



Departamento de Matemáticas, Facultad de Ciencias
Universidad Autónoma de Madrid

# Ciclos universales para permutaciones

**Trabajo de fin de grado**

Grado en Matemáticas

*Autor:* Jaime Pons Garrido

*Tutor:* Daniel Ortega Rodrigo, Fernando Quirós Gracián

Curso 2022-2023

## Resumen

Los ciclos universales se concibieron como una generalización de los ciclos de *De Bruijn* para una familia de estructuras combinatorias. El objetivo de este trabajo de fin de grado es realizar un estudio teórico de las propiedades de los ciclos universales para las permutaciones y proporcionar métodos constructivos que sean capaces de generarlos.

Tras explicar las dificultades iniciales con las que uno se encuentra al intentar construirlos, se presenta una definición general que usa relaciones de equivalencia para sortearlas. Se demuestra su existencia para todo $S_n$ y se da un tamaño óptimo teórico del alfabeto de símbolos requerido para formarlos. A su vez, se presentan algoritmos constructivos simples que permiten generarlos a partir de alfabetos con cardinal subóptimo.

Finalmente, se proporciona una representación alternativa de los mismos variando la relación de equivalencia que se aplica sobre la familia de permutaciones y se demuestra que esta nueva representación solventa algunos de los retos que tiene la representación inicial. El trabajo concluye presentando dos algoritmos que utilizan un alfabeto óptimo a la hora de generar ciclos universales para las permutaciones.

## Abstract

Universal cycles were conceived as a generalization of *De Bruijn* cycles for a family of combinatorial structures. The objective of this undergraduate thesis is to conduct a theoretical study on the properties of universal cycles for permutations and provide constructive methods capable of generating them.

After explaining the initial difficulties encountered in attempting to construct them, a general definition is presented that uses equivalence relations to overcome these difficulties. Their existence is proven for all $S_n$, and an optimal theoretical size of the symbol alphabet required to form them is provided. Additionally, constructive algorithms are presented that allow generating universal cycles from alphabets with suboptimal cardinality.

Finally, an alternative representation of the cycles is provided by varying the equivalence relation applied to the family of permutations. It is proven that this new representation overcomes some of the challenges faced by the initial representation. The thesis concludes by presenting two algorithms that use an optimal alphabet to generate universal cycles for permutations.

# Índice general

# CAPÍTULO 1
# Los ciclos universales

## 1.1. Concepto de ciclo universal

El concepto de ciclo universal es relativamente novedoso en el campo de las matemáticas. Introducidos por primera vez en *"Universal cycles for combinatorial structures"* (1992) [1] por Chung, Diaconis y Graham, son, en esencia, ciclos que contienen todos los elementos de un conjunto de estructuras combinatorias exactamente una vez.

Antes de ser reconocidos, ya se habían estudiado algunos casos particulares. El caso más claro sería el de los ciclos de *De Bruijn*. Se presentará un caso particular de estos ciclos como ejemplo para después introducir el concepto general de ciclo universal. El siguiente ciclo de *De Bruijn* contiene a todos los miembros de la familia de las variaciones con repetición de dos elementos tomados de cuatro en cuatro.

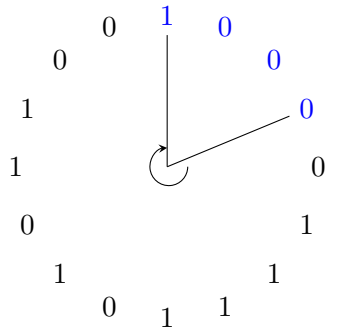



Figura 1.1: Ejemplo de ciclo de *De Bruijn*.

En la Figura 1.1 se pueden tomar secuencias de cuatro números consecutivos en dirección horaria, como se muestra en la figura, a las que llamaremos ventanas de grado cuatro. Al tener dieciséis elementos y poder tomar una ventana comenzando en cualquiera de ellos, obtenemos exactamente dieciséis ventanas y cada una de ellas es un miembro de la familia de las variaciones con repetición de dos elementos tomados de cuatro en cuatro. Por tanto, este ciclo cumple la propiedad de contener todos los

elementos de la estructura combinatoria sin repetir ninguno y, como veremos más adelante, eso lo convierte en un ciclo universal.

Denominaremos familia de permutaciones de $n$ elementos ($n$-permutaciones) $S_n$, al conjunto de funciones biyectivas que forman el grupo simétrico sobre el conjunto finito de elementos $\{1, ..., n\}$.

Antes de adentrarnos en la definición formal de los ciclos universales, es conveniente intentar dar un ejemplo para una familia de permutaciones de dimensión baja como es $S_3$. Notemos que, los ciclos universales para $S_1$ o $S_2$ son triviales y de escaso interés. Por lo que pasamos a buscar un ciclo con seis elementos que contenga a todas las 3-permutaciones.

La familia de permutaciones $S_3$ es el conjunto de las funciones biyectivas $\sigma_i(x) : \{1, 2, 3\} \longrightarrow \{1, 2, 3\}$ con $i : 1, ..., 6$.

| $x$ | 1 | 2 | 3 |
|---|---|---|---|
| $\sigma_1(x)$ | 1 | 2 | 3 |
| $\sigma_2(x)$ | 1 | 3 | 2 |
| $\sigma_3(x)$ | 2 | 1 | 3 |
| $\sigma_4(x)$ | 2 | 3 | 1 |
| $\sigma_5(x)$ | 3 | 1 | 2 |
| $\sigma_6(x)$ | 3 | 2 | 1 |

Procedemos entonces a intentar colocar de forma cíclica todas las permutaciones de $S_3$, de forma que cada permutación aparezca una única vez en el ciclo tomando una secuencias de tres elementos consecutivos en dirección horaria.

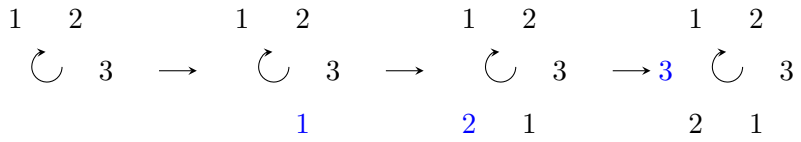



Figura 1.2: Construcción de ciclo no válido como universal para $S_3$.

Si uno se fija en la Figura 1.2, se puede apreciar que el ciclo obtenido no es un ciclo universal ya que las permutaciones 123, 231 y 312 aparecen repetidas y además el ciclo solo se puede construir de ese modo. Esto se debe a que, una vez seleccionada una permutación inicial, el resto de elementos del ciclo ya han quedado prefijados. Por ejemplo, sean 123 los tres primeros elementos de un ciclo candidato, entonces la segunda ventana de ciclo será 231 trivialmente, ya que no se puede tener elementos repetidos dentro de una ventana, esto ocurre sucesivamente para todos los nuevos elementos del ciclo.

Pasa de igual forma para cualquier permutación inicial que se tome, y no solo en $S_3$. Por ejemplo, si se toma la permutación $1234 \in S_4$, se obtienen las siguientes transiciones ya que solo hay una permutación posible en cada caso, obtenida al colocar el primer elemento en última posición:

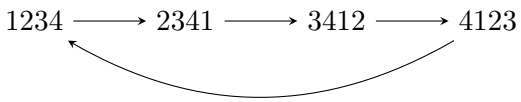



Figura 1.3: Transiciones entre permutaciones en $S_4$ a partir de 1234.

Ante esta perspectiva, parece que la generación de ciclos universales para permutaciones es una tarea imposible. No obstante, en la siguiente sección veremos que esto no es así, gracias a una idea presentada en [1].

## 1.2. Ciclos universales basados en clases de equivalencia

El problema de la generación de ciclos universales para permutaciones se puede interpretar como la incapacidad para encontrar caminos que unan las permutaciones pertenecientes a una misma familia $S_n$ entre sí pasando solo una vez por cada una. Pero está claro que algunas permutaciones no pueden estar incluidas en un mismo camino, como pasa con 123 y 132. A partir de la primera no existe ninguna secuencia de permutaciones que permita llegar a la segunda, véase la Figura 1.2, y esto imposibilita añadir ambas a un mismo ciclo.

La solución proporcionada en [1] es la de utilizar clases de equivalencia para lograr dichas conexiones que, utilizando únicamente los elementos de la familia combinatoria, serían imposibles. Se define una clase de equivalencia para cada elemento de la familia combinatoria y el ciclo universal se generará a partir de representantes de cada una de las clases. Veamos un ejemplo de este proceso. Para ello, comenzaremos definiendo una relación de equivalencia.

**Definición 1.1. Relación de equivalencia de orden**

Sean $a = a_1 \dots a_n$ y $b = b_1 \dots b_n$ dos $n$-tuplas donde tanto $a_i$ como $b_i$ son elementos de un alfabeto ordenado $\mathcal{A}$ para $i : 1, ..., n$. Entonces decimos que $a$ y $b$ son equivalentes ($a \backsim b$) si:

$$a_i < a_j \Leftrightarrow b_i < b_j \quad \forall i, j : 1, \dots, n. \tag{1.1}$$

Aquí el símbolo $<$ representa el orden entre los elementos del alfabeto. Por ejemplo:

$$123 \backsim 456, \quad 749 \backsim 218, \quad 315 \not\backsim 412.$$

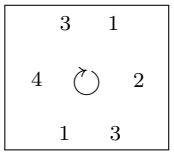
Una vez que tenemos esta relación, intentemos generar un ciclo universal sobre $S_3$ pero con la ayuda de un alfabeto extendido por un símbolo $\mathcal{A} = \{1, 2, 3, 4\}$. Fijémonos en que ahora podemos establecer clases de equivalencia usando la relación definida anteriormente.

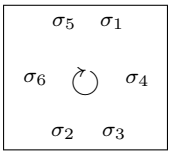
| | | | | | |
|---|---|---|---|---|---|
| 123 | 132 | 213 | 231 | 312 | 321 |
| 124 | 142 | 214 | 241 | 412 | 421 |
| 134 | 143 | 314 | 341 | 413 | 431 |
| 234 | 243 | 324 | 342 | 423 | 432 |

Figura 1.4: Clases de equivalencia dadas por combinaciones de tres elementos de $\mathcal{A}$.

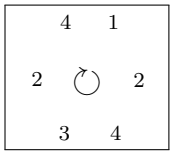
La Figura 1.4 presenta todos los representantes de las clases de equivalencia de $S_3$ con un alfabeto de cuatro elementos. Por tanto, el problema de encontrar un ciclo universal se reduce a hallar caminos formados por representantes de clase unibles que contengan un único representante de cada clase de equivalencia. Debido al reducido tamaño del conjunto de representantes de clase, es posible aplicar un algoritmo comprobatorio de fuerza bruta para encontrar todos los ciclos universales en $S_3$ con un alfabeto de cuatro elementos. El pseudocódigo del algoritmo se proporciona en el Capítulo 2.

