Informe Taller Complejidad 2

Simón Álvarez
David Madrid

August 2020

1 Insertion Sort

Para calcular el número de operaciones necesarias $T(n)$, donde n es la longitud del arreglo, utilizamos la fórmula de los ciclos y obtendremos que: $T(n) = n(c_1 + n(c_2)) + c_3 = c_1 \cdot n + c_2 \cdot n^2 + c_3$, c_n constante.

Con esta fórmula podemos calcular la complejidad del problema. Por reflexividad, $T(n)$ es $O(T(n)) := O(c_1 \cdot n + c_2 \cdot n^2 + c_3)$. Por regla de la suma, obtenemos que lo anterior es equivalente a $O(c_2 \cdot n^2)$ y, por la regla del producto, concluimos que la complejidad final es $O(n^2)$ en el peor de los casos [1] [2].

InsertionSort			
Java		Python	
Length	Time	Length	Time
25000	0.103	2500	0.481
50000	0.426	5000	1.863
75000	0.949	7500	4.211
100000	1.683	10000	7.525
125000	2.603	12500	11.700
150000	3.746	15000	16.663
175000	5.09	17500	22.705
200000	6.733	20000	29.681
225000	8.52	22500	37.754
250000	10.444	25000	46.801
275000	12.726	27500	56.390
300000	15.106	30000	67.153
325000	18.159	32500	78.486
350000	20.667	35000	91.812
375000	23.76	37500	105.517
400000	27.088	40000	119.375
425000	30.683	42500	135.142
450000	34.761	45000	152.022
475000	38.634	47500	168.482
500000	42.662	50000	188.076
525000	47	52500	205.793

Table 1: Time required to sort elements of an array.

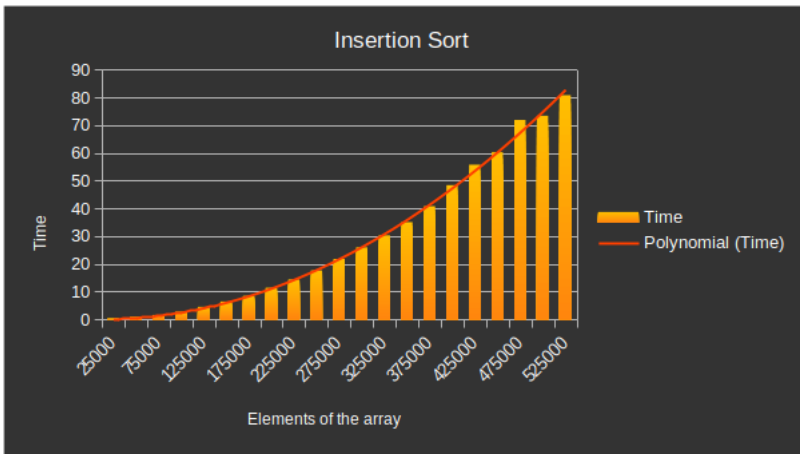

Figure 1: Graph Dataset 1 Java

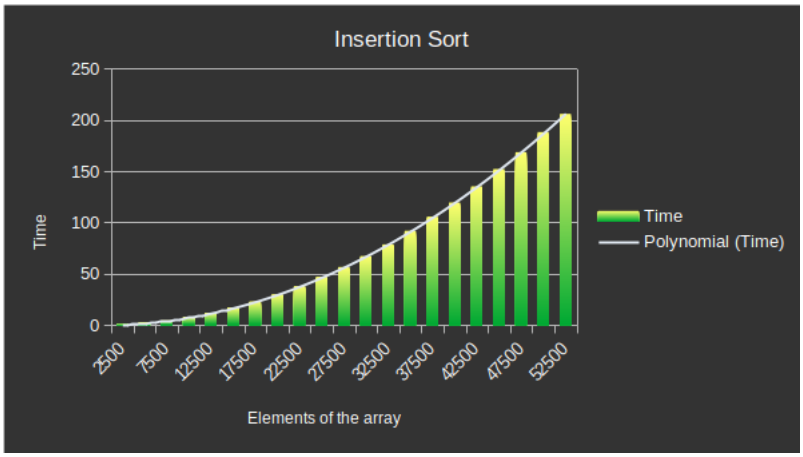

Figure 2: Graph Dataset 1 Python

2 Suma Elementos en un Arreglo

Para calcular el número de operaciones necesarias $T(n)$, donde n es la longitud del arreglo, utilizamos la fórmula de los ciclos y obtendremos que: $T(n) = n \cdot (c_1 + c_2) + c_3 + c_4 + c_5 = k_1 \cdot n + k_2$, c_n y k_n constantes.

