

Constructions for Retransmission Permutation Arrays

Jeffrey H. Dinitz
Mathematics and Statistics
University of Vermont
Burlington VT, 05405, USA
Jeff.Dinitz@uvm.edu

Maura B. Paterson
Economics, Mathematics and Statistics
Birkbeck, University of London
Malet Street, London WC1E 7HX, UK
m.paterson@bbk.ac.uk

Douglas R. Stinson*
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo ON, N2L 3G1, Canada
dstinson@uwaterloo.ca

Ruizhong Wei†
Department of Computer Science
Lakehead University
Thunder Bay ON, P7B 5E1, Canada
rwei@lakeheadu.ca

October 26, 2010

Abstract

Li, Liu, Tan, Viswanathan, and Yang [3] introduced a technique for resolving overlapping channel transmissions that used an interesting new type of combinatorial structure. In connection with this problem, they provided an example of a 4×4 array having certain desirable properties. We define a class of combinatorial structures, which we term *retransmission permutation arrays*, that generalise the example that Li *et al.* provided. We show that these arrays exist for all possible orders. We also define some extensions having additional properties, for which we provide some partial results.

1 Introduction

Li, Liu, Tan, Viswanathan, and Yang [3] introduced a technique for resolving overlapping channel transmissions that used an interesting new type of combinatorial structure. In connection with this problem, they provided an example of a 4×4 array having certain desirable properties. We briefly describe the method used in [3] in order to motivate the combinatorial problem we study in this paper.¹

Suppose there is a sequence of *groups*, denoted G_1, G_2, \dots , each of which is just a set of carrier frequencies. Let n be a positive integer ([3] uses $n = 4$). A *channel* C_i is able to access n consecutive groups, i.e., C_i can access $G_i, G_{i+1}, \dots, G_{i+n-1}$, $i = 1, 2, \dots$. Two channels are *overlapping* if they contain one or more common groups.

*Research supported by NSERC discovery grant 203114-06

†Research supported by NSERC discovery grant 239135-06

¹This section can be skipped by readers who are not interested in the practical motivation for studying this problem.

Each channel can transmit an n -bit message by assigning one bit to each of the n groups it can access. However, if overlapping channels are simultaneously used to transmit two messages, then one or more of the bits in the messages will *collide* and therefore they will not be transmitted successfully. More precisely, if two channels overlap in j groups, then j bits of their respective messages will collide. The consequence is that messages will need to be retransmitted in order that all the bits are received successfully.

The main idea in [3] is that each retransmission should use a different assignment of bits to groups. Since each assignment can be viewed as a permutation of the n bits, the schedule of assignments can be termed a *retransmission permutation array* (or *RPA*). We will give a mathematical definition of this term in the next subsection, once we have clarified certain additional properties required by these arrays.

Example 1.1. A retransmission permutation array for $n = 4$ (from [3]):

1	2	3	4
4	3	2	1
2	1	4	3
3	4	1	2

The RPA presented in Example 1.1 consists of four permutations: 1234, 4321, 2143 and 3412. Each permutation corresponds to a transmission of four bits $b_1b_2b_3b_4$ over a channel C_i , as follows:

- In the first transmission, b_1 is assigned to G_i , b_2 is assigned to G_{i+1} , b_3 is assigned to G_{i+2} , and b_4 is assigned to G_{i+3} .
- In the second transmission, b_4 is assigned to G_i , b_3 is assigned to G_{i+1} , b_2 is assigned to G_{i+2} , and b_1 is assigned to G_{i+3} .
- In the third transmission, b_2 is assigned to G_i , b_1 is assigned to G_{i+1} , b_4 is assigned to G_{i+2} , and b_3 is assigned to G_{i+3} .
- In the fourth transmission, b_3 is assigned to G_i , b_4 is assigned to G_{i+1} , b_1 is assigned to G_{i+2} , and b_2 is assigned to G_{i+3} .

Suppose a simultaneous transmission takes place by a channel which overlaps the given channel in j groups. The j groups could be the first j groups or the last j groups among the $n = 4$ groups. The bits that are successfully received are the bits in the last $n - j$ columns (or the first $n - j$ columns, respectively) of the array. In Example 1.1, the following properties can be verified easily:

- If there is no overlap, then all four bits are received after one transmission.
- If there is an overlap in one or two groups, then all four bits are received after two transmissions.
- If there is an overlap in three groups, then all four bits are received after four transmissions.

In general, if a channel consists of n groups and two channels overlap in j groups, then we will need at least $\tau = \lceil \frac{n}{n-j} \rceil$ transmissions in order to receive all n bits. This number of transmissions will be sufficient if and only if every symbol occurs at least once in the upper left $\tau \times (n - j)$ rectangle of the RPA (when the overlap is in the last j groups), or every symbol occurs at least once in the upper right $\tau \times (n - j)$ rectangle of the RPA (when the overlap is in the first j groups).

1.1 Combinatorial Definitions

A *type 1 retransmission permutation array of order n* (denoted *type-1 RPA(n)*) is an $n \times n$ array, say A , in which each cell contains a symbol from the set $\{1, \dots, n\}$, such that the following properties are satisfied:

- (i) every row of A contains all n symbols (i.e., every row of A is a permutation of the n symbols), and
- (ii) for $1 \leq i \leq n$, the $i \times \lceil \frac{n}{i} \rceil$ rectangle in the upper left hand corner of A contains all n symbols.

Here are some variations of the above definition.

- If property (ii) is modified so it instead holds for rectangles in the upper right corner of A , then we say that the array is a *type 2 retransmission permutation array of order n* (denoted *type-2 RPA(n)*).
- A *type 3 retransmission permutation array of order n* (denoted *type-3 RPA(n)*) is one in which (a modified) property (ii) holds for rectangles in the lower left corner of A .
- Finally, a *type 4 retransmission permutation array of order n* (denoted *type-4 RPA(n)*) is one in which (a modified) property (ii) holds for rectangles in the lower right corner of A .

The problem introduced in [3] asks for arrays that are simultaneously *type-1 RPA(n)* and *type-2 RPA(n)*; we will term these *type-1, 2 RPA(n)*. This notation will be generalised in the obvious way to arrays that satisfy different combinations of variations of property (ii). The array presented in Example 1.1 is a *type-1, 2 RPA(4)*; in fact, it is a *type-1, 2, 3, 4 RPA(4)*.

An $r \times \lceil \frac{n}{r} \rceil$ rectangle is called *basic* if it does not contain an $r' \times \lceil \frac{n}{r'} \rceil$ rectangle where $r' < r$ and $\lceil \frac{n}{r} \rceil = \lceil \frac{n}{r'} \rceil$. For example, when $n = 12$, a 5×3 rectangle is not basic because it contains a 4×3 rectangle and $\lceil \frac{12}{5} \rceil = \lceil \frac{12}{4} \rceil = 3$.

In verifying property (ii), it suffices to consider only basic rectangles. The basic rectangles that must be verified in Example 1.1 have dimensions 1×4 , 2×2 and 4×1 .

An *RPA*, A , of any type is called *latin* if every column of A contains all n symbols (i.e., every column of A is a permutation of the n symbols). A *latin RPA* is in fact a *latin square* of order n . Observe that the *type-1, 2, 3, 4 RPA(4)* constructed in Example 1.1 is *latin*. The application in [3] does not require *latin RPAs*; however, it seems to be a natural and interesting combinatorial problem to investigate the existence of *latin RPAs*.

1.2 Outline of the Paper

Here we summarise the results in the remaining sections of the paper. In Section 2, we discuss relationships between different types of *RPAs*. In Section 3, we present an algorithm which proves that *type-1 RPA(n)* exist for all $n \geq 1$. In Section 4, we extend our algorithm so it will construct *type-1, 2 RPA(n)* for all n . We present a few results on *latin RPAs* in Section 5. Finally, Section 6 mentions some open problems.

The main existence results proven in the paper are summarised in Table 1.

Table 1: Existence results for retransmission permutation arrays

type of RPA	existence result	authority
type-1 $RPA(n)$	all integers $n \geq 1$	Theorem 3.1
type-1, 2 $RPA(n)$	all integers $n \geq 1$	Corollary 4.10
type-1, 3 $RPA(n)$	all integers $n \geq 1$	Theorem 3.2
type-1, 4 $RPA(n)$	all integers $n \geq 1$	Theorem 3.3
type-1, 2, 3, 4 $RPA(n)$	all even integers $n \geq 2$	Corollary 4.8
type-1, 2, 3, 4 $LRPA(n)$	even integers $n \leq 16, n = 36$	Section 5
type-1, 2, 3, 4 $LRPA(n)$	odd integers $n \leq 9$	Section 5

1.3 Related Work

There is some previous work concerning latin squares in which certain rectangular regions contain every symbol at most (or exactly) once. We mention two examples that have been studied in the mathematical literature.

1. Some algorithms for providing conflict-free access to parallel memories (see, for example, Colbourn and Heinrich [1, 2]) seek latin squares of order n where, for a certain fixed value of s , every $s \times \lfloor \frac{n}{s} \rfloor$ rectangle in the latin square contains every symbol at most once.
2. A *Sudoku square* is a latin square of order n , where $n = m^2$, such that it can be partitioned into n square subarrays of side m such that every one of these subarrays contains all n symbols. The case $n = 9$ corresponds to the popular Sudoku puzzles commonly found in newspapers. It has also been observed (see [4]) that Sudoku squares are examples of *gerechte designs* which are used in agricultural experiments.

