P 與 NP：百萬美元問題

完整教材 (依據 5CCS2FC2 Foundations of Computing II 第二週內容編寫)

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. 掌握**漸進符號** O, Ω, Θ 與常見成長階層。
2. 理解**時間複雜度** $T_M(n)$ 與複雜度類 **P**、**NP** 的定義, 以及兩者的關係。
3. 認識**布林可滿足性問題 (Sat)**: 真值表、DNF/CNF 正規形, 並證明 $\textsc{Sat} \in$ **NP**。
4. 理解**可計算函數**與**多項式歸約** $X \leq_p Y$, 以及 **NP-hard**、**NP-complete** 的定義與證明策略。
5. 透過兩個經典歸約 $\textsc{Sat} \leq_p \textsc{Clique}$ 與 $\textsc{Sat} \leq_p \textsc{Hamiltonian}$, 實際操作「NP-完備性證明」。
6. 理解 **P vs NP** 問題的內容、重要性與現況。

Contents

1 漸進符號 (Asymptotic Notation)

要比較演算法的效率, 我們關心的不是在某台電腦上跑幾秒, 而是**執行步數隨輸入大小 n 成長的速度**。漸進符號就是描述「成長速度」的語言。

1.1 Big-O、Big-Omega 與 Big-Theta

定義 1.1 (最終支配)**.** 設 $f, g : \mathbb{N} \to \mathbb{N}$。稱 g **最終支配 (eventually dominates)** f, 若存在常數 $c > 0$ 使得

$$\exists N \in \mathbb{N} \left(\forall n > N ;\ f(n) \le c \cdot g(n) \right).$$

即: 忽略有限多個例外與常數倍率後, f 不超過 g。

定義 1.2 (O、Ω、Θ)**.**

- **Big-O(上界)**:

$$f(n) = O(g(n)) \iff f(n) \text{ 被} g(n) \text{ 最終支配.}$$

- **Big-Omega(下界)**:

$$f(n) \in \Omega(g(n)) \iff f(n) \text{ 最終支配} g(n).$$

- **Big-Theta(緊確界)**:

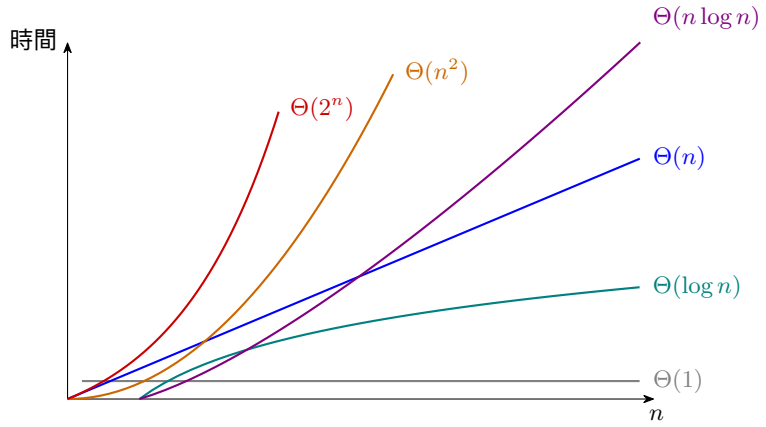
$$\Theta(g(n)) = O(g(n)) \cap \Omega(g(n)).$$

  即 $f \in \Theta(g)$ 表示 f 與 g 成長速度相同 (至多差常數倍)。

例 1.3. $f(n) = 3n^2 + 10n + 5$。取 $c = 4,\ N = 11$: 對所有 $n > 11$ 有 $10n + 5 < n^2$, 故 $f(n) \le 4n^2$, 於是 $f(n) = O(n^2)$。同時顯然 $f(n) \ge n^2$, 故 $f(n) \in \Omega(n^2)$, 合起來 $f(n) \in \Theta(n^2)$。

1.2 常見的成長階層

$$\Theta(1) \subsetneq \Theta(\log_2 n) \subsetneq \Theta(\sqrt{n}) \subsetneq \Theta(n) \subsetneq \Theta(n \log_2 n) \subsetneq \Theta(n^2) \subsetneq \Theta(n^3) \subsetneq \Theta(2^n) \subsetneq \Theta(n!) \subsetneq \cdots$$



直觀感受: 多項式 vs 指數

假設一台機器每秒執行 10^9 步, 輸入大小 $n = 60$: n^2 需 3.6×10^{-6} 秒;n^3 需 2.2×10^{-4} 秒; 但 $2^n \approx 1.15 \times 10^{18}$ 步, 需**約 36 年**;$n!$ 更是天文數字。這就是為什麼複雜度理論把「多項式時間」當作**可行 (tractable)** 的分界線: 多項式可以慢, 但指數注定爆炸。

2 判定問題與語言

延續第一週的觀點:

定義 2.1 (判定問題 = 語言)**.** 每個答案為 YES/NO(或 True/False) 的**判定問題** P, 都可以詮釋為一個語言

$$L_P = \{\, w \in \Sigma^* : w \text{ 是問題} P \text{ 中答案為 YES 的實例} \,\}$$

(對某個合適的字母表 Σ, 用來編碼問題的輸入)。

例 2.2.「圖 G 是否含有大小 k 的 clique?」對應語言 $L_{\text{CLIQUE}} = \{\, \langle G, k\rangle : G \text{ 含有大小} k \text{ 的 clique} \,\}$, 其中 $\langle G, k\rangle$ 表示把圖與整數編碼成字串。於是「解決問題」=「判定語言」, 可以直接沿用圖靈機的理論。

