



## โครงการ

# การเรียนการสอนเพื่อเสริมประสบการณ์

ชื่อโครงการ การขยายรูปแบบกลไฟของมาร์ตินการ์ดเนอร์

An extension of Martin Gardner's card trick

ชื่อนิสิต นางสาวธิดา เหลืองวรัญญู 583 35527 23

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์

สาขาวิชา คณิตศาสตร์

ปีการศึกษา 2563

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

การขยายรูปแบบกลไฟของมาร์ตินการ์ดเนอร์

นางสาวอริษา เหลืองวรัญญู

โครงการนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต  
สาขาวิชาคณิตศาสตร์ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2563

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย



AN EXTENSION OF MARTIN GARDNER'S CARD TRICK

Ms. Athisa Laungvarunyoo

A Project Submitted in Partial Fulfillment of the Requirements  
for the Degree of Bachelor of Science Program in Mathematics

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University


Academic Year 2020

Copyright of Chulalongkorn University

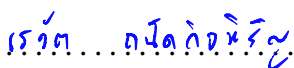
Project Title      AN EXTENSION OF MARTIN GARDNER'S CARD TRICK  
By                      Ms. Athisa Laungvarunyoo  
Field of Study      Mathematics  
Project Advisor     Raywat Tanadkithirun, Ph.D.

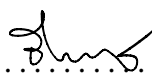
---


Accepted by the Department of Mathematics and Computer Science Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Bachelor's Degree in 2301499 Senior Project

.....  .....  
Dean of the Department of  
Mathematics and Computer  
Science  
(Professor Kritsana Neammanee, Ph.D.)

#### PROJECT COMMITTEE

.....  ..... Project Advisor  
(Raywat Tanadkithirun, Ph.D.)

.....  ..... Examiner  
(Assistant Professor Teeradej Kittipassorn, Ph.D.)

.....  ..... Examiner  
(Professor Wacharin Wichiramala, Ph.D.)

อธิชา เหลืองวรัญญ: การขยายรูปแบบกลไฟของมาร์ตินการ์ดเนอร์. (AN EXTENSION OF MARTIN GARDNER'S CARD TRICK) อ.ที่ปรึกษาโครงการหลัก : อ.ดร.เรวัต ถนัดกิจหิรัญ, 63 หน้า.

กลไฟของมาร์ตินการ์ดเนอร์เป็นกลไฟคณิตศาสตร์ที่เก่าแก่ ที่ใช้ความรู้ในเรื่องเลขฐาน และการจัดเรียงมาสร้างกลไฟ วิธีการเล่นกลไฟ คือ ให้ผู้สังเกตเลือกจำไฟที่ต้องการ หลังจากนั้นแบ่งไฟเป็นหลาย ๆ กอง แล้วผู้สังเกตชี้กองที่มีไฟที่เลือก ทำเช่นนี้หลาย ๆ ครั้ง สุดท้ายไฟที่เลือกจะสามารถไปปรากฏในตำแหน่งที่ต้องการ เนื่องจากกลไฟของการ์ดเนอร์และกลไฟอื่นที่ได้ถูกขยายตามมานั้นยังไม่ครอบคลุมในบางกรณี จึงมีความน่าสนใจที่จะศึกษาเงื่อนไขที่ทำให้กลไฟมีผลเฉลยและวิธีการเล่นแบบต่าง ๆ ของปัญหานั้น กลไฟนี้มี 3 พารามิเตอร์ ได้แก่ จำนวนกองที่แบ่งไฟ จำนวนไฟในแต่ละกอง และจำนวนรอบที่จัดเรียงไฟ ในโครงการนี้เราได้สร้างโปรแกรมสำเร็จรูปที่สามารถตรวจสอบปัญหาของการ์ดเนอร์สำหรับชุดพารามิเตอร์ที่กำหนดว่ามีผลเฉลยหรือไม่มีผลเฉลย ยิ่งไปกว่านั้นสำหรับชุดพารามิเตอร์ที่มีผลเฉลยเราจะได้วิธีการเล่นทั้งหมดสำหรับปัญหานั้นด้วย เราได้ให้ข้อคาดการณ์ของเงื่อนไขความสัมพันธ์ของพารามิเตอร์ที่ทำให้การเล่นกลไฟแบบการ์ดเนอร์มีผลเฉลยหรือไม่มีผลเฉลยอีกด้วย


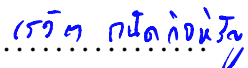
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	ลายมือชื่อนิสิต	<u>.....</u> อธิชา เหลืองวรัญญ
สาขาวิชา	คณิตศาสตร์	ลายมือชื่ออ.ที่ปรึกษาโครงการหลัก	<u>.....</u> เรวัต ถนัดกิจหิรัญ
ปีการศึกษา	2563		

## 5833552723: MAJOR MATHEMATICS

KEYWORDS: MARTIN GARDNER'S CARD TRICK / THE 27 CARD TRICK

ATHISA LAUNGVARUNYOO : AN EXTENSION OF MARTIN GARDNER'S CARD TRICK. ADVISOR : RAYWAT TANADKITHIRUN, Ph.D.,  
63 pp.

Martin Gardner's card trick is a classic mathematical card trick, using number base and ordering to create the magic. This trick lets the inspector memorize a random card and point, after dealing cards into many piles, at the pile that has that card; the process of dealing cards and pointing a pile is repeatly performed; then, the chosen card will finally be at the given position. Since Martin Gardner's card trick and other related card tricks may not be practicable in some situations, it is interesting to find the conditions that make this card trick solvable as well as the ways to perform the trick. This card trick has 3 parameters: the number of piles, the number of cards in each pile and the number of rounds to restack the piles. In this project, we provide an instant program which can tell whether a Gardner's problem with given parameters is solvable or unsolvable. Moreover, for a solvable problem, all possible ways to perform the Gardner's trick are provided. Some anticipated conditions on the parameters that make the Gardner's problem solvable or unsolvable are also given.

Department:	Mathematics and Computer Science	Student's Signature	
Field of Study:	Mathematics	Advisor's Signature	
Academic Year:	2020		

## Acknowledgements

First of all, I would like to express my sincere thanks to my project advisor, Raywat Tanadkithirun, Ph.D., for his invaluable help and constant encouragement throughout the course of this research. I am most grateful to his teaching and suggestion. I would not have achieved this far and this project would not have been completed without all the support that I always have received from him. I also would like to thank my project committees, Assistant Professor Teeradej Kittipassorn, Ph.D. and Professor Wacharin Wichiramala, Ph.D., for their comments and suggestions. In addition, I am grateful to the teachers in Department of Mathematics and Computer Science for suggestions and all of their help.

Finally, I most gratefully acknowledge my parents and my dear friends for all of their support throughout the period of this project.

# CONTENTS

	<b>Page</b>
<b>Abstract (Thai)</b> . . . . .	<b>iv</b>
<b>Abstract (English)</b> . . . . .	<b>v</b>
<b>Acknowledgements</b> . . . . .	<b>vi</b>
<b>Contents</b> . . . . .	<b>vii</b>
<b>List of Tables</b> . . . . .	<b>ix</b>
<b>List of Figures</b> . . . . .	<b>x</b>
<b>1 Introduction</b> . . . . .	<b>1</b>
<b>2 Preliminaries</b> . . . . .	<b>4</b>
2.1 Number Bases . . . . .	4
2.2 Codes for Performing . . . . .	5
2.3 Variables and Conditions . . . . .	6
2.4 Wolfram Language . . . . .	9
<b>3 Gardner's trick program</b> . . . . .	<b>10</b>
3.1 Conditions and Main Result for Our Program . . . . .	11
3.2 Our Proposed Program . . . . .	15
3.3 Some Experiments . . . . .	24
<b>4 Results</b> . . . . .	<b>28</b>
4.1 Fix $M$ and $K$ , run $N$ . . . . .	28
4.2 Fix $M$ and $N$ , run $K$ . . . . .	31
4.3 Fix $K$ , run $M$ and $N$ . . . . .	35
<b>5 Conclusion and Discussion</b> . . . . .	<b>46</b>

<b>References</b> . . . . .	<b>48</b>
<b>Appendix</b> . . . . .	<b>49</b>
<b>Appendix A Project Proposal</b> . . . . .	<b>49</b>
A.1 Background and Rationale . . . . .	49
A.2 Objectives . . . . .	50
A.3 Scope . . . . .	50
A.4 Project Activities . . . . .	50
A.5 Durations . . . . .	51
A.6 Benefits . . . . .	52
A.7 Equipment . . . . .	52
<b>Biography</b> . . . . .	<b>53</b>

# LIST OF TABLES

<b>Table</b>	<b>Page</b>
2.1 Changing 125 to base 3 . . . . .	4
2.2 Changing 22 to base 4 . . . . .	5
2.3 Codes for performing the original Gardner's card trick . . . . .	5
2.4 Code for $M = 2, N = 6, K = 4$ . . . . .	8
2.5 All initial positions are performed with codes $0000_2$ , $0001_2$ , and $0010_2$ , respectively from left to right, for the Gardner's problem with parame- ters $M = 2, N = 6, K = 4$ . . . . .	8
2.6 Code for $M = 3, N = 4, K = 3$ . . . . .	8
2.7 Code for $M = 4, N = 3, K = 2$ . . . . .	9



# LIST OF FIGURES

<b>Figure</b>	<b>Page</b>
3.1 An example of performing: before dealing cards [8] . . . . .	12
3.2 An example of performing: after dealing cards [8] . . . . .	12
3.3 A example of performing: after performing the trick one round [8] . . . . .	13
3.4 The flow chart for our program . . . . .	16
3.5 Input parameters $M, N, K$ . . . . .	17
3.6 Pop-up window(1) . . . . .	17
3.7 Pop-up window(2): input window . . . . .	17
3.8 Check the conditions on parameters . . . . .	18
3.9 Detecting all inputs are not an integer . . . . .	19
3.10 Detecting $M$ or $N$ or $K \leq 0$ . . . . .	19
3.11 Detecting $MN \leq M^K$ . . . . .	19
3.12 Finding all codes . . . . .	19
3.13 Finding all usable codes . . . . .	20
3.14 Usable code and unusable code checking . . . . .	21
3.15 Match the code to the desired position(1) . . . . .	22
3.16 Match the code to the desired position(2) . . . . .	22
3.17 Show a resutling table . . . . .	23
3.18 Solvable pop-up window . . . . .	24
3.19 Unsolvable pop-up window . . . . .	24
3.20 Extra program . . . . .	25
3.21 Fix $M$ and $K$ , run $N$ program . . . . .	26
3.22 Fix $M$ and $N$ , run $K$ program . . . . .	27
3.23 Fix $K$ , run $M$ and $N$ program . . . . .	27

# Chapter I

## INTRODUCTION

The *Gergonne 3-pile problem* [5] is a mathematical card trick, which was proposed in 1813. Hold a deck of 27 cards and the problem is to keep track of the location of the card that was selected in the beginning. At first, a spectator selects any card in a deck, memorizes it and shuffles the deck as much as desired. The magician create three face-up piles of nine cards by dealing the top cards respectively onto the left, middle, then right pile and repeating until all the cards are gone. Now, he gets 3 piles of cards. Next, the participant points to which pile contains the chosen card. After that, the magician gather all those piles and put the chosen pile to the middle. The problem involves distributing these cards with this same process 3 times. Finally, the magician can make the middle of the deck to be the chosen card.

In 1895, Dickson generalized Gergonne 3-pile problem, called *Gergonne  $n$ -pile problem* [3]. We can perform the trick with any  $n^n$  cards by doing the similar procedure as the Gergonne 3-pile problem. The pack is dealt into  $n$  piles of  $n^{n-1}$  cards each and we do the restacking procedure  $n$  times. Not surprisingly, the chosen card shall appear at the middle of the deck.

During the past century the trick was included in books on recreational mathematics. In *Mathematics, Magic and Mystery*, Martin Gardner evolved Gergonne 3-pile problem to the *27-card trick* [4]. In his trick, it is similar to the Gergonne 3-pile problem, but at this time, the spectator can choose the position, say  $x$ , between 1 and 27 at which he wants the chosen card to finally appear. After performing the Gergonne procedure with 3 piles and 3 times assembly restacked in any order the magician wants, the  $x^{th}$  card of the deck is revealed to be the chosen card. In general, we can deal a pack of any  $n^k$  cards into  $n$  piles and repeat this procedure  $k$  times; we can place the chosen card at any desired position in the deck [2].

In 2010, Bolker generalized this trick by using radix sort [1]. Using a mixed radix conversion, you can do Gardner's trick with a deck of any number of cards you wish by dealing the deck into factors of the number of cards in the deck. For example, for a deck of  $54 = 6 \times 3 \times 3$  cards, we can perform by dealing the cards into 6 piles for the first round, and 3 piles for the second and third rounds. You can also perform the trick by dealing the cards into 3, 9 and 2 piles for the first, second and third rounds, respectively.

In this project, we call the problem that we finally put the chosen card to the middle "Gergonne's problem", the problem that we can put the chosen card to the position that the spectator desires "Gardner's problem", to his honor, the problem that can be performed by dealing unequal cards in each round "Bolker's problem". Note that, for Gergonne's problem and Gardner's problem, each pile must have the same height in every single round.

It is interesting to find the deck with the size other than  $n^k$  cards that we can perform Martin Gardner's card trick without mixed radix method. For example, we can perform the Bolker's trick with 52 cards by dealing the deck into 4 piles and 13 piles. However, it seems hard to deal the deck into 13 piles and it does not sound like magic. Thus, as a real-world performance, we want to deal cards into a fixed number of piles, say 4. Now, we want to know how many times that we need to do the restacking procedure to guarantee that we can finally put the chosen card from the spectator to the chosen position.

