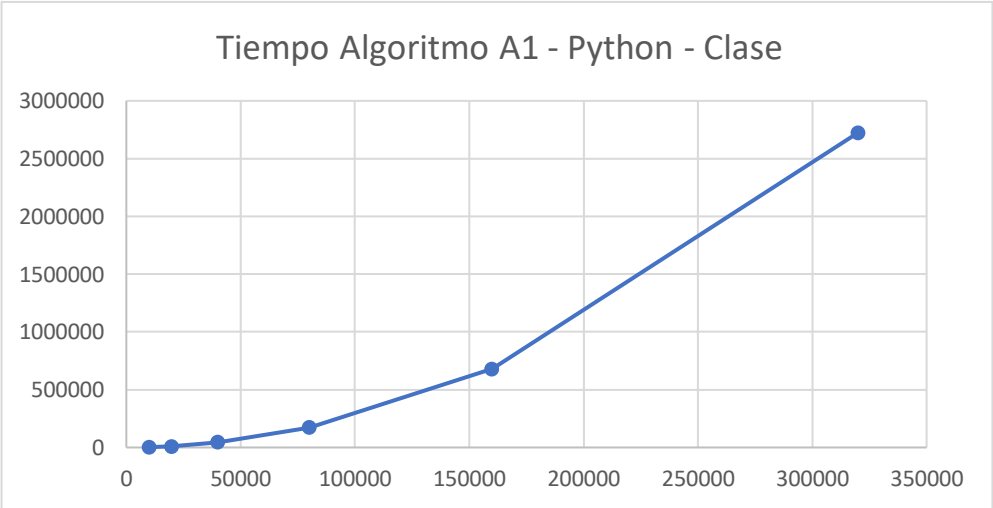
PRÁCTICA 0 ALGORITMIA

1. En las siguientes tablas se quiere representar el tiempo que tarda en ejecutarse el módulo PhythonA1.py utilizando el entorno de desarrollo Python. Además, se compara dicho tiempo tomando mediciones en clase y en casa, con otro ordenador.

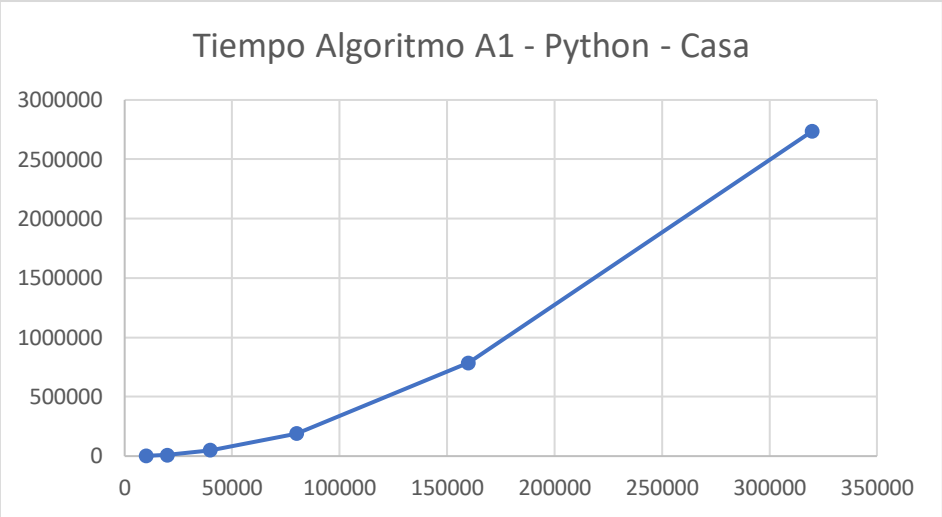
Mediciones realizadas en clase:

n	Tiempo (milisegundos)
10000	2560
20000	10383
40000	43795
80000	172252
160000	680247
320000	2724472
640000	FdT
1280000	FdT

