1 Definition

A recurrence is an equation or inequality that describes a function in terms of its own value on smaller inputs.

We’ve already seen one example (for the run time of MergeSort):

\[ T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & \text{otherwise} \end{cases} \]

Recurrences are important because they are the primary tool for analyzing recursive algorithms.

We’ll look at three different ways to solve recurrences.

- Substitution method
- Recursion trees
- The Master theorem

2 Smoothness rule

**Definition**  A non-negative function \( f(n) \) is called smooth if

\[ f(2n) \in \Theta(f(n)). \]

Example: \( f(n) = n^3 \) is a smooth function, because

\[ f(2n) = (2n)^3 = 8n^3 = \Theta(n^3). \]

Example: \( g(n) = 2^n \) is not a smooth function, because

\[ g(2n) = 2^{2n} \neq \Theta(2^n). \]

**Smoothness rule (informally):** Suppose we show, for some \( b \geq 2 \), that \( f(n) = \Theta(g(n)) \) when \( n \) is a power of \( b \). Then, if \( f(n) \) is a smooth function, we have \( f(n) = \Theta(g(n)) \) for all \( n \).

**Smoothness rule (even more informally):** Most of the time, floors and ceilings do not affect the asymptotic growth rate.
3 Substitution method

The substitution method for solving recurrences has two parts.

1. **Guess** the correct answer.
2. **Prove** by induction that your guess is correct.

4 Example

Use the substitution method to solve \( T(n) = 2T(n/2) + n \).

**Guess:** \( T(n) = O(n \lg n) \)

**Proof:** Use induction on \( n \) to show that there exist \( c \) and \( n_0 \) for which \( T(n) \leq cn \lg n \) for all \( n \geq n_0 \).

- **Base case:** Almost always omitted, because \( T(n) = \Theta(1) \) when \( n \) is sufficiently small, so we can always choose \( c \) large enough.

  Details: CLRS 84

- **Induction step:** Assume that \( T(m) \leq cm \lg m \) for all \( m < n \), to prove that \( T(n) \leq cn \lg n \).

(We’ll find restrictions on \( c \) and \( n_0 \) along the way.)

\[
\begin{align*}
T(n) &= 2T(n/2) + n \\
&\leq 2 \cdot c(n/2) \lg(n/2) + n \\
&= cn \lg(n/2) + n \\
&= cn \lg n - cn \lg 2 + n \\
&\leq cn \lg n \quad \text{[when } c \geq 1]\end{align*}
\]

**In the last step, we replace \(-cn + n\) with 0. This increases the sum if \(-cn + n \leq 0\).**

**Solve for \( c \) to get the constraint \( c \geq 1 \).**

5 Be careful!

Substitution proofs must ensure that they use the same constant as in the inductive hypothesis.

Here’s an example of how to “prove” (incorrectly!) that \( T(n) = O(n) \).

\[
\begin{align*}
T(n) &= 2T(n/2) + n \\
&\leq 2c(n/2) + n \\
&= (c + 1)n \\
&= O(n)
\end{align*}
\]
The problem is that we have not proved the exact form of the inductive hypothesis. In particular, the constant \( c \) we use when we substitute at the beginning must be the same \( c \) we have at the end of the inequalities.

### 6 Even correct guesses can lead to dead ends

Show that \( T(n) = 5T(n/2) + n^2 \) is \( O(n^{\log_2 5}) \).

**Attempt 1:** Use induction to (try to) show that, for some \( c \),

\[
T(n) \leq cn^{\log_2 5}.
\]

"Proof:"

\[
T(n) = 5T(n/2) + n^2 \\
\leq 5 \left( c \left( \frac{n}{2} \right)^{\log_2 5} \right) + n^2 \\
= 5 \left( c \frac{n^{\log_2 5}}{2^{\log_2 5}} \right) + n^2 \\
= cn^{\log_2 5} + n^2
\]

A dead end! No choice of \( c \) makes this inequality true.

### 7 Proving a stronger bound

Show that \( T(n) = 5T(n/2) + n^2 \) is \( O(n^{\log_2 5}) \).

**Attempt 2:** Use induction to show that, for some \( c \) and some \( a \),

\[
T(n) \leq cn^{\log_2 5} - an^2.
\]

**Note:** If we can show that \( T(n) \leq cn^{\log_2 5} - an^2 \) for some positive constant \( a \), then we know immediately that \( T(n) \leq cn^{\log_2 5} \), which is sufficient to show that \( T(n) \in O(n^{\log_2 5}) \). So this strong bound really is doing the job that we need it to.

