Recursion

Algorithm : Design & Analysis

[3]

In the last class…

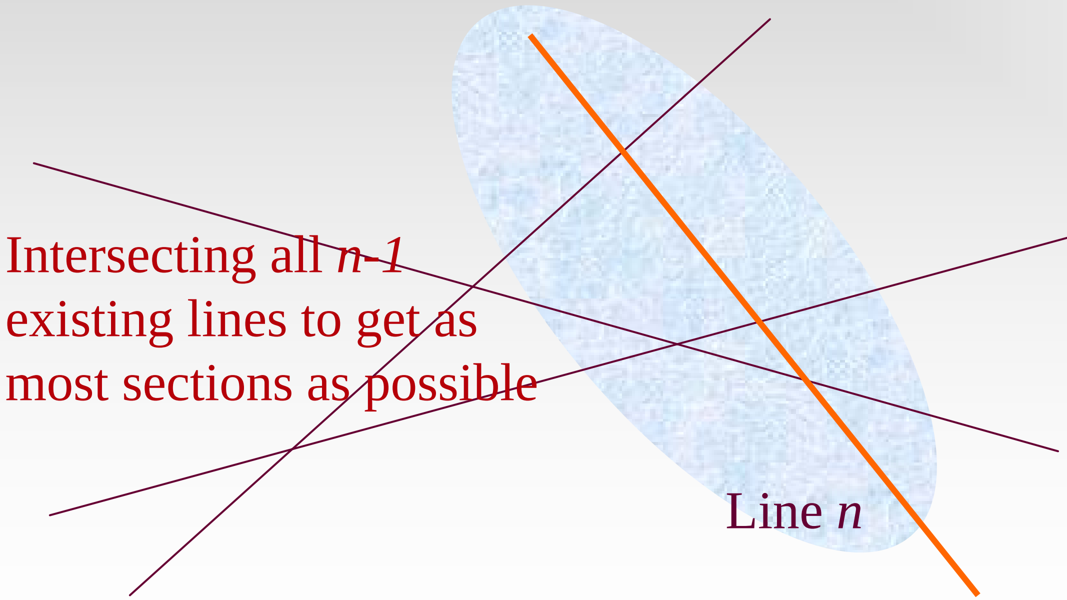
- Asymptotic growth rate
- The Sets O, Ω and Θ
- Complexity Class
- An Example: Maximum Subsequence Sum
  - Improvement of Algorithm
  - Comparison of Asymptotic Behavior
- Another Example: Binary Search
  - Binary Search Is Optimal

Recursion

- Recursive Procedures
- Proving Correctness of Recursive Procedures
- Deriving recurrence equations
- Solution of the Recurrence equations
  - Guess and proving
  - Recursion tree
  - Master theorem
- Divide-and-conquer

Recursion as a Thinking Way

- Cutting the plane
  - How many sections can be generated at most by *n* straight lines with infinite length.



$$L(0) = 1$$
$$L(n) = L(n-1) + n$$

Recursion for Algorithm

Recurrence relations

- Computing n!
  - if n=1 then return 1 else return Fac(n-1)*n

    $M(1)=0$ and $M(n)=M(n-1)+1$ for $n>0$

    (critical operation: multiplication)

- Hanoi Tower
  - if n=1 then move d(1) to peg3 else

    { Hanoi(n-1, peg1, peg2); move d(n) to peg3; Hanoi(n-1, peg2, peg3)

    $M(1)=1$ and $M(n)=2M(n-1)+1$ for $n>1$

    (critical operation: move)

Counting the Number of Bit

- Input: a positive decimal integer *n*
- Output: the number of binary digits in *n*'s binary representation

```
Int BinCounting (int n)
1.  if (n==1) return 1;
2.  else
3.  return BinCounting(n div 2)+1;
```

Correctness of BinCounting

- Proof by induction
- Base case: if n =1, trivial by line 1.
- Inductive hypothesis: for any $0<k<n$, BinCounting(k) return the correct result.
- Induction
  - If $n \neq 1$ then line 3 is excuted
  - (n div 2) is a positive decimal integer (so, the precondition for BinCounting is still hold), and
  - $0<(n \text{ div } 2)<n$, so, the inductive hypothesis applies
  - So, the correctness (the number of bit of n is one more the that of (n div 2)