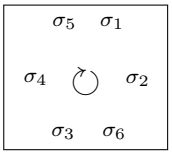
| $\sigma_1$ | $\sigma_4$ | $\sigma_3$ | $\sigma_2$ | $\sigma_6$ | $\sigma_5$ |
|---|---|---|---|---|---|
| 123 | 231 | 213 | 132 | 321 | 312 |
| 124 | 241 | 214 | 142 | 421 | 412 |
| 134 | 341 | 314 | 143 | 431 | 413 |
| 234 | 342 | 324 | 243 | 432 | 423 |



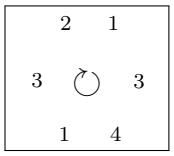



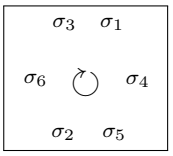

| $\sigma_1$ | $\sigma_2$ | $\sigma_6$ | $\sigma_3$ | $\sigma_4$ | $\sigma_5$ |
|---|---|---|---|---|---|
| 123 | 132 | 321 | 213 | 231 | 312 |
| 124 | 142 | 421 | 214 | 241 | 412 |
| 134 | 143 | 431 | 314 | 341 | 413 |
| 234 | 243 | 432 | 324 | 342 | 423 |



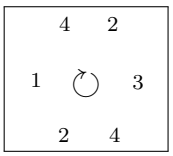




| $\sigma_1$ | $\sigma_4$ | $\sigma_5$ | $\sigma_2$ | $\sigma_6$ | $\sigma_3$ |
|---|---|---|---|---|---|
| 123 | 231 | 312 | 132 | 321 | 213 |
| 124 | 241 | 412 | 142 | 421 | 214 |
| 134 | 341 | 413 | 143 | 431 | 314 |
| 234 | 342 | 423 | 243 | 432 | 324 |




| $\sigma_1$ | $\sigma_4$ | $\sigma_6$ | $\sigma_3$ | $\sigma_2$ | $\sigma_5$ |
|---|---|---|---|---|---|
| 123 | 231 | 321 | 213 | 132 | 312 |
| 124 | 241 | 421 | 214 | 142 | 412 |
| 134 | 341 | 431 | 314 | 143 | 413 |
| 234 | 342 | 432 | 324 | 243 | 423 |



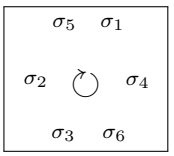




Figura 1.5: Todos los ciclos universales en $S_3$ usando un alfabeto de cuatro elementos $\mathcal{A} = \{1, 2, 3, 4\}$ .

Una vez obtenido este prometedor resultado para $S_3$, sería interesante ver si algo parecido se podría aplicar para el caso de $S_4$. La respuesta es que sí: añadiendo tan solo un elemento más al alfabeto, $\mathcal{A} = \{1, 2, 3, 4, 5\}$, es posible construir un ciclo

universal con relativa facilidad. Dado que este caso es notablemente más complejo, solo se proporcionará la construcción de un ciclo universal. Para ello se consideran las permutaciones en $S_4$:

| | | | | | |
|---|---|---|---|---|---|
| $\gamma_1$=1234 | $\gamma_2$=1243 | $\gamma_3$=1324 | $\gamma_4$=1342 | $\gamma_5$=1423 | $\gamma_6$=1432 |
| $\gamma_7$=2134 | $\gamma_8$=2143 | $\gamma_9$=2314 | $\gamma_{10}$=2341 | $\gamma_{11}$=2413 | $\gamma_{12}$=2431 |
| $\gamma_{13}$=3124 | $\gamma_{14}$=3142 | $\gamma_{15}$=3214 | $\gamma_{16}$=3241 | $\gamma_{17}$=3412 | $\gamma_{18}$=3421 |
| $\gamma_{19}$=4123 | $\gamma_{20}$=4132 | $\gamma_{21}$=4213 | $\gamma_{22}$=4231 | $\gamma_{23}$=4312 | $\gamma_{24}$=4321 |

Empleando representantes de clase, se obtiene la secuencia 123514352143214523142354, generada mediante el siguiente camino entre clases de equivalencia:

$$\gamma_1\gamma_{10}\gamma_{11}\gamma_{20}\gamma_3\gamma_{16}\gamma_{18}\gamma_{21}\gamma_8\gamma_6\gamma_{24}\gamma_{15}\gamma_7\gamma_4\gamma_{17}\gamma_{22}\gamma_9\gamma_{14}\gamma_5\gamma_{13}\gamma_2\gamma_{12}\gamma_{23}\gamma_{19},$$

que podemos presentar en forma de ciclo:

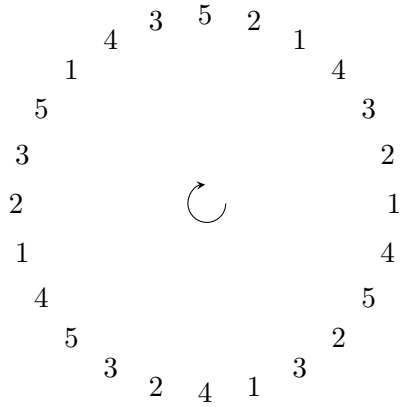



Figura 1.6: Ejemplo de ciclo universal sobre $S_4$.

## 1.3. Definición formal de ciclo universal

En la sección anterior se ha presentado la idea de generar los ciclos universales usando representantes de clases de equivalencia. A continuación veremos una definición formal de un ciclo universal para una familia de estructuras combinatorias $\mathfrak{F}$.

**Definición 1.2. Ciclo universal ($u$-ciclo)**

Sea $\mathfrak{F}$ una familia de estructuras combinatorias con cardinal $m := |\mathfrak{F}|$ y sea "$\backsim$" una relación de equivalencia que asocia cada elemento $F \in \mathfrak{F}$ con secuencias de elementos de un alfabeto ordenado $\mathcal{A}$ dado. La lista $U := a_0 \dots a_{m-1}$ donde $a_i \in \mathcal{A}$ para $i : 0, 1, ..., m-1$ es un ciclo universal de $\mathfrak{F}$ si y sólo si cada $F \in \mathfrak{F}$ está asociado mediante la relación de equivalencia exactamente a una secuencia $a_i \dots a_{i+n-1}$ a la que se denomina como ventana de grado $n$.

Una vez dada esta definición, es conveniente volver a los ciclos de *De Bruijn*, Figura 1.1, e identificar algunos de los elementos de la misma para este caso particular. La

familia de estructuras combinatorias $\mathfrak{F}$ es la familia de variaciones con repetición de longitud cuatro usando dos elementos y el alfabeto utilizado para representar dichos elementos es $\mathcal{A} = \{0, 1\}$. Pero el elemento de mayor interés es la relación de equivalencia utilizada, que es la identidad sobre el conjunto $\mathfrak{F}$. Cada elemento de la familia $\mathfrak{F}$ tendría una clase de equivalencia que solo lo contendría a él mismo. Esto es importante a la hora de comprender el concepto de ciclo universal ya que, aunque los ciclos de *De Bruijn* son particularmente conocidos y tienden a usarse como ejemplo introductorio, es importante resaltar que enmascaran la necesidad de usar clases de equivalencia y nos hacen pensar que los ciclos universales se pueden generar únicamente usando los elementos de la familia de estructuras.

Es sencillo darse cuenta de que el concepto de ciclo universal es intrínsecamente amplio al admitir cualquier familia de estructuras combinatorias y permitir el uso de diferentes alfabetos y maneras de asociar cada ventana a un elemento de $\mathfrak{F}$. Es por eso que es conveniente hallar casos concretos que permitan un estudio más exhaustivo. Este trabajo se centra por tanto en la investigación de los ciclos universales para las familias de permutaciones con $n$ elementos, $\mathfrak{F} = S_n, n \in \mathbb{N}$. Al afrontar su estudio nos planteamos una serie de preguntas fundamentales:

- ¿Existen ciclos universales para todas las familias $S_n$? De existir, ¿cuántos son?
- Respecto al alfabeto de símbolos $\mathcal{A}$, ¿cuál es su cardinal mínimo necesario? Es decir, ¿cuántos símbolos distintos necesito para generar un $u$-ciclo en $S_n$?
- ¿Existe una forma de generarlos mediante algoritmos? Y si la hay, ¿qué características presentan los distintos métodos usados para generarlos?

  **Observación 1.3.** *Notemos que* $\log(U) = |\mathfrak{F}|$ *es necesario para poder tener una ventana asociada a cada elemento de* $\mathfrak{F}$. *En* $S_n$ *esto implica un crecimiento rápido de* $\log(U)$ *conforme* $n$ *aumenta.*

Para finalizar la sección, reflexionemos sobre la relación de equivalencia de orden que hemos definido. Esta relación es, tal vez, la más natural a la hora de asociar las ventanas de un $u$-ciclo con las permutaciones de $S_n$. Sin embargo, no es la única relación de equivalencia que se puede utilizar con este objetivo. En el tercer capítulo se presentará una nueva relación de equivalencia, que siendo menos intuitiva, será de gran utilidad a la hora de resolver algunos obstáculos que surgirán durante el estudio de los ciclos universales en $S_n$.

## 1.4. Existencia de ciclos universales en $S_n$

La existencia de ciclos universales en $S_n$ es un problema que se ha estudiado desde la aparición de los mismos. En el año 2000, Johnson publicó un artículo [7] en el que demostraba su existencia y resolvía la siguiente conjetura publicada en [1].

**Conjetura.** *Para* $n \geq 3$ *se puede obtener un u-ciclo para* $S_n$ *de modo que* $|\mathcal{A}| = n+1$, *siendo* $\mathcal{A}$ *el alfabeto del u-ciclo.*

El lector atento a los ciclos presentados hasta ahora, se habrá percatado de que todos hasta el momento cumplen con la conjetura propuesta. En efecto, Johnson demostró que esta es verdadera a través del siguiente teorema [7]:

**Teorema 1.4.** *Para $n \geq 3$ existe un ciclo de longitud $n!$ sobre un alfabeto ordenado $\mathcal{A}$ de $n+1$ símbolos tal que para cada $p \in S_n$ existe una ventana $v$ de longitud $n$ del ciclo tal que $p \backsim v$.*

### 1.4.1. Conceptos preliminares

Antes de comenzar a demostrar el teorema, es necesario definir algunos conceptos previos como el de grafo de transición. Este tipo de objeto es ampliamente utilizado en problemas de ciclos universales y se define de la siguiente manera cuando se buscan ciclos universales para $S_n$ con un alfabeto de $n+1$ símbolos. En lo que sigue, $[n]$ denotará el conjunto $\{0, 1, \ldots, n-1\}$.

**Definición 1.5.** Un **grafo de transición para** $S_n$ es el grafo dirigido $G_n = (V, A)$:

$$\begin{aligned} V &= \{(a_1 \ldots a_n) : a_i \in [n+1] \text{ y } a_i \neq a_j \text{ para todo } i \neq j\}, \\ A &= \{\{(a_1 \ldots a_n), (b_1 \ldots b_n)\} : a_{i+1} = b_i \text{ para todo } 1 \leq i \leq n-1\}. \end{aligned} \tag{1.2}$$

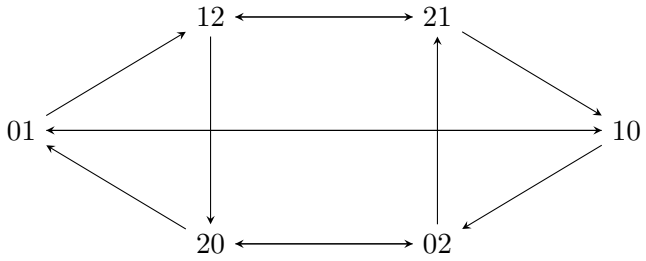



Figura 1.7: Grafo de transición $G_2$.