3 複雜度類 P 與 NP

3.1 時間複雜度

定義 3.1 (執行時間與最壞情況時間複雜度)**.** 給定一台 (會停機的) 機器 M, 定義 $t_M : \Sigma^* \to \mathbb{N}$:

$$t_M(w) := \big\{\ \text{在輸入} w \in \Sigma^* \text{ 上停機所需的步數}\ \big\},$$

並定義**最壞情況時間複雜度 (worst-case time complexity)** $T_M : \mathbb{N} \to \mathbb{N}$:

$$T_M(n) := \max\{\, t_M(w) : \text{所有長度為} n \text{ 的字} w \,\}.$$

3.2 P: 確定性多項式時間

定義 3.2 (類 P)**.** 語言 L **可在多項式時間內求解**, 若存在**確定性**圖靈機 M 使得:

(i) M 接受 (判定)L;

(ii) $T_M(n) \in O(n^k)$, 即 T_M 被某個多項式函數最終支配。

$$\mathbf{P} = \{\, \text{所有可在多項式時間內判定的問題} \,\}.$$

直觀上,**P** 是「可以**有效率地求解**」的問題: 排序、最短路徑、二分圖配對、線性規劃、判定質數 (AKS, 2002) 等都屬於 **P**。

3.3 NP: 非確定性多項式時間

定義 3.3 (類 NP)**.** 語言 L **可在非確定性多項式時間內求解**, 若存在**非確定性**圖靈機 M 使得:

(i) M 接受 L;

(ii) $T_M(n) \in O(n^k)$(每條計算分支的長度都被多項式支配)。

$$\mathbf{NP} = \{\, \text{所有可在非確定性多項式時間內判定的問題} \,\}.$$

回憶第一週:NDTM 的計算是一棵**分支樹**, 只要存在一條接受分支即接受。所以 NP 問題的典型樣貌是:

NP 的「猜測—驗證」模式 (guess and check)

- **猜測 (非確定性)**: 機器「分身」成指數多條分支, 每條分支猜一個候選解 (一列真值表、一個頂點子集、一個排列……)。
- **驗證 (多項式)**: 每條分支只需多項式時間, 即可檢查自己手上的候選解是否正確。

等價說法 (**驗證者定義**, 補充):$L \in \mathbf{NP} \iff$ 存在多項式時間的**驗證者** V 與多項式 p, 使得

$$w \in L \iff \exists\, c,\ |c| \le p(|w|),\ V(w,c) = 1,$$

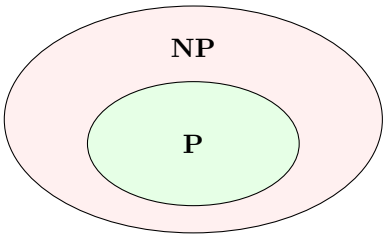
其中 c 稱為**證書 (certificate)** 或**見證 (witness)**。NP=「答案為 YES 時, 存在一個短且可快速檢查的證據」。

定理 3.4. ***P*** $\subseteq$ ***NP***。

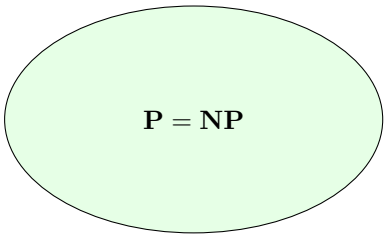
Proof. 確定性圖靈機是非確定性圖靈機的特例 (δ 的每個值都是單元素集合), 故多項式時間的 DTM 本身就是多項式時間的 NDTM。 □

3.4 兩種可能的世界

反方向「$\mathbf{NP} \subseteq \mathbf{P}$?」則是著名的未解問題。世界只有兩種可能:



世界一:$\mathbf{P} \subsetneq \mathbf{NP}$



世界二:$\mathbf{P} = \mathbf{NP}$

P vs NP: 百萬美元問題

「$\mathbf{P}$ 是否等於 $\mathbf{NP}$?」由 Stephen Cook(1971) 與 Leonid Levin(1973) 分別精確提出, 是 **Clay 數學研究所七大千禧年大獎難題**之一, 懸賞 100 萬美元。它問的是:

「凡是答案能被快速**驗證**的問題, 是否必能被快速**求解**?」

多數專家相信 $\mathbf{P} \neq \mathbf{NP}$, 但 50 多年來無人能證明。若 $\mathbf{P} = \mathbf{NP}$, 後果將天翻地覆: 現行密碼學 (RSA 等) 瓦解、最佳化問題全面變容易、甚至「尋找數學證明」本身也能自動化。

4 布林可滿足性問題 (SAT)

4.1 問題定義

布林可滿足性問題 Sat

輸入) 一個命題邏輯公式 F
輸出) True 若且唯若 F 是**可滿足的 (satisfiable)**, 即存在一組真值指派使 F 為真。

4.2 真值表與可滿足性

基本聯結詞的真值表:

P	$\neg P$
True	False
False	True

P	Q	$P \wedge Q$
True	True	True
True	False	False
False	True	False
False	False	False

P	Q	$P \vee Q$
True	True	True
True	False	True
False	True	True
False	False	False

P	Q	$P \to Q$
True	True	True
True	False	False
False	True	True
False	False	True

例 4.1 (可滿足的公式)**.** 公式 $F = (P \vee \neg R) \to \neg(\neg Q \vee R)$ 的完整真值表:

列	P	Q	R	$(P \vee \neg R) \to \neg(\neg Q \vee R)$
1.	True	True	True	False
2.	True	True	False	True
3.	True	False	True	False
4.	True	False	False	False
5.	False	True	True	True
6.	False	True	False	True
7.	False	False	True	True
8.	False	False	False	False

有四列為真 (第 2、5、6、7 列), 故 F **可滿足**。

例 4.2 (不可滿足的公式)**.** $G = \neg(P \to Q) \wedge \neg(P \vee \neg R)$: 前半要求 P 為真, 後半要求 P 為假——八列真值表全為 False,G **不可滿足** (矛盾式)。

4.3 正規形:DNF 與 CNF

定義 4.3 (文字、子句與正規形)**.**

- **文字 (literal)**: 命題變數或其否定, 如 P, $\neg Q$。
- **析取正規形 (DNF)**:「合取塊的析取」$\big(L_1 \wedge \cdots\big) \vee \big(L_1' \wedge \cdots\big) \vee \cdots$。
- **合取正規形 (CNF)**:「子句 (析取塊) 的合取」$\big(L_1 \vee \cdots\big) \wedge \big(L_1' \vee \cdots\big) \wedge \cdots$, 每個 $\big(L_1 \vee L_2 \vee \cdots\big)$ 稱為一個**子句 (clause)**。

定理 4.4. 每個命題公式都等價於某個 *DNF* 公式*;* 也都等價於某個 *CNF* 公式。

證明 *(*改寫演算法*).* 四個步驟, 反覆套用直到形式正確:

步驟 1) 改寫所有蘊含:

$$(F \to G) \;\rightsquigarrow\; (\neg F \vee G).$$

步驟 2) 用 **De Morgan 律**把否定推進括號內, 直到只貼著變數:

$$\neg(F \wedge G) \;\rightsquigarrow\; (\neg F \vee \neg G), \qquad \neg(F \vee G) \;\rightsquigarrow\; (\neg F \wedge \neg G).$$

步驟 3) 用**消去律**刪除雙重否定:

$$\neg\neg F \;\rightsquigarrow\; F.$$

步驟 4) 用**分配律**整理形狀:

- 求 **DNF**: 把析取「浮出表面」

$$F \wedge (G \vee H) \;\rightsquigarrow\; (F \wedge G) \vee (F \wedge H), \qquad (G \vee H) \wedge F \;\rightsquigarrow\; (G \wedge F) \vee (H \wedge F);$$

- 求 **CNF**: 把合取「浮出表面」

$$F \vee (G \wedge H) \;\rightsquigarrow\; (F \vee G) \wedge (F \vee H), \qquad (G \wedge H) \vee F \;\rightsquigarrow\; (G \vee F) \wedge (H \vee F).$$

每步都保持邏輯等價, 且程序必定終止。 □

從真值表直接讀出 DNF 與 CNF 以例 4.1 的 F 為例:

- **DNF—逐列「收錄」為真的列**: 每個為真的列寫成一個合取塊, 描述「恰好是這列」:

$$\underbrace{(P \wedge Q \wedge \neg R)}_{\text{第 2 列}} \vee \underbrace{(\neg P \wedge Q \wedge R)}_{\text{第 5 列}} \vee \underbrace{(\neg P \wedge Q \wedge \neg R)}_{\text{第 6 列}} \vee \underbrace{(\neg P \wedge \neg Q \wedge R)}_{\text{第 7 列}}.$$

- **CNF—逐列「排除」為假的列**: 每個為假的列寫成一個子句, 內容是該列指派的逐項否定 (「避開這列」):

$$\underbrace{(\neg P \vee \neg Q \vee \neg R)}_{\text{避開第 1 列}} \wedge \underbrace{(\neg P \vee Q \vee \neg R)}_{\text{避開第 3 列}} \wedge \underbrace{(\neg P \vee Q \vee R)}_{\text{避開第 4 列}} \wedge \underbrace{(P \vee Q \vee R)}_{\text{避開第 8 列}}.$$

備註 4.5 (正規形的代價 (補充)). 真值表法與分配律都可能讓公式**指數膨脹** (n 個變數 $\Rightarrow 2^n$ 列)。實務上 (如 SAT solver) 使用 Tseitin 轉換: 引入新變數, 可在多項式時間得到等可滿足 (不必等價) 的 CNF。因此通常直接假設 SAT 的輸入已是 CNF:

Sat(CNF 版本)

輸入) 一個 CNF 形式的命題公式 F
輸出) True 若且唯若 F 可滿足。

4.4 SAT 屬於 NP(上界)

定理 4.6. $\textsc{Sat} \in \boldsymbol{NP}$。

Proof. **輸入)** 給定 CNF 公式 F(設有 n 個變數)。

步驟 1) 確定性的做法: 計算整張真值表即可判定可滿足性。但真值表有 2^n 列——**不是多項式**!

