数据结构与算法设计

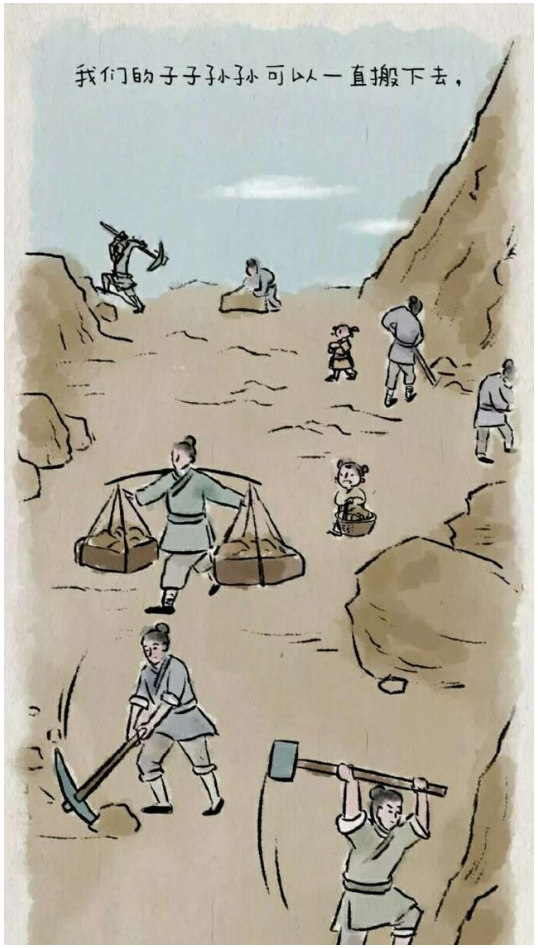
周　可

Mail : zhke@hust.edu.cn

华中科技大学，武汉光电国家研究中心

Preface

愚公移山背后蕴含的递归思想



子子孙孙无穷匮也，而山不加增，何苦而不平?

All recursive algorithms must obey three important laws: A recursive algorithm must have **a base case**. A recursive algorithm **must change its state and move toward the base case**. A recursive algorithm must **call itself**, recursively.

递归式求解

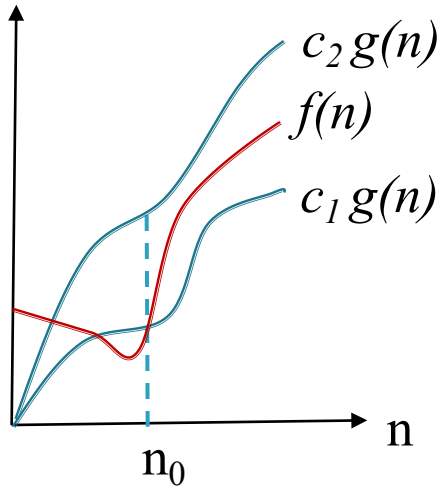
1 代换法

2 递归树法

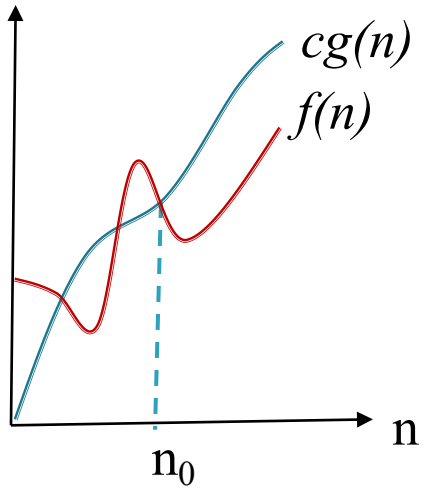
3 主方法

回顾：

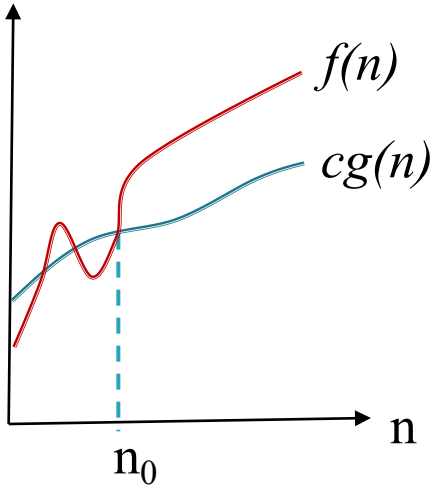
- 1. 分治法、递归求解思想
  - Divide → Conquer → Combine
- 2. 算法的复杂度



$f(n)=\Theta(g(n))$



$f(n)=O(g(n))$



$f(n)=\Omega(g(n))$

recurrences的求解

这里，所谓求解递归式就是化简递归式，以得到形式简单的函数表示。由于很多算法具有递归（循环）特性，对它们分析所得的时间（或空间）复杂度表达式多为递归式，所以递归式的求解就显得重要了。

递归式求解的结果是得到形式简单的渐进限界函数表示（这里要求用O、Ω、Θ的表示）。三种常用方法：

- 代换法
- 递归树法
- 主方法

递归式求解

对表达式细节的简化

为便于处理，通常做如下假设和简化处理

(1) 运行时间函数T(n)的定义中，一般假定自变量为正整数。

(2) 忽略递归式的边界条件，即n较小时函数值的表示。

原因在于，虽然递归式的解会随着T(1)值的改变而改变，但此改变不会超过常数因子，对函数的阶没有根本影响。

(3) 对上取整、下取整运算做合理简化，如

$$T(n)=T(\lceil n/2\rceil)+T(\lfloor n/2\rfloor)+f(n)$$

通常忽略上、下取整函数，直接写作：

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

注：被简化的细节并不是不重要，只是这些细节处理不影响算法分析的渐近界，是“无穷”分析中的合理假设和简化。

在细节被简化处理的同时，也要知道它们在什么情况下是“实际”重要的。这样就可以了解算法在各种情况的具体执行情况。

- 用代换法解递归式基本思想：

先猜测一个界，然后用数学归纳法证明该猜测的正确性。

此时，用该猜测作为归纳假设，在推论证明时作为较小值代替函数的解，然后证明推论的正确性。

- 用代换法解递归式的步骤：

(1) 猜测渐近界的形式（**guess**）

(2) 用数学归纳法证明猜测的正确性（**verify**）

例：用代换法确定下式的上界

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

分析：该式与 $T(n) = 2T(n/2) + n$ 类似，故猜测其解为*O(nlogn)*。

现在设法证明*T(n)≤cnlogn*，并确定常数c的存在。

假设该界对 $\lfloor n/2 \rfloor$ 成立，即 $T(\lfloor n/2 \rfloor) \le c\lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor)$，然后在数学归纳法推论证明阶段对递归式做替换，有：

$$\begin{aligned} T(n) &\le 2(c\lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor)) + n \\ &\le cn\log(n/2) + n \\ &= cn\log n - cn\log 2 + n \\ &= cn\log n - (c-1)n \end{aligned}$$

故，要使*T(n)≤cnlogn成立，只要c≥1就可以，这样的c是存在的、合理的。*

上面的证明过程，证明了当n足够大时猜测的正确性。但边界值呢？

即： $T(n) \le cn\log n$的结论对于小n成立吗？

分析：

事实上，对n＝1，上述结论存在问题：

作为边界条件，我们有理由假设T(1)＝1，但对n＝1，T(1)≤c·1·log1＝0，与T(1)＝1不相符。

也即， $T(n) \le cn\log n$对于归纳证明的基本情况不成立。怎么处理？

从n_0的定义出发：

只需要存在常数n_0，使得$n \geq n_0$时结论成立即可，所以n_0不一定取1。

所以，这里，我们不取$n_0=1$，而取$n_0=2$，并将T(2)、T(3)作为归纳证明中的边界条件代替T(1)（但依旧假设T(1)=1）使得T(2)、T(3) 满足$T(n) \leq cn\log n$。

而$n>3$时，递归计算不再直接依赖T(1)，使用T(2)、T(3)即可完成。

带入T(1)=1，通过递归式有，T(2) = 4，T(3)=5，

如何使T(2)、T(3)满足T(n)≤cnlogn？

只要c取足够大的常数，就有 T(2)≤c2log2 和 T(3)≤c3log3 成立即可。这样的c是什么？

答案：只要c≥2即可。

综上所述，取常数c≥2，最终的结论T(n)≤cnlogn就成立。命题得证。

如何猜测递归式的渐近界呢？

1）主要靠经验

- 尝试1：看有没有形式上类似的表达式，以此推测新递归式的渐近界。
- 尝试2：先猜测一个较宽的界，然后再缩小不确定区间，收缩到精确的渐近界。

2）避免盲目推测

如： $T(n) \le 2(c\lfloor n/2 \rfloor) + n \le cn + n = \mathrm{O}(n)$

原因：没有证明一般形式T(n)≤cn。（cn+n⊀cn）

必要的时候要做一些技术处理

1）去掉一个低阶项（略,算法导论P39）

2）变量代换

对陌生的递归式做些简单的代数变换，使之变成较熟悉的形式。

例：化简 $T(n) \le 2T(\lfloor\sqrt{n}\rfloor) + \log n$

分析：原始形态比较复杂

做代数代换：令 $m = \log n$，则n=2^m，$\sqrt{n} = 2^{m/2}$

同时，为简单起见，忽略下取整细节 $\lfloor\sqrt{n}\rfloor$，直接使用 $\sqrt{n}$ ，得：

$$T(2^m) \le 2T(2^{m/2}) + m$$

$$T(2^m) \le 2T(2^{m/2}) + m$$

再设S(m) = T(2^m)，得以下形式递归式：

$$S(m) \le 2S(m/2) + m$$

从而获得形式上熟悉的递归式.

