UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO
FACULTAD DE CIENCIAS FÍSICO-MATEMATICAS
Mát. Luis M. Rivera Gutiérrez

Comprobación Computacional de una Conjetura de Tablas de Marcas y Anillos de Burnside

TESIS

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS FÍSICO-MATEMATICAS

PRESENTA:

ARIEL MOLINA RUEDA

ASESOR:

LUIS VALERO ELIZONDO

MORELIA, MICHOACÁN, OCTUBRE DEL 2006.

Comprobación Computacional de una Conjetura de Tablas de Marcas y Anillos de Burnside

Ariel Molina Rueda

Gracias a todos

A mi mamá y a mi papá por quererme tanto.

A Edith, Anhuar y Erick.

También a Piña y Bololo.

A Karina.

A mi profe de Historia de la secundaria, a la de Química y al Many.

A Netza, Memo, Anabel, Ignacio, Teresa. También a Norma, Julieth, Juan, Huacu y Anito. A Chucho[1], Panza, Lhuerta, Ubaldo, Abel, El Mario, Pinky, Lázaro y Alejandra.

A mis profesores y profesoras en Fismat.

A todos los que me han arruinado la vida diciéndome que estudiar Matemáticas te hace felíz.

A mi asesor Luis Valero por su enorme paciencia y sus brillantes ideas, finalmente, esta tesis es idea suya.

Morelia, Michoacán
Ariel Molina Rueda
Otoño, 2006

[1]Ojalá que ya no te caigan más perros del cielo

Índice general

RESUMEN. Dados dos grupos tal que que sus Tablas de Marcas son isomorfas, sus Anillos de Burnside son tambien isomorfos. Pero dados dos grupos no isomorfos, con Tablas de Marcas no isomorfas ¿Son sus Anillos de Burnside isomorfos, o no?

Se conjetura que sí, pero en esta tesis tratamos de indagar más acerca de este tema. Usando GAP (*Groups, Algorithms, Programming*) intentamos encontrar un contraejemplo, y sí, encontramos algunos, pero eran falsos positivos.

Por otro lado, a nuestros algoritmos los pusimos a buscar grupos con Tablas de Marcas Isomorfas y con Anillos de Burnside isomorfos; en nuestra búsqueda encontramos un ejemplo muy interesante, dos grupos de órden 96 que tienen Tablas de Marcas isomorfas. Encontramos que dichos grupos no son isomorfos y recordamos que Thevenaz ya habia analizado este tipo de pares de grupos con anterioridad. De acuerdo a Thevenaz, el par de grupos no isomorfos con órden mas chico y tal que tienen Tablas de Marcas Isomorfas es 5×11^2, esto es, un par de órden 605. Antes de esta tesis, el par con órden mas pequeño conocido era éste ejemplo de Thevenaz. Nosotros encontramos un par que cumple estas características y ademas tienen órden 96, mas de 500 órdenes mas chico. Este es un resultado nuevo que presentamos.

CAPíTULO 1

Requisitos Previos

1. Grupos

Un Grupo G es un conjunto finito o infinito de elementos con una operacion binaria, que conjuntamente satisfacen las cuatro propuiedades fundamentales de Cerradura, Asociatividad, Propiedad de Indentidad y Existencia de Inverso. Como se dijo, la operacion bajo la cual se define un grupo es llamada la "Operación de Grupo" y al conjunto se le dice que es un Grupo bajo esta operación.

DEFINICIÓN 1.1. *Sea el conjunto X, y $(\cdot)$ una operacion binaria, entonces decimos que $G = \{X, \cdot\}$ es un grupo bajo la operacion binaria $(\cdot)$, y $(\cdot)$ es la Operacion de Grupo del grupo G si se cumple que:*

1. *Cerradura. La operación es cerrada sobre G. Si $a, b \in G$ entonces $a \cdot b \in G$.*
2. *Asociatividad. La operación es asociativa. $\forall a, b, c \in G$*

$$(a \cdot b) \cdot c = a \cdot (b \cdot c)$$

3. *Identidad. Existe un elemento identidad, a veces llamado 1 o e. Es decir, $\exists e$ tal que $e \cdot a = a \cdot e = a$ para todo $a \in G$*
4. *Inverso. Todo elemento en G tiene un recíproco bajo la operacion. $\forall a \in G$,*

$$\exists b = a^{-1}, a \cdot a^{-1} = e$$

OBSERVACIÓN. Un grupo es un monoide, donde ademas cada elemento tiene inverso. Cuando no se preste a mala interpretación en lugar de escribir $a \cdot b$ se escribe simplemente ab. En esta tesis sólo se hablará de grupos de orden finito.

1.1. Ejemplos de Grupos.

- Grupos Simetricos. Los grupos simétricos son grupos de permutaciones, bajo la operación "permutación" de n objetos.

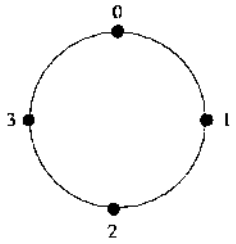

FIGURA 1. $\mathbb{Z}_4$

CÓDIGO FUENTE 1.2. *Código GAP para obtener grupos simétricos y algunos ejemplos simples de como obtener información.*

```
gap> s3 := SymmetricGroup(3);
Sym( [ 1 .. 3 ] )
gap> Identity(s3);
()
gap> s4 := SymmetricGroup(4);
Sym( [ 1 .. 4 ] )
gap> (1,3) in s4;
true
gap>
```

- Grupos cíclicos. Los grupos cíclicos son isomorfos a los enteros módulo n y se denotan $\mathbb{Z}_n$ o C_n, o inclusive $\mathbb{Z}/n\mathbb{Z}$.

CÓDIGO FUENTE 1.3. *Código GAP para obtener grupos cíclicos.*

```
gap> c := CyclicGroup(4);
<pc group of size 4 with 2 generators>
gap>
```

1.2. GAP. GAP tiene una biblioteca de grupos muy grande, y usando el comando `SmallGroup(n,k)`, se puede obtener el k-ésimo grupo de orden n de la biblioteca. Por ejemplo, sabemos que de orden 6 solo existen, el Simetrico S_3 y el Ciclico C_6.

```
gap> NumberSmallGroups(6);
```

```
2
gap> StructureDescription( SmallGroup(6,1) );
"S3"
gap> StructureDescription( SmallGroup(6,2) );
"C6"
gap>
```

Tambien se pueden obtener otros grupos con GAP, ademas de preguntarle a GAP cuantos grupos existen de un determinado orden. Sobretodo lo segundo nos será util para el algoritmo de canonización mencionado en los capítulos finales.

```
gap> NumberSmallGroups(128);
2328
gap> g:=SmallGroup(128,1307);
<pc group of size 128 with 7 generators>
gap>
```

DEFINICIÓN 1.4. *Sea G un grupo, y H un subconjunto de G que cumple a su vez las propiedades de grupo, decimos que H es subgrupo de G y lo denotamos $H \leq G$*

2. G-Conjuntos

NOTACIÓN 1.5. *A lo largo de este trabajo G denota un grupo finito. Además de que H, K se toman como subgrupos de G.*

DEFINICIÓN 1.6. *Un **G-Conjunto** es un conjunto X en donde se define una **acción** de G. Es decir, una función*

$$\cdot : G \times X \to X$$

$$\cdot(g, x) \mapsto g \cdot x$$

que cumple lo siguiente:

1. $1 \cdot x = x; \forall x \in X$
2. $(gh) \cdot x = g \cdot (h \cdot x)$*; donde la operación gh es la del grupo y las demás están dadas por la acción $\cdot$.*

NOTACIÓN 1.7. *gh denota la operación del grupo, mientras que $g \cdot x$ la acción del grupo en el conjunto.*

Cuando un grupo actua en un conjunto, lo que hace es permutar los elementos del conjunto, la identidad del grupo no hace nada (no permuta) y una composición de

2.0.1. Ejemplos:

1. Sea G arbitrario, $X = \emptyset$. La acción de G es la única. A X se le llama el **G-conjunto vacío**.
2. Sea G arbitrario, X un conjunto con un solo elemento x. La acción también es única:

   $$g \cdot x = x; \forall g \in G$$

   a X se le llama el **G-Conjunto trivial**
3. Sea G arbitrario, $X = G$. La acción:

   $$g \cdot x = gx$$

   a X de le llama el **G-Conjunto regular**
4. Sea G arbitrario, $X = G$. La acción es la conjugación en los elementos de G.

   $$g \cdot x = gxg^{-1}$$

5. Sea G arbitrario, $X = \{H | H \leq G\}$ los subgrupos de G. La acción por conjugación en los subgrupos de G. Es decir

   $$g \cdot H = gHg^{-1}$$

6. Sea G arbitrario, X un subgrupo normal de G. La acción es

   $$g \cdot x = gxg^{-1}; \forall x \in X$$

7. Sea G arbitrario, p primo tal que $p \mid |G|$, X la familia de p-subgrupos de Sylow de G. La acción

   $$g \cdot P = gPg^{-1}$$

8. Sea G arbitrario, X la familia de subgrupos cíclicos de G. $X = \{H | H \leq G, H$ cíclico$\}$. La acción
$$g \cdot H = gHg^{-1}$$
9. Sea G arbitrario, n entero positivo, $X = \{x \in G | orden(x) = n\}$
$$g \cdot x = gxg^{-1}$$
10. Sea G arbitrario, $H \leq G$ subgrupo normal, $X = G/H = \{xH | x \in G\}$. La acción
$$g \cdot (xH) = (gx)H$$
Esto está bien definido, pues si $xH = yH \Rightarrow (gx)H = (gy)H$
11. Sea n un entero positivo, $X = \{1, 2, \ldots, n\}$, $G = S_n$ con la acción natural, es decir:
$$\pi \cdot x = \pi(x)$$
12. Sea k un campo, V un k-espacio vectorial, $X = V$, $G = GL(V)$. La acción
$$T \cdot v = T(v)$$
OBSERVACIÓN.
$$GL(V) = \{T : V \to V | T \text{ es isomorf. de esp. vect.}\}$$
13. Sea G arbitrario, $H \leq G$ subgrupo normal, $X = G \backslash H = \{Hx | x \in G\}$. La acción
$$g \cdot (Hx) = H(xg^{-1})$$
Esto está bien definido, pues si $Hx = Hy \Rightarrow H(xg^{-1}) = H(yg^{-1})$
14. Sea X arbitrario, $G = S_X = \{f : X \to X | f$ es biyección$\}$, la acción natural
$$f \cdot x = f(x)$$

OBSERVACIÓN. Sean G un grupo y X un G-conjunto. Sea
$$\mu : G \to S_X$$

dada por:

$$\mu(g) : X \to X$$
$$\mu(g)(x) = g \cdot x$$

$\mu(g)$ es biyección, pues tiene inversa $(\mu(g))^{-1} = \mu(g^{-1})$ Entonces μ es homomorfismo de grupos

$$\mu(gh)(x) = (gh) \cdot x = g \cdot (h \cdot x) = \mu(g)(\mu(h)(x))$$

Inversamente. Si X es un conjunto arbitrario y $\mu : G \to S_X$ un homomorfismo de grupos, entonces X se convierte en G-Conjunto con la acción $g \cdot x = \mu(g)(x)$

2.1. Órbitas y estabilizadores.

DEFINICIÓN 1.8. *Sean X un G-Conjunto y $x \in X$. La **órbita** de x es el conjunto*

$$G \cdot x = \{g \cdot x | g \in G\}$$

DEFINICIÓN 1.9. *El **estabilizador** de x es el subgrupo*

$$Stab_G(x) = \{g \in G | g \cdot x = x\}$$

CÓDIGO FUENTE 1.10. *En gap se pueden obtener las órbitas y estabilizadores asi:*

```
gap> gg:=Group((1,2,3,4)(7,8,9)(5,6));
Group([ (1,2,3,4)(5,6)(7,8,9) ])
gap> Orbit(gg, [1], OnSets);
[ [ 1 ], [ 3 ], [ 2 ], [ 4 ] ]
gap> Orbit(gg, [5], OnSets);
[ [ 5 ], [ 6 ] ]
gap> gg:=Group((1,2,3,4)(7,8,9)(5,6));
Group([ (1,2,3,4)(5,6)(7,8,9) ])
gap> Stabilizer(gg, [7], OnSets);
Group([ (1,2,3,4)(5,6) ])
gap> Stabilizer(gg, [4], OnSets);
Group([ (7,8,9) ])
gap>
```

PROPOSICIÓN 1.11. *Si $g \in G$, entonces: $Stab_G(g \cdot x) = gStab_G(x)g^{-1}$*

DEMOSTRACIÓN. Si $h \in Stab_G(g \cdot x)$ entonces $h \cdot (g \cdot x) = g \cdot x$. Por demostrar: $\exists\ k \in Stab_G(x)$ tal que $h = gkg^{-1}$. Sea $k = g^{-1}hg$, obviamente k es de la forma deseada, queremos ver que está en el estabilizador de x

$$k \cdot x = (g^{-1}hg) \cdot x = g^{-1} \cdot (h \cdot (g \cdot x)) = g^{-1} \cdot (g \cdot x) = (g^{-1}g) \cdot x = 1 \cdot x = x$$

La otra contención es clara, si $k \in Stab_G(x)$,

$$(gkg^{-1}) \cdot (g \cdot x) = gk(g^{-1}g) \cdot x = gk \cdot x = g \cdot x$$

□

PROPOSICIÓN 1.12. *Sea X un G-Conjunto no vacío. Entonces X se parte como la unión disjunta de sus órbitas.*

DEMOSTRACIÓN. Para cada $x \in X$, $x \in G \cdot x$, así cada elemento está en al menos una órbita. Hay que demostrar que dos órbitas distintas son ajenas o que si se intersectan entonces coinciden. Sean θ_1, θ_2 órbitas de X, digamos

$$\theta_1 = G \cdot x$$

$$\theta_2 = G \cdot y$$

Supongamos que $\exists z \in \theta_1 \cap \theta_2$, es decir, $\exists g, h \in G$ tal que $z = g \cdot x = h \cdot y$. Se sigue que

$$g^{-1} \cdot (g \cdot x) = g^{-1} \cdot (h \cdot y)$$

y

$$x = 1 \cdot x = (g^{-1}g) \cdot x = g^{-1} \cdot (g \cdot x) = g^{-1} \cdot (h \cdot y) = (g^{-1}h) \cdot y$$

entonces $x \in \theta_2$. Así $\forall k \in G$ se tiene

$$k \cdot x = k \cdot ((g^{-1}h) \cdot y) = (kg^{-1}h) \cdot y \in \theta_2$$

y por lo tanto $\theta_1 \subseteq \theta_2$, análogamente $\theta_2 \subseteq \theta_1$ y $\theta_1 = \theta_2$. □

DEFINICIÓN 1.13. *Sean X, Y G-Conjuntos, $f : X \to Y$ una función. Decimos que f es un* ***homomorfismo de G-Conjuntos*** *si $f(g \cdot x) = g \cdot f(x) \quad \forall g \in G, \forall x \in X$. Decimos que f es un* ***isomorfismo de G-Conjuntos*** *si es un homomorfismo biyectivo.*

OBSERVACIÓN. Inversos y composiciones de isomorfismos son isomorfismos. La identidad es isomorfismo.

DEFINICIÓN 1.14. *Sea X un G-Conjunto. Decimos que X es* ***transitivo*** *si $X \neq \emptyset$ y X consta de una sola órbita.*

PROPOSICIÓN 1.15. *Sea X un G-Conjunto transitivo, sea*

$$x \in X, H = Stab_G(x)$$

Entonces $X \cong G/H$ como G-Conjuntos.

DEMOSTRACIÓN. Sea $f : G/H \to X$ dada por $f(gH) = g \cdot x$. Tenemos que $\forall h \in H, f(gh) = (gh) \cdot x = g \cdot (h \cdot x) = g \cdot x$ por lo que f está bien definida. Como X es transitivo entonces f es suprayectiva. Además, $\forall k, g \in G$,

$$f(k \cdot (gH)) = f((kg)H) = (kg) \cdot x = k \cdot (g \cdot x) = k \cdot f(gH)$$

por lo que f es homomorfismo de G-Conjuntos.

Finalmente, si $f(gH) = f(kH)$, entonces

$$g \cdot x = k \cdot x$$

$$\Rightarrow x = (g^{-1}k) \cdot x$$

$$\Rightarrow g^{-1}k \in H$$

$$\Rightarrow gH = kH$$

y f es inyectiva y por lo tanto isomorfismo de G-conjuntos. □

COROLARIO. Sean $H \leq G, g \in G$ entonces $G/H \cong G/gHg^{-1}$.

DEMOSTRACIÓN. Sea $X = G/H, x = H$, tenemos que $Stab_G(x) = H$, por lo que $X \cong G/H$.

