Fast enumeration of magic squares

2025.07 Hidetoshi Mino

Magic square of order N

	1	2	…	N
1	7	3	…	21
2	14	8	…	10
⋮	⋮	⋮	⋱	⋮
N	1	9	…	5

- A square grid of distinct integers ($1 \sim N^2$) where the sum of the integers in each row, each column, and both diagonals is the same.

Magic square of order N

	1	2	…	N
1	7	3	•••	21
2	14	8	•••	10
⋮	⋮	⋮	⋱	⋮
N	1	9	•••	5

- A square grid of distinct integers (1～N^2) where the sum of the integers in each row, each column, and both diagonals is the same.

Magic square of order N

	1	2	…	N
1	7	3	…	21
2	14	8	…	10
⋮	⋮	⋮	⋱	⋮
N	1	9	…	5

- A square grid of distinct integers ($1 \sim N^2$) where the sum of the integers in each row, each column, and both diagonals is the same.
- The sum is called 'magic sum' and equals to $(N + N^3)/2$.

Enumeration of magic squares

N	The number of magic squares up to rotations and reflections		
3	1		
4	880	1693	F. de Bessy
5	275305224	1973	R. Schroeppel
6	17753889197660635632	2024	H. Mino

Representations of
NxN square grid of distinct integers
(not necessarily magic)

