Laboratory No. 2
Big O notation and Complexity

David Restrepo Ramirez
EafitUniversity
Medellín, Colombia
emailinegrante1@eafit.edu.co

Juan Felipe Lopez Gutierrez
EafitUniversity
Medellín, Colombia
jflopezg@eafit.edu.coSimulation

3) of questions to support Projects

3.1 The first table si for the best cases (all items are ordered).

# de datos	MergeSort	InsertionSort
1000000	64	7
5000000	307	23
10000000	601	22
15000000	949	17
20000000	1147	19
25000000	1503	12
30000000	1731	16
35000000	2087	21
40000000	2597	19
45000000	2924	24
50000000	3231	29
55000000	3225	30
60000000	3947	39

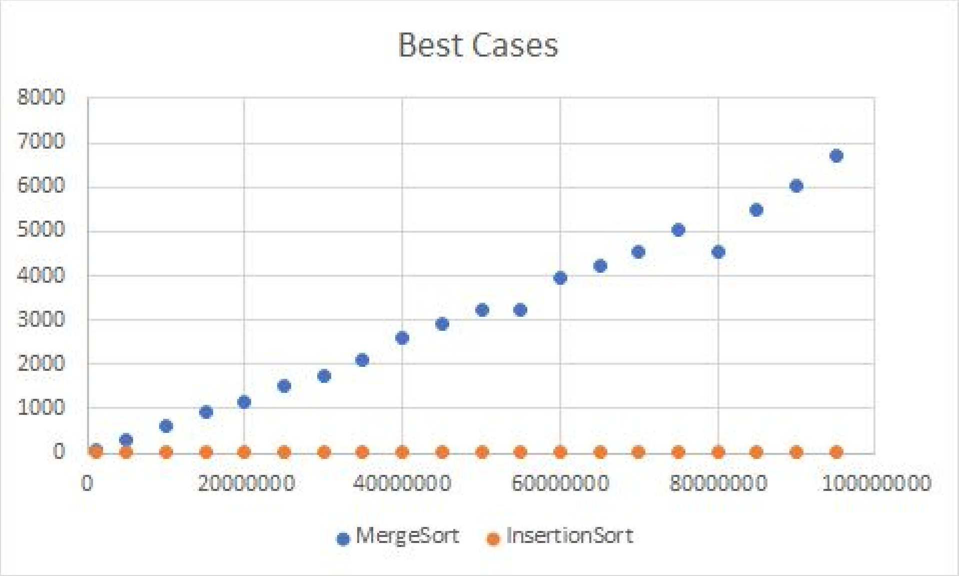
65000000	4205	29
70000000	4533	37
75000000	5028	24
80000000	4525	34
85000000	5496	28
90000000	6047	34
95000000	6717	34

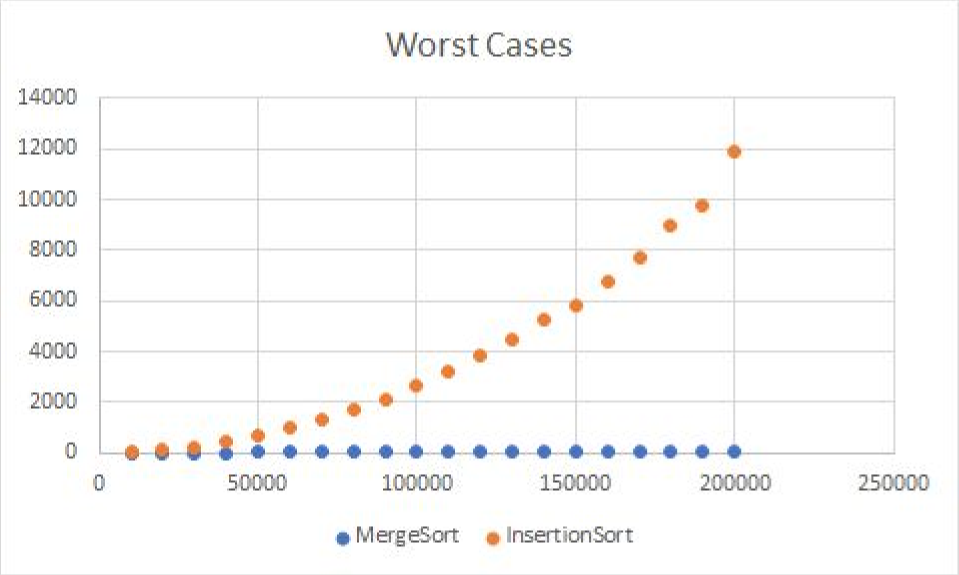
The second table is for the worst cases (all items are disordered)

# de datos	MergeSort	InsertionSort
10000	5	44
20000	3	115
30000	4	239
40000	5	429
50000	7	674
60000	6	1026
70000	8	1293
80000	8	1677
90000	11	2136
100000	12	2615
110000	13	3217
120000	16	3807
130000	27	4495
140000	30	5241

150000	30	5782
160000	30	6774
170000	38	7705
180000	21	9004
190000	15	9740
200000	17	11940

3.2 Like the previous point, the first chart is for the best cases, and the second chart is for the worst cases.





3.3 Normally these sorting algorithms have efficient response times, which is to say, naturally, insertion sort has a complexity between O (n) to O (n^ 2), the latter being the worst of cases and on the other hand the merge sort always oscillates in O (nlog (n)). It is then that the use of merge sort for its ordering is much better for us in the case of a very large array, since it will always stay the same, while insertion sort sacrifices.

