

# 1 NP 問題

從開始學習演算法到現在，我們已經針對許多問題提出了高效的算法，而這些算法通常有多項式級別的複雜度，然而有些問題人類還是一籌莫展，沒辦法發展出多項式級別的解法。底下我們會簡單地介紹  $NP$  系列的複雜度類別，並且介紹一些經典的  $NP$  類問題。在開始之前，我們得先簡單的定義何謂問題：

**Definition 1** 一個問題可以是下列兩種之一：

- $Yes/No$  問題
- 最優化問題

一個問題的解則必須給出對應的方案： $Yes/No$  問題如果答案為 **Yes**，那麼須給出一組滿足條件的解；如果答案為 **No**，也須給出一個滿足條件的證明（本文終將不會討論到 **No** 的情形，所以請先別太在意）；最優化問題中則須給出最優方案的解。舉例而言，背包問題需要給出應該選哪些物品，LIS 問題需要給出子字串。

接著我們要介紹四種問題的級別： $P$ 、 $NP$ 、 $NP-hard$ 、 $NP-complete$  的定義

**Definition 2**  $P$  (*polynomial time*) 是由存在多項式複雜度解演算法的問題形成的集合。

這是我們目前最熟悉的複雜度類型，舉凡有  $O(n)$ ,  $O(n \log n)$ ,  $O(nk)$  時間複雜度解法的問題都屬於此類。

**Definition 3**  $NP$  (*nondeterministic polynomial time*) 是由存在多項式複雜度驗證演算法的問題形成的集合。

※ $NP$  並不是 not polynomial time。

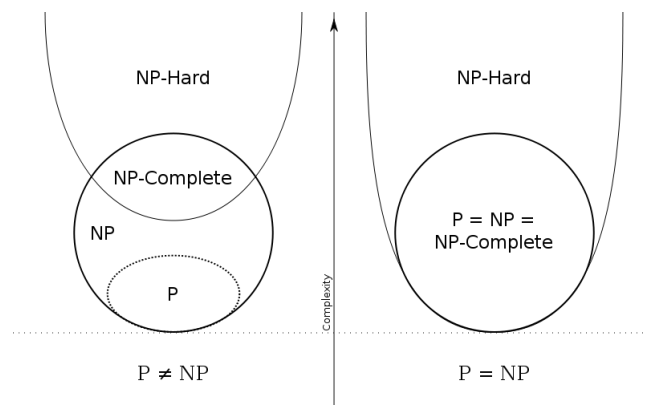
對於一個問題  $P$  以及一個對於此問題的輸入  $w$ ，如果  $P$  對  $w$  有解 ( $P$  為最優化問題，或者  $P$  為  $Yes/No$  問題且答案為 **Yes**)，那麼對於此解  $s$ ，若存在一個演算法能驗證  $s$  是否為  $P$  對  $w$  的解，那麼我們說這個演算法是  $P$  的一個驗證演算法 (identifier)。舉例而言，排序問題 ( $P$ ) 的輸入為一組序列 ( $w$ )，解也是一組序列 ( $s$ )，而我們可以透過檢查  $s$  是否遞增與  $s$  的元素是否可與  $w$  內相同的元素一一對應，來驗證  $s$  是否為排序問題對  $w$  的一組解。我們至少可以在  $O(n^2)$  內完成這件事情，因此排序問題也屬於  $NP$  集合。顯然  $P$  是  $NP$  的子集 (一個可以得到解的演算法本身就可以作為一個驗證演算法)，但是否  $NP$  也是  $P$  的子集 (即  $P=NP$ ) 則尚未確定。

**Definition 4** *NP-hard* 是由滿足如下條件的問題  $P$  組成的集合：如果  $P$  存在多項式解演算法，則任意屬於  $NP$  集合的問題都存在多項式解演算法。意即，所有屬於  $NP$  的問題都可以在多項式時間歸約到  $P$  問題。

也就是說，對於一個屬於 *NP-hard* 的問題  $P$ ，以及任意一個屬於  $NP$  集合的問題  $Q$ ， $Q$  都可以在多項式時間內歸約到  $P$ 。要證明一個問題是 *NP-hard* 非常不容易，因為必須要能夠證明任意一個  $NP$  問題都能夠歸約到該問題。

