Mathematical Preliminaries

2

In this chapter we'll review some mathematical concepts that will be used throughout this course. We'll also learn some new mathematical notations and techniques that are important for analysis of algorithms.

2.1 Floors and Ceilings

In doing computer computations, we sometimes have to discard the fractional part of a real number x, while in other cases we may have to round upward to the first integer larger than or equal to x. It is convenient to have simple mathematical notations that indicate these operations. The "floor" and "ceiling" notations were invented for this purpose.

Notations. We shall use $\mathbf{N}$ for the set of cardinal or natural numbers, $\mathbf{N} = \{0, 1, 2, \ldots\}$, $\mathbf{Z}$ for the set of integers, $\mathbf{Q}$ for the set of rational numbers, and $\mathbf{R}$ for the set of real numbers.

Definition 2.1.1 The *floor* of a real number x, denoted by $\lfloor x \rfloor$, is the largest integer that is less than or equal to x. The *ceiling* of a real number x, denoted by $\lceil x \rceil$, is the smallest integer that is greater than or equal to x. We can express these definitions in the following way:

$$\lfloor x \rfloor = \max\{k : k \in \mathbf{Z}, k \leq x\};$$
$$\lceil x \rceil = \min\{k : k \in \mathbf{Z}, k \geq x\}.$$

Examples 2.1.2

$$\lfloor 3.7 \rfloor = 3;\ \lfloor 0.04 \rfloor = 0;\ \lfloor -0.04 \rfloor = -1;\ \lfloor 6 \rfloor = 6;\ \lfloor 22.104 \rfloor = 22;$$
$$\lceil 3.7 \rceil = 4;\ \lceil 0.04 \rceil = 1;\ \lceil -0.04 \rceil = 0;\quad \lceil 6 \rceil = 6;\ \lceil 22.104 \rceil = 23.$$

D. Vrajitoru and W. Knight, *Practical Analysis of Algorithms*,
Undergraduate Topics in Computer Science, DOI 10.1007/978-3-319-09888-3_2

In a C or C++ program, an expression of the form `n/m` indicates *integer division* when `n` and `m` are of an integer data type. The mathematical expression $\left\lfloor \frac{n}{m} \right\rfloor$ denotes the exact value of the C/C++ expression `n/m` when `n` and `m` are positive. Thus, when reasoning about programs in which integer division is performed, we will make repeated use of expressions such as $\left\lfloor \frac{n}{m} \right\rfloor$.

If you consult the documentation for the C/C++ "`math.h`" or "`cmath`" library header file, you will find that it contains the following two prototypes:

```
double ceil (double x);
double floor(double x);
```

Note that the values returned by these two functions are of type `double`, not `long`. This may surprise you until you realize that floating point real numbers can be *far* larger than the integer `LONG_MAX`, and so in order to have expressions such as `floor(3.182e19)` return a sensible value, the function must return the integer `31820000000000000` in floating point form.

The following lemma gives us a property of integers that we will use often in this chapter.

Lemma 2.1.3 *Let n and m be any integer numbers. If $n < m$, then $n + 1 \leq m$ and $n \leq m - 1$.*

Proof Suppose $n < m$. If $n + 1$ were greater than m, then m would be an integer strictly between n and $n + 1$. That is impossible. Thus $n + 1$ must be less than or equal to m. It follows from $n + 1 \leq m$ that $n \leq m - 1$. ■

The following theorem lists some properties of floor and ceiling that we will use throughout in this book. It gives us a better idea about how the floor and the ceiling relate to their argument and to each other.

Theorem 2.1.4 *Let x and y be any real numbers. Let n, m, and k be any integers.*

(a) $\lfloor x \rfloor \leq x < \lfloor x \rfloor + 1$*; equivalently,* $x - 1 < \lfloor x \rfloor \leq x$.
(b) $\lceil x \rceil - 1 < x \leq \lceil x \rceil$*; equivalently,* $x \leq \lceil x \rceil < x + 1$.
(c) *If* $n \leq x < n + 1$ *(or* $x - 1 < n \leq x$*), then* $n = \lfloor x \rfloor$*; similarly, if* $n - 1 < x \leq n$ *(or* $x \leq n < x + 1$*), then* $n = \lceil x \rceil$.
(d) *If* $x \leq y$*, then* $\lfloor x \rfloor \leq \lfloor y \rfloor$ *and* $\lceil x \rceil \leq \lceil y \rceil$*. Similarly for* $x \geq y$.
(e) *If* $n \leq x$*, then* $n \leq \lfloor x \rfloor$*. If* $n \geq x$ *then* $n \geq \lceil x \rceil$.
(f) *If x has an integer value, then* $\lfloor x \rfloor = x = \lceil x \rceil$*. If x has a non-integer value, then* $\lceil x \rceil = \lfloor x \rfloor + 1$.
(g) $\lfloor (-1)x \rfloor = (-1)\lceil x \rceil$ *and* $\lceil (-1)x \rceil = (-1)\lfloor x \rfloor$.

Proof (a) Take any real number x. The inequality $\lfloor x \rfloor \leq x$ is true because the floor of x is an element of a set of integers less than or equal to x. The inequality $x < \lfloor x \rfloor + 1$ is true because $\lfloor x \rfloor$ is the *largest* of the integers less than or equal

to x, and $\lfloor x \rfloor + 1$ is an integer larger than $\lfloor x \rfloor$, so $\lfloor x \rfloor + 1$ *cannot* be in the set of integers less than or equal to x. Thus $\lfloor x \rfloor + 1$ is greater than x. This completes the proof that $\lfloor x \rfloor \leq x < \lfloor x \rfloor + 1$. To prove that $x - 1 < \lfloor x \rfloor \leq x$, we need only prove that $x - 1 < \lfloor x \rfloor$ because we have already proved that $\lfloor x \rfloor \leq x$. Take the inequality $x < \lfloor x \rfloor + 1$, which we have already proved, and subtract 1 from both sides. This gives $x - 1 < \lfloor x \rfloor$.

(b) Take any real number x. The inequality $x \leq \lceil x \rceil$ is true because the ceiling of x is an element of a set of integers greater than or equal to x. To prove the inequality $\lceil x \rceil - 1 < x$, first note that $\lceil x \rceil$ is the *smallest* of the integers greater than or equal to x, and $\lceil x \rceil - 1$ is smaller than $\lceil x \rceil$, so $\lceil x \rceil - 1$ *cannot* be in the set of integers greater than or equal to x. Thus $\lceil x \rceil - 1$ is smaller than x. This completes the proof that $\lceil x \rceil - 1 < x \leq \lceil x \rceil$. The inequalities $x \leq \lceil x \rceil < x + 1$ are proved by taking apart the inequalities above, adding 1 to both sides of $\lceil x \rceil - 1 < x$, and then putting the pieces back together again in a different order.

(c) Let x be any real number, and let n be an integer satisfying $n \leq x < n+1$. Then n must be the largest of the integers that are less than or equal to x because by Lemma 2.1.3 every integer greater than n is at least $n + 1$. By definition of $\lfloor x \rfloor$, it follows that $n = \lfloor x \rfloor$. The second assertion has a similar proof.

(d) Suppose $x \leq y$. We'll prove that $\lfloor x \rfloor \leq \lfloor y \rfloor$ and leave the other assertions as exercises. First note that either $x < \lfloor y \rfloor$ or $\lfloor y \rfloor \leq x$. If $x < \lfloor y \rfloor$, then we can combine this with the inequality $\lfloor x \rfloor \leq x$ (see part (a)) to conclude that $\lfloor x \rfloor < \lfloor y \rfloor$. If, instead, $\lfloor y \rfloor \leq x$, we can combine this with $x \leq y$ and $y < \lfloor y \rfloor + 1$ to obtain $\lfloor y \rfloor \leq x < \lfloor y \rfloor + 1$. Since $\lfloor y \rfloor$ is an integer, it follows from part (c) that $\lfloor x \rfloor = \lfloor y \rfloor$.

(e) Suppose $n \leq x$. Then by (d), $\lfloor n \rfloor \leq \lfloor x \rfloor$. Since n has an integer value, $\lfloor n \rfloor = n$, which means that $n \leq \lfloor x \rfloor$.
The proof of the second part is similar.

(f) Suppose x has an integer value. Then trivially x belongs to the set $\{k : k \in \mathbf{Z}, k \leq x\}$ and is the largest element of the set, so $x = \lfloor x \rfloor$. Similarly, $x = \lceil x \rceil$. Now suppose x is not an integer. We will prove that $\lceil x \rceil = \lfloor x \rfloor + 1$ by showing that $\lceil x \rceil \leq \lfloor x \rfloor + 1$ and $\lceil x \rceil \geq \lfloor x \rfloor + 1$. Since x is not an integer, then $\lfloor x \rfloor < x < \lceil x \rceil$, from which we get $\lfloor x \rfloor < \lceil x \rceil$. Since both $\lfloor x \rfloor$ and $\lceil x \rceil$ are integers, we can use Lemma 2.1.3 to conclude that $\lfloor x \rfloor + 1 \leq \lceil x \rceil$. By part (a): $x < \lfloor x \rfloor + 1$. Since $\lfloor x \rfloor + 1$ is an integer, we can apply (e) to deduce that $\lceil x \rceil \leq \lfloor x \rfloor + 1$. From the two inequalities that we have proved, we can conclude that $\lceil x \rceil = \lfloor x \rfloor + 1$.

(g) Let us start with the second inequality from (a): $\lceil x \rceil - 1 < x \leq \lceil x \rceil$. Multiply through by -1 to reverses the inequalities:

$$(-1)(\lceil x \rceil - 1) > (-1)x \geq (-1)(\lceil x \rceil).$$

This can be rewritten from right to left as

$$(-1)\lceil x \rceil \leq (-1)x < (-1)\lceil x \rceil + 1.$$

We can see that the integer $(-1)\lceil x \rceil$ satisfies the first part of (c) with respect to the real number $(-1)x$, and thus it must be the floor of this number: $(-1)\lceil x \rceil = \lfloor (-1)x \rfloor$. The second part is proved in a similar way. ■

The following theorem shows some properties of the floor and ceiling functions in relation to arithmetic operations applied to their arguments.

Theorem 2.1.5 *Let x and y be any real numbers. Let n, m, and k be any integers.*

(a) $\lfloor x \pm n \rfloor = \lfloor x \rfloor \pm n$ *and* $\lceil x \pm n \rceil = \lceil x \rceil \pm n$*; but in general,* $\lfloor nx \rfloor \neq n\lfloor x \rfloor$ *and* $\lceil nx \rceil \neq n\lceil x \rceil$ *and* $\lfloor x + y \rfloor \neq \lfloor x \rfloor + \lfloor y \rfloor$ *and* $\lceil x + y \rceil \neq \lceil x \rceil + \lceil y \rceil$.

(b) $\left\lfloor \frac{n}{2} \right\rfloor + \left\lceil \frac{n}{2} \right\rceil = n$.

(c) $\left\lfloor \frac{n+1}{2} \right\rfloor = \left\lceil \frac{n}{2} \right\rceil$*; equivalently,* $\left\lfloor \frac{n-1}{2} \right\rfloor + 1 = \left\lceil \frac{n}{2} \right\rceil$.

(d) *If* $k > 0$*, then* $\left\lfloor \frac{\lfloor y \rfloor}{k} \right\rfloor = \left\lfloor \frac{y}{k} \right\rfloor$ *and* $\left\lceil \frac{\lceil y \rceil}{k} \right\rceil = \left\lceil \frac{y}{k} \right\rceil$*. In particular, if* $m \neq 0$ *we can put* $y = \frac{n}{m}$ *to get these identities:* $\left\lfloor \frac{\lfloor \frac{n}{m} \rfloor}{k} \right\rfloor = \left\lfloor \frac{n}{mk} \right\rfloor$ *and* $\left\lceil \frac{\lceil \frac{n}{m} \rceil}{k} \right\rceil = \left\lceil \frac{n}{mk} \right\rceil$.

Proof (a) Take any particular integer n. We'll prove that $\lfloor x + n \rfloor = \lfloor x \rfloor + n$. Since $\lfloor x \rfloor + n$ is an integer, by Theorem 2.1.4 (c) it suffices to prove that $(\lfloor x \rfloor + n) \leq x + n < (\lfloor x \rfloor + n) + 1$. These two inequalities are logically equivalent to $(\lfloor x \rfloor + n) - n \leq x + n - n < (\lfloor x \rfloor + n) + 1 - n$, which are in turn equivalent to $\lfloor x \rfloor \leq x < x + 1$, which are true for all x (see Theorem 2.1.4 (a)). This completes the proof for floors.

The proof for ceilings is similar. The equation $\lfloor x - n \rfloor = \lfloor x \rfloor - n$ is proved by noting that $\lfloor x - n \rfloor = \lfloor x + (-n) \rfloor = \lfloor x \rfloor + (-n) = \lfloor x \rfloor - n$ because $-n$ is an integer.

The assertions involving $\lfloor nx \rfloor$ and $\lceil nx \rceil$ are demonstrated by using examples: $\lfloor 4 \cdot 0.5 \rfloor \neq 4\lfloor 0.5 \rfloor$ and $\lceil 4 \cdot 0.5 \rceil \neq 4\lceil 0.5 \rceil$.

(b) Take any integer n and real number x. If n is even, then $n = 2m$ for some integer m, and thus $\left\lfloor \frac{n}{2} \right\rfloor + \left\lceil \frac{n}{2} \right\rceil$

$= \left\lfloor \frac{2m}{2} \right\rfloor + \left\lceil \frac{2m}{2} \right\rceil = \lfloor m \rfloor + \lceil m \rceil = m + m = 2m = n$.

If n is odd, then $n = 2m + 1$ for some integer m, and thus $\left\lfloor \frac{n}{2} \right\rfloor + \left\lceil \frac{n}{2} \right\rceil = \left\lfloor \frac{2m+1}{2} \right\rfloor + \left\lceil \frac{2m+1}{2} \right\rceil = \left\lfloor m + \frac{1}{2} \right\rfloor + \left\lceil m + \frac{1}{2} \right\rceil$. By part (a) of this theorem, this last expression equals $m + \left\lfloor \frac{1}{2} \right\rfloor + m + \left\lceil \frac{1}{2} \right\rceil = m + 0 + m + 1 = 2m + 1 = n$.

(c) Take any integer n. If n is even, then $n = 2m$ for some integer m, so $\left\lfloor \frac{n+1}{2} \right\rfloor = \left\lfloor \frac{2m+1}{2} \right\rfloor = \left\lfloor m + \frac{1}{2} \right\rfloor = m + \left\lfloor \frac{1}{2} \right\rfloor = m + 0 = m$ and $\left\lceil \frac{n}{2} \right\rceil = \left\lceil \frac{2m}{2} \right\rceil = \lceil m \rceil = m$ also, so $\left\lfloor \frac{n+1}{2} \right\rfloor = \left\lceil \frac{n}{2} \right\rceil$.

If n is odd, then $n = 2m + 1$ for some integer m, and thus $\left\lfloor \frac{n+1}{2} \right\rfloor = \left\lfloor \frac{2m+2}{2} \right\rfloor = \lfloor m + 1 \rfloor = m + 1$, and $\left\lceil \frac{n}{2} \right\rceil = \left\lceil \frac{2m+1}{2} \right\rceil = \left\lceil m + \frac{1}{2} \right\rceil = m + \left\lceil \frac{1}{2} \right\rceil = m+1$ also, and thus $\left\lfloor \frac{n+1}{2} \right\rfloor = \left\lceil \frac{n}{2} \right\rceil$. The fact that $\left\lfloor \frac{n-1}{2} \right\rfloor + 1 = \left\lfloor \frac{n+1}{2} \right\rfloor$ follows from part (a) and simple algebra.

(d) Take any real number y and any integer $k > 0$. By definition of the floor, $\left\lfloor \frac{y}{k} \right\rfloor$ is an integer. To prove that $\left\lfloor \frac{\lfloor y \rfloor}{k} \right\rfloor = \left\lfloor \frac{y}{k} \right\rfloor$ it suffices by Theorem 2.1.4 (c) to prove that the integer $\left\lfloor \frac{y}{k} \right\rfloor$ satisfies the inequalities $\left\lfloor \frac{y}{k} \right\rfloor \le \frac{\lfloor y \rfloor}{k} < \left\lfloor \frac{y}{k} \right\rfloor + 1$ (take x to be $\frac{\lfloor y \rfloor}{k}$ and n to be $\left\lfloor \frac{y}{k} \right\rfloor$).

First, let's prove $\left\lfloor \frac{y}{k} \right\rfloor \le \frac{\lfloor y \rfloor}{k}$. By Theorem 2.1.4 (a), $\left\lfloor \frac{y}{k} \right\rfloor \le \frac{y}{k}$, so $\left\lfloor \frac{y}{k} \right\rfloor k \le y$. Since $\left\lfloor \frac{y}{k} \right\rfloor k$ is the product of two integers, it is an integer, and it is less or equal to y, so $\left\lfloor \frac{y}{k} \right\rfloor k \le \lfloor y \rfloor$ by Theorem 2.1.4 (e). That proves $\left\lfloor \frac{y}{k} \right\rfloor \le \frac{\lfloor y \rfloor}{k}$.

To prove that $\frac{\lfloor y \rfloor}{k} < \left\lfloor \frac{y}{k} \right\rfloor + 1$, we note that $\left\lfloor \frac{y}{k} \right\rfloor + 1 > \frac{y}{k}$ by Theorem 2.1.4 (a), and $\frac{y}{k} \ge \frac{\lfloor y \rfloor}{k}$ because $y \ge \lfloor y \rfloor$ by Theorem 2.1.4 (a). Thus $\left\lfloor \frac{y}{k} \right\rfloor + 1 > \frac{\lfloor y \rfloor}{k}$, which is what we wanted to prove (in turned-around form). The proof of the assertion about ceilings is quite similar and will be omitted here. ■

As an example of where some of the formulas in Theorems 2.1.4 and 2.1.5 are useful, consider what happens during a binary search of a subarray `a[first..last]`. Here is a fragment of the code for the search (for the complete function see Fig. 5.3, p. 176):

```
while (first <= last && !found) // The value parameters first and last will
{    // be modified during execution of this code
  int mid = (first + last)/2;
  if (target < a[mid])
    .......
```

A similar calculation of `mid = (first + last)/2` occurs during the Merge Sort algorithm, which splits an array into two pieces, sorts them recursively, and then merges them back together.

When analyzing such algorithms, it is useful to have mathematical expressions for the lengths of the following three smaller subarrays

`a[first..mid-1]` `a[first..mid]` `a[mid+1..last]`

in terms of the length of the larger subarray `a[last]`.

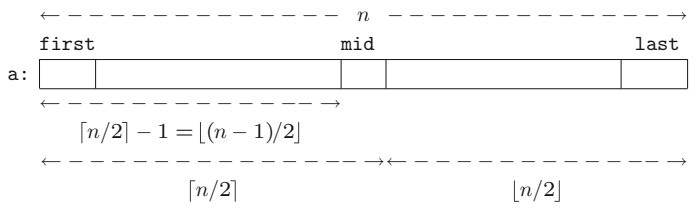

Fig. 2.1 Theorem 2.1.6

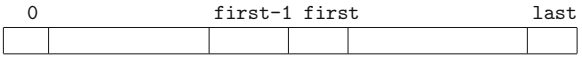

Fig. 2.2 Theorem 2.1.6 a

Theorem 2.1.6 *Consider a subarray* `a[first..last]` *of some array* `a`*, where* $0 \leq$ `first` $\leq$ `last`*. Let n denote the length of the subarray, and let* `mid` *denote the quantity* `(first + last) / 2` *(this is the average of the subscripts at the ends of the subarray). Then*

(a) *n is the value of the expression* `last - first + 1`
(b) *the length of the subarray* `a[first..mid]` *is* $\left\lceil \frac{n}{2} \right\rceil$;
(c) *the length of the subarray* `a[first..mid-1]` *is* $\left\lceil \frac{n}{2} \right\rceil - 1 = \left\lfloor \frac{n-1}{2} \right\rfloor$;
(d) *the length of the subarray* `a[mid+1..last]` *is* $\left\lfloor \frac{n}{2} \right\rfloor$.

