URGENT

Dear Computer Science Majors,

My musical *In Transit* is opening on Broadway **tonight**!

Just this morning I learned that my 90 minute show needs to be divided exactly into two 45 minute halves. (Couldn't they just *reschedule* that fire alarm test?!)

I have contacted everyone that I know and they have not been able to divide the shows by hand. Someone suggested that this could be a Computer Science problem — something about algorithms and complexity? — so I am turning to you.

As a Williams College alum I know that your education has been outstanding, so I know that you can do this.

The list of performances and their times is attached. The performances can be reordered however you like, but they must divide into two 45 minute halves.

Thank you so much! Go Ephs!

Sincerely,

Kristen Anderson-Lopez

"Deep Beneath the City"	193
"Not There Yet"	131
"Do What I Do"	180
"Four Days Home"	277
"Broke"	345
"Saturday Night Obsession"	234
"Wingman"	454
"But, Ya Know"	264
"Not There Yet (Reprise)"	377
"Keep It Goin'"	178
"A Little Friendly Advice"	372
"Choosing Not to Know"	330
"The Moving Song"	299
"We Are Home"	434
"Getting There"	520
"Finale"	812
	5400 sec (90 min)

Gray Code

Objects

- Binary strings of length n

Gray Code

Objects

- Binary strings of length n

0 0 0 1 0 0 1 1

Gray Code

Objects

- Binary strings of length n

Operation

- Complement one bit

Gray Code

Objects

- Binary strings of length n

0 0 0 1 0 0 1 1

Operation

- Complement one bit

  (Hamming distance one)

Gray Code

Objects

- Binary strings of length n

0 0 0 1 0 0 1 1

Operation

- Complement one bit
  (Hamming distance one)

before: 0 0 0 1 0 0 1 1

Gray Code

Objects

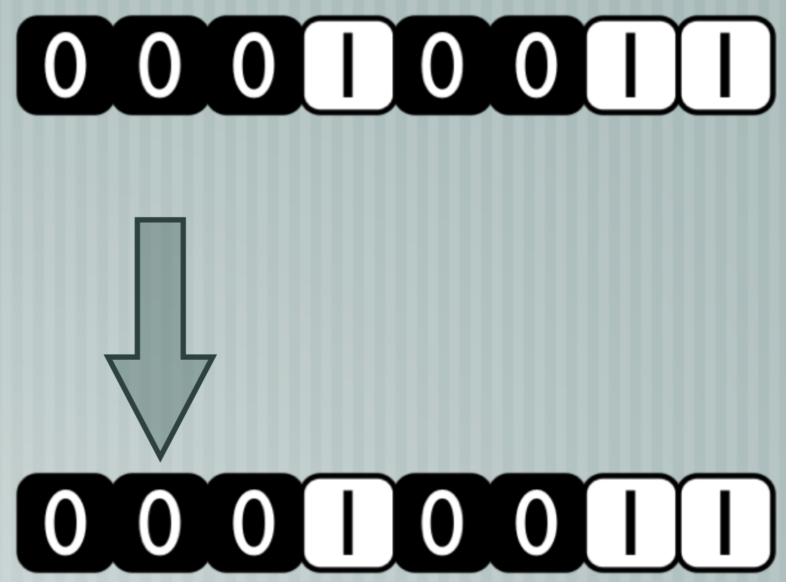
Binary strings of length n

Operation

Complement one bit

(Hamming distance one)

before:



Gray Code

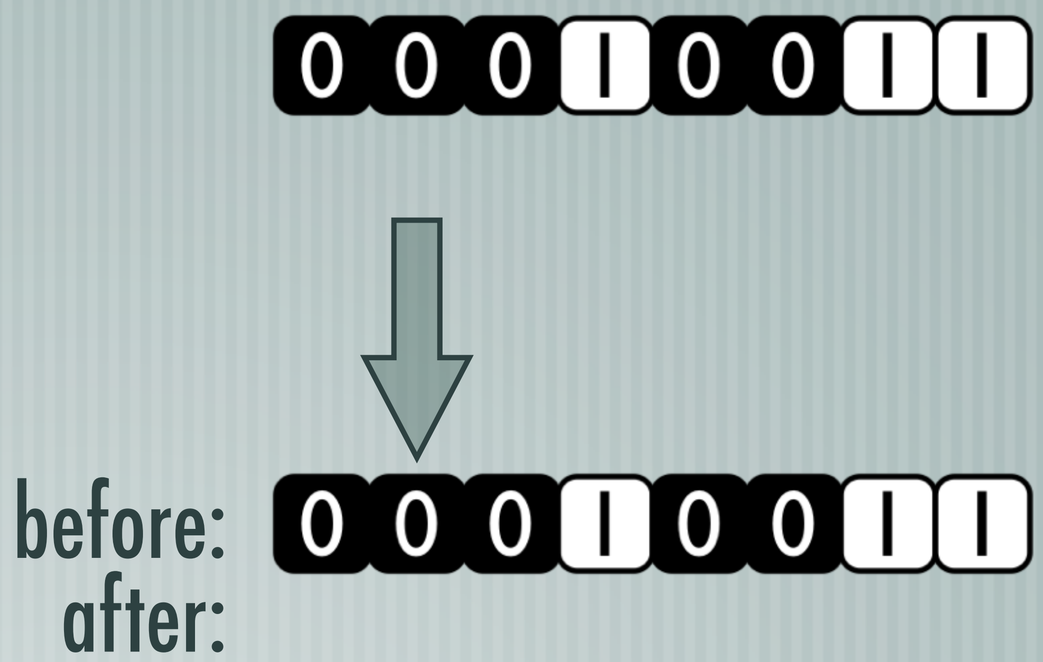
Objects

- Binary strings of length n

Operation

- Complement one bit

  (Hamming distance one)



Gray Code

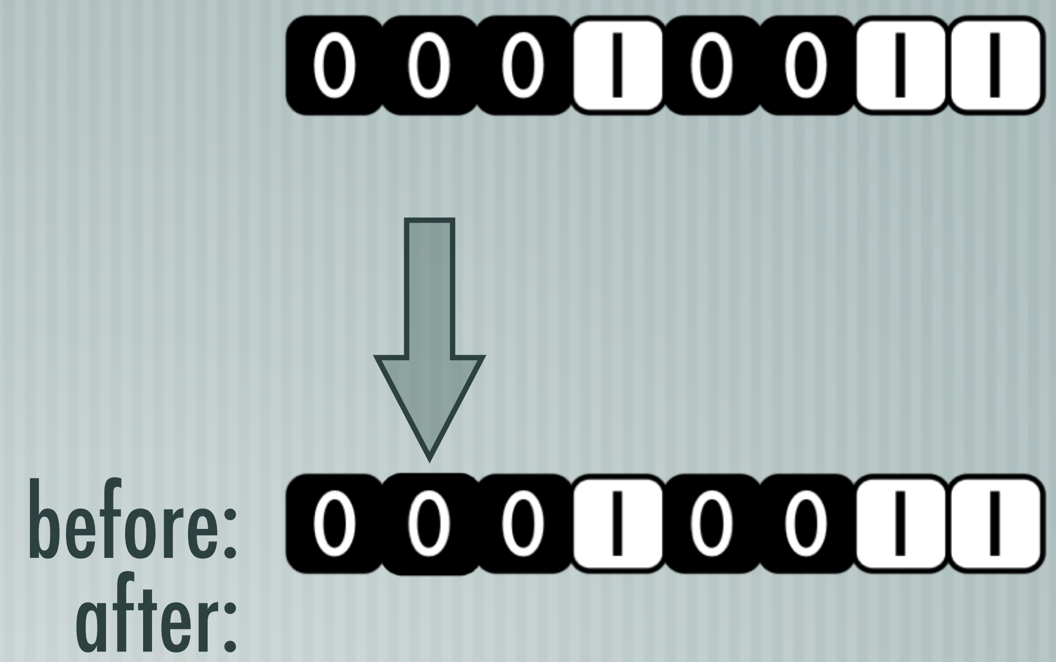
Objects

- Binary strings of length n

Operation

- Complement one bit

  (Hamming distance one)



Gray Code

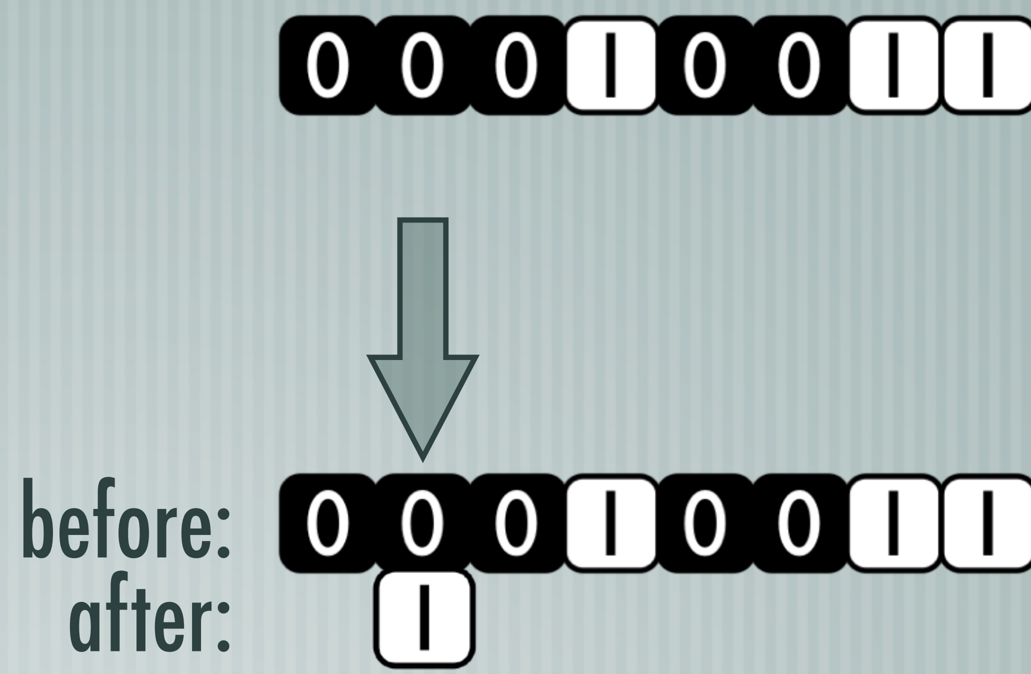
Objects

- Binary strings of length n

Operation

- Complement one bit

  (Hamming distance one)



Gray Code

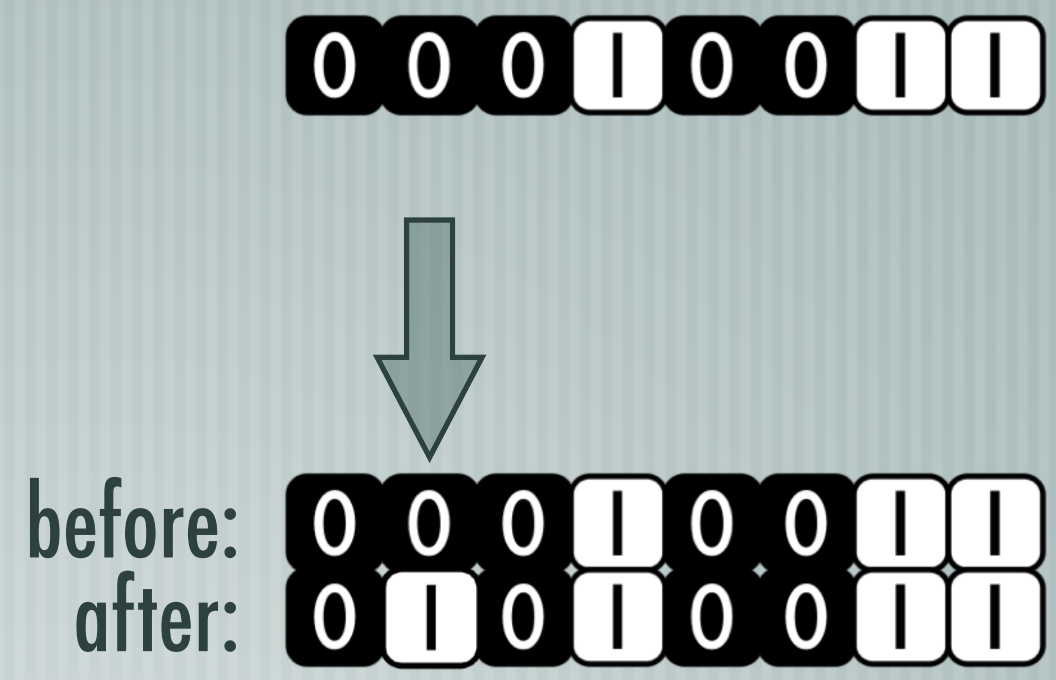
Objects

- Binary strings of length n

Operation

- Complement one bit

  (Hamming distance one)



Gray Code

Objects

- Binary strings of length n

Operation

- Complement one bit

Gray Code

Lexicographic order

0 0 0 0

Objects

Binary strings of length n

Operation

Complement one bit

n = 4

Gray Code

Lexicographic order

0 0 0 0
0 0 0 1

Objects

- Binary strings of length n

Operation

- Complement one bit

n = 4

Gray Code

Lexicographic order

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

Objects

- Binary strings of length n

Operation

- Complement one bit

n = 4

Gray Code

Lexicographic order

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

Objects

- Binary strings of length n

Operation

- Complement one bit

n = 4

Gray Code

Lexicographic order

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

n = 4

Objects

Binary strings of length n

Operation

Complement one bit

Gray Code

Objects

- Binary strings of length n

Operation

- Complement one bit

Lexicographic order

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

n = 4

Gray Code

Objects

- Binary strings of length n

Operation

- Complement one bit

Gray code

Lexicographic order

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

n = 4

NOT a Gray code

Gray Code

Objects

- Binary strings of length n

Operation

- Complement one bit

Gray code

0	0	0	0

n = 4

Lexicographic order

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

n = 4

NOT a Gray code

Gray Code

Objects

- Binary strings of length n

Operation

- Complement one bit

Gray code

0	0	0	0
0	0	0	1

n = 4

Lexicographic order

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

n = 4

NOT a Gray code

Gray Code

Objects

- Binary strings of length n

Operation

- Complement one bit

Gray code

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

n = 4

Lexicographic order

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

n = 4

NOT a Gray code

Gray Code

Objects

- Binary strings of length n

Operation

- Complement one bit

Gray code

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

n = 4

Lexicographic order

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

n = 4

NOT a Gray code

Gray Code

Objects

- Binary strings of length n

Operation

- Complement one bit

Gray code

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

???

n = 4

Lexicographic order

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

n = 4

NOT a Gray code

Gray Code

Objects

Binary strings of length n

Operation

Complement one bit

The Gray Code is <u>cyclic</u> if the last and first strings differ in one bit.

Gray code

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

???

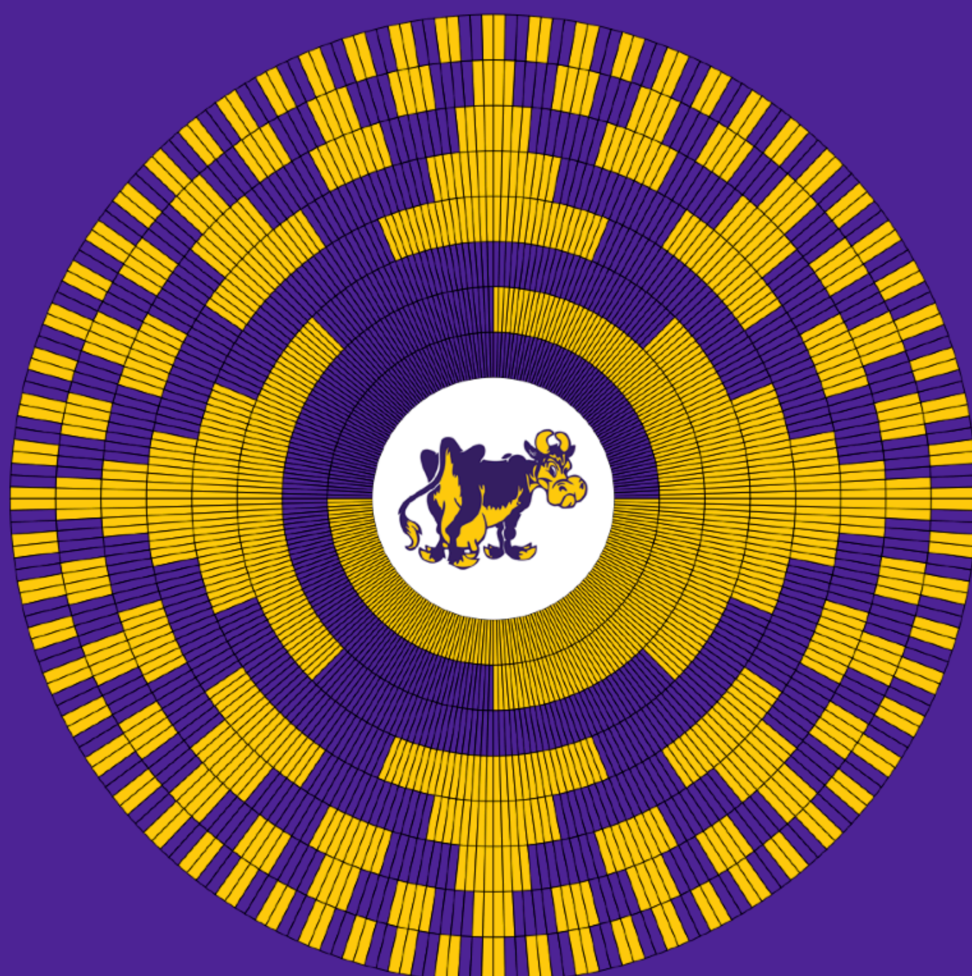
n = 4

Lexicographic order

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

n = 4

NOT a Gray code



Gray Code

A Gray Code has the following properties:

- Every n-bit binary string is listed exactly once.
- Consecutive strings differ in exactly one bit.

A Gray code for the 2-bit binary strings.

A Gray code for the 3-bit binary strings.

Activity

Arrange the 4-bit binary strings as a Gray code.

Questions

- Is your Gray code cyclic?
- How could you build a Gray code for any n?

Binary Reflected Gray Code (BRGC)

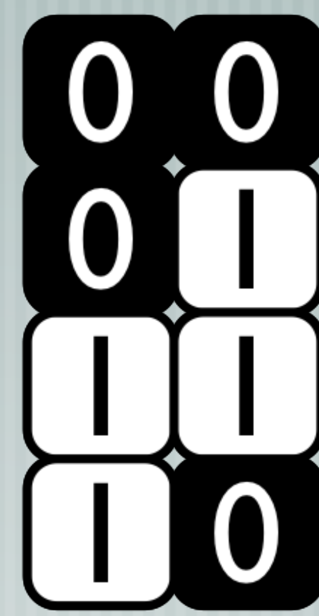
The BRGC for n-bits uses two copies of the BRGC for (n-1)-bits:

- Reflect the second copy.
- Prefix 0 to each string in the first copy.
- Prefix 1 to each string in the second copy.

Binary Reflected Gray Code (BRGC)

The BRGC for n-bits uses two copies of the BRGC for (n-1)-bits:

- Reflect the second copy.
- Prefix 0 to each string in the first copy.
- Prefix 1 to each string in the second copy.



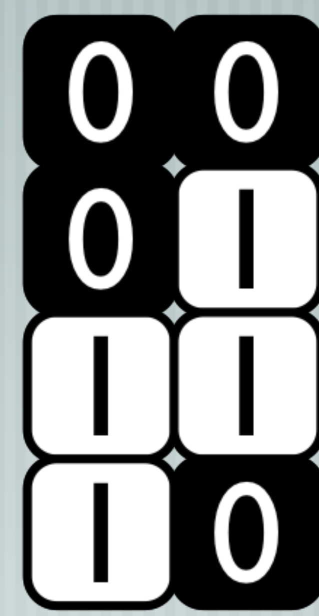
n = 2

n = 3

Binary Reflected Gray Code (BRGC)

The BRGC for n-bits uses two copies of the BRGC for (n-1)-bits:

- Reflect the second copy.
- Prefix 0 to each string in the first copy.
- Prefix 1 to each string in the second copy.



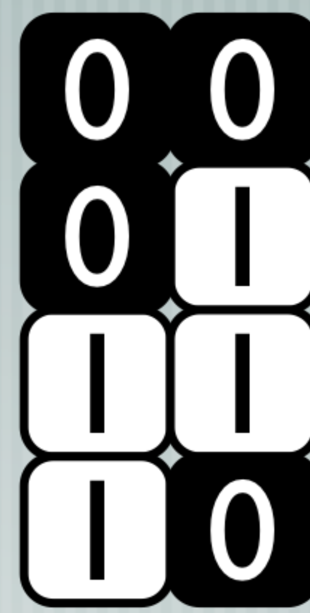
n = 2

n = 3

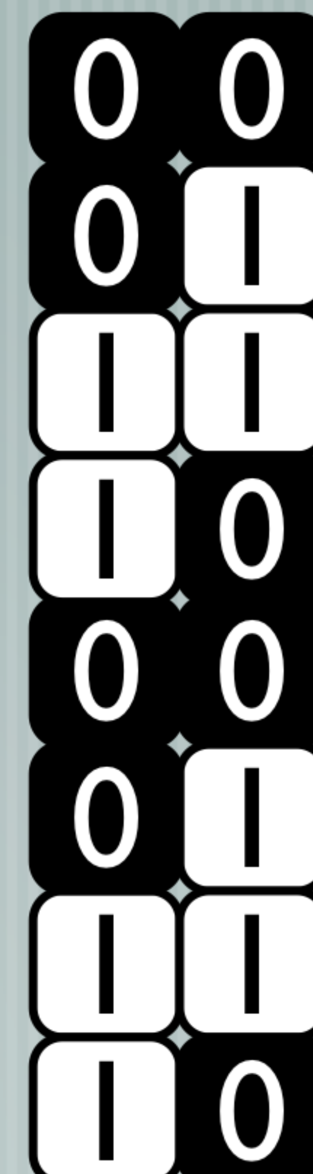
Binary Reflected Gray Code (BRGC)

The BRGC for n-bits uses two copies of the BRGC for (n-1)-bits:

- Reflect the second copy.
- Prefix 0 to each string in the first copy.
- Prefix 1 to each string in the second copy.



n = 2

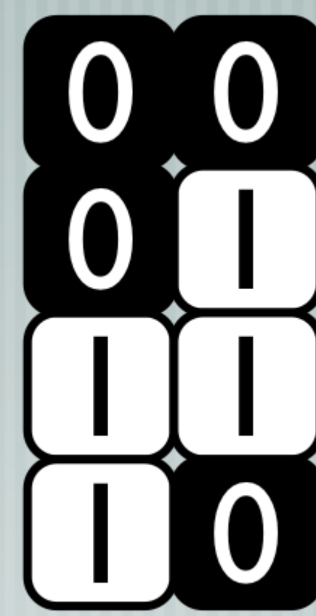


n = 3

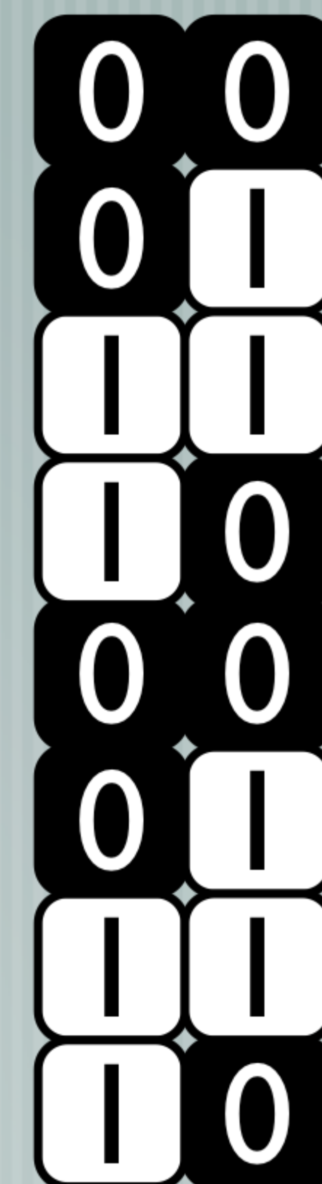
Binary Reflected Gray Code (BRGC)

The BRGC for n-bits uses two copies of the BRGC for (n-1)-bits:

- Reflect the second copy.
- Prefix 0 to each string in the first copy.
- Prefix 1 to each string in the second copy.



n = 2

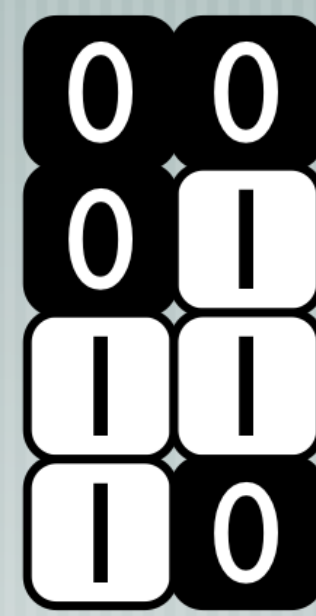


n = 3

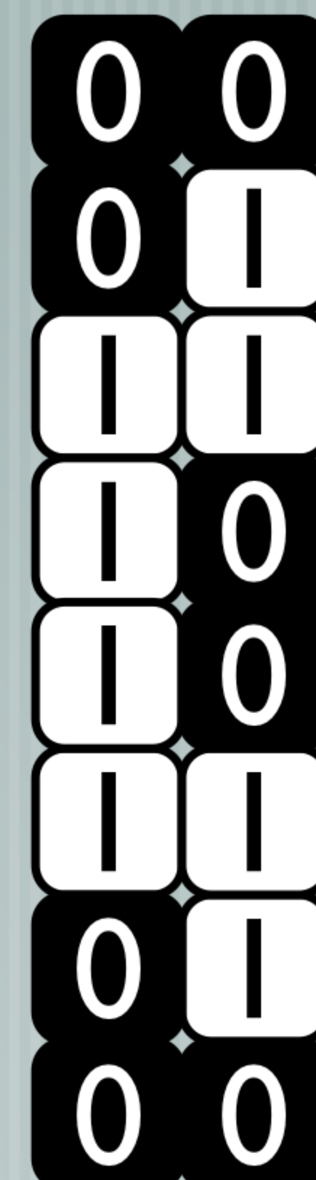
Binary Reflected Gray Code (BRGC)

The BRGC for n-bits uses two copies of the BRGC for (n-1)-bits:

- Reflect the second copy.
- Prefix 0 to each string in the first copy.
- Prefix 1 to each string in the second copy.



n = 2

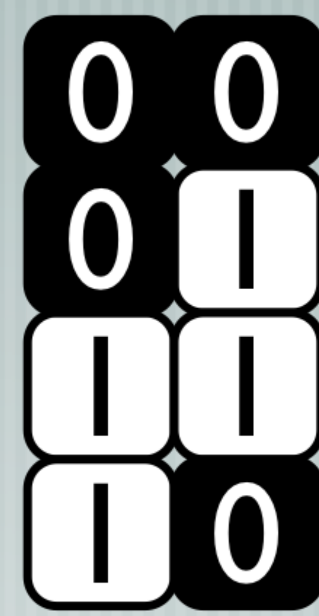


n = 3

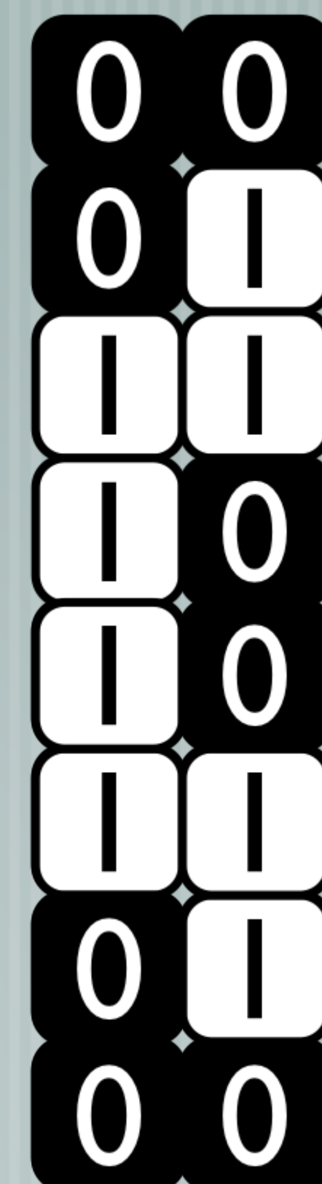
Binary Reflected Gray Code (BRGC)

The BRGC for n-bits uses two copies of the BRGC for (n-1)-bits:

- Reflect the second copy.
- Prefix 0 to each string in the first copy.
- Prefix 1 to each string in the second copy.



n = 2

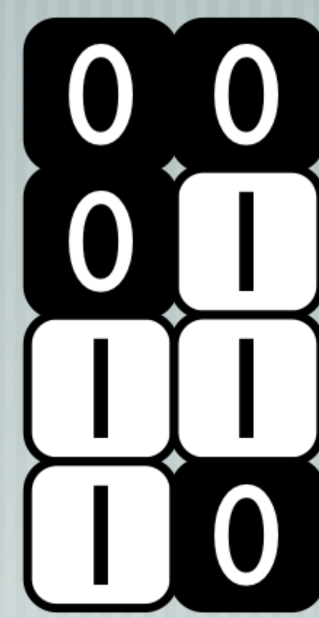


n = 3

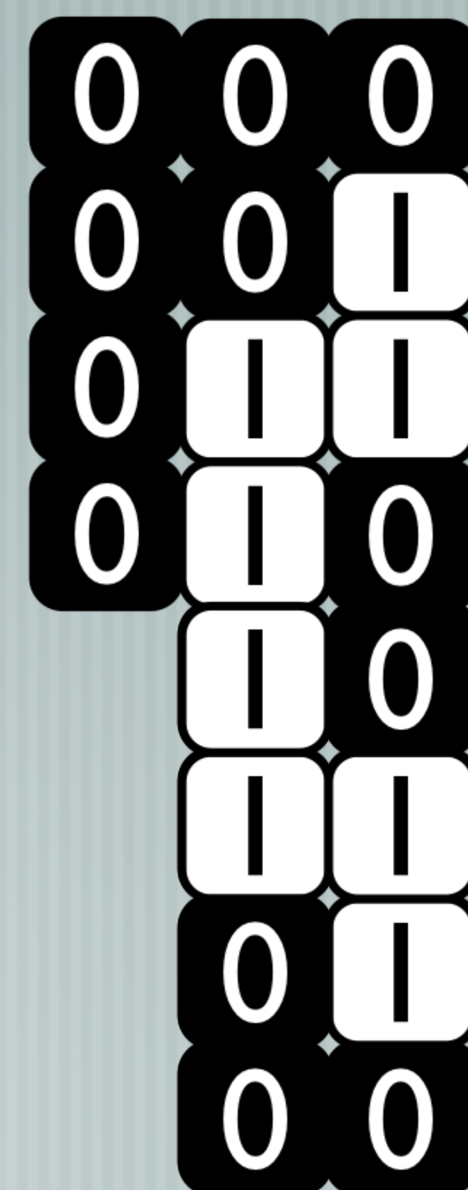
Binary Reflected Gray Code (BRGC)

The BRGC for n-bits uses two copies of the BRGC for (n-1)-bits:

- Reflect the second copy.
- Prefix 0 to each string in the first copy.
- Prefix 1 to each string in the second copy.



n = 2

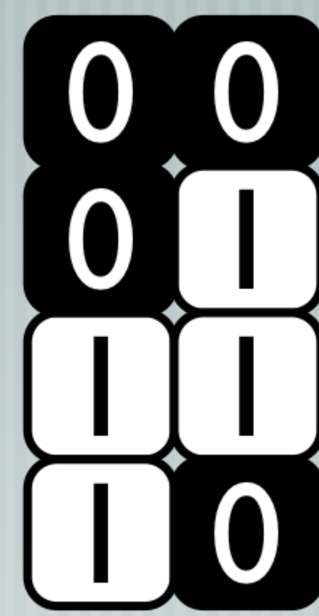


n = 3

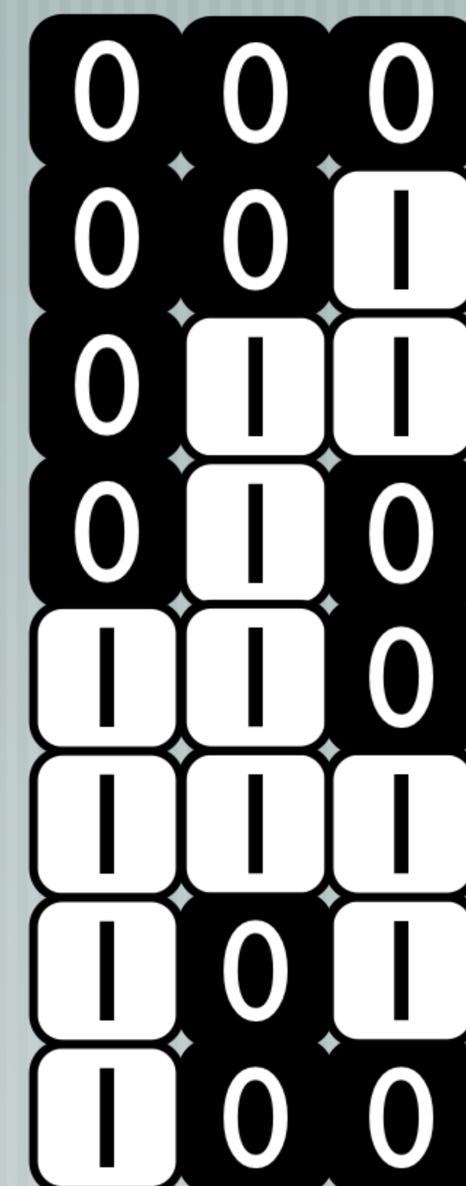
Binary Reflected Gray Code (BRGC)

The BRGC for n-bits uses two copies of the BRGC for (n-1)-bits:

- Reflect the second copy.
- Prefix 0 to each string in the first copy.
- Prefix 1 to each string in the second copy.



n = 2



n = 3

Ruler Sequence

The sequence of complemented bits in the BRGC for n-bits uses two copies of the sequence for (n-1)-bits:

Ruler Sequence

The sequence of complemented bits in the BRGC for n-bits uses two copies of the sequence for (n-1)-bits:

Insert n between the two copies.

Ruler Sequence

The sequence of complemented bits in the BRGC for n-bits uses two copies of the sequence for (n-1)-bits:

- Insert n between the two copies.

1
2
1

n = 2

n = 3

Ruler Sequence

The sequence of complemented bits in the BRGC for n-bits uses two copies of the sequence for (n-1)-bits:

- Insert n between the two copies.

1
2
1

n = 2

1
2
1

n = 3

Ruler Sequence

The sequence of complemented bits in the BRGC for n-bits uses two copies of the sequence for (n-1)-bits:

- Insert n between the two copies.

n = 2	n = 3
1	1
2	2
1	1
	1
	2
	1

Ruler Sequence

The sequence of complemented bits in the BRGC for n-bits uses two copies of the sequence for (n-1)-bits:

- Insert n between the two copies.

n = 2	n = 3
1	1
2	2
1	1
	3
	1
	2
	1

Applications

The binary reflected Gray codes has thousands of applications.

Partition Problem

NP-Complete decision problem:

Partition Problem

NP-Complete decision problem:

- There are n objects

Partition Problem

NP-Complete decision problem:

- There are n objects
- Object i has weight(i)

Partition Problem

NP-Complete decision problem:

- There are n objects
- Object i has weight(i)

Can the objects be partitioned into two equal weight subsets?

Partition Problem

NP-Complete decision problem:

- There are n objects
- Object i has weight(i)

Can the objects be partitioned into two equal weight subsets?

1. Bowie	3:15
2. Robots	3:43
3. Business Time	4:05
4. Hiphopopotamus	2:09
5. Hurt Feelings	2:38
6. Fashion is Danger	2:20
7. Carol Brown	3:26

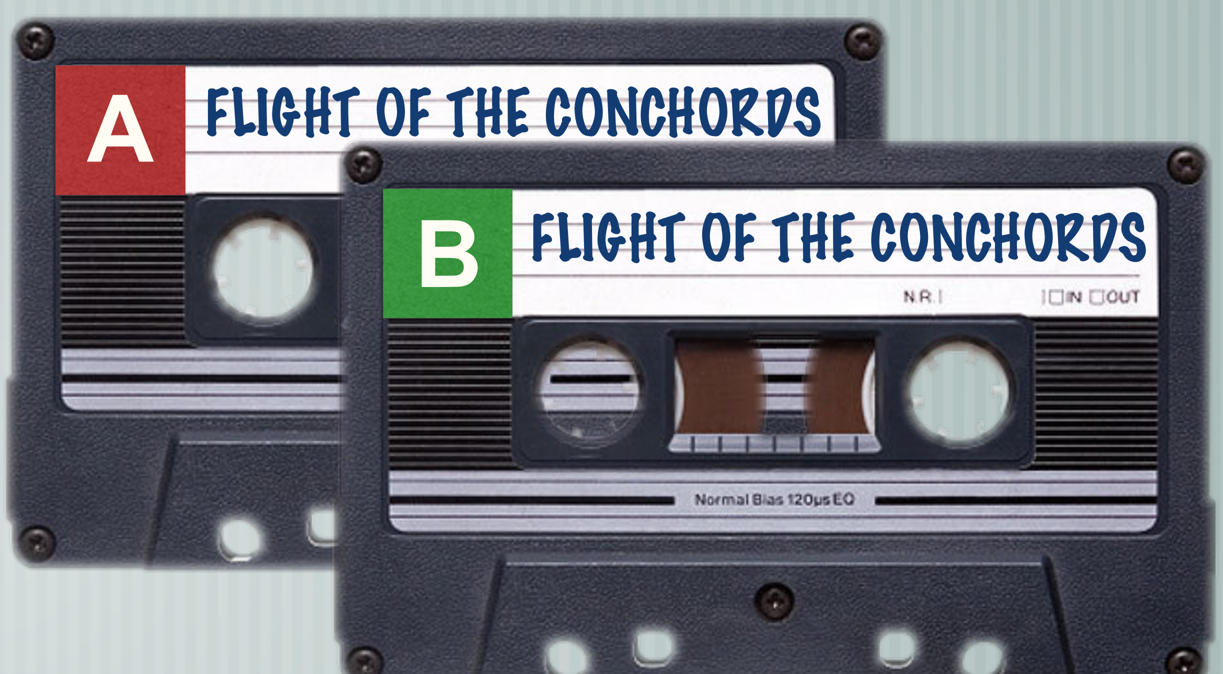
Partition Problem

NP-Complete decision problem:

- There are n objects
- Object i has weight(i)

Can the objects be partitioned into two equal weight subsets?

