Click to add Text

CSE 301
Combinatorial Optimization

Lecture 2
Recurrence

Today's Topic

- Recurrence
  - Substitution Method
  - Recursive Method
  - Master Method
  - Akra-bazzi method
  - Solving homogeneous equation
  - Solving non-homogeneous equation

Recurrence Relations

- A ***recurrence relation*** is the recursive part of a *recursive definition* of either a number sequence or integer function.

Recursively Defined Sequences

- Fibonacci sequence:
  - $\{f_n\} = 0,1,1,2,3,5,8,13,21,34,55,\ldots$
  - Recursive definition for $\{f_n\}$:
  - INITIALIZE: $f_0 = 0$, $f_1 = 1$
  - RECURSE: $f_n = f_{n-1}+f_{n-2}$ for $n > 1$.
  - The recurrence relation is the recursive part
  - $f_n = f_{n-1}+f_{n-2}$.Thus a recurrence relation for a sequence consists of an equation that expresses each term in terms of lower terms.

Substitution method

The most general method:

1. ***Guess*** the form of the solution.
2. ***Verify*** by induction.
3. ***Solve*** for constants.

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

- Guess $T(n) = O(nlogn)$
- Assume that $T(n/2) <= c(n/2)log(n/2)$
- Prove that $T(n) <= cnlogn$ - by induction

Substitution method

$$
\begin{aligned}
T(n) &<= 2(c(n/2)\log(n/2)) + n \\
&<= cn\log(n/2)) + n \\
&= cn\log n - cn\log 2 + n \\
&= cn\log n - cn + n \\
&<= cn\log n.
\end{aligned}
$$

Evaluate recursive equation using Recursion Tree

- Evaluate: $T(n) = T(n/2) + T(n/2) + n$
  - Work copy: $T(k) = T(k/2) + T(k/2) + k$
  - For $k=n/2$, $T(n/2) = T(n/4) + T(n/4) + (n/2)$
- [size|cost]

Recursion-tree method

- A recursion tree models the costs (time) of a recursive execution of an algorithm.
- The recursion tree method is good for generating guesses for the substitution method.
- The recursion-tree method can be unreliable.
- The recursion-tree method promotes intuition, however.

Recursion Tree e.g.

- To evaluate the total cost of the recursion tree
  - sum all the non-recursive costs of all nodes
  - = Sum (rowSum(cost of all nodes at the same depth))
- Determine the maximum depth of the recursion tree:
  - For our example, at tree depth d
    the size parameter is $n/(2^d)$
  - the size parameter converging to base case, i.e. case 1
  - such that, $n/(2^d) = 1$,
  - $d = \lg(n)$
  - The rowSum for each row is n
- Therefore, the total cost, $T(n) = n \lg(n)$

Example of recursion tree

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

Example of recursion tree

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

$$T(n)$$

Example of recursion tree

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

$$\begin{tikzcd}
 & n^2 \arrow[dl, dash] \arrow[dr, dash] & \\
T(n/4) & & T(n/2)
\end{tikzcd}$$

