Towers of Hanoi, Gray Codes, and Coxeter Groups

James Dann

May 5, 2022

Refresher: Induction

- Proof technique for statements over $0, 1, \ldots$

Refresher: Induction

- Proof technique for statements over $0, 1, \ldots$
- Prove case is true for $n = 0$

Refresher: Induction

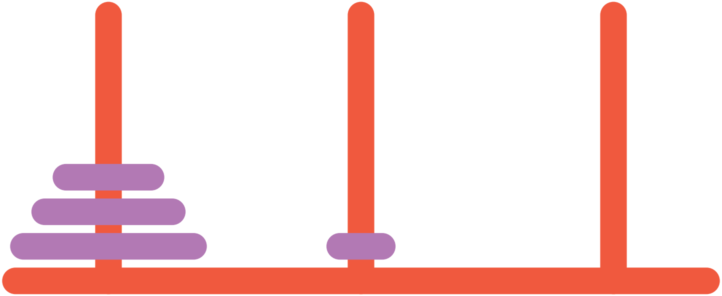
- Proof technique for statements over $0, 1, \ldots$
- Prove case is true for $n = 0$
- Suppose is true for n and prove true for $n + 1$

Refresher: Induction

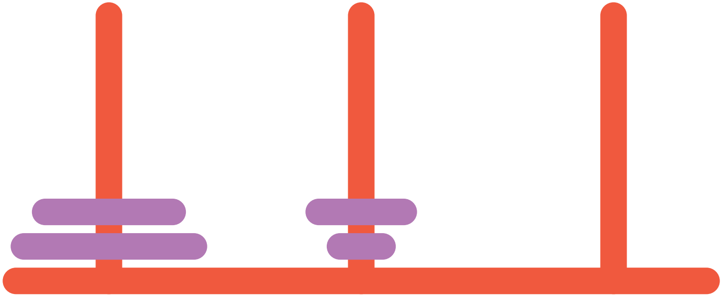
- Proof technique for statements over $0, 1, \ldots$
- Prove case is true for $n = 0$
- Suppose is true for n and prove true for $n + 1$
- True for all $n \in \mathbb{N}$