1. Bowie	3:15
2. Robots	3:43
3. Business Time	4:05
4. Hiphopopotamus	2:09
5. Hurt Feelings	2:38
6. Fashion is Danger	2:20
7. Carol Brown	3:26



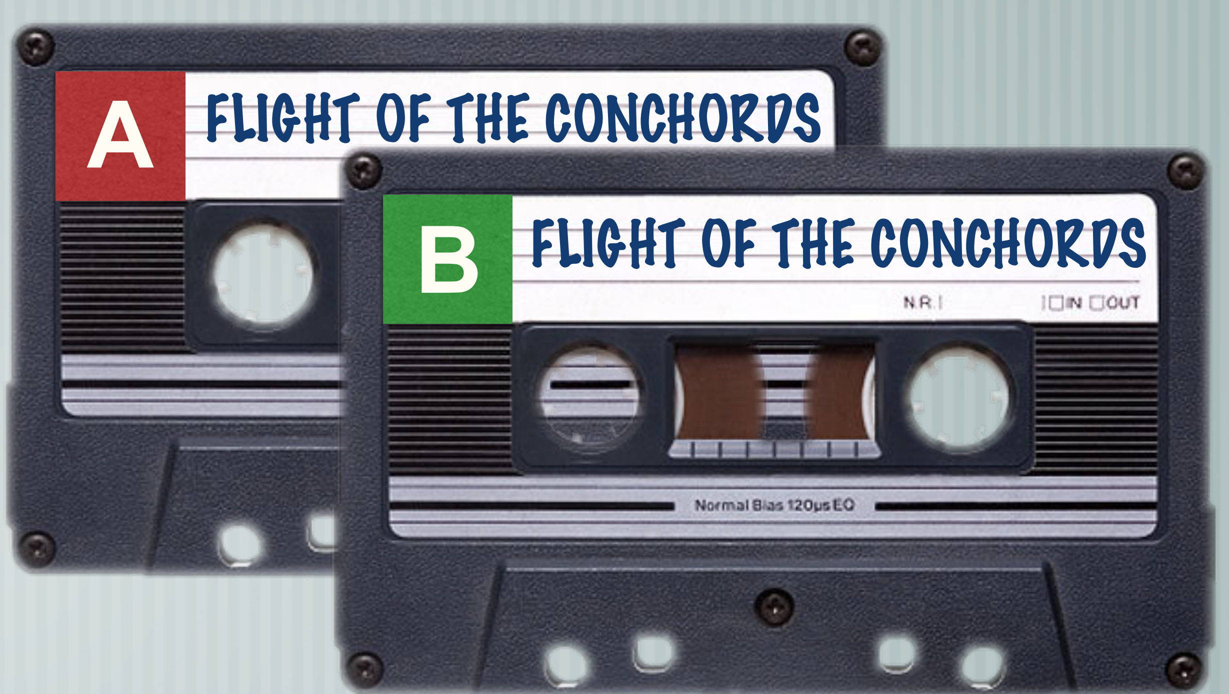
Partition Problem

NP-Complete decision problem:

- There are n objects
- Object i has weight(i)

Can the objects be partitioned into two equal weight subsets?

1. Bowie	3:15
2. Robots	3:43
3. Business Time	4:05
4. Hiphopopotamus	2:09
5. Hurt Feelings	2:38
6. Fashion is Danger	2:20
7. Carol Brown	3:26



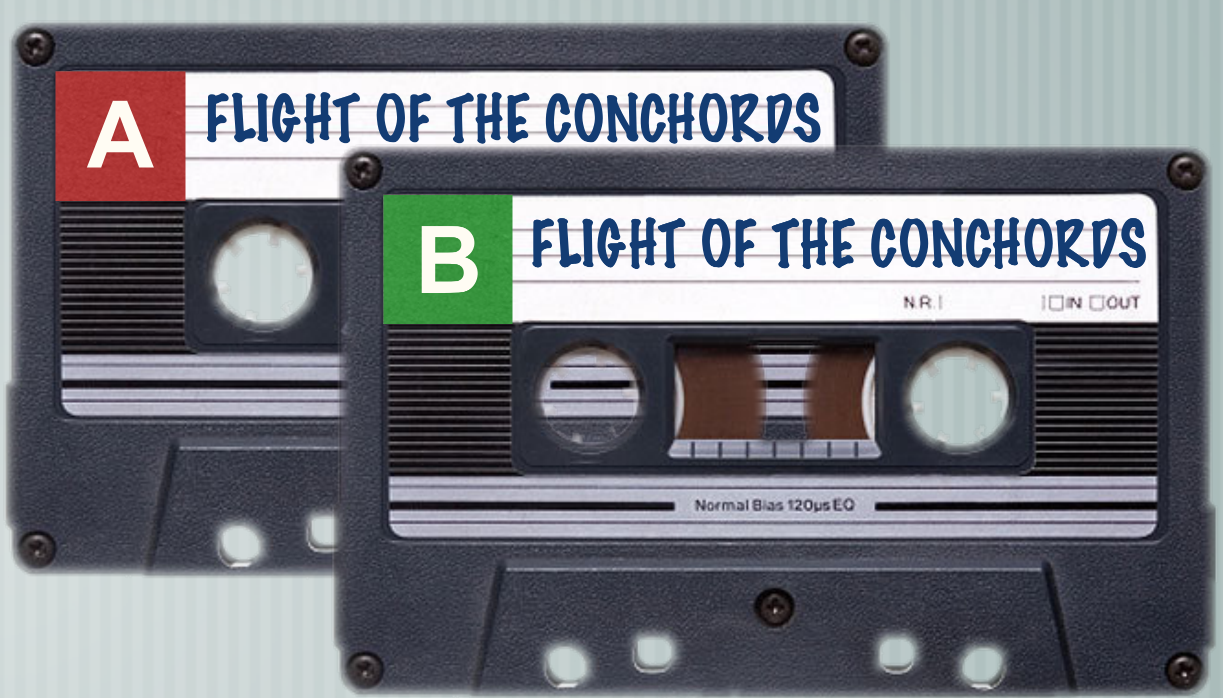
Partition Problem

NP-Complete decision problem:

- There are n objects
- Object i has weight(i)

Can the objects be partitioned into two equal weight subsets?

1. Bowie	3:15
2. Robots	3:43
3. Business Time	4:05
4. Hiphopopotamus	2:09
5. Hurt Feelings	2:38
6. Fashion is Danger	2:20
7. Carol Brown	3:26



Partition Problem

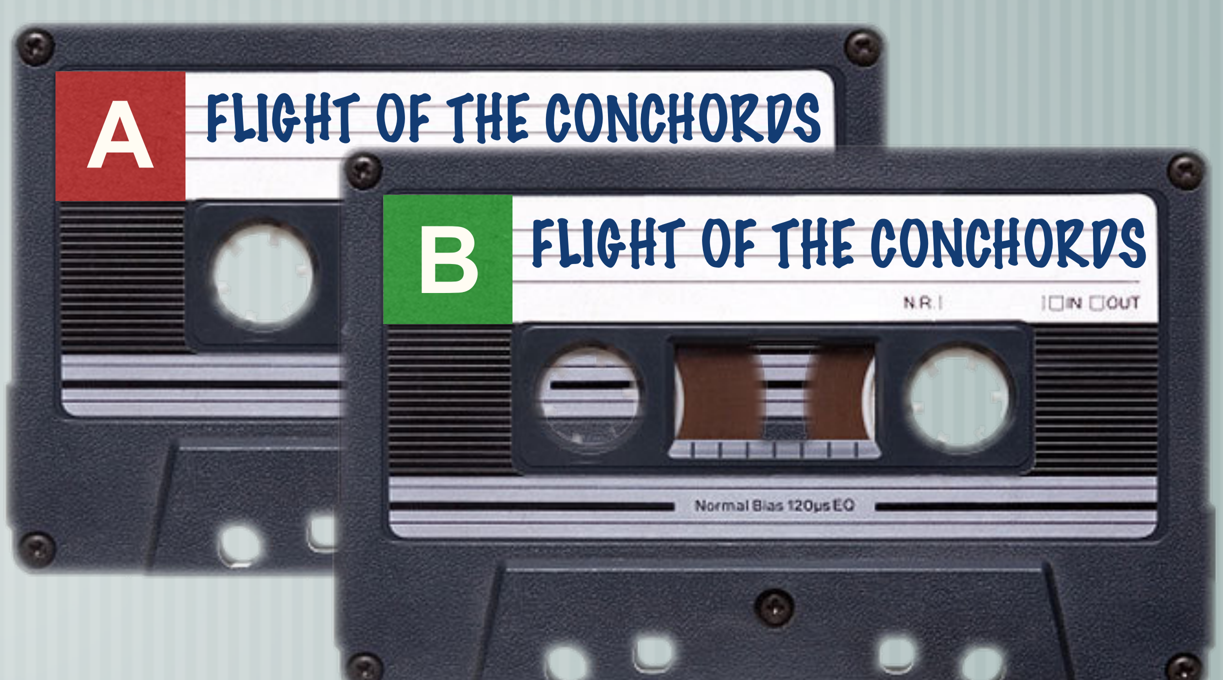
NP-Complete decision problem:

- There are n objects
- Object i has weight(i)

Can the objects be partitioned into two equal weight subsets?

1. Bowie	3:15
2. Robots	3:43
3. Business Time	4:05
4. Hiphopopotamus	2:09
5. Hurt Feelings	2:38
6. Fashion is Danger	2:20
7. Carol Brown	3:26

Exhaustive computation:



Partition Problem

NP-Complete decision problem:

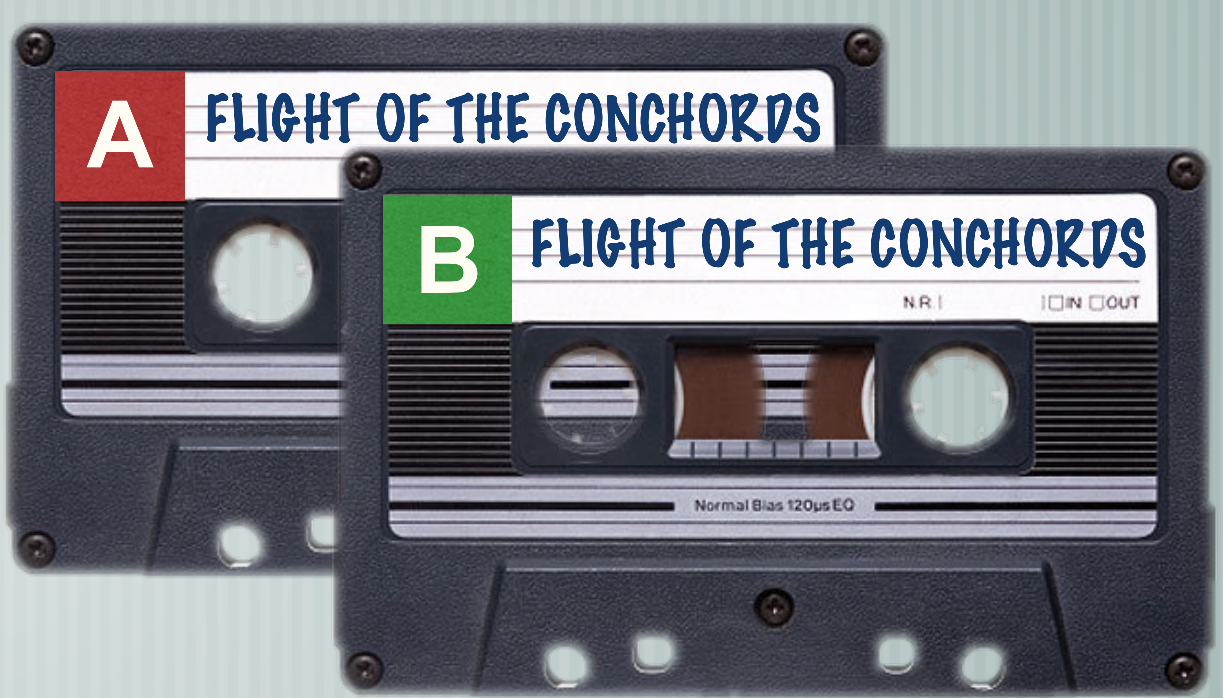
- There are n objects
- Object i has weight(i)

Can the objects be partitioned into two equal weight subsets?

Exhaustive computation:

- There are 2^n partitions

1. Bowie	3:15
2. Robots	3:43
3. Business Time	4:05
4. Hiphopopotamus	2:09
5. Hurt Feelings	2:38
6. Fashion is Danger	2:20
7. Carol Brown	3:26



Partition Problem

NP-Complete decision problem:

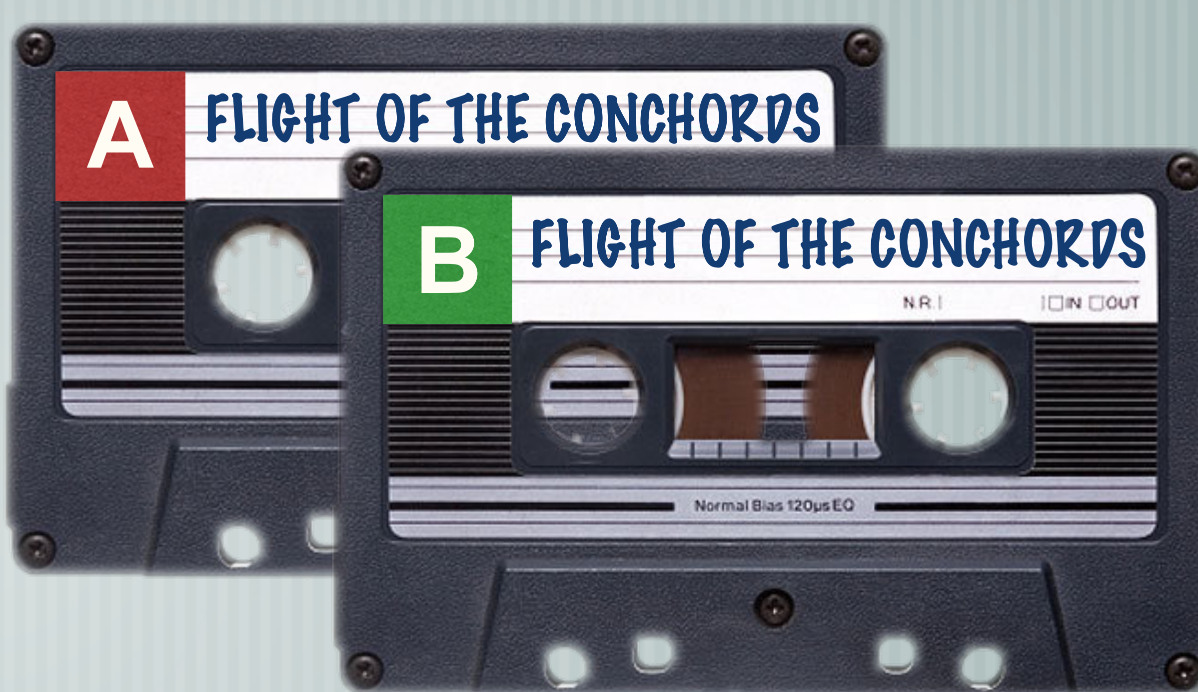
- There are n objects
- Object i has weight(i)

Can the objects be partitioned into two equal weight subsets?

Exhaustive computation:

- There are 2^n partitions
- O(1)-time with Gray code

1. Bowie	3:15
2. Robots	3:43
3. Business Time	4:05
4. Hiphopopotamus	2:09
5. Hurt Feelings	2:38
6. Fashion is Danger	2:20
7. Carol Brown	3:26



Partition Problem

NP-Complete decision problem:

- There are n objects
- Object i has weight(i)

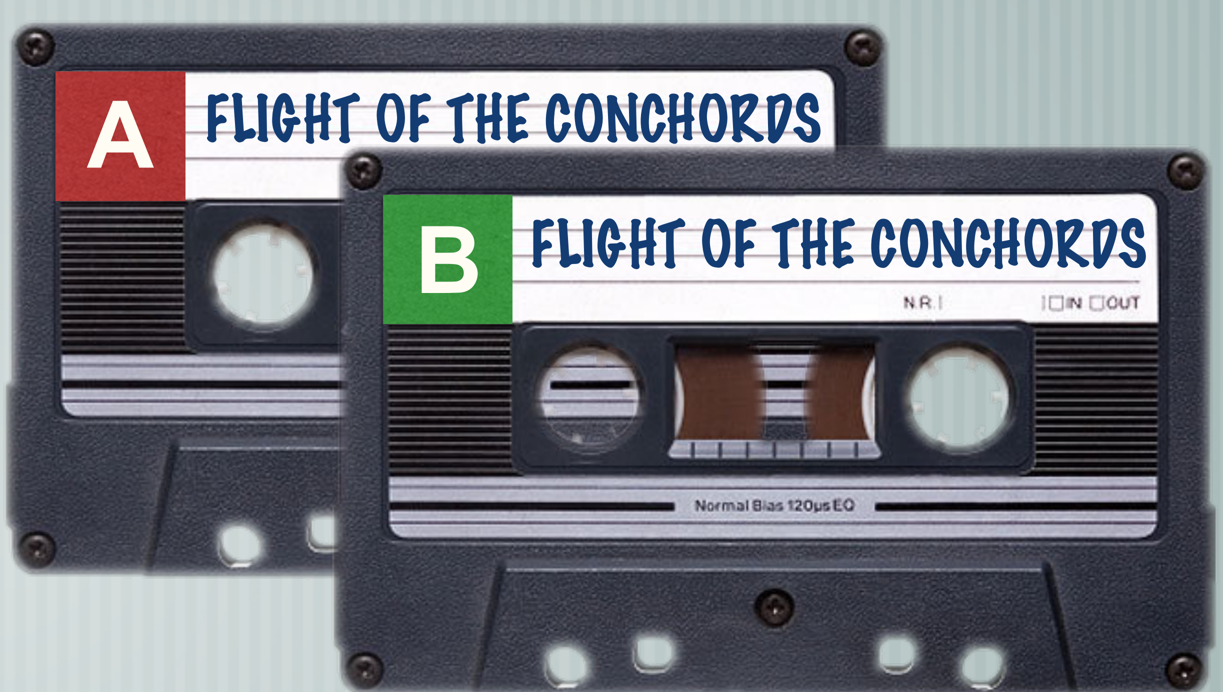
Can the objects be partitioned into two equal weight subsets?

Exhaustive computation:

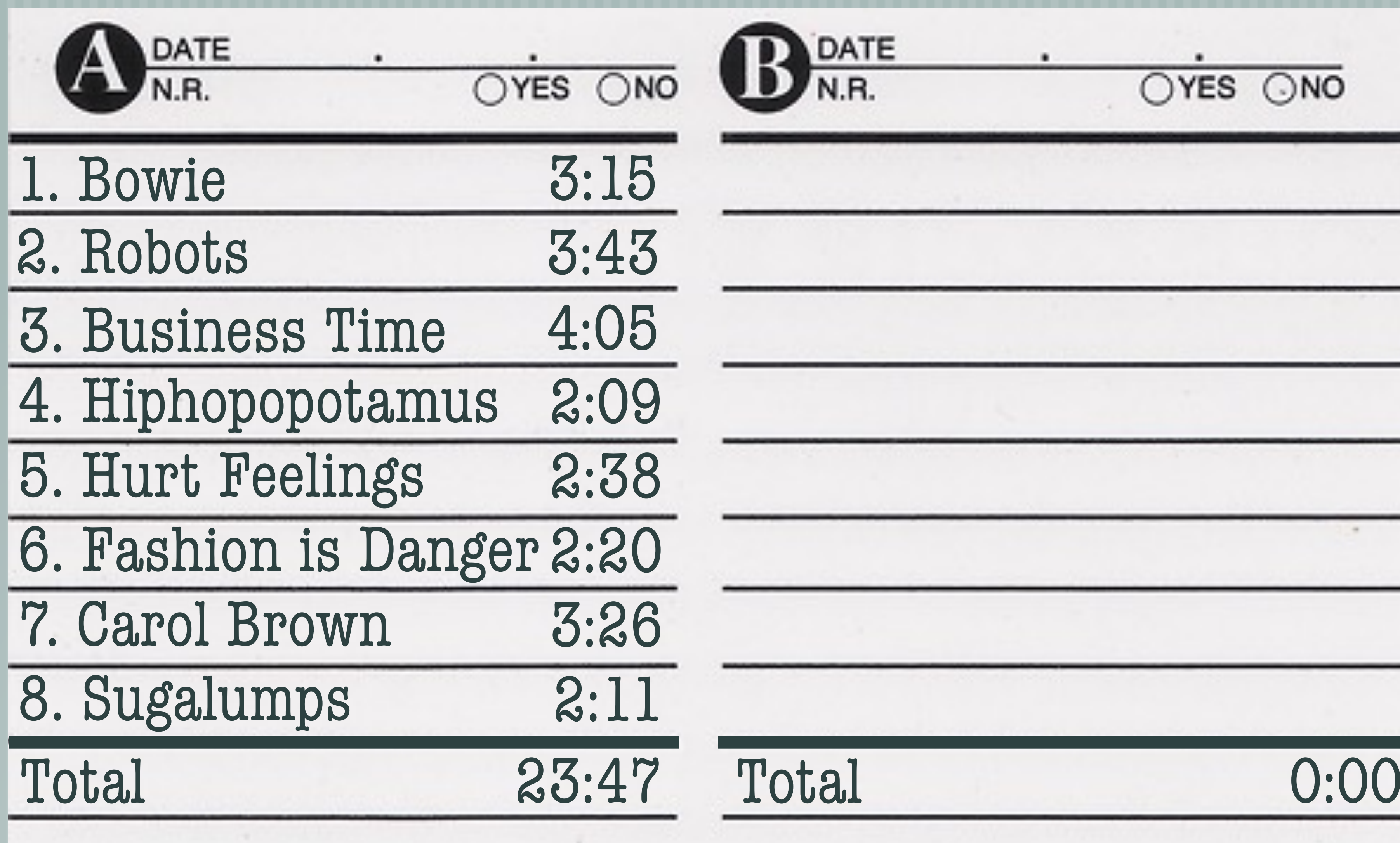
- There are 2^n partitions
- O(1)-time with Gray code

Total is $O(2^n)$-time instead of $O(n \cdot 2^n)$-time for naive algorithm.

