Recurrence

Textbook, Chapter 4

CSCE310: Data Structures and Algorithms

`www.cse.unl.edu/~choueiry/S01-310/`

Berthe Y. Choueiry (Shu-we-ri)
Ferguson Hall, Room 104
choueiry@cse.unl.edu, Tel: (402)472-5444

- The running time of **recursive algorithm** is often described by a **recurrence equation**
- A **recurrence** is a an equation or an inequality that describes a function in terms of **its** value on **smaller** inputs
- A **solution to a recurrence** is an equivalent equation that is not expressed in terms of itself: the n^{th} can be computed directly
- Worst-case running time of `MergeSort`:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T(n/2) + \Theta(n) + \Theta(1) = 2T(n/2) + \Theta(n) & \text{if } n > 1 \end{cases}$$

  we claimed that solution: $T(n) = \Theta(n \lg n)$

Four (4) methods for solving recurrence:

1. Characteristic equation (i.e., pretest!)
2. Substitution method
3. Iteration method
4. Master method

Methods for solving recurrence

1. For homogeneous linear recurrences of the form

$$a_0T(n) + a_1T(n-1) + \ldots + a_kT(n-k) = 0$$

   Write and solve the characteristic equation:

$$a_0x^k + a_1x^{k-1} + \ldots + a_k = 0$$

2. Substitution method: guess a bound and use mathematical induction to prove guess correct
3. Iteration method: Convert recurrence into a summation, then solve using techniques to bound summations
4. Master method provides bounds for recurrence of form:

$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n)$$

   $a \geq 1$ and $b > 1$: constants; and $f(n)$ a given function
   $\longrightarrow$ Three (3) cases, cookbook/recipe

Technicalities

Omit floor, ceilings, and boundary conditions

- $T(n)$ is defined for $n \in \mathbb{N}$
  Example: for `MergeSort` we have in fact

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) & \text{if } n > 1 \end{cases}$$

- Omit boundary conditions:
  For n sufficiently small $T(n) = \Theta(1)$ assumed to hold

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

Changing $T(1)$ changes solution to recurrence by a constant factor only, so order of growth is unchanged

Substitution method

- Guess form of recurrence
- Use mathematical induction to find constants and show how solution works
- Powerful, but requires guessing!
- can be used to establish upper/lower bounds (O, Ω-notations)

Substitution Method: example I

Consider the recurrence $S(n) = \begin{cases} 1 & \text{when } n = 1 \\ S(n-1) + n & \text{otherwise} \end{cases}$

Then the solution is $S(n) = \frac{n(n+1)}{2}$

Proof:

1. Assume: for $n < k$, $S(n) = \frac{n(n+1)}{2}$. Then,

$$\begin{aligned} S(k) &= S(k-1) + k = \frac{(k-1)(k)}{2} + k = \frac{k^2 - k}{2} + k = \frac{k^2 - k + 2k}{2} \\ &= \frac{k^2 + k}{2} = \frac{k(k+1)}{2} \end{aligned}$$

   Thus, $S(n) = \frac{n(n+1)}{2}$ for all $n \geq 1$

2. When $n = 1$, $S(1) = 1(2)/2 = 1$.

Substitution Method: example II

Determine upper bound of: $T(n) = 2T(\lfloor n/2 \rfloor) + n$

- Guess: $T(n) = O(n \lg n$
- Prove: $T(n) \leq cn \lg n$ for $c > 0$
- Assume bound holds for $\lfloor n/2 \rfloor \Rightarrow T(\lfloor n/2 \rfloor) \leq c\lfloor n/2 \rfloor \lg\lfloor n/2 \rfloor)$
- Compute:

$$\begin{aligned} T(n) &\leq 2(c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)) + n \\ &\leq cn \lg(n/2) + n \\ &= cn \lg n - cn \lg 2 + n \\ &= cn \lg n - cn + n \\ &\leq cn \lg n \,, \end{aligned}$$

  holds for $c \geq 1$

- Mathematical induction requires checking boundary conditions $\longleftarrow$ *Sometimes problematic*

Determine upper bound of: $T(n) = 2T(\lfloor n/2 \rfloor) + n$

Checking boundary conditions:

- Prove: $T(n) \leq cn \lg n$ for $c > 0$
- Assume $T(1) = 1$
- $T(1) \leq c1 \lg 1 = 0$ $\rightarrow$ *no way to choose c*
- But, asymptotic notation only requires to prove $T(n) \leq cn \lg n$ for $n \geq n_0$, where n_0 is a constant
  Ignore $n = 1$, check for $n = 2, n = 3$ as boundary conditions
  $T(2) = 4, T(3) = 5$
- Choose c large enough to have $T(2) = c2 \lg 2$ and $T(3) = c3 \lg 3$
  Holds for any $c \geq 2$

Indication: to make inductive assumption work for small n, extend boundary conditions

Making a good guess

1. Similarity with a known case
   Knowing that $T(n) = 2T(\lfloor n/2 \rfloor) + n$ is $O(n \lg n)$
   It is easy to find that of $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$

2. Start with loose upper/lower bounds, reduce range of uncertainty
   Start with $T(n) = \Omega(n)$ and $T(n) = (n^2)$
   Raise lower bound, lower upper bound until getting:
   $T(n) = \Theta(n \lg n)$

Subtleties

You guess the correct bound, but the math does not work

Example: $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$

Guess: $T(n) = O(n)$. Prove: $T(n) \leq cn$, with $c > 0$

Substitute in recurrence: $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 \Rightarrow$
$T(n) \leq c(\lfloor n/2 \rfloor) + c(\lceil n/2 \rceil) + 1 \Rightarrow T(n) \leq cn + 1$

Problem: we didn't prove the exact form of inductive hypothesis

Temptation: $T(n) = O(n \lg n)$? $T(n) = O(n^2)$? *Nooo!*

Revise guess: $T(n) \leq cn - b$, with $c > 0, b \geq 0$

Substitute in recurrence: $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 \Rightarrow$
$T(n) \leq c(\lfloor n/2 \rfloor - b) + c(\lceil n/2 \rceil - b) + 1 = cn - 2b + 1 \Rightarrow$
$T(n) \leq cn - b$ with $b \geq 1$

Key to make induction work: assumed stronger condition on $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$ and proved stronger condition for n.

Pitfalls

Asymptotic notation can be confusing

Consider $T(n) = T(\lfloor n/2 \rfloor) + n$

(False) guess: $T(n) = O(n)$ Prove: $T(n) \leq cn$, with $c > 0$

Substitute in recurrence: $T(n) = 2T(\lfloor n/2 \rfloor) + n \Rightarrow$
$T(n) \leq 2c(\lfloor n/2 \rfloor) + n \Rightarrow T(n) \leq (c+1)n = \mathbf{O(n)}$ $\underline{\longleftarrow \text{ Wrong}}$

We need to prove the exact form of the inductive hypothesis:
$T(n) \leq cn$

Changing variables

Use algebraic variable substitution

Example: $T(n) = 2T(\lfloor \sqrt{n} \rfloor) + \lg n$

Rename: $m = \lg n$, that is $2^m = n$

$T(2^m) = 2T(\lfloor 2^{m/2} \rfloor) + m$

Rename: $S(m) = T(2^m)$, that is $S(m/2) = T(2^{m/2})$

$S(m) = 2S(\lfloor m/2 \rfloor) + m$, which we know $S(m) = O(m \lg m)$

Change back: $T(n) = T(2^m) = S(m) = O(m \lg m) = O(\lg n \lg \lg n)$

Summary: Substitution method

- Guess and prove by induction
- Use and adapt forms already encountered
  Consider changing variables
- Prove induction on a stronger condition
- Beware of asymptotic notation: need to prove the exact form of inductive hypothesis

Iteration method

- No guessing
- More algebra
- Expands (iterate) recurrence and express it as a summation (of terms in n and initial conditions)
- Uses algebraic techniques for summations to find bounds on solution

Iteration method: Example

Consider: $T(n) = 3T(\lfloor n/4 \rfloor) + n$

Iterate:

$$\begin{aligned} T(n) &= n + 3T(\lfloor n/4 \rfloor) \\ &= n + 3(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor)) \\ &= n + 3(\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor))) \\ &= n + 3\lfloor n/4 \rfloor + 9\lfloor n/16 \rfloor + 27T(\lfloor n/64 \rfloor)\ , \end{aligned}$$

where $\lfloor \lfloor n/4 \rfloor /4 \rfloor = \lfloor n/16 \rfloor$

i^{th} term is $3^i \lfloor n/4^i \rfloor$

i^{th} term is $3^i \lfloor n/4^i \rfloor$

Iteration hits $n = 1$ for $\lfloor n/4^i \rfloor = 1$ and $i \geq \log_4 n$

So:

$$\begin{aligned} T(n) &\leq n + 3n/4 + 9n/16 + 27n/64 + \cdots + 3^{\log_4 n}\Theta(1) \\ &\leq n \sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i + \Theta(n^{\log_4 3}) \\ &= 4n + o(n) \\ &= O(n) \, . \end{aligned}$$

