ESTRUCTURA DE DATOS 1
Código ST0245

Laboratory practice No. 2: Algorithm complexity laboratory.

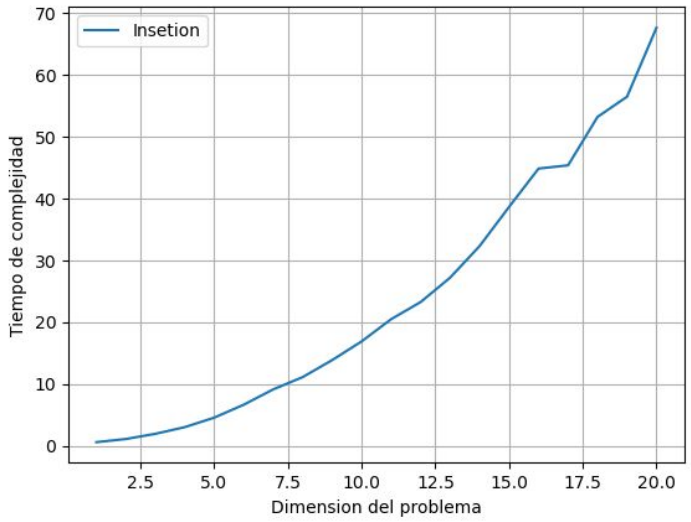
Maria Isabel Arango Palacio
Universidad Eafit
Medellín, Colombia
miarangop@eafit.edu.co

Isabella Montoya Henao
Universidad Eafit
Medellín, Colombia
imontoyah@eafit.edu.co

3) Practice for final project defense presentation

3.2 Graphics

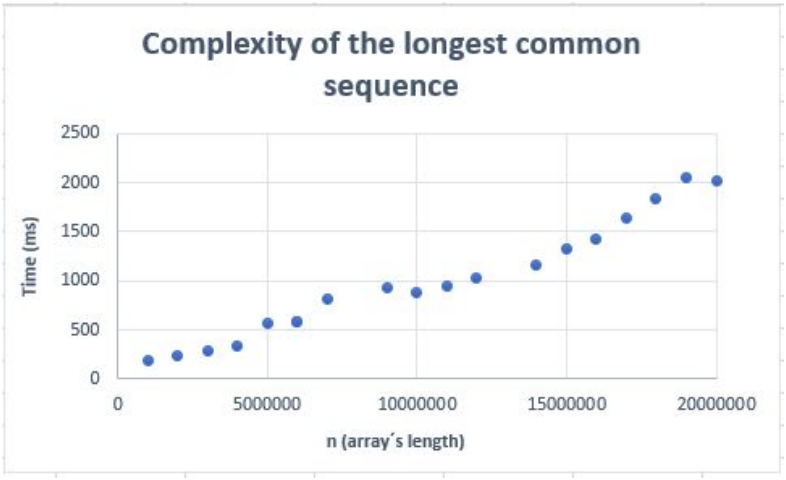
Insertion Sort



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

Merge Sort

n	T(n)
1000000	186
2000000	240
3000000	287
4000000	342
5000000	572
6000000	583
6000000	583
7000000	811
9000000	932
10000000	877
11000000	951
12000000	1031
14000000	1165
15000000	1333
16000000	1418
17000000	1636
18000000	1836
19000000	2056
20000000	2015



3.3 It is not really appropriate to use the insertion sort algorithm in those cases because it has a complexity of $O(n^2)$ and it means that the number of operations that it will do per time(ms) will be two times the number of elements that it has, so it will take a lot of time with millions of elements. In consequence, insertion sort is not very efficient to do a real time rendering because it could be so slow making the video game scenes and that would not benefit the video game.

3.4 The algorithm merge sort has a complexity in time for best, average and worst cases of O(n*log(n)), this is because it's a divide algorithm and the input is repeatedly halved. You divide the array in half log(n) times, and each time you do, for each segment, you have to do a merge on two sorted arrays. Merging two sorted arrays is O(n). The algorithm is just to go through the two arrays, and walk up the one that's lagging.

3.5 In the insertion sort, if the elements of the array are sorted or are equal, the complexity of this algorithm will be equal to the complexity of MergeSort.

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

3.6 CodingBat

Array-2

1. FizzArray

In this exercise the goal is to create a new array that will contain the numbers from 0 to n-1 consecutively. We will have an integer n that will determine the length of the array. So in the method fizzArray we will create a loop that is going to depend on n and it will fill the array with the respective numbers.

$C_1 = 1$
$C_2 = 1$
$C_3 = 3$
$C_4 = 2$
$C_5 = 1$

Worst Case:

$T(n) = C_1 + C_2 * n + C_3 + C_4 * n + C_5$
$T(n) = C_2 * n + C_4 * n \rightarrow Sum\ law$
$T(n) = n \rightarrow Product\ law$
$O(n)$ *where n is the array's length*

2. more14

Here we need to go through an array because the objective is to determine if there are more numbers with the value one than numbers with the value four. It is a good idea to have two variables in which you can store the numbers of ones and fours respectively, because at the end we can compare both and determine the answer. And you can give values to those variables asking if in certain positions of the array the number is one or four and so on.

$C_1 = 2$
$C_2 = 3$
$C_3 = 3$
$C_4 = 4$
$C_5 = 4$
$C_6 = 3$
$C_7 = 1$

Worst Case:

$T(n) = C_1 + C_2 + C_3 * n + C_4 * n + C_5 * n + C_6 + C_7$
$T(n) = C_3 * n + C_4 * n + C_5 * n \rightarrow Sum\ law$

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

$T(n) = n \rightarrow Product\ law$
$O(n)$ *where n is the array's length*

3. ZeroFront

This algorithm receives an array of int and we must return an array that contains the exact same numbers as the given array, but rearranged so that all the zeros are grouped at the start of the array. To reach our goal, we compute a for where if the number in the array is equal to 0, we will keep it in a temporal variable and change the respective positions until we get all the 0`s in the start of the array.

$C_1 = 1$
$C_2 = 1$
$C_3 = 3$
$C_4 = 9$
$C_5 = 1$

Worst Case:

$T(n) = C_1 + C_2 + C_3 * n + C_4 * n + C_5$
$T(n) = C_3 * n + C_4 * n \rightarrow Sum\ law$
$T(n) = n \rightarrow Product\ law$
$O(n)$ *where n is the array's length*

4. CenteredAverage

In this algorithm we must return the "centered" average of an array of ints, which we'll say is the mean average of the values, except ignoring the largest and smallest values in the array. If there are multiple copies of the smallest value, ignore just one copy, and likewise for the largest value. So, we implement a for to store the maximum and minimum element in the array. Also, we implement another loop to go through and array and sum all the elements, and finally to get the average we subtract the maximum and the minimum to the total sum and divide it by the array's length minus 2 (because we won't take into account the max and the min in the array).

$C_1 = 2$
$C_2 = 2$
$C_3 = 1$
$C_4 = 1$
$C_5 = 3$
$C_6 = 3$
$C_7 = 3$
$C_8 = 1$
$C_9 = 3$
$C_{10} = 3$
$C_{11} = 3$
$C_{12} = 4$
$C_{13} = 1$

Worst Case:

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

$$T(n) = C_1 + C_2 + C_3 + C_4 + C_5 * n + C_6 * n + C_7 * n + C_8 + C_9 * n + C_{10} * n$$
$$T(n) = C_5 * n + C_6 * n + C_7 * n + C_9 * n + C_{10} * n \quad \rightarrow Sum\ law$$
$$T(n) = n \quad \rightarrow Product\ law$$
$$O(n)\ where\ n\ is\ the\ array's\ length$$

5. EvenOdd

This algorithm receives an array of int and we must return an array that contains the exact same numbers as the given array, but rearranged so that all the even numbers come before all the odd numbers. So, we compute a for that go through the array looking for the ood numbers, where with an if condition we evaluate if the module 2 of the number in the array is equal to 0, it will be an odd number, so it will be stored in a temporal variable and change the respective positions until we get all the odd`s numbers.

