

Algorithms and Complexity

Herbert S. Wilf
University of Pennsylvania
Philadelphia, PA 19104-6395

Copyright Notice

Copyright 1994 by Herbert S. Wilf. This material may be reproduced for any educational purpose, multiple copies may be made for classes, etc. Charges, if any, for reproduced copies must be just enough to recover reasonable costs of reproduction. Reproduction for commercial purposes is prohibited. This cover page must be included in all distributed copies.

Internet Edition, Summer, 1994

This edition of Algorithms and Complexity is the file "`pub/wilf/AlgComp.ps.Z`" at the anonymous ftp site "`ftp.cis.upenn.edu`". It may be taken at no charge by all interested persons. Comments and corrections are welcome, and should be sent to `wilf@central.cis.upenn.edu`

Chapter 1: Mathematical Preliminaries

1.1 Orders of magnitude

In this section we're going to discuss the rates of growth of different functions and to introduce the five symbols of asymptotics that are used to describe those rates of growth. In the context of algorithms, the reason for this discussion is that we need a good language for the purpose of comparing the speeds with which different algorithms do the same job, or the amounts of memory that they use, or whatever other measure of the complexity of the algorithm we happen to be using.

Suppose we have a method of inverting square nonsingular matrices. How might we measure its speed? Most commonly we would say something like 'if the matrix is $n \times n$ then the method will run in time $16.8n^3$.' Then we would know that if a 100×100 matrix can be inverted, with this method, in 1 minute of computer time, then a 200×200 matrix would require $2^3 = 8$ times as long, or about 8 minutes. The constant '16.8' wasn't used at all in this example; only the fact that the labor grows as the third power of the matrix size was relevant.

Hence we need a language that will allow us to say that the computing time, as a function of n , grows 'on the order of n^3 ,' or 'at most as fast as n^3 ,' or 'at least as fast as $n^5 \log n$,' etc.

The new symbols that are used in the language of comparing the rates of growth of functions are the following five: 'o' (read 'is little oh of'), 'O' (read 'is big oh of'), 'Θ' (read 'is theta of'), '∼' (read 'is asymptotically equal to' or, irreverently, as 'twiddles'), and 'Ω' (read 'is omega of').

Now let's explain what each of them means.

Let $f(x)$ and $g(x)$ be two functions of x . Each of the five symbols above is intended to compare the rapidity of growth of f and g . If we say that $f(x) = o(g(x))$, then informally we are saying that f grows more slowly than g does when x is very large. Formally, we state the

Definition. We say that $f(x) = o(g(x))$ ($x \rightarrow \infty$) if $\lim_{x \rightarrow \infty} f(x)/g(x)$ exists and is equal to 0.

Here are some examples:

- (a) $x^2 = o(x^5)$
- (b) $\sin x = o(x)$
- (c) $14.709\sqrt{x} = o(x/2 + 7 \cos x)$
- (d) $1/x = o(1)$ (?)
- (e) $23 \log x = o(x^{.02})$

We can see already from these few examples that sometimes it might be easy to prove that a 'o' relationship is true and sometimes it might be rather difficult. Example (e), for instance, requires the use of L'Hospital's rule.

If we have two computer programs, and if one of them inverts $n \times n$ matrices in time $635n^3$ and if the other one does so in time $o(n^{2.8})$ then we know that *for all sufficiently large values of n* the performance guarantee of the second program will be superior to that of the first program. Of course, the first program might run faster on small matrices, say up to size $10,000 \times 10,000$. If a certain program runs in time $n^{2.03}$ and if someone were to produce another program for the same problem that runs in $o(n^2 \log n)$ time, then that second program would be an improvement, at least in the theoretical sense. The reason for the 'theoretical' qualification, once more, is that the second program would be known to be superior only if n were sufficiently large.

The second symbol of the asymptotics vocabulary is the 'O.' When we say that $f(x) = O(g(x))$ we mean, informally, that f certainly doesn't grow at a faster rate than g . It might grow at the same rate or it might grow more slowly; both are possibilities that the 'O' permits. Formally, we have the next

Definition. We say that $f(x) = O(g(x))$ ($x \rightarrow \infty$) if $\exists C, x_0$ such that $|f(x)| < Cg(x)$ ($\forall x > x_0$).

The qualifier ' $x \rightarrow \infty$ ' will usually be omitted, since it will be understood that we will most often be interested in large values of the variables that are involved.

For example, it is certainly true that $\sin x = O(x)$, but even more can be said, namely that $\sin x = O(1)$. Also $x^3 + 5x^2 + 77 \cos x = O(x^5)$ and $1/(1+x^2) = O(1)$. Now we can see how the 'o' gives more precise information than the 'O,' for we can sharpen the last example by saying that $1/(1+x^2) = o(1)$. This is

sharper because not only does it tell us that the function is bounded when x is large, we learn that the function actually approaches 0 as $x \rightarrow \infty$.

This is typical of the relationship between O and o . It often happens that a ‘ O ’ result is sufficient for an application. However, that may not be the case, and we may need the more precise ‘ o ’ estimate.

The third symbol of the language of asymptotics is the ‘ Θ .’

Definition. We say that $f(x) = \Theta(g(x))$ if there are constants $c_1 \neq 0$, $c_2 \neq 0$, x_0 such that for all $x > x_0$ it is true that $c_1g(x) < f(x) < c_2g(x)$.

We might then say that f and g are of the same rate of growth, only the multiplicative constants are uncertain. Some examples of the ‘ Θ ’ at work are

$$\begin{aligned}(x+1)^2 &= \Theta(3x^2) \\ (x^2 + 5x + 7)/(5x^3 + 7x + 2) &= \Theta(1/x) \\ \sqrt{3 + \sqrt{2x}} &= \Theta(x^{\frac{1}{4}}) \\ (1 + 3/x)^x &= \Theta(1).\end{aligned}$$

The ‘ Θ ’ is much more precise than either the ‘ O ’ or the ‘ o .’ If we know that $f(x) = \Theta(x^2)$, then we know that $f(x)/x^2$ stays between two nonzero constants for all sufficiently large values of x . The rate of growth of f is established: it grows quadratically with x .

The most precise of the symbols of asymptotics is the ‘ \sim .’ It tells us that not only do f and g grow at the same rate, but that in fact f/g approaches 1 as $x \rightarrow \infty$.

Definition. We say that $f(x) \sim g(x)$ if $\lim_{x \rightarrow \infty} f(x)/g(x) = 1$.

Here are some examples.

$$\begin{aligned}x^2 + x &\sim x^2 \\ (3x + 1)^4 &\sim 81x^4 \\ \sin 1/x &\sim 1/x \\ (2x^3 + 5x + 7)/(x^2 + 4) &\sim 2x \\ 2^x + 7 \log x + \cos x &\sim 2^x\end{aligned}$$

Observe the importance of getting the multiplicative constants exactly right when the ‘ \sim ’ symbol is used. While it is true that $2x^2 = \Theta(x^2)$, it is not true that $2x^2 \sim x^2$. It is, by the way, also true that $2x^2 = \Theta(17x^2)$, but to make such an assertion is to use bad style since no more information is conveyed with the ‘17’ than without it.

The last symbol in the asymptotic set that we will need is the ‘ Ω .’ In a nutshell, ‘ Ω ’ is the negation of ‘ o .’ That is to say, $f(x) = \Omega(g(x))$ means that it is *not* true that $f(x) = o(g(x))$. In the study of algorithms for computers, the ‘ Ω ’ is used when we want to express the thought that a certain calculation takes *at least* so-and-so long to do. For instance, we can multiply together two $n \times n$ matrices in time $O(n^3)$. Later on in this book we will see how to multiply two matrices even faster, in time $O(n^{2.81})$. People know of even faster ways to do that job, but one thing that we can be sure of is this: nobody will ever be able to write a matrix multiplication program that will multiply pairs $n \times n$ matrices with fewer than n^2 computational steps, because whatever program we write will have to look at the input data, and there are $2n^2$ entries in the input matrices.

Thus, a computing time of cn^2 is certainly a *lower bound* on the speed of any possible general matrix multiplication program. We might say, therefore, that the problem of multiplying two $n \times n$ matrices requires $\Omega(n^2)$ time.

The exact definition of the ‘ Ω ’ that was given above is actually rather delicate. We stated it as the negation of something. Can we rephrase it as a positive assertion? Yes, with a bit of work (see exercises 6 and 7 below). Since ‘ $f = o(g)$ ’ means that $f/g \rightarrow 0$, the symbol $f = \Omega(g)$ means that f/g does not approach zero. If we assume that g takes positive values only, which is usually the case in practice, then to say that f/g does not approach 0 is to say that $\exists \epsilon > 0$ and an infinite sequence of values of x , tending to ∞ , along which $|f/g| > \epsilon$. So we don’t have to show that $|f/g| > \epsilon$ for all large x , but only for *infinitely many* large x .

Definition. We say that $f(x) = \Omega(g(x))$ if there is an $\epsilon > 0$ and a sequence $x_1, x_2, x_3, \dots \rightarrow \infty$ such that $\forall j : |f(x_j)| > \epsilon g(x_j)$.

Now let's introduce a hierarchy of functions according to their rates of growth when x is large. Among commonly occurring functions of x that grow without bound as $x \rightarrow \infty$, perhaps the slowest growing ones are functions like $\log \log x$ or maybe $(\log \log x)^{1.03}$ or things of that sort. It is certainly true that $\log \log x \rightarrow \infty$ as $x \rightarrow \infty$, but it

takes its time about it. When $x = 1,000,000$, for example, $\log \log x$ has the value 2.6.

Just a bit faster growing than the 'snails' above is $\log x$ itself. After all, $\log(1,000,000) = 13.8$. So if we had a computer algorithm that could do n things in time $\log n$ and someone found another method that could do the same job in time $O(\log \log n)$, then the second method, other things being equal, would indeed be an improvement, but n might have to be extremely large before you would notice the improvement.

Next on the scale of rapidity of growth we might mention the powers of x . For instance, think about $x^{.01}$. It grows faster than $\log x$, although you wouldn't believe it if you tried to substitute a few values of x and to compare the answers (see exercise 1 at the end of this section).

How would we *prove* that $x^{.01}$ grows faster than $\log x$? By using L'Hospital's rule.

