9up 小日常

Episode 2
圖論的應用：以「倒水問題」為例
Application of Graph Theory: By Considering The 'Bucket Problem'

#數學 #電腦科學 #圖論
#Mathematics #ComputerScience #GraphTheory

1 前言 Introduction

拖咗十世，終於出咗啦！同未睇過嘅人介紹下，呢個系列主要係我亂玩嘅時候發現嘅嘢，或者我好有興趣嘅課題，總之我想講咩就講咩嘅一個偽期刊。

今次講數學（準確嚟講係電腦科學），話說我原本諗住係做偽隨機數生成（Pseudorandom number generation），但係研究咗一陣發現有啲難搞，所以作罷。呢個係用 LaTeX（簡單嚟講係一個排版工具）寫嘅，所以個排版（應該）好靚嘅同時搞到我好痛苦。

2 問題 Problem

唔知你有無係一啲地方見到我會稱之為「**飲水問題**」的所謂測智商問題（真係測唔測到值得再一期去寫），問題如下：

而家有容量 5L 同 3L 嘅水桶各一個，
假設有無限嘅水，
點樣唔用任何工具裝出 4L 嘅水？

解答：先將 5L 嘅水桶裝滿。將水倒落 3L 嘅水桶度。呢個時候 5L 嘅水桶裏面有 2L 水，3L 嘅水桶有 3L 水。將 3L 嘅水桶啲水倒走，再將 5L 水桶嘅水倒過去。呢個時候 5L 嘅水桶無水，3L 嘅水桶有 2L 水。最後，裝滿 5L 嘅水桶，再將 5L 水桶嘅水倒過去。呢個時候，5L 水桶有 4L 水，3L 水桶有 3L 水。再倒走 3L 水桶嘅水後，就有一個裝住 4L 水嘅水桶。

咁你就成功解決呢個問題啦！

3 普遍化 Generalization

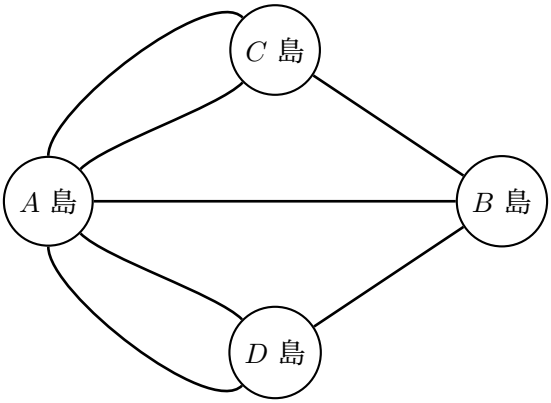
當然我哋唔會淨係滿足於某個特定嘅問題，事實上你仲可能會見到，用 5L+7L，6L+11L，甚至三個水桶嘅「倒水問題」。我哋將問題延展到呢個版本：

而家有 N 個水桶各一個，容量各為 $C_1, C_2, \cdots, C_N$，
假設有無限嘅水，
點樣唔用任何工具裝出 k L 嘅水？

我哋可以利用 Bézout's Identity（見附錄）去解決呢個問題，不過我哋今日會主要集中於**水桶嘅狀態變化**，亦啱係如果我哋由某個初始狀態，經過裝水，倒水等等嘅操作，究竟可唔可以到達某個狀態。比如我哋陣間會見到，3L 水桶同 5L 水桶係呢個問題底下，係整唔出 1L+2L 嘅水。喺涉及狀態變化嘅時候，通常**狀態圖**（State diagram）會幫得上忙，而呢個都引出我哋今日嘅主題：**圖論**（Graph thoery）。

4 圖論介紹 Brief Introduction of Graph theory

所以乜係圖論？故事要返去 1735 年，請到我哋嘅數學大神 Leonhard Euler 出場。呢個係有關七條橋嘅故事，後世就叫做「柯尼斯堡七橋問題」(Seven Bridges of Königsberg)。當時俄羅斯有七座橋，啲人想一次過行曬呢七座橋而唔重覆。

