第二讲 递推公式求解

汪小林

北京大学计算机系

本讲内容

- 代换归纳法（Substitution method）
  - 猜测、归纳证明、调整、变量代换
- 递归树求和法（Recursion-tree method）
  - 递归树展开、数列求和
- 主定理法（Master method）
  - 主定理、主定理的直观含义、通用定理

代换归纳法

- 代换归纳法求解递归式的三个步骤
  – 猜测解的形式
  – 用数学归纳法证明
    - 找出使解有效的常数
  – 确定常数使边界条件成立
- 常用技巧
  – 猜测更紧的解的形式
  – 通过变量替换转变递归式为熟悉的形式

例1： $T(n) = 4T(n/2) + n$

- [假定$T(1)=\Theta(1)$]
- 猜测$T(n)= O(n^3)$（分别证明O和Ω 关系）
- 假设，对于所有的$k<n$

$$T(k) \le ck^3$$

- 通过归纳法证明

$$T(n) \le cn^3$$

例： $T(n) = 4T(n/2) + n$

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

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

$= (c/2)n^3 + n$

$= cn^3 - ((c/2)n^3 - n)$ ⟵ 期望的形式 － 余项

$\le cn^3$ ⟵ 期望的形式

这里要保证：$((c/2)n^3 - n) \ge 0$,

余项

这只需要：$c \ge 2$并且$n \ge 1$

于是有 $T(n) = O(n^3)$

例：$T(n) = 4T(n/2) + n$

- 还必须处理初始情形，才能使归纳成立。
- 注意到，因为对所有的 $1 \le n < n_0$ 都有 $T(n) = \Theta(1)$（其中n_0是某个适当的常数）
- 于是当 $1 \le n < n_0$时，只要 c 足够大，就有

$$“\Theta(1)” \le cn^3$$

- 但这个界并不够紧

更紧的上界

- 我们来证明 $T(n) = O(n^2)$
- 假设对于所有的 $k<n$，有 $T(k) \le ck^2$

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\le 4c(n/2)^2 + n \\
&= \cancel{O(n^2)} \\
&= cn^2 - (-n) \\
&\le cn^2
\end{aligned}
$$

错！必须证明完全一致的形式

期望的形式 – 余项

但对任何 $c > 0$，上式最后一步不可能成立！

更紧的上界

要点：加强归纳假设

*减去一个低阶项

假设：对于 $k < n$，有 $T(k) \le c_1k^2 - c_2k$

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

$\le 4(c_1(n/2)^2 - c_2(n/2)) + n$

$= c_1n^2 - 2c_2n + n$

$= c_1n^2 - c_2n - (c_2n - n)$

$\le c_1n^2 - c_2n$ 当 $c_2 > 1$

可以取 c_1 足够大来处理初始情况。

例：$T(n) = 4T(n/2) + n = \Omega(n^2)$

- 再证明： $T(n) = \Omega(n^2)$
- 假设对于 $k < n$，有 $T(n) \geq ck^2$

$$T(n) = 4T(n/2) + n$$
$$\geq 4c(n/2)^2 + n$$
$$= cn^2$$

- 取 c 足够小来处理初始情况。
- $T(n) = O(n^2)$且$T(n) = \Omega(n^2)$得$T(n) = \Theta(n^2)$

例： $T(n)=2T(\sqrt{n})+\log n$

- 通过改变变量转化递归式，将$\sqrt{n}$转化为整数。令$m = \log n$，于是

  $T(2^m) = 2T(2^{m/2}) + m$

- 再令$S(m) = T(2^m)$，于是

  $S(m) = 2S(m/2) + m$

  $= \Theta(m\log m)$

  $T(n) = T(2^m) = S(m) = \Theta(m\log m)$

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

习题

4.1-1 证明$T(n)=T(\lceil n/2 \rceil)+1$的解为$O(\log n)$

4.1-2 证明$T(n)=2T(\lfloor n/2 \rfloor)+n$的解为$\Theta(n\log n)$

4.1-3 求$T(n)=2T(\sqrt{n})+1$的渐进紧的解

递归树法

- 用递归树对算法递归执行过程的时间开销进行建模
- 递归树法非常适合于为代入法提供一个好的猜测
- 递归树法可能是不可靠的，因为其中涉及到了很多省略
- 递归树法能让我们有直观的认识

