Analysis of Algorithms 1 (Fall 2013) Istanbul Technical University Computer Eng. Dept.

Chapter 4: Recurrences

Course slides from Susan Bridges @MS State have been used in preparation of these slides.

Last updated: Sept 25, 2013

Purpose

- Understand what a recurrence equation is and why we need to solve it.
- Understand methods to solve a recurrence equation.

Outline

- Recurrence Equation: What and why
- Methods to Solve Recurrences:
  - Substitution (constructive induction)
  - Iteration of the recurrence
    - Recurrence Trees
  - Master Theorem

Recurrences

- Definition – a recurrence is an equation or inequality that describes a function in terms of its value on smaller inputs
- Example – recurrence for Merge-Sort

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

Why Recurrences?

- The complexity of many interesting algorithms is easily expressed as a recurrence – especially divide and conquer algorithms
- The form of the algorithm often yields the form of the recurrence
- The complexity of recursive algorithms is readily expressed as a recurrence.

Why solve recurrences?

- To make it easier to compare the complexity of two algorithms
- To make it easier to compare the complexity of the algorithm to standard reference functions.

Example Recurrences for Algorithms

- Insertion sort

$$T(n) = \begin{cases} 1 & \text{for } n \le 1 \\ T(n-1) + n & \text{otherwise} \end{cases}$$

- Linear search of a list

$$T(n) = \begin{cases} 1 & \text{for } n \le 1 \\ T(n-1) + 1 & \text{otherwise} \end{cases}$$

Recurrences for Algorithms, continued

- Binary search

$$T(n) = \begin{cases} 1 & \text{for n } \le 1 \\ \mathrm{T(n/2)} + 1 & \text{otherwise} \end{cases}$$

Casual About Some Details

- Boundary conditions
  - These are usually constant for small n
  - We sometimes use these to fill in the details of a "rough guess" solution
- Floors and ceilings
  - Usually makes no difference in solution
  - Usually assume n is an "appropriate" integer (i.e. a power of 2) and assume that the function behaves the same way if floors and ceilings were taken into consideration

Merge Sort Assumptions

- Actual recurrence is:

$$T(n) = \begin{cases} 1 & \text{for } n \le 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + n & \text{otherwise} \end{cases}$$

- But we typically assume that n = 2^k where k is an integer and use the simpler recurrence.

Methods for Solving Recurrences

- Substitution (constructive induction)
- Iteration of the recurrence
- Master Theorem

Substitution Method (Constructive Induction)

- Use mathematical induction to derive an answer
- Steps
  1. Guess the form of the solution
  2. Use mathematical induction to find constants or show that they can be found and to prove that the answer is correct

Substitution

- Goal

  Derive a function of n (or other variables used to express the size of the problem) that is not a recurrence so we can establish an upper and/or lower bound on the recurrence

  May get an exact solution or may just get upper or lower bounds on the solution

Constructive Induction

- Suppose *T* includes a parameter *n* and *n* is a natural number (positive integer)
- Instead of proving directly that *T* holds for all values of *n*, prove
  - *T* holds for a base case *b* (often $n = 1$)
  - For every $n > b$, if *T* holds for $n-1$, then *T* holds for *n*.
    - Assume *T* holds for $n-1$
    - Prove that *T* holds for *n* follows from this assumption

Example 1

- Given

$$T(n) = \begin{cases} 1 & \text{for } n \leq 1 \\ T(n-1) + n & \text{otherwise} \end{cases}$$

