Laboratory practice No. 2: Big O Notation

Miguel Ángel Correa Manrique
Universidad Eafit
Medellín, Colombia
macorream@eafit.edu.co

Pablo Buitrago Jaramillo
Universidad Eafit
Medellín, Colombia
pbuitragoj@eafit.edu.co

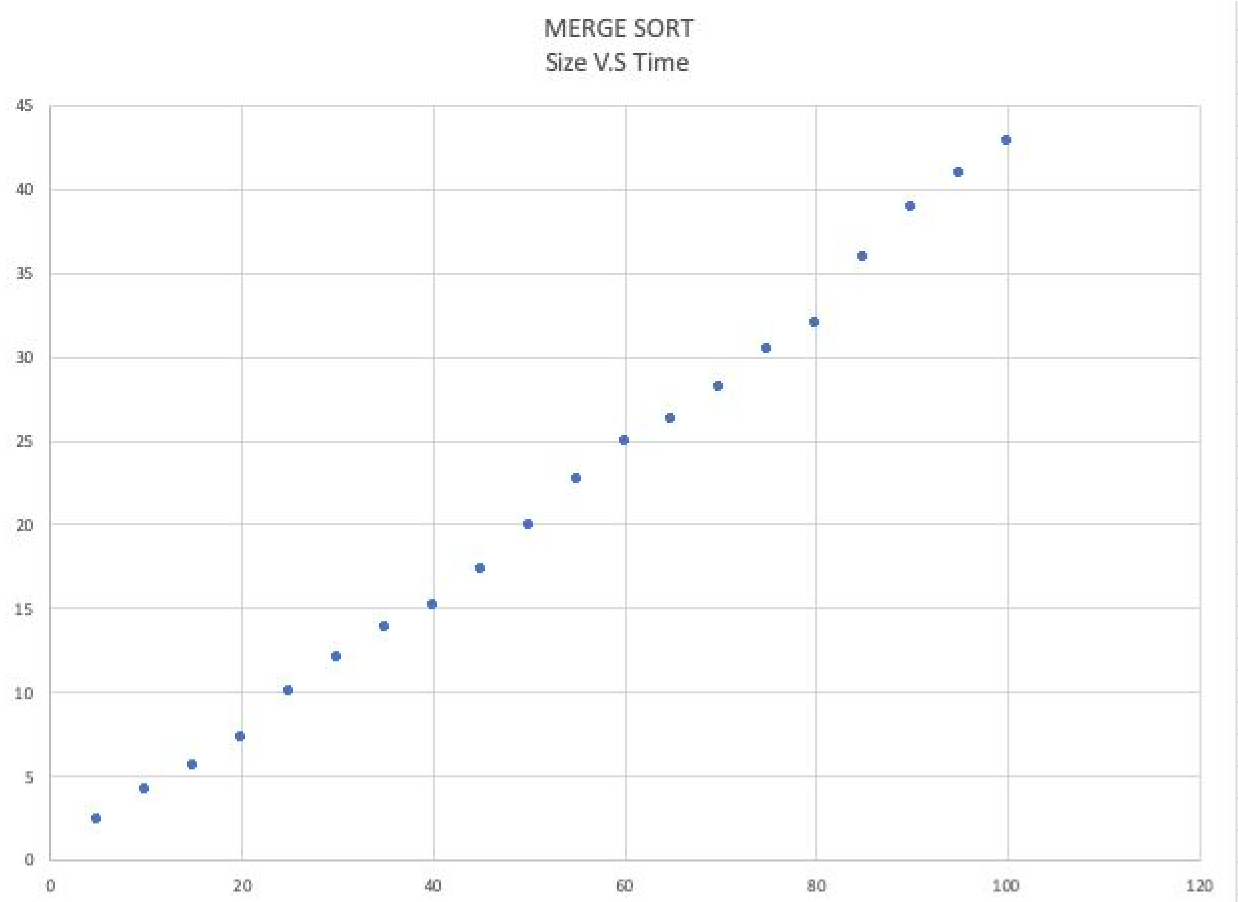
3) Practice for final project defense presentation

