

Laboratorio Nro. 2: Notación O grande

Kevin Arley Parra Henao
Universidad EAFIT
Medellín, Colombia
kaparrah@eafit.edu.co

Daniel Alejandro Mesa Arango
Universidad EAFIT
Medellín, Colombia
damesaa@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos.

Se encuentra en los numerales indicados en la guía.

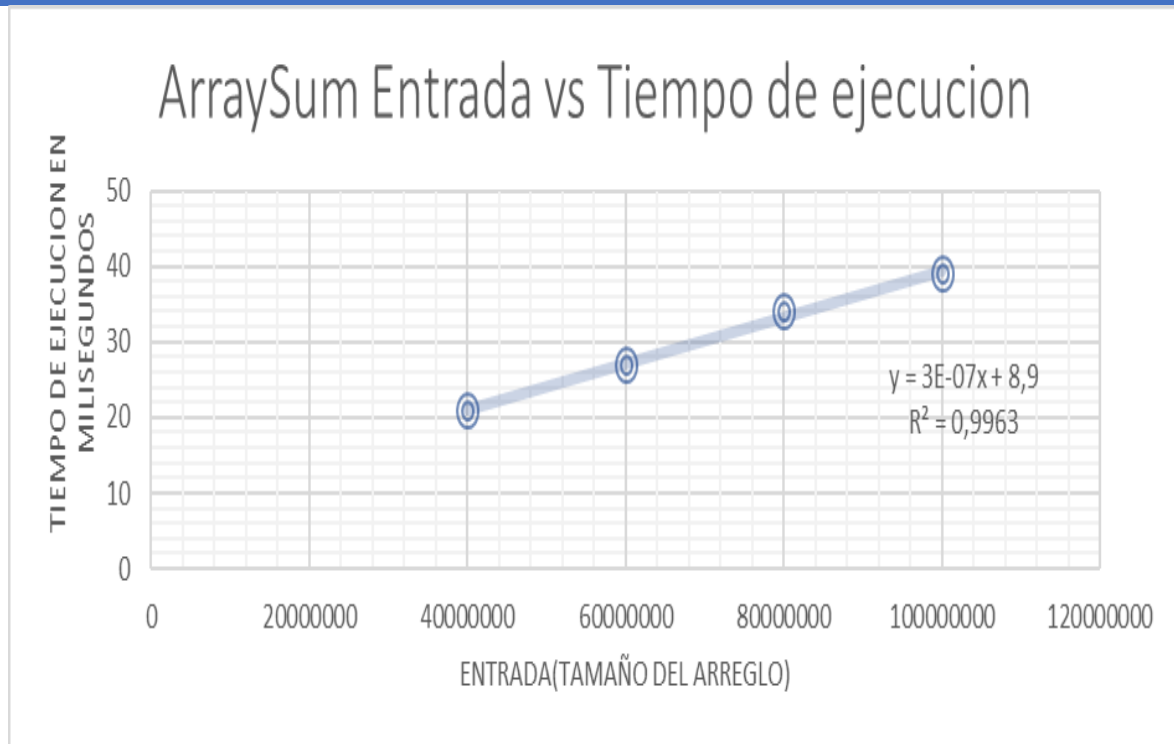
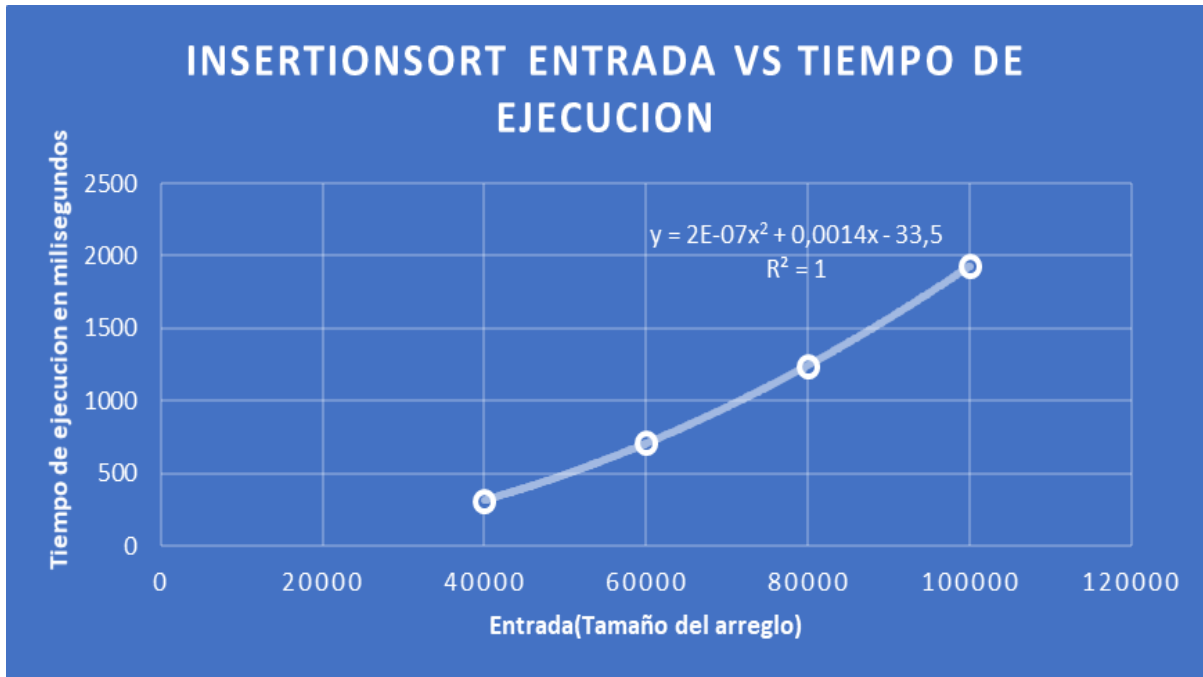
3.1) Tiempos para cada uno de los algoritmos.