$C_1 = 2$ $\quad$ $C_6 = 1$
$C_2 = 2$ $\quad$ $C_7 = 1$
$C_3 = 13$ $\quad$ $C_8 = 3$
$C_4 = 1$ $\quad$ $C_9 = 8$
$C_5 = 3$ $\quad$ $C_{10} = 8$

Worst Case:

$$T(n) = C_1 + C_2 + C_3 + C_4 + C_5 * (n-1) + C_6 * (n-1) + C_7 + C_8 * n + C_9 * n + C_{10}$$
$$T(n) = C_1 + C_2 + C_3 + C_4 + (n-1) + (n-1) + C_7 + n + n + C_{10} \quad \rightarrow Product\ law$$
$$T(n) = n \quad \rightarrow Sum\ law$$
$$O(n)\ where\ n\ is\ the\ array's\ length$$

Array-3

1. MaxSpan

Here we need to do a method that returns the number of elements that are in a certain range. The range is determined by the leftmost and rightmost equal elements. So for that we need to evaluate if the array in the leftmost position is equal to one in the rightmost and return the number of elements between them inclusive (it will be the length of the array). But if they are not equal you won't return the length of the array, otherwise you will need a temp variable and another variable to store the maximum. After that, you must implement two loops where in the first loop you will be evaluating the value in each position of the array, and in the second one you will compare the stored value in the first loop with the other positions. This process will be repeated iteratively, until we reach the array's length. Finally, it returns the maximum that has been stored.

$C_1 = 4$
$C_2 = 8$

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

$C_3 = 3$
$C_4 = 3$
$C_5 = 3$
$C_6 = 3$
$C_7 = 3$

Worst Case:

$T(n) = C1 + C2 + C4 + C3 * n + (C5 + C6 * n) * n + C7 * n^2 + C8 * n^2 + C9 * n^2 + C10$
$T(n) = (C3 + C5) * n + (C6 + C7 + C8 + C9) * n^2$ → *Sum law and common factor*
$T(n) = n + n^2$ → *Product law*
$T(n) = n^2$ → *Sum law*
$O(n^2)$ *where n is the array's length*

2. Fix34

The idea of the method that we will do in this exercise is to return an array with the same number of elements as the given array but rearrange with some parameters. The idea is to rearrange the array in such a way that the 3's is immediately followed by 4's, but you can not move the 3's. So in the method we do two loops, one is to find the 3´s and the other to find the 4´s, with the outermost loop we search the 4´s and with the innermost the 3´s. Then we do a kind of sort with the necessary elements to return the array in the way that we must do.

$C_1 = 1$
$C_2 = 1$
$C_3 = 3$
$C_4 = 1$
$C_5 = 3$
$C_6 = 13$
$C_7 = 1$

Worst Case:

$T(n) = C_1 + (C_2 + C_3 * n) + [(C_4 + C_5 * n) * n] + C_6 * n * n + C_7$
$T(n) = C_3 * n + C_5 * n * n + C_6 * n * n$ → *Sum law and common factor*
$T(n) = n + n * n$ → *Product law*
$T(n) = n * n$ → *Sum law*
$O(n^2)$ *where n is the array's length*

3. Fix45

The idea of the method that we will do in this exercise is very similar to the previous one. We were asked to return an array with the same number of elements as the given array but rearrange with some parameters. The idea is to rearrange the array in a way that the 4's is immediately followed by 5's, but you can not move the 4's. So in the method we do two loops, one is to find the 4´s and the other to find the 5´s, with the outermost loop we

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

search the 5´s and with the innermost the 4´s. Then we do a kind of sort with the necessary elements to return the array in the way that we must do.

