Trabajo Integrador - Programación I

Objetivo General

Llevar adelante una investigación práctica y aplicada sobre conceptos fundamentales y avanzados del lenguaje Python, integrando teoría, casos de usos reales, desarrollo de software y difusión de los resultados obtenidos.

Algoritmos de búsqueda y Ordenamiento en Python

Cecilia M. Cosentino (DNI 31.343.839)

Introducción - Algoritmos y complejidad temporal

Un **algoritmo** es un conjunto de instrucciones que se siguen para lograr un objetivo o producir un resultado. Un problema simple se puede resolver usando muchos algoritmos diferentes. Algunas soluciones simplemente toman menos tiempo y espacio que otras. La **complejidad temporal** es el número de operaciones que realiza un algoritmo para completar su tarea (considerando que cada operación dura el mismo tiempo). El algoritmo que realiza la tarea en el menor número de operaciones se considera el más eficiente en términos de complejidad temporal. Normalmente, los lenguajes de programación más utilizados en el análisis de datos como **Python**, **R** o **Julia** tratan de optimizar al máximo la complejidad computacional y es, de hecho, uno de los motivos por los que hay desarrolladores que prefieren uno u otro.

Existen varias definiciones de complejidades temporales:

- **O (1) Tiempo constante**: La sentencia solo necesita una unidad de tiempo para terminar. Es la mejor medida de todas.
- **O (n) Tiempo lineal**: La sentencia necesita n unidades de tiempo para realizar la tarea.
- **O (n^2)Tiempo cuadrático**: La sentencia necesita n-n unidad de tiempo para realizar la tarea.
- **O (C^n)Tiempo exponencial**: La sentencia es muy ineficiente en tiempo. Necesitará una gran cantidad de tiempo para terminar. Es la peor medida de todas.

Algoritmos de búsqueda y ordenamiento

En el mundo del desarrollo de software, los algoritmos de búsqueda y ordenamiento juegan un papel fundamental, estas técnicas permiten organizar y obtener datos de una manera muy eficiente, lo que es esencial para optimizar el rendimiento de las aplicaciones.

Algoritmos de Ordenamiento

En la informática, los algoritmos de ordenamiento son cruciales para la optimización de una tarea, estos permiten organizar datos de manera que puedan ser accedidos y utilizados de manera más eficiente. Un algoritmo de ordenamiento permite reorganizar una lista de elementos o nodos en un orden específico, por ejemplo de forma ascendente o descendente dependiendo de la ocasión. A continuación se mostrarán ejemplos de dos de los algoritmos de ordenamiento más conocidos en la programación, el algoritmo de ordenamiento de burbuja (Bubble Sort), y el algoritmo de Ordenamiento por inserción (Insertion Sort).

a) Ordenamiento de Burbuja (Bubble Sort)

El algoritmo de ordenamiento por burbuja es uno de los más simples pero menos eficientes. Funciona comparando pares de elementos e intercambiándolos si están en el orden incorrecto, este proceso se hace una y otra vez hasta que la lista esté ordenada de forma correcta.

Ventajas	***Desventajas***
● *Simplicidad:* El algoritmo de burbuja es fácil de entender e implementar, lo que lo convierte en una buena opción para introducir conceptos de ordenamiento en la programación.	● *Lento para listas grandes:* Debido a su complejidad cuadrática el algoritmo de burbuja se vuelve lento en la práctica para listas de tamaño considerable.
● *Implementación sencilla:* Requiere poca cantidad de código y no involucra estructuras de datos complejas.	● *No considera el orden parcial:* A diferencia de otros algoritmos, el algoritmo de burbuja realiza el mismo número de comparaciones e intercambios sin importar si la lista ya está en gran parte ordenada.

b) Ordenamiento por inserción (Insertion Sort)

El algoritmo de ordenamiento por inserción es un algoritmo simple pero eficiente. Funciona dividiendo la lista en dos partes, una parte ordenada y otra desordenada, a medida que se recorre la lista desordenada, se insertan elementos en la posición correcta en la parte ordenada.

Ventajas	***Desventajas***
● *Baja sobrecarga:* Requiere menos comparaciones y movimientos que algoritmos como el ordenamiento de burbuja, lo que lo hace más eficiente en términos de intercambios de elementos.	● *Ineficiencia en listas grandes:* A medida que el tamaño de la lista aumenta, el rendimiento del ordenamiento por inserción disminuye. Su complejidad cuadrática de **O(n^2)** en el pero caso lo hace ineficiente para las listas grandes.
● *Simplicidad:* el ordenamiento por inserción es uno de los algoritmos de ordenamiento más simples de implementar y entender. Esto lo hace adecuado para enseñar conceptos básicos de ordenamiento.	● *No escalable:* Al igual que otros algoritmos de complejidad cuadrática, el ordenamiento por inserción no es escalable para listas grandes, ya que su tiempo de ejecución aumenta considerablemente con el tamaño de la lista.

c) Ordenamiento por selección

El organismo por selección encuentra el elemento más pequeño de la lista y lo coloca al inicio. Repite el proceso con el resto de la lista. Resulta más eficiente que el Bubble Sort, pero sigue siendo lento para listas grandes.

Algoritmos de Búsqueda

Los algoritmos de búsqueda son métodos que permiten encontrar la ubicación de un elemento específico dentro de una lista de elementos. Dependiendo de la lista se necesitará utilizar un algoritmo u otro, por ejemplo si la lista tiene elementos ordenados, se puede usar un algoritmo de **búsqueda binaria**, pero si la lista contiene los elementos de forma desordenada este algoritmo no servirá. Para buscar un elemento en una lista desordenada se debe utilizar un algoritmo de **búsqueda lineal**. Estos algoritmos son dos de los más relevantes y conocidos en la programación.

a) Búsqueda Lineal

Los algoritmos de búsqueda lineal, también conocidos como búsqueda secuencial, implican recorrer una lista de elementos uno por uno hasta encontrar un elemento específico. Este algoritmo es muy sencillo de implementar en código pero puede ser muy ineficiente dependiendo del largo de la lista y la ubicación donde está el elemento.

Ventajas	***Desventajas***
● **Sencillez**: La búsqueda lineal es uno de los algoritmos de búsqueda más simples y fáciles de implementar. Solo requiere iterar a través de la lista de elementos uno por uno hasta encontrar el objetivo. ● **flexibilidad**: La búsqueda lineal puede aplicarse a cualquier tipo de lista, independientemente de si está ordenada o no.	● **Ineficiencia en listas grandes**: La principal desventaja de la búsqueda lineal es su ineficiencia en listas grandes. Debido a que compara cada elemento uno por uno, su tiempo de ejecución crece de manera lineal con el tamaño de la lista. ● **No es adecuada para listas ordenadas**: Aunque puede funcionar en listas no ordenadas, la búsqueda lineal no es eficiente para listas ordenadas. En tales casos, algoritmos de búsqueda más eficientes, como la búsqueda binaria, son preferibles.

b) Búsqueda Binaria

El algoritmo de búsqueda binaria es un algoritmo muy eficiente que se aplica solo a listas ordenadas. Funciona dividiendo repetidamente la lista en dos mitades y comparando el elemento objetivo con el elemento del medio, esto reduce significativamente la cantidad de comparaciones necesarias.

Ventajas	***Desventajas***
● *Eficiencia de listas ordenadas:* La principal ventaja de la búsqueda binaria es su eficiencia en listas ordenadas. Su tiempo de ejecución es de O(log n), lo que significa que disminuye rápidamente a medida que el tamaño de la lista aumenta.	● *Requiere una lista ordenada:* La búsqueda binaria sólo es aplicable a listas ordenadas, Si la lista no está ordenada, se debe realizar una operación adicional para ordenarla antes de usar la búsqueda binaria. ● *Mayor complejidad de implementación:* Comparado con la

• *Menos comparaciones:* Comparado con la búsqueda lineal, la búsqueda binaria realiza menos comparaciones en promedio, lo que lo hace más rápido para encontrar el objetivo.	búsqueda lineal, la búsqueda binaria es más compleja de implementar debido a su naturaleza recursiva.

Bibliografía

- Tutoriales y apuntes teóricos brindados por la cátedra de Programación I - UTN 2025
- Python Docs: https://docs.python.org/3/
- W3Schools Python: https://www.w3schools.com/python/
- GeeksForGeeks: https://www.geeksforgeeks.org/python-programming-language/