Computer Algorithms

This section provides a more complete description of an algorithm than given above, as well as an example algorithm for determining the day of the week for a given date.

Part I: What Is an Algorithm?

An **algorithm** is a finite number of clearly described, unambiguous “do-able” steps that can be systematically followed to produce a desired result for given input in a finite amount of time (that is, it eventually terminates). Algorithms solve general problems (determining whether any given number is a prime number), and not specific ones (determining whether 30753 is a prime number). Algorithms, therefore, are general computational methods used for solving particular problem instances.

The word “algorithm” is derived from the ninth-century Arab mathematician, Al-Khwarizmi, who worked on “written processes to achieve some goal.” (The term “algebra” also derives from the term “al-jabr,” which he introduced.)

Computer algorithms are central to computer science. They provide step-by-step methods of computation that a machine can carry out. Having high-speed machines (computers) that can consistently follow and execute a given set of instructions provides a reliable and effective means of realizing computation.

However, the computation that a given computer performs is only as good as the underlying algorithm used. Understanding what can be effectively programmed and executed by computers, therefore, relies on the understanding of computer algorithms.

Part II: Algorithms and Computers: A Perfect Match

Much of what has been learned about algorithms and computation since the beginnings of modern computing in the 1930s–1940s could have been studied centuries ago, since the study of algorithms does not depend on the existence of computers. The algorithm for performing long division is such an example. However, most algorithms are not as simple or practical to apply manually. Most require the use of computers either because they would require too much time

for a person to apply, or involve so much detail as to make human error likely. Because computers can execute instructions very quickly and reliably without error, algorithms and computers are a perfect match!

Below is an example algorithm for determining the day of the week for any date between January 1, 1800 and December 31, 2099.

To determine the day of the week for a given **month**, **day**, and **year**:

1. Let **century_digits** be equal to the first two digits of the year.
2. Let **year_digits** be equal to the last two digits of the year.
3. Let **value** be equal to **year_digits** + floor(**year_digits** / 4)
4. If **century_digits** equals 18, then add 2 to **value**, else
   if **century_digits** equals 20, then add 6 to **value**.
5. If the **month** is equal to January and **year** is not a leap year,
   then add 1 to **value**, else,
   if the **month** is equal to February and the **year** is a leap year, then
   add 3 to **value**; if not a leap year, then add 4 to **value,** else,
   if the **month** is equal to March or November, then add 4 to **value**, else,
   if the **month** is equal to May, then add 2 to **value**, else,
   if the **month** is equal to June, then add 5 to **value**, else,
   if the **month** is equal to August, then add 3 to **value**, else,
   if the **month** is equal to October, then add 1 to **value**, else,
   if the **month** is equal to September or December, then add 6 to **value**,
6. Set **value** equal to (**value** + **day**) mod 7.
7. If **value** is equal to 1, then the day of the week is Sunday; else
   if **value** is equal to 2, day of the week is Monday; else
   if **value** is equal to 3, day of the week is Tuesday; else
   if **value** is equal to 4, day of the week is Wednesday; else
   if **value** is equal to 5, day of the week is Thursday; else
   if **value** is equal to 6, day of the week is Friday; else
   if **value** is equal to 0, day of the week is Saturday

Note that there is no value to add for the months of April and July.

1. Use the algorithm above to find the day of the week for June 21, 2079.
2. Use the algorithm above to find the day of the week for your birthday in 2085.
3. Use the algorithm above to find the day of the week for your birthday in 1803.

Concepts and Procedures

1. Which of the following are true of an algorithm?
    a. Has a finite number of steps
    b. Produces a result in a finite amount of time
    c. Solves a general problem
    d. All of the above

2. Algorithms were first developed in the 1930–1940s when the first computing machines appeared. (TRUE/FALSE)

3. Algorithms and computers are a “perfect match” because: (Select all that apply.)
    a. Computers can execute a large number of instructions very quickly.
    b. Computers can execute instructions reliably without error.
    c. Computers can determine which algorithms are the best to use for a given problem.

4. Given that the year 2016 is a leap year, what day of the week does April 15th of that year fall on? Use the algorithm in Figure 1-8 for this.

5. Which of the following is an example of an algorithm? (Select all that apply.)
    a. A means of sorting any list of numbers
    b. Directions for getting from your home to a friend’s house
    c. A means of finding the shortest route from your house to a friend’s house.

Problem Solving

1. Using the *Day of the Week* algorithm, show all steps for determining the day of the week for January 24, 2018. (Note that 2018 is not a leap year.)

2. Using the algorithm, determine the day of the week on which you were born.

3. Suppose that an algorithm was needed for determining the day of the week for dates that only occur within the years 2000–2099. Simplify the day of the week algorithm in Figure 1-8 as much as possible by making the appropriate changes.

4. As precisely as possible, give a series of steps (an algorithm) for doing long addition.