In this project, we want to provide an instant program which can tell whether a Gardner's problem with given parameters which are the number of piles, the number of cards in each piles and the number of rounds to restack the piles is able to be performed or not. If it is, all possible ways to perform the Gardner's trick are also provided. We hope that this instant program can be a tool to help study the Gardner's problem.

In Chapter II, some preliminaries and notations that we use are provided. The algorithm for finding the solution, and Gardner's trick solution instant program are provided in Chapter III. We also show the result of some interesting cases in Chapter IV. Finally, conclusion and discussion are provided in Chapter V.

# Chapter II

## PRELIMINARIES

In this chapter, we give some illustrations of the Gardner's card trick. We also provide definitions and notations that we use throughout this work.

### 2.1 Number Bases

The most commonly used number system is decimal, known as base 10. The place value of each digit is a power of ten:  $10^0, 10^1, 10^2, 10^3$ , and so on, which means that, starting from the right most digit, the first digit is worth 1, the  $2^{nd}$  digit worth 10, the  $3^{rd}$  digit worth 100, the  $4^{th}$  digit worth 1000 and so on. The base number is written as a subscript. For example,  $23_7$  is read as 23 base 7, which is 17 in base 10. Normally, we do not need to write the base number for base 10.

In the Gardner's problem with 27 cards, we need to convert from a regular number in base 10 to a ternary number. The place value of each digit of a ternary number is a power of three, which is  $3^0, 3^1, 3^2, 3^3$ , and so on. In the first step, divide your desired number by 3 to get a remainder. Next, repeat this procedure by dividing the previous quotient by 3. Continue repeating this division process until your quotient number becomes zero. The answer is the remainders collected from the last remainder to the first one. Table 2.1 shows an example of converting 125 in base 10 to  $11122_3$ .

**Table 2.1:** Changing 125 to base 3

Division	Quotient	Remainder (Digit)	Place Value
$\frac{125}{3}$	41	2	$3^0$
$\frac{41}{3}$	13	2	$3^1$
$\frac{13}{3}$	4	1	$3^2$
$\frac{4}{3}$	1	1	$3^3$
$\frac{1}{3}$	0	1	$3^4$

We can also convert a number from base 10 to any base  $n$ . The place value of each digit is  $n^0, n^1, n^2, n^3$  and so on. We can follow the same process for a ternary number, but we only need to change the divisor from 3 to  $n$ . For example, we can convert 22 in base 10 to  $112_4$ . Table 2.2 shows details of this example.

**Table 2.2:** Changing 22 to base 4

Division	Quotient	Remainder (Digit)	Place Value
$\frac{22}{4}$	5	2	$4^0$
$\frac{5}{4}$	1	1	$4^1$
$\frac{1}{4}$	0	1	$4^2$

## 2.2 Codes for Performing

Performing the original 27-card Gardner’s trick depends on base three arithmetic [6]. First, we subtract the desired position chosen from the spectator by 1 and convert that resulting number into a three digit number in base 3. We will call this base-3 number a “code”. Next, we decode each digit in the code starting from right to left to perform the Gardner’s trick in each round by placing the indicated pile (from the spectator) in the interpreted position. If the digit is 0, the chosen pile goes on top (none above it); if it is 1, the chosen pile goes at the middle (one stack above it); if it is 2, the chosen pile goes at the bottom (two stacks above it). Performing the trick according to this strategy will finally bring the chosen card to the desired position. Table 2.3 shows all corresponding pairs of desired positions and codes for performing.

**Table 2.3:** Codes for performing the original Gardner’s card trick

Position	1	2	3	4	5	6	7	8	9
Code	$000_3$	$001_3$	$002_3$	$010_3$	$011_3$	$012_3$	$020_3$	$021_3$	$022_3$
Position	10	11	12	13	14	15	16	17	18
Code	$100_3$	$101_3$	$102_3$	$110_3$	$111_3$	$112_3$	$120_3$	$121_3$	$122_3$
Position	19	20	21	22	23	24	25	26	27
Code	$200_3$	$201_3$	$202_3$	$210_3$	$211_3$	$212_3$	$220_3$	$221_3$	$222_3$

### 2.3 Variables and Conditions

**Definition 2.3.1.** For a Gardner's problem, we define parameters  $M, N, K$  as follows:

$M$  is the number of dealing piles in each round,

$N$  is the number of cards in each dealing pile, and

$K$  is the number of dealing rounds.

**Definition 2.3.2.** For a Gardner's problem with given parameters  $M, N, K$ , let  $x$  be the desired position chosen by a spectator at the beginning of the performance.

Note that all possible positions of  $x$  are  $1^{st}, 2^{nd}, \dots, MN^{th}$ . We can convert  $x$  into a code for performing by subtracting it by 1 and converting the result into a  $K$  digit number in base  $M$ . Note that this can be done only when  $MN \leq M^K$  and we will assume this condition.

**Definition 2.3.3.** For a given code, let  $l_i$  be the  $i^{th}$  digit in the code (from the right), for  $i = 1, 2, \dots, K$ , i.e., the given code is  $l_K \dots l_2 l_1$  in base  $M$ .

**Remark.** The interpretation of the value of  $l_i$  is that in round  $i$ , after dealing separated piles, we will put any  $l_i$  piles above the chosen pile from the spectator. Note that the number of dealing rounds,  $K$ , is the number of digits of the code. Decoding a code, we start from the most right digit and move forward to the left and end at the most left digit.

**Example 2.3.4.** Performing with 64 cards, we deal cards into  $M = 4$  piles in each round. Thus, the parameter  $N = 16$ . Suppose that the chosen position  $x = 20$ . At first, we expand  $x - 1 = 20 - 1 = 19$  in base 4 which is  $103_4$ . Thus, the parameter  $K$  is 3. In the first round, we put the chosen pile to the bottom ( $l_1 = 3$ ), i.e., there are 3 piles over the chosen pile. In the second round, the chosen pile goes to the top ( $l_2 = 0$ ), and in the final round, the chosen pile goes to the second order out of four ( $l_3 = 1$ ). Following all of these steps, we can finally move the chosen card to position  $x = 20$ .

Obviously, in each round of performing, we separate the deck into minor piles with the same number of cards. The number of performing cards is  $MN$ , whose form is more general than  $M^K$ . In this project, we focus on searching for some solutions of the Gardner's problem with  $MN$  cards that can be dealt into  $M$  piles, with  $N$  cards each and  $K$  rounds of restacking those  $M$  piles.

**Definition 2.3.5.** Let  $a_0$  be the starting position of the chosen card in the deck.

Notice that all possible values of  $a_0$  are  $1, 2, \dots, MN$ .

**Definition 2.3.6.** For a Gardner's problem with parameters  $M, N, K$ , we call a given code for performing a "usable code", if there exists an  $x \in \{1, 2, \dots, MN\}$  such that for every  $a_0 \in \{1, 2, \dots, MN\}$  when we perform the Gardner's trick according to that given code with the starting position  $a_0$ , the chosen card finally appears at the position  $x$ .

**Definition 2.3.7.** For a Gardner's problem with parameters  $M, N, K$ , if for every desired position  $x \in \{1, 2, \dots, MN\}$  we can find a usable code to perform the Gardner's trick that finally put any chosen card from the spectator to the position  $x$ , then we say that this Gardner's problem is "solvable". Otherwise, we say that the Gardner's problem is "unsolvable".

**Example 2.3.8.** Performing with a deck of 12 cards, we can perform the Gardner's trick in several ways. For example, we can consider the Gardner's problem with parameters  $M = 2, N = 6, K = 4$ ;  $M = 3, N = 4, K = 3$ ; or  $M = 4, N = 3, K = 2$ .

1. Problem 1:  $M = 2, N = 6, K = 4$

Table 2.4 shows all possible  $x$  that can be done within  $K = 4$  rounds with some corresponding codes. Table 2.5 shows performing with code  $0000_2$ ,  $0001_2$ , and  $0010_2$ . In Table 2.5, the first column represents the initial position of the chosen card; the second, third, fourth and fifth columns represent the position of the tracking card after restacking the deck in round 1, 2, 3 and 4, respectively.



**Table 2.4:** Code for  $M = 2, N = 6, K = 4$ 

Position ( $x$ )	1	3	4	6	7	9	10	12
Code	0000 <sub>2</sub>	0011 <sub>2</sub>	0100 <sub>2</sub>	0111 <sub>2</sub>	1000 <sub>2</sub>	1011 <sub>2</sub>	1100 <sub>2</sub>	1111 <sub>2</sub>

**Table 2.5:** All initial positions are performed with codes 0000<sub>2</sub>, 0001<sub>2</sub>, and 0010<sub>2</sub>, respectively from left to right, for the Gardner's problem with parameters  $M = 2, N = 6, K = 4$ 

1	1	1	1	1	1	7	4	2	1	1	1	7	4	2
2	1	1	1	1	2	7	4	2	1	2	1	7	4	2
3	2	1	1	1	3	8	5	3	2	3	2	7	4	2
4	2	1	1	1	4	8	5	3	2	4	2	7	4	2
5	3	2	1	1	5	9	6	3	2	5	3	8	4	2
6	3	2	1	1	6	9	6	3	2	6	3	8	4	2
7	4	2	1	1	7	10	7	4	2	7	4	8	4	2
8	4	2	1	1	8	10	7	4	2	8	4	8	4	2
9	5	3	2	1	9	11	8	4	2	9	5	9	5	3
10	5	3	2	1	10	11	8	4	2	10	5	9	5	3
11	6	3	2	1	11	12	9	5	3	11	6	9	5	3
12	6	3	2	1	12	12	9	5	3	12	6	9	5	3
	0	0	0	0	1	0	0	0	0	1	0	0		

This problem seems to be solvable because they have 16 different ways of performing and we only need just 12 positions for  $x$ . Surprisingly, it is unsolvable. Some desired positions cannot be done within  $K = 4$  rounds. For instance, we cannot perform the trick for  $x = 2$ . Thus, this Gardner's problem is unsolvable. This is because some codes are not usable. For example, performing with code 0001<sub>2</sub> or 0010<sub>2</sub> cannot rearrange all initial positions to the desired position  $x = 2$ . Hence, codes 0001<sub>2</sub> and 0010<sub>2</sub> are not usable codes.

## 2. Problem 2: $M = 3, N = 4, K = 3$

Table 2.6 shows all possible  $x$ , that can be done within  $K = 3$  rounds.

**Table 2.6:** Code for  $M = 3, N = 4, K = 3$ 

Position	1	2	3	4	5	6	7	8	9	10	11	12
Code	000	010	012	021	100	110	112	121	200	210	212	221
	001			022	101			122	201			222

Every desired position  $x$  has a code to perform. Moreover, some  $x$  can be performed in several ways. For example,  $x = 1$  can be performed with code  $000_3$  or  $001_3$ . Thus, this Gardner's problem is solvable.

3. Problem 3:  $M = 4, N = 3, K = 2$

Table 2.7 shows all possible  $x$  that can be done within  $K = 2$  rounds.

**Table 2.7:** Code for  $M = 4, N = 3, K = 2$

Position	1	3	4	6	7	9	10	12
Code	00	03	10	13	20	23	30	33

At the position  $x = 2, 5, 8, 11$  cannot be done within  $K = 2$  rounds. Thus, this Gardner's problem is unsolvable.

## 2.4 Wolfram Language

The Wolfram Language [7] is a general multi-paradigm computational language developed by Wolfram Research and is the programming language of the mathematical symbolic computation program Mathematica and the Wolfram Programming Cloud. It emphasizes symbolic computation, functional programming, and rule-based programming and can employ arbitrary structures and data. Wolfram Mathematica (usually termed Mathematica) is a modern technical computing system spanning most areas of technical computing, including neural networks, machine learning, image processing, geometry, data science, visualizations, and others. The system is used in many technical, scientific, engineering, mathematical, and computing fields. It was conceived by Stephen Wolfram and is developed by Wolfram Research of Champaign, Illinois. The Wolfram Language is the programming language used in Mathematica.

## Chapter III

### GARDNER'S TRICK PROGRAM

Let's dissect roughly about how to perform Gardner's trick in one game with parameters  $M, N, K$ . We have  $M$  choices to put the chosen pile in each round, and we have  $K$  round to perform the Gardner's trick; thus, the most possible ways to perform are  $M^K$ . For each number in  $\{1, 2, \dots, M^K\}$  representing each possible way, we can find a corresponding code to perform the trick by subtracting it by 1 and converting the result from base 10 to base  $M$  to get a code  $l_K l_{K-1} \dots l_2 l_1$ . The code will show us how to rearrange the order of the tracking pile in each round. If we know all usable codes, we can simulate performing in every position. One code of performing should move the tracking card to only one desired position  $x$  no matter what the starting position  $a_o$  of the tracking card is.

From the beginning, the tracking card can be shuffled to any position of the whole deck. There are tremendous cases to generate. However, using computer programming can do hundreds of cases in a second. The overview of our algorithm is as follows.

1. Input the value of  $M, N, K$  and check for validity of those parameters.
2. Find all  $M^K$  possible codes.
3. Run a loop over all possible codes to find all usable codes
4. If every  $x$  can be matched by a usable code, the problem is solvable. If not, it is unsolvable.

**Remark.** The number of possible beginning positions is  $MN$ .

The number of possible performing codes is  $M^K$ .

### 3.1 Conditions and Main Result for Our Program

#### 3.1.1 Conditions on parameters $M, N, K$

Parameters  $M, N, K$  must satisfy some conditions in order to get a possible solvable problem.

