MSEL
Modelling in Science Education and Learning

Modelling in Science Education and Learning
Volume 11 (2), 2018 doi: 10.4995/msel.2018.9720.
Instituto Universitario de Matemática Pura y Aplicada
Universitat Politècnica de València

Jugando con la Teoría de Grupos: rompecabezas, puzzles y otros entretenimientos matemáticos

Playing with Group Theory: brainteasers, puzzles and other mathematical toys

María José Felipe, Víctor Manuel Ortiz Sotomayor
Universitat Politècnica de València
mfelipe@mat.upv.es, vicorso@doctor.upv.es.

Abstract

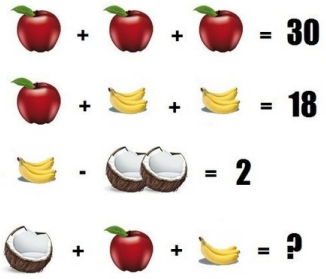
En el presente trabajo mostramos cómo modelizar y analizar la resolubilidad de algunos puzzles, rompecabezas y juegos matemáticos, haciendo uso de conceptos básicos de la teoría de grupos y del sistema algebraico computacional `GAP` *(Groups, Algorithms and Programming). Se trata de motivar al alumnado que se inicia en esta teoría algebraica y al mismo tiempo mostrar su aplicabilidad de una forma lúdica para aquellos estudiantes ya iniciados en este campo.*

In this paper we show how to model and to analyse the solubility of some puzzles, brainteasers and other mathematical toys. We make use of basic group theory concepts and the computational algebraic system `GAP` *(Groups, Algorithms and Programming). The main purpose is to motivate beginner students in this algebraic theory, and also to show the playful applicability to advanced students.*

Palabras clave: Puzzles, Permutaciones, Teoría de Grupos, Modelización
Keywords: Puzzles, Permutations, Group Theory, Modelling.

1. Introducción

Una de las mayores preocupaciones entre los profesionales en la docencia de las matemáticas es la poca interacción entre el alumnado y los diferentes contenidos impartidos. El formalismo de conceptos abstractos junto con densas teorías expuestas tanto en textos educativos como en el aula son quizás una de las principales causas que provoca la falta de motivación de los alumnos. La modelización matemática de juegos y entrenimientos, como puzzles y rompecabezas, puede ser una eficaz estrategia para incentivar dicha motivación, mostrando la aplicabilidad real de los conceptos teóricos desarrollados en clase. Ejemplos de ello podemos encontrarlos en las redes sociales, donde algunos problemas matemáticos representados por iconos, como el representado en la Figura 1, se han hecho virales:



$$\Longrightarrow \quad \begin{aligned} x+x+x &= 30 \\ x+4y+4y &= 18 \\ 4y-2z &= 2 \\ z+x+3y &= ? \end{aligned}$$

Figura 1 – ¿Cuál es el valor de la última ecuación?

Curiosamente, este inocente juego suscita un interés inmediato en numerosos usuarios, los cuales tratan de resolverlo incluso desconociendo su tratamiento algebraico, que posiblemente reduciría su popularidad considerablemente.

El método de utilizar juegos en la enseñanza es una innovadora y eficaz herramienta conocida como *game-based learning*. Esta tendencia pedagógica utiliza una componente lúdica con un objetivo concreto de aprendizaje. En esta línea, nuestro objetivo en este trabajo es doble: por un lado, modelar algunos puzzles y rompecabezas utilizando conceptos básicos de teoría de grupos y, por otro lado, ilustrar algunos métodos de resolución de los mismos con el software `GAP` (Groups, Algorithms and Programming). Desde un punto de vista docente, este tipo de técnicas pueden ayudar a comprender algunos conceptos teóricos de la teoría de grupos (como los grupos de permutaciones, permutaciones pares, grupo alternado, grupos libres, homomorfismos de grupos, construcciones de grupos, etc.) y ver su aplicabilidad.

2. Algunos conceptos de teoría de grupos

La teoría de grupos es conocida tradicionalmente como la teoría matemática de la simetría. Ello es debido a que la estructura de grupo aparece de manera natural al considerar el conjunto de todas las aplicaciones biyectivas que mantienen una estructura determinada. El uso de la teoría de grupos no queda confinado solo al ámbito de la matemática pura, sino que la idea de grupo como medida de simetría desempeña un importante papel en otras ramas del saber científico, como es el caso de la química (química cuántica, teoría de orbitales moleculares, cristalografía...), la física (mecánica cuántica), las ciencias de la computación y teoría de la información (teoría de códigos, criptografía,...), la arquitectura e incluso en ciencias humanas y sociales.

En esta sección presentamos algunas definiciones y resultados básicos de la teoría de grupos necesarios para el resto del trabajo. Para más información puede consultarse Hungerford, T. W. (1974).

Definición 5.1 *Dado un conjunto G y una operación $*$, decimos que G es un **grupo** si se cumplen las siguientes condiciones:*

1. *$a * b \in G$, para todo $a, b \in G$.*
2. *$a * (b * c) = (a * b) * c$, para todo $a, b, c \in G$.*

3. *Existe $e \in G$ tal que $a * e = e * a = a$, para todo $a \in G$. Dicho (único) elemento es denominado* ***elemento neutro****.*

4. *Para todo elemento $a \in G$, existe $a^{-1} \in G$ tal que $a * a^{-1} = a^{-1} * a = e$. Llamaremos a a^{-1} el (único)* ***elemento inverso*** *de a.*

El ***orden*** *de un grupo G es su cardinal como conjunto, y lo denotaremos por $|G|$. Si H es un subconjunto de G con estructura de grupo, diremos que H es un* ***subgrupo*** *de G. Salvo que se mencione explícitamente, en lo que sigue generalmente usaremos notación multiplicativa para la operación y el orden de un grupo es finito.*

Definición 5.2 *Sea X un conjunto finito y consideremos el conjunto $\Sigma(X)$ de todas las biyecciones de X en X. Si tomamos como operación la composición de funciones, entonces $\Sigma(X)$ posee estructura de grupo.*

Los elementos de $\Sigma(X)$ los llamaremos ***permutaciones*** *y a $\Sigma(X)$ lo denominaremos* ***grupo de permutaciones*** *(o* ***simetrías****) de X. Concretamente, cuando $X = \{1, 2, \ldots, n\}$ lo denotaremos por Σ_n, y su orden es $|\Sigma_n| = n!$. Como veremos, este grupo juega un papel crucial en este trabajo y en general en la teoría de grupos.*

Dichas permutaciones pueden ser descritas mostrando las imágenes del conjunto $\{1, 2, \ldots, n\}$ en función de ciclos. Por ejemplo, la permutación σ con imágenes $\sigma(1) = 2, \sigma(2) = 3, \sigma(3) = 1, \sigma(5) = 6$ y $\sigma(6) = 5$ sobre el conjunto $\{1, 2, 3, 4, 5, 6\}$ puede representarse como producto de ciclos $(1, 2, 3)(5, 6)$. Denominaremos a $(i_1, i_2, \ldots, i_k)$ tal que $\sigma(i_j) = i_{j+1}$ y $\sigma(i_k) = i_1$ un ***k-ciclo*** *y, en particular, diremos que un 2-ciclo es una* ***trasposición****. Además, el elemento neutro de este grupo es el ciclo correspondiente a la permutación identidad $()$.*

Un hecho que será muy importante es el siguiente:

Teorema 5.3 *Toda permutación de Σ_n se puede expresar como producto de trasposiciones (no necesariamente disjuntas).*

Cabe señalar que, aunque la descomposición de una permutación como producto de trasposiciones puede no ser única, puede probarse que la paridad del número de ellas es constante, i.e., siempre aparece un número par o impar de trasposiciones. Esto motiva la siguiente definición:

Definición 5.4 *Una permutación $\sigma \in \Sigma_n$ se dice* ***par*** *(respectivamente* ***impar****) si se puede escribir como producto de un número par (respectivamente impar) de trasposiciones.*

El conjunto formado por todas las permutaciones pares de Σ_n forma un grupo que se denomina ***grupo alternado*** *y se denota A_n. Su orden es $|A_n| = n!/2$.*

El análisis de cualquier clase de objetos algebraicos conlleva el estudio esencial de funciones que "preserven" la estructura, denominados morfismos:

Definición 5.5 *Sean G y H dos grupos y sean $*$ y $\star$ sus operaciones, respectivamente. Una función $f : G \longrightarrow H$ es un* ***homomorfismo de grupos*** *si para todo $a, b \in G$ se cumple que $f(a * b) = f(a) \star f(b)$. Si f es inyectiva, suprayectiva o biyectiva, diremos que f es un* ***monomorfismo****,* ***epimorfismo*** *o* ***isomorfismo****, respectivamente.*

Notemos que la intersección de subgrupos vuelve a ser un subgrupo, lo cual motiva la siguiente definición:

Definición 5.6 *Sea G un grupo y X un subconjunto de G. Sean $\{X_i : i \in I\}$ los subgrupos de G que contienen a X. Entonces $\bigcap_{i \in I} X_i$ es el* ***subgrupo*** *de G* ***generado*** *por el conjunto X, y lo denotaremos por $\langle X \rangle$. Si $X = \{x_1, \ldots, x_n\}$, entonces escribiremos $\langle x_1, \ldots, x_n \rangle$ en vez de $\langle X \rangle$ y decimos que $\{x_1, \ldots, x_n\}$ son los* ***generadores*** *de $\langle X \rangle$.*

Lema 5.7 *El subgrupo $\langle X \rangle$ consiste en todos los posibles productos finitos de la forma $x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}$, donde $x_j \in X$ e $i_j \in \mathbb{Z}$ para todo j.*

Ejemplo 5.8 *Es fácil comprobar que el grupo de permutaciones $\Sigma_3 = \langle (1, 2, 3), (1, 2) \rangle = \langle (1, 3, 2), (1, 3) \rangle = \langle (1, 3, 2), (2, 3), (1, 2, 3) \rangle$.*

Finalizamos esta sección viendo el concepto de grupo libre.

