# Laboratorio Nro. 2: Notación O Grande

**Juan Pablo Giraldo Restrepo**
Universidad Eafit
Medellín, Colombia
jpgiraldor@eafit.edu.co

**Mateo Sánchez Toro**
Universidad Eafit
Medellín, Colombia
msanchezt@eafit.edu.co

**Juan Felipe Londoño Gaviria**
Universidad Eafit
Medellín, Colombia
jflondonog@eafit.edu.co

## 3) Project questions drill

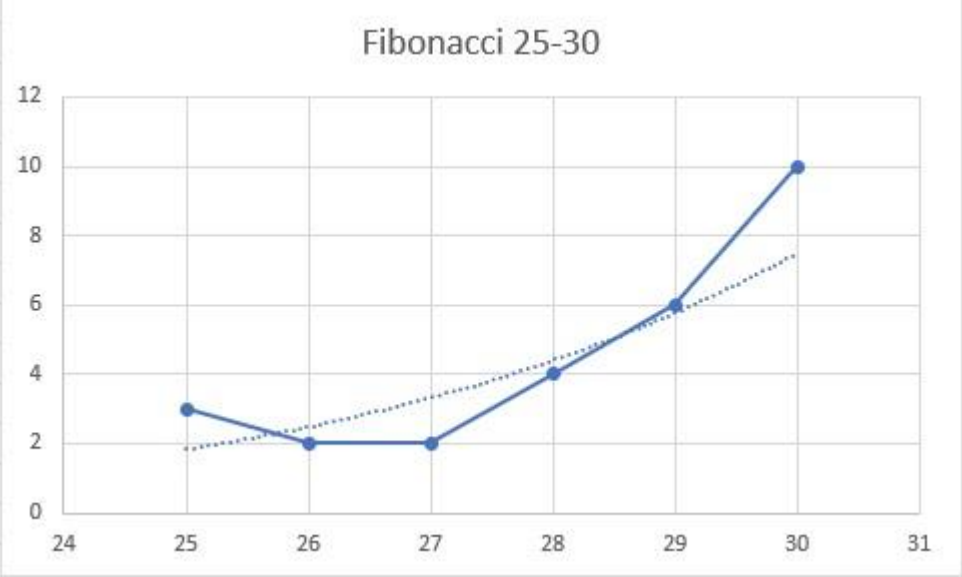
### 3.1

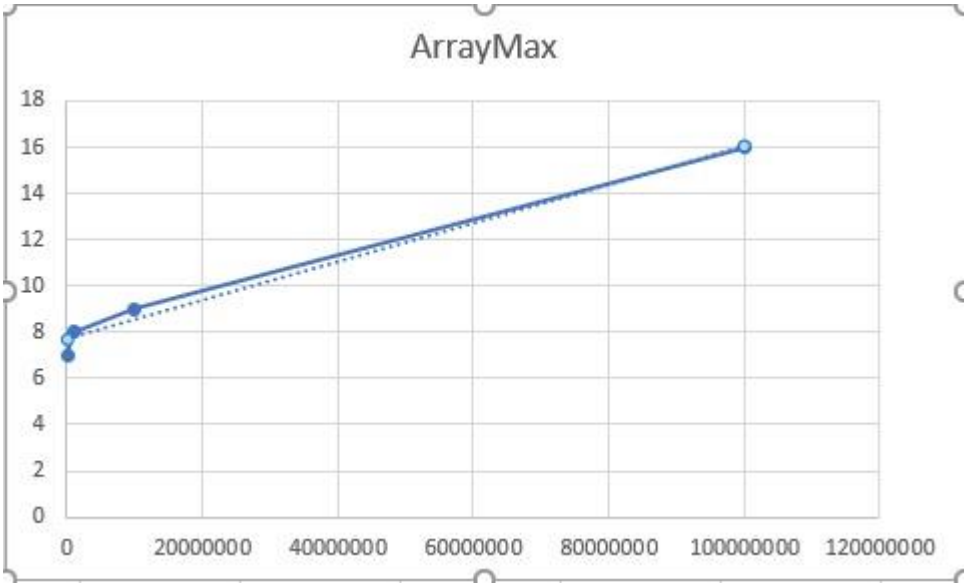
| FIBONACCI | | | | | | |
|---|---|---|---|---|---|---|
| N | 25 | 26 | 27 | 28 | 29 | 30 |
| TIEMPO | 3 | 2 | 2 | 4 | 6 | 10 |

| ARRAYMAX | | | | |
|---|---|---|---|---|
| N | 100.000 | 1.000.000 | 10.000.000 | 100.000.000 |
| TIEMPO | 7 | 8 | 9 | 16 |

