ReflexAct3.3Marco
Marco Antonio García Mendoza

En el código proporcionado, utilizamos diferentes algoritmos de ordenamiento y búsqueda, cada uno con su nivel de eficiencia expresado en notación big O (O(n) o O(log n), por ejemplo). Estos algoritmos son como diferentes métodos para organizar una colección de cartas de juego: podrías ordenarlas por color, número o palo, dependiendo de cómo quieras encontrar una carta en particular.
Algoritmos usados:
Ordenamiento:

- Bubble sort: compara repetidamente cada par de elementos adyacentes y los intercambia si están en el orden incorrecto. Complejidad: O(n^2)
- Insertion Sort: construye una lista ordenada de elementos uno por uno, tomando cada elemento y colocándolo en la posición correcta en la lista ordenada. Complejidad: O(n^2)
- Merge Sort: Divide recursivamente la lista en mitades más pequeñas, ordena esas mitades y luego las fusiona para obtener una lista ordenada. Complejidad: O(n log n)
- Quick Sort: elige un elemento como pivote y particiona la lista alrededor del pivote, de modo que los elementos menores que el pivote estén a su izquierda y los elementos mayores estén a su derecha. Complejidad: peor caso O(n^2) y su mejor caso O(n log n)

Algoritmos de búsqueda:

- Búsqueda Lineal: busca secuencialmente cada elemento en una lista hasta encontrar el elemento buscado o llegar al final de la lista. Complejidad: O(n)
- Búsqueda Binaria: Compara el elemento buscado con el elemento en la mitad de la lista y descarta la mitad no necesaria del arreglo en cada paso. Complejidad: O(log n)
- Búsqueda en árboles binarios de búsqueda (BST): utiliza la estructura de un árbol binario de búsqueda, donde los nodos están organizados de manera que los elementos menores están a la izquierda y los elementos mayores están a la derecha. Complejidad: En promedio O(log n), pero en el peor caso podría ser de O(n) si el árbol no está balanceado.

Las estructuras de datos similares a los árboles, como el heap utilizado en los ejemplos anteriores, son cruciales cuando necesitamos organizar y acceder a datos de manera eficiente. Imaginemos una biblioteca donde los libros se almacenan por categorías. El uso de un heap en estos ejemplos es como tener un sistema automatizado que organiza los libros de acuerdo con su categoría y te permite agregar o quitar libros de manera rápida y ordenada.
La ventaja de utilizar estructuras de datos tipo árbol radica en su capacidad para manejar grandes cantidades de datos de manera rápida y eficiente. Tomando el ejemplo del heap, el proceso de inserción y eliminación de elementos se realiza en un tiempo logarítmico en relación con la cantidad de elementos en la estructura. Esto es similar a cómo un sistema de gestión de inventario en una tienda puede agregar o quitar productos de los estantes de manera rápida y sin esfuerzo.

Cuando se trata de determinar si una red está infectada o no utilizando la aplicación proporcionada, se pueden seguir varios pasos similares a investigar posibles intrusiones en una casa. Primero, al analizar la bitácora de la red, podemos identificar las direcciones IP que están accediendo al sistema, como ver quién está entrando y saliendo de la casa. Luego, al contar la frecuencia de accesos por cada dirección IP y buscar patrones anormales, podemos detectar comportamientos sospechosos, como si alguien estuviera husmeando alrededor de la casa o intentando abrir puertas cerradas. Finalmente, podemos comparar estas direcciones IP con listas negras de direcciones IP conocidas por estar asociadas con actividades maliciosas, similar a verificar si alguien que intenta entrar a la casa está en una lista de personas no deseadas.

Referencias:

Binary Heap. (2014, November). GeeksforGeeks; GeeksforGeeks. https://www.geeksforgeeks.org/binary-heap/

¿Qué es un algoritmo de ordenamiento? (2023). Platzi. https://platzi.com/tutoriales/1469-algoritmos-practico/6031-que-es-un-algoritmo-de-ordenamiento/

Guillermo, J. (2018, June 23). ¿Qué es la complejidad algorítmica y con qué se come? Medium; Medium. https://medium.com/@joseguillermo_/qu%C3%A9-es-la-complejidad-algor%C3%ADtmica-y-con-qu%C3%A9-se-come-2638e7fd9e8c

Belcic, I. (2020, November 5). Cómo analizar y eliminar malware del router. Cómo Analizar Y Eliminar Malware Del Router; Avg. https://www.avg.com/es/signal/remove-router-virus