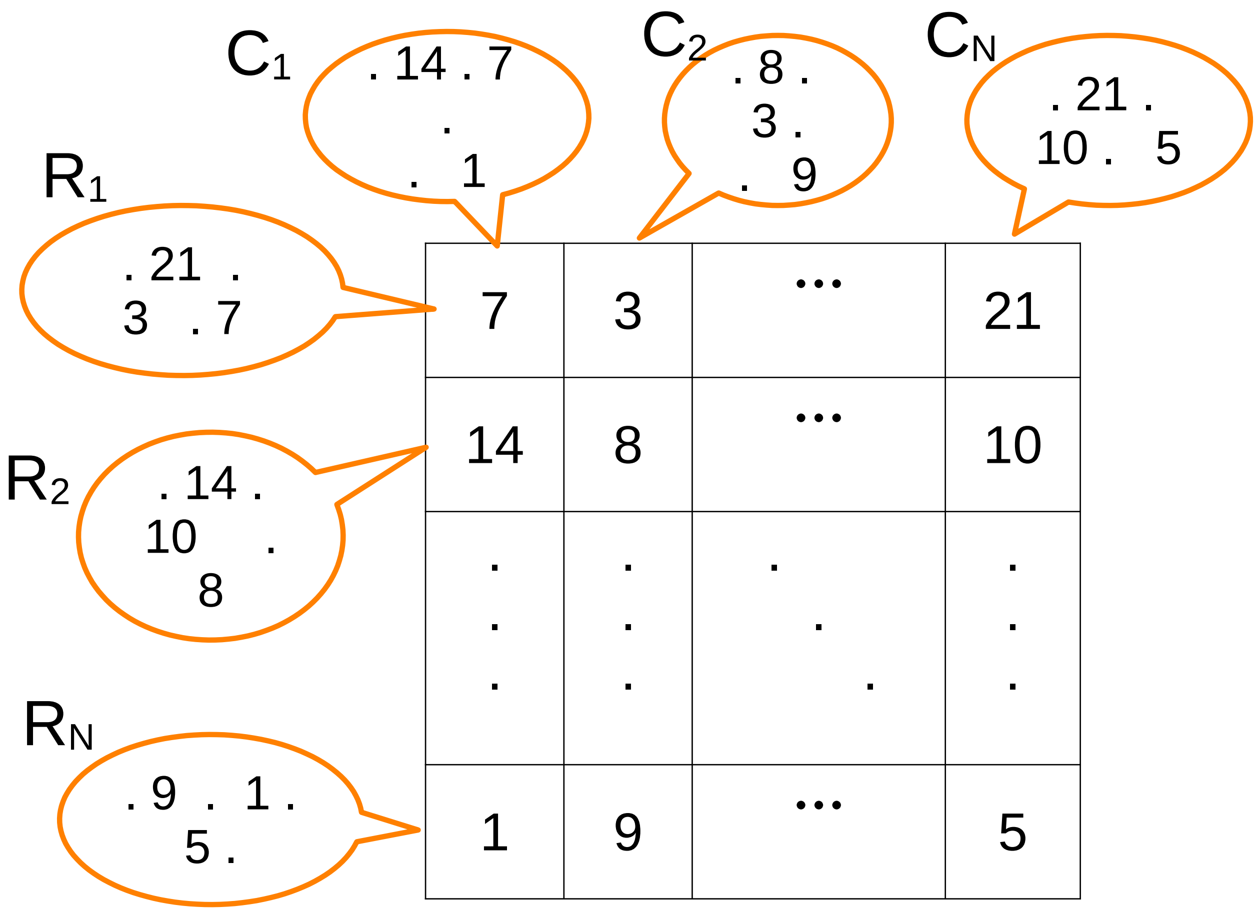
Matrix representation

	1	2	…	N
1	7	3	…	21
2	14	8	…	10
⋮	⋮	⋮	⋱	⋮
N	1	9	…	5

A mapping from a row column indices pair to an element value

example: 2,1 → 14

Row sets-Column sets representation



$C_i \cap C_j = \varnothing$ for $i \neq j$

$R_i \cap R_j = \varnothing$ for $i \neq j$

$| C_i \cap R_j | = 1$

One to one correspondence

Row sets Column Sets

$R_1, R_2, \ldots R_N$
$C_1, C_2, \ldots C_N$

$Ri \cap Rj = \varnothing$ for $i \neq j$
$Ci \cap Cj = \varnothing$ for $i \neq j$

$| Ri \cap Cj | = 1$

$Ri \cap Cj = \{ Mi,j \}$

Matrix

7	3	…	21
14	8	…	10
⋮	⋮	⋱	⋮
1	9	…	5

Binary coding of a distinct integer set

example:

$$\{ 7, 3, 10 \} \Rightarrow 2^{7-1} + 2^{3-1} + 2^{10-1}$$
$$= 1000000_2 + 100_2 + 1000000000_2$$
$$= 1001000100_2$$
$$= 580_{(10)}$$

Natural and convenient representation of a set.

Set operations (intersection, union, complement, etc) can be performed by bitwise logical operations.

Order relation can be defined by the integer comparison.

Binary coded Row sets-Column sets (BRC) representation for the magic square enumeration

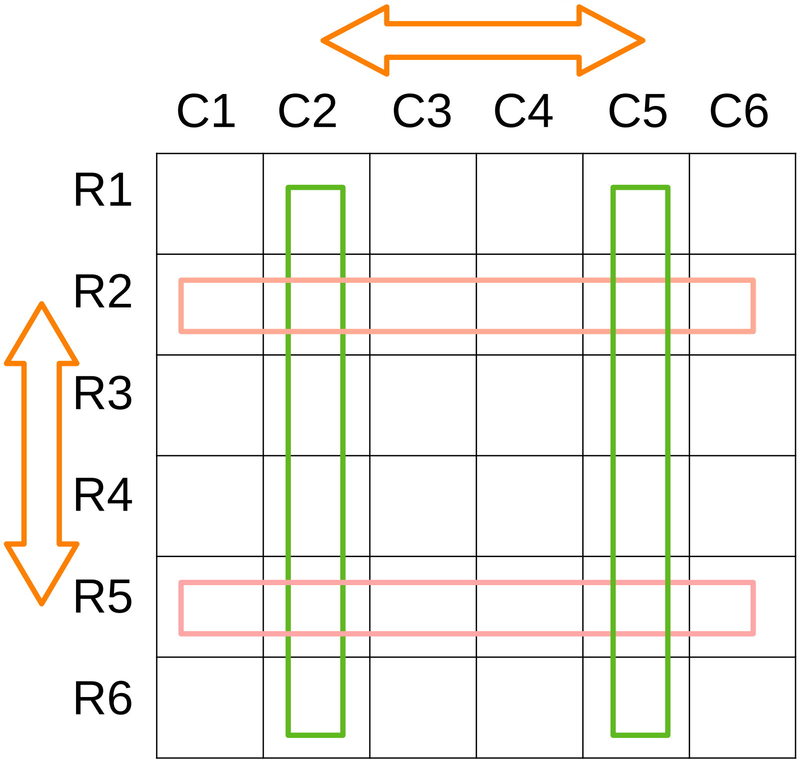
BRC representation for the magic square enumeration

- Merits:
  - Fewer primitive data objects
    - 2N (long int) vs N^2(short int)

      Utilizes the power of the long word CPU/GPU
  - The row and column sum constraints are easily incorporated.
  - Fast set operations
  - Works well with M-transformations.

