Sucesiones-f y ciclos hamiltonianos en el hipercubo

Candidato a doctor 🎓: **Israel Buitrón Dámaso**[1],
Directores: **Guillermo Morales Luna**[1], **Feliú Sagols Troncoso**[2]

5 de octubre de 2018

[1]Departamento de Computación y

[2]Departamento de Matemáticas

Estado actual del trabajo

Propuesta inicial de investigación

- La propuesta inicial esta enfocada en el estudio de protocolos de autenticación, particularmente las pruebas de conocimiento nulo (ZKP's), basadas en problemas difíciles de la teoría de gráficas.
- El análisis de los problemas de gráficas, considerados originalmente, nos mostró la inviabilidad de éstos para su aplicación práctica en los protocolos criptográficos.
- Entonces, el estudio se enfocó al estudio formal de la teoría de gráficas.

Estado actual del trabajo i

- Cada hipercubo es una gráfica hamiltoniana y el número de ciclos hamiltonianos tiene crecimiento doblemente exponencial.
- Describimos cualquier ciclo hamiltoniano mediante una sucesión cuyas entradas son los índices de las direcciones básicas recorridas por el ciclo.
- Esas secuencias de números las llamamos sucesiones-f (suc's-f).
- Las suc's-f se clasifican en clases de equivalencia mediante transformaciones simples, p. ej., rotación circular, reflejo y algunas más.

Estado actual del trabajo ii

- Hemos introducido varias (otras) transformaciones más, para cada una, hemos construido una gráfica de trayectorias particular: los vértices son las suc's-f representantes y las aristas resultan de la transformación, un vértice y su vértice transformado forman una arista.
- Algunos problemas surgen, p. ej. ¿Cuál es el conjunto mínimo de transformaciones que produce una gráfica de trayectorias conexo? y para una transformación dada, ¿Cuál es el número de componentes conexas en la gráfica de trayectorias?

Estado actual del trabajo iii

- Hemos diseñado varios algoritmos para la aplicación de transformaciones y para contestar varios problemas de decisión.
- Hemos construido una biblioteca en C que implementa esos algoritmos.

Sucesiones-f y ciclos hamiltonianos

Hipercubo i

Definición (Gráfica hipercubo)

Sea $Q_n = \{0,1\}^n$, el hipercubo de n dimensiones. Este es un espacio vectorial $\mathbb{F}_2$ de n dimensiones, generado por la base canónica $E = (e_j)_{j=1}^n$, donde $e_j = (\delta_{i,j})_{i=1}^n$ y $\delta_{i,j}$ es la delta de Kronecker. También es un espacio métrico con distancia de Hamming

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

Las *aristas de* Q_n son parejas $(v, v + e_i)$, donde $v \in V(Q_n)$, es decir, estos tiene una distancia de hamming 1. Decimos que *existe una arista a lo largo de la coordenada* i*-ésima*.

Hipercubo ii

Ejemplo

La arista (v_{12}, v_{14}) se extiende a lo largo de la primera coordenada en Q_4. La distancia (de los vértices) es

$$d_H(\underbrace{1100}_{12}, \underbrace{1110}_{14}) = 1.$$

Dado que todos los vértices en Q_n pueden ser enumerados con una cadena binaria de longitud n, entonces serán etiquetados con un subíndice $i \in \mathbb{Z}_{2^n}$ que representa a esa cadena.

Definición (Sucesiones-f)

Una suc-f de orden n con longitud m es una sucesión finita $f = (f_0, \ldots, f_{m-1})$ donde $f_i \in \mathbb{Z}_n$.

Si $m < 2^n$, entonces para cada sucesión propia $(f_k, \ldots, f_l)$, con $0 \leq k < l \leq 2^n - 1$, existe al menos un elemento que aparece un número impar de veces.

Si $m = 2^n$, entonces todos los elementos aparecen un número par de veces.

La longitud máxima de una suc-f de orden n es 2^n. Decimos que ésta es una suc-f *completa*.

Ejemplo

Una suc-f de orden 2 con longitud 2^2 es

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

Una suc-f de orden 3 con longitud 2^3 es

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

Una suc-f de orden 4 con longitud 2^4 es

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

Una suc-f de orden 5 con longitud 2^4 es

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

Tipo de una sucesión i

Definición (Tipo de una sucesión)

Sea $(f_0, \ldots, f_{2^n-1})$ una suc-f completa de orden n.

El *tipo* de una sucesión $(f_i, \ldots, f_j)$, denotada por $tipo(f_i, \ldots, f_j)$, donde $0 \leq i < j < 2^n$, es el conjunto de los elementos en la secuencia que aparecen (en ella) un número impar de veces.

Tipo de una sucesión ii

Ejemplo

Dada la suc-f de Ejemplo 4

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

Equivalencia-f i

Definición (Acciones de la equivalencia-f)

Sea f una suc-f completa de orden n.

1. Si $\pi : \mathbb{Z}_n \to \mathbb{Z}_n$ es una permutación, del grupo S_n, entonces $\pi(f) = (\pi(f_0), \ldots, \pi(f_{2^n-1}))$ es una suc-f completa.
2. La *sucesión reflejo* $f^r = (f_{2^n-1}, \ldots, f_0)$ es una suc-f completa.
3. Cualquier *rotación circular* de f, $(f_i, \ldots, f_{2^n-1}, f_0, \ldots, f_{i-1})$, donde $i \in \mathbb{Z}_{2^n}$, es una suc-f completa.

Equivalencia-f ii

Estas operaciones definen una acción del grupo $S_n \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_{2^n}$ en el conjunto de suc's-f completas.

Diremos que dos suc's-f son *equivalentes-f*, si pertenecen a a la misma órbita (de acción).

Equivalencia-f iii

Ejemplo

La suc-f del Ejemplo 4 es equivalente-f a

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

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

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

Representante de la clase

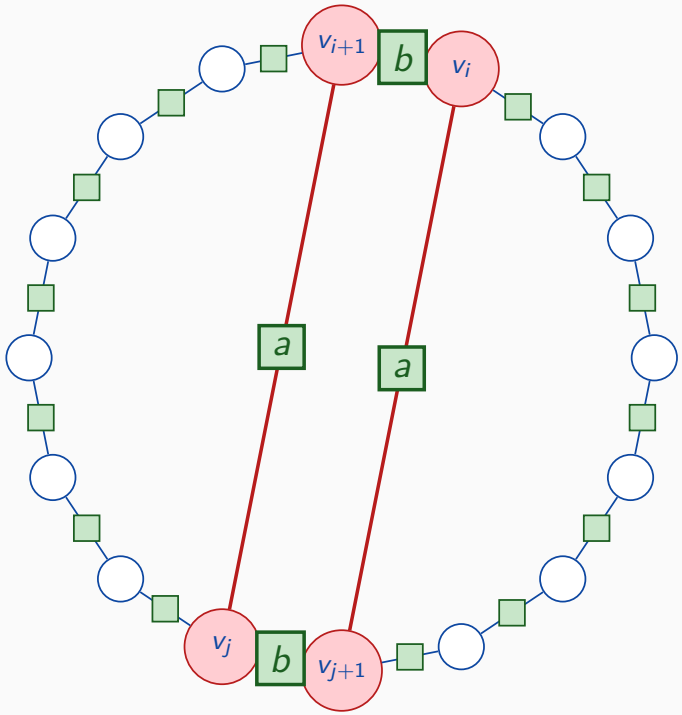
Definición (Sucesión-f mínima)

Una suc-f es mínima, si ésta es el elemento mínimo en su clase de f-equivalence, considerando el orden lexicográfico.