3.4 The use of insertion sort in this not be the most ideal application where millions of data are implemented at the same time would, the best would be to opt for merge sort or another whose complexity gives us a better guarantee of resource use.

3.5 For arrays with many data, the use of insertion sort would be convenient if the data are ordered, or if the data are the same. Again, if the data are the same, or if the data are ordered, insertion sort will be faster than merge sort.

3.6 In this exercise you have to calculate the longest interval between the appearance of a number and its repetition.For this, you have to calculate the length of all these intervals, and then return the longest.

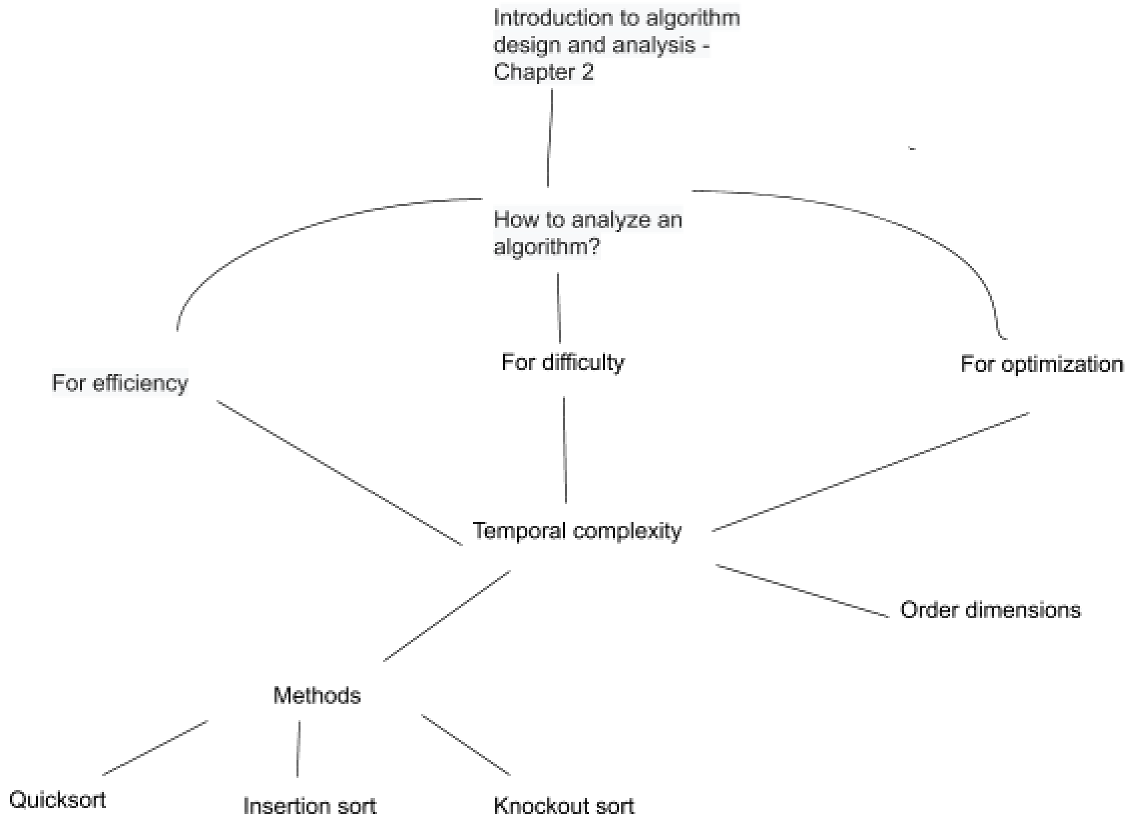
3.7 is made with david code
3.8 is made with david code

4) Partial simulation

4.1 *C*
4.2 *B*

4.3 *C*
4.4 *A*
4.5 *D , No*
4.6 *1s*
4.7 *all except D*
4.8 *A*
4.9 *D*
4.10 *D*
4.11 *C*
4.12 *B*
4.13 *C*
4.14 *A*

5) Recommended reading


Introduction to algorithm design and analysis - Chapter 2
How to analyze an algorithm?
For efficiency
For difficulty
For optimization
Temporal complexity
Order dimensions
Methods
Quicksort
Insertion sort
Knockout sort