F-sequences and Hamiltonian cycles in the hypercube

Ph. D. Candidate 🎓: **Israel Buitrón Dámaso**[1],
Advisors: **Guillermo Morales Luna**[1], **Feliú Sagols Troncoso**[2]

October 5th, 2018

[1]Departament of Computer Science and

[2]Departament of Matemathics

Current status of work

Initial work proposal

- The initial proposal was focused on the study of authentication protocols, namely zero-knowledge proofs (ZKP's), based on hard problems of graph theory.
- The analysis of the originally considered graph problem showed us the infeasibility for its practical application in cryptographic protocols.
- So, the study was refocused into the area of formal graph theory.

Current status of work i

- Each hypercube is a Hamiltonian graph and the number of Hamiltonian cycles has a doubly-exponential growth.
- We describe any Hamiltonian cycle by a sequence whose entries are the indexes of the basic directions traversed according to the cycle. These number sequences are called f-sequences (f-seqs).
- The f-seqs are classified into equivalence classes through elementary transformations (e.g. sequence rotation, sequence reflexion, and some others).

Current status of work ii

- We have introduced (other) several transformations, for each one, we have built a particular trajectory graph: the vertices are the representative f-seqs and the edges result form the transformation, a vertex and its transformed vertex form an edge.
- Several problems arise, e.g. which are the minimal transformation sets that produce connected trajectory graphs, for a given transformation, which is the number of connected components in the trajectory graph.
- We have designed several algorithms for application of transformations and to answer several decision problems.

Current status of work iii

- We have built a C library which implements those algorithms.

Hamiltonian cycles and f-sequences

Hypercube i

Definition (Hypercube graph)

Let be $Q_n = \{0,1\}^n$, the n-dimentional hypercube. It is a n-dimetional $\mathbb{F}_2$-vectorial space, spanned by the canonical basis $E = (e_j)_{j=1}^n$, where $e_j = (\delta_{i,j})_{i=1}^n$ and $\delta_{i,j}$ is Kronecker's delta. It is also a metric space with Hamming distance

$$d_H : (u, w) \mapsto card(\{i \mid u_i \neq w_i\}).$$

The *edges of* Q_n are pairs $(v, v + e_i)$, where $v \in V(Q_n)$, i.e, they have a Hamming distance equals to 1. We say that *it is an edge through* i*-th coordinate*.

Hypercube ii

Example

The edge (v_{12}, v_{14}) through first coordinate into Q_4. Their distances is $d_H(\underbrace{1100}_{12}, \underbrace{1110}_{14}) = 1$.

Since, all vertices in Q_n can be enumerated with a n-bit string, they will be enumerated with a subindex $i \in \mathbb{Z}_{2^n}$ which is the number represented by that string.

f-sequences i

Definition (f-sequence)

An n-order f-seq with length m is a finite sequence $f = (f_0, \ldots, f_{m-1})$ where $f_i \in \mathbb{Z}_n$.

If $m < 2^n$, then in each proper sequence $(f_k, \ldots, f_l)$, with $0 \leq k < l \leq 2^n - 1$, there is at least one item which appears and odd-number times.

If $m = 2^n$, then all items appear an even-number times.

The maximum length of an n-order f-seq is 2^n. We say that it is *complete* f-seq.

f-sequences ii

Example

A 2-order f-seq with length 2^2 is

$$(0, 1, 0, 1).$$

A 3-order f-seq with length 2^3 is

$$(0, 1, 0, 2, 0, 1, 0, 2).$$

A 4-order f-seq with length 2^4 is

$$(0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, 3).$$

A 5-order f-seq with length 2^4 is

$$(0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, 4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, 4).$$

Type of a sequence i

Definition (Type of a sequence)

Let be $(f_0, \ldots, f_{2^n-1})$ a complete n-order f-seq.

The *type* of a sequence $(f_i, \ldots, f_j)$, denoted by $type(f_i, \ldots, f_j)$, where $0 \leq i < j < 2^n$, is the set of items in that sequence such that appear an odd-number times.

Type of a sequence ii

Example

Given the f-seq in Example 4

$$\begin{aligned} type(0,1,0,2) &= \{1,2\} \\ type(0,1,0,2,0) &= \{0,1,2\} \\ type(0,1,0,2,0,1,0,2) &= \emptyset \end{aligned}$$

F-equivalence i

Definition (Actions of f-equivalence)

Let f be a complete n-order f-seq.

1. If $\pi : \mathbb{Z}_n \to \mathbb{Z}_n$ is a permutation, from S_n group, then $\pi(f) = (\pi(f_0), \ldots, \pi(f_{2^n-1}))$ is a complete f-seq.
2. The *reversed sequence* $f^r = (f_{2^n-1}, \ldots, f_0)$, is an f-seq.
3. Any *circular rotation* of a f, $(f_i, \ldots, f_{2^n-1}, f_0, \ldots, f_{i-1})$, where $i \in \mathbb{Z}_{2^n}$, is a complete f-seq.

Those operations define an action of the group $S_n \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_{2^n}$ into the set of complete f-seqs. We will say that two f-seqs are *f-equivalent*, if they belong to the same (action) orbit.

F-equivalence ii

Example

The f-seq from Example 4 is equivalent to

$$(2, 0, 1, 0, 2, 0, 1, 0)$$

$$(0, 2, 0, 1, 0, 2, 0, 1)$$

$$(1, 2, 1, 0, 1, 2, 1, 0).$$

Representant of class

Definition (Minimum f-sequence)

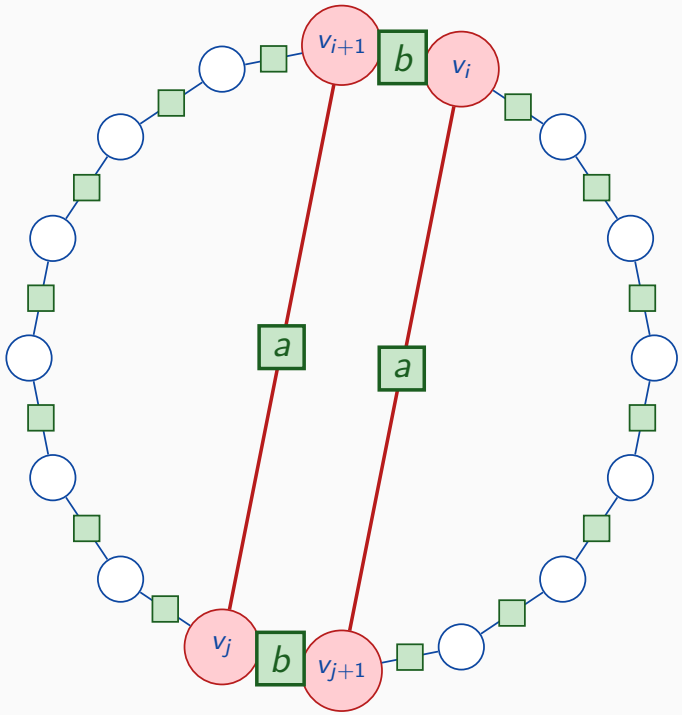
An f-seq is *minimal*, if it is the minimum item in its f-equivalence class, considering lexicographic order.

Squares i

Definition (Straight-square)

Let f be an n-order f-seq and $i, j \in \mathbb{Z}_{2^n}$ two indices, with $i < j$.