Definición 5.9 *Dado un conjunto X de símbolos, podemos construir el* ***grupo libre*** *F_X generado por X como sigue: para todo $x \in X$ definimos un correspondiente elemento "inverso" x^{-1}. Sea $X^{-1} := \{x^{-1} : x \in X\}$ disjunto con X y denotemos por $T := X \cup X^{-1}$. Podemos definir una* ***palabra*** *sobre X como cualquier producto de elementos de T. La palabra vacía es aquella que no consta de ningún símbolo. Si, en una palabra, a un elemento de X le sigue inmediatamente después su "inverso" en X^{-1}, entonces la palabra se puede simplificar omitiendo dicho par de elementos. Una palabra que no pueda ser simplificada la denominaremos* ***reducida****.*

Ejemplo 5.10 *Sea $X = \{a, b, c\}$. Entonces $T = \{a, a^{-1}, b, b^{-1}, c, c^{-1}\}$ y $ab^3c^{-1}ca^{-1}c$ es una palabra sobre X que se puede reducir a $ab^3a^{-1}c$.*

El grupo libre F_X se define como el conjunto formado por todas las palabras reducidas sobre X, siendo operación la concatenación (seguida de reducción si es necesaria). Queda claro que forma un grupo de orden infinito, siendo el elemento neutro la palabra vacía. Observemos que se trata de una construcción general sin restricciones, lo cual justifica el término "libre". Un hecho que será clave es que, dado un grupo G generado por un conjunto (finito) X, siempre podemos construir el grupo libre F_X sobre el conjunto X y un epimorfismo de F_X en G que envía cada generador de F_X al correspondiente generador de G.

Teorema 5.11 *Todo grupo G se puede expresar como imagen epiforma de un grupo libre.*

Este resultado nos permite expresar cada uno de los elementos de G como producto de sus generadores, pudiendo no ser única dicha expresión.

3. El software GAP

El proyecto `GAP` (Groups, Algorithms and Programming) empezó en noviembre de 1985 bajo la dirección del profesor Joachim Neubüser, en el *Lehrstuhl D für Mathematik*, RWTH-Aachen. Desde su jubilación en 1997, la coordinación del proyecto (ahora internacional) se lleva a cabo en St Andrews (Escocia). El software `GAP` es un sistema algebraico computacional, con especial énfasis en teoría de grupos, aunque también es útil para otras muchas ramas de las matemáticas y del álgebra discreta como anillos, cuerpos, espacios vectoriales, álgebras, estructuras combinatorias, etc. El programa `GAP` es un sistema de distribución libre, lo cual es una de las razones importantes por las que se utiliza en investigación y docencia.

Este software proporciona un lenguaje de programación, una librería de miles de funciones que implementan algoritmos algebraicos y grandes librerías de objetos algebraicos, como la librería `SmallGroups` o grupos de orden "pequeño". En The GAP Group (2016) puede encontrarse un manual extenso del uso del programa.

A continuación vamos a dar una breve descripción de los principales comandos que vamos a utilizar en el resto del trabajo. Señalar que todos los comandos en `GAP` deben ir terminados de un punto y coma para que se muestre su resultado, o de dos puntos y comas para que se ejecute pero no se muestre el resultado. Las permutaciones se pueden introducir con la notación en ciclos. Se puede ver el grupo generado por una serie de permutaciones con la instrucción `Group([])`, siendo los elementos entre los corchetes los generadores. El comando `GeneratorsOfGroup()` nos proporciona un sistema de generadores del grupo y su orden viene dado por la instrucción `Size()`. Veamos por ejemplo como introducir Σ_3:

```
gap> S3:=Group([(1,2,3), (1,2)]);
Group([ (1,2,3), (1,2) ])
gap> Size(S3);
6
gap> GeneratorsOfGroup(S3);
[(1,2,3), (1,2)]
```

El grupo libre generado por una serie de símbolos se puede obtener mediante la instrucción `FreeGroup()`, proporcionando los símbolos entre comillas. Una forma de construir homomorfismos entre dos grupos es con el comando `GroupHomomorphismByImages()`, que asigna unas imágenes concretas del homomorfismo a los generadores. La imagen de un elemento la calculamos con `Image()` y un representante de las preimágenes de un elemento lo podemos obtener con `PreImagesRepresentative()`:

```
gap> F:=FreeGroup("a", "b");
<free group on the generators [ a, b ]>
gap> Size(F);
infinity
gap> hom:=GroupHomomorphismByImages(F, S3, GeneratorsOfGroup(F),
> GeneratorsOfGroup(S3));
[ a, b ] -> [ (1,2,3), (1,2) ]
gap> s:=PreImagesRepresentative(hom, (1,2,3)*(1,2));
b^-1*a^-1
gap> Image(hom, s)=(1,2,3)*(1,2);
true
```

Los comandos anteriores nos han proporcionado un homomorfismo $f : F \longrightarrow \Sigma_3$ tal que $f(a) = (1,2,3)$ y $f(b) = (1,2)$. Es evidente, por definición, que $f(ab) = f(a)f(b) = (1,2,3)(1,2)$, luego ab es antiimagen de $(1,2,3)(1,2)$. Sin embargo, el comando `PreImagesRepresentative` nos ha proporcionado otra antiimagen del elemento ab, que es el elemento $b^{-1}a^{-1}$, ya que la preimagen no es única. Además, notemos la relación de los comandos anteriores con el Teorema 5.11.

4. Modelización y resolución de algunos puzzles y rompecabezas

En esta sección vamos a modelizar algunos puzzles y rompecabezas y analizaremos su resolubilidad con GAP. En la referencia Scherphuis, J. se encuentra una web donde es posible jugar a algunos de estos juegos si se dispone de un navegador que soporte *JavaScript*.

4.1. Cubo de Rubik

Aunque cualquier persona sabría instintivamente cómo jugar a este popular puzzle, vamos a describir su funcionamiento y el objetivo general del juego. El **cubo de Rubik** es un rompecabezas tridimensional con forma de cubo, inventado por el escultor y profesor de arquitectura húngaro Erno Rubik en 1974. Fue comercializado por Ideal Toy Company en 1980. Hasta enero de 2009 se habían vendido unos 350 millones de cubos en todo el mundo, convirtiéndolo no solo en el rompecabezas más vendido, sino que es considerado en general el juguete más vendido del mundo.

El cubo de Rubik $3\times3\times3$ está formado por 26 piezas y posee seis colores uniformes, tradicionalmente blanco, rojo, azul, naranja, verde y amarillo. Un mecanismo de ejes permite a cada cara girar independientemente, mezclando así los colores, como se puede observar en la siguiente figura. Para resolver el rompecabezas, cada cara debe volver a quedar de un solo color.

Figura 2 – Cubo de Rubik en posición inicial.

Figura 3 – Cubo de Rubik en posición movida.

Lo interesante de este famoso rompecabezas son las matemáticas existentes en él. El álgebra y concretamente la teoría de grupos nos permite realizar un estudio matemático más exhaustivo de este juego (puede consultarse Esteban Romero, R., 2013). Nosotros vamos a centrarnos en ver qué posiciones son resolubles y, en estos casos,

cómo resolverlo con GAP. Parte del desarrollo que presentamos a continuación está basado en el diverso material que se puede encontrar en The GAP Group (2016). Para comenzar, vamos a numerar las caras del cubo como se indica en la siguiente figura:

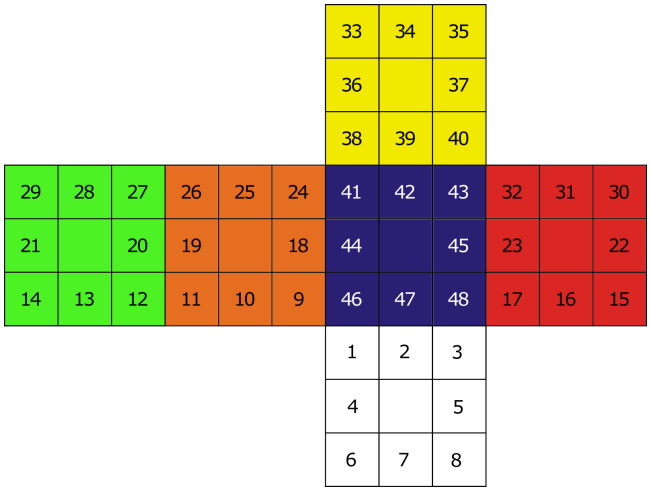


Figura 4 – Numeración inicial del cubo de Rubik.

Cada movimiento va a conllevar implícitamente una permutación. Por ejemplo, el giro de 90^{o} de la cara blanca (B) en sentido horario quedaría de la siguiente forma:

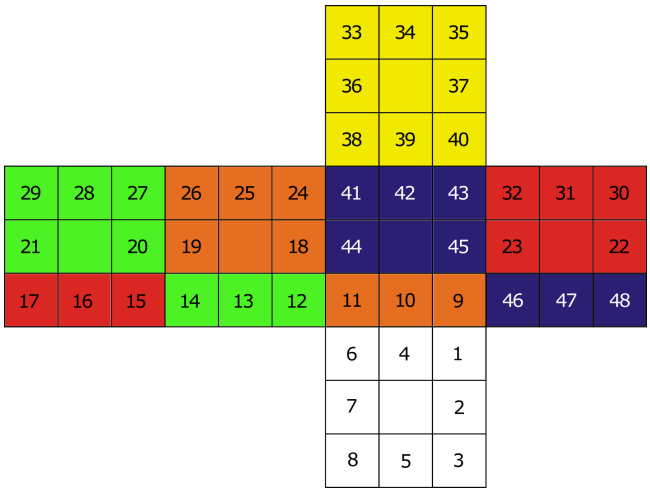


Figura 5 – Numeración del cubo tras un giro de 90^{o} en sentido horario de la cara blanca.

