Universidad de Murcia

Facultad de Matemáticas

Máster en Matemática Avanzada

Trabajo fin de máster

Teoría de grupos y criptografía

Realizado por Paula Foubelo Lillo

Dirigido por Ángel del Río Mateos

Junio 2019

Declaración de originalidad

Paula Foubelo Lillo, autora del TFM titulado *Teoría de grupos y criptografía*, bajo la tutela del profesor Ángel del Río Mateos,

DECLARA

que el trabajo que presenta es original, en el sentido de que ha puesto el mayor empeño en citar debidamente todas las fuentes utilizadas.

En Murcia, a 29 de junio de 2019.

Fdo: Paula Foubelo Lillo

Nota: En la Secretaría de la Facultad de Matemáticas se ha depositado una copia firmada de esta declaración.

Introducción

Imaginemos que dos personas quieren intercambiar información sin que una tercera intercepte el mensaje. Podríamos pensar en utilizar un lenguaje secreto, y eso exactamente es lo que hace la criptografía. Transforma la información en algo que aparentemente no tiene sentido, intentando confundir. En nuestro caso, ese lenguaje secreto se estudia a través de las matemáticas, que permiten formalizar las ideas para poder trabajar. La informática es el soporte actual, debido a la gran cantidad de información que se maneja, y porque permite aplicar técnicas que de otra forma serían sólo conceptos teóricos. Sin embargo, la criptografía es muy anterior. Ya la utilizaba Julio César para enviar mensajes militares. Pero había un problema: las dos partes debían ponerse de acuerdo en una clave, o bien utilizando a alguien de confianza, o en persona. Lo cual no era siempre posible. Aquí tenemos una de las motivaciones para desarrollar la criptografía de clave pública.

En 1976 con la publicación del artículo *New directions in cryptography*, por Whitfield Diffie y Martin Hellman, cambia la manera en la que funcionaban los criptosistemas. En este artículo se trató el problema principal de la criptografía hasta el momento ¿cómo intercambiar las claves de manera segura? En él se presenta el intercambio de claves de Diffie-Hellman. Para explicar cómo funciona, primero nos ponemos en contexto. Hay dos personas, que en los textos de criptografía se suelen llamar Alice y Bob, que quieren intercambiar información sin que una tercera persona, que se suele llamar Eve, se entere. Así, Alice y Bob tiene que ponerse de acuerdo en una clave común. Según el protocolo de intercambio de claves de Diffie-Hellman se daría lo siguiente:

- Alice y Bob se ponen de acuerdo en un grupo cíclico finito G con g generador del grupo.
- Alice elige al azar un número natural, a y envía g^a a Bob.
- Bob elige al azar un número natural, b y envía g^b a Alice.
- Alice calcula $K_A = (g^b)^a$ y Bob a su vez calcula $K_B = (g^a)^b$.

Como $\mathbb{Z}$ es un grupo conmutativo, $ab = ba$ por lo que $K_A = K_B$. De esta forma Alice y Bob obtienen una clave que será la que utilicen para comunicarse. Inicialmente G es el semigrupo multiplicativo $\mathbb{Z}_p$ con p un primo, pero en principio serviría cualquier grupo G que cumpla las condiciones. Este protocolo se considera seguro ya que basa su seguridad en el problema del logaritmo discreto, que resulta difícil de resolver si se eligen los parámetros de manera adecuada. A partir de aquí se plantean muchos protocolos, algunos de ellos en grupos. Y esto conduce a la aparición de problemas computacionales, siendo algunos de ellos de interés en teoría de grupos. Es aquí dónde relacionamos teoría de grupos con teoría de la complejidad.

En teoría de la complejidad se trata de averiguar cómo de difíciles son los problemas computacionales, pero nos interesa estudiarlo de manera precisa, por lo que necesitaremos un marco de trabajo, con definiciones y resultados. Al trabajar en protocolos que utilizan grupos necesitaremos introducir conceptos de teoría de grupos combinatoria. Por ejemplo, en un grupo la forma que tenemos de escribir los elementos es como producto de los generadores, por este motivo introducimos el concepto de presentación. Por otro lado, cuando tratemos un problema computacional, dividiremos las instancias, o inputs, según su tamaño. Para ello utilizaremos el concepto de estratificación, que va unido a la función de tamaño. Estudiaremos cómo crece el número de operaciones que hay que hacer para resolver un problema a medida que crece la complejidad del problema.

Hemos dicho que la teoría de la complejidad trata de clasificar los problemas según su dificultad, pero no hemos dicho cómo medir esta dificultad. La forma clásica consiste en el estudio del peor caso, es decir, el caso que requiere más cálculos. Sin embargo, hay problemas que se sabe que son difíciles en los peores casos (todos los problemas se pueden reducir a ellos) y aún así en la mayoría de casos son fáciles. Por este motivo miramos la complejidad desde otro punto de vista, introduciendo así los conceptos de complejidad media y de complejidad genérica. Finalmente nos centraremos en un protocolo concreto, el de Anshel-Anshel-Goldfeld, y lo analizamos en distintos grupos.

El trabajo está dividido en cuatro capítulos, más una introducción, la bibliografía y el índice terminológico.

El primer capítulo, **Preliminares**, está dividido en dos secciones. En la primera tenemos una breve introducción a la teoría de grupos combinatoria en la que introducimos el concepto de grupo libre, la presentación de un grupo, las formas normales y las condiciones deseables para que un grupo pueda ser utilizado como plataforma en un protocolo de intercambio de claves. En la segunda sección, damos nociones básicas de teoría de la complejidad. Introducimos el modelo de computación que utilizaremos, la complejidad del peor caso, así como una serie de conceptos que nos pondrán en contexto los siguientes capítulos.

En el segundo capítulo, **Complejidad media y genérica**, introducimos dos nuevas formas de medir la complejidad de un problema: complejidad media y complejidad genérica. En la primera sección nos centramos en la complejidad media, estimando la eficiencia del algoritmo según el tiempo que esperamos que vaya a tardar en resolver el problema. Mientras que en la segunda sección tratamos la complejidad genérica, que estudia el comportamiento del algoritmo en la mayoría de los casos para los conjuntos de inputs más probables. Tanto la complejidad media como la genérica resultan más efectivas en la práctica que la complejidad por peor caso. Aún así veremos que lo más conveniente, al menos en criptografía, es considerar la complejidad genérica.

En el tercer capítulo, **Complejidad algorítmica en grupos**, nos centramos en el estudio del comportamiento de los algoritmos cuando sus instancias aumentan de tamaño. Esta es la esencia de la teoría de complejidad, es decir, el objetivo es saber cómo de deprisa crece la dificultad de resolver un problema cuando el tamaño de los inputs crece. El capítulo está dividido en cuatro secciones. En la primera y la segunda sección definimos dos clases de grupos en las que nos será más fácil trabajar, sobretodo en el último capítulo. La tercera sección nos permite agrupar los problemas computacionales que trataremos, así como establecer relaciones entre algunos de ellos mediante resultados de interés. En la última sección describimos el protocolo Anshel-Anshel-Goldfeld, que será el que estudiaremos en el siguiente capítulo.

Finalmente, en **Ataques de longitud y ataques cociente** nos centramos en el protocolo de intercambio de claves de Anshel-Anshel-Goldfeld, explicando dos tipos de ataque que resultan ser efectivos en grupos libres. Por lo que si aplicamos este protocolo, no es recomendable elegir un grupo libre como plataforma.

Índice general

Capítulo 1

Preliminares

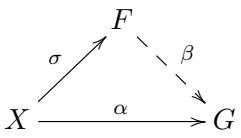
En este capítulo introducimos los conceptos básicos de Teoría de Grupos y complejidad necesarios para comprender el resto de la memoria. Entendemos que el lector conoce la terminología más elemental de Teoría de Grupos. En este capítulo hemos seguido esencialmente [11] y [14].

1.1. Teoría de grupos combinatoria

En esta sección lo primero que haremos será dar la definición de grupo libre, de manera abstracta, después desarrollaremos el concepto de presentación de un grupo y también introduciremos los conceptos de grupo plataforma y forma normal.

1.1.1. Grupos libres

Sea F un grupo, X un conjunto no vacío y $\sigma : X \to F$ una función. Entonces F, o más bien (F, σ), es un grupo libre en X si a cada función α de X a un grupo G le corresponde un único homomorfismo $\beta : F \to G$ tal que $\alpha = \beta\sigma$, o sea, sólo hay un único homomorfismo β para el que el siguiente diagrama es conmutativo:



En tal caso σ es necesariamente inyectiva pues si $\sigma x_1 = \sigma x_2$ con x_1 y x_2 elementos de X y G un grupo con al menos dos elementos distintos g_1 y g_2, tomamos $\alpha : X \to G$ una función tal que $\alpha(x_1) = g_1$ y $\alpha(x_2) = g_2$. Si β el homomorfismo de la definición entonces $g_1 = \alpha(x_1) = \beta\sigma(x_1) = \beta\sigma(x_2) = \alpha(x_2) = g_2$. Esto prueba que en efecto σ es inyectiva. Esto implica que F es libre en $Im(\sigma)$, ya que podemos tomar σ como la inclusión $Im(\sigma) \overset{\text{i}}{\hookrightarrow} F$ y veremos en el siguiente resultado que F está generado por $Im(\sigma)$. Diremos que un subconjunto X de F es una *base libre* para el grupo F si (F, σ) es un grupo libre con $\sigma : X \hookrightarrow F$ la inclusión. Acabamos de ver cómo se definen los grupos libres, pero no sabemos si existen o no. Para ello tenemos el siguiente teorema que nos indica cómo obtener un grupo libre a partir de un conjunto X.

Teorema 1.1.1 *Si X es un conjunto no vacío, existe un grupo F y una función $\sigma : X \to F$ tal que (F, σ) es libre en X. En tal caso, $F =< Im(\sigma) >$ y por tanto $Im(\sigma)$ es una base libre de F.*

Demostración: Tomamos el conjunto $X^{-1} = \{x^{-1} : x \in X\}$, disjunto con X y en correspondencia biunívoca con X mediante la aplicación $x \to x^{-1}$ y convenimos que $(x^{-1})^{-1} = x$. Definimos una *palabra* en X como una sucesión finita de símbolos en $X \cup X^{-1}$ que escribimos como

$$w = x_1 \cdots x_r$$

con $x_i \in X \cup X^{-1}$, $r \geq 0$. Si $r = 0$ la secuencia es vacía y w es la palabra vacía que denotaremos por 1. Dos palabras son iguales si y sólo si tienen los mismos elementos en las posiciones correspondientes.

El producto de dos palabras $w = x_1 \cdots x_r$ y $v = y_1 \cdots y_s$ se define como su yuxtaposición:

$$wv = x_1 \cdots x_r y_1 \cdots y_s$$

De esta manera $1w = w = w1$. Para cada palabra $w = x_1^{\epsilon_1} \cdots x_n^{\epsilon_n}$ con $x_i \in X$ y $\epsilon_i = \pm$ denotamos $w^{-1} = x_r^{-\epsilon_r} \cdots x_1^{-\epsilon_1}$ a su inverso, con $1^{-1} = 1$.

Denotamos por S al conjunto de todas las palabras en X y definimos una relación de equivalencia en S de la siguiente manera: dos palabras w y v son equivalentes, $w \sim v$, si es posible pasar de una palabra a otra mediante una secuencia finita de operaciones del tipo:

(a) Insertar xx^{-1} ó $x^{-1}x$, con x en X, como elementos consecutivos de una palabra.

(b) Eliminar una aparición de xx^{-1} ó $x^{-1}x$ en una palabra.

Esto define una relación de equivalencia en S y a la clase de equivalencia de w la denotaremos por $[w]$.

Definimos F como el conjunto de clases de equivalencia. Queremos ver que F es un grupo y para ello definimos el producto de $[w]$ y $[v]$ como $[w][v] = [wv]$. Esto tiene sentido ya que si $w \sim w'$ y $v \sim v'$ entonces $wv \sim w'v'$. Además $[w][1] = [w] = [1][w]$, por lo que $[1]$ es el elemento neutro, y $[w][w^{-1}] = [ww^{-1}] = [1]$, luego $[w^{-1}]$ es el inverso de $[w]$. También se cumple la propiedad asociativa, ya que $(wv)u) = w(vu)$ y de ahí $([w][v])[u] = [(wv)u] = [w(vu)] = [w]([v][u])$. Queda comprobado que F es un grupo.

Definimos la función $\sigma : X \to F$ como $\sigma(x) = [x]$. Tenemos que probar que (F, σ) es libre en X. Supongamos que $\alpha : X \to G$ es una función de X en un grupo G. Sea $\bar{\beta}$ una función que va de S, conjunto de palabras en X, a G tal que $\bar{\beta}(x_1^{\epsilon_1} \cdots x_r^{\epsilon_r}) = g_1^{\epsilon_1} \cdots g_r^{\epsilon_r}$ con $g_i = \alpha(x_i)$.

Ahora, si $w \sim v$ entonces $\bar{\beta}(w) = \bar{\beta}(v)$ ya que gg^{-1} y $g^{-1}g$ representan al 1_G en G. De esta manera podemos definir una función $\beta : F \to G$ como $\beta([w]) = \bar{\beta}(w)$, así $\beta([w][v]) = \beta([wv]) = \bar{\beta}(wv) = \bar{\beta}(w)\bar{\beta}(v) = \beta([w])\beta([v])$, por lo que β es un homomorfismo de F en G.

Además, para x en X, $\beta\sigma(x) = \beta([x]) = \bar{\beta}(x) = \alpha(x)$ por como hemos definido $\bar{\beta}$. Finalmente si $\gamma : F \to G$ es otro homomorfismo tal que $\sigma\gamma = \alpha$ entonces $\sigma\gamma = \sigma\beta$, por lo que γ y β son iguales en $Im(\sigma)$, pero como $F =< Im(\sigma) >$, tenemos que $\gamma = \beta$. ■

En la demostración anterior hemos podido ver la definición de palabra, así como de su inverso o de la palabra vacía, vemos ahora los conceptos de palabra reducida y cíclicamente reducida:

Una palabra $w = y_1 \cdots y_n$ es *reducida* si para cualquier $i = 1, \ldots, n-1$ se tiene que $y_i \neq y_{i-1}^{-1}$, es decir, w no contiene ninguna subpalabra de la forma yy^{-1} para cualquier $y \in X^{\pm 1}$.

Una palabra $w = y_1 \cdots y_n$ es *cíclicamente reducida* si no empieza simultáneamente con y_i y acaba con y_i^{-1}.

La longitud de una palabra $w = x_1 \cdots x_n$ en X es n. Por convenio la longitud de la palabra vacía es 0. Un elemento de $F(X)$ puede estar representado por palabras de distintas longitudes, pero el siguiente teorema nos dice que tiene exactamente una representación mediante una palabra reducida. Esto nos permite definir la longitud $|w|$ o $|w|_X$ de un elemento de $F(X)$ como la longitud de la única palabra reducida que lo representa.

Teorema 1.1.2 *Cada clase de equivalencia de palabras en X contiene una única palabra reducida.*

Demostración: Es obvio que cada elemento de $F(X)$ está representado por al menos alguna palabra reducida. Sea R el conjunto de todas las palabras reducidas en X. Para cada u en $X \cup X^{-1}$ definimos la función $u' : R \to R$ como

$$u'(x_1^{\epsilon_1} \cdots x_r^{\epsilon_r}) = \begin{cases} u x_1^{\epsilon_1} \cdots x_r^{\epsilon_r} & \text{si} \quad u \neq x_1^{-\epsilon_1} \\ x_2^{\epsilon_2} \cdots x_r^{\epsilon_r} & \text{si} \quad u = x_1^{-\epsilon_1} \end{cases}$$

dónde $x_1^{\epsilon_1} \cdots x_r^{\epsilon_r}$ es una palabra reducida. Entonces u' es una permutación de R ya que $(u^{-1})'$ es su inverso. Utilizaremos una función de X en S_R (grupo simétrico de R) con $x \mapsto x'$ y la propiedad de los grupos libres para construir un homomorfismo

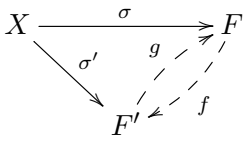
$$\theta : F \to S_R \text{ con } \theta([x]) = x'$$

Sean v y w dos palabras reducidas equivalentes. Entonces $[v] = [w]$ y $\theta([v]) = \theta([w])$. Si $v = x_1^{\epsilon_1} \cdots x_r^{\epsilon_r}$, $[v] = [x_1^{\epsilon_1}] \cdots [x_r^{\epsilon_r}]$ y $\theta([v]) = (x_1^{\epsilon_1})' \cdots (x_r^{\epsilon_r})'$ Aplicando $\theta([v])$ a la palabra vacía 1, obtenemos $x_1^{\epsilon_1} \cdots x_r^{\epsilon_r} = v$, ya que v es reducida. Análogamente, $\theta([w])$ manda 1 en w, por lo que $w = v$. ■

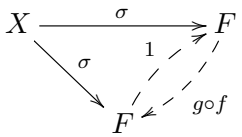
Para acabar esta sección tenemos 3 resultados más sobre grupos libres. El primero implica que dos grupos libres sobre un conjunto X son isomorfos. Esto nos permite usar sin ambigüedad (salvo isomorfismos) la notación $F(X)$ para representar el "grupo libre" en X.