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

$$T(n)$$

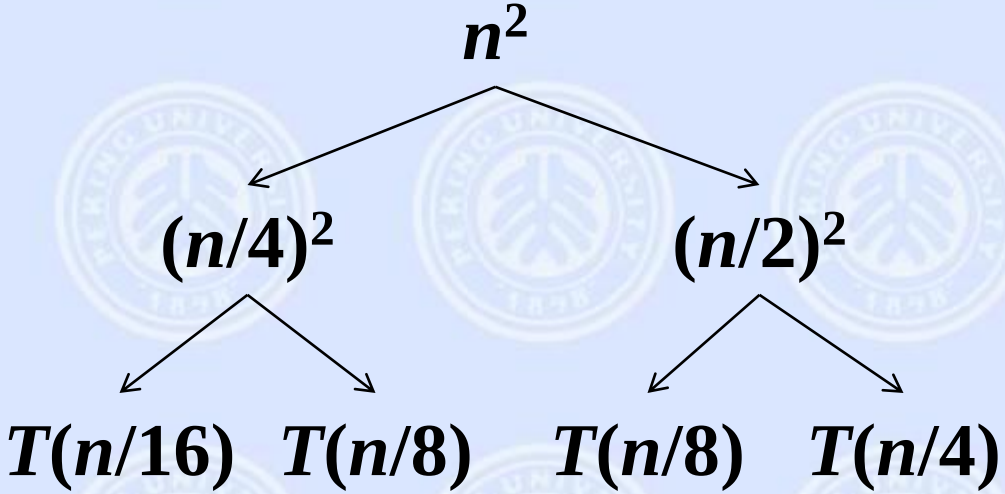
$$T(n) = T(n/4) + T(n/2) + n^2$$

$$T(n/4) + T(n/2) + n^2$$

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

$$\begin{tikzcd}[column sep=small] & n^2 \arrow[dl] \arrow[dr] & \\ T(n/4) & & T(n/2) \end{tikzcd}$$

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



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

$$n^2$$

$$(n/4)^2 \qquad (n/2)^2$$

$$(n/16)^2 \quad (n/8)^2 \quad (n/8)^2 \quad (n/4)^2$$

$$\dots$$

$$\Theta(1)$$

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

n^2 ········ n^2

$(n/4)^2$ $(n/2)^2$ ········ $(5/16)n^2$

$(n/16)^2$ $(n/8)^2$ $(n/8)^2$ $(n/4)^2$ ········ $(25/256)n^2$

… …

$\Theta(1)$

$$T(n) = n^2(1+5/16+(5/16)^2+\ldots)$$
$$= \Theta(n^2)$$

代入法证明 $T(n) = T(n/4) + T(n/2) + n^2 = \Theta(n^2)$

- 先证明 $T(n) = O(n^2)$，
  假设对于 $k < n$，
  都有 $T(k) \le ck^2$

- 取尽量大的 c 可以保证 n 较小时假设成立。

$$
\begin{aligned}
T(n) &= T(n/4) + T(n/2) + n^2 \\
&\le c(n/4)^2 + c(n/2)^2 + n^2 \\
&= \frac{c}{16}n^2 + \frac{c}{4}n^2 + n^2 \\
&= cn^2 - \frac{11c}{16}n^2 + n^2 \\
&= cn^2 - \left(\frac{11c}{16} - 1\right)n^2 \\
&\le cn^2 \quad \text{if} \quad c \ge \frac{16}{11}
\end{aligned}
$$