Knowing: $3^{\log_4 n} = n^{\log_4 3}$ and $\log_4 3 < 1$

Iteration method leads to lots of algebra

Two key parameters:

1. Number of times to iterate recurrence to reach boundary conditions
2. Sum of terms obtained by iterating

Sometimes, iteration helps guessing, so we can use substitution!

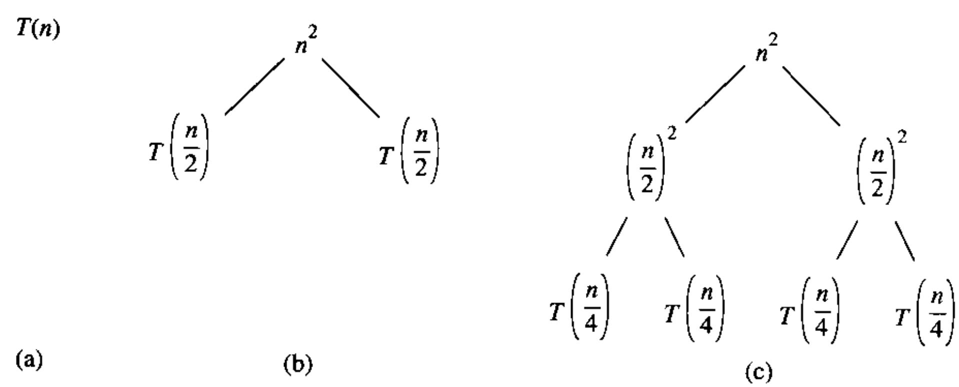
Recursion tree (I)

Convenient for visualization and algebraic bookkeeping

Specially useful in divide-&-conquer algorithms

Recursion tree for $T(n) = 2T(n/2) + n^2$:

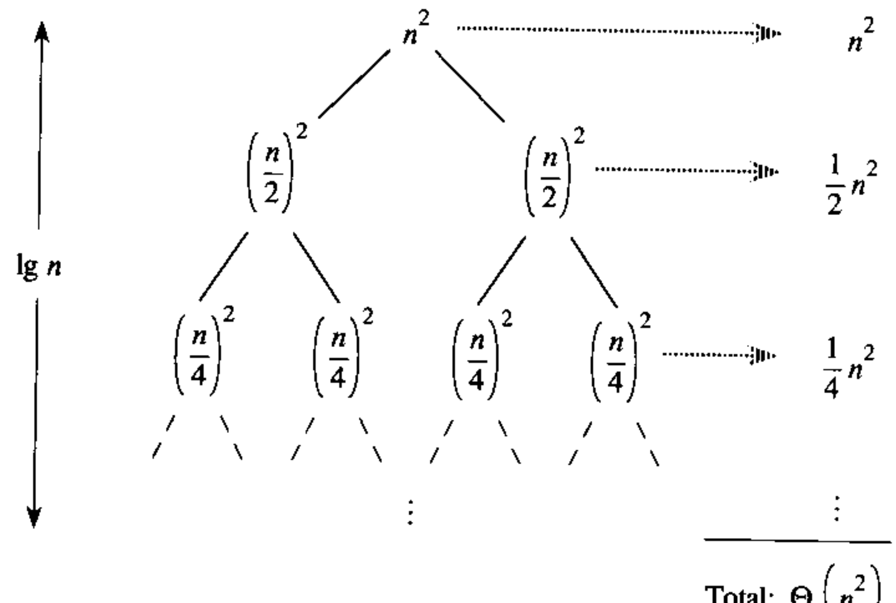
Root: n^2. Two subtrees, each of $n/2$ (assuming n exact power of 2)



Each subtree yields a tree for smaller recurrences..

