Laboratorio Nro. 1: Recursividad

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

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

2.3) Expliquen con sus propias palabras como funciona el ejercicio GroupSum5.

Explicación.

En este ejercicio se pide saber si sumando un conjunto de números se puede obtener un numero entero target. Además, se nos dice que todos los números múltiplos de 5 deben ser considerados para la suma. Para solucionar el ejercicio usamos recursividad, tomando una posición star inicial que será 0, y aumentando en cada llamado recursivo su valor de uno en uno, tenemos una condición de parada en la que si el start alcanza el número de elementos del conjunto (el cual será una arreglo de entero) retorna si efectivamente se pudo obtener el número, es decir, si el target final es cero. En este ejercicio se hacen en realidad dos llamados recursivos, uno "tomado" el numero en la posición del start y el otro sin tomarlo, es decir restando ese valor al target y mirando si se puede llegar al cero en algún punto de los llamados recursivos, ya que para verificar si dos números a + b = c, podemos pasar a y b a restar y si al lado izquierdo da cero, efectivamente la suma de a + b es igual a c, algo así se aplicó en este ejercicio. Por último, dada la restricción sobre los números múltiplos de 5, hicimos un método contM5 que suma todos los números múltiplos de cinco para luego restarlos al target. El ejercicio también nos decía que, si después de un múltiplo de cinco había un 1, este no se tomaba en cuenta, por lo que en el método contM5 se verificaba si esto sucedía y se ponía un 0 en lugar del 1 que estaba en esa posición.

2.4) Calculen la complejidad de los ejercicios en línea de los numerales 2.1 y 2.2.

- **Recursión 1**

```
public int factorial(int n) {
    if (n == 0) {                // C
        return 1;                //C
    }
    return n * factorial(n - 1);  // C + T(n-1)
}
```

T(n) = C + C + C + T(n-1)
T(n) = C + T(n-1)
T(n) = C*n + C1
T(n) es O(C*n + c1)
es O(C*n) RS
es O(n) RP

```
public int bunnyEars(int bunnies) {
    if (bunnies == 0) {            //c
        return 0;                  //c
    }
    return 2 + bunnyEars(bunnies - 1);     //c + T(n-1)
}
```

T(n) = c + T(n-1)
T(n) = c n + c1
T(n) es O(cn + c1)
es O(cn) RS
es O(n) RP

```
public int fibonacci (int n) {
    if (n == 0 || n == 1) {          //c + c
        return n;                    // c
    } else {
        return fibonacci(n - 1) + fibonacci(n - 2);     // T(n-1) + T(n-2)
    }
}
```

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

T(n) = c + T(n-1) + T(n-2)
T(n) = c ((2^n) - 1) + ci 2^(n-1)
T(n) = c(2^n) - c + (ci2^n) / 2
T(n) es O(c(2^n) - c + (ci2^n) / 2)
es O(c(2^n) - c) RS
es O(c(2^n)) RS
es O(2^n) RP

```
public int bunnyEars2(int bunnies) {
    if (bunnies == 0) { //c
        return 0;           //c
    }
    if (bunnies % 2 == 0) {      //c + c
        return bunnyEars2(bunnies - 1) + 3;          // T(n-1) + 3
    }
    return bunnyEars2(bunnies - 1) + 2;         // T(n-1) + 2
}
```

T(n) = c + T(n-1) + 3
T(n) = (c+3)n + c1
T(n) es O((c+3)n + c1)
es O((c+3)n) RS
es O(n) RP

```
public int triangle(int rows) {
    if (rows == 0) {//c
        return 0;           //c
    }

    if (rows == 1) {//c
        return 1;           //c
    }
    return triangle(rows - 1) + rows;    //T(n-1) + c
}
```

T(n) = T(n-1) + c
T(n) = cn + c1
T(n) es O(cn + c1)
es O(cn) RS

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

es O(n) RP

```
public int sumDigits(int n) {
    if (n / 10 == 0) {          //c + c
        return n;               //c
    }
    return sumDigits(n / 10) + n % 10; //T(n / 10) + c
}
```

