Práctica 4

Construcción de modelos para el diseño de algoritmos

José-Manuel Colom, Jorge Bernad, Elvira Mayordomo, Carlos Bobed, Gregorio de Miguel

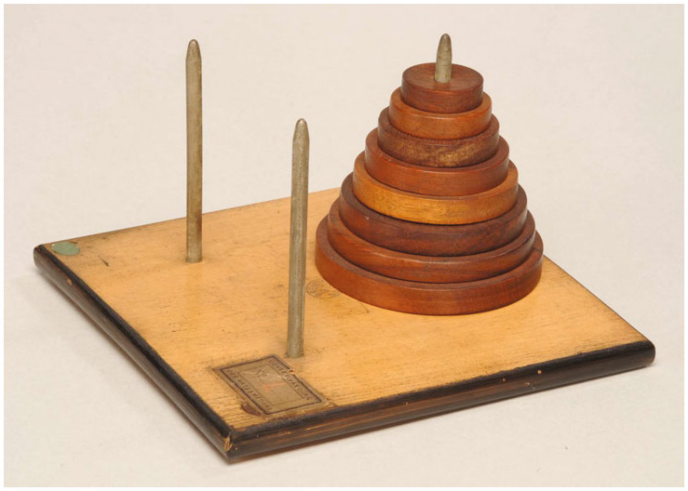
1.1. El Problema de las Torres de Hanoi

Ian Stewart escribe en [2]: *"Mathematics intrigues people for at least three different reasons: because it is fun, because it is beautiful, or because it is useful"*. El puzzle de la Torre de Hanoi (TH) satisface los aspectos anteriormente mencionados. Cubre el aspecto recreativo si se tienen en cuenta las versiones de este puzzle que pueden adquirirse en almacenes especializados en entretenimientos. Su lado estético queda reflejado, por ejemplo, en los versos hindúes y los mosaicos italianos en los que se hay representaciones pictóricas del puzzle. En cuanto a su aplicación práctica, ésta crece continuamente con aplicaciones al mundo de los tests psicológicos y la aplicación de los fundamentos matemáticos detrás de las TH en la Teoría de Códigos o en la descripción de algunos fenómenos físicos.

El problema de la Torre de Hanoi (TH) es un *juego* sobre un puzzle formado por una serie de piezas (discos y palos para colocar los discos), con unas reglas de movimiento de sus piezas, que persigue pasar de una configuración inicial a una configuración final. Una solución del problema de TH es una secuencia de movimientos en el puzzle para obtener la configuración solicitada. El objetivo de este Trabajo de Laboratorio es especificar el problema de TH y algunas variantes, para comprender cómo obtener una solución mediante un algoritmo derivado de su especificación. El método consistirá en la construcción de un modelo formal del puzzle, como por ejemplo un autómata de estados finitos, desde el cual se pueda construir un programa cuya ejecución resolverá el problema al devolver la secuencia de movimientos a realizar.

El material presentado en este opúsculo se ha extraido de o se ha inspirado en el libro [1]. A lo largo de estas páginas se indicará en qué parte de este libro se pueden encontrar extensiones de los contenidos presentados aquí.

Figura 1.1: La Torre de Hanoi original - ©Musée des arts et métiers – CNAM, Paris. Photo: Michèle Favareille. `http://www.arts-et-metiers.net/` (figura 0.5 de [1])



