複雜度類 NP 專題

從「驗證 vs. 求解」談起, 並詳述百萬美元問題 P vs NP

5CCS2FC2 Foundations of Computing II 補充教材
(整理自課程投影片並參考補充文獻擴充編寫)

2026 年 6 月

本講學習目標

1. 從日常直覺理解 **NP** 的核心精神:**「答案難找, 但容易檢查」**。
2. 精確掌握 **NP** 的**兩個等價定義**: 非確定性圖靈機 (NDTM) 與**驗證者—證書**。
3. 透過**大量實際例子** (數獨、SAT、子集和、地圖著色、團、旅行推銷員) 體會何謂 **NP** 問題。
4. 釐清 **P**、**NP**、**co-NP**、**NP**-complete 之間的關係。
5. **詳細**理解 **P vs NP** 這個百萬美元問題: 它在問什麼、歷史、若 **P** = **NP** 的後果、現況與常見誤解。

Contents

1 先建立直覺: 容易檢查 $\neq$ 容易找到

想像三個情境:

- **數獨**: 要你解出一個 9×9 數獨可能要抓破頭; 但如果有人把填好的答案交給你, 你只要逐行、逐列、逐宮檢查一遍, 幾分鐘就能確認對錯。
- **拼圖**: 把上千片拼圖拼好很耗時; 但要檢查一幅已拼好的圖是否完整正確, 瞄一眼就知道。
- **找質因數**: 要把 2773 分解成兩個質數的乘積得試很多次; 但若我說「$2773 = 47\times 59$」, 你只要做一次乘法就能驗證。

這三個例子有一個共同結構:

「找到答案」很難, 但「拿到答案後檢查它對不對」很容易。

複雜度類 **NP** 就是把這個直覺**數學化**的產物。它收集了所有「答案容易驗證」的判定問題。本講的主角——百萬美元問題 **P vs NP**——問的正是:

「**容易檢查**的問題, 是不是其實也**容易求解**?」

一句話記住 NP

NP = 「答案為 YES 時, 存在一個簡短的『證據』, 讓人能在多項式時間內檢查無誤。」

2 前置知識: 判定問題與多項式時間

2.1 判定問題就是語言

定義 2.1 (判定問題)**. 判定問題 (decision problem)** 是答案只有 YES/NO 的問題。把所有 YES 實例編碼成字串, 就得到一個**語言**

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

於是「解決問題」=「判定字串 w 是否屬於 L_P」。

為什麼只談 YES/NO? 因為大部分「求解」問題都能改寫成等價的判定問題。例如「圖最大的團有多大?」可改問「圖中是否有大小 $\geq k$ 的團?」——對不同的 k 問幾次即可。判定版本方便理論分析, 卻不失一般性。

2.2 多項式時間 =「可行」的分界線

回憶第一週/第二週: 機器 M 的**最壞情況時間複雜度**是

$$T_M(n) := \max\{\, t_M(w) : |w| = n \,\},$$

即所有長度 n 的輸入中, 執行步數最多的那個。

定義 2.2 (複雜度類 P)**.**

$$\mathbf{P} = \{\, \text{存在}\textbf{確定性}\text{圖靈機在 } O(n^k) \text{ 時間內判定的問題} \,\}.$$

P 直觀上是**「可以有效率地求解」**的問題。

為什麼用「多項式」當分界?

多項式 $(n,\ n^2,\ n^3, \dots)$ 增長溫和; 指數 $(2^n,\ n!)$ 增長失控。下表是每秒 10^9 步的機器, 在不同輸入大小 n 所需時間:

n	n^2	n^3	2^n	$n!$
10	$0.1\,\mu s$	$1\,\mu s$	$1\,\mu s$	$3.6\,ms$
30	$0.9\,\mu s$	$27\,\mu s$	$1.07\,s$	$\sim 8 \times 10^{15}$ 年
60	$3.6\,\mu s$	$0.2\,ms$	~ 36 年	遠超宇宙年齡
100	$10\,\mu s$	$1\,ms$	$\sim 4 \times 10^{13}$ 年	天文數字

多項式演算法即使慢, 規模翻倍頂多時間翻幾倍; 指數演算法則「多一個輸入, 時間翻一倍」,$n = 60$ 就已經算不完。這就是把多項式時間視為 **tractable(可行)** 的理由。

