****

ANÁLISIS PRÁCTICO DE UN ALGORITMO

INTEGRANTES:

LIZARAZO GUTIERREZ ENDERSON JOEL - 1151874

MORALES CASTAÑEDA YOFER NICOLAS - 1152154

ANDREA CAROLINA VILLAMIL RAMÍREZ - 1151700

DEIBY YESID GARCIA RONDEROS - 1152135

VERA CONTRERAS MILTON JESUS

UNIVERSIDAD FRANCISCO DE PAULA SANTANDER NORTE DE SANTANDER, CÚCUTA

ANÁLISIS DE ALGORITMOS

2023

**Introducción**

El objetivo de este trabajo es evidenciar los resultados que se obtuvieron al implementar la solución del problema “Matriz Loca” del Training Center en cinco lenguajes de programación diferentes y dos sistemas operativos (Linux y Windows), y medir el tiempo de ejecución de cada una de las implementaciones. Además, se generaron cinco escenarios de datos aleatorios de distintos tamaños para evaluar el desempeño de las soluciones. Se registraron los datos obtenidos en una hoja de cálculo y se analizó la complejidad del algoritmo utilizado.

**DOCUMENTACIÓN CASOS DE PRUEBA ( WINDOWS)**

**Sistema Operativo: Windows**

**Lenguaje: Java**

public class MatrizLoca {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = 1;

while (n > 0) {

// Se lee la cantidad de filas y columnas de la matriz que se va a crear

System.out.println("numero de filas");

int a = sc.nextInt();

System.out.println("numero de columnas");

int b = sc.nextInt();

long startTime = System.nanoTime();

// Se crea una matriz de dos dimensiones AB con las dimensiones especificadas

int[][] AB = new int[a][b];

// Se crea otra matriz de dos dimensiones M con las mismas dimensiones que AB

int[][] M = new int[a][b];

// Se recorre la matriz AB y se copian los valores ingresados por el usuario a la matriz M

for (int i = 0; i < a; i++) {

for (int j = 0; j < b; j++) {

AB[i][j] = (int) (Math.random() \* 7);

M[i][j] = AB[i][j];

}

}

for (int i = 0; i < AB.length; i++) {

for (int j = 0; j < AB[i].length; j++) {

System.out.print(AB[i][j] + " ");

}

System.out.println();

}

for (int i = 0; i < a; i++) {

for (int j = 0; j < b; j++) {

if (i == 0 || i % 2 == 0) {

M[i][j] += 1;

}

if (j == 0 || j % 2 == 0) {

M[i][j] += 2;

}

if (i % 2 != 0 && j % 2 != 0) {

M[i][j] -= 3;

}

}

}

System.out.println("------------------------");

// Se llama a la función imprimirMatriz para mostrar la matriz resultante

for (int i = 0; i < M.length; i++) {

for (int j = 0; j < M[i].length; j++) {

System.out.print(M[i][j] + " ");

}

System.out.println();

}

n--;

long endTime = System.nanoTime();

double duration = (endTime - startTime) / 1000000.0;

double duration2 = duration / 1000.0;

System.out.printf("El tiempo de ejecución fue de %.5f segundos", duration2);

}

}

public static void imprimirMatriz(int M[][]) {

// Se recorre la matriz y se imprime por pantalla

for (int i = 0; i < M.length; i++) {

for (int j = 0; j < M[i].length; j++) {

if (j == M[i].length - 1) {

System.out.print(M[i][j]);

} else {

System.out.print(M[i][j] + " ");

}

}

}

}

}

Complejidad del Algoritmo:

\* Complejidad espacial es O(a\*b) *se utilizan dos matrices de tamaño ab*

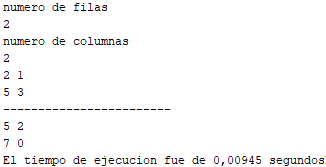
\* La complejidad total del algoritmo es O(n \* a \* b)

\* La ecuación de complejidad de este algoritmo sería: T(n) = O(n^2)

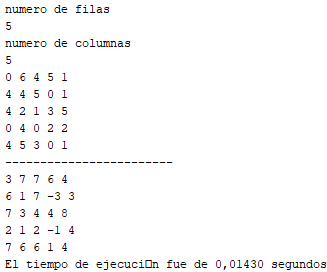
\* La función de complejidad del algoritmo es: f(n) = 4n^2 + 6n + 4

**Casos de prueba Estudiante 1**

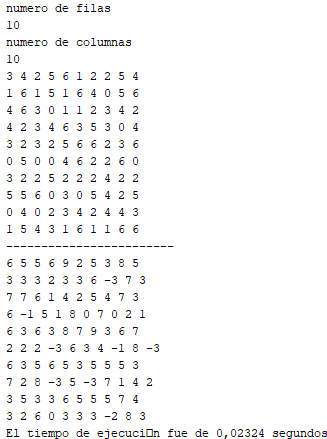
* Matriz 2x2



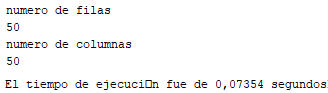
* Matriz 5x5



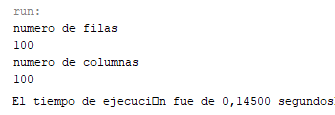
* Matriz 10x10



* Matriz 50x50

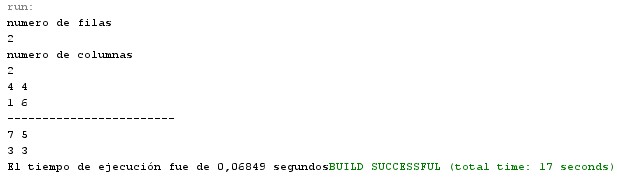


* Matriz 100x100

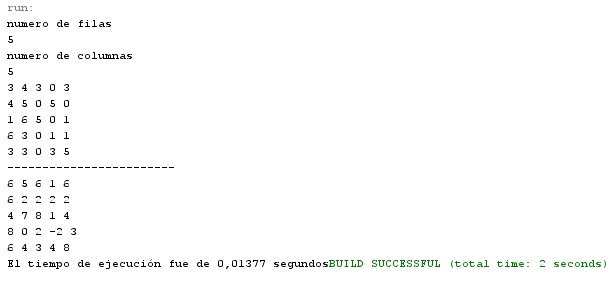


**Casos de prueba Estudiante 2**

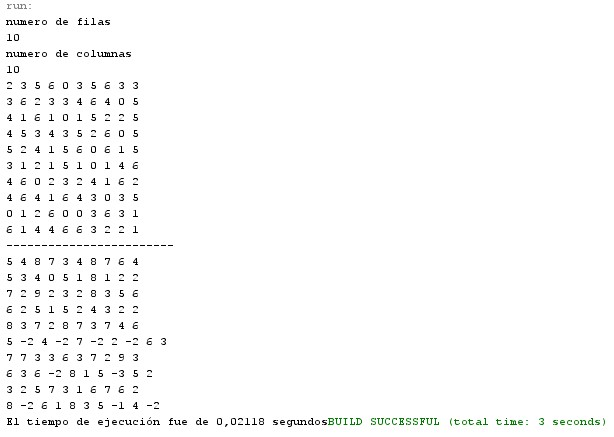
* Matriz 2x2



* Matriz 5x5



* Matriz 10x10



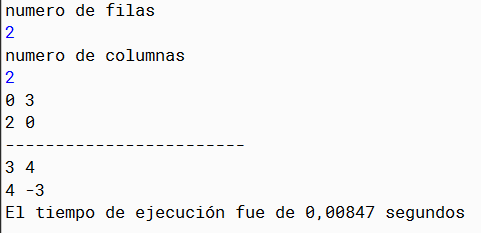
* Matriz 50x50



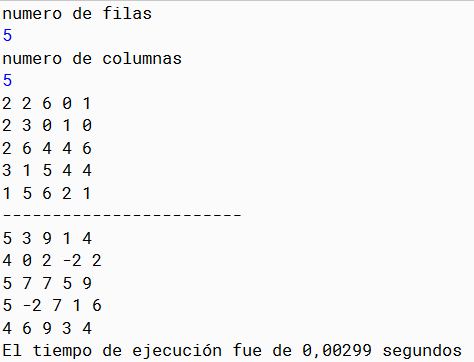
* Matriz 100x100

**Casos de prueba Estudiante 3**

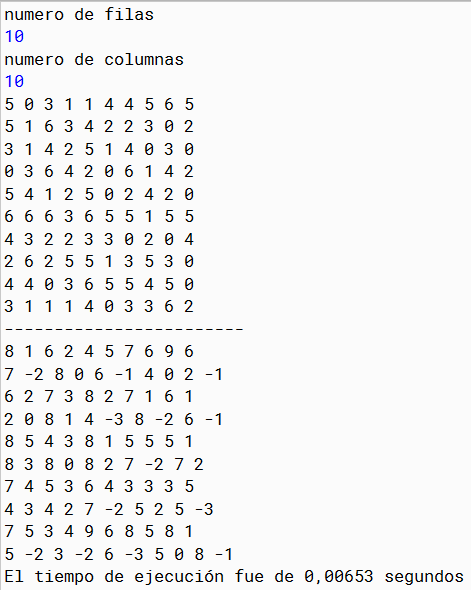
* Matriz 2x2



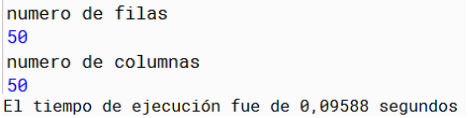
* Matriz 5x5



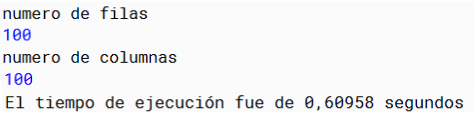
* Matriz 10x10



* Matriz 50x50

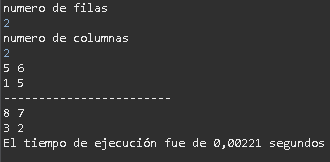


* Matriz 100x100

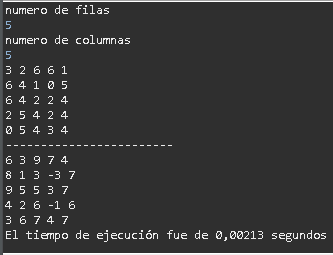


**Casos de prueba Estudiante 4**

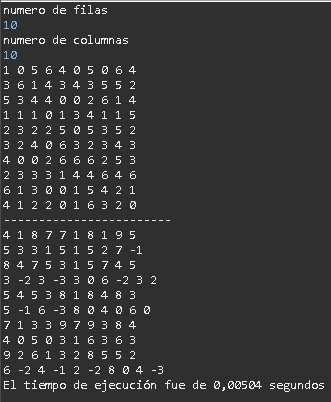
* Matriz 2x2



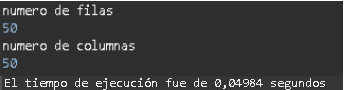
* Matriz 5x5



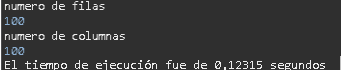
* Matriz 10x10



* Matriz 50x50



* Matriz 100x100



**Lenguaje: Python**

import random

# Se lee la cantidad de filas y columnas de la matriz que se va a crear

a = int(input("Número de filas: "))

b = int(input("Número de columnas: "))

import time

start\_time = time.time()