而新的递归式的上界是：$O(m \log m)$

再将S(m)带回T(n)，有，

$$\begin{aligned} T(n) &= T(2^m) \\ &= S(m) = O(m \log m) \\ &= O(\log n \log\log n) \end{aligned}$$

这里， $m = \log n$

递归式求解

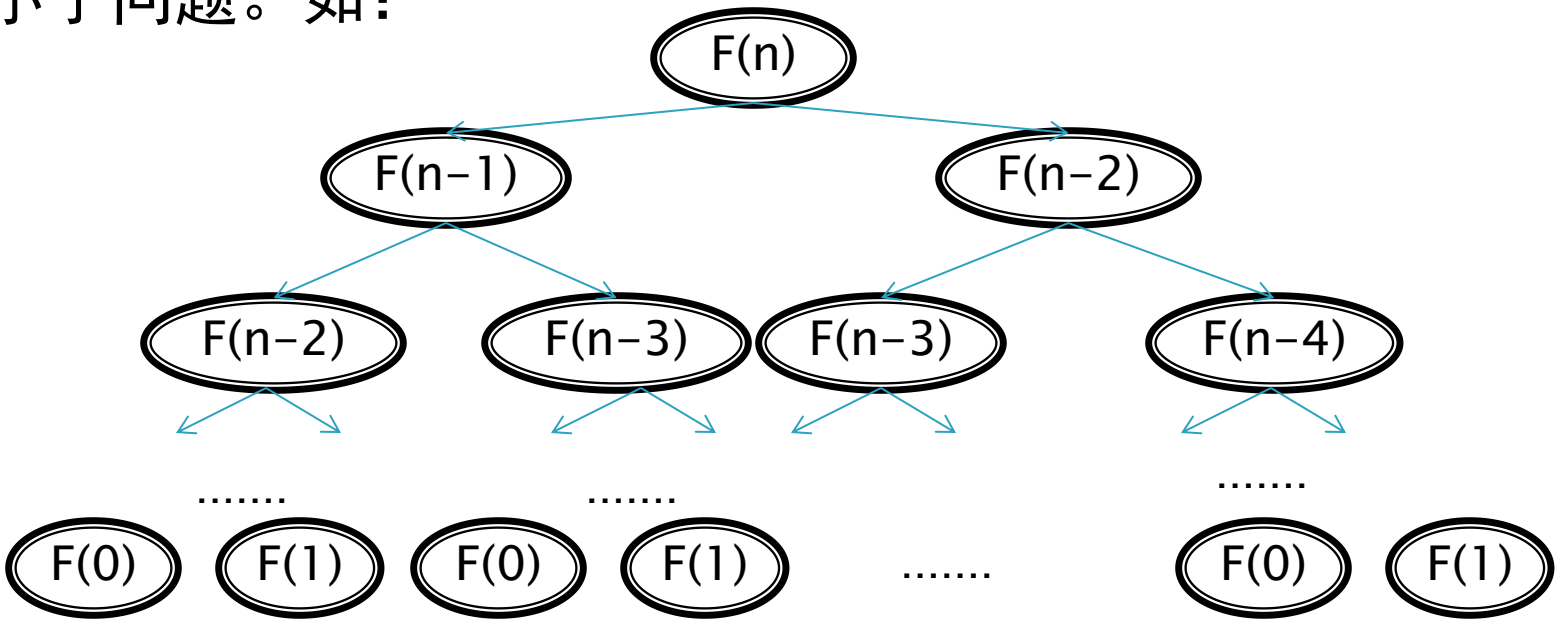
1 代换法

2 递归树法

3 主方法

递归树：用来描述递归调用过程的树。

根节点代表原始问题，根节点的子节点代表第一次分割划分出来的子问题，依次类推。叶子节点代表可以直接求解的最小子问题。如：



好处：很直观、清晰地描述出递归的执行过程，而且可以用我们已经了解到的树方面的性质做一些深入的分析。

基于递归树的时间分析

节点代价：在递归树中，每个节点有求解相应（子）问题的代价(cost)。

层代价：每一层各节点代价之和。

总代价：整棵树的各层代价之和