Por otro lado, $g \cdot x \in X$ transitivo y $Stab_G(g \cdot x) = gHg^{-1}$, así que $X \cong G/gHg^{-1}$ por transitividad $G/H \cong G/gHg^{-1}$. $\square$

CAPíTULO 2

Anillos de Burnside

LEMA 2.1. *Sean X, Y G-conjuntos. La unión disjunta $X \sqcup Y$ es un G-Conjunto con la acción de G ya existente a X y Y. También tenemos que el producto cartesiano $X \times Y$ es un G-Conjunto con la acción de G dada por $g \cdot (x, y) = (g \cdot x, g \cdot y)$.*

DEMOSTRACIÓN. Sea $a \in X \sqcup Y$, si $a \in X$ entonces

$$(gh) \cdot a = g \cdot (h \cdot a), \tag{1}$$

$$1 \cdot a = a \tag{2}$$

por la razón de que X es un G-Conjunto, tenemos tambien, que si $a \in Y$.

$$(gh) \cdot (x, y) = ((gh) \cdot x, (gh) \cdot y) = \tag{3}$$

$$= (g \cdot (h \cdot x), g \cdot (h \cdot y)) = \tag{4}$$

$$= g \cdot (h \cdot x, h \cdot y) = x \cdot (h \cdot (x, y)) \tag{5}$$

y entonces tenemos que,

$$1 \cdot (x, y) = (1 \cdot x, 1 \cdot y) = (x, y) \tag{6}$$

□

DEFINICIÓN 2.2. *Sea C un conjunto y dos operaciones binarias $+, \cdot$, decimos que $S = (C, +, *)$ es un **semianillo** si S es tal que satisface las siguientes condiciones:*

1. *Asociatividad de la suma: $\forall a, b, c \in S$,*

$$(a + b) + c = a + (b + c)$$

2. *Conmutatividad de la suma: $\forall a, b, c \in S$,*

$$a + b = b + a$$

3. *Asociatividad del producto:* $\forall a, b, c \in S$.

$$(a * b) * c = a * (b * c)$$

4. *Distributividad izquierda y derecha:* $\forall a, b, c \in S$

$$a * (b + c) = (a * b) + (a * c)$$
$$(b + c) * a = (b * a) + (c * a)$$

DEFINICIÓN 2.3. *Si S es un semianillo y además le exigimos las siguientes condiciones,*

5. *Existencia de neutro aditivo:* $\exists 0 \in S$ *tal que si* $a \in S$,

$$a + 0 = 0 + a = a$$

6. *Existencia de neutro multiplicativo:* $\exists 1 \in S$ *tal que si* $a \in S$,

$$a * 1 = 1 * a = a$$

7. *Cancelación:* $\forall a, b \in S$, *Si* $a + b = a + c \Rightarrow a = b$
8. *Connmutatividad del producto:* $\forall a, b \in S$,

$$a * b = b * a$$

9. $\exists 0 \in S$ *tal que so* $a \in S$, $a * 0 = 0 * a = 0$

entonces a S le llamamos un ***Semianillo conmutativo*** *con* 1 *y con cancelación.*

OBSERVACIÓN. Cabe mencionar lo siguiente en cuanto a anillos,

1. El ejemplo más simple y no trivial de un semianillo es $\mathbb{N}$, los numeros naturales incluyendo al 0, con la suma y producto ordinarios.
2. La diferencia entre anillos y semianillos es que en un Semianillo, la adición forma un monoide connmutativo, no necesariamente un grupo abeliano.
3. Pregunta: ¿Qué es un grupo abeliano bajo adicion, cerrado, asociativo, distributivo, y posee una maldición? [1].

[1]El Anillo del Nibelungo

LEMA 2.4. *Sea S un semianillo conmutativo con 1. Sea $A_1 = \{(a,b)|a,b \in S\}$, intuitivamente $(a,b) = a - b$. Tomemos $a \sim$ como la relación de equivalencia dada por*

$$(a,b) \sim (c,d) \Leftrightarrow a + d = b + c \tag{7}$$

lo que nos hace pensar naturalmente que $a - b = c - d$.

Sea $A = A_1/\sim$. Procedamos a definir $+$ y $\cdot$ en A como

$$(a,b) + (c,d) = (a + c, b + d) \tag{8}$$

$$(a,b) \cdot (c,d) = (ac + bd, bc + ad) \tag{9}$$

Entonces $+$ y $\cdot$ están bien definidas y $(A, +, \cdot)$ es un anillo conmutativo con 1. Además S está metido en A por

$$a \mapsto (a + x, x) \tag{10}$$

para un x fijo.

DEMOSTRACIÓN. Demostraremos que,

1. $\sim$ **es de equivalencia**
   - Reflexividad. $(a,b) \sim (a,b)$ pues es directo que $a + b = b + a$.
   - Simetría. Si $(a,b) \sim (c,d)$ ent $(c,d) \sim (a,b)$.

$$(a,b) \sim (c,d) \Rightarrow a + d = b + c \tag{11}$$

$$c + b = d + a \Rightarrow (c,d) \sim (a,b) \tag{12}$$

   - Transitividad. Si $(a,b) \sim (c,d)$ y $(c,d) \sim (x,y)$ entonces,

$$(a,b) \sim (c,d) \Rightarrow a + d = b + c, \tag{13}$$

$$(c,d) \sim (x,y) \Rightarrow c + y = d + x \tag{14}$$

   sumando ambas ecuaciones y cancelando queda,

$$a + d + c + y = b + c + d + x \tag{15}$$

$$a + y = b + x \tag{16}$$

   y por lo tanto $(a,b) \sim (x,y)$.

2. $+$ está bien definida. Si $(a,b) \sim (a',b')$ y $(c,d) \sim (c',d') \Rightarrow (a+c, b+d) \sim (a',b') + (c',d')$.

Tenemos

$$(a,b) \sim (a',b') \Rightarrow a + b' = b + a' \tag{17}$$

$$(c,d) \sim (c',d') \Rightarrow c + d' = d + c' \tag{18}$$

sumando ambas ecuaciones tenemos,

$$(a+b') + (c+d') = (b+a') + (d+c') \tag{19}$$

$$a + b' + c + d' = b + a' + d + c' \tag{20}$$

$$(a+c) + (b'+d') = (b+d) + (a'+c') \tag{21}$$

$$\Rightarrow (a+c, b+d) \sim (a',b') + (c',d') \tag{22}$$

3. $\cdot$ está bien definida. $(a,b) \cdot (c,d) = (ac+bd, bc+ad) \sim (a',b') \cdot (c',d') = (a'c' + b'd', b'c' + a'd')$.

Veamos primero que $(a,b) \cdot (c,d) = (ac+bd, bc+ad) \sim (a',b') \cdot (c,d) = (a'c + b'd, b'c + a'd)$ pues

$$ac + bd + a'd + b'c = \tag{23}$$

$$= (b+a')d + (a+b')c \qquad = \tag{24}$$

$$= (a'+b)c + (a+b')d \qquad = \tag{25}$$

$$= ad + bc + a'c + b'd \tag{26}$$

Luego

$$(a',b') \cdot (c,d) = \tag{27}$$

$$= (a'c + b'd, b'c + a'd) \sim (a',b') \cdot (c',d') \qquad = \tag{28}$$

$$= (a'c' + b'd', b'c' + a'd') \tag{29}$$

puesto que

$$a'c + b'd + b'c' + a'd' = \tag{30}$$

$$= a'(c + d') + b'(c' + d) = \tag{31}$$

$$= a'(c' + d) + b'(c + d') = \tag{32}$$

$$= a'd + b'c + a'c' + b'd' \tag{33}$$

4. + es asociativa.

$$((a, b) + (c, d)) + (x, y) = \tag{34}$$

$$= (a + c, b + d) + (x, y) = \tag{35}$$

$$= ((a + c) + x, (b + d) + y) = \tag{36}$$

$$= (a + (c + x), b + (d + y)) = \tag{37}$$

$$= (a, b) + (c + x, d + y) = \tag{38}$$

$$= (a, b) + ((c, d) + (x, y)) \tag{39}$$

5. + es conmutativa.

$$(a, b) + (c, d) = (a + c, b + d) = (c + a, d + b) = (c, d) + (a, b) \tag{40}$$

+ tiene neutro dado por (a, a) para cualquier $a \in S$.

$$(a, a) + (c, d)s = (a + c, a + d) \sim (c, d), \text{ pues } a + c + d = a + d + c. \tag{41}$$

6. + tiene inversos, $-(a, b) = (b, a)$. $(a, b) + (b, a) = (a + b, a + b)$, éste ultimo es el neutro aditivo.

7. $\cdot$ es asociativo.

$$
\begin{aligned}
&((a,b)\cdot(c,d))\cdot(x,y) = && (42)\\
&\qquad = (ac+bd, bc+ad)\cdot(x,y) && = \quad (43)\\
&\qquad = ((ac+bd)x + (ad+bc)y, (ac+bd)y + (ad+bd)x) && = \quad (44)\\
&\qquad = (acx+bdx+ady+bcy, acy+bdy+adx+bcx) && = \quad (45)\\
&\qquad = (a(cx+dy)+b(cy+dx), b(cx+dy)+a(cy+dx)) && = \quad (46)\\
&\qquad = (a,b)\cdot(cx+dy, cy+dx) && = \quad (47)\\
&\qquad = (a,b)\cdot((c,d)\cdot(x,y)) && (48)\\
& && (49)
\end{aligned}
$$

8. $\cdot$ es conmutativo.

$$
\begin{aligned}
&(a,b)\cdot(c,d) = && (50)\\
&\qquad = (ac+bd, bc+ad) && = \quad (51)\\
&\qquad = (ca+db, da+cb) && = \quad (52)\\
&\qquad = (c,d)\cdot(a,b) && (53)
\end{aligned}
$$

9. $\cdot$ tiene neutro dado por $(1+x, x)$ para cualquier $x \in S$.

$$
\begin{aligned}
&(a,b)\cdot(a+x,x) = && (54)\\
&\qquad = (a(a+x)+bx, ax+b(1+x)) && = \quad (55)\\
&\qquad = (a+(ax+bx), b+(ax+bx)) \sim (a,b) && (56)
\end{aligned}
$$

10. $\cdot$ distribuye a $+$.

$$(a,b)\cdot((c,d)+(x,y)) = \tag{57}$$

$$= (a,b)\cdot(c+x,d+y) = \tag{58}$$

$$= (a(c+x)+b(d+y),a(d+y)+b(c+x)) = \tag{59}$$

$$= ((ac+bd)+(ax+by),(ad+bc)+(ay+bx)) = \tag{60}$$

$$= (ac+bd,bc+ad)+(ax+by,bx+ay) = \tag{61}$$

$$= (a,b)\cdot(c,d)+(a,b)\cdot(x,y) \tag{62}$$

Además $f : S \to A, f(a) = (a+x,x)$ es homomorfismo inyectivo. $f(a+b) = (a+b+x,x) \sim (a+x+b+x,x+x) = (a+x,x)+(b+x,x) = f(a)+f(b)$ y $f(ab) = (ab+x,x) \sim (ab+ax+bx+2x^2+x, ax+bx+2x^2+x) \sim ((a+x)(b+x)+x^2,(a+x)x+(b+x)x) = (a+x,x)\cdot(b+x,x) = f(a)\cdot f(b)$ □

Observación. Sean X,Y conjuntos finitos, entonces $\#(X \sqcup Y) = \#(X)+\#(Y)$ y $\#(X\times Y) = \#(X)\#(Y)$. Lo mismo se cumple para el número de órbitas de $X \sqcup Y$ y $X \times Y$

Lema 2.5. *Sean X,Y G-Conjuntos. X es isomorfo a Y como G-Conjuntos si y solo si tienen el mismo número de órbitas y existe una asignación biyectiva de estas tal que a cada órbita de X le corresponde una órbita de Y y son isomorfas como G-Conjuntos.*

Demostración. Si X y Y son isomorfos como G-Conjuntos con isomorfismo f, entonces tienen el mismo número de órbitas y la asignación de órbitas está dada por f. Al contrario, si tenemos la segunda condición, tenemos $\{G\cdot x_1, G\cdot x_2,\ldots,G\cdot x_n\}$ las órbitas de X y $\{G\cdot y_1, G\cdot y_2,\ldots,G\cdot y_n\}$ las correspondientes órbitas de Y entonces definimos $f(x_i) = y_i \quad \forall 1 \le i \le n$ y extendiéndolo en todo X como $f(g\cdot x) = g\cdot f(x)$, el cual es claramente un isomorfismo de G-Conjuntos □

Lema 2.6. *Las clases de isomorfismo de G-Conjuntos finitos junto con la ó disjunta y el producto cartesiano forman un semianillo conmutativo con 1 denotado por*

$B^+(G)$, donde el neutro de la ó disjunta es $\emptyset$ y el neutro del producto cartesiano es G/G

1. $(X \sqcup Y) \sqcup Z \cong X \sqcup (Y \sqcup Z)$
2. $X \sqcup Y \cong Y \sqcup X$
3. $X \sqcup \emptyset \cong \emptyset \sqcup X \cong X$
4. $X \sqcup Z \cong Y \sqcup Z \quad \Rightarrow \quad X \cong Y$
5. $(X \times Y) \times Z \cong X \times (Y \times Z)$
6. $X \times Y \cong Y \times Z$
7. $X \times G/G \cong G/G \times X \cong X$
8. $X \times (Y \sqcup Z) \cong X \times Y \sqcup X \times Z$

Demostración. Todas la proposiciones son claramente consecuencia del lema y la observación anteriores excepto 4. Tenemos una asignación de órbitas de $X \sqcup Z$ y $Y \sqcup Z$, vamos a dar una para X y Y. Para cada órbita $\theta \subseteq X$ tenemos $f(\theta)$ donde f es el isomorfismo de $X \sqcup Z$ a $Y \sqcup Z$, si $f(\theta) \subseteq Y$ entonces le asignamos esa órbita, si $f(\theta) \subseteq Z$ podemos aplicar f otra vez y repetir el procedimiento hasta obtener una órbita en Y; el proceso es finito pues el número de órbitas de Z es finito. Aplicando el lema anterior tenemos que $X \cong Y$. □

Definición 2.7. *El anillo conmutativo con 1 que resulta de completar $B^+(G)$ denotado por $B(G)$ se llama el* ***anillo de Burnside*** *del grupo G.*

Ejemplo: $G = C_2 = \{1, x | x^2 = 1\}$. $B(G) = \mathbb{Z}G/1 \times \mathbb{Z}G/G$.
Si X es un G-Conjunto, $a \in X$, si a es punto fijo bajo G, $\{a\}$ es órbita y $\{a\} \cong G/G$. Si $b \in X$ no es punto fijo, $\{b, x \cdot b\}$ es órbita y $\{b, x \cdot b\} \cong G/1$. Entonces:

$$X \cong \alpha G/1 + \beta G/G$$

donde α, β son los números de órbitas isomorfas a cada cociente. Además: $(X, Y) \cong X - Y = (\alpha G/1 + \beta G/G) - (\alpha' G/1 + \beta' G/G) = (\alpha - \alpha')G/1 + (\beta - \beta')G/G \qquad \alpha - \alpha', \beta - \beta' \in \mathbb{Z}$

CAPíTULO 3

Tablas de Marcas

1. Homomorfismos de marcas

NOTACIÓN 3.1. *$\mathscr{C}_G$ es el conjunto de las clase de conjugación de los subgrupos de G*

DEFINICIÓN 3.2. *El **anillo fantasma** $\tilde{B}(G) = \prod_{\mathscr{C}_G} \mathbb{Z}$*

PROPOSICIÓN 3.3. *El conjunto $\left\{G/H\right\}_{H\in\mathscr{C}_G}$ es una base de $B(G)$ como grupo abeliano.*

Note que $B(G) \cong \tilde{B}(G)$ como grupos.

DEMOSTRACIÓN. P.D. Los $\left\{G/H\right\}_{H\in\mathscr{C}_G}$ generan a $B(G)$.
Sea $X - Y \in B(G)$ con X, Y G-Conjuntos. Basta generar a X, siendo

$$X = \bigsqcup_{órbitas} G \cdot x_i$$

$$G \cdot x_i \cong G/H_i$$

$$X = \sum G/H_i$$

P.D. los $\left\{G/H\right\}_{H\in\mathscr{C}_G}$ son linealmente independientes sobre $\mathbb{Z}$.
Sea

$$\sum_{H\in\mathscr{C}_G} a_H(G/H) = 0$$

Necesitamos hacer ver que $a_H = 0\ \forall H$. Lo haremos por contradicción. Supongamos que hay algunos $a_H < 0$ y algunos $a_H > 0$. Tomamos todos los $a_H < 0$ y los pasamos del otro lado de la igualdad, tenemos entonces:

$$\sum_{a_H}(G/H) = \sum_{-a_K}(G/K)$$

con $a_H \geq 0$ y $-a_K > 0$.

Sean x, y los G-Conjuntos:

$$X = \bigsqcup_{\forall H_i} \underbrace{((G/H_i) \sqcup \cdots \sqcup (G/H_i))}_{a_{H_i}\text{veces (órbitas de X)}}$$
$$Y = \bigsqcup_{\forall K_j} \underbrace{((G/K_j) \sqcup \cdots \sqcup (G/K_j))}_{-a_{K_j}\text{veces (órbitas de Y)}}$$

Tenemos que $X \cong Y$ como G-Conjuntos, entonces $\sum_i a_{H_i} = \sum_j a_{K_j}$ y $\forall H_i \ \exists K_j$ tal que $G/H_i \cong G/K_j$ lo que implica que H_i, K_j son conjugados. Lo cual es una contradicción. Nuestro error fue pensar que existen a_H distintos de cero, $\therefore a_H = 0$ $\forall H \in \mathscr{C}_G$ □

LEMA 3.4. $(G/1)(G/1) = |G|(G/1)$

DEMOSTRACIÓN.

$$G \times G \cong \bigsqcup_{g \in G} \{(xg, x) | x \in G\}$$

□

PROPOSICIÓN 3.5. *Para $G \neq 1$, el anillo de Burnside y el Anillo Fantasma no son isomorfos como anillos,*

$$B(G) \ncong \tilde{B}(G)$$

DEMOSTRACIÓN. Supongamos que $\exists\ f : B(G) \longrightarrow \tilde{B}(G)$ un isomorfismo de anillos con 1. Veamos en particular,

$$f((G/1)(G/1)) = f((G/1)^2) = ((a_1)^2, \cdots, (a_n)^2)$$

pero además usando el lema 3.4

$$f((G/1)(G/1)) = f(|G|(G/1)) = |G|(a_1, \cdots a_n) = |G| f(G/1)$$

entonces

$$((a_1)^2. \cdots, (a_n)^2) = |G|(a_1, \cdots a_n)$$

$$((a_1)^2. \cdots, (a_n)^2) = (|G|a_1, \cdots |G|a_n)$$

$$\Rightarrow \begin{cases} a_i = 0 & \text{ó} \\ a_i = |G| \end{cases}$$

en cualquier caso $|G|$ divide a a_i, pero no todas las $a_{i's}$ pueden ser cero porque entonces las dimensiones serían distintas.

$$\tilde{B}(G) = \prod_{\mathscr{C}_G} \mathbb{Z}$$

es decir,

$$\tilde{B}(G) = \langle f(G/H_1), f(G/H_2), \cdots, f(G/H_n) \rangle$$

Tenemos entonces que como $(G/H_i)_i$ genera al $B(G)$ como grupo abeliano y como f es un isomorfismo de anillos entre $B(G)$ y $\tilde{B}(G) \Rightarrow f(G/H_i)$ genera a $\tilde{B}(G)$ como grupo abeliano.

Sea M la representación matricial del anillo fantasma $\tilde{B}(G) = \langle f(G/1), \cdots, f(G/G) \rangle$.

$$M = \begin{pmatrix} |G|b_1 & |G|b_2 & \cdots & |G|b_n \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{pmatrix}$$

$$\Rightarrow |G| | \det(M)$$

llevemos M a su forma Normal de Schmidt (M_S), pero M_S no puede ser la identidad porque no concordarían los determinantes

$$M_S \neq \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

$$\frac{\tilde{B}(G)}{\langle f(G/1), \cdots, f(G/G) \rangle} \neq 0$$

$$\Rightarrow \mathrm{Im}(f) \neq \tilde{B}(G)$$

$\therefore f$ no es un isomorfismo y

$$B(G) \ncong \tilde{B}(G)$$

□

PROPOSICIÓN 3.6. *Sea $H \leq G$ y sea*

$$\begin{aligned} \varphi_H : B(G) &\longrightarrow \mathbb{Z} \\ G/K &\longmapsto \#(G/K)^H \end{aligned}$$

Y luego extendemos linealmente

$$\begin{aligned} X &\longmapsto \#(X^H) = \#(\sqcup a_i G/K_i)^H = \sum a_i (G/K_i)^H \\ X - Y &\longmapsto \#(X^H) - \#(Y^H) \end{aligned}$$

Siendo $X^H = \{x \in X | hx = x \forall h \in H\}$. Entonces φ_H esta bien definida y es un homomorfismo suprayectivo de anillos con 1, llamado ***la marca de H***

DEMOSTRACIÓN. Que φ_H está bien definida Si $X - Y \sim X' - Y'$, entonces $X \sqcup Y' \sim X' \sqcup Y'$, aplicamos φ_H, $\#X^H + \#Y'^H = \#X'^H + \#Y^H$, $\#X^H - \#Y^H = \#X'^H - \#Y'^H$. φ_H, preserva la operación de suma(unión) y producto claramente. Además, φ_H es suprayectiva porque $G/G \mapsto 1$ □

TEOREMA 3.7 (Burnside). *Sean X y Y G-Conjuntos. Entonces*

$$X \cong Y \Leftrightarrow \varphi_H(X) = \varphi_H(Y) \ \forall H \leq G$$

DEMOSTRACIÓN. Si $X \cong Y \cong \sum a_i (G/K_i)$, es claro que $\varphi_H(X) = \varphi_H(Y) \ \forall H \leq G$. Por otra parte, si $x \in X, H = Stab_G(x), G \cdot x \cong G/H$, esto quiere decir que $x \in X^H$, además, existe $y \in Y^H$ y que $H = Stab_G(y)$, entonces $G \cdot x \cong G \cdot y \cong G/H$, sean $X' = X \backslash G \cdot x, Y' = Y \backslash G \cdot y$ y las descomposiciones como órbitas de los dos G-Conjuntos son iguales y por lo tanto son isomorfos. □

TEOREMA 3.8. *Sean $H, K \leq G$*

$$\varphi_H = \varphi_K \Leftrightarrow H =_G K$$

DEMOSTRACIÓN. Si $H =_G K$, sea X transitivo. $\varphi_H(X) = \#\{x \in X | h \cdot x = x \ \forall h \in H\} = \#\{x \in X | (g^{-1} K g) \cdot x = x \ \forall k \in K\} = \#\{x \in X | (kg) \cdot x = g \cdot x \ \forall k \in K\} = \#\{y \in X | k \cdot y = y \forall k \in K\} = \varphi_K(X)$, esto último porque X es transitivo y nos dice que las φ's son iguales en los G-Conjuntos base por lo tanto en todos. Si $\varphi_H = \varphi_K$,

entonces, por las fórmulas que veremos en la siguiente sección, donde también se dice que significa α, $|N_G(K)|/|K|\alpha(H,K) = \varphi_H(G/K) = \varphi_K(G/K) = |N_G(K)|/|K|$ por lo tanto hay un conjugado de K que contiene a H, análogamente, hay un conjugado de H que contiene a K, por lo tanto $H =_G K$. □

2. Tabla de marcas

NOTACIÓN 3.9. *Ordenaremos las clases de conjugación de G como $H_1, H_2, \ldots, H_n$ donde $|H_i| \leq |H_{i+1}|$. Esto implica que $H_1 = 1_G$ y $H_n = G$*

PROPOSICIÓN 3.10. $\varphi_H(G/K) = \frac{|N_G(H)|}{|K|}\beta(H,K) = \frac{|N_G(K)|}{|K|}\alpha(H,K)$, *donde* $\alpha(H,K) = \#E \leq G | E =_G K, H \leq E$, $\beta(H,K) = E \leq G | E =_G H, E \leq K$

DEMOSTRACIÓN.

$$\varphi_H(G/K) = (G/K)^H = \#\{gK | g^{-1}hgK = K \ \forall h \in H\} =$$

$$= \frac{\#\{g \in G | hgK = gK \ \forall h \in H\}}{|K|} = \frac{\#\{g \in G | g^{-1}hgK = K \ \forall h \in H\}}{|K|} =$$

$$= \frac{\#\{g \in G | g^{-1}hg \in K \ \forall h \in H\}}{|K|} = \frac{\#\{g \in G | g^{-1}Hg \subseteq K\}}{|K|} =$$

$$= \frac{|N_G(H)|\#\{E \leq G | E \text{ es conjugado a } H, E \leq K\}}{|K|} = \frac{N_G(H)}{|K|}\beta(H,K)$$

la otra igualdad la tenemos de

$$\varphi_H(G/K) = \frac{\#\{g \in G | g^{-1}Hg \subseteq K\}}{|K|} = \frac{\#\{g \in G | H \subseteq gKg^{-1}\}}{|K|} =$$

$$\frac{N_G(K)\#\{E \leq G | E \text{ es conjugado a } K, H \leq E\}}{|K|} = \frac{N_G(K)}{|K|}\alpha(H,K)$$

□

DEFINICIÓN 3.11. *La **tabla de marcas** del grupo G es la matriz $(a_{i,j}) = \varphi_{H_i}(G/H_j) = \frac{|N_G(H_i)|}{|H_j|}\beta(H_i,H_j) = \frac{|N_G(H_j)|}{|H_j|}\alpha(H_i,H_j)$*

PROPOSICIÓN 3.12. *La tabla de marcas $A = (a_{i,j})$del grupo G satisface:*

1. *A es cuadrada de tamaño $n \times n$ donde $n = \#\mathscr{C}_G$*

2. *A tiene coordenadas enteras*
3. *A es triangular superior*
4. $a_{i,n} = 1 \quad \forall 1 \leq i \leq n$, *es decir, la última columna es la únidad de* $\prod_{\mathscr{C}_G} \mathbb{Z}$
5. *La primera fila de G corresponde a los índices de los subgrupos de G, esto es:* $a_{1,i} = |G/H_i| \quad \forall 1 \leq i \leq n$, *como casos particulares* $a_{1,1} = |G|, a_{1,n} = 1$
6. $a_{i,i} = |\frac{N_G(H_i)}{H_i}| \quad \forall 1 \leq i \leq n$ *los órdenes de los subgrupos de Weyl.*
7. *Un elemento en la diagonal divide a cada elemento de la misma columna que él, esto es:* $\varphi_{H_i}(G/H_j) \big| \varphi_{H_j}(G/H_i)$
8. $a_{1,1} = |G|$ *es la mayor entrada de A*
9. $|H_i| = \frac{a_{1,1}}{a_{1,i}}$
10. $|N_G(H_i)| = \frac{a_{1,1}}{a_{1,i}/a_{i,i}}$
11. $\alpha(H_i, H_j) = \frac{a_{i,j}}{a_{j,j}}$
12. $\beta(H_i, H_j) = \frac{a_{i,j}a_{1,i}}{a_{i,i}a_{1,j}}$

DEMOSTRACIÓN. 1. Por construcción A es cuadrada y $n = \#\mathscr{C}_G$

2. Basta recordar que $\varphi_{H_i}(G/H_j) = \#(G/H_j)^{H_i}$ son los puntos fijos de H_j bajo H_i lo cual es entero.

3. $a_{i,j} = 0$ si $i > j$ pues si $|H_i| > |H_j|$ entonces $\alpha(H_i, H_j) = 0$, y si $|H_i| = |H_j|$ corresponden a clases de conjugación diferentes y $\alpha(H_i, H_j) = 0$, por lo tanto $\varphi_{H_i}(G/H_j) = 0 = a_{i,j}$

4. $N_G(G) = G, \alpha(H_i, G) = 1$ esto último es porque G no tiene conjugados, entonces $\frac{|N_G(H_n)|}{|H_n|}\alpha(H_i, H_n) = \frac{|G|}{|G|} = 1$

5. $N_G(1_G) = G, \beta(1_G, H_i) = 1$ pues 1_G no tiene conjugados, entonces $\frac{|N_G(H_1)|}{|H_i|}\beta(H_1, H_i) = \frac{|G|}{|H_i|} = |\frac{G}{H_i}|$

6. $\alpha(H_i, H_i) = 1$ y $a_{i,i} = \frac{|N_G(H_i)|}{|H_i|} = |\frac{N_G(H_i)}{H_i}|$

7. $a_{j,i} = \frac{|N_G(H_i)|}{|H_i|}\alpha(H_j, H_i) = a_{i,i}\alpha(H_j, H_i)$ y $a_{i,i} \big| a_{j,i}$

8. $a_{i,j} = \varphi_{H_i}(G/H_j)$ son los puntos fijos de G/H_j bajo la acción de H_i entonces $a_{i,j} \leq |G/H_j| \leq |G|$, donde la última igualdad se da únicamente cuando $j = 1, H_j = 1_G$. Así $a_{1,1} > a_{i,j}$ si $i \neq 1$ o $j \neq 1$

9. $\frac{a_{1,1}}{a_{1,i}} = \frac{|G|}{|G/H_i|} = |H_i|$
10. $\frac{a_{1,1}}{a_{1,i}/a_{i,i}} = |G|/(\frac{|G|/|H_i|}{|N_G(H_i)|/|H_i|}) = |G|/|G/N_G(H_i)| = N_G(H_i)$
11. $\frac{a_{i,j}}{a_{j,j}} = \frac{N_G(H_j)}{H_j}\alpha(H_i,H_j)/\frac{N_G(H_j)}{H_j} = \alpha(H_i,H_j)$
12. $\frac{a_{i,j}a_{1,i}}{a_{i,i}a_{1,j}} = \frac{\frac{|N_G(H_i)|}{|H_j|}\beta(H_i,H_j)\frac{|G|}{|H_i|}}{\frac{|N_G(H_i)|}{|H_i|}\frac{|G|}{|H_j|}} = \beta(H_i,H_j)$

□

OBSERVACIÓN. La definición de GAP[**3**] para la tabla de marcas de G es A^t

Ejemplos de tablas de marcas:

1. Trivial: $\varphi(1/1) = 1$. Tabla de Marcas del Trivial: (1)
2. $G = C_p$, Cíclico de orden primo p. Tiene dos subgrupos $\{1, C_p\}$. De modo que la tabla de marcas de G_p queda:

$$\begin{pmatrix} |G/1| & 1 \\ 0 & |G/C_p| \end{pmatrix} = \begin{pmatrix} p & 1 \\ 0 & 1 \end{pmatrix}$$

   Observación: Si G es un grupo finito y la tabla de marcas es de dimensión 2 entonces G es isomorfo a C_p.
3. $G = C_4$. Tiene tres subgrupos: $\{1, C_2, C_4\}$. Y su tabla de marcas es:

$$\begin{pmatrix} 4 & 2 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

4. $G = S_3$. Tiene 4 subgrupos: $\{1, \langle(1,2)\rangle, \langle(1,2,3)\rangle, S_3\}$. Y su tabla de marcas es:

$$\begin{pmatrix} 6 & 3 & 2 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

5. C_6. Tiene 4 subgrupos: $\{1, C_2, C_3, C_6\}$. Y su tabla de marcas es:

$$\begin{pmatrix} 6 & 3 & 2 & 1 \\ 0 & 3 & 0 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

6. $C_2 \times C_2$. Tiene 5 subgrupos $\{1, H_1, H_2, H_3, C_2 \times C_2\}$, siendo: $H_1 = \langle (1,0) \rangle$, $H_2 = \langle (0,1) \rangle$, $H_3 = \langle (1,1) \rangle$, su tabla de marcas es:

$$\begin{pmatrix} 4 & 2 & 2 & 2 & 1 \\ 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Observación. No toda matriz triangular que cumpla las propiedades listadas anteriormente es la tabla de marcas de un grupo.

$$\begin{pmatrix} 4 & 1 \\ 0 & 1 \end{pmatrix}$$

es un contraejemplo. La primera entrada dice que el orden del grupo, si existiera, sería cuatro y como la matriz es de 2×2 el grupo sería simple, los únicos dos grupos de orden cuatro no son simples.

Consideremos la función $\varphi : B(G) \to \tilde{B}(g)$ dado por $\varphi(G/K) = (\varphi_{H_i}(G/K))_{H_i}$ que es homomorfismo porque lo es por entradas y es inyectivo porque la tabla de marcas es triangular superior y las columnas son linealmente independientes, así, $B(G)$ es isomorfo a su imagen, que es lo generado por las columnas de la tabla de marcas. Entonces el producto de elementos en $B(G)$ se da mediante los generadores denotados por G/K. El siguiente resultado da una forma de muliplicarlos en terminos de grupos.

Sabemos por construcción que $B(G) \subseteq \tilde{B}(G)$

Proposición 3.13. *$|G|$ es el menor número entero n tal que $n\tilde{B}(G) \subseteq B(G)$*

Demostración.

$$n\tilde{B}(G) \subseteq B(G) \Leftrightarrow n(\tilde{B}(G)/B(G)) = 0$$

$$\tilde{B}(G) \subseteq B(G) = \oplus\mathbb{Z}_{\mathscr{C}_G}/(\text{cols de ToM}) \cong \mathbb{Z}/|G|\mathbb{Z} \oplus \cdots \oplus \mathbb{Z}/W_H\mathbb{Z} \oplus \cdots \oplus \mathbb{Z}/1\mathbb{Z}$$

Donde $W_H = |N_G(H)|/|H|$ y

$$\begin{pmatrix} |G| & & & & \\ & \ddots & & & \\ & & W_H & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}$$

Es la forma normal de Schmidt de la tabla de marcas. Como $W_H \mid |N_G(H)| \mid |G|$ entonces

$$|G|(\mathbb{Z}/|G|\mathbb{Z} \oplus \cdots \oplus \mathbb{Z}/W_H\mathbb{Z} \oplus \cdots \oplus \mathbb{Z}/1\mathbb{Z}) = 0$$

y $|G|$ es el menor entero que satisface la ecuación. □

3. Propiedades Conocidas

Definición 3.14. *Sean G,Q grupos finitos. Sea $\psi : \mathscr{C}_G \to \mathscr{C}_Q$ una función de la familia de clases de conjugación de G a la de Q. Dado $H \leq G$, denotamos por H' cualquier representante de $\psi([H])$. ψ es un* ***isomorfismo de marcas*** *o* ***isomorfismo entre las marcas de*** *G* ***y*** *Q si ψ es biyección y $\#(K'^{H'}) = \#(K^H)$, es decir, preserva los puntos fijos de G/K bajo la acción de H, $\varphi_H(G/K) = \varphi_{H'}(Q/K')$.*

Teorema 3.15. *Sean G, Q grupos con tablas de marcas isomorfas. Sean $H, K \leq G$, H', K' representantes en la respectiva clase de conjugación bajo ψ, el isomorfismo de marcas. Entonces:*

1. $|G| = |Q|$
2. $(1_G)' = 1_Q$
3. $G' = Q$
4. $|H| = |H'|$
5. $|N_G(H)| = |N_{G'}(H')|$

6. $\alpha(H,K) = \alpha(H',K')$
7. $\beta(H,K) = \beta(H',K')$

DEMOSTRACIÓN. 1. $|G| = |Q|$ ya que cada uno es el mayor elemento en su respectiva tabla de marcas.

