Laboratory practice No. 2: Algorithm Complexity

Simón Álvarez Ospina
Universidad EAFIT
Medellín, Colombia
salvarezo1@eafit.edu.co

David Madrid Restrepo
Universidad EAFIT
Medellín, Colombia
dmadridr@eafit.edu.co

September 4, 2020

1 Report

i. For the code in 1.1 see [1].

ii. For the code in 2.1 and 2.2 see [2].

iii. *.

  i. *.

  ii. *.

Java			
Insertion Sort		Merge Sort	
Length	Time (s)	Length	Time (s)
10000	0.070553501	10000	0.061487301
20000	0.2288177	20000	0.2688838
30000	0.2117845	30000	0.2063037
40000	0.369743101	40000	0.369607699
50000	0.575291301	50000	0.6483491
60000	0.830919401	60000	0.83301
70000	1.1370137	70000	1.123927499
80000	1.4769437	80000	1.491584301
90000	1.8756835	90000	1.868646399
100000	2.302305099	100000	2.3518008
110000	2.8046215	110000	3.1470506
120000	3.391584699	120000	3.399865599
130000	3.9361273	130000	3.9739017
140000	4.5890445	140000	4.5784086
150000	5.270100401	150000	5.30329
160000	5.959113399	160000	5.9930441
170000	6.7503501	170000	6.7539809
180000	7.576194199	180000	7.5135541
190000	8.456903499	190000	8.3426125
200000	9.5745913	200000	9.3573335

Table 1: Time required to sort arrays with random values.

