Lecture: Analysis of Algorithms (CS483 - 001)

Amarda Shehu

Spring 2017
Outline of Today’s Class

Techniques for Bounding Recurrences

- Iteration Method
- Recursion-tree Method
- Substitution Method
- Master Theorem
Techniques for Bounding Recurrences

What is a Recurrence?

- A recurrence is an equation of inequality that describes a function in terms of its value on smaller inputs
  - Example: $T(n)$ of Mergesort is described in terms of $T(n/2)$
- Recurrences have boundary conditions (bottom out)
  - Example: $T(n) = c$ when $n = 1$

Techniques for Bounding Recurrences

1. Iteration or expansion method
2. Recursion-tree method
3. Substitution method
4. Master Theorem
5. Generating Functions* (beyond scope of this course)
Expand $T(n) = 2T(n/2) + cn$ – iterate down to boundary condition

\[
T(n) = 2T(n/2) + cn = 2 \cdot [2T(n/4) + c\frac{n}{2}] + cn = 4 \cdot [2T(n/8) + c\frac{n}{4}] + 2cn = 8 \cdot T(n/8) + 3cn = 2^3 \cdot T(n/2^3) + 3cn = \ldots \text{do you see the pattern?} = 2^k \cdot T(n/2^k) + kcn
\]

Since the recursion bottoms out at $n = 1$, $k = \lg(n)$. So:

\[
T(n) = n \cdot T(1) + \lg(n) \cdot cn = cn + cn \cdot \lg(n) \in \theta(n \cdot \lg n)
\]

Try to solve $T(n) = T(n - 1) + n$, where $T(1) = 1$.

Try to solve $T(n) = 2T(n/2) + n$, where $T(1) = 1$. 
Recursion-tree Method

Build recursion tree for $T(n) = 2T(n/2) + c \cdot n$:

```
      c \cdot n
     /  \  
   c \cdot n/2  c \cdot n/2
 /     \      /     \    /     \   /     \  
 c \cdot n/4  c \cdot n/4  c \cdot n/4  c \cdot n/4  c \cdot n/4  c \cdot n/4
```

Total: $O(n \log n)$
Example of Recursion-tree Method

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

\[
\begin{array}{c}
\vdots \\
\Theta(1) \\
(\frac{n}{8})^2 \\
(\frac{n}{4})^2 \\
(\frac{n}{16})^2 \\
n^2 \\
(\frac{n}{2})^2 \\
(\frac{n}{8})^2 \\
(\frac{n}{4})^2 \\
\frac{5}{16} n^2 \\
\frac{25}{256} n^2 \\
\vdots \\
\text{Total} = n^2 \left( 1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \cdots \right) \\
= \Theta(n^2)
\end{array}
\]
Guess that \( T(n) = 2T\left(\frac{n}{2}\right) + n \in O(n + n \cdot \lg n) \), where \( T(1) = 1 \). Then use induction to prove that the guess is correct.

1. **Base Case:** The boundary condition states that \( T(1) = 1 \). The guess states that \( T(1) \in O(1 + 1 \cdot \lg 1) \). Since, \( 1 + 1 \cdot \lg 1 = 1 \) and \( 1 \in O(1) \), the guess is correct.

2. **Inductive Step:** Assuming that \( T\left(\frac{n}{2}\right) \in O\left(\frac{n}{2} + \frac{n}{2} \cdot \lg \left(\frac{n}{2}\right)\right) \), we have to show that the guess holds for \( T(n) \):

\[
T(n) = 2T\left(\frac{n}{2}\right) + n \\
\leq 2[c \cdot \left(\frac{n}{2} + \frac{n}{2} \cdot \lg \left(\frac{n}{2}\right)\right)] + n, \text{ where } c > 0 \\
= c \cdot n + c \cdot n \cdot \lg n - cn + n \\
= c \cdot n \cdot \lg n + n \\
\]

Easy to show that \( c \cdot n \cdot \lg n + n \in O(n + n \cdot \lg n) \).
Master Theorem

**Theorem:** Let \( a \geq 1 \) and \( b > 1 \) be constants, let \( f(n) \) be a function, and let \( T(n) \) be defined on the nonnegative integers by the recurrence \( T(n) = a \cdot T(n/b) + f(n) \), where \( n/b \) can mean \( \lfloor n/b \rfloor \) or \( \lceil n/b \rceil \).

1. If \( f(n) \in O(n^{\log_b(a-\epsilon)}) \) for some constant \( \epsilon > 0 \), then \( T(n) \in \theta(n^{\log_b a}) \)
2. If \( f(n) \in \theta(n^{\log_b a}) \), then \( T(n) = \theta(n^{\log_b a} \cdot \log n) \)
3. If \( f(n) \in \Omega(n^{\log_b(a+\epsilon)}) \) for some constant \( \epsilon > 0 \), and if \( a \cdot f(n/b) \leq cf(n) \) for some constant \( c < 1 \) and all sufficiently large \( n \), then \( T(n) = \theta(f(n)) \)

**Examples:** \( T(n) = 9T(n/3) + n \), \( T(n) = T(\frac{2n}{3}) + 1 \), \( T(n) = 3T(\frac{n}{4}) + n\log n \), \( T(n) = 2T(\frac{n}{2}) + n\log n \), \( T(n) = n \cdot T^2(\frac{n}{2}) \).
Idea Behind Master Theorem: Case 1.

Recursion tree:

\[
\begin{align*}
&f(n) \\
&\quad \quad a \quad a \quad \vdots \\
&\quad \quad f(n/b) \quad f(n/b) \quad \cdots \quad f(n/b) \quad af(n/b) \\
&\quad \quad a \quad a \quad \vdots \\
&\quad \quad f(n/b^2) \quad f(n/b^2) \quad \cdots \quad f(n/b^2) \quad a^2 f(n/b^2) \\
&\quad \quad \vdots \\
&T(1)
\end{align*}
\]
Idea Behind Master Theorem: Case 1.

Recursion tree:

- The weight increases geometrically from the root to the leaves.
- The leaves hold a constant fraction of the total weight.

\[ T(n) \in \Theta(n^{\log_b a}) \]

Figure: The weight increases geometrically from the root to the leaves. The leaves hold a constant fraction of the total weight. \( T(n) \in \Theta(n^{\log_b a}) \).
Idea Behind Master Theorem: Case 2.

**Recursion tree:**

- $f(n)$
- $f(n/b)$
- $f(n/b^2)$
- $\cdots$

- $af(n/b)$
- $a^2f(n/b^2)$

$h = \log_b n$

**CASE 2: ($k = 0$)** The weight is approximately the same on each of the $\log_b n$ levels.

$\Theta(n^{\log_b a} \lg n)$
Outline of Today’s Class
Techniques for Bounding Recurrences

Idea Behind Master Theorem: Case 3.

Recursion tree:

\[ f(n) \quad a \quad f(n) \]
\[ f(n/b) \quad f(n/b) \quad \cdots \quad f(n/b) \quad a f(n/b) \]
\[ f(n/b^2) \quad f(n/b^2) \quad \cdots \quad f(n/b^2) \quad a^2 f(n/b^2) \]
\[ f(n/b^3) \quad f(n/b^3) \quad \cdots \quad f(n/b^3) \quad a^3 f(n/b^3) \]

\[ n^{\log_b a} T(1) \]
\[ \Theta(f(n)) \]

CASE 3: The weight decreases geometrically from the root to the leaves. The root holds a constant fraction of the total weight.