步驟 2) 然而, 非確定性機器可以**平行地**評估每一列:NDTM 先非確定性地「猜」一組真值指派 (每個變數分支成 True/False 兩條), 然後在自己那條分支上評估 F。評估一列只需檢查至多 n 個布林運算子——**多項式**!

只要 F 可滿足, 就存在一條分支猜中滿足指派並接受; 反之所有分支都拒絕。故 $\textsc{Sat} \in \mathbf{NP}$。 □

備註 4.7 (用證書的語言重說一次). 證書 $c =$ 一組滿足指派 (長度 n); 驗證 $=$ 代入 F 求值 (多項式時間)。「短證書 + 快驗證」正是 NP 的標誌。

5 可計算函數與多項式歸約

5.1 可計算函數

定義 5.1 (可計算函數)**.** 函數 $f : \Sigma^* \to \Sigma^*$ 是**可計算的 (computable)**, 若存在 (確定性) 圖靈機 M_f 使得:

(i) M_f 接受所有輸入字 $w \in \Sigma^*$;

(ii) 從組態 $(q_{\text{init}}, \varepsilon, w)$ 出發, 機器最終停在組態 $(q_{\text{accept}}, \varepsilon, f(w))$。

換句話說: 機器把磁帶內容從 w **改寫成** $f(w)$, 並把讀寫頭歸位到字首。

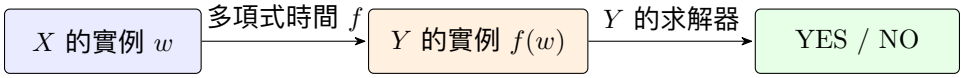
若 M_f 在多項式時間內停機, 則稱 f **可在多項式時間內計算**。

5.2 多項式歸約

定義 5.2 (多項式歸約)**.** 從問題 X 到問題 Y 的**多項式歸約 (polynomial reduction)** 是一個可在多項式時間內計算的函數 $f : \Sigma^* \to \Sigma^*$, 使得

$$w \in X \iff f(w) \in Y.$$

記作 $X \le_p Y$, 讀作「X 可多項式歸約到 Y」。



答案被**原封不動**地保留:$w \in X \iff f(w) \in Y$

直觀:$X \le_p Y$ 表示「會解 Y 就會解 X」——把 X 的輸入翻譯成 Y 的輸入即可。因此 Y 至少和 X 一樣難。

引理 5.3 (歸約的基本性質)**.** *(a)* **傳遞性***:* 若 $X \le_p Y$ 且 $Y \le_p Z$, 則 $X \le_p Z$*(*多項式合成仍是多項式*)*。

(b) **向上封閉***:* 若 $X \le_p Y$ 且 $Y \in \boldsymbol{P}$, 則 $X \in \boldsymbol{P}$; 同理若 $Y \in \boldsymbol{NP}$ 則 $X \in \boldsymbol{NP}$。

5.3 NP-hard 與 NP-complete

定義 5.4 (NP-hard)**.** 問題 X 是 **NP-hard(NP-困難)** 的, 若**每個** NP 問題都可多項式歸約到它:

$$Y \le_p X \quad \text{對所有} Y \in \mathbf{NP}.$$

(X 至少和每一個 NP 問題一樣難。)

定理 5.5 (NP-hardness 的傳遞)**.** 若 Y 是 *NP-hard* 且 $Y \leq_p X$, 則 X 也是 *NP-hard*。

Proof. **步驟 1)** 假設 (i) Y 是 NP-hard,(ii) $Y \leq_p X$。
步驟 2) 由 (i), 對所有 $Z \in \mathbf{NP}$ 有 $Z \leq_p Y$。
步驟 3) 由 (ii) 與傳遞性 (引理 5.3a):

$$Z \leq_p Y \leq_p X \quad \text{對所有} Z \in \mathbf{NP},$$

故 X 是 NP-hard。 □

證明 NP-hardness 的標準策略

不必對「每個」NP 問題各證一次! 只要找**一個**已知的 NP-hard 問題 Y, 證明 $Y \leq_p X$ 即可。最典型的選擇是 SAT:

$$\boxed{\text{SAT} \leq_p X \implies X \text{ 是 NP-hard}}$$

注意**方向**: 是把已知困難的問題歸約到 X(困難 $\to$ 未知), 而不是反過來。

定義 5.6 (NP-complete)**.** 問題 X 是 **NP-complete(NP-完備)** 的, 若

(i) X 是 NP-hard(**下界**: 至少跟所有 NP 問題一樣難);

