

Taller 5

Materia: Análisis de algoritmo

Nombres: Fabian Diaz, Jhordan Huera, Pablo Jiménez, Jimmy Cuatucuamba, Giuliana Espinoza

Fecha: 03/ 06 / 2025

Tema: taller 5 grupal

Realzar en grupos de trabajo un programa secuencial y recursivo en Python.

Metodo Secuencial.-

<pre>def multiplicar_matrices(matriz): filas = len(matriz) columnas = len(matriz[0]) if filas != columnas: print("\nError: Para multiplicar A*A, la matriz debe ser cuadrada") return None resultado = [] for i in range(filas): fila_resultado = [] for j in range(columnas): elemento = 0 for k in range(columnas): elemento += matriz[i][k] * matriz[k][j] fila_resultado.append(elemento) resultado.append(fila_resultado) return resultado</pre>	<p>matriz → 1 vez filas = len(matriz) → 1 vez columnas = len(matriz[0]) → 1 vez if filas != columnas: → 1 vez return None → 1 vez resultado = [] → 1 vez return resultado → 1 vez Total= 7 vez</p> <p>-----Tiempo de k----- #1 vez # n+1 # n # n T(k)=3n+2</p> <p>-----Tiempo de J ----- #1 vez # n+1 # n # n #(3n + 2)n = 3n^2+2n #n T(j)= 3n^2 + 6n + 2</p> <p>-----Tiempo i ----- #1 vez # n+1 # n #n #n #(3n^2 + 6n + 2)n = 3n^3+6n^2+2n</p>
---	--

	$T(i) = 3n^3 + 6n^2 + 6n + 2$ <p>-----Tiempo total -----</p> $\#T \text{ total} = 3n^3 + 6n^2 + 6n + 2 + 7$ $\#T \text{ total} = 3n^3 + 6n^2 + 6n + 9$ $\#T \text{ total} = O(n^3)$
--	---

Metodo recursivo

```
def multiplicacion_recursiva(self, otra):
    if len(self.matriz[0]) != len(otra.matriz):
        raise ValueError("No es válida la multiplicación porque las
dimensiones no coinciden.")
    resultado = Matriz(filas=len(self.matriz),
columnas=len(otra.matriz[0]))

    def calcular_elemento(i, j, k):
        if k < 0:
            return 0
        return self.matriz[i][k] * otra.matriz[k][j] + calcular_elemento(i,
j, k - 1)
        # b + T(n-1) --> n+1 sería el cambio, pero le revertimos porque
va de 0 a n, y evaluamos como si fuese de n a 0
        # T(-1:tam) = a
        # T(0) = b + a
        # T(1) = 2b + a
        # T(2) = 3b + a
        # T(3) = 4b + a
        # T(n) = bn + a    b(n+1)+a

    def llenar(i, j):
        if i < 0:
            return
        if j < 0:
            llenar(i - 1, len(otra.matriz[0]) - 1)
            return
        resultado.matriz[i][j] = calcular_elemento(i, j,
len(self.matriz[0]) - 1) #bn+a
        llenar(i, j - 1)
        # b + T(n-1) --> n+1 sería el cambio, pero le revertimos porque
va de 0 a n, y evaluamos como si fuese de n a 0
        # T(-1:tam) = a
        # T(0) = b + a
        # T(1) = 2b + a
        # T(2) = 3b + a
        # T(n) = bn + a    b(n+1)+a
        # b = bn+b+a
```

$T(n) = n(bn+b+a) + a = bn^2+an+bn+a = bn+n(a+b)+a$

```
llenar(len(self.matriz) - 1, len(otra.matriz[0]) - 1)
return resultado
```

Datos	Secuencial	Recursivo
100		10101
200		40201
300		90301
400		160401
500		250501
600		360601
700		490701
800		640801
900		810901
1000		1001001



Datos	Secuencial	Recursivo
100	3060609	10101
200	24241209	40201
300	81541809	90301
400	192962409	160401
500	376503009	250501
600	650163609	360601
700	1.032E+09	490701
800	1.54E+09	640801
900	2.192E+09	810901
1000	3.006E+09	1001001



```
"C:\Users\jhord\Documents\Universidad\Sexto Semestre\repo-6to\ANAALG\Clase 10\.venv\Scripts\python.exe" "C:\Users\jhord\Documents\Universidad\Sexto Semestre\repo-6to\ANAALG\Clase 10\main.py"
Ingrese el número de filas (n): 4
Ingrese el número de columnas (m): 4

Matriz A:
[[7 7 3 3]
 [8 1 2 5]
 [3 3 1 7]
 [6 1 9 2]]

Resultado multiplicación secuencial A * A':
[[116 84 66 82]
 [ 84 94 64 77]
 [ 66 64 68 44]
 [ 82 77 44 122]]

Resultado multiplicación recursiva A * A':
[[116 84 66 82]
 [ 84 94 64 77]
 [ 66 64 68 44]
 [ 82 77 44 122]]

Resultado usando NumPy (A @ A.T):
[[116 84 66 82]
 [ 84 94 64 77]
 [ 66 64 68 44]
 [ 82 77 44 122]]

Process finished with exit code 0
```

Funcionamiento: