第四章 分治法

Part 1 分治策略

实现策略分三步：

1. Divide 分解问题
2. Conquer 递归地求解子问题
3. 结合基础结论or计算得出结果

分治策略可以用递归的方式实现 递归实现的样式：

```
function:
    //基本情况的结果(触底后向上)
    if base case:
        do...


    //进入子问题并求解(向下)
    else if recursion case:
        function
        ...

    //必要的运算后得出结果(向上)
    make the result
    return result
```

Part 2 矩阵乘法的Strassen算法

一般算法：

```
MATRIX-MULTIPLY-RECURSIVE(A, B, C, n)
if n == 1
// Base case.
    c11 = c11 + a11 · b11
    return
// Divide.
partition A, B, and C into n/2 × n/2 submatrices
    A11, A12, A21, A22; B11, B12, B21, B22;
    and C11, C12, C21, C22; respectively
// Conquer.
MATRIX-MULTIPLY-RECURSIVE(A11, B11, C11, n/2)
MATRIX-MULTIPLY-RECURSIVE(A11, B12, C12, n/2)
MATRIX-MULTIPLY-RECURSIVE(A21, B11, C21, n/2)
MATRIX-MULTIPLY-RECURSIVE(A21, B12, C22, n/2)
MATRIX-MULTIPLY-RECURSIVE(A12, B21, C11, n/2)
MATRIX-MULTIPLY-RECURSIVE(A12, B22, C12, n/2)
MATRIX-MULTIPLY-RECURSIVE(A22, B21, C21, n/2)
MATRIX-MULTIPLY-RECURSIVE(A22, B22, C22, n/2)
```

Strassen算法的思路：

$$ a^2-b^2 = (a+b)*(a-b) $$

Strassen的做法

1. 把每个矩阵拆成四个矩阵
2. 计算10个矩阵加法，产生10个新矩阵

$$
\begin{aligned}
S_1 &= B_{12} - B_{22} , \\
S_2 &= A_{11} + A_{12} , \\
S_3 &= A_{21} + A_{22} , \\
S_4 &= B_{21} - B_{11} , \\
S_5 &= A_{11} + A_{22} , \\
S_6 &= B_{11} + B_{22} , \\
S_7 &= A_{12} - A_{22} , \\
S_8 &= B_{21} + B_{22} , \\
S_9 &= A_{11} - A_{21} , \\
S_{10} &= B_{11} + B_{12} .
\end{aligned}
$$

3. 计算7个矩阵乘法(递归)，产生7个新矩阵

$$
\begin{aligned}
P_1 &= A_{11} \cdot S_1 \quad (= A_{11} \cdot B_{12} - A_{11} \cdot B_{22}) , \\
P_2 &= S_2 \cdot B_{22} \quad (= A_{11} \cdot B_{22} + A_{12} \cdot B_{22}) , \\
P_3 &= S_3 \cdot B_{11} \quad (= A_{21} \cdot B_{11} + A_{22} \cdot B_{11}) , \\
P_4 &= A_{22} \cdot S_4 \quad (= A_{22} \cdot B_{21} - A_{22} \cdot B_{11}) , \\
P_5 &= S_5 \cdot S_6 \quad (= A_{11} \cdot B_{11} + A_{11} \cdot B_{22} + A_{22} \cdot B_{11} + A_{22} \cdot B_{22}) , \\
P_6 &= S_7 \cdot S_8 \quad (= A_{12} \cdot B_{21} + A_{12} \cdot B_{22} - A_{22} \cdot B_{21} - A_{22} \cdot B_{22}) , \\
P_7 &= S_9 \cdot S_{10} \quad (= A_{11} \cdot B_{11} + A_{11} \cdot B_{12} - A_{21} \cdot B_{11} - A_{21} \cdot B_{12}) .
\end{aligned}
$$

4. 计算好多个矩阵加法，修改原矩阵的内容

$$C_{11} = C_{11} + P_5 + P_4 - P_2 + P_6 .$$

Expanding the calculation on the right-hand side, with the expansion of each P_i on its own line and vertically aligning terms that cancel out, we see that the update to C_{11} equals

$$\begin{array}{lllllll}
A_{11} \cdot B_{11} & + A_{11} \cdot B_{22} & + A_{22} \cdot B_{11} & + A_{22} \cdot B_{22} & & & \\
 & & - A_{22} \cdot B_{11} & & + A_{22} \cdot B_{21} & & \\
 & - A_{11} \cdot B_{22} & & & & - A_{12} \cdot B_{22} & \\
 & & & - A_{22} \cdot B_{22} & - A_{22} \cdot B_{21} & + A_{12} \cdot B_{22} & + A_{12} \cdot B_{21} \\
\hline
A_{11} \cdot B_{11} & & & & & & + A_{12} \cdot B_{21} ,
\end{array}$$