(ii) $X \in \mathbf{NP}$(**上界**: 沒有超出 NP 的難度)。

NP-complete 問題是 NP 中「最難的問題」: 任何一個若能在多項式時間求解, 則 $\mathbf{P} = \mathbf{NP}$。

5.4 Cook–Levin 定理與 NP-complete 問題大家族

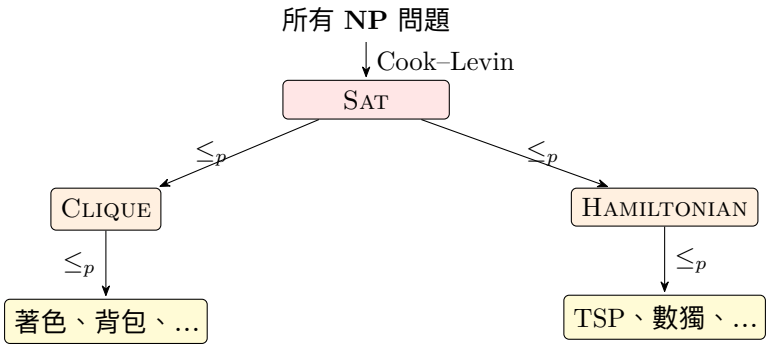
定理 5.7 (Cook–Levin 定理,1971/1973)**.** *SAT* 是 *NP-complete*。

證明. 上界已證 (定理 4.6)。下界 (每個 NP 問題 $\leq_p$ SAT) 的想法: 把任意多項式時間 NDTM 的計算表「編碼成布林公式」——詳細證明見下一週。 Q.E.D.(soon) □

有了第一個 NP-complete 問題, 即可用歸約鏈擴散出整個家族:

- 布林可滿足性問題 (SAT)、3-SAT
- Clique 問題 (CLIQUE)
- Hamiltonian 迴圈問題 (HAMILTONIAN)
- 旅行推銷員問題 (TSP)
- 圖著色問題 (Graph Colouring)
- 背包問題 (Knapsack)
- 許多遊戲與謎題:$n \times n$ 數獨、Lemmings、Pokémon、踩地雷……

(完整清單見 Wikipedia: *List of NP-complete problems*;Karp 在 1972 年一口氣證明了 21 個。)



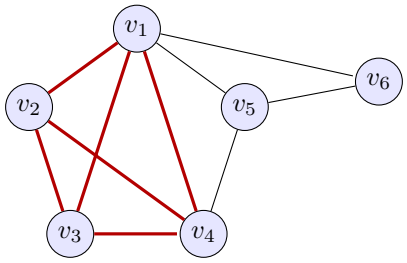
6 Clique 問題: 第一個歸約實戰

6.1 問題定義

Clique 問題 Clique

輸入) 一個無向圖 $G = (V, E)$, 以及整數 $k > 2$
輸出) True 若且唯若 G 含有大小 k 的 **clique**(完全子圖:k 個頂點兩兩相鄰)。

例 6.1. 下圖中 $\{v_1, v_2, v_3, v_4\}$ 構成大小 4 的 clique(紅色邊):



6.2 CLIQUE 屬於 NP(上界)

定理 6.2. *CLIQUE* $\in$ ***NP***。

Proof. **輸入)** 無向圖 $G = (V, E)(n = |V|)$ 與整數 $k > 2$。

步驟 1) 確定性的做法: 逐一檢查每個大小 k 的頂點子集。但子集共有 $\binom{n}{k}$ 個——**不是多項式**!

步驟 2) 然而, 非確定性機器可以**平行地**檢查每個子集: 每條分支猜一個大小 k 的子集, 然後只需檢查 $\binom{k}{2} \leq k^2$ 條邊是否都存在——**多項式**! □

6.3 SAT 可歸約到 CLIQUE(下界)

定理 6.3. *SAT* $\leq_p$ *CLIQUE*。

Proof. **輸入)** 給定 CNF 公式 F, 設其有 k 個子句。

步驟 1) 目標: 構造圖 $G_F = (V, E)$ 並選整數 k, 使得

$$F \text{ 可滿足} \iff G_F \text{ 含有大小}k \text{ 的 clique.}$$

步驟 2) 頂點: 子句 i 中每個文字 L, 都建立一個頂點 L^i(同一文字出現在不同子句要分開計)。
邊:

$$(L_1^i,\ L_2^j) \in E \iff i \neq j \text{ 且 } L_1 \neq \neg L_2.$$

即: **不同子句**的兩個文字之間連邊, **除非**它們互相矛盾 (P 對 $\neg P$)。同一子句內部一律不連邊。

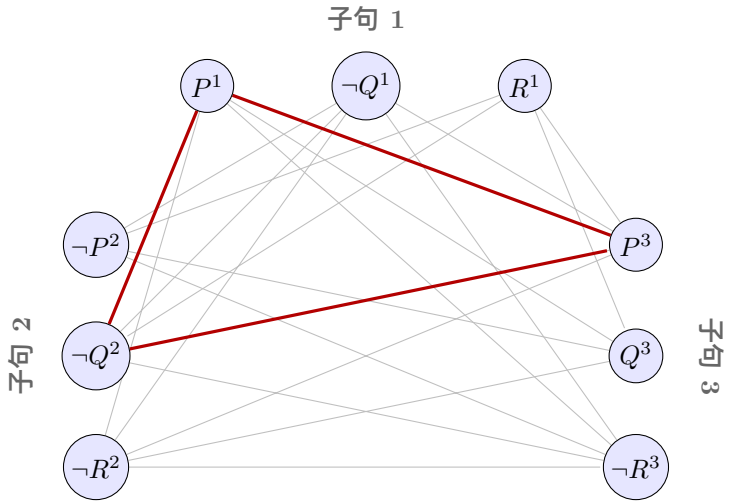
步驟 3) 取 $k = F$ 的子句數。

步驟 4) 驗證兩個方向:

- ($\Rightarrow$) 若 F 可滿足, 取一組滿足指派; 每個子句至少有一個為真的文字, 各挑一個, 得到 k 個頂點。它們來自不同子句, 且同時為真的文字不可能互相矛盾, 故兩兩相鄰——大小 k 的 clique。
- ($\Leftarrow$) 若 G_F 有大小 k 的 clique: 同一子句內無邊, 故 k 個頂點必然**每個子句恰一個**; 又 clique 中無矛盾對, 故「讓這些文字全為真」是一致的指派 (其餘變數任意), 它滿足每個子句——F 可滿足。

構造顯然可在多項式時間完成, 故 SAT $\leq_p$ CLIQUE。 □

例 6.4 (投影片例子)**.** $F = (P \vee \neg Q \vee R) \wedge (\neg P \vee \neg Q \vee \neg R) \wedge (P \vee Q \vee \neg R)$,$k = 3$。頂點分三組; 下圖灰邊為 E 中所有邊, 紅色粗邊標出一個大小 3 的 clique $\{P^1,\ \neg Q^2,\ P^3\}$, 對應滿足指派 $P = \text{True},\ Q = \text{False}$($R$ 任意):



注意 P^1 與 $\neg P^2$ 之間**沒有**邊 (矛盾對),R^1 與 $\neg R^2$、$\neg Q^1$ 與 Q^3 之間亦然。

推論 6.5. *CLIQUE 是 NP-complete。*

Proof. 上界: 定理 6.2(CLIQUE $\in$ **NP**)。下界:SAT 是 NP-hard(Cook–Levin) 且 SAT $\leq_p$ CLIQUE(定理 6.3), 由定理 5.5,CLIQUE 是 NP-hard。 □

7 Hamiltonian 迴圈問題: 第二個歸約實戰

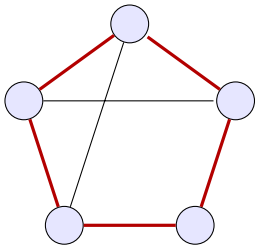
7.1 問題定義

Hamiltonian 迴圈問題 Hamiltonian

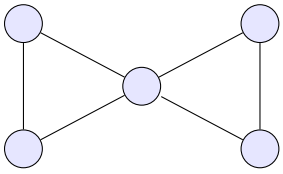
輸入) 一個無向圖 $G = (V, E)$

輸出) True 若且唯若 G 含有 **Hamiltonian 迴圈**: 一條**恰好經過每個頂點一次**並回到起點的迴圈。

例 7.1. 左圖有 Hamiltonian 迴圈 (紅色); 右圖則沒有 (中心點是割點, 任何迴圈都得經過它兩次):



✓ 有 Hamiltonian 迴圈



✗ 無 Hamiltonian 迴圈

7.2 HAMILTONIAN 屬於 NP(上界)

定理 7.2. *HAMILTONIAN* $\in$ ***NP***。

Proof. **輸入)** 無向圖 $G = (V, E)$,$n = |V|$。

步驟 1) 確定性的做法: 逐一檢查頂點的每種排列是否構成迴圈。但排列共有 $n!$ 種——**不是多項式**!

步驟 2) 然而,NDTM 可以**平行地**檢查每個排列: 每條分支猜一個排列, 然後只需檢查 n 條邊 (相鄰頂點是否相連、首尾是否相連)——**多項式**! □

7.3 SAT 可歸約到 HAMILTONIAN(下界)

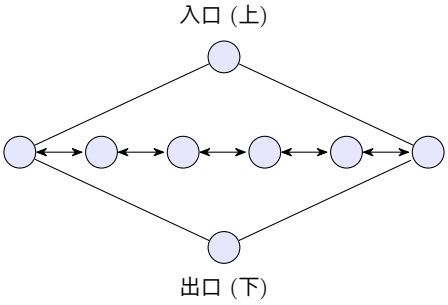
定理 7.3. *SAT* $\leq_p$ *HAMILTONIAN*。

證明 *(構造概要)*. **輸入)** 給定 CNF 公式 F(變數 $x_1, \ldots, x_n$, 子句 $C_1, \ldots, C_k$)。

步驟 1) 目標: 構造圖 G_F 使得

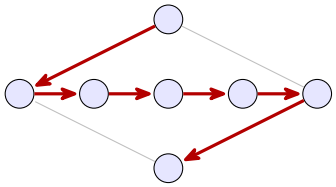
$$F \text{ 可滿足} \iff G_F \text{ 有 Hamiltonian 迴圈.}$$

步驟 2) 變數 gadget: 每個變數 x_i 配一個「菱形」結構——上下兩個端點, 中間夾一條**水平節點列** (相鄰節點間有雙向邊):

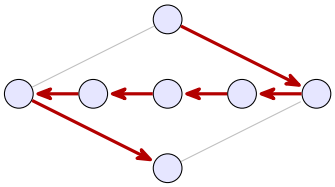


步驟 3) 關鍵觀察:Hamiltonian 路徑要吃掉整條水平列, 通過這個 gadget **只有兩種走法**:

$$\underbrace{\text{由左到右}}_{\text{解讀為}x_i=\text{True}} \quad \text{或} \quad \underbrace{\text{由右到左}}_{\text{解讀為}x_i=\text{False}}$$