Si observamos, la posición 1 se ha desplazado a la posición 3, la cual a su vez se ha desplazado a la posición 8, la 8 ahora está en la posición 6 y la 6 en el lugar de la posición 1. Esto se expresa matemáticamente con el ciclo $(1, 3, 8, 6)$. Si continuamos con el resto de desplazamientos que se han producido al mover esta cara, podemos expresar dicho movimiento con la permutación siguiente

$$(1, 3, 8, 6)(2, 5, 7, 4)(9, 48, 15, 12)(10, 47, 16, 13)(11, 46, 17, 14).$$

En general, la permutación asociada a estos movimientos la podemos obtener con el comando `PermListList`, el cual le damos como primer input un vector representando la situación actual y como segundo input otro vector con la situación original [1..48]:

```
gap> B:=PermListList([6,4,1,7,2,8,5,3,12,13,14,15,16,17,48,47,46,18,19,
> 20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,
> 41,42,43,44,45,11,10,9],[1..48]);
(1,3,8,6)(2,5,7,4)(9,48,15,12)(10,47,16,13)(11,46,17,14)
```

Con este movimiento podemos obtener también por ejemplo los giros de 180^{o} de esa cara como B^2 o los giros de 90^{o} en sentido antihorario como B^3 o B^{-1}.

El grupo del cubo de Rubik tiene seis generadores, que se corresponden con los movimientos descritos en cada una de las seis caras, ya que estos son los movimientos básicos que nos generan los demás. Vamos a llamar 'B' al giro de 90^{o} en sentido horario de la cara con el centro blanco y los análogos movimientos para el resto de caras los denotaremos por las iniciales de sus colores, los cuales quedarían así:

```
gap> B:=(1, 3, 8, 6)(2,5,7,4)(9,48,15,12)(10,47,16,13)(11,46,17,14);;
gap> V:=(6,15,35,26)(7,22,34,19)(8,30,33,11)(12,14,29,27)(13,21,28,20);;
gap> N:=(1,12,33,41)(4,20,36,44)(6,27,38,46)(9,11,26,24)(10,19,25,18);;
gap> Az:=(1,24,40,17)(2,18,39,23)(3,9,38,32)(41,43,48,46)(42,45,47,44);;
gap> R:=(3,43,35,14)(5,45,37,21)(8,48,40,29)(15,17,32,30)(16,23,31,22);;
gap> Am:=(24,27,30,43)(25,28,31,42)(26,29,32,41)(33,35,40,38)
> (34,37,39,36);;
```

Cualquier movimiento del cubo está generado por las permutaciones anteriores. El grupo generado por dichos movimientos lo vamos a llamar `rubik` y notemos que es un subgrupo de Σ_{48}. Con los comandos de `GAP`, la instrucción que utilizaríamos sería la siguiente:

```
gap> rubik:=Group([B, V, N, Az, R, Am]);;
gap> Size(rubik);
43252003274489856000
gap> PrintFactorsInt(Size(rubik));
2^27*3^14*5^3*7^2*11
```

El último comando nos permite hallar la descomposición en factores primos del número total de movimientos posibles en el cubo de Rubik. Por citar una curiosidad relacionada con el tamaño del grupo, J. A. Paulos dijo en Innumeracy: "*Ideal Toy Company stated on the package of the original Rubik cube that there were more than three billion possible states the cube could attain. It's analogous to Mac Donald's proudly announcing that they've sold more than 120 hamburgers.*"

Una vez hemos modelizado el grupo de simetrías del cubo, vamos a centrarnos en cómo resolverlo, esto es, cómo invertir una sucesión cualquiera de movimientos. Por ejemplo, supongamos que tenemos el siguiente cubo por resolver.

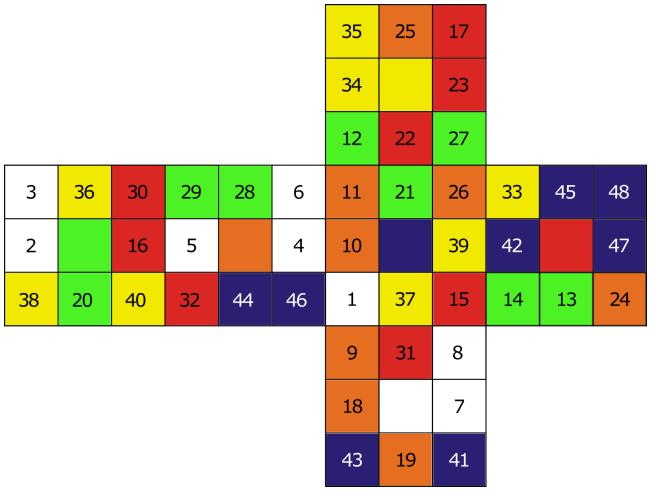


Figura 6 – Numeración del cubo de Rubik a resolver.

La idea clave para su resolución es el Teorema 5.11 sobre la construcción del grupo libre. Vamos a construir un homomorfismo de grupos entre el grupo libre generado por $X = \{B, V, N, Az, R, Am\}$ y el grupo del cubo de Rubik:

```
gap> F:=FreeGroup("B", "V", "N", "Az", "R", "Am");;
gap> hom:=GroupHomomorphismByImages(F, rubik, GeneratorsOfGroup(F),
> GeneratorsOfGroup(rubik));
[ B, V, N, Az, R, Am ] ->
[ (1,3,8,6)(2,5,7,4)(9,48,15,12)(10,47,16,13)(11,46,17,14),
  (6,15,35,26)(7,22,34,19)(8,30,33,11)(12,14,29,27)(13,21,28,20),
  (1,12,33,41)(4,20,36,44)(6,27,38,46)(9,11,26,24)(10,19,25,18),
  (1,24,40,17)(2,18,39,23)(3,9,38,32)(41,43,48,46)(42,45,47,44),
  (3,43,35,14)(5,45,37,21)(8,48,40,29)(15,17,32,30)(16,23,31,22),
  (24,27,30,43)(25,28,31,42)(26,29,32,41)(33,35,40,38)(34,37,39,36) ]
```

Escribiendo en GAP el vector con la configuración actual (Figura 6), podemos obtener la permutación correspondiente tal y como hemos descrito anteriormente:

```
gap> PosicionFinal:=PermListList([9, 31, 8, 18, 7, 43, 19, 41, 46, 44, 32,
> 40, 20, 38, 24, 13, 14, 4, 5, 16, 2, 47, 42, 6, 28, 29, 30, 36, 3, 48,
> 45, 33, 35, 25, 17, 34, 23, 12, 22, 27, 11, 21, 26, 10, 39, 1, 37, 15],
> [1..48]);
(1,46,9)(2,21,42,23,37,47,22,39,45,31)(3,29,26,43,6,24,15,48,30,27,40,12,
38,14,17,35,33,32,11,41,8)(4,18)(5,19,7)(10,44)(13,16,20)(36,28,25,34);;
```

Queremos recuperar una combinación de movimientos que nos ha dado lugar a esta situación. Se trata de hallar una preimagen del homomorfismo de PosicionFinal.

```
gap> t:=PreImagesRepresentative(hom, PosicionFinal);
B*R^-1*B*V*R^-1*V^-1*N^-1*R^-1*V^-1*R*N*B*V^-1*N*V*Az*Am*N^2*Am^-1*Az^3*
R*Az*R^-1*B^-1*Az^-2*R^-1*B^-1*R*B^-1*Az*B*Az*N^-1*Az^-1*N*Az^-1*B^-2*
R^-1*B*R*Az*B^-1*Az*B*Az*N^-1*Az^-1*N*B^-1*Az^-1*B^-1*N^-1*Az*N*Az^-1*
B^-1*Az^-1*B*Az*R*Az^-1*R^-1*Az*B*Az*B^-1*Az^-1*B^-1*Az*B*N*B^-1*N^-1*
Az^-1*B*N^-1*Az^-1*B^-1*Az*B*N
gap> Image(hom, t)=PosicionFinal;
true
```

Así, hemos obtenido unos movimientos que nos han llevado desde la posición inicial del cubo a la actual. Para resolver el cubo y llegar de nuevo a la posición original, tendremos que ir ejecutando los inversos de dichos movimientos, desde el último (el movimiento N) hasta el primero (el movimiento B). Es decir, empezaríamos realizando el movimiento N^{-1} seguido de B^{-1}, posteriormente Az^{-1}, B, etc.

Por supuesto, esta representación no tiene que ser la más corta, pues el homomorfismo no es inyectivo (i.e., existen diferentes preimágenes como ya hemos comentado). Debido a los métodos aleatorios utilizados en la función GroupHomomorphismByImages, podemos recibir otras soluciones más cortas si ensayamos en diferentes sesiones de GAP.

Observemos que los seis generadores del cubo fijan los centros de cada cara. Podríamos considerar otros dos movimientos del cubo que fijan las esquinas y desplazan centros: el giro de 90^{o} de la fila central que mueve el centro de la cara naranja al centro de la cara azul y reordenando los centros (para no perder nuestra representación de la Figura 4), y análogamente el giro de 90^{o} que traslada el centro de la cara blanca al centro de la cara azul y reordenando centros. Realmente estos dos movimientos se pueden escribir como combinaciones de los seis generadores descritos, pero en la práctica pueden ayudarnos a encontrar antes la solución, ahorrándonos movimientos. Llamemos m_1 al primero de ellos. Vamos a comprobar efectivamente que $m_1 = B^3 * Am$ y que el grupo del cubo de Rubik también está generado por $\langle B, V, N, Az, R, Am, m_1\rangle$.

```
gap> m1:=PermListList([ 3, 5, 8, 2, 7, 1, 4, 6, 48, 47, 46, 9, 10, 11, 12,
> 13, 14, 18, 19, 20, 21, 22, 23, 43, 42, 41, 24, 25, 26, 27, 28, 29,
> 38, 36, 33, 39, 34, 40, 37, 35, 32, 31, 30, 44, 45, 17, 16, 15 ],
> [1..48]);
(1,6,8,3)(2,4,7,5)(9,12,15,48)(10,13,16,47)(11,14,17,46)(24,27,30,43)
(25,28,31,42)(26,29,32,41)(33,35,40,38)(34,37,39,36)
gap> m1=B^3*Am;
true
gap> Size(rubik)=Size(Group([B, V, N, Az, R, Am, m1]));
true
```

Existen variaciones con diversos números de cuadrados por cara como el $2 \times 2 \times 2$ (o *el cubo de bolsillo*) o el $4 \times 4 \times 4$ (o *la venganza de Rubik*), los cuales podrían tratarse de una manera análoga a la aquí presentada. Por otro lado, también existen otras variantes del cubo de Rubik donde se modifica la estructura geométrica

o los ángulos de giro en los movimientos, como es el caso por ejemplo del cubo *Skewb*, el cual tratamos en la siguiente sección, o el *Pyraminx* con forma de tetaedro.

