51.** If  $L$  is a language over  $\Sigma^*$ , then we define the powers of  $L$  as follows:

1.  $L^0 = \{\lambda\}$
2.  $L^1 = L$
3.  $L^n = LL^{n-1}$  (the concatenation of  $L$  with  $L^{n-1}$ ).

**Definition 52.** If  $L$  is a language over  $\Sigma^*$ , then the *star*<sup>1</sup> of  $L$  is defined to be the infinite union of all powers of the language  $L$ , that is,  $L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \dots$ .

**Problem 187.** Suppose  $\Sigma = \{0, 1\}$ . Let  $L = \{1, 01\}$  and compute  $L^0, L^1, L^2$  and  $L^3$  and list the first 15 elements of  $L^*$ .

Let's summarize. If  $\Sigma$  is a finite set of symbols, (like our alphabet), then  $\Sigma^*$  is the infinite set of all strings that can be created by concatenating symbols from this alphabet. A language  $L$  (like the words in our dictionary) is simply a subset of  $\Sigma^*$  and  $L^*$  is simply a larger language of all the strings that we can create by concatenating strings from  $L$  (like compound words such as "mangoat" or "girlflower"). Thus, we have three sets of strings satisfying  $L \subseteq L^* \subseteq \Sigma^*$ .

**Problem 188.** Suppose  $\Sigma$  is an alphabet. Find a language  $L \subset \Sigma^*$  so that  $L^*$  is finite. There are only two such languages. Find the other one.

**Problem 189.** Suppose  $\Sigma$  is an alphabet and  $L$  is a language. Define a new language,

$$K = \{\lambda\} \cup \{w \in \Sigma^* \mid w = u_1 u_2 \dots u_n \text{ and } n \geq 1 \text{ and } u_1, \dots, u_n \in L\}$$

Does  $K = L^*$ ?

We now introduce a new set of objects, called **regular expressions**. A **regular expression** is a short hand for representing a language. If  $w$  is a regular expression, then  $L(w)$  will denote the language associated with  $w$ . We will begin by defining three regular expressions.

**Definition 53.** If  $\Sigma$  is a language and  $a \in \Sigma$  then each of  $\emptyset$ ,  $\lambda$  and  $a$  is a **regular expression**, where

<sup>1</sup>If you like this, please thank S. C. Kleene who developed the ideas of star and regular expressions. This is also called the Kleene star or Kleene closure

$\emptyset$  represents the language  $\emptyset$  or formally,  $L(\emptyset) = \emptyset$

$\lambda$  represents the language  $\{\lambda\}$  or formally,  $L(\lambda) = \{\lambda\}$

$a$  represents the language  $\{a\}$  or formally,  $L(a) = \{a\}$

We now define three operations on regular expressions which parallel the three operations on languages of union, concatenation and star.

**Definition 54.** Suppose that  $\Sigma$  is a language and  $\alpha$  and  $\beta$  are regular expressions.

1.  $\alpha\beta$  is a regular expression and denotes the concatenation of the languages associated with  $\alpha$  and  $\beta$ , formally  $L(\alpha\beta) = L(\alpha)L(\beta)$
2.  $\alpha \cup \beta$  is a regular expression and denotes the union of the languages associated with  $\alpha$  and  $\beta$ , formally  $L(\alpha \cup \beta) = L(\alpha) \cup L(\beta)$
3.  $\alpha^*$  is a regular expression and denotes the star of the language associated with  $\alpha$ , formally  $L(\alpha^*) = L(\alpha)^*$
4.  $(\alpha)$  is a regular expression and denotes the same language as  $\alpha$ , formally  $L((\alpha)) = L(\alpha)$

**Problem 190.** Suppose  $\Sigma = \{a, b, c\}$  is our alphabet.

1. What language does  $b^*$  represent? I.e.  $L(b^*) = \text{_____?}$
2. What language does  $(cc)^*$  represent? I.e.  $L((cc)^*) = \text{_____?}$
3. What language does  $b^*a^*$  represent? I.e.  $L(b^*a^*) = \text{_____?}$

**Problem 191.** Suppose  $\Sigma = \{a, b, c\}$  is our alphabet.

1. What language does  $a^* \cup c$  represent? I.e.  $L(a^* \cup c) = \text{_____?}$
2. What language does  $(a^* \cup c)^*$  represent? I.e.  $L((a^* \cup c)^*) = \text{_____?}$
3. Does  $(a^* \cup c)^* = (a \cup c)^*$ ?

