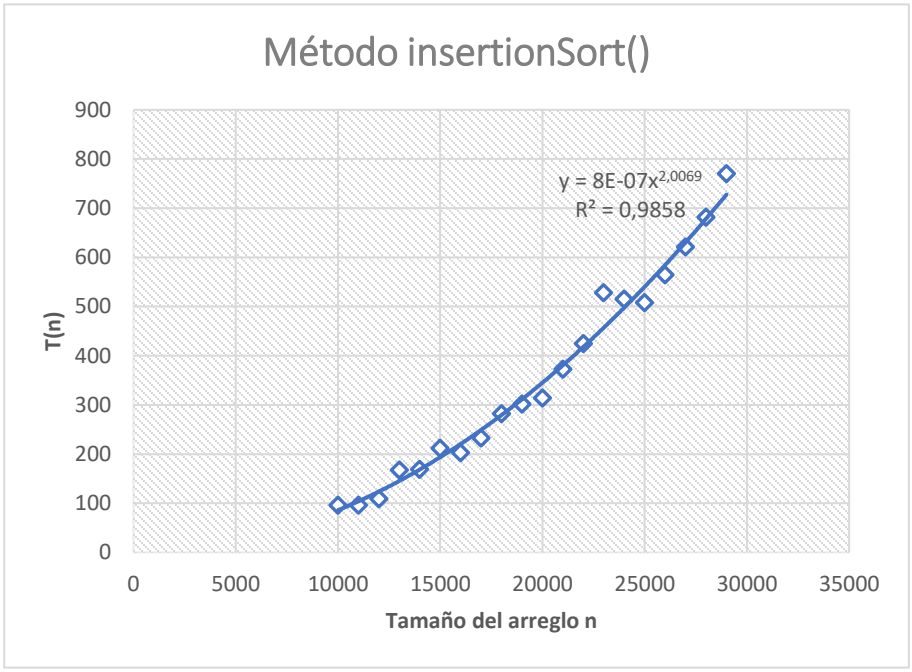
PRUEBAS Y GRÁFICOS – *Taller 5 ED1* *Presentado por: Jacobo Rave y Sebastián Guerra*

- **Método insertionSort()**

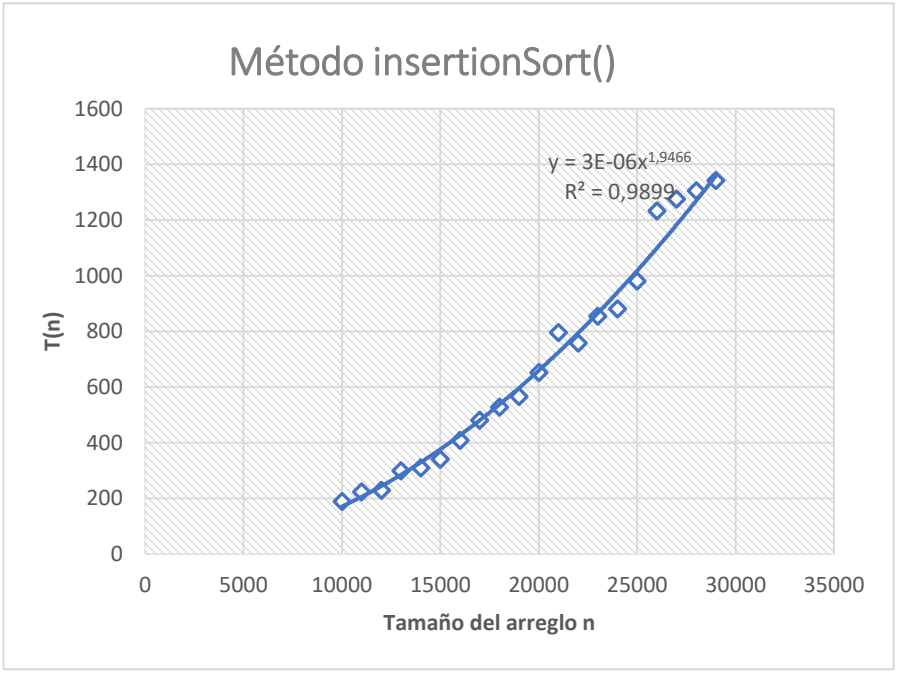
PRUEBA 1: Caso aleatorio (Arreglo con valores aleatorios)

Tamaño del arreglo n	T(n)
10000	96
11000	96
12000	109
13000	168
14000	169
15000	212
16000	203
17000	233
18000	282
19000	302
20000	314
21000	373
22000	425
23000	528
24000	515
25000	508
26000	565
27000	621
28000	682
29000	770



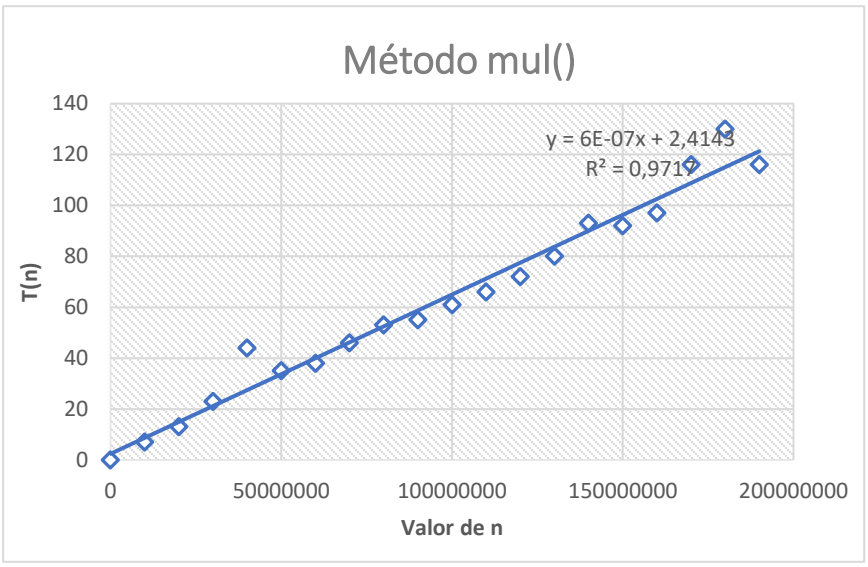
PRUEBA 2: Peor caso (Ordenados de mayor a menor)