If $type(v_i, \ldots, v_{i+1}) = type(v_j, \ldots, v_{j+1}) = \{b\}$ and $type(v_{i+1}, \ldots, v_j) = type(v_{j+1}, \ldots, v_i) = \{a\}$, then these four vertices form a 4-cycle. We call it a *straight-square*.

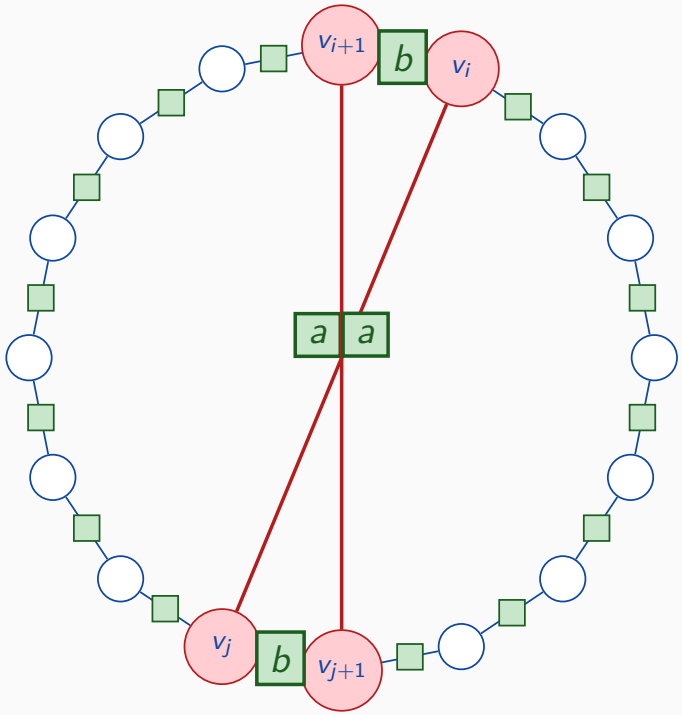


Squares ii

Definition (Twisted-square)

Let f be an n-order f-seq and $i, j \in \mathbb{Z}_{2^n}$ two indices, with $i < j$.

If $type(v_i, \ldots, v_{i+1}) = type(v_j, \ldots, v_{j+1}) = \{b\}$ and $type(v_{i+1}, \ldots, v_{j+1}) = type(v_i, \ldots, v_j) = \{a\}$, then these four vertices form a 4-cycle. We call it a *twisted-square*.



Operators of f-sequences

Clover transformation

Definition (Clover)
Let f be an n-order f-seq which can be expressed as $AaBbCaDb$, where A, B, C, D are non-empty proper sequences of f and a, b are proper sequences of length 1. All of them are pairwise disjoint.

If $type(AbD) = type(BbC) = a'$ and $type(AaB) = type(CaD) = b'$, where a', b' are proper sequences of length 1, then $Aa'Db'Ca'Bb'$ is a complete f-seq non f-equivalent to f. We call this a *clover* transformation.

Twisted clover transformation i

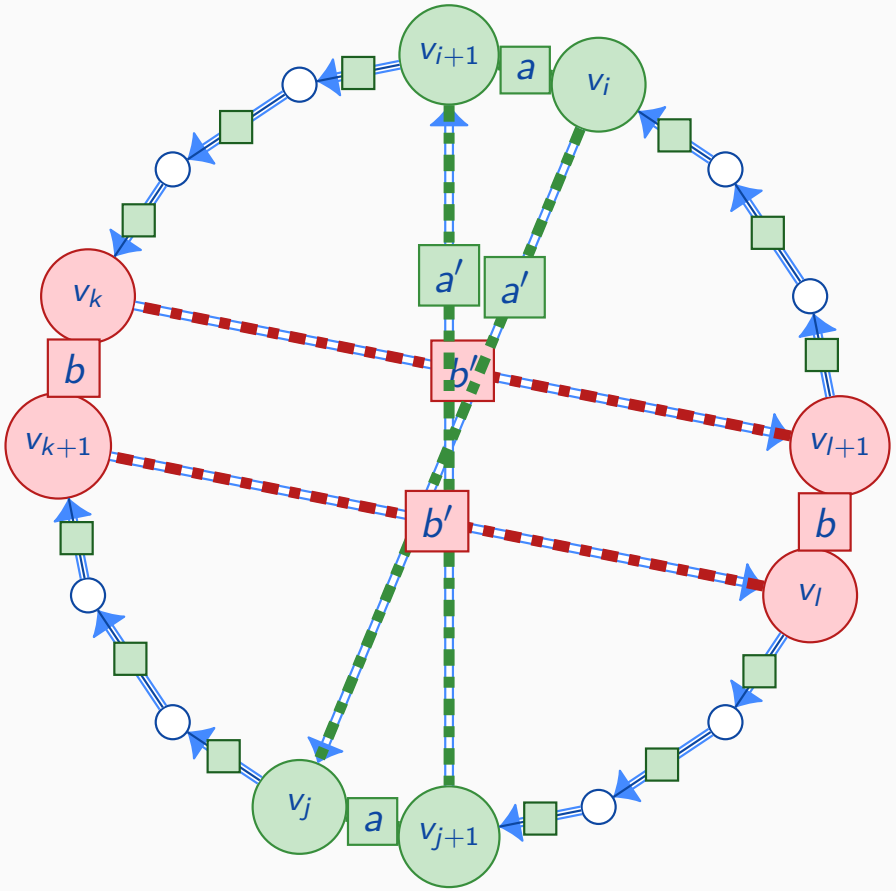
Definition (Twisted clover)
Let f be a n-order f-seq which can be expressed as $AaBbCaDb$, where A, B, C, D are non-empty proper sequences of f and a, b are proper sequences of length 1. All of them are pairwise disjoint.

If $type(AbD) = type(BbC) = a'$ and $type(AaB) = type(CaD) = b'$, where a', b' are proper sequences of length 1, then

