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

## Laboratorio Nro. 1: Recursividad

Carla Daniela  
Rendón Baliero

Sebastian Arboleda Botero


**Nombre completo de integrante 1**  
 Universidad Eafit  
 Medellín, Colombia

**Nombre completo de integrante 2**  
 Universidad Eafit  
 Medellín, Colombia

2)

### 2.3 Expliquen con sus propias palabras cómo funciona el ejercicio GroupSum5

**R/:** Para la resolución de este algoritmo se utiliza una función recursiva, que tenga como parámetros de entrada: Un Start que representará la cantidad de elementos del arreglo y que a su vez, de manera decreciente funcione como un contador, también se utiliza un arreglo y un target (que es el objetivo con el cual se desea verificar, si a través de una suma recursiva de los elementos del arreglo, es posible llegar a el), a diferencia del algoritmo groupsum que ya todos conocemos, este tiene una variante, y es que debe incluir todos los “5” que estén en el arreglo, menos los que estén seguidos de un “1”, para ello, en el “else” de la función se agregan dos variantes internas (un if con su respectivo else), en el “if” se coloca como condición a cumplir que el elemento del arreglo que se está tomando en cuenta sea múltiplo de 5 y además si el elemento que sigue es diferente de 1, si esto se cumple al start le restamos 1 y al target le restamos el Arreglo en la posición Start. Para el caso del “else” tomamos en consideración que el elemento actual del arreglo sea múltiplo de 5 y que el elemento siguiente del arreglo sea igual a 1, en caso de cumplirse esto, al start se le resta 2, para que salte la posición y no cuente el 5 en la suma recursiva, al target lo dejamos tal y como está.

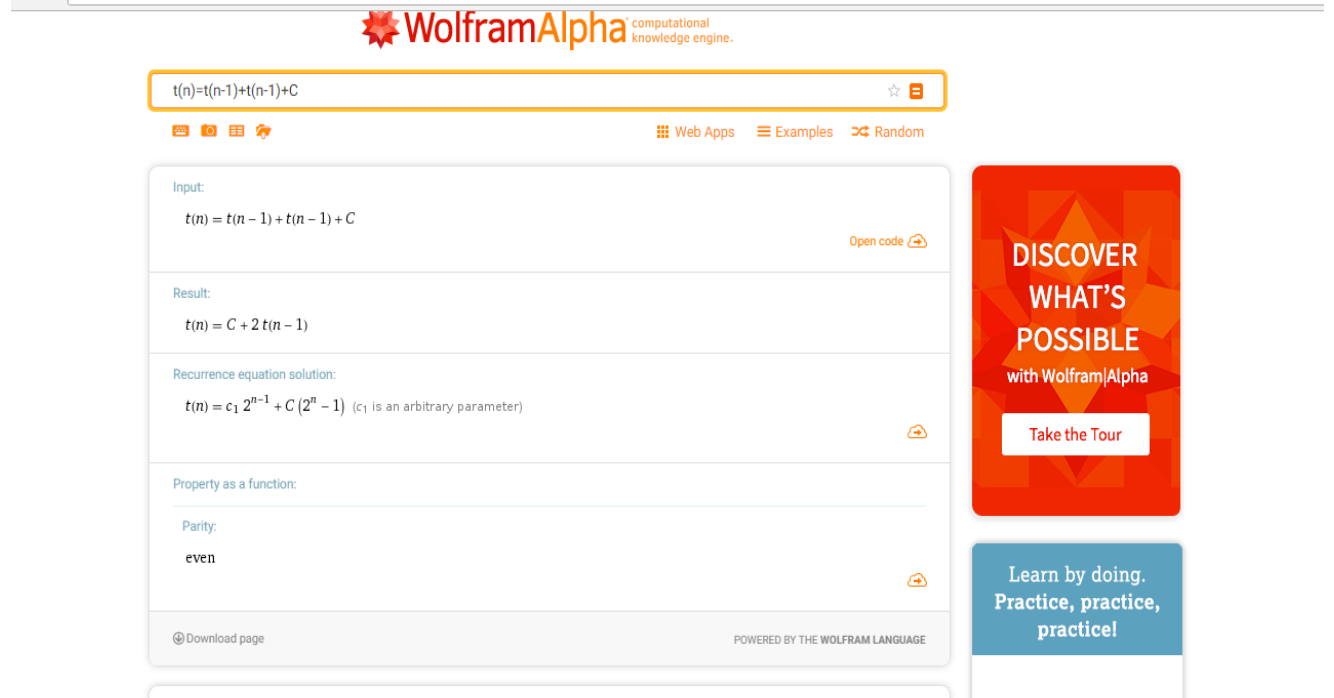
	<p style="text-align: center;"><b>UNIVERSIDAD EAFIT</b>  <b>ESCUELA DE INGENIERÍA</b>  <b>DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS</b></p>	<p><b>Código: ST245</b></p> <p><b>Estructura de Datos 1</b></p>
---	--	---

