Chap 4. Divide-and-Conquer

孫光天

Divide the problem into a number of subproblems that are **smaller** instances of **the same problem**.

Conquer the subproblems by solving them **recursively**.

Combine the solutions to the **subproblems** into the solution for the **original problem**.

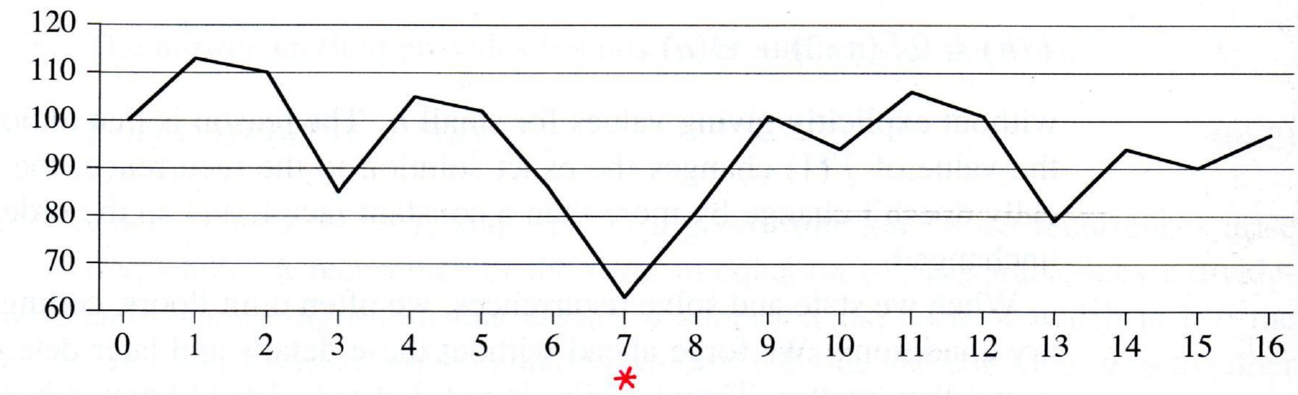
Recurrences: an equation or inequality that describes a function in terms of its value on **smaller** inputs. For example,

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

When we state and solve recurrences, we often omit floors, ceilings, and boundary conditions.

4.1 The maximum-subarray problem

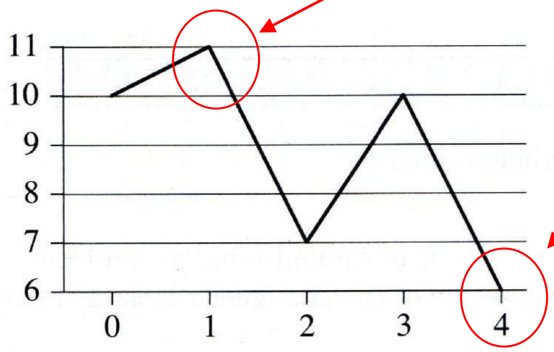
When you are restricted to buy and sell a stock, and allow to learn what the price of the stock will be in the future. Your goal is to maximize your profit. See Figure 4.1 a stock price for 17 days. (看成物品買賣)



Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Price	100	113	110	85	105	102	86	63	81	101	94	106	101	79	94	90	97
Change		13	−3	−25	20	−3	−16	−23	18	20	−7	12	−5	−22	15	−4	7

Wrong idea: 由最高 price 往左找最低 或 由最低 price 往右找最高

(Fig. 4.2 錯誤例)



Day	0	1	2	3	4
Price	10	11*	7	10	6*
Change		1	–4	3	–4

A brute force solution

找所有任兩天組合：**C(n, 2) → O(n^2)**

A transformation(轉爲另一形態表示)

We want to find a sequence of days over which the net change from the first day to the last is maximum. (找一連續片段時間其淨獲利和最大)

→ Maximum subarray problem

我們將兩天之間獲利情況建一 **array *A*， see Fig. 4.3**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	13	–3	–25	20	–3	–16	–23	18	20	–7	12	–5	–22	15	–4	7

maximum subarray

→ 第 **7** 天買 (第 **8** 天開始)到第 **11** 天 賣，可得最大獲利 ：**43** 元

→ Maximum subarray: A[8 . . 11]

(註：若 ***A*** 元素全部爲正值(一直漲)，則全長爲最大値)

Brute force: C(n-1, 2) → O(n^2).