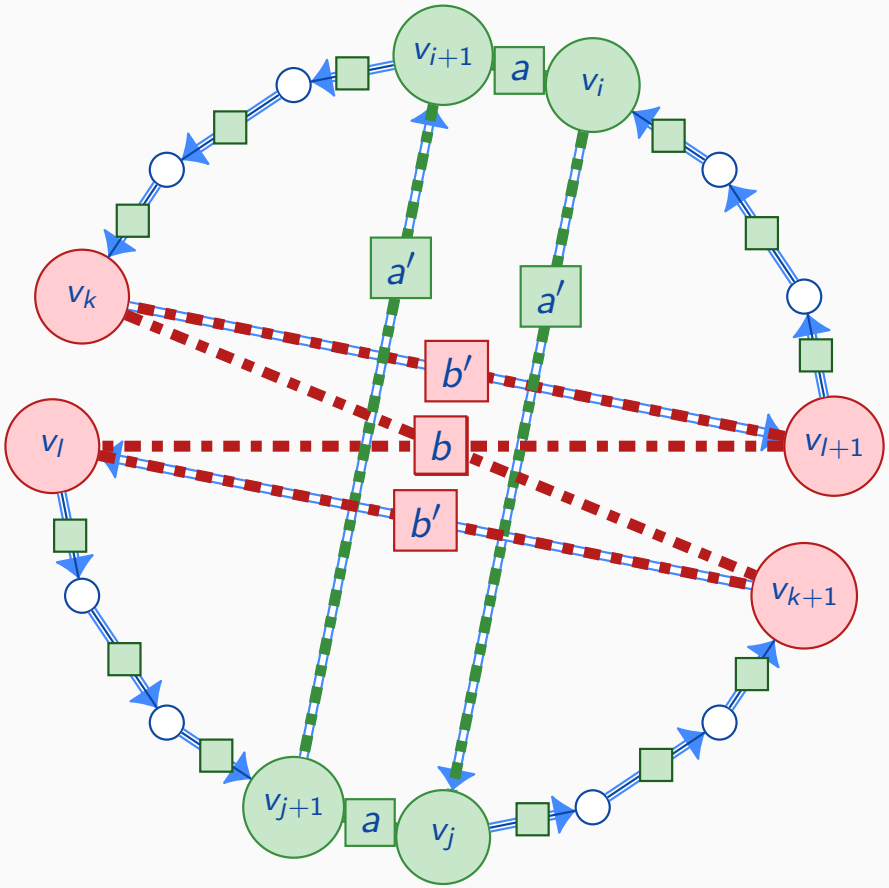
$$Aa'D^r b'C^r b'Bb'$$

is a complete f-seq non f-equivalent to f. We call this a *twisted-clover* transformation.

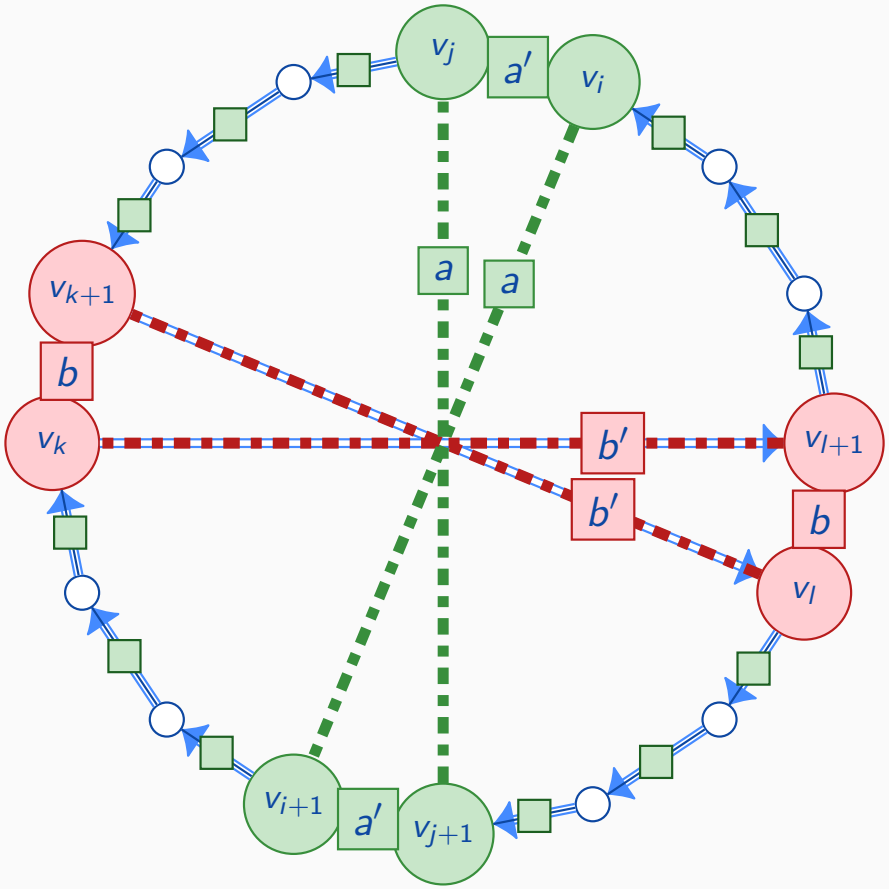
Twisted clover transformation ii



Twisted clover transformation iii



Twisted clover transformation iv



Mushroom transformation i

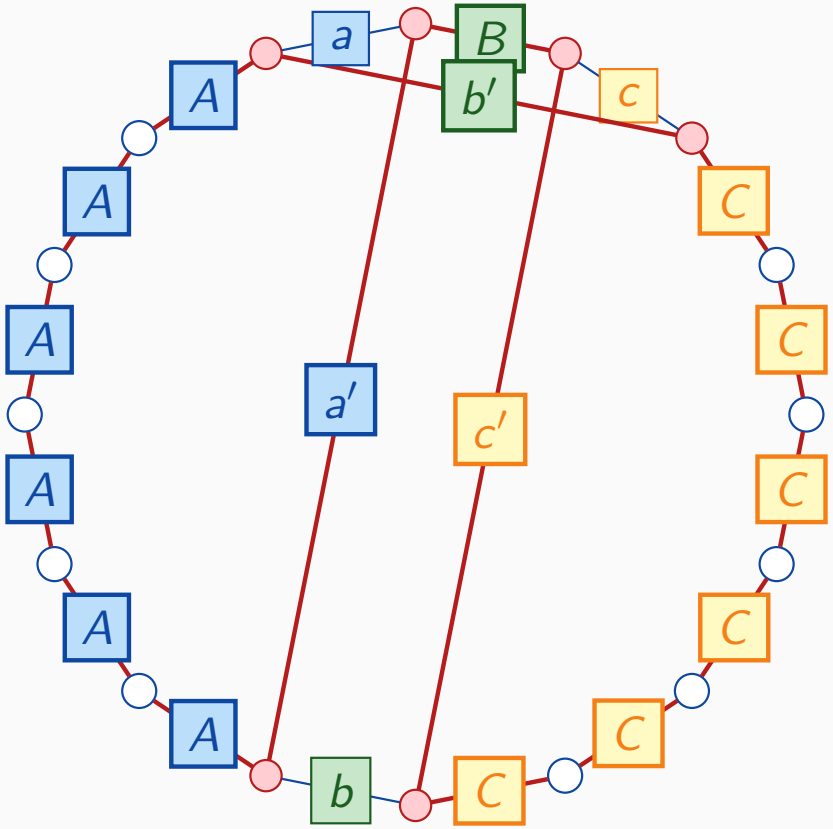
Definition (Mushroom)
Let f be an n-order f-seq which can be expressed as $AaBbCc$, where A, B, C are non-empty proper sequences of f and a, b, c are proper sequences of length 1. All of them are pairwise disjoint.

