Laboratorio Nro. 1: Recursividad

Rafael Villegas Michel
Universidad Eafit
Medellín, Colombia
rvillegasm@eafit.edu.co

Felipe Cortés Jaramillo
Universidad Eafit
Medellín, Colombia
fcortesj@eafit.edu.co

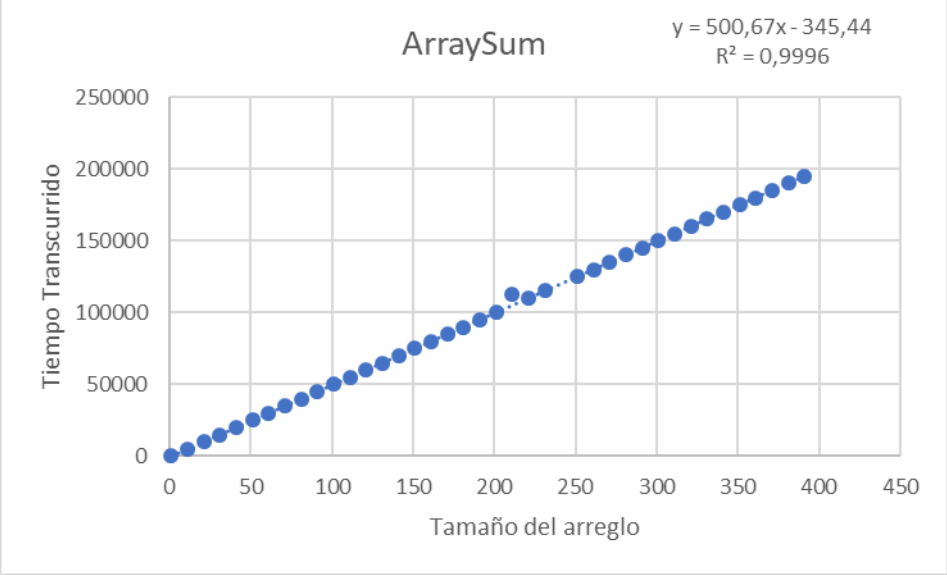
3) Simulacro de preguntas de sustentación de Proyectos

1. Tabla de datos (en milisegundos):

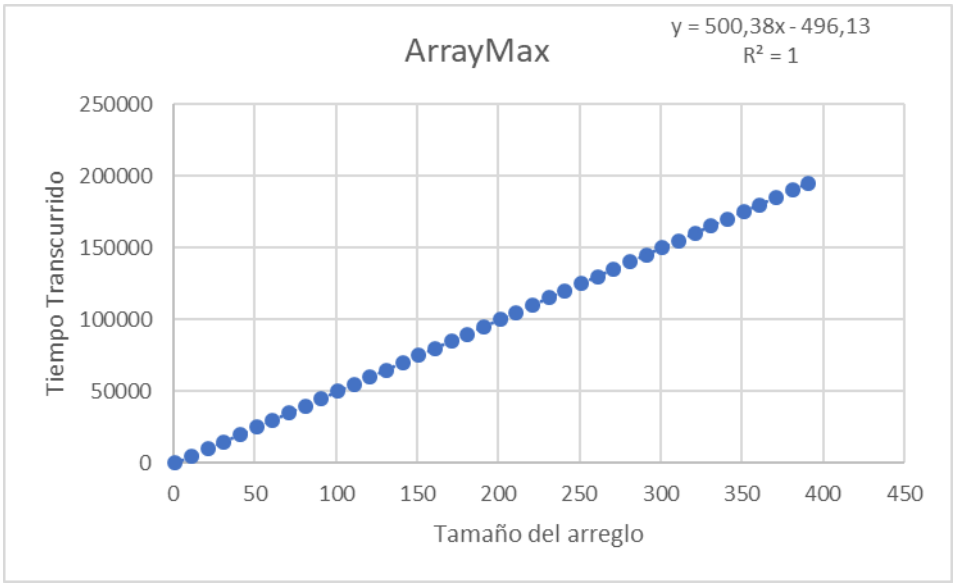
//los valores de Fibonacci son los que están dentro de los paréntesis, ya que se tenían que coger valores mucho mas pequeños.

