Laboratory practice No. 2: Algorithm Complexity

Andres Felipe Agudelo Ortega
Universidad EAFIT
Medellín, Colombia
afagudeloo@eafit.edu.co

Juan Camilo Gutiérrez Urrego
Universidad EAFIT
Medellín, Colombia
jcgutierru@eafit.edu.co

March 14, 2021

3) Practice for final project defense presentation

3.1. Make a table for exercise 1, take 20 examples to do this table

Array Size	Time Complexity
2	0
502	0.03
1002	0.14
1502	0.35
2002	0.63
2502	0.93
3002	1.38
3502	1.84
4002	2.49
4502	3.20
5002	3.89
5502	4.74
6002	5.55
6502	6.68
7002	7.79
7502	8.90
8002	10.10
8502	11.40
9002	12.46
9502	13.99

Table 1: Insertion Sort. Time Complexity vs Array Size

Array Size	Time Complexity
1	0
7501	0.05
15001	0.10
22501	0.16
30001	0.31
37501	0.30
45001	0.36
52501	0.52
60001	0.46
67501	0.54
75001	0.63
82501	0.75
90001	0.75
97501	0.86
105001	0.92
112501	1.1
120001	1.17
127501	1.17
135001	1.29
142501	1.36

Table 2: Merge Sort. Time Complexity vs Array Size

3.2. Plot the times you took for each algorithm

For **Insertion Sort**, the worst case would be an array that is completely sorted but from higher to lower, so we programmed 20 different array sizes ordered in this way and stored them to see how the time behaved against them, so we obtained Figure 1.

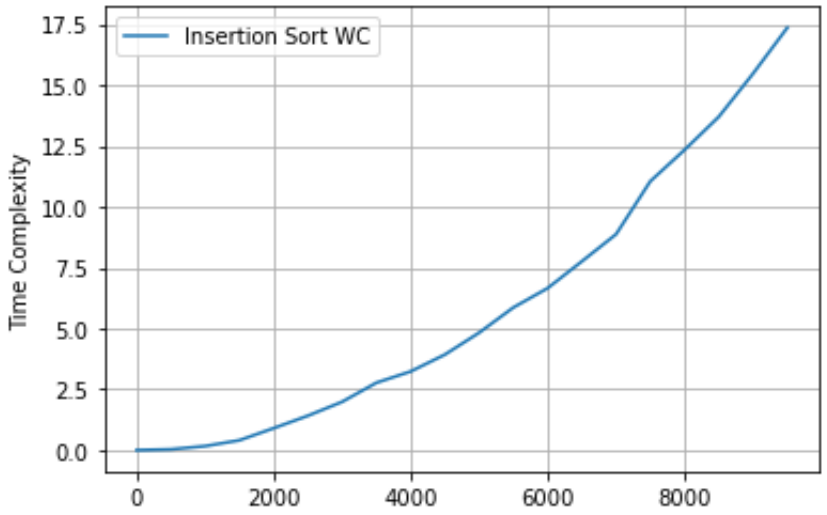

Figure 1: Array size vs Time (s) $O(n^2)$

For **Merge Sort**, the worst case would be an array organized in a way as if the algorithm was working backwards, so we programmed 20 different array sizes ordered in this way, using a code we found on internet(Wilde, M (2015) worstCaseArrayOfSize(n) (Version 1.0) [Source code]. https://stackoverflow.com/questions/24594112/when-will-the-worst-case-of-merge-sort-occur.) and stored them to see how the time behaved against them, so we obtained Figure 2.

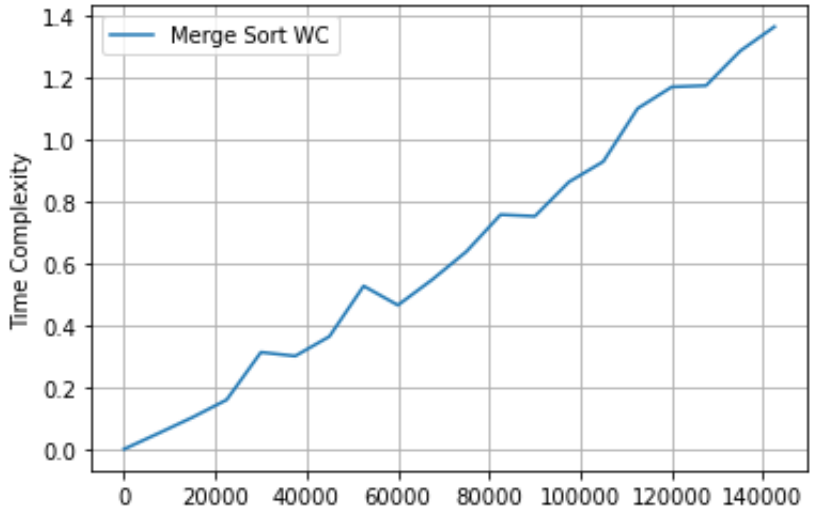

Figure 2: Array size vs Time (s) $O(n * log(n))$

Professor Mauricio Toro Bermudez
Phone: (+57)(4)2619500 Ext. 9473. Office: 19 – 627
E-mail: mtorobe@eafit.edu.co

3.3. ¿Is it appropriate to use Insertion Sort for a video game with millions of elements?

