INSTITUTO POLITÉCNICO NACIONAL

Escuela Superior de Física y Matemáticas

Multiplicación Rápida de Matrices

Tesis que presenta

Yanira Pachuca Herrera*

Para obtener el Titulo de
Licenciado en Física y Matemáticas

Director de Tesis

Dr. Adrián Alcántar Torres*

México, D. F. 13 de diciembre de 2008

* Participante en los proyectos CONACyT 49251-F, SIP-IPN 20071670 y 20082743

Agradecimientos

En primer lugar, dedico este trabajo a mi Padre y Hermano Pedro Pachuca Pérez, a mi Madre y amiga Margarita Herrera Alemán, por su Amor a lo largo de toda mi vida.

Agradezco a Claudia Pachuca Herrera y a Miguel Angel Riezco Lagunes, por el apoyo al inicio de este trabajo.

Al Dr. Adrián Alcántar Torres, por haberme ofrecido el tema de esta tesis y por su paciencia durante el desarrollo de la misma.

A mis Profesores y Amigos: Antonio de Gante Ugarte, Emigdio Salazar Cordero, Pablo Lam Estrada, Abelardo Santaella Quintas, Guillermo Arreguin Sánchez, y a mi querida amiga Guillermina Regalado.

A la E.S.F.M y al I.P.N.

Prefacio

La multiplicación de matrices es una operación muy común en ciencias e ingeniería, sin embargo, su costo computacional es elevado; por lo que es necesario disponer de alternativas para la ejecución de esta tarea de manera eficiente disminuyendo el tiempo de procesamiento. El estudio de este trabajo es dar a conocer uno de los algoritmos que resuelve el problema de multiplicar matrices de dimensiones grandes, donde principalmente, se busca reducir el tiempo de cómputo empleado. Para lograr este objetivo se decidío dividir este trabajo en los siguientes capítulos:

1. En el capítulo 1 Recordaremos la definición de Matriz, la definición de un producto de matrices, así como sus propiedades y concluiremos con un ejemplo que nos permita convencernos sobre el hecho de que asociando adecuadamente las matrices a la hora de intentar calcular un producto de matrices, reduce el número de operaciones que se requiere para efectuar dicho producto, trayendo como consecuencia un *ahorro de tiempo* para obtener la respuesta.

2. Dedicaremos el capítulo 2 al estudio de la *función generatriz*. Comenzaremos con algunos ejemplos muy sencillos los cuales nos permitirán formarnos una idea sobre esta valiosisíma herramienta, después presentaremos su definición formal y finalmente la emplearemos para resolver algunos problemas.

3. Comenzaremos el capítulo 3 con la definición de Árbol, veremos paso a paso como se construyen y como estos objetos nos permitirán calcular el número de formas distintas en que se se puede parentizar un producto de matrices, y así poder concluir como al aumentar el número de matrices en un producto, el número de parentizaciones distintas crece dramáticamente.

4. Estudiaremos en el capítulo 4 la Notación Asintótica, veremos su aplicación como herramienta para medir la eficiencia de un algoritmo. Después enunciaremos el *Teorema Maestro* el cual es uno de los teoremas principales en el Análisis de Algoritmos y veremos mediante algunos ejemplos la forma tan contundente de saber de manera exacta la complejidad de un algoritmo al aplicar este teorema.

5. El capítulo 5 es considerado de gran importancia, ya que en el explicaremos a detalle el concepto de Programación dinámica y la forma en que emplearemos este método para resolver el problema de multiplicar una cadena de matrices encontrando la parentización óptima. Al final del capítulo se presenta el código del programa diseñado en base a las ideas centrales de la Programación dinámica

el cual al ser implementado en un computador devuelve la parentización óptima y el producto ya calculado.

6. Aquí veremos y analizaremos el Algoritmo de Strassen y lo aplicaremos a un ejemplo sencillo el cual nos permitirá deducir que este algoritmo solo conviene para calcular productos de matrices suficientemente grandes. Además veremos como la eficiencia de este algoritmo supera al algoritmo clásico para multiplicar matrices.

Índice general

Capítulo 1

Matrices

1.1. Vectores, Escalares, Matrices

1.1.1. Definiciones básicas

Vector renglón de n componentes: Definamos un *vector renglón de n componentes(o n-dimensional)* como un conjunto *ordenado* de n números escrito como

$$(x_1, x_2, \cdots, x_n) \tag{1.1}$$

Vector columna de n componentes: Un *vector columna de n componentes(o n-dimensional)* es un conjunto *ordenado* de n números escrito como

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \tag{1.2}$$

En (1.1) o (1.2) x_1 se llama la *primera componente* del vector, x_2 es la *segunda componente*, y así sucesivamente. En general, x_k es la *k-ésima componente* del vector.

Por simplicidad, nos referimos a un vector renglón *n-dimensional* como un *vector renglón* o un *n-vector*. De igual manera, usamos el término *vector columna* o n-vector para denotar a un vector *n-dimensional*. Cualquier vector con todas sus componentes iguales a cero se llama *vector cero*.

Ejemplo 1: Los siguientes son ejemplos de vectores:

i. $(3, 6)$ es un vector renglón con dos componentes.

ii. $\begin{pmatrix} 2 \\ -1 \\ 5 \end{pmatrix}$ es un vector columna con tres componentes.

Definición 1 *Una matriz* A *de tamaño* m x n *es un arreglo rectangular de* mn *números distribuidos en un orden de* m *renglones y* n *columnas:*

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2j} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{i1} & a_{i2} & \dots & a_{ij} & \dots & a_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mj} & \dots & a_{mn} \end{pmatrix}$$

El número a_{ij}, que aparece en el renglón i-ésimo y en la columna j-ésima de A, se conoce como la ij-ésima *componente* de A. Por conveniencia, la matriz A se escribe a veces A= (a_{ij}). Comunmente las matrices se denotan con letras mayúsculas.

Observación : Los vectores pueden ser vistos como matrices.

En el caso de vectores, los registros en el arreglo son llamados *coordenadas o componentes*. Los objetos que se usan como registros para matrices se llaman *escalares*. En este texto un *escalar* es un número real. Sin embargo, se acostumbra a usar otros *objetos* por escalares, por ejemplo, números racionales, números complejos, funciones.

1.1.2. Transpuesta de una matriz

Definición 2 Transpuesta. *Sea A*= (a_{ij}) *una matriz de tamaño m* x *n. Entonces la transpuesta de A, escrita* $A^t = (a^t_{ij})$*, donde* $(a^t_{ij}) = (a_{ji})$ *es la matriz de tamaño n* x *m.*

$$Si\ \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2j} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{i1} & a_{i2} & \dots & a_{ij} & \dots & a_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mj} & \dots & a_{mn} \end{pmatrix}$$

entonces,

$$\mathbf{A^t} = \begin{pmatrix} a_{11} & a_{21} & \dots & a_{i1} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{i2} & \dots & a_{m2} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{1j} & a_{2j} & \dots & a_{ij} & \dots & a_{mj} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{1n} & a_{2n} & \dots & a_{in} & \dots & a_{mn} \end{pmatrix}$$

1.1.3. Productos de vectores y matrices

Definición 3 Producto escalar de dos vectores. *Sean:*

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}^t$$

y

$$\mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}^t$$

dos n-vectores.

Entonces el *producto escalar* de $\mathbf{a}$ y $\mathbf{b}$, representado por $\mathbf{a} \cdot \mathbf{b}$ está dado por

$$\boxed{\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n} \tag{1.3}$$

Por la notación en (1.3), el producto escalar de dos vectores frecuentemente es conocido como *producto punto o interno* de los vectores.

Observación. Cuando hacemos el producto escalar de $\mathbf{a}$ y $\mathbf{b}$ necesitamos que $\mathbf{a}$ y

b tengan el mismo número de componentes.

Frecuentemente efectuamos el producto escalar de un vector renglón y un vector columna. En este caso tenemos

$$\begin{pmatrix} a_1 & a_2 & \cdots & a_n \end{pmatrix} . \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}^t = a_1b_1 + a_2b_2 +, \cdots, +a_nb_n$$

Ejemplo 2: Sean

$$\mathbf{a} = \begin{pmatrix} 2 & -3 & 4 & -6 \end{pmatrix}$$

y

$$\mathbf{b} = \begin{pmatrix} 1 \\ 2 \\ 0 \\ 3 \end{pmatrix}$$

Calcule a $\cdot\ b^t$.

Solución: a $\cdot\ b^t = (2)(1) + (-3)(2) + (4)(0) + (-6)(3) = -22$

Observación. A continuación presentaremos la definición de producto de matrices pero antes es importante comentar lo siguiente: de manera natural, podríamos definir el producto de dos matrices A= (a_{ij}) y B= (b_{ij}) de tamaño m x n y n x p respectivamente, como la matriz AB=C de tamaño m x p cuya *ij-ésima* componente es $a_{ij}b_{ij}$. Sin embargo, debido a las importantes aplicaciones de las matrices, el producto de matrices se define de manera diferente, esto se debe a que cuando las matrices son vistas como transformaciones lineales, el producto de matrices debe coincidir con la composición de transformaciones.

Definición 4 Producto de dos matrices. *Sea A= (a_{ij}) una matriz de tamaño m x n cuyo i-ésimo renglón denotamos por a_i. Sea B= (b_{ij}) una matriz de tamaño n x p cuya j-ésima columna denotamos por b_j. Entonces el producto de A y B es una matriz C= (c_{ij}) de tamaño m x p, donde*

$$\boxed{c_{ij} = \mathbf{a}_i \cdot \mathbf{b}_j} \tag{1.4}$$

Esto es, el ij-ésimo elemento de AB es el producto del i-ésimo renglón de $A(a_i)$ y la j-ésima columna de $B(b_j)$. Si desarrollamos esto obtenemos:

$$\boxed{c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + ... + a_{in}b_{nj}} \tag{1.5}$$

Observación. Es importante mencionar que dos matrices pueden multiplicarse sólo si el número de columnas de la primera es igual al número de renglones de la segunda. De otra forma, los vectores $\mathbf{a}_i$ y $\mathbf{b}_j$ tendrían diferente número de componentes y el producto de la ecuación anterior no estaría definido.

Ejemplo 3:

Sean:

$$\mathbf{A} = \begin{pmatrix} 2 & 0 & -3 \\ 4 & 1 & 5 \end{pmatrix}$$

y

$$\mathbf{B} = \begin{pmatrix} 7 & -1 & 4 & 7 \\ 2 & 5 & 0 & -4 \\ -3 & 1 & 2 & 3 \end{pmatrix}$$

Calculemos **AB**.

Primero notemos que A es una matriz de tamaño 2 x 3 y B es una matriz de tamaño 3 x 4. Por tanto el número de columnas de A es igual número de renglones de B. El producto de AB, por consiguiente esta definido y es una matriz de tamaño 2 x 4. Sea $AB{=}C{=}\,(c_{ij})$. Entonces:

$$\mathbf{c_{11}} = \begin{pmatrix} 2 & 0 & -3 \end{pmatrix} . \begin{pmatrix} 7 \\ 2 \\ -3 \end{pmatrix}^t = 23$$

$$\mathbf{c_{12}} = \begin{pmatrix} 2 & 0 & -3 \end{pmatrix} . \begin{pmatrix} -1 \\ 5 \\ 1 \end{pmatrix}^t = -5$$

$$\mathbf{c_{13}} = \begin{pmatrix} 2 & 0 & -3 \end{pmatrix} . \begin{pmatrix} 4 \\ 0 \\ 2 \end{pmatrix}^t = 2$$

$$\mathbf{c_{14}} = \begin{pmatrix} 2 & 0 & -3 \end{pmatrix} . \begin{pmatrix} 7 \\ -4 \\ 3 \end{pmatrix}^t = 5$$

$$\mathbf{c_{21}} = \begin{pmatrix} 4 & 1 & 5 \end{pmatrix} . \begin{pmatrix} 7 \\ 2 \\ -3 \end{pmatrix}^t = 15$$

$$\mathbf{c_{22}} = \begin{pmatrix} 4 & 1 & 5 \end{pmatrix} . \begin{pmatrix} -1 \\ 5 \\ 1 \end{pmatrix}^t = 6$$

$$\mathbf{c_{23}} = \begin{pmatrix} 4 & 1 & 5 \end{pmatrix} . \begin{pmatrix} 4 \\ 0 \\ 2 \end{pmatrix}^t = 26$$

$$\mathbf{c_{24}} = \begin{pmatrix} 4 & 1 & 5 \end{pmatrix} . \begin{pmatrix} 7 \\ -4 \\ 3 \end{pmatrix}^t = 39$$

Por tanto:

$$\mathbf{AB} = \begin{pmatrix} 23 & -5 & 2 & 5 \\ 15 & 6 & 26 & 39 \end{pmatrix}$$

Esto termina el ejemplo.

Puesto que ya hemos presentado al lector el concepto de multiplicación de dos matrices, ahora nuestro siguiente objetivo será contar el número total de operaciones (como sumas y multiplicaciones) implicadas en este cálculo.

Más adelante, una vez que recordemos la *ley asociativa para el producto de matrices* el saber cuantas sumas y multipicaciones se requiere para calcular cada asociación nos llevará a conclusiones realmente sorprendentes.

Recordemos nuevamente como se multiplican matrices:

Sea $\mathbf{A}(a_{ij})$ de tamaño m x n cuyo i-ésimo renglón denotamos por a_i y $\mathbf{B}= (b_{ij})$ una matriz de tamaño n x p cuya j-ésima columna denotamos por b_j, el producto de $\mathbf{A}$ y $\mathbf{B}$ es una matriz $\mathbf{C}= (c_{ij})$ de tamaño m x p, donde

$$\boxed{c_{ij} = a_i \cdot b_j = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj}} \tag{1.6}$$

Esto es, el ij-ésimo elemento de **AB** es el producto escalar del i-ésimo renglón de $\mathbf{A}(a_i)$ y la j-ésima columna de $\mathbf{B}(b_j)$.

Observemos que en la ecuación (1.6) el número de multiplicaciones involucradas en el producto de $a_i \cdot b_j$ es igual al número de componentes de a_i que por definición es igual al número de componentes de b_j para que $a_i \cdot b_j$ tenga sentido. Puesto que cada a_i se puede ver como un n-vector, así el número de multiplicaciones involucradas en $a_i \cdot b_j$ son n y el número de sumas involucradas en este cálculo son $n - 1$.

Un hecho clave es que el tamaño de los vectores involucrados en el cálculo de cada entrada c_{ij} de la matriz $\mathbf{C}= (c_{ij})$ es siempre el mismo, haciendo que el número de multiplicaciones y sumas que se efectuen para obtener cada una de las entradas sea idéntico. Esto aunado al hecho de que nuestra matriz resultante $\mathbf{C}= (c_{ij})$ es de tamaño m x p (es decir se calculan m x p entradas) nos permite concluir que para obtener la matriz $\mathbf{C}= (c_{ij})$ el número total de multiplicaciones y sumas que se requieren son, *(n)(mp)* y $(n - 1)$*(mp)* respectivamente.

Ejemplo 4:

Sea:

$$\mathbf{A} = \begin{pmatrix} -4 & 5 & 1 \\ 0 & 4 & 2 \end{pmatrix}$$

de tamaño 2 x 3

$$\mathbf{B} = \begin{pmatrix} 3 & -1 & 1 \\ 5 & 6 & 4 \\ 0 & 1 & 2 \end{pmatrix}$$

de tamaño 3 x 3

La matriz resultante de multiplicar **AB** es una matriz **C** de tamaño 2 x 3. Observemos que el tamaño de los vectores implicados en el cálculo de cada entrada de la matriz **C** es 3. Por lo tanto en total se tendran que calcular 3 x 2 x 3 = 18 multiplicaciones y 2 x 2 x 3 = 12 sumas en total.

Otra forma de ver esto es **AB**=**C**= (c_{ij}) de 2 x 3 donde:

$$\mathbf{c_{11}} = a_1 \cdot b_1 = \begin{pmatrix} -4 & 5 & 1 \end{pmatrix} . \begin{pmatrix} 3 \\ 5 \\ 0 \end{pmatrix}^t = (-4)(3) + (5)(5) + (1)(0) = 13$$

Total de operaciones: 3 multiplicaciones y 2 sumas.

$$\mathbf{c_{12}} = a_1 \cdot b_2 = \begin{pmatrix} -4 & 5 & 1 \end{pmatrix} . \begin{pmatrix} -1 \\ 6 \\ 1 \end{pmatrix}^t = (-4)(-1) + (5)(6) + (1)(1) = 35$$

Total de operaciones: 3 multiplicaciones y 2 sumas.

$$\mathbf{c_{13}} = a_1 \cdot b_3 = \begin{pmatrix} -4 & 5 & 1 \end{pmatrix} . \begin{pmatrix} 1 \\ 4 \\ 2 \end{pmatrix}^t = (-4)(1) + (5)(4) + (1)(2) = 18$$

Total de operaciones: 3 multiplicaciones y 2 sumas.

$$\mathbf{c_{21}} = a_2 \cdot b_1 = \begin{pmatrix} 0 & 4 & 2 \end{pmatrix} . \begin{pmatrix} 3 \\ 5 \\ 0 \end{pmatrix}^t = (0)(3) + (4)(5) + (2)(0) = 20$$

Total de operaciones: 3 multiplicaciones y 2 sumas.

$$\mathbf{c_{22}} = a_2 \cdot b_2 = \begin{pmatrix} 0 & 4 & 2 \end{pmatrix} . \begin{pmatrix} -1 \\ 6 \\ 1 \end{pmatrix}^t = (0)(-1) + (4)(6) + (2)(1) = 26$$

Total de operaciones: 3 multiplicaciones y 2 sumas.

$$\mathbf{c_{23}} = a_2 \cdot b_3 = \begin{pmatrix} 0 & 4 & 2 \end{pmatrix} . \begin{pmatrix} 1 \\ 4 \\ 2 \end{pmatrix}^t = (0)(1) + (4)(4) + (2)(2) = 20$$

Total de operaciones: 3 multiplicaciones y 2 sumas.

Luego el total de operaciones involucradas en el producto de **AB** son: 18 multiplicaciones y 12 sumas.

1.1.4. Ley asociativa para la multiplicación de matrices

Sea $A = (a_{ij})$ una matriz de tamaño n x m, $B = (b_{ij})$ una matriz de tamaño m x p y $C = (c_{ij})$ de tamaño p x q. Entonces la *ley asociativa*

$$\boxed{A(BC) = (AB)C} \tag{1.7}$$

es válida y ABC, definida por cualquier lado de (1.7) es una matriz de tamaño n x q.

Ejemplo 5:

Verifique la ley asociativa para:

$$\mathbf{A} = \begin{pmatrix} 1 & -3 \\ 0 & 2 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} 2 & -1 & 4 \\ 3 & 1 & 5 \end{pmatrix}$$

y

$$\mathbf{C} = \begin{pmatrix} 0 & -2 & 1 \\ 4 & 3 & 2 \\ -5 & 0 & 6 \end{pmatrix}$$

Primero notemos que A es de tamaño 2 x 2, B es de tamaño 2 x 3 y C es de tamaño 3 x 3. Así, todos los productos empleados en el enunciado de la ley asociativa estan definidos y el producto resultante será una matriz de tamaño 2 x 3. Entonces calculamos:

$$\mathbf{AB} = \begin{pmatrix} 1 & -3 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} 2 & -1 & 4 \\ 3 & 1 & 5 \end{pmatrix} = \begin{pmatrix} -7 & -4 & -11 \\ 6 & 2 & 10 \end{pmatrix}$$

$$(\mathbf{AB})\mathbf{C} = \begin{pmatrix} -7 & -4 & -11 \\ 6 & 2 & 10 \end{pmatrix} \begin{pmatrix} 0 & -2 & 1 \\ 4 & 3 & 2 \\ -5 & 0 & 6 \end{pmatrix} = \begin{pmatrix} 39 & 2 & -81 \\ -42 & -6 & 70 \end{pmatrix}$$

Igualmente,

$$\mathbf{BC} = \begin{pmatrix} 2 & -1 & 4 \\ 3 & 1 & 5 \end{pmatrix} \begin{pmatrix} 0 & -2 & 1 \\ 4 & 3 & 2 \\ -5 & 0 & 6 \end{pmatrix} = \begin{pmatrix} -24 & -7 & 24 \\ -21 & -3 & 35 \end{pmatrix}$$

$$\mathbf{A(BC)} = \begin{pmatrix} 1 & -3 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} -24 & -7 & 24 \\ -21 & -3 & 35 \end{pmatrix} = \begin{pmatrix} 39 & 2 & -81 \\ -42 & -6 & 70 \end{pmatrix}$$

Asi, $(AB)C = A(BC)$

La ley asociativa se puede extender a productos más grandes. Por ejemplo, si AB, BC y CD estan definidos, entonces

$$\boxed{ABCD=(A(B(CD)))=(A((BC)D))=(((AB)C)D)= ((A(BC))D) = ((AB)(CD))} \quad (1.8)$$

De aquí en adelante a cualquiera de estas asociaciones la denotaremos con $ABCD$ por (1.8); ya que obtenemos el mismo resultado sin importar cómo se efectúe la multiplicación. Sin embargo, veremos a continuación mediante un ejemplo muy sencillo como **el tiempo que se emplea para calcular el producto de matrices** está en función de la manera en que asociemos dichas matrices, lo cual puede generar un gran impacto puesto que el número de calculos puede variar sorprendentemente de una asociación a otra.

Ejemplo 6: Consideremos el problema de multiplicar tres matrices $A_1A_2A_3$. Supongamos que el tamaño de las matrices son 10 x 10,000, 10,000 x 5, y 5 x 5,000, respectivamente.

Si nosotros multiplicamos de acuerdo a la asociación $((A_1A_2)A_3)$, para calcular el producto(A_1A_2) tendríamos que calcular $(10,000)(10)(5) = 500,000$, multiplicaciones y obtendríamos una matriz resultante de tamaño 10 x 5. Al multiplicar esta matriz resultante de tamaño 10 x 5 con la matriz A_3 de tamaño 5 x 5,000 se calcularían $(5)(10)(5,000) = 250,000$ multiplicaciones para darnos un total de $500,000 + 250,000 =$**750,000** multiplicaciones por efectuarse.

Ahora de acuerdo a la asociación $(A_1(A_2A_3))$, el número de multiplicaciones por calcularse serían primero $(5)(10,000)(5,000) = 250,000,000$, para calcular el producto (A_2A_3) obteniendose una matriz de tamaño 10,000 x 5,000, y finalmente para multiplicar esta matriz de tamaño 10,000 x 5,000 por la matriz A_1 matriz de tamaño 10 x 10,000 se requerirán $(10,000)(10)(5,000) = 500,000,000$ multiplicaciones. Por lo tanto para calcular el producto $(A_1(A_2A_3))$ se requerirán un total de $250,000,000 + 500,000,000 =$**750,000,000** multiplicaciones por calcularse.

Así, calculando el producto de acuerdo a la primera asociación esta es **1,000 veces más rápida de calcularse**.

La conclusión que de aquí se desprende sobre el tiempo que se emplea en calcular el producto de matrices será nuestro punto de partida para comenzar nuestro estudio

sobre la multiplicación de matrices y la importancia de optimizar la forma en que se calcula dicho producto.

Antes de concluir este capítulo es importante mencionar que si nosotros aumentamos el número de matrices en un producto, el número de maneras en que este producto se puede asociar aumentará dramáticamente y encontrar la asociación que requiera el mínimo de operaciones para calcular dicho producto ya no será tan fácil. Para convencernos de este hecho necesitamos primero conocer y estudiar ciertas herramientas que nos permitirán concluir lo antes citado.

Así pues comencemos con su estudio.

Capítulo 2

Función Generatriz

Con el último ejemplo visto en el capítulo anterior se pudo concluir que el tiempo que se requiere para calcular el producto de matrices depende de la manera en que asociemos dichas matrices. Y mencionamos que al aumentar el número de matrices implicadas en un producto, el número de asociaciones también aumentará de manera considerable. Ahora nuestra tarea es convencernos sobre este hecho.

Por lo tanto comenzemos respondiendo la siguiente pregunta: dado un producto de varias matrices ¿De cuántas formas distintas puedo asociar dicho producto?, lo cual nos conduce a resolver un problema de conteo. Es importante señalar que la pregunta anterior no podrá ser respondida en este capítulo ya que aún no contamos con las herramientas necesarias para hacerlo, sin embargo estudiaremos en este capítulo una herramienta conocida como *función generatriz* la cual es utilizada precisamente para resolver problemas de este tipo.

En lugar de definir en este punto una función generatriz, examinaremos algunos ejemplos para derivar la idea de función generatriz a partir de ellos y después nos extenderemos a la definición formal.

Ejemplo 1: En sus compras del sábado, Mónica compró 12 naranjas para sus hijos, Lidia, Wendy e Ivan. ¿De cuántas maneras puede ella distribuir las naranjas de tal forma que Lidia obtenga al menos cuatro, Wendy e Ivan obtengan al menos dos, sin que Ivan obtenga más de cinco? La Cuadro (2.1) enumera todas las distribuciones posibles. Vemos que tenemos todas las soluciones enteras de la ecuacion $c_1 + c_2 + c_3 = 12$ tal que $4 \leq c_1$, $2 \leq c_2$, y $2 \leq c_3 \leq 5$.

Parecería rápido enumerar los casos de el Cuadro (2.1), pero al pasar a problemas con más incógnitas y mayores cantidades que distribuir este método se complica, por lo tanto resolveremos este mismo problema de otra forma, lo haremos utilizando una observación clave en la cual fundamentaremos una manera más rápida de resolver este problema y otros problemas con mayor número de incógnitas y cantidades.

La observación clave es la siguiente: nosotros sabemos que al multiplicar polinomios

Cuadro 2.1: Las 14 distribuciones posibles

L	W	I	L	W	I
4	3	5	6	2	4
4	4	4	6	3	3
4	5	3	6	4	2
4	6	2	7	2	3
5	2	5	7	3	2
5	3	4	8	2	2
5	4	3			
5	5	2			

sumamos las potencias de la variable, usando este hecho multipliquemos los siguientes tres polinomios,

$$(x^4 + x^5 + x^6 + x^7 + x^8)(x^2 + x^3 + x^4 + x^5 + x^6)(x^2 + x^3 + x^4 + x^5).$$

Si observamos este producto con más cuidado, nos daremos cuenta de que obtenemos el producto $x^i x^j x^k$ para toda terna (i,j,k) que aparece en la tabla anterior.

Veamos con más detalle que sucede al multiplicar paso a paso los polinomios pues las observaciones que se deriven de ello serán muy importantes.

