Theorietag 2026
TU Ilmenau
April 1, 2026

Minimizing a word by permuting its coordinates

Dominik Scheder, TU Chemnitz

Theorietag 2026
TU Ilmenau
April 1, 2026

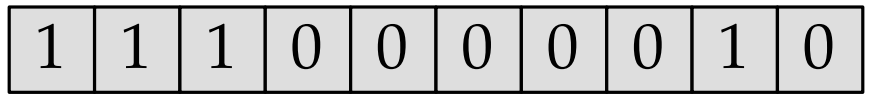
Minimizing a word by permuting its coordinates

Dominik Scheder, TU Chemnitz

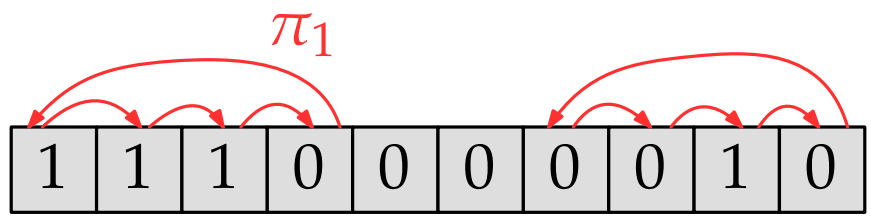
Based on joint work with Johannes Tantow, also TU Chemnitz

A Concrete Example

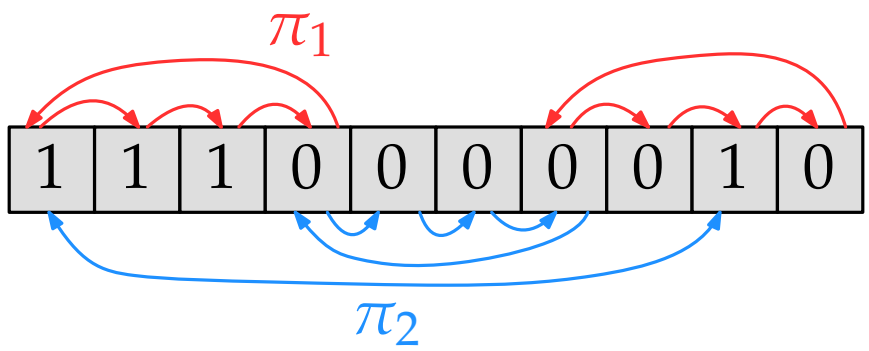
A Concrete Example



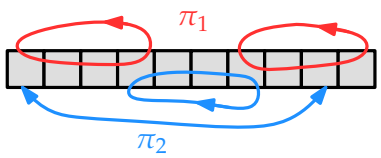
A Concrete Example



A Concrete Example

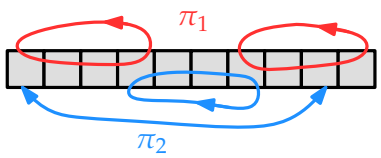


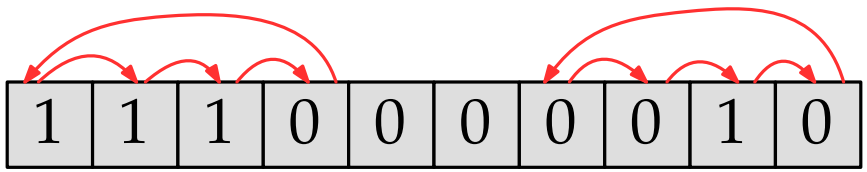
A Concrete Example



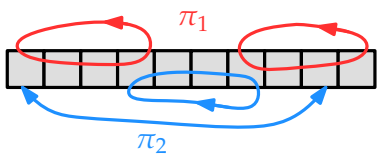
1	1	1	0	0	0	0	0	1	0

A Concrete Example

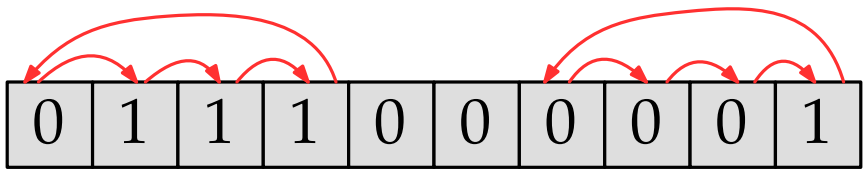




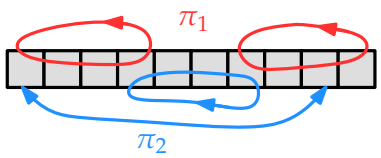
A Concrete Example



applying π_1 makes the word smaller

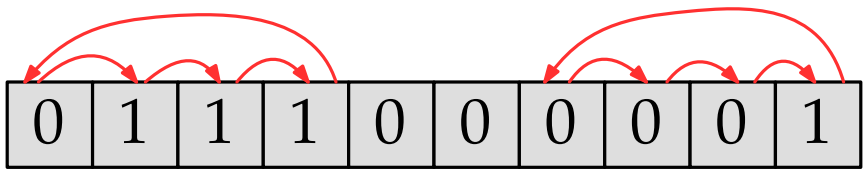


A Concrete Example

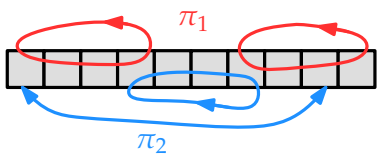


applying π_1 makes the word smaller

applying π_1 again would make word larger



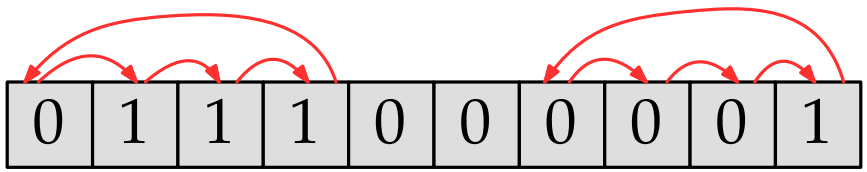
A Concrete Example



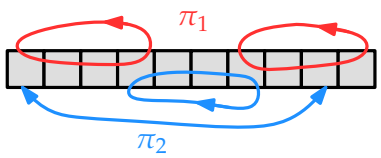
applying π_1 makes the word smaller

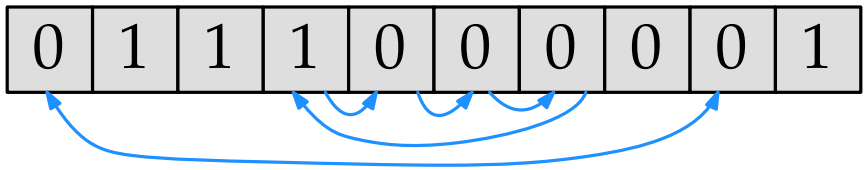
applying π_1 again would make word larger

cannot apply π_1 again



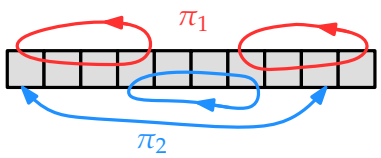
A Concrete Example

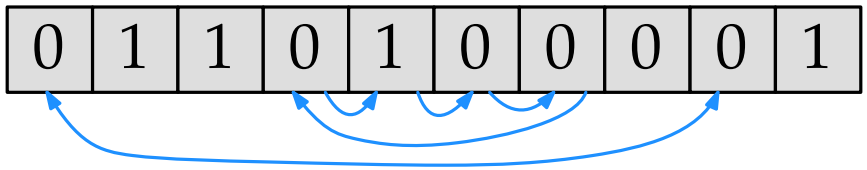




but can apply π_2

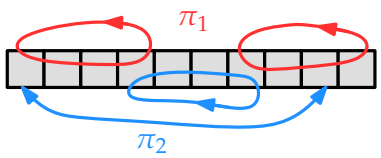
A Concrete Example

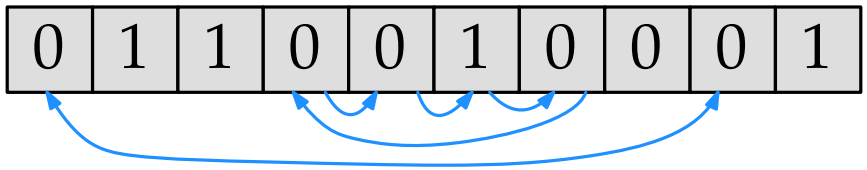




but can apply π_2

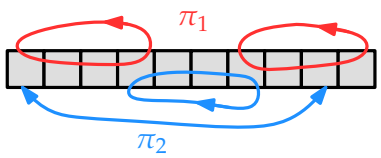
A Concrete Example

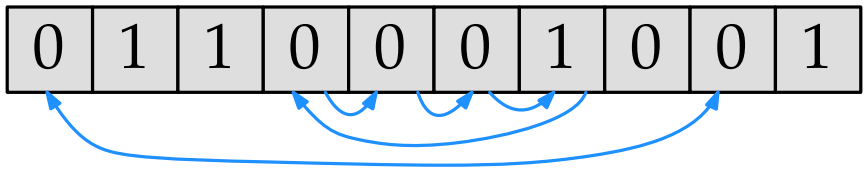




but can apply π_2
and again

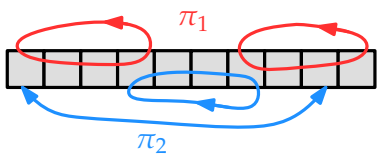
A Concrete Example

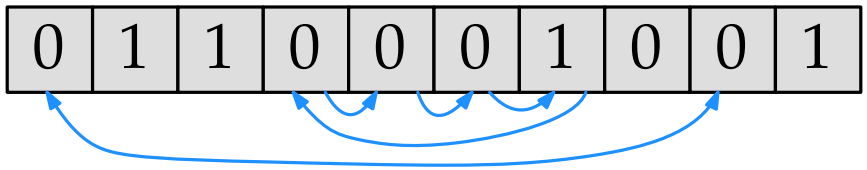




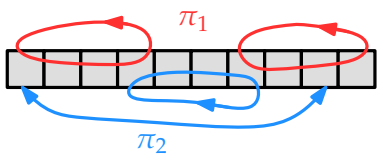
but can apply π_2
and again
and again

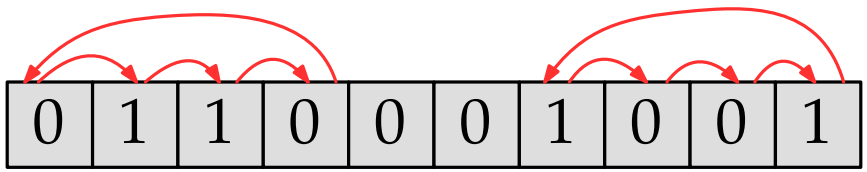
A Concrete Example



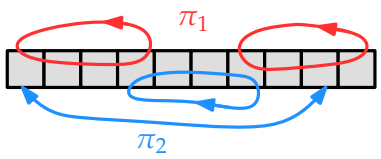


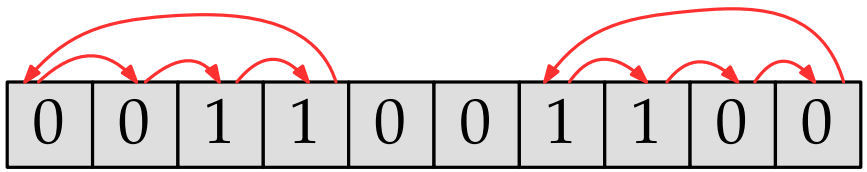
A Concrete Example



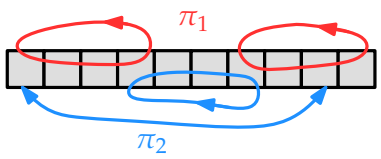


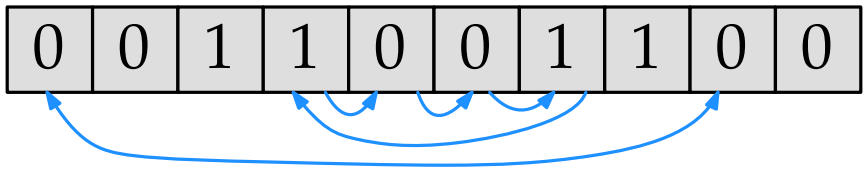
A Concrete Example



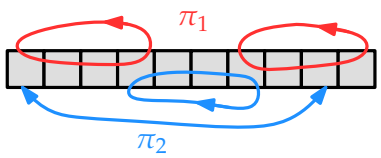


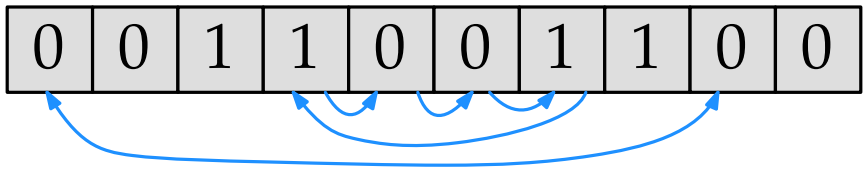
A Concrete Example





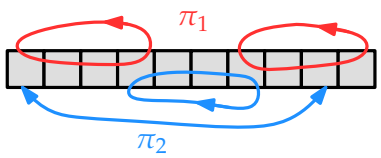
A Concrete Example

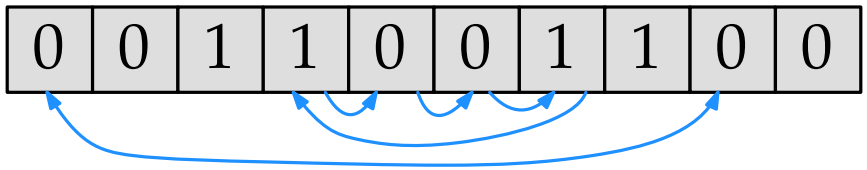




Applying π_2 would make the word larger

A Concrete Example

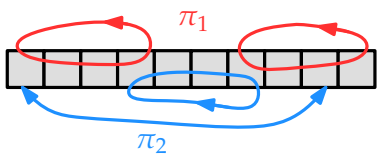


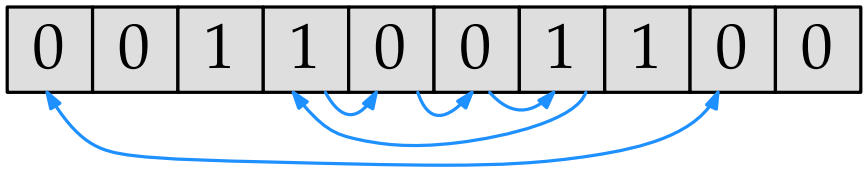


Applying π_2 would make the word larger

This is a local minimum.

A Concrete Example



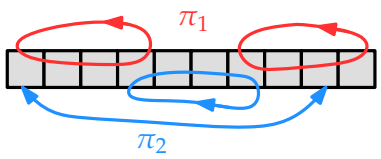


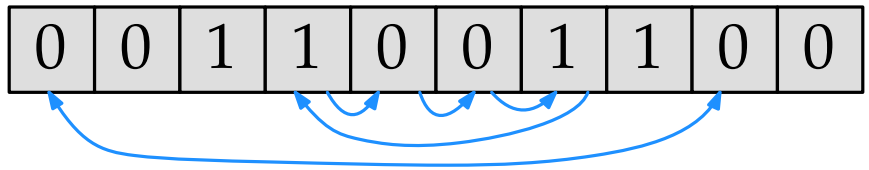
Applying π_2 would make the word larger

This is a local minimum.

But not a global minimum.

A Concrete Example





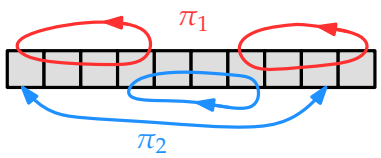
Applying π_2 would make the word larger

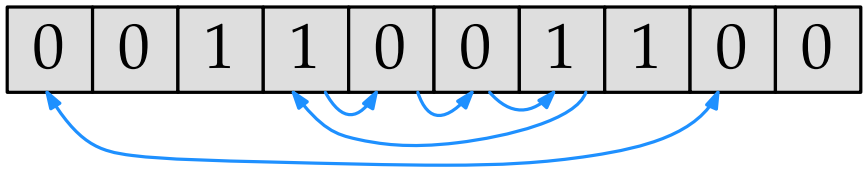
This is a local minimum.

But not a global minimum.

This "greedy" procedure always finds a local minimum.

A Concrete Example





Applying π_2 would make the word larger

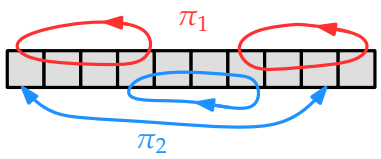
This is a local minimum.

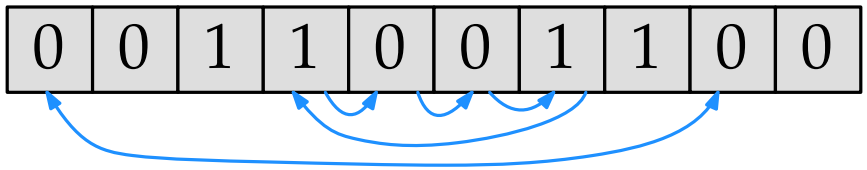
But not a global minimum.

This "greedy" procedure always finds a local minimum.

How long can it take?

A Concrete Example





Applying π_2 would make the word larger

This is a local minimum.

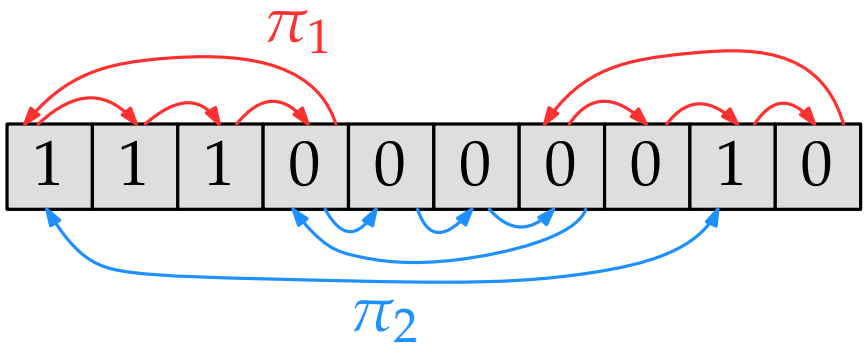
But not a global minimum.

This "greedy" procedure always finds a local minimum.

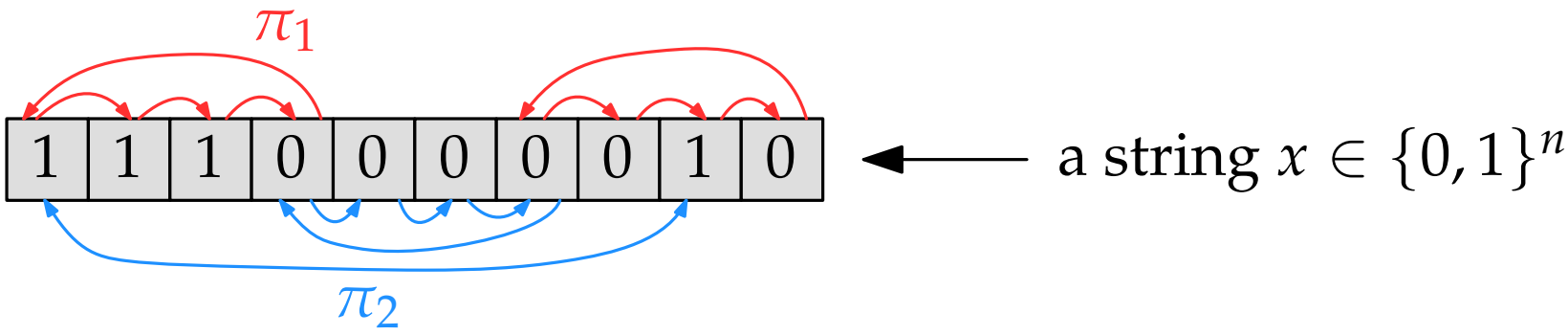
How long can it take?

Can we compute a local minimum directly?

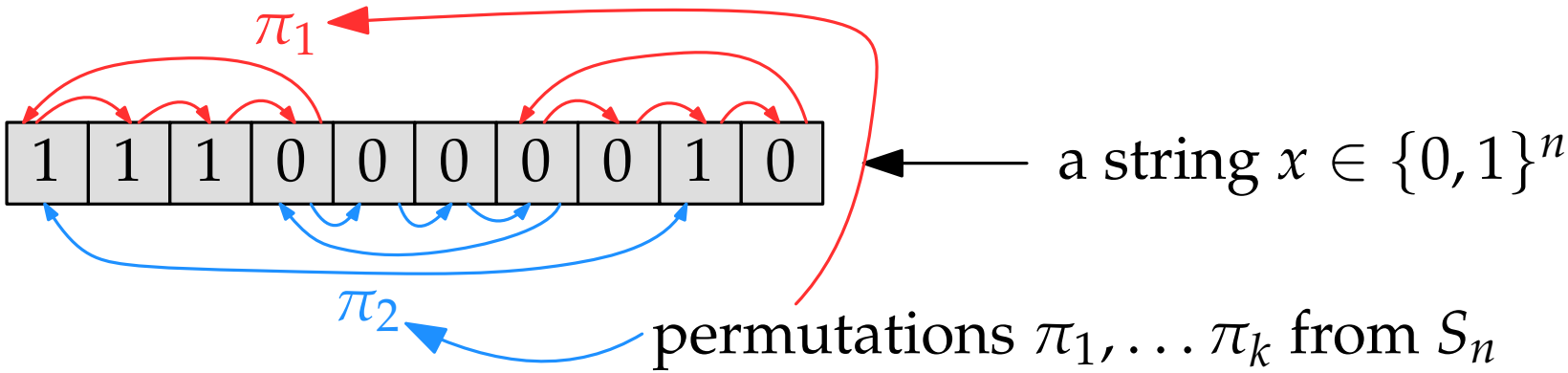
The Formal Setup



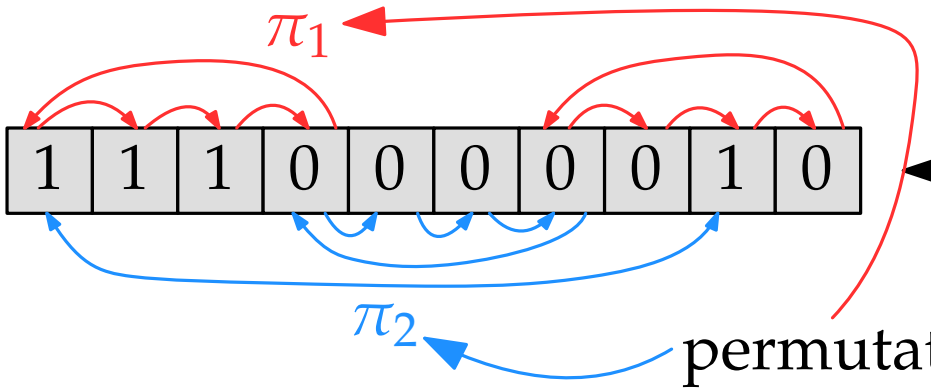
The Formal Setup



The Formal Setup



The Formal Setup

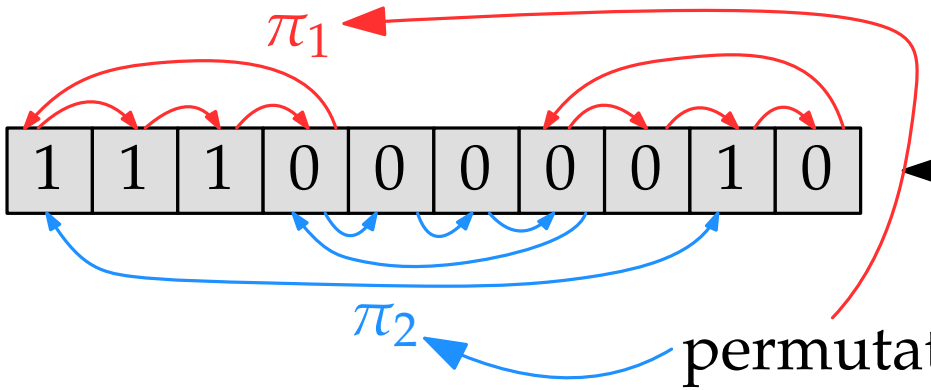


a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

they generate a subgroup $G = \langle \pi_1, \ldots, \pi_k \rangle$

The Formal Setup



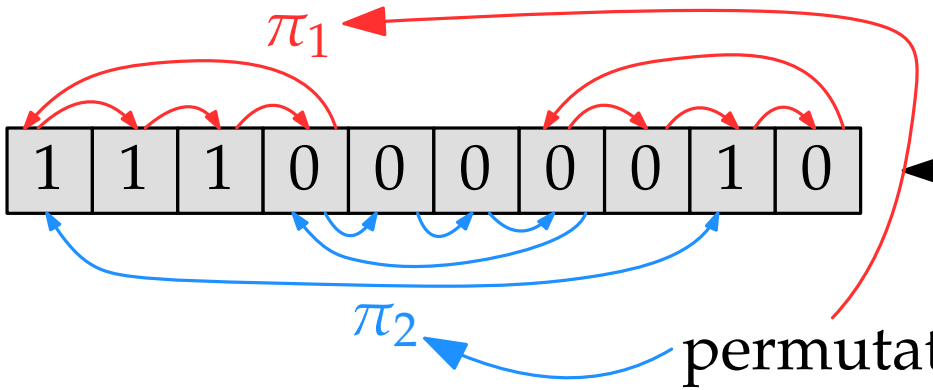
a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

they generate a subgroup $G = \langle \pi_1, \ldots, \pi_k \rangle$

they send x into the orbit $Gx = \{x \circ \pi \mid \pi \in G\}$

The Formal Setup



a string $x \in \{0,1\}^n$

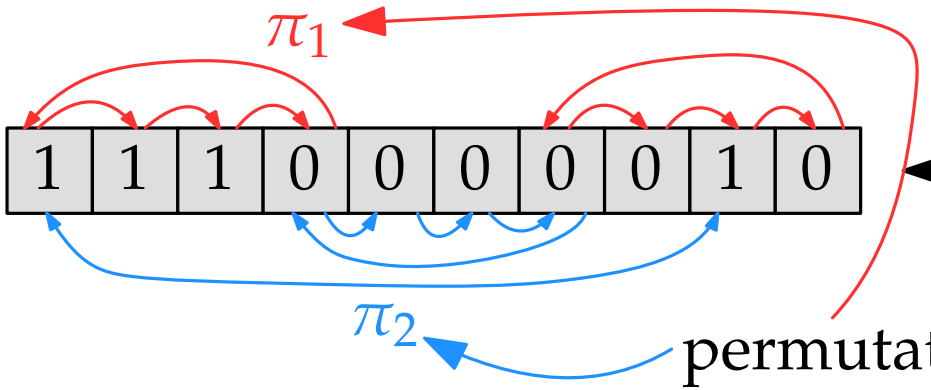
permutations $\pi_1, \ldots \pi_k$ from S_n

they generate a subgroup $G = \langle \pi_1, \ldots, \pi_k \rangle$

they send x into the orbit $Gx = \{x \circ \pi \mid \pi \in G\}$

that's the correct order of composition by the way, since $\pi : [n] \to [n]$ and $x : [n] \to \{0,1\}$, so $x(\pi(i))$ makes sense syntactically

The Formal Setup



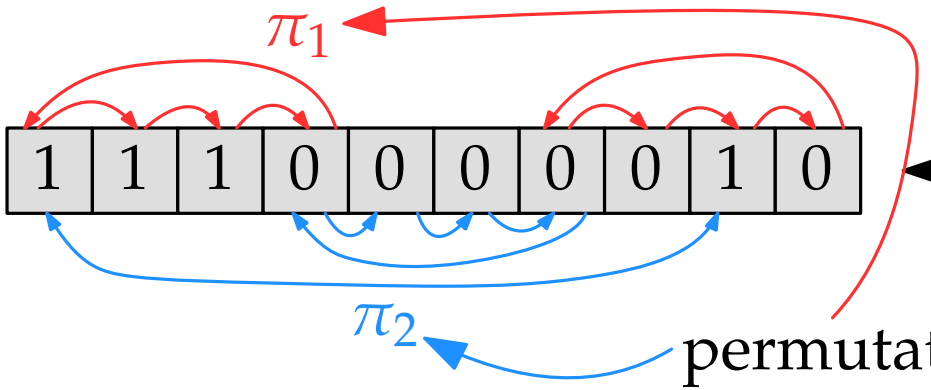
a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

they generate a subgroup $G = \langle \pi_1, \ldots, \pi_k \rangle$

they send x into the orbit $Gx = \{x \circ \pi \mid \pi \in G\}$

The Formal Setup



a string $x \in \{0,1\}^n$

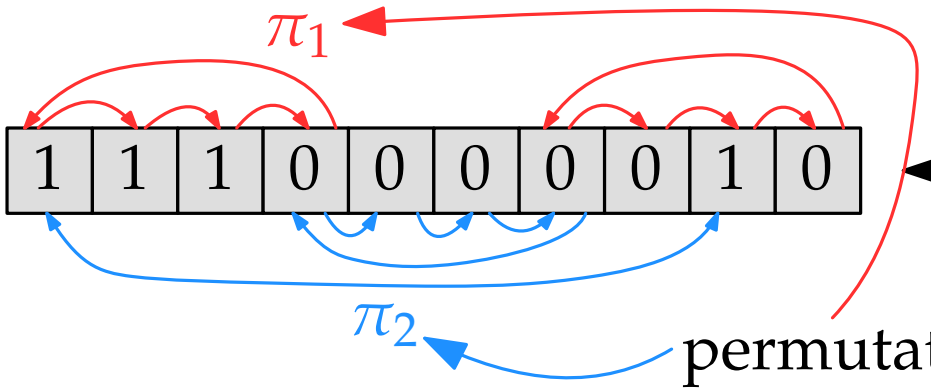
permutations $\pi_1, \ldots \pi_k$ from S_n

they generate a subgroup $G = \langle \pi_1, \ldots, \pi_k \rangle$

they send x into the orbit $Gx = \{x \circ \pi \mid \pi \in G\}$

global optimum: smallest element $y \in Gx$

The Formal Setup



a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

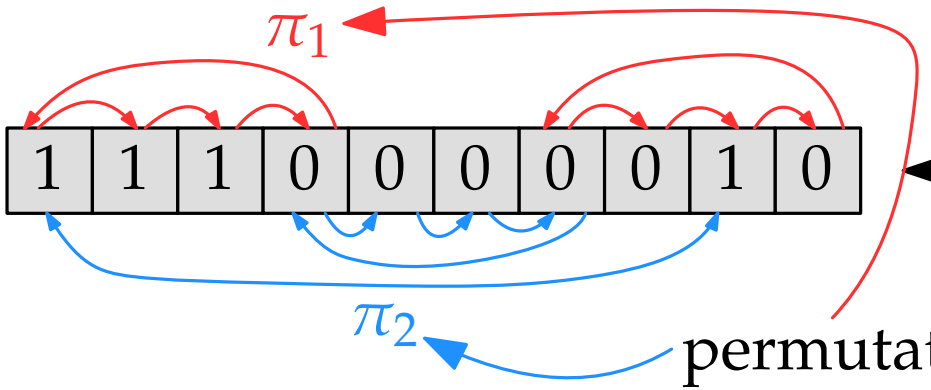
they generate a subgroup $G = \langle \pi_1, \ldots, \pi_k \rangle$

they send x into the orbit $Gx = \{x \circ \pi \mid \pi \in G\}$

global optimum: smallest element $y \in Gx$

in the lexicographic order on $\{0,1\}^n$

The Formal Setup



a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

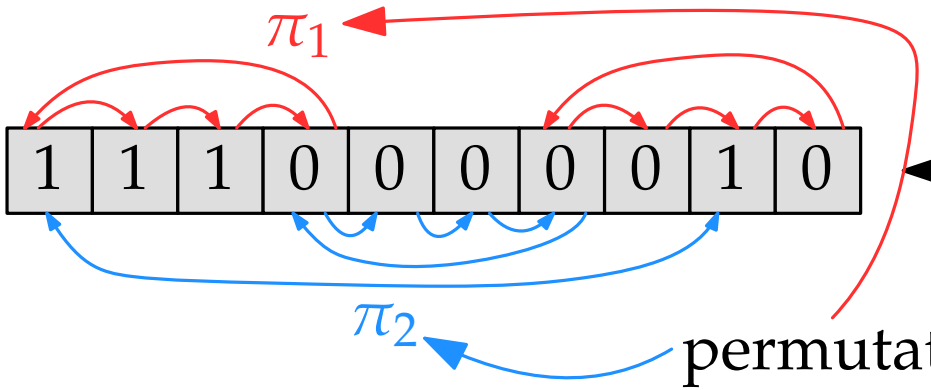
they generate a subgroup $G = \langle \pi_1, \ldots, \pi_k \rangle$

they send x into the orbit $Gx = \{x \circ \pi \mid \pi \in G\}$

global optimum: smallest element $y \in Gx$

local optimum: $y \in Gx$ such that $y \circ \pi_i \succeq y$ for all $1 \leq i \leq k$

The Formal Setup



a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

they generate a subgroup $G = \langle \pi_1, \ldots, \pi_k \rangle$

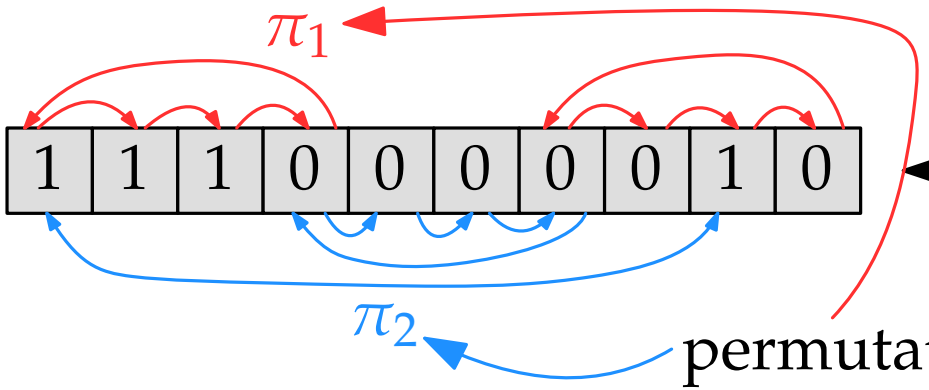
they send x into the orbit $Gx = \{x \circ \pi \mid \pi \in G\}$

global optimum: smallest element $y \in Gx$

local optimum: $y \in Gx$ such that $y \circ \pi_i \succeq y$ for all $1 \leq i \leq k$

Motivation. This question was posed by Kołodziejczyk and Thapen in the context of simulating symmetry-breaking rules in certain proof systems.

About the Global Optimum

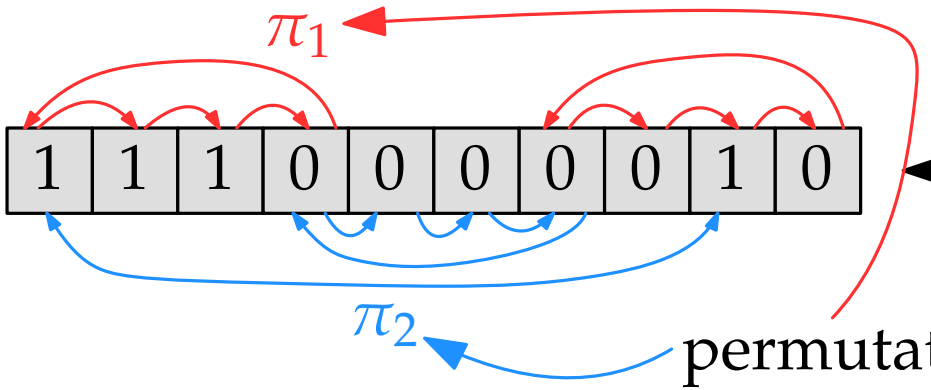


a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

global optimum: smallest element of orbit $\{x \circ \pi \mid \pi \in \langle \pi_1, \ldots, \pi_k \rangle\}$

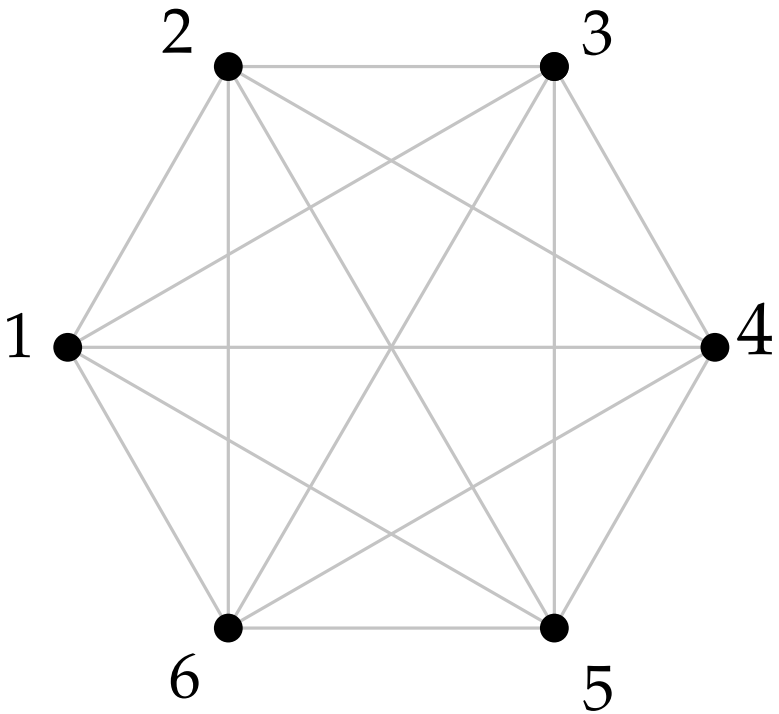
About the Global Optimum



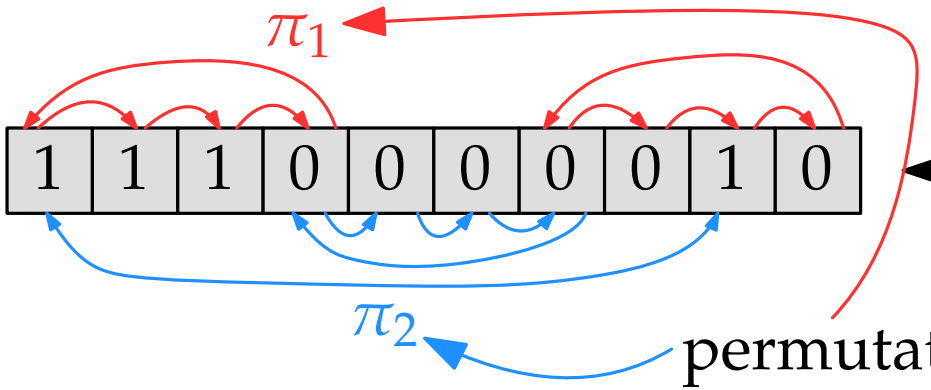
a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

global optimum: smallest element of orbit $\{x \circ \pi \mid \pi \in \langle \pi_1, \ldots, \pi_k \rangle\}$



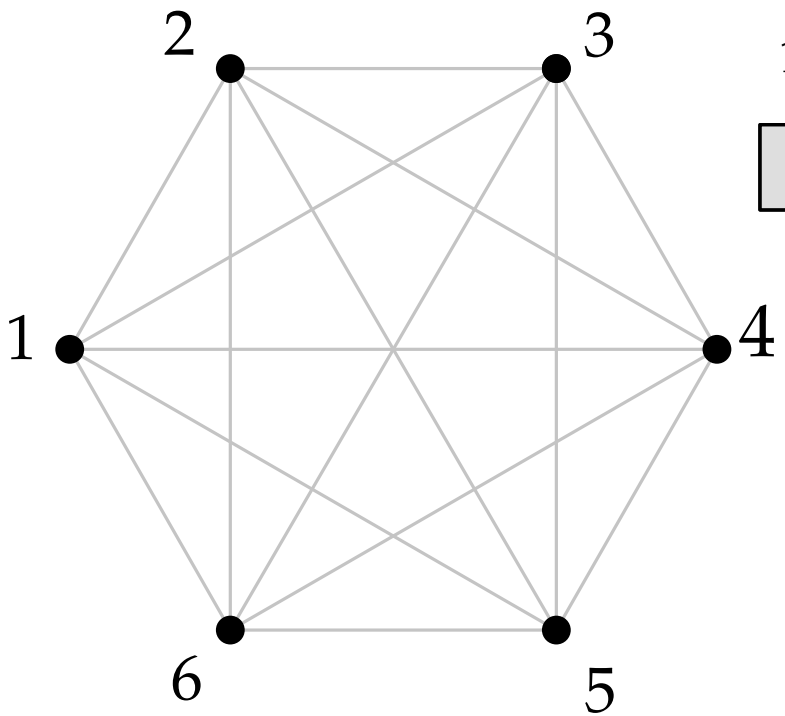
About the Global Optimum

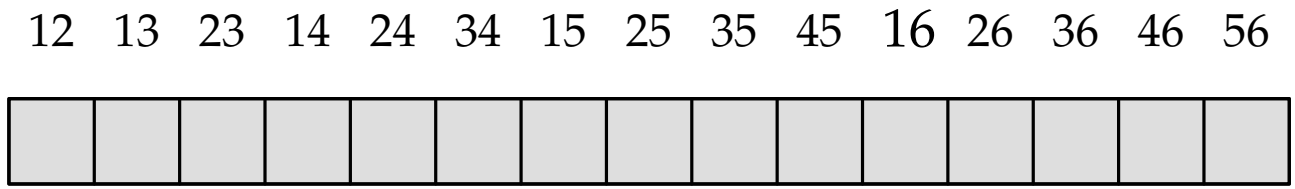


a string $x \in \{0,1\}^n$

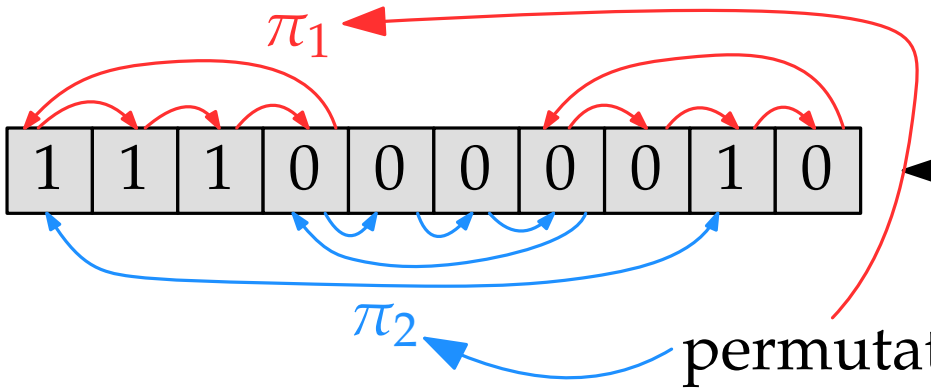
permutations $\pi_1, \ldots \pi_k$ from S_n

global optimum: smallest element of orbit $\{x \circ \pi \mid \pi \in \langle \pi_1, \ldots, \pi_k \rangle\}$





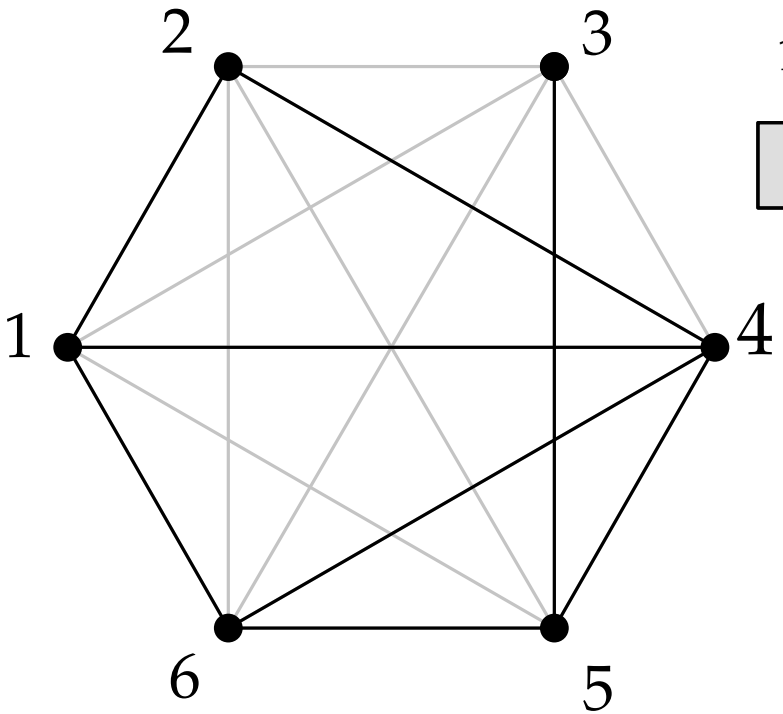
About the Global Optimum

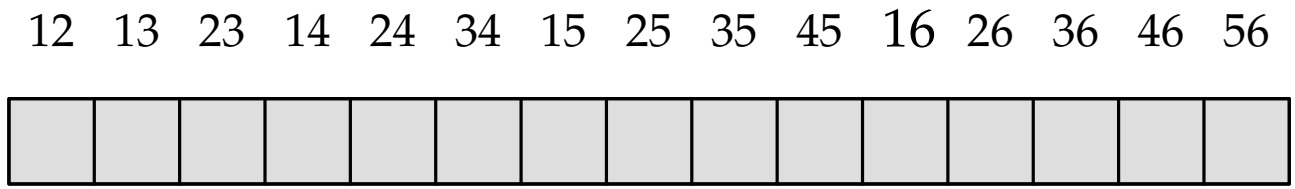


a string $x \in \{0,1\}^n$

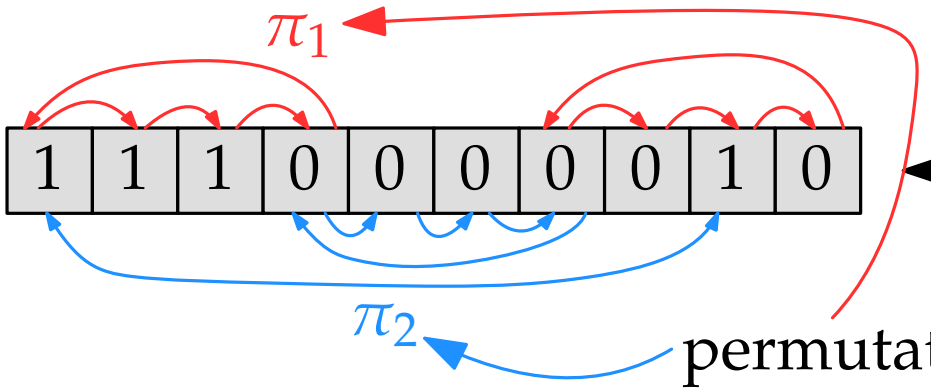
permutations $\pi_1, \ldots \pi_k$ from S_n

global optimum: smallest element of orbit $\{x \circ \pi \mid \pi \in \langle \pi_1, \ldots, \pi_k \rangle\}$





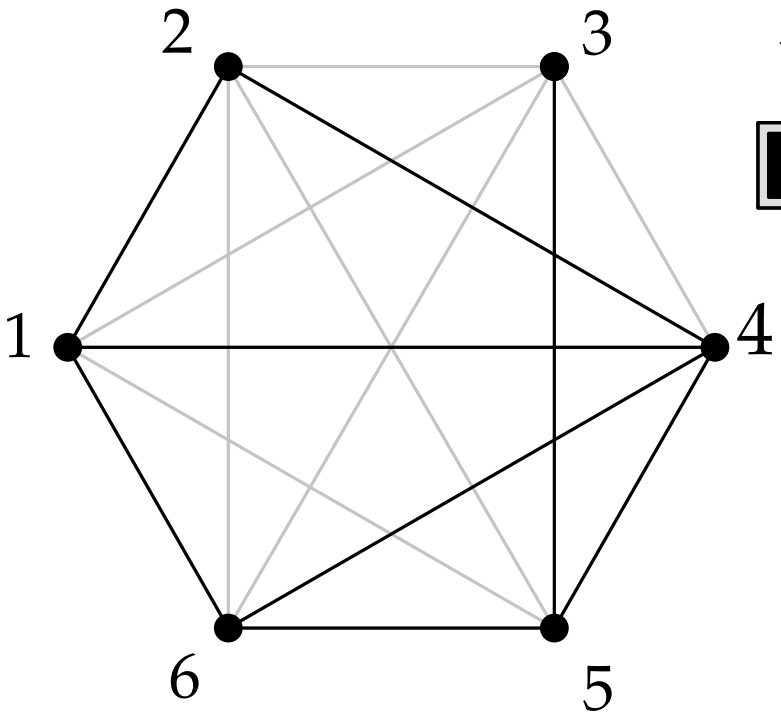
About the Global Optimum

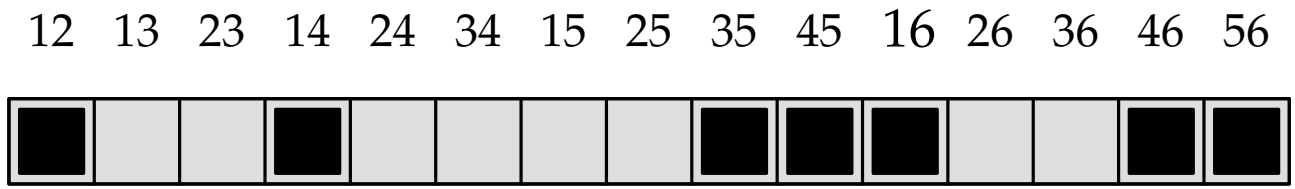


a string $x \in \{0,1\}^n$

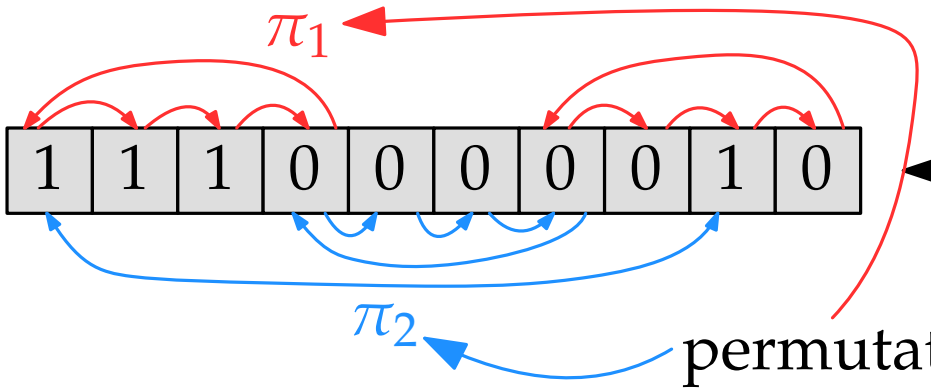
permutations $\pi_1, \ldots \pi_k$ from S_n

global optimum: smallest element of orbit $\{x \circ \pi \mid \pi \in \langle \pi_1, \ldots, \pi_k \rangle\}$





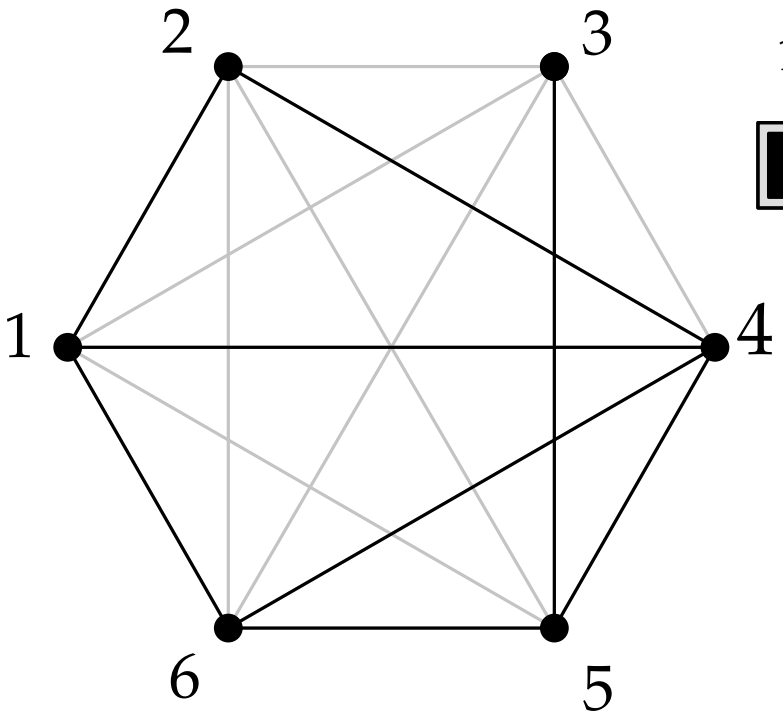
About the Global Optimum

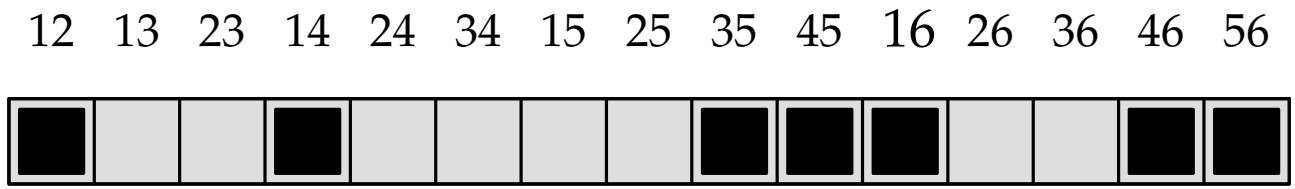


a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

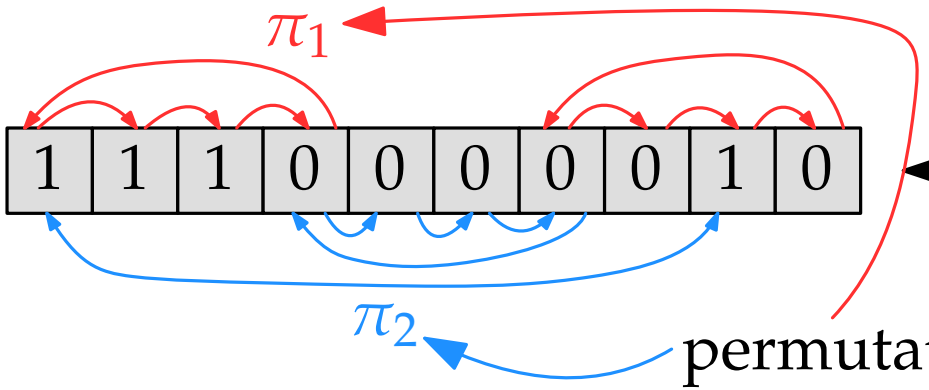
global optimum: smallest element of orbit $\{x \circ \pi \mid \pi \in \langle \pi_1, \ldots, \pi_k \rangle\}$





π_1 := switch vertex 1 and 2

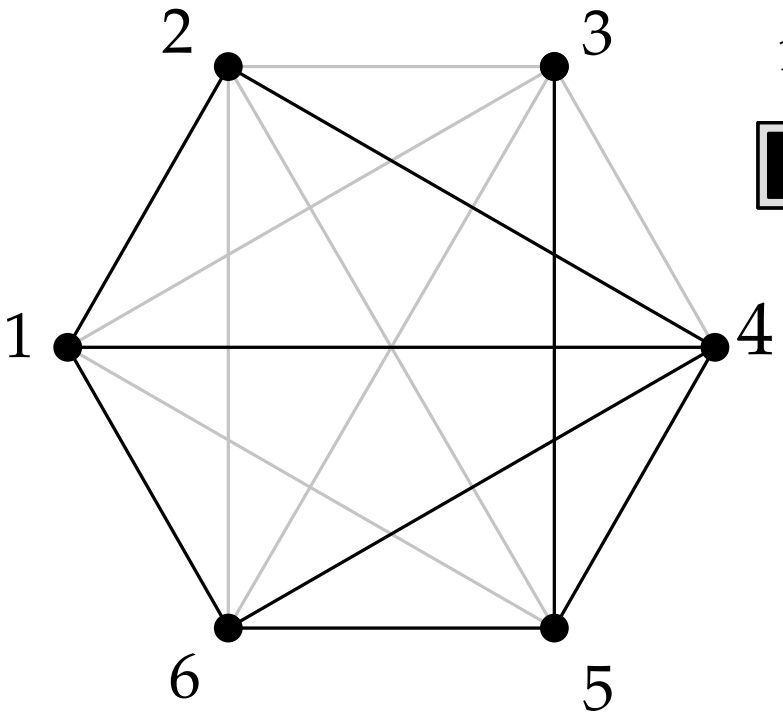
About the Global Optimum

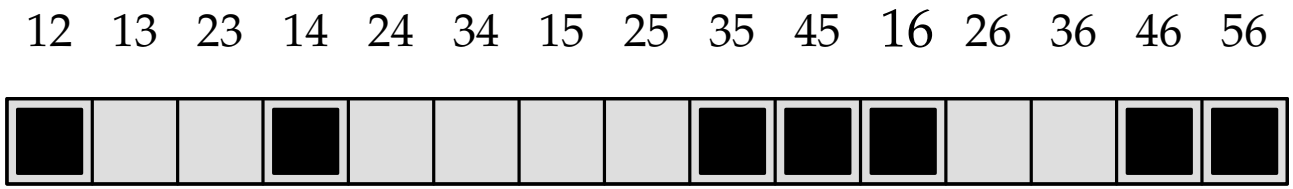


a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

global optimum: smallest element of orbit $\{x \circ \pi \mid \pi \in \langle \pi_1, \ldots, \pi_k \rangle\}$