T(n) = c + T(n/10)
T(n) = (clog(n))/log(10)+c1
T(n) = (clog(n)) + c1
T(n) es O(clog(n) + c1)
T(n) es O(clog(n)) RS
T(n) es O(log(n)) RP

- **Recursión 2**

Ejercicio 1:

```
public boolean groupSum6(int start, int[] nums, int target) {
    if (start >= nums.length) {         //C1
        return (target == 0);
    }

    int diff = target - (6 * count6(start, nums));          //C2 + C3 + m
    if (groupSum(start, nums, diff)) {              //C4 + 2**n
        return true;
    }
    return false;

}
```

Complejidad:
T(n) = C1 + C2 +C3 + m + 2**n
T(n) = C + n + 2**n
luego T(n) es O(2**n)

```
public int count6(int start, int[] nums) {
    int count = 0;          //C1
```

```
        if (nums[start] == 6) {        //C2
            count++;
            nums[start] = 0;
        }
        if (start + 1 < nums.length) {      //C3
            count += count6(start + 1, nums);     //C4 + T(n-1)
        }
        return count;      //C5
    }
```

Complejidad
T(n) = C1+ C2 + C3 + C4 + T(n-1) + C5
T(n) = C + T(n-1)
Luego T(n) es O(n)

Ejercicio 2:

```
    public boolean groupNoAdj(int start, int[] nums, int target) {
        if (start >= nums.length) {       //C1
            return target == 0;           //C2
        }
        return groupNoAdj(start + 1, nums, target) || groupNoAdj(start + 2, nums, target
- nums[start]);       //2[T(n-1)]
    }
```

Complejidad
T(n) = C1+2[T(n-1)]
luego T(n) es O(n)

Ejercicio 3:

```
    public boolean groupSum5(int start, int[] nums, int target) {
        if (start >= nums.length) {                  //C1
            return (target == 0);
        }

        int cin = target - (countM5(start, nums));    //C2 + n
        if (groupSum(start, nums, cin)) {           // C3 + 2**n
            return true;
        }
        return false;
```

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

```
    }

     Complejidad:
     T(n) = C1 + C2 + C3 + n + 2**n
     T(n) = C + n + 2**n
     luego T(n) es O(2**n)


    public int countM5(int start, int[] nums) {
        int count = 0;                          //C1
        if (nums[start]%5 == 0) {               //C2
            count+=nums[start];
            nums[start] = 0;
            if (start + 1 < nums.length) {
                if (nums[start+1] == 1) {
                    nums[start+1] = 0;
                }

            }
        }
        if (start + 1 < nums.length) {          //C3
            count += countM5(start + 1, nums);    //C4 + T(n-1)
        }
        return count;
    }
Complejidad:
     T(n) = C1 + C2 + C3 + C4 + T(n-1)
     T(n) = C + T(n-1)
     luego T(n) es O(n)

    Ejercicio 4
    public boolean groupSumClump(int start, int[] nums, int target) {
        if (start >= nums.length) {             //C1
            return (target == 0);
        }
        tomarRep(start, nums, target);          //C2 + n
        System.out.println(target);

        if (groupSum(start, nums, target)) {    //C3+ 2**n
            return true;
```

```
    }
    return false;

}
```

Complejidad:
T(n) = C1+ C2 + n + C3 + 2**n
T(n) = C + n + 2**n
luego T(n) es O(2**n)

```
public int tomarRep(int start, int[] nums, int target) {
    int cont = 0;                                   //C1
    boolean hayR = false;                           //C2
    for (int i = 0; i < nums.length; i++) {         //C3 + C*n
        if (i + 1 < nums.length) {                  //C4 * n
            if (nums[i + 1] == nums[i]) {
                cont += nums[i];
                nums[i] = 0;
                hayR = true;
                System.out.println("tomando: " + nums[i]);
            } else if (hayR) {
                nums[i] = cont + nums[i];
                System.out.println("Contador: " + cont);
                hayR = false;
                cont = 0;
                System.out.println("tomando: " + nums[i]);
            }
        }
    }
    return cont;
}
```