A solution using divide-and-conquer

We can divide the array A into two subarrays:

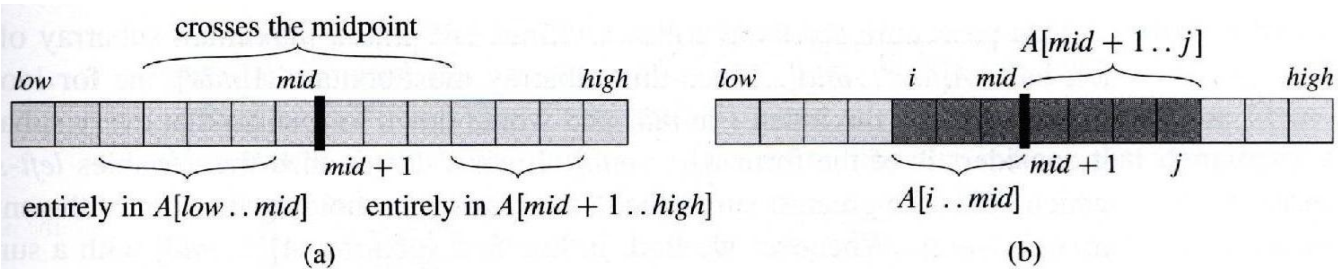
A[low .. mid], A[mid+1 .. high]

則最大值 **subarray A[i ..j]** 一定為其中之一情形 **(see Fig. 4.4 (a))**

1. 位於 **A[low .. mid]** 中

2. 位於 **A[mid+1 .. high]** 中

3. 位於 兩個 **subarray** 中間**(midpoint)**，**low ≦ i ≦mid < j ≦ high**



針對 **case 3**：只要找 **A[i..mid]**最大值**(**由 **mid** 往前找**)** 與 **A[mid+1 .. j]** 最大值 **(**由 **mid +1** 往後找**)**，兩者合併，則可得到 **A[i .. j]**。**(**上圖 **(b))**

演算法如下：

```
FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
left-sum = -∞
sum = 0
for i = mid downto low                    由 mid
    sum = sum + A[i]                      往前找
    if sum > left-sum
        left-sum = sum
        max-left = i
right-sum = -∞
sum = 0
for j = mid + 1 to high                   由 mid+1
    sum = sum + A[j]                      往後找
    if sum > right-sum
        right-sum = sum
        max-right = j
return (max-left, max-right, left-sum + right-sum)
```

整個演算法如下：

```
FIND-MAXIMUM-SUBARRAY(A, low, high)
1   if high == low
2       return (low, high, A[low])          // base case: only one element
3   else mid = ⌊(low + high)/2⌋
4       (left-low, left-high, left-sum) =
            FIND-MAXIMUM-SUBARRAY(A, low, mid)   左
5       (right-low, right-high, right-sum) =
            FIND-MAXIMUM-SUBARRAY(A, mid + 1, high)   右
6       (cross-low, cross-high, cross-sum) =
            FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)   Cross
7       if left-sum ≥ right-sum and left-sum ≥ cross-sum
8           return (left-low, left-high, left-sum)
9       elseif right-sum ≥ left-sum and right-sum ≥ cross-sum
10          return (right-low, right-high, right-sum)
11      else return (cross-low, cross-high, cross-sum)
```

Analyzing the divide-and-conquer

當 **n=1，line 2** 只花 **constant time Θ (1).**

n > 1:

$$\begin{aligned} T(n) &= \Theta(1) + 2T(n/2) + \Theta(n) + \Theta(1) \\ &= 2T(n/2) + \Theta(n) . \end{aligned}$$