Problem 1: Towers of Hanoi

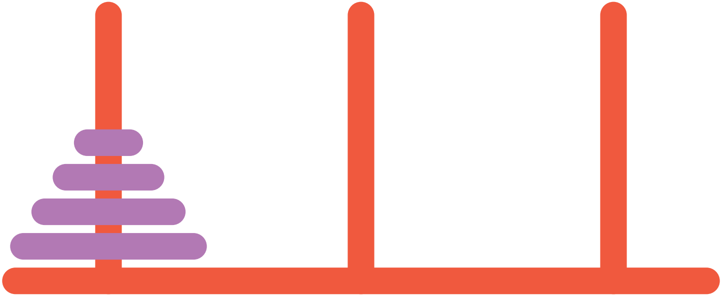
Problem 1: Towers of Hanoi



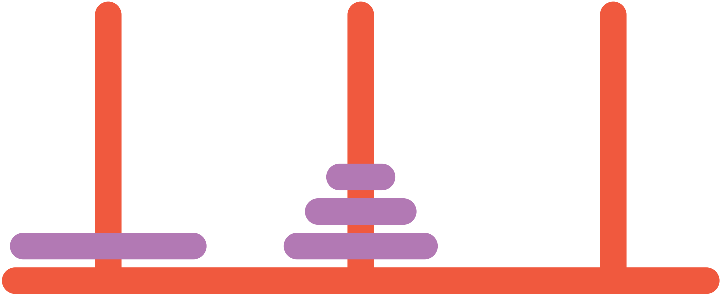
Problem 1: Towers of Hanoi



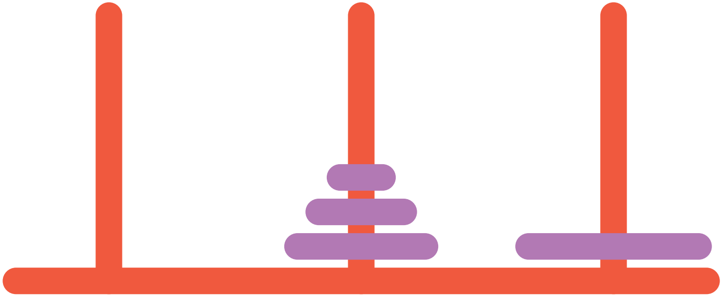
Solution 1: Towers of Hanoi



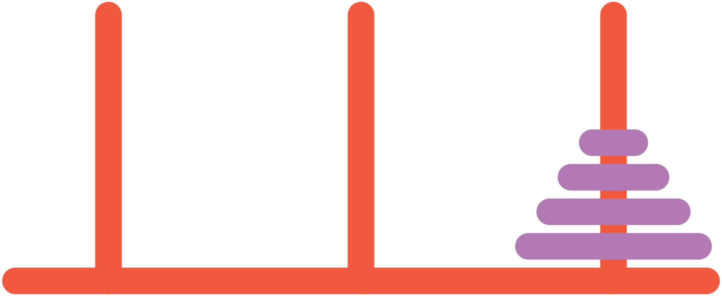
Solution 1: Towers of Hanoi



Solution 1: Towers of Hanoi



Solution 1: Towers of Hanoi



Problem 1: Towers of Hanoi

- Solution involves using the induction hypothesis twice, separated by a single simple move

Problem 1: Towers of Hanoi

- Solution involves using the induction hypothesis twice, separated by a single simple move
- $2^n - 1$ moves required

Problem 1: Towers of Hanoi

- Solution involves using the induction hypothesis twice, separated by a single simple move
- $2^n - 1$ moves required
- Moves through 2^n states

Problem 2: Gray Codes

- Enumerate all binary strings of length n in a cycle such that adjacent strings differ in exactly one place

Problem 2: Gray Codes

- Enumerate all binary strings of length n in a cycle such that adjacent strings differ in exactly one place
- 00, 01, 11, 10

Problem 2: Gray Codes

- Enumerate all binary strings of length n in a cycle such that adjacent strings differ in exactly one place
- 00, 01, 11, 10
- Problem has many applications in signal processing

Problem 2: Gray Codes

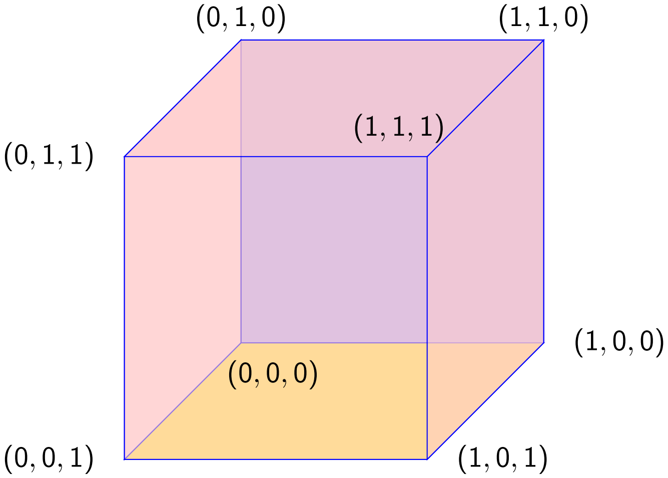
- Enumerate all binary strings of length n in a cycle such that adjacent strings differ in exactly one place
- 00, 01, 11, 10
- Problem has many applications in signal processing
- Connection to n-dimensional cubes

n-dimensional cubes

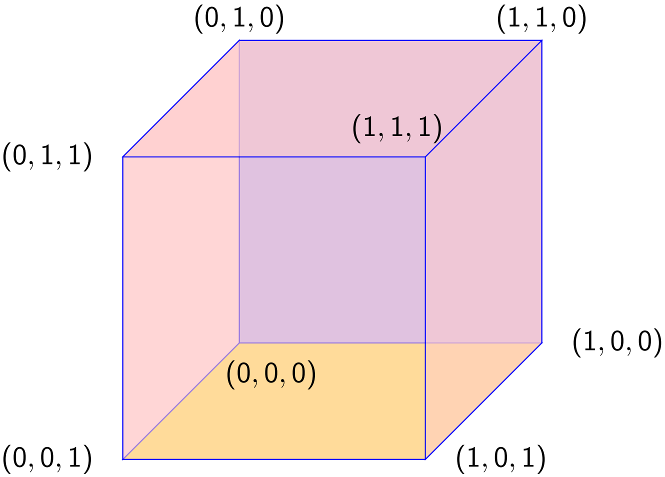
- n-dimensional cube is a graph where the vertices are binary strings and the edges are between strings that differ in exactly one place

n-dimensional cubes

- n-dimensional cube is a graph where the vertices are binary strings and the edges are between strings that differ in exactly one place

