Laboratorio Nro. 1: Recursión

Alejandra Martínez Vega
Universidad Eafit
Medellín, Colombia
amartinezv@eafit.edu.co

Nombre completo de integrante 2
Universidad Eafit
Medellín, Colombia
Correointegrante2@eafit.edu.co

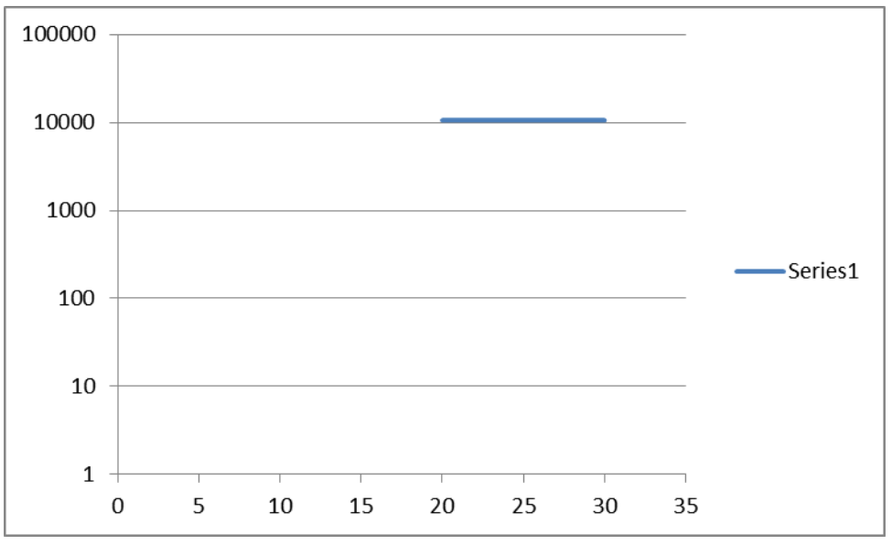
3) Simulacro de preguntas de sustentación de Proyectos

1.

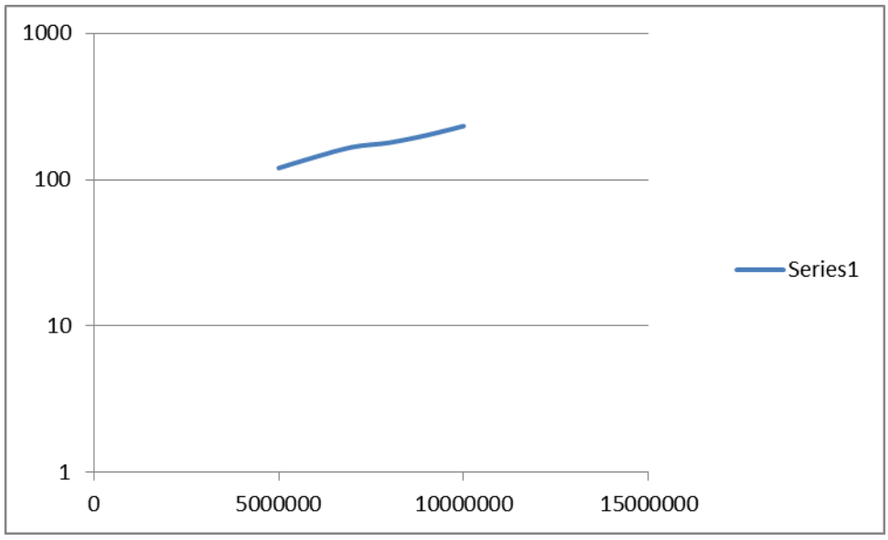
	N = 100.000	N = 1'000.000	N = 10'000.000	N = 100'000.000
R Array sum	2.5	39	241	398
R Array max	Más de 1 min	Más de 1 min	Más de 1 min	Más de 1 min
R Fibonacci	Más de 1 min	Más de 1 min	Más de 1 min	Más de 1 min

2.

