中華民國第 56 屆中小學科學展覽會

作品說明書

國小組　數學科

最佳團隊合作獎

080407

滑動方塊中的數學

學校名稱：臺中市神岡區豐洲國民小學

作者：	指導老師：
小五 張文碩	徐瑛黛
小五 陳玠融	黃怡妮
小五 趙芝雨	
小五 吳孟綸	

關鍵詞：App 遊戲、滑動方塊、旗標軌跡

壹、摘要：

本作品是研究手機平板遊戲 WrapSlide 的遊戲規則與滑動技巧，針對遊戲的動作，觀察其滑動過程與動作的結果，探討其連續滑動之動作並建立相關的數學公式，定義出旗標方格與軌跡，探討方陣與旗標軌跡的旋轉對稱性，並建構一個數學的估算方法，去計算完成遊戲中各種難度或層次所需的最多滑動次數，並將上述探討之滑動方塊的性質，擴充與應用至無限大之方陣。

貳、研究動機：

學校買了新的平板電腦，上課時，老師常會讓我們拿來查資料或上課討論。在一次上課中，發現平板裡面還有遊戲真好玩。可是有一個 App 遊戲好難，於是問了其他同學，大家一起討論，沒想到產生了高度興趣，於是提議可以一起去請教老師；老師看了這個 App 遊戲後，說這個問題很不錯，大家可以一起製作推盤來研究看看，於是我們便開啟了滑動方塊的研究，同時開始蒐集相關資料，老師也和我們約定了每週練習與討論的時間，指導我們開始做一連串的研究。

參、研究目的：

一、探討在方陣的方格操作時，『有去有回』的連續動作設計的概念。
二、探討在方陣的方格中，旗標方格的意義與旗標方格數目的探討。
三、探討在方陣的方格操作時，旗標方格軌跡與其軌跡逆向性的探討。
四、探討遊戲中對稱性以及方陣旋轉對動作的影響。
五、研究遊戲中遊戲的規則與建立數學公式，並估算出遊戲各種難度所需之最多滑動次數，以及定義出『有去有回』的上帝之數(God's number with go-back)。

肆、滑動方塊的由來與介紹

一、滑動方塊的由來

最近在手機與平板上出現的一款 益智遊戲WrapSlide App，是由南非斯泰倫博斯大學 (Stellenbosch University) 的研究員 Alewyn Burger 所設計，它由魔術方塊簡化而來，可以說是一款平面型的魔術方塊，它像是一個滑動的拼圖，在其滑動的格子中具有多種顏色的方塊，遊戲初始時會將方塊顏色隨機混和，玩家必須將這些具有不同方塊顏色的方格藉由滑動與遮蔽的動作，將顏色方格分開，該遊戲中提供3種不同方陣大小，4x4、6x6和8x8方陣，以及4種不同顏色的難易度，提供玩家選擇，玩家的挑戰是如何以最少的滑動次數完成所有方格。

二、滑動方塊WrapSlide App下載網址

Apple: https://itunes.apple.com/app/wrapdlide/id795712935?mt=8

Android:https://play.google.com/store/apps/details?id=com.wrapslide.android.wrapslide

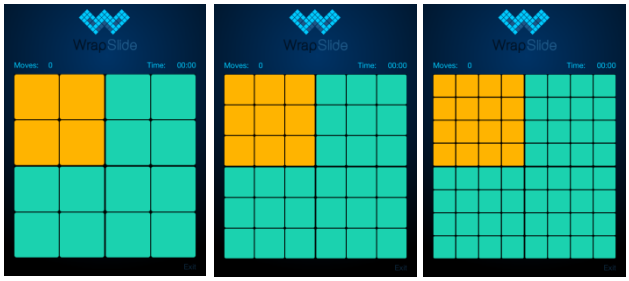
三、玩法的介紹

WrapSlide App 像是一款平面型的魔術方塊，中間以粗黑線條隔開，縱的一條、橫的一條，就像是平面的 x-y 軸一般，此兩粗黑線條將方陣隔開成四個象限，滑動時一次必須移動兩個象限，也就是上軸、下軸、左軸、右軸，例如上軸即包含第一與第二象限，兩象限包裹起來可以左右移動，所以可以有向左、向右兩種動作，下軸亦是如此，以此類推，右軸、左軸也分別可以上下移動，因此遊戲可以有八種滑動動作，將整個推盤移動成同色的在一起如下圖一至圖三，其中四個象限的顏色次序不拘，只要顏色相同即可，在遊戲軟體中有三種方陣大小，亦即 4x4 方陣，6x6 方陣，8x8 方陣請參考圖四，難易度有區分為：第一種是一種顏色加一底色(如圖一)，例如黃色為標的顏色，天空色為底色，當將黃色集中在一象限時即完成遊戲，第二種是兩種顏色加一底色(如圖二)，第三種是三種顏色加一底色(如圖三)，以上三種採初始啟動時隨機編排方格顏色，這裡三種方陣大小搭配一色、兩色、三色三種配色方式，所以一共有九種難度的遊戲可供選擇，另開發者開發一種方格顏色週期排列的初始狀況如圖五，圖五(a)三色的 4x4 方陣，圖五(b)一色的 6x6 方陣，以及圖五(c)兩色的 8x8 方陣，雖然初始狀況不同，但遊戲原則相同，必須完成每一象限均是相同顏色。

圖一、6x6 方陣一顏色加一底色

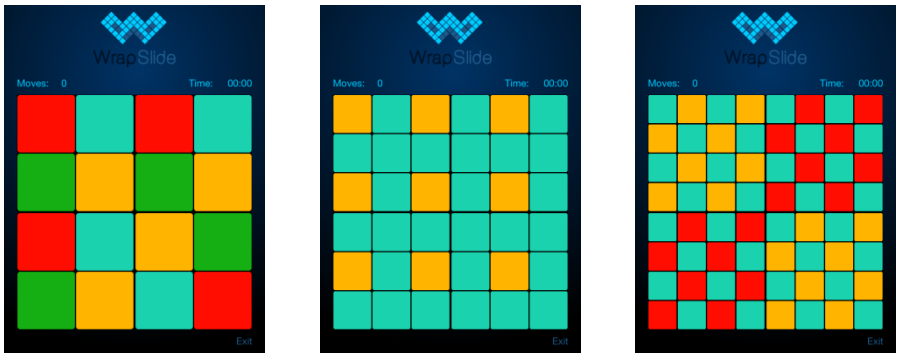
圖二、6x6 方陣二顏色加一底色

圖三、6x6 方陣三顏色加一底色



(a) (b) (c)

圖四、WrapSlide 中三種方陣(a) 4x4 方陣，(b) 6x6 方陣，(c) 8x8 方陣



(a) (b) (c)

圖五、WrapSlide 中顏色週期排列的三種方陣(a)三色 4×4 方陣，(b)一色 6x6 方陣，(c)兩色 8x8 方陣

四、名詞介紹

方陣：平面上長與寬具有相同數目的方格數。

方格：方陣中的格子點，其為正方形格子。

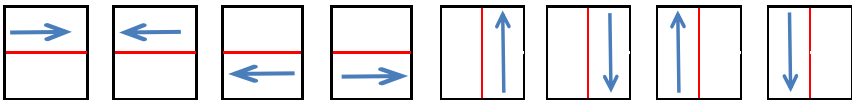
象限：將一平面以x軸與y軸隔成相等的四等分，每一等分為一象限，右上角定義為第一象限，依逆時針方向旋轉，左上角為第二象限，左下角為第三象限，右下角為第四象限。

旗標：在連續多次滑動動作中，方陣中的某些方格會完全依照動作移動，則這些方格可以稱為旗標方格。

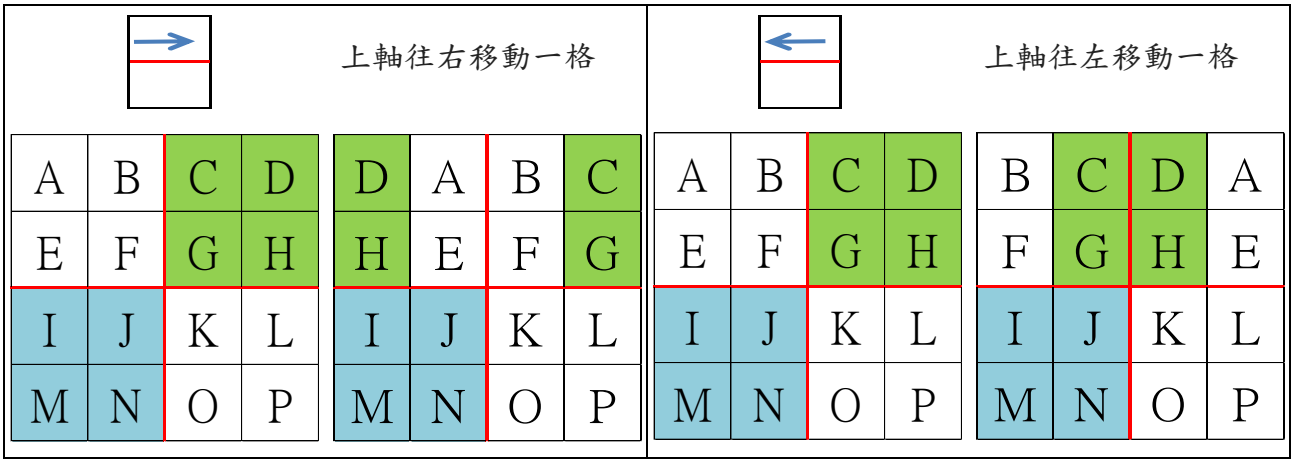
旗標軌跡：旗標方格滑動的足跡稱為旗標方格軌跡，簡稱旗標軌跡。

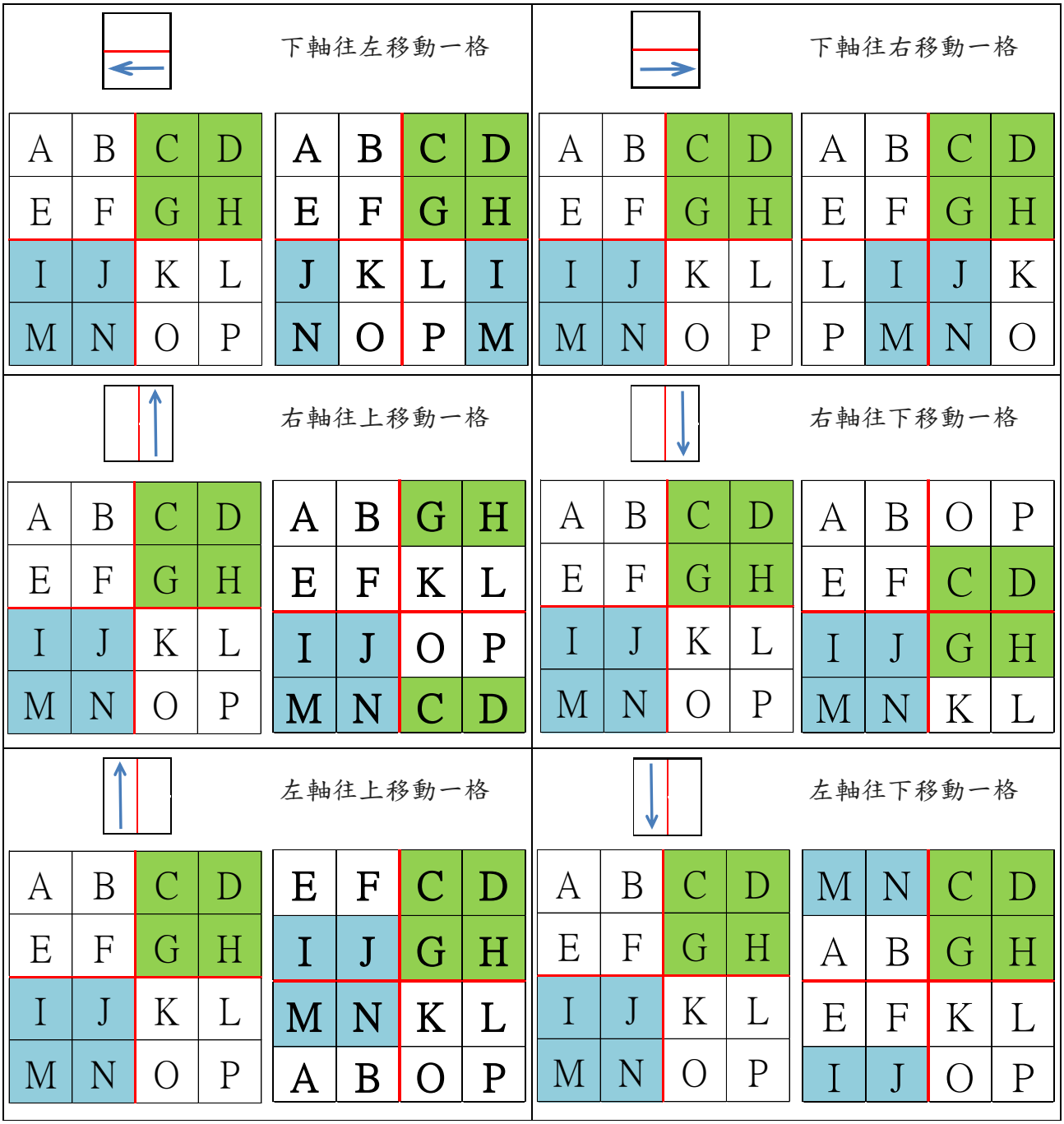
五、滑動方塊的基本動作表示圖形介紹

用以下這 8 個符號來代表 8 種不同的基本動作：



上面動作說明圖中，紅色直線代表基準線，紅色橫線分開上下軸，紅色縱線分開左右軸，藍色箭號線代表動作滑動的方向，下面以 4×4 方陣舉例說明，並將每一方格標上字母使易於觀察方格移動的動向。

