Laboratory practice No. 2: Big O Notation

Paulina Pérez Garcés
Universidad Eafit
Medellín, Colombia
pperezg@eafit.edu.co

Santiago Cartagena Agudelo
Universidad Eafit
Medellín, Colombia
scartagena@eafit.edu.co

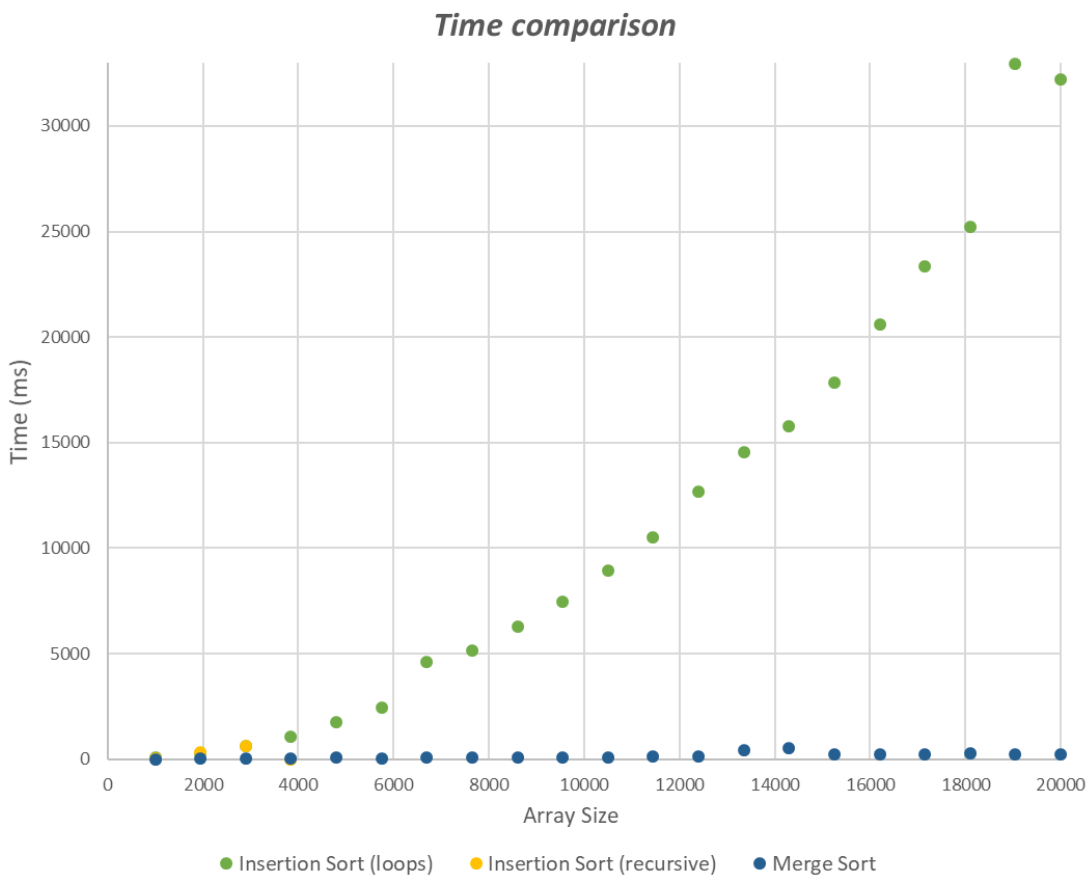
3) Practice for final project defense presentation

