遞迴、分治法與主定理

完整教材 (依據 5CCS2FC2 Foundations of Computing II 第三週內容編寫)

附:Cook–Levin 定理 (Sat 是 NP-complete)

整理自 Dr. Christopher Hampson(King's College London) 之投影片
並參考補充文獻擴充編寫

2026 年 6 月

本週學習目標

1. 熟練**數學歸納法**與**強歸納法**, 並用以證明遞迴關係的封閉式解。
2. 理解**分治法 (Divide-and-Conquer)** 的三步驟, 並掌握兩個經典演算法: **河內塔**與**合併排序**。
3. 學會由演算法**建立遞迴關係式**, 再求其漸進成長率。
4. 精通**主定理 (Master Theorem)** 的三種情形, 並能快速套用。
5. 理解 **Cook–Levin 定理**如何把任意 **NP** 問題的計算「編碼成布林公式」, 證明 Sat 是 NP-complete。

Contents

1 數學歸納法: 代入法 **3**
1.1 (弱) 歸納法的骨架 . . . 3
1.2 範例一:$T(n) = 2T(n-1) + 1$ . . . 3
1.3 範例二:$T(n) = T(n-1) + n$ . . . 3

2 強歸納法 **4**
2.1 範例三:$T(n) = 2T(\lceil n/2 \rceil) + n$ 的下界 . . . 4
2.2 範例四: 費氏數列的 Binet 公式 . . . 4

3 分治法 (Divide-and-Conquer) **5**

4 兩個經典分治演算法 **5**
4.1 河內塔 (The Towers of Hanoi) . . . 5
4.2 合併排序 (Merge Sort) . . . 6

5 主定理 (The Master Theorem) **7**
5.1 為什麼成立: 遞迴樹的直覺 . . . 8
5.2 套用範例 . . . 8

6 Cook–Levin 定理:SAT 是 NP-complete **9**
6.1 第一步: 任取 NP 問題, 取得多項式時間 NDTM 9
6.2 第二步: 用命題變數描述「計算表 (tableau)」 9
6.3 第三步: 用四組子公式約束「合法且接受的計算表」 10
6.4 第四步: 結論 10

7 本週重點整理 **11**

8 練習題 (附解答) **11**

參考資料 **12**

1 數學歸納法: 代入法

分析遞迴演算法時, 我們常先猜出一個封閉式解, 再用**數學歸納法 (proof by induction)** 驗證它正確。歸納法也常被稱為**代入法 (method of substitution)**。

1.1 (弱) 歸納法的骨架

數學歸納法

要證明命題 $P(n)$ 對所有 $n \geq 1$ 成立:

- **基底情形 (Base Case)**: 證明 $P(1)$ 成立。
- **歸納步驟 (Inductive Case)**:

  (i) **假設** $P(k)$ 對某個 $k \geq 1$ 成立 (此為**歸納假設**,Induction Hypothesis, 簡記 (I.H.));
  (ii) **代入**證明 $P(k+1)$ 也成立。

直覺上這就是「骨牌效應」: 第一張倒 (基底), 且每一張都會推倒下一張 (歸納步驟), 於是全部都倒。

1.2 範例一:$T(n) = 2T(n-1) + 1$

定理 1.1. 設遞迴關係 $T(1) = 1,\ T(n) = 2T(n-1) + 1 (n > 1)$。則對所有 $n \geq 1$,

$$T(n) = 2^n - 1.$$

Proof. **基底情形.** 在 $n = 1$: 左式 $T(1) = 1$; 右式 $2^1 - 1 = 1$。兩者相等。✓

歸納步驟. 假設 (I.H.):$T(k) = 2^k - 1$ 對某個 $k \geq 1$ 成立。則在 $n = k + 1$:

$$T(k+1) = 2\,T(k) + 1 \overset{\text{(I.H.)}}{=} 2(2^k - 1) + 1 = (2 \cdot 2^k - 2) + 1 = 2^{k+1} - 2 + 1 = 2^{k+1} - 1.$$