2 Relationship Between Different Types of RPA s

The next five theorems are obvious, following by reflections through a vertical or horizontal axis.

Theorem 2.1. *The existence of the following are equivalent: a type-1 $RPA(n)$, a type-2 $RPA(n)$, a type-3 $RPA(n)$ and a type-4 $RPA(n)$.*

Theorem 2.2. *The existence of the following are equivalent: a type-1,2 $RPA(n)$ and a type-3,4 $RPA(n)$.*

Theorem 2.3. *The existence of the following are equivalent: a type-1,3 $RPA(n)$ and a type-2,4 $RPA(n)$.*

Theorem 2.4. *The existence of the following are equivalent: a type-1,4 $RPA(n)$ and a type-2,3 $RPA(n)$.*

Theorem 2.5. *The existence of the following are equivalent: a type-1,2,3 $RPA(n)$, a type-1,2,4 $RPA(n)$, a type-1,3,4 $RPA(n)$ and a type-2,3,4 $RPA(n)$.*

Now we prove some results establishing that, when n is even, *RPA*s of certain types can be “extended” to yield *RPA*s of stronger types.

Theorem 2.6. *If n is even and there exists a type-1 $RPA(n)$, then there exists a type-1,3 $RPA(n)$.*

Proof. Let $A = (a_{i,j})$ be a type-1 $RPA(n)$ on symbol set $\{1, \dots, n\}$, where n is even, and let X denote the set of symbols in the first $n/2$ cells in the first column of A (note that these $n/2$ symbols are distinct). Denote $Y = \{1, \dots, n\} \setminus X$. Let $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ be any bijection in which $\pi(X) = Y$ (hence it follows that $\pi(Y) = X$). Now define an $n \times n$ array $B = (b_{i,j})$ as follows:

$$b_{i,j} = \begin{cases} a_{i,j} & \text{if } i \leq \frac{n}{2} \\ \pi(a_{n+1-i,j}) & \text{if } i > \frac{n}{2}. \end{cases}$$

It is easy to verify that B is a type-1,3 $RPA(n)$. □

Theorem 2.7. *If n is even and there exists a type-1,2 $RPA(n)$, then there exists a type-1,2,3,4 $RPA(n)$.*

Proof. This is an extension of the proof of Theorem 2.6. Let $A = (a_{i,j})$ be a type-1,2 $RPA(n)$ on symbol set $\{1, \dots, n\}$, where n is even. Define X and Y as in the proof of Theorem 2.6. Let Z denote the set of symbols in the first $n/2$ cells in the last column of A (note that these $n/2$ symbols are distinct) and denote $W = \{1, \dots, n\} \setminus Z$. We want to find a bijection $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ such that $\pi(X) = Y$ and $\pi(Z) = W$. This is easily done as follows: Define $Z' = Z \cap X$, $Z'' = Z \cap Y$, $W' = W \cap X$ and $W'' = W \cap Y$. Then it is easy to verify that $|W'| = |Z''| = n/2 - |Z'|$ and $|W''| = |Z'|$. Therefore π can be defined from any bijection from Z' to W'' together with any bijection from Z'' to W' . Now define an $n \times n$ array $B = (b_{i,j})$ as follows:

$$b_{i,j} = \begin{cases} a_{i,j} & \text{if } i \leq \frac{n}{2} \\ \pi(a_{n+1-i,j}) & \text{if } i > \frac{n}{2}. \end{cases}$$

It is easy to verify that B is a type-1,2,3,4 $RPA(n)$. □

Theorem 2.8. *If n is even and there exists a type-1 $RPA(n)$, then there exists a type-1,4 $RPA(n)$.*

Proof. This is a slight modification of the proof of Theorem 2.7. Let $A = (a_{i,j})$ be a type-1 $RPA(n)$ on symbol set $\{1, \dots, n\}$, where n is even. We require that the first $n/2$ cells in the last column of A contain distinct symbols. For an arbitrary type-1 $RPA(n)$, this might not occur; however, we can easily transform any type-1 $RPA(n)$ into a type-1 $RPA(n)$ in which this property does hold. This is done as follows. For each row i , $2 \leq i \leq n/2$, if $a_{i,n} \in \{a_{1,n}, \dots, a_{i-1,n}\}$, then we find a symbol $a_{i,j}$ (for some j , $n/2 \leq j \leq n-1$) such that $a_{i,j} \notin \{a_{1,n}, \dots, a_{i-1,n}\}$ (such an $a_{i,j}$ exists by the pigeon-hole principle). Then swap $a_{i,n}$ and $a_{i,j}$. Observe that the swapped symbols do not lie in any basic rectangle, so the transformed square is still a type-1 $RPA(n)$.

Now define X, Y, Z, W, Z', Z'', W' and W'' as in the proof of Theorem 2.7. Note that $|Z| = n/2$ due to the transformation carried out above.

We want to find a bijection $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ such that $\pi(X) = W$ and $\pi(Z) = Y$. This is easily done by defining any bijection from Z' to W'' and then fixing both Z'' and W' setwise. Now define an $n \times n$ array $B = (b_{i,j})$ as follows:

$$b_{i,j} = \begin{cases} a_{i,j} & \text{if } i \leq \frac{n}{2} \\ \pi(a_{n+1-i,n+1-j}) & \text{if } i > \frac{n}{2}. \end{cases}$$

It is easy to verify that B is a type-1, 4 $RPA(n)$. □

In the next two corollaries, which are immediate consequences of Theorems 2.1–2.7, we observe that, when n is even, a type-1 $RPA(n)$ yields RPA s of various types and a type-1, 2 $RPA(n)$ yields RPA s of all remaining types.

Corollary 2.9. *If n is even and there exists a type-1 $RPA(n)$, then the following arrays exist: a type-1 $RPA(n)$, a type-2 $RPA(n)$, a type-3 $RPA(n)$, a type-4 $RPA(n)$, a type-1, 3 $RPA(n)$, a type-1, 4 $RPA(n)$, a type-2, 3 $RPA(n)$ and a type-2, 4 $RPA(n)$.*

Corollary 2.10. *If n is even and there exists a type-1, 2 $RPA(n)$, then the following arrays exist: a type-1, 2 $RPA(n)$, a type-3, 4 $RPA(n)$, a type-1, 2, 3 $RPA(n)$, a type-1, 2, 4 $RPA(n)$, a type-1, 3, 4 $RPA(n)$, a type-2, 3, 4 $RPA(n)$ and a type-1, 2, 3, 4 $RPA(n)$.*

We do not have analogues of Theorems 2.6–2.8 for odd n . However, we are able to modify these two theorems so they hold for odd n , provided that the “input” type-1 $RPA(n)$ satisfies a certain property. We will revisit this at the end of Section 3.

3 A Construction for type-1 $RPA(n)$

For any positive integer n , the algorithm presented in Figure 1 constructs a type-1 $RPA(n)$ which we denote by A . This algorithm considers the basic rectangles one at a time, in increasing order of the number of rows they have.

We say that an $r \times \lceil \frac{n}{r} \rceil$ basic rectangle R is *canonical* if the following property is satisfied:

- (*) There is a partition $n = a_1 + \dots + a_r$ where $\lceil \frac{n}{r} \rceil \geq a_1 \geq \dots \geq a_r > 0$, such that the union of the first a_i cells of row i (for $1 \leq i \leq r$) contains every symbol in $\{1, \dots, n\}$ exactly once².

Suppose there are b basic rectangles. After step k of the algorithm (where $1 \leq k \leq b$), the k th basic rectangle will be canonical. After step b is completed, we fill in the remaining empty cells in A .

Example 3.1 illustrates the algorithm presented in Figure 1.

Example 3.1. *Suppose $n = 7$. The basic rectangles have dimensions 1×7 , 2×4 , 3×3 , 4×2 , and 7×1 .*

step 1

We begin by filling in the 1×7 basic rectangle (i.e., the first row of A):

1	2	3	4	5	6	7
---	---	---	---	---	---	---

step 2

Next, we consider the 2×4 basic rectangle. We place the symbols 5, 6, 7 in the first three cells of the second row of this rectangle:

1	2	3	4	5	6	7
5	6	7				

²This decomposition of n is analogous to a Ferrers diagram.