AB = [[random.randint(0, 7) for j in range(b)] for i in range(a)]

M = [[AB[i][j] for j in range(b)] for i in range(a)]

for i in range(a):

for j in range(b):

if i == 0 or i % 2 == 0:

M[i][j] += 1

if j == 0 or j % 2 == 0:

M[i][j] += 2

if i % 2 != 0 and j % 2 != 0:

M[i][j] -= 3

# Se imprime la matriz AB

print("Matriz AB:")

for i in range(a):

for j in range(b):

print(AB[i][j], end=' ')

print()

# Se imprime la matriz M

print("------------------------")

print("Matriz M:")

for i in range(a):

for j in range(b):

print(M[i][j], end=' ')

print()

# Se calcula el tiempo de ejecución

duration = time.time() - start\_time

print(f"El tiempo de ejecucion fue de {duration:.5f} segundos\n")

Complejidad del Algoritmo:

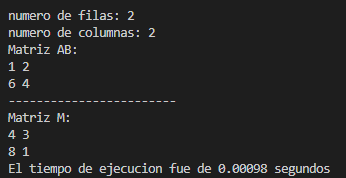
\* Complejidad total es O(a\*b)

\* La ecuación de complejidad de este algoritmo sería: T(n) = O(n^2)

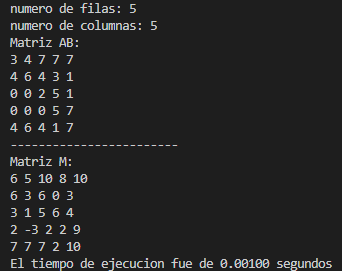
\* La función de complejidad del algoritmo es: f(n) = 6n^2 + 8n + 4

**Casos de prueba Estudiante 1**

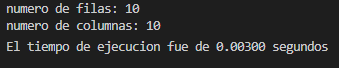
* Matriz 2x2



* Matriz 5x5



* Matriz 10x10



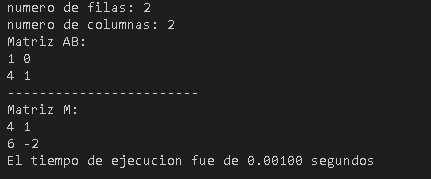
* Matriz 50x50

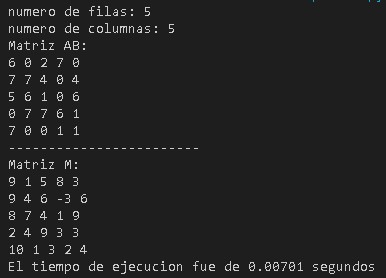


* Matriz 100x100

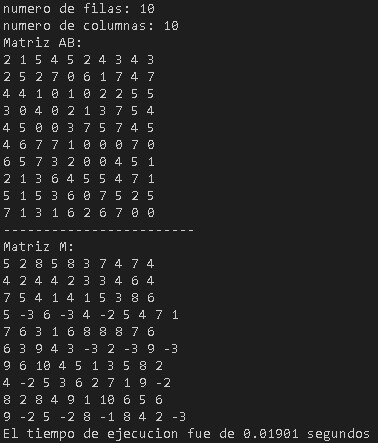


**Casos de prueba Estudiante 2**

* Matriz 2x2
* Matriz 5x5



* Matriz 10x10



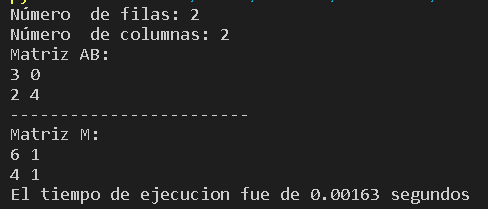
* Matriz 50x50



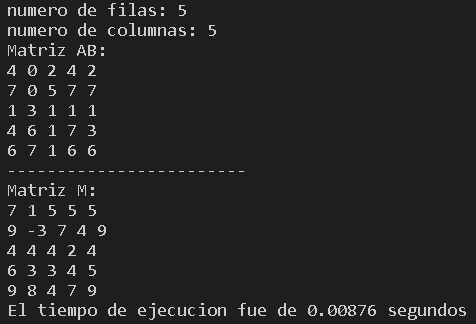
* Matriz 100x100

**Casos de prueba Estudiante 3**

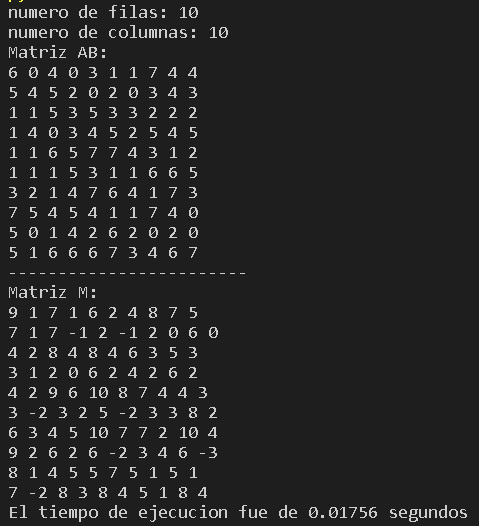
* Matriz 2x2



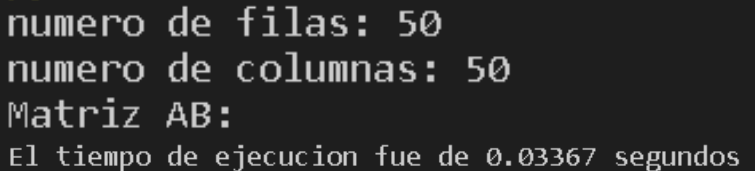
* Matriz 5x5



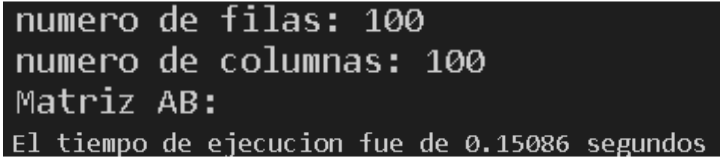
* Matriz 10x10



* Matriz 50x50

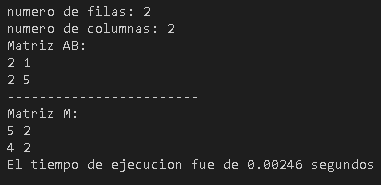


* Matriz 100x100

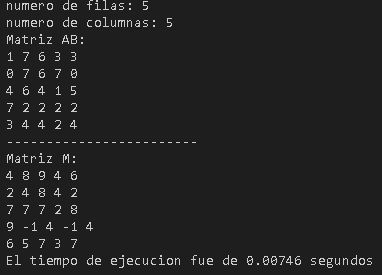
****

**Casos de prueba Estudiante 4**

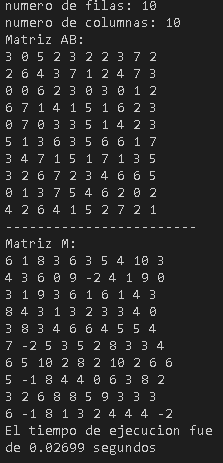
* Matriz 2x2



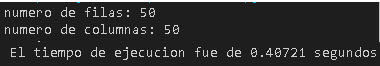
* Matriz 5x5



* Matriz 10x10



* Matriz 50x50



* Matriz 100x100



**Lenguaje: C++**

#include <iostream>

#include <vector>

#include <cstdlib>

#include <ctime>

using namespace std;

void imprimirMatriz(vector<vector<int>>& M) {

int a = M.size();

int b = M[0].size();

for (int i = 0; i < a; i++) {

for (int j = 0; j < b; j++) {

cout << M[i][j] << " ";

}

cout << endl;

}

}

int main() {

int a, b;

cout << "numero de filas: ";

cin >> a;

cout << "numero de columnas: ";

cin >> b;

unsigned t0, t1;

t0=clock();

vector<vector<int>> AB(a, vector<int>(b));

vector<vector<int>> M(a, vector<int>(b));

for (int i = 0; i < a; i++) {

for (int j = 0; j < b; j++) {

AB[i][j] = rand() % 8; // Genera un número aleatorio entre 0 y 7

M[i][j] = AB[i][j];

}

}

for (int i = 0; i < a; i++) {

for (int j = 0; j < b; j++) {

if (i == 0 || i % 2 == 0) {

M[i][j] += 1;

}

if (j == 0 || j % 2 == 0) {

M[i][j] += 2;