2. $\#(1_G)^{\prime(1_G)'} = \#1_G^{1_G} = \varphi_{1_G}(G/1_G) = |G| = |Q| = \varphi_{1_Q}(Q/1_Q) = \#1_Q^{1_Q}$. Así, $(1_G)' = 1_Q$.
3. $|Q/G'| = \frac{|Q|}{|G'|} = 1$, entonces $G' = Q$ y denotaremos así a Q.
4. $\frac{|G|}{|H|} = |G/H| = \#H^{1_G} = \#H'^{1_{G'}} = (G'/H') = \frac{|G'|}{|H'|}$. Sabiendo que $|G| = |G'|$, entonces $|H| = |H'|$
5. $\frac{|N_G(H)|}{|H|} = \#H^H = \#H'^{H'} = \frac{|N_{G'}(H')|}{|H'|}$. Como $|H| = |H'|$ el resultado queda demostrado
6. $\frac{|N_G(K)|}{|K|}\alpha(H,K) = \#K^H = \#K'^{H'} = \frac{|N_{G'}(K')|}{|K'|}\alpha(H',K')$. Por los resultados anteriores, $\alpha(H,K) = \alpha(H',K')$.
7. $\frac{|N_G(H)|}{|K|}\beta(H,K) = \#K^H = \#K'^{H'} = \frac{|N_{G'}(H')|}{|K'|}\beta(H',K')$ y $\beta(H,K) = \beta(H',K')$.

□

NOTACIÓN 3.16. *G' es un grupo con tabla de marcas isomorfas a la de G. H' denota a un representante en $\psi([H])$ si $H \leq G$*

Sin embargo dos grupos no isomorfos pueden tener tablas de marcas isomorfas.

Sean p, q primos tales que

1. $q|p-1$
2. $a, b \in \mathbb{Z}_p^*$ monoide con el producto tales que

$$o(a) = o(b) = q, a \neq b$$

3. Sea $Q =< z >$ donde $o(z) = q$
4. Sean $P_a =< x >, P_b =< y >$ donde

$$o(x) = o(y) = p$$

Sean $G_{(a,b)} = (P_a \times P_b) \rtimes Q$ donde el producto semidirecto está dado por:

$$zxz^{-1} = x^a, zyz^{-1} = y^b$$

PROPOSICIÓN 3.17.

$$G_{(a,b)} \cong G_{(a',b')} \Leftrightarrow \exists k, q \nmid k$$

tal que

$$\{a', b'\} = \{a^k, b^k\}$$

DEMOSTRACIÓN. La demostración puede ser encontrada en [**1**] □

Así cuando $q = 5, p = 11, a = 3, b = 4, a' = 3, b' = 5$ entonces $G_{(a,b)} \ncong G_{(a',b')}$ pero tienen tablas de marcas isomorfas.

PROPOSICIÓN 3.18. *Si A, A' son las tablas de marcas de G, G', entonces son isomorfas si y solo si existe una matriz de permutación P tal que $A' = PAP^{-1}$.*

DEMOSTRACIÓN. Las tablas de marcas son isomorfas si se pueden permutar los $H_i's$, esto es lo mismo que intercambiar renglones y luego las respectivas columnas que es igual que a que exista la matriz P que satisfaga la propiedad requerida. □

LEMA 3.19. *H es un subgrupo maximal de G si y solo si H' es un subgrupo maximal en G'*

DEMOSTRACIÓN. Sea H maximal en G, supongamos que H' no lo es en G' entonces $\exists K' \leq G$ tal que $H' \leq K'$, como φ es biyección, existe una clase de conjugación $[K] \in \mathscr{C}_G$ que cumple $K' \in \phi([K])$. Además, $0 \neq \alpha(H', K') = \alpha(H, K)$, un conjugado de K contiene a H. Como H es maximal, $K = H$ entonces $K' = H'$, o bien $K = G$ y $K' = G'$ demostrando que H' es maximal.
Si H' es maximal en G' la demostración es análoga. □

PROPOSICIÓN 3.20. *H es normal en G si y solo si $\#(H^H) = \#(H^{1_G})$*

DEMOSTRACIÓN. Recordemos que $\#(H^H) = \frac{|N_G(H)|}{|H|}$ y $\#(H^{1_G}) = \frac{|G|}{|H|}$.
Si $H \trianglelefteq G$ entonces $|N_G(H)| = |G|$, $\frac{|N_G(H)|}{|H|} = \frac{|G|}{|H|}$ y $\#(H^H) = \#(H^{1_G})$.
Si $\#(H^H) = \#(H^{1_G})$, $\frac{|N_G(H)|}{|H|} = \frac{|G|}{|H|}$, $|N_G(H)| = |G|$ y H es normal en G. □

LEMA 3.21. *H es normal en G si y solo si H' es normal en G'*

DEMOSTRACIÓN. $H \trianglelefteq G$, entonces $\#(H'^{H'}) = \#(H^H) = \#(H^{1_G}) = \#(H'^{1_{G'}})$ y $H' \trianglelefteq G'$ □

LEMA 3.22. *Si $H \trianglelefteq G$ entonces G/H y G'/H' tienen tablas de marcas isomorfas.*

DEMOSTRACIÓN. Sea $K_H \leq G/H$, por el teorema de la correspondencia tenemos que $K_H \cong G/K$ para algún K, $H \leq K \trianglelefteq G$ entonces $H' \leq K' \trianglelefteq G'$ y $\psi(K_H) = K'_{H'} \cong G'/K'$ y ψ es biyección. Además el teorema de la correspondencia implica que:

$$|K_H| = |K_{H'}|,\ |N_{G/H}(K_H)| = |N_{G'/H'}(K'_{H'})|,$$

$$\alpha(K_H, L_H) = \alpha(K'_{H'}, L'_{H'}),\ \beta(K_H, L_H) = \beta(K'_{H'}, L'_{H'})$$

Recordando que $\#(K_H^{L_H}) = \frac{|N_{G/H}(K_H)|}{|K_H|}\alpha(L_H, K_H)$, ψ lo preserva y es isomorfismo de marcas entre G/H y G'/H' □

LEMA 3.23. *Si $K \leq H$ y uno de ellos es normal en G, entonces $K' \leq H'$ para cualquier elección de representantes en H', K'.*

DEMOSTRACIÓN. Si $H \trianglelefteq G$ entonces $\alpha(K, H) = 1$ porque H no tiene conjugados, $H' \trianglelefteq G'$ implica que solo hay una elección de H', para cualquier elección de K', $\alpha(K', H') = 1$ y $K' \leq H'$.
Si $K \trianglelefteq G$, $\beta(K, H) = 1$ y la demostración es análoga. □

PROPOSICIÓN 3.24. *H es cíclico si y solo si H' lo es.*

DEMOSTRACIÓN. Hay que recordar que un grupo cíclico H está caracterizado porque para cada $d||H|$ existe un único $K \leq H$, $|K| = d$, es decir, hay un único K que cumple $|K| = d$ y $\beta(K, H) = 1$ y estas propiedades se preservan bajo el isomorfismo. □

LEMA 3.25. *Si $K \trianglelefteq G, H \trianglelefteq G$ entonces $(K \cap H)' = K' \cap H'$ y $(KH)' = K'H'$.*

DEMOSTRACIÓN. $L = K \cap H$ es el grupo más grande tal que $L \leq H$, $L \leq K$, como H y K son normales, $L' \leq H'$, $L' \leq K'$, entonces $L' \leq H' \cap K'$. Análogamente $K' \cap H' \leq L'$ y $(K \cap H)' = K' \cap H'$.

$M = KH$ es subgrupo de G porque H es normal y es el menor grupo tal que $K \leq M,\ H \leq M$. Como H y K son normales en G, $K' \leq M',\ H' \leq M'$ y $K'H' \leq M'$. Análogamente $M' \leq K'H'$ y $(KH)' = K'H'$. □

COROLARIO. Dos subgrupos normales con intersección trivial se corresponden con dos subgrupos normales con intersección trivial. Es decir, $K \cap H = 1_G$ si y solo si $K' \cap H' = 1_{G'}$. Además si $G = H \times K$ entonces $G' = H' \times K'$, K y K' tienen tablas de marcas isomorfas y H y H' tienen tablas de marcas isomorfas.

DEMOSTRACIÓN. La primera parte es consecuencia del resultado anterior.
Además, $G = H \times K \ \Leftrightarrow\ G = HK,\ H \cap K = 1_G$, propiedades que se preservan bajo el isomorfismo de marcas.
$H \cong G/K,\ H' \cong G'/K'$ tienen tablas de marcas isomorfas. $K \cong G/H,\ K' \cong G'/H'$ tienen tablas de marcas isomorfas. □

PROPOSICIÓN 3.26. *Los subgrupos de Frattini se corresponden, es decir* $(\Phi(G))' = \Phi(G')$

DEMOSTRACIÓN. Hay que recordar que $\Phi(G)$ es la intersección de todos los grupos maximales de G y $\Phi(G) \trianglelefteq G$.
Sea $X = \Phi(G)$, como todos los subgrupos maximales se corresponden y X' es normal en G entonces X' está contenido en todos los subgrupos maximales de G', como consecuencia $X' \leq \Phi(G')$. Simétricamente, se demuestra el análogo y $(\Phi(G))' = \Phi(G')$. □

PROPOSICIÓN 3.27. *Si G es p-grupo entonces $(S(Z(G)))' = S(Z(G'))$. Es decir, los soclos de los centros de G y G' se corresponden.*

DEMOSTRACIÓN. El soclo del centro de un p-grupo está caracterizado como el grupo normal más pequeño de G que tiene intersección no trivial con todos los subgrupos normales de G. Estas propiedades se preservan bajo el isomorfismo. □

PROPOSICIÓN 3.28. *Si G es abeliano, entonces $G \cong G'$. Más aun, si G es Hamiloniano, entonces $G \cong G'$*

DEMOSTRACIÓN. La demostración está en [**4**] □

PROPOSICIÓN 3.29. *Los subgrupos conmutadores se corresponden. Es decir* $[G,G]' = [G',G']$.

DEMOSTRACIÓN. El subgrupo conmutador de G es el grupo normal más pequeño con cociente abeliano. Estas propiedades son preservadas por ψ □

COROLARIO. Los grupos abelianizados son isomorfos i. e. $(G/[G,G])' \cong G'/[G',G']$

DEMOSTRACIÓN. Como$[G,G]$ y $[G',G']$ se corresponden y son normales entonces $G/[G,G]$ y $G'/[G',G']$ tienen tablas de marcas isomorfas y como son abelianos, entonces son isomorfos. □

PROPOSICIÓN 3.30. *Si S es un p-subgrupo de Sylow de G entonces S' es un p-subgrupo de Sylow de G'*

DEMOSTRACIÓN. Sólo hay que notar que los órdenes de los grupos se corresponden. □

PROPOSICIÓN 3.31. *Si G es nilpotente, entonces G' es nilpotente.*

DEMOSTRACIÓN. Un grupo nilpotente está caracterizado porque todos sus subgrupos de Sylow son normales. ψ preserva esta propiedad. □

PROPOSICIÓN 3.32. *Si G es un p-grupo con un subgrupo cíclico de índice p, entonces $G \cong G'$*

DEMOSTRACIÓN. Los p-grupos con un subgrupo cíclico de índice p están clasificados. Si $p \neq 2$, los grupos(de orden p^n) son: $C_{p^n}, C_{p^{n-1}} \times C_p, C_{p^{n-1}} \rtimes C_p$, los dos primeros son abelianos, que ya se vió que el correspondiente será isomorfo y el tercero, por eliminación también. Si $p = 2$ la clasificación se puede encontrar en [**2**] □

LEMA 3.33. *Sea $H \leq G$, $d_H||H|$. Sea n_{d_H} el número de elementos de H con orden d_H. Entonces $n_{d_H} = n_{d_{H'}}$.*

DEMOSTRACIÓN. Sean $K_1, K_2, K_3, \cdots, K_l$ las clases de conjugación de G isomorfas a C_{d_H} el grupo cíclico de orden d_H. Entonces $n_{d_H} = \sum_{i=1}^{l} \beta(K_i, H)\varphi(d_H)$. Donde φ es la función phi de Euler y $\beta(K_i, H)$ puede ser cero para algunas $i's$. Como φ solo depende de d_H, $n_{d_H} = n_{d_{H'}}$. □

NOTACIÓN 3.34.

$$C_p^n = \underbrace{C_p \times C_p \times \cdots \times C_p}_{n\text{veces}}$$

Es el grupo elemental abeliano de orden p^n.

LEMA 3.35. *Sea* p *primo,* $a, b \in G$ *de orden* p, A *el subgrupo de* G *generado por* a *y* b. *Si* A *tiene orden* p^2, *entonces* S *es un grupo elemental abeliano y* $\#\{(c,d) \in G \times G| < c,d >= A\} = \#\{(e,f) \in C_p^2 \times C_p^2|| < e,f > | = p^2\}$.

DEMOSTRACIÓN. El grupo A es abeliano porque tiene orden p^2. Como A está generado por dos elementos de orden p que conmutan, no puede ser cíclico. Además, $\#\{(c,d) \in G \times G| < c,d >= A\} = \#\{(x,y) \in A \times A| < x,y >= A\} = \#\{(z,w) \in C_p^2 \times C_p^2| < z,w >= C_p^2\}$ □

PROPOSICIÓN 3.36. *Sea* p *número primo,* P, P' *grupos tales que* $|P| = |P'| = p^n$ *con* $n > 1$. *Supongamos que* $P \cong C_p^n$, *que todos los elementos no triviales de* P' *tienen orden* p *y que* P' *no es abeliano. Sea* μ*(respectivamente* μ'*) el número de subgrupos de* $P(P')$ *de orden* p^2. *Entonces* $\mu' < \mu$.

DEMOSTRACIÓN. Sea $\Lambda = \{(a,b) \in P \times P|| < a,b > | = p^2\}$, $\Lambda' = \{(a',b') \in P' \times P'|| < a',b' > | = p^2\}$. Definimos relaciones de equivalencia $\sim$,$\sim'$ en Λ, Λ' respectivamente por: $(a,b) \sim (c,d)$ sii $< a,b >=< c,d >$ y $(a',b') \sim' (c',d')$ sii $< a',b' >=< c',d' >$. Por el lema anterior, cada clase de equivalencia en Λ tiene la misma cardinalidad que cada clase de equivalencia que Λ', denotemos por τ a esa cardinalidad. Tenemos que $\mu = \#(\Lambda/\sim) = \#\Lambda/\tau$ y $\mu' = \#(\Lambda'/\sim') = \#\Lambda'/\tau$. Debemos mostrar que $\#\Lambda' < \#\Lambda$. Sea $\Gamma' = \{(a',b') \in P' \times P'|a' \neq 0, b' \notin < a' >\}$. Vamos a mostrar que Λ' es un subconjunto propio de Γ' y que Λ y Γ' tienen la misma cardinalidad.

Sea $(a',b') \in \Lambda'$. Como $| < a,b > | = p^2$, entonces $a' \neq 0$ (porque $| < b' > | = p$) y $b' \notin < a' >$ (porque $| < a' > | = p$), entonces $(a',b') \in \Gamma'$, así $\Lambda' \subseteq \Gamma'$. Por otra parte, como P' no es abeliano, existen $a', b' \in P'$ que no conmutan, entonces $| < a',b' > | > p^2, a' \neq 0$ y $b' \notin < a' >$, entonces (a',b') está en Γ' pero no en Λ' y $\Lambda' \subset \Gamma$

Por último, los elementos de ambos conjuntos Λ y Γ' se pueden contar como sigue: hay p^n-1 formas de escoger el primer elemento y p^n-p formas de escoger el segundo, cada conjunto tiene $(p^n-1)(p^n-p)$ elementos. □

TEOREMA 3.37. *Si H es un subgrupo elemental abeliano de G entonces H' es un subgrupo elemental abeliano de G'.*

DEMOSTRACIÓN. Si $|H| = p^n$ entonces el número de elementos no triviales de orden p es $p^n - 1$ que es el mismo para H' porque ese número se preserva. Si H' no fuera abeliano, por el lema anterior tendría menos subgrupos de orden p^2 que H, pero ese numero también se preserva. □

LEMA 3.38. *Sea n un entero positivo mayor o igual que 5. Sea G un grupo de orden $n!$ tal que G tiene un único subgrupo propio normal H. Supongamos que H tiene índice 2 en G y que hay un subgrupo T de G de índice n. Entonces $G \cong S_n$, donde S_n es el grupo de permutaciones de n elementos.*

DEMOSTRACIÓN. Consideremos la acción de G en las clases laterales izquierdas gT de T en G. Esto nos da un morfismo $\phi : G \to S_n$. Como G actúa transitivamente en G/T el núcleo de ϕ no puede ser G o H. Como ϕ es inyectivo y los ordenes concuerdan, es isomorfismo. □

TEOREMA 3.39. *Sea G un grupo con tabla de marcas isomorfa a la de S_n para algún $n \geq 5$ entonces $G \cong S_n$.*

DEMOSTRACIÓN. Se satisfacen las condiciones del lema anterior. □

CAPíTULO 4

Algoritmos de Minimización

0.1. Caracteristicas en GAP de una tabla de marcas. Las Tablas de Marcas son:

1. Matrices cuadradas y enteras.
2. Matrices triangulares inferiores, pues GAP usa la transpuesta.
3. La entrada (1,1), que es la esquina superior izquierda, es el orden del Grupo y es la entrada mas grande.
4. Todas las entradas $(i, 1)$, es decir, las entradas debajo del orden del grupo, dividen al orden del grupo.
5. Toda la fila inferior esta llena de 1's.