1.  $M, N, K \in \mathbb{N}, M \geq 2$

Recall that  $M$  is the number of piles,  $N$  is the number of cards in each pile, and  $K$  is the number of performing rounds. Obviously,  $M, N, K$  have to be positive integers. Also, the number of piles must be greater than 1 to create a magic.

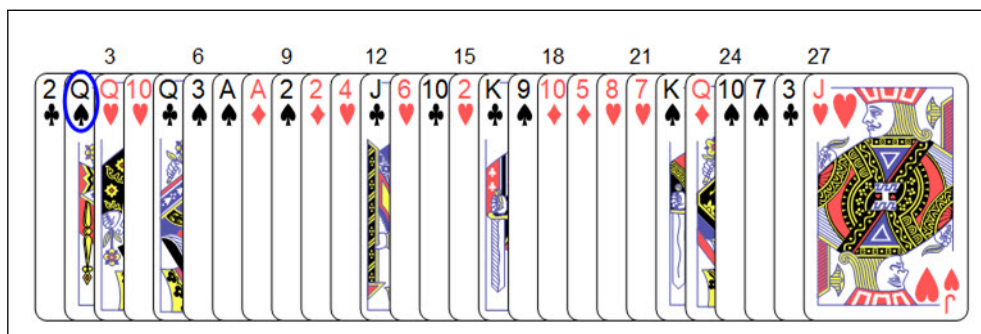
2.  $MN \leq M^K$  or equivalently  $N \leq M^{K-1}$

Observe that the number of possible desired positions is  $MN$  and the number of possible performing codes is  $M^K$ . In an ideal situation when all possible codes to perform the trick are usable. Then, we have a map from each possible code to an  $x \in \{1, 2, \dots, MN\}$ . Suppose  $MN > M^K$ . Then, there will be an  $x$  which does not have a usable code to perform. Thus, the problem with this set of parameters  $M, N, K$  is unsolvable. Hence,  $M^K$  must be greater than or equal to  $MN$ .

#### 3.1.2 New position after performing the trick one round

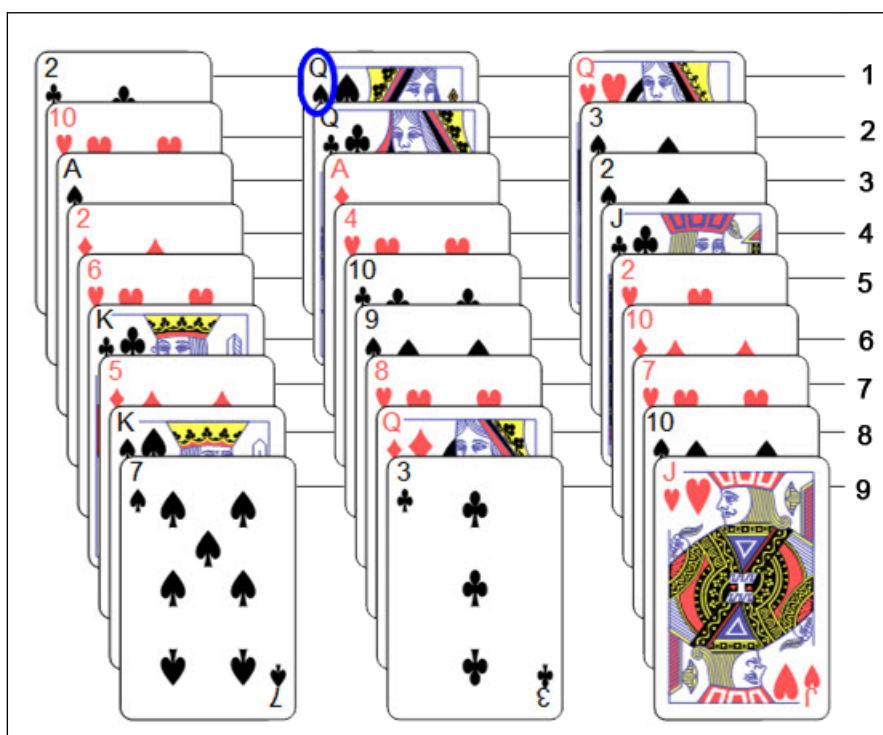
We will create an equation describing all possible positions of the tracking card after one round of performing. Let us recall a procedure to perform the trick in each round. Firstly, we separate  $MN$  cards into  $M$  piles. Hence, the tracking card starting at position  $a$ , in that round, will be at the position  $\left\lceil \frac{a}{M} \right\rceil$  of a minor pile after dealing the cards. Next, we rearrange the order of the tracking pile according to a digit  $l$  in a code. Then,  $l$  is the number of piles above the tracking pile. Since each minor pile has  $N$  cards, we can conclude that the number of card above the tracking pile is  $lN$ .

**Example 3.1.1.** Suppose that we perform one round with 27 cards using parameters  $M = 3$  and  $N = 9$ .



**Figure 3.1:** An example of performing: before dealing cards [8]

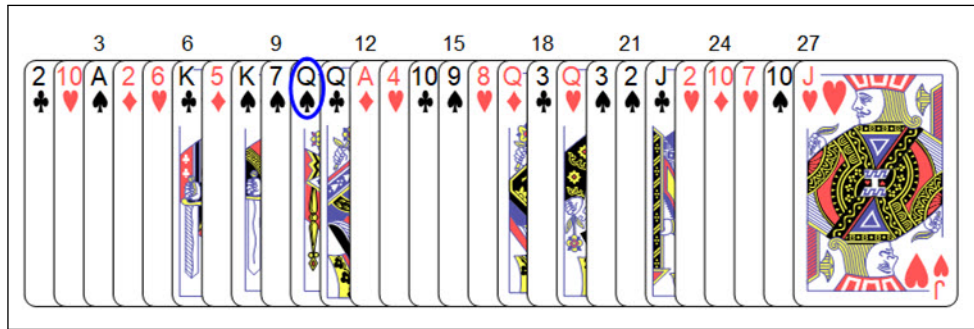
Assume that the chosen card is  $Q♠$ . As you can see from Figure 3.1, the position at the beginning is the  $2^{nd}$  from the top of the deck when the deck faces down. Then, we deal this deck into 3 piles.



**Figure 3.2:** An example of performing: after dealing cards [8]

From Figure 3.2, the position of the tracking card in the separated pile is now the  $1^{st}$ . In other cases,  $2♣$  and  $Q♥$  also go to the  $1^{st}$ ;  $10♥$ ,  $Q♣$  and  $3♠$  go to the  $2^{nd}$ , and

so on. On the other hand, we have a map  $f$  from a starting position to a position of the tracking card on the minor pile, say,  $\{1, 2, 3\} \rightarrow \{1\}$ ,  $\{4, 5, 6\} \rightarrow \{2\}$ , ... ,  $\{25, 26, 27\} \rightarrow \{9\}$ . Thus, the function  $f$  is given by  $f(a) = \left\lceil \frac{a}{M} \right\rceil$ , where  $a$  is the position from the beginning of that round and  $M$  is the number of separated piles, which is 3 in this example.



**Figure 3.3:** A example of performing: after performing the trick one round [8]

Afterwards, suppose that the digit in a code of that round is  $l = 1$ ; thus, we rearrange the deck by putting the  $Q♠$ 's pile to the second place. This means that there is 1 pile above the  $Q♠$ 's pile. Now, the new position of  $Q♠$  in the deck is the position in a separated pile which is  $\left\lceil \frac{2}{3} \right\rceil$  adding up with the number of cards above it which is the number of pile(s) above it times the number of cards in each pile. Thus, the new position is  $\left\lceil \frac{2}{3} \right\rceil + 1(9) = 10^{th}$  as we can see from Figure 3.3.

In general, the new position of the chosen card is given by the following formula.

**Lemma 3.1.2.** For each round  $i \in \{1, 2, \dots, K\}$ ,

let  $P_s$  be the starting position of the tracking card,

$P_e$  be the position after performing the trick one round, and

$l_i$  be the value of the  $i^{th}$  digit in the code.

Then,

$$P_e = \left\lceil \frac{P_s}{M} \right\rceil + l_i N.$$

**Remark.** We assure that this formula for  $P_e$  gives a valid value for the position of a card in the deck, i.e.  $1 \leq P_e \leq MN$ . To see this, assume that  $1 \leq P_s \leq MN$ . Then,

$$\begin{aligned} \frac{1}{M} &\leq \frac{P_s}{M} \leq N \\ 1 = \left\lceil \frac{1}{M} \right\rceil &\leq \left\lceil \frac{P_s}{M} \right\rceil \leq \lceil N \rceil = N \\ 1 + l_i N &\leq \left\lceil \frac{P_s}{M} \right\rceil + l_i N \leq N + l_i N \\ 1 &\leq P_e \leq MN \end{aligned}$$

because  $0 \leq l_i \leq M - 1$  for all  $i = 1, 2, \dots, K$ .

### 3.1.3 Main Theorem

The possibility of the tracking card position after shuffling the deck can be at the 1<sup>st</sup>, 2<sup>nd</sup>, ..., or  $MN^{\text{th}}$  position. Recall that  $a_0$  is the starting position of the chosen card in the deck. Notice that all possible values of  $a_0$  are  $1, 2, \dots, MN$ .

**Definition 3.1.3.** With the starting position  $a_0$  and a given code of performing, we define  $a_i$  to be the position of the tracking card after performing in round  $i$  for all  $i = 1, 2, \dots, K$ .

**Definition 3.1.4.** With a given code of performing, define  $I_0 = 1$  and  $E_0 = MN$ , and for all  $i = 0, 1, \dots, K - 1$ , we define recursively

$$I_{i+1} = \left\lceil \frac{I_i}{M} \right\rceil + l_i N \quad \text{and} \quad E_{i+1} = \left\lceil \frac{E_i}{M} \right\rceil + l_i N.$$

**Remark.**  $I_i$  and  $E_i$  represent the position of the tracking card after performing in round  $i$  with  $a_0 = 1$  and  $a_0 = MN$ , respectively. For a usable code, we must have that  $I_K = E_K$ . For the case that we considered to be a usable code, performing from  $I_0$  or  $E_0$  should give the same position of  $x$ , i.e., if we start from  $I_0$  and  $E_0$ , after  $K$  rounds of performing,  $I_K$  and  $E_K$  must be equal. We will show that for a usable code starting from any position  $a_0$  will eventually end up at  $I_K = E_K$  as well.

**Theorem 3.1.5.**  $I_i \leq a_i \leq E_i$  for all  $i = 0, 1, 2, \dots, K$ .

*Proof.* We will prove by induction. Clearly,  $I_0 = 1 \leq a_0 \leq MN = E_0$ . For each  $i = 0, 1, \dots, K - 1$ , assume that  $I_i \leq a_i \leq E_i$ . Then, by Lemma 3.1.2 we have that

$$\begin{aligned} \frac{I_i}{M} &\leq \frac{a_i}{M} \leq \frac{E_i}{M} \\ \left\lceil \frac{I_i}{M} \right\rceil &\leq \left\lceil \frac{a_i}{M} \right\rceil \leq \left\lceil \frac{E_i}{M} \right\rceil \\ \left\lceil \frac{I_i}{M} \right\rceil + l_i N &\leq \left\lceil \frac{a_i}{M} \right\rceil + l_i N \leq \left\lceil \frac{E_i}{M} \right\rceil + l_i N \\ I_{i+1} &\leq a_{i+1} \leq E_{i+1} \quad \square \end{aligned}$$

Note that for a usable code, we must have that  $a_K = x$  (for some  $x$ ) for any initial position  $a_0 \in \{1, 2, \dots, MN\}$ . From Theorem 3.1.5, we have that for a usable code  $I_K = a_K = E_K$ . Thus, we get the main result for our program as the following corollary.

**Corollary 3.1.6.** *The given code is usable if and only if  $I_K = E_K$ . In such case, the corresponding desired position for the usable code is  $x = I_K$  (or equivalently  $E_K$ ).*

To check if a problem is solvable, we can create a computer program to run a loop to check for every starting position whether we can find a usable code to perform or not; and then check if every desired position can be performed by a usable code. From Corollary 3.1.6, we can construct a program that can run significantly faster than the obvious choice of program mentioned above by keeping track of only two sequences of positions,  $(I_i)_{i=0}^K$  and  $(E_i)_{i=0}^K$ , since earlier we had to clarify every  $a_0 \in \{1, 2, \dots, MN\}$ .

### 3.2 Our Proposed Program

We make an instant program using Mathematica which can tell whether a Gardner's problem with given parameters is solvable or unsolvable. Moreover, for a solvable problem, the program can show all possible ways to perform the Gardner's trick. Figure 3.4 is the flow chart of our program which is described as follows.