3 NP 的兩個等價定義

NP 有兩種看似不同、實則等價的定義。第一種沿用圖靈機, 第二種更貼近「驗證」的直覺。

3.1 定義一: 非確定性圖靈機

定義 3.1 (NP,NDTM 版本)**.**

$$\mathbf{NP} = \{\, \text{存在}\textbf{非確定性}\text{圖靈機在 } O(n^k) \text{ 時間內判定的問題} \,\}.$$

回憶非確定性圖靈機 (NDTM) 的計算是一棵**分支樹**, 面對選擇時可以「分身」探索所有可能; 只要存在一條分支到達接受狀態, 就算接受。所以一台多項式時間的 NDTM 可以理解成:

「**猜**一個可能的答案 (非確定性分支), 再花多項式時間**檢查**它。」

3.2 定義二: 驗證者與證書 (更直覺)

定義 3.2 (NP, 驗證者版本)**.** 語言 $L \in \mathbf{NP}$ 若且唯若存在一個**多項式時間**的確定性圖靈機 V(稱為**驗證者 verifier**) 與一個多項式 p, 使得對所有 w:

$$w \in L \iff \exists\, c \,\big(\ |c| \le p(|w|) \text{ 且 } V(w, c) = 1\ \big).$$

其中字串 c 稱為 w 的**證書 (certificate)** 或**見證 (witness)**。

白話翻譯這個定義的三個要件:

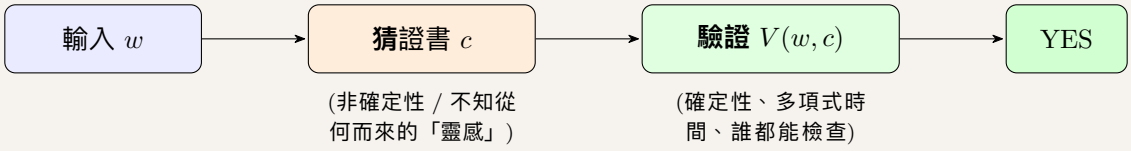
1. **證書要短**:$|c| \le p(|w|)$——證據的長度不能超過輸入的多項式倍 (不能把整本「指數長的解答」當證據)。
2. **驗證要快**:V 在多項式時間內跑完。
3. **健全且完備**:w 是 YES 實例 $\iff$ 存在至少一個證書能通過驗證; 若 w 是 NO 實例, 則任何證書都無法騙過 V。

定理 3.3 (兩定義等價)**.** 定義 *3.1* 與定義 *3.2* 定出相同的類 ***NP***。

證明概要. ($\Rightarrow$) 給定多項式時間 NDTM M, 把「每一步該選哪個分支」的選擇序列當作證書 c; 驗證者 V 依 c 確定性地模擬 M 那一條分支, 檢查是否接受。分支長度為多項式, 故 c 短、V 快。

($\Leftarrow$) 給定驗證者 V, 造一台 NDTM: 先非確定性地**猜**出證書 c 的每個符號 (分支), 再執行 $V(w, c)$。若有證書能通過, 就存在一條接受分支。 □

NP 的精神:Guess and Check(猜測—驗證)



難的是**猜** (找答案);**NP** 的要求只是「猜中之後**驗證**要快」。

4 實際例子: 這些都是 NP 問題

判斷一個問題在不在 **NP**, 標準作法是**指出它的證書與驗證程序**。以下用具體例子示範。

4.1 例一: 數獨 (Sudoku)

$n^2 \times n^2$ 數獨 (判定版)

輸入) 一個部分填好的數獨盤面
輸出) True 若且唯若能把空格補滿, 使每行、每列、每宮都是 $1 \sim n^2$ 的排列。

- **證書**: 一份填滿的完整盤面。
- **驗證**: 檢查每行、每列、每宮是否都不重複, 並與題目給定的數字相符。只需掃過所有格子常數次——多項式時間。

故 (廣義) 數獨 $\in$ **NP**。 (事實上一般 n 的數獨是 **NP**-complete。)

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

解
(指數搜尋空間)

給定一份「填好的盤面」當證書, 只要逐行逐列逐宮掃一遍即可**快速驗證**——這正是 **NP** 的標誌。

4.2 例二: 布林可滿足性 SAT

Sat

輸入) 一個命題公式 F(常為 CNF)
輸出) True 若且唯若存在一組真值指派使 F 為真。

- **證書**: 一組真值指派, 如 $P = \text{True},\ Q = \text{False},\ R = \text{True}$。
- **驗證**: 把指派代入 F 求值, 檢查是否為真。掃過所有運算子一次——多項式時間。