## 2.4 Cálculo de complejidad para los programas de recursividad.

Para el ejercicio de Fibonacci.

$$T_n = T(n-1) + T(n-2) + c1$$

→ [https://www.wolframalpha.com/input/?i=t\(n\)%3Dt\(n-1\)%2Bt\(n-1\)%2Bc1](https://www.wolframalpha.com/input/?i=t(n)%3Dt(n-1)%2Bt(n-1)%2Bc1)



WolframAlpha computational knowledge engine.

Input:  
 $t(n) = t(n-1) + t(n-1) + C$

Result:  
 $t(n) = C + 2 t(n-1)$

Recurrence equation solution:  
 $t(n) = c_1 2^{n-1} + C (2^n - 1)$  ( $c_1$  is an arbitrary parameter)

Property as a function:  
 Parity:  
 even

Download page

POWERED BY THE WOLFRAM LANGUAGE

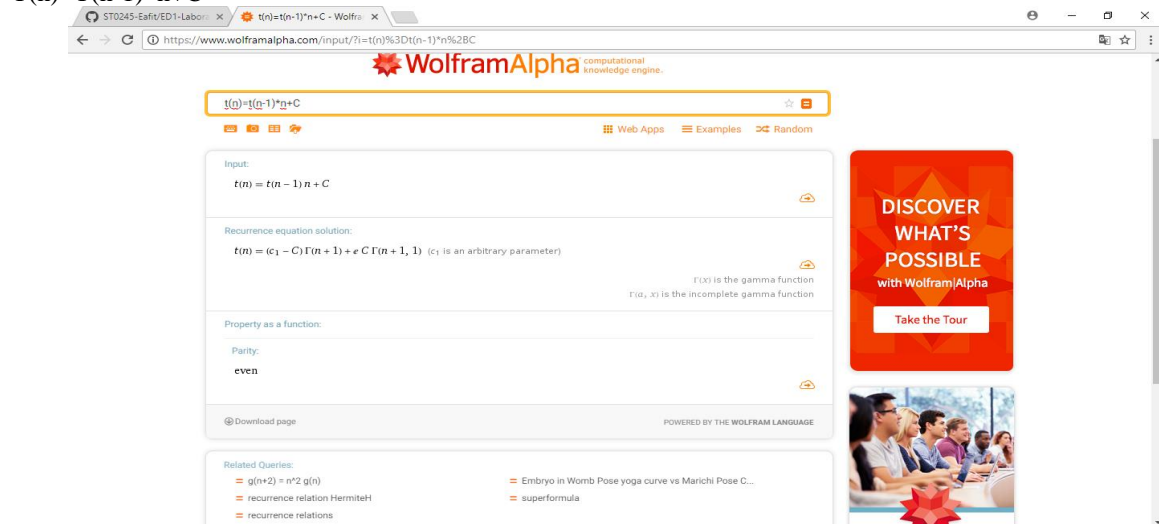
DISCOVER WHAT'S POSSIBLE with Wolfram|Alpha  
 Take the Tour

Learn by doing.  
 Practice, practice, practice!

Para este caso la complejidad del algoritmo sería de 2 exponente n.

Para el ejercicio de factorial.

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



WolframAlpha computational knowledge engine.

Input:  
 $t(n) = t(n-1) n + C$

Recurrence equation solution:  
 $t(n) = (c_1 - C) \Gamma(n+1) + e^C \Gamma(n+1, 1)$  ( $c_1$  is an arbitrary parameter)