**Problem 192.** If  $\alpha = (cc)^*(b^*(a^* \cup c)^*)$ , then using one rule per step from Definition 54 fill in the ... below and list which rule you used by each line.

$$\begin{aligned} L(\alpha) &= L((cc)^*)L(b^*)L((a^* \cup c)^*) \\ &= \dots \\ &= \{cc\}^* \{b\}^* \{a, c\}^*. \end{aligned}$$

**Problem 193.** Let  $\alpha = (cc)^*(b^*(a^* \cup c)^*)$ . List at least ten elements of  $L(\alpha)$  and describe (in English or Spanish or Gaelic) what types of strings are in this language.

**Definition 55.** A language that can be represented by a regular expression is called a **regular language** and the set of all regular languages over some alphabet is denoted by

$$\text{REG} = \{L(\alpha) \mid \alpha \text{ is a regular expression}\}.$$

**Problem 194.** Construct a FSA to recognize the language  $L(\alpha)$  where

$$\alpha = (010)^*(333)(1 \cup 4)^* \cup 0^*(3^* \cup 2^*).$$

**Problem 195.** If  $w \in \Sigma^*$ , we define  $w^R \in \Sigma^*$  recursively as follows.

1.  $\lambda^R = \lambda$  and  $a^R = a$  for all  $a \in \Sigma$ .
2. If  $w = au$  where  $a \in \Sigma$  and  $u \in \Sigma^*$ , then  $w^R = u^R a$ .

What string is  $w^R$  if  $w = tar$ ? if  $w = \text{senihcamdnasegaungal}$ ? Describe  $L$  in English.

**Problem 196.** Let  $\Sigma = \{a, b\}$ . Give examples of two strings in  $L$  and two strings not in  $L$  where  $L = \{w \in \Sigma^* \mid w^R w = w w^R\}$ .

**Problem 197.** Let  $\Sigma = \{a, b\}$ . Make a complete list of all of the strings in the finite language  $L = \{w \in \Sigma^* \mid w = uu^R u \text{ for some } u \in \Sigma^*\}$ .

**Problem 198.** Let  $\Sigma = \{0, 1\}$ . Find a regular expression representing the languages consisting of

1. all strings in  $\Sigma^*$  with no more than three 1's;
2. all strings in  $\Sigma^*$  with a number of 1's divisible by 3;
3. all strings in  $\Sigma^*$  not containing the substring 11.

**Problem 199.** Let  $\Sigma = \{a, b, c\}$ . Compute  $L(\alpha)$ , where  $\alpha = (a^* \cup b^* \cup c^*)^* b (a^* b^* c^*)^*$ , and give a simple verbal description of the strings in this language.

**Problem 200.** Let  $\Sigma = \{a, b\}$ . Describe a method to generate all of the strings in the language  $L = \{w \in \Sigma^* \mid www = uu \text{ for some } u \in \Sigma^*\}$ , and explain why your method is correct.

### Minimal Finite State Acceptors

The **number of internal states**,  $|Q|$ , of a finite state acceptor  $M = \langle \Sigma, Q, s, \delta, Y \rangle$  provides us with a useful measure of the **complexity** of  $M$ . Our problem is to find a method to generate from a regular language  $L$  a finite state acceptor  $M$  such that

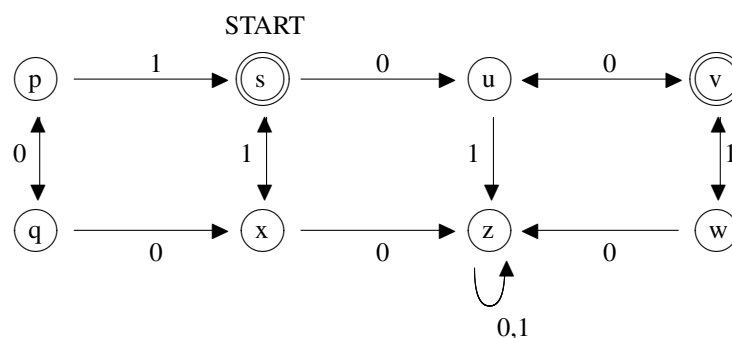


Figure 10.1: A Finite State Acceptor,  $M$

- $M$  recognizes  $L$  and
- no other FSA that recognizes  $L$  has less states than  $M$ .