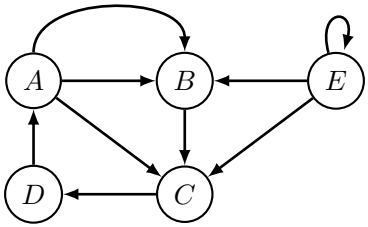


上圖中嘅線就係嗰七座橋。你搵唔搵到行曬七座橋但係唔重覆嘅走法呢？

答案：Euler 證明咗係無可能㗎！其實呢個問題嘅本質係「一筆畫問題」。

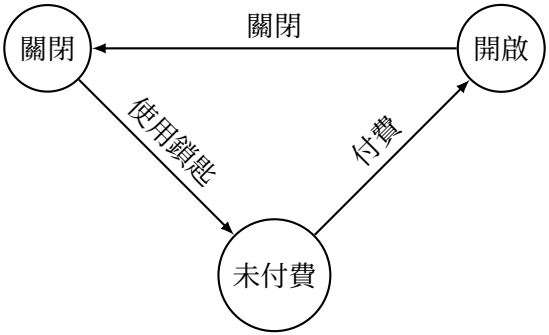
你可以仔細諗下點解無可能（提示：注意每個島嘅橋數目）。當然我哋今日嘅主題唔係一筆畫問題，但係呢個就係圖論嘅起源。我哋啱啱見到嘅就係所謂嘅**圖**。圖就係一堆**頂點**（Node / Vertex）同佢哋之間嘅**邊**（Edge）。

上圖中嘅「A 島」，「B 島」，「C 島」同「D 島」就係呢張圖嘅頂點，至於個七條橋就係邊。注意上圖中嘅邊係無方向嘅，呢啲叫做無向圖（Undirected Graph），相反嘅係有向圖（Directed Graph）, 啫係邊有方向嘅圖。



圖其中一個好用嘅 intuition 就係當佢係 IG Followers。例如係頂點 C 有條邊去 D，代表 C follow 咗 D，但係 D 無邊返去 C，代表 D 無 follow C。當然呢個比喻唔係完美，但係都算可以。注意喺上述嘅圖之中，有一條由 E 去 E 嘅邊，自己返去自己嘅邊叫做自環（Loop）。仲有，兩個頂點之間亦可以存在多條邊，例如 A 同 B。

有向圖非常適合表達狀態（State）嘅流動，例如我哋可以以有向圖表達付費儲物櫃嘅狀態圖：



係上圖中，有三個狀態（圖論中嘅頂點）：「關閉」，「未付費」同「開啟」。佢哋之間存在一啲「轉換」，例如由「未付費」狀態，只要畀錢就可以

去到「開啟」狀態，呢啲「轉換」去返圖到，就係邊。

5 問題正式化 Formalization

依家我哋有咗圖論嘅知識，我哋可以點樣去解決呢個問題？見到數學問題嘅時候，最好可以用數學嘅語言嚟描述佢。我哋首先定義問題嘅基礎條件。

定義 Definition 1. 有 N 個水桶的倒水問題，記作 $B(C_1, C_2, \cdots, C_N)$，其中第 i 個水桶的最大容量為 C_i。

呢個記號雖然睇落嚟好複雜，但係舉下例子就好容易理解：$B(3,5)$ 就係我哋當初嘅 3L+5L 水桶遊戲，有一個 3L 水桶同 5L 水桶。而 $B(2,3,6)$ 就係一個有三個水桶嘅水桶遊戲，最大容量分別係 2L，3L，6L。

喺呢個問題中，水桶嘅狀態會因為我哋倒水而變化，所以圖論可能會幫到手。我哋會想構建一個圖嚟表達水桶狀態之間嘅轉換。我哋先要定義係呢個問題中水桶嘅「狀態」，啱係水桶嘅水量。

