Design and Analysis of Algorithms
Lecture 3: Solving Recurrences

童咏昕

北京航空航天大学
计算机学院

从杠铃增重问题到排序问题

排序问题回顾

排序问题

Sorting Problem

输入

- 包含n个数字的序列 $< a_1, \ldots, a_n >$

输出

- 输入序列的升序

$$< a_1', a_2', \ldots, a_n' >$$

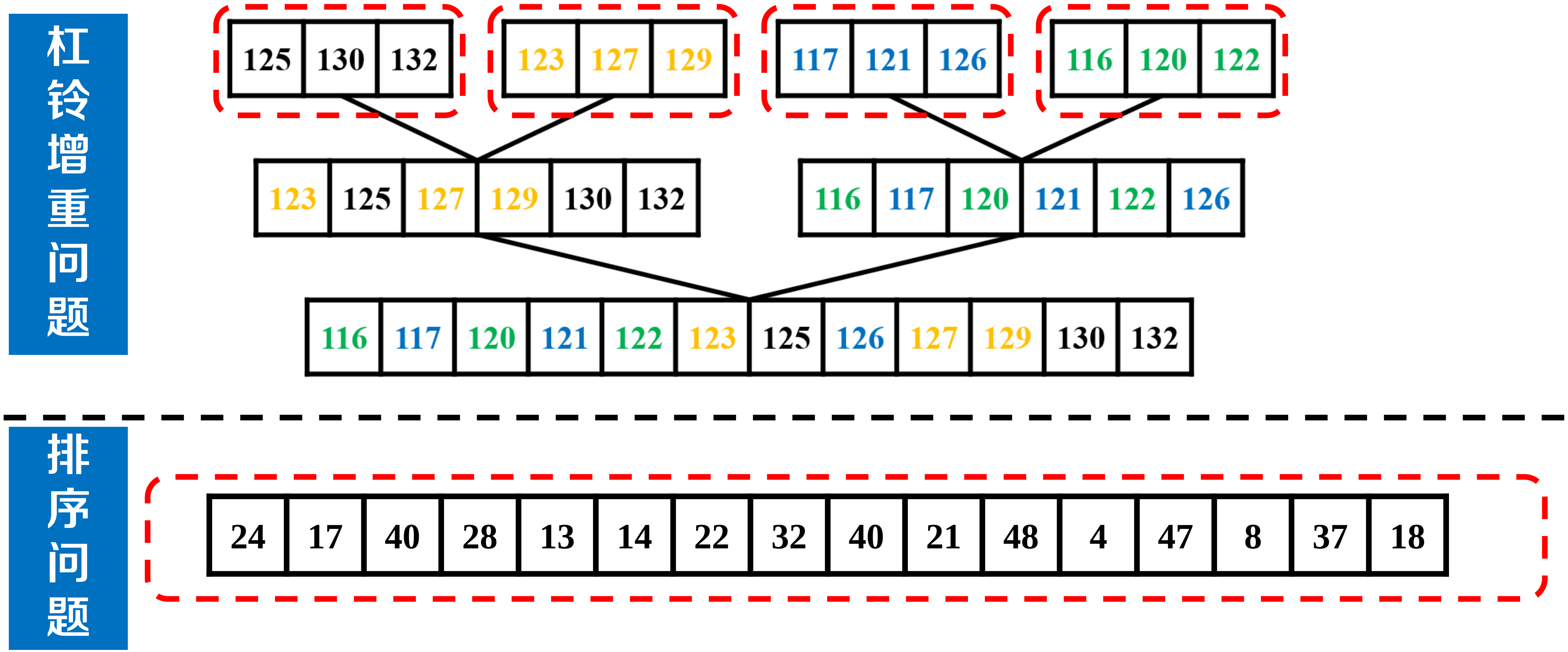
满足$a_1' \leq a_2' \leq \cdots \leq a_n'$

示例

24	17	40	28	13	14	22	32	40	21	48	4	47	8	37	18

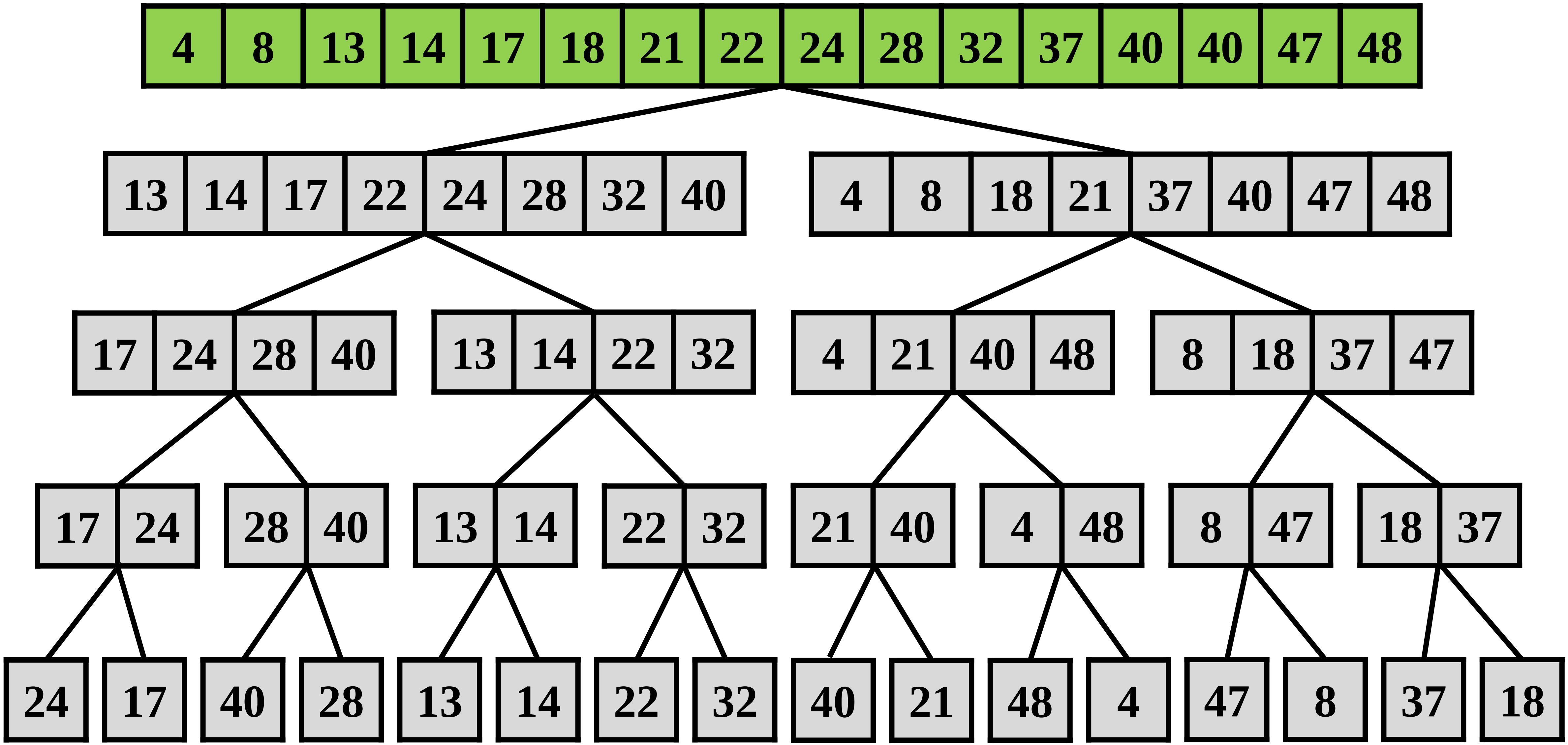
从杠铃增重问题到排序问题

- 问题输入变化
  - 完整数组输入
  - 局部有序缺失

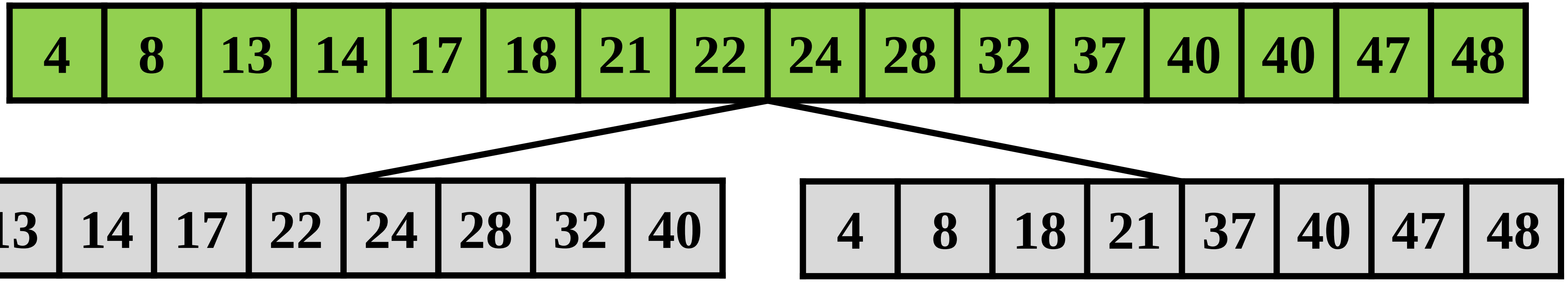


从杠铃增重问题到排序问题

- 构建有序数组：两两合并



归并排序



- 算法流程
  - 将数组$A[1,n]$排序问题分解为$A[1,\left\lfloor\frac{n}{2}\right\rfloor]$和$A[\left\lfloor\frac{n}{2}\right\rfloor+1,n]$排序问题
  - 递归解决子问题得到两个有序的子数组
  - 将两个有序子数组合并为一个有序数组

分解原问题

解决子问题

合并问题解

归并排序：分解数组，递归求解，合并排序

归并排序：伪代码

- **MergeSort**$(\boldsymbol{A}, \boldsymbol{left}, \boldsymbol{right})$

初始调用：MergeSort$(\boldsymbol{A}, \boldsymbol{1}, \boldsymbol{n})$

输入: 数组$\boldsymbol{A[1..n]}$**,数组下标**$\boldsymbol{left, right}$
输出: 递增数组$\boldsymbol{A[left..right]}$
if $left \geq right$ **then**
| **return** $A[left..right]$
end
$mid \leftarrow \lfloor \frac{left+right}{2} \rfloor$
MergeSort$(A, left, mid)$
MergeSort$(A, mid+1, right)$
Merge$(A, left, mid, right)$
return $A[left..right]$

归并排序：伪代码

- $\mathbf{MergeSort}(\boldsymbol{A}, \boldsymbol{left}, \boldsymbol{right})$

初始调用：$\mathbf{MergeSort}(\boldsymbol{A}, \mathbf{1}, \boldsymbol{n})$

输入: 数组$A[1..n]$,数组下标$left, right$
输出: 递增数组$A[left..right]$

if $left \geq right$ **then**
  return $A[left..right]$
end
$mid \leftarrow \lfloor \frac{left+right}{2} \rfloor$
MergeSort(A, $left$, mid)
MergeSort(A, $mid+1$, $right$)
Merge($A, left, mid, right$)
return $A[left..right]$

递归终止：仅有一个元素

归并排序：伪代码

- $\mathbf{MergeSort}(\boldsymbol{A}, \boldsymbol{left}, \boldsymbol{right})$

初始调用：$\mathbf{MergeSort}(\boldsymbol{A}, \mathbf{1}, \boldsymbol{n})$

输入: 数组$\boldsymbol{A}[1..n]$,数组下标$\boldsymbol{left}, \boldsymbol{right}$
输出: 递增数组$\boldsymbol{A}[\boldsymbol{left}..\boldsymbol{right}]$
if $left \geq right$ **then**
  | **return** $A[left..right]$
end
$mid \leftarrow \lfloor \frac{left+right}{2} \rfloor$
$\text{MergeSort}(A, left, mid)$
$\text{MergeSort}(A, mid+1, right)$
$\text{Merge}(A, left, mid, right)$
return $A[left..right]$

计算子问题规模

归并排序：伪代码

- MergeSort$(A, left, right)$

初始调用：MergeSort$(A, 1, n)$

输入: 数组$A[1..n]$,数组下标$left, right$
输出: 递增数组$A[left..right]$
if $left \geq right$ **then**
  | **return** $A[left..right]$
end
$mid \leftarrow \lfloor \frac{left+right}{2} \rfloor$
MergeSort$(A, left, mid)$
MergeSort$(A, mid+1, right)$
Merge$(A, left, mid, right)$
return $A[left..right]$

递归求解子问题

归并排序：伪代码

- $\mathbf{MergeSort}(\boldsymbol{A}, \boldsymbol{left}, \boldsymbol{right})$

初始调用：$\mathbf{MergeSort}(\boldsymbol{A}, \mathbf{1}, \boldsymbol{n})$

输入: 数组$A[1..n]$,数组下标$left, right$
输出: 递增数组$A[left..right]$
if $left \geq right$ **then**
　| **return** $A[left..right]$
end
$mid \leftarrow \lfloor \frac{left+right}{2} \rfloor$
MergeSort$(A, left, mid)$
MergeSort$(A, mid+1, right)$
Merge$(A, left, mid, right)$
return $A[left..right]$

合并子问题解

归并排序：伪代码

- **Merge($A, left, mid, right$)**

输入: 数组$A[1..n]$,数组下标$left, mid, right$
输出: 递增数组$A[left..right]$

$A'[left..right] \leftarrow A[left..right]$
$i \leftarrow left,\ j \leftarrow mid+1,\ k \leftarrow 0$
while $i \leq mid\ and\ j \leq right$ **do**
  if $A'[i] \leq A'[j]$ **then**
    $A[left+k] \leftarrow A'[i]$
    $k \leftarrow k+1, i \leftarrow i+1$
  end
  else
    $A[left+k] \leftarrow A'[j]$
    $k \leftarrow k+1, j \leftarrow j+1$
  end
end
if $i \leq mid$ **then**
  $A[left+k..right] \leftarrow A'[i..mid]$
end
else
  $A[left+k..right] \leftarrow A'[j..right]$
end
return $A[left..right]$

初始化

归并排序：伪代码

- Merge($A, left, mid, right$)

输入: 数组$A[1..n]$,数组下标$left, mid, right$
输出: 递增数组$A[left..right]$

$A'[left..right] \leftarrow A[left..right]$
$i \leftarrow left,\ j \leftarrow mid+1,\ k \leftarrow 0$
while $i \leq mid$ *and* $j \leq right$ **do**
    if $A'[i] \leq A'[j]$ **then**
        $A[left+k] \leftarrow A'[i]$
        $k \leftarrow k+1, i \leftarrow i+1$
    end
    else
        $A[left+k] \leftarrow A'[j]$
        $k \leftarrow k+1, j \leftarrow j+1$
    end
end
if $i \leq mid$ **then**
    $A[left+k..right] \leftarrow A'[i..mid]$
end
else
    $A[left+k..right] \leftarrow A'[j..right]$
end
return $A[left..right]$

遍历子数组，进行合并

归并排序：伪代码

- Merge($A, left, mid, right$)

输入: 数组$A[1..n]$,数组下标$left, mid, right$
输出: 递增数组$A[left..right]$

```
A'[left..right] ← A[left..right]
i ← left, j ← mid + 1, k ← 0
while i ≤ mid and j ≤ right do
    if A'[i] ≤ A'[j] then
        A[left + k] ← A'[i]
        k ← k + 1, i ← i + 1
    end
    else
        A[left + k] ← A'[j]
        k ← k + 1, j ← j + 1
    end
end
if i ≤ mid then
    A[left + k..right] ← A'[i..mid]
end
else
    A[left + k..right] ← A'[j..right]
end
return A[left..right]
```

添加剩余元素保证有序

归并排序：伪代码

- Merge$(A, left, mid, right)$

输入: 数组$A[1..n]$,数组下标$left, mid, right$
输出: 递增数组$A[left..right]$

$A'[left..right] \leftarrow A[left..right]$
$i \leftarrow left,\ j \leftarrow mid+1,\ k \leftarrow 0$
while $i \leq mid$ and $j \leq right$ **do**
    if $A'[i] \leq A'[j]$ **then**
        $A[left+k] \leftarrow A'[i]$
        $k \leftarrow k+1, i \leftarrow i+1$
    end
    else
        $A[left+k] \leftarrow A'[j]$
        $k \leftarrow k+1, j \leftarrow j+1$
    end
end
if $i \leq mid$ **then**
    $A[left+k..right] \leftarrow A'[i..mid]$
end
else
    $A[left+k..right] \leftarrow A'[j..right]$
end
return $A[left..right]$

时间复杂度：$O(n)$

归并排序：复杂度分析

- $T(n)$ ： 完成MergeSort$(A, 1, n)$ 的运行时间
  - 为便于分析，假设n是2的幂

归并排序：复杂度分析

- $T(n)$：完成MergeSort$(A, 1, n)$ 的运行时间
  - 为便于分析，假设n是2的幂
- MergeSort$(A, left, right)$

输入: 数组$A[1..n]$,数组下标$left, right$
输出: 递增数组$A[left..right]$
if $left \geq right$ **then**
  | **return** $A[left..right]$
end
$mid \leftarrow \lfloor \frac{left+right}{2} \rfloor$
MergeSort$(A, left, mid)$
MergeSort$(A, mid+1, right)$
Merge$(A, left, mid, right)$
return $A[left..right]$

归并排序：复杂度分析

- $T(n)$：完成MergeSort$(A,1,n)$ 的运行时间
  - 为便于分析，假设n是2的幂
- MergeSort$(A, left, right)$

输入: 数组$A[1..n]$,数组下标$left, right$
输出: 递增数组$A[left..right]$

$$\left.\begin{array}{l}\textbf{if } left \geq right \textbf{ then} \\ \quad \textbf{return } A[left..right] \\ \textbf{end} \\ mid \leftarrow \lfloor \frac{left+right}{2} \rfloor \end{array}\right\} O(1)$$

MergeSort$(A, left, mid)$ → $T(n/2)$
MergeSort$(A, mid+1, right)$ → $T(n/2)$
Merge$(A, left, mid, right)$ → $O(n)$
return $A[left..right]$

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

归并排序：复杂度分析

- $T(n)$：完成MergeSort$(A, 1, n)$ 的运行时间
  - 为便于分析，假设n是2的幂
- MergeSort$(A, left, right)$

```
输入: 数组A[1..n],数组下标left, right
输出: 递增数组A[left..right]
if left ≥ right then
|  return A[left..right]
end
mid ← ⌊(left+right)/2⌋
MergeSort(A, left, mid)
MergeSort(A, mid+1, right)
Merge(A, left, mid, right)
return A[left..right]
```

$O(1)$

$T(n/2)$

$T(n/2)$

$O(n)$

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

?