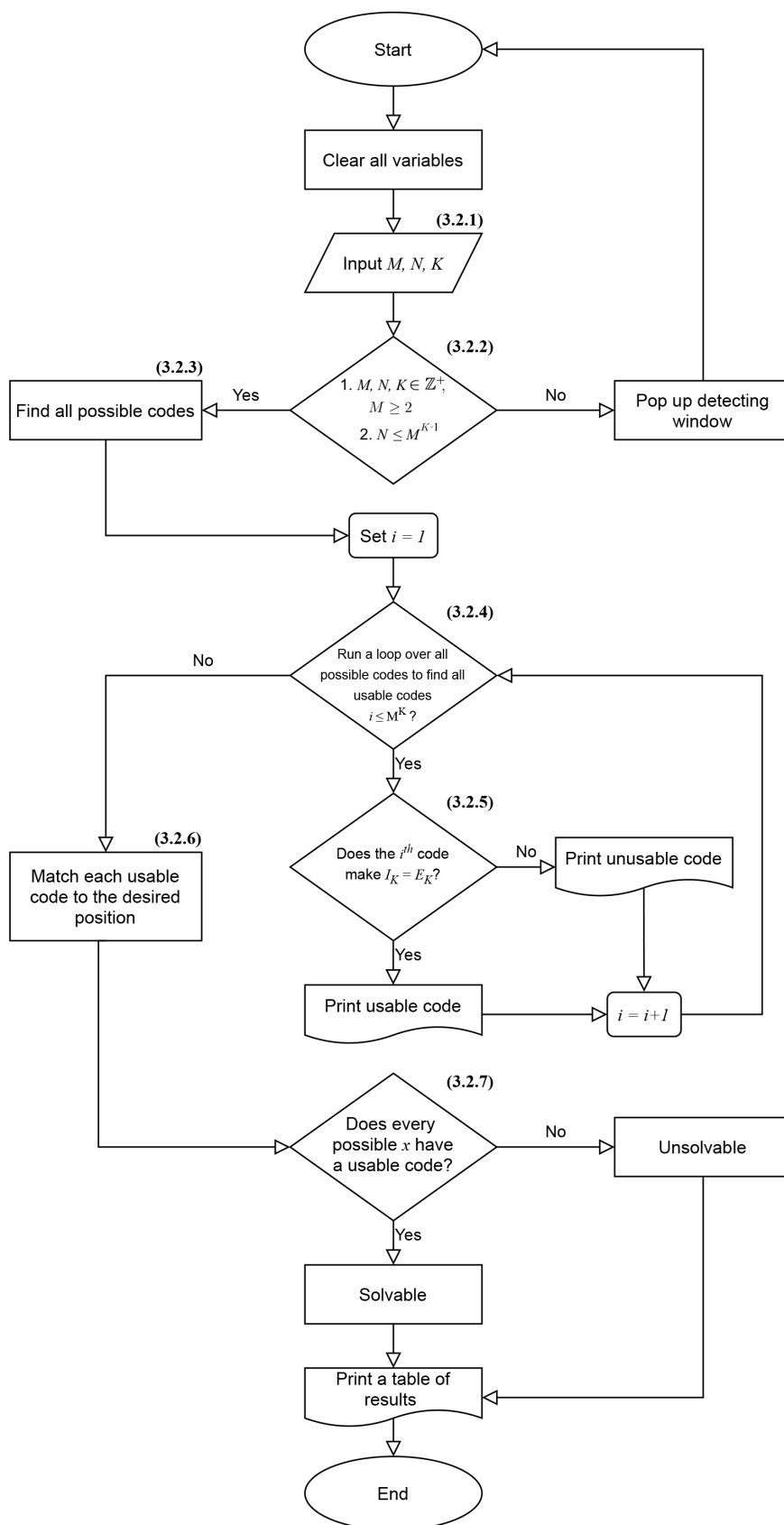


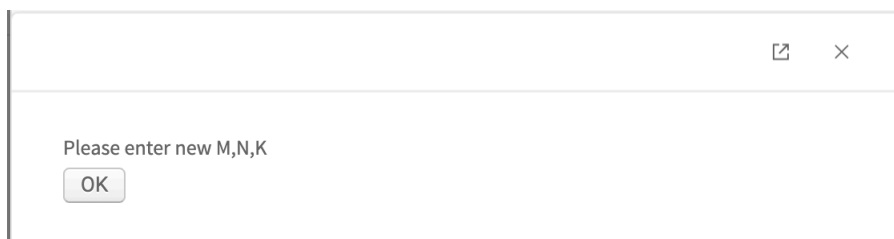
Figure 3.4: The flow chart for our program

### 3.2.1 Input parameters $M$ , $N$ , $K$

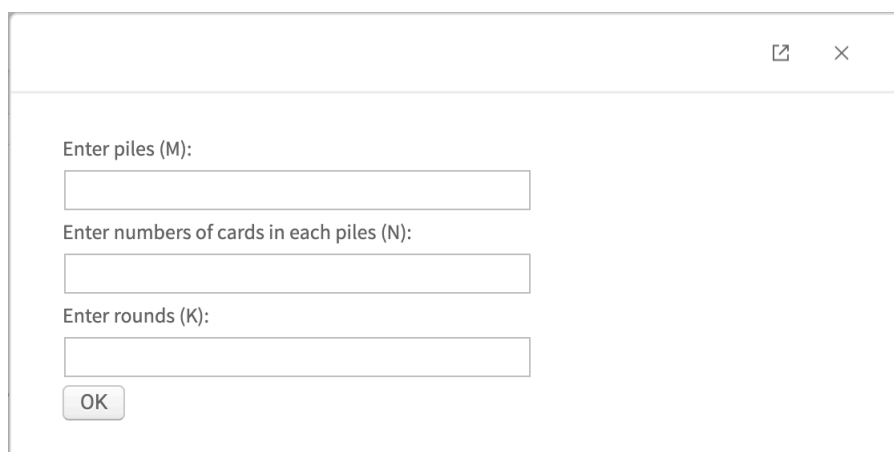
```
ClearAll["Global`*"]
CreateDialog[DialogNotebook[{TextCell["Enter piles (M): "], InputField[Dynamic[m], String],
  TextCell["Enter numbers of cards in each piles (N): "], InputField[Dynamic[n], String],
  TextCell["Enter rounds (K): "], InputField[Dynamic[k], String],
  DefaultButton[DialogReturn[]]}
]];
CreateWindow[DialogNotebook[{TextCell["Please enter M, N, K"], DefaultButton[]}],
  WindowFloating -> True];
```

**Figure 3.5:** Input parameters  $M$ ,  $N$ ,  $K$

First of all, we start with clearing all variable. Then, the program will show two pop-up windows. The first window reminds us to input the parameters  $M$ ,  $N$ ,  $K$ ; we press the OK button and then the second pop-up window will allow us to insert the values of  $M$ ,  $N$ ,  $K$ . Now, the program keeps the value of  $M$ ,  $N$ ,  $K$  as a string.



**Figure 3.6:** Pop-up window(1)



**Figure 3.7:** Pop-up window(2): input window

### 3.2.2 Check the conditions on parameters $M, N, K$

```

m = ToExpression[m];
n = ToExpression[n];
k = ToExpression[k];
mn = m * n;

If[IntegerQ[m] && IntegerQ[n] && IntegerQ[k],
  If[m > 1 && n > 0 && k > 0,
    If[n - mk-1 ≤ 0,
      Return[],

      Print["\" error: N > \\(\\*SuperscriptBox[\\(M\\), \\((-1) + K\\)]\\) \"];
      CreateWindow[DialogNotebook[{TextCell[Row[{"M = ", ToString[m], " N = ", ToString[n], " K =
        ", ToString[k], " is unsolvable, Please enter new M, N, K"}]], DefaultButton[]],
        WindowFloating → True]]];
      Exit[],

    Print[" error: M ≤ 1 or N and K ≤ 0"];
    CreateWindow[DialogNotebook[{TextCell["M or N or K ≤ 0, Please enter new M, N, K"],
      DefaultButton[]], WindowFloating → True]]];
    Exit[],

    Print["error: All inputs are not an interger"];
    CreateWindow[DialogNotebook[{TextCell["All inputs are not an interger, Please enter new M,
      N, K"], DefaultButton[]], WindowFloating → True]]];
    Exit[]]

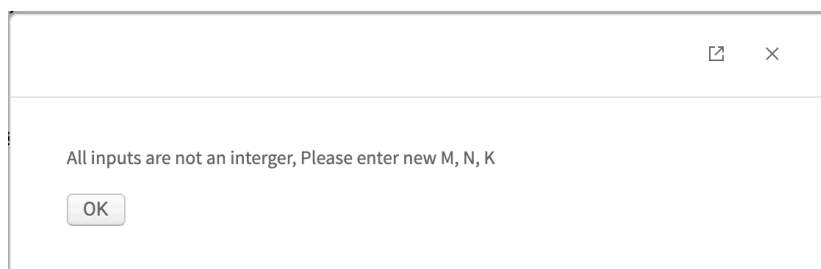
```

**Figure 3.8:** Check the conditions on parameters

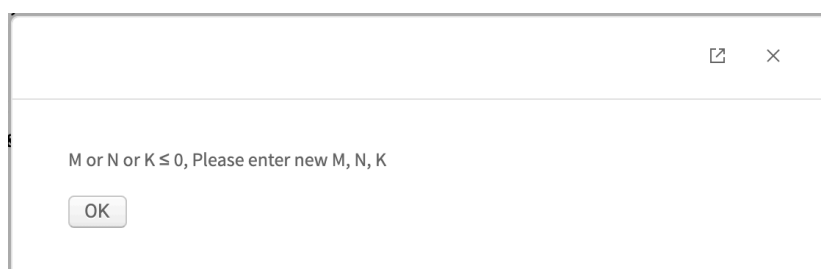
After we press OK, the program collects the values of  $M, N, K$  in the variables  $m, n, k$ , respectively. The program will convert the string to an expression of the number, and now we can  $m, n, k$  to calculate and proceed our algorithm. This blog of coding check the basic conditions of  $M, N, K$  from 3.1.1 which are as follows.

1. All inputs must be integers.
2.  $M \geq 2, N > 0$  and  $K > 0$ .
3.  $MN \leq M^K$ .

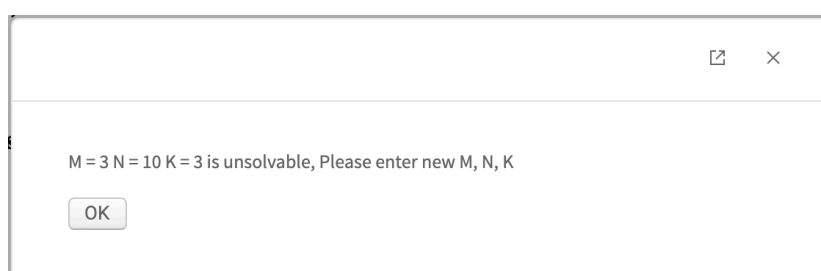
If one value of the parameters  $M, N, K$  is not conformed to condition 1, 2 or 3, then the program will pop up a detecting window and inform us to input new  $M, N, K$ .



**Figure 3.9:** Detecting all inputs are not an integer



**Figure 3.10:** Detecting  $M$  or  $N$  or  $K \leq 0$



**Figure 3.11:** Detecting  $MN \leq M^K$

### 3.2.3 Create all possible codes to perform

```
case = Tuples[Range[0, m - 1], k];
```

**Figure 3.12:** Finding all codes

*case* is a set of tuples that collects all possible performing codes. The sample space for rearranging the chosen pile is  $0, 1, \dots, M - 1$ . Coming to the event of finding all possible codes, each digit of a code would yield  $0, 1, \dots, M - 1$ . Performing  $K$  rounds needs  $K$  digits of a code; thus, *case* has  $M^K$  members.

**Example 3.2.1.** For  $M = 3, K = 2$ , we have

$case = \{\{0, 0\}, \{0, 1\}, \{0, 2\}, \{1, 0\}, \{1, 1\}, \{1, 2\}, \{2, 0\}, \{2, 1\}, \{2, 2\}\}.$

### 3.2.4 Run a loop over all possible codes to find all usable codes

```

rs = List[];
rf = List[];
For[i = 1, i ≤ mk, i++,
  f = case[[i]];
  pstart = 1;
  pend = mn;
  For[j = 1, j ≤ k, j++,
    l = f[[j]];

    pstart = Ceiling[ $\frac{pstart}{m}$ ] + (l * n);

    pend = Ceiling[ $\frac{pend}{m}$ ] + (l * n);

    Print[f, ",", j, ",", l, ",", pstart, ",", pend];
  ];

```

**Figure 3.13:** Finding all usable codes

- $rs = List[]$ , prepare empty list for usable codes.
- $rf = List[]$ , prepare empty list for unusable codes.
- $pstart = 1 = I_0$
- $pend = mn = E_0$
- $(new)pstart = \lceil \frac{pstart}{m} \rceil + ln$
- $(new)pend = \lceil \frac{pend}{m} \rceil + ln$

Coding in this section is a simulation of performing all possible codes for every  $a_0 \in \{1, 2, \dots, MN\}$ . However, from Theorem 3.1.5, it suffices to perform only 2 positions,  $I_0$  and  $E_0$ ; in this context, they are  $pstart$  and  $pend$ , respectively. The loop is run through a variable  $j = 1, 2, \dots, K$  which represents the performing round and a variable  $i = 1, 2, \dots, M^K$  which represents each possible codes ordered in a monotone

increasing manner, i.e.,  $i = 1$  represents the code  $000\dots 0_M$ ,  $i = 2$  represents the code  $000\dots 1_M$ , and so on. In loop  $j$ ,  $pstart$  and  $pend$  will be recomputed to keep the values for  $I_j$  and  $E_j$ . After the final round of looping,  $pstart$  and  $pend$  will keep the values for  $I_K$  and  $E_K$ , respectively.

### 3.2.5 Classify usable or unusable codes

```

If[pstart == pend
  , AppendTo[rs, {pstart, pend, f}];
  , AppendTo[rf, {pstart, pend, f}];
];
];

Print["====Success Results===="];
Print[rs];
Print["====Fail Results===="];
Print[rf];
Print["====="];

```

**Figure 3.14:** Usable code and unusable code checking

From Corollary 3.1.6,  $I_K$  of the usable code must equal  $E_K$ . In this context, if  $pstart = pend$ , it will be considered a usable code and it will be appended to  $rs$ , otherwise it is an unusable code and it will be appended to  $rf$ . The lists for both kinds of codes will be displayed, sorted by the values of  $I_K$  and distributed into success results and fail results, e.g.,  $\{10, 10, \{0, 3\}\} \in rs$ , which means  $I_K = 10 = E_K$  and a code to perform the trick is  $30_M$ ; and  $\{7, 8, \{1, 2\}\} \in rf$ , which means  $I_K = 7 \neq 8 = E_K$ , and the code  $21_M$  is not a usable code.

