Recurrence Equations

David Fernández-Baca

Computer Science 311
Iowa State University

February 16, 2022

Recurrence Equations

A recurrence equation describes the running time, $T(n)$, of a recursive algorithm on problem of size n in terms of its running time on smaller inputs.

What is the recurrence for BinarySearch?

```
BinarySearch(A, v, left, right)
    Input: A sorted array A and a value v.
    Output: true if there is an index i ∈ {left, left + 1, ..., right}
            such that A[i] == v; false if no such i exists.
    if left > right then
        return false
    mid = ⌊(left + right)/2⌋
    if A[mid] == v then
        return true
    if v < A[mid] then
        return BinarySearch(A, v, left, mid − 1)
    else
        return BinarySearch(A, v, mid + 1, right)
```

Some Recurrence Equations

Merge Sort:

$$T(n) = 2T(n/2) + cn.$$

Randomized selection:

$$T(n) = T(3n/4) + cn.$$

General divide-and conquer:

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

Worst case of quick sort:

$$T(n) = T(n-1) + cn.$$

Order-d homogeneous linear recurrences with constant coefficients:

$$T(n) = c_1T(n-1) + c_2T(n-2) + \cdots + c_dT(n-d).$$

We assume throughout that $T(1)$ is $\Theta(1)$.

Some Methods for Solving Recurrences

- Substitution method: Guess a bound and then use mathematical induction to prove that guess is correct.
- Recursion-tree method: Convert the recurrence into a tree whose nodes represent the costs incurred at various levels of the recursion. Then add up the work over all nodes in the tree.
- Master method: Provides bounds for divide-and-conquer recurrences that have the form

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

Finite and Infinite Geometric series

For real $x \neq 1$, the summation

$$\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n$$

is a geometric or exponential series and its value is

$$\sum_{k=0}^{n} x^k = \frac{x^{n+1} - 1}{x - 1}.$$

When the summation is infinite and $|x| < 1$,

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1 - x}.$$

The Substitution Method

Example 1

Recurrence: $T(n) = T(n-1) + n$, with $T(1) = 1$.

Guess: $T(n) \leq cn^2$, for some constant c.

Proof of Guess.

- Base case: For $n = 1$, $T(1) = 1 \leq c$, if $c > 1$
- Hypothesis: $T(k) \leq ck^2$, for $k = 1, 2, \ldots, n-1$.
- Inductive step: Consider $n > 1$.

$$
\begin{aligned}
T(n) &= T(n-1) + n \\
&\leq c(n-1)^2 + n && \text{by hypothesis} \\
&= cn^2 - 2cn + c + n \\
&\leq cn^2 && \text{for } c \geq 1
\end{aligned}
$$

□

Example 2

Recurrence: $T(n) = 2T(\lfloor n/2 \rfloor) + n$, with $T(1) = \Theta(1)$.

Guess: $T(n) \le n \log n + cn$, for some c; i.e., $T(n) = O(n \log n)$.

Proof of Guess.

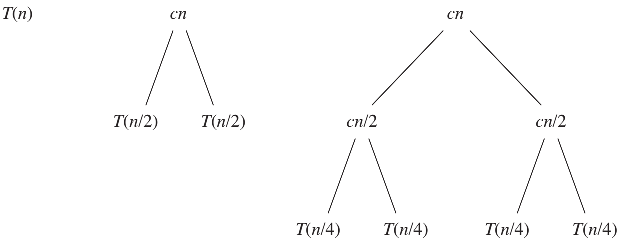
- Base case: For $n = 1$, $T(1) = b \le c$, if $c \ge b$
- Hypothesis: $T(k) \le ck \log k$, for $k = 1, 2, \ldots, n-1$.
- Inductive step: Consider $n > 1$.

$$\begin{aligned}
T(n) &= 2T(\lfloor n/2 \rfloor) + n \\
&\le 2(\lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor) + c\lfloor n/2 \rfloor) + n && \text{by hypothesis} \\
&\le n \log n - n \log 2 + cn + n \\
&= n \log n - 2n + cn + n \\
&\le n \log n + cn && \text{for } c \ge 1.
\end{aligned}$$