Tamaño del arreglo n	T(n)
10000	188
11000	223
12000	229
13000	299
14000	309
15000	340
16000	408
17000	481
18000	528
19000	565
20000	652
21000	795
22000	758
23000	854
24000	881
25000	980
26000	1232
27000	1275
28000	1305
29000	1341



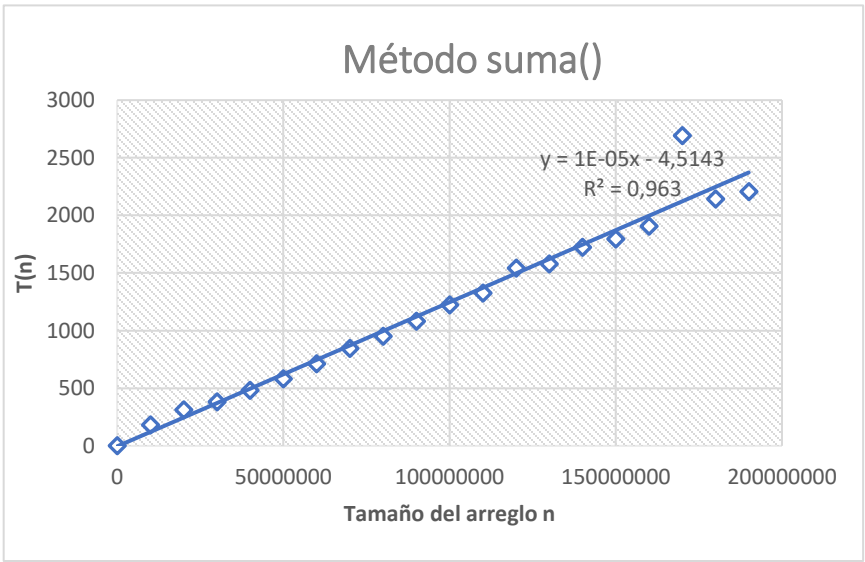
- **Método mul()**

Valor de n	T(n)
0	0
$1x10^7$	7
$2x10^7$	13
$3x10^7$	23
$4x10^7$	44
$5x10^7$	35
$6x10^7$	38
$7x10^7$	46
$8x10^7$	53
$9x10^7$	55
$1x10^8$	61
$1.1x10^8$	66
$1.2x10^8$	72
$1.3x10^8$	80
$1.4x10^8$	93
$1.5x10^8$	92
$1.6x10^8$	97
$1.7x10^8$	116
$1.8x10^8$	130
$1.9x10^8$	116



- **Método suma()**

Tamaño del arreglo n	T(n)
0	1
$1x10^7$	181
$2x10^7$	314
$3x10^7$	381
$4x10^7$	483
$5x10^7$	581
$6x10^7$	712
$7x10^7$	846
$8x10^7$	951
$9x10^7$	1083
$1x10^8$	1224
$1.1x10^8$	1326
$1.2x10^8$	1543
$1.3x10^8$	1578
$1.4x10^8$	1723
$1.5x10^8$	1796
$1.6x10^8$	1908
$1.7x10^8$	2693
$1.8x10^8$	2143
$1.9x10^8$	2208



ANÁLISIS Y CONCLUSIONES

- **Método insertionSort()**

Al encontrarnos este método podemos ver que hay un ciclo anidado. El ciclo externo tiene como límite la longitud del arreglo que queremos organizar (es decir, O(n)), mientras que el interno va a ir recorriendo los elementos anteriores a la posición actual del arreglo, por lo que va a ir aumentando en 1 su recorrido cada vez que se avanza en el primer ciclo (visto de forma equivalente, O(n-1)). Por lo tanto, para determinar la complejidad de este algoritmo se debe multiplicar lo anterior al tratarse de ciclos anidados:

$$n \cdot (n-1) = n^2 - n$$

Y, aplicando la propiedad de reflexividad y suma:

$$T(n)\ es\ O(n^2)$$

Ahora bien, siendo esta su complejidad, no es recomendable para el manejo de grandes cantidades de información debido al crecimiento potencial que implica.

- **Método mul()**

Este método esta formado por un solo ciclo, quien tiene como límite el valor n que se le ingresa. De esta forma, la notación asintótica del algoritmo va a ser de:

$$O(n)$$

Ahora bien, el comportamiento del gráfico se adapta muy bien a la complejidad lineal denotada. Por esa razón puedo afirmar que la implementación nos dio datos experimentales que corresponden con O(n).

- **Método suma()**

Al igual que el anterior, este método esta formado por un único ciclo que tiene como limite la longitud del arreglo, por lo que su complejidad va a ser de:

$$O(n)$$

Cabe recordar la implementación recursiva que se hizo de este método en talleres anteriores, cuya complejidad era de:

$$O(2^n)$$

De esta forma podemos estar seguros de que la implementación con ciclos tiene menor consumo de recursos que la implementación recursiva cuando la magnitud de los datos tiende a ser mayor, ya que la segunda implica un crecimiento exponencial del consumo de recursos.