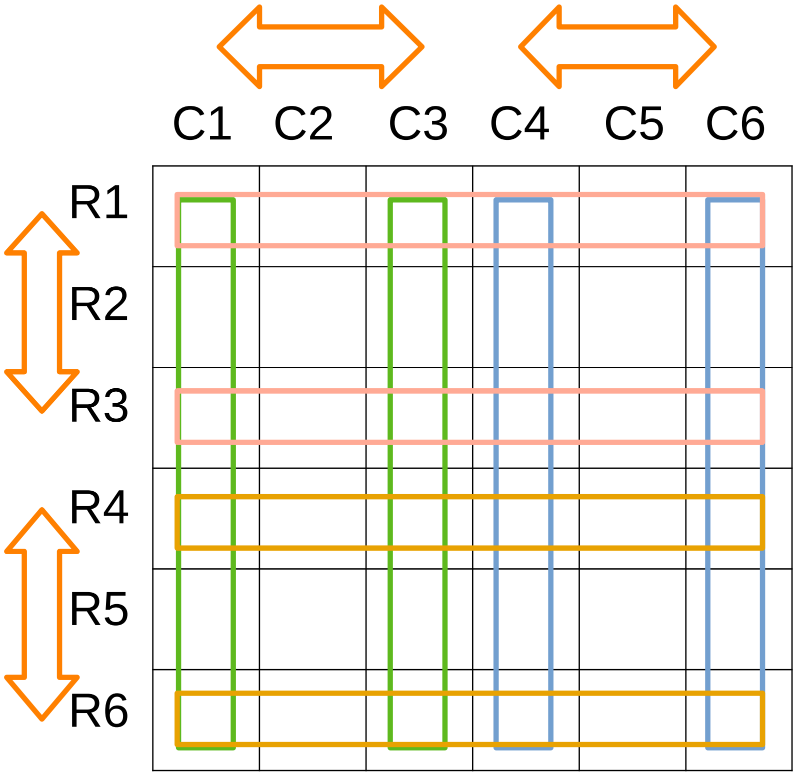
M-transformations

- Transformations which make one magic square into another magic square.
- an example in N=6
  - exchange R2 and R5
  - and
  - exchange C2 and C5



M-transformations

- another example in N=6
  - exchange C1 and C3
  - and
  - exchange C4 and C6
  - and
  - exchange R1 and R3
  - and
  - exchange R4 and R6
- and more ...

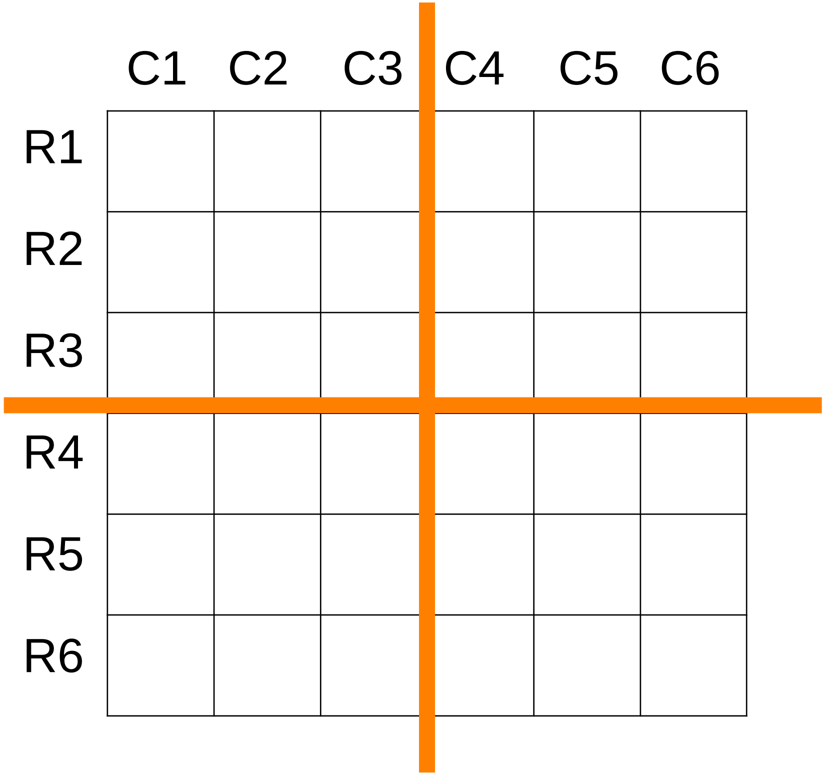


M-transformations

- Permuting rows and permute columns conjointly
- The permutation must be symmetric with respect to the center lines.

N	3	4	5	6	7	8
Multiplicity	1	4	4	24	24	192

- Conjoint total reflection is the 180deg rotation and is excluded.



M-transformations

- M-transformations consist of permutations of rows and that of columns.
- All magic squares generated by M-transformations correspond to the same combination of row sets and column sets.