If $type(Ab) = a'$, $type(aBb) = b'$ y $type(bC) = c'$, where a', b', c' are proper sequences of length 1, then

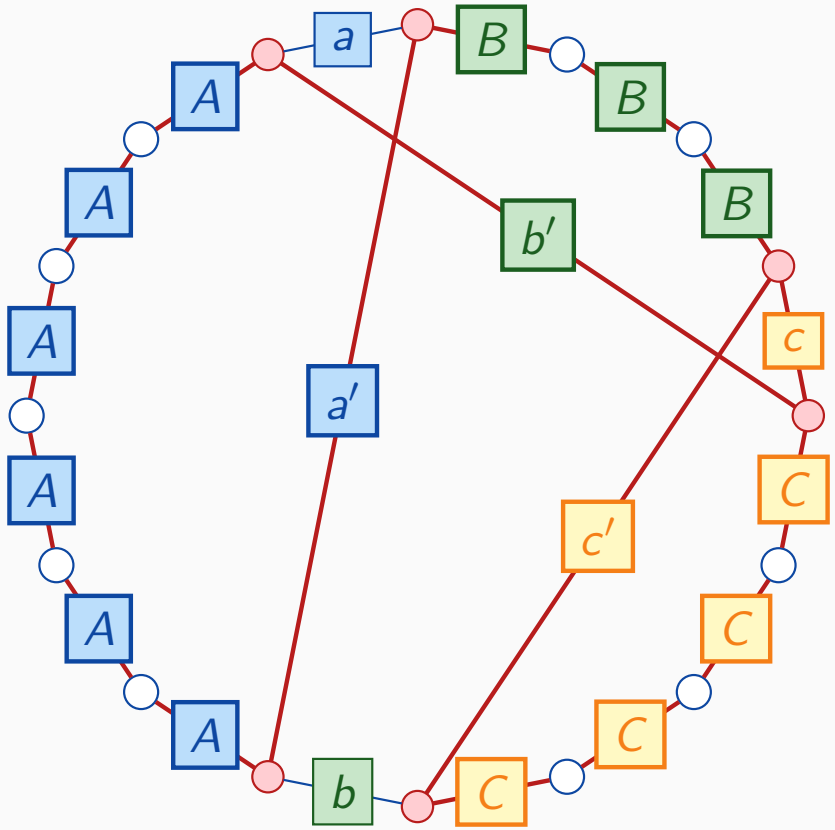
$$Ab'Cc'B^r a'$$

is a complete f-seq non f-equivalent to f. We call this a *mushroom* transformation.

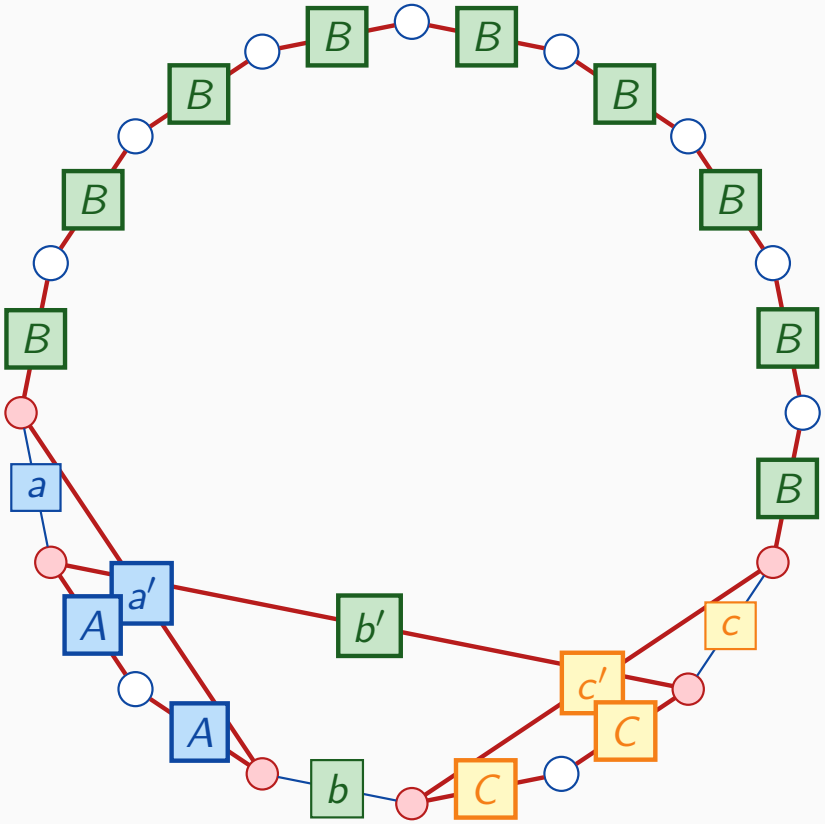
Mushroom transformation ii



Mushroom transformation iii



Mushroom transformation iv



Algorithms and implementations

Algorithms and implementations

Algorithms

Validation of f-sequences

Algorithm 1: `fsval` – Validation of f-sequences.

Input: s, a sequence of length 2^n with items in $\mathbb{Z}_n$.
Output: 1 if s is a complete n-order f-seq, otherwise 0.

```
1  begin
2      for i ← 0 to 2^n − 1 do                    // Clear 2^n memory
3          h[i] ← 0
4      v ← 0
5      for i ← 0 to 2^n − 1 do
6          v ← v ⊕ (1 << s[i])
7          if h[v] ≠ 0 then                       // Unexpected loop path
8              return 0                           // s is not an f-seq
9          h[v] ← 1                               // Mark vertex as visited
10     return 1                                   // s is an f-seq
```

Pseudorandom f-sequence generation

Algorithm 2: `dac_prfsg` – Pseudorandom f-sequences generation.

Input: $n \in \mathbb{Z}^+$ such that $n > 2$.
Output: f, a complete n-order f-seq randomly generated.

```
1  begin
2      if n = 2 then                                  // When f-seqs order 2
3          b ←$ {0,1}
4          f ← [(0,1,0,1),(1,0,1,0)]
5          return f[b]                  // Pick randomly f-seq of order 2
6      else                                        // When f-seqs of order > 2
7          f ← []                                              // f-seq memory
8          f[0] ← dac_prfsg(n − 1)                  // High-half of f-seq
9          f[1] ← dac_prfsg(n − 1)                   // Low-half of f-seq
10         if f[0][−1] = f[1][−1] then
11             f[0][−1] ← (f[1][−1] ← n)            // Connect both halves
12         else
13             while f[0][−1] = f[1][−1] do
14                 b ←$ {0,1}                              // Pick randomly...
15                 B ← f[b]                               // High or low half
16                 b ←$ {0,1}                              // Pick randomly...
17                 if b = 0 then B << 1                  // left-rotation, or
18                 else B >> 1                             // right-rotation
19             f[0][−1] ← (f[1][−1] ← n)                // Connect both halves
20         return f                                // Return random f-seq built
```

Squares search

Algorithm 3: `sqrs_search` – Square search into f-sequences.

Input: f, a complete n-order f-seq.
Output: $Q = \{(i, j, k) : i, j \in \mathbb{Z}_{2^n}, k \in \{0, 1\}$, a set of squares in f, where i y j are indices into f and k is the square class.

```
1  begin
2      Q ← ∅
3      foreach i = 0 to 2^n − 2 do
4          foreach j = i + 1 to 2^n − 1 do
5              if f[i] = f[j] then
6                  if type(f, i, j − 1) = 1 then
7                      Q ← Q ∪ {(i, j, 1)}
8                  else if type(f, i, j) = 1 then
9                      Q ← Q ∪ {(i, j, 0)}
10     return Q                                  // Set of squares found
```

Twisted-clover application

Algorithm 4: `twsq_app` – Twisted-clover application.

Input: f, a complete n-order f-seq, $t = (i, j)$ coordinates of a twisted-clover in f.