The idea is to find a minimal FSA that recognizes the same language,  $L$ , as any given FSA.

**Problem 201.** Let  $L_0$  be the regular language recognized by the FSA in Figure 10.1. Note that this FSA has two accepting states,  $s$  and  $v$ . What is  $L_0$ ? That is, what strings does the above FSA accept?

**Problem 202.** Let  $L_0$  be the regular language recognized by the FSA in Figure 10.1. Construct an FSA  $M_0$  that also recognizes  $L_0$  but has only 4 states, one of which will be the dead state.

**Problem 203.** Let  $L_0$  be the regular language recognized by the FSA in Figure 10.1. Let  $M'_0$  be any finite state acceptor that recognizes  $L_0$ . Let  $s'$  be the start state. Explain why  $s'$ ,  $\delta(s', 0)$ ,  $\delta(s', 1)$  and  $\delta^*(s', 10) = \delta(\delta(s', 1), 0)$  must all be distinct states.

You have just shown that any FSA  $M'_0$  that recognizes  $L$  has at least as many states as  $M_0$  that you constructed in Problem 202. Thus you have constructed a minimal FSA for  $M$  in Figure 10.1. Our goal is to find a general procedure that will produce, from any regular language  $L$ , a minimal FSA  $M$  that recognizes  $L$ .

**Definition 56.** Let  $L \subseteq \Sigma^*$  be a language and  $w \in \Sigma^*$ . We define the **right-set for  $w$**  to be  $R_w = \{z \in \Sigma^* \mid wz \in L\}$ .

If  $w$  is not a prefix of a string in  $L$ , then  $R_w = \emptyset$ .

**Definition 57.** The set of all strings that are not a prefix of any member of  $L$  is denoted by  $Z(L) = \{w \in \Sigma^* \mid R_w = \emptyset\}$ .

**Definition 58.** Let  $L \subseteq \Sigma^*$  be a language. If  $x, y \in L$  then we write  $x \equiv_L y$  iff  $R_x = R_y$ .

**Problem 204.** Show that if  $L \subseteq \Sigma^*$  is any language then  $x \equiv_L y$  is an equivalence relation on  $L$ .

**Problem 205.** In Problem 201 you computed  $L_0$ . For the language  $L_0$ , describe the elements of  $\Sigma^*$  that are in each of these right-sets.

1.  $R_0$
2.  $R_{110}$
3.  $R_{00110}$
4.  $R_{01}$
5.  $R_{10}$

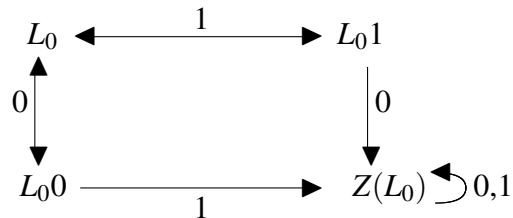
**Problem 206.** In Problem 201 you computed  $L_0$ . Describe each of these sets.

1.  $0L_0 = \{0u \mid u \in L_0\}$
2.  $L_00 = \{u0 \mid u \in L_0\}$
3.  $1L_0 = \{1u \mid u \in L_0\}$
4.  $L_01 = \{u1 \mid u \in L_0\}$
5.  $[0]$
6.  $[11]$

For each  $w \in \{0, 1\}^*$ , the right-set  $R_w$  is either  $L_0$ ,  $0L_0$ ,  $1L_0$  or  $\emptyset$ . These choices give us exactly four different  $\equiv_{L_0}$ -classes:

$$[\lambda] = L_0 \quad [0] = L_00 \quad [1] = L_01 \quad [01] = \Sigma^* \sim (L_0 \cup L_00 \cup L_01) = Z(L_0).$$

If any member of  $L_00$  is extended on the right by adding a 0, the result is always a string in  $L_0$ . If any member of  $L_00$  is extended on the right with a 1 instead, we get a string in  $Z(L_0)$ . These facts are summarized in the following diagram.