**Observación 1.6.** *Cada vértice del grafo de transición $G_n$ tiene grado de salida y entrada igual a 2.*

Para comprobar la observación, basta con notar que la restricción impuesta para las aristas $(A)$ nunca fija $b_n$ y al tener un alfabeto compuesto por $n+1$ símbolos tan solo caben dos posibilidades de transición, fijando un grado de salida igual a dos. A partir de esto, es trivial ver que el grado de entrada también es dos ya que es imposible que un vértice tenga más de dos aristas entrantes.

**Definición 1.7.** Dado $b \in [j]$, se define $s_b(x) : [j] \longrightarrow [j+1]$ como:

$$s_b(x) = \begin{cases} x+1 & \text{si } x \geq b, \\ x & \text{si } x < b. \end{cases} \tag{1.3}$$

Aplicando un abuso de notación, decimos que, dada $a = a_1 ... a_n$, $s_b(a) = s_b(a_1) ... s_b(a_n)$. Por ejemplo, $s_2(2134) = 3145$.

**Observación 1.8.** *Si $a = a_1...a_n \in S_n$ y $b \in [n+1]$, entonces $s_b(a)$ es la única tupla con elementos pertenecientes a $[n+1] \setminus \{b\}$ que, según la relación de equivalencia 1.1,* pertenece a la misma clase de equivalencia que $a$.

**Definición 1.9.** Definimos como **rotación izquierda de una permutación** a la función $r(a_1...a_n) = a_2a_3...a_{n-1}a_na_1$. La notación $r^i(a)$ denota $i$ rotaciones a la izquierda de $a$. En particular $r^n(a) = a$.

### 1.4.2. Demostración del teorema de Johnson

La idea principal de la demostración es usar el alfabeto $\mathcal{A} = [n+1]$ para construir un ciclo que contenga una ventana asociada mediante la relación de equivalencia de la Definición 1.1 a cada elemento de $S_n$. El teorema demuestra los casos $n \geq 5$, debido a que su demostración incluye un paso por inducción basado en la generación de un subárbol $T_n$ que debe de cumplir una serie de condiciones que no se pueden dar para $n$ menor que cinco.

**Observación 1.10.** *Para completar la prueba, recordemos que el caso $n = 3$ ya ha sido proporcionado en la Figura* **??** y para el caso $n = 4$ proporcionamos un ejemplo en la Figura 1.6.

Para demostrar la existencia de un $u$-ciclo para $S_n$ con $n \geq 5$ se construirá un árbol $T_n$ de $(n-1)!$ nodos, donde cada uno será una permutación distinta de $S_{n-1}$. A partir de cada nodo se generará un ciclo de longitud $n$ de modo que todos los ciclos generados sean disjuntos. Se explicará cómo los ciclos generados por nodos adyacentes se pueden unir para acabar formando un ciclo de longitud $n!$ en el grafo $G_n$, que será el $u$-ciclo buscado. Cuando ese resultado quede demostrado, se dará un paso por inducción en el que uniendo $n$ copias del árbol $T_n$ se obtendrá un árbol $T_{n+1}$ que al unir los ciclos generados por todos sus vértices dará un $u$-ciclo para $S_{n+1}$.

El primer paso de la demostración es hallar un conjunto de ciclos de longitud $n$ en $G_n$ de tal manera que, entre todos ellos, contengan exactamente un representante de cada clase de equivalencia de $S_n$. Para hallar cada ciclo se tomará la permutación $a_1 \dots a_{n-1} \in S_{n-1}$ y se construirá la tupla $0a = (0a_1...a_{n-1})$. Aplicando la función dada en la Definición 1.7 se obtiene $s_{l(a)}(0a)$. La etiqueta $l(a)$ es el único elemento del alfabeto $\mathcal{A}$ que no aparecerá en el ciclo generado, véase 1.8, y su elección es arbitraria, aunque más tarde se verá que debe de cumplir ciertos criterios para que el paso de inducción sea válido. El ciclo buscado $C(a, l(a))$ se obtiene rotando $s_{l(a)}(0a)$ a la izquierda $n-1$ veces:

$$C(a, l(a)) = s_{l(a)}(0a)r(s_{l(a)}(0a)) \dots r^{n-1}(s_{l(a)}(0a)). \tag{1.4}$$

Por ejemplo, consideremos $C(45321, 2)$. Para obtenerlo se debe calcular $s_2(045321) = 056431$ y rotar la tupla obtenida a la izquierda cinco veces, véase la Figura 1.8.

Se puede comprobar que dadas dos permutaciones distintas $a, b \in S_{n-1}$, para cualquier elección de etiquetas $l(a)$ y $l(b)$, se cumple que los ciclos $C(a, l(a))$ y $C(b, l(b))$ son disjuntos. Dado que $s_{l(a)}(0a)$ mantiene las relaciones de orden como se menciona

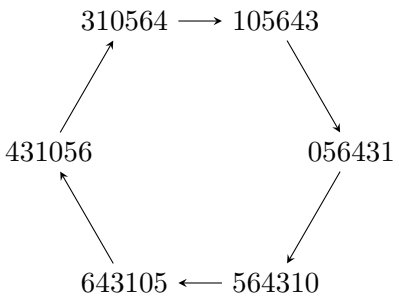



Figura 1.8: $C(45321, 2)$ expresado como un conjunto de rotaciones a la izquierda.

en 1.8, queda garantizado que $C(a, l(a))$ y $C(b, l(b))$ tendrán relaciones de orden distintas entre sus primeros $n-1$ elementos y sus ciclos generados al rotar a la izquierda mantendrán dichas relaciones entre los elementos desplazados. Por tanto, independientemente de la elección de etiquetas:

$$\bigcup_{a \in S_{n-1}} C(a, l(a)), \tag{1.5}$$

es una unión disjunta.

El siguiente paso de la demostración trata de enlazar con éxito dos de estos ciclos. La cuestión fundamental es definir cuándo dos ciclos son **enlazables**.

**Definición 1.11.** Dos ciclos $C_a := (a, l(a))$ y $C_b := (b, l(b))\, 1 \leq l(a) \leq l(b)-2 \leq n-1$ son **enlazables** si se satisface que fijado el ciclo $C_a$ los elementos $b_i$ cumplen:

$$b_i = \begin{cases} a_i & \text{si } 1 \leq a_i \leq l(a) - 1 \vee l(b) \leq a_i \leq n-1, \\ a_i + 1 & \text{si } l(a) \leq a_i \leq l(b) - 2, \\ l(a) & \text{si } a_i = l(b) - 1. \end{cases} \tag{1.6}$$

Esta construcción busca que $s_{l(a)}(a)$ y $s_{l(b)}(b)$ tan solo difieran en un elemento, concretamente en $s_{l(a)}(a)_t = l(b)$ y $s_{l(b)}(b)_t = l(a)$. Veamos un ejemplo usando $C_1 = C(42135, 2)$ y $C_2 = C(23145, 5)$:

$$s_2(42135) = 053146, \qquad s_5(23145) = 023146.$$

En este caso $t = 2$, $s_2(42135)_2 = 5$ y $s_5(23145)_2 = 2$. Esta propiedad permitirá que al rotar los ciclos a la izquierda exista una posibilidad de transición en $G_n$ que comunique un ciclo con el otro.

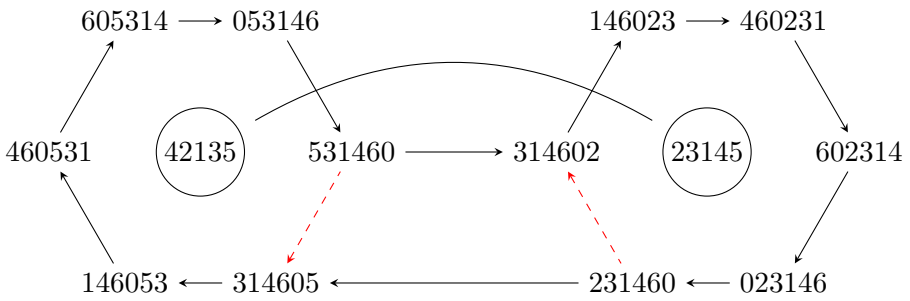



Figura 1.9: Dos ciclos enlazables, $C_1 = C(42135, 2)$ y $C_2 = C(23145, 5)$.

Como se puede apreciar en la Figura 1.9 la unión de los vertices de $C_1$ y $C_2$ genera un nuevo ciclo de longitud $2n$ una vez que eliminamos las transiciones con flechas rayadas de la figura, que corresponden con:

$$r^t(s_2(0a)) \to r^{t+1}(s_2(0a)),\ r^t(s_t(0b)) \to r^{t+1}(s_5(0b)).$$

Buscamos poder unir una cantidad finita de estos ciclos, en concreto nos interesa unir uno por cada elemento de $S_{n-1}$. Para ello definimos el grafo $H_n$.

**Definición 1.12.** Definimos el grafo $H_n$ como el grafo con vértices $V$ y aristas $A$:

$$\begin{aligned} V &= \{(a, x) : a \in S_{n-1}, x \in [n]\}, \\ A &= \{\{(a, x)(b, y)\} : C(a, x), C(b, y) \text{ enlazables}\}. \end{aligned} \tag{1.7}$$

Una vez definido $H_n$, demostremos que si existiese un subárbol $T_n$ de $H_n$ que contuviese $(n-1)!$ vértices de la forma $(a, l(a))$ uno para cada $a \in S_{n-1}$, ya se tendría un ciclo universal.

Cada vértice $(a, l(a))$ tendría asociado su ciclo correspondiente $C(a, l(a))$. El objetivo sería unir el ciclo asociado a cada vértice con los de sus vecinos para formar un ciclo universal. Al ser $H_n$ un grafo sabemos que, por definición de adyacencia, los ciclos asociados a cada vértice serían enlazables con los de sus vecinos.

Una cuestión diferente sería saber el "aspecto" que tendría esa unión, ya que podría darse el caso de que dos uniones con ciclos distintos requiriesen eliminar las mismas transiciones de un ciclo $C(a, l(a))$ dado. Pero nótese que para que esto sucediese un elemento de $G_n$ tendría que tener más de dos enlaces (el de su propio ciclo y un enlace a, al menos, otros dos ciclos). Por la observación 1.6, eso implicaría un grado de salida mayor que dos, lo cual es imposible, como se aprecia en la Figura 1.10.


Figura 1.10: Posibles transiciones de 460531 en un ciclo con alfabeto $\mathcal{A} = [n+1]$.