	N = 40'000.000	N = 60'000.000	N = 80'000.000	N = 100'000.000
Array sum	21	27	34	39
Array maximum	21	27	33	41

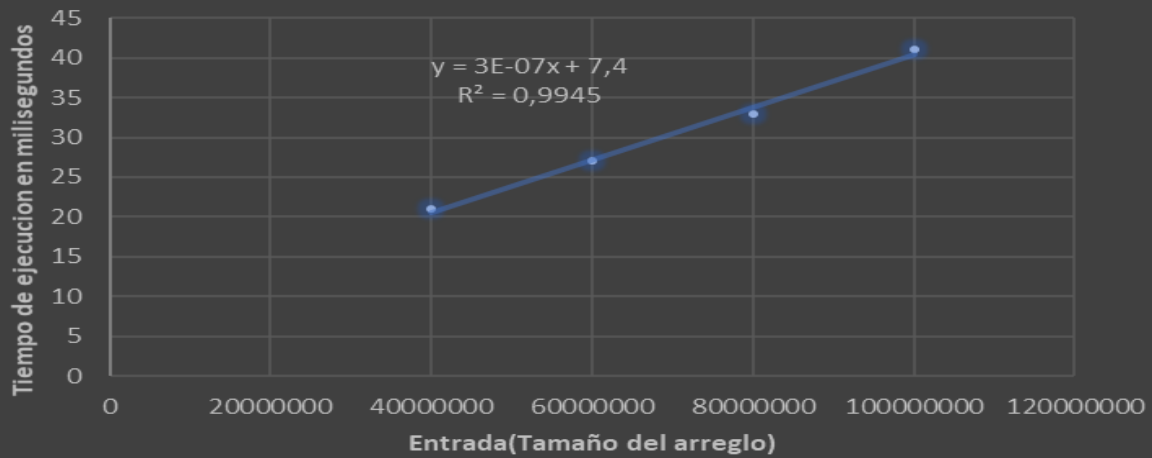
Merge Sort	
Entada N	Tiempo en milisegundos
40000000	0,00016323
60000000	1,72E-04
80000000	0,00017131
100000000	0,00016659
120000000	1,68E-04
140000000	1,69E-04
160000000	1,69E-04
180000000	1,71E-04
200000000	0,00017047