Definición (Cuadro recto)

Sea f una suc-f de orden n y los índices $i, j \in \mathbb{Z}_{2^n}$ con $i < j$.

Si $tipo(v_i, \ldots, v_{i+1}) = tipo(v_j, \ldots, v_{j+1}) = \{b\}$ y $tipo(v_{i+1}, \ldots, v_j) = tipo(v_{j+1}, \ldots, v_i) = \{a\}$, entonces esos cuatro vértices forman un 4-ciclo. Lo llamaremos un cuadro recto.

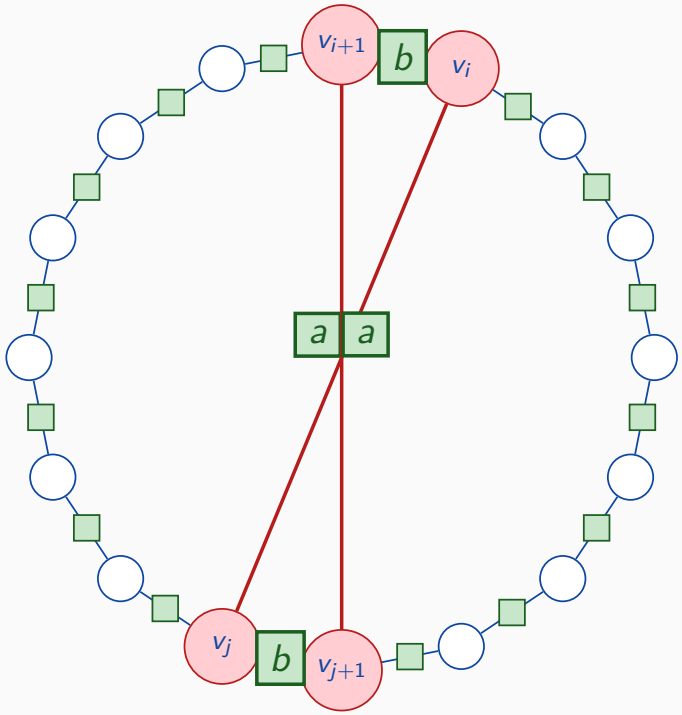


Cuadros ii

Definición (Cuadro torcido)

Sea f una suc-f de orden n y los índices $i, j \in \mathbb{Z}_{2^n}$ con $i < j$.

Si $tipo(v_i, \ldots, v_{i+1}) = tipo(v_j, \ldots, v_{j+1}) = \{b\}$ y $tipo(v_{i+1}, \ldots, v_{j+1}) = tipo(v_i, \ldots, v_j) = \{a\}$, entonces esos cuatro vértices forman un 4-ciclo. Lo llamamos un cuadro torcido.



Operadores de sucesiones-f

Clover Transformación trébol

Definición (Trébol)
Sea f una suc-f de orden n que puede ser escrita como $AaBbCaDb$, donde A, B, C, D son sucesiones propias no vacías de f y a, b sucesiones propias de longitud 1. Todas ellas ajenas entre sí.

Si $tipo(AbD) = tipo(BbC) = a'$ y $tipo(AaB) = tipo(CaD) = b'$, donde a', b' son sucesiones propias de longitud 1, entonces

$$Aa'Db'Ca'Bb'$$

es una suc-f completa no equivalente-f a f. Llamamos a esto una transformación *trébol*.

Transformación trébol torcido i

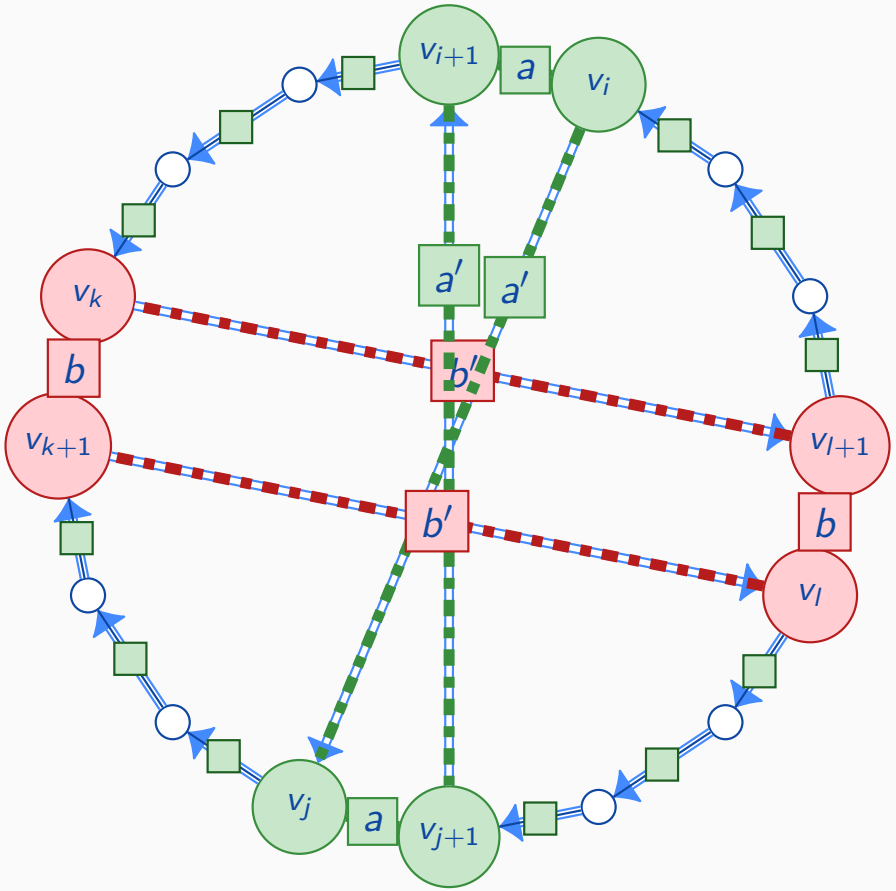
Definición (Trébol torcido)
Sea f una suc-f de orden n que puede ser escrita como $AaBbCaDb$, donde A, B, C, D son sucesiones propias no vacías de f y a, b sucesiones propias de longitud 1. Todas ellas ajenas entre sí.

