Introduction to

Algorithm Design and Analysis

[3] Recursion

Yu Huang
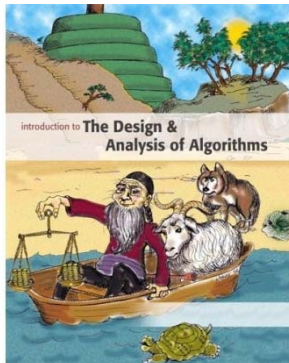
http://cs.nju.edu.cn/yuhuang

Institute of Computer Software

Nanjing University

In the Last Class ...

- **Asymptotic growth rate**
  - O, Ω, Θ
  - o, ω

- **Brute force algorithms**
  - By iteration
  - By recursion

Recursion

- **Recursion in algorithm design**
  - The divide and conquer strategy
  - Proving the correctness of recursive procedures

- **Solving recurrence equations**
  - Some elementary techniques
  - Master theorem

Recursion in Algorithm Design

- **Computing n! with Fac(n)**
  - o 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**
  - o 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)

Recursion in Algorithm Design

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

Int BitCounting (int n)
1. If(n==1) return 1;
2. Else
3.   return BitCounting(n div 2) +1;

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

Divide and Conquer

- **Divide**
  - Divide the "big" problem to smaller ones

- **Conquer**
  - Solve the "small" problems by recursion

- **Combine**
  - Combine results of small problems, and solve the original problem

Divide and Conquer

The general pattern
solve(*I*)

   n=**size**(*I*);

   if (*n*≤smallSize)

      solution=**directlySolve**(*I*)

   else

      divide *I* into $I_1, \dots I_k$;

      for each $i \in \{1,\dots,k\}$

         S_i=**solve**(I_i);

      solution=**combine**($S_1 ,\dots ,S_k$);

 return solution

$$T(n)=B(n) \text{ for } n \leq smallSize$$

$$T(n)=D(n)+\sum_{i=1}^{k}T(size(I_i))+C(n)$$

$$\text{for } n>smallSize$$

Examples

Max sum subsequence

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

Part 1 | Part 2
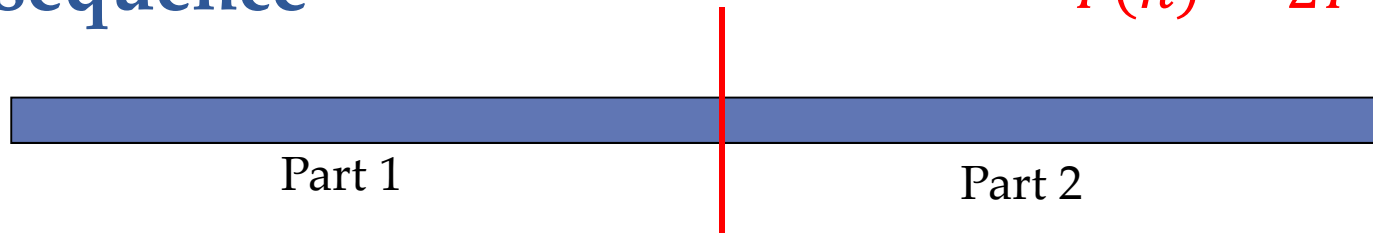
the sub with largest sum may be in:

recursion

Part 1 | Part 2

or:

The largest is the result

Part 1 | Part 2
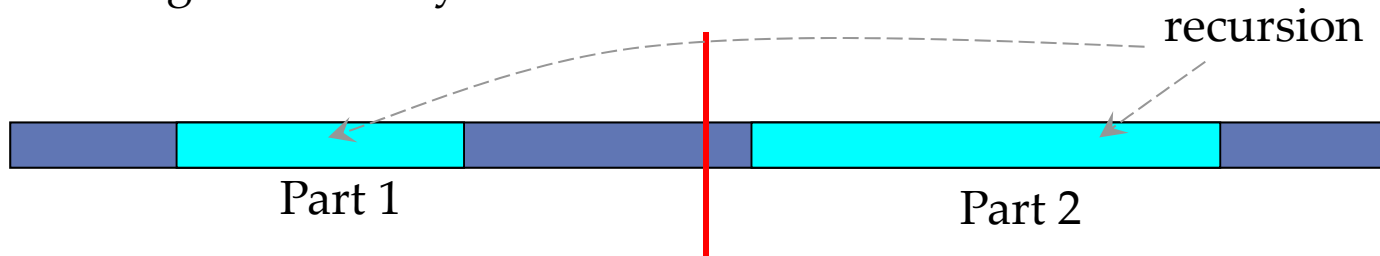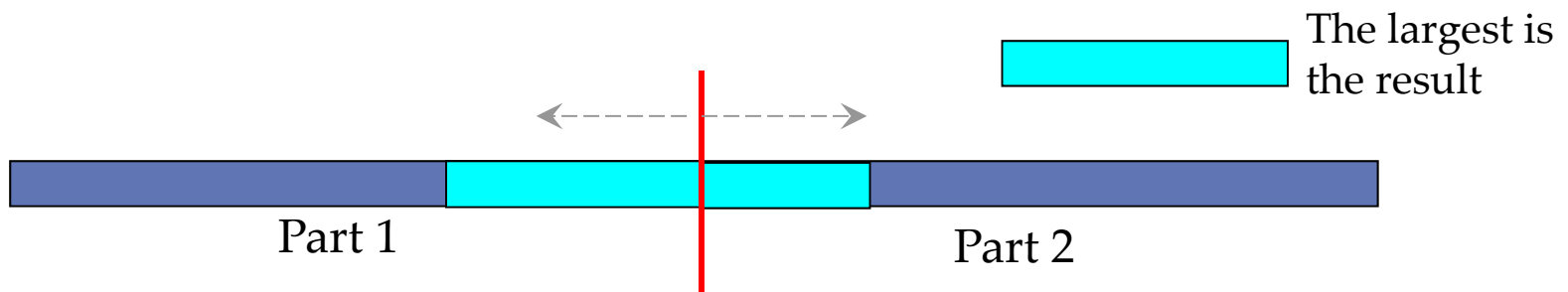
Examples

- **Arrays**
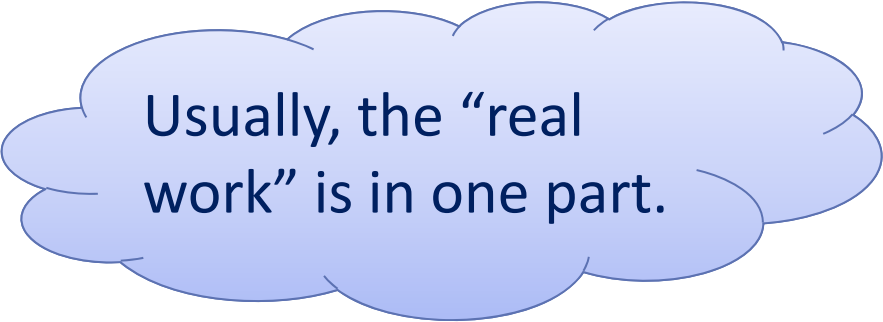
3	5	7	8	9	12	15

- **Trees**
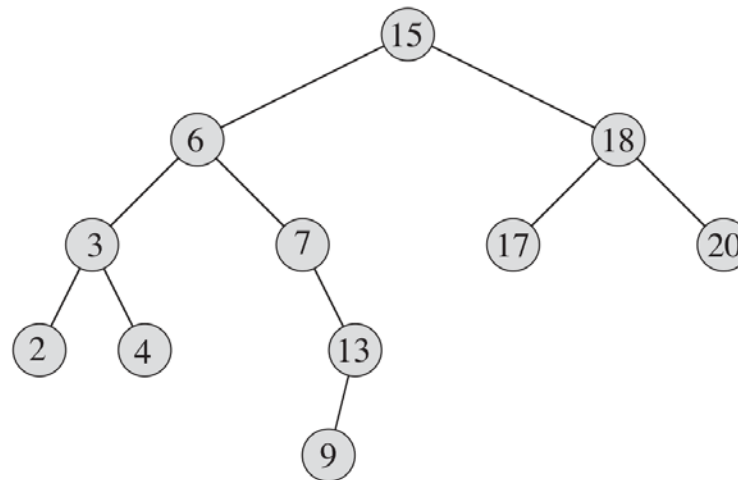
Workhorse

- "Hard division, easy combination"

- "Easy division, hard combination"

Usually, the "real work" is in one part.

Correctness of Recursion

Recursion

...