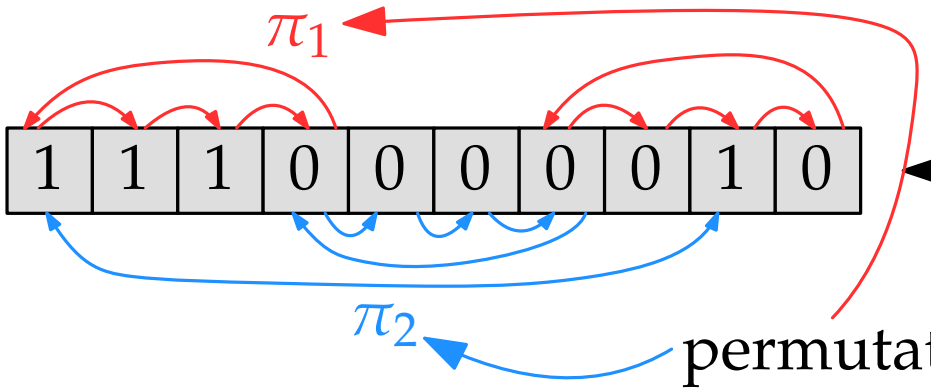




π_1 := switch vertex 1 and 2

π_2 := rotate vertices clockwise

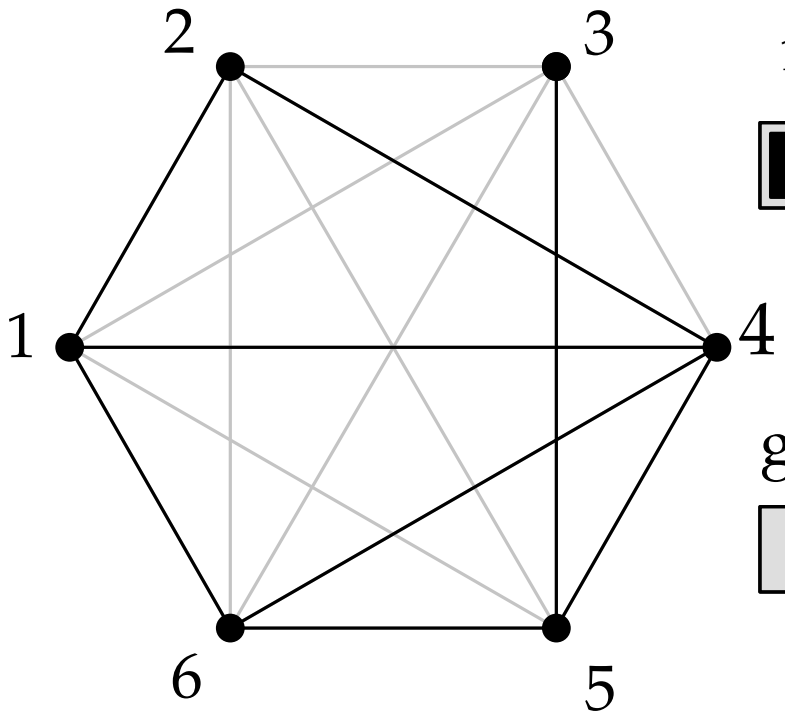
About the Global Optimum

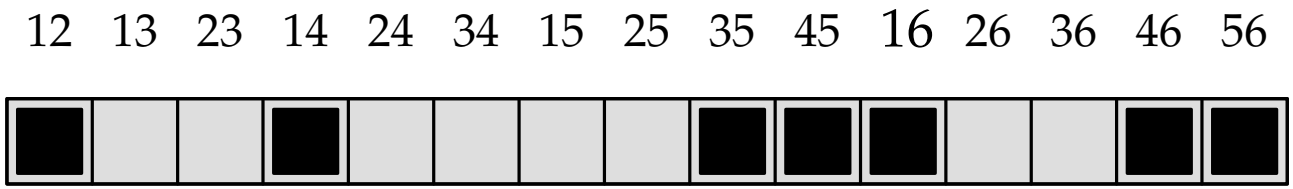


a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

global optimum: smallest element of orbit $\{x \circ \pi \mid \pi \in \langle \pi_1, \ldots, \pi_k \rangle\}$

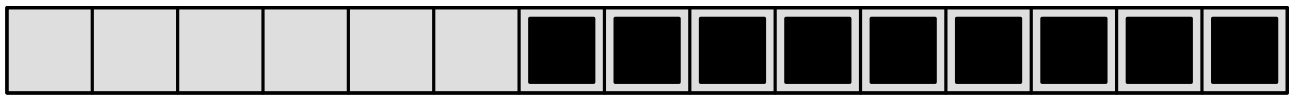




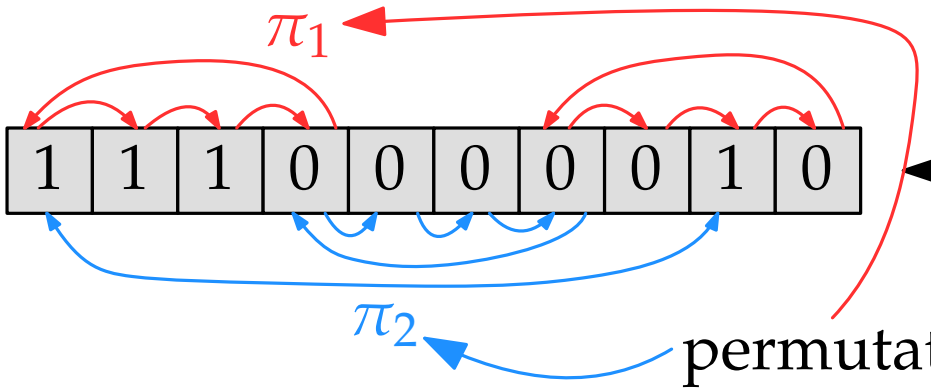
π_1 := switch vertex 1 and 2

π_2 := rotate vertices clockwise

global min $\preceq$



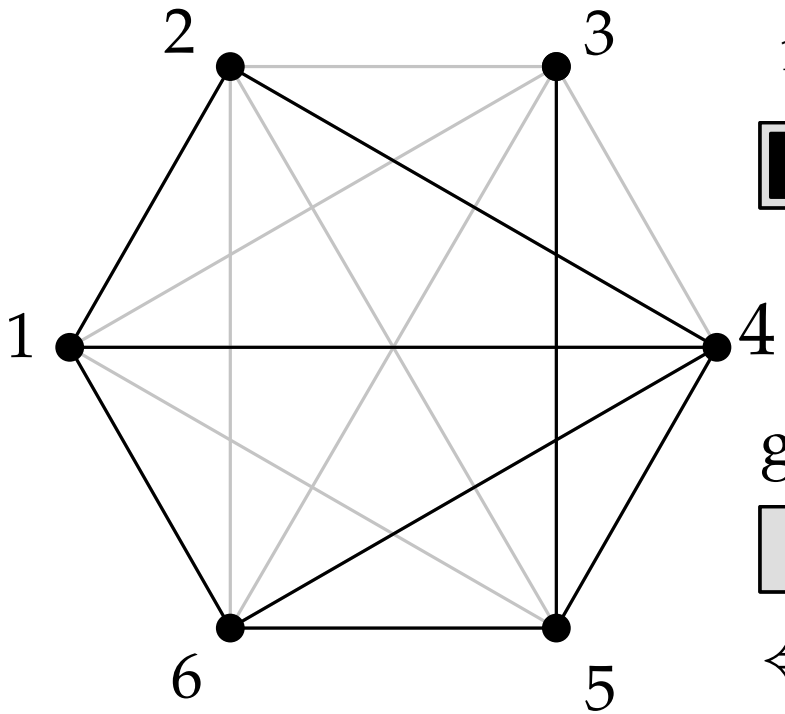
About the Global Optimum

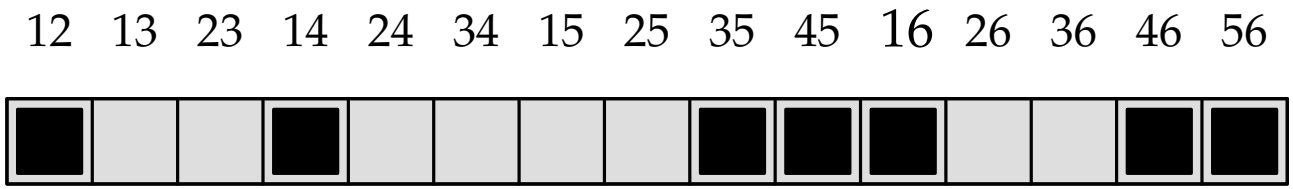


a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

global optimum: smallest element of orbit $\{x \circ \pi \mid \pi \in \langle \pi_1, \ldots, \pi_k \rangle\}$

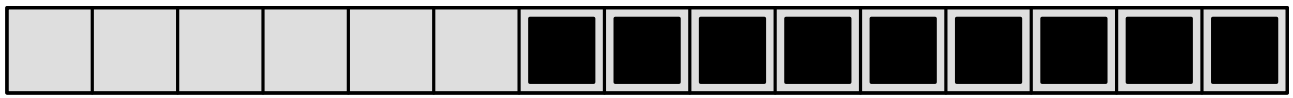




π_1 := switch vertex 1 and 2

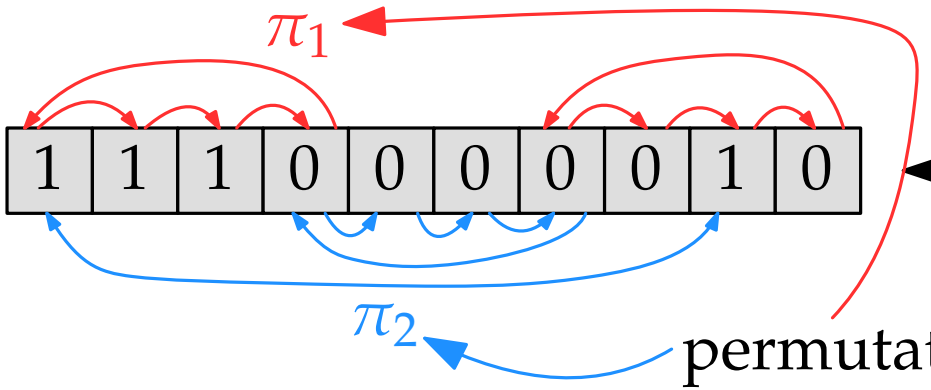
π_2 := rotate vertices clockwise

global min $\preceq$



$\Longleftrightarrow$ G has independent set of size k

About the Global Optimum



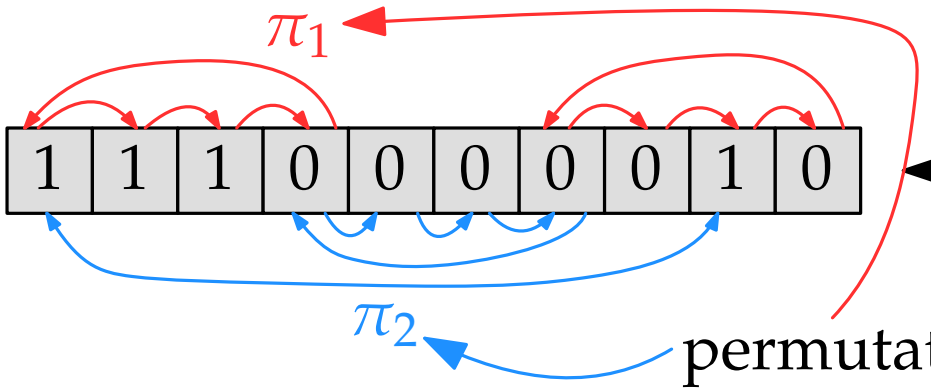
a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

global optimum: smallest element of orbit $\{x \circ \pi \mid \pi \in \langle \pi_1, \ldots, \pi_k \rangle\}$

Theorem (Babai and Luks, 1983). This is NP-hard, even for $k = 2$.

About the Global Optimum



a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

global optimum: smallest element of orbit $\{x \circ \pi \mid \pi \in \langle \pi_1, \ldots, \pi_k \rangle\}$

Theorem (Babai and Luks, 1983). This is NP-hard, even for $k = 2$.

Theorem (S. and Tantow, 2025). This is NP-hard, even for $k = 1$.

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

x

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

$x \prec x \circ \pi_{i_1}$

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

$x \prec x \circ \pi_{i_1} \prec x \circ \pi_{i_1} \circ \pi_{i_2}$

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

$$x \prec x \circ \pi_{i_1} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3}$$

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

$$x \prec x \circ \pi_{i_1} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \circ \pi_{i_4}$$

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

$$x \prec x \circ \pi_{i_1} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \circ \pi_{i_4}$$

We call this an *improving path*.

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

$$x \prec x \circ \pi_{i_1} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \circ \pi_{i_4}$$

We call this an *improving path*.

An improving path will arrive at a local optimum.

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

$$x \prec x \circ \pi_{i_1} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \circ \pi_{i_4}$$

We call this an *improving path*.

An improving path will arrive at a local optimum.

How long can this take?

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

$$x \prec x \circ \pi_{i_1} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \circ \pi_{i_4}$$

We call this an *improving path*.

An improving path will arrive at a local optimum.

How long can this take?

only polynomially many steps

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

$$x \prec x \circ \pi_{i_1} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \circ \pi_{i_4}$$

We call this an *improving path*.

An improving path will arrive at a local optimum.

How long can this take?

only polynomially many steps
then all is well

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

$$x \prec x \circ \pi_{i_1} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \circ \pi_{i_4}$$

We call this an *improving path*.

An improving path will arrive at a local optimum.

How long can this take? → exponentially many steps

↓

only polynomially many steps
then all is well

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

$$x \prec x \circ \pi_{i_1} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \circ \pi_{i_4}$$

We call this an *improving path*.

An improving path will arrive at a local optimum.

How long can this take? → exponentially many steps

shortcut to finding a local minimum?

↓

only polynomially many steps

then all is well

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

$$x \prec x \circ \pi_{i_1} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \circ \pi_{i_4}$$

We call this an *improving path*.

An improving path will arrive at a local optimum.

How long can this take? → exponentially many steps

shortcut to finding a local minimum? ↓ clever algorithms

↓ only polynomially many steps then all is well

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

$$x \prec x \circ \pi_{i_1} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \circ \pi_{i_4}$$

We call this an *improving path*.

An improving path will arrive at a local optimum.

How long can this take?

exponentially many steps

shortcut to finding a local minimum?

only polynomially many steps

then all is well

clever algorithms

hardness results

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

$$x \prec x \circ \pi_{i_1} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \circ \pi_{i_4}$$

We call this an *improving path*.

An improving path will arrive at a local optimum.

How long can this take?

exponentially many steps

shortcut to finding a local minimum?

only polynomially many steps

then all is well

clever algorithms

(few and far between)

hardness results

Finding a Local Optimum

a string $x \in \{0,1\}^n$

permutations $\pi_1, \ldots \pi_k$ from S_n

$$x \prec x \circ \pi_{i_1} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \prec x \circ \pi_{i_1} \circ \pi_{i_2} \circ \pi_{i_3} \circ \pi_{i_4}$$

We call this an *improving path*.

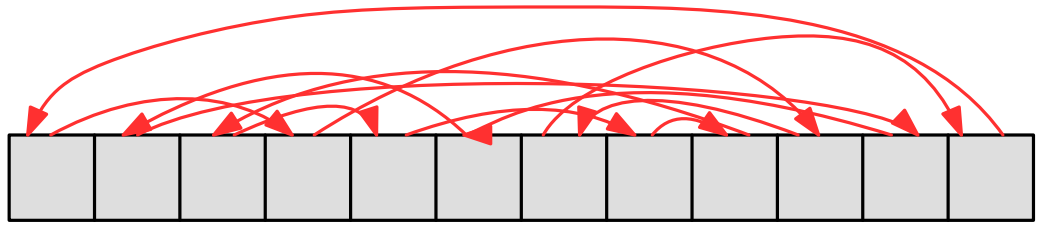
An improving path will arrive at a local optimum.

How long can this take?

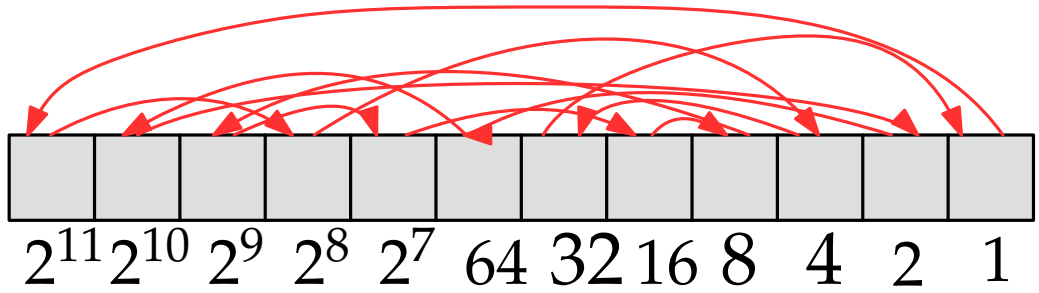
- only polynomially many steps then all is well
- exponentially many steps shortcut to finding a local minimum?
  - clever algorithms (few and far between)
  - hardness results (S. and Tantow, 2025)