The choice of adding \(-an^2\) here is based on the dead end from the previous slide — we had an extra \( n^2 \) term, and we’re hoping that the new term in the inductive hypothesis will counteract that. This doesn’t always work out, but it often does, and it’s a good place to start.
Proof:

\[
T(n) = 5T(n/2) + n^2 \\
\leq 5 \left( c \left( \frac{n}{2} \right)^{\log_2 5} - a \left( \frac{n}{2} \right)^2 \right) + n^2 \\
= 5 \left( \frac{cn^{\log_2 5}}{2^{\log_2 5}} - \frac{n^2 a}{4} \right) + n^2 \\
= cn^{\log_2 5} + \left( 1 - \frac{5a}{4} \right)n^2 \\
\leq cn^{\log_2 5} - an^2 \quad [a \geq 4]
\]

This is enough to conclude \( T(n) = O(n^{\log_2 5}) \), because \( cn^{\log_2 5} - 4n^2 \leq cn^{\log_2 5} \).

In the last step of the proof, we replace \( 1 - \frac{5a}{4} \) with \(-a\). We want \(-a\), because we need to match the exact form of the inductive hypothesis.

This replacement increases the expression (or leaves it unchanged), making the \( \leq \) we write there correct, when

\[
\left( 1 - \frac{5a}{4} \right) \leq -a.
\]

Solving for \( a \) (first multiply both sides by 4, then add \( 5a \) to both sides) we get:

\[
4 - 5a \leq -4a \\
4 \leq a
\]

Hence the constraint \( a \geq 4 \).

8 Change of variables

Solve the recurrence \( T(n) = 2T(\sqrt{n}) + \lg n \).

Solution: Change of variables. Let \( m = \lg n \) and \( S(m) = T(2^m) \).

Note that \( n = 2^m \) and \( \sqrt{n} = 2^{m/2} \).

Then we get:

\[
T(n) = T(2^m) \\
= 2T(2^{m/2}) + \lg (2^m) \\
= 2S(m/2) + m \\
= O(m \log m) \\
= O(\log n \log \log n)
\]
Recall that we've solved the recurrence $S(m) = 2S(m/2) + n$ already, a few slides back.

9 Recursion trees
We can solve many recurrences by drawing a recursion tree.

- **Nodes**: Label with the contribution to the total for that ‘recursive call’.

  …not counting what happens inside children.

- **Children**: One for each appearance of a recurrent term.

After drawing such a tree, we can solve the recurrence:

1. Compute (or bound) the depth of the leaves.
2. Compute (or bound) the sum for each level.
3. Compute (or bound) the sum across all levels.

10 Example: Mergesort recurrence

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & \text{otherwise} \end{cases}$$

![Recursion Tree Diagram]

- Depth of the leaves: $\lg n$
- Sum for each level: $cn$
- Sum across all levels: $cn \lg n = \Theta(n \log n)$. 

11 Example: Mergesort recurrence
12 Example: Another divide-and-conquer recurrence

\[ T(n) = \begin{cases} 
\Theta(1) & \text{if } n \leq 1 \\
3T(\frac{n}{4}) + \Theta(n^2) & \text{otherwise}
\end{cases} \]

\[ T(n) = \frac{n}{4} T\left(\frac{n}{4}\right) + \frac{n}{4} T\left(\frac{n}{4}\right) + \Theta(n^2) \]

\[ T(n) = \Theta(1) \cdot \cdots \cdot \cdots \cdot \Theta(1) \]

13 Example continued

- Depth of the leaves: \( \log_4 n \)
- Sum for each level: \( 3^i c(n/4^i)^2 = (3/16)^i cn^2 \).
- Sum across all levels:

\[
T(n) = \sum_{i=0}^{\log_4 n} \left( \frac{3}{16} \right)^i cn^2 \\
\leq \sum_{i=0}^{\infty} \left( \frac{3}{16} \right)^i cn^2 \\
= \frac{1}{1 - (3/16)} cn^2 \\
= \frac{16}{13} cn^2 \\
= O(n^2)
\]

Note also: \( T(n) = \Omega(n^2) \). (Why?)