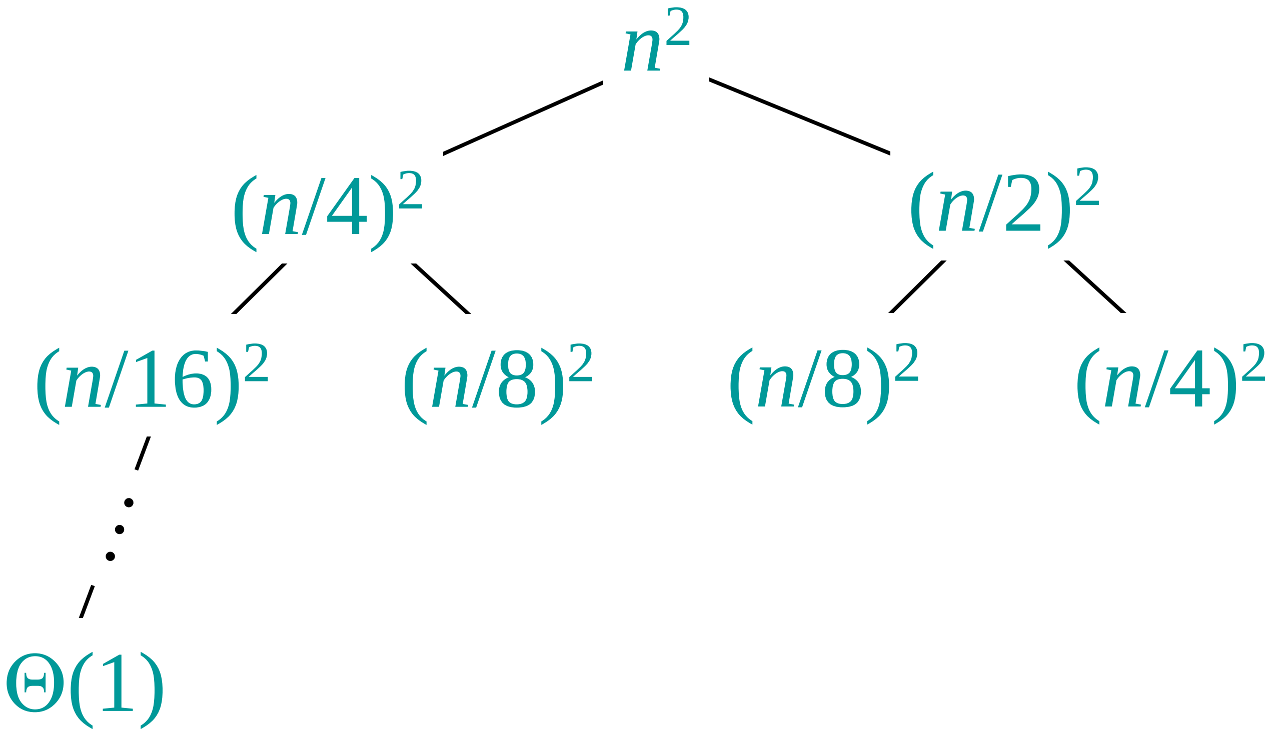
Example of recursion tree

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

$$\begin{tikzcd}
 & & & n^2 \arrow[dll, dash] \arrow[drr, dash] & & & \\
 & (n/4)^2 \arrow[dl, dash] \arrow[dr, dash] & & & & (n/2)^2 \arrow[dl, dash] \arrow[dr, dash] & \\
T(n/16) & & T(n/8) & & T(n/8) & & T(n/4)
\end{tikzcd}$$