which corresponds to equation (4.5). Similarly, setting

$$C_{12} = C_{12} + P_1 + P_2$$

means that the update to C_{12} equals

$$\begin{array}{lll}
A_{11} \cdot B_{12} & - A_{11} \cdot B_{22} & \\
 & + A_{11} \cdot B_{22} & + A_{12} \cdot B_{22} \\
\hline
A_{11} \cdot B_{12} & & + A_{12} \cdot B_{22} ,
\end{array}$$

corresponding to equation (4.6). Setting

$$C_{21} = C_{21} + P_3 + P_4$$

means that the update to C_{21} equals

$$\begin{array}{lll}
A_{21} \cdot B_{11} & + A_{22} \cdot B_{11} & \\
 & - A_{22} \cdot B_{11} & + A_{22} \cdot B_{21} \\
\hline
A_{21} \cdot B_{11} & & + A_{22} \cdot B_{21} ,
\end{array}$$

corresponding to equation (4.7). Finally, setting

$$C_{22} = C_{22} + P_5 + P_1 - P_3 - P_7$$

means that the update to C_{22} equals

$$\begin{array}{lllllll}
A_{11} \cdot B_{11} & + A_{11} \cdot B_{22} & + A_{22} \cdot B_{11} & + A_{22} \cdot B_{22} & & & \\
 & - A_{11} \cdot B_{22} & & & + A_{11} \cdot B_{12} & & \\
 & & - A_{22} \cdot B_{11} & & & - A_{21} \cdot B_{11} & \\
- A_{11} \cdot B_{11} & & & & - A_{11} \cdot B_{12} & + A_{21} \cdot B_{11} & + A_{21} \cdot B_{12} \\
\hline
 & & & A_{22} \cdot B_{22} & & & + A_{21} \cdot B_{12} ,
\end{array}$$

该算法时间复杂度的递推式如下图：

$$T(n) = 7T(n/2) + \Theta(n^2) .$$

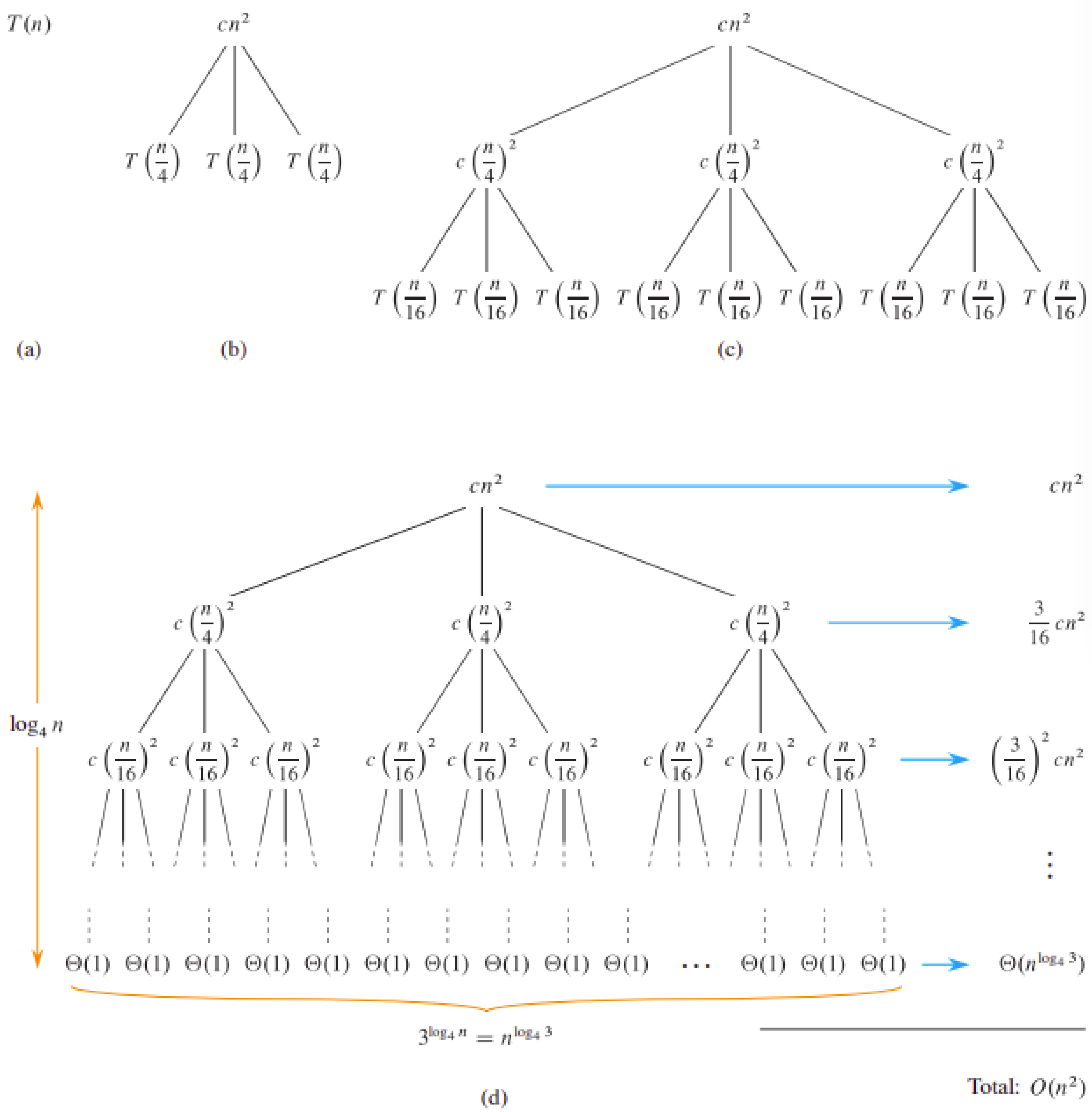
书上所说：Strassen算法的时间复杂度为：
$$ T(n) = \Theta(n^{\log_2 7}) $$