}

if (i % 2 != 0 && j % 2 != 0) {

M[i][j] -= 3;

}

}

}

// Se imprime la matriz AB

cout << "Matriz AB:" << endl;

imprimirMatriz(AB);

// Se imprime la matriz M

cout << "------------------------" << endl;

cout << "Matriz M:" << endl;

imprimirMatriz(M);

t1 = clock();

double time = (double(t1-t0)/CLOCKS\_PER\_SEC);

cout << "El Tiempo de ejecicion fue: " << time << endl;

return 0;

}

Complejidad del Algoritmo:

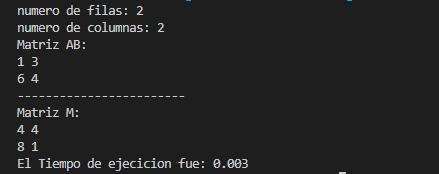
\* Complejidad espacial es O(a\*b) *se utilizan dos matrices de tamaño ab*

\* La ecuación de complejidad de este algoritmo sería: T(n) = O(n^2)

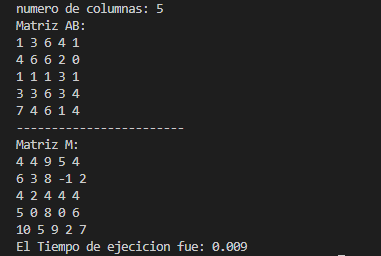
\* La función de complejidad del algoritmo es: f(n) = 4n^2 + 6n + 4

**Casos de prueba Estudiante 1**

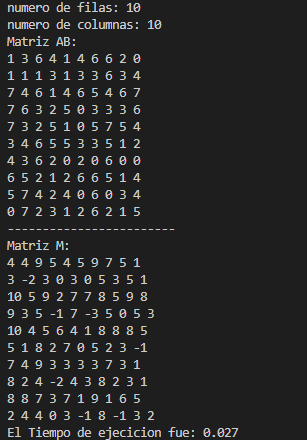
* Matriz 2x2



* Matriz 5x5



* Matriz 10x10



* Matriz 50x50



* Matriz 100x100



**Casos de prueba Estudiante 2**

* Matriz 2x2
* Matriz 5x5
* Matriz 10x10
* Matriz 50x50
* Matriz 100x100

**Casos de prueba Estudiante 3**

* Matriz 2x2
* Matriz 5x5
* Matriz 10x10
* Matriz 50x50
* Matriz 100x100

**Casos de prueba Estudiante 4**

* Matriz 2x2
* Matriz 5x5
* Matriz 10x10
* Matriz 50x50
* Matriz 100x100

**Lenguaje: Javascript**

**Node.js**

const readline = require('readline');

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

rl.question('Ingrese el número de filas: ', (a) => {

rl.question('Ingrese el número de columnas: ', (b) => {

let AB = Array.from({length: parseInt(a)}, () => Array.from({length: parseInt(b)}, () => Math.floor(Math.random() \* 8)));

let M = Array.from({length: parseInt(a)}, () => Array.from({length: parseInt(b)}, () => 0));

let start\_time = new Date().getTime();

for (let i = 0; i < parseInt(a); i++) {

for (let j = 0; j < parseInt(b); j++) {

if (i === 0 || i % 2 === 0) {

M[i][j] += 1;

}

if (j === 0 || j % 2 === 0) {

M[i][j] += 2;

}

if (i % 2 !== 0 && j % 2 !== 0) {

M[i][j] -= 3;

}

}

}

// Se imprime la matriz AB

console.log("Matriz AB:");

for (let i = 0; i < parseInt(a); i++) {

for (let j = 0; j < parseInt(b); j++) {

process.stdout.write(AB[i][j] + ' ');

}

console.log();

}

// Se imprime la matriz M

console.log("------------------------");

console.log("Matriz M:");

for (let i = 0; i < parseInt(a); i++) {

for (let j = 0; j < parseInt(b); j++) {

process.stdout.write(M[i][j] + ' ');

}

console.log();

}

// Se calcula el tiempo de ejecución

let end\_time = new Date().getTime();

let duration = (end\_time - start\_time) / 1000;

console.log(`El tiempo de ejecucion fue de ${duration.toFixed(5)} segundos`);

rl.close();

});

});

Complejidad del Algoritmo:

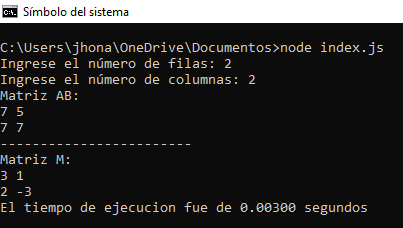
\* Complejidad total es O(a\*b)

\* La ecuación de complejidad de este algoritmo sería: T(n) = O(ab)

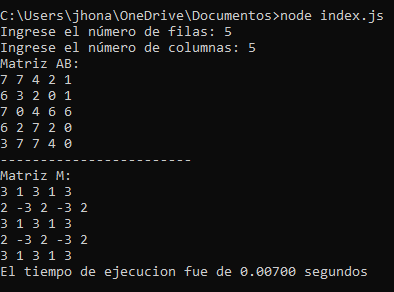
\* La función de complejidad del algoritmo es: f(a,b) = 9ab + 11a + 6