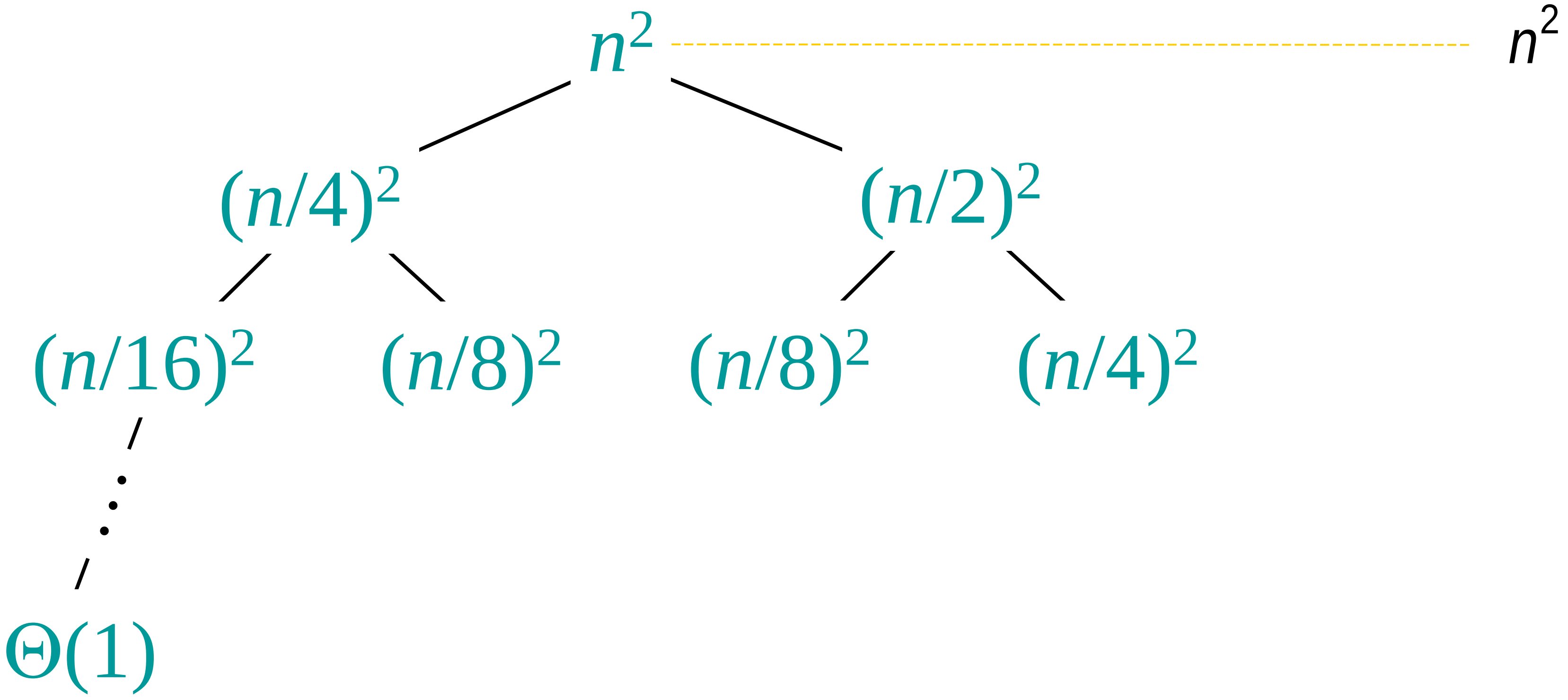
Example of recursion tree

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



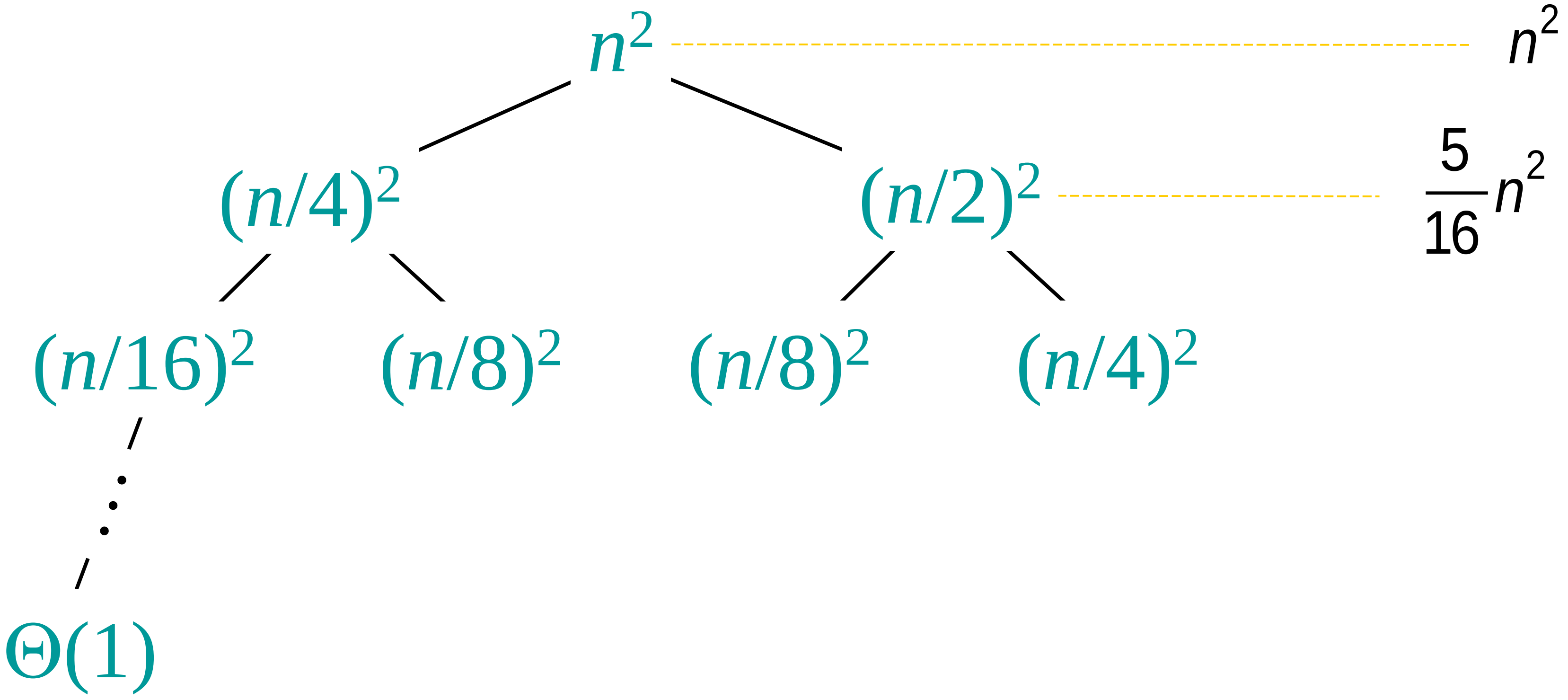
Example of recursion tree

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



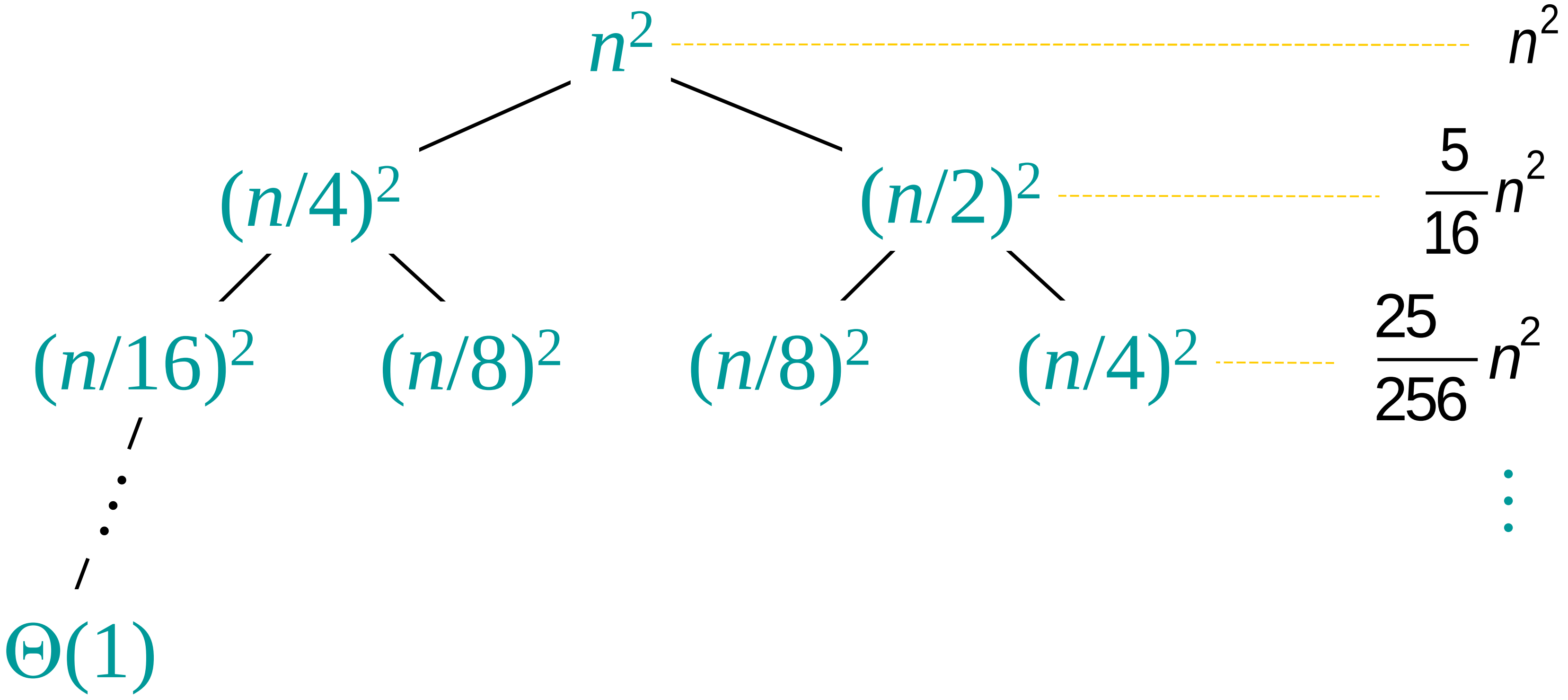
Example of recursion tree

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



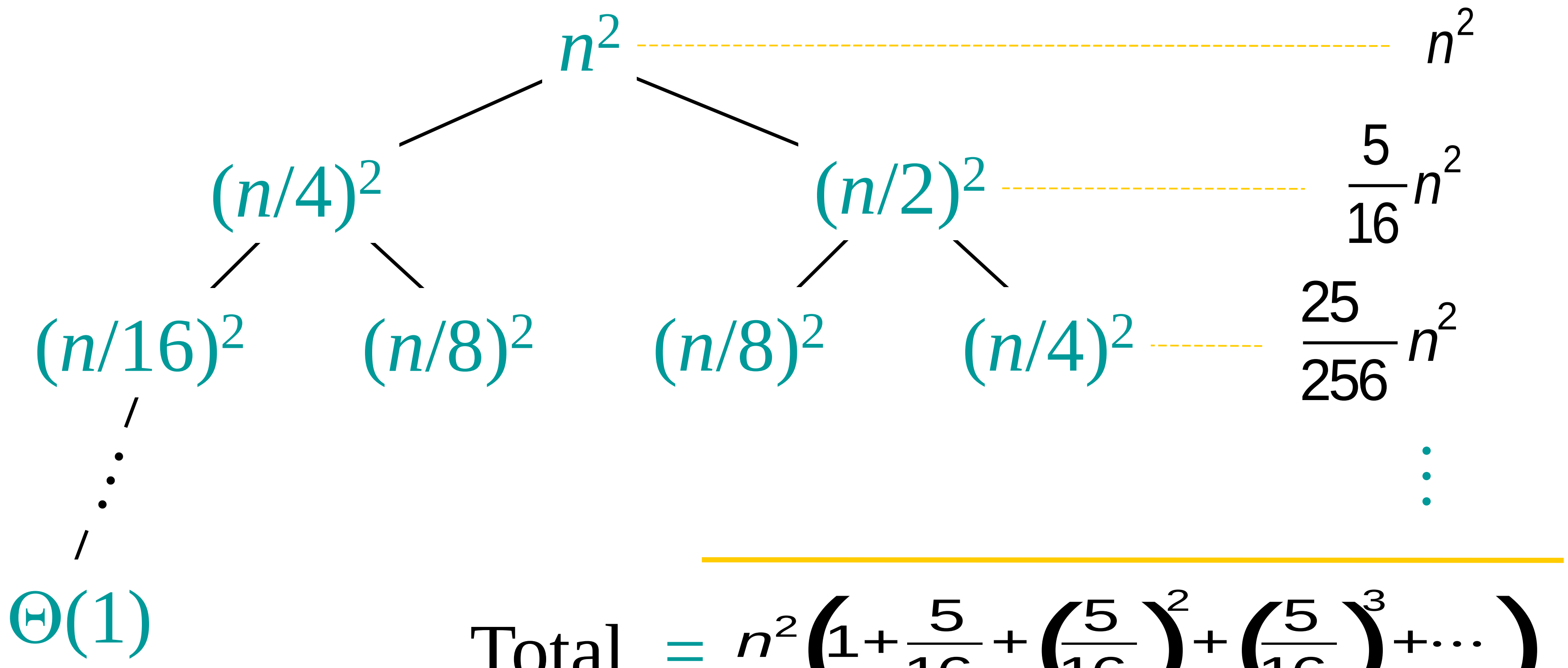
Example of recursion tree

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



Example of recursion tree

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



$$\text{Total} = n^2\left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \cdots\right)$$

$$= \Theta(n^2)$$ *geometric series*

The divide-and-conquer design paradigm

1. *Divide* the problem (instance) into subproblems.
2. *Conquer* the subproblems by solving them recursively.
3. *Combine* subproblem solutions.

Example: merge sort

1. ***Divide:*** Trivial.
2. ***Conquer:*** Recursively sort 2 subarrays.
3. ***Combine:*** Linear-time merge.

$$T(n) = \underbrace{2}_{\textit{\# subproblems}}\, T(\underbrace{n/2}_{\textit{subproblem size}}) + \underbrace{O(n)}_{\textit{work dividing and combining}}$$

The master method

The master method applies to recurrences of the form

$$T(n) = a\,T(n/b) + f(n)\ ,$$

where $a \geq 1$, $b > 1$, and f is asymptotically positive.

Three common cases

Compare $f(n)$ with $n^{\log_b a}$:

1. $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$.
   - $f(n)$ grows polynomially slower than $n^{\log_b a}$ (by an n^{ε} factor).

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

2. $f(n) = \Theta(n^{\log_b a})$.
   - $f(n)$ and $n^{\log_b a}$ grow at similar rates.

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

Three common cases (cont.)

Compare $f(n)$ with $n^{\log_b a}$:

3. $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$.
   - $f(n)$ grows polynomially faster than $n^{\log_b a}$ (by an n^{ε} factor),

   and $f(n)$ satisfies the ***regularity condition*** that $a f(n/b) \le c f(n)$ for some constant $c < 1$.

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

Examples

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

$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n.$

CASE 1: $f(n) = O(n^{2-\varepsilon})$ for $\varepsilon = 1$.