归并排序：复杂度分析

- 递归树法：用树的形式表示抽象递归

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

归并排序：复杂度分析

- 递归树法：用树的形式表示抽象递归

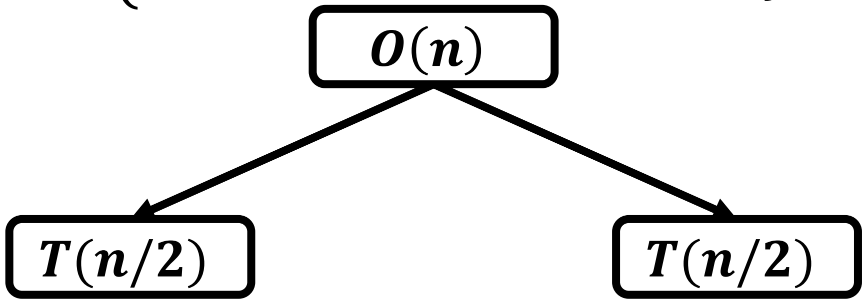
$$T(n)=\begin{cases}2T(n/2)+O(n), & if\ n>1\\ O(1), & if\ n=1\end{cases}$$

$$\boxed{T(n)}$$

归并排序：复杂度分析

- 递归树法：用树的形式表示抽象递归

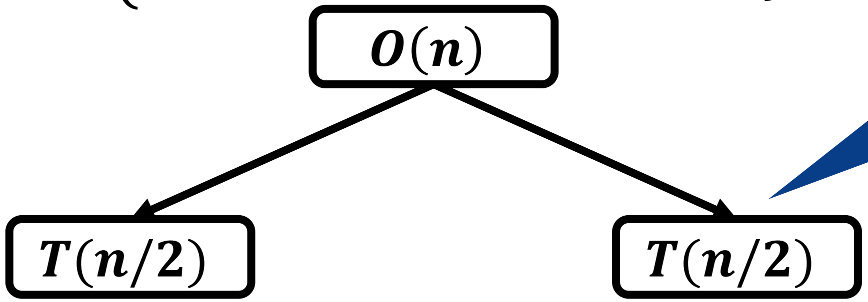
$$T(n) = \begin{cases} 2T(n/2) + O(n), & if\ n > 1 \\ O(1), & if\ n = 1 \end{cases}$$



归并排序：复杂度分析

- 递归树法：用树的形式表示抽象递归

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

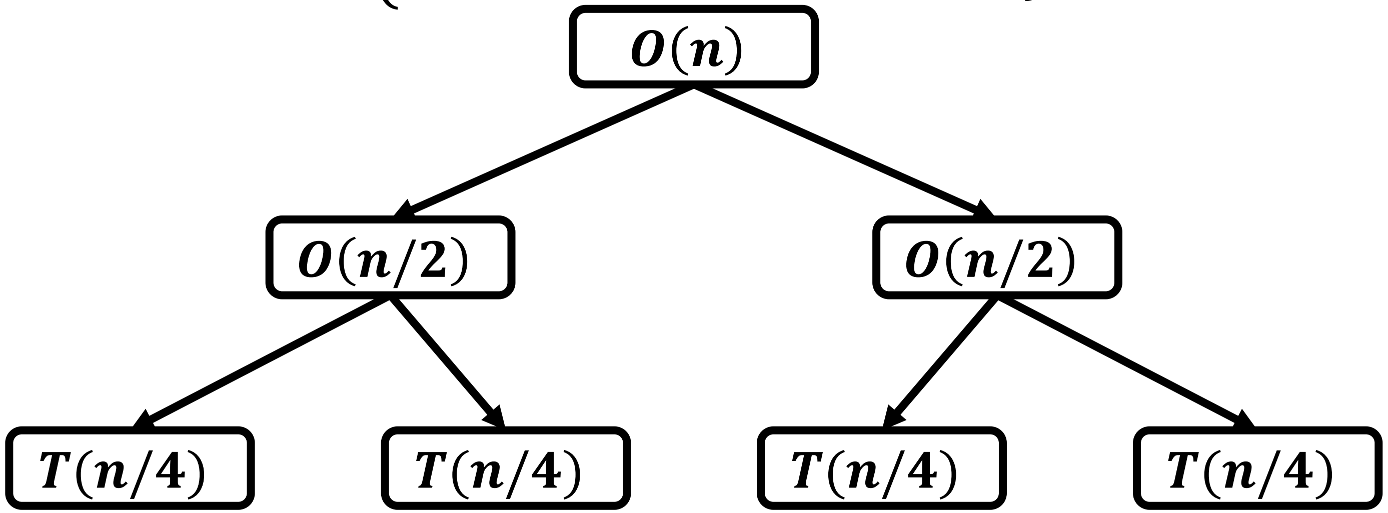


每个结点代表解决一个子问题的代价

归并排序：复杂度分析

- 递归树法：用树的形式表示抽象递归

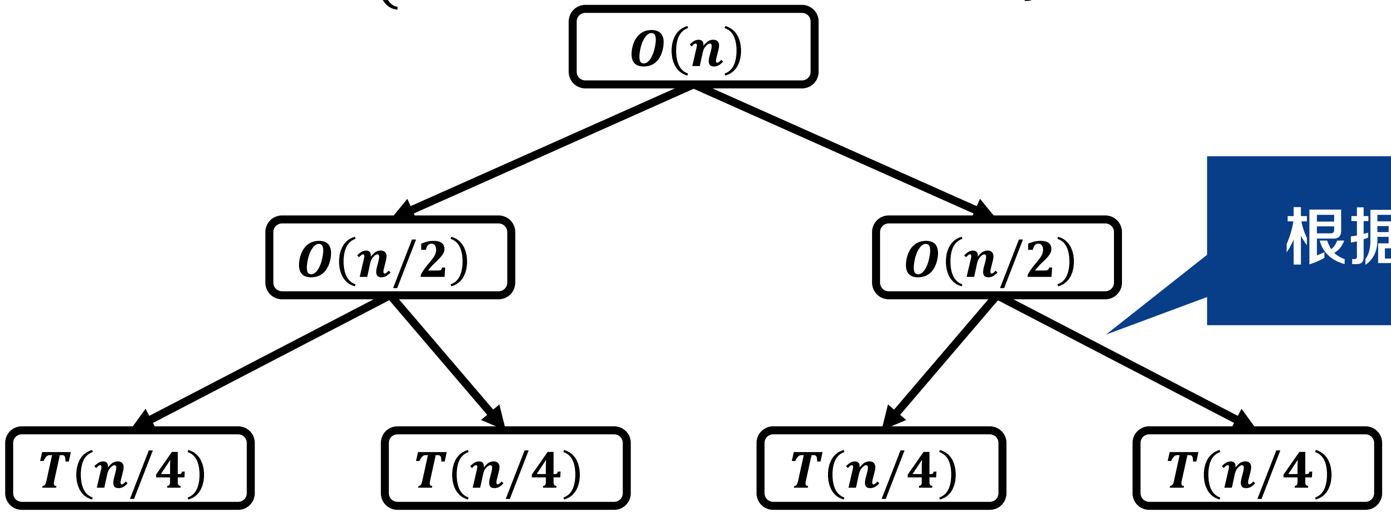
$$T(n) = \begin{cases} 2T(n/2) + O(n), & if\ n > 1 \\ O(1), & if\ n = 1 \end{cases}$$



归并排序：复杂度分析

- 递归树法：用树的形式表示抽象递归

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

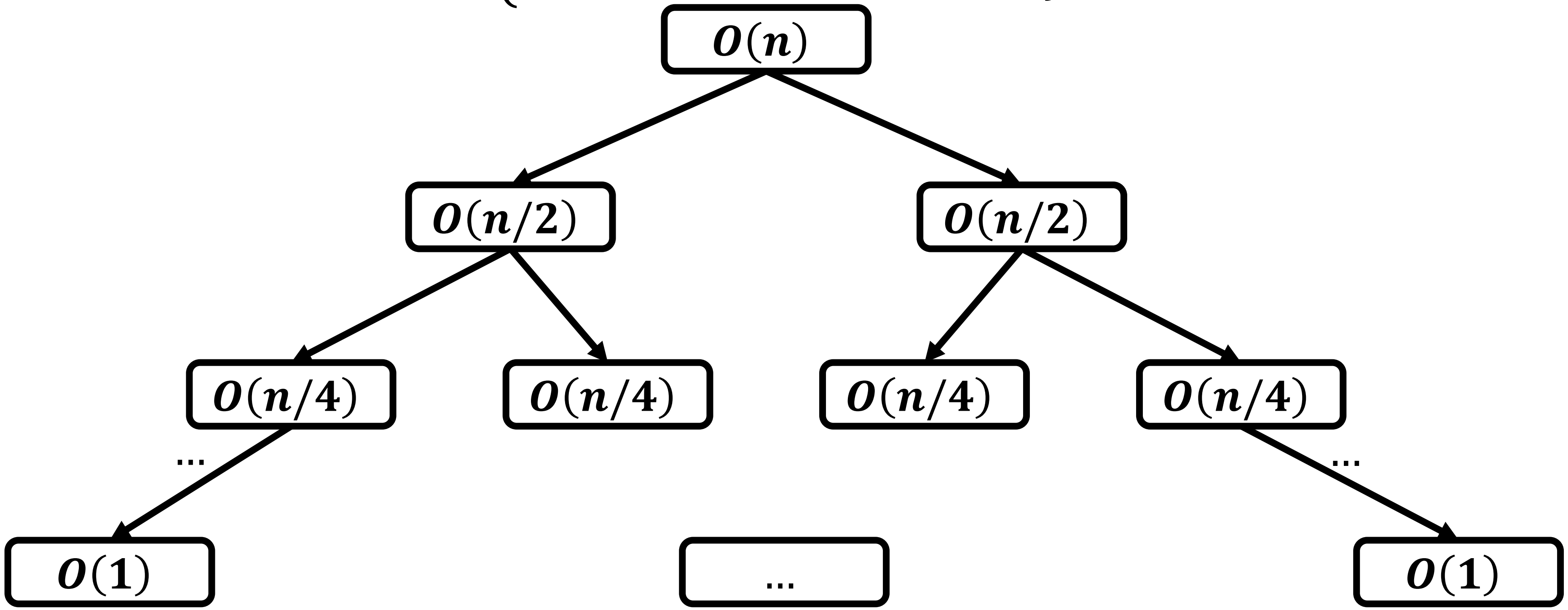


根据递归式向下分解

归并排序：复杂度分析

- 递归树法：用树的形式表示抽象递归

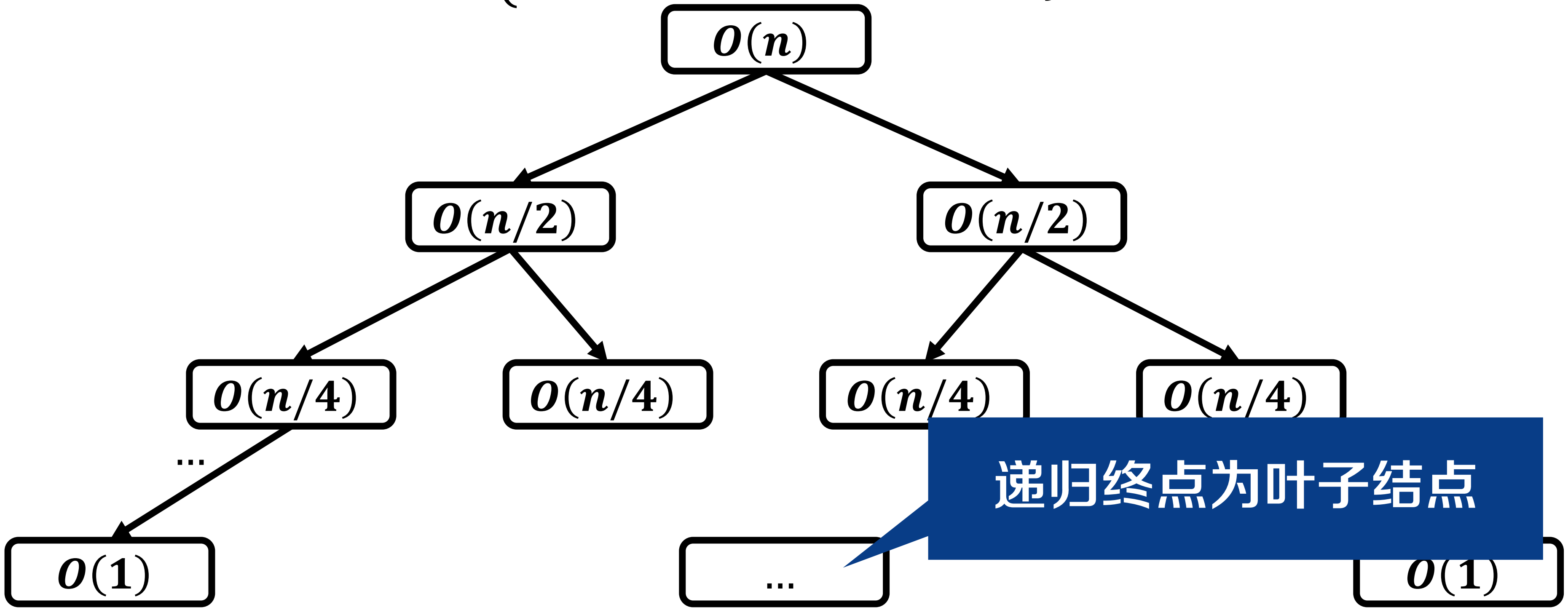
$$T(n) = \begin{cases} 2T(n/2) + O(n), & if\ n > 1 \\ O(1), & if\ n = 1 \end{cases}$$



归并排序：复杂度分析

- 递归树法：用树的形式表示抽象递归

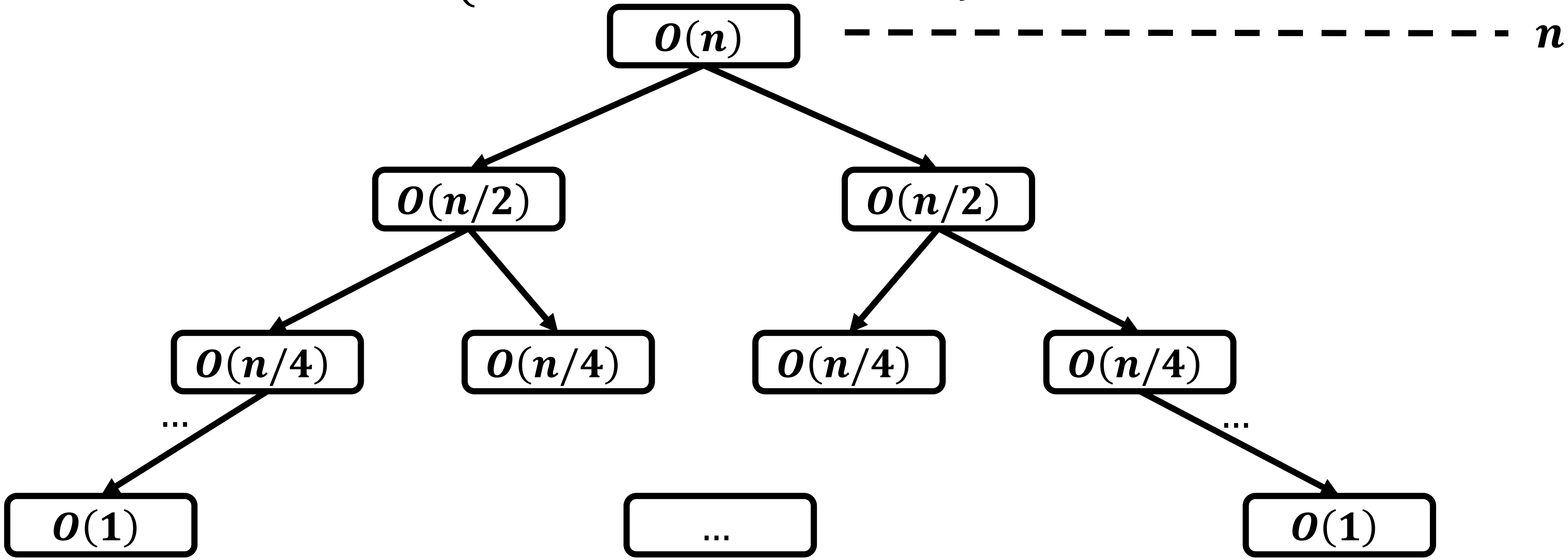
$$T(n) = \begin{cases} 2T(n/2) + O(n), & if\ n > 1 \\ O(1), & if\ n = 1 \end{cases}$$



归并排序：复杂度分析

- 递归树法：用树的形式表示抽象递归

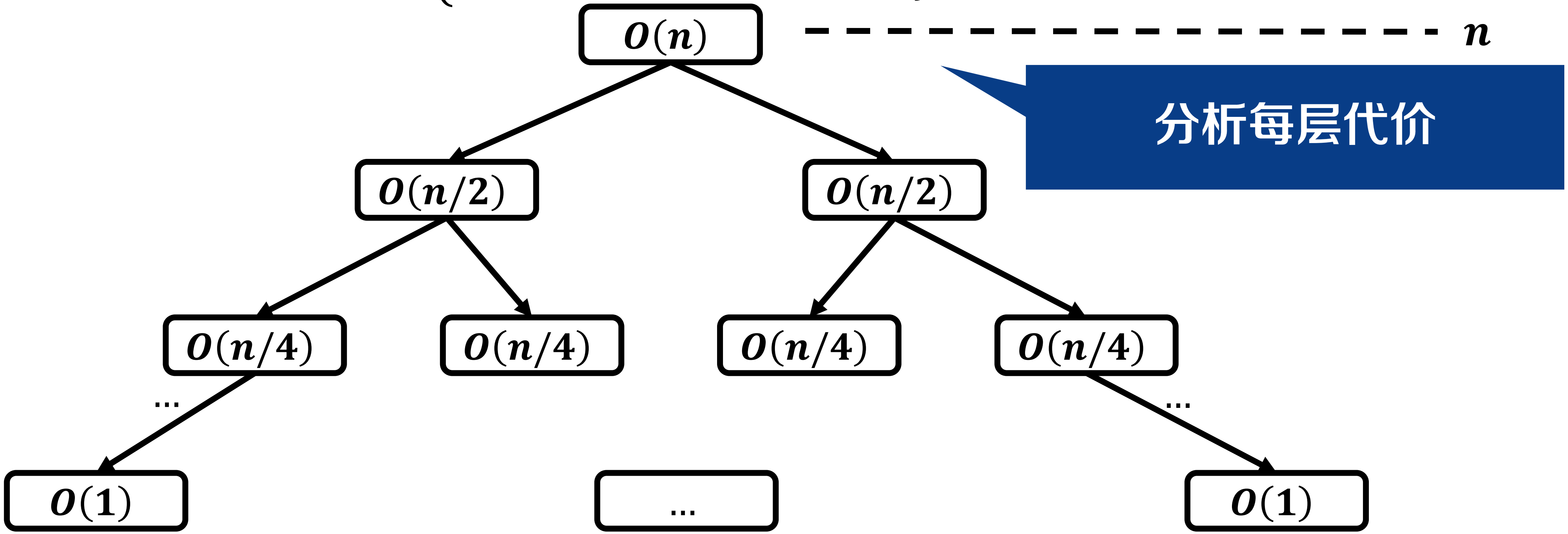
$$T(n)=\begin{cases}2T(n/2)+O(n), & if\ n>1\\ O(1), & if\ n=1\end{cases}$$



归并排序：复杂度分析

- 递归树法：用树的形式表示抽象递归

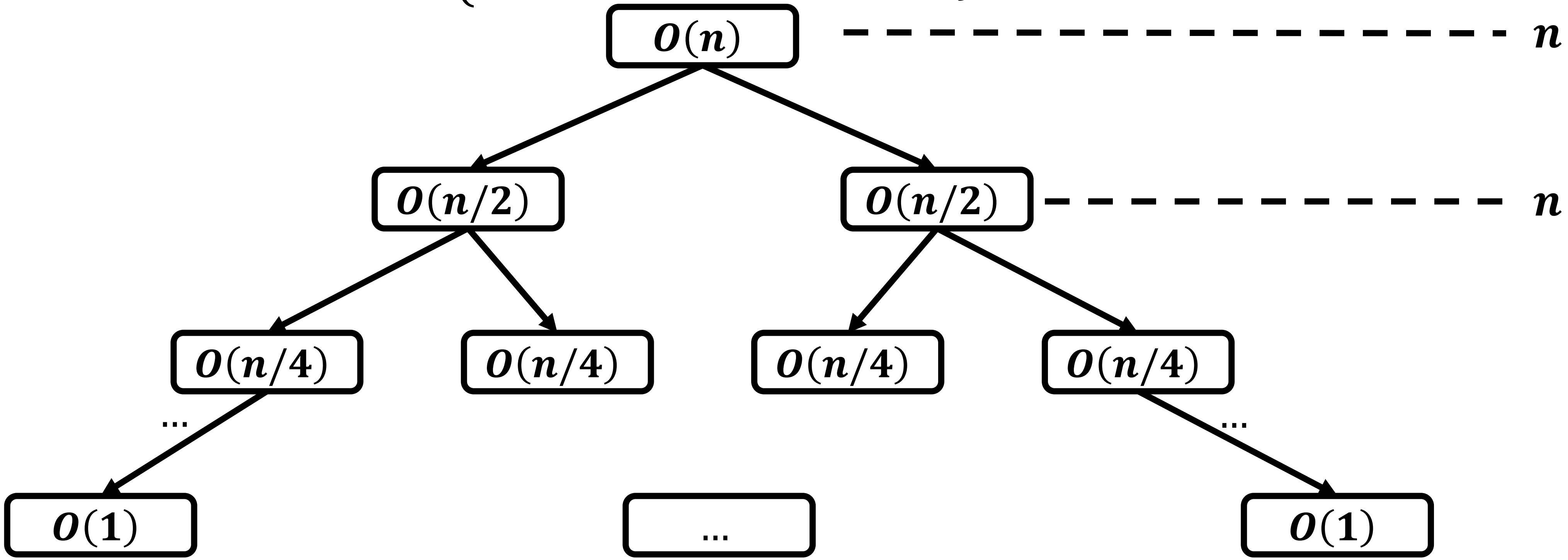
$$T(n) = \begin{cases} 2T(n/2) + O(n), & if\ n > 1 \\ O(1), & if\ n = 1 \end{cases}$$