代入法证明 $T(n) = T(n/4) + T(n/2) + n^2 = \Theta(n^2)$

- 再证明 $T(n) = \Omega(n^2)$，假设对于 $k < n$，都有 $T(k) \geq ck^2$

- 取尽量小的 c 可以保证 n 较小时假设成立。

$$
\begin{aligned}
T(n) &= T(n/4) + T(n/2) + n^2 \\
&\geq c(n/4)^2 + c(n/2)^2 + n^2 \\
&= \frac{c}{16}n^2 + \frac{c}{4}n^2 + n^2 \\
&= cn^2 - \frac{11c}{16}n^2 + n^2 \\
&= cn^2 + \left(1 - \frac{11c}{16}\right)n^2 \\
&\geq cn^2 \quad \text{if} \quad c \leq \frac{16}{11}
\end{aligned}
$$

序列求和

$$\sum_{k=1}^{n} a_k = \frac{n(a_1 + a_n)}{2}$$

等差级数$\{a_k\}$

$$\sum_{k=0}^{n} aq^k = \frac{a(1-q^{n+1})}{1-q}$$

等比级数$\{aq^k\}$

$$\sum_{k=1}^{n} \frac{1}{k} = \ln n + \mathrm{O}(1)$$

调和级数$\{1/k\}$

例题 求和

$$\sum_{k=1}^{n-1}\frac{1}{k(k+1)}$$

$$=\sum_{k=1}^{n-1}\left(\frac{1}{k}-\frac{1}{k+1}\right)$$

$$=\sum_{k=1}^{n-1}\frac{1}{k}-\sum_{k=1}^{n-1}\frac{1}{k+1}$$

$$=\sum_{k=1}^{n-1}\frac{1}{k}-\sum_{k=2}^{n}\frac{1}{k}$$

$$=1-\frac{1}{n}$$

$$\sum_{t=1}^{k}t2^{t-1}=\sum_{t=1}^{k}t\left(2^{t}-2^{t-1}\right)=\sum_{t=1}^{k}t2^{t}-\sum_{t=1}^{k}t2^{t-1}$$

$$=\sum_{t=1}^{k}t2^{t}-\sum_{t=0}^{k-1}(t+1)2^{t}$$

$$=\sum_{t=1}^{k}t2^{t}-\sum_{t=0}^{k-1}t2^{t}-\sum_{t=0}^{k-1}2^{t}$$

$$=k2^{k}-\left(2^{k}-1\right)$$

$$=(k-1)2^{k}+1$$

例题 调和级数求和的界

$$\sum_{k=1}^{n}\frac{1}{k}\geq\int_{1}^{n+1}\frac{\mathrm{d}x}{x}=\ln\left(n+1\right)$$

$$\sum_{k=1}^{n}\frac{1}{k}=1+\sum_{k=2}^{n}\frac{1}{k}\leq 1+\int_{1}^{n}\frac{\mathrm{d}x}{x}=\ln n+1$$

