

**4.2-6**

How quickly can you multiply a  $kn \times n$  matrix by an  $n \times kn$  matrix, using Strassen's algorithm as a subroutine? Answer the same question with the order of the input matrices reversed.

**4.2-7**

Show how to multiply the complex numbers  $a + bi$  and  $c + di$  using only three multiplications of real numbers. The algorithm should take  $a, b, c$ , and  $d$  as input and produce the real component  $ac - bd$  and the imaginary component  $ad + bc$  separately.

---

**4.3 The substitution method for solving recurrences**

Now that we have seen how recurrences characterize the running times of divide-and-conquer algorithms, we will learn how to solve recurrences. We start in this section with the “substitution” method.

The *substitution method* for solving recurrences comprises two steps:

1. Guess the form of the solution.
2. Use mathematical induction to find the constants and show that the solution works.

We substitute the guessed solution for the function when applying the inductive hypothesis to smaller values; hence the name “substitution method.” This method is powerful, but we must be able to guess the form of the answer in order to apply it.

We can use the substitution method to establish either upper or lower bounds on a recurrence. As an example, let us determine an upper bound on the recurrence

$$T(n) = 2T(\lfloor n/2 \rfloor) + n, \quad (4.19)$$

which is similar to recurrences (4.3) and (4.4). We guess that the solution is  $T(n) = O(n \lg n)$ . The substitution method requires us to prove that  $T(n) \leq cn \lg n$  for an appropriate choice of the constant  $c > 0$ . We start by assuming that this bound holds for all positive  $m < n$ , in particular for  $m = \lfloor n/2 \rfloor$ , yielding  $T(\lfloor n/2 \rfloor) \leq c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$ . Substituting into the recurrence yields

$$\begin{aligned} T(n) &\leq 2(c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)) + n \\ &\leq cn \lg(n/2) + n \\ &= cn \lg n - cn \lg 2 + n \\ &= cn \lg n - cn + n \\ &\leq cn \lg n, \end{aligned}$$

where the last step holds as long as  $c \geq 1$ .

Mathematical induction now requires us to show that our solution holds for the boundary conditions. Typically, we do so by showing that the boundary conditions are suitable as base cases for the inductive proof. For the recurrence (4.19), we must show that we can choose the constant  $c$  large enough so that the bound  $T(n) \leq cn \lg n$  works for the boundary conditions as well. This requirement can sometimes lead to problems. Let us assume, for the sake of argument, that  $T(1) = 1$  is the sole boundary condition of the recurrence. Then for  $n = 1$ , the bound  $T(n) \leq cn \lg n$  yields  $T(1) \leq c1 \lg 1 = 0$ , which is at odds with  $T(1) = 1$ . Consequently, the base case of our inductive proof fails to hold.

We can overcome this obstacle in proving an inductive hypothesis for a specific boundary condition with only a little more effort. In the recurrence (4.19), for example, we take advantage of asymptotic notation requiring us only to prove  $T(n) \leq cn \lg n$  for  $n \geq n_0$ , where  $n_0$  is a constant *that we get to choose*. We keep the troublesome boundary condition  $T(1) = 1$ , but remove it from consideration in the inductive proof. We do so by first observing that for  $n > 3$ , the recurrence does not depend directly on  $T(1)$ . Thus, we can replace  $T(1)$  by  $T(2)$  and  $T(3)$  as the base cases in the inductive proof, letting  $n_0 = 2$ . Note that we make a distinction between the base case of the recurrence ( $n = 1$ ) and the base cases of the inductive proof ( $n = 2$  and  $n = 3$ ). With  $T(1) = 1$ , we derive from the recurrence that  $T(2) = 4$  and  $T(3) = 5$ . Now we can complete the inductive proof that  $T(n) \leq cn \lg n$  for some constant  $c \geq 1$  by choosing  $c$  large enough so that  $T(2) \leq c2 \lg 2$  and  $T(3) \leq c3 \lg 3$ . As it turns out, any choice of  $c \geq 2$  suffices for the base cases of  $n = 2$  and  $n = 3$  to hold. For most of the recurrences we shall examine, it is straightforward to extend boundary conditions to make the inductive assumption work for small  $n$ , and we shall not always explicitly work out the details.

### Making a good guess

Unfortunately, there is no general way to guess the correct solutions to recurrences. Guessing a solution takes experience and, occasionally, creativity. Fortunately, though, you can use some heuristics to help you become a good guesser. You can also use recursion trees, which we shall see in Section 4.4, to generate good guesses.

If a recurrence is similar to one you have seen before, then guessing a similar solution is reasonable. As an example, consider the recurrence

$$T(n) = 2T(\lfloor n/2 \rfloor + 17) + n ,$$

which looks difficult because of the added “17” in the argument to  $T$  on the right-hand side. Intuitively, however, this additional term cannot substantially affect the

solution to the recurrence. When  $n$  is large, the difference between  $\lfloor n/2 \rfloor$  and  $\lfloor n/2 \rfloor + 17$  is not that large: both cut  $n$  nearly evenly in half. Consequently, we make the guess that  $T(n) = O(n \lg n)$ , which you can verify as correct by using the substitution method (see Exercise 4.3-6).

Another way to make a good guess is to prove loose upper and lower bounds on the recurrence and then reduce the range of uncertainty. For example, we might start with a lower bound of  $T(n) = \Omega(n)$  for the recurrence (4.19), since we have the term  $n$  in the recurrence, and we can prove an initial upper bound of  $T(n) = O(n^2)$ . Then, we can gradually lower the upper bound and raise the lower bound until we converge on the correct, asymptotically tight solution of  $T(n) = \Theta(n \lg n)$ .