Case n=k+1

Case n=k

...

Case n=2

Base case n=1
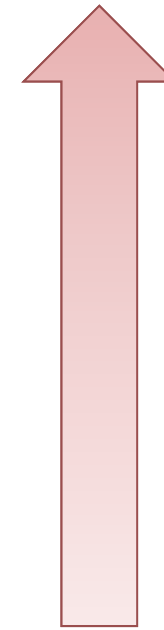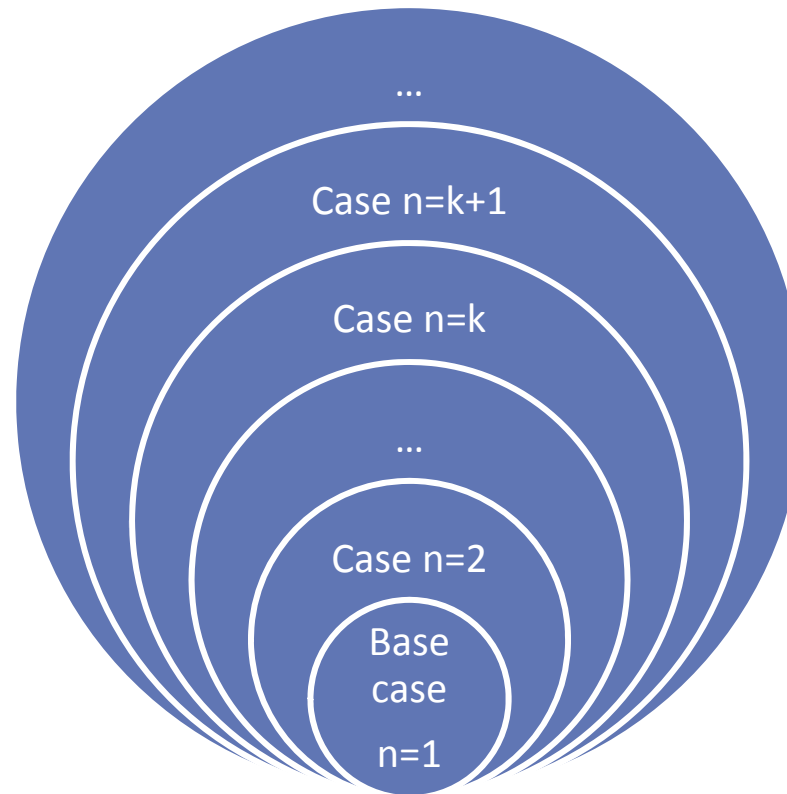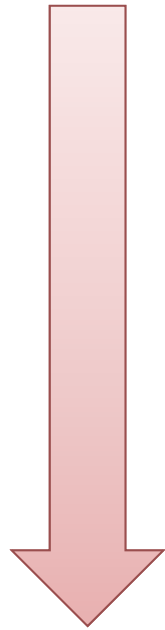
Induction

Analysis of Recursion

- **Solving recurrence equations**

- **E.g., Bit counting**
  - Critical operation: add
  - The recurrence relation

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

Analysis of Recursion

- **Backward substitutions**

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

For simplicity , let $n = 2^k$ (k is a nonnegativ e integer), that is, $k = \log n$

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

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

Smooth Functions

- $f(n)$
  - Nonnegative eventually non-decreasing function defined on the set of natural numbers

- $f(n)$ is called **smooth**
  - If $f(2n) \in \Theta(f(n))$.

- **Examples of smooth functions**
  - $\log n$, n, $n\log n$ and n^{α} ($\alpha \geq 0$)
  - E.g., $2n\log 2n = 2n(\log n + \log 2) \in \Theta(n\log n)$

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 non-decreasing 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 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, ...