$C_1 = 1$ $\quad$ $C_5 = 3$
$C_2 = 1$ $\quad$ $C_6 = 13$
$C_3 = 3$ $\quad$ $C_7 = 1$
$C_4 = 1$

Worst Case:

$T(n) = C_1 + (C_2 + C_3 * n) + [(C_4 + C_5 * n) * n] + C_6 * n * n + C_7$
$T(n) = C_3 * n + C_5 * n * n + C_6 * n * n \rightarrow$ *Sum law and common factor*
$T(n) = n + n * n \rightarrow$ *Product law*
$T(n) = n * n \rightarrow$ *Sum law*
$O(n^2)$ *where n is the array's length*

4. CanBalance

This algorithm receives a non-empty array,and we have to return true if there is a place to split the array so that the sum of the numbers on one side is equal to the sum of the numbers on the other side. For that, we declared two variables to store the sums, and one loop to store in the sum1 the sum of the first elements not including the last one and the sum2 to store the last element. Since here, the algorithm starts evaluating if both sum1 and sum2 are the same, if they don't, it starts subtracting the previous element of the first loop to the sum1 and adding it to sum2, if sum1 and sum2 reach the same value before the iterator >0, it will return "true", other way it returns "false".

$C_1 = 1$ $\quad$ $C_7 = 4$
$C_2 = 1$ $\quad$ $C_8 = 3$
$C_3 = 3$ $\quad$ $C_9 = 3$
$C_4 = 1$ $\quad$ $C_{10} = 3$
$C_5 = 3$ $\quad$ $C_{11} = 3$
$C_6 = 4$ $\quad$ $C_{12} = 2$

Worst Case:

$T(n) = C_1 + C_2 + C_4 + C_3 * (n-1) + C_5 * n + C_6 * n + [C_7 + C_8 * (n-3)] * n + C_9 * n^2 + C_{10} * n^2 + C_{11} * n^2 + C_{12}$
$T(n) = C_3 * (n-1) + (C_5 + C_6) * n + (C_8 * (n-3)) * n + (C_9 + C_{10} + C_{11}) * n^2 \rightarrow$ *Sum law and common factor*
$T(n) = (n-1) + n + (n-3) * n + n^2 \rightarrow$ *Product law*
$T(n) = n + n^2 \rightarrow$ *Sum law and common factor*
$T(n) = n^2 \rightarrow$ *Sum law*
$O(n^2)$ *where n is the array's length*

5. LinearIn

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

This problem receives two arrays of ints sorted in increasing order, outer and inner, and asks us to return true if all of the numbers in inner appear in outer. We implemented two loops that go through each array, and a condition that evaluates if the number in the inner array is equal to the number in the outer array, it will increase a counter variable. Finally, we evaluated another condition, where if the counter variable has the same length of the inner array it returns true, otherwise it returns false.

$C_1 = 1$ $C_6 = 4$
$C_2 = 1$ $C_7 = 1$
$C_3 = 3$ $C_8 = 1$
$C_4 = 1$ $C_9 = 4$
$C_5 = 3$ $C_{10} = 1$

Worst Case:

$$T(n,m) = C_1 + C_2 + C_3 * n + (C_4 + C_5 * n) * m + C_6 * n * m + C_7 * n * m + C_8 * n * m + C_9 + C_{10}$$
$$T(n,m) = C_3 * n + C_5 * n * m + C_6 * n * m + C_7 * n * m + C_8 * n * m \rightarrow Common\ factor\ and\ sum\ law$$
$$T(n,m) = n + n * m \rightarrow Product\ law$$
$$T(n,m) = n * m \rightarrow Sum\ law$$
$$O(n * m)\ \ Where\ n\ is\ the\ inner`s\ length\ and\ m\ is\ the\ outer`s\ length$$

6. CountClump

The function here will return the number of series of 2 or more consecutively elements that are in the given array. We have a counter which we are going to increase everytime that we find a series, also we have a boolean that will help to evaluate some conditions like if the array in one condition is equal to the array in the position plus one. We will do a loop to go through the array and inside it we will evaluate the condition previously named, if it is true the counter increases and continue evaluating other conditions.

