算法分析与设计

Analysis and Design of Algorithm

Lesson 06

要点回顾

- 分治策略
  - 基本思想、适用条件、基本步骤
  - 分治效率分析：给出了通用计算公式
  - 两个例子：二分搜索、大整数乘法
- 递推方程的求解方法
  - 换元迭代法
  - 公式法

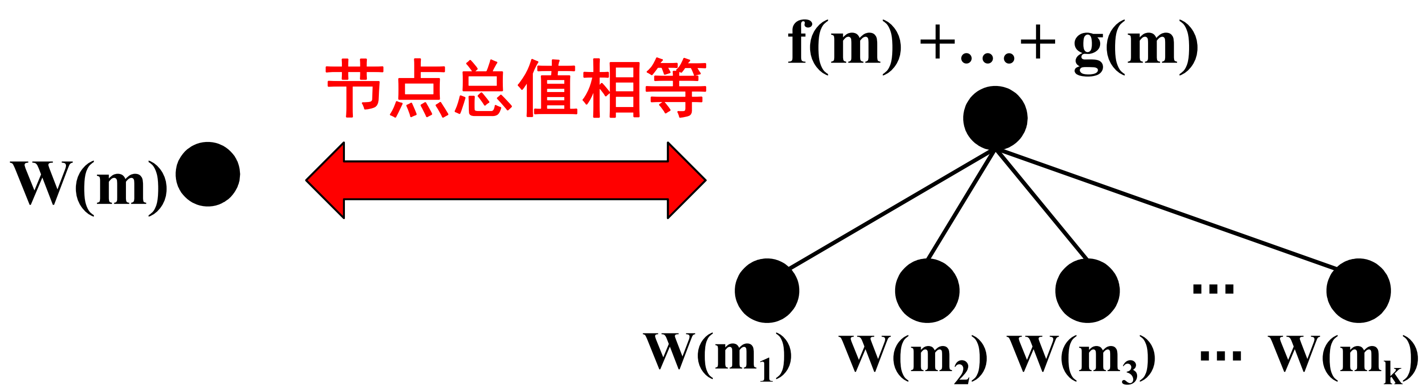
$$T(n)=\begin{cases} O(1) & n=1 \\ kT(n/m)+f(n) & n>1 \end{cases}$$

$$T(n)=n^{\log_m k}+\sum_{j=0}^{\log_m n-1} k^j f(n/m^j)$$

递归树法

- 迭代在递归树中的表示
  - 如果递归树上某节点标记为W(m)
  - $W(m)= W(m_1) +…+ W(m_k)$
    $+ f(m) +…+ g(m) , m_1,.., m_k<m$