Con esta fórmula podemos calcular la complejidad del problema. Por reflexividad, $T(n)$ es $O(T(n)) := O(k_1 \cdot n + k_2)$. Por regla de la suma, obtenemos que lo anterior es equivalente a $O(k_1 \cdot n)$ y, por la regla del producto, concluimos que la complejidad final es $O(n)$ en el peor de los casos [3] [4].

SumArray			
Java		Python	
Length	Time	Length	Time
7.40E+07	0.073	7.40E+06	0.305
1.48E+08	0.107	1.48E+07	0.586
2.22E+08	0.14	2.22E+07	0.872
2.96E+08	0.201	2.96E+07	1.155
3.70E+08	0.212	3.70E+07	1.440
4.44E+08	0.281	4.44E+07	1.729
5.18E+08	0.317	5.18E+07	2.040
5.92E+08	0.355	5.92E+07	2.335
6.66E+08	0.393	6.66E+07	2.666
7.40E+08	0.432	7.40E+07	2.936
8.14E+08	0.471	8.14E+07	3.271
8.88E+08	0.516	8.88E+07	3.577
9.62E+08	0.558	9.62E+07	3.879
1.04E+09	0.585	1.04E+08	4.168
1.11E+09	0.625	1.11E+08	4.471
1.18E+09	0.661	1.18E+08	4.759
1.26E+09	0.704	1.26E+08	5.049
1.33E+09	0.74	1.33E+08	5.258
1.41E+09	0.785	1.41E+08	5.832
1.48E+09	0.821	1.48E+08	5.877
1.55E+09	0.865	1.55E+08	6.042

Table 2: Time required to sum the integers of an array with different lengths.

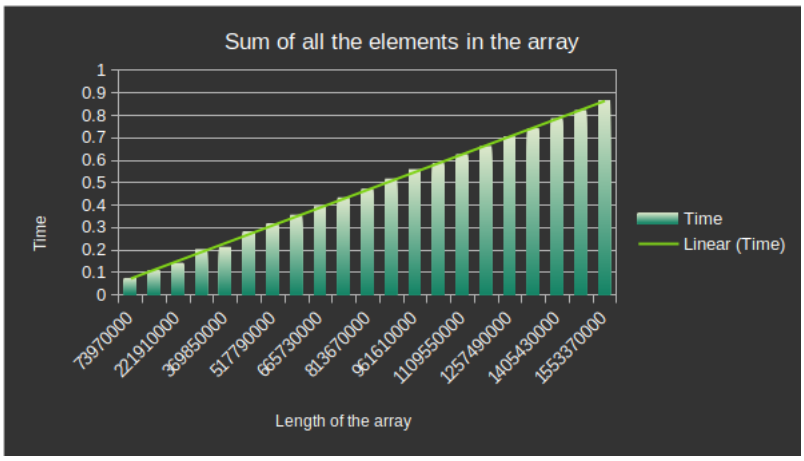

Figure 3: Graph Dataset Java

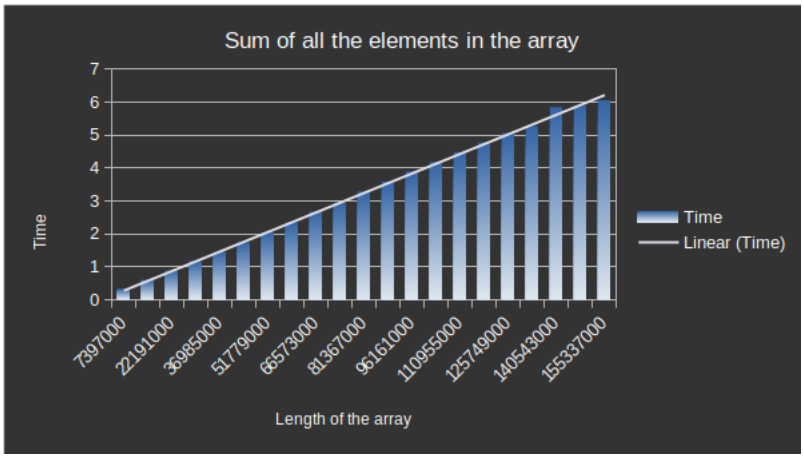

Figure 4: Graph Dataset Python

3 Multiplicación

Para calcular el número de operaciones necesarias $T(n)$, donde n es el límite para el cual se va a calcular la multiplicación, utilizamos la fórmula de los ciclos y obtendremos que: $T(n) = c_1 \cdot n + c_2$, c_n constante.

Con esta fórmula podemos calcular la complejidad del problema. Por reflexividad, $T(n)$ es $O(T(n)) := O(c_1 \cdot n + c_2)$. Por regla de la suma, obtenemos que lo anterior es equivalente a $O(c_1 \cdot n)$ y, por la regla del producto, concluimos que la complejidad final es $O(n)$ [5] [6].