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

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:**

$$x^2 - r_1 x - r_2 = 0$$

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

$Remember \quad equation \ : \ x^2 - r_1 x - r_2 = 0$

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

$$us_1^n + vs_2^n = us_1^{n\,2} s_1^2 + vs_2^{n\,2} s_2^2$$

$$= us_1^{n\,2}(\ r_1 s_1 + r_2\ ) + vs_2^{n\,2}(\ r_1 s_2 + r_2\ )$$

$$= r_1 us_1^{n\,1} + r_2 us_1^{n\,2} + r_1 vs_2^{n\,1} + r_2 vs_2^{n\,2}$$

$$= r_1(\ us_1^{n\,1} + vs_2^{n\,1}\ ) + r_2(\ us_1^{n\,2} + vs_2^{n\,2}\ )$$

$$= r_1 a_{n\,1} + r_2 a_{n\,2}$$

Back to Fibonacci Sequence

$f_0 = 0$

$f_1 = 1$

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

$\Longrightarrow$ 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 and Prove

- **Example: T(n)=2T($\lfloor$n/2$\rfloor$) +n**

- **Guess**

  - $T(n) \in O(n)$?
    - $T(n) \leq cn$, to be pro
  - $T(n) \in O(n^2)$?
    - $T(n) \leq cn^2$, to be prove
  - **Or maybe**, $T(n) \in O(n\log$
    - $T(n) \leq cn\log n$, to be prove

- **Prove**
  - by substitution

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

However:

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

Divide and Conquer Recursions

- **Divide and conquer**
  - o Divide the "big" problem to smaller ones
  - o Solve the "small" problems by recursion
  - o Combine results of small problems, and solve the original problem

- **Divide and conquer recursion**

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

divide   combine
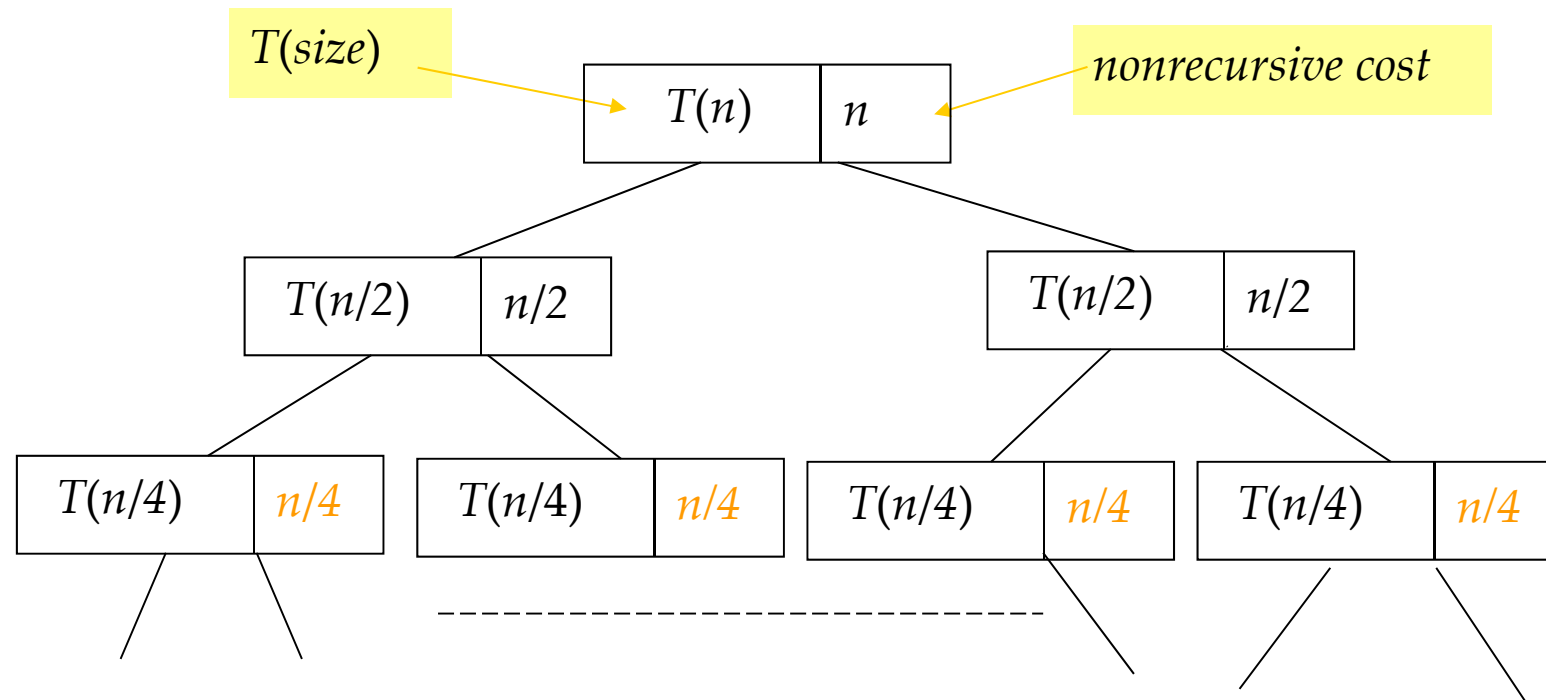
conquer

Recursion Tree

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

Recursion Tree

- **Node**
  - o Non-leaf
    - Non-recursive cost
    - Recursive cost
  - o Leaf
    - Base case