220000000	1,74E-04
240000000	1,75E-04
260000000	1,75E-04
280000000	1,77E-04
300000000	0,00017441
320000000	0,00017208
340000000	1,75E-04
360000000	1,73E-04
380000000	1,77E-04
400000000	0,00017403

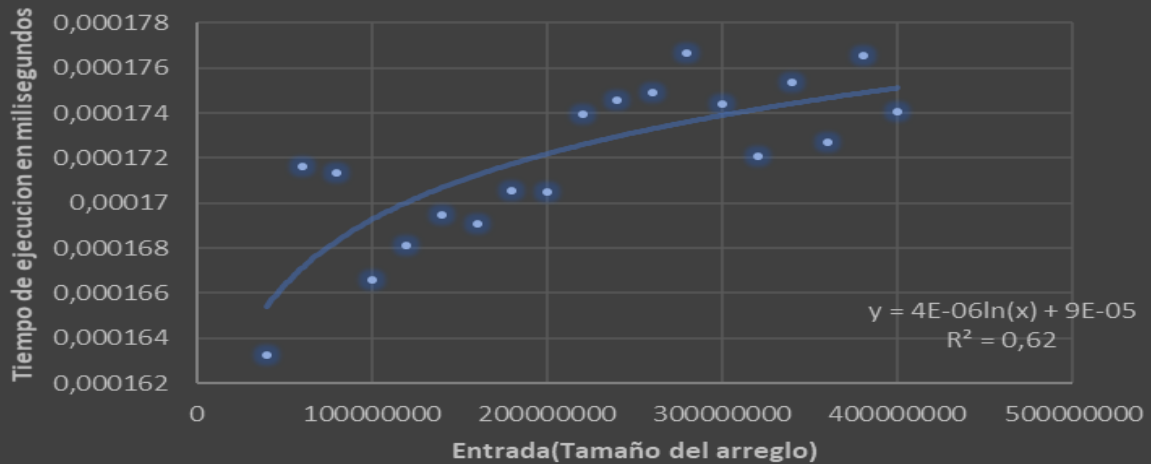
	N = 40000	N = 60000	N = 80000	N = 100000
Insertion sort	313	712	1242	1933

3.2) Grafica de los tiempos tomados en el numeral anterior.

ArrayMax Entrada vs Tiempo de ejecucion



Merge Sort Entrada vs Tiempo de ejecucion



DOCENTE MAURICIO TORO BERMÚDEZ

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627

Correo: mtorobe@eafit.edu.co

3.3) ¿Qué concluyen con respecto a los tiempos obtenidos en el laboratorio y los resultados teóricos obtenidos con la notación O?

Los resultados del laboratorio dan acorde a los resultados de la notación O, tanto en ArraySum como en ArrayMax su comportamiento es lineal, es decir $O(n)$, para InsertionSort los tiempos son bastante con valores grandes llegando a varios minutos con un tamaño de problema de 100000, su notación es polinómica es $O(n^2)$, para mergeSort arroja buenos tiempos con entradas grandes ya que es $O(n(\log n))$ cada una de las gráficas corresponde a su línea de tendencia ajustándose a lo experimentado con lo teórico.

3.4) Teniendo en cuenta lo anterior, ¿Qué sucede con *Insertion Sort* para valores grandes de N?

Se convierte en un algoritmo ineficiente ya que tiene que recorrer $n*n$ veces el arreglo y con valores muy grandes este tiende a tardar demasiado tiempo rondando los 100000 elementos en el arreglo tiende a tardar varios minutos, depende también de que tan desorganizado este el arreglo, aunque considerando siempre el peor de los casos terminaría por recorrer el arreglo dos veces.