Multiplicación			
Java		Python	
Length	Time	Length	Time
2400	0.031	3600	1.407
4800	0.052	4200	1.896
7200	0.138	4800	2.476
9600	0.174	5400	3.133
12000	0.306	6000	3.874
14400	0.396	6600	4.690
16800	0.558	7200	5.598
19200	0.743	7800	6.558
21600	0.883	8400	7.601
24000	1.485	9000	8.742
26400	1.858	9600	9.927
28800	2.23	10200	11.204
31200	2.403	10800	12.642
33600	2.741	11400	14.061
36000	3.244	12000	15.529
38400	3.641	12600	17.136
40800	4.266	13200	18.794
43200	4.14	13800	20.540
45600	4.792	14400	22.416
48000	5.45	15000	24.435
50400	5.647	15600	26.569

Table 3: Time required to calculate the numbers in a multiplication table of $n \cdot n$.

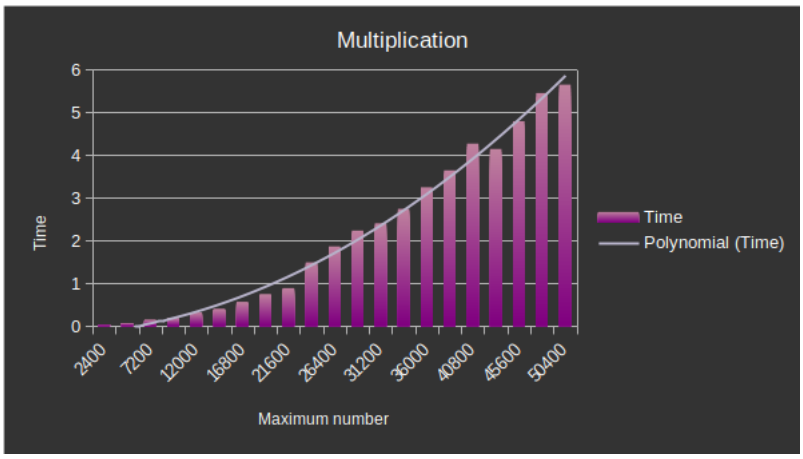

Figure 5: Graph Dataset 3 Java

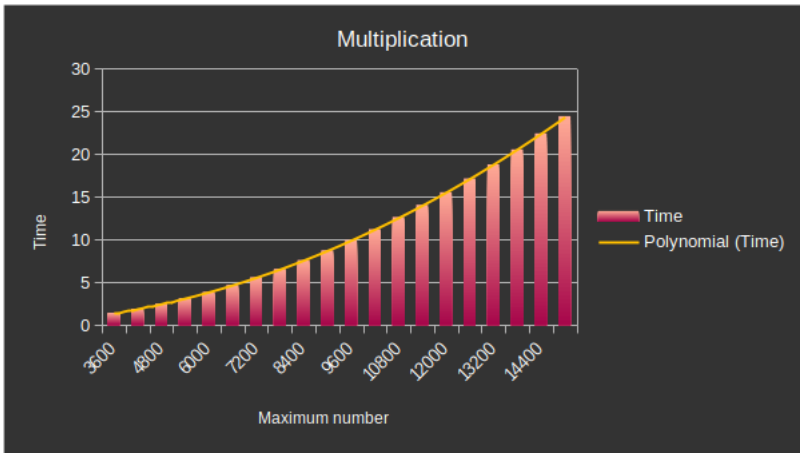

Figure 6: Graph Dataset 3 Python

References

[1] S. Álvarez and D. Madrid. (). Insertion Sort Java, [Online]. Available: https://github.com/dmadridr/ST0245-002/blob/master/talleres/taller05/Java%20Language/InsertionSort.java.

[2] ——, (). Insertion Sort Python, [Online]. Available: https://github.com/dmadridr/ST0245-002/blob/master/talleres/taller05/Python%20Language/InsertionSort.py.

[3] ——, (). Sum Elements in Array Java, [Online]. Available: https://github.com/dmadridr/ST0245-002/blob/master/talleres/taller05/Java%20Language/SumArray.java.

[4] ——, (). Sum Elemens in Array Python, [Online]. Available: https://github.com/dmadridr/ST0245-002/blob/master/talleres/taller05/Python%20Language/SumArrays.py.

[5] ——, (). Multiplication in Java, [Online]. Available: https://github.com/dmadridr/ST0245-002/blob/master/talleres/taller05/Java%20Language/Tables.java.

[6] ——, (). Multiplication in Python, [Online]. Available: https://github.com/dmadridr/ST0245-002/blob/master/talleres/taller05/Python%20Language/Tables.py.