Example. Consider the limit of $x^{.01}/\log x$ for $x \rightarrow \infty$. As $x \rightarrow \infty$ the ratio assumes the indeterminate form ∞/∞ , and it is therefore a candidate for L'Hospital's rule, which tells us that if we want to find the limit then we can differentiate the numerator, differentiate the denominator, and try again to let $x \rightarrow \infty$. If we do this, then instead of the original ratio, we find the ratio

$$.01x^{-.99}/(1/x) = .01x^{.01}$$

which obviously grows without bound as $x \rightarrow \infty$. Therefore the original ratio $x^{.01}/\log x$ also grows without bound. What we have proved, precisely, is that $\log x = o(x^{.01})$, and therefore in that sense we can say that $x^{.01}$ grows faster than $\log x$. ■

To continue up the scale of rates of growth, we meet x^2 , x , x^{15} , $x^{15} \log^2 x$, etc., and then we encounter functions that grow faster than *every* fixed power of x , just as $\log x$ grows slower than every fixed power of x .

Consider $e^{\log^2 x}$. Since this is the same as $x^{\log x}$ it will obviously grow faster than x^{1000} , in fact it will be larger than x^{1000} as soon as $\log x > 1000$, i.e., as soon as $x > e^{1000}$ (don't hold your breath!).

Hence $e^{\log^2 x}$ is an example of a function that grows faster than every fixed power of x . Another such example is $e^{\sqrt{x}}$ (why?).

Definition. A function that grows faster than x^a , for every constant a , but grows slower than c^x for every constant $c > 1$ is said to be of *moderately exponential growth*. More precisely, $f(x)$ is of moderately exponential growth if for every $a > 0$ we have $f(x) = \Omega(x^a)$ and for every $\epsilon > 0$ we have $f(x) = o((1 + \epsilon)^x)$.

Beyond the range of moderately exponential growth are the functions that grow exponentially fast. Typical of such functions are $(1.03)^x$, 2^x , x^{97x} , and so forth. Formally, we have the

Definition. A function f is of exponential growth if there exists $c > 1$ such that $f(x) = \Omega(c^x)$ and there exists d such that $f(x) = O(d^x)$.

If we clutter up a function of exponential growth with smaller functions then we will not change the fact that it is of exponential growth. Thus $e^{\sqrt{x}+2x}/(x^{49} + 37)$ remains of exponential growth, because e^{2x} is, all by itself, and it resists the efforts of the smaller functions to change its mind.

Beyond the exponentially growing functions there are functions that grow as fast as you might please. Like $n!$, for instance, which grows faster than c^n for every fixed constant c , and like 2^{n^2} , which grows much faster than $n!$. The growth ranges that are of the most concern to computer scientists are 'between' the very slowly, logarithmically growing functions and the functions that are of exponential growth. The reason is simple: if a computer algorithm requires more than an exponential amount of time to do its job, then it will probably not be used, or at any rate it will be used only in highly unusual circumstances. In this book, the algorithms that we will deal with all fall in this range.

Now we have discussed the various symbols of asymptotics that are used to compare the rates of growth of pairs of functions, and we have discussed the pecking order of rapidity of growth, so that we have a small

catalogue of functions that grow slowly, medium-fast, fast, and super-fast. Next let's look at the growth of sums that involve elementary functions, with a view toward discovering the rates at which the sums grow.

Think about this one:

$$\begin{aligned} f(n) &= \sum_{j=0}^n j^2 \\ &= 1^2 + 2^2 + 3^2 + \cdots + n^2. \end{aligned} \tag{1.1.1}$$

Thus, $f(n)$ is the sum of the squares of the first n positive integers. How fast does $f(n)$ grow when n is large?

Notice at once that among the n terms in the sum that defines $f(n)$, the biggest one is the last one, namely n^2 . Since there are n terms in the sum and the biggest one is only n^2 , it is certainly true that $f(n) = O(n^3)$, and even more, that $f(n) \leq n^3$ for all $n \geq 1$.

Suppose we wanted more precise information about the growth of $f(n)$, such as a statement like $f(n) \sim ?$. How might we make such a better estimate?

The best way to begin is to visualize the sum in (1.1.1) as shown in Fig. 1.1.1.

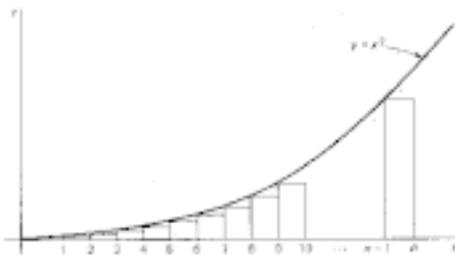


Fig. 1.1.1: How to overestimate a sum

In that figure we see the graph of the curve $y = x^2$, in the x - y plane. Further, there is a rectangle drawn over every interval of unit length in the range from $x = 1$ to $x = n$. The rectangles all lie *under* the curve. Consequently, the total area of all of the rectangles is smaller than the area under the curve, which is to say that

$$\begin{aligned} \sum_{j=1}^{n-1} j^2 &\leq \int_1^n x^2 dx \\ &= (n^3 - 1)/3. \end{aligned} \tag{1.1.2}$$

If we compare (1.1.2) and (1.1.1) we notice that we have proved that $f(n) \leq ((n + 1)^3 - 1)/3$.

Now we're going to get a *lower* bound on $f(n)$ in the same way. This time we use the setup in Fig. 1.1.2, where we again show the curve $y = x^2$, but this time we have drawn the rectangles so they lie *above* the curve.

From the picture we see immediately that

$$\begin{aligned} 1^2 + 2^2 + \cdots + n^2 &\geq \int_0^n x^2 dx \\ &= n^3/3. \end{aligned} \tag{1.1.3}$$

Now our function $f(n)$ has been bounded on both sides, rather tightly. What we know about it is that

$$\forall n \geq 1 : \quad n^3/3 \leq f(n) \leq ((n + 1)^3 - 1)/3.$$

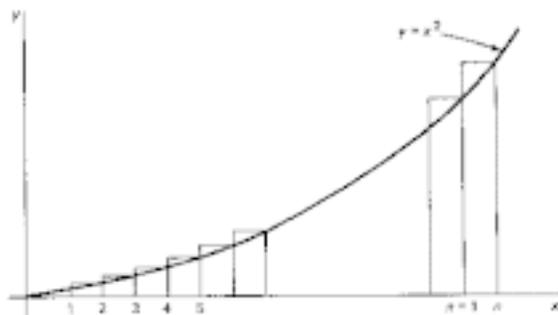


Fig. 1.1.2: How to underestimate a sum

From this we have immediately that $f(n) \sim n^3/3$, which gives us quite a good idea of the rate of growth of $f(n)$ when n is large. The reader will also have noticed that the ‘ \sim ’ gives a much more satisfying estimate of growth than the ‘ O ’ does.

Let’s formulate a general principle, for estimating the size of a sum, that will make estimates like the above for us without requiring us each time to visualize pictures like Figs. 1.1.1 and 1.1.2. The general idea is that when one is faced with estimating the rates of growth of sums, then one should try to compare the sums with integrals because they’re usually easier to deal with.

Let a function $g(n)$ be defined for nonnegative integer values of n , and suppose that $g(n)$ is nondecreasing. We want to estimate the growth of the sum

$$G(n) = \sum_{j=1}^n g(j) \quad (n = 1, 2, \dots). \quad (1.1.4)$$

Consider a diagram that looks exactly like Fig. 1.1.1 except that the curve that is shown there is now the curve $y = g(x)$. The sum of the areas of the rectangles is exactly $G(n-1)$, while the area under the curve between 1 and n is $\int_1^n g(t)dt$. Since the rectangles lie wholly under the curve, their combined areas cannot exceed the area under the curve, and we have the inequality

$$G(n-1) \leq \int_1^n g(t)dt \quad (n \geq 1). \quad (1.1.5)$$

On the other hand, if we consider Fig. 1.1.2, where the graph is once more the graph of $y = g(x)$, the fact that the combined areas of the rectangles is now *not less than* the area under the curve yields the inequality

$$G(n) \geq \int_0^n g(t)dt \quad (n \geq 1). \quad (1.1.6)$$

If we combine (1.1.5) and (1.1.6) we find that we have completed the proof of

Theorem 1.1.1. *Let $g(x)$ be nondecreasing for nonnegative x . Then*

$$\int_0^n g(t)dt \leq \sum_{j=1}^n g(j) \leq \int_1^{n+1} g(t)dt. \quad (1.1.7)$$

The above theorem is capable of producing quite satisfactory estimates with rather little labor, as the following example shows.

Let $g(n) = \log n$ and substitute in (1.1.7). After doing the integrals, we obtain

$$n \log n - n \leq \sum_{j=1}^n \log j \leq (n+1) \log(n+1) - n. \quad (1.1.8)$$

We recognize the middle member above as $\log n!$, and therefore by exponentiation of (1.1.8) we have

$$\left(\frac{n}{e}\right)^n \leq n! \leq \frac{(n+1)^{n+1}}{e^n}. \quad (1.1.9)$$

This is rather a good estimate of the growth of $n!$, since the right member is only about ne times as large as the left member (why?), when n is large.

By the use of slightly more precise machinery one can prove a better estimate of the size of $n!$ that is called *Stirling's formula*, which is the statement that

$$x! \sim \left(\frac{x}{e}\right)^x \sqrt{2x\pi}. \quad (1.1.10)$$

Exercises for section 1.1

1. Calculate the values of $x^{.01}$ and of $\log x$ for $x = 10, 1000, 1,000,000$. Find a single value of $x > 10$ for which $x^{.01} > \log x$, and prove that your answer is correct.