Teorema 1.1.3 *Sean F y F' dos grupos, X un conjunto no vacío y $\sigma : X \to F$, $\sigma' : X \to F'$ dos funciones tales que (F, σ) y (F', σ') son grupos libres en X. Entonces F y F' son grupos isomorfos.*

Demostración: Tenemos el siguiente diagrama



Tomando la composición $g \circ f$ obtenemos



Obsérvese que 1_F y $g \circ f$ satisfacen la condición del homomorfismo. Por la unicidad en la definición obtenemos $g \circ f = 1$. Análogamente se obtiene $f \circ g = 1$. Se concluye que F y F' son grupos isomorfos. ■

El teorema anterior implica que si X e Y son conjuntos del mismo cardinal, los grupos libres en X e Y son isomorfos. El recíproco también se verifica pero la demostración excede el objetivo de este trabajo.

Teorema 1.1.4 *Sean F y F' grupos libres en X e Y respectivamente, entonces $F \simeq F'$ si y sólo si $|X| = |Y|$.*

Demostración: Ver [14] ■

Este resultado nos permite definir el *rango* de un grupo libre como el cardinal de cualquier conjunto en el que sea libre.

El siguiente teorema implica que todo grupo es la imagen de un grupo libre.

Teorema 1.1.5 *Sea G un grupo generado por un subconjunto X. Entonces G es isomorfo a un cociente de $F(X)$.*

Demostración: Podemos extender la inclusión a un homomorfismo de $F(X)$ en G que será un epimorfismo por ser X conjunto generador de G. ■

Damos ahora la definición de conjunto *reducido de Nielsen*, que necesitaremos en los últimos capítulos.

Definición 1.1.6 *Sea Y un conjunto de elementos no triviales de $F(X)$, decimos que Y es reducido de Nielsen respecto a la base libre X si se cumplen:*

(1) *Si $u, v \in Y \cup Y^{-1}, u \neq v^{-1}$ entonces $|uv|_X \geq |u|_X, |uv|_X \geq |v|_X$.*

(2) *Si $u, v, w \in Y \cup Y^{-1}, u \neq w^{-1}, v \neq w^{-1}$ entonces $|uwv|_X > |u|_X + |v|_X - |w|_X$.*

La primera condición nos asegura que no se cancela más de la mitad de u ni de v al hacer el producto uv. Mientras que la segunda condición dice que al menos una letra de w sobrevive después de las cancelaciones que se producen al calcular uwv.

El siguiente teorema será de gran utilidad más adelante y está relacionado tanto con grupos libres como con el concepto de reducido de Nielsen.

Teorema 1.1.7 *Sea $Y = \{y_1, \ldots, y_k\}$ un conjunto de elementos no triviales de $F(X)$. Si Y es reducido de Nielsen respecto a la base libre X entonces Y genera un subgrupo libre de rango k.*

Demostración: Ver [11]. ■

1.1.2. Presentación de un grupo

Hemos visto que cada grupo se puede obtener como la imagen de un grupo libre. Esto nos va a permitir describir grupos a partir de lo que se llama presentación de un grupo.

Sea G un grupo. Hay dos formas alternativas de definir lo que es una presentación de G, que en realidad son dos formas de decir lo mismo. Hemos visto que existe un homomorfismo suprayectivo $f : F(X) \to G$ para un conjunto X. Decimos entonces que la sucesión exacta $1 \to Ker(f) \to F(X) \to G \to 1$ es una *presentación* de G. En tal caso $G \simeq F(X)/Ker(f)$. La otra forma de dar la definición de una presentación es dando un subconjunto S de $F(X)$ tal que $Ker(f)$ es la clausura normal de S en $F(X)$, que denotaremos por S^F. En tal caso se dice que $< X|S >$ es una presentación de G, o sea,

$$< X|S >= F(X)/S^F$$

Obsérvese que G es isomorfo a $F(X)/S^F$ con lo que efectivamente de ambas maneras G queda completamente determinado salvo isomorfismo.

Los elementos de S se llaman *relaciones* de la presentación $< X|S >$. En realidad los elementos de S^F también se llaman relaciones cuando pensamos en la primera definición. Diremos que una presentación $< X|S >$ es *finita* cuando el conjunto de generadores X y el conjunto de relaciones S sean ambos finitos.

1.1.3. Grupos plataforma y formas normales

En esta subsección vamos a mencionar unos conceptos que no son estrictamente matemáticos pero que son importantes en las aplicaciones que estudiaremos. La criptografía consiste en intercambiar información de manera segura, por lo que saber ocultarla resulta fundamental. Sin embargo, en este caso, aunque parezca contradictorio para poder ocultar información primero es necesario organizarla de manera que podamos trabajar con ella, verla con claridad. Además, al final necesitamos que el receptor elegido pueda comprender el mensaje enviado, con lo que es

necesario que se puedan describir los elementos del mensaje y el proceso criptográfico de forma exacta y precisa. Para esto utilizamos las *formas normales*, mecanismos que permiten transformar las instancias de un problema en representaciones únicas y precisas de los elementos de forma que las operaciones se puedan realizar de forma clara y podamos distinguir elementos distintos. Las formas normales deben cumplir dos propiedades:

1. Cada objeto debe tener una única forma normal.
2. Dos objetos con la misma forma normal deben ser iguales.

Dado que la forma normal debe ser algo que nos ayude a manejar la información, la elegiremos dependiendo de la naturaleza del problema a tratar.

Ejemplo 1.1.8 *Sea C_n grupo cíclico de orden n con g elemento generador. Los elementos de C_n tienen formas normales, que son g^i con $i = 0, \ldots, n-1$.*

Ejemplo 1.1.9 *Sea $D_{2n} =< a, b | a^n = 1 = b^2, a^b = a^{-1} >$ el grupo diédrico. Las formas normales de sus elementos son $1, a, a^2, \ldots, a^{n-1}, b, ab, a^2b, \ldots, a^{n-1}b$. También podemos usar $1, a, \ldots, a^{n-1}, b, ba, \ldots, ba^{n-1}$ teniendo en cuenta que $ba^i = a^{-i}b$.*

Los dos ejemplos anteriores pueden hacer pensar que sólo los grupos finitos tienen formas normales. Eso no es correcto.

Ejemplo 1.1.10 *Tanto en $\mathbb{Z}$ como en $\mathbb{Q}$ podemos encontrar formas normales que permiten distinguir y a la vez "esconder" sus elementos, mientras que en $\mathbb{R}$ nos resulta imposible.*

Ejemplo 1.1.11 *Dado F un grupo libre, por el Teorema 1.1.2, cada uno de sus elementos se puede escribir de manera única como $[w]$ dónde $w = x_1 \cdots x_r$ es una palabra reducida, con $r \geq 0$. Identificando w con $[w]$ tenemos que*

$$w = x_1 \cdots x_s$$

es una forma normal para w.

A la hora de utilizar un protocolo de intercambio de claves necesitamos un lugar de trabajo, es decir, un entorno en el que elegir elementos y operar con ellos. En nuestro caso, este entorno será un grupo. Pero, ¿sirve cualquier grupo? Pues en principio si, a no ser que se especifique lo contrario. Esto no quiere decir que en todos los grupos un protocolo determinado sea igual de eficiente o seguro, para esto debemos estudiar el protocolo y de acuerdo con sus propiedades elegir el grupo, la plataforma, que nos ofrezca una mayor seguridad. En cuanto a lo que le pedimos a

un grupo G para que sea la plataforma en un protocolo (no especificamos cual ahora, sino que nos referimos a uno en general) debe cumplir las siguientes propiedades:

($GP0$) El grupo debe ser conocido y haber sido estudiado.

($GP1$) Debemos ser capaces de distinguir sus elementos. En otras palabras: sus elementos deben tener una forma normal que sea fácil de obtener y existe una manera eficiente de distinguir formas normales.

($GP2$) Hay un método eficiente de calcular la forma normal de xy a partir de la de dos elementos x e y de G. Si el grupo G aparece en función de sus generadores y de las relaciones entre ellos, sin forma normal, entonces pedimos que sus relaciones sean cortas o al menos tenemos que tener en cuenta que en la práctica sólo se usarán elementos de una longitud limitada.

($GP3$) El grupo G debe tener crecimiento superpolinomial, que quiere decir que el número de elementos en G de tamaño n, en su forma normal, debe crecer más rápido que cualquier polinomio en n. Esto impide ataques que se centran en inspeccionar el espacio de claves al completo, ya que de esta manera, el espacio de claves será demasiado grande. Esta condición la entenderemos mejor cuando hayamos introducido los conceptos de crecimiento en el Capítulo 2.

1.2. Introducción a la complejidad computacional

En esta sección introduciremos algunos conceptos que nos harán falta más adelante, sobre todo a la hora de definir cómo de complejo es un algoritmo, es decir, si es eficaz. A lo largo del trabajo haremos referencia a dos tipos de problemas que introducimos aquí por primera vez. Estos problemas se pueden dividir en dos tipos:

1. *Problemas de decisión*: Dada una propiedad $\mathbb{P}$ y un objeto $\mathbb{O}$, comprobar si $\mathbb{O}$ cumple la propiedad $\mathbb{P}$.
2. *Problemas de búsqueda*: Dada una propiedad $\mathbb{P}$ y la información de que hay objetos que cumplen esta propiedad, encontrar al menos un objeto que la cumpla.

Introducimos también la siguiente notación, que utilizaremos a lo largo del trabajo. Dadas dos funciones $f, g : \mathbb{R} \to \mathbb{R}$ ó $f, g : \mathbb{N} \to \mathbb{R}$ introducimos la siguiente notación:

- $g = O(f)$ (ó $g(x) = O(f(x))$ si existen $x_0, c > 0$ tales que para todo $x \geq x_0 > 0$ se tiene $0 \leq |g(x)| \leq c|f(x)|$.
- $g = o(f)$ (ó $g(x) = o(f(x))$) si existen $x_0, c > 0$ tales que para todo $x \geq x_0 > 0$ se tiene $0 \leq c|f(x)| \leq |g(x)|$, es decir, $\lim_{x \to x_0} \frac{f(x)}{g(x)} = 0$.
- $f \sim g$ si $\lim_{n \to \infty} \frac{f(n)}{g(n)} > 0$.

A continuación introducimos las máquinas de Turing, que será lo que utilicemos como principal modelo de computación.

1.2.1. Máquinas de Turing

¿Qué es un problema computacional? Hemos hablado de los tipos de problemas que vamos a utilizar pero sin dar una definición de qué consideramos un problema, quizás porque creemos que es evidente. En matemáticas se trabaja con definiciones formales, con objetos que cumplen propiedades y que podemos estudiar, por lo que en este caso también necesitamos saber qué es un *problema* (computacional) desde un punto de vista abstracto para poder estudiarlo.

En teoría de la computación un problema computacional es un objeto matemático que representa una serie de preguntas que un ordenador puede ser capaz de responder o la búsqueda de un objeto que satisface unas condiciones. Por ejemplo, el problema de la 3-satisfacibilidad, en el que dada una expresión formada por cláusulas de como máximo 3 variables cada una, se busca una asignación booleana que haga que la expresión sea cierta. El ordenador trabaja con algoritmos, métodos que permiten resolver estos problemas de manera más o menos eficaz. No hemos dicho qué significa que un método sea eficaz o que un problema sea resoluble, para ello introducimos las máquinas de Turing.

Las máquinas de Turing son un modelo abstracto de computación que proporcionan una definición formal y precisa de lo que significa que una función sea computable. Pensamos en ellas como el objeto en el que visualizar el funcionamiento de un algoritmo. Las claves de este modelo de computación son:

1. Cantidad finita de estados.
2. Cantidad infinita de almacenamiento externo de datos.
3. Un programa especificado por un número finito de instrucciones en un lenguaje predeterminado.

4. Auto-referencia: que el lenguaje de programación sea lo suficientemente expresivo como para interpretar sus programas.

De manera informal pensamos en una máquina de Turing como una máquina que es capaz de leer lo escrito en una cinta de longitud infinita, en la que están escritos símbolos pertenecientes a un alfabeto finito. La cinta está formada por una sucesión numerada de casillas que identificamos con $\mathbb{N}$. Dependiendo del símbolo que lea y del estado en que se encuentre, la máquina escribe un nuevo símbolo y se mueve a la izquierda de la cinta, a la derecha de la cinta, o se queda quieta. Además modifica (o no) un estado y vuelve a repetir el proceso. También puede ocurrir que pare y de manera opcional proporcione una respuesta Sí, No o de otro tipo. La aplicación que determina el comportamiento de la máquina es la función de transición.

Si nos referimos a la complejidad temporal o espacial de una máquina de Turing, la primera hace referencia al número de veces que se mueve la cinta o lee un símbolo cuando la máquina trabaja mientras que la segunda nos indica el número de celdas o espacios que la máquina ha escrito. Estos dos conceptos están relacionados, ya que si para un input w tenemos que la complejidad espacial es $f(w)$ entonces la temporal debe ser al menos de $f(w)$, ya que para escribir $f(w)$ celdas debe haberse movido al menos $f(w)$ veces. Nosotros sólo consideraremos la complejidad temporal.

En los siguientes apartados daremos definiciones formales.

Máquinas de Turing deterministas

Definición 1.2.1 *Una* máquina de Turing *(TM) de cinta infinita (one-tape) M es una 5-tupla $(Q, \Sigma, s, f, \delta)$ donde:*

- *Q es un conjunto finito cuyos elementos llamamos* estados.
- *Σ es un conjunto finito que representa el* alfabeto *de la cinta.*
- *s y f son elementos de Q que representan los* estados inicial *y* final *respectivamente.*
- *δ es una aplicación $\delta : Q \times \Sigma \longrightarrow Q \times \Sigma \times \{\leftarrow, \rightarrow, -\}$ que llamamos* función de transición *y será la que determine el comportamiento de la máquina.*
- *El conjunto Σ contiene dos símbolos especiales, $\triangleright$ y $\sqcup$ que representan respectivamente el inicio y el final de la cinta escrita. Para cada estado q en Q si $\delta(q, \triangleright) = (p, \sigma, d)$, entonces $\sigma = \triangleright$ y $d \neq \leftarrow$ (es decir, la máquina no intenta reescribir el símbolo inicial ni moverse a su izquierda) Sin embargo sí que podemos reescribir el símbolo final o movernos a su derecha,*

esto será necesario cuando la computación requiera más espacio del dado en la cadena de inputs.

Para ver cómo funciona la máquina necesitamos definir su configuración en todo momento (estado de la cinta, estado de la máquina y posición en la cinta) así como las reglas que determinan sus movimientos.

Denotamos por Σ^* al conjunto de todas las secuencias finitas de elementos de Σ. A un elemento de Σ^* lo denotamos por x y a los elementos de la secuencia x los denotamos por $x_0, x_1, \ldots, x_{n-1}$ siendo n la longitud de x que denotamos por $|x|$, o sea, $|x| = n$.

Una *configuración* de la máquina de Turing $M = (Q, \Sigma, s, f, \delta)$ es un vector $(x, q, k) \in \Sigma^* \times Q \times \mathbb{N}$ dónde x es la cadena escrita en la cinta en un instante, q el estado en ese instante y k la posición de la máquina en la cinta. Exigimos que x empiece por $\triangleright$ y acabe por $\sqcup$, y que $0 \leq k < |x|$.

Dada una configuración (x, q, k) en un momento dado, su configuración en el paso siguiente (x', q', k') vendrá dada de la siguiente manera: sea $\delta(q, x_k) = (p, \sigma, d)$. La cadena x' se obtiene a partir de x cambiando x_k por σ. Si además $k = |x| - 1$ y $\sigma \neq \sqcup$ entonces añadimos $\sqcup$ al final de x. El nuevo estado q' es p y $k' = k-1, k+1, k$ dependiendo de si d es $\leftarrow, \rightarrow$ ó $-$ respectivamente. Este cambio lo denotamos por $(x, q, k) \xrightarrow{M} (x', q', k')$. Podemos definir la relación "produce en r-pasos" denotándola por $\xrightarrow{M^r}$ y la relación "produce" (en un número indeterminado de pasos), $\xrightarrow{M^*}$.

Una computación en M es una secuencia de configuraciones (x_i, q_i, k_i) dónde $0 \leq i \leq T$ que cumplen:

- La máquina empieza en una configuración inicial (x, q_0, k_0) donde $q_0 = s$ y $k_0 = 0$. Entendemos x como el input o entrada representando una instancia del problema en cuestión.
- Cada par consecutivo de configuraciones representa una transición válida, i.e, para $0 \leq i < T$, $(x_i, q_i, k_i) \xrightarrow{M} (x_{i+1}, q_{i+1}, k_{i+1})$.
- Si $T = \infty$ decimos que la computación "no para".
- Si $T < \infty$ se requiere $q_T = f$. Entonces decimos que la computación "para" en tiempo T.

En tal caso interpretamos la palabra $M(x)$, que resulta de eliminar los símbolos inicial $\triangleright$ y final $\sqcup$ de la cadena x_T, como el output de la máquina. Decimos que M se detiene en una cadena

x en Σ^* si la configuración $(x, s, 0)$ produce una configuración (x', f, k'), en un número finito de pasos que denotamos por $T_M(x)$.

Diremos que M *resuelve el problema de decisión* D definido sobre un alfabeto Σ si M se detiene en cada entrada x en Σ^* devolviendo *Sí* o *No*. O sea, si el output $M(x)$ representa nuestra forma de escribir *Sí* o *No*. De manera análoga diremos que M resuelve un problema de cálculo de entrada x, cuyo conjunto de soluciones sea Q_x, si M devuelve un output en Q_x. O sea $M(x)$ representa en nuestro alfabeto un elemento de Q_x. Diremos que M decide parcialmente el problema D si decide correctamente en un subconjunto D' de D y en $D \setminus D'$ o bien no para o se detiene devolviendo la respuesta *No sé*. Una definición similar es posible para problemas de Búsqueda.