Since  $\lambda \in L_0$ , we can take any string  $w \in \{0, 1\}^*$  and find which class  $w$  is in by tracing the diagram, building  $w$  by starting with  $\lambda$  and successively adding 0s or 1s on the right until we have  $w$ . Our ending point will be the  $\equiv_{L_0}$ -class of  $w$ . If we land in  $L_0$ , then  $w \in L_0$ ; otherwise it is not. What we have done is to build a four state FSA, whose start state and only accepting state is  $L_0$ , that recognizes  $L_0$ !

**Problem 207.** Let  $L = 0^*1^*2^*$ . Compute enough right-sets to determine the distinct  $\equiv_L$ -classes and construct the standard FSA for  $L$ .

**Problem 208.** Let  $L = 00(1^* \cup 2^*)$ . Compute enough right-sets to determine the distinct  $\equiv_L$ -classes and construct the standard FSA for  $L$ .

**Problem 209.** Let  $L = (1^* \cup 2^*)00$ . Compute enough right-sets to determine the distinct  $\equiv_L$ -classes and construct the standard FSA for  $L$ .

**Problem 210.** Let  $L = 111 \cup (020)^*$ . Compute enough right-sets to determine the distinct  $\equiv_L$ -classes and construct the standard FSA for  $L$ .

**Problem 211.** Let  $L = (111)^*1$ . Compute enough right-sets to determine the distinct  $\equiv_L$ -classes and construct the standard FSA for  $L$ .

Can we repeat this construction with any language  $L$ , using the  $\equiv_L$ -classes to find an FSA for  $L$ ? There is one important hitch, as the following example shows.

**Problem 212.** Let  $L_{=} = \{a^n b^n \mid n = 1, 2, 3, \dots\}$ . Show that there are infinitely many  $\equiv_{L_{=}}$ -classes by making an infinite list of strings, no two of which are equivalent by  $\equiv_{L_{=}}$ .

Obviously, we cannot use the  $\equiv_L$ -classes as states of a *finite* state acceptor if there are infinitely many of them! But if the set of  $\equiv_L$ -classes is finite, then this construction will always work.

**Theorem 23. Standard FSA Theorem** Suppose that  $L \subseteq \Sigma^*$  is a language such that  $\Sigma^*$  has only finitely many different  $\equiv_L$ -classes. Then  $L$  is recognized by an FSA which has exactly as many states as there are  $\equiv_L$ -classes in  $\Sigma^*$ .

Proof: Let  $M = \langle \Sigma, Q, s, Y, \delta \rangle$  where

- $Q$  is the (finite) set of all  $\equiv_L$ -classes;
- $s = [\lambda]$  is the start state;
- $Y = \{[x] \mid x \in L\}$  (which is unambiguously defined since you can check that, for all  $x, y \in \Sigma^*$ , if  $x \in L$  and  $x \equiv_L y$ , then  $y \in L$ );

- $\delta([x], a) = [xa]$  (which is unambiguously defined since you can check that, for all  $x, y \in \Sigma^*$  and  $a \in \Sigma$ , if  $x \equiv_L y$ , then  $xa \equiv_L ya$ ).

This defines an FSA  $M$  with exactly as many states as there are  $\equiv_L$ -classes (since its states *are* the  $\equiv_L$ -classes!). To see which strings  $w$  are accepted by  $M$ , we need to know when  $\delta^*(s, w) \in Y$ . If  $w = a_1a_2 \dots a_n \in \Sigma^*$ , then

$$\begin{aligned} \delta^*(s, w) &= \delta^*([\lambda], a_1a_2 \dots a_n) = \delta^*(\delta([\lambda], a_1), a_2 \dots a_n) \\ &= \delta^*([a_1], a_2 \dots a_n) = \delta^*([a_1a_2], \dots a_n) = \dots = [a_1a_2 \dots a_n] = [w]. \end{aligned}$$

Thus  $\delta^*(s, w) \in Y$  if and only if  $w \in L$ , so  $M$  recognizes  $L$ .  $\square$

The FSA  $M$  constructed in this proof is called the **standard** FSA for  $L$  as it is constructed in a uniform way from the language  $L$  itself.

In case  $L$  is a regular language, the relation  $\equiv_L$  is very special. To see this, let  $M = \langle \Sigma, Q, s, \delta, Y \rangle$  be an arbitrary FSA which recognizes  $L$ . We now define a second equivalence relation on  $\Sigma^*$ , this one *based on the machine*  $M$ . For  $x, y \in \Sigma^*$ ,

$$x \equiv_M y \quad \text{if} \quad \delta^*(s, x) = \delta^*(s, y).$$

Thus two strings are equivalent if  $M$  lands in the same state reading one as reading the other. A state  $q$  of an FSA  $M$  is said to be **accessible** if there is *some* string  $x$  such that  $\delta^*(s, x) = q$ , that is,  $M$  can reach  $q$  from the start state. For each accessible state  $q$  we have a distinct  $\equiv_M$ -class consisting of all strings  $x$  such that  $\delta^*(s, x) = q$ .

**Theorem 24. Myhill-Nerode Theorem** *Let  $L$  be a regular language. Then the set of  $\equiv_L$ -classes is finite, and every FSA that recognizes  $L$  has at least as many states as the standard FSA for  $L$  has.*

Proof: Let  $M$  be an FSA that recognizes  $L$  and let  $q_1, q_2, \dots, q_n$  be a list of the accessible states of  $M$ . Then  $K_{q_1}, K_{q_2}, \dots, K_{q_n}$  is a list of all of the  $\equiv_M$ -classes. Since  $M$  has at least  $n$  states, it will suffice to show that the number of  $\equiv_L$ -classes in  $\Sigma$  is no more than  $n$ . We will do this by showing that each  $\equiv_L$ -class is a union of  $\equiv_M$ -classes. (This fact is sometimes stated by saying that the partition induced by  $\equiv_M$  is a **refinement** of the partition induced by  $\equiv_L$ .)

Let  $x \equiv_M y$ . We must show that  $x \equiv_L y$ . Since  $x \equiv_M y$ , we have, for all  $z \in \Sigma^*$ , that  $\delta^*(s, x) = \delta^*(s, y)$  and consequently

$$\delta^*(s, xz) = \delta^*(\delta^*(s, x), z) = \delta^*(\delta^*(s, y), z) = \delta^*(s, yz).$$

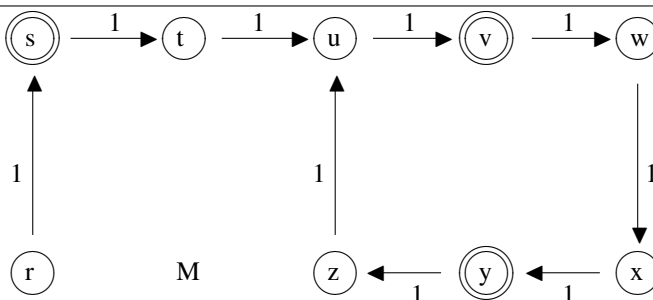


Figure 10.2: FSA for  $L = (111)^*1$

Thus  $\delta^*(s, xz) \in Y$  if and only if  $\delta^*(s, yz) \in Y$ , and therefore  $xz \in L$  if and only if  $yz \in L$ . But this tells us that  $x \equiv_L y$ .  $\square$

Our original finite state acceptor  $M$  for  $L_0$ , for example, has six accessible states  $s, u, v, w, z$  and  $x$  which give us a partition of each of the  $\equiv_{L_0}$ -classes into  $\equiv_M$ -classes:  $L_0 = K_s \cup K_v$ ,  $L_01 = K_x \cup K_w$ ,  $L_00 = K_u$  and  $Z(L_0) = K_z$ .