Complexity Analysis of BinCounting

- The critical operation: addition
- The recurrence relation

$$T(n) = \begin{cases} 0 & n = 1 \\ T(\lfloor n/2 \rfloor) + 1 & n > 1 \end{cases}$$

Solution by backward substitutions

By the recursion equation: $T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 1$

For simplicity, let $n = 2^k$ (k is a nonnegative integer),

that is, $k = \log n$

$$T(n) = T\left(\frac{n}{2}\right) + 1 = T\left(\frac{n}{4}\right) + 1 + 1 = T\left(\frac{n}{8}\right) + 1 + 1 + 1 = \ldots\ldots$$

$$T(n) = T\left(\frac{n}{2^k}\right) + \log n = \log n \qquad (\ T(1)=0\ )$$

Smooth Functions

- Let $f(n)$ be a nonnegative eventually non-decreasing function defined on the set of natural numbers, $f(n)$ is called **smooth** if $\boldsymbol{f(2n) \in \Theta(f(n))}$.
- Note: $\log n$, n, $n\log n$ and n^α ($\alpha \geq 0$) are all smooth.
  - For example: $2n\log 2n = 2n(\log n + \log 2) \in \Theta(n\log n)$

Even Smoother

- Let $f(n)$ be a smooth function, then, for any fixed integer $b \geq 2$, $f(bn) \in \Theta(f(n))$.
- That is, there exist positive constants c_b and d_b and a nonnegative integer n_0 such that

$$d_b f(n) \leq f(bn) \leq c_b f(n) \quad \text{for } n \geq n_0.$$

It is easy to prove that the result holds for $b = 2^k$,

for the second inequality :

$f(2^k n) \leq c_2^k f(n)$ for $k = 1,2,3\ldots$ and $n \geq n_0$.

For an arbitrary integer $b \geq 2$, $2^{k-1} \leq b \leq 2^k$

Then, $f(bn) \leq f(2^k n) \leq c_2^k f(n)$, we can use c_2^k as c_b.

Smoothness Rule

- Let $T(n)$ be an eventually nondecreasing function and $f(n)$ be a smooth function. If $T(n) \in \Theta(f(n))$ for values of n that are powers of b ($b \geq 2$), then $T(n) \in \Theta(f(n))$.

Just proving the big-Oh part:

By the hypothsis: $T(b^k) \leq cf(b^k)$ for $b^k \geq n_0$.

By the prior result: $f(bn) \leq c_b f(n)$ for $n \geq n_0$.

Let $n_0 \leq b^k \leq n \leq b^{k+1}$,

$$T(n) \leq T(b^{k+1}) \leq cf(b^{k+1}) = cf(bb^k) \leq cc_b f(b^k) \leq cc_b f(n)$$

Non-decreasing hypothesis Prior result Non-decreasing

Computing the *n*th Fibonacci Number

$f_1=0$

$f_2=1$

$f_n= f_{n-1}+ f_{n-2}$

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

$$a_n = r_1 a_{n-1} + r_2 a_{n-2} + \cdots + r_m a_{n-k}$$

is called linear homogeneous relation of degree *k*.

For the special case of Fibonacci: $a_n=a_{n-1}+a_{n-2}$, $r_1=r_2=1$

Characteristic Equation

- For a linear homogeneous recurrence relation of degree *k*

$$a_n = r_1 a_{n-1} + r_2 a_{n-2} + \cdots + r_k a_{n-k}$$

  the polynomial of degree *k*

$$x^k = r_1 x^{k-1} + r_2 x^{k-2} + \cdots + r_k$$

  is called its characteristic equation.