Multipliquemos $(x^4 + x^5 + x^6 + x^7 + x^8)(x^2 + x^3 + x^4 + x^5 + x^6)$

$$\begin{array}{l}
x^4 + x^5 + x^6 + x^7 + x^8 \\
\underline{x^2 + x^3 + x^4 + x^5 + x^6} \\
x^{2+4} + x^{2+5} + x^{2+6} + x^{2+7} + x^{2+8} \\
\qquad x^{3+4} + x^{3+5} + x^{3+6} + x^{3+7} + x^{3+8} \\
\qquad\qquad x^{4+4} + x^{4+5} + x^{4+6} + x^{4+7} + x^{4+8} \\
\qquad\qquad\qquad x^{5+4} + x^{5+5} + x^{5+6} + x^{5+7} + x^{5+8} \\
\qquad\qquad\qquad\qquad \underline{x^{6+4} + x^{6+5} + x^{6+6} + x^{6+7} + x^{6+8}}
\end{array}$$

Al hacer este producto hemos visto a detalle como se suman las potencias. Nos resta multiplicar cada una de estas variables por el polinomio $(x^2 + x^3 + x^4 + x^5)$ pero hacerlo y presentarlo como al producto anterior sería complicado, por lo tanto en el Cuadro (2.2) solo mostraremos aquellas variables resultantes del producto de $(x^4 + x^5 + x^6 + x^7 + x^8)(x^2 + x^3 + x^4 + x^5 + x^6)$ las cuales al ser multiplicadas por (x^2, x^3, x^4, x^5) respectivamente dan como resultado variables cuya potencia es 12, las cuales suman un total de 14 posibilidades.

Observemos que el total de posibilidades presentados en el cuadro anterior es precisamente el coeficiente de la variable cuya potencia es 12 que aparece en el producto de:

Cuadro 2.2: Los 14 productos cuya potencia es 12

x^2	x^3	x^4	x^5
$x^{2+8}x^2$	$x^{2+7}x^3$	$x^{2+6}x^4$	$x^{2+5}x^5$
$x^{3+7}x^2$	$x^{3+6}x^3$	$x^{3+5}x^4$	$x^{3+4}x^5$
$x^{4+6}x^2$	$x^{4+5}x^3$	$x^{4+4}x^4$	
$x^{5+5}x^2$	$x^{5+4}x^3$		
$x^{6+4}x^2$			

$(x^4 + x^5 + x^6 + x^7 + x^8)(x^2 + x^3 + x^4 + x^5 + x^6)(x^2 + x^3 + x^4 + x^5) =$
$= x^8 + 3x^9 + 6x^{10} + 10x^{11} + 14x^{12} + 16x^{13} + 16x^{14} + 14x^{15} + 10x^{16} + 6x^{17} + 3x^{18} + x^{19}.$

En consecuencia, el coeficiente de x^{12} en

$$f(x) = (x^4 + x^5 + x^6 + x^7 + x^8)(x^2 + x^3 + x^4 + x^5 + x^6)(x^2 + x^3 + x^4 + x^5)$$

cuenta el número buscado de formas de distribuir las 12 naranjas, a saber 14.

Pero, ¿de dónde salieron los factores de este producto?

Por ejemplo: La condición sobre Lidia indica que puede recibir $4, 5, 6, 7$ *u* 8 naranjas, es decir comenzamos con 4 naranjas y nos detenemos en 8 puesto que Wendy e Ivan deben recibir cada uno al menos dos. Así se obtiene el factor $(x^4+x^5+x^6+x^7+x^8)$.

En el caso de Ivan, debe recibir al menos dos naranjas asi comenzando con 2 nos detenemos en cinco para no recibir más de cinco naranjas es decir, Ivan puede recibir $2, 3, 4$ *o* 5. Así, se obtiene el término $(x^2 + x^3 + x^4 + x^5)$.

Como Wendy debe recibir al menos dos naranjas esto indica que podrá recibir $2, 3, 4, 5$ *o* 6, ¿porqué detenemos en 6 el número de naranjas que puede recibir Wendy?. Supongamos que el número de naranjas que puede recibir Wendy es 6, más dos naranjas que es la condición mínima sobre Ivan se obtienen 8 naranjas y solo faltarían 4 naranjas para completar las doce, pero estas 4 naranjas son la condición mínima sobre Lidia cumpliéndose así la ecuación $c_1 + c_2 + c_3 = 12$ tal que $4 \leq c_1$, $2 \leq c_2$, y $2 \leq c_3 \leq 5$.

De otro modo, supongamos que Wendy recibe 7 o más naranjas, más dos naranjas que es nuevamente la condición mínima sobre Ivan se tendrían 9 naranjas y faltarían 3 para completar las doce naranjas, pero c_1 en la ecuación $c_1 + c_2 + c_3 = 12$ solo puede tomar los valores de $4, 5, 6, 7$ *u* 8.

Por lo tanto el factor que representa la condición sobre Wendy queda expresado como $(x^2 + x^3 + x^4 + x^5)$.

La mayoría de nosotros nos hemos convencido razonablemente de que el coeficiente de x^{12} en *f(x)* nos da la respuesta sobre el número de formas en que Mónica puede repartir las 12 naranjas. Sin embargo, algunos serán un poco escépticos acerca de esta nueva idea. Por ahora examinemos dos ejemplos más.

Ejemplo 2: Si existe un número ilimitado (o al menos 24 de cada color) de dulces de jalea de color rojo, verde, blanco y negro, ¿de cuántas formas puede seleccionar un niño 24 de estos dulces de tal manera que tenga un número par de dulces blancos y al menos seis dulces negros?

Los polinomios asociados con los colores de los dulces de jalea son los siguientes:

- rojo ó verde: $1 + x + x^2 + ... + x^{24}$, donde el 1 representa al x^0, ya que una posibilidad es que ninguno de los colores rojo(y verde) sea seleccionado. Así este polinomio representa el número de posibilidades para que el color rojo (y verde) sean seleccionados.

- blanco: $(1 + x^2 + x^4 + x^6 + ... + x^{24})$, donde el 1 representa al x^0, (la posibilidad de que el color blanco no sea seleccionado), además el 0 es considerado un número par.

- negro: $(x^6 + x^7 + x^8 + ... + x^{24})$ puesto que se tiene la condición de que, de los 24 dulces se tengan al menos seis dulces negros.

Así, la respuesta es el coeficiente de x^{24} en la función:

$$f(x) = (1 + x + x^2 + ... + x^{24})(1 + x^2 + x^4 + x^6 + ... + x^{24})(x^6 + x^7 + x^8 + ... + x^{24})$$

Ejemplo 3: ¿ Cuántas soluciones enteras tiene la ecuacion $c_1 + c_2 + c_3 + c_4 = 25$ si $0 \leq c_i$ para todo $1 \leq i \leq 4$?

Otra forma es preguntar ¿de cuántas formas se pueden distribuir 25 monedas de un centavo (idénticas) entre cuatro niños?

Podemos describir las posibilidades para cada niño mediante el polinomio $1 + x + x^2 + x^3 + ... + x^{25}$. Entonces la respuesta a este problema es el coeficiente de x^{25} en la función *f(x)*$= (1 + x + x^2 + ... + x^{25})^4$.

Estos ejemplos son suficientes para mostrar la primera definición de *función generatriz*. Esta definición fue dada a conocer por Pierre Simon de **Laplace** $(1749 - 1827)$

en su **Memoria de las series**(1782).

Definición 5 *FUNCIÓN GENERATRIZ (P.S.Laplace, 1782): " Si se imagina una función A de una variable, desarrollada en una serie ascendente respecto a las potencias de dicha variable, el coeficiente de una cualquiera de estas potencias será una función del índice o exponente de la misma. Es A a lo que yo llamo* ***función generatriz*** *de dicho coeficiente o de la función del índice."*

Por lo tanto regresando a nuestros ejemplos anteriores para el ejemplo 1, la *función generatriz* es

$$f(x) = (x^4 + x^5 + x^6 + x^7 + x^8)(x^2 + x^3 + x^4 + x^5 + x^6)(x^2 + x^3 + x^4 + x^5)$$

$$= x^8 + 3x^9 + 6x^{10} + 10x^{11} + 14x^{12} + 16x^{13} + 16x^{14} + 14x^{15} + 10x^{16} + 6x^{17} + 3x^{18} + x^{19}$$

Para el ejemplo 2, la *función generatriz* es

$$f(x) = (1 + x + x^2 + ... + x^{24})(1 + x^2 + x^4 + x^6 + ... + x^{24})(x^6 + x^7 + x^8 + ... + x^{24})$$

Y finalmente para el ejemplo 3, la *función generatriz* es

$$f(x) = (1 + x + x^2 + ... + x^{25})^4$$

Observemos, para este último ejemplo que también podemos obtener la respuesta a la pregunta ¿de cuántas formas se pueden distribuir 25 monedas de un centavo (idénticas) entre cuatro niños? mediante el coeficiente x^{25} en la función generatriz $g(x)$= $(1 + x + x^2 + x^3 + ... + x^{25} + x^{26} + ...)^4$, ya que nunca se usan los términos x^k, para toda $k \geq 26$ para obtener la respuesta.[Observemos que mientras que $f(x)$ es un polinomio $g(x)$ es una *serie de potencias* en x].

Pero ¿como podemos convencernos de esto?. En lugar de resolver este problema que es muy complicado debido al número de variables, examinemos un ejemplo más sencillo para derivar la idea a partir de él.

Ahora el problema quedará expresado en términos de una distribución de 4 monedas entre cuatro niños. Y su solución será el coeficiente de x^4 en la función generatriz $f(x)$= $(1 + x + x^2 + x^3 + x^4)^4$.

Así pues, desarrollemos

$$(1 + x + x^2 + x^3 + x^4)^4$$

$(1 + x + x^2 + x^3 + x^4)^4 = (1 + x + x^2 + x^3 + x^4)(1 + x + x^2 + x^3 + x^4)(1 + x + x^2 + x^3 + x^4)(1 + x + x^2 + x^3 + x^4) = 1 + 4x + 10x^2 + 20x^3 + 35x^4 + 52x^5 + 68x^6 + 80x^7 + 85x^8 + 80x^9 + 68x^{10} + 52x^{11} + 35x^{12} + 20x^{13} + 10x^{14} + 4x^{15} + x^{16}$.

Si buscamos el coeficiente de x^4, entonces la respuesta es 35.

De igual forma si nosotros desarrollamos $(1+x+x^2+x^3+x^4+x^5+...)^4$ obtendríamos el mismo resultado, es decir el coeficiente de x^4 en la función generatriz *g(x)*=$(1+x+x^2+x^3+x^4+x^5+...)^4$, el cual es también 35. Las variables x^k, para toda $k \geq 5$ y sus coeficientes nunca se usan para obtener la respuesta si expresamos la pregunta en términos de distribucione de 4 monedas entre 4 niños, por lo tanto si estas variables no se usan entonces por que nos preocupamos por ellos?. La respuesta es: porque habrá veces que será más fácil contar con una serie de potencias que con un polinomio.

Es necesario darse cuenta que hasta aquí, la definición que hemos dado de *función generatriz* nos ha ayudado a resolver problemas muy sencillos de conteo, sin embargo para los fines de este trabajo necesitamos otra definición de función generatriz. La definición que daremos a continuación tiene que ver precisamente con series de potencias y solo es una versión de la definición principal, la cual fue modificada adecuadamente para resolver otro tipo de problemas más complejos.

Definición 6 *Sea $a_0, a_1, a_2...$ una sucesión de números reales. La función $f(x) = a_0 + a_1x + a_2x^2 + ... = \sum_{i=0}^{\infty} a_ix^i$ es la función generatriz de la sucesión dada.*

A continuación examinaremos algunos ejemplos relacionados con las series de potencias que usaremos para obtener los coeficientes de términos particulares de una función generatriz.

Ejemplo 1: Para cualquier $n \in \mathbb{Z}^+$

$$(1+x)^n = \binom{n}{0} + \binom{n}{1}x + \binom{n}{2}x^2 + \cdots + \binom{n}{n}x^n,$$

de modo que $(1+x)^n$ es la *función generatriz* para la sucesión

$$\binom{n}{0}, \binom{n}{1}, \binom{n}{2}, \cdots, \binom{n}{n}, 0, 0, 0, \cdots$$

Ejemplo 2:

a) Para $n \in \mathbb{Z}^+$

$$(1-x^{n+1}) = (1-x)(1+x+x^2+x^3+\cdots+x^n)$$

Por lo tanto,

$$\frac{(1-x^{n+1})}{(1-x)} = (1+x+x^2+x^3+\cdots+x^n)$$

y

$$\frac{(1-x^{n+1})}{(1-x)}$$

es la *función generatriz* para la sucesión $1,\ 1,\ 1, \cdots 1,\ 0,\ 0,\ 0, \cdots$,

donde los primeros $n+1$ términos son 1.

b) Con

$$\frac{1}{(1-x)} = 1 + x + x^2 + x^3 + x^4 + ... = \sum_{i=0}^{\infty} x^i,$$

tomando la derivada de cada lado obtenemos

$$\begin{aligned}
\frac{d}{dx}\frac{1}{(1-x)} &= \frac{d}{dx}(1-x)^{-1} \\
&= (-1)(1-x)^{-2}(-1) \\
&= \frac{1}{(1-x)^2} \\
&= \frac{d}{dx}(1 + x + x^2 + x^3 + x^4 + ...) \\
&= 1 + 2x + 3x^2 + 4x^3 + ...
\end{aligned}$$

En consecuencia,

$$\frac{1}{(1-x)^2}$$

es *la función generatriz* para la sucesión $1,\ 2,\ 3,\ 4, \cdots$

Observación: Para todo $n \in \mathbb{Z}^+$, el Teorema del binomio señala que

$$(1+x)^n = \binom{n}{0} + \binom{n}{1}x + \binom{n}{2}x^2 + \cdots + \binom{n}{n}x^n$$

Queremos extender esta idea a los casos en que $(a) n < 0$ y (b) n no sea necesariamente un entero.

Si n,$r \in \mathbb{Z}^+$ y $n \geq r > 0$, tenemos

$$\begin{aligned}
\binom{n}{r} &= \frac{n!}{r!(n-r)!} \\
&= \frac{n(n-1)(n-2)\cdots(n-r+1)}{r!}
\end{aligned}$$

Si $n \in \mathbb{R}$, definimos

$$\binom{n}{r}$$

como

$$\frac{n(n-1)(n-2)\cdots(n-r+1)}{r!}$$

Entonces, por ejemplo, si $n \in \mathbb{Z}^+$, tenemos

$$\begin{aligned}\binom{-n}{r} &= \frac{(-n)(-n-1)(-n-2)\cdots(-n-r+1)}{r!} \\ &= \frac{(-1)^r n(n+1)(n+2)\cdots(n+r-1)}{r!} \\ &= \frac{(-1)^r(n+r-1)!}{(n-1)!r!} \\ &= (-1)^r\binom{n+r-1}{r}\end{aligned}$$

Finalmente, para cualquier número *real* n, definimos

$$\binom{n}{0} = 1.$$

Ejemplo 3:

Para n,$r \in \mathbb{Z}^+$, el desarrollo en serie de Maclaurin para $(1+x)^{-n}$ está dada por

$$\begin{aligned}(1+x)^{-n} &= 1 + (-n)x + \frac{(-n)(-n-1)x^2}{2!} + \frac{(-n)(-n-1)(-n-2)x^3}{3!} + ... \\ &= 1 + \sum_{r=1}^{\infty} \frac{(-n)(-n-1)(-n-2)...(-n-r+1)}{r!}x^r \\ &= \sum_{r=0}^{\infty}(-1)^r\binom{n+r-1}{r}x^r\end{aligned}$$

Por lo tanto,

$$\begin{aligned}(1+x)^{-n} &= \binom{-n}{0} + \binom{-n}{1}x + \binom{-n}{2}x^2 + \cdots \\ &= \sum_{r=0}^{\infty}\binom{-n}{r}x^r\end{aligned}$$

Esto generaliza el teorema del binomio y muestra que es $(1+x)^{-n}$ la *función generatriz* de la sucesión

$$\binom{-n}{0}, \binom{-n}{1}, \binom{-n}{2}, \binom{-n}{3}, \cdots$$

Ejemplo 4:

Encuentre el coeficiente de x^5 en $(1-2x)^{-7}$. Con $y = -2x$, usando el resultado del ejemplo anterior para escribir

$$\begin{aligned}(1-2x)^{-7} &= (1+y)^{-7} \\ &= \sum_{r=0}^{\infty} \binom{-7}{r} y^r \\ &= \sum_{r=0}^{\infty} \binom{-7}{r} (-2x)^r\end{aligned}$$

En consecuencia, el coeficiente de x^5 es

$$\begin{aligned}\binom{-7}{5}(-2)^5 &= (-1)^5 \binom{7+5-1}{5}(-32) \\ &= 32\binom{11}{5} \\ &= 32(462) \\ &= 14,784\end{aligned}$$

Ejemplo 5:

Para cualquier número real n, el desarrollo en serie de Maclaurin para $(1+x)^n$ es

$$\begin{aligned}(1+x)^n &= 1 + nx + \frac{n(n-1)x^2}{2!} + \frac{n(n-1)(n-2)x^3}{3!} + \cdots \\ &= 1 + \sum_{r=1}^{\infty} \frac{n(n-1)(n-2)+\cdots(n-r+1)}{r!} x^r\end{aligned}$$

Por lo tanto,

$$\begin{aligned}(1+3x)^{-\frac{1}{3}} &= 1 + \sum_{r=1}^{\infty} \frac{(-\frac{1}{3})(-\frac{4}{3})(-\frac{7}{3})\cdots(\frac{-3r+2}{3})}{r!}(3x)^r \\ &= 1 + \sum_{r=1}^{\infty} \frac{(-1)(-4)(-7)\cdots(-3r+2)}{r!} x^r\end{aligned}$$

y

$$(1+3x)^{-\frac{1}{3}}$$

genera la sucesión

$$1,\ -1,\ \frac{(-1)(-4)}{2!},\ \frac{(-1)(-4)(-7)}{3!},\cdots\ \frac{(-1)(-4)(-7)\cdots(-3r+2)}{r!}\cdots,$$

Ejemplo 6:

Determine el coeficiente de x^{15} en $f(x)=(x^2+x^3+x^4+\cdots)^4$.

Como

$$\begin{aligned}(x^2+x^3+x^4+\cdots) &= x^2(1+x+x^2+\cdots)\\ &= \frac{x^2}{(1-x)}\end{aligned}$$

el coeficiente de x^{15} en $f(x)$ es el coeficiente de x^{15} en

$$\left(\frac{x^2}{1-x}\right)^4=\frac{x^8}{(1-x)^4}$$

Por lo tanto, el coeficiente buscado es el de x^7 en $(1-x)^{-4}$, es decir,

$$\begin{aligned}\binom{-4}{7}(-1)^7 &= (-1)^7\binom{4+7-1}{7}(-1)^7\\ &= \binom{10}{7}\\ &= 120\end{aligned}$$

Ejemplo 7:

¿De cuántas formas se pueden seleccionar r objetos de n distintos objetos si se permite la repetición?

Para cada uno de los n distintos objetos, la serie geométrica

$$1+x+x^2+x^3+\cdots$$

representa las posibles elecciones de ese objeto. Considerando todos los n distintos objetos, la función generatriz es

$$f(x)=(1+x+x^2+x^3+\cdots)^n,$$

y la respuesta requerida es el coeficiente de x^r en $f(x)$. Ahora bien, puesto que

$$\frac{1}{(1-x)}=1+x+x^2+x^3+\cdots$$

y

$$\begin{aligned}\frac{1}{(1-x)^n} &= \binom{-n}{0} + \binom{-n}{1}x + \binom{-n}{2}x^2 + \cdots \\ &= \sum_{i=0}^{\infty}(-1)^r\binom{-n}{i}(-x)^i \\ &= 1 + (-1)\binom{n+1-1}{1}(-x) + (-1)^2\binom{n+2-1}{2}(-x)^2 + \cdots \\ &= \sum_{i=0}^{\infty}\binom{n+i-1}{i}x^i\end{aligned}$$

se tiene que

$$\begin{aligned}(1+x+x^2+x^3+\cdots)^n &= \left(\frac{1}{(1-x)}\right)^n \\ &= \frac{1}{(1-x)^n} \\ &= \sum_{i=0}^{\infty}\binom{n+i-1}{i}x^i,\end{aligned}$$

de modo que el coeficiente de x^r es

$$\binom{n+r-1}{r},$$

Ejemplo 8:

¿De cuántas formas puede un capitán de policía distribuir 24 cargas de rifle a cuatro oficiales de forma que cada oficial reciba al menos tres cargas, pero no más de ocho?

Las opciones para el número de cargas que recibe cada oficial están representadas por

$$x^3 + x^4 + \cdots + x^8.$$

Hay cuatro oficiales, así que la función generatriz resultante es

$$f(x) = (x^3 + x^4 + \cdots + x^8)^4$$

Buscamos el coeficiente de x^{24} en $f(x)$. Con

$$\begin{aligned}(x^3 + x^4 + \cdots + x^8)^4 &= x^{12}(1 + x + x^2 + \cdots + x^5)^4 \\ &= x^{12}\left(\frac{1-x^6}{1-x}\right)^4,\end{aligned}$$

la respuesta es el coeficiente de x^{12} en

$$(1-x^6)^4(1-x)^{-4} \quad =$$

$$= \left[1 - \binom{4}{1}x^6 + \binom{4}{2}x^{12} - \binom{4}{3}x^{18} + x^{24}\right] * \left[\binom{-4}{0} + \binom{-4}{1}(-x) + \cdots\right],$$

que es

$$\left[\binom{-4}{12}(-1)^{12} - \binom{4}{1}\binom{-4}{6}(-1)^6 + \binom{4}{2}\binom{-4}{0}\right] = \left[\binom{15}{5} - \binom{4}{1}\binom{9}{6} + \binom{4}{2}\right]$$
$$= 125$$

Observemos que al extender nuestra experiencia anterior con los polinomios al caso de las series de potencias, y al extender el teorema del binomio $(1+x)^n$ a los casos en que n no tiene que ser positivo ni entero, encontramos las herramientas necesarias para calcular los coeficientes de estas funciones generatrices, ahorrandonos en partícular conteos tediosos.

Veremos más adelante al aplicar esta herramienta (*la función generatriz*), que el habernos detenido a analizar este concepto valió la pena.

Capítulo 3

Árboles

Continuando con nuestro estudio de las herramientas que nos permitirán resolver el problema de multiplicar varias matrices, en este capítulo nos detendremos a estudiar previamente un objeto utilizado en el estudio de las estructuras de datos, las ordenaciones, la teoría de codificación y en la solución de ciertos problemas de optimización. Este objeto es conocido como **árbol** ya que emula la forma o estructura de un árbol.

Estos objetos, los árboles, tendrán un papel central en muchos de los resultados que obtendremos en capítulos posteriores de este texto.

3.1. Árboles

Los árboles fueron utilizados por primera vez en 1847 por Gustav Kirchoff (1824-1887) en su trabajo de redes eléctricas, aunque posteriormente fueron desarrollados y definidos de nuevo por Arthur Cayley(1821-1895).

En la [Figura(3.1)], se muestra un primer acercamiento a estas estructuras.

Un **árbol** como lo mencionamos en renglones anteriores es un objeto que emula la forma de un árbol (conjunto de nodos conectados). Un **nodo** es la unidad sobre la que se construye el árbol y puede tener cero o más nodos hijos conectados a él. Siguiendo la figura anterior **a, b, c, d, e, f** son nodos, y se dice que un nodo **d** es **padre** de un nodo **f** si existe un enlace desde **d** hasta **f** (en este caso también decimos que **f** es hijo de **d**).

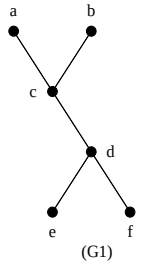
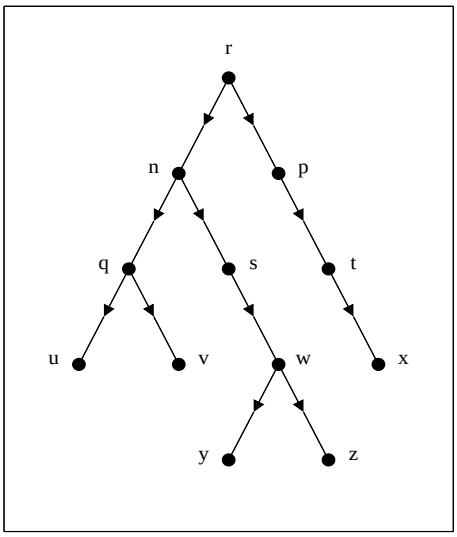


Figura 3.1: Árbol

Figura 3.2: Árbol con raíz



En ocasiones, sólo hay un único nodo sin padres, que llamaremos **raíz**. Así, podemos definir un **árbol con raíz** si existe un único vértice, que es raíz, y para todos los demás vértices(si los hay), el número de enlaces que llegan a ellos es uno. El árbol de la [Figura (3.2)] tiene a **r** como raíz.

Trazaremos árboles con raíz como en la [Figura (3.2)], pero con el convenio de que las direcciones de su trazado van del nivel superior al inferior. Consideremos el vértice s. El camino desde la raíz r hasta s es de longitud 2, por lo que decimos que s está en el *nivel* 2 del árbol, o que s tiene *número de nivel* 2. En forma análoga, x está en el nivel 3, mientras que y tiene número de nivel 4. Decimos que s es un hijo de n y n es padre de s. Los vértices w, y y z son *descendientes* de s, n y r, mientras que s, n y r son *ascendientes* de w, y y z.

EJEMPLO 1: En la [Figura (3.3)] usamos un árbol con raíz para representar el índice de un libro con tres capítulos $(C1, C2, C3)$. Los vértices con el número de nivel 2 son para las secciones de un capítulo; los del nivel 3 representan las subsecciones de una sección.

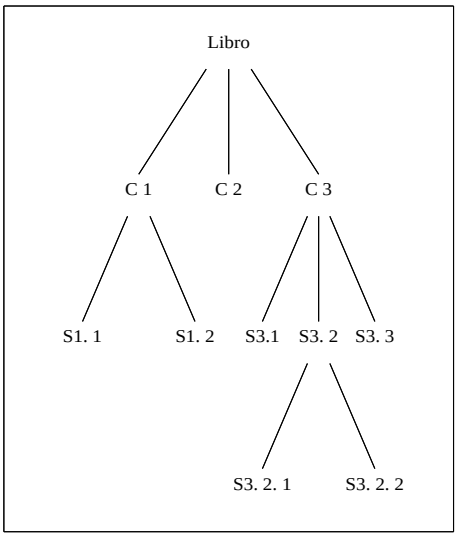
El árbol de la [Figura (3.3)] indica un orden para los vértices, si analizamos los subárboles $(C1, C2, C3)$ de izquierda a derecha.

EJEMPLO 2: El árbol T que se muestra en la [Figura (3.4)], las aristas (o ramas, como se les llama a menudo) que salen de cada vértice interno están *ordenadas* de izquierda a derecha. Por lo tanto, T es un árbol ordenado con *raíz*.

Etiquetemos los vértices de este árbol mediante el siguiente algoritmo.

Paso 1 : Primero asignamos a la raíz la etiqueta (o dirección) 0.

Figura 3.3: Índice de un libro



Paso 2 : A continuación, asignamos los enteros positivos 1, 2, 3,$\cdots$ a los vértices del nivel 1, de izquierda a derecha.