3.5) Teniendo en cuenta lo anterior, ¿Qué sucede con *ArraySum* para valores grandes de N? ¿Por qué los tiempos no crecen tan rápido como *Insertion Sort*?

ArraySum, suma todos los elementos de un arreglo sin importar su orden, por esto solo hace un recorrido al arreglo es decir las operaciones se hacen n veces comportándose de manera casi constante, mientras InsertionSort recorre el arreglo dos veces al tiempo haciendo $n*n$ operaciones ya que lo que hace este algoritmo es ordenar los elementos de menor a mayor en el arreglo.

3.6) Teniendo en cuenta lo anterior, ¿Qué tan eficiente es *Merge sort* con respecto a *Insertion sort* para arreglos grandes? ¿Qué tan eficiente es *Merge sort* con respecto a *Insertion sort* para arreglos pequeños?

Para arreglos grandes Merge sort es mucho más eficiente pues lo máximo de complejidad que alcanza dentro de su algoritmo es $O(n(\log n))$ mientras que Insertion sort es $O(n^2)$ Merge divide el problema, mientras que Insertion no lo hace.

3.7) Expliquen con sus propias palabras cómo funciona el ejercicio *maxSpan* y ¿por qué?

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

- Declaramos las variables maxS, que se usara para retornar el Span más largo, maxR que tomara el máximo span por cada valor, y span que se usara para contar y saber cuánto mide un span, se inicia en uno para incluir al primer valor

```
int maxS = 0;  
int maxR = 0;  
int span = 1;
```

- Establecemos un ciclo que recorra el arreglo

```
for (int i = 0; i < nums.length; i++) {
```

- Inicializamos cada span y maxR en 1

```
span = 1;  
maxR = 1;
```

- Establecemos otro ciclo que empezara en la posición de la variable i + 1, para recorrer los elementos que están más adelante en el arreglo

```
for (int j = i + 1; j < nums.length; j++) {
```

- Sumamos al span de uno en uno para contar los elementos hasta que volvamos a encontrar el mismo número que está en la posición num[i]

```
span = span + 1;
```

- Establecemos un condicional que verifica si encontramos otro elemento igual al que esa en la posición nums[i]

```
if (nums[i] == nums[j]) {
```

- Si encuentra otro elemento igual, verifica que el span sean mayor al maxR y asigna al maxR el valor de este, así nos aseguramos de que tome el mayor span para un valor

```
        if (span > maxR) {  
            maxR = span;  
        }  
    }  
}
```

- Verifica si maxR es mayor a maxS y de ser así asigna a maxS el valor de maxR

```
        if (maxR > maxS) {  
            maxS = maxR;  
        }  
    }  
}
```

- Retorna el valor de maxS que será el mayor span encontrado
return maxS;
}

3.8) Calculo de complejidad de los ejercicios en línea.

//EJERCICIOS ARRAY 2:

```
i. public int countEvens(int[] nums) {  
    int c = 0;                                //C1  
    for (int i = 0; i < nums.length; i++) {   //C2 + C3*n  
        if (nums[i] % 2 == 0) {               //C4*n  
            c++;  
        }  
    }  
    return c;                                //C5  
}
```

Complejidad:

$$T(n) = C1 + C2 + C3*n + C4*n + C5$$

$$T(n) = C + (C3+C5)*n$$

$$T(n) \text{ es } O(n)$$

```
ii. public int bigDiff(int[] nums) {
```

```
int max = nums[0];           //C1
int min = nums[0];           //C2
for (int i = 0; i < nums.length; i++) { //C3 + C4*n
    max = Math.max(max, nums[i]); //C5*n
    min = Math.min(min, nums[i]); //C6*n
}
return max - min;           //C7
}
```

Complejidad:

$$T(n) = C1 + C2 + C3*n + C4*n + C5*n + C7$$

$$T(n) = C + (C3+C4+C5)*n$$

T(n) es O(n)

```
iii. public int centeredAverage(int[] nums) {
    int max = nums[0];           //C1
    int min = nums[0];           //C2
    int sum = 0; //C3
    for (int i = 0; i < nums.length; i++) { //C4 + C5*n
        max = Math.max(max, nums[i]); //C6*n
        min = Math.min(min, nums[i]); //C7*n
        sum += nums[i];           //C8*n
    }
    return ((sum - min) - max) / (nums.length - 2); //C9
}
```

