Trabajo Práctico Integrador: Algoritmos de búsqueda y ordenamiento

Franco Herrera, Brian Gutierrez Colque

¿Por qué Algoritmos de búsqueda y ordenamiento?

Decidimos abordar esta temática ya que los algoritmos de búsqueda y ordenamiento son herramientas fundamentales en el mundo de la programación. Estos permiten manejar datos de manera eficiente y efectiva. En un entorno donde la información se genera a ritmo vertiginoso, comprender estos algoritmos no solo es útil, sino esencial para optimizar el rendimiento de cualquier aplicación. Consideramos que son herramientas básicas que todo programador debe dominar, ya que su comprensión no solo facilita la resolución efectiva de problemas complejos, sino que también sienta las bases para el desarrollo de software robusto y eficiente.

¿Qué son los algoritmos de búsqueda?

Un algoritmo de búsqueda es un conjunto de instrucciones diseñadas para encontrar un elemento específico dentro de una colección de datos. Dependiendo del tipo de estructura en la que se haga esta búsqueda, existen distintos algoritmos adecuados para cada caso.

Estos algoritmos son comunes cuando se necesita encontrar el menor o mayor elemento, si es que se puede establecer un orden, o buscar el índice de un elemento determinado. Son usados por casi todas las aplicaciones.

Tipos comunes de algoritmos de búsqueda

- **Búsqueda Lineal**: Es el más sencillo. Consiste en recorrer secuencialmente todos los elementos hasta encontrar el objetivo o determinar que no está presente. Es funcional pero poco eficiente.
- **Búsqueda Binaria**: Se utiliza sobre estructuras ordenadas. Divide repetidamente el espacio de búsqueda a la mitad, lo que reduce el tiempo necesario para encontrar un elemento.
- **Búsqueda de interpolación**: Es similar a la búsqueda binaria, pero usa una fórmula para estimar dónde puede encontrarse el valor buscado. Es más eficiente que la búsqueda binaria en casos en los que la estructura está ordenada y uniformemente distribuida.
- **Búsqueda exponencial**: Combina estrategias lineales y exponenciales, siendo muy útil cuando se desconoce el tamaño de la estructura en la que se busca.
- **Búsqueda por saltos**: Se basa en saltar algunos elementos y realizar búsqueda lineal dentro del bloque donde probablemente se encuentre el elemento deseado.

Aplicaciones prácticas de algoritmos de búsqueda

- **Sistemas de Base de Datos**: Permiten realizar consultas eficientes para extraer información específica mediante búsquedas rápidas.
- **Motores de Búsqueda**: Optimizan la manera de recuperar resultados relevantes para consultas realizadas por usuarios.
- **Inteligencia Artificial**: Se usan en algoritmos que requieren encontrar soluciones eficientes a problemas complejos, como juegos o rutas óptimas.
- **Procesamiento de Lenguaje Natural (PLN)**: Facilitan tareas como la búsqueda semántica o recuperación eficiente en grandes conjuntos textuales.

¿Qué son los algoritmos de ordenamiento?

Los algoritmos de ordenamiento son un conjunto de procedimientos usados para organizar datos en una secuencia específica, ya sea ascendente o descendente. La ordenación es una operación fundamental en programación que permite mejorar la eficiencia y optimización en la búsqueda de datos, facilitando tareas como la visualización y análisis de información.

Estos algoritmos son importantes por su capacidad para manejar conjuntos de datos grandes y complejos, permitiendo a los programadores crear aplicaciones más eficientes.

Tipos comunes de algoritmos de ordenamiento

- **Ordenamiento por selección**: Divide el conjunto de datos en dos partes: la parte ordenada y la parte no ordenada. En cada iteración, se busca el elemento más chico (o más grande) en la parte no ordenada y se intercambia con el primer elemento no ordenado.
- **Ordenamiento por inserción**: Este algoritmo va insertando cada elemento en su posición correcta respecto a los anteriores. Es muy eficiente para listas pequeñas o casi ordenadas.
- **Ordenamiento rápido**: Usa el principio “divide y vencerás”. Se selecciona un “pivote” y se reorganizan los elementos para que todo los menores al pivote estén a su izquierda y los mayores a su derecha. Luego, se aplica recursivamente este proceso a las sublistas.
- **Ordenamiento por mezcla**: Divide el conjunto original hasta que cada sublista tenga un solo elemento. Luego, combina las sublistas para crear listas ordenadas.
- **Ordenamiento burbuja**: Compara pares adyacentes e intercambia elementos si están en el orden incorrecto. Este proceso se repite hasta que no hay más intercambios necesarios, indicando que la lista está ordenada.

Aplicaciones prácticas de algoritmos de ordenamiento

- **Búsqueda eficiente**: Algunos algoritmos de búsqueda funcionan mejor en conjuntos de datos que ya están ordenados, como la búsqueda binaria.
- **Análisis estadístico**: La organización de datos permite realizar análisis como cálculos de promedios o medianas con mayor facilidad.
- **Visualización**: La presentación clara de datos mediante gráficos o tablas organizadas se logra gracias a los algoritmos de ordenamiento.