- Prove $T(n) \in O(n^2)$
  - Note that this is the recurrence for insertion sort and we have already shown that this is $O(n^2)$ using other methods

$$T(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \in O(n^2)$$

Proof for Example 1

- Guess that the solution for T(n) is a quadratic equation $T(n) = an^2 + bn + c$
- Base case T(1) = a + b + c. We need to find values for a, b, and c first.
- Assume this solution holds for *n-1*

$$T(n-1) = a(n-1)^2 + b(n-1) + c$$

- Now consider the case for *n.* Begin with the recurrence for *T(n)*

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

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

By assumption

$T(n) = a(n-1)^2 + b(n-1) + c + n$

$= an^2 + 2an + a + bn - b + c + n$

$= an^2 + (1-2a+b)n + (a-b+c)$

If we are to conclude T(n) = $an^2 + bn + c$ then it must be the case that

$a = a \qquad b = 1 - 2a + b \qquad c = a - b + c$

From these we can calculate a and b

from $b = 1 - 2a + b$ we get $2a - 1 = 0$, or $a = 1/2$

from $c = a - b + c$ we get $a - b = 0$ or $b = 1/2$

The values for a and b are now constrained, but the value for c is not. But we now have a more complete hypothesis and we can use this new hypothesis and the definition of the recurrence to get a value for c.

$T(n) = \frac{1}{2}(n)^2 + \frac{1}{2}(n) + c$ and $T(n) = 1$ for $n = 1$

$$T(1) = \tfrac{1}{2}(1)^2 + \tfrac{1}{2}(1) + c$$
$$= 1 + \mathrm{c}$$

but since $T(n) = 1$ for $n = 1$, $\mathrm{c} = 0$ and

$T(n) = \frac{1}{2}(n)^2 + \frac{1}{2}(n)$ for $n \geq 1$

Example 2– Establishing an Upper Bound

Recurrence: $T(n) = 4T(n/2) + n$

Guess: $T(n) \in O(n^3)$

Assumption: $n = 2^k$ where k is an integer

In this case we want to prove that $T(n) \leq cn^3 \quad \forall n \geq n_0$

Assume $T(n/2) \leq c(n/2)^3 \quad \forall n \geq n_0$

Starting with the recurrence for $T(n)$

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\leq 4c(n/2)^3 + n \\
&\leq 1/2cn^3 + n
\end{aligned}
$$

This is not quite what we need: $T(n) \leq c(n)^3$

We want to prove that $T(n) \le cn^3 \quad \forall n \ge n_0$

$T(n) \le 1/2cn^3 + n$

Trick

$$T(n) \le \tfrac{1}{2}cn^3 + n$$
$$\le (cn^3 - \tfrac{1}{2}cn^3) + n$$
$$\le cn^3 - (\tfrac{1}{2}cn^3 - n)$$
$$\le cn^3 \quad \forall c > 2 \quad \text{and } n > 1$$

General heuristic – try to write the expression in the form

< answer you want > - < something greater than 0 >

We still need a boundary condition specified

We have shown that $T(n) \le cn^3 \quad \forall c > 2$ and $n \ge 1$

Select a c value that is large enough to satisfy a specified boundary condition. In this case we can select a $c = 3$ for a boundary condition of $n = 1$

Note that we have established an upper bound, but it is not a tight bound. See the next example.

Ex. 3–Fallacious Argument

Recurrence: $T(n) = 4T(n/2) + n$

Guess: $T(n) \in O(n^2)$ (Assume $n = 2^k$ for an integer k)

In this case we want to prove that $T(n) \leq cn^2 \quad \forall n \geq n_0$

Assume $T(n/2) \leq c(n/2)^2 \quad \forall n \geq n_0$

Starting with the recurrence for $T(n)$

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\leq 4c(n/2)^2 + n \\
&\leq cn^2 + n
\end{aligned}
$$

$\therefore T(n) \in O(n^2)$ But this is incorrect, because $cn^2 + n \leq cn^2$ only holds for $n \leq 0$ (must hold for all n ≥ base)

Ex. 3–Try again

When you get to this point

$$T(n) \le cn^2 + n$$

Revise the inductive hypothesis

Heuristic

When you find yourself in the situation

$$T(n) \le \text{< term you want > + < something + >}$$

start over with a new inductive hypothesis in which you substract a lower order term

Guess $T(n) \le c_1 n^2 - c_2 n$

Assume $T(n/2) \le c_1 (n/2)^2 - c_2 (n/2)$

Ex. 3–Try again cont.