See Fig. 2.1 for a graphical representation.

Proof Part (a) is proved by considering the full array that starts at location 0 and ends at location `last`. There are `last + 1` cells in that subarray. If we then remove the cells in the subarray `a[0..first-1]` we produce the subarray `a[first..last]`. We are removing `(first-1)+1 = first` cells from `last + 1` cells, leaving `(last+1)-first = last-first+1` cells. See Fig. 2.2.

For the proof of statement (b), we begin by noting that the subarray `a[first..mid]` contains `mid-first+1` cells, and `mid` $= \lfloor$`(first+last)/2` $\rfloor$. Then we can write

$$\texttt{mid} - \texttt{first} + 1 = \left\lfloor \frac{\texttt{first} + \texttt{last}}{2} \right\rfloor - \texttt{first} + 1$$

$$= \left\lfloor \frac{\texttt{first} + \texttt{last}}{2} - \texttt{first} + 1 \right\rfloor \quad \text{(by Theorem 2.1.5 (a))}$$

$$= \left\lfloor \frac{\texttt{first} + \texttt{last} - 2\,\texttt{first} + 2}{2} \right\rfloor = \left\lfloor \frac{\texttt{last} - \texttt{first} + 2}{2} \right\rfloor$$
$$= \left\lfloor \frac{(\texttt{last} - \texttt{first} + 1) + 1}{2} \right\rfloor$$
$$= \left\lfloor \frac{n+1}{2} \right\rfloor = \left\lceil \frac{n}{2} \right\rceil.$$

To prove part (c), note that the subarray `a[first..mid-1]` has one less cell than `a[first..mid]`. By part (b), the length of `a[first..mid]` is $\lceil n/2 \rceil$, so the length of `a[first..mid-1]` is

$$\left\lceil \frac{n}{2} \right\rceil - 1 = \left\lfloor \frac{n+1}{2} \right\rfloor - 1 = \left\lfloor \frac{n-1}{2} \right\rfloor$$

by Theorem 2.1.5 (c).

To prove part (d), subtract the number of cells in `a[first..mid]` from n to obtain the number of cells in `a[mid+1..last]`. We get $n - \left\lceil \frac{n}{2} \right\rceil = \left\lfloor \frac{n}{2} \right\rfloor$ by Theorem 2.1.5 (b). ■

2.1.1 Further Examples

Example 2.1.7 (a) $\left\lfloor \frac{26}{7} \right\rfloor =?$ (b) $\left\lceil \frac{26}{7} \right\rceil =?$ (c) $\left\lfloor \frac{7}{26} \right\rfloor =?$ (d) $\left\lceil \frac{7}{26} \right\rceil =?$

Solution (a) $\left\lfloor \frac{26}{7} \right\rfloor = \lfloor 3 + 5/7 \rfloor = 3$ (b) 4 (c) 0 (d) 1

Example 2.1.8 Use part (a) of Theorem 2.1.4 to prove that $0 \le x - \lfloor x \rfloor < 1$ for all real numbers x.

Solution By Theorem 2.1.4 (a) we know that $\lfloor x \rfloor \le x < \lfloor x \rfloor + 1$ for all real x. We can pull this expression apart into the following pair of inequalities: $\lfloor x \rfloor \le x$ and $x < \lfloor x \rfloor + 1$ for all real x. From the first of these it follows that $0 \le x - \lfloor x \rfloor$ (subtract $\lfloor x \rfloor$ from both sides of the inequality); from the second it follows similarly that $x - \lfloor x \rfloor < 1$. Thus we have $0 \le x - \lfloor x \rfloor$ and $x - \lfloor x \rfloor < 1$ for all x. Putting these back together gives the desired expression: $0 \le x - \lfloor x \rfloor < 1$.

Example 2.1.9 Verify part (a) of Theorem 2.1.5 in the case where $x = 12.46$ and $n = -5$.

Solution $\lfloor x+n \rfloor = \lfloor 12.46+(-5) \rfloor = \lfloor 7.46 \rfloor = 7$, and $\lfloor x \rfloor + n = \lfloor 12.46 \rfloor + (-5) = 12 - 5 = 7$. Thus we have verified in this one particular case that $\lfloor x+n \rfloor = \lfloor x \rfloor + n$. Similarly, $\lceil x + n \rceil = \lceil 12.46 + (-5) \rceil = \lceil 7.46 \rceil = 8$ and $\lceil 12.46 \rceil + (-5) = 13 - 5 = 8$.

Example 2.1.10 Verify part (c) of Theorem 2.1.5 for some odd integer n and also for some even integer n.

Solution For $n = 13$, is it true that $\left\lfloor \frac{13+1}{2} \right\rfloor = \left\lceil \frac{13}{2} \right\rceil$, i.e., that $\lfloor 7 \rfloor = \lceil 6.5 \rceil$? Yes: both are equal to 7. For $n = 14$, is it true that $\left\lfloor \frac{14+1}{2} \right\rfloor = \left\lceil \frac{14}{2} \right\rceil$, i.e., that $\lfloor 7.5 \rfloor = \lceil 7 \rceil$? Yes: both are equal to 7.

Example 2.1.11 For each of the assertions below, cite one or more parts of Theorems 2.1.4 and 2.1.5 to help you justify the assertion.

(a) For all integers p and q, $\left\lfloor \frac{p+q}{2} \right\rfloor + \left\lceil \frac{p+q}{2} \right\rceil = p + q$.
(b) For all integers m, $\lfloor 2m + \sqrt{m} \rfloor = 2m + \lfloor \sqrt{m} \rfloor$.
(c) For all integers n, p, and q, if $p = \left\lfloor \frac{n}{2} \right\rfloor$ and $q = \left\lfloor \frac{p}{2} \right\rfloor$, then $q = \left\lfloor \frac{n}{4} \right\rfloor$. More generally, if $p_1 = \left\lfloor \frac{n}{2} \right\rfloor$, $p_2 = \left\lfloor \frac{p_1}{2} \right\rfloor$, $p_3 = \left\lfloor \frac{p_2}{2} \right\rfloor$, $\ldots$, $p_n = \left\lfloor \frac{p_{n-1}}{2} \right\rfloor$, then $p_n = \left\lfloor \frac{n}{2^n} \right\rfloor$.
(d) For all integers n, $\left\lfloor \frac{n}{2} \right\rfloor = \left\lceil \frac{n+1}{2} \right\rceil - 1$.

Solution (a) Use part (b), with $n = p + q$.
(b) Use part (a), with $x = \sqrt{m}$ and $n = 2m$.
(c) Use part (d) with $m = 2$ and $k = 2$. That is, $q = \left\lfloor \frac{p}{2} \right\rfloor = \left\lfloor \frac{\lfloor \frac{n}{2} \rfloor}{2} \right\rfloor = \left\lfloor \frac{n}{2 \cdot 2} \right\rfloor = \left\lfloor \frac{n}{4} \right\rfloor$. The second sentence in (c) is proved by induction.
(d) Use part (c), with $m = 2$.

Example 2.1.12 In the C++ loop given below, assume that the variable `n` has some positive integer value. When the loop is executed, how many times will the body of the loop be executed? (In this loop, the body of the loop consists of just the `cout` statement.) Express your answer as a mathematical formula in terms of the variable `n`.

```
int k; for (k = 1; (3*k)+1 <= n; ++k)
  cout << k << endl;
```

Solution First note that the initial value computed for (3*k)+1 is 4, so if the value of `n` is 3 or less, the body of the loop will be executed 0 times. So now let's look at the cases where the value of `n` is at least 4. In these cases the body of the loop will be executed at least once.

Since the variable `k` starts with the value 1 and is incremented by 1 each time the body of the loop is executed, `k` is a loop counter. That is, when its value is 1 in

the `cout` statement, the body of the loop is being executed for the first time. When its value is 2 in the `cout` statement, the body of the loop is being executed for the second time. Etc. Let L denote the value of `k` in the `cout` statement the last time the body of the loop is executed. Then that number L is the total number of times the body of the loop is executed. That is, L is the number we want to compute.

Let `n` denote the value of the program variable `n`. Then $3L + 1 \leq \texttt{n}$ because when `k` has the value L, the loop control condition `(3*k)+1 <= n` must be true. (If that were not so, the body of the loop would not be executed when `k` gets the value L.) Also note that when `k` is then incremented to $L + 1$ (in the expression `k++`), the loop control condition must then be false in order to make the loop halt. Thus $3(L + 1) + 1 > \texttt{n}$. We now have the following two inequalities involving L and `n`:

$$3L + 1 \leq \texttt{n} < 3(L + 1) + 1.$$

Subtracting 1 from each the part of this expression, and then dividing each part by 3 gives the inequalities

$$L \leq \frac{\texttt{n} - 1}{3} < L + 1$$

It follows that the integer L must be the floor of the fraction `(n-1)/3` (see Theorem 2.1.4 (c)). That is, $L = \left\lfloor \dfrac{\texttt{n} - 1}{3} \right\rfloor$. This is the number of times the body of the loop is executed when $\texttt{n} \geq 4$. In fact, this formula gives the correct number of times even in the case where `n` is 1, 2, or 3.

Example 2.1.13 Prove that for all real numbers x and y we have $\lceil x \rceil + \lceil y \rceil \leq \lceil x + y \rceil + 1$.

Solution Take any real numbers x and y. By Theorem 2.1.4 (b), $\lceil x \rceil < x + 1$ and $\lceil y \rceil < y + 1$. It follows that $\lceil x \rceil + \lceil y \rceil < x + y + 2$. By Theorem 2.1.4 (b), $x + y \leq \lceil x + y \rceil$, so $x + y + 2 \leq \lceil x + y \rceil + 2$. Putting together the inequalities in the two preceding sentences we get $\lceil x \rceil + \lceil y \rceil < \lceil x + y \rceil + 2$. Note that the left and right sides of this inequality are integers. If $i < j$ for integers i and j, then we know that $i \leq j - 1$, so $\lceil x \rceil + \lceil y \rceil \leq \lceil x + y \rceil + 1$.

2.1.2 Exercises

2.1.14 Let `a` be the name of an array of length `n > 0` in a C or C++ program. Explain why the expression `a[n/2]` will be accepted by the compiler (even though `n` may be an odd integer) but `a[floor(n/2.0)]` will not compile.

2.1.15 Use part (b) of Theorem 2.1.4 to prove that $0 \leq \lceil x \rceil - x < 1$ for all real numbers x.

2.1.16 Is it always true that $\lceil x \rceil = 1 + \lfloor x \rfloor$? If so, prove this fact. If not, give a counterexample, i.e., an example of a real number for which the equation is not true.

2.1.17 Theorem 2.1.4 (d) says that if $x \leq y$ then $\lfloor x \rfloor \leq \lfloor y \rfloor$ for all real numbers x and y. Can we conclude that if $x < y$ then $\lfloor x \rfloor < \lfloor y \rfloor$?

2.1.18 (a) Give an example of an integer n and a real number x for which $\lfloor nx \rfloor \neq n \lfloor x \rfloor$. (See part (a) of Theorem 2.1.5.)
(b) Give an example of two real numbers x and y for which $\lfloor x + y \rfloor \neq \lfloor x \rfloor + \lfloor y \rfloor$. (See part (a) of Theorem 2.1.5.)

2.1.19 Verify part (b) of Theorem 2.1.5 for the odd integer $n = 11$ and also for the even integer $n = 12$.

2.1.20 (a) Verify part (d) of Theorem 2.1.5 in the case where $n = 37$, $m = 8$, and $k = 3$.
(b) Verify part (d) of Theorem 2.1.5 in the case where $n = 26$, $m = 7$, and $k = 2$.

2.1.21 Is it true that $\lfloor x \rfloor - n = \lfloor x - n \rfloor$ for all integers n and real numbers x? Answer the same question for the equation $n - \lfloor x \rfloor = \lfloor n - x \rfloor$.

2.1.22 For each of the assertions below, cite one or more parts of Theorems 2.1.4 and 2.1.5 to help you justify the assertion.

(a) For all integers n, $\left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{n+1}{2} \right\rfloor = n$.
(b) For all non-negative real numbers x, $\lfloor \sqrt{x} - 1 \rfloor = \lfloor \sqrt{x} \rfloor - 1$.
(c) For all integers n, $\left\lceil \frac{n+1}{2} \right\rceil + \left\lceil \frac{n}{2} \right\rceil = n + 1$.
(d) For all real numbers x and non-negative integers p, $\lfloor x^2 + 2^p \rfloor = \lfloor x^2 \rfloor + 2^p$.
(e) For all real numbers x, $\lfloor x + \lfloor x \rfloor \rfloor = 2 \lfloor x \rfloor$.

2.1.23 In the C++ loop given below, assume that the integer variable `n` has some positive value. How many times is the *body* of the loop executed? Express your answer as a mathematical expression in terms of the value of `n`. Justify your answer by an argument similar to the one in Example 2.1.12 on p. 17. Also check your answer carefully by determining what integers are sent to output when `n` = 50.

```
int k;
for (k = 1; k*k <= n; ++k)
  cout << k << endl;
```

2.1.24 In the C++ loop given below, assume that the integer variable `n` has some positive value. How many times is the *body* of the loop executed? Express your answer as a mathematical expression in terms of the value of `n`. Justify your answer by an argument similar to the one in Example 2.1.12 on p. 17. Also check your answer carefully by determining what integers are sent to output when `n` = 10 and also when `n` = 11.

```
int k;
for (k = 1; 2*k < n; ++k)
  cout << k << endl;
```

2.1.25 (a) Prove that for all real numbers x and y we have $\lfloor x \rfloor + \lfloor y \rfloor \leq \lfloor x + y \rfloor \leq \lfloor x \rfloor + \lfloor y \rfloor + 1$.
(b) Prove that for all real numbers x and y we have $\lceil x \rceil + \lceil y \rceil - 1 \leq \lceil x + y \rceil \leq \lceil x \rceil + \lceil y \rceil$.
(c) Prove that for all non-negative real numbers x and y, $\lfloor x \rfloor \lfloor y \rfloor \leq \lfloor xy \rfloor \leq \lfloor x \rfloor \lfloor y \rfloor + (\lfloor x \rfloor + \lfloor y \rfloor)$.

2.1.26 Let n and m be two positive integer numbers. Prove that $\dfrac{n-m+1}{m} \leq \left\lfloor \dfrac{n}{m} \right\rfloor \leq \dfrac{n}{m}$ and $\dfrac{n}{m} \leq \left\lceil \dfrac{n}{m} \right\rceil \leq \dfrac{n+m-1}{m}$.

2.1.27 Let n be an integer number. If n is odd, then $\left\lfloor \dfrac{n}{2} \right\rfloor = \dfrac{n-1}{2}$ and $\left\lceil \dfrac{n}{2} \right\rceil = \dfrac{n+1}{2}$.

2.2 Logarithms

Definition 2.2.1 For a fixed real number $b > 1$, the *logarithm base b function* is defined on positive real numbers x as follows: $\log_b(x) =$ "the power on b that produces x". In other words, $\log_b(x) = y$ if and only if $b^y = x$.

Examples:
$\log_3(9) = 2$ because $3^2 = 9$.
$\log_5(125) = 3$ because $5^3 = 125$.
$\log_4(4) = 1$ because $4^1 = 4$.
$\log_6(1) = 0$ because $6^0 = 1$.
Two important equations follow instantly from this definition.

Theorem 2.2.2 *For any fixed real number $b > 1$,*

(a) $b^{\log_b(x)} = x$ *for all real* $x > 0$;
(b) $\log_b(b^y) = y$ *for all real* y.

Logarithms were invented nearly 400 years ago because they have several important properties that could be used to simplify certain difficult arithmetic calculations: they convert products to sums, they convert quotients to differences, and they convert powers to products. Here is a precise statement of these properties.

Theorem 2.2.3 *For any fixed real number $b > 1$,*

(a) $\log_b(s \cdot t) = \log_b(s) + \log_b(t)$ *for all positive real numbers* s *and* t*;*
(b) $\log_b(s/t) = \log_b(s) - \log_b(t)$ *for all positive real numbers* s *and* t*;*
(c) $\log_b(t^p) = p\ \log_b(t)$ *for all real numbers* p *and all positive real numbers* t.

Proof The proofs are given in algebra or pre-calculus textbooks and will not be repeated here. ■

Example 2.2.4 Most of us met logarithms base 10 in our high school algebra class.
$\log_{10}(1000) = 3$; $\log_{10}(100) = 2$; $\log_{10}(10) = 1$; $\log_{10}(1) = 0$ because $10^0 = 1$;
$\log_{10}(0.01) = -2$ because $0.01 = 10^{-2}$;
$\log_{10}(-100)$ is undefined because $10^x > 0$ for all real x;
$1 < \log_{10}(25) < 2$ because $10^1 < 25 < 10^2$; by calculator, $\log_{10}(25) \approx 1.4$;
$\log_{10}(1/t) = -\log_{10}(t)$.

Example 2.2.5 Here is the logarithm you met in calculus; it's called the *natural logarithm*:

$$\log_{\mathrm{e}}(x) = \ln(x) = \int_1^x \frac{1}{t}\,\mathrm{d}t \quad \text{for all real } x > 0, \quad \text{where } \mathrm{e} \approx 2.71828.$$

Of course, $\mathrm{e}^{\ln(x)} = x$ if $x > 0$, and $\ln(\mathrm{e}^y) = y$ for all real numbers y.

In computer science, the logarithm that we encounter more often than any other is the *logarithm base 2 function*, denoted by "$\lg(x)$" or $\log_2(x)$. Here are some easily computed values of this function:

$\lg(32) = 5$; $\lg(8) = 3$; $\lg(1024) = 10$; $\lg(1) = 0$; $\lg(2) = 1$; $\lg(1/16) = -4$.

What is the numerical value of $\lg(100)$? We can see that it must be between 6 and 7 because $2^6 < 100 < 2^7$. If we want a more exact answer we will need to consult a table of logarithms or use an electronic calculator. Most pocket calculators, however, don't have a "lg" button. This is not a problem because, as we will now prove, *all logarithm functions are just constant multiples of each other*.

Theorem 2.2.6 *For all real numbers* a *and* b *greater than 1 and all positive real numbers* x*,*

$$\log_b(x) = \frac{\log_a(x)}{\log_a(b)}.$$

Proof Take the logarithm, base a, of both sides of the identity $b^{\log_b(x)} = x$. ■

Example 2.2.7 $\ln(x) = \dfrac{\log_{10}(x)}{\log_{10}(e)} \approx 2.303 \log_{10}(x)$ by using a calculator.

Similarly, $\lg(x) = \dfrac{\ln(x)}{\ln(2)} \approx \dfrac{\ln(x)}{0.693} \approx 1.44 \ln(x)$; also, $\lg(x) = \dfrac{\log_{10}(x)}{\log_{10}(2)} \approx 3.32 \log_{10}(x)$.

