RESULTADOS PRACTICA 1.1

TOMA DE TIEMPOS DE EJECUCIÓN:

- ***¿Cuántos años más podremos seguir utilizando esta forma de contar? (currentTimeMillis())***

Si continuamos utilizando esta forma de contar podremos seguir utilizando el método de la clase System durante 292 millones de años más.

- ***¿Qué significa que el tiempo medido sea 0?***

Suponiendo que el algoritmo este bien desarrollado y el JDK no omitiese la operación por que considerase que fuese inútil se debería a que el inicio y final de la operación ocurren en un tiempo inferior a un milisegundo

- **2. ¿A partir de qué tamaño del problema (n) empezamos a obtener datos fiables?**

Cuanto mas grande el n mas fiable el valor, los n por debajo de 50 los depreciamos directamente, para n

3.CRECIMIENTO DEL TAMAÑO DEL PROBLEMA

- ***1. ¿Qué pasa con el tiempo si el tamaño del problema se multiplica por 5***

Al tratarse un algoritmo con complejidad O(n) si multiplicamos por 5 el tamaño del problema debido a su complejidad el tiempo también se vería incrementado por 5. t*5

- ***¿Los tiempos obtenidos son los que se esperaban de la complejidad lineal O(n)?***

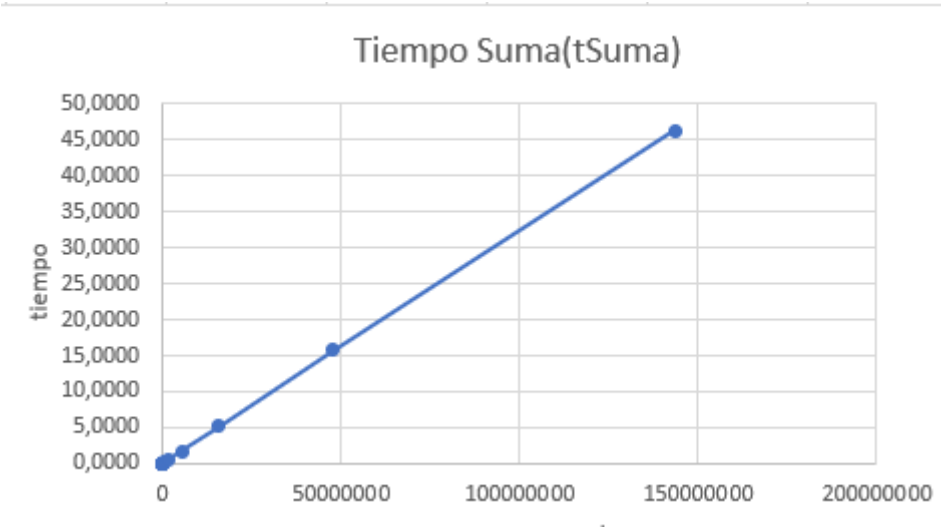
Si ya que al tener una complejidad O(n) el tiempo aumenta de acuerdo a lo esperado, linealmente.

RESULTADOS DE LAS MEDICIONES DE SUMA Y MAXIMO

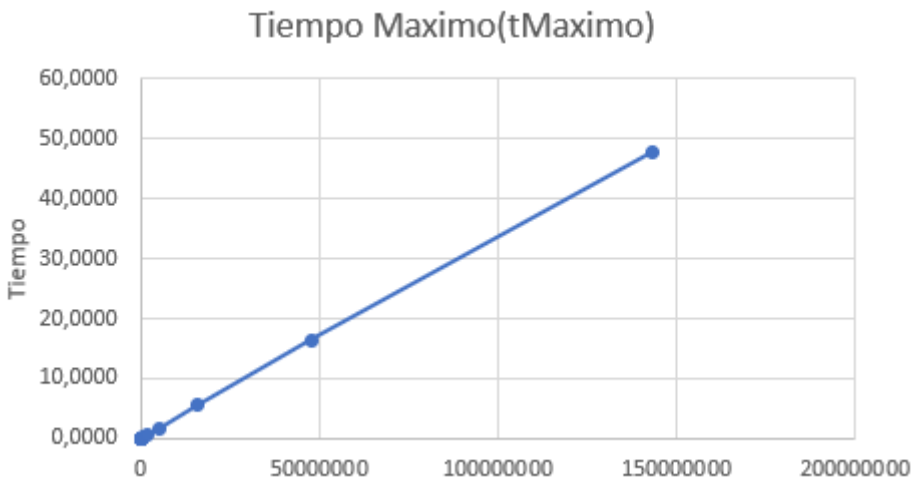
n	tSuma	tMaximo	repeticiones
10	0,0336	0,0404	100000
30	0,0472	0,0067	100000
90	0,0539	0,0404	100000
270	0,0347	0,0404	100000
810	0,0824	0,0311	100000
2430	0,0013	0,0897	100000
7290	0,0041	0,0015	100000
21870	0,0111	0,0048	100000
65610	0,0365	0,0145	100000
196830	0,0515	0,0445	100000
590490	0,1632	0,1328	100000
1771470	0,5364	0,4672	10000
5314410	1,7209	1,6260	10000
15943230	5,2363	5,6060	10000
47829690	15,7610	16,4520	1000
143489070	46,2762	47,8430	1000