Paso 3 : Ahora, sea ν un vértice en el nivel $n \geq 1$ y sean ν_1, ν_2,$\cdots$,ν_k los hijos de ν (yendo de izquierda a derecha). Si a es la etiqueta asignada al vértice ν, asignamos las etiquetas a.1, a.2,$\cdots$, a.k a los hijos ν_1, ν_2,$\cdots$,ν_k, respectivamente.

En consecuencia, cada vértice de T, excepto la raíz, tiene una etiqueta de la forma $a_1.a_2.a_3 \cdots a_n$ si y sólo si ese vértice tiene número de nivel n. Esto se conoce como el *sistema universal de direcciones*.

Este algoritmo proporciona una manera de *ordenar* todos los vértices de T.

Definición 7 *Un árbol con raíz es binario si cada vértice ν tiene a lo más dos hijos.*

Veamos en las partes (a) y (b) de la [Figura (3.5)] como podemos representar una operación binaria a través de un árbol binario con raíz. Para evitar confusiones al trabajar con una operación no conmutativa " $\circ$ ", etiquetemos la raíz como " $\circ$ " y pedimos que el resultado sea $a \circ b$, donde a es el hijo izquierdo y b es el hijo derecho de la raíz.

En la [Figura (3.6)] extendemos las ideas representadas en la [Figura (3.5)] para construir el árbol binario con raíz para la expresión algebraica

$$((7 - a)/5) * ((a + b) \uparrow 3),$$

Figura 3.4: Árbol T ordenado con raíz

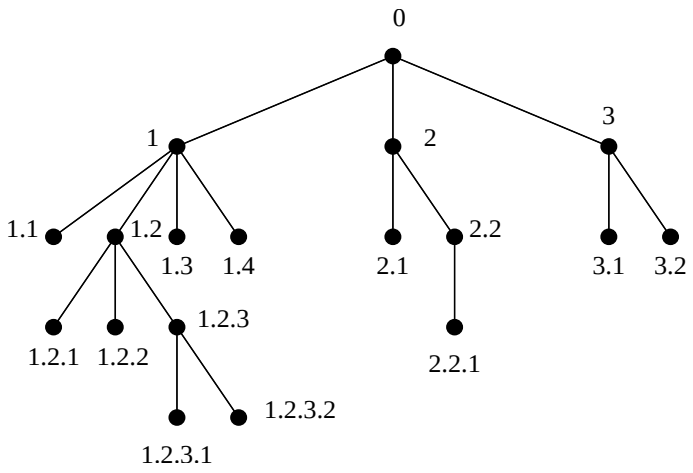
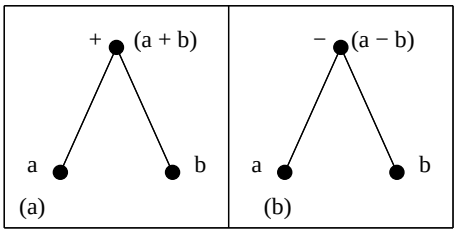


Figura 3.5: Árbol binario con raíz



donde $*$ denota la multiplicación y $\uparrow$ la exponenciación. En este caso construimos el árbol, como aparece en la parte (e) de la Figura, de arriba hacia abajo. En primer lugar construimos un subárbol para la expresión $7 - a$ en la parte (a) de la [Figura (3.6)]. Esto se incorpora (como el subárbol izquierdo de /) en el árbol binario con raíz correspondiente a la expresión $(7 - a)/5$ de la [Figura 3.6 (b)]. Luego, de modo similar, construimos los árboles binarios con raíz de las partes (c) y (d) para las expresiones $a + b$ y $(a + b) \uparrow 3$, respectivamente. Por último, usamos los dos subárboles de las partes (b) y (d) como los subárboles izquierdo y derecho, respectivamente, para $*$, con lo que obtenemos el árbol binario con raíz [Fig. 3.6(e)] para $((7 - a)/5) * ((a + b) \uparrow 3)$.

Observemos que en el ejemplo anterior se considera el concepto adicional de hijos izquierdo y derecho. Veamos un ejemplo más.

Los dos árboles binarios con raíz que se muestran en la [Figura (3.7)] no se consideran iguales. Como árboles binarios con raíz son iguales. Sin embargo, al considerar el concepto adicional de hijos izquierdo y derecho, vemos que, en la parte (a), el vértice ν tiene a como hijo derecho, mientras que en la parte (b), el vértice a es el hijo izquierdo de ν. En consecuencia, cuando se tiene en cuenta la diferencia entre el hijo izquierdo y el derecho, estos árboles ya

Figura 3.6: Árbol binario con raíz para una expresión algebraica

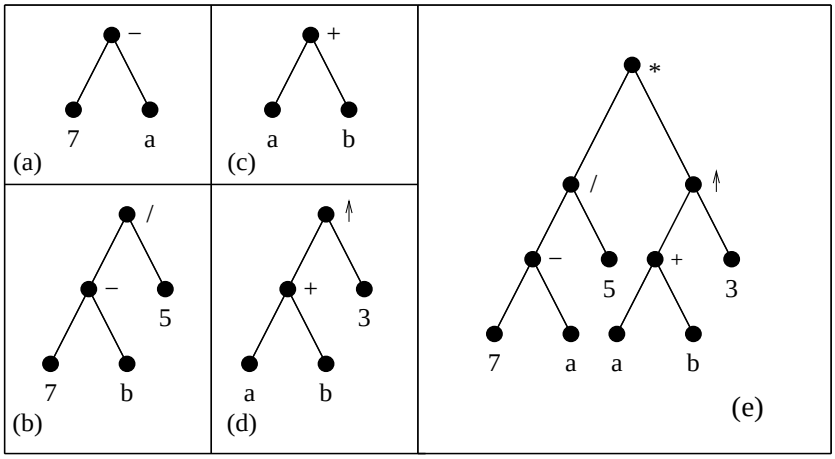
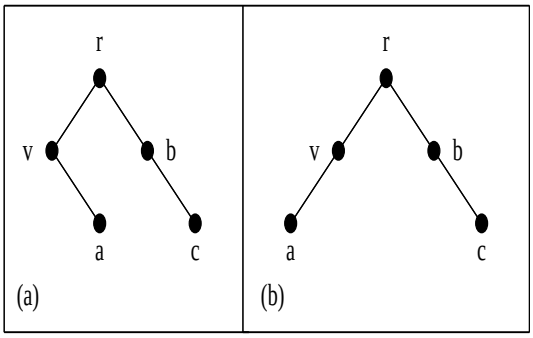


Figura 3.7: Árboles binarios con raíz diferentes



no pueden verse como el mismo.

Como resultado de los comentarios anteriores podemos formular la siguiente pregunta: ¿Cuántos árboles binarios con raíz son iguales en el sentido de que tengan el mismo conjunto de vértices?

Para responder esta pregunta es importante definir primero los árboles binarios con raíz más simples. Por convención, el árbol binario con raíz más simple será **el vacío**. Así, si n representa el número de vértices de un árbol binario, habrá un único árbol binario con raíz para n=0, el vacío.

Ahora si $n = 1$ el árbol binario con raíz queda representado por $\bullet$ (la raíz). Por lo tanto habrá un único árbol binario con raíz que tiene un vértice.

Ahora con esta información veamos si podemos saber cuántos árboles binarios con raíz hay, que tengan 2 vértices.

Figura 3.8: Construcción de los árboles binarios con raíz para n= 2

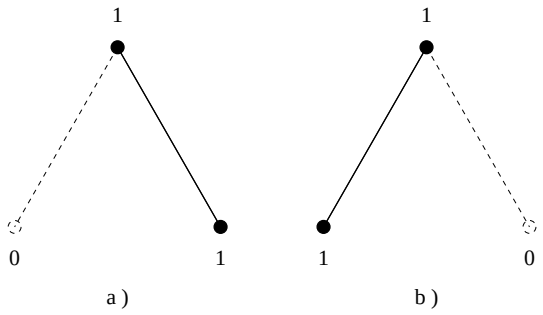
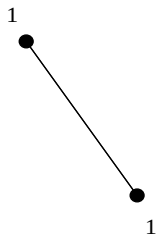


Figura 3.9: Reducción del Árbol binario con raíz con n=2 cuyo hijo izquierdo es nulo.



Para el caso n= 2, la forma en que pueden separarse los dos vértices seleccionando un vértice como la raíz en estos subárboles es:

(1) Asignamos un uno a la raíz.

(2) Cero vértices a la izquierda, un vértice a la derecha es decir $0+1$ vértices, más el vértice que consideramos como raíz da un total de dos vértices.

(3) Un vértice a la izquierda, cero vértices a la derecha es decir $1+0$ vértices, más el vértice que consideramos como raíz son en total dos vértices.

Para el árbol de la [Figura 3.8(a)], el hijo izquierdo del árbol tiene el valor de cero, en este caso el hijo izquierdo es nulo o no existente por lo cual esta figura se reduce a la [Figura (3.9)].

De manera similar en la [Figura 3.8(b)] el hijo derecho del árbol es nulo reduciendose a la [Figura 3.10].

Por lo tanto para el caso de dos vértices existen dos árboles binarios con raíz.

Para el caso $n = 3$ la manera en que pueden separarse los tres vértices es seleccionando nuevamente un vértice como la raíz. Estos subárboles quedan representados por la [Figura (3.11)]

Figura 3.10: Reducción del Árbol binario con raíz con n=2 cuyo hijo derecho es nulo.

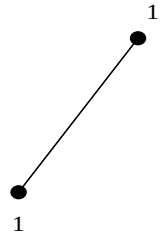
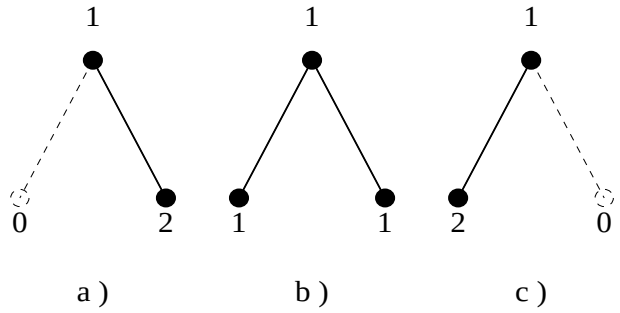


Figura 3.11: Construcción de los árboles binarios con raíz para n=3



Cada vértice distinto de la raíz será visto ahora como la raíz de nuevos árboles binarios con raíz que nacerán a partir de él y el número que aparece en cada vértice representa el número de vértices que tendrán estos nuevos árboles binarios con raíz.

(1) Asignamos un uno a la raíz.

(2) Cero al vértice de la rama izquierda, dos al vértice de la rama derecha es decir $0 + 2$ vértices, más el vértice que consideramos como raíz. Esto produce un total de tres vértices.

(3) Uno al vértice de la rama izquierda, uno al vértice de la rama derecha es decir $1 + 1$ vértices, más el vértice que consideramos como raíz son en total tres vértices.

(4) Dos al vértice de la rama izquierda, cero al vértice de la rama derecha es decir $2 + 0$ vértices, más el vértice que consideramos como raíz resultan tres vértices en total.

Para el caso de la [Figura (3.11)] incisos (a) y (c) significa que a partir del vértice que tiene asignado el número dos nacerán nuevos árboles binarios con raíz que tengan dos vértices. Pero nosotros ya sabemos que existen únicamente dos árboles binarios con raíz de dos vértices diferentes, más aún, sabemos como son y que el vértice que tiene asignado el número 0 es nulo. Para la [Figura 3.11 (b)] sabemos que existe un único árbol binario con raíz que tiene un vértice (•).

Por lo tanto los cinco árboles binarios con raíz para $n = 3$ que resultan de la [Figura (3.11)] incisos (a),(b) y (c) se muestran en la [Figura (3.12)].

Figura 3.12: Construcción y reducción de los árboles binarios con raíz para n= 3

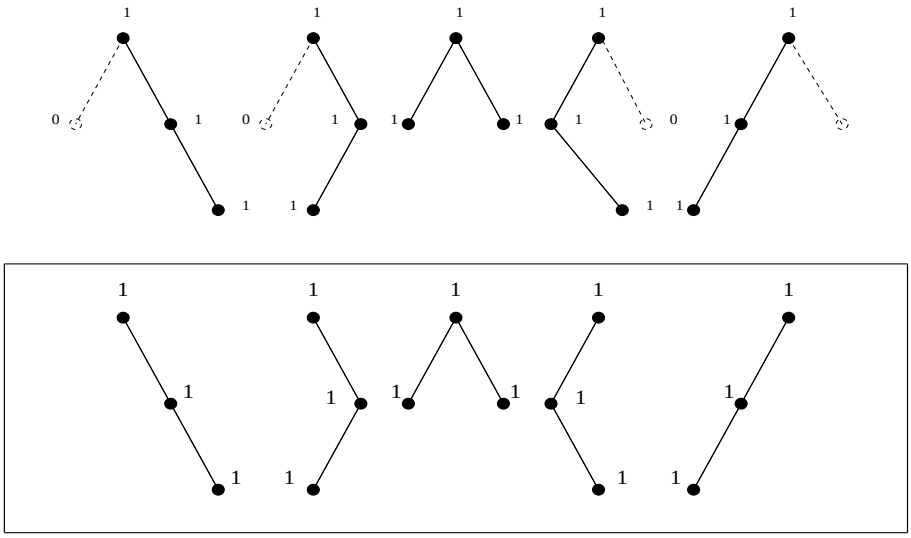
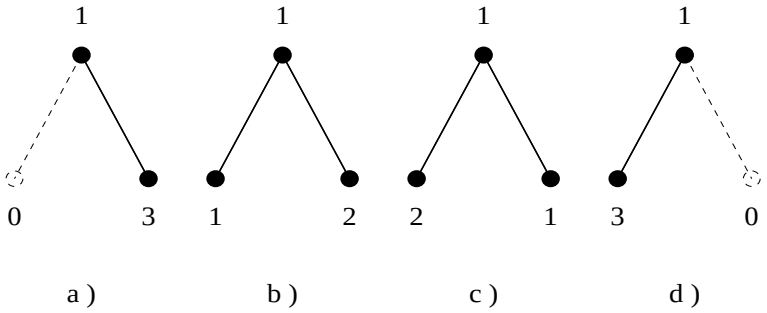


Figura 3.13: Construcción de los árboles binarios con raíz para n= 4



Para $n = 4$, la manera en que pueden quedar separados los cuatro vértices seleccionando un vértice como la raíz quedan representados en la [Figura (3.13)].

Así los vértices que tienen asignados los números 0, 3, 1 y 2 serán vistos como las raíces de nuevos subárboles de 0, 3, 1 y 2 vértices respectivamente que nacerán a partir de estos vértices. Para el caso de la [Figura 3.13(a)], nosotros ya sabemos que el vértice que tiene asignado el número cero es nulo y que hay cinco árboles binarios con raíz diferentes que tienen tres vértices (estos árboles son los que se construyeron en el caso n=3) por lo tanto habrá en total cinco árboles binarios con raíz diferentes los cuales se muestran en la [Figura (3.14)].

Los árboles binarios con raíz resultantes de la [Figura (3.13)] incisos (b) y (c) se muestran en la [Figura 3.15)] puesto que hay un único árbol binario con raíz con un vértice, y dos árboles binarios con raíz de dos vértices diferentes.

Los árboles binarios con raíz que se desprenden de la [Figura 3.13(d)] quedan represen-

Figura 3.14: Reducción de los árboles binarios con raíz para n=4 cuyo hijo izquierdo es nulo, caso a) en la Fig. 3.13

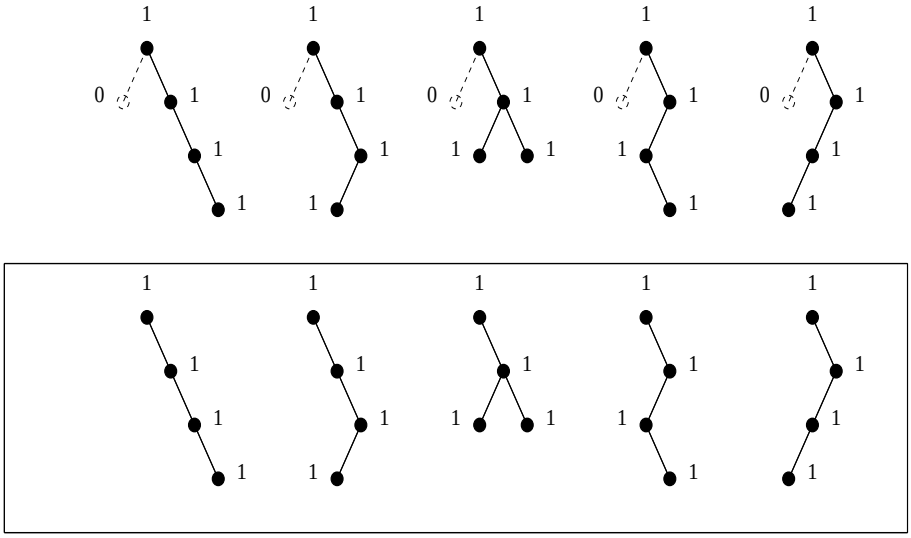
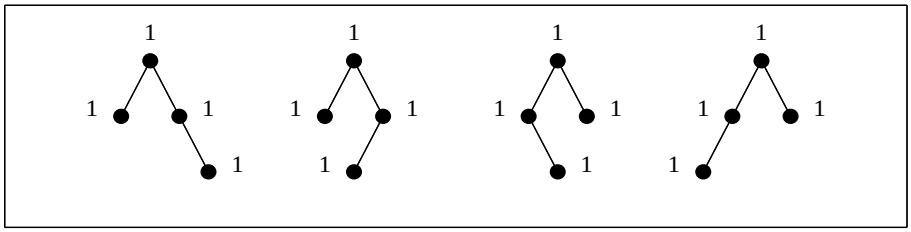


Figura 3.15: Más de los árboles binarios construidos con raíz para n=4, casos b) y c) en la Fig. 3.13



tados en la [Figura (3.16)] los cuales resultan a partir de los árboles con n=0, n=1, n=2, y n=3 vértices construidos anteriormente, siendo 14 el número de árboles binarios con raíz con $n = 4$ vértices

Finalmente para el caso $n = 5$, los árboles binarios con raíz quedan representados de la siguiente manera [Figura (3.17)]:

Nuevamente si asignamos un uno a la raíz, los vértices que tienen asignados los números 0, 4, 1, 3 y 2 serán vistos como las raíces de los nuevos árboles binarios con raíz de 0, 4, 1, 3 y 2 vértices respectivamente. Puesto que hay 14 posibles árboles binarios con raíz diferentes de 4 vértices (los cuales construimos en el caso anterior) entonces en el caso de la [Figura 3.17(a)] habrá 14 posibles árboles binarios con raíz diferentes entre si, que se podrán construir a partir de esta representación [Figura (3.18)].

De la representación del inciso b) habrá 5 posibles árboles binarios con raíz diferentes los

Figura 3.16: Reducción de los árboles binarios con raíz para n=4 cuyo hijo derecho es nulo, caso d) en la Fig. 3.13

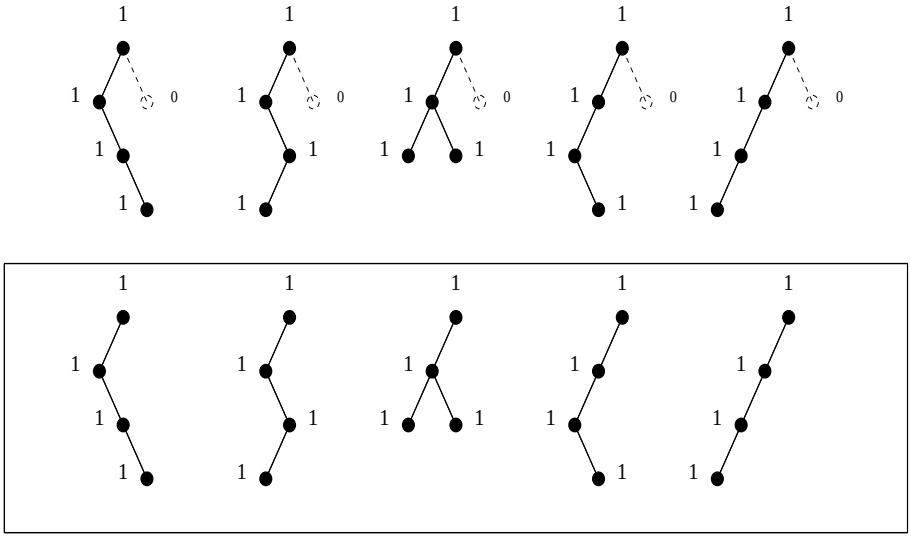
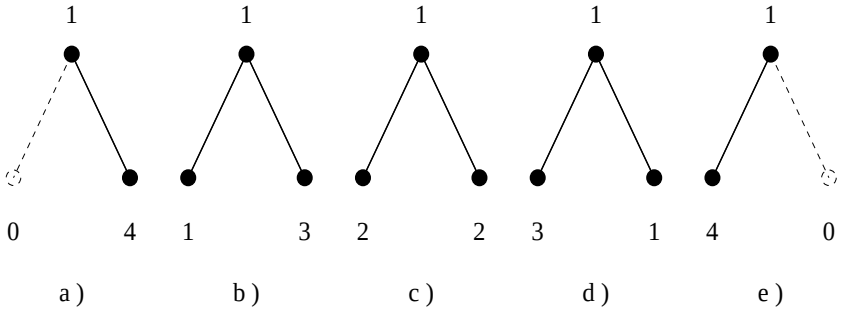


Figura 3.17: Construcción de los árboles binarios con raíz para n= 5



cuales se muestran en la [Figura (3.19)].

Para el inciso c) de manera similar para el vértice asignado con el número dos habrá dos posibles árboles binarios con raíz de dos vértices (los cuales ya sabemos como son) este árbol quedará representado por los árboles binarios con raíz de la [Figura (3.20)].

Continuando, para el inciso d) habrán 5 posibles árboles binarios con raíz diferentes [Figura (3.21)] puesto que las razones para la construcción de estos árboles son las mismas que para el inciso b).

Finalmente los posibles árboles binarios con raíz diferentes que se construyen para el caso del inciso e) son nuevamente 14 [Figura (3.22)] puesto que hay 14 posibles árboles binarios con raíz de 4 vértices.

Por lo tanto hay 42 posibles árboles binarios con raíz de cinco vértices en total formados a partir de los posibles árboles binarios con raíz de 4, 3 2 y 1 vértices construidos anteriormente.

Como resultado de estas construcciones ya podemos responder la pregunta sobre cuántos árboles binarios con raíz son iguales en el sentido de tener el mismo conjunto de vértices, y también podremos dar la conclusión de que el número de árboles binarios con raíz iguales crece rápidamente pues para el caso de n=3 se construyeron 5 posibles árboles binarios con raíz, 14 posibles árboles con raíz para n= 4 y finalmente 42 árboles binarios con raíz para n=5 lo cual significaría que construir y calcular los posibles árboles binarios con raíz para un n suficientemente grande nos llevaría mucho tiempo por lo cual ahora nos haremos la siguiente pregunta ¿Habrá alguna manera menos complicada de saber cuántos árboles binarios con raíz de n vértices hay para n suficientemente grande sin que el saberlo nos lleve mucho tiempo?.

Ahora nuestra pregunta se reduce a contar, para $n \geq 0$, el número b_n de árboles binarios ordenados con raíz de n vértices. Usemos pues la misma idea que utilizamos para construir los áboles binarios de igual número de vértices. Así, si conocemos los valores de b_i para $0 \leq i \leq n$, para obtener b_{n+1} seleccionamos un vértice como la raíz y vemos que las subestructuras que descienden a la izquierda y a la derecha de la raíz son árboles (binarios ordenados con raíz) más pequeños. Estos árboles más pequeños se denominan *subárboles* del árbol dado.

Comenzemos pues a contar el número b_n de árboles binarios ordenados con raíz de n vértices.

Recordemos que existe un único árbol de cero vértices, este es el árbol vacío por lo tanto $b_0 = 1$.

Para el caso del árbol binario con raíz de un sólo vértice, $(b_1 = 1)$

Para contar el número de árboles binarios con raíz de dos vértices separamos los dos vértices en subárboles, seleccionando un vértice como la raíz. [Figura(3.23)]

(a) 0 vértices a la izquierda, 1 vértice a la derecha. Esto produce un total de $b_0 b_1$ subestructuras por contar.

(b) 1 vértice a la izquierda, 0 vértices a la derecha, lo que produce $b_0 b_1$ árboles binarios ordenados con raíz, con 2 vértices.

Pueso que nosotros ya conocemos los valores para b_0 y b_1, para obtener b_2 [Figura (3.23)], $b_2 = b_0\ b_1 + b_1\ b_0 = (1)\ (1) + (1)\ (1) = 2$, por lo tanto en total son dos el número b_2 de árboles binarios ordenados con raíz de dos vértices.

Ahora que ya conocemos los valores de b_0, b_1 y b_2 para obtener b_3 recordemos nuevamente los árboles binarios con raíz de tres vértices y la forma en que separamos los tres vértices seleccionando de igual manera un vértice como la raíz [Figura (3.24)].

(a) 0 vértices a la izquierda, 2 vértices a la derecha. En total $b_0 b_2$ de subestructuras.

(b) 1 vértice a la izquierda, 1 vértices a la derecha. Esto es $b_1 b_1$ árboles binarios con raíz.

(c) 2 vértices a la izquierda, 0 vértices a la derecha, en total $b_2 b_0$ de los árboles.

Así, $b_3 = b_0\ b_2 + b_1\ b_1 + b_2\ b_0 = (1)\ (2) + (1)\ (1) + (2)\ (1) = 5$. Esto produce un total de cinco árboles binarios ordenados con raíz de tres vértices.

Continuando ahora para el caso de los árboles binarios ordenados con raíz de cuatro vértices los modelos para este tipo de árbol son [Figura (3.25)] en los cuales separamos los cuatro vértices seleccionando un vértice como la raíz.

(1) 0 vértices a la izquierda, 3 vértices a la derecha, obteniendose un total de $b_0 b_3$ árboles binarios con raíz.

(2) 1 vértice a la izquierda, 2 vértices a la derecha, lo que produce $b_1 b_2$ en total de árboles.

(3) 2 vértice a la izquierda, 1 vértices a la derecha, resultando $b_2 b_1$ subestructuras por contar.

(4) 3 vértice a la izquierda, 0 vértices a la derecha, para un total de $b_3 b_0$ árboles.

Además puesto que en los casos anteriores hemos calculado $b_0 = 1$, $b_1 = 1$, $b_2 = 2$, y $b_3 = 5$ con ellos calculemos b_4.

$b_4 = b_0\ b_3 + b_1\ b_2 + b_2\ b_1 + b_3\ b_0 = (1)\ (5) + (1)\ (2) + (2)\ (1) + (5)\ (1) = 14$, el total de árboles binarios ordenados con raíz de cuatro vértices.

Por último calculemos b_5.

Examinemos la forma en que podemos separar los cinco vértices seleccionando un vértice como la raíz [Figura (3.26)].

(1) 0 vértices a la izquierda, 4 vértices a la derecha. En total $b_0 b_4$ árboles binarios con raíz.