Complejidad:
T(n) = C1 + C2 + C3 * n + C4*n
T(n) = C*n
luego T(n) es O(n)

Ejercicio 5:

```
public boolean splitArray(int[] nums) {
if(nums.length == 1)                    //C1
  return false;
```

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

```
  if(nums.length == 0)                    //C2
    return true;
  int total = totalSum(nums , nums.length);       //C3+n
  if(total % 2 == 0)                      //C4
  {
    if(groupSum(0, nums.length, nums, total/2))       //C5+2**n
       return true;
  }
  return false;

  }

  Complejidad:
  T(n) = C2 + C3 + C5 + n + 2**n
  T(n) = C + n + 2**n
  T(n) es O(2**n)

  public int totalSum(int [] nums, int n)
  {
  if (n == 1)       //C1
     return nums[0];
  else
     return totalSum(nums, n-1) + nums[n-1];          //C2 * T(n-1)
  }
Complejidad:
    T(n) es O(n)
public boolean groupSum(int start, int[] nums, int target) {
      if (start >= nums.length) {//C1
         return (target == 0);//C2
      }
      if (groupSum(start + 1, nums, target - nums[start])) {//C3+T(n-1)
         return true;//C4
      }
      if (groupSum(start + 1, nums, target)) {//C5+T(n-1)
         return true;//C6
      }
      return false;
  }

   Complejidad
   T(n) = C1+C3+C5+2[T(n-1)]
```

T(n) = C+2[T(n-1)]
luego T(n) es O(2**n)

```
public boolean groupSum(int start, int end, int[] nums, int target) {
if (start >= end)  //C1
 if(target == 0)
    return true;
 else return false;
if (groupSum(start + 1, end, nums, target - nums[start])) return true;
  //C2+T(n-1)
if (groupSum(start + 1, end, nums, target)) return true;  //C3+T(n-1)
return false;

}
```

Complejidad:
T(n) es O(2**n)

2.5) Expliquen con sus palabras las variables (qué es 'n', qué es 'm', etc.) del cálculo de complejidad del ejercicio 2.4

En estos ejercicios solo se utilizó "n" para referirnos al tamaño del problema, no vimos la necesidad de otra variable ya que no presentamos ejercicios como recorrer una matriz o algo parecido que involucran otra variable más, entonces "n" se resume en el tamaño del problema u operaciones, también utilizamos "c" haciendo referencia a constante (número cualquiera).

3) Simulacro de preguntas de sustentación de Proyectos

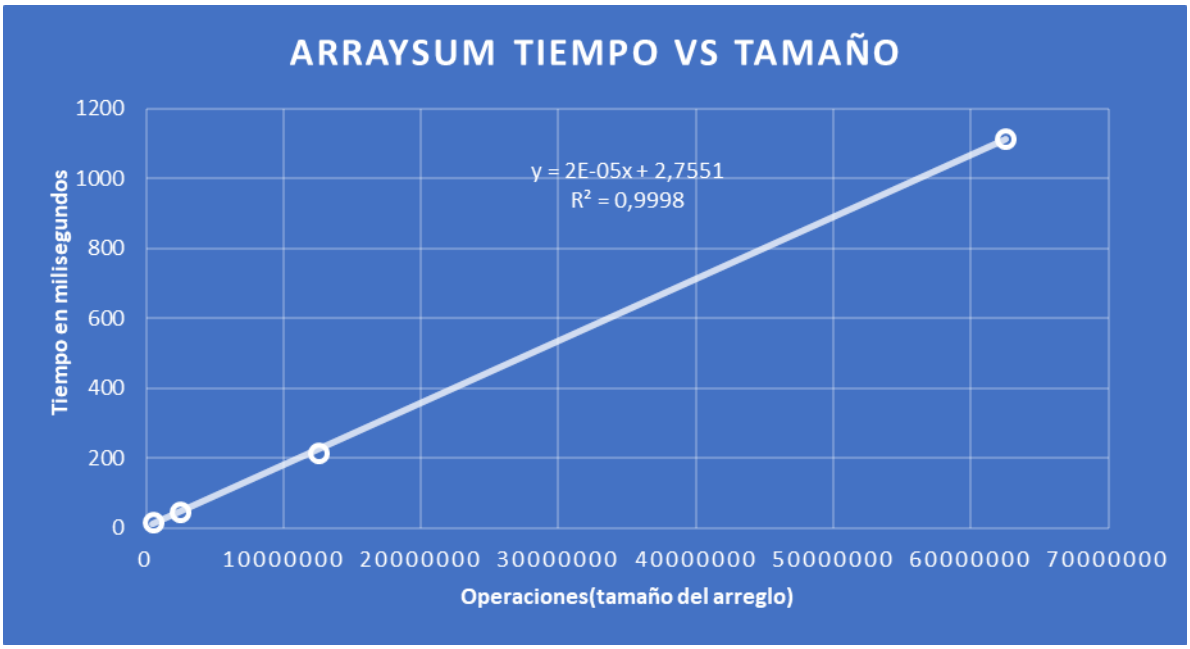
- Resueltas en los anteriores ejercicios