In analysis of algorithms, we seldom have occasion to compute $\lg(x)$ when x is a non-integer real number. Mostly we will encounter $\lg(n)$ where n is a positive integer. Moreover, the "lg" usually appears in connection with the floor or ceiling function. The following theorem gives a number of useful identities involving the lg function combined with floors and ceilings.

Theorem 2.2.8 *Let n be a positive integer.*

(a) $2^{\lfloor \lg(n) \rfloor} \leq n < 2^{\lfloor \lg(n) \rfloor + 1}$; *equivalently,* $\dfrac{n}{2} < 2^{\lfloor \lg(n) \rfloor} \leq n$.

(b) $\left\lfloor \dfrac{n}{2^{\lfloor \lg(n) \rfloor}} \right\rfloor = 1$ *and* $\left\lfloor \dfrac{n}{2^{\lfloor \lg(n) \rfloor + 1}} \right\rfloor = 0$. *Moreover, the only integer k for which* $\left\lfloor \dfrac{n}{2^k} \right\rfloor = 1$ *is* $k = \lfloor \lg(n) \rfloor$.
We can interpret this as saying that $\lfloor \lg(n) \rfloor$ is the number of "integer divisions by 2" that are required to reduce n down to 1.

(c) $\lceil \lg(n+1) \rceil = 1 + \lfloor \lg(n) \rfloor$ *for all* $n \geq 1$.

(d) $\lfloor \lg(\lfloor n/2 \rfloor) \rfloor = -1 + \lfloor \lg(n) \rfloor$ *for all* $n \geq 2$.

(e) $\lceil \lg(\lceil n/2 \rceil) \rceil = -1 + \lceil \lg(n) \rceil$ *for all* $n \geq 2$.

(f) $\lfloor \lg(\lceil n/2 \rceil) \rfloor = -1 + \lfloor \lg(n+1) \rfloor$ *for all* $n \geq 1$.

(g) $\lceil \lg(\lfloor n/2 \rfloor) \rceil = -1 + \lceil \lg(n-1) \rceil$ *for all* $n \geq 3$.

Proof To prove part (a), we can start from the fact that

$$\lfloor \lg(n) \rfloor \leq \lg(n) < \lfloor \lg(n) \rfloor + 1,$$

which is an instance of Theorem 2.1.4 (a), with $x = \lg(n)$. From this it follows that

$$2^{\lfloor \lg(n) \rfloor} \leq 2^{\lg(n)} < 2^{\lfloor \lg(n) \rfloor + 1}.$$

Since $2^{\lg(n)} = n$, this proves that $2^{\lfloor \lg(n) \rfloor} \leq n < 2^{\lfloor \lg(n) \rfloor + 1}$.

To prove the first of the two equations in part (b), divide the first double inequality in part (a) through by $2^{\lfloor \lg(n) \rfloor}$ to obtain

$$1 \leq \frac{n}{2^{\lfloor \lg(n) \rfloor}} < 2.$$

By Theorem 2.1.4 (c), this immediately implies that the integer 1 is the floor of $\dfrac{n}{2^{\lfloor \lg(n) \rfloor}}$; that is $\left\lfloor \dfrac{n}{2^{\lfloor \lg(n) \rfloor}} \right\rfloor = 1$.

To prove the second part we start from the double inequality above and divide it through by 2 to obtain $\dfrac{1}{2} \leq \dfrac{n}{2^{\lfloor \lg(n) \rfloor + 1}} < 1$. Since $0 < \dfrac{1}{2}$, we can deduce that $0 < \dfrac{n}{2^{\lfloor \lg(n) \rfloor + 1}} < 1$, which implies that $\left\lfloor \dfrac{n}{2^{\lfloor \lg(n) \rfloor + 1}} \right\rfloor = 0$.

To prove the last assertion in part (b), let's start with n and divide it repeatedly by 2, discarding remainders. We'll give names to the integers we produce along the way:

$n_0 = n = \left\lfloor \frac{n}{2^0} \right\rfloor$ (here we are using the fact that $\lfloor n \rfloor = n$ and $2^0 = 1$)

$n_1 = \left\lfloor \frac{n_0}{2} \right\rfloor = \left\lfloor \frac{n}{2^1} \right\rfloor$

$n_2 = \left\lfloor \frac{n_1}{2} \right\rfloor = \left\lfloor \frac{n}{2^2} \right\rfloor$ (here we are using Theorem 2.1.5 (d); note we've divided by 2 twice)

$n_3 = \left\lfloor \frac{n_2}{2} \right\rfloor = \left\lfloor \frac{n}{2^3} \right\rfloor$ (again we're using Theorem 2.1.5 (d); note we've divided by 2 three times) etc. For any positive integer k, after k divisions by 2 we obtain

$n_k = \left\lfloor \frac{n_{k-1}}{2} \right\rfloor = \left\lfloor \frac{n}{2^k} \right\rfloor$.

In particular, when $k = \lfloor \lg(n) \rfloor$, we see by the first equation in part (b) that we will arrive at the value 1. That is, by dividing n by 2 (discarding remainders) $\lfloor \lg(n) \rfloor$ times, we reduce n to 1.

To prove the equation in part (c), it suffices by Theorem 2.1.4 (c) to prove that the integer $1 + \lfloor \lg(n) \rfloor$ satisfies the inequalities

$$(1 + \lfloor \lg(n) \rfloor) - 1 < \lg(n+1) \leq 1 + \lfloor \lg(n) \rfloor,$$

which are equivalent to

$$\lfloor \lg(n) \rfloor < \lg(n+1) \quad \text{and} \quad \lg(n+1) \leq 1 + \lfloor \lg(n) \rfloor.$$

The first one is easy because $\lfloor \lg(n) \rfloor \leq \lg(n) < \lg(n+1)$. For the second, start with the inequality $n < 2^{\lfloor \lg(n) \rfloor + 1}$ that we proved in part (a). Both sides of the inequality are integers, so it must be true that $n + 1 \leq 2^{\lfloor \lg(n) \rfloor + 1}$. Taking logarithms on both sides gives the desired inequality above.

To prove part (d) we begin with the first inequalities in part (a) and divide by 2 to obtain

$$2^{\lfloor \lg(n) \rfloor - 1} \leq n/2 < 2^{\lfloor \lg(n) \rfloor} \quad \text{for all positive integers } n.$$

If $n \geq 2$, then $2^{\lfloor \lg(n) \rfloor - 1}$ and $2^{\lfloor \lg(n) \rfloor}$ are both integers, and thus

$$2^{\lfloor \lg(n) \rfloor - 1} \leq \lfloor n/2 \rfloor < 2^{\lfloor \lg(n) \rfloor} \quad \text{for all integers } n \geq 2.$$

Taking the logarithm of every term gives

$$\lfloor \lg(n) \rfloor - 1 \leq \lg(\lfloor n/2 \rfloor) < \lfloor \lg(n) \rfloor,$$

which is exactly what is needed to show that $\lfloor \lg(n) \rfloor - 1 = \lfloor \lg(\lfloor n/2 \rfloor) \rfloor$ (see Theorem 2.1.4 (c)).

To prove part (e), note that

$$\begin{aligned} \lceil \lg(\lceil n/2 \rceil) \rceil &= \left\lceil \lg\left(\left\lfloor \frac{n-1}{2} \right\rfloor + 1 \right) \right\rceil \quad \text{(by Theorem 2.1.5 (c))} \\ &= 1 + \left\lfloor \lg\left(\left\lfloor \frac{n-1}{2} \right\rfloor \right) \right\rfloor \quad \text{(by part (c) above)} \\ &= 1 + (-1 + \lfloor \lg(n-1) \rfloor) \quad \text{(by part (d) above)} \end{aligned}$$

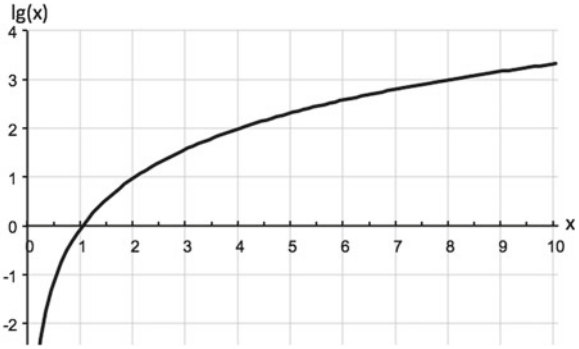

Fig. 2.3 Plot for the function $y = \lg(x)$

$$= \lfloor \lg(n-1) \rfloor = -1 + \lceil \lg((n-1)+1) \rceil \quad \text{(by part (c) above, with } n \text{ replaced by } n-1)$$
$$= -1 + \lceil \lg(n) \rceil.$$

The proofs of parts (f) and (g) are similar, and left as an exercise. ■

Example 2.2.9 To illustrate part (b) of Theorem 2.2.8, take $n = 100$. Then

$$\lfloor 100/2 \rfloor = 50; \quad \lfloor 50/2 \rfloor = 25; \quad \lfloor 25/2 \rfloor = 12; \quad \lfloor 12/2 \rfloor = 6; \quad \lfloor 6/2 \rfloor = 3; \quad \lfloor 3/2 \rfloor = 1.$$

There are 6 integer divisions by 2 in that sequence, and indeed, $\lfloor \lg(100) \rfloor = 6$ because $2^6 < 100 < 2^7$, so $6 < \lg(100) < 7$.

Since we encounter many "divide and conquer" algorithms that involve continually splitting a list of length n into two roughly equal pieces, Theorem 2.2.8 (b) gives us some insight into why these algorithms often have running times that involve $\lg(n)$ in one way or another.

The most important thing to understand about $\lg(n)$ is how very slowly it grows as n increases. Even when n gets extremely large, $\lg(n)$ tends to remain relatively small. The reason is this: doubling the argument in $\lg(n)$ just adds 1 to the value of the logarithm $\lg(2n) = 1 + \lg(n)$. This is illustrated in the graph of $y = \lg(x)$ shown in Fig. 2.3.

You might wonder whether there is some horizontal asymptote for this graph, i.e., a horizontal line that the graph approaches as $n \to \infty$. In fact, there is *no such asymptote*. If we draw a horizontal line at height h, the graph of $y = \lg(x)$ will intersect that line when x reaches the value 2^h, and will continue to rise slowly above the line thereafter. In mathematical notation,

$$\lim_{x \to \infty} \lg(x) = +\infty \tag{2.1}$$

Later in this book we will analyze methods for sorting data into key order. Some sorting algorithms require an amount of time proportional to n^2 in order to sort n elements, while certain other algorithms require an amount of time proportional to

$n \lg(n)$ in order to sort n elements. Which of these algorithms would require the least time when sorting n elements? The answer is that the algorithm requiring time proportional to $n \lg(n)$ would require less time when n is large. The reason is this: for any fixed positive constants of proportionality A and B,

$$A\, n\, \lg(n) \ll B\, n\, n = B\, n^2$$

for all large values of n (the symbol $\ll$ is read "is much smaller than").

The following theorem states a curious identity that will prove to be useful.

Theorem 2.2.10 *For all positive real numbers x and y and $b > 1$ we have $x^{\log_b(y)} = y^{\log_b(x)}$.*

Proof Since the logarithm function is inductive (one-to-one) on its domain (the set of all positive real numbers), we can prove that two positive numbers are equal by proving that their logarithms (in any base) are equal. Thus the formula in the theorem is true if and only if $\log_b(x^{\log_b(y)}) = \log_b(y^{\log_b(x)})$ which is equivalent to $\log_b(y) \cdot \log_b(x) = \log_b(x) \cdot \log_b(y)$. The latter is surely true for any positive x and y. ■

2.2.1 Further Examples

Example 2.2.11 For all positive integers n, $\quad 3^{\lg(n)} = n^{\lg(3)} \approx n^{1.58} \approx n\sqrt{n}$.

Example 2.2.12 Do the following multiple choice exercises *without using a calculator*.

(a) $\ln(e^{7x}) =$ Choose one: (i) $7\ln(x)$ (ii) $7x$ (iii) $7e^x$ (iv) $\ln(7) + \ln(x)$
(b) $\ln(x^7) =$ Choose one: (i) $7\ln(x)$ (ii)$7x$ (iii)$7e^x$ (iv) $\ln(7) + \ln(x)$
(c) $\ln(7x) =$ Choose one: (i) $7\ln(x)$ (ii) $7x$ (iii) $7e^x$ (iv) $\ln(7) + \ln(x)$
(d) $e^{\ln(7x)} =$ Choose one: (i) $7\ln(x)$ (ii) 7x (iii) $7e^x$ (iv) $7 + x$
(e) $e^{x+\ln(7)} =$ Choose one: (i) $7\ln(x)$ (ii) $7x$ (iii) $7e^x$ (iv) $7 + x$
(f) $\lg(2^{-n}) =$ Choose one: (i) $-n$ (ii) $1/n$ (iii) $1 - n$ (iv) $(1/2)\lg(n)$
(g) $\lg(8n) =$ Choose one: (i) $3n$ (ii) $8n$ (iii) $3 + n$ (iv) $3 + \lg(n)$
(h) $\lg(8 + n) =$ Choose one: (i) $3n$ (ii) $8n$ (iii) $3 + n$ (iv) $3 + \lg(n)$
(i) $\lg(n/32) =$ Choose one: (i) $n/5$ (ii) n^5 (iii) $5 - n$ (iv) $\lg(n) - 5$
(j) $2^{\lg(18)} =$ Choose one: (i) 2^9 (ii) 18 (iii) 11 (iv) 36
(k) $2^{3\lg(3)} =$ Choose one: (i) 27 (ii) 64 (iii) 18 (iv) 81
(l) $2^{-\lg(n^2)} =$ Choose one: (i) n^{-2} (ii) $n^{-1/2}$ (iii) 2^{-n} (iv) $2n^2$
(m) $\lg(2^{18}) =$ Choose one: (i) 18 (ii) 36 (iii) $2\lg(18)$ (iv) 18^2
(n) $\lg(\sqrt{2}) =$ Choose one: (i) 1 (ii) 0.5 (iii) $\sqrt{\lg(2)}$ (iv) $2^{0.5}$
(o) $\ln(e^{x^2}) =$ Choose one: (i) $2\ln(x)$ (ii) $e^{2\ln(x)}$ (iii) x^2 (iv) $2x$
(p) $e^{\ln(n+1)} =$ Choose one: (i) $n + e$ (ii) en (iii) n (iv) $n + 1$
(q) $e^{n\ln(x)} =$ Choose one: (i) x^n (ii) n^x (iii) nx (iv) $n^{\ln(x)}$

(r) $e^{\ln(n)-\ln(m)} =$ Choose one: (i) $n-m$ (ii) n/m (iii) $\ln(n-m)$ (iv) $e^{\ln(n-m)}$
(s) $2^{4\lg(3)} =$ Choose one: (i) 12 (ii) 24 (iii) 48 (iv) 81
(t) $\lg(32^{20}) =$ Choose one: (i) 25 (ii) 52 (iii) $5\lg(20)$ (iv) 100

Solution (a) (ii) because $\ln(e^r) = r$ for any real number r
(b) (i) because $\ln(x^p) = p\ln(x)$ for any real number p
(c) (iv) because $\ln(st) = \ln(s) + \ln(t)$ (d) (ii) because $e^{\ln(t)} = t$
(e) (iii) because $e^{a+b} = e^a \cdot e^b$ by the properties of exponents
(f) (i) because $\lg(2^p) = p$
(g) (iv) because $\lg(8n) = \lg(8) + \lg(n) = 3 + \lg(n)$
(h) There is no correct answer. It was a trick question.
(i) (iv) because $\lg(n/32) = \lg(n)g(n) - 5$
(j) (ii) because $2^{\lg(x)} = x$
(k) (i) because $2^{3\lg(3)} = 2^{\lg(3^3)} = 2^{\lg(27)} = 27$
(l) (i) because $2^{-\lg(n^2)} = \dfrac{1}{2^{\lg(n^2)}} = \dfrac{1}{n^2} = n^{-2}$
(m) (i) (n) (ii) because $\sqrt{2} = 2^{0.5}$ (o) (iii) (p) (iv)
(q) (i) because $n\ln(x) = \ln(x^n)$
(r) (ii) because $\ln(n) - \ln(m) = \ln(n/m)$
(s) (iv) because $4\lg(3) = \lg(3^4) = \lg(81)$
(t) (iv) because $\lg(32^{20}) = 20\lg(32)$

Example 2.2.13 Show how to compute the value of $\lfloor \lg(1500) \rfloor$ without using a calculator.

Solution $2^{10} = 1024 < 1500 < 2048 = 2^{11}$, so $10 < \lg(1500) < 11$, so $\lfloor \lg(1500) \rfloor = 10$.

Example 2.2.14 In the block of C++ code shown below, assume that the variable `n` has a positive integer value. When the code is executed, how many times is the body of the loop executed? Express the answer in terms of the value of `n`.

```
int k = 1;
int k, p;
for (k = 1, p = 2;  p <= n;  p *= 2)  // Double p after body of loop is executed.
{
  cout << k << endl;
  ++k;
}
```

Solution First consider the special case where `n` has the value 1. In that case, the loop control condition `n <= n` is immediately false, so the body of the loop is executed 0 times. Now let's look at the cases where the value of `n` is at least 2, so that the body of the loop will be executed at least once. As in Example 2.1.12 on p. 17, the variable `k` is a loop counter. Let L denote the value of `k` in the `cout` statement the last time the body of the loop is executed. Then L is the number we want to compute.

When the body of the loop is executed, the initial value 1 of k is sent to output and then k is incremented to 2. Next, the value of p is doubled to 4 (= 2^2) and then the loop control condition p <= n is tested. If it is found to be true, then the value 2 of k will be sent to output, k will be incremented to 3, and p will be doubled to 8 (= 2^3). Again the loop control condition will be tested. When k gets the value L (which we want to compute), the variable p will then get the value 2^L, and the loop control condition will be true for the last time. The number L will be sent to output, the variable k will be incremented to $L+1$, and p will be doubled to 2^{L+1}. The loop condition will now be found to be false. If we let n denote the (fixed) value of the program variable n, then the following inequalities must be true:

$$2^L \leq n < 2^{L+1}.$$

Taking the logarithm base 2 of each part gives us

$$L \leq \lg(n) < L + 1.$$

As we know from Theorem 2.1.4 (c), this means that $L = \lfloor \lg(n) \rfloor$. This is the number of times the body of the loop is executed. Note that this formula is valid even when n has the value 1, for in that case $\lg(n) = 0$.

In the solution of the preceding example, the key observation was that there exists a simple mathematical relationship between the values k and p of the program variables k and p each time the loop control condition is tested. That relationship is $p = 2^k$, and it is true even when the loop control is tested for the last time and found to be false (at that point $k = L + 1$ and $p = 2^{L+1}$). Any condition that's true for the program variables every time a loop control condition is tested is called a *loop invariant* for the loop. In the block of code in Example A.2.14, the loop invariant $p = 2^k$ is easy to spot, and seeing it was the key to solving the given problem. There are other loop invariants for the block of code in Example 2.2.14. Examples are $p > 1$ and $k < p$, but these invariants are not useful in solving the problem that we were given.