  其中$W(m_1) ,…,W(m_k)$称为函数项

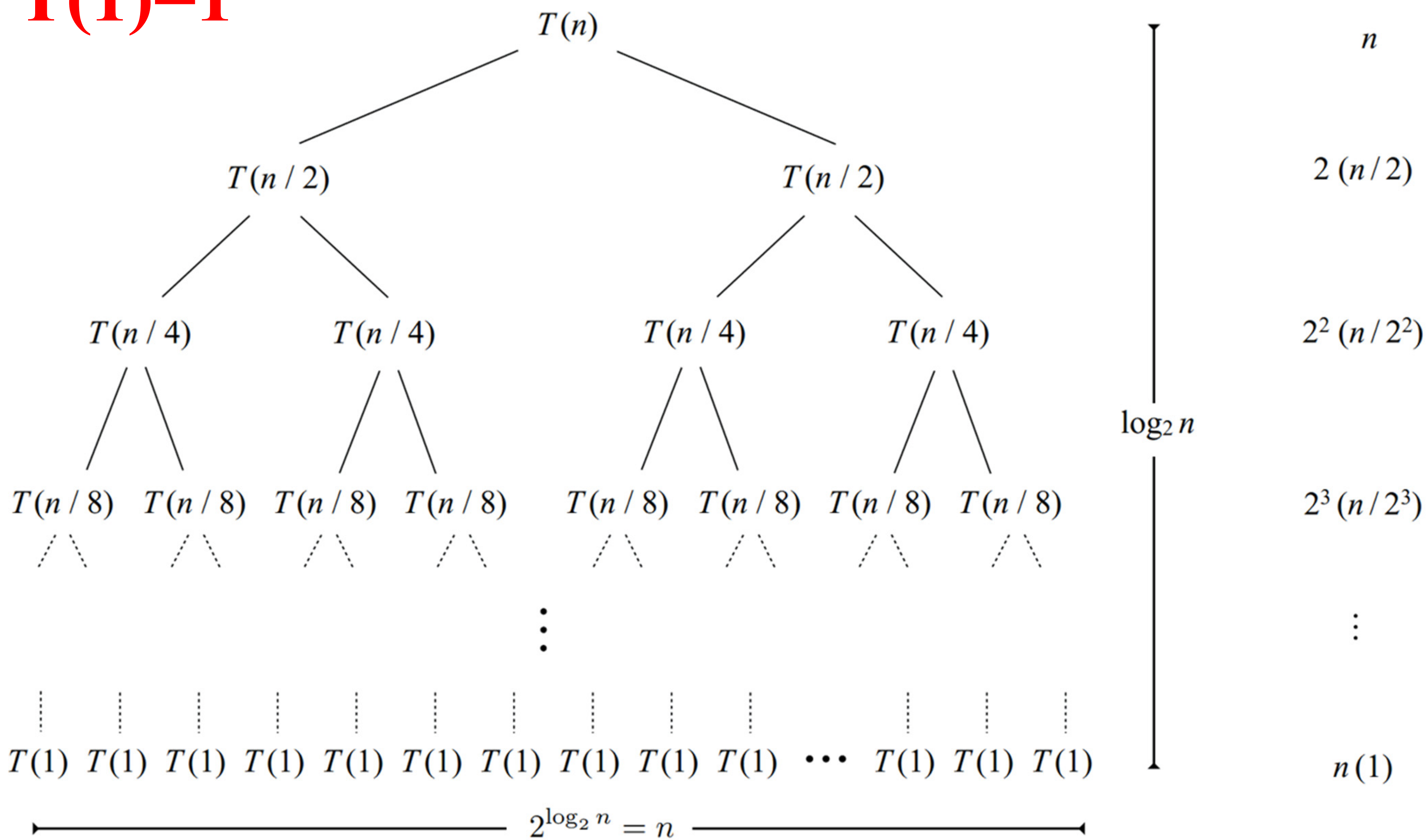


每个叶节点是一个函数项

例子： 画出$T(n)=2T(n/2)+n$递归树

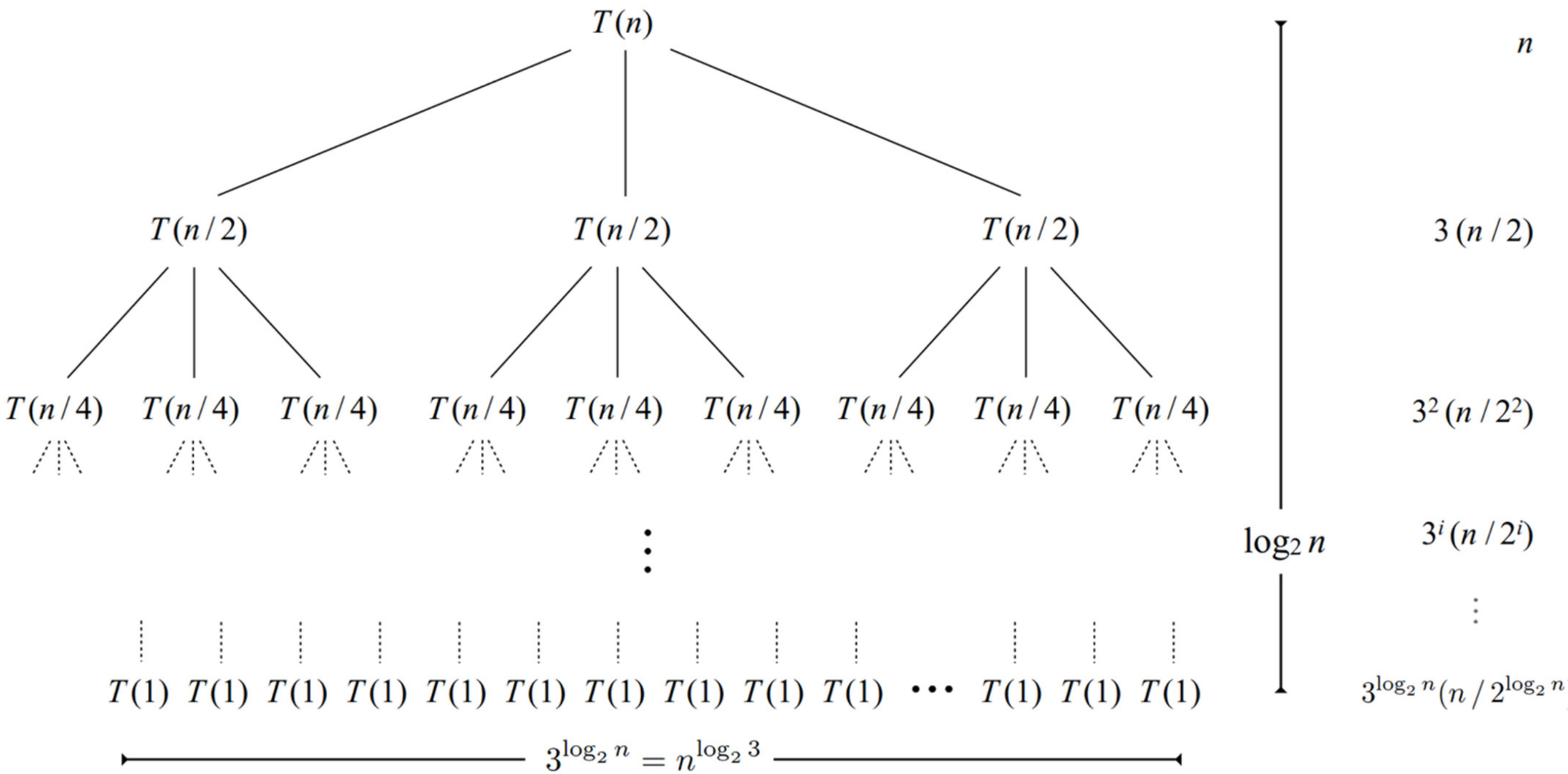
$T(1)=1$

$T(n)=2T(n/2)+n$
$T(1)=1$



$r = 1$ $$T(n) = (1 + r + r^2 + r^3 + \ldots + r^{\log_2 n})\, n \;=\; n\,(\log_2 n + 1)$$

递归树求解：T(*n*)=3T(*n*/2)+*n* ，T(1)=1



$$r = 3/2 > 1 \qquad T(n) = (1 + r + r^2 + r^3 + \ldots + r^{\log_2 n})\, n = \frac{r^{1+\log_2 n} - 1}{r - 1}\, n = 3n^{\log_2 3} - 2n$$

课堂练习

- 画出如下函数的递归树

$$T(n)=\begin{cases} O(1) & n=1 \\ 4T(n/2)+O(n) & n>1 \end{cases}$$

再举个例子

- 用递归树求解方程：

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

再次回到公式

$$T(n)=\begin{cases} O(1) & n=1 \\ aT(n/b)+f(n) & n>1 \end{cases}$$

方程的解：

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

- 第一项为所有最小子问题的计算工作量
- 第二项为迭代过程归约到子问题及合并解的工作量

哪一项更主要？

主定理(Master定理)

定理： 设$a \geqslant 1$，$b>1$为常数，$f(n)$为函数，$T(n)$为非负数，且$T(n)=aT(n/b)+f(n)$，则：

1. 若 $f(n)=O(n^{(log_b a)-\varepsilon})$，存在$\varepsilon>0$ 是常数，则有 $T(n)=\Theta(n^{log_b a})$
2. 若 $f(n)=\Theta(n^{\log_b a})$, 则有$T(n)=\Theta(n^{\log_b a}\log n)$
3. 若 $f(n)=\Omega(n^{(\log_b a)+\varepsilon})$，存在$\varepsilon>0$ 是常数，且对所有充分大的 n 有 $af(\frac{n}{b}) \le cf(n)$，$c<1$ 是常数，则有$T(n)=\Theta(f(n))$