3.1

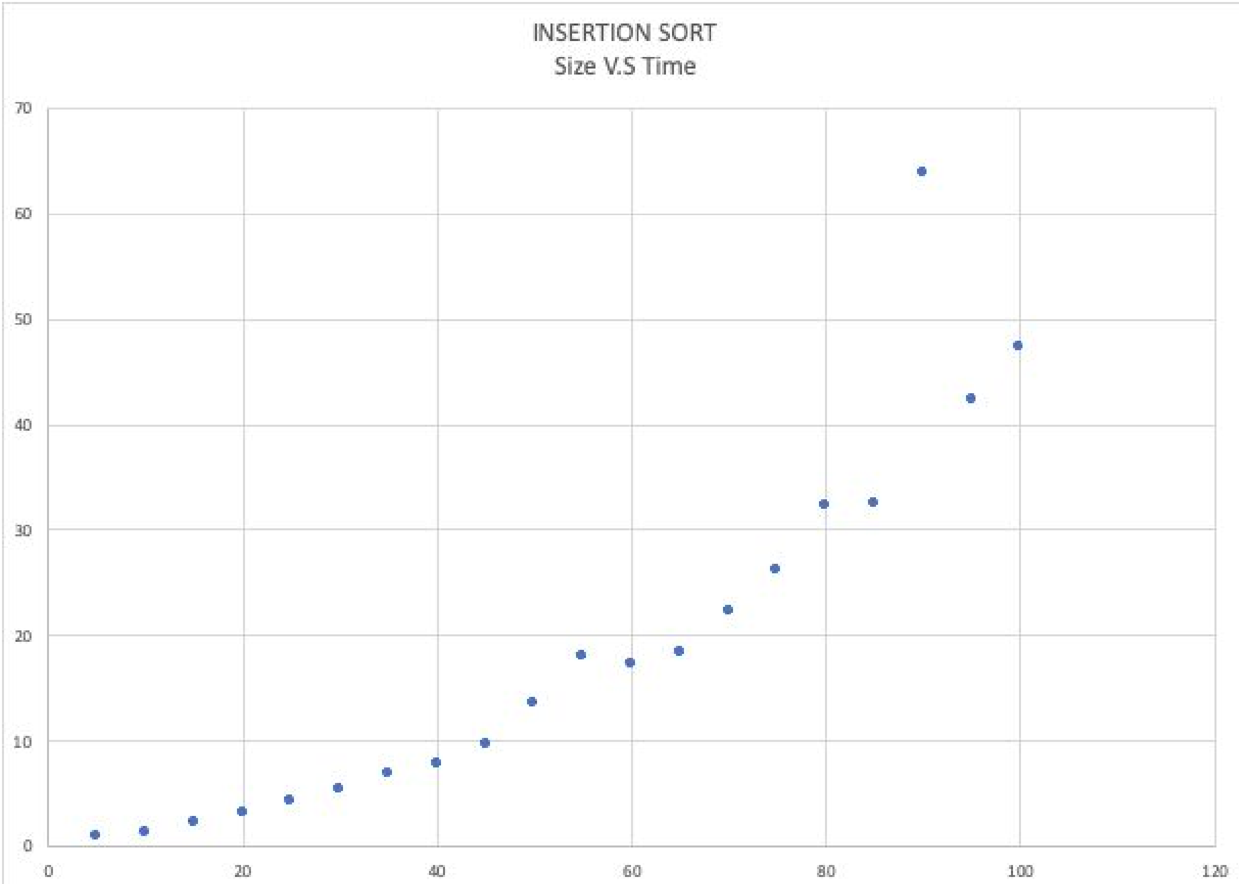
MERGE SORT	
Size	Time
5	2.444
10	4.149
15	5.625
20	7.323
25	10.104
30	12.027
35	13.827
40	15.188
45	17.289
50	19.96
55	22.663
60	24.934
65	26.312
70	28.218
75	30.399
80	31.968
85	35.882
90	38.939
95	40.968
100	42.833

INSERTION ORT	
Size	Time
5	0.892
10	1.268
15	2.219
20	3.27
25	4.394
30	5.464
35	6.953
40	7.817
45	9.746
50	13.618
55	17.979
60	17.242
65	18.458
70	22.226
75	26.13
80	32.26
85	32.555
90	63.841
95	42.28
100	47.318

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 Merge Sort algorithm is more efficient than insertion sort in worst general terms, because its time complexity is O(nlogn) whether insertion sort is O(n²), but this changes when it's all about space complexity, since merge sort uses recursive algorithms to perform, it's going to use a lot of memory for itself while increases, in contrast to insertion sort algorithm where its space complexity is constant.

In general merge sort is better, but it requires more memory usage than insertion sort, for little array size or elements almost already sorted insertion sorts is better.

3.4 No, since insertion sort has a time complexity of O(n²), its processing is going to be slower by every time the array increases on quadratic time complexity.

3.5 For big arrays, it would be the case that if the array is given almost already arrange could perform faster than merge sort, considering a good case scenario of course, because merge sort will continue performing no matter what, take a look to the code and the condition arr[j-1] > arr[j] presented at the second for loop.

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

```
void insertionSort(int arr[], int n){
        for(int i = 0; i < n; i++){

                for(int j = i; j > 0 && arr[j-1] > arr[j]; j--){

                        int temp = arr[j];

                        arr[j] = arr[j-1];

                        arr[j-1] = temp;


                }

        }

}
```

Also, as we mentioned merge sort will use more memory space than insertion sort.

3.6 CodingBat. Array3. MaxSpan.

Consider the leftmost and righmost appearances of some value in an array. We'll say that the "span" is the number of elements between the two inclusive. A single value has a span of 1. Returns the largest span found in the given array.

```
public int maxSpan(int[] nums) {
        int span = 0;
        int aux = 0;
         for(int i=0; i<nums.length; i++){
                for(int j=0; j<nums.length; j++){
                        if(nums[j] == nums[i]) aux = (j-i)+1;
                }
                 span = Math.max(span, aux);
        }
        return span;
}
```

So, to understand how this algorithm works lets begin with the two ints that are created: “span” is the max span that we will return at the end of the process, it will have to be actualized multiple times; “aux” is an auxiliary int that we will compare with “span” for its later update; then we have the first *for* loop that will go through all the positions in *nums* (the integer array) using an *i* int, this is

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

because we need to find and compare the biggest *span* for each of the different values that the array could have, so this first loop will allow us to update “span” taking into account every possible value; then we have the second *for* loop that will go through all the positions in the array using a *j* int, at this point, every time that the value in the array located in the *i* index is equal to the value in the array located in the *j* index means that we found a *span* so we need to store the span`s size, for that we will say that “aux” is equal to the difference between *j* and *i* plus one -we are assuming that each of this iterators represents an index, so this calculation will give us the *span* size, but in consequence that the first index is ‘0’ and not ‘1’ we need to add one to the size to get the real *span* size-; every time the second loop ends its iterations we will update “span” using the *Math.max()* function between “span” and “aux”, as “aux” is the max *span* for the value in the *i* index we need to make this update for each iterator that the first loop makes so we can get the maximum of all the *span* that the array contains. Lastly we will return “span”, at this point it has been updated all the possible times and represents the max *span* in the array.