2.2.2 Exercises

2.2.15 Do the following multiple choice exercises *without* using an electronic calculator.

(a) $2^{5+\lg(t)} =$ Choose one: (i) $32 + t$ (ii) $32t$ (iii) $5 + t$ (iv) $5t$
(b) $\lg(2^{5n}) =$ Choose one: (i) $5 + n$ (ii) $32 + n$ (iii) $5n$ (iv) $32n$
(c) $\lg(32p) =$ Choose one: (i) $5\lg(p)$ (ii) $32\lg(p)$ (iii) $5p$ (iv) $5 + \lg(p)$
(d) $\lg(n^8) =$ Choose one: (i) $8\lg(n)$ (ii) $3\lg(n)$ (iii) $3n$ (iv) $3 + \lg(n)$
(e) $2^{3\lg(k)} =$ Choose one: (i) $8\lg(k)$ (ii) $8k$ (iii) k^3 (iv) $3k$
(f) $\lg(8^{-k}) =$ Choose one: (i) $8/k$ (ii) $-3k$ (iii) $1/(3k)$ (iv) $3\lg(k)$

(g) $\lfloor \lg(32x) \rfloor =$ Choose one: (i) $5 + \lfloor \lg(x) \rfloor$ (ii) $5x$ (iii) $32 + x$ (iv) $5\lfloor \lg(x) \rfloor$
(h) $\lg(a\ b\ c) =$ Choose one: (i) $a + b + c$ (ii) $2^{a\ b\ c}$ (iii) 2^{a+b+c} (iv) $\lg(a) + \lg(b) + \lg(c)$
(i) $\lg(8/n^2) =$ Choose one: (i) $6 - \lg(n)$ (ii) $3/\lg(n^2)$ (iii) $3/n^2$ (iv) $3 - 2\lg(n)$
(j) $2^{\lg(5)} =$ Choose one: (i) 5 (ii) 32 (iii) 25 (iv) $\lg(25)$
(k) $2^{3\lg(5)} =$ Choose one: (i) 13 (ii) 15 (iii) 40 (iv) 125
(l) $2^{-\lg(8n)} =$ Choose one: (i) $-8n$ (ii) $1/(8n)$ (iii) $8/n$ (iv) $64n^2$
(m) $\lg(2^8) =$ Choose one: (i) 256 (ii) 3 (iii) 8 (iv) 6
(n) $\lg(\sqrt{38}) =$ Choose one: (i) 19 (ii) $\frac{1}{2}\lg(38)$ (iii) $\sqrt{\lg(38)}$ (iv) 2^{38}
(t) $\lg(8^5) =$ Choose one: (i) 15 (ii) 40 (iii) $8\lg(5)$ (iv) $3 + \lg(5)$

2.2.16 For each of the following, give a single numerical value without using a calculator.
(a) $\lg(36) - \lg(9)$ (b) $\lg(56) - \lg(7)$ (c) $\lg(\sqrt{128})$ (d) $\lg(\sqrt[3]{32})$

2.2.17 (a) Show how to compute the value of $\lfloor \lg(1500) \rfloor$ without using a calculator.
(b) Show how to compute the value of $\lceil \lg(1900) \rceil$ without using a calculator.

2.2.18 (a) Express $\log_3(x)$ as some constant C times $\log_{10}(x)$. Use a pocket calculator to determine the numerical value of the constant C.
(b) Express $\log_5(x)$ as some constant C times $\ln(x)$. Use a pocket calculator to determine the numerical value of the constant C.

2.2.19 (a) Use Theorem 2.2.10 to express $2^{\ln(n)}$ as a power of n. Use a calculator to find the numerical value of the power.
(b) Use Theorem 2.2.10 to express $5^{\ln(2n)}$ as some constant times a power of n. Use a calculator to find the numerical value of the power.

2.2.20 We noted that the first of the two equations in Theorem 2.2.8 (b) can be interpreted as saying that $\lfloor \lg(n) \rfloor$ is the number of "integer divisions by 2" that are required to reduce n down to 1. Give a similar interpretation for the second of the two equations in Theorem 2.2.8 (b).

2.2.21 Consider the following block of C++ code in which `n` is a positive integer.

```
int k;
for (k = n; k > 1; k = k/2)
  cout << "Hello" << endl;
```

(a) How many times will the `cout` statement be executed if the value of `n` is 1 when the loop begins?
(b) How many times will the `cout` statement be executed if the value of `n` is 2 when the loop begins?

(c) How many times will the `cout` statement be executed if the value of `n` is 3 when the loop begins?
(d) Give an expression involving the variable `n` for the number of times that the code above will send "Hello" to output. Cite one of the theorems in this section to justify your answer.

2.2.22 (a) Use (among other things) Theorem 2.1.4 on floors and ceilings to prove that for all positive integers n, $n \leq 2^{\lceil \lg(n) \rceil} < 2n$. Warning: in general, $2^{\lceil \lg(n) \rceil} \neq \lceil 2^{\lg(n)} \rceil$. *Hint:* look at the proof for Theorem 2.2.8 (a).
(b) Use (among other things) Theorem 2.1.4 on floors and ceilings to prove that for all positive integers n, $\frac{n}{10} < 10^{\lfloor \log_{10}(n) \rfloor} \leq n$. Warning: in general, $10^{\lfloor \log_{10}(n) \rfloor} \neq \lfloor 10^{\log_{10}(n)} \rfloor$. *Hint:* look at the proof for Theorem 2.2.8 (a).

2.2.23 (a) Use Theorem 2.1.5 (c) and part (d) of Theorem 2.2.8 to prove part (f) of Theorem 2.2.8.
(b) Use Theorem 2.1.5 (c) and part (e) of Theorem 2.2.8 to prove part (g) of Theorem 2.2.8.

2.2.24 (a) How many many times is the body of the loop below executed? (Note: the block differs from the code in Example 2.2.14 only in the initial value of `p`.) Justify your answer by using a loop invariant. Assume that `n` has a positive integer value.

```
int k, p;
for (k=1, p = 1; p <= n; p *= 2) // Double p after body is executed.
{
  cout << k << endl;
  ++k;
}
```

(b) How many many times is the body of the loop below executed? Assume that `n` has a positive integer value. You may base your answer on your answer for part (a).

```
int p;
for (p = 1; p <= n; p *= 2) // Double p after body is executed.
  cout << p << endl;
```

(c) How many times is the body of the loop below executed? Assume that `n` has a positive integer value.

```
int p;
for (p = 1;  p < n;  p *= 2)  // Double p each time body of loop is executed.
  cout << p << endl;
```

Hint. Imagine that a loop-counting variable `k` with initial value 1 is introduced into the code. Give a loop invariant involving the values k and p of the program variables `k` and `p`. Let L denote the value of `k` the last time the body of the loop is executed. Use your loop invariant to calculate L. You will need to cite Theorem 2.1.4 (c) at some point.

2.3 Sums and Summation Notation

What is the value of the sum

$$1^2 + 3^2 + 5^2 + 7^2 + \cdots + 19^2?$$

To answer this question, you first have to fill in the missing terms that are indicated by the "3 dots" in the sum. In this case that's not hard to do. The 3 dots must surely stand for

$$9^2 + 11^2 + 13^2 + 15^2 + 17^2.$$

What about the sum

$$2 + 6 + 12 + 20 + 30 + \cdots + 90?$$

Here we may or may not see that the terms can be written as

$$1 \cdot 2 + 2 \cdot 3 + 3 \cdot 4 + 4 \cdot 5 + 5 \cdot 6 + \cdots + 9 \cdot 10,$$

in which case the missing terms are $42 + 56 + 72 = 6 \cdot 7 + 7 \cdot 8 + 8 \cdot 9$. It is useful to have a notation that lets us express sums in a more precise way that eliminates the guesswork associated with the 3 dots notation. The sigma summation notation was invented for this purpose. The two sums given above can be written this way:

$$\sum_{k=1}^{10}(2k-1) \quad \text{and} \quad \sum_{k=1}^{9} k(k+1)$$

More generally, suppose a real function $f(k)$ is defined on integers k in the range $m, \ldots, n$. The expression

$$\sum_{k=m}^{n} f(k)$$

where $\sum$ is the upper case Greek letter "sigma", is a precise and concise way of writing the sum

$$f(m) + f(m+1) + \cdots + f(n).$$

The numerical value represented by these expressions is exactly the value that would be produced in the variable `sum` by executing the following fragment of C code:

```
sum = 0;
for (k = m; k <= n; ++k)
  sum += f(k);
```

In particular, by convention, $\sum_{k=m}^{n} f(k)$ has the value 0 if $n < m$ (look at the loop above).

Examples 2.3.1 (a) $\sum_{k=2}^{5} \frac{k}{k+1} = \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + \frac{5}{6} = \frac{183}{60}$

(b) $\sum_{k=0}^{7} k2^{k-1} = 0\cdot 2^{-1} + 1\cdot 2^0 + 2\cdot 2^1 + 3\cdot 2^2 + 4\cdot 2^3 + 5\cdot 2^4 + 6\cdot 2^5 + 7\cdot 2^6 = 769$

(c) $\sum_{k=4}^{4} \lfloor 15/k \rfloor = \lfloor 15/4 \rfloor = 3$

(d) $\sum_{k=10}^{25} 3 = 3 + 3 + 3 + \cdots + 3 = 48$ (because there are 15 + 1 terms here, not just 15). More generally, $\sum_{k=m}^{n} c = c(n - m + 1)$ for any fixed real number c and integers $m \le n$.

(e) $\sum_{j=1}^{3} \left(\sum_{k=1}^{j} (j-k)^2 \right) = \left(\sum_{k=1}^{1} (1-k)^2 \right) + \left(\sum_{k=1}^{2} (2-k)^2 \right) + \left(\sum_{k=1}^{3} (3-k)^2 \right)$
$= (0^2) + (1^2 + 0^2) + (2^2 + 1^2 + 0^2) = 6$

The variable k in the expression $\sum_{k=m}^{n} f(k)$ is called the *summation index*. It is important to understand that it is a "dummy index" that can be replaced by any other letter that has no pre-assigned meaning. Thus, for example, all four of the following expressions have the same value:

$$\sum_{k=2}^{7} k(k+1) \qquad \sum_{i=2}^{7} i(i+1) \qquad \sum_{t=2}^{7} t(t+1) \qquad \sum_{a=2}^{7} a(a+1).$$

Another important fact is that summation is a "linear operator"; that is,

$$\sum_{k=m}^{n} c\cdot f(k) = c\left(\sum_{k=m}^{n} f(k) \right) \quad \text{and} \quad \sum_{k=m}^{n} [f(k) \pm g(k)] = \sum_{k=m}^{n} f(k) \pm \sum_{k=m}^{n} g(k). \tag{2.2}$$

Occasionally we have to "interchange the order of summation" in a double sum. That is, we have to use the fact that

$$\sum_{k=m}^{n} \left(\sum_{j=p}^{q} f(j,k) \right) = \sum_{j=p}^{q} \left(\sum_{k=m}^{n} f(j,k) \right).$$

This equation simply says that if we sum over each row separately in the following array, and then add up the sums we get, we obtain the same answer as if we sum over each column separately and then add up those sums. We can do this as long as the boundaries p and q for the interior sum are independent of the index variable of the exterior sum k.

$$\begin{array}{llll} f(p,m) & f(p+1,m) & f(p+2,m) & \dots f(q,m) \\ f(p,m+1) & f(p+1,m+1) & f(p+2,m+1) & \dots f(q,m+1) \\ f(p,m+2) & f(p+1,m+2) & f(p+2,m+2) & \dots f(q,m+2) \\ \dots & \dots & \dots & \dots \\ f(p,n) & f(p+1,n) & f(p+2,n) & \dots f(q,n) \end{array}$$

We are now going to look at several types of sums that occur frequently in analysis of algorithms. Our goal will be to find some shortcuts for computing the values of these sums.

Our first example is the simple sum $1 + 2 + 3 + \cdots + n$, where n is a positive integer. (When n is 1 or 2 or 3, then by $1 + 2 + 3 + \cdots + n$ we mean simply 1 or $1 + 2$ or $1 + 2 + 3$ respectively.) We can write this sum in the sigma summation notation this way: $\sum_{k=1}^{n} k$. This is called a *triangular sum* because if we draw a bar graph with bars of heights 1, 2, 3, . . ., n, they form a rough triangle. It turns out that there is a simple formula for the value of the triangular sum. It can be obtained by the following algebraic trick. Let $S(n)$ denote the sum $\sum_{k=1}^{n} k$. Then

$$\begin{array}{ll} & S(n) = 1 \quad +2 \quad +3 \quad +\cdots+(n-1)+n. \\ \text{In reverse order we have} & S(n) = n \quad +(n-1)+(n-2)+\cdots+2+\quad 1. \\ \text{Adding them gives} & 2S(n) = (n+1)+(n+1)+(n+1)+\cdots+(n+1)+(n+1) = n(n+1) \\ \text{so} & S(n) = \dfrac{n(n+1)}{2}. \end{array}$$

Theorem 2.3.2 *For any positive integer n we have*

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}.$$

More generally, for any integers m and n satisfying $m \leq n$ we have

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

Proof We have already proved the first of these summation formulas. The second can be proved in a similar way. ■

These formulas should be memorized. It will help you to note that in both formulas above the first factor on the right side is the *number of terms* on the left side, while

the second factor on the right side is the *average* of the first and last numbers on the left side.

In general, when we are confronted with an expression X involving a sigma-summation symbol or a 3-dots ellipsis we like to have a **closed form** for the expression, by which we mean a mathematical expression Y whose value is the same as that of X, but Y is compact and easily computed with a fixed number of operations including exponentiation and formation of logarithms. (We regard 2^n as a closed form because it can be evaluated in two steps: let $x = n\ln(2)$; then $2^n = \exp(x)$.) The calculation in Theorem 2.3.2 shows that $\dfrac{n(n+1)}{2}$ is a closed form for the expressions $1+2+3+\cdots+n$ and $\displaystyle\sum_{k=1}^{n} k$.

Our next example of a frequently occurring sum looks like this: $1+t+t^2+t^3+\cdots+t^n$, where t is any real number and n is any non-negative integer. We can write the sum in sigma notation this way: $\displaystyle\sum_{k=0}^{n} t^k$. Note that the lower limit of the summation is $k=0$. By convention, in sigma sums of this form, t^0 is understood to stand for 1, *even when* $t=0$. (Normally in mathematics the expression 0^0 is considered to be undefined.) Sums of this form are called *(finite) geometric sums*. We can find a closed form for the sum by a trick similar to the one used for triangular sums. Let $G(t,n)$ denote $\displaystyle\sum_{k=0}^{n} t^k$. Then

$$G(t,n) = 1+t+t^2+t^3+\cdots+t^n.$$

Multiplying by t gives $t\,G(t,n) = t+t^2+t^3+t^4+\cdots+t^{n+1}$.

Subtracting them gives $G(t,n) - t\,G(t,n) = 1-t^{n+1}$ (other terms cancel).

Factoring gives $G(t,n)(1-t) = 1-t^{n+1}$. If $t \neq 1$, then we obtain $G(t,n) = \dfrac{1-t^{n+1}}{1-t}$.

That is, $1+t+t^2+t^3+\cdots+t^n = \dfrac{1-t^{n+1}}{1-t} = \dfrac{t^{n+1}-1}{t-1}$. when $t \neq 1$.

Theorem 2.3.3 *For all real numbers t and non-negative integers n,*

$$\sum_{k=0}^{n} t^k = \begin{cases} \dfrac{1-t^{n+1}}{1-t} & \text{if } t \neq 1; \\ n+1 & \text{if } t = 1. \end{cases}$$

Proof See paragraphs preceding this theorem. ■

Our third example of a frequently occurring sum looks like this: $1+\dfrac{1}{2}+\dfrac{1}{3}+\cdots+\dfrac{1}{n}$, which can be written in sigma summation notation this way: $\displaystyle\sum_{k=1}^{n} \frac{1}{k}$. We call a sum of

this form a *harmonic sum* and denote it by H(n). (In music, if strings of like material, diameter, and torsion have lengths 1, 1/2, 1/3, etc., then when set in vibration they produce harmonic tones.) Sadly, there is no closed form for harmonic sums. Nevertheless, we can approximate their values very closely by using a technique that is applicable both here and in other similar situations. The technique is given by the following theorem.

Theorem 2.3.4 *Let $f(x)$ be a strictly decreasing function defined over an interval $[m, n]$, where m and n are integers such that $m < n$. Then*

$$f(n) + \int_m^n f(x)dx \;<\; \sum_{k=m}^{n} f(k) \;<\; f(m) + \int_m^n f(x)dx.$$

Similarly, if $f(x)$ is strictly increasing on $[m, n]$, then

$$f(m) + \int_m^n f(x)dx \;<\; \sum_{k=m}^{n} f(k) \;<\; f(n) + \int_m^n f(x)dx.$$

Proof The proof is easier if we begin by assuming that $f(x) \geq 0$ for all x in $[m, n]$. At the end of the proof we will see how to take care of the cases where $f(x)$ has some negative values.

For the proof when $f(x) \geq 0$ and decreasing we use a graph of the equation $y = f(x)$ on $[m, n]$, draw some rectangles, and compare their areas with the area of the region under the curve $y = f(x)$ shown in Fig. 2.4.

Note that the area of the left-most rectangle in Fig. 2.4 is $1 \times f(m) = f(m)$, the area of the next rectangle is $1 \times f(m + 1) = f(m + 1)$, and so on to the last one, which has area $1 \times f(n - 1) = f(n - 1)$. Now consider the region R bounded by the x-axis, the curve $y = f(x)$, and the vertical lines at $x = m$ and $x = n$. This region R lies inside the collection of rectangles shown, and thus the sum of areas of

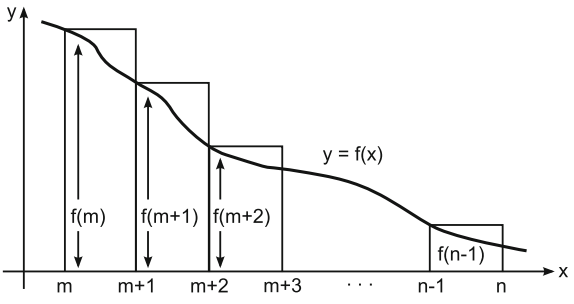

Fig. 2.4 Theorem 2.3.4

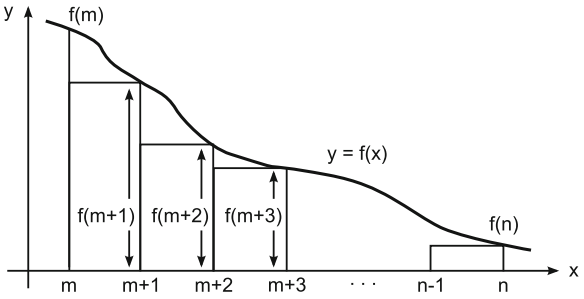

Fig. 2.5 Theorem 2.3.4

these rectangles is greater than the area of R. This can be expressed as follows:

$$\text{area of region } R = \int_m^n f(x)\,\mathrm{d}x \;<\; f(m)+f(m+1)+\cdots+f(n-1) = \sum_{k=m}^{n-1} f(k).$$

Adding $f(n)$ to both sides of the inequality above gives the first of the inequalities about decreasing functions.

For the second of the inequalities about decreasing functions, consider the graph shown in Fig. 2.5. All the rectangles in Fig. 2.5 lie inside the region S bounded by the x-axis, the curve $y = f(x)$, and the vertical lines at $x = m$ and $x = n$, so the sum of the areas of the rectangles is less than the area of S. This can be expressed as follows:

$$f(m+1)+f(m+2)+\cdots+f(n) \;<\; \text{area of region } S = \int_m^n f(x)\,\mathrm{d}x.$$

Adding $f(m)$ to both sides of the inequality above gives the second of the inequalities about decreasing functions.