Finalmente, nos planteamos una pregunta alternativa pero de interés similar a la resolución del cubo: ¿es posible llegar desde la posición inicial a la posición que aparece en la siguiente figura?

Figura 7 – Cubo de huevos fritos.

Se trata de una posición del cubo en la cual cada cara posee forma de *huevo frito*, esto es, el cuadrado central amarillo esta rodeado de cuadrados blancos y el resto de cuadrados centrales están rodeados a su vez de cuadrados del mismo color pero distinto del central. Esta cuestión es analizada en Sangroniz, J. (2015) de forma divulgativa. Nuestro objetivo es contestar a dicha pregunta de una manera práctica. Para ello, debemos ver qué numeración se corresponde con la figura anterior, donde vamos a asumir que los colores de las *claras* que aparecen son blanco, verde y rojo. Supongamos que al cubo resuelto de la Figura 2 podemos extraerle el mecanismo interno que une las seis piezas centrales de cada cara, que es una cruceta en la que tres segmentos se cortan perpendicularmente en sus puntos medios. Ahora recolocamos esta estructura para rellenar los seis vacíos que se han originado de la siguiente manera: el segmento amarillo-blanco lo volvemos a colocar en su mismo agujero pero en sentido opuesto, el segmento verde-azul lo posicionamos en el agujero de las caras naranja-rojo de forma que el extremo verde quede en la cara roja, y finalmente el segmento rojo-naranja lo colocamos en el último agujero de manera que el extremo rojo quede en la cara verde. Si observamos, la numeración resultante queda así:

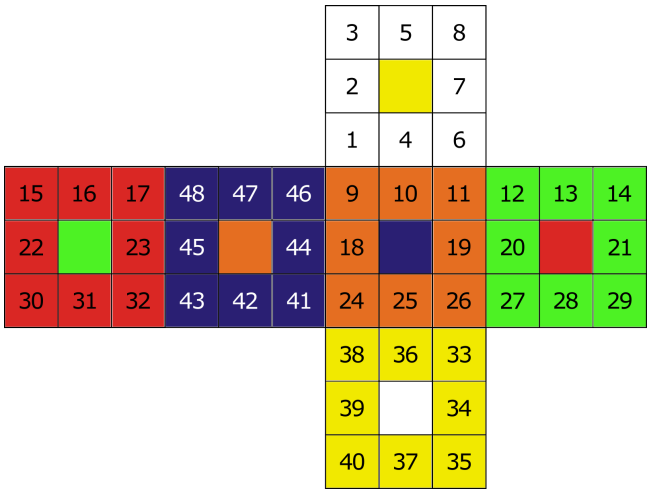


Figura 8 – Numeración del cubo de huevos fritos de la Figura 7.

Basta ahora con preguntarle a GAP si existe una antiimagen de la permutación asociada a dicha numeración por el homomorfismo que teníamos contruido:

```
gap> huevo_frito1:=PermListList([38,36,33,39,34,40,37,35,41,
> 42,43,32,31,30,29,28,27,44,45,23,22,21,20,46,47,48,17,
> 16,15,14,13,12,3,5,8,2,7,1,4,6,9,10,11,18,19,24,25,26],[1..48]);
(1,38)(2,36)(3,33)(4,39)(5,34)(6,40)(7,37)(8,35)(9,41)(10,42)
(11,43)(12,32)(13,31)(14,30)(15,29)(16,28)(17,27)(18,44)(19,45)
(20,23)(21,22)(24,46)(25,47)(26,48)
gap> PreImagesRepresentative(hom, huevo_frito1);
fail
```

Esto quiere decir que es imposible realizar una serie de movimientos que nos lleven de la posición inicial a la de la Figura 7. Sin embargo, según se afirma en Sangroniz, J. (2015), existen 8 posiciones con *huevos fritos* en todas las caras que sí son alcanzables. Una de ellas es la que aparece en la siguiente imagen:

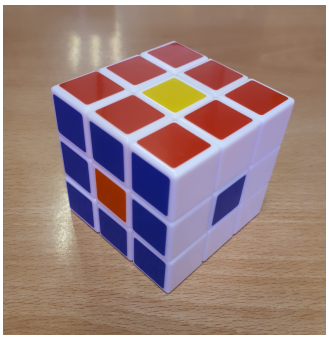


Figura 9 – Otro cubo de huevos fritos.

Para probar que efectivamente esta posición sí es posible, calculamos su numeración correspondiente con el mismo procedimiento descrito anteriormente y comprobamos si posee antiimagen:

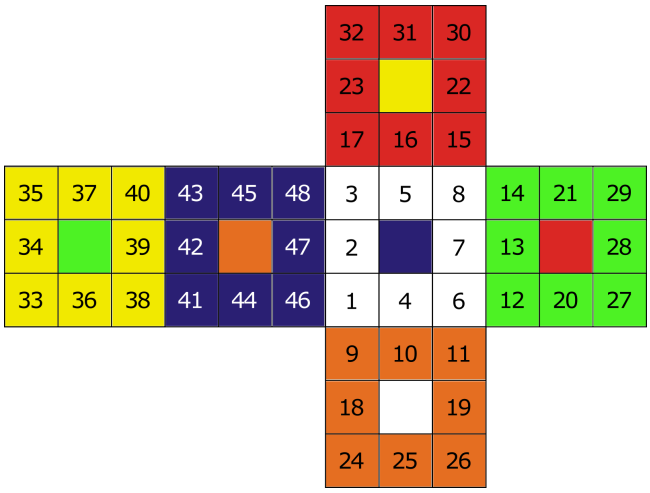


Figura 10 – Numeración de unos *huevos fritos* resolubles.

```
gap> huevo_frito2:=PermListList([9,10,11,18,19,24,25,26,46,44,
> 41,38,36,33,27,20,12,47,42,39,34,28,13,48,45,43,40,37,35,
> 29,21,14,32,31,30,23,22,17,16,15,3,5,8,2,7,1,4,6],[1..48]);
(1,46,9)(2,44,10)(3,41,11)(4,47,18)(5,42,19)(6,48,24)(7,45,25)
(8,43,26)(12,17,38)(13,23,36)(14,32,33)(15,40,27)(16,39,20)
(21,31,34)(22,37,28)(29,30,35)
gap> PreImagesRepresentative(hom, huevo_frito2);
Am*R*B*V*R^-1*V^-1*Az*R*Am^-1*R^-1*B^-1*V^-1*N^-2*V*Am*N^-1*Am^-1*B*R*
```

```
Az^-1*R^-1*Az^-1*B*Az^-2*B^-1*Az^-1*B*N*B*N^-1*Az^-1*B^-1*Az*B^-1*Az*
N^-1*Az^-1*N*Az^-1*B^-1*Az*B^-1*Az*N*B*N^-1*B^-1*Az^-1*N^-1*B^-1*Az^-1*
B*Az*N*B^-2*Az^-1*R^-1*B^-1*R*B*Az*B^-1*Az*B*N*B^-1*N^-1*Az^-1*B*N^-1*
Az^-1*B^-1*Az*B*N^2*V*B*V^-1*B^-1*N^-1*B^-1*Az^-1*B^-1*R^-1*B^-2*R*Az*
N*V*B^-1*V^-1*N^-1*B
```

Mencionar también que los movimientos del cubo de Rubik *actúan* sobre las orientaciones de las 8 esquinas tricolores y las 12 piezas bicolores, dando lugar a construcciones abstractas como productos *orlados*. Para más información sobre este tema véase Joyner, D. (2008) y Mulholland, J.

4.2. Cubo Skewb

El **cubo Skewb** es un rompecabezas mecánico del tipo del cubo de Rubik, compuesto por piezas que pueden rotar y cambiar de posición. Su nombre proviene de las palabras inglesas *skew* (oblicuo, torcido) y *cube* (cubo). Fue inventado por el periodista inglés Tony Durham y comercializado por Uwe Meffert, en un principio con el nombre de *Pyraminx Cube*. Douglas Hofstadter acuñó la palabra "Skewb" en un artículo de la Scientific American en julio de 1982.

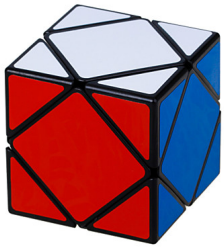


Figura 11 – Skewb en posición inicial.

Figura 12 – Skewb en movimiento.

Mientras que el cubo de Rubik está cortado por seis planos paralelos a las caras, el Skewb está cortado por solo cuatro planos perpendiculares a las diagonales principales. Cada uno de estos planos divide al cubo en dos partes iguales. El Skewb está formado por dos clases de piezas: ocho esquinas piramidales, con tres colores cada una, y seis centros monocromos de forma cuadrada. En este cubo los giros se le realizan a las esquinas en sentido horario (en 120°) en vez de a las caras y se desplaza el "medio cubo oblicuo" correspondiente a dicha esquina (ver Figura 12). Por tanto, cada movimiento desplaza cuatro esquinas y tres centros.

Para comenzar su modelado matemático, realizamos primeramente un etiquetado de las piezas tal y como indicamos a continuación.