Caso práctico

Consiste en un programa que permite al usuario hacer una búsqueda sobre una lista de dos maneras posibles: **Búsqueda lineal** y **Búsqueda binaria**. Además, se le permite al usuario ordenar la lista a través de dos algoritmos de ordenamiento: **Ordenamiento rápido** y **Ordenamiento burbuja**. El programa muestra el tiempo de ejecución de cada algoritmo seleccionado.

El tipo de dato que decidimos usar es **diccionario**, para darle más sentido al programa y que no se trate simplemente de ordenar números o palabras, sino de ordenar **alumnos**.

¿Qué es un diccionario?

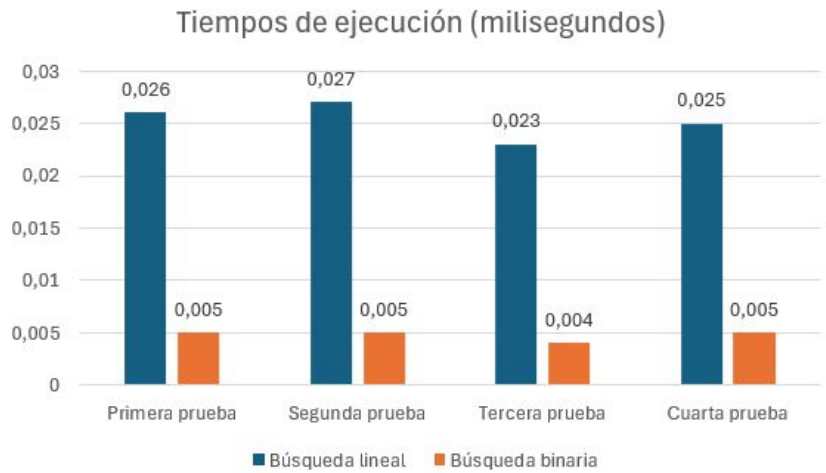
Antes de pasar a la prueba del programa, necesitamos definir qué es un diccionario. Un diccionario es un tipo de dato estructurado que permite guardar pares clave-valor. Entonces, en nuestro caso, cada alumno está conformado por un **nombre**, **legajo** y **promedio**:

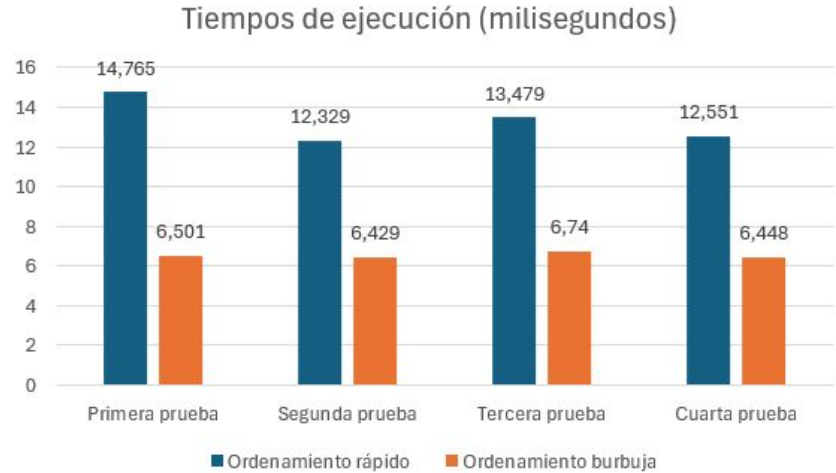
```
alumno = {
    "nombre": "Franco",
    "legajo": 1,
    "promedio": 75
}
```

Probamos el programa…

Resultados obtenidos

Para probar nuestro caso práctico, creamos una función que genera una lista de alumnos de manera dinámica. El problema fue que esta lista se genera para cada ejecución, por lo que no usamos exactamente la misma lista para cada caso. Sin embargo, nos aseguramos de que la lista contenga siempre la misma cantidad de elementos.





Conclusiones

Los algoritmos de búsqueda y ordenamiento son pilares fundamentales en el campo de la programación, ya que permiten gestionar y manipular datos de manera eficiente. La capacidad de buscar un elemento específico en una colección o de ordenar datos para facilitar su análisis es esencial en la mayoría de las aplicaciones informáticas.

En este trabajo, aplicamos en un caso práctico cuatro tipos de algoritmos diferentes, dos de ordenamiento y dos de búsqueda. Pudimos observar como la búsqueda binaria se impone ante la búsqueda lineal, con la desventaja de que la primera necesita recibir una lista ordenada. También vimos cómo el ordenamiento burbuja reduce los tiempos de ejecución a la mitad en comparación con el ordenamiento rápido. Así, dimos cuenta de que cada algoritmo tiene sus propias ventajas y desventajas en tiempos de complejidad temporal y espacial.