由左到右:x_i = True

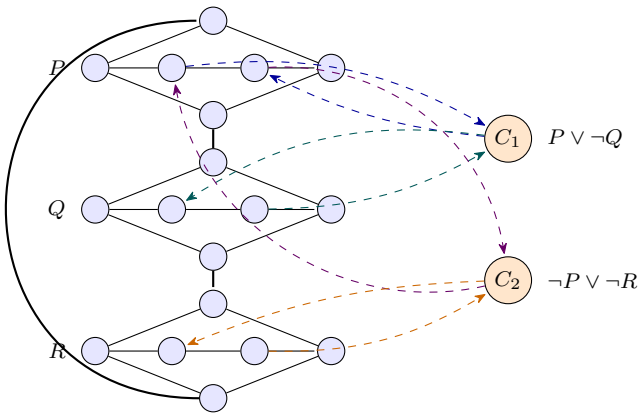


由右到左:x_i = False

步驟 4) 把 n 個菱形**上下串接**成一個大環 (上一個的出口接下一個的入口, 最後一個接回第一個)。此時圖恰有 2^n 條 Hamiltonian 迴圈, 與 2^n 組真值指派一一對應。

接著為每個子句 C_j 增設**一個子句節點**: 在水平列上為每個子句保留一對相鄰節點; 若文字 x_i 出現在 C_j, 就從 x_i 那列的第 j 對節點, 沿「左到右」方向接出一條到 C_j 再接回來的**繞道 (detour)**; 若出現的是 $\neg x_i$, 則沿「右到左」方向接繞道。

以 $F = (P \vee \neg Q) \wedge (\neg P \vee \neg R)$ 為例的整體結構 (示意; 繞道以虛線表示):



步驟 5) 驗證兩個方向:

- ($\Rightarrow$) 若 F 有滿足指派: 每個變數依其真值決定該列方向; 每個子句 C_j 至少有一個文字為真, 挑其中一個, 其所在列的行進方向恰好「順向」, 可順路繞經 C_j 一次再回來。所有節點 (含子句節點) 恰被經過一次——Hamiltonian 迴圈。
- ($\Leftarrow$) 若 G_F 有 Hamiltonian 迴圈: 可證迴圈必須「規規矩矩」地逐個菱形走 (進入子句節點後必須立刻回到同一列的相鄰位置, 否則必有節點無法被覆蓋)。於是每列有明確方向, 讀出指派: 左到右 = True、右到左 = False。迴圈造訪了每個 C_j, 表示每個子句至少有一個文字順向——即至少一個文字為真,F 被滿足。

整個構造的大小是 $O(nk)$, 可在多項式時間完成, 故 SAT $\leq_p$ HAMILTONIAN。 □

推論 7.4. *HAMILTONIAN 是 NP-complete。*

Proof. 上界: 定理 7.2。下界:SAT $\leq_p$ HAMILTONIAN(定理 7.3) 加上定理 5.5。 □

備註 7.5 (gadget 證明的一般心法 (補充)). 歸約構造常用 **gadget(小零件)** 思維:

- **變數 gadget**: 提供「二選一」的結構, 編碼 True/False;
- **子句 gadget**: 設計成「至少要被滿足一次才走得通」的關卡;
- **連接方式**: 確保整體解恰好對應一組一致的指派。

CLIQUE 歸約中「每子句挑一個頂點、矛盾不相鄰」與 HAMILTONIAN 歸約中「方向 = 真值、繞道 = 滿足子句」都是這個模式的化身。

8 本週重點整理

概念	內容
$O/\Omega/\Theta$	上界/下界/緊確界;「最終支配」= 忽略常數與有限例外
P	確定性 TM、多項式時間**可求解**
NP	非確定性 TM、多項式時間; 等價地: 解**可在多項式時間驗證**
$X \le_p Y$	多項式時間可計算的 f 使 $w \in X \iff f(w) \in Y$;「會解 Y 就會解 X」
NP-hard	所有 **NP** 問題 $\le_p$ 它 (下界); 證法:$\textsc{Sat} \le_p X$
NP-complete	NP-hard $\wedge$ $\in$ **NP**(下界 + 上界);NP 中最難的問題
Cook–Levin	$\textsc{Sat}$ 是 NP-complete(**第一塊骨牌**)

- NP-membership 的固定套路:「確定性枚舉爆炸 (2^n、$\binom{n}{k}$、$n!$)→ NDTM 平行猜測 + 多項式驗證」。
- NP-hardness 的固定套路:「從已知 NP-hard 問題 (通常 $\textsc{Sat}$) 歸約過來」, 注意方向。
- $\mathbf{P} \subseteq \mathbf{NP}$;$\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ 是百萬美元未解問題。任一 NP-complete 問題若屬於 **P**, 則 $\mathbf{P} = \mathbf{NP}$。

9 練習題 (附提示與解答)

練習 1. 證明 $f(n) = 5n^3 + n\log_2 n + 7$ 滿足 $f(n) \in \Theta(n^3)$。

解答

上界: 對 $n > 2$,$n\log_2 n < n^2 < n^3$ 且 $7 < n^3$, 故 $f(n) \le 7n^3$, 即 $f = O(n^3)$(取 $c = 7, N = 2$)。
下界: 顯然 $f(n) \ge 5n^3 \ge n^3$, 故 $f \in \Omega(n^3)$。兩者合併得 $f \in \Theta(n^3)$。