定義 Definition 2. 在 $B(C_1, C_2, \cdots, C_N)$ 中，一個「狀態」（即一個頂點）是指一個長度為 N 的整數組合 $S(c_1, c_2, \cdots, c_N)$，其中對於每個整數 $1 \leq i \leq N$，有 $0 \leq c_i \leq C_i$。而「空狀態」則指 $S(0, 0, \cdots, 0)$。我們使用 $S(c_1, c_2, \cdots, \overset{n}{c_n}, \cdots, c_N)$ 中的 $\overset{n}{c_n}$ 表達 c_n 處於第 n 個位置。

上面嘅定義係咪又睇落嚟好複雜？其實呢個嘅意思就係水桶嘅水量，不過用咗比較數學嘅語言去形容。例如係 3L+5L 嘅倒水問題（亦啱係 $B(3,5)$）$S(0,0)$（亦啱係空狀態）表達咗所有水桶都係空嘅情況。而 $S(0,4)$ 就係表達咗第一個水桶係空，但係第二個水桶有 4L 水。注意 $S(4,0)$ 係 $B(3,5)$ 之中唔存在，一個最大容量係 3L 嘅水桶點裝 4L 嘅水？所以係定義度，先會有咁多不等式。

小問題：係 $B(5,3)$ 之中，$S(4,0)$ 存唔存在？$S(0,4)$ 呢？

我哋定義咗頂點，係時候要定義圖之中嘅邊。喺狀態圖之中，邊係「狀

態轉換嘅方式」。例如喺儲物櫃個例子之中，「付費」就係由「未付費」狀態轉換到「開啟」狀態嘅方式。咁係倒水問題裏面嘅狀態點轉換？就係透過倒水。

我哋可以將倒水分成三種：

I. 為一個水桶裝水

填滿一個水桶，無論裏邊有幾多水，亦啫係嗰個水桶嘅水量變成水桶嘅最大容量。

II. 倒走一個水桶嘅水

清空一個水桶，無論裏邊有幾多水，亦啫係嗰個水桶嘅水量變成 0。

III. 轉移一個水桶嘅水到另一個水桶

將水桶 A 嘅水轉移至水桶 B。為咗方便啲，設水桶 A 有 W_A 咁多水，而填滿水桶 B 所需嘅水量叫 W_B。

如果 $W_A \leq W_B$，咁我哋就可以將 A 裏面嘅全部水轉移過去，代表 A 水量下降 W_A（被清空），B 水量上升 W_A。如果 $W_A > W_B$，水桶 B 淨係夠裝 W_B，所以 A 只可以轉移 W_B 咁多水，代表 A 水量下降 W_B，B 水量上升 W_B（被填滿）。我哋可以見到水桶 A 總係轉移「W_A 同 W_B 之間最細嗰個」咁多水。

如果我哋將「W_A 同 W_B 之間最細嗰個」叫做 $E = \overset{\text{最小值}}{\min}(W_A, W_B)$ 嘅話，呢種倒水嘅本質就係 A 水量下降 E，B 水量上升 E。

我哋仔細研究咗每種倒水嘅方式。注意對於每個水桶狀態都有呢啲轉換，之後我哋嚟嚴格咁定義佢哋。我哋將由 A 到 B 嘅有向邊寫做 $A \to B$。

定義 Definition 3. 在 $B(C_1, C_2, \cdots, C_N)$ 中的一個頂點，$S(c_1, c_2, \cdots, c_N)$，存在以下有向邊（以下的 n, m 是正整數，且 $1 \leq n, m \leq N$ 和 $m \neq n$）：

I. Full_n：$S(c_1, \cdots, c_N) \to S(c_1, \cdots, \overset{n}{C_n}, \cdots, c_N)$

II. Dispose_n：$S(c_1, \cdots, c_N) \to S(c_1, \cdots, \overset{n}{0}, \cdots, c_N)$

III. $\text{Transfer}_{n,m}$，其中 $E = \min(c_n, C_m - c_m)$：

$$S(c_1, \cdots, c_N) \to \begin{cases} S(c_1, \cdots, \overset{m}{c_m + E}, \cdots, \overset{n}{c_n - E}, \cdots, c_N) & (m < n) \\ S(c_1, \cdots, \overset{n}{c_n - E}, \cdots, \overset{m}{c_m + E}, \cdots, c_N) & (n < m) \end{cases}$$