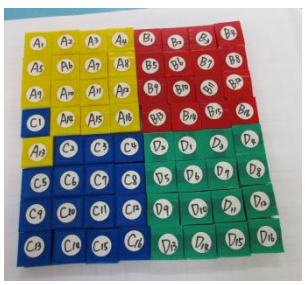




伍、研究設備及器材：

(a)平板電腦App、(b)自製推盤、(c)Excel。

(a)



(b)

A1	A2	A3	B1	B2	B3
A4	A5	A6	B4	B5	B6
A7	A8	A9	B7	B8	B9
C1	C2	C3	D1	D2	D3
C4	C5	C6	D4	D5	D6
C7	C8	C9	D7	D8	D9

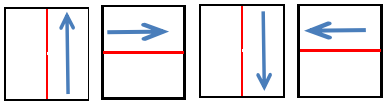
(c)

陸、研究過程與研究結果：

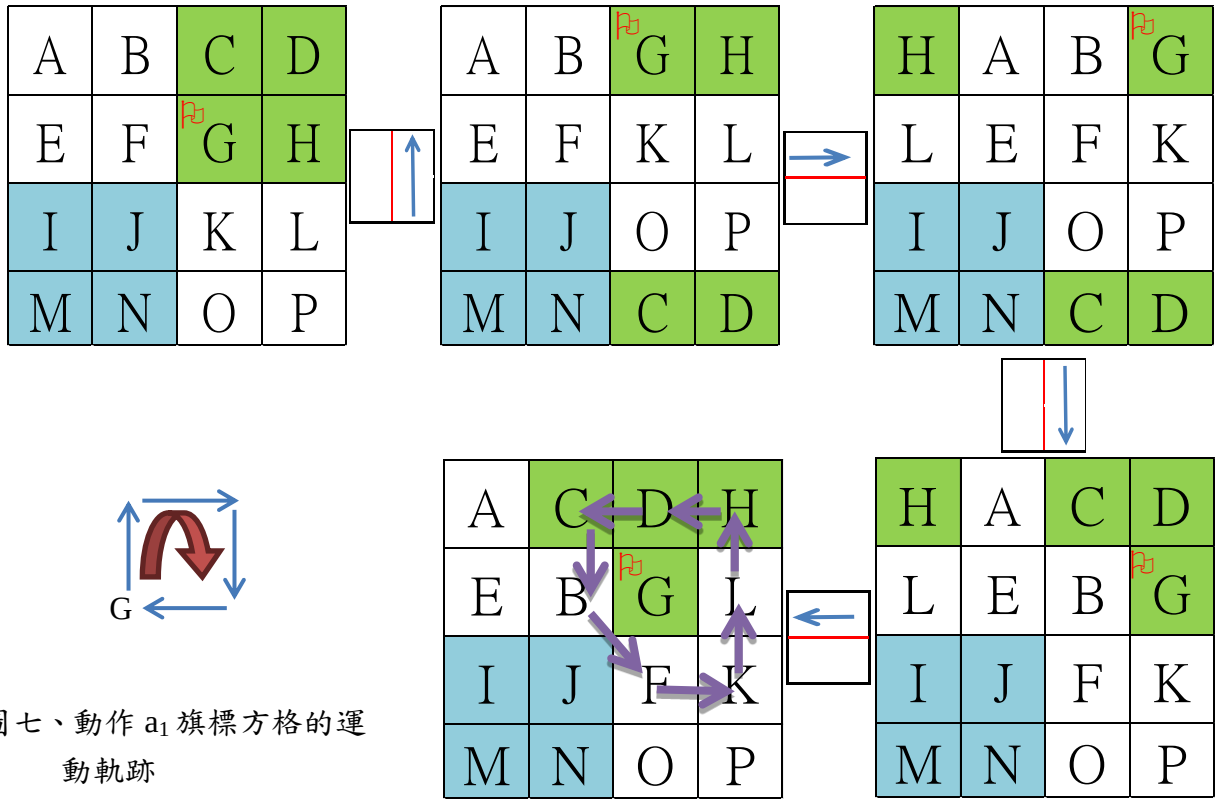
【問題研究一】探討『有去有回』的操作方法對方陣元素的影響？

根據參考資料[2]中的陳宏賓先生談到，在一般的魔術方塊或是推盤中，常用一種『有去有回』的動作方式，進而觀察執行動作前後，方陣中的方格相對應改變的地方，因此我們將以 4x4 方陣為例，將方陣上的每一個方格編上符號，利用自制的推盤與 EXCELL 表格來滑動，這一連串動作採用『有去有回』的原則執行，我們可以找到旗標方格，它會依循每一個動作滑動並回到原始點，也可以說旗標方格的軌跡完全與動作相符，而非旗標方格或有回到原點，但其軌跡不會與動作相符。

首先定義一動作 a_1，它是一個四次滑動的動作

定義動作 a_1：  (順時針有去有回)

以下是 4x4 方陣的實際操作過程



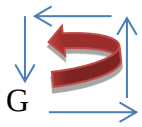
圖七、動作 a_1 旗標方格的運動軌跡

圖六、編上符號後的 4x4 方陣經動作 a_1 操作的過程

如圖六的動作 a_1 分解圖，經過動作 a_1 之後並與原來的圖作比較，發現第三象限不變，第二象限 A、E 不變，第四象限 O、P 不變，與旗標點 G 亦不變，其餘環繞方格 G 之外圈逆時針移動一格，因此以方格 G 點為參考點(第一象限左下角點)，順時針執行動作 a_1 的 4 個基本動作，可以得到如圖七的旗標軌跡，研究所有方陣中的方格的軌跡，只有方格 G 有此性質，

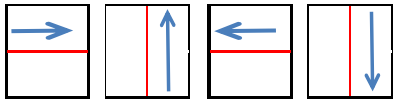
因此我們稱方格 G 為動作 a_1 的旗標方格，操作時只要將手放在方格 G 上滑動 a_1 的四個動作即可。

當我們觀察動作 a_1 的旗標方格軌跡時，可發現當旗標方格軌跡為順時針運作時，其操作結果卻是逆時針移動，因此我們發現 a_1 的逆動作，記為a_1^{-1}，同時旗標方格仍為方格 G 點，此時a_1^{-1}的軌跡為

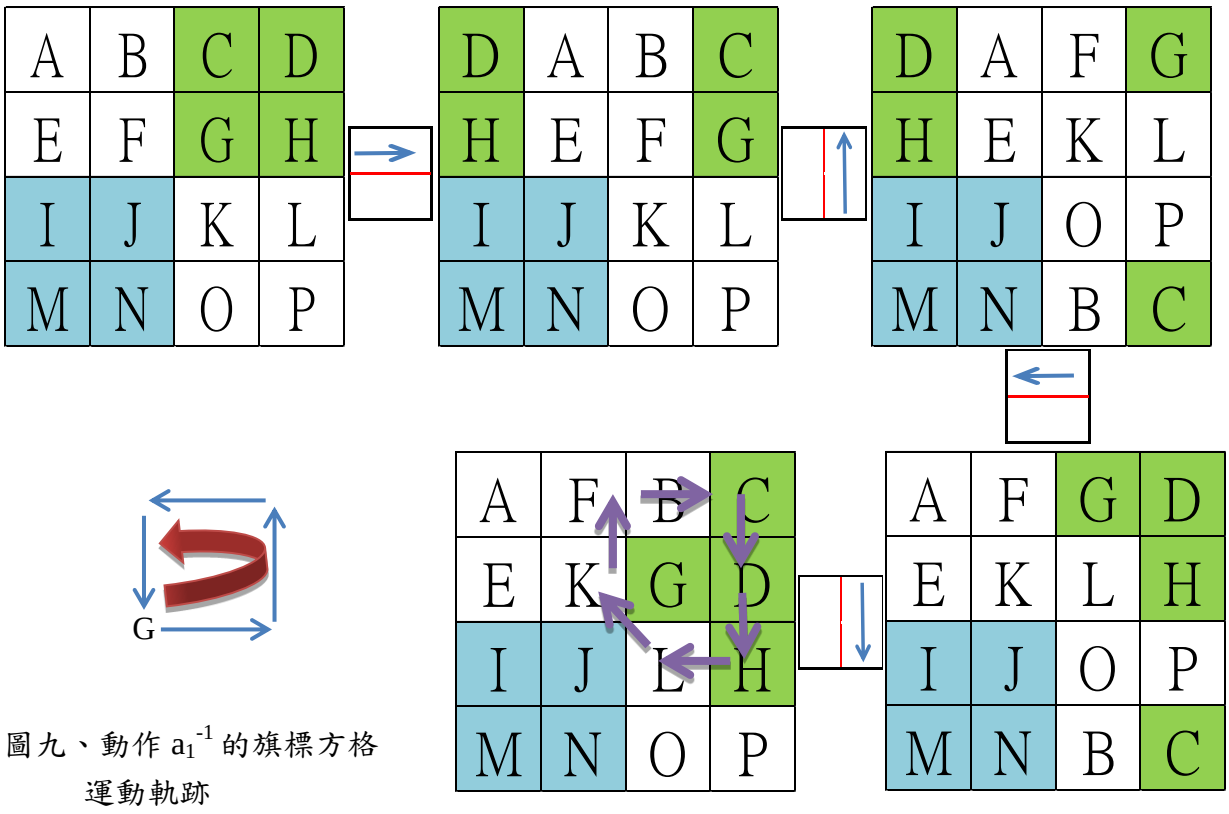


(逆時針有去有回)

所以a_1^{-1}：



以下是 4x4 方陣的實際操作a_1^{-1}的過程



圖九、動作 a_1^{-1} 的旗標方格運動軌跡

圖八、編上符號後的 4x4 方陣經動作 a_1^{-1} 操作的過程

經過動作 a_1^{-1}（為逆時針操作）之後，與動作a_1相同不變的是第三象限，第二象限的 A、E 與第四象限 O、P，旗標 G 點也是不變，其餘環繞 G 的外圈方格則是順時針移動一格。

【問題研究一結論】

當我們執行有去有回的動作時，可以找到旗標方格，其軌跡遵循我們的滑動動作，當執行順時針的動作 a_1 時，會使環繞旗標 G 的方格逆時針移動一格，反之若執行逆時針的動作 a_1^{-1} 時，將使環繞旗標 G 的方格順時針移動一格。

【問題研究二】探討動作 a_1 的旋轉對稱性？

由於本研究中方陣的兩邊都有兩象限方格，因此每一邊方格均為偶數，因此我們將方陣大小稱為 2n×2n 方陣，這裡 n 必須是大於或等於 2 以上整數，當 n=1 時該遊戲沒有任何可以玩得挑戰，這裡我們將探討方陣的旋轉對稱性，以 n=2，也就是 4×4 方陣為例，如圖十中所示，若將方陣順時針旋轉 90 度，則第一象限之方格將轉至第四象限：

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

(a)

⟹

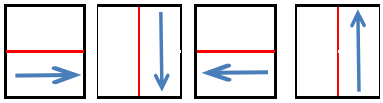
M	I	E	A
N	J	F	B
O	K	G	C
P	L	H	D

(b)

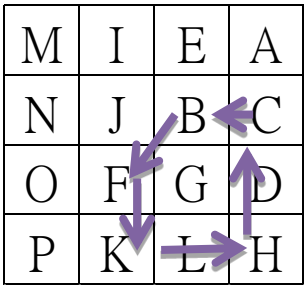
圖十、(a)原始 4×4 方陣，(b)順時針轉動 90^o 後之 4×4 方陣

此時以第四象限的旗標 G 為參考點，操作動作a_4(與動作 a_1 同為順時針)

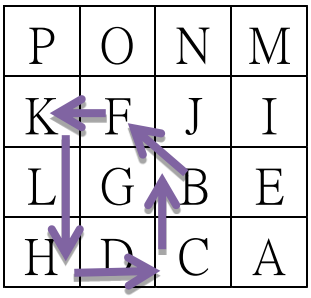
a_4：



如圖十一經動作 a_4 操作後，一樣可以得到環繞第四象限中 G 點外圍方格逆時針移動一格的動作，因此動作 a_4 可以看成是動作a_1 的旋轉 90 度投影，因此若將方陣轉動 180 度，方陣如圖十二，可以得到動作a_3 ，以第三象限 G 點為參考指標，順時針運作『有去有回』的動作，環繞 G 點的方格，逆時針移動一格。