Finding a Local Optimum for $k = 1$

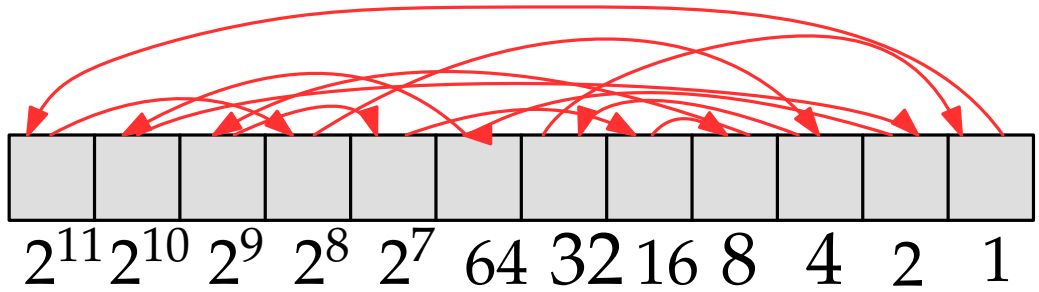
Finding a Local Optimum for $k = 1$

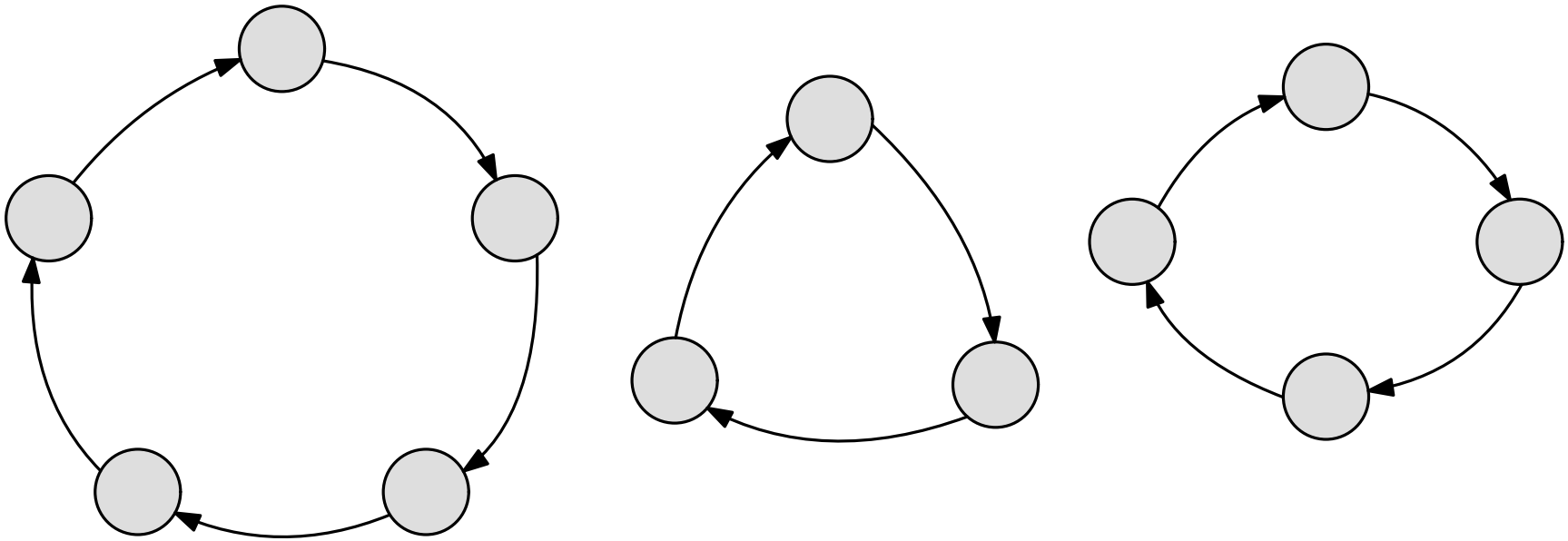


Finding a Local Optimum for $k = 1$

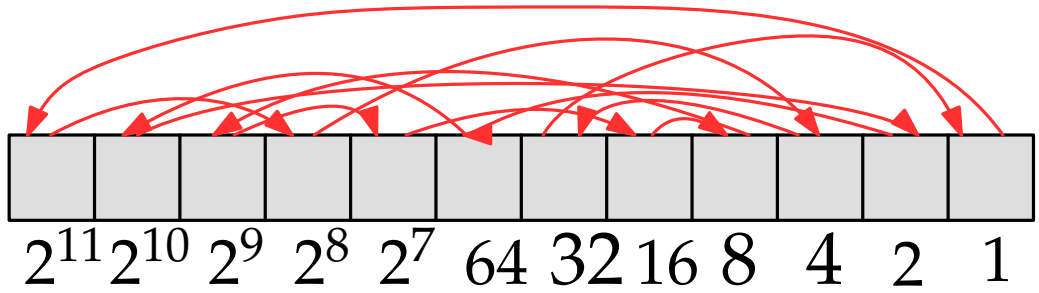


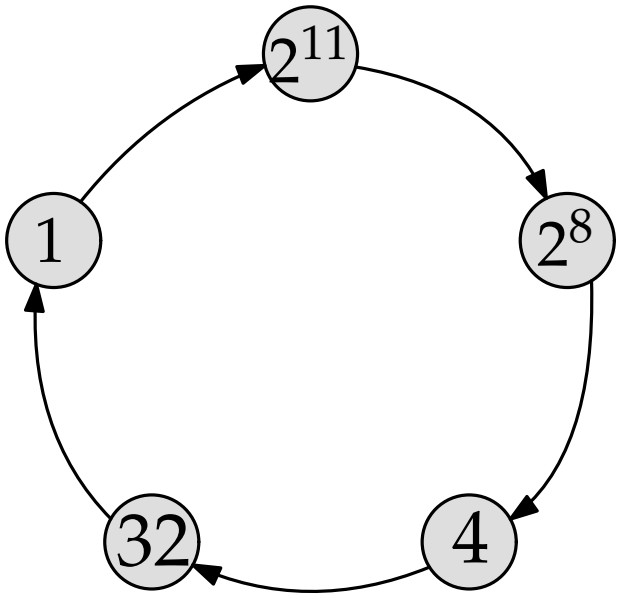
Finding a Local Optimum for $k = 1$

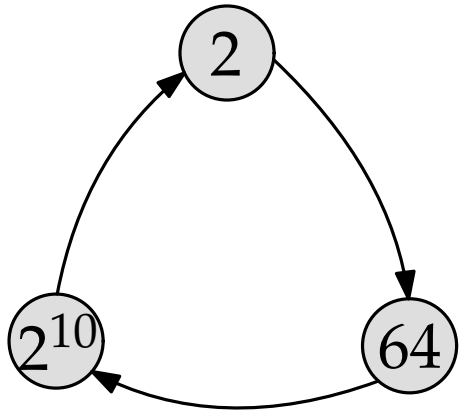


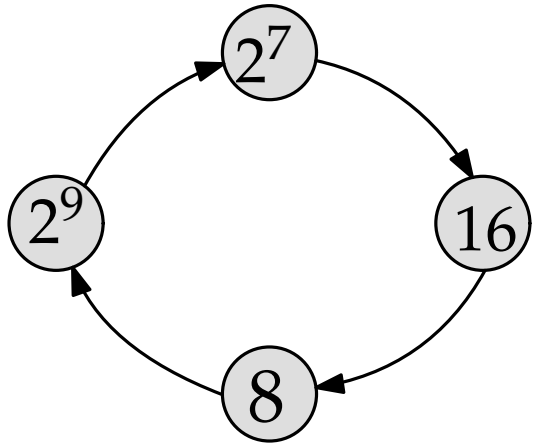


Finding a Local Optimum for $k = 1$

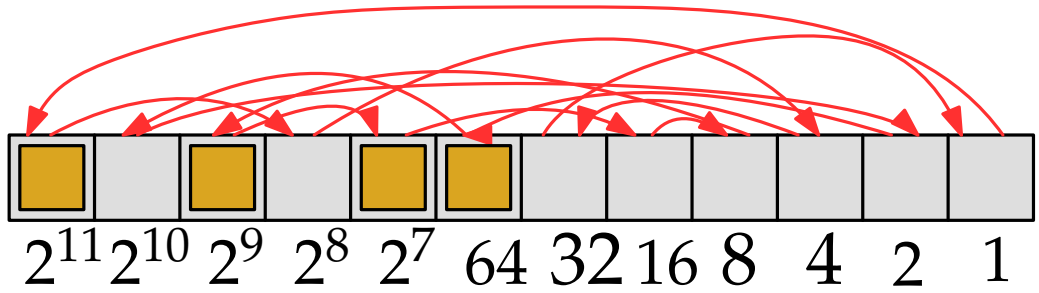


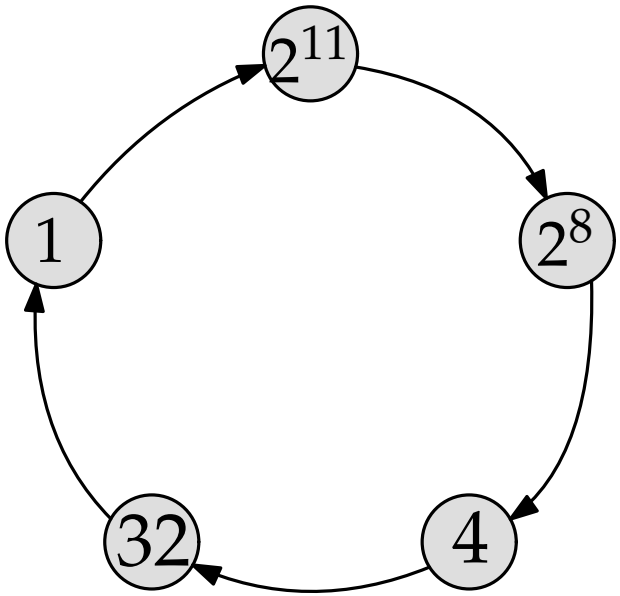


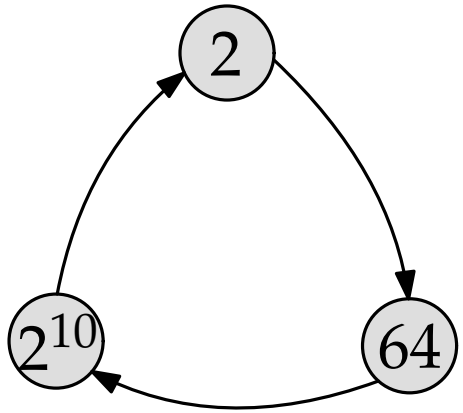


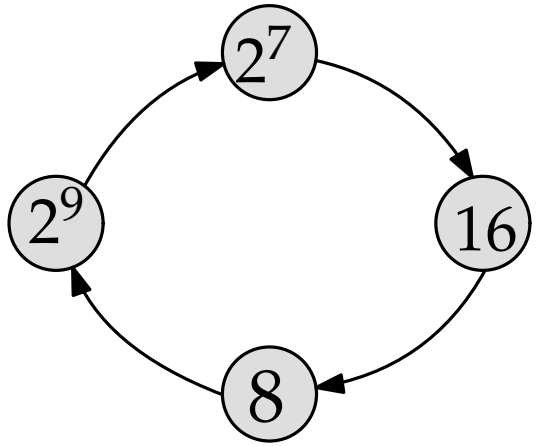


Finding a Local Optimum for $k = 1$

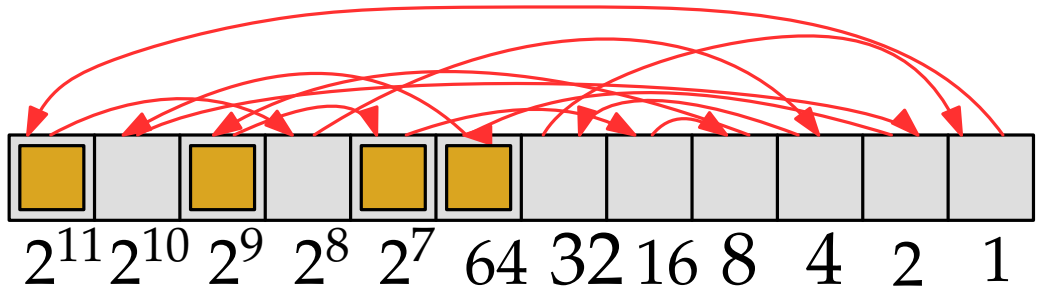


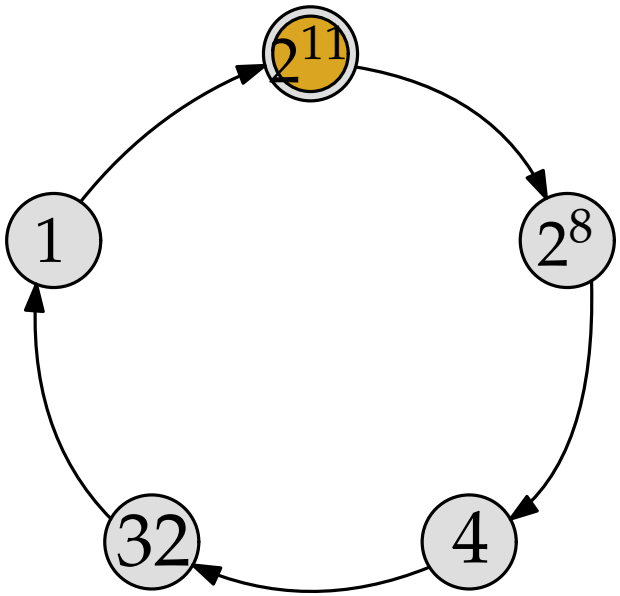


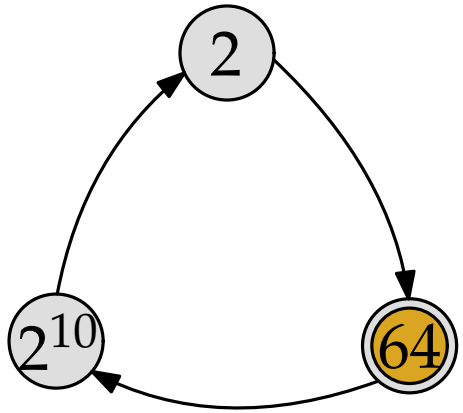


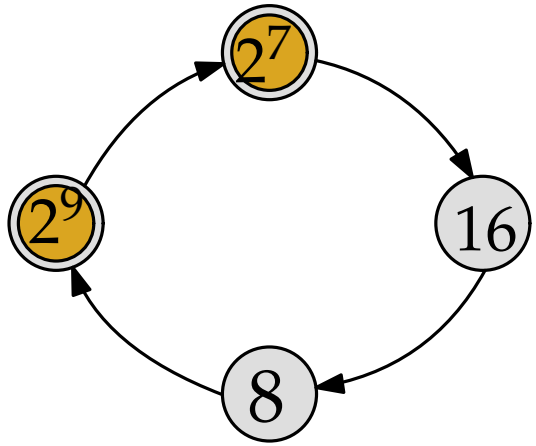


Finding a Local Optimum for $k = 1$

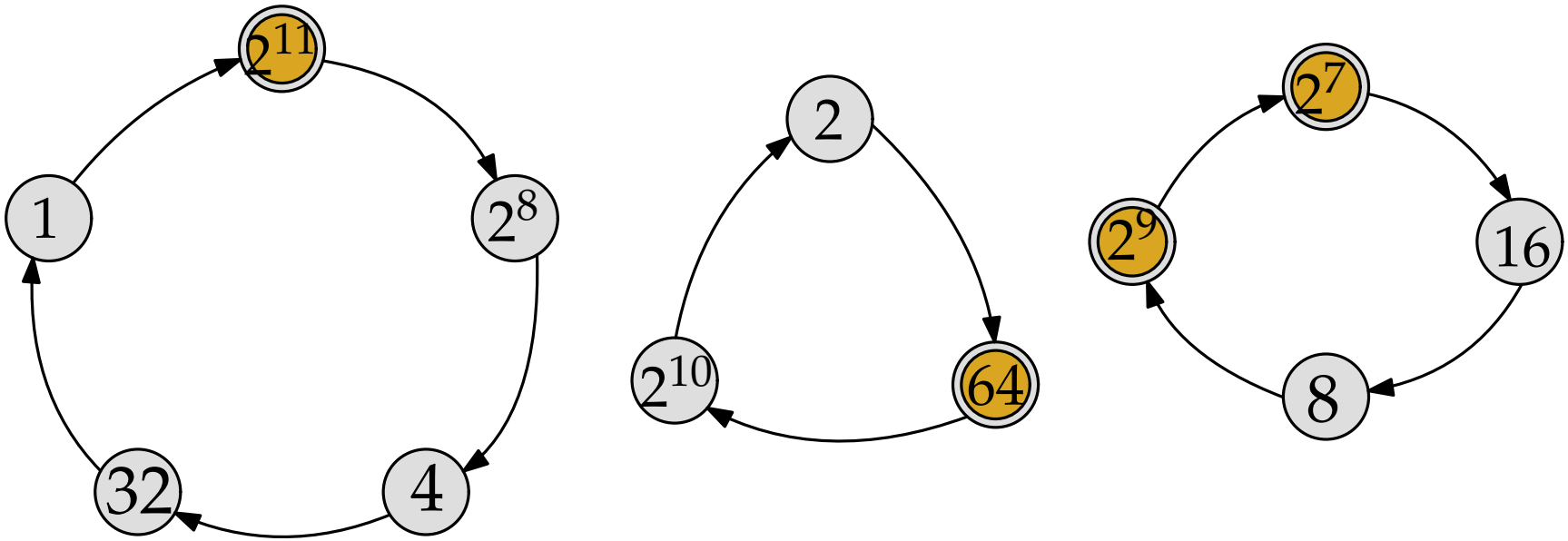




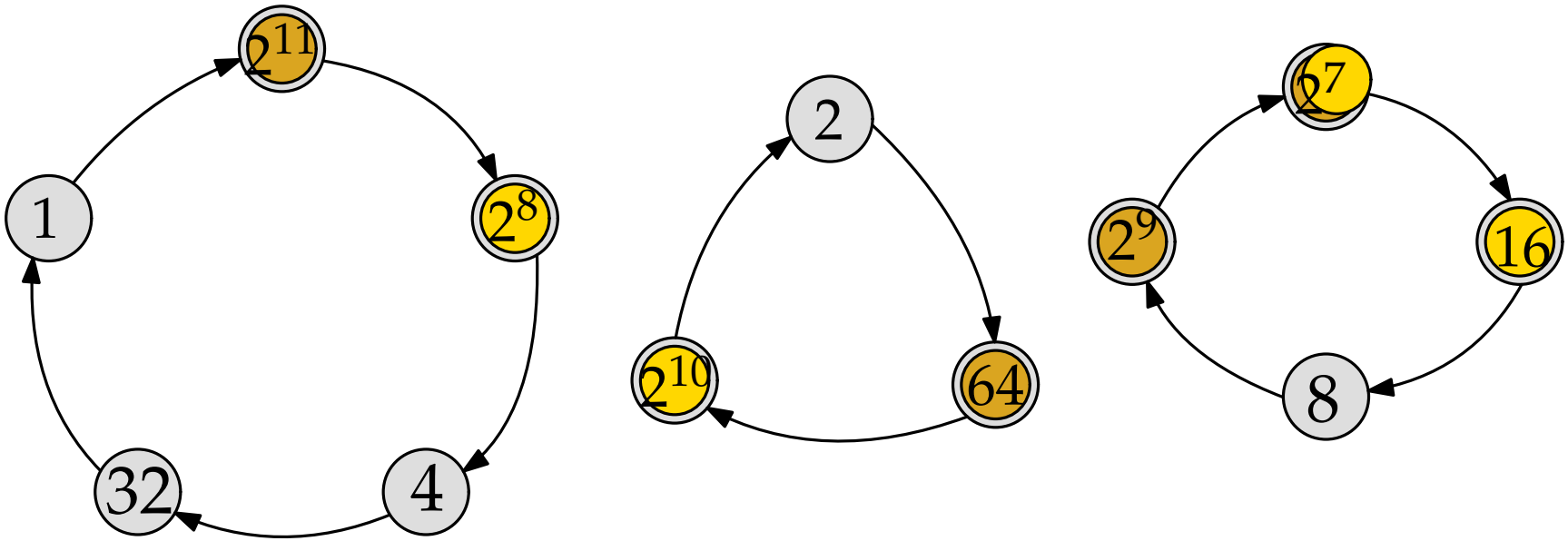




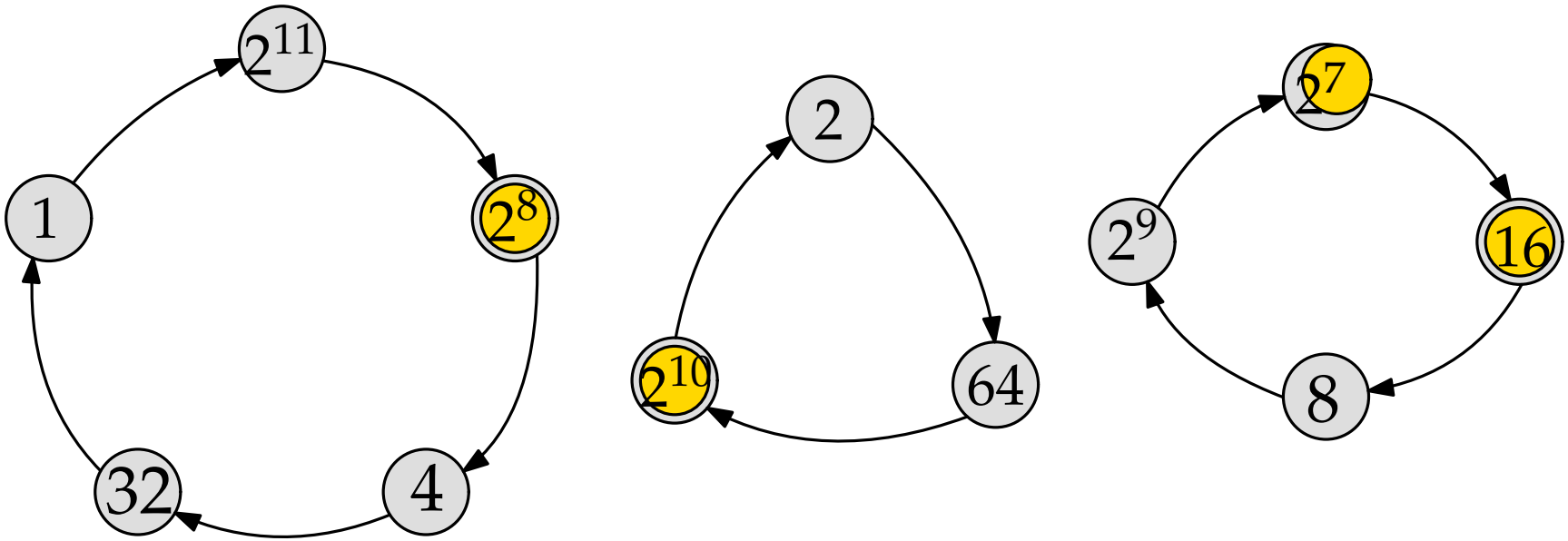
Finding a Local Optimum for $k = 1$



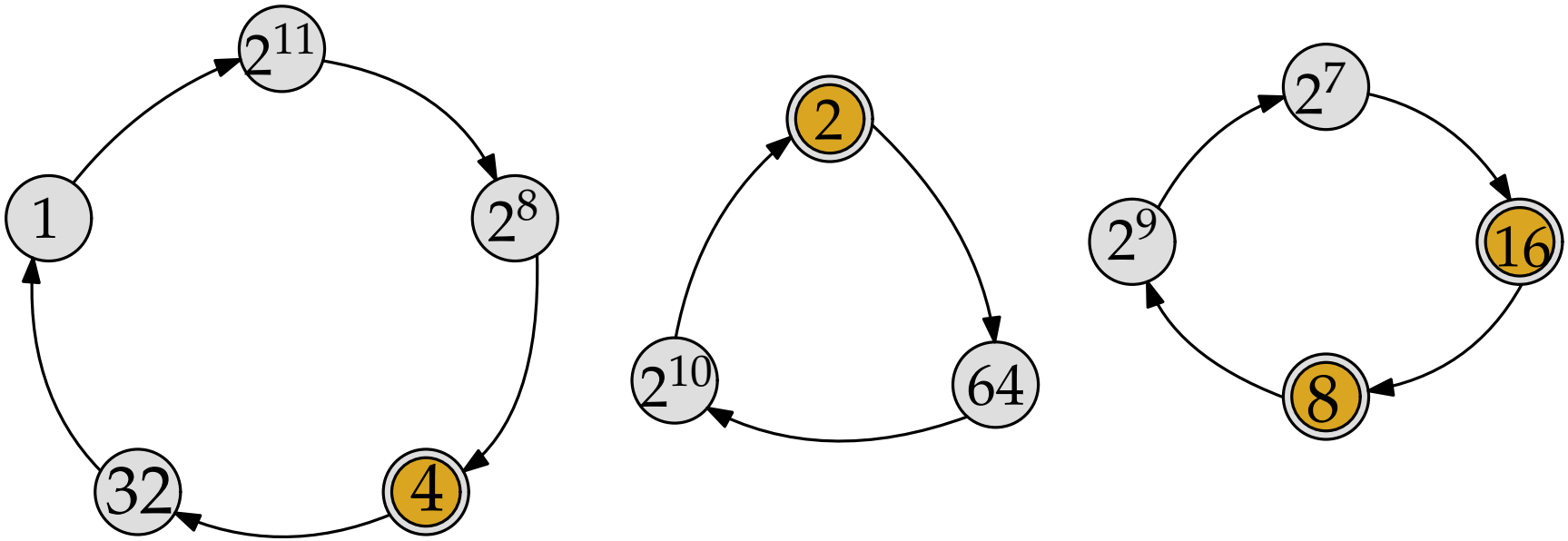
Finding a Local Optimum for $k = 1$



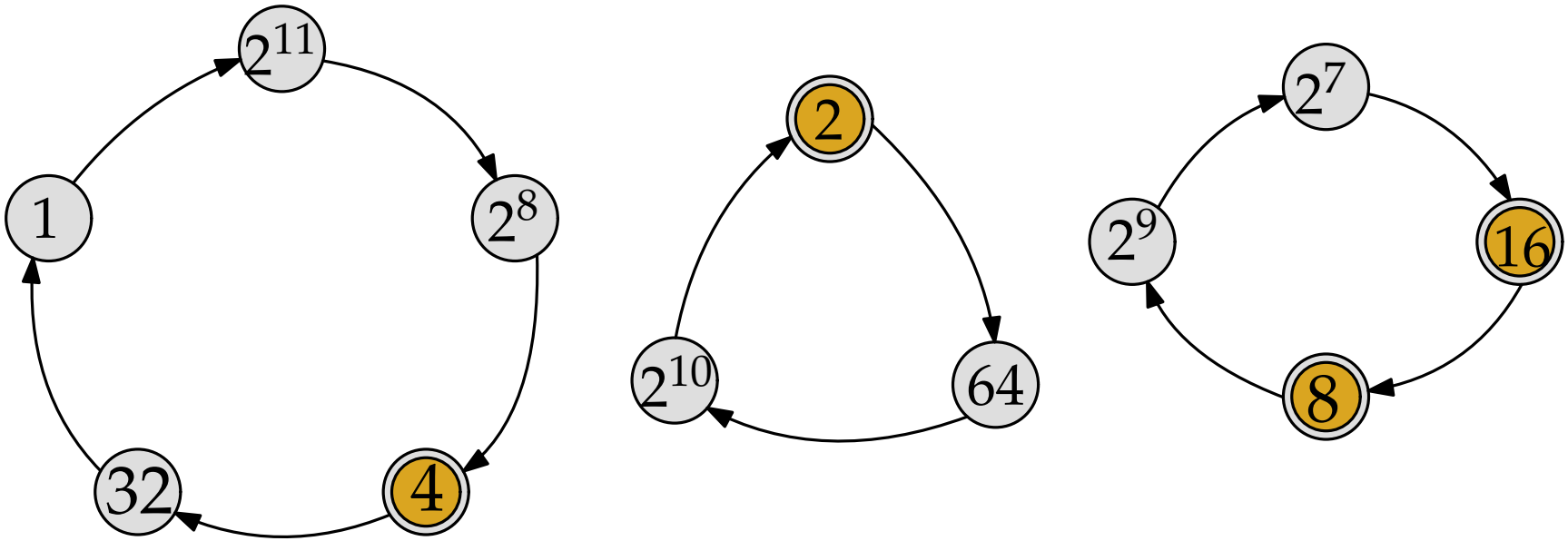
Finding a Local Optimum for $k = 1$



Finding a Local Optimum for $k = 1$

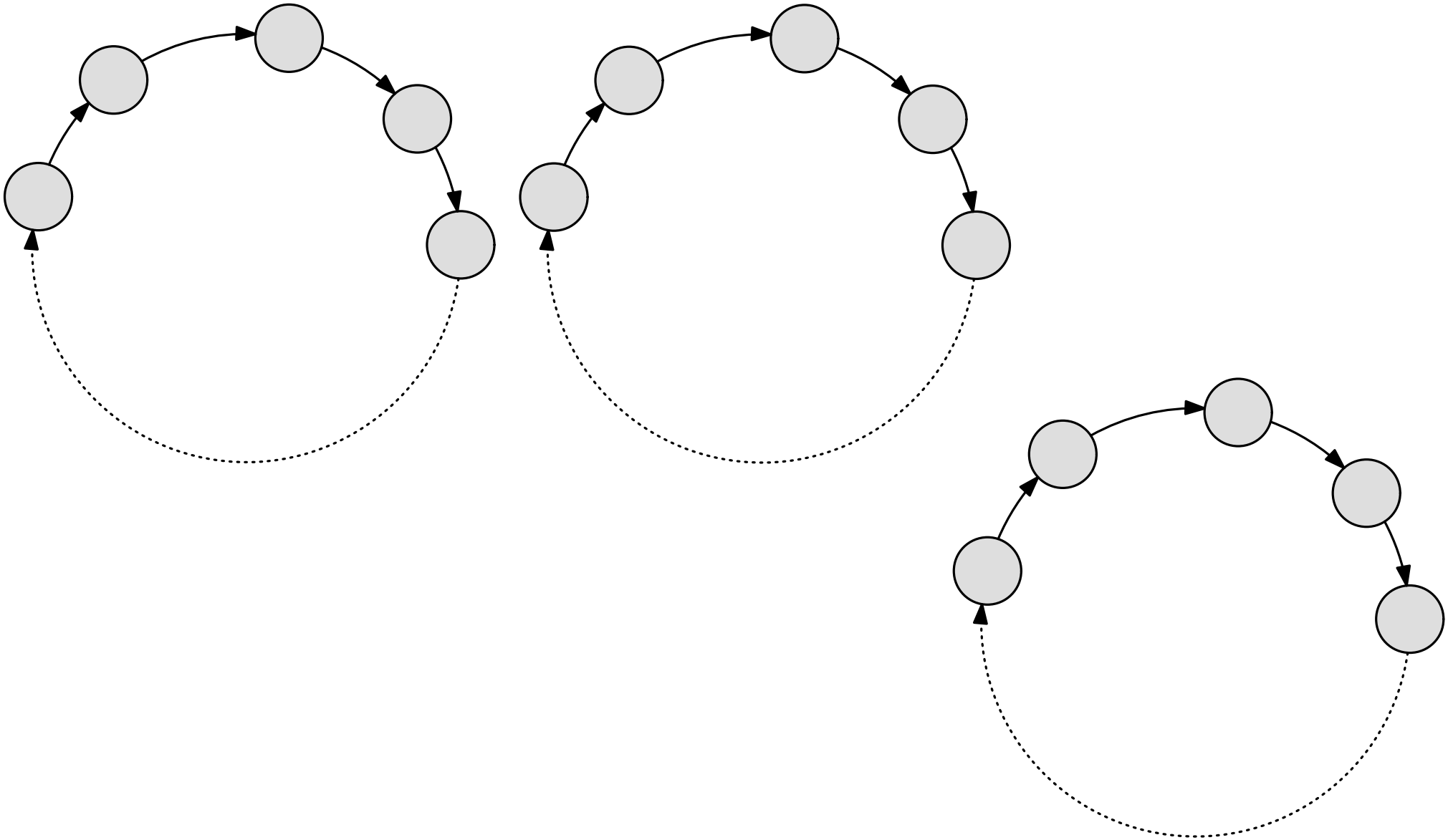


Finding a Local Optimum for $k = 1$

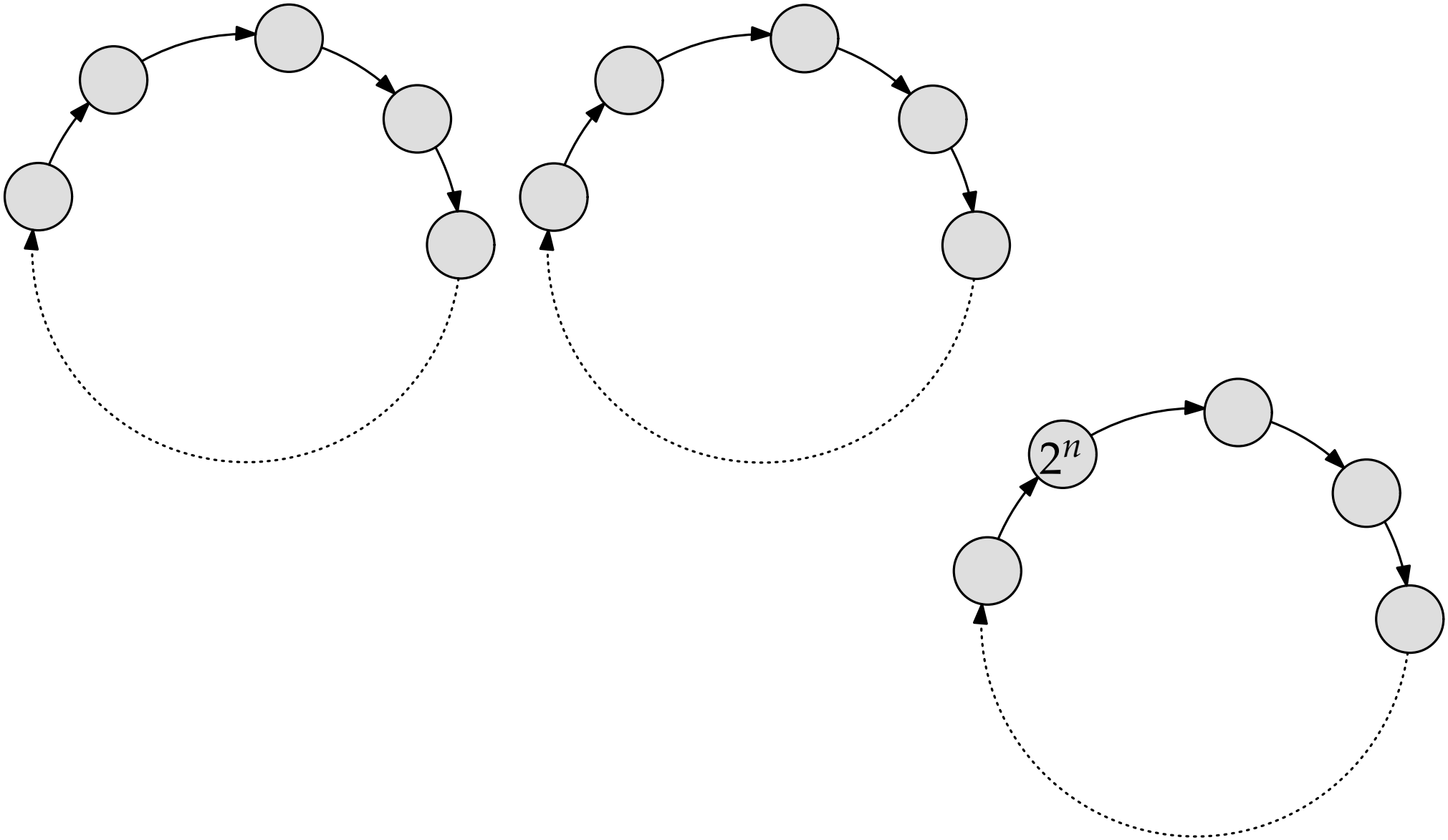


After two steps, the improving path has reached a local minimum

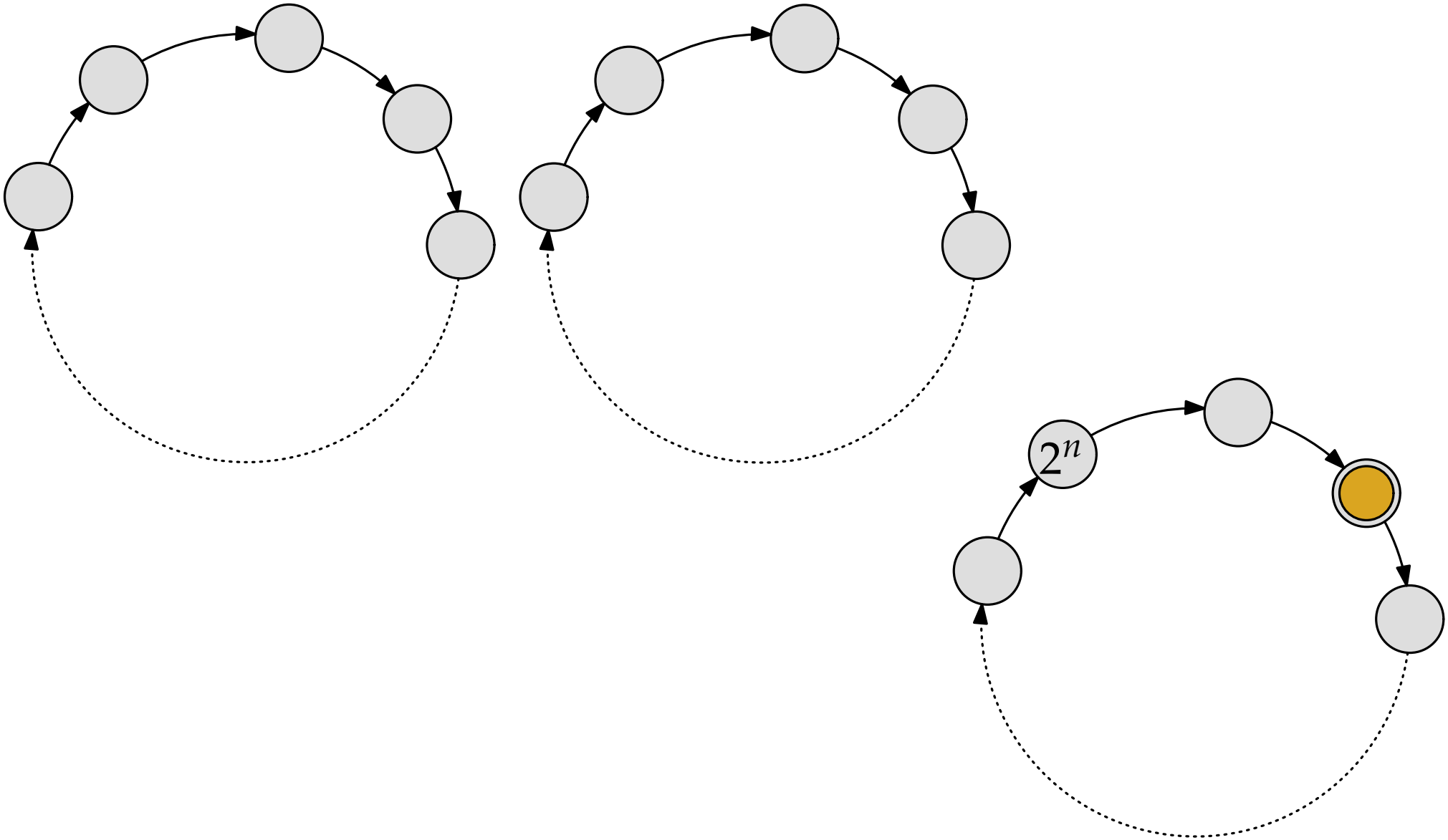
Finding a Local Optimum for $k = 1$



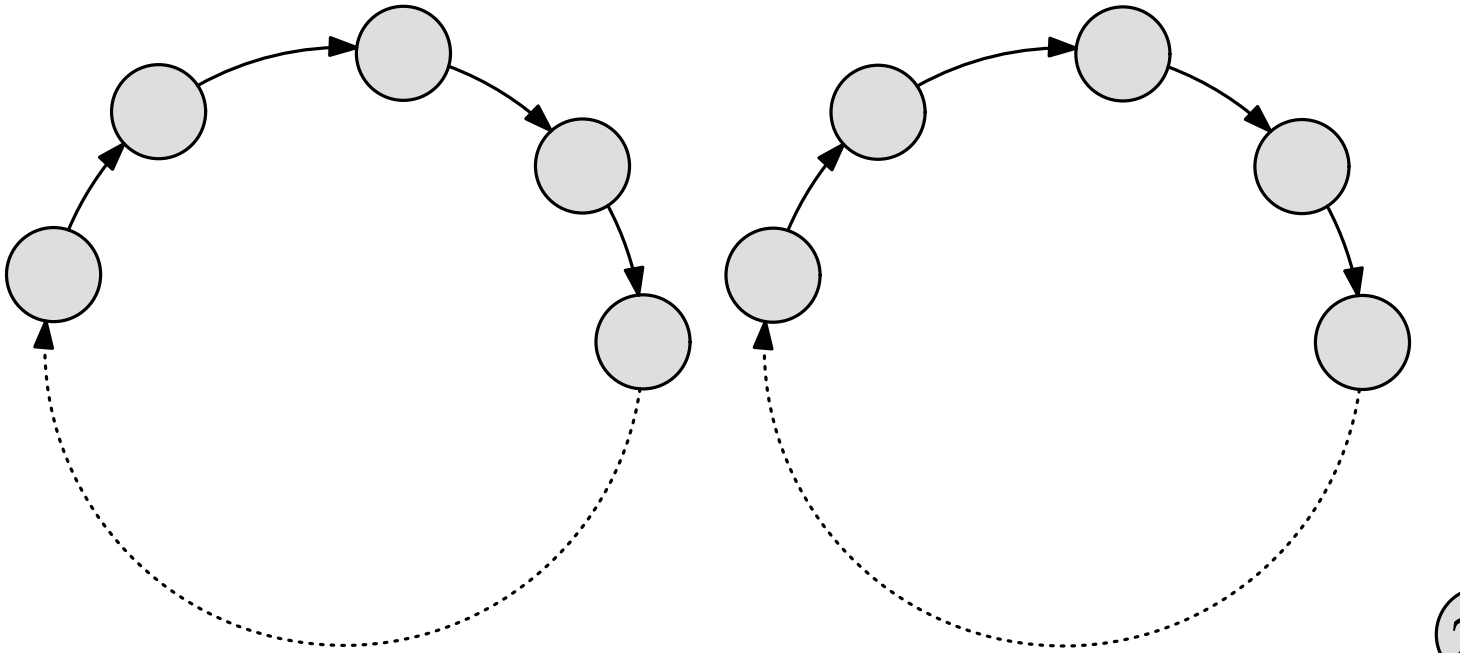
Finding a Local Optimum for $k = 1$

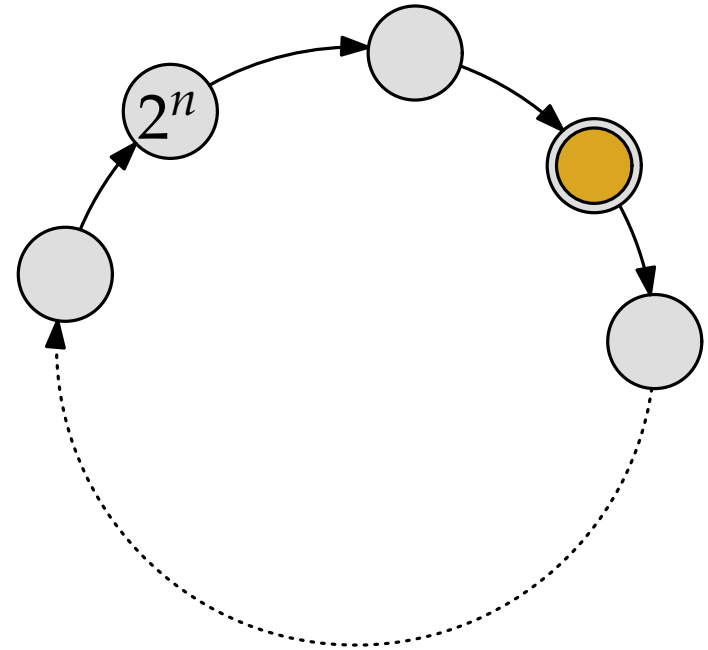


Finding a Local Optimum for $k = 1$



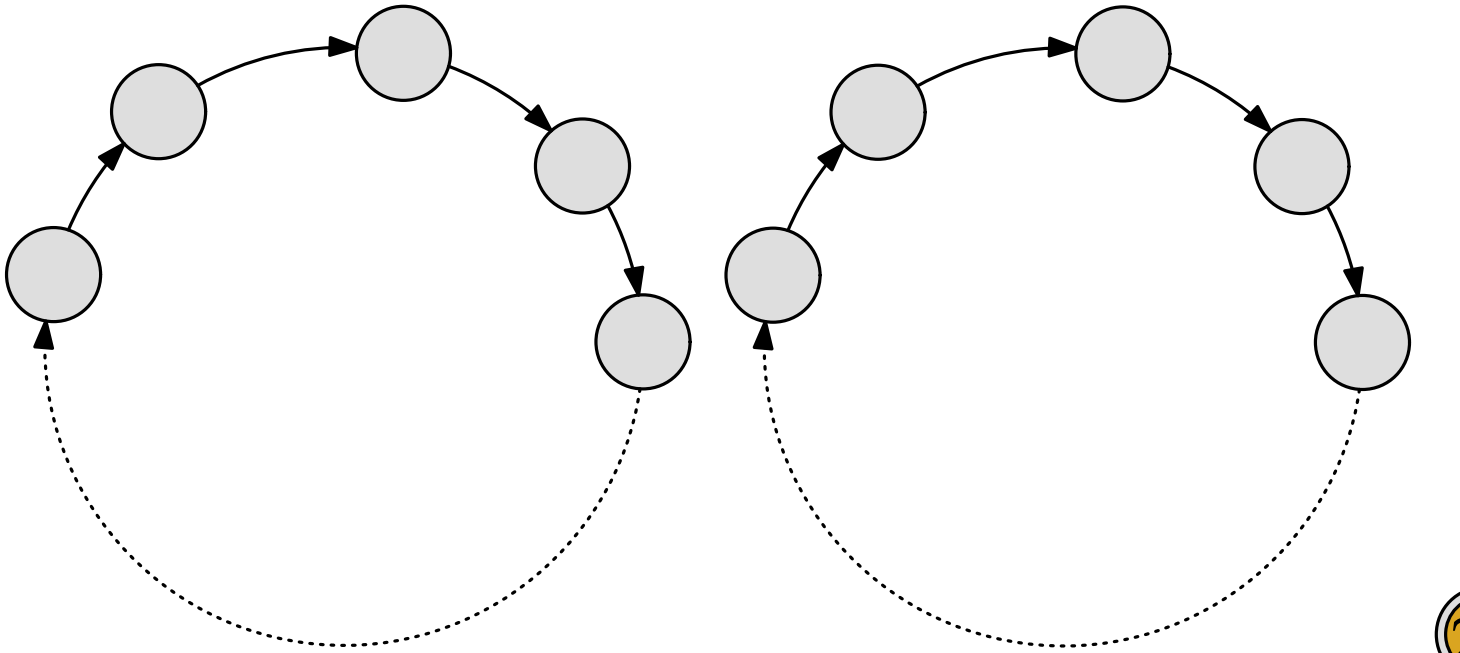
Finding a Local Optimum for $k = 1$

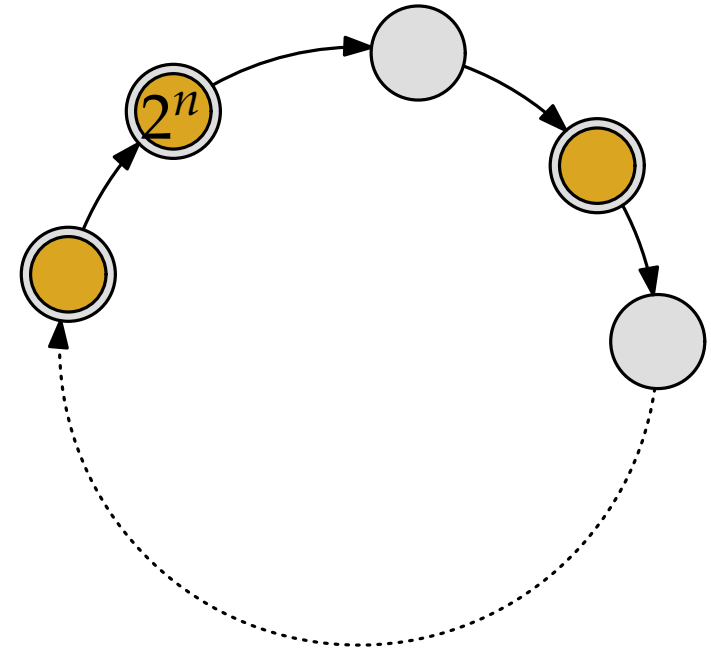




Within at most $n - 1$ steps, the 2^n-position will change from no-coin to yes-coin and the weight of the string $x \circ \pi$ will increase!

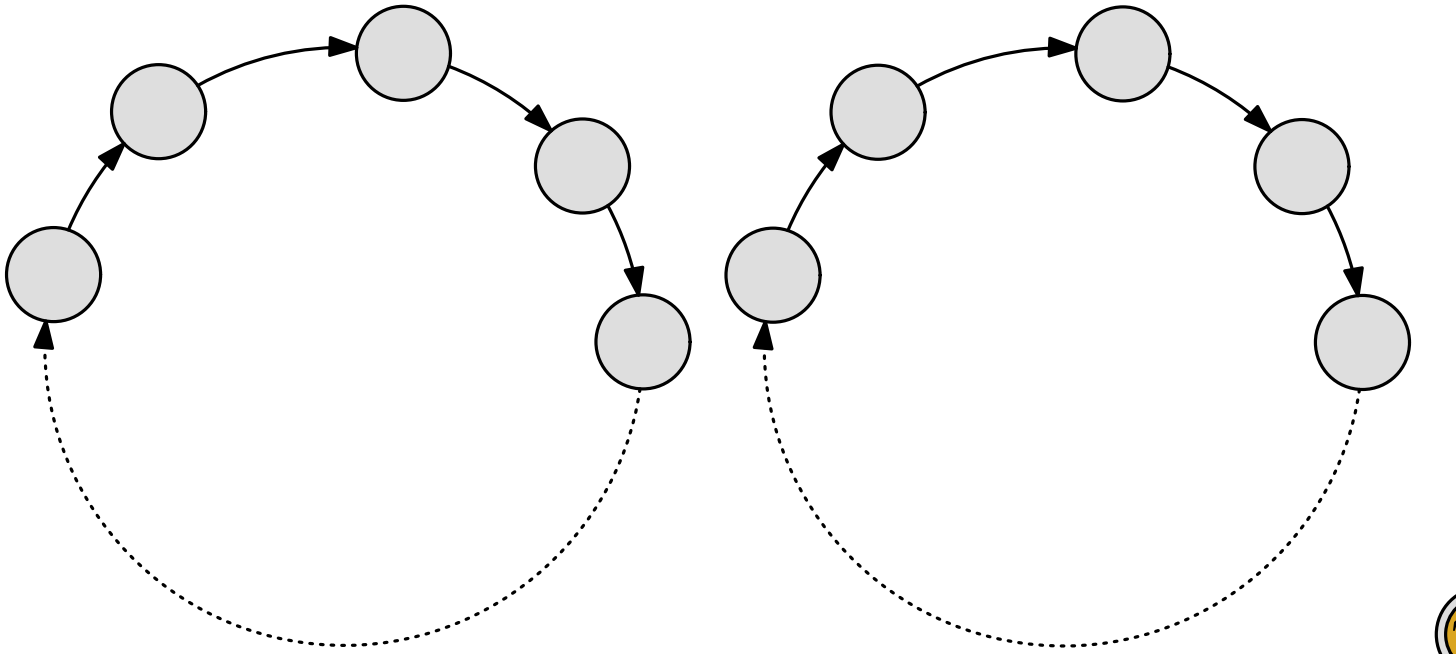
Finding a Local Optimum for $k = 1$

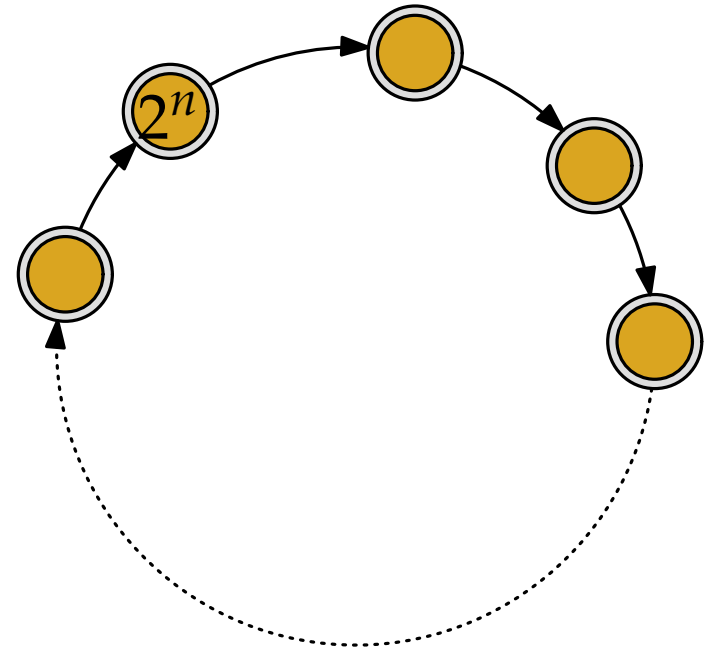




Within at most $n - 1$ steps, the 2^n-position will change from no-coin to yes-coin and the weight of the string $x \circ \pi$ will increase!

Finding a Local Optimum for $k = 1$

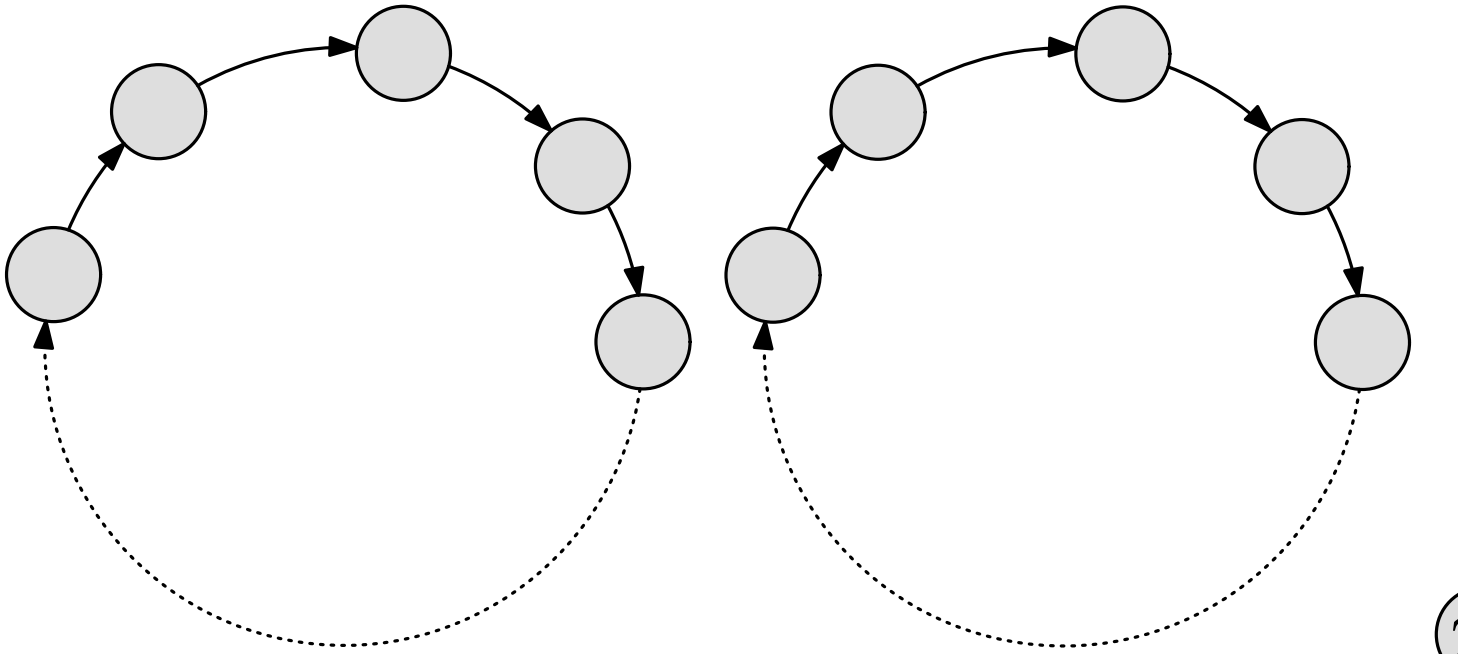


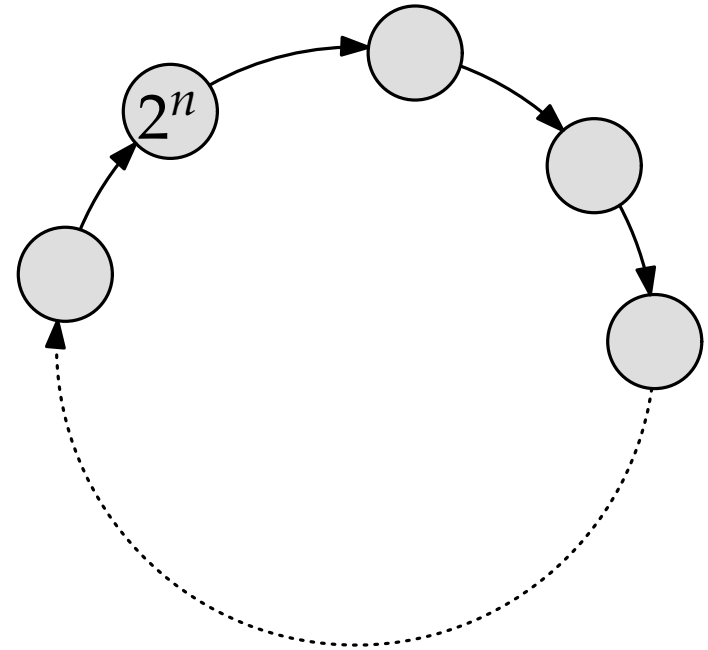


Within at most $n - 1$ steps, the 2^n-position will change from no-coin to yes-coin and the weight of the string $x \circ \pi$ will increase!

Except if cycles is all-coins

Finding a Local Optimum for $k = 1$



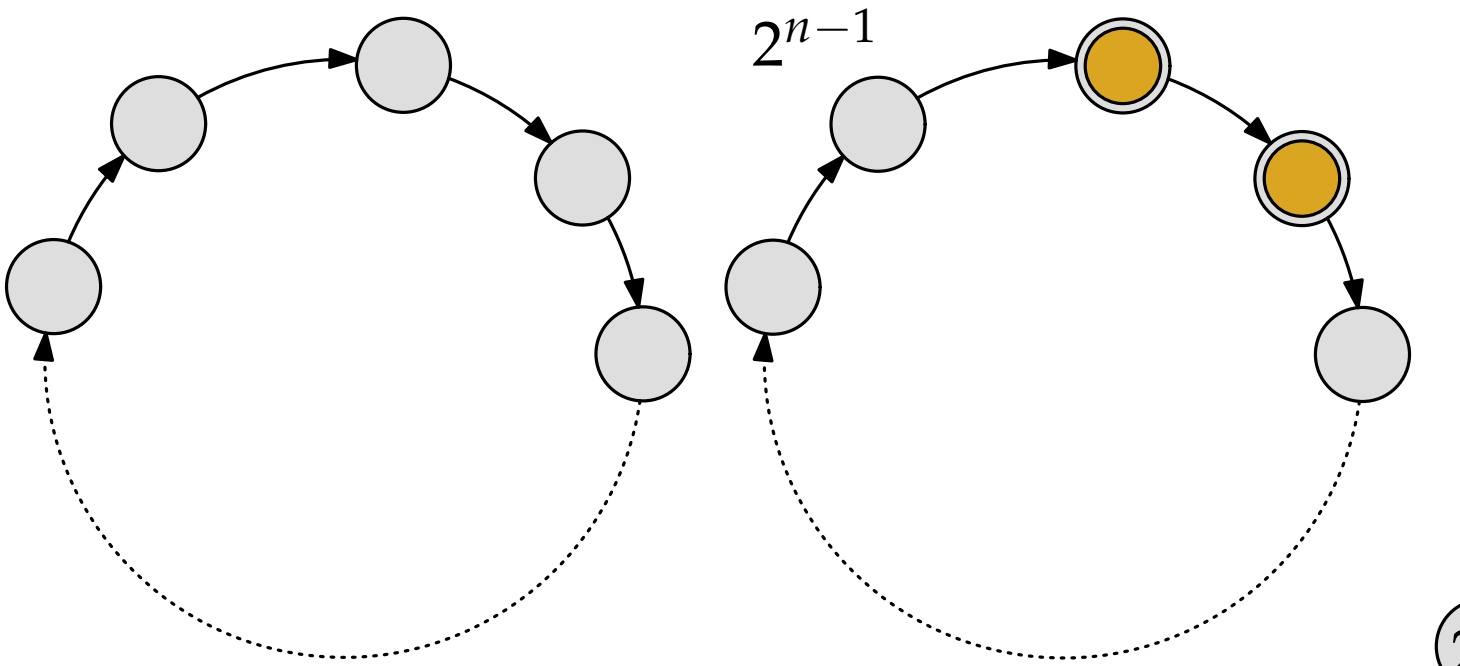


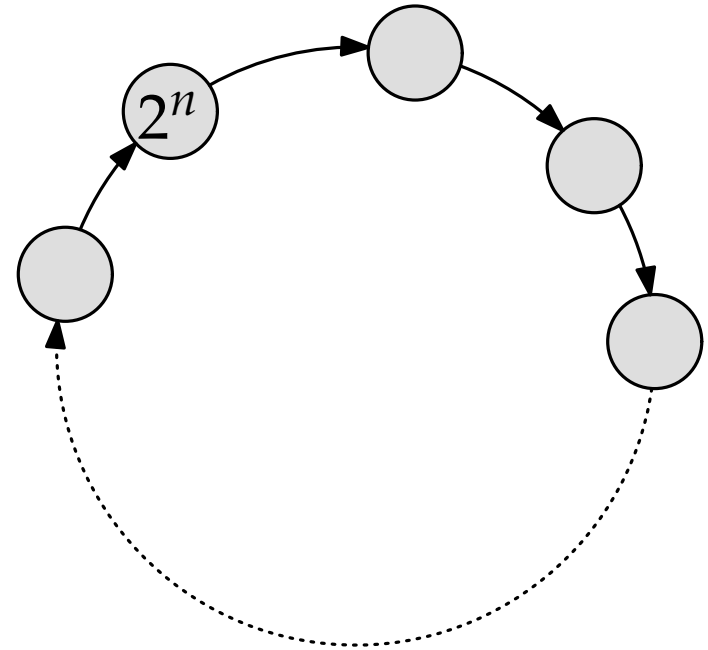
Within at most $n - 1$ steps, the 2^n-position will change from no-coin to yes-coin and the weight of the string $x \circ \pi$ will increase!

Except if cycles is all-coins

Or no-coins

Finding a Local Optimum for $k = 1$





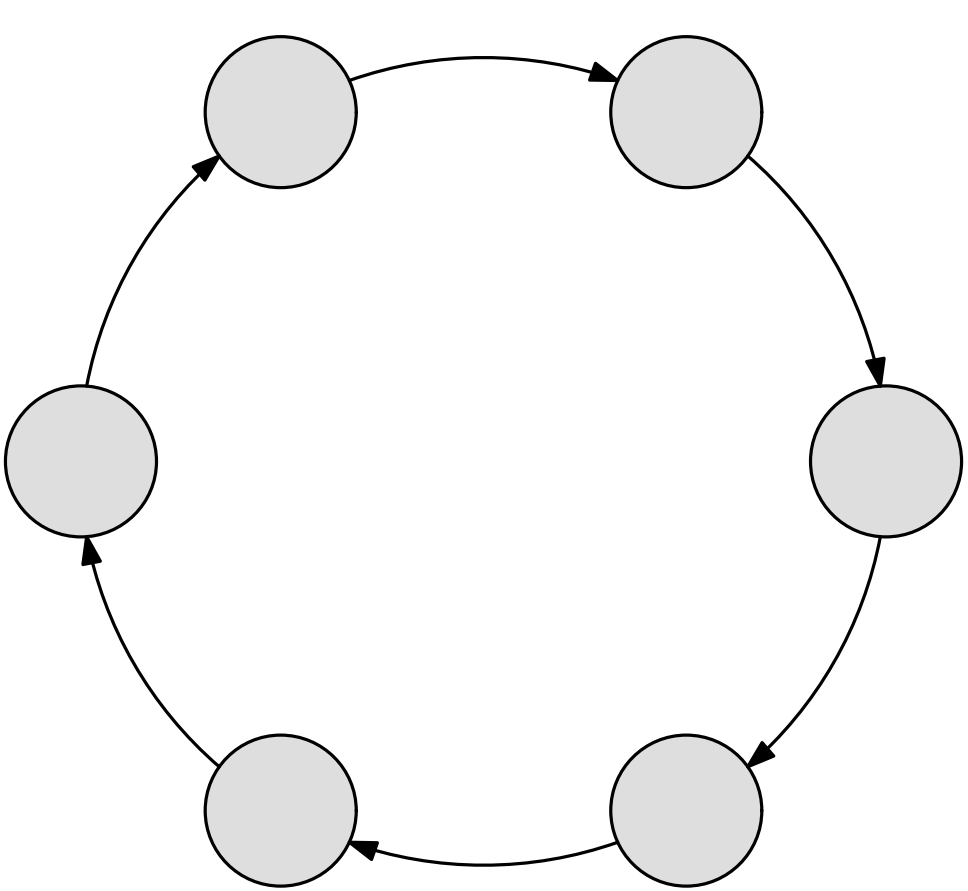
Within at most $n - 1$ steps, the 2^n-position will change from no-coin to yes-coin and the weight of the string $x \circ \pi$ will increase!

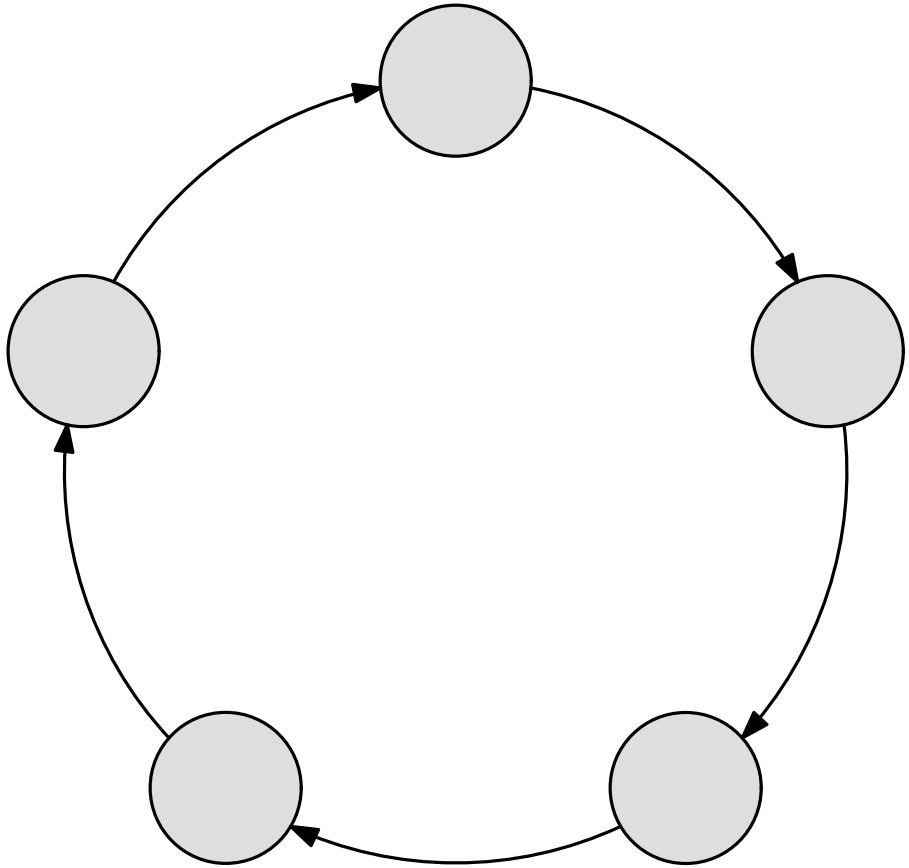
Except if cycles is all-coins

Or no-coins

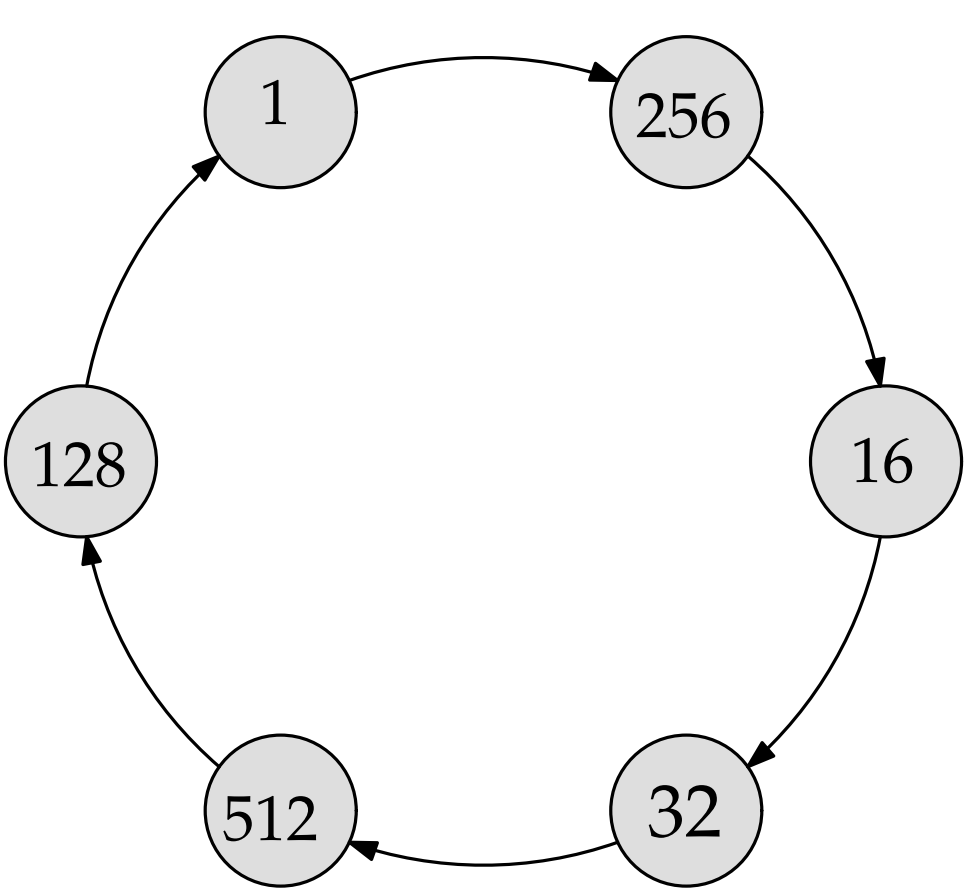
Then just look at second most important cycle.