- **Edge**
  - o Recursion

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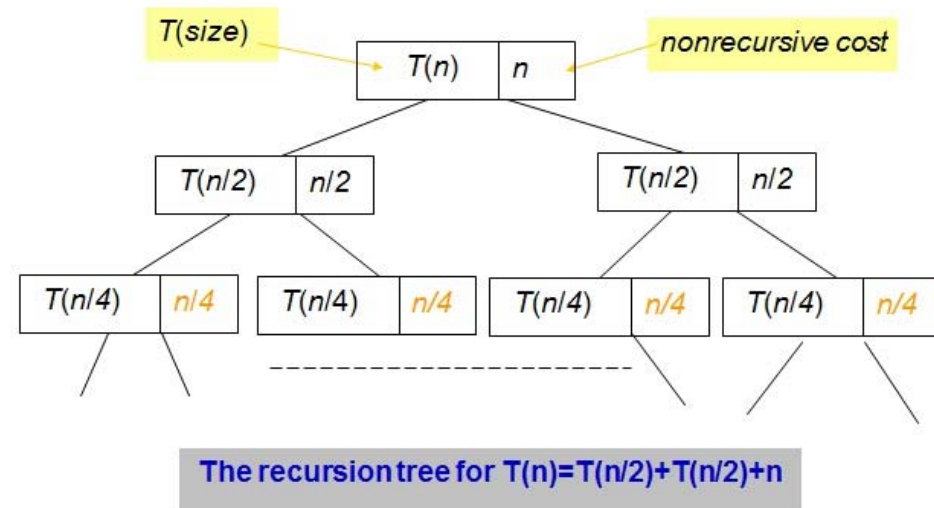
Recursion Tree

Recursive cost   Non-recursive cost

- $T(n) = 3T(n/4) + \Theta(n^2)$

\# of sub-problems   size of sub-problems

- **Total cost**

Σ

Sum of row sums

cn^2

$c(\tfrac{1}{4}n)^2$ $c(\tfrac{1}{4}n)^2$ $c(\tfrac{1}{4}n)^2$

$c((1/16)n)^2 c((1/16)n)^2$ ...... $c((1/16)n)^2$ $c((1/16)n)^2$ ...... $c((1/16)n)^2$

$T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$

$3^{\log_4 n} = n^{\log_4 3}$

Sum of Row-sums

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

$cn^2 \dashrightarrow \boldsymbol{cn^2}$

$c(\tfrac{1}{4} n)^2 \qquad c(\tfrac{1}{4} n)^2 \qquad c(\tfrac{1}{4} n)^2 \dashrightarrow \dfrac{3}{16} cn^2$

$\log_4 n$

$c((1/16)n)^2 \; c((1/16)n)^2 \; \cdots\cdots \qquad c((1/16)n)^2 \qquad c((1/16)n)^2 \; \cdots\cdots \; c((1/16)n)^2 \dashrightarrow \left(\dfrac{3}{16}\right)^2 cn^2$

$\cdots\cdots \qquad \cdots\cdots$

$T(1) \; T(1) \; T(1) \; T(1) \; T(1) \; T(1) \; T(1) \; T(1) \; T(1) \; T(1) \, T(1) \; T(1) \; T(1) \; T(1) \dashrightarrow \Theta\!\left(n^{\log_4 3}\right)$