Complejidad:

$$T(n) = C1 + C2 + C3 + C4 + C5*n + C6*n + C7*n + C8*n + C9$$

$$T(n) = C + (C5+C6+C7+C8)*n$$

T(n) es O(n)

```
iv. public int[] evenOdd(int[] nums) {
    int count = 0;           //c
    int count2 = nums.length-1; //c
    int [] nums2 = new int[nums.length]; //c
    for(int i = 0; i < nums.length; i++) //c + c*n
    {
```

DOCENTE MAURICIO TORO BERMÚDEZ

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627

Correo: mtorobe@eafit.edu.co

```
if(nums[i] % 2 == 0)           //c*n + c*n
{
    nums2[count] = nums[i];    // c*n +c*n
    count++;                  //c*n
}else if(nums[i] % 2 != 0)    //c*n + c*n
{
    nums2[count2] = nums[i];   //c*n + c*n
    count2--;                  // c*n
}
}
return nums2;                  //c
}
```

Complejidad:

$T(n) = c + c + c + c + c + c + c*n + c*n + c*n + c*n + c*n + c*n + c*n + c*n + c*n + c*n$

$T(n)$ es $O(c + c + c + c + c + c + c*n + c*n + c*n + c*n + c*n + c*n + c*n + c*n + c*n)$

$T(n)$ es $O(c*n)$ RS

$T(n)$ es $O(n)$ RP

```
v. public boolean sum28(int[] nums) {
    int sum = 0;    //c
    for(int i = 0; i < nums.length; i++)//c+ c*n
    {
        if(nums[i] == 2)    //c*n + c*n
        {
            sum = sum + 2;    //c*n
        }
    }
    if(sum == 8)return true;    //c //c
    return false;              //c
}
```

Complejidad:

$T(n) = c + c + c + c + c + c + c*n + c*n + c*n + c*n$

$T(n)$ es $O(c + c + c + c + c + c + c*n + c*n + c*n + c*n)$

$T(n)$ es $O(c*n)$ RS

$T(n)$ es $O(n)$ RP

DOCENTE MAURICIO TORO BERMÚDEZ

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627

Correo: mtorobe@eafit.edu.co

//EJERCICIOS ARRAY 3:

```
i. public int maxSpan(int[] nums) {  
    int maxS = 0; //}  
    int maxR = 0; // }C1  
    int span = 1; //}  
    for (int i = 0; i < nums.length; i++) { //C2 + C3*n  
        span = 1; //C4*n  
        maxR = 1; //C5*n  
        for (int j = i + 1; j < nums.length; j++) { //C6*n + C7*n*(n-(i+1))  
            span = span + 1; //C8*n*(n-(i+1))  
            if (nums[i] == nums[j]) { //C9*n*(n-(i+1))  
                if (span > maxR) {  
                    maxR = span;  
                }  
            }  
        }  
        if (maxR > maxS) { //C10*n*(n-(i+1))  
            maxS = maxR;  
        }  
    }  
    return maxS; //C11  
}
```

Complejidad:

$$T(n) = C1 + C2 + C3*n + C4*n + C5*n + C6*n + C7*n*(n-(i+1)) + C8*n*(n-(i+1)) + C9 + C10*n*(n-(i+1)) + C11$$

$$T(n) = C + C3*n + C4*n + C5*n + C6*n + C7*n*(n-(i+1)) + C8*n*(n-(i+1)) + C10*n*(n-(i+1))$$