2. Some of the following statements are true and some are false. Which are which?

- (a) $(x^2 + 3x + 1)^3 \sim x^6$
- (b) $(\sqrt{x} + 1)^3 / (x^2 + 1) = o(1)$
- (c) $e^{1/x} = \Theta(1)$
- (d) $1/x \sim 0$
- (e) $x^3(\log \log x)^2 = o(x^3 \log x)$
- (f) $\sqrt{\log x + 1} = \Omega(\log \log x)$
- (g) $\sin x = \Omega(1)$
- (h) $\cos x/x = O(1)$
- (i) $\int_4^x dt/t \sim \log x$
- (j) $\int_0^x e^{-t^2} dt = O(1)$
- (k) $\sum_{j \leq x} 1/j^2 = o(1)$
- (l) $\sum_{j \leq x} 1 \sim x$

3. Each of the three sums below defines a function of x . Beneath each sum there appears a list of five assertions about the rate of growth, as $x \rightarrow \infty$, of the function that the sum defines. In each case state which of the five choices, if any, are true (note: more than one choice may be true).

$$h_1(x) = \sum_{j \leq x} \{1/j + 3/j^2 + 4/j^3\}$$

(i) $\sim \log x$ (ii) $= O(x)$ (iii) $\sim 2 \log x$ (iv) $= \Theta(\log x)$ (v) $= \Omega(1)$

$$h_2(x) = \sum_{j \leq \sqrt{x}} \{\log j + j\}$$

(i) $\sim x/2$ (ii) $= O(\sqrt{x})$ (iii) $= \Theta(\sqrt{x} \log x)$ (iv) $= \Omega(\sqrt{x})$ (v) $= o(\sqrt{x})$

$$h_3(x) = \sum_{j \leq \sqrt{x}} 1/\sqrt{j}$$

(i) $= O(\sqrt{x})$ (ii) $= \Omega(x^{1/4})$ (iii) $= o(x^{1/4})$ (iv) $\sim 2x^{1/4}$ (v) $= \Theta(x^{1/4})$

4. Of the five symbols of asymptotics $O, o, \sim, \Theta, \Omega$, which ones are *transitive* (e.g., if $f = O(g)$ and $g = O(h)$, is $f = O(h)$)?

5. The point of this exercise is that if f grows more slowly than g , then we can always find a third function h whose rate of growth is between that of f and of g . Precisely, prove the following: if $f = o(g)$ then there

is a function h such that $f = o(h)$ and $h = o(g)$. Give an explicit construction for the function h in terms of f and g .

6. {This exercise is a warmup for exercise 7.} Below there appear several mathematical propositions. In each case, write a proposition that is the negation of the given one. Furthermore, in the negation, do not use the word ‘not’ or any negation symbols. In each case the question is, ‘If this *isn’t* true, then what *is* true?’

- (a) $\exists x > 0 \ni f(x) \neq 0$
- (b) $\forall x > 0, f(x) > 0$
- (c) $\forall x > 0, \exists \epsilon > 0 \ni f(x) < \epsilon$
- (d) $\exists x \neq 0 \ni \forall y < 0, f(y) < f(x)$
- (e) $\forall x \exists y \ni \forall z : g(x) < f(y)f(z)$
- (f) $\forall \epsilon > 0 \exists x \ni \forall y > x : f(y) < \epsilon$

Can you formulate a general method for negating such propositions? Given a proposition that contains ‘ \forall ,’ ‘ \exists ,’ ‘ \ni ,’ what rule would you apply in order to negate the proposition and leave the result in positive form (containing no negation symbols or ‘not’s’).

7. In this exercise we will work out the definition of the ‘ Ω .’

- (a) Write out the precise definition of the statement ‘ $\lim_{x \rightarrow \infty} h(x) = 0$ ’ (use ‘ ϵ ’s).
- (b) Write out the negation of your answer to part (a), as a positive assertion.
- (c) Use your answer to part (b) to give a positive definition of the assertion ‘ $f(x) \neq o(g(x))$,’ and thereby justify the definition of the ‘ Ω ’ symbol that was given in the text.

8. Arrange the following functions in increasing order of their rates of growth, for large n . That is, list them so that each one is ‘little oh’ of its successor:

$$2\sqrt{n}, e^{\log n^3}, n^{3.01}, 2^{n^2}, \\ n^{1.6}, \log n^3 + 1, \sqrt{n!}, n^{3 \log n}, \\ n^3 \log n, (\log \log n)^3, n^{.5} 2^n, (n+4)^{12}$$

9. Find a function $f(x)$ such that $f(x) = O(x^{1+\epsilon})$ is true for every $\epsilon > 0$, but for which it is not true that $f(x) = O(x)$.

10. Prove that the statement ‘ $f(n) = O((2+\epsilon)^n)$ for every $\epsilon > 0$ ’ is equivalent to the statement ‘ $f(n) = o((2+\epsilon)^n)$ for every $\epsilon > 0$.’

1.2 Positional number systems

This section will provide a brief review of the representation of numbers in different bases. The usual decimal system represents numbers by using the digits 0, 1, ..., 9. For the purpose of representing whole numbers we can imagine that the powers of 10 are displayed before us like this:

$$\dots, 100000, 10000, 1000, 100, 10, 1.$$

Then, to represent an integer we can specify how many copies of each power of 10 we would like to have. If we write 237, for example, then that means that we want 2 100’s and 3 10’s and 7 1’s.

In general, if we write out the string of digits that represents a number in the decimal system, as $d_m d_{m-1} \dots d_1 d_0$, then the number that is being represented by that string of digits is

$$n = \sum_{i=0}^m d_i 10^i.$$

Now let’s try the *binary system*. Instead of using 10’s we’re going to use 2’s. So we imagine that the powers of 2 are displayed before us, as

$$\dots, 512, 256, 128, 64, 32, 16, 8, 4, 2, 1.$$

To represent a number we will now specify how many copies of each power of 2 we would like to have. For instance, if we write 1101, then we want an 8, a 4 and a 1, so this must be the decimal number 13. We will write

$$(13)_{10} = (1101)_2$$

to mean that the number 13, in the base 10, is the same as the number 1101, in the base 2.

In the binary system (base 2) the only digits we will ever need are 0 and 1. What that means is that if we use only 0's and 1's then we can represent every number n in exactly one way. The unique representation of every number, is, after all, what we must expect and demand of any proposed system.

Let's elaborate on this last point. If we were allowed to use more digits than just 0's and 1's then we would be able to represent the number $(13)_{10}$ as a binary number in a whole lot of ways. For instance, we might make the mistake of allowing digits 0, 1, 2, 3. Then 13 would be representable by $3 \cdot 2^2 + 1 \cdot 2^0$ or by $2 \cdot 2^2 + 2 \cdot 2^1 + 1 \cdot 2^0$ etc.

So if we were to allow too *many* different digits, then numbers would be representable in more than one way by a string of digits.

If we were to allow *too few* different digits then we would find that some numbers have no representation at all. For instance, if we were to use the decimal system with only the digits 0, 1, ..., 8, then infinitely many numbers would not be able to be represented, so we had better keep the 9's.

The general proposition is this.

Theorem 1.2.1. *Let $b > 1$ be a positive integer (the 'base'). Then every positive integer n can be written in one and only one way in the form*

$$n = d_0 + d_1b + d_2b^2 + d_3b^3 + \dots$$

if the digits d_0, d_1, \dots lie in the range $0 \leq d_i \leq b - 1$, for all i .

Remark: The theorem says, for instance, that in the base 10 we need the digits 0, 1, 2, ..., 9, in the base 2 we need only 0 and 1, in the base 16 we need sixteen digits, etc.

Proof of the theorem: If b is fixed, the proof is by induction

on n , the number being represented. Clearly the number 1 can be represented in one and only one way with the available digits (why?). Suppose, inductively, that every integer $1, 2, \dots, n - 1$ is uniquely representable. Now consider the integer n . Define $d = n \bmod b$. Then d is one of the b permissible digits. By induction, the number $n' = (n - d)/b$ is uniquely representable, say

$$\frac{n - d}{b} = d_0 + d_1b + d_2b^2 + \dots$$

Then clearly,

$$\begin{aligned} n &= d + \frac{n - d}{b}b \\ &= d + d_0b + d_1b^2 + d_2b^3 + \dots \end{aligned}$$

is a representation of n that uses only the allowed digits.

Finally, suppose that n has some other representation in this form also. Then we would have

$$\begin{aligned} n &= a_0 + a_1b + a_2b^2 + \dots \\ &= c_0 + c_1b + c_2b^2 + \dots \end{aligned}$$

Since a_0 and c_0 are both equal to $n \bmod b$, they are equal to each other. Hence the number $n' = (n - a_0)/b$ has two different representations, which contradicts the inductive assumption, since we have assumed the truth of the result for all $n' < n$. ■

The bases b that are the most widely used are, aside from 10, 2 ('binary system'), 8 ('octal system') and 16 ('hexadecimal system').

The binary system is extremely simple because it uses only two digits. This is very convenient if you're a computer or a computer designer, because the digits can be determined by some component being either 'on' (digit 1) or 'off' (digit 0). The binary digits of a number are called its *bits* or its *bit string*.

The octal system is popular because it provides a good way to remember and deal with the long bit strings that the binary system creates. According to the theorem, in the octal system the digits that we need are 0, 1, ..., 7. For instance,

$$(735)_8 = (477)_{10}.$$

The captivating feature of the octal system is the ease with which we can convert between octal and binary. If we are given the bit string of an integer n , then to convert it to octal, all we have to do is to group the bits together in groups of three, starting with the least significant bit, then convert each group of three bits, independently of the others, into a single octal digit. Conversely, if the octal form of n is given, then the binary form is obtainable by converting each octal digit independently into the three bits that represent it in the binary system.

For example, given $(1101100101)_2$. To convert this binary number to octal, we group the bits in threes,

$$(1)(101)(100)(101)$$

starting from the right, and then we convert each triple into a single octal digit, thereby getting

$$(1101100101)_2 = (1545)_8.$$

If you're a working programmer it's very handy to use the shorter octal strings to remember, or to write down, the longer binary strings, because of the space saving, coupled with the ease of conversion back and forth.

The hexadecimal system (base 16) is like octal, only more so. The conversion back and forth to binary now uses groups of *four* bits, rather than three. In hexadecimal we will need, according to the theorem above, 16 digits. We have handy names for the first 10 of these, but what shall we call the 'digits 10 through 15'? The names that are conventionally used for them are 'A,' 'B,' ..., 'F.' We have, for example,

$$\begin{aligned} (A52C)_{16} &= 10(4096) + 5(256) + 2(16) + 12 \\ &= (42284)_{10} \\ &= (1010)_2(0101)_2(0010)_2(1100)_2 \\ &= (1010010100101100)_2 \\ &= (1)(010)(010)(100)(101)(100) \\ &= (122454)_8. \end{aligned}$$

Exercises for section 1.2

1. Prove that conversion from octal to binary is correctly done by converting each octal digit to a binary triple and concatenating the resulting triples. Generalize this theorem to other pairs of bases.
2. Carry out the conversions indicated below.
 - (a) $(737)_{10} = (?)_3$
 - (b) $(101100)_2 = (?)_{16}$
 - (c) $(3377)_8 = (?)_{16}$
 - (d) $(ABCD)_{16} = (?)_{10}$
 - (e) $(BEEF)_{16} = (?)_8$
3. Write a procedure *convert* (n, b :integer, *digitstr*:string), that will find the string of digits that represents n in the base b .