1. Bowie	3:15
2. Robots	3:43
3. Business Time	4:05
4. Hiphopopotamus	2:09
5. Hurt Feelings	2:38
6. Fashion is Danger	2:20
7. Carol Brown	3:26



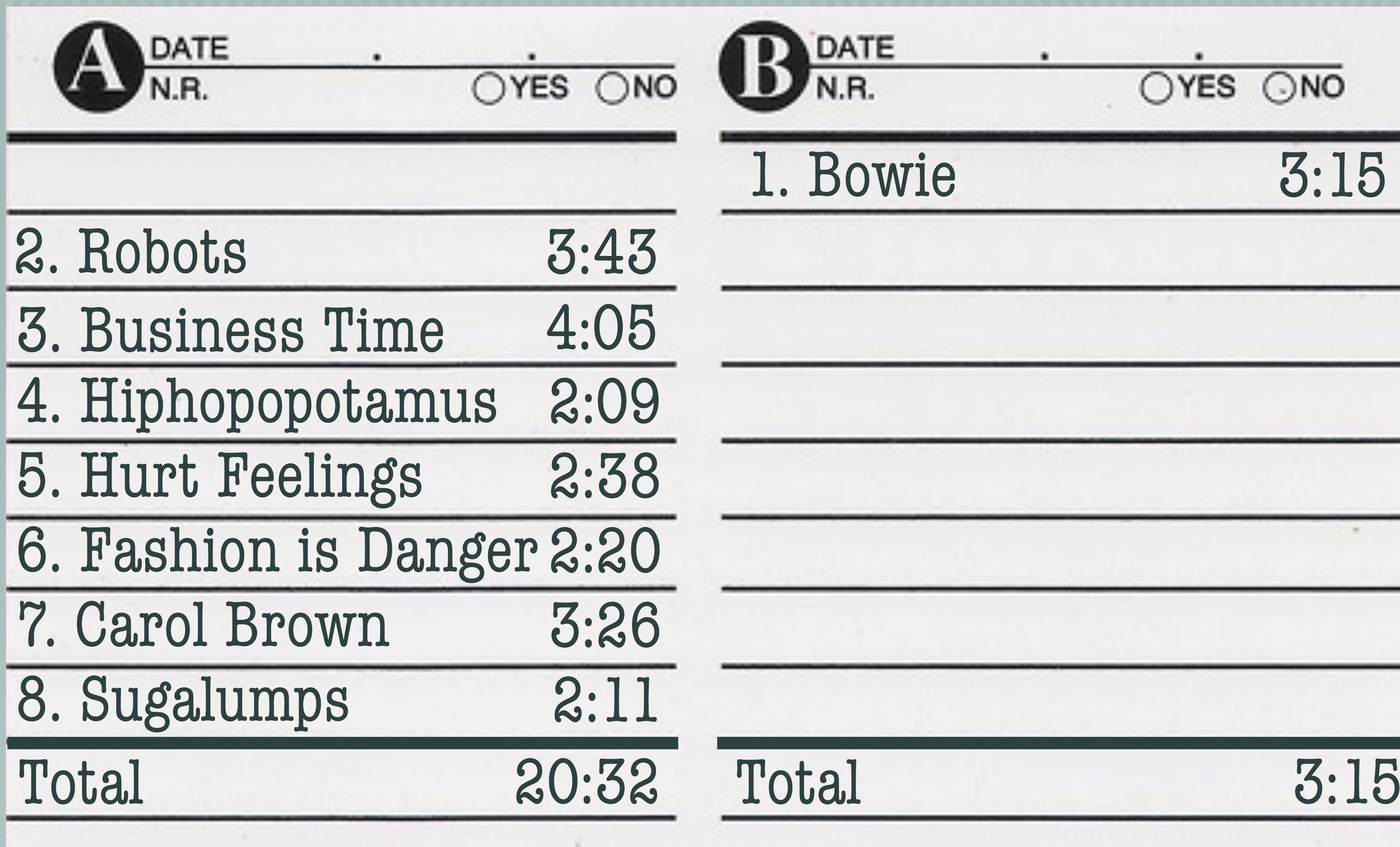
Example



A DATE . . N.R. ◯YES ◯NO		B DATE . . N.R. ◯YES ◯NO	
1. Bowie	3:15		
2. Robots	3:43		
3. Business Time	4:05		
4. Hiphopopotamus	2:09		
5. Hurt Feelings	2:38		
6. Fashion is Danger	2:20		
7. Carol Brown	3:26		
8. Sugalumps	2:11		
Total	23:47	Total	0:00

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

Example



A DATE . . N.R. ○YES ○NO		B DATE . . N.R. ○YES ○NO	
		1. Bowie	3:15
2. Robots	3:43		
3. Business Time	4:05		
4. Hiphopopotamus	2:09		
5. Hurt Feelings	2:38		
6. Fashion is Danger	2:20		
7. Carol Brown	3:26		
8. Sugalumps	2:11		
Total	20:32	Total	3:15

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

Example

A DATE . .
N.R. ◯YES ◯NO

3. Business Time	4:05
4. Hiphopopotamus	2:09
5. Hurt Feelings	2:38
6. Fashion is Danger	2:20
7. Carol Brown	3:26
8. Sugalumps	2:11
Total	13:49

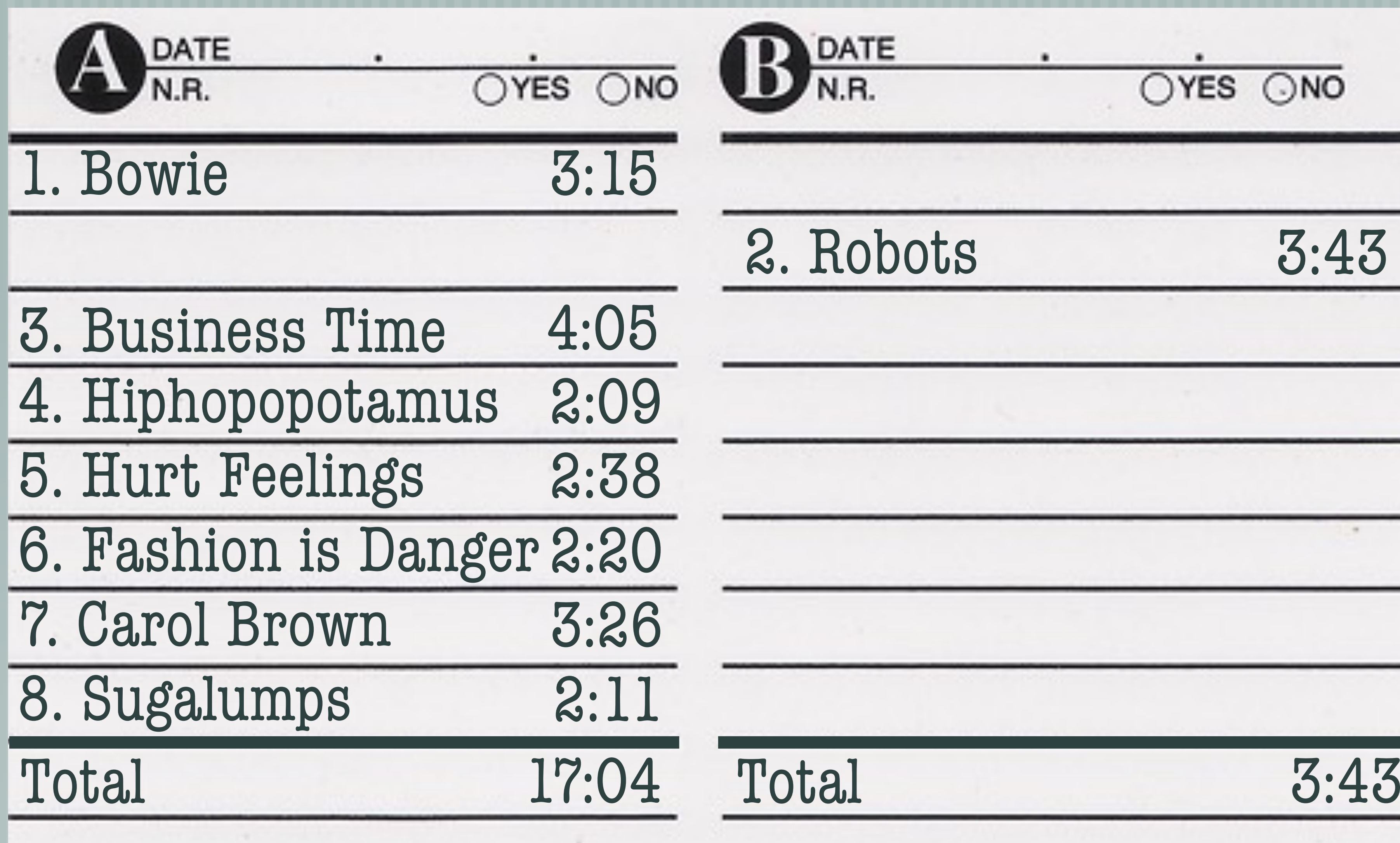
B DATE . .
N.R. ◯YES ◯NO

1. Bowie	3:15
2. Robots	3:43
Total	6:58

0 0 0 0 0 0 1 1

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

Example



A DATE N.R. ○YES ○NO		B DATE N.R. ○YES ○NO	
1. Bowie	3:15		
		2. Robots	3:43
3. Business Time	4:05		
4. Hiphopopotamus	2:09		
5. Hurt Feelings	2:38		
6. Fashion is Danger	2:20		
7. Carol Brown	3:26		
8. Sugalumps	2:11		
Total	17:04	Total	3:43

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

Example

A DATE . . N.R. ◯YES ◯NO

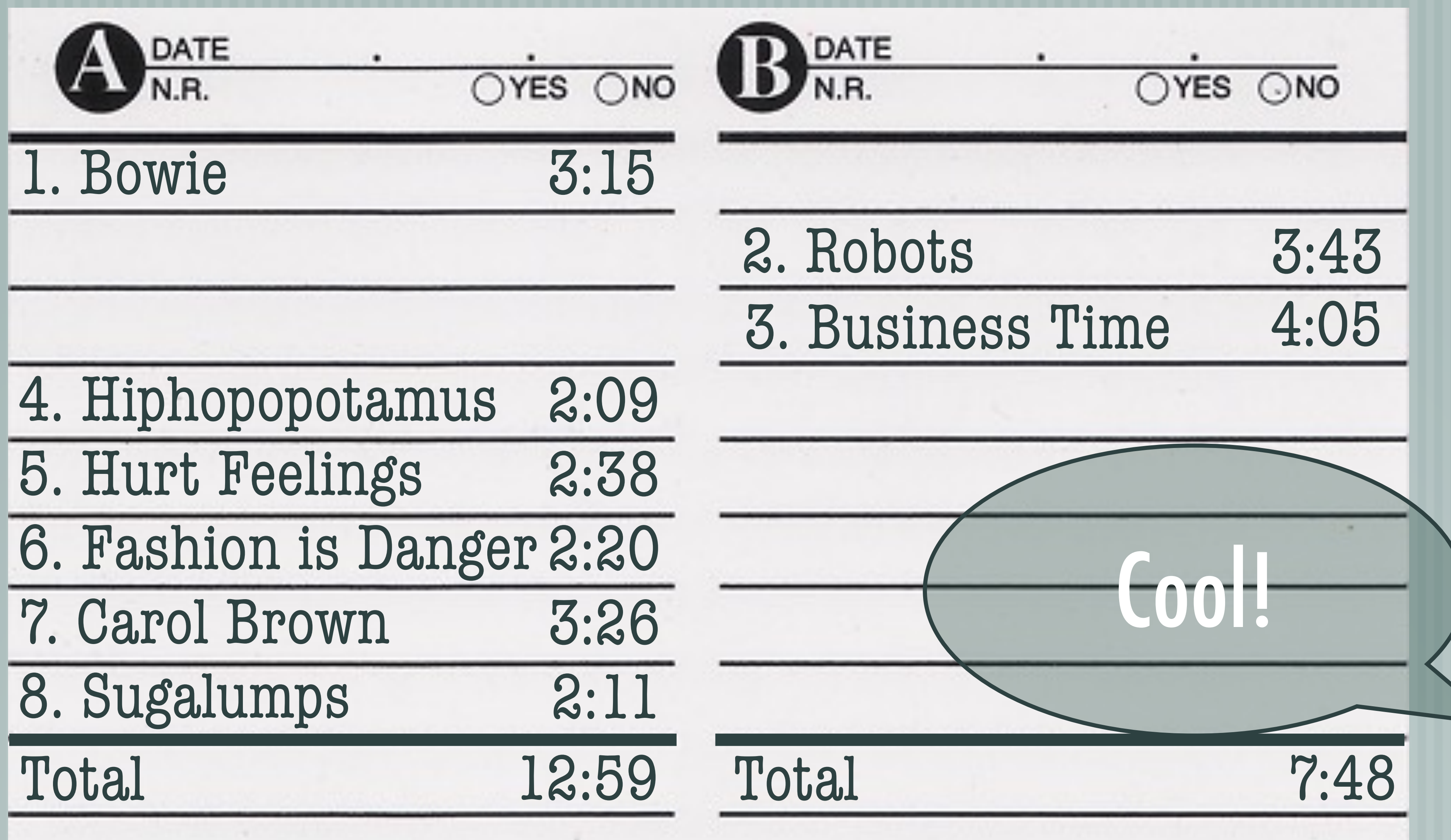
1. Bowie	3:15
4. Hiphopopotamus	2:09
5. Hurt Feelings	2:38
6. Fashion is Danger	2:20
7. Carol Brown	3:26
8. Sugalumps	2:11
Total	12:59

B DATE . . N.R. ◯YES ◯NO

2. Robots	3:43
3. Business Time	4:05
Total	7:48

0	0	0	0	0	1	1	0
	7.	6.	5.	4.	3.	2.	1.

Example


A DATE N.R. YES NO

1. Bowie	3:15
4. Hiphopopotamus	2:09
5. Hurt Feelings	2:38
6. Fashion is Danger	2:20
7. Carol Brown	3:26
8. Sugalumps	2:11
Total	12:59

B DATE N.R. YES NO

2. Robots	3:43
3. Business Time	4:05
Total	7:48

0 0 0 0 0 1 1 0

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

Position Encoder

Volume

Position Encoder

Volume

Position Encoder

Volume

Which volume is selected?

Position Encoder

Volume

Which volume is selected?

- Encode each position with a binary string

Position Encoder

Volume

Lexicographic order

0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

Which volume is selected?

- Encode each position with a binary string

Position Encoder

Volume

Lexicographic order

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Which volume is selected?

- Encode each position with a binary string
  - A unique n-bit string per volume level

Position Encoder

Volume

Lexicographic order

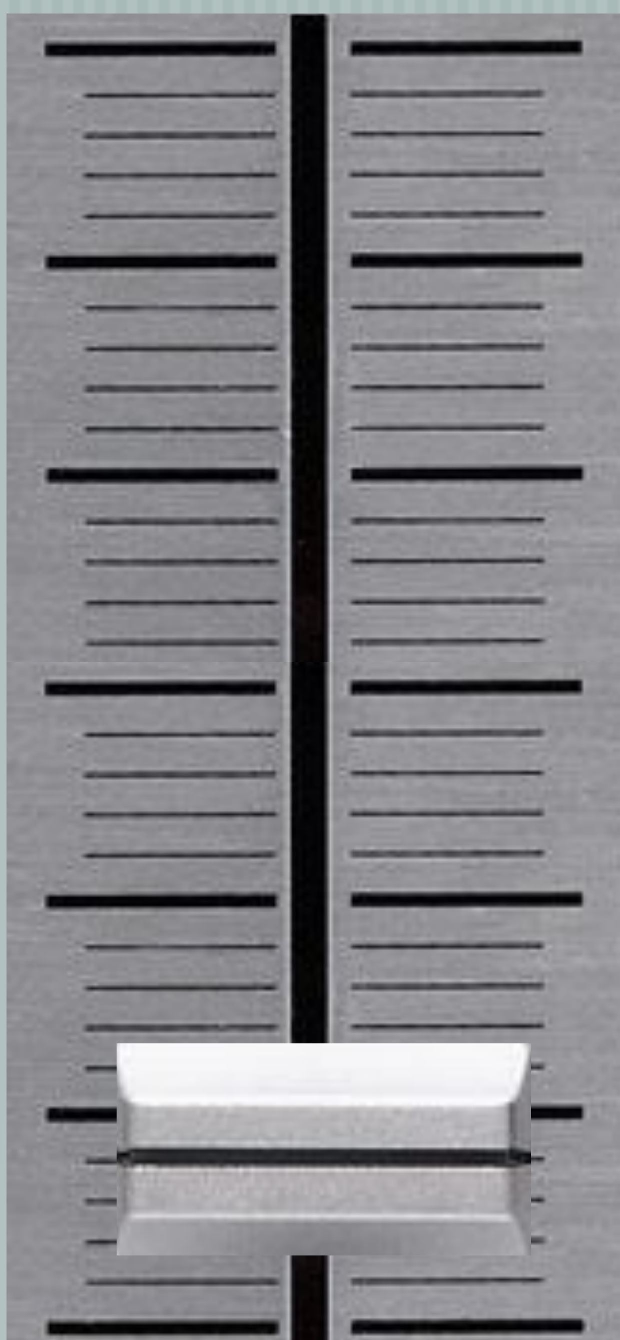
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Which volume is selected?

- Encode each position with a binary string
  - A unique n-bit string per volume level
  - Sensor behind the lever

Position Encoder

Volume



Lexicographic order

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Which volume is selected?

- Encode each position with a binary string
  - A unique n-bit string per volume level
  - Sensor behind the lever

Position Encoder

Volume

Lexicographic order

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Which volume is selected?

- Encode each position with a binary string
  - A unique n-bit string per volume level
  - Sensor behind the lever

Position Encoder

Volume

Lexicographic order

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Which volume is selected?

- Encode each position with a binary string
  - A unique n-bit string per volume level
  - Sensor behind the lever

Position Encoder

Volume

Lexicographic order

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Which volume is selected?

- Encode each position with a binary string
  - A unique n-bit string per volume level
  - Sensor behind the lever

What happens between positions?