$\text{Note:} \; 3^{\log_4 n} = n^{\log_4 3}$
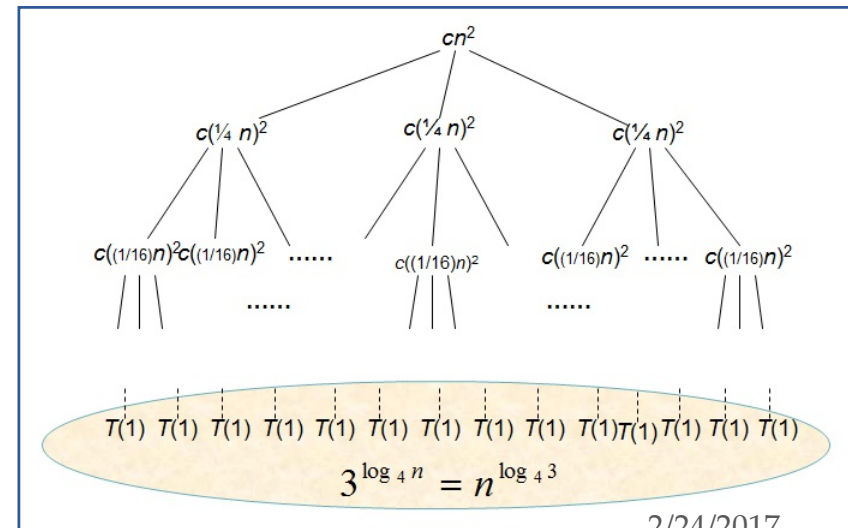
Total: $\Theta(n^2)$
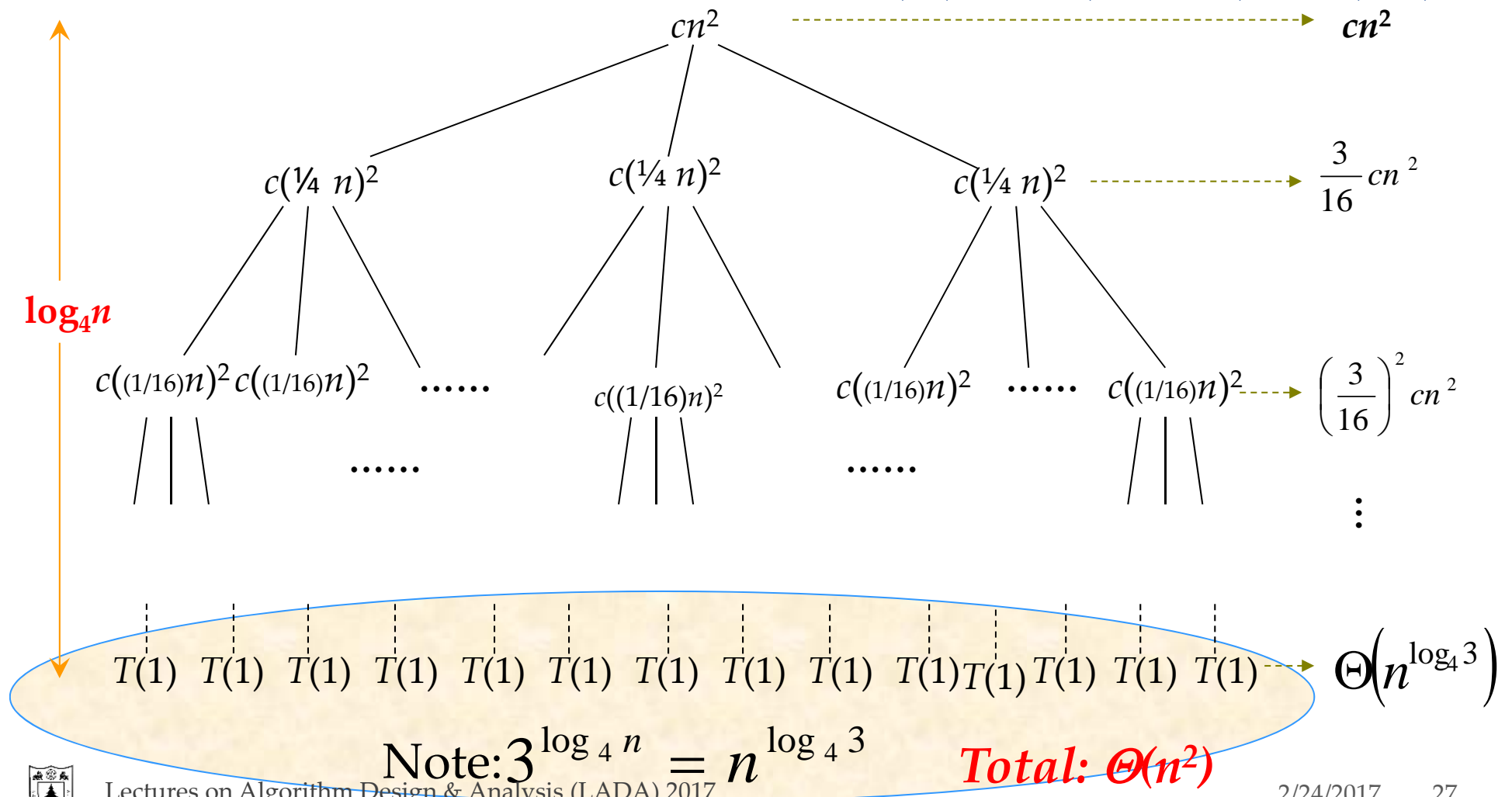
Solving the Divide-and-Conquer Recurrence
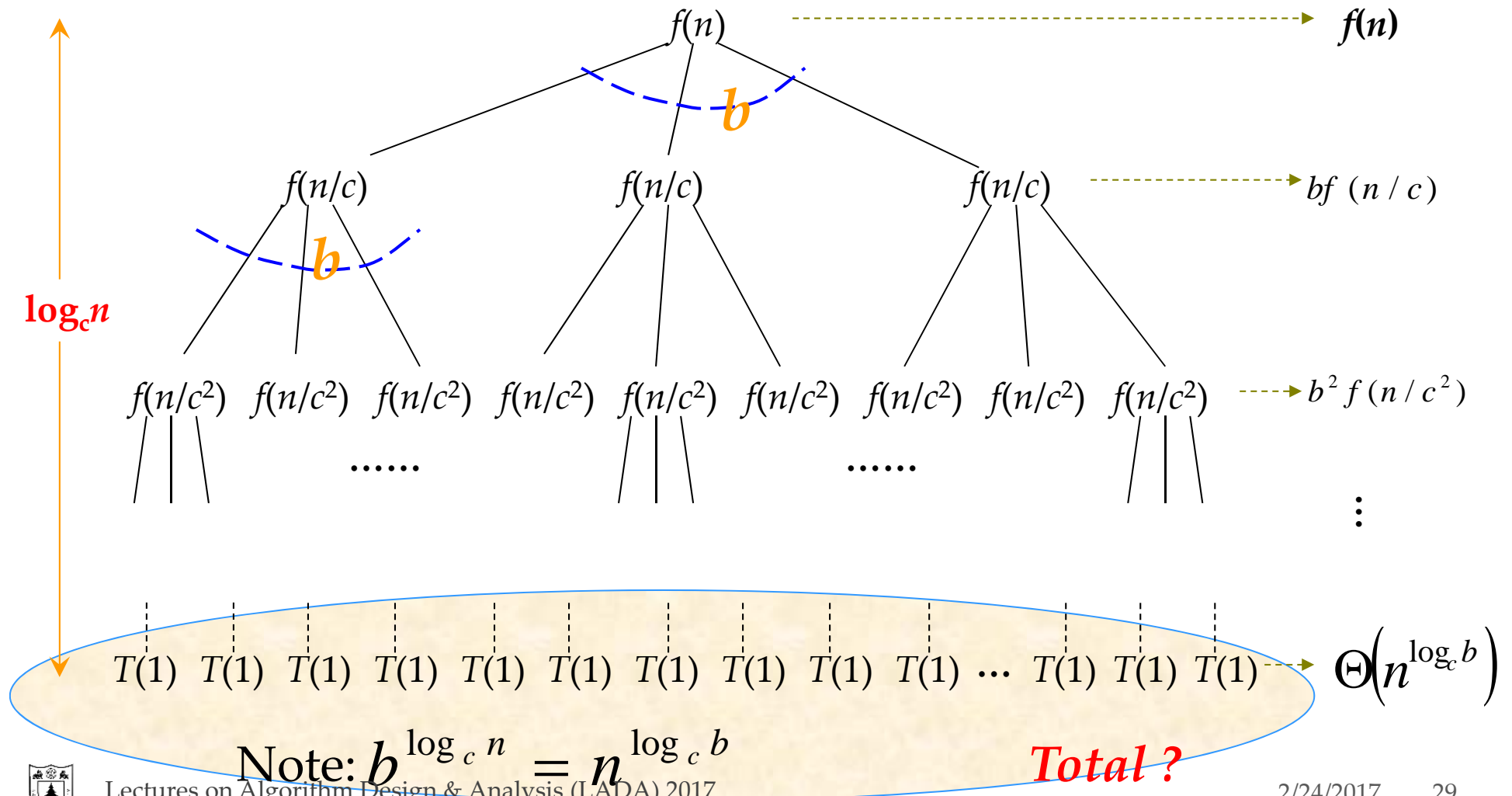
- **The recursion equation for divide-and-conquer, the general case: $T(n)=bT(n/c)+f(n)$**