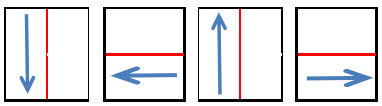


圖十一、動作 a_4 操作完的結果



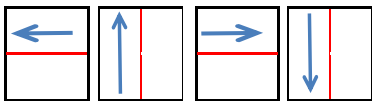
圖十二、動作 a_3 操作完的結果

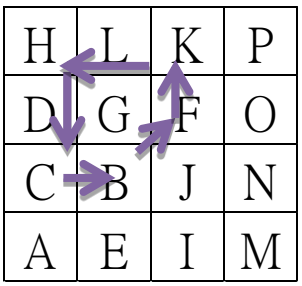
a_3：



同理，依照這種旋轉對稱的概念，當方陣旋轉 270 度時，可以得到動作a_2，請參照圖十三。

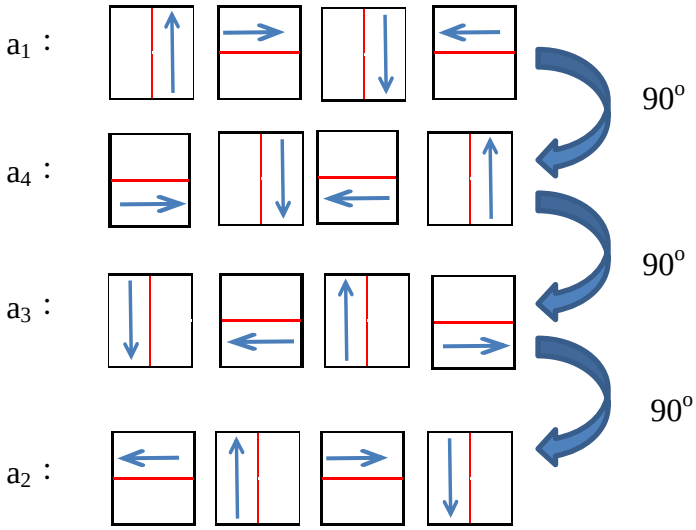
a_2：





圖十三、動作 a_2 操作完的結果

根據以上操作結果，我們將動作 a_1、a_2、a_3、a_4 整理如下表，我們以下標 1,2,3,4 代表該動作的旗標方格所在的象限。



【問題研究二結論】

a_4 是a_1 的旋轉 90 度對稱，也就是將方陣順時針旋轉 90 度後，a_1 就變成 a_4，而且a_1 中的 4 個步驟，同時也是依順時針轉 90 度，而變成a_4 中的 4 個步驟。同理，將方陣轉 180 度，a_1 轉成 a_3；方陣轉 270 度，a_1 轉成 a_2 所以我們可以統稱a_1 、 a_2 、 a_3 、 a_4 為動作 A 的家族。我們不需要記憶 a_4、 a_3 、a_2 的動作，只需要將方陣中想要操作的象限，經由方陣旋轉（**平板旋轉**）至第一象限，然後操作動作 a_1 就可以達到環繞參考旗標點，逆時針移動一格的目的。而操作a_1^{-1} （a_1 的逆動作），則可環繞參考旗標點順時針移動一格。

【問題研究三】探討如何利用動作 a_1 將第一象限填滿相同顏色之方格與旗標方格數？

(a)如何利用動作 a_1 將第一象限填滿

接下來我們想要探討利用動作a_1 將特定的方格移入第一象限中，例如 Wrapslide 中相同顏色方格，收集放入第一象限，以下介紹原始方陣中(如圖十四)，G 點為參考旗標點，經 a_1 動作後並不會改變位置，這裡假設目標為 E 點，想要將 E 點放入第一象限中 H 的位置，我們需先將 E 移動到 L 的位置，然後操作動作a_1 ，將 E 逆時針移動一格，擠入第一象限的 H 位置中。這裡的標的方格 E 可以是第二、三、四象限的任何一方格，當第一象限固定的情況下，這三個象限可以有下軸和左軸自由移動，如同 X、Y 平面一般，只要有兩垂直軸可以移動，便可以移動到任意座標點，像這裡的例子將 E 點依左軸下移一格，然後操作下軸右移 3 格，即可到達 L 的位置，因此只需要滑動 0~3 次如圖十五，再搭配操作 a_1 即可將第二、三、四象限中的任意一方格，放入第一象限中，依此循環操作，我們可將所有與 G 點相同顏色的方格（或相同性質），放入填滿第一象限，在遊戲競賽中，滑動次數越少越好，因此在圖十五中，以標的顏色方格所在位置次數較低者優先滑動，並且滑動過程中在高次數位置的方格，可能因為前低次方格移動而被帶到滑動次數較低的區域。

此外，若是我們想操作的是動作a_1^{-1} ，那操作動作a_1^{-1}前的位置是第二象限中 B 的位置，當操作為 a_1^{-1} 後，B 位置的方格會順時針滑入第一象限中 C 的位置。

經由以上介紹，我們可以利用動作a_1 或a_1^{-1}將第一象限中，參考旗標點外的環繞點填滿與 G 點相同性質的方格。

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

圖十四、4x4 原始方陣相對位置

2	2		
2	2		
1	1	1	0
2	2	3	3

圖十五、將標的方格滑動至方格 L 所需的滑動次數圖

(b)計算動作 a_1 的旗標方格數

由 4×4 方陣中，我們已得知如何操作相同性質的方格，將其收集於第一象限中。當方陣變大，如 6×6，8×8……2n×2n，(n≥2 時)，其動作 a_1 的旗標點方格數也將隨之增多，如圖十六中可知在 4×4 方陣中，旗標點僅有 G 點一方格，在 6×6 方陣中，旗標方格有 4 格，在 8×8 方陣中，旗標方格有 9 格。

4×4 方陣

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

6×6 方陣

A1	A2	A3	B1	B2	B3
A4	A5	A6	B4	B5	B6
A7	A8	A9	B7	B8	B9
C1	C2	C3	D1	D2	D3
C4	C5	C6	D4	D5	D6
C7	C8	C9	D7	D8	D9

8×8 方陣

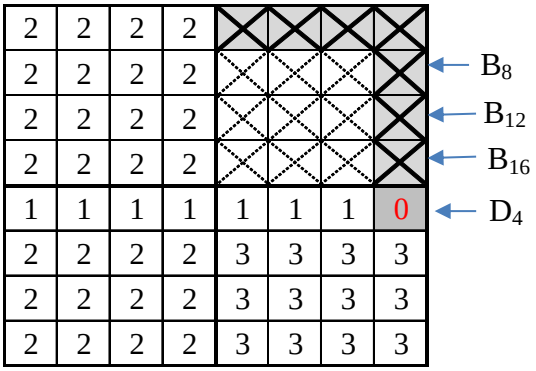
A1	A2	A3	A4	B1	B2	B3	B4
A5	A6	A7	A8	B5	B6	B7	B8
A9	A10	A11	A12	B9	B10	B11	B12
A13	A14	A15	A16	B13	B14	B15	B16
C1	C2	C3	C4	D1	D2	D3	D4
C5	C6	C7	C8	D5	D6	D7	D8
C9	C10	C11	C12	D9	D10	D11	D12
C13	C14	C15	C16	D13	D14	D15	D16

圖十六、4x4、6×6、8×8 方陣的旗標方格

由以上推論得知在 2n×2n 的方陣中，操作動作 a_1 的旗標方格會有$(n-1)^2$格，例如在 100×100 的方陣中（n＝50），旗標方格為$(n-1)^2=49^2=2401$ 格，因此當方陣越大時，想要藉由操作動作 a_1 完成一象限的顏色，必須先準備的旗標方格也越大，幸運的是我們可以藉由第一象限的滑動(上軸滑動)，輔助完成旗標方格的準備，以下討論來探討滑動次數。

(c)以動作 a_1 為基礎，計算完成一象限顏色所需滑動次數？

這裡以 8×8 方陣來說明動作原理，參考圖十六中的 8×8 方陣，方格 D4 是操作動作 a_1 的預備位置，如前述 4×4 方陣中所提的方法，經由操作下軸與左軸，參考圖十七藉由 0~3 次滑動即可將所要的方格移至 D4，這裡標示數字 0~3 代表將該位置方格移動至 D4 位置所需之滑動次數，所以如原在 D4 位置則不需滑動，目標是將第一象限中，最右邊的行中B_{16}、B_{12}、B_8，藉由連續的操作三次（0~3 次滑動+a_1），讓B_{16}、B_{12}、B_8 方格性質(顏色)相同，此時即搭配一上軸向左滑動一格，將第一象限的第四行移動到第三行，爾後可再進行重複的（0~3 次滑動+a_1），執行擠入第四行的動作；如此在 2n×2n 方格的旗標格準備時，將需執行 n-1 次的第一象限上軸左移一格的動作，而滑動前執行的（0~3 次滑動+a_1）為 n-1 次。



圖十七、將標的方格滑動至預備方格 D4 所需的滑動次數

我們可估計準備旗標方格約需滑動多少次，其中動作 a_1 中有 4 次滑動，使標的方格進入 D4 的位置需要 0～3 次滑動，所以旗標方格每一行最多需要有（n-1）×（4+3）＝7（n-1）次滑動，旗標方格有（n-1）行，且準備過程第一象限最右邊的行需搭配左移共（n-1）次，因此總滑動次數最高為

（n-1）×7（n-1）+(n-1)=$7(n-1)^2+(n-1)$次

共（n-1）行　每一行　左移次數

註：有些狀況，方格剛好在 D4 的軸上，可以少一次的滑動。

♣**若想要計算在 2n×2n 方陣中，填滿第一象限所需的滑動次數**，扣除旗標格之外的方格為 $n^2-(n-1)^2=n^2-(n^2-2n+1)=2n-1$ 格，因此除了旗標格之外，尚須執行(2n-1)×7 次（每次推入一格要操作 7 次），所以完成第一象限所需總滑動次數為

$7(n-1)^2+(n-1)+7\times(2n-1)=7[(n^2-2n+1)+2n-1]+n-1=7n^2+(n-1)$次

♣另一種思考方式是，考慮推入一個方格進入第一象限中，需要準備最多 3 次滑動，再加上操作 a_1 需要 4 次滑動，並且第一象限中一共有 n^2 個方格，再加上製備旗標方格需滑動(n-1)次，因此最多總滑動次數需要 $7n^2+(n-1)$次，或是說以此動作 a_1 操作完成第一象限，總滑動次數必不超過 $7n^2+(n-1)$次，以此方法再大的方陣都可以完成其第一象限為均是相同顏色的方格。

♣關於上帝之數的探討

西元 1974 年匈牙利的魯比克教授發明了魔術方塊，也稱為魯比克立方『Rubik's Cube』(參考文獻五)，在方塊的解碼發展史中，可以發現研究愛好者關心的一個指標，**上帝之數『God's number』**，它是將方塊混亂成最複雜之狀態，將其回復所需的必要移動次數，這裡以 G_n 表示，此時也可說是回復深度最深的狀態，而其它方塊混亂之後的狀態，其回復所需移動次數都低於 G_n。參考文獻六我們將重要的發展里程碑摘錄於附件一，以 3 階魔術方塊(3×3×3 cube)為例，從附件一可以發現從 1981 年開始，上帝之數 G_n 僅能推測介於某一**上界**(upper bound)與某一**下界**(lower bound)之間，此後發展其上、下界之間隙不斷地被縮小，一直到 2010 年 7 月，幾位教授、數學家、電腦工程師證實，上帝之數為 20。

WrapSlide 滑動方塊的發明者 Alewyn Burger 亦對 3 階 WrapSlide(6×6 方陣)的**上帝之數**感到興趣，參考文獻七的研究部落格並摘錄於附件二，在其研究部落格中談到，對於 6×6 方陣的任意散亂狀態中，他幸運地找到一個狀態是無法以低於 21 次滑動來回復，亦即上帝之數的**下界**目前至少為 21，至於**上界**他希望交給專家去解決。

參考以上對於上帝之數 G_n 的探討與歷史發展，我們希望以有去有回的動作 a_1 為操作條件限制下，去探討上帝之數，因此將其定義為 G_n with go-back，在 2n×2n 方陣的滑動方塊中，我們探討條件在最混亂的狀態下，也就是回復所需必要滑動次數為最多，前述所推導之滑動次數 $7n^2+(n-1)$可以定義成 G_n with go-back 的上界(upper bound)，因為前述成果顯示完成第一象限為同一顏色方格必不超過 $7n^2+(n-1)$次，而在最混亂的狀態下，參考圖十七中需有 n^2 個目標方格需滑動至預備位置，且其滑動次數至少為 2 次或 3 次(最嚴苛的狀態下)，因此加計預備滑動 2 次，以及推入第一象限動作 a_1 的 4 次滑動與填滿旗標方格的(n-1)次，因此下界(lower bound)可以定義在 $6n^2+(n-1)$，雖然這裡為 G_n with go-back 定義上下界區間間隙(Gap)稍嫌過大，但可以確定 G_n with go-back 必落在此有限區間中，如同數學家**張益唐**先生證明出相異兩質數必小於七千萬的推導後，其後者更將此目標縮小低於七千萬，因此這也許只是一個開始，而且這裡與魔術方塊不同之處是，此規則可以適用所有的 2n×2n 方陣，後續的研究也許可以專注在此上下界間隙之縮小上。

【問題研究三結論】

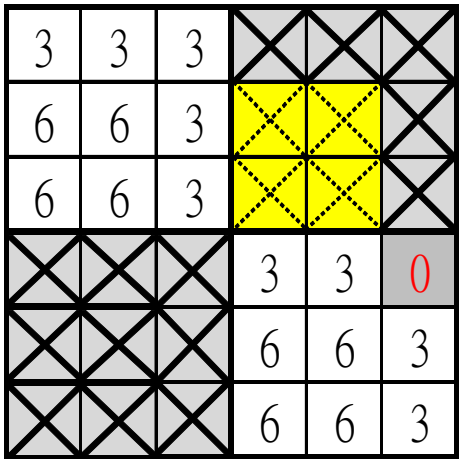
1. 在 2n×2n 方陣中，操作 a_1 或是 a_1^{-1} 前，使相同顏色方格(標的方格)進入預備位置所需滑動次數為 0~3 次，以此方法搭配 a_1 或是 a_1^{-1} 連續操作可以完成第一象限方格為相同顏色。
2. 在 2n×2n 的方陣中，操作動作 a_1 的**旗標方格**會有**$(n-1)^2$**格。
3. 在 2n×2n 的方陣中，以動作 a_1 的方式完成第一象限顏色方格，所需滑動次數最多不超過**$7n^2+(n-1)$**次。
4. 在 2n×2n 的方陣中，以動作 a_1 的方式完成第一象限顏色方格，我們定義出上帝之數 **G_n with go-back** 適用於所有方陣，其上界與下界分別為 $7n^2+(n-1)$與 $6n^2+(n-1)$。