归并排序：复杂度分析

- 递归树法：用树的形式表示抽象递归

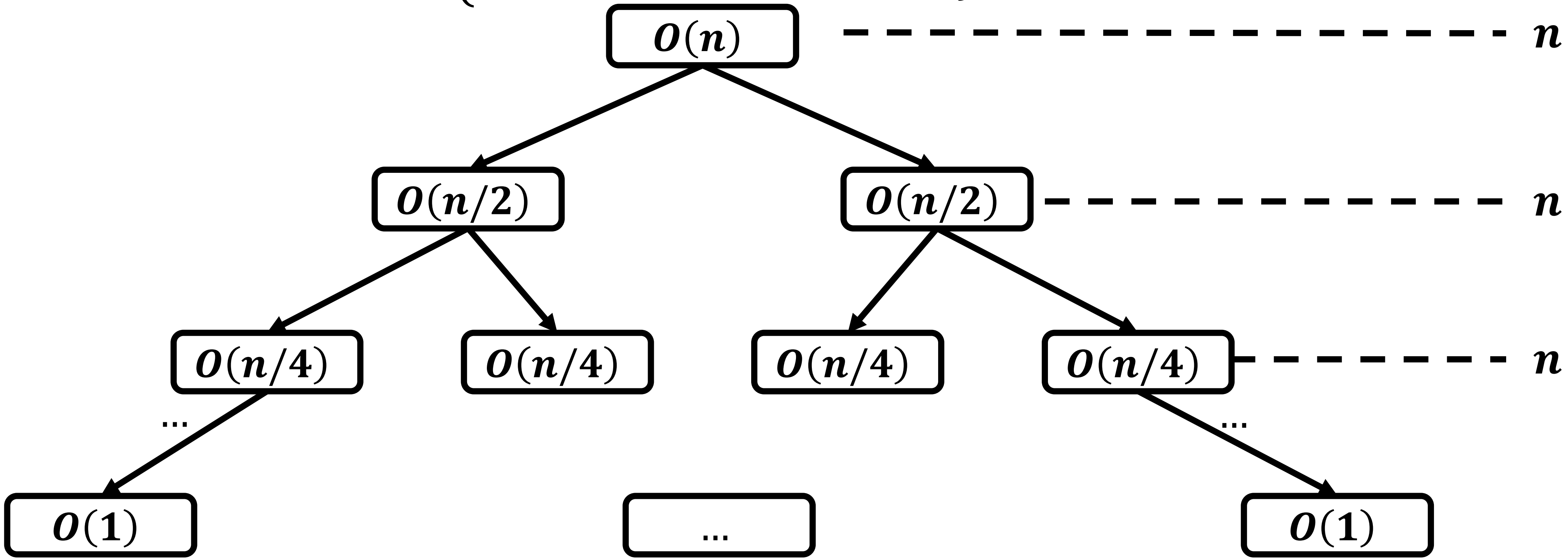
$$T(n) = \begin{cases} 2T(n/2) + O(n), & if\ n > 1 \\ O(1), & if\ n = 1 \end{cases}$$



归并排序：复杂度分析

- 递归树法：用树的形式表示抽象递归

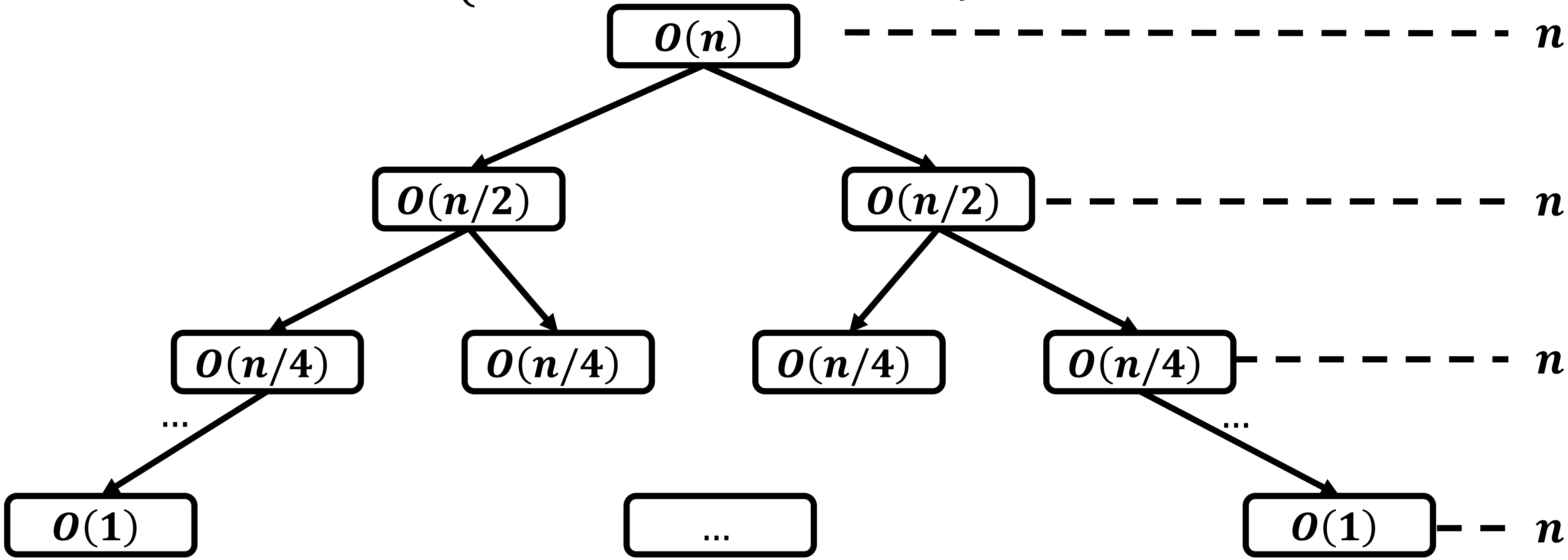
$$T(n)=\begin{cases}2T(n/2)+O(n), & if\ n>1\\ O(1), & if\ n=1\end{cases}$$



归并排序：复杂度分析

- 递归树法：用树的形式表示抽象递归

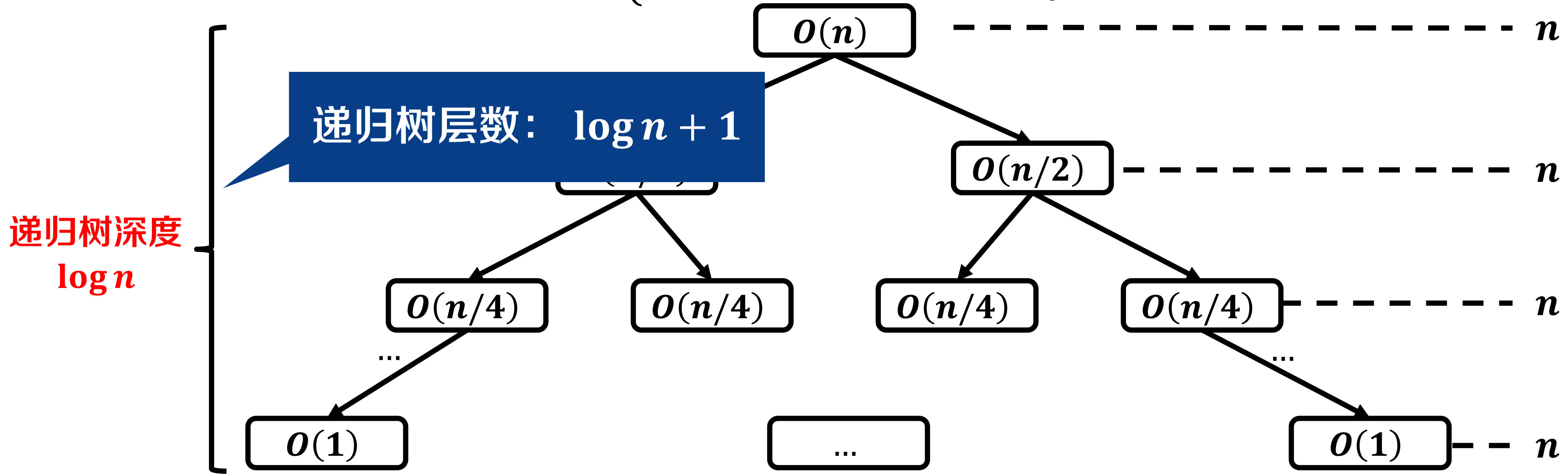
$$T(n)=\begin{cases}2T(n/2)+O(n), & if\ n>1\\ O(1), & if\ n=1\end{cases}$$



归并排序：复杂度分析

- 递归树法：用树的形式表示抽象递归

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

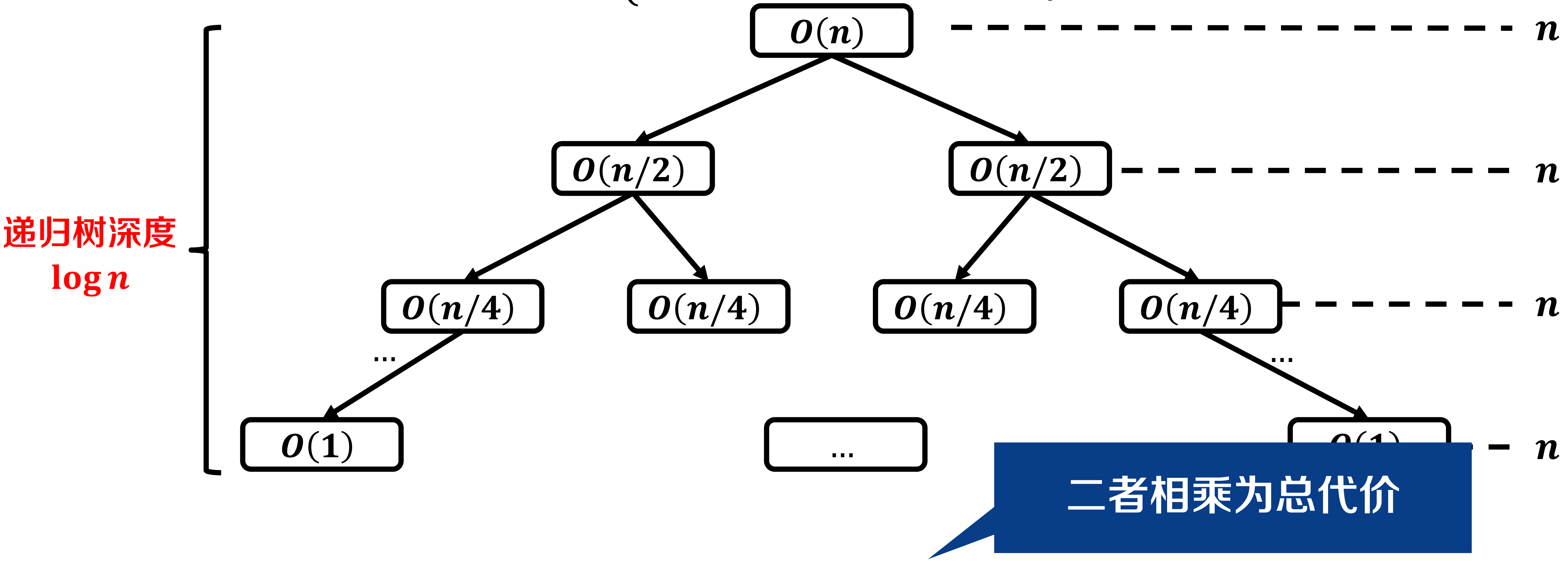


由于树的深度通常由0开始计数，故层数=深度+1，后续统一用“深度”

归并排序：复杂度分析

- 递归树法：用树的形式表示抽象递归

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



$T(n) = O(n \log n)$

递归式分析方法

递归树法

代入法

主定理法

递归式分析方法

递归树法

代入法

主定理法

递归树法：实例

$$T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & if\ n \geq 4 \\ 1, & if\ n < 4 \end{cases}$$

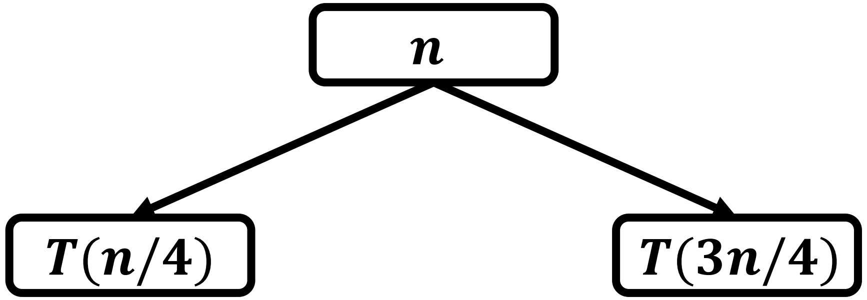
递归树法：实例

$$T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & if\ n \geq 4 \\ 1, & if\ n < 4 \end{cases}$$

