**Pablo David Soberanis Cetz ADA4 - Busq SecBin**

**Ejercicio 1: Búsqueda Secuencial**

[**https://github.com/pablosoberanisc/UNIDAD\_6.git**](https://github.com/pablosoberanisc/UNIDAD_6.git)

**Descripción:**

En este ejercicio, se da un arreglo desordenado de números enteros y se busca la posición de un número específico dentro de ese arreglo. Si el número no se encuentra en el arreglo, se retorna -1.

**Explicación:**

La búsqueda secuencial consiste en recorrer cada elemento del arreglo uno por uno y comparar si el elemento actual es igual al objetivo. Si se encuentra, se retorna el índice; de lo contrario, se continúa la búsqueda hasta recorrer todo el arreglo.

**Justificación:**

Se usa este método porque el arreglo está desordenado y la búsqueda secuencial no requiere que el arreglo esté ordenado. Es un método simple y fácil de implementar.

**Mejoras:**

Este método no se puede mejorar directamente con otra técnica sin ordenar primero el arreglo. Sin embargo, si se tiene acceso al arreglo repetidamente y el costo de ordenarlo una vez es aceptable, se puede ordenar el arreglo y luego usar búsqueda binaria para búsquedas subsecuentes, lo que mejorará significativamente la eficiencia.

**Conclusiones:**

La búsqueda secuencial es adecuada para arreglos desordenados y pequeños, pero su eficiencia es lineal, 𝑂(𝑛), lo que puede ser ineficiente para arreglos grandes.

**Ejercicio 2: Búsqueda Binaria**

[**https://github.com/pablosoberanisc/UNIDAD\_6.git**](https://github.com/pablosoberanisc/UNIDAD_6.git)

**Descripción:**

En este ejercicio, se da un arreglo ordenado de números enteros y se busca la posición de un número específico utilizando búsqueda binaria. Si el número no se encuentra en el arreglo, se retorna -1.

**Explicación:**

La búsqueda binaria consiste en dividir repetidamente el rango de búsqueda a la mitad y comparar el elemento del medio con el objetivo. Dependiendo de la comparación, se descarta la mitad del rango, reduciendo así el número de comparaciones necesarias.

**Justificación:**

Se usa este método porque el arreglo está ordenado, lo que permite aprovechar la eficiencia de la búsqueda binaria, que tiene una complejidad de 𝑂(log𝑛), mucho más eficiente que la búsqueda secuencial para arreglos grandes.

**Mejoras:**

Para datos que cambian frecuentemente, mantener el arreglo ordenado puede ser costoso. Sin embargo, para arreglos estáticos o que se ordenan una vez, la búsqueda binaria es óptima. No hay una mejora directa sobre la búsqueda binaria para arreglos estáticos ordenados.

**Conclusiones:**

La búsqueda binaria es altamente eficiente para arreglos ordenados, reduciendo el tiempo de búsqueda drásticamente comparado con la búsqueda secuencial. Sin embargo, requiere que el arreglo esté ordenado de antemano.

**Comparación y Conclusión**

* **Búsqueda Secuencial:** Es adecuada para arreglos desordenados y es fácil de implementar, pero no es eficiente para grandes conjuntos de datos debido a su complejidad lineal 𝑂(𝑛).
* **Búsqueda Binaria:** Es altamente eficiente con una complejidad logarítmica 𝑂(log𝑛), pero requiere que el arreglo esté ordenado. Es ideal para grandes conjuntos de datos donde se pueden permitir los costos iniciales de ordenación.

La elección del método de búsqueda depende de las características del arreglo (ordenado vs desordenado) y la frecuencia de las búsquedas. Para arreglos desordenados, la búsqueda secuencial es una solución rápida y simple. Para arreglos ordenados o cuando se puede ordenar previamente, la búsqueda binaria es más eficiente.