這正是 $n = k + 1$ 時的目標式。由歸納法, 結論對所有 $n \geq 1$ 成立。 □

1.3 範例二:$T(n) = T(n-1) + n$

定理 1.2. 設 $T(1) = 1,\ T(n) = T(n-1) + n (n > 1)$。則對所有 $n \geq 1$,

$$T(n) = \frac{n^2 + n}{2}.$$

Proof. **基底情形.** $n = 1$: 左式 $T(1) = 1$; 右式 $\dfrac{1^2 + 1}{2} = 1$。✓

歸納步驟. 假設 (I.H.):$T(k) = \dfrac{k^2 + k}{2}$。則

$$T(k+1) = T(k) + (k+1) \overset{\text{(I.H.)}}{=} \frac{k^2 + k}{2} + (k+1) = \frac{k^2 + k}{2} + \frac{2(k+1)}{2} = \frac{k^2 + 3k + 2}{2} = \frac{(k+1)^2 + (k+1)}{2}.$$

故結論對所有 $n \geq 1$ 成立。 □

備註 1.3. 這就是著名的「$1+2+\cdots+n=\dfrac{n(n+1)}{2}$」。歸納法把「驗證一條公式」變成兩個簡單步驟。

2 強歸納法

有些遞迴會用到多個前項 (例如 $T(n)$ 同時依賴 $T(n-1)$ 與 $T(n-2)$), 或依賴 $T(\lceil n/2\rceil)$ 這種「跳很遠」的項。這時單一的「假設 $P(k)$」不夠用, 需要**強歸納法**。

強歸納法 (Strong Induction)

- **基底情形**: 證明 $P(1)$ 成立 (有時需要多個基底)。
- **歸納步驟**:

  (i) **假設** $P(m)$ 對**所有** $m\le k$ 成立 ((I.H.));
  (ii) **代入**證明 $P(k+1)$ 也成立。

差別只在歸納假設更強: 從「假設前一項」變成「假設前面所有項」。

2.1 範例三:$T(n)=2T(\lceil n/2\rceil)+n$ 的下界

定理 2.1. 設 $T(1)=1,\ T(n)=2\,T\big(\lceil\frac{n}{2}\rceil\big)+n\,(n>1)$。則對所有 $n\ge 1$,

$$T(n)\ge n\log_2 n.$$

Proof. **基底情形.** $n=1$: 左式 $T(1)=1$; 右式 $1\cdot\log_2 1=0$。確有 $1\ge 0$。✓

歸納步驟. 假設 (I.H.):$T(m)\ge m\log_2 m$ 對所有 $m\le k$ 成立。在 $n=k+1$, 因 $\lceil(k+1)/2\rceil\le k$, 可套用 (I.H.):

$$\begin{aligned}
T(k+1)&=2\,T\big(\lceil\tfrac{k+1}{2}\rceil\big)+(k+1)\\
&\overset{\text{(I.H.)}}{\ge} 2\,\lceil\tfrac{k+1}{2}\rceil\log_2\big(\lceil\tfrac{k+1}{2}\rceil\big)+(k+1)\\
&\ge 2\cdot\tfrac{k+1}{2}\,\log_2\big(\tfrac{k+1}{2}\big)+(k+1)\qquad(\text{因 }\lceil x\rceil\ge x)\\
&=(k+1)\big[\log_2(k+1)-1\big]+(k+1)\\
&=(k+1)\log_2(k+1).
\end{aligned}$$

故 $T(n)\ge n\log_2 n$ 對所有 $n\ge 1$ 成立。 □

備註 2.2. 此例正是合併排序的時間下界:$T(n)\in\Omega(n\log n)$。下一節會看到它的遞迴正是這個形式。

2.2 範例四: 費氏數列的 Binet 公式

定理 2.3 (Binet 公式)**.** 設費氏數 $F(0)=0,\ F(1)=1,\ F(n)=F(n-1)+F(n-2)\,(n>1)$。則

$$F(n)=\frac{\Phi^n-\varphi^n}{\sqrt{5}},\qquad\text{其中}\quad\Phi=\frac{1+\sqrt{5}}{2},\quad\varphi=\frac{1-\sqrt{5}}{2}$$