1.3 Manipulations with series

In this section we will look at operations with power series, including multiplying them and finding their sums in simple form. We begin with a little catalogue of some power series that are good to know. First we have the finite geometric series

$$(1 - x^n)/(1 - x) = 1 + x + x^2 + \cdots + x^{n-1}. \quad (1.3.1)$$

This equation is valid certainly for all $x \neq 1$, and it remains true when $x = 1$ also if we take the limit indicated on the left side.

Why is (1.3.1) true? Just multiply both sides by $1 - x$ to clear of fractions. The result is

$$\begin{aligned} 1 - x^n &= (1 + x + x^2 + x^3 + \cdots + x^{n-1})(1 - x) \\ &= (1 + x + x^2 + \cdots + x^{n-1}) - (x + x^2 + x^3 + \cdots + x^n) \\ &= 1 - x^n \end{aligned}$$

and the proof is finished.

Now try this one. What is the value of the sum

$$\sum_{j=0}^9 3^j ?$$

Observe that we are looking at the right side of (1.3.1) with $x = 3$. Therefore the answer is $(3^{10} - 1)/2$. Try to get used to the idea that *a series in powers of x becomes a number if x is replaced by a number*, and if we know a formula for the sum of the series then we know the number that it becomes.

Here are some more series to keep in your zoo. A parenthetical remark like ‘ $(|x| < 1)$ ’ shows the set of values of x for which the series converges.

$$\sum_{k=0}^{\infty} x^k = 1/(1 - x) \quad (|x| < 1) \quad (1.3.2)$$

$$e^x = \sum_{m=0}^{\infty} x^m/m! \quad (1.3.3)$$

$$\sin x = \sum_{r=0}^{\infty} (-1)^r x^{2r+1}/(2r+1)! \quad (1.3.4)$$

$$\cos x = \sum_{s=0}^{\infty} (-1)^s x^{2s}/(2s)! \quad (1.3.5)$$

$$\log(1/(1 - x)) = \sum_{j=1}^{\infty} x^j/j \quad (|x| < 1) \quad (1.3.6)$$

Can you find a simple form for the sum (the logarithms are ‘natural’)

$$1 + \log 2 + (\log 2)^2/2! + (\log 2)^3/3! + \cdots ?$$

Hint: Look at (1.3.3), and replace x by $\log 2$.

Aside from merely substituting values of x into known series, there are many other ways of using known series to express sums in simple form. Let’s think about the sum

$$1 + 2 \cdot 2 + 3 \cdot 4 + 4 \cdot 8 + 5 \cdot 16 + \cdots + N2^{N-1}. \quad (1.3.7)$$

We are reminded of the finite geometric series (1.3.1), but (1.3.7) is a little different because of the multipliers $1, 2, 3, 4, \dots, N$.

The trick is this. When confronted with a series that is similar to, but not identical with, a known series, write down the known series as an equation, with the series on one side and its sum on the other. Even though the unknown series involves a particular value of x , in this case $x = 2$, keep the known series with its variable unrestricted. Then reach for an appropriate tool that will be applied to both sides of that equation, and whose result will be that the known series will have been changed into the one whose sum we needed.

In this case, since (1.3.7) reminds us of (1.3.1), we'll begin by writing down (1.3.1) again,

$$(1 - x^n)/(1 - x) = 1 + x + x^2 + \dots + x^{n-1} \quad (1.3.8)$$

Don't replace x by 2 yet, just walk up to the equation (1.3.8) carrying your tool kit and ask what kind of surgery you could do to *both sides of* (1.3.8) that would be helpful in evaluating the unknown (1.3.7).

We are going to reach into our tool kit and pull out ' $\frac{d}{dx}$.' In other words, we are going to *differentiate* (1.3.8). The reason for choosing differentiation is that it will put the missing multipliers $1, 2, 3, \dots, N$ into (1.3.8). After differentiation, (1.3.8) becomes

$$1 + 2x + 3x^2 + 4x^3 + \dots + (n-1)x^{n-2} = \frac{1 - nx^{n-1} + (n-1)x^n}{(1-x)^2}. \quad (1.3.9)$$

Now it's easy. To evaluate the sum (1.3.7), all we have to do is to substitute $x = 2$, $n = N + 1$ in (1.3.9), to obtain, after simplifying the right-hand side,

$$1 + 2 \cdot 2 + 3 \cdot 4 + 4 \cdot 8 + \dots + N2^{N-1} = 1 + (N-1)2^N. \quad (1.3.10)$$

Next try this one:

$$\frac{1}{2 \cdot 3^2} + \frac{1}{3 \cdot 3^3} + \dots \quad (1.3.11)$$

If we rewrite the series using summation signs, it becomes

$$\sum_{j=2}^{\infty} \frac{1}{j \cdot 3^j}.$$

Comparison with the series zoo shows great resemblance to the species (1.3.6). In fact, if we put $x = 1/3$ in (1.3.6) it tells us that

$$\sum_{j=1}^{\infty} \frac{1}{j \cdot 3^j} = \log(3/2). \quad (1.3.12)$$

The desired sum (1.3.11) is the result of dropping the term with $j = 1$ from (1.3.12), which shows that the sum in (1.3.11) is equal to $\log(3/2) - 1/3$.

In general, suppose that $f(x) = \sum a_n x^n$ is some series that we know. Then $\sum n a_n x^{n-1} = f'(x)$ and $\sum n a_n x^n = x f'(x)$. In other words, if the n^{th} coefficient is multiplied by n , then the function changes from f to $(x \frac{d}{dx})f$. If we apply the rule again, we find that multiplying the n^{th} coefficient of a power series by n^2 changes the sum from f to $(x \frac{d}{dx})^2 f$. That is,

$$\begin{aligned} \sum_{j=0}^{\infty} j^2 x^j / j! &= (x \frac{d}{dx})(x \frac{d}{dx})e^x \\ &= (x \frac{d}{dx})(xe^x) \\ &= (x^2 + x)e^x. \end{aligned}$$

Similarly, multiplying the n^{th} coefficient of a power series by n^p will change the sum from $f(x)$ to $(x \frac{d}{dx})^p f(x)$, but that's not all. What happens if we multiply the coefficient of x^n by, say, $3n^2 + 2n + 5$? If the sum previously was $f(x)$, then it will be changed to $\{3(x \frac{d}{dx})^2 + 2(x \frac{d}{dx}) + 5\}f(x)$. The sum

$$\sum_{j=0}^{\infty} (2j^2 + 5)x^j$$

is therefore equal to $\{2(x \frac{d}{dx})^2 + 5\}\{1/(1-x)\}$, and after doing the differentiations we find the answer in the form $(7x^2 - 8x + 5)/(1-x)^3$.

Here is the general rule: if $P(x)$ is any polynomial then

$$\sum_j P(j)a_j x^j = P(x \frac{d}{dx}) \left\{ \sum_j a_j x^j \right\}. \quad (1.3.13)$$

Exercises for section 1.3

- Find simple, explicit formulas for the sums of each of the following series.
 - $\sum_{j \geq 3} \log 6^j / j!$
 - $\sum_{m > 1} (2m + 7) / 5^m$
 - $\sum_{j=0}^{19} (j/2^j)$
 - $1 - x/2! + x^2/4! - x^3/6! + \dots$
 - $1 - 1/3^2 + 1/3^4 - 1/3^6 + \dots$
 - $\sum_{m=2}^{\infty} (m^2 + 3m + 2) / m!$
- Explain why $\sum_{r \geq 0} (-1)^r \pi^{2r+1} / (2r+1)! = 0$.
- Find the coefficient of t^n in the series expansion of each of the following functions about $t = 0$.
 - $(1 + t + t^2)e^t$
 - $(3t - t^2) \sin t$
 - $(t+1)^2 / (t-1)^2$

1.4 Recurrence relations

A recurrence relation is a formula that permits us to compute the members of a sequence one after another, starting with one or more given values.

Here is a small example. Suppose we are to find an infinite sequence of numbers x_0, x_1, \dots by means of

$$x_{n+1} = cx_n \quad (n \geq 0; x_0 = 1). \quad (1.4.1)$$

This relation tells us that $x_1 = cx_0$, and $x_2 = cx_1$, etc., and furthermore that $x_0 = 1$. It is then clear that $x_1 = c$, $x_2 = c^2, \dots, x_n = c^n, \dots$

We say that the *solution* of the recurrence relation (= 'difference equation') (1.4.1) is given by $x_n = c^n$ for all $n \geq 0$. Equation (1.4.1) is a *first-order* recurrence relation because a new value of the sequence is computed from just one preceding value (*i.e.*, x_{n+1} is obtained solely from x_n , and does not involve x_{n-1} or any earlier values).

Observe the format of the equation (1.4.1). The parenthetical remarks are essential. The first one ' $n \geq 0$ ' tells us for what values of n the recurrence formula is valid, and the second one ' $x_0 = 1$ ' gives the starting value. If one of these is missing, the solution may not be uniquely determined. The recurrence relation

$$x_{n+1} = x_n + x_{n-1} \quad (1.4.2)$$

needs two starting values in order to 'get going,' but it is missing both of those starting values and the range of n . Consequently (1.4.2) (which is a second-order recurrence) does not uniquely determine the sequence.

The situation is rather similar to what happens in the theory of ordinary differential equations. There, if we omit initial or boundary values, then the solutions are determined only up to arbitrary constants.

Beyond the simple (1.4.1), the next level of difficulty occurs when we consider a first-order recurrence relation with a variable multiplier, such as

$$x_{n+1} = b_{n+1}x_n \quad (n \geq 0; x_0 \text{ given}). \quad (1.4.3)$$

Now $\{b_1, b_2, \dots\}$ is a given sequence, and we are being asked to find the unknown sequence $\{x_1, x_2, \dots\}$.