Starting with recurrence

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\le 4(c_1(n/2)^2 - c_2(n/2)) + n \\
&\le c_1 n^2 - 2c_2 n + n \\
&\le c_1 n^2 - c_2 n - (c_2 n - n)
\end{aligned}
$$

Now the first two terms are in the correct form and the last term is positive for all values of $c_2 \ge 1$ so

$T(n) \le c_1 n^2 - c_2 n$ for all $c_2 \ge 1$

Select c_1 to be large enough to handle the initial conditions.

Boundary Conditions

- Boundary conditions are not usually important because we do not need an actual *c* value (if polynomially bounded)
- But sometimes it makes a big difference
  - Exponential solutions
  - Suppose we are searching for a solution to:

$$T(n) = T(n/2)^2$$

and we find the partial solution

$$T(n) = c^n$$

Boundary Conditions cont.

If the boundary condition is

$$T(1) = 2$$

this implies that $T(n) \in \Theta(2^n)$.

But if the boundary condition is

$$T(1) = 3$$

this implies that $T(n) \in \Theta(3^n)$.

And $\Theta(3^n) \neq \Theta(2^n)$.

The results are even more dramatic if $\mathrm{T}(1) = 1$

$$T(1) = 1 \Rightarrow T(n) = \Theta(1^n) = \Theta(1)$$

Boundary Conditions

- The solutions to the recurrences below have very different upper bounds

$$T(n) = \begin{cases} 1 & \text{for } n = 1 \\ T(n/2)^2 & \text{otherwise} \end{cases}$$

$$T(n) = \begin{cases} 2 & \text{for } n = 1 \\ T(n/2)^2 & \text{otherwise} \end{cases}$$

$$T(n) = \begin{cases} 3 & \text{for } n = 1 \\ T(n/2)^2 & \text{otherwise} \end{cases}$$

Changing Variables

Can sometimes change a recurrence to a more familiar one.

$\mathrm{T(n)} = 2\mathrm{T}\left(\left\lfloor \sqrt{\mathrm{n}} \right\rfloor\right) + \lg n$

Assumption: won't worry about rounding to integers.

Rename $m = \lg n$ or $n = 2^m$

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

Rename $\mathrm{S(m)} = T(2^m)$ to get the new recurrence

$$S(m) = 2S(m/2) + m$$

Changing Variables continued

$$S(m) = 2S(m/2) + m$$

is a recurrence we have already solved

$$S(m) = O(m \lg m)$$

Changing from S(m) to T(n), we obtain

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

$$= O(\lg n \lg \lg n)$$

Iterating the Recurrence

- The math can be messy with this method
- Can sometimes use this method to get an estimate that we can use for the substitution method

Example 4

$T(n) = n + 4T(n/2)$

Start iterating the recurrence

$T(n) = n + 4(n/2 + 4T(n/4))$

$\quad = n + 2n + 16T(n/4)$

Iterate the recurrence again

$T(n) = n + 2n + 16(n/4 + 4T(n/8))$

$\quad = n + 2n + 4n + 64T(n/8)$

We observe that the *ith* term in the series is $2^i n$

How far do we iterate before we reach a boundary condition? If we use 1 as a boundary condition, it will be when we reach $n/2^i = 1$.