為兩個黃金比例 (它們是 $x^2=x+1$ 的兩根, 故 $\Phi^2=\Phi+1$,$\varphi^2=\varphi+1$)。

Proof. **基底情形.** 由於遞迴用到兩個前項, 需**兩個**基底。$n=0$:$\dfrac{\Phi^0-\varphi^0}{\sqrt{5}}=\dfrac{1-1}{\sqrt{5}}=0=F(0)$ ✓; $n=1$:$\dfrac{\Phi^1-\varphi^1}{\sqrt{5}}=\dfrac{\Phi-\varphi}{\sqrt{5}}=\dfrac{\sqrt{5}}{\sqrt{5}}=1=F(1)$ ✓。

歸納步驟. 假設 (I.H.):$F(m)=\dfrac{\Phi^m-\varphi^m}{\sqrt{5}}$ 對所有 $m\le k$ 成立。則

$$\begin{aligned}F(k+1)&=F(k)+F(k-1)\overset{\text{(I.H.)}}{=}\frac{\Phi^k-\varphi^k}{\sqrt{5}}+\frac{\Phi^{k-1}-\varphi^{k-1}}{\sqrt{5}}\\&=\frac{(\Phi^k+\Phi^{k-1})-(\varphi^k+\varphi^{k-1})}{\sqrt{5}}=\frac{(\Phi+1)\Phi^{k-1}-(\varphi+1)\varphi^{k-1}}{\sqrt{5}}\\&=\frac{\Phi^2\cdot\Phi^{k-1}-\varphi^2\cdot\varphi^{k-1}}{\sqrt{5}}\qquad(\text{用 }\Phi^2=\Phi+1,\ \varphi^2=\varphi+1)\\&=\frac{\Phi^{k+1}-\varphi^{k+1}}{\sqrt{5}}.\end{aligned}$$

故公式對所有 $n\ge 0$ 成立。 □

何時用強歸納法?

當 $T(k+1)$ 依賴的不只是 $T(k)$, 而是更前面或更小的項 (如 $T(k-1)$、$T(\lceil (k+1)/2\rceil)$) 時, 就需要「假設前面所有項都成立」的強歸納法。費氏數列 (依賴兩項)、分治演算法 (依賴 $T(n/2)$) 都是典型情境。

3 分治法 (Divide-and-Conquer)

分治法三步驟

- **分 (Divide)**: 把問題切成數個結構相同但規模更小的子問題。
- **治 (Conquer)**: 遞迴地解這些子問題。
- **合 (Combine)**: 把子問題的解組合成原問題的解。

分治法的執行成本天生具有**遞迴關係**的形式:

$$T(n)=\underbrace{a}_{\text{子問題個數}}\cdot T\Big(\underbrace{\tfrac{n}{b}}_{\text{子問題規模}}\Big)+\underbrace{f(n)}_{\text{分與合的成本}}.$$

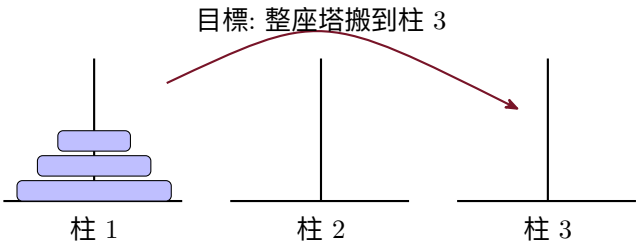
第 5 節的主定理就是用來「一眼解出」這類遞迴的工具。

4 兩個經典分治演算法

4.1 河內塔 (The Towers of Hanoi)

把 n 個大小不一的圓盤從第 1 根柱子搬到第 3 根, 規則:

- **限制 1**: 一次只能搬**一個**圓盤。
- **限制 2**: 任何時候都**不能**把大盤疊在小盤上。



分治思路: 要把 n 個盤從柱 1 搬到柱 3, 只需:(1) 把上面 $n-1$ 個盤搬到柱 2;(2) 把最大的盤搬到柱 3;(3) 再把 $n-1$ 個盤從柱 2 搬到柱 3。