At depth \( i \), the ‘problem size’ is \( n/2^i \). To get down to the base case, we need this value to be 1 or less.

14 A lopsided tree

\[ T(n) = T(n/3) + T(2n/3) + O(n) \]

\[ c(n) \]

\[ c(n/3) \quad \text{and} \quad c(2n/3) \]

\[ c(n/9) \quad c(2n/9) \quad c(2n/9) \quad c(4n/9) \]

\[
\text{\textcopyright 2005 \hspace{1cm} Department of Computer Science \hspace{1cm} University of Maryland}
\]
• Depth of the (deepest) leaves: \( \log_{3/2} n \)
• Sum for each level: \( \leq cn \)
• Sum across all levels: \( cn \log_{3/2} n = O(n \log n) \).

15 Some branches terminate before others
Note that this recurrence does not produce a complete tree!

For \( n = 6 \) (assuming \( \lfloor n/3 \rfloor \) and \( \lfloor 2n/3 \rfloor \)):

```
6c
 /   \
|     |
2c   4c
 |     |
θ(1) θ(1) θ(1) 2c
 |         |
θ(1) θ(1)
```

Therefore, the sum from the previous slide gives an upper bound. We could also get a lower bound by truncating the tree at the level of its shallowest leaves.

16 Master theorem: Simple version
Theorem: Consider the recurrence

\[
T(n) = aT(n/b) + \Theta(n^d).
\]

If \( a > b^d \) then \( T(n) = \Theta(n^{\log_b a}) \).
If \( a = b^d \) then \( T(n) = \Theta(n^d \log n) \).
If \( a < b^d \) then \( T(n) = \Theta(n^d) \).

For this simple version, the final added part must be a polynomial.

17 Master theorem: Real version
Theorem: Consider the recurrence

\[
T(n) = aT(n/b) + f(n).
\]

1. If there exists \( \epsilon > 0 \), for which \( f(n) = O(n^{\log_b a - \epsilon}) \), then \( T(n) = \Theta(n^{\log_b a}) \).
2. If \( f(n) = \Theta(n^{\log_b a}) \), then \( T(n) = \Theta(n^{\log_b a \log n}) \).
3. If there exists \( \epsilon > 0 \), for which \( f(n) = \Omega(n^{\log_b a + \epsilon}) \), and \( af(n/b) \leq cf(n) \) for some constant \( c \) and sufficiently large \( n \), then \( T(n) = \Theta(f(n)) \).
18 Example 1

\[ T(n) = 9T(n/3) + n \]

We have \( a = 9, b = 3, \) and \( f(n) = n. \)

Compare \( n^{\log_3 9} = n^2 \) to \( n. \) Observe that \( n = O(n^{2-\epsilon}) \), with \( \epsilon = 1. \)

Therefore, the first case applies, and \( T(n) = \Theta(n^{\log_3 a}) = \Theta(n^2). \)

19 Example 2

\[ T(n) = T(2n/3) + 1 \]

We have \( a = 1, b = 3/2, \) and \( f(n) = 1. \)

Compare \( n^{\log_{3/2} 1} = 1 \) to \( 1. \) Observe that \( 1 = \Theta(1). \)

Therefore, the second case applies, and \( T(n) = \Theta(n^{\log_3 a} \log n) = \Theta(\log n). \)

20 Example 3

\[ T(n) = 3T(n/4) + n \log n \]

We have \( a = 3, b = 4, \) and \( f(n) = n \log n. \)

Compare \( n^{\log_4 3} \) to \( n \log n. \) Observe that \( n \log n = \Omega(n^{\log_4 3+\epsilon}) \), as long as \( \log_4 3 + \epsilon \leq 1. \) (For example, choose \( \epsilon = 0.2. \))

The “regularity condition” also holds.

Therefore, the third case applies, and \( T(n) = \Theta(n \log n). \)

21 Example 4

\[ T(n) = 2T(n/2) + n \log n \]

We have \( a = 2, b = 2, \) and \( f(n) = n \log n. \)

Compare \( n^{\log_2 2} = n \) to \( n \log n. \) Observe that, although \( n \log n = \Omega(n), \) for any \( \epsilon > 0, n \log n \neq \Omega(n^{1+\epsilon}). \)

(Intuition: Even for a very small \( \epsilon, n^{1+\epsilon} \) will eventually grow faster than \( n \log n. \))

Therefore, the Master theorem does not apply.