1.2. Reseña histórica

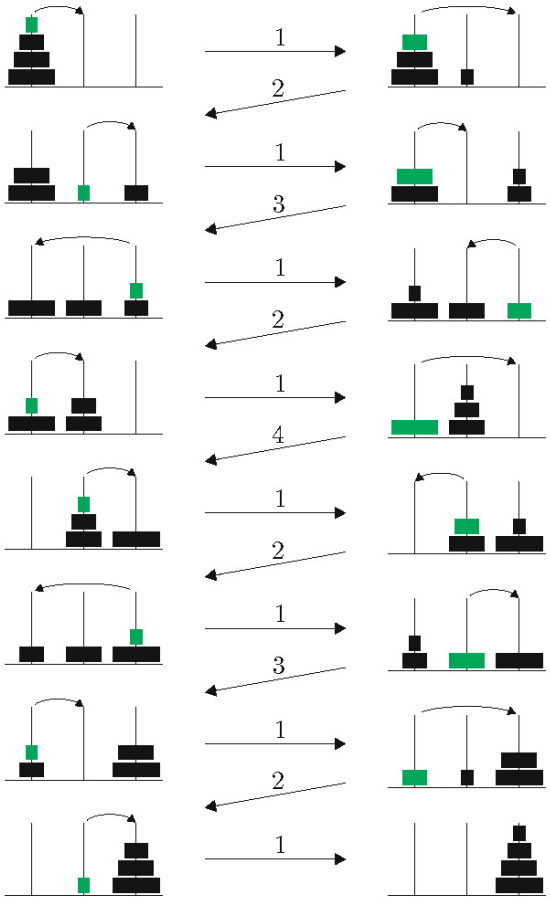
La Torre de Hanoi (también llamada la Torre de Brahma o la Torre de Lucas) fue un juego concebido por el matemático francés Édouard Lucas en el siglo XIX (Figura 1.1). Está asociado con una leyenda de un gran templo hindú en Benares en la que se afirmaba que el rompecabezas se usaba para disciplinar la mente de los sacerdotes jóvenes. En la leyenda, los jóvenes sacerdotes se enfrentaban a una pila (la Torre de Brahma) de 64 discos de oro cuidadosamente apilados en uno de tres postes. En la pila, cada disco estaba encima de un disco un poco más grande, el disco inferior reposaba en el suelo. El reto propuesto a los sacerdotes era reconstruir la pila en un poste diferente a aquel en el que encontraron inicialmente la pila. Para reconstruir la pila, solo lo podían hacer moviendo los discos a otros postes, pero solo un disco en cada movimiento. En estos movimientos se debían respetar dos reglas (las Leyes de Brahma): (Regla 1) Para mover un disco, éste no puede tener discos apoyados sobre él; (Regla 2) Un disco nunca puede colocarse encima de un disco más pequeño que él.

Los sacerdotes encontraron la forma más eficiente de resolver el problema. Sin embargo, como se verá más adelante, se puede calcular que llevar a la práctica dicha solución, incluso moviendo los discos a una velocidad de uno por segundo, es inviable. Efectivamente, tomaría casi 585 mil millones de años (584.942.417.355 años) terminar el trabajo. ¡Eso es más de 42 veces la edad del universo (13,787 mil millones de años)!

1.3. Definiciones y notación básica

En esta sección se analiza el problema TH clásico sobre 3 postes. Además de introducir algunos aspectos sobre la notación más cómoda para razonar so-

Figura 1.2: Solución de la Torre de Hanoi con 4 discos (figura 2.2 de [1])



bre este problema, también se analizan los algoritmos conocidos como método iterativo y método recursivo para obtener la secuencia óptima de movimientos (secuencia con el mínimo número de movimientos) para trasladar la pila completa situada en un poste, a otro poste diferente. Este problema se denominará *Problema TH Tipo 0*.

Estados regulares y estados perfectos en TH

El TH consta de tres postes verticales, sobre una placa base horizontal, y un número de discos de diámetros mutuamente diferentes. Cada disco está perforado en su centro para que pueda insertarse en los postes y así formar una pila. Se denomina ***estado regular del problema de TH*** a cualquier distribución de los discos en los tres postes que satisfaga que todo disco está apoyado en un disco de mayor tamaño que él o sobre la placa base. Se denomina ***estado perfecto del problema de TH*** a un estado regular en el que todos los discos se encuentran apilados en el mismo poste. Se define el conjunto T como:

$T = \{0, 1, 2\}$ para etiquetar 0, 1 y 2 los postes del problema TH. El número total de discos del problema TH se denotará como $n > 0$. Los discos se etiquetarán $1, 2, \ldots, n$, con 1 el más pequeño, y asignando números crecientes con el orden creciente del diámetro de los discos.