【問題研究四】探討如何利用動作 a_1 將第二種顏色方格填滿？

當完成第一象限方格為相同顏色之後，我們若有第二種顏色的方格需要去收集且集結成為另一象限，此時根據動作 a_1 移動的方格觀察得知，第三象限並不會受動作 a_1 影響改變，因此第三象限可以作為操作 a_1 時保護第一種顏色的區域，這裡以 6×6 方陣舉例來說明：

A1	A2	A3	B1	B2	B3
A4	A5	A6	B4	B5	B6
A7	A8	A9	B7	B8	B9
C1	C2	C3	D1	D2	D3
C4	C5	C6	D4	D5	D6
C7	C8	C9	D7	D8	D9

圖十八、6×6 方陣中說明預備方格 D3

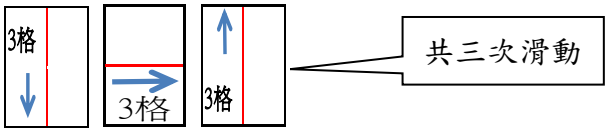


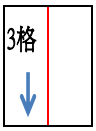
圖十九、滑動至預備方格 D3 所需的滑動次數圖

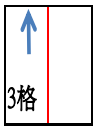
當操作動作a_1 時，旗標方格 B4、B5、B7、B8 不會改變，另第三象限[C]方陣中 C1～C9 也不會改變，而第二象限中，僅最右邊的行元素（A3、A6、A9）會移動，以上觀察，若我們先前已完成方陣一個象限的方格（或一種顏色），而要繼續操作第二種顏色時，可以將完成的象限置於第三象限，然後如同準備第一種方格顏色一樣，將第二、四象限中，第二種顏色的方格，設法移動到 D3 的位置，再操作動作a_1 ，使其滑入第一象限中，這裡比較不同的是必須保持位於第三象限中，第一種顏色完成的方格完整不被破壞。

另外從第二、四象限中的方格，滑動到操作目標方格 D3 位置的滑動次數會比較多，方法是當左軸要移動時，整個[C]方陣需移至第四象限，當下軸要移動時，整個[C]方陣要移至第二象限，因此若要將第二象限的[A]方陣的方格元素或第四象限中的[D]方陣方格元素，移至 D3 位置所需滑動次數為 0～6 次不等，圖十九中說明相對滑動次數表，若剛好落在 D3 位置則不需任何滑動。以下舉例說明滑動的方式：

例 1：在圖十八中，若是要將 A3 方格滑至 D3 位置，則需共三次滑動且維持[C]方陣在第三象限，不被操作動作a_1 時影響。



這裡說明滑動的週期性，以 6×6 方陣為例，左軸往下 3 格 

與左軸往上 3 格 

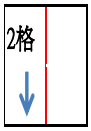
是相同的動作，同理下軸往右 3 格 

與下軸往左 3 格 

是相同的動作。

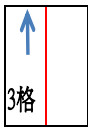
例 2：在圖十八中，若是要將 A5 方格滑至 D3 位置，則需以下動作共 6 次滑動。

（將[C]方陣移至第四象限）



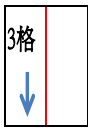
（將 A5 移到第一象限的下一列，與原 D3 位置同一列）

（將[C]方陣移回第三象限），此時 A5 在原 D2 的位置，離目標原 D3 位置尚下軸差向右一移動格。



（將[C]方陣移至第二象限）

（下軸又移一格，使 A5 進入原 D3 預備位置）



（將[C]方陣移回第三象限）

以下將估計完成推入第二種顏色方格進入第一象限需多少滑動次數，此時保持第一種顏色於第三象限，首先計算推入一方格進入第一象限，共需總滑動次數最多不超過[6 次滑動+4 次(操作 a_1)]＝10 次，其中旗標方格有$(n-1)^2$格，所以完成旗標方格最多需要 $10\times(n-1)^2+(n-1)$ 次，其中(n-1)次是完成第一象限最右邊一行之後需搭配左移所需要的次數，而完成旗標方格之外的第一象限元素方格需要 $10\times(2n-1)$次，所以總共滑動次數不超過 $10(n-1)^2+10(2n-1)+(n-1)$ ＝$10n^2+(n-1)$次。

如同完成第一種顏色之研究方法，探討完成第二種顏色之有去有回的上帝之數(亦即 G_n with go-back)，參考上述計算，完成第二種顏色的 G_n with go-back 其上界(upper bound)可定義 $10n^2+(n-1)$，另參考圖十九中在最混亂的狀態下(完成難度最高、滑動次數需最多)，滑至預備位置平均至少需要 3 次或 6 次滑動，因此其預備滑動次數最少需要 3 次，再考慮推入動作 a_1 的 4 次滑動，因此推入一個方格進入第一象限平均至少需要 7 次滑動，最後考慮總共有 n^2 個方格需要滑動，則其下界(lower bound)可定義為 $7n^2+(n-1)$，其中(n-1)是過程中準備旗標方格所產生，但事實上在最混亂的情況下，因為預備滑動次數接近 6 次的機會較多，因此 G_n with go-back 會比較接近上界(upper bound)。

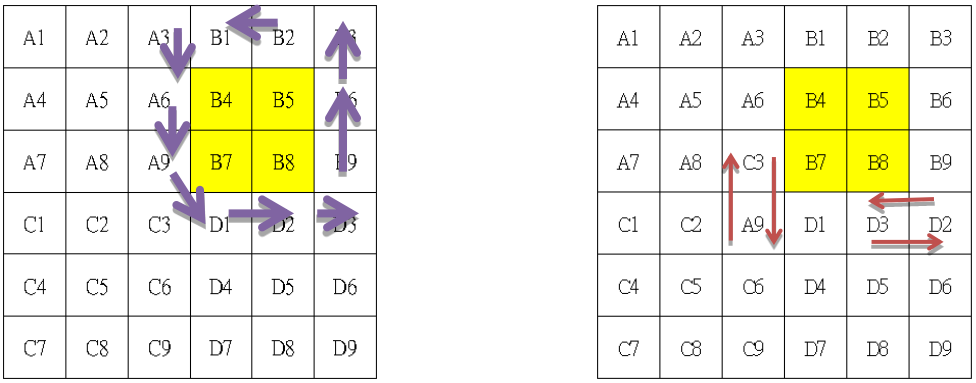
【問題研究四結論】

1. 操作動作 a_1 完成 $2n\times2n$ 方陣中第二種顏色方格，所需滑動次數必不超過 $10n^2+(n-1)$次。
2. 操作動作 a_1 完成 $2n\times2n$ 方陣中第二種顏色方格，其有去有回的上帝之數(G_n with go-back)具有上、下界分別為 $10n^2+(n-1)$與 $7n^2+(n-1)$。

【問題研究五】探討『有去有回』動作 b_1 的性質？

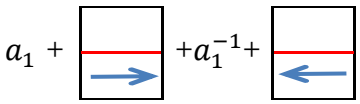
(a)動作 b_1 的設計、旗標軌跡、旗標方格與對方陣的改變

在上面的研究中聚焦 WrapSlide 遊戲滑動方塊 App，以動作a_1 為基礎，可以處理 2n×2n 方陣中兩種顏色，放入二個象限中，但當遊戲難度增加為 3 種標的顏色時，動作a_1 將無法滿足第三種顏色的聚集或蒐集，因此以下我們將討論仍以『有去有回』為基礎的延伸動作，我們以 6×6 方陣為說明舉例，首先建構原始 6×6 方陣如下：



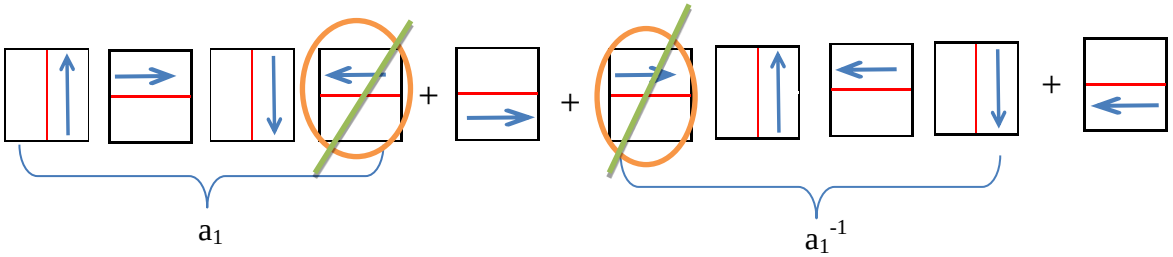
圖二十、6×6 方陣中說明(a)動作 a_1 與(b)動作 b_1 的變化

在圖二十中，考慮執行動作a_1 時，環繞旗標區 B4、B5、B7、B8 外的方格，逆時針移動一格如圖二十(a)，若執行動作a_1^{-1}時，則順時針移動一格，所以動作a_1 +動作a_1^{-1}之後方格元素將完全不改變，因此考慮加入下軸的移動動作，成為



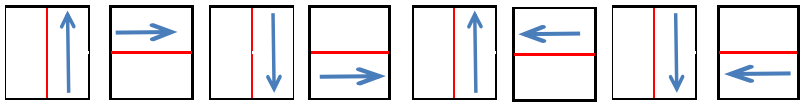
定義出此連續動作為 b_1

因此 b_1＝

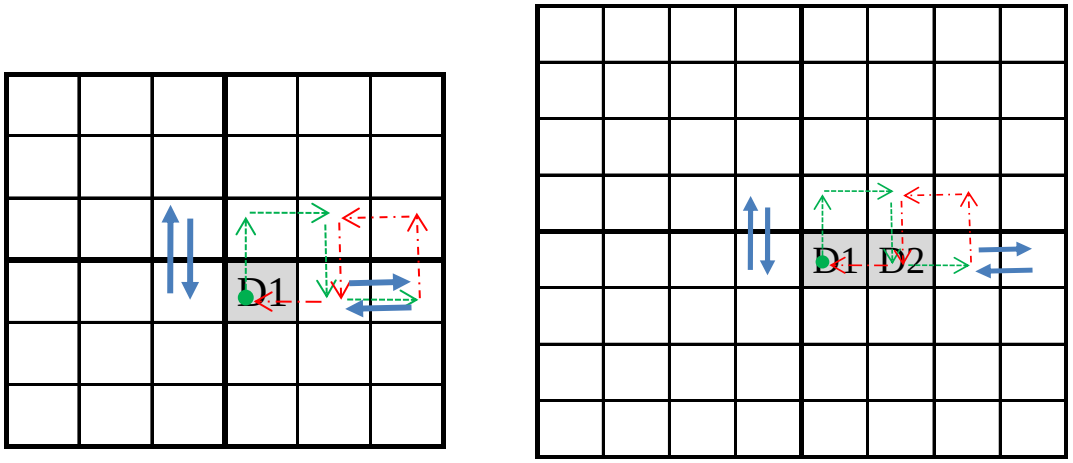


其中橘色部分兩動作為上軸相反動作，且中間只有一下軸動作，不影響上軸，因此上軸這兩相反動作可對消，以減少滑動次數，最後 b_1 為 8 個滑動次數的動作。

b1:



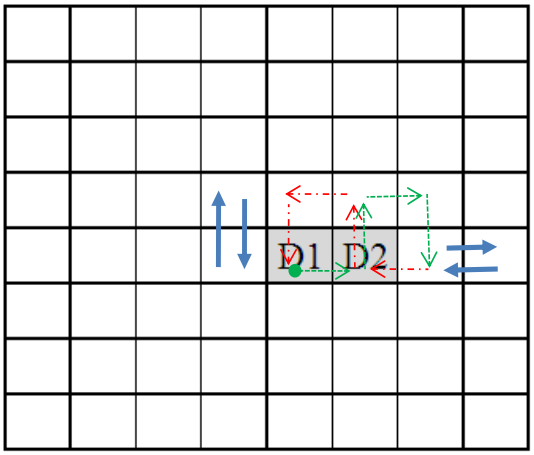
最後對原始方陣操作動作 b_1 之後，可得僅有 4 個方格變動的方陣，在圖二十(b)中呈現 A9 與 C3，D2 與 D3 兩兩交換的狀況，另外觀察動作 b_1 中一共有 8 個滑動動作，圖二十一(a)中可以看出旗標方格的軌跡呈現 **8 字形**，起點在方格 D1 的位置，軌跡中箭頭代表滑動方向，綠色 4 次滑動可以看成是『去』的方向，紅色四次滑動可以看成是『回』的方向，因此這是一個『有去有回』的動作，在 6×6 方陣中，動作 b_1 的旗標方格僅有一個 D1，而在 8×8 方陣中，旗標方格有兩個，如圖二十一(b)中之 D1 與 D2，亦即可以將旗標起點放在 D2 的位置，其動作 b_1 操作結果將會與放在 D1 位置相同，以上依此類推，對於 2n×2n 方陣中，操作動作 b_1 的旗標方格會有(n-2)個，因此在 4×4 方陣中將不會有動作 b_1 的旗標方格出現，因為此時旗標方格為 n-2=2-2=0，但仍可以操作 b_1 的八個滑動動作，只是手不能固定在某一方格上，且其動作成果仍為兩兩交換。



(a) (b)

圖二十一、(a) 6×6 方陣中動作 b_1 的旗標軌跡與(b) 8×8 方陣中動作 b_1 的旗標軌跡

如同動作 a_1 與 a_1^{-1}，動作 b_1 亦存在逆動作 b_1^{-1}，如圖二十二中 b_1^{-1} 的旗標軌跡是 b_1 的逆方向，但是動作結果仍然是相同位置的四個方格兩兩交換，因此若是操作 $b_1+b_1^{-1}$ 與操作 b_1+b_1 其動作結果會相同，一樣回到原始方陣，所以 b_1 與 b_1^{-1} 的動作結果是相同的，因此 b_1 的 8 字形旗標軌跡順繞與反繞具有相同動作結果(但需遵循數字 8 字的寫法)，因此雖然 b_1 與 b_1^{-1} 的旗標軌跡不同，但卻可看成性質相同的動作，因此也不需特別記憶 b_1^{-1} 的滑動動作。