Algorithm 1 MoveTower(n, from, to)

```
if n = 1 then
    Move(from, to)
else
    MoveTower(n − 1, from, other)
    Move(from, to)
    MoveTower(n − 1, other, to)
```

定理 4.1. *MoveTower* 所需的搬移次數滿足

$$T(1) = 1, \qquad T(n) = 2\,T(n-1) + 1 \quad (n > 1),$$

因此 *(*由定理 *1.1)*$T(n) = 2^n - 1$。

Proof. 演算法對規模 n 呼叫自己兩次 (各為規模 $n-1$), 中間加一次 Move:

$$T(n) = T(n-1) + 1 + T(n-1) = 2T(n-1) + 1.$$

封閉式 $T(n) = 2^n - 1$ 已於定理 1.1用歸納法證明。 □

n	1	2	3	4	5	6	7	8
$T(n)$	1	3	7	15	31	63	127	255

備註 4.2 (指數爆炸). $T(n) = 2^n - 1 \in \Theta(2^n)$。傳說中 64 個金盤的河內塔需 $2^{64} - 1 \approx 1.8 \times 10^{19}$ 次搬移; 即使每秒一次, 也要約 5850 億年——遠超宇宙年齡。這是「指數演算法不可行」的經典寫照。

4.2 合併排序 (Merge Sort)

把陣列排序的分治法: 對半切、各自遞迴排序、再**合併**兩個已排序的半段。

Algorithm 2 MergeSort($[a_1, \dots, a_n]$)

```
if n = 1 then
    return [a_1]
else
    L ← MergeSort([a_1, ..., a_{⌊n/2⌋}])
    R ← MergeSort([a_{⌊n/2⌋+1}, ..., a_n])
    A ← Merge(L, R)
    return A
```

其中 Merge 把兩段已排序陣列「拉鍊式」併成一段, 需 $\Theta(n)$ 時間 (每個元素比較、放置一次)。

定理 4.3. *MergeSort* 的步數滿足

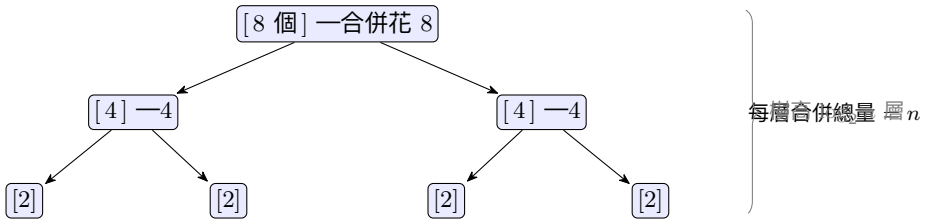
$$T(1) = 1, \qquad T(n) = T\big(\lfloor \tfrac{n}{2} \rfloor\big) + T\big(\lceil \tfrac{n}{2} \rceil\big) + n \;\approx\; 2\,T\big(\lceil \tfrac{n}{2} \rceil\big) + n.$$

Proof. 兩次遞迴呼叫各處理約一半元素, 合併花 n 步:$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + n$。 □

由定理 2.1已知 $T(n) \geq n \log_2 n$; 下一節用主定理可得緊確界 $T(n) = \Theta(n \log n)$。

n	1	2	3	4	5	6	7	8
$T(n)$	1	4	11	12	27	28	31	32

遞迴展開的視覺化 以 $n = 8$ 為例, 合併排序的呼叫樹: 每層的「合併工作」總量都是 $n = 8$, 而樹高為 $\log_2 n = 3$, 故總工作量 $\approx n \log_2 n$。



5 主定理 (The Master Theorem)

主定理是分治遞迴的「萬用解算器」: 不必每次都展開或歸納, 套公式即可得到漸進成長率。

定理 5.1 (主定理)**.** 設 $T(n)$ 是單調遞增的遞迴關係

$$T(n) = a\,T\Big(\frac{n}{b}\Big) + f(n), \qquad a \geq 1,\ b \geq 2 \text{ 為常數.}$$