1.2.2. Problemas computacionales de búsqueda y decisión

Sea X un conjunto no vacío. Recordamos que X^* representa las sucesiones finitas de elementos de X.

Podemos formular el concepto de problema de decisión de la siguiente manera: para un subconjunto $L \subseteq I$ de X^* *¿Hay algún algoritmo que dada una palabra $w \in I$ determine si w pertenece a L o no?* Si este algoritmo existe lo llamamos *algoritmo de decisión* para L con conjunto de instancias I, y decimos que el problema de decisión (L, I) es *decidible*. De manera más formal, un problema de decisión viene dado por un par $D = (L, I)$ con $L \subset I \subseteq X^*$. El conjunto L será la parte positiva del problema y su complemento $\bar{L}(D) = I \setminus L$ será la parte negativa. Un caso particular se da cuando $I = X^*$. Si no existen algoritmos de decisión para D decimos que el problema *no es decidible*. Otra forma más general en que pueden aparecer estos problemas es cuando $D = (L, I)$ donde $L \subset I$ e I es un subconjunto del producto cartesiano $X^* \times \cdots \times X^*$, de $k \geq 1$ copias de X^*.

Un problema de búsqueda se puede describir como un subconjunto D del producto cartesiano $I \times Y^*$ con $I \subseteq X^*$. El conjunto X es el alfabeto en el que escribimos las instancias del problema e Y en el que escribimos las soluciones. Además tenemos que las soluciones de una instancia $x \in I$ son los elementos $y \in Y^*$ de forma que $(x, y) \in D$. Por ejemplo, el problema diofántico para los enteros, en el que dado un polinomio con coeficientes enteros se quiere encontrar una raíz entera. En este caso, x sería una ecuación polinomial (o una palabra que describe esta ecuación) $E_x(t) = 0$ en una k-tupla de variables t y D estaría formado por las parejas (x, y), donde x representa el polinomio en el alfabeto X e y representa una k-tupla de enteros escritos en el alfabeto Y en el que se escriben los enteros, de forma que $E_x(y) = 0$.

1.2.3. Funciones de tamaño

La complejidad consiste en decidir cómo se complica el problema al aumentar su tamaño. Por ejemplo: cuanto más grande es un número, más difícil es decidir si es primo. Pero ¿cómo medimos el tamaño de un input, o la complejidad de un algoritmo? Ahora veremos algunas de las maneras que nos permiten manejar estos conceptos, muy importantes a la hora de evaluar la eficacia de un algoritmo.

Para estudiar la complejidad de los algoritmos tomamos una máquina de Turing que resuelva el problema y analizamos $T_M(x)$ para las distintas instancias x del problema. Lo que queremos es saber cómo crece $T_M(x)$ cuando crece x, para lo que necesitamos medir x y $T_M(x)$. Por supuesto $T_M(x)$ ya tiene una medida: es el número de pasos que la máquina da hasta parar cuando el input es x. Pero necesitamos asociar a x un tamaño. Por tanto, elegiremos una función $s : I \to \mathbb{N}$ que represente de forma natural el tamaño de las instancias y a la que llamaremos *función de tamaño*. Por ejemplo, si I es el conjunto de los números naturales, $s(x)$ puede ser el número de dígitos binarios o en otra base del número x. En situaciones más complejas, s puede tener formas más complicadas. Por ejemplo, si I está formado por los subgrupos finitamente generados de un grupo libre podríamos representar los elementos de I a partir de un conjunto de generadores $g_1, \ldots, g_n$ y tomar como $s(g_1, \ldots, g_n)$ la suma de las longitudes de sus formas reducidas, o el máximo de ellas. En cualquier caso la elección de la función de tamaño tiene que representar la naturaleza del problema. Además para que esta función sea útil se tienen que cumplir las condiciones:

$C1$. Para cada $n \in \mathbb{N}$, $s^{-1}(n)$ es un subconjunto finito de I o, si tenemos una medida definida en I, un subconjunto medible de I.

$C2$. Para cada $x \in I$, $s(x)$ se puede calcular fácilmente.

1.2.4. Estratificación

La estratificación es una herramienta muy útil que utilizaremos más adelante cuando estudiemos el comportamiento asintótico de los subconjuntos formados por los inputs de un problema. Nos permitirá conocer cuándo un algoritmo es eficiente a la hora de resolver un problema. A modo de idea, la estratificación consiste en separar los inputs o entradas de un algoritmo en estratos según su tamaño para poder estudiar el problema en los diferentes estratos, a los que llamaremos bolas o esferas.

Sea $s : I \longrightarrow \mathbb{N}$ una función de tamaño que cumple las condiciones $C1$ y $C2$ vistas en las

subsección 1.2.3. Dado $n \in \mathbb{N}$, denotamos por

$$I_n = \{x \in I; s(x) = n\}$$

al conjunto de elementos en I que tienen tamaño n, y lo llamamos *esfera de radio n* .

Por otro lado, denotamos por

$$B_n(I) = \bigcup_{k=1}^{n} I_k = \{x \in I; s(x) \leq n\}$$

a la *bola de radio n*.

Llamamos *estratificación por tamaño* de I a la partición

$$I = \bigcup_{k=1}^{\infty} I_k \tag{1.1}$$

mientras que a la unión de las bolas $B_n(I)$,

$$I = \bigcup_{k=1}^{\infty} B_k(I)$$

la llamamos *descomposición por volumen* de I.

Una partición como (1.1) induce una función de tamaño en I tal que cada $x \in I$ tiene tamaño k si y sólo si $x \in I_k$. En este caso la condición $C1$ se cumplirá si y sólo si los conjuntos I_k son finitos y $C2$ si y sólo si la partición (1.1) es computable.

Las estratificaciones por tamaño y volumen nos permiten estudiar el comportamiento asintótico de los subconjuntos de I. Esto lo veremos más detalladamente en la subsección 1.2.6.

1.2.5. Medida y pseudo-medida

A menudo, para medir la complejidad de los problemas computacionales, como veremos en el siguiente capítulo, necesitaremos asignar a cada instancia una probabilidad. Haremos esto porque no todas las instancias de un problema son igual de probables, por lo que no sería realista hacer un estudio del problema sin tener esto en cuenta. De esta manera trabajaremos con una medida de probabilidad en el conjunto de instancias o con una colección de medidas de probabilidad en cada esfera I_n o bola B_n. Al estar trabajando en estratos, las esferas o las bolas, será necesario establecer una relación entre la medida y la función de tamaño. También puede ocurrir que no se tenga una medida de probabilidad definida en el conjunto de instancias pero se tenga

lo que llamamos una pseudo-medida. Este concepto nos será de utilidad cuando estudiemos el comportamiento asintótico de problemas computacionales. En esta subsección seguimos [4].

Sea I un conjunto. Una *$\sigma-$álgebra* en I es un subconjunto $\mathcal{A}$ del conjunto de las partes de I tal que $\emptyset \subset \mathcal{A}$, $\mathcal{A}$ es cerrado para complementos en I y uniones numerables. Una *medida* en I es una aplicación $\mu : \mathcal{A} \to \mathbb{R}^+$ donde $\mathcal{A}$ es una $\sigma-$álgebra de I y se cumplen: $\mu(\emptyset) = 0$ y $\mu(\sum_{n\geq 0} A_n) = \sum_{n\geq 0} \mu(A_n)$ para $(A_n)_{n\in\mathbb{N}}$ una familia de conjuntos disjuntos en $\mathcal{A}$. Si además $\mu(I) = 1$, decimos que μ es una medida de probabilidad. Los elementos de $\mathcal{A}$ diremos que son $\mu-$medibles.

Una medida μ es *atómica* o *discreta* si todos los conjuntos con un elemento son medibles. Si μ es una función de medida en I y $J \in \mathcal{A}$ un subconjunto de I, entonces la restricción de μ a los subconjuntos de J que son medibles es una medida en J. Por otro lado, si $\mu(I) \neq 0$ entonces, para $A \in \mathcal{A}$, la función $A \to \frac{\mu(A)}{\mu(I)}$ define una medida de probabilidad en I. En particular podemos asociar una medida de probabilidad μ a J tomando $\mu_J(A) = \frac{\mu(A)}{\mu(J)}$ para todo $A \subseteq I$, con $A \in \mathcal{A}$. En el caso en que μ sea una medida atómica, a la aplicación $p : I \to \mathbb{R}$ dada por $p(a) = \mu(\{a\})$ la llamamos *función de densidad*.

Sea μ una medida en un conjunto I y sea $\mathcal{A}$ la familia de conjuntos $\mu-$medibles. Si A es un subconjunto de I, denotamos por

$$\chi_A(x) = \begin{cases} 1, & x \in A; \\ 0, & x \in I \setminus A. \end{cases}$$

Una función $\mathcal{A}-$simple es una función $f : I \to \mathbb{R}$ de la forma $f = \sum_{i=1}^n a_i \chi_{A_i}$ con $a_i \in [0, +\infty)$ y $A_1, \ldots, A_n$ elementos disjuntos de $\mathcal{A}$. En tal caso, definimos

$$\int f d\mu = \sum_{i=1}^n a_i \mu(A_i).$$

Diremos que la integral $\int f d\mu$ converge si

$$\int f d\mu = \sum_{i=1}^n a_i \mu(A_i) < \infty.$$

Más generalmente si $f : I \to \mathbb{R}$ es una función cualquiera definimos

$$\int f d\mu = \sup\{\int s d\mu : s \text{ es simple y } s(x) \leq f(x) \text{ para todo } x \in I\}$$

Si $A \in \mathcal{A}$ definimos

$$\int_A f d\mu = \int f_{|A} d\mu_{|A},$$

es decir $\int_A f d\mu$ es la integral de $f_{|_A}$ sobre la medida $\mu_{|_A}$ restringida a A.

En el caso en que I sea un conjunto numerable, para definir una medida atómica basta partir de una función $p : I \to \mathbb{R}$ tal que $p(x) \geq 0$ para todo x en I y $\sum_{x\in I} p(x) = 1$. Entonces $\mu(S) = \sum_{x\in S} p(x)$ para todo $S \subseteq I$, p será la función de densidad.

Una de las ideas generales a la hora de introducir una distribución natural en I es hacerlo mediante estratificaciones. Sea $s : I \to \mathbb{N}$ una función de tamaño, diremos que una distribución atómica μ en I respeta la función s si para todo x, y en I con el mismo tamaño tienen también la misma media, es decir, si $s(x) = s(y)$ entonces $\mu(\{x\}) = \mu(\{y\})$. En tal caso a la medida μ en I la llamaremos *invariante por tamaño*, *homogénea*, *uniforme* o $s-$*invariante*. Así una distribución homogénea μ induce una distribución uniforme en cada esfera $I_k = \{x \in I; s(x) = k\}$.

Lema 1.2.2 *Sea μ medida atómica en I y s función de tamaño en I con esferas finitas I_k.*

(1) *Si μ es $s-$invariante entonces*

$$\begin{array}{rccc} d_\mu & : & \mathbb{N} & \to & \mathbb{R} \\ & & k & \mapsto & \mu(I_k) \end{array}$$

es la función de densidad de una medida atómica de probabilidad en $\mathbb{N}$.

(2) *Si $d : \mathbb{N} \to \mathbb{R}$ es la función de densidad de una medida de probabilidad atómica en $\mathbb{N}$ entonces la función $\mu : P(I) \to \mathbb{R}$ dada por*

$$\mu(A) = \sum_{n=0}^{\infty} \frac{d(s(x))|I_n \cap A|}{|I_n|}$$

es una medida de probabilidad $s-$invariante en I.

Demostración: (1) Si $k \in \mathbb{N}$ entonces

$$d_\mu(k) = \mu(I_k) = \sum_{x\in I_k} \mu(x) = \mu(x)|I_k|$$

con $\mu(x) \geq 0$ para todo x en I. Por hipótesis sabemos que μ es una medida atómica, por lo que $\mu(x)$ está definido para todo x. Además por ser $s-$invariante, todos los elementos de I_k tienen la misma medida. Con esto, $d_\mu(k) \geq 0$ para todo k en $\mathbb{N}$. Además $\sum_{k\in\mathbb{N}} d_\mu(k) = \sum_{k\in\mathbb{N}} \sum_{x\in I_k} \mu(x) = \sum_{x\in I} \mu(x) = 1$. Esto demuestra que d_μ es la función de densidad de una medida atómica en $\mathbb{N}$.

(2) Obvio. ■

Sea $D = (L, I)$ un problema computacional donde el conjunto I aparece con una colección de medidas de probabilidad $\{\mu_n\}$, cada una de ellas definida en la esfera I_n ó en B_n y en I no tiene por qué haber ninguna medida definida. En este caso a $\{\mu_n\}$ lo llamaremos *encadenamiento de distribuciones*. En el caso que las medidas μ_n estén definidas en esferas, diremos que es un *encadenamiento esférico* y si están definidas en las bolas, las denotaremos por σ_n, y será un *encadenamiento de volumen*, en el que σ_n es una distribución en la bola B_n, y σ_{n-1} es la distribución inducida en B_{n-1} por σ_n.

Ejemplo 1.2.3 *Supongamos que las esferas I_n son finitas para cada n en $\mathbb{N}$. Entonces la distribución uniforme μ_n en I_n da lugar a un encadenamiento esférico de distribuciones $\mu = \{\mu_n\}$ en I.*

Sean μ una medida de probabilidad en I y s una función de tamaño en I. Decimos que μ es *compatible* con la función de tamaño s si s es $\mu-$medible, es decir, si para cada n la esfera I_n es un conjunto $\mu-$medible. Si la distribución μ es compatible con s entonces para todo n tal que $\mu(I_n) \neq 0$, μ induce una medida μ_n en I_n dada por $\mu_n(A) = \frac{\mu(A)}{\mu(I_n)}$ para todo subconjunto medible A contenido en I_n. Es decir, induce un encadenamiento esférico de distribuciones en I.

Sea $\{\mu_n\}$ un encadenamiento esférico en I y $d : \mathbb{N} \to \mathbb{R}$ una función de densidad en $\mathbb{N}$. Dado un subconjunto R de I definimos $R_n = R \cap I_n$ para cada n en $\mathbb{N}$. Supongamos que R_n es μ_n-medible para cada n, entonces definimos

$$\mu_d(R) = \sum_n d(n)\mu_n(R_n)$$

Pasamos ahora a introducir el concepto de pseudo-medida. Una *pseudo-medida* en I es una función real no negativa $\mu : \mathcal{A} \to \mathbb{R}^+$ definida en un subconjunto $\mathcal{A} \subseteq P(I)$ tal que:

(1) $\mathcal{A}$ contiene a I y es cerrado bajo el complemento y para la unión de conjuntos disjuntos.

(2) Para cualquier par de conjuntos disjuntos $A, B \in A$,

$$\mu(A \cup B) = \mu(A) + \mu(B)$$

En particular, $\mu(\bar{A}) = 1 - \mu(A)$.

Si además $\mu(I) = 1$ diremos que μ es una *pseudo-medida de probabilidad*. Si A es una subálgebra de $P(I)$, es decir, si A es cerrado para uniones, intersecciones y complementos, entonces μ es una medida. Los elementos de A decimos que son $\mu-$medibles (o simplemente medibles si μ está clara por el contexto).

Una *pseudo-medida* μ es *atómica* si $\mu(Q)$ está definida para cualquier subconjunto finito Q de I.

Sea μ una pseudo-medida en I, decimos que un subconjunto Q de A es *genérico* si contiene a un subconjunto R de A que es medible y tal que $\mu(R) = 1$. Y diremos que un subconjunto Q de A es *despreciable* si está contenido en un subconjunto R de A que es medible y tal que $\mu(R) = 0$.

1.2.6. Densidad asintótica

Hemos introducido el concepto de pseudo-medida porque la función de densidad asintótica, que veremos ahora y que será una de las herramientas principales, en general es una pseudo-medida, pero no necesariamente una medida.

Sean I un conjunto numerable y μ una medida de probabilidad atómica en I. Si $\mu(w) > 0$ para todo w en I, entonces I es el único conjunto genérico en I. Así que en este caso las nociones de conjunto genérico y despreciable no sirven de mucho. Por el contrario, si en el conjunto I está presente una estratificación natural, la medida μ dará lugar a la densidad asintótica ρ_μ que definimos ahora.

Definición 1.2.4 *Sea I un conjunto con una función de tamaño $s : I \to \mathbb{N}^+$ y sea $\mu = \{\mu_n\}$ un encadenamiento de distribuciones esféricas para I. Sea R un subconjunto de I. La* densidad esférica asintótica *de R, respecto al encadenamiento μ es*

$$\rho_\mu(R) = \lim_{n\to\infty} \mu_n(R \cap I_n).$$

Ejemplo 1.2.5 *Supongamos que la esferas I_n son finitas para cada $n \in \mathbb{N}$. Denotamos por μ_n a la distribución uniforme en I_n. Entonces para $R \subseteq I$ tenemos que $\mu_n(R) = \frac{|R\cap I_n|}{|I_n|}$ es la frecuencia con la que aparecen elementos de R en cada esfera I_n. Por lo que $\rho_\mu(R) = \lim_{n\to\infty} \frac{|R\cap I_n|}{|I_n|}$, que indicaría con que frecuencia aparecen los elementos de R cuando las instancias tienen gran tamaño. Normalmente denotaremos por $\rho(R)$ a $\rho_\mu(R)$.*

En el siguiente ejemplo vemos que la densidad asintótica no siempre existe.