Si $tipo(AbD) = tipo(BbC) = a'$ y $tipo(AaB) = tipo(CaD) = b'$, donde a', b' son sucesiones propias de longitud 1, entonces

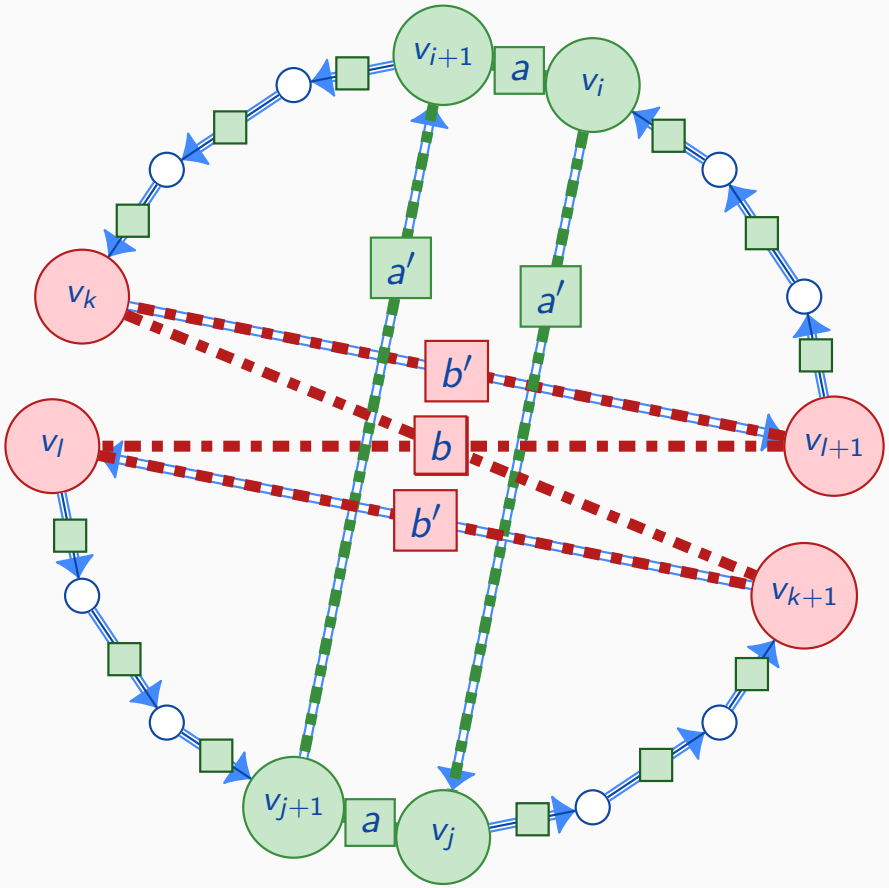
$$Aa'D^r b'C^r b'Bb'$$

es una suc-f completa no equivalente-f a f. Llamamos a esto una transformación *trébol torcido*.

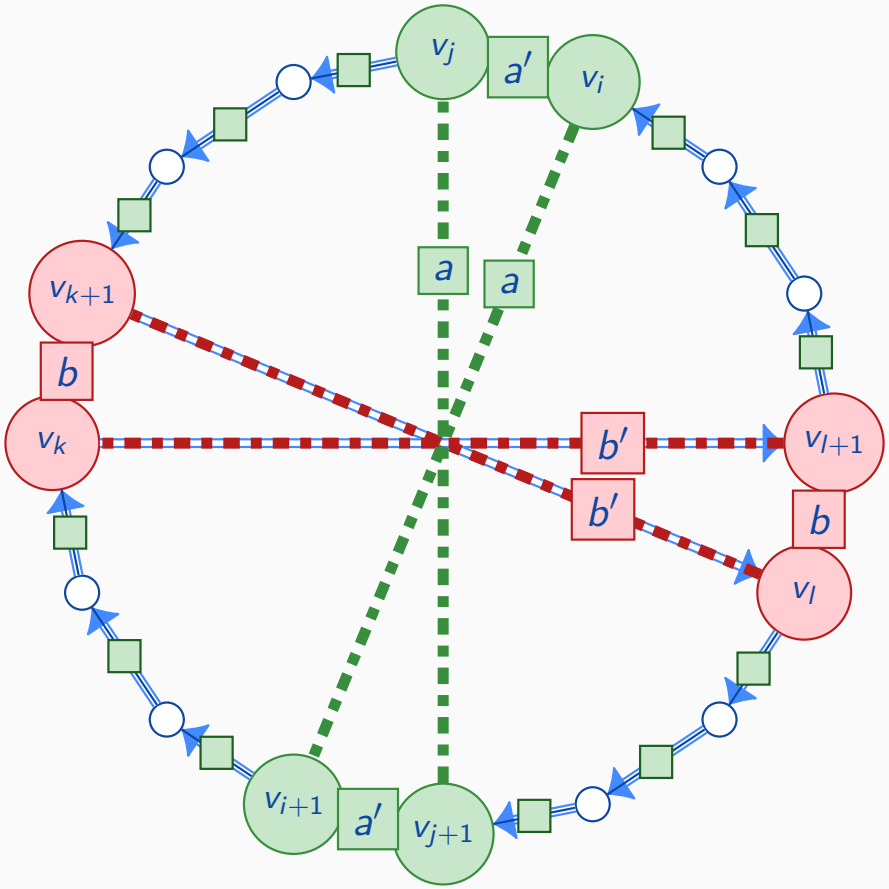
Transformación trébol torcido ii



Transformación trébol torcido iii



Transformación trébol torcido iv



Transformación hongo i

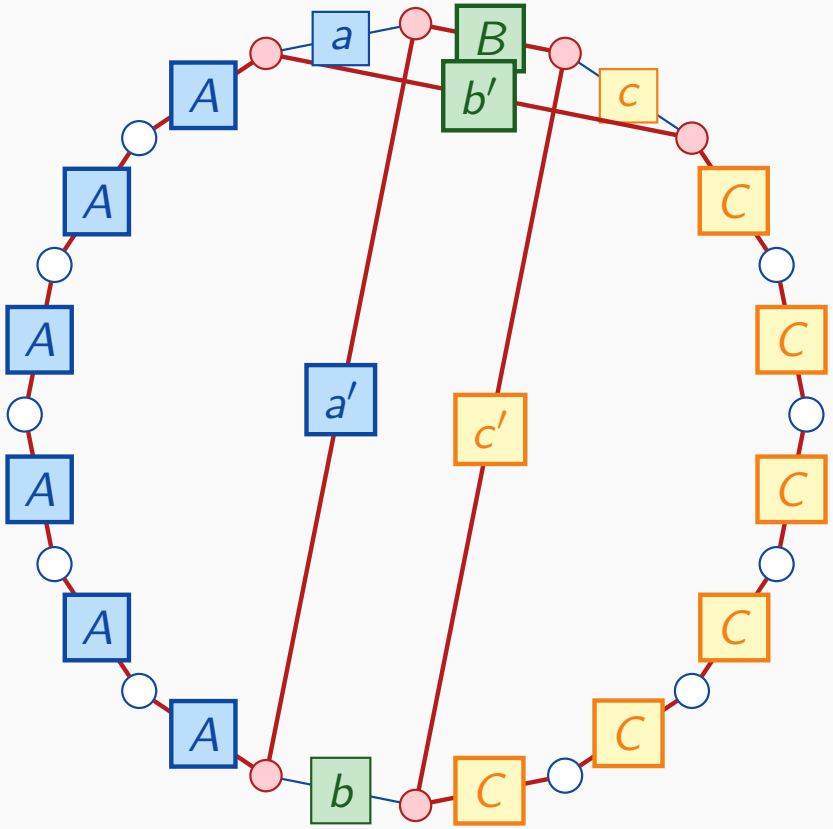
Definición (Hongo)
Sea f una suc-f de orden n que puede ser escrita como $AaBbCc$, donde A, B, C son sucesiones propias no vacías de f y a, b, c sucesiones propias de longitud 1. Todas ellas ajenas entre sí.