**Casos de prueba Estudiante 1**

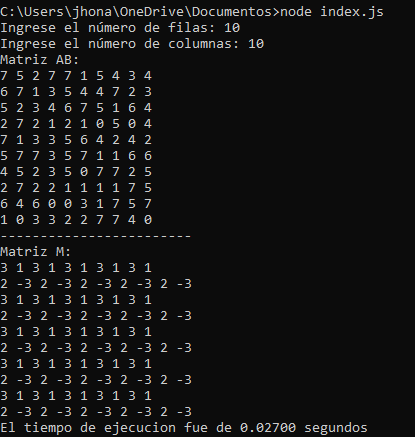
* Matriz 2x2



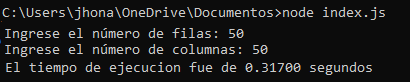
* Matriz 5x5



* Matriz 10x10



* Matriz 50x50

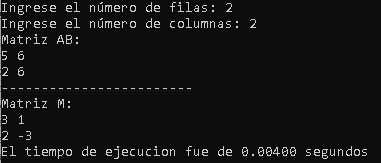


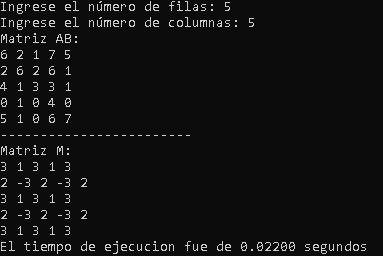
* Matriz 100x100

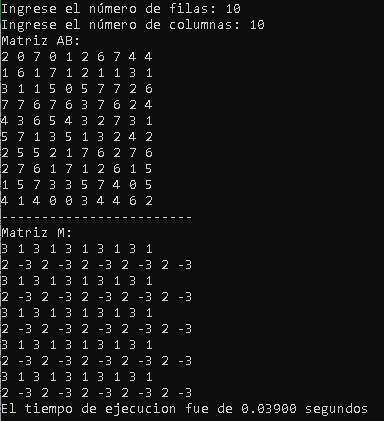


**Casos de prueba Estudiante 2**

* Matriz 2x2



* Matriz 5x5
* Matriz 10x10



* Matriz 50x50





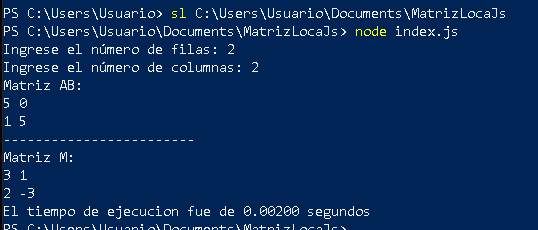
* Matriz 100x100



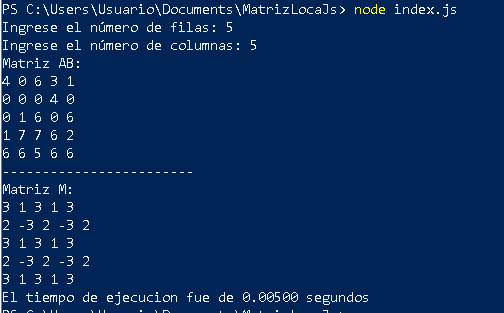


**Casos de prueba Estudiante 3**

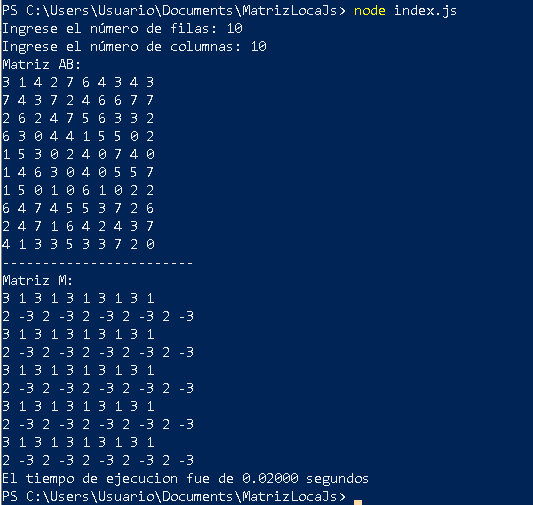
* Matriz 2x2



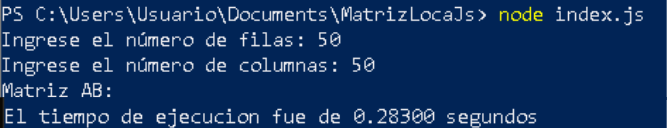
* Matriz 5x5



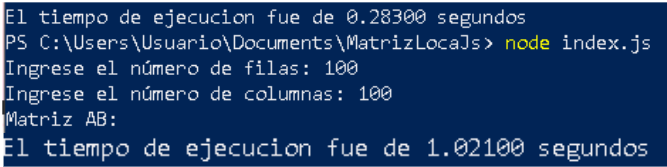
* Matriz 10x10



* Matriz 50x50

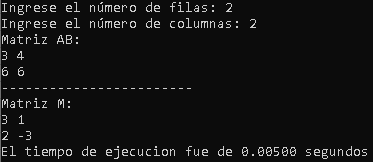


* Matriz 100x100

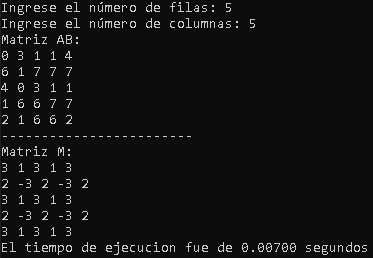


**Casos de prueba Estudiante 4**

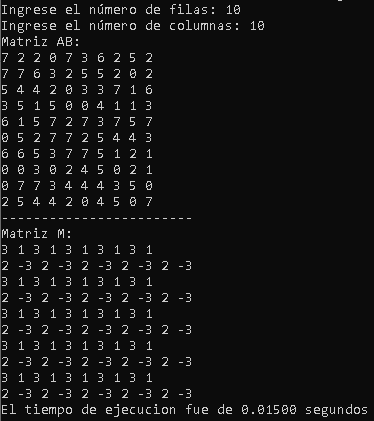
* Matriz 2x2



* Matriz 5x5



* Matriz 10x10



* Matriz 50x50



* Matriz 100x100



**Lenguaje: PHP**

<?php

// Se lee la cantidad de filas y columnas de la matriz que se va a crear

$a = (int)readline("numero de filas: ");

$b = (int)readline("numero de columnas: ");

$start\_time = microtime(true);

$AB = array();

for ($i = 0; $i < $a; $i++) {

$row = array();

for ($j = 0; $j < $b; $j++) {

$row[] = rand(0, 7);

}

$AB[] = $row;

}

$M = array();

for ($i = 0; $i < $a; $i++) {

$row = array();

for ($j = 0; $j < $b; $j++) {

$row[] = $AB[$i][$j];

}

$M[] = $row;

}

for ($i = 0; $i < $a; $i++) {

for ($j = 0; $j < $b; $j++) {

if ($i == 0 || $i % 2 == 0) {

$M[$i][$j] += 1;

}

if ($j == 0 || $j % 2 == 0) {

$M[$i][$j] += 2;

}

if ($i % 2 != 0 && $j % 2 != 0) {

$M[$i][$j] -= 3;

}

}

}

// Se imprime la matriz AB

echo "Matriz AB:\n";

for ($i = 0; $i < $a; $i++) {

for ($j = 0; $j < $b; $j++) {

echo $AB[$i][$j] . ' ';

}

echo "\n";

}

// Se imprime la matriz M

echo "------------------------\n";

echo "Matriz M:\n";

for ($i = 0; $i < $a; $i++) {

for ($j = 0; $j < $b; $j++) {

echo $M[$i][$j] . ' ';

}

echo "\n";

}

// Se calcula el tiempo de ejecución

$end\_time = microtime(true);

$execution\_time = ($end\_time - $start\_time) \* 1000; // Convertir a milisegundos

echo "El tiempo de ejecución fue de {$execution\_time} milisegundos";

?>

Complejidad del Algoritmo:

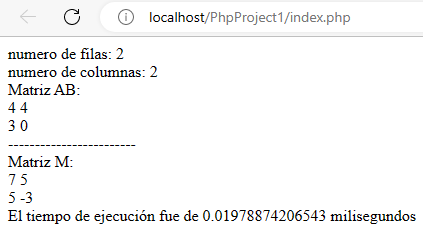
\* Complejidad total es O(a\*b)

\* La ecuación de complejidad de este algoritmo sería: T(n) = O(ab)

\* La función de complejidad del algoritmo es: f(a,b) = 9ab + 11a + 6

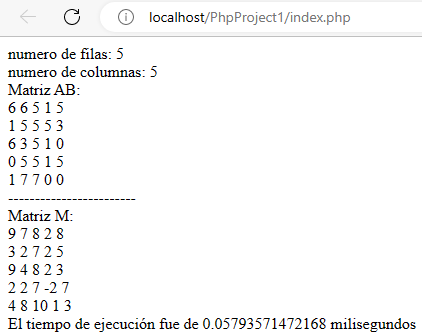
**Casos de prueba Estudiante 1**

* Matriz 2x2



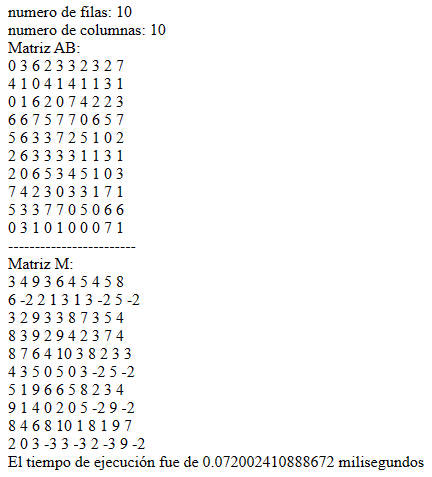
0.01978874206543 / 1000 = 0.00001978874206543 Segundos

* Matriz 5x5



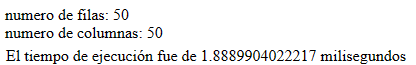
0.05793571472168 / 1000 = 0.00005793571472168 segundos

* Matriz 10x10



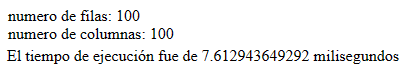
0.072002410888672 / 1000 = 0.000072002410888672 segundos

* Matriz 50x50



1.8970966339111 / 1000 = 0.0018970966339111 segundos

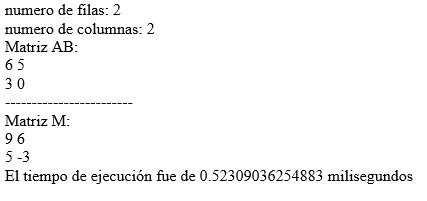
* Matriz 100x100



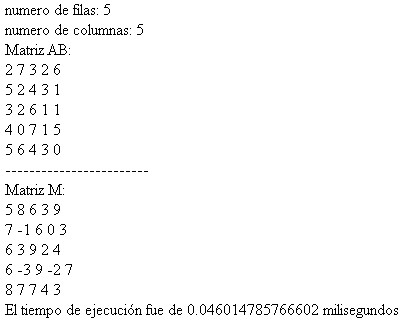
7.612943649292 / 1000 = 0.007612943649292 segundos

**Casos de prueba Estudiante 2**

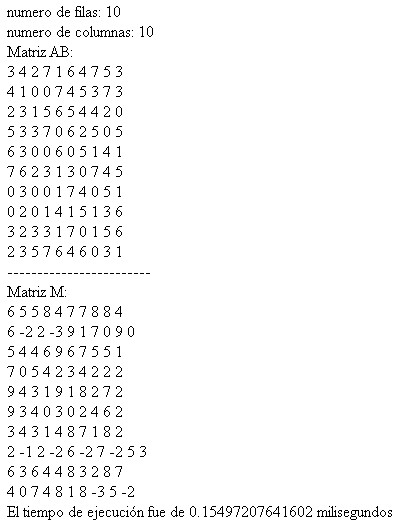
* Matriz 2x2



* Matriz 5x5



* Matriz 10x10



* Matriz 50x50





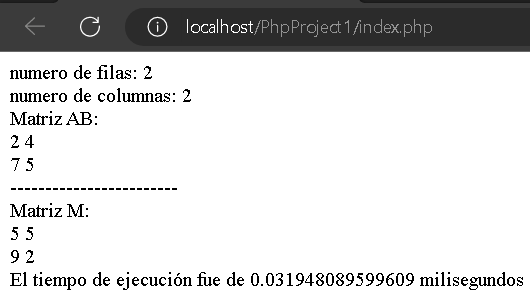
* Matriz 100x100





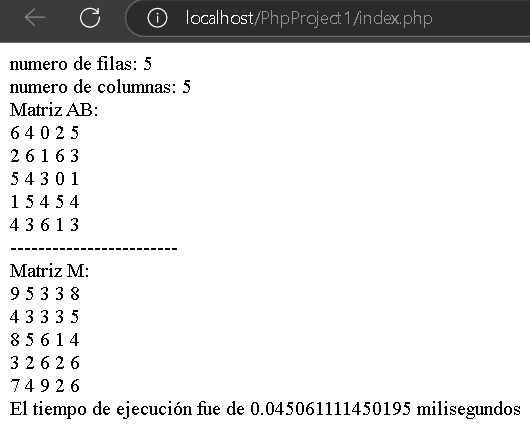
**Casos de prueba Estudiante 3**

* Matriz 2x2



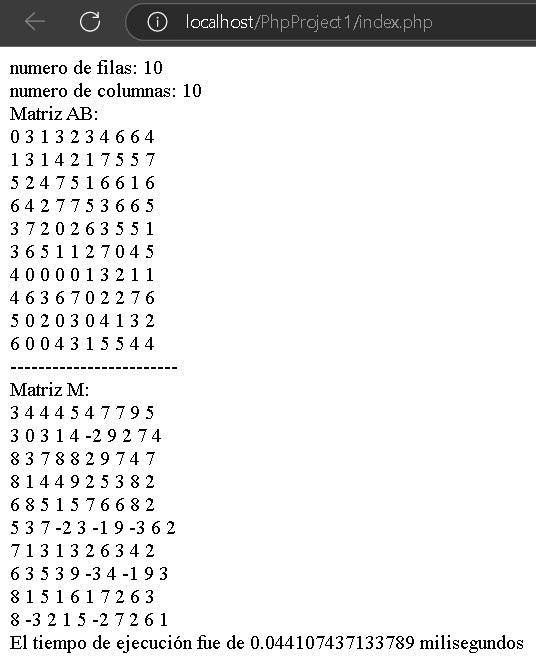
\* 0.031948089599609 ms / 1000 = 0.0000319481 segundos