3.1) Tabla con tiempos en milisegundos:

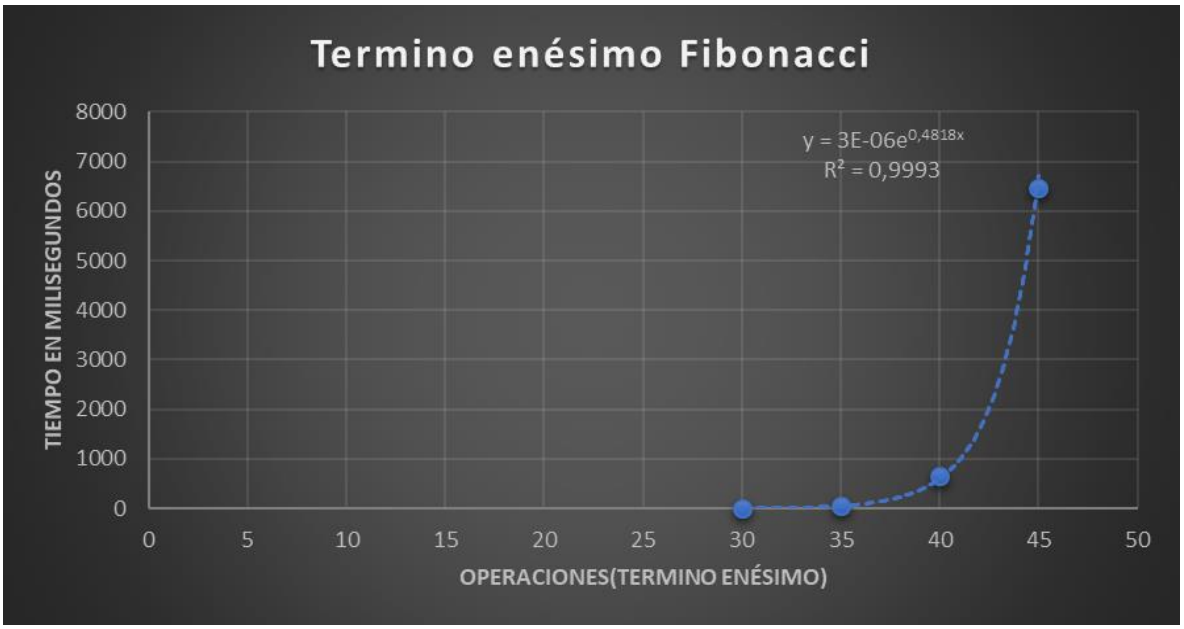
	N = 30	N = 35	N = 40	N = 45
R Fibonnacci (Tiempo en Milisegundos)	5	50	668	6472

