Nombre: Miguel Angel Cárdenas Díaz **Código:** 1151929

```
public boolean[][] getMultiplicacion_IA() {
    int filasM1 = m1.length;                      // 2 (1 acceso + 1 asignación)
    int columnasM1 = m1[0].length;                // 3 (2 accesos + 1 asignación)
    int columnasM2 = m2[0].length;                // 3 (2 accesos + 1 asignación)

    boolean[][] resultado = new boolean[filasM1][columnasM2]; // 2 (creación + asignación)

    for (int i = 0; i < filasM1; i++) {           // 1 (i=0) + Σ [de i=1 a n] (2: comparación + i++)
        for (int j = 0; j < columnasM2; j++) {    // 1 (j=0) + Σ [de j=1 a m] (2)
            boolean suma = false;                 // 1
            for (int k = 0; k < columnasM1; k++) { // 1 (k=0) + Σ [de k=1 a p] (2)
                suma ^= (m1[i][k] && m2[k][j]);   // 5 (2 accesos + AND + XOR + asignación)
            }
            resultado[i][j] = suma;               // 2 (acceso + asignación)
        }
    }
    return resultado;                             // 1
}
```

n = filasM1, m = columnasM2, p = columnasM1

$T(n) = 10 + 1 + 2n + \Sigma$ [de i=1 a n] $(1 + 2m + \Sigma$ [de j=1 a m] $(1 + 2p + 7p + 2))$
$= 11 + 2n + \Sigma$ [de i=1 a n] $(1 + 2m + m(1 + 9p + 2))$
$= 11 + 2n + \Sigma$ [de i=1 a n] $(1 + 2m + 9mp + 3m)$
$= 11 + 2n + n + 5\Sigma m + 9\Sigma mp$

$T(n) \approx 11 + 3n + 5n^2 + 9n^3$
Big-O: $O(n^3)$

(el viejo)

```java
public boolean[][] getMultiplicacion() {
   int filasM1 = m1.length;                                   // 2
   int columnasM1 = m1[0].length;                         // 3
   int columnasM2 = m2[0].length;                        // 3

   boolean[][] resultado = new boolean[filasM1][columnasM2];    // 2

   for (int i = 0; i < filasM1; i++) {                     // 1 + Σ [de i=1 a n] (2)
      for (int j = 0; j < columnasM2; j++) {              // 1 + Σ [de j=1 a m] (2)
         int suma = 0;                                          // 1
         for (int k = 0; k < columnasM1; k++) {         // 1 + Σ [de k=1 a p] (2)
            if (m1[i][k] && m2[k][j]) {                     // 4 (2 accesos + AND + comparación if)
               suma ^= 1;                                       // 2 (XOR + asignación)
            }
         }
         resultado[i][j] = (suma == 1);                    // 3 (comparación + acceso + asignación)
      }
   }
   return resultado;                                          // 1
}
```

n = filasM1, m = columnasM2, p = columnasM1

T(n) = 10 + 1 + 2n + Σ [de 1 a n] (1 + 2m + Σ [de 1 a m] (1 + 2p + 6 + 3))
= 11 + 2n + Σ [de 1 a n] (1 + 2m + m(1 + 2p + 9))
= 11 + 2n + Σ [de 1 a n] (1 + 2m + 2mp + 10m)
= 11 + 2n + n + 12Σm + 2Σmp

$T(n) \approx 11 + 3n + 12n^2 + 2n^3$

Big-O: $O(n^3)$

(El nuevo)

Argumento:

Es verdad que ambas terminan siendo $O(n^3)$ y aunque las dos soluciones son correctas, la mía usa operaciones más sencillas (números en vez de booleanos) que la computadora procesa más rápido. Es como si resolvieras un problema matemático: ambos llegamos a la respuesta correcta, pero yo uso un atajo que me ahorra pasos. Para matrices grandes, estos pequeños ahorros suman y hacen que mi programa corra más rápido. Además, al ser más simple, es más fácil mejorarla después si fuera necesario.