3.7
Array2:

A. isEverywhere

```
public boolean isEverywhere(int[] nums, int val) {

        if(nums.length == 0 || nums.length == 1) return true; // C

        if(nums[0] == val){  //C

                for(int i = 0; i < nums.length; i = i + 2){ //n/2 + 1 -> O(n)

                if(nums[i] != val) return false; //n/2 * C

                }

                return true; // C

        } else if(nums[1] == val){ //C

                for(int i = 1; i < nums.length; i = i + 2){//n/2 -> O(n)

                if(nums[i] != val) return false; //n/2 * C

                }

                 return true; //C

        }
```

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

```
        return false; //C

    }
```

- The time complexity of this algorithm is O(n) because of the rule of the fastest growing term, where n/2 + 1 steps represents a time complexity of O(n).

B. zeroFront

```
  public int[] zeroFront(int[] nums) {

          int[] arr = new int[nums.length]; // C -> O(1)

          int k = 0; // C -> O(1)

          for(int i = 0; i < nums.length; i++) if(nums[i] == 0){ //n+1 -> O(n)
                 arr[k] = 0; //n * C
                 k++; //n * C

          }

          for(int i = 0; i < nums.length; i++) if(nums[i] != 0){ //n+1 -> O(n)
                 arr[k] = nums[i]; //n * C
                 k++; //n * C
          }

          return arr; //C

  }
```

- The time complexity as above is O(n) since the for produces n+1 steps, it is going to grow on linear terms.

C. bigDiff

```
public int bigDiff(int[] nums) {
          int max = nums[0];  //O(1)
          int min = nums[0];  //O(1)

          for(int i = 0; i < nums.length; i++)  // O(n)
          {
                    max = Math.max(max, nums[i]); //n*C
                    min = Math.min(min, nums[i]); //n*C
```

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

```
        }
        return max - min; //C
}
```

- The asymptotic complexity of “bigDiff” algorithm is O(n), will grow in linear terms

D. **has12**

```
public boolean has12(int[] nums) {
        boolean one = false; //O(1)
        boolean two = false; //O(1)
        for(int i=0; i<nums.length; i++){ //n+1 -> O(n)
                if(nums[i] == 1){   //n*C
                        one = true;   //n*C
                        for(int j = i; j<nums.length; j++){ //O(n)*n -> O(n^2)
                        if(nums[j] == 2) two = true;
                        }
                }
        }
        return (one && two); //O(1)
}
```

- The asymptotic complexity of “has12” algorithm is O(n^2), will grow in second grade polynomial terms

E. **fizzBuzz**

```
public String[] fizzBuzz(int start, int end) {
        String[] array = new String[end - start]; //O(1)
        for(int i = start; i<end; i++){ //O(n)
                if(i%3 == 0 && i%5 == 0) array[i-start] = "FizzBuzz"; //n*C
                else if(i%3 == 0) array[i-start] = "Fizz"; //n*C
                else if(i%5 == 0) array[i-start] = "Buzz"; //n*c
                else array[i-start] = String.valueOf(i); //n*C
        }
        return array;
}
```

- The asymptotic complexity of “fizzBuzz” algorithm is O(n), will grow in linear terms

Array3:

A. **Fix34**

```
public int[] fix34(int[] nums) {
```

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

```
        int[] arr = new int[nums.length]; // C

        int counter = 0; //C

        for(int i = 0; i < nums.length; i++){ //(n+1)*C -> O(n)
            if(nums[i] == 3){ //n*C
                arr[i] = 3;//n*C
                if(i < nums.length-1){ // n*C
                    arr[i+1] = 4; //n*C
                    counter = counter + 2; //n*C
                }
            }
        }

        int[] gar = new int[nums.length-counter]; //C

        counter = 0;  //C
        for(int i = 0; i < nums.length; i++) if(nums[i] != 3 && nums[i] != 4){ //n+1 -> O(n)
            gar[counter] = nums[i]; //n*C
            counter++;  //n*C
        }


        counter = 0; //C


        for(int i = 0; i < nums.length; i++){ //(n+1)*C -> O(n)

            if(arr[i] == 0){ //n* C
                arr[i] = gar[counter]; //n*C
                counter++; //n*C

            }


        }

        return arr; //C


    }
```

- Because of the fastest growing term, this exercise has a time complexity of O(n) for our solution.