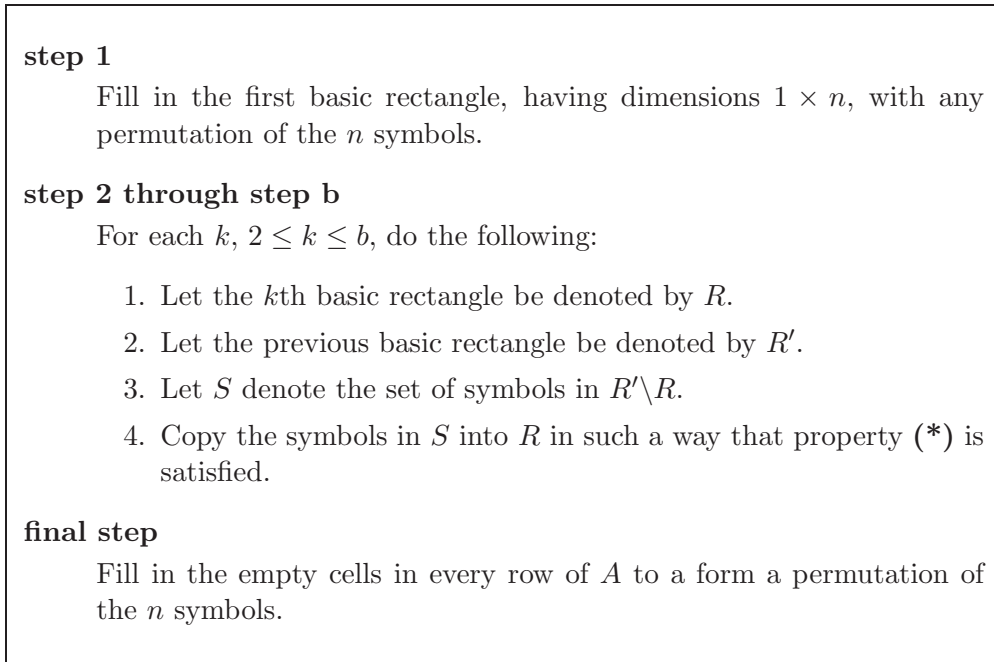


Figure 1: Algorithm to construct a type-1 $RPA(n)$.

step 3

Now we turn to the 3×3 basic rectangle, filling in the first cell of the third row with the symbol 4:

1	2	3	4	5	6	7
5	6	7				
4						

step 4

Next, we look at the 4×2 basic rectangle. We have to fill in the symbols 3 and 7:

1	2	3	4	5	6	7
5	6	7				
4	3					
7						

step 5

The last basic rectangle has dimensions 7×1 . It is completed by filling in the symbols 2,6

and 3 into the first cells in the last three rows:

1	2	3	4	5	6	7
5	6	7				
4	3					
7						
2						
6						
3						

step 6

Finally, we fill in all remaining cells in such a way that each row is a permutation, for example,

1	2	3	4	5	6	7
5	6	7	1	2	3	4
4	3	1	2	5	6	7
7	1	2	3	4	5	6
2	1	3	4	5	6	7
6	1	2	3	4	5	7
3	1	2	4	5	6	7

Theorem 3.1. For all integers $n \geq 1$, there exists a type-1 RPA(n).

Proof. We prove that the Algorithm 1 is correct, i.e., that it always produces a type-1 RPA(n). To do this, we first observe that each basic rectangle R is canonical immediately after it is considered in the relevant step of the algorithm. Because each symbol occurs only once in a basic rectangle R' , it follows that when the symbols in S are copied from R' to the next basic rectangle R , there are a sufficient number of empty cells in R to hold the symbols in S .

Also due to the canonicity property, it follows that, whenever a symbol is copied from a basic rectangle R' to the next basic rectangle R , it is placed in a lower row. This ensures that no row of A contains more than one occurrence of a symbol after step b is executed. Hence the final step of completing every row to a permutation can be carried out successfully. \square

Theorem 3.2. For all positive integers n , there exists a type-1,3 RPA(n).

Proof. For even n , the desired result is an immediate corollary of Theorems 2.6 and 3.1. For odd n , we make use a certain modification of Theorem 2.6.

Let $A = (a_{i,j})$ be a type-1 RPA(n) on symbol set $\{1, \dots, n\}$, where n is odd, such that A contains a canonical $(n+1)/2 \times 2$ basic rectangle, say U , in its upper left corner (a type-1 RPA(n) constructed from the algorithm in Figure 1 will have this property). Denote $n_2 = (n+1)/2$ and let X denote the set of symbols in the first $n_2 - 1$ cells in the first column of A . Denote $a = a_{n_2,1}$ and let $Y = \{1, \dots, n\} \setminus (X \cup \{a\})$. Let $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ be any bijection in which $\pi(X) = Y$, $\pi(Y) = X$ and $\pi(a) = a$. Now define an $n \times n$ array $B = (b_{i,j})$ as follows:

$$b_{i,j} = \begin{cases} a_{i,j} & \text{if } i \leq n_2 \\ \pi(a_{n+1-i,j}) & \text{if } i > n_2. \end{cases}$$

We claim that B is a type-1, 3 $RPA(n)$. Most of the verifications are straightforward. The only tricky point is to ensure that the $n_2 \times 2$ basic rectangle in the lower left corner of B , say L , contains all n symbols. L consists of the last row of U , together with the image of the first $n_2 - 1$ rows of U under the bijection π . Because U is canonical, it follows that the first $n_2 - 1$ rows of U contain all the symbols in $\{1, \dots, n\} \setminus \{a\}$, and hence the last $n_2 - 1$ rows of L contain all the symbols in $\{1, \dots, n\} \setminus \{a\}$. The desired result follows. \square

Theorem 3.3. *For all positive integers n , there exists a type-1, 4 $RPA(n)$.*

Proof. For even n , the desired result is an immediate corollary of Theorems 2.8 and 3.1. For odd n , we make use a certain modification of Theorem 2.8. Actually, we will assume that $n \geq 9$, noting that type-1, 4 $RPA(3)$, type-1, 4 $RPA(5)$ and type-1, 4 $RPA(7)$ are constructed in Figures 9, 10 and 11, respectively.

Let $A = (a_{i,j})$ be a type-1 $RPA(n)$ on symbol set $\{1, \dots, n\}$, where n is odd, such that A contains a canonical $(n+1)/2 \times 2$ basic rectangle, say U , in its upper left corner. Denote $n_2 = (n+1)/2$, $X = \{a_{2,1}, \dots, a_{n_2-1,1}\}$, $b = a_{1,1}$, $a = a_{n_2,1}$ and $Y = \{1, \dots, n\} \setminus (X \cup \{a, b\})$.

We require some additional properties to hold in A :

1. The first $n_2 - 1$ cells in the last column of A must contain n_2 distinct symbols in the set $\{1, \dots, n\} \setminus \{a, b\}$.
2. $a_{n_2,n} = b$.

An arbitrary type-1 $RPA(n)$ might not satisfy these properties; however, we can easily transform any type-1 $RPA(n)$ into a type-1 $RPA(n)$ in which these properties hold. This is done as follows:

1. If $a_{n_2,n} \neq b$, then swap $a_{n_2,n}$ with the cell in row n_2 that contains b (note that $a_{n_2,1} = a \neq b$).
2. If $a_{1,n} = a$, then swap $a_{1,n}$ with some $a_{1,j} \neq a, b$, where $j > n_2$ (in order to be sure that this is possible, we require $(n-1)/2 \geq 2$, noting that $a_{1,1} = b$).
3. If $a_{2,n} \in \{a_{1,n}, a, b\}$, then swap $a_{2,n}$ with some $a_{2,j} \neq a, b, a_{1,n}$, where $j > n_2$ (in order to be sure that this is possible, we require $(n-1)/2 \geq 4$).
4. For each row i , $3 \leq i \leq n_2 - 1$, if $a_{i,n} \in \{a, b, a_{1,n}, \dots, a_{i-1,n}\}$, then we find a symbol $a_{i,j}$ (for some j , $\lceil \frac{n}{3} \rceil < j \leq n-1$) such that $a_{i,j} \notin \{a, b, a_{1,n}, \dots, a_{i-1,n}\}$ (such an $a_{i,j}$ exists by the pigeon-hole principle). Then swap $a_{i,n}$ and $a_{i,j}$.

Observe that the transformed square is still a type-1 $RPA(n)$. Define $Z = \{a_{1,n}, \dots, a_{n_2-1,n}\}$ and define $W = \{1, \dots, n\} \setminus (Z \cup \{a, b\})$. Then $|Y| = |Z| = n_2 - 1$ and $|X| = |W| = n_2 - 2$.

Now, let $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ be any bijection in which $\pi(X) = W$, $\pi(Z) = Y$, $\pi(a) = b$ and $\pi(b) = a$. Then define an $n \times n$ array $B = (b_{i,j})$ as follows:

$$b_{i,j} = \begin{cases} a_{i,j} & \text{if } i \leq n_2 \\ \pi(a_{n+1-i, n+1-j}) & \text{if } i > n_2. \end{cases}$$

We claim that B is a type-1, 4 $RPA(n)$. Most of the verifications are straightforward. The only tricky points are to ensure that the $n \times 1$ and $n_2 \times 2$ basic rectangles in the lower right corner of B contains all n symbols. The $n \times 1$ basic rectangle in the lower right corner of B is just the last

column of B , which contains the symbols in $Z \cup \{b\} \cup \pi(X) \cup \{\pi(b)\} = \{1, \dots, n\}$. Let U be the $n_2 \times 2$ basic rectangle in the upper left corner of A . The $n_2 \times 2$ basic rectangle in the lower right corner of B , say L , contains the symbol b together with the image of the first $n_2 - 1$ rows of U under the bijection π . Because U is canonical, it follows that the first $n_2 - 1$ rows of U contain all the symbols in $\{1, \dots, n\} \setminus \{a\}$, and hence the last $n_2 - 1$ rows of L contain all the symbols in $\{1, \dots, n\} \setminus \{b\}$. The desired result follows. \square

4 A Construction for type-1, 2 $RPA(n)$

4.1 Some Properties of Basic Rectangles

As before, we denote the number of basic rectangles by b . For $1 \leq k \leq b$, the k th basic rectangle has dimensions $r_k \times c_k$, where $c_k = \left\lceil \frac{n}{r_k} \right\rceil$ and we assume that $r_1 < r_2 < \dots < r_b$. The proof of the following lemma is straightforward.