Output: f', a complete n-order f-seq such that is not f-equivalent to f.

```
1  begin
2      (m, a') ← parity(f, t_i, t_{j-1})
3      foreach i = 0 to 2^n − 2 do
4          if i + t_i + 1 = t_j then        // Sequence a' between A and B^r
5              f'[i] = a'
6              continue
7          if i + 1 = 2^n then              // Sequence a' between B^r and A
8              f'[i] = a'
9              continue
10         if i < (t_j − 1 − t_i) then                    // Sequence A
11             f'[i] = f[(i + t_i + 1) mod 2^n]
12             continue
13         if t_j < (i + t_i + 1) then                    // Sequence B^r
14             f'[i] = f[(t_j − i − 1) mod 2^n]
15             continue
16     return f'
```

Algorithms and implementations

Implementation

Fseq4c – Our C Library

- Source code is public.
  It is stored and managed by a **git** repository:
  `https://gitlab.com/israelbuitron/fseq4c`
- Source code is documented.

Algorithms and implementations

Demo

Pseudorandom f-sequence generation

Example (Watch a video)
Execution of *pseudorandom f-seq generation* from order 3 to 23, at:

`https://www.youtube.com/watch?v=KlljXlUBv1c`

Squares search

Example (Execute)
Go to terminal and run:

```
>_ ./lab_bulk_count_squares -n 4 -i fs-n4.csv
```

Mushroom instances search

Example (Execute)
Go to terminal and run:

```
>_ ./lab_bulk_search_mushrooms -n 4 -i fs-n4.csv
```

Twisted-squares instances search

Example (Execute)
Go to terminal and run:

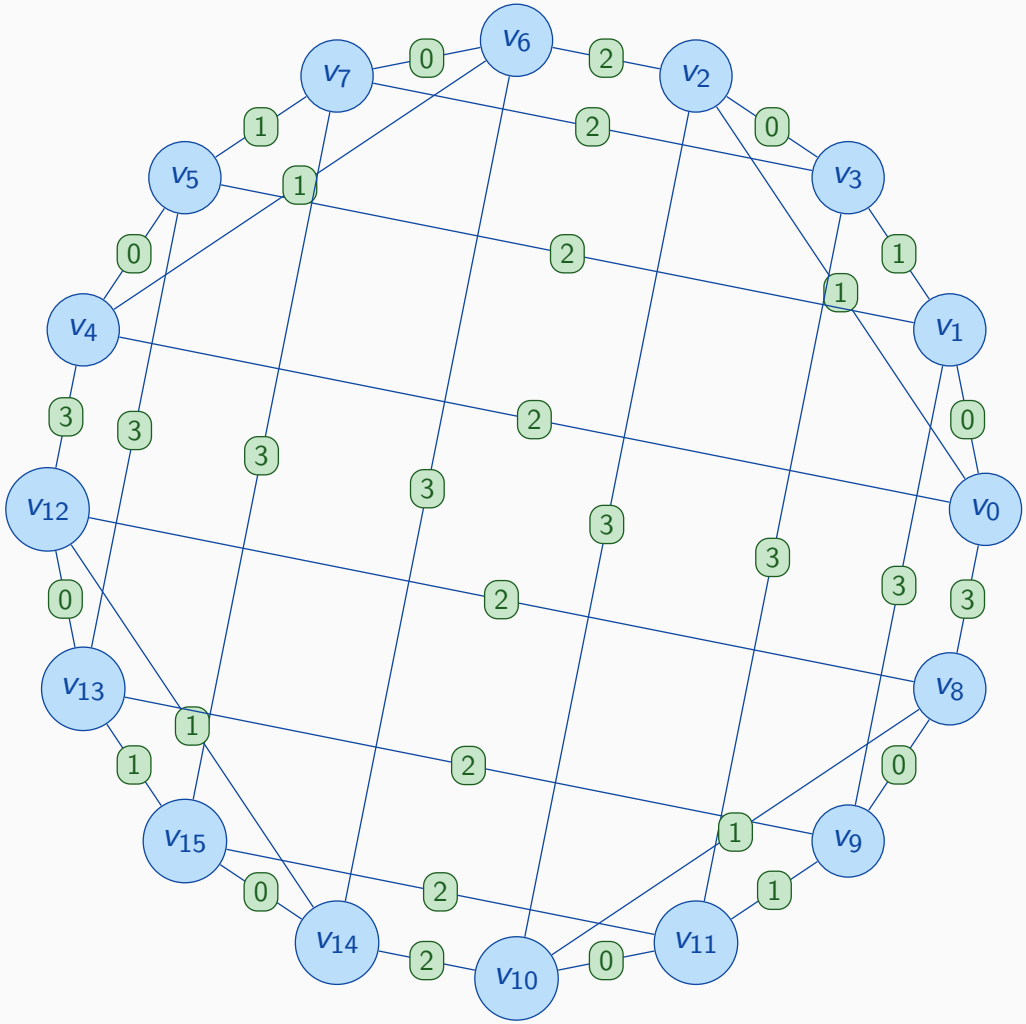
```
>_ ./lab_bulk_apply_twisted_squares -n 4 -i twsq-n4.csv
```

Thank you!

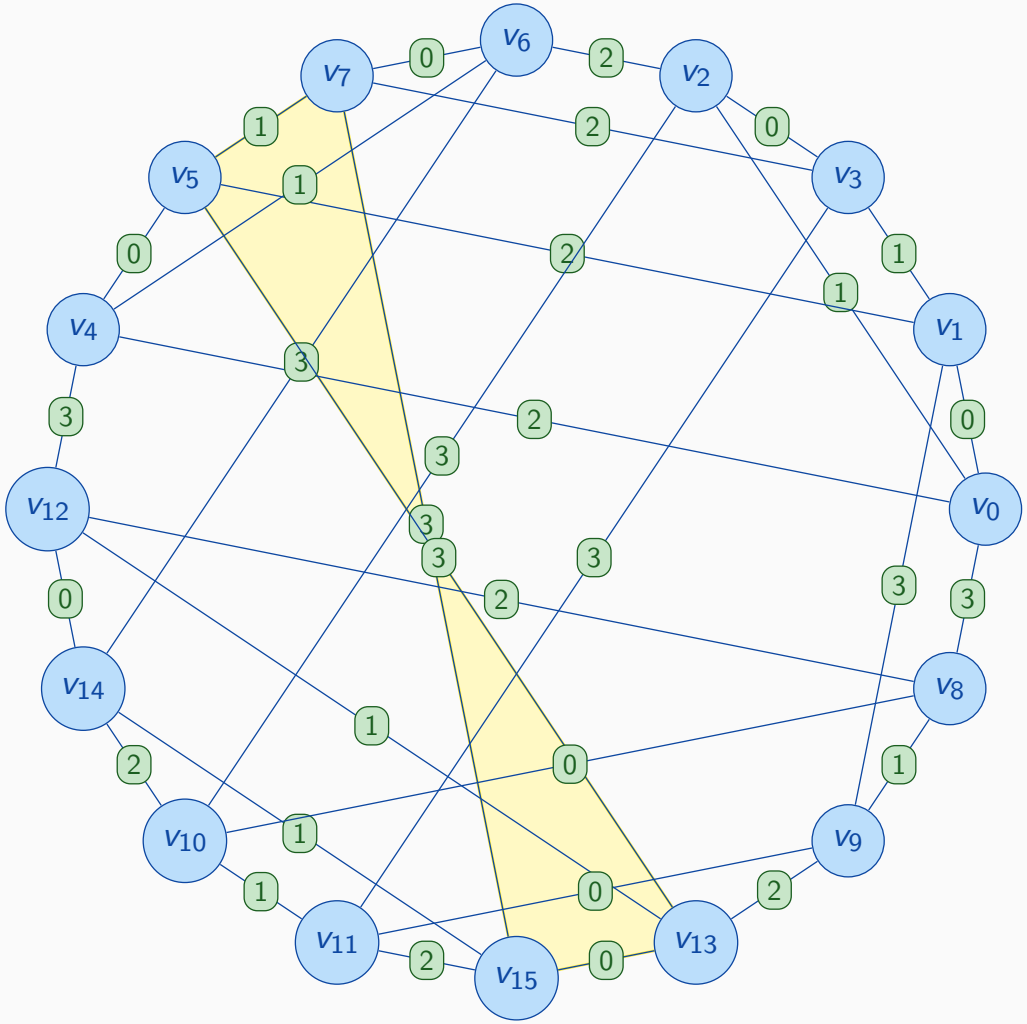
:)