Position Encoder

Volume

Lexicographic order

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

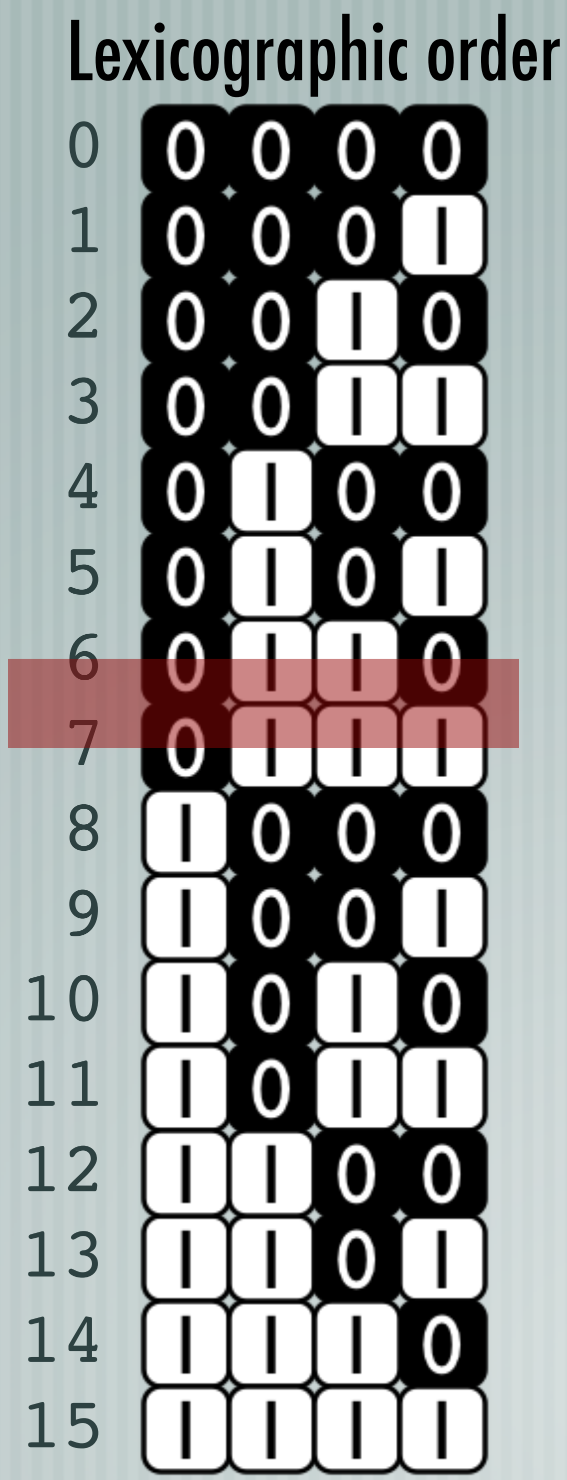
Which volume is selected?

- Encode each position with a binary string
  - A unique n-bit string per volume level
  - Sensor behind the lever

What happens between positions?

- Non-issue if there is one ambiguous bit

Position Encoder



Which volume is selected?

- Encode each position with a binary string
  - A unique n-bit string per volume level
  - Sensor behind the lever

What happens between positions?

- Non-issue if there is one ambiguous bit

Position Encoder

Volume

Lexicographic order

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Which volume is selected?

- Encode each position with a binary string
  - A unique n-bit string per volume level
  - Sensor behind the lever

What happens between positions?

- Non-issue if there is one ambiguous bit
- Problem if there are multiple ambiguous bits

Position Encoder

Lexicographic order

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Which volume is selected?

- Encode each position with a binary string
    - A unique n-bit string per volume level
    - Sensor behind the lever

What happens between positions?

- Non-issue if there is one ambiguous bit
- Problem if there are multiple ambiguous bits

Position Encoder

Volume

Lexicographic order

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Which volume is selected?

- Encode each position with a binary string
  - A unique n-bit string per volume level
  - Sensor behind the lever

What happens between positions?

- Non-issue if there is one ambiguous bit
- Problem if there are multiple ambiguous bits

Problem is solved by using a Gray code

Position Encoder

Volume

Lexicographic order

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Which volume is selected?

- Encode each position with a binary string
  - A unique n-bit string per volume level
  - Sensor behind the lever

What happens between positions?

- Non-issue if there is one ambiguous bit
- Problem if there are multiple ambiguous bits

Problem is solved by using a Gray code

Position Encoder

Volume

Gray code

0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1
14	1	0	0	1
15	1	0	0	0

Which volume is selected?

- Encode each position with a binary string
  - A unique n-bit string per volume level
  - Sensor behind the lever

What happens between positions?

- Non-issue if there is one ambiguous bit
- Problem if there are multiple ambiguous bits

Problem is solved by using a Gray code

Position Encoder

Volume

Gray code

0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1
14	1	0	0	1
15	1	0	0	0

Which volume is selected?

- Encode each position with a binary string
    - A unique n-bit string per volume level
    - Sensor behind the lever

What happens between positions?

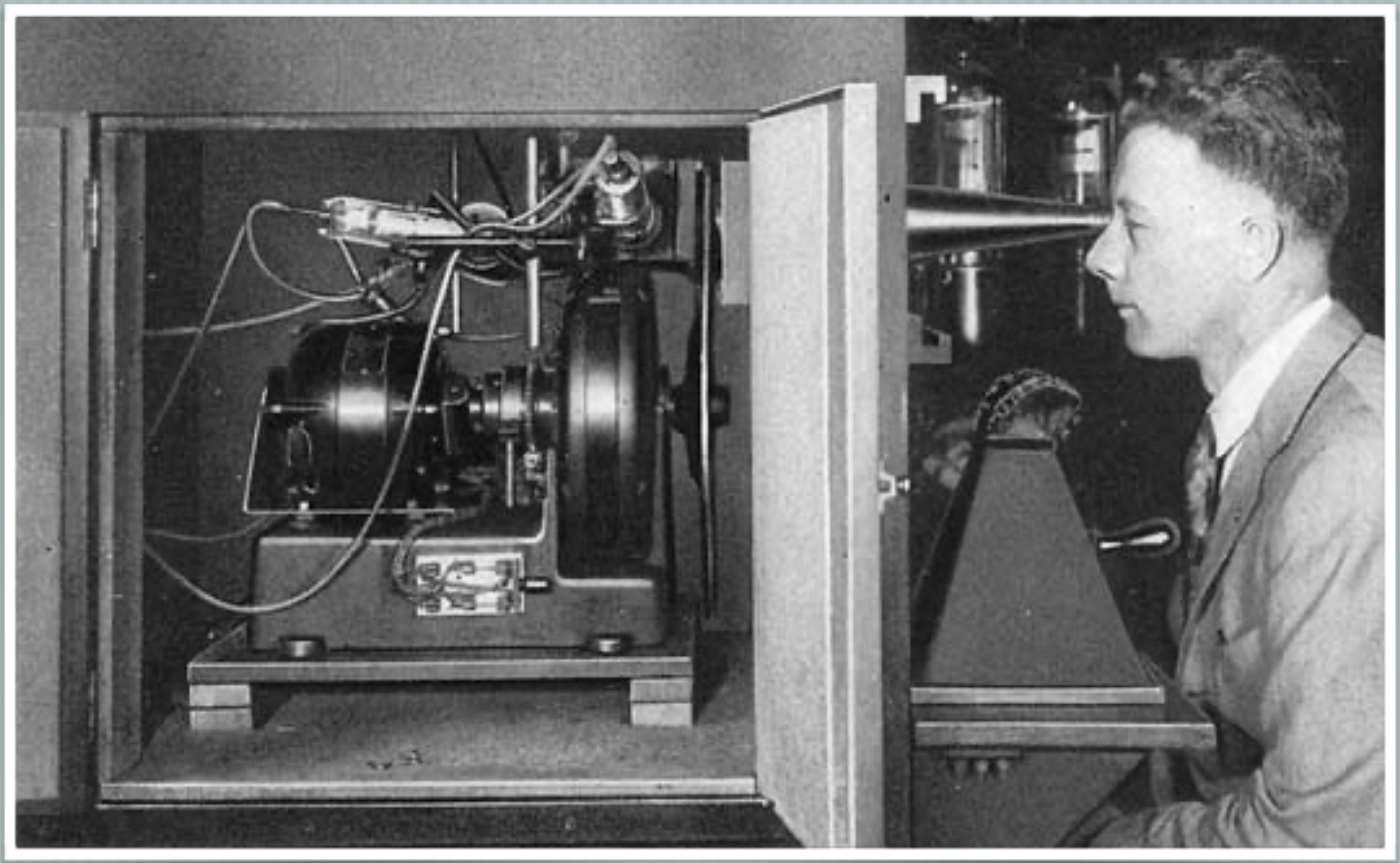
- Non-issue if there is one ambiguous bit
- Problem if there are multiple ambiguous bits

Problem is solved by using a Gray code

- Requires ranking/unranking

U.S. Patent on Pulse Communication

The term "Gray code" comes from this type of application.



Frank Gray at Bell Labs circa 1950

Position Encoder

Rotary encoder

From Wikipedia, the free encyclopedia

A **rotary encoder**, also called a **shaft encoder**, is an electro-mechanical device that converts the angular position or motion of a shaft or axle to an analog or digital code.

There are two main types: absolute and incremental (relative). The output of absolute encoders indicates the current position of the shaft, making them angle transducers. The output of incremental encoders provides information about the *motion* of the shaft, which is typically further processed elsewhere into information such as speed, distance, RPM and position.

Rotary encoders are used in many applications that require precise shaft unlimited rotation—including industrial controls, robotics, special purpose photographic lenses,[1] computer input devices (such as optomechanical mice and trackballs), controlled stress rheometers, and rotating radar platforms.

Contents [hide]

Gray code pattern
n = 13

graycodes.com

- Home
- Permutations
- Multiperms
- Combinations
- Bitstrings
- Middle levels
- Partitions
- Dyck paths
- Plane trees
- Spanning trees
- De Bruijn
- Semigroups
- References

Generate permutations of a multiset

Generate all permutations of a multiset. Provide the multiset by specifying its frequency sequence, e.g. "1 1 3 2" for the multiset $\{0, 1, 2, 2, 2, 3, 3\}$. Output limit is 10.000 objects.

Frequency sequence: 1 1 3 2 (max. 20)

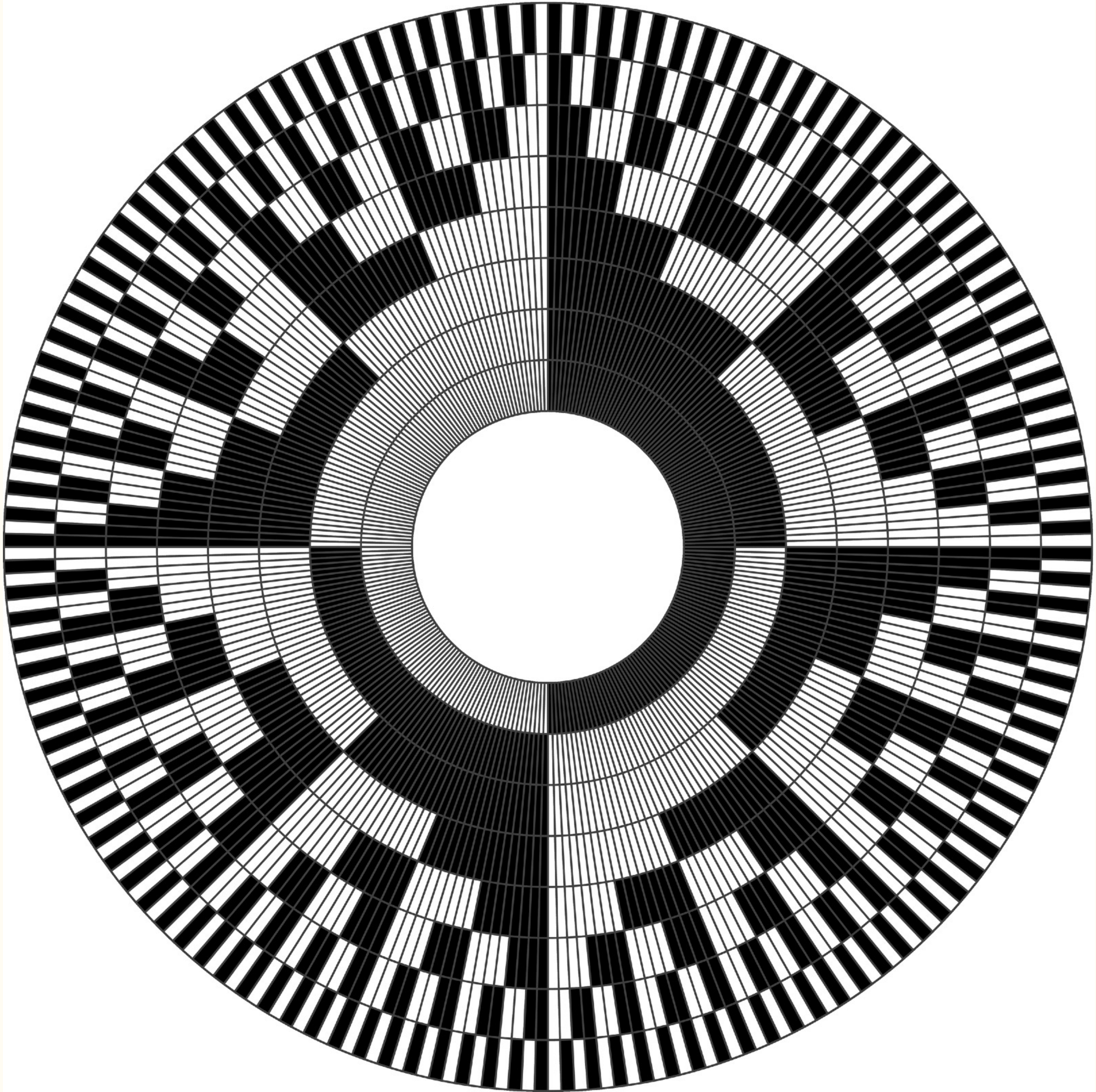
Algorithm: Prefix rotation (Williams' cool-lex)

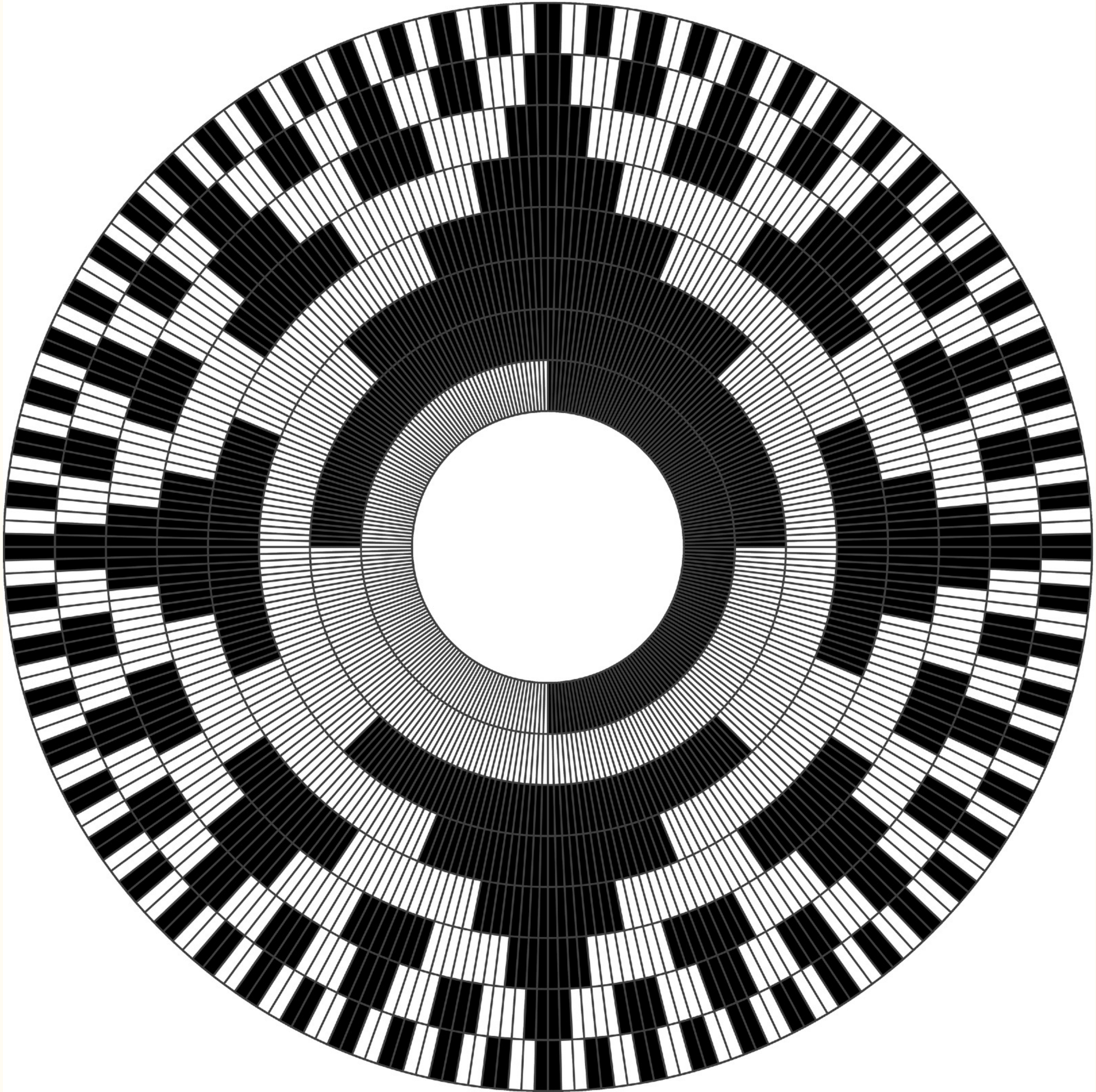
Run

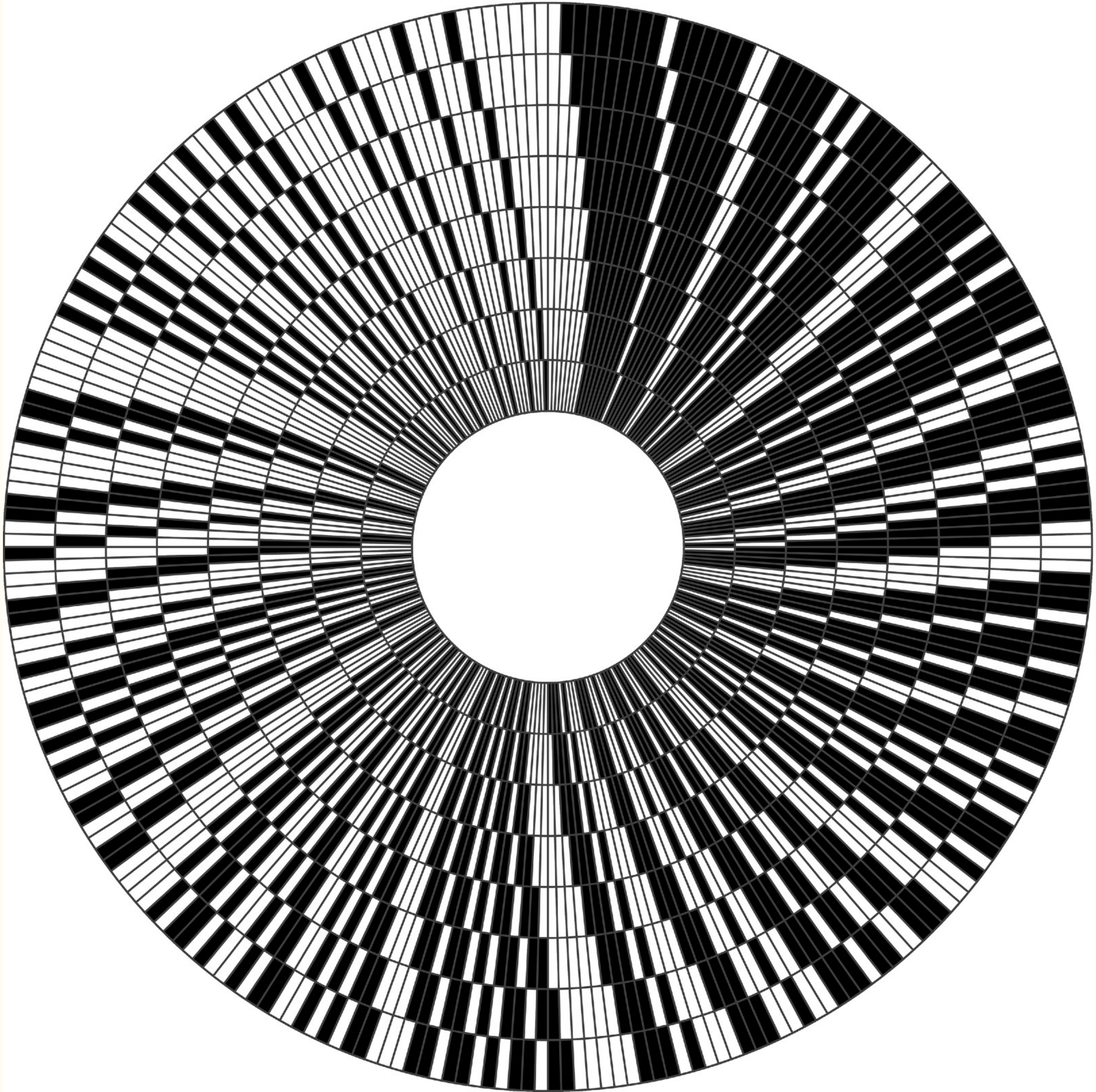
The first three algorithms running on this page are part of Jörg Arndt's FXT library. The iterative algorithm for lexicographic ordering is explained in Jörg Arndt's FXT library [Section 13.2.2, Arn10]. The minimal-change algorithm follows a Steinhaus-Johnson-Trotter adjacent transposition