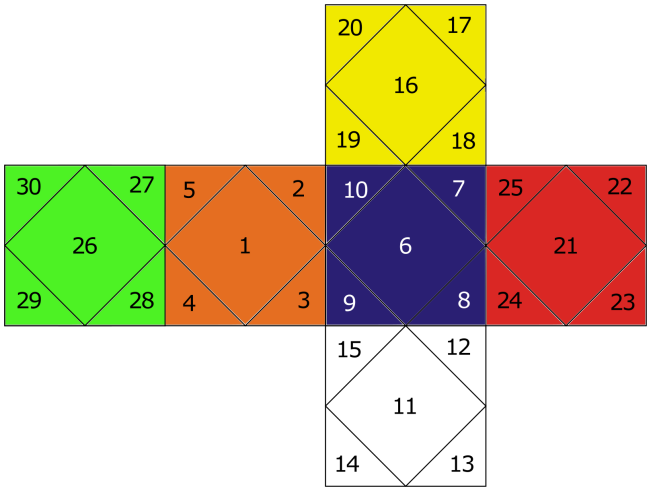


Figura 13 – Numeración del cubo Skewb.

Posteriormente, debemos identificar los movimientos básicos que van a generar todos los demás, es decir, que van a generar el grupo del cubo Skewb. En este caso, los 8 giros de 120^{o} de cada esquina del cubo son suficientes. Por ejemplo, el giro de 120^{o} en sentido horario de la esquina azul-amarillo-rojo (AzAmR) se corresponde con la siguiente imagen y con la permutación que le sigue a continuación.

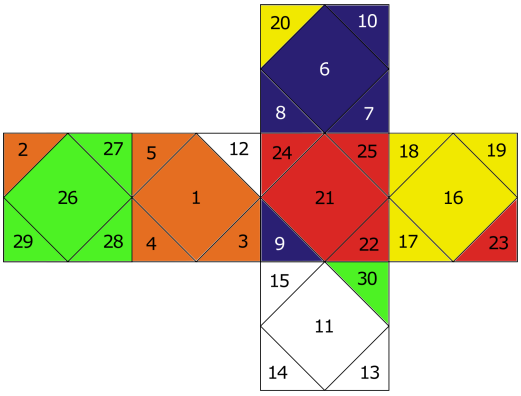


Figura 14 – Giro de la esquina azul-amarillo-rojo.

```
gap> AzAmR:=PermListList([1,12,3,4,5,21,25,22,9,24,11,30,13,
> 14,15,6,10,7,8,20,16,19,23,17,18,26,27,28,29,2], [1..30]);
(2,30,12)(6,16,21)(7,18,25)(8,19,22)(10,17,24)
```

El resto de movimientos básicos son los siguientes:

```
gap> AmVR := (5,13,7)(16,26,21)(17,30,22)(18,27,23)(20,29,25);;
gap> AmNV := (1,26,16)(2,28,17)(4,30,19)(5,27,20)(10,14,22);;
gap> AmAzN := (1,16,6)(2,19,10)(3,20,7)(5,18,9)(15,27,25);;
gap> AzRB := (3,18,29)(6,21,11)(7,23,15)(8,24,12)(9,25,13);;
gap> BRV := (4,8,17)(11,21,26)(12,22,28)(13,23,29)(14,24,30);;
gap> AzBN := (1,6,11)(2,8,14)(3,9,15)(4,10,12)(19,24,28);;
gap> BVN := (1,11,26)(3,13,27)(4,14,28)(5,15,29)(9,23,20);;
gap> skewb := Group([AzAmR, AmVR, AmNV, AmAzN, AzRB, BRV, AzBN, BVN]);;
gap> Size(skewb);
37791360
gap> PrintFactorsInt(Size(skewb));
2^7*3^10*5
```

Supongamos ahora que queremos resolver la posición de la Figura 15.

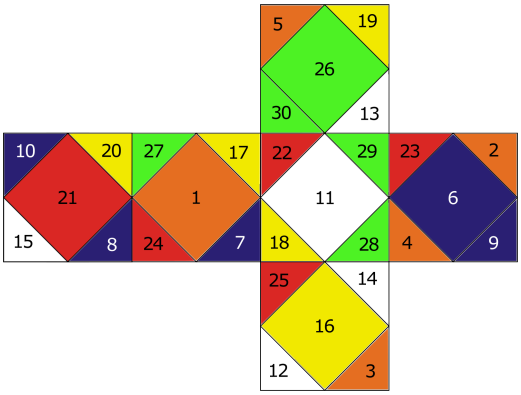


Figura 15 – Cubo Skewb a resolver.

Al igual que en el cubo de Rubik, construimos un homomorfismo apropiado entre el grupo que generan estas permutaciones y el grupo libre generado por 8 símbolos, y resolveremos el cubo de la Figura 15.

```
gap> F:=FreeGroup("AzAmR", "AmVR", "AmNV", "AmAzN", "AzRB", "BRV",
> "AzBN", "BVN");;
gap> hom:=GroupHomomorphismByImages(F, skewb, GeneratorsOfGroup(F),
> GeneratorsOfGroup(skewb));
[ AzAmR, AmVR, AmNV, AmAzN, AzRB, BRV, AzBN, BVN ] ->
[ (2,30,12)(6,16,21)(7,18,25)(8,19,22)(10,17,24),
  (5,13,7)(16,26,21)(17,30,22)(18,27,23)(20,29,25),
  (1,26,16)(2,28,17)(4,30,19)(5,27,20)(10,14,22),
  (1,16,6)(2,19,10)(3,20,7)(5,18,9)(15,27,25),
  (3,18,29)(6,21,11)(7,23,15)(8,24,12)(9,25,13),
  (4,8,17)(11,21,26)(12,22,28)(13,23,29)(14,24,30),
  (1,6,11)(2,8,14)(3,9,15)(4,10,12)(19,24,28),
  (1,11,26)(3,13,27)(4,14,28)(5,15,29)(9,23,20) ]
gap> PosicionFinal:=PermListList([ 1, 17, 7, 24, 27, 11, 29, 28, 18, 22,
> 16, 14, 3, 12, 25, 26, 19, 13, 30, 5, 6, 2, 9, 4, 23, 21, 20, 8, 15,
> 10 ], [1..30]);
(2,22,10,30,19,17)(3,13,18,9,23,25,15,29,7)(4,24)(5,20,27)(6,21,26,16,11)
(8,28)(12,14)
gap> t:=PreImagesRepresentative(hom, PosicionFinal);
AmAzN*AzAmR^-1*BRV^-1*AzAmR*AmNV*AzAmR*AmVR^-2*AzAmR^-1*AmVR*
AzAmR*AmVR^-1*AzAmR^-1*AmVR^-1*AzAmR^-1*AmNV^-1*AzAmR^-1*AmVR*
AmNV*AzAmR*AmNV
gap> Image(hom, t)=PosicionFinal;
true
```

El primer movimiento que tendríamos que realizar para volver a la posición original es el giro de 120º antihorario de la esquina amarillo-naranja-verde (es decir, $AmNV^{-1}$, posteriormente otro giro antihorario de 120º del vértice azul-amarillo-rojo (esto es $AzAmR^{-1}$), etc.

Por otro lado, al igual que con el cubo de Rubik, podemos analizar si en el cubo Skewb es posible realizar una serie de movimientos que nos construyan en todas las caras "huevos fritos". Por ejemplo, estudiemos si los cubos de huevos fritos de las siguientes imágenes son posibles. Notar que ambos se corresponden con los de las Figuras 8 y 10.

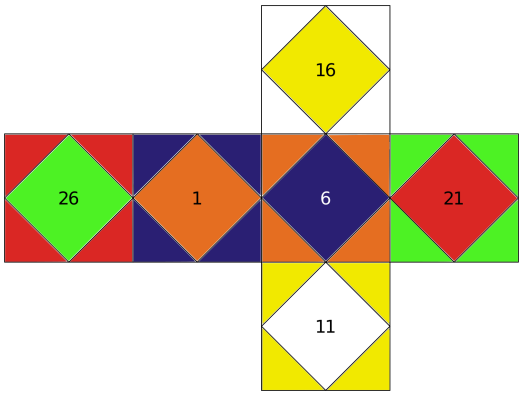


Figura 16 – Cubo Skewb de huevos fritos.

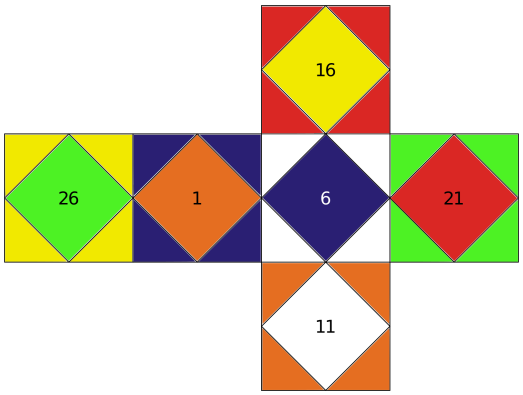


Figura 17 – Otro cubo Skewb de huevos fritos.

Un dato relevante es que, a diferencia del cubo de Rubik, en el cubo Skewb los centros de cada cara sí que están numerados, pues son afectados por los movimientos básicos. Por consiguiente, ver si es posible las posiciones del cubo Skewb de las imágenes anteriores es equivalente a ver si es posible realizar las permutaciones $(1,6)(11,16)(21,26)$ y $(1,6,11)(16,21,26)$, respectivamente. Señalar que podemos obviar la numeración de las esquinas, pues en este caso es equivalente ver cómo se han movido los centros a ver cómo se han movido las esquinas.

```
gap> PreImagesRepresentative(hom, (1,6)(11,16)(21,26));
fail
gap> PreImagesRepresentative(hom, (1,6,11)(16,21,26));
AmVR^-1*AzAmR^-1*AmNV^-1*AzAmR^-1*AmVR*AmNV*AzAmR*AmNV*AmVR*AzAmR*AmVR^
-1*AzAmR^-1*AmVR^2*AzAmR^-1*AmVR^-1*AzAmR*AmNV^-1*AmVR^-1*AzAmR*AmNV^-1*
BRV^-1*AmNV^-1
```

Como podemos observar, solo podemos construir el segundo de los cubos de huevos fritos.