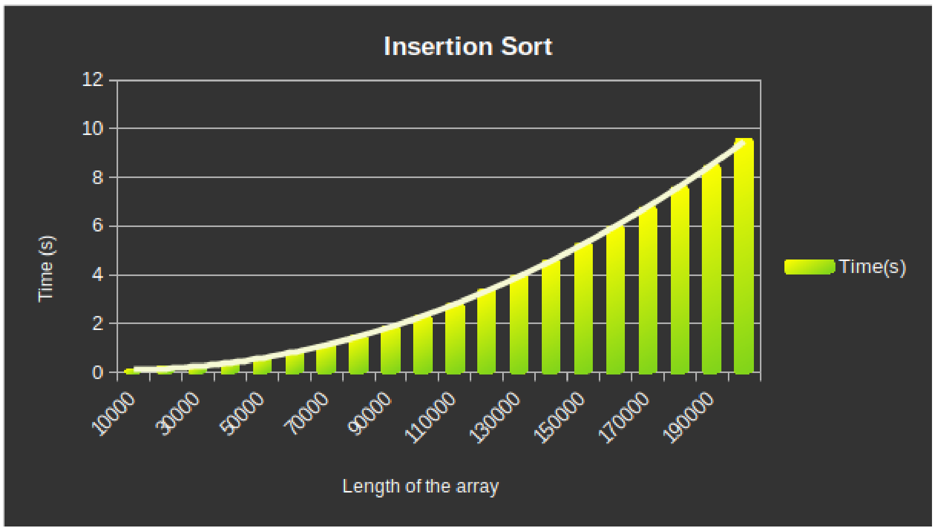

Figure 1: Graphic of Table 1, Insertion Sort

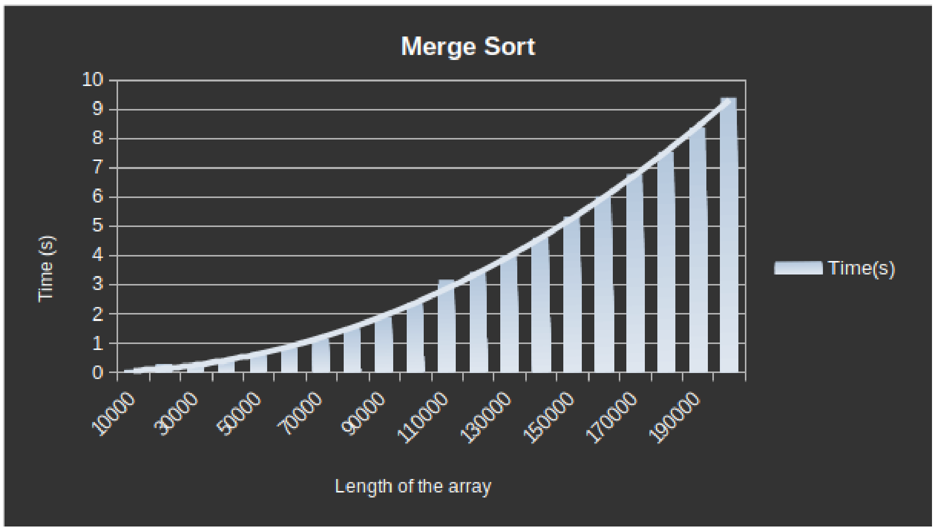

Figure 2: Graphic of Table 1, Merge Sort

iii. Suppose $T(n) = n^2$, where n is the length of the array in Insertion Sort. Suppose $n \geq 10^6$, then $T(n) \geq 10^{12}$. If an average computer makes $2.5 \cdot 10^9$ operations per second, then the time required to make $T(n)$ is greater than $\frac{10^3}{2.5} := 400s$. If we think that for every k millions more, the time required increases in $k^2 \cdot 400s$, then this isn't a good algorithm for big arrays.

iv. Merge Sort handelt von splitting the array in two parts recursively, doing the same until the length of all the parts is equal to 1. After that, the algorithm reassemble the array. Because every time splitting the array in two parts make $T(n) = c_1 + f(n) + k \cdot T(n/2)$, where $f(n)$ is a function and c_1, k are constants, which means that the original function depends of itself divided by a constant, hence, we can "guess" that the complexity will be of the form $T(n) = f(n) + g(n) \cdot log(n)$ [3].

v. *.

- **i.** The complexity of the following exercises in Array 2 [4] is the same for all of them, which is $O(n)$, where the meaning of n is the length of the array in all the cases.

  - **i.** **sum13**
  - **ii.** **sum28**
  - **iii.** **only14**
  - **iv.** **more14**
  - **v.** **fizzArray2**

ii. Array3 [5] is a more challenging situation, and despite there are some exercises that are $O(n^2)$ in complexity, there are some few others with two variables in their complexity.

   i. **maxSpan:** The complexity of this one is $O(n^2)$, where n is the length of the array.
   ii. **canBalance:** The complexity of this exercise is $O(n^2)$, where n is the length of the array.
   iii. **seriesUp:** Here, the complexity is $O(n^2)$ where n is the maximum number where the array will get.
   iv. **linearIn:** In this case, the complexity will be O(n,m), where n is the length of the array inner, and m is the length of the array outer.
   v. **fix34:** Finally, this last algorithm has a complexity of $O(n^2)$, where n is the length of the array.

2 Midterm Exam

i. *.

- **i.** *c*) $O(n+m)$.
- **ii.** *d*) $O(m \cdot n)$.
- **iii.** *b*) $O(width)$.
- **iv.** *b*) $O(n^3)$.
- **v.** *.
  - **i.** *d*) $T(n) = T(\frac{n}{10}) + c$, $O(log(n))$.
  - **ii.** *a*) Yes.
- **vi.** 10000 ms, 10 s.
- **vii.** *Everyone*.
- **viii.** *a*) $c + T(n-1)$.
- **ix.** *a*) $O(n^3)$.
- **x.** *c*) Executes less than $n \cdot log(n)$ steps.
- **xi.** *c*) Executes $T(n) = T(n-1) + T(n-2) + c$ steps.
- **xii.** *b*) $O(m \cdot n \cdot log(n) + m^2 \cdot n + n^2 \cdot log(n) + m^3$.
- **xiii.** *c*) $T(n) = 2 \cdot T(\frac{n}{2}) + n$.
- **xiv.** *c*) $O(m^{\frac{3}{2}} + n^3$.

References

[1] S. Álvarez and D. Madrid. (). For code 1.1, [Online]. Available: https://github.com/dmadridr/ST0245-002/tree/master/laboratorios/lab02/codigo/Java%20Language.

[2] ——, (). For code 2.1 and 2.2, [Online]. Available: https://github.com/dmadridr/ST0245-002/tree/master/laboratorios/lab02/ejercicioEnLinea.

[3] geeksforgeeks. (). Merge sort, [Online]. Available: https://www.geeksforgeeks.org/merge-sort/.

[4] Codingbat. (). Array 2 algorithms, [Online]. Available: https://github.com/dmadridr/ST0245-002/tree/master/laboratorios/lab02/ejercicioEnLinea/Array2.

[5] ——, (). Array 3 algorithms, [Online]. Available: https://github.com/dmadridr/ST0245-002/tree/master/laboratorios/lab02/ejercicioEnLinea/Array3.