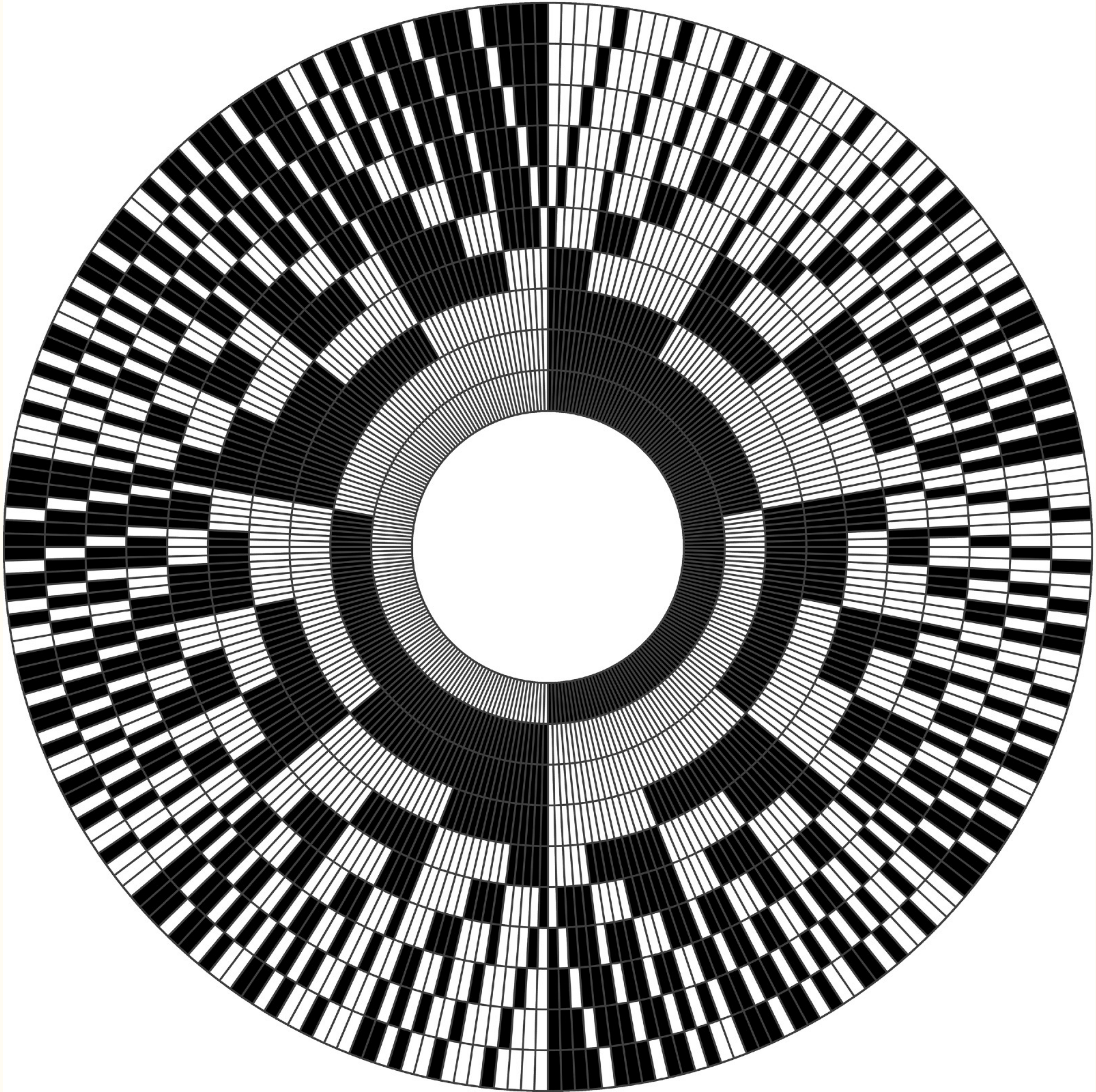
Results

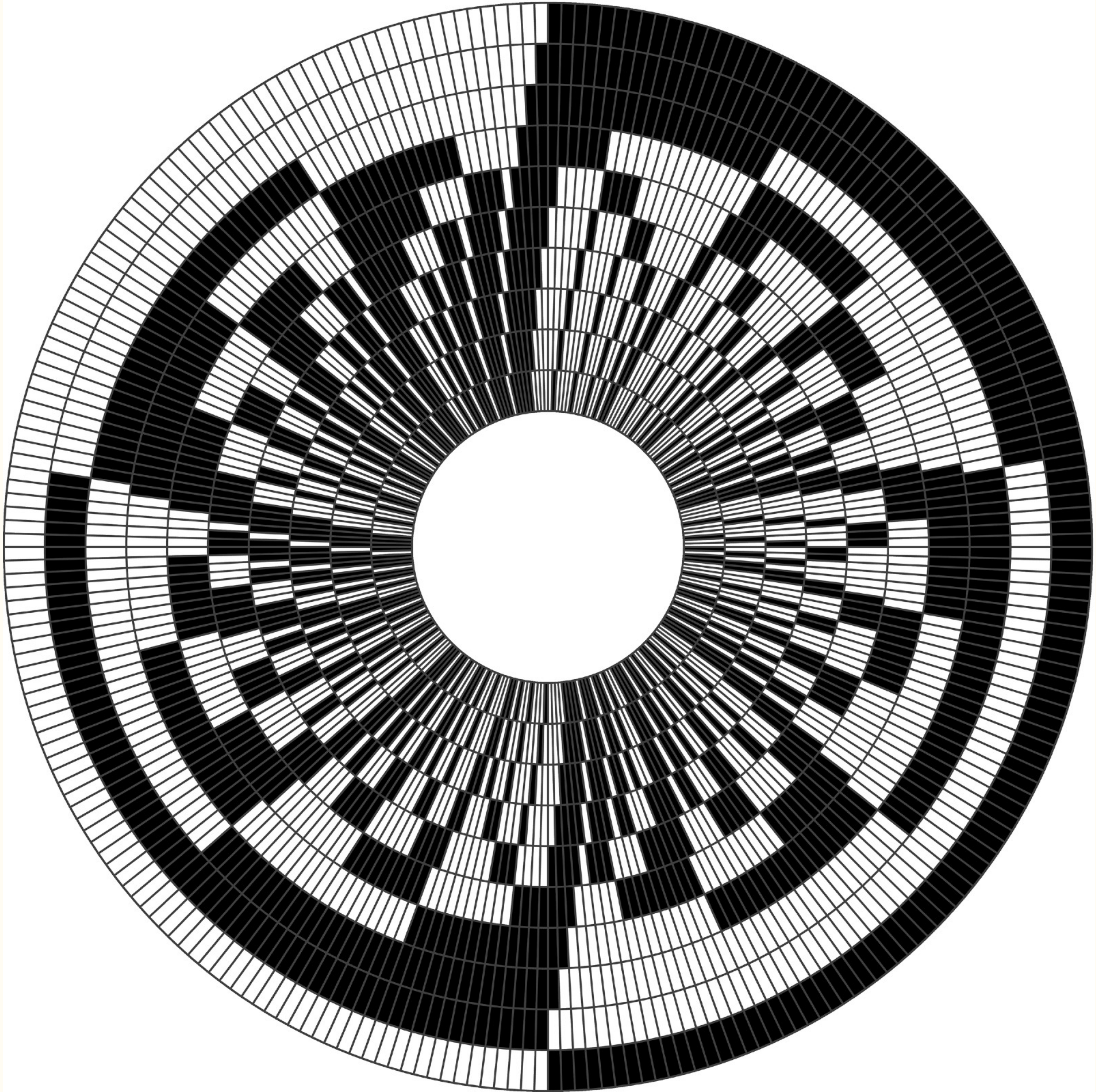
```
1: [ 0 3 3 2 2 2 1 ]
2: [ 3 0 3 2 2 2 1 ]
3: [ 3 3 0 2 2 2 1 ]
4: [ 2 3 3 0 2 2 1 ]
5: [ 3 2 3 0 2 2 1 ]
6: [ 0 3 2 3 2 2 1 ]
7: [ 3 0 2 3 2 2 1 ]
8: [ 2 3 0 3 2 2 1 ]
9: [ 0 2 3 3 2 2 1 ]
10: [ 2 0 3 3 2 2 1 ]
11: [ 3 2 0 3 2 2 1 ]
12: [ 3 3 2 0 2 2 1 ]
13: [ 2 3 3 2 0 2 1 ]
14: [ 3 2 3 2 0 2 1 ]
15: [ 2 3 2 3 0 2 1 ]
16: [ 2 2 3 3 0 2 1 ]
17: [ 3 2 2 3 0 2 1 ]
18: [ 0 3 2 2 3 2 1 ]
19: [ 3 0 2 2 3 2 1 ]
20: [ 2 3 0 2 3 2 1 ]
21: [ 0 2 3 2 3 2 1 ]
22: [ 2 0 3 2 3 2 1 ]
23: [ 3 2 0 2 3 2 1 ]
24: [ 2 3 2 0 3 2 1 ]
25: [ 2 2 3 0 3 2 1 ]
26: [ 0 2 2 3 3 2 1 ]
27: [ 2 0 2 3 3 2 1 ]
28: [ 2 2 0 3 3 2 1 ]
29: [ 3 2 2 0 3 2 1 ]
```