### 3.2.6 Match each usable code to the corresponding desired position

```
rs = Sort[rs];
tposition = . ;
tcode = . ;
tposition = List[Position];
tcode = List[Code];
j = 1;

isfindall = "Y";
For[i = 1, i ≤ mn, i++,
  AppendTo[tposition, i];

  listcode = List[];
  isfind = "N";
```

Figure 3.15: Match the code to the desired position(1)

- $tposition = List[Position]$  prepares a list of desired positions for the resulting table.
- $tcode = List[Code]$  prepares a list of codes for the resulting table.
- Loop  $i$  will create the  $i^{th}$  desired position and append it to  $tposition$ .
- $listcode = List[]$  prepares a list collecting all codes.

```
For[j = j, j ≤ Length[rs], j++,
  rsc = rs[[j]];
  rsccl = rsc[[1]];
  If[i ≠ rsccl, Break[]];

  isfind = "Y";
  rsccl3 = rsc[[3]];
  rsccl3 = Reverse[rsccl3];
  rsccl3 = StringDelete[StringDelete[ToString[rsccl3], PunctuationCharacter],
    WhitespaceCharacter];
  AppendTo[listcode, rsccl3];
  ];

  If[isfind == "N", isfindall = "N"];

  AppendTo[tcode, listcode];
  ];
```

Figure 3.16: Match the code to the desired position(2)

Now, for a usable code,  $I_K = E_K$ . The variable  $i$  representing the desired position  $x$  will run through  $1, 2, \dots, MN$ . For each  $i$ , we will check whether the variable  $pstart$  is equal to  $i$ . On the other hand, we check if  $I_K = x$ . If they are matched, the program will reverse the code; for example, from 1202 to 2021, because performing from the code in base  $M$ , we will begin interpreting the code from the most right digit. The matched pair will be appended to  $listcode$  from the  $1^{st}$  position to the  $MN^{th}$  position. The program will run through every case and pair one by one until there is no member left in  $rs$ . Then, the list collecting all codes is ready to appear in the result table. Now, we get

- $tposition = \{position, 1, 2, \dots, mn\}$
- $tcode = \{\{“code”\}, \{1^{st}code\}, \dots, \{MN^{th}code\}\}$

At first, we set  $isfindall = “Y”$ . If all usable codes can be paired with all  $x$ , we set another value  $isfind = “Y”$ . If not, we set  $isfind = “N”$ . After running through all cases, if  $isfind = “N”$ , we set  $isfindall = “N”$ , which means that the program cannot find a performing code for some  $x$ .

### 3.2.7 Check whether the Gardner’s problem is solvable or not and show the result

```

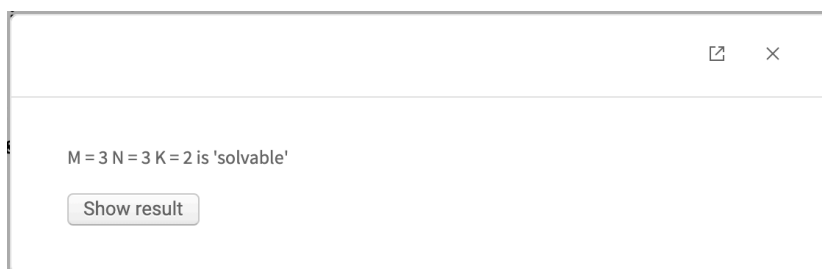
If[isfindall == "N"
, CreateWindow[DialogNotebook[{TextCell[Row[{"M = ", ToString[m], " N = ", ToString[n],
" K = ", ToString[k], " is 'unsolvable'"}]], Button["Show result",
DialogReturn[CreateWindow[DocumentNotebook[TableForm[{tposition,
tcode}]]]]]]];
, CreateWindow[DialogNotebook[{TextCell[Row[{"M = ", ToString[m], " N = ", ToString[n],
" K = ", ToString[k], " is 'solvable'"}]], Button["Show result",
DialogReturn[CreateWindow[DocumentNotebook[TableForm[{tposition,
tcode}]]]]]]];
];

```

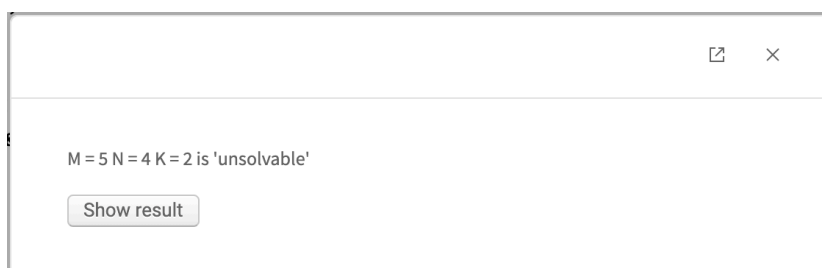
**Figure 3.17:** Show a resulting table

If  $isfindall = “N”$ , we create the window saying that  $M, N, K$  is “unsolvable”. If not, we create the window saying that  $M, N, K$  is “solvable”. Both of these windows have a button which allows clicking to show results as a table created from  $tposition$  and  $tcode$ .





**Figure 3.18:** Solvable pop-up window



**Figure 3.19:** Unsolvable pop-up window

### 3.3 Some Experiments

The base program can work for any set of parameters  $M, N, K$  perfectly. However, it is good and fast only if it cannot be given many sets of parameters at once. After I try some of parameters  $M, N, K$ , I wish I have an overview program to see the relationship of  $M, N, K$ , where I might find some conditions on the parameters that make the Gardner's problem solvable or unsolvable. Then, an extra program appears. Figure 3.20 shows command function name as "Solver".

```

Solver[a_, b_, c_] := Run[

ab = a * b;
case = Tuples[Range[0, a - 1], c];
rs = List[];
rf = List[];
For[i = 1, i ≤ a^c, i++,
  f = case[[i]];
  pstart = 1;
  pend = ab;
  For[j = 1, j ≤ c, j++,
    l = f[[j]];
    pstart = Ceiling[ $\frac{pstart}{a}$ ] + (l * b);

    pend = Ceiling[ $\frac{pend}{a}$ ] + (l * b);

  ];
  If[pstart == pend
    , AppendTo[rs, {pstart, pend, f}];
    , AppendTo[rf, {pstart, pend, f}];
  ];
];

rs = Sort[rs];
tposition = List[Position];
tcode = List[Code];
j = 1;

isfindall = "Y";
For[i = 1, i ≤ ab, i++,
  AppendTo[tposition, i];

  listcode = List[];
  isfind = "N";

  For[j = j, j ≤ Length[rs], j++,
    rsc = rs[[j]];
    rsccl = rsc[[1]];
    If[i ≠ rsccl, Break[]];

    isfind = "Y";
    rsccl3 = rsc[[3]];
    rsccl3 = Reverse[rsccl3];
    rsccl3 = StringDelete[StringDelete[ToString[rsccl3],
      PunctuationCharacter], WhitespaceCharacter];
    AppendTo[listcode, rsccl3];
  ];

  If[isfind == "N", isfindall = "N"]; ];

AppendTo[tcode, listcode];
];

If[isfindall == "N"
  , result = ab - Length[rs]
  , result = Y
]
]

```

Figure 3.20: Extra program

“Solver” command is quite similar to coding in 3.2, except that checking  $M, N, K$  conditions from 3.2.1 and 3.2.2 is gone and the last block from 3.2.7 is changed to the red block. Instead of popping up the window, if the program cannot find the code to every  $x$ , we have that *result* is equal to  $MN$  minus the number of desired positions,  $x$ , that have a usable code. Now, *result* will keep the number of  $x$  that cannot be performed. If the Gardner’s problem with a set of parameters  $M, N, K$  is solvable, *result* will keep the value  $Y$ . On the other hands, the resulting table will display a number, if it is an unsolvable case; and it will display  $Y$ , if it is a solvable case. I decide to simulate this extra program for some experiments as follows.

### 3.3.1 Fix $M$ and $K$ , run $N$

For any given values of  $M$  piles and  $K$  rounds, it is interesting to search for values of  $N$  cards that can make the Gardner's problem solvable. For any given fixed values of  $M$  and  $K$ , this program will solve for  $N \in \{1, 2, \dots, N_0\}$  when we input the threshold  $N_0$ .

```

m = ;
n = ;
k = ;
tn = List["n"];
tr = List[m];
For[d = 1, d ≤ n, d++,
  Print["m = ", m, ", n = ", d];
  AppendTo[tn, d];
  Solver[m, d, k];
  AppendTo[tr, result];
]
TableForm[{tn, tr}]

```

**Figure 3.21:** Fix  $M$  and  $K$ , run  $N$  program

- You input  $m$  and  $k$  as fixed variables, and  $n$  as a running variable starting from 1 to  $N_0$  by yourself behind the equal sign. Note that this program will not detect whether your inputs are inappropriate variables.
- $tn$  is a list for showing all values of  $N$ .
- $tr$  is a list prepared for all results.
- Loop  $d$ : after solved by Solver command, the result will be appended to  $tr$ .

### 3.3.2 Fix $M$ and $N$ , run $K$

For a given number of deck that will dealt into  $M$  piles and each pile have  $N$  cards, it is interesting to search for the number of rounds,  $K$ , that can make the Gardner's problem solvable. For any given fixed values of  $M$  and  $N$ , this program will solve for  $K = \{1, 2, \dots, K_0\}$  when we input the threshold when we input the threshold  $K_0$ .

```

m = ;
n = ;
k = ;
tn = List["k"];
tr = List["Result"];
For[d = 1, d ≤ k, d++,
  AppendTo[tn, d];
  Solver[m, n, d];
  AppendTo[tr, result];
]
TableForm[{tn, tr}]

```

**Figure 3.22:** Fix  $M$  and  $N$ , run  $K$  program

This program is the same as 3.3.1, only change running variable from  $n$  to  $k$ .

### 3.3.3 Fix $K$ , run $M$ and $N$

For any given number of round,  $K$ , it is interesting to search for  $M$  and  $N$  that can perform the Gardner's problem. For any given fixed values of  $K$ , this program will solve for  $M = \{1, 2, \dots, M_0\}$  and  $N = \{1, 2, \dots, N_0\}$  when we input the threshold  $M_0$  and  $N_0$ .

```

m = ;
n = ;
k = ;
tn = List["n"];
tr = List[];
For[i = 1, i ≤ m, i++,
  AppendTo[tr, {i}];
];
For[d = 1, d ≤ n, d++,
  AppendTo[tn, d];
  For[e = 1, e ≤ m, e++,
    Solver[e, d, k];
    AppendTo[tr[[e]], result];
  ]
]
PrependTo[tr, tn];
CreateWindow[DocumentNotebook[TableForm[tr]]]

```

**Figure 3.23:** Fix  $K$ , run  $M$  and  $N$  program

This program is extended from 3.3.1 by adding another loop for running  $M$ .

The extra program is good for an overview. However, for the parameters  $M, N, K$ , solving for a code to perform the Gardner's trick still needs to use the main program.

# Chapter IV

## RESULTS

### 4.1 Fix $M$ and $K$ , run $N$

The following results are from 3.3.1.

(i)  $M = 2$

a)  $K = 2$

"n"	1	2	3	4	5	6	7	8	9	10
2	Y	Y	6	8	10	12	14	16	18	20

b)  $K = 3$

"n"	1	2	3	4	5	6	7	8	9	10
2	Y	Y	2	Y	10	12	14	16	18	20

c)  $K = 4$

"n"	1	2	3	4	5	6	7	8	9	10
2	Y	Y	Y	Y	2	4	10	Y	18	20

d)  $K = 5$

"n"	1	2	3	4	5	6	7	8	9	10	11
2	Y	Y	Y	Y	Y	Y	Y	Y	2	4	10
	12	13	14	15	16	17	18	19	20		
	8	18	20	26	Y	34	36	38	40		

(ii)  $M = 3$ a)  $K = 2$ 

"n"	1	2	3	4	5	6	7	8	9	10
3	Y	Y	Y	12	15	18	21	24	27	30

b)  $K = 3$ 

"n"	1	2	3	4	5	6	7	8	9	10
3	Y	Y	Y	Y	Y	Y	12	18	Y	30

c)  $K = 4$ 

"n"	1	2	3	4	5	6	7	8	9	10
3	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	11	12	13	14	15	16	17	18	19	20
	Y	Y	Y	Y	Y	12	18	Y	30	36
	21	22	23	24	25	26	27	28	29	30
	36	48	54	54	66	72	Y	84	87	90

(iii)  $M = 4$ a)  $K = 2$ 

"n"	1	2	3	4	5	6	7	8	9	10
4	Y	Y	4	Y	20	24	28	32	36	40

b)  $K = 3$ 

"n"	1	2	3	4	5	6	7	8	9	10	11
4	Y	Y	Y	Y	Y	Y	Y	Y	4	8	20
	12	13	14	15	16	17	18	19	20		
	16	36	40	52	Y	68	72	76	80		

c)  $K = 4$ 

"n"	1	2	3	4	5	6	7	8	9	10
4	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	11	12	13	14	15	16	17	18	19	20
	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	21	22	23	24	25	26	27	28	29	30
	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	31	32	33	34	35	36	37	38	39	40
	Y	Y	4	8	20	16	36	40	52	32
	41	42	43	44	45	46	47	48	49	50
	68	72	84	80	100	104	116	64	132	136
	51	52	53	54	55	56	57	58	59	60
	148	144	164	168	180	160	196	200	212	208
	61	62	63	64	65	66	67	68	69	70
	228	232	244	Y	260	264	268	272	276	280

(iv)  $M = 5$ a)  $K = 2$ 

"n"	1	2	3	4	5	6	7	8	9	10
5	Y	Y	Y	10	Y	30	35	40	45	50

b)  $K = 3$ 

"n"	1	2	3	4	5	6	7	8	9	10
5	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	11	12	13	14	15	16	17	18	19	20
	Y	Y	Y	10	Y	30	40	50	60	50
	21	22	23	24	25	26	27	28	29	30
	80	90	100	110	Y	130	135	140	145	150

## 4.2 Fix $M$ and $N$ , run $K$

The following results are from 3.3.2.