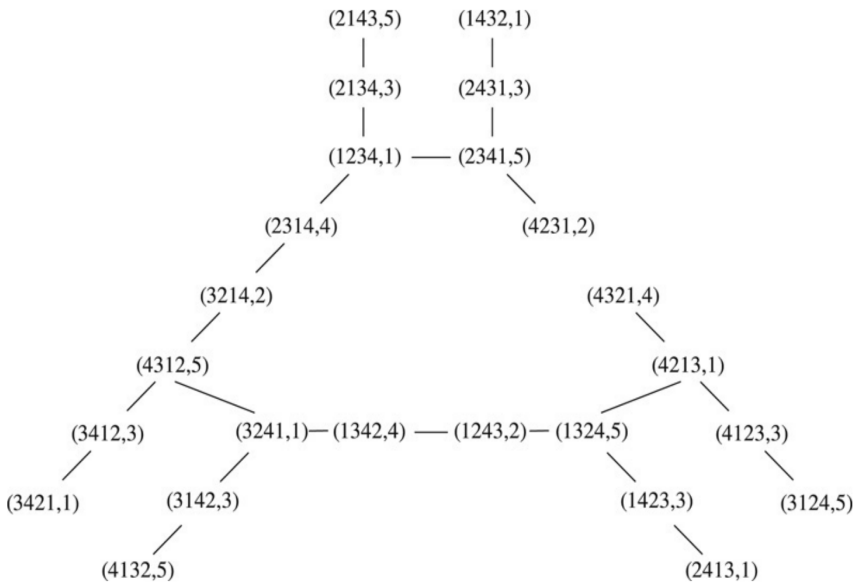



Figura 1.11: Ejemplo de árbol $T_5$. Fuente: [7].

Visto esto, es claro que al ser el subárbol $T_n$ conexo por definición, los ciclos asociados a cada vértice se podrían unir y se obtendría un ciclo universal.

El último paso es demostrar que el subárbol $T_n$ mencionado realmente existe, para ello se demostrará por inducción que para todo $n \geq 5$ existe $T_n$ contenido en $H_n$ que satisface:

1. $\forall a \in S_{n-1}$ existe un único $x \in [n]$ tal que $(a, x) \in V(T_n)$. Denotaremos por $v(a)$ al único vértice en $V(T_n)$ de la forma $(a, x)$;
2. $(12 \dots (n-1), 1) \in V(T_n)$;
3. $(23 \dots (k-1)1(k)(k+1) \dots (n-1), k) \in V(T_n)$ para todo $3 \leq k \leq n$;
4. $(32145 \dots (n-1), 2) \in V(T_n)$;
5. $(24315 \dots (n-1), 3) \in V(T_n)$;
6. $v(31245 \dots (n-1))$ es una hoja en $T_n$;
7. $v(24135 \dots (n-1))$ es una hoja en $T_n$.

**Observación 1.13.** *Dadas las condiciones del paso de inducción, es sencillo ver que no existe ningún subárbol $T_3$ o $T_4$ que las cumpla. Un subárbol $T_3$ está compuesto por los vértices $(12, x)$ y $(21, y)$ que no son enlazables. En el caso $T_4$, imponiendo las condiciones 1 y 3, se obtiene la componente $(123, 1) - (231, 4)$ dentro del subárbol, que no es enlazable con ningún otro vértice.*

Es posible encontrar con relativa facilidad [7] un subárbol para $n = 5$, véase la Figura 1.11. Luego supongamos que para $n \geq 5$ existe $T_n$ obtenido a partir de $H_n$ que cumple las condiciones listadas. Utilizaremos esta información para hallar $T_{n+1}$ en $H_{n+1}$.

**Observación 1.14.** *Sea* $(a_1 \dots a_{n-1}, l(a)) \in V(H_n)$ *entonces:*

$$(1(a_1+1)\dots(a_{n-1}+1), l(a)+1) \in V(H_{n+1}).$$

*La construcción* $H_{n+1}$ *preserva la adyacencia de* $H_n$*. Además, aplicar una permutación fija a cualquier tupla perteneciente a un vértice de* $H_{n+1}$ *proporciona un automorfismo en* $H_{n+1}$*. Esto implica que los subgrafos de* $H_{n+1}$ *poseen copias isomorfas en* $H_{n+1}$*.*

De la observación 1.14 se sigue que cada subgrafo de $H_n$ estaría relacionado mediante un isomorfismo a una copia isomorfa en $H_{n+1}$. Tomamos $n$ copias de $T_n$. Estas copias serán modificadas para ser las partes del nuevo subárbol $T_{n+1}$.

En la primera copia, denominada $T_{n+1}^{(0)}$, cada vértice $(a_1 ... a_{n-1}, l(a))$ es reemplazado de la siguiente forma:

$$(1(a_3+1)(a_1+1)(a_4+1)(a_2+1)(a_5+1)(a_6+1)\dots(a_{n-1}+1), l(a)+1).$$

Nótese que, según la observación 1.14, es una copia isomorfa de $T_n$ en $H_{n+1}$. En la segunda copia $T_{n+1}^{(1)}$ se reemplaza cada vértice por:

$$((a_3+1)1(a_2+1)(a_1+1)(a_4+1)(a_5+1)\dots(a_{n-1}+1), l(a)+1).$$

En las copias restantes $T_{n+1}^{(k)}$ con índice $2 \leq k \leq n-1$ se reemplaza siguiendo la fórmula:

$$((a_k+1)(a_1+1)(a_2+1)\dots(a_{k-1}+1)1(a_{k+1}+1)\dots(a_{n-1}+1), l(a)+1).$$

**Definición 1.15.** Denominamos $F_{n+1}$ al bosque contenido en $H_{n+1}$ tal que:

$$F_{n+1} = \bigcup_{0 \leq k \leq n} T_{n+1}^{(k)}.$$

Los árboles $T_{n+1}^{(k)}$ que componen el bosque son disjuntos entre sí, ya que 1 aparece en una coordenada diferente en los vértices de cada uno de ellos. Además, para cada permutación $a \in S_n$ existe un vértice en $F_{n+1}$ de la forma $(a, l(a))$ con $l(a) \in [n+1]$.

**Observación 1.16.** *Por construcción de los* $T_{n+1}^{(k)}$ *se tiene que si* $(a_1 \dots a_n)$ *es una hoja en* $T_n$ *entonces todos los vértices de las siguientes formas lo serán en* $F_{n+1}$*:*

- $v(1(a_3+1)(a_1+1)(a_4+1)(a_2+1)(a_5+1)(a_6+1)\dots(a_{n-1}+1))$,
- $v((a_3+1)1(a_2+1)(a_1+1)(a_4+1)(a_5+1)\dots(a_{n-1}+1))$,
- $v((a_k+1)(a_1+1)(a_2+1)\dots(a_{k-1}+1)1(a_{k+1}+1)\dots(a_{n-1}+1))$ *para* $2 \leq k \leq n-1$.

Tras describir el bosque $F_{n+1}$ recordamos que para obtener el ciclo universal el bosque debe de tener un único componente, es decir, debe ser un árbol. Para ello realizamos la siguiente modificación:

Sabemos, por definición, que $v(12\dots n)$ es una hoja en $T^{(0)}_{n+1}$ y por tanto una hoja en $F_{n+1}$. Esto es sencillo de probar si recordamos que $v(24135\dots(n-1))$ es una hoja en $T_n$ y aplicamos la observación 1.16. Entonces se sustituye ese vértice por $(123\dots n,1)$ y se crean aristas desde el sustituto hacia todos los elementos de $T^{(k-1)}_{n+1}$ con $2 \leq k \leq n-1$.

Usando las observaciones previas se puede ver que estos vértices se encuentran en $F_{n+1}$, y que el bosque solo tiene dos componentes. Con un razonamiento similar, tenemos que, por hipótesis, $v(31245\dots(n-1))$ es una hoja en $T_n$ lo que implica que $v(342156\dots n)$ es una hoja en $F_{n+1}$. Se elimina el vértice $v(342156\dots n)$ y es reemplazado por $(342156\dots n,1)$ añadiendo a su vez aristas desde el sustituto hacia $(341256\dots n,3)$ y $(143256\dots n,4)$.

La construcción de $T^{(2)}_{n+1}$ y el hecho de que $(32145\dots(n-1),2) \in V(T_n)$ implican que $(342156\dots n,1) \in V(F_{n+1})$. De igual forma, la construcción de $T^{(0)}_{n+1}$ y que $(24315\dots(n-1),3) \in V(T_n)$ implican que $(143256\dots n,4) \in T^{(0)}_{n+1}$ Con esto se tiene que el bosque $F_{n+1}$ tiene un único componente al que se denota como $T_{n+1}$.

Por último, solo queda comprobar que $F_{n+1}$ cumple las propiedades inductivas. Por construcción se tiene que $T_{n+1}$ contiene exactamente un vértice de la forma $(a,l(a))$ para todo $a \in S_n$. Además $(12\dots n,1)$ y $(23\dots a1(a+1)(a+2)\dots n,a+1)$ son vértices de $T_{n+1}$ para todo $2 \leq a \leq n$. Luego se cumplen las hipótesis 1, 2 y 3.

Los supuestos de las condiciones 4 y 5 se cumplen gracias a la construcción de $T^{(2)}_{n+1}$ y $T^{(3)}_{n+1}$, que asegura que los vértices requeridos por las mismas estén contenidos en $T_{n+1}$.

Dado que $v(31245\dots(n-1))$ es una hoja en $T_n$ y que $T^{(1)}_{n+1}$ está contenido en la unión que conforma $F_{n+1}$ implica que $v(31245\dots n)$ es una hoja en $F_{n+1}$, que equivale a la propiedad 6. Un argumento paralelo sirve para demostrar que $v(241356\dots n)$ es también una hoja en $F_{n+1}$ usando la construcción de $T^{(2)}_{n+1}$ y el hecho de que $v(31245\dots(n-1))$ sea hoja en $T_n$, propiedad 7. Con esto quedan demostradas todas las condiciones del paso inductivo y por tanto concluye la demostración.

## 1.5. Estimación del cardinal de ciclos universales sobre $S_n$

En [7] Johnson facilita unas cotas que acotan el cardinal de los ciclos universales sobre $S_n$. El resultado está basado parcialmente en la demostración del teorema de la sección anterior y solo contempla la construcción de ciclos sobre un alfabeto de $n+1$ símbolos.

**Teorema 1.17.** *Sea $|U_n|$ el cardinal de los ciclos universales sobre $S_n$ construidos sobre el alfabeto $\mathcal{A} = [n+1]$ para $n \geq 5$. Entonces se cumple que:*

$$420^{\frac{(n-1)!}{24}} \leq |U_n| \leq (n+1)2^{n!-n}. \tag{1.8}$$

*Demostración.* Para establecer el límite inferior se consideran todos los subárboles de $H_n$ que cumplen las siete condiciones establecidas en la demostración del teorema de Johnson. Todos esos subárboles generan ciclos universales distintos, por lo que pueden dar un límite inferior.

Por otro lado, dados $t_n$ subárboles de $H_n$ se puede utilizar que $H_{n+1}$ está construido en base a $n$ copias de $T_n$, lo que implica que $H_{n+1}$ tendrá al menos una cantidad $t_n^n$ de subárboles. Iterando hasta $t_5$, que es es el caso base de la demostración, se obtiene el siguiente cardinal:

$$t_5^{\frac{(n+1)!}{4!}}. \tag{1.9}$$

Finalmente Johnson se vale de un caso específico de árbol $T_5$, véase la Figura 1.11, para establecer la cota. En ese caso, para que todos los subárboles de este grafo cumplan las propiedades atribuidas a $T_5$ solo es necesario añadir los siguientes enlaces:

- $(1432, 1) \to (2143, 5)$;
- $(3421, 1) \to (4132, 5)$;
- $(4231, 2) \to (4321, 4)$.