$\therefore T(n) = \Theta(n^2).$

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

$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n^2.$

CASE 2: $f(n) = \Theta(n^2).$

$\therefore T(n) = \Theta(n^2 \lg n).$

Examples

Ex. $T(n) = 4T(n/2) + n^3$
$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n^3$.
CASE 3: $f(n) = \Omega(n^{2+\varepsilon})$ for $\varepsilon = 1$
and $4(cn/2)^3 \le cn^3$ (reg. cond.) for $c = 1/2$.
$\therefore\ T(n) = \Theta(n^3)$.

Ex. $T(n) = 4T(n/2) + n^2/\lg n$
$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n^2/\lg n$.
Master method does not apply. In particular, for every constant $\varepsilon > 0$, we have $n^\varepsilon = \omega(\lg n)$.

General method (Akra-Bazzi)

- The Master method is fairly powerful and results in a closed form solution for divide-and-conquer recurrences with a special form.
- Akra and Bazzi discovered a far more general solution to divide-and-conquer recurrences.

The Akra-Bazzi Method

$$T(x) = \begin{cases} \Theta(1) & \text{for } 1 \le x \le x_0 \\ \sum_{i=1}^{k} a_i T(b_i x) + g(x) & \text{for } x > x_0 \end{cases} \tag{1}$$

where[1]

1. $x \ge 1$ is a real number,
2. x_0 is a constant such that $x_0 \ge 1/b_i$ and $x_0 \ge 1/(1 - b_i)$ for $1 \le i \le k$,
3. $a_i > 0$ is a constant for $1 \le i \le k$,
4. $b_i \in (0, 1)$ is a constant for $1 \le i \le k$,
5. $k \ge 1$ is a constant, and
6. $g(x)$ is a nonnegative function that satisfies the polynomial-growth condition specified below.

Definition. We say that $g(x)$ satisfies the *polynomial-growth condition* if there exist positive constants c_1, c_2 such that for all $x \ge 1$, for all $1 \le i \le k$, and for all $u \in [b_i x, x]$,

$$c_1 g(x) \le g(u) \le c_2 g(x).$$

The Akra-Bazzi Solution

Theorem 1 ([1]). *Given a recurrence of the form specified in Equation 1, let p be the unique real number for which $\sum_{i=1}^{k} a_i b_i^p = 1$. Then*

$$T(x) = \Theta\left(x^p\left(1 + \int_1^x \frac{g(u)}{u^{p+1}}\,du\right)\right).$$

Examples.

- If $T(x) = 2T(x/4) + 3T(x/6) + \Theta(x \log x)$, then $p = 1$ and $T(x) = \Theta(x \log^2 x)$.
- If $T(x) = 2T(x/2) + \frac{8}{9}T(3x/4) + \Theta(x^2/\log x)$, then $p = 2$ and $T(x) = \Theta(x^2/\log\log x)$.
- If $T(x) = T(x/2) + \Theta(\log x)$, then $p = 0$ and $T(x) = \Theta(\log^2 x)$.
- If $T(x) = \frac{1}{2}T(x/2) + \Theta(1/x)$, then $p = -1$ and $T(x) = \Theta((\log x)/x)$.
- If $T(x) = 4T(x/2) + \Theta(x)$, then $p = 2$ and $T(x) = \Theta(x^2)$.

Recurrence Relations for Counting

- Often it is very hard to come up with a closed formula for counting a particular set, but coming up with recurrence relation easier.
- Question:
  - Find a recurrence relation for the number of bit strings of length *n* which contain the string 00.

Recurrence Relations for Counting

A: a_n= #(length *n* bit strings containing 00):

I. If the first n-1 letters contain 00 then so does the string of length n. As last bit is free to choose get contribution of $2a_{n-1}$

II. Else, string must be of the form u00 with u a string of length n-2 not containing 00 and **not ending in 0** (why not?). But the number of strings of length n-3 which don't contain 00 is the total number of strings minus the number that do. Thus get contribution of $2^{n-3}-a_{n-3}$

Solution: $a_n = 2a_{n-1} + 2^{n-3} - a_{n-3}$

Q: What are the initial conditions:

Recurrence Relations for Counting

A: Need to give enough initial conditions to avoid ensure well-definedness. The smallest n for which length is well defined is n=0. Thus the smallest n for which $a_n = 2a_{n-1} + 2^{n-3} - a_{n-3}$ makes sense is n=3. Thus need to give a_0, a_1 and a_2 explicitly.

$a_0 = a_1 = 0$ (strings to short to contain 00)

$a_2 = 1$ (must be 00).

Note: example 6 on p. 313 gives the simpler recursion relation $b_n = b_{n-1} + b_{n-2}$ for strings which do ***not*** contain two consecutive 0's.

Solving Recurrence Relations

- We will learn how to give closed solutions to certain kinds of recurrence relations. Unfortunately, most recurrence relations cannot be solved analytically.
- However, recurrence relations can all be solved quickly by using *dynamic programming*.