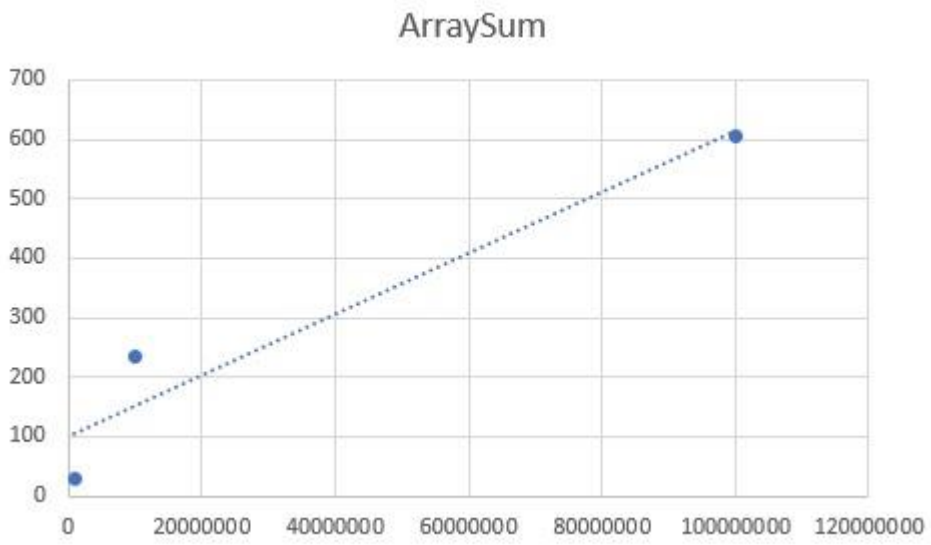
| ARRAYSUM | | | | |
|---|---|---|---|---|
| N | 100.000 | 1.000.000 | 10.000.000 | 100.000.000 |
| TIEMPO | 5 | 28 | 233 | 605 |

### 3.2




**DOCENTE MAURICIO TORO BERMÚDEZ**
**Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627**
**Correo: mtorobe@eafit.edu.co**




### 3.3

It is concluded such arraySum as well as arrayMax are lineal, in other words O(n) and Fibonacci is O(n^2). For this reason Fibonacci takes too much time for large values, that's why the other values could not be calculated.

### 3.4

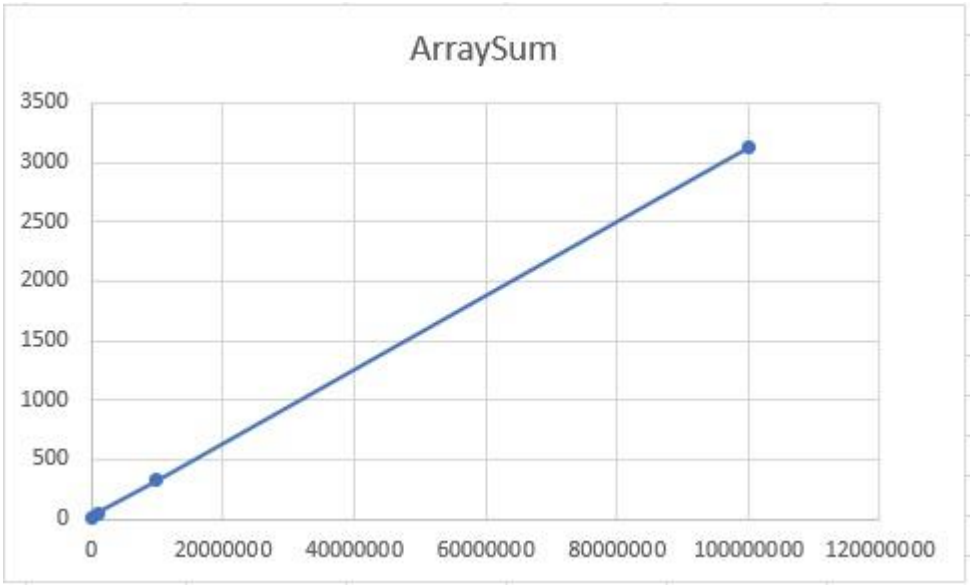
| ArraySum | | | | |
|---|---|---|---|---|
| N | 100000 | 1000000 | 10000000 | 100000000 |
| TIEMPO | 11 | 49 | 326 | 3125 |

