

2.4 Actividad 4

findFirstAndLast		
Entradas	Nanosegundos algoritmo 1	Nanosegundos algoritmo 2
[1,4,2,1,6,2,9], 2	≈ 358	≈ 202
[4,2,7,5,4,3,7,2,5,3,4,1], 15	≈ 329	≈ 197
[3,2,1,4,2], 1	≈ 280	≈ 307

Justificación.

El algoritmo 2 que se implemento consta de un solo for anidando if-else por lo que esta última estructura de control tiene una complejidad $O(n)$, disminuyendo considerablemente la complejidad que se tenía al inicio, y aproximandose a $O(n/2)$.

isSudokuValid		
Entradas	Milisegundos algoritmo 1	Milisegundos algoritmo 2
ejemplo2a	≈ 38	≈ 9
ejemplo2b	≈ 33	≈ 8

Justificación.

EL algoritmo 2 que se implemento consta de métodos auxiliares utilizando un solo for con if-else por lo que sus complejidades son $O(n)$ y al implementar el método isSudokuValid se sumo estas dichas complejidades, generando así una complejidad $O(n)$

rotateArray		
Entradas	Milisegundos algoritmo 1	Milisegundos algoritmo 2
[1,4,2,1,6,2,9], 5	≈ 37	≈
[4,2,7,5,4,3,7,2,5,3,4,1], 0	≈ 37	≈
[3,2,1,4,2], 2	≈ 37	≈

Justificación.