(i)  $M = 3, N = 3$

"k"	1	2	3	4	5	6	7	8
"Result"	9	Y	Y	Y	Y	Y	Y	Y

a)  $K = 2$ : Solvable

Position	1	2	3	4	5	6	7	8	9
Code	"00"	"01"	"02"	"10"	"11"	"12"	"20"	"21"	"22"

b)  $K = 3$ : Solvable

Position	1	2	3	4	5	6	7	8	9
Code	"000"	"010"	"020"	"100"	"110"	"120"	"200"	"210"	"220"
	"001"	"011"	"021"	"101"	"111"	"121"	"201"	"211"	"221"
	"002"	"012"	"022"	"102"	"112"	"122"	"202"	"212"	"222"

c)  $K = 4$ : Solvable

Position	1	2	3	4	5
	"0000"	"0100"	"0200"	"1000"	"1100"
	"0010"	"0110"	"0210"	"1010"	"1110"
	"0020"	"0120"	"0220"	"1020"	"1120"
	"0001"	"0101"	"0201"	"1001"	"1101"
Code	"0011"	"0111"	"0211"	"1011"	"1111"
	"0021"	"0121"	"0221"	"1021"	"1121"
	"0002"	"0102"	"0202"	"1002"	"1102"
	"0012"	"0112"	"0212"	"1012"	"1112"
	"0022"	"0122"	"0222"	"1022"	"1122"
	6	7	8	9	
	"1200"	"2000"	"2100"	"2200"	
	"1210"	"2010"	"2110"	"2210"	
	"1220"	"2020"	"2120"	"2220"	
	"1201"	"2001"	"2101"	"2201"	
	"1211"	"2011"	"2111"	"2211"	
	"1221"	"2021"	"2121"	"2221"	
	"1202"	"2002"	"2102"	"2202"	
	"1212"	"2012"	"2112"	"2212"	
	"1222"	"2022"	"2122"	"2222"	



(ii)  $M = 3, N = 7$ 

"k"	1	2	3	4	5	6	7	8
"Result"	21	21	12	Y	Y	Y	Y	Y

a)  $K = 3$ : Unsolvable

Position	1	2	3	4	5	6	7	8	9	10	11
Code	"000"	""	""	"011"	""	""	"022"	"100"	""	""	"111"
		12	13	14	15	16	17	18	19	20	21
		""	""	"122"	"200"	""	""	"211"	""	""	"222"

b)  $K = 4$ : Solvable

Position	1	2	3	4	5	6	7
Code	"0000"	"0020"	"0100"	"0110"	"0200"	"0210"	"0220"
	"0001"	"0011"	"0101"	"0111"	"0121"	"0211"	"0221"
	"0002"	"0012"	"0022"	"0112"	"0122"	"0202"	"0222"
	8	9	10	11	12	13	14
	"1000"	"1020"	"1100"	"1110"	"1200"	"1210"	"1220"
	"1001"	"1011"	"1101"	"1111"	"1121"	"1211"	"1221"
	"1002"	"1012"	"1022"	"1112"	"1122"	"1202"	"1222"
	15	16	17	18	19	20	21
	"2000"	"2020"	"2100"	"2110"	"2200"	"2210"	"2220"
	"2001"	"2011"	"2101"	"2111"	"2121"	"2211"	"2221"
	"2002"	"2012"	"2022"	"2112"	"2122"	"2202"	"2222"

(iii)  $M = 4, N = 3$ 

"k"	1	2	3	4	5	6	7
"Result"	12	4	Y	Y	Y	Y	Y

a)  $K = 2$ : Unsolvable

Position	1	2	3	4	5	6	7	8	9
Code	"00"	""	"03"	"10"	""	"13"	"20"	""	"23"
	10	11	12						
	"30"	""	"33"						

b)  $K = 3$ : Solvable

Position	1	2	3	4	5	6
Code	"000"	"020"	"030"	"100"	"120"	"130"
	"010"	"021"	"031"	"110"	"121"	"131"
	"001"	"012"	"032"	"101"	"112"	"132"
	"002"	"013"	"023"	"102"	"113"	"123"
	"003"	"033"	"103"		"133"	
	7	8	9	10	11	12
	"200"	"220"	"230"	"300"	"320"	"330"
	"210"	"221"	"231"	"310"	"321"	"331"
	"201"	"212"	"232"	"301"	"312"	"322"
	"202"	"213"	"223"	"302"	"313"	"323"
	"203"		"233"	"303"		"333"

c)  $K = 4$ : Solvable

Position	1	2	3	4	5	6
Code	"0000"	"0200"	"0300"	"1000"	"1200"	"1300"
	"0100"	"0210"	"0310"	"1100"	"1210"	"1310"
	"0010"	"0120"	"0320"	"1010"	"1120"	"1230"
	"0110"	"0220"	"0230"	"1110"	"1220"	"1330"
	"0020"	"0130"	"0330"	"1020"	"1130"	"1301"
	"0030"	"0201"	"0301"	"1030"	"1201"	"1311"
	"0001"	"0211"	"0311"	"1001"	"1101"	"1211"
	"0101"	"0121"	"0321"	"1101"	"1211"	"1321"
	"0011"	"0221"	"0231"	"1011"	"1121"	"1231"
	"0021"	"0131"	"0331"	"1021"	"1131"	"1331"
	"0031"	"0202"	"0302"	"1031"	"1202"	"1302"
	"0002"	"0112"	"0312"	"1002"	"1112"	"1312"
	"0102"	"0212"	"0322"	"1102"	"1212"	"1322"
	"0012"	"0122"	"0232"	"1012"	"1122"	"1232"
	"0022"	"0132"	"0332"	"1022"	"1132"	"1332"
	"0032"	"0203"	"0303"	"1032"	"1203"	"1303"
	"0003"	"0113"	"0313"	"1003"	"1113"	"1313"
	"0103"	"0213"	"0223"	"1103"	"1213"	"1223"
	"0013"	"0123"	"0323"	"1013"	"1123"	"1323"
	"0023"	"0133"	"0233"	"1023"	"1133"	"1233"
"0033"		"0333"	"1033"		"1333"	
	7	8	9	10	11	12
	"2000"	"2200"	"2300"	"3000"	"3200"	"3300"
	"2100"	"2210"	"2310"	"3100"	"3210"	"3310"
	"2010"	"2120"	"2320"	"3010"	"3120"	"3320"
	"2110"	"2220"	"2230"	"3110"	"3220"	"3230"
	"2020"	"2130"	"2330"	"3020"	"3130"	"3330"
	"2030"	"2201"	"2301"	"3030"	"3201"	"3301"
	"2001"	"2211"	"2311"	"3001"	"3101"	"3211"
	"2101"	"2121"	"2321"	"3101"	"3211"	"3321"
	"2011"	"2221"	"2231"	"3011"	"3121"	"3231"
	"2021"	"2131"	"2331"	"3021"	"3131"	"3331"
	"2031"	"2202"	"2302"	"3031"	"3202"	"3302"
	"2002"	"2212"	"2312"	"3002"	"3102"	"3212"
	"2102"	"2122"	"2322"	"3102"	"3212"	"3322"
	"2012"	"2222"	"2232"	"3012"	"3122"	"3232"
	"2022"	"2132"	"2332"	"3022"	"3132"	"3332"
	"2032"	"2203"	"2303"	"3032"	"3203"	"3303"
	"2003"	"2213"	"2313"	"3003"	"3103"	"3213"
	"2103"	"2123"	"2223"	"3103"	"3213"	"3323"
	"2013"	"2133"	"2233"	"3013"	"3123"	"3233"
	"2023"	"2133"	"2233"	"3023"	"3123"	"3233"
	"2033"		"2333"	"3033"	"3133"	"3333"

(iv)  $M = 4, N = 13$ 

<b>"k"</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>
<b>"Result"</b>	<b>52</b>	<b>52</b>	<b>36</b>	<b>Y</b>	<b>Y</b>	<b>Y</b>	<b>Y</b>

a)  $K = 3$ : Unsolvable

Position	Code						
1	"000"	14	"100"	27	"200"	40	"300"
2	""	15	""	28	""	41	""
3	""	16	""	29	""	42	""
4	""	17	""	30	""	43	""
5	"011"	18	"111"	31	"211"	44	"311"
6	""	19	""	32	""	45	""
7	""	20	""	33	""	46	""
8	""	21	""	34	""	47	""
9	"022"	22	"122"	35	"222"	48	"322"
10	""	23	""	36	""	49	""
11	""	24	""	37	""	50	""
12	""	25	""	38	""	51	""
13	"033"	26	"133"	39	"233"	52	"333"

b)  $K = 4$ : Solvable

Position	Code	""	""	""					
1	"0000"	"0001"	"0002"	"0003"	27	"2000"	"2001"	"2002"	"2003"
2	"0020"	"0011"	"0012"	"0013"	28	"2020"	"2011"	"2012"	"2013"
3	"0030"	"0031"	"0022"	"0023"	29	"2030"	"2031"	"2022"	"2023"
4	"0100"	"0101"	"0102"	"0033"	30	"2100"	"2101"	"2102"	"2033"
5	"0110"	"0111"	"0112"	"0113"	31	"2110"	"2111"	"2112"	"2113"
6	"0130"	"0121"	"0122"	"0123"	32	"2130"	"2121"	"2122"	"2123"
7	"0200"	"0201"	"0132"	"0133"	33	"2200"	"2201"	"2132"	"2133"
8	"0210"	"0211"	"0212"	"0203"	34	"2210"	"2211"	"2212"	"2203"
9	"0220"	"0221"	"0222"	"0223"	35	"2220"	"2221"	"2222"	"2223"
10	"0300"	"0231"	"0232"	"0233"	36	"2300"	"2231"	"2232"	"2233"
11	"0310"	"0311"	"0302"	"0303"	37	"2310"	"2311"	"2302"	"2303"
12	"0320"	"0321"	"0322"	"0313"	38	"2320"	"2321"	"2322"	"2313"
13	"0330"	"0331"	"0332"	"0333"	39	"2330"	"2331"	"2332"	"2333"
14	"1000"	"1001"	"1002"	"1003"	40	"3000"	"3001"	"3002"	"3003"
15	"1020"	"1011"	"1012"	"1013"	41	"3020"	"3011"	"3012"	"3013"
16	"1030"	"1031"	"1022"	"1023"	42	"3030"	"3031"	"3022"	"3023"
17	"1100"	"1101"	"1102"	"1033"	43	"3100"	"3101"	"3102"	"3033"
18	"1110"	"1111"	"1112"	"1113"	44	"3110"	"3111"	"3112"	"3113"
19	"1130"	"1121"	"1122"	"1123"	45	"3130"	"3121"	"3122"	"3123"
20	"1200"	"1201"	"1132"	"1133"	46	"3200"	"3201"	"3132"	"3133"
21	"1210"	"1211"	"1212"	"1203"	47	"3210"	"3211"	"3212"	"3203"
22	"1220"	"1221"	"1222"	"1223"	48	"3220"	"3221"	"3222"	"3223"
23	"1300"	"1231"	"1232"	"1233"	49	"3300"	"3231"	"3232"	"3233"
24	"1310"	"1311"	"1302"	"1303"	50	"3310"	"3311"	"3302"	"3303"
25	"1320"	"1321"	"1322"	"1313"	51	"3320"	"3321"	"3322"	"3313"
26	"1330"	"1331"	"1332"	"1333"	52	"3330"	"3331"	"3332"	"3333"

### 4.3 Fix $K$ , run $M$ and $N$

The following results are from 3.3.3.