$$\boxed{T(n)}$$

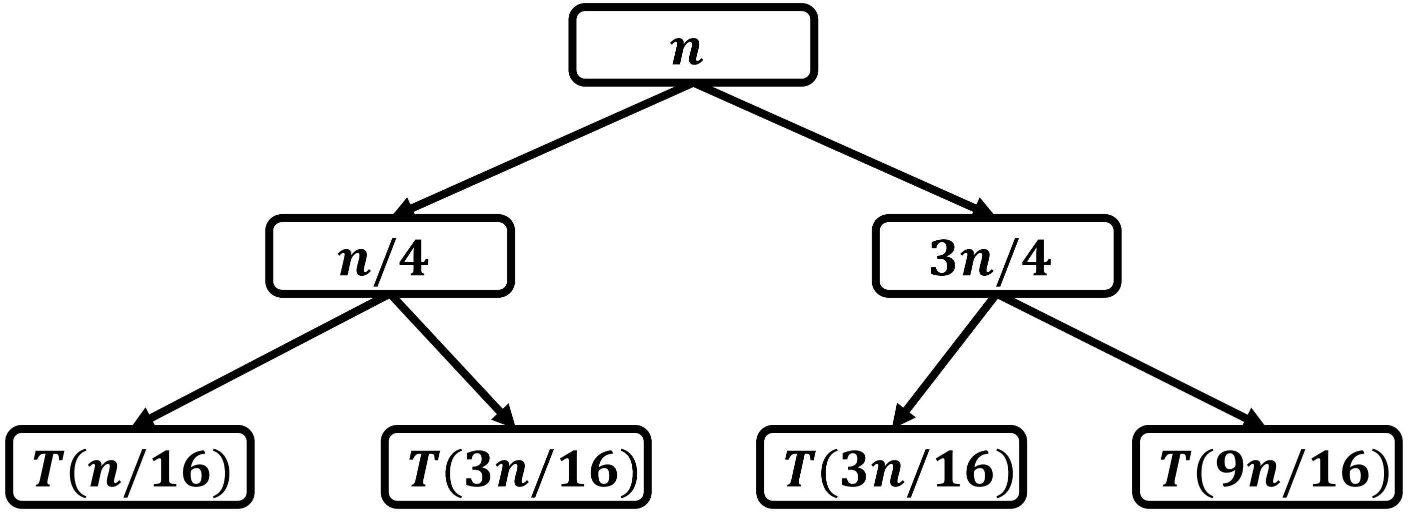
递归树法：实例

$$T(n)=\begin{cases}T(n/4)+T(3n/4)+n, & if\ n\geq 4\\ 1, & if\ n<4\end{cases}$$



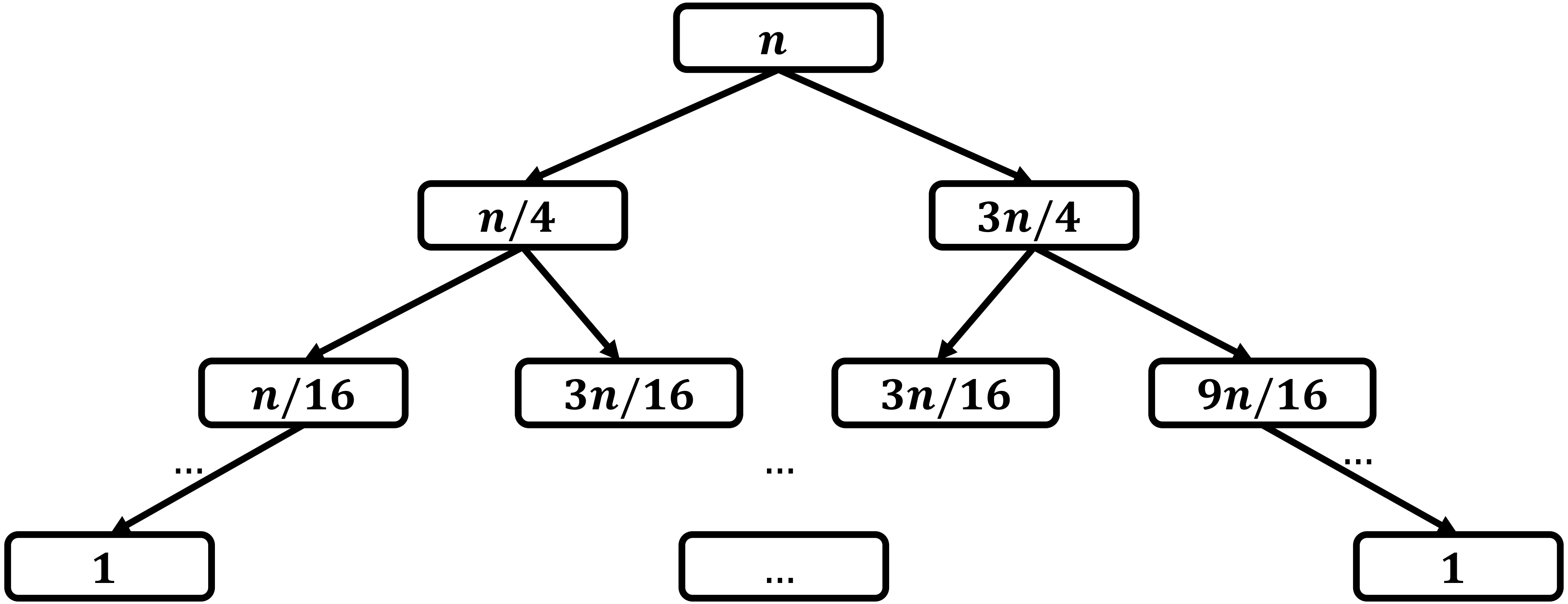
递归树法：实例

$$T(n)=\begin{cases}T(n/4)+T(3n/4)+n, & if\ n\geq 4\\ 1, & if\ n<4\end{cases}$$



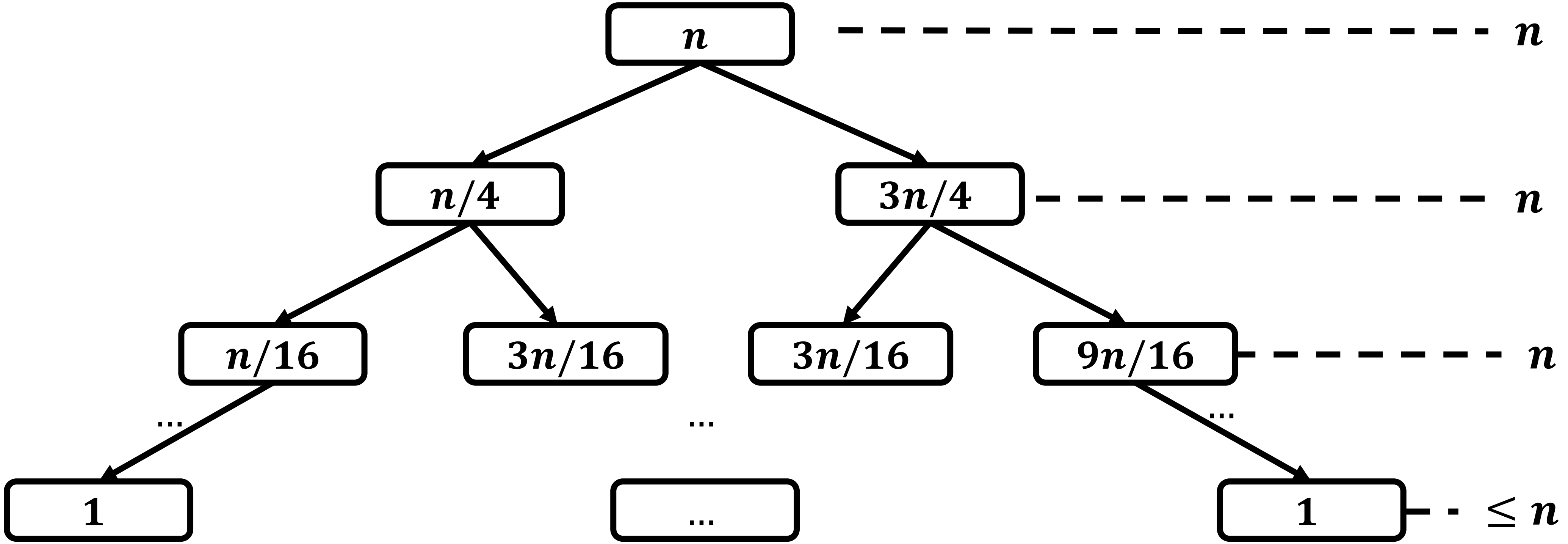
递归树法：实例

$$T(n)=\begin{cases}T(n/4)+T(3n/4)+n, & if\ n\geq 4\\ 1, & if\ n<4\end{cases}$$



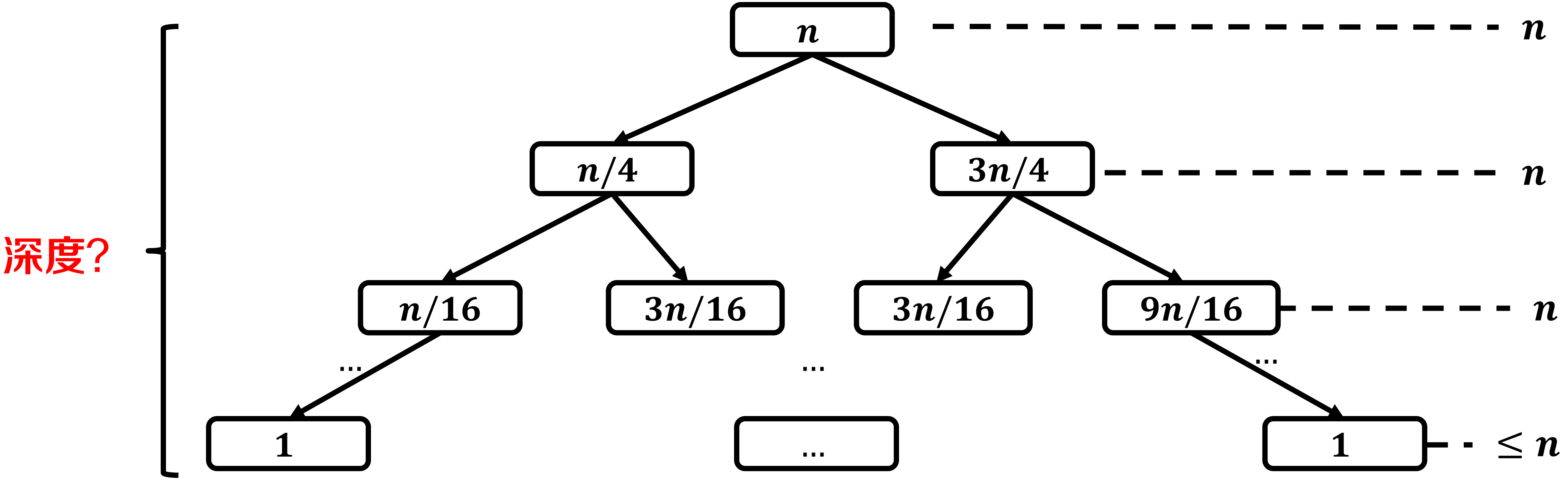
递归树法：实例

$$T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & if\ n \geq 4 \\ 1, & if\ n < 4 \end{cases}$$



递归树法：实例

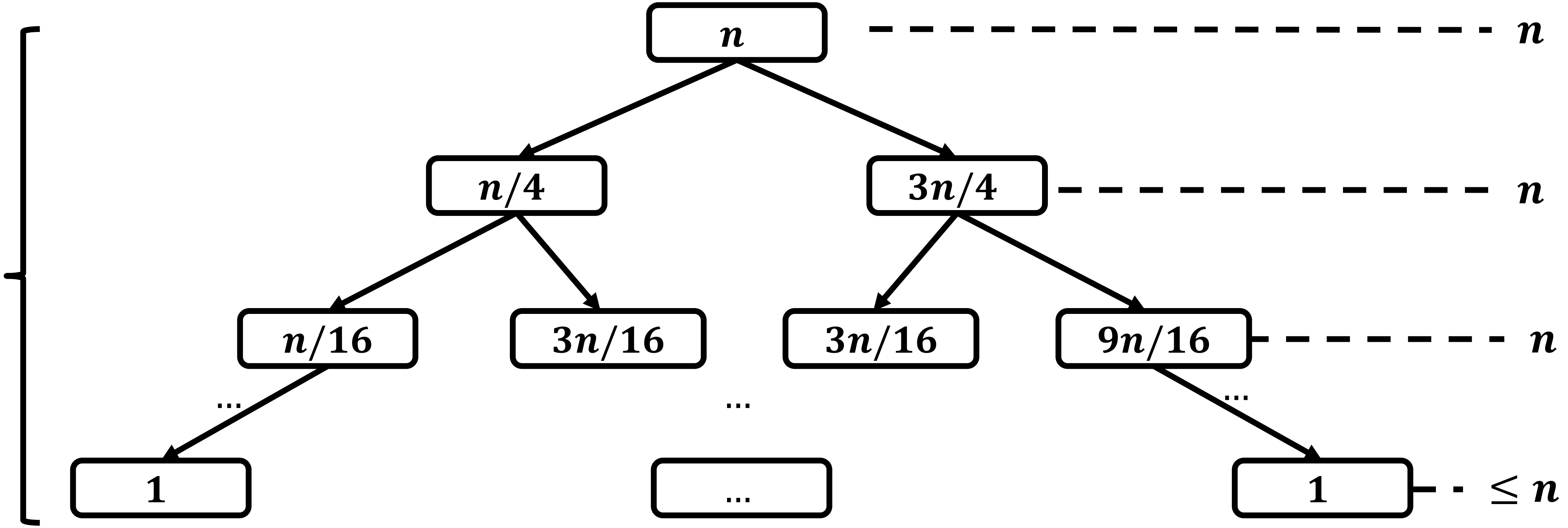
$$T(n)=\begin{cases}T(n/4)+T(3n/4)+n, & if\ n\geq 4\\ 1, & if\ n<4\end{cases}$$



递归树法：实例

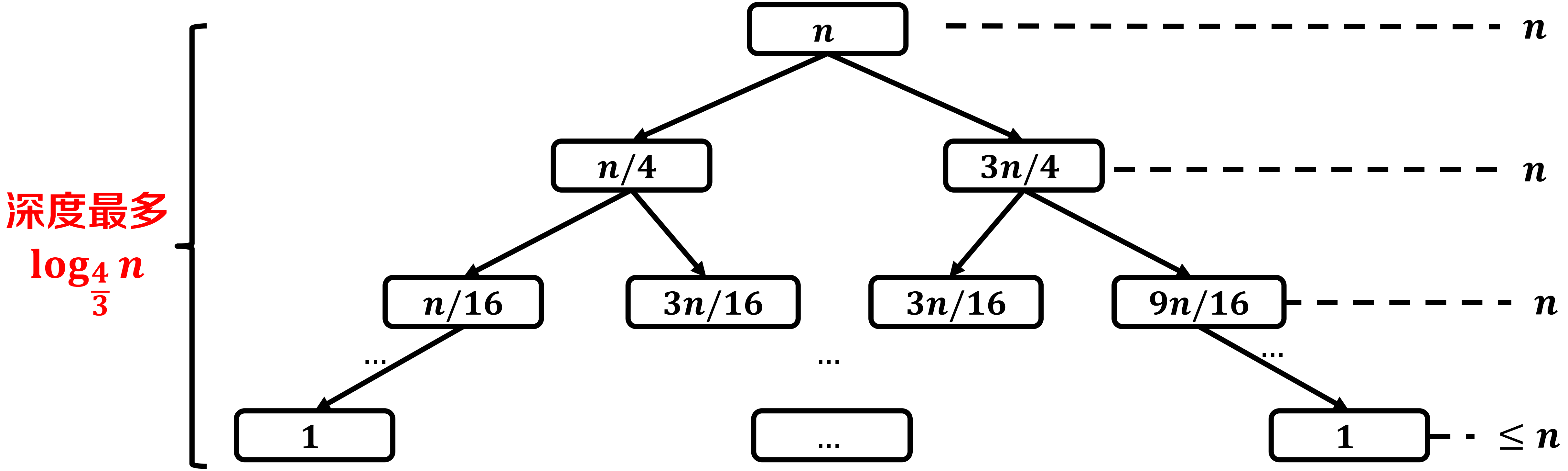
$$T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & \text{if } n \geq 4 \\ 1, & \text{if } n < 4 \end{cases}$$

深度最多 $\log_{\frac{4}{3}} n$



递归树法：实例

$$T(n)=\begin{cases}T(n/4)+T(3n/4)+n, & if\ n\ge 4\\ 1, & if\ n<4\end{cases}$$



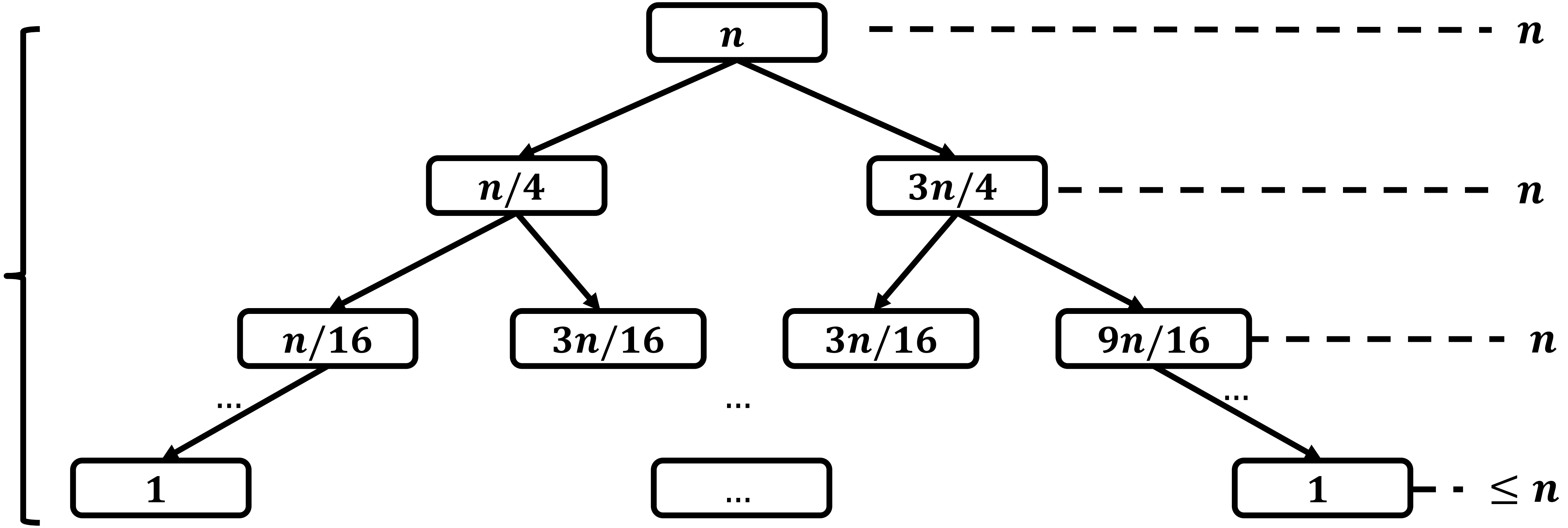
$$T(n)=O(n\log_{\frac{4}{3}} n)$$

递归树法：实例

$$T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & if\ n \geq 4 \\ 1, & if\ n < 4 \end{cases}$$

深度最多

$\log_{\frac{4}{3}} n$



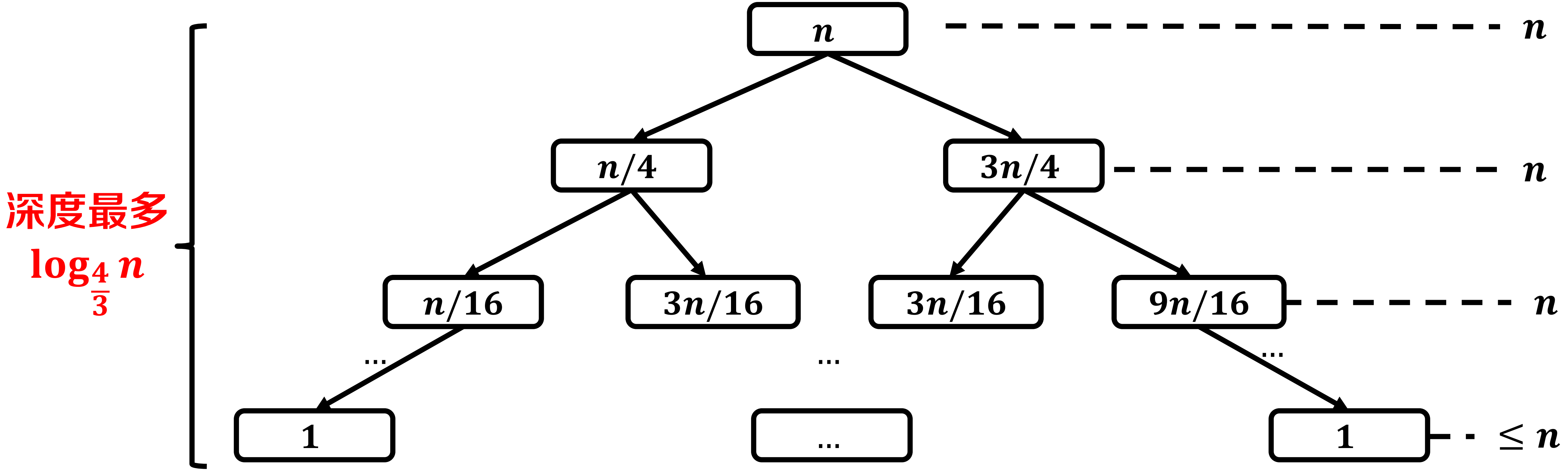
$$T(n) = O(n \log_{\frac{4}{3}} n) = O\left(n \frac{\log_2 n}{\log_2 \frac{4}{3}}\right) = O(n \log n)$$

对数换底公式

$$\log_x N = \frac{\log_y N}{\log_y x}$$

递归树法：实例

$$T(n)=\begin{cases}T(n/4)+T(3n/4)+n, & \text{if } n\geq 4\\ 1, & \text{if } n<4\end{cases}$$



$$T(n)=O(n\log n)$$

问题：该界是否为渐进紧确界?

递归式分析方法

递归树法

代入法

主定理法

代入法：实例

- $$T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & n \geq 4 \\ 1, & n < 4 \end{cases}$$

代入法：实例

- $T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & n \geq 4 \\ 1, & n < 4 \end{cases}$
- 猜测：$T(n) = \Theta(n \log n)$

代入法：实例

- $T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & n \geq 4 \\ 1, & n < 4 \end{cases}$
- 猜测：$T(n) = \Theta(n \log n)$
  - 即证明$\exists c_1, c_2, n_0 > 0$，使得$\forall n > n_0$, $c_1 \cdot n \log n \leq T(n) \leq c_2 \cdot n \log n$

Θ记号

定义：

- 对于给定的函数$g(n)$，$\Theta(g(n))$表示以下函数的集合：

  $\Theta(g(n)) = \{T(n): \exists\, c_1, c_2, n_0 > 0, \text{使得} \forall\, n \geq n_0, c_1 g(n) \leq T(n) \leq c_2 g(n)\}$

代入法：实例

- $T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & n \geq 4 \\ 1, & n < 4 \end{cases}$
- 猜测：$T(n) = \Theta(n \log n)$
  - 即证明$\exists c_1, c_2, n_0 > 0$，使得$\forall n > n_0$, $c_1 \cdot n \log n \leq T(n) \leq c_2 \cdot n \log n$

使用数学归纳法证明该命题

代入法：实例

- $T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & n \geq 4 \\ 1, & n < 4 \end{cases}$
- 猜测：$T(n) = \Theta(n \log n)$
  - 即证明$\exists c_1, c_2, n_0 > 0$，使得$\forall n > n_0$, $c_1 \cdot n \log n \leq T(n) \leq c_2 \cdot n \log n$
- 数学归纳法
  - $n = 3$时：使$c_1 \cdot 3 \log 3 \leq 1 \leq c_2 \cdot 3 \log 3$，需取$0 < c_1 \leq \frac{1}{3 \log 3}, c_2 \geq \frac{1}{3 \log 3}$

代入法：实例

- $T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & n \geq 4 \\ 1, & n < 4 \end{cases}$
- 猜测：$T(n) = \Theta(n\log n)$
  - 即证明$\exists c_1, c_2, n_0 > 0$，使得$\forall n > n_0$, $c_1 \cdot n\log n \leq T(n) \leq c_2 \cdot n\log n$
- 数学归纳法
  - $n = 3$时：使$c_1 \cdot 3\log 3 \leq 1 \leq c_2 \cdot 3\log 3$，需取$0 < c_1 \leq \frac{1}{3\log 3}, c_2 \geq \frac{1}{3\log 3}$
  - 小于n时：假设命题成立