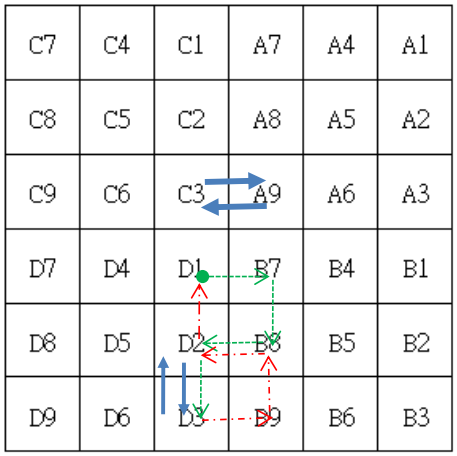


b_1^{-1} 旗標軌跡從 D1 或 D2 出發，沿綠色虛線走 8 字形回到原出發點，其操作結果，仍是圖中藍色箭頭處的兩兩交換。

圖二十二、8×8 方陣中動作 b_1^{-1} 的旗標軌跡

(b)動作 b_1 的旋轉對稱性

與動作a_1 相同，動作 b_1 也存在旋轉對稱性，也就是當我們將方陣順時針旋轉 90 度時，如圖二十三：

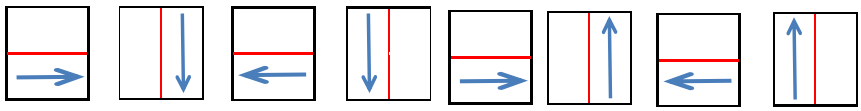


圖二十三、6×6 方陣中動作 b_4 的旗標軌跡

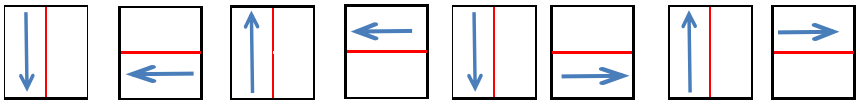
A1	A2	A3	B1	B2	B3
A4	A5	A6	B4	B5	B6
A7	A8	A9	B7	B8	B9
C1	C2	C3	D1	D2	D3
C4	C5	C6	D4	D5	D6
C7	C8	C9	D7	D8	D9

圖二十四、6×6 原始方陣

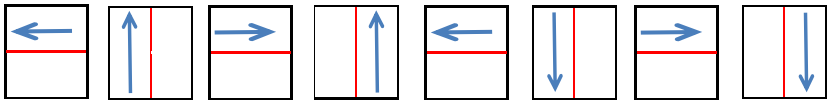
在此旋轉 90 度的方陣上執行動作 b_4，可得到圖中 C3 與 A9，D2 與 D3 兩兩交換的效果，這裡 b_4 動作內的 8 個子動作是 b_1 內動作元素順時針旋轉 90 度而成，

亦即 b_4＝ 

同理當方陣旋轉 180 度時，可得 b_3

b_3＝ 

而旋轉 270 度時，可得 b_2

b_2＝ 

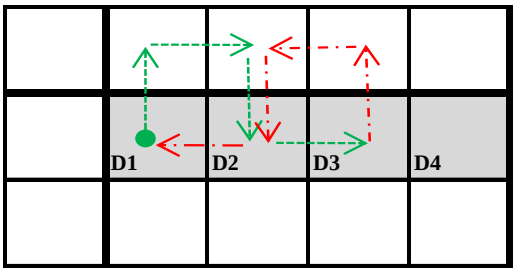
因此這裡我們亦無須記憶動作 b_4、b_3、b_2，僅需將原始方陣作逆時針旋轉即可。例如圖二十四的原始方陣中，若想將 C3 與 D1，B1 與 B4 兩兩交換，僅需將方陣順時針轉 90 度（平板順時針轉動 90 度），再操作動作 b_1 即可，亦或想將 B7 與 D1，A7 與 A8 兩兩交換，僅需轉動平板（方陣）180 度，再操作動作 b_1 即可，而平板順時針旋轉 270 度則可讓 A9 與 B7，C6 與 C9 兩兩交換。

因為逆時針轉 270 度＝順時針轉 90 度，順時針轉 270 度＝逆時針轉 90 度，可避免平板轉動太大角度，因而不小心掉落平板摔壞。最後，若是只想要二方格交換，例如原 6×6 方陣中之 A9 與 C3 交換，此時可以讓 D2 與 D3 方格顏色相同，這樣經 b_1 操作後，雖然 D2 與 D3

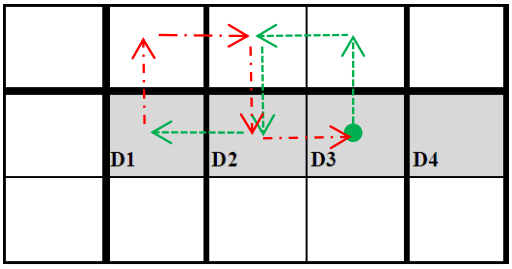
仍有交換，但 D2 與 D3 顏色看起來仍相同，此時即可達到方陣中，僅僅兩方格交換而已。所以在第三種顏色方格填蒐集過程中，任何想交換的方格，可將其操作至 A9 與 C3 的位置，並設法讓 D2 顏色＝D3 顏色，有時亦可搭配旋轉 90 度、180 度或 270 度來達到上述的目的。

(c)動作 b_1 的 8 字形旗標軌跡家族成員

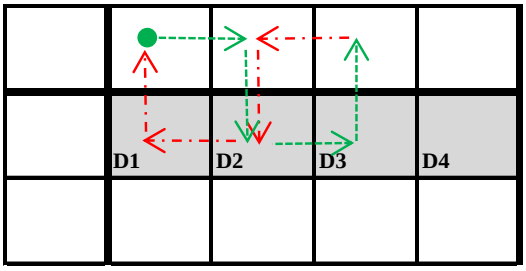
在 8x8 方陣中，首先將 b_1 旗標軌跡放大如圖二十五(a)，觀察 8 字形軌跡中有六個角點，這裡將以綠色圓形點表示起點與終點(因為動作是以『有去有回』為基礎)，**綠色線**代表『**去**』的四個動作，**紅色線**代表『**回**』的四個動作，如圖二十五中將 8 字形軌跡的六的角點定義為旗標軌跡起點，並且 8 字形中間兩點各自有兩種繞法，因此 b_1 家族有八個成員，可定義動作 B＝{b_1，c_1，d_1，e_1，f_{11}，f_{12}，g_{11}，g_{12}}，成員中軌跡方向都相同，但是起點或繞法各自不同。



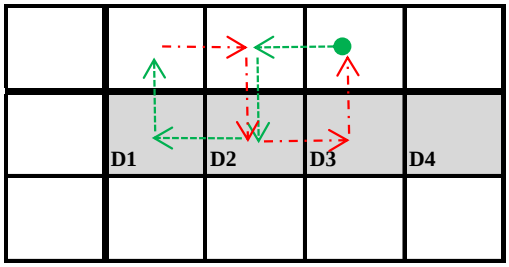
(a) b_1 的軌跡



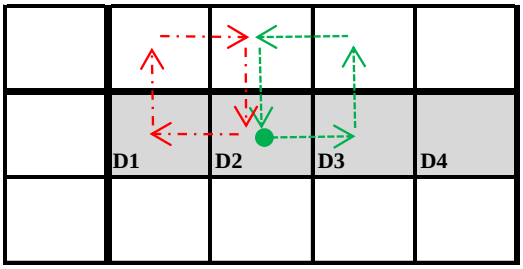
(b) c_1 的軌跡



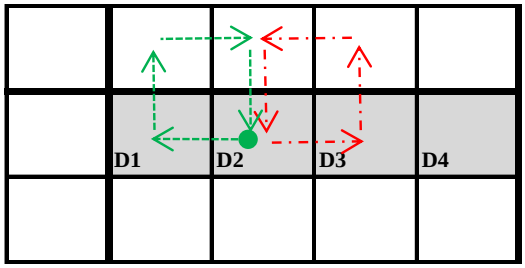
(d) d_1 的軌跡



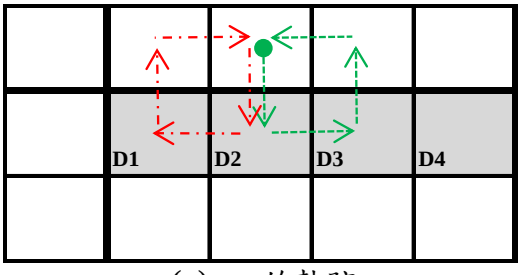
(c) e_1 的軌跡



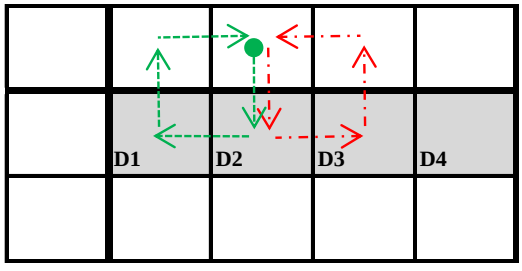
(f) f_{11} 的軌跡



(e) f_{12} 的軌跡



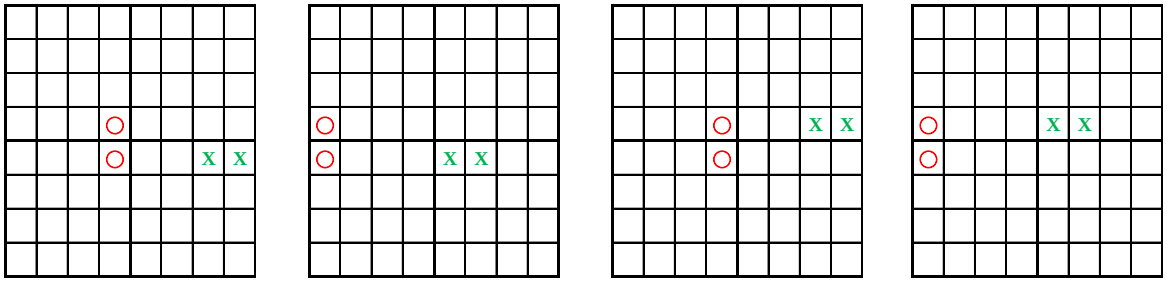
(g) g_{11} 的軌跡



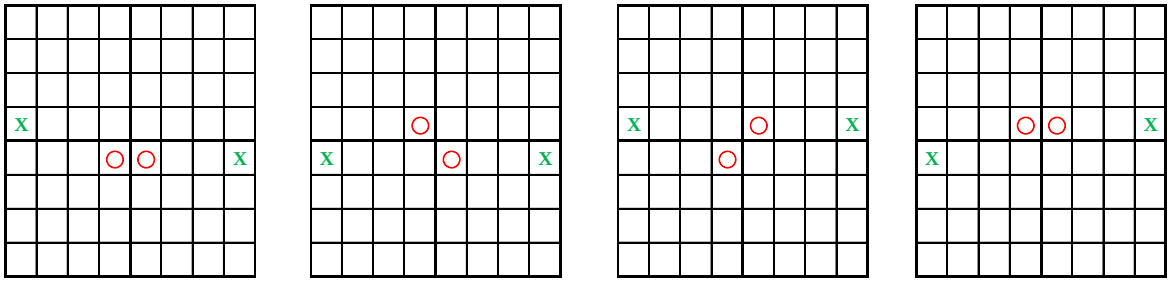
(h) g_{12} 的軌跡

圖二十五、8×8 方陣中動作 B 家族的旗標軌跡

上述動作 B 家族成員中，其旗標軌跡的共同特徵是『有去有回』，亦即起點與終點相同，並遵循 8 字形軌跡動作，旗標方格數均是(n-2)個，以下圖二十六將介紹這些旗標軌跡操作後的動作結果。



(1)b_1 操作結果　(2)c_1 操作結果　(3)d_1 操作結果　(4)e_1 操作結果



(5)f_{11} 操作結果　(6)f_{12} 操作結果　(7)g_{11} 操作結果　(8)g_{12} 操作結果

圖二十六、8×8 方陣中動作 B 家族的操作以推盤驗證的結果

從圖二十六中可發現動作 B 家族的 8 個軌跡滑動的結果，均是兩兩交換的狀況，其中◯和◯交換，X 和 X 交換，既然是兩兩交換，本研究前面有談到動作 b_1 和其逆動作 b_1^{-1}，其動作結果相同，其特徵是兩兩交換，因此動作 B 家族的 8 個成員均存在逆動作，並且其逆動作操作結果與動作 B 相同，因此可以說動作 B^{-1}={ b_1^{-1}，c_1^{-1}，d_1^{-1}，e_1^{-1}，f_{11}^{-1}，f_{12}^{-1}，g_{11}^{-1}，g_{12}^{-1}}，因此也不需刻意記憶逆動作的操作，僅需記憶旗標方格的位置即可。

在動作 B 家族或是 B^{-1} 家族中，我們發現旗標軌跡落在橫軸中間上下兩列中，其對應操作動作成果亦落在橫軸中間上下兩列中，且其中的動作 b_1，c_1，d_1，e_1 四個動作結果，有兩個 XX 落在同一象限中，比較有利於相同顏色方格操作，並且此四個動作軌跡起點落在 8 字形軌跡的四個外角上。

在前面探討動作 b_1 時，我們發現動作 b_1 存在旋轉對稱性，亦即動作 b_4，b_3，b_2，但我們不需記憶，因為只需要將方陣旋轉 90~270 度並利用操作 b_1 即可完成，這裡整個動作 B 家族與動作 B^{-1} 家族亦同時存在旋轉對稱性，因此若是標的方格落在不同象限對應位置，只要將方陣或平板旋轉即可達到目的。

【問題研究五結論】

1. 動作 B 與動作 B^{-1} 家族旗標軌跡操作結果均是兩兩交換，因此動作 B 家族成員的旗標軌跡操作結果會與對應動作 B^{-1} 家族成員旗標軌跡的操作結果相同。
2. 動作 B 與動作 B^{-1} 家族旗標軌跡存在旋轉對稱性。
3. 在 2n×2n 方陣中，動作 B 與動作 B^{-1} 家族成員的旗標軌跡起點均落在 8 字形的六個角點上，旗標方格數均是(n-2)格。

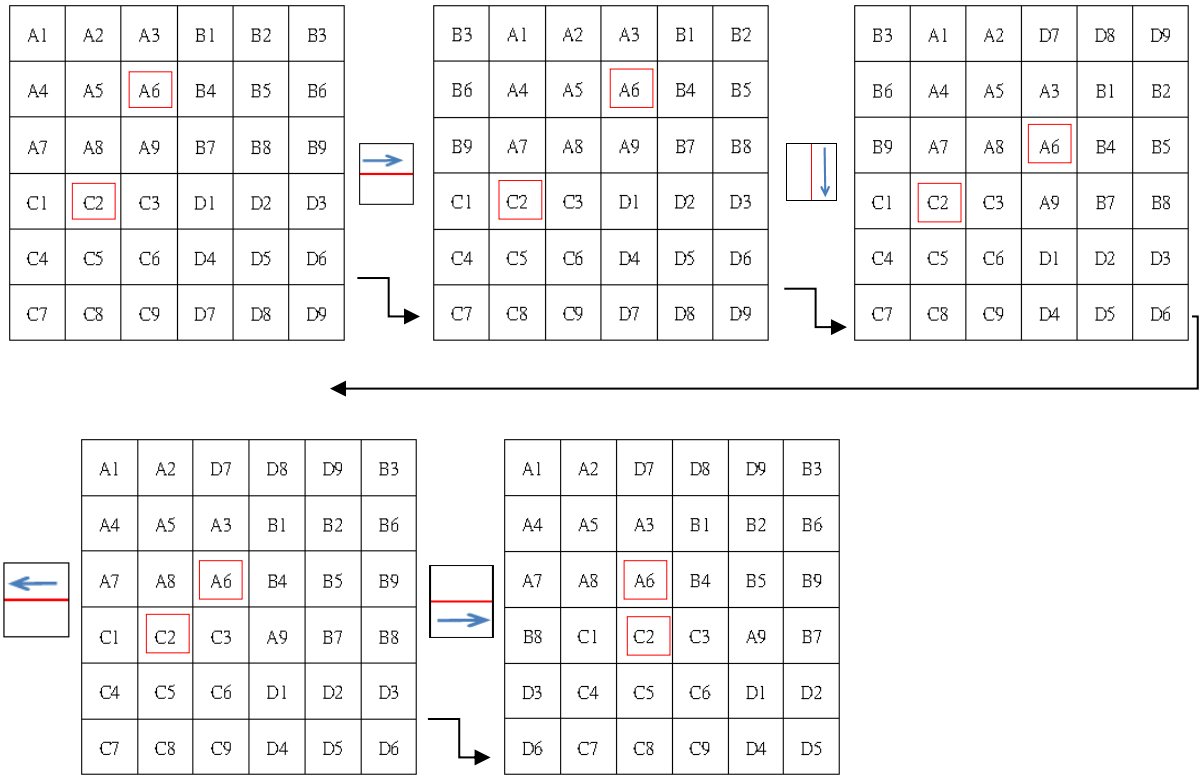
【問題研究六】探討動作 X 與動作 b_1 搭配完成第三種顏色

(a)介紹動作 X 以及連續動作 $X+b_1+X^{-1}$

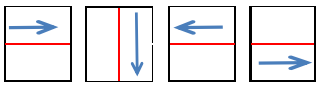
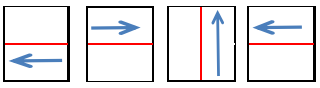
因為操作動作 b_1 時有特定位置，因此在操作前必須將標的方格移至特定位置(參考 b_1 動作結果)，並且在操作 b_1 前的滑動方法數較難預測，且每個狀況不盡相同，我們將此未知前置動作命名為 X，所以會有動作 X+動作 b_1+動作 X^{-1}，且必須確保操作過程中，前兩種已完成顏色方格未被破壞，以下舉實例說明。

實例一：

若是希望在 6x6 方陣中將 A6 與 C2 交換，並採用 b_1 動作使其交換，首先必須先設計一 X 動作使 A6 與 C2 進入原來方陣中 A9 與 C3 的位置，以下嘗試 X 的過程。



假設我們想要將 A6 與 C2 交換，而此時 A9 與 B7 恰是相同顏色方格，則可定義

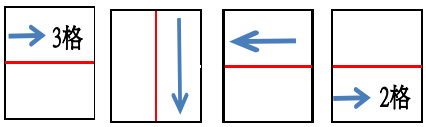
X=  X^{-1}=  為 X 之逆動作

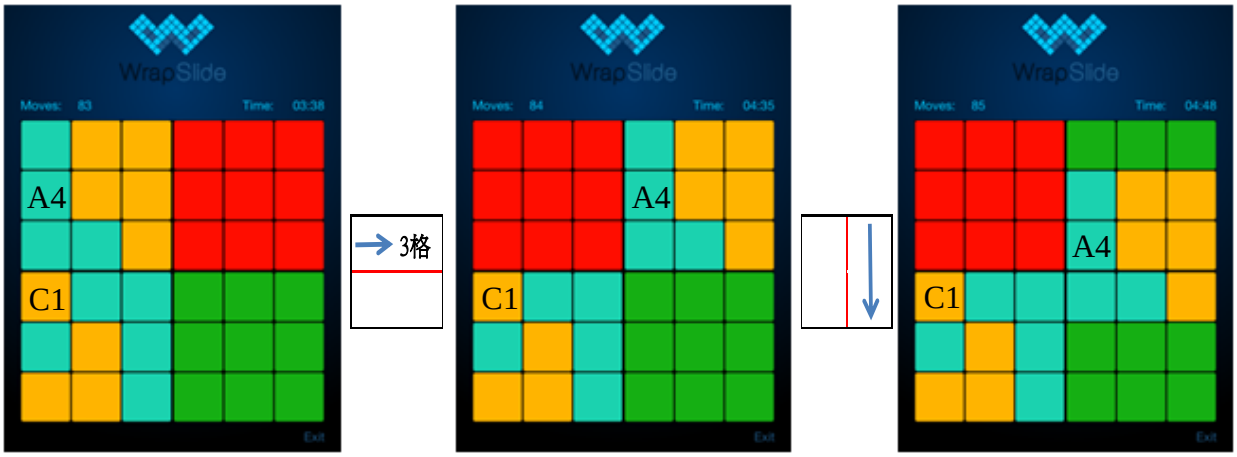
而操作 $X+b_1+X^{-1}$ 之後可得如右圖可發現 A6 確實和 C2 交換，並且 B7 與 A9 交換（若顏色相同，則察覺不出來）。

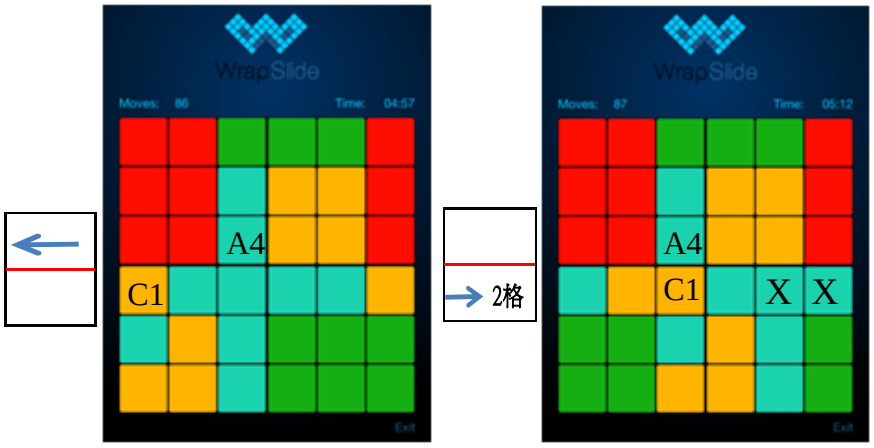
A1	A2	A3	B1	B2	B3
A4	A5	C2	B4	B5	B6
A7	A8	B7	A9	B8	B9
C1	A6	C3	D1	D2	D3
C4	C5	C6	D4	D5	D6
C7	C8	C9	D7	D8	D9

實例二：

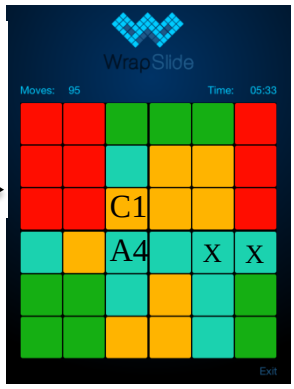
本例中已完成兩種顏色的狀況，此時若要將 C1 的黃色方格放入 A4 的位置，則可設計一動作 X

X= 

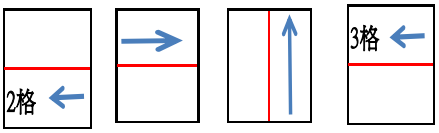


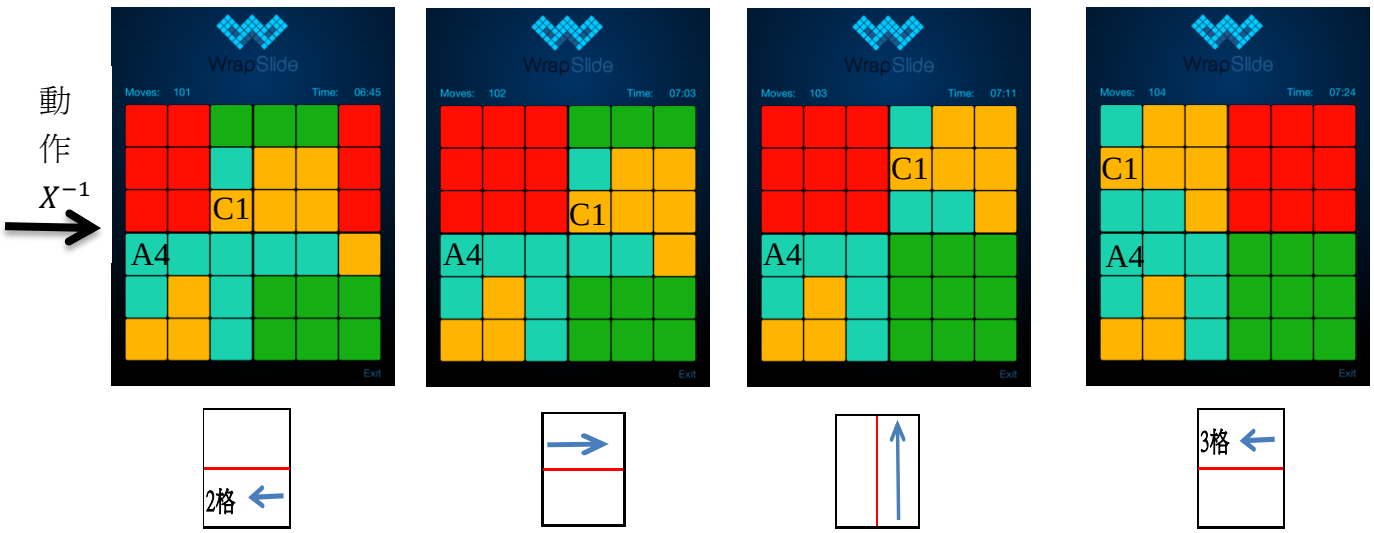


經動作 X 之後，C1 與 A4 可進入動作 b_1 的標的位置

動作 b1 

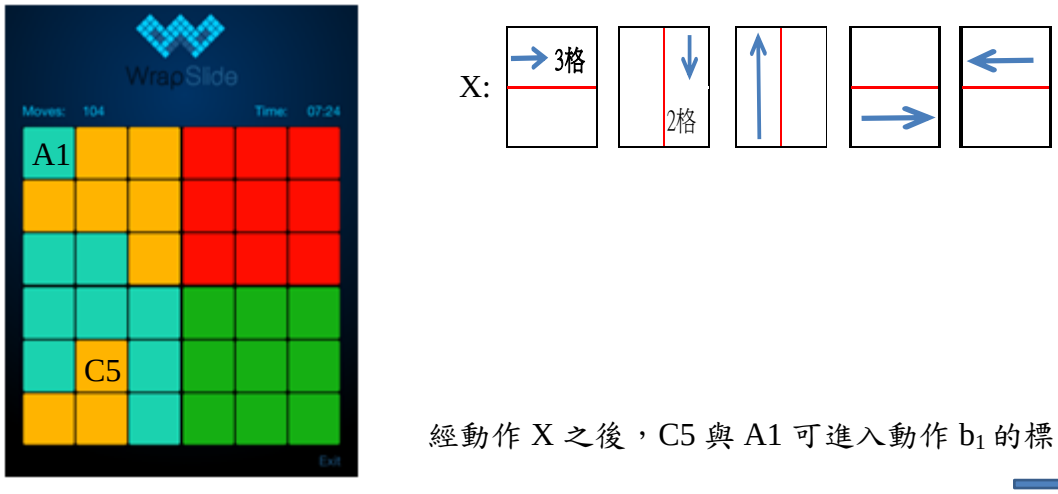
經 b_1 操作後，可得 A4 與 C1 交換，同時圖中藍色☒方格亦有交換，但兩藍色☒方格，因顏色相同，所以看不出來有交換的效果，最後進行X^{-1}動作，可以得到僅 C1 與 A4 的交換，其餘與一開始的狀態相同。

