Estructuras de Datos 2022-1 Práctica Uno: Complejidad Computacional

01 de octubre de 2021

Pintor Muñoz Pedro Joshue - 420053796

Ortiz Castañeda José Ramón - 318357115

**Actividad cuatro (2 puntos)**

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) porqué el algoritmo que diseñaste mejora la complejidad en tiempo de cada una de las actividades.

**mergeSortedArraySolution**

Realiza un solo recorrido mediante un for a un nuevo arreglo con longitud de la suma de los dos enteros recibidos. Mediante tres if valida cuál de los dos elementos de los extremos izquierdos es menor y lo inserta en la primera posición del nuevo arreglo, de tal forma que quedarán ordenados de forma ascendente. Cada if valida que no se esté excediendo el número de elementos solicitados por arreglo para evitar que se inserten más de lo permitido. En caso de ser iguales los dos extremos, se colocan los próximos lugares del nuevo arreglo y hace un doble incremento en las posiciones. Mejora la complejidad en tiempo pues realiza menos recorridos en comparación al algoritmo original.

|  |  |  |
| --- | --- | --- |
| mergeSortedArray(int[], int, int[], int) | | |
|  | Milisegundos algoritmo 1 | Milisegundos algoritmo 2 |
| ArrayA1.txt, 500, ArrayA2.txt, 700 | 2 | 1 |
| ArrayB1.txt, 2000, ArrayB2.txt, 3500 | 11 | 0 |
| ArrayC1.txt, 4000, ArrayC2.txt, 4000 | 9 | 0 |
| ArrayD1.txt, 7000, ArrayD2.txt, 8000 | 32 | 1 |
| ArrayE1.txt, 15000, ArrayE2.txt, 19000 | 167 | 1 |
| ArrayF1.txt, 30000, ArrayF2.txt, 25000 | 503 | 0 |

|  |  |  |
| --- | --- | --- |
| isValidBoard(int[][]) | | |
|  | Milisegundos algoritmo 1 | Milisegundos algoritmo 2 |
| BoardA.txt | 3 | 1 |
| BoardB.txt | 10 | 1 |
| BoardC.txt | 5135 | 23 |
| BoardD.txt | 136 | 0 |
| BoardE.txt | 66306 | 114 |
| BoardF.txt | 161849 | 190 |

En este algoritmo isValidBoardSolution la complejidad en el tiempo de ejecución se ve mejorada por mucho ya que trabaja con solo 2 ciclos for anidados para recorrer todas las lineas y columnas del tablero a comparación de isValidBoard, que utiliza 3 ciclos for anidados, haciendo que el nuestro trabaje de una complejidad O(n3) a O(n2).

|  |  |  |
| --- | --- | --- |
| rotateArray(int[], int) | | |
|  | Milisegundos algoritmo 1 | Milisegundos algoritmo 2 |
| ArrayA1.txt, 500 | 2 | 0 |
| ArrayB1.txt, 1000 | 2 | 0 |
| ArrayC1.txt, 2000 | 4 | 0 |
| ArrayD1.txt, 3000 | 0 | 0 |
| ArrayE1.txt, 10000 | 7 | 0 |
| ArrayF1.txt, 20000 | 18 | 0 |