Todas las mediciones han sido realizadas desde la misma maquina sin añadirle mas carga al procesador que las mismas mediciones sin ser realizadas simultaneamente para asegurarnos de que los resultados del experimento no se ven corrompidos por otros procesos del procesador.

Las caracteristicas de la maquina son las siguientes, un procesador i7-10700k 3.80GHz y 32,0GB de memoria RAM.



Como vemos los resultados son los esperados, el tiempo aumenta linealmente lo cual concuerda correctamente con la complejidad del problema suma que estamos trarando.



Como vemos el tiempo se corresponde con la complejidad O(n) del problema maximo, quedandonos una relación complejidad/tiempo lineal.

- ***¿cumplen los valores obtenidos con lo esperado?***

Como se señala en el comentario de sendas graficas los valores cumplen con lo esperado ya que la relacion de la complejidad de ambos problemas O(n) se corresponde con la relacion tiempo/complejidad de los resultados de los experimentos.

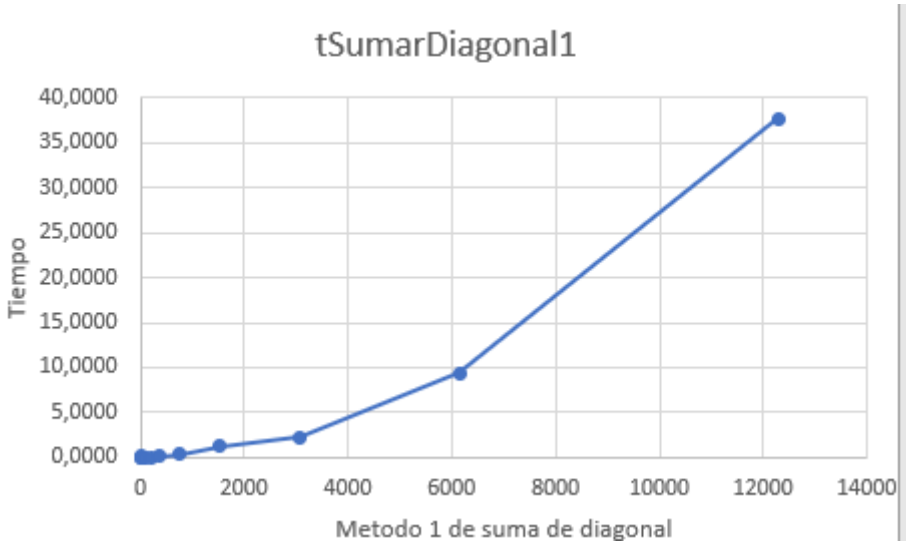
RESULTADOS DE LAS MEDICIONES DE DIAGONALES

n	tSumarDiagonal1	repeticiones	tSumarDiagonal2	repeticiones
3	0,0269	100000	0,0134	100000
6	0,0269	100000	0,0134	100000
12	0,0183	100000	0,02021	100000
24	0,1282	100000	0,0229	100000
48	0,0014	100000	0,0236	100000
96	0,0057	100000	0,0336	100000
192	0,0209	100000	0,01831	100000
384	0,0796	100000	0,0714	100000
768	0,3180	100000	0,1575	100000
1536	1,2341	100000	0,01348	100000
3072	2,2500	10000	0,03721	100000
6144	9,3900	10000	0,08683	100000
12288	37,7400	1000	0,22012	100000

La maquina utilizada para realizar las mediciones de diagonales tiene las mismas caracteristicas, un procesador i7-10700k 3.80GHz y 32,0GB de memoria RAM.

