Solving Recurrences
Verifying through induction

C. Seshadhri

University of California, Santa Cruz
sesh@ucsc.edu

April 8, 2016
Yet another recurrence

Find the “best function” that satisfies the recurrence

\[ T(n) \leq 3T(n/4) + cn^2 \]

\[ T(1) \leq c \]
Which is true about the recursion tree? For nodes at depth $i$:

- (R) Input size is $n/2^i$ and number of such nodes is $2^i$
- (B) Input size is $n/4^i$ and number of such nodes is $3^i$
- (G) Input size is $n/3^i$ and number of such nodes is $2^i$
- (O) Input size is $n/4^i$ and number of such nodes is $4^i$
Which is true about the recursion tree? For nodes at depth $i$:

- (R) Input size is $n/2^i$ and number of such nodes is $2^i$
- (B) Input size is $n/4^i$ and number of such nodes is $3^i$
- (G) Input size is $n/3^i$ and number of such nodes is $2^i$
- (O) Input size is $n/4^i$ and number of such nodes is $4^i$

$T(n) \leq 3T(n/4) + cn^2$. So each node has three children. Input size goes down by a factor of 4.
Cost per level

What is the total cost of nodes at depth \( i \)?

- (R) \( cn^2 \)
- (B) \( cn^2 / 4^i \)
- (G) \( c(3/4)^i n^2 \)
- (O) \( c(3/16)^i n^2 \)
What is the total cost of nodes at depth $i$?

- (R) $cn^2$
- (B) $cn^2/4^i$
- (G) $c(3/4)^i n^2$
- (O) $c(3/16)^i n^2$

There are $3^i$ nodes at depth $i$, and input size is $n/4^i$. The cost per node is $cn^2/16^i$. Multiply by $3^i$ to get the answer.
Total cost

What is the total cost?

- (R) $O(n^2)$
- (B) $O(n^2 \log n)$
- (G) $O(n^3)$
- (O) $O(n^{2+3/16})$
What is the total cost?

- (R) $O(n^2)$
- (B) $O(n^2 \log n)$
- (G) $O(n^3)$
- (O) $O(n^{2+3/16})$

\[
\sum_{i=0}^{\text{max depth}} c \left(\frac{3}{16}\right)^i n^2 = cn^2 \\
\sum_{i=0}^{\text{max depth}} \left(\frac{3}{16}\right)^i
\]
Total cost

What is the total cost?

- (R) \( O(n^2) \)
- (B) \( O(n^2 \log n) \)
- (G) \( O(n^3) \)
- (O) \( O(n^{2+3/16}) \)

\[
\text{max depth} \sum_{i=0}^{\infty} \frac{3}{16}^i n^2 = cn^2 \leq \sum_{i=0}^{\infty} \frac{3}{16}^i = O(n^2)
\]
Recursion tree bounds

Typically, recurrences end up in three cases

- **All levels have same cost:** Just like the Mergesort recurrence, and we end up with an extra $O(\log n)$ factor. $O(n \log n)$

- **Cost forms increasing geometric progression:** Like $\sqrt{2^i n}$ for $T(n) \leq 2T(n/2) + c\sqrt{n}$. The last level dominates the cost. $O(n)$

- **Cost forms decreasing geometric progression:** Like $(3/16)^i n^2$ for $T(n) \leq 3T(n/4) + cn^2$. Root dominates the cost. $O(n^2)$

The master theorem formalizes this (under certain conditions) for these situations.
The Broad Approach

- **Draw a Recursion Tree:** Like we did yesterday. Add up the work. Cut corners to get a good guess
- **Use Induction:** To formally verify your guess
Example 1

\[ T(n) \leq 2T(n/2) + cn \]

with \( T(1) = 1 \).

- Make a guess: \( T(n) = O(n \log_2 n) \)
- We will therefore need to prove that

\[ T(n) \leq c'n \log n \]

for some constant \( c' > 0 \) and \( n \geq n_0 \).
Choose \( c = c \) and \( n_0 = 2 \).

- **Base case:** \( c \times 2 \times \log_2 2 = 2c \geq T(2) \).
Substitution

Induction Hypothesis

Choose $c = c$ and $n_0 = 2$.

- Base case: $c \times 2 \times \log_2 2 = 2c \geq T(2)$.
- Induction:

\[
T(n) \leq 2T(n/2) + cn \\
\leq 2c(n/2) \log_2(n/2) + cn \\
= cn \log_2(n/2) + cn \\
= cn(\log_2 n - 1) + cn \\
= cn \log_2 n
\]
Suppose we want to solve

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

- Make a guess: \( T(n) = O(n) \)
- We will therefore need to prove that

\[ T(n) \leq cn \]

for some constant \( c > 0 \) and \( n \geq n_0 \).
Suppose that \( c > 0 \) and \( T(m) \leq cm \) for all \( m < n \).

Then

\[
T(n) = 2T(n/2) + n \\
\leq 2cn/2 + n \\
= cn + n \\
= (c + 1)n
\]

Observing that \( T(n) \) is bounded by \( const. \times n \) completes the proof.

What went wrong?
Another Subtle Example

Suppose we want to solve

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

- Make a guess: \( T(n) = O(n) \)
- We will therefore need to prove that

\[ T(n) \leq cn \]

for some constant \( c > 0 \) and \( n \geq n_0 \).
Suppose that $c > 0$ and $T(m) \leq cm$ for all $m < n$.

Then

$$T(n) = 2T(n/2) + 1 \leq 2cn/2 + 1 = cn + 1$$

But this does not imply that $T(n) \leq cn$ :(
We will prove that $T(n) \leq n - 1$

Then

\[
T(n) = 2T(n/2) + 1 \\
\leq 2((n/2) - 1) + 1 \\
= n - 2 + 1 \\
\leq n - 1
\]
What’s the guess?

\[ T(n) \leq 2T(n/2) + c\sqrt{n}, \quad T(1) = c. \]

What should the induction hypothesis be?

- (R) \( T(n) \leq 4(n - c) \)
- (B) \( T(n) \leq 4c(n + \sqrt{n}) \)
- (G) \( T(n) \leq 4c(n - \sqrt{n}) \)
- (O) \( T(n) \leq 4cn \)
Recurrence Problem

Suppose we want to solve

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

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the non-negative integers by the recurrence

$$T(n) = aT(n/b) + f(n).$$

Then $T(n)$ has the following asymptotic bounds

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, 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 $f(n) = O(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and $af(n/b) \leq cf(n)$ for some constant $c < 1$ and sufficiently large $n$, then $T(n) = \Theta(f(n))$
Examples

Suppose we want to solve

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

- Observe that \( a = 9, \ b = 3, \) and \( f(n) = n \)
- The function \( n^{\log_b a} = n^{\log_3 9} = n^2 \)
- We have \( f(n) = n = O(n^{\log_b a - \epsilon}) \) with \( \epsilon = 1. \)
- Therefore \( T(n) = \Theta(n^2) \)