Cada estado regular se representa de forma única con la n–tupla

$$s = s_n \ldots s_1 \in T^n$$

donde $s_k \in T$ es el poste en el que se encuentra el disco k. Por ejemplo, en el problema de TH con 5 discos, el estado perfecto en el que los cinco discos se encuentran sobre el poste 0 se denota $00000 = 0^5$. El estado regular no perfecto en el que los discos 5 y 3 se encuentran en el poste 0, los discos 4 y 1 sobre el poste 2 y el disco 2 en el poste 1, se denota 02012.

Sea $m \in [1, \ldots, n]$ y $s \in T^m$, entonces $s \cdot T^{n-m} = sT^{n-m} = \{st | t \in T^{n-m}\}$. Obviamente, $|sT^{n-m}| = |T^{n-m}|$. Además, $T^n = \sum_{s \in T^m} sT^{n-m}$. Por ejemplo, para el caso $m = 1$, el conjunto T^n se descompone en los tres conjuntos (disjuntos) $0T^{n-1}$, $1T^{n-1}$ y $2T^{n-1}$, cada uno caracterizado por la posición del disco más grande.

Movimientos válidos en TH

Un disco solo se puede mover si se encuentra en la parte superior de una pila en un poste, sin discos apoyados sobre él. El destino de un disco en un movimiento será a la parte superior de la pila (posiblemente vacía) en otro poste, siempre que se apoye sobre un disco de mayor tamaño que él o la placa base (la regla divina). El objetivo del *Problema TH Tipo 0* es pasar de un estado perfecto a otro estado perfecto mediante la secuencia más corta de movimientos válidos (de mínimo número de movimientos). En la Figura 1.2 se presenta una solución para el TH con 4 discos. La columna central de la figura muestra las etiquetas de los discos que se mueven entre dos configuraciones consecutivas.

Solución óptima para 3 postes en el *Problema TH Tipo 0*

En una solución óptima, resulta obvio que el disco 1 debe moverse en el primer paso y se deberá mover también a continuación del segundo movimiento, que no seá del disco 1, y así sucesivamente. Por lo tanto, el disco 1 se moverá en cada movimiento impar (véanse los movimientos de la columna de la izquierda a la derecha en la Figura 1.2). Los movimientos pares vienen dados luego por la regla divina ya que el disco superior más pequeño, que no es el disco 1, es el que se tiene que mover. Además, los movimientos de los discos 2 a n deben realizarse de una forma que en realidad se trata de encontrar, otra vez, una solución óptima con respecto a la misma tarea pero con sólo $n-1$ discos (véase la columna de la izquierda en la Figura 1.2). Esto se debe a que el disco 1 siempre se puede quitar de la cima de una pila para mover otro disco.

Si se considera la secuencia de números enteros siguiente, denominada *secuencia de Gros*,

$$g = 1, 2, 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2, 1, 5, 1, 2, 1, 3, 1, \ldots$$

Se puede demostrar la siguiente propiedad de estas secuencias,

Propiedad 1 *Para cada* $k \in \mathbb{N}$

$$g_k = \begin{cases} 1 & k \text{ impar}, \\ g_{k/2} + 1 & k \text{ par}. \end{cases}$$

A partir de estos comentarios y el tipo de secuencia definida,se pueden formular los siguientes resultados sobre el número de movimientos en la secuencia óptima.