□

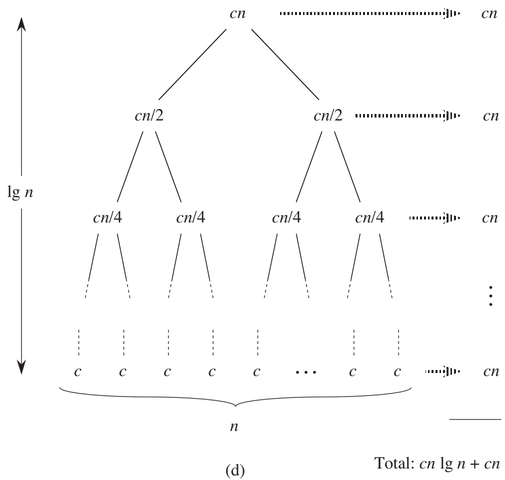
The Recursion-Tree Method

Example 1: $T(n) = 2T(n/2) + cn$



Source: CLRS

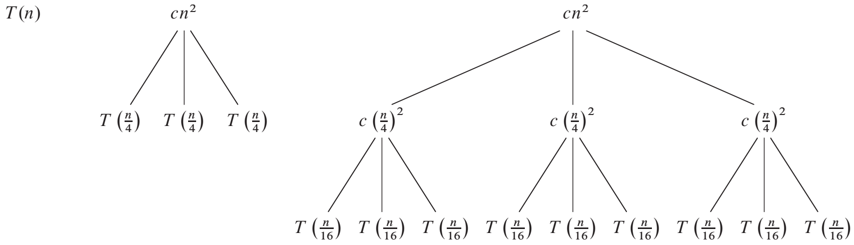
Example 1: $T(n) = 2T(n/2) + cn$



$$
\begin{aligned}
T(n) &= \overbrace{cn + \cdots + cn}^{\log n + 1 \text{ times}} \\
&= cn \log n + cn \\
&= \Theta(n \log n)
\end{aligned}
$$

Source: CLRS

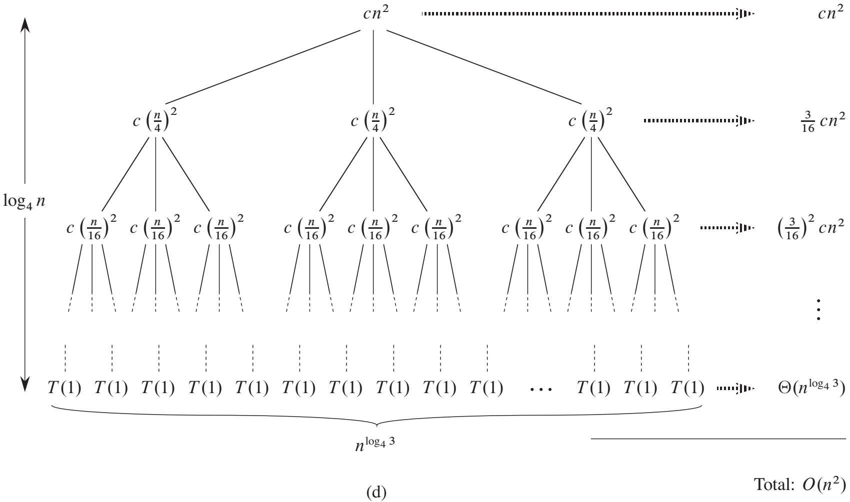
Example 2: $T(n) = 3T(n/4) + cn^2$



(a) (b) (c)

Source: CLRS

Example 2: $T(n) = 3T(n/4) + cn^2$



(d)

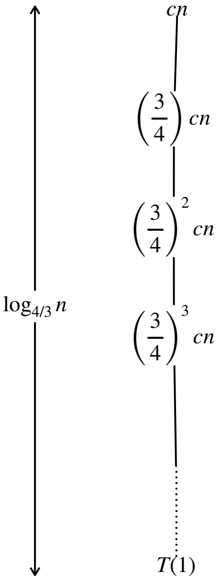
Source: CLRS

Last level has $\leq 3^{\log_4 n} = n^{\log_4 3}$ nodes.