Lemma 4.1. *Let $\sigma = \lceil \sqrt{n} \rceil$.*

1. *Suppose $n \leq \sigma(\sigma - 1)$. Then $b = 2\sigma - 2$. For $1 \leq k \leq \sigma - 1$, we have $r_k = k$ and $c_k = \left\lceil \frac{n}{k} \right\rceil$. For $\sigma \leq k \leq 2\sigma - 2$, we have $c_k = r_{2\sigma-1-k}$ and $r_k = c_{2\sigma-1-k}$.*
2. *Suppose $n > \sigma(\sigma - 1)$. Then $b = 2\sigma - 1$. For $1 \leq k \leq \sigma$, we have $r_k = k$ and $c_k = \left\lceil \frac{n}{k} \right\rceil$. For $\sigma + 1 \leq k \leq 2\sigma - 1$, we have $c_k = r_{2\sigma-k}$ and $r_k = c_{2\sigma-k}$.*

Remark: The two cases are distinguished by the fact that in case 2, one of the basic rectangles is a $\lceil \sqrt{n} \rceil \times \lceil \sqrt{n} \rceil$ square.

Lemma 4.2. *For $2 \leq k \leq b - 2$, we have that $c_k < 2c_{k+1}$.*

Proof. We will prove that $c_{k+1} > c_k/2$ except when $k \in \{1, b-1\}$ and n is even, or when $k = b-1$ and n is odd (in these cases, $c_{k+1} = c_k/2$). The only cases that are not obvious are when $2 \leq k \leq \sigma - 1$. In these cases, what we want to prove is that

$$\left\lceil \frac{n}{k+1} \right\rceil > \frac{1}{2} \left\lceil \frac{n}{k} \right\rceil.$$

We will prove the stronger result that

$$\frac{n}{k+1} > \frac{1}{2} \left\lceil \frac{n}{k} \right\rceil.$$

Let $n = qk - s$, where $0 \leq s \leq k - 1$. Then

$$\left\lceil \frac{n}{k} \right\rceil = q.$$

So the inequality to be proven is

$$2n > q(k+1).$$

We will prove this by contradiction. Suppose that $2n \leq q(k+1)$. Since $n = qk - s$, we have that

$$2(qk - s) \leq q(k+1),$$

or

$$q(k-1) \leq 2s.$$

Since $s \leq k-1$, we get

$$q(k-1) \leq 2(k-1),$$

and hence

$$(q-2)(k-1) \leq 0.$$

This can occur only if $k \in \{0, 1\}$ or $q \in \{1, 2\}$. The first case does not occur because $k \geq 2$. In the second case, we have

$$n \leq 2k - s \leq 2k \leq 2(\sigma - 1),$$

which is a contradiction for $n \geq 2$. So the second case does not occur, either. \square

4.2 The Basic Idea

For the time being, we assume that n is even. A promising way to attempt to construct a type-1, 2 $RPA(n)$ is to impose some additional structure on a type-1 $RPA(n)$ so that it is in fact a type-1, 2 $RPA(n)$. We consider arrays $A = (a_{i,j})$ where, for all $1 \leq i, j \leq n$, it holds that

$$a_{i,j} + a_{i,n+1-j} = n + 1. \tag{1}$$

Suppose we run the algorithm described in Figure 1, ensuring that, at the end of step b , no row contains two symbols that sum to $n+1$ (except for the first row, which is already a permutation of the n symbols). If this can be done, then we can easily fill in the rest of A to construct a type-1, 2 $RPA(n)$. This is done in two stages. First, for every filled cell (i, j) , we define $a_{i,n+1-j} = n+1 - a_{i,j}$. At this point, no row contains any symbol more than once, so it is then a simple matter to complete each row to a permutation of the n symbols.

We illustrate by constructing a type-1, 2 $RPA(8)$.

Example 4.1. *Suppose $n = 8$. The basic rectangles have dimensions 1×8 , 2×4 , 3×3 , 4×2 , and 8×1 .*

step 1

We begin by filling in the 1×8 basic rectangle (i.e., the first row of A):

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

step 2

Next, we consider the 2×4 basic rectangle. We place the symbols 5, 6, 7, 8 in the first four cells of the second row of this rectangle, noting that no two of these symbols sum to 9:

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

step 3

Now we turn to the 3×3 basic rectangle, filling in the first two cells of the third row with the symbols 4 and 8 (note that $4 + 8 \neq 9$):

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

step 4

Next, we look at the 4×2 basic rectangle. We have to fill in the symbols 3 and 7 (note that $3 + 7 \neq 9$):

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

step 5

The last basic rectangle has dimensions 8×1 . It is completed by filling in the symbols 2, 6, 8 and 7 into the first cells in the last four rows:

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

step 6

Now, we “reflect” each row to ensure that (1) holds:

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

step 7

Finally, we fill in all remaining cells in such a way that each row is a permutation. (We can

continue to require that (1) holds, if desired, but it is not necessary in this final step.)

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

In general, the above strategy will not always work, since we may be forced to fill in two elements in a row of a basic rectangle that sum to $n + 1$. So we need to do something a bit more clever.

4.3 The Construction Strategy

We are going to construct a sequence of canonical basic rectangles R_k , $2 \leq k \leq b - 2$, which satisfy the following additional properties:

For any two symbols a, b in the same row of R_k , it holds that $a + b \neq n + 1$. (2)

For any two symbols a, b in the last $c_k - c_{k+1}$ columns of R_k , it holds that $a + b \neq n + 1$. (3)

A canonical rectangle that satisfies (2) and (3) will be termed a *sum-free canonical basic rectangle*.

The property (3) will help to ensure that, when we copy symbols from the basic rectangle R_k ($k \notin \{1, k - 1\}$) to the next basic rectangle, R_{k+1} , we will not have a pair of symbols in a row that sum to $n + 1$ (so R_{k+1} will therefore satisfy property (2)).

In order to achieve property (3), we will carry out a sequence of operations of the following type: we define a *simple exchange operation* (*SE*) on a canonical basic rectangle to consist of a swap of two non-empty cells in the same row of the rectangle. It is obvious that the following lemma holds.

Lemma 4.3. *A canonical basic rectangle is still a canonical basic rectangle after an SE operation.*

Assume $2 \leq k \leq b - 2$. Suppose a canonical basic rectangle R_k is divided into two parts left_k and right_k , where left_k consists of the first c_{k+1} columns of R_k and right_k consists of the last $c_k - c_{k+1}$ columns of R_k . Note that left_k has more columns than right_k , by Lemma 4.2.

We form a graph G_k whose vertices are the n non-empty cells in R_k , where two cells are joined by an edge if the symbols in the two cells sum to $n + 1$. Observe that G_k consists of $n/2$ disjoint edges, and no two cells in a row of R_k form an edge in G_k .

Partition all the edges of G_k into three classes:

type A edges Both endpoints are in right_k .

type B edges One endpoint is in left_k and the other endpoint is in right_k .

type C edges Both endpoints are in left_k .

Lemma 4.4. *Suppose $k \notin \{1, b - 1\}$, R_k is a canonical basic rectangle, and left_k , right_k and G_k are as defined above. Let I denote any i rows of R_k , where $1 \leq i \leq r_k$. Then at least one of the following conditions holds:*

1. There is at least one edge of type C in G_k , where one or both endpoints of the edge are in rows in I .
2. There is an edge of type B in G_k where one endpoint is in $I \cap \text{left}_k$ and the other endpoint is in $I^c \cap \text{right}_k$ (where I^c denotes the set of rows of R_k disjoint from I).

Remark: We observe that case 2 cannot occur if $i = r_k$, because $I^c = \emptyset$.

Proof. Suppose case 1 does not occur. Then all the edges with an endpoint in $I \cap \text{left}_k$ are of type B . However, the number of cells in $I \cap \text{left}_k$ is greater than the number of cells in $I \cap \text{right}_k$. Therefore one of these edges has an endpoint in a row of right_k that is not in I . \square

The main step in our construction is given in the following lemma.

Lemma 4.5. *Suppose $k \notin \{1, b-1\}$ and suppose we have a canonical basic rectangle R_k , such that any elements a, b within a row satisfy $a + b \neq n + 1$. Then there exists a sum-free canonical basic rectangle R'_k obtained from R_k by a sequence of SE operations.*

Proof. Define left_k , right_k and G_k as above. Let a_k denote the number of edges of type A in G_k . We are going to prove that if $a_k > 0$, then we can use SE operations to reduce a_k by at least one. Therefore a sequence of SE operations will reduce a_k to 0.

Suppose $a_k > 0$ and let x_1y_1 be an edge of type A . Suppose x_1 is in row i_1 and y_1 is in row i_2 . Define $I = I_1 = \{i_1, i_2\}$ and apply Lemma 4.4. There are two possible cases, at least one of which must arise.

case 1 There is an edge x_2y_2 of type C where x_2 is in row i_1 or i_2 .

case 2 There is an edge x_2y_2 of type B with $x_2 \in I_1 \cap \text{left}_k$ and $y_2 \in I_1^c \cap \text{right}_k$.

If case 1 occurs, then we stop. Otherwise, in case 2, we let i_3 be the row containing y_2 , set $I = I_2 = \{i_1, i_2, i_3\}$ and apply Lemma 4.4 again. Again there are two cases: either we get an edge of type C with at least one endpoint in I_2 , or we get an edge x_3y_3 of type B with $x_3 \in I_2 \cap \text{left}_k$ and $y_3 \in I_2^c \cap \text{right}_k$. In the first case, we stop, while in the second case, we let i_4 be the row containing y_3 , set $I = I_3 = \{i_1, i_2, i_3, i_4\}$ and apply Lemma 4.4 again. We continue this process until we first reach a situation where we find a type C edge x_jy_j , which must happen eventually (see the Remark following Lemma 4.4).