Master定理

- **例子1：** 求解 $T(n) = 9T\left(\frac{n}{3}\right) + n$

  $a = 9, b = 3, f(n) = n, n^{\log_b a} = n^2$

  $\because f(n) = n < O\left(n^{(\log_b a)}\right) = n^2$ 取ε =1即可

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

Master定理

- 例子2：求解$T(n) = T\left(\frac{2n}{3}\right) + 1$

  $a = 1, b = \frac{3}{2}, f(n) = 1, n^{\log_b a} = n^{\log_{3/2} 1} = 1$

  $\because f(n) = 1 = \Theta(n^{\log_b a})$,

  $\therefore T(n) = \Theta\left(n^{\log_b a} \log n\right) = \Theta(\log n)$

Master定理

- 例子3：求解$T(n) = 3T\left(\frac{n}{4}\right) + n\log n$

$a = 3, b = 4, f(n) = n\log n$， $n^{\log_b a} = n^{\log_4 3} \approx n^{0.793}$

$f(n) = n\log n = \Omega(n^{\log_4 3+\varepsilon}) \approx \Omega(n^{0.793+\varepsilon})$ 取ε =0.2即可

Master定理

- **例子3：** 求解$T(n)=3T\left(\frac{n}{4}\right)+n\log n$

  条件验证 $af\left(\frac{n}{b}\right)\le cf(n)$

  $a=3, b=4, f(n)=n\log n$，代入上式

  $3\left(\frac{n}{4}\right)\log\left(\frac{n}{4}\right)\le c\times \mathrm{nlogn}$ 只要c≥3/4即可

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

递归算法分析

- **二分检索：** $T(n)=T(n/2)+1, T(1)=1$

$$a=1, b=2, n^{\log_2 1}=1, f(n)=1$$

属于第二种情况

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

不能使用主定理的例子

- 例如：求解 $T(n) = 2T(n/2) + n\log n$

$$a = b = 2, n^{\log_2 2} = n, f(n) = n\log n$$

不存在$\varepsilon > 0$使右式成立 $n\log n = \Omega(n^{1+\varepsilon})$

不存在$c < 1$使$af(\frac{n}{b}) \le cf(n)$对所有充分大的$n$成立

$$2(n/2)\log(n/2) = n(\log n - 1) \le cn\log n$$

可以考虑递归树！！！

递推方程求解方法小结

- 迭代法
- 换元迭代法
- 递归树
- 公式法
- 主定理

Strassen矩阵乘法

A和B的乘积矩阵C中的元素C[i,j]定义为:

$$C[i][j]=\sum_{k=1}^{n}A[i][k]B[k][j]$$

分析：若依此定义来计算A和B的乘积矩阵C，则每计算C的一个元素C[i][j]，需要做n次乘法和n-1次加法。因此，算出矩阵C的n^2个元素所需的计算时间为O(n^3)

Strassen矩阵乘法

分治法:

使用与大整数相乘类似的技术，将矩阵A，B和C中每一矩阵都分块成4个大小相等的子矩阵。由此可将方程C=AB重写为：

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

由此可得:

$$C_{11} = A_{11} B_{11} + A_{12} B_{21}$$
$$C_{12} = A_{11} B_{12} + A_{12} B_{22}$$
$$C_{21} = A_{21} B_{11} + A_{22} B_{21}$$
$$C_{22} = A_{21} B_{12} + A_{22} B_{22}$$

复杂度分析

1）n=2，子矩阵阶为1，8次乘和4次加，直接求出；
2）子矩阵阶大于2，为求子矩阵积可继续分块，直到子矩阵阶降为2。

此想法就产生了一个分治降阶递归算法。

两个n阶方阵的积➔8个n/2阶方阵积和4个n/2阶方阵加。

可在O(n^2)时间内完成

计算时间耗费 $T(n)=\begin{cases} O(1) & n=2 \\ 8T(n/2)+O(n^2) & n>2 \end{cases}$

所以 $T(n)=O(n^3)$**，与原始定义计算相比并不有效。**