# Práctica 4. “Solución a problemas computacionales mediante técnicas de programación dinámica y algoritmos voraces”

Benítez Morales Manuel Emilio Tellez Pérez Juan Manuel

## I. INTRODUCCIÓN

Uno de los problemas computacionales que se nos ha presentado es el de la multiplicación de una secuencia de matrices. Como sabemos, cuando se hace un multiplicación de matrices, una debe de ser de la forma PxR y la otra de la forma RxQ para que así tengamos una matriz resultante de la forma PxQ, así que cuando tenemos una secuencia de matrices, gracias a la propiedad asociativa, pueden existir diferentes formas para poder multiplicar dicha secuencia.

Así que el propósito de esta práctica será el de ocupar las técnicas de algoritmos voraces y programación dinámica para poder dar una asociación óptima para que la multiplicación tenga el menor número de operaciones.

A continuación vamos a describir cómo se desarrolla el algoritmo voraz y el algoritmo de programación dinámica para resolver este problema.

## II. ALGORITMO VORAZ

Para poder resolver este problema con un algoritmo voraz tenemos que agrupar nuestras matrices en dos listas ordenadas, una con respecto a las columnas de mayor a menor y la otra con respecto a las filas de mayor a menor.

Con los siguientes pasos vamos a describir el resto del algoritmo.

Mientras haya más de una matriz el la lista 1 se realizan los siguientes pasos:

a.- Se toma la última matriz de la lista 1 a la que llamaremos Aj

b.- Se elige una matriz Ak de la lista 2 tal que pueda multiplicarse con la matriz Aj. (Si no existe una matriz en la lista 2 que pueda multiplicarse, entonces se elige la matriz (Ak) anterior de la lista 1)

c.- Se colocan paréntesis intermedios entre las dos matrices (Ak Aj)

d.- Se agrega a la lista 1 la nueva matriz Akj (producto de las matrices seleccionadas)

e.- Se borran los elementos Aj y Ak en ambas listas.

Después de implementar el algoritmo pudimos generar una tabla de resultados donde vemos la cantidad de operaciones que se harán en la multiplicación y el tiempo de ejecución que tarde el algoritmo.

```
la-MacBook-Pro-de-Juan:Practica4 Juan$ python3 matriz_voraz.py

Suma: 130000

Suma: 815000

Suma: 326

              Operaciones         Tiempo
       Caso1  130000                  0.1537799835205078
       Caso2  815000                  0.051021575927734375
       Caso3  326                   0.07700920104980469
```

## III. PROGRAMACIÓN DINÁMICA

La solución con este método consiste en dividir la multiplicación general de un conjunto de matrices desde i hasta j, en dos multiplicaciones diferentes, cada una se realizará y al final deben juntarse. Esta división se realizará por un índice intermedio k, el cual tomará las decisiones de la mejor opción.

Se utilizan 3 tablas auxiliares para que esto sea posible:
Tabla M: Guarda el número de operaciones óptimo, es decir, el menor de las posibilidades.
Tabla S: Guarda las decisiones que k tomó, es decir, si hay 3 opciones y la más óptima es la segunda, entonces k=2 y ese valor se inserta en S.
Tabla P: Guarda las dimensiones de cada una de las matrices.

las iteraciones se realizan operando los valores que se encuentran en la tabla P y lo que se va insertando en las tablas M y S llenando primeramente las diagonales de las 2 últimas con 0 y asignando valores hacia arriba también en forma diagonal. Las decisiones de k comienzan a partir de la segunda iteración general l o longitud de cadena, comparando si el valor insertado en la tabla M hasta ese momento es menor, en caso de que no, se inserta el valor nuevo, de manera que al final, la última celda de la primera fila contiene el número óptimo de operaciones.

```
E:\RespMyPart\Semestres_ESCOM\6to_Semestre\Análisis de algoritmos\Practica4>python matrizDin.py 6
Matriz A1
        Filas: 30
        Columnas: 35

Matriz A2
        Filas: 35
        Columnas: 15

Matriz A3
        Filas: 15
        Columnas: 5

Matriz A4
        Filas: 5
        Columnas: 10

Matriz A5
        Filas: 10
        Columnas: 20

Matriz A6
        Filas: 20
        Columnas: 25

Tabla P: [30, 35, 15, 5, 10, 20, 25]

Tabla M:
0       15750   7875    9375    11875   15125
0       0       2625    4375    7125    10500
0       0       0       750     2500    5375
0       0       0       0       1000    3500
0       0       0       0       0       5000
0       0       0       0       0       0

Tabla S:
0       1       1       3       3       3
0       0       2       3       3       3
0       0       0       3       3       3
0       0       0       0       4       5
0       0       0       0       0       5
0       0       0       0       0       0

Multiplicación óptima:
((A1(A2A3))((A4A5)A6))
0.0
15125 operaciones.
```

## IV. RESULTADOS

A(50x30), B(30x20) y C(20x100)

| Algoritmos | Operaciones | Asociación | Tiempo |
|---|---|---|---|
| Técnica voraz | 130000 | | 0.15377 |
| Programación dinámica | 130000 | ((AB)C) | 0.00000 |

A(10x200), B(200x300), C(300x50), D(50x90) y E(90x10)

| Algoritmos | Operaciones | Asociación | Tiempo |
|---|---|---|---|
| Técnica voraz | 815000 | | 0.05102 |
| Programación dinámica | 800000 | (((A1A2)A3)(A4A5)) | 0.00000 |

A(1x2), B(2x3), C(3x4), D(4x5), E(5x6) y F(6x7)

| Algoritmos | Operaciones | Asociación | Tiempo |
|---|---|---|---|
| Técnica voraz | 376 | | 0.07700 |
| Programación dinámica | 110 | (((((A1A2)A3)A4)A5)A6) | 1.00064 |

## V. ARTÍCULO

En este artículo se presenta un algoritmo paralelo muy rápido para la aproximación de la solución de este problema y también para encontrar una triangulación óptima a el polígono convexo.

Un algoritmo óptimo para ambos problemas fue dado por hu y Shing y corre en un tipo de O(n log n) y está basado en programación dinámica. Este algoritmo produce el orden de la multiplicación de matrices y la triangulación de un polígono convexo con un error de a lo mucho 15.47%.

## VI. ALGORITMO POR TRIANGULACIÓN DE POLÍGONOS

Entrada: Dos String A y B de longitud M y N respectivamente
Salida: Longitud de la subcadena más larga.

```
para i=1 hasta n
        L[i, 0]= 0
para j=0 hasta m
        L[0, j]= 0

para i=1 hasta n
        para j=1 hasta m
                si xi-yi entonces
                        L[i, j]=L[i-1, j-1]+1
                si no
                        L[i,  j]=  max{L[i-1,
j], L[i, j-1]}

retornar L
```

## VII. COMPARACIÓN DE ALGORITMOS

La ventaja del método dinámico es la facilidad de implementación y que las operaciones son menores a las obtenidas por el método voraz, sin embargo, puede llegar a tardar más, debido a la forma en que obtiene los resultados, ya que debe crear más tablas, mientras que el algoritmo voraz realiza todo con operaciones nativas sin necesidad de nuevas estructuras de código y por lo tento, de procesamiento.

Por otro lado, la programación dinámica arroja siempre el mejor resultado, y lo obtenido por técnica voraz no garantiza eso.

## VIII. CONCLUSIÓN

La multiplicación encadenada de matrices es una herramienta matemática muy útil, sin embargo representa una cantidad enorme de operaciones a realizar y no siempre es sencillo identificar cuál es la mejor manera de realizarlas.

Para tardar el menor tiempo posible con un buen resultado debe analizarse cada forma de operar dichas matrices, para ello se utilizan algoritmos como los reportados aquí, esto permite saber de manera rápida de qué forma debemos multiplicarlas.

Con esta práctica reafirmamos el conocimiento y comprensión de la técnica voraz y programación dinámica haciendo posible la optimización de la multiplicación encadenada de matrices.