X^{-1}： 

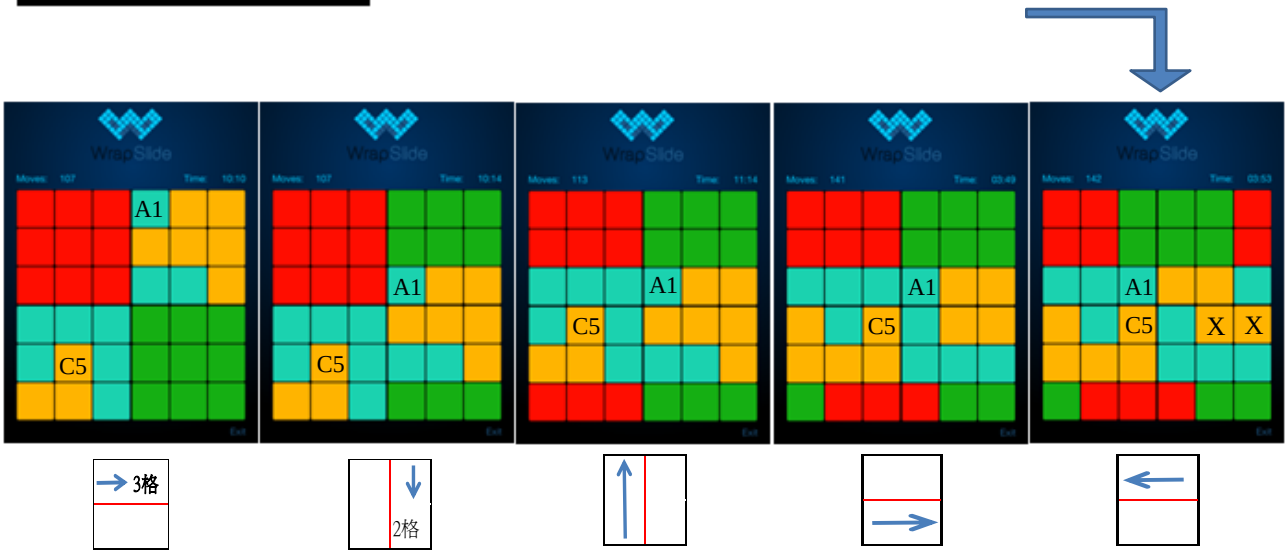


實例三：

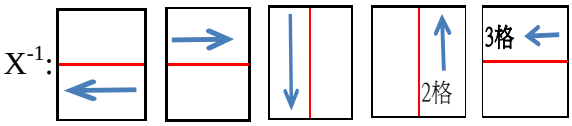
本例中已完成兩種顏色的狀況，此時若要將 C5 的黃色方格放入 A1 的位置，則可設計一動作 X 如下。

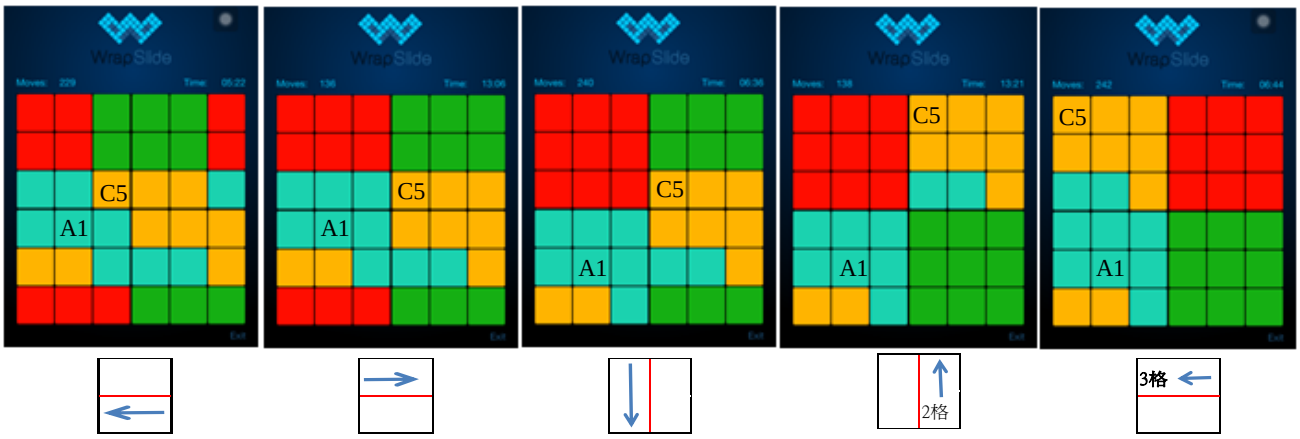


經動作 X 之後，C5 與 A1 可進入動作 b_1 的標的位置



經動作 b_1 操作後，可得 A1 與 C5 交換，同時圖中黃色方格亦有交換，但兩黃色方格，因顏色相同，所以看不出來有交換的效果，最後進行 X^{-1} 動作，可以得到僅 C5 與 A1 的交換，其餘與一開始的狀態相同。





在以上實例中，可發現當操作 b_1 時會有四個標的方格兩兩交換，以下圖二十七 6×6 方陣為例再次說明，操作動作 b_1 的標的方格為 A9 與 C3，以及 D2 與 D3，所以前面的例子中，任何想互相交換的方格，可移至 A9 與 C3 的位置，我們稱此前置動作為 X 動作，因為每次狀況不同，所以我們以 X 命名，但此時伴隨 X 動作之後，D2 和 D3 位置剛好為相同顏色方格，但是若碰到 D2 與 D3 顏色不同的狀況，此時，單是只想兩方格交換的意圖，便無法達成。

A1	A2	A3	B1	B2	B3
A4	A5	A6	B4	B5	B6
A7	A8	A9	B7	B8	B9
C1	C2	C3	D1	D2	D3
C4	C5	C6	D4	D5	D6
C7	C8	C9	D7	D8	D9

圖二十七、6×6 原始方陣

因此在第三種顏色的操作方法上，我們必須做出修正，既然前面已完成第一和第二種顏色方格，也就是在四個象限中，已完成二個象限為同色的狀況，此時剩下第三種顏色和底色

分佈在剩下的兩個象限中，此時我們選擇第四象限放置已完成的方格，固定為一種顏色，而且不移動，此時方陣只剩下上軸和左軸可移動，這裡只藉由上軸和左軸移動並設計出 X 動作，使得想要操作的標的方格進入 A9 與 C3 的位置，第四象限不動可確保 D2 與 D3 顏色必定相同，這時執行連續動作 $X+b_1+X^{-1}$ 即可達到只有兩方格交換的目的，而不會有前述碰到的困難。實例四中說明將第四象限固定顏色時的操作方式。

實例四：

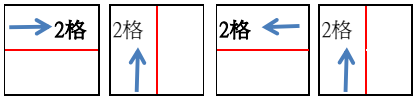
如下圖本例題中，想要將第三象限中 C7 的黃色方格放入第二象限中的 A3 底色方格，此時我們想要將第四象限綠色方格保持不動，只操作左軸與上軸來達到設計 X 動作的目的。

圖二十九、原始方陣

由圖二十九之動作圖可知，此連續四個動作，可以使 A3 和 C7 進入執行 b_1 動作的位置，並且保持第四象限方格顏色不變(圖中綠色部分)，因此上述四個連續動作可定義為 X

X= 

因此經動作 b_1 操作後，A3 和 C7 必須執行 X^{-1} 回復原來的狀態

X^{-1}= 

動作 b_1 

以上經由操作 $X+b_1+X^{-1}$ 之後達到 A3 與 C7 交換之目的，這個例子中 X 動作有 4 次滑動，因此 X^{-1} 動作也有 4 次滑動，再加上 b_1 動作有 8 次滑動，因此總共花費 16 次滑動達到目的。

(b)以連續動作 $X+b_1+X^{-1}$ 完成方陣中第三種顏色的方法數估算

經由修訂後第四象限固定的技巧，我們可以估算在 2n×2n 的方陣中，操作第三類顏色完成所需的滑動次數，若前兩種顏色已經完成，則剩下第三種顏色和底色分佈在兩象限中，因此我們可滑動此二象限，使其顏色分配對比愈大愈好，最差的狀況是每一象限中，第三種顏色方格與底色方格一樣多，此時要完成象限同色所需移動的方格數最多為$(1/2)n^2$，其中每一象限有 n^2 個方格，但有一半是底色，因此只需要操作這些底色方格變為第三種顏色即可。

前述中第四象限固定顏色的操作要將兩目標方格移入 A9 與 C3 的位置，且只依靠滑動上軸與左軸來完成，此時的 X 動作有兩個標的方格要操作，此兩標的方格可交互操作，每一標的方格最多滑動 3 次可到達目標，也就是到達 A9 或是 C3 位置且保持第四象限不動，因此 X 動作中雖然每個狀況不同，但最多不會超過 6 次滑動（前例中有些狀況常低於 6 次），因此以最高滑動次數估計動作 X 需 6 次滑動，動作 b_1 則需 8 次滑動，動作 X^{-1} 亦需 6 次滑動，因此動作 X+動作 b_1+動作 X^{-1} 需要 6+8+6＝20 次，最多有$(1/2)n^2$ 個方格需要交換。所以操作完成第三種顏色滑動次數最高為 $20\times(1/2)n^2=10n^2$ 次。此最高滑動次數估計說明了完成第三種顏色的滑動次數，將不會高於完成第二種顏色的最高滑動次數。

此最高滑動次數 $10n^2$ 可定義為完成第三種顏色有去有回的上帝之數(G_n with go-back)的上界(upper bound)，在考慮最混亂的狀態下，動作 X 的平均滑動次數多是介於 3~6 次之間，因此這裡將其下界定義為動作 X 需 3 次滑動+動作 b_1 則需 8 次滑動＋動作 X^{-1} 亦需 3 次滑動 =14 次滑動，且需要滑動$(1/2)n^2$ 個方格，所以下界(lower bound)定義為 $14\times(1/2)n^2=7n^2$，然此下界比起前面所提第一種顏色或是第二種顏色的下界而言，卻是約束比較不嚴謹(poor bound)。

最後關於 WrapSlide 中的 2n×2n 方陣，以『有去有回』的方式去完成各種難度的遊戲，其中在三顏色＋一底色，難度最高級的狀態下，估算其『有去有回』之上帝之數(G_n with go-back)時，上界為$[7n^2+(n-1)]+[10n^2+(n-1)]+10n^2=27n^2+(2n-2)$，而其下界為$[6n^2+(n-1)]+[7n^2+(n-1)]+7n^2 = 20n^2+(2n-2)$。

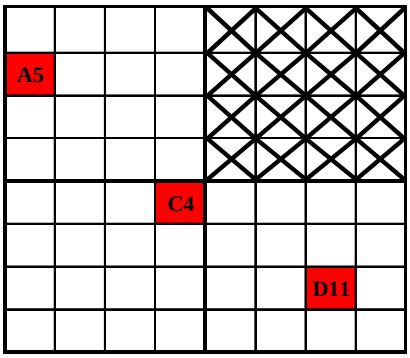
【問題研究六結論】

1. 在 2n×2n 的方陣中，以 Xb_1X^{-1} 的方式並保持第四象限完整，操作完成第三種顏色最高滑動次數不超過 $10n^2$ 次，完成三顏色＋一底色難度最高級的狀態下，滑動總次數將不超過 $27n^2+(2n-2)$次。
2. 在 2n×2n 的方陣中，以『有去有回』的基礎概念動作去完成第三種顏色，其滑動次數卻是低於完成第二種顏色的滑動次數。
3. 在 2n×2n 的方陣中，在三顏色＋一底色難度最高級的狀態下，其有去有回之上帝之數(G_n with go-back)具有上、下界分別為 $27n^2+(2n-2)$與 $20n^2+(2n-2)$。

【問題研究七】無旗標方格的操作法--兩軸切割法

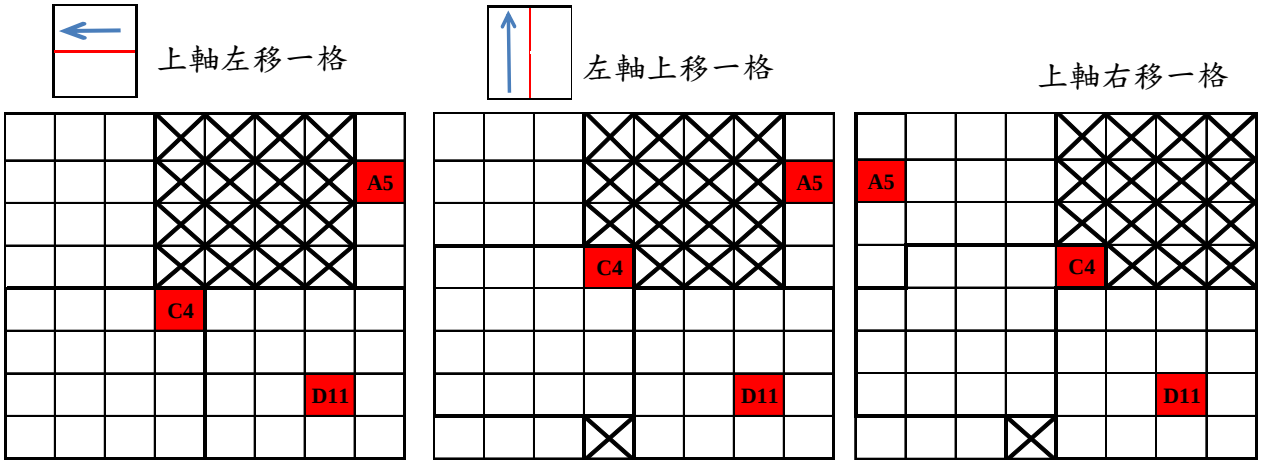
前面的問題研究中，都是以『有去有回』並且有旗標方格操作的方式達成任務，例如動作 a_1 和動作 b_1、c_1 等等 B 家族成員，這種方法雖然不是最快的方式，但是因為有旗標軌跡可以遵循，所以只要將手放置在操作該動作的旗標方格上即可輕易達成，而且容易以軌跡印象去記憶。但這裡將研究以一種 x 軸、y 軸兩軸切割的方式，蒐集相同型性質的方格放入第一