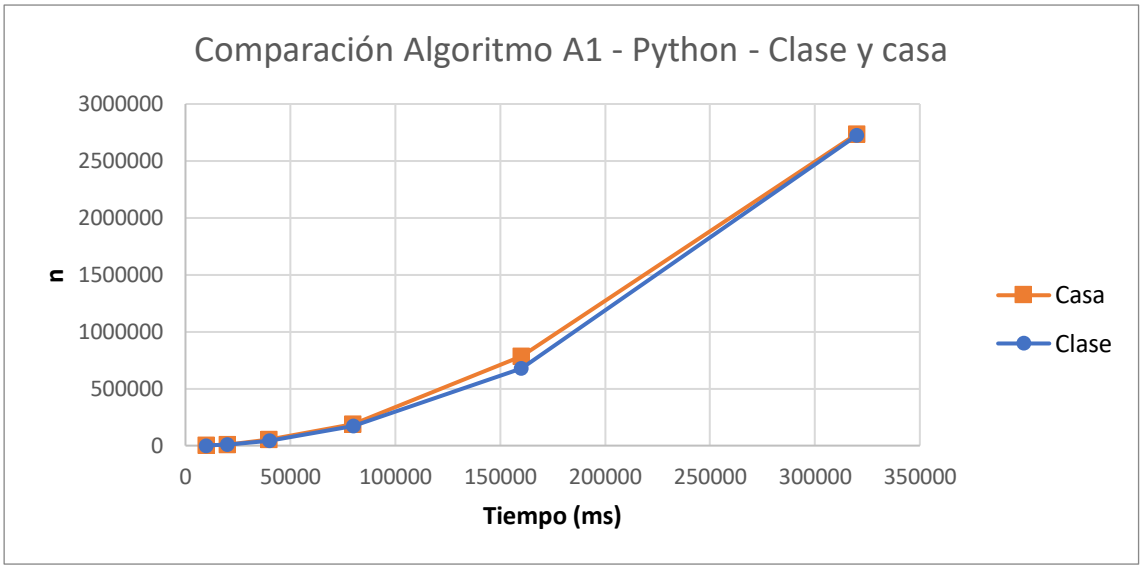


Mediciones realizadas en casa:

n	Tiempo (milisegundos)
10000	2075
20000	8378
40000	51441
80000	189400
160000	785271
320000	2738128
640000	FdT
1280000	FdT



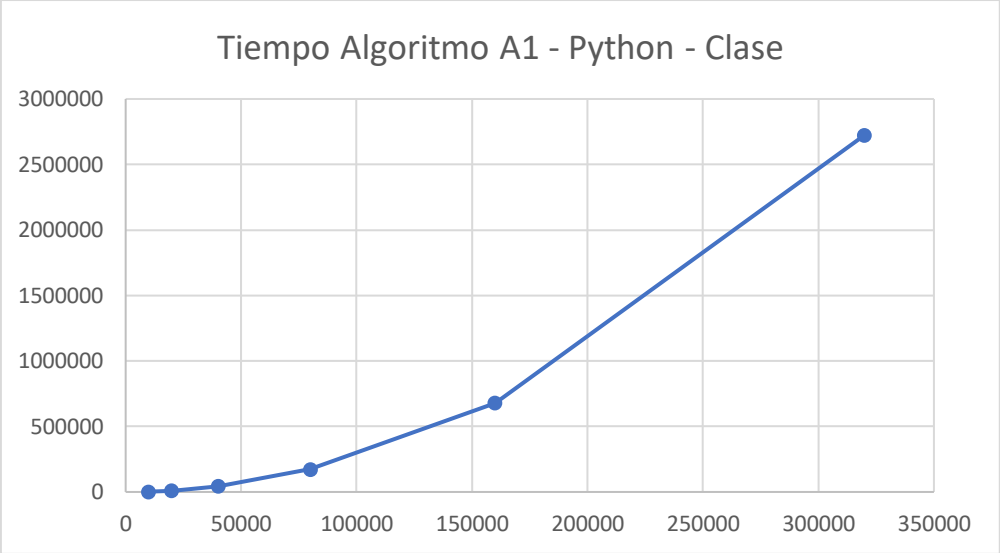
Comparación:



Como se puede observar, los tiempos medidos en el ordenador de casa son un poco más bajos que los tomados en clase para valores más grandes de n. Para entender por qué, habrá que tener el cuenta especificaciones de cada ordenador.