$C_1 = 2$ $C_5 = 8$
$C_2 = 4$ $C_6 = 5$
$C_3 = 1$ $C_7 = 1$
$C_4 = 3$

Worst Case:

$$T(n) = C_1 + C_2 + (C_3 + C_4) * (n-1) + C_5 * (n-1) + C_6 * (n-1) + C_7$$
$$T(n) = C_4 * (n-1) + C_5 * (n-1) + C_6 * (n-1) \rightarrow Sum\ law\ and\ common\ factor$$
$$T(n) = (n-1) \rightarrow Product\ law$$
$$T(n) = n \rightarrow Sum\ law$$
$$O(n)\ where\ n\ is\ the\ array's\ length$$

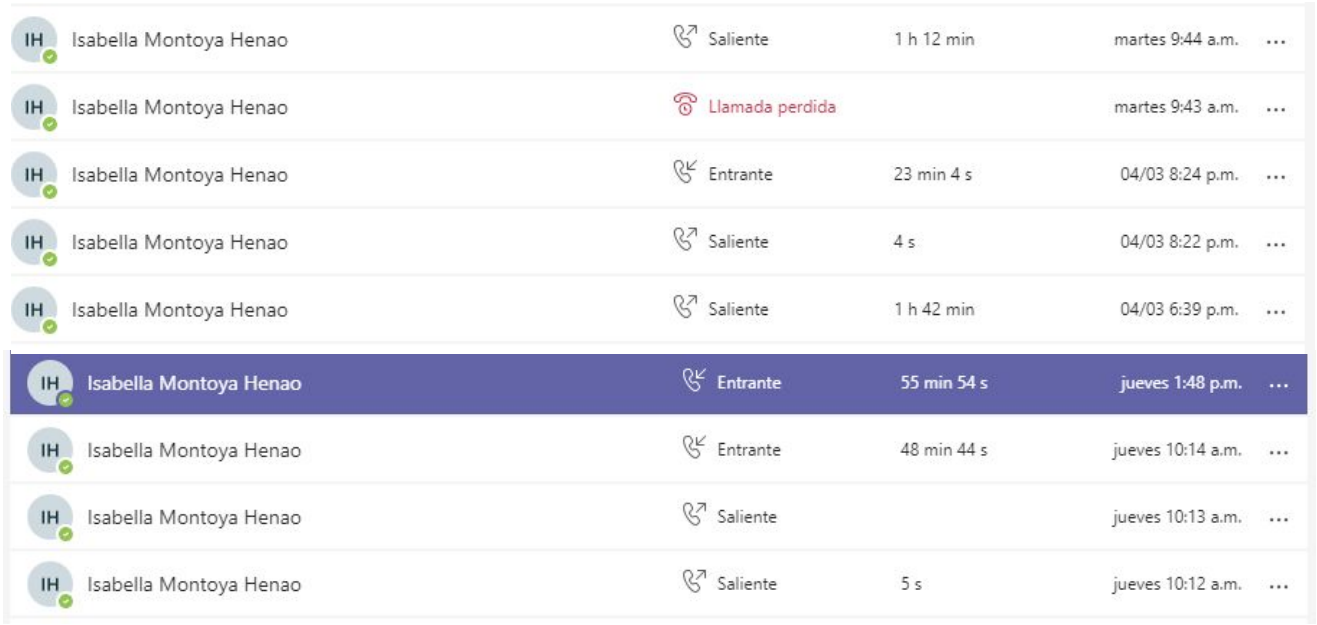
PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