Example 2: $T(n) = 3T(n/4) + cn^2$

$$
\begin{aligned}
T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n-1} cn^2 + \Theta(n^{\log_4 3}) \\
&= \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{1}{1-(3/16)}cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{16}{13}cn^2 + \Theta(n^{\log_4 3}) \\
&= O(n^2)
\end{aligned}
$$

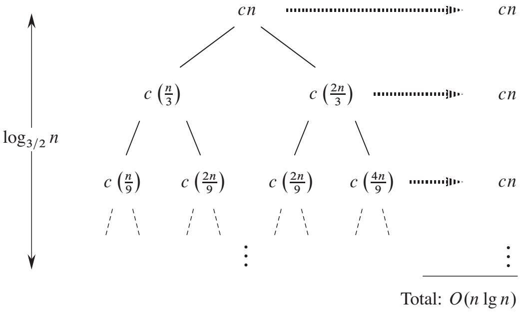
Example 3: $T(n) = T\left(\frac{3}{4}n\right) + cn$



$$
\begin{aligned}
T(n) &= \sum_{i=0}^{\log_{4/3} n-1} \left(\frac{3}{4}\right)^i cn + T(1) \\
&= \sum_{i=0}^{\log_{4/3} n-1} \left(\frac{3}{4}\right)^i cn + \Theta(1) \\
&\le \sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i cn + \Theta(1) \\
&= \frac{1}{1-\frac{3}{4}} cn + \Theta(1) \\
&= 4cn + \Theta(1) \\
&= O(n).
\end{aligned}
$$

Example 4: $T(n) = T(n/3) + T(2n/3) + cn$

To make a guess, draw recursion tree.



Source: CLRS

Guess: $T(n) \leq dn \log n$, for some d.

Example 4: $T(n) = T(n/3) + T(2n/3) + cn$

Use substitution to verify the guess.

$$\begin{aligned}
T(n) &\le T(n/3) + T(2n/3) + cn \\
&\le d(n/3)\log(n/3) + d(2n/3)\log(2n/3) + cn \\
&= (d(n/3)\log n - d(n/3)\log 3) \\
&\qquad + (d(2n/3)\log n - d(2n/3)\log(3/2)) + cn \\
&= dn\log n - d((n/3)\log 3 + (2n/3)\log(3/2)) + cn \\
&= dn\log n - d((n/3)\log 3 + (2n/3)(\log 3 - \log 2)) + cn \\
&= dn\log n - d((n/3)\log 3 + (2n/3)\log 3 - (2n/3)\log 2) + cn \\
&= dn\log n - d(n\log 3 - (2n/3)) + cn \\
&= dn\log n - dn(\log 3 - 2/3) + cn \\
&\le dn\log n,
\end{aligned}$$

which holds if $d \ge c/(\log 3 - 2/3)$.

The Master Theorem

Theorem (The 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 nonnegative integers by the recurrence

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

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. 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 $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $a \cdot f(n/b) \leq d \cdot f(n)$ for some constant $d < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.*

Applying the Master Theorem: Example 1

$$T(n) = 2T(n/2) + cn.$$

- $a = b = 2$ and $\log_b a = \log_2 2 = 1$.
- $n^{\log_b a} = n^1 = n$
- $f(n) = cn = \Theta(n^{\log_b a}) = \Theta(n)$.
- Thus, Case 2 of the Master Theorem applies, so

$$T(n) = \Theta(n^{\log_b a} \log n) = \Theta(n \log n).$$

Applying the Master Theorem: Example 2

$$T(n) = T(3n/4) + cn.$$

- $a = 1, b = 4/3, f(n) = cn$, and $\log_{4/3} 1 = 0$.
- Thus, $n^{\log_b a} = 1$ and $n^{\log_b a + \epsilon} = n^{\epsilon}$.
- $f(n) = cn = \Omega(n^{\epsilon})$, for any ϵ, $0 < \epsilon < 1$.
- $1 \cdot f(3n/4) = 3cn/4 \leq d \cdot cn$, when $3/4 < d < 1$.
- Thus, Case 3 of the Master Theorem applies, so

$$T(n) = \Theta(f(n)) = \Theta(n).$$

Applying the Master Theorem: Example 3

$$T(n) = 7T(n/2) + cn^2.$$

- $a = 7, b = 2$ and $\log_2 7 \approx 2.8074$.
- $n^{\log_b a} = n^{2.8074\ldots}$
- $f(n) = cn^2 = O(n^{\log_b a - \epsilon})$, for any ϵ such that $0 < \epsilon \leq \log_2 7 - 2$.
- Thus, Case 1 of the Master Theorem applies, so