Teorema 1 *El número de movimientos de una solución óptima para el problema TH con* $n \in \mathbb{N}$ *se puede alcanzar en no menos de* $M_n = (1^n)_{(2}$, *donde* 1^n *representa un número binario compuesto por* n *unos consecutivos.*

Demostración 1 *La demostración es por inducción. El enunciado es trivialmente cierto para el caso* $n = 0$. *Si el problema se puede resolver para* n *discos en* M_n *movimientos, se puede resolver para* $n + 1$ *discos en* $M_n + 1 + M_n = 2M_n + 1 = M_{n+1}$ *movimientos transfiriendo la torre formada por los* n *discos de menor tamaño al palo del medio, luego el disco* $n+1$ *se mueve al palo objetivo del desplazamiento total, y finalmente la torre de los* n *discos más pequeños también se mueve al palo objetivo final. Antes del primer movimiento del disco* $n + 1$ *en cualquier solución, se debe transferir una torre de* n *discos desde desde el palo inicial a otro palo, lo que requiere al menos* M_n *movimientos por la hipótesis de la inducción. Después del último movimiento del disco* $n + 1$ *nuevamente, una torre de* n *cambia de posición desde algun palo al palo que es el objetivo de destino final, consumiendo otros* M_n *movimientos de disco individuales. Dado que el disco* $n + 1$ *tiene que moverse al menos una vez, la solución necesita al menos* M_{n+1} *pasos.*

Teorema 2 *El número de movimientos de una solución óptima para el problema TH con* $n \in \mathbb{N}$ *discos tiene una solución óptima única de longitud* $2^n - 1$. *El* k*-ésimo movimiento en esta solución lo realiza el disco* g_k, $k \in [1 \dots 2^n - 1]$.

Demostración 2 *Procedemos por inducción en* n, *donde en el paso de inducción de* n *a* $n + 1$ *empleamos la solución óptima única de longitud* $2^n - 1$ *que obtenemos por la hipótesis de inducción para los discos* 2 *a* $n + 1$ *en movimientos pares. Los primeros* 2^{n-1} *movimientos impares son realizados por el disco* 1 *de una manera única, de modo que se pueden realizar los movimientos pares. Finalmente, el disco* 1 *hace el último de un total de* $2(2^n - 1) + 1 = 2^{n+1} - 1$ *movimientos. El segundo enunciado del teorema también se puede demostrar por inducción: en el paso de inducción, los movimientos impares* k *se realizan, como hemos visto, por el disco* $1 = g_k$ *y los movimientos pares* k *por el disco* $g_{k/2} + 1 = g_k$.

Una alternativa al paso de inducción en la demostración anterior se puede rehacer en base a la siguiente observación.

Proposición 1 *En la solución óptima para mover* $n \in \mathbb{N}$ *discos de un palo a otro, el disco* $d \in [1 \dots n]$ *se mueve por primera vez en el paso* 2^{d-1} *y por última vez en el paso* $2^n - 2^{d-1}$; *en particular, el disco más grande* n *se mueve exactamente una vez, es decir, en el medio de la solución.*

1.4. Algoritmo de Olive

La discusión previa a la demostración del Teorema 2 ha puesto de manifiesto que el disco 1 se desplaza en cada movimiento impar y los movimientos pares están dictados por la *regla divina*. Sin embargo, ahora hay que analizar la dirección o sentido de los movimientos del disco 1. Se puede observar al considerar un pequeño número de discos que en la solución óptima el disco 1 se mueve cíclicamente entre los postes. Cuando n es impar, se mueve del poste de origen a través del poste objetivo final al poste intermedio; si n es par, el ciclo va desde el poste de origen, a través del poste intermedio, hasta llegar al poste objetivo final. Esta observación la realizó el sobrino de Lucas, Olive, y permite escribir el Algoritmo 1. (Obsérvese que $3 - i - j$ es el número de poste que es diferente tanto de i como de j, si el número de poste es 0, 1, ó 2.)

Algoritmo 1 Algoritmo iterativo de Olive para el cálculo de la secuencia de movimientos más corta para resolver el problema de la Torres de Hanoi entre dos estados perfectos

Entradas
  n: Número de discos $\{n \in \mathbb{N}\}$
  i: Poste origen $\{i \in T = \{0, 1, 2\}\}$
  j: Poste objetivo final $\{j \in T = \{0, 1, 2\}\}$
inicio
  si $n = 0$ or $i = j$ **entonces**
    Fin_Algoritmo
  fin si
  si n es impar **entonces**
    Mover el disco 1 del poste i al poste j
  sino
    Mover el disco 1 del poste i al poste $3 - i - j$
  fin si
  Memorizar el sentido del movimiento del disco 1
  mientras que No todos los discos están en el poste j **hacer**
    Mover legalmente el disco que se puede mover y diferente al 1
    Mover el disco 1 siguiendo su ciclo en el sentido memorizado
  fin mientras que
  Fin_Algoritmo

1.5. Algoritmo recursivo

A continuación, Algoritmo 2, se presenta el algoritmo recursivo clásico para resolver el problema de las Torres de Hanoi entre estados perfectos.

Algoritmo 2 Algoritmo recursivo para el cálculo de la secuencia de movimientos más corta para resolver el problema de la Torres de Hanoi entre dos estados perfectos

Procedure THanoi(n,i,j)
Entradas
n: Número de discos $\{n \in \mathbb{N}\}$
i: Poste origen $\{i \in T = \{0, 1, 2\}\}$
j: Poste objetivo final $\{j \in T = \{0, 1, 2\}\}$
inicio
si $n \neq 0$ and $i \neq j$ **entonces**
$k = 3 - i - j$ {el poste auxiliar diferente a i y a j}
THanoi($n-1$,i,k) {mueve $n-1$ discos pequeños a poste auxiliar}
Mover disco n de i a j {mueve el disco más grande a poste final}
THanoi($n-1$,k,j) {mueve los $n-1$ discos más pequeños al poste final}
fin si
Fin_Algoritmo

Ejercicio 1

El problema de las Torres de Hanoi que se presenta tradicionalmente en los cursos introductorios de Programación se refieren al problema de encontrar la secuencia de movimientos mínima que permite alcanzar un estado perfecto desde otro estado perfecto como se ha definido previamente. El algoritmo presentado para este fin mayoritariamente es el algoritmo recursivo para ilustrar el concepto de recursividad. En menor medida se usa el algoritmo iterativo de Olive.
Se pide una especificación del problema de las Torres de Hanoi para 3 discos y 3 postes con ayuda de autómatas de estados finitos. A partir del autómata que especifique el problema, en los ejercicios siguientes se implementará un programa que resuelva el caso de encontrar una secuencia de movimientos mínima que permita alcanzar un estado regular desde otro estado regular dados a priori.

Para construir la especificación del problema a resolver se recomienda seguir los siguientes pasos,

1. Definición de los estados que pueden alcanzar el conjunto de postes y discos respetando las reglas de posicionamiento de los discos sobre los postes.

2. Definición de las transiciones posibles entre estados. Cada transición, obviamente, se produce como consecuencia del movimiento legal de un disco desde un poste a otro conforme a las reglas de movimiento en este problema. El conjunto de las ternas *(disco poste_ origen poste_ destino)* es el alfabeto que sirve para etiquetar las transiciones del autómata. Por ejemplo, la etiqueta `302` significa que el disco 3, el más grande, está en el palo 0 y se mueve al palo 2.

3. Definición del estado inicial del autómata como la configuración de partida dada de los discos sobre los postes.

4. Definición del estado final del autómata como la configuración a alcanzar desde la de partida mediante movimientos de los discos conforme a las

reglas establecidas.

El lenguaje que reconoce el autómata a construir está formado por las palabras que representan secuencias de movimientos que permiten pasar de la configuración de partida de los discos sobre los postes, a la configuración final a alcanzar especificada.
Ayuda. Para identificar los estados a definir se recomienda utilizar la codificación especificada a continuación.