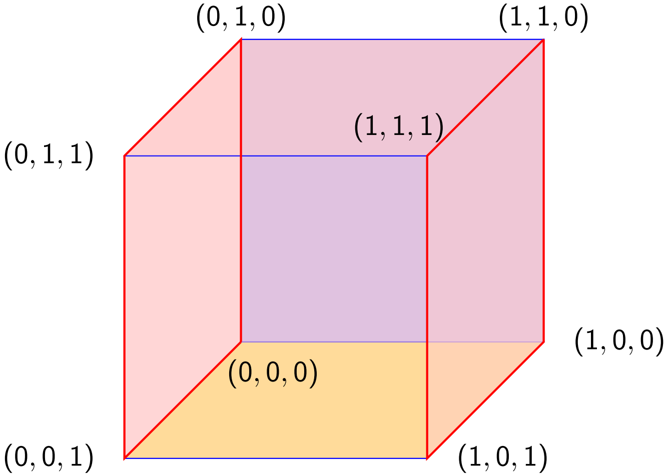


n-dimensional cubes



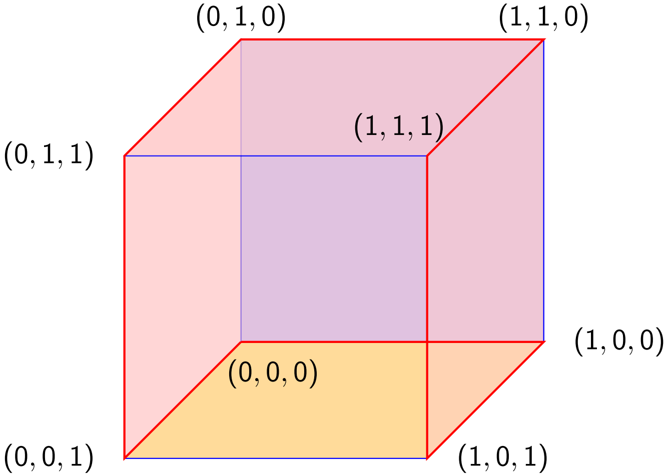
00
01
11
10

n-dimensional cubes



00
01
11
10

n-dimensional cubes



00	000
01	001
11	011
10	010
	110
	111
	101
	100

Problem 2: Gray Code

- 2^n edges required

Problem 2: Gray Code

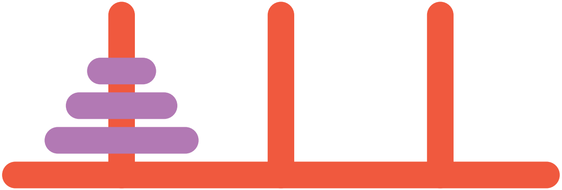
- 2^n edges required
- Moves through 2^n vertices

Problem 2: Gray Code

- 2^n edges required
- Moves through 2^n vertices
- Solution involves using the induction step twice, separated by a single simple move

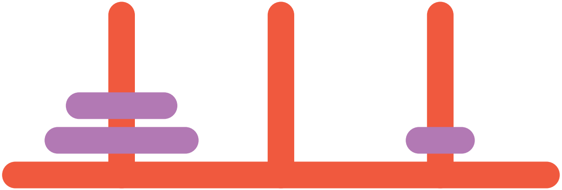
Connection??

String	Diff
000	-
001	0
011	1
010	0
110	2
111	0
101	1
100	0



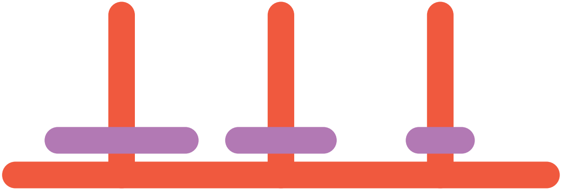
Connection??

String	Diff
000	-
001	**0**
011	1
010	0
110	2
111	0
101	1
100	0



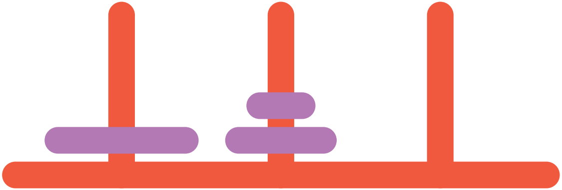
Connection??