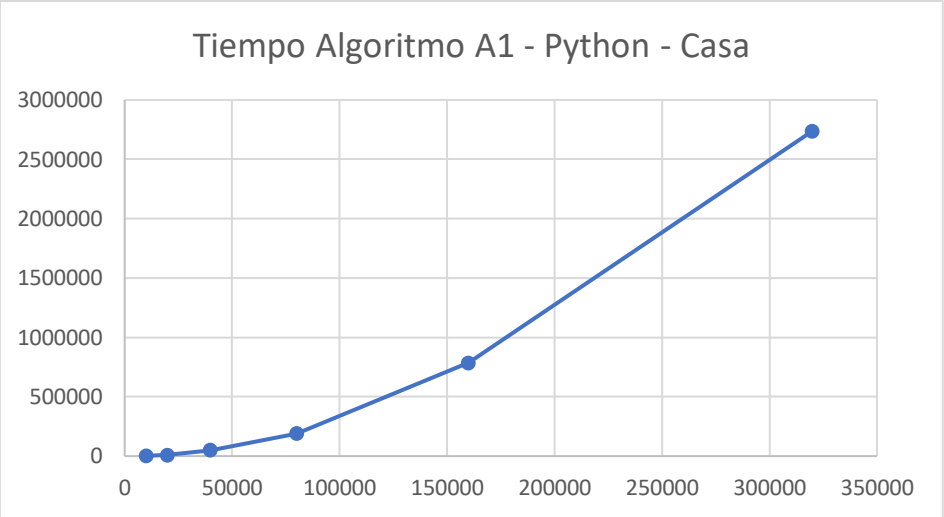
2. Comparación de los tiempos de ejecución en dos ordenadores distintos, referenciando en cada caso la CPU y la cantidad de memoria principal RAM. Esta comparación es importante ya que el tiempo que tarda en ejecutarse el algoritmo depende de la potencia del ordenador en el que se ejecute.
   Mediciones realizadas en clase:
   CPU = Intel Core i5
   Memoria Principal = 8GB

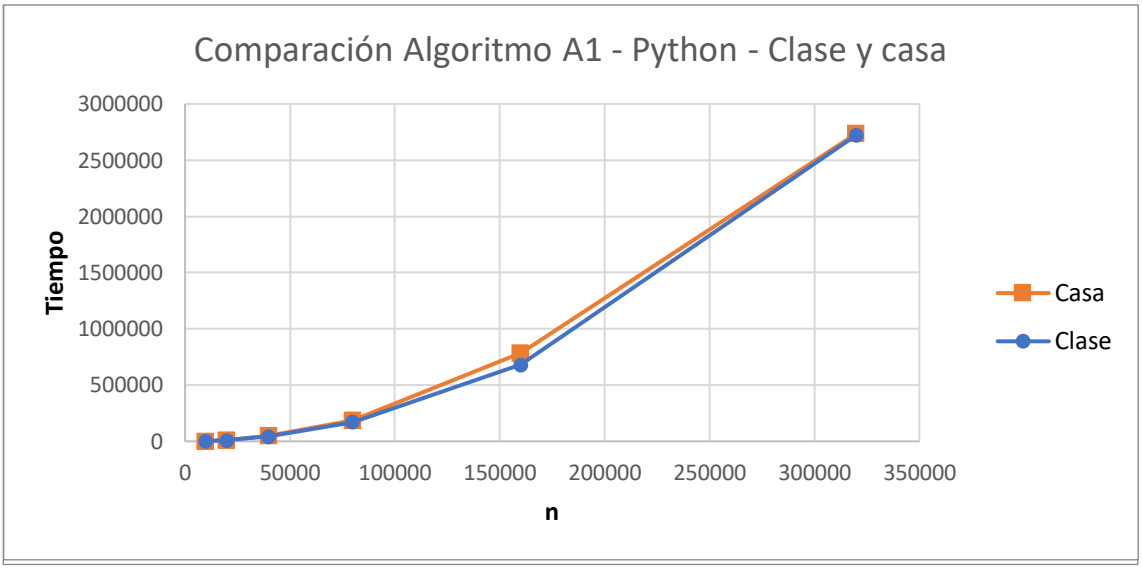
n	Tiempo (milisegundos)
10000	2560
20000	10383
40000	43795
80000	172252
160000	680247
320000	2724472
640000	FdT
1280000	FdT