令**臨界指數**$k = \log_b a$。則*:*

$$T(n) = \begin{cases} \Theta\big(n^k\big) & \text{若} f(n) \in O(n^{k-\varepsilon}) \quad \textit{(情形 1: 葉子主導)} \\ \Theta\big(n^k \log_2 n\big) & \text{若} f(n) \in \Theta(n^k) \quad \textit{(情形 2: 各層均衡)} \\ \Theta(f(n)) & \text{若} f(n) \in \Omega(n^{k+\varepsilon})\ (\dagger) \quad \textit{(情形 3: 根部主導)} \end{cases}$$

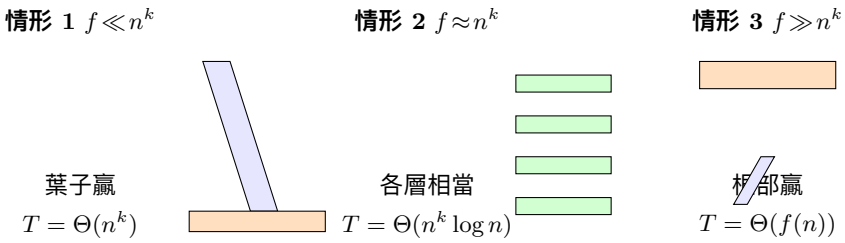
其中 $\varepsilon > 0$ 為某常數;(†) 還需**正則條件**$a\,f(n/b) \le c\,f(n)$ 對某 $c < 1$ 成立 *(*分治演算法幾乎都滿足*)*。

5.1 為什麼成立: 遞迴樹的直覺

把遞迴展開成一棵樹: 根做 $f(n)$ 的工作, 分裂出 a 個子問題, 每個規模 n/b.....

- 樹高 $h = \log_b n$(規模一路除以 b 直到 1)。
- 第 i 層有 a^i 個節點, 每個規模 n/b^i。
- 葉子數 $= a^{\log_b n} = n^{\log_b a} = n^k$。

總成本 $T(n) = \Theta(n^k) + \displaystyle\sum_{i=0}^{\log_b n - 1} a^i\, f\!\left(\frac{n}{b^i}\right)$。比較「葉子工作 n^k」與「根部工作 $f(n)$」誰大:



- **情形 1**:f 增長慢於 n^k, 成本集中在葉子 (數量 n^k), 故 $T = \Theta(n^k)$。
- **情形 2**: 每層工作量都 $\approx n^k$, 共 $\log_b n$ 層, 故 $T = \Theta(n^k \log n)$。
- **情形 3**:f 增長快於 n^k, 成本集中在根部, 故 $T = \Theta(f(n))$。

證明概要 *(*假設 n 為 b 的次方*)*. 由幾何級數 $\displaystyle\sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i c\,n^d (f(n) = c\,n^d)$: 比值 a/b^d 與 1 的大小, 正對應 $d < k$、$d = k$、$d > k$ 三種情形, 分別讓級數收斂 (葉子主導)、各項相等 (乘上層數 $\log n$)、或被首項主導 (根部)。樓地板/天花板的取整不影響漸進階。 □

5.2 套用範例

例 5.2 (情形 3)**.** $T(n) = 9\,T\!\left(\left\lfloor \frac{n}{3} \right\rfloor\right) + \sqrt{(n+1)^5}$。

- 參數:$a = 9,\ b = 3 \Rightarrow k = \log_3 9 = 2$。
- $f(n) = \sqrt{(n+1)^5} \in \Theta(n^{2.5})$。
- 因 $2.5 > 2$, 即 $f \in \Omega(n^{k+\varepsilon})$, 屬**情形 3**。

故 $T(n) = \Theta(n^{2.5})$。

例 5.3 (情形 2 ——合併排序)**.** $T(n) = 2\,T\!\left(\left\lceil \frac{n}{2} \right\rceil\right) + n$。