4.3. TopSpin

El puzzle **TopSpin** fue inventado por Ferdinand Lammertink en el año 1989. Consiste en 20 piezas redondas formando una cadena con forma de óvalo, pudiéndose desplazar todas a lo largo de la cadena. Cada pieza lleva representado un número del 1 al 20. También contiene un disco giratorio que contiene siempre a 4 piezas adyacentes, las cuales gira en orden inverso con un giro de 180^{o}. Como se ha comentado en la introducción de esta sección, se puede jugar a este juego online en la web de Scherphuis, J. Queremos mencionar que parte de este análisis del TopSpin con `GAP` también se puede encontrar en Long, C.

Observemos que los dos únicos movimientos posibles son los siguientes: el primero corresponde al giro de 180^{o} del disco, y el segundo al desplazamiento una unidad en sentido horario de todas las fichas. Además, si observamos detenidamente, el disco posee una zona inferior y otra superior distinguibles.

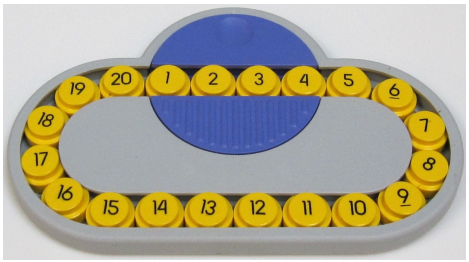


Figura 18 – TopSpin.

Figura 19 – Movimiento del disco.

El propósito final del juego es colocar una serie de piezas desordenadas aleatoriamente en el orden original numérico. Una pregunta que surge naturalmente es si todas las posibles posiciones de las piezas son alcanzables y cómo llevarlas a cabo. Posteriormente, veremos que la respuesta es afirmativa.

Para su modelado matemático, vamos a etiquetar al extremo superior e inferior del disco como las "piezas" 21 y 22 respectivamente. De nuevo, se trata de construir en `GAP` un homomorfismo de grupos entre el grupo libre generado por dos letras y el grupo de permutaciones generado por los dos movimientos posibles.

```
gap> GiroRueda:=PermListList([4,3,2,1,5,6,7,8,9,10,11,12,13,14,15,16,17,
> 18,19,20,22,21], [1..22]);
(1,4)(2,3)(22,21)
gap> Traslacion:=PermListList([20,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
> ,17,18,19,21,22], [1..22]);
(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)
```

Definimos ahora el homomorfismo con el grupo libre, donde x indicará el giro e y la traslación. Por otro lado, vamos a ver también que, sorprendemente, tras una serie de movimientos podemos invertir el disco 180^{o} de manera que la secuencia numérica quede en el mismo orden, i.e., que la permutación $(21, 22)$ tiene antiimagen.

```
gap> TopSpin:=Group(GiroRueda, Traslacion);;
gap> F:=FreeGroup("x","y");;
gap> hom:=GroupHomomorphismByImages(F,TopSpin,GeneratorsOfGroup(F),
> GeneratorsOfGroup(TopSpin));;
gap> t:=PreImagesRepresentative(hom,(21,22));
y*x^-1*y^2*x*y^-1*x*y*x*y^-1*x*y^-2*x*y^2*x*y^-1*x*y*x*y*x*y^-2*x^-1*
y*x^-1*y^-2*x^-1
gap> Image(hom, t)=(21, 22);
true
```

Veamos ahora que podemos llegar a cualquier ordenación de los números del 1 al 20 (no de las piezas 21 y 22). Esto es equivalente a ver que Σ_{20} es un subgrupo del grupo TopSpin, lo cual se puede comprobar en `GAP` con el comando `IsSubgroup`.

```
gap> IsSubgroup(TopSpin, SymmetricGroup(20));
true
gap> Size(TopSpin);
4865804016353280000
gap> PrintFactorsInt(Size(TopSpin));
2^19*3^8*5^4*7^2*11*13*17*19
```

Por tanto todas las permutaciones del 1 al 20 son posibles y como el disco puede girarse 180^{o} manteniendo un orden dado, podemos comprobar que el orden del grupo TopSpin es $20! \cdot 2 = 4,865,804,016,353,280,000$. De hecho, dicho grupo tiene estructura de *producto directo* (ver definición en Hungerford, T. W. ,1974) de los grupos Σ_{20} y $\langle(21,22)\rangle$.

Unas variaciones del puzzle TopSpin que surgen naturalmente son cuando cambiamos el número de piezas o cuando el disco giratorio intercambia otro número de piezas distinto de 4. ¿Seguirán siendo posibles todas las permutaciones de piezas? A continuación vamos a analizar algunas de estas variaciones, donde consideraremos n piezas numeradas del 1 al n y k piezas giradas 180^{o} por el disco.

Comencemos estudiando el puzzle del tipo TopSpin correspondiente a $n = 19$ y $k = 4$. Notemos que la parte superior e inferior del disco pasan ahora a numerarse por 20 y 21. Para ver si siguen siendo posibles todas las permutaciones de las 19 piezas, basta con reescribir en `GAP` los nuevos movimientos y ver si el grupo resultante contiene como subgrupo al grupo simétrico de 19 elementos.

```
gap> Traslacion:=PermListList([19,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
> ,17,18,20,21], [1..21]);
(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19)
gap> GiroRueda:=PermListList([4,3,2,1,5,6,7,8,9,10,11,12,13,14,15,16,17,
> 18,19,21,20], [1..21]);
(1,4)(2,3)(20,21)
gap> TopSpin_19_4:=Group(GiroRueda, Traslacion);;
gap> IsSubgroup(TopSpin_19_4, SymmetricGroup(19));
false
```

Por tanto no todas las permutaciones son posibles en este caso. Esto no debería sorprendernos, ya que el movimiento de traslación y el movimiento de rotación del disco sobre las piezas del 1 al 19 provocan permutaciones pares sobre dichas piezas, luego no podemos construir permutaciones impares. Veremos que sí que contiene como subgrupo al grupo A_{19} y que, además, la trasposición $(20,21)$ sigue teniendo antiimagen, es decir, que sigue siendo posible girar el disco 180^{o} manteniendo el orden inicial del resto de piezas:

```
gap> IsSubgroup(TopSpin_19_4, AlternatingGroup(19));
true
gap> F:=FreeGroup("x", "y");;
gap> hom:=GroupHomomorphismByImages(F, TopSpin_19_4, GeneratorsOfGroup(F),
> GeneratorsOfGroup(TopSpin_19_4));
[ x, y ] -> [ (1,4)(2,3)(20,21), (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,
17,18,19) ]
gap> PreImagesRepresentative(hom, (20, 21));
y*x^-1*y^2*x*y^-1*x*y*x*y^-1*x*y^-2*x*y^2*x*y^-1*x*y*x*y*x*y^-2*x^-1*y*
x^-1*y^-2*x^-1
```

Se puede ver que la estructura del grupo ahora es el producto directo de los grupos A_{19} y $\langle(20,21)\rangle$. Comprobémoslo viendo que este grupo está contenido en el grupo TopSpin y que sus órdenes son iguales. Para ello, usaremos la instrucción `DirectProduct()` que nos calcula el producto directo de dos grupos.

```
gap> IsSubgroup(TopSpin_19_4, DirectProduct(AlternatingGroup(19),
> Group([(20,21)])));
true
gap> Order(TopSpin_19_4)=Order(AlternatingGroup(19))*Order(
>Group([(20,21)]));
true
```

En Kaufmann, S. (2011) y Mulholland, J., puede encontrarse un análisis más general cuando se consideran cualquier n y cualquier k en el puzzle TopSpin. En particular, si $k = 4$ y $n \geq 6$, entonces el grupo resultante cuando n es par tiene estructura de producto directo de Σ_n y $\langle R\rangle$, siendo R la trasposición correspondiente al giro 180^{o} del disco. Cuando n es impar, la estructura es de producto directo entre A_n y $\langle R\rangle$. Los casos $n \in \{4,5\}$ son excepciones, ya que aparecen los grupos de rotaciones y simetrías de un cuadrado y un pentágono, i.e., los grupos *dihédricos* (ver definición en Hungerford, T. W., 1974) de orden 8 y 10, respectivamente.

Pasemos ahora a estudiar el caso en el que modificamos el número de piezas giradas por el disco. Por ejemplo, analicemos el puzzle correspondiente a $n = 20$ y $k = 3$. Comencemos viendo si son posibles todas las permutaciones de las 20 primeras piezas:

```
gap> GiroRueda:=PermListList([3,2,1,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
> 18,19,20,22,21], [1..22]);
(1,3)(21,22)
gap> Traslacion:=PermListList([20,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
> ,17,18,19,21,22], [1..22]);
(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)
gap> TopSpin_20_3:=Group(GiroRueda, Traslacion);;
gap> IsSubgroup(TopSpin_20_3, SymmetricGroup(20));
false
```

De nuevo no deberíamos sorprendernos, ya que el giro del disco provoca el intercambio de dos piezas que son siempre dos números pares o dos impares. Por tanto, el disco *actúa* permutando las piezas pares entre sí y las impares entre sí. Comprobemos por ejemplo que no podemos intercambiar las piezas 1 y 2.

```
gap> F:=FreeGroup("x", "y");;
gap> hom:=GroupHomomorphismByImages(F, TopSpin_20_3, GeneratorsOfGroup(F),
> GeneratorsOfGroup(TopSpin_20_3));
[ x, y ] -> [ (1,3)(21,22), (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
  18,19, 20) ]
gap> PreImagesRepresentative(hom, (1, 2));
fail
gap> PreImagesRepresentative(hom, (21,22));
fail
```

Tampoco podemos girar el disco 180^{o} manteniendo un orden dado. Estos hechos nos indican que en este caso la estructura del grupo es de producto *semidirecto* (ver definición en Hungerford, T. W., 1974) de $\Sigma_{10} \times \Sigma_{10}$ y $\langle (21, 22) \rangle$. En general, si $n = 20$ y $k \geq 2$, entonces el grupo resultante del puzzle TopSpin cuando k es par tiene estructura de producto directo de los grupos Σ_{20} y $\langle R \rangle$, siendo R la trasposición correspondiente al giro del disco. Por tanto cualquier ordenación de las 20 piezas es resoluble, cosa que no sucede cuando k es impar.