Mediciones realizadas en casa:
CPU = 11th Gen Intel(R) Core(TM) i7-1165G7
Memoria Principal = 8,00 GB

n	Tiempo (milisegundos)
10000	2075
20000	8378
40000	51441
80000	189400
160000	785271
320000	2738128
640000	FdT
1280000	FdT





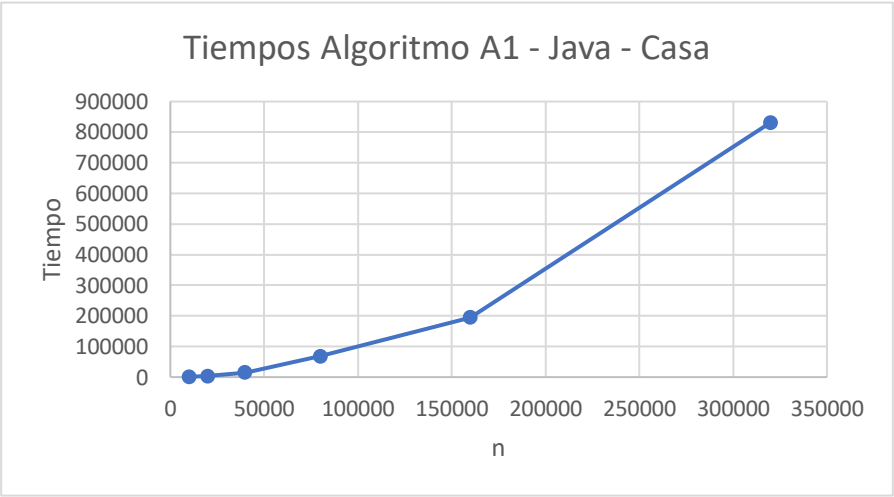
Como se puede observar, los tiempos de ejecución para valores muy grandes de n, son bastante mejores (más bajos) en el ordenador de casa que en el de clase. Esto es así debido principalmente a la CPU, que es mucho más moderna que la de clase, y en menor medida a la memoria principal.

3. Hay que tener en cuenta que el tiempo de ejecución también depende del entorno de desarrollo. Por esto, ahora se van a tomar los tiempos de ejecución en Java utilizando el mismo algoritmo.
   Tiempos tomados en casa con Java:

n	Tiempo (milisegundos)
10000	834
20000	2989
40000	15009
80000	67960
160000	195191
320000	831660
640000	FdT
1280000	FdT

Tiempos tomados en casa con Java:

n	Tiempo (milisegundos)
10000	2075
20000	8378
40000	51441
80000	189400
160000	785271
320000	2738128
640000	FdT
1280000	FdT