Now let's take care of the case where some or all of the values of $f(x)$ are negative. Let C denote the value of $f(n)$. Since $f(x)$ is decreasing on the interval $[m, n]$, it follows that $f(x) \geq C$ for all x in that interval. This implies that $f(x) - C \geq 0$ on that interval. Now define a new function $g(x)$ by the formula $g(x) = f(x) - C$. Then $g(x) \geq 0$ everywhere on $[m, n]$. Moreover, $g(x)$ is strictly decreasing on that interval. By the first part of this proof,

$$g(n) + \int_m^n g(x)\mathrm{d}x \;<\; \sum_{k=m}^{n} g(k) \;<\; g(m) + \int_m^n g(x)\mathrm{d}x.$$

Replacing $g(x)$ by $f(x) - C$ above gives us

$$f(n)-C+\int_m^n [f(x)-C]\,\mathrm{d}x \;<\; \sum_{k=m}^{n} [f(k)-C] \;<\; f(m)-C+\int_m^n [f(x)-C]\,\mathrm{d}x.$$

Now we use properties of summation mentioned in Examples 2.3.1 (d) and in formula (2.3) on p. 29, together with standard integration formulas to produce the following inequalities:

$$f(n) - C + \int_m^n f(x)\mathrm{d}x \; - \; C(n-m) \; < \; \sum_{k=m}^{n} f(k) \; - \; C(m-n+1) \; < \; f(m) - C$$

$$+ \int_m^n f(x)\mathrm{d}x \; - \; C(n-m).$$

Adding $C(n-m+1)$ to each part of the inequalities above gives the inequalities for $f(x)$ stated in the theorem.

The proofs for the inequalities involving increasing functions are similar and will be left as an exercise. ■

Theorem 2.3.5 (Approximation for Harmonic Sums). *For all integers $n > 1$ we have*

$$\frac{1}{n} + \ln(n) < \mathrm{H(n)} < 1 + \ln(\mathrm{n}).$$

Proof We want to approximate $\mathrm{H}(n) = \sum_{k=1}^{n} \frac{1}{k}$ for $n > 1$.

To do this we apply Theorem 2.3.4 to the strictly decreasing function $f(x) = \frac{1}{x}$, with $m = 1$ and $n > 1$. We get

$$\frac{1}{n} + \int_1^n \frac{1}{x}\mathrm{d}x < \sum_{k=1}^{n} \frac{1}{k} < \frac{1}{1} + \int_1^n \frac{1}{x}\mathrm{d}x \quad \text{for all } n > 1,$$

which immediately yields the inequalities stated in the theorem. ■

The important thing to remember from Theorem 2.3.5 is that for all large n we have $\mathrm{H}(n) \approx \ln(n)$. In rare cases where we want a more accurate approximation for $\mathrm{H}(n)$ when n is large, we can use the following theorem discovered by Leonhard Euler.

Theorem 2.3.6 (Euler) $\lim_{n\to\infty} [H(n) - \ln(n)] = 0.5772$ *approximately and therefore* $H(n) \approx 0.5772 + \ln(n)$ *for all large n.*

Proof See almost any advanced calculus textbook. The number 0.5772 given in this theorem is an approximation to the actual limit, which is known to be an irrational number. ■

When $n \geq 15$, Euler's approximation is correct to within 1 % of the actual value. Of course, for small values of n, we can calculate $\mathrm{H}(n)$ very accurately with a pocket calculator.

Closed forms for several other types of sums are occasionally needed in analysis of algorithms. They do not, however, occur with sufficient frequency to make them worth memorizing. We give four such sums below.

Theorem 2.3.7 *For all positive integers n and real numbers $t \neq 1$,*

$$1 + 2t + 3t^2 + \cdots + nt^{n-1} = \frac{1 + nt^{n+1} - (n+1)t^n}{(t-1)^2}.$$

Proof The left side is just the derivative with respect to t of the finite geometric sum $1 + t + t^2 + t^3 + \cdots + t^n$. Thus we can obtain the formula above simply by differentiating both sides of the formula given in Theorem 2.3.3 with respect to t. ■

Theorem 2.3.8 *For all positive integers n,*

$$\sum_{k=2}^{n} \lceil \lg(k) \rceil = n \lfloor \lg(n) \rfloor + (n+1) - 2^{\lfloor \lg(n) \rfloor + 1}$$

and

$$\sum_{k=2}^{n} \lfloor \lg(k) \rfloor = (n+1) \lfloor \lg(n+1) \rfloor + 2 - 2^{\lfloor \lg(n+1) \rfloor + 1}.$$

Proof When $k = 2$ we have $\lceil \lg(k) \rceil = 1$. When $k = 3$ or $k = 4$ we have $\lceil \lg(k) \rceil = 2$. When $k = 5$ or 6 or 7 or 8 we have $\lceil \lg(k) \rceil = 3$. Etc. Thus, the terms in $\sum_{k=2}^{n} \lceil \lg(k) \rceil$ can be grouped as follows

$$\overset{k=2}{1} + \underbrace{\overbrace{(2+2)}^{k=3,4}}_{2^1\text{terms}} + \underbrace{\overbrace{(3+3+3+3)}^{k=5,6,7,8=2^3}}_{2^2\text{terms}} + \underbrace{\overbrace{(4+4+\cdots+4)}^{k=9,10,\ldots,16=2^4}}_{2^3\text{terms}} + \cdots + \underbrace{\overbrace{(\lfloor \lg(n) \rfloor \cdots + \lfloor \lg(n) \rfloor)}^{k=2^{\lfloor \lg(n) \rfloor - 1}+1,\ldots,2^{\lfloor \lg(n) \rfloor}}}_{2^{\lfloor \lg(n) \rfloor - 1}\text{terms}} + \textit{more}$$
$$2^0\text{term}$$

The "*more*" (above) gives the remaining terms, corresponding to $k = 2^{\lfloor \lg(n) \rfloor} + 1$ up to $k = n$ (this set of terms is empty if n is a power of 2). Each of these terms is equal to $\lfloor \lg(n) \rfloor + 1$. Thus we have

$$\sum_{k=2}^{n} \lceil \lg(k) \rceil = \sum_{p=1}^{\lfloor \lg(n) \rfloor} 2^{p-1} p + \left(n - 2^{\lfloor \lg(n) \rfloor} \right) (\lfloor \lg(n) \rfloor + 1). \tag{2.3}$$

The sigma sum on the right side is just the sum given in Theorem 2.3.7, but with k replaced by p, n replaced by $\lfloor \lg(n) \rfloor$, and t replaced by 2. Thus we get

$$\sum_{p=1}^{\lfloor \lg(n) \rfloor} 2^{p-1} p = \frac{1 + \lfloor \lg(n) \rfloor 2^{\lfloor \lg(n) \rfloor + 1} - (\lfloor \lg(n) \rfloor + 1) 2^{\lfloor \lg(n) \rfloor}}{(2-1)^2}.$$

Inserting this into Eq. (2.3) gives us

$$\sum_{k=2}^{n} \lceil \lg(k) \rceil = 1 + \lfloor \lg(n) \rfloor 2^{\lfloor \lg n \rfloor + 1} - (\lfloor \lg(n) \rfloor + 1) 2^{\lfloor \lg n \rfloor} + n \lfloor \lg(n) \rfloor + n - \lfloor \lg(n) \rfloor 2^{\lfloor \lg n \rfloor} - 2^{\lfloor \lg n \rfloor},$$

which reduces to the first of the two sums in the statement of the theorem.

To prove the second identity in this theorem, begin by replacing the symbol n by the symbol $n + 1$ everywhere in the first equation stated in the theorem. This gives

$$\sum_{k=2}^{n+1} \lceil \lg(k) \rceil = (n+1) \lfloor \lg(n+1) \rfloor + (n+2) - 2^{\lfloor \lg(n+1) \rfloor + 1} \quad \text{for all positive integers } n.$$

By Theorem 2.2.8 (c), with n replaced by $k - 1$, we get $\lceil \lg(k) \rceil = 1 + \lfloor \lg(k-1) \rfloor$, so the sum above is

$$\sum_{k=2}^{n+1} (1 + \lfloor \lg(k-1) \rfloor) = (n+1) \lfloor \lg(n+1) \rfloor + (n+2) - 2^{\lfloor \lg(n+1) \rfloor + 1}. \quad (2.4)$$

The left side can be expressed as

$$\sum_{k=2}^{n+1} 1 + \sum_{k=2}^{n+1} \lfloor \lg(k-1) \rfloor = n + 0 + \lfloor \lg(2) \rfloor + \lfloor \lg(3) \rfloor + \cdots + \lfloor \lg(n) \rfloor = n + \sum_{k=2}^{n} \lfloor \lg(k) \rfloor.$$

Putting this into Eq. (2.4) gives

$$n + \sum_{k=2}^{n} \lfloor \lg(k) \rfloor = (n+1) \lfloor \lg(n+1) \rfloor + (n+2) - 2^{\lfloor \lg(n+1) \rfloor + 1}.$$

Subtracting n from both sides of this equation gives the desired equation. ■

Example 2.3.9 Many computer science textbooks discuss the representation of positive integers using various bases. For example, the integer 6029_{10} (base 10 notation) can also be written in binary notation (base 2 notation) as 1011110001101_2 or in octal notation (base 8) as 13615_8 or in hexadecimal notation (base 16) as $178D_{16}$. Of course, it is also possible to write this integer in these bases with one or more leading zeroes: 006029_{10}, $0000101111000110 1_2$, 013615_8, $178D_{16}$. Let's agree to use the phrase *standard representation of n base b* to describe the representation of a positive integer n without leading zeroes in base b notation. Note that the number of digits required by the standard representation of n depends on the base used as well as on the size of n. For example, the standard representations of the number discussed in the second sentence of this paragraph require 4, 13, 5, and 4 digits, respectively, in base 10, base 2, base 8, and base 16.

How many digits are there in the standard decimal (i.e., base 10) representation of the integer 34^{295}? How many bits (binary digits) does its standard binary representation contain? More generally, given any positive integers n and b, can we find a formula for the number of digits that will be required for the standard representation of n base b?

Solution Let's solve the general case first. Assume we are given positive integers n and b. Let k denote the (unknown) number of digits in the standard representation of n base b. Denote those digits by $d_0, d_1, d_2, \ldots, d_{k-1}$, where d_0 is the least significant digit and d_{k-1} is the most significant (and is therefore not zero). Each of these digits lies in the range from 0 to $b-1$. (For example, if b is 10, then each digit lies in the range from 0 to 9.) Moreover, d_{k-1} is at least 1. Then

$$n = d_{k-1}\, b^{k-1} + \cdots + d_2\, b^2 + d_1\, b^1 + d_0\, b^0.$$

Since $d_{k-1} \geq 1$ and all other $d_i \geq 0$, it follows that

$$n \geq 1\, b^{k-1} + \cdots + 0 + 0 + 0.$$

That is, $n \geq b^{k-1}$. Similarly, since every $d_i \leq b-1$, it follows that

$$\begin{aligned} n &\leq (b-1)b^{k-1} + \cdots + (b-1)b^2 + (b-1)b^1 + (b-1)b^0 \\ n &\leq (b-1)[b^{k-1} + \cdots + b^2 + b + 1] \\ n &\leq (b-1)\frac{b^k - 1}{b-1} \\ n &\leq b^{k-1}. \end{aligned}$$

From this it follows that $n < b^k$. We have now established the inequalities

$$b^{k-1} \leq n < b^k,$$

where k is the number of digits in the standard representation of n base b. Taking logarithms base b of each of the quantities above gives us these inequalities:

$$k - 1 \leq \log_b(n) < k.$$

Since $k-1$ is an integer, it follows from Theorem 2.1.4 (c) that $k - 1 = \lfloor \log_b(n) \rfloor$, or equivalently, $k = 1 + \lfloor \log_b(n) \rfloor$. By Theorem 2.2.8 (c), $k = \lceil \log_b(n+1) \rceil$. Thus we have proved the following useful fact:

Theorem 2.3.10 *Let n and b be positive integers with $b \geq 2$. The number of digits in the standard representation of n base b is*

$$1 + \lfloor \log_b(n) \rfloor = \lceil \log_b(n+1) \rceil.$$

Proof This formula is derived in Example 2.3.9. ■

Now we can answer the question we asked in Example 2.3.9 about the number of digits in the standard decimal representation of the integer 34^{295}. The answer, obtained with the aid of a calculator, is

$$1 + \lfloor \log_{10}(34^{295}) \rfloor = 1 + \lfloor 295 \ \log_{10}(34) \rfloor = 1 + \lfloor 295 \times 1.5315 \rfloor = 1 + \lfloor 451.79 \rfloor = 452.$$

The number of bits in the standard binary representation of 34^{295} is

$$1 + \lfloor \log_2(34^{295}) \rfloor = 1 + \lfloor 295 \ \log_2(34) \rfloor = 1 + \lfloor 295 \times 5.087 \rfloor = 1 + \lfloor 1500.80 \rfloor = 1501.$$

The sum in the following theorem appears here and there in analysis of algorithms. Unfortunately it has no simple closed form, so it is helpful to have good lower and upper bounds for it.

Theorem 2.3.11 *The sum* $\displaystyle\sum_{k=1}^{\lfloor \lg(n) \rfloor} \left\lfloor \frac{n}{2^k} \right\rfloor$ *is asymptotic to n. In particular, it satisfies the inequalities*

$$n - 2 - \lfloor \lg(n) \rfloor \ < \ \sum_{k=1}^{\lfloor \lg(n) \rfloor} \left\lfloor \frac{n}{2^k} \right\rfloor \ \leq \ n - 1. \tag{2.5}$$

Proof Using the fact that $x - 1 < \lfloor x \rfloor \leq x$ for all real numbers x, we can write that

$$\sum_{k=1}^{\lfloor \lg(n) \rfloor} \left(\frac{n}{2^k} - 1 \right) < \sum_{k=1}^{\lfloor \lg(n) \rfloor} \left\lfloor \frac{n}{2^k} \right\rfloor \leq \sum_{k=1}^{\lfloor \lg(n) \rfloor} \frac{n}{2^k}.$$

These inequalities are equivalent to

$$\sum_{k=1}^{\lfloor \lg(n) \rfloor} \frac{n}{2^k} - \lfloor \lg(n) \rfloor < \sum_{k=1}^{\lfloor \lg(n) \rfloor} \left\lfloor \frac{n}{2^k} \right\rfloor \leq \sum_{k=1}^{\lfloor \lg(n) \rfloor} \frac{n}{2^k}. \tag{2.6}$$

The sum $\displaystyle\sum_{k=1}^{\lfloor \lg(n) \rfloor} \frac{n}{2^k}$ that appears in these inequalities can be computed by factoring out the n that occurs in every term and then using the geometric sum formula. Note that the term for $k = 0$ is missing from the sum and must therefore be subtracted from it:

$$\begin{aligned} \sum_{k=1}^{\lfloor \lg(n) \rfloor} \frac{n}{2^k} &= n \sum_{k=1}^{\lfloor \lg(n) \rfloor} \left(\frac{1}{2} \right)^k = n \left(\frac{1 - \left(\frac{1}{2}\right)^{\lfloor \lg(n) \rfloor + 1}}{1 - \frac{1}{2}} - 1 \right) = n \left(\frac{2^{\lfloor \lg(n) \rfloor + 1} - 1}{2^{\lfloor \lg(n) \rfloor}} - 1 \right) \\ &= n \left(2 - \frac{1}{2^{\lfloor \lg(n) \rfloor}} - 1 \right) \\ &= n - \frac{n}{2^{\lfloor \lg(n) \rfloor}}. \end{aligned}$$

Inserting this last formula into line 2.6 we get

$$n - \frac{n}{2^{\lfloor \lg(n) \rfloor}} - \lfloor \lg(n) \rfloor \ < \ \sum_{k=1}^{\lfloor \lg(n) \rfloor} \left\lfloor \frac{n}{2^k} \right\rfloor \ \leq \ n - \frac{n}{2^{\lfloor \lg(n) \rfloor}}. \tag{2.7}$$

To get from inequation (2.7) to the desired inequation (2.5) we must estimate the term $-\frac{n}{2^{\lfloor \lg(n) \rfloor}}$. Theorem 2.2.8 (a) gives us these useful inequalities:

$$2^{\lfloor \lg(n) \rfloor} \leq n < 2^{\lfloor \lg(n) \rfloor + 1}.$$

Dividing all terms by $2^{\lfloor \lg(n) \rfloor}$ and then multiplying throughout by -1 gives

$$-1 \geq -\frac{n}{2^{\lfloor \lg(n) \rfloor}} > -2.$$

It follows that

$$n - 2 - \lfloor \lg(n) \rfloor < n - \frac{n}{2^{\lfloor \lg(n) \rfloor}} - \lfloor \lg(n) \rfloor \quad \text{and} \quad n - \frac{n}{2^{\lfloor \lg(n) \rfloor}} \leq n - 1.$$

Combining this pair of inequalities with inequality (2.7) gives the desired result (2.5). ■

2.3.1 Further Examples

Example 2.3.12 Evaluate the following sums. In each case, give a single numerical answer.

(a) $\sum_{k=1}^{5} \frac{1}{k(k+1)}$ (b) $\sum_{r=0}^{3} \frac{r+1}{2^r}$ (c) $\sum_{p=1}^{4} \left(\sum_{q=1}^{p} \frac{p}{q} \right)$ (d) $\sum_{k=100}^{50} \frac{\lfloor \lg(k) \rfloor}{2k}$

Solution (a) $\frac{1}{2} + \frac{1}{6} + \frac{1}{12} + \frac{1}{20} + \frac{1}{30} = \frac{5}{6}$ (b) $\frac{1}{1} + \frac{2}{2} + \frac{3}{4} + \frac{4}{8} = \frac{13}{4}$

(c) $\sum_{j=1}^{1} \frac{1}{j} + \sum_{j=1}^{2} \frac{2}{j} + \sum_{j=1}^{3} \frac{3}{j} + \sum_{j=1}^{4} \frac{4}{j} = 1 + \left(\frac{2}{1} + \frac{2}{2} \right) + \left(\frac{3}{1} + \frac{3}{2} + \frac{3}{3} \right) + \left(\frac{4}{1} + \frac{4}{2} + \frac{4}{3} + \frac{4}{4} \right) = \frac{107}{6}$

(d) The value of this sum is 0 because the lower limit of summation (100) is larger than the upper limit (50).

Example 2.3.13 Write each of the following sums in sigma summation notation:

(a) $\frac{2}{(1)^2} + \frac{3}{(2)^2} + \frac{4}{(3)^2} + \frac{5}{(4)^2}$ (b) $1 + \frac{1}{3} + \frac{1}{9} + \frac{1}{27} + \cdots + \frac{1}{3^{10}}$

Solution (a) $\sum_{k=1}^{4} \frac{k+1}{k^2}$ or $\sum_{j=0}^{3} \frac{j+2}{(j+1)^2}$ or $\sum_{r=2}^{5} \frac{r}{(r-1)^2}$ (b) $\sum_{k=0}^{10} \frac{1}{3^k}$

Example 2.3.14 For each of the following sums, evaluate it numerically or express it in an algebraic closed form.

(a) $1000 + 999 + 998 + 997 + \cdots + 1$ (b) $100 + 101 + 102 + \cdots + 349 + 350$

(c) $5+10+15+\cdots+745+750$ (d) $5+8+11+14+\cdots+98=\sum_{k=1}^{32}(3k+2)$

(e) $1+2+3+\cdots+(2n-1)+2n$ (Express your answer in terms of n.)