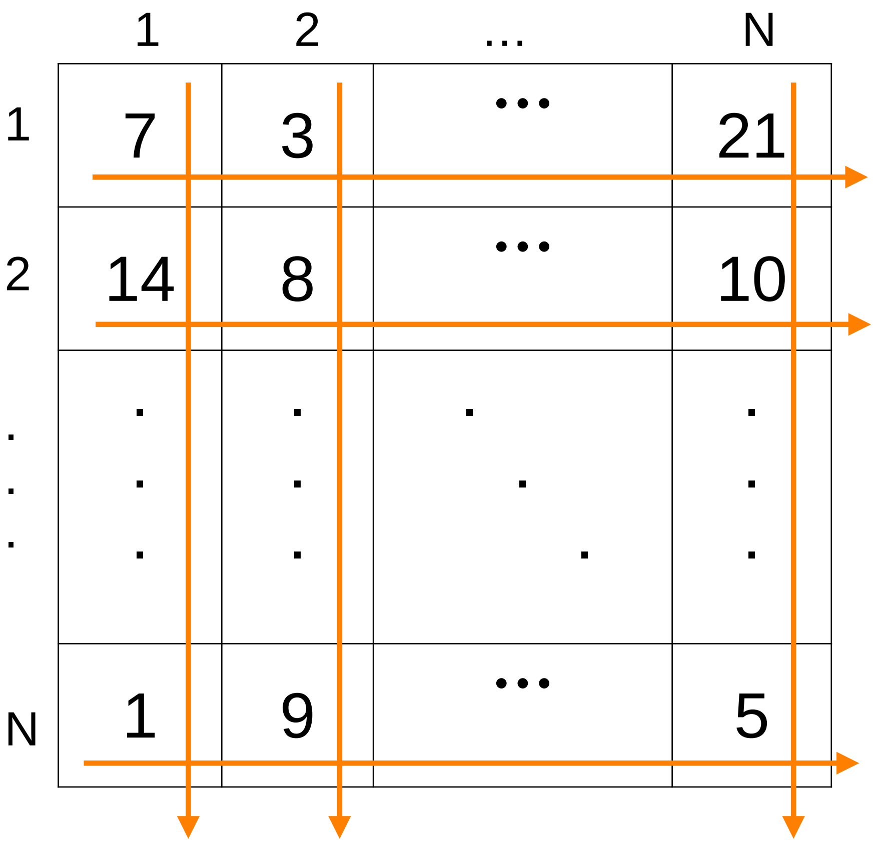
  M-transformations change only the order of sets within row sets and that within column sets.

BRC representation for the magic square enumeration

- Concern:
  - Not immediately clear how the diagonal sum constraints are incorporated.
    - This presentation shows a solution.

Enumerate semi-magic (including magic) squares

Semi-magic (including magic) squares



- A square grid of distinct integers ($1 \sim N^2$) where the sum of the integers in each row, each column are the same.
- Some people count magic's in semi-magic's, others don't.
- We are going to enumerate inclusively in any case.

Semi-magic (including magic) squares

- Let us, for the moment, ignore the diagonal constraints.
  - Semi-magic squares are included int the enumeration.
  - The inclusive enumeration is easier than the exclusive one of magic squares.

Magic series

- Magic series of order N :
  - A set of N distinct integers in the range 1 ∼ N^2 which add up to the magic sum.
  - Example:
    - { 2, 9, 4 } is a magic series of order 3
      - The sum equals to 15.
    - { 10, 7, 14, 3 } is a magic series of order 4
      - The sum equals to 34.

Magic series

- Every Row set Ri and every Column set Cj of a semi-magic or magic square is a magic series.
- We make the list of all magic series as the first step of the enumeration.

N	The number of magic series
3	8
4	86
5	1,394
6	32,134
7	9,957,332

Enumeration of semi-magic squares

Enumerate all possibilities of Ri and Cj such that

$R_1, R_2, \ldots R_N$: magic series

$C_1, C_2, \ldots C_N$: magic series

$Ri \cap Rj = \emptyset$ for $i \neq j$

$Ci \cap Cj = \emptyset$ for $i \neq j$

$| Ri \cap Cj | = 1$

1 to 1 correspondence

7	3	…	21
14	8	…	10
⋮	⋮	⋱	⋮
1	9	…	5

$(N!)^2/4$ multiplicity

We only have to enumerate Ri and Cj such that

$R_1, > R_2, > \ldots > R_N$

$C_1, > C_2, > \ldots > C_N$

All permutations of R’s and that of C’s generate different semi-magic and magic squares. Hence, multiply by $(N!)^2/4$.

