ESTRUCTURA DE DATOS 1
Código ST0245

Laboratory practice No. 2: Big O Notation

Isabella Quintero Villegas
Universidad Eafit
Medellín, Colombia
iquinterov@eafit.edu.co

Sofia Vega Lopez
Universidad Eafit
Medellín, Colombia
svegal@eafit.edu.co

1) Code references

Insertion sort
Mishra. R(No date). Insertion Sort. https://www.geeksforgeeks.org/insertion-sort/
Merge Sort
Mishra. R(No date). Merge Sort. https://www.geeksforgeeks.org/merge-sort/

3) *Practice for final project defense presentation*

3.1

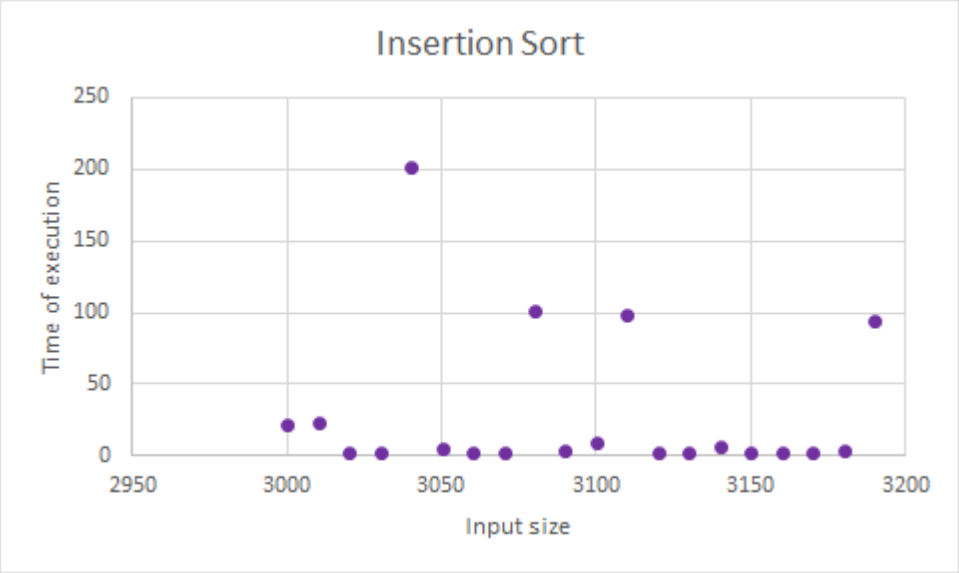
Insertion Sort	
3000	22
3010	23
3020	3
3030	3
3040	202
3050	5
3060	2
3070	2
3080	101
3090	4
3100	9
3110	98
3120	3
3130	3
3140	6
3150	2
3160	3
3170	2
3180	4

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

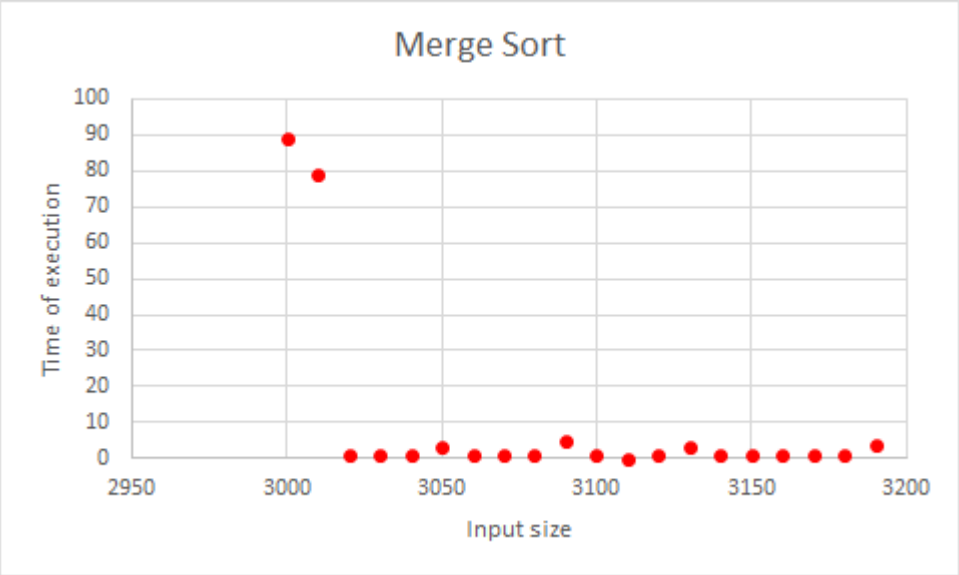
3190	94

Merge Sort	
3000	89
3010	79
3020	1
3030	1
3040	1
3050	3
3060	1
3070	1
3080	1
3090	5
3100	1
3110	0
3120	1
3130	3
3140	1
3150	1
3160	1
3170	1
3180	1
3190	4

3.2



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



3.3 Based on the results that we had in the points 3.1 and 3.2, we finally reach to the conclusion that merge sort is more efficient than insertion sort, it can be appreciate in the graphics that sometimes the insertion sort algorithm took much longer time than the other algorithm, also at the end of merge sort it had a tendency line without extreme values in the time of execution.

