Laboratorio Nro. 1: Recursión

Andrés Sánchez Castrillón
Universidad Eafit
Medellín, Colombia
asanchezc@eafit.edu.co

Santiago Soto Marulanda
Universidad Eafit
Medellín, Colombia
ssotom@eafit.edu.co

Jamerson Correa Correa
Universidad Eafit
Medellín, Colombia
jscorreac@eafit.edu.co

2) Ejercicios en línea

2.3. GroupSum5: Este Método lo que hace es, que dado un arreglo de enteros y un entero; decir si es posible o no, que la suma de algunos o todos los elementos del arreglo sea igual a el entero dado, pero se tiene algunas restricciones, todos los múltiplos de 5 serán tenidos en cuenta en la suma a excepción de si el siguiente número del múltiplo de 5 es 1. Primero se define la condición de parada, se define las condiciones de las restricciones, luego se recorre el arreglo de forma recursiva.

2.4.
Recursion 1:

Fibonacci:
$T(n) = C + T(n-1) + T(n-2)$
$T(n) = C * 2^n + C$
$T(n)$ es $O(C * 2^n + C)$ //Aplicación notación O
$O(c * 2^n)$ //Regla de la suma
$O(2^n)$ //Regla del producto

Triangle:
$T(n) = C + T(n-1)$
$T(n) = C * n + C$
$T(n)$ es $O(C * n + C)$ //Aplicación notación O

O(C * n) //Regla de la suma
O(n) //Regla del producto

BunnyEars2:
T(n) = C + T(n - 1)
T(n) = C * n + C
T(n) es O(C * n + C) //Aplicación notación O
O(C * n) //Regla de la suma
O(n) //Regla del producto

noX:
T(n) = C + T(n - 1)
T(n) = C * n + C
T(n) es O(C * n + C) //Aplicación notación O
O(C * n) //Regla de la suma
O(n) //Regla del producto

PowerN:
T(n) = C + T(n-1)
T(n) = C * n + C
T(n) es O(C * n + C) //Aplicación notación O
O(C * n) //Regla de la suma
O(n) //Regla del producto

Recursion 2:

GroupSum:
T(n) = C + 2T(n-1)
T(n) = 2^(n - 1) + C (2^n - 1)
T(n) es O(2^(n - 1) + C (2^n - 1)) // Aplicación notación O
O(C (2^n - 1)) //Regla de la suma
O(2^n - 1) //Regla del producto
O(2^n) //Regla de la suma

GroupNoAdj:
T(n) = C + 2T(n-1)

T(n) = 2^(n - 1) + C (2^n - 1)
T(n) es O(2^(n - 1) + C (2^n - 1)) //Aplicación notación O
O(C (2^n - 1))   //Regla de la suma
O(2^n - 1)   //Regla del producto
O(2^n)   //Regla de la suma

GroupSumClump:
T(n) = C + C*n + 2T(n-1)
T(n) = 2^(n - 1) (8 C) - C (n + 3)
T(n) es O(2^(n - 1) (8 C) - C (n + 3)) //Aplicación notación O
O(2^(n - 1) (8 C))   //Regla suma
O(2^(n - 1))   //Regla producto

GroupSum5:
T(n) = C + T(n-1)
T(n) = C n
T(n) es O(Cn)   //Aplicación notación O
O(n)   //Regla producto

SplitArray:
T(n) = C + 2T(n-1)
T(n) = 2^(n - 1) + C (2^n - 1)
T(n) es O(2^(n - 1) + C (2^n - 1)) // Aplicación notación O
O(C (2^n - 1))   //Regla de la suma
O(2^n - 1)   //Regla del producto
O(2^n)   //Regla de la suma

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

2.5

Recursion 1:

Fibonacci: En este caso n representa al n-simo término de la sucesión de fibonacci.
Triangle: En este caso n representa la altura del triángulo a analizar.
BunnyEars2: En este caso n representa la cantidad de conejos en la granja.
noX: En este caso n representa la longitud de la cadena dada.
PowerN: En este caso n representa el exponente al cual se eleva la base.

Recursion 2:

GroupSum: En este caso n representa el tamaño del arreglo.
GroupNoAdj: En este caso n representa el tamaño del arreglo.
GroupSumClump: En este caso n representa el tamaño del arreglo.
GroupSum5: En este caso n representa el tamaño del arreglo.
SplitArray: En este caso n representa el tamaño del arreglo.