### Subtleties

Sometimes you might correctly guess an asymptotic bound on the solution of a recurrence, but somehow the math fails to work out in the induction. The problem frequently turns out to be that the inductive assumption is not strong enough to prove the detailed bound. If you revise the guess by subtracting a lower-order term when you hit such a snag, the math often goes through.

Consider the recurrence

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1.$$

We guess that the solution is  $T(n) = O(n)$ , and we try to show that  $T(n) \leq cn$  for an appropriate choice of the constant  $c$ . Substituting our guess in the recurrence, we obtain

$$\begin{aligned} T(n) &\leq c \lfloor n/2 \rfloor + c \lceil n/2 \rceil + 1 \\ &= cn + 1, \end{aligned}$$

which does not imply  $T(n) \leq cn$  for any choice of  $c$ . We might be tempted to try a larger guess, say  $T(n) = O(n^2)$ . Although we can make this larger guess work, our original guess of  $T(n) = O(n)$  is correct. In order to show that it is correct, however, we must make a stronger inductive hypothesis.

Intuitively, our guess is nearly right: we are off only by the constant 1, a lower-order term. Nevertheless, mathematical induction does not work unless we prove the exact form of the inductive hypothesis. We overcome our difficulty by *subtracting* a lower-order term from our previous guess. Our new guess is  $T(n) \leq cn - d$ , where  $d \geq 0$  is a constant. We now have

$$\begin{aligned} T(n) &\leq (c \lfloor n/2 \rfloor - d) + (c \lceil n/2 \rceil - d) + 1 \\ &= cn - 2d + 1 \\ &\leq cn - d, \end{aligned}$$

as long as  $d \geq 1$ . As before, we must choose the constant  $c$  large enough to handle the boundary conditions.

You might find the idea of subtracting a lower-order term counterintuitive. After all, if the math does not work out, we should increase our guess, right? Not necessarily! When proving an upper bound by induction, it may actually be more difficult to prove that a weaker upper bound holds, because in order to prove the weaker bound, we must use the same weaker bound inductively in the proof. In our current example, when the recurrence has more than one recursive term, we get to subtract out the lower-order term of the proposed bound once per recursive term. In the above example, we subtracted out the constant  $d$  twice, once for the  $T(\lfloor n/2 \rfloor)$  term and once for the  $T(\lceil n/2 \rceil)$  term. We ended up with the inequality  $T(n) \leq cn - 2d + 1$ , and it was easy to find values of  $d$  to make  $cn - 2d + 1$  be less than or equal to  $cn - d$ .

### Avoiding pitfalls

It is easy to err in the use of asymptotic notation. For example, in the recurrence (4.19) we can falsely “prove”  $T(n) = O(n)$  by guessing  $T(n) \leq cn$  and then arguing

$$\begin{aligned} T(n) &\leq 2(c \lfloor n/2 \rfloor) + n \\ &\leq cn + n \\ &= O(n), \quad \Leftarrow \text{wrong!!} \end{aligned}$$

since  $c$  is a constant. The error is that we have not proved the *exact form* of the inductive hypothesis, that is, that  $T(n) \leq cn$ . We therefore will explicitly prove that  $T(n) \leq cn$  when we want to show that  $T(n) = O(n)$ .

### Changing variables

Sometimes, a little algebraic manipulation can make an unknown recurrence similar to one you have seen before. As an example, consider the recurrence

$$T(n) = 2T(\lfloor \sqrt{n} \rfloor) + \lg n,$$

which looks difficult. We can simplify this recurrence, though, with a change of variables. For convenience, we shall not worry about rounding off values, such as  $\sqrt{n}$ , to be integers. Renaming  $m = \lg n$  yields

$$T(2^m) = 2T(2^{m/2}) + m.$$

We can now rename  $S(m) = T(2^m)$  to produce the new recurrence

$$S(m) = 2S(m/2) + m,$$

which is very much like recurrence (4.19). Indeed, this new recurrence has the same solution:  $S(m) = O(m \lg m)$ . Changing back from  $S(m)$  to  $T(n)$ , we obtain  $T(n) = T(2^m) = S(m) = O(m \lg m) = O(\lg n \lg \lg n)$ .

### Exercises

#### 4.3-1

Show that the solution of  $T(n) = T(n-1) + n$  is  $O(n^2)$ .

#### 4.3-2

Show that the solution of  $T(n) = T(\lceil n/2 \rceil) + 1$  is  $O(\lg n)$ .

#### 4.3-3

We saw that the solution of  $T(n) = 2T(\lfloor n/2 \rfloor) + n$  is  $O(n \lg n)$ . Show that the solution of this recurrence is also  $\Omega(n \lg n)$ . Conclude that the solution is  $\Theta(n \lg n)$ .

#### 4.3-4

Show that by making a different inductive hypothesis, we can overcome the difficulty with the boundary condition  $T(1) = 1$  for recurrence (4.19) without adjusting the boundary conditions for the inductive proof.

#### 4.3-5

Show that  $\Theta(n \lg n)$  is the solution to the “exact” recurrence (4.3) for merge sort.

#### 4.3-6

Show that the solution to  $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$  is  $O(n \lg n)$ .

#### 4.3-7

Using the master method in Section 4.5, you can show that the solution to the recurrence  $T(n) = 4T(n/3) + n$  is  $T(n) = \Theta(n^{\log_3 4})$ . Show that a substitution proof with the assumption  $T(n) \leq cn^{\log_3 4}$  fails. Then show how to subtract off a lower-order term to make a substitution proof work.

#### 4.3-8

Using the master method in Section 4.5, you can show that the solution to the recurrence  $T(n) = 4T(n/2) + n^2$  is  $T(n) = \Theta(n^2)$ . Show that a substitution proof with the assumption  $T(n) \leq cn^2$  fails. Then show how to subtract off a lower-order term to make a substitution proof work.