Numerical Solutions
Dynamic Programming

Recursion + Lookup Table = Dynamic Programming

Consider a recurrence relation of the form:

$$a_n = f(a_0, a_1, \ldots, a_{n-2}, a_{n-1})$$

Then can always solve the recurrence relation for first *n* values by using following pseudocode:

```
integer-array a(integers n, $a_0$)
  $table_0$ = $a_0$
  for(i = 1 to n)
    $table_i$ = f($table_0$,$table_1$,…,$table_{i-1}$)
  return table
```

Dynamic Program for String Example

Solve $a_n = 2a_{n-1} + 2^{n-3} - a_{n-3}$ up to n=7.
Pseudocode becomes:

```
integer-array a(integer n)
    $table_0$ = $table_1$ = 0
    $table_2$ = 1
    for(i = 3 to n)
     $table_i = 2^{i-3} - table_{i-3} + 2*table_{i-1}$
    return table
```

Dynamic Program for String Example

Solve $a_n = 2a_{n\text{-}1} + 2^{n\text{-}3} - a_{n\text{-}3}$ up to n=7:

i	$2^{i-3}-a_{i-3}+2a_{i-1} = a_i$
0	0
1	0
2	1
3	
4	
5	
6	
7	

Dynamic Program for String Example

Solve $a_n = 2a_{n-1} + 2^{n-3} - a_{n-3}$ up to $n=7$:

i	$2^{i-3} - a_{i-3} + 2a_{i-1} = a_i$
0	0
1	0
2	1
3	$1-0+2\cdot 1 = 3$
4	
5	
6	
7	

Dynamic Program for String Example

Solve $a_n = 2a_{n-1} + 2^{n-3} - a_{n-3}$ up to n=7:

i	$2^{i-3} - a_{i-3} + 2a_{i-1} = a_i$
0	0
1	0
2	1
3	$1-0+2\cdot1 = 3$
4	$2-0+2\cdot3 = 8$
5	
6	
7	

Dynamic Program for String Example

Solve $a_n = 2a_{n-1} + 2^{n-3} - a_{n-3}$ up to n=7:

i	$2^{i-3} - a_{i-3} + 2a_{i-1} = a_i$
0	0
1	0
2	1
3	$1-0+2\cdot1 = 3$
4	$2-0+2\cdot3 = 8$
5	$4-1+2\cdot8 = 19$
6	
7	

Dynamic Program for String Example

Solve $a_n = 2a_{n-1} + 2^{n-3} - a_{n-3}$ up to $n=7$:

i	$2^{i-3} - a_{i-3} + 2a_{i-1} = a_i$
0	0
1	0
2	1
3	$1-0+2\cdot1 = 3$
4	$2-0+2\cdot3 = 8$
5	$4-1+2\cdot8 = 19$
6	$8-3+2\cdot19 = 43$
7	

Dynamic Program for String Example

Solve $a_n = 2a_{n-1} + 2^{n-3} - a_{n-3}$ up to $n=7$:

i	$2^{i-3} - a_{i-3} + 2a_{i-1} = a_i$
0	0
1	0
2	1
3	$1-0+2\cdot1 = 3$
4	$2-0+2\cdot3 = 8$
5	$4-1+2\cdot8 = 19$
6	$8-3+2\cdot19 = 43$
7	$16-8+2\cdot43 = 94$

Closed Solutions by Telescoping

1) Plug recurrence into itself repeatedly for smaller and smaller values of *n*.
2) See the pattern and then give closed formula in terms of initial conditions.
3) Plug values into initial conditions getting final formula.

Telescoping also called ***back-substitution***

Telescope Example

Find a closed solution to $a_n = 2a_{n-1}$, $a_0 = 3$:

$$a_n = 2a_{n-1}$$

Telescope Example

Find a closed solution to $a_n = 2a_{n-1}$, $a_0 = 3$:

$a_n = 2a_{n-1}$ $= 2^2 a_{n-2}$

Telescope Example

Find a closed solution to $a_n = 2a_{n-1}$, $a_0 = 3$:

$$a_n = 2a_{n-1} = 2^2 a_{n-2} = 2^3 a_{n-3}$$

Telescope Example

Find a closed solution to $a_n = 2a_{n-1}$, $a_0 = 3$:

$$a_n = 2a_{n-1} = 2^2 a_{n-2} = 2^3 a_{n-3} = \ldots$$

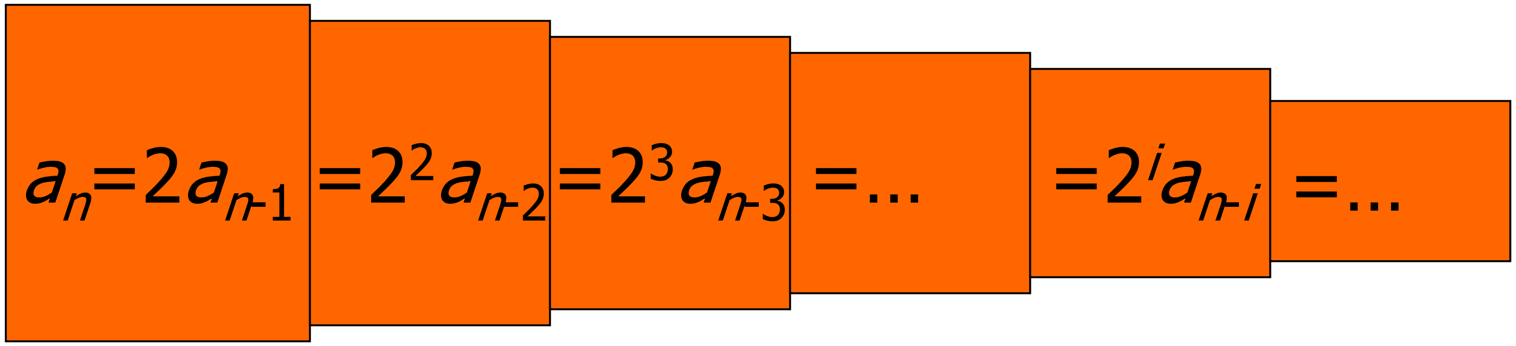
Telescope Example

Find a closed solution to $a_n = 2a_{n-1}$, $a_0 = 3$:

$$a_n = 2a_{n-1} = 2^2 a_{n-2} = 2^3 a_{n-3} = \ldots = 2^i a_{n-i}$$

Telescope Example

Find a closed solution to $a_n = 2a_{n-1}$, $a_0 = 3$:



$$a_n = 2a_{n-1} = 2^2 a_{n-2} = 2^3 a_{n-3} = \ldots = 2^i a_{n-i} = \ldots$$

Telescope Example

Find a closed solution to $a_n = 2a_{n-1}$, $a_0 = 3$:

$$a_n = 2a_{n-1} = 2^2 a_{n-2} = 2^3 a_{n-3} = \ldots = 2^i a_{n-i} = \ldots = 2^n a_0$$

