PRACTICA 1.1 –

EJERCICIO 1 - ¿CALCULAR CUÁNTOS AÑOS MÁS PODREMOS SEGUIR UTILIZANDO ESTA FORMA DE CONTAR? EXPLICA EL RAZONAMIENTO SEGUIDO PARA REALIZAR EL CÁLCULO.

Un long (sin signo) como el de currentTimeMillis() puede almacenar hasta 18.446.744.073.709.551.615 números, a partir de ahí, sacamos que…

En total podemos calcular hasta que pasen 5,8494E+8 años…… menos 56 años, porque empezó a utilizarse en 1970 y la fecha actual es 2026.

EJERCICIO 2 - ¿POR QUÉ A VECES EL TIEMPO MEDIDO SALE 0?

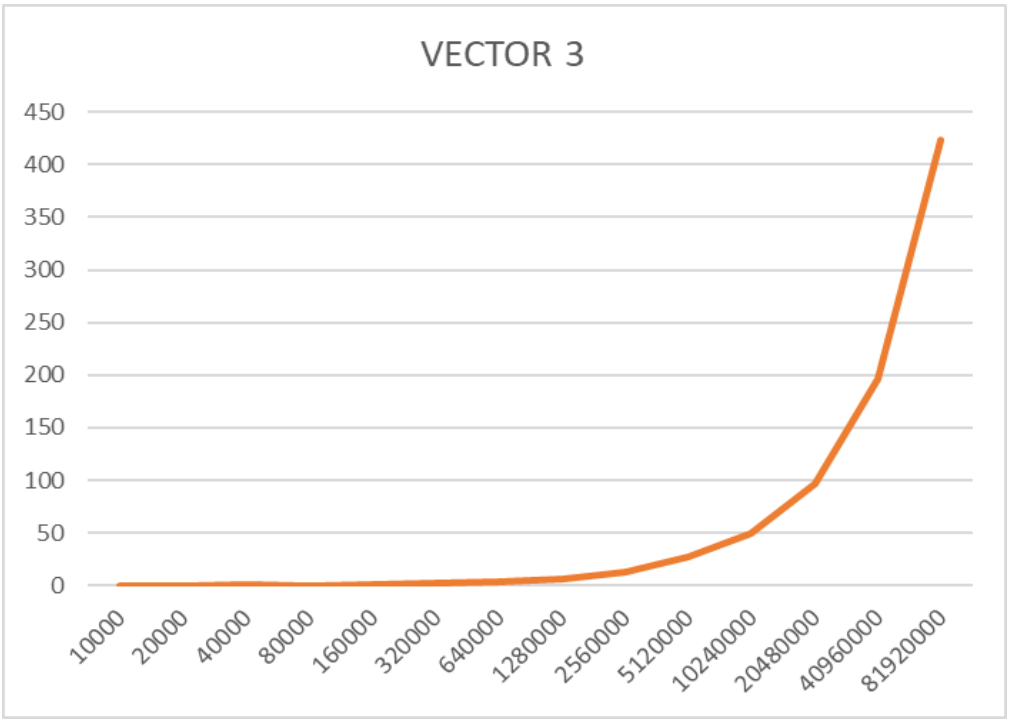
Por el recolector de basura de Java, que puede interrumpir el proceso del programa

¿A PARTIR DE QUÉ TAMAÑO DE PROBLEMA (N) EMPEZAMOS A OBTENER TIEMPOS FIABLES?

TAMAÑO DE PROBLEMA (N)	TIEMPO (ms)
1000000	5 ms / INVALIDO
10000000	48/52/49 ms || Borde ||
50000000	245 ms
100000000	477 ms
200000000	1014 ms

Comenzamos a obtener tiempos fiables a partir de diez millones de tamaño de problema

EJERCICIO 3 – VECTOR 3



n	tTiempo
10000	0
20000	0
40000	0
80000	1
160000	2
320000	1
640000	3
1280000	6
2560000	12
5120000	24
10240000	48
20480000	100
40960000	201
81920000	399

Salen estos tiempos debido a que los valores son muy bajos, si se cogieran valores más altos, la tendencia mostrada por la gráfica se asemejaría más a una lineal, debido a que el método que se mide tiene una complejidad O(n).

EJERCICIO 4 –

¿Qué pasa con el tiempo si el tamaño del problema se multiplica por 2?

Se multiplica el tiempo que se tarda por dos

¿Qué pasa con el tiempo si el tamaño del problema se multiplica por otro k que no sea 2? (Pruebe, por ejemplo, para k=3 y k=4 y compruebe los tiempos obtenidos.)