4) Practice for midterms

4.1 10 seconds
4.2 *D*
4.3 *A*
4.4 *1. The time complexity in the worst case will be O(n*m)*
*2. The memory complexity in the worst case will be O(n*m)*
4.5 *4.5.1 D*
4.5.2 A. It always finishes when $n \varepsilon Z$.
4.6 *A*
4.7 *All of them are true.*
4.8 *A*
4.9 *C*
4.10 *C*
4.11 *C*
4.12 *A*

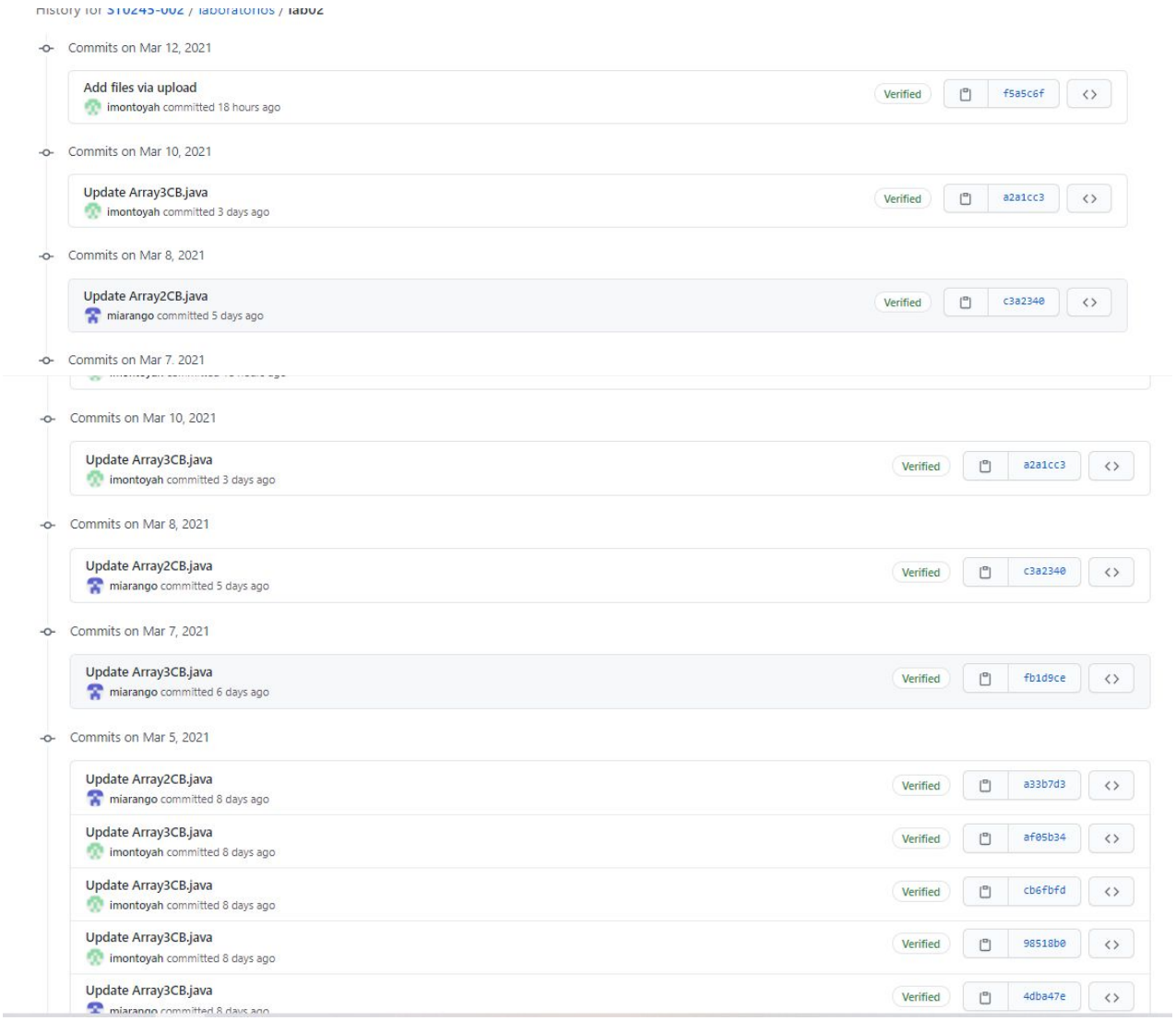
6) Teamwork and gradual progress (optional)