Property as a function:  
 Parity:  
 even

Download page

POWERED BY THE WOLFRAM LANGUAGE

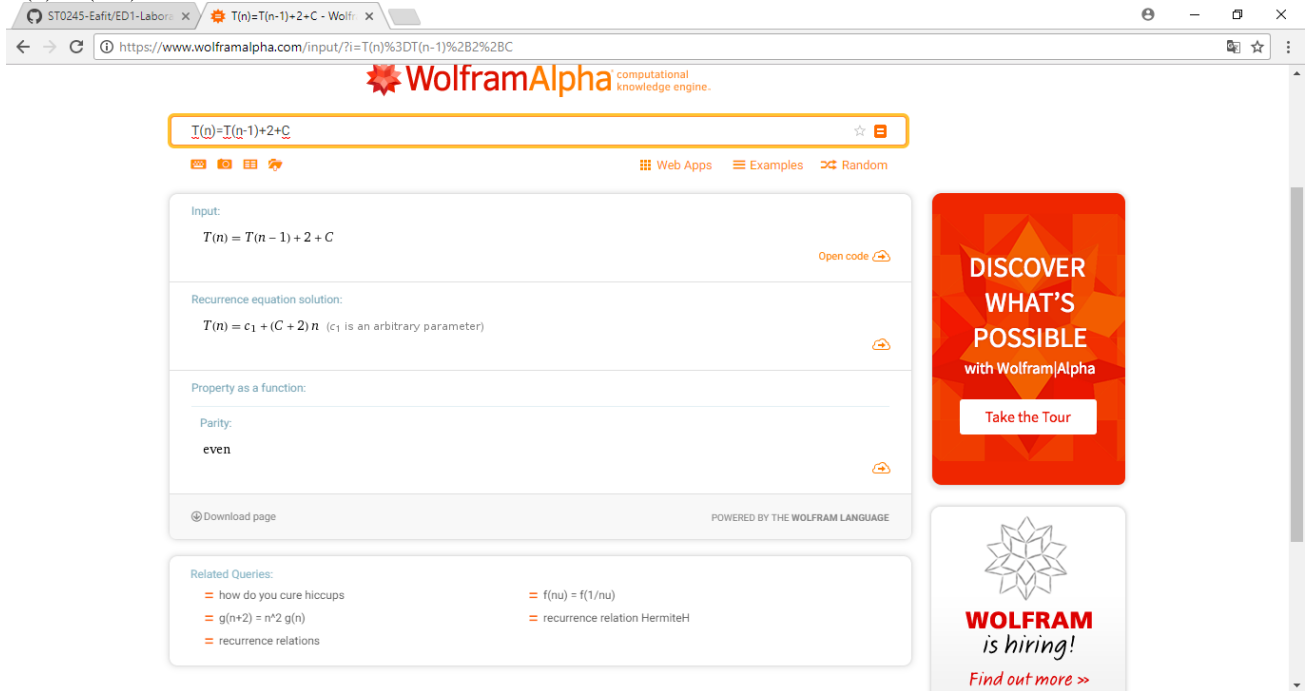
DISCOVER WHAT'S POSSIBLE with Wolfram|Alpha  
 Take the Tour

Related Queries:  
 -  $g(n+2) = n^2 g(n)$   
 - recurrence relation HermiteH  
 - recurrence relations  
 - Embryo in Womb Pose yoga curve vs Marichi Pose C...  
 - superformula

Para el ejercicio de los Conejos con dos orejas grandes.

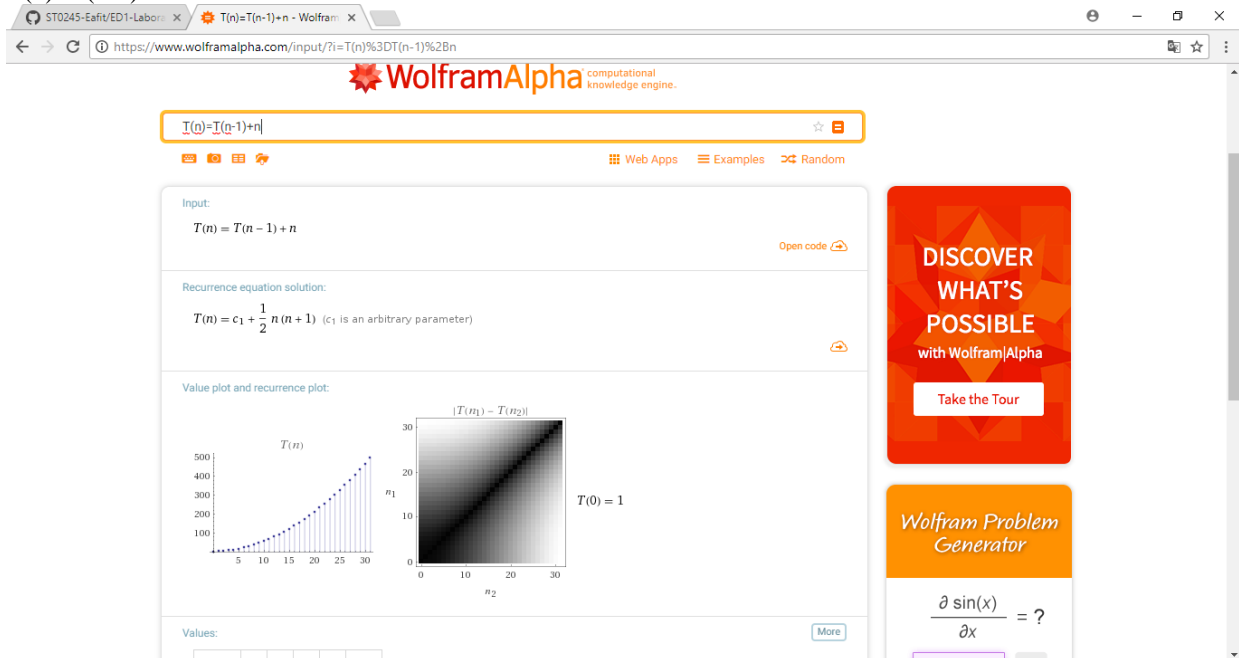
**DOCENTE MAURICIO TORO BERMÚDEZ**  
**Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627**  
**Correo: mtorobe@eafit.edu.co**