$$T(n) = C + C(n) + C(n)(n-(i+1))$$

$$T(n) = C + C(n) + Cn^2 - Cn*i - Cn$$

Sabemos que $i \leq n$, luego

$$T(n) \text{ es } O(n^2)$$

DOCENTE MAURICIO TORO BERMÚDEZ

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627

Correo: mtorobe@eafit.edu.co

```
ii. public int[] fix34(int[] nums) {  
    int temp = 0; //C1  
    for (int i = 0; i < nums.length; i++) { //C2 + C3*n  
        if (nums[i] == 3) { //C4*n  
            for (int j = 1; j < nums.length; j++) { //C5*n + C6*n*(n-1)  
                if (nums[j] == 4 && nums[j - 1] != 3) { //C7*n*(n-1)  
                    temp = nums[i + 1];  
                    nums[i + 1] = nums[j];  
                    nums[j] = temp;  
                    break;  
                }  
            }  
        }  
    }  
    return nums; //C8  
}
```

Complejidad:

$$T(n) = C1 + C2 + C3*n + C4*n + C5*n + C6*n*(n-1) + C7*n*(n-1) + C8$$

$$T(n) = C + C4*n + C5*n + C6*n*(n-1) + C7*n*(n-1)$$

$$T(n) = C + C(n) + C(n)(n-1)$$

$$T(n) = C + C(n) + Cn^2 - C(n)$$

$$T(n) \text{ es } O(n^2)$$

```
iii. public int[] fix45(int[] nums) {  
    int temp = 0; //C1  
    for (int i = 0; i < nums.length; i++) { //C2 + C3*n  
        if (nums[i] == 5) { //C4*n  
            for (int j = 0; j < nums.length; j++) { //C5*n + C6*n*n  
                if (nums[j] == 4) { //C7*n*n  
                    if (j + 1 < nums.length) {  
                        temp = nums[j + 1];  
                    }  
                    if (temp != 5) {  
                        temp = nums[j + 1];  
                        nums[j + 1] = nums[i];  
                        nums[i] = temp;  
                        break;  
                    }  
                }  
            }  
        }  
    }  
}
```

DOCENTE MAURICIO TORO BERMÚDEZ

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627

Correo: mtorobe@eafit.edu.co

```
    }  
  }  
}  
return nums;           //C8  
}
```

Complejidad:

$$T(n) = C1 + C2 + C3*n + C4*n + C5*n + C6*n*n + C7*n*n + C8$$

$$T(n) = C + C3*n + C4*n + C5*n + C6*n*n + C7*n*n$$

$$T(n) = C + C*n + C*n*n$$

$$T(n) = C + C*n + Cn^2$$

$$T(n) \text{ es } O(n^2)$$

```
iv. public boolean canBalance(int[] nums) {  
    for (int cS = 0; cS < nums.length; cS++) { //C1 * C2*n  
        int sum1 = 0;                          //C3*n  
        int sum2 = 0;                          //C4*n  
        for (int i = 0; i < cS; i++) {          //C5*n + C6*n*m  
            sum1 = sum1 + nums[i];  
        }  
        for (int j = cS; j < nums.length; j++) { //C7*n + C8*n*k  
            sum2 = sum2 + nums[j];  
        }  
        if (sum1 == sum2) {                    //C9*n  
            return true;  
        }  
    }  
    return false;                             //C10  
}
```

Complejidad:

$$T(n) = C1 + C2*n + C3*n + C4*n + C5*n + C6*n*m + C7*n + C8*n*k + C9*n + C10$$

$$T(n) = C + C*n + C6*n*m + C8*n*k$$

Supongamos que $m = n/2$ y $k = n/2$, luego

$$T(n) = C + C*n + C*n*(n/2)$$

$$T(n) = C + C*n + C(n^2)/2$$

$$T(n) \text{ es } O(n^2)$$

DOCENTE MAURICIO TORO BERMÚDEZ

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627

Correo: mtorobe@eafit.edu.co

```
v. public boolean linearIn(int[] outer, int[] inner) {  
    int contv = 0; //C1  
    for (int i = 0; i < inner.length; i++) { //C2 + C3*n  
        for (int j = 0; j < outer.length; j++) { //C4*n + C5*n*m  
            if (outer[j] == inner[i]) { //C6*n*m  
                contv++;  
                break;  
            }  
        }  
    }  
    return contv == inner.length; //C7  
}
```