Ejemplo 1.2.6 *Sea $I = X^*$ el conjunto de todas las palabras en el alfabeto finito $X = \{x_1, \ldots, x_n\}$ y R el subconjunto de todas las palabras de longitud par. Entonces $\rho_n(R) = 1$ para n par y $\rho_n(R) = 0$ cuando n es impar, por lo que el límite $\rho(R)$ no existe.*

Una manera de garantizar la existencia de la densidad asintótica es sustituyendo $\lim_{n\to\infty}$ por $\limsup_{n\to\infty}$ en la definición. Otra forma de suavizar la definición en reemplazando las esferas

I_n por las bolas B_n. Si $\sigma = \{\sigma_n\}$ es un encadenamiento de distribuciones de volumen para I, entonces la *densidad asintótica de volumen* ρ^*_σ relativa a σ para $R \subseteq I$ es

$$\rho^*_\sigma(R) = \lim_{n\to\infty} \sigma_n(R).$$

Lema 1.2.7 *Las densidades asintóticas esférica y de volumen en I son pseudo-medidas de probabilidad en I.*

Demostración: Comprobamos que $\rho_\mu : I \to \mathbb{R}$ es una pseudo-medida en I. Sabemos I es cerrado bajo el complemento y la unión de conjuntos disjuntos en I. Para la segunda condición calculamos

$$\rho_\mu(I) = \lim_{n\to\infty} \mu_n(I \cap I_n) = \lim_{n\to\infty} \mu_n(I_n) = 1.$$

Dados A, B en I disjuntos,

$$\rho_\mu(A \cup B) = \lim_{n\to\infty} \mu_n((A \cup B) \cap I_n) = \lim_{n\to\infty} \mu_n((A \cap I_n) \cup (B \cap I_n))$$

por ser μ_n medida de probabilidad, tenemos

$$\lim_{n\to\infty} (\mu_n(A \cap I_n) + \mu_n(B \cap I_n)) = \lim_{n\to\infty} \mu_n(A \cap I_n) + \lim_{n\to\infty} \mu_n(B \cap I_n) = \rho_\mu(A) + \rho_\mu(B)$$

De forma análoga se prueba para la densidad asintótica de volumen. ■

La siguiente proposición demuestra que las densidades asintóticas esféricas y de volumen son iguales en algunas situaciones. Sin demostración uso el siguiente lema, cuya demostración puede encontrarse en [13]

Lema 1.2.8 *Sean $\{a_n\}$ y $\{b_n\}$ dos sucesiones de números reales tales que:*

- *$\{b_n\}$ es monótona creciente y divergente.*
- *$\lim_{n\to+\infty} \frac{a_{n+1}-a_n}{b_{n+1}-b_n} = \lambda$ con $\lambda \in \mathbb{R}$*

entonces $\lim_{n\to\infty} \frac{a_n}{b_n} = \lambda$.

Proposición 1.2.9 *Supongamos que todas las esferas I_n son finitas y no vacías para todo $n \in \mathbb{N}$. Sea μ medida en I que induce la distribución uniforme μ_n en cada esfera I_n (Ejemplo 1.2.5). Si la densidad esférica asintótica $\rho(R)$ existe para un subconjunto R de I, entonces la densidad de volumen estándar $\rho^*(R)$ existe y coincide con la anterior, $\rho^*(R) = \rho(R)$.*

Demostración: Sea $x_n = |R \cap B_n|$ e $y_n = |B_n|$. Entonces $y_n < y_{n+1}$ y $\lim y_n = \infty$. Entonces por el lema anterior tenemos

$$\rho^*(R) = \lim_{n\to\infty} \frac{x_n}{y_n} = \lim_{n\to\infty} \frac{x_n - x_{n-1}}{y_n - y_{n-1}} = \lim_{n\to\infty} \frac{|R \cap I_n|}{|I_n|} = \rho(R)$$

■

La elección de una medida adecuada es siempre importante pues si ésta es poco natural, los resultados que se obtienen parecen contrarios a la intuición. El siguiente ejemplo ilustra esto.

Ejemplo 1.2.10 *Sea $I = \{0,1\}^*$. Definimos el tamaño de una palabra $w \in I$, $|w|$, como su longitud. Supongamos que para cada n descomponemos I_n en dos subconjuntos disjuntos $I_n = I'_n \cup I''_n$ con $|I'_n| = |I''_n| = 2^{n-1}$. Sean*

$$I' = \bigcup_n I'_n,\ I'' = \bigcup_n I''_n$$

y definimos la función $f : I \to \mathbb{N}$ como

$$f(w) = \begin{cases} 0 & si \quad w \in I' \\ 2^{|w|} & si \quad w \in I'' \end{cases}$$

Ahora definimos la medida μ' tal que $\mu'(I'_n) = \frac{2^{n+1}-1}{2^{n+1}}$ y $\mu'(I''_n) = \frac{1}{2^{n+1}}$ Entonces I' es genérico respecto a la densidad asintótica $\rho_{\mu'}$, por lo que I'' es despreciable. También podemos definir otra medida μ'' tal que I'' sea genérico e I' sea despreciable.

1.2.7. Tasa de convergencia

Sea I un conjunto con función de tamaño s. Supongamos que para cada $n \in \mathbb{N}$ la esfera I_n y la bola B_n están equipadas con distribuciones de probabilidad μ_n y σ_n. De esta manera las densidades asintóticas ρ_μ, ρ_σ están definidas y son pseudo-medidas de probabilidad (Lema 1.2.7), por lo que podemos aplicar en ellas los conceptos de conjunto genérico y despreciable. Veremos que las densidades asintóticas no sólo permiten distinguir entre conjuntos grandes y pequeños, sino que permiten estudiar el comportamiento de los conjuntos en el infinito. A lo largo de esta subsección supondremos que $\mu = \{\mu_n; n \in \mathbb{N}\}$ es un encadenamiento de distribuciones esférico fijo. Las mismas nociones son válidas para un encadenamiento esférico.

Sea R un subconjunto de I para el que existe densidad asintótica $\rho_\mu(R)$. A la función $\delta_R : n \to \mu_n(R \cap I_n)$ se la llama *función de frecuencia* de R en I respecto a una distribución esférica μ.

La función δ_R mide cómo de rápido convergen las frecuencias $\mu_n(R \cap I_n)$ a la densidad asintótica $\rho(R)$, en el caso de que exista. Así que estudiando la tasa de convergencia de la función δ_R podremos diferenciar entre conjuntos genéricos ó despreciables.

Sea $R \subseteq I$ y supongamos que R tiene densidad asintótica $\rho_\mu(R)$. Decimos que la densidad asintótica de R tiene tasa de convergencia

1) de grado n^k si $\lim_{n\to\infty} \frac{|\rho(R)-\delta_R(n)|}{cn^k} = 1$, para alguna constante c.

2) superpolinomial si $|\rho(R) - \delta_R(n)| = o(n^{-k})$ para cualquier número natural k.

3) exponencial si $\lim_{n\to\infty} \frac{|\rho(R)-\delta_R(n)|}{c^n} = 1$, para alguna constante $0 < c < 1$.

Por ejemplo, diremos que R es *genérico superpolinomial* (**exponencial**) si es genérico y su tasa de convergencia es superpolinomial (**exponencial**). Nos referiremos a conjuntos genéricos superpolinomiales como conjuntos *fuertemente genéricos*.

1.2.8. Clases de complejidad y Complejidad clásica

¿Cómo medir la complejidad los problemas? ¿Cuándo podemos decir que un problema es difícil? Al decir problema difícil nos referimos a aquellos cuyas soluciones son complicadas de obtener (los algoritmos pensados para obtener una solución tardan demasiado, pero ¿qué se considera tardar demasiado?). Para definir una clase de complejidad necesitamos especificar lo siguiente: un modelo de computación (en nuestro caso será una máquina de Turing), un modo de computación, unos recursos que debemos controlar (por ejemplo el tiempo) y cotas para cada recurso.

Consideremos una función $f : \mathbb{N} \to \mathbb{R}^+$. Definimos $\mathbf{TIME}(f)$ como la clase formada por todos los problemas de decisión para los que tenemos una función de tamaño s definida en un conjunto de instancias I y existe una máquina de Turing M que resuelve el problema con $T_M(x) = O(f(s(x)))$ para todo $x \in I$. Es decir, el número de pasos que da la máquina para todas las instancias x en I_n es $O(f(n))$.

Llamamos *Clase* $\boldsymbol{P}$ a la formada por todos los problemas de decisión que una Máquina de Turing determinista puede resolver en tiempo polinomial. Formalmente

$$\mathbf{P} = \bigcup_{k=1}^{\infty} \mathbf{TIME}\ (n^k)$$

Un ejemplo de problema en la clase **P** es el de calcular el máximo común divisor de dos números. En 2002 quedó demostrado que el problema de determinar si un número es primo o no está en la clase **P** [1].

La *Clase* ***NP*** está formada por los problemas de decisión cuya solución se puede verificar en tiempo polinomial por una Máquina de Turing determinista. Es decir, aquellos problemas para los que existe un algoritmo que en tiempo polinomial comprueba una solución positiva al problema. Por ejemplo, el problema de obtener una factorización no trivial de un número compuesto está en esta clase. Pues podemos comprobar si $n = ab$ es una factorización no trivial viendo que efectivamente a y b no son 1 y $n = ab$. Esto se puede formalizar de forma obvia con el concepto de máquina de Turing. Sin embargo no se sabe si este problema está en la clase **P**. De hecho no se sabe si **P**=**NP**, y este es uno de los problemas del milenio.

1.2.9. Reducciones y problemas completos

En esta subsección veremos qué significa que un problema sea más, menos o igual de difícil que otro. Primero definimos un *oráculo* como una máquina abstracta que se utiliza para estudiar problemas de decisión. Lo veremos como una caja negra capaz de resolver problemas computacionales, es decir, le damos una instancia de un problema, y nos devuelve una solución. No contabilizamos el tiempo que emplea en obtenerla, ni tampoco nos preocupamos de cómo obtiene la respuesta. Intuitivamente, si un problema D_1 se puede reducir a un problema D_2 en tiempo $f(n)$, significa que hay un oráculo para D_2 y un algoritmo que resuelve D_1 (máquina de Turing) que usa este oráculo para dar soluciones a D_2 para todas las instancias que sea necesario, y el tiempo de dicho algoritmo contabilizando las llamadas del oráculo como tiempo 0 está en **TIME**(f).

Sea S un conjunto de problemas de decisión. Como hemos mencionado antes, nos centraremos en tiempo polinomial, por lo que diremos que un problema C es *completo* en S si todos los problemas de S se reducen a C en tiempo polinomial y $C \in S$. Los problemas completos en **NP** se suelen denominar **NP**-completos. Por ejemplo, el Problema de la 3-satisfacibilidad en la sección 1.2.1 y el problema de decisión del circuito Hamiltoniano son problemas **NP**-completos. Este problema consiste en decidir si un grafo no dirigido tiene un circuito hamiltoniano, es decir, un camino que pase por todos los vértices una sola vez.

Capítulo 2

Complejidad media y genérica

En **Preliminares** hemos revisado el concepto de complejidad clásica, que llamaremos de peor caso. Veremos más adelante que esta clase de complejidad en algunos casos no es útil, al menos en criptografía, ya que en el peor de los casos un problema puede ser irresoluble y sin embargo en la mayoría de sus inputs puede requerir tiempo polinomial. Por lo que si ese problema forma parte de nuestro protocolo criptográfico, éste no será tan seguro como podíamos pensar. Justificamos así este capítulo, **Complejidad media y genérica**, ya que sabemos que hay problemas que son **NP**-completos, sin embargo, a la hora de resolverlos, no resultan difíciles en la mayoría de los casos.

2.1. Complejidad media

A partir de ahora trabajaremos con problemas computacionales cuyos conjuntos de instancias tienen una medida de probabilidad definida o un encadenamiento de distribuciones.

Un *problema computacional con distribución* es un par (D, μ) en el que D es un problema computacional con conjunto de instancias I y μ es una medida de probabilidad en I. De igual manera, un *problema computacional estratificado* es una terna $(D, s, \{\mu_n\})$ donde D es un problema computacional con conjunto de instancias I, s es una función de tamaño en I y para cada $n \in \mathbb{N}$, μ_n es una medida definida en la esfera I_n o en la bola B_n. En el segundo caso, μ_{n-1} es la restricción de μ_n a B_{n-1}.

Veamos una serie de nociones distintas que hacen referencia al comportamiento de una función

$f : I \to \mathbb{R}^+$ en media con respecto a una función de tamaño $s : I \to \mathbb{R}$ que se mantendrá todo el rato. En algunos casos tenemos una medida en I y en otros un encadenamiento de distribuciones esféricas o de volumen.

Definición 2.1.1 *Sea $f : I \to \mathbb{R}^+$ una función. Decimos que f es* polinomial en esferas *("expected polynomial on spheres") respecto a un encadenamiento de distribuciones $\{\mu_n\}$, si para cada $n \in \mathbb{N}$ y para algún polinomio $p(x)$,*

$$\int_{I_n} f(x)\mu_n(x) \leq p(n)$$

o equivalentemente si existe $k \geq 1$ tal que

$$\int_{I_n} f(w)\mu_n(w) = O(n^k) \tag{2.1}$$

Definición 2.1.2 *Sea $f : I \to \mathbb{R}^+$ una función. Decimos que f es* polinomial en $\mu-$media *si existe $\epsilon > 0$ tal que*

$$\int_I \frac{f(w)^\epsilon}{s(w)}\mu(w) < \infty$$

En el caso en que $\epsilon = 1$ decimos que f es lineal en $\mu-$media. *Así, equivalentemente decimos que f es polinomial en $\mu-$media si $f \leq p(l)$ para alguna función l lineal en media y para p un polinomio.*

Definición 2.1.3 *Sea $f : I \to \mathbb{R}^+$ una función. Decimos que f es* polinomial en media en esferas *(respecto a un encadenamiento de distribuciones $\{\mu_n\}$) si existe $\epsilon > 0$ tal que*

$$\int_{I_n} f(w)^\epsilon \mu_n(w) = O(n). \tag{2.2}$$

Esto es equivalente a que exista $k \geq 1$ tal que

$$\int_{I_n} f^{\frac{1}{k}}(w)\mu_n(w) = O(n).$$

Usando que $s(w) = n$ si $w \in I_n$, reescribimos la condición (2.2) como

$$\int_{I_n} \frac{f(w)^\epsilon}{s(w)}\mu_n(w) = O(1)$$

La siguiente definición, introducida en [9], hace referencia a las funciones en media, pero respecto a encadenamientos de distribuciones de volumen en las bolas B_n, a los que denotaremos por $\{\sigma_n\}$.

Definición 2.1.4 *Sea $\{\sigma_n\}$ un encadenamiento de distribuciones de volumen en las bolas $\{B_n\}$ de I y sea $f : I \to \mathbb{R}$ una función. Decimos que f es* polinomial en media en volumen *respecto a $\{\sigma_n\}$ si existe un $\epsilon > 0$ tal que*

$$\int_{B_n} f(x)^{\epsilon}\sigma_n(x) = O(n)$$

Las funciones polinomiales en esferas pertenecen a una clase de funciones muy pequeña que no es cerrada para el producto.

Ejemplo 2.1.5 *Sea $I = \{0,1\}^*$ el conjunto de palabras en el alfabeto $\{0,1\}$, $s(w) = |w|$ la longitud de la palabra w y para cada $n \in \mathbb{N}$ sea μ_n la distribución uniforme en I_n.*

Para cada $n \in \mathbb{N}$ fijamos un subconjunto S_n de I_n que contenga $2^n - 1$ elementos. Definimos ahora una función $f : I \to \mathbb{N}$ como :

$$f(w) = \begin{cases} n, & si\ w \in S_n; \\ 2^n, & si\ w \notin S_n. \end{cases}$$

Así, $\int_{I_n} f(w)\mu_n(w) = \int_{I_n} f(w)\frac{1}{|I_n|} = \frac{1}{2^n}\left(\int_{I_n\setminus S_n} f(w) + \int_{S_n} f(w)\right) = 2^{-n}\left(\sum_{I_n\setminus S_n} 2^n + \sum_{S_n} n\right) =$

$2^{-n}|I_n \setminus S_n|2^n + 2^{-n}|S_n|n = 1 + 2^{-n}n(2^n - 1) = 1 + n - 2^{-n} \sim n$

Por lo que f cumple la condición (2.1)*, sin embargo, su cuadrado f^2 no la cumple ya que*

$$\int_{I_n} f^2(w)\mu(w) = 2^{-n}\sum_{I_n\setminus S_n} 2^{2n} + \sum_{S_n} n^2 = 2^{-n}2^{2n} + 2^{-n}n^2(2^n - 1) = 2^n + n^2(2^n - 1) \sim 2^n$$

Este ejemplo justifica la Definición 2.1.3, que introduce una clase más amplia de funciones polinomiales en media.

A continuación presentamos dos resultados que establecen equivalencias o relaciones entre las definiciones anteriores. La siguiente proposición demuestra que toda función polinomial en esferas es polinomial en media en esferas.