(2) 1 vértice a la izquierda, 3 vértices a la derecha, lo que produce $b_1 b_3$ en total de árboles.

(3) 2 vértices a la izquierda, 2 vértices a la derecha, resultando $b_2 b_2$ subestructuras por contar.

(4) 3 vértices a la izquierda, 1 vértice a la derecha, en total de $b_3 b_1$ árboles.

(5) 4 vértices a la izquierda, 0 vértices a la derecha, es decir $b_4 b_0$ árboles.

En consecuencia, para b_5 se tiene:

$$\begin{aligned} b_5 &= b_0 b_4 + b_1 b_3 + b_2 b_2 + b_3 b_1 + b_4 b_0 \\ &= (1)(14) + (1)(5) + (2)(2) + (5)(1) + (14)(1) \\ &= 42 \end{aligned}$$

Por lo tanto hay 42 árboles binarios ordenados con raíz de cinco vértices.

Ahora que hemos analizado varios ejemplos, finalmente generalizaremos los resultados anteriores:

contemos para $n \geq 0$, el número b_n de árboles binarios ordenados con raíz de n vértice. Así, si conocemos los valores de b_i para $0 \leq i \leq n$, para obtener b_{n+1} seleccionamos un vértice como la raíz y observemos la forma en que pueden separarse los n vértices en estos subárboles [Figura (3.27)]:

(1) 0 vértices a la izquierda, n vértices a la derecha. Esto produce un total de $b_0 b_n$ subestructuras por contar en b_{n+1}

(2) 1 vértice a la izquierda, $n-1$ vértices a la derecha, lo que produce $b_1 b_{n-1}$ árboles binarios ordenados con raíz, con $n+1$ vértices.

...

(i+1) i vértices a la izquierda, n-i vértices a la derecha, para un total de $b_i b_{n\text{-}i}$ hacia b_{n+1}.

...

(n+1) n vértices a la izquierda y ninguno a la derecha, lo que contribuye con $b_n b_0$ de los árboles.

Por lo tanto, para todo $n \geq 0$,

$$b_{n+1} = b_0 b_n + b_1 b_{n-1} + b_2 b_{n-2} + \cdots + b_{n-1} b_1 + b_n b_0 \tag{3.1}$$

y

$$\sum_{n=0}^{\infty} b_{n+1} x^{n+1} = \sum_{n=0}^{\infty} (b_0 b_n + b_1 b_{n-1} + \cdots + b_{n-1} b_1 + b_n b_0) x^{n+1} \tag{3.2}$$

Buscamos ahora dar una expresión explícita de los términos b_n. Para ello, sea $f(\mathrm{x}) = \sum_{n=0}^{\infty} b_n x^n$ la función generatriz de $b_0, b_1, b_2, \cdots$ se tiene que

$$\begin{aligned} f(x)f(x) &= \sum_{n=0}^{\infty} (b_0 b_n + b_1 b_{n-1} + \cdots + b_n b_0) x^n \\ &= \sum_{n=0}^{\infty} b_{n+1} x^n \end{aligned}$$

por (3.1). Luego,

$$x[f(x)]^2 = \sum_{n=0}^{\infty} b_{n+1}x^{n+1} = b_1x + b_2x^2 + \cdots .$$

es decir

$$x[f(x)]^2 + b_0 = f(x)$$

por lo tanto

$$x[f(x)]^2 + 1 = f(x)$$

puesto que $b_0 = 1$.

Esto produce la ecuación cuadrática (en f(x))

$x[f(x)]^2 - f(x) + 1 = 0$, y $\quad f(x) = [1 \pm \sqrt{1-4x}]/2x$

Siendo f(x) una función, debemos eliminar la ambiguedad del doble signo.

Tenemos

$$\begin{aligned}\sqrt{1-4x} &= (1-4x)^{\frac{1}{2}} \\ &= \binom{1/2}{0} + \binom{1/2}{1}(-4x) + \binom{1/2}{2}(-4x)^2 + \cdots\end{aligned}$$

donde el coeficiente de x^n, $n \geq 1$, es

$$\begin{aligned}\binom{1/2}{n}(-4)^n &= \frac{(1/2)\big((1/2)-1\big)\big((1/2)-2\big)\big((1/2)-n+1\big)}{n!}(-4)^n \\ &= (-1)^{n-1}\frac{(1/2)(1/2)(3/2)\cdots\big((2n-3)/2\big)}{n!}(-4)^n \\ &= \frac{(-1)2^n(1)(3)\cdots(2n-3)}{n!} \\ &= \frac{(-1)2^n(n!)(1)(3)\cdots(2n-3)(2n-1)}{(n!)(n!)(2n-1)} \\ &= \frac{(-1)(2)(4)\cdots(2n)(1)(3)\cdots(2n-1)}{(n!)(n!)(2n-1)} \\ &= \frac{(-1)}{(2n-1)}\binom{2n}{n}\end{aligned}$$

En f(x), seleccionamos el radical negativo; en caso contrario, tendríamos valores negativos para b_n. Entonces

$$f(x) = \frac{1}{2x}\Big[1 - \Big(1 - \sum_{n=1}^{\infty}\frac{1}{(2n-1)}\binom{2n}{n}x^n\Big)\Big] = \sum_{n=1}^{\infty}\frac{1}{(2n-1)}\binom{2n}{n}x^n.$$

que también es, recordemos, $f(\mathrm{x})=\sum_{n=0}^{\infty} b_n x^n$ la función generatriz de $b_0, b_1, b_2, \cdots$

Aquí b_n, el coeficiente de x_n en $f(\mathrm{x})$, es la mitad del coeficiente de x^{n+1} en

$$\sum_{n=1}^{\infty} \frac{1}{(2n-1)} \binom{2n}{n} x^n.$$

Así,

$$\begin{aligned}
b_n &= \frac{1}{2}\left[\frac{1}{2(n+1)-1}\right]\binom{2(n+1)}{n+1} \\
&= \frac{1}{2}\left[\frac{1}{2n+2-1}\right]\left[\frac{(2n+2)!}{(n+1)!(2n+2-n-1)!}\right] \\
&= \frac{1}{2}\left[\frac{1}{2n+1}\right]\left[\frac{(2n+2)!}{(n+1)!(n+1)!}\right] \\
&= \frac{1}{2}\left[\frac{1}{2n+1}\right]\left[\frac{(2n+2)(2n+2-1)(2n+2-2)(2n+2-3)(2n+2-4)\cdots(3)(2)(1)}{n!(n+1)(n+1)!}\right] \\
&= \frac{1}{2}\left[\frac{1}{2n+1}\right]\left[\frac{2(n+1)(2n+1)(2n)(2n-1)(2n-2)\cdots(3)(2)(1)}{(n+1)n!(n+1)!}\right] \\
&= \frac{2n!}{(n+1)!n!} = \frac{2n!}{n!n!(n+1)} = \frac{1}{(n+1)}\binom{2n}{n}
\end{aligned}$$

En este momento vemos que para $n \geq 0$, el número b_n de árboles binarios ordenados con raíz de n vértices es:

$$b_n = \frac{1}{(n+1)}\binom{2n}{n}$$

El cual crece exponencialmente para n suficientemente grande. De hecho se puede probar que $b_n = \frac{1}{(n+1)}\binom{2n}{n}$ es de orden de complejidad $\Omega\left(\frac{4^n}{n^{\frac{3}{2}}}\right)$. El concepto de orden de complejidad y la notación Ω serán tratados en el capítulo siguiente.

Los números b_n llamados los **números de Catalan**, en honor al matemático belga Eugene Catalan$(1814-1894)$, los usó **para determinar el número de formas para colocar paréntesis en la expresión** $x_1, x_2 \cdots x_n$. Los primeros 11 números de Catalan son $b_0 = 1, b_1 = 1, b_2 = 2, b_3 = 5, b_4 = 14, b_5 = 42, b_6 = 132, b_7 = 429, b_8 = 1430, b_9 = 4862, b_{10} = 16796$. Estos números tomarán sentido en el capítulo 5 cuando se quiera determinar el número de formas de colocar paréntesis a la hora de intentar calcular el producto de varias matrices.

Figura 3.18: Reducción de los árboles binarios con raíz para n=5 cuyo hijo izquierdo es nulo, caso a) en la Fig. 3.17

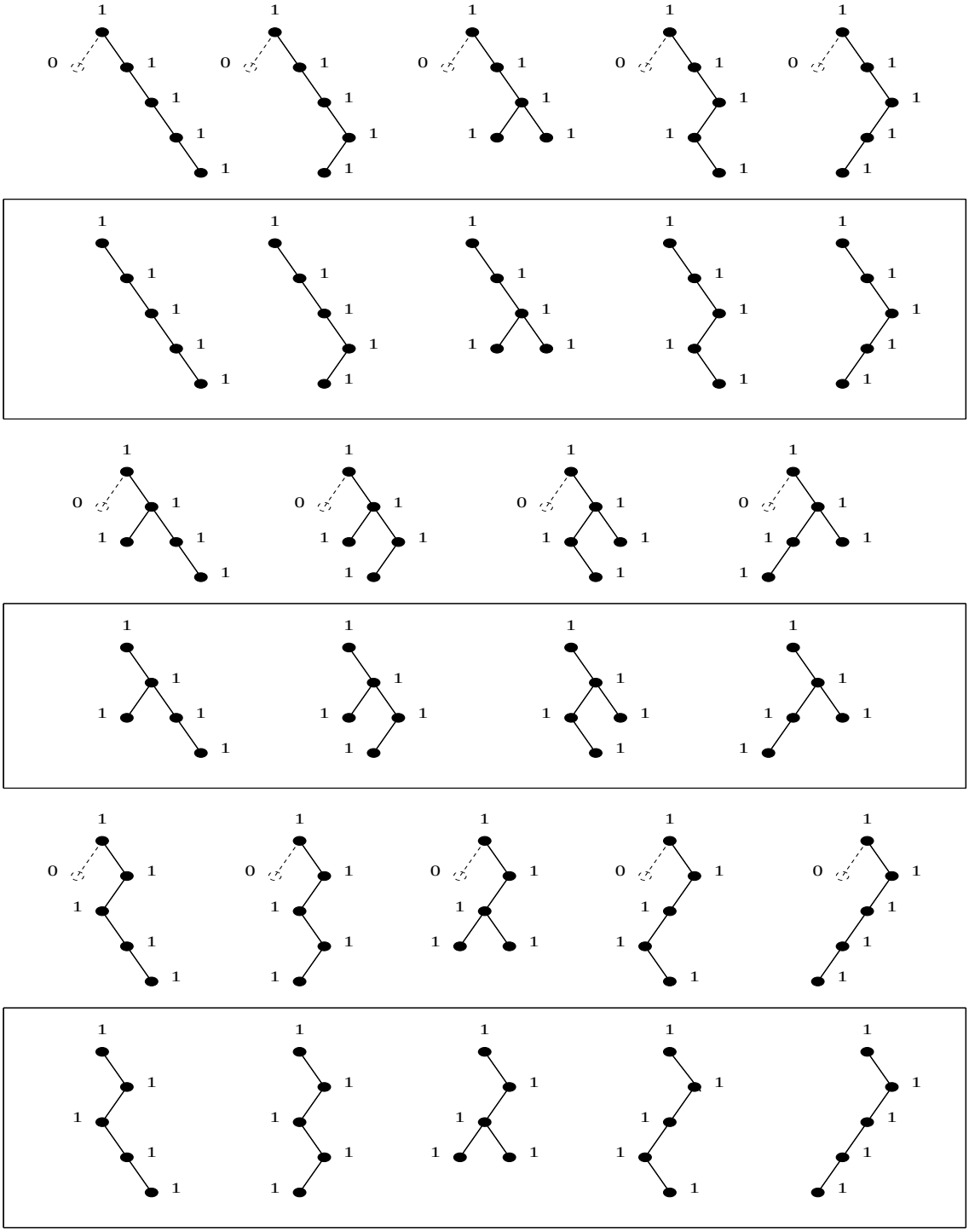


Figura 3.19: Más árboles binarios con raíz para n=5, caso b) en la Fig. 3.17

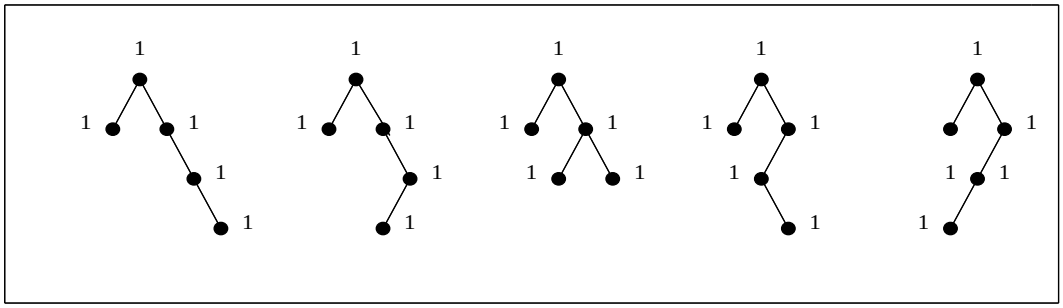


Figura 3.20: Más árboles binarios con raíz para n= 5, caso c) en la Fig. 3.17

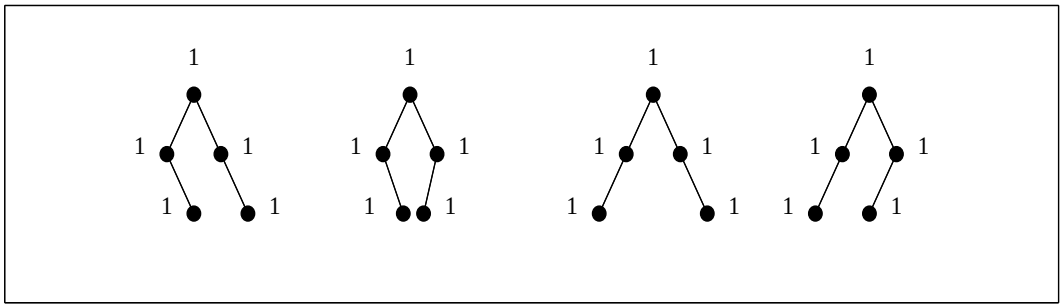


Figura 3.21: Más árboles binarios con raíz para n=5, caso d) en la Fig. 3.17

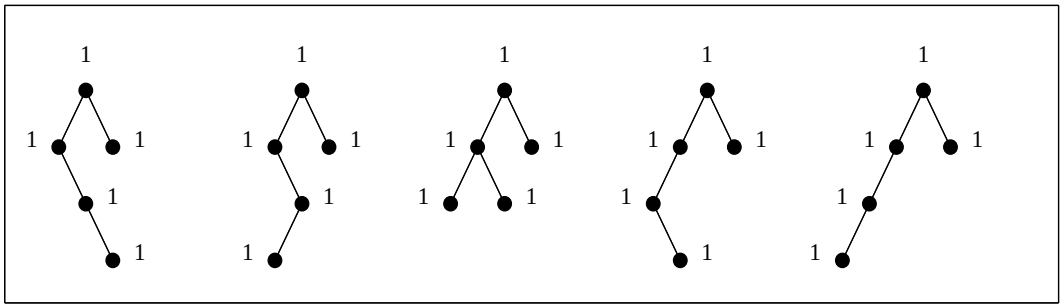


Figura 3.22: Reducción de los árboles binarios con raíz para n= 5 cuyo hijo derecho es nulo, caso e) en la Fig. 3.17

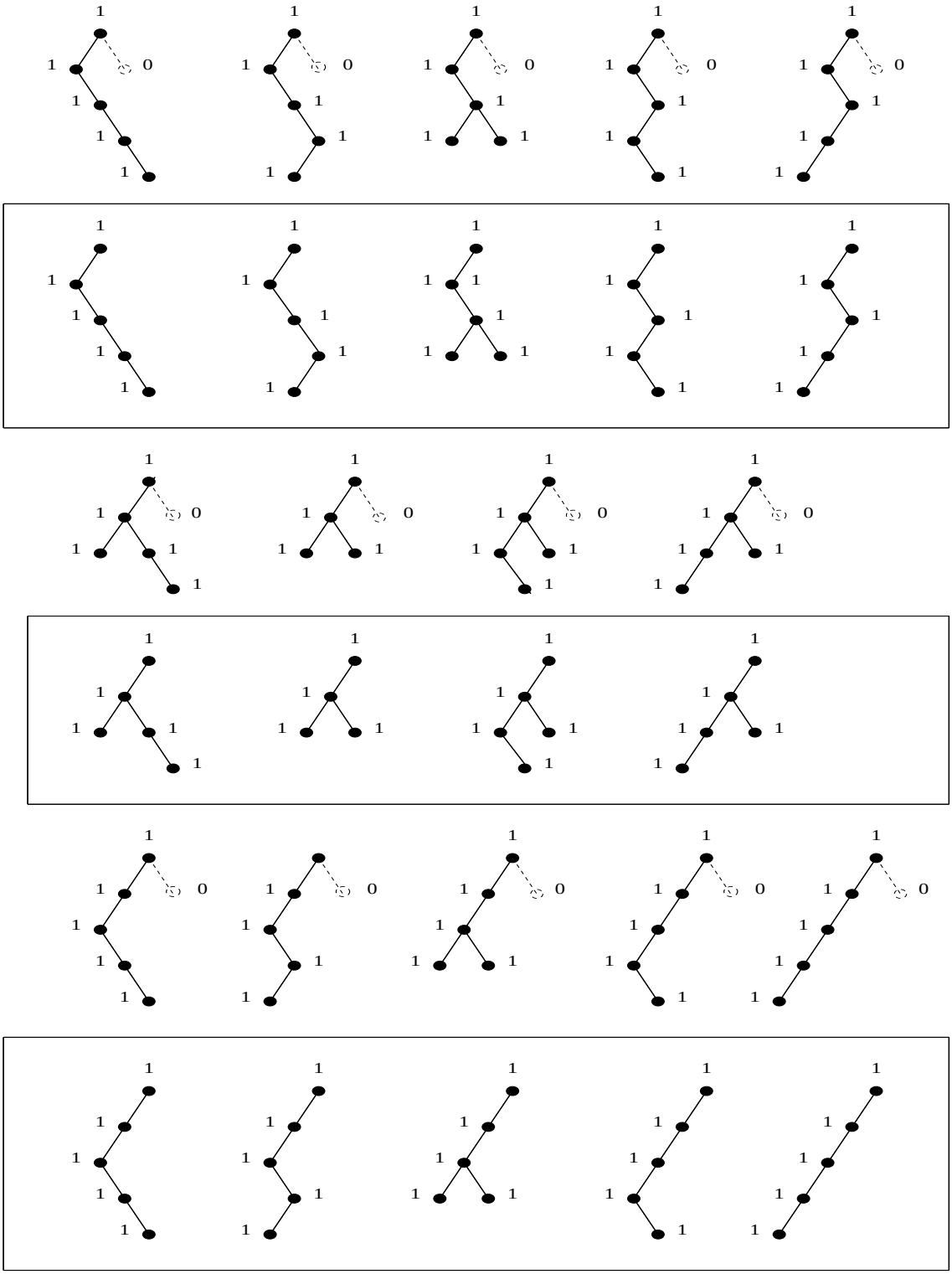


Figura 3.23: Construcción y conteo de los árboles binarios para n=2

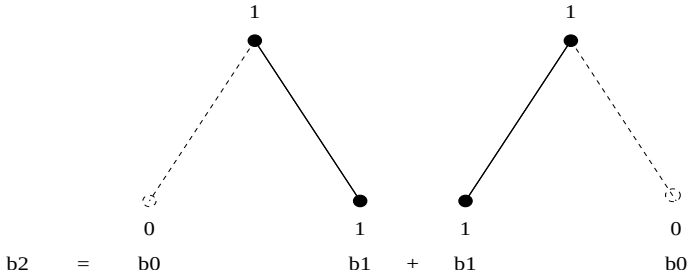


Figura 3.24: Construcción y conteo de los árboles binarios ordenados con raíz de tres vértices

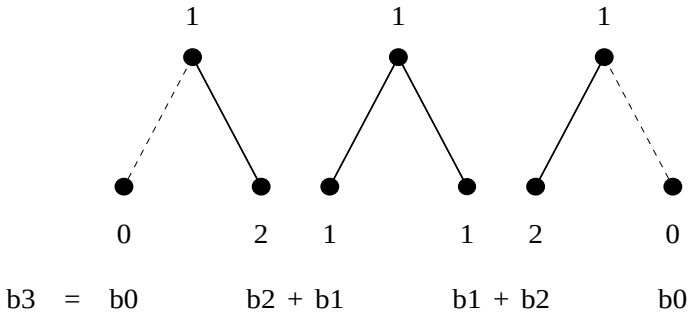


Figura 3.25: Construcción y conteo de los árboles binarios ordenados con raíz de cuatro vértices

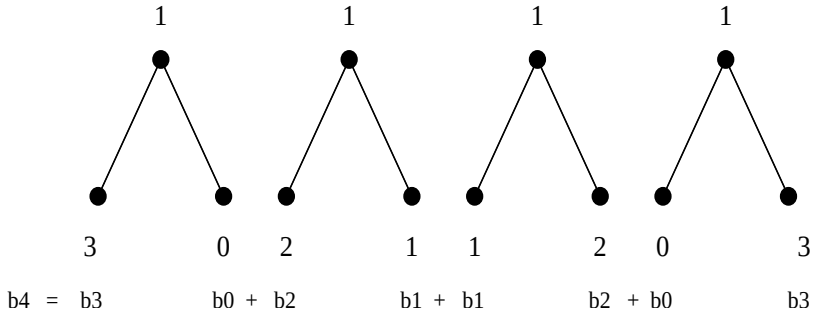


Figura 3.26: Construcción y conteo de los árboles binarios ordenados con raíz de cinco vértices

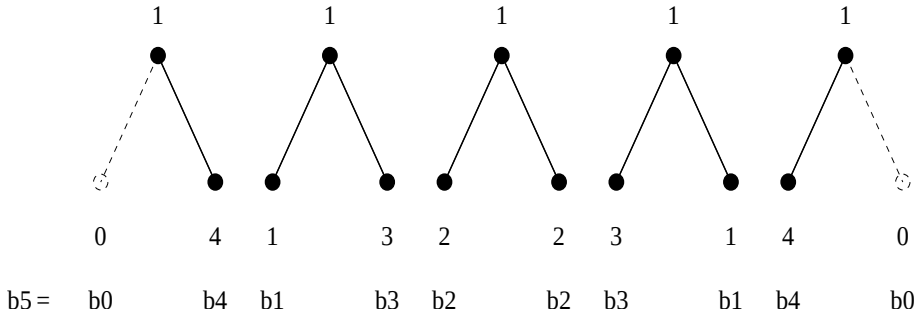
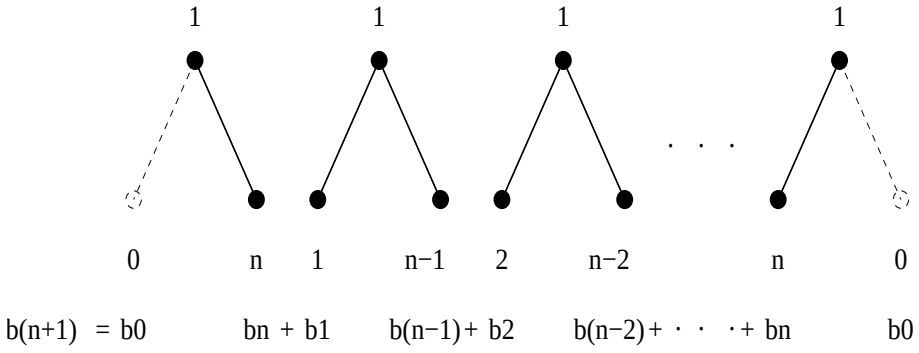


Figura 3.27: Construcción y conteo de los árboles binarios ordenados con raíz de n vértices



Capítulo 4

Notación Asintótica

4.1. Complejidad Algorítmica

En la ciencia de la computación los algoritmos son más importantes que las computadoras, la solución de un problema haciendo uso de las computadoras requiere por una parte un algoritmo o método de solución y por otra un programa o codificación del algoritmo. Ambas componentes tienen importancia; pero la del algoritmo es absolutamente indispensable; sabemos que un algoritmo es una secuencia de pasos para resolver un problema, sus caracteristicas son:

- *Independencia:* De la máquina.
- *Definido:* De pasos claros y concretos.
- *Finito:* En el número de pasos que usará.
- *Preciso:* Cada paso arroja un cálculo correcto.
- *Recibe datos:* Debe poseer datos de entrada.
- *Arroja Información:* Debe arrojar información de salida.

Debemos saber que una solución es un conjunto único, pero no es el único conjunto de pasos que entregan la solución, existen muchas alternativas de solución y estas alternativas pueden diferir por:

- *Número de pasos*
- *Estructuras*

Ahora que sabemos que puede haber muchas alternativas de solución para un problema, debemos seleccionar el algoritmo más eficiente, el mejor conjunto de pasos, el que tarde menos en ejecutarse y que tenga menos líneas de codigo. Esta elección puede ser ejecutada a simple vista con solo observar la cantidad de líneas del programa, pero cuando el programa crece se requiere una medición más exacta y apropiada, para esto se realizan ciertas operaciones matemáticas que establecen la eficiencia teórica del programa, al estudio de estos casos se denomina **Complejidad Algorítmica**.

Un algoritmo será más eficiente comparado con otro, siempre que consuma menos recursos, como el tiempo y espacio de memoria necesarios para ejecutarlo.

La eficiencia de un algoritmo puede ser cuantificada con las siguientes medidas de complejidad.

- *Complejidad Temporal o Tiempo de ejecución:* Tiempo de cómputo necesario para ejecutar algún programa.

- *Complejidad Espacial:* Memoria que utiliza un programa para su ejecución.

Este análisis se basa en las Complejidades Temporales, con este fin, para cada problema determinaremos una medida n, que llamaremos tamaño de la entrada o número de datos a procesar por el programa, intentaremos hallar respuesta en función de dicha n.

El concepto exacto que cuantifica n dependerá de la naturaleza del problema si hablamos de una matriz se puede ver a n como el número de elementos que la componen, para un árbol, podría ser el número de nodos, no se puede establecer una regla para n pues cada problema acarrea su propia lógica y complejidad.

4.2. Tiempo de ejecución

El tiempo de ejecución de un programa se mide en función de n (entrada o número de datos), lo que designaremos como *T(n)*.

Esta función se puede calcular físicamente ejecutando el programa acompañado de un reloj, o calcularse directamente sobre el código, contando las instrucciones a ser ejecutadas y multiplicado por el tiempo requerido por cada instrucción.