Complejidad:

$$T(n) = C1 + C2 + C3*n + C4*n + C5*n*m + C6*n*m + C7$$

$$T(n) = C + C3*n + C4*n + C5*n*m + C6*n*m$$

$$T(n) = C + C*n + C*n*m$$

$$T(n) = C + C*n + C*n*m$$

$$T(n) \text{ es } O(n*m)$$

3.9) Expliquen con sus palabras las variables (qué es 'n', qué es 'm', etc.) del cálculo de complejidad del numeral anterior.

En este caso el tamaño del problema estaba dado por la variable n, que representa el tamaño del arreglo y por múltiples constantes C, que representaban operaciones que eran constantes. En algunos ejercicios, el tamaño n variaba, ya que por ejemplo en algunos no se tomaba en cuenta el primer elemento. Esto se puede ver en el ejercicio can balance en el que se usaron dos variables adicionales, m y k, las cuales hacían referencias también al tamaño del arreglo, solo que no era tamaño completo, si no que iba variando el elemento desde el cual empezaba el ciclo for al cual hacía referencia esta variable, sin embargo, en el peor de los casos, el for tenía que completarse por lo que $m = k = n$. En general estas variables hacían referencia al tamaño del arreglo como límite para una iteración y se modificaba este límite según se necesitaba.

4) Simulacro de Parcial

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

6. Trabajo en Equipo y Progreso Gradual (Opcional)

a) Actas de reunión

Integrante	Fecha	Hecho	Haciendo	Por hacer
Daniel Mesa	09/09/2017	Estructura de git	Punto 2.1	Complejidad
Kevin Parra	09/09/2017	Punto de 2.2 y tres ejercicios de array 2	Completando código	Punto 3
Daniel Mesa	09/09/2017	Dos puntos de array dos	graficar y tomar tiempos	parcial
Kevin Parra	09/09/2017	Punto del parcial	Explicación del maxSpan	Ayudar en el punto 3
Daniel Mesa	10/09/2017	puntos 3.3, 3.4, 3.5	organizando guía de informe	Preguntar al profesor sobre gráfica y complejidad de mergeSort