Proposición 2.1.6 *Sea $\{\mu_n\}$ un encadenamiento de distribuciones esféricas. Si una función $f : I \to \mathbb{R}^+$ es polinomial en esferas (definición 2.1.1) relativa a $\{\mu_n\}$ entonces es polinomial en media en esferas (Definición 2.1.3).*

Demostración: Supongamos que para algún $k \geq 1$

$$\int_{I_n} f(w)\mu_n(w) \leq cn^k.$$

Tomamos $\epsilon = \frac{1}{k}$ y denotamos por $S_n = \{w \in I_n; f(w)^\epsilon \leq s(w)\}$ con $\bar{S_n} = I_n \setminus S_n$. Entonces

$$\begin{aligned}\int_{I_n} \frac{f(w)^\epsilon}{s(w)}\mu_n(w) &= \int_{S_n} \frac{f(w)^\epsilon}{s(w)}\mu_n(w) + \int_{\bar{S_n}} \frac{f(w)^\epsilon}{s(w)}\mu_n(w) \leq 1 + \int_{\bar{S_n}} (\frac{f(w)^\epsilon}{s(w)})^k \mu_n(w) \\ &\leq 1 + \int_{I_n} \left(\frac{f(w)^\epsilon}{s(w)}\right)^k \mu_n(w) \leq 1 + \int_{I_n} \frac{f(w)}{n^k}\mu_n(w) \leq 1 + c\end{aligned}$$

■

El siguiente resultado dice que en el caso en que un encadenamiento de distribuciones de volumen $\{\sigma_n\}$ proviene de la restricción de una medida σ entonces polinomial en $\sigma-$media y en media en volumen son equivalentes.

Proposición 2.1.7 *Sea σ una medida de distribución en I y $\{\sigma_n\}$ el encadenamiento de volumen inducido por σ. Entonces, una función $f : I \to \mathbb{R}^+$ es polinomial en σ-media si y sólo si es polinomial en media respecto a $\{\sigma_n\}$.*

Demostración: Supongamos que f es polinomial en σ-media, luego existe $\epsilon > 0$ tal que

$$\int_I \frac{f^\epsilon(w)}{s(w)}\sigma(w) < \infty$$

entonces

$$\begin{aligned}\int_{B_n} f^\epsilon(w)\sigma_n(w) &\leq \int_{B_n} \frac{n}{s(w)} f^\epsilon(w) \frac{\sigma(w)}{\sigma(B_n)} \\ &\leq \frac{n}{\sigma(B_{min})} \int_{B_n} \frac{1}{s(w)} f^\epsilon(w)\sigma(w) = O(n)\end{aligned}$$

donde B_{min} es la bola de radio mínimo para la que $\sigma(B_{min}) \neq 0$. Así, f es polinomial en media relativa a $\{\sigma_n\}$

Supongamos ahora que f es polinomial en media respecto a $\{\sigma_n\}$, así, para algún $\epsilon > 0$

$$\int_{B_n} f(x)^\epsilon \sigma_n(x) = O(n)$$

Tomamos $S = \{w \in I; f(w)^{\frac{\epsilon}{3}} \leq s(w)\}$, entonces

$$\begin{aligned}\int_I f(w)^{\frac{\epsilon}{3}} s^{-1}(w)\sigma(w) &= \int_S f(w)^{\frac{\epsilon}{3}} s^{-1}(w)\sigma(w) + \int_{\bar{S}} f(w)^{\frac{\epsilon}{3}} s^{-1}(w)\sigma(w) \\ &\leq \int_I \mu(w) + \int_{\bar{S}} \frac{f(w)^\epsilon}{f(w)^{\frac{2\epsilon}{3}} s(w)}\sigma(w)\end{aligned}$$

Dado w en $\bar{S}$ tenemos $f(w)^{\frac{\epsilon}{3}} > s(w)$, por tanto $f(w)^{\frac{2\epsilon}{3}} > s(w)^2$. Por lo que continuando la expresión anterior

$$\begin{aligned}
\int_I \sigma(w) + \int_{\bar{S}} \frac{f(w)^\epsilon}{f(w)^{\frac{2\epsilon}{3}} s(w)} \sigma(w) &\leq 1 + \int_{\bar{S}} \frac{f(w)^\epsilon}{s(w)^3} \sigma(w) \leq 1 + \int_I \frac{f(w)^\epsilon}{s(w)^3} \sigma(w) \\
&\leq 1 + \sum_n \int_{I_n} \frac{f(w)^\epsilon}{n^3} \sigma(w) \leq 1 + \sum_n \frac{1}{n^3} \int_{B_n} f(w)^\epsilon \sigma(w) \\
&\leq 1 + \sum_n \frac{1}{n^3} \int_{B_n} f(w)^\epsilon \sigma_n(w) \\
&= 1 + \sum_n \frac{O(n)}{n^3} < \infty
\end{aligned}$$

Así la integral converge. ■

Puede resultar complicado comprobar si una función se ajusta a alguna de las definiciones anteriores, para ello introducimos los siguientes conceptos.

Decimos que la función f es una *función de rareza* ("rarity function") si la siguiente integral converge

$$\int_I f(w)\mu(w)$$

Esta función se corresponde con el valor esperado de f.

Proposición 2.1.8 *Una función $f : I \to \mathbb{R}^+$ es polinomial en media si y sólo si existe una función de rareza $h : I \to \mathbb{R}^+$ y un polinomio $p(x)$ tal que para cada $w \in I$*

$$f(w) \leq p(s(w), h(w))$$

Demostración: Ver [6] ■

Con el siguiente resultado obtenemos otra condición suficiente para que la función f sea polinomial en media.

Proposición 2.1.9 *Si para algún polinomio $p(x)$,*

$$\int_I \frac{f(w)}{p(s(w))} \mu(w) < \infty$$

entonces la función f es polinomial en $\mu-$media.

Demostración: En este caso tenemos

$$\int_I \frac{f(w)}{s(w)p(s(w))}\mu(w) \leq \int_I \frac{f(w)}{p(s(w))}\mu(w) < \infty$$

así que la función $l(w) = \frac{f(w)}{p(s(w))}$ es lineal en μ-media. Luego $f(w) = l(w)p(s(w))$ es polinomial en media. ■

Uno de los problemas más conocidos en teoría de la complejidad es el problema de **P** versus **NP**, que al menos por el momento sigue sin tener solución. Imaginemos que la hipótesis **P** = **NP** es falsa, entonces podríamos pensar que algunos de los problemas de la clase **NP** serían muy difíciles de resolver, pero ¿es esto cierto?

Acabamos de ver una serie de definiciones y resultados que nos permiten estudiar el comportamiento de los algoritmos desde un punto de vista diferente al del peor caso. El siguiente teorema, cuya demostración se puede encontrar en [8] indica que si bien un problema puede ser **NP**-completo no tiene por qué ser difícil de resolver en media.

Teorema 2.1.10 *Hay problemas **NP**-completos que son polinomiales en media respecto a algunas distribuciones naturales.*

Aunque acabemos de ver que el caso medio es más útil, o al menos más adecuado, cuando estudiamos el comportamiento de un algoritmo que el peor caso, el siguiente ejemplo muestra que no siempre es la herramienta acertada si queremos describir el comportamiento de una función en la mayoría de los inputs de un problema.

Ejemplo 2.1.11 *Sea $I = \{0,1\}^*$, el conjunto de todas las palabras en el alfabeto $\{0,1\}$. Definimos el tamaño de una palabra $w \in I$ como su longitud y tomamos μ como una distribución en I que restringida a cada esfera I_n es la distribución uniforme.*

Para cada n en $\mathbb{N}$ elegimos un subconjunto S_n de I_n de medida $\frac{2^n-1}{2^n}$, o sea, S_n tiene $2^n - 1$ elementos. Definimos la función $l : I \to \mathbb{N}$ como

$$l(w) = \begin{cases} 0 & si \quad w \in S_n \\ 2^n & si \quad w \notin S_n \end{cases}$$

Entonces

$$\begin{aligned} \int_I l(w)|w|^{-1}\mu(w) &= \sum_n \frac{1}{n}\int_{I_n} l(w)\mu(w) = \sum_n \frac{1}{n}\int_{I_n} l(w)\mu_n(w) \\ &= \sum_n \frac{1}{n}\int_{I_n \setminus S_n} 2^n \frac{\mu_n(w)}{|I_n \setminus S_n|} = \sum_n \frac{1}{n}2^n < \infty \end{aligned}$$

Por lo que la función l es lineal en media. Definimos ahora una función $f : I \to \mathbb{N}$ por sus valores en cada esfera I_n como

$$f(w) = \begin{cases} 1, & si \quad w \in S_n \\ 2^{2^n}, & si \quad w \notin S_n. \end{cases}$$

Vemos que $f(w) = 2^{l(w)}$, luego f es exponencial en media, sin embargo $f(w) = 1$ en muchas palabras $w \in I$.

2.2. Complejidad genérica

Aunque la complejidad media nos ofrece un punto de vista más práctico que la complejidad por peor caso, en general sigue sin ser del todo útil, sobretodo cuando estudiamos el comportamiento de los algoritmos en la mayoría de sus inputs. Por ejemplo, a veces, en criptografía, el caso medio no nos es de utilidad, ya que un algoritmo (que representa la seguridad de un criptosistema) puede tener tiempo exponencial en media pero tener tiempo lineal en la mayoría de sus inputs, lo que equivale a baja seguridad. Esta es la motivación de esta sección en la que estudiaremos el caso genérico que será una herramienta más útil.

Sea I un conjunto, s una función de tamaño en I y $\mu = \{\mu_n\}$ un encadenamiento esférico de distribuciones en I. Sea D un problema computacional y A un algoritmo de decisión parcial para D con conjunto de parada H_A y función de tiempo $T_A : I \to \mathbb{N}$. Si A no se para en x una instancia de I, entonces $T_A(x) = \infty$.

Sea D un problema computacional. Un algoritmo de decisión parcial A para D *resuelve genéricamente* el problema D si el conjunto donde A resuelve correctamente el problema D es genérico en el conjunto de instancias I respecto al encadenamiento de distribuciones $\mu = \{\mu_n\}$. En este caso diremos que D es genéricamente decidible.

Recordamos que el conjunto de parada H_A para un algoritmo A es el conjunto de inputs o instancias en las que A devuelve una respuesta. Y definimos por el *tiempo esperado* de un algoritmo A como

$$\int_I T_A(w)\mu(w)$$

Sea D un problema computacional estratificado y A un algoritmo de decisión parcial para D. Decimos que una función $f(n)$ es una *cota superior genérica* para A si el conjunto

$$H_{A,f} = \{w \in H_A; T_A(w) \leq f(s(w))\}$$

es genérico en I respecto a la densidad asintótica ρ_μ. De igual manera podemos decir que el algoritmo A tiene una*cota superior fuertemente genérica* (**exponencialmente genérica**, ...) si el conjunto $H_{A,f}$ es fuertemente genérico (**exponencialmente genérico**,...).

Pasamos ahora a definir clases de complejidad genérica en los problemas algorítmicos. Decimos que un problema de decisión D es

- *decidible genéricamente en tiempo polinomial*, **GPtime**, si existe un algoritmo de decisión A para D con una cota superior polinomial genérica.
- *decidible fuertemente genérico en tiempo polinomial*, **SGPtime**, si existe un algoritmo de decisión A para D con una cota superior polinomial fuertemente genérica.

Denotamos por **GenP** y $\mathbf{Gen_{str}P}$ a las clases de problemas decidibles en tiempo polinomial genérico y fuertemente genérico.

En el caso de complejidad genérica buscamos un algoritmo que resuelva el problema genéricamente mientras que para el caso medio requerimos algoritmos que resuelvan el problema siempre. Lo que significa que en caso genérico podemos tratar problemas decidibles y problemas no decidibles no así en el caso medio. Otra diferencia es que el caso genérico se centra en el comportamiento del algoritmo en la mayoría de los inputs (en los subconjuntos genéricos de I) mientras que el caso medio se enfoca en el tiempo esperado del algoritmo.

A la hora de formalizar ambos conceptos la complejidad genérica resulta más fácil de entender, ya que estudia el tiempo que tarda el algoritmo en la mayoría de las instancias, mientras que en el caso medio se hace referencia a la fracción de inputs difíciles y cuánto tiempo se tarda en ellos, lo que a la hora de utilizar una medida, dificulta la tarea.

La siguiente proposición muestra una relación entre las versiones de complejidad media y complejidad genérica.

Proposición 2.2.1 *Si una función $f : I \to \mathbb{R}^+$ es polinomial en media en esferas entonces f es genéricamente polinomial relativa a la densidad asintótica ρ_μ.*

Demostración: Por hipótesis, existirá una constante c y $k \geq 1$ tales que

$$\int_{I_n} f^{\frac{1}{k}}(w)\mu_n(w) \leq cn. \tag{2.3}$$

Dado un polinomio q ponemos $A_q = \{x \in I_n; f^{\frac{1}{k}}(x) > q(n)cn\}$ Veamos que para cualquier polinomio $q \neq 0$ se tiene

$$\mu_n(A_q) \leq \frac{1}{q(n)}$$

Por reducción al absurdo supongamos que existe un polinomio $q(n)$ tal que $\mu_n(A_q) > \frac{1}{q(n)}$ Usando (2.3) tenemos

$$\begin{aligned} cn \geq \int_{I_n} f^{\frac{1}{k}}(w)\mu_n(w) &= \int_{I_n \setminus A_q} f^{\frac{1}{k}}(w)\mu_n(w) + \int_{A_q} f^{\frac{1}{k}}(w)\mu_n(w) \\ &> \int_{I_n \setminus A_q} f^{\frac{1}{k}}(w)\mu_n(w) + \int_{A_q} q(n)cn\mu_n(w) \\ &= \int_{I_n \setminus A_q} f^{\frac{1}{k}}(w)\mu_n(w) + q(n)cn\mu(A_q) \\ &> \int_{I_n \setminus A_q} f^{\frac{1}{k}}(w)\mu_n(w) + cn \geq cn, \end{aligned}$$

lo que supone una contradicción.

Sea $S(f,q,k) = \{x \in I; f(x) \geq (cq(s(x))s(x))^k\}$ el conjunto de instancias en I tales que $f(x)$ no está acotada por $(cq(s(x))s(x))^k$. Entonces

$$\mu_n(I_n \cap S(f,q,k)) = \mu_n(A_q) \leq \frac{1}{q(n)}$$

Así, la densidad asintótica de $S(f,q,k)$ existe y es igual a 0. Por lo que f está genéricamente acotada por el polinomio $(cq(n)n)^k$ para todo polinomio q no constante. ■

Esta proposición da una clase de funciones que siendo polinómicas en el caso medio, también lo son genéricamente.

Corolario 2.2.2 *Sea A un algoritmo de decisión para el problema D. Si el tiempo esperado de A está acotado por un polinomio entonces A decide D en* ***GPtime****.*

Capítulo 3

Complejidad algorítmica en grupos

En este capítulo vamos a estudiar algunos problemas de computación en grupos finitamente generados desde el punto de vista de su complejidad computacional. Durante todo el capítulo denotamos por G a un grupo finitamente generado y por X a un conjunto generador finito de G. Cuando tengamos una palabra en X, o sea, un elemento del grupo libre $F(X)$, diremos que esa palabra cumple una propiedad en G si su imagen por el homomorfismo que nos da la propiedad universal cumple la propiedad en G. De igual manera, cuando hablemos del subgrupo de G generado por palabras en X nos estaremos refiriendo al subgrupo generado por la imagen de esas palabras por el homomorfismo de la propiedad universal. Dado un subgrupo finitamente generado H de G, una posible descripción δ de este subgrupo sería una tupla de palabras $(u_1, \ldots, u_n)$ en el alfabeto X que represente un conjunto generador de H. Así describimos los subgrupos de G mediante una lista de palabras. Denotamos por Δ al conjunto de descripciones de todos los subgrupos finitamente generados de G, es decir, Δ es el conjunto de tuplas finitas en $F(X)$. Para fijar una noción de tamaño necesitamos fijar una función $s : \Delta \to \mathbb{N}$ tal que la bola de radio n, $B_n = \{\delta \in \Delta; s(\delta) \leq n\}$ sea finita. De esta manera obtendremos una estratificación de Δ, que nos permite ver el conjunto como la unión finita de las bolas B_n,

$$\Delta = \bigcup_{n=1}^{\infty} B_n, \tag{3.1}$$

o la unión disjunta de las esferas

$$\Delta = \bigcup_{n=1}^{\infty} \Delta_n, \Delta_n = B_n \setminus B_{n-1}$$

El tamaño de una tupla $(u_1, \ldots, u_k)$ lo podemos definir como la suma de las longitudes de los generadores, $s(u_1, \ldots, u_k) = |u_1| + \ldots + |u_k|$, y otra forma sería $s(u_1, \ldots, u_k) = \max\{|u_1|, \ldots, |u_k|\}$.

Donde siempre entendemos $|g|$ como la longitud $|w|_X$ de la palabra más corta en X que representa a g. Nuestro procedimiento funciona para las dos definiciones, por lo que no es necesario especificar cual utilizamos.

Para poder estudiar las propiedades asintóticas de los subgrupos, lo que haremos será estudiarlas a través de sus descripciones, por lo que trabajaremos en el conjunto Δ utilizando conceptos que hemos definido en las secciones anteriores. También necesitaremos fijar medidas de probabilidad. Al tener bolas con un número finito de elementos, podemos asignar una medida de probabilidad σ_n a cada bola (por ejemplo, σ_n puede ser la distribución uniforme en la bola B_n). Teniendo así un encadenamiento de distribuciones $\{\sigma_n\}$ y pudiendo así trabajar con la función de densidad asintótica (ver subsección 1.2.6). A lo largo de este capítulo, cuando hablemos de la densidad asintótica de un subconjunto R de Δ, por comodidad, utilizaremos la notación $\rho(R)$.

Sea k un entero positivo fijo. Denotamos por P una propiedad en el conjunto de descripciones que corresponden a subgrupos $k-$generados en G. Por $P(G)$ denotamos el conjunto de descripciones en Δ que satisfacen P en G y por $\rho(P(G))$ a la densidad asintótica de $P(G)$.