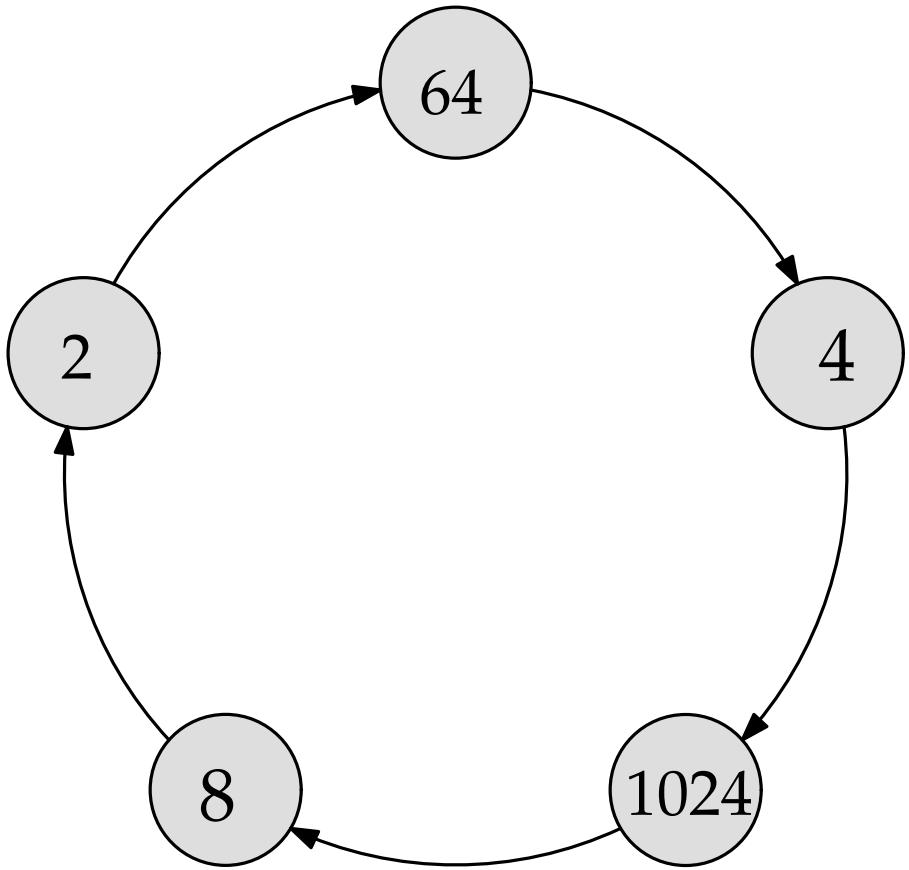
Finding a Local Optimum for $k = 1$



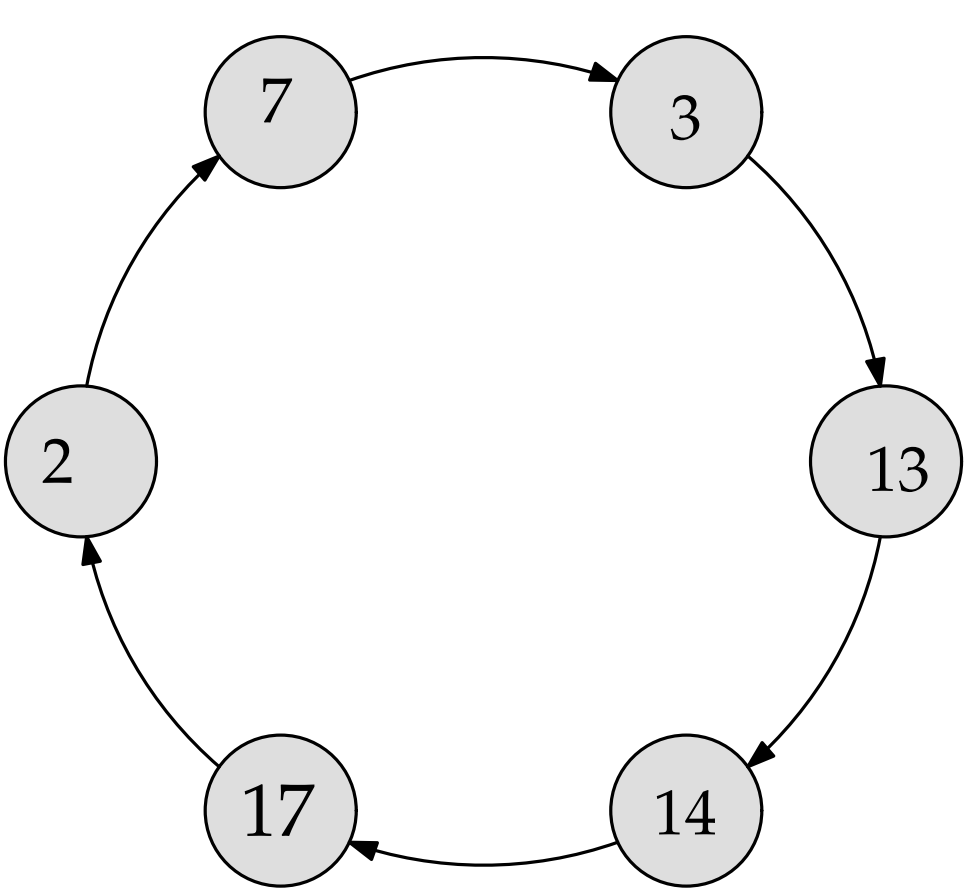


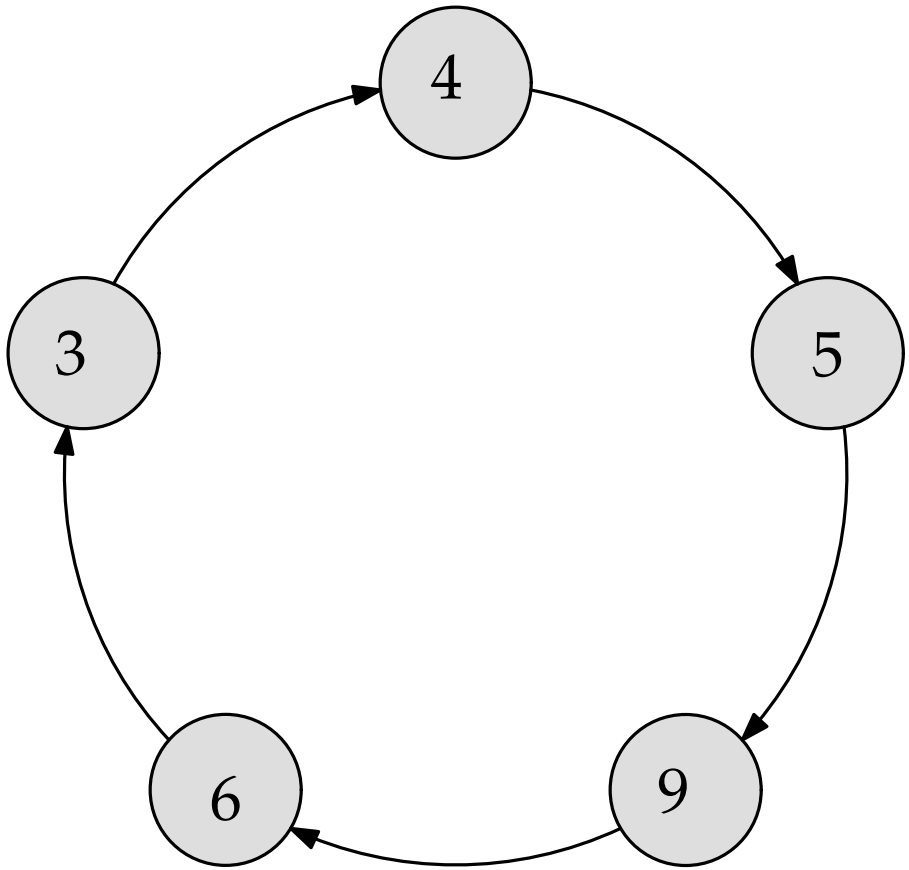
Finding a Local Optimum for $k = 1$



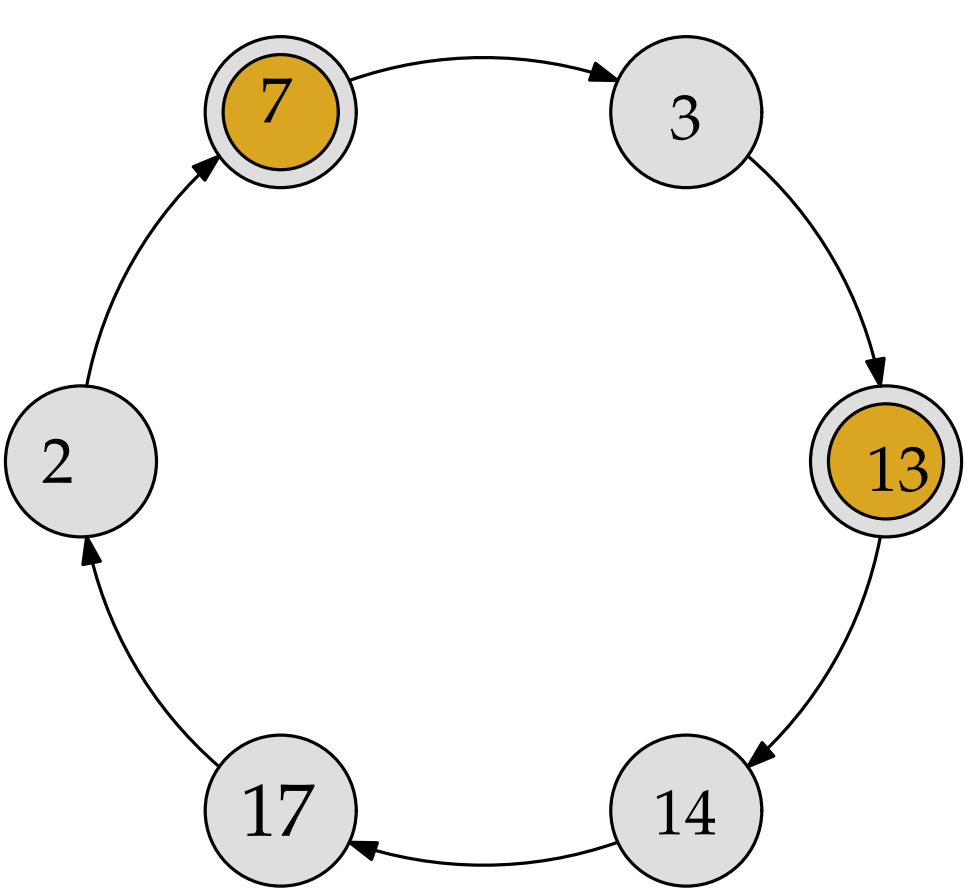


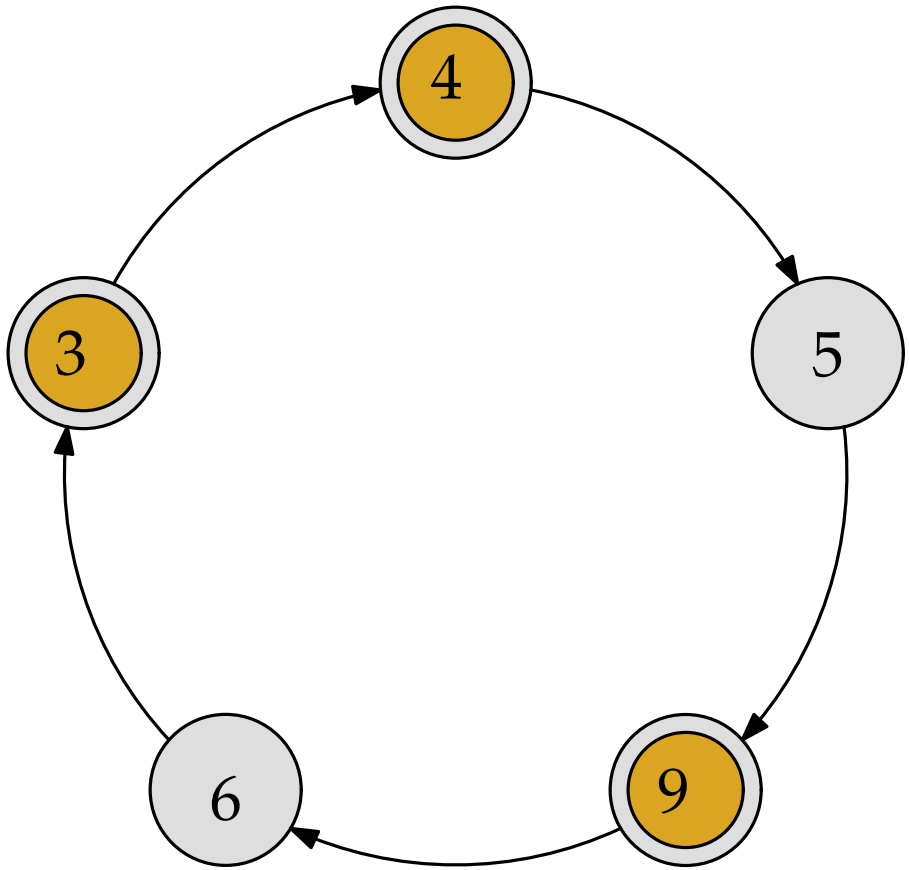
Finding a Local Optimum for $k = 1$



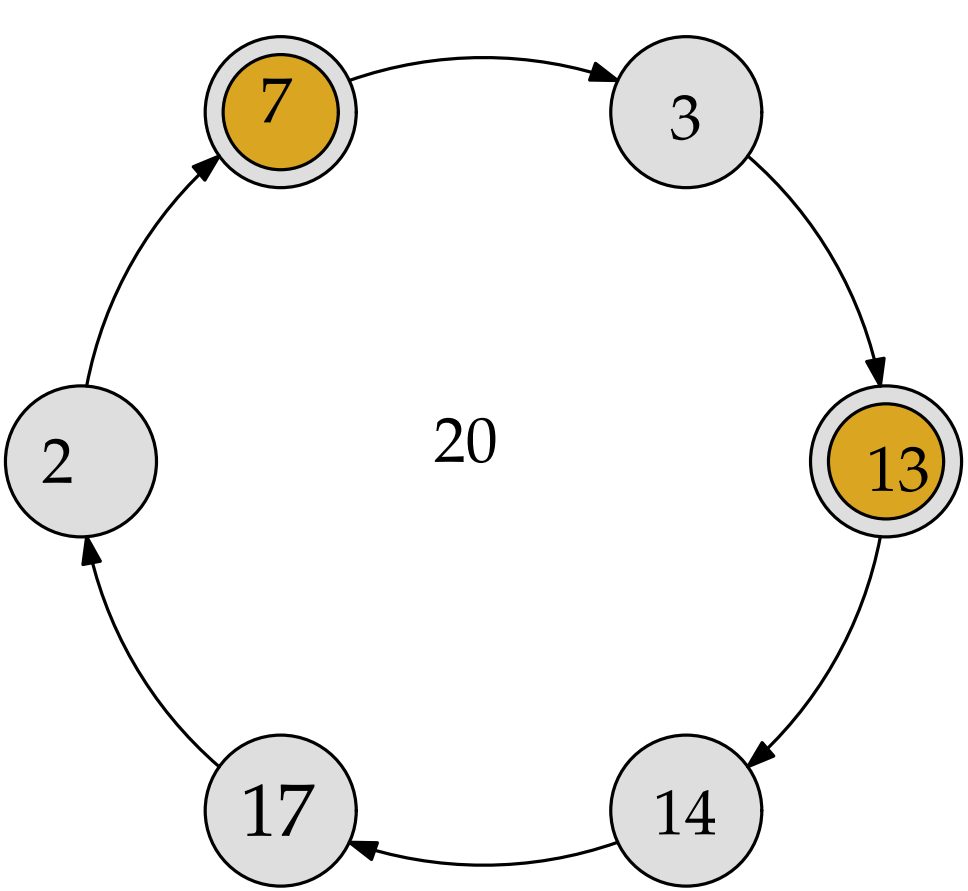


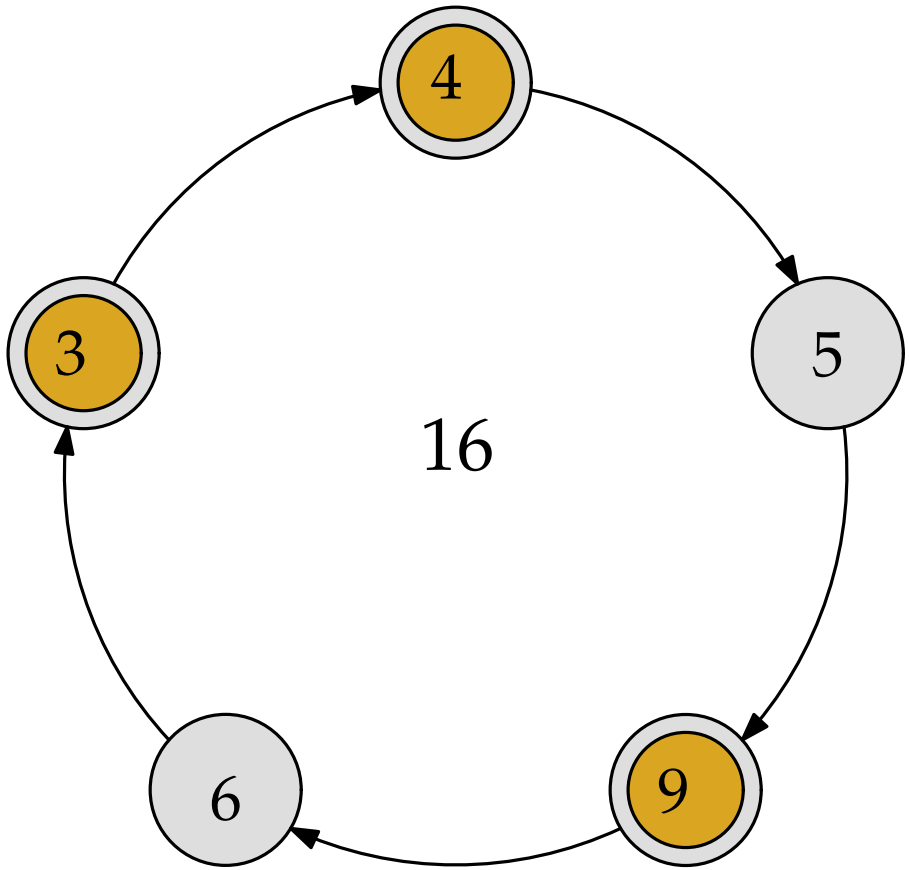
Finding a Local Optimum for $k = 1$



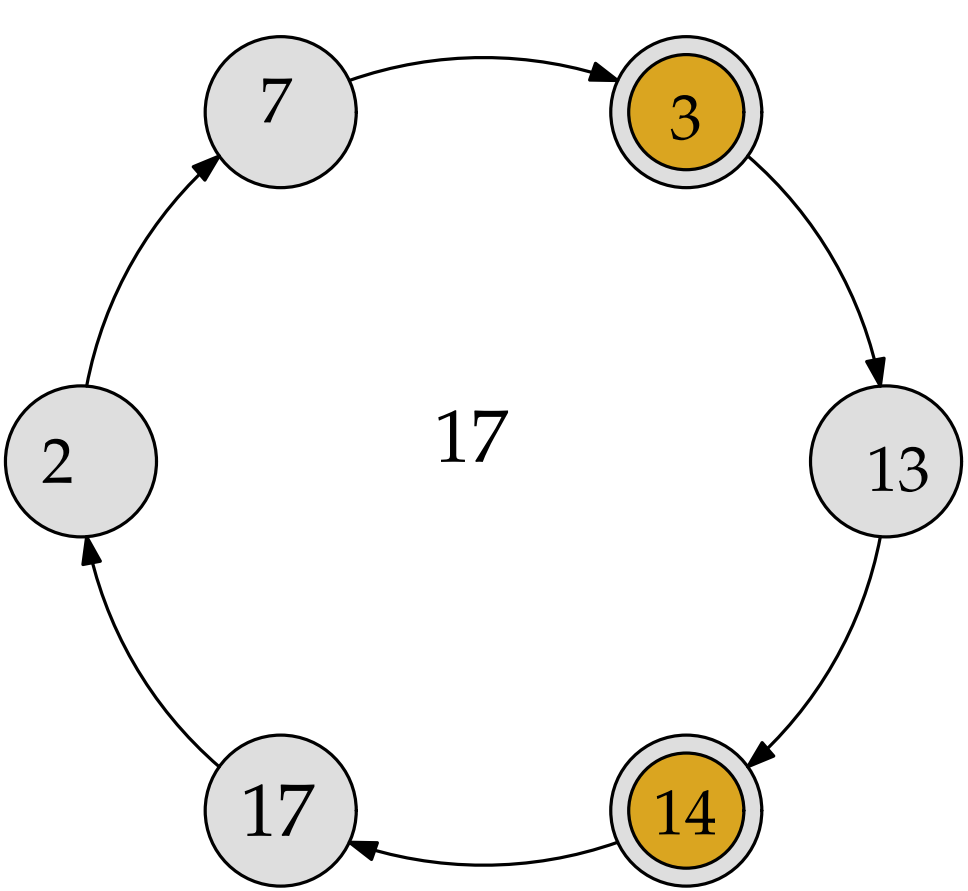


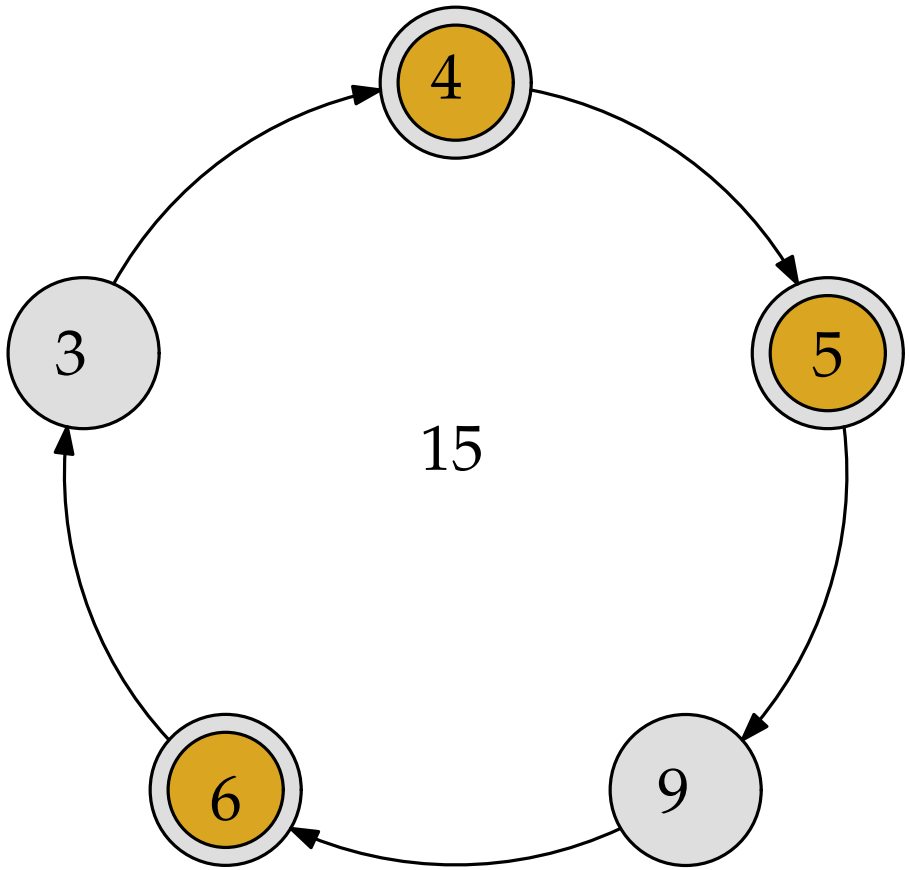
Finding a Local Optimum for $k = 1$



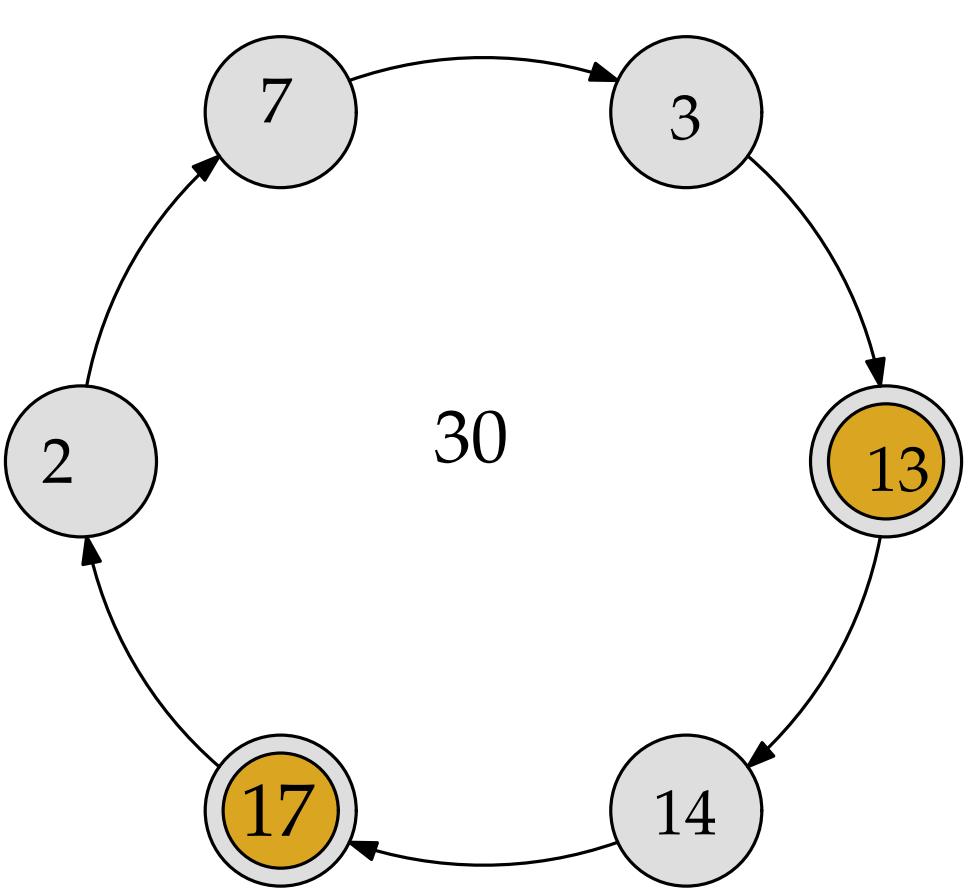


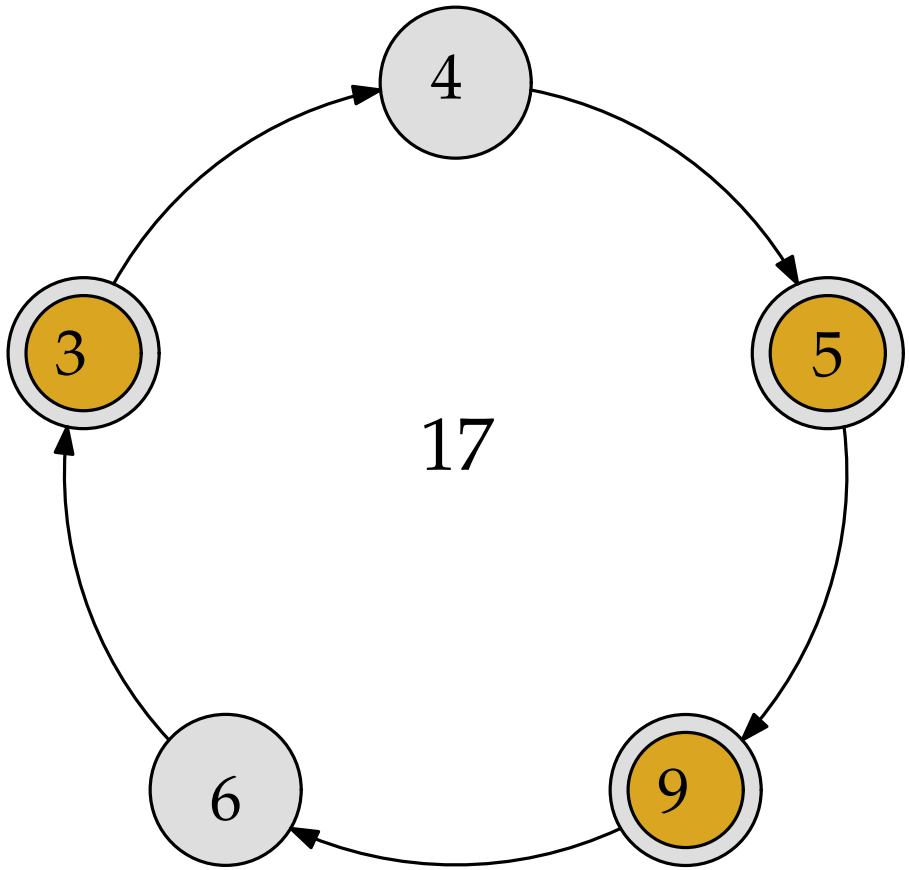
Finding a Local Optimum for $k = 1$



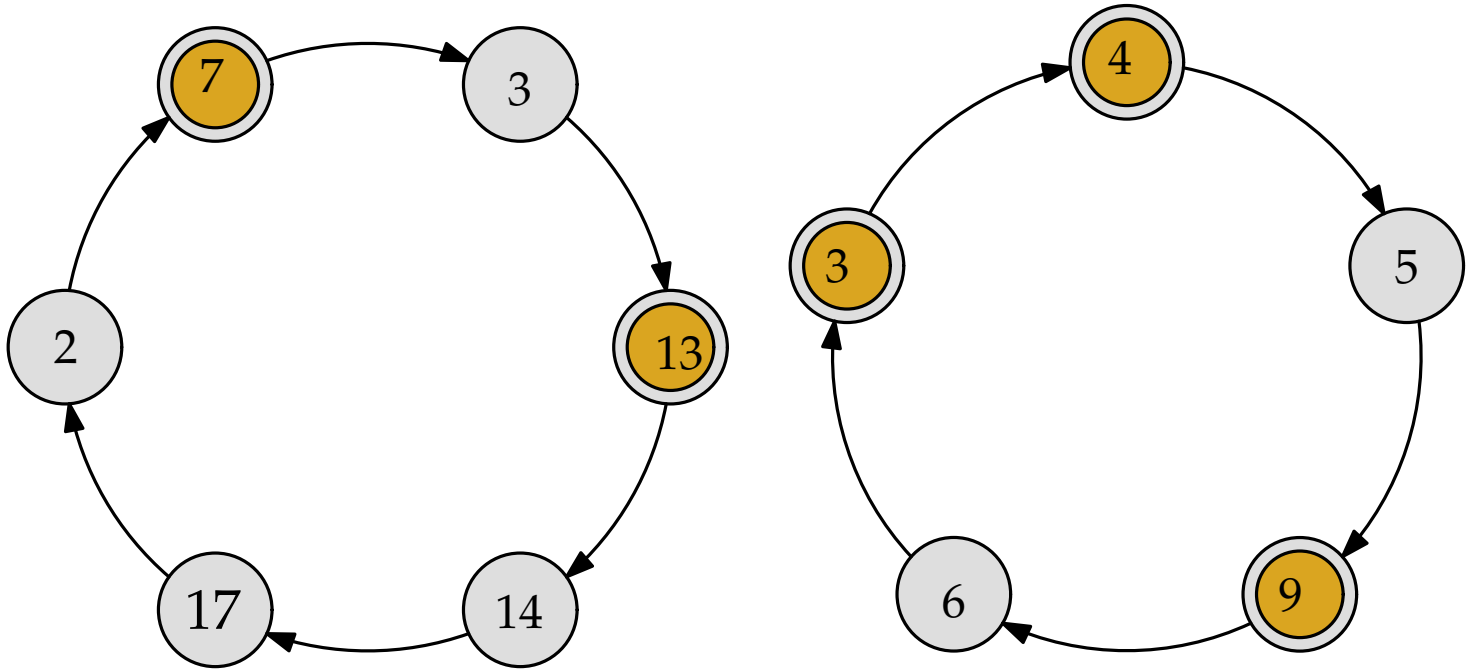


Finding a Local Optimum for $k = 1$

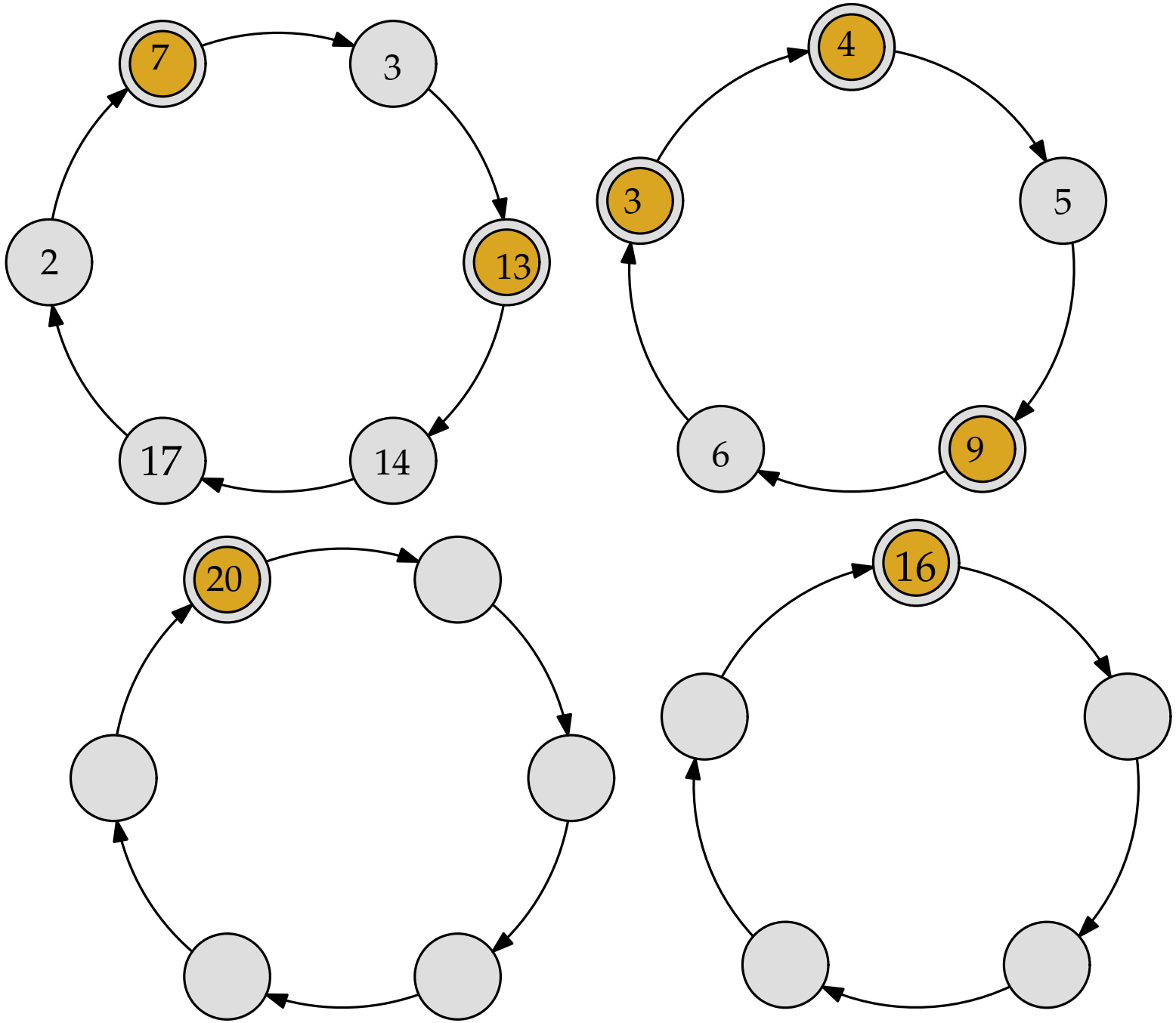




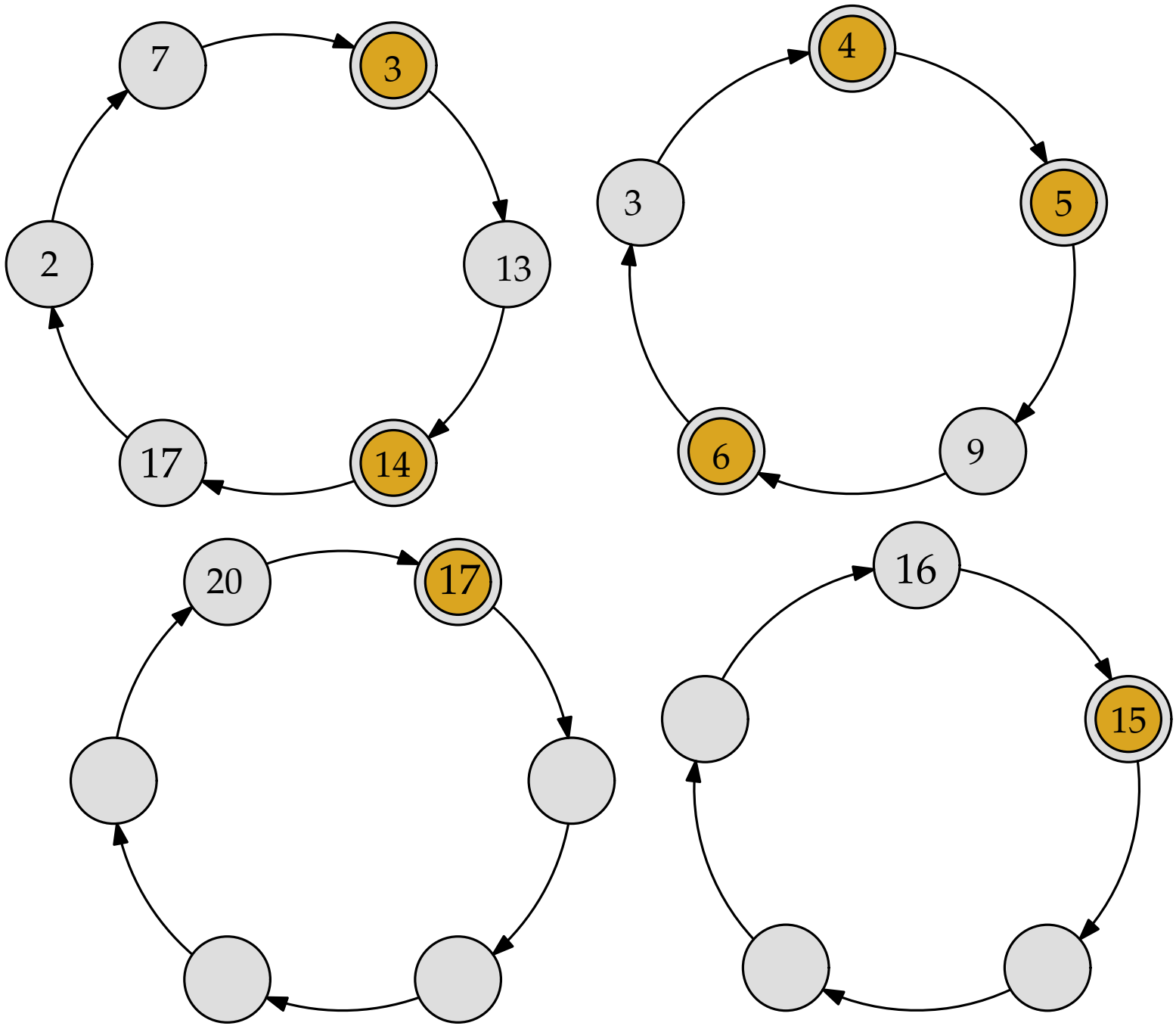
Finding a Local Optimum for $k = 1$



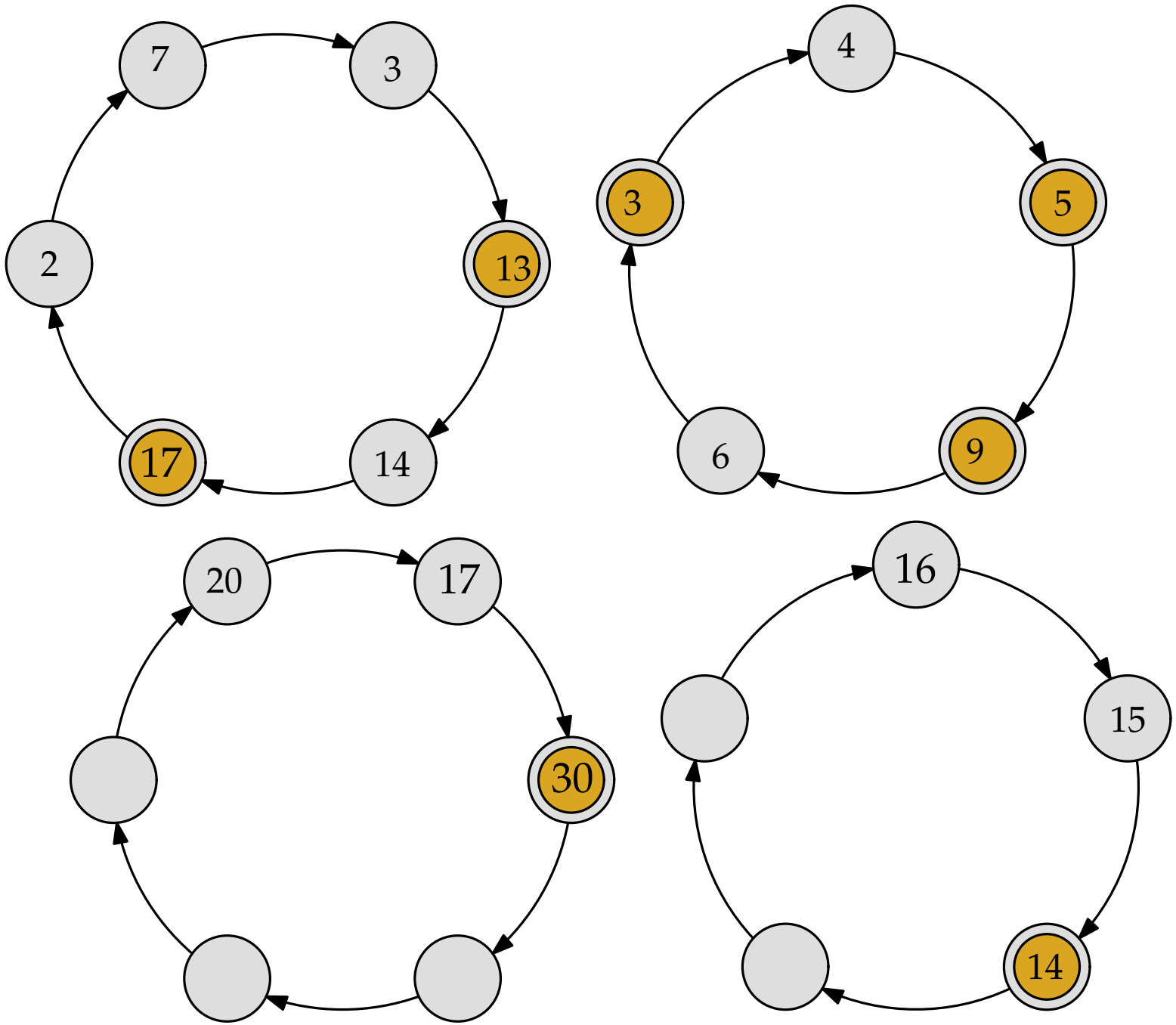
Finding a Local Optimum for $k = 1$



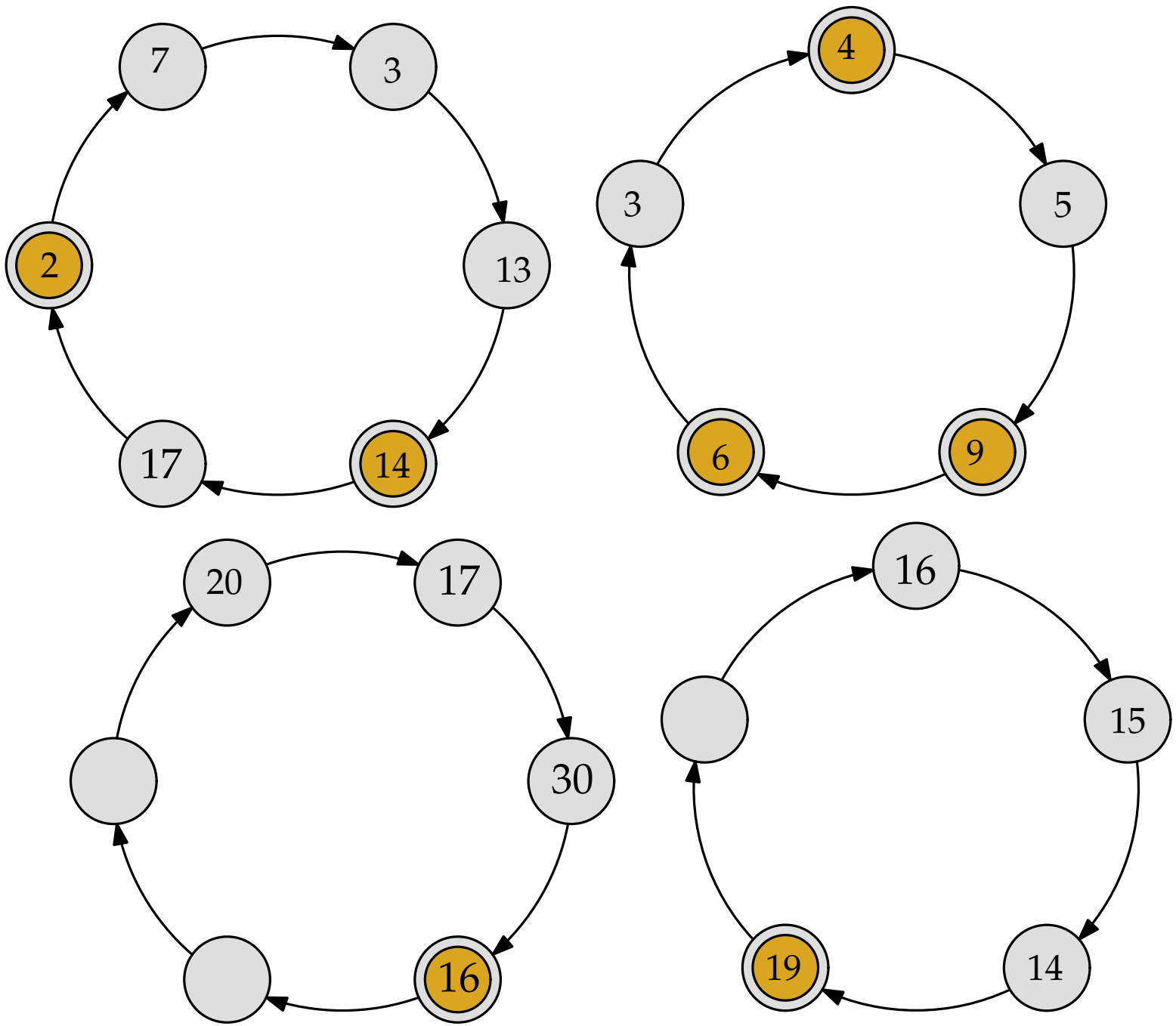
Finding a Local Optimum for $k = 1$



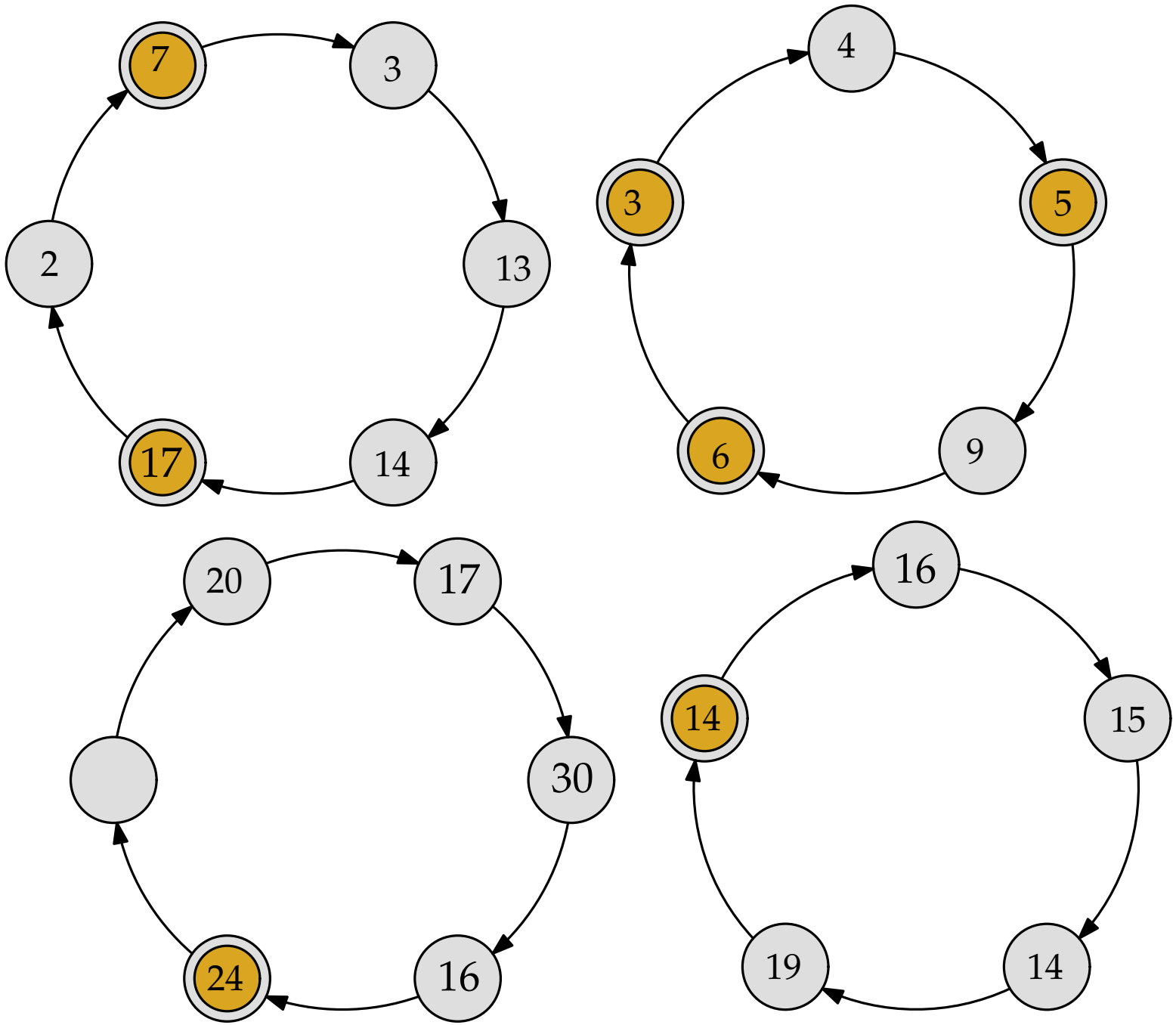
Finding a Local Optimum for $k = 1$



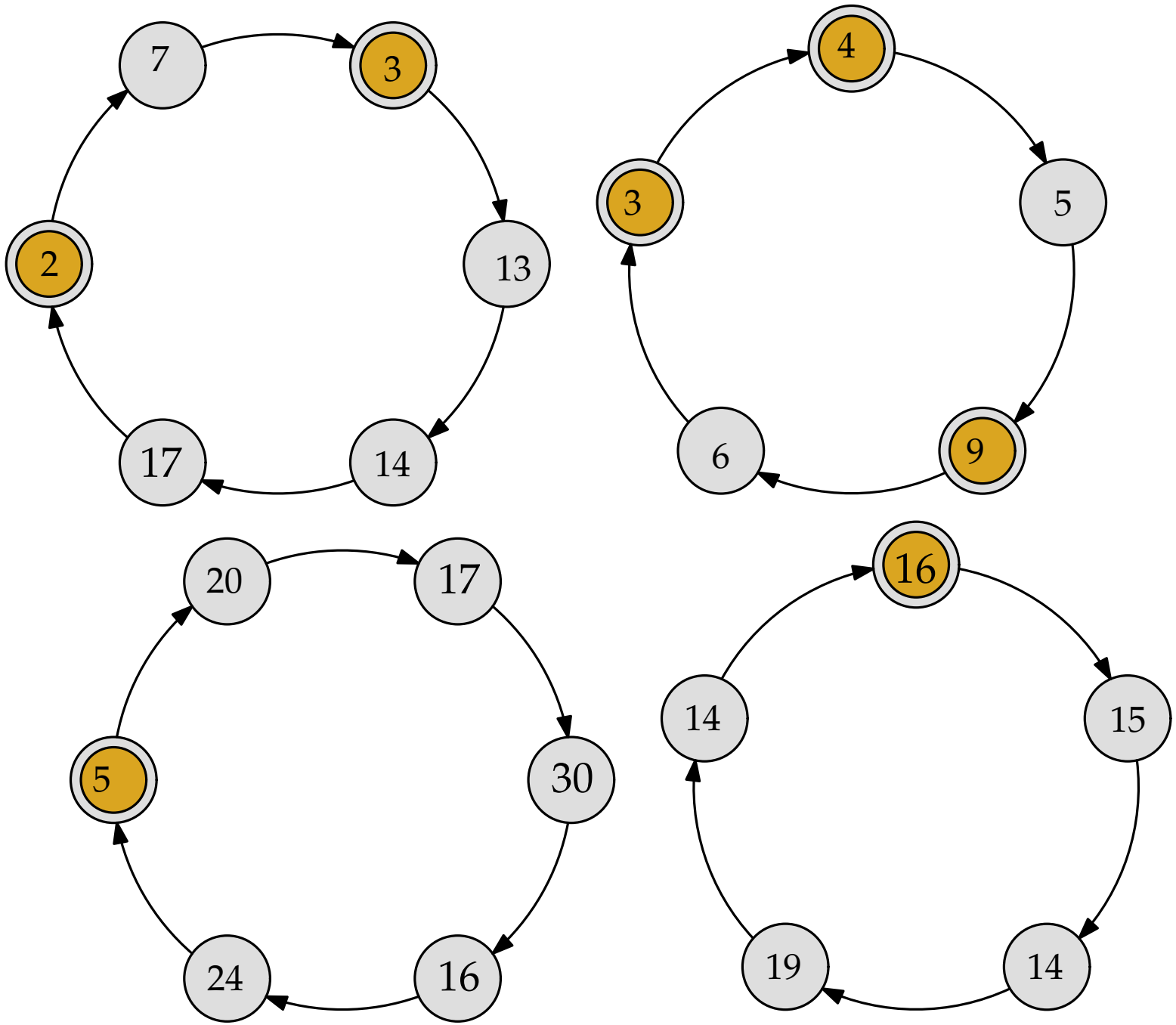
Finding a Local Optimum for $k = 1$



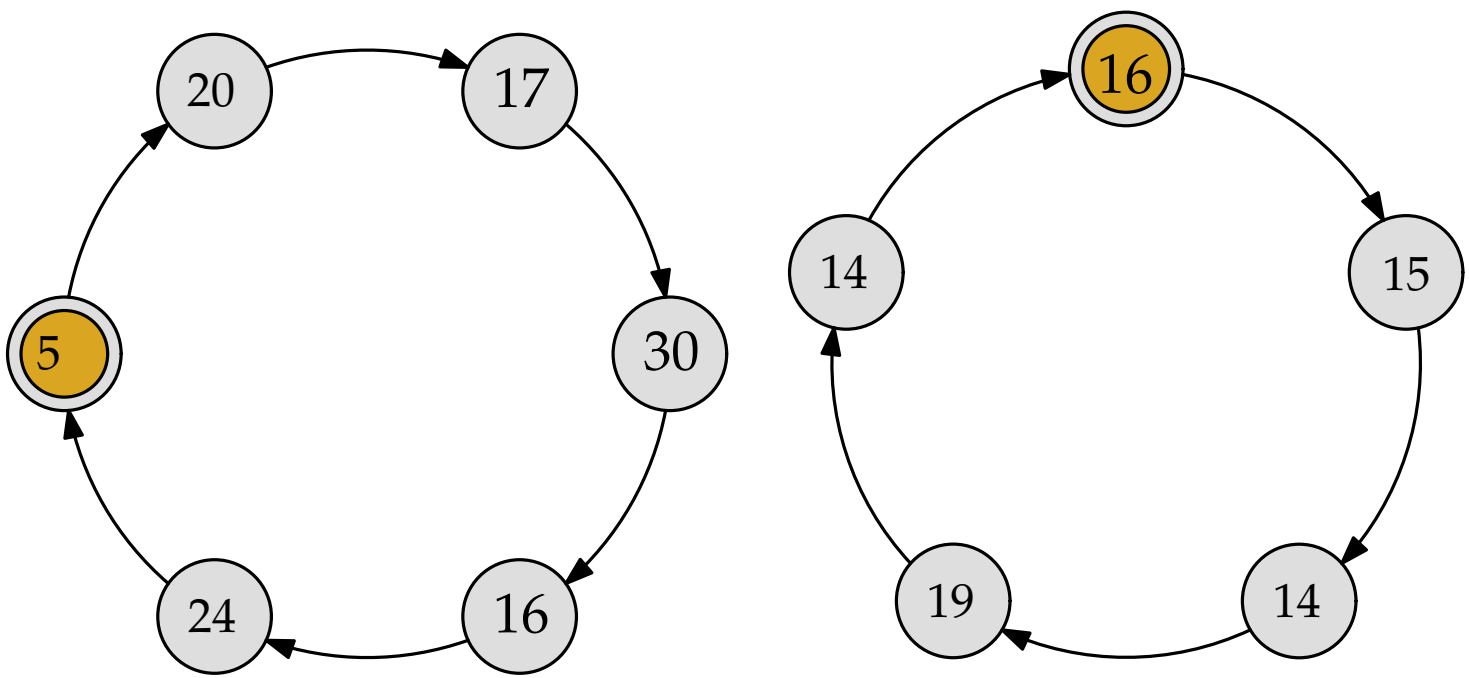
Finding a Local Optimum for $k = 1$



Finding a Local Optimum for $k = 1$

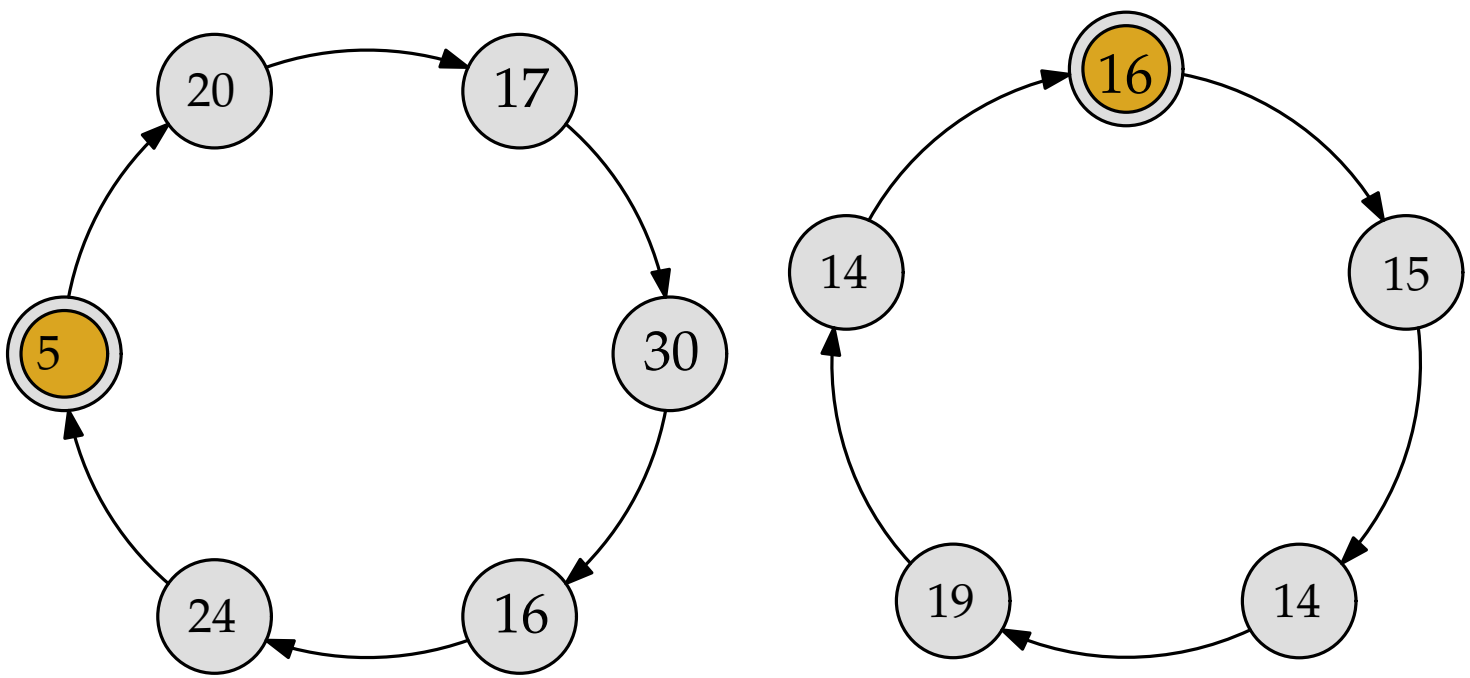


Finding a Local Optimum for $k = 1$



Finding a Local Optimum for $k = 1$

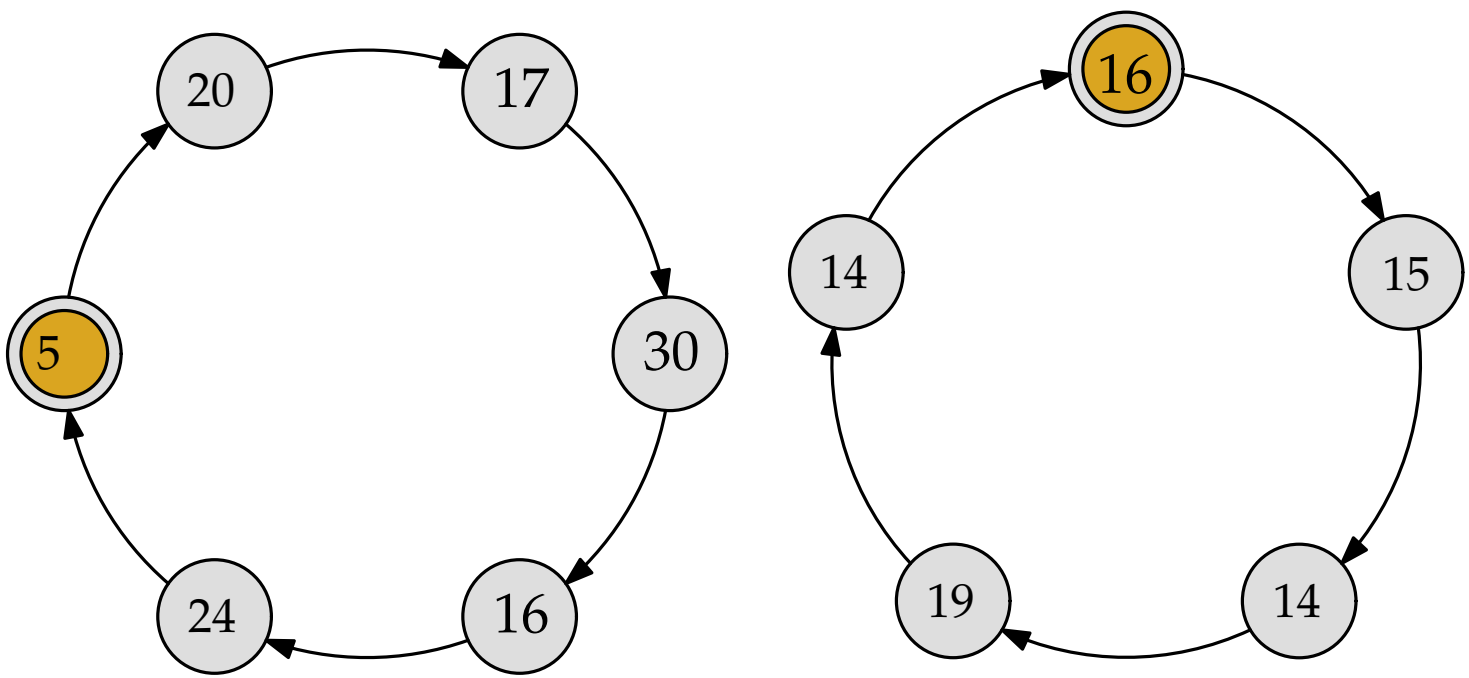
Without loss of generality only one "coin" per cycle.



Finding a Local Optimum for $k = 1$

Without loss of generality only one "coin" per cycle.

Functions $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$

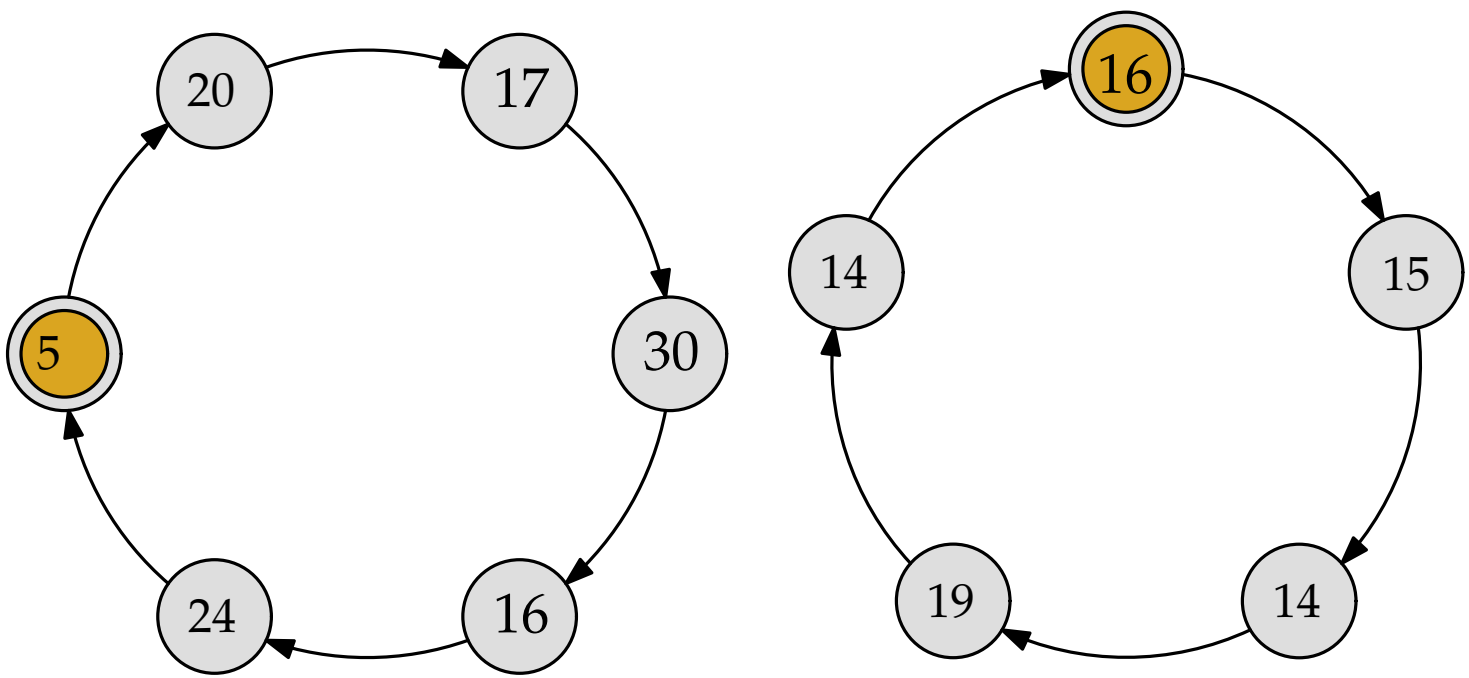


Finding a Local Optimum for $k = 1$

Without loss of generality only one "coin" per cycle.

Functions $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$

with f_i being periodic with period p_i

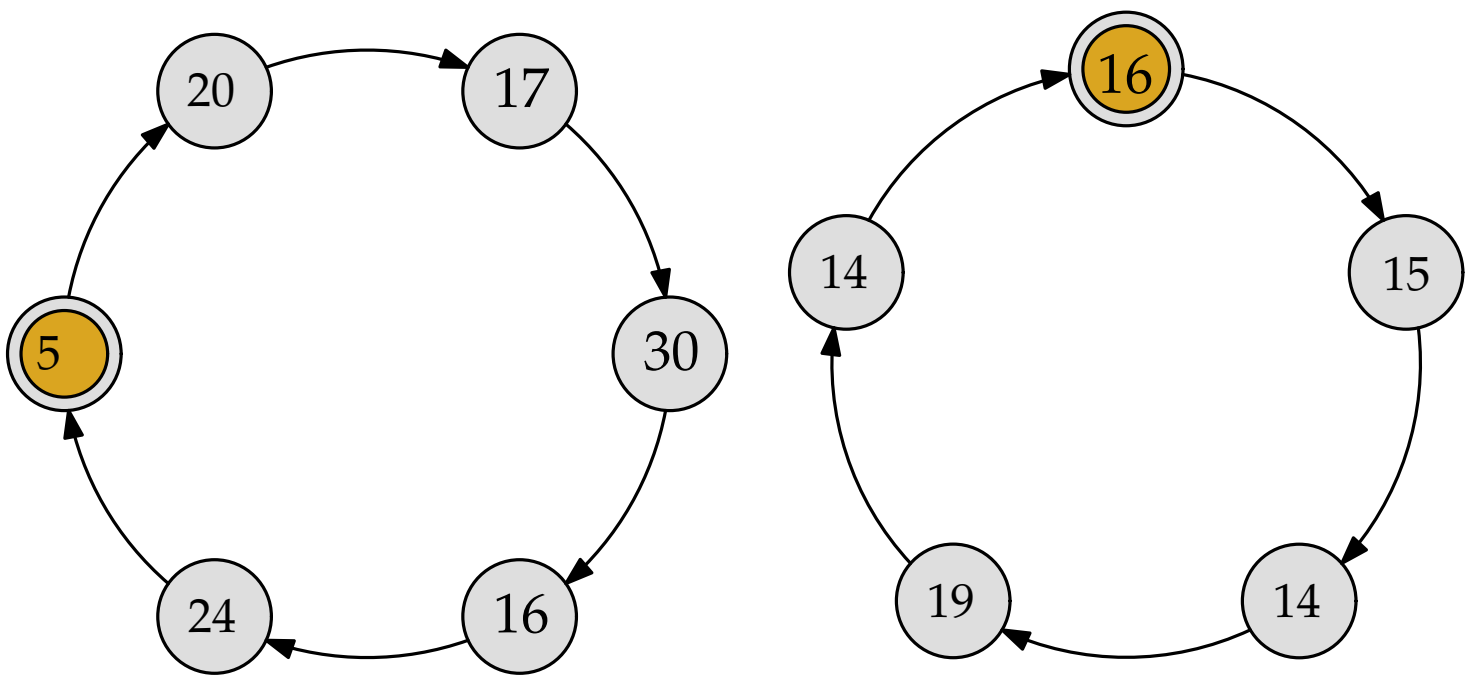


Finding a Local Optimum for $k = 1$

Without loss of generality only one "coin" per cycle.

Functions $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$

with f_i being periodic with period p_i and $p_1 + \cdots + p_l = n$



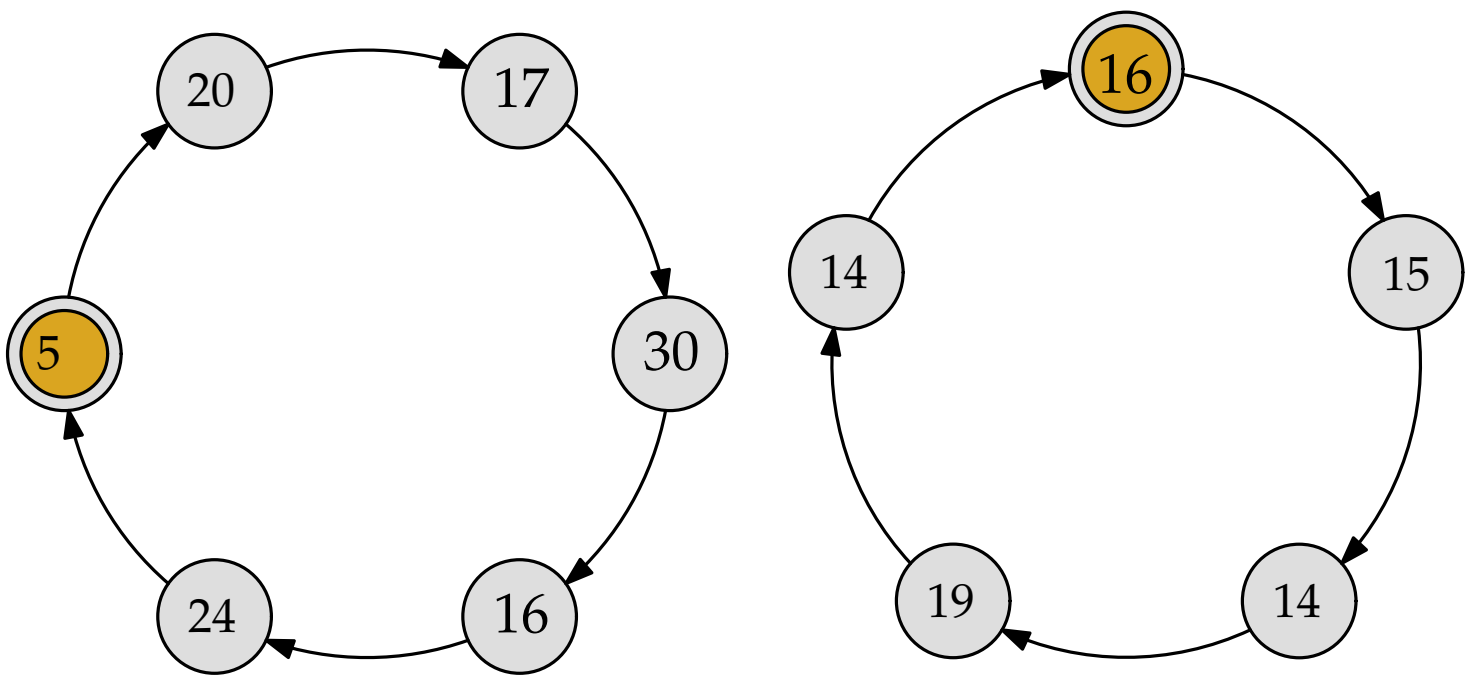
Finding a Local Optimum for $k = 1$

Without loss of generality only one "coin" per cycle.

Functions $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$

with f_i being periodic with period p_i and $p_1 + \cdots + p_l = n$

Function $f := f_1 + \cdots + f_l$



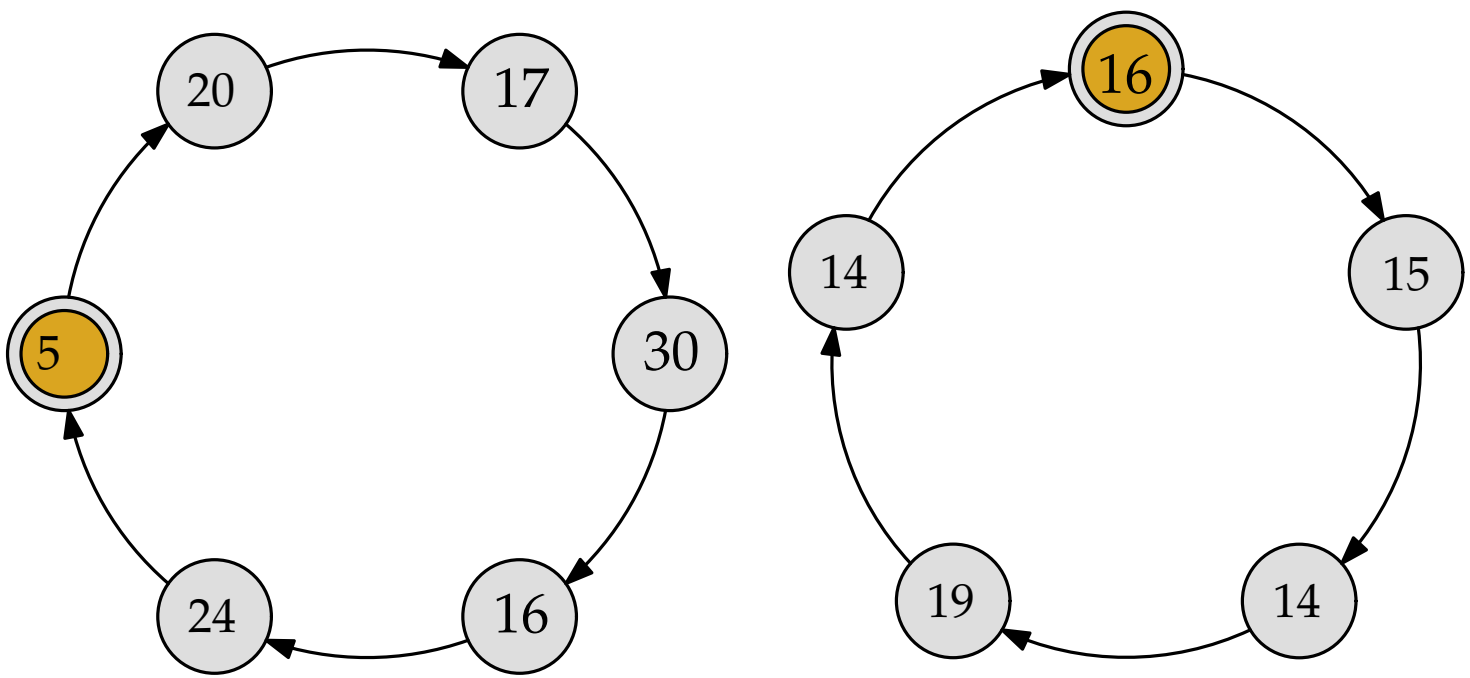
Finding a Local Optimum for $k = 1$

Without loss of generality only one "coin" per cycle.