* Matriz 5x5



\* 0.045061111450195 / 1000 = 0.0000450611 segundos

* Matriz 10x10



\* 0.044107437133789 / 1000 = 0.0000441074 segundos

* Matriz 50x50



\* 0.94103813171387 / 1000 = 0.00094104 segundos

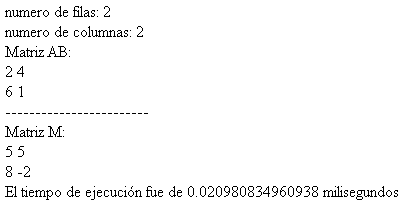
* Matriz 100x100



\* 6.486177444458 / 1000 = 0.00648618 segundos

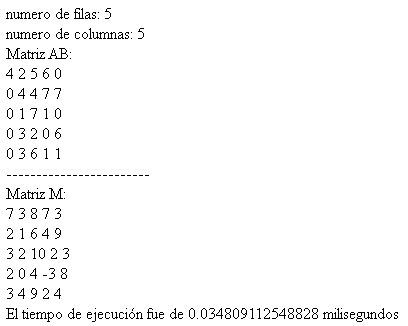
**Casos de prueba Estudiante 4**

* Matriz 2x2



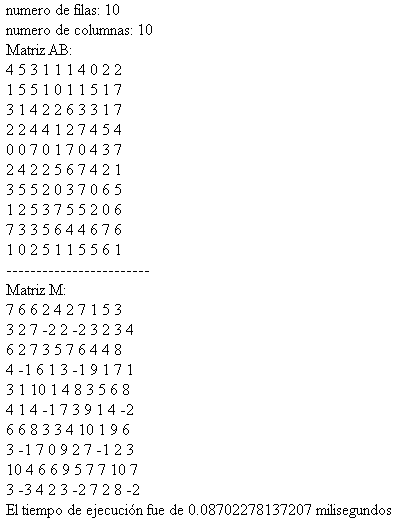
0,020981 / 1000 = 0,000020981 segundos

* Matriz 5x5



0,034809 / 1000 = 0,000034809 segundos

* Matriz 10x10



0,087023 / 1000 = 0,000087023 segundos

* Matriz 50x50



2,604961 / 1000 = 0,002604961 segundos

* Matriz 100x100



8,338928 / 1000 = 0,008338928 segundos

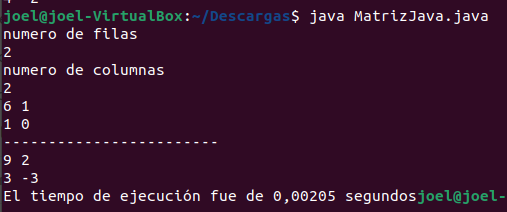
**DOCUMENTACIÓN CASOS DE PRUEBA (LINUX)**