Recursion tree (II)

Expand each subtree until we hit boundary condition



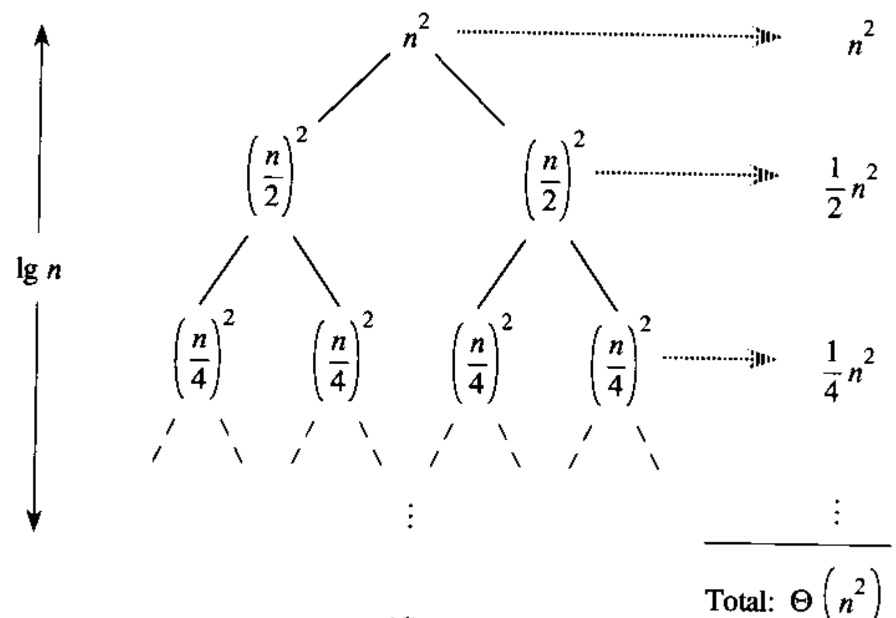
(d)

Value at each level?

Height? At deepest level i, $(\frac{n}{i})^2 = 1$

How many levels?

Evaluate recurrence: add values across each level in the tree



(d)

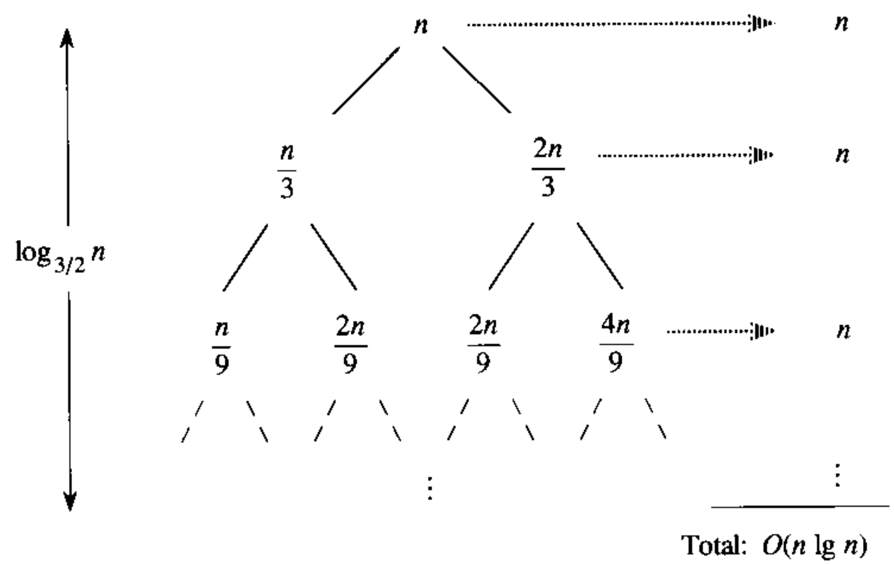
Values decrease geometrically, sum at most a constant factor of first term

$\longrightarrow$ Solution of recurrence equation is $\Theta(n^2)$

Recursion tree: another example

Recursion tree for $T(n) = T(n/3) + T(2n/3) + n$:

(omitting floors and ceilings for simplicity)



Add values across levels: n at each level

At deepest level k, we have $(2/3)^k n = 1$ when $k = \log_{3/2} n$

Thus height is $\log_{3/2} n$

Recurrence is at most $n \log_{3/2} n$, $O(n \lg n)$

Master method: memorize solutions for 3 cases

Cookbook for recurrences of the form:

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

where $a \geq 1, b > 1$, constants

Equation: running time of an algorithm that divides a problem into a subproblems, each of size n/b

a subproblems solved recursively, each in time $T(n/b)$

Cost of dividing, then combining results is $f(n)$ $(= D(n) + C(n))$

Example: `MergeSort`, $a = b = 2$ and $f(n) = \Theta(n)$

(again, we ignore floors and ceilings)

The Master theorem

Let $a \geq 1, b > 1$ constants; $f(n)$ a function; $T(n)$ defined on nonnegative integers ($\in \mathbb{N}$) by recurrence: $T(n) = aT(n/b) + f(n)$

$T(n)$ can be bounded asymptotically with:

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

Intuition: In each case, we compare $f(n)$ with $n^{\log_b a}$

the larger dominates and determines solution to recurrence equation

Case 1: $n^{\log_b a}$ dominates $\Rightarrow T(n) = \Theta(n^{\log_b a})$

Case 3: $f(n)$ dominates $\Rightarrow T(n) = f(n)$

Case 2: Same size $\Rightarrow \times \lg n$, $T(n) = \Theta(n^{\log_b a} \lg n) = \Theta(f(n) \lg n)$

Warning

$T(n) = aT(n/b) + f(n)$ can be bounded asymptotically with:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, ... then $T(n) = \Theta(f(n))$

1. **Case 1:** $f(n)$ polynomially smaller than $n^{\log_b a}$, asymptotically smaller by a factor of n^{ϵ}
2. **Case 3:** $f(n)$ polynomially larger than $n^{\log_b a}$, and satisfy $af(n/b) \leq cf(n)$ (regularity condition)
3. **Gaps:** 3 cases do not cover all possibilities.
4. **Master theorem:** does not hold if we are in gaps or if regularity condition is not verified

Master method: example I

$$T(n) = 9T(n/3) + n$$

$a = 9$, $b = 3$, $f(n) = n$, $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$
$f(n) = n = O(n^{\log_3 9 - \epsilon})$ where $\epsilon = 1$

Apply case 1 of master theorem, $T(n) = \Theta(n^2)$

Master method: example II

$$T(n) = T(2n/3) + 1$$

$a = 1$, $b = 2/3$, $f(n) = 1$, $n^{\log_b a} = n^{\log_{2/3} 1} = n^0 = 1$
$f(n) = n^0 = 1$

Apply case 2 of master theorem, $T(n) = \Theta(\lg n)$

Master method: example III

$$T(n) = 3T(n/4) + n \lg n$$

$a = 3$, $b = 4$, $f(n) = n \lg n$, $n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$
$f(n) = n \lg n = \Omega(n^{log_4 3+\epsilon})$, $\epsilon \approx 0,2$
And regularity condition applies for sufficiently large n

Apply case 3 of master theorem, $T(n) = \Theta(n \lg n)$

Master method: (counter)-example IV

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

$f(n) = n \lg n$ and $n^{\log_b a} = n^{\log_2 2} = n$
$f(n)$ is not <u>polynomially</u> larger

Recurrence falls between in gap between case 2 and case 3

Common functions

Floors and ceilings:

- The floor of x: $\lfloor x \rfloor$
- The ceiling of x: $\lceil x \rceil$
- $\forall\, n \in \mathbb{N}$, $\lfloor n/2 \rfloor + \lceil n/2 \rceil = n$

Logarithms:

- Binary logarithm: $\lg n = \log_2 n$
- Natural logarithm: $\ln n = \log_e n$
- $\log_a n = \frac{\ln n}{\ln a}$ and $\log n = \frac{\ln n}{\ln 10}$
- $\ln e = 1$, $\ln 1 = 0$, $\log_k 1 = 0$, $\log_k k = 1$
- Exponentiation: $\lg^k n = (\lg n)^k$
- Composition: $\lg \lg n = \lg(\lg n)$
- $a = b^{\log_b a}$
- $\log_b a^n = n \log_b a$
- $a^{\log_b n} = n^{\log_b a}$

Sum of finite series:

- Arithmetic: $S_n = \frac{n(t_1 + t_n)}{2}$
- Geometric: $S_n = t_1 \frac{r^n - 1}{r - 1} = \frac{t_n r - t_1}{r - 1}$