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

所以，**T(n) =Θ (n log n)**

EXERCISE: Coding "FIND-Maximum-Subarray (*A*, low, high)" (上面 recurasive 方式)

Data: p.4-2 (Fig. 4.3) 範例 array A

(下週 報告 1. 演算法 與 Source code ; 2. 執行過程(要印出)； 3.結果)

4.2 Strassen's algorithm for matrix multiplication

For the $n \times n$ matrices A, B and C. If C = A $\times$ B, then

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj} .$$

演算法：

```
SQUARE-MATRIX-MULTIPLY(A, B)
n = A.rows
let C be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        c_ij = 0
        for k = 1 to n
            c_ij = c_ij + a_ik · b_kj
return C
```

Time complexity = $O(n^3)$

Divide-and-Conquer

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix},$$

so that we rewrite the equation $C = A \cdot B$ as

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}.$$

Equation (4.10) corresponds to the four equations

$$\begin{aligned} C_{11} &= A_{11} \cdot B_{11} + A_{12} \cdot B_{21} , \\ C_{12} &= A_{11} \cdot B_{12} + A_{12} \cdot B_{22} , \\ C_{21} &= A_{21} \cdot B_{11} + A_{22} \cdot B_{21} , \\ C_{22} &= A_{21} \cdot B_{12} + A_{22} \cdot B_{22} . \end{aligned}$$

Time complexity =

$$
\begin{aligned}
T(n) &= \Theta(1) + 8T(n/2) + \Theta(n^2) \\
&= 8T(n/2) + \Theta(n^2)\ .
\end{aligned}
$$

(8 個乘法， 4 個 $n^2/4$ 加法= $O(n^2)$)

T(n) =Θ (n^3) **(此時，divide-and-conquer 並未節省時間？)**

改進：

Strassen's method

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

7 個乘法

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

此時，$C_{11}, C_{12}, C_{21}, C_{22}$ 爲

$C_{11} = P_5 + P_4 - P_2 + P_6$. ，　將 P_2, P_4, P_5, P_6 代入：

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

$C_{12} = P_1 + P_2$,

and so C_{12} equals

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

$C_{21} = P_3 + P_4$

makes C_{21} equal

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

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

so that C_{22} equals

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

求 **P1 ~ P7，共需 7 個乘法：**

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 , \\ 7T(n/2) + \Theta(n^2) & \text{if } n > 1 . \end{cases}$$

$T(n) = \Theta(n^{\lg 7}) = \Theta(n^{2.81})$

4.3 The substitution method

The substitution method for solving recurrences entails two steps:

1. **Guess the form of the solution.**
2. **Use mathematical induction to find the constants and show that solution works.**

For example,

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

We guess that the solution is $T(n) = O(n \lg n)$

We want to prove $T(n) \le c\, n \lg n$

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

For the boundary condition n=1, T(1) = c 1 lg 1 =0 is at odds with T(1) =1. (n=1 與題目已知條件不符) The base case of our inductive proof fails to hold. But for the base cases of *n* =2 and *n* =3 to hold. (符合) It is straightforward to extend boundary conditions to make the inductive (歸納) assumption work for small *n*.

Making a good guess

For example, $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$.

We may start with a **lower bound of $T(n) = \Omega(n)$** and an **upper bound of $T(n) = O(n^2)$**. Then, we can gradually **lower the upper bound** and **raise the lower bound** until we converge on the correct, asymptotically tight solution of $T(n) = \Theta(n \lg n)$.

Subtleties (巧妙、敏銳)

We can revise the guess by subtracting a lower-bound term often permits the math go through. For example, $T(n)=T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$.

We guess $T(n) = O(n)$, and try to show that $T(n) \le cn$.

$T(n) \le \quad c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1$

$= c\,n + 1$ **does not imply** $T(n) \le cn$? (差一點)

Try a **larger** guess $T(n) = O(n^2)$ which works. (符合，但是值太大 ($T(n^2/2) = c(n/2)^2 = c\,n^2/4$)

A new guess by subtracting a lower-order term,

a **new guess** $T(n) \le cn - b$

where $b \ge 0$ is constant. We now have

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

(OK !)