例 4.1. $F = (P \vee \neg Q \vee R) \wedge (\neg P \vee \neg Q \vee \neg R) \wedge (P \vee Q \vee \neg R)$。證書 $P = \text{True},\ Q = \text{False},\ R = \text{True}$: 三個子句分別由 P、$\neg Q$、P 滿足, 代入得 True, 驗證通過。✓

故 SAT $\in$ **NP**。 (由 Cook–Levin 定理,SAT 還是 **NP**-complete。)

4.3 例三: 子集和問題 (Subset-Sum)——有具體數字

Subset-Sum

輸入) 一組整數 $S = \{a_1, \ldots, a_m\}$ 與目標值 t
輸出) True 若且唯若 S 有一個子集, 其元素總和恰為 t。

例 4.2. 設 $S = \{3,\ 34,\ 4,\ 12,\ 5,\ 2\}$, 目標 $t = 9$。

- **找答案難**:$m = 6$ 個數, 子集有 $2^6 = 64$ 個; 一般 m 個數就有 2^m 個子集要試。
- **證書**: 一個子集, 例如 $\{4,\ 5\}$。
- **驗證**: 把子集元素加起來, 檢查是否 $= t$。$4 + 5 = 9$ ✓, 一次加法掃過——多項式時間。

故 SUBSET-SUM $\in$ **NP**。這個例子特別能凸顯「驗證 (一次加法)」與「求解 (指數搜尋)」的天壤之別。

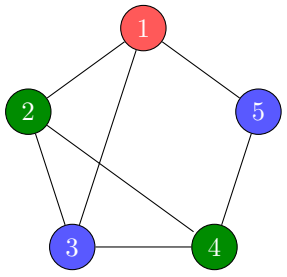
4.4 例四: 地圖三著色 (3-Colouring)

3-Colouring

輸入) 一個無向圖 $G = (V, E)$
輸出) True 若且唯若能用 3 種顏色為頂點上色, 使**相鄰頂點顏色不同**。

這就是「地圖著色」的抽象版: 國家 = 頂點, 接壤 = 邊。

- **證書**: 一份著色方案 (每個頂點配一個顏色)。
- **驗證**: 檢查每條邊的兩端顏色是否相異。掃過所有邊一次——多項式時間。



證書:{1=紅, 2=綠, 3=藍, 4=綠, 5=藍}。逐邊檢查兩端不同色 ✓——多項式驗證。

故 3-COLOURING $\in$ **NP**。 (亦為 **NP**-complete; 而 **2**-著色卻在 **P**! 差一個顏色, 難度天差地別。)

4.5 例五: 團問題與旅行推銷員 (回顧)

- **Clique**: 輸入 (G, k), 問 G 是否有 k 個兩兩相鄰的頂點。**證書** = 那 k 個頂點; **驗證** = 檢查 $\binom{k}{2}$ 條邊都在。
- **旅行推銷員 (Tsp, 判定版)**: 輸入帶權圖與預算 B, 問是否有一條經過所有城市恰一次、總長 $\leq B$ 的環路。**證書** = 一條城市排列; **驗證** = 把沿途距離加總, 檢查 $\leq B$。

兩者都 $\in$ **NP**, 也都是 **NP**-complete。

固定套路: 如何證明「問題 $\in$ NP」

1. 說清楚**證書**是什麼 (填好的盤面、一組指派、一個子集、一條路徑……), 並確認它**長度是多項式**。
2. 描述**驗證程序**, 並說明它在**多項式時間**內完成。
3. 確認:YES 實例存在通過驗證的證書;NO 實例沒有任何證書能通過。

5 P、NP、co-NP 與 NP-complete 的版圖

5.1 P 一定在 NP 之中

定理 5.1. ***P*** $\subseteq$ ***NP***。

Proof. 若 $L \in$ **P**, 則有多項式時間機器直接判定它。把它當驗證者 V, **忽略證書** c、自己算出答案即可——「不需要證據也能快速判斷」是「有證據就能快速驗證」的特例。故 $L \in$ **NP**。 □

5.2 co-NP:NO 實例容易驗證

NP 關心「YES 容易驗證」; 對稱地,**co-NP** 關心「NO 容易驗證」。

定義 5.2 (co-NP)**.** $L \in$ **co-NP** 若且唯若其補語言 $\overline{L} \in$ **NP**。即: 當答案為 NO 時, 存在簡短可驗證的「反例證書」。