Functions $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$

with f_i being periodic with period p_i and $p_1 + \cdots + p_l = n$

Function $f := f_1 + \cdots + f_l$ and $f(t) = \mathrm{val}(x \circ \pi^t)$



Finding a Local Optimum for $k = 1$

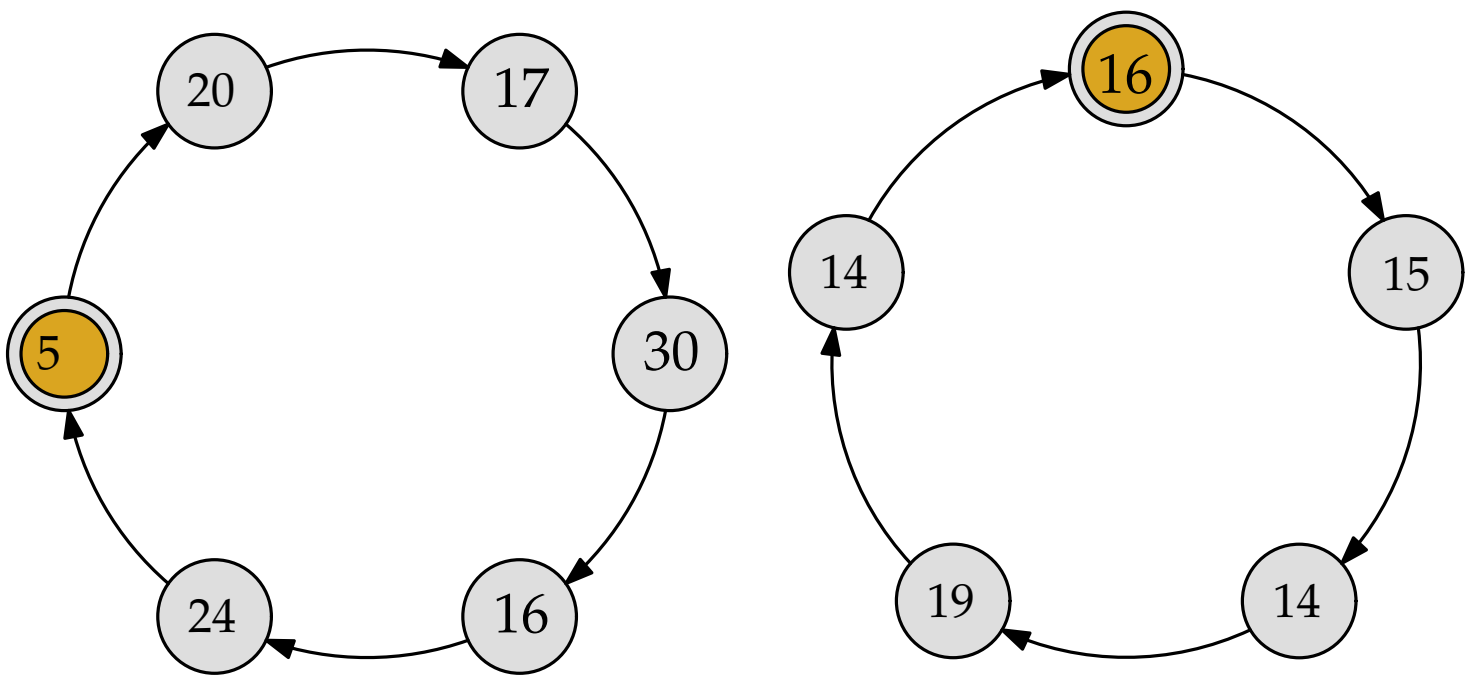
Without loss of generality only one "coin" per cycle.

Functions $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$

with f_i being periodic with period p_i and $p_1 + \cdots + p_l = n$

Function $f := f_1 + \cdots + f_l$ and $f(t) = \mathrm{val}(x \circ \pi^t)$

On how large an interval $[a, b]$ can f be strictly decreasing?



Finding a Local Optimum for $k = 1$

Lemma. Let $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$ be periodic functions, f_i having period p_i, and set $f := f_1 + \cdots + f_l$.
If $f(a) > f(a+1) > \cdots > f(b-1) > f(b)$ then
$b - a \leq (p_1 - 1) + (p_2 - 1) + \cdots + (p_l - 1)$

Finding a Local Optimum for $k = 1$

Lemma. Let $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$ be periodic functions, f_i having period p_i, and set $f := f_1 + \cdots + f_l$.
If $f(a) > f(a+1) > \cdots > f(b-1) > f(b)$ then
$b - a \leq (p_1 - 1) + (p_2 - 1) + \cdots + (p_l - 1)$

Proof. Induction on l. For $l = 0$ it's true because 0 is not decreasing.

Finding a Local Optimum for $k = 1$

Lemma. Let $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$ be periodic functions, f_i having period p_i, and set $f := f_1 + \cdots + f_l$.
If $f(a) > f(a+1) > \cdots > f(b-1) > f(b)$ then
$b - a \leq (p_1 - 1) + (p_2 - 1) + \cdots + (p_l - 1)$

Proof. Induction on l. For $l = 0$ it's true because 0 is not decreasing. For $l \geq 1$ replace each f_i by

$$F_i(t) := f_i(t) + f_i(t+1) + \cdots + f_i(t + p_1 - 1) \,.$$

Finding a Local Optimum for $k = 1$

Lemma. Let $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$ be periodic functions, f_i having period p_i, and set $f := f_1 + \cdots + f_l$.
If $f(a) > f(a+1) > \cdots > f(b-1) > f(b)$ then
$b - a \leq (p_1 - 1) + (p_2 - 1) + \cdots + (p_l - 1)$

Proof. Induction on l. For $l = 0$ it's true because 0 is not decreasing. For $l \geq 1$ replace each f_i by

$$F_i(t) := f_i(t) + f_i(t+1) + \cdots + f_i(t + p_1 - 1) \,.$$

Each F_i is still p_i-periodic.

Finding a Local Optimum for $k = 1$

Lemma. Let $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$ be periodic functions, f_i having period p_i, and set $f := f_1 + \cdots + f_l$.
If $f(a) > f(a+1) > \cdots > f(b-1) > f(b)$ then
$b - a \leq (p_1 - 1) + (p_2 - 1) + \cdots + (p_l - 1)$

Proof. Induction on l. For $l = 0$ it's true because 0 is not decreasing. For $l \geq 1$ replace each f_i by

$$F_i(t) := f_i(t) + f_i(t+1) + \cdots + f_i(t + p_1 - 1) \,.$$

Each F_i is still p_i-periodic. And F_1 is constant.

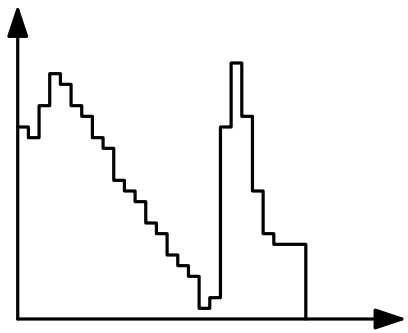
Finding a Local Optimum for $k = 1$

Lemma. Let $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$ be periodic functions, f_i having period p_i, and set $f := f_1 + \cdots + f_l$.
If $f(a) > f(a+1) > \cdots > f(b-1) > f(b)$ then
$b - a \leq (p_1 - 1) + (p_2 - 1) + \cdots + (p_l - 1)$

Proof. Induction on l. For $l = 0$ it's true because 0 is not decreasing. For $l \geq 1$ replace each f_i by

$$F_i(t) := f_i(t) + f_i(t+1) + \cdots + f_i(t + p_1 - 1) \,.$$

Each F_i is still p_i-periodic. And F_1 is constant.



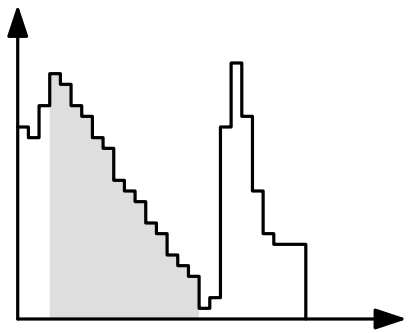
Finding a Local Optimum for $k = 1$

Lemma. Let $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$ be periodic functions, f_i having period p_i, and set $f := f_1 + \cdots + f_l$.
If $f(a) > f(a+1) > \cdots > f(b-1) > f(b)$ then
$b - a \leq (p_1 - 1) + (p_2 - 1) + \cdots + (p_l - 1)$

Proof. Induction on l. For $l = 0$ it's true because 0 is not decreasing. For $l \geq 1$ replace each f_i by

$$F_i(t) := f_i(t) + f_i(t+1) + \cdots + f_i(t + p_1 - 1) \,.$$

Each F_i is still p_i-periodic. And F_1 is constant.



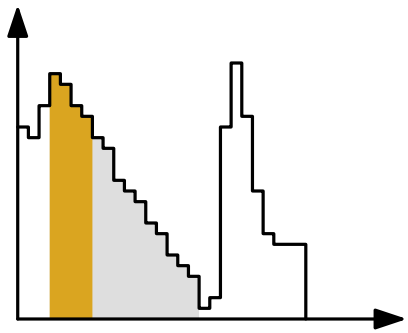
Finding a Local Optimum for $k = 1$

Lemma. Let $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$ be periodic functions, f_i having period p_i, and set $f := f_1 + \cdots + f_l$.
If $f(a) > f(a+1) > \cdots > f(b-1) > f(b)$ then
$b - a \leq (p_1 - 1) + (p_2 - 1) + \cdots + (p_l - 1)$

Proof. Induction on l. For $l = 0$ it's true because 0 is not decreasing. For $l \geq 1$ replace each f_i by

$$F_i(t) := f_i(t) + f_i(t+1) + \cdots + f_i(t + p_1 - 1) \,.$$

Each F_i is still p_i-periodic. And F_1 is constant.



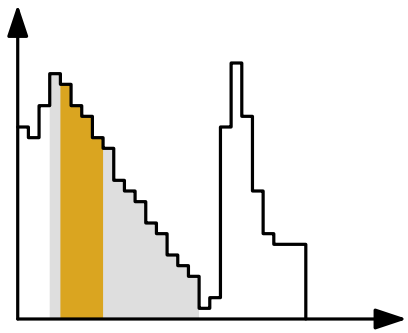
Finding a Local Optimum for $k = 1$

Lemma. Let $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$ be periodic functions, f_i having period p_i, and set $f := f_1 + \cdots + f_l$.
If $f(a) > f(a+1) > \cdots > f(b-1) > f(b)$ then
$b - a \leq (p_1 - 1) + (p_2 - 1) + \cdots + (p_l - 1)$

Proof. Induction on l. For $l = 0$ it's true because 0 is not decreasing. For $l \geq 1$ replace each f_i by

$$F_i(t) := f_i(t) + f_i(t+1) + \cdots + f_i(t + p_1 - 1) \,.$$

Each F_i is still p_i-periodic. And F_1 is constant.



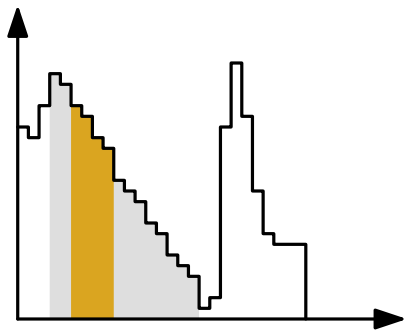
Finding a Local Optimum for $k = 1$

Lemma. Let $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$ be periodic functions, f_i having period p_i, and set $f := f_1 + \cdots + f_l$.
If $f(a) > f(a+1) > \cdots > f(b-1) > f(b)$ then
$b - a \leq (p_1 - 1) + (p_2 - 1) + \cdots + (p_l - 1)$

Proof. Induction on l. For $l = 0$ it's true because 0 is not decreasing. For $l \geq 1$ replace each f_i by

$$F_i(t) := f_i(t) + f_i(t+1) + \cdots + f_i(t + p_1 - 1) \,.$$

Each F_i is still p_i-periodic. And F_1 is constant.



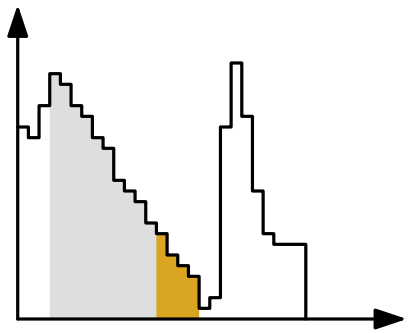
Finding a Local Optimum for $k = 1$

Lemma. Let $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$ be periodic functions, f_i having period p_i, and set $f := f_1 + \cdots + f_l$.
If $f(a) > f(a+1) > \cdots > f(b-1) > f(b)$ then
$b - a \leq (p_1 - 1) + (p_2 - 1) + \cdots + (p_l - 1)$

Proof. Induction on l. For $l = 0$ it's true because 0 is not decreasing. For $l \geq 1$ replace each f_i by

$$F_i(t) := f_i(t) + f_i(t+1) + \cdots + f_i(t + p_1 - 1) \,.$$

Each F_i is still p_i-periodic. And F_1 is constant.



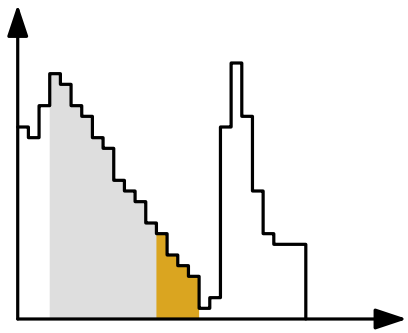
Finding a Local Optimum for $k = 1$

Lemma. Let $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$ be periodic functions, f_i having period p_i, and set $f := f_1 + \cdots + f_l$.
If $f(a) > f(a+1) > \cdots > f(b-1) > f(b)$ then
$b - a \leq (p_1 - 1) + (p_2 - 1) + \cdots + (p_l - 1)$

Proof. Induction on l. For $l = 0$ it's true because 0 is not decreasing. For $l \geq 1$ replace each f_i by

$$F_i(t) := f_i(t) + f_i(t+1) + \cdots + f_i(t+p_1-1)\,.$$

Each F_i is still p_i-periodic. And F_1 is constant.



$$F_i(a) < F_i(a+1) < \cdots < F_i(b-p_1+1)$$

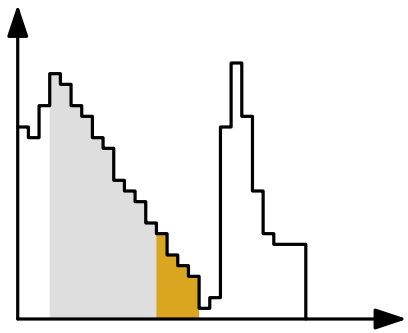
Finding a Local Optimum for $k = 1$

Lemma. Let $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$ be periodic functions, f_i having period p_i, and set $f := f_1 + \cdots + f_l$.
If $f(a) > f(a+1) > \cdots > f(b-1) > f(b)$ then
$b - a \leq (p_1 - 1) + (p_2 - 1) + \cdots + (p_l - 1)$

Proof. Induction on l. For $l = 0$ it's true because 0 is not decreasing. For $l \geq 1$ replace each f_i by

$$F_i(t) := f_i(t) + f_i(t+1) + \cdots + f_i(t + p_1 - 1) \,.$$

Each F_i is still p_i-periodic. And F_1 is constant.



$$F_i(a) < F_i(a+1) < \cdots < F_i(b - p_1 + 1)$$

Now use induction on $F_2, \ldots, F_l$.

Finding a Local Optimum for $k = 1$

Lemma. Let $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$ be periodic functions, f_i having period p_i, and set $f := f_1 + \cdots + f_l$.
If $f(a) < f(a+1) < \cdots < f(b-1) < f(b)$ then
$b - a \leq (p_1 - 1) + (p_2 - 1) + \cdots + (p_l - 1)$

Finding a Local Optimum for $k = 1$

Lemma. Let $f_1, f_2, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$ be periodic functions, f_i having period p_i, and set $f := f_1 + \cdots + f_l$.
If $f(a) < f(a+1) < \cdots < f(b-1) < f(b)$ then
$b - a \leq (p_1 - 1) + (p_2 - 1) + \cdots + (p_l - 1)$

Corollary. For $k = 1$, the improving path for $x \in \{0,1\}^n$ under the action of π_1 contains at most n strings.
This is quite obvious for the lexicographic weight function and follows from the lemma for general weight functions.

Finding a Local Optimum for General k

Finding a Local Optimum for General k

Theorem (S., Tantow, 2025). Finding a local minimum of $x \in \{0,1\}^n$ under permutations $\pi_1, \ldots, \pi_k$ is PLS-complete, even for lexicographic weights.

Finding a Local Optimum for General k

Theorem (S., Tantow, 2025). Finding a local minimum of $x \in \{0,1\}^n$ under permutations $\pi_1, \ldots, \pi_k$ is PLS-complete, even for lexicographic weights.
PLS stands for Polynomial Local Search, and is the complexity class of problems that ask for some kind of local optimum.

Finding a Local Optimum for General k

Theorem (S., Tantow, 2025). Finding a local minimum of $x \in \{0,1\}^n$ under permutations $\pi_1, \ldots, \pi_k$ is PLS-complete, even for lexicographic weights.
PLS stands for Polynomial Local Search, and is the complexity class of problems that ask for some kind of local optimum.

Theorem (S., Tantow, 2026). Even if the $\pi_1, \ldots, \pi_k$ generate

- an Abelian group or
- a group where every element has order 2 or
- a cyclic group.

Finding a Local Optimum for General k

Theorem (S., Tantow, 2025). Finding a local minimum of $x \in \{0,1\}^n$ under permutations $\pi_1, \ldots, \pi_k$ is PLS-complete, even for lexicographic weights.
PLS stands for Polynomial Local Search, and is the complexity class of problems that ask for some kind of local optimum.

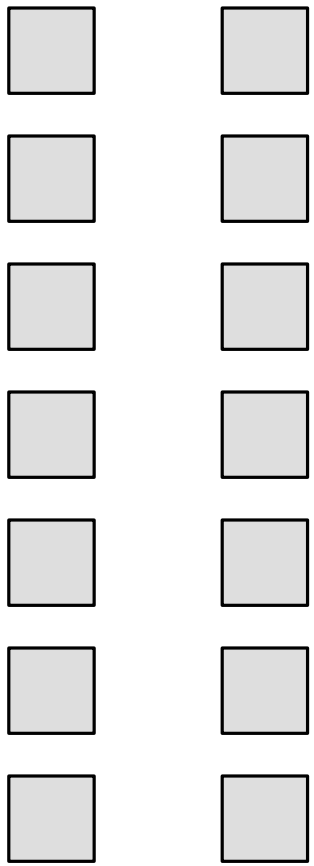
Theorem (S., Tantow, 2026). Even if the $\pi_1, \ldots, \pi_k$ generate

- an Abelian group or
- a group where every element has order 2 or
- a cyclic group.

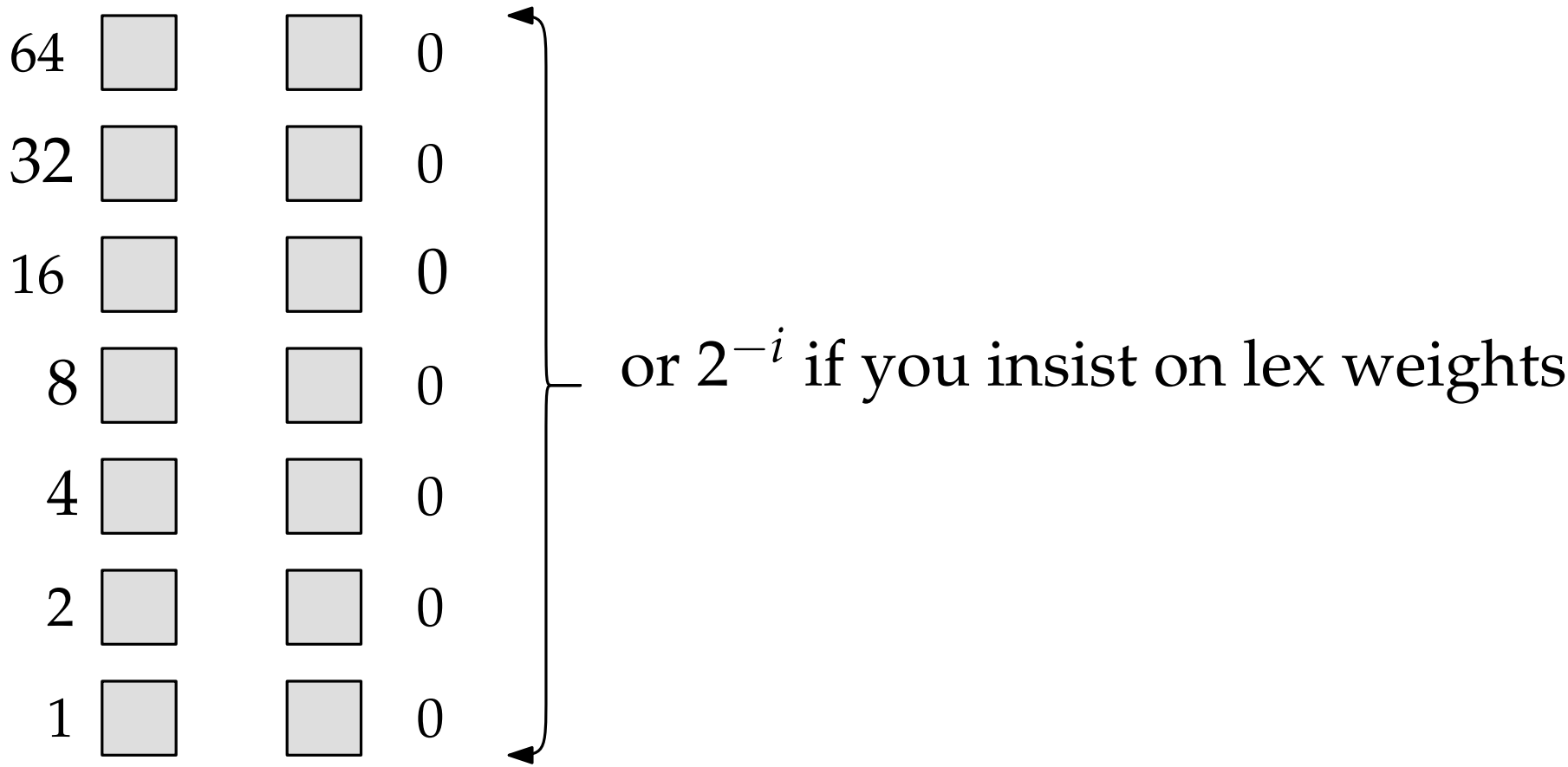
This talk: not complexity theory but how long can improving path get?

General k: Improving Path can be exponential

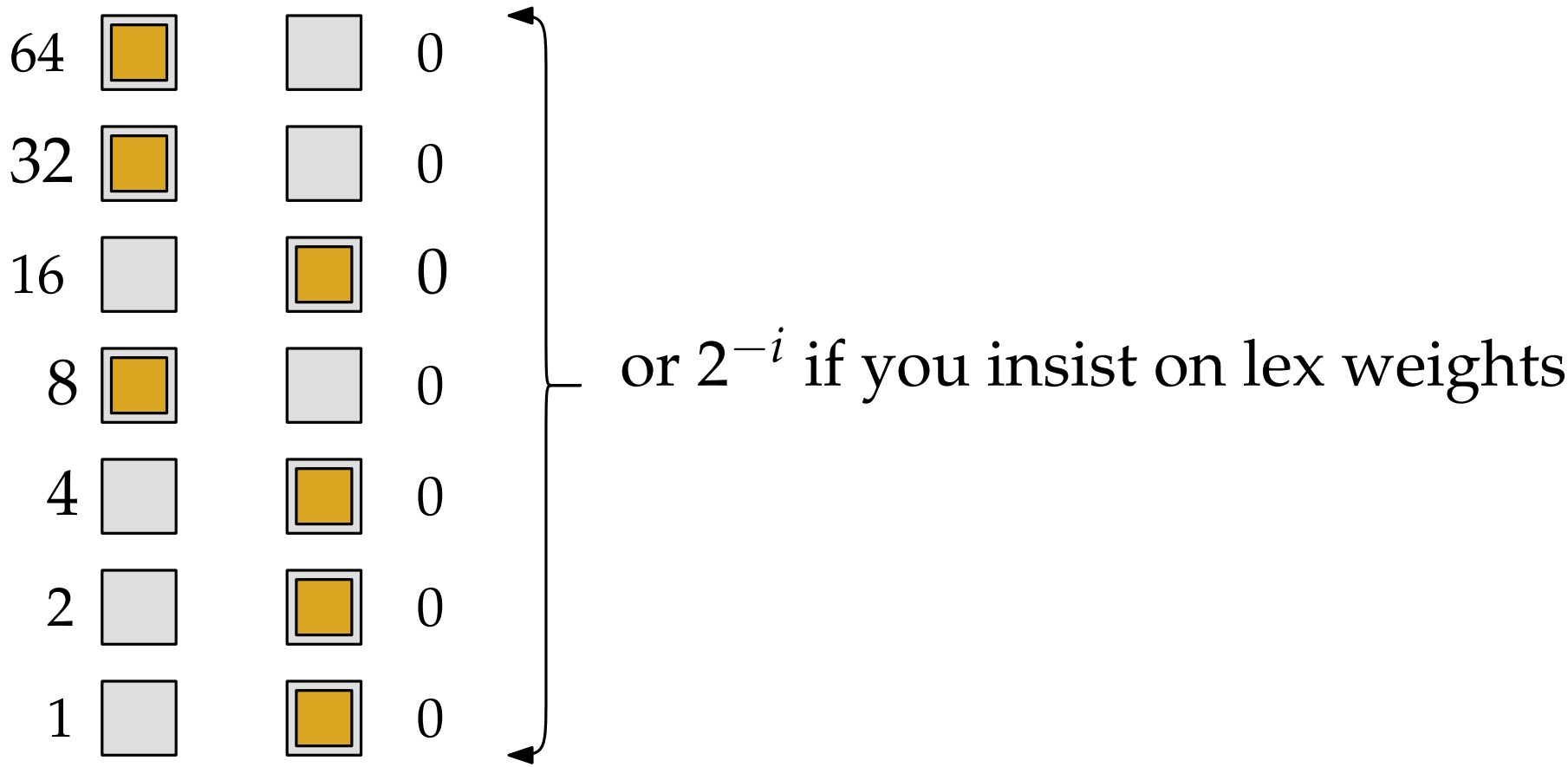
General k: Improving Path can be exponential



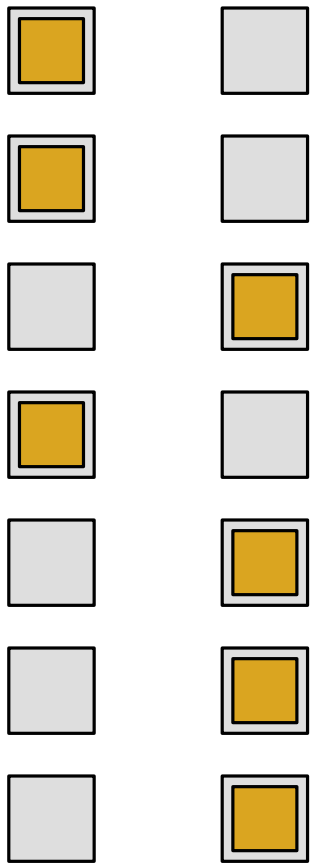
General k: Improving Path can be exponential



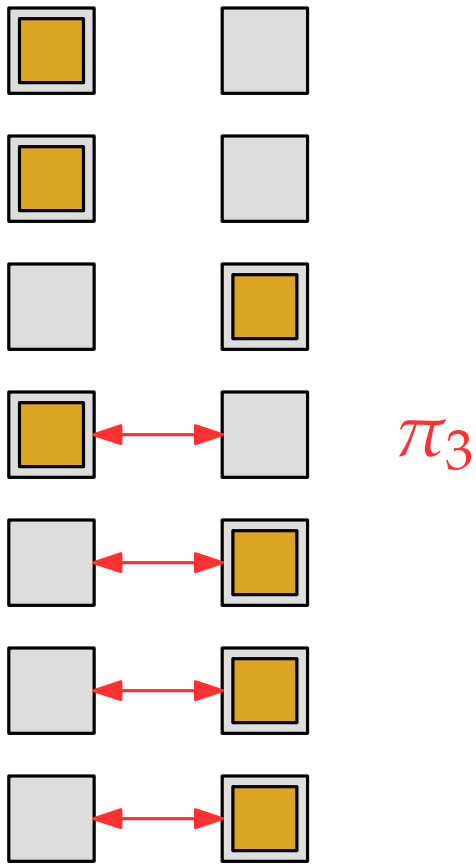
General k: Improving Path can be exponential



General k: Improving Path can be exponential

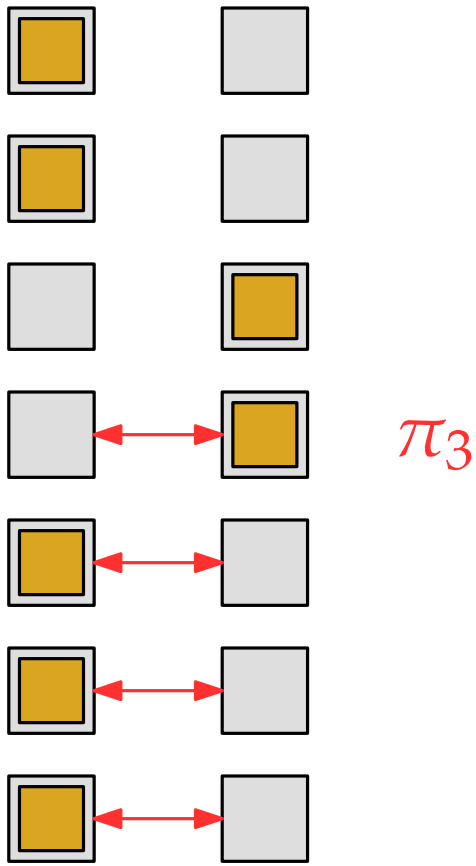


General k: Improving Path can be exponential



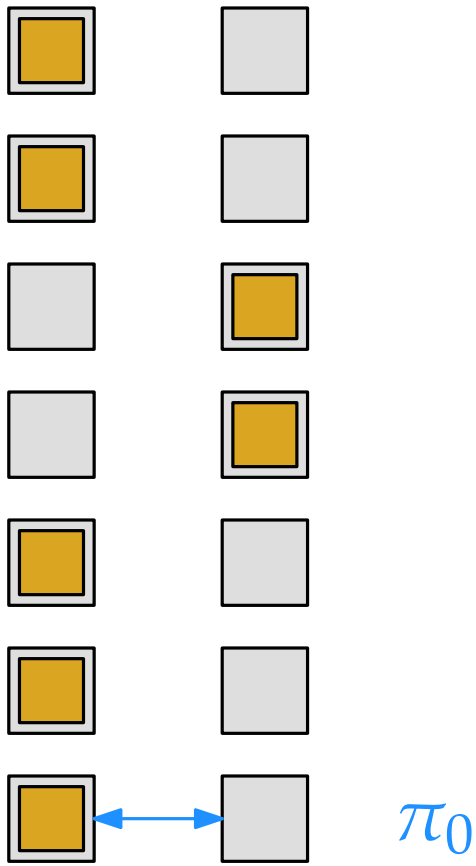
π_3

General k: Improving Path can be exponential



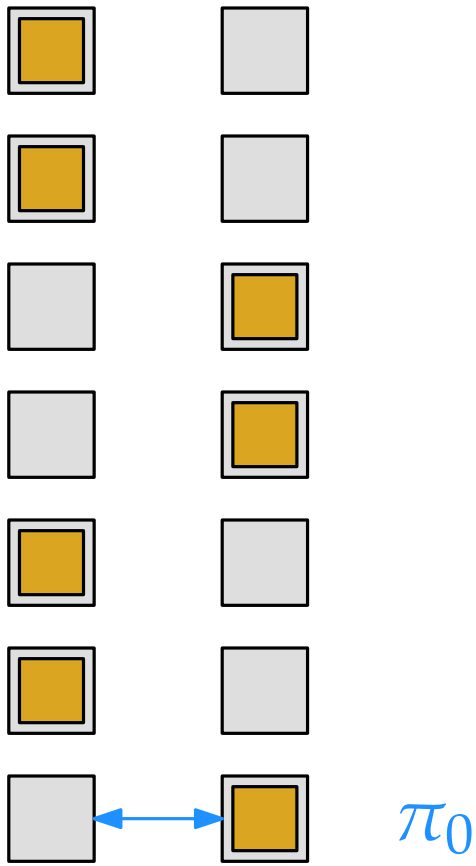
π_3

General k: Improving Path can be exponential



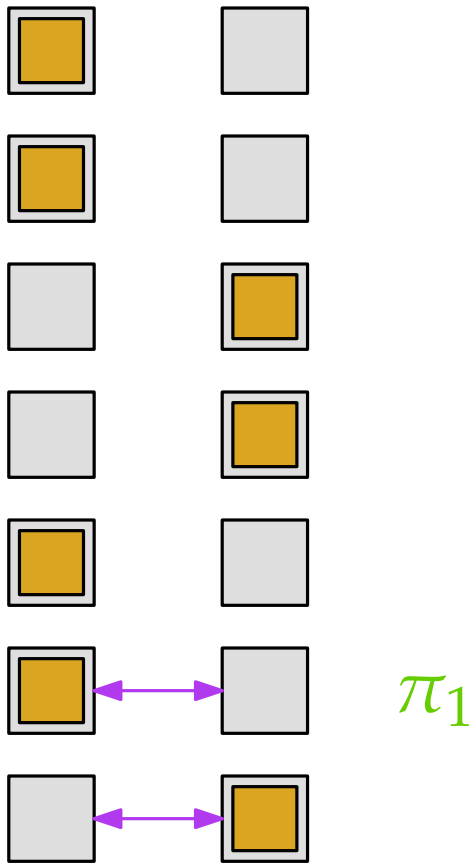
π_0

General k: Improving Path can be exponential



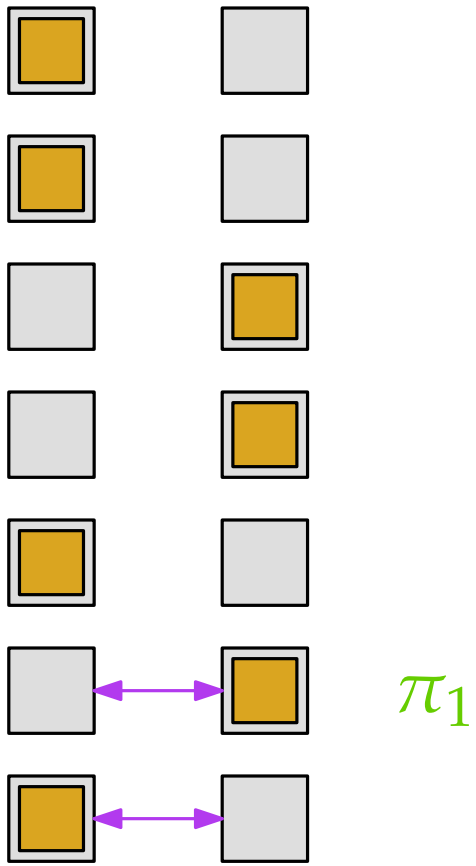
π_0

General k: Improving Path can be exponential



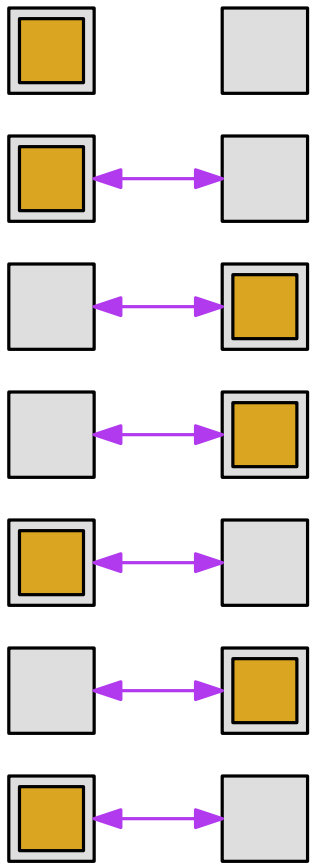
π_1

General k: Improving Path can be exponential



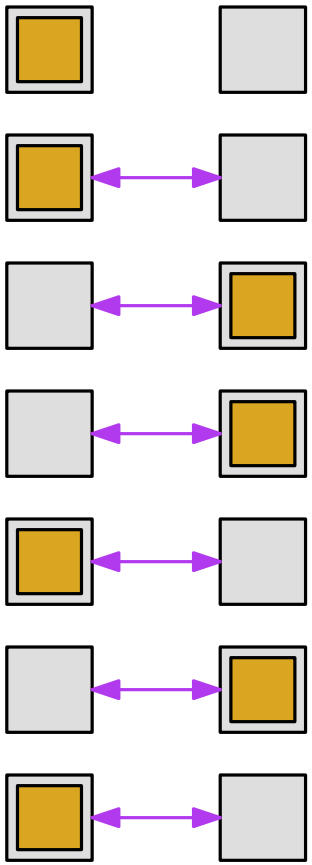
π_1

General k: Improving Path can be exponential



π_5

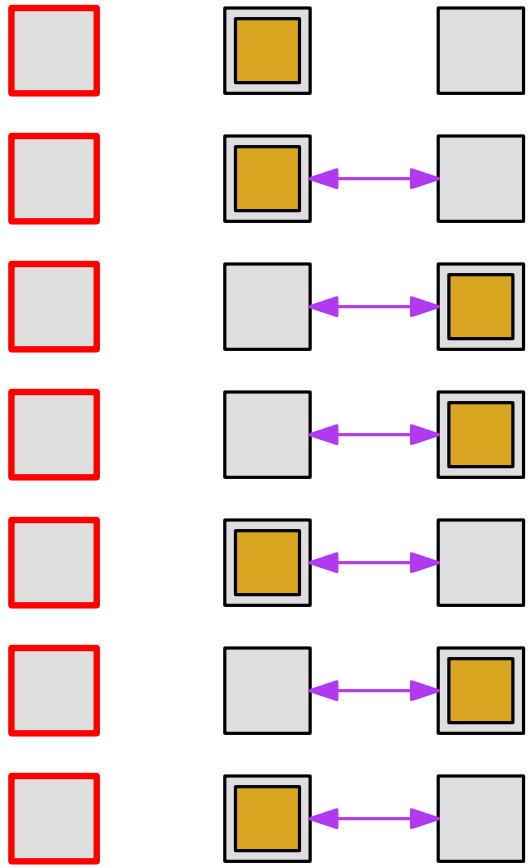
General k: Improving Path can be exponential



π_5

Wait! We have to prevent the "player" to apply π_5 prematurely.

General k: Improving Path can be exponential



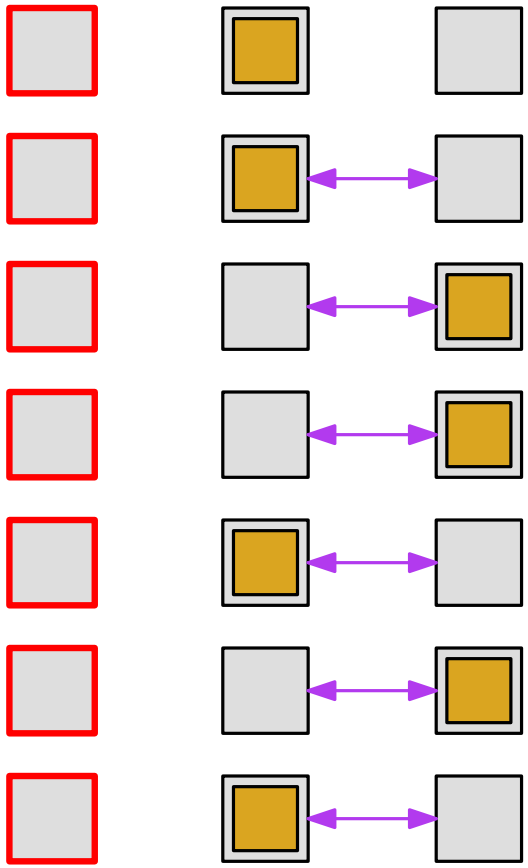
π_5

Wait! We have to prevent the "player" to apply π_5 prematurely.

Death positions of infinite weight (or 2^i for $i \geq n$ if you insist)

They will never carry a coin during an improving path.

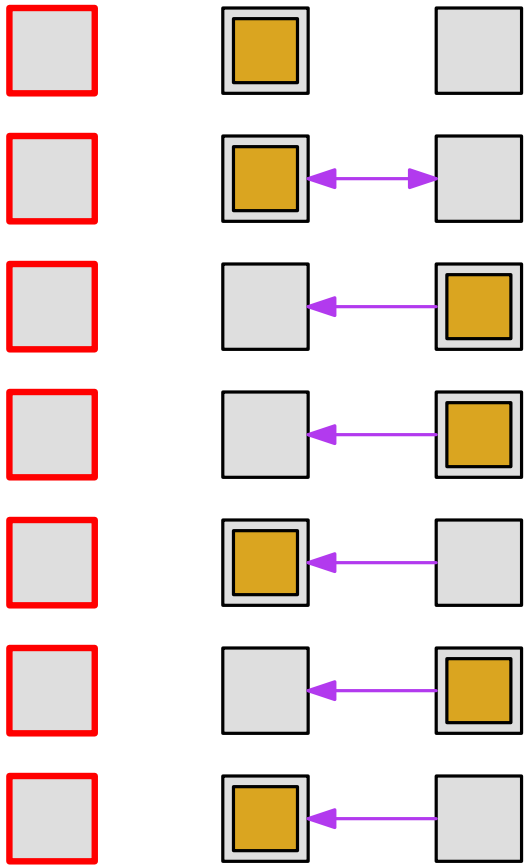
General k: Improving Path can be exponential



π_5

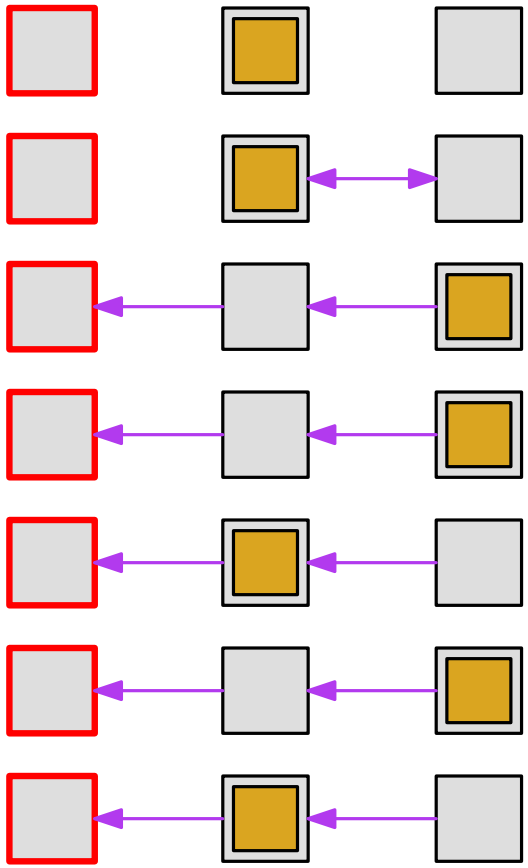
Wait! We have to prevent the "player" to apply π_5 prematurely.

General k: Improving Path can be exponential



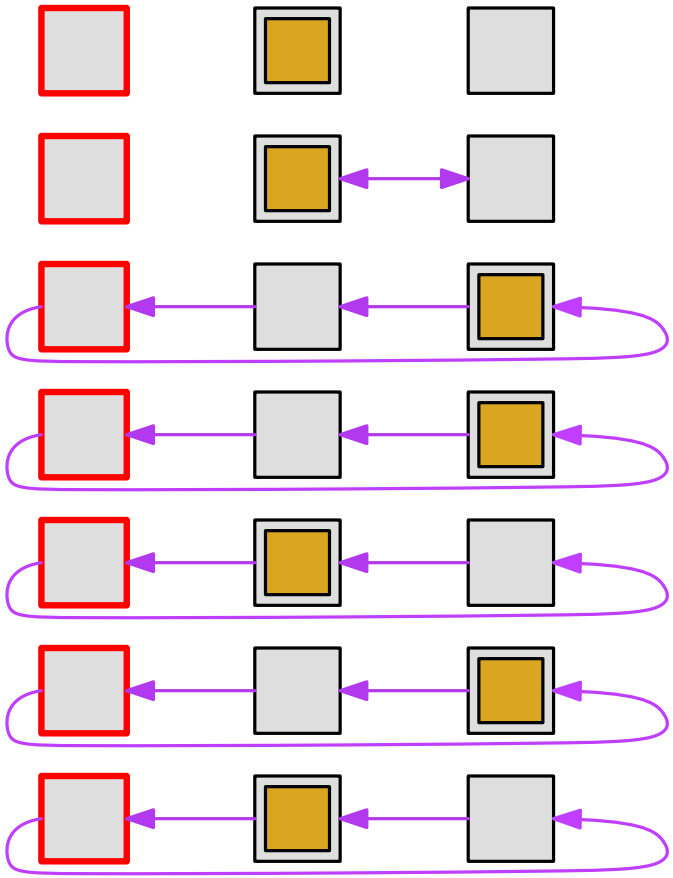
Wait! We have to prevent the "player" to apply π_5 prematurely.

General k: Improving Path can be exponential



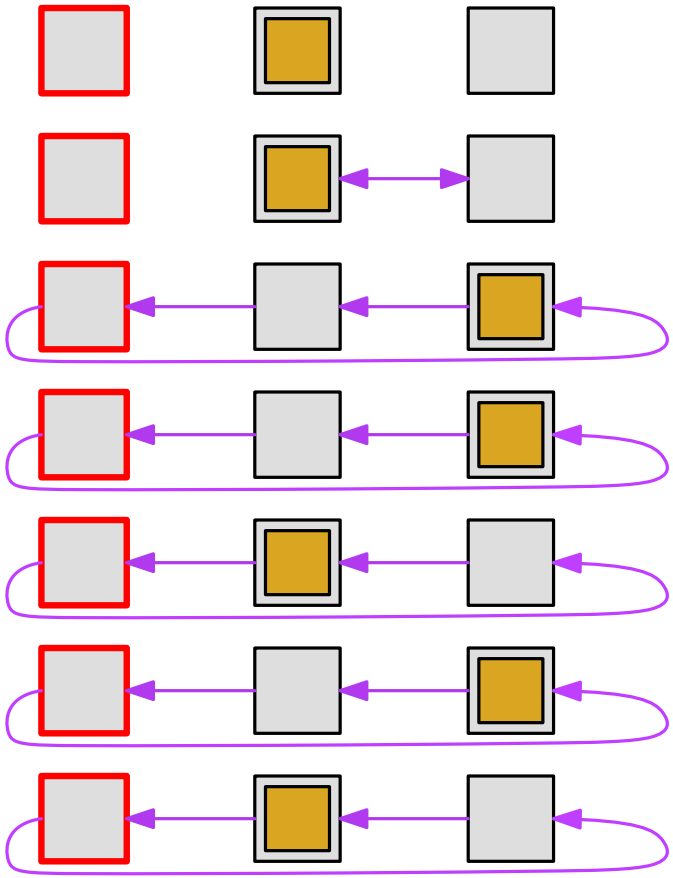
Wait! We have to prevent the "player" to apply π_5 prematurely.

General k: Improving Path can be exponential



Wait! We have to prevent the "player" to apply π_5 prematurely.

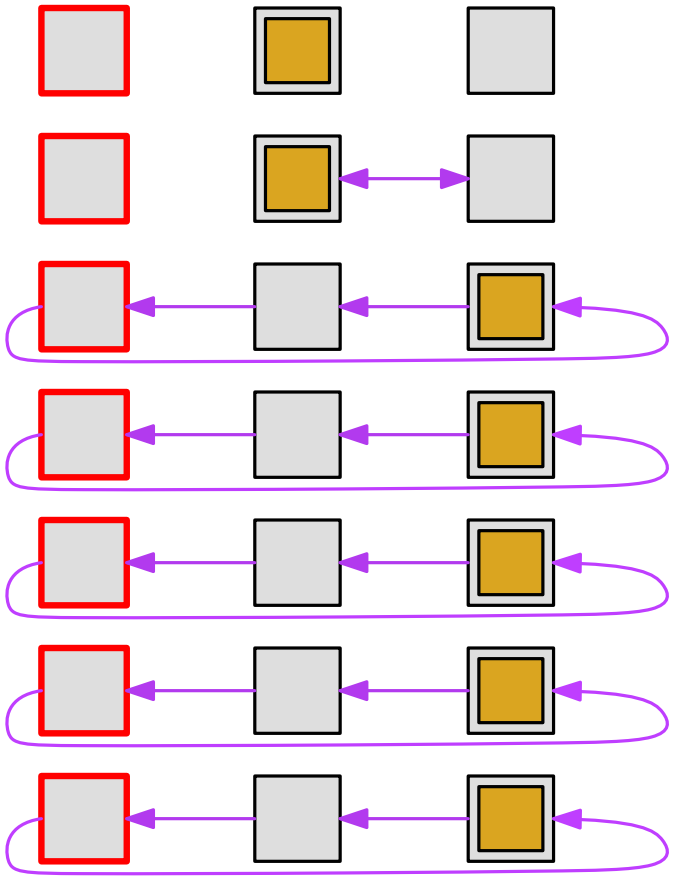
General k: Improving Path can be exponential



Wait! We have to prevent the "player" to apply π_5 prematurely.

Only once $b_0 = b_1 = \cdots = b_4 = 0$ can we apply π_5!

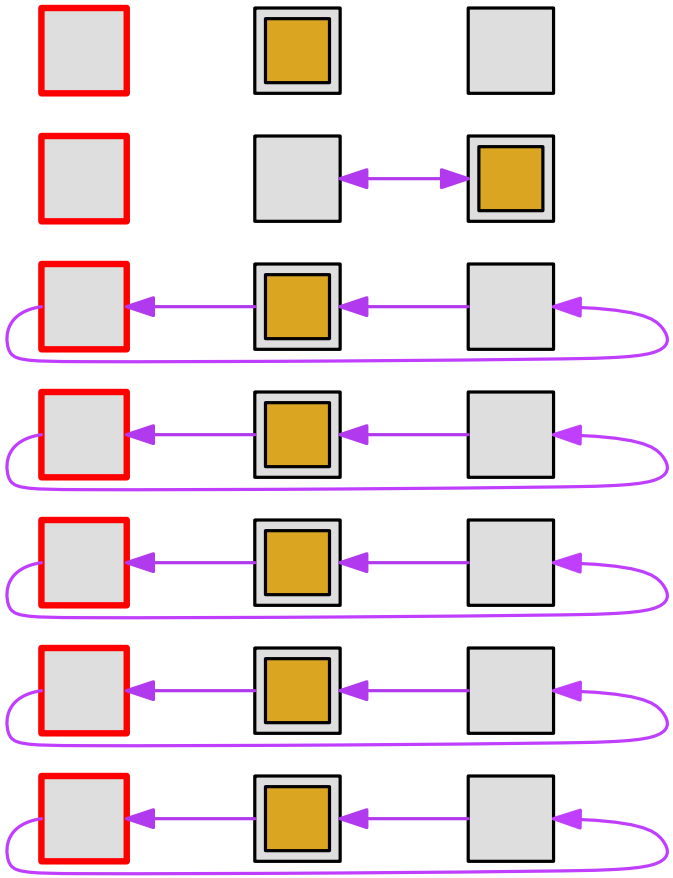
General k: Improving Path can be exponential



Wait! We have to prevent the "player" to apply π_5 prematurely.

Only once $b_0 = b_1 = \cdots = b_4 = 0$ can we apply π_5!

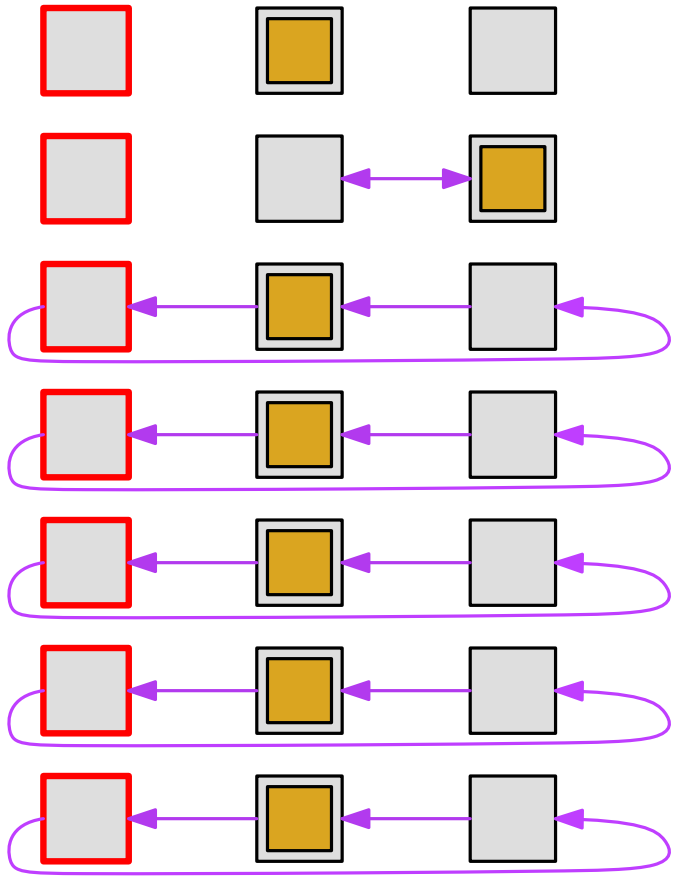
General k: Improving Path can be exponential



Wait! We have to prevent the "player" to apply π_5 prematurely.

Only once $b_0 = b_1 = \cdots = b_4 = 0$ can we apply π_5!

General k: Improving Path can be exponential



Wait! We have to prevent the "player" to apply π_5 prematurely.

Only once $b_0 = b_1 = \cdots = b_4 = 0$ can we apply π_5!

With n permutations and $3n$ positions we can build a counter that counts from $2^n - 1$ to 0.

Constant k: Improving Path can be exponential?

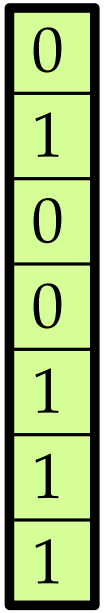
Surely for *constant* k the length of an improving path must be polynomial?

Constant k: Improving Path can be exponential?

Surely for *constant* k the length of an improving path must be polynomial?

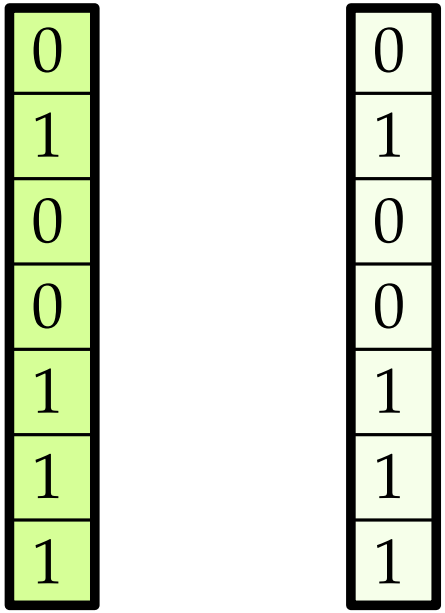
Construction. A machine with a constant number of permutations implementing a binary counter.

A counter with constantly many permutations



A register containing the current number in binary. These positions are pretty valuable.

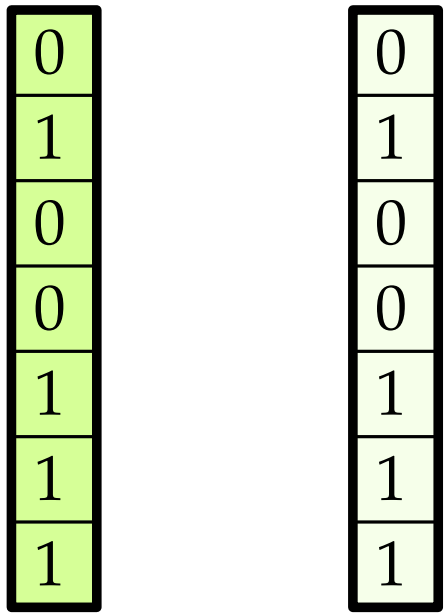
A counter with constantly many permutations



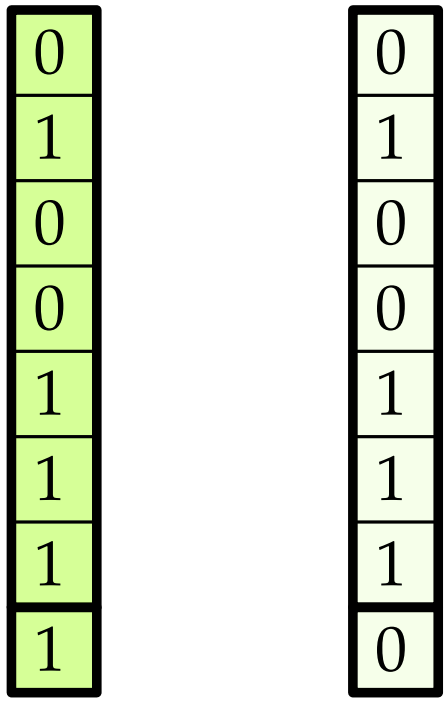
A copy of it, but the positions here are basically worthless.

A register containing the current number in binary. These positions are pretty valuable.