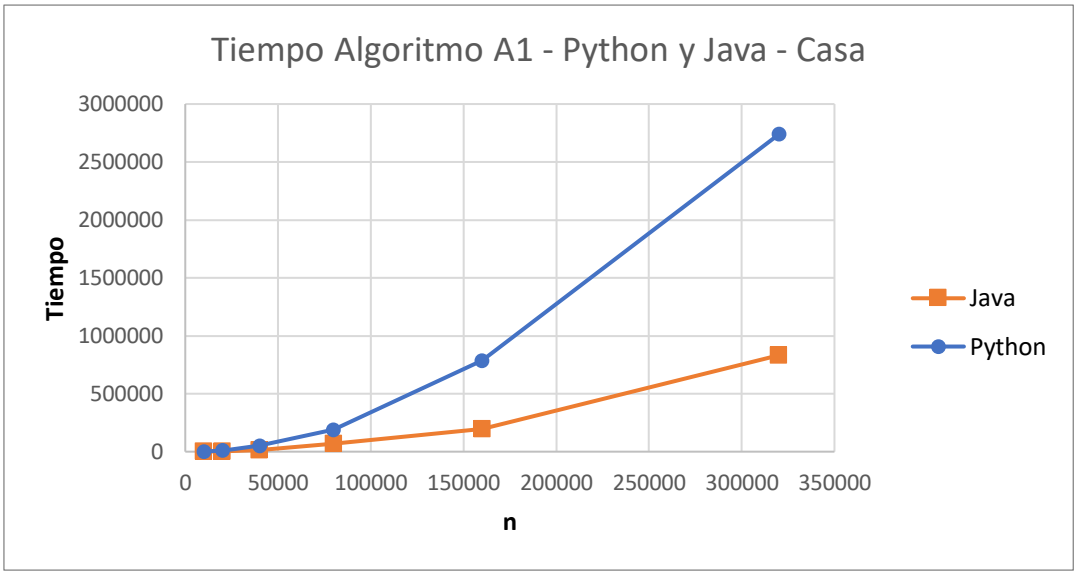


Tiempos tomados en casa con Python:

n	Tiempo (milisegundos)
10000	2560
20000	10383
40000	43795
80000	172252
160000	680247
320000	2724472
640000	FdT
1280000	FdT

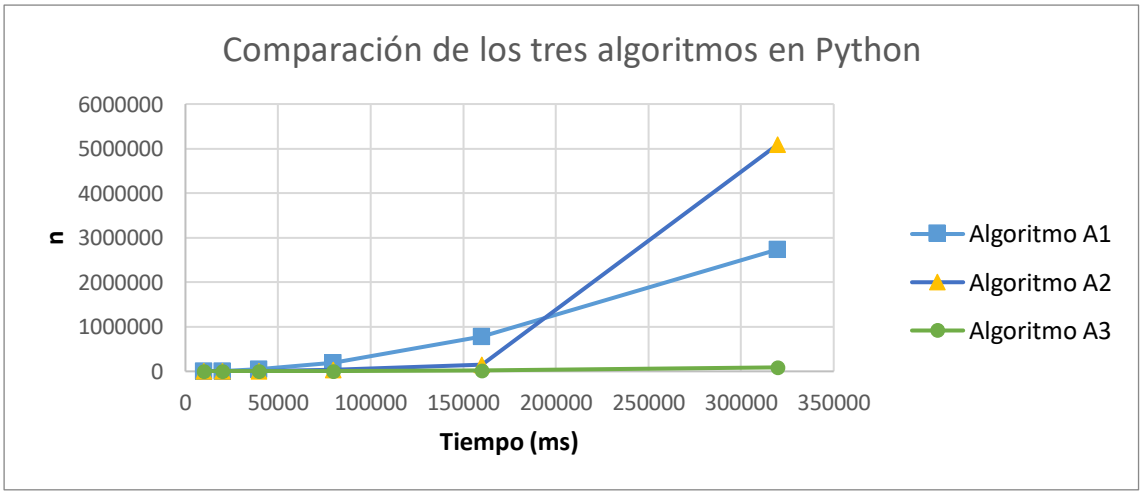
Como se puede ver los tiempos de Java son considerablemente menores que los tiempos tomados mediante el entorno de desarrollo Python. Por esta razón, podemos considerar Java un entorno de desarrollo más eficiente.

