Recurrence tree can get complicated, so want simpler method. Often can solve recurrence just by looking at the recurrence.

**Master's Theorem/Method**
- Simplified version (book has more complex).
- Uses a formula.

**Pros:** just apply formula
- Always gives $\Theta$

**Cons:** requires knowing formula/rules
- Applies only to recurrences of form: $T(n) = aT(n/b) + cn^d$ for $c > 0$.

Based on $a$, $b$, $c$, can solve the recurrence.
Look at previous examples to find patterns.

<table>
<thead>
<tr>
<th>Rec (1)</th>
<th>Rec (2)</th>
<th>Rec (3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T(n) = 2T(n/2) + cn$</td>
<td>$T(n) = 2T(n/2) + cn^2$</td>
<td>$T(n) = 4T(n/2) + cn$</td>
</tr>
</tbody>
</table>

**Time**

<table>
<thead>
<tr>
<th>Time</th>
<th>Time</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$cn$</td>
<td>$cn^2$</td>
<td>$cn$</td>
</tr>
<tr>
<td>$cn$</td>
<td>$cn^2$</td>
<td>$cn$</td>
</tr>
<tr>
<td>$\frac{cn}{2}$</td>
<td>$\frac{cn}{2}$</td>
<td>$\frac{cn}{2}$</td>
</tr>
<tr>
<td>$\frac{cn}{4}$</td>
<td>$\frac{cn}{4}$</td>
<td>$\frac{cn}{4}$</td>
</tr>
</tbody>
</table>

Time equal at every level. Time dominated by root leaves.
### Cases

1. Property
   - Time equal at every level
     - \( a = b^i \)
     - \( \approx \) time per level \( \times \) # levels
     - \( (n^i) \times \log_b n = \Theta(n^i \log_b n) \)

2. Time dominated by root
   - \( a < b^i \)
   - \( \approx \) time at root
   - \( \Theta(n^i) \)

3. Leaves
   - \( a > b^i \)
   - \( \approx \) time at leaves \( \approx \) # leaves
   - \( \Theta(n^{\log_b n}) \)

### Notes
- Which property holds depends on \( a, b, i \)
- (Won't prove but knowing what \( a, b, i \) are gives some intuition)

- \( a \) : branching factor
- \( b \) : reduction in input size from previous level
- \( i \) : \( \approx \) time at root

### Examples

1. \( T(n) = 4T(n^{1/2}) + n^3 \)
   - \( a = 4 \), \( b = 2 \), \( i = 3 \)
   - \( a < b^i \) \( \Rightarrow \) Case (2)
   - \( T(n) = \Theta(n^3) \)

2. \( T(n) = T(n^{1/3}) + T(2n^{1/3}) + n \Rightarrow \) Master's does not apply
   - Can solve with recurrence tree or last method: Substitution
**Substitution Method**

Motivate a bit.

To show \( T(n) = o(f(n)) \), must show: \( \exists \ c > 0 \)

\[ T(n) \leq c \cdot f(n) \quad \text{for all } n > 1 \quad (\text{or } 2, 3, \text{ i.e. smallest } n \text{ with recursion}). \]

Which mathematical proof technique can we use?

**Proof by induction**

1. Guess the solution (will start to recognize properties, get a sense from recurrence tree, for exams - will be given)

2. Use main induction to find \( c > 0 \) that satisfies \( T(n) \leq c \cdot f(n) \)

(a) Show true for Base Case (smallest \( n \) with recursion)

(b) Assume \( T(k) \leq c \cdot f(k) \) true for all values up to some \( k < n \).