A counter with constantly many permutations

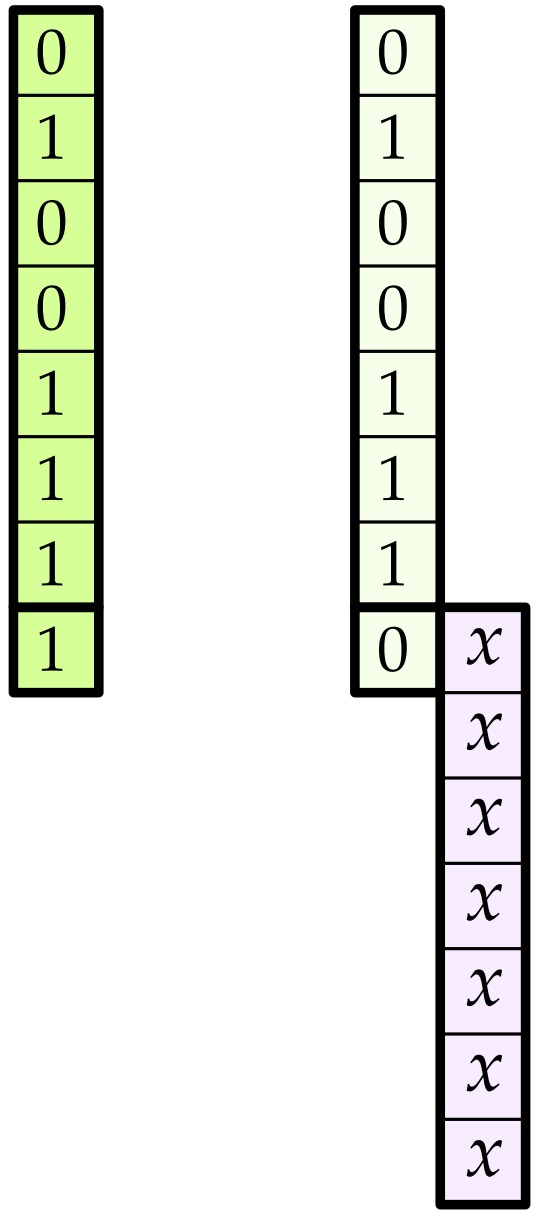


A counter with constantly many permutations



The two numbers are not equal: they differ by $\frac{1}{2}$ because there is one more position "after the dot".

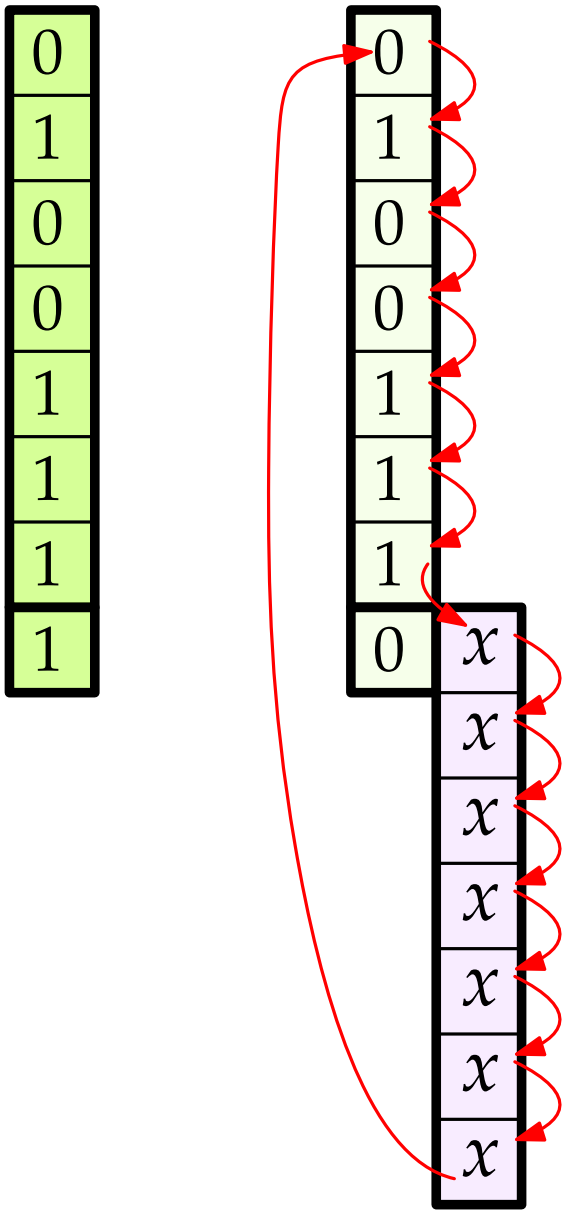
A counter with constantly many permutations



Some manoeuvering space filled with a new symbol, x. For example, we can encode $0, 1, x$ using three positions per cell.

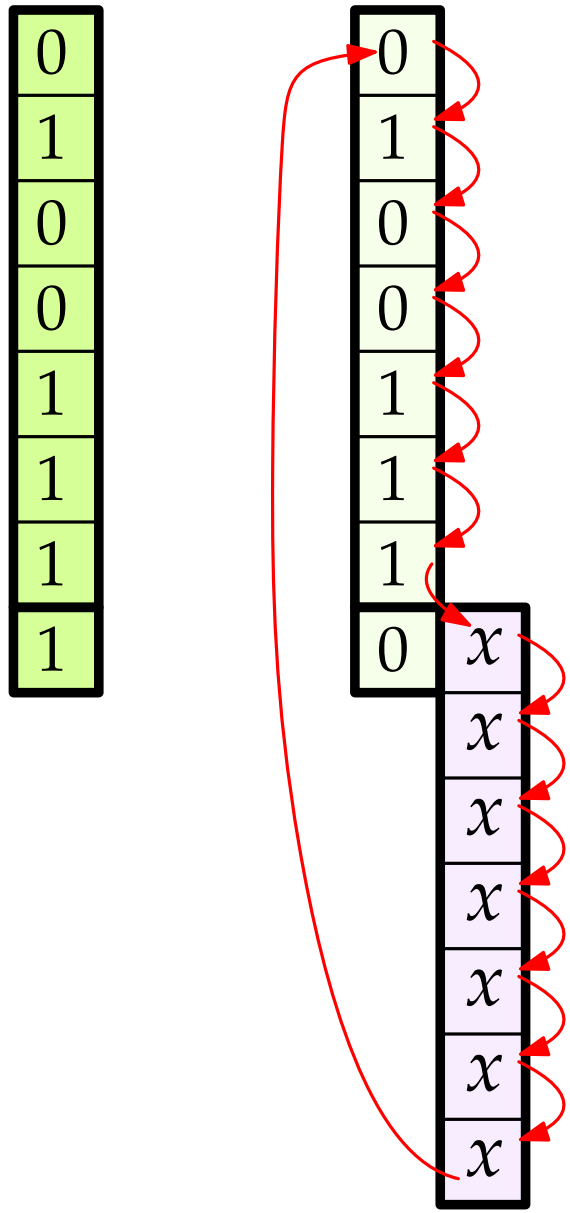
This space "likes" to have 1's but does not tolerate 0's. A 0 here would give payoff $-\infty$.

A counter with constantly many permutations



The permutation π

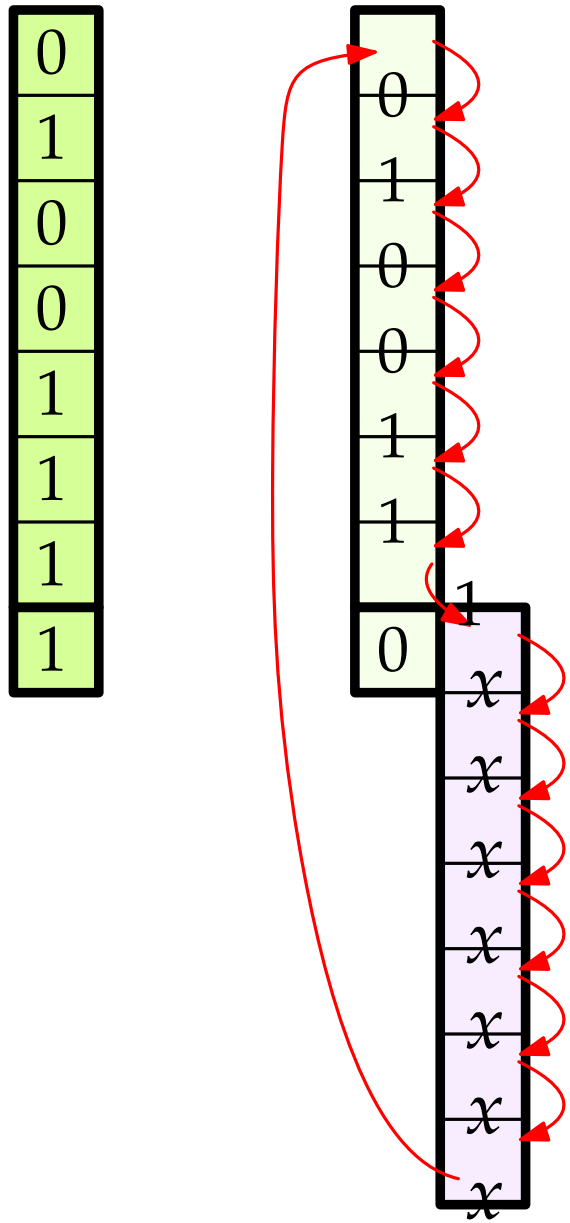
A counter with constantly many permutations



The permutation π

Applying π increases the payoff because the manoeuvering buffer gives us a payoff for each 1.

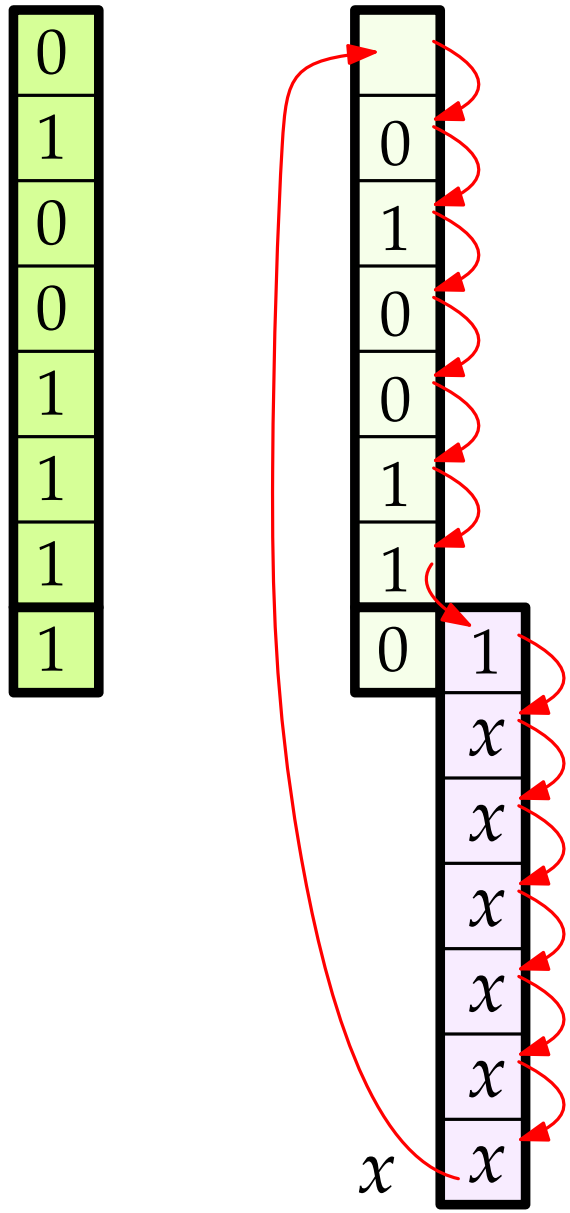
A counter with constantly many permutations



The permutation π

Applying π increases the payoff because the manoeuvering buffer gives us a payoff for each 1.

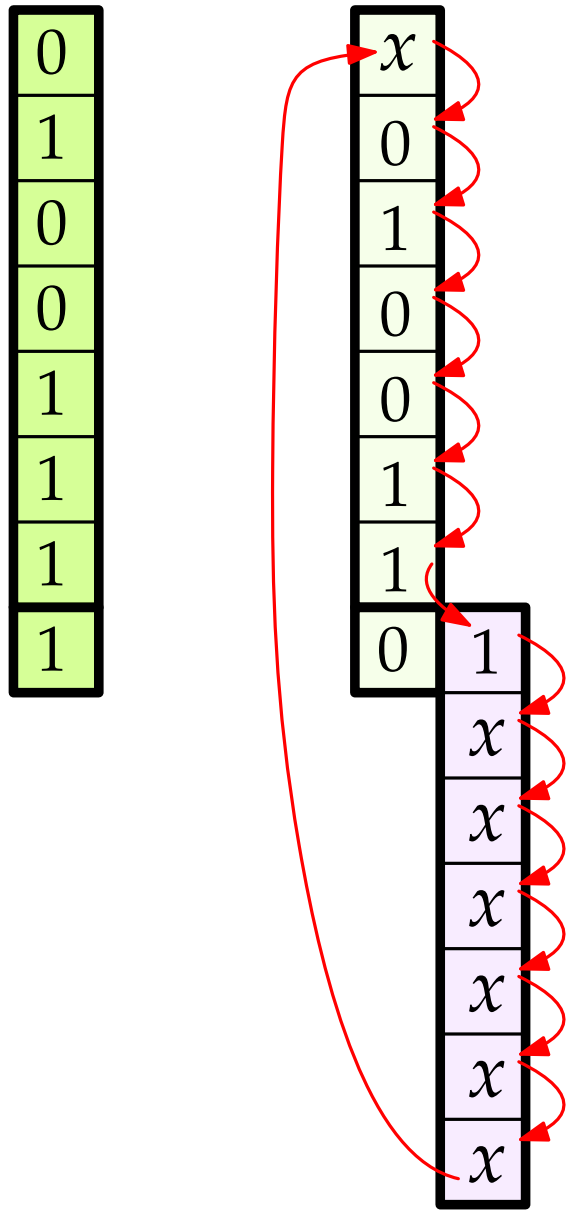
A counter with constantly many permutations



The permutation π

Applying π increases the payoff because the manoeuvering buffer gives us a payoff for each 1.

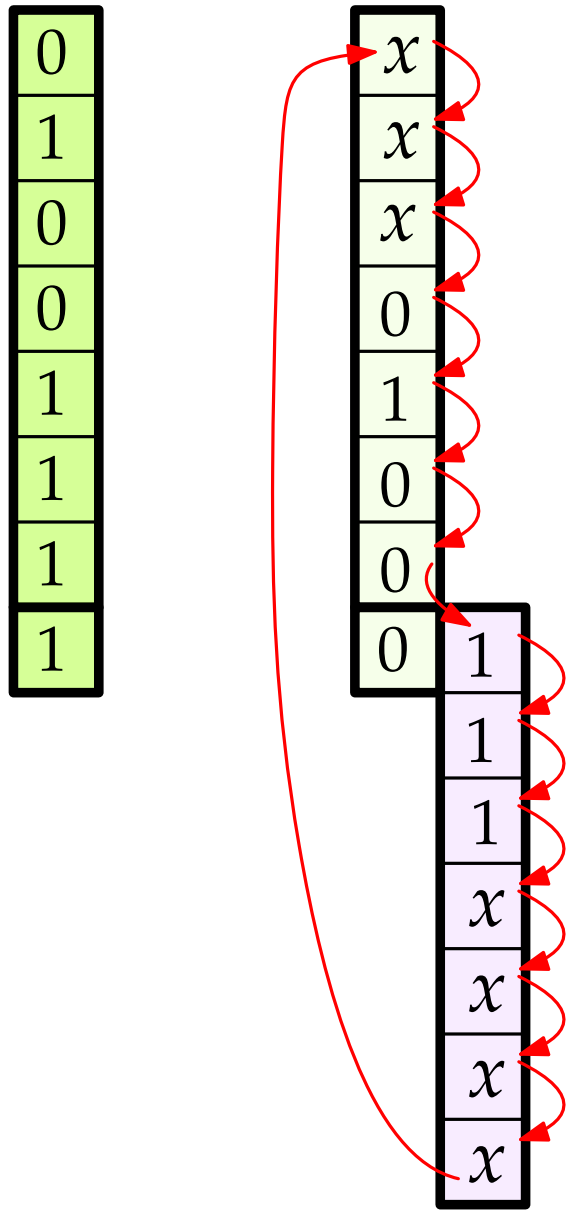
A counter with constantly many permutations



The permutation π

Applying π increases the payoff because the manoeuvering buffer gives us a payoff for each 1.

A counter with constantly many permutations

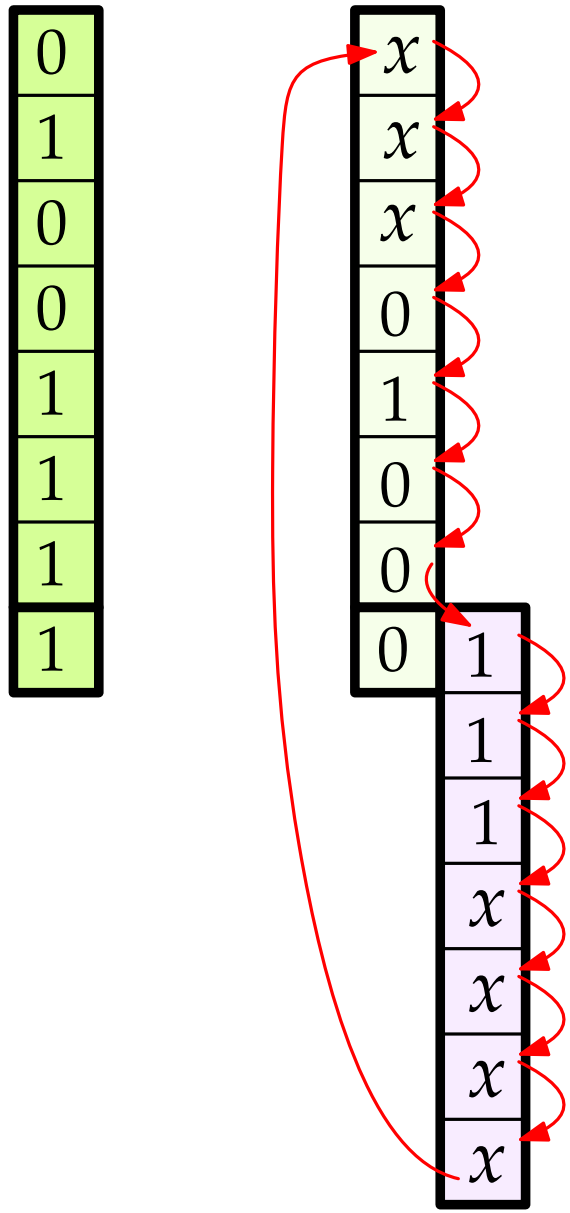


The permutation π

Applying π increases the payoff because the manoeuvering buffer gives us a payoff for each 1.

...and twice more

A counter with constantly many permutations



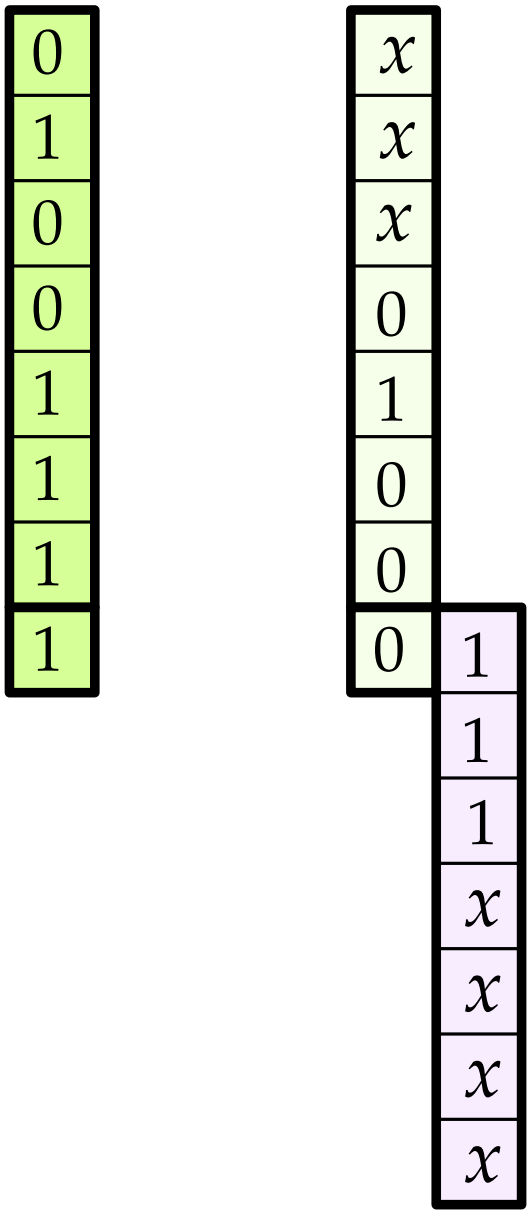
The permutation π

Applying π increases the payoff because the manoeuvering buffer gives us a payoff for each 1.

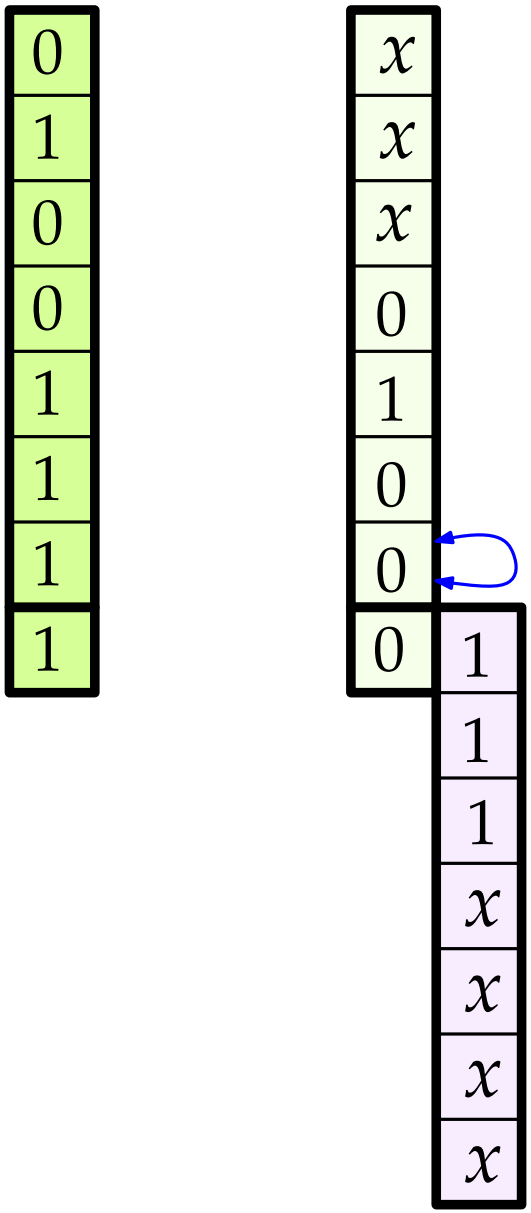
...and twice more

Cannot do it again because a 0 on violet carries an extreme penalty.

A counter with constantly many permutations

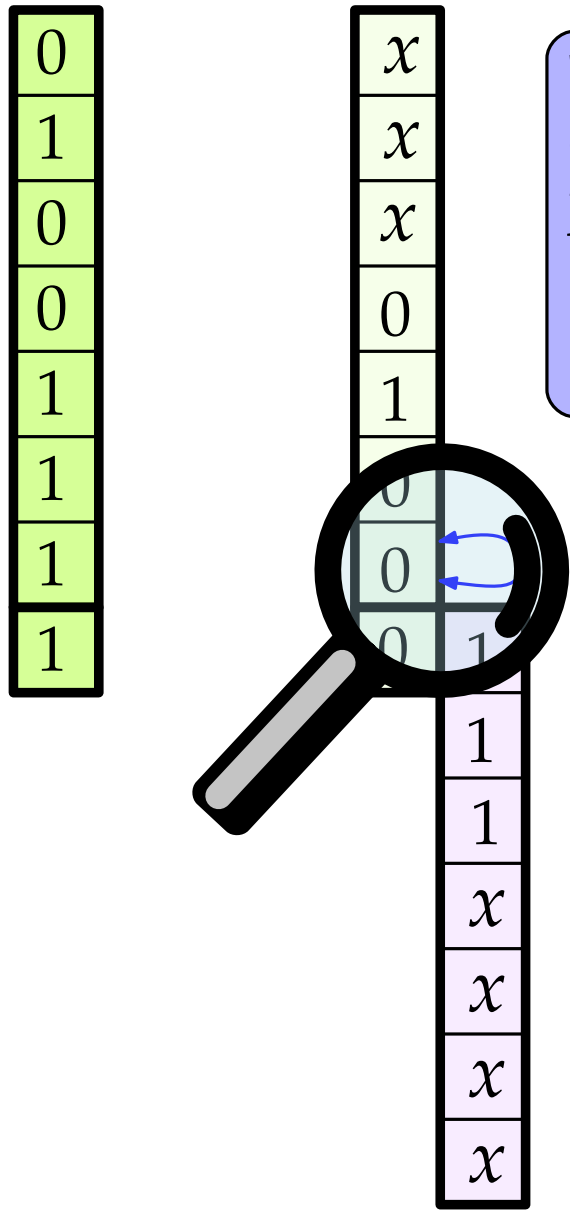


A counter with constantly many permutations



The permutation σ inverts the lowest cell in the copy register, but can only be applied if it has a 0 there.

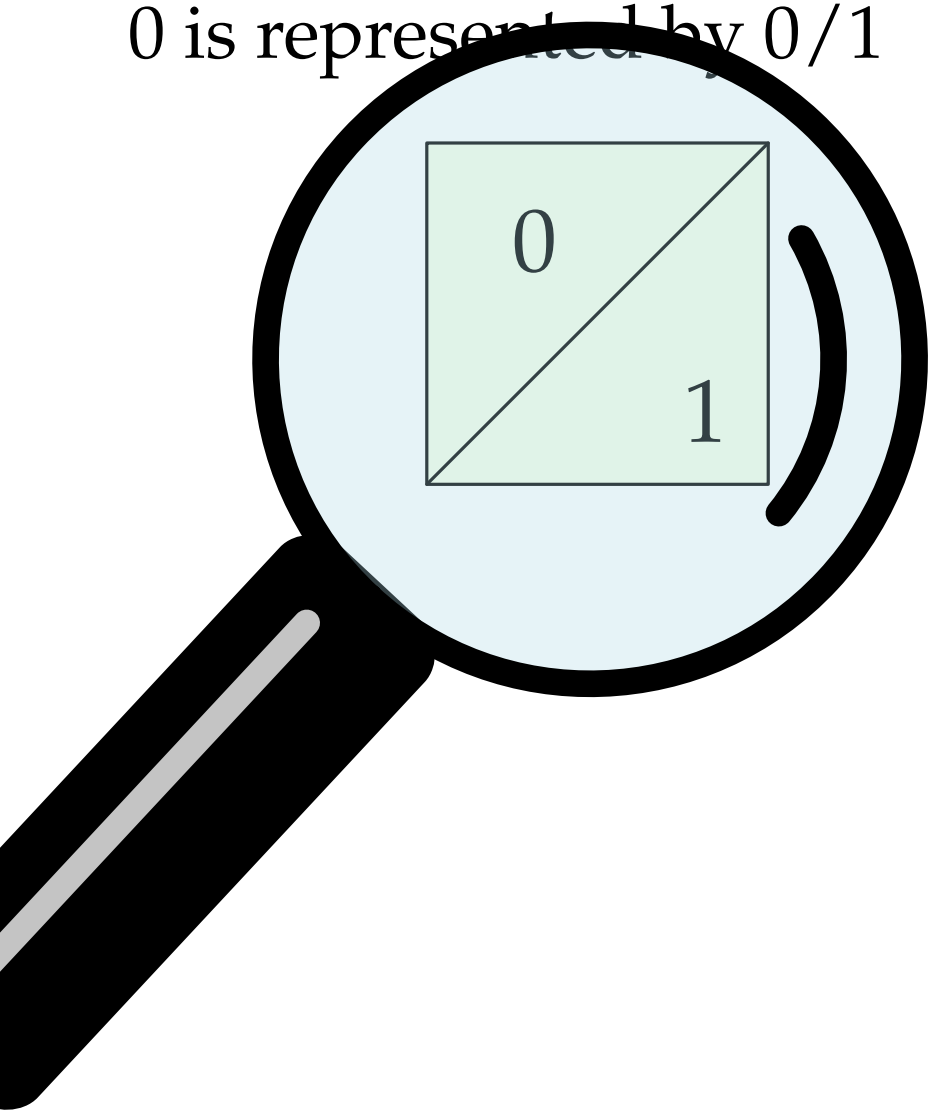
A counter with constantly many permutations



The permutation σ inverts the lowest cell in the copy register, but can only be applied if it has a 0 there.

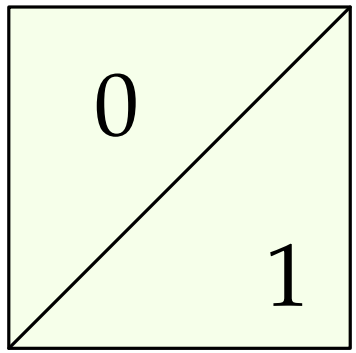
A counter with constantly many permutations

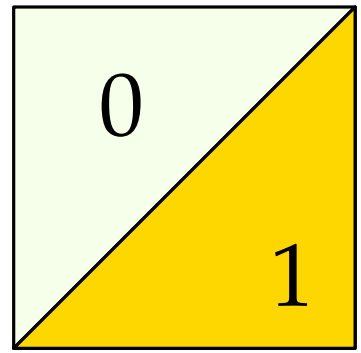
0 is represented by 0/1



A counter with constantly many permutations

0 is represented by 0/1

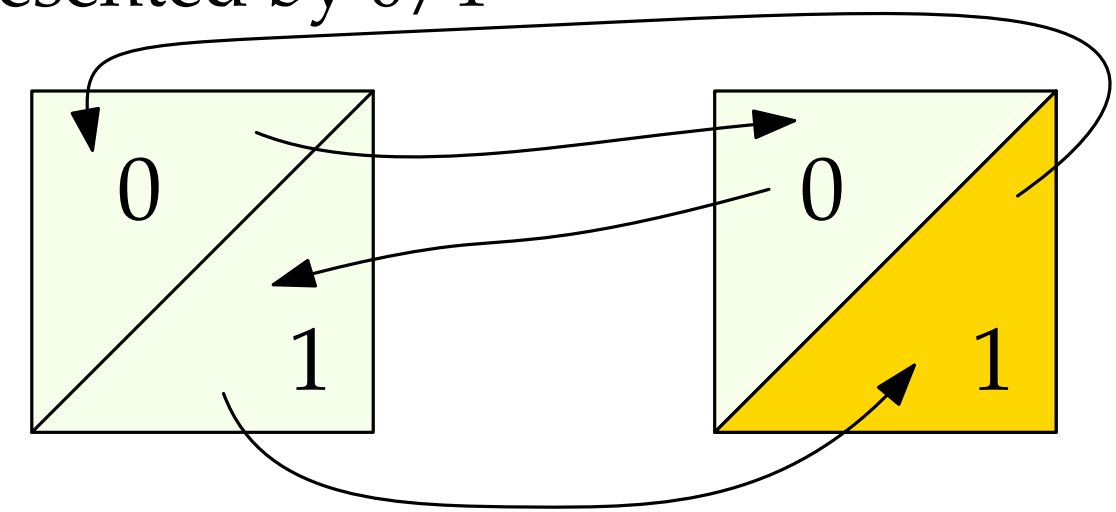




The golden 1 is extremely valuable

A counter with constantly many permutations

0 is represented by 0/1

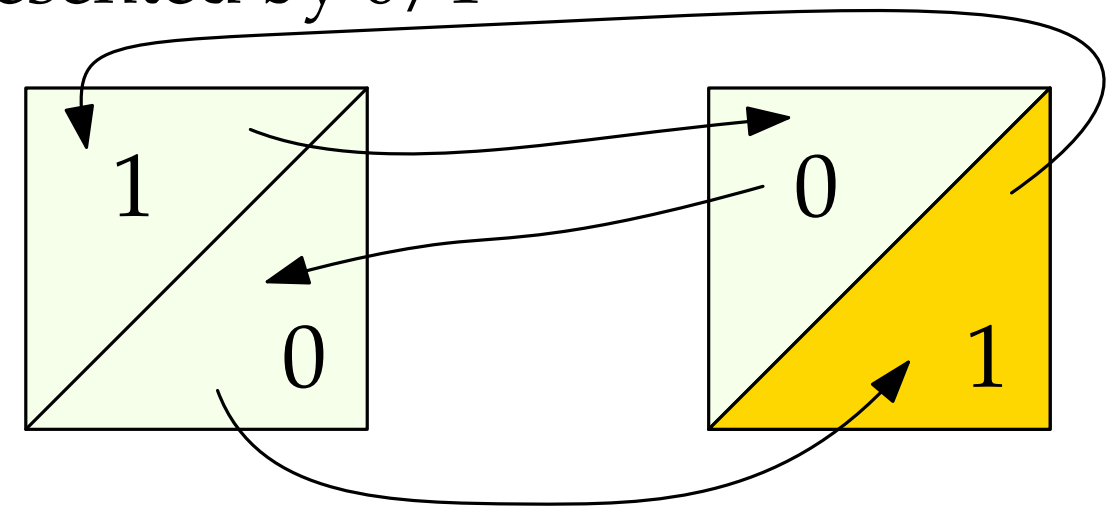


The golden 1 is extremely valuable

If the left square is in state 0 (represented by 0/1) then applying σ inverts the left cell and leaves the right cell unchanged.

A counter with constantly many permutations

0 is represented by 0/1

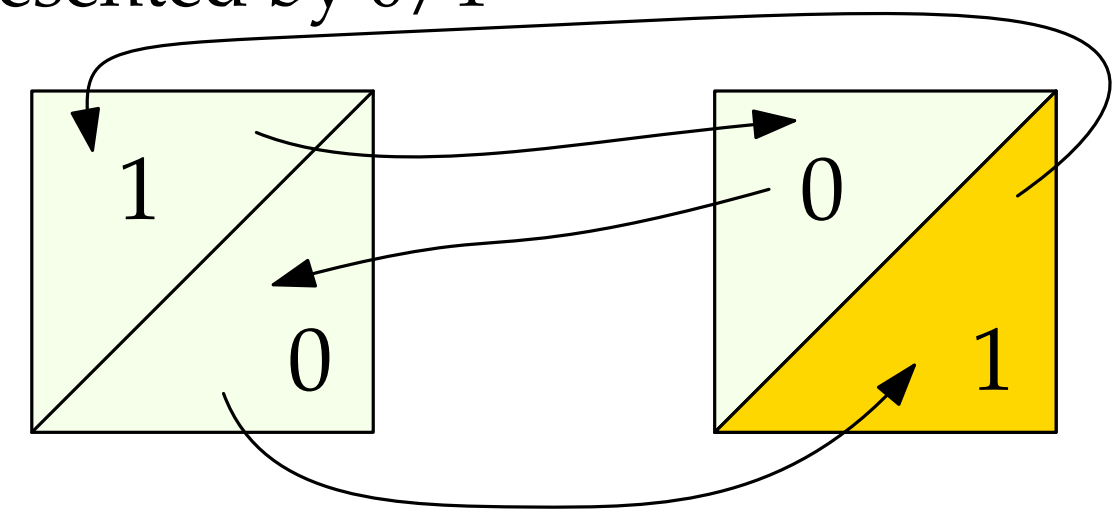


The golden 1 is extremely valuable

If the left square is in state 1 (represented by 1/0) then applying σ would put a 0 into the golden cell.

A counter with constantly many permutations

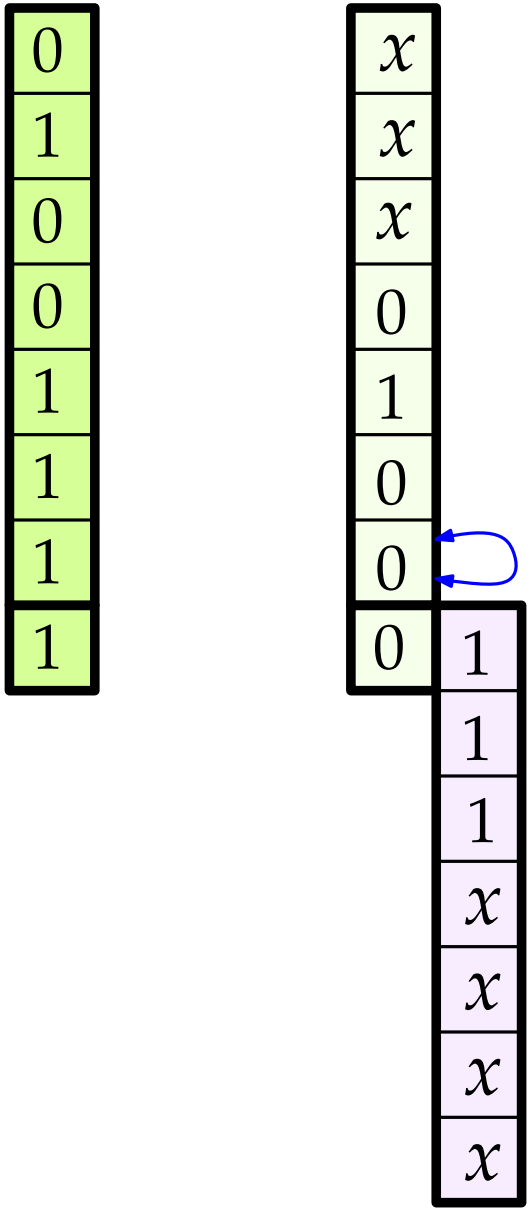
0 is represented by 0/1



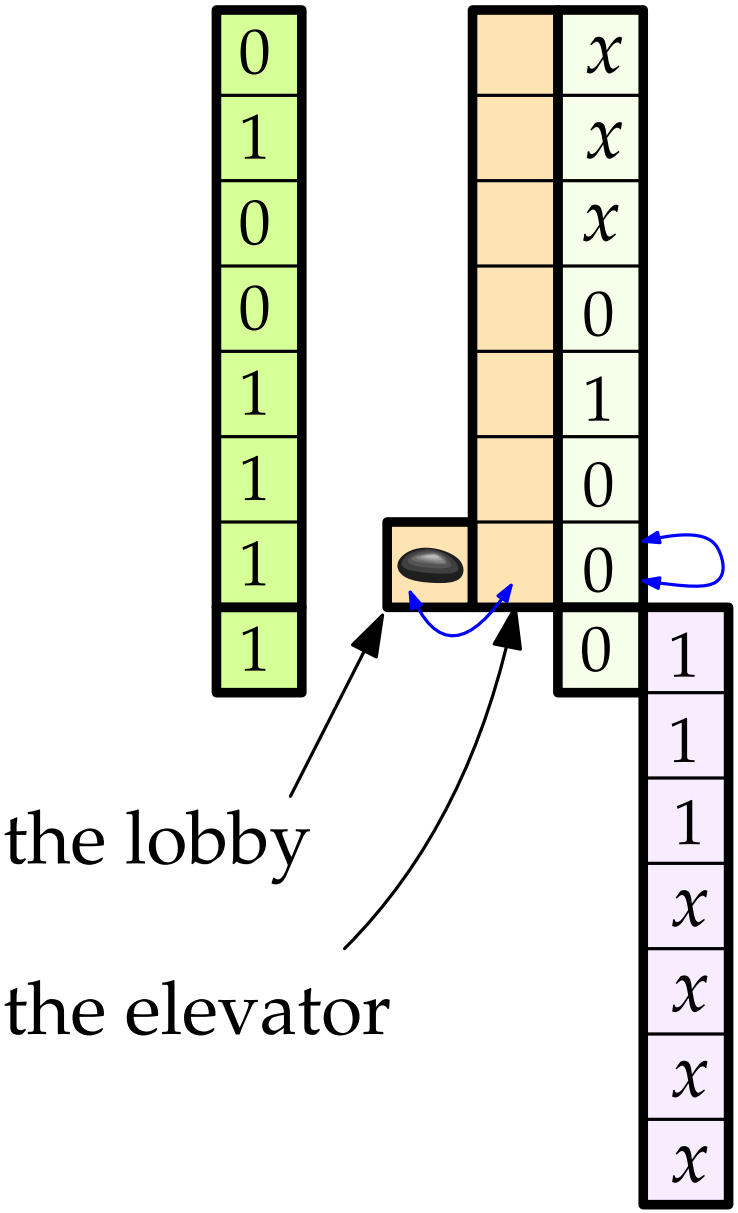
The golden 1 is extremely valuable

Actually we will use more than two positions per cell, to represent the value x. Maybe 3 positions are enough. A similar construction will work in this case.

A counter with constantly many permutations



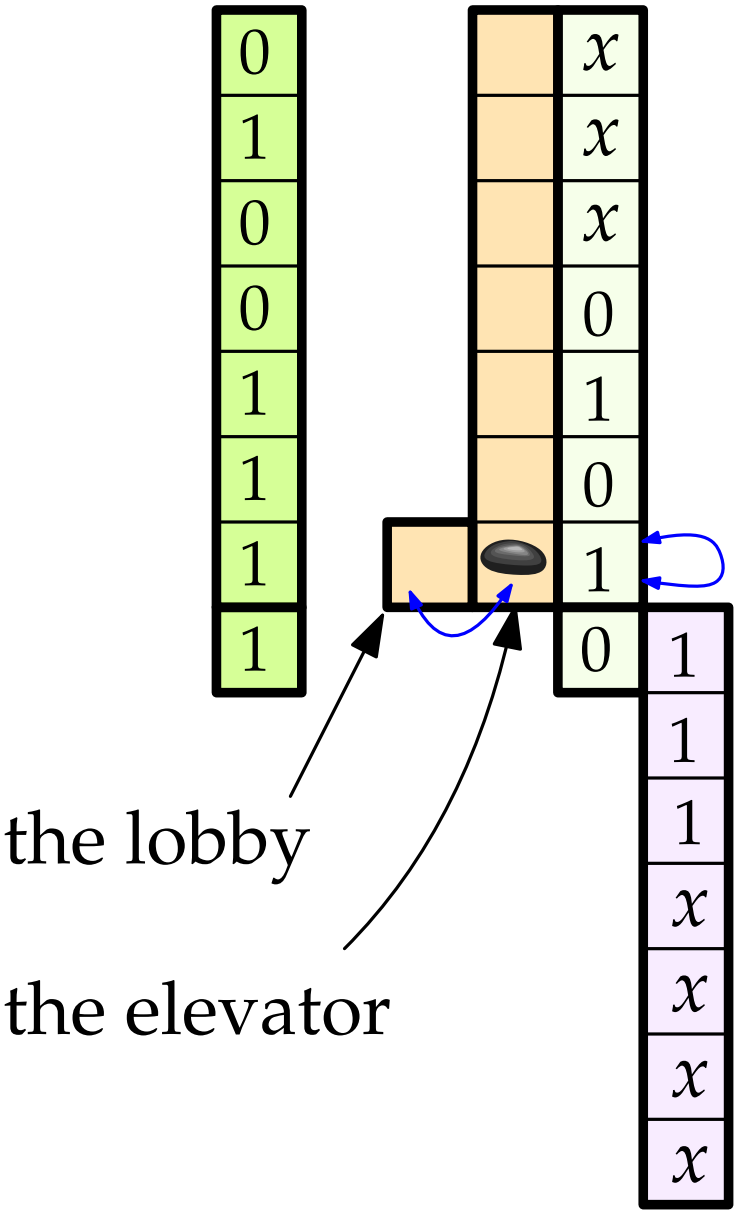
A counter with constantly many permutations



Furthermore, σ moves the pebble from the "lobby" into the "elevator".

This is only possible if pebble really is in the lobby. Otherwise it would give a payoff of $-\infty$.

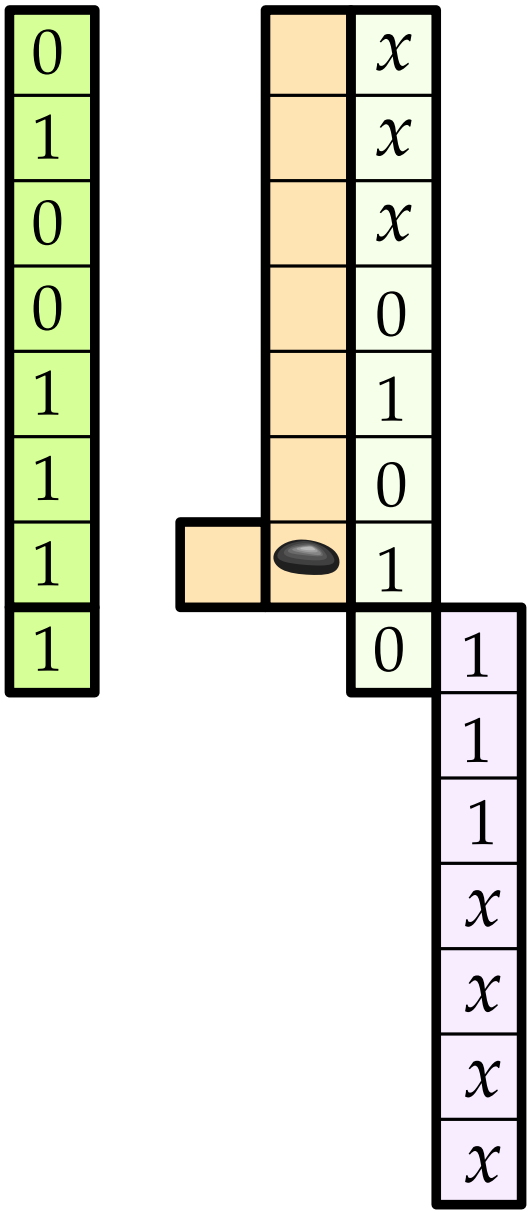
A counter with constantly many permutations



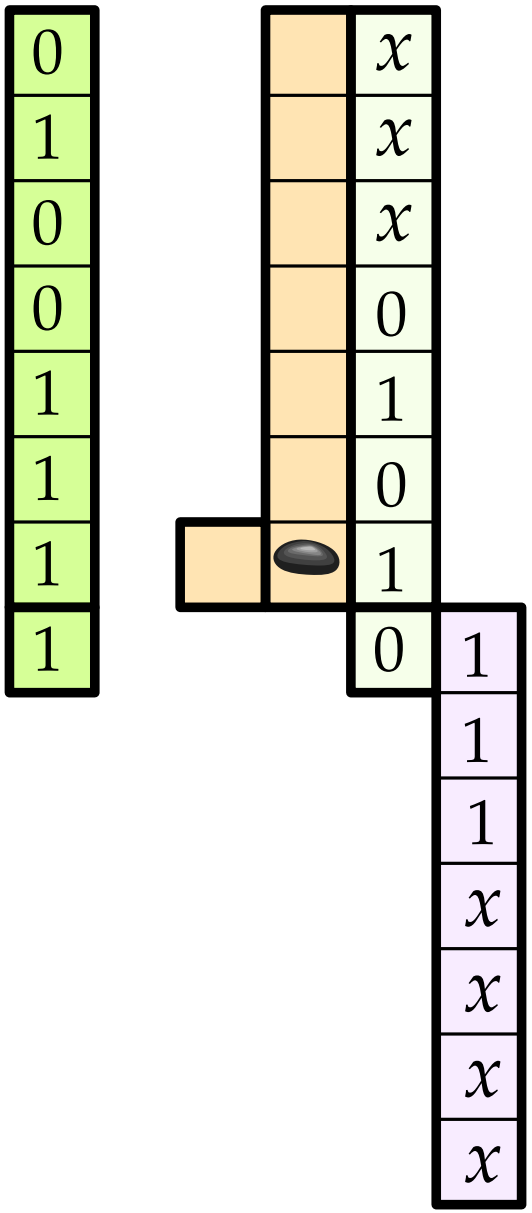
Furthermore, σ moves the pebble from the "lobby" into the "elevator".

This is only possible if pebble really is in the lobby. Otherwise it would give a payoff of $-\infty$.

A counter with constantly many permutations

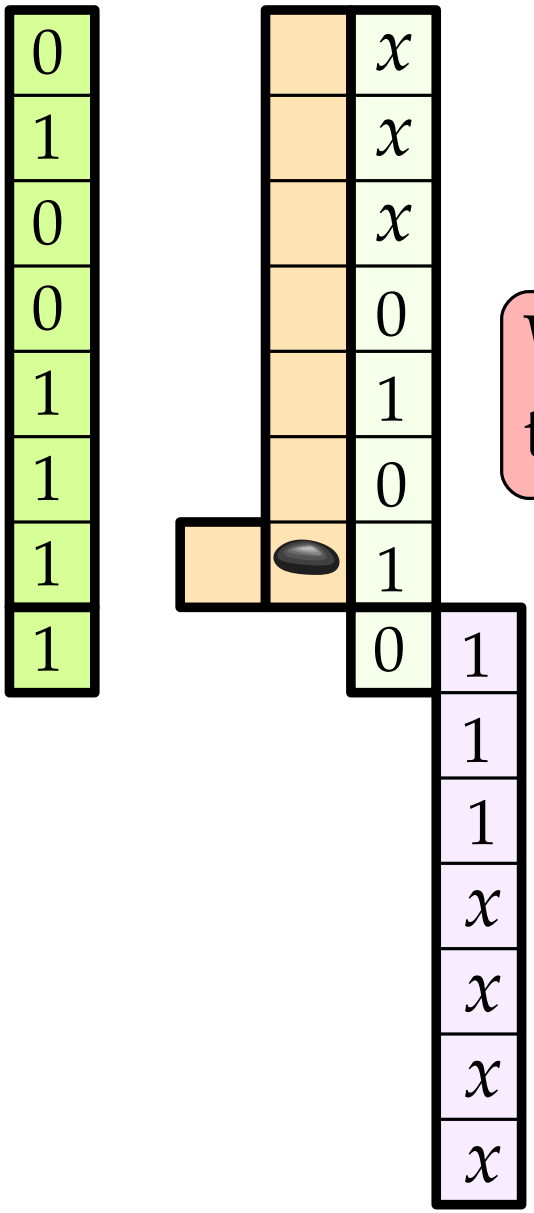


A counter with constantly many permutations



Wait, can't we apply π again und push the new 1 down?

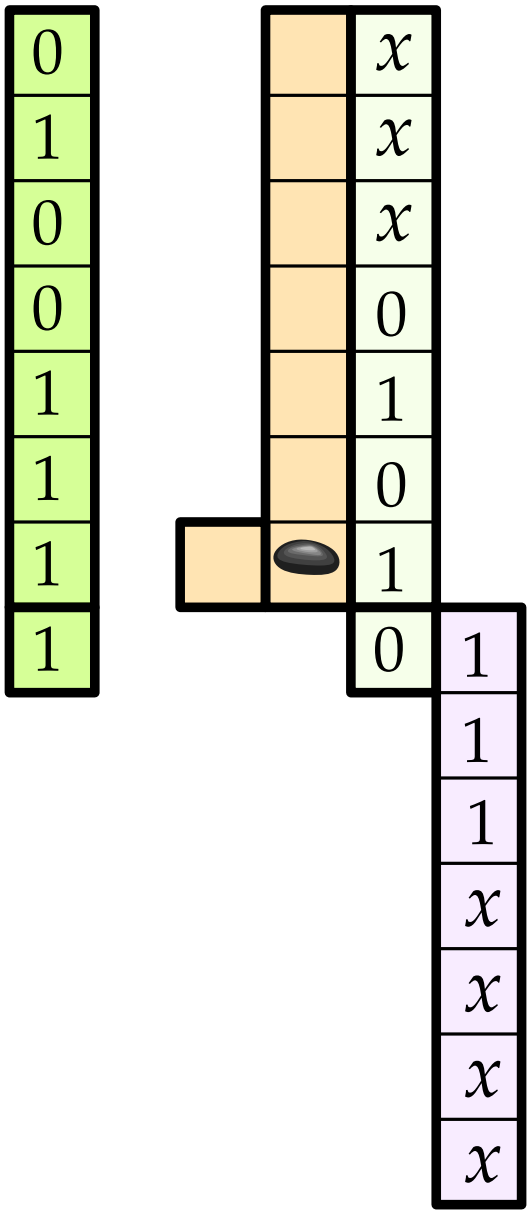
A counter with constantly many permutations



Wait, can't we apply π again und push the new 1 down?

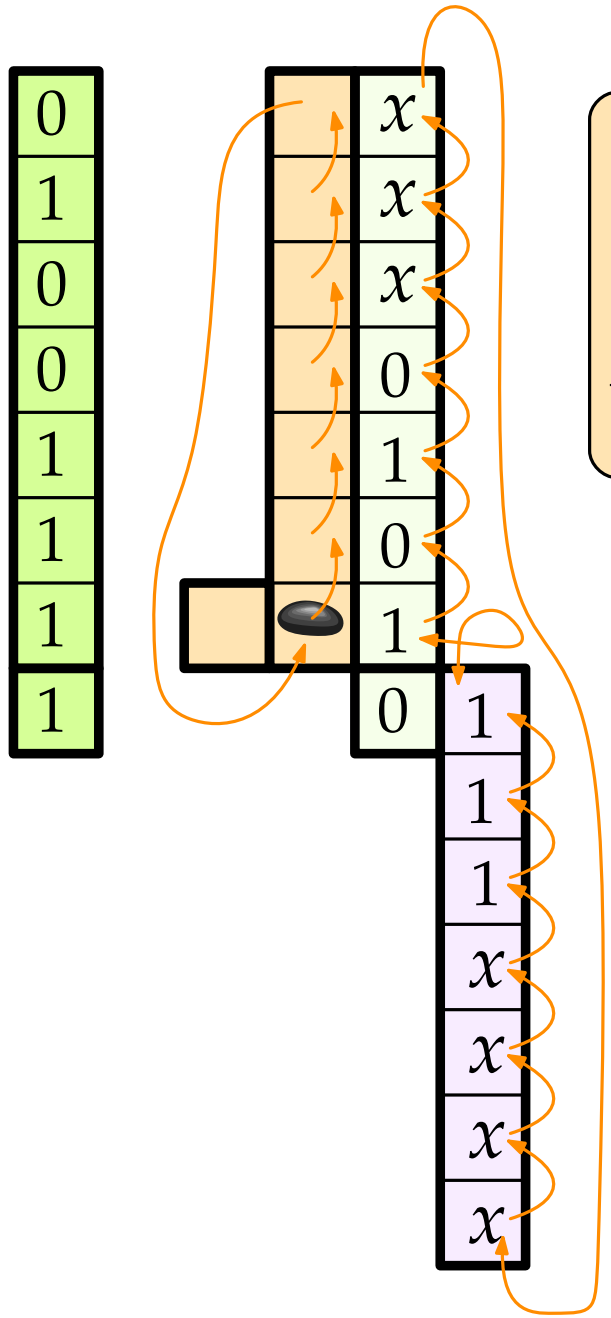
We need to add to π a check that the elevator is completely empty!

A counter with constantly many permutations



The higher the stone, the higher the payoff. This is stronger than the violet buffer wanting 1's.

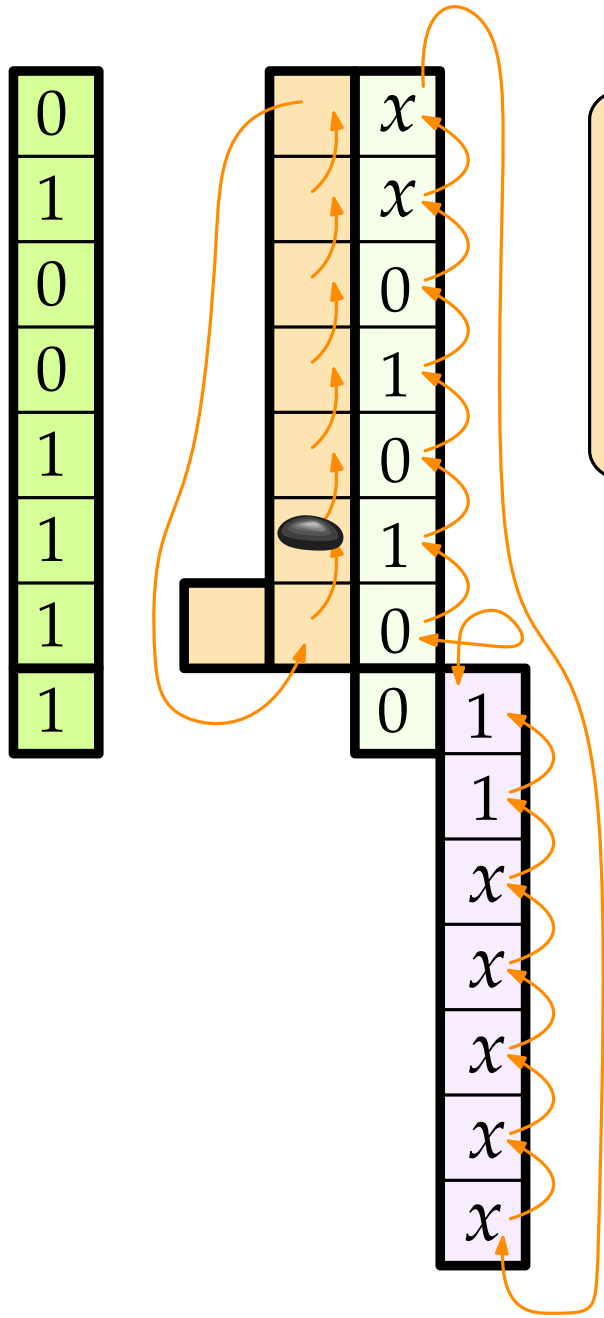
A counter with constantly many permutations



The higher the stone, the higher the payoff. This is stronger than the violet buffer wanting 1's.

