2.4. Actividad 4

Crea un archivo Test.pdf, donde llenes las tablas con los resultados obtenidos a partir de la comparación del algoritmo propuesto y el algoritmo mejorado, explicando brevemente (de 2 a 4 renglones) porque el algoritmo que diseñaron mejora la complejidad en tiempo de cada una de las actividades.

Alumnos: Francisco Javier Becerril Lara No Cuenta 317114490 y Joel Miguel Maya Castrejón 417112602.

Decidimos medir los tiempos de ambos integrantes por eso la tabla tiene 4 columnas.

mergeSortedArray(int[], int, int[], int)				
Entradas	Milisegundos algoritmo 1 Pc Miguel	Milisegundos algoritmo 2 Pc Miguel	Milisegundos algoritmo 1 Pc Javier	Milisegundos algoritmo 2 Pc Javier
ArrayA1.txt, 500, ArrayA2.txt, 700	11	2	26	1
ArrayB1.txt, 2000, ArrayB2.txt, 3500	13	7	21	1
ArrayC1.txt, 4000, ArrayC2.txt, 4000	31	4	19	2
ArrayD1.txt, 7000, ArrayD2.txt, 8000	71	7	52	2
ArrayE1.txt, 15000, ArrayE2.txt, 19000	341	7	240	12
ArrayF1.txt, 30000, ArrayF2.txt, 25000	983	6	700	5

Algoritmo mejorado de mergeSortedArray: al copiar los elementos de los arreglos originales al mismo tiempo se están ordenando y se insertan de manera ordenada al arreglo resultante, tal que mejorar la complejidad a O(n+m) .

isValidBoard(int[][])				
Entradas	Milisegundos algoritmo 1 Pc Miguel	Milisegundos algoritmo 2 Pc Miguel	Milisegundos algoritmo 1 Pc Javier	Milisegundos algoritmo 2 Pc Javier
BoardA.txt	6	2	8	1
BoardB.txt	97	55	68	30
BoardC.txt	20853	302	18755	233
BoardD.txt	899	471	656	305
BoardE.txt	292998	1199	233664	974
BoardF.txt	558515	2062	507799	1537

Algoritmo mejorado de isValidBoard: Al usar un arreglo auxiliar con contadores en cada posición podemos revisar si hay posiciones repetidas en renglones y columnas al mismo tiempo sin el uso de for anidados tal que reduce la complejidad a O(n^2) .

rotateArray(int[], int)				
Entradas	Milisegundos algoritmo 1 Pc Miguel	Milisegundos algoritmo 2 Pc Miguel	Milisegundos algoritmo 1 Pc Javier	Milisegundos algoritmo 2 Pc Javier
ArrayA1.txt, 500	4	0	3	0
ArrayB1.txt, 1000	5	0	4	0
ArrayC1.txt, 2000	3	1	7	0
ArrayD1.txt, 3000	9	0	8	0
ArrayE1.txt, 10000	67	0	54	0
ArrayF1.txt, 20000	231	0	187	1

Algoritmo mejorado de rotateArray: Se utiliza únicamente un ciclo para recorrer el arreglo y asignar una nueva posición conforme al número de rotaciones requeridas,esto definido en el método reversa. Esto resulta en complejidad O(n).

Tiempo Total de Ejecución	
Minutos, Segundos Pc Miguel	Minutos, Segundos Pc Javier
16,06	12,45