目　　标：利用树的性质，获取对递归式解的猜测，然后用代换法或其它方法加以验证。

例：已知递归式$T(n)=3T(\lfloor n/4 \rfloor)+\Theta(n^2)$，求其上界

准备性工作：为简单起见，对一些细节做必要、合理的简化和假设，这里为：

（1）去掉底函数的表示

理由：底函数和顶函数对递归式求解并不“重要”。

（2）假设n是4的幂，即$n=4^k$，$k=\log_4 n$。

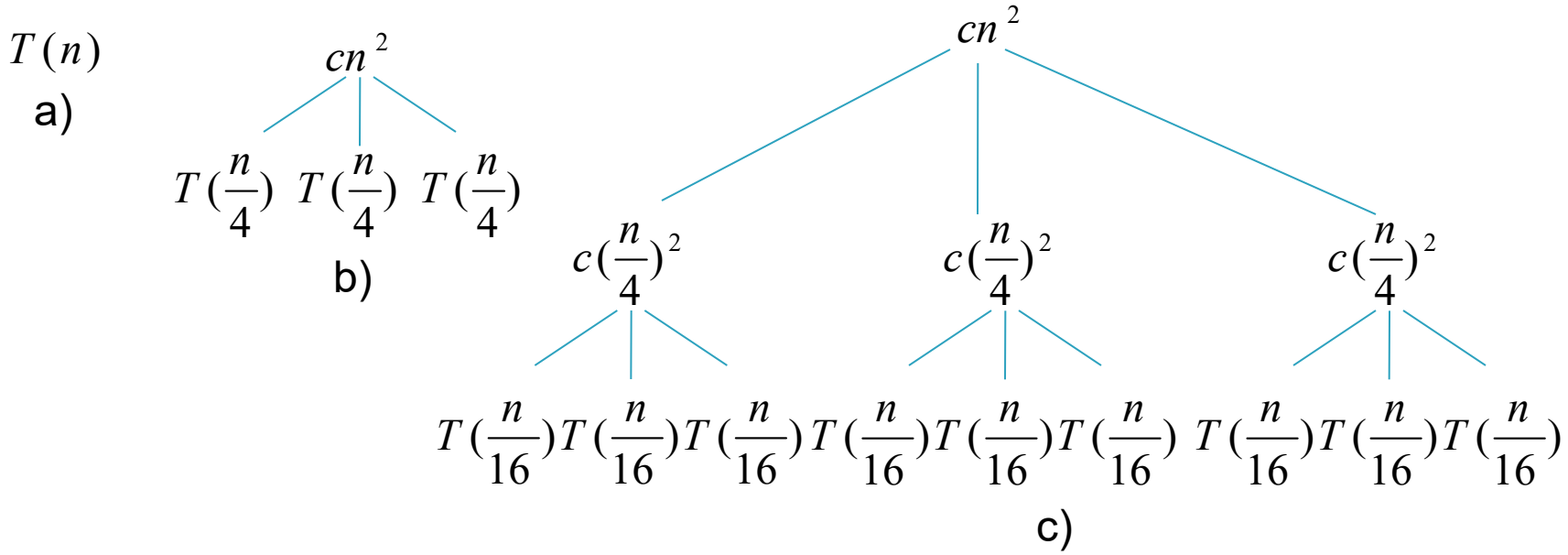
一般，当证明$n=4^k$成立后，再加以适当推广，就可以把结论推广到n不是4的幂的一般情况了。

（3）对 $\Theta(n^2)$，假设其常系数为c，$c>0$，从而可以去掉Θ符号，转变成cn^2的形式。

最终得以下形式的递归式：

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

用递归树描述T(n)的演化过程:

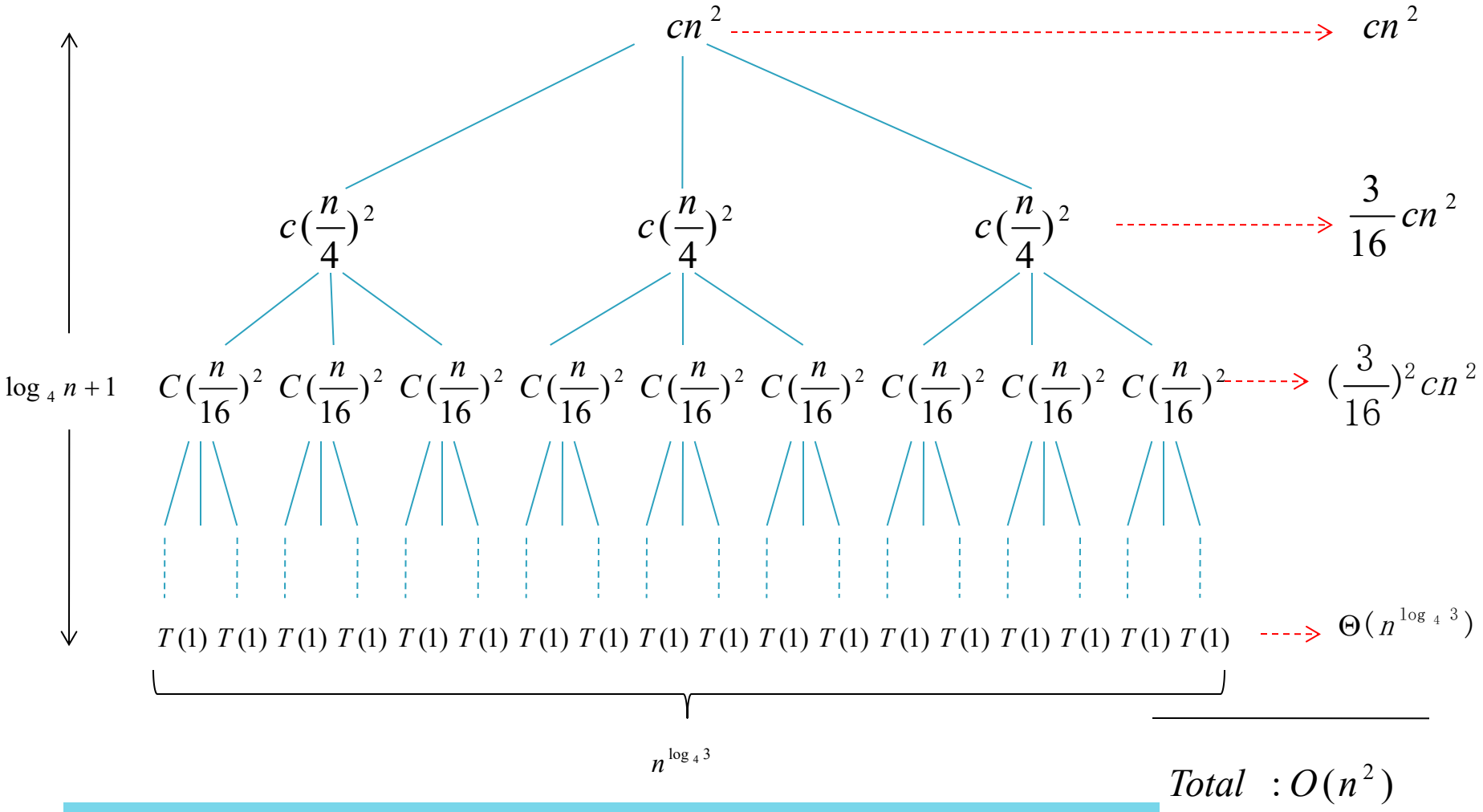


a) 原始问题的T(n)描述。

b) 第一层递归调用的分解情况，cn^2是顶层计算除递归以外的代价，T(n/4)是分解出来的规模为n/4的子问题的代价，总代价$T(n)=3T(n/4)+cn^2$。

c) 第二层递归调用的分解情况。$c(n/4)^2$是三棵子树除递归以外的代价。

继续扩展下去，直到递归最底层，得到如下形式的递归树：



d) 完全扩展的递归树，递归树深度为$\log_4 n$(共有$\log_4 n+1$层)

树的深度：子问题的规模按1/4的方式减小，在递归树中，深度为i的节点，子问题的大小为$n/4^i$。

当$n/4^i=1$时，子问题规模仅为1，达到边界值。

所以，

- 节点高度：$0\sim\log_4 n$
- 树共有$\log_4 n+1$层
- 从第2层起，每一层上的节点数为上层节点数的3倍
- 深度为i的层节点数为3^i。

代价计算

（1）内部节点：位于0~$\log_4 n-1$层

深度为i的节点的代价为 $c(n/4^i)^2$ ，

i层节点的总代价为：$3^i c(n/4^i)^2=(3/16)^i cn^2$。

（2）叶子节点：位于$\log_4 n$ 层，共有 $3^{\log_4 n}=n^{\log_4 3}$

个节点，每个节点的代价为T(1)，

总代价为 $n^{\log_4 3}T(1)=\Theta(n^{\log_4 3})$

（3）树的总代价

整棵树的总代价为各层代价之和，则有

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

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

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

利用等比数列化简上式。

对于实数x≠1，和式 $\sum_{k=0}^{n} x^k = 1 + x + x^2 + ... + x^n$ 是一个几何级数（等比数列），其值为 $\sum_{k=0}^{n} x^k = \frac{x^{n+1} - 1}{x - 1}$

当和是无穷的且$|x| < 1$时，得到无穷递减几何级数，此时

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1 - x}$$

T(n)中，cn^2项的系数构成一个递减的等比级数。将T(n)扩展到无穷，即有

$$
\begin{aligned}
T(n) &= \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{1}{1-(3/16)} cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) \\
&= O(n^2)
\end{aligned}
$$

至此，获得T(n)解的一个猜测：$T(n) \le dn^2$，成立吗？

猜测的证明（代换法）

将$T(n)\le dn^2$作为归纳假设，d是待确定的常数，带入归纳的推论计算过程，有

$$\begin{aligned}T(n) &= 3T(\lfloor n/4 \rfloor) + \Theta(n^2)\\ &\le 3T(\lfloor n/4 \rfloor) + cn^2 \le 3d\lfloor n/4 \rfloor^2 + cn^2\\ &\le 3d(n/4)^2 + cn^2\\ &= \frac{3}{16}dn^2 + cn^2\end{aligned}$$