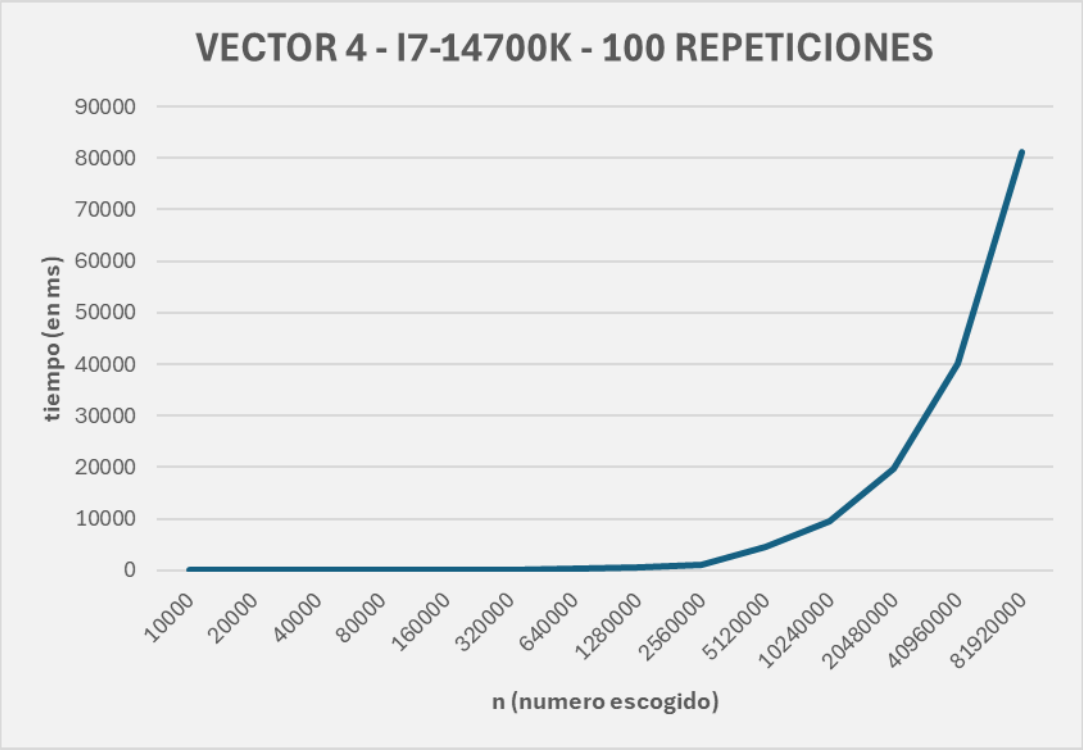
Se multiplica el tiempo que se tarda por la constante K;

```
C:\Users\raulg\Desktop\p11>java -Xint p11.Vector4 10
repeticiones = 10
n       Tiempo
10000   0
20000   1
40000   2
80000   3
160000  6
320000  12
640000  24
1280000 48
2560000 100
5120000 198
10240000        393
20480000        2089
40960000        4173
81920000        8889

Fin de la medicion de tiempos *****
```

```
C:\Users\raulg\Desktop\p11>java -Xint p11.Vector5 10
repeticiones = 10
n       Tiempo
10000   1
20000   2
40000   5
80000   9
160000  19
320000  37
640000  73
1280000 145
2560000 291
5120000 589
10240000        2456
20480000        6520
40960000        12027
81920000        25927

Fin de la medicion de tiempos *****
```

A partir de la marca de los 5120000 hay un escalado fuerte del tiempo no correspondiente a la tendencia (al replicarlo múltiples veces continua apareciendo pero no debería)

¿Razone si los tiempos obtenidos son los que se esperaban de la complejidad lineal O(n)?



La curva descrita por el método define una curva similar a $O(n^2)$. Esto es porque en realidad el método de Vector4 es una curva de $O(n^2)$. Se puede deducir esto del método de Vector 4, que invoca el método “suma” de Vector 1 (que tiene una complejidad de O(n)) dentro de un bucle de complejidad (O(n)) (por tanto, la complejidad es $O(n^2)$ en total)

A partir de lo visto en Vector4.java midiendo los tiempos de suma, realice las tres siguientes Clases: Vector5.java para medir los tiempos del máximo, Vector6.java para medir los tiempos de coincidencias1 y Vector7.java para medir los tiempos de coincidencias2.

5. Con los tiempos obtenidos (en milisegundos) de las clases anteriores, rellene las dos tablas (recuerde lo visto en la práctica anterior: si para algún n tardara más de 60 segundos puede poner FdT ("Fuera de Tiempo")).

Utilizando 50 repeticiones para que las comparaciones entre Tsuma y Tmaximo sean más fiables:

n	Tsuma	Tmaximo
10000	2	2
20000	4	5
40000	7	10
80000	15	20
160000	30	39
320000	61	78
640000	119	155
1280000	242	310
2560000	488	630
5120000	989	1732
10240000	4900	5778
20480000	9850	111257
40960000	20168	22144
81920000	43174	44719

Debido a los largos tiempos de ejecución, no se han incrementado las repeticiones de los métodos de Vector 6 y Vector 7 (repeticiones = 1)

n	Tcoincidencias1	Tcoincidencias2
10000	482	1
20000	1923	0
40000	10278	1
80000	42519	1
160000	FDT	0
320000	FDT	1
640000	FDT	3

1280000	FDT	5
2560000	FDT	11
5120000	FDT	23
10240000	FDT	47
20480000	FDT	356
40960000	FDT	493
81920000	FDT	970