Ejemplo 1: En el siguiente párrafo reproducimos un segmento del programa en Pascal que implementa un algoritmo para el cálculo de $n!$ para $n \in \mathbf{Z}^+$. Suponiendo que todas las operaciones implicadas tardan el mismo tiempo en ejecutarse, en este caso, el usuario introduce el valor de n, que es el dato del programa. Las variables i y Factorial (ya declaradas con anterioridad en el programa) son variables enteras.

```
Begin
  i := 1;                          (Inicializa el contador)
  Factorial := 1;                  (Inicializa el valor de Factorial)

  While i <= n do
     Begin
        Factorial := i*Factorial;
        i := i + 1
     End;

  Writeln ( 'El valor de ' , n,  ' factorial es ',  Factorial, '.')
End;
```

El programa comienza con dos proposiciones de asignación, donde se inicializan los valores de las variables enteras i y Factorial. Después se ejecuta el ciclo While n veces. Cada una de estas ejecuciones implica las siguientes cinco operaciones:

1) Comparar el valor actual de contador i con n

2) Incrementar el valor de Factorial como i* Factorial; esto implica una multiplicación y una asignación

3) Incrementar el valor del contador en 1; esto implica una suma y una asignación

Por último, hay otra comparación. Ésta se realiza cuando $i = n+1$, de modo que el ciclo While se termina y las otras cuatro operaciones (de los pasos 2 y 3 anteriores) no se llevan a cabo.

Por lo tanto, *T(n)*= $(2+5n+1)t_1 = (5n+3)t_1$ donde t_1 es el tiempo que lleva ejecutarse cada operación.

Toda función *T(n)* encierra referencias al párametro n, y a una serie de constantes t_i dependientes de factores externos al algoritmo. Se tratará de analizar los algoritmos dándoles autonomía frente a estos factores externos, buscando estimaciones generales ampliamente válidas.

Además podemos hablar de un rango de valores:

$$T_{min}(n) \leqq T(n) \leqq T_{max}(n)$$

Estos extremos son llamados *" el peor caso "* y el *" mejor caso "* y entre ambos se puede hallar *"el caso promedio"*, brindándonos una medida pesimista pero fiable.

4.3. Notación Asintótica

El análisis de la efeciencia algorítmica nos lleva a estudiar el comportamiento de los algoritmos frente a condiciones extremas. Diferentes funciones *g(n)* determinan el uso de recursos, pudiendo existir infinidad de funciones. Estas funciones *g* serán congregadas en familias usando como criterio de agrupación su comportamiento asintótico, pues representa el comportamiento cuando el tamaño de la entrada "n" tiende a infinito, pues para valores grandes es cuando puede haber problemas de tiempo o memoria.

Para cada uno de estos conjuntos se suele identificar un miembro *f(n)* que se utiliza como representante de la familia.

4.3.1. O-notación

Definición 8 *Dada una función, f(n) denotamos por O(*f(n)*) al conjunto de funciones O(*f(n)*)={g:existen constantes positivas c y* η_0 *tal que* $0 \leq g(n) \leq cf(n)$ *para todo* $n \geq \eta_0$*}.*

La familia de funciones que compartan este comportamiento asintótico será llamado un *Orden de Complejidad*. Este conjunto esta formado por aquellas funciones que crecen a un ritmo menor o igual que el de *f(n)*.

De las funciones *g* que forman este conjunto O(*f(n)*) se dice que estan dominados asintóticamente por *f*, en el sentido de que para *n* suficientemente grande, y salvo una constante multiplicativa *c*, *f(n)* es una cota superior de *g(n)*.

Para indicar que una función es un miembro de O(*f(n)*), escribimos

$$g(n) = O(f(n))$$

y para indicar que una función *no* es un miembro de O(*f(n)*), escribiremos

$$g(n) \neq O(f(n))$$

Ejemplo 1: ¿Es $2^{n+1} = O(2^n)$?

Para saber esto, debemos determinar constantes positivas c y η_0 tal que $0 \leq 2^{n+1} \leq c(2^n)$ para todo $n \geq \eta_0$.

Para darnos una idea de como pueden ser estas constantes positivas, veamos como se comportan estas funciones para distintos valores de n. El cálculo de los valores de 2^{n+1} y 2^n, se encuentran detallados en el siguiente Cuadro (4.1)

Como podemos observar, para $n \geq 2$ al comparar por renglón uno a uno los valores de 2^{n+1} y 2^n nos damos cuenta que estos difieren siempre en un valor constante, 2. Esto nos permite darnos una idea intuitiva del valor de las constantes positivas que necesitamos seleccionar para comprobar que en efecto $2^{n+1} = O(2^n)$. Por lo tanto escogiendo $c = 2$ y $\eta_0 = 2$ se tiene que,

$$0 \leq 2^{n+1} \leq 2(2^n)$$

Cuadro 4.1: Tabla de valores

n	2^{n+1}	2^n
0	2	1
1	4	2
2	8	4
3	16	8
4	32	16
5	64	32
6	128	64
7	256	128

para todo $n \geq 2$.

La notación " $O()$ " ignora los factores constantes, desconoce si se hace una mejor o peor implementación del algoritmo, además de ser independiente de los datos de entrada del algoritmo. Es decir, la utilidad de aplicar esta notación a un algoritmo es en contar el limite superior de su tiempo de ejecución, " el peor caso".

Ejemplo 2: Sean *f, g*:$\mathbf{Z}^+ \to \mathbf{R}$ dadas por $f(n) = 5n$, $g(n) = n^2$, para $n \in \mathbf{Z}^+$. Si calculamos *f(n)* y *g(n)* para $1 \leq n \leq 4$, encontramos que $f(1)= 5$, $g(1)= 1$; $f(2)= 10$, $g(2)= 4$; $f(3)= 15$, $g(3)= 8$; y, $f(4)= 20$, $g(4)= 16$. Sin embargo, $n \geq 5 \Longrightarrow n^2 \geq 5n$, y tenemos que $|f(n)|= 5n \leq n^2 = |g(n)|$. Así, si $c= 1$ y $\eta_0= 5$, tenemos que $n \geq \eta_0$, $|f(n)| \leq c\ |g(n)|$. En consecuencia, g domina a f y f=O(g).

También nos damos cuenta de que para cualquier $n \in \mathbf{Z}^+$, $|f(n)| = 5n \leq 5n^2 = 5|g(n)|$. Aquí se muestra el dominio de f por g, con $\eta_0= 1$ y $c = 5$. Esto es suficiente para demostrar que las constantes η_0 y c de la definición *no* necesitan ser únicas.

Además, podemos generalizar este resultado si consideramos las funciones f_1, g_1:$\mathbf{Z}^+ \to \mathbf{R}$ definidas por $f_1(n) = an$, $g_1(n) = bn^2$, donde *a,b* son números reales distintos de cero. Si $c \in \mathbf{R}^+$ y $c\,|b| \geq |a|$, entonces para todo $n \geq 1 (= \eta_0)$, $|f_1(n)|= |an| = |a|\, n \leq c\,|b|\, n \leq c\,|b|\, n^2 = c\,|bn^2| = c\,|g_1(n)|$, y así $f_1 = O\,(g_1)$.

En el ejemplo anterior , observamos que $f = O\,(g)$. Si revisamos las funciones *f, g*, queremos ahora mostrar que $g \neq O\,(f)$.

Ejemplo 3: Una vez más, sean *f, g*:$\mathbf{Z}^+ \to \mathbf{R}$ dadas por $f(n) = 5n, g(n) = n^2$, para $n \in \mathbf{Z}^+$.

Si $g = O\,(f)$, entonces en términos de cuantificadores, tendríamos que $\exists$c $\in \mathbf{R}^+ \exists \eta_0 \in \mathbf{Z}^+ \forall$n $\in \mathbf{Z}^+ [(n \geq \eta_0) \Rightarrow |g(n)| \leq c\ |f(n)|]$.

En consecuencia, para mostrar que $g \neq O\,(f)$, necesitamos verificar que
$\forall$c $\in \mathbf{R}^+ \forall \eta_0 \in \mathbf{Z}^+ \exists$n $\in \mathbf{Z}^+ [(n \geq \eta_0) \wedge (|g(n)| > c\ |f(n)|)]$.

Para esto, primero debemos darnos cuenta de que m y η_0 son arbitrarios, por lo que no tenemos control sobre sus valores. El único número sobre el cual tenemos control es el entero positivo n que seleccionamos. Ahora bien, independientemente de los valores de m y η_0, podemos seleccionar $n \in \mathbf{Z}^+$ tales que $n > \textit{máx}\{5c, \eta_0\}$. Entonces $n \geq \eta_0$ *(en realidad,* $n > \eta_0$*)* y $n > 5c \Longrightarrow n^2 > 5\ cn$ por lo que $|g(n)| = n^2 > 5cn = c\ |5n| = c\ |f(n)|$ y $g \neq O(f)$.

Para quienes prefieren el método de demostración por contradicción, presentamos un segundo punto de vista. Si $g = O(f)$, entonces tendríamos que

$$n^2 = |g(n)| \leq c\ |f(n)| = cn$$

para todo $n \geq \eta_0$, donde η_0 es algún entero fijo positivo y c es una *constante* (real). Pero entonces, de $n^2 \leq cn$ deducimos que $n \leq c$. Esto es imposible ya que $n \in (\mathbf{Z}^+)$ es una variable que puede crecer sin límite mientras que c es una constante.

Ejemplo 4:

a) Sean f, g: $\mathbf{Z}^+ \to \mathbf{R}$ dadas por $f(n) = 5n^2 + 3n + 1, g(n) = n^2$. Entonces $|f(n)| = |5n^2 + 3n + 1| = 5n^2 + 3n + 1 \leq 5n^2 + 3n^2 + n^2 = 9n^2 = 9|g(n)|$. De aquí tenemos que para todo $n \geq 1 (= \eta_0)$, $|f(n)| \leq c\ |g(n)|$ para cualquier $c \geq 9$, y $f = O(g)$. En este caso también podemos escribir $g = O\left(n^2\right)$.

Además, $|g(n)| = n^2 \leq 5n^2 \leq 5n^2 + 3n + 1 = |f(n)|$ para todo $n \geq 1$ por lo que $|g(n)| = c|f(n)|$ para cualquier $c \geq 1$ y para todo $n \geq \eta_0 \geq 1$. En consecuencia, $g = O(f)$. [De hecho $O(g) = O(f)$; es decir , cualquier función de $\mathbf{Z}^+$ a $\mathbf{R}$ dominada por f o por g está también dominada por la otra función].

b) Consideremos ahora f, g: $\mathbf{Z}^+ \to \mathbf{R}$ dadas por $f(n) = 3n^3 + 7n^2 - 4n + 2$ y $g(n) = n^3$. Aquí tenemos que $|f(n)| = |3n^3 + 7n^2 - 4n + 2| \leq |3n^3| + |7n^2| + |-4n| + |2| \leq 3n^3 + 7n^3 + 4n^3 + 2n^3 = 16n^3 = 16|g(n)|$, para todo $n \geq 1$. Así, si $c = 16$ y $\eta_0 = 1$, tenemos que f está dominada por g, y $f = O(g)$, o $f = O\left(n^3\right)$.

Como $7n - 4 > 0$ para todo $n \geq 1$, podemos escribir $n^3 \leq 3n^3 \leq 3n^3 + (7n - 4)n + 2$ si $n \geq 1$. Entonces $|g(n)| \leq |f(n)|$ para todo $n \geq 1$, y $g = O(f)$. [Como en la parte (a), también tenemos $O(f) = O(g) = O\left(n^3\right)$].

Finalmente generalizemos los resultados del ejemplo anterior como sigue:

Sea f: $\mathbf{Z}^+ \to \mathbf{R}$ tal que $f(n) = a_t n^t + a_{t-1} n^{t-1} + \cdots + a_2 n^2 + a_1 n + a_0$, para $a_t, a_{t-1}, \cdots, a_2, a_1, a_0 \in \mathbf{R}$, $a_t \neq 0$, $t \in \mathbf{N}$. Entonces $f(n)| = |a_t n^t + a_{t-1} n^{t-1} + \cdots + a_2 n^2 + a_1 n + a_0| \leq |a_t n^t| + |a_{t-1} n^{t-1}| + \cdots + |a_2 n^2| + |a_1 n| + |a_0| = |a_t| n^t + |a_{t-1}| n^{t-1} + \cdots + |a_2| n^2 + |a_1| n + |a_0| \leq |a_t| n^t + |a_{t-1}| n^t + \cdots + |a_2| n^t + |a_1| n^t + |a_0| n^t = |a_t| + |a_{t-1}| + \cdots + |a_2| + |a_1| + |a_0|) n^t$.

En la definición anterior, sean $c = (|a_t| + |a_{t-1}| + \cdots + |a_2| + |a_1| + |a_0|)$ y $\eta_0 = 1$, y sea g: $\mathbf{Z}^+ \to \mathbf{R}$ dada por $g(n) = n^t$. Entonces $|f(n)| \leq c\ |g(n)|$ para todo $n \geq \eta_0$, por lo que f

Cuadro 4.2: Funciones de complejidad más habituales

Forma O mayúscula	Nombre
O(1)	Constante
$O(\log_2 n)$	Logarítmica
$O(\mathrm{n})$	Lineal
$O(n \log_2 n)$	$n \log_2 n$
$O(n^2)$	Cuadrática
$O(n^3)$	Cúbica
$O(n^m)$, $m= 0, 1, 2, 3, \cdots$	Polinomial
$O(c^n)$, $c>= 1$	Exponencial
$O(\mathrm{n!})$	Factorial

está dominada por g, o $f = O\left(n^t\right)$.

También es cierto que $g = O\left(f\right)$ y que $O\left(f\right) = O\left(g\right) = O\left(n^t\right)$.

Ordenes de complejidad

La familia $O\left(f(n)\right)$ define un *Orden de Complejidad*. Elejiremos como representante de este *Orden* de *Complejidad* a la función *f(n)* más sencilla perteneciente a esta familia.

Las funciones de complejidad más habituales en las cuales el único factor del que dependen es del tamaño de la muestra de entrada n, ordenados de mayor a menor eficiencia se muestran en el Cuadro (4.2)

Se identifica una *Jerarquía de Ordenes de Complejidad* que coincide con el orden de la tabla mostrada, jerarquía en el sentido de que cada orden de complejidad inferior tiene a las superiores como subconjuntos.

4.3.2. Ω-notación

Definición 9 *Para una función g(n) dada, denotamos por* $\Omega\left(g\right)$ *al conjunto de funciones* $\Omega\left(g\right)$=$\{$f(n): existen constantes positivas c y η_0 tal que $0 \leq cg(n) \leq f(n)$ para todo $n \geq \eta_0\}$

La idea intuitiva sobre Ω-notación nos dice que para todos los valores de n a la derecha de η_0, el valor de *f(n)* esta en o por encima de *cg(n)*.

4.3.3. Θ-notación

Definición 10 *Dada una función g(n), denotamos por* $\Theta\left(g\right)$ *al conjunto de funciones* $\Theta\left(g\right)$=$\{$f(n):existen constantes positivas c_1, c_2, y η_0 tales que $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ para todo $n \geq \eta_0\}$

Cuadro 4.3: Tiempos para un algoritmo de complejidad exponencial 2^n

n	*Tiempo*
1	1 diezmilésima de segundo
10	$\approx$ 1 décima de segundo
20	$\approx$ 2 minutos
30	$\approx$ 1 día
40	mayor a tres años
50	mayor a tres milenios
100	$4{,}019693685^{19}$ milenios

Cuadro 4.4: Tiempos para un algoritmo de complejidad n^3

n	*Tiempo*
1	1 diezmilésima de segundo
10	$\approx$ 1 décima de segundo
20	$\approx$ 8 décimas de segundo
100	$\approx$ 1.7 minutos
200	$\approx$ 13.3 minutos
1000	$\approx$ 1 día

4.4. Importancia de la Eficiencia

¿Que utilidad tiene diseñar algoritmos eficientes si los computadores procesan información cada vez más rápido?

Bien: para demostrar la importancia de la elaboración de algoritmos eficientes, se plantea el siguiente problema:

Contamos con una computadora capaz de procesar datos en 10^{-4} segundos. En esta computadora se ejecuta un algoritmo que lee registros de una base de datos, dicho algoritmo tiene una complejidad exponencial 2^n, ¿Cuánto tiempo se tardará en procesar una entrada n de datos?. Ver Cuadro (4.3)

Ahora se tiene la misma computadora capaz de procesar datos en 10^{-4} segundos. Pero se ejecuta un algoritmo que hace el mismo trabajo antes citado; pero este algoritmo tiene una complejidad cúbica n^3, ¿Cuánto tiempo tardará en procesar una entrada n de datos?. Ver Cuadro (4.4)

Se puede concluir, que solo un algoritmo eficiente, con un orden de complejidad bajo, puede tratar grandes volúmenes de datos, se razona que un algoritmo es:

Muy eficiente si su complejidad es de orden *log n*.

Eficiente si su complejidad es de orden n^3.

Ineficiente si su complejidad es de orden 2^n.

4.5. El Teorema Maestro (The Master Theorem)

En capítulos posteriores analizaremos el *Algoritmo de Strassen* el cual nos ayudará a resolver el problema de multiplicar matrices de forma rápida. En general la idea de *Strassen* es dividir el problema inicial en subproblemas del mismo tipo del problema inicial, resolver estos subproblemas recursivamente, y combinar estas soluciones para obtener la solución del problema original.

Strassen utiliza esta idea para multiplicar dos matrices y obtiene que requiere solo 7 multiplicaciones recursivas de $\frac{n}{2}$ x $\frac{n}{2}$ matrices y 18 adiciones y sustracciones. Puesto que las operaciones aditivas tienen costo $\Theta(n^2)$, el costo del Algoritmo de Strassen quedará expresado como: T(n)= 7T($\frac{n}{2}$)+*f(n)*, donde *f(n)*= $\Theta(n^2)$.

Esta relación es de la forma *T(n)*= $aT(\frac{n}{b})+f(n)$ donde a≥ 1 y $b > 1$.

Aquí:

- a es el número de llamadas recursivas.
- $\frac{1}{b}$ denota la porción del problema original representado por cada subproblema,
- $f(n)$ es el costo de dividir el problema, más el costo de combinar la solución.

Dada una relación de esta forma es posible determinar un límite asintótico, por lo tanto es posible determinar un límite asintótico para T(n)= 7T($\frac{n}{2}$)+*f(n)*, donde *f(n)*= $\Theta(n^2)$, (tiempo que requiere el Algoritmo de Strassen para multiplicar matrices).

Para ello es necesario enunciar uno de los Teoremas principales en el estudio del Análisis de Algoritmos, *El Teorema Maestro*, el cual proporciona una solución en términos asintoticos para las relaciones de recursión de los tipos que ocurren en la práctica. Sin embargo, no todas las relaciones de recursión se pueden solucionar con el uso del *Teorema Maestro.*

Teorema 1 (Teorema Maestro) *Sean* $a\geq 1$ *y* $b > 1$ *constantes, sea* $f(n)$ *una función y* $T(n)$ *definida sobre los enteros no negativos. Entonces* T(n)= $aT(\frac{n}{b})+f(n)$ *puede ser limitada asintoticamente como sigue:*

- *Si* $f(n) = O\left(n^{log_b a-\epsilon}\right)$ *para alguna constante* $\epsilon > 0$, *entonces* $T(n) = \Theta\left(n^{log_b a}\right)$.
- *Si* $f(n) = \Theta\left(n^{log_b a}\right)$, *entonces* $T(n) = \Theta\left(n^{log_b a} \lg n\right)$.
- *Si* $f(n) = \Omega\left(n^{log_b a+\epsilon}\right)$, *para alguna constante* $\epsilon > 0$, *y si* $af\left(\frac{n}{b}\right) \leq cf(n)$ *para alguna constante* $c < 1$ *y para n suficientemente grande, entonces* $T(n) = \Theta(f(n))$.

4.5.1. Aplicando El Teorema Maestro

Caso 1: Tiempo que requiere el Algoritmo de Strassen para multiplicar matrices

Si es verdad que:

$f(n) = O\left(n^{log_b a-\epsilon}\right)$ para alguna constante $\epsilon > 0$ se sigue que, $T(n) = \Theta\left(n^{log_b a}\right)$

Ejemplo:

Sea T(n)= 7T($\frac{n}{2}$)+*f(n)*, donde *f(n)*= $\Theta(n^2)$, el tiempo que requiere Strassen para multiplicar matrices.

Como se puede ver en la expresión anterior las variables *a, b* toman los siguientes valores: a=7, b= 2.

Ahora tenemos que verificar si se cumple la condición del primer caso, es decir que $f(n) = O\left(n^{log_b a-\epsilon}\right)$ para alguna constante $\epsilon > 0$. Subtituyendo los valores de a y b, calculando $log_b a = log_2 7 = 2.8073...$, proponiendo que $\epsilon = 0.80735...$ y tomando en cuenta el hecho de que $\Theta(n) \subseteq O(n)$, podemos concluir que $f(n) = O\left(n^2\right)$.

Puesto que se cumple la condición del primer caso de *El Teorema Maestro* y substituyendo los valores de a y b se concluye que:

$$T(n) = \Theta\left(n^{2.8073...}\right)$$

El capítulo 6, lo dedicaremos al estudio del Algoritmo de Strassen y será hasta entonces cuando haremos algunos comentarios sobre este Algoritmo.

Por el momento solo mencionaremos que se conocen algoritmos mejores cuya cota superior es $O\left(n^{2.376}\right)$. A continuación presentaremos los Algoritmos que han mejorado el tiempo de Strassen.

- Algoritmo Clásico $\longrightarrow O\left(n^3\right)$
- Algoritmo *V. Strassen* (1969)$\longrightarrow O\left(n^{2.8073...}\right)$
- Algoritmo V. Pan (1984) $\longrightarrow O\left(n^{2.795}\right)$
- Algoritmo D. Coppersmith y S. Winograd (1990) $\longrightarrow O\left(n^{2.376}\right)$

Caso 2

Si es verdad que $f(n) = \Theta\left(n^{log_b a}\right)$, se sigue que $T(n) = \Theta\left(n^{log_b a}\log(n)\right)$.

Ejemplo 2

Sea $T(n)= 2T(\frac{n}{2})+10n$. Como se puede ver en en la fórmula las variables a,b y $f(n)$ tienen los siguientes valores: $a= 2$, $b= 2$, $f(n) = 10n$. Ahora, calculemos $log_b a = log_2 2 = 1$. Puesto que $f(n) = 10n = \Theta(n) = \Theta\left(n^1\right) = \Theta\left(n^{log_2 2}\right)$, se cumple con la condición del segundo caso de *El Teorema Maestro* obteniéndose como resultado la conclusión:

$$T(n) = \Theta\left(n^{log_b a} \log(n)\right)$$

insertando los valores de arriba se concluye que:

$$T(n) = \Theta(n \log(n))$$

Caso 3

Si es verdad que $f(n) = \Omega\left(n^{log_b a+\epsilon}\right)$, para alguna constante $\epsilon > 0$, y si es también verdad que $af\left(\frac{n}{b}\right) \leq cf(n)$ para alguna constante $c < 1$ y para n suficientemente grande. Se sigue que:

$$T(n) = \Theta(f(n))$$

.

Ejemplo 3:

$$T(n) = 2T(\frac{n}{2}) + n^2$$

Como se puede ver en en la fórmula las variables a, b y $f(n)$ tienen los siguientes valores: $a= 2$, $b= 2$ y $f(n) =n^2$. Ahora, calculemos $log_b a = log_2 2 = 1$.

Veamos que $f(n) = \Omega\left(n^{log_b a+\epsilon}\right)$.

Si insertamos los valores de arriba y eligiendo $\epsilon = 1$, puesto que $f(n) =n^2=\Omega\left(n^2\right)$, se tiene que $f(n) =n^2=\Omega\left(n^2\right) = \Omega\left(n^{log_2 2+1}\right)$.

Ahora falta comprobar que la segunda condición se cumple:

$$af\left(\frac{n}{b}\right) \leq cf(n)$$

Insertando los valores de arriba, se tiene que $2\left(\frac{n}{2}\right)^2 \leq cn^2 \Leftrightarrow \frac{1}{2}n^2 \leq cn^2$.

Si se elije $c= \frac{1}{2}$, es verdad que $\frac{1}{2}n^2 \leq \frac{1}{2}n^2 \ \forall \ n \geq 1$.

Puesto que esta ecuación cumple con las condiciones del tercer caso de *El Teorema Maestro* si se substituyen una vez más los valores de arriba, se tiene:

$$T(n) = \Theta\left(n^2\right).$$

Como se pudo constatar a través de los ejemplos anteriores la aplicación de *El Teorema Maestro* es muy poderosa pues podemos conocer la complejidad exacta en pocas lineas al aplicarlo.

Capítulo 5

Programación dinámica

Con este capítulo empezaremos a desarrollar el tema principal de este trabajo. Aquí al analizar, tratar de comprender y resolver el problema de multiplicar una cadena de matrices mostraremos la naturaleza vérsatil de uno de los métodos más eficientes, basado en la recursividad, el cual resuelve problemas de optimización que surgen en muchas situaciones diferentes de la matemática. Este método es conocido como **Método de Programación dinámica**. Además desarrollaremos este método de optimización en forma algorítmica para facilitar su implementación en una computadora.

5.1. El problema de multiplicar una cadena de matrices

En esta sección examinaremos el problema de multiplicar una cadena de matrices el cual queda expresado de la siguiente manera: Dada una sucesión (o cadena) de matrices $\langle A_1, A_2, \cdots, A_n \rangle$ de tamaño n las cuales serán multiplicadas, calcular el producto:

$$A_1 A_2 \cdots A_n \tag{5.1}$$

Antes de intentar resolver este problema sería adecuado recordar que existe una propiedad asociativa para la multiplicación de matrices, es decir la expresión (5.1) se puede asociar de distintas maneras y todas estas asociaciones diferentes entre sí nos devolverán el mismo valor.

Ahora que hemos recordado esta propiedad asociativa de las matrices, una pregunta natural que surgiría sería la siguiente: ¿Como puedo asociar esta expresión si en este texto o en otros textos relacionados con el estudio de matrices cuando se hace referencia al producto entre matrices sabemos ya sea, por la definición o un algoritmo (bien definido) como calcular el producto de dos matrices únicamente?.

Es cierto que en general no existe un algoritmo que calcule el producto de tres, cuatro o más matrices por lo tanto para evaluar la expresión (5.1) contamos únicamente con la definición sobre el producto de dos matrices. Pero veamos de que manera.

Antes de comenzar a calcular el producto $A_1 A_2 \cdots A_n$ intentemos resolver un caso más sencillo en donde la cadena de matrices sea $\langle A_1, A_2, A_3 \rangle$, por lo tanto ¿Para calcular este

producto de tres matrices el saber como multiplicar dos matrices me permitirá calcular este producto $A_1A_2A_3$?.

La respuesta es sí y la forma natural en la que se asociaría la expresión $A_1A_2A_3$ para obtener el producto, sería de tal manera que el algoritmo o definición para multiplicar pares de matrices fuera utilizado como una subrutina y quedará especificado mediante paréntesis. Por lo tanto de ahora en adelante las asociaciones serán llamadas parentizaciones.

Así, para calcular el producto $A_1A_2A_3$ la manera natural en que se perentizaría la expresión $A_1A_2A_3$ sería: $(A_1(A_2A_3))$ y $((A_1A_2)A_3)$.