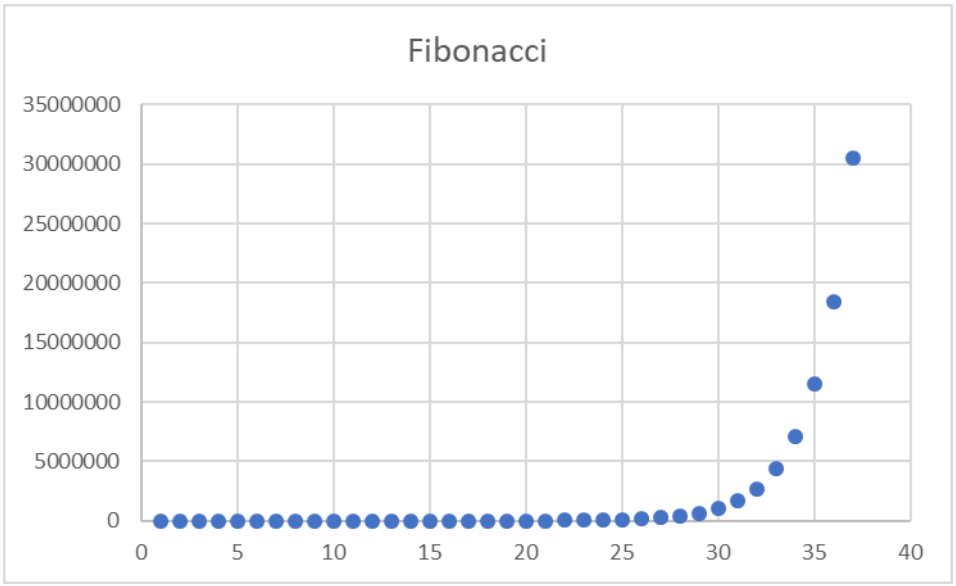
Algoritmo	N=71 (n=7)	N=101 (n=10)	N=201 (n=20)	N=301 (n=30)
ArraySum	35026	50043	100088	150135
ArrayMax	35038	50038	100091	150123
Fibonacci	16	73	8455	1032722

2. Gráficas:



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





3. **Respuesta:** Podemos concluir que la complejidad hallada para cada problema (ArraySum, ArrayMax y Fibonacci) y luego pasándola a la notación O coincide con las gráficas de los datos obtenidos en la parte experimental de los algoritmos, mostrando en cada grafica como los datos van variando, en caso de ArraySum, es un cambio lineal, al igual que arrayMax, mientras que en fibonacci es una ecuación exponencial.
4. **Respuesta:** El Stack Overflow usualmente pasa cuando hay un llamado recursivo incorrecto, que tiene ya sea una mala señalización del contador o

no hay una condición de parada, ocasionando que el método se llame a si mismo eternamente, creando así infinitas variables que se crean a partir de la ejecución del método. Si esto pasa, todas estas variables que son ubicadas en el "montículo" (heap), empiezan a crecer con cada proceso realizado hasta que el espacio que esta disponible para el heap se llena y al buscar otros espacios disponibles, encuentra los que están disponibles para el stack, colisionando así el montículo con la pila, de tal forma que se presenta el error.

5. **Respuesta:** El valor más grande que se pudo encontrar con Fibonacci fue el de 37 elementos, ya que ese fue el punto en que el algoritmo se demoraba demasiado para ejecutarse con cierto número de elementos, que tuvimos que para la ejecución. No se puede ejecutar con 1 millón de elementos ya que el tiempo de ejecución sería demasiado grande, y no se obtendría la respuesta sino hasta después de mucho tiempo, haciéndolo ineficiente.
6. **Respuesta:** En este caso, habría que quitarle el tiempo de espera que tiene incorporado nuestro algoritmo, para que se demore mucho menos en calcular resultados y se puedan evaluar números mucho más grandes sin que tome tanto tiempo.
7. **Respuesta:** La complejidad en los algoritmos de recursion1 casi siempre depende del número "n" de elementos usados, ya que solo tienen un llamado recursivo, mientras que los de recursion2 dependen de 2^n generalmente, ya que, a diferencia de los primeros, tienen dos llamados recursivos.

4) Simulacro de Parcial

1. Start+1, nums, target
2. *T(n/2)+c*
3. *1. n-a, a, b, c*
   2. res, solucionar(n-b, c, a, b)+1
   3. res, solucionar(n-c, b, c, a)+1
4. *E)*

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