**Análisis del rendimiento de los algoritmos utilizados en la tarea\_45**

**Búsqueda secuencial**: Es el algoritmo de búsqueda más simple y en principio menos eficiente, consiste en recorrer uno a uno los elementos de una lista o array y comparar cada elemento con el valor que se está buscando. Es decir, hay que recorrer la lista de principio a fin, con lo que para una lista de n elementos, en el peor de los casos (si el elemento que buscamos se halla al final de la lista) habría que realizar n comparaciones y la complejidad aumenta de manera lineal con respecto a la cantidad de datos en la lista. La notación Big O es por tanto del tipo lineal**, O(n).**

**Búsqueda binaria**: Es un ejemplo de la estrategia “divide y vencerás” y como requisito es necesario que la búsqueda se realice en una lista ordenada. La búsqueda consiste en comparar el valor a buscar con uno de los elementos (llamado pivote) de una lista ordenada, normalmente se escoge el elemento central. Si el valor que buscamos corresponde con el pivote, ya hemos terminado la búsqueda, si el valor que buscamos es menor que el pivote, se descarta la parte superior de la búsqueda y se realiza el mismo procedimiento con la parte inferior de la lista, por último, si el valor que buscamos es mayor que el pivote, se descarta la parte inferior de la lista y se realiza el mismo proceso con la parte superior de la lista. De esta manera, con cada iteración se reduce a la mitad el tamaño de lista donde hay que buscar.

1 división -> n/2

2 división -> n/22

3 división -> n/23

.

.

.

x división -> n/2x

En el peor de los casos, habría que seguir dividiendo hasta que sólo queda en la lista un elemento:

1= n/2x

Y despejando la x para saber cuántas divisiones o iteraciones hay que realizar:

2x= n, x= log2 (n)

Por tanto, la notación O grande es **O (log2(n))**

**Ordenamiento\_Quicksort:** El algoritmo consiste en lo siguiente:

- Dividir el vector que se desea ordenar en dos bloques. En el primero se sitúan todos los elementos que son menores que un cierto valor que se toma como referencia (pivote), mientras que en segundo irían el resto.

–Este procedimiento se repite dividiendo a su vez cada uno de estos bloques y repitiendo la operación anteriormente descrita.

–La condición de parada se da cuando el bloque que se desea ordenar está formado por un único elemento (bloque ordenado).

–El esquema seguido por este algoritmo es el de “divide y vencerás”.

En este caso, el número de operaciones que habría que relizar serían:

Por un lado, las veces que habría que dividir las listas sería:

x= log2n

Por otro lado, en cada sublista el número de operaciones para ordenar los elementos llevaría asociado un tiempo de ejecución O(n).

Por tanto, la notación O grande del algoritmo sería O (n log2n)

**Conclusiones**: En cuanto a qué algoritmo de búsqueda sería más eficiente, la respuesta depende del tamaño y la naturaleza de los datos de entrada. Si por ejemplo, los elementos sobre los que se realiza la búsqueda están ordenados, entonces (y teniendo en cuenta el peor de los casos) el algoritmo de búsqueda binaria es mucho más rápido que el secuencial (O(log n) frente a O(n)). Esa diferencia se hace patente cuando el tamaño de la lista de entrada es muy grande. Sin embargo, si la lista no está ordenada, en el caso de la búsqueda binaría habría que usar también un algoritmo de ordenación, como Quicksort, que lleva asociado un tiempo de ejecución, con lo que la eficiencia de la búsqueda binaria decrecería.

Tablas con notación Big-O asociada a diferentes estructuras de datos y diferentes algoritmos de búsqueda aquí: <https://www.bigocheatsheet.com/>