6.1 Meeting minutes

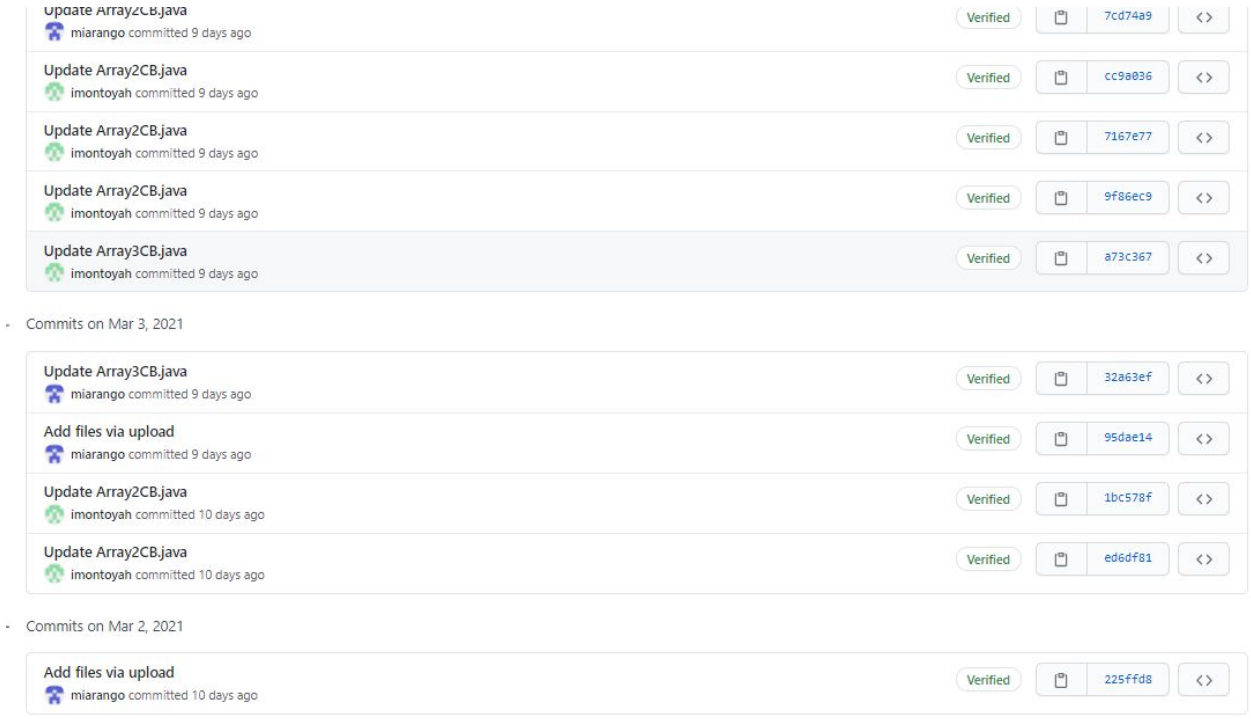


PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

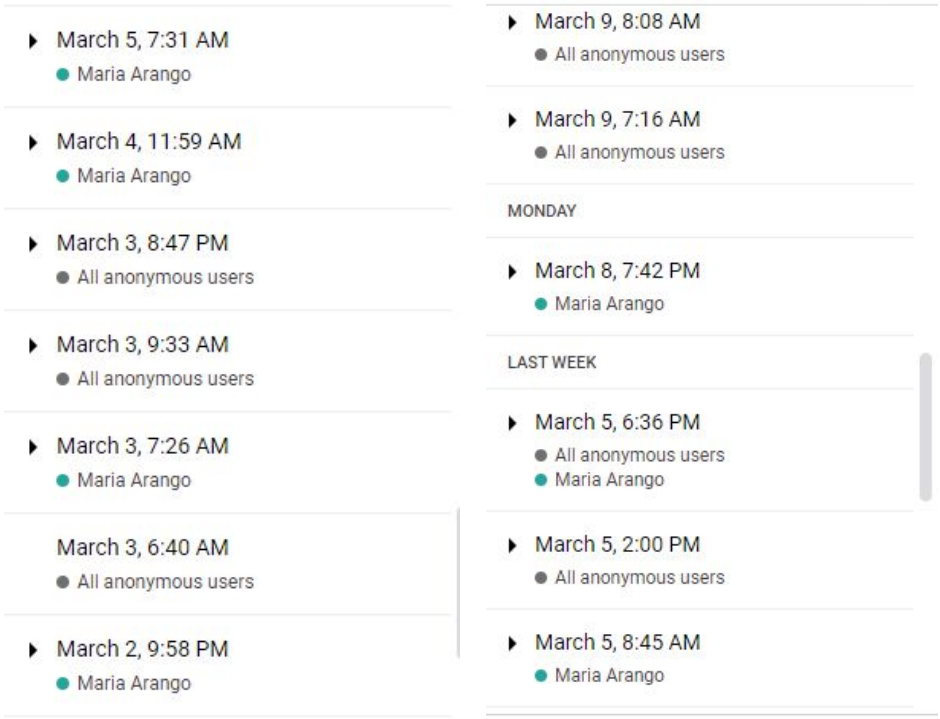
6.2 History of changes of the code



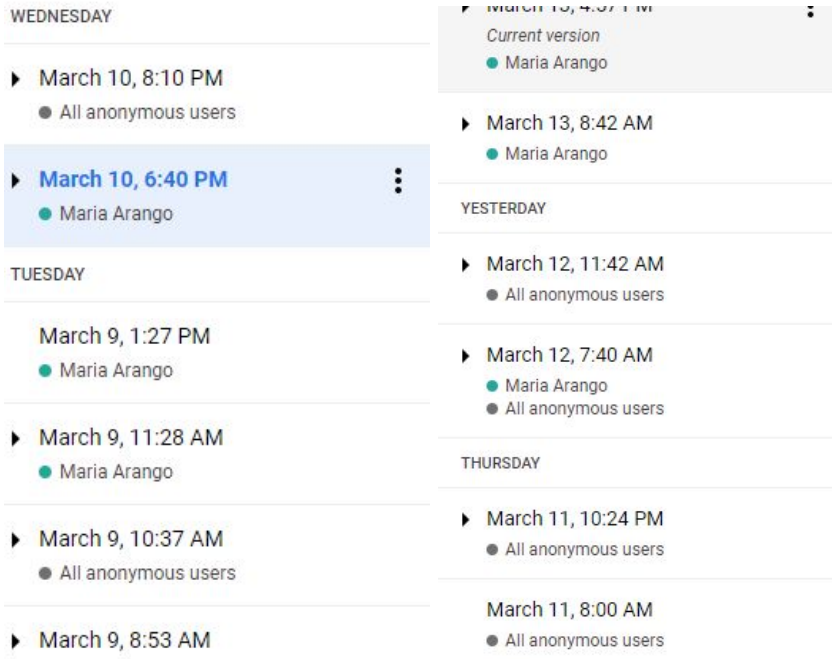
PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473



6.3 History of changes of the report



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



References

[ST0245-Eafit/ED1-Laboratorio2 Vr 16.0.pdf at master · mauriciotoro/ST0245-Eafit (github.com)](#)

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