To eliminate the remaining trivial duplicates, add a constraint $R_1 > C_1$

Result for N = 6

The result for N= 6:
The number of combinations of Ri and Cj such that

- $R_1, > R_2, > \ldots > R_6$: magic series, $R_i \cap R_j = \varnothing$ for $i \neq j$
- $C_1, > C_2, > \ldots > C_6$: magic series, $C_i \cap C_j = \varnothing$ for $i \neq j$
- $R_1 > C_{1,}$ $| R_i \cap C_j | = 1$

is 729 866 205 597 222 196.

x 6! x 6! / 4 => 94 590 660 245 399 996 601 600.

matches the result by A. Ripatti (2018).

How about the magic square?

Highly optimized code on Nvidia RTX-4090 can completes the semi-magic enumeration roughly in 10,000 hours (14 months).

Can we incorporate the diagonal constraints into this approach without prohibiting overheads?

YES

Fast enumeration of magic squares

A naive approach

- As in the case of semi-magic square, generate R's and C's such that
  - $R_1, > R_2, > \ldots > R_N$: magic series, $R_i \cap R_j = \emptyset$ for $i \neq j$
  - $C_1, > C_2, > \ldots > C_N$: magic series, $C_i \cap C_j = \emptyset$ for $i \neq j$
  - $R_1 > C_{1,}$ $| R_i \cap C_j | = 1$
- Check the diagonal sums for every permutations for the row sets and column sets. $(N!)^2 / 4$ ways of permutations to check.
- It is a correct method, but practically prohibitive.

A naive approach

- As in the case of se… …re, generate R's and C's such that
  - $R_1, > R_2, > ..$ … R… …agic series … $\cap R_j = \varnothing$ for $i \neq j$
  - $C_1, > C_2, > .$ … C_N . …gic series, … $\cap C_j = \varnothing$ for $i \neq j$
  - $R_1 > C_1,$ | … $\cap C_j$ | …
- Check the diag… …l sums for … …ry p… …utations for the row sets and col… …n sets. (N!) … …ays of permutations to ch…
- It is a correct method, but … …tically prohibitive.

Add diagonal candidate magic series

- Generate R's, C's, and X's such that
  - $R_1, > R_2, > \ldots > R_N$: magic series, $R_i \cap R_j = \varnothing$ for $i \neq j$
  - $C_1, > C_2, > \ldots > C_N$: magic series, $C_i \cap C_j = \varnothing$ for $i \neq j$
  - $R_1 > C_1,$ $|R_i \cap C_j| = 1$
  - $X_1 > X_2$: magic series
  - $|X_i \cap R_j| = 1$
  - $|X_i \cap C_j| = 1$
  - $|X_1 \cap X_2| = N \bmod 2$

Add diagonal (candidate) magic series

- The existence of R's, C's, and X's is not sufficient but only necessary for magic squares.
- How do we identify magic squares?
- Given sets of R's and of C's, a semi-magic square is determined.
- Let us see the arrangement of X's on the square.

Arrangement of diagonal candidates

	X1			X2	
		X2	X1		
		X1			X2
X1			X2		
	X2				X1
X2				X1	

- By Permuting R's and C's, can we make X's diagonal?

⇧

- Answer this question without trying many permutations!

Arrangement of X is a permutation

	X1			X2	
		X2	X1		
		X1			X2
X1			X2		
	X2				X1
X2				X1	

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

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

- Elements of X appear once and only once in each row and in each column.
- Arrangement of X represents a permutation of N object, or an element of the symmetric group S_N.
- Let up denote
  - Arrangement of X1 by $d \in S_N$
  - Arrangement of X2 by $u \in S_N$