Permutation τ pushes up the pebble, the buffer, and the right register. It can work against the buffer's love for 1's because lifting up the pebble creates a higher payoff. Also, it inverts the 1 that goes from buffer to register.

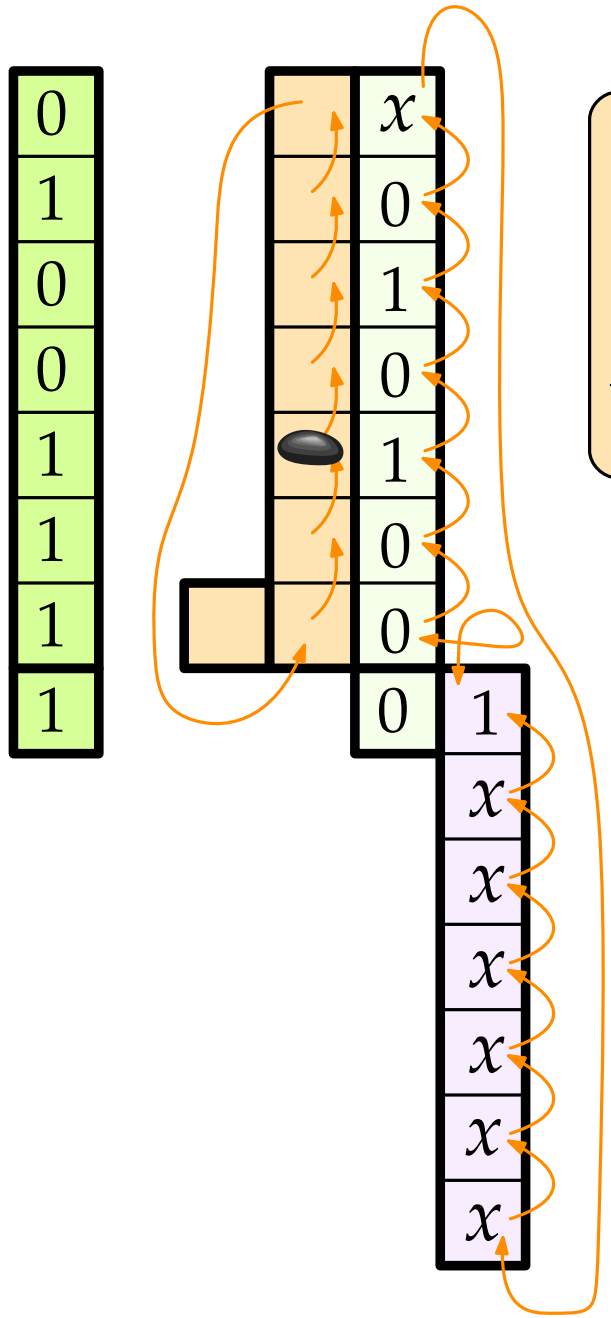
A counter with constantly many permutations



The higher the stone, the higher the payoff. This is stronger than the violet buffer wanting 1's.

Permutation τ pushes up the pebble, the buffer, and the right register. It can work against the buffer's love for 1's because lifting up the pebble creates a higher payoff. Also, it inverts the 1 that goes from buffer to register.

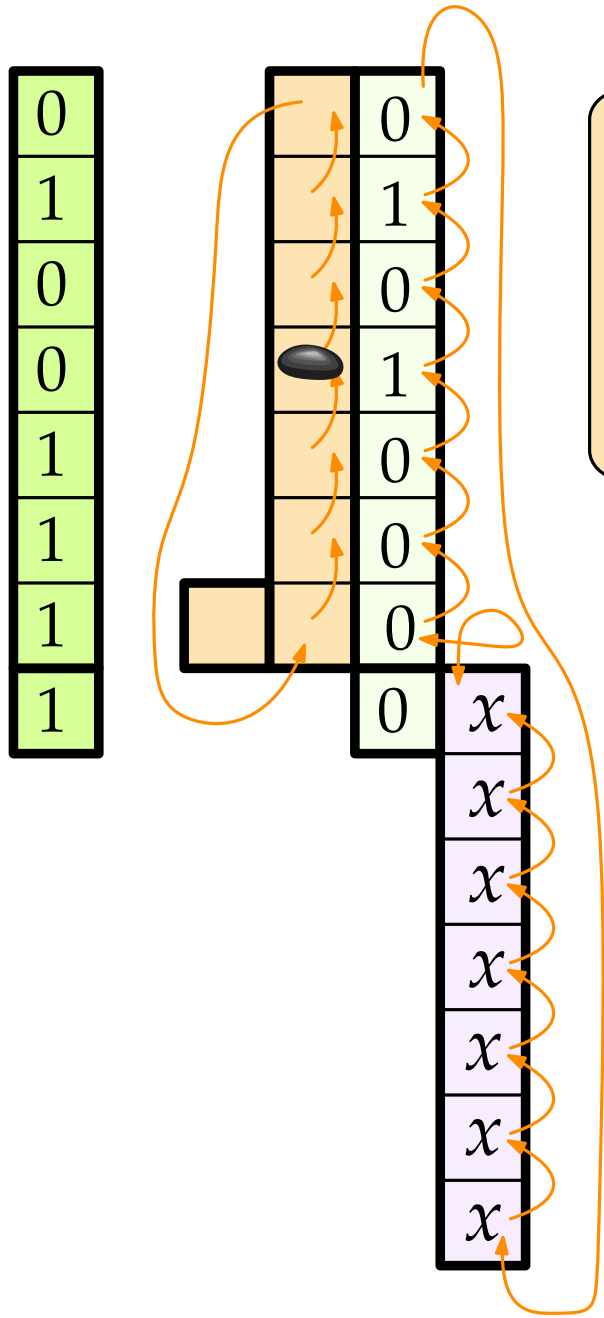
A counter with constantly many permutations



The higher the stone, the higher the payoff. This is stronger than the violet buffer wanting 1's.

Permutation τ pushes up the pebble, the buffer, and the right register. It can work against the buffer's love for 1's because lifting up the pebble creates a higher payoff. Also, it inverts the 1 that goes from buffer to register.

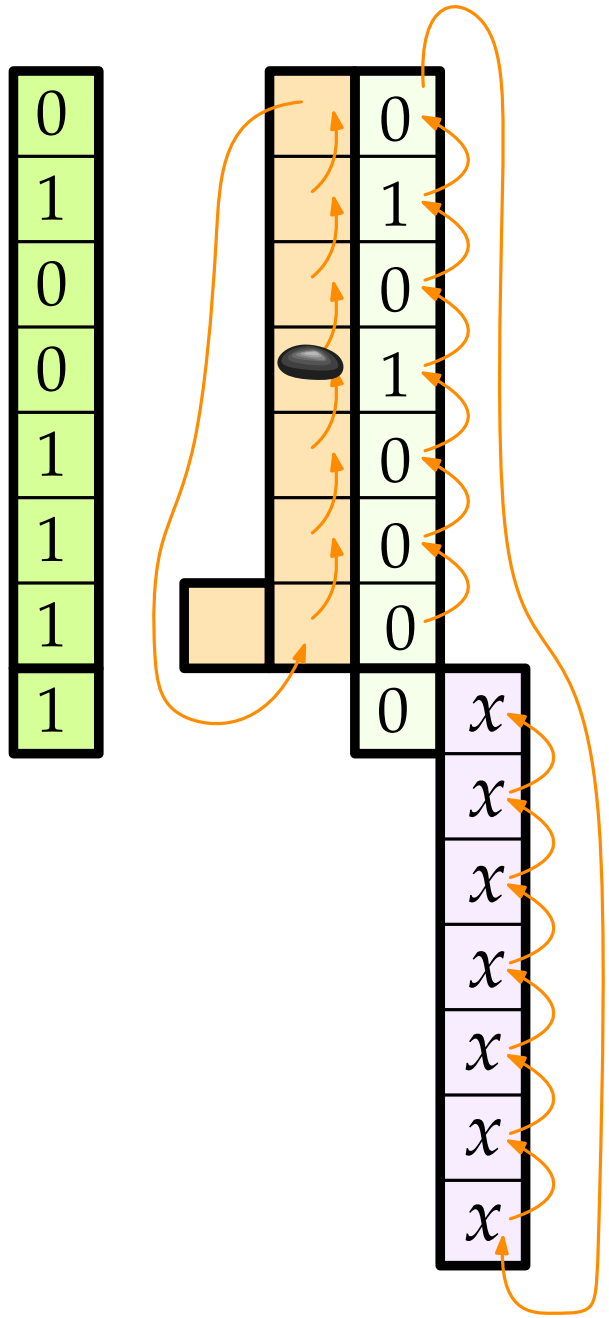
A counter with constantly many permutations



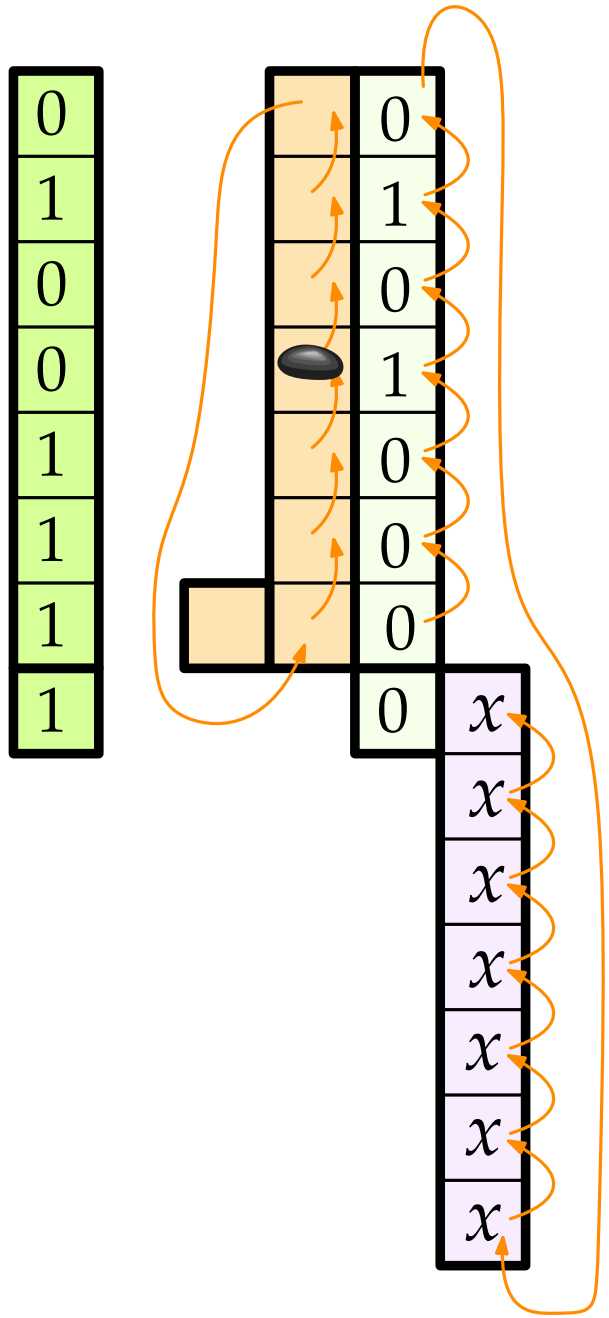
The higher the stone, the higher the payoff. This is stronger than the violet buffer wanting 1's.

Permutation τ pushes up the pebble, the buffer, and the right register. It can work against the buffer's love for 1's because lifting up the pebble creates a higher payoff. Also, it inverts the 1 that goes from buffer to register.

A counter with constantly many permutations

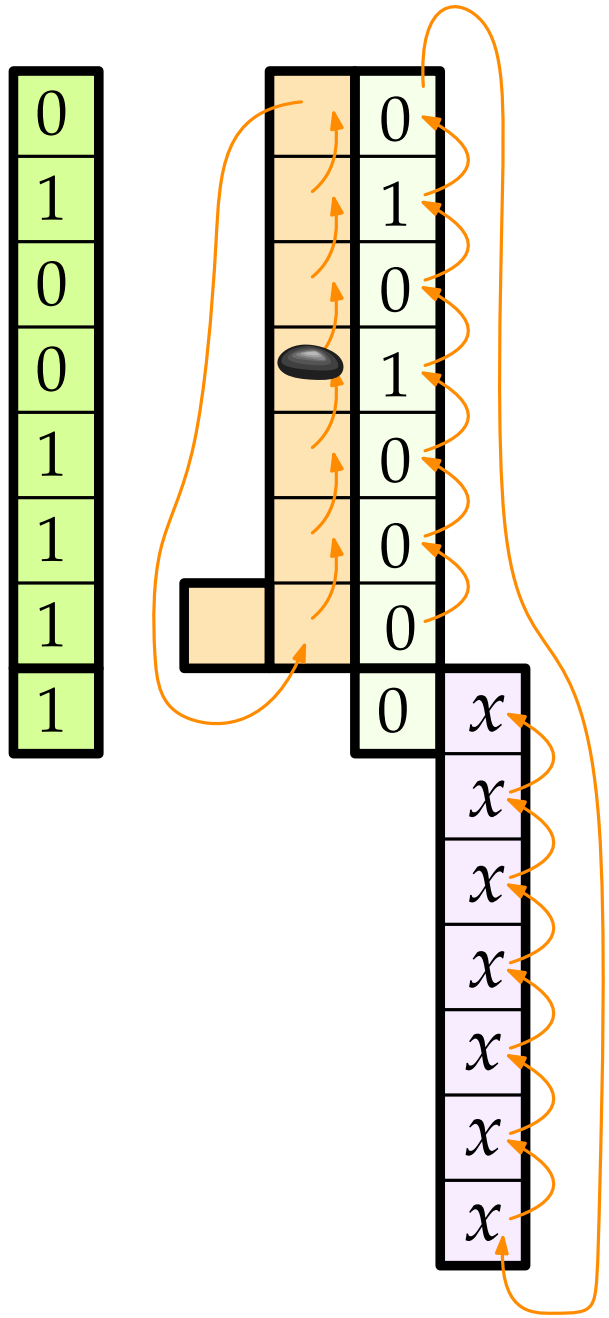


A counter with constantly many permutations



Cannot apply τ again because x in the right register causes payoff $-\infty$.

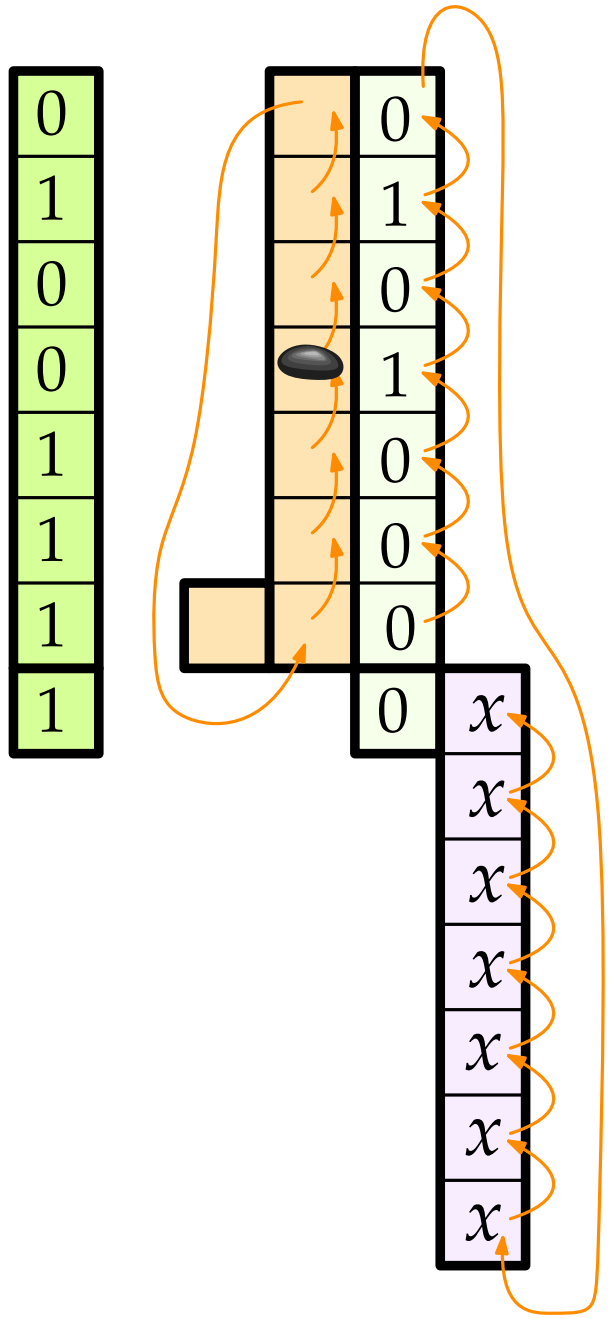
A counter with constantly many permutations



Cannot apply τ again because x in the right register causes payoff $-\infty$.

Cannot apply σ because σ checks that the pebble is in the lobby, otherwise payoff $-\infty$

A counter with constantly many permutations

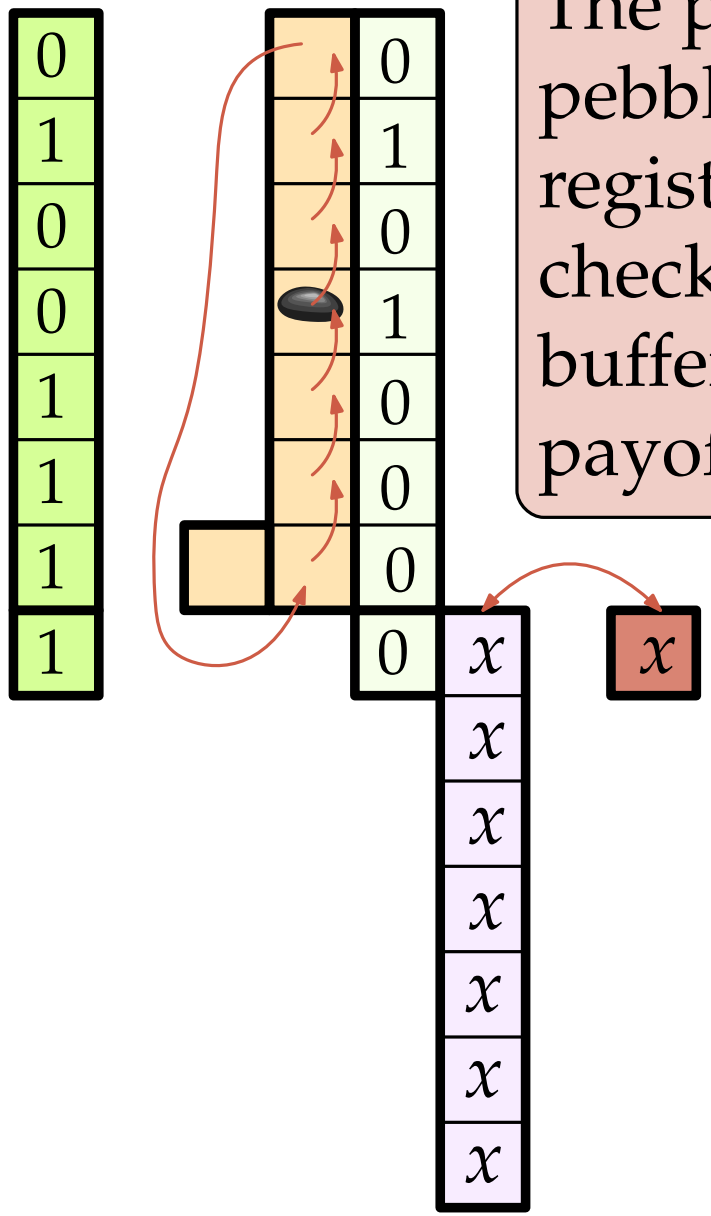


Cannot apply τ again because x in the right register causes payoff $-\infty$.

Cannot apply σ because σ checks that the pebble is in the lobby, otherwise payoff $-\infty$

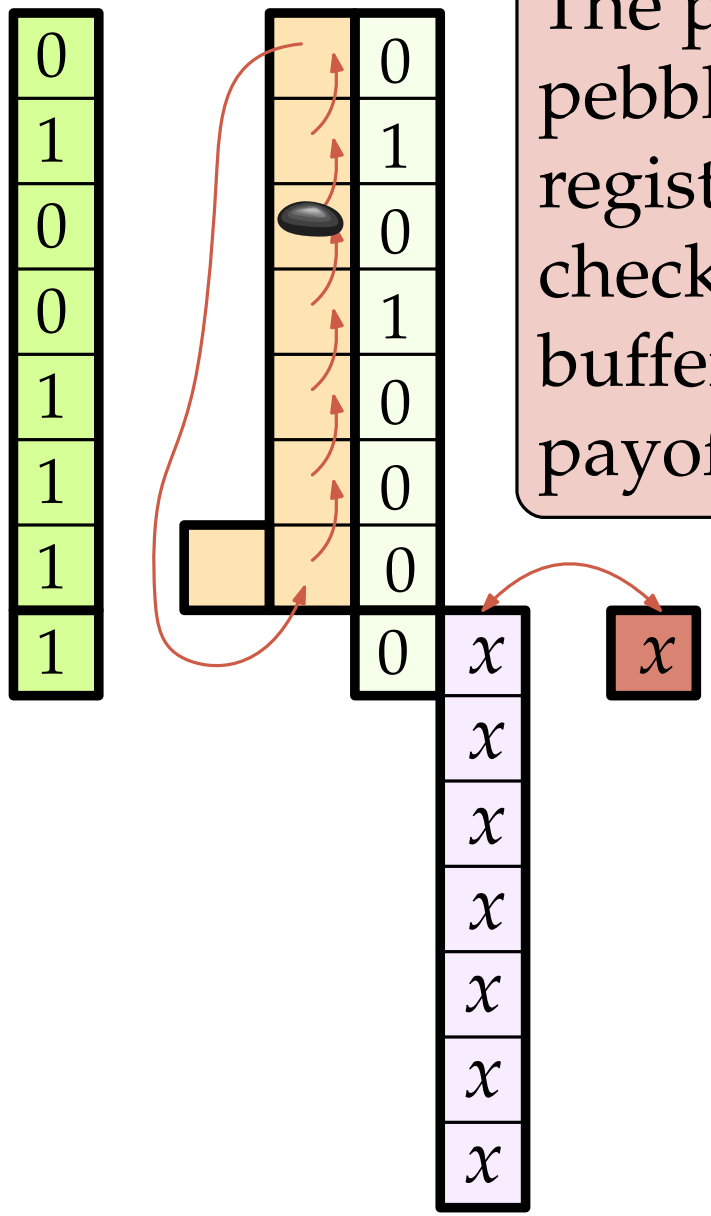
Cannot apply π because π would push the pebble down the elevator (we didn't show the elevator when introducing π).

A counter with constantly many permutations



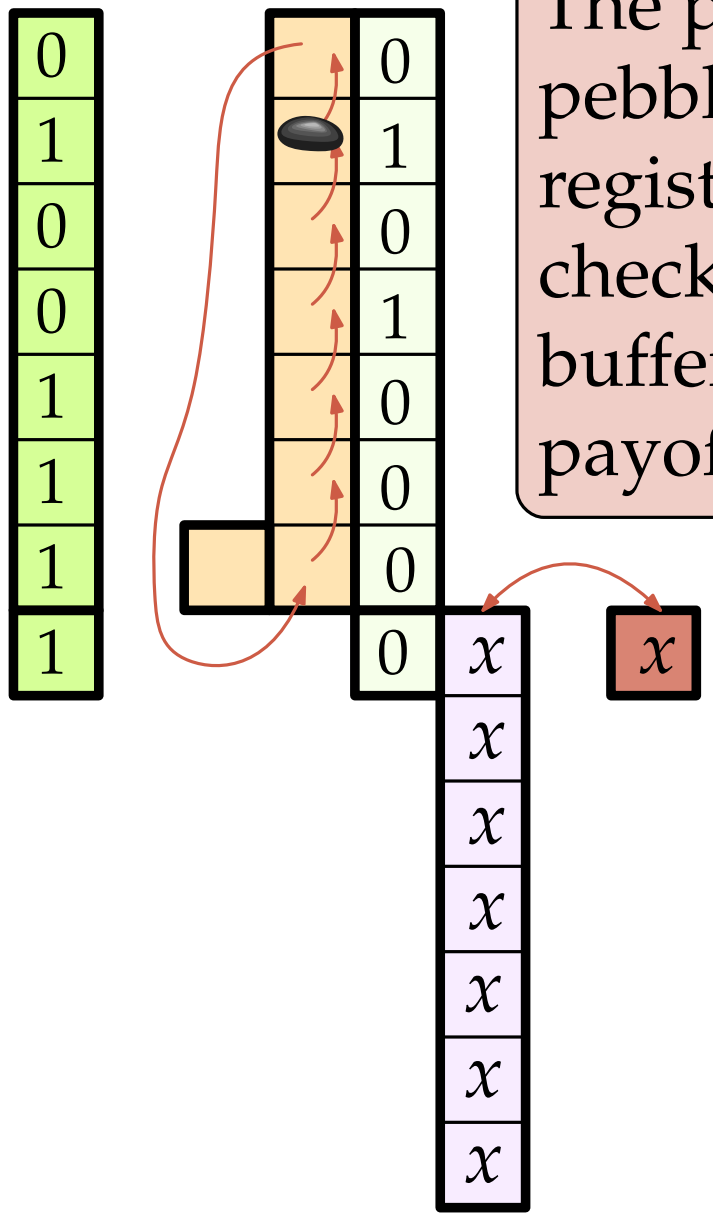
The permutation π' pushes the pebble up the elevator but leaves register and buffer unchanged. It checks whether the top cell of the buffer really is an x (if not then payoff $-\infty$).

A counter with constantly many permutations



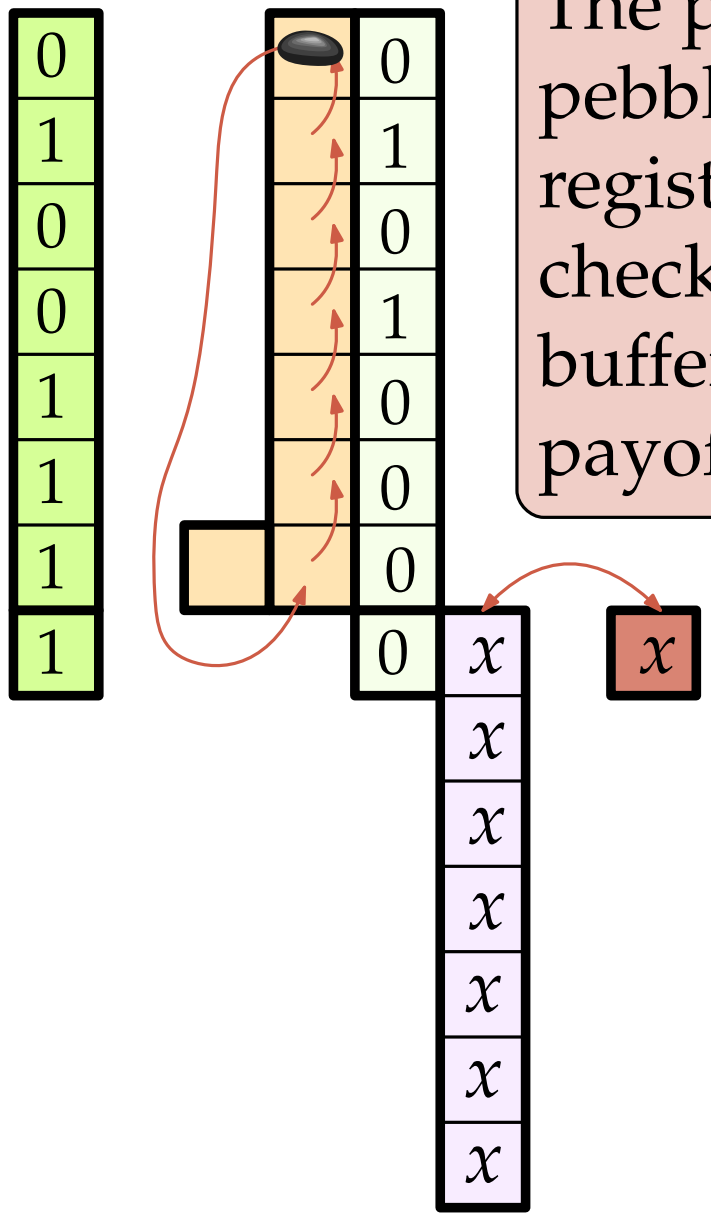
The permutation π' pushes the pebble up the elevator but leaves register and buffer unchanged. It checks whether the top cell of the buffer really is an x (if not then payoff $-\infty$).

A counter with constantly many permutations



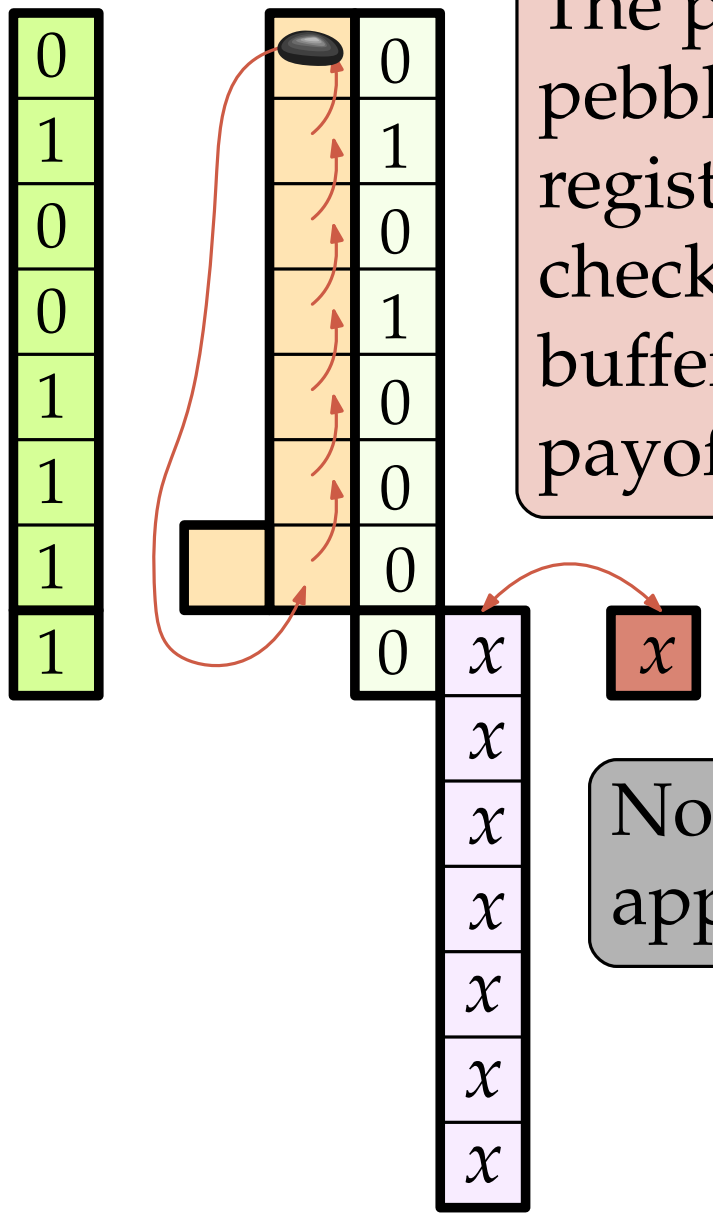
The permutation π' pushes the pebble up the elevator but leaves register and buffer unchanged. It checks whether the top cell of the buffer really is an x (if not then payoff $-\infty$).

A counter with constantly many permutations



The permutation π' pushes the pebble up the elevator but leaves register and buffer unchanged. It checks whether the top cell of the buffer really is an x (if not then payoff $-\infty$).

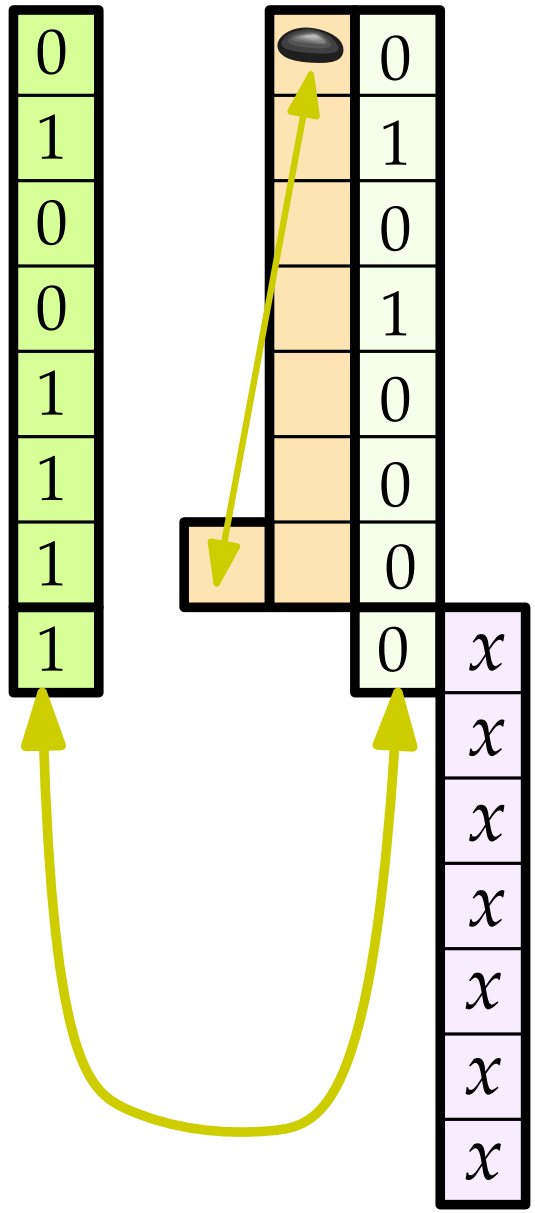
A counter with constantly many permutations



The permutation π' pushes the pebble up the elevator but leaves register and buffer unchanged. It checks whether the top cell of the buffer really is an x (if not then payoff $-\infty$).

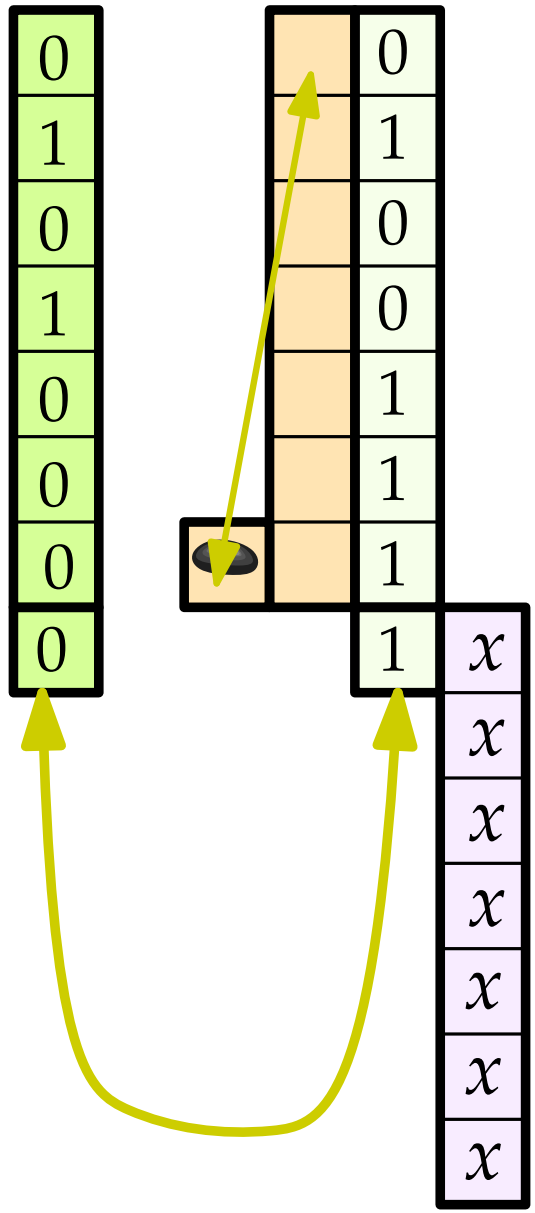
No permutation can be applied anymore.

A counter with constantly many permutations

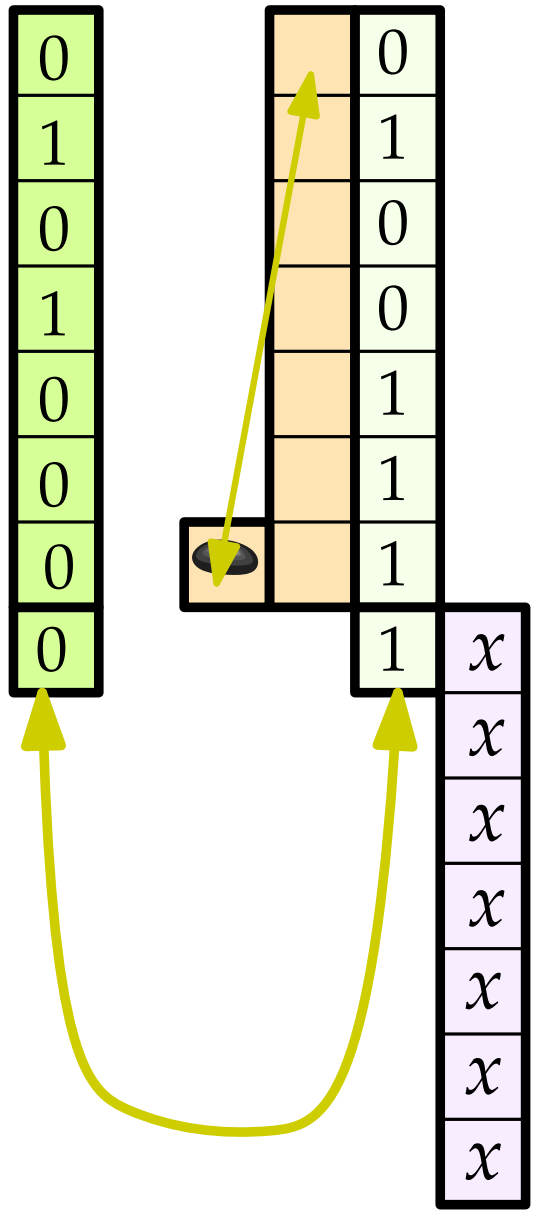


Now comes the final permutation: α. It exchanges left and right register and places the pebble back into the lobby. This increases the value if the right register encodes a higher value than the left register; note that the elevator is worth less than the left register.

A counter with constantly many permutations



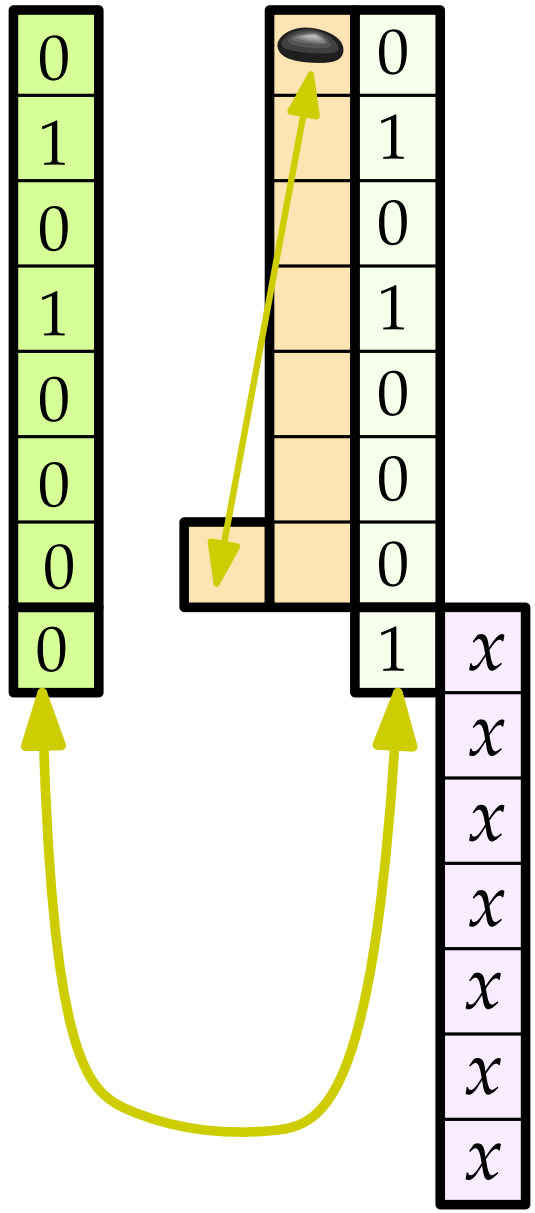
A counter with constantly many permutations



Now repeat

1. π: pushing down
2. σ: inverting the buffer and lowest register cell
3. τ, τ': pushing up

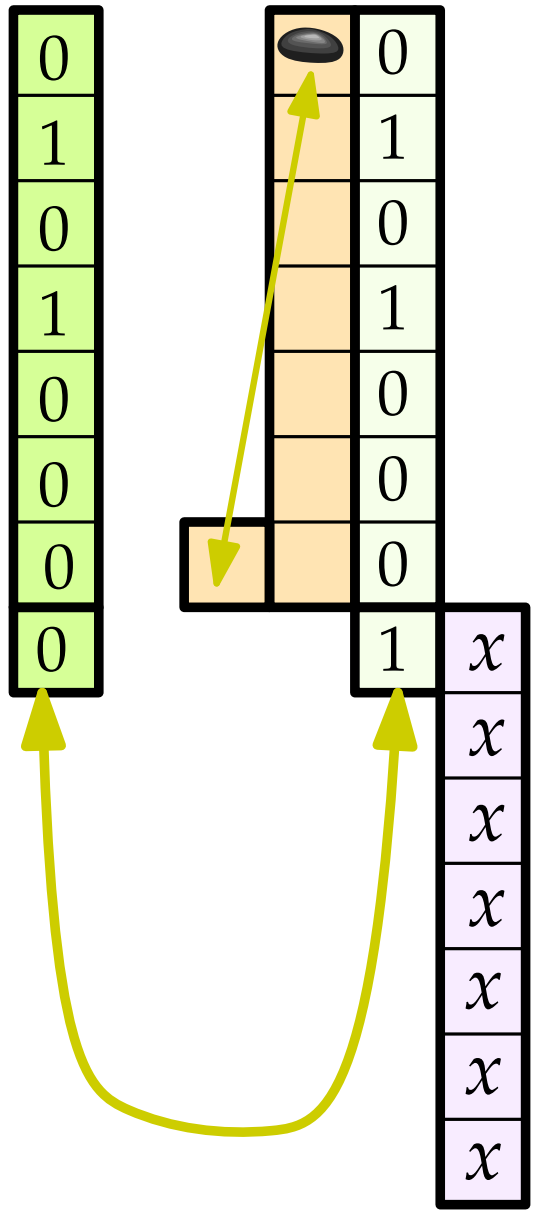
A counter with constantly many permutations



Now repeat

1. π: pushing down
2. σ: inverting the buffer and lowest register cell
3. τ, τ': pushing up

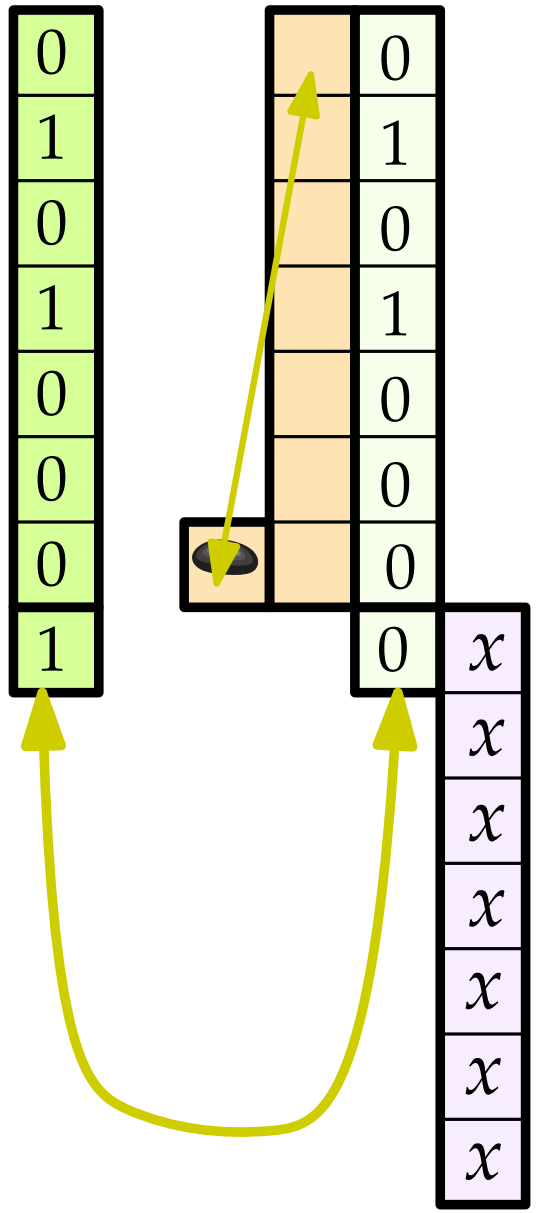
A counter with constantly many permutations



Now repeat

1. π: pushing down
2. σ: inverting the buffer and lowest register cell
3. τ, τ': pushing up
4. α: swap the registers and move pebble into lobby.

A counter with constantly many permutations

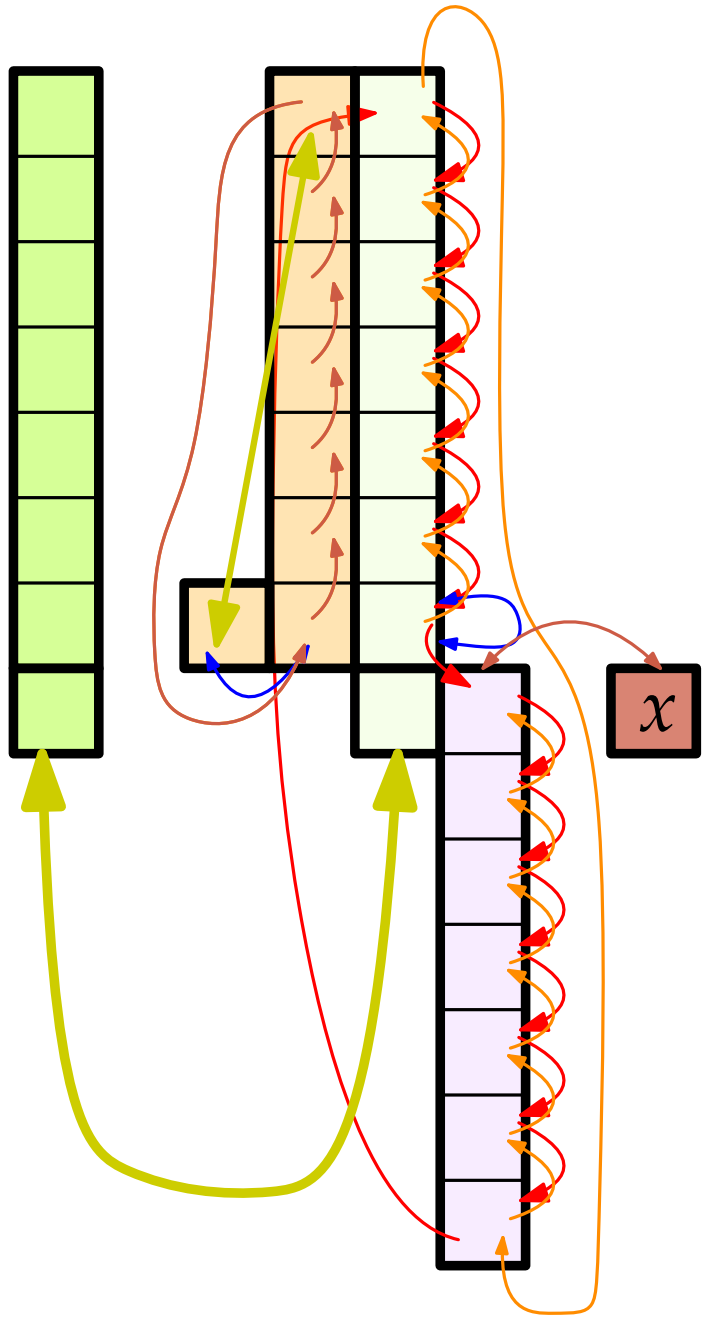


Now repeat

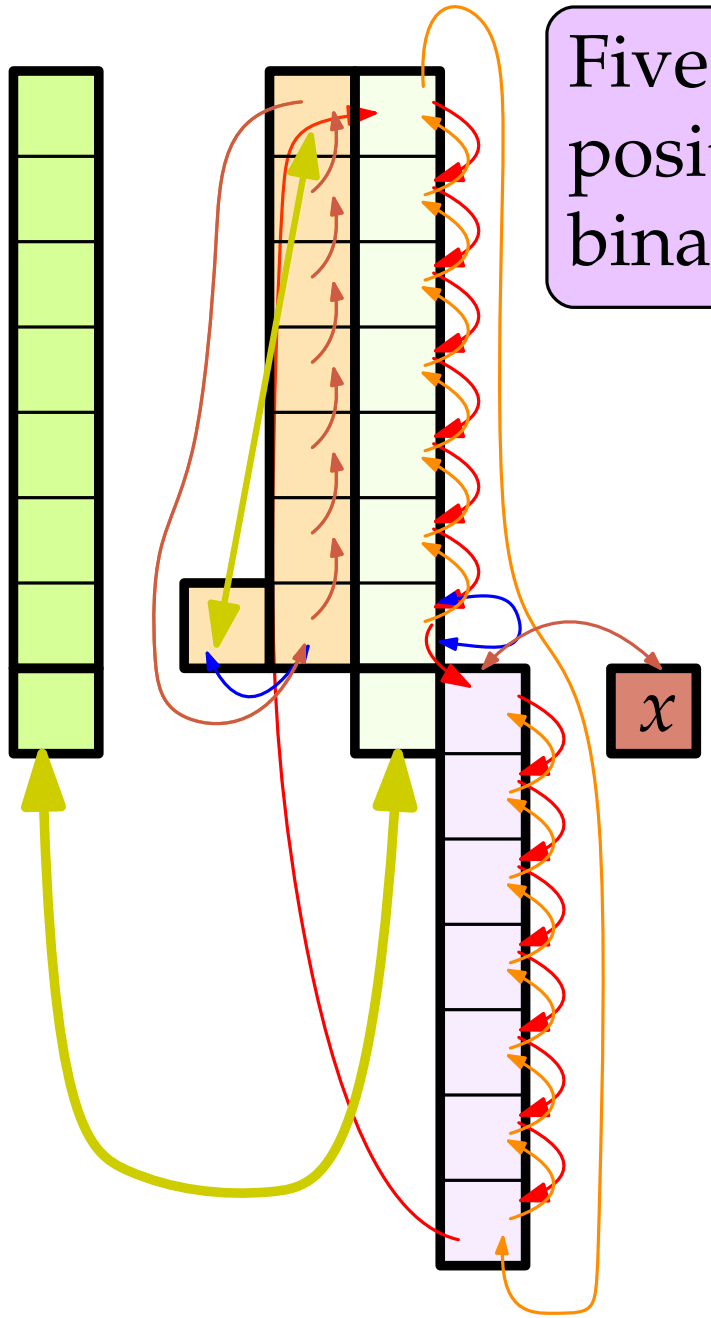
1. π: pushing down
2. σ: inverting the buffer and lowest register cell
3. τ, τ': pushing up
4. α: swap the registers and move pebble into lobby.

This is still an improving switch, but only because of the bit after the dot

A counter with constantly many permutations

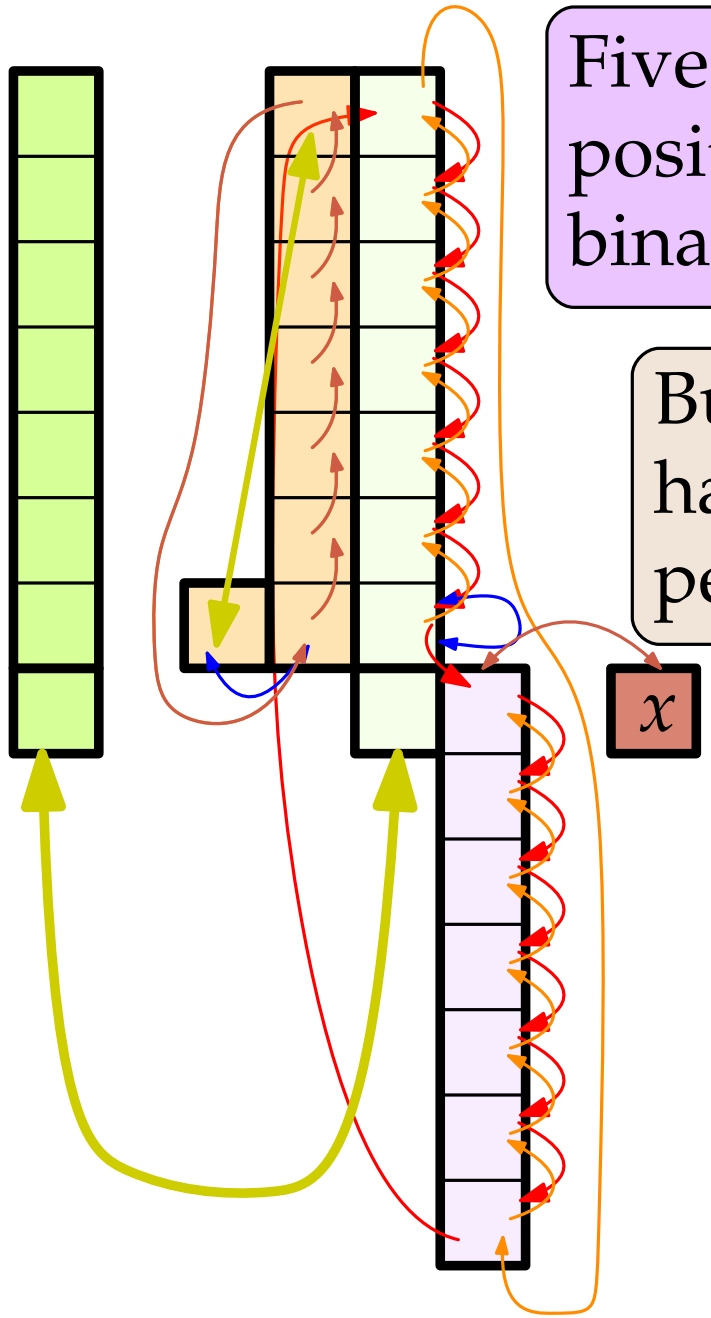


A counter with constantly many permutations



Five permutations and $O(n)$ positions suffice to build an n-bit binary counter.