Si $tipo(Ab) = a'$, $tipo(aBb) = b'$ y $tipo(bC) = c'$, donde a', b', c' son sucesiones propias de longitud 1, entonces

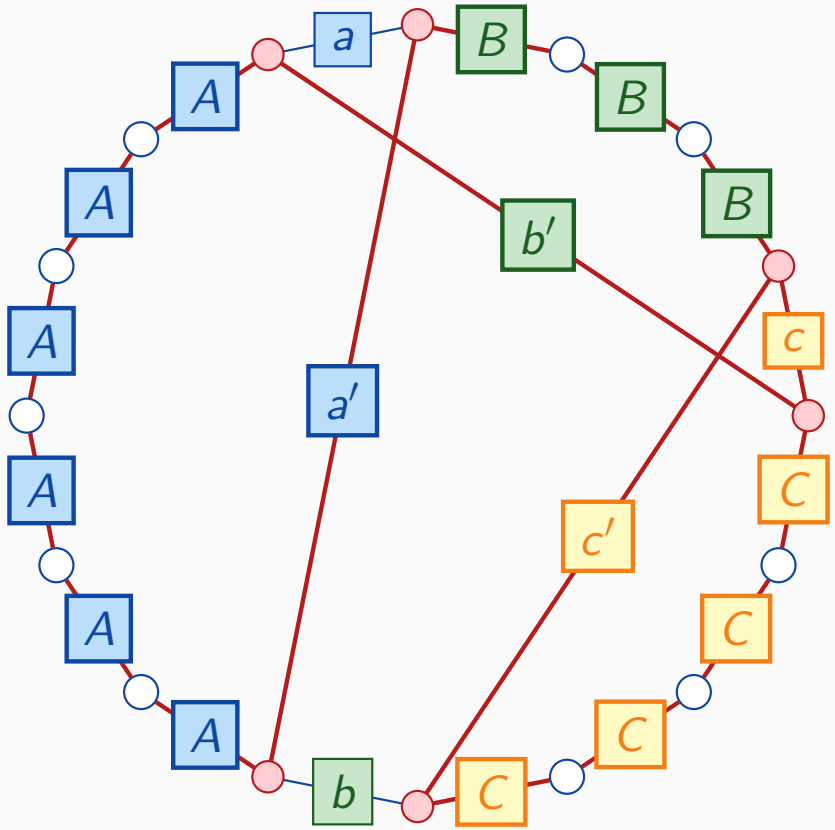
$$Ab'Cc'B^r a'$$

es una suc-f completa no equivalente-f a f. Llamamos a esto una tranformación *hongo*.

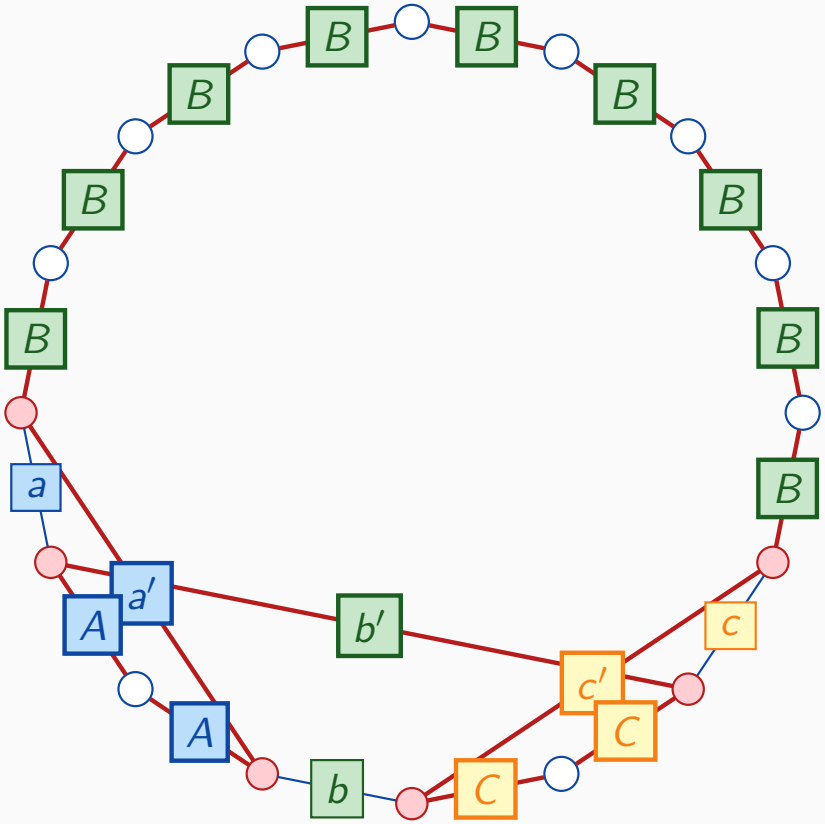
Transformación hongo ii



Transformación hongo iii



Transformación hongo iv



Algoritmos e implementaciones

Algoritmos e implementaciones

Algoritmos

Validación de sucesiones-f

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
```

Generación seudoaleatoria de sucesiones-f

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
```

Búsqueda de cuadros

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
```

Aplicación de tréboles torcidos

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'
```

Algoritmos e implementaciones

Implementación

Fseq4c – Our C Library

- El código fuente es público.
  Está registrado y administrado mediante un repositorio **git**:
  `https://gitlab.com/israelbuitron/fseq4c`
- El código fuente está documentado.