**Sistema Operativo: Linux**

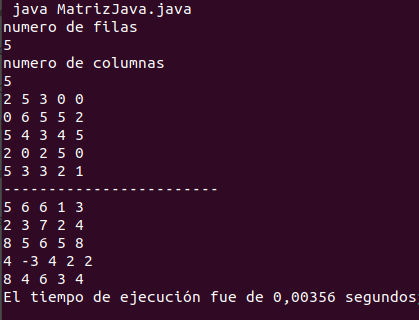
**Lenguaje: Java**

**Casos de prueba Estudiante 1**

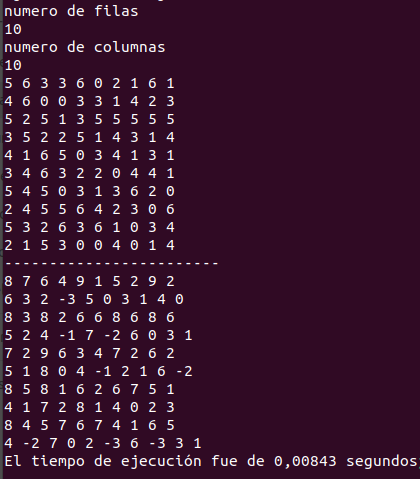
* Matriz 2x2



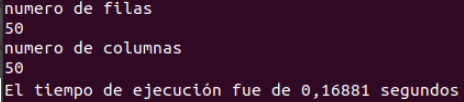
* Matriz 5x5



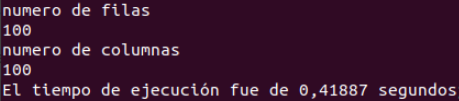
* Matriz 10x10



* Matriz 50x50

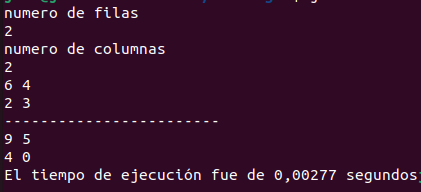


* Matriz 100x100

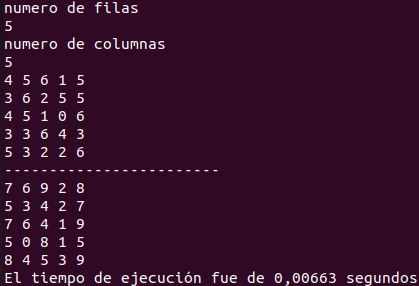


**Casos de prueba Estudiante 2**

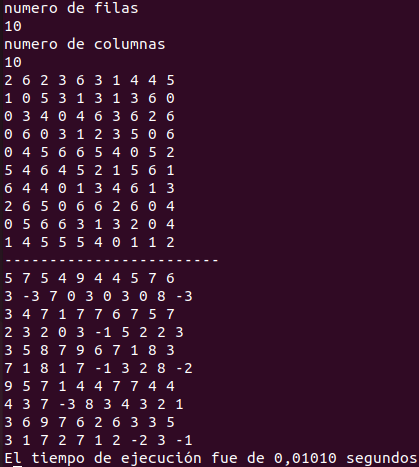
* Matriz 2x2



* Matriz 5x5



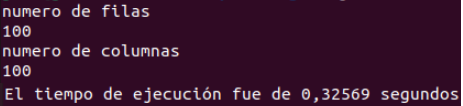
* Matriz 10x10



* Matriz 50x50

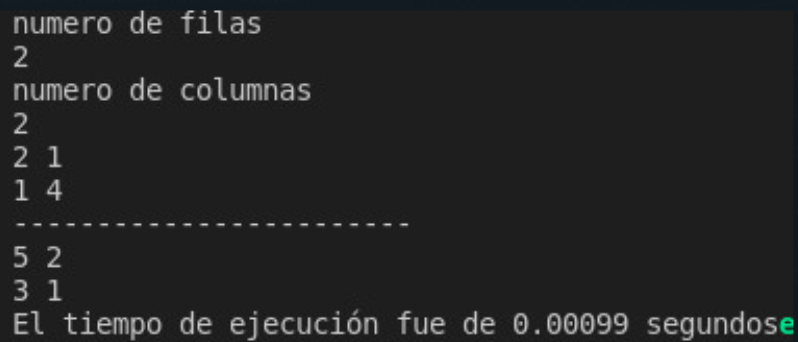


* Matriz 100x100

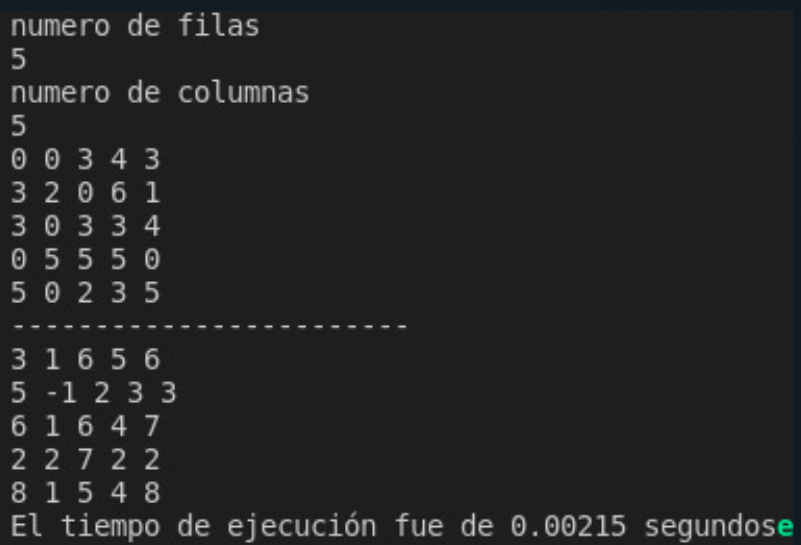


**Casos de prueba Estudiante 3**

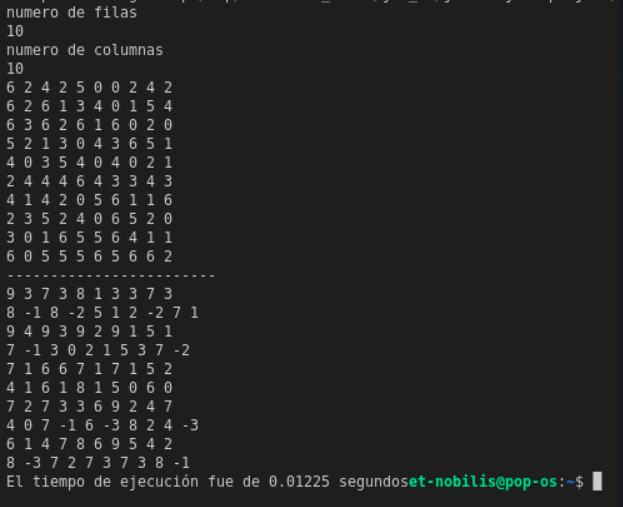
* Matriz 2x2



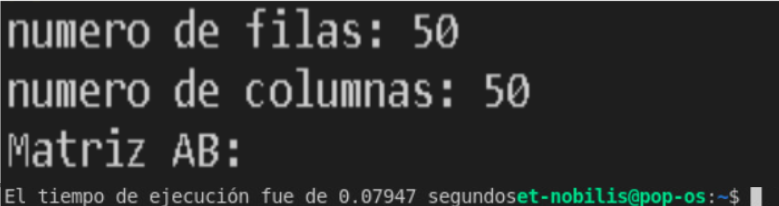
* Matriz 5x5



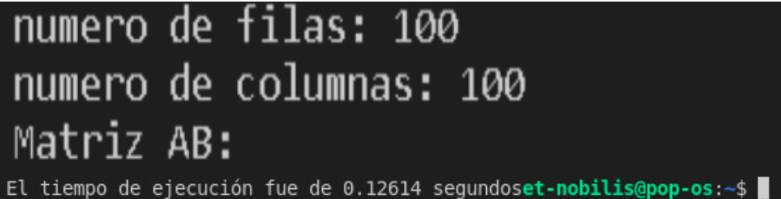
* Matriz 10x10



* Matriz 50x50



* Matriz 100x100



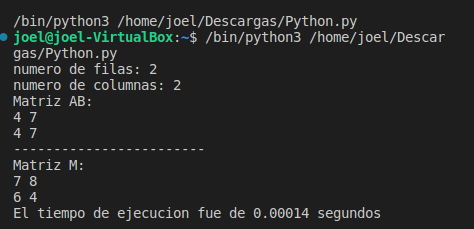
**Casos de prueba Estudiante 4**

* Matriz 2x2
* Matriz 5x5
* Matriz 10x10
* Matriz 50x50
* Matriz 100x100

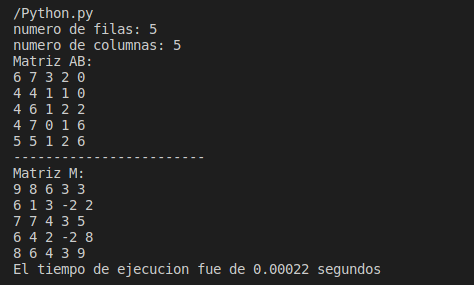
**Lenguaje: Python**

**Casos de prueba Estudiante 1**

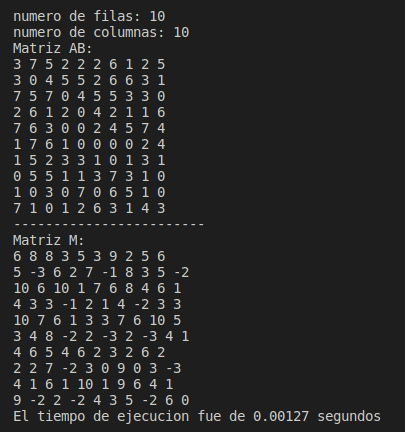
* Matriz 2x2



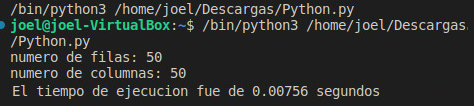
* Matriz 5x5



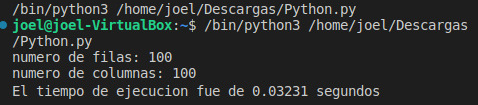
* Matriz 10x10



* Matriz 50x50

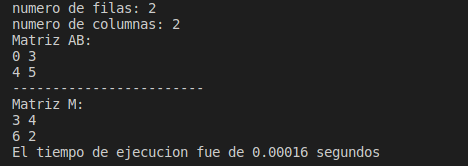