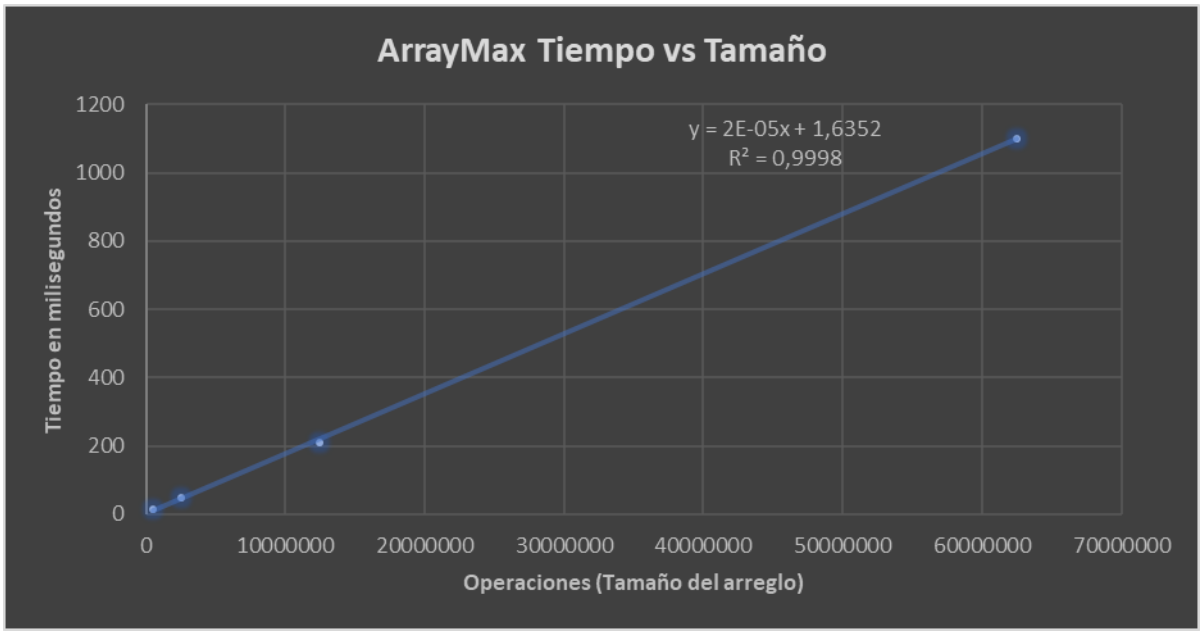
	N = 500.000	N = 2’500.000	N = 12’500.000	N = 62'500.000
R Array sum (Tiempo en Milisegundos)	18	49	215	1116
R Array máximum (Tiempo en Milisegundos)	16	48	211	1100

3.2) Grafica de los tiempos que tomó en ejecutarse Array Sum, Array Máximum y Fibonacci recursivo:



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





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

3.3) Conclusión respecto los tiempos obtenidos con la teoría.

Fueron acertados, los tiempos coincidieron con el tipo de complejidad de cada algoritmo, tanto ArraySum como ArrayMax eran lineales por lo cual era menos tiempo de ejecución y los restos arrojados fueron satisfactorios, en el caso de Fibonacci, su complejidad era mucho mayor tomando mayores tiempos de ejecución para valores más grandes y lo comprobamos al ver como se ajustaba la gráfica a una exponencial y que crecía en grandes cantidades conforme el número de operaciones aumenta.

3.4) Stack Overflow ¿Qué aprendimos?

Para manejar recursión es necesario tener cuidado con la condición de para porque esta puede desbordar la memoria en la que los llamados recursivos (la pila) se dan es decir algo así, como un ciclo infinito, aunque no solo eso lo puede causar, probablemente la condición de para esta bien pero solo es funcional para evaluar valores grandes de lo contrario podría ser un ciclo infinito y finalmente, puede que todo este perfecto pero aun así se necesita hacer muchos llamados recursivos como es el caso de ArraySum para valores de millones esto llena el stack, por lo cual hay que asignarle más memoria de la que java normalmente puede reservar.

3.5) Cuál es el valor más grande qué pudo calcular para Fibonnacci? ¿Por qué? ¿Por qué no se puede ejecutar Fibonacci con 1 millón?