Express \( k \) in terms of \( n \) (Substitution Step)

(c) Use recurrence and (b) to show true for all \( n \)

Start with a simple example.

\[
T(n) = \begin{cases} 
  d & \text{ (not to be confused with } c) \quad \text{for } n = 1 \\
  2T\left( \frac{n}{2} \right) & \text{for } n > 2
\end{cases}
\]

**Ex:**

1. Guess?

\( T(n) = o(n), f(n) = n \)

**Notes**

\[
a = 2 \quad b = 2 \quad c = 0 \\
\text{or } n = 10^3 \]
2. (a) Base Case: \( n = 2 \)

Find a \( c > 0 \) (we get to choose \( c! \)) s.t.

\[
T(n) \leq c \cdot f(n)
\]

\[
T(2) \leq c \cdot f(2)
\]

\[
2T(\frac{n}{2}) \leq 2c \quad \text{(from recurrence)}
\]

\[
d \leq 2c
\]

\[
c \geq d
\]

\[\rightarrow\] Any \( c \geq d \) works for B.C.

(b) Assume \( T(k) \leq c \cdot f(k) \) for some \( k < n \).

Set \( k = \frac{n}{2} \) (convenient, see why later)

\[
T(\frac{n}{2}) \leq c \cdot f(\frac{n}{2})
\]

\[
T(\frac{n}{2}) \leq \frac{c \cdot n}{2}
\]

(c) To show true for all \( n \): (use recurrence + (b))

\[
T(n) = 2T(\frac{n}{2}) \quad \text{(from recurrence)}
\]

\[
T(n) \leq 2 \cdot \frac{c \cdot n}{2} \quad \text{(from Ind. Hyp. in (b))}
\]

\[
T(n) \leq \frac{c \cdot n}{2}
\]

\[\rightarrow\] Any \( c > 0 \) works here!

Final \( c \) is any \( c \geq \left( \max \left( d, \frac{c}{2} \right) \right) = d \)
\[ T(n) = \begin{cases} d & \text{for } n = 1, 2 \\ 3T(\frac{n}{3}) + n & \text{for } n \geq 3 \end{cases} \]

1. **Guess?** (Similar to mergesort) \( \Rightarrow \Theta(n \log n) \)
   \( \Rightarrow f(n) = n \log n \).

2. **(a) Base Case** Find \( c > 0 \) s.t. \( T(3) \leq c f(3) \)
   
   \[ 3T(\frac{3}{3}) + 3 \leq c (3 \log 3) \]
   
   \[ 3T(1) + 3 \leq c (3 \log 3) \]
   
   \[ \frac{3d + 3}{\log 3} \leq c \]

   \[
   \text{for any } c > \frac{d+1}{\log 3}, \text{ works for } B+C.
   \]

   **(b) Assume** true for up to some \( k < n \)

   \[ T(k) \leq c \cdot f(k) \]

   \[ T(k) \leq c \cdot k \log k \cdot \]

   **Set** \( k = \frac{n}{3} \).

   **(From recurrence)** \( T(\frac{n}{3}) \leq c (\frac{n}{3}) \log (\frac{n}{3}) \).

   **(c) Use recurrence + (b) to show true for all \( n \)**

   \[
   \begin{align*}
   \text{rec:} & \quad T(n) = 3T(\frac{n}{3}) + n \\
   \text{(b):} & \quad \leq 3c (\frac{n}{3}) \log (\frac{n}{3}) + n
   \end{align*}
   \]

   \(<\text{Need to find } c \text{ s.t. } \leq \Theta(n)?>\]
Let's simplify!

\[ T(n) \leq \frac{2e}{\log^2} \log \left( \frac{n}{3} \right) + n \]

\[ T(n) \leq c n \left[ \log n - \log \frac{n}{3} \right] + n \]

\[ T(n) \leq c n \log n - c n \log 3 + n \]

When is this \( \leq c n \log n \)? (For what \( c \)?)

\( c n \log n - c n \log 3 + n \leq c n \log n \)

\[ \frac{c n \log 3}{n} \geq x \]

\[ c \geq \frac{1}{\log 3} \] works.

Our overall \( c \)?

Base Case: \( d \)

Inductive Case:

\[ c \geq \frac{d+1}{\log 3} \]

\[ c \geq \frac{1}{\log 3} \]

This is larger, so any \( c > \frac{d+1}{\log 3} \) works.

Question: what if we guess incorrectly?

Inductive proof will break.

Let's try incorrect guesses for \( c \) (1)