所以，要使得$T(n)\le dn^2$成立，只要$c\le(13/16)d$即可，故，$T(n)\leqslant dn^2$的猜测成立。

定理得证（边界条件的讨论略）。

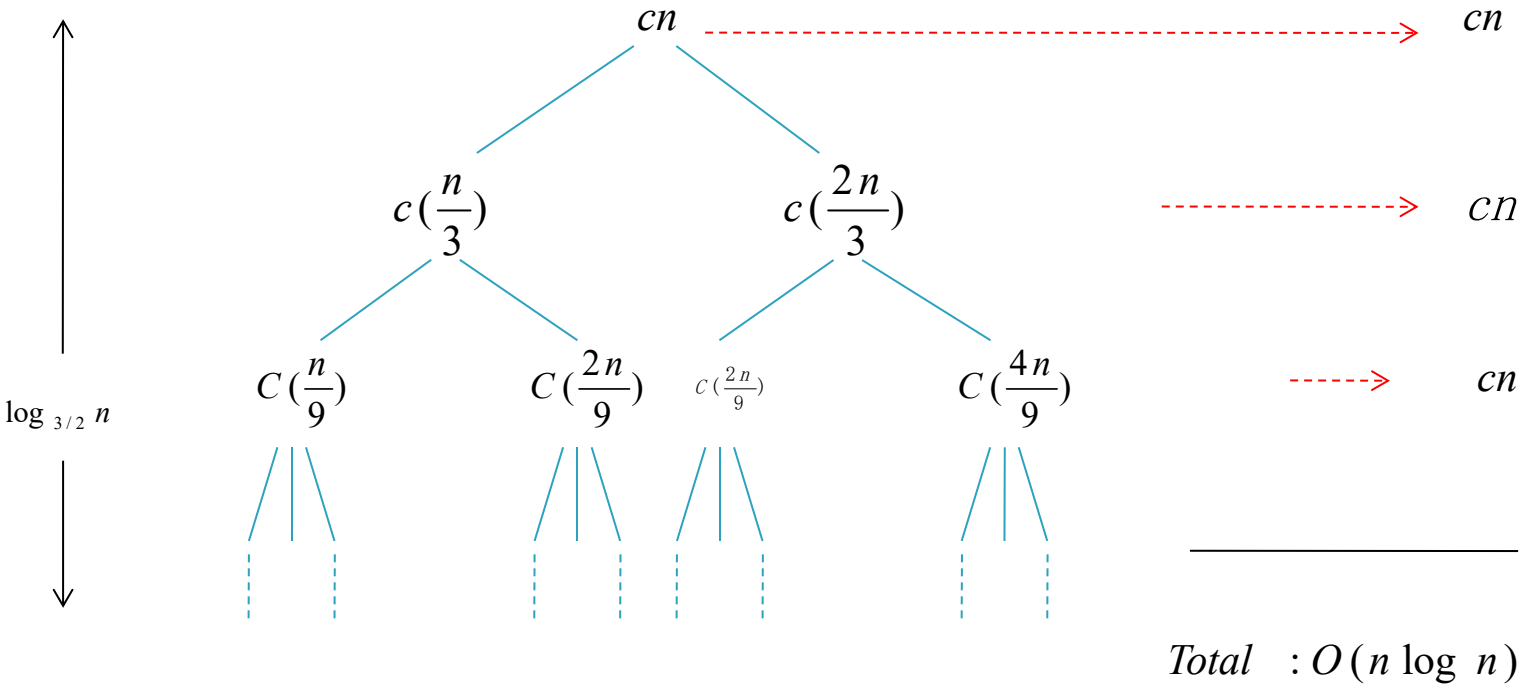
例 求表达式的$T(n)=T(n/3)+T(2n/3)+O(n)$上界

（表达式中直接省略了下取整和上取整函数）

进一步地，取常数c，展开O(n)，得：

$$T(n) \le T(n/3) + T(2n/3) + cn$$

其递归树为：



递归式 $T(n)=T(n/3)+T(2n/3)+O(n)$ 的递归树

注：该树并不是一个完全的二叉树。从根往下，越来越多的内节点消失(1/3分叉上)，因此每层的代价并不都是cn，而是≤cn的某个值。

树的深度：在上述形态中，最长的路径是最右侧路径，由 $n \to (2/3)n \to (2/3)^2 n \to \ldots \to 1$ 组成。当 $k=\log_{3/2} n$ 时，$(3/2)^k/n=1$，所以树的深度为

$$\log_{3/2} n$$

递归式解的猜测：至此，我们可以合理地猜测该树的总代价至多是层数乘以每层的代价，并鉴于上面关于层代价的讨论，我们可以假设递归式的上界为

$$O(cn\log_{3/2} n)=O(n\log n)$$

注：这里，我们假设每层的代价为cn，事实上，cn为每层代价的上界，这一假设是合理的细节简化处理。

猜测的证明

证明T(n)≤dnlog$n$，d是待确定的合适正常数。

$$
\begin{aligned}
T(n) &\le T(n/3) + T(2n/3) + cn \\
&\le d(n/3)\log(n/3) + d(2n/3)\log(2n/3) + cn \\
&= (d(n/3)\log n - d(n/3)\log 3) + (d(2n/3)\log n - d(2n/3)\log 3/2)) + cn \\
&= dn\log n - d((n/3)\log 3 + (2n/3)\log(3/2)) + cn \\
&= dn\log n - d((n/3)\log 3 + (2n/3)\log 3 - (2n/3)\log 2) + cn \\
&= dn\log n - dn(\log 3 - 2/3) + cn \le dn\log n
\end{aligned}
$$