When $n/2^i = 1$, or $2^i = \text{n}$, then $i = \lg n$

Now, since we know that the *ith* term is $2^i n$

we can rewrite the series as

$$T(n) = n + 2n + 4n + \ldots + 2^{\lg n} nT(1)$$

Remember that $a^{\log_b n} = n^{\log_b a}$

$$T(n) = n + 2n + 4n + \ldots + n^{\lg 2} n$$

$$= n + 2n + 4n + \ldots + n^2$$

$$= n + 2n + 4n + \ldots + 2^{\lg n - 1} n + n^2$$

$$T(n) == n + 2n + 4n + \ldots + 2^{\lg n - 1} n + n^2 T(1)$$

Factor out a geometric progression

$$\sum_{i=0}^{n} x^k = \frac{x^{n+1} - 1}{x - 1} \quad \text{for } x \neq 1$$

$$T(n) = n(2^0 + 2^1 + 2^2 \ldots + 2^{\lg n \text{-} 1}) + n^2 \mathrm{T}(1)$$

$$= n\left(\frac{2^{\lg n} - 1}{2 - 1}\right) + \Theta(n^2)$$

$$= n(n - 1) + \Theta(n^2)$$

$$= \Theta(n^2) + \Theta(n^2)$$

$$= \Theta(n^2)$$

Example 5

- Eq. to be solved: $T(n) = 4\,T(n-1) + 1$

$$
\left.
\begin{aligned}
T(n) &= 4\,T(n-1) + 1 \\
T(n-1) &= 4\,T(n-2) + 1 \\
T(n-2) &= 4\,T(n-3) + 1 \\
T(n-3) &= 4\,T(n-4) + 1 \\
&\vdots \\
T(3) &= 4\,T(2) + 1 \\
T(2) &= 4\,T(1) + 1
\end{aligned}
\right\} n - 1
$$

- $T(n) = \cancel{4\,T(n-1)} + 1$
- $\cancel{4^1\,T(n-1)} = \cancel{4^1\,4\,T(n-2)} + 4^1\,1$
- $\cancel{4^2\,T(n-2)} = \cancel{4^2\,4\,T(n-3)} + 4^2\,1$
- $\cancel{4^3\,T(n-3)} = \cancel{4^3\,4\,T(n-4)} + 4^3\,1$
- $\cancel{4^{n-3}\,T(3)} = \cancel{4^{n-3}\,4\,T(2)} + 4^{n-3}\,1$
- $\cancel{4^{n-2}\,T(2)} = 4^{n-2}\,4\,T(1) + 4^{n-2}\,1$