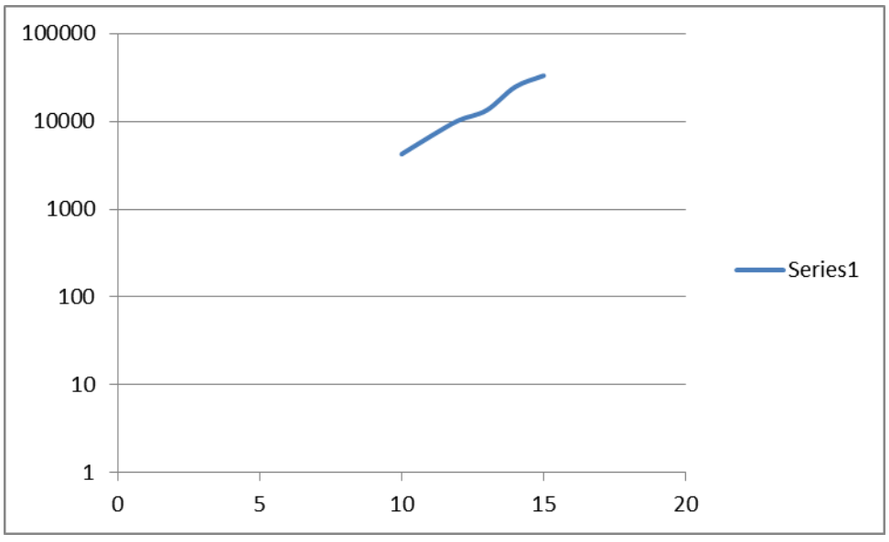
AARAY MAX	
20	10579
22	10579
24	10581
26	10582
28	10583
30	10583



ARRAY SUM	
5000000	120
6000000	143
7000000	167
8000000	179
9000000	201
10000000	232



FIBONACCI	
10	4231
11	6700
12	10226
13	13399
14	24683
15	33146



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

3. **Los tiempos que se obtuvieron en la ejecución de los algoritmos en el laboratorio son coherentes y similares con los resultados teóricos de complejidad que se obtuvieron.**
4. **La causa común de un desbordamiento de pila es una llamada recursiva incorrecta. Normalmente, esto se produce cuando sus funciones recursivas no tienen la condición de terminación correcta, por lo que termina llamándose para siempre. Para tratar con ellos, tendrá que examinar su código. Si usted tiene funciones que se llaman a continuación, compruebe que tiene una condición de terminación. Si tiene que comprobar que al llamar a la función que al menos ha modificado uno de los argumentos, de lo contrario no habrá ningún cambio visible para la función llamada recursivamente y la condición de terminación es inútil.**
5. **El valor más grande que pude calcular para Fibonacci fue 50, no se puede calcular Fibonacci para un millón por que la complejidad del problema es muy alta y tardaría demasiado en calcularlo.**
6. **Se puede calcular Fibonacci de valores grande utilizando la la programación dinámica – Memoización que almacene el resultado de los subproblemas para que no tenga que calcular de nuevo. Por lo tanto, primero se comprueba si la solución ya está disponible, si es así, entonces se utiliza, se calcula y guarda para el futuro. Se puede enfocar en dos tipos de programación ascendente: supongamos que necesitamos resolver el problema para N, comenzamos a resolver el problema con los insumos más pequeños posibles y lo almacenamos para el futuro. Ahora, mientras calcula los valores más grandes, se usa las soluciones almacenadas (solución para problemas más pequeños). Y descendente: romper el problema en sub-problemas y resolverlos según sea necesario y almacenar la solución para el futuro.**
7. **Se puede observar los problemas de CodingBat Recursion 1 tienen una complejidad buena una línea recta cuya pendiente no es un numero grande en cambio la complejidad de los problemas Recursion 2 es horrible una parábola cóncava hacia arriba que cada vez que crece es peor.**

4) Simulacro de Parcial

1. **start+1, nums, target**
2. **T(n)=T(n/2)+C**
3.
   - 3.1 **int res = solucionar(n - a, a, b, c) + 1;**
   - 3.2 **res = max(res, solucionar(n - b, a, b, c) + 1);**
   - 3.3 **res = max(res, solucionar(n -c, a, b, c) + 1);**
4. **e. La suma de los elementos del arreglo a y es O(n)**

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