In an easy case like this we can write out the first few x 's and then guess the answer. We find, successively, that $x_1 = b_1x_0$, then $x_2 = b_2x_1 = b_2b_1x_0$ and $x_3 = b_3x_2 = b_3b_2b_1x_0$ etc. At this point we can guess that the solution is

$$x_n = \left\{ \prod_{i=1}^n b_i \right\} x_0 \quad (n = 0, 1, 2, \dots). \quad (1.4.4)$$

Since that wasn't hard enough, we'll raise the ante a step further. Suppose we want to solve the first-order *inhomogeneous* (because $x_n = 0$ for all n is not a solution) recurrence relation

$$x_{n+1} = b_{n+1}x_n + c_{n+1} \quad (n \geq 0; x_0 \text{ given}). \quad (1.4.5)$$

Now we are being given two sequences b_1, b_2, \dots and c_1, c_2, \dots , and we want to find the x 's. Suppose we follow the strategy that has so far won the game, that is, writing down the first few x 's and trying to guess the pattern. Then we would find that $x_1 = b_1x_0 + c_1$, $x_2 = b_2b_1x_0 + b_2c_1 + c_2$, and we would probably tire rapidly.

Here is a somewhat more orderly approach to (1.4.5). Though no approach will avoid the unpleasant form of the general answer, the one that we are about to describe at least gives a method that is much simpler than the guessing strategy, for many examples that arise in practice. In this book we are going to run into several equations of the type of (1.4.5), so a unified method will be a definite asset.

The first step is to define a new unknown function as follows. Let

$$x_n = b_1b_2 \cdots b_n y_n \quad (n \geq 1; x_0 = y_0) \quad (1.4.6)$$

define a new unknown sequence y_1, y_2, \dots . Now substitute for x_n in (1.4.5), getting

$$b_1b_2 \cdots b_{n+1}y_{n+1} = b_{n+1}b_1b_2 \cdots b_n y_n + c_{n+1}.$$

We notice that the coefficients of y_{n+1} and of y_n are the same, and so we divide both sides by that coefficient. The result is the equation

$$y_{n+1} = y_n + d_{n+1} \quad (n \geq 0; y_0 \text{ given}) \quad (1.4.7)$$

where we have written $d_{n+1} = c_{n+1}/(b_1 \cdots b_{n+1})$. Notice that the d 's are *known*.

We haven't yet solved the recurrence relation. We have only changed to a new unknown function that satisfies a simpler recurrence (1.4.7). Now the solution of (1.4.7) is quite simple, because it says that each y is obtained from its predecessor by adding the next one of the d 's. It follows that

$$y_n = y_0 + \sum_{j=1}^n d_j \quad (n \geq 0).$$

We can now use (1.4.6) to reverse the change of variables to get back to the original unknowns x_0, x_1, \dots , and find that

$$x_n = (b_1b_2 \cdots b_n) \left\{ x_0 + \sum_{j=1}^n d_j \right\} \quad (n \geq 1). \quad (1.4.8)$$

It is not recommended that the reader memorize the solution that we have just obtained. It *is* recommended that the method by which the solution was found be mastered. It involves

- (a) make a change of variables that leads to a new recurrence of the form (1.4.6), then

- (b) solve that one by summation and
- (c) go back to the original unknowns.

As an example, consider the first-order equation

$$x_{n+1} = 3x_n + n \quad (n \geq 0; x_0 = 0). \quad (1.4.9)$$

The winning change of variable, from (1.4.6), is to let $x_n = 3^n y_n$. After substituting in (1.4.9) and simplifying, we find

$$y_{n+1} = y_n + n/3^{n+1} \quad (n \geq 0; y_0 = 0).$$

Now by summation,

$$y_n = \sum_{j=1}^{n-1} j/3^{j+1} \quad (n \geq 0).$$

Finally, since $x_n = 3^n y_n$ we obtain the solution of (1.4.9) in the form

$$x_n = 3^n \sum_{j=1}^{n-1} j/3^{j+1} \quad (n \geq 0). \quad (1.4.10)$$

This is quite an explicit answer, but the summation can, in fact, be completely removed by the same method that you used to solve exercise 1(c) of section 1.3 (try it!).

That pretty well takes care of first-order recurrence relations of the form $x_{n+1} = b_{n+1}x_n + c_{n+1}$, and it's time to move on to linear second order (homogeneous) recurrence relations with constant coefficients. These are of the form

$$x_{n+1} = ax_n + bx_{n-1} \quad (n \geq 1; x_0 \text{ and } x_1 \text{ given}). \quad (1.4.11)$$

If we think back to *differential* equations of second-order with constant coefficients, we recall that there are always solutions of the form $y(t) = e^{\alpha t}$ where α is constant. Hence the road to the solution of such a differential equation begins by trying a solution of that form and seeing what the constant or constants α turn out to be.

Analogously, equation (1.4.11) calls for a trial solution of the form $x_n = \alpha^n$. If we substitute $x_n = \alpha^n$ in (1.4.11) and cancel a common factor of α^{n-1} we obtain a quadratic equation for α , namely

$$\alpha^2 = a\alpha + b. \quad (1.4.12)$$

'Usually' this quadratic equation will have two distinct roots, say α_+ and α_- , and then the general solution of (1.4.11) will look like

$$x_n = c_1 \alpha_+^n + c_2 \alpha_-^n \quad (n = 0, 1, 2, \dots). \quad (1.4.13)$$

The constants c_1 and c_2 will be determined so that x_0, x_1 have their assigned values.

Example. The recurrence for the Fibonacci numbers is

$$F_{n+1} = F_n + F_{n-1} \quad (n \geq 1; F_0 = F_1 = 1). \quad (1.4.14)$$

Following the recipe that was described above, we look for a solution in the form $F_n = \alpha^n$. After substituting in (1.4.14) and cancelling common factors we find that the quadratic equation for α is, in this case, $\alpha^2 = \alpha + 1$.

If we denote the two roots by $\alpha_+ = (1 + \sqrt{5})/2$ and $\alpha_- = (1 - \sqrt{5})/2$, then the general solution to the Fibonacci recurrence has been obtained, and it has the form (1.4.13). It remains to determine the constants c_1, c_2 from the initial conditions $F_0 = F_1 = 1$.

From the form of the general solution we have $F_0 = 1 = c_1 + c_2$ and $F_1 = 1 = c_1 \alpha_+ + c_2 \alpha_-$. If we solve these two equations in the two unknowns c_1, c_2 we find that $c_1 = \alpha_+/\sqrt{5}$ and $c_2 = -\alpha_-/\sqrt{5}$. Finally, we substitute these values of the constants into the form of the general solution, and obtain an explicit formula for the n^{th} Fibonacci number,

$$F_n = \frac{1}{\sqrt{5}} \left\{ \left(\frac{1 + \sqrt{5}}{2} \right)^{n+1} - \left(\frac{1 - \sqrt{5}}{2} \right)^{n+1} \right\} \quad (n = 0, 1, \dots). \quad (1.4.15)$$

The Fibonacci numbers are in fact 1, 1, 2, 3, 5, 8, 13, 21, 34, ... It isn't even obvious that the formula (1.4.15) gives integer values for the F_n 's. The reader should check that the formula indeed gives the first few F_n 's correctly.

Just to exercise our newly acquired skills in asymptotics, let's observe that since $(1 + \sqrt{5})/2 > 1$ and $|(1 - \sqrt{5})/2| < 1$, it follows that when n is large we have

$$F_n \sim ((1 + \sqrt{5})/2)^{n+1}/\sqrt{5}.$$

■

The process of looking for a solution in a certain form, namely in the form α^n , is subject to the same kind of special treatment, in the case of repeated roots, that we find in differential equations. Corresponding to a *double* root α of the associated quadratic equation $\alpha^2 = a\alpha + b$ we would find two independent solutions α^n and $n\alpha^n$, so the general solution would be in the form $\alpha^n(c_1 + c_2n)$.

Example. Consider the recurrence

$$x_{n+1} = 2x_n - x_{n-1} \quad (n \geq 1; x_0 = 1; x_1 = 5). \quad (1.4.16)$$

If we try a solution of the type $x_n = \alpha^n$, then we find that α satisfies the quadratic equation $\alpha^2 = 2\alpha - 1$. Hence the 'two' roots are 1 and 1. The general solution is $x_n = 1^n(c_1 + nc_2) = c_1 + c_2n$. After inserting the given initial conditions, we find that

$$x_0 = 1 = c_1; \quad x_1 = 5 = c_1 + c_2$$

If we solve for c_1 and c_2 we obtain $c_1 = 1$, $c_2 = 4$, and therefore the complete solution of the recurrence (1.4.16) is given by $x_n = 4n + 1$. ■

Now let's look at recurrent *inequalities*, like this one:

$$x_{n+1} \leq x_n + x_{n-1} + n^2 \quad (n \geq 1; x_0 = 0; x_1 = 0). \quad (1.4.17)$$

The question is, what restriction is placed on the growth of the sequence $\{x_n\}$ by (1.4.17)?

By analogy with the case of difference *equations* with constant coefficients, the thing to try here is $x_n \leq K\alpha^n$. So suppose it is true that $x_n \leq K\alpha^n$ for all $n = 0, 1, 2, \dots, N$. Then from (1.4.17) with $n = N$ we find

$$x_{N+1} \leq K\alpha^N + K\alpha^{N-1} + N^2.$$

Let c be the positive real root of the equation $c^2 = c + 1$, and suppose that $\alpha > c$. Then $\alpha^2 > \alpha + 1$ and so $\alpha^2 - \alpha - 1 = t$, say, where $t > 0$. Hence

$$\begin{aligned} x_{N+1} &\leq K\alpha^{N-1}(1 + \alpha) + N^2 \\ &= K\alpha^{N-1}(\alpha^2 - t) + N^2 \\ &= K\alpha^{N+1} - (tK\alpha^{N-1} - N^2). \end{aligned} \quad (1.4.18)$$

In order to insure that $x_{N+1} < K\alpha^{N+1}$ what we need is for $tK\alpha^{N-1} > N^2$. Hence as long as we choose

$$K > \max_{N \geq 2} \{N^2/t\alpha^{N-1}\}, \quad (1.4.19)$$

in which the right member is clearly finite, the inductive step will go through.