Una Tabla de Marcas típica en GAP:

```
gap> Display(TableOfMarks("S4"));
1:  24
2:  12 4
3:  12 . 2
4:   8 . . 2
5:   6 6 . . 6
6:   6 2 . . . 2
7:   6 2 2 . . . 2
8:   4 . 2 1 . . . 1
9:   3 3 1 . 3 1 1 . 1
10:   2 2 . 2 2 . . . . 2
11:   1 1 1 1 1 1 1 1 1 1 1

gap>
```

0.2. Nociones comunes de los algoritmos de Canonización.

0.2.1. Empates. Los empates se definen en cada algoritmo de minimización, el primer algoritmo toma como empates a entradas iguales en la diagonal, en el ejemplo del grupo $S4$, se ve que los empates son en las filas $[3, 4, 6, 7, 10]$ pues todas ellas tienen en la diagonal un 2, de la misma manera están empatadas las filas $[8, 9, 11]$, sin embargo es imposible intercambiar la fila 11 con ninguna otra, pues de hacerlo se perdería la forma trangular inferior.

La gran mayoria de las matrices de Tablas de Marcas tienen empates y por eso se tratará de encontrar una forma canonizada, al canonizar se intenta de dar cierto peso a algunas filas empatadas y de esa manera desempatarlas, siempre manteniendo la forma triangular inferior.

0.3. Restricciones de GAP. GAP no tiene apuntadores ni memoria dinámica, GAP mismo se encarga de crear variables cuando sea necesario, segun se vayan usando, sólo las variables locales deben declararse al principio de la función.

GAP no tiene una forma explícita de pasar argumentos "Por Referencia" y "Por Valor" como lo tiene C por ejemplo, y por lo tanto es mejor tratar con variables globales para evitar desperdiciar memoria debido a las enormes Tablas de Marcas de cada grupo.

1. Algoritmo 0

1.0.1. Los empates. En esta forma de canonizacion se se consideran empates a las columnas que tengan las mismas entradas en la diagonal.

```
gap> Display(TableOfMarks("S4"));
1:  24
2:  12 4
3:  12 . 2
4:   8 . . 2
5:   6 6 . . 6
6:   6 2 . . . 2
7:   6 2 2 . . . 2
```

```
8:   4 . 2 1 . . . 1
9:   3 3 1 . 3 1 1 . 1
10:   2 2 . 2 2 . . . . 2
11:   1 1 1 1 1 1 1 1 1 1 1

gap>
```

Estan empatadas las filas $[3, 4, 6, 7, 10]$ y las filas $[8, 9]$, la ultima fila no se considera pues es imposible cambiarla con ninguna otra, y por la misma razón tampoco se considera la primer fila.

A pesar de definir empates, en el Algoritmo 0, no son tan importantes, ya que las entradas de la diagonal estan en desorden y serán reordenadas, al hacer la reodenación se toman todas las filas y no solo los empates.

1.0.2. Algoritmo esqueleto. El siguiente es el pseudocódigo esqueleto, que es el algoritmo principal, pero no el mas importante, el mas importante es el que canoniza las matrices.

- Para un orden n
  - Encontrar grupos no abelianos de orden n
    - Hacer una lista L de matrices de mismas dimensiones de dicho orden
    - Canonizar las matrices de la lista L
    - Comparar las matrices del mismo orden (**importante**)
      - Si 2 matrices son distintas (no isomorfas) y su forma canonica de smith es identica $\Rightarrow$ ¡Posible contraejemplo!

1.0.3. Algoritmo de canonización. Este es el algoritmo interno mas importante, parte del Algoritmo 0. Sirve para canonizar una matriz.

- Se toma la matriz de tabla de marcas mtom.
- (*) $mtom := stom$ (matriz de tabla de marcas y una copia para trabajar en ella)
  - Analizar posibles cambios (permutaciones de filas y columnas)
    - Comparaciones a hacer

1. Verificar cuales filas y columnas son intercambiables
2. Comparar valor entero de las diagonales permutables
3. Comparar valor de las entradas de las filas y columnas permutables
4. Comparar la cantidad de ceros en las filas y columnas permutables
5. Comparacion adicional de $<$ de GAP como último recurso

- Aplicar los cambios que surgieron a la matriz *stom*

- Si $mtom = stom$ el agoritmo para y devuelve *stom*. Si no, ir a (*)

1.0.4. Encontrando anillos de Burnside isomorfos en el Algoritmo 0. El algoritmo 0, usa las matrices canonizadas y luego les encuentra su Forma Normal de Smith, si las formas normales de Smith son iguales es un indicador de que los Anillos de Burnside pueden ser iguales.

1.1. Salida del algoritmo. Salida del algoritmo para los grupos de orden 12 hasta el 15

```
gap> Read("algoritmo05.g");
Orden 12 hay 5
        MatTom: SmallGroup (12,1) ToM nxn, n = 6
        MatTom: SmallGroup (12,2) es abeliano
        MatTom: SmallGroup (12,3) ToM nxn, n = 5
        MatTom: SmallGroup (12,4) ToM nxn, n = 10
        MatTom: SmallGroup (12,5) es abeliano
         -> Buscando matrices (no abelianas) de las mismas dimensiones++
        [ [ 1, 2 ], [ 4, 5 ] ]
        Checando:[ 1, 2 ]
                Canonizando Tom's
                 (12, 1)**
                 (12, 2)**
                Buscando Colisiones Smith y Tom en: [ 1, 2 ]
1:2
```

```
          [ 1, 1, 2, 2, 12, 12 ]
          [ 1, 1, 2, 6, 12, 12 ]
        Checando:[ 4, 5 ]
                Canonizando Tom's
                 (12, 4)**
                 (12, 5)***
                Buscando Colisiones Smith y Tom en: [ 4, 5 ]
1:2
          [ 1, 1, 2, 2, 2, 2, 2, 2, 12, 12 ]
          [ 1, 1, 2, 2, 2, 6, 6, 6, 12, 12 ]
Orden 13 hay 1
Orden 14 hay 2
        MatTom: SmallGroup (14,1) ToM nxn, n = 4
        MatTom: SmallGroup (14,2) es abeliano
         -> Buscando matrices (no abelianas) de las mismas dimensiones+
        [ [ 1, 2 ] ]
        Checando:[ 1, 2 ]
                Canonizando Tom's
                 (14, 1)*
                 (14, 2)**
                Buscando Colisiones Smith y Tom en: [ 1, 2 ]
1:2
          [ 1, 1, 2, 14 ]
          [ 1, 1, 14, 14 ]
Orden 15 hay 1
```

1.1.1. Resultados. A pesar de la fé, lo triste es que el Algoritmo 0 resultó ser muy pesado, lento y tosco. Es dificil de ver la manera en que se podría justificar matemáticamente. Pues para el desempate furiosamente se aplicaron comparaciones de muchas cosas, la mas loca de ellas es la comparacion de ceros, que resulta ser muy rara. Veamos como.s

Tenemos por ejemplo la lista de GAP $[8,4,2,0,0,0,1]$ que representa a una fila de una tabla de marcas, sin contar las entradas que estan despues de la diagonal. Vamos a compararle los ceros, el procedimiento es como sigue:

1. Cambiar los digitos no cero por ceros y los ceros por 1's,

$$[8,4,2,0,0,0,1] \rightarrow [0,0,0,1,1,1,0]$$

2. Concatenar los digitos y tomarlos como un numero binario,

$$[0,0,0,1,1,1,0] \rightarrow 1110_2$$

El resultado, 110_2 es un valor que puede compararse con otros valores resultado de otras filas y columnas. Para la siguiente matriz, el valor de ceros de la fila 6 es 11_2 y el de la fila 9 es 1000_2.

```
gap> Display(TableOfMarks("S4"));
1:  24
2:  12 4
3:  12 . 2
4:   8 . . 2
5:   6 6 . . 6
6:   6 2 . . . 2
7:   6 2 2 . . . 2
8:   4 . 2 1 . . . 1
9:   3 3 1 . 3 1 1 . 1
10:   2 2 . 2 2 . . . . 2
11:   1 1 1 1 1 1 1 1 1 1 1

gap>
```

¿Y por que se aplicaron tantas comparaciones extrañas?, pues porque el Algoritmo 0 fue el producto de la unión de algunos de algoritmos previos que no tenian tantas comparaciones pero que todod daban muchisimos contraejemplos que obviamente no lo eran, y en el afán de encontrar una verdadera forma canónica se fusionaron todos. De cualquier forma, fué un buen intento.

Eso pasa por fiarse de la fé. Desechemos pues, este Algoritmo inútil y dejémoslo como una curiosidad y como relleno para la tesis.

2. Algoritmo 2

2.0.2. Los empates. Este algoritmo considera empates a las filas que tienen las mismas entradas en la primera entrada, es decir, se fila en los empates de la primera columna. Despues se desempatan usando varios criterios.

```
gap> Display(TableOfMarks("S4"));
1:  24
2:  12 4
3:  12 . 2
4:   8 . . 2
5:   6 6 . . 6
6:   6 2 . . . 2
7:   6 2 2 . . . 2
8:   4 . 2 1 . . . 1
9:   3 3 1 . 3 1 1 . 1
10:   2 2 . 2 2 . . . . 2
11:   1 1 1 1 1 1 1 1 1 1 1

gap>
```

Estan empatadas las filas $[2, 3]$ y $[5, 6, 7]$. Este algoritmo solo trabajará sobre los empates, las demas filas no se intercambian pues se consideran ya ordenadas de mayor a menor.

2.0.3. Algoritmo esqueleto. El siguiente es el pseudocódigo esqueleto, que es el algoritmo principal, pero no el mas importante, el mas importante es el que canoniza las matrices.

- Para un orden n
  - Encontrar grupos no abelianos de orden n

- Hacer una lista L de matrices de mismas dimensiones de dicho orden
- Canonizar las matrices de la lista L
- Comparar las matrices del mismo orden (**importante**)
  - Si 2 matrices son distintas (no isomorfas) y el algoritmo de anillos Burnside devuelve que son isomorfos $\Rightarrow$ ¡Posible contraejemplo!

2.0.4. Algoritmo de canonización.

- Se recibe la matriz de tabla de marcas mtom.
  - Crear una lista de empates a partir de la primera columna (ej. $[[2,3],[5,6,7]]$)
  - Para cada empate de la lista de empates (por ejemplo $[2,3]$ y luego $[5,6,7]$).
    - Ver si hay intercambios de filas recomendables
      - Para comparar un par de filas
        1. Si los elementos correspondientes a la diagonal de la matriz son iguales, devolver una comparación elemento a elemento de las filas, es decir **return filai <filaj**.
        2. Si no, tomar las filas y ordenarlas usando **Sort(f)**, compararlas y devolver el resulta2do.
    - Seguir intercambiando filas hasta que ya no sea posible.
  - Cuando todos los empates esten canonizados, devolver la matriz resultante.

3. Algoritmo de Anillos de Burnside Isomorfos

El Algoritmo 2 requiere de una comparación de Anillos de Burnside, la cual es la siguiente, implementada en GAP.

```
#---------------------------------
espVectIguales:=function (mat1, mat2)
# Checar que la i-esima fila de de mat1 este generada
# por las i-1 filas de mat2 y viceversa.
```

```
local base1, base2, f, coef1, coef2, result, coefResults;
base1:=Basis(VectorSpace(Rationals, mat1),  mat1);
base2:=Basis(VectorSpace(Rationals, mat2),  mat2);
coefResults:=[];
# No probamos la primera fila pues es el orden del grupo, y
# ya sabemos que es el mismo en ambas, ent empezamos desde fila 2.
for f in [2..(DimensionsMat(mat1)[1])] do
coef1 := Coefficients(base1, mat2[f]);
coef2 := Coefficients(base2, mat1[f]);
if (IsList(coef1)) and (IsList(coef2)) then
if ForAny(coef1, noEntero) or ForAny(coef2, noEntero) then
Add(coefResults, false);
else
Add(coefResults, true);
fi;
else # Si no son listas algo fallo, tal ves no se pudieron siquiera poner
Add(coefResults, false);
fi;
od;
# en caso de haber al menos un "false" ...
if Length( Filtered(coefResults, x-> not x)) > 0  then
return false; #Todo falla,
else
return true; # o bien se generan una a la otra de manera entera.
fi;
end;
#--------------------------------
```

El algoritmo **espVectIguales(mat1, mat1)** recibe 2 matrices de argumento.

Sean $mat1$ y $mat2$ dos matrices de tablas de marcas, sean V_1 y V_2 los espacios vectoriales generados por $mat1$ y $mat2$ sobre $\mathbb{Q}$, tomemos a $base1$ y $base2$ como las bases de V_1 y V_2 respectivamente.

Entonces las filas de cada matriz estan en su espacio vectorial respectivo. Si toda r_i fila de $mat1$ esta generada de manera entera por los elementos de $base2$ y toda u_i fila de $mat2$ esta generada de manera entera por los elementos de $base1$, decimos para nuestros propósitos que dichas matrices generan Anillos de Burnside isomorfos, o algo muy parecido. Suficiente para que en nuestro algoritmo sea capaz de distinguir anillos de Burnside isomorfos y anillos no-isomorfos.

Aunque notemos lo siguiente, el algoritmo podria reconocer que ciertas matrices generan anillos de Burnside isomorfos cuando no sea verdad, pero lo cierto es que podemos confiar en él cuando devuelva que los anillos de Burnside no son isomorfos.

Es entonces una buena aproximación, que dará falsos positivos en el peor de los casos pero nunca falsos negativos, por lo cual, aunque seguramente contemos algunos de mas, no se nos escapará ninguno.

CAPíTULO 5

Resultados

DEFINICIÓN 5.1. *Sean G, Q grupos finitos. sea ψ una funciona de una familia de clases de conjugación de los subgrupos de G a la familia de clases de conjugación de los subgrupos de Q. Dado un subgrupo H de G, denotamos por H' a cualquier representante de $\psi([H])$. Decimos que ψ es un isomorfismo entre tablas de marcas de G y Q si ψ es una bijección, y si $\#(K'^{H'}) = \#(K^H)$ para todos los subgrupos H, K de G.*

1. Atributos que se preservan.

LEMA 5.2. *Sea n un entero positivo mayor o igual a 5 Sea G un grupo de orden $n!$ tal que G tiene un único subgrupo normal propio H. Asumimos que H tiene índice 2 en G, y que hay otro subgrupo T de G de índice n. Entonces $G \cong S_n$, donde S_n es el grupo simétrico de órden n.*

DEMOSTRACIÓN. Consideramos la acción de G en las clases laterales gT de T en G. Esto nos da un morfismo $\phi : G \longrightarrow S_n$. Puesto que G actúa transitivamente en G/T, el kernel de ϕ no puede ser G o H. Entonces ϕ es injectiva, se sigue que es entonces un isomorfismo. □

TEOREMA 5.3. *Sea G un grupo cuya tablas de marcas es isomorfa a la de S_n para algun $n \geq 5$. Entonces G es isomorfo a S_n.*

DEMOSTRACIÓN. Las hipotesis del lema 5.2 se satisfacen. □

TEOREMA 5.4. *Sean G, Q grupos finitos con tablas de marcas isomorfas. Sean K, H subgrupos de G, y sean K', H' representantes en sus respectivas clases de conjugación de subgrupos bajo el isomorfismo entre sus tablas de marcas.*

Entonces se tiene que:

- $G' = Q, (1_G)' = 1_Q, |G| = |G'|, |H| = |H'|, \alpha(H,K) = \alpha(H',K'), \beta(H,K) = \beta(H',K'), |N_G(H)| = |N_{G'}(H')|$.
- *Si G es abeliano entonces $G \cong G'$.*
- *El grupo G es nilpotente si y solo si G' es nilpotente. sin embargo, hay p-grupos no isomorfos con tablas de marcas isomorfas.*
- *Si G es un p-grupo con un subgrupo cíclico de índice p, entonces $G \cong G'$.*
- *Los grupos abelianizados son isomorfos, es decir, $G/[G,G] \cong G'/[G',G']$.*
- *Los subgrupos del conmutador se corresponden, es decir, $[G,G]' = [G',G']$.*
- *Los subgrupos de Frattini se corresponden, es decir, $\Phi(G)' = \Phi(G')$.*
- *Si G es un p-grupo, entonces $Zoclo(Z(G))' = Zoclo(Z(G'))$.*
- *El subgrupo H es cíclico si y solo si H' es cíclico.*
- *El subgrupo H es un elemental abeliano si y solo si H' es un elemental abeliano.*
- *El subgrupo H es normal en G si y solo si H' es normal en G'. En este caso, G/H y G'/H' tienen tablas de marcas isomorfas.*
- *Si $K \leq H$ y al menos uno de los siguientes grupos es normal en G, entonces $K' \leq H'$: K' o H'.*
- *Si K y H son subgrupos normales de G, entonces $(K \cap H)' = K' \cap H'$ y $(KH)' = K'H'$. En particular, dos subgrupos normales con intersección trivial se corresponden a dos subgrupos normales con intersección trivial. Además, si $G = K \times H$, entonces $G' = K' \times H'$, K y K' tienen tablas de marcas isomorfas, y H y H' tienen tablas de marcas isomorfas.*
- *El subgrupo H es maximal en G si y solo si H' es maximal en G'.*

Demostración. Solo probamremos algunas propiedades.