象限中，其中 C4 的位置是預置方格位置，它是第一象限的對頂角，也就是必須先將標的方格滑動至此。



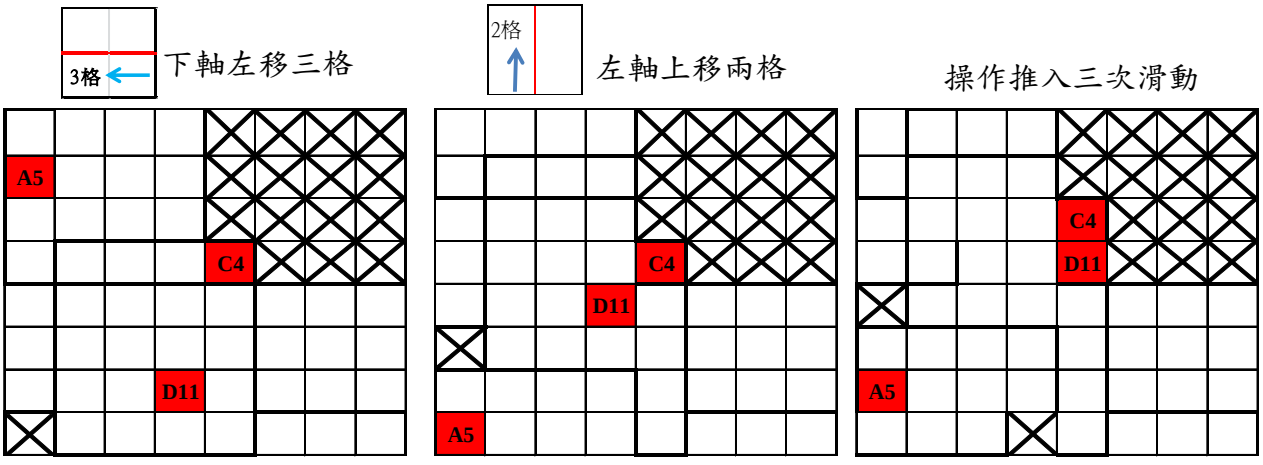
圖三十、8×8 方陣中將 C4、A5、D11 放入第一象限

當預置方格準備好之後，便可將標的方格推入第一象限中，一共需要三次滑動執行推入之動作，x、y 軸可以想成刀片，將第一象限以上軸往左推時，可以利用左軸切開第一象限並將 C4 推入，當上軸往右推回時 C4 成為第一象限的左下角。



圖三十一、8×8 方陣中將 C4 放入第一象限的推移過程

以上圖三十一中的三個動作，說明完成推入一個方格進入第一象限中，接下來要將標的方格 D11 滑至預備位置，如下圖中，經由下軸左移三格，即可進入預備位置，然後操作上述所提三個動作推入第一象限中。



圖三十二、8×8 方陣中將 D11 放入第一象限的推移過程

經由滑動推入第一象限的準備，預備位置為第一象限的對頂角方格，滑動準備次數為兩次，推入需要滑動三次，因此在 2n×2n 的方陣中，完成一象限之顏色最多需要滑動$(2+3)n^2=5n^2$次，常有狀況比此預測數少，例如兩方格連續接在一起時便可以降低滑動次數，此方式明顯比用動作 a_1 完成一象限方格所需滑動次數少，此外，這個方法也同時適用於蒐集第二、三、四象限的單一顏色操作，其預備位置均是該象限的對頂角，滑動次數與第一象限相同，因此也可以說兩軸切割法具有旋轉對稱性，但此法無法順利完成第三種顏色方格操作。

【問題研究七結論】

在 2n×2n 方陣且無旗標方格的操作中，可藉由兩軸切割法將同一種顏色方格推入第一象限，其滑動次數最多不超過 $5n^2$ 次。

柒、結論：

我們對滑動方塊數學的研究進行推理與推盤驗證後發現：

一、當我們執行『有去有回』的動作時，可以找到旗標方格，其軌跡遵循我們的滑動動作，當執行順時針的動作 a_1 時，會使環繞旗標 G 的方格逆時針移動一格，反之若執行逆時針的動作 a_1^{-1} 時，將使環繞旗標 G 的方格順時針移動一格。

二、a_4 是 a_1 的旋轉 90 度對稱，也就是將方陣順時針旋轉 90 度後，a_1 就變成 a_4，而且 a_1 中的 4 個步驟，同時也是依順時針轉 90 度，而變成 a_4 中的 4 個步驟。同理，將方陣轉 180 度，a_1 轉成 a_3；方陣轉 270 度，a_1 轉成 a_2 所以我們可以統稱 a_1、a_2、a_3、a_4 為動作 A 的家族。我們不需要記憶 a_4、a_3、a_2 的動作，只需要將方陣中想要操作的象限，經由方陣旋轉（**平板旋轉**）至第一象限，然後操作動作 a_1 就可以達到環繞參考旗標點，逆時針移動一格的目的。而操作 a_1^{-1}（a_1 的逆動作），則可環繞參考旗標點順時針移動一格。

三、在 2n×2n 方陣中，操作 a_1 或是 a_1^{-1} 前，使同色方格(標的方格)進入預備位置所需滑動次數為 0~3 次，以此方法搭配 a_1 或 a_1^{-1} 連續操作可以完成第一象限方格為相同顏色。

四、在 2n×2n 的方陣中，操作動作 a_1 的旗標方格會有$(n-1)^2$格，以動作 a_1 的方式完成第一象限顏色方格，所需滑動次數最多不超過 $7n^2+(n-1)$次。

五、在 2n×2n 的方陣中，以動作 a_1 的方式完成第一象限顏色方格，我們定義出上帝之數 **G_n with go-back** 適用於所有方陣，其上界(upper bound)與下界(lower bound)分別為 $7n^2+(n-1)$與 $6n^2+(n-1)$。

六、以操作動作 a_1 的方式，完成 2n×2n 方陣中第二種顏色，所需滑動次數不超過 $10n^2+n-1$ 次。

七、操作動作 a_1 完成 2n×2n 方陣中第二種顏色方格，其有去有回的上帝之數(G_n with go-back)具有上、下界分別為 $10n^2+(n-1)$與 $7n^2+(n-1)$。

八、動作 B 與 B^{-1} 家族旗標軌跡操作結果均是兩兩交換，因此動作 B 家族成員的旗標軌跡操作結果會與對應動作 B^{-1} 家族成員旗標軌跡的操作結果相同。

九、在 2n×2n 方陣中，動作 B 與動作 B^{-1} 家族成員的旗標軌跡起點均落在 8 字形軌跡的六個角點上，旗標方格數均是(n-2)格，旗標軌跡存在旋轉對稱性。

十、在 2n×2n 的方陣中，以 Xb_1X^{-1} 的方式並保持第四象限完整，操作完成第三種顏色最高滑動次數不超過 $10n^2$ 次，在難度最高級的狀態下完成三顏色＋一底色，總滑動次數必不超過 $27n^2+(2n-2)$次。

十一、在 2n×2n 的方陣中，在三顏色＋一底色難度最高級的狀態下，其『有去有回』之上帝之數(G_n with go-back)具有上、下界分別為 $27n^2+(2n-2)$與 $20n^2+(2n-2)$。

十二、在 2n×2n 的方陣中，以『有去有回』的基礎概念動作去完成第三種顏色，其滑動次數卻是低於完成第二種顏色的滑動次數。

十三、在 2n×2n 方陣且無旗標方格的操作中，可藉由兩軸切割法將同一種顏色方格推入第一象限，其滑動次數最多不超過 $5n^2$ 次，此法操作次數比動作 a_1 來的低。

捌、參考資料及網路資源：

（一）WrapSlide 應用程式專頁・臉書社群・取自 https://www.facebook.com/wrapslide

（二）**陳宏賓**（2015 年 4 月 8 日）・利用數學解開手機益智遊戲 WrapSlide－只用一招! ・UniMath・取自 https://sites.google.com/a/g2.nctu.edu.tw/unimath/2015-04/wrapslideformula

（三）Apple iOS 系統手機平板用戶 WrapSlide App 下載 https://itunes.apple.com/app/wrapdlide/id795712935?mt=8

（四）Android系統手機平板用戶WrapSlide App下載 https://play.google.com/store/apps/details?id=com.wrapslide.android.wrapslide

（五）**游森棚**，魔術方塊的上帝之數，科學月刊，491期，p.820，2010年11月。

（六）God’s number is 20, http://www.cube20.org

（七）What is God’s Number for WrapSlide?， http://cubezzz.dyndns.org/drupal/?q=node%2Fview%2F540

研究成果自評：

一、本研究創新發現『有去有回』的動作 a_1 與動作 b_1 的**旗標方格**與**旗標軌跡**的存在，這是參考文獻中尚未敘述的，因此本研究建立了**旗標方格的計算公式**。

二、本研究由動作 b_1(文獻二中有記載敘述 b_1)建立出旗標軌跡，進而發現了其 **8 字型軌跡**，並建立出由 8 字形的六個角點延伸出 8 種動作$\{b_1,c_1,d_1,e_1,f_{11},f_{12},g_{11},g_{12}\}$，我們將其命名為 **B 家族**成員。

三、本研究發現了**旋轉對稱**性。

四、本研究為各種難度的玩法，估算出其該難度下的**必要滑動次數的最大值**，並將其數學公式化，以及參考魔方的上帝之數定義，我們以『有去有回』限制研究方法應用在 2n×2n 的方陣中，定義出『有去有回』的上帝之數(G_n with go-back)的上界(upper bound)與下界(lower bound)。

五、以上研究建立之公式均可適用至**無限大之方陣**，並不侷限於 WrapSlide App 中之最大的 8×8 方陣。

附件一 (摘錄自參考文獻六)

A History of God's Number

By 1980, a lower bound of 18 had been established for God's Number by analyzing the number of effectively distinct move sequences of 17 or fewer moves, and finding that there were fewer such sequences than Cube positions. The first upper bound was probably around 80 or so from the algorithm in one of the early solution booklets. This table summarizes the subsequent results.

Date	Lower bound	Upper bound	Gap	Notes and Links
July, 1981	18	52	34	Morwen Thistlethwaite proves 52 moves suffice.
December, 1990	18	42	24	Hans Kloosterman improves this to 42 moves.
May, 1992	18	39	21	Michael Reid shows 39 moves is always sufficient.
May, 1992	18	37	19	Dik Winter lowers this to 37 moves just one day later!
January, 1995	18	29	11	Michael Reid cuts the upper bound to 29 moves by analyzing Kociemba's two-phase algorithm.
January, 1995	20	29	9	Michael Reid proves that the "superflip" position (corners correct, edges placed but flipped) requires 20 moves.
December, 2005	20	28	8	Silviu Radu shows that 28 moves is always enough.
April, 2006	20	27	7	Silviu Radu improves his bound to 27 moves.
May, 2007	20	26	6	Dan Kunkle and Gene Cooperman prove 26 moves suffice.
March, 2008	20	25	5	Tomas Rokicki cuts the upper bound to 25 moves.
April, 2008	20	23	3	Tomas Rokicki and John Welborn reduce it to only 23 moves.
August, 2008	20	22	2	Tomas Rokicki and John Welborn continue down to 22 moves.
July, 2010	20	20	0	Tomas Rokicki, Herbert Kociemba, Morley Davidson, and John Dethridge prove that God's Number for the Cube is exactly 20.

附件二（摘錄自參考文獻七）

What is God's Number for WrapSlide?

Submitted by **Alewyn Burger** on Wed, 10/22/2014 - 08:36.

I developed a slide-puzzle called WrapSlide that reminds of Rubik's Cube. I am interested in determining God's number for WrapSlide. I think my initial approach is too naive and may be I should leave it to the experts.

First let me describe WrapSlide:
The main puzzle is a 6x6 grid of colored tiles which are separated into four quadrants of 3x3 tiles. When it is unmixed all the tiles in a quadrant have the same colour. A move consists of sliding either the top, bottom, left or right two quadrants of tiles 1 to 5 units horizontally or vertically. Stated differently, a move consists of sliding either the top, bottom, left of right half (consisting of 3 rows or 3 columns) relative to the other half, thus giving 4x5 possible moves to choose from. As with Rubik's cube the puzzle is to return it to its unmixed state after it is scrambled. (For the unmixed state we don't care which color goes into which quadrant)

Some info:
WrapSlide has roughly half the number of configurations that Rubik's Cube have: 36!/(9!9!9!9!) = 21452752266265320000. (It was proven by two Dutch students that all states are reachable from the unmixed state)

God's number for WrapSlide is at least 21. I think I was lucky to find this state that can't be solved in 20 moves:

0 0 0 3 0 3
2 3 2 3 1 2
0 1 0 3 0 3
2 1 2 1 1 1
2 0 3 3 2 3
2 1 2 1 0 1

An upper bound is 31, because any one color can always be fixed in 12 moves (or less), and the sub-puzzle of fixing 3 colours doing (say) only left and lower moves can always be done in 19 moves. Thus giving 12+19 as an upper bound. This is a very poor upper bound, but it is a start.

【評語】080407

探討手機遊戲「滑動方塊」連續滑動之動作，十分有趣，作品說明清楚，值得肯定，惟應強調出本研究與相關參考資料之異同。