The conclusion is that (1.4.17) implies that for every fixed $\epsilon > 0$, $x_n = O((c + \epsilon)^n)$, where $c = (1 + \sqrt{5})/2$. The same argument applies to the general situation that is expressed in

Theorem 1.4.1. Let a sequence $\{x_n\}$ satisfy a recurrent inequality of the form

$$x_{n+1} \leq b_0x_n + b_1x_{n-1} + \cdots + b_px_{n-p} + G(n) \quad (n \geq p)$$

where $b_i \geq 0$ ($\forall i$), $\sum b_i > 1$. Further, let c be the positive real root of* the equation $c^{p+1} = b_0c^p + \cdots + b_p$. Finally, suppose $G(n) = o(c^n)$. Then for every fixed $\epsilon > 0$ we have $x_n = O((c + \epsilon)^n)$.

Proof: Fix $\epsilon > 0$, and let $\alpha = c + \epsilon$, where c is the root of the equation shown in the statement of the theorem. Since $\alpha > c$, if we let

$$t = \alpha^{p+1} - b_0\alpha^p - \cdots - b_p$$

then $t > 0$. Finally, define

$$K = \max \left\{ |x_0|, \frac{|x_1|}{\alpha}, \dots, \frac{|x_p|}{\alpha^p}, \max_{n \geq p} \left\{ \frac{G(n)}{t\alpha^{n-p}} \right\} \right\}.$$

Then K is finite, and clearly $|x_j| \leq K\alpha^j$ for $j \leq p$. We claim that $|x_n| \leq K\alpha^n$ for all n , which will complete the proof.

Indeed, if the claim is true for $0, 1, 2, \dots, n$, then

$$\begin{aligned} |x_{n+1}| &\leq b_0|x_0| + \cdots + b_p|x_{n-p}| + G(n) \\ &\leq b_0K\alpha^n + \cdots + b_pK\alpha^{n-p} + G(n) \\ &= K\alpha^{n-p}\{b_0\alpha^p + \cdots + b_p\} + G(n) \\ &= K\alpha^{n-p}\{\alpha^{p+1} - t\} + G(n) \\ &= K\alpha^{n+1} - \{tK\alpha^{n-p} - G(n)\} \\ &\leq K\alpha^{n+1} \end{aligned}$$

■

Exercises for section 1.4

1. Solve the following recurrence relations

- (i) $x_{n+1} = x_n + 3 \quad (n \geq 0; x_0 = 1)$
- (ii) $x_{n+1} = x_n/3 + 2 \quad (n \geq 0; x_0 = 0)$
- (iii) $x_{n+1} = 2nx_n + 1 \quad (n \geq 0; x_0 = 0)$
- (iv) $x_{n+1} = ((n+1)/n)x_n + n + 1 \quad (n \geq 1; x_1 = 5)$
- (v) $x_{n+1} = x_n + x_{n-1} \quad (n \geq 1; x_0 = 0; x_1 = 3)$
- (vi) $x_{n+1} = 3x_n - 2x_{n-1} \quad (n \geq 1; x_0 = 1; x_1 = 3)$
- (vii) $x_{n+1} = 4x_n - 4x_{n-1} \quad (n \geq 1; x_0 = 1; x_1 = \xi)$

2. Find x_1 if the sequence \mathbf{x} satisfies the Fibonacci recurrence relation and if furthermore $x_0 = 1$ and $x_n = o(1) \quad (n \rightarrow \infty)$.

3. Let x_n be the average number of trailing 0's in the binary expansions of all integers $0, 1, 2, \dots, 2^n - 1$. Find a recurrence relation satisfied by the sequence $\{x_n\}$, solve it, and evaluate $\lim_{n \rightarrow \infty} x_n$.

4. For what values of a and b is it true that no matter what the initial values x_0, x_1 are, the solution of the recurrence relation $x_{n+1} = ax_n + bx_{n-1} \quad (n \geq 1)$ is guaranteed to be $o(1) \quad (n \rightarrow \infty)$?

5. Suppose $x_0 = 1, x_1 = 1$, and for all $n \geq 2$ it is true that $x_{n+1} \leq x_n + x_{n-1}$. Is it true that $\forall n : x_n \leq F_n$? Prove your answer.

6. Generalize the result of exercise 5, as follows. Suppose $x_0 = y_0$ and $x_1 = y_1$, where $y_{n+1} = ay_n + by_{n-1} \quad (\forall n \geq 1)$. If furthermore, $x_{n+1} \leq ax_n + bx_{n-1} \quad (\forall n \geq 1)$, can we conclude that $\forall n : x_n \leq y_n$? If not, describe conditions on a and b under which that conclusion would follow.

7. Find the asymptotic behavior in the form $x_n \sim ? \quad (n \rightarrow \infty)$ of the right side of (1.4.10).

* See exercise 10, below.

8. Write out a complete proof of theorem 1.4.1.
9. Show by an example that the conclusion of theorem 1.4.1 may be false if the phrase ‘for every fixed $\epsilon > 0 \dots$ ’ were replaced by ‘for every fixed $\epsilon \geq 0 \dots$ ’
10. In theorem 1.4.1 we find the phrase ‘... the positive real root of ...’ Prove that this phrase is justified, in that the equation shown always has exactly one positive real root. Exactly what special properties of that equation did you use in your proof?

1.5 Counting

For a given positive integer n , consider the set $\{1, 2, \dots, n\}$. We will denote this set by the symbol $[n]$, and we want to discuss the number of subsets of various kinds that it has. Here is a list of all of the subsets of $[2]$: $\emptyset, \{1\}, \{2\}, \{1, 2\}$. There are 4 of them.

We claim that the set $[n]$ has exactly 2^n subsets.

To see why, notice that we can construct the subsets of $[n]$ in the following way. Either choose, or don’t choose, the element ‘1,’ then either choose, or don’t choose, the element ‘2,’ etc., finally choosing, or not choosing, the element ‘ n .’ Each of the n choices that you encountered could have been made in either of 2 ways. The totality of n choices, therefore, might have been made in 2^n different ways, so that is the number of subsets that a set of n objects has. ■

Next, suppose we have n distinct objects, and we want to arrange them in a sequence. In how many ways can we do that? For the first object in our sequence we may choose any one of the n objects. The second element of the sequence can be any of the remaining $n - 1$ objects, so there are $n(n - 1)$ possible ways to make the first two decisions. Then there are $n - 2$ choices for the third element, and so we have $n(n - 1)(n - 2)$ ways to arrange the first three elements of the sequence. It is no doubt clear now that there are exactly $n(n - 1)(n - 2) \cdots 3 \cdot 2 \cdot 1 = n!$ ways to form the whole sequence.

Of the 2^n subsets of $[n]$, how many have exactly k objects in them? The number of elements in a set is called its *cardinality*. The cardinality of a set S is denoted by $|S|$, so, for example, $|[6]| = 6$. A set whose cardinality is k is called a ‘ k -set,’ and a subset of cardinality k is, naturally enough, a ‘ k -subset.’ The question is, for how many subsets S of $[n]$ is it true that $|S| = k$?

We can construct k -subsets S of $[n]$ (written ‘ $S \subseteq [n]$ ’) as follows. Choose an element a_1 (n possible choices). Of the remaining $n - 1$ elements, choose one ($n - 1$ possible choices), etc., until a sequence of k different elements have been chosen. Obviously there were $n(n - 1)(n - 2) \cdots (n - k + 1)$ ways in which we might have chosen that sequence, so the number of ways to choose an (ordered) sequence of k elements from $[n]$ is

$$n(n - 1)(n - 2) \cdots (n - k + 1) = n!/(n - k)!$$

But there are more sequences of k elements than there are k -subsets, because any particular k -subset S will correspond to $k!$ different ordered sequences, namely all possible rearrangements of the elements of the subset. Hence the number of k -subsets of $[n]$ is equal to the number of k -sequences divided by $k!$. In other words, there are exactly $n!/k!(n - k)!$ k -subsets of a set of n objects.

The quantities $n!/k!(n - k)!$ are the famous *binomial coefficients*, and they are denoted by

$$\binom{n}{k} = \frac{n!}{k!(n - k)!} \quad (n \geq 0; 0 \leq k \leq n). \quad (1.5.1)$$

Some of their special values are

$$\binom{n}{0} = 1 \quad (\forall n \geq 0); \quad \binom{n}{1} = n \quad (\forall n \geq 0);$$

$$\binom{n}{2} = n(n - 1)/2 \quad (\forall n \geq 0); \quad \binom{n}{n} = 1 \quad (\forall n \geq 0).$$

It is convenient to define $\binom{n}{k}$ to be 0 if $k < 0$ or if $k > n$.

We can summarize the developments so far with

Theorem 1.5.1. For each $n \geq 0$, a set of n objects has exactly 2^n subsets, and of these, exactly $\binom{n}{k}$ have cardinality k ($\forall k = 0, 1, \dots, n$). There are exactly $n!$ different sequences that can be formed from a set of n distinct objects.

Since every subset of $[n]$ has *some* cardinality, it follows that

$$\sum_{k=0}^n \binom{n}{k} = 2^n \quad (n = 0, 1, 2, \dots). \tag{1.5.2}$$

In view of the convention that we adopted, we might have written (1.5.2) as $\sum_k \binom{n}{k} = 2^n$, with no restriction on the range of the summation index k . It would then have been understood that the range of k is from $-\infty$ to ∞ , and that the binomial coefficient $\binom{n}{k}$ vanishes unless $0 \leq k \leq n$.

In Table 1.5.1 we show the values of some of the binomial coefficients $\binom{n}{k}$. The rows of the table are thought of as labelled ‘ $n = 0$,’ ‘ $n = 1$,’ etc., and the entries within each row refer, successively, to $k = 0, 1, \dots, n$. The table is called ‘Pascal’s triangle.’

				1					
				1	1				
			1	2	1				
		1	3	3	1				
	1	4	6	4	1				
1	5	10	10	5	1				
1	6	15	20	15	6	1			
1	7	21	35	35	21	7	1		
1	8	28	56	70	56	28	8	1	
.....									
..									

Table 1.5.1: Pascal’s triangle

Here are some facts about the binomial coefficients:

(a) Each row of Pascal’s triangle is symmetric about the middle. That is,

$$\binom{n}{k} = \binom{n}{n-k} \quad (0 \leq k \leq n; n \geq 0).$$

(b) The sum of the entries in the n^{th} row of Pascal’s triangle is 2^n .

(c) Each entry is equal to the sum of the two entries that are immediately above it in the triangle.