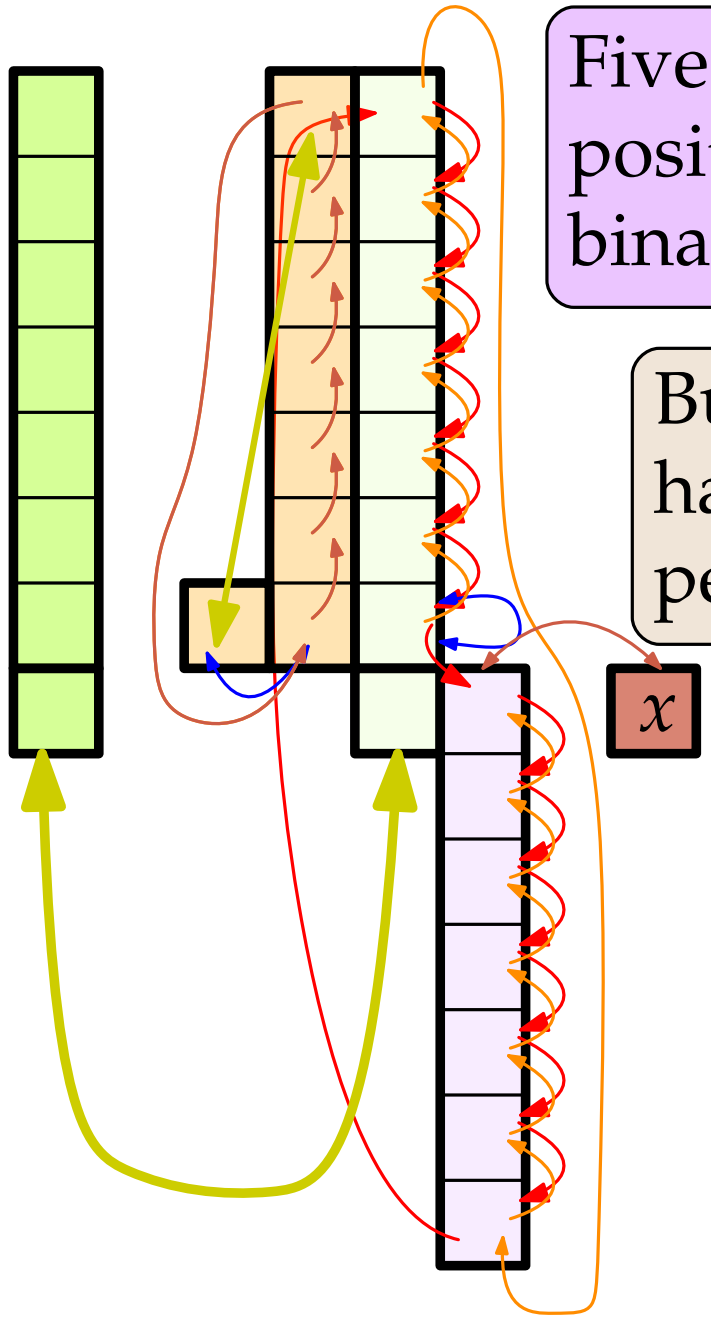
A counter with constantly many permutations



Five permutations and $O(n)$ positions suffice to build an n-bit binary counter.

But surely this would *never* happen for an Abelian group of permutations, would it?

A counter with constantly many permutations

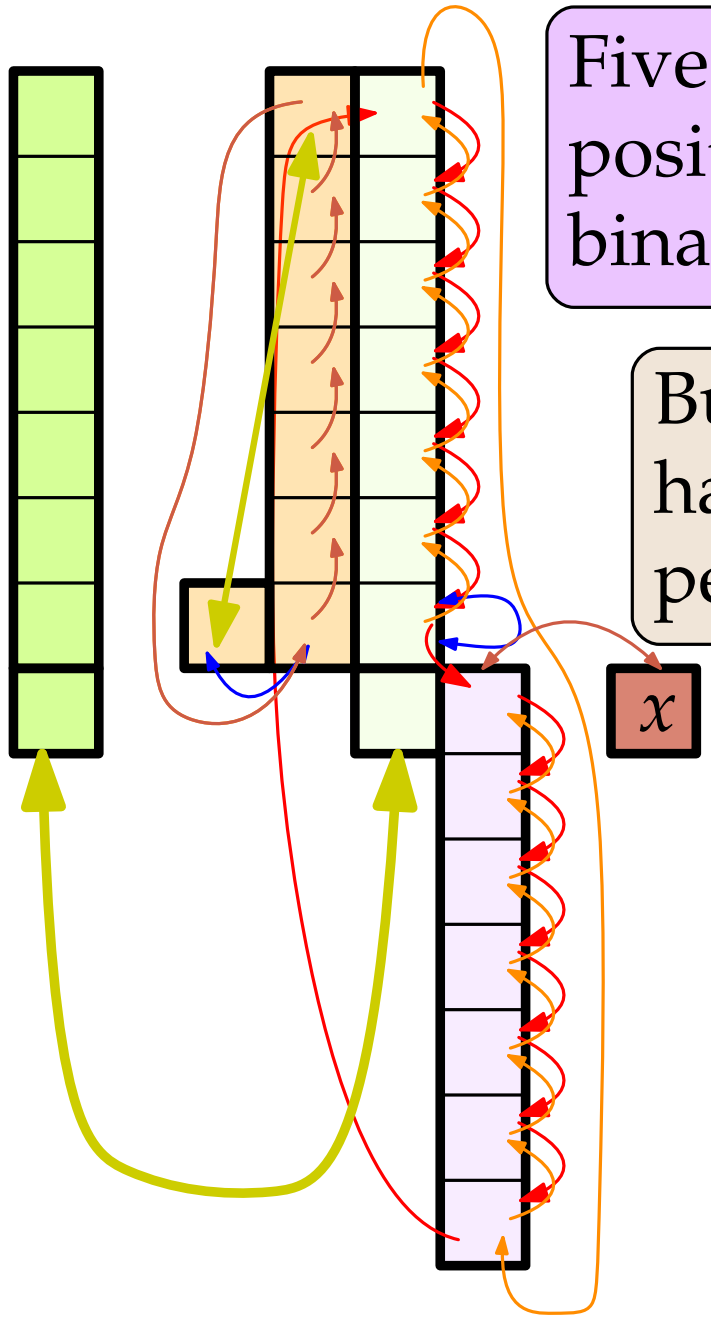


Five permutations and $O(n)$ positions suffice to build an n-bit binary counter.

But surely this would *never* happen for an Abelian group of permutations, would it?

...we don't know...

A counter with constantly many permutations



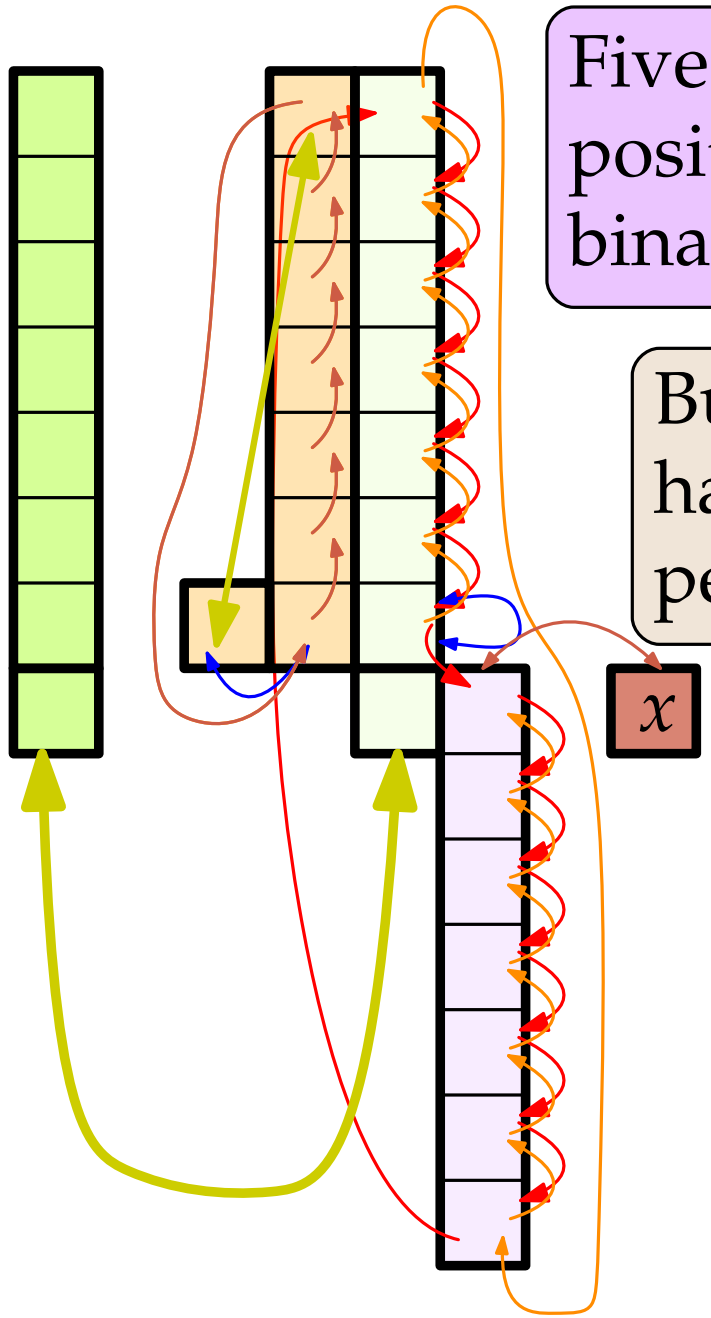
Five permutations and $O(n)$ positions suffice to build an n-bit binary counter.

But surely this would *never* happen for an Abelian group of permutations, would it?

...we don't know...

...even for just 2 permutations...

A counter with constantly many permutations



Five permutations and $O(n)$ positions suffice to build an n-bit binary counter.

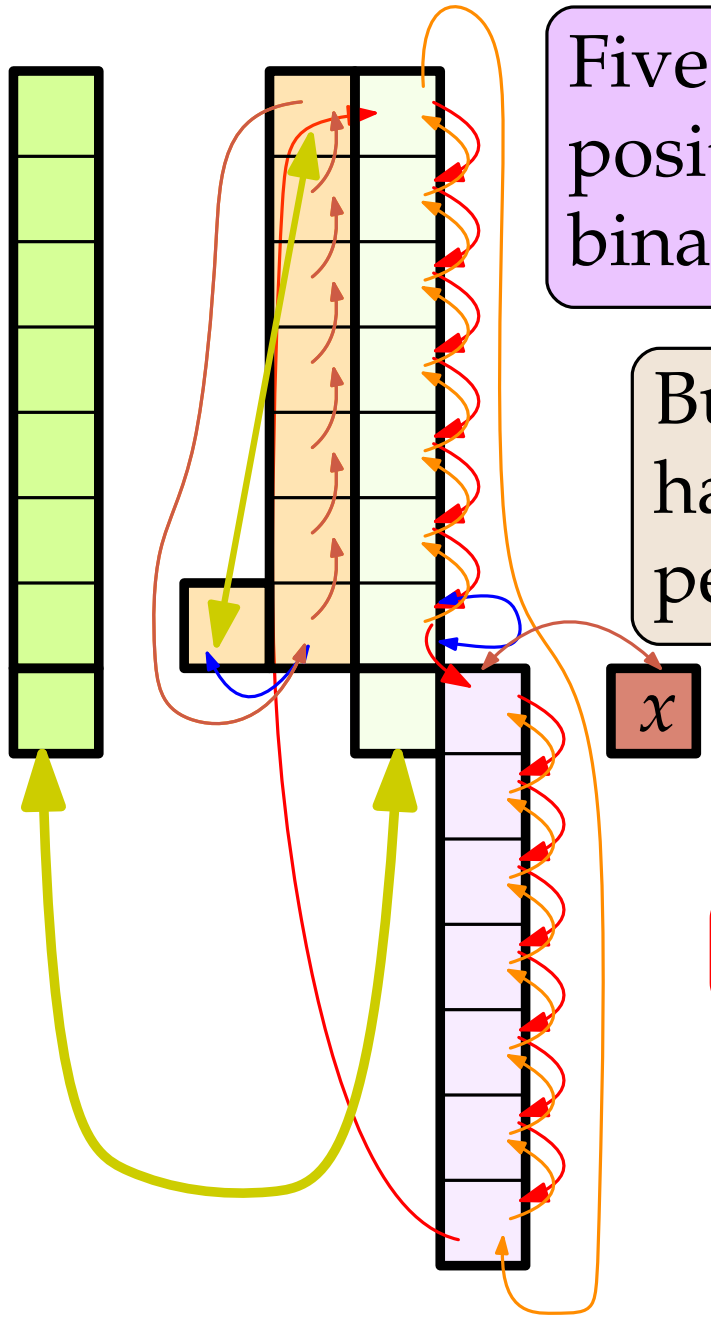
But surely this would *never* happen for an Abelian group of permutations, would it?

...we don't know...

...even for just 2 permutations...

... even for π and π^2...

A counter with constantly many permutations



Five permutations and $O(n)$ positions suffice to build an n-bit binary counter.

But surely this would *never* happen for an Abelian group of permutations, would it?

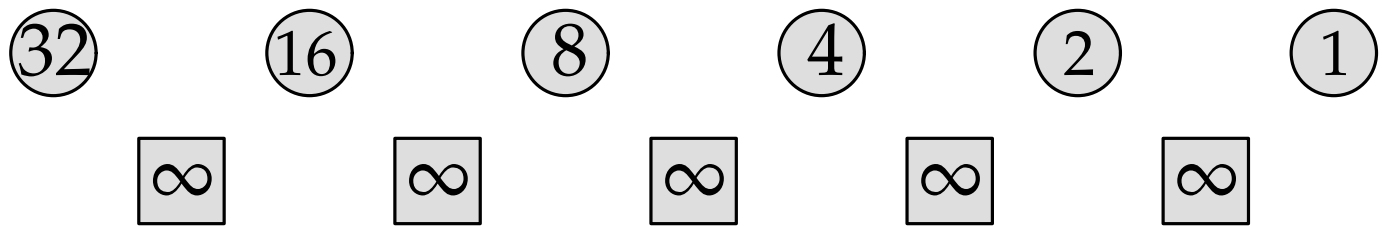
...we don't know...

...even for just 2 permutations...

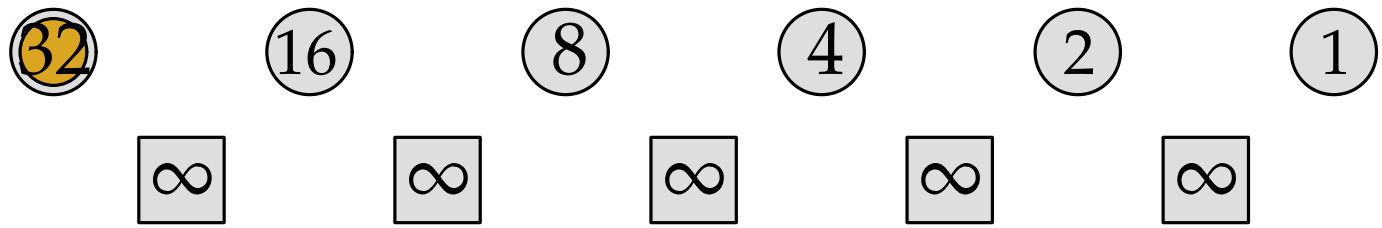
... even for π and π^2...

Quadratic Path for 2 Commuting Permutations

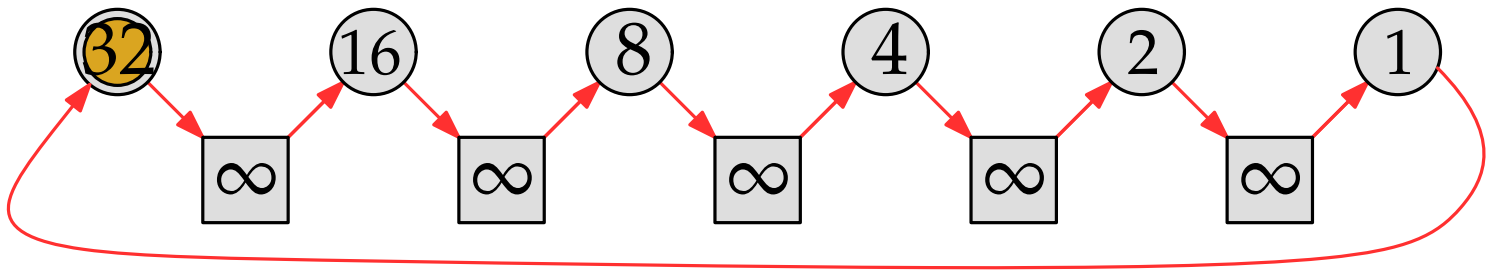
Quadratic Path for 2 Commuting Permutations



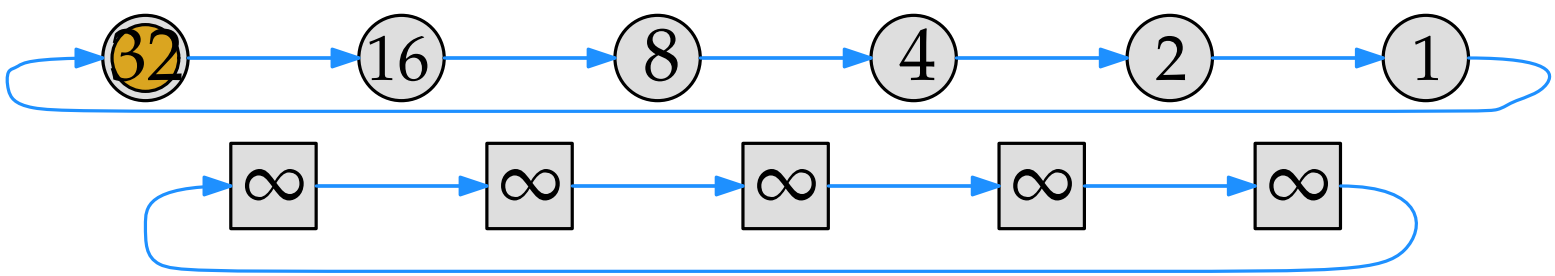
Quadratic Path for 2 Commuting Permutations



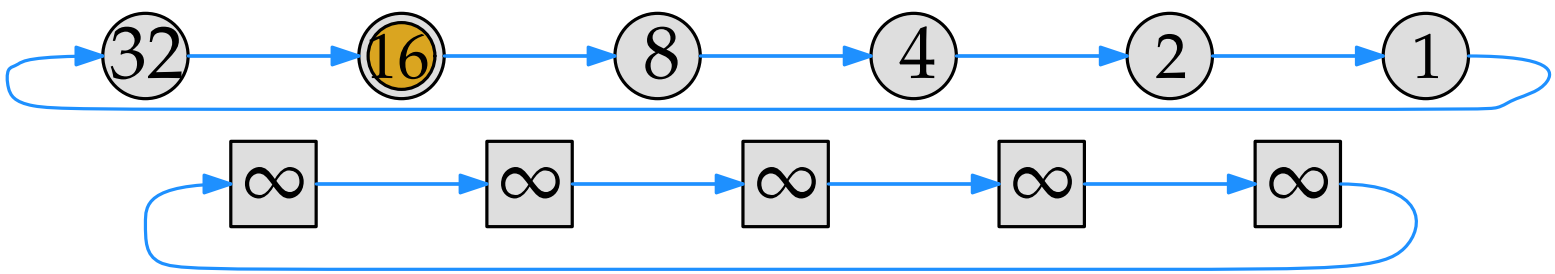
Quadratic Path for 2 Commuting Permutations



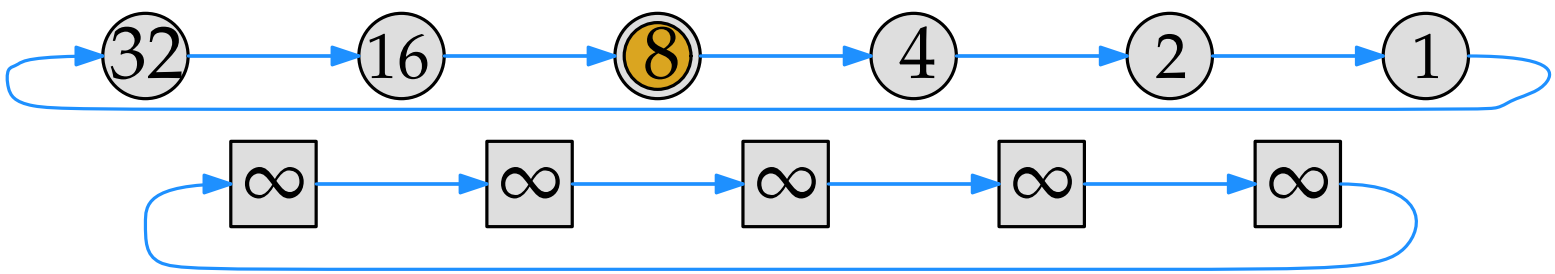
Quadratic Path for 2 Commuting Permutations



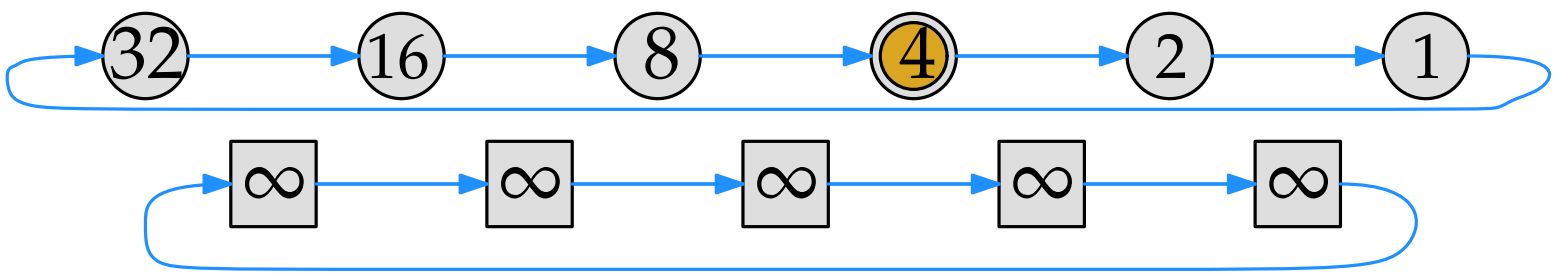
Quadratic Path for 2 Commuting Permutations



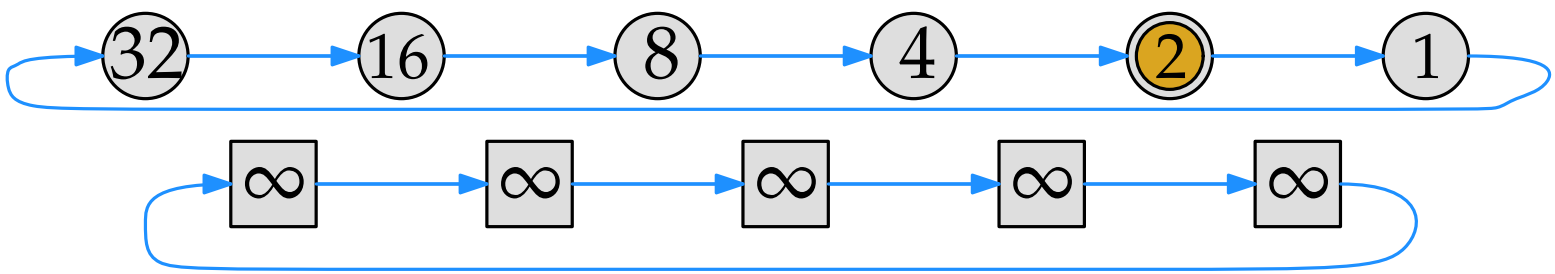
Quadratic Path for 2 Commuting Permutations



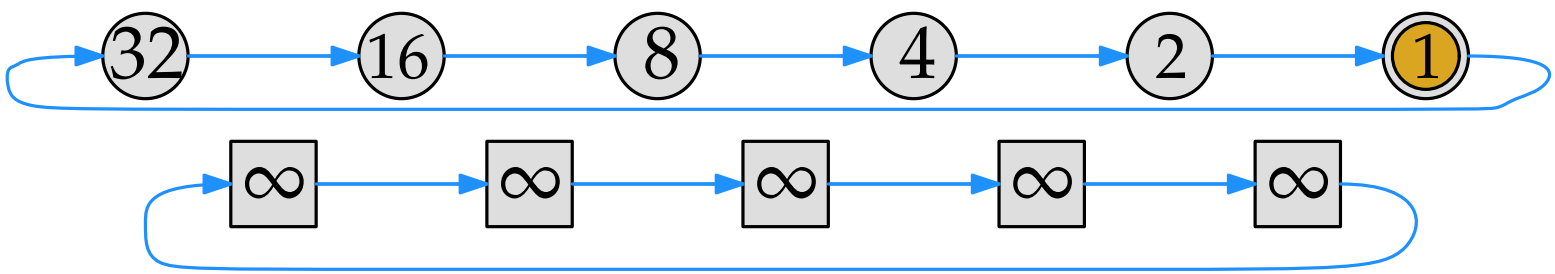
Quadratic Path for 2 Commuting Permutations



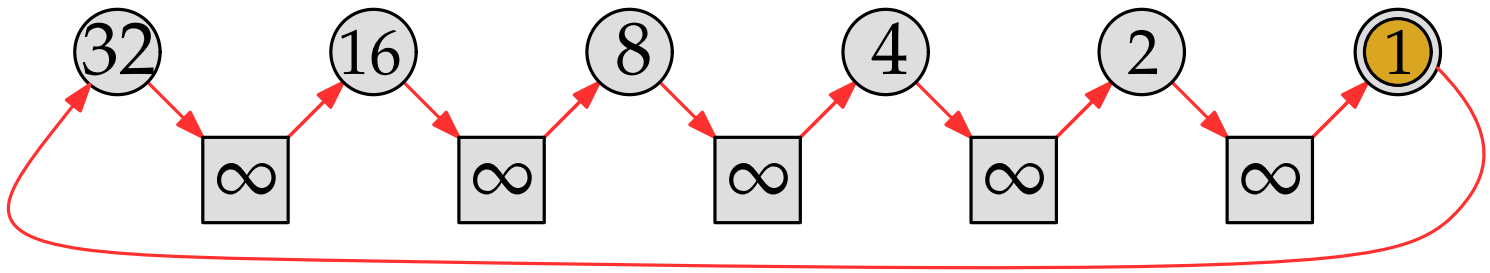
Quadratic Path for 2 Commuting Permutations



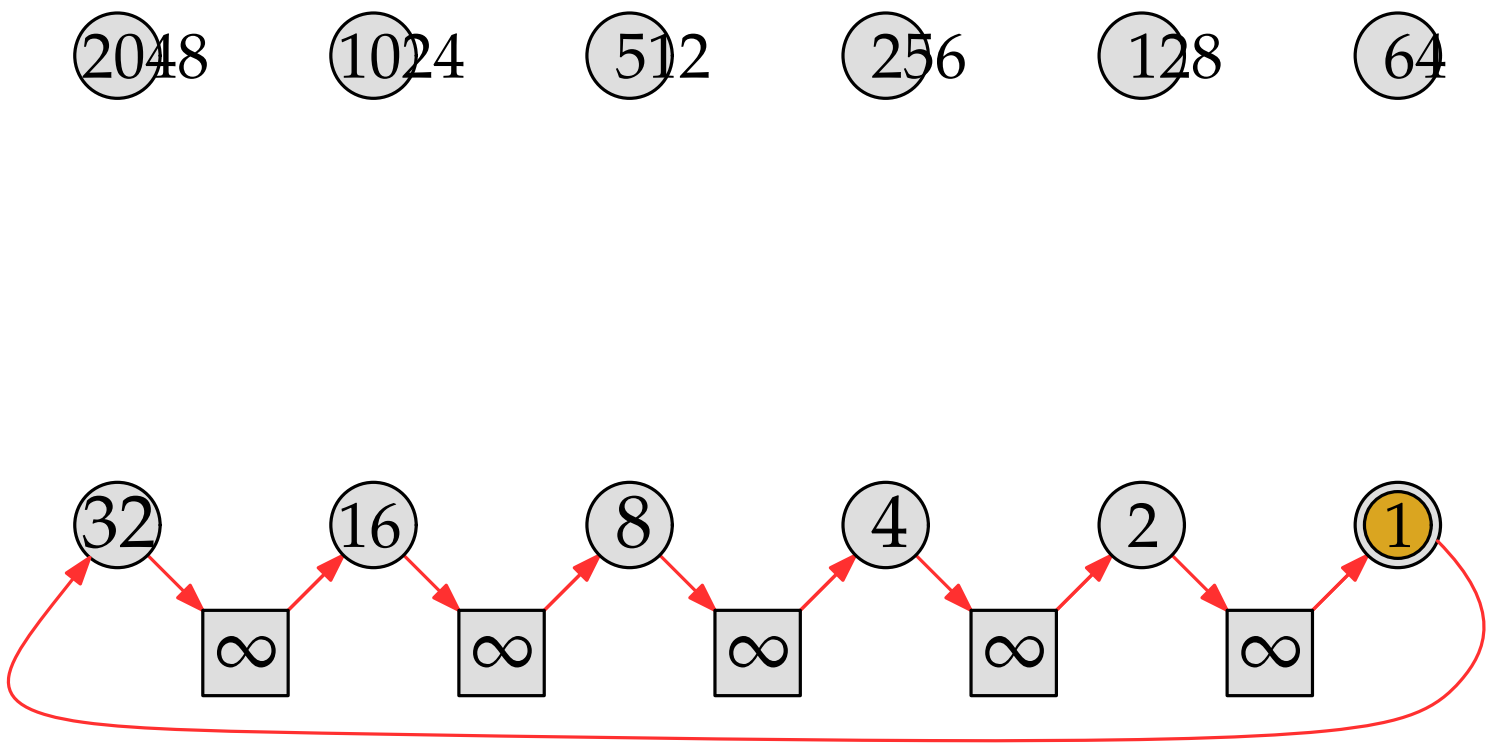
Quadratic Path for 2 Commuting Permutations



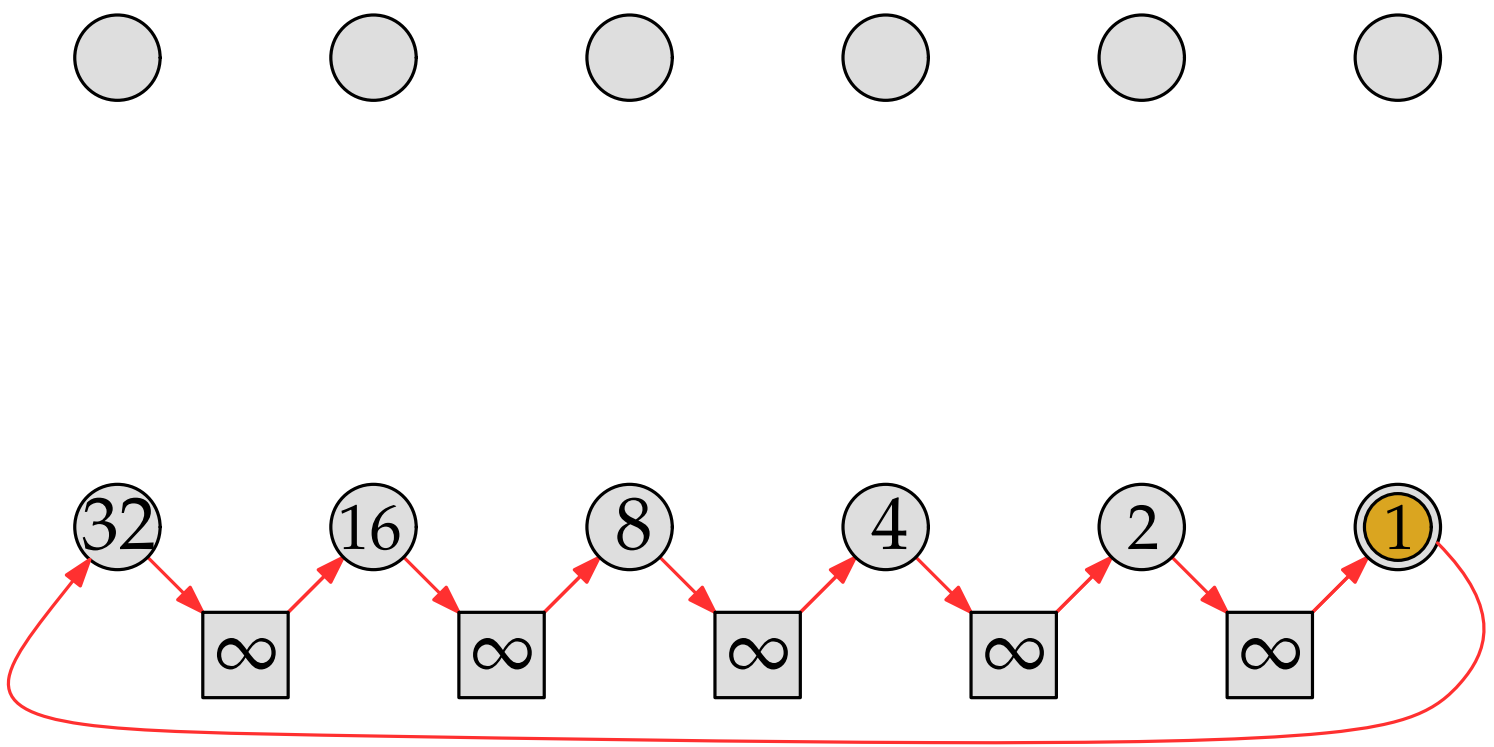
Quadratic Path for 2 Commuting Permutations



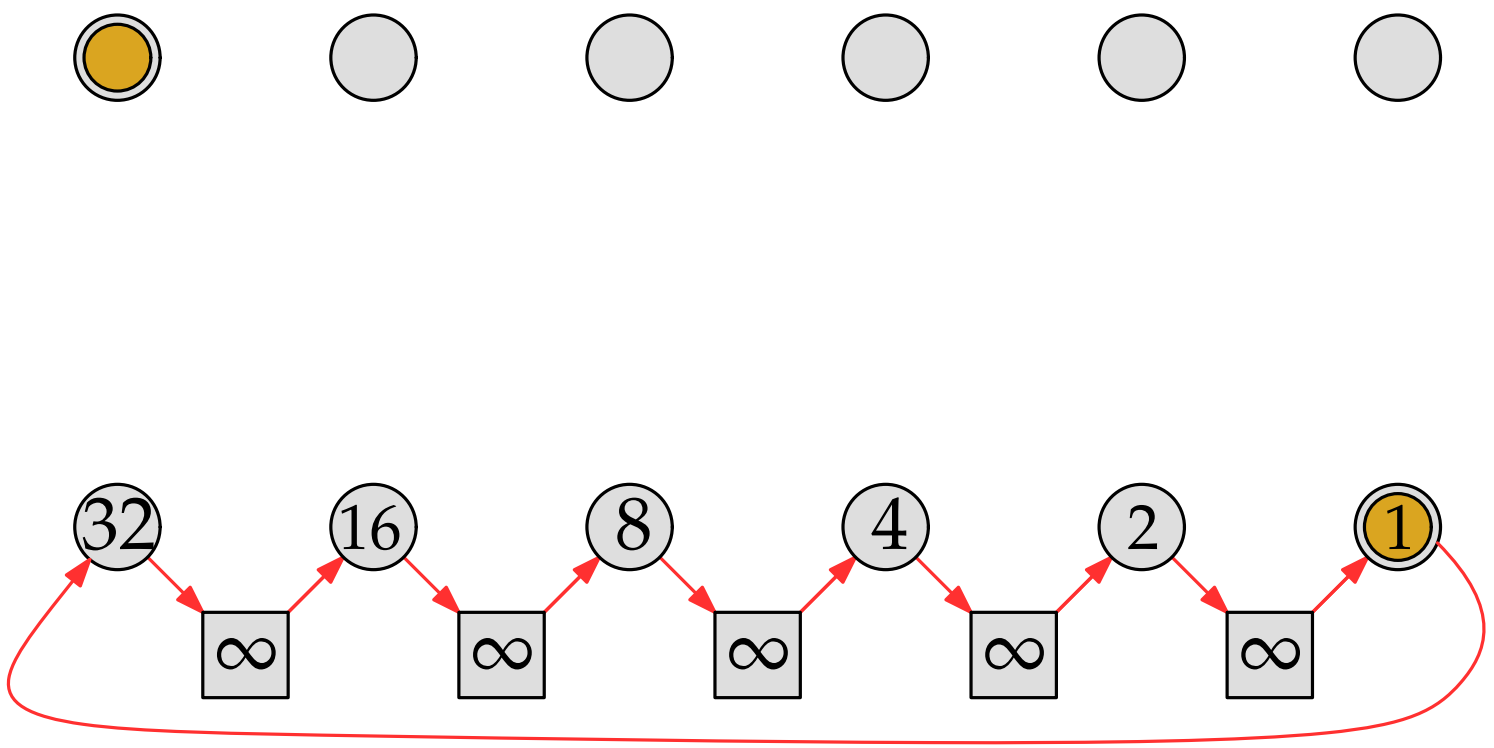
Quadratic Path for 2 Commuting Permutations



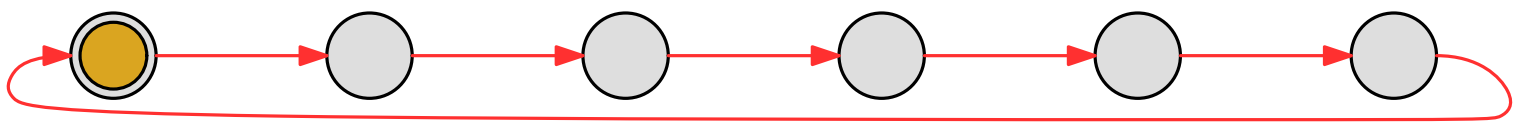
Quadratic Path for 2 Commuting Permutations

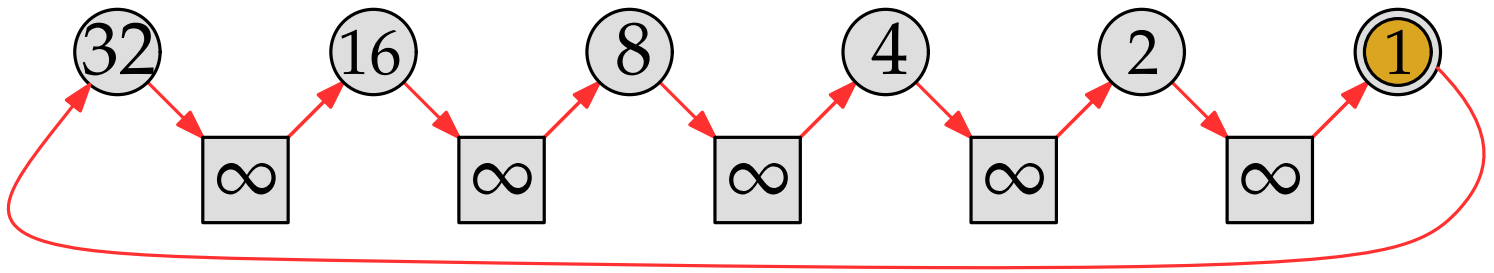


Quadratic Path for 2 Commuting Permutations

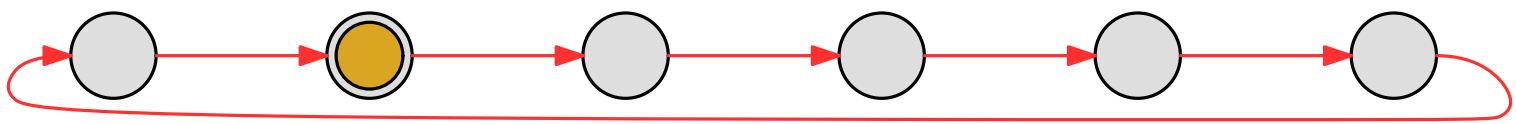


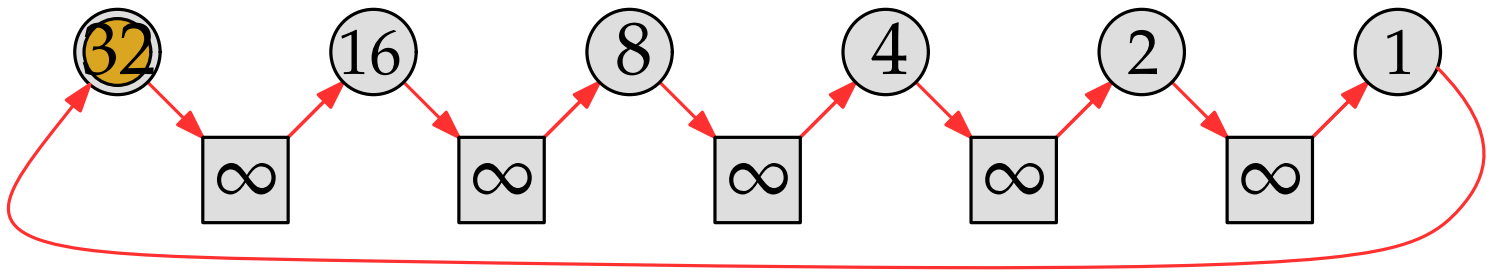
Quadratic Path for 2 Commuting Permutations



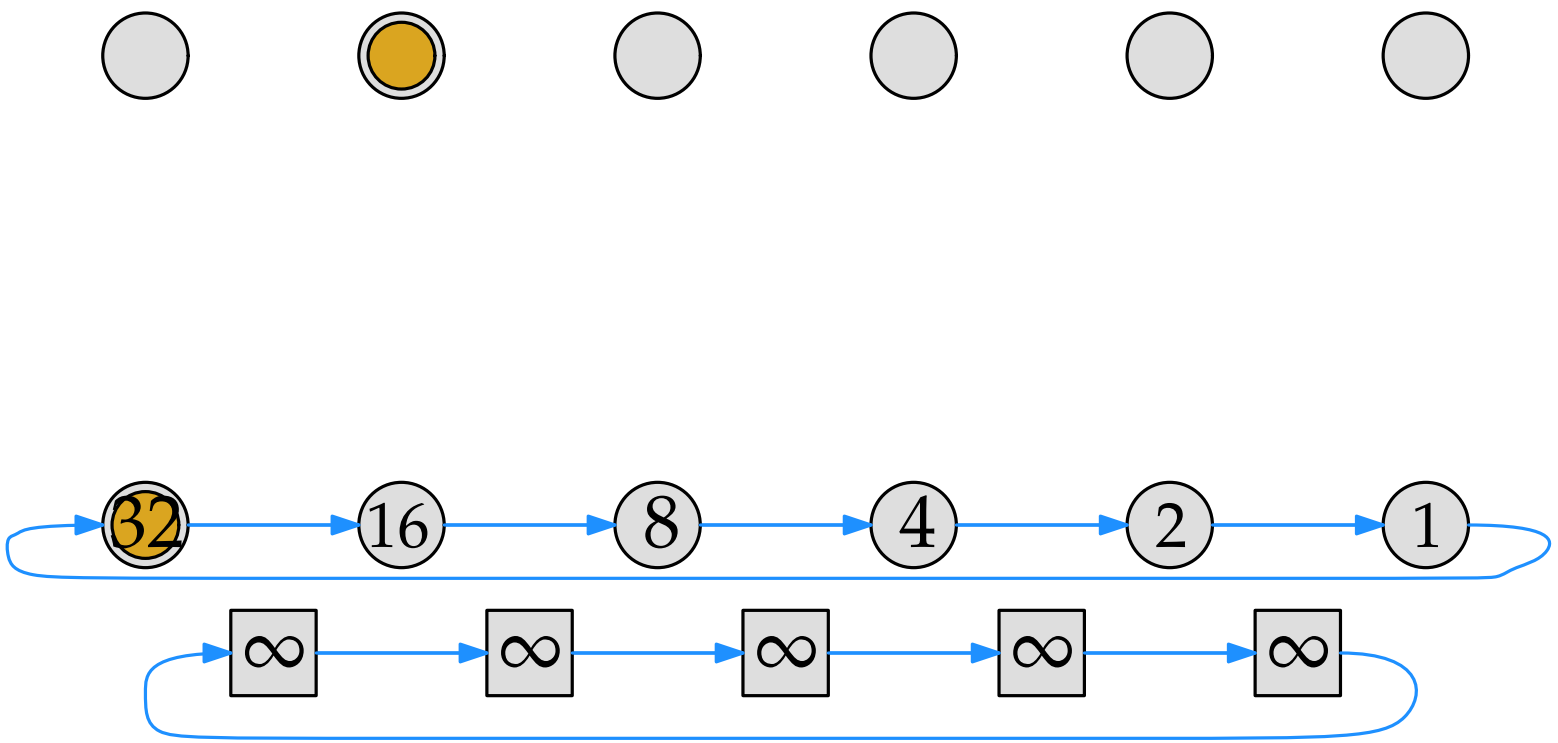


Quadratic Path for 2 Commuting Permutations

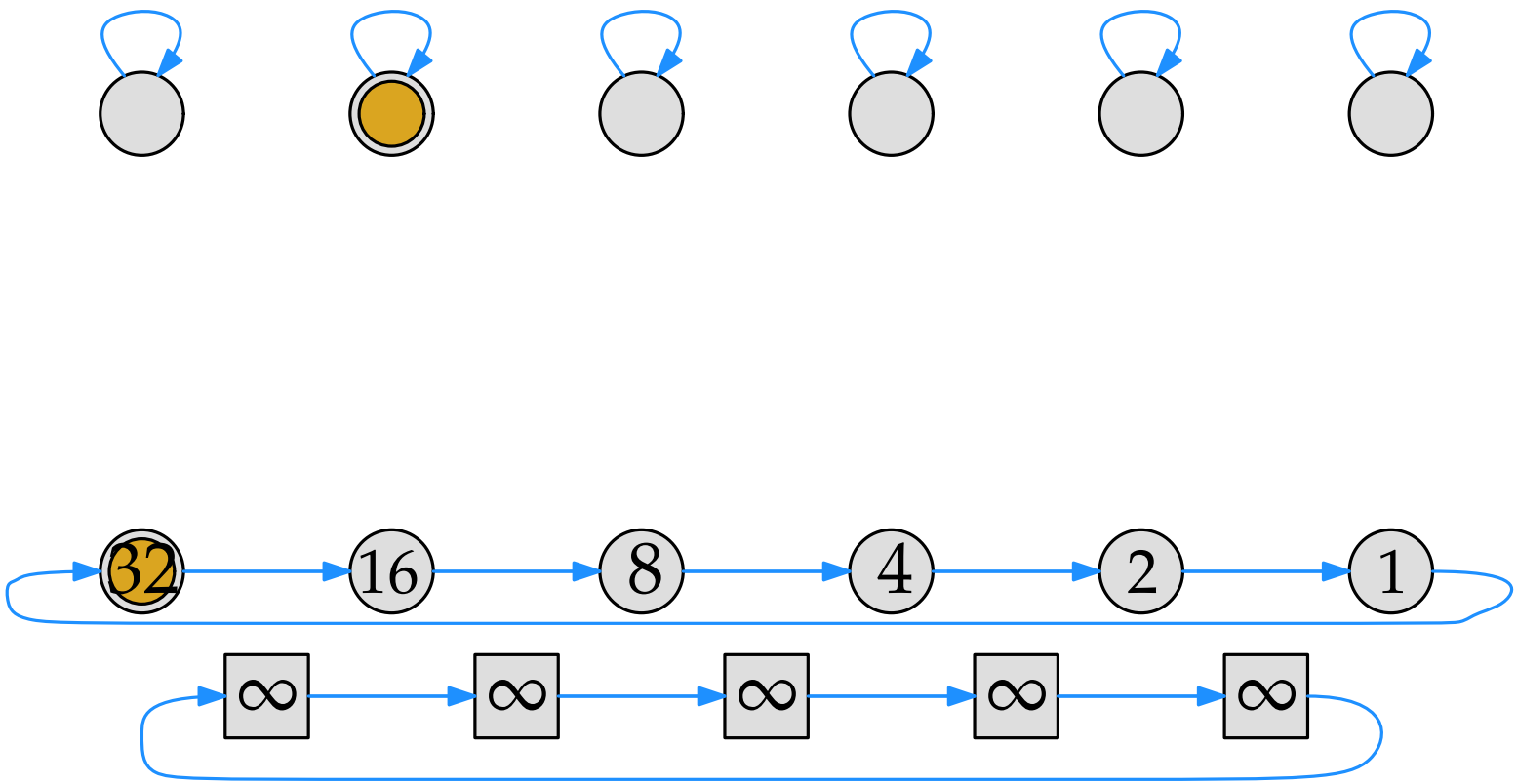




Quadratic Path for 2 Commuting Permutations

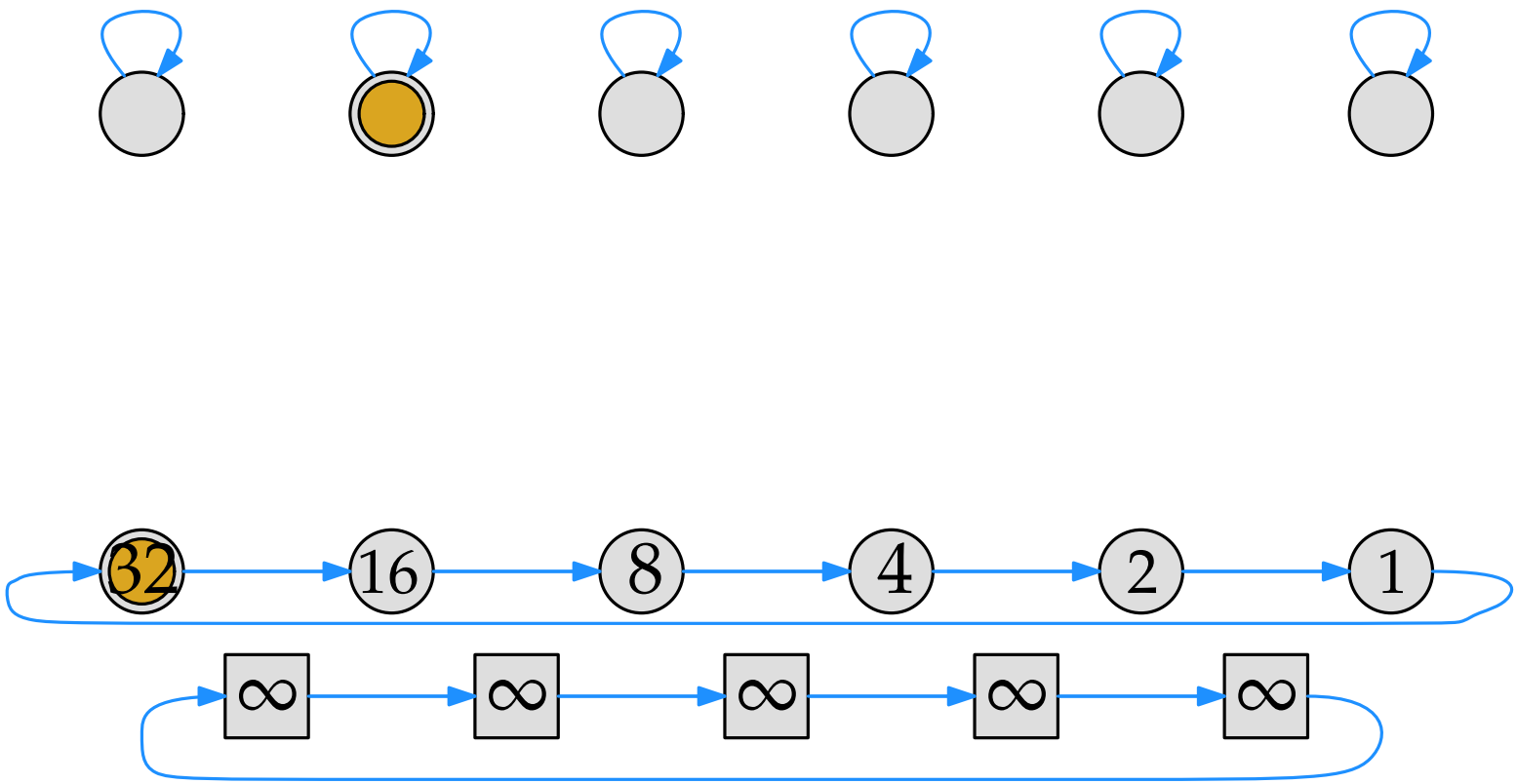


Quadratic Path for 2 Commuting Permutations



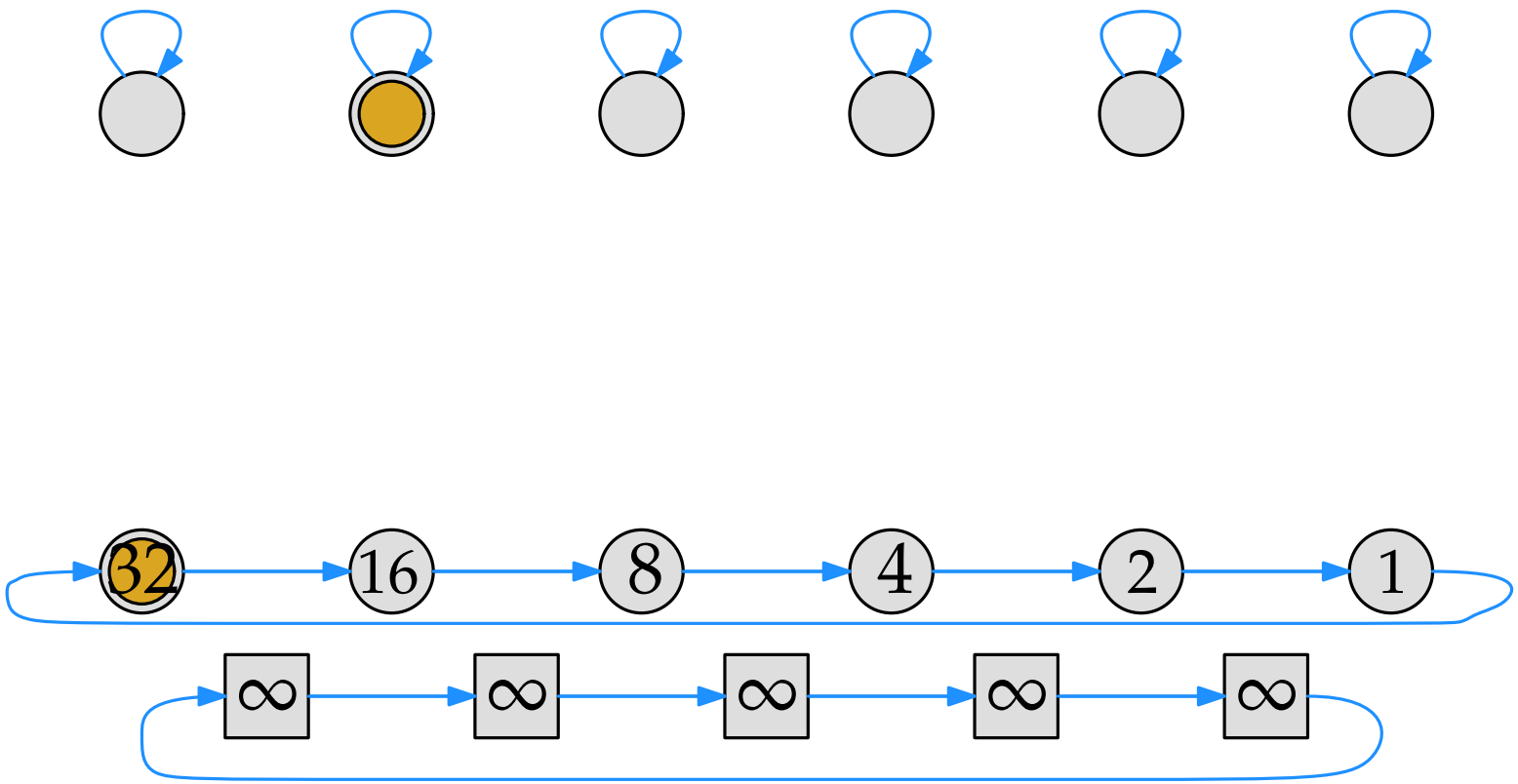
Quadratic Path for 2 Commuting Permutations

Lemma. The improving path under 2 commuting permutations on n positions can have length $\Omega(n^2)$.



Quadratic Path for 2 Commuting Permutations

Lemma. The improving path under 2 commuting permutations on n positions can have length $\Omega(n^2)$. In general: k commuting permutations can give an improving path of length $\Omega(n^k)$.



Two permutations π and π^2

Simplest Open Problem. What about having just two permutations, π and π^2?

Two permutations π and π^2

Simplest Open Problem. What about having just two permutations, π and π^2?

Fact. A local minimum can be found in polynomial time.

Two permutations π and π^2

Simplest Open Problem. What about having just two permutations, π and π^2?

Fact. A local minimum can be found in polynomial time.

Proof. We leave it as an exercise.

Two permutations π and π^2

Simplest Open Problem. What about having just two permutations, π and π^2?

Fact. A local minimum can be found in polynomial time.

Proof. We leave it as an exercise.

But can we actually show that the path has at most polynomial length?

Two permutations π and π^2

Simplest Open Problem. What about having just two permutations, π and π^2?

Two permutations π and π^2

Simplest Open Problem. What about having just two permutations, π and π^2?

Definition. A sequence $a_1, a_2, \ldots, a_t$ is called *2-jump decreasing* if $a_i > \min(a_{i+1}, a_{i+2})$.

Two permutations π and π^2

Simplest Open Problem. What about having just two permutations, π and π^2?

Definition. A sequence $a_1, a_2, \ldots, a_t$ is called *2-jump decreasing* if $a_i > \min(a_{i+1}, a_{i+2})$.

Conjecture. Suppose $f_1, \ldots, f_l : \mathbb{Z} \to \mathbb{Z}$ are periodic functions with f_i having period p_i. Set $f := f_1 + \cdots f_l$.
If $f(1), f(2), f(3), \ldots, f(b)$ is 2-jump decreasing
then b is polynomial in n.