- Sea H tal que es un subgrupo maximal de G. Sea M' un subgrupo de G' entre H' y G', y sea M un subgrupo correspondiente en G. Puesto que $0 \neq \alpha(H',M') = \alpha(H,M)$, un conjugado de M contiene a H. Pero H es maximal, entonces este conjugado es H (ent. $M' = H'$) o bien es G (ent. $M' = G'$).
- Sea $X = \Phi(G)$. La simetría, es suficiente para mostrar que $X' \leq \Phi(G')$. Note que X es un subgrupo normale de G contenido en todos los subgrupos

maximales. Puesto que los subgrupos maximales se corresponden, X' es un subgrupo normal de G' contenido en todos los subgrupos maximales, entonces $X' \leq \Phi(G')$.

- Note que el Zoclo del centro de un p-group se caracteriza por ser el subgrupo normal mas chico de G tal que tiene una intersección no trivial con cada subgrupo normal no trivial de G. Esta propiedad se preserva bajo la correspondencia.
- El conmutador de un subgrupo es el subgrupo normal mas pequeño de G con un cociente abeliano. Esta propiedad se preserva bajo la correspondencia.
- Los p-grupos con un subgrupo cíclico de índice p estan clasificados.

□

2. Subgrupos elementales abelianos

LEMA 5.5. *Sea G un grupo finito, p un numero primo, a, b dos elementos en G de orden p, A el subgrupo generado por a y b. Si A tiene orden p^2, entonces A es un grupo elemental abeliano y $\#\{(c,d) \in G \times G \mid < c,d >= A\} = \#\{(e,f) \in C_p^2 \times C_p^2 \mid |< e,f >| = p^2\}$, donde C_p^2 es el grupo elemental abeliano de orden p^2.*

DEMOSTRACIÓN. El grupo A es abeliano porque es de orden p^2. Puesto que A es generado por dos elementos de orden p que conmutan, A no pueden ser ciclico, entonces tiene que ser elemental abeliano. Note que $\#\{(c,d) \in G \times G \mid < c,d >= A\} = \#\{(x,y) \in A \times A \mid < x,y >= A\} = \#\{(z,w) \in C_p^2 \times C_p^2 \mid < z,w >= C_p^2\}$. □

DEMOSTRACIÓN. Sea p un número primo, P, P' grupos tal que $|P| = |P'| = p^n$, with $n > 1$. Asumimos que P es elemental abeliano, que todos los elementos no triviales de P' tienen orden p, y que P' no es abeliano. Sea μ (respectivamente, μ') el número de subgrupos de P (respectivamente, P') de orden p^2. Entonces $\mu' < \mu$. □

DEMOSTRACIÓN. Sea $\Lambda = \{(a,b) \in P \times P \mid |< a,b >| = p^2\}, \Lambda' = \{(a',b') \in P' \times P' \mid |< a',b' >| = p^2\}$. Definamos relaciones equivalentes $\tilde{,}'$ en Λ y Λ' respectivamente de la siguiente manera: $(a,b)\tilde{(}c,d)$ si y solo si $< a,b >=< c,d >$, y $(a',b')\tilde{}'(c',d')$ si y solo si $< a',b' >=< c',d' >$. Note que por el Lema 5.5, cada clase de equivalencia

en Λ tiene la misma cardinalidad que cada clase de equivalencia en Λ'. Sea τ esta cardinalidad. Entonces tenemos que, $\mu = \#(\Lambda/\tilde{)} = \#\Lambda/\tau$, y $\mu' = \#(\Lambda'/\tilde{'}) = \Lambda/\tau$. Debemos mostrar que $\#\Lambda' < \#\Lambda$. Sea $\Gamma = \{(a', b') \in P' \times P' \mid a' \neq 0, b' \notin < a' >\}$. Debemos probar que Λ' es un subconjunto propio de Γ' y que Γ' y Λ tienen la misma cardinalidad.

Sea $(a', b') \in \Lambda'$. Puesto que $| < a', b' > | = p^2$, entonces $a' \neq 0$ (pues $| < b' > | = p$), y $b' \notin < a' >$ (pues $| < a' > | = p$), es decir, $(a', b') \in \Gamma'$. Por otro lado, ya que P' no es abeliano, existe a', b' en P' tal que no conmuta, entonces en particula tenemos que$| < a', b' > | > p^2$, $a' \neq 0$, y $b' \notin < a' >$, es decir, (a', b') esta en Γ' pero no esta en Λ'.

Finalmente, note que tanto los elementos de Γ' como los de Λ pueden ser contados de ls siguiente manera: hay $p^n - 1$ maneras de elegir el primer elemento, y $p^n - p$ formas de elegir el segundo, entonces cada conjunto tiene $(p^n - 1)(p^n - p)$ elementos. □

Corolario. Sean G y G' grupos finitos con tablas de marcas isomorfas. Si H es subgrupo elemental abeliano de G, entonces H' es subgrupo elemental abeliano de G'.

3. Contraejemplos

El software que GAP que escribí en para encontrar contrajemplos, fue modificado para encontrar grupos no isomorfos con tablas de marcas isomorfas sobre la librería de GAP.

El primer ejemplo (el mas chico) conocido antes de esta tesis era el par de grupos de Thevenaz de orden $5 * 11^2 = 625$.

Ahora el ejemplo mas chico es de orden 96.

Definición 5.6. *Sea M el producto directo $S_3 \times C_8$ (esto es, el grupo simétrico de orden 3, por el grupo cíclico de orden 8). Daremos dos productos semidirectos de M con el grupo cíclico de oden 2. Denotemos los elementos de M como (σ, x^n), donde σ es una permutación en S_3 y x es el generador de C_8. El grupo M es generado por los elementos $((1,2,3),1), ((1,2),1), ((),x)$. Consideremos los siguientes dos automorfismos de orden 2 de M, α y β, dados por: $\alpha((1,2,3),1) =$*

$\beta((1,2,3),1) = ((1,2,3),1), \alpha((1,2),1) = \beta((1,2),1) = ((1,2),x^4), \alpha((),x) = ((),x)$ *and* $\beta((),x) = ((),x^5)$. *Sea* G *el pruducto semidirecto de* M *con* C_2 *usando* α, *y sea* Q *el producto semidirecto de* M *con* C_2 *usando* β. *Ambos* G *y* Q *son grupos de orden* 96, *pero no son isomorfos; en realidad, en GAP* G *es* ***SmallGroup(96,108)***, *y* Q *es* ***SmallGroup(96,114)***.

Consideremos el siguiente código escrito en GAP:

```
G := SmallGroup(96,108);
Q := SmallGroup(96,114);
#-----------------------------
testlattice := function(g,h)
# Explore the lattices of subgroups of the groups g and h.
  local lat, conj, ans, subg, lath, conjh, n, subh;

  lat := LatticeSubgroups(g);
  conj := ConjugacyClassesSubgroups(lat);
  lath := LatticeSubgroups(h);
  conjh := ConjugacyClassesSubgroups(lath);

  for n in [1..Size(conj)] do
    subg := ClassElementLattice(conj[n],1);
    subh := ClassElementLattice(conjh[n],1);

     if not(IsAbelian(subg)=IsAbelian(subh)) then
       Print("Subgroup number ",n,", Order ",Order(subg),
             "  ",IsAbelian(subg),"  ",IsAbelian(subh),"\n");
     fi;
  od;
end;
#--------------------------------------
subn := function(g,n)
```

```
# Return a representative of the n-th conjugcy class of subgroups
# of g.
  return ClassElementLattice(ConjugacyClassesSubgroups
           (LatticeSubgroups(g))[n],1);
end;
```

Despues de cargar este archivo en GAP, checamos las tablas de marcas de G y Q, son idénticas en GAP:

```
gap> Read("testiso");
gap> MatTom(TableOfMarks(G))=MatTom(TableOfMarks(Q));
true
```

Esto significa que hay un isomorfismo entre las tablas de marcas de G y Q. Mas aún, con las tablas de marcas por defecto asignadas por GAP, este isomorfismo mapea la n-esíma clase de conjugación de subgrupos de G a la n-esíma clase de conjugación de subgrupos de Q. Nos preguntamos si el centro de G y el centro de Q se corresponden bajo este isomorfismo.

```
gap> Size(Centre(G));
8
gap> Size(Centre(Q));
4
```

Lo anterior prueba que los centros de G y Q no pueden corresponderse bajo este o bajo ningún otro isomorfismo entre las tablas de marcas (pues los ismorfismos deben preservar el orden de los subgrupos). Enseguida nos preguntamos si los subgrupos abelianos de G se corresponden do los subgrupos abelianos de Q.

La funcion de GAP siguiente, **testlattice(G,Q)**, corre sobre todos las clases de conjugación de los subgrupos de G y Q (los cuales tienen la misma longitud), y prueba si los ubgrupos correspondientes son abelianos o no-abelianos. Cuando encuentra un par de subgrupos que no concuerdan, los imprime en pantalla, muestra su orden y dice si son abelianos o no.

```
gap> testlattice(G,Q);
Subgroup number 36, Order 16  true  false
```

```
Subgroup number 37, Order 16  false  true
Subgroup number 38, Order 16  false  true
Subgroup number 40, Order 16  true  false
Subgroup number 58, Order 48  true  false
gap>
```

Esos son los unicos subgrupos correspondientes en los que no son ambos abelianos o no son ambos no-abelianos. Note que hay exactamente un subgrupo abeliano de G de orden 48 el cual no corresponde a ningun grupo abeliano de Q, en realidad, GAP nos dice que Q ni siqueira tiene subgrupos de orden 48. Esto significa que G tiene exactamente un grupo abeliano más de orden 48 que Q, entonces no puede haber isomorfismos entre las tablas de marcas de G y Q que preserve los subgrupos abelianos.

Finalmente, mostramos que la tabla de marcas no puede proveer de suficiente información para determinar el normalizador de un subgrupo. Considere la función `subn(g,n)`, la cual regresa un representante de la n-ésima clase de conjugación de subgrupos del grupo g.

```
gap> Normalizer(G,subn(G,2))=subn(G,58);
true
gap> Normalizer(Q,subn(Q,2))=subn(Q,58);
false
```

En ambos casos teníamos un subgrupo en la segunda clase de conjugacion de subgrupos; en G, su normalizador era el único subgrupo normal en la clase de conjugación 58, pero en Q, el subgrupo correspondiente no es el normalizador.

Podemos recapitular todo esto en el siguiente resultado.

TEOREMA 5.7. *Sea G y Q grupos finitos con tablas de marcas isomorfas, y sea $H \mapsto H'$ el isomorfismo entre sus tablas de marcas. Tenemos que:*

1. *H y H' pueden no ser isomorfas.*
2. *Aunque H es abeliano, H' puede no ser abeliano.*
3. *H y H' pueden tener diferentes tablas de marcas.*

4. *Aunque $K \times L = H$, puede no ser posible encontrar K', L' and H' tal que $K' \times L' = H'$.*
5. *Aunque K es normal en H, puede no ser posible escojer K' y H' tal que K' sea normal en H'*
6. *Dado H, la tabla de marcas no determina cual subgrupo de G es el normalizador de H en G.*

DEMOSTRACIÓN. Sea G el `SmallGroup(96,108)` de GAP y Q el `SmallGroup(96,114)`. Sea H el unico subgrupo abeliano de G de orden 48 tal que H' no es abeliano.

1. Esto es conocido desde el ejemplo de Thévenaz's, pero es tambien consecuencia de nuestro contraejemplo.
2. Es inmediato.
3. Se sigue del anterior y de que la tabla de marcas determina un grupo abeliano hasta isomorfismo
4. Si fuera cierto, y ya que los subgrupos cíclicos se corresponden, se seguiria que los subgroups abelianos se corresponden con subgroups abelianos.
5. El subgrupo `subn(Q,2)` fué un contraejemeplo.
6. Ya lo habíamos notado antes.

□

CAPíTULO 6

Conclusiones

1. Resúmen de Aportaciones

Los algoritmos que escribí guiado por mi asesor fueron modificados para encontrar grupos no isomorfos con tablas de marcas isomorfas sobre la librería de GAP. Desde el primer algoritmo (Algoritmo 0), ya se tenian algunos resultados que luego hubo que corroborar con otros algoritmos, mi asesor intuyó correctamente que dichos resultados serían importantes en el desarrollo de la tesis y que podrían dar lugar a algunos felices descubrimientos, pero se dejaron momentáneamente de lado.

Luego retomamos dichos resultados para analizarlos con más calma. Al analizarlos, logramos encontrar el ejemplo de `SmallGroup(96,108)`, `SmallGroup(96,114)`, que son dos grupos no isomorfos con Tablas de Marcas isomorfas. Un hecho por demas curioso es que estos dos grupos muestran tablas de marcas idénticas en GAP, mostramos enseguida que no puede haber ningún isomorfismo entre dichos grupos. Esto ya es un descubrimiento nuevo.

El primer ejemplo (el mas chico) conocido antes de esta tesis era el par de grupos de Thevenaz de orden $5 \times 11^2 = 625$. En cuanto a eso, con esta tesis mi asesor y yo hemos conseguido un ejemplo de orden mas pequeño.

Ahora el ejemplo es de orden 96, bastante mas chico que el de orden 625 obra de Thevenaz.

2. Investigación Futura

Mediante el uso del código de GAP, logramos hacer algunos algoritmos para intentar canonizar las Tablas de Marcas de los grupos de la librería de GAP. Sin embargo a lo largo del desarrollo de esta tesis encontramos ciertos tropiezos pues la Tablas de Marcas son muy grandes y es difícil canonizarlas pues debemos mantenerlas todas en

memoria cuando estamos procesando cada grado de grupo. Para los grupos de órdenes bajos son pocas, pero aumentan demasiado mientras aumentamos el orden por la gran cantidad de grupos que existen mientras tomamos un orden mas alto. Ademas las matrices se vuelven cada vez mas grandes.

Logramos dos algoritmos principales que pueden ser ejecutados, pero que resuelven poco acerca de la canonización. Un algoritmo final pude ser de orden combinatórico y puede ser ejecutado en computadoras futuras.

Los Algoritmos de Canonización ejecutarían mucho mejor si se usara un lenguaje de más bajo nivel, como por ejemplo C. Pero eso traería algunas complicaciones, pues no se pueden leer los grupos de la biblioteca de GAP directamente ya que no estan en formato de texto plano. También es un tema a investigar la forma en que GAP genera las Tablas de Marcas a partir de sus archivos fuente, pues es conocido que GAP optimiza el espacio usado mediante la inclusión de algunas Tablas de Marcas pequeñas dentro de otras más grandes, y luego usa algún algoritmo para generarlas "al vuelo" cuando le son requeridas. Es importante mencionar que las relaciones que usa GAP para incluir una Tabla de Marcas en otra mas grande no necesariamente tienen que ver con relaciones matemáticas, son simplemente estrategias para compresión de datos que tratan a las Tablas de Marcas como meros grupos de numeros. Es fundamental para una mejora futura el escribir rutinas en C para leer las tablas de marcas de los grupos de la librería de GAP.

La diversión con las Tablas de Marcas no acaba.

APÉNDICE A

Código fuente

No creo que alguien se quiera poner a transcribir lo siguiente, pues esta en el CD y basta copiarlo de ahi, pero debo ponerlos porque si no, a lo mejor no me creen que de verdad hice algo. Finalmente, la tesis se trataba de eso, de escribir codigo e intentar canonizar las tablas de marcas. Esto que viene son muchas hojas pues latex utiliza mucho espacio para poner los que llama "verbatims", simplemente las puedes saltar y tomar el CD. Lamentablemente no hay un ZIP para libros que funcione directamente en el cerebro y descomprima libros escritos en ZIP, tal vez en el futuro.