Decimos que $P(G)$ es :

1) *asintóticamente visible* en G si $\rho(P(G)) > 0$
2) *genérica* en G si $\rho(P(G)) = 1$
3) *fuertemente genérica* en G si $\rho(P(G)) = 1$ y la tasa de convergencia de $\rho(P(G))$ es superpolinomial.
4) *exponencialmente genérica* en G si $\rho(P(G)) = 1$ y la tasa de convergencia de $\rho(P(G))$ es exponencial.

3.1. Propiedad de la base libre en grupos

Decimos que una tupla $(u_1, \ldots, u_k) \in F(X)^k$ cumple la *propiedad de la base libre* en G, que denotamos por FB, si genera un subgrupo libre de rango k en G.

En [12], se demuestra que la propiedad FB es exponencialmente genérica en $F(X)$ para todo $k \geq 1$ con respecto a la base X. Podemos trazar un paralelismo con el concepto de linealmente independientes, en un espacio vectorial de dimensión n, si elegimos al azar n vectores, la probabilidad de que sean linealmente independientes es 1, sin embargo, si elegimos $n+1$ la probabilidad

es 0. La propiedad de la base libre es aún más genérica que lo de ser linealmente independientes pues, si en vez de en espacios vectoriales pensamos en grupos libres no abelianos, la probabilidad de que al elegir de manera aleatoria una $k-$tupla de elementos ésta genera un grupo libre de rango k es 1 y esto para todo k.

Diremos que un grupo G cumple la propiedad de la base libre de forma genérica (**fuertemente genérica**, **exponencialmente genérica**) si FB es una propiedad genérica (**fuertemente genérica**, **exponencialmente genérica**) en G para cada $k \geq 1$ y para cada conjunto generador finito de G. Denotamos por FB_{gen} , FB_{st} , FB_{exp} las clases de los grupos finitamente generados que cumplen la propiedad de la base libre de forma genérica, fuertemente genérica y exponencialmente genérica respectivamente. En el siguiente resultado vemos como se relacionan estas propiedades al pasar a cocientes.

Teorema 3.1.1 *Sea G un grupo finitamente generado y N un subgrupo normal de G. Si el grupo cociente G/N está en las clases FB_{gen} , FB_{st} o FB_{exp} entonces G está en la misma clase.*

Demostración: Sea $\varphi : G \to G/N$ homomorfismo canónico y fijemos un conjunto generador X de G. Describimos los subgrupos finitamente generados de G y de G/N como listas de palabras en X. Sea $A_G = \{(a_1, \ldots, a_k) \in F(X)^k; a_1, \ldots, a_k$ representa un grupo libre de rango k en $G\}$ y definimos $A_{G/N}$ de forma análoga. Por hipótesis $A_{G/N}$ es genérico con lo que basta demostrar que A_G contiene a $A_{G/N}$. Para ver esto denotemos por $f : F(X) \to G$ y $g : F(\varphi(X)) \to G/N$ los homomorfismos que extienden las inclusiones de X en G y de $\varphi(X)$ en G/N respectivamente, y por $\psi : F(X) \to F(\varphi(X))$ el homomorfismo que extiende la restricción de φ a X. Entonces

$$A_G = \{(a_1, \ldots, a_k) \in F(X)^k; < f(a_1), \ldots, f(a_k) > \text{ es libre de rango } k\}$$

y

$$A_{G/N} = \{(a_1, \ldots, a_k) \in F(X)^k :< g\psi(a_1), \ldots, g\psi(a_k) > \text{ es libre de rango } k\}.$$

Por tanto basta demostrar que si $< g\psi(a_1), \ldots, g\psi(a_k) >$ es libre de rango k entonces $< f(a_1), \ldots, f(a_k) >$ es libre de rango k. Para esto vemos primero que el siguiente diagrama es conmutativo

$$\begin{array}{ccccc} X & \hookrightarrow & F(X) & \xrightarrow{f} & G \\ \downarrow{\scriptstyle \varphi} & & \downarrow{\scriptstyle \psi} & & \downarrow{\scriptstyle \varphi} \\ \varphi(X) & \hookrightarrow & F(\varphi(X)) & \xrightarrow{g} & H \end{array}$$

En efecto, el cuadrado de la izquierda es conmutativo por la definición de ψ y si $x \in X$ entonces por las definiciones de f y g tenemos $\varphi f(x) = \varphi(x) = g\varphi(x) = g\psi(x)$. Entonces por la propiedad universal del grupo libre $\varphi \circ f$ y $g \circ \psi$ no solo coinciden en X sino en todo $F(X)$.

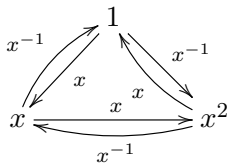
Por tanto $< g\psi(a_1), \ldots, g\psi(a_k) >=< \varphi f(a_1), \ldots, \varphi f(a_k) >$, luego si $(a_1, \ldots, a_k) \in A_{G/N}$ entonces $< \varphi f(a_1), \ldots, \varphi f(a_k) >$ es libre de rango k. Sea $Y = \{1, \ldots, k\}$ y sea $h : F(Y) \to G$ el único homomorfismo tal que $h(i) = f(a_i)$. Si $(a_1, \ldots, a_k) \in A_{G/N}$ entonces $\varphi \circ h : F(Y) \to < \varphi f(a_1), \ldots, \varphi f(a_k) >$ es un isomorfismo con lo que $h : F(Y) \to < f(a_1), \ldots, f(a_k) >$ es inyectiva y como también es suprayectiva, $< f(a_1), \ldots, f(a_k) >$ es libre de rango k. Por tanto $(a_1, \ldots, a_k) \in A_G$. Esto demuestra $A_{G/N} \subseteq A_G$ como queríamos. ■

3.2. Subgrupos cuasi-isométricamente embebidos

Introducimos brevemente una propiedad de los subgrupos de G que es relevante en criptografía. Para ello necesitamos introducir el grafo de Cayley de un grupo G con respecto a un conjunto generador X.

El *grafo de Cayley* $\Gamma(G, X)$ es un grafo dirigido con etiquetas en X y cuyo conjunto de vértices es G. Dos vértices g, h en G están conectados por una arista con etiqueta x con origen en g y final en h si y sólo si $gx = h$ en G. Por comodidad asumiremos que el conjunto X es cerrado bajo inversión, es decir, para cada x en X se cumple x^{-1} en X, lo que implica que las flechas van en parejas. Obsérvese que $\Gamma(G, X)$ es conexo porque G está generado por X. Podemos introducir una métrica d_X en G tomando $d_X(g, h)$ como la longitud de la palabra más corta en $X \cup X^{-1}$ que representa al elemento $g^{-1}h$ en G. De esta manera $d_X(g, h)$ es igual a la longitud del camino más corto de g a h en el grafo de Cayley $\Gamma(G, X)$. Veamos un ejemplo de cómo es el grafo de Cayley.

Ejemplo 3.2.1 *Sea* $G =< x >= \{1, x, x^2\}$ *grupo cíclico de orden* 3. *Tendríamos el siguiente grafo:*



Diremos que $l_X(g)$ es la longitud de la palabra más corta en X que representa a g, a la que nos referiremos como *longitud geodésica* del elemento g así $l_X(g) = d_X(1, g)$. Al introducir esta métrica obtenemos un espacio métrico (G, d_X). Sea H un subgrupo de G generado por el conjunto finito Y. Entonces en H tenemos dos métricas: d_Y como la hemos descrito en el párrafo anterior, y d_X la métrica inducida en H por el espacio métrico (G, d_X). La definición de subgrupo cuasi-isométricamente embebido nos permitirá comparar estas dos métricas, pero para ello primero recordamos el concepto de embebimiento cuasi-isométrico. Decimos que una

aplicación $f : M_1 \to M_2$ con (M_1, d_1) y (M_2, d_2) espacios métricos, es un *embebimiento cuasi-isométrico* si existen constantes $\lambda > 1$, $c > 0$ tales que para todos los elementos x, y en M_1 se cumple

$$\frac{1}{\lambda} d_1(x, y) - c \leq d_2(f(x), f(y)) \leq \lambda d_1(x, y) + c \tag{3.2}$$

Decimos que un subgrupo H con Y un conjunto finito generador de H está *embebido cuasi-isométricamente respecto a X e Y* en G si la inclusión $i : H \hookrightarrow G$ es un embebimiento cuasi-isométrico de espacios métricos $i : (H, d_Y) \hookrightarrow (G, d_X)$. Obsérvese que la desigualdad de la derecha en (3.2) siempre se cumple ya que para todos f, h en H tenemos

$$d_X(i(f), i(h)) \leq \max_{y \in Y} \{l_X(y)\} d_Y(f, h).$$

Sea G un grupo con X un conjunto generador finito de G y H un subgrupo de G generado por el conjunto Y. Entonces H está cuasi-isométricamente embebido en G con respecto a X e Y si existen constantes $\lambda > 1$, $c > 0$ tales que para todos los elementos f, h en H se cumple

$$d_Y(f, h) - c \leq c + \lambda d_X(f, h). \tag{3.3}$$

Decimos que una tupla $(u_1, \ldots, u_k) \in F(X)^k$ cumple la *propiedad del embebimiento cuasi-isométrico*, QI, en G si el subgrupo que genera en G está cuasi-isométricamente embebido en G con respecto a X y $\{u_1, \ldots, u_k\}$. Denotamos por QI_{gen}, QI_{st} y por QI_{exp} a las clases de grupos finitamente generados que cumplen la propiedad el embebimiento cuasi-isométrico de forma genérica, fuertemente genérica y exponencialmente genértica respectivamente.

Teorema 3.2.2 *Sean G un grupo finitamente generado y N un subgrupo normal de G. Si G/N pertenece a $FB_* \cap QI_*$ entonces G está en $FB_* \cap QI_*$, con $* \in \{$ gen, st, exp $\}$*

Demostración: Sea G un grupo con X un conjunto generador finito de G, N un subgrupo normal de G tal que G/N está en $FB_* \cap QI_*$. Sea $\varphi : G \to G/N$ el epimorfismo canónico. Por Teorema 3.1.1 sabemos que $G \in FB_*$, falta demostrar que G está en QI_*.

Sea H un subgrupo $k-$generado de G y sea $Y = (u_1, \ldots, u_k) \in F(X)^k$ representando a un conjunto generador de H. Supongamos que $Y \in FB_*(G/N) \cap QI_*(G/N)$, es decir, que $\varphi(Y)$, la imagen de Y por φ en G/N, genera un grupo libre cuasi-isométricamente embebido en G/N. Observamos que para cada g en G se tiene $l_X(g) \geq l_{\varphi(X)}(\varphi(g))$ dónde $l_{\varphi(X)}$ es la función de longitud en G/N relativa al conjunto $\varphi(X)$ de generadores. Como el subgrupo $\varphi(H)$ está cuasi-isométricamente embebido en G/N, el espacio métrico $(\varphi(H), d_{\varphi(Y)})$ está cuasi-isométricamente

embebido en $(\varphi(G), d_{\varphi(X)})$. Por otro lado, φ transforma H en $\varphi(H)$ isomórficamente de manera que para cada g, h en H se tiene $d_Y(g,h) = d_{\varphi(Y)}(\varphi(g), \varphi(h))$. Para cada g, h en H tenemos

$$\frac{1}{\lambda} d_Y(g,h) - c = \frac{1}{\lambda} d_{\varphi(Y)}(\varphi(g), \varphi(h)) - c \leq d_{\varphi(X)}(\varphi(g), \varphi(h)) \leq d_X(g,h)$$

con λ y c constantes del embebimiento cuasi-isométrico de $\varphi(H)$ en G/N. Por lo que, H está embebido cuasi-isométricamente en G. ■

3.3. Problemas computacionales relacionados con teoría de grupos

En esta subsección veremos una serie de problemas computacionales de Teoría de Grupos, a los que haremos referencia también a lo largo del capítulo siguiente. La mayoría de los problemas que definiremos vienen acompañados de su versión de búsqueda, ya que en criptografía tienen un papel más importante. El primero que introducimos es uno de los más conocidos, en todos ellos tomamos un grupo G:

Problema de la palabra: Dada una presentación $< X|S >$ de G y w una palabra en X, averiguar si $w = 1$ en G.

Problema de búsqueda de la palabra: Dada una presentación $< X|S >$ de G y una palabra w en X que representa al 1, encontrar una expresión de w como producto de conjugados en $F(X)$ de los elementos de X y sus inversos.

El problema del conjugado (CP): Dados dos elementos u, v en G decidir si existe x en G tal que $u^x = v$.

Problema de búsqueda del conjugado (CSP): Dados $u, v \in G$ tales que $u^x = v$ tiene solución en G, encontrar una solución.

El problema simultáneo del conjugado (SCP): Dados u_i, v_i con $i = 1, \ldots, n$ en G decidir si existe x en G solución al sistema $u_i^x = v_i$ para todo $i = 1, \ldots, n$.

Problema de búsqueda del conjugado simultáneo (SCSP): Dados $u_i, v_i \in G$ tales que el sistema $u_i^x = v_i$, $i = 1, \ldots, n$ tiene solución en G, encontrar una solución.

Problema de búsqueda del conjugado simultáneo relativo a un subgrupo (SCSP*): Dados $u_i, v_i \in G$ y un subgrupo $A = (a_1, \ldots, a_k)$ de G con $a_1, \ldots, a_k$ palabras de $F(X)$, tales

que el sistema $u_i^x = v_i$, $i = 1, \ldots, m$ tiene solución en A, encontrar una solución en A.

El problema de búsqueda de la raíz (RSP): Dada una palabra w en $F(X)$, encontrar la palabra más corta u en $F(X)$ tal que $w = u^n$ para n algún entero positivo.

Lema 3.3.1 *El problema de la palabra se puede resolver en tiempo lineal en grupos libres.*

Demostración: Sea w una palabra en X. Eliminando apariciones consecutivas de xx^{-1} ó $x^{-1}x$ en w obtenemos una palabra reducida de w en $[w]$. Esto requiere un tiempo $O(n)$. Si $w_1 = 1$ entonces w representa 1 en $F(X)$ y en caso contrario no representa al 1. ■

El siguiente resultado establece una relación entre algunos de los problemas anteriores en grupos libres.

Teorema 3.3.2 *Los problemas SCP y SCSP son reducibles en tiempo lineal a CP, CSP y RSP en grupos libres. En particular, son decidibles en tiempo lineal.*

Demostración: Damos una idea del algoritmo que resuelve SCP y SCSP usando oráculos para CP,CSP y RSP. Es decir, partimos de un sistema finito de ecuaciones conjugadas

$$\begin{cases} u_1^x &= v_1 \\ &\vdots \\ u_n^x &= v_n \end{cases}$$

y el algoritmo decide si el sistema tiene solución en $F(X)$ o no y si la tiene la encuentra. Usando el oráculo para CP podemos comprobar si hay alguna ecuación en el sistema que no tenga solución en $F(X)$, si la hay, habríamos acabado. Suponiendo que todas las ecuaciones tienen solución, utilizamos el oráculo para CSP en $F(X)$ y encontramos una solución particular d_i para cada $u_i^x = v_i$. En este caso el conjunto de soluciones de $u_i^x = v_i$ es $C(u_i)d_i$. Como en grupos libres el centralizador de un elemento no trivial es un grupo cíclico, para encontrar el generador de $C(u_i)$ aplicamos el algoritmo de RSP.

Consideramos ahora las dos primeras ecuaciones del sistema

$$u_1^x = v_1, \qquad u_2^x = v_2 \tag{3.4}$$

Así, (3.4) tendrá solución en $F(X)$ si y sólo si $V = C(u_1)d_1 \cap C(u_2)d_2 = (C(u_1) \cap C(u_2))d$ para algún $d \in F(X)$, es no vacío. Dividimos nuestro problema en dos casos:

Si $[u_1, u_2] = 1$ entonces V será no trivial si y sólo si las clases laterales coinciden. Esto lo podemos comprobar en tiempo lineal ya que el problema de la palabra se puede resolver en tiempo lineal en $F(X)$ (Lema 3.3.1). Luego en tiempo lineal comprobamos si (3.4) tiene solución o no. Si no la tiene hemos acabado, y si la tiene entonces podemos eliminar la primera ecuación, ya que es equivalente a la segunda, reduciendo así el sistema original y concluyendo por inducción.

Si $[u_1, u_2] \neq 1$ entonces $C(u_1) \cap C(u_2) = 1$ por lo que o bien $V = \emptyset$ ó $V = \{d\}$. Si se da la primera situación, $V = \emptyset$, entonces el sistema no tiene solución. Si por el contrario $V = \{d\}$ entonces d es la única solución posible, por lo que comprobando si cumple el resto de ecuaciones habríamos acabado. El problema es comprobar en tiempo lineal si V es un conjunto vacío o no, es decir, si la siguiente ecuación tiene solución o no

$$u_1^m d_1 = u_2^k d_2 \tag{3.5}$$

para m, k enteros. Vamos reescribiendo (3.5) de la siguiente manera:

$$u_1^m d_1 = u_2^k d_2 = d_2 (u_2^{d_2})^k$$

por lo que

$$u_1^m d_1 (u_2^{d_2})^{-k} = d_2 \tag{3.6}$$

Si w_1 y w_2 son palabras cíclicamente reducidas que obtenemos a partir de u_1 y u_2 respectivamente, entonces (3.6) es equivalente a

$$w_2^{-k} c w_1^m = b \tag{3.7}$$

Si w_2^{-1} ó cw_1 no fueran cíclicamente reducidas puedo ir conjugando y así arreglar la parte problemática hasta llegar a

$$w_2^{-s} c w_1^l = b' \tag{3.8}$$

Distinguimos dos situaciones: en la primera w_1 y w_2 conmutan. Al estar trabajando en un grupo libre esto significa que están en el mismo grupo cíclico, por lo que son potencias de un mismo generador. Así la ecuación se convierte en una ecuación de potencias y la podemos resolver aplicando el oráculo para RSP. Si por el contrario w_1 y w_2 no conmutan, cancelamos lo máximo posible en (3.8) y llegamos a

$$w_2^{-r} c' w_1^t = \bar{b} \tag{3.9}$$

dónde no hay cancelaciones y que podemos resolver. ■

Veamos ahora algunos problemas que implican descripciones de subgrupos:

El problema de la pertenencia (MP): Sea $A =< a_1, \ldots, a_m >$ un subgrupo de G descrito por unas palabras $a_1, \ldots, a_m$ en $F(X)$. Dada una palabra w en $F(X)$ decidir si pertenece a A.