- **Observations:**
  - Let base-cases occur at depth D(leaf), then $n/c^D=1$, that is $D=\log(n)/\log(c)$
  - Let the number of leaves of the tree be L, then $L=b^D$, that is $L=b^{(\log(n)/\log(c))}$.
  - By a little algebra: $L=n^E$, where $E=\log(b)/\log(c)$, called *critical exponent*.

Recursion Tree for

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

$f(n)$ → $f(n)$

b

$f(n/c)$ $f(n/c)$ $f(n/c)$ → $bf(n/c)$

b

$\log_c n$

$f(n/c^2)$ $f(n/c^2)$ $f(n/c^2)$ $f(n/c^2)$ $f(n/c^2)$ $f(n/c^2)$ $f(n/c^2)$ $f(n/c^2)$ $f(n/c^2)$ → $b^2 f(n/c^2)$

....

$T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ ... $T(1)$ $T(1)$ $T(1)$ → $\Theta\left(n^{\log_c b}\right)$

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

Total ?

Divide-and-Conquer - the Solution

- **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**

  o The recursion tree has depth $D=\log(n)/\log(c)$, so there are about that many row-sums.

- **The 0th row-sum**

  o is $f(n)$, the nonrecursive cost of the root.

- **The Dth row-sum**

  o is n^E, assuming base cases cost 1, or $\Theta(n^E)$ in any event.

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

- **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 if $bf(n/c) \leq \theta f(n)$ for some constant $\theta < 1$ and all sufficiently large n, then:
  $$T(n) \in \Theta(f(n))$$

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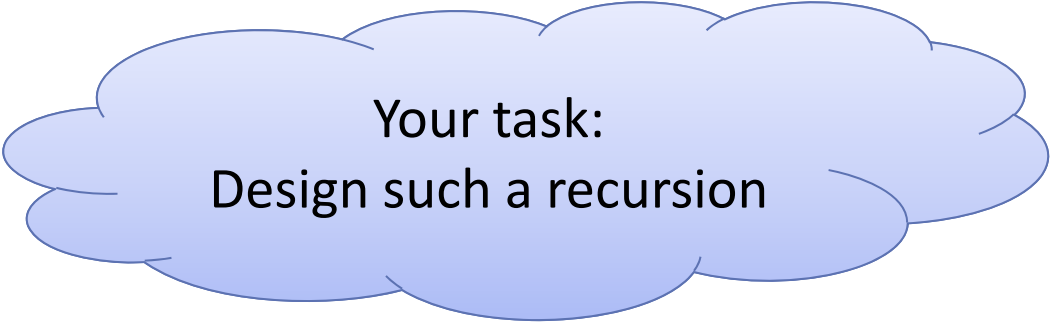
Using Master Theorem

- Example 1: $T(n) = 9T(\frac{n}{3}) + n$

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

  Case 1 applies: $T(n) = \Theta(n^2)$

- Example 2: $T(n) = T(\frac{2}{3}n) + 1$

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

  Case 2 applies: $T(n) = \Theta(\log n)$

- Example 3: $T(n) = 3T(\frac{n}{4}) + n \log n$

  $b = 3, c = 4, E = \log_4 3, f(n) = \Omega(n^{E+\epsilon})$

  $bf(\frac{n}{4}) = \frac{3}{4}n \log n - \frac{3}{2}n$

  Case 3 applies: $T(n) = \Theta(n \log n)$

Looking at the Gap

- **Often, none of the 3 cases in the Master Theorem apply.**

Your task:
Design such a recursion

Thank you!

Q & A

Yu Huang

yuhuang@nju.edu.cn

http://cs.nju.edu.cn/yuhuang