**David Alejandro Lozano Arreola A01722728**

Programación de estructuras de datos y algoritmos fundamentales
Evidencia 3 - Reflexión

Durante esta evidencia se utilizaron principalmente los algoritmos relacionados con “Heap” y “Binary Search Tree”. Ambas se implementaron con el objetivo de optimizar el proceso de ordenamiento y búsqueda de datos.

La estructura de datos “Heap” tiene el propósito de ordenar datos de manera jerárquica en base a su importancia, la cual puede ser su tamaño. En base a dicha propiedad, se formó el algoritmo de ordenamiento “Heapsort” que simplemente remueve el elemento más importante de la Heap. Al hacerlo, se ejecuta un código interno de la Heap llamado “downsort” que mantiene la jerarquía de la Heap en orden. La razón principal de la implementación del algoritmo Heapsort se debe a que la mayoría de los algoritmos de ordenamiento tienen una complejidad de O(n^2), mientras que Heapsort tiene una complejidad de O(nlogn) la cual es más eficiente. (Programiz, 2023) De manera similar, la estructura de datos BST (Binary Search Tree), se implementó con el objetivo de optimizar la búsqueda de datos. Siendo esta comúnmente de O(n), mientras que un BST utiliza tiempo O(logn) en la mayoría de los casos. Sin embargo, cabe destacar que en el peor de los casos, un BST tarda O(n) en encontrar un dato. (Programiz, 2023)

Finalmente, cabe destacar que los algoritmos de ordenamiento se hacen con el fin de cubrir ciertas necesidades específicas. Es por eso que en ciertos casos un algoritmo puede ser muy eficiente, mientras que en otros simplemente no funciona. Por lo cual, es necesario evaluar la situación antes de decidir el algoritmo a implementar.

**Referencias**

*Binary Search Tree. (2023). Programiz.com.* https://www.programiz.com/dsa/binary-search-tree

*Heap Sort (With Code in Python, C++, Java and C)*. (2023). Programiz.com. https://www.programiz.com/dsa/heap-sort