#### 4.3.1 $K = 2$

$m \backslash n$	1	2	3	4	5	6	7	8	9	10
2	Y	Y	6	8	10	12	14	16	18	20
3	Y	Y	Y	12	15	18	21	24	27	30
4	Y	Y	4	Y	20	24	28	32	36	40
5	Y	Y	Y	10	Y	30	35	40	45	50
6	Y	Y	Y	Y	18	Y	42	48	54	60
7	Y	Y	Y	Y	14	28	Y	56	63	70
8	Y	Y	Y	Y	8	16	40	Y	72	80
9	Y	Y	Y	Y	Y	Y	36	54	Y	90
10	Y	Y	Y	Y	Y	Y	30	40	70	Y

(i)  $M = 2, N = 2$ : Solvable

Position	1	2	3	4
Code	"00"	"01"	"10"	"11"

(ii)  $M = 3$

a)  $N = 2$ : Solvable

Position	1	2	3	4	5	6
Code	"00"	"02"	"10"	"12"	"20"	"22"

b)  $N = 3$ : Solvable (The result table is in 4.2(i)a)

(iii)  $M = 4$

a)  $N = 2$ : Solvable

Position	1	2	3	4	5	6	7	8
Code	"00"	"02"	"10"	"12"	"20"	"22"	"30"	"32"
	"01"	"03"	"11"	"13"	"21"	"23"	"31"	"33"

b)  $N = 3$ : Unsolvable (The result table is in 4.2(iii)a)

c)  $N = 4$ : Solvable

Position	1	2	3	4	5	6	7	8
Code	"00"	"01"	"02"	"03"	"10"	"11"	"12"	"13"
	9	10	11	12	13	14	15	16
	"20"	"21"	"22"	"23"	"30"	"31"	"32"	"33"

(iv)  $M = 5$

a)  $N = 2$ : Solvable

Position	1	2	3	4	5	6	7	8	9	10
Code	"00"	"03"	"10"	"13"	"20"	"23"	"30"	"33"	"40"	"43"
	"01"	"04"	"11"	"14"	"21"	"24"	"31"	"34"	"41"	"44"

b)  $N = 3$ : Solvable

Position	1	2	3	4	5	6	7	8
Code	"00"	"02"	"04"	"10"	"12"	"14"	"20"	"22"
	9	10	11	12	13	14	15	
	"24"	"30"	"32"	"34"	"40"	"42"	"44"	

c)  $N = 4$ : Unsolvable

Position	1	2	3	4	5	6	7	8	9	10
Code	"00"	""	""	"04"	"10"	""	""	"14"	"20"	""
	11	12	13	14	15	16	17	18	19	20
	""	"24"	"30"	""	""	"34"	"40"	""	""	"44"

d)  $N = 5$ : Solvable

Position	1	2	3	4	5	6	7	8	9
Code	"00"	"01"	"02"	"03"	"04"	"10"	"11"	"12"	"13"
	10	11	12	13	14	15	16	17	18
	"14"	"20"	"21"	"22"	"23"	"24"	"30"	"31"	"32"
	19	20	21	22	23	24	25		
	"33"	"34"	"40"	"41"	"42"	"43"	"44"		

(v)  $M = 6$ a)  $N = 2$ : Solvable

Position	1	2	3	4	5	6
Code	"00"	"03"	"10"	"13"	"20"	"23"
	"01"	"04"	"11"	"14"	"21"	"24"
	"02"	"05"	"12"	"15"	"22"	"25"
	7	8	9	10	11	12
	"30"	"33"	"40"	"43"	"50"	"53"
	"31"	"34"	"41"	"44"	"51"	"54"
	"32"	"35"	"42"	"45"	"52"	"55"

b)  $N = 3$ : Solvable

Position	1	2	3	4	5	6	7	8	9
Code	"00"	"02"	"04"	"10"	"12"	"14"	"20"	"22"	"24"
	"01"	"03"	"05"	"11"	"13"	"15"	"21"	"23"	"25"
	10	11	12	13	14	15	16	17	18
	"30"	"32"	"34"	"40"	"42"	"44"	"50"	"52"	"54"
	"31"	"33"	"35"	"41"	"43"	"45"	"51"	"53"	"55"

c)  $N = 4$ : Solvable

Position	1	2	3	4	5	6	7	8
Code	"00"	"02"	"03"	"05"	"10"	"12"	"13"	"15"
	9	10	11	12	13	14	15	16
	"20"	"22"	"23"	"25"	"30"	"32"	"33"	"35"
	17	18	19	20	21	22	23	24
	"40"	"42"	"43"	"45"	"50"	"52"	"53"	"55"

d)  $N = 5$ : Unsolvable

Position	1	2	3	4	5	6	7	8	9	10
Code	"00"	""	""	""	"05"	"10"	""	""	""	"15"
	11	12	13	14	15	16	17	18	19	20
	"20"	""	""	""	"25"	"30"	""	""	""	"35"
	21	22	23	24	25	26	27	28	29	30
	"40"	""	""	""	"45"	"50"	""	""	""	"55"

e)  $N = 6$ : Solvable

<b>Position</b>	1	2	3	4	5	6	7	8	9
<b>Code</b>	"00"	"01"	"02"	"03"	"04"	"05"	"10"	"11"	"12"
	10	11	12	13	14	15	16	17	18
	"13"	"14"	"15"	"20"	"21"	"22"	"23"	"24"	"25"
	19	20	21	22	23	24	25	26	27
	"30"	"31"	"32"	"33"	"34"	"35"	"40"	"41"	"42"
	28	29	30	31	32	33	34	35	36
	"43"	"44"	"45"	"50"	"51"	"52"	"53"	"54"	"55"

(vi)  $M = 7$

a)  $N = 2$ : Solvable

<b>Position</b>	1	2	3	4	5	6	7
<b>Code</b>	"00"	"04"	"10"	"14"	"20"	"24"	"30"
	"01"	"05"	"11"	"15"	"21"	"25"	"31"
	"02"	"06"	"12"	"16"	"22"	"26"	"32"
	8	9	10	11	12	13	14
	"34"	"40"	"44"	"50"	"54"	"60"	"64"
	"35"	"41"	"45"	"51"	"55"	"61"	"65"
	"36"	"42"	"46"	"52"	"56"	"62"	"66"

b)  $N = 3$ : Solvable

<b>Position</b>	1	2	3	4	5	6	7
<b>Code</b>	"00"	"03"	"05"	"10"	"13"	"15"	"20"
	"01"	"06"	"06"	"11"	"13"	"16"	"21"
	8	9	10	11	12	13	14
	"23"	"25"	"30"	"33"	"35"	"40"	"43"
		"26"	"31"		"36"	"41"	
	15	16	17	18	19	20	21
	"45"	"50"	"53"	"55"	"60"	"63"	"65"
	"46"	"51"		"56"	"61"		"66"

c)  $N = 4$ : Solvable

Position	1	2	3	4	5	6	7
Code	"00"	"02"	"04"	"06"	"10"	"12"	"14"
	8	9	10	11	12	13	14
	"16"	"20"	"22"	"24"	"26"	"30"	"32"
	15	16	17	18	19	20	21
	"34"	"36"	"40"	"42"	"44"	"46"	"50"
	22	23	24	25	26	27	28
	"52"	"54"	"56"	"60"	"62"	"64"	"66"

d)  $N = 5$ : Unsolvable

Position	1	2	3	4	5	6	7
Code	"00"	""	"03"	""	"06"	"10"	""
	8	9	10	11	12	13	14
	"13"	""	"16"	"20"	""	"23"	""
	15	16	17	18	19	20	21
	"26"	"30"	""	"33"	""	"36"	"40"
	22	23	24	25	26	27	28
	""	"43"	""	"46"	"50"	""	"53"
	29	30	31	32	33	34	35
	""	"56"	"60"	""	"63"	""	"66"

e)  $N = 6$ : Unsolvable

Position	Code						
1	"00"	13	"20"	25	"40"	37	"60"
2	""	14	""	26	""	38	""
3	""	15	""	27	""	39	""
4	""	16	""	28	""	40	""
5	""	17	""	29	""	41	""
6	"06"	18	"26"	30	"46"	42	"66"
7	"10"	19	"30"	31	"50"		
8	""	20	""	32	""		
9	""	21	""	33	""		
10	""	22	""	34	""		
11	""	23	""	35	""		
12	"16"	24	"36"	36	"56"		





(i)  $M = 2$ a)  $N = 2$ : Solvable

Position	1	2	3	4
Code	"000" "001"	"010" "011"	"100" "101"	"110" "111"

b)  $N = 3$ : Unsolvable

Position	1	2	3	4	5	6
Code	"000"	""	"011"	"100"	""	"111"

c)  $N = 4$ : Solvable

Position	1	2	3	4	5	6	7	8
Code	"000"	"001"	"010"	"011"	"100"	"101"	"110"	"111"

(ii)  $M = 3$ a)  $N = 2$ : Solvable

Position	1	2	3	4	5	6
Code	"000" "010" "001" "002"	"020" "021" "012" "022"	"100" "110" "101" "102"	"120" "121" "112" "122"	"200" "210" "201" "202"	"220" "221" "212" "222"

b)  $N = 3$ : Solvable (The result table is in 4.2(i)b)c)  $N = 4$ : Solvable

Position	1	2	3	4	5	6
Code	"000" "001"	"010"	"012"	"021" "022"	"100" "101"	"110"
	7	8	9	10	11	12
	"112"	"121" "122"	"200" "201"	"210"	"212"	"221" "222"

d)  $N = 5$ : Solvable

Position	1	2	3	4	5	6	7	8
Code	"000"	"002"	"011"	"020"	"022"	"100"	"102"	"111"
	9	10	11	12	13	14	15	
	"120"	"122"	"200"	"202"	"211"	"220"	"222"	

e)  $N = 6$  : Solvable

Position	1	2	3	4	5	6	7	8	9
Code	"000"	"002"	"010"	"012"	"020"	"022"	"100"	"102"	"110"
	10	11	12	13	14	15	16	17	18
	"112"	"120"	"122"	"200"	"202"	"210"	"212"	"220"	"222"

(iii)  $M = 4$

a)  $N = 2$ : Solvable

Position	1	2	3	4	5	6	7	8
	"000"	"020"	"100"	"120"	"200"	"220"	"300"	"320"
	"010"	"030"	"110"	"130"	"210"	"230"	"310"	"330"
	"001"	"021"	"101"	"121"	"201"	"221"	"301"	"321"
Code	"011"	"031"	"111"	"131"	"211"	"231"	"311"	"331"
	"002"	"022"	"102"	"122"	"202"	"222"	"302"	"322"
	"012"	"032"	"112"	"132"	"212"	"232"	"312"	"332"
	"003"	"023"	"103"	"123"	"203"	"223"	"303"	"323"
	"013"	"033"	"113"	"133"	"213"	"233"	"313"	"333"

b)  $N = 3$ : Solvable (The result table is in 4.2(iii)b)

c)  $N = 4$ : Solvable

Position	1	2	3	4	5	6	7	8
	"000"	"010"	"020"	"030"	"100"	"110"	"120"	"130"
Code	"001"	"011"	"021"	"031"	"101"	"111"	"121"	"131"
	"002"	"012"	"022"	"032"	"102"	"112"	"122"	"132"
	"003"	"013"	"023"	"033"	"103"	"113"	"123"	"133"
	9	10	11	12	13	14	15	16
	"200"	"210"	"220"	"230"	"300"	"310"	"320"	"330"
	"201"	"211"	"221"	"231"	"301"	"311"	"321"	"331"
	"202"	"212"	"222"	"232"	"302"	"312"	"322"	"332"
	"203"	"213"	"223"	"233"	"303"	"313"	"323"	"333"

d)  $N = 5$ : Solvable

Position	1	2	3	4	5	6	7	8	9	10
Code	"000" "001" "002"	"010" "011"	"020" "013"	"022" "023"	"031" "032" "033"	"100" "101" "102"	"110" "111"	"120" "113"	"122" "123"	"131" "132" "133"
	11	12	13	14	15	16	17	18	19	20
	"200" "201" "202"	"210" "211"	"220" "213"	"222" "223"	"231" "232" "233"	"300" "301" "302"	"310" "311"	"320" "313"	"322" "323"	"331" "332" "333"

e)  $N = 6$ : Solvable

Position	1	2	3	4	5	6	7	8
Code	"000" "001"	"010" "003"	"012" "013"	"020" "021"	"030" "023"	"032" "033"	"100" "101"	"110" "103"
	9	10	11	12	13	14	15	16
	"112" "113"	"120" "121"	"130" "123"	"132" "133"	"200" "201"	"210" "203"	"212" "213"	"220" "221"
	17	18	19	20	21	22	23	24
	"230" "223"	"232" "233"	"300" "301"	"310" "303"	"312" "313"	"320" "321"	"330" "323"	"332" "333"

f)  $N = 7$ : Solvable

Position	Code	""		
1	"000"	"001"	15	"200"
2	"003"	""	16	"203"
3	"011"	""	17	"211"
4	"020"	"013"	18	"220"
5	"022"	""	19	"222"
6	"030"	""	20	"230"
7	"032"	"033"	21	"232"
8	"100"	"101"	22	"300"
9	"103"	""	23	"303"
10	"111"	""	24	"311"
11	"120"	"113"	25	"320"
12	"122"	""	26	"322"
13	"130"	""	27	"330"
14	"132"	"133"	28	"332"
				"333"

g)  $N = 8$ : Solvable

Position	Code	""			
1	"000"	"001"	17	"200"	"201"
2	"002"	"003"	18	"202"	"203"
3	"010"	"011"	19	"210"	"211"
4	"012"	"013"	20	"212"	"213"
5	"020"	"021"	21	"220"	"221"
6	"022"	"023"	22	"222"	"223"
7	"030"	"031"	23	"230"	"231"
8	"032"	"033"	24	"232"	"233"
9	"100"	"101"	25	"300"	"301"
10	"102"	"103"	26	"302"	"303"
11	"110"	"111"	27	"310"	"311"
12	"112"	"113"	28	"312"	"313"
13	"120"	"121"	29	"320"	"321"
14	"122"	"123"	30	"322"	"323"
15	"130"	"131"	31	"330"	"331"
16	"132"	"133"	32	"332"	"333"

h)  $N = 9$ : Unsolvable

Position	Code				
1	"000"	13	"112"	25	"223"
2	"002"	14	""	26	"231"
3	"010"	15	"121"	27	"233"
4	"012"	16	"123"	28	"300"
5	""	17	"131"	29	"302"
6	"021"	18	"133"	30	"310"
7	"023"	19	"200"	31	"312"
8	"031"	20	"202"	32	""
9	"033"	21	"210"	33	"321"
10	"100"	22	"212"	34	"323"
11	"102"	23	""	35	"331"
12	"110"	24	"221"	36	"333"