Avoiding pitfalls (犯錯)

For example, $T(n) = O(n)$ by **guessing** $T(n) \le cn$ and prove

$$\begin{aligned} T(n) &\le 2(c\lfloor n/2 \rfloor) + n \\ &\le cn + n \\ &= O(n) , \qquad \Longleftarrow \textit{wrong!!} \end{aligned}$$

(是小於 cn 而非 cn + n)

We **haven't proved** the **exact form** of the **inductive hypothesis**, that is, that $T(n) \le cn$

Changing variables

We can change an unknown recurrence similar to one you have been seen before. For example,

$$T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \lg n ,$$

which looks difficult. We can simplify this recurrence, though, wit variables. For convenience, we shall not worry about rounding offi as $\sqrt{n}$, to be integers. Renaming $m = \lg n$ yields

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

$$\boxed{\mathbf{n = 2^m}}$$

We can now rename $S(m) = T(2^m)$ to produce the new recurrence

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

令此式 $S(m) = T(2^m)$:

which is very much like recurrence (4.4). Indeed, this new recurr same solution: $S(m) = O(m \lg m)$. Changing back from $S(m)$ to $T($
$T(n) = T(2^m) = S(m) = O(m \lg m) = O(\lg n \lg \lg n)$.

[補充 ： recursive 代入法 (假設 $n = 2^k$)]

$$
\begin{aligned}
T(n) &= 2\boxed{T(\frac{n}{2})} + cn \\
&= 2\boxed{[2T(\frac{n}{4}) + c \cdot \frac{n}{2}]} + cn \\
&= 2^2 T(\frac{n}{2^2}) + cn + cn \\
&= 2^2 T(\frac{n}{2^2}) + 2cn \\
&= 2^3 T(\frac{n}{2^3}) + 3cn \\
&= \ldots\ldots\ldots\ldots \\
&= 2^k T(\frac{n}{2^k}) + kcn \\
&= 2^k T(1) + kcn \\
&= an + c \cdot n \cdot \log n
\end{aligned}
$$