F-sequences in Q_4

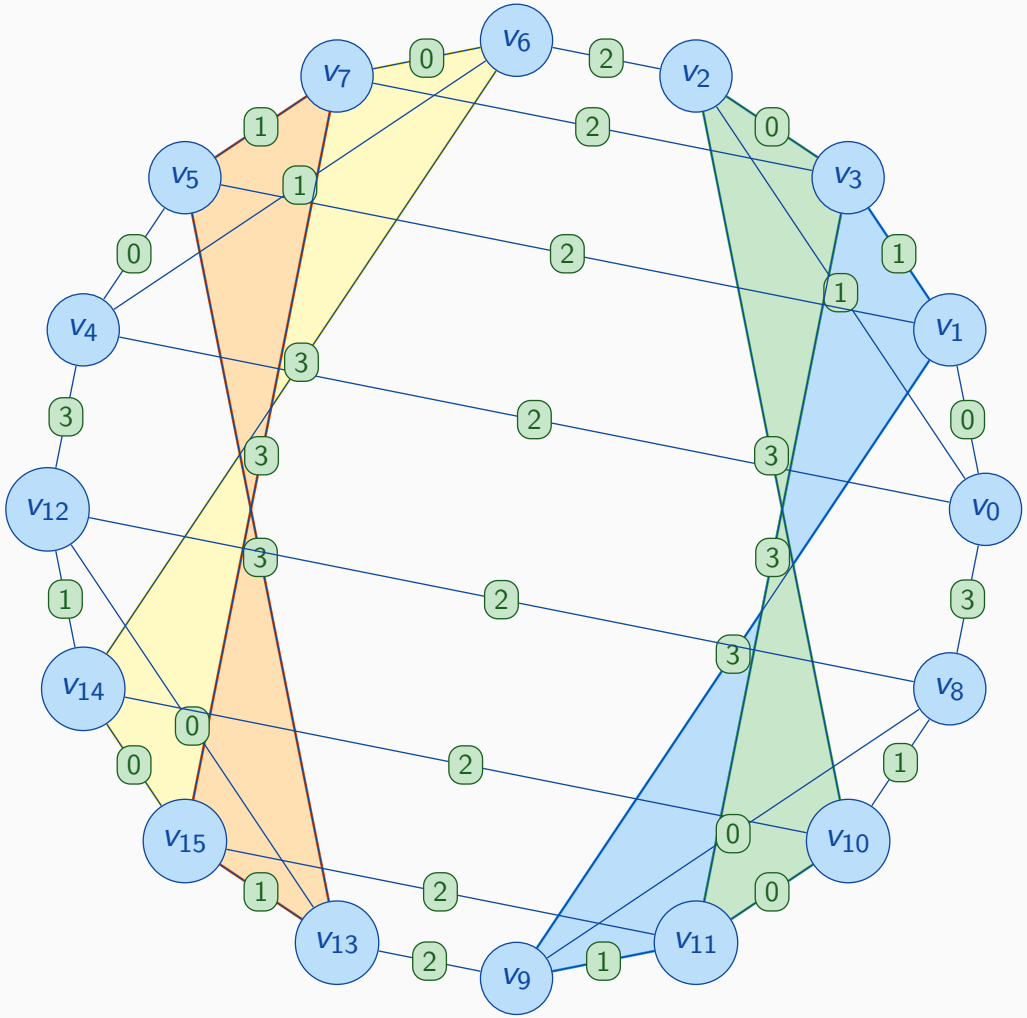
F-sequences in Q_4 i



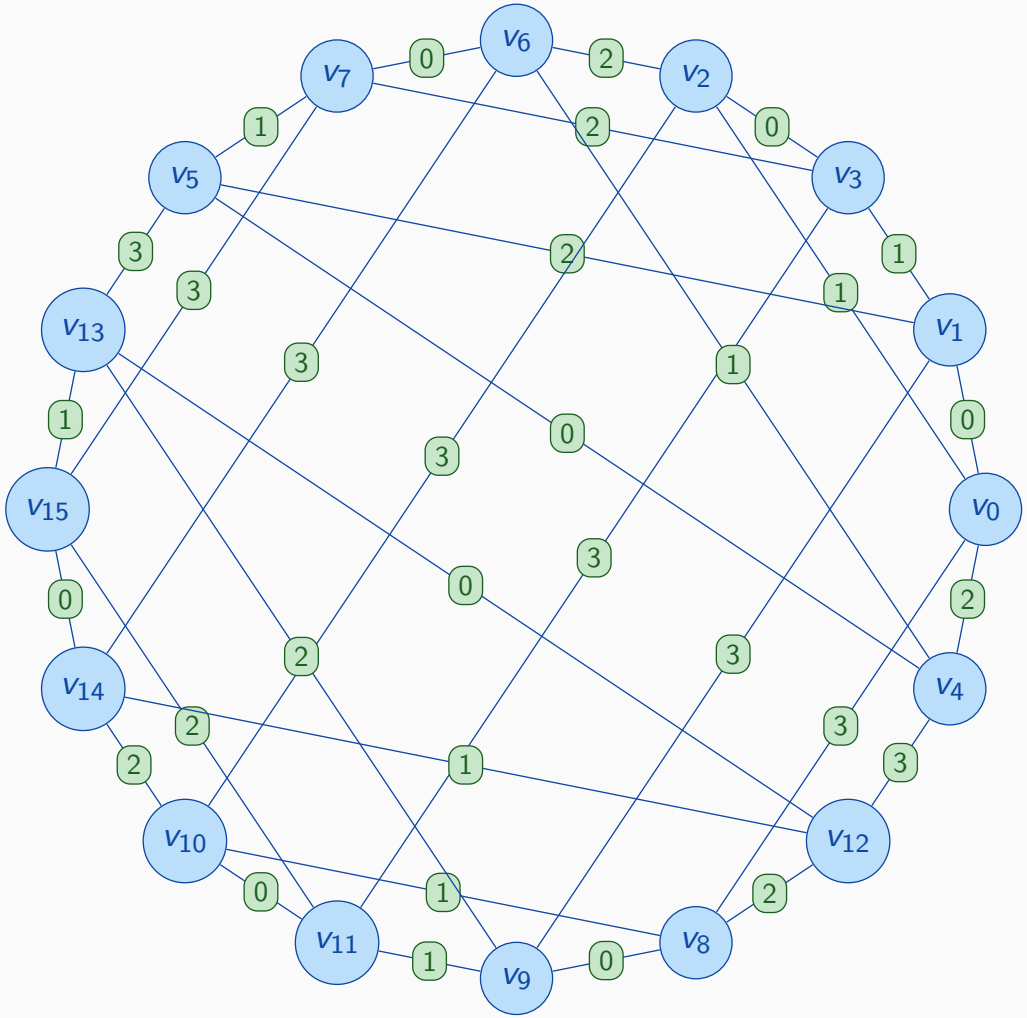
F-sequences in Q_4 ii