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{2.8074\ldots}).$$

Some cases where the Master Theorem does not apply

- Number of subproblems is not a constant.

$$T(n) = n \cdot T(n/2) + n^2.$$

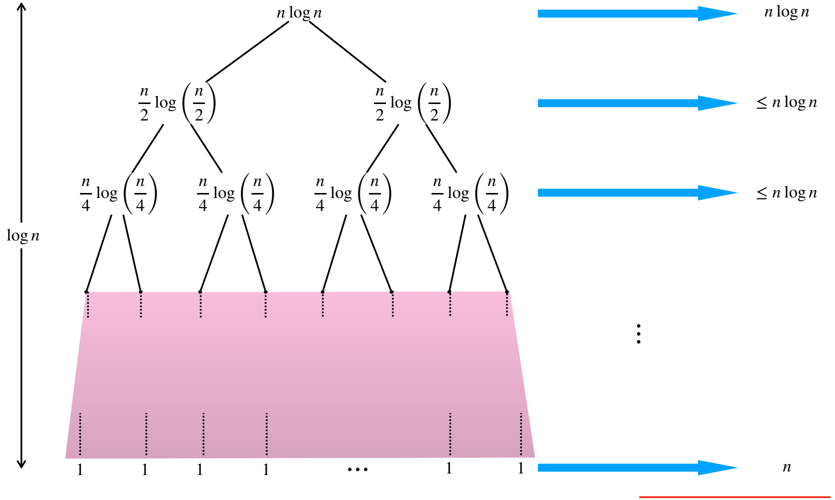
- Number of subproblems is less than 1.

$$T(n) = \frac{1}{2} T(n/2) + n^2.$$

- Work to divide and combine subproblems is not $\Theta(n^c)$.

$$T(n) = 2T(n/2) + n \log n$$

$T(n) = 2T(n/2) + n \log n$

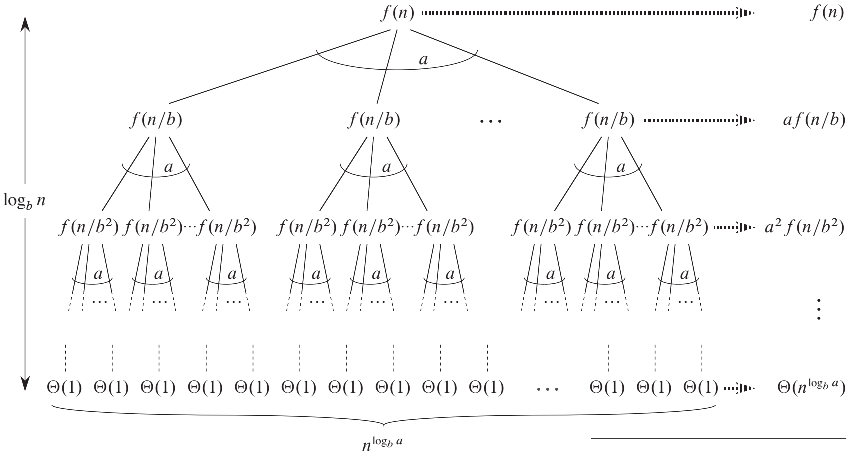


Total $\leq (n \log n) \cdot \log n + n$

$$T(n) = O(n(\log n)^2)$$

Proving the Master Theorem

Proving the Master Theorem



Total: $\Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$

Source: CLRS

Proving the Master Theorem

From the recursion tree,

$$T(n) = \Theta(n^{\log_b a}) + g(n), \quad \text{where} \quad g(n) = \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j).$$

The Master Theorem follows from:

Lemma

1. *If* $f(n) = O(n^{\log_b a - \epsilon})$ *for some constant* $\epsilon > 0$, *then* $g(n) = \Theta(n^{\log_b a})$.
2. *If* $f(n) = \Theta(n^{\log_b a})$, *then* $g(n) = \Theta(n^{\log_b a} \log n)$.
3. *If* $a \cdot f(n/b) \leq d \cdot f(n)$ *for some constant* $d < 1$ *and all sufficiently large* n, *then* $g(n) = \Theta(f(n))$.

Bibliography

References

[CLRS] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, *Introduction to Algorithms* (3rd edition), MIT Press, 2009.

[KT] Jon Kleinberg and Éva Tardos, *Algorithm Design*, Addison-Wesley, 2006.