Thus we get a sequence of edges $x_1y_1, x_2y_2, \dots, x_jy_j$ which satisfy the following properties:

- x_1 is in row i_1 of right_k and y_1 is in row i_2 of right_k .
- x_2 is in row i_1 or i_2 of left_k and y_2 is in row i_3 of right_k , where $i_3 \notin \{i_1, i_2\}$.
- ...
- x_{j-1} is in one of rows i_1, i_2, \dots, i_{j-1} of left_k and y_{j-1} is in row i_j of right_k , where $i_j \notin \{i_1, i_2, \dots, i_{j-1}\}$.
- x_j is in one of rows i_1, i_2, \dots, i_j of left_k and y_j is in left_k .

Now we proceed “backwards”, identifying a certain subset of the edges $\mathcal{E} = \{x_1y_1, x_2y_2, \dots, x_jy_j\}$. Denote $\mathcal{X} = \{x_1, \dots, x_j\}$ and $\mathcal{Y} = \{y_1, \dots, y_j\}$.

- Start by selecting the edge $x_j y_j$. Then define $e_1 = x_j$ and $f_1 = y_j$.
- Select the edge $xy \in \mathcal{E}$ where $x \in \mathcal{X}$, $y \in \mathcal{Y}$ and y is in the same row as e_1 . Then define $e_2 = x$ and $f_2 = y$.
- Select the edge $xy \in \mathcal{E}$ where $x \in \mathcal{X}$, $y \in \mathcal{Y}$ and y is in the same row as e_2 . Then define $e_3 = x$ and $f_3 = y$.
- ...
- Eventually, we will reach a situation where $e_{\ell-1} \in \mathcal{X}$, and either x_1 or y_1 is in the same row as $e_{\ell-1}$. If y_1 is in the same row as $e_{\ell-1}$, then define $e_\ell = x_1$ and $f_\ell = y_1$; otherwise (if x_1 is in the same row as $e_{\ell-1}$), define $e_\ell = y_1$ and $f_\ell = x_1$.

Observe that $e_1 f_1$ is of type C , $e_\ell f_\ell$ is of type A , and the remaining $\ell - 2$ selected edges are of type B .

Now we swap e_1 and f_2 ; we swap e_2 and f_3 ; and we continue swapping pairs of vertices, ending by swapping $e_{\ell-1}$ and f_ℓ . (The only two vertices in the selected edges that are *not* swapped with some other vertex are f_1 and e_ℓ .) The result is that the selected edges of types A and C become edges of type B , and the other edges remain as type B edges. \square

We need another preliminary lemma before we present the algorithm to construct a type-1,2 $RPA(n)$.

Lemma 4.6. *Suppose $w > b_L \geq \dots \geq b_1$ are positive integers and suppose d is a positive integer. Denote $b = \sum_{i=1}^L b_i$ and suppose that*

$$0 \leq t \leq (L + d)w - b. \quad (4)$$

Suppose B_1, \dots, B_L are pairwise disjoint sets such that $|B_i| = b_i$ for $1 \leq i \leq L$. Finally, suppose that $|B| = t$. Then there exists a partition

$$B = \left(\bigcup_{i=1}^L C_i \right) \cup \left(\bigcup_{i=1}^d D_i \right),$$

where the following properties are satisfied:

1. $w \geq b_L + |C_L| \geq b_{L-1} + |C_{L-1}| \geq \dots \geq b_1 + |C_1| \geq |D_1| \geq \dots \geq |D_d|$.
2. $C_i \cap B_i = \emptyset$ for $1 \leq i \leq L$.

Remark: The inequality (4) is an obvious necessary condition for the desired partition to exist, so this lemma in fact gives necessary and sufficient conditions for the stated conclusion to be true.

Proof. We will denote an instance of this problem by a tuple of the form $(L, w, d, b; B_1, \dots, B_L; t)$. We prove the desired result by induction on L .

If $L = 0$, then (4) guarantees that $0 \leq t \leq dw$, and the result is obvious. Suppose $L > 0$. The strategy will be to choose C_L so $|C_L|$ is as large as possible, and then apply induction.

We distinguish three cases:

case 1 $t \geq w$.

Here we have $|B \setminus B_L| \geq t - b_L \geq w - b_L$. Choose $C_L \subseteq B \setminus B_L$ with $|C_L| = w - b_L$, and note that $b_L + |C_L| = w$. We now define a smaller instance $I' = (L', w', d', b'; B'_1, \dots, B'_{L'}; t')$ of the same problem where $L' = L - 1$, $w' = w$, $d' = d$, $B'_1 = B_1, \dots, B'_{L'} = B_{L-1}$, $b' = b - b_L$, $B' = B \setminus C_L$ and $t' = |B'| = t - w + b_L$.

We need to verify that $0 \leq t' \leq (L' + d')w' - b'$, which is equivalent to $0 \leq t - w + b_L \leq (L - 1 + d)w - (b - b_L)$, or $w - b_L \leq t \leq (L + d)w - b$, which follows immediately from (4), since $t \geq w$. Therefore, by induction, we can solve the instance I' . The solution to I' , along with C_L , is a solution to I .

case 2 $t \leq b_L$.

Here we define $C_L = \emptyset$. We now define the instance $I' = (L', w', d', b'; B'_1, \dots, B'_{L'}; t')$ where $L' = L - 1$, $w' = b_L$, $d' = d$, $B'_1 = B_1, \dots, B'_{L'} = B_{L-1}$, $b' = b - b_L$, $B' = B$ and $t' = t$.

We need to verify that $0 \leq t' \leq (L' + d')w' - b'$, which is equivalent to $0 \leq t \leq (L - 1 + d)b_L - (b - b_L)$, or $0 \leq t \leq (L + d)b_L - b$. We have that

$$\begin{aligned} (L + d)b_L - b &= db_L + \sum_{i=1}^L (b_L - b_i) \\ &\geq b_L + 0 \\ &\geq t, \end{aligned}$$

as desired. Therefore, by induction, we can solve the instance I' . The solution to I' , along with C_L , is a solution to I .

case 3 $b_L < t < w$.

Here we have $|B \setminus B_L| \geq t - b_L$. Choose $C_L \subseteq B \setminus B_L$ with $|C_L| = t - b_L$, and note that $b_L + |C_L| = t$. We now define the instance $I' = (L', w', d', b'; B'_1, \dots, B'_{L'}; t')$ where $L' = L - 1$, $w' = t$, $d' = d$, $B'_1 = B_1, \dots, B'_{L'} = B_{L-1}$, $b' = b - b_L$, $B' = B \setminus C_L$ and $t' = b_L$.

We need to verify that $0 \leq t' \leq (L' + d')w' - b'$, which is equivalent to $0 \leq b_L \leq (L - 1 + d)t - (b - b_L)$, or $0 \leq (L - 1 + d)t - b$. We have that

$$\begin{aligned} (L - 1 + d)t - b &= (d - 1)t + \sum_{i=1}^L (t - b_i) \\ &\geq (d - 1)t + 0 \\ &\geq 0, \end{aligned}$$

as desired. Therefore, by induction, we can solve the instance I' . The solution to I' , along with C_L , is a solution to I . □

We now describe how to make use of Lemma 4.6 when we are copying symbols from one basic rectangle to the next one. Let the k th basic rectangle be denoted by R , where $k \geq 2$, and let the previous basic rectangle be denoted by R' . Let S denote the set of symbols in $R' \setminus R$. We want to copy the symbols in S into the empty cells R so that R is canonical and no pair of symbols in a

row of R sum to $n + 1$. If two symbols x and y in a row of R sum to $n + 1$, then exactly one of x, y is in S . So we just need to ensure that we never place a symbol $y \in S$ in a row of R that already contains the symbol $x = n + 1 - y$.

Consider the state of R just before the symbols in S are copied into R . Suppose there are L non-empty non-filled rows of R , which contain b_L, \dots, b_1 elements, where $b_L \geq \dots \geq b_1$. Name these rows r_L, \dots, r_1 , respectively, and note that r_L is the “first” of these L rows. Further, suppose there are d empty rows in R , which will immediately follow row r_1 ; note that $d \geq 1$. For $1 \leq i \leq L$, define

$$B_i = \{n + 1 - y : y \text{ is in a cell in row } r_i \text{ of } R\}.$$

It is clear that the B_i 's are pairwise disjoint. Let w denote the number of columns in R , let $B = S$, and apply Lemma 4.6. The resulting partition of B determines how the elements in S can be copied into R so that the desired properties are satisfied.

The algorithm presented in Figure 2 constructs a type-1,2 $RPA(n)$ for even n by suitably modifying the algorithm presented in Figure 1. Here is a small example to illustrate Algorithm 2.