* Matriz 100x100

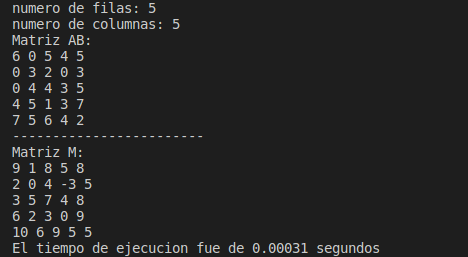


**Casos de prueba Estudiante 2**

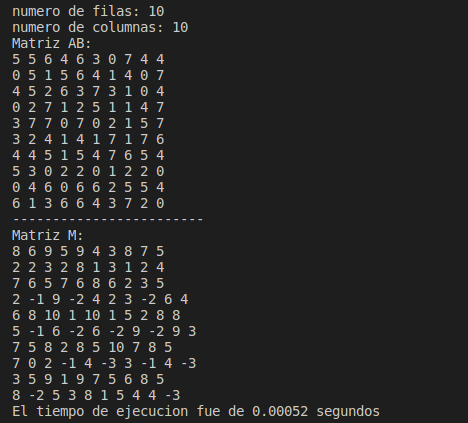
* Matriz 2x2



* Matriz 5x5



* Matriz 10x10



* Matriz 50x50

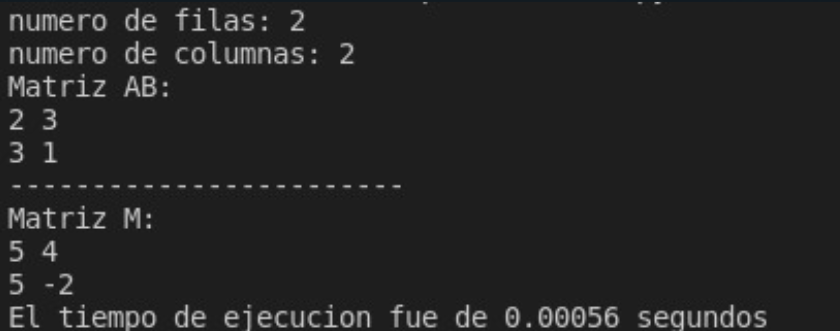


* Matriz 100x100

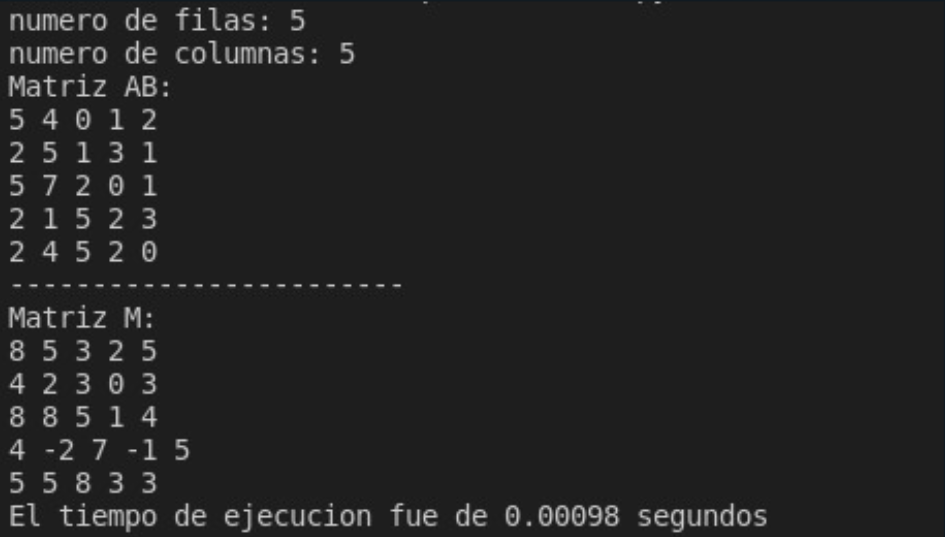


**Casos de prueba Estudiante 3**

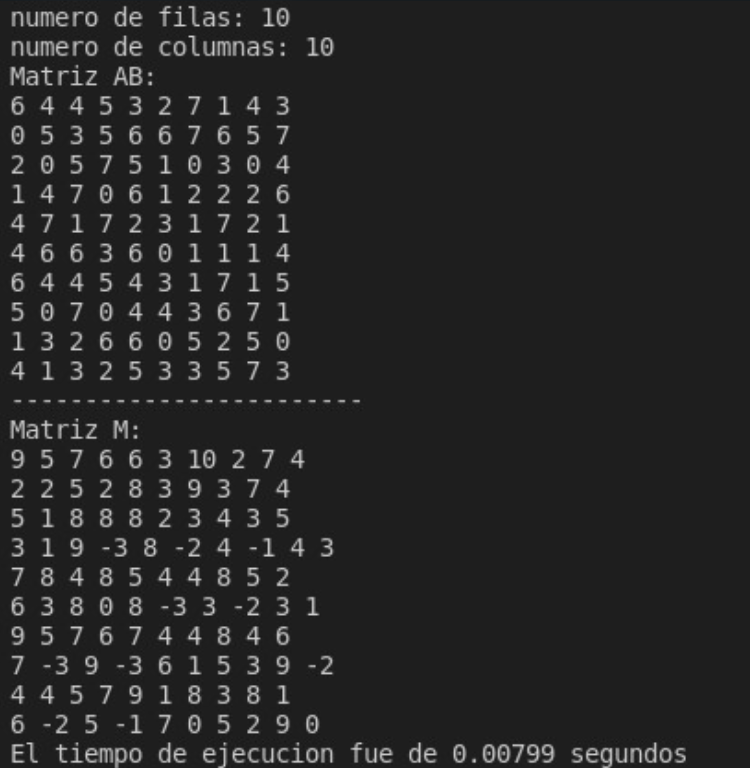
* Matriz 2x2



* Matriz 5x5



* Matriz 10x10



* Matriz 50x50

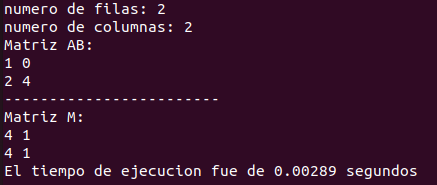


* Matriz 100x100

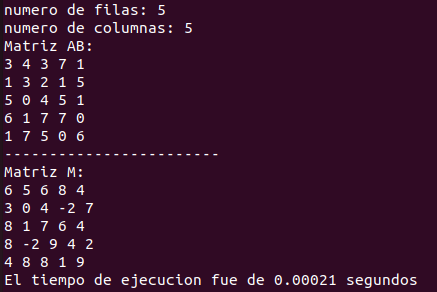


**Casos de prueba Estudiante 4**

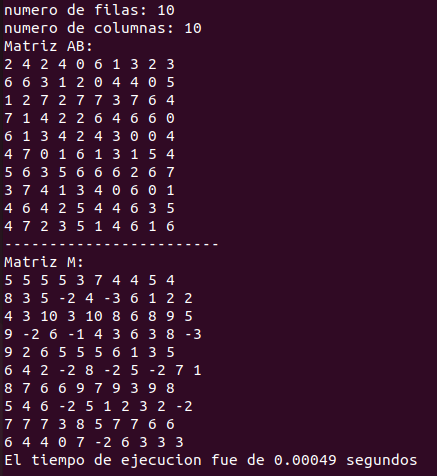
* Matriz 2x2



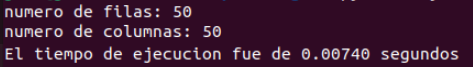
* Matriz 5x5



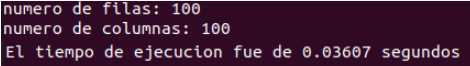
* Matriz 10x10



* Matriz 50x50



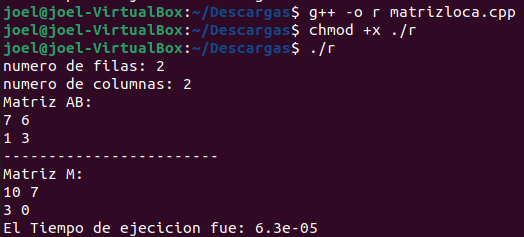
* Matriz 100x100



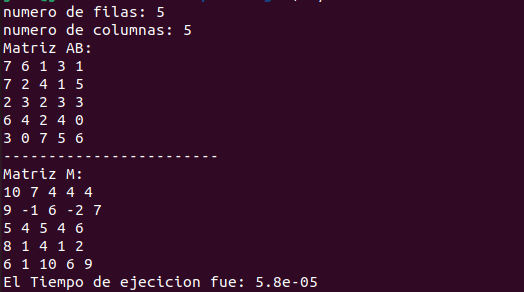
**Lenguaje: C++**

**Casos de prueba Estudiante 1**

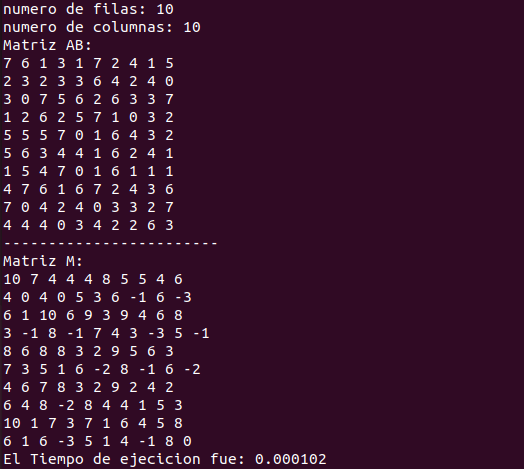
* Matriz 2x2



* Matriz 5x5



* Matriz 10x10



* Matriz 50x50



* Matriz 100x100



**Casos de prueba Estudiante 2**

* Matriz 2x2
* Matriz 5x5
* Matriz 10x10
* Matriz 50x50
* Matriz 100x100

**Casos de prueba Estudiante 3**

* Matriz 2x2
* Matriz 5x5
* Matriz 10x10
* Matriz 50x50
* Matriz 100x100

**Casos de prueba Estudiante 4**

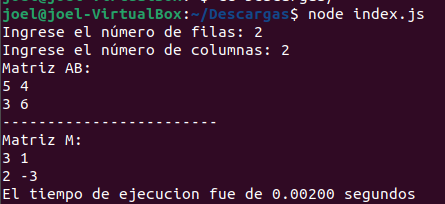
* Matriz 2x2
* Matriz 5x5
* Matriz 10x10
* Matriz 50x50
* Matriz 100x100

**Lenguaje: Javascript**

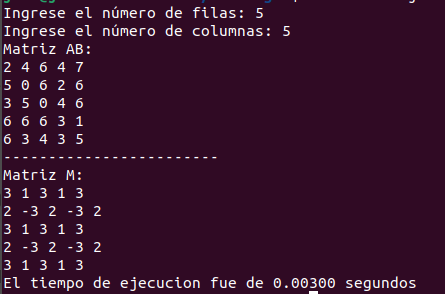
**Node.js**

**Casos de prueba Estudiante 1**

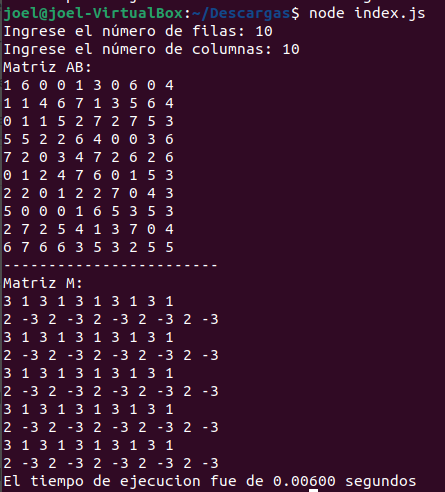
* Matriz 2x2



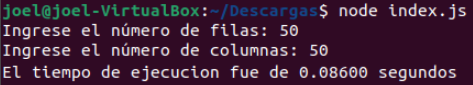
* Matriz 5x5



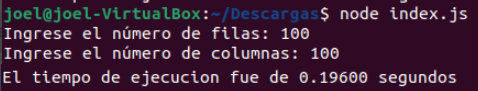
* Matriz 10x10



* Matriz 50x50

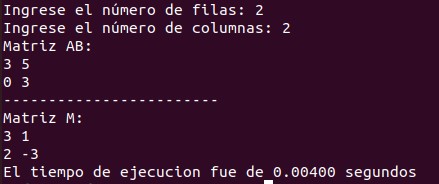


* Matriz 100x100

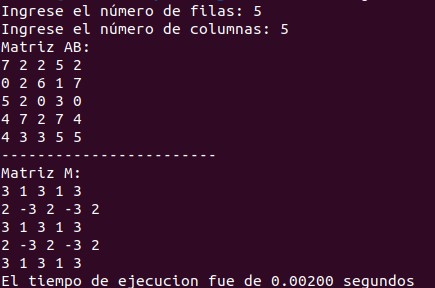


**Casos de prueba Estudiante 2**

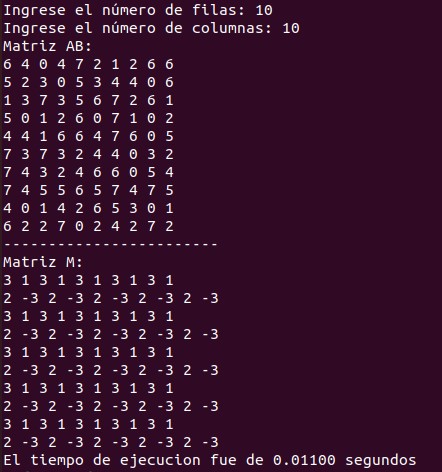
* Matriz 2x2



* Matriz 5x5



* Matriz 10x10



* Matriz 50x50





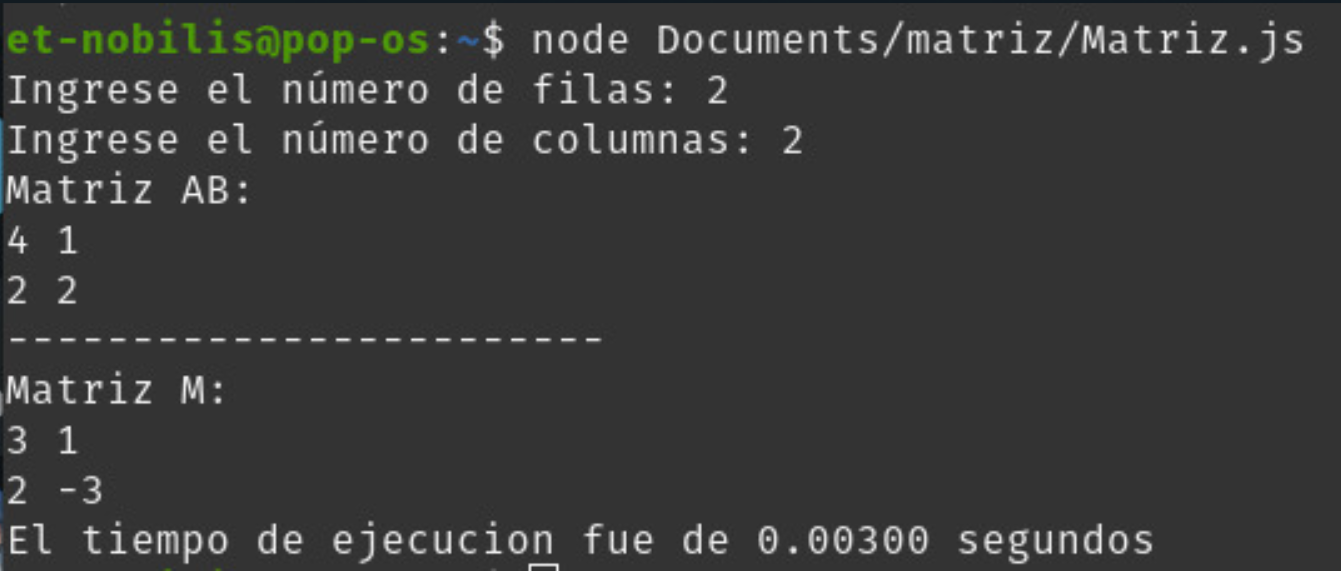
* Matriz 100x100



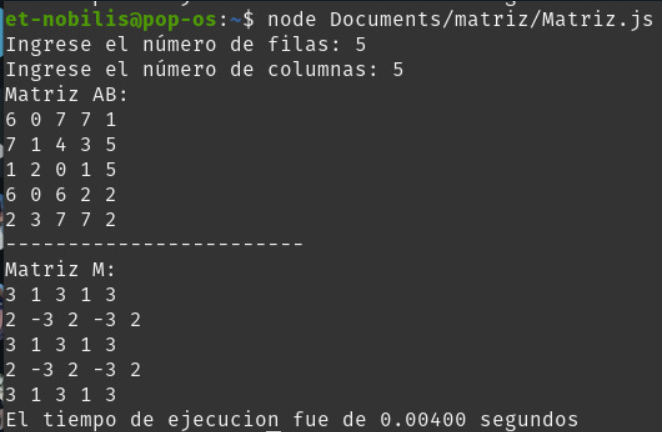


**Casos de prueba Estudiante 3**

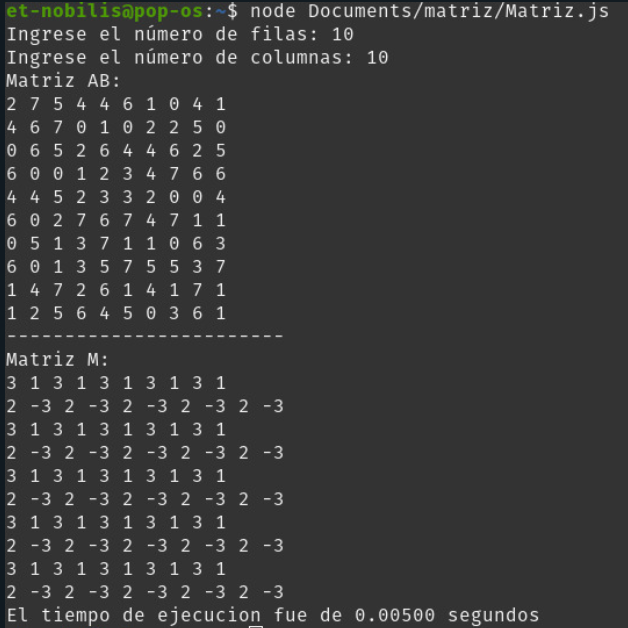
* Matriz 2x2



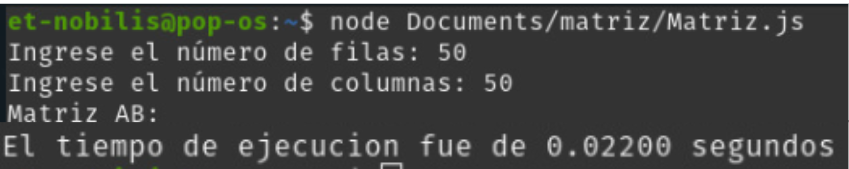
* Matriz 5x5



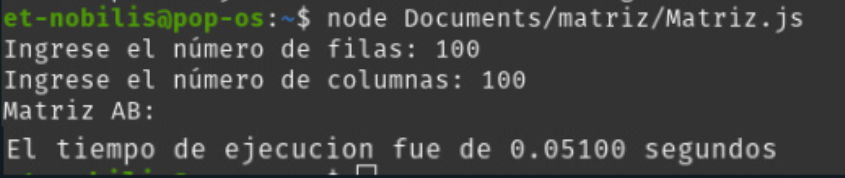
* Matriz 10x10



* Matriz 50x50



* Matriz 100x100



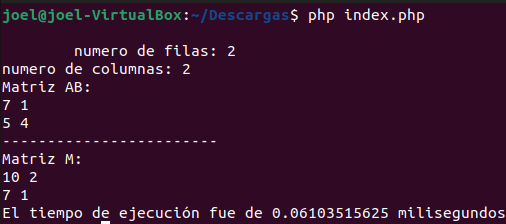
**Casos de prueba Estudiante 4**

* Matriz 2x2
* Matriz 5x5
* Matriz 10x10
* Matriz 50x50
* Matriz 100x100

**Lenguaje: PHP**

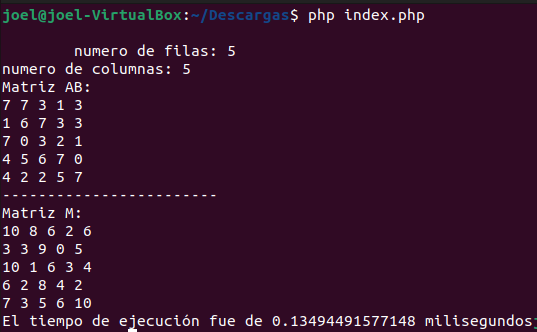
**Casos de prueba Estudiante 1**

* Matriz 2x2



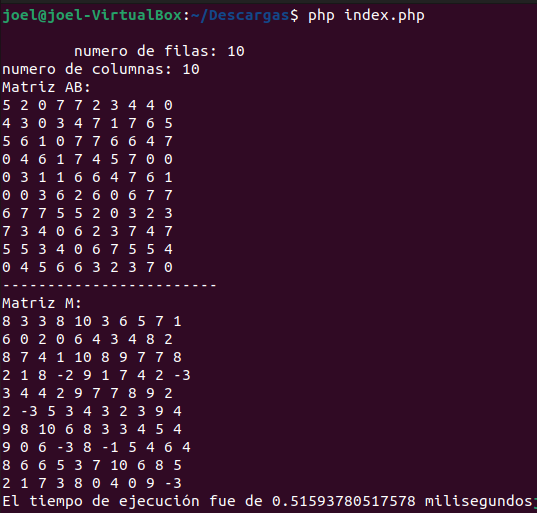
0.061035 / 1000 = 0.000061035 segundos

* Matriz 5x5



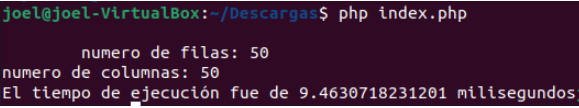
0.134944 / 1000 = 0.000134944 segundos

* Matriz 10x10



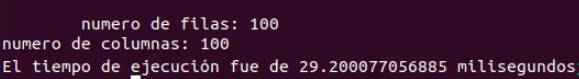
0.515937 / 1000 = 0.000515937 segundos

* Matriz 50x50



9.46307 / 1000 = 0.00946307 segundos

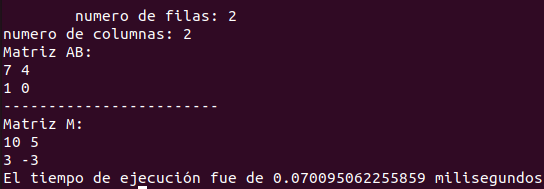
* Matriz 100x100



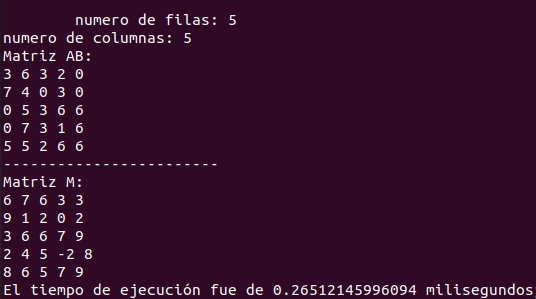
29.20007 / 1000 = 0.02920007 segundos

**Casos de prueba Estudiante 2**

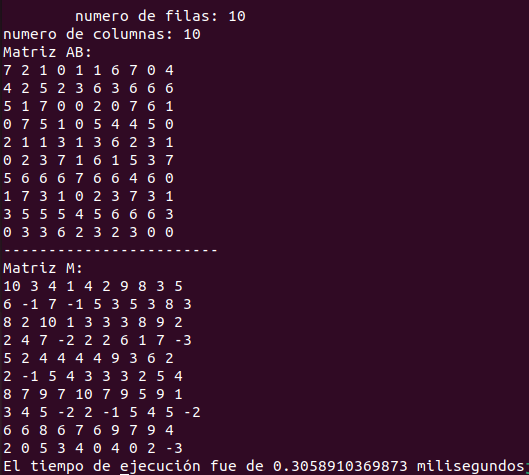
* Matriz 2x2



* Matriz 5x5



* Matriz 10x10



* Matriz 50x50





* Matriz 100x100

**Casos de prueba Estudiante 3**

* Matriz 2x2
* Matriz 5x5
* Matriz 10x10
* Matriz 50x50
* Matriz 100x100
* Matriz 2x2
* Matriz 5x5
* Matriz 10x10
* Matriz 50x50
* Matriz 100x100

**Conclusión**

Se implementó la misma solución en cinco lenguajes de programación diferentes y en dos sistemas operativos. Se midió el tiempo de ejecución de cada una de las implementaciones utilizando cinco escenarios de datos aleatorios de distintos tamaños. Se registraron los datos en una hoja de cálculo y se analizó la complejidad del algoritmo utilizado.

Se concluye que el tiempo empleado para ejecutar el algoritmo se verá afectado por varios factores, uno de estos siendo el sistema operativo, ya que es posible que algunos sistemas operativos sean más eficientes que otros para ejecutar algoritmos.

Otro factor es el lenguaje de programación, algunos lenguajes de programación pueden ser más adecuados para algoritmos específicos.

El rendimiento del hardware de cada computador también contribuyen de cierta forma en la eficiencia de ejecución.

La cantidad y el tamaño de los datos utilizados en cada caso puede tener un impacto significativo en el tiempo empleado para ejecutar el algoritmo, en cuanto más complejos sean estos datos, el tiempo de ejecución será mayor.