$$\sum_{k=1}^{n}\frac{1}{k}=\Theta\left(\log n\right)$$

习题

4.2-1 猜测并证明 $T(n)=3T(\lfloor n/2 \rfloor)+n$ 的渐进上界

4.2-4 找 $T(n)=T(n-a)+T(a)+cn$ 的渐进紧确界

其中 $a \geq 1$ 且 $c > 0$ 事常数

主定理法

- 主定理法适用于形如下式的递归式

$$T(n) = aT(\frac{n}{b}) + f(n)$$

- 其中 $a \geq 1, b>1$，并且 f 是渐进正的函数。

主定理有三种情形

1. 如果存在正数 $\varepsilon > 0$，使得

$$f(n) = O(n^{\log_b a - \varepsilon})\ \text{则}\ T(n) = \Theta(n^{\log_b a})$$

2. 如果

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

3. 如果对于某个常数 $\varepsilon > 0$，使得

$f(n) = \Omega(n^{\log_b a + \varepsilon})$，并且对某个常数 $c <$

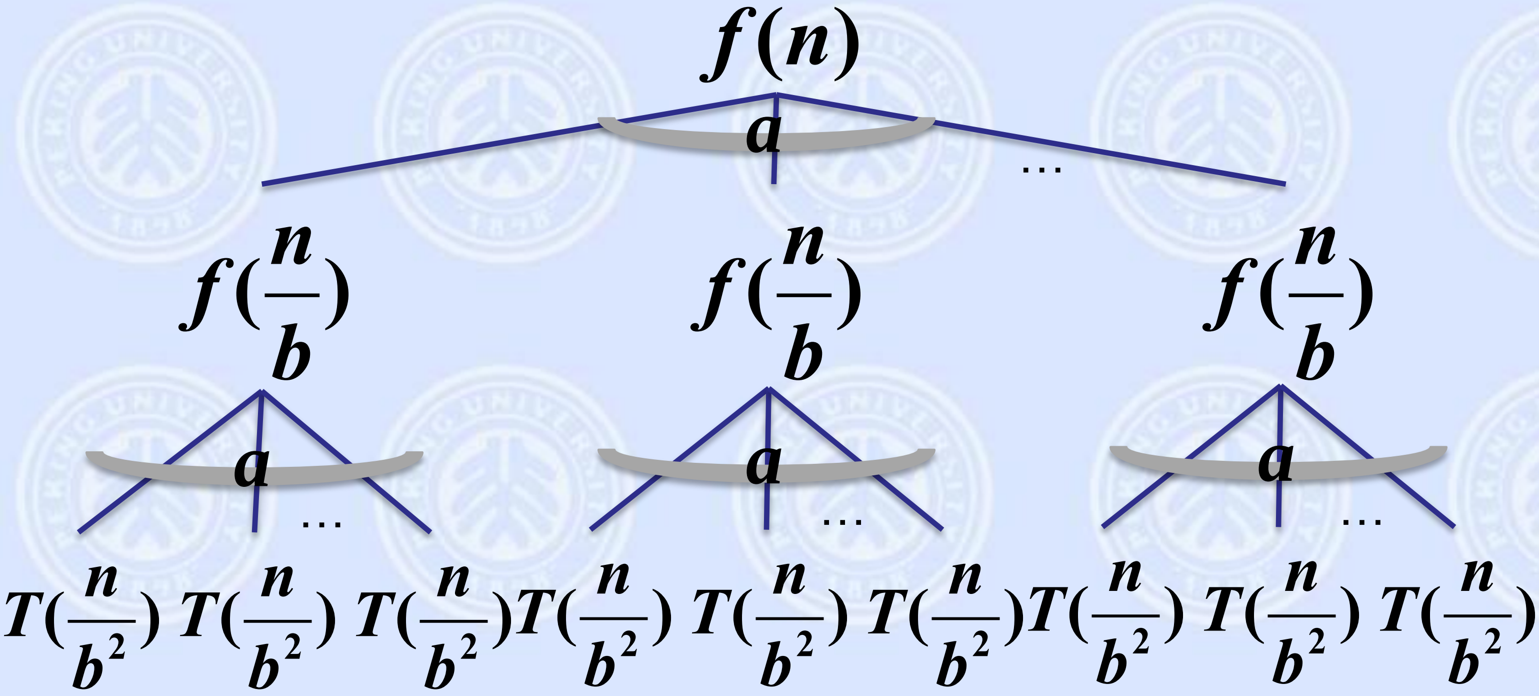
和所有足够大的 n 都有

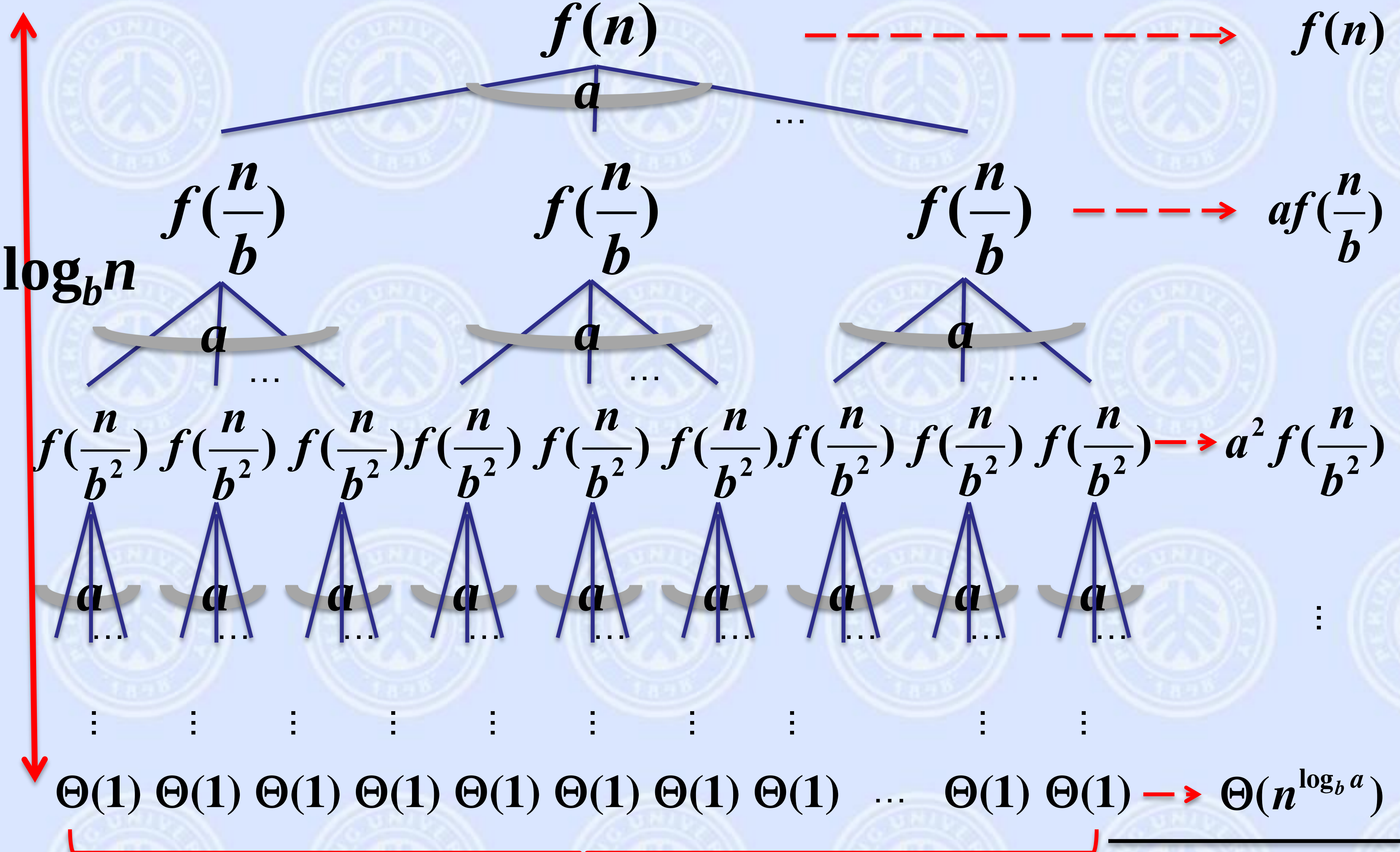
$$af(\frac{n}{b}) \le cf(n)\ \text{则}\ T(n) = \Theta(f(n))$$

$$f(n)$$

a …

$$T\left(\frac{n}{b}\right) \qquad T\left(\frac{n}{b}\right) \qquad T\left(\frac{n}{b}\right)$$


$f(n)$
a
…
$f(\frac{n}{b})$ $f(\frac{n}{b})$ $f(\frac{n}{b})$
a a a
… … …
$T(\frac{n}{b^2})$ $T(\frac{n}{b^2})$ $T(\frac{n}{b^2})$ $T(\frac{n}{b^2})$ $T(\frac{n}{b^2})$ $T(\frac{n}{b^2})$ $T(\frac{n}{b^2})$ $T(\frac{n}{b^2})$ $T(\frac{n}{b^2})$



