**ANALISIS DE RENDIMIENTO DE LOS ALGORITMOS EMPLEADOS EN LA TAREA 45**

Los algoritmos empleados en la resolución de la tarea han sido los siguientes:

Algoritmo de ordenación: Algoritmo Quicksort

Algoritmo de búsqueda: Algoritmo secuencial y Algoritmo de búsqueda binaria

**Análisis de rendimiento:**

En el análisis de rendimiento de los algoritmos interesa saber el tiempo de ejecución de los mismos y como este incrementa con el paso del tiempo. Cuando se habla de tiempo de ejecución realmente se está hablando del número de operaciones que el algoritmo tiene que realizar y, como éstas se incrementan con el aumento del número de datos de entrada. La notación O permite comparar el número de operaciones y ver como estas crecen en función del número de datos de entrada.

Algoritmo de ordenación: El algoritmo Quicksort es un algoritmo en el que se escoge un pivote que divide el arreglo en 2 subarreglos. Por un lado, queda el subarreglo con los valores más bajos que el pivote y, por otro lado, el otro subarreglo contiene los valores más altos que el pivote. Se realiza esta operación tantas veces como sea necesario, en los subarreglos resultantes, hasta llegar a un arreglo de un elemento que no necesita ser ordenado. Utiliza la técnica recursiva de Divide y vencerás.

El rendimiento de este algoritmo depende de la elección del pivote de forma que en la situación más desfavorable, que es con un arreglo ordenado donde escogemos de pivote el primer elemento, el rendimiento será O(n2). Esto se produce porque se generan tantos niveles en la pila de llamadas como elementos tenemos, ya que al ir eligiendo siempre de pivote el primer elemento siempre queda un subarreglo vacío a la izquierda y otro con el resto de elementos a la derecha. Además, en cada nivel de la pila hay que tocar todos los elementos para ordenarlos de esta forma tenemos O(n) de tamaño de la pila de llamadas y O(n) dentro de los niveles lo que da O(n2)

Si, en vez de coger el primer elemento de pivote se coge, en un arreglo ordenado, el elemento del medio el tamaño de la pila de llamadas se reduce a la mitad, es decir, 4. Este es el mejor caso, por lo que se tiene O(log n). Sin embargo, dentro de cada nivel hay que tocar todos los elementos para ordenarlos en cada subarreglo así que tenemos O(n). El rendimiento promedio para este algoritmo es O(n log n). El caso promedio difiere significativamente del caso peor, sobre todo en la forma en la que crece al aumentar significativamente los datos de entrada

En el caso de los algoritmos de búsqueda utilizados tenemos:

* el algoritmo secuencial que compara uno a uno todos los elementos con el buscado hasta que se encuentra este en el arreglo
* el algoritmo de búsqueda binaria, en un arreglo ordenado, compara el elemento a buscar con el que está en mitad del arreglo y dependiendo de si es menor o mayor va generando subarreglos, que son de la mitad de tamaño, en donde se encuentre el elemento buscado.

En el caso del algoritmo secuencial su eficiencia expresada en notación asintótica será O (n) para el caso peor que se daría si el número buscado no está en la lista. Si el número buscado fuera el primero de la lista, caso mejor, tendríamos O(1), y el caso promedio se podría establecer como O(n/2) que no difiere mucho del caso peor. El crecimiento que experimentan en tiempo de ejecución al aumentar sensiblemente el número de datos de entrada es semejante en ambos casos.

En el algoritmo de búsqueda binaria, el mejor caso también es O(1), si el elemento buscado coincide en la primera operación. Sin embargo, para el caso peor es O(log n) ya que en cada operación se divide por la mitad el arreglo por lo que cada vez hay menos elementos en donde buscar.

**Comparación rendimiento de Algoritmos de búsqueda:**

En el algoritmo secuencial su tiempo de ejecución aumenta proporcionalmente al número de entrada de datos. Sin embargo, en la búsqueda binaria el tiempo de ejecución aumenta de forma logarítmica por lo que para grandes cantidades de datos el algoritmo de búsqueda binaria es mucho más eficiente que el de búsqueda secuencial y la diferencia se agranda a más datos de entrada:

Sin embargo, el algoritmo de búsqueda binaria necesita de un arreglo previamente ordenado para poder ejecutarse, a diferencia de la búsqueda secuencial, por lo que en caso de darse un arreglo desordenado habría que sumar a la eficiencia del algoritmo binario la eficiencia del algoritmo de ordenación que se utilice. La eficiencia de los algoritmos de ordenación oscilan por lo general entre O (n log n) y O(n2)(se añade a la tarea un archivo phyton denominado ordenamiento.py donde se indican algunos de los algoritmos de ordenamiento y se explica brevemente su complejidad en notación Big-O. Este archivo se puede utilizar como modulo, si se quiere, para ordenar la lista que figura en el archivo tarea\_45.py con cualquiera de las funciones existentes en él). Esto quiere decir que para arreglos desordenados es más eficiente la búsqueda lineal, ya que a la complejidad del algoritmo de búsqueda binaria habría que sumarle la complejidad del algoritmo de ordenación y el resultado siempre será una función con un crecimiento en tiempo de ejecución mas acentuado que el que se registra para búsqueda lineal.