The proof of (c) above can be interesting. What it says about the binomial coefficients is that

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \quad ((n, k) \neq (0, 0)). \tag{1.5.3}$$

There are (at least) two ways to prove (1.5.3). The hammer-and-tongs approach would consist of expanding each of the three binomial coefficients that appears in (1.5.3), using the definition (1.5.1) in terms of factorials, and then cancelling common factors to complete the proof.

That would work (try it), but here’s another way. Contemplate (this proof is by contemplation) the totality of k -subsets of $[n]$. The number of them is on the left side of (1.5.3). Sort them out into two piles: those k -subsets that contain ‘1’ and those that don’t. If a k -subset of $[n]$ contains ‘1,’ then its remaining $k - 1$ elements can be chosen in $\binom{n-1}{k-1}$ ways, and that accounts for the first term on the right of (1.5.3). If a k -subset does not contain ‘1,’ then its k elements are all chosen from $[n - 1]$, and that completes the proof of (1.5.3). ■

The *binomial theorem* is the statement that $\forall n \geq 0$ we have

$$(1+x)^n = \sum_{k=0}^n \binom{n}{k} x^k. \quad (1.5.4)$$

Proof: By induction on n . Eq. (1.5.4) is clearly true when $n = 0$, and if it is true for some n then multiply both sides of (1.5.4) by $(1+x)$ to obtain

$$\begin{aligned} (1+x)^{n+1} &= \sum_k \binom{n}{k} x^k + \sum_k \binom{n}{k} x^{k+1} \\ &= \sum_k \binom{n}{k} x^k + \sum_k \binom{n}{k-1} x^k \\ &= \sum_k \left\{ \binom{n}{k} + \binom{n}{k-1} \right\} x^k \\ &= \sum_k \binom{n+1}{k} x^k \end{aligned}$$

which completes the proof. ■

Now let's ask how big the binomial coefficients are, as an exercise in asymptotics. We will refer to the coefficients in row n of Pascal's triangle, that is, to

$$\binom{n}{0}, \binom{n}{1}, \dots, \binom{n}{n}$$

as the coefficients of *order* n . Then, by (1.5.2) (or by (1.5.4) with $x = 1$), the sum of all of the coefficients of order n is 2^n . It is also fairly apparent, from an inspection of Table 1.5.1, that the largest one(s) of the coefficients of order n is (are) the one(s) in the middle.

More precisely, if n is odd, then the largest coefficients of order n are $\binom{n}{(n-1)/2}$ and $\binom{n}{(n+1)/2}$, whereas if n is even, the largest one is uniquely $\binom{n}{n/2}$.

It will be important, in some of the applications to algorithms later on in this book, for us to be able to pick out the largest term in a sequence of this kind, so let's see how we could *prove* that the biggest coefficients are the ones cited above.

For n fixed, we will compute the ratio of the $(k+1)^{st}$ coefficient of order n to the k^{th} . We will see then that the ratio is larger than 1 if $k < (n-1)/2$ and is < 1 if $k > (n-1)/2$. That, of course, will imply that the $(k+1)^{st}$ coefficient is bigger than the k^{th} , for such k , and therefore that the biggest one(s) must be in the middle.

The ratio is

$$\begin{aligned} \frac{\binom{n}{k+1}}{\binom{n}{k}} &= \frac{n!/\{(k+1)!(n-k-1)!\}}{n!/k!(n-k)!} \\ &= \frac{k!(n-k)!}{(k+1)!(n-k-1)!} \\ &= (n-k)/(k+1) \end{aligned}$$

and is > 1 iff $k < (n-1)/2$, as claimed.

OK, the biggest coefficients are in the middle, but how big are they? Let's suppose that n is even, just to keep things simple. Then the biggest binomial coefficient of order n is

$$\begin{aligned} \binom{n}{n/2} &= \frac{n!}{(n/2)!^2} \\ &\sim \frac{(\frac{n}{e})^n \sqrt{2n\pi}}{\{(\frac{n}{2e})^{\frac{n}{2}} \sqrt{n\pi}\}^2} \\ &= \sqrt{\frac{2}{n\pi}} 2^n \end{aligned} \quad (1.5.5)$$

where we have used Stirling's formula (1.1.10).

Equation (1.5.5) shows that the single biggest binomial coefficient accounts for a very healthy fraction of the sum of *all* of the coefficients of order n . Indeed, the sum of all of them is 2^n , and the biggest one is $\sim \sqrt{2/n\pi}2^n$. When n is large, therefore, the largest coefficient contributes a fraction $\sim \sqrt{2/n\pi}$ of the total.

If we think in terms of the subsets that these coefficients count, what we will see is that a large fraction of all of the subsets of an n -set have cardinality $n/2$, in fact $\Theta(n^{-.5})$ of them do. This kind of probabilistic thinking can be very useful in the design and analysis of algorithms. If we are designing an algorithm that deals with subsets of $[n]$, for instance, we should recognize that a large percentage of the customers for that algorithm will have cardinalities near $n/2$, and make every effort to see that the algorithm is fast for such subsets, even at the expense of possibly slowing it down on subsets whose cardinalities are very small or very large.

Exercises for section 1.5

- How many subsets of even cardinality does $[n]$ have?
- By observing that $(1+x)^a(1+x)^b = (1+x)^{a+b}$, prove that the sum of the squares of all binomial coefficients of order n is $\binom{2n}{n}$.
- Evaluate the following sums in simple form.
 - $\sum_{j=0}^n j \binom{n}{j}$
 - $\sum_{j=3}^n \binom{n}{j} 5^j$
 - $\sum_{j=0}^n (j+1) 3^{j+1}$
- Find, by direct application of Taylor's theorem, the power series expansion of $f(x) = 1/(1-x)^{m+1}$ about the origin. Express the coefficients as certain binomial coefficients.
- Complete the following twiddles.
 - $\binom{2n}{n} \sim ?$
 - $\binom{n}{\lfloor \log_2 n \rfloor} \sim ?$
 - $\binom{n}{\lfloor \theta n \rfloor} \sim ?$
 - $\binom{n^2}{n} \sim ?$
- How many ordered pairs of unequal elements of $[n]$ are there?
- Which one of the numbers $\{2^j \binom{n}{j}\}_{j=0}^n$ is the biggest?

1.6 Graphs

A graph is a collection of *vertices*, certain unordered pairs of which are called its *edges*. To describe a particular graph we first say what its vertices are, and then we say which pairs of vertices are its edges. The set of vertices of a graph G is denoted by $V(G)$, and its set of edges is $E(G)$.

If v and w are vertices of a graph G , and if (v, w) is an edge of G , then we say that vertices v, w are *adjacent* in G .

Consider the graph G whose vertex set is $\{1, 2, 3, 4, 5\}$ and whose edges are the set of pairs $(1,2), (2,3), (3,4), (4,5), (1,5)$. This is a graph of 5 vertices and 5 edges. A nice way to present a graph to an audience is to draw a picture of it, instead of just listing the pairs of vertices that are its edges. To draw a picture of a graph we would first make a point for each vertex, and then we would draw an arc between two vertices v and w if and only if (v, w) is an edge of the graph that we are talking about. The graph G of 5 vertices and 5 edges that we listed above can be drawn as shown in Fig. 1.6.1(a). It could also be drawn as shown in Fig. 1.6.1(b). They're both the same graph. Only the pictures are different, but the pictures aren't 'really' the graph; the graph is the vertex list and the edge list. The pictures are helpful to us in visualizing and remembering the graph, but that's all.

The number of edges that contain ('are incident with') a particular vertex v of a graph G is called the *degree* of that vertex, and is usually denoted by $\rho(v)$. If we add up the degrees of every vertex v of G we will have counted exactly two contributions from each edge of G , one at each of its endpoints. Hence, for every

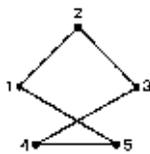


Fig. 1.6.1(a)

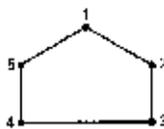


Fig. 1.6.1(b)

graph G we have

$$\sum_{v \in V(G)} \rho(v) = 2|E(G)|. \quad (1.6.1)$$

Since the right-hand side is an even number, there must be an even number of odd numbers on the left side of (1.6.1). We have therefore proved that *every graph has an even number of vertices whose degrees are odd.** In Fig. 1.6.1 the degrees of the vertices are $\{2, 2, 2, 2, 2\}$ and the sum of the degrees is $10 = 2|E(G)|$.

Next we're going to define a number of concepts of graph theory that will be needed in later chapters. A fairly large number of terms will now be defined, in rather a brief space. Don't try to absorb them all now, but read through them and look them over again when the concepts are actually used, in the sequel.

A *path* \mathcal{P} in a graph G is a walk from one vertex of G to another, where at each step the walk uses an edge of the graph. More formally, it is a sequence $\{v_1, v_2, \dots, v_k\}$ of vertices of G such that $\forall i = 1, k-1 : (v_i, v_{i+1}) \in E(G)$.

A graph is *connected* if there is a path between every pair of its vertices.

A path \mathcal{P} is *simple* if its vertices are all distinct, *Hamiltonian* if it is simple and visits every vertex of G exactly once, *Eulerian* if it uses every *edge* of G exactly once.

A *subgraph* of a graph G is a subset S of its vertices together with a subset of just those edges of G both of whose endpoints lie in S . An *induced subgraph* of G is a subset S of the vertices of G together with *all* edges of G both of whose endpoints lie in S . We would then speak of 'the subgraph induced by S .'

In a graph G we can define an equivalence relation on the vertices as follows. Say that v and w are equivalent if there is a path of G that joins them. Let S be one of the equivalence classes of vertices of G under this relation. The subgraph of G that S induces is called a *connected component* of the graph G . A graph is *connected* if and only if it has exactly one connected component.

A *cycle* is a closed path, *i.e.*, one in which $v_k = v_1$. A cycle is a *circuit* if v_1 is the only repeated vertex in it. We may say that a circuit is a simple cycle. We speak of Hamiltonian and Eulerian circuits of G as circuits of G that visit, respectively, every vertex, or every edge, of a graph G .

Not every graph has a Hamiltonian path. The graph in Fig. 1.6.2(a) has one and the graph in Fig. 1.6.2(b) doesn't.