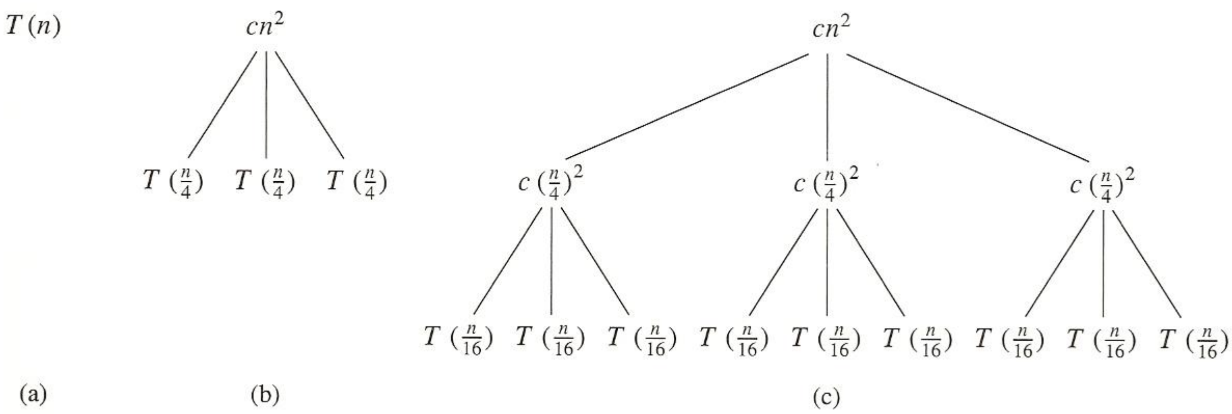
=> $T(n) = O(n \log n)$

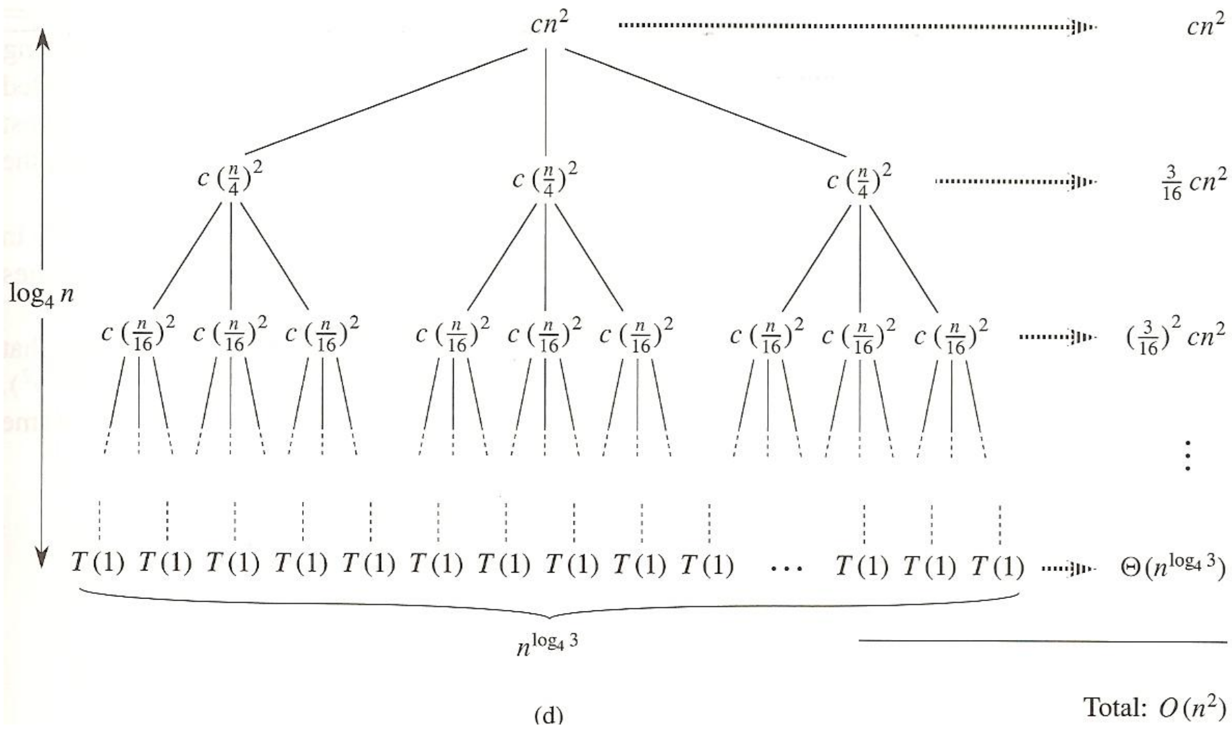
4.4 The recursion-tree method

We will use the recursion trees to generate good guesses. For example, $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$, a recurrence $T(n) = 3T(n/4) + c\,n^2$.

Figure 4.1 shows the recursion tree.