- 參數:$a = 2,\ b = 2 \Rightarrow k = \log_2 2 = 1$。
- $f(n) = n \in \Theta(n^1)$, 與 n^k 同階, 屬**情形 2**。

故 $T(n) = \Theta(n \log_2 n)$——印證了合併排序的 $\Theta(n \log n)$。

例 5.4 (情形 1)**.** $T(n) = 2\,T\big(\lfloor \frac{n}{3} \rfloor\big) + \log_2(5n^2)$。

- 參數:$a = 2,\ b = 3 \Rightarrow k = \log_3 2 \approx 1.5849$。
- $f(n) = \log_2(5n^2) \in \Theta(\log_2 n)$, 增長遠慢於 n^k, **屬情形 1**。

故 $T(n) = \Theta\big(n^{\log_3 2}\big)$。

套用主定理的三步驟 SOP

1. **找參數**: 辨識 a, b, 算出 $k = \log_b a$。
2. **找 $f(n)$ 的階**: 把 $f(n)$ 化為 $\Theta(n^d)$ 或 $\Theta(\log \cdots)$。
3. **比較並判定情形**: 比 $f(n)$ 與 n^k——f 小 $\to$ 情形 1、相等 $\to$ 情形 2、f 大 $\to$ 情形 3。

小提醒: 主定理不能處理所有遞迴 (例如 a 不是常數、f 落在「縫隙」中、或非 $aT(n/b)$ 形式); 這些情況須回到遞迴樹或 (強) 歸納法。

6 Cook–Levin 定理:SAT 是 NP-complete

第二週我們已證 SAT$\in$ **NP**, 並指出「只要找到一個 NP-hard 問題, 再歸約過去」即可證其他問題 NP-hard。本節補上最關鍵的**第一塊骨牌**:SAT 本身就是 NP-hard。

定理 6.1 (Cook–Levin 定理)**.** 布林可滿足性問題 *SAT* 是 ***NP-complete***。

要證明的是: 每個問題 $X \in$ **NP** 都能多項式歸約到 SAT, 即 $X \le_p$ SAT。核心想法是把「NDTM 的一段接受計算」**編碼成一個布林公式**, 讓

$$M \text{ 有一段長度為多項式、接受} w \text{ 的計算} \iff F_{M,w} \text{ 可滿足.}$$

6.1 第一步: 任取 NP 問題, 取得多項式時間 NDTM

設 $X \in$ **NP**。依定義存在非確定性圖靈機 M, 使得

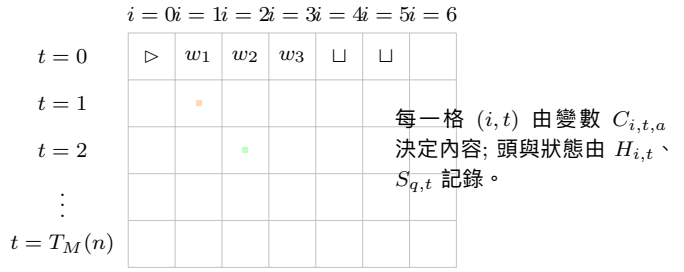
$$M \text{ 有一段長度為多項式的計算接受} w \iff w \in X,$$

且計算步數 $\le T_M(n)$ 為 $n = |w|$ 的多項式。

6.2 第二步: 用命題變數描述「計算表 (tableau)」

把 M 在 w 上某條計算路徑畫成一張 $T_M(n) \times T_M(n)$ 的**計算表**: 第 t 列代表「時間 t 的組態 (整條磁帶 + 狀態 + 讀寫頭位置)」。對每個狀態 $q \in Q$、符號 $a \in \Sigma$、與 $i, t \le T_M(n)$, 引入命題變數:

變數	為真的意義
$C_{i,t,a}$	在時間 t, 磁帶第 i 格的內容是符號 a
$H_{i,t}$	在時間 t, 讀寫頭位於第 i 格
$S_{q,t}$	在時間 t, 機器處於狀態 q