**Definition 5**  $NP-complete(NPC) = NP \cap NP-hard$

*NP-hard* 並不一定完全包含  $NP$ ，兩者的交集為  $NPC$ 。以下為  $P \neq NP$  和  $P = NP$  時， $P$ 、 $NP$ 、 $NPC$ 、*NP-hard* 之間的關係：(Image from Wiki)



要證明一個問題屬於 *NP-complete* 非常困難，因為要同時證明屬於  $NP$  和 *NP-hard*。一般來說，要證明一個問題屬於  $NP$  其實不太困難，瓶頸還是在證明屬於 *NP-hard*。以下將介紹第一個被證明為 *NP-complete* 的問題。

**Definition 6** *CNF (Conjunctive normal form)* 的定義如下：

- 一個布林變數 (Boolean variable)  $v$  只可能為兩種值：True(T) 或 False(F)。若  $v = T$ ，則  $\neg v = F$ ；若  $v = F$ ，則  $\neg v = T$ 。
- 一個文字 (literal)  $l$  形如  $v$  或  $\neg v$
- 一個子句 (clause)  $c_i$  形如  $(l_1 \vee l_2 \vee \dots \vee l_{m_i})$
- 一個合取範式 (CNF) 的布林算式 (Boolean formula) 形如  $c_1 \wedge c_2 \wedge \dots \wedge c_n$

(注： $\vee$  為布林運算中的「或 (OR)」， $\wedge$  為布林運算中的「且 (AND)」)

舉例來說，算式  $(a \vee b) \wedge (\neg b \vee c)$  是 CNF，而算式  $(a \wedge b) \vee (\neg b \wedge c)$  則不是 CNF。

**Definition 7** *CNF-SAT (Boolean Satisfiability problem)* 的輸入為一個 CNF 算式，問題為「是否包含一組變數組使得輸入的算式結果為真」。

依照一開始對問題的定義，如果該問題答案為 **Yes**，需要給出每個變數的值；如果答案為 **No**，則需要給出證明（我們不會討論到這個部分）。以上述的算式  $(a \vee b) \wedge (\neg b \vee c)$  而言，答案為 **Yes**，可以使運算結果為真的變數值為  $a = T, b = F, c = T$ 。

**Theorem 1** (*Cook-Levin Theorem*)  $CNF-SAT \in NPC$ 。

有了以上定理，我們就可以更容易的證明一個問題  $P$  屬於  $NP-hard$  了：只要隨意找一個屬於  $NP-hard$  的問題  $Q$ （例如  $CNF-SAT$ ），並證明  $Q$  可以在多項式時間內歸約到  $P$  就可以了。如此一來，所有屬於  $NP$  的問題都可以先歸約到  $Q$ （由  $NP-hard$  的定義），再由  $Q$  歸約到  $P$ ，於是所有屬於  $NP$  的問題都可以在多項式時間內歸約到  $P$ ，因此  $P$  就屬於  $NP-hard$ 。

### 例題 1

3-CNF 為一種特殊形態的 CNF，其中每個 clause 都恰好有 3 個 literal。請證明  $3-CNF-SAT \in NPC$ 。

### 解答

證明分為兩部分，需要證明  $3-CNF-SAT \in NP$  和  $3-CNF-SAT \in NP-hard$

(1) 首先證明  $3-CNF-SAT \in NP$ 。

驗證解答很簡單，只要將每個變數的值代入，就可以在  $O(n)$  ( $n$  為 clause 數量) 時間內得到算式的結果，再判斷是否為 **T** 就可以了。由於可以在多項式時間內驗證解答， $3-CNF-SAT \in NP$ 。

(2) 為了證明  $3-CNF-SAT \in NP$ ，我們將已知為  $NP-hard$  的  $CNF-SAT$  問題歸約到  $3-CNF-SAT$  問題。

對於任意的 CNF 算式  $f = c_1 \wedge c_2 \wedge \cdots \wedge c_n$ ， $c_i$  不一定恰好包含 3 個 literal，但我們可以用以下方法產生等價的 3-CNF 算式。