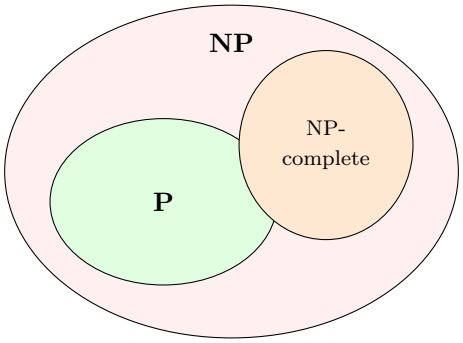
例 5.3. Tautology(「公式是否恆真?」) 屬於 **co-NP**: 若答案為 NO(不恆真), 證書就是一組讓它為假的指派, 代入即可驗證。是否 **NP** = **co-NP** 同樣是未解問題 (一般相信不相等)。

5.3 NP-complete:NP 裡最難的問題

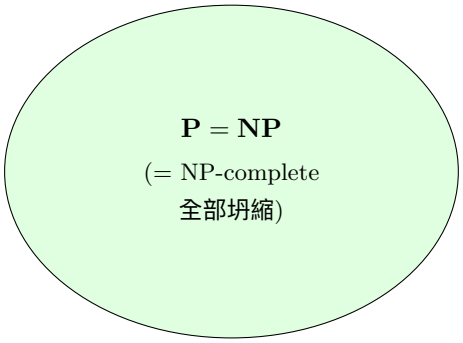
定義 5.4 (NP-complete)**.** 問題 X 是 **NP-complete** 若 (i) $X \in$ **NP**, 且 (ii) 每個 **NP** 問題都能多項式歸約到 X(即 X 是 **NP**-hard)。

NP-complete 問題是 **NP** 的「最難代表」: **只要其中任何一個落入 P, 整個 NP 就崩塌進 P**, 於是 **P** = **NP**。本講提到的 Sat、Clique、Hamiltonian、Tsp、3-Colouring、Subset-Sum、數獨……全都是 **NP**-complete。

5.4 兩種可能的世界



世界一:$\mathbf{P \neq NP}$(多數人相信)



世界二:$\mathbf{P = NP}$

在世界一,NP-complete 問題是一塊「永遠困難」的孤島, 與 **P** 不相交。在世界二, 整個版圖坍縮成一個圈, 所有「容易驗證」的問題都「容易求解」。**我們至今不知道自己活在哪個世界**——這就是下一節的百萬美元問題。

6 百萬美元問題:P vs NP

$1,000,000 —Clay 千禧年大獎難題

P vs NP 問題: 是否 $\mathbf{P = NP}$?

換句話說: 凡是答案能被電腦**快速驗證**的問題, 是否必然也能被電腦**快速求解**?

2000 年, 美國 Clay 數學研究所 (Clay Mathematics Institute) 選出七個「千禧年大獎難題」, 每題懸賞 100 萬美元。P vs NP 是其中之一, 也被廣泛認為是**理論計算機科學最重要的未解問題**。截至目前 (2026 年) 仍未解決; 七題中至今僅 Poincaré 猜想被攻克。

6.1 它到底在問什麼?

把前面建立的直覺翻譯成正式語句:

$$\mathbf{P} \subseteq \mathbf{NP} \text{ 已知為真 (定理 5.1)}, \qquad \text{問題是反方向 } \mathbf{NP} \overset{?}{\subseteq} \mathbf{P}.$$

- 若 $\mathbf{P = NP}$:「容易檢查」就蘊含「容易求解」。數獨、排課、晶片佈線、蛋白質摺疊等成千上萬的難題, 全都會有快速演算法。
- 若 $\mathbf{P \neq NP}$: 存在一些問題, 其答案**可以快速驗證, 卻注定無法快速找到**——「驗證」與「發明」之間有一道不可跨越的鴻溝。

一個迷人的哲學版本 (Scott Aaronson)

若 $\mathbf{P = NP}$, 那麼**欣賞一首交響曲與創作它一樣容易, 理解一個證明與發現它一樣容易**。因為「創作/發現」本質上是在巨大的可能性空間中搜尋一個「能通過驗證」的對象。$\mathbf{P = NP}$ 等於宣稱: 凡是能認得出好東西的人, 就能有效率地造出好東西。直覺上這太美好而不真實——這也是多數人相信 $\mathbf{P \neq NP}$ 的原因之一。