$n^{\log_b a}$

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

主定理法 例题 1

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

$$a = 4, \quad b = 2 \quad \Rightarrow \quad n^{\log_b a} = n^2$$

$$f(n) = n$$

Case 1: $f(n) = O\left(n^{2-\varepsilon}\right)$ for $\varepsilon = 1$

$$\therefore T(n) = \Theta\left(n^2\right)$$

主定理法 例题 2

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

$$a = 4, \quad b = 2 \quad \Rightarrow \quad n^{\log_b a} = n^2$$

$$f(n) = n^2$$

$$\text{Case 2: } f(n) = \Theta\left(n^2\right)$$

$$\therefore T(n) = \Theta\left(n^2 \log n\right)$$

主定理法 例题 3

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

$$a = 4, \quad b = 2 \quad \Rightarrow \quad n^{\log_b a} = n^2$$

$$f(n) = n^3$$

Case 3: $f(n) = \Omega\left(n^{2+\varepsilon}\right)$ **for** $\varepsilon = 1$ **and**

$4f(n/2) = 4(n/2)^3 \le cn^3$ **for** $c = 1/2.$

$$\therefore T(n) = \Theta\left(n^3\right)$$

主定理法 例题 4

$$T(n) = 4T(n/2) + n^2/\log n$$

$$a = 4, \quad b = 2 \quad \Rightarrow \quad n^{\log_b a} = n^2$$

$$f(n) = n^2/\log n$$

主方法不适用

对于任意 $\varepsilon > 0$， $n^{\varepsilon} = \omega(\log n)$

主定理法 练习

- 题目 4-1: 递归式例题

a) $T(n) = 2T(n/2) + n^3$ $T(n) = \Theta(n^3)$ Case 3

b) $T(n) = T(9n/10) + n$ $T(n) = \Theta(n)$ Case 3

c) $T(n) = 16T(n/4) + n^2$ $T(n) = \Theta(n^2 \log n)$ Case 2

- 题目 4-4: 递归式例题

e) $T(n) = 2T(n/2) + n/\log n$ $T(n) = \Theta(n\log(\log n))$

$$T(n) = 2T(n/2) + n/\log n$$

$$T(n)$$

$$T(n) = 2T(n/2) + n/\log n$$

$$2T(n/2) + n/\log n$$

$$T(n) = 2T(n/2) + n/\log n$$

$$\begin{tikzcd} & n/\log n \arrow[dl] \arrow[dr] & \\ T(n/2) & & T(n/2) \end{tikzcd}$$

$T(n) = 2T(n/2) + n/\log n$

$$
\begin{tikzcd}[column sep=small]
 & & & n/\log n \arrow[dll] \arrow[drr] & & & \\
 & n/(2\log(n/2)) \arrow[dl] \arrow[dr] & & & & n/(2\log(n/2)) \arrow[dl] \arrow[dr] & \\
T(n/4) & & T(n/4) & & \ldots & & \ldots
\end{tikzcd}
$$

$$T(n) = 2T(n/2) + n/\log n$$

$n/\log n$ $\quad\quad$ $n/\log n$

$n/(2\log(n/2))$ $\quad$ $n/(2\log(n/2))$ $\quad\quad$ $n/(\log n-1)$

$n/(4\log(n/4))$ … $\quad$ … $\quad$ … $\quad\quad$ $n/(\log n-2)$

…

…

$\Theta(1)$

$$T(n) = \Theta(n(1+1/2+1/3+\ldots+1/\log n))$$

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

$$T(n) = 2T(n/2) + n/\log n = O(n\log(\log n))$$

- 代换法，先证明$T(n) = O(n\log(\log n))$
- 假设对于$k < n$，都有$T(k) \le ck\log(\log k)$

$$T(n) = 2T(n/2) + n/\log n$$

$$\le 2c(n/2)\log\log(n/2) + n/\log n$$

$$= cn\log\log n - cn\log\log n + cn\log\log(n/2) + n/\log n$$

$$= cn\log\log n - (n/\log n)\left(c\log n\left(\log\log n - \log\log(n/2)\right) - 1\right)$$

$$\le cn\log\log n \quad \text{if} \quad c\log n\left(\log\log n - \log\log(n/2)\right) \ge 1$$