Puesto que el número de matrices en la cadena es muy pequeño es fácil verificar que son todas las parentizaciones posibles. Ahora que hemos obtenido todas las parentizaciones una pregunta natural sería ¿cuál de todas estas parentizaciones es la maś rápida de calcularse?.

Recordemos que en el capítulo dedicado al estudio de matrices consideramos el problema de multiplicar el producto de tres matrices A_1, A_2, A_3 cuyos tamaños eran 10 x 10, 000, 10, 000 x 5 y 5 x 5, 000, respectivamente. Y concluimos que si nosotros multiplicabamos estas matrices de acuerdo a la asociación $((A_1A_2)A_3)$ el número de multiplicaciones por escalar implicadas en total serían 750, 000.

Sin embargo de acuerdo a la asociación $(A_1(A_2A_3))$ el número de multiplicaciones escalares por desarrollarse serían 750, 000, 000 en total y que por lo tanto , calcular el producto de acuerdo a la primera asociación sería 1, 000 veces más rápido.

Así, para conocer que parentización me permite obtener el producto de matrices de forma rápida una vez que se conoce el tamaño de las matrices, se calcula el número de multiplicaciones escalares que se requiere para cada parentización, se comparan entre sí y se selecciona la parentización que implica el menor número de multiplicaciones escalares.

Analizemos otro caso, ahora si el número de elementos de la cadena aumenta y la cadena es $\langle A_1, A_2, A_3, A_4\rangle$, el producto $A_1A_2A_3A_4$ podría ser parentizado en cinco formas distintas:

1) $(A_1(A_2(A_3A_4)))$

2) $(A_1((A_2A_3)A_4))$

3) $((A_1A_2)(A_3A_4))$

4) $((A_1(A_2A_3))A_4)$

5) $(((A_1A_2)A_3)A_4)$

Ahora solo falta conocer el tamaño de las matrices para saber cual es el número de multiplicaciones escalares que se necesitan efectuar en cada parentización y seleccionar aquella parentización que requiera el menor número de multiplicaciones escalares.

Pero, ¿serán estas las únicas formas en que podemos parentizar este producto, o habrá más?.

Hasta ahora, en este caso como en el caso anterior, puesto que el número de matrices implicadas en los productos entre matrices ha sido pequeño, podemos verificar que efectivamente estas son todas las posibles formas distintas en que se pueden parentizar estos productos, sin embargo, al aumentar el número de matrices en la cadena, que serán multiplicadas, posiblemente el número de parentizaciones aumente también. Por lo tanto, el trabajo de conocer todas las posibles parentizaciones, verificar que efectivamente son todas las posibilidades y dado los tamaños de las matrices calcular el número de multiplicaciones escalares para cada perentización, seleccionando de entre todas ellas la parentización que implique el menor número de multiplicaciones escalares puede ser muy tardado.

Pero antes de dar una conclusión final sobre los comentarios anteriores es necesario saber si el número de parentizaciónes aumenta, si aumenta el número de elementos en la cadena de matrices. Si observamos los ejemplos anteriores, para la cadena $\langle A_1, A_2, A_3\rangle$, el producto $A_1A_2A_3$ se parentizó de dos formas únicas distintas, al aumentar el número de elementos de la cadena a $\langle A_1, A_2, A_3, A_4\rangle$ el producto $A_1A_2A_3A_4$ se parentizó en cinco únicas formas distintas. Así nuestra intuición nos dice que efectivamente al aumentar el número de elementos en la cadena de matrices el número de formas distintas en que puede parentizarse el producto de dichas matrices también aumenta.

Ahora bien, existe una herramienta que nos ayuda a conocer cuántas y cuales son exactamente las distintas formas en que podemos parentizar un producto y dejar expresada cada distinta asociación de tal manera que el algoritmo para multiplicar pares de matrices se utilice como una subrutina en la parentización. Esta herramienta son los árboles binarios ordenados con raíz que estudiamos en capítulos anteriores.

Esta nueva aplicación de los árboles binarios con raíz consiste en asociar el producto de matrices que se quiere calcular, con un árbol binario con raíz, cuyo número total de vértices será igual al número de multiplicaciones entre matrices que se requieran para obtener el resultado, el cual depende del número de matrices que serán multiplicadas y es independiente de como las matrices son asociadas. Por ejemplo, para calcular el producto A_1A_2 se requiere una sola multiplicación de matrices, para calcular $A_1A_2A_3$, se requieren dos multiplicaciones entre matrices, por lo tanto para multiplicar $A_1A_2\cdots A_n$, se requieren $(n-1)$ multiplicaciones entre matrices. Así estos productos A_1A_2, $A_1A_2A_3$, $A_1A_2\cdots A_n$, serán asociados con árboles de 1, 2, y $(n-1)$ vértices, respectivamente.

Después la información que contenga cada árbol será interpretada en el producto y dicha información nos permitirá construir una parentización para el producto. Las parentizaciones comienzan a construirse a partir del nodo principal o raíz el cual además, representará la última multiplicación (la cual quedará indicada por los paréntesis más externos), cada uno de los demás vértice que aparecen en un árbol representarán un producto entre dos matrices y se denotará entre paréntesis.

Para ello, es necesario asignar a las matrices un subíndice lo cual nos permitirá mantener cierto orden en las ideas.

Así pues comencemos a aplicar esta idea. Dada la cadena $\langle A_0, A_1\rangle$ el producto A_0A_1 es asociado con un árbol binario con raíz cuyo número total de vértices n= 1, puesto que se requiere un sólo producto entre matrices para calcular la expresión A_0A_1. Ya que nosotros

Figura 5.1: Primer árbol binario con raíz de dos vértices

Figura 5.2: Segundo árbol binario con raíz de dos vértices

sabemos que existe un único árbol binario con raíz de un solo vértice representado por • (la raíz), al ser interpretada la información de este árbol su único vértice indica que hay un sólo producto entre matrices el cual queda expresado entre paréntesis como (A_0A_1). Entonces esta es la única parentización para este producto

Ahora veamos que sucede para la cadena $\langle A_0, A_1, A_2\rangle$, puesto que hay tres matrices implicadas en el producto $A_0A_1A_2$, el número de multiplicaciones entre matrices que se requiere para obtener el producto es dos, por lo tanto la expresión $A_0A_1A_2$ será asociado con un árbol binario con raíz de n= 2 vértices. Puesto que existen sólo dos árboles binarios con raíz de dos vértices, los cuales son [Figura (5.1)] y [Figura (5.2)], cada árbol al ser interpretado nos permitirá costruir dos distintas parentizaciones para el producto $A_0A_1A_2$.

Para interpretar la información de estos árboles binarios y para los árboles binarios en general, estos serán recorridos de manera convencional, es decir, de arriba hacia abajo y de izquierda a derecha. Por lo tanto las parentizaciones comienzan a construirse a partir del nodo principal o raíz el cual además, representará la última multiplicación (la cual quedará indicada por los paréntesis más externos) que deberá efectuarse al construir la parentización del producto, los vértices que aparecen en el segundo nivel del árbol representarán las penúltimas multiplicaciones entre matrices que deberán efectuarse y estas quedarán indicadas dentro del producto que quedo señalado por los paréntesis más externos, y así sucesivamente hasta llegar a los vértices que aparecen en el último nivel en el árbol, los cuales representarán finalmente la primera multiplicación entre matrices la cual quedará representada por paréntesis (los paréntesis más internos) dentro de los productos que representan los paréntesis denotados por los productos anteriores.

Además por la estructura de los árboles binarios con raíz podemos hablar de ramas izquierdas y ramas derechas, aquí los vértices que aparecen en las ramas izquierdas o derechas serán interpretados como productos izquierdos y productos derechos al costruir la perentización.

Por lo tanto, el árbol binario con raíz de la [Figura (5.1)] será interpretado de la siguiente forma:

El nodo principal representará a la última multiplicación y se indicará por los parentesis más externos, así comienzan a colocarse los paréntesis en el producto $A_0A_1A_2$ quedando este nodo representado como $(A_0A_1A_2)$ en la parentización. Ahora si comenzamos a descender por el árbol nos encontramos con una rama derecha la cual contiene un vértice el cual será interpretado como un producto derecho dentro de la expresión $(A_0A_1A_2)$ quedando indicado como $(A_0(A_1A_2))$ siendo esta una parentización del producto $A_0A_1A_2$.

Ahora, el árbol binario con raíz de la [Figura (5.2)] se interpreta como sigue:

Aquí la raíz representa el último producto por realizarse en la parentización el cual se expresa como $(A_0A_1A_2)$ en el producto $A_0A_1A_2$. Ahora al descender por el árbol viajamos ahora por una rama izquierda por lo tanto se interpreta como un producto izquierdo en la expresión $(A_0A_1A_2)$ siendo $((A_0A_1)A_2)$ su representación.

En conclusión puesto que existen dos únicos árboles binarios con raíz de dos vértices, existen dos únicas formas distintas de parentizar el producto $A_0A_1A_2$ cada una correspondiendo a la información en cada árbol.

Continuemos, si la cadena se extiende ha $\langle A_0, A_1, A_2, A_3\rangle$, el producto $A_0A_1A_2A_3$ requiere de tres multiplicaciones entre matrices para poder calcularse, por lo tanto es asociado con un árbol binario de n= 3 vértices. Pero existen cinco formas distintas de representar un árbol binario de n= 3 vértices. Por lo tanto habrá cinco formas distintas de parentizar el producto $A_1A_2A_3A_4$. Vamos a construirlas.

Los cinco árboles binarios distintos de n= 3 vértices son: [Figura (5.3)]. La información del primer árbol será interpreta da la siguiente manera: El nodo principal es representado en el producto como $(A_0A_1A_2A_3)$, observemos que al comenzar a viajar por el árbol descendemos primero por una rama derecha y que a partir del nodo de esta rama nace una subrama derecha con su respectivo nodo, esto quiere decir que dentro del producto derecho correspondiente a la primer rama por la que viajamos, se efectuaá dentro de él otro producto derecho. Así el primer producto derecho debe contener el número de matrices necesarias para que en el interior de él pueda efectuarse otro producto. Por lo tanto, el primer producto derecho correspondiente a la rama derecha por la que descendimos se representa por $(A_0(A_1A_2A_3))$ y el segundo producto derecho correspondiente a la subrama derecha que se debe ejecutar dentro de él queda representado por $(A_0(A_1(A_2A_3)))$ siendo esta una parentización del producto $A_0A_1A_2A_3$.

La segunda parentización correspondiente al segundo árbol se construye a partir del nodo raíz el cual se interpreta asi $(A_0A_1A_2A_3)$, la rama derecha indica un producto derecho pero a partir de esta rama derecha nace una subrama izquierda por lo tanto como en el caso anterior es importante considerar el número de las matrices que se van a tomar en cuenta al indicar los productos si es que se deben efectuar otros productos en el interior de estos. Por lo tanto la rama derecha se traduce como $(A_0(A_1A_2A_3))$ y la subrama izquierda como $(A_0((A_1A_2)A_3))$.

Al interpretar el tercer árbol el nodo principal se indica en el producto como $(A_0A_1A_2A_3)$, ahora observese que a partir del nodo principal surgen dos ramas una izquierda y otra derecha

Figura 5.3: Árboles binarios con raíz para n= 3

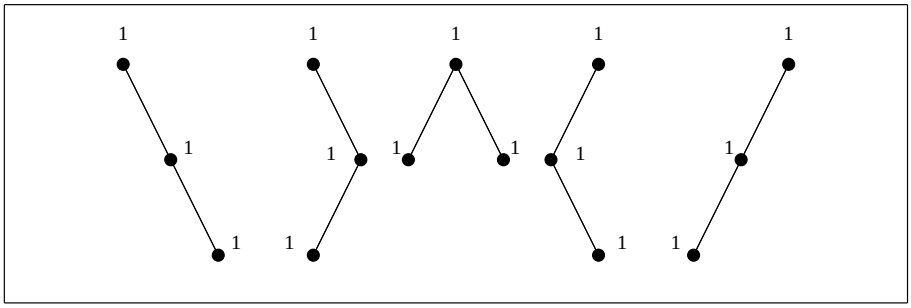
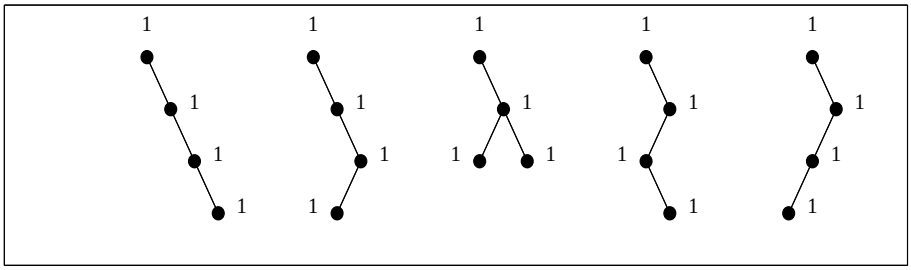


Figura 5.4: Árboles binarios con raíz para n= 4



que se encuantran en el mismo nivel del árbol cuando viajamos atravez de él, esto quiere decir que se van a efectuar dos multiplicaciones una derecha y una izquierda dentro de la expresión y que hay solo cuatro matrices para hacer esto. Por lo tanto la manera correcta de interpretar esta información es la siguiente $((A_0A_1)(A_2A_3))$ siendo esta una tercera forma de parentizar el producto $A_0A_1A_2A_3$.

Para el cuarto y quinto árbol las interpretaciones son similares a las descritas anteriormente. Así, la parentización correspondiente al cuarto árbol es $((A_1(A_2A_3))A_4)$ y para el quinto árbol la parentización es $(((A_0A_1)A_2)A_3)$.

Por lo tanto, son cinco las distintas parentizaciones en total para el producto $A_0A_1A_2A_3$.

Si continuamos en el caso en que la cadena ahora sea $\langle A_0, A_1, A_2, A_3, A_4\rangle$, el producto $A_0A_1A_2A_3A_4$ será asociado con un árbol binario de n= 4 vértices. Sin embargo, sabemos que existen 14 árboles binarios diferentes cuyo número total de vértices es n= 4. Estos se presentan a continuación en las [Figuras (5.4), (5.5) y (5.6)], junto con las 14 distintas parentizaciones que se construyen a partir de la información proporcionada por cada uno de estos árboles.

A continuación presentamos las primeras 9 parentizaciones correspondientes a las [Figuras (5.4), (5.5)]:

Figura 5.5: Más árboles binarios con raíz para n= 4

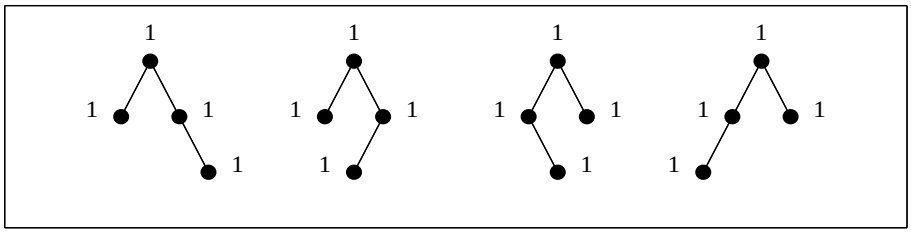
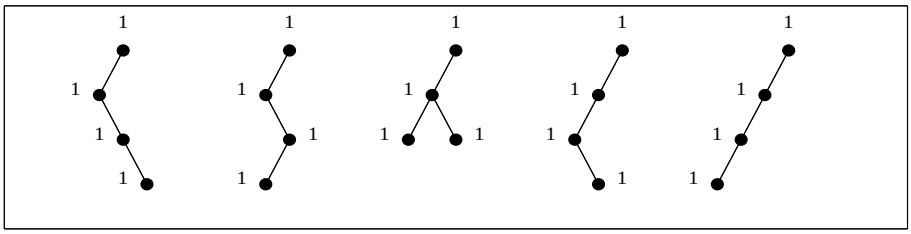


Figura 5.6: Últimos árboles binarios con raíz para n= 4



1) $(A_0(A_1(A_2(A_3A_4))))$

2) $(A_0(A_1((A_2A_3)A_4)))$

3) $(A_0((A_1A_2)(A_3A_4)))$

4) $(A_0((A_1(A_2A_3))A_4))$

5) $(A_0(((A_1A_2)A_3)A_4))$

6) $((A_0A_1)(A_2(A_3A_4)))$

7) $((A_0A_1)((A_2A_3)A_4))$

8) $((A_0(A_1A_2))(A_3A_4))$

9) $(((A_0A_1)A_2)(A_3A_4))$

Parentizaciones correspondientes a las [Figuras (5.6)]:

10) $((A_0(A_1(A_2A_3)))A_4)$

11) $((A_0((A_1A_2)A_3))A_4)$

12) $(((A_0A_1)(A_2A_3))A_4)$

13) $(((A_0(A_1A_2))A_3)A_4)$

14) $((((A_0A_1)A_2)A_3)A_4)$

Finalmente, expondremos el caso para la cadena $\langle A_0, A_1, A_2, A_3, A_4, A_5\rangle$ donde el producto $A_0A_1A_2A_3A_4A_5$ será asociado ahora con un árbol binario cuyo número total de vértices es n=5, para este caso existen 42 árboles binarios diferentes, los cuales al ser interpretados nos permitirán construir 42 parentizaciones distintas para este producto.

Ya no es necesario mostrar ni costruir sus correspondientes parentizaciones pues nuestro objetivo principal era convencernos sobre como puede aumentar el número de parentizaciones de un producto al aumentar el número de elementos en la cadena a partir de la cual se genera el producto. Y observar estos ejemplos, nos ha permitido obtener una idea intuitiva sobre este hecho.

Además, en los ejemplos, puesto que el número de matrices en las cadenas y sus respectivos productos involucran un número pequeño de matrices, no es tarea difícil verificar que en efecto estas son todas las parentizaciones posibles para cada uno de los casos expuestos. Aún para la cadena $\langle A_0, A_1, A_2, A_3, A_4, A_5\rangle$, el producto cuyo total de parentizaciones distintas es 42, calcular el número de multiplicaciones escalares para cada una de estas parentizaciones, una vez que se conozca el tamaño de cada matriz y elegir el número más pequeño de entre todos estos, sería una tarea que podría realizarse en poco tiempo.

Sin embargo, no olvidemos que nuestra misión es resolver el problema de multiplicar una cadena de matrices $\langle A_0, A_1, \cdots, A_n\rangle$ de tamaño n. Para ello siguiendo el método necesitamos primero asociar este producto con un árbol binario cuyo número total de vértices es n. Pero haciendo uso de la expresión $b_n = \frac{1}{(n+1)}\binom{2n}{n}$, la cual me permite calcular para $n \geq 0$, el número b_n de árboles binarios con raíz de n vértices, concluimos que en total el número de formas para parentizar el producto $A_0A_1\cdots A_n$ es $\frac{1}{(n+1)}\binom{2n}{n}$. Saber que $\frac{1}{(n+1)}\binom{2n}{n} = \Omega(\frac{4^n}{n^{\frac{3}{2}}})$ nos advierte sobre el hecho de que la expresión $\frac{1}{(n+1)}\binom{2n}{n}$ crece exponencialmente.

Hasta aquí, para conocer que parentización me permite obtener el producto de matrices de forma rápida una vez que se conoce el tamaño de las matrices, es construir todas las parentizaciones posibles después, calcular el número de multiplicaciones escalares que se requiere para cada una de estas parentizaciones, compararlas entre sí y seleccionar la parentización que implica el menor número de multiplicaciones escalares. Este procedimiento es un ejemplo donde se usa la técnica que llamamos, apropiadamente, método exhaustivo o de fuerza bruta. El uso de este método es razonable cuando trabajamos con un universo pequeño, para valores grandes de n, es la peor estrategia para determinar la parentización óptima de una cadena de matrices.

Cabe hacer el siguiente comentario, el número de árboles binarios distintos de n= 1, 2, 3, 4, 5 vértices a los que se hacia alusión en los ejemplos anteriores se pudieron haber calculado directamente a través de los números de Catalan, puesto que estos números ya habían sido estudiados en capítulos anteriores, sin embargo decidimos hacer uso de los árboles binarios permitiéndonos conocer una nueva aplicación y la forma en que fueron útiles en la construcción de las distintas formas para colocar paréntesis en los productos, a partir de la información que representaba cada árbol.

Ahora que nos hemos convencido que resolver el problema de multiplicar una cadena de matrices por el método de fuerza bruta no es un algoritmo eficiente, es momento de intentar resolver este problema mediante algún otro método el cual me permita obtener una parenti-

zación óptima de el producto $A_0A_1 \cdots A_n$ sin que el hacerlo me lleve mucho tiempo.

Pues bien, existe un método llamado método de *Programación dinámica* el cual puede ser aplicable para resolver este problema, puesto que la programación dinámica se aplica típicamente en los problemas de optimización. Además se dice que este método se basa en la idea general de recursividad es decir, resuelve problemas descomponiéndolos en subproblemas similares más pequeños y usando las soluciones parciales de estos subproblemas se llega a la respuesta final construyendo una solución del problema original, cada subproblema se resuelve sólo una vez y se guarda almacenándose esta respuesta evitando el trabajo de recalcularla cada vez que el problema es nuevamente encontrado.

Utilizemos como ejemplo a los *números de Fibonacci* los cuales nos servirán como guía para comprender con más detalle la idea de recursividad. Los *números de Fibonacci* se definen como

1) $F_0 = 0$, $F_1 = 1$; y

2) $F_n = F_{n-1} + F_{n-2}$, para $n \in \mathbb{Z}^+$ con $n \geq 2$

Supongamos que deseamos calcular F_5.

Puesto que $F_5 = F_4 + F_3$, para obtener este valor necesitamos calcular F_4 y F_3. Pero como $F_4 = F_3 + F_2$ tenemos primero que obtener los valores de F_3 y F_2. Ya que $F_3 = F_2 + F_1$, es necesario calcular F_2. Como $F_2 = F_1 + F_0 = 1 + 0 = 1$ (aquí ya no es necesario calcular F_1 y F_0 puesto que la condición inicial para la relación de recurrencia son los números F_1 y F_0), este valor lo substituimos de inmediato y obtenemos que $F_3 == F_2 + F_1 = 1 + 1 = 2$, ya con el valor de F_3 y F_2 calculamos $F_4 = F_3 + F_2 = 2 + 1 = 3$ y finalmente obtenemos que $F_5 = F_4 + F_3 = 3 + 2 = 5$.

Observemos que respetando de manera estricta la forma recursiva en que estos números fuerón definidos se intuye que hay un orden implícito en la idea de recursión.

Ahora, si los números de Fibonacci pudieran ser calculados mediante el método de programación dinámica se respetaría la idea recursiva en cierto sentido, sin embargo, por las características tan particulares de este método, el orden estricto implícito en la idea de recursión se vería en cierta forma "alterado" siendo menor el tiempo que nos llevaría calcular estos números. Veamos por que.

Puesto que $F_5 = F_4 + F_3$, para obtener este valor necesitamos calcular F_4 y F_3. Pero como $F_4 = F_3 + F_2$ tenemos primero que obtener los valores de F_3 y F_2. Ya que $F_3 = F_2 + F_1$, es necesario calcular F_2 únicamente ya que conocemos el valor de $F_1 = 1$, pues es una de las condiciones iniciales para la relación de recurrencia. Sustituimos de inmediato este valor no sólo en esta ecuación, sino en todas aquellas donde aparece $F_1 = 1$, quedando $F_3 = F_2 + 1$ y $F_2 = 1 + 0 = 1$ (aquí, también se substituye de inmediato $F_0 = 0$ y en todas las ecuaciones donde aparece), por lo tanto $F_3 = 1 + 1 = 2$. Ya con el valor de F_3 se substituye de forma automática en $F_4 = F_3 + 1 = 2 + 1 = 3$ y finalmente obtenemos que $F_5 = F_4 + 2 = 3 + 2 = 5$.

Como consecuencia de estos hechos observemos que si se pudiera resolver el problema de calcular los números de Fibonacci mediante este método, se agilizaría la obtención del

resultado final reduciendo nuestro trabajo en forma considerable puesto que no recalculamos los valores cada vez que aparecen sino una vez que los obtenemos los sustituimos de forma inmediata.

Si esta idea se implementara en una computadora mediante un algoritmo bien definido, los subproblemas serían resueltos sólo una vez y las respuestas serían guardadas en una tabla en memoria evitando la molestia de volverlos a calcular.

Una vez que hemos analizado las carecterísticas principales de este método es momento de conocer el desarrollo de un *algoritmo de programación dinámica* el cual puede ser dividido en una sucesión de cuatro pasos:

1) Caracterizar la estructura de una solución óptima.

2) Definir recursivamente el valor de una solución óptima.

3) Calcular el valor de una solución óptima en un "bottom-up fashion" (de abajo hacia arriba).

4) Construir una solución óptima de información calculada.

Dado el problema de multiplicar una cadena de matrices de modo que se realicen el menor número de multiplicaciones escalares, veremos a detalle sólo dos características clave que debe tener este problema para que la programación dinámica sea una técnica de solución viable:

1) Caracterizar la estructura de una parentización óptima.

2) Definir una solución recursiva.

Paso 1) Caracterizar la estructura de una parentización óptima.

Antes de comenzar a desarrollar este paso, por conveniencia, adoptaremos la notación $A_{i\cdots j}$ para la matriz que resulta de evaluar el producto $A_iA_{i+1}\cdots A_j$.

Para el problema de multiplicar una cadena de matrices, veamos la manera en que se genera una parentización óptima. Una parentización óptima se produce partiendo el producto $A_1A_2\cdots A_n$ en A_k y A_{k+1}, para algún entero k en el rango $1 \leq k < n$. Esto es, para algún valor de k, primero calculamos las matrices $A_{1\cdots k}$ y la matriz $A_{k+1\cdots n}$ y después multiplicamos ambas para obtener el producto final $A_{1\cdots n}$. Por lo tanto, el costo de obtener una parentización óptima, es el costo de calcular la matriz $A_{1\cdots k}$, más el costo de calcular $A_{k+1\cdots n}$, más el costo de multiplicar ambas.

Pero para calcular las matrices $A_{1\cdots k}$ y A_{k+1} es necesario parentizar de alguna forma los subproductos que representa cada matriz. Una manera de obtener las parentizaciones correspondientes a cada subproducto es partiendo estos subproductos en sub-subproductos y calculando las matrices correspondientes a estos sub-subproductos. Se vuelven a partir estos sub-subproductos en sub-subsubproductos y se calculan sus respectivas matrices. Así, se genera una subrutina la cual consiste en partir cada producto generando nuevos subproductos los cuales también serán partidos. Cada subproducto, sub-subproducto, sub-subsubproducto, etc. que se genera debe quedar indicado de alguna forma dentro de la parentización óptima

pues a partir de ellos se construye.