String	Diff
000	-
001	0
011	**1**
010	0
110	2
111	0
101	1
100	0



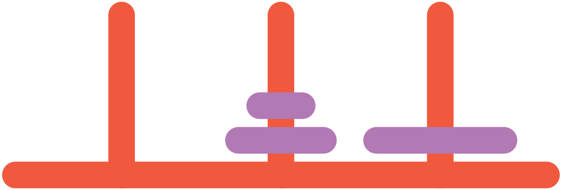
Connection??

String	Diff
000	-
001	0
011	1
010	**0**
110	2
111	0
101	1
100	0



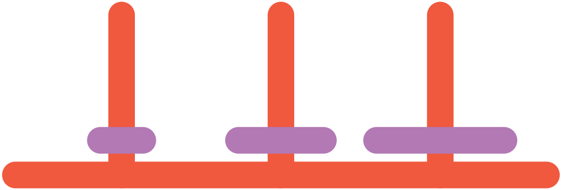
Connection??

String	Diff
000	-
001	0
011	1
010	0
110	**2**
111	0
101	1
100	0



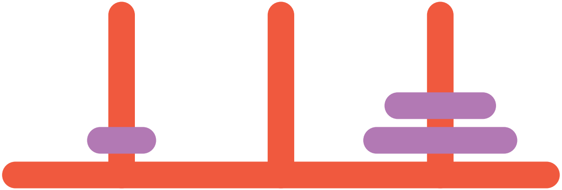
Connection??

String	Diff
000	-
001	0
011	1
010	0
110	2
111	**0**
101	1
100	0



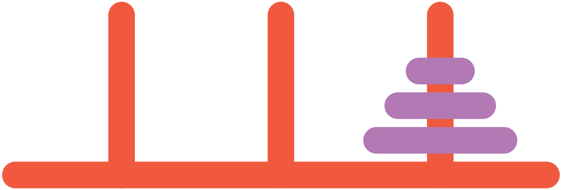
Connection??

String	Diff
000	-
001	0
011	1
010	0
110	2
111	0
101	**1**
100	0



Connection??

String	Diff
000	-
001	0
011	1
010	0
110	2
111	0
101	1
100	**0**



Connection??

String	Diff	Peg 1	Peg 2	Peg 3
000	-	2 1 0		
001	0	2 1		0
011	1	2	1	0
010	0	2	1 0	
110	2		1 0	2
111	0	0	1	2
101	1	0		2 1
100	0			2 1 0

Connection??

String	Diff	Peg 1	Peg 2	Peg 3
000	-	2 1 0		
001	0	2 1		0
011	1	2	1	0
010	0	2	1 0	
110	2		1 0	2
111	0	0	1	2
101	1	0		2 1
100	0			2 1 0

- Flip 0th bit $=$ swap all disks ≤ 0 between pegs containing 0 and 1, if same peg then swap with ? peg

Connection??

String	Diff	Peg 1	Peg 2	Peg 3
000	-	2 1 0		
001	0	2 1		0
011	1	2	1	0
010	0	2	1 0	
110	2		1 0	2
111	0	0	1	2
101	1	0		2 1
100	0			2 1 0

- Flip 0th bit = swap all disks ≤ 0 between pegs containing 0 and 1, if same peg then swap with ? peg
- Flip 1st bit = swap all disks ≤ 1 between pegs containing 1 and 2, if same peg then swap with ? peg

Connection??

String	Diff	Peg 1	Peg 2	Peg 3
000	-	2 1 0		
001	0	2 1		0
011	1	2	1	0
010	0	2	1 0	
110	2		1 0	2
111	0	0	1	2
101	1	0		2 1
100	0			2 1 0

- Flip 0th bit = swap all disks ≤ 0 between pegs containing 0 and 1, if same peg then swap with ? peg
- Flip 1st bit = swap all disks ≤ 1 between pegs containing 1 and 2, if same peg then swap with ? peg
- Flip 2nd bit = swap all disks ≤ 2 between left peg and peg with 2, if same peg then swap with ? peg