Part 3 分治法的时间复杂度分析

递归树

栗子:

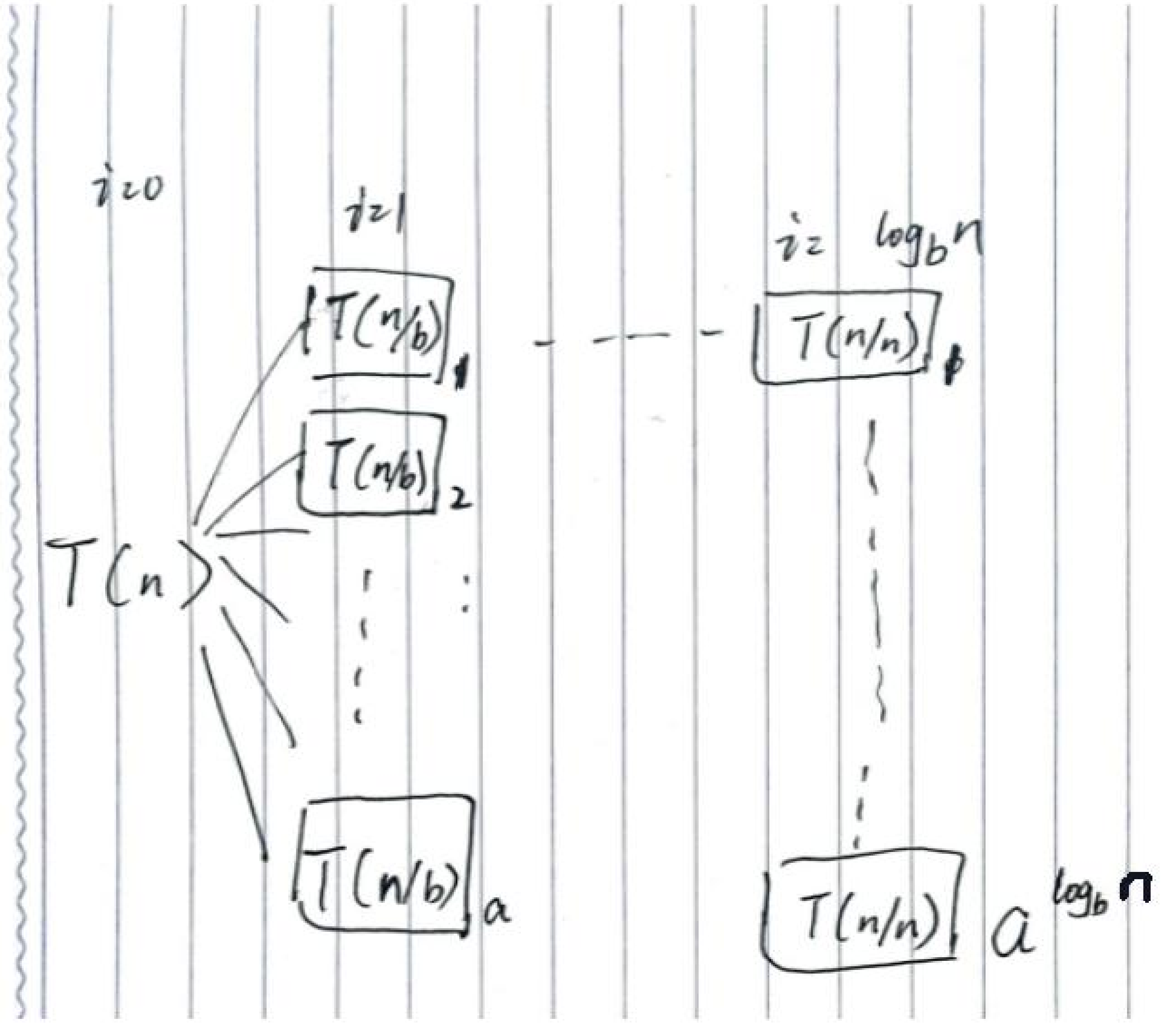
$$ T(n) = 3T(n/4)+cn^2 $$



$$
\begin{aligned}
T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n} cn^2 + \Theta(n^{\log_4 3}) \\
&= \sum_{i=0}^{\log_4 n} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{1}{1-(3/16)} cn^2 + \Theta(n^{\log_4 3}) \qquad \text{(by equation (A.7) on page 1142)} \\
&= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) \\
&= O(n^2) \qquad (\Theta(n^{\log_4 3}) = O(n^{0.8}) = O(n^2)) .
\end{aligned}
$$

问题1： 仿照上面的推导，能否类似画出strassen算法的递归树，求出strassen的时间复杂度？

尝试推导通式：
$$ T(n)=aT(n/b)+cn^d $$ 画图



有：
$$ T(n) = a^{\log_b n}T(1) + \Sigma_{i=0}^{\log_b n -1}[c(n/b^i)^d~a^{i}] $$ 提取系数：
$$ T(n) = c_2 n^{\log_b a} + c n^d \Sigma_{i=0}^{\log_b n -1}(a/b^d)^i $$ **若(a/b^d = 1),则：**
$$ T(n) = c_2 n^d +cn^d\log_b n = \Theta(n^d\log_b n) $$ 否则，等比数列求和：
$$ T(n) = c_2 n^{log_b a} + c_1 n^d[1-(a/b^d)^{\log_b n}]/(1-a/b^d) $$ 化简：
$$ T(n) = c_2 n^{log_b a} + c_1/(1-a/b^d)~ n^d[1-(a/b^d)^{\log_b n}] $$