$T(n) = 2T(n/2) + n/\log n = O(n\log(\log n))$

$$\log n\left(\log\log n - \log\log\left(n/2\right)\right)$$

$$= \log n\left(\log\log n - \log\left(\log n - \log 2\right)\right)$$

$$= \log n\left(\log\log n - \log\left(\log n - 1\right)\right)$$

$$= \log n\left(\log\left(\log n \big/ (\log n - 1)\right)\right)$$

$$= \log\left(\log n \big/ (\log n - 1)\right)^{\log n}$$

$$= \log\left(1 + 1\big/(\log n - 1)\right)^{\log n - 1 + 1}$$

$$T(n) = 2T(n/2) + n/\log n = O(n\log(\log n))$$

$$= \log\left(1 + 1/(\log n - 1)\right)^{\log n - 1 + 1}$$

$$= \log\left(\left(1 + 1/(\log n - 1)\right)\left(1 + 1/(\log n - 1)\right)^{\log n - 1}\right)$$

$$\geq \log(1 \cdot 2) = 1 \quad \text{if } n \geq 4$$

$$\therefore c\log n\left(\log\log n - \log\log(n/2)\right) \geq 1 \quad \text{if} \quad c \geq 1$$

取尽量大的 c 可以保证 $n < 4$ 时

$T(n) = \Theta(1) \leq cn\log\log n$

$$T(n) = 2T(n/2) + n/\log n = \Omega(n\log(\log n))$$

- 代换法，再证明$T(n) = \Omega(n\log(\log n))$
- 假设对于$k < n$，都有$T(k) \geq ck\log(\log k)$

$$T(n) = 2T(n/2) + n/\log n$$

$$\geq 2c(n/2)\log\log(n/2) + n/\log n$$

$$= cn\log\log n - cn\log\log n + cn\log\log(n/2) + n/\log n$$

$$= cn\log\log n + (n/\log n)\big(1 - c\log n(\log\log n - \log\log(n/2))\big)$$

$$\geq cn\log\log n \quad \text{if} \quad c\log n(\log\log n - \log\log(n/2)) \leq 1$$

$$T(n) = 2T(n/2) + n/\log n = \Omega(n\log(\log n))$$

$$\log n\left(\log\log n - \log\log\left(n/2\right)\right)$$

$$= \log\left(\left(1+1/(\log n - 1)\right)\left(1+1/(\log n - 1)\right)^{\log n - 1}\right)$$

$$\leq \log\left(2\cdot e\right) \quad \text{if } n \geq 4$$

$$\therefore c\log n\left(\log\log n - \log\log\left(n/2\right)\right) \leq 1 \quad \text{if} \quad c \leq 1/\log\left(2e\right)$$

取尽量小的 c 可以保证 $n < 4$ 时

$$T(n) = \Theta(1) \geq cn\log\log n$$

通用方法 (Akra-Bazzi)

$$T(n) = \sum_{i=0}^{k} a_i T\left(\frac{n}{b_i}\right) + f(n)$$

如果

$$\sum_{i=0}^{k}\left(\frac{a_i}{(b_i)^p}\right) = 1$$

其中 p 为上式的唯一解。

则我们有和主方法类似的结论

只是用 n^p 代替了 $n^{\log_b a}$

通用方法 例题

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

解:

$a_1=a_2=1,\ b_1=3,\ b_2=3/2$

令 $p = 1$

则 $a_1/b_1^p + a_2/b_2^p = 1/3 + 2/3 = 1$

$f(n) = cn = \Theta(n^p)$

于是 $T(n) = \Theta(n\log n)$ （情形 2）

思考题4-2: 寻找缺失的整数

- 某数组 $A[1 \cdots n]$ 包含有从 0 到 n 的整数，但其中一个整数不在数组中。通过一个辅助数组 $B[0 \cdots n]$ 来记录 A 中出现的整数，很容易在$O(n)$时间内找出所缺的整数。但在这个问题中，我们却不能由一个单一操作来访问 A 中的一个完整的整数，因为 A 中的元素是以二进制表示的。我们所能用的唯一操作就是“取 $A[i]$ 的第 j 位”，这个操作所花时间为常数。
- 证明：如果访问数组 A 中信息的唯一方式是这种单一位操作，仍能在 $O(n)$ 时间内找出所缺失的整数。