El valor más grande que se pudo calcular para Fibonacci fue del termino 45 aproximadamente, ya que a partir de este empezaba a arrojar términos negativos por ser ya números muy grandes y solo se soporta 4 bits, además un tiempo de ejecución bastante grande, esto pasa porque Fibonacci es de complejidad 2^n por lo cual realiza muchos sub-problemas y podemos añadirle que los calcula repetidamente siendo ineficiente y tomando más tiempo.

3.6) ¿Cómo se puede hacer para calcular el Fibonacci de valores grandes?

Para eliminar el problema del tamaño de los números se puede utilizar el tipo de dato long, aunque probablemente se pueda volver a encontrar el problema cuando los números vuelvan a crecer demasiado y finalmente para minimizar el problema de tiempo de ejecución se puede eliminar calcular repetitivamente los sub-problemas utilizando programación dinámica, es decir una vez calculado un sub-problema este se guarda para ser utilizado de nuevo si se necesita y asi no se tiene que volver a calcular.

3.7) ¿Qué concluyen sobre la complejidad de los problemas de *CodingBat Recursión 1* con respecto a los de *Recursión 2*?

La complejidad en este tipo de ejercicios de recursión dos, que requerían de más de un llamado recursivo y de emplear otras funciones para completarlos, era por lo general 2 a la n, ya que siempre había un método base en el que se debían hacer dos llamados recursivos, como ere el caso de groupSum, que se empleó para otros métodos y en este se hacían dos llamados recursivos, mientras que en recursión uno eran menos complejos en su mayoría eran O(n) no requerían de otros métodos complementarios y tampoco excepto Fibonacci requerían de dos llamados recursivos, Fibonacci igualmente poseía un O(2^n), y sumar dígitos es O(long n) siendo esta ultima una complejidad un poco normal y baja, podemos concluir entonces complejidades muy diferentes en recursión uno y dos, además la dificultad para resolver el problema en recursión dos era evidentemente mucho mayor.

4) Simulacro de Parcial

1. `(Start+1, nums, target)`
2. a
3. primera línea vacía: (n - a, a, b, c)
   segunda línea vacía: (res, solucionar(n - b, a, b, c) + 1)
   tercera línea vacía: (res, solucionar(n -c, a, b, c) + 1)
4. e