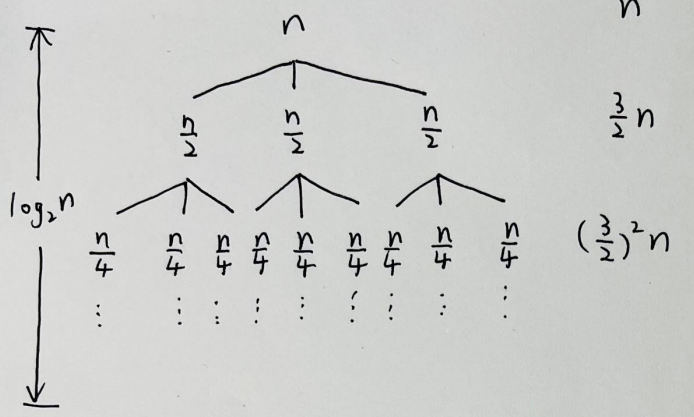
成立吗？

上式的成立条件：$d \ge c/(\log 3 - (2/3))$，存在！

∴猜测正确，递归式解得证。

4.4-1 对递归式 $T(n)=3T(\lfloor n/2\rfloor)+n$，利用递归树确定一个好的渐近上界，用代入法进行验证。

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



$$T(n)=n+\frac{3}{2}n+(\frac{3}{2})^2n+\cdots(\frac{3}{2})^{\log_2 n}n$$

$$=\sum_{i=0}^{\log_2 n}(\frac{3}{2})^i n$$

$$=\frac{1-(\frac{3}{2})^{\log_2 n}}{1-\frac{3}{2}}\cdot n$$

$$=2n\left[(\frac{3}{2})^{\log_2 n}-1\right]$$

$$=2n\cdot\frac{n^{\lg 3}}{n}-2n$$

$$=2n^{\lg 3}-2n$$

故可猜测 $T(n)\le cn^{\lg 3}-2n$

利用数学归纳法有

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

$$\le 3\left[c(\frac{n}{2})^{\lg 3}-n\right]+n$$

$$=3c\cdot\frac{n^{\lg 3}}{3}-2n$$

$$=cn^{\lg 3}-2n$$

递归式求解

1 代换法

2 递归树法

3 主方法

如果递归式如下形式，在满足一定的条件下，可以用主方法直接给出渐近界：

$$T(n) = aT(n/b) + f(n)$$

其中，a、b是常数，且a≥1，b>1；f(n)是一个渐近正的函数。

含义：规模为n的原问题被分为a个子问题，每个子问题的规模是n/b，a和b是正常数。子问题被递归地求解，T(n)是原始问题的时间，每个子问题的时间为T(n/b)；划分出来的子问题的答案合并及其它有关运算的代价由函数f(n)描述。

上式给出了算法总代价与子问题代价的关系。

$$T(n) = aT(n/b) + f(n)$$

注：这里采用了细节的简化，没有考虑n/b的取整问题，省略了下取整、上取整，但本质上不影响对递归式渐近行为的分析。

对上述形式的递归式渐近界的求解是依赖称之为“主定理”的结论给出的。

定理5.1 主定理

设a≥1和b>1为常数，设f(n)为一函数，T(n)由以下递归式给出 $T(n) = aT(n/b) + f(n)$

对于非负整数定义，其中n/b指$\lfloor n/b \rfloor$ 或$\lceil n/b \rceil$。则T(n)可能有如下的渐近界：

1）若对于某常数ϵ>0，有$f(n) = O(n^{\log_b a - \varepsilon})$，则 $T(n) = \Theta(n^{\log_b a})$

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

3）若对某常数ϵ>0，有$f(n) = \Omega(n^{\log_b a + \varepsilon})$，且对常数c<1与所有足够大的n，有 $af(n/b) \le cf(n)$，则 $T(n) = \Theta(f(n))$。

理解主定理：

1）T(n)的解似乎与f(n)和 $n^{\log_b a}$ 有“密切关联”：

f(n)和 $n^{\log_b a}$ 比较，T(n)取了其中较大的一个。

如：第一种情况，函数 $n^{\log_b a}$ 比较大，所以 $T(n)=\Theta(n^{\log_b a})$

第三种情况，函数f(n) 比较大，所以 $T(n)=\Theta(f(n))$

第二种情况，两个函数一样大，则乘以对数因子，得

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

2) 在第一种情况 $f(n)=O(n^{\log_b a-\varepsilon})$ 中，f(n)要**多项式**地小于 $n^{\log_b a}$。即，对某个常量ϵ>0，f(n)必须渐近地小于 $n^{\log_b a}$，两者相差了一个 n^{ε} 因子。