練習 2. 把 $F = (P \to Q) \to R$ 分別改寫成 CNF 與 DNF。

解答

步驟 1(消蘊含):$\neg(\neg P \vee Q) \vee R$。步驟 2–3(De Morgan、消雙重否定):$(P \wedge \neg Q) \vee R$——這已是 **DNF**。步驟 4(分配律求 CNF):$(P \vee R) \wedge (\neg Q \vee R)$——**CNF**。

練習 3. 判斷 $G = (P \vee Q) \wedge (\neg P \vee Q) \wedge (P \vee \neg Q) \wedge (\neg P \vee \neg Q)$ 是否可滿足, 並說明理由。

解答

不可滿足。四個子句分別「排除」了 $(\text{False}, \text{False}), (\text{True}, \text{False}), (\text{False}, \text{True}), (\text{True}, \text{True})$ 四種指派 (每個子句恰好擋掉一列),2^2 種指派全被排除。

練習 4. 對 $F = (P \vee Q) \wedge (\neg P \vee \neg Q)$ 執行 $\textsc{Sat} \le_p \textsc{Clique}$ 的構造: 畫出 G_F、寫出 k, 並找出一個大小 k 的 clique 與對應的滿足指派。

解答

$k = 2$。頂點:P^1, Q^1(子句 1)、$\neg P^2, \neg Q^2$(子句 2)。邊:P^1–$\neg Q^2$、Q^1–$\neg P^2$(排除矛盾對 P^1–$\neg P^2$、Q^1–$\neg Q^2$)。clique $\{P^1, \neg Q^2\}$ 對應 $P = \text{True}, Q = \text{False}$ ✓;clique $\{Q^1, \neg P^2\}$ 對應 $P = \text{False}, Q = \text{True}$ ✓。

練習 5. 獨立集問題 Independent-Set: 給定 (G, k), 問 G 是否有 k 個頂點**兩兩不相鄰**。證明 Clique $\leq_p$ Independent-Set。

解答

取補圖:$f(\langle G, k\rangle) = \langle \overline{G}, k\rangle$, 其中 $\overline{G}$ 與 G 頂點相同, 邊恰好互補。S 在 G 中是 clique $\iff$ S 在 $\overline{G}$ 中是獨立集, 故 $\langle G, k\rangle \in$ Clique $\iff$ $\langle \overline{G}, k\rangle \in$ Independent-Set; 補圖可在 $O(n^2)$ 時間建好。(由 Clique 的 NP-hardness,Independent-Set 也是 NP-hard; 它顯然在 NP, 故 NP-complete。)

練習 6 (思考題)**.** 同學甲宣稱:「我把 X 歸約到了 Sat($X \leq_p$ Sat), 而 Sat 是 NP-hard, 所以 X 是 NP-hard。」這個論證錯在哪裡?

解答

方向反了。$X \leq_p$ Sat 只說明「X 不比 Sat 難」(這對每個 NP 問題都成立, 毫無新資訊)。要證 X 是 NP-hard, 必須把困難問題歸約到 X:Sat $\leq_p X$。記法:$\leq_p$ 的箭頭指向「更難 (或一樣難)」的那一方。

練習 7 (思考題)**.** 若有人明天證明了 Clique $\in \mathbf{P}$, 會發生什麼事? 請用本週的定理推導。

解答

Clique 是 NP-hard, 故所有 $Y \in \mathbf{NP}$ 滿足 $Y \leq_p$ Clique。由引理 (向上封閉):$Y \leq_p$ Clique 且 Clique $\in \mathbf{P} \Rightarrow Y \in \mathbf{P}$。於是 $\mathbf{NP} \subseteq \mathbf{P}$, 加上 $\mathbf{P} \subseteq \mathbf{NP}$, 得 $\mathbf{P} = \mathbf{NP}$——百萬美元到手, 順便讓全世界的密碼學家失業。

參考資料

1. C. Hampson, *5CCS2FC2 Foundations of Computing II, Week 2* 投影片 (pvsnp / sat / np-complete / clique / hampath),King's College London.
2. M. Sipser, *Introduction to the Theory of Computation*, 3rd ed., Cengage, 2012.(第 7 章: 時間複雜度、NP-完備性、HamPath 歸約)
3. S. Arora, B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, 2009.(NP 的證書定義、Cook–Levin)
4. J. Kleinberg, É. Tardos, *Algorithm Design*, Pearson, 2006.(第 8 章: 多項式歸約、3-SAT$\leq_p$Hamiltonian Cycle)
5. S. A. Cook, "The Complexity of Theorem-Proving Procedures," *STOC*, 1971;L. Levin, 1973.
6. R. M. Karp, "Reducibility Among Combinatorial Problems," 1972.(Karp 的 21 個 NP-complete 問題)
7. Wikipedia: *P versus NP problem*; *NP (complexity)*; *List of NP-complete problems*.

8. Clay Mathematics Institute: *Millennium Prize Problems*, https://www.claymath.org/millennium-problems/.