Johnson calculó que, en ese caso, existían 420 subárboles de $T_5$.

La estimación superior se basa en las posibles elecciones tomadas al construir un $u$-ciclo elemento a elemento. Se establece una primera secuencia de longitud $n$ que pertenezca a la clase de equivalencia de $(12\ldots n)$ según la relación de equivalencia de la Definición 1.1, lo cual se puede hacer de $n+1$ maneras. Consecuentemente quedan por completar $n! - n$ elementos del $u$-ciclo; para esas posiciones no podemos escoger ninguno de los símbolos usados en las $n-1$ posiciones anteriores, lo cual nos deja dos posibilidades. Con esto llegamos a la cota:

$$(n+1)2^{n!-n}. \tag{1.10}$$

□

Es evidente que, aunque la diferencia entre cotas es significativa, la cota inferior es muy alta para todo $n$, lo cual deja patente que hay muchas posibles construcciones de ciclos universales sobre $S_n$. La siguiente tarea natural, que cubriremos en las próximas secciones, es estudiar formas constructivas mediante las cuales podemos obtener dichos $u$-ciclos.

# CAPÍTULO 2
# Algoritmos generadores de $u$-ciclos en $S_n$

En este capítulo se tratarán diferentes perspectivas a la hora de abordar el problema de la generación de $u$-ciclos en $S_n$. Para ello se presentarán diferentes métodos capaces de generarlos, se analizará el fundamento teórico subyacente de cada uno de ellos y se discutirán sus propiedades y características más relevantes.

Una de las propiedades deseables de un algoritmo generador de $u$-ciclos es que su método de construcción utilice un alfabeto $\mathcal{A}$ que tenga un cardinal reducido. En este sentido, es lógico prever que el cardinal de un potencial alfabeto para un $u$-ciclo en $S_n$ venga acotado de la siguiente forma: $n+1 \leq |\mathcal{A}| \leq n!$ donde la cota superior es la longitud total del $u$-ciclo y la inferior es el resultado del teorema de Johnson visto en la sección anterior. Otra característica relevante es la eficiencia a la hora de construir un $u$-ciclo, una cualidad extremadamente deseable dado que la longitud de los $u$-ciclos para las permutaciones es $n!$ lo que implica altos tiempos de computación.

Finalmente, es importante saber si el algoritmo es capaz de generar $u$-ciclos distintos para $S_n$ en cada ejecución o si, por el contrario, solo es capaz de generar uno. Esto se traduce en si se trata de un algoritmo determinista o no. A la hora de obtener un algoritmo no determinista es importante considerar que no se conoce ningún método que devuelva todos los $u$-ciclos existentes para un $S_n$ dado, por lo que es conveniente plantearse si es posible generarlos por fuerza bruta.

## 2.1. Búsqueda de $u$-ciclos mediante algoritmo comprobatorio

Una primera aproximación a la generación no determinista de $u$-ciclos en $S_n$ es la de generar ciclos aleatoriamente y comprobar si son universales mediante un algoritmo comprobatorio.

**Algoritmo comprobatorio de $u$-ciclos en $S_n$**

Se presenta una palabra inicial $\Pi$ con longitud $n!$, construida sobre un alfabeto $\mathcal{A} = \{1, ..., n+1\}$.

```
1: let permutaciones = Set()
2: ventanas = extraer_ventanas(Π, n)
3: for ventana in ventanas do
4:     permutacion = clase_de_orden(ventana)
5:     if permutacion in permutaciones then
6:         return False
7:     end if
8:     permutaciones.add(permutacion)
9: end for
10: return True
```

Este enfoque es extremadamente simple de implementar y es capaz de generar todos los ciclos universales posibles. No obstante, su principal desventaja es su eficiencia, ya que para que el algoritmo sea razonable, la probabilidad de que un ciclo aleatorio sea universal debe de ser alta.

A la hora de evaluar la viabilidad de este enfoque, el Teorema 1.17 es de gran utilidad. La cota inferior proporcionada en el teorema se dividirá entre la superior para dar una cota inferior de la probabilidad. No se usará el cardinal total de ciclos con $n!$ elementos porque se asume que los ciclos se generarán conforme al método detallado en la demostración de la cota superior del teorema.

$$P_{u-ciclo}(n) \geq \frac{420^{\frac{(n-1)!}{24}}}{(n+1)2^{n!-n}}. \tag{2.1}$$

Usando esta fórmula se obtiene que $P_{u-ciclo}(5) \geq \frac{420}{2{,}49e35} \approx 0$. Debido a lo extremadamente baja que es la cota inferior esta no proporciona ninguna información útil que permita evaluar la viabilidad del algoritmo. No obstante, para un $n$ bajo como es cinco se obtiene una cota superior de orden $10^{35}$. De pretender generar todos los ciclos universales para $S_5$ por fuerza bruta, se necesitaría generar 2,49e35 ciclos y comprobar la validez de todos ellos. Con la potencia de computación actual esto es imposible. Esto nos hace preguntarnos si sería mejor simplificar el problema hallando únicamente un $u$-ciclo para cada familia $S_n$ mediante un algoritmo determinista.

## 2.2. Algoritmo de generación voraz

A continuación se presenta un algoritmo voraz que viene dado en [3] y está basado en el algoritmo de Martin para generar ciclos de *De Bruijn* [9]. La construcción se basa en la generación de una palabra de longitud $n! + n - 1$ que contenga por si misma ventanas relacionadas con todos los elementos de $S_n$ para posteriormente eliminar los $n-1$ últimos elementos de la palabra y aplicar una reducción que permita obtener el $u$-ciclo.

**Definición 2.1.** Dada una permutación $a$ su forma reducida $\mathsf{red}(a)$ se obtiene sustituyendo cada elemento de $a$ por su orden relativo en la permutación.

$$a = 4512,\ \mathsf{red}(a) = 3412.$$

Definimos una palabra inicial $\Pi_0 = 12\ldots(n-1)$, donde el subíndice indica el número de iteraciones realizadas y aplicamos de forma iterativa el siguiente algoritmo, que busca extensiones de la palabra $\Pi_k$. Dichas extensiones se generan usando la función $s_b$ que se empleó para la demostración del teorema de Johnson para evitar que se repitan símbolos dentro de una misma ventana de la palabra. La condición que deben de cumplir estas extensiones es que los últimos $n$ elementos extendidos no pertenezcan a la misma clase de equivalencia de alguna ventana ya presente en la palabra $\Pi_k$. Finalmente, se obtendrá una palabra no extensible $\Pi$ de longitud $n!+n-1$, se eliminarán los últimos $n-1$ elementos y se tomará la forma reducida $\mathsf{red}(\Pi') \in U_n$, con lo que finaliza el algoritmo.

**Algoritmo voraz para construir $u$-ciclos en $S_n$**

```
1: while extension_posible do
2:     extension_posible = false
3:     let F(Π_k) = extraer_ventanas(Π_k, n)
4:     for i = 1, 2, ..., n + k do
5:         let extension(Π_k, i) = s_i(π_{k+1}) s_i(π_{k+2}) ... s_i(π_{n+k-1}) i
6:         for ventana in F(Π_k) do
7:             if relacion_de_orden(ventana, extension(Π_k, i)) then
8:                 descartar_extension(extension(Π_k, i))
9:                 break
10:            end if
11:        end for
12:        if no_descartada(extension(Π_k, i)) then
13:            Π_{k+1} = s_i(π_1) s_i(π_2) ... s_i(π_{n+k-1}) i
14:            extension_posible = true
15:            break
16:        end if
17:    end for
18: end while
19: let Π' = Π − {π_{n!+1} ... π_{n!+n-1}}
20: return red(Π')
```

Mostramos un ejemplo de ejecución del algoritmo para $n = 3$, primero se genera la palabra $\Pi$ usando el primer paso del algoritmo:

$$12 \longrightarrow 231 \longrightarrow 3421 \longrightarrow 45312 \longrightarrow 564132 \longrightarrow 6751324 \longrightarrow 78613245 = \Pi.$$

Finalmente se eliminan los últimos dos elementos y se reduce la palabra para obtener el $u$-ciclo en $S_3$:

$$78613245 \longrightarrow 786132 \longrightarrow \mathsf{red}(786132) \longrightarrow 564132 = \mathsf{red}(\Pi') \in U_n.$$

Una vez presentado el algoritmo y dado un ejemplo es el momento de considerar su fundamento teórico.

**Definición 2.2.** Una palabra $W$ es una **$u$-palabra** para $S_n$ si y solo si cada ventana de grado $n$ de $W$ es un representante de la clase de equivalencia de orden de una única permutación de $S_n$ y $W$ contiene $n!$ ventanas.

Para demostrar la corrección del algoritmo es necesario probar que el ciclo $\mathsf{red}(\Pi')$ es un $u$-ciclo en $S_n$. Para hacerlo deberemos dar un paso previo, demostrar que $\Pi$ es una $u$-palabra para $S_n$.

**Observación 2.3.** *Es sencillo ver que la última condición implica que $W$ es una palabra de longitud $n! + n - 1$.*

Definimos los siguientes elementos, teniendo en cuenta que un paso intermedio del algoritmo genera la palabra $\Pi_{n,k} := \pi_1\pi_2 \dots \pi_{k+n-1}$:

$$\theta_k = \mathsf{red}(\pi_k \dots \pi_{k+n-1}), \qquad \theta'_k = \mathsf{red}(\pi_{k+1} \dots \pi_{k+n-1}),$$

$$J_k = |\{j \leq k : \theta'_j = \theta'_k\}|.$$

Notemos que $J_k$ es el número de veces que se repite la $(n-1)$-permutación en la palabra $\Pi_{n,k}$.

**Lema 2.4.** *El algoritmo voraz finaliza en $k \Leftrightarrow \theta_k = 12 \dots n$.*

*Demostración.* Si $\theta_k = 12 \dots n$ es sencillo comprobar que $\theta'_k = \theta'_{k-1} = 12 \dots (n-1)$ y $J_{k-1} = n$. Por tanto al sumar $\theta'_k$ se tendrá $J_k = n + 1$. Por definición todas las $i$-extensiones con $i < J_k$ ocurren en $\Pi_{n,k}$, pero la extensión $J_k$ no. Teniendo en cuenta que una $(n-1)$-permutación solo puede tener $n$ extensiones, se tiene que si $J_k = n+1$ el algoritmo termina en $k$.

Para la otra implicación aplicamos reducción al absurdo y suponemos que $\theta_k \neq 12 \dots n$, lo cual implica que $\theta_j \neq 12 \dots n$ para todo $j \leq k$, esto es trivial una vez demostrada la otra implicación.

Es fácil ver que, como $\pi_{j+1} \dots \pi_{j+n-1}$ viene siempre precedida de $\pi_j$ y la permutación $\theta_j$ ocurre solo una vez por el funcionamiento del algoritmo, tendremos $\theta'_j = \theta'_k$ como mucho en $n$ índices diferentes $j \leq 1$.

Si $\theta'_k \neq 12 \dots (n-1) \Rightarrow \theta'_0 \neq \theta'_k \Rightarrow J_k \leq n$.