A generalisation

- Think about the set of all binary strings W as a group

A generalisation

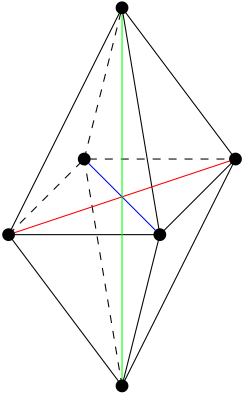
- Think about the set of all binary strings W as a group
- $\mathbb{Z}_2^n$

A generalisation

- Think about the set of all binary strings W as a group
- $\mathbb{Z}_2^n$
- Reflections

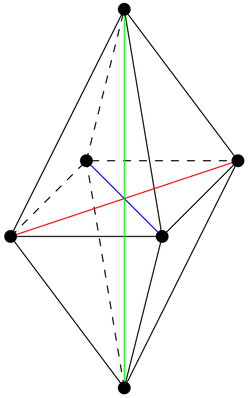
A generalisation

- Think about the set of all binary strings W as a group
- $\mathbb{Z}_2^n$
- Reflections



A generalisation

- Think about the set of all binary strings W as a group
- $\mathbb{Z}_2^n$
- Reflections
- Hang on if you don't know group theory...



Reflection Group

- Three "simple" generators s_0, s_1, s_2 such that $1 = s_0^2 = s_1^2 = s_2^2$

Reflection Group

- Three "simple" generators s_0, s_1, s_2 such that $1 = s_0^2 = s_1^2 = s_2^2$
- All elements can be built out of these generators

Reflection Group

- Three "simple" generators s_0, s_1, s_2 such that $1 = s_0^2 = s_1^2 = s_2^2$
- All elements can be built out of these generators
- These elements commute with each other ($s_i s_j = s_j s_i$)

Reflection Group

- Three "simple" generators s_0, s_1, s_2 such that $1 = s_0^2 = s_1^2 = s_2^2$
- All elements can be built out of these generators
- These elements commute with each other ($s_i s_j = s_j s_i$)
- The elements of $\mathbb{Z}_2^n$ are the elements obtained by multiplication, subject to some relations

Reflection Group

- Three "simple" generators s_0, s_1, s_2 such that $1 = s_0^2 = s_1^2 = s_2^2$
- All elements can be built out of these generators
- These elements commute with each other ($s_i s_j = s_j s_i$)
- The elements of $\mathbb{Z}_2^n$ are the elements obtained by multiplication, subject to some relations

$$\mathbb{Z}_2^n = \left\langle s_0, s_1, s_2 \mid 1 = s_i^2 = (s_0 s_1)^2 = (s_1 s_2)^2 = (s_0 s_2)^2 \right\rangle$$

Coxeter System

- Set of generators $S = \{s_1, s_2, \ldots, s_n\}$

Coxeter System

- Set of generators $S = \{s_1, s_2, \ldots, s_n\}$
- Represents reflections in (hyper)planes

Coxeter System

- Set of generators $S = \{s_1, s_2, \ldots, s_n\}$
- Represents reflections in (hyper)planes
- Every generator is its own inverse ($s_i^2 = 1$)

Coxeter System

- Set of generators $S = \{s_1, s_2, \ldots, s_n\}$
- Represents reflections in (hyper)planes
- Every generator is its own inverse ($s_i^2 = 1$)
- Relations of the form $(s_i s_j)^{m(i,j)} = 1$

Coxeter System

- Set of generators $S = \{s_1, s_2, \dots, s_n\}$
- Represents reflections in (hyper)planes
- Every generator is its own inverse ($s_i^2 = 1$)
- Relations of the form $(s_i s_j)^{m(i,j)} = 1$
- Represents angle between s_i and s_j: $\frac{\pi}{m(i,j)}$

Symmetric Group

- Permutations on n letters

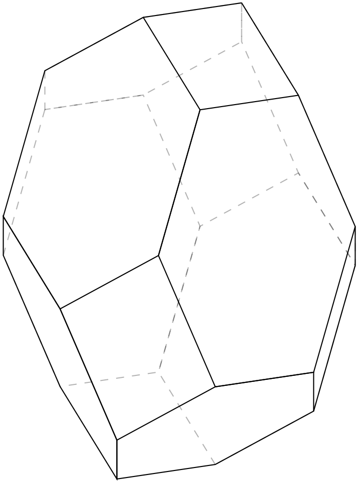
Symmetric Group

- Permutations on n letters
- Generated by adjacent transpositions $(1\ 2), (2\ 3), \ldots$