Telescope Example

Find a closed solution to $a_n = 2a_{n-1}$, $a_0 = 3$:

$$a_n = 2a_{n-1} = 2^2 a_{n-2} = 2^3 a_{n-3} = \ldots = 2^i a_{n-i} = \ldots = 2^n a_0$$

Plug in $a_0 = 3$ for final answer:

$$a_n = 3 \cdot 2^n$$

Blackboard Exercise for 5.1

- 5.1.21: Give a recurrence relation for the number of ways to climb *n* stairs if the climber can take one or two stairs at a time.

Linear Recurrences

- The only case for which telescoping works with a high probability is when the recurrence gives the next value in terms of a single previous value. But…
- There is a class of recurrence relations which *can* be solved analytically in general. These are called *linear recurrences* and include the Fibonacci recurrence.
- Begin by showing how to solve Fibonacci:

Solving Fibonacci

Recipe solution has 3 basic steps:

1) Assume solution of the form $a_n = r^n$
2) Find all possible r's that seem to make this work. Call these[1] r_1 and r_2. Modify assumed solution to ***general solution*** $a_n = Ar_1^n + Br_2^n$ where A,B are constants.
3) Use initial conditions to find A,B and obtain specific solution.

Solving Fibonacci

1) Assume exponential solution of the form $a_n = r^n$:

Plug this into $a_n = a_{n-1} + a_{n-2}$:

$$r^n = r^{n-1} + r^{n-2}$$

Notice that all three terms have a common r^{n-2} factor, so divide this out:

$$r^n / r^{n-2} = (r^{n-1} + r^{n-2}) / r^{n-2} \Rightarrow r^2 = r + 1$$

This equation is called the ***characteristic equation*** of the recurrence relation.

Solving Fibonacci

2) Find all possible r's that solve characteristic

$$r^2 = r + 1$$

Call these r_1 and r_2.[1] General solution is $a_n = Ar_1^n + Br_2^n$ where A,B are constants.

Quadratic formula[2] gives:

$$r = (1 \pm \sqrt{5})/2$$

So $r_1 = (1+\sqrt{5})/2$, $r_2 = (1-\sqrt{5})/2$

General solution:

$$a_n = A\,[(1+\sqrt{5})/2]^n + B\,[(1-\sqrt{5})/2]^n$$

Solving Fibonacci

3) Use initial conditions $a_0 = 0$, $a_1 = 1$ to find A,B and obtain specific solution.

$0=a_0 = A\,[(1+\sqrt{5})/2]^0 + B\,[(1-\sqrt{5})/2]^0 = A + B$

$1=a_1 = A\,[(1+\sqrt{5})/2]^1 + B\,[(1-\sqrt{5})/2]^1 = A(1+\sqrt{5})/2 + B\,(1-\sqrt{5})/2 = (A+B)/2 + (A-B)\sqrt{5}/2$

First equation give $B = -A$. Plug into 2nd:

$1 = 0 + 2A\sqrt{5}/2$ so $A = 1/\sqrt{5}$, $B = -1/\sqrt{5}$

Final answer:

(CHECK IT!)

$$a_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

Linear Recurrences with Constant Coefficients

Previous method generalizes to solving "*linear recurrence relations with constant coefficients*":

DEF: A recurrence relation is said to be ***linear*** if a_n is a linear combination of the previous terms plus a function of n. I.e. no squares, cubes or other complicated function of the previous a_i can occur. If in addition all the coefficients are constants then the recurrence relation is said to have ***constant coefficients***.

Linear Recurrences with Constant Coefficients

Q: Which of the following are linear with constant coefficients?

1. $a_n = 2a_{n-1}$
2. $a_n = 2a_{n-1} + 2^{n-3} - a_{n-3}$
3. $a_n = a_{n-1}^2$

Linear Recurrences with Constant Coefficients

A:

1. $a_n = 2a_{n-1}$: YES
2. $a_n = 2a_{n-1} + 2^{n-3} - a_{n-3}$: YES
3. $a_n = a_{n-1}^2$: NO. Squaring is not a linear operation. Similarly $a_n = a_{n-1}a_{n-2}$ and $a_n = \cos(a_{n-2})$ are non-linear.

Homogeneous Linear Recurrences

To solve such recurrences we must first know how to solve an easier type of recurrence relation:

DEF: A linear recurrence relation is said to be ***homogeneous*** if it is a linear combination of the previous terms of the recurrence *without* an additional function of *n*.

Q: Which of the following are homogeneous?

1. $a_n = 2a_{n\text{-}1}$
2. $a_n = 2a_{n\text{-}1} + 2^{n\text{-}3} - a_{n\text{-}3}$

Linear Recurrences with Constant Coefficients

A:

1. $a_n = 2a_{n-1}$: YES
2. $a_n = 2a_{n-1} + 2^{n-3} - a_{n-3}$: No. There's an extra term $f(n) = 2^{n-3}$

Homogeneous Linear Recurrences with Const. Coeff.'s

- The 3-step process used for the Fibonacci recurrence works well for general homogeneous linear recurrence relations with constant coefficients. There are a few instances where some modification is necessary.