Recursion tree for $T(n) = 3T(n/4) + c\,n^2$



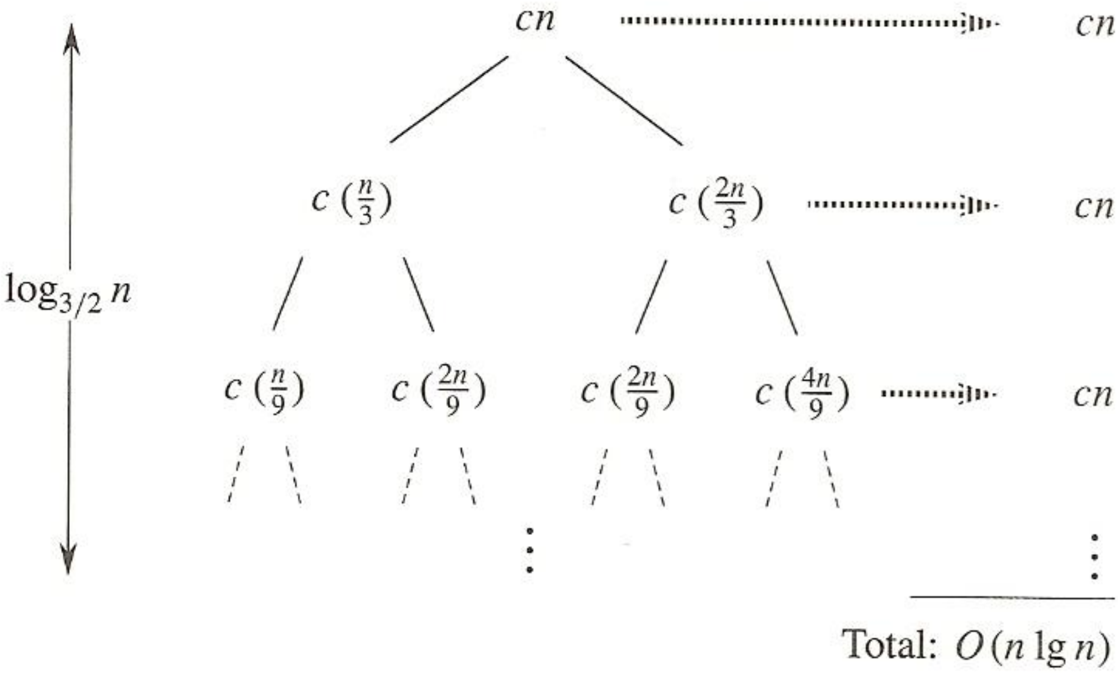


最後一層有 3^k 個 (其中， $k = \log_4 n$，所以 $3^{\log_4 n} = n^{\log_4 3}$ 個 T(1))

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

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

Another example, T(n) = T(n/3) + T(2n/3) + O(n)



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

4.5 The Master method

The master method provide a "cookbook" method for solving recurrences of the form T(n) = a T(n/b) + f(n), where a ≥ 1 and b ≥ 1 are constants and f(n) is an asymptotically positive function.

Theorem 4.1 (Master theorem)

Let $a \ge 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$\boxed{T(n) = aT(n/b) + f(n)\ ,}$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ can be bounded asymptotically as follows.

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $\underline{T(n) = \Theta(n^{\log_b a})}$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $\underline{T(n) = \Theta(n^{\log_b a}\lg n)}$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \le cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $\underline{T(n) = \Theta(f(n))}$. ■

For example,

1. **T(n) = 9T(n/3) + n, a = 9, b=3, f(n) = n.** $n^{\log_b a} = n^{\log_3 9} = n^2$. **Since f(n) = O($n^{2-1}$), the solution is T(n) = Θ(n^2)**
2. **T(n) = T(2n/3) + 1, a=1, b=3/2, f(n)=1, and** $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$. **Since f(n) =Θ($n^0$) = Θ(1), the solution T(n) =Θ(lg n)**
3. **T(n)=3T(n/4)+n lgn, a=3, b=4, f(n)=*n*lg *n*, and** $n^{\log_b a} = n^{\log_4 3} = n^{0.793}$

Since f(n) = Ω($n^{\log_4 3+\varepsilon}$), where ε ≈ 0.2, the solution T(n) =Θ(*n* lg *n*)

The master method does not apply to any recurrence, for example,

T(n) = 2 T(n/2) + n lg n. a=2, b=2, f(n) = n lg n is asymptotically larger than $n^{\log_b a} = n$. The problem is that it is not polynomially larger. The ratio f(n)/ $n^{\log_b a}$ = (n lg n) / n = lg n is asymptotically less than n^{ε} for any positive constant ε. (沒有一常數 ε 可以 bound 住) Consequently, the recurrence falls into the gap between case 2 and case 3.

(Home work Ans: $n\lg^2 n$ (= n (lg n)^2))

EXERCISE: 解 T(n) = 2 T(n/2) + n lg n