El problema de búsqueda de la pertenencia (MSP): Sea $A =< a_1, \ldots, a_m >$ subgrupo de G descrito por palabras $a_1, \ldots, a_m$ en $F(X)$, dado w en $F(X)$ que pertenece a A encontrar una representación de w como producto de los generadores $a_1, \ldots, a_m$ y sus inversos.

El problema uniforme de la pertenencia (UMP): Dados $w, a_1, \ldots, a_m$ palabras en $F(X)$ decidir si w pertenece al subgrupo generado por $a_1, \ldots, a_m$.

El problema uniforme de búsqueda de la pertenencia (UMSP): Dados $w, a_1, \ldots, a_m$ en $F(X)$ tales que $w \in A =< a_1, \ldots, a_m >$ encontrar una representación de w como producto de los generadores $a_1, \ldots, a_m$ y sus inversos.

Los dos últimos problemas se llaman así porque una solución positiva para MSP implica una solución positiva para MP en todos los grupos.

Introducimos ahora una serie de problemas relacionados con el cálculo de la longitud geodésica de un elemento en G:

Cálculo de la longitud geodésica en un grupo (GLP): Dado un elemento w de G como producto de generadores de G, calcular $l_X(w)$.

Cálculo de la longitud geodésica en un subgrupo (GLSP): Sea A un subgrupo de G generado por el conjunto $Y = \{a_1, \ldots, a_k\}$ con $a_i \in F(X)$. Dado un elemento w de A como producto de generadores de A, calcular $l_Y(w)$.

Cálculo de la longitud geodésica en un subgrupo (GLSP*): Sea A un subgrupo de G generado por el conjunto $Y = \{a_1, \ldots, a_k\}$ con $a_i \in F(X)$. Dado w en A como una palabra en $F(X)$, calcular $l_Y(w)$.

Introducimos aquí un lema obvio que establece una relación entre los dos problemas anteriores.

Lema 3.3.3 *Sea G un grupo finitamente generado y A un subgrupo de G finitamente generado. Entonces:*

1) *GLSP se reduce en tiempo lineal a GLSP*.*

2) *GLSP* se reduce en tiempo lineal a GLSP y a MSP en A.*

Cálculo de una aproximación lineal de la longitud geodésica en un grupo (AGL): Dada una palabra $w \in F(X)$, calcular una aproximación lineal de la longitud geodésica de

w. Concretamente, encontrar un algoritmo que para $w \in F(X)$ devuelva w' en $F(X)$ tal que $\lambda l_X(w) + c \geq l_X(w')$ dónde λ y c son independientes de w.

Cálculo de una aproximación lineal de la longitud geodésica en un subgrupo (AGLS): Sea A un subgrupo de G generado por un conjunto finito $Y = \{a_1, \ldots, a_k\}$ con $a_i \in F(X)$. Dado un elemento w de A como una palabra en $F(X)$ calcular una aproximación lineal de la longitud geodésica $l_Y(w)$.

La relación entre los dos problemas anteriores no queda clara, excepto en subgrupos cuasi-isométricamente embebidos, para los que tenemos el siguiente resultado.

Teorema 3.3.4 *Sean G un grupo finitamente generado, con X un conjunto generador de G, H un subgrupo de G finitamente generado con Y un conjunto generador y A un algoritmo que resuelve AGL en G respecto a X. Si H está cuasi-isométricamente embebido en G, entonces para cada $w \in H$ dado como una palabra en $F(X)$, el algoritmo A devuelve una palabra $w' \in F(X)$ tal que $l_Y(w) \leq \mu l_X(w') + d$ para μ y d constantes que dependen de A y de H.*

Demostración: Ver [13] ■

Cuando trabajamos con un problema queremos acotar su complejidad tanto inferiormente como superiormente. No queremos que sea demasiado fácil, ya que sería muy fácil de resolver, pero tampoco queremos que nos cueste demasiado generar las instancias del problema. Veamos esto con dos ejemplos:

Ejemplo 3.3.5 *Supongamos que queremos generar un subgrupo. Fijamos cuatro números naturales K_0, K_1, L_0, L_1, y elegimos de manera aleatoria un número natural k que pertenece al intervalo $[K_0, K_1]$. Luego elegimos k palabras $w_1, \ldots, w_k$ en $F(X)$ de manera aleatoria cuyas longitudes estén en el intervalo $[L_0, L_1]$. Obtenemos así una descripción $(w_1, \ldots, w_k)$ correspondiente a un subgrupo de G.*

Ejemplo 3.3.6 *Supongamos que queremos generar de manera aleatoria instancias para SCSP*. Primero fijamos dos números naturales k y m y elegimos $L_0, L_1, N_0, N_1, P_0, P_1$ número naturales. A los inputs de SCSP* los denotamos por $\alpha = (T, b)$ con $T = (a_1, \ldots, a_k, u_1, \ldots, u_m)$ y $b = (v_1, \ldots, v_m)$. Así, elegimos k palabras $a_1, \ldots, a_k$ en $F(X)$ de manera aleatoria, con longitudes en el intervalo $[L_0, L_1]$. Después elegimos m palabras $u_1, \ldots, u_m$ en $F(X)$ con longitudes en el intervalo $[N_0, N_1]$. Seleccionamos de manera aleatoria un elemento w del subgrupo $A =< a_1, \ldots, a_k >$ y lo expresamos como producto de los generadores de A, es decir, $w = a_{i_1} \cdots a_{i_c}$ con un número de factores c en $[P_0, P_1]$. Por último, calculamos $u_i^w = v_i$.*

3.4. El protocolo de Anshel-Anshel-Goldfeld

Tenemos un grupo G finitamente generado, con X conjunto generador. Los elementos de G son las claves de un protocolo criptográfico y el objetivo es que Alice y Bob se pongan de acuerdo en una clave, sin necesidad de contar con un canal seguro para comunicarse. El protocolo consiste en lo siguiente:

- Tanto Alice como Bob eligen un subgrupo de G de manera aleatoria. A los subgrupos elegidos los denotamos por $A =< a_1, \ldots, a_m >$ y $B =< b_1, \ldots, b_n >$ respectivamente y son públicos, es decir, todo el mundo conoce los generadores de A y B. Los generadores a_i, b_j están descritos como palabras en los generadores en X, es decir, como elementos del grupo libre $F(X)$.
- Alice y Bob, cada uno por su lado, eligen aleatoriamente un elemento de su subgrupo. Para ello cada uno elige una palabra $u_A = u(x_1, \ldots, x_m) \in F_m$ y $v_B \in F_n$ y la evalúa en sus generadores. Así Alice obtiene el elemento $a = u(a_1, \ldots, a_m) \in A$ y Bob obtiene $b = v(b_1, \ldots, b_n) \in B$.
- Alice calcula los conjugados $b_1^a, \ldots, b_n^a$ y se los envía a Bob.
- Una vez Alice ha recibido $a_1^b, \ldots, a_m^b$ y calcula $a^{-1}a^b$.
- Bob repite lo mismo que Alice con sus elementos.
- La clave que comparten es $K = a^{-1}a^b = (b^a)^{-1}b = [a, b]$.

¿Cómo de seguro es este protocolo? Intentaremos dar una respuesta en el capítulo siguiente. Partimos de la idea de que Eve quiere conocer la clave que comparten Alice y Bob pero sólo conoce los elementos que se han hecho públicos o que han compartido luego el problema que Eve tiene que resolver es el siguiente:

Problema de AAG: Dados un grupo G, elementos $a_1, \ldots, a_m, b_1, \ldots, b_n$ en G y $b_1^a, \ldots, b_n^a, a_1^b, \ldots, a_m^b$, encontrar $[a, b]$.

Este problema no es un problema estándar en Teoría de Grupos y no sabemos gran cosa acerca de su complejidad computacional, por lo que consideramos conveniente reducirlo a otros problemas que ya nos son conocidos, si bien la relación entre estos problemas y el problema de AAG no está clara.

Lo que nos interesa en relación a estos problemas no es si podemos encontrar la solución al sistema, puesto que ya sabemos que existe, si no encontrar un algoritmo que la devuelva en

un tiempo aceptable, es decir, un algoritmo eficiente que funcione en tiempo polinomial en el tamaño de los inputs. La relación entre el problema AAG y SCSP* la da el siguiente resultado.

Proposición 3.4.1 *Para cualquier grupo G, el problema AAG se puede reducir en tiempo lineal a SCSP*.*

Demostración: Supongamos que tenemos el grupo G, los subgrupos $A =< a_1, \ldots, a_m >$ y $B =< b_1, \ldots, b_n >$ y los elementos $b_1^a, \ldots, b_n^a, a_1^b, \ldots, a_m^b$. Si SCSP* relativo a los subgrupos A y B es decidible en G, entonces resolviendo el sistema

$$b_1^x = b_1^a, \ldots, b_n^x = b_n^a \tag{3.10}$$

en A podemos encontrar una solución, que denotamos por $f \in A$. Análogamente, resolviendo

$$a_1^y = a_1^b, \ldots, a_m^y = a_m^b \tag{3.11}$$

en B encontramos una solución $g \in B$.

Las soluciones del sistema (3.10) son de la forma ca donde c es un elemento cualquiera de $C_G(B)$, el *centralizador de B en G* y las soluciones del sistema (3.11) son de la forma db con $d \in C_G(A)$, luego existen $c \in C_G(B)$ y $d \in C_G(A)$ con $f = ca$ y $g = db$. Pero $[f, g] = [ca, db] = [a, b]$, por lo que nos daría una solución para AAG. ■

En el siguiente capítulo analizaremos la dificultad del SCSP* en varios grupos y veremos algunos ataques que han resultado efectivos en el protocolo AAG desde el punto de vista asintótico.

Capítulo 4

Ataques de longitud y ataques cociente

En este capítulo analizaremos dos tipos de ataques frente al protocolo Anshel-Anshel-Goldfeld. El primero que introducimos es el ataque de longitud, LBA, y después veremos los ataques cociente.

4.1. Ataques de longitud (LBA)

Los *ataques de longitud*, que denotamos LBA, atacan AAG resolviendo las ecuaciones conjugadas en las instancias del protocolo (ver Proposición 3.4.1). Veremos LBA como un algoritmo de búsqueda parcial para un tipo particular de SCSP* en un grupo G. A lo largo de esta sección comprobaremos que este tipo de ataque resulta efectivo genéricamente en grupos libres, así como en grupos de FB_{exp}.

Sea G un grupo con X un conjunto generador finito de G. Supongamos que tenemos un sistema de ecuaciones $u_i^x = v_i$ con $i = 1, \ldots, m$ que tiene solución en un subgrupo $A =< Y >$ generado por un conjunto finito Y de elementos en G y queremos encontrar una solución en A, es decir, queremos resolver SCSP*. LBA se basa en que la siguiente suposición se verifique de forma genérica en el grupo G:

(L) Para elementos $w, y_1, \ldots, y_k \in G$ elegidos de manera aleatoria, el elemento w tiene longitud l_X minimal de entre todos los elementos de la forma w^y dónde y pertenece al subgrupo de G generado por $y_1, \ldots, y_k$. Es decir, más precisamente existe un subconjunto genérico A de $< y_1, \ldots, y_k >$ tal que $l_X(w^y) > l_X(w)$ para todo $y \in A$ con $\mu(A) = 1$ para cierta medida

μ.

No queda claro si estas suposiciones son correctas para un grupo de plataforma determinado, pero volveremos a esta cuestión más adelante cuando estudiemos este tipo de ataque en grupos libres. De momento suponemos que existe un algoritmo A que calcula $l_X(w)$ para todo w en G.

Consideramos los conjugados de Alice, $b_1^a, \ldots, b_n^a$ dónde $a = a_{s_1}^{\epsilon_1} \cdots a_{s_L}^{\epsilon_L}$. Cada b_i^a es el resultado de una secuencia de conjugaciones de b_i por los generadores de A:

$$\begin{array}{ccc} & b_i & \\ & \downarrow & \\ a_{s_1}^{-\epsilon_1} & b_i & a_{s_1}^{\epsilon_1} \\ & \downarrow & \\ a_{s_2}^{-\epsilon_2} a_{s_1}^{-\epsilon-1} & b_i & a_{s_1}^{\epsilon_1} a_{s_2}^{\epsilon_2} \\ & \downarrow & \\ & \cdots & \\ a_{s_L}^{-\epsilon_L} \cdots a_{s_2}^{-\epsilon_2} a_{s_1}^{-\epsilon_1} & b_i & a_{s_1}^{\epsilon_1} a_{s_2}^{\epsilon_2} \cdots a_{s_L}^{\epsilon_L} \end{array}$$

Esta sucesión es la misma para cada b_i y está definida por el elemento a. Nuestro objetivo es revertir esta secuencia para recuperar cada factor $a_{s_j}^{\epsilon_j}$ y obtener el elemento a como producto de generadores de A. Esto es lo que hace el siguiente algoritmo en el que los inputs son las tuplas $(a_1, \ldots, a_m)$, $(b_1, \ldots, b_n)$ y $(c_1, \ldots, c_n)$ y un output es un elemento a de $< a_1, \ldots, a_m >$ tal que $b_i^a = c_i$ para todo $i = 1, \ldots, n$.

- **Entrada:** Tres tuplas $(a_1, \ldots, a_m), (b_1, \ldots, b_n)$ y $(c_1, \ldots, c_n)$ formadas por elementos de un grupo $G =< X >$.
- **Salida:** Un elemento $a \in < a_1, \ldots, a_m >$ con $b_i^a = c_i$ para todo $i = 1, \ldots, n$.
- (**Inicialización**)
  - Sea $x = 1$ dónde 1 es la identidad en G.
  - Sea $l = \sum_{i=1}^{n} l_X(c_i)$ para $i = 1, \ldots, n$.
- (**Bucle principal**) Para cada $i = 1, \ldots, m$ y $\epsilon = \pm 1$ calculamos
  - Si $b_i = c_i$ para todo i, entonces devuelve a.

- Para cada $i = 1, \ldots, n$ y cada $\epsilon = \pm 1$ calculamos $l_{i,\epsilon} = \sum_{j=1}^{n} l_X(a_i^{\epsilon} c_j a_i^{-\epsilon})$.
- Si $l_{i,\epsilon} \geq l$ parra todo $i = 1, \ldots, n$ y todo $\epsilon = \pm 1$ entonces devuelve *Fallo*.
- En otro caso se eligen i y ϵ que dan el menor valor de $l_{i,\epsilon}$ y cambiamos a por $aa_i^{\epsilon_i}$
- Vuelta al punto inicial de bucle.

La idea del algoritmo es que si se cumple la hipótesis (L) entonces el proceso de construcción de $c_1, \ldots, c_n)$ a partir de $a = a_{S_1}^{\epsilon_1} \cdots a_{S_L}^{\epsilon_L}$, como se refleja en el esquema previo al algoritmo en el que se ve la secuencia de conjugaciones, los valores intermedios $a_{S_i}^{-\epsilon_i} \cdots a_{S_1}^{-\epsilon_1} b_j a_{S_1}^{\epsilon_1} \cdots a_{S_i}^{\epsilon_i}$ van creciendo de longitud genéricamente y por tanto al intentar deshacer el proceso vamos buscando el camino en el que la longitud vaya decreciendo.

Aunque esto puede parecer extraño es bastante común pues es más probable que al elegir los $a_{S_i}^{\epsilon_i}$ al azar no tomemos los símbolos del principio y final en las formas normales de los valores intermedios y en consecuencia, en general, en el proceso la longitud tiende a crecer.

Por otro lado es común que no exista un procedimiento efectivo para calcular de forma precisa la longitud l_X. Pero a menudo basta con encontrar estimaciones con las que se puede hacer efectivo el procedimiento anterior.

4.1.1. LBA en grupos libres

En esta subsección vamos a analizar el ataque LBA en grupos libres.

Sean $k \in \mathbb{N}$ fijo, $Y = \{y_1, \ldots, y_k\} \subseteq F(X)^k$ un conjunto de palabras y $\lambda \in (0, \frac{1}{2})$. Decimos que Y satisface la $\lambda-$*condición* si para cualquier u, v en Y, con $u^{-1} \neq v$ se tiene $\frac{l_X(u)+l_X(v)-l_X(uv)}{2} < \lambda \min\{l_X(u), l_X(v)\}$, es decir, que el número de elementos cancelados al hacer el producto de u por v es estrictamente menor que $\lambda \min\{l_X(u), l_X(v)\}$. Esto sirve como una medida del crecimiento de la longitud cuando multiplicamos por elementos de Y. El siguiente teorema, que puede encontrarse en [12], muestra que la $\lambda-$condición es mucho más común de lo que podía parecer en un principio.