- The characteristic equation of linear homogeneous recurrence relation of degree 2 is:

$$\boxed{x^2 - r_1 x - r_2 = 0}$$

Solution of Recurrence Relation

- If the characteristic equation $x^2 - r_1 x - r_2 = 0$ of the recurrence relation $a_n = r_1 a_{n-1} + r_2 a_{n-2}$ has two distinct roots s_1 and s_2, then

$$a_n = u s_1^n + v s_2^n$$

  where u and v depend on the initial conditions, is the explicit formula for the sequence.

- If the equation has a single root s, then, both s_1 and s_2 in the formula above are replaced by s

Proof of the Solution

Remember the equation : $\underline{x^2 - r_1 x - r_2 = 0}$

We need prove that : $\underline{us_1^n + vs_2^n = r_1 a_{n-1} + r_2 a_{n-2}}$

$$
\begin{aligned}
us_1^n + vs_2^n &= us_1^{n-2} s_1^2 + vs_2^{n-2} s_2^2 \\
&= us_1^{n-2}(r_1 s_1 + r_2) + vs_2^{n-2}(r_1 s_2 + r_2) \\
&= r_1 us_1^{n-1} + r_2 us_1^{n-2} + r_1 vs_2^{n-1} + r_2 vs_2^{n-2} \\
&= r_1(us_1^{n-1} + vs_2^{n-1}) + r_2(us_1^{n-2} + vs_2^{n-2}) \\
&= r_1 a_{n-1} + r_2 a_{n-2}
\end{aligned}
$$

Return to Fibonacci Sequence

$f_1=1$

$f_2=1$

$f_n = f_{n-1} + f_{n-2}$

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ……

Explicit formula for Fibonacci Sequence

The characteristic equation is $x^2-x-1=0$, which has roots:

$$s_1 = \frac{1+\sqrt{5}}{2} \quad and \quad s_2 = \frac{1-\sqrt{5}}{2}$$

Note: (by initial conditions) $f_1 = us_1 + vs_2 = 1 \;\; and \;\; f_2 = us_1{}^2 + vs_2{}^2 = 1$

which results:

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

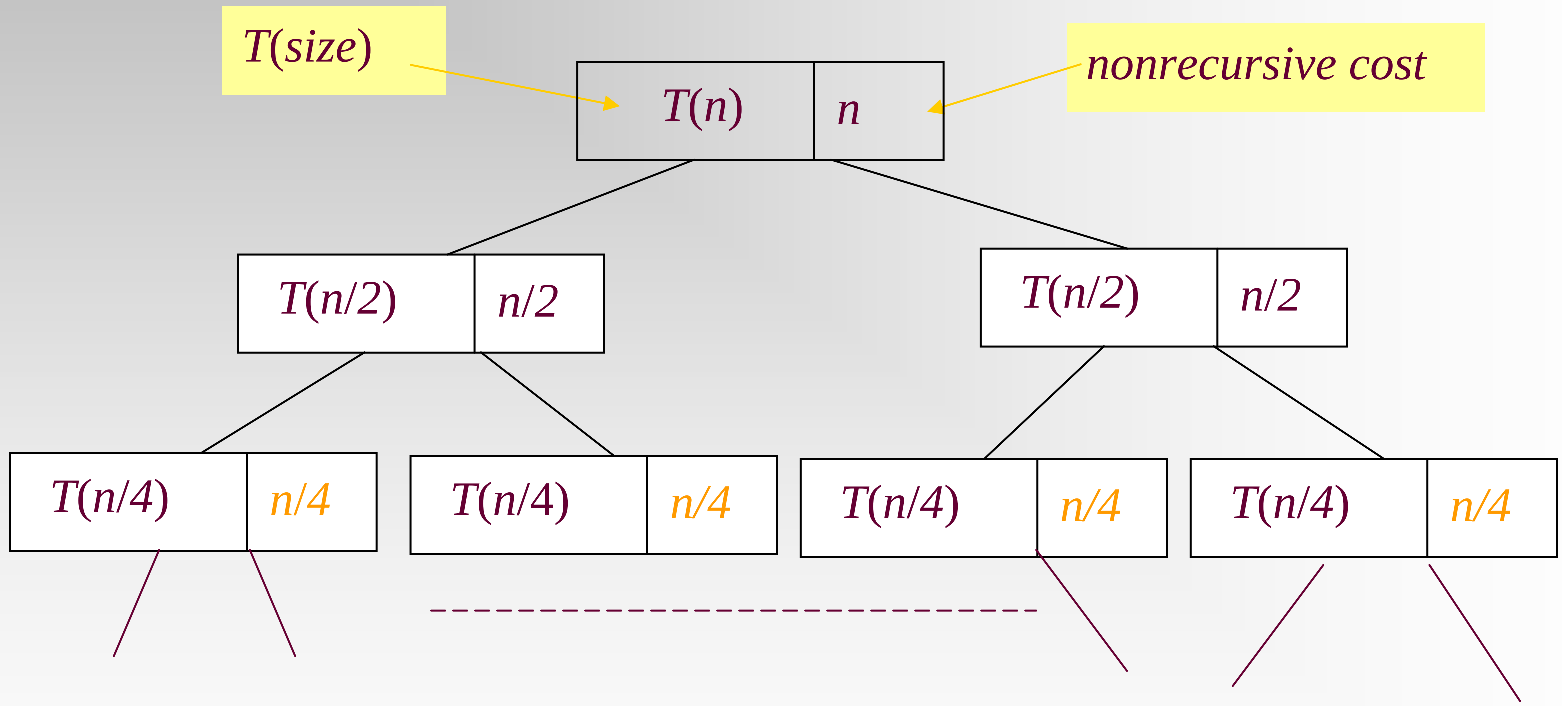
Guess the Solutions

- Example: $T(n)=2T(\lfloor n/2 \rfloor) + n$
- Guess
  - $T(n) \in O(n)$?
    - $T(n) \leq cn$, to be prov
  - $T(n) \in O(n^2)$?
    - $T(n) \leq cn^2$, to be prov
  - **Or maybe**, $T(n) \in O(n\log n)$?
    - $T(n) \leq cn\log n$, to be proved for c large enough

Try to prove $T(n) \leq cn$:

$$
\begin{aligned}
T(n) &= 2T(\lfloor n/2 \rfloor) + n \\
&\leq 2(c\lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor)) + n \\
&\leq cn\log(n/2) + n \\
&= cn\log n - cn\log 2 + n \\
&= cn\log n - cn + n \\
&\leq cn\log n \quad \text{for } c \geq 1
\end{aligned}
$$

Recursion Tree



The recursion tree for T(n)=T(n/2)+T(n/2)+n

Recursion Tree Rules

- Construction of a recursion tree
  - work copy: use auxiliary variable
  - root node
  - expansion of a node:
    - recursive parts: children
    - nonrecursive parts: nonrecursive cost
  - the node with base-case size

Recursion tree equation

- For any subtree of the recursion tree,
  - size field of root =

    Σ nonrecursive costs of expanded nodes +

    Σ size fields of incomplete nodes
- Example: divide-and-conquer:

$$T(n) = bT(n/c) + f(n)$$

  - After kth expansion:

$$T(n) = b^k T\left(\frac{n}{c^k}\right) + \sum_{i=1}^{k-1} b^{i-1} f\left(\frac{n}{c^{i-1}}\right)$$

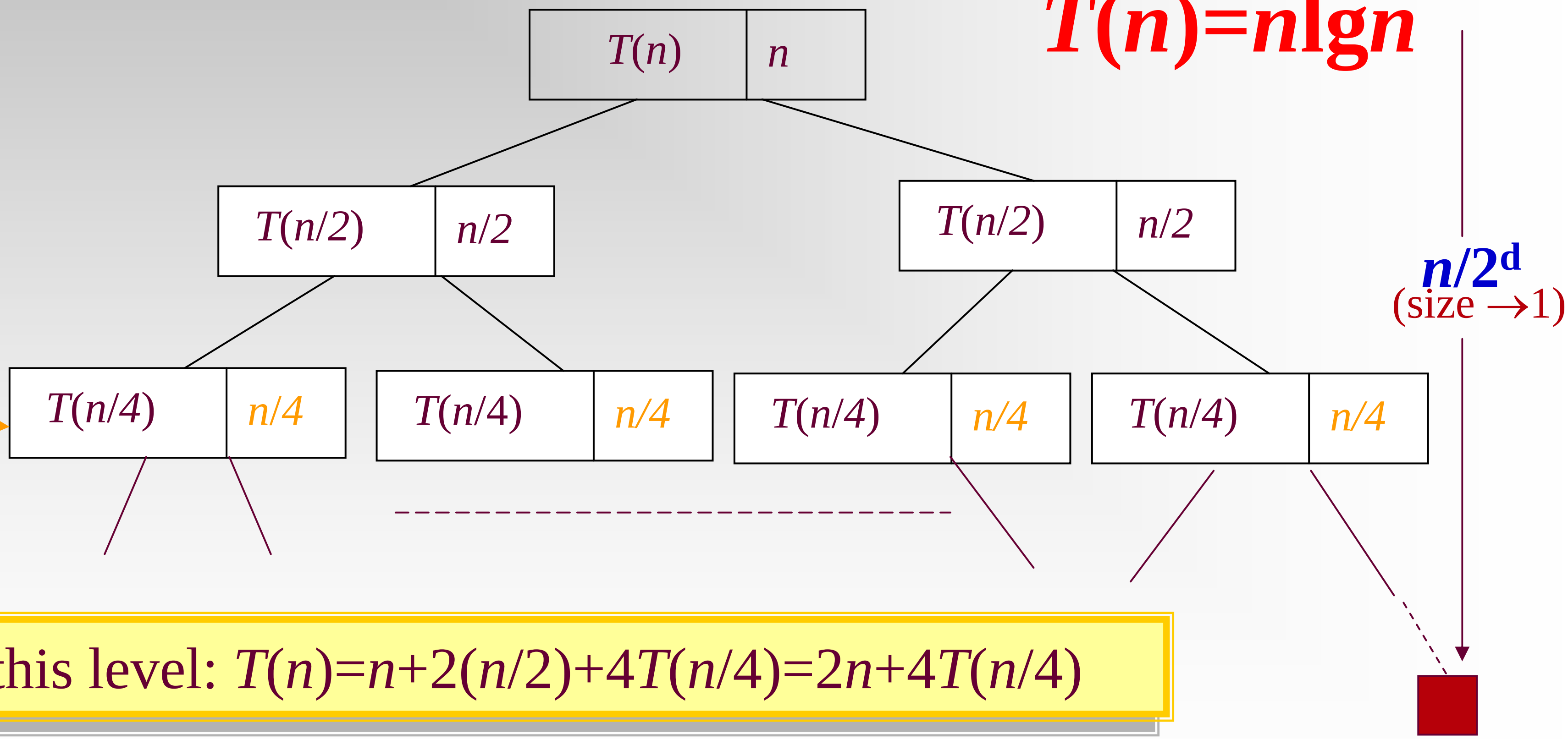
Evaluation of a Recursion Tree

- Computing the sum of the ***nonrecursive costs*** of all nodes.
- Level by level through the tree down.
- Knowledge of the maximum depth of the recursion tree, that is the depth at which the size parameter reduce to a base case.