0.1. Algoritmo 0.

0.1.1. algoritmo05.g. A pesar del nombre, es el en realidad el Algoritmo 0. Solo que es la 5a versión del Algoritmo 0, despues de integrar en él algunos parches.

```
#--------------------
#Leemos las librerias
#--------------------
LoadPackage("guava");

Read("hayCeros.g");
Read("comparaPatitas.g");
Read("difCeros.g");
Read("elmejorcambio.g");
Read("swapRows.g");
Read("minimizatom.g");
Read("guardar-en-archivo.g");
Read("buscarSmithyTomIguales.g");
#--------------------
```

```
ordenIni:=1;
ordenMax:=42;
orden:=ordenIni;

result:=0;

while (orden<=ordenMax) do
dimensionesDeMatrices:=[];

cuantosSmallGroupsHay:=NumberSmallGroups(orden);
Print("\nOrden ", orden);
Print(" hay ",cuantosSmallGroupsHay);
nGrupo:=1;

log:=Concatenation("\nOrden ", String(orden), " hay ", String(cuantosSmallGroupsHa
AppendToFile ("log.txt", log);

if (cuantosSmallGroupsHay > 1) then # no tiene caso calcular nada si solo hay 1 gr

# Obteniendo las matrices de tablas de marcas
while (nGrupo <= NumberSmallGroups(orden)) do
if (not IsAbelian(SmallGroup(orden,nGrupo))) then
tablaDeMarcas:=TableOfMarks(SmallGroup(orden,nGrupo));
nxn:=DimensionsMat(MatTom(tablaDeMarcas))[1];
log:=Concatenation("\n\tMatTom: SmallGroup (", String(orden), ",", String(nGrupo),
else
log:=Concatenation("\n\tMatTom: SmallGroup (", String(orden), ",", String(nGrupo),
fi;
```

```
Print(log);
AppendToFile ("log.txt", log);

Add(dimensionesDeMatrices, nxn);
nGrupo:=nGrupo+1;
od; # end while

# Contando matrices de dimensiones nxn iguales
gruposConMatsDeDimensionIgual:=[];
mcuenta:=1;
nGrupo := nGrupo-1;
listaMatdeDimIguales:=[];
tmpYaContados:=[];
Print ("\n\t -> Buscando matrices (no abelianas) de las mismas dimensiones");

# Como Bubble sort, casi, chafa la busqueda!!
while (mcuenta < nGrupo) do
tempListGs:=[];
#Print("\n\t->", mcuenta, ":", dimensionesDeMatrices[mcuenta]);

# Solo se cuentan los que aun no se han contado, evita duplicidad
if (not dimensionesDeMatrices[mcuenta] in tmpYaContados) then
#Print ("\t", dimensionesDeMatrices[mcuenta], " not in ", tmpYaContados);
#Print ("OK");
  Add(tmpYaContados, dimensionesDeMatrices[mcuenta]);
  Add(tempListGs, mcuenta);

m2cuenta := mcuenta+1;
while (m2cuenta <= nGrupo) do
#Print (":", dimensionesDeMatrices[m2cuenta]);
```

```
if (dimensionesDeMatrices[mcuenta] = dimensionesDeMatrices[m2cuenta]) then
  Add(tempListGs, m2cuenta);
Print ("+");
fi;
m2cuenta:=m2cuenta+1;
od;
fi; # Solo se cuentan los que aun no se han contado, evita duplicidad

if (Length(tempListGs)>1) then
# Print(tempListGs);
Add(listaMatdeDimIguales, tempListGs);
fi;
mcuenta:=mcuenta+1;
od;
Print("\n\t",listaMatdeDimIguales);


# Ahora usamos la lista listaMatdeDimIguales para buscar contraejemplos a la conje
if (Length(listaMatdeDimIguales) > 0) then
scuenta:=1;
while (scuenta <= Length(listaMatdeDimIguales)) do
TomCanonizadas:=[];
TomCanonizadasSmithDiagonal:=[];
Print ("\n\tChecando:", listaMatdeDimIguales[scuenta]);
AppendToFile ("log.txt", Concatenation("\n\tChecando: ", String(listaMatdeDimIgual
tcuenta:=1;
Print("\n\t\tCanonizando Tom's");
while (tcuenta <= Length(listaMatdeDimIguales[scuenta])) do
Print ("\n\t\t (",orden,", ",listaMatdeDimIguales[scuenta][tcuenta],")");
tablaDeMarcas:=TableOfMarks(SmallGroup(orden,listaMatdeDimIguales[scuenta][tcuenta
```

```
#Display(tablaDeMarcas);
# Se minimiza/canoniza ----------------------
isMinimal:=false;
while(isMinimal=false)do
result:=canonizaTom(tablaDeMarcas);
isMinimal:=result[1];
tablaDeMarcas:=result[2];
od;
#Print("Tabla Final:\n");
#Display(tablaDeMarcas);
Add(TomCanonizadas, tablaDeMarcas);
# De paso calculamos Forma de Normal de Smith
Add(TomCanonizadasSmithDiagonal, DiagonalOfMat(NormalFormIntMat(MatTom(tablaDeMarc
#Unbind(tablaDeMarcas);
# Fin de la canonizacion --------------------
tcuenta:=tcuenta+1;
od; # fin de canonizando tablas de marcas


#TomCanonizadasSmithDiagonal[2]:=TomCanonizadasSmithDiagonal[1];
#TomCanonizadas[2]:=TomCanonizadas[1];

# Buscar Colisiones Smith ..................................
# Dos o mas matrices que tengan la misma Forma Normal de Smith
Print("\n\t\tBuscando Colisiones Smith y Tom en: ", listaMatdeDimIguales[scuenta])
#Print("\n", TomCanonizadasSmithDiagonal);

# if (listaMatdeDimIguales[scuenta] = [ 5, 6, 7, 8, 9, 10 ]) then
# break;
# fi;
```

```
santoGrial:=buscarSmithyTomIguales(TomCanonizadas, TomCanonizadasSmithDiagonal);
#Display(santoGrial);
# ..................................................
if (Length(santoGrial) > 0) then
Print ("\n\t ### HIT ");
AppendToFile ("log.txt", Concatenation ("\n\t #### HIT: Smith Iguales, Tom distint
fi;

#Unbind(TomCanonizadas);
#Unbind(TomCanonizadasSmith);
scuenta:=scuenta+1;
od;
fi;
fi; # no tiene caso calcular nada si solo hay 1 grupo.
#Unbind (listaMatdeDimIguales);

orden:=orden+1;
od; # end while
```

0.1.2. hayCeros.g. Rutina que se encarga de verificar que dos filas sean intercambiables, lo que hace es verificar que haya ceros en las entradas requeridas.

```
#
# Esta funcion recibe dos numeros de filas i, k y la matriz
# a la cual pertenecen.
# Supone i < k.
# Da por hecho que:
# matrix[i][i] = matrix[k][k] y solo se fija en los ceros.
# para ver si es posible el intercambiar la fila i con la k.
#
# Por ejemplo, para m:=MatTom(TableOfMarks(SmallGroup(36,4)));
```

```
#
# hayCeros(6,11,m) --> true
# hayCeros(3,12,m) --> false
#
hayCeros:=function(i,k,matrix)
local t;
t:=i+1;

#Si la esquina no pasa aqui paramos de una vez
if matrix[k][i] <> 0 then
# Print("Esquina no cerooooooo!!!!!....");
return false;
fi;

# La esquina paso (hay un cero), probaremos lo demas
while t < k do
#Print("\n", matrix[t][i]);
#Print(" ", matrix[k][t]);
if (matrix[t][i] <> 0) or (matrix[k][t] <> 0) then
#Print ("Una entrada no es cerooooooo!!!");
return false;
fi;
t:=t+1;
od;
return true;
end;
```

0.1.3. comparaPatitas.g.

```
manosyPies:=function(i,k,matrix)
local t,m1,p1,m2,p2,x,counter,valores,string,n;
```

```
valores:=[];
string:=["","","",""];
m1:=[];
m2:=[];
x:=[];
t:=1;

n:=DimensionsMat(matrix)[1];
# Las manos
while t<i do
#Print ("\n", matrix[i][t], ",", matrix[k][t]);
Add(m1, matrix[i][t]);
Add(m2, matrix[k][t]);
t:=t+1;
od;
# Agregamos las Manitas
Add(x,m1);
Add(x,m2);

p1:=[];
p2:=[];
# Los pies
t:=k+1;
while t<=n do
#Print ("\n", matrix[t][i], ",", matrix[t][k]);
Add(p1, matrix[t][i]);
Add(p2, matrix[t][k]);
t:=t+1;
od;
# Agregamos las Patitas
```

```
Add(x,p1);
Add(x,p2);

################################
# Aqui se hace la comparacion #
################################

counter:=1;
while counter <= Length (x[1]) do
string[1]:=Concatenation(string[1], String(x[1][counter]));
string[2]:=Concatenation(string[2], String(x[2][counter]));
#Print("\nconcatenando: ", string[1], " - ", string[2]);
counter:=counter+1;
od;
counter:=1;
while counter <= Length (x[3]) do
string[3]:=Concatenation(string[3], String(x[3][counter]));
string[4]:=Concatenation(string[4], String(x[4][counter]));
#Print("\nconcatenando: ", string[3], " - ", string[4]);
counter:=counter+1;
od;


#Print("\nCadenas ");
#Display(string);

string[1]:=Int(string[1]);
string[2]:=Int(string[2]);
string[3]:=Int(string[3]);
string[4]:=Int(string[4]);
```

```
#Print("\nValores ");
#Display(string);

#Print("\n m1-m2: ", string[1]-string[2]);
#Print("\n p1-p2: ", string[3]-string[4]);
#Nunca deberiamos llegar aqui!
return x;
end;
```

0.1.4. elmejorcambio.g.

```
#
# elMejorCambio
#
# Recibe una lista y una matriz
#
# Encuentra el mejor lugar para el _primer elemento_ de la lista
# Se trata de comparar las patitas y manitas de la primer columna con las de
# todas las restantes, valorarlas y decidir con cual se debe cambiar en caso
# de que convenga cambiar.
#
# Presupone que el primer elemento de la lista es intercambiable con todos
# los demas, aunque quizas los demas no sean intercambiables entre si.
#

################
elMasPesado:=function(pesos)
local contador, ganador, temp;

   # tomamos como ganador al primero, por default ...
```

```
ganador:=2;
   #.. y ya no comparamos ese.
contador:=2;

while (contador <= Length(pesos)) do
#Print ("\n\t\tActual Ganador: ", ganador, " comparando con: ", contador);

if (pesos[contador][1] > [0]) then
#Print (" Gano: ", contador, " Pues ", pesos[contador][1], "> [0]" );
ganador:=contador;
fi;

contador:=contador+1;
od;
if (pesos[1] < [0]) then
ganador:=0; # Sin cambio
fi;
Print ("\t\t**Final Ganador ", ganador, " de ", Length(pesos));

end;

#########################################################################
elMejorCambio:=function(lista,matrix)
local ganador,difDeCeros,valorDiagonal,contador, pesos, permutaOrden,manosPies, va

pesos:=[];
normal1:=[];
normal2:=[];
ganador:="a";
```

```
# Empezaremos a valorar los cambios de lista[1] con los demas
contador:=2;

#Print ("\n\tAnalizando ", lista[1], ": ", lista);

while (contador <= Length(lista)) do
val:=[];
# Extraemos la manos y pies
manosPies := manosyPies(lista[1], lista[contador],matrix);
   # Diferencia de valores de diagonal [i,i]-[j,j]
# si este valor es positivo CAMBIAR
# si es cero evaluar otros elementos
# si es negativo NO cambiar. ------------------------------------------------
Add(val, matrix[lista[1]][lista[1]]- matrix[lista[contador]][lista[contador]]);
#-------------------------------------------------------------------------------

difDeCeros:= difCeros(manosPies);
Add(val,difDeCeros[1]);
Add(val,difDeCeros[2]);

Add(val, (Int(difDeCeros[3]) - Int(difDeCeros[4]))  );
Add(val, (Int(difDeCeros[5]) - Int(difDeCeros[6]))  );

# ------------------------------------------------------------

susto:=0;
if (manosPies[1] > manosPies[2]) then
susto:=susto+1;
fi;
if (manosPies[2] > manosPies[3]) then
```

```
susto:=susto+1;
fi;
Add(val, susto);

# Print ("\n\t[D, Zm, Zp, BinDifM, BinDifP, S] = ", val);

if (ganador = "a") then # Por defecto el primero es el ganador
ganador:=[];
Add(ganador, lista[contador]);
Add(ganador, val);
else # Si ya habia un ganador, compararlo con el valor actual
if (ganador[2]<val) then # Si el actual es menor, tenemos un nuevo ganador
ganador[1]:=lista[contador];
ganador[2]:=val;
fi;
fi;
contador:=contador+1;
od;
#Print (ganador);
# Print("\tCambiarlo a ", ganador[1]);
if (ganador[2] > [0,0,0,0,0,0]) then
#Print("\n\t", ganador);
return ganador[1];
else
return 0; # Se devuelve 0 en caso de que no deba cambiarse
fi;
end;
#######################################
```

0.1.5. difCeros.g.

```
difCeros:=function(array)
```

```
local ceros, contador, diferenciaDeCeros,binCeros;

diferenciaDeCeros:=[0,0];
ceros:=[0,0,0,0];
binCeros:=["","","",""];
contador:=1;

while (contador <= Length (array[1]) ) do
if (array[1][contador] = 0) then
ceros[1]:= ceros[1]+1;
binCeros[1]:=Concatenation(binCeros[1], "1");
else
binCeros[1]:=Concatenation(binCeros[1], "0");
fi;

if (array[2][contador] = 0) then
ceros[2]:= ceros[2]+1;
binCeros[2]:=Concatenation(binCeros[2], "1");
else
binCeros[2]:=Concatenation(binCeros[2], "0");
fi;

contador:=contador + 1;
od;

contador:=1;
while (contador <= Length (array[3]) ) do
if (array[3][contador] = 0) then
ceros[3]:= ceros[3]+1;
binCeros[3]:=Concatenation(binCeros[3], "1");
```

```
else
binCeros[3]:=Concatenation(binCeros[3], "0");
fi;

if (array[4][contador] = 0) then
ceros[4]:= ceros[4]+1;
binCeros[4]:=Concatenation(binCeros[4], "1");
else
binCeros[4]:=Concatenation(binCeros[4], "0");
fi;

contador:=contador + 1;
od;
diferenciaDeCeros[1]:=ceros[2]-ceros[1];
diferenciaDeCeros[2]:=ceros[4]-ceros[3];
Add(diferenciaDeCeros, binCeros[1]);
Add(diferenciaDeCeros, binCeros[2]);
Add(diferenciaDeCeros, binCeros[3]);
Add(diferenciaDeCeros, binCeros[4]);

return diferenciaDeCeros;
end;
```

0.1.6. swapRows.g.

```
# Esta es una funcion destructiva,
# pues modifica directamente la matriz
# que se envia como argumento: m
# Pero esto la hace mas veloz!
swapRows:=function(m, i,j)
local t;
t:=m[i];
```

```
m[i] := m[j];
m[j] := t;
end;
```

0.1.7. minimizatom.g.

```
canonizaTom:=function(tom)
local listaCambios,fixedList,permuTom,i,n,k,m,tmpList,cambiar,permut,stom,mattom,m
#Display(tom);
# inicializamos las listas
Print("*");
#Display(tom);

listaCambios:=[];
fixedList := [];

stom:=tom;
mattom:=MatTom(tom);
m:=ShallowCopy(mattom);

n:=DimensionsMat(mattom)[1];

# Inicializamos la lists de ordenaciones/permutaciones
permuTom:=[];

i:=2; #no necesitamos probar la primer fila, empezamos en la 2.

while i< n-1 do
k := i+1;
tmpList:=[];
Add(tmpList, i);
while k < n do
```

```
if (hayCeros(i,k,m)) then
Add(tmpList, k);
fi;
k := k+1;
od;
# Display(tmpList);
# tmpList[1] se puede cambiar con los demas
# por ejemplo [22,23,24,60,67] significa que
# 22 se puede cambiar con 23,24,60,67, pero
# NO significa que 60 se puede cambiar con 67.

if (Length (tmpList) > 1) then
Add(listaCambios, tmpList);
cambiar:=elMejorCambio(tmpList, m);
# cambiar=0 en caso de
# que no se deba cambiar.
# Si cambiar>0, cambiar es el elemento por el
# cual se debe cambiar.
if (cambiar > 0) then
permut:=(tmpList[1], cambiar);
#Print(permut);
#Display(m);

#permuTom es una lista que contiene todas las permutaciones
# que se van produciendo
Add(permuTom, permut);
# En estas 2 siguientes lineas se cambian
# filas y columnas de la matriz
swapRows(m,tmpList[1],cambiar);
m:=PermutedCols(m, (tmpList[1],cambiar));
```

```
# Y se cambia tambien la tabla de marcas
# stom es una abreviacion para sortedTom
stom:=SortedTom(stom, permut);
fi;
fi;
i:=i+1;
od;
#Display(stom);
minimal:=false;
if (MatTom(tom)=MatTom(stom)) then
# Print("\nAcabado!!");
minimal:=true;
fi;
return [minimal,stom];
end;
```

0.1.8. guardar-en-archivo.g.

```
# Funcion para escribir en archivo
#
# Recibe como parametro el nombre del archivo y la cadena a guardar
# si el archivo ya existe lo agrega al final,
# si el archivo no existe, crea uno nuevo.
#
AppendToFile := function ( file, s )
    local  otf;
    otf := OutputTextFile( file, true );
    SetPrintFormattingStatus( otf, false );
    AppendTo( otf, s );
    CloseStream( otf );
end;
```