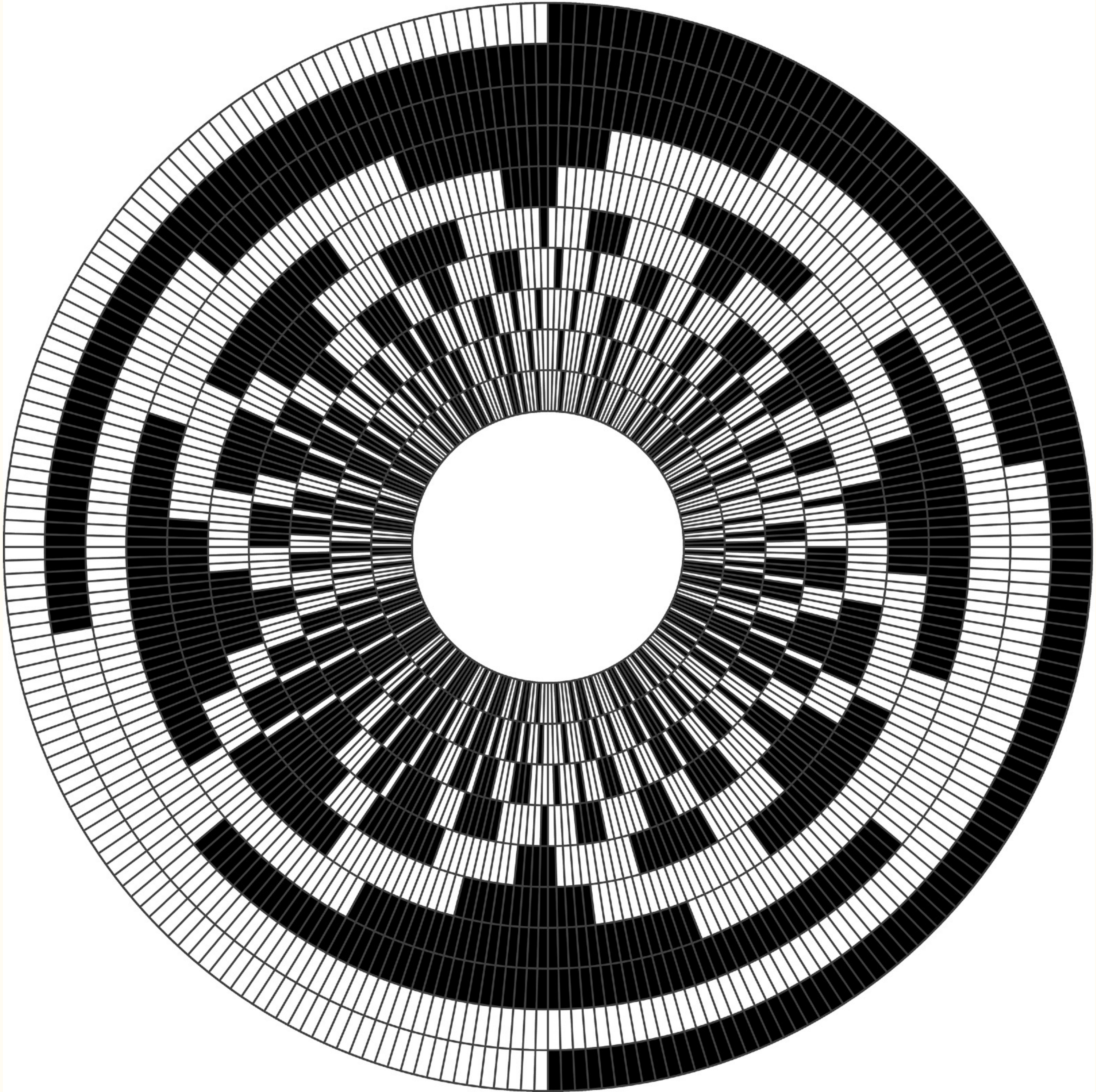


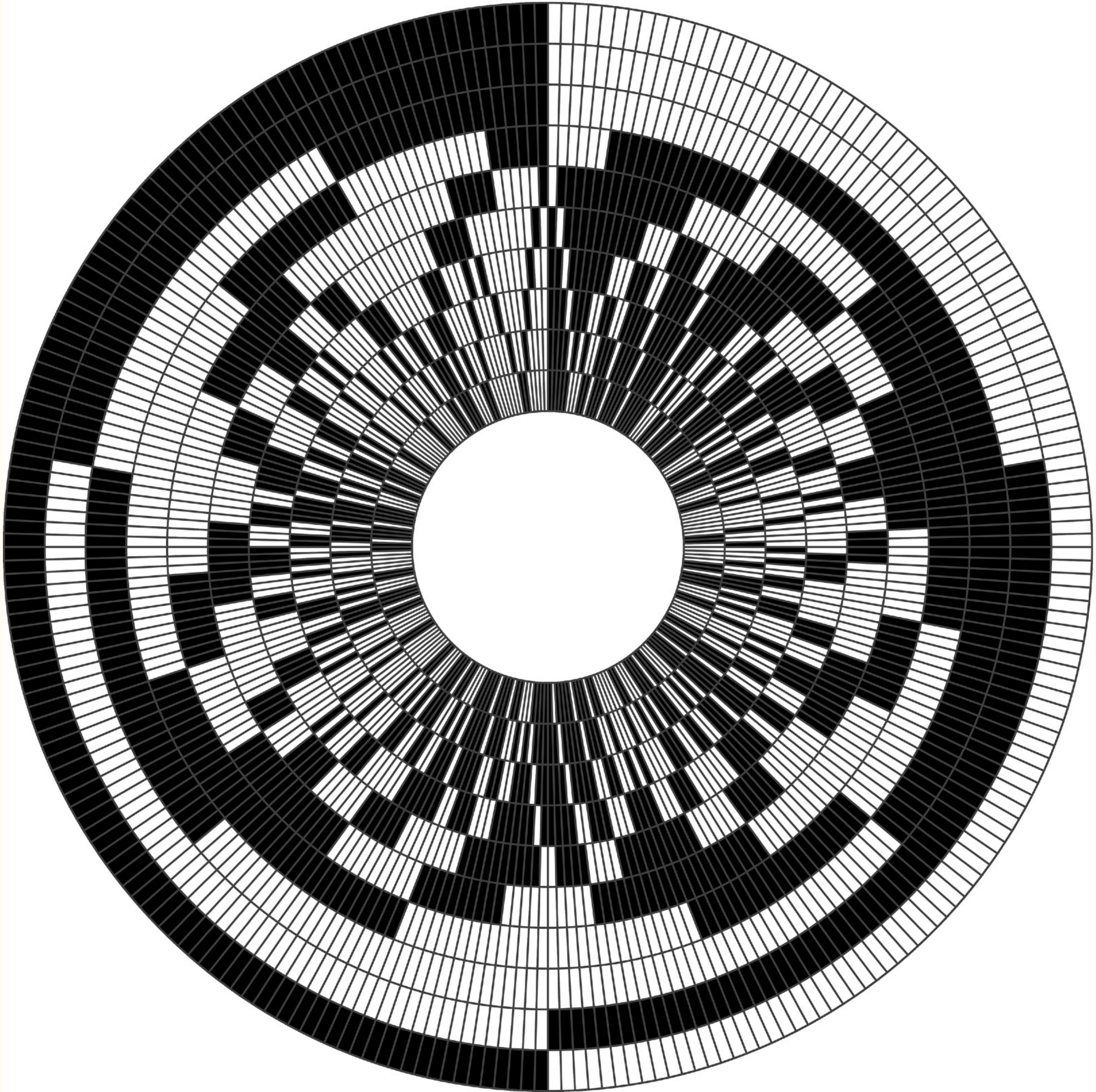


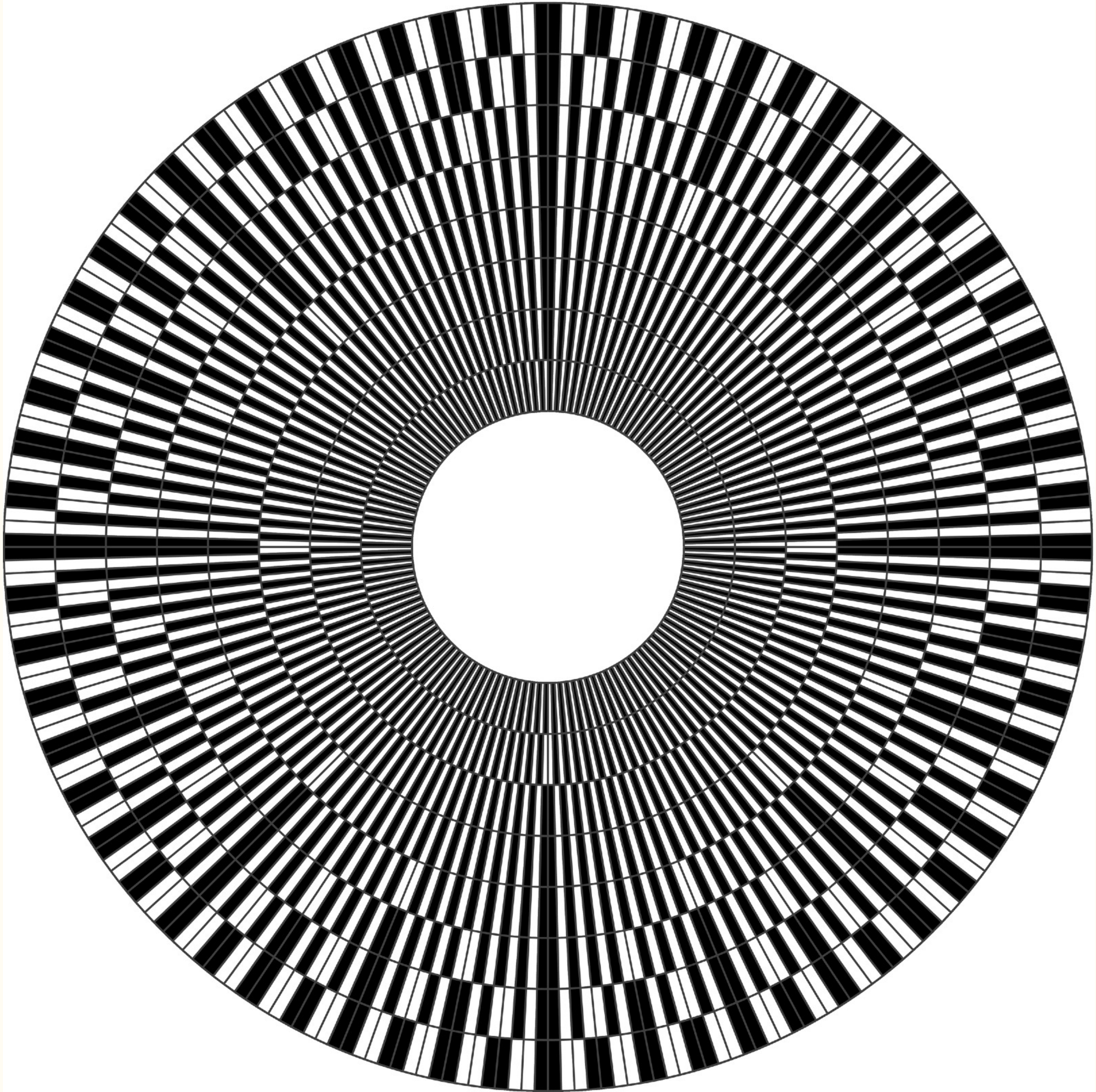


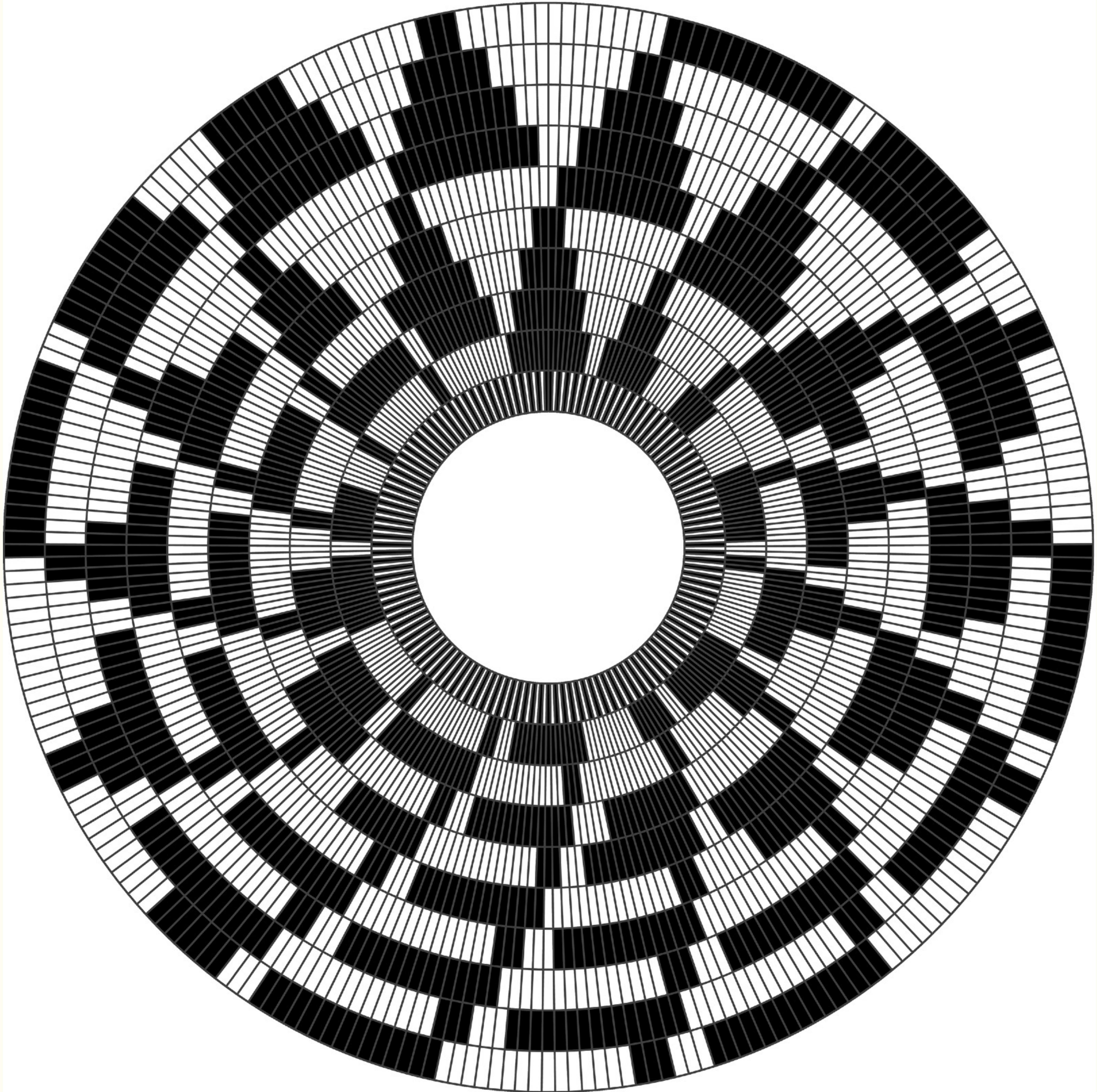


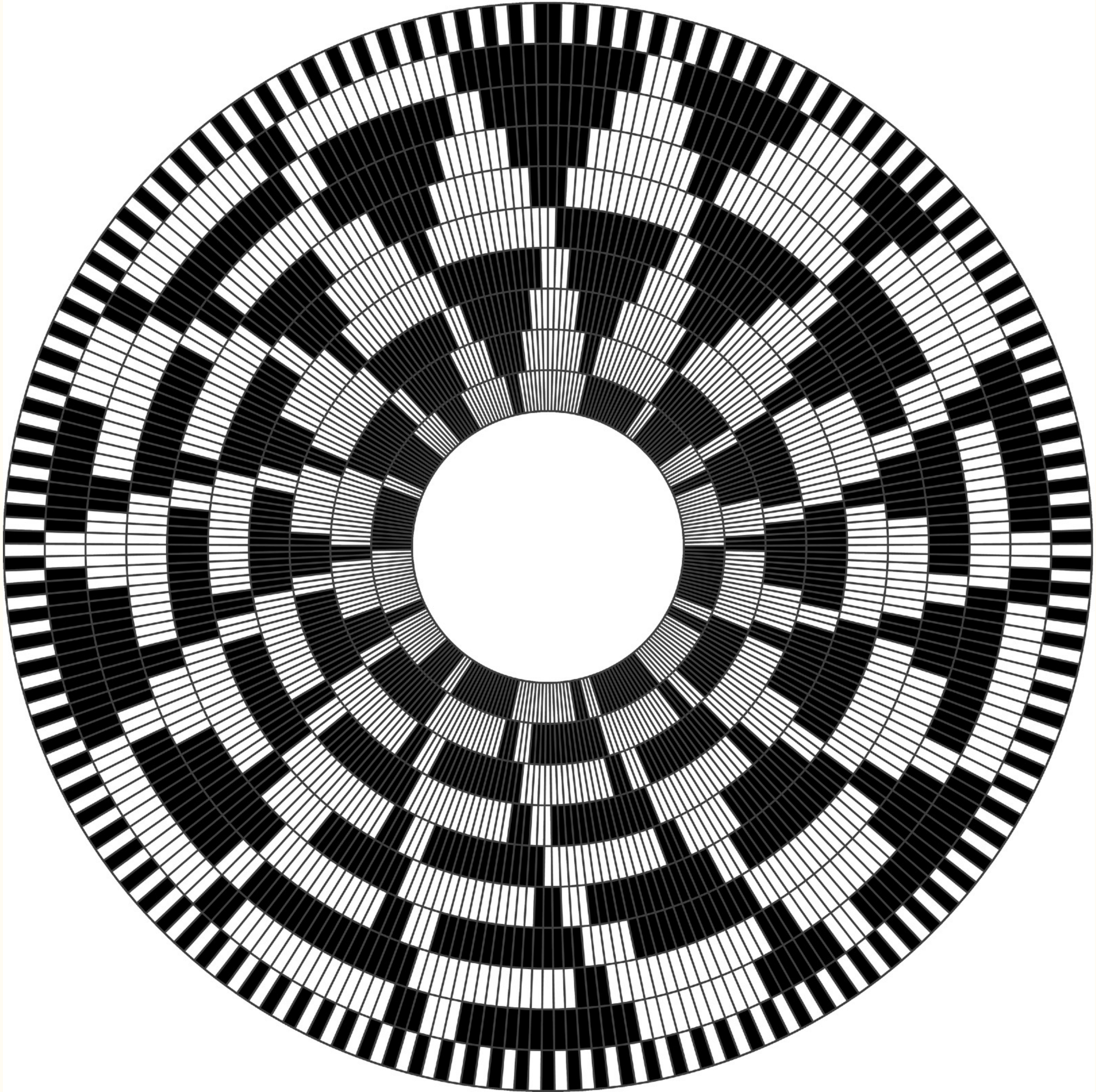


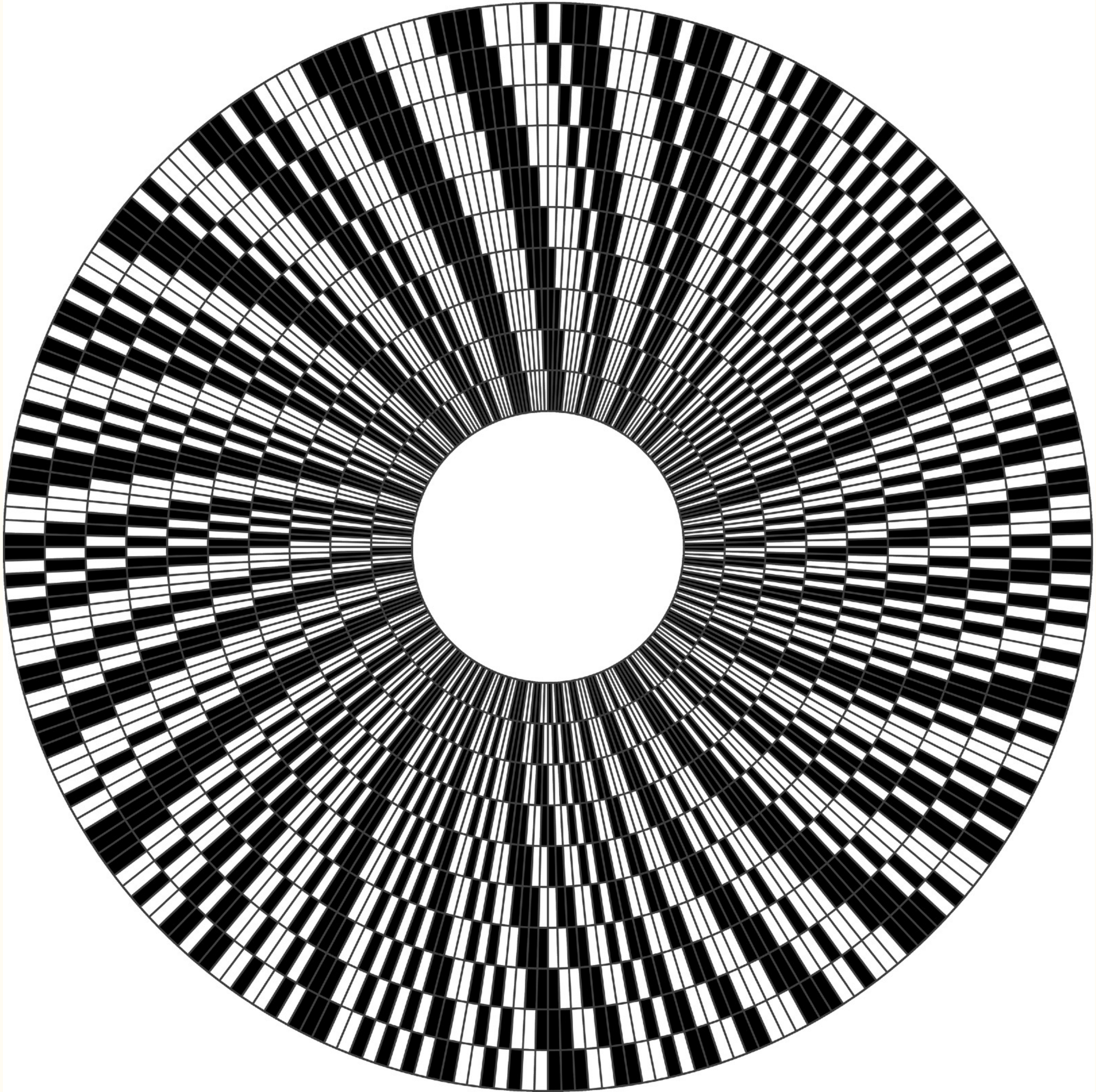


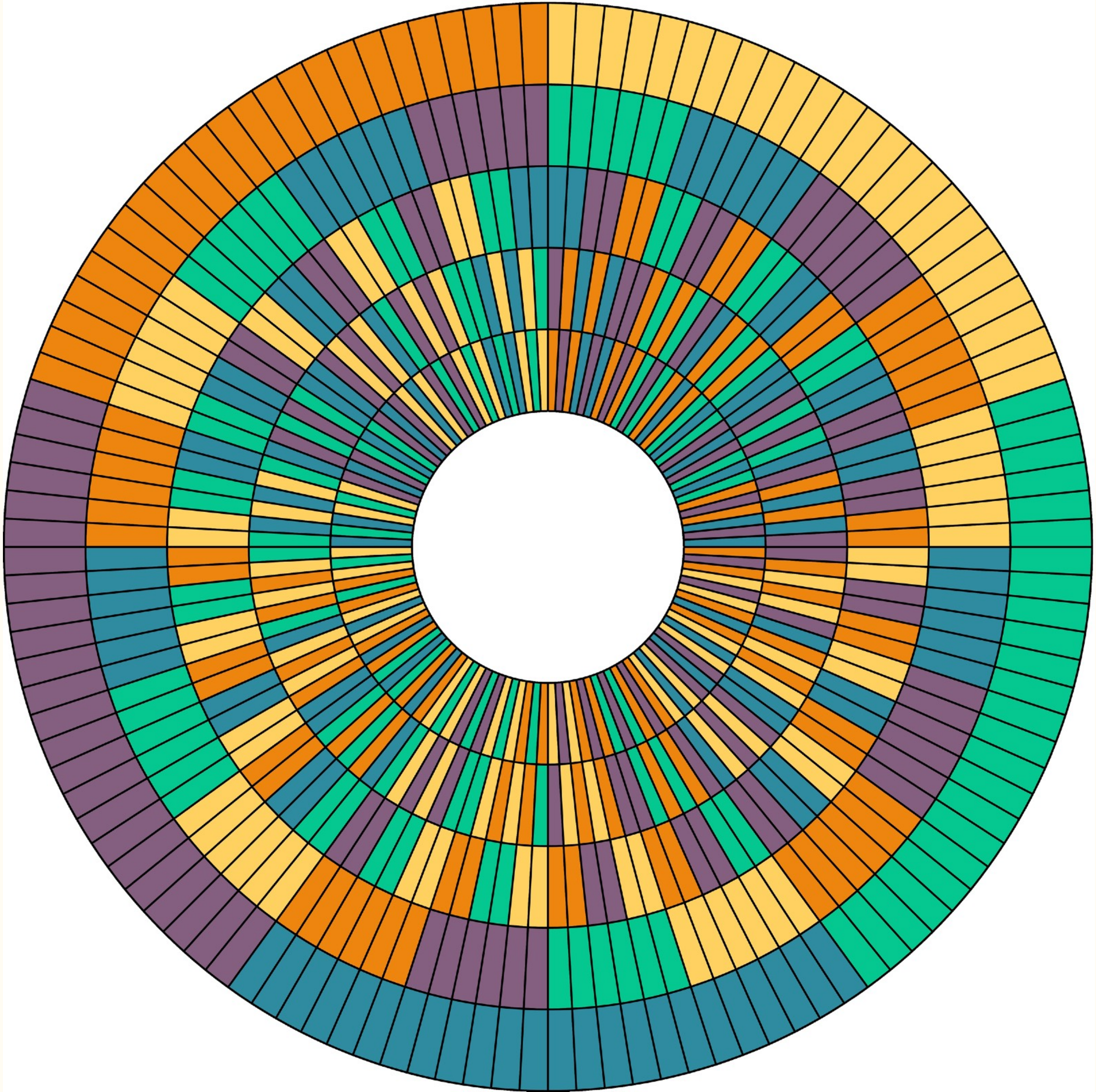


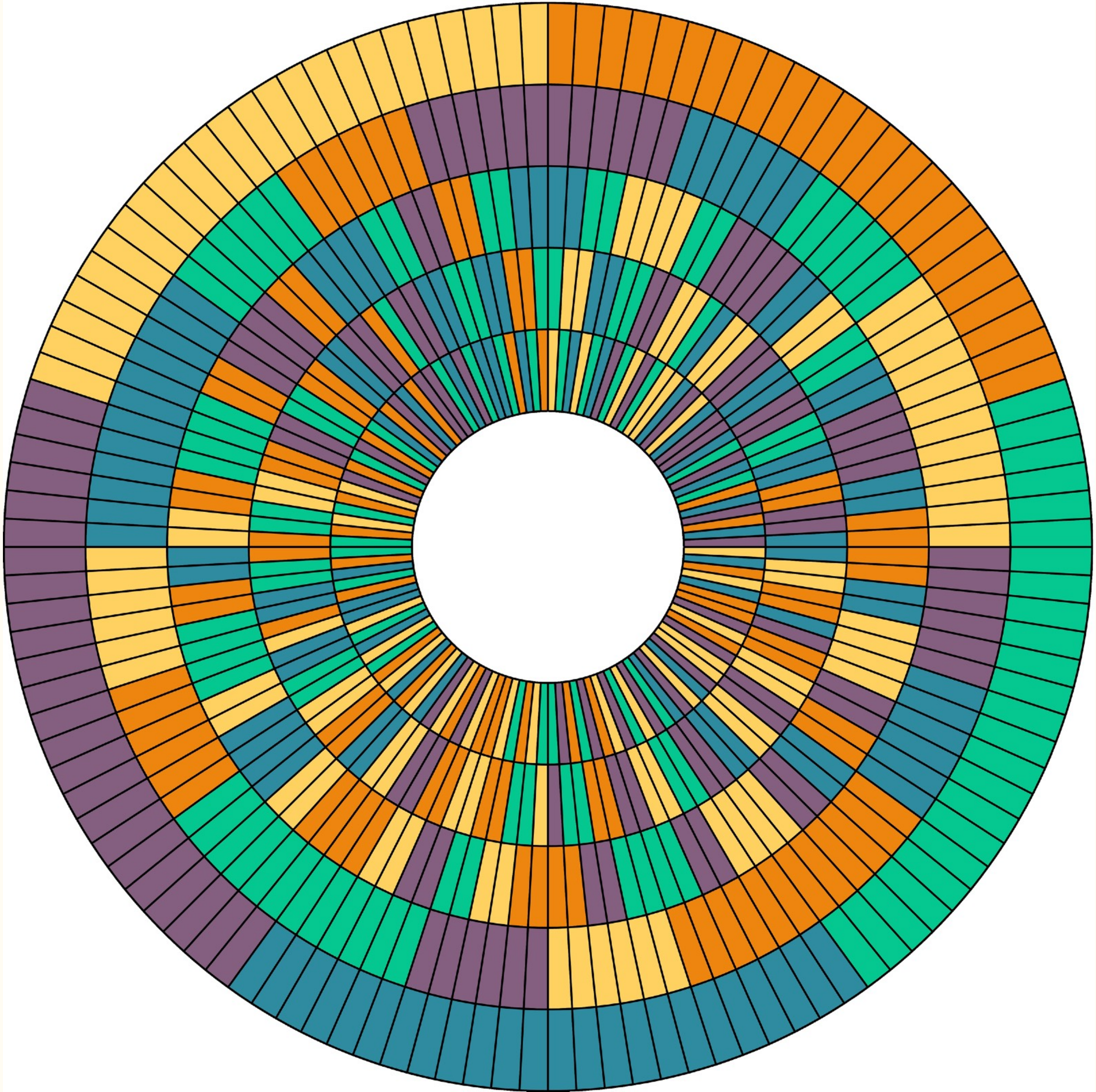