4.4. Anillo de Rubik

El **Anillo de Rubik** consiste en dos anillos intersectados que contienen 34 bolas, de tres colores distintos. Estos anillos se intersectan en dos lugares en los cuales comparten una bola. Todas las bolas se pueden desplazar alrededor de cada anillo y, por tanto, pueden mezclarse. El objetivo de nuevo es volver desde una posición mezclada a la posición inicial. En el anillo de Rubik, los anillos se intersectan en un ángulo, formando una pieza tridimensional. Una variante de este rompecabezas es el *anillo húngaro*, inventado por el ingeniero húngaro Endre Pap, el cual posee 4 colores y su geometría es bidimensional. Ambas versiones pueden jugarse en la web de Scherphuis, J.

Figura 20 – Anillo de Rubik.

Figura 21 – Anillo húngaro.

Para su modelado, basta con numerar las bolas en un determinado orden y ver qué movimientos básicos son los que nos determinarán el grupo correspondiente a todos los movimientos posibles.

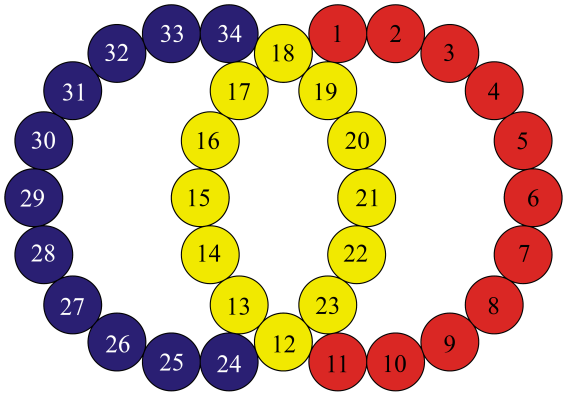


Figura 22 – Anillo de Rubik posición inicial.

Los movimientos básicos son el desplazamiento una unidad en sentido antihorario del anillo derecho y análogamente lo mismo para el anillo izquierdo. Introduciendo las posiciones resultantes en GAP, nos proporcionan las siguientes permutaciones:

```
gap> ruedaD:=PermListList([2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
> 18,1,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34],[1..34]);
(1,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2)
gap> ruedaI:=PermListList([1,2,3,4,5,6,7,8,9,10,11,24,13,14,15,16,17,
> 19,20,21,22,23,12,25,26,27,28,29,30,31,32,33,34,18],[1..34]);
(12,23,22,21,20,19,18,34,33,32,31,30,29,28,27,26,25,24)
```

Definamos de nuevo el grupo libre y el homomorfismo correspondientes:

```
gap> Anillo:=Group([ruedaD, ruedaI]);
Group([ (1,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2),
  (12,23,22,21,20,19,18,34,33,32,31,30,29,28,27,26,25,24) ])
gap> F:=FreeGroup("x", "y");;
gap> hom:=GroupHomomorphismByImages(F, Anillo, GeneratorsOfGroup(F),
> GeneratorsOfGroup(Anillo));
[ x, y ] -> [ (1,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2),
  (12,23,22,21,20,19,18,34,33,32,31,30,29,28,27,26,25,24) ]
```

Supongamos que queremos resolver la siguiente disposición de bolas. Basta entonces con proceder de nuevo mediante la antiimagen del homomorfismo anterior:

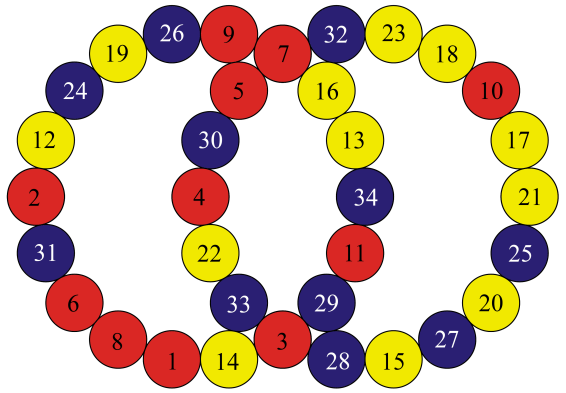


Figura 23 – Anillo de Rubik desordenado.

```
gap> PosicionFinal:=PermListList([32,23,18,10,17,21,25,20,27,15,28,3,33,
> 22,4,30,5,7,16,13,34,11,29,14,1,8,6,31,2,12,24,19,26,9],[1..34]);
(1,25,7,18,3,12,30,16,19,32)(2,29,23)(4,15,10)(5,17)(6,27,9,34,21)
 (8,26,33,13,20)(11,22,14,24,31,28)
```

No incluimos la solución de `PreImagesRepresentative(hom,PosicionFinal);` pues da lugar a una expresión demasiado larga para ser escrita aquí. Sin embargo, cabe destacar que este procedimiento solo daría lugar a una solución, que no es la única, pues por ejemplo si consiguiéramos llegar a la posición inicial con el 5 y el 6 intercambiados no sería necesario seguir realizando movimientos pues comparten color (ver Figura 22). De forma general, otras soluciones posibles son cualquier permutación de números correspondientes a piezas del mismo color. Esto quiere decir que el método anterior solamente proporciona una solución (de muchas otras posibles).

Por otra parte, en principio sabemos que el grupo que genera el anillo de Rubik es un subgrupo de Σ_{34}. Cabe plantearse también, como en algunas de las secciones anteriores, qué posiciones son resolubles.

```
gap> Size(Anillo)=Size(SymmetricGroup(34));
true
```

Acabamos de comprobar que el grupo del anillo de Rubik es exactamente Σ_{34} y, por tanto, cualquier posición dada es resoluble.

Otro puzzle similar es el *Turnstile* (ver Scherphuis, J.), donde existen dos discos giratorios los cuales también se intersectan. Por tanto, podríamos modelizar dicho rompecabezas y el anillo húngaro de forma análoga al anillo de Rubik. Mencionar que un análisis de este tipo del anillo húngaro puede consultarse en Mulholland, J.

4.5. Puzzle del deslizamiento

El **puzzle del deslizamiento** es un juego de deslizamiento de piezas que presentan un determinado orden inicial dentro de un cuadrado o rectángulo. Tiene su origen en Nueva Inglaterra (Estados Unidos) en el siglo XIX, aunque rápidamente se extendió al resto de América y Europa. El reto original, de 16 piezas en un cuadrado 4×4, trataba de volver a la posición inicial ordenada a partir del ordenamiento donde las piezas 14 y 15 estaban invertidas, como se muestra a continuación:

Figura 24 – Puzzle de 15 piezas.

Figura 25 – Puzzle intercambiado.

Realmente se desconoce a quién atribuir la invención de este puzzle. En los años 1890, Sam Loyd ofreció un premio de 1000$ a quien pudiera mostrar una solución. Por esta razón comúnmente se le atribuye la creación del juego. Mucha gente se adelantó anunciando que había encontrado una solución, aunque o eran incapaces de demostrar en público la secuencia de movimientos posible o habían entendido mal el juego.

La diferencia de este rompecabezas con los anteriores es que los movimientos que pueden realizarse dependen de donde esté situada la pieza vacía y, por tanto, no pueden ser modelizados como permutaciones de un grupo. Existen algoritmos en diferentes lenguajes de programación que dan soluciones a este puzzle y su estudio no será tratado en este trabajo (vease *The Fifteen Puzzle - The Algorithm*, para más información). Sin embargo, sí podemos investigar qué posiciones dadas son resolubles y comprobar que el problema planteado por Loyd no es factible.

Para modelar dicho puzzle, asociamos a cada configuración de las piezas una permutación de manera análoga a las secciones anteriores. Por ejemplo, queda claro que la trasposición $(14, 15)$ representa la configuración de la Figura 25. Calculemos ahora la permutación asociada al siguiente puzzle:

13	3	15	10
14		8	9
1	5	4	2
6	11	12	7

(1)

Para ello, vamos a hacer uso de nuevo del comando `PermListList`:

```
gap> PosicionActual:=PermListList([ 13, 3, 15, 10, 14, 16, 8, 9, 1, 5, 4,
> 2, 6, 11, 12, 7 ],[1..16]);
(1,9,8,7,16,6,13)(2,12,15,3)(4,11,14,5,10)
```

Cada movimiento básico del puzzle implica un intercambio de posiciones entre la pieza vacía (la pieza 16) y otra pieza adyacente. Si las piezas en las posiciones i y j son intercambiadas permaneciendo las demás intactas, este movimiento puede describirse con la trasposición (i, j). Así, dada cualquier permutación σ, si la postmultiplicamos por (i, j) el resultado nos dará una nueva permutación en la que se han intercambiado solamente las posiciones i y j. Veamos esto con el siguiente ejemplo:

13	3	15	10
14		8	9
1	5	4	2
6	11	12	7

$\rightsquigarrow$

3	14	15	10
13		8	9
1	5	4	2
6	11	12	7

(2)

Obviamente pasar de una configuración a otra del puzzle no es posible con un solo movimiento básico, sino con una composición de ellos, tal y como mostramos a continuación:

13	3
14	

$\rightsquigarrow$

13	3
	14

$\rightsquigarrow$

	3
13	14

$\rightsquigarrow$

3	
13	14

$\rightsquigarrow$

3	14
13	

(3)