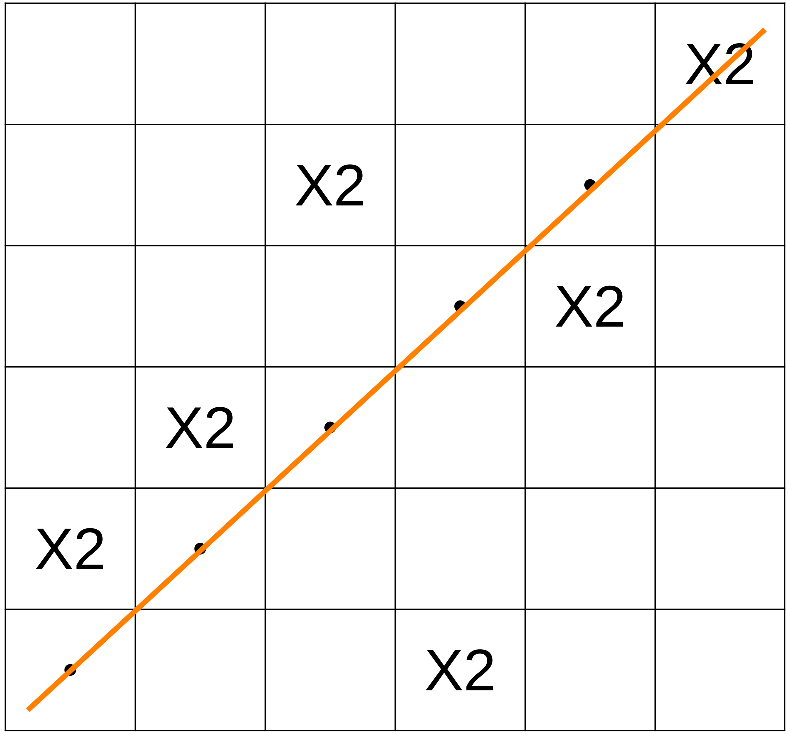
X_1 aligned down diagonal

X1					X2
	X1	X2			
		X1		X2	
	X2		X1		
X2				X1	
			X2		X1

No, in this case. Why?

- By permuting columns(or rows), we can always align X1 down diagonal.
  $d \rightarrow dd^{-1} = e, \quad u \rightarrow ud^{-1}$
- To keep X1 down diagonal,
  - further rearrangement is restricted to conjoint permutations of rows and columns.
- Can we align X2 up diagonal?

Conjugacy relation



- Conjoint permutation of rows and columns is a conjugate transformation:
  - $X \rightarrow P^{-1} X P, \quad X, P \in S_N$
- Can we align X2 upward diagonal?

⇕

- Is ud^{-1} conjugate to the upward diagonal (reversed order) permutation?

ud^{-1} 1 2 3 4 5 6   up diagonal 1 2 3 4 5 6

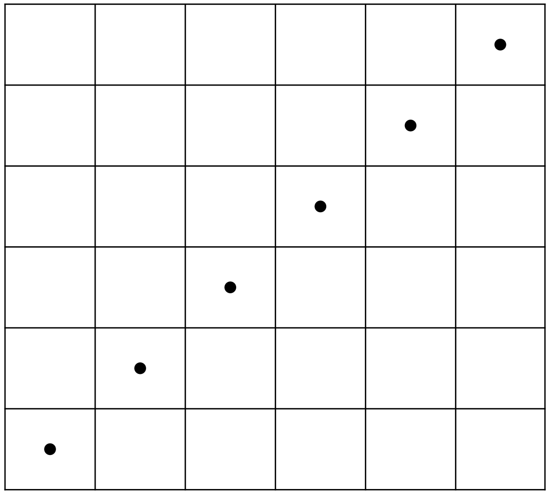
5 4 2 6 3 1   6 5 4 3 2 1

A theorem in the symmetric group

- Two permutations are conjugate if and only if they have the same cycle type.

Cycle type of the upward diagonal permutation

- The cycle type of the upward diagonal (reversed order) permutation consists as many 2-cycles as possible and only one unpaired 1-cycle for odd N.
  - 1↔N, 2↔N-1, 3↔N-2 ….
    - (N+1)/2 is a 1-cycle for odd N



Up diagonal perm.
1 2 3 4 5 6
6 5 4 3 2 1

No need to worry about 1-cycle

X1					X2
	X1	X2			
		X1		X2	
	X2		X1		
X2				X1	
			X2		X1

- Because we have imposed that

  $|X_1 \cap X_2| = N \bmod 2$

- No elements of X_2 appears on the diagonal line for even N.

  Only one appears for odd N.

- This ensure automatically the correct 1-cycle components.

All we have to check

X1					X2
	X1	X2			
		X1		X2	
	X2		X1		
X2				X1	
			X2		X1

- ud^{-1} has no cycles longer than 2,
- or $(ud^{-1})^2 = e$

How to check $(ud^{-1})^2 = e$?

	X1			X2	
		X2	X1		
		X1			X2
X1			X2		
	X2				X1
X2				X1	

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

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

- For a permutation p, we define a function p( ) such that

  p: 1 2 3 … N
  p(1) p(2) p(3) …. p(N)

- $(ud^{-1})^2 = e$ is expressed as

  $u(d^{-1}(u(d^{-1}(1)))) = 1$
  $u(d^{-1}(u(d^{-1}(2)))) = 2$
  $u(d^{-1}(u(d^{-1}(3)))) = 3$
  ….
  How far to go ?
  No need to check all.

How to compute u( ) and d^{-1}( )?