b) El reporte de cambios en el código

```
commit 5aa46b6a55062a2975a6fba4304d38b023d97ffa (HEAD -> master, origin/master,
origin/HEAD)
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date: Mon Sep 11 12:04:03 2017 -0500

Add files via upload

commit 8f359d12dc69178047b5cc2a5fb40233546983da
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date: Mon Sep 11 10:56:39 2017 -0500

correccion

commit 4a9cee16318ef1ca7a0cbab48db3800c106c8b86
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date: Mon Sep 11 10:55:41 2017 -0500

subire una correccion

commit 93b50f46a1b37d68ac93e99e8d6balecccaa3e9d
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date: Sun Sep 10 19:26:18 2017 -0500

Borrador del informe codigos terminados, documentacion casi completa

commit 0db7099d81a8dc7f699e07c2a1ddf3d33db92416
commit 5aa46b6a55062a2975a6fba4304d38b023d97ffa (HEAD -> master, origin/master,
origin/HEAD)
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date: Mon Sep 11 12:04:03 2017 -0500

Add files via upload

commit 8f359d12dc69178047b5cc2a5fb40233546983da
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date: Mon Sep 11 10:56:39 2017 -0500

correccion

commit 4a9cee16318ef1ca7a0cbab48db3800c106c8b86
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date: Mon Sep 11 10:55:41 2017 -0500

subire una correccion

commit 93b50f46a1b37d68ac93e99e8d6balecccaa3e9d
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date: Sun Sep 10 19:26:18 2017 -0500

Borrador del informe codigos terminados, documentacion casi completa

commit 0db7099d81a8dc7f699e07c2a1ddf3d33db92416
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date: Sat Sep 9 18:32:02 2017 -0500

Ejercicios en linea terminados


commit 06831ab90a46acb7ccf7d966dc525c1396a2089d
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date: Sat Sep 9 17:39:45 2017 -0500

Listo array 3, faltan los 2 de array 2

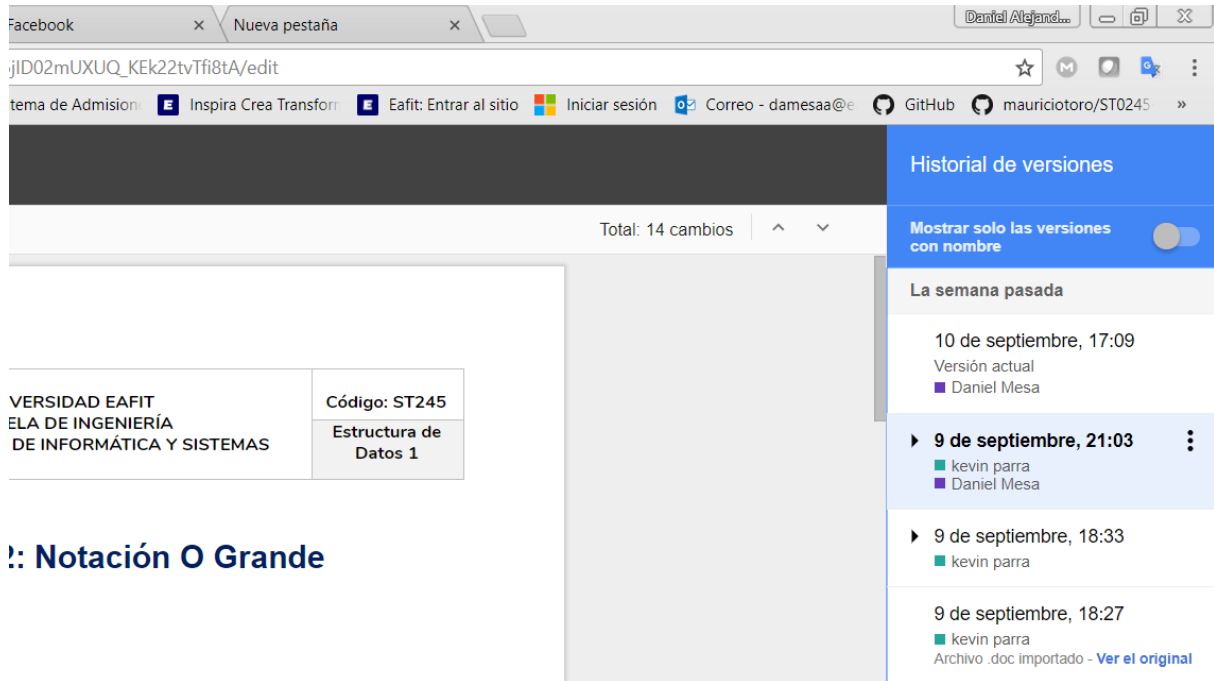
commit d3ec2edf13d6f9ebde13235d42de54caed2fb7d0
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date: Sat Sep 9 15:35:54 2017 -0500

Subiendo estructura
```

Nota: se adjunta también como PDF.

	UNIVERSIDAD EAFIT ESCUELA DE INGENIERÍA DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS	Código: ST245
		Estructura de Datos 1

c) El reporte de cambios del informe de laboratorio



UNIVERSIDAD EAFIT ESCUELA DE INGENIERÍA DE INFORMÁTICA Y SISTEMAS	Código: ST245 Estructura de Datos 1
---	--

! Notación O Grande

Historial de versiones

Mostrar solo las versiones con nombre ☐

La semana pasada

- 10 de septiembre, 17:09
Versión actual
Daniel Mesa
- 9 de septiembre, 21:03
kevin parra
Daniel Mesa
- 9 de septiembre, 18:33
kevin parra
- 9 de septiembre, 18:27
kevin parra
Archivo .doc importado - [Ver el original](#)