代入法：实例

- $T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & n \geq 4 \\ 1, & n < 4 \end{cases}$
- 猜测：$T(n) = \Theta(n \log n)$
  - 即证明$\exists c_1, c_2, n_0 > 0$，使得$\forall n > n_0$, $c_1 \cdot n \log n \leq T(n) \leq c_2 \cdot n \log n$
- 数学归纳法
  - $n = 3$时：使$c_1 \cdot 3 \log 3 \leq 1 \leq c_2 \cdot 3 \log 3$，需取$0 < c_1 \leq \frac{1}{3 \log 3}, c_2 \geq \frac{1}{3 \log 3}$
  - 小于n时：假设命题成立
  - 等于n时：代入可得
    - $T(n) = T(n/4) + T(3n/4) + n \leq c_2 \cdot \frac{n}{4} \cdot \log \frac{n}{4} + c_2 \cdot \frac{3n}{4} \cdot \log \frac{3n}{4} + n$

代入法：实例

- **代入并整理表达式**

$$
\begin{aligned}
T(n) &= T(n/4) + T(3n/4) + n \\
&\le c_2 \cdot \frac{n}{4} \cdot \log\frac{n}{4} + c_2 \cdot \frac{3n}{4} \cdot \log\frac{3n}{4} + n \\
&= \left(c_2 \cdot \frac{n}{4} \cdot (\log n - \log 4)\right) + \left(c_2 \cdot \frac{3n}{4} \cdot \left(\log n - \log\frac{4}{3}\right)\right) + n
\end{aligned}
$$

代入法：实例

- 代入并整理表达式

$$
\begin{aligned}
T(n) &= T(n/4) + T(3n/4) + n \\
&\le c_2 \cdot \frac{n}{4} \cdot \log\frac{n}{4} + c_2 \cdot \frac{3n}{4} \cdot \log\frac{3n}{4} + n \\
&= \left(c_2 \cdot \frac{n}{4} \cdot (\log n - \log 4)\right) + \left(c_2 \cdot \frac{3n}{4} \cdot \left(\log n - \log\frac{4}{3}\right)\right) + n
\end{aligned}
$$

代入法：实例

- **代入并整理表达式**

$$
\begin{aligned}
T(n) &= T(n/4) + T(3n/4) + n \\
&\le c_2 \cdot \frac{n}{4} \cdot \log\frac{n}{4} + c_2 \cdot \frac{3n}{4} \cdot \log\frac{3n}{4} + n \\
&= \left(\boxed{c_2 \cdot \frac{n}{4} \cdot (\log n} - \log 4)\right) + \left(\boxed{c_2 \cdot \frac{3n}{4} \cdot \left(\log n\right.} - \left.\log\frac{4}{3}\right)\right) + n \\
&= c_2 n \log n
\end{aligned}
$$

代入法：实例

- **代入并整理表达式**

$$
\begin{aligned}
T(n) &= T(n/4) + T(3n/4) + n \\
&\le c_2 \cdot \frac{n}{4} \cdot \log\frac{n}{4} + c_2 \cdot \frac{3n}{4} \cdot \log\frac{3n}{4} + n \\
&= \left(c_2 \cdot \frac{n}{4}\ (\log n - \log 4)\right) + \left(c_2 \cdot \frac{3n}{4} \cdot \left(\log n - \log\frac{4}{3}\right)\right) + n \\
&= c_2 n \log n - \left(c_2 n \left(\frac{1}{4}\log 4 + \frac{3}{4}\log 4 - \frac{3}{4}\log 3\right)\right) + n
\end{aligned}
$$

代入法：实例

- **代入并整理表达式**

$$
\begin{aligned}
T(n) &= T(n/4) + T(3n/4) + n \\
&\leq c_2 \cdot \frac{n}{4} \cdot \log\frac{n}{4} + c_2 \cdot \frac{3n}{4} \cdot \log\frac{3n}{4} + n \\
&= \left(c_2 \cdot \frac{n}{4} \cdot (\log n - \log 4)\right) + \left(c_2 \cdot \frac{3n}{4} \cdot \left(\log n - \log\frac{4}{3}\right)\right) + n \\
&= c_2 n \log n - \left(c_2 n \left(\frac{1}{4}\log 4 + \frac{3}{4}\log 4 - \frac{3}{4}\log 3\right)\right) + n
\end{aligned}
$$

代入法：实例

- 代入并整理表达式

$$
\begin{aligned}
T(n) &= T(n/4) + T(3n/4) + n \\
&\le c_2 \cdot \frac{n}{4} \cdot \log \frac{n}{4} + c_2 \cdot \frac{3n}{4} \cdot \log \frac{3n}{4} + n \\
&= \left(c_2 \cdot \frac{n}{4} \cdot (\log n - \log 4)\right) + \left(c_2 \cdot \frac{3n}{4} \cdot \left(\log n - \log \frac{4}{3}\right)\right) + n \\
&= c_2 n \log n - \left(c_2 n \left(\boxed{\frac{1}{4}\log 4 + \frac{3}{4}\log 4} - \frac{3}{4}\log 3\right)\right) + n \\
&= c_2 n \log n - \left(c_2 \left(\boxed{\log 4} - \frac{3}{4}\log 3\right) - 1\right) n
\end{aligned}
$$

代入法：实例

- **代入并整理表达式**

$$
\begin{aligned}
T(n) &= T(n/4) + T(3n/4) + n \\
&\le c_2 \cdot \frac{n}{4} \cdot \log\frac{n}{4} + c_2 \cdot \frac{3n}{4} \cdot \log\frac{3n}{4} + n \\
&= \left(c_2 \cdot \frac{n}{4} \cdot (\log n - \log 4)\right) + \left(c_2 \cdot \frac{3n}{4} \cdot \left(\log n - \log\frac{4}{3}\right)\right) + n \\
&= c_2 n \log n - \left(c_2 \boxed{n} \left(\frac{1}{4}\log 4 + \frac{3}{4}\log 4 - \frac{3}{4}\log 3\right)\right) + \boxed{n} \\
&= c_2 n \log n - \left(c_2 \left(\log 4 - \frac{3}{4}\log 3\right) - 1\right) \boxed{n}
\end{aligned}
$$

代入法：实例

- 代入并整理表达式

$$
\begin{aligned}
T(n) &= T(n/4) + T(3n/4) + n \\
&\le c_2 \cdot \frac{n}{4} \cdot \log \frac{n}{4} + c_2 \cdot \frac{3n}{4} \cdot \log \frac{3n}{4} + n \\
&= \left(c_2 \cdot \frac{n}{4} \cdot (\log n - \log 4)\right) + \left(c_2 \cdot \frac{3n}{4} \cdot \left(\log n - \log \frac{4}{3}\right)\right) + n \\
&= c_2 n \log n - \left(c_2 n \left(\frac{1}{4}\log 4 + \frac{3}{4}\log 4 - \frac{3}{4}\log 3\right)\right) + n \\
&= c_2 n \log n - \left(c_2 \left(\log 4 - \frac{3}{4}\log 3\right) - 1\right) n
\end{aligned}
$$

代入法：实例

- **代入并整理表达式**

$$
\begin{aligned}
T(n) &= T(n/4) + T(3n/4) + n \\
&\leq c_2 \cdot \frac{n}{4} \cdot \log \frac{n}{4} + c_2 \cdot \frac{3n}{4} \cdot \log \frac{3n}{4} + n \\
&= \left(c_2 \cdot \frac{n}{4} \cdot (\log n - \log 4)\right) + \left(c_2 \cdot \frac{3n}{4} \cdot \left(\log n - \log \frac{4}{3}\right)\right) + n \\
&= c_2 n \log n - \left(c_2 n \left(\frac{1}{4}\log 4 + \frac{3}{4}\log 4 - \frac{3}{4}\log 3\right)\right) + n \\
&= \boxed{c_2 n \log n - \left(c_2\left(\log 4 - \frac{3}{4}\log 3\right) - 1\right) n}
\end{aligned}
$$

希望此式$\leq c_2 n \log n$

代入法：实例

- 代入并整理表达式

$$
\begin{aligned}
T(n) &= T(n/4) + T(3n/4) + n \\
&\leq c_2 \cdot \frac{n}{4} \cdot \log \frac{n}{4} + c_2 \cdot \frac{3n}{4} \cdot \log \frac{3n}{4} + n \\
&= \left(c_2 \cdot \frac{n}{4} \cdot (\log n - \log 4)\right) + \left(c_2 \cdot \frac{3n}{4} \cdot \left(\log n - \log \frac{4}{3}\right)\right) + n \\
&= c_2 n \log n - \left(c_2 n \left(\frac{1}{4}\log 4 + \frac{3}{4}\log 4 - \frac{3}{4}\log 3\right)\right) + n \\
&= c_2 n \log n - \boxed{\left(c_2\left(\log 4 - \frac{3}{4}\log 3\right) - 1\right) n}
\end{aligned}
$$

只需此部分≥ 0

代入法：实例

- 代入并整理表达式

$$
\begin{aligned}
T(n) &= T(n/4) + T(3n/4) + n \\
&\le c_2 \cdot \frac{n}{4} \cdot \log\frac{n}{4} + c_2 \cdot \frac{3n}{4} \cdot \log\frac{3n}{4} + n \\
&= \left(c_2 \cdot \frac{n}{4} \cdot (\log n - \log 4)\right) + \left(c_2 \cdot \frac{3n}{4} \cdot \left(\log n - \log \frac{4}{3}\right)\right) + n \\
&= c_2 n \log n - \left(c_2 n \left(\frac{1}{4}\log 4 + \frac{3}{4}\log 4 - \frac{3}{4}\log 3\right)\right) + n \\
&= c_2 n \log n - \boxed{\left(c_2\left(\log 4 - \frac{3}{4}\log 3\right) - 1\right) n}
\end{aligned}
$$

只需此部分≥ 0

- 令$\left(c_2\left(\log 4 - \frac{3}{4}\log 3\right) - 1\right) n \geq 0$，解得$c_2 \geq \frac{1}{\log 4 - \frac{3}{4}\log 3} > 0$

代入法：实例

- $T(n)=\begin{cases}T(n/4)+T(3n/4)+n, & n\geq 4\\ 1, & n<4\end{cases}$
- 猜测：$T(n)=\Theta(n\log n)$
  - 即证明$\exists c_1,c_2,n_0>0$，使得$\forall n>n_0$, $c_1\cdot n\log n\leq T(n)\leq c_2\cdot n\log n$
- 数学归纳法
  - $n=3$时：使$c_1\cdot 3\log 3\leq 1\leq c_2\cdot 3\log 3$，需取$0<c_1\leq\frac{1}{3\log 3}, c_2\geq\frac{1}{3\log 3}$
  - 小于n时：假设命题成立
  - 等于n时：代入可得
    - $T(n)=T(n/4)+T(3n/4)+n\leq c_2\cdot\frac{n}{4}\cdot\log\frac{n}{4}+c_2\cdot\frac{3n}{4}\cdot\log\frac{3n}{4}+n$
    - 若想$T(n)\leq c_2\cdot n\log n$，需取$c_2\geq\frac{1}{\log 4-\frac{3}{4}\log 3}$

代入法：实例

- $T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & n \geq 4 \\ 1, & n < 4 \end{cases}$
- 猜测：$T(n) = \Theta(n\log n)$
  - 即证明$\exists c_1, c_2, n_0 > 0$，使得$\forall n > n_0$, $c_1 \cdot n\log n \leq T(n) \leq c_2 \cdot n\log n$
- 数学归纳法
  - $n = 3$时：使$c_1 \cdot 3\log 3 \leq 1 \leq c_2 \cdot 3\log 3$，需取$0 < c_1 \leq \frac{1}{3\log 3}$, $\boxed{c_2 \geq \frac{1}{3\log 3}}$
  - 小于n时：假设命题成立
  - 等于n时：代入可得
    - $T(n) = T(n/4) + T(3n/4) + n \leq c_2 \cdot \frac{n}{4} \cdot \log \cdots 4 + c_2 \cdot \frac{\quad}{4} \cdot \log\frac{\quad}{4} + n$
    - 若想$T(n) \leq c_2 \cdot n\log n$，需取$\boxed{c_2 \geq \frac{1}{\log 4 - \frac{3}{4}\log 3}}$

两条件需同时满足

代入法：实例

- $T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & n \geq 4 \\ 1, & n < 4 \end{cases}$
- 猜测：$T(n) = \Theta(n \log n)$
  - 即证明$\exists c_1, c_2, n_0 > 0$，使得$\forall n > n_0$, $c_1 \cdot n \log n \leq T(n) \leq c_2 \cdot n \log n$
- 数学归纳法
  - $n = 3$时：使$c_1 \cdot 3 \log 3 \leq 1 \leq c_2 \cdot 3 \log 3$，需取$0 < c_1 \leq \frac{1}{3 \log 3}, c_2 \geq \frac{1}{3 \log 3}$
  - 小于n时：假设命题成立
  - 等于n时：代入可得
    - $T(n) = T(n/4) + T(3n/4) + n \leq c_2 \cdot \frac{n}{4} \cdot \log \frac{n}{4} + c_2 \cdot \frac{3n}{4} \cdot \log \frac{3n}{4} + n$
    - 若想$T(n) \leq c_2 \cdot n \log n$，需取$c_2 \geq \frac{1}{\log 4 - \frac{3}{4} \log 3}$
    - $c_2 \geq \max\left\{\frac{1}{\log 4 - \frac{3}{4} \log 3}, \frac{1}{3 \log 3}\right\}$

代入法：实例

- $T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & n \geq 4 \\ 1, & n < 4 \end{cases}$
- 猜测：$T(n) = \Theta(n \log n)$
  - 即证明$\exists c_1, c_2, n_0 > 0$，使得$\forall n > n_0$, $c_1 \cdot n \log n \leq T(n) \leq c_2 \cdot n \log n$
- 数学归纳法
  - $n = 3$时：使$c_1 \cdot 3 \log 3 \leq 1 \leq c_2 \cdot 3 \log 3$，需取$0 < c_1 \leq \frac{1}{3 \log 3}, c_2 \geq \frac{1}{3 \log 3}$
  - 小于n时：假设命题成立
  - 等于n时：代入可得
    - 取$c_2 \geq \max\left\{\frac{1}{\log 4 - \frac{3}{4}\log 3}, \frac{1}{3 \log 3}\right\}$，可得$T(n) \leq c_2 \cdot n \log n$

代入法：实例

- $T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & n \geq 4 \\ 1, & n < 4 \end{cases}$
- 猜测：$T(n) = \Theta(n \log n)$
  - 即证明$\exists c_1, c_2, n_0 > 0$，使得$\forall n > n_0,\ c_1 \cdot n \log n \leq T(n) \leq c_2 \cdot n \log n$
- 数学归纳法
  - $n = 3$时：使$c_1 \cdot 3 \log 3 \leq 1 \leq c_2 \cdot 3 \log 3$，需取$0 < c_1 \leq \frac{1}{3 \log 3}, c_2 \geq \frac{1}{3 \log 3}$
  - 小于n时：假设命题成立
  - 等于n时：代入可得
    - 取$c_2 \geq \max\left\{\frac{1}{\log 4 - \frac{3}{4} \log 3}, \frac{1}{3 \log 3}\right\}$，可得$T(n) \leq c_2 \cdot n \log n$
    - 取$0 < c_1 \leq \min\left\{\frac{1}{\log 4 - \frac{3}{4} \log 3}, \frac{1}{3 \log 3}\right\}$，可得$T(n) \geq c_1 \cdot n \log n$

代入法：实例

- $T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & n \geq 4 \\ 1, & n < 4 \end{cases}$
- 猜测：$T(n) = \Theta(n \log n)$
  - 即证明$\exists c_1, c_2, n_0 > 0$，使得$\forall n > n_0$, $c_1 \cdot n \log n \leq T(n) \leq c_2 \cdot n \log n$
- 数学归纳法
  - $n = 3$时：使$c_1 \cdot 3 \log 3 \leq 1 \leq c_2 \cdot 3 \log 3$，需取$0 < c_1 \leq \frac{1}{3 \log 3}, c_2 \geq \frac{1}{3 \log 3}$
  - 小于n时：假设命题成立
  - 等于n时：代入可得
    - 取$c_2 \geq \max\left\{\frac{1}{\log 4 - \frac{3}{4} \log 3}, \frac{1}{3 \log 3}\right\}$，可得$T(n) \leq c_2 \cdot n \log n$
    - 取$0 < c_1 \leq \min\left\{\frac{1}{\log 4 - \frac{3}{4} \log 3}, \frac{1}{3 \log 3}\right\}$，可得$T(n) \geq c_1 \cdot n \log n$
  - 得证$T(n) = \Theta(n \log n)$

代入法：实例

- $T(n) = \begin{cases} T(n/4) + T(3n/4) + n, & n \geq 4 \\ 1, & n < 4 \end{cases}$
- 猜测：$T(n) = \Theta(n \log n)$
  - 即证明$\exists c_1, c_2, n_0 > 0$，使得$\forall n > n_0$, $c_1 \cdot n \log n \leq T(n) \leq c_2 \cdot n \log n$
- 数学归纳法
  - $n = 3$时：使$c_1 \cdot 3\log 3 \leq 1 \leq c_2 \cdot 3 \log 3$，需取$0 < c_1 \leq \frac{1}{3\log 3}, c_2 \geq \frac{1}{3 \log 3}$
  - 小于n时：假设命题成立
  - 等于n时：代入可得
    - 取$c_2 \geq \max\left\{\frac{1}{\log 4 - \frac{3}{4}\log 3}, \frac{1}{3\log 3}\right\}$，可得$T(n) \leq c_2 \cdot n \log n$
    - 取$0 < c_1 \leq \min\left\{\frac{1}{\log 4 - \frac{3}{4}\log 3}, \frac{1}{3\log 3}\right\}$，可得$T(n) \geq c_1 \cdot n \log n$
  - 得证$T(n) = \Theta(n \log n)$