Algoritmos e implementaciones

Demo

Generación seudoaleatoria de sucesiones-f

Ejemplo (Ver video)
Ejecución de la *generación seudoaleatoria de suc's-f* de orden desde 3 a 23, en:

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

Búsqueda de cuadros

Ejemplo (Ejecución)
Ejecutar en una terminal:

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

Búsqueda de instancias de hongos

Ejemplo (Ejecución)
Ejecutar en una terminal:

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

Aplicación de cuadros torcidos

Ejemplo (Ejecución)
Ejecutar en una terminal:

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

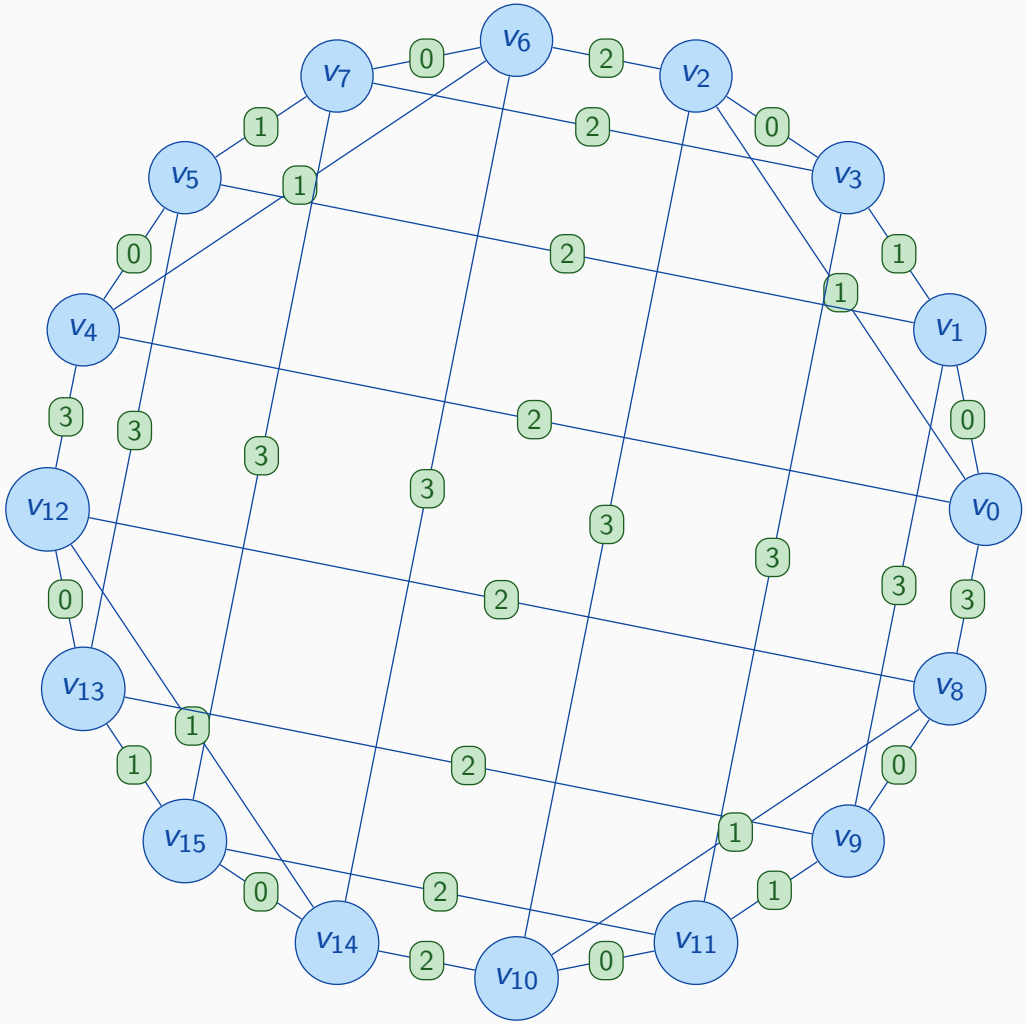
¡Gracias!

:)

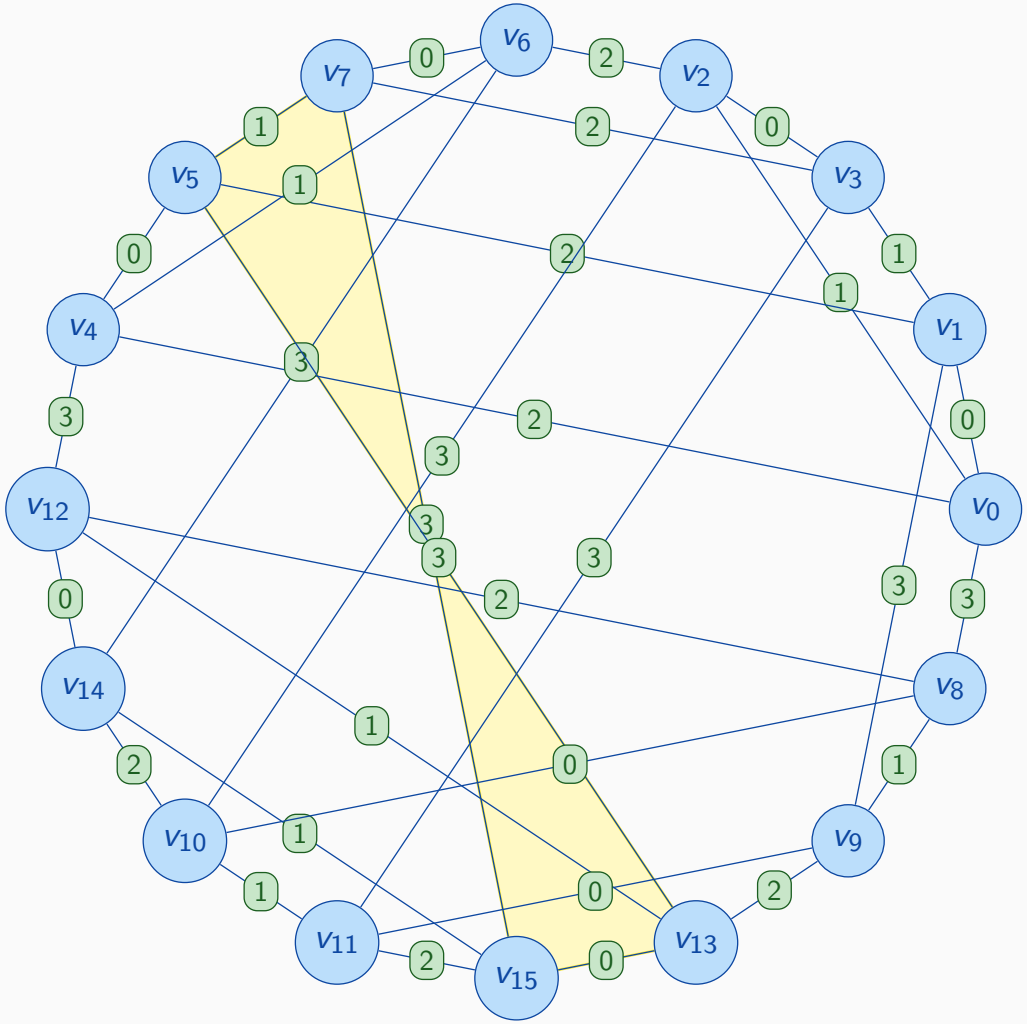
¡Tlazohcamati nonantzintzin!, neh miac nimitzilnamiqui ❤

Sucesiones-f en Q_4

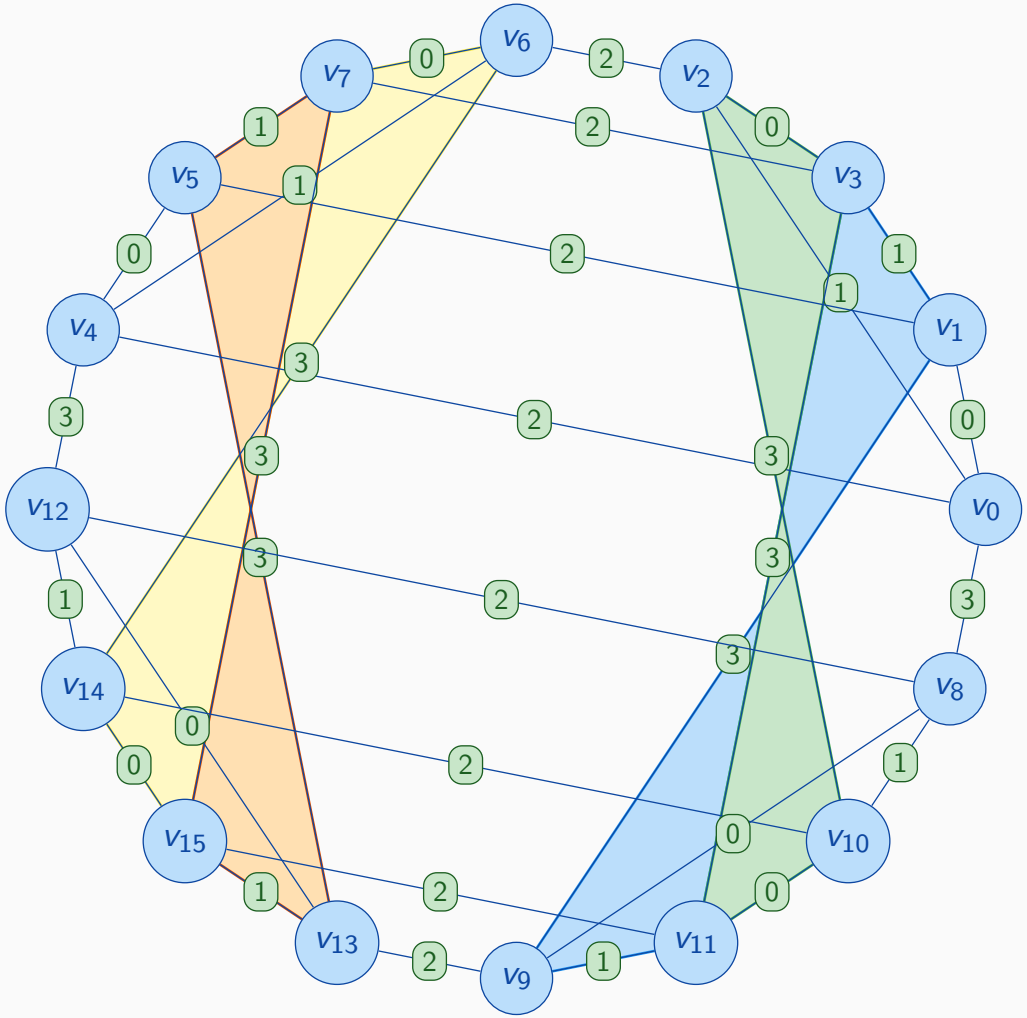
Sucesiones-f en Q_4 i



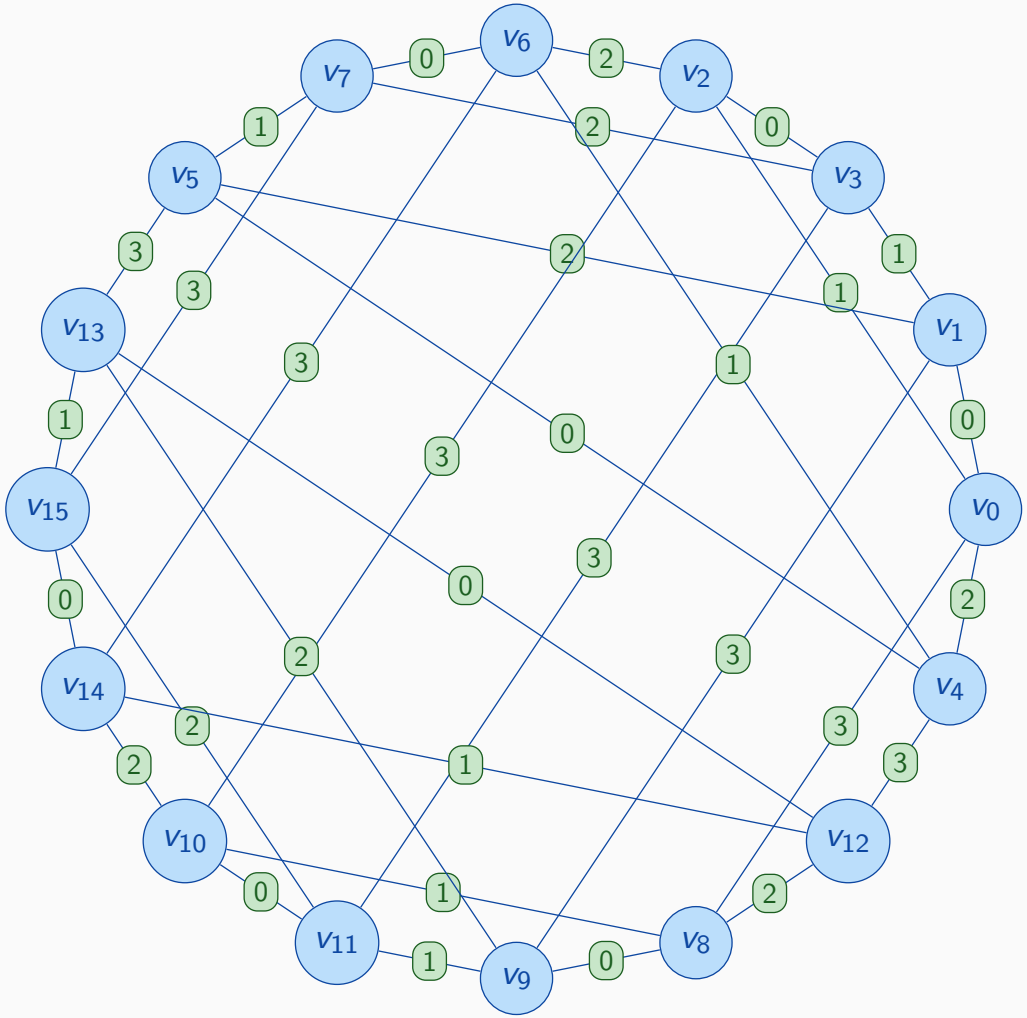
Sucesiones-f en Q_4 ii