Si $\theta'_k = 12 \dots (n-1)$ entonces $\theta'_j = \theta'_k$ para como mucho $n-1$ índices diferentes $\Rightarrow J_k \leq n$.

Por tanto se puede concluir que el algoritmo no termina en $k$ si $\theta_k \neq 12 \dots n$. □

**Teorema 2.5.** *La palabra* $\Pi$ *generada por el algoritmo voraz es una u-palabra en* $S_n$.

*Demostración.* Primero observamos que aplicando el primer paso del algoritmo la palabra $\Pi$ nunca contendrá dos ventanas de la misma clase de equivalencia, ya que siempre se escoge una $i$-extensión que no comparta clase con ninguna otra ventana ya presente en la palabra. La única dificultad restante es comprobar que todas las permutaciones de $S_n$ aparecen en $\Pi$.

Consideramos una prueba por contradicción. Supongamos que existe una permutación $\pi = \pi_1 \dots \pi_n \in S_n$ que no sea equivalente a ninguna ventana de $\Pi$. Entonces $\pi_2 \dots \pi_n$ aparecerá como mucho $n-1$ veces en $\Pi$.

De esto se deduce que $\mathsf{red}(\pi_2 \dots \pi_n)n$ no está cubierto en $\Pi$, ya que de estarlo implicaría que $\pi_2 \dots \pi_n i \in \Pi$ para todo $i \leq n$, lo cual sería contradictorio, porque $\pi_2 \dots \pi_n$ aparecería $n$ veces en $\Pi$.

Generalizando lo previamente dicho se tiene que si $\mathsf{red}(\pi_k \dots \pi_n)(n-k+2) \dots n$ no está cubierto, entonces:

$$\mathsf{red}(\mathsf{red}(\pi_k \dots \pi_n)(n-k+2) \dots n)n = \mathsf{red}(\pi_k \dots \pi_n)(n-k+1) \dots n \tag{2.2}$$

no está cubierto. Esto implica que, tomando $k = n-1$, la palabra no cubre la permutación $12 \dots n$. Vemos que esto no es posible por el Lema 2.4. □

Visto esto, solo queda comprobar que quitando los últimos $n-1$ elementos de la palabra $\Pi$ y tomando la forma reducida se obtiene un $u$-ciclo en $S_n$.

**Teorema 2.6.** *Sea* $\Pi'$ *la palabra generada por el algoritmo voraz menos sus últimos* $n-1$ *elementos. Entonces* $\mathsf{red}(\Pi')$ *es un ciclo universal en* $S_n$.

*Demostración.* Aplicando el Teorema 2.5 es suficiente ver que las ventanas de $\Pi$ que contenían alguno de los elementos $\pi_{n!+1} \dots \pi_{n!+n-1}$ se pueden generar usando los primeros elementos de la palabra $\pi_1 \dots \pi_{n-1}$, o más formalmente:

$$\mathsf{red}(\pi_k \dots \pi_{k+n-1}) = \mathsf{red}(\pi_k \dots \pi_{n!}\pi_1 \dots \pi_{k+n-n!-1}), \qquad n! - n + 2 \leq k \leq n!. \tag{2.3}$$

Para demostrar esto primero notamos que para todo $i < n$ se tiene que $\theta_i$ termina en 1. Esto quiere decir que para los primeros $n-1$ símbolos de la palabra las ventanas que comienzan por ellos cumplen la condición de que el último elemento añadido siempre es el más pequeño. Por ejemplo, en la palabra para $S_3$ se tiene:

$$\underline{786}13245, \qquad 7\underline{861}3245,$$

lo que implica $\pi_k < \pi_1 < \ldots < \pi_{n-1}$ para todo $k \geq n$.

A continuación, siguiendo un argumento similar al de la demostración anterior, se demuestra que $\theta_i$ termina en $n$ $\forall i \geq n! - n + 2$. Al tratarse de una $u$-palabra se tiene $\theta'_{i-1} = \theta'_{j-1}$ para algún $j < i$. Pero $\theta_{n!} = 12 \dots n$ implica:

$$\pi_{n!+i-j} < \pi_{n!+i-j+1} < \ldots < \pi_{n!+n-1},$$

y $\theta'_{i-1} = \theta'_{j-1}$ implica:

$$\pi_{n!+i-j} < \pi_{n!+i-j+1} < \ldots < \pi_{i+n-2} < \ldots < \pi_{n!+n-1}.$$

Aplicando esta propiedad de manera iterativa se tiene $\pi_j < \pi_{j+1} < \ldots < \pi_{n!+n-1}$ y por tanto $\pi_i < \pi_{i+1} < \ldots < \pi_{n!+n-1}$, lo que implica que $\theta_i$ debería terminar forzosamente en $n$ llegando a una contradicción.

Una vez demostrado que $\theta_i$ termina en $n$ para todo $i \geq n! - n + 2$ se puede afirmar:

$$\pi_k < \pi_{n!+1} < \pi_{n!+2} < \ldots < \pi_{n!+n-1};\ \forall n! - n + 2 \leq k \leq n!;$$

lo que a su vez implica:

$$\theta_k = \mathsf{red}(\pi_k \ldots \pi_{n!})(n! - k + 2) \ldots n = \mathsf{red}(\pi_k \ldots \pi_{n!}\pi_1 \ldots \pi_{k+n-n!-1}) \tag{2.4}$$

para cualquier k tal que $n! - n + 2 \leq k \leq n!$. □

Una vez demostrado esto fijémonos en otras propiedades del algoritmo. Al principio del capítulo se mencionó que los algoritmos podrían ser o no deterministas. En este caso el algoritmo es determinista pues se trata de un algoritmo voraz y el dato inicial $\Pi_0 = 1 \ldots (n-1)$ es fijo a la hora de buscar $u$-ciclos en $S_n$.

La otra propiedad de interés es el alfabeto utilizado para generar los ciclos universales. Si nos fijamos bien, el ejemplo dado para $S_3$ utiliza el alfabeto $\mathcal{A} = \{1, 2, 3, 4, 5, 6\}$, que tiene cardinal mayor que $n + 1$ que, como se mencionó en el capítulo anterior, es el mínimo necesario para poder generar el $u$-ciclo. En [3] se comenta que esto se debe a que durante los primeros $n$ pasos el algoritmo se ve forzado a utilizar un nuevo símbolo por paso y se da una cota para el número de símbolos necesarios. Siendo $|\mathcal{A}|$ el cardinal del alfabeto, este algoritmo voraz requiere que se cumpla:

$$|\mathcal{A}| \geq 2n - 2 > n + 1 \text{ para todo } n \geq 4. \tag{2.5}$$

# CAPÍTULO 3

# Ciclos universales *shorthand* o abreviados

Como vimos en el capítulo anterior, generar ciclos universales de manera algorítmica produce resultados que no son óptimos a la hora de solventar una de las principales cuestiones acerca de los mismos, el tamaño del alfabeto utilizado. Esta sección presenta una forma alternativa de comprender los $u$-ciclos para las permutaciones que se basa en la idea de utilizar una "representación" distinta de las permutaciones pertenecientes a $S_n$. Esto se traduce en relacionar las ventanas de los ciclos con los elementos de $S_n$ usando una nueva relación de equivalencia. A modo de adelanto, esta nueva perspectiva merece atención porque ofrece una manera sencilla y eficiente a nivel computacional de generar ciclos universales sobre $S_n$ con un alfabeto de tan solo $n$ símbolos.

En el año 2005, Knuth [8] publicó un artículo en el que propuso una nueva relación de equivalencia aplicable a los ciclos universales para las permutaciones. La idea fundamental es que una $n$-permutación, siempre que no se permitan alfabetos ampliados, puede ser representada con tan solo $n-1$ símbolos, puesto que el último queda siempre fijado por todos los demás. Por ejemplo, la 3-permutación 231 es la permutación abreviada de la 4-permutación 2314.

Esta observación, a priori trivial, ha marcado una amplia línea de investigación. Al final del capítulo de introducción se mencionó que la relación de equivalencia de orden no sería la única que se usaría en el documento para relacionar las ventanas de un $u$-ciclo con los elementos de $S_n$. Ahora se introducirá la relación de equivalencia abreviada que permite la generación de este nuevo tipo de $u$-ciclo, que se denominará como ciclo universal abreviado.

**Definición 3.1. Relación de equivalencia abreviada**

Sean $a = a_1 \dots a_{n-1} \in S_{n-1}$ y $b = b_1 \dots b_n \in S_n$. Entonces $a$ y $b$ pertenecen a la misma clase de equivalencia abreviada ($a \backsim_{sh} b$) si:

$$a_i = b_i : 1 \leq i \leq n-1.$$

**Observación 3.2.** *Nótese que esta relación de equivalencia es extremadamente simple ya que cada clase solo contiene dos elementos, la permutación en $S_n$ y su forma abreviada (eliminando el último elemento) en $S_{n-1}$.*

La existencia de los ciclos universales abreviados ya había sido demostrada por Jackson [6] antes de que Knuth publicara su artículo, pero no se había prestado atención a las posibilidades prácticas que suponía describir las permutaciones de este modo. De hecho, ni siquiera Knuth propuso una construcción explícita de estos ciclos, sino que tendrían que ser Williams y Ruskey [10] quienes lo hicieran en 2010. En su artículo, los autores proponen un algoritmo que genera los ciclos abreviados de forma bastante eficiente, creando cada nuevo símbolo del $u$-ciclo generado en $O(1)$ como máximo. A continuación, se muestra un ejemplo de un $u$-ciclo abreviado generado por dicho algoritmo.

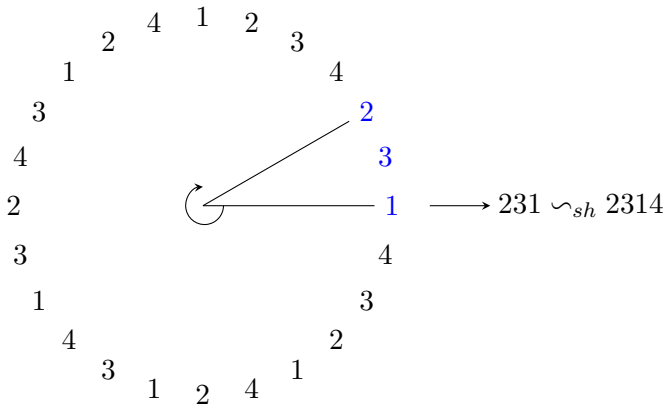



Figura 3.1: Ejemplo de $u$-ciclo abreviado para $S_4$.

Véase que cada ventana en este caso es de longitud o grado 3 $(n-1)$ y está relacionada con exactamente una permutación de 4 elementos $(n)$.

$$124 \backsim_{sh} 1243.$$

Esta nueva forma de representar los $u$-ciclos sobre las permutaciones permite la construcción de otro tipo de algoritmos que toman ventaja del elemento omitido en la representación abreviada para generar $u$-ciclos sobre $S_n$ con ventanas de grado $n-1$. Gracias a eso limitan el cardinal del alfabeto a $n$ símbolos. En la siguiente sección se introducen dos algoritmos que generan ciclos universales abreviados de este tipo.

## 3.1. Concepto de reciclabilidad

Antes de comenzar a trabajar con los algoritmos abreviados, es necesario definir el concepto de reciclabilidad que viene presentado en [4].