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

B. countClumps

```
public int countClumps(int[] nums) {
        if(nums.length == 0) return 0; //C
        int x = nums[0]; //C
        boolean ad = true; //C
        int c = 0; //C
        for(int i = 0; i < nums.length-1; i++){(n+1)*C -> O(n)
                if(x==nums[i+1] && ad){//n*C
                        c++;//n*C
                        ad = false; //n*C
                } else if(x!=nums[i+1]) { //n*C
                        x = nums[i+1]; //n*C
                        ad = true; //n*C
                }
        }
        return c; //C
}
```

- Since this problem executes n+1 times its time complexity is O(n)

C. squareUp

```
public int[] squareUp(int n) {
        int[] arr = new int[n*n]; //C
        int m = n; //C
        int x = 1; //C
```

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(int i = n*n-1; i >= 0; i--){ //(w+1)*C -> O(n)

            if((i+1)%n == 0 && i!=n*n-1){ //C*w
                m--; //C*w
                x = 1; //C*w
            }

            if(x > m){ //C*w

                arr[i] = 0; //C*w

            } else {

                arr[i] = x; //C*w
                x++; //C*w

            }

        }

        return arr; //C

    }
```

- O(n) time complexity, because it executes w+1 times, those times translated to linear behavior, notice that the new array size is n*n, then w = n*n.

D. canBalance

```
public boolean canBalance(int[] nums) {
        int sum1 = 0; //O(1)
        int sum2 = 0; //O(1)
        for(int i=0; i<nums.length; i++){ //O(n)
            sum1 += nums[i]; //O(n)
            for(int j=i+1; j<nums.length; j++){ //O(n)*n ->O(n^2)
                sum2 += nums[j]; //(On^2)
            }
            if(sum1 == sum2) return true; //O(n)
            sum2 = 0; //O(n)
        }
        return false; //O(1)
    }
```

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

- O(n^2) time complexity for “canBalance” algorithm, because it executes O(n) time complexity n times, those time translated to a grade two polynomial behavior.

E. maxSpan

```
public int maxSpan(int[] nums) {
        int span = 0; //O(1)
        int aux = 0; //O(1)
        for(int i=0; i<nums.length; i++){ //O(n)
                for(int j=0; j<nums.length; j++){ //O(n^2)
                        if(nums[j] == nums[i]) aux = (j-i)+1 //O(n^2);
                }
                span = Math.max(span, aux); //O(n)
        }
        return span; //O(1)
}
```

- O(n^2) time complexity for “maxSpan” algorithm, because it executes (n-1)*n times, those time translated to a grade two polynomial behavior.

3.8

Note: We measured the time complexity in a non rigorous way, for example when we said that O(n) time complexity n times is $O(n^2)$, we are not just simply multiplying, it is the result from our analysis, since you have a linear grow and it is executed n times its behavior is going to be quadratic.

Variable	**Meaning**
n	Numbers of elements in an array, note that is also used to define the linear or quadratic grow into the Big-O notation.
c	Constant operations, its cost.
w	Numbers of elements in an array.

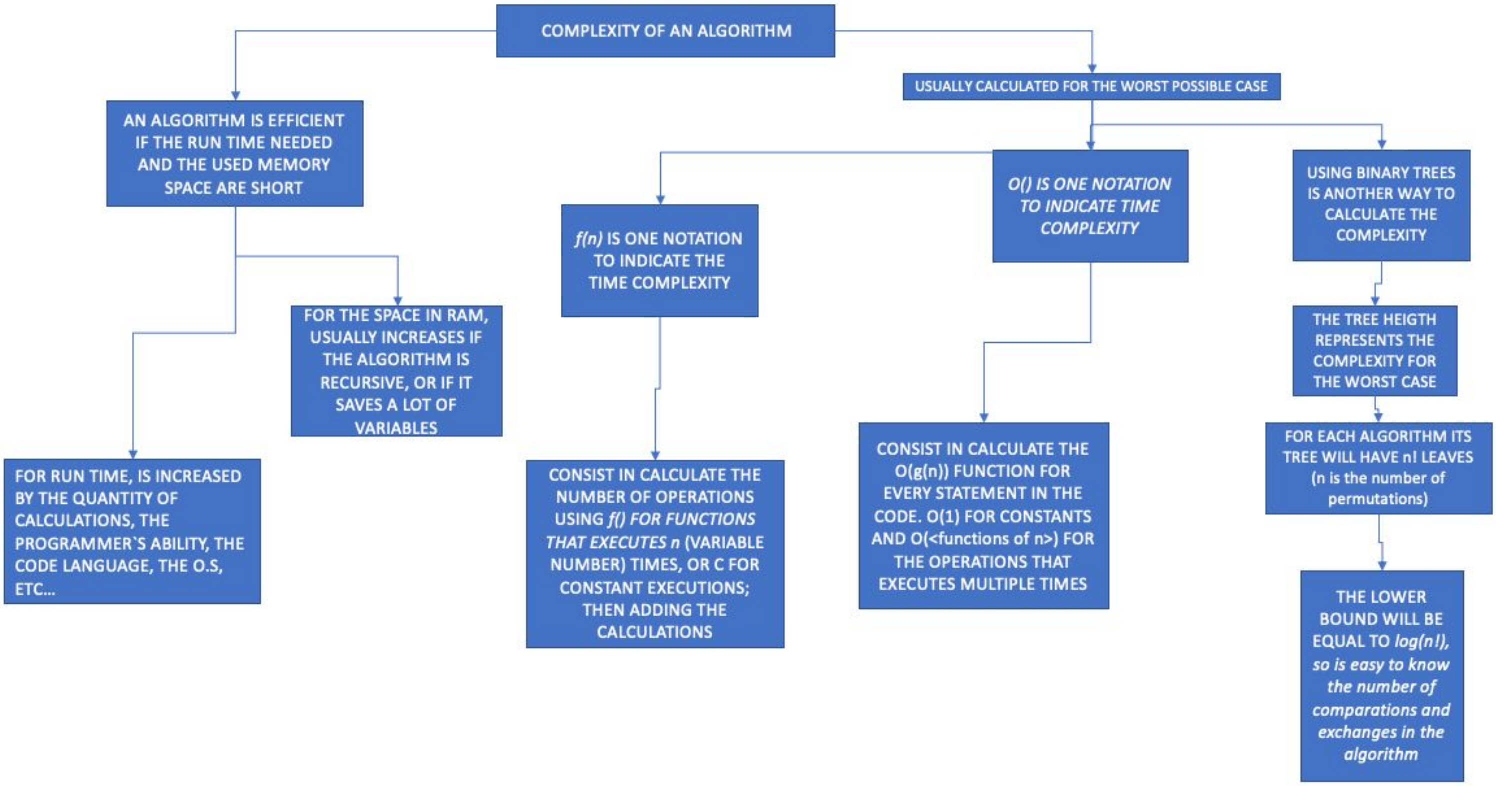
4) Practice for midterms

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

4.1 *c O(n+m)*
4.2 *d O(m*n)*
4.3 *b O(ancho)*
4.4 *b* $O(n^3)$
4.5 *d* $O(n^2)$
4.6 *a T(n-1) + C*
4.7
    4.7.1 T(n) = T(n-1) + C
    4.7.2 O(n)
4.8 *a Esta ejecuta T(n) = T(n-1) + C*
4.9 *c Ejecuta más de n*m pasos*
4.10 *Ejecuta menos de nlogn pasos*
4.11 *Ejecuta T(n) = T(n-1) + T(n-2) + C pasos*
4.12 *c* $O(m*n*\log n+n*m^2+n^2*\log(n)+m^3$
4.13 *c 2T(n/2) + n*
4.14 $O(n^3 + n\log(\log(m)) + m*\mathrm{sqrt}(m))$

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

5) Recommended reading (optional)



6) Team work and gradual progress (optional)

6.1 Meeting minutes

Member	Date	Done	Doing	To do
Miguel	27/08/2019	Insertion Sort Algorithm on C++ implemented.	Merge Sort Algorithm on C++.	Worksheet, points 2, 3, 4 and 6
Miguel	28/08/2019	2 : Array2 Exercises 3 : Array3	Merge Sort Algorithm on C++.	Worksheet point 6.

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

		Exercises on JAVA implemented.	Worksheet point 3 and 4.	
Miguel	28/08/2019	All answered on point 4. Merge Sort on C++ implemented.	Worksheet point 3.	Worksheet point 6.
Miguel	29/08/2019	Time complexity calculated for Miguel point 3 exercises.	Worksheet point 6.	
Pablo	31/08/2019		3 : Array2 Exercises 2 : Array3 Exercises on JAVA.	Worksheet point 3 (graphs and maxSpan explanation) and 5.
Pablo	31/08/2019	3 : Array2 Exercises 2 : Array3 Exercises on JAVA implemented.	Worksheet point 3 (graphs and maxSpan explanation).	Worksheet point 5.
Pablo	01/09/2019	Worksheet point 3 (graphs).	Worksheet point 3 (maxSpan explanation).	Worksheet point 5.
Pablo	01/09/2019	Worksheet point 3 (maxSpan explanation).	Worksheet point 5.	
Pablo	01/09/2019	Worksheet point 5.		
Miguel	01/09/2019	Worksheet point 6.		

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