Teorema 4.1.1 *Sea $\lambda \in (0, \frac{1}{2})$. El conjunto S de $k-$tuplas $(u_1, \ldots, u_k)$ en $F(X)^k$ que satisfacen la λ-condición es exponencialmente genérico.*

Lema 4.1.2 *Si $Y = \{y_1, \ldots, y_k\}$ satisface la $\lambda-$condición para algún λ en $(0, \frac{1}{2})$ entonces :*

(1) *Y es reducido de Nielsen. En particular, Y genera un subgrupo libre de rango k y por tanto*

cualquier $w \in < Y >$ *se puede representar de manera única como una palabra reducida en los generadores de* Y.

(2) *La longitud geodésica de los elementos de* $< Y >$ *es computable en tiempo lineal.*

Demostración: Demostraremos sólo la primera afirmación. La demostración de la otra afirmación se puede encontrar en [3]. Queremos demostrar que Y es un conjunto reducido de Nielsen. Recordando la Definición 1.1.6 primero comprobamos que $y_j \neq 1$ para todo $j = 1, \ldots, k$. Supongamos sin pérdida de generalidad que $y_1 = 1$ entonces para cualquier $j = 2, \ldots, k$ se tendría $\frac{l_X(1)+l_X(y_j)-l_X(y_j)}{2} < \lambda \min\{l_X(1), l_X(y_j)\}$, pero $l_X(1) = 0$ obteniendo así $0 < 0$. Por lo que $1 \notin Y$.

Lo siguiente que debemos comprobar es que dados dos y_j, y_h con $j \neq h$ e $y_j y_h \neq 1$ $l_X(y_j y_h) \geq l_X(y_j), l_X(y_h)$. Para esto supongamos sin pérdida de generalidad que $l_X(y_j) \leq l_X(y_h)$, entonces

$$\frac{l_X(y_j) + l_X(y_h) - l_X(y_j y_h)}{2} < \lambda \min\{l_X(y_j), l_X(y_h)\} = \lambda l_X(y_j)$$

lo que implica

$$l_X(y_h) < (1 - 2\lambda) l_X(y_j) + l_X(y_h) < l_X(y_j y_h).$$

Como $\lambda \in (0, \frac{1}{2})$ se tiene $2\lambda < 1$ por lo que

$$l_X(y_j) + l_X(y_h) - l_X(y_j) < l_X(y_j) + l_X(y_h) - 2\lambda y_j < l_X(y_j y_h),$$

entonces

$$l_X(y_h) < l_X(y_j y_h)$$

y queda así demostrada la segunda condición.

Para la tercera condición tenemos que demostrar que para todo y_j, y_h e y_l con $y_j y_h \neq 1$ $y_h y_l \neq 1$ se verifica $l_X(y_j y_h y_l) > l_X(y_j) - l_X(y_h) + l_X(y_l)$. Tenemos

$$\begin{aligned}
l_X(y_j y_h y_l) &= l_X(y_j y_h) + l_X(y_h y_l) - l_X(y_h) \\
&> l_X(y_j) + l_X(y_h) - 2\lambda \min\{l_X(y_j), l_X(y_h)\} + l_X(y_h) + l_X(y_l) - \\
&\quad 2\lambda \min\{l_X(y_h), l_X(y_l)\} - l_X(y_h) \\
&> l_X(y_j) + l_X(y_h) + l_X(y_l) - \min\{l_X(y_j), l_X(y_h)\} - \min\{l_X(y_h), l_X(y_l)\} \\
&\geq l_X(y_j) - l_X(y_h) + l_X(y_l)
\end{aligned}$$

La última desigualdad se obtiene considerando caso por caso la ordenación entre $l_X(y_j), l_X(y_h), l_X(y_l)$.
■

Como consecuencia inmediata del Teorema 4.1.1 y del Lema 4.1.2 deducimos el siguiente corolario.

Corolario 4.1.3 *El conjunto de $k-$tuplas que son reducidas de Nielsen en $F(X)$ es un conjunto exponencialmente genérico.*

El siguiente resultado muestra que el ataque LBA es efectivo en grupos libres:

Teorema 4.1.4 *En grupos libres el ataque LBA resuelve SCSP* en un conjunto exponencialmente genérico de inputs en tiempo lineal.*

Demostración: Sean n y m enteros fijos. Sea S el conjunto de $(n+m)-$tuplas $(u_1, \ldots, u_n, a_1, \ldots, a_m) \in F(X)^{n+m}$ que cumplen la $\frac{1}{4}-$condición. Por Teorema 4.1.1, el conjunto S es exponencialmente genérico. Eso implica que el conjunto de inputs $(a_1, \ldots, a_m), (u_1, \ldots, u_n), (v_1, \ldots, v_n)$ para SCSP* para los que $(a_1, \ldots, a_m, u_1, \ldots, u_n) \in S$ es exponencialmente genérico. Consideremos uno de esos inputs y sea $Z = \{a_1, \ldots, a_m, u_1, \ldots, u_n\}$. Por el Lema 4.1.2 $F(Z)$ es libre de rango $m+n$ y l_Z es computable en tiempo lineal. Además, como Z cumple la $\frac{1}{4}-$condición, la condición (L) se verifica y por tanto el ataque LBA es efectivo en este caso. ■

Como vimos al final del capítulo anterior, Proposición 3.4.1, el problema AAG se reduce a SCSP* en tiempo lineal, por lo que:

Corolario 4.1.5 *Sea F un grupo libre, el problema AAG en F es decidible en tiempo lineal en un conjunto de inputs exponencialmente genérico.*

Esto nos dice que cuando apliquemos el protocolo de Anshel-Anshel-Goldfeld, utilizar un grupo libre no es seguro, ya que el corolario anterior da la idea de que el adversario no tardaría mucho en romper el criptosistema.

4.1.2. LBA en grupos de FB_{exp}

Veremos que LBA funciona en grupos que en principio parecen alejados de grupos libres. Recordamos que los inputs de SCSP* son de la forma $\alpha = (T, b)$ con $T = (a_1, \ldots, a_k, u_1, \ldots, u_m) \in F(X)^{k+m}$ y $b = (v_1, \ldots, v_m)$ de manera que $v_i = u_i^x$ tiene solución en el subgrupo $A =< a_1, \ldots, a_k >$.

Observación 4.1.6 1) *La elección de la tupla $T = (a_1, \ldots, a_k, u_1, \ldots, u_m) \in F(X)^{k+m}$ co-*

rresponde a la elección de los generadores elegidos en el Ejemplo 3.3.5.

2) *Las propiedades asintóticas de los subgrupos generados por T corresponden a las propiedades asintóticas de los subgrupos vistas en el capítulo anterior.*

Lema 4.1.7 *Sean G un grupo y X un conjunto finito generador de G. Sea $I_{k,m}$ el conjunto de inputs (T, b) para LBA en G con $T \in F(X)^{k+m}$ y $b \in F(X)^m$. Sea*

$$I_{free} = \{(T, b) \in I_{k,m};\ T \text{ genera un subgrupo libre de rango } k+m \text{ en } G\}$$

Supongamos que existe un subconjunto S de I_{free} que es exponencialmente genérico y un algoritmo A que calcula la longitud l_T de los elementos de $< T >$, con (T, b) en S, cuando los elementos vienen dados como palabras de $F(X)$. Entonces existe un subconjunto S' de I_{free} exponencialmente genérico tal que LBA se detiene en los inputs de S' y devuelve una solución a SCSP en como mucho tiempo cuadrático más el tiempo que tarda el algoritmo A.*

Demostración: El resultado se sigue del Teorema 4.1.4. ■

Teorema 4.1.8 *Sean G un grupo y X un conjunto generador finito de G y supongamos que G está en FB_{exp}. Entonces existe un subconjunto S de $I_{k,m}$ exponencialmente genérico que contiene todos los inputs de LBA para los que LBA relativo a l_T se detiene en esos inputs y ofrece una solución a SCSP*. Además, la complejidad temporal de LBA en S es como mucho cuadrática más la complejidad del algoritmo A que computa l_T en los elementos de $< T >$ cuando vienen dados como palabras de $F(X)$.*

Demostración: Por el Lema 4.1.7 existe un subconjunto S de I_{free} exponencialmente genérico tal que LBA se para en los inputs de S y devuelve una solución a SCSP*. Además la complejidad temporal de LBA en S es como mucho cuadrática más el tiempo que tarde A. Basta ver que I_{free} es exponencialmente genérico en el conjunto de todos los inputs para LBA en G, que denotaremos por I. Por la Observación 4.1.6 la densidad asintótica de I_{free} en I es la misma que la de T en $F(X)^{k+m}$. Pero T cumple la propiedad de la base libre en G y $G \in FB_{exp}$. Así, I_{free} es exponencialmente genérico en I. ■

4.2. Ataques cociente

Los ataques cociente son algoritmos que traspasan la solución de un problema en un grupo G a la solución de dicho problema en algunos grupos cociente G/N en los que se conoce el

algoritmo para resolver el problema. En esta sección enfocaremos los problemas de pertenencia y conjugación desde el punto de vista de los grupos libres a través de ataques cociente. Lo primero que haremos será ver una cota temporal superior para SCSP* en grupos libres y para terminar estudiaremos MSP y SCSP* en grupos algunos grupos.

Observación 4.2.1 *En la demostración del Teorema 3.3.2 vemos que una de las principales dificultades que se nos presentan cuando queremos resolver SCSP es el cálculo del conjunto V, que es la intersección de dos subgrupos (o de dos de sus clases laterales). Trabajando en un grupo libre, por [3], podemos resolver este problema en tiempo como mucho cuadrático.*

El siguiente resultado es consecuencia del Teorema 3.3.2.

Corolario 4.2.2 *SCSP* en grupos libres es decidible en como mucho tiempo cuadrático.*

Demostración: El algoritmo utilizado en el Teorema 3.3.2 concluye que dado el sistema de ecuaciones conjugadas: no hay solución, que la solución es única o da la solución como Cd clase lateral, con C el centralizador de algún elemento. En el primer caso SCSP* no tendría solución. En el segundo caso deberíamos comprobar si la solución que devuelve el algoritmo está en el subgrupo que nos interesa, al que llamamos A, construimos el autómata que acepta al subgrupo A y comprobamos si la solución también es aceptada. Esto tarda un tiempo $n \log_2^* n$, ver [3]. Por último, si el algoritmo ha devuelto Cd debemos comprobar si $Cd \cap A$ es vacío o no, y por la Observación 4.2.1 esto lo podemos hacer en como mucho tiempo cuadrático. ■

Como vimos al final del capítulo anterior, Proposición 3.4.1, el problema AAG se reduce a SCSP* en tiempo lineal, por lo que tenemos el siguiente corolario.

Corolario 4.2.3 *Sea F un grupo libre, el problema AAG en F es decidible en como mucho tiempo cuadrático en el tamaño del input.*

Por último tratamos MSP y SCSP* en grupos con "buenos" cocientes. Sea G un grupo fijo con X un conjunto generador finito de G, N subgrupo normal de G, G/N cociente de G y $\varphi : G \to G/N$ el epimorfismo canónico. Denotaremos por $\varphi(u_i)$ a la imagen de u_i por φ en G/N. Sea $H =< u_1, \ldots, u_k >$, representamos los elementos de G como palabras em X y abusamos de la notación denotándolos con el mismo símbolo. Para resolver MSP en H tenemos el siguiente algoritmo heurístico:

Algoritmo 4.2.4 - *Input:* $(w, u_1, \ldots, u_k) \in F(X)^{k+1}$ *representando elementos de G.*

- *Output: Representación de w como una palabra en $u_1, \ldots, u_k$, o Fallo.*

- *Computación:*

  A. *Calculamos* $\varphi(u_1), \ldots, \varphi(u_k)$ *generadores de* $\varphi(H)$ *en* G/N.

  B. *Calculamos* $\varphi(w)$ *y resolvemos MSP para* $\varphi(w)$ *en* $\varphi(H)$. *Encontramos* $W(\varphi(u_1), \ldots, \varphi(u_k))$ *representación de* $\varphi(w)$ *como producto de los generadores de* $\varphi(H)$ *y sus inversos.*

  C. *Comprobar si* $W(u_1, \ldots, u_k) = w$ *en* G. *Si coinciden, devuelve* w, *si no, devuelve Fallo.*

Vemos que para poder aplicar este algoritmo necesitamos saber resolver MSP en G/N y comprobar el resultado en G (problema de la palabra). Si se dan estas condiciones el algoritmo es correcto, pero aún siendo correcto, ¿es útil en algún grupo?

Teorema 4.2.5 *Sea* G *grupo fijo con* X *un conjunto generador finito de* G *y con problema de la palabra resoluble en tiempo* $O(f_1)$. *Supongamos que* G/N *es un cociente de* G *tal que*

1) $G/N \in FB_{exp}$.

2) *El epimorfismo canónico* $\varphi : G \to G/N$ *es computable en tiempo* $O(f_2)$.

3) *Para todo* $k \in \mathbb{N}$, *existe un algoritmo* A_k *de tiempo* $O(f_3)$ *que resuelve MSP en* G/N *para un subconjunto exponencialmente genérico* M_k *de* $F(X)^k$ *de descripciones de subgrupos* $k-$*generados en* G/N.

Entonces, para todo k, *el Algoritmo 4.2.4 resuelve MSP en tiempo* $O(f_1 + f_2 + f_3)$ *en un subconjunto exponencialmente genérico* T_k *de* $F(X)^k$ *de descripciones de subgrupos de* G.

Demostración: Sea S_k el subconjunto de $k-$tuplas de $F(X)^k$ cuyas imágenes en G/N generan un subgrupo libre. Por la hipótesis 1), S_k es exponencialmente genérico, así como también lo es M_k conjunto de $k-$tuplas de $F(X)^k$ en el que se aplica el algoritmo A_k. Así, $T_k = S_k \cap M_k$ también es un conjunto exponencialmente genérico en $F(X)^k$.

Queremos ver que el Algoritmo 4.2.4 se aplica a las tuplas de T_k, y así es. El algoritmo A_k se aplica a los subgrupos generados por las tuplas $Y = (u_1, \ldots, u_k)$ de T_k, por lo que si $\varphi(w) \in \varphi(H) =< \varphi(Y) >$ entonces A_k devuelve la representación de $\varphi(w) = W(\varphi(Y))$ en G/N. Como $Y \in S_k$, el subgrupo $\varphi(H)$ está finitamente generado por $\varphi(Y)$, por lo que φ en inyectiva en H. Así, $w = W(Y)$ en G, como queríamos. ■

Los Teoremas 4.2.5 y 4.1.8 dan lugar al siguiente corolario.

Corolario 4.2.6 *Sea G un grupo como en el Teorema 4.2.5. Entonces, para todo $k, m > 0$ existe un algoritmo $C_{k,m}$ que resuelve SCSP* en un subconjunto de inputs de $I_{k,m}$ exponencialmente genérico con un tiempo $O(n^2 + f_1(n) + f_2(n) + f_3(n))$.*

Bibliografía

[1] M. Agrawal, N. Kayal and N. Saxena, *Primes in P*, Annals of Mathematics 160 (2004), pp. 781 – 793.

[2] T.H. Cormen, C.E. Leierson, R.L. Rivest and C. Stein, *Introduction to Algorithms*, Second Edition, The MIT Press, 2001.

[3] I. Kapovich and A.G. Miasnikov, *Stallings foldings and subgroups of free groups*, J. Algebra 248(2002), pp. 608 – 668.

[4] A. García Nogales, *Teorías de la Medida y de la Probabilidad*, Universidad de Extremadura, 2008.

[5] R. Gilman, A.G Miasnikov, A.D Miasnikov and A. Ushakov, *Generic complexity of algorithmic problems*, Vestnik OMGU Special Issue (2007), pp. 103 – 110.

[6] Y. Gurevich, *Average case completeness*, J. Comput. Syst. Sci. 42 (1991), pp. 346 – 398.

[7] Y. Gurevich, *The Challenger-Solver game: Variations on the theme of P=? NP*. Logic in Computer Science COlumn, The Bulletin of EATCS, pp. 112 – 121, October, 1989.

[8] Y. Gurevich and S. Shelah, *Expected computation time for Hamiltonian Path Problem*, SIAM J. Comput. 16 (1987), pp. 486 – 502.

[9] R. Impagliazzo, *A personal view of average-case complexity*. Proceedings of the 10th Annual Structure in Complexity Theory Conference (SCT'95), pp.134 – 147, 1995.

[10] V. Klee and G. Minty, *How good is the simplex algorithm?*. Inequalities, III (Proc. Third Sympos., Univ. California, Los Angeles, Calif., 1969),pp. 159 – 175. Academic Press, 1972.

[11] W. Magnus, A. Karrass and D. Solitar, *Combinatorial Group Theory*. Springer-Verlag,1977.

[12] A. Martino, E. Turner and E. Ventura, *The density of injective endomorphisms of a free group*, not published.

[13] A. Myasnikov, V. Shpilrain and A. Ushakov, *Group-based cryptography*,Springer, Berlin, 2008.

[14] D.J.S. Robinsons, *A Course in the Theory of Groups*, Second Edition, Springer, New York, 1996.

[15] N. Touikan, *A Fast Algorithm for Stallings' Folding Process*, Internat. J. Algebra and Comput. 16 (2006), pp. 1031 – 1046.

[16] L. Trevisan, *Computational Complexity*. Standford University, 2010.

Índice terminológico