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

6. Trabajo en Equipo y Progreso Gradual (Opcional)

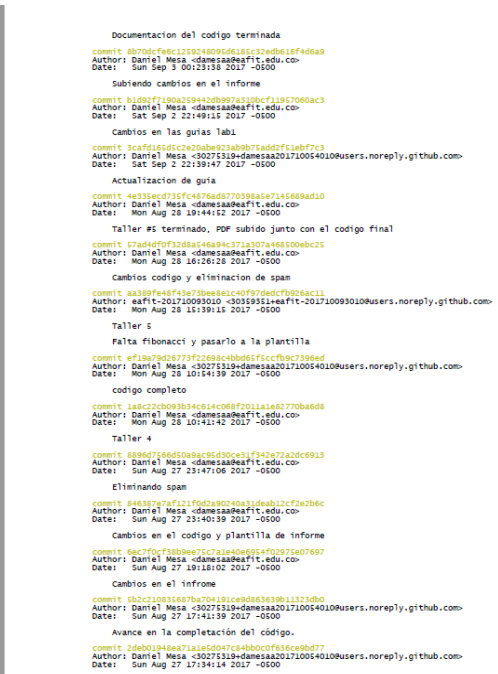
a) Actas de reunión

Integrante	Fecha	Hecho	Haciendo	Por hacer
Kevin Parra	19/08/2017	Creación de la tabla para las actas de trabajo	Documento en Google Docs para reporte de cambios del informe	Empezar Ejercicios del CodingBat
Daniel Mesa	27/08/2017	Numeral 1.1 de la guía del laboratorio.	Numeral 3.1 Guía del laboratorio.	tomar tiempos 3.1
Kevin Parra	29/08/2017	Numeral 2.1 Ejercicios en línea CodingBat	Numeral 2.2 Ejercicios en CodingBat	Complejidad Recursión 1 y 2
Daniel Mesa	3/09/2017	Numeral 3.1 y 3.2	Complejidad recursión 1	Complejidad recursión 2
Kevin Parra	3/09/2017	Complejidad Recursión 2	Numeral 2.2 Ejercicios en CodingBat Explicación de GroupSum	Subir el Laboratorio a github
Daniel Mesa	3/09/2017	numerales 2.5, 3.4,	Numerales 3.5, 3.6	Simulacro Parcial
Kevin Parra, Daniel Mesa	3/09/2017	Simulacro Parcial		Subir el Laboratorio a github
Daniel Mesa	3/09/2017	Revisar trabajo	Revisar el trabajo cada uno	Corroborar mis puntos Kevin
Kevin Parra	3/09/2017	Revisar trabajo	Revisar el trabajo de los compadres	Verificar puntos

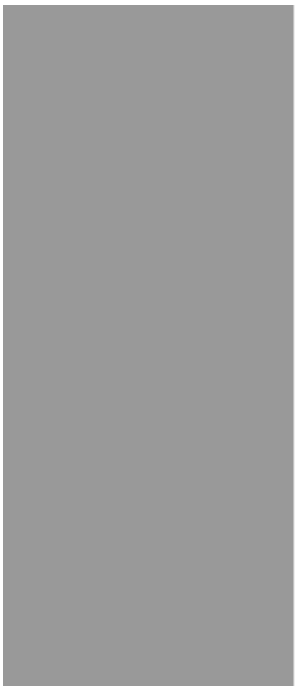
b) El reporte de cambios en el código

```
commit e618ca9849683df92be3f06733c02525b83e8191 (HEAD -> master, origin/master,
origin/HEAD)
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date:   Sun Sep 3 20:54:47 2017 -0500

    Cambios en la guia

commit 7ab9i764eec99e44d65a4ce2dce83dfde9f0a54ca
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com
>
Date:   Sun Sep 3 18:26:32 2017 -0500

    Add files via upload

    Listo lo de la explicacion de gruopSum y recursion 2

commit 1c27d6a07529fe2aa644ac08dddfbfbf75ba8f2
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com
>
Date:   Sun Sep 3 18:25:53 2017 -0500

    Lo borro por que soy malo JUAJAUAJu

:...skipping...
commit e618ca9849683df92be3f06733c02525b83e8191 (HEAD -> master, origin/master,
origin/HEAD)
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date:   Sun Sep 3 20:54:47 2017 -0500

    Cambios en la guia

commit 7ab9i764eec99e44d65a4ce2dce83dfde9f0a54ca
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date:   Sun Sep 3 18:26:32 2017 -0500

    Add files via upload

    Listo lo de la explicacion de gruopSum y recursion 2

commit 1c27d6a07529fe2aa644ac08dddfbfbf75ba8f2
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date:   Sun Sep 3 18:25:53 2017 -0500

    Lo borro por que soy malo JUAJAUAJu

commit 9650e70356130547fc730lcdeede321648ceb4b
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date:   Sun Sep 3 17:46:04 2017 -0500

    Update Recurison2.java

commit 8aa68e4dda8ab52215fd3421845c43987e0c87a7
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date:   Sun Sep 3 17:34:21 2017 -0500

    Update Recurison2.java

commit 2ce438055a844da31268009ebd718d61a10e881f8
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date:   Sun Sep 3 17:19:17 2017 -0500

    Update and rename Recurison2 to Recurison2.java

commit f0a3f7cbf9d2a35dc4c6d15ca1d8f15c9cc4a912
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date:   Sun Sep 3 15:49:20 2017 -0500

    Guia actualizada

commit d476f1614cbcd131cbe6b1c296ddlcfbf8765ce6
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date:   Sun Sep 3 14:45:51 2017 -0500

    Revisar la citación del código

    Tambien si gusta revise la complejidad

commit f09ef6c77a36ef7ad580702bde0e02163c0fe07c
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date:   Sun Sep 3 10:47:48 2017 -0500
```

