

Recurrences

When an algorithm contains a recursive call to itself, its running time can often be described by a recurrence. A **recurrence** is an equation or inequality that describes a function in terms of its value on smaller inputs.

For example, the merge sort that we looked at in the first lecture can be expressed as a recurrence:

$$T(n) = \begin{cases} c & \text{if } n = 1 \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 \end{cases}$$

To solve recurrences, we will be focusing on the following methods:

- **Substitution method:** guess a bound and then use mathematical induction to prove our guess correct.
- **Iteration method:** convert the recurrence into a summation and then rely on techniques for bounding summations to solve the recurrence.

1 Substitution Method

There are two parts to the substitution method:

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

For example, let's take the recurrence $T(n) = 2T(\lfloor n/2 \rfloor) + n$, and try to find an upper bound for it. We guess that the solution is $T(n) = O(n \lg n)$. Now we need to prove that $T(n) \leq cn \lg n$ for some $c > 0$. Assuming that this bound holds for $\lfloor n/2 \rfloor$, we substitute into the equation:

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

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

We now need to prove that our solution holds for the boundary conditions. Asymptotic notation only requires us to prove for $n \geq n_0$, hence we can choose $n = 2$ and $n = 3$. Substituting in, it is clear that the solution holds for both of these.

1.1 Issues

There are several issues with the substitution method. For starters, there is no general way to guess the correct solution to a recurrence. Although there are a few methods that help, such as the recursion tree method, it usually boils down to experience and creativity.

- Subtleties?
- Avoiding pitfalls?
- Changing variables?

2 The Iteration Method

The iteration method does not require guessing the answer, but it may require more algebra than the substitution method. The idea is to expand (iterate) the recurrence and express it as a summation of terms dependent only on n and the initial conditions. For example, consider:

$$T(n) = 3T(\lfloor n/4 \rfloor) + n^2$$