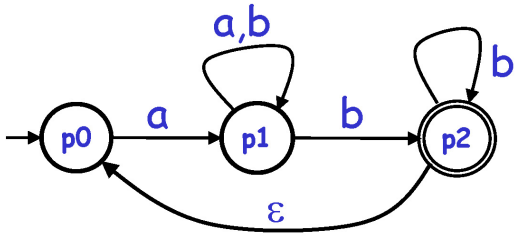
1. Sea $T = \{0, 1, 2\}$ el conjunto de valores para etiquetar los postes 0, 1 y 2.
2. El número total de discos del problema se denotará como $n > 0$.
3. Los discos se etiquetarán $1, 2, \ldots, n$, con 1 el más pequeño, y asignando números crecientes con el orden creciente del diámetro de los discos.
4. Un estado regular se representa con la n–tupla $s = s_n \ldots s_1 \in T^n$, donde $s_k \in T$ es el poste en el que se encuentra el disco k.

Ejemplo. En el problema de TH con 5 discos, el estado perfecto en el que los cinco discos se encuentran sobre el poste 0 se denota $00000 = 0^5$. El estado regular no perfecto en el que los discos 5 y 3 se encuentran en el poste 0, los discos 4 y 1 sobre el poste 2 y el disco 2 en el poste 1, se denota 02012.
Observación. Todo movimiento de un disco, siguiendo las reglas del problema, es reversible. Dos estados, S_1 y S_2, alcanzables por un movimiento de un disco se diferencian en la codificación propuesta en una sola componente y los números que aparecen en la componente diferenciadora de estos estados representan los números de poste entre los que se mueve el disco. Es decir, si $S_1 = s_n^1 \ldots s_1^1 \in T^n$ y $S_2 = s_n^2 \ldots s_1^2 \in T^n$ son mutuamente alcanzables por el movimiento de un disco siguiendo las reglas, entonces existe un $i \in \{1, ..., n\}$ tal que $s_i^1 \neq s_i^2$, y para todo $j \in \{1, ..., n\}$, $j \neq i$, $s_i^1 = s_i^2$. Lo cual quiere decir que el disco i se mueve entre los postes s_i^1 y s_i^2 para hacer mutuamente alcanzables los estados S_1 y S_2. Por ejemplo, $S_1 = 0000$ y $S_2 = 0001$ son mutuamente alcanzables al mover el disco 1 entre los postes 0 y 1.

Ejercicio 2

Para construir un programa que determine el mínimo número de movimientos que permite pasar de un estado regular origen cualquiera, a otro estado regular destino cualquiera, lo más directo es utilizar el diagrama de estados del autómata construido en el Ejercicio 1. Para ello, el programa a construir deberá leer una descripción de dicho diagrama y a partir del diagrama construir la matriz de adyacencia del grafo que se utilizará en el Ejercicio 3 para calcular la secuencia de movimientos mínima.
Se pide diseñar un lenguaje para la descripción de grafos. El lenguaje se formalizará mediante una gramática independiente de contexto a través de la cual se podrá construir una descripción textual de un grafo en un fichero de texto.
Ayuda. Para el diseño del lenguaje de descripción de grafos se recomienda analizar el lenguaje de descripción de Grafos ***DOT***. Al visitar la página

Figura 1.3: Autómata de estados finito que reconoce el lenguaje descrito por la expresión regular $aa^*b(b + a^*b)^*$



`https://graphviz.org/doc/info/lang.html`

se encontrará una gramática que lo describe y partir de la cual se puede construir otra más simple para describir solamente Grafos dirigidos como los diagramas de estado sobre los que se construyen los autómatas. A modo de ejemplo e inspiración, el diagrama de estados presentado en la Figura 1.3 se podría representar textualmente por:

```
graph un_automata
{
p0 ->p1(a);
p1 ->p1(a),p1(b),p2(b);
p2 ->p2(b),p0();
}
```

Se pide además que con la gramática diseñada y usando las herramientas Bison y Flex, se construya un traductor de grafos descritos en el lenguaje diseñado a su correspondiente matriz de adyacencia que represente el grafo para el algoritmo de cálculo que se deberá diseñar en el Ejercicio 3.