	X1			X2	
		X2	X1		
		X1			X2
X1			X2		
	X2				X1
X2				X1	

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

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

- Provided
  - Intersec(X, C)
    - The common element of two magic series.
  - Elem2row(elem)
  - Elem2col(elem)
    - Which row/column an element belongs to.
- d^{-1}(i) = Elem2col(Intersec(X1, Ri))
- u(i) = Elem2row(Intersec(X2, Ci))

Intersec( ), Elem2row/col[ ]

- Intersec(X, Y)
  - Find the bit position of X & Y (bit-wise and)
  - using Count Leading Zero instruction, or ...
- Elem2row/col[elem]
  - Maintain auxiliary arrays which map elements to row/column.
- See sequel videos for details.

For N=6

if $u(d^{-1}(u(d^{-1}(1)))) == 1$

  we find a 2-cycle $(1, u(d^{-1}(1)))$.

else no magic square, stop

If($u(d^{-1}(1)) \ne 2$) — printed as: If($u(d^{-1}(1))$!= 2)

  check if $u(d^{-1}(u(d^{-1}(2)))) == 2$

else

  check if $u(d^{-1}(u(d^{-1}(3)))) == 3$

- If passed all the above checks, we have two 2-cycles. Remaining elements cannot form a 3-cycle or longer.

Check function at the deepest level

- Bool MagicSquare(R[ ], C[ ], X[ ], elem2row[ ], elem2col[ ]) {

```
    int c1 = elem2col[ Intersec( X[1], R[1] ) ];
    If ( elem2row[ intersec( X[2], C[ c1 ] ] != 1 )
        return false;
    int r2 = ( c1==2 ) ? 3 : 2;
    int c2 = elem2col[ Intersec( X[1], R[ r2 ] ) ];
    return elem2row[ intersec( X[2], C[ c2 ] ) ] == r2;
}
```

Check function at the deepest level

- Bool MagicSquare(R[ ], C[ ], X[ ], elem2row[ ], elem2col[ ]) {
  - No loops
  - No function calls
  - No large tables
  - No side effects (important for parallelization)