```
*   commit 61143c7d71c31ef82f8d3b69371f2525ff5759e2 (HEAD -> master, origin/master, origin/HEAD)
|\  Merge: b9b816c cf3f83a
| | Author: miguelmque <macorream@eafit.edu.co>
| | Date:   Sun Sep 1 12:49:11 2019 -0500
| |
| |     Merge remote-tracking branch 'origin'
| |
| * commit cf3f83a5f8d905d475ff7bccec915c7866692bae
| | Author: pablo4buitrago <52968548+pablo4buitrago@users.noreply.github.com>
| | Date:   Sat Aug 31 22:36:53 2019 -0500
| |
| |     Update Array3.java
| |
| |     Added missing exercises
| |
| * commit 0fffb6ca78bedfa22d0d8b8cda8eac482fdd3685
| | Author: pablo4buitrago <52968548+pablo4buitrago@users.noreply.github.com>
| | Date:   Sat Aug 31 21:41:29 2019 -0500
| |
| |     Update Array2.java
| |
| |     Added the missing execises
| |
* | commit b9b816cf5638be50ea59d179c1a2beb5ed493496
|/  Author: miguelmque <macorream@eafit.edu.co>
|   Date:   Sun Sep 1 12:48:08 2019 -0500
|
|       forgot to comment @return Void on insertionSort and helper method
|
* commit 5e800aabfd5ea238a03f92284554bd370d78ba5e
:
| * commit cf3f83a5f8d905d475ff7bccec915c7866692bae
| | Author: pablo4buitrago <52968548+pablo4buitrago@users.noreply.github.com>
| | Date:   Sat Aug 31 22:36:53 2019 -0500
| |
| |     Update Array3.java
| |
| |     Added missing exercises
```

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