Ayuda. Para un grafo dirigido y etiquetado en sus arcos, como los grafos correspondientes a los diagramas de estado subyacentes en un autómata, la matriz de adyacencia es una matriz cuadrada que tiene tantas filas y columnas como número de estados. Es decir, si $V = \{v_1, v_2, \ldots, v_n\}$ es el conjunto de estados del diagrama de estados, la matriz de adyacencia, C, es una matriz de dimensiones $n \times n$, donde el conjunto de filas y columnas está indexado por el conjunto V de estados. El elemento $V[v_i, v_j]$ de la matriz es igual $\emptyset$ si no existe una transición (arco) desde el estado v_i al estado v_j, en caso que exista dicha transición entonces $V[v_i, v_j]$ es igual al símbolo del alfabeto que etiqueta el arco. En el caso que existiesen múltiples arcos desde el estado v_i al estado v_j, entonces el valor de $V[v_i, v_j]$ será igual a la suma formal de los símbolos que etiquetan esos arcos.

Ejemplo. En la Figura 1.3 se describe un Autómata de Estados Finitos no Determinista. La matriz de adyacencia que corresponde al diagrama de estados de este autómata es,

$$\mathbf{C} = \begin{array}{c} \\ p_0 \\ p_1 \\ p_2 \end{array} \begin{array}{c} \begin{array}{ccc} p_0 & p_1 & p_2 \end{array} \\ \begin{pmatrix} \emptyset & a & \emptyset \\ \emptyset & a+b & b \\ \varepsilon & \emptyset & b \end{pmatrix} \end{array}$$

Ejercicio 3

Para construir el programa que determine el mínimo número de movimientos que permite pasar de un estado regular origen cualquiera, a otro estado regular destino cualquiera, a partir de la matriz de adyacencia construida en el Ejercicio 2, se utilizará el método basado en calcular las potencias de la matriz de adyacencia C de un grafo.

Se pide diseñar una función a la que se pasan como parámetros el estado regular origen, v_i, el estado regular destino, v_j, la matriz de adyacencia del grafo, C, y calcula la potencia $k > 0$ más pequeña de la matriz de adyacencia, C^k tal que $C^{k-1}[v_i, v_j] = \emptyset$ y $C^k[v_i, v_j] \neq \emptyset$. El valor devuelto por la función es $C^k[v_i, v_j]$ que representa el conjunto de caminos de longitud k en el grafo que permiten alcanzar v_j desde v_i.

Ayuda. El producto matricial a utilizar en este apartado calcula, por ejemplo, el elemento $C^2[v_i, v_j] = \sum_{k=1}^{n} C[v_i, v_k] \cdot C[v_k, v_j]$, donde n es la dimensión de la matriz cuadrada, el producto es el operador concatenación de palabras y la suma es el operador suma de expresiones regulares.

Ejemplo. En el autómata de la Figura 1.3 estamos interesados en encontrar un camino de longitud mínima que vaya del estado p_0 al estado p_2. Al inspeccionar la matriz de adyacencia se observa que $C[p_0, p_2] = \emptyset$ lo cual significa que no existe ningún camino de longitud 1 que vaya de p_0 a p_2.

$$\mathbf{C} = \begin{array}{c} \\ p_0 \\ p_1 \\ p_2 \end{array} \begin{array}{c} \begin{array}{ccc} p_0 & p_1 & p_2 \end{array} \\ \begin{pmatrix} \emptyset & a & \emptyset \\ \emptyset & a+b & b \\ \varepsilon & \emptyset & b \end{pmatrix} \end{array}$$

Sin embargo, si se calcula la potencia 2 de la matriz de adyacencia, en este caso se obtiene que $C^2[p_0, p_2] = ab \neq \emptyset$, lo cual significa que existe un camino de longitud 2 de p_0 a p_2 y la palabra construida en este recorrido es ab. Este camino es de longitud mínima ya que $C^1[p_0, p_2] = C[p_0, p_2] = \emptyset$.