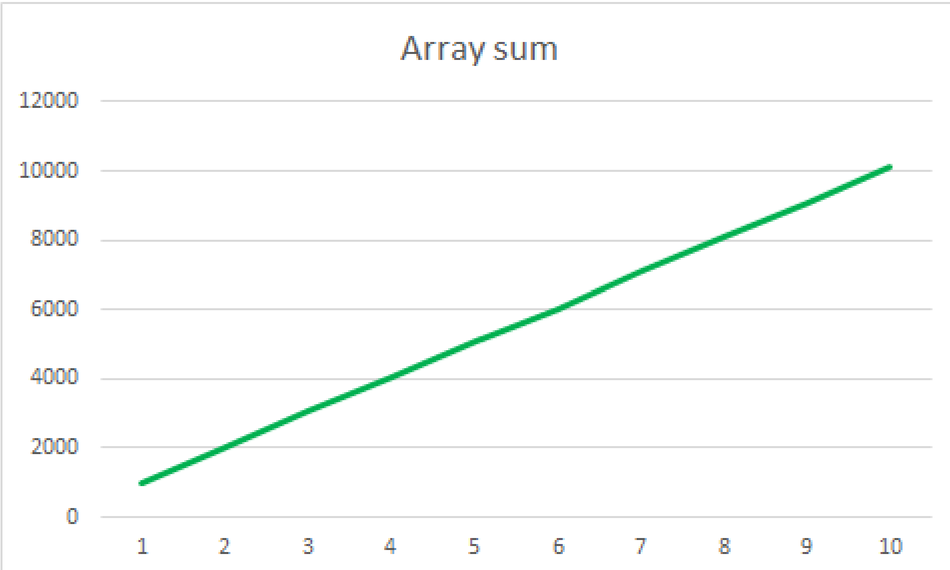
3) Simulacro de preguntas de sustentación de Proyectos