}

A graphical view

	X1			X2	
		X2	X1		
		X1			X2
X1			X2		
	X2				X1
X2				X1	

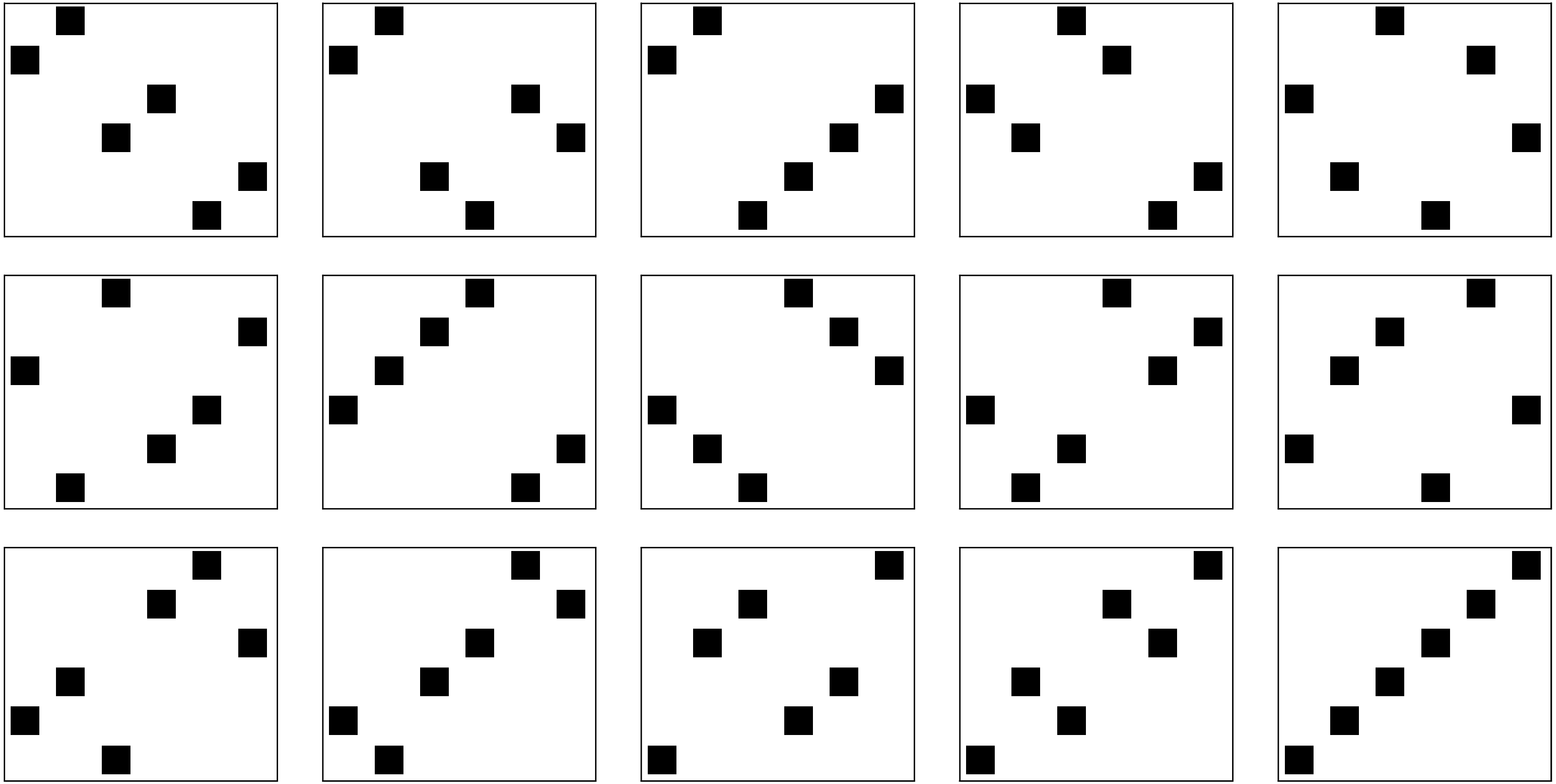
A graphical view

X1					X2
	X1	X2			
		X1		X2	
	X2		X1		
X2				X1	
			X2		X1

- Which arrangement of X2 after X1 diagonalized leads to a magic square?

Symmetric patterns lead to magic squares

Symmetric patterns of up-diagonal candidates for N=6 (A conjugacy class of S_6)



A graphical view

X1					X2
	X1	X2			
		X1		X2	
	X2		X1		
X2				X1	
			X2		X1

- Place 1 in place of X2 and 0 otherwise.

Permutation matrix

0	0	0	0	0	1
0	0	1	0	0	0
0	0	0	0	1	0
0	1	0	0	0	0
1	0	0	0	0	0
0	0	0	1	0	0

- A permutation matrix P is an orthogonal matrix which represents a permutation.
- $P^2 = I$ for the eligible arrangements.
- $P^2 = I \rightarrow P = P^{-1} \rightarrow P = P^T$
- P must be symmetric.

Result for N = 6

$N = 6$

- Generate R's, C's, and X's such that
  - $R_1, > R_2, > \ldots > R_6$: magic series, $R_i \cap R_j = \varnothing$ for $i \neq j$
  - $C_1, > C_2, > \ldots > C_6$: magic series, $C_i \cap C_j = \varnothing$ for $i \neq j$
  - $R_1 > C_{1,}$ $|R_i \cap C_j| = 1$
  - $X_1 > X_2$: magic series
  - $|X_i \cap R_j| = 1$
  - $|X_i \cap C_j| = 1$
  - $|X_1 \cap X_2| = N \bmod 2$
- And check if MagicSquare(Rs, Cs, Xs) return true.

Result for N = 6

- Generate R’s, C’s, and X’s such that
  - ….
- And check if MagicSquare(R’s, C’s, X’s) returns true.

- There are
  - 739 745 383 235 859 818 combinations.

M-transformation

- M-transformations consist of permutation of rows and that of columns and preserve elements in diagonal lines.
- All magic squares generated by M-transformations correspond to the same combination of R’s, C’s, and X’s.
- Every time the function MagicSquare(R[ ], C[ ], X[ ], elem2row, elem2col) return true, we get all magic squares generated by the M-transformation as a set.
- Simply multiply the result by 24 for N=6.

Result for N = 6

- There are
  - 739 745 383 235 859 818 combinations.
  - x24 = 17 753 889 197 660 635 632 magic squares.

Computation time

- An optimized code running on a single RTX-4090 system enumerates 6x6 magic squares in about

  54,000 hours. = 6 years and 3 months
- The final result was obtained by performing the enumeration twice to detect and correct hardware errors.

  The entire calculation took about 2 years of real time using many GPU resources on cloud services.

To be explained

To be explained

- High degree parallelization
  - Roughly 20,000 threads on an RTX-4090
- Factoring, optimizations, and tricks
  - Some are GPU specific
- How to maintain elem2row/column tables
- Halving the enumeration using complementarity
- Utilizing cheap resources on clouds
- All the above are indispensable for N=6 enumeration to be feasible.
- See sequel videos for details.

References

- See the description text of this video.