问题4-2: 寻找缺失的整数

- 解题思路
  - 进行类似二分搜索的查找
  - 我们知道在 **0** 到 n 这 n+1 个数中
  - 大于等于$2^{[\log n-1]}$数，其第**[log*n*-1]**位为**1**，否则为 **0**
  - 通过查数组*A*中有多少个**[log*n*-1]**位为 **1** 的数，可以判断*A*中缺的数大于等于 $2^{[\log n-1]}$，还是小于之
  - 继续判断包含缺失数的那部分数的下一位，可以再次将包含缺失数的集合大小减半
  - （也可以从低位到高位一词判断）

思考题4-6 VLSI芯片测试与练习

- Diogenes教授有 n 个被认为是完全相同的VLSI芯片，原则上它们是可以互相测试的。教授的测试装置一次可测两片，当该装置中放有两片芯片时，每一片就对另一片做测试并报告其好坏。一个好的芯片总能够报告另一片的好坏，但一个坏的芯片的结果是不可靠的。这样，每次测试的四种可能结果如下：
  - *B* 是好的 *A*是好的　　两片都是好的，或两片都是坏的
  - *B* 是好的 *A* 是坏的　　至少一片是坏的
  - *B* 是坏的 *A* 是好的　　至少一片是坏的
  - *B* 是坏的 *A* 是坏的　　至少一片是坏的
- 如果多于 $n/2$ 的芯片是坏的，就无法确定那个芯片是好的。
- 如果多于 $n/2$ 的芯片是好的，则可用 $\Theta(n)$ 对芯片测试找出一个好芯片。给出并解答表达测试次数的递归式。

问题4-6: VLSI芯片测试

算法思路1（基于队列的算法 ）：

1. 任选两个芯片，如果两者同好或同坏，留下二者，转步骤2；如果两者至少有一个坏的，两者都丢掉，转步骤1。
2. 如果没有留下的芯片，转步骤1；否则取留下的一个芯片和一个新芯片比较，如果两者同好或同坏，新芯片也留下；如果两者至少有一个坏的，两者都丢掉，转步骤2。
3. 当没有新的芯片时，剩下的芯片就都是好的。

说明：

一、在这个过程中，丢掉的芯片中，坏的芯片一定比好的芯片多

二、留下的芯片要么都是好的，要么都是坏的；

三、好的芯片比坏的多，所以留下的不可能是坏芯片。

循环不变式：

"并且队列中的芯片同为好或同为坏；所有留下的芯片中好的多"

问题4-6: VLSI芯片测试

算法思路2（递归的算法）：

1.芯片两两测试，如果结果是至少有一个是坏的，则两个芯片都丢掉；否则，丢掉一个留下另一个。最后留下的芯片中，好的芯片一定不比坏的芯片少。（恰好进行了 ***n*/2** 次测试）

2.如果存在一个未参与过测试的芯片，则用它与留下的芯片依次比较。如果一半或一半以上的测试结果说这个芯片是好的，则这个芯片一定是好芯片（为什么？反证法），好芯片找到，算法结束；否则这个芯片是坏芯片，丢掉它。（最多进行了 ***n*/2** 次测试）

3.对留下的芯片中，自然好的芯片占多数。重复步骤 1 至 2，直到只剩下一个芯片，这个芯片是好芯片，算法结束。

$$T(n) \le T(\lfloor n/2 \rfloor) + n$$

递归式求解小结

- 代换法
  - 利用数学归纳法分别证明O和Ω
  - 完全一致的归纳假设形式
  - 取合适的值保证初始条件成立
- 递归树法
  - 用来猜测合理的递归式解的形式{用代换法严格证明}
  - 级数求和
- 主方法
  - 可以根据情形简单套用方法即可快速求解递归式
- 良好的数学知识是求解递归式的基础