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



```
    Documentacion del codigo terminada

commit 8b70dcfe6c1259248095d6d185c32edb616f4d6a9
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date:   Sun Sep 3 00:23:18 2017 -0500

    Subiendo cambios en el informe

commit b1d92f2190a259442db997a110bcf11957060ac3
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date:   Sat Sep 2 22:49:15 2017 -0500

    Cambios en las guias lab1

commit 3ca5d1656d5c2e20abe023ab9b75add2f51abf7c3
Author: Daniel Mesa <30275319+damesaa2017100540100@users.noreply.github.com>
Date:   Sat Sep 2 22:39:47 2017 -0500

    Actualizacion de guia

commit 4e335ecd735fc4876ad770398a5e7145689ad10
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date:   Mon Aug 28 19:44:52 2017 -0500

    Taller #5 terminado. PDF subido junto con el codigo final

commit 57a64df0f328da546e94c371a307a468100ebc25
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date:   Mon Aug 28 16:26:28 2017 -0500

    Cambios codigo y eliminacion de spam

commit aa389fe48f43e73bee8e1c40f97dedcfb926ac11
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date:   Mon Aug 28 15:39:15 2017 -0500

    Taller 5

    Falta fibonacci y pasarlo a la plantilla

commit ef19a79d26773f26998c4bbd65f5ccfb9c7396ed
Author: Daniel Mesa <30275319+damesaa2017100540100@users.noreply.github.com>
Date:   Mon Aug 28 10:54:39 2017 -0500

    codigo completo

commit 1a8c22cb093b34c614c068f2011a1e82770ba6d8
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date:   Mon Aug 28 10:41:42 2017 -0500

    Taller 4

commit 8896d7166d50a9ac9d30ce31f342e712a4c6913
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date:   Sun Aug 27 23:47:06 2017 -0500

    Eliminando spam

commit 846387e7af121f0d2a90240a1deab12cf2e2b6c
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date:   Sun Aug 27 23:40:39 2017 -0500

    Cambios en el codigo y plantilla de informe

commit 6ec7f0cf98b8ee75c7a1a40a981ef02971e07697
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date:   Sun Aug 27 19:18:02 2017 -0500

    Cambios en el infrome

commit 5b2c210835687ba704101ce9d863630b11323db0
Author: Daniel Mesa <30275319+damesaa2017100540100@users.noreply.github.com>
Date:   Sun Aug 27 17:41:39 2017 -0500

    Avance en la completación del código.

commit 2deb01948ea71a1a1d047c84bb0c0f636ce9bd77
Author: Daniel Mesa <30275319+damesaa2017100540100@users.noreply.github.com>
Date:   Sun Aug 27 17:34:14 2017 -0500

    Avance en el informe
```



```
commit d035a0a907e3ae382f0f233fb86ce8fd2950ac0d
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date:   Mon Aug 21 18:37:56 2017 -0500

    Eliminando archivo TestFile

commit 39319936aeb0ca9778f60e7961f389709acd1e76
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date:   Mon Aug 21 18:31:09 2017 -0500

    Moviendo TestFile

commit 0fd77347f9e7681d395a37b154af74cf09a68321
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date:   Mon Aug 21 18:26:38 2017 -0500

    Cambios

commit bb8f3432fb794a6cdb8f85e19dd4394e5a1150
Author: Daniel Mesa <damesaa@eafit.edu.co>
Date:   Mon Aug 21 18:22:01 2017 -0500

    Ingresando mi primer cambio

commit 3d2e8513b4b49d6ad65952f6561bec29e9b1665
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date:   Sat Aug 19 21:57:49 2017 -0500

    Avance

    Listo el punto 2.1

commit 5f1561d4d991816c1e33d30af1cbe2ccbaaef817
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date:   Sat Aug 19 21:06:20 2017 -0500

    avance

commit dd2fb91e6eae5f3abf722ca67817f3e0daefb475
Author: eafit-201710093010 <30359351+eafit-201710093010@users.noreply.github.com>
Date:   Sat Aug 19 12:11:54 2017 -0500

    First recursion exercise

    Starting with the lab 1.
```

Nota: Se adjunta en documento .PDF.

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