```
| |
| * commit 0fffb6ca78bedfa22d0d8b8cda8eac482fdd3685
| | Author: pablo4buitrago <52968548+pablo4buitrago@users.noreply.github.com>
| | Date:   Sat Aug 31 21:41:29 2019 -0500
| |
| |     Update Array2.java
| |
| |     Added the missing execises
| |
* | commit b9b816cf5638be50ea59d179c1a2beb5ed493496
|/  Author: miguelmque <macorream@eafit.edu.co>
|   Date:   Sun Sep 1 12:48:08 2019 -0500
|
|       forgot to comment @return Void on insertionSort and helper method
|
* commit 5e800aabfd5ea238a03f92284554bd370d78ba5e
| Author: miguelmque <macorream@eafit.edu.co>
| Date:   Wed Aug 28 23:00:35 2019 -0500
|
|     documentation done
|
|     documentation done
|
* commit e7965eaced2d1470268f9223fc1774f7b700a431
| Author: miguelmque <macorream@eafit.edu.co>
| Date:   Wed Aug 28 20:18:03 2019 -0500
|
|     point 1 solved, c++ is great
|
* commit ce4871632d61b8c1dcb8ef723decd95563bcd81c
| Author: miguelmque <macorream@eafit.edu.co>
| Date:   Wed Aug 28 12:42:04 2019 -0500
|
|     still wrong, have to manage memory C++ problems
|
* commit d3bba97eaa995ba9d4543e5f54716d9fafbe68b2
| Author: miguelmque <macorream@eafit.edu.co>
| Date:   Wed Aug 28 12:40:17 2019 -0500
|
|     3 Exercise commited, my part
|
* commit c3afe7a66c4139245c749814024b9171532f02fc
```

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

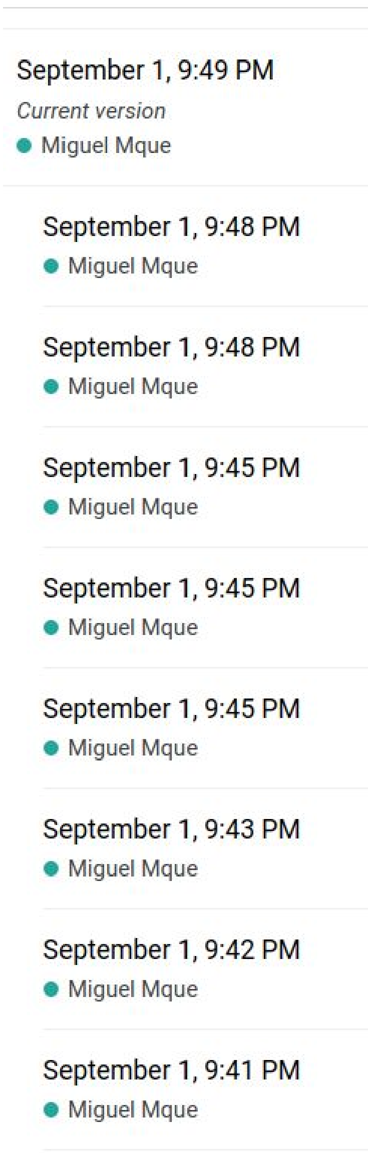
```
| Author: miguelmque <macorream@eafit.edu.co>
| Date:   Wed Aug 28 12:34:31 2019 -0500
|
|     Array2 section added
|
* commit 439c5c77fd67713e48bf5101f7cd53799680c8e9
| Author: miguelmque <macorream@eafit.edu.co>
| Date:   Sun Aug 25 16:29:31 2019 -0500
|
|     Taller08 solved on C++ with all optional points
|
* commit 8d4f837a4c692eb050728f233923a555b4830dba
| Author: miguelmque <macorream@eafit.edu.co>
| Date:   Sun Aug 25 15:17:07 2019 -0500
|
|
|     Taller08 solved on c++, but without optional points
|
* commit f299b7e1d10a44b8290a044a97d729ff1c82bc7e
| Author: miguelmque <macorream@eafit.edu.co>
| Date:   Sun Aug 25 12:39:13 2019 -0500
|
|     Taller08 point 1 solved
|
* commit 329d210b035923e52f8d8bfb2e4a090d0cda2351
| Author: miguelmque <macorream@eafit.edu.co>
| Date:   Sun Aug 25 00:00:37 2019 -0500
|/  Author: miguelmque <macorream@eafit.edu.co>
|   Date:   Sun Sep 1 12:48:08 2019 -0500
|
|       forgot to comment @return Void on insertionSort and helper method
|
* commit 5e800aabfd5ea238a03f92284554bd370d78ba5e
| Author: miguelmque <macorream@eafit.edu.co>
| Date:   Wed Aug 28 23:00:35 2019 -0500
|
|     documentation done
|
* commit e7965eaced2d1470268f9223fc1774f7b700a431
| Author: miguelmque <macorream@eafit.edu.co>
| Date:   Wed Aug 28 20:18:03 2019 -0500
|
```

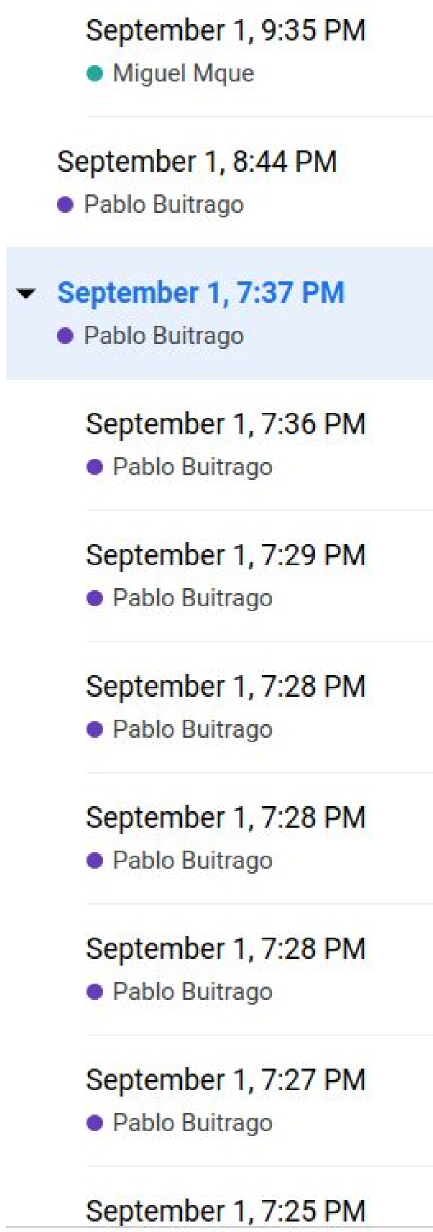
PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