(f) $\sum_{r=1}^{2^n} r$ (g) $\sum_{t=1+m^2}^{m^3} t$

Solution (a) Triangular Sum: $1+2+3+\cdots+1000=\dfrac{1000\cdot 1001}{2}=500500.$

(b) A Generalized Triangular Sum: $100+101+102+\cdots+349+350=$
$=(350-100+1)\frac{100+350}{2}=251\cdot 225=56,475$. Alternatively, the sum is

$$(1+2+3+\cdots+350)-(1+2+3+\cdots+99)=\frac{350\cdot 351}{2}-\frac{99\cdot 100}{2}=56,475.$$

(c) Factor out 5: $5(1+2+3+\cdots+150)=5\dfrac{150\cdot 151}{2}=56,625.$

(d) $\sum_{k=1}^{32}(3k+2)=\sum_{k=1}^{32}3k+\sum_{k=1}^{32}2=3\sum_{k=1}^{32}k+2\cdot 32=3\dfrac{32\cdot 33}{2}+64=1648.$

(e) Replace n by $2n$ in the Triangular Sum formula: $1+2+\cdots+2n=\dfrac{2n(2n+1)}{2}$
$=2n^2+n$.

(f) Replace n by 2^n in the Triangular Sum formula: $\sum_{r=1}^{2^n} r=\dfrac{2^n(2^n+1)}{2}$.

(g) A Generalized Triangular Sum: $(m^3-(1+m^2)+1)\dfrac{1+m^2+m^3}{2}=$
$m^2(m-1)\dfrac{1+m^2+m^3}{2}$.

Example 2.3.15 Consider the following nested pair of loops:

```
int j, k;
for (k = 2; k <= 50; ++k)
  for (j = 1; j < k; ++j)
    cout << "hello" << endl;
```

How many times, exactly, will the word "hello" be sent to output by the block of code shown above?

Solution When `k` is given the value 2 in the outer loop, the inner loop then outputs "hello" just once because j is initialized to 1 and then the body of the loop is executed only while `j < 2`. Next, `k` is incremented to the value 3 in the outer loop; then the inner loop outputs "hello" twice. Next, `k` is incremented to the value 4 in the outer loop; then the inner loop outputs "hello" three times. Etc. When `k` is given the value 50, the inner loop outputs "hello" 49 times. Then `k` is incremented once more, but

now the outer loop exits. The total number of times that "hello" is sent to output is $1 + 2 + 3 + \cdots + 49 = \dfrac{49 \cdot 50}{2} = 1225$.

Example 2.3.16 Suppose a block of high-level language computer code has 26 instructions, one after another. The *2-way complexity* of the block is the number of 2-way interactions that are possible between different instructions in the block. Assuming that any two instructions can give rise to an interaction, what is the 2-way complexity of the block of 26 instructions? *Hint:* suppose we label the instructions A through Z. Then A can interact with B, C, D, ..., Z, which is 25 possible interactions. Moreover, B can interact with C, D, ..., Z, which is 24 more possible interactions. Etc. How many are there altogether? More generally, if a block of code has n instructions in sequence, what is the 2-way complexity of the block?

Solution The total number of possible interactions is $25 + 24 + 23 + \cdots + 1 = \dfrac{25 \cdot 26}{2} = 325$. More generally, there are $\dfrac{(n-1)n}{2} \approx \dfrac{1}{2}n^2$ possible 2-way interactions in a block of n instructions. This is why we modularize our programs, to keep the complexity from growing as the square of the number of instructions.

Example 2.3.17 For each of the following sums, evaluate it or express it in closed form:

(a) $1 + 4 + 16 + 64 + \cdots + 4^7$ (b) $1 + \dfrac{1}{3} + \dfrac{1}{9} + \dfrac{1}{27} + \cdots + \dfrac{1}{3^{10}}$ [Hint: $\dfrac{1}{3^k} = (1/3)^k$]

(c) $1 + x + x^2 + \cdots + x^{n-1}$ (d) $\displaystyle\sum_{p=7}^{30} t^p$ (e) $\displaystyle\sum_{r=0}^{2n-1} 3^r$

(f) $1 + x^3 + x^6 + x^9 + \cdots + x^{3n}$ [Hint: $x^{3k} = (x^3)^k$.] (g) $\displaystyle\sum_{k=1}^{n} (\lfloor \lg(t) \rfloor)^k$

Solution (a) Geometric: $1 + 4^1 + 4^2 + \cdots + 4^7 = \dfrac{4^8 - 1}{4 - 1} = 21,845$, obtained using a calculator.

(b) $1 + (1/3)^1 + (1/3)^2 + \cdots + (1/3)^{10} = \dfrac{(1/3)^{11} - 1}{(1/3) - 1} = \dfrac{-.999994355}{-2/3} \approx 1.5$, by using a calculator.

(c) $\dfrac{x^{n-1+1} - 1}{x - 1} = \dfrac{x^n - 1}{x - 1}$ if $x \neq 1$. The sum is n if $x = 1$.

(d) Factor out t^7 : $t^7(1 + t + t^2 + \cdots + t^{23}) = t^7 \dfrac{t^{24} - 1}{t - 1} = \dfrac{t^{31} - t^7}{t - 1}$ if $t \neq 1$. The sum is 24 if $t = 1$.

(e) $\displaystyle\sum_{r=0}^{2n+1} 3^r = 1 + 3 + 3^2 + \cdots + 3^{2n+1} = \frac{3^{2n+2} - 1}{3 - 1} = \frac{(3^2)^{n+1} - 1}{2} = \frac{9^{n+1} - 1}{2}$.

(f) $1+x^3+(x^3)^2+(x^3)^3+(x^3)^4+\cdots+(x^3)^n=\dfrac{(x^3)^{n+1}-1}{x^3-1}$ if $x\neq 1$; the sum is $n+1$ is $x=1$.

(g) $(\lfloor \lg(t)\rfloor)^1+(\lfloor \lg(t)\rfloor)^2+(\lfloor \lg(t)\rfloor)^3+\cdots+(\lfloor \lg(t)\rfloor)^n=\lfloor \lg(t)\rfloor\Big(1+(\lfloor \lg(t)\rfloor)^1+(\lfloor \lg(t)\rfloor)^2+\cdots+(\lfloor \lg(t)\rfloor)^{n-1}\Big)$ if $t\neq 2$; the sum is n if $t=2$ because $\lg(2)=1$.
$=\lfloor \lg(t)\rfloor\dfrac{(\lfloor \lg(t)\rfloor)^n-1}{\lfloor \lg(t)\rfloor-1}$

Example 2.3.18 (With apologies to Mother Goose.) As I was coming from St. Ives, I met a man with 7 wives. Each wife had 7 sacks, each sack had 7 cats, each cat had 7 kits, each kit had 7 ticks. Ticks, kits, cats, sacks, wives, man. How many were *going to* St. Ives?

Solution $1(\text{man})+7(\text{wives})+7^2(\text{sacks})+7^3(\text{cats})+7^4(\text{kits})+7^5(\text{ticks})=\dfrac{7^6-1}{7-1}=19,608$.

Example 2.3.19 For each of the following sums, tell which variable the sum is a function of.

(a) $\displaystyle\sum_{k=1}^{n}\frac{k^2}{k+1}$ (b) $\displaystyle\sum_{n=1}^{p}2^{n+p}$ (c) $\displaystyle\sum_{j=1}^{100}j^k$

Solution (a) $\displaystyle\sum_{k=1}^{n}\frac{k^2}{k+1}$ is a function of the variable n; we can assign any positive integer value to n in the sum. For example, assigning n the value 25 produces $\displaystyle\sum_{k=1}^{25}\frac{k^2}{k+1}$. The given sum is NOT a function of k; we are not allowed to assign any value we choose to k; the values of k are determined by the value of n; whatever value of n we pick, the values of k will then be $1,\ldots,n$.

(b) $\displaystyle\sum_{n=1}^{p}2^{n+p}$ is a function of the variable p; we can assign any positive integer value to p in the sum. For example, if we assign p the value 4 we get the sum $\displaystyle\sum_{n=1}^{4}2^{n+4}=2^5+2^6+2^7+2^8$. The given sum is NOT a function of n; we cannot assign any values we choose to n; instead, the values of n are determined by the value of p; whatever value of p we pick, the values of n will be $1,2,\ldots,p$.

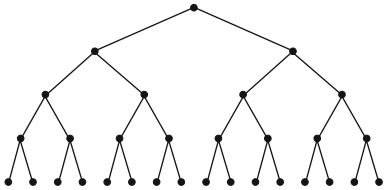

Fig. 2.6 A perfect tree with 5 levels

(c) $\sum_{j=1}^{100} j^k$ is a function of k; we can assign any positive integer value to k. For example, if we assign k the value 3 we get $\sum_{j=1}^{100} j^3$. The given sum is NOT a function of j. The values of j must be $1, 2, \ldots, 100$.

Example 2.3.20 A *perfect binary tree* is one in which every node has either no children (i.e., is a leaf) or 2 children, and all leaves of the tree are at the same level. In other words, every non-empty level in the tree is completely filled. It has 5 levels, numbered 0 (root level), 1, 2, 3, and 4 (bottom level). See Fig. 2.6 for an example.

(a) How many nodes are there in the level number k of a perfect binary tree?
(b) How many nodes are there altogether in a perfect binary tree with levels $0, 1, 2, \ldots, n$? Solve this by adding up the numbers in all the various levels.
(c) Do parts (a) and (b) over again, this time with a perfect ternary tree, i.e., a tree in which every non-leaf node has 3 children, and all leaves are at the same level.

Solution (a) 2^k (b) $2^0 + 2^1 + 2^2 + \cdots + 2^n = \dfrac{2^{n+1} - 1}{2 - 1} = 2^{n+1} - 1$
(c) 3^k, $\dfrac{3^{n+1} - 1}{2}$

Example 2.3.21 For each of the following sums, evaluate it or express it in a closed form.

(a) $1 + 2 \cdot 2 + 3 \cdot 2^2 + 4 \cdot 2^3 + \cdots + 10 \cdot 2^9$ (b) $\sum_{k=0}^{11} \dfrac{k+1}{2^k}$ *Hint:* $\dfrac{1}{2^k} = (1/2)^k$
(c) $t + 2t^2 + 3t^3 + \cdots + nt^n$ (Express your answer in terms of t and n.)

Solution (a) Take $t = 2$ and $n = 10$ in Theorem 2.3.7: $\dfrac{1 + 10 \cdot 2^{11} - 11 \cdot 2^{10}}{(2-1)^2} = 9217$.

(b) The sum is $1 + 2(1/2) + 3(1/2)^2 + 4(1/2)^3 + \cdots + 12(1/2)^{11} = \dfrac{1 + 12(1/2)^{13} - 13(1/2)^{12}}{((1/2) - 1)^2} = 4(1 + 12/2^{13} - 13/2^{12}) = 4 + \dfrac{6}{2^{10}} - \dfrac{13}{2^{10}} = 4 - \dfrac{7}{1024} \approx 4 - 0.007 = 3.993.$

(c) Factor out t : $t(1+2t+3t^2+\cdots+nt^{n-1}) = \dfrac{nt^{n+2} - (n+1)t^{n+1} + t}{(t-1)^2}$ if $t \neq 1$. The sum is n if $t = 1$.

Example 2.3.22 Use Theorem 2.3.4 to find lower and upper bounds for the sum $\sum_{k=1}^{n} \sqrt{k}$. The bounds will be functions of n.

Solution We apply Theorem 2.3.4 to the function $f(x) = \sqrt{x}$ on the interval $[1, n]$. Since this $f(x)$ is strictly increasing on $[1, n]$ for every positive integer n, Theorem 2.3.4 gives us

$$\sqrt{1} + \int_1^n \sqrt{x}\, dx \;<\; \sum_{k=1}^{n} \sqrt{k} \;<\; \sqrt{n} + \int_1^n \sqrt{x}\, dx \quad \text{for all integers } n > 1 .$$

Since $\int \sqrt{x}\, dx = \int x^{1/2}\, dx = \dfrac{2}{3} x^{3/2}$ the inequalities above yield

$$1 + \frac{2}{3} n^{3/2} - \frac{2}{3} \;<\; \sum_{k=1}^{n} \sqrt{k} \;<\; n^{1/2} + \frac{2}{3} n^{3/2} - \frac{2}{3} \quad \text{for all integers } n > 1 .$$

Example 2.3.23 Explain why $\sum_{k=1}^{2^n} \dfrac{1}{k}$ is *roughly* equal to $n \ln(2) \approx \dfrac{7}{10} n$. Also give a more precise approximation.

Solution $\sum_{k=1}^{n} \dfrac{1}{k} = \mathrm{H}(n) \approx \ln(n)$ (roughly), so $\sum_{k=1}^{2^n} \dfrac{1}{k} = \mathrm{H}(2^n) \approx \ln(2^n) = n \ln(2) \approx 0.693n \approx \dfrac{7}{10} n$. A more accurate approximation uses Euler's constant: $\sum_{k=1}^{n} \dfrac{1}{k} = \mathrm{H}(n) \approx \ln(n) + 0.5772$, so $\sum_{k=1}^{2^n} \dfrac{1}{k} = \mathrm{H}(2^n) \approx \ln(2^n) + 0.5772 = n \ln(2) + 0.5772 \approx 0.693n + 0.5772$.

Example 2.3.24 Use the inequalities we have derived and a pocket calculator to find a numerical lower bound and a numerical upper bound for the harmonic sum $1 + \dfrac{1}{2} + \dfrac{1}{3} + \cdots + \dfrac{1}{100}$. Also give the best estimate you can using Euler's approximation.

Solution As shown in Theorem 2.3.5, $\frac{1}{100} + \ln(100) < H(n) < 1 + \ln(100)$. Using a calculator we get 4.615 as a lower bound and 5.605 as an upper bound for H(100). Euler's approximation says that a very good estimate for H(100) is $0.5772 + \ln(100) \approx 5.182$.

Example 2.3.25 How many bits are there in the standard binary representation of the integer
725194×386011?

Solution As shown in Example 2.3.9, the answer is $1 + \lfloor \log_2(725194 \times 386011) \rfloor$. Using the fact that $\log_b(xy) = \log_b(x) + \log_b(y)$ and the fact that $\log_2(x) = \frac{\log_{10}(x)}{\log_{10}(2)}$, we obtain the answer

$$1 + \left\lfloor \frac{\log_{10}(725194)}{\log_{10}(2)} + \frac{\log_{10}(386011)}{\log_{10}(2)} \right\rfloor = 1 + \lfloor 19.468 + 18.588 \rfloor = 39$$

with the aid of a calculator.

2.3.2 Exercises

2.3.26 In each of these two problems, show the computations leading to a single numerical answer.

(a) If $f(x) = \lfloor \lg(x) \rfloor$, then $\sum_{k=1}^{7} f(k) = ?$ (b) If $f(x) = \lceil \lg(x) \rceil$, then $\sum_{k=1}^{9} f(k) = ?$

2.3.27 Write the following sums in sigma summation notation:

(a) $1 + 4 + 7 + 10 + \cdots + 97 + 100$ (b) $\frac{2}{\sqrt{1}} + \frac{4}{\sqrt{3}} + \frac{6}{\sqrt{5}} + \frac{8}{\sqrt{7}} + \cdots + \frac{1000}{\sqrt{999}}$

2.3.28 (a) Write the following sums in "3 dot" notation. Note that despite superficial resemblances, they are actually different sums: $\sum_{k=1}^{n} k^n$ and $\sum_{n=1}^{k} k^n$.

(b) Write the following sums in "3 dot" notation. Note that despite superficial resemblances, they are actually different sums: $\sum_{k=1}^{n} \frac{k+1}{n+1}$ and $\sum_{n=1}^{k} \frac{k+1}{n+1}$.

2.3.29 Suppose a positive integer n is given in binary representation by a sum of the form

$$n = b_k 2^k + b_{k-1} 2^{k-1} + \cdots + b_2 2^2 + b_1 2^1 + b_0 2^0.$$

where the numbers $b_0, b_1, b_2, \ldots, b_k$ denote the binary digits (the bits) in the *standard* binary representation of n (i.e., no leading zeroes; see Example 2.3.9). Express

the value of the number k here in terms of the integer n. Then write the equation above in sigma summation notation.

2.3.30 There is a well-known algorithm for deriving the binary representation of a positive integer n from its decimal (base 10) representation. Using the notation in Exercise 2.3.29 above, the least significant bit b_0 is calculated as the remainder of n after division by 2. That is, $b_0 = 0$ if n is even, but $b_0 = 1$ if n is odd. That is, $b_0 = \text{parity}(n)$, so $b_0 = n - 2\lfloor n/2 \rfloor$. Then we divide n by 2 and discard the remainder to produce a smaller integer, call it n_1, which is given by the equation

$$n_1 = \left\lfloor \frac{n}{2} \right\rfloor = b_k 2^{k-1} + b_{k-1} 2^{k-2} + \cdots + b_2 2^1 + b_1 2^0$$

(the b_0 term has been discarded). Using the same procedure, we can now calculate the value of b_1 : it's the remainder after n_1 is divided by 2; that is,

$$b_1 = \text{parity}(n_1) = n_1 - 2\left\lfloor \frac{n_1}{2} \right\rfloor = \left\lfloor \frac{n}{2} \right\rfloor - 2\left\lfloor \frac{n}{2^2} \right\rfloor.$$

Now we divide n_1 by 2 and discard the remainder to produce a smaller integer n_2:

$$n_2 = \left\lfloor \frac{n_1}{2} \right\rfloor = b_k 2^{k-2} + b_{k-1} 2^{k-3} + \cdots + b_2 2^0.$$

(a) Express b_2 as a difference of two floor expressions involving n.
(b) Express b_3 in a similar way as a difference of two floor expressions involving n. Generalize.
(c) For all positive integers n let $B(n)$ denote the number of 1 s in the standard binary representation of n. Then $B(1) = 1$, $B(2) = 1$, $B(3) = 2$, $B(4) = 1$, $B(5) = 2$, $B(6) = 2$, $B(7) = 3$, and so on. In the notation we have used in this exercise and Exercise 2.3.29,

$$B(n) = b_0 + b_1 + b_2 + \cdots + b_{k-1} + b_k$$

Into this sum, substitute the formulas for b_0, b_1, b_2, b_3, b_4 etc. derived in earlier parts of this exercise, and simplify a little to derive this formula: $B(n) = n - \sum_{p=1}^{\lfloor \lg(n) \rfloor} \left\lfloor \frac{n}{2^p} \right\rfloor$. You will need to use the fact that for all positive integers n, $\left\lfloor \frac{n}{2^{\lfloor \lg(n) \rfloor} + 1} \right\rfloor = 0$ (Theorem 2.2.8 (b)).

2.3.31 For each of the following sums, tell what variable the sum is a function of. In other words, if you computed each of those sums, what would the resulting expression be a function of? See Example 2.3.19 for a similar exercise.