3.4 No, use insertion sort in a game with millions of elements could be counterproductive, because in the middle of the game it can take a lot of time to run the executions and it is not uniform, this means that the game sometimes can be extremely faster or extremely low.

3.5 Insertion Sort algorithm could be more effectively if in the given array it does not have to do many alterations, this means that this array is almost organized at the beginning, so it would be faster than the merge sort algorithm.

3.6 maxSpan algorithm takes an array of numbers and considers the rightmost and leftmost appearance of a value in it, then it counts the number of digits between this positions, including them, which we will call a span, finally it returns the maximum span in the array; it works in the following way: first, we stablish two variables, one is for calculating the spans in the array (span), and the other is for saving the maximum value found(maxSpan), both of them initialized at cero; then we use two loops, one inside the other, the first one will go through the array from beginning (position i), the second one also goes through the array, but from the end (position j), when the number in position I equals to the number in position j, we will calculate the span as j - i + 1, next, we evaluate if span is greater than maxSpan, if it is, maxSpan, will be span, at the end, the method returns the largest span found, after finalizing the loops. HOLA JOFI

3.7

Exercise	***Complexity (T(n))***	***Asymptotic complexity (O(n))***
countEvens	T(n) = n + C	O(n)

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

centeredAverage	T(n) = n + C	O(n)
Lucky13	T(n) = n + C	O(n)
matchUp	T(n) = n + C	O(n)
fizzArray2	T(n) = n + C	O(n)
Exercise	**Complexity (T(n))**	**Asymptotic complexity (O(n))**
maxSpan	T(n) = n^2 + C	O(n^2)
fix34	T(n) = n^2 + C	O(n^2)
fix45	T(n) = n^2 + C	O(n^2)
seriesUp	T(n) = n + C	O(n)
squareUp	T(n) = m + m(m+1)/2 + C	O(m^2)

3.8

Variable	***Definition***
n	Length of the array used in each exercise (parameter)
m	Length of the array(parameter^2)

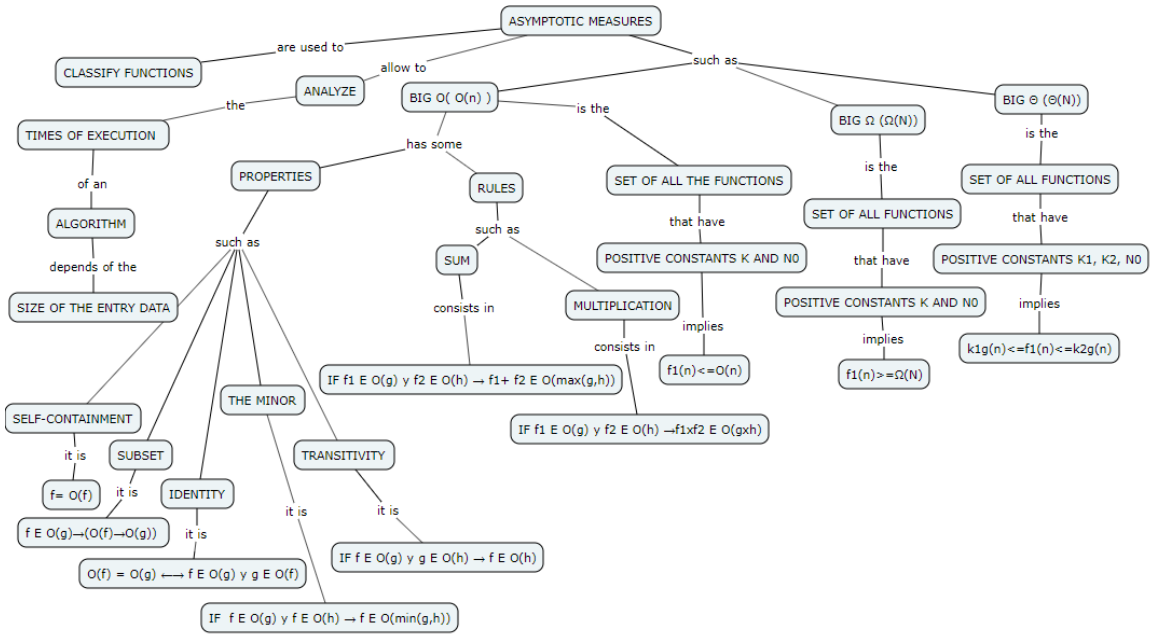
4) Practice for midterms

4.1 *c*
4.2 d
4.3 b
4.4 b
4.5 d
4.6 a
4.7 T(n) = T(n-1) + C / O(n)
4.8 a
4.9 d
4.10 c
4.11 c
4.12 b
4.13 c
4.14 a or c

5) Recommended reading (optional)

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

ESTRUCTURA DE DATOS 1
Código ST0245



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