No. Because the algorithm complexity is $O(n^2)$ for the worst case scenario, so if we put millions of elements, then it would take too much time to process all the data, which is unacceptable on a video game.

3.4. Why is there a logarithm in the asymptotic complexity of Merge or Insertion sort

We find this behaviour in the **Merge Sort** algorithm, because at first we need to divide the initial array by two as much as we can, until we get groups of only one number and this is defined by the mathematical operator log_2 of n.

3.5.

3.6.

3.7. Calculate the complexity of the exercises of the numerals 2.1 and 2.2

For this point we will only show the worst case of each algorithm after having divided every line of the code in all the constants and functions of its complexity. This division will be in the attached python files on the "ejercicioEnLinea" GitHub folder.

3.7.1 Array 2

i. sum13

Worst case:

$$T(n) = c_5 * n$$

Big O notation:

$$T(n) = O(n)$$

ii. only14

Worst case:

$$T(n) = c_5 * n$$

Big O notation:

$$T(n) = O(n)$$

Professor Mauricio Toro Bermudez
Phone: (+57)(4)2619500 Ext. 9473. Office: 19 – 627
E-mail: mtorobe@eafit.edu.co

iii. SameEnds

Worst case:

$$T(n) = c_3 * n$$

Big O notation:

$$T(n) = O(n)$$

iv. fizzArray

Worst case:

$$T(n) = c_2 * n$$

Big O notation:

$$T(n) = O(n)$$

v. fizzArray3

Worst case:

$$T(n) = c_2 * n$$

Big O notation:

$$T(n) = O(n)$$

vi. has22

Worst case:

$$T(n) = c_2 * n$$

Big O notation:

$$T(n) = O(n)$$

3.7.2 Array 3

i. maxSpan

Worst case:

$$T(n) = c_6 * T(n^2)$$

Big O notation:

$$T(n) = O(n^2)$$

ii. fix34

Worst case:

$$T(n) = c_{11} * T(n^2)$$

Big O notation:

$$T(n) = O(n^2)$$

iii. fix45

Worst case:

$$T(n) = c_{11} * T(n^2)$$

Big O notation:

$$T(n) = O(n^2)$$

iv. squareUp

Worst case:

$$T(n) = c_7 * T(n^2)$$

Big O notation:

$$T(n) = O(n^2)$$

v. canBalance

Worst case:

$$T(n) = c_9 * T(n)$$

Big O notation:

$$T(n) = O(n)$$

3.8. Explain with your own words what the variables n and m from the last point meant

3.8.1 Array 2

i. sum13

n is the size of the array

ii. only14

n is the size of the array

iii. SameEnds

n is the number of initial and final position to be compared

iv. fizzArray

n is the size of the wanted array

v. fizzArray3

n is the size of the wanted array

vi. has22

n is the size of the array

3.8.2 Array 3

i. maxSpan

n is the size of the array

ii. fix34

n is the size of the array

iii. fix45

n is the size of the array

iv. squareUp

n is a natural number

v. canBalance

n is the size of the array

4) Practice for midterms

4.1.

$T(n) = c * n^2$

$T(100) = 1\text{ms}$

$c = \frac{T(n)}{(n^2)}$

$c = \frac{1[\text{ms}]}{10000[\text{num datos}^2]}$

$T(n) = \frac{1}{10000} * n^2$

$T(10000) = \frac{1[\text{ms}]}{10000[\text{num datos}^2]} * 10000^2[\text{num datos}^2]$

$T(10000) = 10000ms = 10s$

4.2. ***b***

4.3. ***a***

4.4.

4.4.1 $O(m * n)$

4.4.2 $O(m * n)$

Professor Mauricio Toro Bermudez
Phone: (+57)(4)2619500 Ext. 9473. Office: 19 – 627
E-mail: mtorobe@eafit.edu.co

4.5.

4.5.1 d

4.5.2 a

Professor Mauricio Toro Bermudez
Phone: (+57)(4)2619500 Ext. 9473. Office: 19 – 627
E-mail: mtorobe@eafit.edu.co