$$T(n) = T(n-1) + 2 + C$$




Para el ejercicio del Triangulo con Bloques:

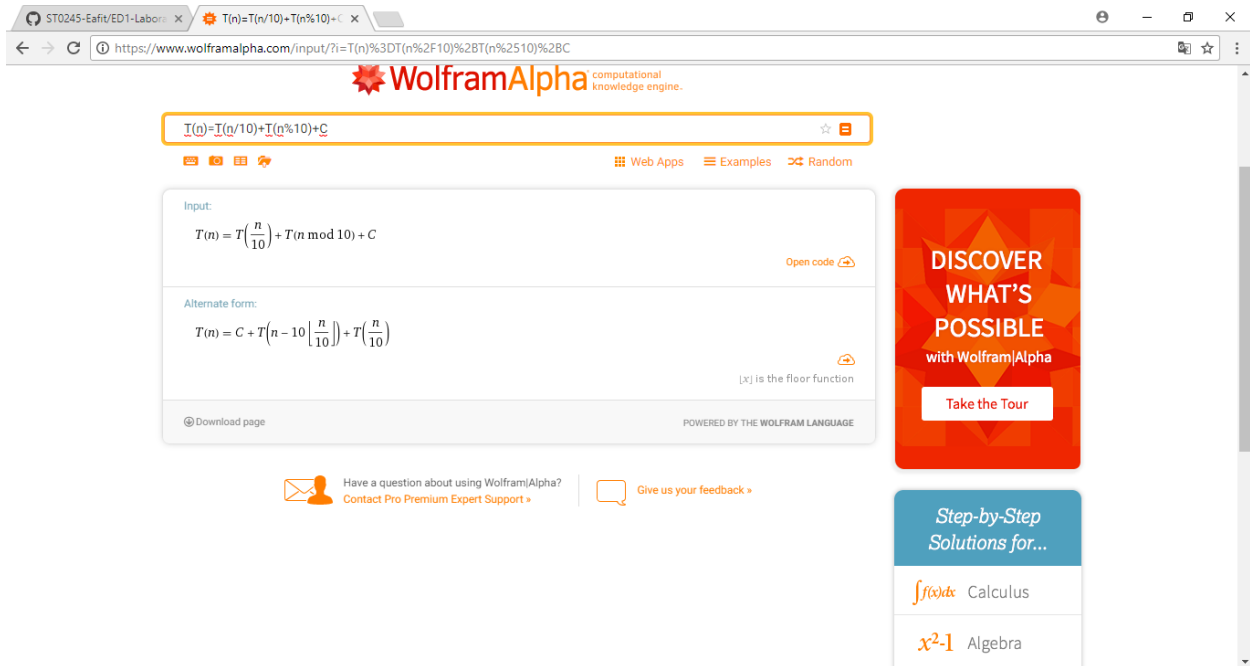
$$T(n) = T(n-1) + n + C$$



Para el ejercicio de Sumar dígitos:

$$T(n) = T(n/10) + T(n \% 10) + C$$

	<p style="text-align: center;"><b>UNIVERSIDAD EAFIT</b>  <b>ESCUELA DE INGENIERÍA</b>  <b>DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS</b></p>	<p><b>Código: ST245</b></p>
		<p><b>Estructura de Datos 1</b></p>



The screenshot shows the WolframAlpha interface with the input  $T(n) = T(n/10) + T(n \% 10) + C$ . The alternate form displayed is  $T(n) = C + T\left(n - 10 \left\lfloor \frac{n}{10} \right\rfloor\right) + T\left(\frac{n}{10}\right)$ . The page also includes a sidebar with 'Discover What's Possible with Wolfram|Alpha' and 'Step-by-Step Solutions for...' with links to Calculus and Algebra.