Example 4.2. Suppose $n = 10$. The basic rectangles have dimensions 1×10 , 2×5 , 3×4 , 4×3 , 5×2 and 10×1 .

step 1

We begin by filling in the 1×10 basic rectangle:

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

step 2

Next, we complete the 2×5 basic rectangle:

1	2	3	4	5	6	7	8	9	10
6	7	8	9	10					

step 3

Now we complete the 3×4 basic rectangle:

1	2	3	4	5	6	7	8	9	10
6	7	8	9	10					
5	10								

step 4

The 4×3 basic rectangle is now filled in:

1	2	3	4	5	6	7	8	9	10
6	7	8	9	10					
5	10	4							
9									

The next basic rectangle has dimensions 5×2 . At this point, right_4 consists of a single column containing the three symbols 3, 8, and 4. Since $3 + 8 = 11$, we need to perform one or more

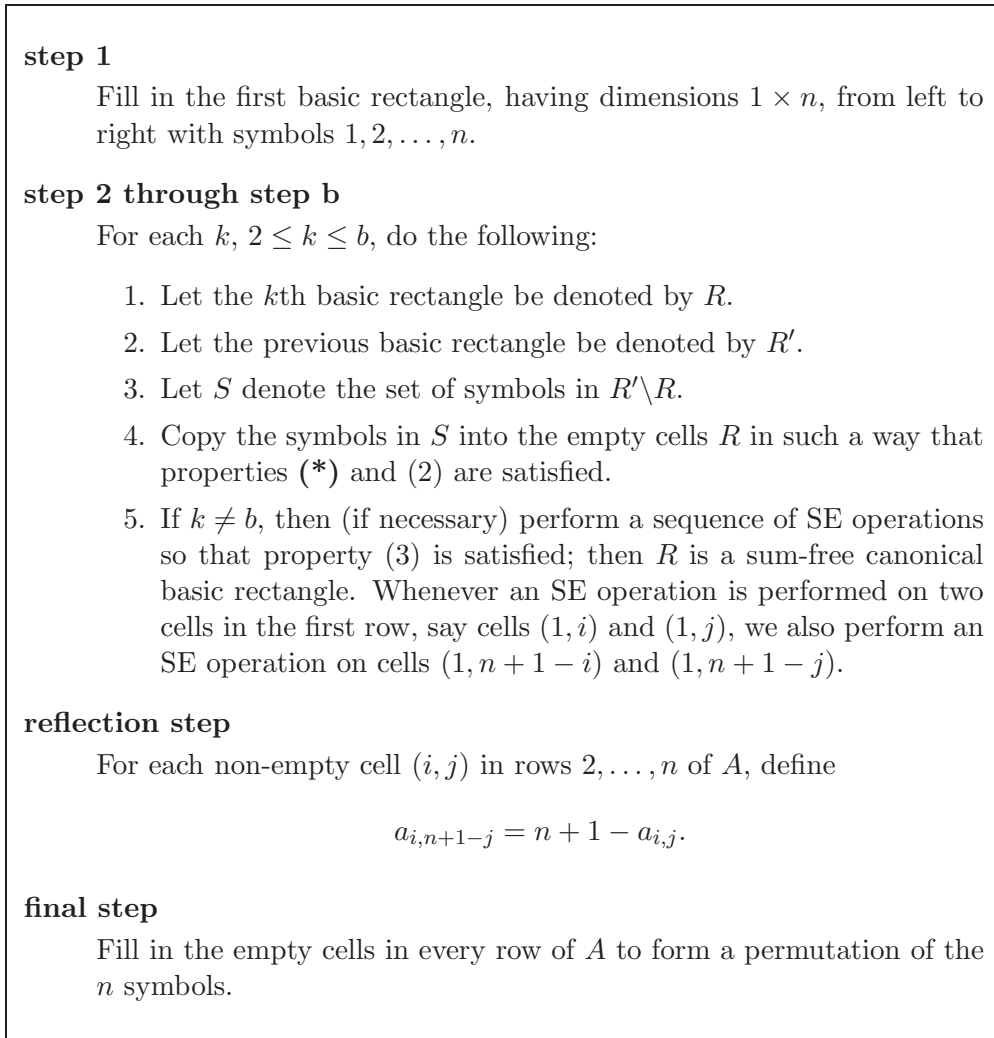


Figure 2: Algorithm to construct a type-1,2 $RPA(n)$ for even n .

SE operations. The edge $\{3, 8\}$ is of type A and we see that $\{1, 10\}$ is an edge of type C. Here we just swap 1 and 3. Since these cells are in the first row, we also swap 8 and 10, obtaining the following:

3	2	1	4	5	6	7	10	9	8
6	7	8	9	10					
5	10	4							
9									

step 5

Now we can fill in the 5×2 basic rectangle:

3	2	1	4	5	6	7	10	9	8
6	7	8	9	10					
5	10	4							
9	1								
8	4								

The remaining steps are straightforward.

We summarise the results of this section in the following theorem.

Theorem 4.7. *The algorithm presented in Figure 2 produces a type-1,2 RPA(n) for any even positive integer n .*

Proof. The first basic rectangle, R_1 , is initially filled in with the integers $1, \dots, n$ from left to right. The second basic rectangle initially contains $1, \dots, n/2$ in its first row and $n/2 + 1, \dots, n$ in its second row.

For $k = 2, 3, \dots, b - 3$, when proceeding from R_k to R_{k+1} , Lemma 4.6 guarantees that property (2) can be satisfied in R_{k+1} . Then Lemma 4.5 shows, from a succession of SE operations, that R_{k+1} can be made to also satisfy property (3). Hence R_1, R_2, \dots, R_{b-2} are sum-free canonical basic rectangles.

Now R_{b-1} satisfies (2) but it might not satisfy property (3). However, this does not matter because R_b has only one column and hence (2) will automatically be satisfied in R_b . When we carry out the “reflection” step, every basic rectangle in the upper right corner is the image of a basic rectangle in the upper left corner under the mapping $i \mapsto n + 1 - i$, and therefore it contains all n symbols. Also, after the reflection step, no symbol occurs twice in a row, so the final step is straightforward. \square

The following result is an immediate corollary of Theorems 2.7 and 4.7.

Corollary 4.8. *For all even integers $n \geq 2$, there exists a type-1,2,3,4 RPA(n).*

4.4 A Modified Construction for Odd n

When n is odd, we need to alter the algorithm presented in Figure 2 slightly. The modified algorithm is presented in Figure 3. The modified algorithm is very similar to the previous algorithm. The only changes are the swap of two symbols in step 1, and the reflection step.

Example 4.3. *Suppose $n = 9$. The basic rectangles have dimensions 1×9 , 2×5 , 3×3 , 5×2 and 9×1 .*

step 1

We begin by filling in the 1×9 basic rectangle and swapping symbols 1 and 5:

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

step 1

Denote $n_2 = (n + 1)/2$. Fill in the first basic rectangle, having dimensions $1 \times n$, from left to right with symbols $1, 2, \dots, n$. Then interchange the symbols 1 and n_2 .

step 2 through step b

For each k , $2 \leq k \leq b$, do the following:

1. Let the k th basic rectangle be denoted by R .
2. Let the previous basic rectangle be denoted by R' .
3. Let S denote the set of symbols in $R' \setminus R$.
4. Copy the symbols in S into the empty cells R in such a way that properties (*) and (2) are satisfied.
5. If $k \neq b$, then (if necessary) perform a sequence of SE operations so that property (3) is satisfied; then R is a sum-free canonical basic rectangle. Whenever an SE operation is performed on two cells in the first row, say cells $(1, i)$ and $(1, j)$, we also perform an SE operation on cells $(1, n + 1 - i)$ and $(1, n + 1 - j)$.

reflection step

Define $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ as follows:

$$\pi(i) = \begin{cases} n + 1 - i & \text{if } i \notin \{1, n_2\} \\ n_2 & \text{if } i = 1 \\ n & \text{if } i = n_2. \end{cases}$$

For each non-empty cell (i, j) in rows $2, \dots, n$ of A , define

$$a_{i, n+1-j} = \pi(a_{i, j}).$$

final step

Fill in the empty cells in every row of A to form a permutation of the n symbols.

Figure 3: Algorithm to construct a type-1, 2 $RPA(n)$ for odd n .

step 2

Next, we complete the 2×5 basic rectangle:

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

The next basic rectangle has dimensions 3×3 . At this point, right_2 consists two columns containing the three symbols 1, 4, and 9. Since $1 + 9 = 10$, we need to perform one or more SE operations. The edge $\{1, 9\}$ is of type A and $\{3, 7\}$ is an edge of type C. Here we just swap 7 and 9, obtaining the following:

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

step 3

Now we complete the 3×3 basic rectangle:

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

step 4

The 5×2 basic rectangle is now filled in:

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

step 5

Now we can fill in the 9×1 basic rectangle:

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

reflection step

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

Theorem 4.9. *The algorithm presented in Figure 3 produces a type-1,2 RPA(n) for any odd positive integer n .*

Proof. The first basic rectangle, R_1 , contains integers $1, \dots, n$. The second basic rectangle contains $1, \dots, n_2$ in its first row and $n_2 + 1, \dots, n$ in its second row. The $2 \times n_2$ rectangle in the upper right corner contains 1 and $n_2 + 1, \dots, n$ in its first row and $1, \dots, n_2$ in its second row (note that the middle cell in the second row contains n_2 after the final step). Each basic rectangle R_3, \dots contains all n symbols, as in the proof of the previous algorithm. Each corresponding basic rectangle in the upper right corner is the image of a basic rectangle under the bijection π , so it also contains all n symbols. Finally, after the reflection step, no symbol occurs twice in a row (here we use the easily verified fact that the symbol n_2 only occurs in the first cell of the first row, and it is never moved by any SE operation), so the final step is straightforward. \square

The next corollary is an immediate consequence of Theorems 4.7 and 4.9.