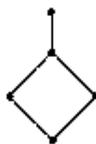


Fig. 1.6.2(a)

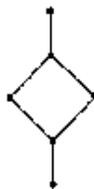


Fig. 1.6.2(b)

* Did you realize that the number of people who shook hands an odd number of times yesterday is an even number of people?

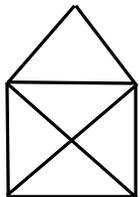


Fig. 1.6.3(a)

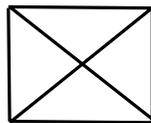


Fig. 1.6.3(b)

Likewise, not every graph has an Eulerian path. The graph in Fig. 1.6.3(a) has one and the graph in Fig. 1.6.3(b) doesn't.

There is a world of difference between Eulerian and Hamiltonian paths, however. If a graph G is given, then thanks to the following elegant theorem of Euler, it is quite easy to decide whether or not G has an Eulerian path. In fact, the theorem applies also to *multigraphs*, which are graphs except that they are allowed to have several different edges joining the same pair of vertices.

Theorem 1.6.1. *A (multi-)graph has an Eulerian circuit (resp. path) if and only if it is connected and has no (resp. has exactly two) vertices of odd degree.*

Proof: Let G be a connected multigraph in which every vertex has even degree. We will find an Eulerian circuit in G . The proof for Eulerian paths will be similar, and is omitted.

The proof is by induction on the number of edges of G , and the result is clearly true if G has just one edge.

Hence suppose the theorem is true for all such multigraphs of fewer than m edges, and let G have m edges. We will construct an Eulerian circuit of G .

Begin at some vertex v and walk along some edge to a vertex w . Generically, having arrived at a vertex u , depart from u along an edge that hasn't been used yet, arriving at a new vertex, etc. The process halts when we arrive for the first time at a vertex v' such that all edges incident with v' have previously been walked on, so there is no exit.

We claim that $v' = v$, *i.e.*, we're back where we started. Indeed, if not, then we arrived at v' one more time than we departed from it, each time using a new edge, and finding no edges remaining at the end. Thus there were an odd number of edges of G incident with v' , a contradiction.

Hence we are indeed back at our starting point when the walk terminates. Let W denote the sequence of edges along which we have so far walked. If W includes all edges of G then we have found an Euler tour and we are finished.

Else there are edges of G that are not in W . Erase all edges of W from G , thereby obtaining a (possibly disconnected multi-) graph G' . Let C_1, \dots, C_k denote the connected components of G' . Each of them has only vertices of even degree because that was true of G and of the walk W that we subtracted from G . Since each of the C_i has fewer edges than G had, there is, by induction, an Eulerian circuit in each of the connected components of G' .

We will thread them all together to make such a circuit for G itself.

Begin at the same v and walk along 0 or more edges of W until you arrive for the first time at a vertex q of component C_1 . This will certainly happen because G is connected. Then follow the Euler tour of the edges of C_1 , which will return you to vertex q . Then continue your momentarily interrupted walk W until you reach for the first time a vertex of C_2 , which will surely happen because G is connected, etc., and the proof is complete. ■

It is extremely difficult computationally to decide if a given graph has a Hamilton path or circuit. We will see in Chapter 5 that this question is typical of a breed of problems that are the main subject of that chapter, and are perhaps the most (in-)famous unsolved problems in theoretical computer science. Thanks to Euler's theorem (theorem 1.6.1) it is *easy* to decide if a graph has an *Eulerian* path or circuit.

Next we'd like to discuss graph *coloring*, surely one of the prettier parts of graph theory. Suppose that there are K colors available to us, and that we are presented with a graph G . A *proper* coloring of the vertices of G is an assignment of a color to each vertex of G in such a way that $\forall e \in E(G)$ the colors of

the two endpoints of e are different. Fig. 1.6.4(a) shows a graph G and an attempt to color its vertices properly in 3 colors ('R,' 'Y' and 'B'). The attempt failed because one of the edges of G has had the same color assigned to both of its endpoints. In Fig. 1.6.4(b) we show the same graph with a successful proper coloring of its vertices in 4 colors.

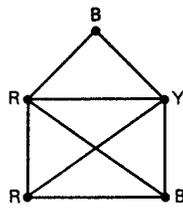


Fig. 1.6.4(a)

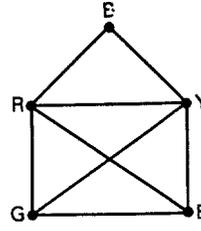


Fig. 1.6.4(b)

The *chromatic number* $\chi(G)$ of a graph G is the minimum number of colors that can be used in a proper coloring of the vertices of G . A *bipartite* graph is a graph whose chromatic number is ≤ 2 , *i.e.*, it is a graph that can be 2-colored. That means that the vertices of a bipartite graph can be divided into two classes 'R' and 'Y' such that no edge of the graph runs between two 'R' vertices or between two 'Y' vertices. Bipartite graphs are most often drawn, as in Fig. 1.6.5, in two layers, with all edges running between layers.

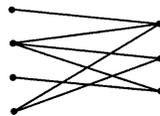


Fig. 1.6.5: A bipartite graph

The *complement* \overline{G} of a graph G is the graph that has the same vertex set that G has and has an edge exactly where G does not have its edges. Formally,

$$E(\overline{G}) = \{(v, w) \mid v, w \in V(G); v \neq w; (v, w) \notin E(G)\}.$$

Here are some special families of graphs that occur so often that they rate special names. The *complete graph* K_n is the graph of n vertices in which every possible one of the $\binom{n}{2}$ edges is actually present. Thus K_2 is a single edge, K_3 looks like a triangle, etc. The *empty graph* \overline{K}_n consists of n isolated vertices, *i.e.*, has no edges at all.

The *complete bipartite graph* $K_{m,n}$ has a set S of m vertices and a set T of n vertices. Its edge set is $E(K_{m,n}) = S \times T$. It has $|E(K_{m,n})| = mn$ edges. The n -cycle, C_n , is a graph of n vertices that are connected to form a single cycle. A *tree* is a graph that (a) is connected and (b) has no cycles. A tree is shown in Fig. 1.6.6.

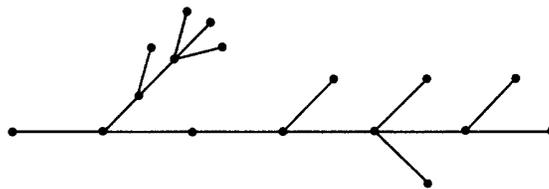


Fig. 1.6.6: A tree

It is not hard to prove that the following are equivalent descriptions of a tree.

- (a) A tree is a graph that is connected and has no cycles.
- (b) A tree is a graph G that is connected and for which $|E(G)| = |V(G)| - 1$.
- (c) A tree is a graph G with the property that between every pair of distinct vertices there is a *unique* path.

If G is a graph and $S \subseteq V(G)$, then S is an *independent set* of vertices of G if no two of the vertices in S are adjacent in G . An independent set S is *maximal* if it is not a proper subset of another independent set of vertices of G . Dually, if a vertex subset S induces a complete graph, then we speak of a *complete subgraph* of G . A maximal complete subgraph of G is called a *clique*.

A graph might be *labeled* or *unlabeled*. The vertices of a labeled graph are numbered $1, 2, \dots, n$. One difference that this makes is that there are a lot more labeled graphs than there are unlabeled graphs. There are, for example, 3 labeled graphs that have 3 vertices and 1 edge. They are shown in Fig. 1.6.7.



Fig. 1.6.7: Three labeled graphs...

There is, however, only 1 unlabeled graph that has 3 vertices and 1 edge, as shown in Fig. 1.6.8.

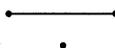


Fig. 1.6.8: ... but only one unlabeled graph

Most counting problems on graphs are much easier for labeled than for unlabeled graphs. Consider the following question: how many graphs are there that have exactly n vertices? Suppose first that we mean *labeled* graphs. A graph of n vertices has a maximum of $\binom{n}{2}$ edges. To construct a graph we would decide which of these possible edges would be used. We can make each of these $\binom{n}{2}$ decisions independently, and for every way of deciding where to put the edges we would get a different graph. Therefore the number of labeled graphs of n vertices is $2^{\binom{n}{2}} = 2^{n(n-1)/2}$.

If we were to ask the corresponding question for unlabeled graphs we would find it to be very hard. The answer is known, but the derivation involves Burnside's lemma about the action of a group on a set, and some fairly delicate counting arguments. We will state the approximate answer to this question, which is easy to write out, rather than the exact answer, which is not. If g_n is the number of unlabeled graphs of n vertices then

$$g_n \sim 2^{\binom{n}{2}}/n!.$$

Exercises for section 1.6

1. Show that a tree is a bipartite graph.
2. Find the chromatic number of the n -cycle.
3. Describe how you would find out, on a computer, if a given graph G is bipartite.
4. Given a positive integer K . Find two different graphs each of whose chromatic numbers is K .
5. Exactly how many labeled graphs of n vertices and E edges are there?
6. In how many labeled graphs of n vertices do vertices $\{1, 2, 3\}$ form an independent set?
7. How many cliques does an n -cycle have?
8. True or false: a Hamilton circuit is an induced cycle in a graph.
9. Which graph of n vertices has the largest number of independent sets? How many does it have?
10. Draw all of the connected, unlabeled graphs of 4 vertices.
11. Let G be a bipartite graph that has q connected components. Show that there are exactly 2^q ways to properly color the vertices of G in 2 colors.
12. Find a graph G of n vertices, other than the complete graph, whose chromatic number is equal to 1 plus the maximum degree of any vertex of G .

13. Let n be a multiple of 3. Consider a labeled graph G that consists of $n/3$ connected components, each of them a K_3 . How many maximal independent sets does G have?
14. Describe the complement of the graph G in exercise 13 above. How many cliques does it have?
15. In how many labeled graphs of n vertices is the subgraph that is induced by vertices $\{1, 2, 3\}$ a triangle?
16. Let H be a labeled graph of L vertices. In how many labeled graphs of n vertices is the subgraph that is induced by vertices $\{1, 2, \dots, L\}$ equal to H ?
17. Devise an algorithm that will decide if a given graph, of n vertices and m edges, does or does not contain a triangle, in time $O(\max(n^2, mn))$.
18. Prove that the number of labeled graphs of n vertices all of whose vertices have *even* degree is equal to the number of all labeled graphs of $n - 1$ vertices.