Comparación de los tiempos tomados con Java y Python:



4. A continuación, se hará una comparación de los distintos algoritmos y estructuras de datos utilizadas, prestando especial atención al tiempo de ejecución de estos.
   Tiempos medidos en Python con los tres algoritmos:

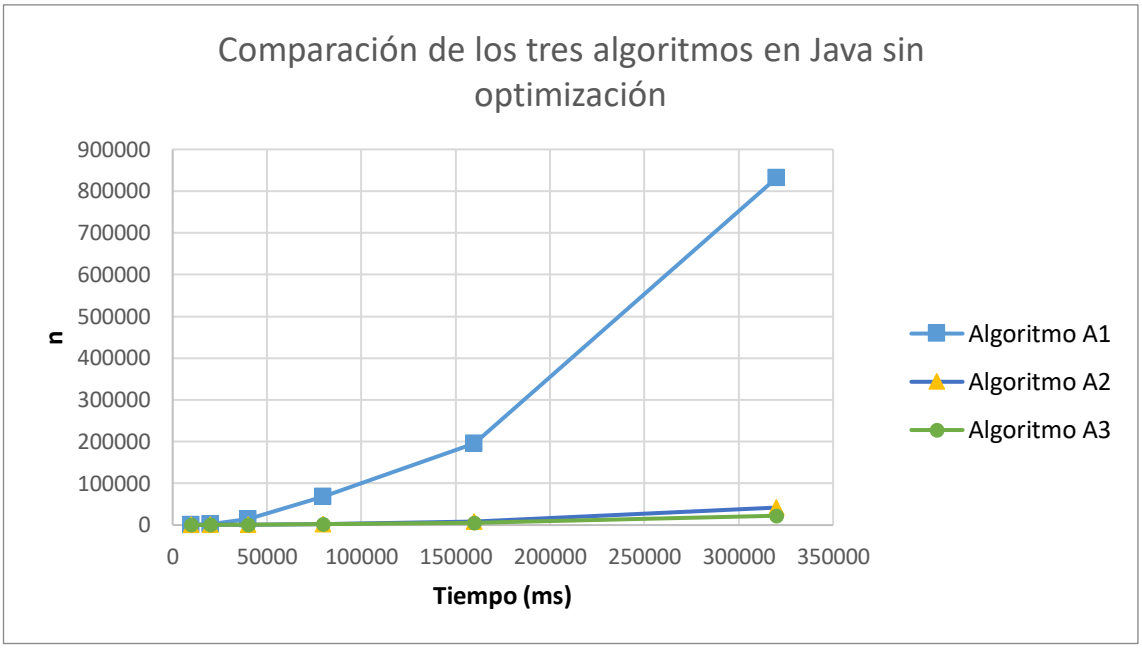
	ALGORITMO A1	**ALGORITMO A2**	**ALGORITMO A3**
n	Tiempo (milisegundos)	Tiempo (milisegundos)	Tiempo (milisegundos)
10000	2075	251	109
20000	8378	863	421
40000	51441	3228	1563
80000	189400	30338	6210
160000	785271	153929	23961
320000	2738128	5098877	89448
640000	FdT	FdT	FdT
1280000	FdT	FdT	FdT



Como se puede ver en la tabla o en la gráfica, el algoritmo 3 en Python es el más eficiente de los 3, pues es el que consigue tiempos de ejecución menores.

Tiempos medidos en Java sin optimización con los tres algoritmos:

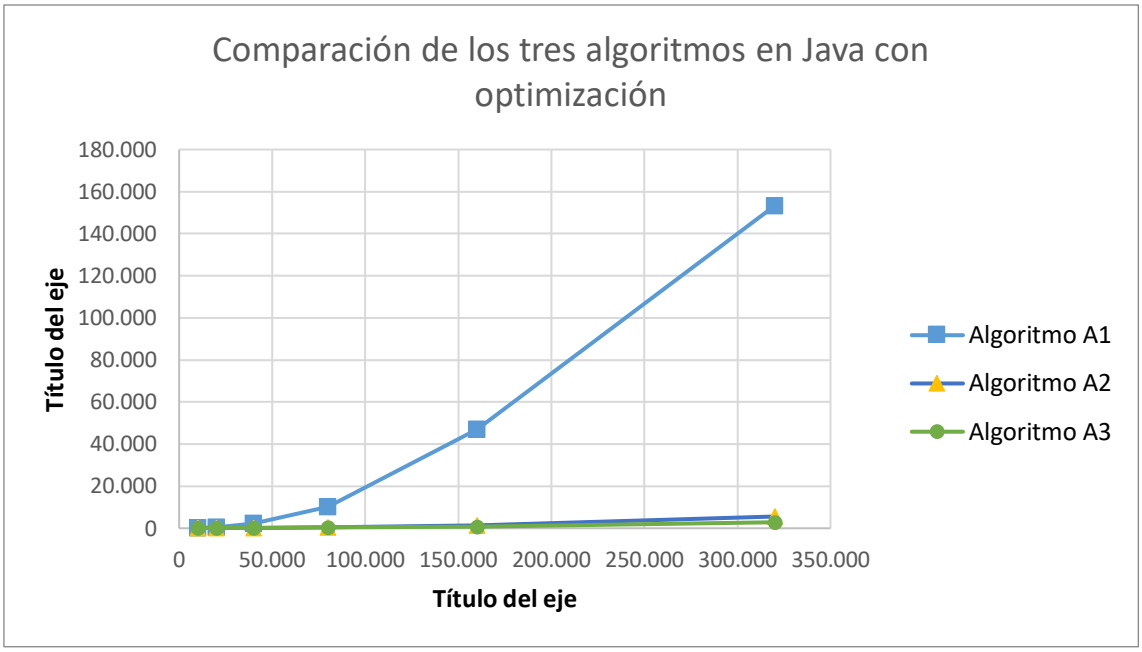
	ALGORITMO A1	ALGORITMO A2	ALGORITMO A3
n	Tiempo (milisegundos)	Tiempo (milisegundos)	Tiempo (milisegundos)
10000	834	50	58
20000	2989	155	207
40000	15009	561	827
80000	67960	1909	2047
160000	195191	8788	5650
320000	831660	41920	22329
640000	FdT	FdT	FdT
1280000	FdT	FdT	FdT



En lo que respecta a Java, los tiempos se miden de dos formas: sin optimización y con optimización. Para empezar, sin optimización los tiempos de los algoritmos A2 y A3 son parecidos al principio, sin embargo, cuanto más grande n más diferencia hay entre ambos. Aunque se podría decir que el algoritmo A3 sigue siendo el más eficiente.

Tiempos medidos en Java con optimización con los tres algoritmos:

	ALGORITMO A1	ALGORITMO A2	ALGORITMO A3
n	Tiempo (milisegundos)	Tiempo (milisegundos)	Tiempo (milisegundos)
10000	142	12	11
20000	554	28	31
40000	2306	107	117
80000	10297	398	306
160000	47089	1512	866
320000	153263	5587	2866
640000	FdT	FdT	FdT
1280000	FdT	FdT	FdT



Finalmente, los tiempos de los tres algoritmos en Java con optimización. Sucede lo mismo que sin optimización, al principio los tiempos de ejecución son parecidos para los algoritmos A2 y A3, pero al final el algoritmo A3 sigue siendo el más eficiente.

CONCLUSIÓN

Durante esta práctica se ha podido observar cómo diferentes factores afectan a la eficiencia de un algoritmo: las estructuras de datos utilizadas, el entorno de desarrollo, el algoritmo utilizado…

Entre la mediciones realizadas, se puede afirmar que el algoritmo A3 es el más eficiente de todos, tanto en Python como en Java, con y sin optimización. Sin embargo, comparando este algoritmo entre las distintas plataformas, se puede observar que es visiblemente más eficiente ejecutándolo en Java con optimización.