Recursion Tree

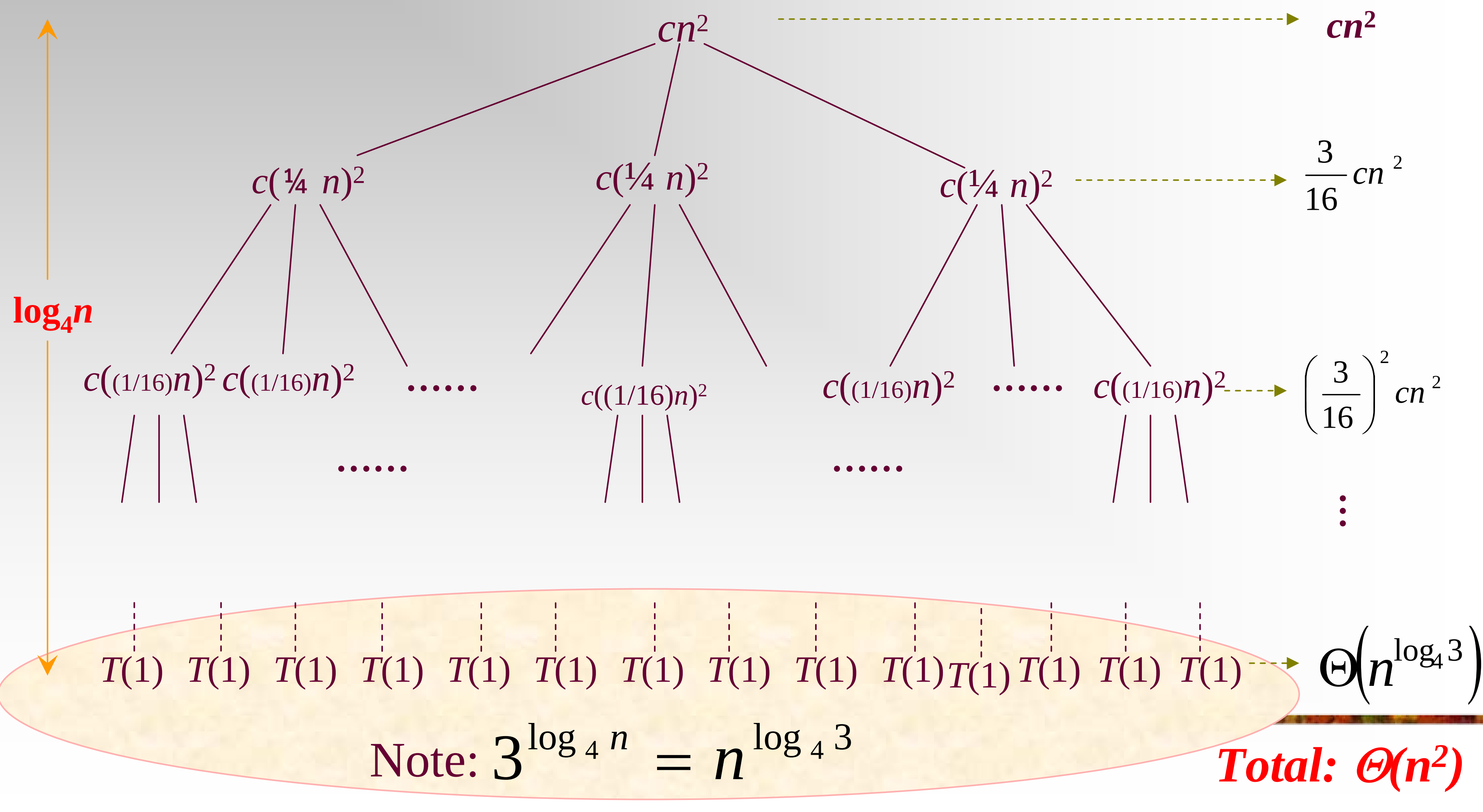
Work copy: T(k)=T(k/2)+T(k/2)+k

$\boldsymbol{T(n)=n\lg n}$



At this level: $T(n)=n+2(n/2)+4T(n/4)=2n+4T(n/4)$

Recursion Tree for $T(n)=3T(\lfloor n/4 \rfloor)+\Theta(n^2)$



Verifying "Guess" by Recursive Tree

$$T(n) = \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3})$$

$$< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3})$$

$$= \frac{1}{1 - \left(\frac{3}{16}\right)} cn^2 + \Theta(n^{\log_4 3})$$

$$= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) = O(n^2)$$

$$T(n) \le 3T(\lfloor n/4 \rfloor) + cn^2$$

$$\le 3d\lfloor n/4 \rfloor^2 + cn^2$$

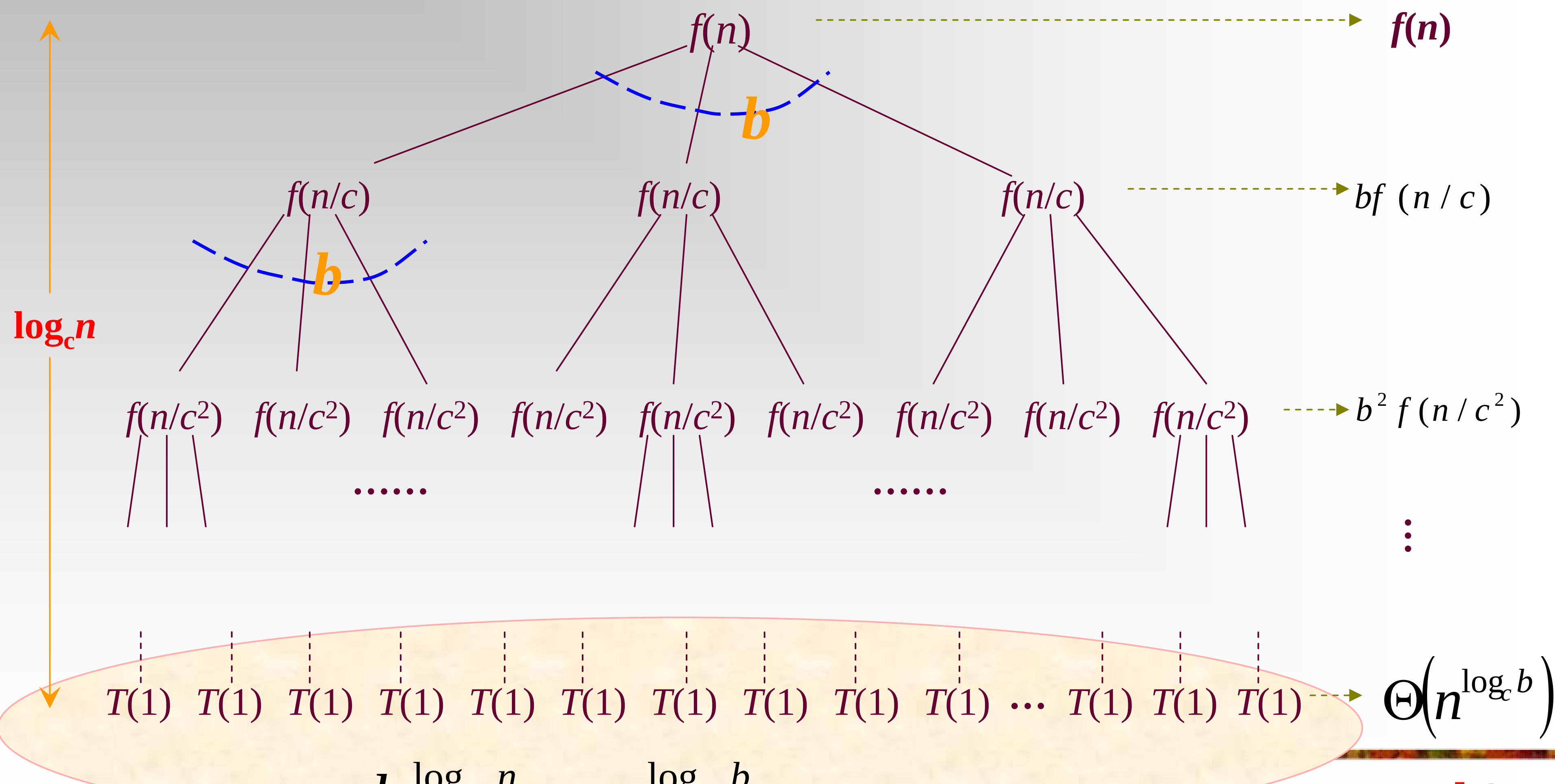
$$\le 3d(n/4)^2 + cn^2$$

$$= \frac{3}{16} dn^2 + cn^2$$

$$\le dn^2 \quad when\ \ d \ge \frac{16}{13}c$$

Inductive hypothesis

Recursion Tree for $T(n)=bT(n/c)+f(n)$



Note: $b^{\log_c n} = n^{\log_c b}$

Solving the Divide-and-Conquer

- The recursion equation for divide-and-conquer, the general case: $T(n)=bT(n/c)+f(n)$
- Observations:
  - Let base-cases occur at depth D(leaf), then $n/c^D=1$, that is $D=\lg(n)/\lg(c)$
  - Let the number of leaves of the tree be L, then $L=b^D$, that is $L=b^{(\lg(n)/\lg(c))}$.