$$ T(n) = c_2 n^{log_b a} + c_1/(1-a/b^d)~ n^d[1-n^{\log_b a -d}] $$ $$ T(n) = c_2 n^{log_b a} + c_1/(1-a/b^d)~ n^d - c_1/(1-a/b^d)n^{\log_b a} $$ $$ T(n) = [1- c_1/(1-a/b^d)]~ n^d - c_1/(c_2-a/b^d)n^{\log_b a} $$

重点(判断符号)

当a < b^d,即\log_b a < d:
(1-a/b^d) < 0 $$ T(n) = \Theta(n^d) $$

当a > b^d,即\log_b a > d:
(1-a/b^d) > 0 $$ T(n) = \Theta(n^{\log_a b}) $$

而这，就是**4-5中的主方法**的结论

Strassen算法通过减少树的分支(bush)使得计算量下降

下面是一些常见的形式，代入上面的式子也能得到相应的形式：

4.3-1

Use the substitution method to show that each of the following recurrences defined on the reals has the asymptotic solution specified:

a. $T(n) = T(n-1) + n$ has solution $T(n) = O(n^2)$.

b. $T(n) = T(n/2) + \Theta(1)$ has solution $T(n) = O(\lg n)$.

c. $T(n) = 2T(n/2) + n$ has solution $T(n) = \Theta(n \lg n)$.

d. $T(n) = 2T(n/2 + 17) + n$ has solution $T(n) = O(n \lg n)$.

e. $T(n) = 2T(n/3) + \Theta(n)$ has solution $T(n) = \Theta(n)$.

f. $T(n) = 4T(n/2) + \Theta(n)$ has solution $T(n) = \Theta(n^2)$.

Part 4 主方法的证明(没那个能力,故略)