0.1.9. buscarSmithyTomIguales.g.

```
#
# list
# contiene una lista de vectores, cada vector es la diagonal de la forma normal de
#
# busca vectores iguales entre los vectores recibidos, devuelve una lista (de list
# Ejemplo:
#         1   2   3   4   5   6   7
# list:=(v1, v1, v1, v2, v3, v4, v3)   devuelve [[1,2,3], [5,7]]
#
buscarSmithyTomIguales:=function(tomCanonicas, smiths)
local miContador, otroContador,tmpYaConte, tmpHits, hitList;

tmpYaConte:=[];
tmpHits:=[];
miContador:=1;
hitList:=[];
while (miContador < Length(smiths)) do
otroContador:=miContador+1;

tmpHits:=[];
#Print("\n", miContador);
if (not smiths[miContador] in tmpYaConte) then
Add (tmpYaConte, smiths[miContador]);
Add(tmpHits, miContador);
while(otroContador <= Length(smiths)) do
Print("\n", miContador, ":", otroContador);
Print("\n\t ", smiths[miContador]);
Print("\n\t ", smiths[otroContador]);
if (smiths[miContador] = smiths[otroContador]) then
Print("\t Smith Hit");
```

```
if (not (MatTom(tomCanonicas[miContador]) = MatTom(tomCanonicas[otroContador]))) t
Print("\t ToM Distinta");
Display (tomCanonicas[miContador]);
Display (tomCanonicas[otroContador]);
Add(tmpHits, otroContador);
fi;
fi;
otroContador:=otroContador+1;
od;
fi;
if (Length(tmpHits) > 1) then
# Print("\n\t\t", tmpHits);
Add(hitList, tmpHits); #<<<<<<<======= ESTE ES EL DETECTOR DE CONTRAEJEMPLOS: hitL
fi;
miContador:=miContador+1;
od;

# Print("\n\t * Resultado: ", hitList);
return hitList;

end;
```

0.1.10. canon2.g. Este es el algoritmo último de canonización, en este archivo (canon2.g) estan contenidas rutinas principales para extracción de empates tomando en cuenta la primer columna de la tabla de marcas, y las rutinas para encontrar posibles contraejemplos y tambien para encontrar grupos con tablas de marcas isomorfas. Este último resulta que nos dio una sorpresa al encontrar que dos grupos de orden 96 tenian exactamente la misma tabla de marcas en GAP, es decir, que ambos grupos tienen tablas de marcas isomorfas, esto a pesar de ser grupos no isomorfos; y como se platicó el mas pequeño conocido antes era de orden $11 * 11 * 5$.

```
#
```

```
# Necesitamos GUAVA
# guava se puede conseguir en
# http://www.gap-system.org/Packages/guava.html
#
LoadPackage("Guava");
tom:=[];
stom:=[];
mtom:=[];
tomInfo:=[];
Read("hayCeros.g");
Read("swapRows.g");
Read("menorquec2.g");
#--------------------------------
#
# filtraEmpates (col)
#
# Recibe una lista, y regresa un record con una lista de empates y
# una lista de elementos unicos (sin repetir).
# Probar usando:
# col:=[360,180,120,120,72,90,90,90,60,60,45,40,36,30,30,20,15,15,10,6,6,1]
#
# Se usa para mandarle la primera columna de la mTom y descrubir empates
# este es el filtro principal de empates
filtraEmpates:=function (col)
# Esta funcion solo considera empates en la columna que recibe,
# no toma en cuenta valores de diagonal, ni ninguna otra cosa
# recibe una lista de enteros y devuelve empates de esa lista.
local unique, lstEmpates, c, empate, d;

unique:=[]; # Numeros de la lista 'col', sin repetir
```

```
lstEmpates:=[];
for c in [1..Length(col)] do
empate:=[];
if not col[c] in unique then
Add(empate,c);
Add(unique,col[c]);
for d in [(c+1)..Length(col)] do
if col[c] = col[d] then
Add(empate, d);
fi;
od;
if Length(empate) > 1 then
Add(lstEmpates, empate);
fi;
fi;
od;
return rec(empates:=lstEmpates, elementos:=unique);
end;
#--------------------------------
# filtraEmpates2 (col)
# Se usa para mandarle la primera columna de la mTom y descrubir empates
# este es el filtro principal de empates  BORRAME
filtraEmpates2:=function (col)
# Esta funcion considera empates en la columna que recibe y ademas
# desempata usando la diagonal, luego desempata haciendo sort de los
# elementos de la fila y comparandolos.

# BORRAME: filtraEmpates2 NO SE USA PARA NADA!

local unique, lstEmpates, empate, c, d, e, colunique;
```

```
unique:=[]; # Contendra los numeros de la lista 'col', sin repetir
lstEmpates:=[];
# Primero detectar empates en la lista recibida,
# estos son empates en la primera columna
for c in [1..Length(col)] do
empate:=[];
if not col[c] in unique then
Add(empate,c);
Add(unique,col[c]);
for d in [(c+1)..Length(col)] do
if col[c] = col[d] then
Add(empate, d);
fi;
od;
if Length(empate) > 1 then
Add(lstEmpates, empate);
fi;
fi;
od;
# Desempatar usando la diagonal,
# los que tengan la diagonal igual aun son empates.
for c in [1..Length(lstEmpates)] do
Print("\n",lstEmpates[c]);
for d in [1..Length(lstEmpates[c])-1] do
#  Print ("\t", lstEmpates[c][d]);
for e in [d+1..Length(lstEmpates[c])] do
Print ("\t", lstEmpates[c][d],":", lstEmpates[c][e]);
od;
od;
```

```
od;

return rec(empates:=lstEmpates, elementos:=unique);
end;
#--------------------------------
noEntero:=function(num)
return not IsInt(num);
end;
#--------------------------------
espVectIguales:=function (mat1, mat2)
# Checar que la i-esima fila de de mat1 este generada
# por las i-1 filas de mat2 y viceversa.
local base1, base2, f, coef1, coef2, result, coefResults;
base1:=Basis(VectorSpace(Rationals, mat1),  mat1);
base2:=Basis(VectorSpace(Rationals, mat2),  mat2);
coefResults:=[];
# No probamos la primera fila pues es el orden del grupo, y
# ya sabemos que es el mismo en ambas, ent empezamos desde fila 2.
for f in [2..(DimensionsMat(mat1)[1])] do
coef1 := Coefficients(base1, mat2[f]);
coef2 := Coefficients(base2, mat1[f]);
if (IsList(coef1)) and (IsList(coef2)) then
if ForAny(coef1, noEntero) or ForAny(coef2, noEntero) then
Add(coefResults, false);
else
Add(coefResults, true);
fi;
else # Si no son listas algo fallo, tal ves no se pudieron siquiera poner
Add(coefResults, false);
fi;
```

```
od;
# en caso de haber al menos un "false" ...
if Length( Filtered(coefResults, x-> not x)) > 0  then
return false; #Todo falla,
else
return true; # o bien se generan una a la otra de manera entera.
fi;
end;
#---------------------------------
canonizaG:=function()
local col, empates, count;
col:=TransposedMat(mtom)[1]; # Primera columna
empates:=filtraEmpates(col).empates;
if Length (empates) > 0 then # Si hay empates, hay que canonizar
for count in [1..Length(empates)] do
  canonizaEmpateC4(empates[count]);
od;
fi;
end;
#---------------------------------
canonDistBurnsideIgual:=function()
local t,u;
Print("\nBuscando contraejemplos:");
for t in [1..( Length(tomInfo)-1 )] do
Print("\n Comparando: ", tomInfo[t].num, " con: ");
for u in [(t+1)..Length(tomInfo)] do
Print(" ", tomInfo[u].num);
#Checamos que las dimensiones de la matriz
if DimensionsMat(tomInfo[t].canonizada)[1] =
DimensionsMat(tomInfo[u].canonizada)[1] and
```

```
(tomInfo[t].canonizada <> tomInfo[u].canonizada) then
Print("*");
# Solo falta checar si los anillos de burnside son isomorfos, de serlo...
# es un posible contraejemplo !!
if espVectIguales (tomInfo[t].canonizada, tomInfo[u].canonizada) then
Print("\n\t\t ** Posible Contraejemplo: ", tomInfo[t].num, ",", tomInfo[u].num);
Print(espVectIguales (tomInfo[t].canonizada, tomInfo[u].canonizada));
fi;
fi;
od;
od;
end;
#--------------------------------
canonIguales:=function() # Este sirve para encontrar matrices ToM isomorfas
local t,u;
Print("\nBuscando ToM isomorfas:");
for t in [1..( Length(tomInfo)-1 )] do
Print("\n Comparando: ", tomInfo[t].num, " con: ");
for u in [(t+1)..Length(tomInfo)] do
Print(" ", tomInfo[u].num);
#Checamos que las dimensiones de la matriz sean iguales
if (tomInfo[t].canonizada = tomInfo[u].canonizada) then
Print("\n\t\t ** Son Isomorfas: ", tomInfo[t].num, " con ", tomInfo[u].num);
# Solo falta checar si los anillos de burnside son isomorfos, de serlo...
# es un posible contraejemplo !!
fi;
od;
od;
end;
#----------------------------------------
```

```
#          ALGORITMO PRINCIPAL
#----------------------------------------
gInicial:= 65;
gFinal  := 127;
tomInfo:=[]; # contiene records de todas las tom's de gpos
 # no abelianos de algun orden n.
for n in [gInicial..gFinal] do
tomInfo:=[]; # contiene records de todas las tom's de gpos
 # no abelianos de algun orden n.
grupos := gruposNoAbelianos(n);
Print ("\nNo Abelianos de Orden ", n, " hay ", Length(grupos));

if Length(grupos) > 1 then
for m in [1..Length(grupos)] do
tom  := TableOfMarks(grupos[m].grp);
stom := tom; # Una copia, sorted tom;
mtom := ShallowCopy(MatTom(TableOfMarks(grupos[m].grp)));
Print ("\nCanonizando ", n, ",", grupos[m].n);
canonizaG(); # Canoniza tom y la guarda en stom
Add(tomInfo, rec(num:=grupos[m].n, canonizada:=mtom,
vectSpace:=VectorSpace(Rationals,mtom)));
Unbind(tom);
Unbind(mtom);
Unbind(stom);
od;
fi; # Hasta este punto tenemos todos las Tom de SmallGroup(n,*) canonizadas

#----------------------------------------
# Encontrar toms con distinta forma canonica y anillos isomorfos,
# de encontrarse son candidatos a contradiccion de conjetura
```

```
#----------------------------------------

# Este se puede comentar y descomentar segun se requiera
canonDistBurnsideIgual(); # Busca contraejemplos

#----------------------------------------
# Encontrar toms forma canonica igual
# estas son ToM's isomorfas
#----------------------------------------

# Este se puede comentar y descomentar segun se requiera
#  canonIguales(); # Busca TOM's isomorfas

#----------------------------------------
   Unbind(tomInfo);
od;
```

0.2. menorqueC2.g. El Santo Grial, la funcion **menorqueCX** contiene los algoritmos que discriminan una fila de la otra, algunos muy simples como menorqueC2 que devuelven un valoor booleano, hasta otros mas inteligentes como menorqueC4 que es capaz de devolver 3 valores, cierto, falso e iguales, para indicar si la fila i es menor, mayor o igual que la j.

Tambien las funciones canonizaEmpateCX, se hicieron para usar las funciones menorqueCX respectivas. Estas funciones canonizan un empate de filas, es decir, varias filas que se consideran empatadas.

Las funciones hayPermutaCX detectan una y solo una permutación de una lista de empates recibidos como argumentos, se usan conjuntamente con canonizaEmpateCX y menorqueCX.

```
#-------------------------------------------------
# Decide si la fila i es "menor" que la fija j, en mtom
```

```
# Usando la Canonizacion 2 de Valero.
#
# Presupone que i, j son intercambiables, i.e. hayCeros(i,j,mtom) devuelve true.
#
# -> Usa la variable Global mtom, que contiene la matriz de Tabla de Marcas
menorQueC2:=function(i, j)
local c, alto,imenorquej;
alto:=false;
imenorquej:=false;
c:=2;
while (c < (Minimum(i,j)  ) and not alto) do
   Print(" {", c,"} |",mtom[i][c],":", mtom[j][c]);
if mtom[i][c] <> mtom[j][c] then
alto:=true;
imenorquej:=mtom[i][c] < mtom[j][c];
fi;
c:=c+1;
od;
if not alto then
Print(" \tDiag:[",mtom[i][i],",", mtom[j][j],"]");
imenorquej:=mtom[i][i] < mtom[j][j];
fi;
#Print (mtom[i], mtom[j]);
return imenorquej;
end;

#---------------------------------
menorQueC3:=function(i,j)
local cDiag, c, indistinguibles, imenorkj, u, v;
cDiag := Minimum(i,j); # Columna donde esta la diagonal mas cercana
```

```
imenorkj := false;
indistinguibles:=false;
c:=2;

while (mtom[i][c] = mtom[j][c] and c < cDiag) do # Se brinca las que son iguales
c:=c+1;
od;

if c< cDiag then # Si no comparó todas las columnas...
imenorkj := mtom[i][c] < mtom[j][c]; # hay una col con entradas distintas.
fi;

if c>= cDiag then  # Si todas las entradas son iguales...
if mtom[i][i] = mtom[j][j] then
indistinguibles:=true; # ...y tambien la diagonal, ent son indistinguibles.
else
imenorkj:= mtom[i][i] < mtom[j][j]; # Si no, comparamos la diagonal.
fi;
fi;

return
rec(cLeidas:=c-2,
diag :=[mtom[i][i],mtom[j][j]],
imenorquej := imenorkj,
indist:=indistinguibles);
end;
#-----------------------------------
menorQueC4:=function(i,j) # Devuelve si la fila i es menor que la fila j, o si son
local cDiag, c, indistinguibles, imenorkj, filai, filaj;
```

```
cDiag := Minimum(i,j); # Columna donde esta la diagonal mas cercana

imenorkj := false;
indistinguibles:=false;

filai:= ShallowCopy(mtom[i]);
filaj:= ShallowCopy(mtom[j]);

Sort(filai); # Ordenamos los elementos de las filas
Sort(filaj);

#     ____filas_____      ______diagonales______
if filai = filaj and mtom[i][i] = mtom[j][j] then
return "iguales";
  else # Ocurrio que las filas no son iguales o que las diagonales no son iguales,
# Si la diagonal es igual
if mtom[i][i] = mtom[j][j] then
return filai < filaj; # Si la diagonal es igual, entonces las filas son distintas
else
#Si la diagonal no es igual(es distinta), usamos la diagonal para saber quien es m
return mtom[i][i] < mtom[j][j];
fi;
fi;
end;
#----------------------------------
# Devuelve una lista con grupos no
# abelianos de orden = "orden"
#----------------------------------
gruposNoAbelianos:=function(orden)
local numeroDeGrupos, grupos, i;
```

```
grupos:=[];
for i in [1..NumberSmallGroups(orden)] do
if not IsAbelian(SmallGroup(orden, i)) then
Add(grupos, rec(grp:=SmallGroup(orden, i), n:=i));
fi;
od;
return grupos;
end;
#-----------------------------------
hayPermutaC3:=function(list)
local i;
for i in [ 1..(Length(list)-1) ] do
if menorQueC3(list[i], list[i+1]).imenorquej then
return [list[i],list[i+1]];
fi;
od;
return [];
end;
#-----------------------------------
hayPermutaC4:=function(list)
local i;
for i in [ 1..(Length(list)-1) ] do
#  Print ("::",menorQueC4(list[i], list[i+1]));
if menorQueC4(list[i], list[i+1]) = true then
return [list[i],list[i+1]];
fi;
od;
return [];
end;
#-----------------------------------
```

```
buscaIndistinguiblesC4:=function(rows)
# Recibe una lista de filas conteniendo puros empates
# y busca indistinguibles
# Ej Smallgorp(16,6), buscaIndistinguiblesC4([4,5,6]) devuelve [5,6]
local contados, lstEmpates, c, empate, d;

contados:=[];
lstEmpates:=[];
for c in [1..Length(rows)] do
empate:=[];
if not rows[c] in contados then
Add(empate,rows[c]);
Add(contados,rows[c]);
for d in [(c+1)..Length(rows)] do
Print("\t", rows[c], ":" , rows[d]);
if menorQueC4(rows[c], rows[d])="iguales" then
Add(empate, rows[d]);  # Tenemos un (par?) indistinguible
Add(contados,rows[d]); # Para no contarla de nuevo
fi;
od;
Display(contados);
if Length(empate) > 1 then # queremos 2 o mas indistinguibles
Add(lstEmpates, empate);# un indist solo, no es indistinguible
fi;
fi;
od;
return lstEmpates; # Estos son indistinguibles

end;
#----------------------------------
```

```
canonizaEmpateC3:=function(rows)
local res;
res:=hayPermutaC3(rows);
while ( Length(res) > 0 ) do
swapRows(mtom, res[1], res[2]);
mtom:=PermutedCols(mtom, (res[1], res[2]) );
stom:=SortedTom(stom, (res[1], res[2]) );
res:=hayPermutaC3(rows);
od;
end;

canonizaEmpateC4:=function(rows)
local res;
res:=hayPermutaC4(rows);
while ( Length(res) > 0 ) do
swapRows(mtom, res[1], res[2]);
mtom:=PermutedCols(mtom, (res[1], res[2]) );
stom:=SortedTom(stom, (res[1], res[2]) );
res:=hayPermutaC4(rows);
od;
end;
```

Bibliografía

[1] Serge Bouc. Burnside rings. *Handbook of Algebra*, 2:739–804, 2000.

[2] Kenneth S. Brown. *Cohomology of groups*. Springer, 1982.

[3] The GAP Group. GAP – Groups, Algorithms and Programming. Version 4.3; 2002 (http://www.gap-system.org).

[4] Alberto G. Raggi-Cárdenas and Luis Valero-Elizondo. Groups with isomorphic burnside rings. *Archiv der Mathematik*, 2005.