- $c_i = (l_1)$ ：將  $c_i$  替換為  $(l_1 \vee l_1 \vee l_1)$
- $c_i = (l_1 \vee l_2)$ ：將  $c_i$  替換為  $(l_1 \vee l_2 \vee l_2)$
- $c_i = (l_1 \vee l_2 \vee l_3)$ ：維持不變
- $c_i = (l_1 \vee l_2 \vee \cdots \vee l_{m_i})$ ：將  $c_i$  替換為  $(l_1 \vee l_2 \vee t_1) \wedge (\neg t_1 \vee l_3 \vee t_2) \wedge (\neg t_2 \vee l_4 \vee t_3) \wedge \cdots \wedge (\neg t_{m_i-3} \vee l_{m_i-1} \vee l_{m_i})$ ，其中  $t_j$  為新增的變數。

替換的複雜度  $O(nm)$ ，為多項式時間。經替換後的算式  $f'$  為 3-CNF-SAT 的合法輸入，且雖然變數數量不同 ( $f'$  新增了許多變數)，但  $f$  有解  $\Leftrightarrow f'$  有解

(術語為 *equisatisfiable* )，於是我們就將 CNF-SAT 問題歸約到 3-CNF-SAT，從而證明了  $3\text{-CNF-SAT} \in NP\text{-hard}$ 。

由 (1),(2)，得證  $3\text{-CNF-SAT} \in NPC$ 。

## 習題

1. 請找出可以讓以下布林算式的運算結果為 **T** 的解。如果無解，請直接寫無解。

(a) (10 pts)  $(a \vee b) \wedge (\neg a \vee c) \wedge (\neg b \vee \neg c) \wedge (\neg a \vee b \vee c)$

(b) (10 pts)  $(a \vee \neg d) \wedge (\neg a \vee b) \wedge (\neg b \vee c) \wedge (\neg a \vee \neg b \vee \neg c) \wedge (a \vee \neg c \vee d)$

2. De Morgan's laws (狄摩根定律)：

- $\neg(a \vee b) = (\neg a) \wedge (\neg b)$

- $\neg(a \wedge b) = (\neg a) \vee (\neg b)$

DNF (Disjunctive normal form) 的定義如下：

- 一個布林變數  $v$  只可能為兩種值：**True(T)** 或 **False(F)**。
- 一個文字 (literal)  $l$  形如  $v$  或  $\neg v$
- 一個子句 (clause)  $c_i$  形如  $(l_1 \wedge l_2 \wedge \cdots \wedge l_{m_i})$
- 一個 DNF 的布林算式形如  $c_1 \vee c_2 \vee \cdots \vee c_n$

※ 以下題目皆假設每個 clause 不會有重複的布林變數，例如  $(a \vee b \vee \neg a)$ 。

(a) (10 pts) 給定一個 CNF 算式  $f$ ，請將  $\neg f$  用 DNF 表示。

(b) (10 pts) 給定一個 CNF 算式  $f$ ，目前並沒有多項式時間的演算法，可以判斷  $f$  的運算結果是否可以為 **T**。那麼，請問有沒有多項式時間的演算法，可以判斷  $f$  的運算結果是否可以為 **F** 呢？

(c) (10 pts) 假設  $P \neq NP$ ，給定一個 DNF 算式  $f$ ，請問是否存在多項式時間的演算法，可以判斷  $f$  的運算結果是否可以為 **T**？如果有，請描述該算法；如果沒有，請證明。

(d) (20 pts) 假設  $P \neq NP$ ，給定一個 DNF 算式  $f$ ，請問是否存在多項式時間的演算法，可以判斷  $f$  的運算結果是否可以為 **F**？如果有，請描述該算法；如果沒有，請證明。

(e) (20 pts) 已知可以利用布林運算的分配律和 De Morgan's laws 將所有的布林算式轉換成 CNF 形式。假設  $P \neq NP$ ，請問是否存在演算法可以在多項式時間內將任意布林算式轉換成 CNF 形式呢？如果存在，請描述該算法；如果不存在，請證明。

3. (20 pts) 兩個布林算式可能看起來不同，但表示的其實是同一個函數，如以下兩個算式是等價的 (分配律)：

$$(a \vee b) \wedge c = (a \wedge c) \vee (b \wedge c)$$

請證明：判斷兩個布林算式是否不相等為 *NP-hard* 問題。