Homogeneous -Complications

1) *Repeating roots* in characteristic equation. Repeating roots imply that don't learn anything new from second root, so may not have enough information to solve formula with given initial conditions. We'll see how to deal with this on next slide.
2) *Non-real number roots* in characteristic equation. If the sequence has periodic behavior, may get complex roots (for example $a_n = -a_{n-2}$)[1]. We won't worry about this case (in principle, same method works as before, except use complex arithmetic).

Complication: Repeating Roots

EG: Solve $a_n = 2a_{n-1} - a_{n-2}$, $a_0 = 1$, $a_1 = 2$

Find characteristic equation by plugging in $a_n = r^n$:

$$r^2 - 2r + 1 = 0$$

Since $r^2 - 2r + 1 = (r-1)^2$ the root $r = 1$ repeats.

If we tried to solve by using general solution

$$a_n = Ar_1^n + Br_2^n = A1^n + B1^n = A + B$$

which forces a_n to be a constant function ($\rightarrow\leftarrow$).

SOLUTION: Multiply second solution by n so general solution looks like:

$$a_n = Ar_1^n + Bnr_1^n$$

Complication: Repeating Roots

Solve $a_n = 2a_{n\text{-}1}\text{-}a_{n\text{-}2}$, $a_0 = 1$, $a_1 = 2$

General solution: $a_n = A1^n+Bn1^n = A+Bn$

Plug into initial conditions

$1 = a_0 = A+B\cdot0\cdot1^0= A$

$2 = a_0 = A\cdot1^1+B\cdot1\cdot1^1= A+B$

Plugging first equation $A = 1$ into second:

$2 = 1+B$ implies $B = 1$.

Final answer: $a_n = 1+n$

(CHECK IT!)

The Nonhomogeneous Case

Consider the Tower of Hanoi recurrence (see Rosen p. 311-313) $a_n = 2a_{n-1}+1$.

Could solve using telescoping. Instead let's solve it methodically. Rewrite:

$$a_n - 2a_{n-1} = 1$$

1) Solve with the RHS set to 0, i.e. solve the homogeneous case.

2) Add a particular solution to get general solution. I.e. use rule:

$$\boxed{\text{General Nonhomogeneous}} = \boxed{\text{General homogeneous}} + \boxed{\text{Particular Nonhomogeneous}}$$

The Nonhomogeneous Case

$$a_n - 2a_{n-1} = 1$$

1) Solve with the RHS set to 0, i.e. solve

$$a_n - 2a_{n-1} = 0$$

Characteristic equation: $r - 2 = 0$

so unique root is $r = 2$. General solution to homogeneous equation is

$$a_n = A \cdot 2^n$$

The Nonhomogeneous Case

2) Add a particular solution to get general solution for a_n - $2a_{n\text{-}1}$ = 1. Use rule:

$$\boxed{\text{General Nonhomogeneous}} = \boxed{\text{General homogeneous}} + \boxed{\text{Particular Nonhomogeneous}}$$

There are little tricks for guessing particular nonhomogeneous solutions. For example, when the RHS is constant, the guess should also be a constant.[1]

So guess a particular solution of the form $b_n=C$.

Plug into the original recursion:

$1 = b_n - 2b_{n\text{-}1} = C - 2C = \text{-}C$. Therefore $C = \text{-}1$.

General solution: $a_n = A\cdot 2^n - 1$.

The Nonhomogeneous Case

Finally, use initial conditions to get closed solution. In the case of the Towers of Hanoi recursion, initial condition is:

$$a_1 = 1$$

Using general solution $a_n = A\cdot 2^n - 1$ we get:

$1 = a_1 = A\cdot 2^1 - 1 = 2A - 1$.

Therefore, $2 = 2A$, so $A = 1$.

Final answer: $a_n = 2^n - 1$

More Complicated

EG: Find the general solution to recurrence from the bit strings example: $a_n = 2a_{n-1} + 2^{n-3} - a_{n-3}$

1) Rewrite as $a_n - 2a_{n-1} + a_{n-3} = 2^{n-3}$ and solve homogeneous part:

Characteristic equation: $r^3 - 2r + 1 = 0$.

Guess root $r = \pm 1$ as integer roots divide.

$r = 1$ works, so divide out by $(r - 1)$ to get

$r^3 - 2r + 1 = (r - 1)(r^2 + r - 1)$.

More Complicated

$r^3 - 2r + 1 = (r - 1)(r^2 + r - 1)$.

Quadratic formula on $r^2 + r - 1$:

$$r = (-1 \pm \sqrt{5})/2$$

So $r_1 = 1$, $r_2 = (-1+\sqrt{5})/2$, $r_3 = (-1-\sqrt{5})/2$

General homogeneous solution:

$$a_n = A + B\,[(-1+\sqrt{5})/2]^n + C\,[(-1-\sqrt{5})/2]^n$$

More Complicated

2) Nonhomogeneous particular solution to $a_n - 2a_{n-1} + a_{n-3} = 2^{n-3}$

Guess the form $b_n = k\,2^n$. Plug guess in:

$$k\,2^n - 2k\,2^{n-1} + k\,2^{n-3} = 2^{n-3}$$

Simplifies to: $k = 1$.

So particular solution is $b_n = 2^n$

General Nonhomogeneous	=	General homogeneous	+	Particular Nonhomogeneous

Final answer:

$a_n = A + B\,[(-1+\sqrt{5})/2]^n + C\,[(-1-\sqrt{5})/2]^n + 2^n$