**Definición 3.3.** Sea $P = p^1, \ldots, p^{n!}$ un listado de $n$-permutaciones distintas. Escribimos $R(P) = (n+1)p^1(n+1)p^2...(n+1)p^{n!}$ expresando un ciclo. Entonces la lista $P$ es ***reciclable*** si $R(P)$ es un $u$-ciclo abreviado para las $(n+1)$-permutaciones.

**Observación 3.4.** *Fijémonos en que los u-ciclos abreviados se generan a partir de listados concretos de sucesivas permutaciones e introducen el concepto de* ***periodicidad*** *en los u-ciclos ya que el símbolo $n+1$ se repite cada $n+1$ símbolos.*

Una vez dada la definición de reciclabilidad, el siguiente resultado fundamental proporciona dos condiciones equivalentes a ser reciclable.

**Teorema 3.5.** *Se tiene que $P = p^1, \ldots, p^{n!}$ es reciclable si y solo si se cumplen las siguientes condiciones:*

1. *Si $p$ y $p'$ son permutaciones sucesivas en $P$, se tiene que $p_i^{-1} - {p'}_i^{-1} \leq 1$ para todo $i : 1, ..., n$.*
2. *Si $p$ y $v$ son permutaciones sucesivas en $P$ y además $p'$ y $v'$ también lo son, entonces cuando exista un índice $i : 1, ..., n$ tal que:*

$$p_{i+1}...p_n v_1...v_{i-1} = p'_{i+1}...p'_n v'_1...v'_{i-1},$$

*se tiene que $p = p'$ y $v = v'$.*

*Demostración.* El objetivo de la demostración es usar las hipótesis para demostrar que $R(P) = (n+1)p^1(n+1)p^2...(n+1)p^{n!}$ es un $u$-ciclo abreviado para $S_{n+1}$. Para conseguir esto es necesario demostrar que $R(P)$ tiene longitud $(n+1)!$ (trivial), que tomando ventanas de grado $n$ sobre $R(P)$ nunca se tienen números repetidos, y por último, que ninguna de esas ventanas se repite en el ciclo.

Utilicemos la primera hipótesis para demostrar el segundo requisito. Es claro que, por construcción, nunca se tendrán números repetidos en una ventana que no contenga a $n+1$ (ya que son $n$-permutaciones). Luego las ventanas deben tener esta forma:

$$p_{i+1}...p_n(n+1)v_1...v_{i-1}, \tag{3.1}$$

donde $p$ y $v$ son permutaciones sucesivas en $P$. Podemos comprobar que si un número rota más de una posición a la izquierda entre permutaciones sucesivas la primera condición no se cumple:

$$(p(i) = j \Rightarrow p^{-1}(j) = i \text{ y } v(i-2) = j \Rightarrow v^{-1}(j) = i-2) \Rightarrow p^{-1}(j) - v^{-1}(j) \geq 2.$$

Esto nos permite ver que los números rotan como máximo una posición a la izquierda y esto implica que en las ventanas que contienen a $n+1$, como la presentada anteriormente, se tiene $\{p_{i+1}, ..., p_n\} \cap \{v_1, ..., v_{i-1}\} = \emptyset$.

La segunda condición garantiza el tercer requisito, que es que ninguna ventana de grado $n$ se repita. De nuevo si $w$ es una ventana de grado $n$ de $R(P)$ y no contiene a $n+1$ entonces es una permutación y no es posible que se repita. Veamos que pasa si lo contiene. Entonces $w$ es de la forma descrita en la Ecuación (3.1). Si $w$ no fuese única entonces existirían $p'$ y $v'$ tales que $w' = p'_{i+1}...p'_n(n+1)v'_1...v'_{i-1}$ y por la segunda condición $p' = p$ y $v' = v$. Pero esto es una contradicción ya que la lista $P$ no contiene dos veces la misma permutación. □

## 3.2. Algoritmos generadores de $u$-ciclos abreviados

La sección anterior proporciona una clave extremadamente importante para definir los algoritmos que se describen a continuación. Si los ciclos universales abreviados nos brindan la oportunidad de poder generar los $u$-ciclos de la forma $R(P)$, entonces los algoritmos tan solo deben centrarse en buscar listas $P$ de permutaciones que cumplan la condición de reciclabilidad.

Los algoritmos que se mencionan a continuación definen órdenes que permiten generar listas $P$ de permutaciones que cumplen con esta propiedad y por tanto son capaces de generar $u$-ciclos abreviados. Cabe mencionar que en [5] se explica la relación entre estos algoritmos y el concepto informático de *gray code*, expuesto en [2].

### 3.2.1. Algoritmo *bell-ringer*

Este algoritmo, descrito en [5], debe su curioso nombre a la técnica usada para hacer sonar las campanas de las iglesias en Inglaterra. Este método descrito en [11], consiste en hacer sonar una campana haciéndola rotar 360 grados con la ayuda de una rueda.

Para poder aplicar el algoritmo *bell-ringer* a un $u$-ciclo sobre las permutaciones, es necesario introducir un orden al que Holroyd, Ruskey y Williams denominarían **orden** 7. Este orden consiste tomar cada permutación $a \in S_{n-1}$ e insertar $n$ en todas las posiciones posibles siguiendo un orden específico; generando de esa forma $n$-permutaciones que se añaden a la lista.

Para cada $a \in S_{n-1}$ se añaden $na$ y $an$ como primer y segundo elemento, después $n$ rota a la izquierda una posición y la permutación resultante se sigue añadiendo a la lista hasta que $n$ alcance la segunda posición. Por ejemplo, el orden 7 para $n = 3$ o $(7_3)$ es el siguiente:

$$7_3 = \{321, 213, 231, 312, 123, 132\}.$$

El siguiente resultado relacionado con el orden 7 es clave para construir el algoritmo *bell-ringer*:

**Teorema 3.6.** *Las listas generadas por el orden 7 $(7_n)$ son reciclables.*

*Demostración.* Para obtener este resultado se busca cumplir las dos condiciones equivalentes a ser reciclable expuestas en el Teorema 3.5. La primera condición es trivial si tenemos en cuenta la construcción rotando hacia la izquierda propia del orden 7. Recordemos, que para que no se cumpliese la primera condición, un número debería rotar más de una posición a la izquierda entre permutaciones consecutivas en la lista $P$.

Comprobemos ahora la segunda condición mediante reducción al absurdo. Supongamos que existen parejas $p$, $v$ y $p'$, $v'$ de permutaciones consecutivas en $P$ tales que $p \neq p'$, $v \neq v'$ y para un índice $j : 0, ..., n-1$:

$$p_{j+1}...p_n v_1...v_{j-1} = p'_{j+1}...p'_n v'_1...v'_{j-1}.$$

Es sencillo ver que $n \in p_{j+1}...p_n$ o $n \in v_1...v_{j-1}$ ya que la lista está compuesta por permutaciones y el elemento $n$ rota a la izquierda entre permutaciones consecutivas. En el primer caso, se sabe por construcción que:

$$p = v_1...v_{j-1}xp_{j+1}...p_n,$$

con $x \in \{1, ..., n\} \setminus \{v_1, ..., v_{j-1}, p_{j+1}, ..., p_n\}$. Esto se debe a que los $j-1$ primeros elementos de $p$ permanecen inalterados al rotar $n$. Si $n \in p_{j+1}...p_n$ entonces $n \in p'_{j+1}...p'_n$, por tanto:

$$p' = v'_1...v'_{j-1}xp'_{j+1}...p'_n = v_1...v_{j-1}xp_{j+1}...p_n = p.$$

Luego se tiene una contradicción, ya que $p = p'$. Si $n = v_k$ con $k : 2, ..., j-1$ entonces por un razonamiento similar:

$$p = v_1...v_{k-1}v_{k+1}nv_{k+2}...v_{j-1}xp_{j+1}...p_n = p',$$

de nuevo obteniendo una contradicción. Por último, si $n = v_1$, nos encontramos en el caso en el que $n$ llegó a la segunda posición en $p$ y por tanto $v$ es una permutación de la forma $na$ con $a \in S_{n-1}$. Este caso se puede demostrar aplicando inducción sobre la variable $n$ en la segunda condición. Por tanto, se dan las dos condiciones del teorema. □

Una vez probado que el orden 7 es reciclable, podemos afirmar que cualquier ciclo de la forma $R(P_n)$ donde $P_n$ sea una lista de permutaciones de $S_n$ generada por el séptimo orden es un $u$-ciclo universal abreviado en $S_{n+1}$.

Finalmente, en [5] se enuncia un teorema que, directamente, proporciona la lista de permutaciones que genera el algoritmo *bell-ringer*.

**Teorema 3.7.** *Sea $p = p_1...p_n \in S_n$ una permutación contenida en el $u$-ciclo generado por el algoritmo bell-ringer para $S_n$. Sea $m$ el valor máximo entre $p_1$ y $p_n$, y $k$ el máximo valor que cumple que la sucesión $n(n-1)...k$ aparece en $p$. Si $k-1 \leq m \leq n-1$ entonces la siguiente permutación es $p_2p_3...p_{n-1}p_1p_n$. De otro modo, la siguiente permutación es $p_2p_3...p_np_1$.*

### 3.2.2. Algoritmo *cool-lex*

El segundo algoritmo de generación se presenta en [12]. En este caso, se introducen dos nuevos operadores que permiten generar prefijos a la izquierda y a la derecha de las permutaciones. Estos operadores se definen a continuación:

**Definición 3.8.** Se define el operador prefijo a la izquierda ($\lhd$) de una permutación $p = p_1...p_n$ como: $\lhd(p, j) = p_jp_1p_2...p_{j-1}p_{j+1}p_{j+2}...p_n$.

**Definición 3.9.** Se define el operador prefijo a la derecha ($\rhd$) de una permutación $p = p_1...p_n$ como: $\rhd(p, j) = p_2p_3...p_{j-1}p_1p_{j+1}p_{j+2}...p_n$.

Como se puede observar en las definiciones, el operador $\lhd$ mueve un símbolo desde la posición $j$ a la primera posición y el operador $\rhd$ hace justo lo contrario desplazando el primer elemento a la posición $j$.

El algoritmo *cool-lex* se basa en la idea de generar ciclos a partir de la aplicación reiterada de uno de estos operadores. En este caso, se utilizará el movimiento de prefijo a la derecha ($\rhd$) pero utilizar el otro operador también conduce a los resultados esperados.

**Definición 3.10.** El **prefijo no incremental** de una permutación $p = p_1...p_n$ es

$$\nearrow(p) = p_1 p_2 ... p_j,$$

donde $j$ es el máximo valor que cumple que $p_{i-1} \geq p_i$ si $2 \leq i \leq j$.

Por ejemplo: $\nearrow(44527) = 44$ y $\nearrow(66432678) = 66432$. Dada una lista de secuencias se entiende por lista reflejada a la lista obtenida al invertir el orden de las secuencias. Este concepto es el que se aplica para generar el orden *cool-lex* y se obtiene aplicando repetidas veces el movimiento de prefijo a la derecha.