(iv)  $M = 5$ a)  $N = 13$ : Solvable

Position	Code								
1	"000"	14	"100"	27	"200"	40	"300"	53	"400"
2	"002"	15	"102"	28	"202"	41	"302"	54	"402"
3	"004"	16	"104"	29	"204"	42	"304"	55	"404"
4	"011"	17	"111"	30	"211"	43	"311"	56	"411"
5	"013"	18	"113"	31	"213"	44	"313"	57	"413"
6	"020"	19	"120"	32	"220"	45	"320"	58	"420"
7	"022"	20	"122"	33	"222"	46	"322"	59	"422"
8	"024"	21	"124"	34	"224"	47	"324"	60	"424"
9	"031"	22	"131"	35	"231"	48	"331"	61	"431"
10	"033"	23	"133"	36	"233"	49	"333"	62	"433"
11	"040"	24	"140"	37	"240"	50	"340"	63	"440"
12	"042"	25	"142"	38	"242"	51	"342"	64	"442"
13	"044"	26	"144"	39	"244"	52	"344"	65	"444"

b)  $N = 14$ : Unsolvable

Position	Code								
1	"000"	15	"100"	29	"200"	43	"300"	57	"400"
2	"002"	16	"102"	30	"202"	44	"302"	58	"402"
3	"004"	17	"104"	31	"204"	45	"304"	59	"404"
4	"011"	18	"111"	32	"211"	46	"311"	60	"411"
5	""	19	""	33	""	47	""	61	""
6	"014"	20	"114"	34	"214"	48	"314"	62	"414"
7	"021"	21	"121"	35	"221"	49	"321"	63	"421"
8	"023"	22	"123"	36	"223"	50	"323"	64	"423"
9	"030"	23	"130"	37	"230"	51	"330"	65	"430"
10	""	24	""	38	""	52	""	66	""
11	"033"	25	"133"	39	"233"	53	"333"	67	"433"
12	"040"	26	"140"	40	"240"	54	"340"	68	"440"
13	"042"	27	"142"	41	"242"	55	"342"	69	"442"
14	"044"	28	"144"	42	"244"	56	"344"	70	"444"

c)  $N = 15$ : Solvable

Position	Code								
1	"000"	16	"100"	31	"200"	46	"300"	61	"400"
2	"002"	17	"102"	32	"202"	47	"302"	62	"402"
3	"004"	18	"104"	33	"204"	48	"304"	63	"404"
4	"010"	19	"110"	34	"210"	49	"310"	64	"410"
5	"012"	20	"112"	35	"212"	50	"312"	65	"412"
6	"014"	21	"114"	36	"214"	51	"314"	66	"414"
7	"020"	22	"120"	37	"220"	52	"320"	67	"420"
8	"022"	23	"122"	38	"222"	53	"322"	68	"422"
9	"024"	24	"124"	39	"224"	54	"324"	69	"424"
10	"030"	25	"130"	40	"230"	55	"330"	70	"430"
11	"032"	26	"132"	41	"232"	56	"332"	71	"432"
12	"034"	27	"134"	42	"234"	57	"334"	72	"434"
13	"040"	28	"140"	43	"240"	58	"340"	73	"440"
14	"042"	29	"142"	44	"242"	59	"342"	74	"442"
15	"044"	30	"144"	45	"244"	60	"344"	75	"444"

# Chapter V

## CONCLUSION AND DISCUSSION

In this project, we provide an instant program as a tool to study Gardner's problem. The program can tell whether a Gardner's problem with given parameters is solvable or unsolvable. For a solvable problem, all possible ways to perform the Gardner's trick are provided. We also provide 3 extra programs for an overview of solvable cases and unsolvable cases. Given the number of piles and the number of cards in each pile, the first program can give some values of the given number of rounds that make the Gardner's problem solvable or unsolvable. Given the number of piles and the number of rounds, the second program can give some values of the given number of cards in each pile that make the Gardner's problem solvable or unsolvable. Given the number of rounds of performing, the program can give some values of the given number of piles and some values of the given number of cards in each pile that make the Gardner's problem solvable or unsolvable. Moreover, for each unsolvable problem, the number of desired position  $x$  that cannot be performed are also provided.

From our main result, for any  $M, N, K \in \mathbb{N}$ ,  $M \geq 2$ , we can conclude conditions for the parameters that make the Gardner's problem solvable or unsolvable as follows.

(i)  $N > M^{K-1}$

The Gardner's problem is unsolvable. This is discussed in 3.1.1. According to our results, we conjecture the number of unusable positions in each case is  $MN$ .

(ii)  $N = M^{K-1}$

The Gardner's problem is solvable. For the number of deck is exactly  $M^K$ , this condition have been reveal in *Mathematics Card Tricks* [2]. Furthermore, we know a way to perform the desired position  $x$  by converting  $x - 1$  to base

$M$  as a  $K$  digit performing code.

(iii)  $N < M^{K-1}$

From our experiments, we conjecture the following results.

- a) If  $N = M^{K-1} - 1$ , then the Gardner's problem is unsolvable. Moreover, the number of unusable positions in such case is  $M(M^{K-1} - 3)$ , for every  $M^{K-1} \geq 3$ .
- b) If  $M$  is an even number and  $N \leq \frac{M^{K-1}}{2}$ , then the Gardner's problem is solvable.
- c) If  $M$  is an odd number and  $N \leq \left\lfloor \frac{M^{K-1}}{2} \right\rfloor + 1$ , then the Gardner's problem is solvable. Moreover, if  $M$  is divisible by 3 and  $N \leq \left\lfloor \frac{M^{K-1}}{2} \right\rfloor + 2$ , then the Gardner's problem is solvable.

For  $\left\lfloor \frac{M^{K-1}}{2} \right\rfloor < N < M^{K-1}$ , most sets of parameters  $M, N, K$  are unsolvable. However, there are some interesting cases that are solvable. The relationship condition that makes those sets of parameters solvable still remains open for future works.

Although we have an easy way to perform the Gardner's trick for a deck of size  $n^k$ , we did not study about how to perform the trick for a person when we actually get the desired position from a spectator in real life when we actually get the desired position from a spectator, since we need a computer to tell us the code to perform. We hope that this instant program and the anticipated conditions can be useful for study the Gardner's problem in future works.



## REFERENCES

- [1] Bolker, E. D. (2010). *Gergonne's Card Trick, Positional Notation, and Radix Sort*, Mathematics Magazine, Volume 830 : 46-49. Available from: <https://www.maa.org/sites/default/files/Bolker-MMz-201053228.pdf>.
- [2] Chang, P. (2015). *Mathematical Card Tricks*, Harvard University [Online]. Available from: [https://scholar.harvard.edu/files/peter\\_chang\\_portfolio/files/mathematical-card-tricks.pdf](https://scholar.harvard.edu/files/peter_chang_portfolio/files/mathematical-card-tricks.pdf)
- [3] Dickson, L. E. (1895). *Gergonne's pile problem*, Bull. Amer. Math. Soc. 1, 184–186, <http://projecteuclid.org/euclid.bams/1183414376>.
- [4] Gardner, M. (1956). *Mathematics Magic and Mystery*, Dover Publications Inc., Mineola, N. Y.
- [5] Gergonne, J. D. (1813). *R'ecr'eatons Math'ematiques: Recherches sur un tour de cartes*, Annales de Math'ematiques Pures et Appliqu'ees, 4 : 276–283.
- [6] Quintero, R. (2000). *The Gergonne m-pile trick and the base m counting system* [Online]. Available from: [http://icm2006.mathunion.org/v\\_f/AbsDef/Shorts/abs\\_0799.pdf](http://icm2006.mathunion.org/v_f/AbsDef/Shorts/abs_0799.pdf).
- [7] *Wikipedia: Wolfram Language*, Wikipedia Foundation Inc. Encyclopedia online [Online]. Available from: [https://en.wikipedia.org/wiki/Wolfram\\_Language](https://en.wikipedia.org/wiki/Wolfram_Language).
- [8] Yang, S. (2020). *Martin Gardner's 27-Card Trick on Numberphile in Slow Motion*, Wolfram Community, Wolfram [Online]. Available from: <https://community.wolfram.com/groups/-/m/t/1909911>.

# Appendix I

The Project Proposal of course 2301399

Project Proposal

Academic Year 2018

Project Tittle (Thai)	การขยายรูปแบบกลไพ่ของมาร์ตินการ์ดเนอร์
Project Tittle (English)	An Extension of Martin Gardner's Card Trick
Project Advisor	Raywat Tanadkithirun Ph.D.
By	Athisa Laungvarunyoo ID 5833552723 Mathematics, Department of Mathematics and Computer Science Faculty of Science, Chulalongkorn University

---

## A.1 Background and Rationale

The Gergonne  $p$ -pile problem [5] was first proposed by Gergonne in 1813. At first, a spectator select any card in a deck, memorizes it and shuffles the deck as much as desired. We creates three face-up piles of nine cards, dealing the top card onto the left, middle, then right deck and repeating until all the cards are gone. Now we get 3 piles card. Next, the participant then points to which pile contains the chosen card. The problem involves distributing these cards repeated this same proceses many times. Finally, we can make the middle of the deck to be the chosen card. During the past century the trick was included in books on recreational mathematics. In *Mathematics, Magic and Mystery* [4], Martin Gardner evolved Gergonne  $p$ -pile problem to the 27 card trick. In his trick is like the Gergonne  $p$ -pile problem, but this time the spectator can choose the positon between 1 and 27 of deck that they want the chosen card to be (call this number  $n$ ). After the procedure with 3 piles

and 3 times assembly, we can reveal the  $n$ th card to be position of the chosen card in the packet of 27.

Dickson [3] generalized 27 card trick to the trick for any deck  $n^n$  cards which the pack is dealt into  $n$  piles of  $n^{n-1}$  cards each. Finally, after the  $n^{th}$  times procedure the card selected will be the  $n^{th}$  from the top. Moreover, in general, if you deal a pack of  $n^k$  cards into  $n$  piles and repeat this procedure  $k$  times, you can place the card at any desired position in the deck. Recently, Bolker [1] generalized this trick by using radix sort. Using a mixed radix you can do Gergonne's trick with a deck of any cards you wish. Example, 54 cards, first time, the pack is dealt into 6 piles, second times and third times is dealt into 3 piles. So, it is interesting to find the others number of deck which not in form of  $n^k$  cards that can play Martin Gardner's card trick without mixed radix method.

## A.2 Objectives

To search for some solution of number of cards ( $MN$  cards) that can be dealt to  $M$  piles and  $N$  cards each and number of times ( $K$  times) to do the procedure which can play Martin Gardner's card trick.

## A.3 Scope

In this study, we only consider a number of packets of cards with composite number and the minimum number of times that can play Martin Gardner's card trick.

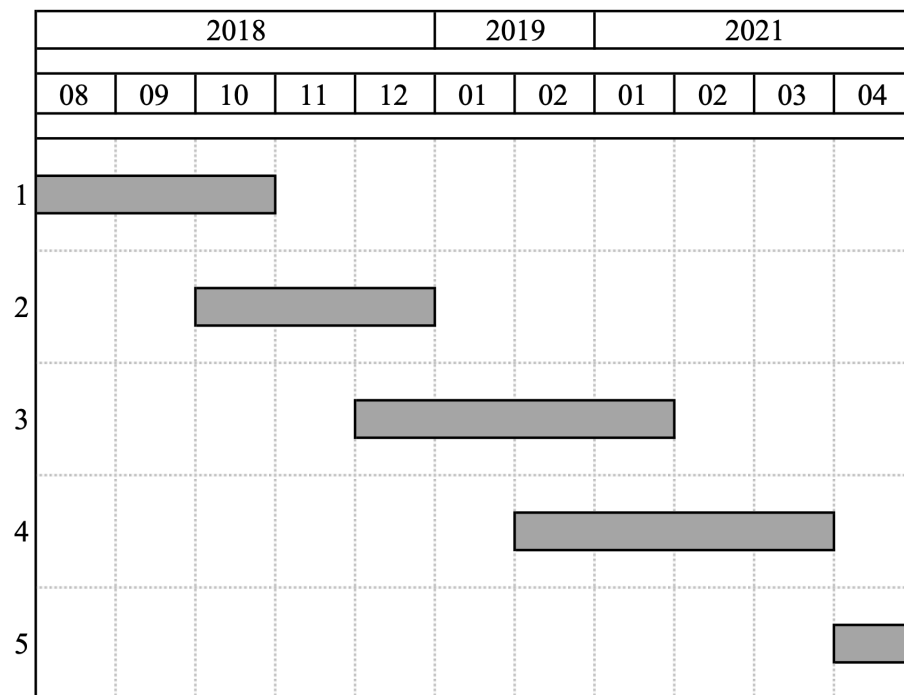
## A.4 Project Activities

1. Study Martin Gardner's card trick and other related study from [5], [6] and other.
2. Select the scope for study and consider algorithm of Martin Gardner's card trick.

3. Write code in Mathematica program.
4. Write a report.
5. Present the project.

### A.5 Durations

1. Study Martin Gardner's card trick and other related study.
2. Select the scope for study and consider algorithm of Martin Gardner's card trick.
3. Write code in Mathematica program.
4. Write a report.
5. Present the project.



## **A.6 Benefits**

1. The benefits for student who implement this project.
  - Learn and understand sorting algorithm.
  - Improve Mathematica coding skill.
2. The benefits for users of the project.
  - Know how to play Gardner's trick with others number of cards.
  - Understand mathematics after a magic trick.

## **A.7 Equipment**

1. Computer
  - i. Microsoft word
  - ii. Adobe PDF
  - iii. Wolfram Mathematica
2. Deck of cards

## Biography



Athisa Laungvarunyoo

Student ID: 583 35527 23

Place of Birth: Bangkok, Thailand

Field of Study: Mathematics

Department of Mathematics and Computer Science

Faculty of Science, Chulalongkorn University