Notación Big O en algoritmos

1. **Algoritmos de búsqueda**
2. Secuential Search

* Mejor caso: Notación Big O = O(1) ya que el target se encuentra en la primera posición del array.
* Peor caso: Notación Big O = O(n) ya que el algoritmo recorre todo el array encontrando el target en la última posición o no está dentro del array.
* Caso promedio: Notación Big O = O(n) ya que el target está en alguna posición entre el inicio y el final del array

Ya que el algoritmo solo utiliza un for para recorrer el arreglo la notación Big O común para este tipo de algoritmo de búsqueda es O(n).

1. Binary Search

* Mejor caso: Notación Big O = O(1) ya que el target se encuentra en la mitad del array.
* Peor caso: Notación Big O = O(log n) ya que el número se encuentra en los extremos del array, o no se encuentra, y en cada iteración el algoritmo va dividiendo en mitades cada vez más pequeñas el array.
* Caso promedio: Notación Big O = O(log n) ya que el número se encuentra en algún lugar del array (pero no en el medio o en los extremos) y similar al peor caso tiene que ir dividiendo el array hasta encontrarlo.

**Conclusiones y recomendaciones de los algoritmos de búsqueda:**

Se puede ver a través de las complejidades que nos brinda la notación Big O que el algoritmo de búsqueda binaria es mucho más eficiente que el algoritmo secuencial ya que el secuencial en el peor de los casos tendrá que iterar n veces un arreglo hasta encontrar el valor esperado, pero en el caso de la búsqueda binaria cada vez el rango de búsqueda se va reduciendo. Y esta ventaja se verá mejor reflejada con arreglos con una gran cantidad de posiciones, si la cantidad de posiciones es mínima, posiblemente no se note esta diferencia o incluso podría se podría ver más eficiente el secuencial si el target está en las primeras posiciones.

1. **Algoritmos de ordenamiento**
2. Bubble Sort

* Mejor caso: Notación Big O = O(n) ya que el array ya está ordenado y se recorrerá el for una sola vez.
* Peor caso: Notación Big O = O(n²) ya que en este caso tenemos la estructura do while más un for interno que se ejecutará n veces, es decir, se logrará ordenar el array hasta la última iteración.
* Caso promedio: Notación Big O = O(n²) ya que en este caso tenemos la estructura do while más un for interno que se ejecutará más de 1 vez y menos de n veces para lograr ordenar el array.

1. Insertion Sort

* Mejor caso: Notación Big O = O(n) ya que el array ya está ordenado y se harán comparaciones solo una vez por elemento
* Peor caso: Notación Big O = O(n²) ya que el array se encuentra completamente desordenado y se tendrán que hacer comparaciones con cada uno de los elementos del array (n veces) hasta que se ordene.
* Caso promedio: Notación Big O = O(n²) ya que el array no está completo desordenado pero se tendrán que hacer menos de n iteraciones por cada elemento hasta que el algoritmo lo ordene por completo.

1. Selection sort

* Para este algoritmo tanto para el mejor, peor o caso promedio la notación Big O es O(n²) ya que independientemente esté ordenado, o no, el array siempre intentará buscar el valor mínimo para reemplazarlo en la posición actual que tiene el valor mínimo y como podemos observar requiere de dos estructuras repetitivas para lograr este cometido (para nuestra implementación se utilizaron 2 for anidados).

**Conclusiones y recomendaciones de los algoritmos de ordenamiento:**

En este caso podemos observar que si nos basamos en la notación Big O los tres algoritmos de ordenamientos tienen la misma complejidad O(n²) ya que cada uno de estos algoritmos hace una serie de intercambios o comparaciones por cada uno de las posiciones que se tienen en el array. A pesar de esto, en la práctica podemos ver que el que peor resultados tiene es el Bubble sort ya que hace muchos intercambios por cada posición en cambio el insertion sort es el que pudimos observar tiende a dar mejores resultados ya que en cada pasada hace un menor número de iteraciones por cada posición.