**Definición 3.11.** Sea $p = p_1...p_n$, $p' = p_2...p_n$ y $k' = |\nearrow(p')|$. Entonces

$$\overrightarrow{\mathsf{cool}}(p) = \begin{cases} \rhd(p, k'+1) & \text{si } k' \leq n-2 \wedge p_1 > p_{k'+1}, \\ \rhd(p, k'+2) & \text{si } k' \leq n-2 \wedge p_1 < p_{k'+1}, \\ \rhd(p, k'+1) & \text{si } k' \geq n-1. \end{cases} \tag{3.2}$$

Observemos la aplicación de este operador sobre la permutación $p = 231$. $k' = |\nearrow(31)| = 2$ dado que $k' \geq n-1 = 2$ se tiene que $\overrightarrow{\mathsf{cool}}(p) = \rhd(p_i, 3)$. Es decir, $\overrightarrow{\mathsf{cool}}(p)$ será igual a la permutación $p$ con el primer elemento desplazado a la tercera posición. En este caso, $\overrightarrow{\mathsf{cool}}(p) = 312$.

Esta operación aplicada sobre una permutación $p \in S_n$ genera una lista de permutaciones denominada $\overrightarrow{C}(n)$. Por ejemplo:

$$\overrightarrow{C}(3) = \{321, 213, 123, 231, 312, 132\}.$$

Fijémonos en que el primer elemento de la lista podría ser cualquier 3-permutación. Una vez visto esto se propone el siguiente resultado.

**Teorema 3.12.** *La lista $\overrightarrow{C}(n)$ generada aplicando el orden $\overrightarrow{\mathsf{cool}}(p)$ es reciclable.*

En [4] se presenta una justificación de que $R(\overrightarrow{C}(n))$ es de hecho un $u$-ciclo universal abreviado. No obstante, no se propone una demostración estándar mediante el concepto de reciclabilidad de $\overrightarrow{C}(n)$. Para finalizar este trabajo nos proponemos demostrar este resultado, que dotaría a la teoría desarrollada de una mayor consistencia.

*Demostración.* El objetivo de la demostración es demostrar que una lista $\overrightarrow{C}(n)$ generada por el orden *cool-lex* es reciclable. Para ello se vuelve a utilizar el Teorema 3.5.

Comenzamos demostrando la primera condición del teorema por casos, sea $p = p_1...p_n$ una permutación sobre la que se aplica el orden *cool-lex*.

**Caso 1:** $k' \leq n-2$ **y** $p_1 > p_{k'+1}$**.**

En este caso, sea $\overrightarrow{p} = \overrightarrow{\mathsf{cool}}(p) = p_2...p_{k'}p_{k'+1}p_1p_{k'+2}...p_n$. Notemos que para los índices entre 2 y $k'+1$, $\overrightarrow{\mathsf{cool}}(p)$ es tan solo un desplazamiento a la izquierda, luego cumplen la condición. Los índices desde $k'+2$ hasta $n$ son los mismos que en $p$, por lo que también cumplen la condición. Por último, identificamos tres casos concretos donde es necesario comprobar que la diferencia entre las permutaciones inversas es menor o igual a 1. Sean:

$$\begin{aligned} p(1) &= p_1, \quad & p(k'+1) &= p_{k'+1}; \\ \overrightarrow{p}(1) &= p_2, \quad & \overrightarrow{p}(k'+1) &= p_1. \end{aligned}$$

Al invertir las permutaciones obtenemos:

$$\begin{aligned} p^{-1}(p_1) &= 1, \quad & p^{-1}(p_{k'+1}) &= k'+1; \\ \overrightarrow{p}^{-1}(p_2) &= 1, \quad & \overrightarrow{p}^{-1}(p_1) &= k'+1. \end{aligned}$$

En estos índices afectados por la operación de prefijo, calculamos la diferencia entre las permutaciones inversas:

$$\begin{aligned} p^{-1}(p_1) - \overrightarrow{p}^{-1}(p_1) &= 1 - (k'+1) = -k' - 1 \leq 1; \\ p^{-1}(p_2) - \overrightarrow{p}^{-1}(p_2) &= 1 - 1 = 0 \leq 1; \\ p^{-1}(p_{k'+1}) - \overrightarrow{p}^{-1}(p_{k'+1}) &= k'+1-k' = 0 \leq 1. \end{aligned}$$

Todos cumplen la condición, luego en este caso se da la primera condición del teorema de equivalencia.

**Caso 2:** $k' \leq n-2$ **y** $p_1 < p_{k'+1}$**.**

Es fácil ver que si en el caso anterior se cumple la condición, tomando $k'+2$ en vez de $k'+1$ como índice a remplazar por la operación de prefijo, se sigue cumpliendo la condición mediante una prueba similar.

**Caso 3:** $k' \geq n-1$**.**

Este caso es tal vez el más sencillo de demostrar ya que $\overrightarrow{\mathsf{cool}}(p) = p_2...p_np_1$ que es un caso claro donde todos los elementos se desplazan una posición a la izquierda. De esto se sigue que la primera condición se cumple.

Queda únicamente comprobar que se cumple la segunda condición. Se tienen $p$, $v = \overrightarrow{\mathsf{cool}}(p)$ y $p'$, $v' = \overrightarrow{\mathsf{cool}}(p')$ tal que para un índice $j : 0, ..., n-1$:

$$p_{j+1}...p_nv_1...v_{j-1} = p'_{j+1}...p'_nv'_1...v'_{j-1}.$$

Debemos comprobar que $p = p'$ y $v = v'$. Para ello, consideremos primero el caso $k'+2 < j \leq n-1$. Entonces es claro que el operador de prefijo a la derecha actuando sobre $p$ cumple o bien $k' \leq n-2$ y $p_1 > p_{k'+1}$, o bien $k' \leq n-2$ y $p_1 < p_{k'+1}$. Sin

pérdida de generalidad asumimos que se trata del primer caso. Con las condiciones dadas podemos escribir el lado derecho de la desigualdad anterior como:

$$p_{j+1}...p_n p_2...p_{k'+1}p_1p_{k'+2}...p_{j-1} \Rightarrow p_2...p_{k'+1}p_1p_{k'+2}...p_{j-1} = v'_1...v'_{j-1}.$$

Además conocemos que $v' = \overrightarrow{\text{cool}}(p')$ y que $k'' = |\nwarrow(p'_2...p'_n)|$. De estas afirmaciones se puede concluir lo siguiente:

$$p_2...p_{k'+1}p_1p_{k'+2}...p_{j-1} = v'_1...v'_{j-1} = p'_2...p'_{j-1}.$$

Lo que significa que $k' = k''$ ya que los primeros $k'+1$ símbolos de la permutación coinciden para $p$ y $p'$. Esto junto a la igualdad anterior implica que $p = p'$ y $v = v'$.

Ahora consideremos $k'+2 > j$. En este caso, por hipótesis:

$$p_{j+1}...p_n p_2...p_{j-1} = p_{j+1}...p_n v_2...v_{j-1} = p'_{j+1}...p'_n p'_2...p'_{j-1}.$$

Luego $p = p'$ y se cumple la segunda condición. □

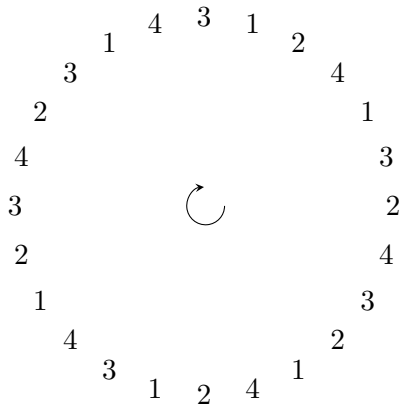


Figura 3.2: Ejemplo de $u$-ciclo abreviado $R(\overrightarrow{C}(3))$ sobre $S_4$.

De nuevo consideremos el ciclo generado $R(\overrightarrow{C}(n))$, que surge al añadir $n+1$ al final de cada permutación. En el caso $R(\overrightarrow{C}(3))$ obtenemos el ciclo de la Figura 3.2. Aplicando ahora el Teorema 3.12 y la Definición 3.3 concluimos:

**Teorema 3.13.** *$R(\overrightarrow{C}(n))$, donde $\overrightarrow{C}(n)$ es la lista generada por el algoritmo cool-lex es un ciclo universal abreviado en $S_{n+1}$.*

Finalmente, se facilita un enlace a un repositorio que incluye el código desarrollado para ejecutar todos los algoritmos de generación de ciclos universales expuestos en este trabajo: https://github.com/Jaime47/CiclosUniversales-Permutaciones-Algoritmos.git.

# Bibliografía


[1] Fan Chung, Persi Diaconis y Ron Graham. "Universal cycles for combinatorial structures". En: *Discrete Mathematics* 110, nº1-3 (1992). ISSN: 0012365X. DOI: 10.1016/0012-365X(92)90699-G.

[2] Robert C. Compton y S. Gill Williamson. "Doubly adjacent gray codes for the symmetric group". En: *Linear and Multilinear Algebra* 35, nº3-4 (1993), págs. 237-293. DOI: 10.1080/03081089308818261.

[3] Alice Gao, Sergey Kitaev y Philip B Zhang. "On a Greedy Algorithm to Construct Universal Cycles for Permutations". En: *International Journal of Foundations of Computer Science* 30 (2017). DOI: 10.1142/S0129054119400033.

[4] Alexander Holroyd, Frank Ruskey y Aaron Williams. "Faster Generation of Shorthand Universal Cycles for Permutations". En: (2010), págs. 298-307. DOI: 10.1007/978-3-642-14031-0_33.

[5] Alexander Holroyd, Frank Ruskey y Aaron Williams. "Shorthand Universal Cycles for Permutations". En: *Algorithmica* 64 (2012), págs. 1-31. DOI: 10.1007/s00453-011-9544-z.

[6] B. W. Jackson. "Universal Cycles of K-Subsets and k-Permutations". En: *Discrete Math.* 117, nº1–3 (1993), 141–150. ISSN: 0012-365X. DOI: 10.1016/0012-365X(93)90330-V.

[7] J. Robert Johnson. "Universal cycles for permutations". En: *Discrete Mathematics* 309 (2009). ISSN: 0012365X. DOI: 10.1016/j.disc.2007.11.004.

[8] Donald E. Knuth. *The Art of Computer Programming, Volume 4, Fascicle 2: Generating All Tuples and Permutations (Art of Computer Programming)*. Addison-Wesley Professional, 2005. ISBN: 0201853930.

[9] M. H. Martin. "A problem in arrangements". En: *Bulletin of the American Mathematical Society* 40 (1934), 859 – 864.

[10] Frank Ruskey y Aaron Williams. "An explicit universal cycle for the (n-1)-permutations of an n-set". En: *ACM Transactions on Algorithms* 6 (2010). DOI: 10.1145/1798596.1798598.

[11] Arthur T. White. "Fabian Stedman: The First Group Theorist?" En: *The American Mathematical Monthly* 103.9 (1996), págs. 771-778. DOI: 10.1080/00029890.1996.12004816.

[12] Aaron Williams. "Loopless Generation of Multiset Permutations Using a Constant Number of Variables by Prefix Shifts". En: *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '09. New York, New York: Society for Industrial y Applied Mathematics, 2009, 987–996. URL: https://dl.acm.org/doi/10.5555/1496770.1496877.