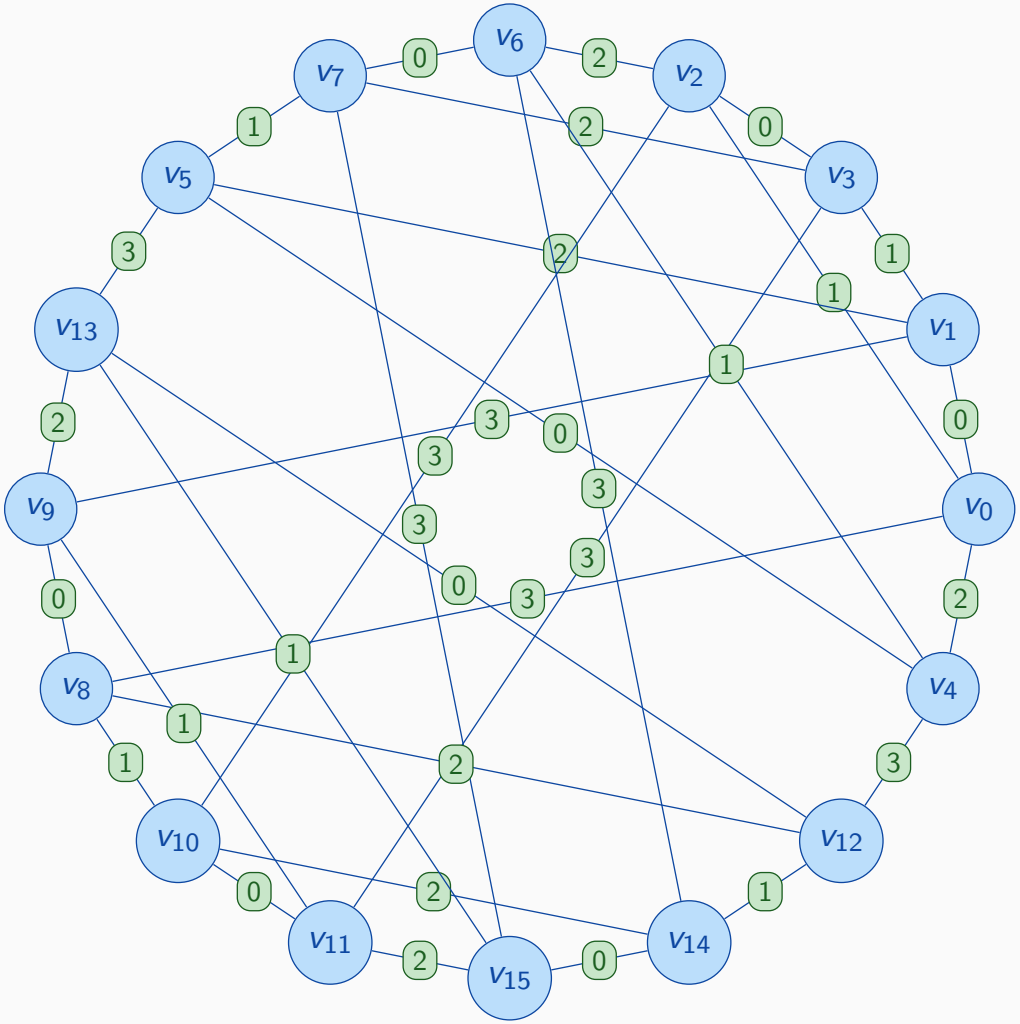
F-sequences in Q_4 iii



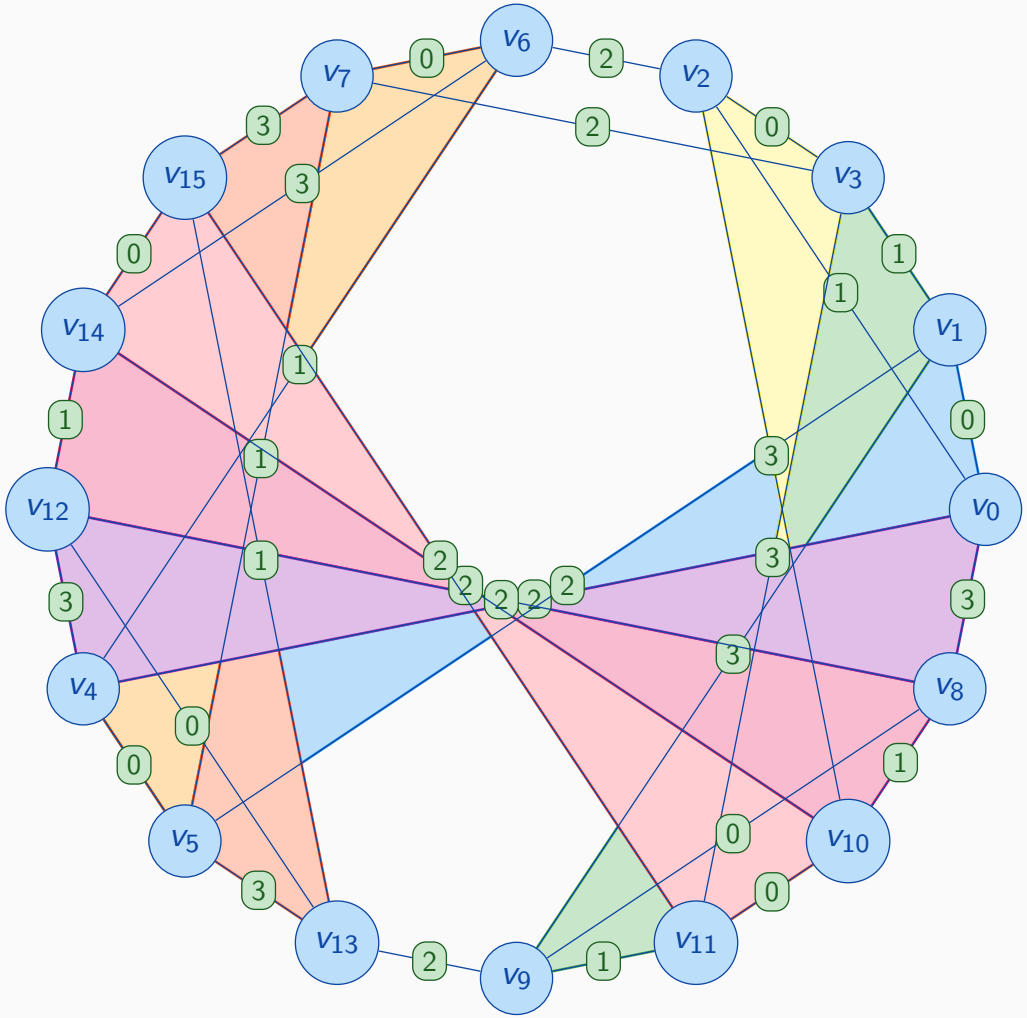
F-sequences in Q_4 iv



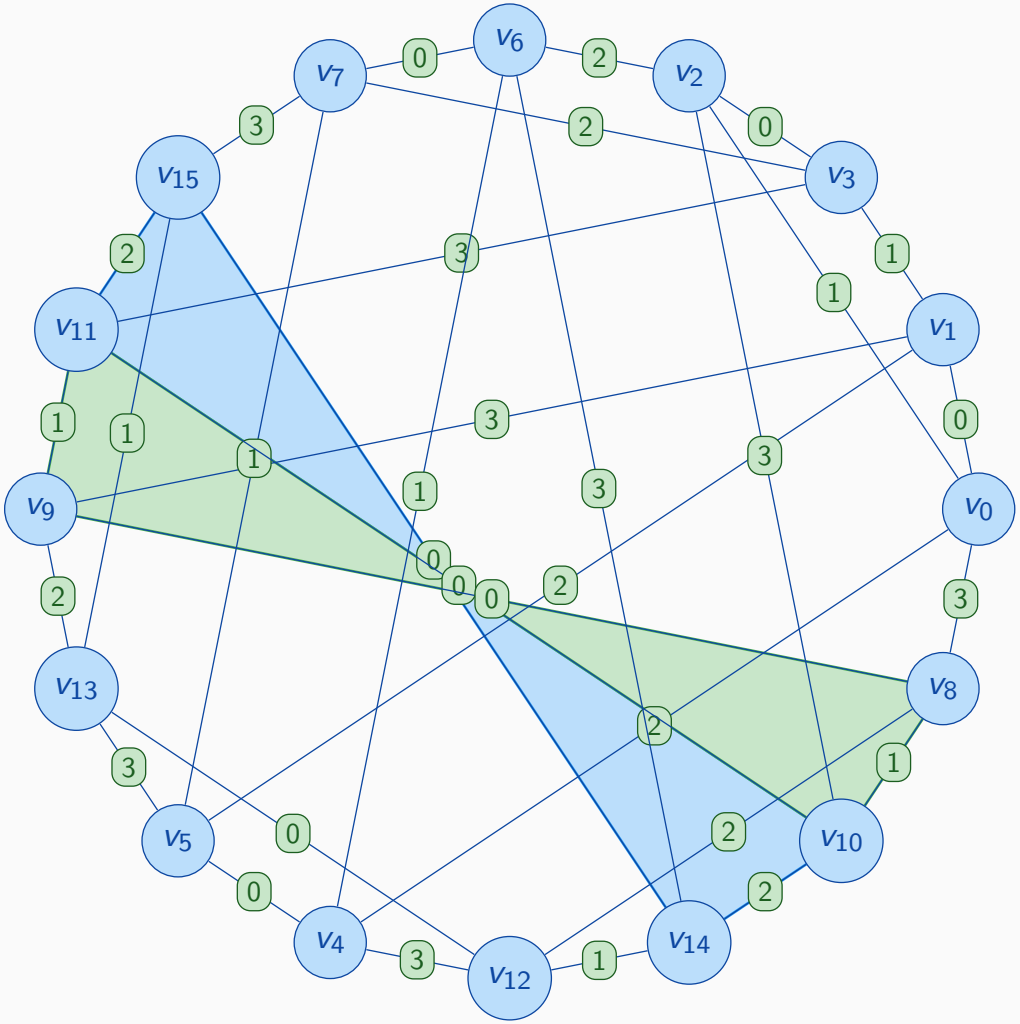