- $$T(n) = 4^{n-1}T(1) + \sum_{i=0}^{n-2} 4^i$$

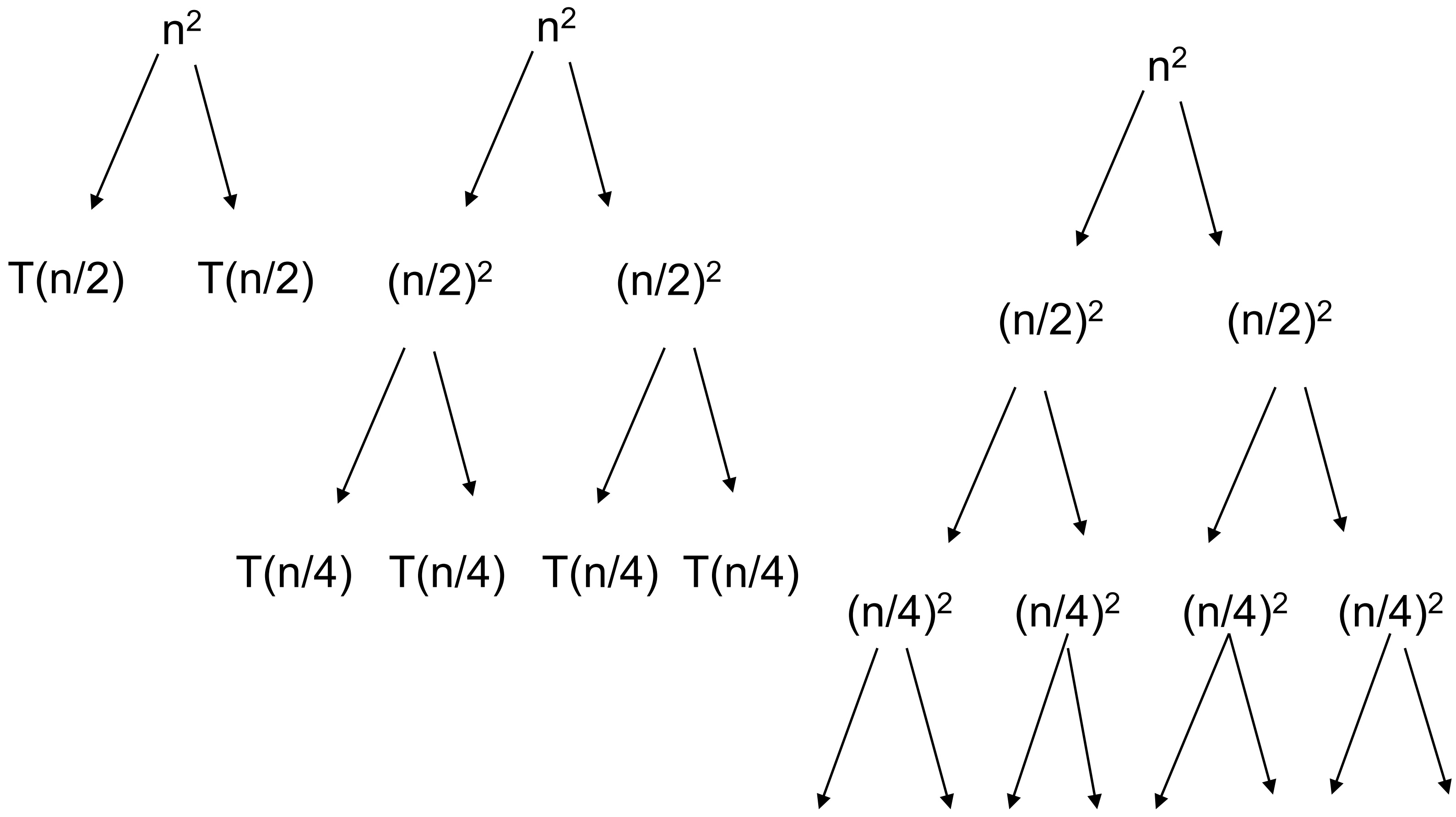
$$T(n) = 4^{n-1} + \frac{4^{n-1}-1}{4-1}$$

$$T(n) = \frac{4^n - 1}{3}$$

Recurrence Trees

- Allow you to visualize the process of iterating the recurrence
- Allows you make a good guess for the substitution method
- Or to organize the bookkeeping for iterating the recurrence
- Example

$$T(n) = 2T(n/2) + n^2$$


n^2
$T(n/2)$ $T(n/2)$
n^2
$(n/2)^2$ $(n/2)^2$
$T(n/4)$ $T(n/4)$ $T(n/4)$ $T(n/4)$
n^2
$(n/2)^2$ $(n/2)^2$
$(n/4)^2$ $(n/4)^2$ $(n/4)^2$ $(n/4)^2$

Counting things

Inverse harmonic series (page 44)