3) 在第三种情况 $f(n) = \Omega(n^{\log_b a+\varepsilon})$ 中，f(n)不仅要大于 $n^{\log_b a}$ ，而且要多项式地大于 $n^{\log_b a}$，还要满足一个“规则性”条件 $af(n/b) \le cf(n)$ 。

4) 若递归式中的f(n)与 $n^{\log_b a}$ 的关系不满足上述性质：

- f(n)小于 $n^{\log_b a}$ ，但不是多项式地小于。
- f(n)大于 $n^{\log_b a}$ ，但不是多项式地大于。

则不能用主方法求解该递归式。

使用主方法：分析递归式满足主定理的哪种情形，然后直接写出相应的答案，而无需证明（保证正确）。

例　解递归式 $T(n)=9T(n/3)+n$

分析：这里，a=9，b=3，f(n)=n。

则 $n^{\log_b a}=n^{\log_3 9}=\Theta(n^2)$ 。

因为 $f(n)=n=O(n^{\log_3 9-\varepsilon})$，其中ϵ=1，

所以对应主定理的第一种情况，于是有

$$T(n)=O(n^2)$$

例　解递归式 $T(n)=T(2n/3)+1$

分析：这里，a=1，b=3/2，f(n)=1，

$n^{\log_b a}=n^{\log_{3/2}1}=n^0=1$，主定理第二种情况成立，

因为 $f(n)=\Theta(n^{\log_b a})=\Theta(1)$，故T(n)=Θ(log*n*)

例　　解递归式 $T(n) = 3T(n/4) + n\log n$

分析：这里，a=3，b=4，f(n)=nlog*n*,

$$n^{\log_b a} = n^{\log_4 3} = O(n^{0.793}) \quad ,$$

故，$f(n) = \Omega(n^{\log_4 3+\varepsilon})$ 成立，其中ϵ≈0.2。

同时，对足够大的n，$af(n/b) \le cf(n)$ 成立吗？

$$af(n/b) = 3(n/4)\log(n/4) \le (3/4)n\log n = cf(n)$$

其中c = 3/4。

所以第三种情况成立，T(n) = Θ(nlogn)。

例　　递归式 $T(n)=2T(n/2)+n\log n$ 能用主方法求解吗？

分析：这里，a=2，b=2，

$$n^{\log_b a}=n^{\log_2 2}=O(n)$$ ，

且，f(n)=nlog$n$ 渐进大于 $n^{\log_b a}=O(n)$

第三种情况成立吗？ $f(n)=\Omega(n^{\log_b a+\varepsilon})$

事实上不成立，因为对于任意正常数ϵ，

$$f(n)/n^{\log_b a}=(n\log n)/n=\log n<n^{\varepsilon}$$

不满足 $f(n)=\Omega(n^{\log_b a+\varepsilon})$ （这里要求 $f(n)/n^{\log_b a}>n^{\varepsilon}$）。

因此该递归式落在情况二和情况三之间，条件不成立，不能用主定理求解。

为什么主定理是正确的？

主定理证明：（略，P45）

注：在使用主定理时不用证明其正确性。

还有没有其它方法？

4）直接化简

根据递推关系，展开递推式，找出各项系数的构造规律（如等差、等比等），最后得出化简式的最终形式。

例：化简递归式 $T(n) = 2T(n/2) + n\log n$

$$
\begin{aligned}
T(n) &= 2T(n/2) + n\log n \\
&= 2(2T(n/4) + (n/2)\log(n/2)) + n\log n \\
&= 2^2 T(n/2^2) + n\log n - n + n\log n \\
&= 2^2 T(n/2^2) + 2n\log n - n \\
&= 2^2(2T(n/2^3) + (n/4)\log(n/4)) + 2n\log n - n \\
&= 2^3 T(n/2^3) + n\log n - 2n + 2n\log n - n \\
&= 2^3 T(n/2^3) + 3n\log n - 2n - n \\
&= \ldots \\
&= 2^k T(n/2^k) + kn\log n - n\sum_{i=1}^{k-1} i \\
&= n + kn\log n - n(k-1)k/2 \\
&= n + n\log^2 n - (n/2)\log^2 n + n\log n/2 \\
&= \mathrm{O}(n\log^2 n)
\end{aligned}
$$

4.5-1 对下列递归式，使用主方法求出渐近紧确界。

a. $T(n)=2T(n/4)+1$

b. $T(n)=2T(n/4)+\sqrt{n}$

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

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

a. $\Theta(\sqrt{n})$

b. $\Theta(\sqrt{n}\lg(n))$

c. $\Theta(n)$

d. $\Theta(n^2)$

作业

- 1）4.4-6
- 2）4.5-3

Thank You!

Q&A