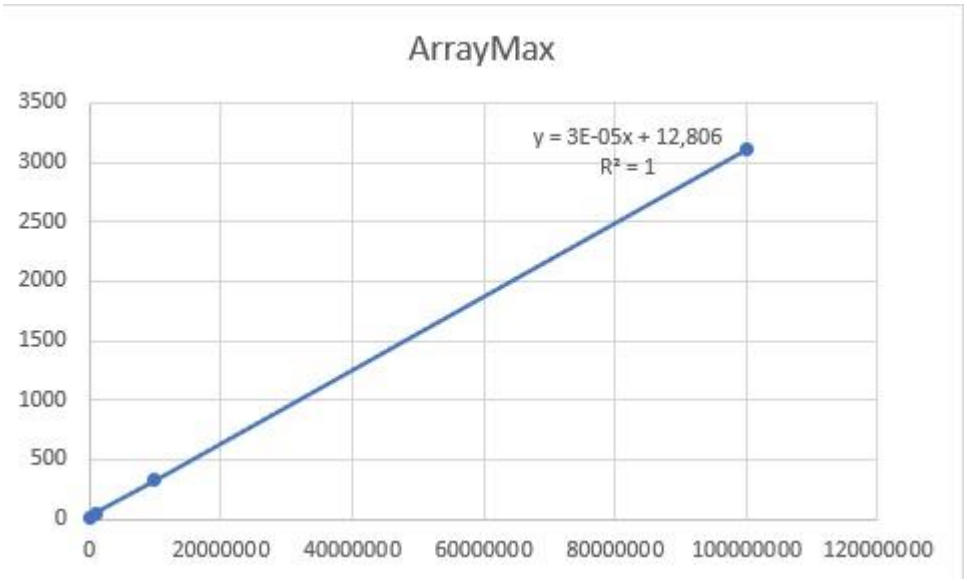
| ArrayMax | | | | |
|---|---|---|---|---|
| N | 100000 | 1000000 | 10000000 | 100000000 |
| TIEMPO | 11 | 47 | 324 | 3106 |

| Insertion Sort | | | | |
|---|---|---|---|---|
| N | 100000 | 1000000 | 10000000 | 100000000 |
| TIEMPO | 7329 | 731133 | Más de 5 | Más de 5 |

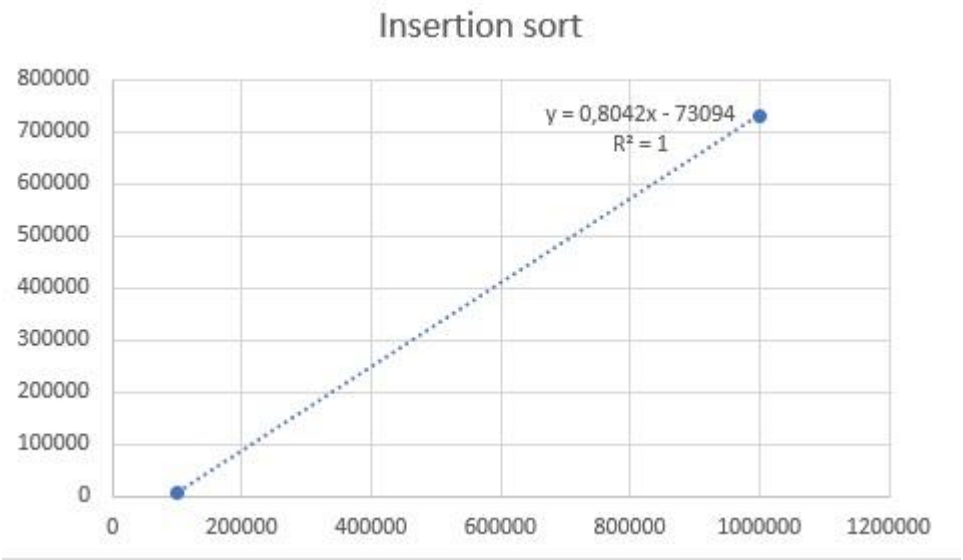
| MergeSort | | | | |
|---|---|---|---|---|
| N | 100000 | 1000000 | 10000000 | 100000000 |
| TIEMPO | 70 | 473 | 4367 | 43589 |