呢個定義睇落嚟非常恐怖，但係其實佢就係我哋上面講嘅三種倒水類型：「為一個水桶裝水」對應住 Full_n，「倒走一個水桶嘅水」對應住 Dispose_n，而「轉移一個水桶嘅水到另一個水桶」就對應住 $\text{Transfer}_{n,m}$。

先睇 Full_n，你見到喺 Full_n 之中第 n 個位置變成咗 C_n，啫係代表第 n 個水桶嘅水量變成 C_n，亦係嗰個水桶嘅最大容量，可以見到呢個對應住裝滿第 n 個水桶。係 Dispose_n 之中，第 n 個位置變咗 0，代表第 n 個水桶嘅水量變 0，表示倒空第 n 個水桶。

至於最麻煩但又最關鍵嘅 $\text{Transfer}_{n,m}$，我哋見到佢有分兩個情況，$m < n$ 同 $n < m$，但係呢個其實唔重要，只係為咗整啱個位置。重點係，第 n 個位置對比之前嘅 c_n 減少咗 E，而第 m 個位置比之前 c_m 多咗 E。我哋見到水桶 A 轉移咗 E 咁多水到水桶 B。再睇返 $E = \min(c_n, C_m - c_m)$，當中 c_n 就係第 n 個水桶原本有嘅水量，而 $C_m - c_m$ 就係填滿第 m 個水桶所需嘅水量。呢個正正就係對應住將第 n 個水桶嘅水轉移到第 m 個水桶。

再舉下例子，喺 $B(3,5)$ 之中，$S(2,1)$ 有 6 條往外嘅邊：

$$\begin{cases} \text{Full}_1: & S(2,1) \to S(3,1) \\ \text{Full}_2: & S(2,1) \to S(2,5) \\ \text{Dispose}_1: & S(2,1) \to S(0,1) \\ \text{Dispose}_2: & S(2,1) \to S(2,0) \end{cases} \quad \begin{cases} \text{Transfer}_{1,2}: & S(2,1) \to S(0,3) \\ \text{Transfer}_{2,1}: & S(2,1) \to S(3,0) \end{cases}$$

當然，有陣時唔係所有邊都有意義。例如喺 $B(3,5)$ 之中嘅空狀態 $S(0,0)$，佢依然有 6 條往外嘅邊，但係只有 2 條通去其他頂點嘅邊：通去 $S(3,0)$

嘅 Full_1 同通去 $S(0,5)$ 嘅 Full_2，其他嘅邊都係通去自己嘅自環。無水又點樣倒空水或者轉移水呢？

小問題：係 $B(C_1, C_2, \cdots, C_N)$，啱係有 N 個水桶嘅情況下，一個頂點有幾多條向外嘅邊？(答案：$N(N+1)$，點解？)

到呢度，我哋已經成功定義咗我哋嘅研究目標：水桶狀態圖 $B(C_1, C_2, \cdots, C_N)$ 啦！

6 廣度優先搜索 BFS (Breadth First Search)

依家我哋有一個圖，我哋可以點樣喺裏面提取需要嘅資訊，例如我想知道喺 $B(3,5)$，由空狀態開始，我需要幾多次倒水先可以達到 $S(1,0)$？呢個時候有關圖嘅一啲**演算法**（Algorithm）就會有用。

演算法係咩？我哋喺電腦科學講嘅演算法，通常唔淨係指網絡媒體推薦內容嘅方式，例如 Youtube 推送影片，或者 IG 嘅 Reels 一樣。更加宏觀咁講，演算法係「一步步用嚟處理資料嘅指令」。

演算法非常似食譜，教機器一步步去做嘢。例如食譜會叫你落 10ml 油，之後再落 15g 糖嘅樣。演算法都類似，會好精準咁叫部機器做嘢。

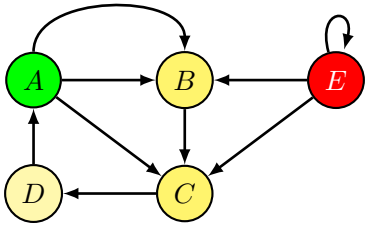
其中一個最簡單嘅演算法就係二分搜尋法（Binary Search）。假設我比你一堆由低到高排列嘅數字，當中肯定有 100，每次只可以睇一個數字，你會點樣用最少嘅時間搵到 100？可能會有人即刻答先睇最中間嗰個，如果佢細過 100，就睇右半邊，如果大過 100，就睇左半邊，向睇嗰邊再做類似嘅嘢，直到搵到 100 為止。呢個將目標範圍一半一半咁縮小嘅尋找方法就叫做 Binary Search。

關於演算法仲有好多嘢可以講，不過今日我哋主要集中喺有關圖嘅演算法。我哋而家有一個圖 $B(C_1, C_2, \cdots, C_N)$，啲應該點樣由呢個圖身上提取有用嘅資料？有一類演算法叫做圖遍歷演算法(Graph Traversal Algorithms)可以幫到我哋。咩叫做遍歷？就係行完一張圖裏面所有嘅頂點。呢類演算法就係幫我哋以特定嘅方法去行完一張圖嘅頂點，其中一個就係**廣度優先搜**

索，簡稱 **BFS** (Breadth First Search)。

BFS 嘅步驟係咁嘅：我哋首先指定一個開始嘅頂點。我哋將所有呢個頂點去到嘅，而又未去過嘅頂點行一次，記錄落嚟。對於每個記錄落嚟嘅新頂點，我哋當佢係第二個開始頂點，重覆將呢個頂點去到嘅，而又未去過嘅頂點行一次，記錄落嚟。等到原本嘅記錄嘅頂點都做曬呢個過程之後，再對新紀錄嘅頂點做同一樣嘢。

更加形象化咁講，BFS 有啲似喺流水，由開始頂點流到第一層，由第一層流到第二層……我哋攞上次嘅一個圖做示例：



圖用咗顏色標示，方便大家睇。圖中綠色嘅頂點 A 就係開始頂點。然後我哋開始睇呢個頂點往外嘅邊，有一條通去 C 嘅邊，仲有兩條通去 B 嘅邊。所以我哋記錄低 B 同 C。

到下一個循環，B 得一條邊去 C，但係 C 已經去過，所以可以忽略。C 有一條邊通向 D，所以我哋記錄低 D。

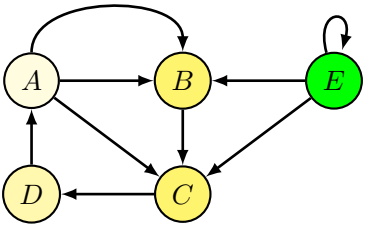
下一個循環，D 得一條邊去 A，但係 A 已經去過（開始頂點），所以可以忽略。到呢度，無新嘅頂點被記錄低，所以演算法結束。

我哋檢視下啱啱人腦執行嘅演算法結果，每個循環記錄嘅頂點：(A)（開始頂點），(B, C)，(D)。我哋發現記錄嘅時間，同由 A 去嗰個頂點嘅距離（最少要行過幾多邊先可以到達個頂點）有關。

第一個循環嘅由 A 到 A 嘅距離自然係 0。第二個循環，由 A 到 B 嘅距離係 1，由 A 到 C 嘅距離都係 1。第三個循環，由 A 到 D 嘅距離係 2。可以見到 BFS 可以幫我哋搵到一個頂點到其他頂點之間嘅距離。

我哋將圖裏邊嘅開始頂點設作綠色，將第二層（距離 $= 1$）嘅頂點設作黃色，將第三層（距離 $= 2$）嘅頂點設作淺黃色，一層比一層淺。注意我哋係 BFS 中從來都無記錄過 E，因為由 A 無可能去到 E。我哋將呢啲頂點稱做從 A 不可達（Unreachable from A）。我哋將不可達嘅頂點設作紅色。

要注意嘅係 BFS 嘅開始頂點唔同，結果都好唔同。我哋試下由 E 開始：



我哋認識咗 BFS 之後，咁可以喺我哋嘅問題上做到啲咩？我哋知道 BFS 可以幫我哋搵到一個頂點同其他頂點之間嘅距離，如果應用到倒水問題上，我哋就可以研究係空狀態（或者其他任何狀態）需要倒幾多次水先可以到其他狀態，又或者幾多狀態係可以由某個狀態經過有限次倒水而到達。呢種問題我哋都可以用 BFS 去解決。

所以我就用咗 Python 寫咗個演算法出嚟，整體除咗用到 BFS 之外，上面我哋努力定義嘅數學模型亦都派上用場。而最後嘅結果我放咗上 Github，一個（主要）存放程式碼嘅地方，大家可以自行取閱，亦可以試下去運行佢。

喺呢度大概講下要點用：你開始運行之後會出現一個問題 `Buckets capacity:`，你可以打 `3|5` 嚟表示你想執行喺倒水問題 $B(3,5)$，3 同 5 分別喺水桶嘅容量，而 `2|3|7` 就係 $B(2,3,7)$。啲數字要用 `|` 隔開。

再下一個問題係開始頂點 `Initial state:`，你可以好似啱啱咁打 `1|2` 嚟表示開始頂點係 $S(1,2)$，亦可以漏空表示開始頂點係空狀態。

跟住程式就會幫你執行 BFS 嚟搵到每一層可以到達邊啲狀態。例如呢個係對 $B(3,5)$ 做 BFS，開始頂點係空狀態嘅結果：

```
0: 0|0
1: 3|0 0|5
2: 0|3 3|5 3|2
3: 3|3 0|2
4: 1|5 2|0
5: 1|0 2|5
6: 0|1 3|4
7: 3|1 0|4
-----
STATE COUNT: 16
IDEAL STATE COUNT: 24
COVER RATE: 66.7%
```

上面描述咗每一層有邊啲頂點，由開始頂點到每個數字後面頂點嘅距離就會係嗰個數字。例如由空狀態到 `0|2` 嘅距離係 3，亦啫係可以由空狀態透過三次倒水嚟到達 $S(0,2)$。而我哋亦都得到原本倒水問題「點樣得到 4L 嘅水」，所需要嘅最少倒水次數：七次。

下面就係一啲統計數據：`STATE COUNT` 係由開始頂點出發可以到達嘅頂點數目，亦啫係可達嘅頂點數目。`IDEAL STATE COUNT` 就係可能（唔一定可以由開始狀態到達）嘅狀態數目。最後 `COVER RATE` 就係可以到達嘅頂點數目佔總可能狀態數目。

以下係嗰 Github Repository：

`https://github.com/lazy-student217/bucket-problem`

呢個係網上版，可以即刻運行到（不過我唔肯定連結會唔會失效）。

`https://www.programiz.com/online-compiler/9K0Iqygep1feT`

7 後記 Trivia

終於完喇 !! 其實今次拖咁耐除咗因為我懶（主要原因）之外就係內容嘅長度比之前長咗好多，足足寫咗十幾頁。除此之外就係萬惡嘅 LaTeX。如果有錯嘅地方，或者講得唔清楚嘅地方可以隨便講。

其實我覺得呢類型題目應該冇咩人睇（原本呢個就無人睇，都係 for fun 啫），不過都係做咗（I don't care）。如果你堅持咗 10 頁嚟到呢度，可以留個反應證明你嘅恆心同忍受我唔清唔楚語言嘅能力。

之後呢個系列都係隨緣更新，做呢啲唔討好又晒氣嘅嘢好難迅速產出。

返去呢個題目，大概收個結。其實圖論呢個數學分支（同相關嘅演算法）比你想像中更加常出現，例如 Google Map 幫你搵嘅最佳路徑就用咗圖論同佢啲演算法：Dijkstra's algorithm 同 A* algorithm。今次舉嘅呢個例子只可以部分反映圖論嘅某啲用途，但係如果可以引起你呢方面嘅興趣就已經算係達到咗我嘅目標。當然無其實都無關係，當係學咗個奇怪嘅新知識，如果日後不幸地要用到都可以知道關咩事。

話說之前莫名奇妙喺學校地下學會壁報嗰處，搵到數學學會都有講圖論，仲要交問題證明畀某麥姓老師有小禮物。咁我應該（應該）寫好過佢，無論喺有趣定其他方面都係（自肥）。歡迎去睇下然後返嚟同我呢個做比較。

頂唔住 LaTeX，之後去學 Typst。LaTeX 實在太令人崩潰（然而依家幾乎絕大部分學術論文都用 LaTeX 寫，唔好提 Word，更加垃圾）。

大概就係咁。

記於二零二五年三月九日。

8 附錄 Appendix / 以數論方法解倒水問題

我哋會用數論方法簡單證明，喺 $B(C_1, C_2, \cdots, C_N)$ 之中，如果 $\overset{\text{最大公因數}}{\gcd}(C_1, \cdots, C_N) = 1$，亦啫係 $C_1, \cdots, C_N$ 嘅最大公因數係 1，嗰我哋就可以倒

出任何水量（$\leq$ 最大水桶嘅水量）嘅水。

喺證明之前，我哋簡單介紹 **Bézout's identity**。呢一個定理有小小抽象。如果我有 n 個數字 $a_1, \cdots, a_n$，佢哋嘅最大公因數 $A = \gcd(a_1, \cdots, a_n)$ 有一啲特別。我會有一啲特別嘅整數 $x_1, \cdots, x_n$，令到 $x_1a_1 + \cdots + x_na_n = A$。例如 $\gcd(4, 3) = 1$，咁嗰啲特別整數就係 1 同 -1，因為 $1(4) + (-1)3 = 1 = \gcd(4, 3)$。

依家證明（非嚴謹）開始：

假設 $B(C_1, C_2, \cdots, C_N)$ 中最大嘅水桶容量為 $C_{\max}$，而最大嘅水桶係第 M 個水桶。注意對於第 n 個水桶，可以都透過 Fill_n，$\text{Transfer}_{n,M}$ 嚟令第 M 個水桶增加 C_n 咁多水。假設第 M 水桶裝唔曬啲水，咁第 n 個水桶就會剩低一部份水，啫係 $(c_M + C_n) \bmod C_{\max}$，其中 c_M 係未轉移前 M 水桶有嘅水（點解？）。我亦可以透過 $\text{Transfer}_{M,n}$ 加 Dispose_n 嚟令水桶 M 少咗 C_n 咁多水。

簡單嚟講，對於第 n 個水桶，我可以令水桶 M 增加或者減少 C_n 咁多水。因此，我哋可以強硬咁將水桶 M 嘅水量整到 $a_1C_1 + a_2C_2 + \cdots + a_NC_N$ 咁多水，無論 $a_1, a_2, \cdots, a_N$ 係咩都得。而因為 $\gcd(C_1, \cdots, C_N) = 1$，根據 Bézout's identity，有一啲整數 $x_1, \cdots, x_N$ 令到 $x_1C_1 + \cdots + x_NC_N = \gcd(C_1, \cdots, C_N) = 1$，代表我哋可以整到水桶 M 有 1L 嘅水。倍大成條式 m 倍，得到 $mx_1C_1 + \cdots + mx_NC_N = m$，代表我哋可以整到水桶 M 有 mL 嘅水，無論 m 係幾多。因此，我哋可以倒出任何水量嘅水。 □