问题：猜测解不易得时如何求解递归式？

递归式分析方法

递归树法

代入法

主定理法

递归式分析：主定理法

- 对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式

常数$a \geq 1, b > 1$

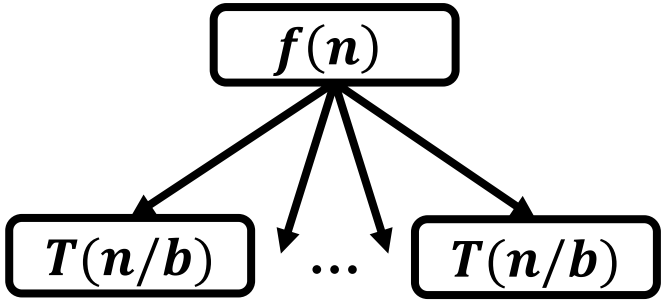
递归式分析：主定理法

- 对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式

$$\boxed{T(n)}$$

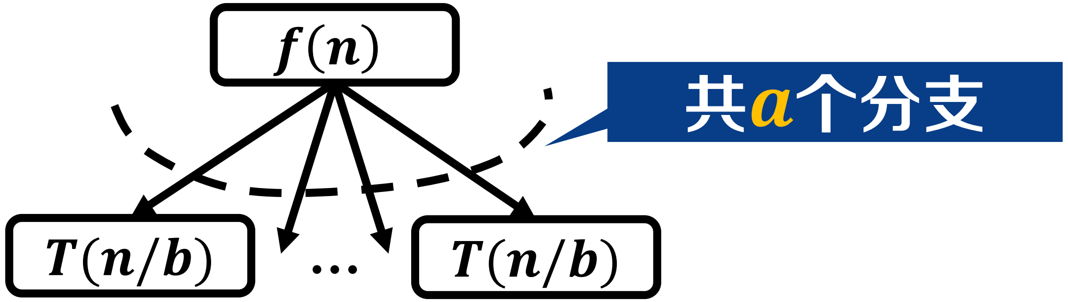
递归式分析：主定理法

- 对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式



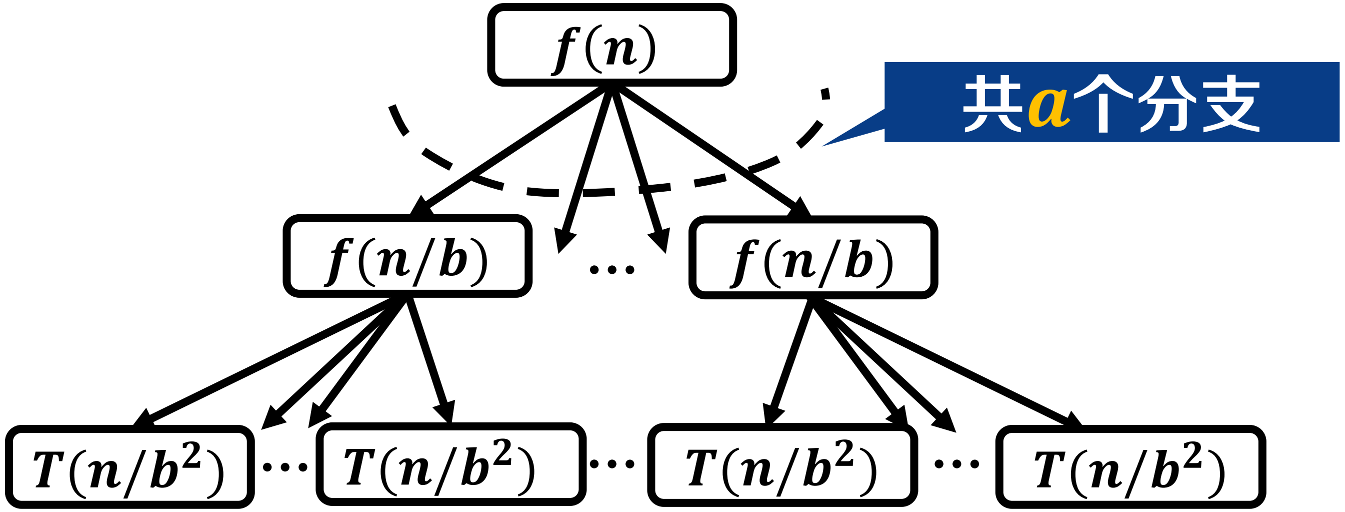
递归式分析：主定理法

- 对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式



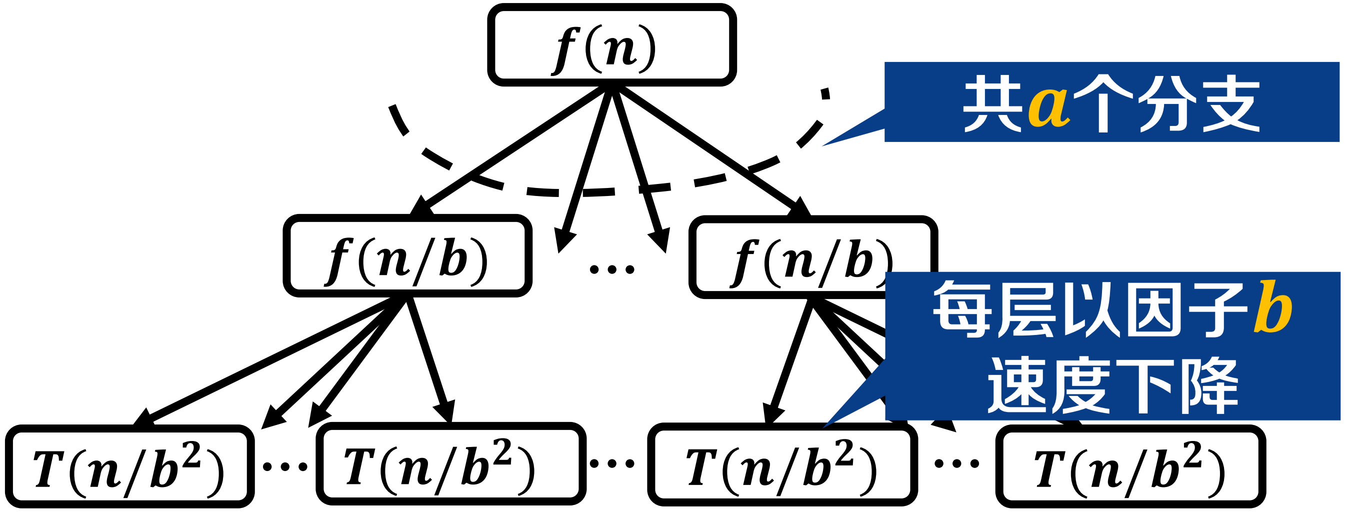
递归式分析：主定理法

- 对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式



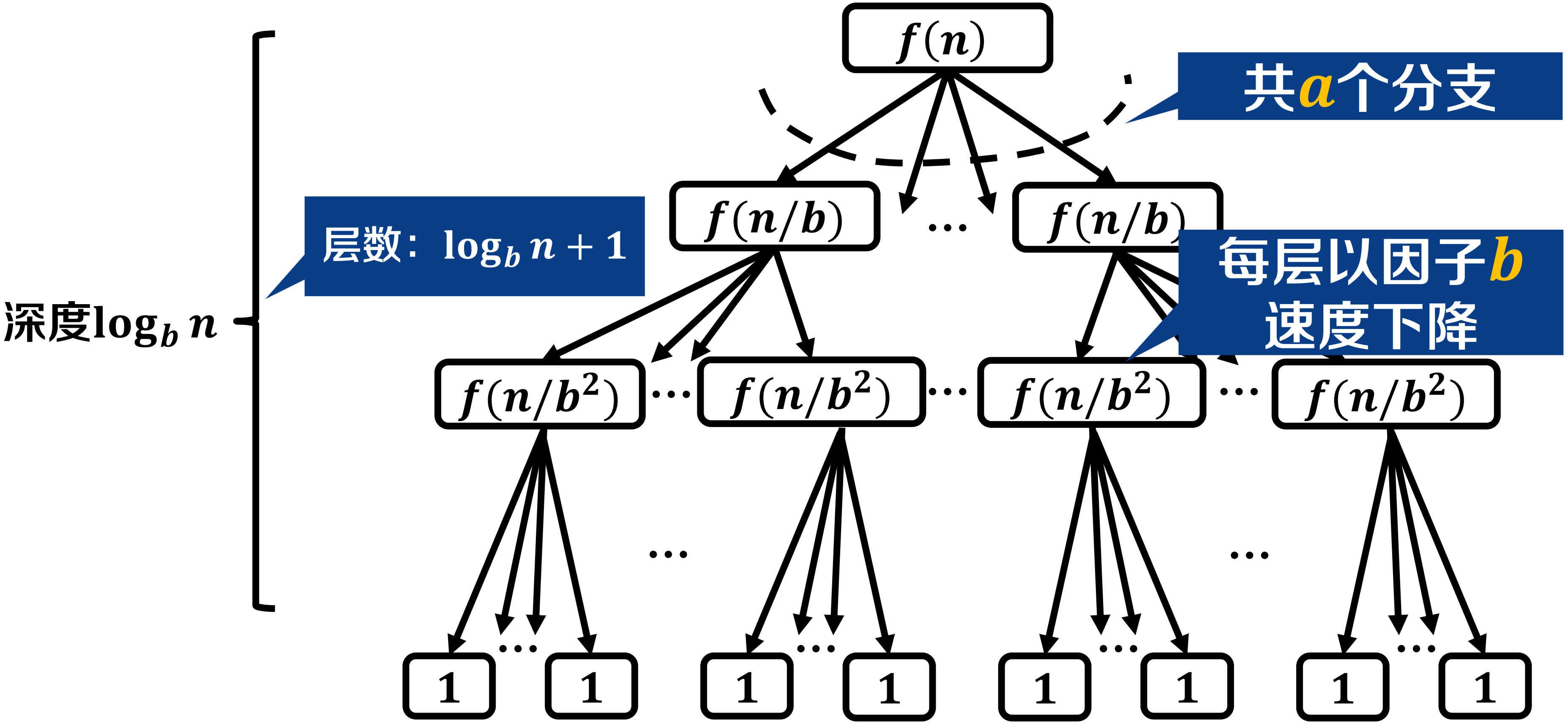
递归式分析：主定理法

- 对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式



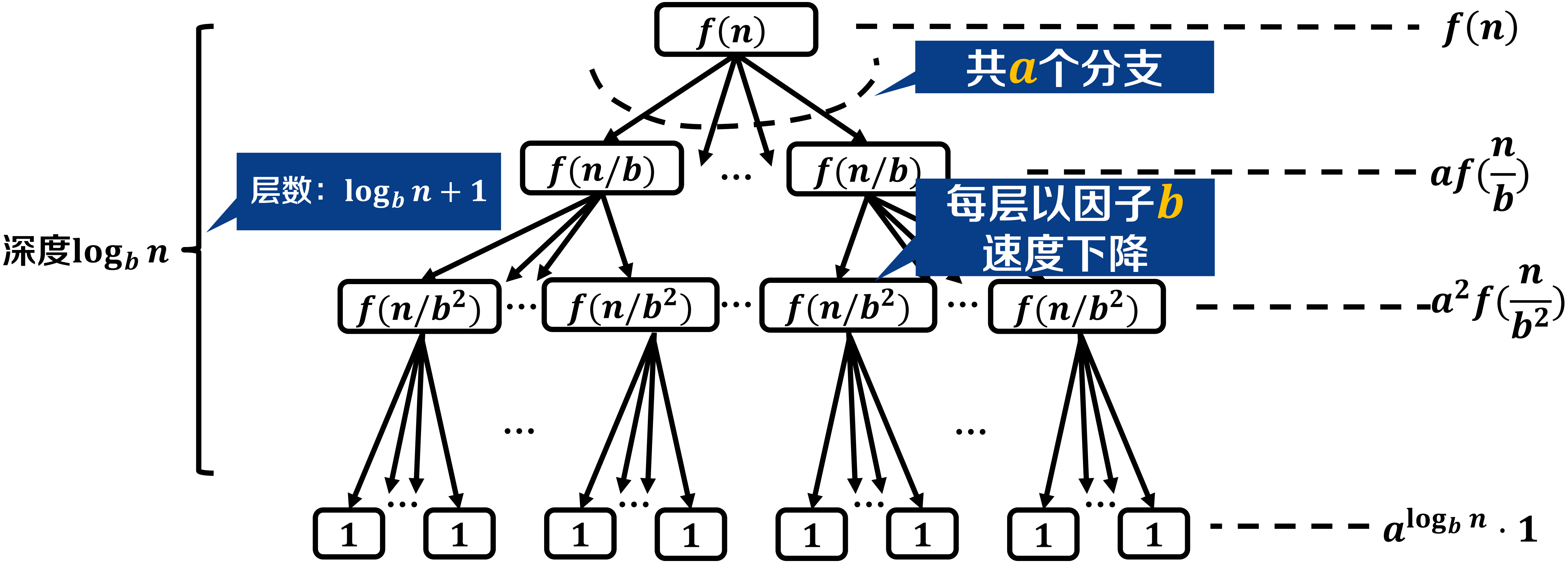
递归式分析：主定理法

- 对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式



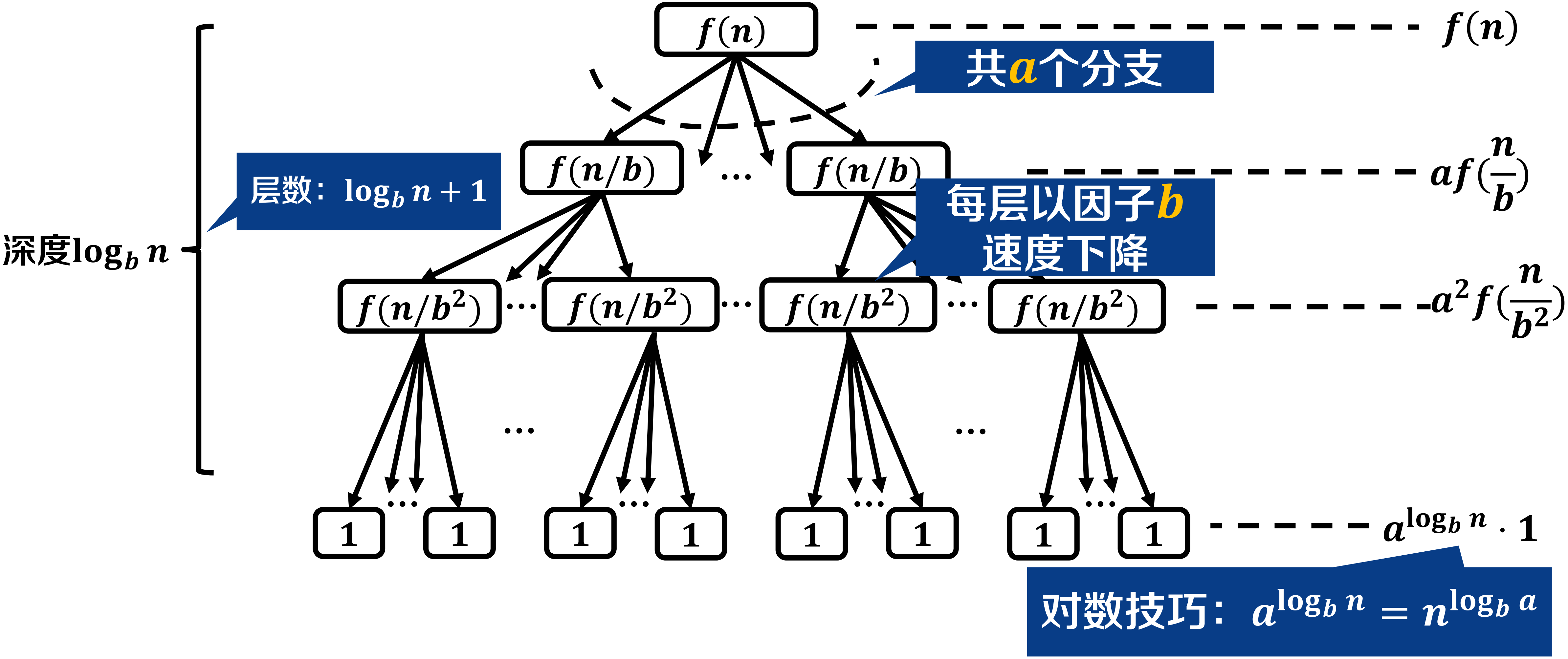
递归式分析：主定理法

- 对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式



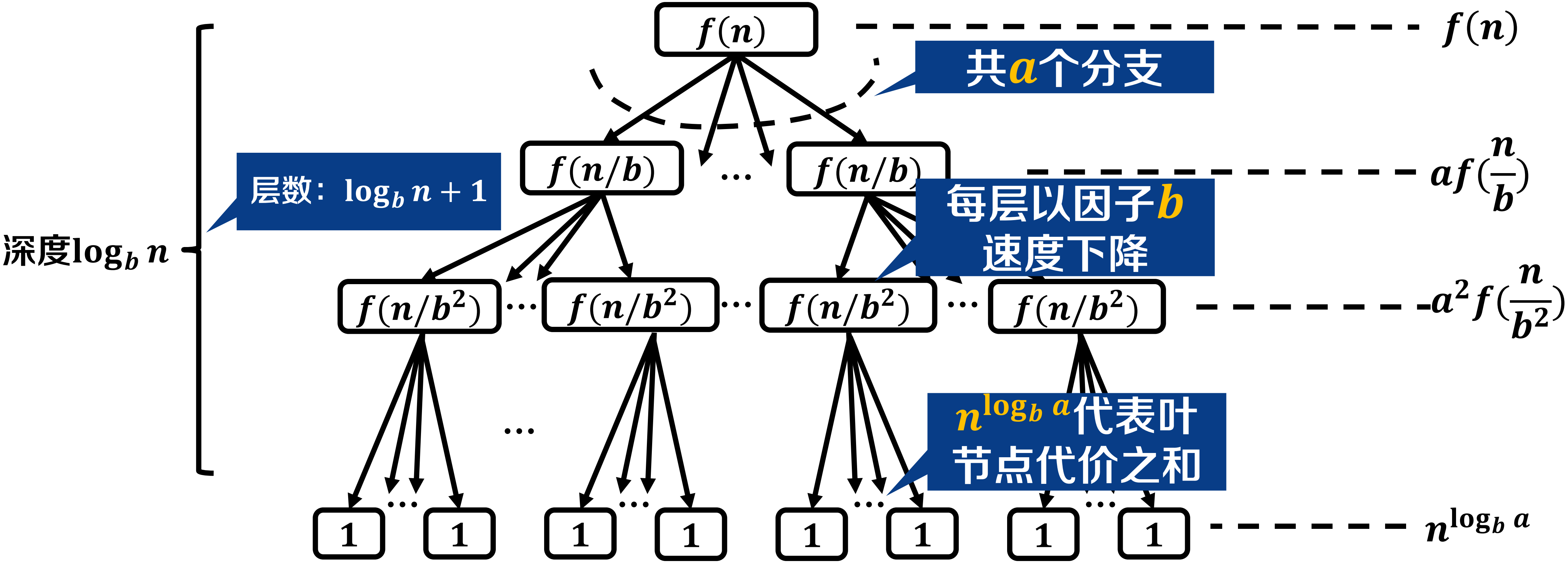
递归式分析：主定理法

- 对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式



递归式分析：主定理法

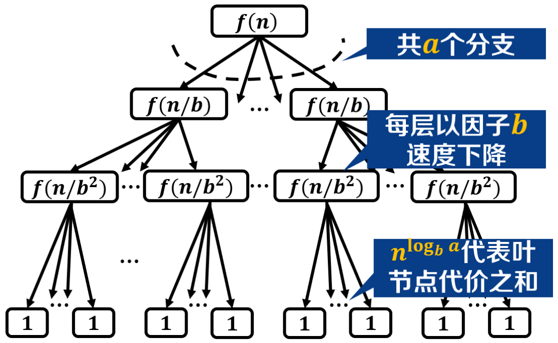
- 对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式



$$T(n) = \Theta\left(n^{\log_b a}\right) + \sum_{i=0}^{\log_b n-1} a^i f\left(\frac{n}{b^i}\right)$$

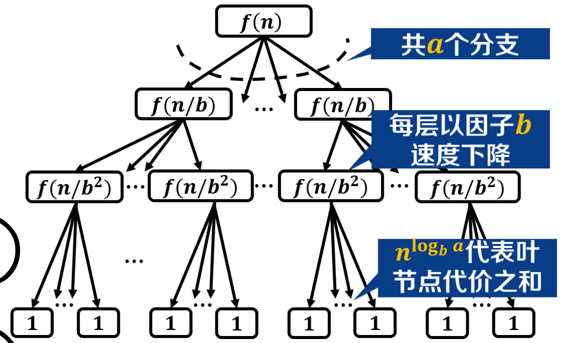
递归式分析：主定理法

- 主定理：对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式



$$T(n) = \Theta\left(n^{\log_b a}\right) + \sum_{i=0}^{\log_b n-1} a^i f\left(\frac{n}{b^i}\right)$$

递归式分析：主定理法



$T(n) = \Theta\left(n^{\log_b a}\right) + \sum_{i=0}^{\log_b n-1} a^i f\left(\frac{n}{b^i}\right)$

- 主定理：对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式

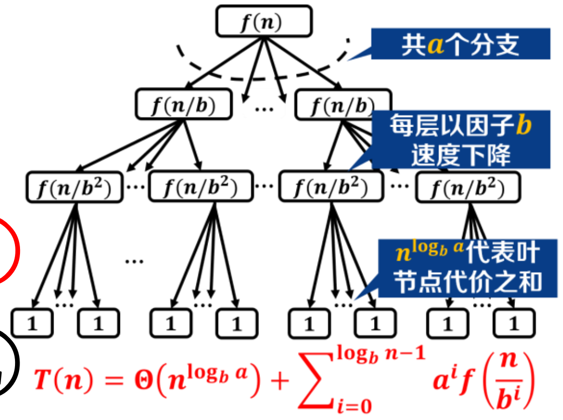
$$T(n) = \begin{cases} \Theta(f(n)) & if\ f(n) = \Omega\left(n^{\log_b a+\epsilon}\right) \ ① \\ \Theta\left(n^{\log_b a}\log n\right) & if\ f(n) = \Theta\left(n^{\log_b a}\right) \ ② \\ \Theta\left(n^{\log_b a}\right) & if\ f(n) = O\left(n^{\log_b a-\epsilon}\right) \ ③ \end{cases}$$