(a) $\sum_{k=1}^{m} \frac{m-k}{m+k}$ (b) $\sum_{p=1}^{20} \lfloor \lg(p+t) \rfloor$ (c) $\sum_{i=1}^{v} v^i$ (d) $\sum_{r=p}^{2^p} r^3$ (e) $\sum_{n=i}^{100} i^n$

2.3.32 Evaluate the following sums or express them in closed form.

(a) $50 + 49 + 48 + 47 + \cdots + 10$
(b) $100 + 102 + 104 + 106 + 108 + \cdots + 200$
(c) $1 + 2 + 3 + \cdots + (n-3) + (n-2)$ (Express your answer in terms of n.)
(d) $n + (n+1) + (n+2) + \cdots + (3n-1) + 3n$ (Express your answer in terms of n.)
(e) $\sum_{k=1}^{1000}(2k+7)$
(f) $\sum_{i=n}^{n^2} i$ (Express your answer in terms of n.)
(g) $\frac{1}{n} + \frac{2}{n} + \frac{3}{n} + \cdots + \frac{n}{n}$ (in terms of n)
(h) $n + 2n + 3n + \cdots + n^2$ (express in terms of n)

2.3.33 Evaluate the following sums or express them in closed form.

(a) $25 + 26 + 27 + \ldots + 75$
(b) $100 + 99 + 98 + 97 + \cdots + 31$
(c) $1 + 2 + 3 + \cdots + (n+1) + (n+2)$ (Express your answer in terms of n.)
(d) $2n + (2n+1) + (2n+2) + \cdots + (5n-1) + 5n$ (Express your answer in terms of n.)
(e) $\sum_{k=1}^{100}(7k+2)$
(f) $\sum_{j=\lfloor\sqrt{n}\rfloor}^{2\lfloor\sqrt{n}\rfloor} j$ (Express your answer in terms of n.)
(g) $n^2 + 2n^2 + 3n^2 + \cdots + 100n^2$ (Express your answer in terms of n.)
(h) $\frac{1}{n+1} + \frac{2}{n+1} + \frac{3}{n+1} + \cdots + \frac{n}{n+1}$ (Express your answer interms of n.)

2.3.34 Consider the block of code in Example 2.3.15.

(a) How many times, exactly, will the initialization "`j = 1`" be executed in that block of code?
(b) How many times, exactly, will the inequality "`j < k`" be tested (i.e., evaluated)?
(c) How many times, exactly, will the increment operation "`++j`" be executed?

2.3.35 Give a closed form for each of the following sums.

(a) $2^p + 2^{2p} + 2^{3p} + 2^{4p} + \cdots + 2^{mp} = \sum_{k=1}^{m} 2^{kp}$, where p is a non-zero real constant. Hint: $2^{xy} = (2^x)^y = (2^y)^x$.
(b) $(a+b)^2 + (a+b)^3 + (a+b)^4 + \cdots + (a+b)^n = \sum_{k=2}^{n}(a+b)^k$, where a and b are real constants with $a + b \neq 1$.
(c) $n + 2n + 4n + 8n + \cdots + n2^n = \sum_{k=0}^{n} n2^k$, where n is a positive integer.

2.3.36 Give a closed form for each of the following sums.

(a) $3^{p+1} + 3^{p+2} + 3^{p+3} + \cdots + 3^{p+m} = \sum_{k=1}^{m} 3^{p+k}$, where p is a non-zero real constant. Hint: $3^{x+y} = 3^x \cdot 3^y$.

(b) $(1+x)^2+(1+x)^3+(1+x)^4+\cdots+(1+x)^n=\sum_{k=2}^{n}(1+x)^k$, where x is a non-zero real constant.

(c) $\frac{1}{n}+\frac{2}{n}+\frac{4}{n}+\frac{8}{n}+\cdots+\frac{2^n}{n}=\sum_{k=0}^{n}\frac{2^k}{n}$, where n is a positive integer.

2.3.37 (a) Let t be a real number not equal to -1. Give a closed form for the following sum:

$$1+\left(\frac{t}{t+1}\right)+\left(\frac{t}{t+1}\right)^2+\left(\frac{t}{t+1}\right)^3+\cdots+\left(\frac{t}{t+1}\right)^n=\sum_{p=0}^{n}\left(\frac{t}{t+1}\right)^p$$

Your answer should be one of the following:

$$(i)\,\frac{(t+1)^{n+1}-t^{n+1}}{(t+1)^n}\qquad (ii)\,\frac{(t+1)^n-t^{n+1}}{(t+1)^n}\qquad (iii)\,\frac{t^n+t^{n+1}}{(t+1)^n}$$

(b) Let x be a real number not equal to 1. Give a closed form for the following sum:

$$1+\sqrt[9]{x}+(\sqrt[9]{x})^2+(\sqrt[9]{x})^3+\cdots+(\sqrt[9]{x})^9=\sum_{r=0}^{9}(\sqrt[9]{x})^r$$

2.3.38 A clever man has done the sultan a tremendous favor, and the sultan asks the man to name his own payment. The man says, "Oh, I don't really want very much. Just place a grain of corn on one square of this chessboard, then 2 grains on another square, and twice 2 grains on another square, and so on, doubling the number of grains each time, until all the squares of the board are filled." How many grains of corn has the man asked to be paid? Express the answer in closed form. (Note: for those who don't know, there are 64 squares on a chessboard.) How many decimal digits will be required to express the answer in base 10?

2.3.39 Prove the part of Theorem 2.3.4 that deals with *increasing* functions.

2.3.40 (a) Prove that if $f(x)$ is a strictly decreasing function defined over an interval $[m-1, n+1]$, where m and n are integers such that $m \le n$, then

$$\int_{m}^{n+1} f(x)\mathrm{d}x \;<\; \sum_{k=m}^{n} f(k) \;<\; \int_{m-1}^{n} f(x)\mathrm{d}x.$$

(b) Prove that if $f(x)$ is a strictly increasing function defined over the interval $[m-1, n+1]$, then

$$\int_{m-1}^{n} f(x)\mathrm{d}x \;<\; \sum_{k=m}^{n} f(k) \;<\; \int_{m}^{n+1} f(x)\mathrm{d}x.$$

2.3.41 Use Theorem 2.3.4 to prove that for all real numbers $p > 0$,

$$1 + \frac{1}{p+1}n^{p+1} < \sum_{k=1}^{n} k^p < n^p + \frac{1}{p+1}n^{p+1}$$

2.3.42 Evaluate each of the following sums by computing the values of all the fractions in the sum and adding them all up (use a calculator to do this). In each case, compare your answer with the Euler approximation given in Theorem 2.3.6 (use a calculator for this also).

(a) $\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{17}$ (b) $\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{15}$.

2.3.43 (a) Use a calculator to help you compute a fairly exact numerical value for the sum

$$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{1500} = \sum_{k=1}^{1500} \frac{1}{k}.$$

(b) Use a calculator to help you compute a fairly exact numerical value for the sum

$$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{2000} = \sum_{k=1}^{2000} \frac{1}{k}.$$

2.3.44 (a) Use a calculator to help you compute a fairly exact numerical value for the sum

$$\frac{1}{250} + \frac{1}{251} + \frac{1}{252} + \cdots + \frac{1}{700} = \sum_{k=250}^{700} \frac{1}{k}.$$

(b) Use a calculator to help you compute a fairly exact numerical value for the sum

$$\frac{1}{400} + \frac{1}{401} + \frac{1}{402} + \cdots + \frac{1}{650} = \sum_{k=400}^{650} \frac{1}{k}.$$

2.3.45 (a) Explain why $\sum_{k=1}^{mn} \frac{1}{k}$ is roughly equal to $\sum_{k=1}^{m} \frac{1}{k} + \sum_{k=1}^{n} \frac{1}{k}$ for all large positive integers m and n.

(b) Explain why $\sum_{k=1}^{100n} \frac{1}{k}$ is approximately equal to $5.18 + \sum_{k=1}^{n} \frac{1}{k}$ for all large positive integers n. You may use a calculator.

2.3.46 (a) Show that for all large positive integers n the sum $\frac{1}{2n+1} + \frac{1}{2n+2} + \frac{1}{2n+3} + \cdots + \frac{1}{3n}$ is approximately equal to 0.405. You may use a calculator.

(b) Show that for all large positive integers n the sum $\frac{1}{n+1}+\frac{1}{n+2}+\frac{1}{n+3}+\cdots+\frac{1}{2n}$ is approximately equal to 0.693. You may use a calculator.

2.3.47 Find an approximate closed form for the sum $\frac{n}{1}+\frac{n}{2}+\frac{n}{3}+\frac{n}{4}+\cdots+\frac{n}{n}$.

2.3.48 (Index Shifting in Sigma Sums) In Example 2.3.13, p. 41, the instruction is to write the sum $\frac{2}{1^2}+\frac{3}{2^2}+\frac{4}{3^2}+\frac{5}{4^2}$ in sigma summation notation. Three answers were given:

$$\sum_{k=1}^{4}\frac{k+1}{k^2} \quad \text{or} \quad \sum_{j=0}^{3}\frac{j+2}{(j+1)^2} \quad \text{or} \quad \sum_{r=2}^{5}\frac{r}{(r-1)^2}$$

It is possible to pass from one of these answers to another by a mechanical process of substitution.

For example, starting from the first sum, which we write in the form $\sum_{k=1}^{k=4}\frac{k+1}{k^2}$ (emphasizing that the upper limit 4 is the last value of k), replace k everywhere by $j+1$ to obtain $\sum_{j+1=1}^{j+1=4}\frac{j+1+1}{(j+1)^2}$. Simplification reduces this to the answer $\sum_{j=0}^{j=3}\frac{j+2}{(j+1)^2}$. Alternatively, replace k everywhere by $r-1$ to obtain $\sum_{r-1=1}^{r-1=4}\frac{r-1+1}{(r-1)^2}$, which simplifies to $\sum_{r=2}^{r=5}\frac{r}{(r-1)^2}$.

(a) What sum is obtained if you replace k everywhere by $k+a$ in the sum $\sum_{k=a}^{b}\frac{k-a}{k+a}$?

(b) The sum $\sum_{m=2}^{21}\lfloor\lg(m+3)\rfloor$ is equal to the sum $\sum_{p=?}^{?}\lfloor\lg(p)\rfloor$ (give the correct limits of summation in the second sum).

2.3.49 (a) Calculate the exact number of digits in the standard decimal representation of 7702914^5.

(b) Calculate the exact number of bits in the standard binary representation of 1024^{99}.

2.3.50 Use mathematical induction to prove that for every positive integer n,

$$\sum_{k=1}^{n}k^2=\frac{n(n+1)(2n+1)}{6}.$$

2.3.51 (This exercise requires a knowledge of combinatorial counting techniques.) Let n denote a positive integer. Consider any set S that contains exactly n elements. Label the elements of S by $x_1,\ x_2, \ldots,\ x_n$. Now take k to be any positive integer somewhat less than n.

(a) How many ways can we construct a subset of S of size k?
(b) How many ways can we construct a subset of S of size k that contains the last element x_n?
(c) How many ways can we construct a subset of S of size k that contains x_{n-1} but not x_n?
(d) How many ways can we construct a subset of S of size k that contains x_{n-2} but not x_{n-1} and not x_n?
(e) How many ways can we construct a subset of S of size k that contains x_{k+1} but not any of the elements $x_{k+2} \ldots x_n$?
(f) How many ways can we construct a subset of S of size k that contains x_k but not any of the elements $x_{k+1} \ldots x_n$?
(g) Use parts (a) through (f) above to help explain why

$$\binom{n-1}{k-1}+\binom{n-2}{k-1}+\binom{n-3}{k-1}+\cdots+\binom{k+1}{k-1}+\binom{k}{k-1}+\binom{k-1}{k-1}=\binom{n}{k}.$$

(h) Show that if you divide both sides of the equation in part (g) by $\binom{n}{k}$ and perform a lot of cancellations you produce the following summation formula:

$$\frac{k}{n}+\frac{(n-k)k}{n(n-1)}+\frac{(n-k)(n-k-1)k}{n(n-1)(n-2)}+\cdots+\frac{(n-k)(n-k-1)(n-k-2)...(1)k}{n(n-1)(n-2)(n-3)...(k)}=1.$$

(i) Show that by factoring out a common factor on the left side and then performing a little bit of algebra, you can produce the following summation formula:

$$1+\frac{n-k}{n-1}+\frac{(n-k)(n-k-1)}{(n-1)(n-2)}+\cdots+\frac{(n-k)(n-k-1)(n-k-2)...(1)}{(n-1)(n-2)(n-3)...(k)}=\frac{n}{k}.$$

(j) Let b denote the integer $n-1$, and let a denote the integer $n-k$. (Note that $a \leq b$.) Then

$$1+\frac{a}{b}+\frac{a(a-1)}{b(b-1)}+\frac{a(a-1)(a-2)}{b(b-1)(b-2)}+\cdots+\frac{a!}{b(b-1)(b-2)...(b-a+1)}=\_\_\_\_\_.$$

(fill in the blank on the right side of the equation)

2.4 Factorials

Definition 2.4.1 The *factorial function* is defined on the non-negative integers as follows:

$$0!=1,\ 1!=1,\ \text{and } n!=1\cdot 2\ldots(n-1)\cdot n=n\cdot(n-1)\ldots 2\cdot 1 \quad \text{for all } n\geq 2,$$

where "$n!$" is read "n factorial". Alternatively, one can define factorials recursively as follows:

$$0! = 1, \text{ and } n! = (n-1)! \cdot n \quad \text{for all } n \geq 1.$$

Examples 2.4.2 (a) Is $(2n)! = 2(n!)$ in general?
No. The left side is $(2n)! = (2n)(2n-1)(2n-2)\ldots 2 \cdot 1$. The right side is $2[n(n-1)\ldots 2 \cdot 1]$.

(b) Express the following product in terms of factorials: $2 \cdot 4 \cdot 6 \cdot 8 \ldots (2n)$.
The solution involves noticing that each factor is divisible by 2, so we could factor out all the 2s and leave $1 \cdot 2 \cdot 3 \cdot 4 \ldots n$. There are n 2s, so we find that $2 \cdot 4 \cdot 6 \cdot 8 \ldots (2n) = 2^n \cdot n!$.

(c) Express the following product in terms of factorials: $1 \cdot 3 \cdot 5 \cdot 7 \ldots (2n-1)$.
The solution for this involves introducing "missing" factors that could give us a factorial, and then compensating by dividing the expression by the introduced factors. That is,

$$1 \cdot 3 \cdot 5 \cdot 7 \ldots (2n-1) = \frac{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \ldots (2n-1)(2n)}{2 \cdot 4 \cdot 6 \cdot 8 \ldots (2n)} = \frac{(2n)!}{2^n \cdot n!}.$$

The factorial function $n!$ grows extremely rapidly as n increases; e.g. $10! \approx 3,000,000$. In fact, with most C/C++ compilers, the value of 13! overflows a `long int` variable. Thus, when computing with factorials in a C/C++ program, you'll almost always need to use double precision floating point variables instead of integer or long integer variables.

We can get some idea of how fast the factorial function grows by using our theorem on approximation of sums by integrals. We start by deriving a valuable approximation for $\ln(n!)$, which turns up from time to time in the analysis of algorithms.

Theorem 2.4.3 *For all integers $n \geq 2$,*

$$n\ln(n) - n + 1 < \ln(n!) < (n+1)\ln(n) - n + 1. \tag{2.8}$$

Proof Use Theorem 2.2.3 (a) to write

$$\ln(n!) = \ln(1) + \ln(2) + \ln(3) + \cdots + \ln(n) = \sum_{k=1}^{n} \ln(k).$$

This identity allows us to use Theorem 2.3.4 to obtain lower and upper bounds for $\ln(n!)$. The function $\ln(x)$ is strictly increasing on the open interval $(0, +\infty)$, so Theorem 2.3.4 gives us

$$\ln(1) + \int_1^n \ln(x)\mathrm{d}x \; < \; \sum_{k=1}^{n} \ln(k) \; < \ln(n) + \int_1^n \ln(x)\mathrm{d}x \quad \text{for all } n > 1.$$

From an integral table or integration by parts we find that $\int \ln(x)\mathrm{d}x = x\ln(x) - x$, so

$$0 + \Big[x\ln(x) - x\Big]_{x=1}^{x=n} < \sum_{k=1}^{n} \ln(k) < \ln(n) + \Big[x\ln(x) - x\Big]_{x=1}^{x=n} \text{ for all } n > 1,$$

and thus

$$[n\ln(n) - n] - [0 - 1] < \ln(n!) < \ln(n) + [n\ln(n) - n] - [0 - 1].$$

This simplifies to the inequalities stated in the theorem. ■

Corollary 2.4.4 *For all integers $n > 1$,*

$$e\left(\frac{n}{e}\right)^n < n! < ne\left(\frac{n}{e}\right)^n.$$

Proof Starting from the inequality (2.8) in Theorem 2.4.3, we find that

$$\ln(n^n) - n + 1 < \ln(n!) < \ln(n^n) + \ln(n) - n + 1.$$

Exponentiating gives

$$\mathrm{e}^{\ln(n^n)-n+1} < \mathrm{e}^{\ln(n!)} < \mathrm{e}^{\ln(n^n)-n+\ln(n)+1},$$

which in turn gives

$$n^n\mathrm{e}^{-n}\mathrm{e}^1 < n! < n^n\mathrm{e}^{-n}n\mathrm{e}^1. \quad ■$$

If you think of $n!$ as growing roughly at the same rate as n^n you'll be in the right ballpark. When we need an even more accurate estimate for the growth of $n!$ we can use an approximation discovered several hundred years ago by the English mathematician John Stirling.

Theorem 2.4.5 Stirling's Formula: $n! \approx \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$ *for all large values of n. More precisely,*

$$\lim_{n\to\infty} \frac{n!}{(n/e)^n\sqrt{2\pi n}} = 1.$$

Proof This can be found in most books on advanced calculus. The proof is somewhat lengthy. ■

In Stirling's approximation, the *percent error* approaches zero as $n \to \infty$. This is the meaning of the formulation involving the limit of 1. By contrast, the *absolute error*, i.e., the difference between the left and right sides of the approximation formula, grows without bound as $n \to \infty$.

2.4.1 Further Examples

Example 2.4.6 Compute $\frac{13!}{4!\,9!}$ and $\frac{20!}{15!\,5!}$ as efficiently as possible without using a calculator.

Solution $\frac{13!}{4!\,9!} = \frac{10 \cdot 11 \cdot 12 \cdot 13}{2 \cdot 3 \cdot 4} = 5 \cdot 11 \cdot 13 = 715;$
$\frac{20!}{15!\,5!} = \frac{16 \cdot 17 \cdot 18 \cdot 19 \cdot 20}{2 \cdot 3 \cdot 4 \cdot 5} = 8 \cdot 17 \cdot 6 \cdot 19 = 15504.$

Example 2.4.7 In each case below, write out the three largest factors of the product indicated. Assume that n is a positive integer of size at least 6.
(a) $(n^2)!$ (b) $(\lfloor n/2 \rfloor)!$ (c) $(2^n)!$

Solution (a) $(n^2)(n^2-1)(n^2-2)\ldots(1)$
(b) $\lfloor n/2 \rfloor(\lfloor n/2 \rfloor - 1)(\lfloor n/2 \rfloor - 2)\ldots(1)$
(c) $2^n(2^n-1)(2^n-2)\ldots(1)$

Example 2.4.8 Simplify the expressions $\frac{(n+1)!}{n!}$, $\frac{n!}{(n+2)!}$, and $\frac{(n-1)!}{(n+1)!}$, where n is a positive integer.

Solution $\frac{(n+1)!}{n!} = n+1;$
$\frac{n!}{(n+2)!} = \frac{n!}{(n+2)(n+1)\cdot n!} = \frac{1}{(n+2)(n+1)}$
$\frac{(n-1)!}{(n+1)!} = \frac{(n-1)!}{(n-1)!n(n+1)} = \frac{1}{n(n+1)}$

Example 2.4.9 Calculate 12! exactly, and then compare with the value you get using Stirling's approximation and a calculator.