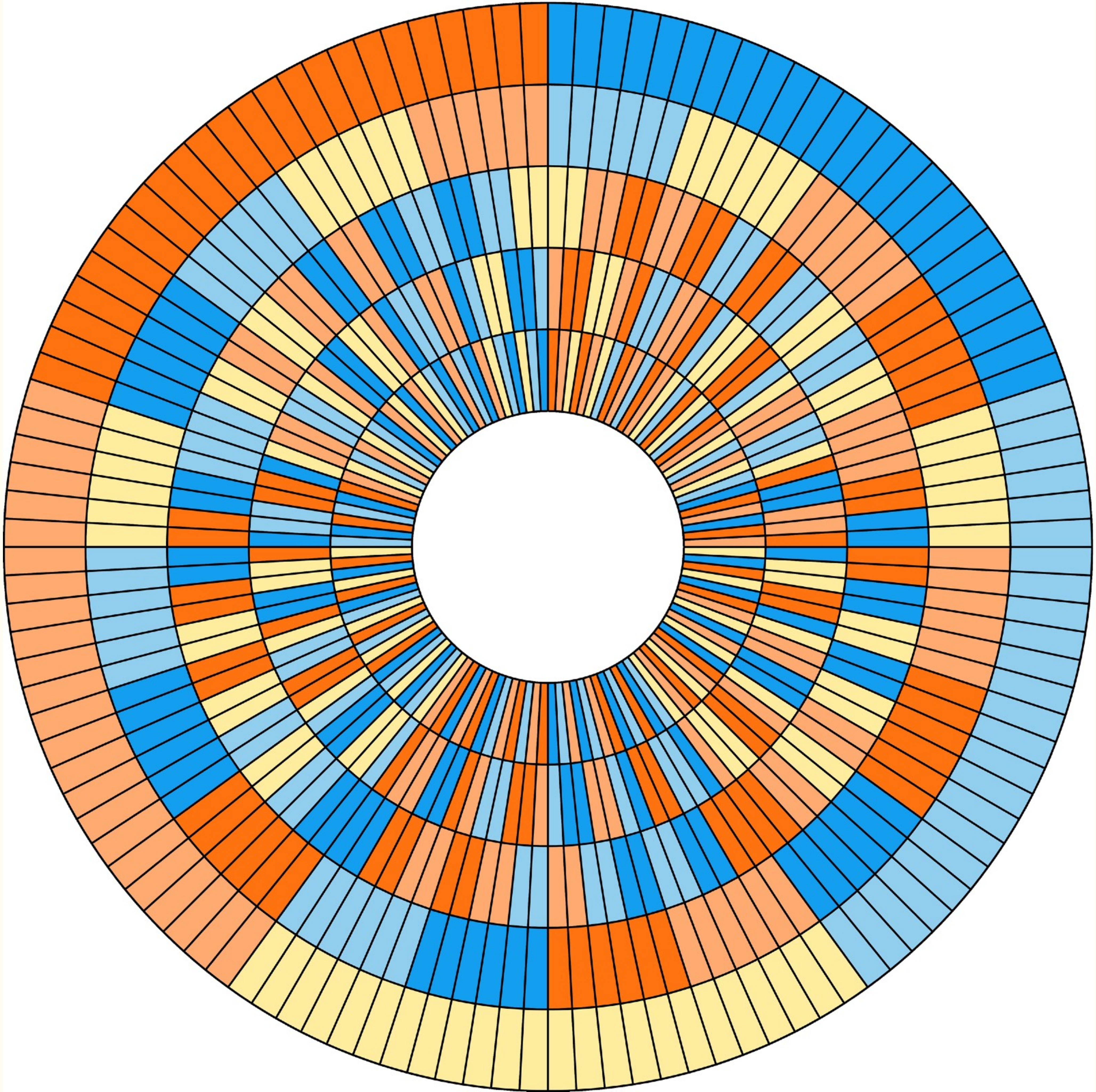


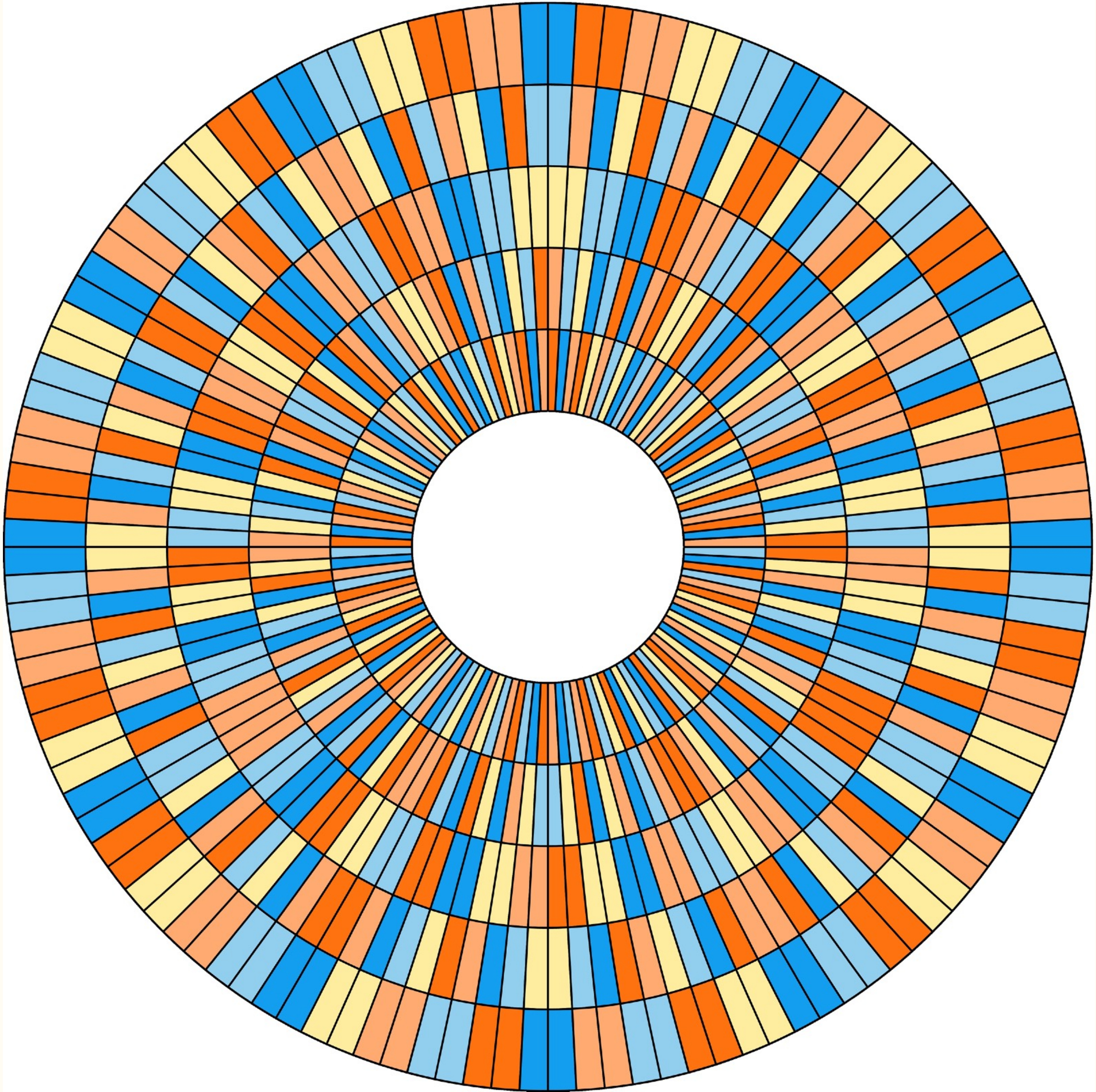


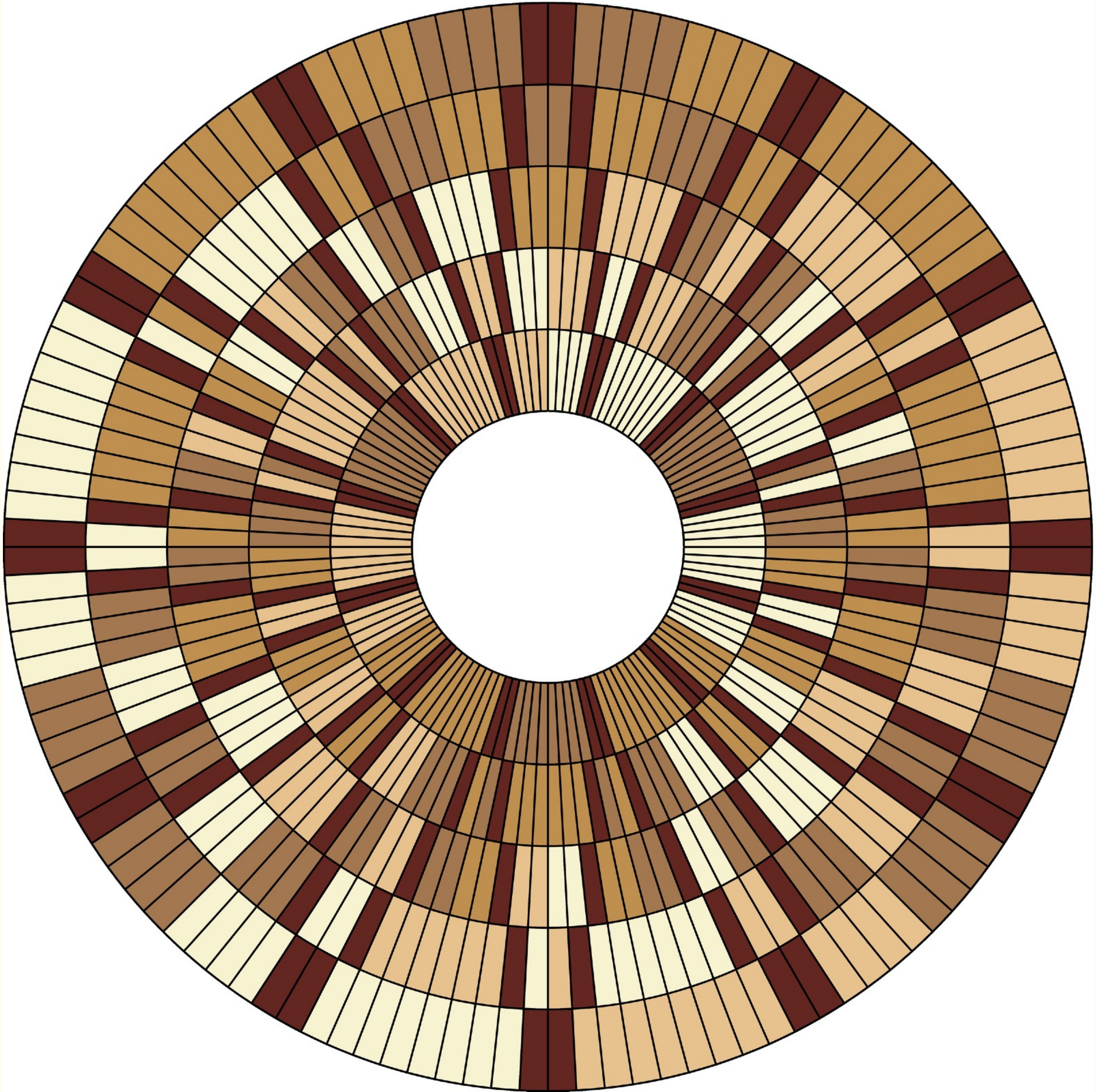


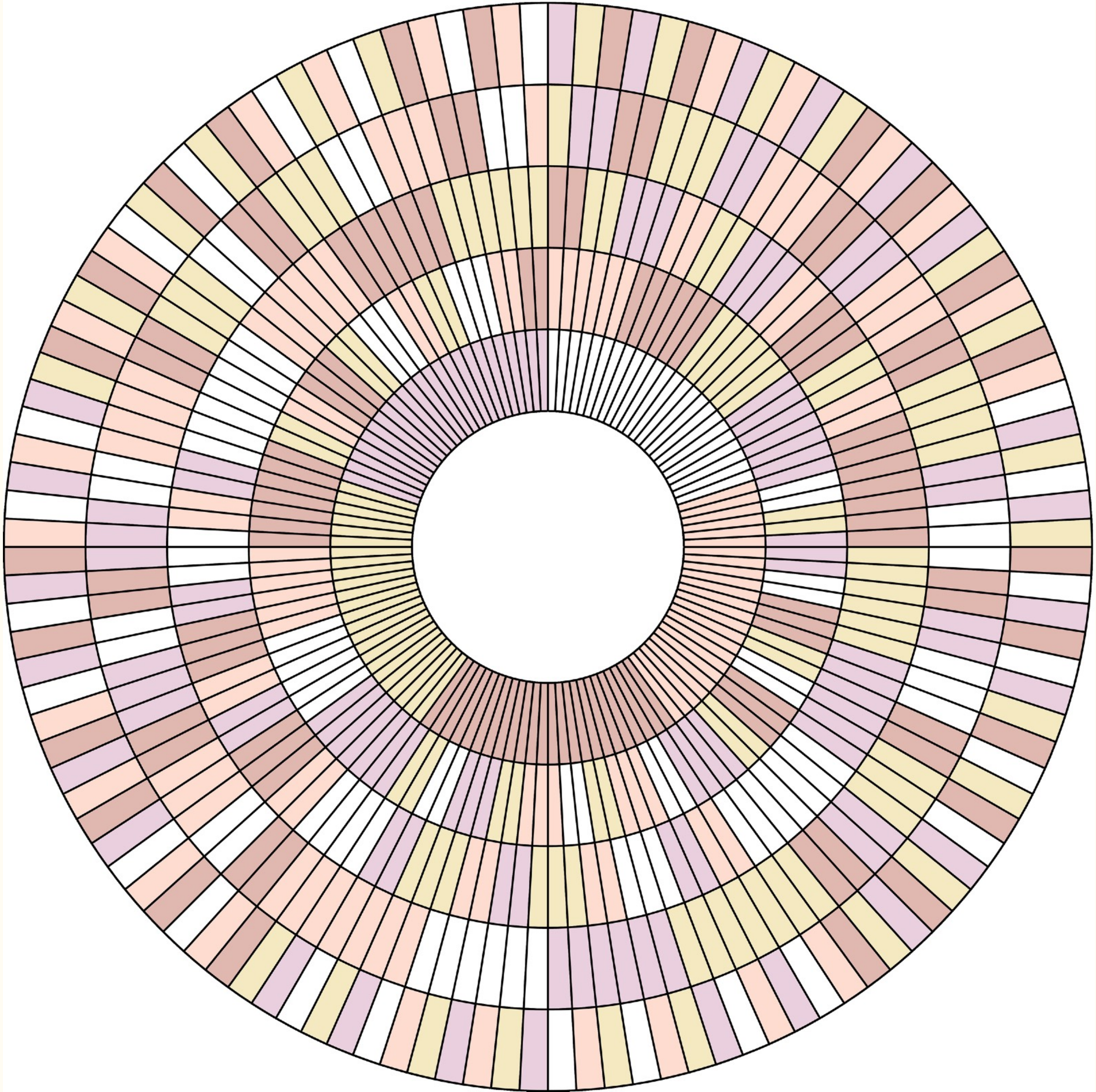


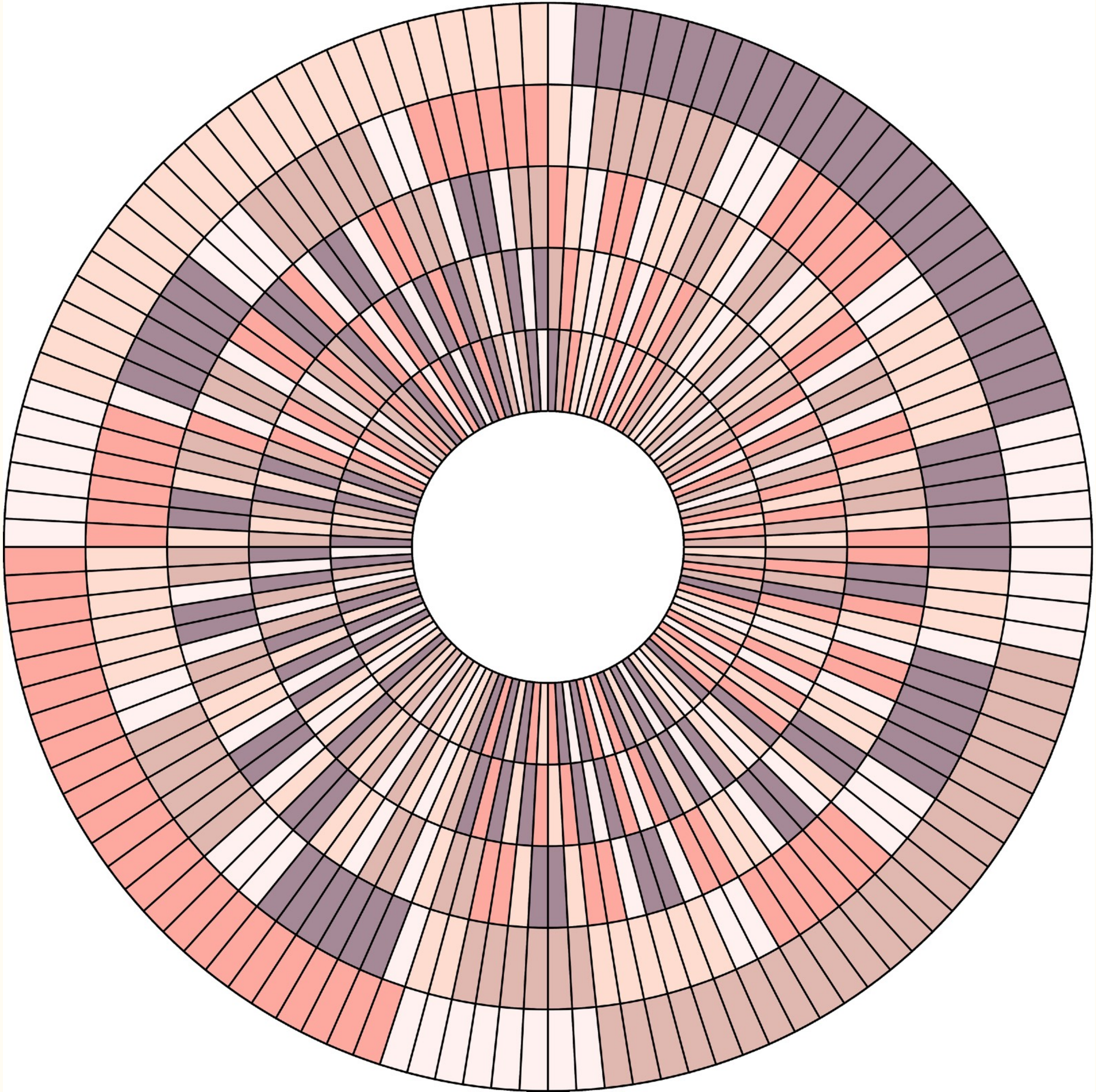













THERE ARE 10
TYPES OF PEOPLE.
THOSE WHO KNOW
BINARY
AND THOSE WHO DON'T.


THERE ARE 11
TYPES OF PEOPLE.
THOSE WHO KNOW
GRAY CODE
AND THOSE WHO DON'T.