6.2 歷史脈絡

- **源頭**:1930 年代 Hilbert 的「判定問題」促成了圖靈機與可計算性理論;1950 年代 Gödel 在給 von Neumann 的信中已隱約觸及「搜尋證明能否避免窮舉」的問題。
- **1971**:Stephen Cook 在論文〈The Complexity of Theorem-Proving Procedures〉中正式提出 P vs NP, 並證明 SAT 是 **NP**-complete(Cook–Levin 定理)。
- **1973**:Leonid Levin 在蘇聯獨立得到同樣結果。
- **1972**:Richard Karp 一口氣證明 21 個經典問題皆為 **NP**-complete, 讓人們意識到這個現象**無所不在**。
- **2000**:Clay 研究所將其列為千禧年大獎難題, 懸賞百萬美元。

6.3 如果 P = NP, 世界會怎樣?

這取決於證明是**構造性**還是**非構造性**, 不可一概而論:

- **若是「構造性」且演算法夠快 (低次多項式)**——影響驚天動地:
  - **密碼學崩潰**:RSA、橢圓曲線等公鑰系統的安全性建立在「某些問題難解」之上。若能快速解 SAT, 就能破解金融加密、數位簽章; 對稱加密 (AES) 與雜湊 (支撐區塊鏈/比特幣) 也會被「反推原像」攻破。網路安全將需全面改用**資訊理論安全**的方案。
  - **最佳化全面變容易**: 物流排程、晶片設計、班表、蛋白質摺疊、疫苗設計等, 都能算出 (近) 最佳解。
  - **數學自動化**:「是否存在長度 $\leq n$ 的證明」是 **NP** 問題;**P** = **NP** 意味著電腦能在多項式時間內**自動尋找數學定理的證明** (只要證明不太長)。
- **若是「非構造性」或演算法是高次多項式 (如 n^{100})**——理論驚天動地, 實務上卻可能**毫無影響**: 我們只知道快速演算法「存在」, 卻不知如何寫出, 或它慢到不能用。

常見誤解的澄清

- ✗「**P** = **NP** 一定會讓比特幣瞬間被駭。」——只有在證明構造出低次多項式演算法時才成立; 非構造性證明不會給你可用的程式。
- ✗「NP 代表『非多項式 (Non-Polynomial)』。」——**錯**!NP 是 *Non-deterministic Polynomial*(非確定性多項式)。**P** 其實是 **NP** 的子集。
- ✗「NP 問題就是無解/不可計算。」——**NP** 問題全都**可解** (暴力枚舉證書即可), 只是可能很慢; 不可計算性 (如停機問題) 是另一回事。
- ✗「量子電腦能解所有 NP 問題, 所以 P vs NP 無所謂。」——一般相信量子電腦**無法**有效率地解 **NP**-complete 問題 (它擅長的是因數分解等特殊結構問題)。

6.4 現況與為何如此困難

- **學界共識**: 歷次調查中, 絕大多數理論計算機科學家相信 **P** $\neq$ **NP**, 但這只是「信念」, 不是證明。

- **已知的障礙**: 相對化 (relativization)、自然證明 (natural proofs)、代數化 (algebrization) 等結果顯示, 現有的證明技術**本質上不足以**解決 P vs NP——任何成功的證明必須引入全新方法。
- **它牽動整片版圖**: 解決 P vs NP 會一併釐清密碼學是否可能、最佳化的極限、隨機性與計算的關係等一大類問題。Aaronson 形容它是「我們不知如何回答的一整類『什麼對電腦可行』問題的旗艦」。

領取百萬獎金的兩條路 (任一即可)

1. **證明 $\mathbf{P} = \mathbf{NP}$**: 為某一個 **NP**-complete 問題 (如 SAT) 給出多項式時間演算法。由歸約, 所有 **NP** 問題隨之落入 **P**。
2. **證明 $\mathbf{P} \neq \mathbf{NP}$**: 嚴格證明某一個 **NP** 問題**不可能**有多項式時間演算法。