Corollary 4.10. *For all positive integers n , there exists a type-1,2 RPA(n).*

5 Latin RPAs

In this section, we present some results on latin RPAs, which we denote as LRPA's. It is obvious that Theorems 2.1, 2.2, 2.3, 2.4 and 2.5 hold for LRPA's. In contrast, the proofs of Theorems 2.6, 2.7 and 2.8 do not seem to be easily generalisable to LRPA's. However, we have a result that applies to certain latin RPAs that does not necessarily hold for non-latin RPAs.

Theorem 5.1. *Suppose that A is a type-1,2 LRPA(n) and B is obtained from A by reflecting A in its main diagonal. Then B is a type-1,3 LRPA(n). Conversely, if B is a type-1,3 LRPA(n) and A is obtained from B by reflecting B in its main diagonal, then A is a type-1,2 LRPA(n).*

The next four corollaries give the weakest types we know of that will produce an LRPA of each specified type.

Corollary 5.2. *If there exists a type-1 LRPA(n), then the following arrays exist: a type-2 LRPA(n), a type-3 LRPA(n) and a type-4 LRPA(n).*

Corollary 5.3. *If there exists a type-1,2 LRPA(n), then the following arrays exist: a type-3,4 LRPA(n), a type-1,3 LRPA(n) and a type-2,4 LRPA(n).*

Figure 4: A type-1, 2, 3, 4 $LRPA(6)$

1	2	3	4	5	6
4	5	6	1	2	3
3	6	5	2	1	4
2	1	4	3	6	5
5	4	1	6	3	2
6	3	2	5	4	1

Figure 5: A type-1, 2, 3, 4 $LRPA(8)$

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

Corollary 5.4. *If there exists a type-1, 4 $LRPA(n)$, then a type-2, 3 $LRPA(n)$ exists.*

Corollary 5.5. *If there exists a type-1, 2, 3 $LRPA(n)$, then the following arrays exist: a type-1, 2, 4 $LRPA(n)$, a type-1, 3, 4 $LRPA(n)$ and a type-2, 3, 4 $LRPA(n)$.*

5.1 Some small $LRPA$ s

We begin by observing that any latin square of order 2 or 3 is automatically a type-1, 2, 3, 4 $LRPA(2)$ or type-1, 2, 3, 4 $LRPA(3)$, respectively. Furthermore, any type-1 $LRPA(4)$ is a type-1, 2, 3, 4 $LRPA(4)$.

Finding general constructions for $LRPA$ s seems to be quite difficult. However, several small examples of $LRPA$ s for even n are presented in Figures 4–8 (Figures 6–8 can be found in the Appendix). The $LRPA$ s in Figures 4–8 all satisfy the symmetry condition (1), i.e., that $a_{i,j} + a_{i,n+1-j} = n+1$. In addition, the $LRPA$ s in Figures 5–8 have a symmetry about the main diagonal, which we describe now. For the type-1, 2, 3, 4 $LRPA(8)$, define the permutation π as follows (π is given in disjoint cycle representation):

$$\pi = (1)(2\ 5)(3\ 8)(4\ 7)(6).$$

Then it can be verified that the following condition holds for all i, j :

$$a_{i,j} = \pi(a_{j,i}). \tag{5}$$

The $LRPA$ s of orders 10, 12 and 14 satisfy similar symmetry properties.

We also have constructed type-1, 2, 3, 4 $LRPA(n)$ for $n \in \{3, 5, 7, 9\}$; see Figures 9–12 in the Appendix. It is interesting to note that any type-1, 2, 3, 4 $LRPA(9)$ is in fact a Sudoku square.

5.2 A Construction for type-1, 2, 3, 4 LRPA(n) for $n = 16$ and 36

One possible approach to constructing LRPA's is to make use of the following lemma:

Lemma 5.6. *Let $n \geq 2$ be even, and suppose there exists an $\frac{n}{2} \times \frac{n}{2}$ latin square S with the property that for all i with $2 \leq i \leq \frac{n}{2}$, the $i \times \lceil \frac{n}{i} \rceil$ rectangle in the upper left hand corner of S contains each of the symbols from 1 to $\frac{n}{2}$ at least twice. Then there exists a type-1, 2, 3, 4 LRPA(n).*

Proof. We construct a type-1, 2, 3, 4 LRPA(n), A , from S as follows:

step 1

Each of the $i \times \lceil \frac{n}{i} \rceil$ rectangles in the upper left hand corner of S contains each symbol x with $1 \leq x \leq \frac{n}{2}$ twice. By considering each such rectangle in turn and replacing appropriately chosen copies of x by $n + 1 - x$ we show that we can construct a new array S' for which each of the $i \times \lceil \frac{n}{i} \rceil$ rectangles in the upper left hand corner contain each of the symbols from 1 to n . This is done as follows.

For each symbol x , we need to decide which occurrences of x to replace by $n + 1 - x$. We do this by constructing a certain bipartite graph whose vertices are the cells containing x , and then 2-colouring the vertices of the graph.

Consider the cells containing x in the union of all the $i \times \lceil \frac{n}{i} \rceil$ rectangles. Suppose they are cells (r_i, c_i) , $1 \leq i \leq L$, where $r_1 < r_2 < \dots < r_L$. We construct a directed graph D on vertex set $\{1, \dots, L\}$ as follows. For $2 \leq i \leq L$, we have an arc from vertex i to j , where $c_j = \min\{c_1, \dots, c_{i-1}\}$ (i.e., r_j is the row of the leftmost vertex "above" vertex i).

Every vertex $2, \dots, L$ in D has outdegree 1, and every arc is directed from a higher-numbered vertex to a lower-numbered vertex. Therefore the underlying undirected graph G is a tree. Hence G is bipartite and we can colour the vertices with two colours, say red and blue, such that there are no monochromatic edges. For every red vertex, replace x by $n + 1 - x$ in S , and call the resulting array S' .

Now we prove that this works. Consider an $i \times \lceil \frac{n}{i} \rceil$ rectangle R . Let s be the highest-numbered vertex of D in R . Suppose s is directed to vertex t . Since R contains at least two vertices of D by assumption, it follows that t is also in R . The vertices s and t have different colours, so one of the occurrences of x in R is relabelled to $n + 1 - x$.

Note that S' still has the property that any symbol appears at most once in each row and each column. Furthermore, the fact that S was a latin square of order $n/2$ implies that x and $n + 1 - x$ never appear in the same row or in the same column.

step 2

Now we let S' form the top left corner of A , and "reflect" it by applying the symmetry condition (1), to fill in the top right corner of A . Finally, we carry out a similar reflection vertically to fill in the rest of A . The result is an LRPA that is symmetric under rotation through 180 degrees.

□

Example 5.1. *We give an example of an 8×8 latin square S with the required properties. Note that the shaded cells are cells that are contained in basic rectangles in the resulting 16×16 latin*

square.

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

We now adjust the entries in the top left rectangles so that each rectangle contains all the numbers from 1 to 16:

1	2	3	4	10	9	11	12
15	5	6	7	13	16	14	8
14	11	12	8	16	2	4	7
13	10	9	16	2	3	5	6
7	4	16	2	5	6	8	3
8	16	2	3	6	5	7	4
6	3	4	5	8	7	1	2
12	9	7	6	3	4	2	1

Finally, we “reflect” the result to obtain a type-1,2,3,4 LRPA(16):

1	2	3	4	10	9	11	12	5	6	8	7	13	14	15	16
15	5	6	7	13	16	14	8	9	3	1	4	10	11	12	2
14	11	12	8	16	2	4	7	10	13	15	1	9	5	6	3
13	10	9	16	2	3	5	6	11	12	14	15	1	8	7	4
7	4	16	2	5	6	8	3	14	9	11	12	15	1	13	10
8	16	2	3	6	5	7	4	13	10	12	11	14	15	1	9
6	3	4	5	8	7	1	2	15	16	10	9	12	13	14	11
12	9	7	6	3	4	2	1	16	15	13	14	11	10	8	5
5	8	10	11	14	15	15	16	1	2	4	3	5	7	9	12
11	14	13	12	9	10	16	15	2	1	7	8	5	4	3	6
9	1	15	14	11	12	10	13	4	7	5	6	3	2	16	8
10	13	1	15	12	11	9	14	3	8	6	5	2	16	4	7
4	7	8	1	15	14	12	11	6	5	3	2	16	9	10	13
3	6	5	9	1	15	13	10	7	4	2	16	8	12	11	14
2	12	11	10	4	1	3	9	8	14	16	13	7	6	5	15
16	15	14	13	7	8	6	5	12	11	9	10	4	3	2	1

This approach was used to generate the type-1,2,3,4 LRPA(36) that is presented in Figure 13 in the Appendix.

6 Open Problems

We mention two open problems.

1. Does there exist a type-1,2,3,4 $RPA(n)$ for all odd positive integers n ? One potential way to prove this would be to generalise Theorem 2.7 to odd n .
2. At the present time, there is no known infinite class of type-1 $LRPA(n)$. However, we do not hesitate to conjecture that type-1,2,3,4 $LRPA(n)$ exist for all positive integers n .

Acknowledgements

Thanks to Michael Dinitz for bringing this problem to our attention.