for $|x| < 1$

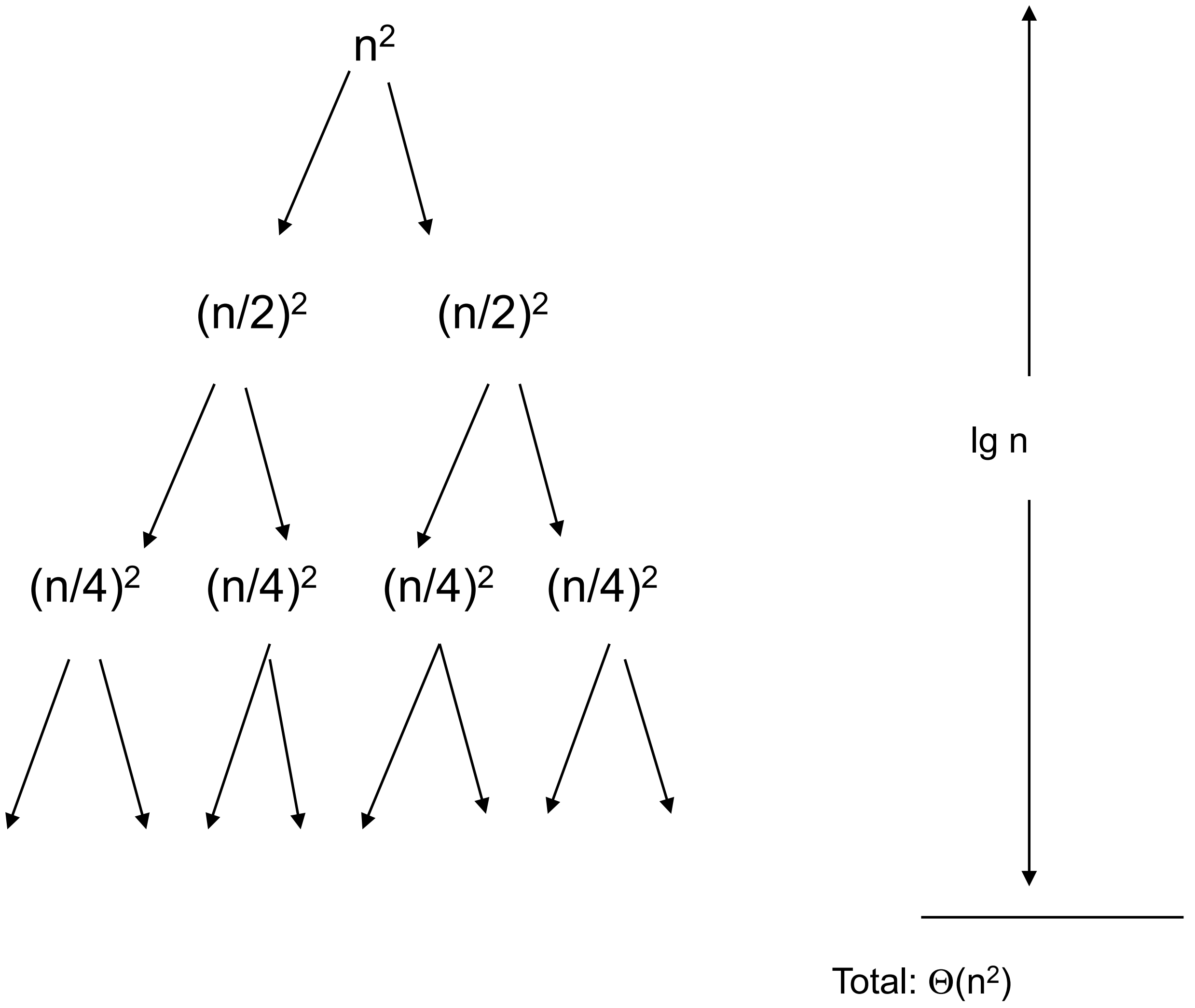
$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}$$