F-sequences in Q_4 v



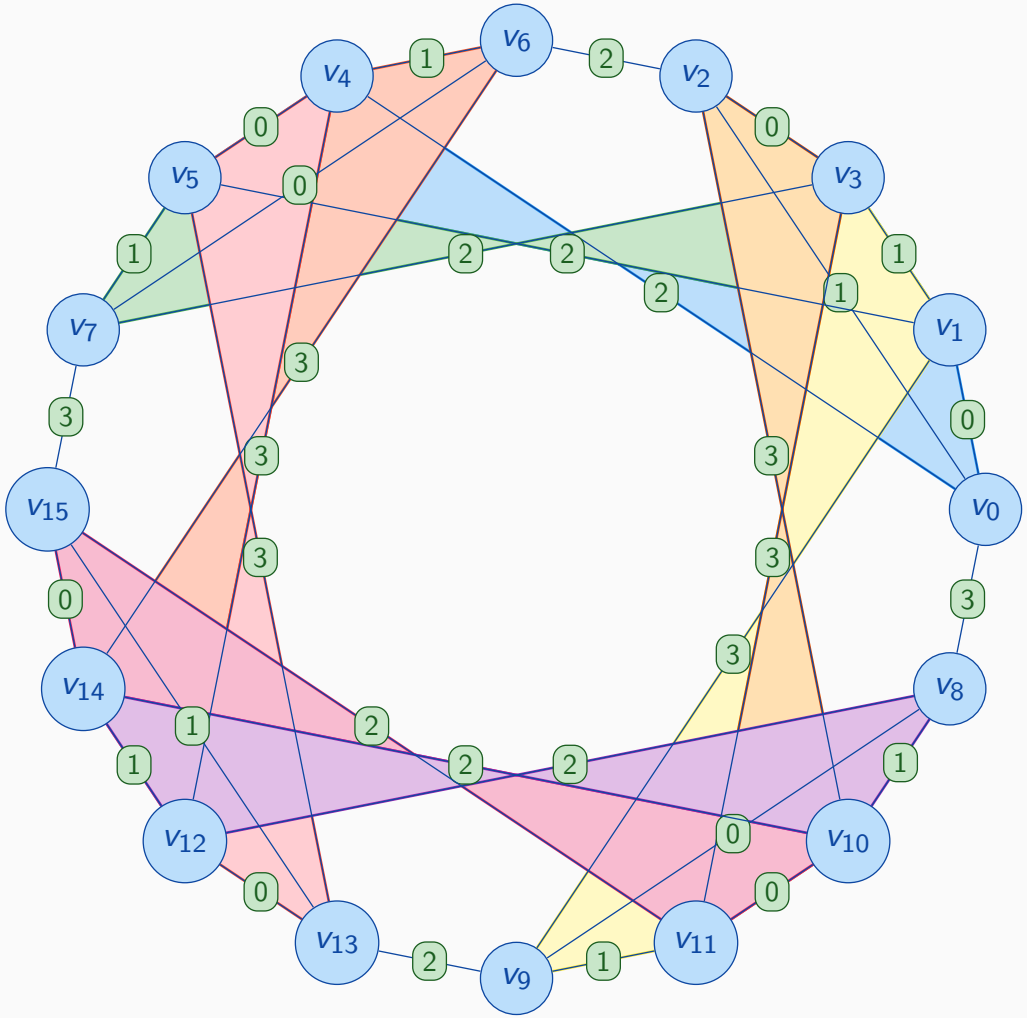
F-sequences in Q_4 vi



F-sequences in Q_4 vii



F-sequences in Q_4 viii



F-sequences in Q_4 ix