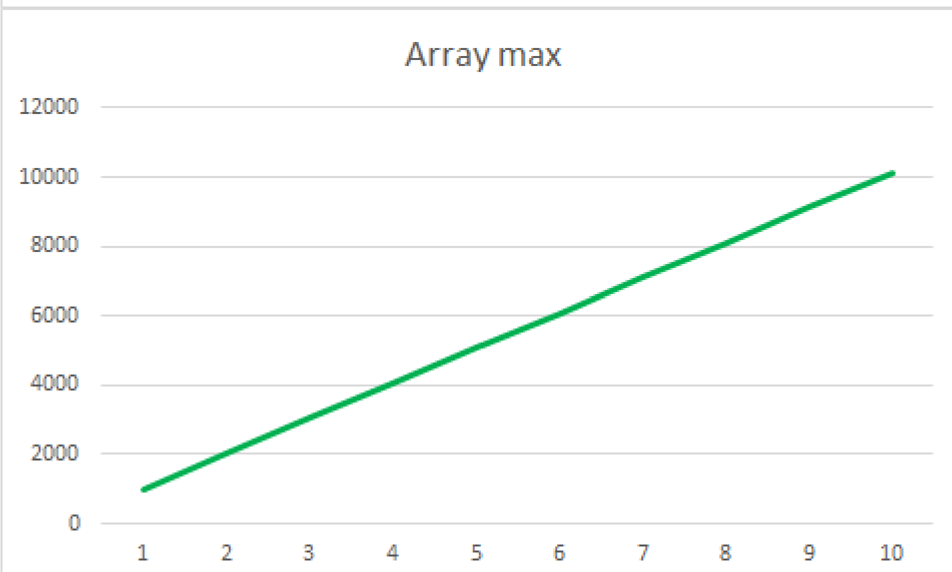
3.1

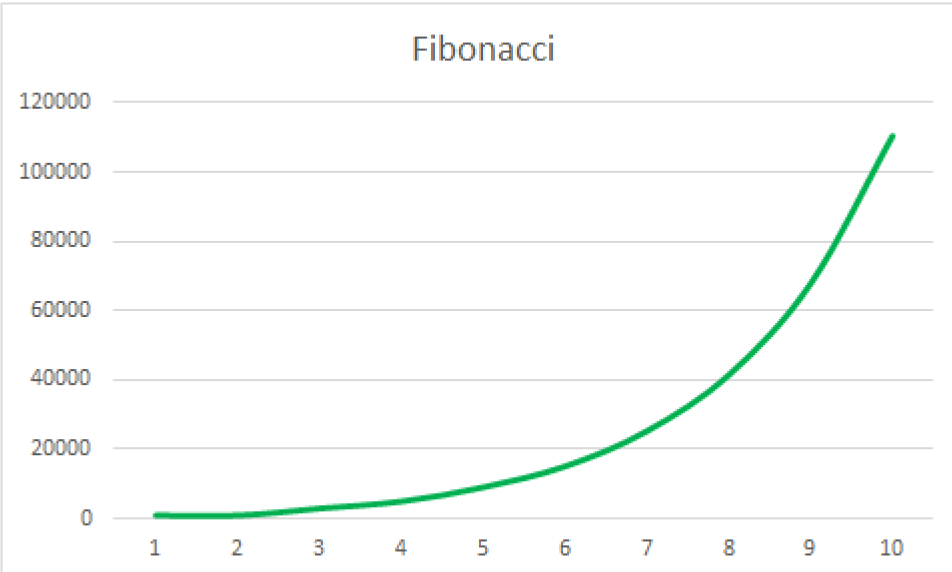
Nota: Estos tiempos fueron tomados usando Delay

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
R Array sum	1014	2028	3042	4031	5064	6022	7089	8085	9090	10126
R Array max	1014	2028	3042	4029	5070	6059	7098	8102	9109	10138
R Fibonacci	1014	1014	3028	5057	9116	15162	25302	4146 7	67761	110225

3.2


Array sum
12000
10000
8000
6000
4000
2000
0
1
2
3
4
5
6
7
8
9
10


Array max
12000
10000
8000
6000
4000
2000
0
1
2
3
4
5
6
7
8
9
10



3.3 Al tomar los datos experimentales respecto al tiempo de cada algoritmo, logramos observar que existe cierta cercanía a los datos teóricos, esta cercanía entre ambos datos depende de la máquina que estemos utilizando y la eficiencia del algoritmo implementado.

3.4 Stack overflow es un sitio web muy útil para los entusiastas de la programación, debido a que es una comunidad en donde diferentes programadores y desarrolladores de software pueden formular, encontrar y responder problemas reales sobre el área de programación. En pocas palabras la intencionalidad de la comunidad Stack Overflow es generar una biblioteca en donde exista un registro de respuestas detalladas para esta clase de problemas.

3.5 Nos fue posible calcularlo hasta el número 48 sin usar Delay, esto debido a que si el número n-simo que se espera obtener de la serie fibonacci (implementada con recursión) es muy grande, se deberán ejecutar y repetir una y otra vez gran cantidad de operaciones para obtener el número n-simo esperado. Por ejemplo si ejecutamos la serie fibonacci de un millón, ya que la complejidad de fibonacci es O(2^n), se realizarían infinidad de operaciones para obtener este valor.

3.6 Para implementar la serie de fibonacci para números grandes es necesario hacer uso de la programación dinámica. Ésta es utilizada cuando en un programa recursivo se deben resolver sub-problemas una y otra vez. su objetivo principal es identificar si existen sub-problemas en un problema recursivo que se estén resolviendo varias veces y así almacenar sus resultados para usarlos a futuro y no tener que resolverlos de nuevo. la implementación del código con programación dinámica para la serie fibonacci sería el siguiente.

3.7 Al desarrollar los ejercicios de Recursión 1 y Recursión 2 de codingBat notamos que el nivel de complejidad de los de Recursión 2 es más alto comparado con los de Recursión 1.

4) Simulacro de Parcial

1. *start + 1, nums, target.*
2. *T(n) = T(n/2) + C*
3.
4. *La suma de los elementos del arreglo y su complejidad es O(n)*

6. Trabajo en Equipo y Progreso Gradual

a) Actas de reunión

Integrante	Fecha	Hecho	Haciendo	Por Hacer
Santiago Soto	19/08/17	Ejercicios codingBat Recursión 1		Ejercicios codingBat Recursión 2
Andres Sanchez	19/08/17	Ejercicios codingBat		Ejercicios codingBat

		Recursión 1		Recursión 2
Santiago Soto	19/08/17	Ejercicios codingBat Recursión 2	Explicando Código GroupSum5	Tabla tiempos
Andres Sanchez	19/08/17	Ejercicios codingBat Recursión 2		Gráficas
Santiago Soto	01/09/17	Calcular complejidad ejercicios 2	Gráficas	
Andres Sanchez	02/09/17	Calcular complejidad Recursión 1		Explicar n
Andres Sanchez	02/09/17	Solución puntos Simulacro parcial	Calculando complejidad	
Jamerson Correa	02/09/17	Solución puntos del 3.3 al 3.7	Documentar código	

b) El reporte de cambios del informe de laboratorio