**Problem 213.** Figure 10.2 is an FSA  $M$  for the language  $L = (111)^*1$ . Find a regular expression for each  $\equiv_M$ -class:  $K_r, K_s, K_t, K_u, K_v, K_w, K_x, K_y$  and  $K_z$ .

**Problem 214.** Express each  $\equiv_L$ -classes as a union of  $\equiv_M$ -classes, thereby illustrating the proof that  $M$  must have at least as many states as there are  $\equiv_L$ -classes.

### 10.1 Project: Languages that are not Regular

Combining the two previous theorems gives us a very useful characterization of regular languages which will allow us to say which languages are *not* regular.

**Corollary 1. MYHILL-NERODE COROLLARY:** A language  $L \subseteq \Sigma^*$  is regular if and only if the set of  $\equiv_L$ -classes in  $\Sigma^*$  is finite.

Proof: If  $L$  is regular, then the M-N Theorem tells us that the set of  $\equiv_L$ -classes is finite. If the set of  $\equiv_L$ -classes is finite, then the Standard FSA Theorem shows us how to construct the standard FSA for  $L$  which guarantees that  $L$  is a regular language.  $\square$

For example, this corollary tells us for certain that the language  $L_ =$  in Problem 212 is *not* a regular language because there are infinitely many  $\equiv_{L_ =}$ -classes. Using the same ideas we can formulate a direct explanation as to why this and certain other languages are not regular without reference to equivalence relations.

**Example 14.** The language  $L_{=} = \{a^n b^n \mid n = 1, 2, 3, \dots\}$  is not regular.

Let  $M$  be any FSA. We will show that  $M$  does not recognize  $L_{=}$  by finding two strings,  $u \in L_{=}$  and  $v \notin L_{=}$ , such that  $M$  ends in the same state reading either  $u$  or  $v$ . Imagine that we ask  $M$  to read strings of  $a$ 's:  $a, a^2, a^3, a^4, \dots$ , starting in its start state  $s$ . Let  $q_n = \delta^*(s, a^n)$  be its ending state when it reads  $a^n$ . Since  $M$  has only finitely many states, there must be two different indices  $m < n$  such that  $q_m = q_n$ . Taking  $u = a^m b^m$  and  $v = a^n b^m$ , we have

$$\delta^*(s, u) = \delta^*(s, a^m b^m) = \delta^*(q_m, b^m) = \delta^*(q_n, b^m) = \delta^*(s, a^n b^m) = \delta^*(s, v).$$

This tells us that  $M$  either accepts both  $u \in L_{=}$  and  $v \notin L_{=}$ , or it accepts neither. In either case,  $M$  is not an FSA for  $L_{=}$ .  $\square$

**Example 15.** The language  $L_{alg}$  of all meaningful algebraic expressions over  $\Sigma = \{w, x, y, z, (, ), +, -, \cdot, \div\}$  is not regular.

Again let  $M$  be any FSA. Notice that, in a string  $w \in L_{alg}$ , the number of '('s and ')'s must be exactly the same. This time imagine  $M$  beginning in its start state  $s$  reading  $(, ((, (((, ((((, \dots$ . Again there must be  $m < n$  such that  $M$  is in the same state after reading  $m$  '('s as after reading  $n$  '('s. Let  $u = ((\dots((x+x)+x)\dots+x)+x) \in L_{alg}$  where there are  $n$  '('s before the first 'x+', followed by  $n$  '+x's. Now let  $v$  be obtained from  $u$  by removing the first  $n - m$  '('s leaving only  $m$  '('s. Then  $M$ , starting in  $s$ , will end in the same state reading  $u$  or reading  $v$ , so it either accepts both or rejects both. Thus  $M$  does not recognize  $L_{alg}$ .

For the next problems, use an argument like the ones given above to show that each of the following languages is not regular.

**Problem 215.**  $L = \{0^n 10^n \mid n = 1, 2, 3, \dots\}$

**Problem 216.**  $L = \{wcw \mid w \in \{a, b\}^*\}$

**Problem 217.**  $L_{re}$  is the set of regular expressions over the alphabet  $\{a, b, c\}$ .

**Problem 218.**  $L \subseteq \{0, 1\}$  is the set of  $w$  containing the same number of  $a$ 's as  $b$ 's.