兩條路都無比艱難——這正是它值一百萬美元的原因。

7 本講重點整理

概念	內容
NP 的精神	答案為 YES 時, 存在**簡短證書**, 能**多項式時間驗證**
定義一	多項式時間**非確定性**圖靈機可判定
定義二	存在多項式時間**驗證者** V 與短證書 c:$w \in L \iff \exists c,\ V(w,c) = 1$
證明 $\in$ **NP**	指出證書 (多項式長)+ 驗證程序 (多項式時間)
$\mathbf{P} \subseteq \mathbf{NP}$	能快速求解 $\Rightarrow$ 能快速驗證 (忽略證書即可)
NP-complete	**NP** 中最難; 任一個落入 **P** 則 $\mathbf{P} = \mathbf{NP}$
P vs NP	「容易驗證」是否蘊含「容易求解」? 百萬美元未解問題

- **NP** $\neq$「非多項式」; 它是**非確定性多項式**, 而且 $\mathbf{P} \subseteq \mathbf{NP}$。
- 實例: 數獨、SAT、SUBSET-SUM、3-COLOURING、CLIQUE、TSP 都靠「短證書 + 快驗證」入列 **NP**。
- P vs NP 問的是「驗證 vs. 求解」的鴻溝; 影響涵蓋密碼學、最佳化、AI、數學自動化。多數人相信 $\mathbf{P} \neq \mathbf{NP}$, 但無人能證明。

8 練習題 (附解答)

練習 1. 為「**合數問題**」(輸入整數 N, 問 N 是否為合數) 指出一個證書與驗證程序, 說明它 $\in$ **NP**。

解答

證書: 一個非平凡因數 $d(1 < d < N)$。驗證: 檢查 $1 < d < N$ 且 $N \bmod d = 0$, 一次除法即可——多項式時間。故合數問題 $\in \mathbf{NP}$。(順帶一提: 質數問題也 $\in \mathbf{NP} \cap \mathbf{co\text{-}NP}$, 且由 AKS 演算法 (2002) 知其實 $\in \mathbf{P}$。)

練習 2. 針對例 4.2 的 $S = \{3, 34, 4, 12, 5, 2\}$, 目標改為 $t = 20$。找一個證書, 並說明如何驗證。

解答

證書 $\{3, 12, 5\}$:$3 + 12 + 5 = 20$ ✓。驗證只需一次加總並比對目標, 多項式時間。(另一個證書:$\{34 - \dots\}$ 不行, 因為 $34 > 20$; 但 $\{4, ...\}$ 亦可探索——重點是只要存在一個通過驗證的子集即為 YES 實例。)

練習 3. 有人說「因為 SAT 要試 2^n 種指派, 所以 SAT $\notin$ **NP**」。指出這句話的錯誤。

解答

2^n 是**求解** (找指派) 的代價, 與 **NP** 無關。**NP** 只要求「**驗證**一個給定指派」要快, 而代入求值確實是多項式時間。故 SAT $\in$ **NP**。把「求解難」當成「不屬於 **NP**」是典型誤解。

練習 4. 為什麼 NP-complete 問題中只要有一個屬於 **P**, 就能推出 **P** = **NP**?

解答

設 NP-complete 問題 $X \in \mathbf{P}$。對任意 $Y \in \mathbf{NP}$, 由 NP-完備性有 $Y \leq_p X$(多項式歸約)。把 Y 的輸入用多項式時間翻成 X 的輸入, 再用 X 的多項式演算法求解, 合起來仍是多項式時間, 故 $Y \in \mathbf{P}$。於是 $\mathbf{NP} \subseteq \mathbf{P}$; 配合 $\mathbf{P} \subseteq \mathbf{NP}$, 得 **P** = **NP**。

練習 5 (思考題)**.** 判斷對錯並說明:「若有人給出**非構造性**的證明說 **P** = **NP**, 則隔天所有銀行加密就會被破解。」

解答

錯。非構造性證明只保證快速演算法「存在」, 並不給出演算法。沒有實際可執行的程式, 就無法破解任何加密。實務衝擊取決於證明是否構造出低次多項式的演算法。

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:NP、驗證者定義、NP-完備性)
3. S. Arora, B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, 2009.
4. S. A. Cook, "The Complexity of Theorem-Proving Procedures," *STOC*, 1971;L. Levin, 1973.
5. R. M. Karp, "Reducibility Among Combinatorial Problems," 1972.
6. S. Aaronson, "P $\stackrel{?}{=}$ NP," 與部落格文章〈P vs. NP for Dummies〉, https://scottaaronson.blog/?p=459.

7. Wikipedia: *P versus NP problem*; *NP (complexity)*.

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