Solution $12! = 479,001,600;$ $\left(\frac{12}{e}\right)^{12}\sqrt{2\pi \cdot 12} \approx 475,687,500.$

The percent error in the approximation is $\frac{479001600 - 475687500}{479001600} \cdot 100 \approx 0.7\,\%$, i.e., the error is less than 1 % (even though the magnitude of the error is more than 3 million).

2.4.2 Exercises

2.4.10 Without using a calculator, find the exact value of
(a) $\frac{82!}{3!\,79!}$ (b) $\frac{42!}{3!\,39!}$.

2.4.11 In each case below, write out the four largest factors of the product indicated. Assume that n is a positive integer of size at least 16.
(a) $(2n+3)!$ (b) $(\lfloor \lg(n) \rfloor)!$

2.4.12 In each case below, write out the four largest factors of the product indicated. Assume that n is a positive integer of size at least 3.
(a) $(5n+2)!$ (b) $(n!)!$

2.4.13 Simplify the following expressions in which n is a positive integer:
(a) $\dfrac{(3n+1)!}{(3n-1)!}$ (b) $\dfrac{(2n-1)!}{(2n+1)!}$

2.4.14 Use the multiplication key on your calculator to compute 15! (don't use the ! key if your calculator has one). Then use the calculator to compute Stirling's approximation for 15!. Calculate the percent error in the approximation.

2.5 Remainders in Integer Division

In this section we'll explore an arithmetic concept that plays an important role in a number of interesting algorithms. We begin with the mathematical definition of a "remainder" in integer division.

Definition 2.5.1 Let n be any integer and let m be a strictly positive integer. Then the *remainder of n divided by m* (also called the *residue of n modulo m*) is defined to be the integer

$$n - m\left\lfloor \frac{n}{m} \right\rfloor.$$

We denote this integer by the expression $n \bmod m$.

Examples 2.5.2 $27 \bmod 12 = 3$ because $27 - 12\left\lfloor \dfrac{27}{12} \right\rfloor = 27 - 12\lfloor 2.25 \rfloor = 27 - 12(2) = 3$.

$33 \bmod 7 = 5$ because $33 - 7\left\lfloor \dfrac{33}{7} \right\rfloor = 33 - 7\lfloor 4.714 \rfloor = 33 - 7(4) = 5$.

$72 \bmod 9 = 0$ because $72 - 9\left\lfloor \dfrac{72}{9} \right\rfloor = 72 - 9\lfloor 8 \rfloor = 72 - 9(8) = 0$.

$6 \bmod 10 = 6$ because $6 - 10\left\lfloor \dfrac{6}{10} \right\rfloor = 6 - 10\lfloor 0.6 \rfloor = 6 - 10(0) = 6$.

$-27 \bmod 12 = 9$ because $-27 - 12\left\lfloor \dfrac{-27}{12} \right\rfloor = -27 - 12\lfloor -2.25 \rfloor = -27 - 12(-3) = 9$.

$-33 \bmod 7 = 2$ because $-33 - 7\left\lfloor \dfrac{-33}{7} \right\rfloor = -33 - 7\lfloor -4.714 \rfloor = -33 - 7(-5) = 2$.

$$-72 \bmod 9 = 0 \text{ because } -72 - 9\left\lfloor \frac{-72}{9} \right\rfloor = -72 - 9\lfloor -8 \rfloor = -72 - 9(-8) = 0.$$

$$-6 \bmod 10 = 4 \text{ because } -6 - 10\left\lfloor \frac{-6}{10} \right\rfloor = -6 - 10\lfloor -0.6 \rfloor = -6 - 10(-1) = 4.$$

Integers of the form $n \bmod 12$ can be visualized in a simple way involving a clock. Imagine that the hours on the face of a clock are labeled $1, 2, 3, \ldots, 11$, and 0 (instead of the usual 12). Then for any positive integer n the integer $n \bmod 12$ will turn out to be the label we stop at if we advance the hour hand clockwise by n hours, starting at label 0. In Examples 2.5.2 we saw that $27 \bmod 12 = 3$ and indeed, advancing the hour hand by 27 h, starting at 0, takes the hand past 0 twice (24 h go by) and brings it to the label 3. As a second example, $60 \bmod 12 = 0$ because advancing the hour hand by 60 h moves it full circle 5 times. For a negative integer n we can obtain $n \bmod 12$ by turning the hour hand backward (counter-clockwise) by $|n|$ hours, starting at label 0. in Examples 2.5.2 we saw that $-27 \bmod 12 = 9$ and now we see that turning the hour hand backward by 27 h brings it to the label 9. As still another example, $-5 \bmod 12 = 7$ because turning the hour hand backward by 5 h brings it to the label 7. And of course $-36 \bmod 12 = 0$.

It is easy to generalize this clock interpretation to expressions of the form $n \bmod m$, where m is any positive integer. Just replace the standard 12 h clock with a clock whose circular face is divided into m periods of time with labels $\{0, 1, 2, \ldots, m-1\}$.

You might ask whether the integer $n \bmod m$, as we've specified it in Definition 2.5.1, always turns out to belong to the set $\{0, 1, 2, \ldots, m-1\}$. The following theorem says that it does.

Theorem 2.5.3 *For every integer n and positive integer m the remainder $n \bmod m$ satisfies the inequalities $0 \leq n \bmod m < m$. Furthermore, if $0 \leq n < m$, then $n \bmod m = n$.*

Proof By Theorem 2.1.4 (a), with $\frac{n}{m}$ in place of x, we have $\frac{n}{m} - 1 < \left\lfloor \frac{n}{m} \right\rfloor \leq \frac{n}{m}$. Multiplying throughout by the positive integer m yields $n - m < m\left\lfloor \frac{n}{m} \right\rfloor \leq n$. Subtracting n from each part yields $-m < -n + m\left\lfloor \frac{n}{m} \right\rfloor \leq 0$. Multiplying all parts by -1 reverses the directions of the inequalities to produce $m > n - m\left\lfloor \frac{n}{m} \right\rfloor \geq 0$. This is equivalent to the double inequality stated in the theorem.

For the second part, $0 \leq n < m$ implies $0 \leq \frac{n}{m} < 1$, and so $\left\lfloor \frac{n}{m} \right\rfloor = 0$. It follows that $n \bmod m = n - m\left\lfloor \frac{n}{m} \right\rfloor = n$. ■

Algorithms that make use of "modular arithmetic" typically involve addition and multiplication of integers followed by a "mod" operation. For example, most encryption algorithms involve computing many remainders of the form $n^p \bmod m$ for

positive integers n, p, and m that are several hundred digits long. If the only way this computation could be performed was to first raise n to the power p and afterward carry out the "mod" operation, the computation of n^p would generate integers having trillions of trillions of digits—far too many to be stored in a computer's memory. Fortunately, the computation can be accomplished using the following useful facts about modular arithmetic.

Theorem 2.5.4 *For all integers n and q and positive integers m,*

$$(n + m\,q)\ mod\ m = n\ mod\ m.$$

Proof By the formulas in Definition 2.5.1,

$$(n + m\,q) \bmod m = n + m\,q - m\left\lfloor \frac{n + m\,q}{m} \right\rfloor = n + m\,q - m\left\lfloor \frac{n}{m} + q \right\rfloor.$$

Since q is an integer, it can be extracted from the floor by Theorem 2.1.5 (a), which results in

$$n + m\,q - m\,q - m\left\lfloor \frac{n}{m} \right\rfloor = n - m\left\lfloor \frac{n}{m} \right\rfloor.$$

We recognize this last expression as $n \bmod m$ by Definition 2.5.1. ■

Theorem 2.5.5 Division Theorem. *For every integer n and and positive integer m, there exist unique integers q and r such that* $n = q\,m + r$ *and* $0 \leq r < m$.

Proof If we denote $\left\lfloor \frac{n}{m} \right\rfloor$ by the letter q and $n \bmod m$ by the letter r, then by Definition 2.5.1 and Theorem 2.5.3 the integers q and r satisfy the properties in the theorem. We still must prove that they are unique.

Suppose that we have $q_1,\ q_2,\ r_1,\ r_2, 0 \leq r_1, r_2 < m$ such that $n = q_1\,m + r_1$ and $n = q_2\,m + r_2$. Then

$$q_1\,m + r_1 = q_2\,m + r_2$$

which can be rewritten as

$$(q_1 - q_2)m = r_2 - r_1$$

which yields

$$q_1 - q_2 = \frac{r_1 - r_2}{m}.$$

This shows that $\dfrac{r_1 - r_2}{m}$ is an integer.

We know that $0 \leq r_2 < m$. From $0 \leq r_1 < m$ we deduce that $-m < -r_1 \leq 0$. By combining these inequalities we get $-m < r_2 - r_1 < m$, which yields $-1 < \dfrac{r_1 - r_2}{m} < 1$. Since we have shown that $\dfrac{r_1 - r_2}{m}$ is an integer, and we now see that is lies strictly between -1 and 1, we must conclude that $\dfrac{r_1 - r_2}{m}$ is 0. This implies that $r_1 = r_2$, which then yields $q_1 = q_2$. ■

Example 2.5.6 As an example of the usefulness of the Division Theorem 2.5.5, let's use it to prove the following: for all integers n and positive integers m,

$$\left\lfloor \frac{n}{m} \right\rfloor = \left\lceil \frac{n+1}{m} \right\rceil - 1.$$

The proof goes as follows: let q and r denote the unique integers for which $n = qm+r$ and $0 \leq r < m$. Then $\left\lfloor \frac{n}{m} \right\rfloor = q$. We'll now proceed to prove that $\left\lceil \frac{n+1}{m} \right\rceil - 1 = q$. We note that

$$\left\lceil \frac{n+1}{m} \right\rceil - 1 = \left\lceil q + \frac{r+1}{m} \right\rceil - 1 = q + \left\lceil \frac{r+1}{m} \right\rceil - 1.$$

Now $1 \leq r+1 \leq m$, so $\frac{1}{m} \leq \frac{r+1}{m} \leq 1$, from which it follows that $\left\lceil \frac{r+1}{m} \right\rceil = 1$. Thus,

$$q + \left\lceil \frac{r+1}{m} \right\rceil - 1 = q + 1 - 1 = q$$

as desired. This completes the proof.

Theorem 2.5.7 *Let a and b be any integers, and let m be a positive integer. The remainder $(a+b)$ mod m can be computed by first computing the separate remainders of a and b, adding those remainders together, and then computing the remainder of their sum. In symbols,*

$$(a+b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m.$$

Similarly for the difference $a - b$ and the product ab :

$$(a-b) \bmod m = ((a \bmod m) - (b \bmod m)) \bmod m,$$

$$(ab) \bmod m = ((a \bmod m)(b \bmod m)) \bmod m.$$

Proof We start by using Theorem 2.5.5 to express a as $q_1\, m + r_1$ and b as $q_2\, m + r_2$, where $0 \leq r_1, r_2 < m$. In particular, the proof of this theorem tells us that $a \bmod m = r_1$ and $b \bmod m = r_2$. Then

$$(a+b) \bmod m = (q_1\, m + r_1 + q_2\, m + r_2) \bmod m.$$

We can use Theorem 2.5.4 to reduce the above expression to

$$(r_1 + r_2) \bmod m.$$

The right hand side of the equation in the theorem is equal to

$$((a \bmod m) + (b \bmod m)) \bmod m = (r_1 + r_2) \bmod m$$

The proofs of the equations for the difference $a - b$ and the product ab are very similar and left as an exercise. ■

As an example, let's use Theorem 2.5.7 to compute the remainder $75^6 \bmod 11$ without first computing 75^6. We begin by computing the remainder $75 \bmod 11 = 9$. Then by Theorem 2.5.7,

$$75^2 \bmod 11 = ((75 \bmod 11)(75 \bmod 11)) \bmod 11 = (9)(9) \bmod 11 = 81 \bmod 11 = 4.$$

Again by Theorem 2.5.7 and the values we've already computed, we find that

$$75^3 \bmod 11 = ((75 \bmod 11)(75^2 \bmod 11)) \bmod 11 = (9)(4) \bmod 11 = 36 \bmod 11 = 3.$$

Finally,

$$75^6 \bmod 11 = ((75^3 \bmod 11)(75^3 \bmod 11)) \bmod 11 = (3)(3) \bmod 11 = 9 \bmod 11 = 9.$$

You may wonder how $n \bmod m$ is defined when $m < 0$. The answer is that we seldom find it useful to have that operation defined, so we will not give a definition in this text.

2.5.1 Implementation of the Mod Operator in C and C++

We now consider the problem of using C or C++ to compute remainders of pairs of integers. Suppose that `n` and `m` are program variables of some integer data type, and let n and m denote the mathematical values of those variables. Assume in what follows that $m > 0$. How can we use the program variables `n` and `m` to compute $n \bmod m$, the remainder of n divided by m?

By our mathematical definition, $n \bmod m = n - m\left\lfloor \frac{n}{m} \right\rfloor$. As we noted in Sect. 2.1, if n is positive, the value of the C/C++ expression `n/m` is $\left\lfloor \frac{n}{m} \right\rfloor$, so the C/C++ expression

$$\texttt{n} - \texttt{m} * (\texttt{n/m}) \tag{2.9}$$

can be used as the remainder $n \bmod m$ when $n > 0$. Trivially, this expression also gives the correct value when n is 0.

The C and C++ languages have a "modulus" operator, which is denoted by the symbol `%`. The standards for the C and C++ languages require that the expression `n % m` have exactly the same value as the expression in formula 2.9 for all non-zero values of `m`. Thus `n % m` returns $n \bmod m$ when $n \geq 0$. (Keep in mind that we are assuming $m > 0$.)

When $n < 0$ the expression `n % m` will return $n \bmod m$ if and only if the integer division expression `n/m` returns $\left\lfloor \frac{n}{m} \right\rfloor$. Some C and C++ compilers implement `n/m` that way. However, the majority of C and C++ compilers implement it as the result of the function `truncate` applied to the real number $\frac{n}{m}$. This function eliminates the fractional part of the real number. The result is that `n/m` will return $\left\lfloor \frac{n}{m} \right\rfloor$ when $n \geq 0$ and $\left\lceil \frac{n}{m} \right\rceil$ when $n < 0$. For these compilers the expression `n % m` does *not*

return n mod m when $n < 0$, except in those cases where the fractional part of the real number $\frac{n}{m}$ is 0.

It would be legal for a compiler to implement the integer division expression `n/m` as the value obtained by applying the function `round` to the real number $\frac{n}{m}$ when $n < 0$. With this function, the integer division expression `n/m` will return $\left\lfloor \frac{n}{m} \right\rfloor$ if $\frac{n}{m} - \left\lfloor \frac{n}{m} \right\rfloor > \left\lceil \frac{n}{m} \right\rceil - \frac{n}{m}$, and will return $\left\lceil \frac{n}{m} \right\rceil$ otherwise.

The possibilities for `n/m` described above are the only sensible choices when $n < 0$, because they are the only ones that produce values for `n % m` that lie strictly between $-m$ and m, which is where we expect "remainders" to fall. Note that for each of these possibilities, `n/m` returns either $\left\lfloor \frac{n}{m} \right\rfloor$ or $\left\lceil \frac{n}{m} \right\rceil$.

If the value returned is $\left\lfloor \frac{n}{m} \right\rfloor$ then, as we noted earlier, `n % m` will return n mod m, which by Theorem A.5.3 is non-negative.

If the value returned by `n/m` is not $\left\lfloor \frac{n}{m} \right\rfloor$, then it must be returning $\left\lceil \frac{n}{m} \right\rceil$, in which case Theorem 2.1.4 (f) tells us that $\frac{n}{m}$ is not an integer and

$$\left\lceil \frac{n}{m} \right\rceil = 1 + \left\lfloor \frac{n}{m} \right\rfloor .$$

It follows that `n % m` returns the value

$$n - m \left\lceil \frac{n}{m} \right\rceil = n - m \left(1 + \left\lfloor \frac{n}{m} \right\rfloor\right) = n - m - m \left\lfloor \frac{n}{m} \right\rfloor = n \bmod m - m.$$

From this we can deduce that `n%m + m` will return n mod m. By Theorem 2.5.3 we can also see that the value returned by `n%m` is negative.

The preceding paragraph implies the following: if `n%m` returns a non-negative value, then the value returned by `n/m` must be $\left\lfloor \frac{n}{m} \right\rfloor$, so by the paragraph before that, `n%m` returns n mod m. Similarly, if `n%m` returns a negative value, the value of `n/m` cannot be $\left\lfloor \frac{n}{m} \right\rfloor$, so it must be $\left\lceil \frac{n}{m} \right\rceil$, and thus `n % m + m` returns n mod m. From these observations we can write the following portable function implementing the mathematical mod function.

```
// This function assumes that the parameter m has a strictly positive value.
// It returns n mod m, i.e., the remainder (or residue) of n divided by m,
// which is defined mathematically as n - m * floor(n/m).
// The function has been written so as to be portable among
// C/C++ compilers.
int mod(int n, int m)
{
  if (m <= 0)                     // We do not define n mod m for m <= 0.
    "error; throw an exception"

  int n_percent_m = n % m; // Store it for multiple use

  if (n_percent_m >= 0) // The % operator correctly implements the
    return n_percent_m;           // mathematical mod operator in this case.
  else
    return n_percent_m + m;       // This is when n % m < 0.
}
```

2.5.2 Exercises

2.5.8 Evaluate the following: $3 \bmod 2$; $2 \bmod 3$; $17 \bmod 5$; $34 \bmod 6$; $63 \bmod 7$; $-3 \bmod 2$; $-2 \bmod 3$; $-17 \bmod 5$; $-34 \bmod 6$; $-63 \bmod 7$.

2.5.9 Evaluate $n \bmod 1$ for all integers n. Evaluate $n \bmod 2$ for all integers n. What are the possible values of $n \bmod 100$, where n can be any integer (positive, negative, or zero)?

2.5.10 Prove the identity $(a\,b) \bmod m = ((a \bmod m)(b \bmod m)) \bmod m$ stated in Theorem 2.5.7.

2.5.11 Prove the following variants of the identities in Theorem 2.5.7:

(a) $(a + b) \bmod m = (a + (b \bmod m)) \bmod m$.
(b) $(a\,b) \bmod m = ((a \bmod m)b) \bmod m$.
These identities are examples of the rule that says when we want to calculate the remainder modulo m of any expression involving addition, subtraction, and multiplication, we can apply the "mod" operator at any or all stages of the evaluation process. For example

$$(a - bc + d) \bmod m = ((a \bmod m) - b(c \bmod m) + d) \bmod m.$$

2.5.12 Without using an electronic calculator, compute $21436597^8 \bmod 11$. Show the steps in your computation. Hint: $n^8 = n^4 \times n^4$; $n^4 = n^2 \times n^2$.

2.5.13 Use a similar idea to Example 2.5.6 to prove that for all integers n and m and all positive integers k,

$$\left\lfloor \frac{m}{k} \right\rfloor + \left\lfloor \frac{n-1-m}{k} \right\rfloor \geq \left\lfloor \frac{n}{k} \right\rfloor - 1.$$

[*Hint:* express m in terms of its integer quotient and remainder when divided by k, and substitute that expression for m into the left side of the inequality above. Make use of Theorem 2.1.5 (a).]

2.5.14 Prove that for all integers n, $(-n) \bmod 2 = n \bmod 2$. Give an example to show that it is not always true that $(-n) \bmod 3 = n \bmod 3$.

http://www.springer.com/978-3-319-09887-6

Practical Analysis of Algorithms

Vrajitoru, D.; Knight, W.

2014, XII, 466 p. 245 illus., Softcover
ISBN: 978-3-319-09887-6