References

- [1] C.J. Colbourn and K.E. Heinrich. Conflict-free access to parallel memories, *Journal of Parallel and Distributed Computing* **14** (1992), 193–200.
- [2] D.L. Erickson and C.J. Colbourn. Conflict-free access to rectangular subarrays, *Congressus Numerantium* **90** (1992), 239–253.
- [3] Li Erran Li, Junliang Liu, Kun Tan, Harish Viswanathan, and Yang Richard Yang. Retransmission \neq repeat: simple retransmission permutation can resolve overlapping channel collisions, *Eighth ACM Workshop on Hot Topics in Networks (HotNets-VIII)*, 2009.
- [4] R.A. Bailey, Peter Cameron and Robert Connelly. Sudoku, gerechte designs, resolutions, affine space, spreads, reguli, and Hamming codes, *American Mathematical Monthly*, Volume 115, Number 5, May 2008, pp 383–404.

A Some Examples of type-1, 2, 3, 4 $LRPA(n)$

Figure 6: A type-1, 2, 3, 4 $LRPA(10)$

1	2	3	4	5	6	7	8	9	10
6	7	8	9	10	1	2	3	4	5
3	4	10	5	9	2	6	1	7	8
8	9	5	1	4	7	10	6	2	3
5	10	9	8	7	4	3	2	1	6
2	1	6	7	8	3	4	5	10	9
7	6	2	10	3	8	1	9	5	4
4	3	1	2	6	5	9	10	8	7
9	8	7	6	1	10	5	4	3	2
10	5	4	3	2	9	8	7	6	1

Figure 7: A type-1, 2, 3, 4 $LRPA(12)$

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

Figure 8: A type-1, 2, 3, 4 $LRPA(14)$

1	2	3	4	5	6	7	8	9	10	11	12	13	14
8	9	10	11	12	13	14	1	2	3	4	5	6	7
7	13	14	5	6	3	11	4	12	9	10	1	2	8
4	6	12	14	2	5	8	7	10	13	1	3	9	11
12	5	11	8	1	9	2	13	6	14	7	4	10	3
11	10	7	12	9	1	13	2	14	6	3	8	5	4
3	14	6	2	8	10	4	11	5	7	13	9	1	12
2	1	4	3	10	8	6	9	7	5	12	11	14	13
9	8	5	13	11	14	12	3	1	4	2	10	7	6
13	7	9	10	14	11	3	12	4	1	5	6	8	2
6	4	13	1	3	7	10	5	8	12	14	2	11	9
5	12	1	7	4	2	9	6	13	11	8	14	3	10
10	11	8	9	13	12	1	14	3	2	6	7	4	5
14	3	2	6	7	4	5	10	11	8	9	13	12	1

Figure 9: A type-1, 2, 3, 4 $LRPA(3)$

1	2	3
2	3	1
3	1	2

Figure 10: A type-1, 2, 3, 4 $LRPA(5)$

1	2	3	4	5
4	5	1	2	3
3	4	2	5	1
2	3	5	1	4
5	1	4	3	2

Figure 11: A type-1, 2, 3, 4 $LRPA(7)$

1	2	3	4	5	6	7
5	6	7	1	2	3	4
4	3	2	7	6	1	5
7	5	1	6	3	4	2
2	4	6	5	1	7	3
3	1	4	2	7	5	6
6	7	5	3	4	2	1

Figure 12: A type-1, 2, 3, 4 $LRPA(9)$

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

Figure 13: A type-1, 2, 3, 4 $LRPA(36)$

1	2	3	4	5	6	26	22	27	28	24	23	29	12	20	19	21	30	7	16	18	17	25	8	14	13	9	10	15	11	31	32	33	34	35	36
35	7	8	9	10	11	31	36	18	17	25	16	34	13	14	15	32	33	4	5	22	23	24	3	21	12	20	19	1	6	26	27	28	29	30	2
34	29	12	13	14	15	20	19	21	30	32	33	1	2	6	9	10	11	26	27	28	31	35	36	4	5	7	16	18	17	22	23	24	25	8	3
33	28	24	25	16	17	23	30	32	1	2	3	6	8	10	11	15	18	19	22	26	27	29	31	34	35	36	5	7	14	20	21	12	13	9	4
32	27	23	21	30	18	36	2	3	4	6	8	9	11	12	13	17	15	22	20	24	25	26	28	29	31	33	34	35	1	19	7	16	14	10	5
31	26	22	20	19	36	2	3	4	5	7	9	10	14	8	12	13	16	21	24	25	29	23	27	28	30	32	33	34	35	1	18	17	15	11	6
11	6	17	14	36	2	3	4	7	8	9	5	12	10	15	16	18	13	24	19	21	22	27	25	32	28	29	30	33	34	35	1	23	20	31	26
15	36	18	30	2	3	4	5	6	10	8	11	13	9	16	14	12	17	20	25	23	21	28	24	26	29	27	31	32	33	34	35	7	19	1	22
10	19	16	5	3	4	7	6	1	2	11	12	14	15	13	17	8	9	28	29	20	24	22	23	25	26	35	36	31	30	33	34	32	21	18	27
9	20	30	1	4	5	8	10	2	3	14	13	15	16	18	6	11	12	25	26	31	19	21	22	24	23	34	35	27	29	32	33	36	7	17	28
13	25	5	2	6	7	9	8	11	14	15	17	16	18	1	3	4	10	27	33	34	36	19	21	20	22	23	26	29	28	30	31	35	32	12	24
14	21	4	3	8	9	5	11	12	13	1	15	18	17	2	10	7	6	31	30	27	35	20	19	22	36	24	25	26	32	28	29	34	33	16	23
8	3	1	6	9	10	12	13	14	15	16	18	17	4	11	7	2	5	32	35	30	26	33	20	19	21	22	23	24	25	27	28	31	36	34	29
12	24	2	8	11	14	10	16	15	18	17	1	4	6	7	5	9	3	34	28	32	30	31	33	36	20	19	22	21	27	23	26	29	35	13	25
17	23	6	10	12	8	15	9	13	16	18	2	11	7	5	4	3	1	36	34	33	32	30	26	35	19	21	24	28	22	29	25	27	31	14	20
18	22	9	11	13	12	16	14	17	6	3	10	7	5	4	2	1	8	29	36	35	33	32	30	27	34	31	20	23	21	25	24	26	28	15	19
16	5	10	15	17	13	18	12	8	11	4	7	2	3	9	1	6	14	23	31	36	28	34	35	30	33	26	29	25	19	24	20	22	27	32	21
30	4	11	18	15	16	13	17	9	12	10	6	5	1	3	8	14	2	35	23	29	34	36	32	31	27	25	28	20	24	21	22	19	26	33	7
7	33	26	19	22	21	24	20	28	25	27	31	32	36	34	29	23	35	2	14	8	3	1	5	6	10	12	9	17	13	16	15	18	11	4	30
21	32	27	22	20	24	19	25	29	26	33	30	35	34	28	36	31	23	14	6	1	9	3	2	7	4	11	8	12	18	13	17	15	10	5	16
19	15	28	26	24	25	21	23	20	31	34	27	30	32	33	35	36	29	8	1	2	4	5	7	10	3	6	17	14	16	12	13	11	9	22	18
20	14	31	27	25	29	22	28	24	21	19	35	26	30	32	33	34	36	1	3	4	5	7	11	2	18	16	13	9	15	8	12	10	6	23	17
25	13	35	29	26	23	27	21	22	19	20	36	33	31	30	32	28	34	3	9	5	7	6	4	1	17	18	15	16	10	14	11	8	2	24	12
29	34	36	31	28	27	25	24	23	22	21	19	20	33	26	30	35	32	5	2	7	11	4	17	18	16	15	14	13	12	10	9	6	1	3	8
23	16	33	34	29	28	32	26	25	24	36	22	19	20	35	27	30	31	6	7	10	2	17	18	15	1	13	12	11	5	9	8	3	4	21	14
24	12	32	35	31	30	28	29	26	23	22	20	21	19	36	34	33	27	10	4	3	1	18	16	17	15	14	11	8	9	7	6	2	5	25	13
28	17	7	36	33	32	29	27	35	34	23	24	22	21	19	31	26	25	12	11	6	18	16	15	13	14	3	2	10	8	5	4	1	30	20	9
27	18	21	32	34	33	30	31	36	35	26	25	23	22	24	20	29	28	9	8	17	13	15	14	12	11	2	1	6	7	4	3	5	16	19	10
22	1	19	7	35	34	33	32	31	27	29	26	24	28	21	23	25	20	17	12	14	16	9	13	11	8	10	6	5	4	3	2	30	18	36	15
26	31	20	23	1	35	34	33	30	29	28	32	25	27	22	21	19	24	13	18	16	15	10	12	5	9	8	7	4	3	2	36	14	17	6	11
6	11	15	17	18	1	35	34	33	32	30	28	27	23	29	25	24	21	16	13	12	8	14	10	9	7	5	4	3	2	36	19	20	22	26	31
5	10	14	16	7	19	1	35	34	33	31	29	28	26	25	24	20	22	15	17	13	12	11	9	8	6	4	3	2	36	18	30	21	23	27	32
4	9	13	12	21	20	14	7	5	36	35	34	31	29	27	26	22	19	18	15	11	10	8	6	3	2	1	32	30	23	17	16	25	24	28	33
3	8	25	24	23	22	17	18	16	7	5	4	36	35	31	28	27	26	11	10	9	6	2	1	33	32	30	21	19	20	15	14	13	12	29	34
2	30	29	28	27	26	6	1	19	20	12	21	3	24	23	22	5	4	33	32	15	14	13	34	16	25	17	18	36	31	11	10	9	8	7	35
36	35	34	33	32	31	11	15	10	9	13	14	8	25	17	18	16	7	30	21	19	20	12	29	23	24	28	27	22	26	6	5	4	3	2	1