Con esta idea en mente sobre como se genera una parentización óptima del producto $A_1A_2\cdots A_n$, es necesario puntualizar una observación clave la cual nos permitirá entonces, caracterizar la estructura de una parentización óptima . Esta observación clave es que las parentizaciones de los subproductos, sub-subproductos, sub-subsubproductos, etc, que se van generando deben ser parentizaciones óptimas. Una pregunta natural que surgiría sería la siguiente: ¿Por que?, la respuesta es la siguiente, si existiera otro camino en que se parentizaran estos subproductos de tal manera que se obtuvieran de manera más rápida, sustituyendo esta parentización en la parentización ópima de $A_1A_2\cdots A_n$ podría producir otra parentización de $A_1A_2\cdots A_n$ cuya forma de obtener el producto $A_1A_2\cdots A_n$ fuera más rápida que la óptima generando una contradicción.

Por lo tanto, una solución óptima de el problema de multiplicar matices, se genera a partir de las soluciones óptimas de los subproblemas en que este fue dividido.

Paso 2) Definir una solución recursiva.

El segundo paso del algoritmo de programación dinámica aplicado al problema de multiplicar una cadena de matrices consiste en definir el valor de una solución óptima.

La idea es definir este valor en forma recursiva, es decir, el valor de una solución óptima se expresará en términos de otros resultados anteriores similares, para el problema de multiplicar una cadena de matrices, será a través de soluciones óptimas de subproblemas.

Para el problema de multiplicar una cadena de matrices, elegimos de manera natural como nuestros subproblemas el problema de determinar el número mínimo de multiplicaciones escalares que se necesitan para calcular el producto $A_iA_{i+1}\cdots A_j$, para $1 \leq i \leq j \leq n$.

Sea m[i,j] el número mínimo de multiplicaciones escalares que se requieren para calcular la matriz $A_{i\cdots j}$. Así, el número mínimo de multiplicaciones que se requiere para calcular $A_{1\cdots n}$ quedará expresado como m[1,n].

Podemos defir m[i,j] de manera recursiva como sigue:

1) Si $i = j$, la cadena consiste de una sola matriz $A_{i\cdots i} = A_i$, por lo tanto, para calcular este producto no se requieren multiplicaciones escalares.

   Entonces,
   $$m[i,i] = 0$$
   para
   $$i = 1, 2, \cdots n$$

2) Para calcular m[i,j] cuando $i < j$ hacemos uso de la estructura de una solución óptima analizada en el paso anterior, lo cual nos será muy benefico.

Recordemos que la parentización óptima del producto $A_iA_{i+1}\cdots A_j$ se genera partiendo este producto en A_k y A_{k+1} donde $1 \leq k < n$. Entonces m[i,j] es igual a el número mínimo de multiplicaciones escalares que se requieren para calcular los subproductos $A_{i\cdots k}$ y $A_{k+1\cdots j}$, más el número de multiplicaciones escalares que se requieren para multiplicar ambas matrices.

El número de multiplicaciones escalares que se requieren para multiplicar $A_{i\cdots k}$ y $A_{k+1\cdots j}$ lo obtendremos de la siguiente manera: supongamos que el tamaño de A_i es $p_{i-1}p_i$, por lo tanto, el tamaño de A_{i+1} será p_ip_{i+1}, luego el tamaño de A_{i+2} será entonces $p_{i+1}p_{i+2}$ y así, bajo esta suposición podremos conocer el tamaño de A_k el cual será $p_{k-1}p_k$. Esto nos permitiría establecer el tamaño de la matriz $A_{i\cdots k}$ el cual es $p_{i-1}p_k$.

Si continuamos calculando los tamaños para las matrices A_{k+1} el cual es p_kp_{k+1}, para la matriz A_{k+2} sería $p_{k+1}p_{k+2}$ y así sucesivamente, el tamaño de la matriz A_j sería $p_{j-1}p_j$ y el tamaño de la matriz $A_{k\cdots j}$ sería p_kp_j. Así al multiplicar las matrices $A_{i\cdots k}$ y $A_{k\cdots j}$ cuyos tamaños son $p_{i-1}p_k$ y p_kp_j respectivamente, el número total de multiplicaciones por escalar que se necesitarían efectuar serían en total $p_{i-1}p_kp_j$.

Por lo tanto una vez hecha esta observacion se obtiene que:

$$m[i,j] = m[i,k] + m[k,j] + p_{i-1}p_kp_j$$

Una observación que sería pertinente hacer es que la ecuación recursiva asume que nosotros conocemos el valor de k, lo cual no es cierto y que además existen únicamente j-i valores para k es decir $k = i, i+1, \cdots, j-1$. Por lo tanto la parentización óptima debe utilizar uno de estos valores para k, nosotros solo necesitamos checar cada uno de estos valores y encontrar el mejor en el sentido de que este valor para k me permite determinar donde puedo partir el producto $A_iA_{i+1}\cdots A_j$ para obtener una parentización óptima.

Ahora nuestra definición recursiva que me permite calcular el mínimo número de multiplicaciones escalares que se necesitan para calcular el producto $A_iA_{i+1}\cdots A_j$ se expresa como

$$m[i,j] = \begin{cases} 0 & \text{si } i = j \\ min_{1\leq k<j}\{m[i,k] + m[k+1,j] + p_{i-1}p_kp_j\} & \text{si } i < j \end{cases}$$

Los valores de los m[i,j] nos proporcionan los costos de las soluciones óptimas de los subproblemas.

Finalmente definamos s[i,j], como el valor de k, el cual parte el producto

$$A_iA_{i+1}\cdots A_j$$

para obtener una parentización óptima. Esto es, s[i,j] es igual al valor de k tal que $m[i,j] = m[i,k] + m[k+1,j] + p_{i-1}p_kp_j$.

Como lo mencionamos al inicio de este capítulo, proporcionamos un posible programa diseñado con las características de un algoritmo de programación dinámica que encuentra la parentización óptima. El siguiente codigo, asume que las matriz A_i tiene dimensiones $p_{i-1} * p_i$ para $i = 1, 2, \cdots, n$. Los datos de entrada es una sucesión $\langle p_0, p_1, p_2, \cdots, p_n \rangle$ de números, cada número indica la primera componente de la dimension de cada una de las matrices que se van a multiplicar. El procedimiento usa una tabla auxiliar m$[1 \cdots n, 1 \cdots n]$ para almacenar los m[i,j] costos y una tabla auxiliar s$[1 \cdots n, 1 \cdots n]$, donde se anota que índice k logra el costo óptimo al calcular m[i,j], al generar estas tablas se está haciendo el cálculo de costos óptimos el cual sería el equivalente al paso tres de las cracterísticas principales de un algoritmo de Programación dinámica.

El cuarto paso, el cual equivale a construir una soluión óptima para nuestro problema, consiste en construir una solución óptima a partir de información calculada. Este paso también queda desarrollado dentro del algoritmo.

Es importante mencionar que este método es mucho más eficiente que el método de *fuerza bruta* del cual ya hemos hablado en capítulos anteriores.

```
#include <iostream>
#include <limits>

using namespace std;

typedef int* intP;

intP* M;
intP* S;
int n;
const int infinity = numeric_limits<int>::max();

void CreateMandS()
{
  M = new intP[n];
  for( int i=0; i<n; i++ )
    M[i] = new int[n];
  S = new intP[n];
  for( int i=0; i<n; i++ )
    S[i] = new int[n];
  for( int row=0; row<n; row++ )
    for( int col=0; col<n; col++ )
      M[row][col]=-1;
  for( int row=0; row<n; row++ )
    for( int col=0; col<n; col++ )
      S[row][col]=-1;
}

void PrintMandS()
```

```
{
  cout<<"M = "<<endl;
  for( int row=0; row<n; row++ )
    {
    for( int col=0; col<n; col++ )
      cout<<M[row][col]<<" ";
    cout<<endl;
    }
  cout<<"S = "<<endl;
  for( int row=0; row<n; row++ )
    {
    for( int col=0; col<n; col++ )
      cout<<S[row][col]<<" ";
    cout<<endl;
    }
}

int MultiplicationCost( int i, int k, int j , int* p )
{
  return p[i]*p[k+1]*p[j+1];
}

// Introduction to Algorithms
// Cormen, Leiserson, Rivest
// MIT Presss
// page 306

void MatrixChainOrder( int* p )
{
  for( int i=0; i<n; i++ )
    M[i][i]=0;
  for( int l=2; l<=n; l++)
    for( int i=0; i<(n-l+1); i++ )
    {
      int j=i+l-1;
      M[i][j]=infinity;
      for( int k=i; k<=j-1; k++ )
{
  int q = M[i][k]+M[k+1][j]+MultiplicationCost(i,k,j,p);
  if( q < M[i][j] )
    {
       M[i][j]=q;
       S[i][j]=k;
    }
}
    }
}
```

```
// Introduction to Algorithms
// Cormen, Leiserson, Rivest
// MIT Presss
// Exercise 16.1-2
// page 308

void PrintOptimalParens( int i, int j, int** s, int n )
{
  if( j > i )
    {
      cout<<"(";
      PrintOptimalParens( i, s[i][j], s, n);
      cout<<" ";
      PrintOptimalParens( s[i][j]+1, j, s, n);
      cout<<")";
    }
  else
    cout<<"A["<<i<<"]";
}

int main()
{
  cout<<endl;
  cout<<"How many matrices do you have : ";
  cin>>n;
  CreateMandS();
  int* p = new int[n+1];
  for( int i=0; i<n+1; i++ )
    {
      cout<<"p["<<i<<"] = ";
      cin>>p[i];
    }
  cout<<endl;
  MatrixChainOrder( p );
  PrintMandS();
  cout<<endl;
  PrintOptimalParens( 0, n-1, S, n );
  cout<<endl<<endl;
}
```

Finalmente, recordemos que al inicio de este capítulo se dio un listado de todas las parentizaciones del producto A_0, A_1, A_2, A_3, A_4 las cuales son catorce y obviamente a nosotros nos gustaría que dado el tamaño de las matrices la parentización óptima fuera de las primeras parentizaciones de este listado. Sin embargo no siempre es así. Si implementamos el codigo anterior en un computador e introducimos los siguientes valores:

How many matrices do you have: 5

$p[0] = 5$, $p[1] = 5$, $p[2] = 20$, $p[3] = 10$, $p[4] = 90$, $p[5] = 10$.

La parentización óptima que arroja el programa es: $\langle\langle\langle A[0]\langle A[1]A[2]\rangle\rangle A[3]\rangle A[4]\rangle$ la cual corresponde al número 13 en el listado.

Ahora, veamos otros ejemplos sólo variando el tamaño de las matrices:

How many matrices do you have: 5

$p[0] = 10$, $p[1] = 8$, $p[2] = 5$, $p[3] = 30$, $p[4] = 40$, $p[5] = 5$.

La parentización óptima que ahora arroja el programa es: $\langle A[0]\langle A[1]\langle A[2]\langle A[3]A[4]\rangle\rangle\rangle\rangle$ la cual corresponde a la número 1 en el listado.

How many matrices do you have: 5

$p[0] = 6$, $p[1] = 9$, $p[2] = 5$, $p[3] = 8$, $p[4] = 7$, $p[5] = 5$.

La parentización óptima que arroja el programa es: $\langle\langle A[0]A[1]\rangle\langle\langle A[2]A[3]\rangle A[4]\rangle\rangle$ la cual corresponde a la número 7 en el listado.

Por lo tanto de los ejemplos que acabamos de ver podemos concluir que no se puede conocer de inmediato la parentización óptima ya que esta podría llegar a ser cualquiera.

Capítulo 6

Algoritmo de Strassen

Por razones de simplicidad, comenzaremos nuestro análisis en función de matrices cuadradas de tamaño n.

Primer caso: Si n es potencia de dos (es decir que existe un entero no negativo k tal que n= 2^k) :

Sean A y B dos matrices de tamaño n x n y C su producto.

El algoritmo clásico calcula:

$$C_{ij} = \sum A_{ik} B_{kj}; \quad i, j = 1, \cdots, n$$

Es decir, precisa un tiempo $\Theta(n^3)$.

6.1. Algoritmo divide y vencerás

La técnica de divide y vencerás, también conocido como divide y conquista, se basa en reducir un problema dado en dos o más subproblemas más pequeños y, sucesivamente, volver a aplicar el mismo método sobre los resultados obtenidos hasta alcanzar subproblemas de resolución trivial. De manera general, estos son los pasos a seguir a la hora de crear un algorimo basado en el método divide y vencerás:

1) Dado un problema identificar los problemas del mismo tipo.

2) Resolver recursivamente cada subproblema.

3) Combinar las soluciones obtenidas en una única solución al problema inicial.

Veamos un bosquejo general sobre como podemos aplicar este algoritmo al problema de multiplicar dos matrices. El producto de A por B deseamos guardarlo en la matriz C.

Observemos que si generalizamos al caso de matrices de tamaño n x n (n potencia de dos), siempre podemos dividir cada matriz A, B, C en cuatro submatrices, de tamaño $\frac{n}{2}$ x $\frac{n}{2}$, es decir, se reescribe la ecuación C=AB como sigue:

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & g \\ f & h \end{pmatrix}$$

donde :

$$r = ae + bf \tag{6.1}$$

$$s = ag + bh \tag{6.2}$$

$$t = ce + df \tag{6.3}$$

$$u = cg + dh \tag{6.4}$$

Cada una de estas cuatro ecuaciones especifica dos multiplicaciones de tamaño $\frac{n}{2}$ x $\frac{n}{2}$ y la adición de sus $\frac{n}{2}$ x $\frac{n}{2}$ productos. El algoritmo divide y vencerás efectúa: 8 multiplicaciones y 4 sumas.

Por lo tanto, el tiempo *T(n)* para multiplicar dos n x n matrices, es

$$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$$

que de hecho si aplicamos el Teorema Maestro se puede probar que está en $\Theta(n^3)$ y que, por lo tanto, no se ha conseguido mejorar el coste del algoritmo clásico.

Sin embargo, **¡podemos ahorrarnos una multiplicación!** ya que existen algoritmos más eficientes para resolver este problema:

6.2. Algoritmo de Strassen

El **Algoritmo de Strassen** consume un tiempo

$$T(n) = 7T(\frac{n}{2}) + \Theta(n^2)$$

el cual como vimos en el capítulo 4, al aplicar el Teorema Maestro probamos que está en $\Theta(n^{lg7}) = \Theta(n^{2.8073})$. Por lo tanto su costo es menor que el algoritmo clásico y que el algoritmo divide y vencerás.

El general la idea de Strassen es reducir el número de multiplicaciones a costa de aumentar el número de sumas y restas e introducir variables auxiliares. Para multiplicar dos matrices, se requieren sólo 7 multiplicaciones y 18 sumas y restas. Desde este punto de vista, el método de Strassen es preferible al algoritmo clásico.

El método de Strassen dice que se puede obtener su producto mediante las siguientes operaciones:

$$m_1 = (a_{12} - a_{22}).(b_{21} + b_{22})$$

$$m_2 = (a_{11} + a_{22}).(b_{11} + b_{22})$$

$$m_3 = (a_{11} - a_{21}).(b_{11} + b_{12})$$

$$m_4 = (a_{11} + a_{12}).b_{22}$$

$$m_5 = a_{11}.(b_{12} - b_{22})$$

$$m_6 = a_{22}.(b_{21} - b_{11})$$

$$m_7 = (a_{21} + a_{22}).b_{11}$$

$$\mathbf{AB} = \begin{pmatrix} m_1 + m_2 - m_4 + m_6 & m_4 + m_5 \\ m_6 + m_7 & m_2 - m_3 + m_5 - m_7 \end{pmatrix}$$

Así, Strassen divide a la matriz en cuatro submatrices, cuyos elementos superiores izquierdos son, si consideramos $k = \frac{n}{2}$, los siguientes:

$$\left(\begin{array}{c|c|c} a_{11} & & a_{1,1+k} \\ \hline & & \\ \hline a_{1+k,1} & & a_{1+k,1+k} \end{array}\right)$$

Si n es mayor que dos, podemos pensar que el problema se transforma en la siguiente forma:

Al calcular m_1, por ejemplo, en lugar de estar calculando un escalar, estaríamos calculando una submatriz. La submatriz sería de tamaño $\frac{n}{2}$ x $\frac{n}{2}$. Así la diferencia entre a_{12} y a_{22} se convierte en la diferencia entre submatrices $a_{1+k,1}$ y $a_{1+k,1+k}$ respectivamente.

Al dividir la matriz obtenemos cuatro submatrices, y entonces verificamos si son de dos por dos, si es así aplicamos directamente la fórmula de Strassen para escalares. Si son de dimensión mayor, volvemos a partir y aplicamos la fórmula de Strassen usando matrices en lugar de escalares.

Desde esta base, el algoritmo de Strassen divide el problema inicial en subproblemas. Tras resolver estos subproblemas combina sus soluciones en una única solución que corresponda al problema global de partida. Si, a pesar de esto, aun los subproblemas fueran grandes, el algoritmo de Strassen puede volverse a aplicar.

Hay que destacar que generalmente, los subproblemas resultantes son del mismo tipo que el problema inicial. En este caso, la reaplicación del algoritmo de Strassen se hace mediante el uso de procedimientos recursivos.

Lo que pretenderemos a continuación es presentar el método de Strassen, como un algoritmo y justificar todo lo que se ha mencionado sobre él. Asi mismo aprovecharemos para ilustrar este algoritmo con un ejemplo.

El Algoritmo de Strassen consta de 4 pasos:

- 1) Divide las matrices A y B en submatrices de tamaño $\frac{n}{2}$ x $\frac{n}{2}$.
- 2) Usando $\Theta(n^2)$ sumas y restas, calculan 14 matrices A_1, B_1, A_2, B_2, A_3, B_3,A_4, B_4, A_5, B_5, A_6, B_6, A_7, B_7 de tamaño $\frac{n}{2}$ x $\frac{n}{2}$.
- 3) Recursivamente calcula los 7 productos entre matrices $P_i = A_iB_i$ para $i= 1, \cdots, 7$.
- 4) Calcula las submatrices *r, s, t, u* de la matriz resultante C sumando y substrayendo varias combinaciones de las P_i matrices, usando solo $\Theta(n^2)$ sumas y restas.

No es claro como Strassen exactamente descubre los productos de submatrices que son la clave para hacer que este algoritmo funcione. Sin embargo, nosotros reconstruimos una posible forma.

Descubrimos que cada producto de matrices P_i puede ser escrito de la forma:

$$\begin{aligned} P_i &= A_iB_i \\ &= (\alpha_{i1}a + \alpha_{i2}b + \alpha_{i3}c + \alpha_{i4}d)(\beta_{i1}e + \beta_{i2}f + \beta_{i3}g + \beta_{i4}h) \end{aligned}$$

donde los coeficientes α_{ij}, β_{ij} están en el conjunto $\{-1, 0, 1\}$.

Es decir, descubrimos que cada producto es calculado sumando o substrayendo algunas de las submatrices de A, sumando o restando algunas de las submatrices de B, y después multiplicando ambos resultados.

Por ejemplo, por conveniencia usando matrices de tamaño 4 x 4 podemos reescribir la ecuación (6.4) representandola como combinaciones lineales de productos de submatrices, donde cada producto combina una submatriz de A con una submatriz de B.

Así,

$$r = ae + bf = \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} +1 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}$$

$$\begin{matrix} a \\ b \\ c \\ d \end{matrix} \begin{pmatrix} + & \bullet & \bullet & \bullet \\ \bullet & + & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{pmatrix}$$

La expresión anterior usa una notación abreviada en la cual (+) representa +1, (•) representa 0, y (−) representa −1. Usando esta notación tenemos la siguiente ecuación para las otras submatrices de la matriz C:

$$s = ag + bh = \begin{pmatrix} \bullet & \bullet & + & \bullet \\ \bullet & \bullet & \bullet & + \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{pmatrix}$$

$$t = ce + df = \begin{pmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ + & \bullet & \bullet & \bullet \\ \bullet & + & \bullet & \bullet \end{pmatrix}$$

$$u = cg + dh = \begin{pmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & + & \bullet \\ \bullet & \bullet & \bullet & + \end{pmatrix}$$

Ahora, si observamos que la submatriz s puede ser calculada como s= $P_1 + P_2$, donde P_1 y P_2 son calculados usando una multiplicación para cada una.

$$P_1 = A_1B_1 = a(g - h) = ag - ah = \begin{pmatrix} \bullet & \bullet & + & - \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{pmatrix}$$

$$P_2 = A_2B_2 = (a + b)h = ah + bh = \begin{pmatrix} \bullet & \bullet & \bullet & + \\ \bullet & \bullet & \bullet & + \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{pmatrix}$$

La matriz t puede ser calculada de una manera similar como t= $P_3 + P_4$

$$P_3 = A_3B_3 = (c+d)e = ce + de = \begin{pmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ + & \bullet & \bullet & \bullet \\ + & \bullet & \bullet & \bullet \end{pmatrix}$$

$$P_4 = A_4B_4 = d(f-e) = df - de = \begin{pmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ - & + & \bullet & \bullet \end{pmatrix}$$

Detengamonos ahora y definamos un *término esencial* como uno de los 8 términos que aparecen en el lado derecho de cada una de las ecuaciones (6.1) - (6.4).

Observemos que hasta aquí hemos relizado cuatro productos para calcular las dos submatrices s y t cuyos términos esencials son *ag, bh, ce* y *df*. Notemos que P_1 calcula el término esencial *ag*, P_2 calcula el término esencial *bh*, P_3 calcula el término esencial *ce*, y P_4 calcula el término esencial *df*.

Esto nos recuerda que para calcular las dos submatrices r y u, cuyos términos esenciales son los términos de la diagonal: *ae, bf,cg* y *dh* no debemos realizar más de tres productos.

Ahora trataremos de calcular P_5 para después calcular dos términos esenciales, haciendo algunas manipulaciones.

$$P_5 = A_5B_5 = (a+d)(e+h) = ae + ah + de + dh = \begin{pmatrix} + & \bullet & \bullet & + \\ \bullet & \bullet & \bullet & \bullet \\ + & \bullet & \bullet & + \\ - & + & \bullet & \bullet \end{pmatrix}$$

Adicionalmente para calcular los dos términos esenciales *ae* y *dh*, P_5 calcula los términos esenciales *ah* y *de*, los cuales necesitan ser cancelados, nosotros podemos usar P_4 y P_2 para cancelarlos, pero otros términos esenciales aparecen.

$$P_5 + P_4 - P_2 = ae + dh + df - bh = \begin{pmatrix} + & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & - \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & + & \bullet & + \end{pmatrix}$$

Si agregamos un producto adicional:

$$P_6 = A_6B_6 = (b-d)(f+h) = bf + bh - df - dh = \begin{pmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & + & \bullet & + \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & - & \bullet & - \end{pmatrix}$$

obtenemos,

$$r = P_5 + P_4 - P_2 + P_6 = ae + bf = \begin{pmatrix} + & \bullet & \bullet & \bullet \\ \bullet & + & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{pmatrix}$$

De manera similar obtenemos u de P_5 usando P_1 y P_3 para mover los términos no esenciales inecesarios de P_5 en una forma diferente:

$$P_5 + P_1 - P_3 = ae + ag - ce + dh = \begin{pmatrix} + & \bullet & + & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ - & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & + \end{pmatrix}$$

Ahora calculando un producto adicional,

$$P_7 = A_7B_7 = (a-c)(e+g) = ae + ag - ce - cg = \begin{pmatrix} + & \bullet & + & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ - & \bullet & \bullet & - \\ \bullet & \bullet & \bullet & \bullet \end{pmatrix}$$

obtenemos,

$$u = P_5 + P_1 - P_3 - P_7 = cg + dh = \begin{pmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & + & \bullet \\ \bullet & \bullet & \bullet & + \end{pmatrix}$$

Los siete productos P_1, P_2, P_3, P_4, P_5, P_6, P_7 de submatrices pueden ser usados para calcular el producto C=AB, los cuales completan la descripción del Método de Strassen.

Ejemplo 1: Usemos el Algoritmo de Strassen para calcular el producto:

$$\begin{pmatrix} 1 & 3 \\ 5 & 7 \end{pmatrix} \begin{pmatrix} 8 & 4 \\ 6 & 2 \end{pmatrix}$$

es decir, tenemos que calcular *r, s, t, u* en la siguiente expresión:

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} 1 & 3 \\ 5 & 7 \end{pmatrix} \begin{pmatrix} 8 & 4 \\ 6 & 2 \end{pmatrix}$$

haciendo una analogía con

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & g \\ f & h \end{pmatrix}$$

se tiene que $a = 1$, $b = 3$, $c = 5$, $d = 7$, $e = 8$, $f = 6$, $g = 4$, $h = 2$. Los productos serán encerrados en un círculo, con el propósito de poder ser identificados mejor. Para este caso los productos serán entre escalares.

Utilizando las ecuaciones propuestas por Strassen calculemos primero:

$$s = p_1 + p_2$$

donde $p_1 = a \odot (g - h) = 1 \odot (4 - 2) = 2$ y $p_2 = (a + b) \odot h = 4 \odot 2 = 8$ por lo tanto,

$$s = 2 + 8 = 10$$

Ahora calculemos

$$t = p_3 + p_4$$

Aquí $p_3 = (c + d) \odot e = (12) \odot 8 = 96$, $p_4 = d \odot (f - e) = 7 \odot (-2) = -14$ por lo tanto,

$$t = (96 - 14) = 82$$

Hasta este momento hemos calculado s y t con cuatro productos. Ahora solo falta calcular r y u. Para ello tenemos que calcular primero p_5 y p_6:

$$p_5 = (a + d) \odot (e + h) = (1 + 7) \odot (8 + 2) = 8 \odot 10 = 80$$

y

$$p_6 = (b - d) \odot (f + h) = (3 - 7) \odot (6 + 2) = (-4) \odot (8) = -32$$

puesto que

$$r = p_5 + p_4 - p_2 + p_6 = 80 - 14 - 8 - 32 = 26$$

, observemos que para obtener r se calcularon dos productos los cuales también fueron encerrados en un círculo y finalmente calculemos

$$p_7 = (a - c) \odot (e + g) = (1 - 5) \odot (8 + 4) = (-4) \odot (12) = -48$$

para poder calcular u ya que

$$u = p_5 + p_1 - p_3 - p_7 = 80 + 2 - 96 + 48 = 34$$

Notemos que el total total de productos utilizando el Algoritmo de Strassen es 7 que es menor a los 8 productos que se necesitan para calcular este mismo producto por el Algoritmo clásico.

Substituyendo estos valores en la expresión:

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} 1 & 3 \\ 5 & 7 \end{pmatrix} \begin{pmatrix} 8 & 4 \\ 6 & 2 \end{pmatrix}$$

se tiene que

$$\begin{pmatrix} 26 & 10 \\ 82 & 34 \end{pmatrix} = \begin{pmatrix} 1 & 3 \\ 5 & 7 \end{pmatrix} \begin{pmatrix} 8 & 4 \\ 6 & 2 \end{pmatrix}$$

Ahora veamos otro ejemplo en donde el tamaño de las matrices que se van a multiplicar aumenta y que nos permitirá apreciar la relación de recurrencia en la que se basa el método de Strassen:

Ejemplo 2: Multipliquemos

$$\begin{pmatrix} 7 & 6 & 9 & 4 \\ 8 & 5 & 3 & 2 \\ 7 & 7 & 9 & 9 \\ 4 & 3 & 8 & 5 \end{pmatrix} \begin{pmatrix} 3 & 2 & 9 & 5 \\ 7 & 5 & 6 & 4 \\ 4 & 6 & 9 & 1 \\ 9 & 3 & 8 & 7 \end{pmatrix}$$

Nuevamente haciendo una analogía se tiene que:

$$\left(\begin{array}{cc|cc} 7 & 6 & 9 & 4 \\ 8 & 5 & 3 & 2 \\ \hline 7 & 7 & 9 & 9 \\ 4 & 3 & 8 & 5 \end{array}\right) \left(\begin{array}{cc|cc} 3 & 2 & 9 & 5 \\ 7 & 5 & 6 & 4 \\ \hline 4 & 6 & 9 & 1 \\ 9 & 3 & 8 & 7 \end{array}\right) = \left(\begin{array}{c|c} R & S \\ \hline T & U \end{array}\right)$$

obviamente R, S, T, U serán matrices de tamaño 2 x 2.