```
|     point 1 solved, c++ is great
|
* commit ce4871632d61b8c1dcb8ef723decd95563bcd81c
| Author: miguelmque <macorream@eafit.edu.co>
| Date:   Wed Aug 28 12:42:04 2019 -0500
|
|     still wrong, have to manage memory C++ problems
|
* commit d3bba97eaa995ba9d4543e5f54716d9fafbe68b2
| Author: miguelmque <macorream@eafit.edu.co>
| Date:   Wed Aug 28 12:40:17 2019 -0500
|
|     3 Exercise commited, my part
|
* commit c3afe7a66c4139245c749814024b9171532f02fc
| Author: miguelmque <macorream@eafit.edu.co>
| Date:   Wed Aug 28 12:34:31 2019 -0500
|
|     Array2 section added
|
```

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

- Pablo Buitrago

September 1, 7:25 PM
- Pablo Buitrago

September 1, 7:23 PM
- Pablo Buitrago

September 1, 7:21 PM
- Pablo Buitrago

September 1, 7:20 PM
- Pablo Buitrago

September 1, 7:19 PM
- Pablo Buitrago

September 1, 7:18 PM
- Pablo Buitrago

September 1, 7:16 PM
- Pablo Buitrago

September 1, 7:15 PM
- Pablo Buitrago

September 1, 7:15 PM
- Pablo Buitrago

- Pablo Buitrago

September 1, 7:12 PM
- Pablo Buitrago

September 1, 7:09 PM
- Pablo Buitrago

September 1, 7:09 PM
- Pablo Buitrago

September 1, 7:09 PM
- Pablo Buitrago

September 1, 7:08 PM
- Pablo Buitrago

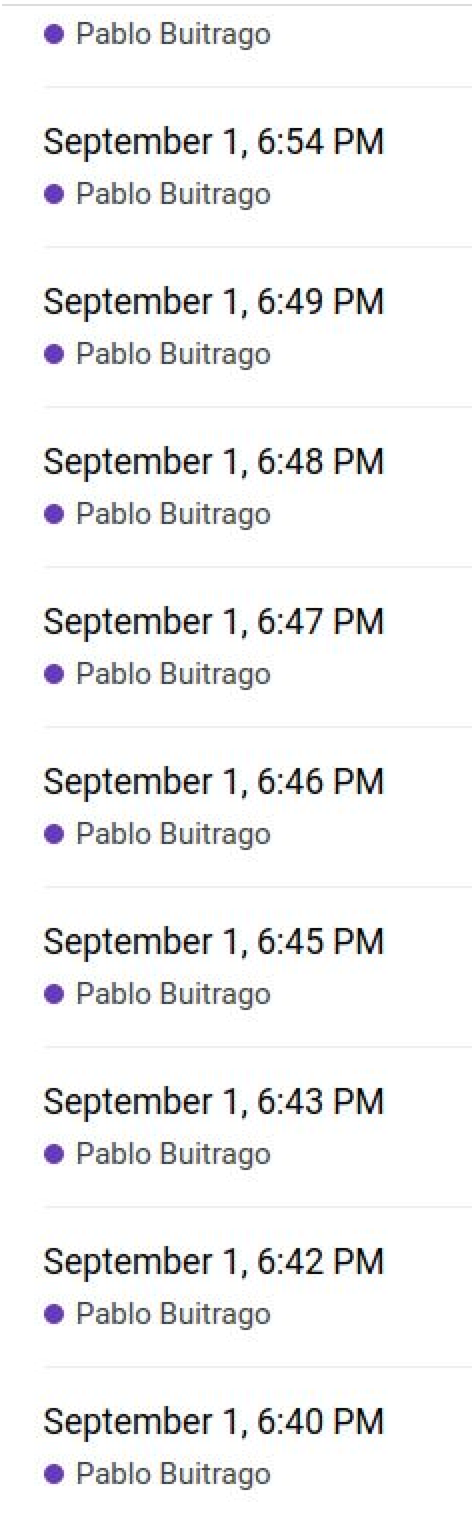
September 1, 7:07 PM
- Pablo Buitrago

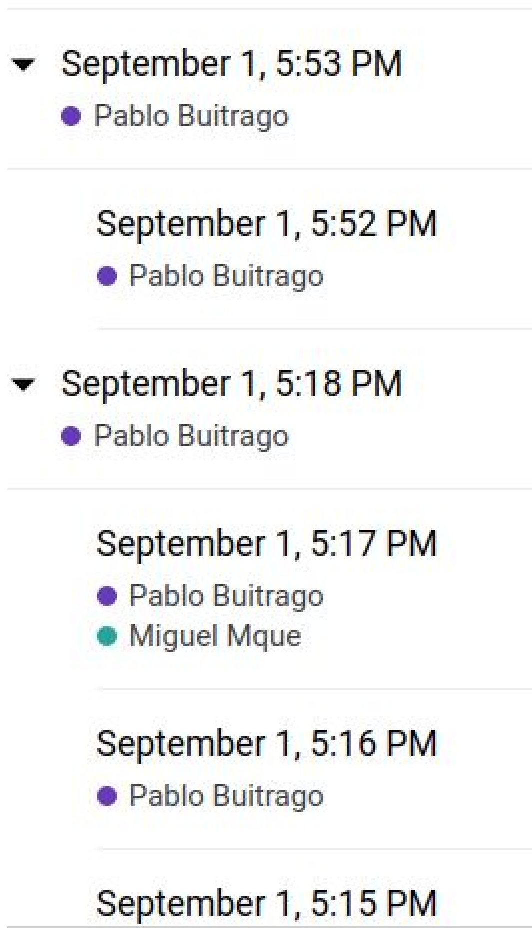
September 1, 7:05 PM
- Pablo Buitrago

September 1, 7:05 PM
- Pablo Buitrago

September 1, 7:05 PM
- Pablo Buitrago

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


Pablo Buitrago
September 1, 6:54 PM
Pablo Buitrago
September 1, 6:49 PM
Pablo Buitrago
September 1, 6:48 PM
Pablo Buitrago
September 1, 6:47 PM
Pablo Buitrago
September 1, 6:46 PM
Pablo Buitrago
September 1, 6:45 PM
Pablo Buitrago
September 1, 6:43 PM
Pablo Buitrago
September 1, 6:42 PM
Pablo Buitrago
September 1, 6:40 PM
Pablo Buitrago


September 1, 5:53 PM
Pablo Buitrago
September 1, 5:52 PM
Pablo Buitrago
September 1, 5:18 PM
Pablo Buitrago
September 1, 5:17 PM
Pablo Buitrago
Miguel Mque
September 1, 5:16 PM
Pablo Buitrago
September 1, 5:15 PM

Pablo Buitrago