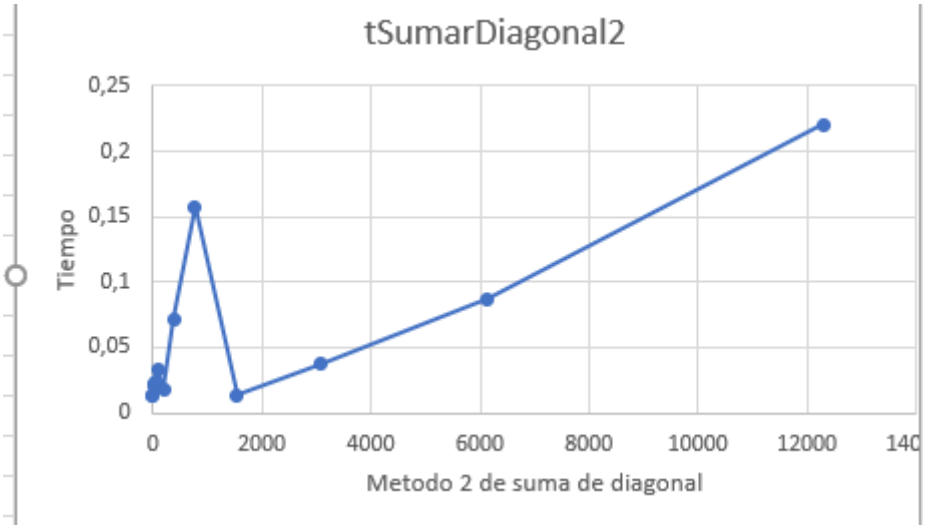
Todas las mediciones han sido realizadas asegurandonos de no añadirle carga al procesador para no corromper las mediciones de los datos

METODO 1 SUMA DIAGONAL COMPLEJIDAD O(n^2)



Como podemos ver en la curva realizada por la relacion n/tiempo en se cumple la complejidad O(n^2) si pudiesemos meter valores aun mayores veriamos como se pronuncia aun mas la curva aumentando mas el tiempo

METODO 2 SUMA DIAGONAL COMPLEJIDAD O(n)



El tiempo aumenta linealmente, como podemos ver existen dos valores atipicos, los cuales seguramente se deban a algun tipo de anomalia en el procesador que pauso temporalmente el proceso en el cual se estaba realizando la suma de la diagonal. Se puede ver como la grafica cumple la relacion esperada respecto a la complejidad/tiempo

RESULTADOS DEL BENCHMARKING:

- ***¿A qué se deben las diferencias de tiempo de ejecución entre uno y otro programa?***

Esta diferencia se debe a que mientras que PYTHON es un lenguaje compilado JAVA es un lenguaje interpretado, se pueden observar diferencias mas que considerables entre los tiempos de ejecucion de ambos lenguajes.

Como podemos observar JAVA es **MUCHO** mas rapido que python existiendo diferencias tan exageradas como esta:

CONTADOR=40960000 n=6400 TIEMPO=14

CONTADOR 40960000 n= 6400 *** tiempo 1998

Este ejemplo es para una complejidad cuadrática pero para una complejidad lineal ocurre lo mismo con todas las mediciones siendo JAVA extremadamente más rápido que PYTHON.

- ***Independientemente de los tiempos concretos, ¿existe alguna analogía en el comportamiento de las dos implementaciones?***

Si, independientemente de las mas que enormes diferencias de tiempo de ejecución entre ambos lenguajes si analizamos ambos por separado la complejidad se mantiene estrictamente en ambos siendo O(n) y O(n**2)

Podemos observar que pese a que elegir un lenguaje de programación más rápido nos puede dar muchísima velocidad, el hecho de elegir la mejor complejidad posible sigue siendo la mayor prioridad para realizar algoritmos más rápidos y ahorrar tiempo de ejecución.