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.-**

|  |  |
| --- | --- |
| **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** | matriz à 1 vez  filas = len(matriz)à 1 vez  columnas = len(matriz[0])à 1 vez  if filas != columnas: à 1 vez  return None à 1 vez  resultado = []à 1 vez  return resultado à  1 vez  **Total= 7 vez**    **-------Tiempo de k-----------**  #1 vez  # n+1  # n  # n  **T(k)=3n+2**    **------Tiempo de J ---------**  #1 vez  # n+1  # n  # n  **#(**3n + 2)n = 3n^2+2n  **#**n  **T(j)= *3n^2 + 6n + 2***    ***------Tiempo i ----------***  #1 vez  # n+1  # n  #n  #n  #(3n^2 + 6n + 2)n = ***3n^3+6n^2+2n***  **T(i)= *3n^3 + 6n^2 + 6n + 2***    ***--------Tiempo total ----------***  ***#T total = 3n^3 + 6n^2 + 6n + 2 + 7***  ***# T total = 3n^3 + 6n^2 + 6n + 9***  ***#T 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*

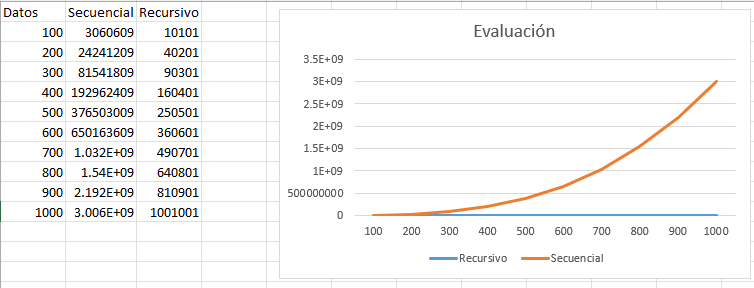
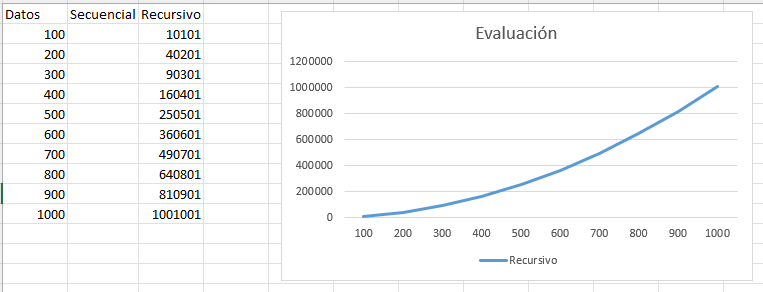
*# T(n) = bn + a  b(n+1)+a*

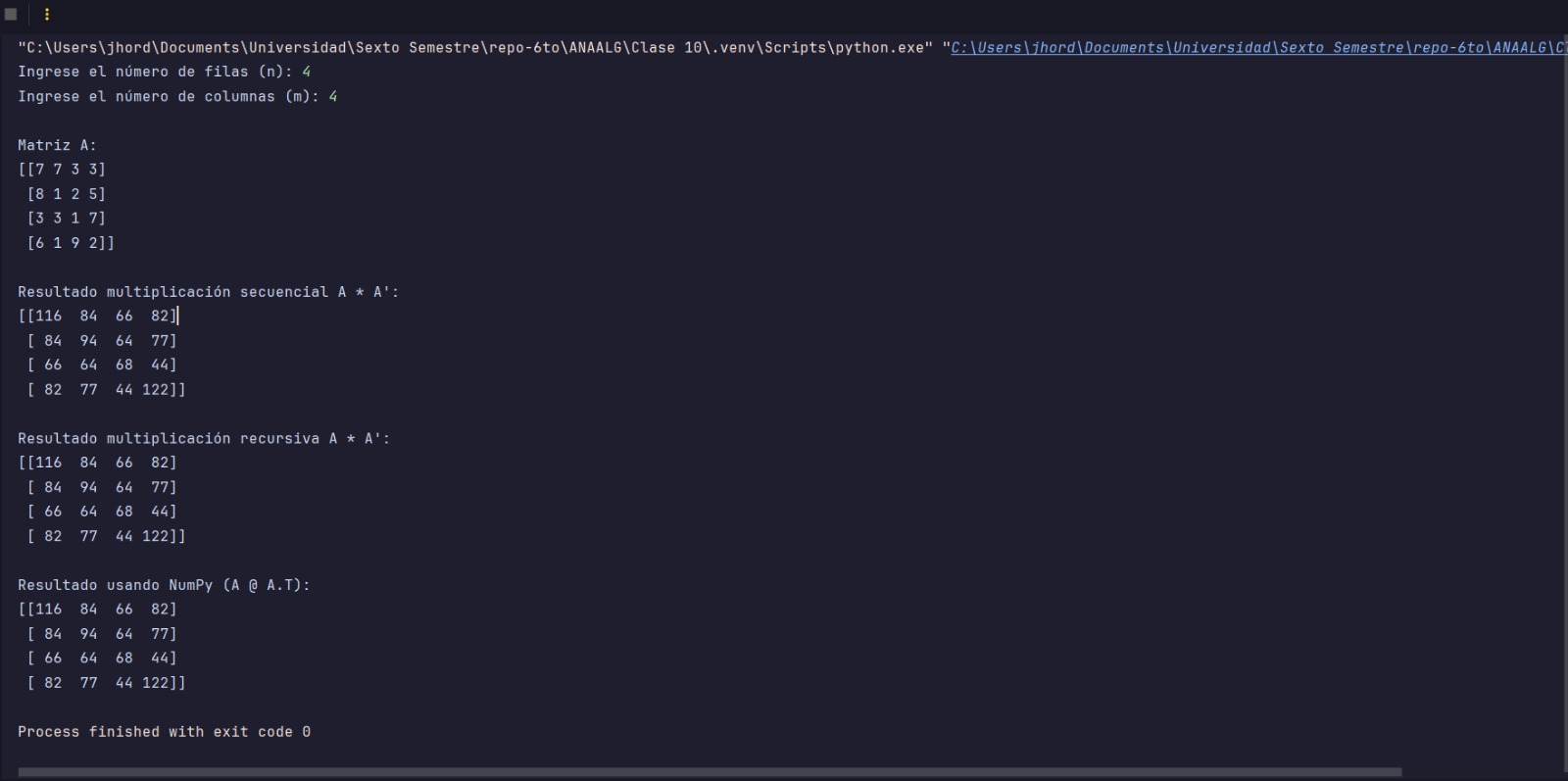
*# b = bn+b+a*

*# T(n) = n(bn+b+a) + a = bn²+an+bn+a = bn+n(a+b)+a*

        llenar(len(*self*.matriz) - 1, len(otra.matriz[0]) - 1)

        return resultado



**Funcionamiento:** 