由於 $T_M(n)$ 是多項式, 變數總數 $O(T_M(n)^2)$ 也是**多項式**——這是整個歸約能在多項式時間完成的關鍵。

6.3 第三步: 用四組子公式約束「合法且接受的計算表」

公式 $F_{M,w} = \varphi_{\text{cell}} \wedge \varphi_{\text{start}} \wedge \varphi_{\text{move}} \wedge \varphi_{\text{accept}}$, 各部分意義如下, 並附投影片中的具體例子:

- $\varphi_{\textbf{cell}}$**(格子合法)**: 每格恰好含一個符號; 機器在每個時刻恰好處於一個狀態。
  - 「機器不能在 $t=2$ 同時處於 q_4 與 q_5」: $\neg\big(S_{q_4,2} \wedge S_{q_5,2}\big)$
  - 「若 $t=3$、格 1 是 a, 則它不是 b」: $C_{3,1,a} \to \neg C_{3,1,b}$
- $\varphi_{\textbf{start}}$**(起始組態)**: 第 0 列正是初始組態 $(q_{\text{init}}, \varepsilon, w)$, 磁帶寫著輸入 w。
- $\varphi_{\textbf{move}}$**(轉移合法)**: 每一列都依 M 的轉移函數 δ 由前一列推得。例如:
  - 「$t=3$ 時頭必在位置 $0,1,2$ 或 3 之一」: $(H_{0,3} \vee H_{1,3} \vee H_{2,3} \vee H_{3,3})$
  - 「若 $t=6$ 機器在 q_1、頭在位置 4、格 4 是 a, 則 $t=7$ 機器在 q_2、頭移到位置 3、格 4 變成 b」:
    $$(S_{q_1,6} \wedge H_{4,6} \wedge C_{4,6,a}) \;\to\; (S_{q_2,7} \wedge H_{3,7} \wedge C_{4,7,b})$$

  這類「相鄰兩列、局部 2×3 視窗合法」的約束, 確保整張表是一條真實的計算。
- $\varphi_{\textbf{accept}}$**(會接受)**: 表中某時刻出現接受狀態 q_{accept}, 即 $\bigvee_t S_{q_{\text{accept}},t}$。

6.4 第四步: 結論

依構造, 任何讓 $F_{M,w}$ 為真的指派, 恰好對應 M 的一條接受計算; 反之亦然。於是

$$w \in X \iff F_{M,w} \text{ 可滿足.}$$

由於 $F_{M,w}$ 的大小為多項式、且能由 M 與 w 在多項式時間內機械地寫出, 這是一個合法的多項式歸約:

$$X \le_p \textsc{Sat} \qquad \text{對所有 } X \in \mathbf{NP}.$$

因此 SAT 是 NP-hard; 再加上 SAT$\in \mathbf{NP}$(第二週), 得 SAT 是 **NP-complete**。 ■

Cook–Levin 的歷史與意義

此定理由 **Stephen Cook**(1971) 提出、**Leonid Levin**(1973) 獨立得到, 是複雜度理論的奠基石。它的威力在於: 一旦有了第一個 NP-complete 問題 (SAT), 要證明別的問題 NP-complete 就只需「從 SAT(或其他已知 NP-complete 問題) 歸約過去」, 不必再重做這套繁瑣的編碼。Karp 在 1972 年就用此法一口氣證出 21 個經典問題 NP-complete。

變數約 $O(p(n)^2)$ 個、子句約 $O(p(n)^3)$ 個, 故公式大小是 n 的多項式——「多項式」正是這整個論證的命脈。

7 本週重點整理

主題	重點
數學歸納法	基底 $P(1)$ + 由 $P(k)$ 推 $P(k+1)$; 驗證封閉式解的標準工具
強歸納法	假設「所有 $m \le k$」都成立; 用於多前項或 $T(n/2)$ 型遞迴 (費氏、分治)
分治法	分 $\to$ 治 $\to$ 合; 成本 $T(n) = aT(n/b) + f(n)$
河內塔	$T(n) = 2T(n-1) + 1 = 2^n - 1 \in \Theta(2^n)$(指數)
合併排序	$T(n) = 2T(n/2) + n \in \Theta(n \log n)$
主定理	$k = \log_b a$; 比 $f(n)$ 與 n^k: 小 $\to \Theta(n^k)$、等 $\to \Theta(n^k \log n)$、大 $\to \Theta(f(n))$
Cook–Levin	把 NDTM 計算表編碼成布林公式 $\Rightarrow$ 每個 **NP** 問題 $\le_p$ SAT;SAT 為 NP-complete

8 練習題 (附解答)

練習 1. 用歸納法證明 $\displaystyle\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$。

解答

基底 $n=1$: 左 $=1$, 右 $=\frac{1\cdot 2\cdot 3}{6}=1$ ✓。歸納: 設對 k 成立, 則 $\sum_{i=1}^{k+1} i^2 = \frac{k(k+1)(2k+1)}{6} + (k+1)^2 = \frac{(k+1)[k(2k+1)+6(k+1)]}{6} = \frac{(k+1)(k+2)(2k+3)}{6}$, 正是 $n=k+1$ 式。

練習 2. 用主定理求解:(a) $T(n) = 4T(n/2) + n$;(b) $T(n) = 2T(n/2) + n^2$;(c) $T(n) = 3T(n/2) + n$。

解答

(a) $k = \log_2 4 = 2, f = n \in O(n^{2-\varepsilon})$, 情形 1:$\Theta(n^2)$。
(b) $k = \log_2 2 = 1, f = n^2 \in \Omega(n^{1+\varepsilon})$ 且滿足正則條件, 情形 3:$\Theta(n^2)$。
(c) $k = \log_2 3 \approx 1.585, f = n \in O(n^{k-\varepsilon})$, 情形 1:$\Theta(n^{\log_2 3})$。(此即 Karatsuba 乘法的階。)

練習 3. 二分搜尋的遞迴為 $T(n) = T(n/2) + 1$。用主定理求其階。

解答

$a = 1, b = 2 \Rightarrow k = \log_2 1 = 0$, 故 $n^k = 1$。$f(n) = 1 \in \Theta(n^0)$, 屬情形 2:$T(n) = \Theta(n^0 \log n) = \Theta(\log n)$。

練習 4. 為何主定理不能直接套用在 $T(n) = 2^n T(n/2) + n$?

解答

主定理要求 a 是**常數**。這裡「子問題個數」$a = 2^n$ 隨 n 變動, 不符前提, 故不能套用, 須另尋方法 (遞迴樹)。

練習 5 (思考題)**.** 在 Cook–Levin 證明中, 為什麼「計算表的邊長取 $T_M(n)$」是合理的? 若 M 不是多項式時間, 證明會出什麼問題?

解答

因 M 在多項式時間 $T_M(n)$ 內停機, 故至多用到 $T_M(n)$ 個時間步、且讀寫頭至多移動 $T_M(n)$ 格, 整段計算可裝進 $T_M(n) \times T_M(n)$ 的表內。若 M 只是非多項式, 表的邊長就不是 n 的多項式, 變數與公式大小將爆炸成指數, 歸約不再是「多項式時間」, 證明失效——這正是定義 **NP** 時堅持「多項式長度計算」的原因。

參考資料

1. C. Hampson, *5CCS2FC2 Foundations of Computing II, Week 3* 投影片 (induction / divide-conquer / masterthm / cooklevin),King's College London.
2. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, 3rd/4th ed., MIT Press.(第 2、4 章: 分治、遞迴、主定理)
3. M. Sipser, *Introduction to the Theory of Computation*, 3rd ed., Cengage, 2012.(第 7.4 節:Cook–Levin 定理、tableau 證明)
4. J. Kleinberg, É. Tardos, *Algorithm Design*, Pearson, 2006.(第 5 章: 分治與遞迴)
5. S. A. Cook, "The Complexity of Theorem-Proving Procedures," *STOC*, 1971;L. Levin, 1973.
6. Wikipedia: *Master theorem (analysis of algorithms)*; *Cook–Levin theorem*; *Tower of Hanoi*; *Merge sort*.