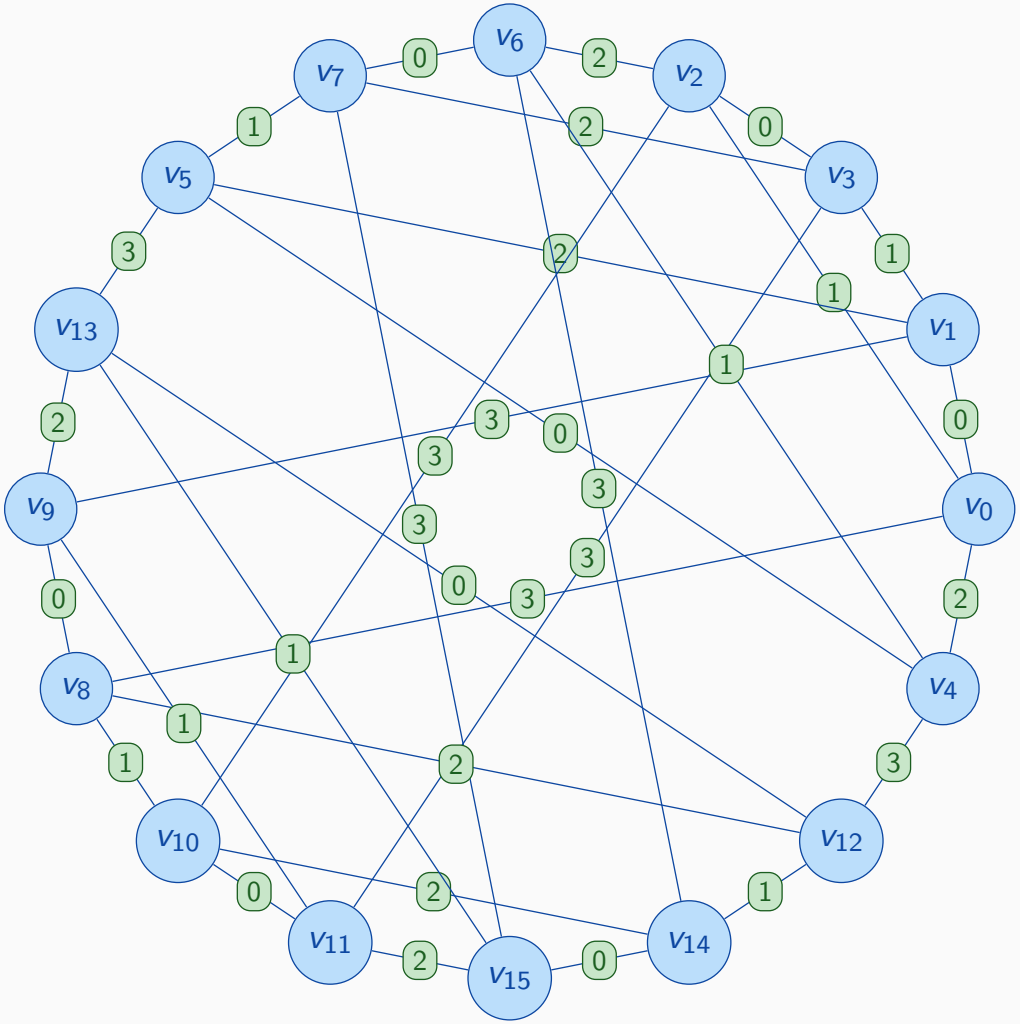
Sucesiones-f en Q_4 iii



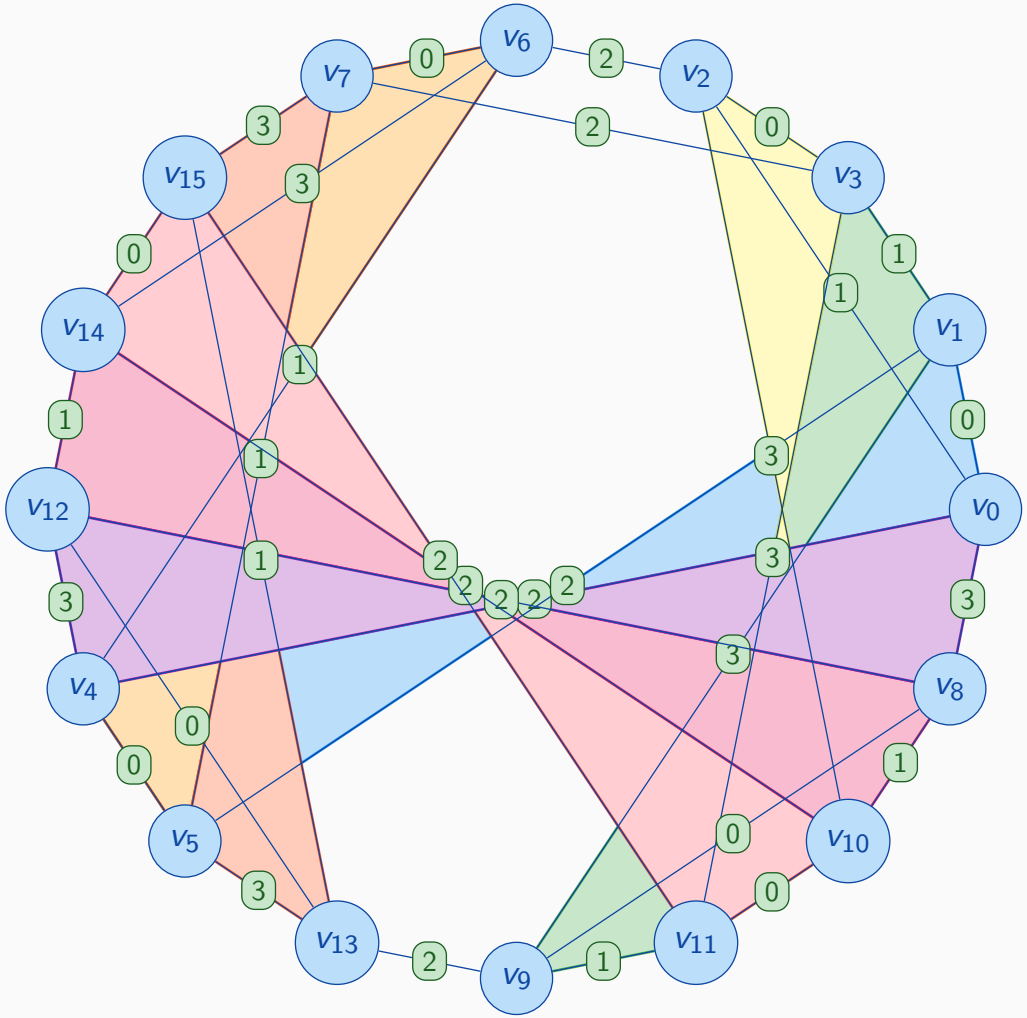
Sucesiones-f en Q_4 iv



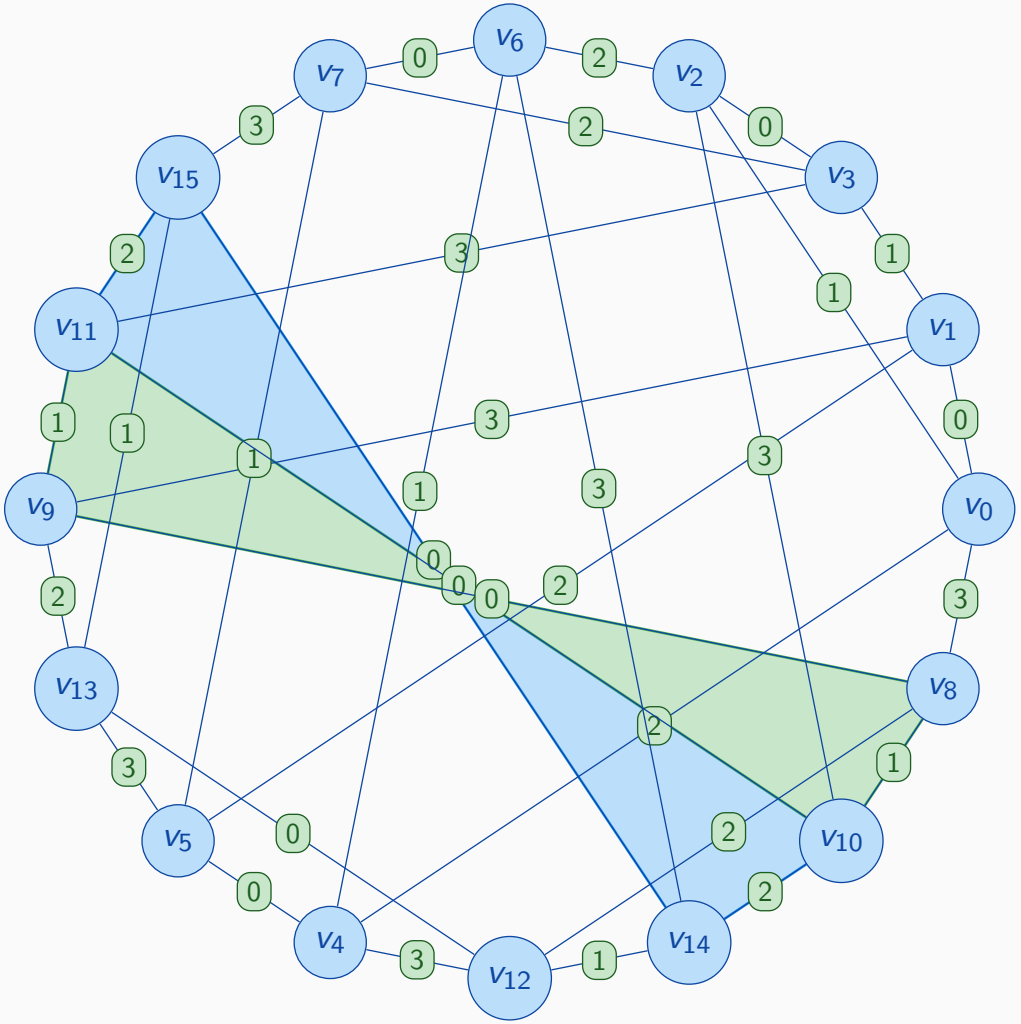