When $x = 1/2$ the series is

$$1 + 1/2 + 1/4 + \ldots. = {}^{1}\!/_{{}^{1}\!/_{2}} = 2$$

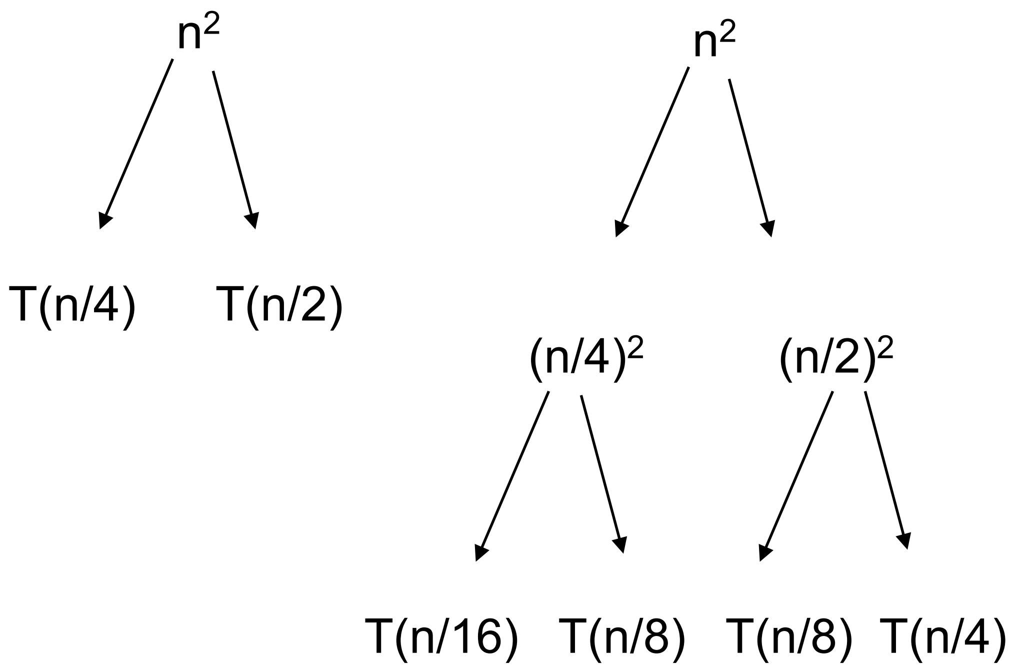
So this is an upper bound on the series in our recurrence.

Recurrence Tree Example



Another Example

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



n^2

$5/16\ n^2$

$25/256\ n^2 = (5/16)^2 n^2$

Since the values decrease geometrically, the total is at most a constant factor more than the largest term and hence the solution is $\Theta(n^2)$

Master Method

- Solving a class of recurrences having the form of

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

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

$f(n)$ is asymptotically positive.

Master Theorem (case 1)

$f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$

- $f(n)$ grows polynomially (by factor n^{ε}) slower than $n^{\log_b a}$

- leaf level work dominates
  - Summation of recursion-tree levels $O(n^{\log_b a})$
  - Cost of all the leaves $\Theta(n^{\log_b a})$
  - The total cost $\Theta(n^{\log_b a})$

Master Theorem (case 2)

if $\quad f(n) = \Theta(n^{\log_b a})$

$f(n)$ and $n^{\log_b a}$ are asymptotically the same

work is distributed equally throughout the tree

(level cost) X (number of levels)

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

Master Theorem (case 3)

$f(n) = \Omega(n^{\log_b a+\varepsilon})$ for some constant $\varepsilon > 0$

- – Inverse of the first case
- – $f(n)$ grows polynomially faster than $n^{\log_b a}$
- – Also need a regularity condition

$\exists c < 1$ and $n_0 > 0$ such that $af(n/b) \le cf(n) \quad \forall n > n_0$

- root work dominates

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

Master Theorem (all cases)

Having a recurrence in the form of

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

1 $f(n) = O\left(n^{\log_b a-\varepsilon}\right) \Rightarrow T(n) = \Theta\left(n^{\log_b a}\right)$

2 $f(n) = \Theta\left(n^{\log_b a}\right) \Rightarrow T(n) = \Theta\left(n^{\log_b a}\log_2 n\right)$

3 $f(n) = \Omega\left(n^{\log_b a+\varepsilon}\right) \quad and \quad af(n/b) \le cf(n),$