比较根节点代价$f(n)$与
叶节点代价之和$n^{\log_b a}$

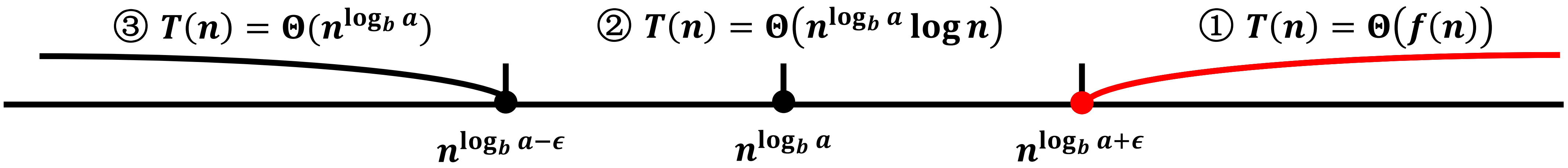
递归式分析：主定理法

- 主定理：对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式

$$T(n) = \begin{cases} \Theta\big(f(n)\big) & if\ f(n) = \Omega(n^{\log_b a+\epsilon}) \ ① \\ \Theta\big(n^{\log_b a}\log n\big) & if\ f(n) = \Theta(n^{\log_b a}) \ ② \\ \Theta\big(n^{\log_b a}\big) & if\ f(n) = O(n^{\log_b a-\epsilon}) \ ③ \end{cases}$$



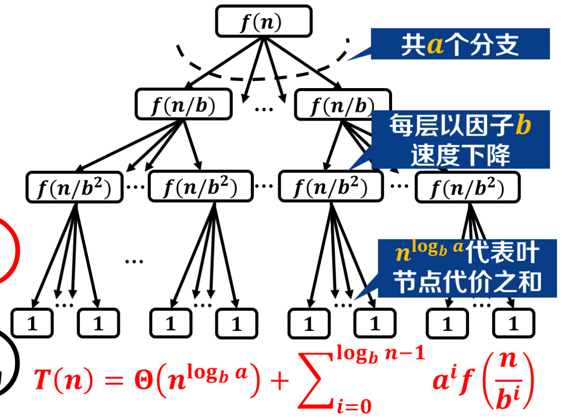
- 若存在常数$\epsilon > 0$使$f(n) = \Omega(n^{\log_b a+\epsilon})$，且存在常数$c < 1$和足够大的 n 使得 $af\left(\frac{n}{b}\right) \le cf(n)$，则$T(n) = \Theta\big(f(n)\big)$



递归式分析：主定理法

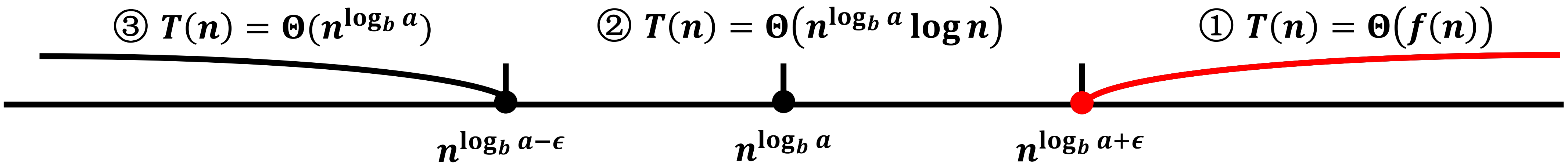
- 主定理：对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式

$$T(n) = \begin{cases} \Theta(f(n)) & \text{if } f(n) = \Omega(n^{\log_b a+\epsilon}) \ ① \\ \Theta(n^{\log_b a}\log n) & \text{if } f(n) = \Theta(n^{\log_b a}) \ ② \\ \Theta(n^{\log_b a}) & \text{if } f(n) = O(n^{\log_b a-\epsilon}) \ ③ \end{cases}$$



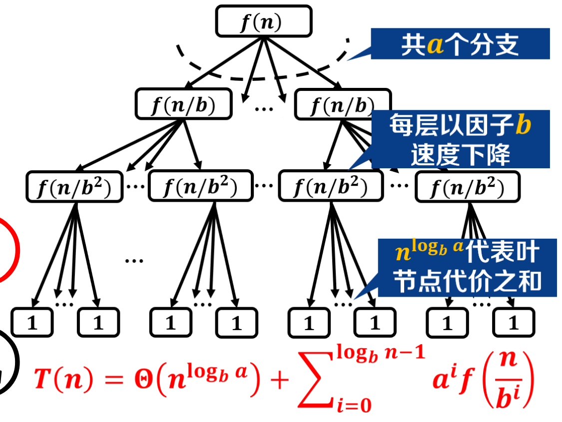
- 若存在常数$\epsilon > 0$使$f(n) = \Omega(n^{\log_b a+\epsilon})$，且存在常数$c < 1$和足够大的$n$
使得 $af\left(\frac{n}{b}\right) \le cf($

$f(n)$多项式意义大于$n^{\log_b a}$：
不止渐进大于且相差因子n^{ϵ}



递归式分析：主定理法

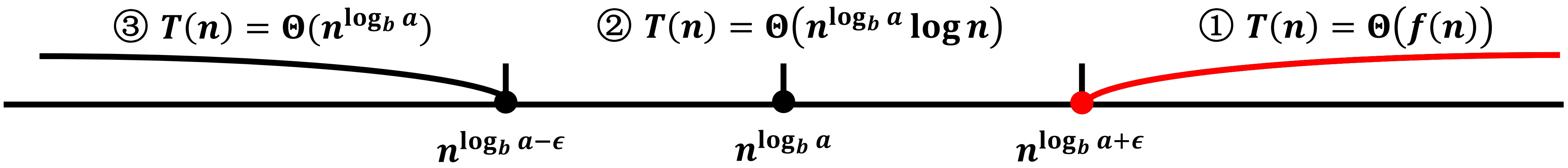
- 主定理：对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式



$$T(n) = \begin{cases} \Theta\left(f(n)\right) & if\ f(n) = \Omega\left(n^{\log_b a+\epsilon}\right) \quad ① \\ \Theta\left(n^{\log_b a}\log n\right) & if\ f(n) = \Theta\left(n^{\log_b a}\right) \quad ② \\ \Theta\left(n^{\log_b a}\right) & if\ f(n) = O\left(n^{\log_b a-\epsilon}\right) \quad ③ \end{cases}$$

- 若存在常数$\epsilon > 0$使$f(n) = \Omega(n^{\log_b a+\epsilon})$，且存在常数$c < 1$和足够大的 n 使得 $af\left(\frac{n}{b}\right) \le cf(n)$，则$T(n) = \Theta\left(f(n)\right)$

称为“正则”条件



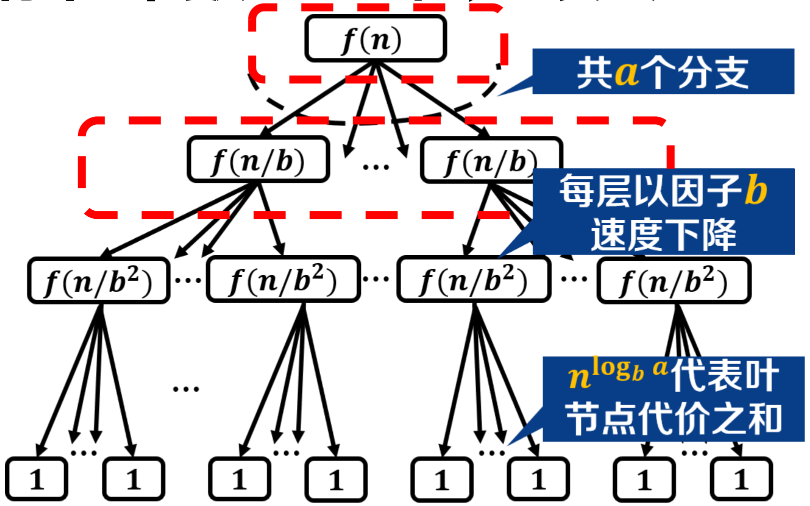
递归式分析：主定理法

- 主定理：对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式

$$T(n) = \begin{cases} \Theta(f(n)) & if\ f(n) = \Omega(n^{\log_b a+\epsilon}) \text{ ①} \\ \Theta(n^{\log_b a}\log n) & if\ f(n) = \Theta(n^{\log_b a}) \text{ ②} \\ \Theta(n^{\log_b a}) & if\ f(n) = O(n^{\log_b a-\epsilon}) \text{ ③} \end{cases}$$

- 若存在常数$\epsilon > 0$使$f(n) = \Omega(n^{\log_b a+\epsilon})$，且存在常数$c < 1$和足够大的 n 使得 $af\left(\frac{n}{b}\right) \le cf(n)$，则$T(n) = \Theta(f(n))$

保证了根节点代价
大于下一层代价之和

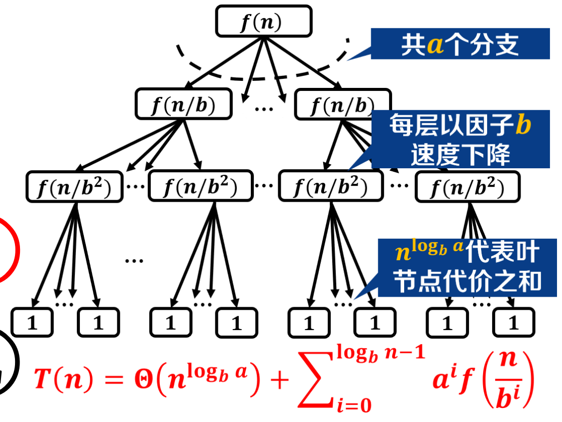


$$T(n) = \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n-1} a^i f\left(\frac{n}{b^i}\right)$$

递归式分析：主定理法

- 主定理：对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式

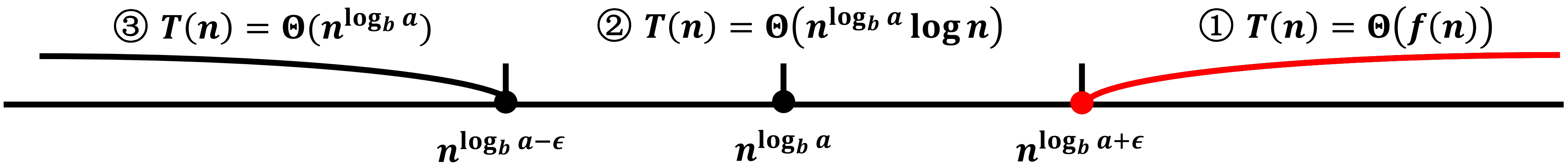
$$T(n) = \begin{cases} \Theta\left(f(n)\right) & if\ f(n) = \Omega\left(n^{\log_b a+\epsilon}\right) \ ① \\ \Theta\left(n^{\log_b a}\log n\right) & if\ f(n) = \Theta\left(n^{\log_b a}\right) \ ② \\ \Theta\left(n^{\log_b a}\right) & if\ f(n) = O\left(n^{\log_b a-\epsilon}\right) \ ③ \end{cases}$$



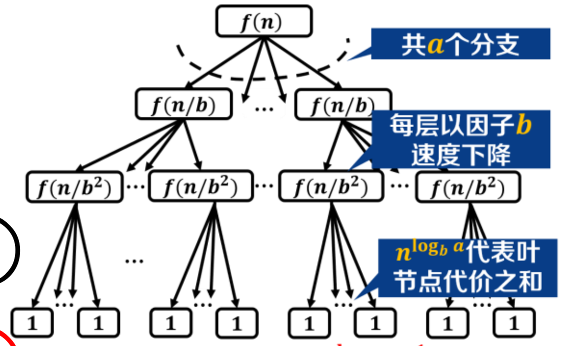
- 若存在常数$\epsilon > 0$使$f(n) = \Omega(n^{\log_b a+\epsilon})$，且存在常数$c < 1$和足够大的 n 使得 $af\left(\frac{n}{b}\right) \le cf(n)$，则$T(n) = \Theta\left(f(n)\right)$

称为“正则”条件

保证了根节点代价大于下一层代价之和



递归式分析：主定理法

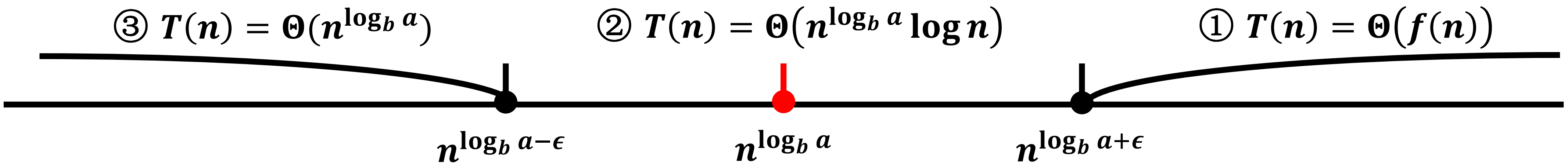


- 主定理：对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式

$$T(n) = \begin{cases} \Theta(f(n)) & \text{if } f(n) = \Omega(n^{\log_b a+\epsilon}) \quad ① \\ \Theta(n^{\log_b a}\log n) & \text{if } f(n) = \Theta(n^{\log_b a}) \quad ② \\ \Theta(n^{\log_b a}) & \text{if } f(n) = O(n^{\log_b a-\epsilon}) \quad ③ \end{cases}$$

$T(n) = \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n-1} a^i f\left(\frac{n}{b^i}\right)$

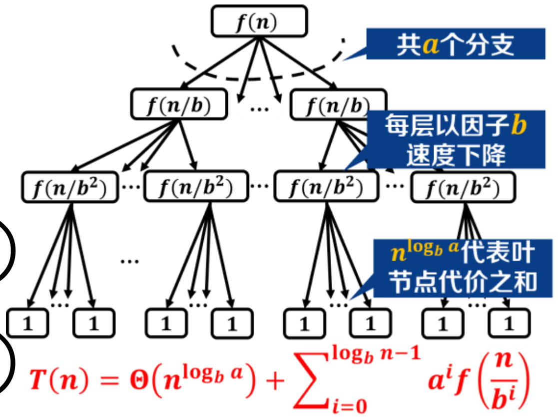
- 若$f(n) = \Theta(n^{\log_b a})$，则$T(n) = \Theta(n^{\log_b a}\log n)$



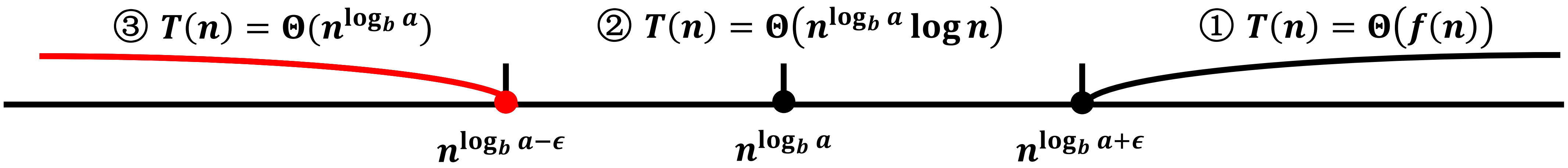
递归式分析：主定理法

- 主定理：对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式

$$T(n) = \begin{cases} \Theta(f(n)) & if\ f(n) = \Omega(n^{\log_b a+\epsilon}) \ ① \\ \Theta(n^{\log_b a}\log n) & if\ f(n) = \Theta(n^{\log_b a}) \ ② \\ \Theta(n^{\log_b a}) & if\ f(n) = O(n^{\log_b a-\epsilon}) \ ③ \end{cases}$$



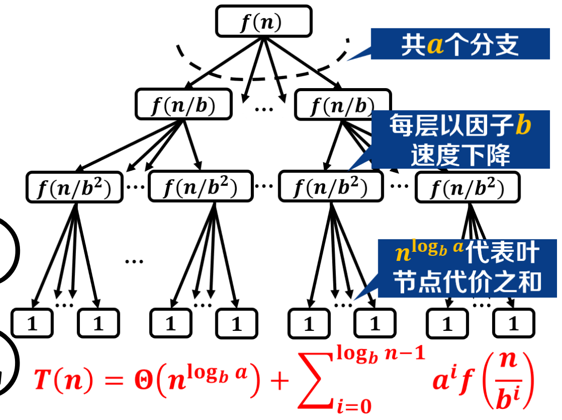
- 若存在常数$\epsilon > 0$使$f(n) = O(n^{\log_b a-\epsilon})$，则$T(n) = \Theta(n^{\log_b a})$



递归式分析：主定理法

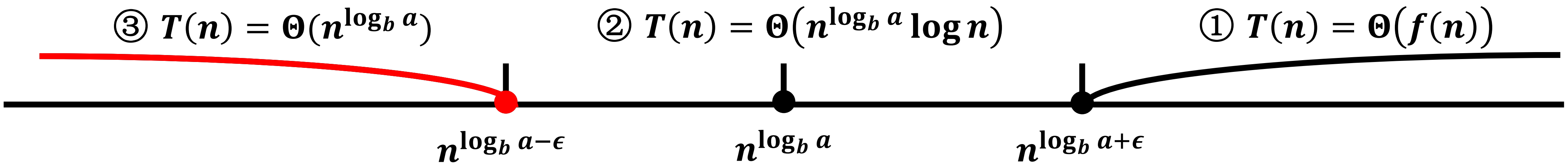
- 主定理：对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式

$$T(n) = \begin{cases} \Theta(f(n)) & if\ f(n) = \Omega(n^{\log_b a+\epsilon}) \quad ① \\ \Theta(n^{\log_b a} \log n) & if\ f(n) = \Theta(n^{\log_b a}) \quad ② \\ \Theta(n^{\log_b a}) & if\ f(n) = O(n^{\log_b a-\epsilon}) \quad ③ \end{cases}$$



- 若存在常数$\epsilon > 0$使$f(n) = O(n^{\log_b a-\epsilon})$，则$T(n) = \Theta(n^{\log_b a})$

$f(n)$多项式意义小于$n^{\log_b a}$：
不止渐进小于且相差因子n^{ϵ}

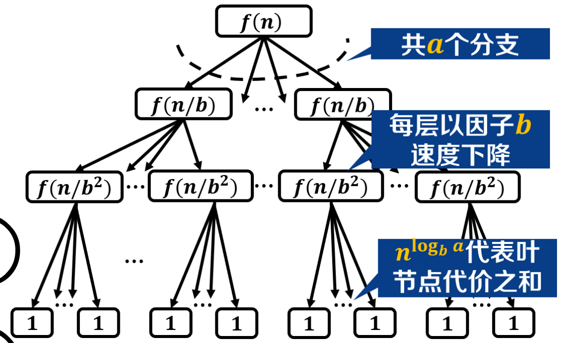


递归式分析：主定理法

- 主定理：对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式

当$f(n)$形式为n^k时，可简化主定理公式

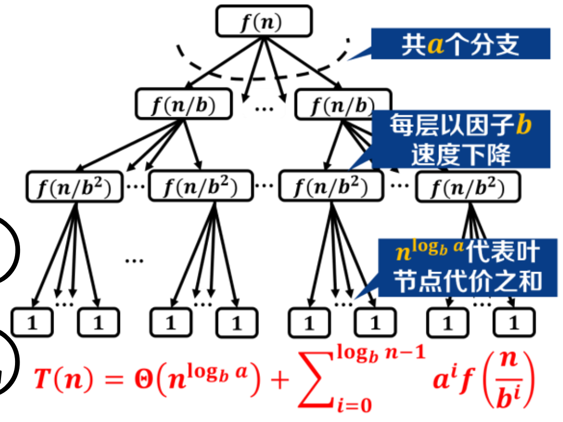
$$T(n) = \begin{cases} \Theta \quad \cdots \quad {}_b\, a+\epsilon) & ① \\ \Theta\left(n^{\log_b a} \log n\right) & if\ f(n) = \Theta\left(n^{\log_b a}\right) \quad ② \\ \Theta\left(n^{\log_b a}\right) & if\ f(n) = O\left(n^{\log_b a-\epsilon}\right) \quad ③ \end{cases}$$



$$T(n) = \Theta\left(n^{\log_b a}\right) + \sum_{i=0}^{\log_b n-1} a^i f\left(\frac{n}{b^i}\right)$$