Para el ejercicio GroupSum6.

$$T(n) = T(n-1, m, \tilde{n} - m[n]) + C6$$

$$T(n, m, \tilde{n}) = c1 + c2 + c3 + c4 + c5 + c6 \\ T(n-1, m, \tilde{n} - m[n]) + C6$$

Para el ejercicio GroupSum5.

$$T(n) = T(n-1, m, \tilde{n} - m[n]) + C6$$

$$T(n, m, \tilde{n}) = c1 + c2 + c3 + c4 + c5 + c6 \\ T(n-1, m, \tilde{n} - m[n]) + C6$$

### Ejercicio 2.5 Explicación de variables usadas en los ejercicios.

Las variables  $m$ ,  $n$  y  $\tilde{n}$  representan los parámetros de entrada de las funciones recursivas, para el caso de los primeros 5 ejercicios vemos que solamente tienen un parámetro de entrada, por ello se utiliza la  $n$ , para representar dicho parámetro. Ya para los últimos ejercicios, debido a que estos son mas complejos, es necesario usar mas parámetros de entrada, en este caso  $m$ ,  $n$ ,  $\tilde{n}$  que representan el start, el arreglo y el objetivo o target respectivamente.

## 3) Simulacro de preguntas de sustentación de Proyectos

### 1. ¿Que aprendieron sobre Stack overflow?

R/:

Stack overflow o en español desbordamiento de pila, sucede cuando el Stack (el espacio donde se van almacenando las operaciones o rutinas) se llena generando que no se puedan realizar mas operaciones, ej: como si una pila de libros llegara hasta el techo y ya no hay forma de ubicar mas libros sobre dicha pila.

**Informacion extraida de:** <https://blog.makeitreal.camp/que-es-un-stack-overflow-desbordamiento-de-pila/>

**2. ¿Cuál es el valor más grande que pudo calcular para Fibonacci? ¿Por qué? ¿Por qué no se puede ejecutar Fibonacci con 1 millón?**

```
R/: "public static int fib (int n){  
    if(n<=0)return n;return fib(n-1)+fib(n-2);  
}"
```

Con este código el número más grande que se puede calcular fue 47, porque esta clase de operaciones tienen la forma  $2^n$  lo que quiere decir que crecen de una forma muy rápida, ocupando demasiado tiempo para realizar dicha operación, por eso un Fibonacci de 1 millón tendría la forma de  $2^{1'000.000}$  lo que tardaría demasiado tiempo incluso para una computadora.

**3. ¿Cómo se puede hacer para calcular el Fibonacci de valores grandes?**

```
R/: " public static int fibo(int n){  
    if(n==0)return n;  
    if(n<=2)return 1;  
    else{  
        int k1,k2;  
        k1 = fibo((n + 1) / 2);  
        k2 = fibo((n - 1) / 2);  
        return k1 * k1 + k2 * k2;  
    }  
}"
```

**Código sacado de :**

[https://foro.elhacker.net/programacion\\_cc/sucesion\\_de\\_fibonacci\\_rekursiva\\_optimizada-t395169.0.html](https://foro.elhacker.net/programacion_cc/sucesion_de_fibonacci_rekursiva_optimizada-t395169.0.html)

**4. ¿Qué concluyen sobre la complejidad de los problemas de CodingBat Recursion 1 con respecto a los de Recursion 2?**

R/: La principal diferencia es que en Recursion 1 no había que trabajar con arreglos, mientras que en Recursion 2 todos los problemas eran de arreglos, lo que genera un grado mayor de dificultad ya que había que hacer códigos que los recorrieran recursivamente y evaluaran ya fueran cual era mayor o su suma. Etc.

**4) Simulacro de Parcial**

1.  $start+1, nums, target$
2.  $t(n)=2.t(n/2)+Cn$
- 3.
4. La suma de los elementos del arreglo  $a$  es  $O(n)$
5. Línea 2 =  $n$ , Línea 3 =  $n-1$ , Línea 4 =  $n+1$ .  
5.2.  $T(n)=T(n-1)+T(n-2)+C$
6. Línea 10 = 0.