Laboratory practice No. 2: Algorithm’s complexity

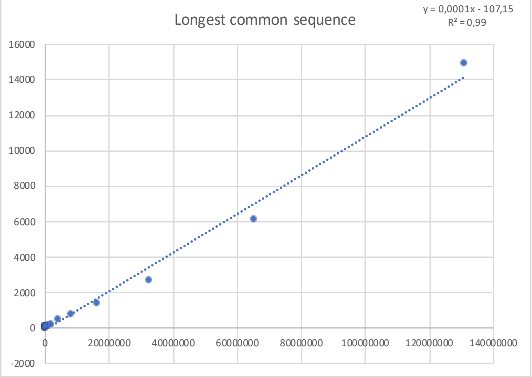
Vicente Aristizabal Ospina
Universidad Eafit
Medellín, Colombia
Varisti7@eafit.edu.co

3) Practice for final project defense presentation

3.1, 3.2:

Merge Sort:

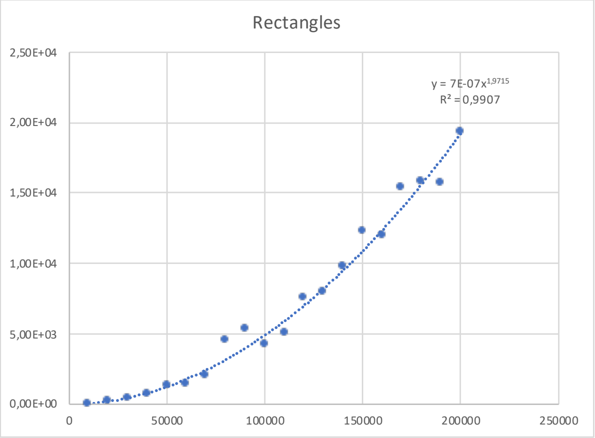
Size (n)	Time (s)
1000	1
2000	3
4000	3
8000	17
16000	10
32000	16
64000	33
128000	45
256000	58
512000	49
1024000	122
2048000	181
4096000	426
8192000	696
16384000	1317
32768000	2666
65536000	6115
131072000	14882



O(nlog(n))

Insertion Sort:

Size (n)	Time (s)
10000	6,30E+01
20000	2,00E+02
30000	3,97E+02
40000	6,75E+02
50000	1295
60000	1452
70000	2075
80000	4493
90000	5352
100000	4195
110000	5054
120000	7598
130000	7942
140000	9747
150000	12269
160000	11963
170000	15349
180000	15784
190000	15718
200000	19282



Common sequence by complexity	
Tamaño (n)	200000
Tamaño total	4,E+10
Segundos (Tt/2GHz)	20

O(n^2)

3.3:

No, because the complexity in the worst case is n^2 and a renderization in which n is for example one million, the complexity would be 1.000.000.000.000 and that divided by lets say 2.5 GHz would take more or less 400 seconds, something slow for videogames

3.4

Because the size of the arrays we are sorting increases by 2*lastSize, so the number of times that the algorithm enters the loop is log(n).

3.5

The following complexities are the ones that express the worst case for each algorithm.

// Array 2

// Array-2 > countEvens
O(n)
// Array-2 > bigDiff
O(n)
// Array-2 > sum13
O(n)
// Array-2 > sum67
O(n)
// Array-2 > has22
O(n)
// Array 3

// Array-3 > fix34
O(n^2)
// Array-3 > fix45
O(n^2)
// Array-3 > canBalance
O(n^2)
// Array-3 > squareUp
O(n)
// Array-3 > seriesUp
O(n^2)

3.6

N: the size of the array that receives the algorithm.

4) Practice for midterms

4.2. D.
4.5. D, (a) si.
4.6. 100.000 segundos
4.7. 1, 3 y 4.
4.9. A
4.14. A.