递归式分析：主定理法

- 主定理：对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式



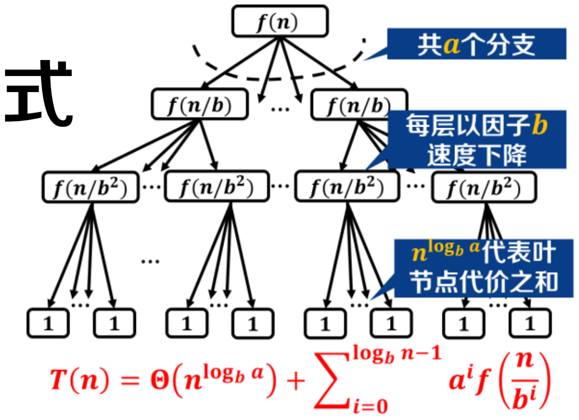
$$T(n) = \begin{cases} \Theta(f(n)) & if\ f(n) = \Omega(n^{\log_b a+\epsilon}) \quad ① \\ \Theta(n^{\log_b a}\log n) & if\ f(n) = \Theta(n^{\log_b a}) \quad ② \\ \Theta(n^{\log_b a}) & if\ f(n) = O(n^{\log_b a-\epsilon}) \quad ③ \end{cases}$$

- 主定理(简化形式)：对形如 $T(n) = aT\left(\frac{n}{b}\right) + n^k$的递归式

$$T(n) = \begin{cases} \Theta(n^k) & if\ k > \log_b a \quad ① \\ \Theta(n^k \log n) & if\ k = \log_b a \quad ② \\ \Theta(n^{\log_b a}) & if\ k < \log_b a \quad ③ \end{cases}$$

主定理法：实例一

- 主定理(简化形式)：对形如 $T(n) = aT\left(\frac{n}{b}\right) + n^k$的递归式

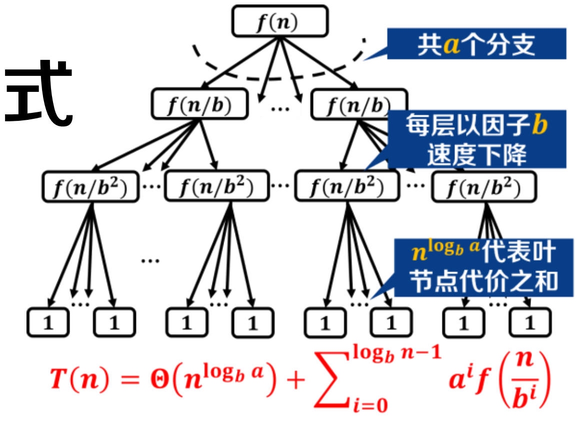


$$T(n) = \begin{cases} \Theta\left(n^k\right) & \text{if } k > \log_b a \quad ① \\ \Theta\left(n^k \log n\right) & \text{if } k = \log_b a \quad ② \\ \Theta\left(n^{\log_b a}\right) & \text{if } k < \log_b a \quad ③ \end{cases}$$

- 例一： $T(n) = 2T\left(\frac{n}{2}\right) + n$
  - $k = 1$
  - $a = 2, b = 2, \log_b a = 1$
  - $k = \log_b a$，属于情况②
  - $T(n) = \Theta\left(n^k \log n\right) = \Theta(n \log n)$

主定理法：实例二

- 主定理(简化形式)：对形如 $T(n)=aT\left(\frac{n}{b}\right)+n^k$的递归式

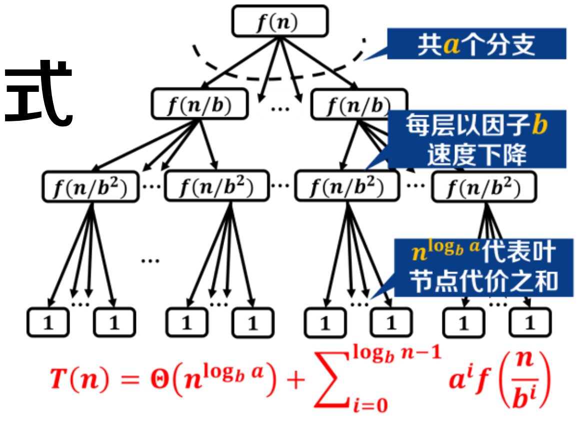


$$T(n)=\begin{cases}\Theta\left(n^k\right) & if\ k>\log_b a \quad ① \\ \Theta\left(n^k\log n\right) & if\ k=\log_b a \quad ② \\ \Theta\left(n^{\log_b a}\right) & if\ k<\log_b a \quad ③\end{cases}$$

- 例二：$T(n)=5T\left(\frac{n}{2}\right)+n^3$
  - $k=3$
  - $a=5, b=2, \log_b a=\log_2 5$
  - $k>\log_b a$，属于情况①
  - $T(n)=\Theta\left(n^k\right)=\Theta\left(n^3\right)$

主定理法：实例三

- 主定理(简化形式)：对形如 $T(n) = aT\left(\frac{n}{b}\right) + n^k$的递归式

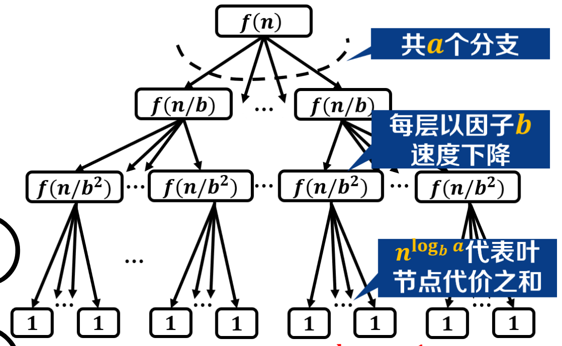


$$T(n) = \begin{cases} \Theta(n^k) & if\ k > \log_b a \quad ① \\ \Theta(n^k \log n) & if\ k = \log_b a \quad ② \\ \Theta(n^{\log_b a}) & if\ k < \log_b a \quad ③ \end{cases}$$

- 例三：$T(n) = 4T\left(\frac{n}{4}\right) + \sqrt{n}$
  - $k = \frac{1}{2}$
  - $a = 4, b = 4, \log_b a = \log_4 4 = 1$
  - $k < \log_b a$，属于情况③
  - $T(n) = \Theta(n^{\log_b a}) = \Theta(n)$

主定理法：实例四

- 主定理：对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式

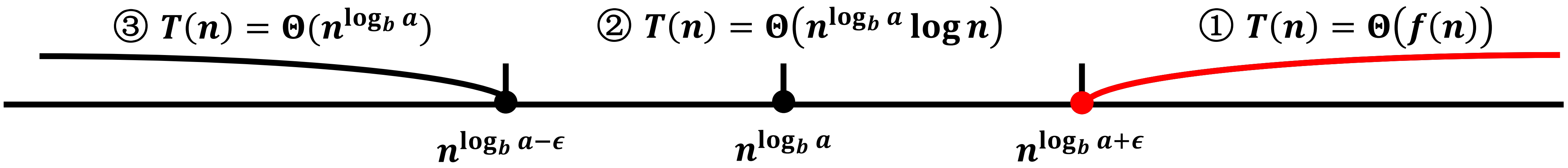


$T(n) = \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n-1} a^i f\left(\frac{n}{b^i}\right)$

$$T(n) = \begin{cases} \Theta(f(n)) & if\ f(n) = \Omega(n^{\log_b a+\epsilon}) \text{ ①} \\ \Theta(n^{\log_b a}\log n) & if\ f(n) = \Theta(n^{\log_b a}) \text{ ②} \\ \Theta(n^{\log_b a}) & if\ f(n) = O(n^{\log_b a-\epsilon}) \text{ ③} \end{cases}$$

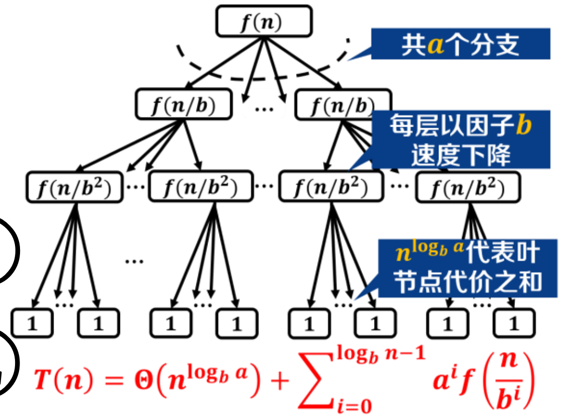
- 例四：$T(n) = 3T\left(\frac{n}{4}\right) + n\log n$
  - $\log_b a = \log_4 3 < 1$, 则$\exists\epsilon > 0$, 使得$\log_b a + \epsilon < 1$, 故$f(n) = \Omega(n^{\log_b a+\epsilon})$
  - $\exists c = \frac{3}{4}$时，$af\left(\frac{n}{b}\right) = \frac{3n}{4}\log(\frac{n}{4}) < cf(n) = \frac{3}{4}n\log n$，属于情况①
  - $T(n) = \Theta(f(n)) = \Theta(n\log n)$

“正则”条件满足



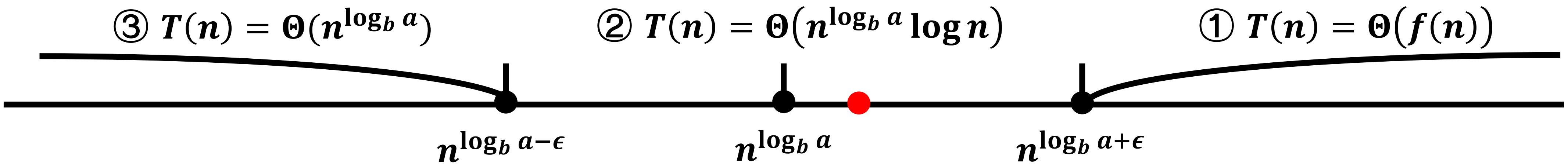
主定理法：实例五

- 主定理：对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式



$$T(n) = \begin{cases} \Theta\big(f(n)\big) & if\ f(n) = \Omega\left(n^{\log_b a+\epsilon}\right) \text{ ①} \\ \Theta\left(n^{\log_b a}\log n\right) & if\ f(n) = \Theta\left(n^{\log_b a}\right) \text{ ②} \\ \Theta\left(n^{\log_b a}\right) & if\ f(n) = O\left(n^{\log_b a-\epsilon}\right) \text{ ③} \end{cases}$$

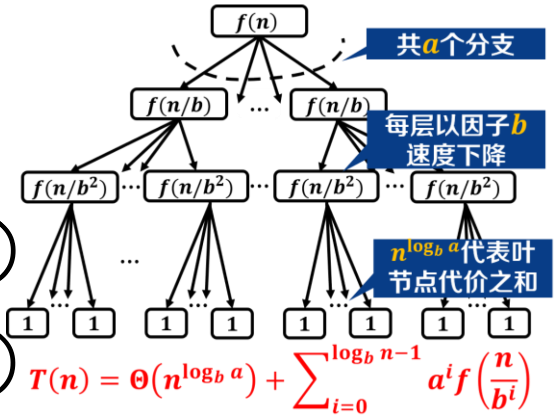
- 例五：$T(n) = 2T\left(\frac{n}{2}\right) + n\log n$
  - $\log_b a = \log_2 2 = 1, f(n) = \Omega\left(n^{\log_b a}\right)$
  - 然而对$\forall\epsilon > 0, \log n$渐进小于$n^\epsilon$,故$\nexists\epsilon > 0$使$f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$
  - 该情况落入①和②之间，不能使用主定理



主定理法：实例五

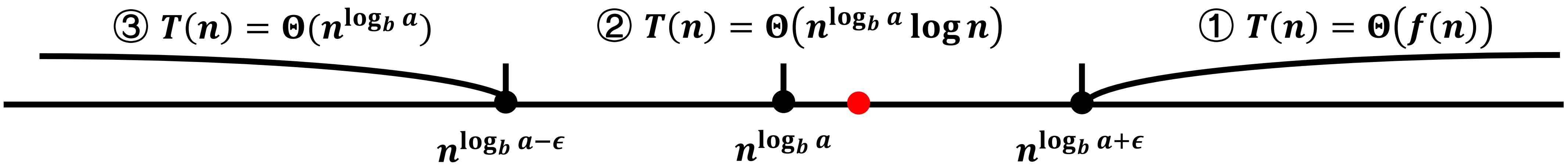
- 主定理：对形如 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$的递归式

$$T(n) = \begin{cases} \Theta(f(n)) & if\ f(n) = \Omega(n^{\log_b a+\epsilon}) \text{ ①} \\ \Theta(n^{\log_b a}\log n) & if\ f(n) = \Theta(n^{\log_b a}) \text{ ②} \\ \Theta(n^{\log_b a}) & if\ f(n) = O(n^{\log_b a-\epsilon}) \text{ ③} \end{cases}$$



上述主定理不适用
扩展形式主定理可解决

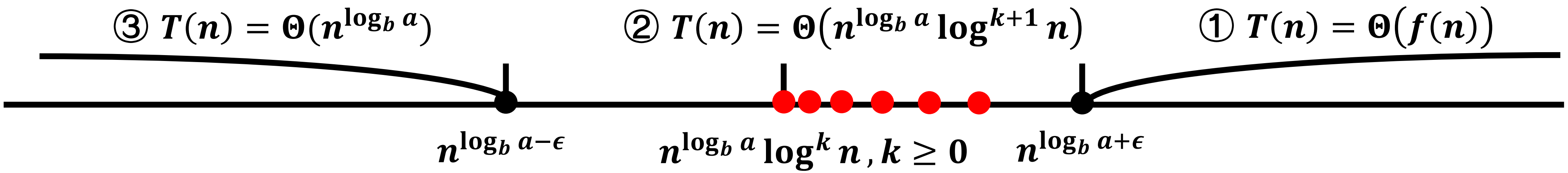
- 例五：$T(n) = 2T\left(\frac{n}{2}\right) + n\log n$
  - $\log_b a = \log_2 2 = 1, f(n) = \Omega(n^{\log_b a})$
  - 然而对$\forall\epsilon > 0, \log n$ 渐进小于n^{ϵ}, 故$\nexists\epsilon > 0$使$f(n) = \Omega(n^{\log_b a+\epsilon})$
  - 该情况落入①和②之间，不能使用主定理



递归式分析：主定理法

- 主定理(扩展形式)：对形如 $T(n)=aT\left(\frac{n}{b}\right)+f(n)$的递归式

$$T(n)=\begin{cases}\Theta(f(n)) & \text{if } f(n)=\Omega\left(n^{\log_b a+\epsilon}\right) \quad ① \\ \Theta\left(n^{\log_b a}\log^{k+1} n\right) & \text{if } f(n)=\Theta\left(n^{\log_b a}\log^{k} n\right), k\geq 0 \quad ② \\ \Theta\left(n^{\log_b a}\right) & \text{if } f(n)=O\left(n^{\log_b a-\epsilon}\right) \quad ③\end{cases}$$

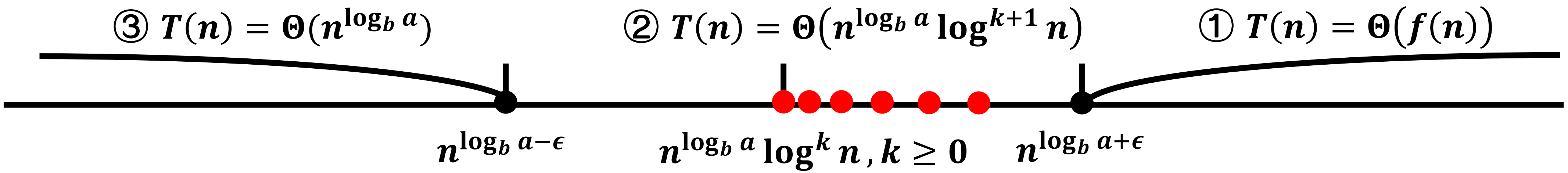


主定理法：例五

- 主定理(扩展形式)：对形如 $T(n)=aT\left(\frac{n}{b}\right)+f(n)$的递归式

$$T(n)=\begin{cases}\Theta\left(f(n)\right) & if\ f(n)=\Omega\left(n^{\log_b a+\epsilon}\right) \quad ① \\ \Theta\left(n^{\log_b a}\log^{k+1} n\right) & if\ f(n)=\Theta\left(n^{\log_b a}\log^{k} n\right), k\geq 0 \quad ② \\ \Theta\left(n^{\log_b a}\right) & if\ f(n)=O\left(n^{\log_b a-\epsilon}\right) \quad ③\end{cases}$$

- 例五：$T(n)=2T\left(\frac{n}{2}\right)+n\log n$
  - $\log_b a=\log_2 2=1$
  - $k=1, f(n)=\Theta\left(n^{\log_b a}\log^k n\right)$，属于情况②
  - $T(n)=\Theta\left(n^{\log_b a}\log^{k+1} n\right)=\Theta\left(n\log^2 n\right)$



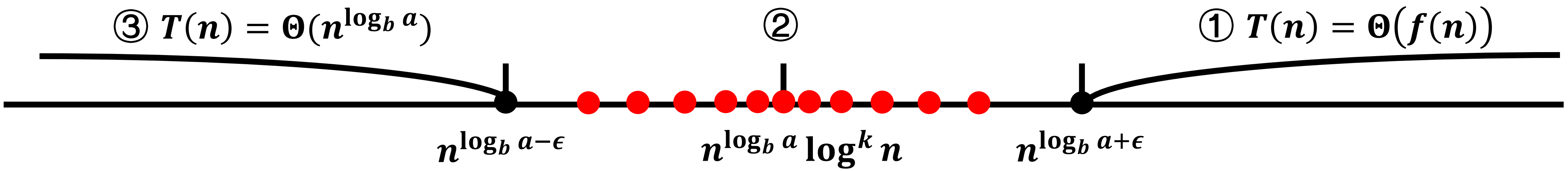
递归式分析：主定理法

- 主定理(扩展形式)：对形如 $T(n)=aT\left(\frac{n}{b}\right)+f(n)$的递归式

$$T(n)=\begin{cases}\Theta\big(f(n)\big) & if\ f(n)=\Omega(n^{\log_b a+\epsilon}) \quad ① \\ \Theta\big(n^{\log_b a}\log^{k+1} n\big) & if\ f(n)=\Theta(n^{\log_b a}\log^k n), k\geq 0 \quad ② \\ \Theta\big(n^{\log_b a}\big) & if\ f(n)=O(n^{\log_b a-\epsilon}) \quad ③\end{cases}$$

- 情况②的三种扩展

$$T(n)=\begin{cases}\Theta(n^{\log_b a}\log^{k+1} n) & k>-1 \\ \Theta\big(n^{\log_b a}\log\log n\big) & k=-1 \\ \Theta\big(n^{\log_b a}\big) & k<-1\end{cases} \quad ②$$



小结

- 递归式分析方法比较

分析方法	优点	缺点
递归树法	直观形象	难以展开
代入法	适用广泛	难猜通解
主定理法	形式简洁	适用有限

谢谢

BDA

Beihang University