3.1 Table for exercise number 1:

Insertion Sort		Insertion Sort (Recursive)		Merge Sort	
Array size	Time (ms)	Array size	Time (ms)	Array size	Time (ms)
1000	83.10	1000	62.41	1000	10.53
1950	293.46	1950	340.75	1950	34.80
2900	649.24	2900	640.26	2900	40.23
3850	1090.22	3850	The recursive version of insertion sort doesn't run on our computer for arrays with more than 3932 elements.	3850	64.16
4800	1757.80	4800		4800	68.98
5750	2469.03	5750		5750	63.00
6700	4598.43	6700		6700	66.19
7650	5175.22	7650		7650	79.04
8600	6294.73	8600		8600	84.09
9550	7473.03	9550		9550	102.95
10500	8945.84	10500		10500	112.90
11450	10528.21	11450		11450	126.40
12400	12666.18	12400		12400	136.46
13350	14573.18	13350		13350	432.49
14300	15798.98	14300		14300	529.95
15250	17822.61	15250		15250	229.79
16200	20622.56	16200		16200	217.08
17150	23331.42	17150		17150	221.44
18100	25228.31	18100		18100	261.38
19050	32966.89	19050		19050	227.21
20000	32225.10	20000		20000	251.17

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

3.2 Chart for exercise number 1:



3.3 After the previous test, it is easily seen that merge sort method is more useful than insertion sort. The former doesn't work on bigger problems, and the only alternative is to use a different version with loops, which takes more time than the normal type.

3.4 The use of Insertion Sort in videogames with millions of elements wouldn't be optimal. For starters, the recursive algorithm won't work with bigger numbers; likewise, the algorithms that uses loops takes too much time when running with bigger arrays. A better solution for this would be to use a method such as Merge Sort.

3.5 Insertion sort is only a better algorithm when it runs with arrays that are already (or almost) sorted. The reason is that this process doesn't do any operation in these cases; on the other hand, merge sort always divides and groups the array, even when it is sorted.

3.6 MaxSpan's objective is to find how many elements are between two other elements which have the same value. The one in the left is going to be called as "the leftmost element " and the one in the right is going to be called "the rightmost element ", so we are going to called "Span" the number of elements between the leftmost element and the rightmost element, and we want to know which one is the largest Span of any given array.

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

For example, if the given array is an array with just one value, the span will be 1, another thing we must consider is that we need to include in our count also the rightmost and the leftmost element.
Now the algorithm works like this: we created a variable called “m” which is going to store the Span, then we make a nested loop, the outside loop is for the leftmost element and the inside loop is for the rightmost element, in both loops the condition is that the element has to be less than the array length, next we introduce a conditional if to compare the equality between the leftmost element and the rightmost element and also the rightmost element minus the leftmost element plus 1 have to be more than our variable m, in case that the previous conditional sentence be true, the variable m will be equal to the following operation : rightmost element – leftmost element + 1
This program will be continue doing the previous mentioned steps until the element in the leftmost position be more or equal to the length of the array and then the program will return the variable m, which will be the largest Span and this is the answer to the problem.

3.7 Complexity for online exercises:

EXERCISE	LEVEL	COMPLEXITY(T(N))	COMPLEXITY(BIG(O))
1. CountEvens	2	T(n) = n + C	O(n)
2. Sum13	2	T(r) = r + C	O(r)
3. Only14	2	T(n) = n	O(n)
4. is Everywhere	2	T(n) = n + C	O(n)
5. Fizz Buzz	2	T(k) = k + C	O(k)
1. maxSpan	3	T(m) = m2 + C	O(m2)
2. Fix34	3	T(t) = t2 + C	O(t2)
3. can Balance	3	T(n) = 2n + c	O(n)
4. squareUp	3	T(s) = s2 + C	O(s2)
5. maxMirror	3	T(l) = l2 + C	O(l2)

3.8

VARIABLE	DEFINITION
n	Array for which the algorithm will count the even numbers, or also Split it
r	Sum of the array.
m	The largest span between two numbers.
k	Difference between the end and the start of the array.
l	Length of the array.
s	Multiplication of the array parameter by itself.
t	Temporal variable which stored the position of the array plus one.

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

4) Practice for midterms

4.1 b
4.2 b
4.3 b
4.4 b
4.5 d
4.6 a
4.7
- **4.7.1** T(n) = T(n-1) + c
- **4.7.2** O(n) = n

4.8 a
4.9 d
4.11 c
4.12 b
4.13 a
4.14 a

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473