  - By a little algebra: $L=n^E$, where $E=\lg(b)/\lg(c)$, called *critical exponent*.

$$L = b^{\frac{\lg n}{\lg c}} = 2^{\lg b \frac{\lg n}{\lg c}} = 2^{\frac{\lg n}{\lg c}\lg b} = 2^{\lg n \frac{\lg b}{\lg c}}$$

Divide-and-Conquer: the Solution

- The recursion tree has depth $D = \lg(n)/\lg(c)$, so there are about that many row-sums.
- The 0^{th} row-sum is $f(n)$, the nonrecursive cost of the root.
- The Dth row-sum is n^E, assuming base cases cost 1, or $\Theta(n^E)$ in any event.
- The solution of divide-and-conquer equation is the non-recursive costs of all nodes in the tree, which is the sum of the row-sums.

Solution by Row-sums

- [Little Master Theorem] Row-sums decide the solution of the equation for divide-and-conquer:
  - Increasing geometric series: $T(n) \in \Theta(n^E)$
  - Constant: $T(n) \in \Theta(f(n) \log n)$
  - Decreasing geometric series: $T(n) \in \Theta(f(n))$

This can be generalized to get a result not using explicitly row-sums.

Master Theorem

The positive ε is critical, resulting gaps between cases as well

- Loosening the restrictions on $f(n)$
  - Case 1: $f(n) \in O(n^{E-\varepsilon})$, $(\varepsilon > 0)$, then:
    $$T(n) \in \Theta(n^E)$$
  - Case 2: $f(n) \in \Theta(n^E)$, as all node depth contribute about equally:
    $$T(n) \in \Theta(f(n)\log(n))$$
  - case 3: $f(n) \in \Omega(n^{E+\varepsilon})$, $(\varepsilon > 0)$, and $f(n) \in O(n^{E+\delta})$, $(\delta \geq \varepsilon)$, then:
    $$T(n) \in \Theta(f(n))$$

Using Master Theorem

$$\textit{Example} \quad 1 \quad T(n) = 9T\left(\frac{n}{3}\right) + n$$

$$b = 9,\ c = 3,\ E = 2,\ f(n) = n = O(n^{E-1}),$$

$$\textit{case}\ \ 1\ \textit{applies}\ ,\ T(n) = \Theta(n^2)$$

$$\textit{Example} \quad 2 \quad T(n) = T\left(\frac{2n}{3}\right) + 1$$

$$b = 1,\ c = \frac{3}{2},\ E = 0,\ f(n) = 1 = \Theta(n^0),$$

$$\textit{case}\ \ 2\ \textit{applies}\ ,\ T(n) = \Theta(\lg n)$$

Using Master Theorem

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

$$b = 3, c = 4, E = \log_4 3 \approx 0.793,$$

$$f(n) = n\lg n = \Omega(n^{E+0.21}) = O(n^{E+1.21})$$

$$case\ 3\ applies, T(n) = \Theta(n\lg n)$$

Looking at the Gap

- $T(n)=2T(n/2)+n\lg n$
  - $a=2$, $b=2$, $E=1$, $f(n)=n\lg n$
  - We have $f(n)=\Omega(n^E)$, but no $\varepsilon>0$ satisfies $f(n)=\Omega(n^{E+\varepsilon})$, since $\lg n$ grows slower that n^ε for any small positive ε.
  - So, case 3 doesn't apply.
  - However, neither case 2 applies.

Home Assignment

- pp.143-
  - 3.4
  - 3.6
  - 3.9-3.11