Aquí,

$$A = \begin{pmatrix} 7 & 6 \\ 8 & 5 \end{pmatrix}$$

$$B = \begin{pmatrix} 9 & 4 \\ 3 & 2 \end{pmatrix}$$

$$C = \begin{pmatrix} 7 & 7 \\ 4 & 3 \end{pmatrix}$$

$$D = \begin{pmatrix} 9 & 9 \\ 8 & 5 \end{pmatrix}$$

$$E = \begin{pmatrix} 3 & 2 \\ 7 & 5 \end{pmatrix}$$

$$F = \begin{pmatrix} 4 & 6 \\ 9 & 3 \end{pmatrix}$$

$$G = \begin{pmatrix} 9 & 5 \\ 6 & 4 \end{pmatrix}$$

$$H = \begin{pmatrix} 9 & 1 \\ 8 & 7 \end{pmatrix}$$

Por lo tanto apliquemos las ecuaciones propuestas por Strassen y calculemos P_1 y P_2 para obtener S= P_1+P_2. Aquí, los productos entre las matrices serán denotados por $\otimes$, y los productos escalares se encerrarán en un círculo como en el ejemplo anterior.

Recordemos que:

$P_1 = A \otimes (G - H) =$

$$= \begin{pmatrix} 7 & 6 \\ 8 & 5 \end{pmatrix} \otimes \left(\begin{pmatrix} 9 & 5 \\ 6 & 4 \end{pmatrix} - \begin{pmatrix} 9 & 1 \\ 8 & 7 \end{pmatrix} \right) = \begin{pmatrix} 7 & 6 \\ 8 & 5 \end{pmatrix} \otimes \begin{pmatrix} 0 & 4 \\ -2 & -3 \end{pmatrix} = \begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

para calcular r, s, t, u apliquemos nuevamente las ecuaciones propuestas por Strassen. Ahora $a = 7$, $b = 6$, $c = 8$, $d = 5$, $e = 0$, $g = 4$, $f = -2$, $h = -3$.

Así, $p_1 = 7 \odot \big(4 - (-3)\big) = 7 \odot (4 + 3) = 49$, $p_2 = (7 + 6) \odot (-3) = -39$ por lo tanto $s = p_1 + p_2 = 49 - 39 = 10$, $p_3 = (8 + 5) \odot 0 = 0$, $p_4 = 5 \odot (-2 - 0) = -10$, luego $t = p_3 + p_4 = -10$, $p_5 = (7 + 5) \odot \big(0 + (-3)\big) = 12 \odot (-3) = -36$, $p_6 = (6 - 5) \odot (-2 - 3) = 1 \odot (-5) = -5$ y $r = p_5 + p_4 - p_2 + p_6 = -36 - 10 + 39 - 5 = -12$.

Finalmente calculemos $p_7 = (7 - 8) \odot (0 + 4) = (-1) \odot (4) = -4$ para obtener $u = p_5 + p_1 - p_3 - p_7$.

Con estos valores $u = -36 + 49 - 0 + 4 = 17$.

Substituyendo los valores de $r = -12$, $s = 10$, $t = -10$, $u = 17$ se tiene que

$$P_1 = \begin{pmatrix} 7 & 6 \\ 8 & 5 \end{pmatrix} \otimes \begin{pmatrix} 0 & 4 \\ -2 & -3 \end{pmatrix} = \begin{pmatrix} -12 & 10 \\ -10 & 17 \end{pmatrix}$$

Ahora regresemos a calcular $P_2 = (A + B) \otimes H =$

$$= \left(\begin{pmatrix} 7 & 6 \\ 8 & 5 \end{pmatrix} + \begin{pmatrix} 9 & 4 \\ 3 & 2 \end{pmatrix} \right) \otimes \begin{pmatrix} 9 & 1 \\ 8 & 7 \end{pmatrix} = \begin{pmatrix} 16 & 10 \\ 11 & 7 \end{pmatrix} \otimes \begin{pmatrix} 9 & 1 \\ 8 & 7 \end{pmatrix} = \begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

para calcular r, s, t, u apliquemos nuevamente las ecuaciones propuestas por Strassen. Ahora $a = 16$, $b = 10$, $c = 11$, $d = 7$, $e = 9$, $g = 1$, $f = 8$, $h = 7$.

Así, $p_1 = 16 \odot (1 - 7) = 16 \odot (-6) = -96$, $p_2 = (16 + 10) \odot (7) = 182$ por lo tanto $s = p_1 + p_2 = 86$,$p_3 = (11 + 7) \odot 9 = 162$, $p_4 = 7 \odot (8 - 9) = -7$, luego $t = p_3 + p_4 = 162 + (-7) = 155$, $p_5 = (16+7) \odot (9+7) = 23 \odot 16 = 368$, $p_6 = (10-7) \odot (8+7) = 3 \odot 15 = 45$ y $r = p_5 + p_4 - p_2 + p_6 = 368 + (-7) - 182 + 45 = 224$

Finalmente calculemos $p_7 = (16 - 11) \odot (9 + 1) = 50$ para obtener $u = p_5 + p_1 - p_3 - p_7$.

Así $u = 368 - 96 - 162 - 50 = 60$. Substituyendo los valores de $r = 224$, $s = 86$, $t = 155$, $u = 60$ se tiene que

$$P_2 = \begin{pmatrix} 16 & 10 \\ 11 & 7 \end{pmatrix} \otimes \begin{pmatrix} 9 & 1 \\ 8 & 7 \end{pmatrix} = \begin{pmatrix} 224 & 86 \\ 155 & 60 \end{pmatrix}$$

con los valores de P_1 y P_2 ahora ya podemos determinar el valor de

$$S = P_1 + P_2 = \begin{pmatrix} -12 & 10 \\ -10 & 17 \end{pmatrix} + \begin{pmatrix} 224 & 86 \\ 155 & 60 \end{pmatrix} = \begin{pmatrix} 212 & 96 \\ 145 & 77 \end{pmatrix}$$

Ahora obtengamos $P_3 = (C + D) \otimes E =$

$$= \left(\begin{pmatrix} 7 & 7 \\ 4 & 3 \end{pmatrix} + \begin{pmatrix} 9 & 9 \\ 8 & 5 \end{pmatrix} \right) \otimes \begin{pmatrix} 3 & 2 \\ 7 & 5 \end{pmatrix} = \begin{pmatrix} 16 & 16 \\ 12 & 8 \end{pmatrix} \otimes \begin{pmatrix} 3 & 2 \\ 7 & 5 \end{pmatrix} = \begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

para calcular r, s, t, u apliquemos nuevamente las ecuaciones propuestas por Strassen. Ahora $a = 16$, $b = 16$, $c = 12$, $d = 8$, $e = 3$, $g = 2$, $f = 7$, $h = 5$.

Así, $p_1 = 16 \odot (2 - 5) = -48$, $p_2 = (16 + 16) \odot (5) = 160$ por lo tanto $s = p_1 + p_2 = -48 + 160 = 112$, $p_3 = (12 + 8) \odot 3 = 60$, $p_4 = 8 \odot (7 - 3) = 32$, luego $t = p_3 + p_4 = 92$, $p_5 = (16 + 8) \odot ((3 + 5)) = 24 \odot 8 = 192$, $p_6 = (16 - 8) \odot (7 + 5) = 8 \odot 12 = 96$ y $r = p_5 + p_4 - p_2 + p_6 = 192 + 32 - 160 + 96 = 160$.

Finalmente calculemos $p_7 = (16 - 12) \odot (3 + 2) = 20$ para obtener $u = p_5 + p_1 - p_3 - p_7$. Así, $u = 192 - 48 - 60 - 20 = 64$.

Substituyendo los valores de $r = 160$, $s = 112$, $t = -92$, $u = 64$ se tiene que

$$P_3 = \begin{pmatrix} 16 & 16 \\ 12 & 8 \end{pmatrix} \otimes \begin{pmatrix} 3 & 2 \\ 7 & 5 \end{pmatrix} = \begin{pmatrix} 160 & 112 \\ 92 & 64 \end{pmatrix}$$

Ahora calculemos $P_4 = D \otimes (F - E) =$

$$= \begin{pmatrix} 9 & 9 \\ 8 & 5 \end{pmatrix} \otimes \left(\begin{pmatrix} 4 & 6 \\ 9 & 3 \end{pmatrix} - \begin{pmatrix} 3 & 2 \\ 7 & 5 \end{pmatrix} \right) = \begin{pmatrix} 9 & 9 \\ 8 & 5 \end{pmatrix} \otimes \begin{pmatrix} 1 & 4 \\ 2 & -2 \end{pmatrix} = \begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

para calcular r, s, t, u apliquemos nuevamente las ecuaciones propuestas por Strassen. Ahora $a = 9$, $b = 9$, $c = 8$, $d = 5$, $e = 1$, $g = 4$, $f = 2$, $h = -2$.

Así $p_1 = 9 \odot (4+2) = 9 \odot 6 = 54$, $p_2 = (9+9) \odot (-2) = -36$ por lo tanto $s = p_1 + p_2 = 18$, $p_3 = (8 + 5) \odot 1 = 13$, $p_4 = 5 \odot (2 - 1) = 5$, luego $t = p_3 + p_4 = 18$, $p_5 = (9 + 5) \odot (1 - 2) = 14 \odot (-1) = -14$,$p_6 = (9 - 5) \odot (2 - 2) = 0$ y $r = p_5 + p_4 - p_2 + p_6 = -14 + 5 + 36 + 0 = 27$.

Finalmente calculemos $p_7 = (9 - 8) \odot (1 + 4) = 5$ para obtener $u = p_5 + p_1 - p_3 - p_7$. Así $u = -14 + 54 - 13 - 5 = 22$.

Substituyendo los valores de $r = 27$, $s = 18$, $t = 18$, $u = 22$ se tiene que

$$P_4 = \begin{pmatrix} 9 & 9 \\ 8 & 5 \end{pmatrix} \otimes \begin{pmatrix} 1 & 4 \\ 2 & -2 \end{pmatrix} = \begin{pmatrix} 27 & 18 \\ 18 & 22 \end{pmatrix}$$

con los valores de P_3 y P_4 ahora ya podemos determinar el valor de

$$T = P_3 + P_4 = \begin{pmatrix} 160 & 112 \\ 92 & 64 \end{pmatrix} + \begin{pmatrix} 27 & 18 \\ 18 & 22 \end{pmatrix} = \begin{pmatrix} 187 & 130 \\ 110 & 86 \end{pmatrix}$$

Para obtener el valor de R es necesario calcular previamente P_5 y P_6. Así,

$$P_5 = (A + D) \otimes (E + H) =$$

$$= \left(\begin{pmatrix} 7 & 6 \\ 8 & 5 \end{pmatrix} + \begin{pmatrix} 9 & 9 \\ 8 & 5 \end{pmatrix} \right) \otimes \left(\begin{pmatrix} 3 & 2 \\ 7 & 5 \end{pmatrix} + \begin{pmatrix} 9 & 1 \\ 8 & 7 \end{pmatrix} \right) =$$

$$= \begin{pmatrix} 16 & 15 \\ 16 & 10 \end{pmatrix} \otimes \begin{pmatrix} 12 & 3 \\ 15 & 12 \end{pmatrix} = \begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

para calcular r, s, t, u apliquemos nuevamente las ecuaciones propuestas por Strassen. Ahora $a = 16$, $b = 15$, $c = 16$, $d = 10$, $e = 12$, $g = 3$, $f = 15$, $h = 12$.

Así, $p_1 = 16 \odot (3 - 12) = -144$, $p_2 = (16 + 15) \odot (12) = 372$ por lo tanto $s = p_1 + p_2 = -144+372 = 228$, $p_3 = (16+10) \odot 12 = 312$, $p_4 = 10 \odot (15-12) = 30$, luego $t = p_3+p_4 = 342$, $p_5 = (16 + 10) \odot (12 + 12) = 26 \odot 24 = 624$, $p_6 = (15 - 10) \odot (15 + 12) = 5 \odot 27 = 135$ y $r = p_5 + p_4 - p_2 + p_6 = 624 + 30 - 372 + 135 = 417$.

Finalmente calculemos $p_7 = (16 - 16) \odot (12 + 3) = 0$ para obtener

$$u = p_5 + p_1 - p_3 - p_7$$

Así $u = 624-144-312-0 = 168$. Substituyendo los valores de $r = 417$, $s = 228$, $t = 342$, $u = 168$ se tiene que

$$P_5 = \begin{pmatrix} 16 & 15 \\ 16 & 10 \end{pmatrix} \otimes \begin{pmatrix} 12 & 3 \\ 15 & 12 \end{pmatrix} = \begin{pmatrix} 417 & 228 \\ 342 & 168 \end{pmatrix}$$

Ahora calculemos $P_6 = (B - D) \otimes (F + H) =$

$$= \left(\begin{pmatrix} 9 & 4 \\ 3 & 2 \end{pmatrix} - \begin{pmatrix} 9 & 9 \\ 8 & 5 \end{pmatrix} \right) \otimes \left(\begin{pmatrix} 4 & 6 \\ 9 & 3 \end{pmatrix} + \begin{pmatrix} 9 & 1 \\ 8 & 7 \end{pmatrix} \right) =$$

$$= \begin{pmatrix} 0 & -5 \\ -5 & -3 \end{pmatrix} \otimes \begin{pmatrix} 13 & 7 \\ 17 & 10 \end{pmatrix} = \begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

para calcular r, s, t, u apliquemos nuevamente las ecuaciones propuestas por Strassen. Ahora $a = 0$, $b = -5$, $c = -5$, $d = -3$, $e = 13$, $g = 7$, $f = 17$, $h = 10$.

Así, $p_1 = 0 \odot (7 - 10) = 0$, $p_2 = \big(0 + (-5)\big) \odot 10 = -50$ por lo tanto $s = p_1 + p_2 = -50$, $p_3 = (-5 - 3) \odot 13 = -104$, $p_4 = -3 \odot (17 - 13) = -12$, luego $t = p_3 + p_4 = -116$, $p_5 = (0 - 3) \odot (13 + 10) = -69$,$p_6 = (-5 + 3) \odot (17 + 10) = -54$ y $r = p_5 + p_4 - p_2 + p_6 = -69 - 12 + 50 - 54 = -85$.

Finalmente calculemos $p_7 = 5 \odot (13 + 7) = 100$ para obtener

$$u = p_5 + p_1 - p_3 - p_7$$

Así, $u = -69 + 0 + 104 - 100 = -65$. Substituyendo los valores de $r = -85$, $s = -50$, $t = -116$, $u = -65$ se tiene que

$$P_6 = \begin{pmatrix} 0 & -5 \\ -5 & -3 \end{pmatrix} \otimes \begin{pmatrix} 13 & 7 \\ 17 & 10 \end{pmatrix} = \begin{pmatrix} -85 & -50 \\ -116 & -65 \end{pmatrix}$$

con los valores de P_5 y P_6 ahora ya podemos determinar el valor de $R = P_5+P_4-P_2+P_6 =$

$$\begin{pmatrix} 417 & 228 \\ 342 & 168 \end{pmatrix} + \begin{pmatrix} 27 & 18 \\ 18 & 22 \end{pmatrix} - \begin{pmatrix} 224 & 86 \\ 155 & 60 \end{pmatrix} + \begin{pmatrix} -85 & -50 \\ -116 & -65 \end{pmatrix} = \begin{pmatrix} 135 & 110 \\ 89 & 65 \end{pmatrix}$$

Por último calculemos el valor de u pero para ello es necesario calcular previamente P_7 . Así,

$$P_7 = (A - C) \otimes (E + G) =$$

$$= \left(\begin{pmatrix} 7 & 6 \\ 8 & 5 \end{pmatrix} - \begin{pmatrix} 7 & 7 \\ 4 & 3 \end{pmatrix} \right) \otimes \left(\begin{pmatrix} 3 & 2 \\ 7 & 5 \end{pmatrix} + \begin{pmatrix} 9 & 5 \\ 6 & 4 \end{pmatrix} \right) =$$

$$= \begin{pmatrix} 0 & -1 \\ 4 & 2 \end{pmatrix} \odot \begin{pmatrix} 12 & 7 \\ 13 & 9 \end{pmatrix} = \begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

para calcular r, s, t, u apliquemos nuevamente las ecuaciones propuestas por Strassen. Ahora $a = 0, b = -1, c = 4, d = 2, e = 12, g = 7, f = 13, h = 9$.

Así, $p_1 = 0 \odot (7-9) = 0$, $p_2 = (-1) \odot 9 = -9$ por lo tanto $s = p_1 + p_2 = -9$, $p_3 = (4+2) \odot 12 = 72$, $p_4 = 2 \odot (13-12) = 2$, luego $t = p_3 + p_4 = 74$, $p_5 = (0+2) \odot (12+9) = 2 \odot 21 = 42$, $p_6 = (-1-2) \odot (13+9) = -3 \odot 22 = -66$ y $r = p_5 + p_4 - p_2 + p_6 = 42 + 2 + 9 - 66 = -13$

Finalmente calculemos $p_7 = (0-4) \odot (12+7) = -76$ para obtener $u = p_5 + p_1 - p_3 - p_7$. Así $u = 42 + 0 - 72 + 76 = 46$.

Substituyendo los valores de $r = -13$, $s = -9$, $t = 74$, $u = 46$ se tiene que

$$P_7 = \begin{pmatrix} 0 & -1 \\ 4 & 2 \end{pmatrix} \otimes \begin{pmatrix} 12 & 7 \\ 13 & 9 \end{pmatrix} = \begin{pmatrix} -13 & -9 \\ 74 & 46 \end{pmatrix}$$

Con este valor, ya podemos obtener el valor de $U = P_5 + P_1 - P_3 - P_7 =$

$$= \begin{pmatrix} 417 & 228 \\ 342 & 168 \end{pmatrix} + \begin{pmatrix} -12 & 10 \\ -10 & 17 \end{pmatrix} - \begin{pmatrix} 160 & 112 \\ 92 & 64 \end{pmatrix} - \begin{pmatrix} -13 & -9 \\ 74 & 46 \end{pmatrix} = \begin{pmatrix} 258 & 135 \\ 166 & 75 \end{pmatrix}$$

Substituyendo los valores de R, S, T, U

$$R = \begin{pmatrix} 135 & 110 \\ 89 & 65 \end{pmatrix}$$

$$S = \begin{pmatrix} 212 & 96 \\ 145 & 77 \end{pmatrix}$$

$$T = \begin{pmatrix} 187 & 130 \\ 110 & 86 \end{pmatrix}$$

$$U = \begin{pmatrix} 258 & 135 \\ 166 & 75 \end{pmatrix}$$

se tiene que

$$\left(\begin{array}{cc|cc} 7 & 6 & 9 & 4 \\ 8 & 5 & 3 & 2 \\ \hline 7 & 7 & 9 & 9 \\ 4 & 3 & 8 & 5 \end{array}\right)\left(\begin{array}{cc|cc} 3 & 2 & 9 & 5 \\ 7 & 5 & 6 & 4 \\ \hline 4 & 6 & 9 & 1 \\ 9 & 3 & 8 & 7 \end{array}\right) = \left(\begin{array}{c|c} R & S \\ \hline T & U \end{array}\right) =$$

$$= \left(\begin{array}{cc|cc} 135 & 110 & 212 & 96 \\ 89 & 65 & 145 & 77 \\ \hline 187 & 130 & 258 & 135 \\ 110 & 86 & 166 & 75 \end{array}\right)$$

En este ejemplo al utilizar el Algoritmo de Strassen para obtener el resultado, si contamos todos los productos que calculamos suman un total de 63 que es menor que los 64 productos que se necesitan efectuar utilizando el Algoritmo clásico para calcular este mismo producto. Sin embargo, es claro que en este ejemplo como en el ejemplo enterior calcular este producto mediante la serie de ecuaciones que propone Strassen es más tardado que calcularlo mediante el Algoritmo clásico aunque nos ahorremos una multiplicación. Por lo tanto es claro que necesitamos hacer algunos comentarios sobre el Algoritmo de Strassen que nos permitirán apreciar mejor en que casos es viable utilizar este método.

6.3. Comentarios sobre el algoritmo de Strassen

El algoritmo de Strassen tuvo un enorme impacto teórico ya que fue el primer algoritmo de multiplicación de matrices cuya complejiad era $\Theta(n^{lg7}) = \Theta(n^{2.8073})$. Además fue uno de los primeros casos en que las técnicas algorítmicas superaban lo que hasta el momento parecían una barrera infranqueable.

Se han obtenido algoritmos todavía más eficientes. El más eficiente conocido es el de Coppersmith y Winograd (1986) cuyo coste es $\Theta(n^{2.376})$.

El algoritmo de Strassen no es competitivo en la práctica, excepto si el número y tamaño n de matrices es muy grande ($n >> 100$).

Segundo caso: ¿Y si n no es potencia de dos?:

¿Que pasa si quiero multiplicar matrices que no son cuadradas y de diferente tamaño; es posible?.

Si es posible, el método a usar es rellenar con ceros y hacer cuadrada la matriz que no lo es, y el tamaño estará determinado por la siguiente potencia de 2, por ejemplo si es de tamaño 3 x 2:

$$\mathbf{A} = \begin{pmatrix} 1 & 5 \\ 2 & 4 \\ 7 & 9 \end{pmatrix}$$

El tamaño de la matriz, se debe rellenar con ceros y dejar de tamaño 4 x 4 ya que 4 sería la siguiente potencia depués de 3 que es el número mayor que tiene el tamaño de esta matriz.

$$\mathbf{A} = \begin{pmatrix} \mathbf{1} & \mathbf{5} & 0 & 0 \\ \mathbf{2} & \mathbf{4} & 0 & 0 \\ \mathbf{7} & \mathbf{9} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Ahora, si quiero multiplicar esta matriz **A** de tamaño 3 x 2 por una matriz **B** de tamaño 4 x 5, deberá aumentarse el tamaño de ambas matrices para que el producto tenga sentido. Entonces, el tamaño de la matriz 3 x 2 deberá aumentar a 8 x 8 ya que 8 es la siguiente potencia de 5 que es el tamaño mayor de la segunda matriz y ambas quedarían de tamaño 8 x 8.

$$\mathbf{B} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 1 \\ 2 & 3 & 4 & 5 & 5 \\ 7 & 8 & 9 & 1 & 2 \end{pmatrix}$$

Entonces, el tamaño de la matriz 3 x 2 deberá aumentar a 8 x 8 ya que 8 es la siguiente potencia de 5 que es el tamaño mayor de la segunda matriz y ambas quedarían de tamaño 8 x 8.

$$\mathbf{B} = \begin{pmatrix} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & 0 & 0 & 0 \\ \mathbf{6} & \mathbf{7} & \mathbf{8} & \mathbf{9} & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{5} & 0 & 0 & 0 \\ \mathbf{7} & \mathbf{8} & \mathbf{9} & \mathbf{1} & \mathbf{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

De esta manera, ya es posible aplicar el método de Strassen para multiplicar matrices.

$$\mathbf{AB} = \begin{pmatrix} \mathbf{1} & \mathbf{5} & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{2} & \mathbf{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{7} & \mathbf{9} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & 0 & 0 & 0 \\ \mathbf{6} & \mathbf{7} & \mathbf{8} & \mathbf{9} & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{5} & 0 & 0 & 0 \\ \mathbf{7} & \mathbf{8} & \mathbf{9} & \mathbf{1} & \mathbf{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Conclusiones

Como lo vimos a lo largo de este trabajo se han estudiado, diseñado, implementado y experimentado distintas formas de multiplicar matrices, donde principalmente, se busca reducir el tiempo de cómputo empleado. Los métodos de fuerza bruta, programación dinámica y divide y vencerás intentaban mejorar el tiempo de ejecución del algoritmo clásico sin embargo, todos alcanzan el mismo orden de complejiad: $\Theta(n^3)$.

Fue hasta el último capítulo, después de habernos convencido que multiplicar matrices no es tan fácil que presentamos el algoritmo de Strassen, uno de los algoritmos más innovadores en cuanto a la multiplicación de matrices resuelto de manera recursiva. Aunque la cantidad de operaciones aritméticas para este método es más difícil de calcular de manera exacta que para la multiplicación de matrices clásica, se pudo estimar en términos de órdenes de complejidad, aplicando el Teorema Maestro. Por lo tanto el método de Strassen tiene a su favor que reduce la complejidad o cantidad de cálculos entre números a $\Theta(n^{lg7}) = \Theta(n^{2.8073})$ considerando como referencia el método clásico.

Desde el punto de vista de la implementación del método de Strassen se tiene lo siguiente:

- Las operaciones entre los elementos de las matrices son distintas de las definidas por el método convencional y por lo tanto los efectos de redondeo y estabilidad númerica pueden ser diferentes dependiendo de los valores de los elementos de las matrices a multiplicar.
- Es necesario calcular datos intermedios, los P_1, P_2, P_3, P_4, P_5, P_6, P_7 para llegar a los valores definitivos a calcular. La eliminación de esos datos intermedios es muy difícil o imposible y de hecho no se considera, por lo que los requerimientos de memoria del método de Strassen son bastante mayores a los del método tradicional.
- Existe cierto desbalance de carga computacional ya que para calcular P_1 por ejemplo, son necesarias una suma y una multiplicación, pero para el cálculo de de P_6 son necesarias dos sumas y una multiplicación. Esto afecta tanto a la cantidad de datos que se accede como a la cantidad de operaciones entre escalares que se deben llevar a cabo. De todas maneras se debe puntualizar que estas diferencias son a nivel de operaciones de $\Theta(n^2)$ (sumas o restas de matrices) frente a operaciones de multiplicación que tienen complejidad de $\Theta(n^3)$.

Es por estas observaciones que **el algoritmo de Strassen no es competitivo en la práctica, excepto si el número y tamaño n de matrices es muy grande** ($n >> 100$).

Bibliografía

Grimaldi, R. P. (1997). *Matemáticas Discreta y Combinatoria*. Addison Wesley Iberoamericana, S.A., México D.F, tercera edition.

Grossman, S. I. (1987). *Álgebra Lineal*. Grupo Editorial Iberoamérica, México D.F, segunda edition.

Thomas, C. H., Charles, L. E., and Ronald, R. L. (2001). *Introduction to Algorithms*. Mc Graw-Hill Book Company, USA, segunda edition.