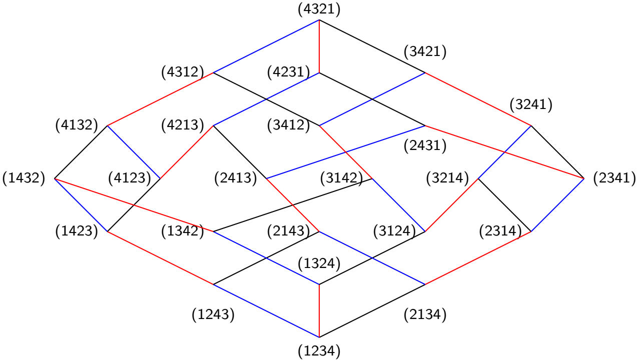
Symmetric Group

- Permutations on n letters
- Generated by adjacent transpositions $(1\ 2), (2\ 3), \ldots$
- Satisfy aforementioned relations, with $(s_i s_{i+1})^3 = 1$

Tikz is hard

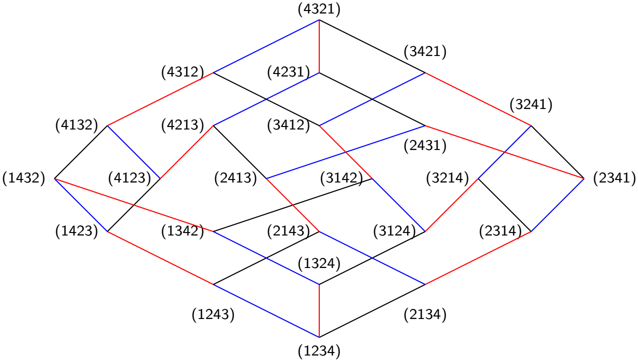


Decomposition



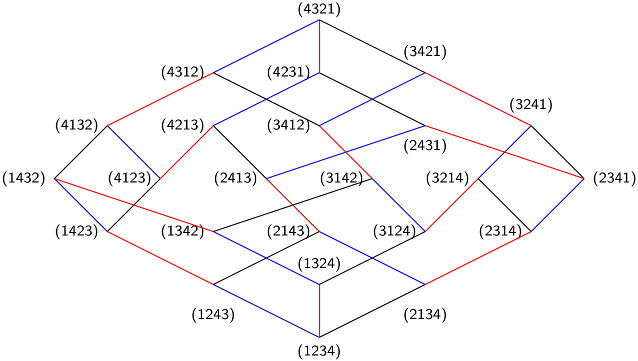
Decomposition

- Inductive decomposition



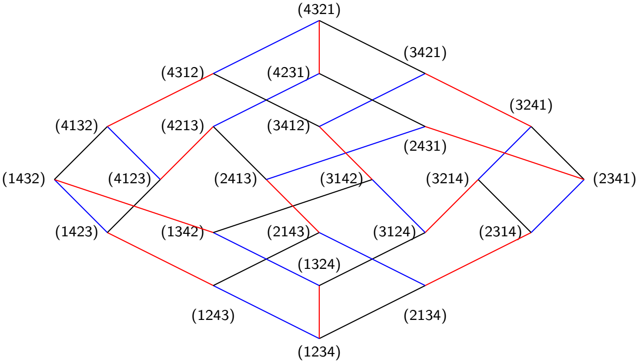
Decomposition

- Inductive decomposition
- $g =$ something that ends only in blue * something never using blue



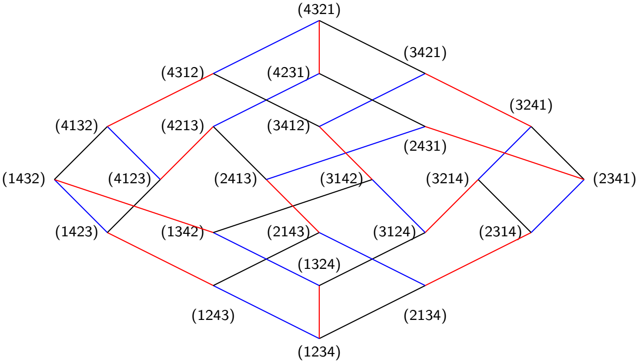
Decomposition

- Split into smaller parts with Hamilton cycles



Decomposition

- Split into smaller parts with Hamilton cycles
- Always connected



Decomposition

- Split into smaller parts with Hamilton cycles
- Always connected
- Connect hamilton cycles