$$\mathbf{C^2} = \mathbf{C} \cdot \mathbf{C} = \begin{pmatrix} \emptyset & a & \emptyset \\ \emptyset & a+b & b \\ \varepsilon & \emptyset & b \end{pmatrix} \cdot \begin{pmatrix} \emptyset & a & \emptyset \\ \emptyset & a+b & b \\ \varepsilon & \emptyset & b \end{pmatrix} = \begin{pmatrix} \emptyset & aa+ab & ab \\ b & (a+b)^2 & ab+bb \\ b & a & b^2 \end{pmatrix}$$

Por último, obsérvese que, por ejemplo, la suma de C y C^2 describe los caminos de longitud menor o igual que 2 que conectan dos nodos en el grafo.

$$\mathbf{Paths}^{1+2} = \mathbf{C^2} + \mathbf{C} = \begin{pmatrix} \emptyset & a+aa+ab & ab \\ b & (a+b)+(a+b)^2 & b+ab+bb \\ \varepsilon+b & a & b+b^2 \end{pmatrix}$$

Material a entregar y procedimiento de entrega

Ejercicio 1. Incluir en la memoria las explicaciones oportunas sobre la codificación elegida para resolver el problema de las Torres de Hanoi para tres palos y

tres discos, así como una imagen con el autómata generado. Esta imagen se puede hacer en un papel a bolígrafo para posteriormente escanearla e introducirla como imagen en la memoria.

Ejercicios 2 y 3. Incluir en la memoria las explicaciones oportunas sobre el lenguaje elegido para describir el grafo del autómata y una gramática que reconozca tal lenguaje. Se incluirá un fichero de texto llamado `thP3D3.txt` con la descripción textual del grafo que resuelve el problema de las Torres de Hanoi con 3 palos y 3 discos. También se aportarán los ficheros `th.y` y `th.l` con el código en Bison y Flex implementando la gramática propuesta. El programa generado con Bison y Flex tomará como entrada:

- el fichero de texto `thP3D3.txt`;
- y dos nodos inicial y final predefinidos como variables del programa.

La salida del programa será la secuencia mínima de movimientos que se deberían realizar desde la posición descrita por el nodo inicial para acabar en la posición descrita por el nodo final (no es necesario, ni aconsejable, preguntar a un usuario por teclado o cualquier otro medio cuál es el nodo inicial y final). Se adjuntan con el material de la práctica las plantillas `th_plantilla.l` y `th_plantilla.y` como sugerencia de solución.

Procedimiento de entrega Elabora la memoria de la práctica y entrégala junto con los ficheros fuente según el Procedimiento de Entrega de Prácticas explicado en la Introducción a las Prácticas de la Asignatura. La fecha límite de entrega es el día **17 de diciembre de 2025** (incluido). Se recuerda especialmente lo siguiente:

- Verifica que todos tus ficheros fuente (*.l* y *.y*) contienen como comentario en sus primeras líneas los NIPs y nombres de los autores. Todos los programas deberán estar debidamente documentados.
- Verifica que los ficheros que vas a presentar compilan y ejecutan correctamente en hendrix.
- Crea un fichero comprimido .zip llamado

  `nip`Pr4.zip

  donde *nip* es el identificador personal de la persona que someta la práctica.
- Utiliza el enlace a una tarea de moodle disponible en la sección "Prácticas de Laboratorio" para entregar el fichero `nia`Pr4.zip

Bibliografía

[1] A.M. Hinz, S. Klavžar, and C. Petr. *The Tower of Hanoi – Myths and Maths*. Birkhäuser, Cham, 2nd edition, 2018.

[2] I. Stewart. *How to Cut a Cake: And Other Mathematical Conundrums*. Oxford University Press, Oxford, United Kingdom, 2006.