$\qquad for \quad \exists c \quad c < 1 \quad and \quad n > n_0$

$\Rightarrow T(n) = \Theta\left(f(n)\right)$

Master Theorem Case 1

$$T(n) = 4T(n/2) + 3n\log_2 n$$

$$f(n) = O\left(n^{\log_b a - \varepsilon}\right) \Rightarrow T(n) = \Theta\left(n^{\log_b a}\right)$$

$T(1) = 1, n \geq 1,$ a power of 2

$$\begin{aligned}
T(n) &= \left(\sum_{i=0}^{(\log_2 n)-1} 4^i \left(3 \cdot \frac{n}{2^i} \log_2 \frac{n}{2^i}\right)\right) + T(1) \cdot 4^{\log_2 n} \\
&= \left(3n \sum_{i=0}^{(\log_2 n)-1} 2^i (\log_2 n - i)\right) + n^2 \\
&= \left(3n \log_2 n \sum_{i=0}^{(\log_2 n)-1} 2^i\right) - \left(3n \sum_{i=0}^{(\log_2 n)-1} i \cdot 2^i\right) + n^2 \\
&= 3n \log_2 n(2^{\log_2 n} - 1) - 3n\left(2^{\log_2 n}(\log_2 n - 2) + 2\right) + n^2 \\
&= 3n^2 \log_2 n - 3n \log_2 n - 3n^2 \log_2 n + 6n^2 - 6n + n^2 \\
&= 7n^2 - 3n \log_2 n - 6n = \Theta(n^2)
\end{aligned}$$

Source: http://www.cs.wustl.edu/~sg/CSE241_SP06/recurrence-exs.pdf

Master Theorem Case 2

$$f(n) = \Theta\left(n^{\log_b a}\right) \Rightarrow T(n) = \Theta\left(n^{\log_b a} \log_2 n\right)$$

$$T(n) = 2T(n/4) + \sqrt{n}$$

$$T(1) = 1, n \geq 1, \text{ a power of } 4$$

$$\begin{aligned} T(n) &= \left(\sum_{i=0}^{(\log_4 n)-1} 2^i \sqrt{n/4^i}\right) + T(1) \cdot 2^{\log_4 n} \\ &= \left(\sum_{i=0}^{(\log_4 n)-1} \sqrt{n}\right) + \sqrt{n} = \sqrt{n} \log_4 n + \sqrt{n} = \Theta(\sqrt{n}) \end{aligned}$$

Source: http://www.cs.wustl.edu/~sg/CSE241_SP06/recurrence-exs.pdf

Master Theorem Case 3

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

$$T(1) = 1, n \ge 1, \text{ a power of } 2$$

$$f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right) \;\; and \;\; af(n/b) \le cf(n),$$
$$for \;\; \exists c \;\; c < 1 \;\; and \;\; n > n_0$$
$$\Rightarrow T(n) = \Theta\left(f(n)\right)$$

$$\begin{aligned}
T(n) &= \left(\sum_{i=0}^{(\log_2 n)-1} 2^i \cdot 3\left(\frac{n}{2^i}\right)^2\right) + T(1) \cdot 2^{\log_2 n} \\
&= \left(3n^2 \sum_{i=0}^{(\log_2 n)-1} 2^i/4^i\right) + 2n \\
&= \left(3n^2 \sum_{i=0}^{(\log_2 n)-1} (1/2)^i\right) + 2n \\
&= 3n^2\left(\frac{1-(1/2)^{\log_2 n}}{1-1/2}\right) + 2n \\
&= 6n^2\left(1-\frac{1}{n}\right) + 2n \\
&= 6n^2 - 4n = \Theta(n^2)
\end{aligned}$$

Source: http://www.cs.wustl.edu/~sg/CSE241_SP06/recurrence-exs.pdf

Summary

- Recurrence Equation: What and why
- Methods to Solve Recurrences:
  - Substitution (constructive induction)
  - Iteration of the recurrence
    - Recurrence Trees
  - Master Theorem