Sucesiones-f en Q_4 v



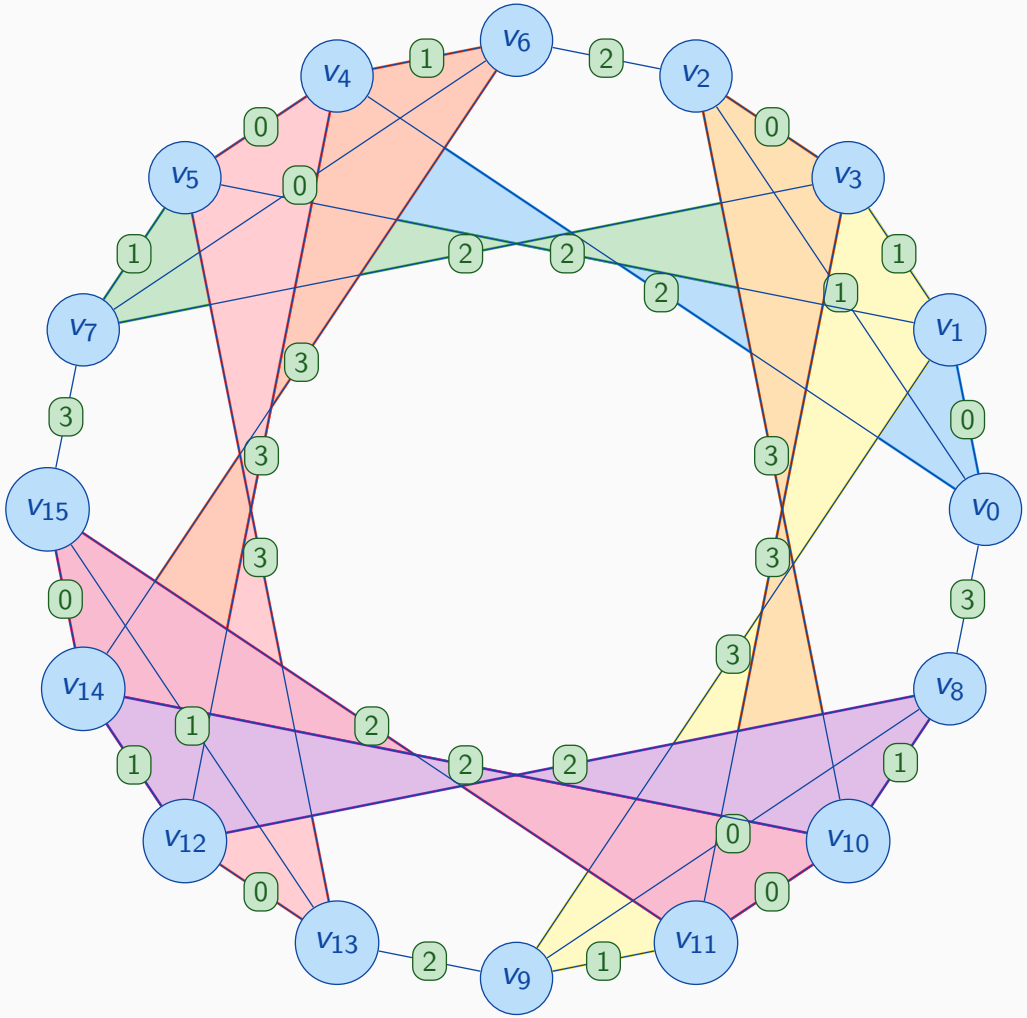
Sucesiones-f en Q_4 vi



Sucesiones-f en Q_4 vii



Sucesiones-f en Q_4 viii



Sucesiones-f en Q_4 ix