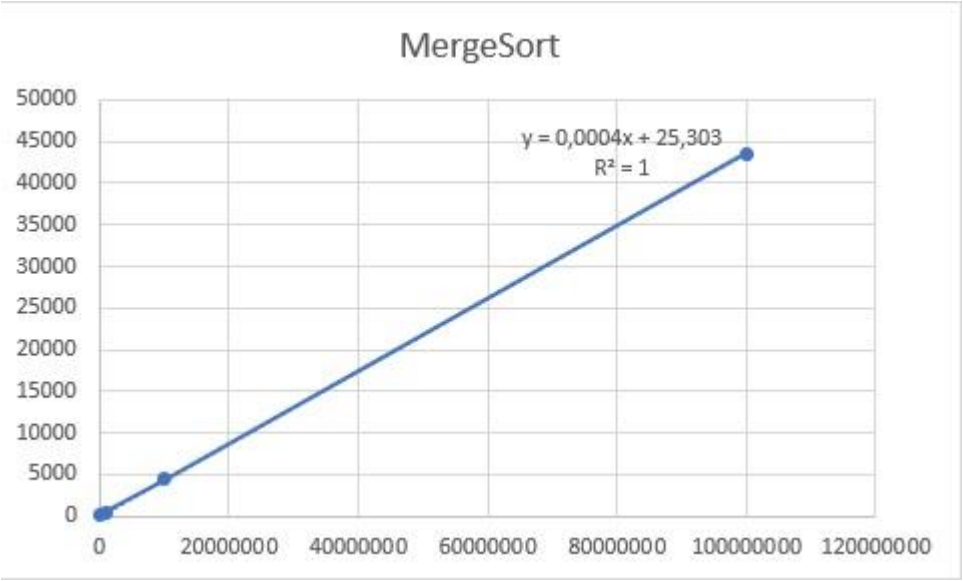
3.5




DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co




**3.6**

All times are lineal in all the graphics and big O notation. One would think that insertion sort is no lineal due to the amount of time that it takes, but regardless of this even in the worst case it's lineal.

**3.7**

The complexity of insertion sort is n^2, what means that when using large arrays, the amount of instructions will be n times larger than the length of the array and will take much more time.

**3.8**

Due the complexity of array Sum being O(n) the number of instructions done by the algorithm will be the same as the length of the array, n times smaller than insertion Sort.

### 3.9

The complexity of merge sort when searched on the internet is of (nLogn) but when we took the times for different array sizes the graph showed a lineal complexity. Regardless of this the complexity is lower than the one for insertion sort.

### 3.10

**maxSpan:** The algorithm consists of two "for loops" which will used for a double check of the array, and two variables, "maxSpan" and "currentSpan". "maxSpan" will serve as champion counter and will allow the algorithm to compare the last longer span with a possible new span. "currentSpan" is used to save a new span and then compare it with the span in the champion counter. Both used loops are nested, and in the first loop with the variable i, initialized in 0, the variable "currentSpan" is initialized in 0 since this counter must be restarted every time we finish counting a span. The second loop with the variable j, initialized in i, since we do not care about the numbers behind the index i because they were already compared. In this loop we look for a pair for the position [i] in the index j of the array, since if in the index i there's the same number that is in the index j then there will be a span from the position [i] to the position [j], which is equal to the subtraction: j - (i + 1), we add 1 since the position i starts from 0. At the end the variable "maxSpan" will be returned since it saves the length of the longest span.

### 3.11

**-Array2:**

- countEvens: O(n)
- bigDiff: O(n)
- lucky13: O(n)
- sum28: O(n)
- more14: O(n)

**-Array3:**

- maxSpan: O(n²)
- fix34: O(n²)
- fix45: O(n²)
- canBalance: O(n²)
- linearIn: O(n²)

### 3.12

**-Array2:**

In this set of exercises n is the size of the array and there is no m.

**-Array3**
In this exercise set the n is the array size but in linearIn besides the n, the m is the size of the second array.

### *4) Exam Drill*

1. *c*
2. *b*
3. *b*
4. *b*
5. *d*
6. *d*

7.1 *C_2 + T(n-1)*
7.2 *O(n)*

**DOCENTE MAURICIO TORO BERMÚDEZ**
**Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627**
**Correo: mtorobe@eafit.edu.co**