Notemos que solamente hemos representado la sección del puzzle donde se realizan los intercambios de piezas. Por tanto, hemos aplicado consecutivamente las trasposiciones $(5, 6)$, $(1, 5)$, $(1, 2)$, $(2, 6)$, cuyo producto nos va a determinar la permutación $(1, 5, 2)$ deseada. Comprobemos esto con `GAP`.

```
gap> PosicionActual*(5,6);
(1,9,8,7,16,5,10,4,11,14,6,13)(2,12,15,3)
gap> Permuted([1..16], (1,9,8,7,16,5,10,4,11,14,6,13)(2,12,15,3));
[ 13, 3, 15, 10, 16, 14, 8, 9, 1, 5, 4, 2, 6, 11, 12, 7 ]
gap> PosicionActual*(5,6)*(1,5);
(1,9,8,7,16)(2,12,15,3)(4,11,14,6,13,5,10)
gap> Permuted([1..16], (1,9,8,7,16)(2,12,15,3)(4,11,14,6,13,5,10));
[ 16, 3, 15, 10, 13, 14, 8, 9, 1, 5, 4, 2, 6, 11, 12, 7 ]
gap> PosicionActual*(5,6)*(1,5)*(1,2);
(1,9,8,7,16,2,12,15,3)(4,11,14,6,13,5,10)
gap> Permuted([1..16], (1,9,8,7,16,2,12,15,3)(4,11,14,6,13,5,10));
[ 3, 16, 15, 10, 13, 14, 8, 9, 1, 5, 4, 2, 6, 11, 12, 7 ]
gap> PosicionActual*(5,6)*(1,5)*(1,2)*(2,6);
(1,9,8,7,16,6,13,5,10,4,11,14,2,12,15,3)
gap> Permuted([1..16], (1,9,8,7,16,6,13,5,10,4,11,14,2,12,15,3));
[ 3, 14, 15, 10, 13, 16, 8, 9, 1, 5, 4, 2, 6, 11, 12, 7 ]
gap> (5,6)*(1,5)*(1,2)*(2,6);
(1,5,2)
```

El comando `Permuted()` nos proporciona una lista de elementos permutados por una permutación dada. Por tanto, nos ha ido devolviendo la posición del puzzle tras intercambiar las posiciones i y j. Observemos que el último vector obtenido con el comando `Permuted` se corresponde con la configuración final del puzzle mostrado en (2). Aquí es donde reside la principal dificultad del juego: dada una permutación, encontrar una descomposición suya como producto de trasposiciones concatenadas donde se vaya moviendo consecutivamente la pieza vacía (recordemos además que dicha descomposición puede no ser única, ver el Teorema 5.3).

Una vez que hemos analizado cómo modelizar los movimientos del puzzle, vamos a ver su resolubilidad. Por consiguiente, el reto original planteado por Loyd es equivalente a probar si existe o no una serie de movimientos (trasposiciones concatenadas) que nos lleven de la posición inicial a la posición donde solamente están intercambiadas las piezas 14 y 15, siendo su permutación correspondiente $(14, 15)$. Así, estamos suponiendo que existen n trasposiciones $\tau_1, \ldots, \tau_n$ en el grupo de permutaciones de 16 elementos de forma que $(14, 15) = ()\tau_1\tau_2\cdots\tau_n$. Es decir, realizaremos una serie de trasposiciones que nos construirán un camino del tipo siguiente:

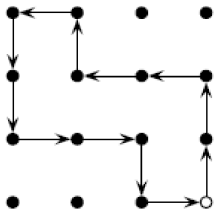


Figura 26 – Un posible camino en el puzzle del deslizamiento.

Observemos que para cada movimiento hacia la izquierda tendremos que realizar otro a la derecha en algún momento y cada movimiento hacia arriba conllevará otro hacia abajo. Es decir, tenemos que realizar un número par de movimientos, lo que conlleva un número par de trasposiciones. Esto implicaría que la trasposición $(14, 15)$ es una permutación par, lo que nos da lugar a una contradicción que muestra la imposibilidad de la solución. Como consecuencia, se puede ver que un ordenamiento dado que posea la pieza vacía en la posición 16 solamente es resoluble cuando la permutación asociada τ al resto de piezas es par, i.e., cuando $\tau \in A_{15}$. Para una formalización matemática más exhaustiva puede consultarse (Beeler, R. A., Conrad, K., Joyner, D., 2008) y Mulholland, J.). A modo de ejemplo, veamos si el puzzle (1) es resoluble: primero colocamos la pieza vacía en la esquina inferior derecha. Notemos que este paso deja invariante la paridad de la permutación asociada, pues cualquier camino que realicemos va a conllevar un número par de trasposiciones (se repite la idea de la Figura 26). Para una formalización más estricta de este paso véase Mulholland, J., Theorem 9.2.

13	3	15	10
14		8	9
1	5	4	2
6	11	12	7

$\rightsquigarrow$

13	3	15	10
14	8	9	
1	5	4	2
6	11	12	7

$\rightsquigarrow$

13	3	15	10
14	8	9	2
1	5	4	7
6	11	12	

Seguidamente, calculamos su permutación asociada y vemos su paridad con el comando `SignPerm()`, que resulta 1 cuando es par y -1 cuando es impar.

```
gap> Posicion:=PermListList([ 13, 3, 15, 10, 14, 8, 9, 2, 1, 5,
> 4, 7, 6, 11, 12, 16], [1..16]);
(1,9,7,12,15,3,2,8,6,13)(4,11,14,5,10)
gap> SignPerm((1,9,7,12,15,3,2,8,6,13)(4,11,14,5,10));
-1
```

Concluimos que dicho puzzle no es resoluble. Finalmente, cabe mencionar que el análisis presentado en esta sección se podría adaptar también a cualquier puzzle del deslizamiento $n \times m$, con $n, m \geq 1$, siendo las permutaciones resolubles (que dejan la posición vacía en una esquina del puzzle) las que pertenecen al grupo A_{nm-1} (ver Figura 27).

Otra variación que puede considerarse del puzzle del deslizamiento es cuando algunas piezas puedan repetirse, como sucede en la Figura 28 con la letras A y N. El objetivo original de este puzzle, que producía controversia, es si la configuración CANAMA PANAL es resoluble. Observemos que, si consideramos todas las piezas diferentes, esto no sería posible pues se corresponde con la trasposición $(1, 7) \notin A_{11}$. Sin embargo, si consideramos válidas permutaciones de letras iguales como las A de las posiciones 2 y 4, entonces obtenemos su resolubilidad pues $(1, 7)(2, 4) \in A_{11}$. Para más información sobre este puzzle ver The Panama Canal Puzzle.

Figura 27 – Puzzle deslizable 3×3.

Figura 28 – Puzzle deslizable 6×2.

5. Conclusiones

Podríamos pensar que los rompecabezas y puzzles presentados en este trabajo pueden servir sólo de entretenimiento y diversión. Cualquier persona puede atreverse a intentar su resolución sin tener demasiados conocimientos matemáticos. Esta es una de las principales causas que han propiciado su popularidad. Sin embargo, dichos entretenimientos pueden entenderse como juegos realmente educativos, no sólo porque desarrollan nuestro intelecto, nuestra visión espacial, capacidad lógica y de raciocinio, sino porque constituyen además una eficaz herramienta en el entendimiento de algunas estructuras algebraicas que, debido a su alto grado abstracción matemática, son de difícil comprensión. Se trata de "jugar con la teoría de grupos", descubriendo ejemplos sencillos que permanecen escondidos tras la geometría intrínseca de estos juegos. El estudio de los movimientos "legales" en estos rompecabezas nos permiten estimular el aprendizaje inicial de las propiedades más elementales de la teoría de grupos mediante permutaciones. Al mismo tiempo, nos permite dar una visión lúdica y más aplicada de dicha teoría a los estudiantes de matemáticas, presentando de forma natural ejemplos relativos a conceptos tan abstractos como: grupo libre, homomorfismos de grupos, conjugación, normalidad, producto directo, producto semidirecto, producto orlado, etc. En Scherphuis, J. puede encontrarse una recopilación amplia de otros juegos que pueden ser tratados de forma similar a los aquí presentados. Hemos de señalar que este trabajo es una simple introducción al estudio algebraico de estos entretenimientos, existen tratamientos más teóricos y profundos que los aquí presentados (puede consultarse por ejemplo Joyner, D. (2008) y Mulholland, J.). Sin embargo, hemos visto cómo aprovechando las utilidades computacionales del sofware `GAP`, la teoría de la computación nos ha acercado al entendimiento de la teoría de grupos. Es por esta razón que hemos preferido dar un tratamiento más computacional tanto a la modelización como al estudio de la resolución de estos puzzles.

Agradecimientos

Este trabajo está financiado por el Proyecto Prometeo II/2015/011, Generalitat Valenciana (España) y por la beca predoctoral ACIF/2016/170, Generalitat Valenciana (España). Nos gustaría agradecer a Laura Avellán Carrión la creación de algunas de las figuras presentadas y a Ana Martínez Pastor sus siempre valiosos comentarios.

Referencias


Beeler, R. A.
The Fifteen Puzzle: a motivating example for the alternating group.
http://faculty.etsu.edu/beelerr/fifteen-supp.pdf


Conrad, K.
The 15-puzzle (and Rubik's cube).
http://www.math.uconn.edu/~kconrad/blurbs/grouptheory/15puzzle.pdf


Esteban Romero, R. (2013).
Las matemáticas del cubo de Rubik.
Pensamiento Matemático, 3(2), 97–110.


Hungerford, T. W. (1974).
Algebra.
Springer-Verlag, New York.


Joyner, D. (2008).
Adventures in group theory: Rubik's cube, Merlin's machine, and other mathematical toys.
The Johns Hopkins University Press, Baltimore.


Kaufmann, S. (2011).
A mathematical analysis of the generalized Oval Track Puzzle.
Rose-Hulman Undergraduate Mathematics Journal 12(1): article 5 .


Long, C.
Solving the TopSpin Puzzle using GAP.
http://angrystatistician.blogspot.com.es/2013/07/solving-topspin-puzzle-using-gap.html


Mulholland, J.
Permutation Puzzles: A Mathematical Perspective.
http://www.sfu.ca/~jtmulhol/math302/notes/302notes.pdf


Sangroniz, J. (2015).
¿Se puede hacer un huevo frito en el cubo de Rubik?
La Gaceta de la RSME, 18(1), 103–120.


Scherphuis, J.
https://www.jaapsch.net/puzzles


The Fifteen Puzzle - The Algorithm.
https://rosettacode.org/wiki/15_Puzzle_Game


The GAP Group (2016).
GAP - Groups, Algorithms, and Programming, Versión 4.8.5
http://www.gap-system.org


The Panama Canal Puzzle
http://www.cs.brandeis.edu/~storer/JimPuzzles/SLIDE/PanamaCanal/PanamaCanal.pdf.