September 1, 5:15 PM
Pablo Buitrago

September 1, 5:13 PM
Pablo Buitrago

September 1, 5:13 PM
Pablo Buitrago

September 1, 5:12 PM
Pablo Buitrago

September 1, 5:11 PM
Pablo Buitrago

September 1, 5:10 PM
Pablo Buitrago

September 1, 5:10 PM
Pablo Buitrago

September 1, 5:01 PM
Pablo Buitrago

September 1, 5:00 PM
Pablo Buitrago

August 29, 1:11 AM
Miguel Mque

August 29, 1:10 AM
Miguel Mque

August 29, 1:09 AM
Miguel Mque

August 29, 1:05 AM
Miguel Mque

August 29, 1:04 AM
Miguel Mque

August 29, 1:04 AM
Miguel Mque

August 29, 1:02 AM
Miguel Mque

August 29, 1:02 AM
Miguel Mque

August 29, 1:00 AM
Miguel Mque

August 29, 1:00 AM
Miguel Mque

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

● Miguel Mque

August 28, 11:30 PM
● Miguel Mque

August 28, 11:29 PM
● Miguel Mque

August 28, 11:23 PM
● Miguel Mque

August 28, 11:23 PM
● Miguel Mque

August 28, 11:22 PM
● Miguel Mque

August 28, 11:21 PM
● Miguel Mque

August 28, 11:21 PM
● Miguel Mque

August 28, 11:20 PM
● Miguel Mque

August 28, 11:18 PM
● Miguel Mque

● Miguel Mque

August 28, 11:17 PM
● Miguel Mque

August 28, 11:16 PM
● Miguel Mque

August 28, 11:16 PM
● Miguel Mque

August 28, 11:15 PM
● Miguel Mque

August 28, 11:14 PM
● Miguel Mque

August 28, 11:14 PM
● Miguel Mque

August 28, 11:14 PM
● Miguel Mque

August 28, 11:13 PM
● Miguel Mque

August 28, 11:12 PM
● Miguel Mque

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

Miguel Mque

August 28, 11:11 PM
Miguel Mque

August 28, 11:10 PM
Miguel Mque

August 28, 11:09 PM
Miguel Mque

August 28, 11:07 PM
Miguel Mque

August 28, 11:06 PM
Miguel Mque

August 28, 11:06 PM
Miguel Mque

August 28, 11:05 PM
Miguel Mque

August 28, 11:05 PM
Miguel Mque

August 28, 11:03 PM
Miguel Mque

WEDNESDAY

August 28, 1:32 PM
Miguel Mque

August 28, 1:31 PM
Miguel Mque

August 28, 1:30 PM
Miguel Mque

August 28, 1:23 PM
Miguel Mque

August 28, 1:21 PM
Miguel Mque

August 28, 1:17 PM
Miguel Mque

TUESDAY

August 27, 9:08 PM
Miguel Mque

August 27, 9:05 PM
Miguel Mque
Imported .docx file - View original

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