***Algoritmos de clasificación (Sorting)***

Si por ejemplo, tenemos un listado de elementos (como en una especie de matriz e idealmente contabilizados como números) y deseamos encontrar un elemento en especifico de dicho listado... Sería útil valerse de un motor de busqueda que fuera lo suficientemente eficiente y recursivo para buscar al elemento en cuestión, más en los casos especiales donde nos enfrentamos a un listado exhorbitantemente grande. Entonces, si queremos encontrar un elemento puntual de ese listado, hay dos formas de hacer esto (siendo una indudablemente mucho más eficiente que la otra): ***busqueda líneal (linear search) y busqueda binaria (binary search).***

Respectivamente, la ***busqueda líneal*** consiste en revisar elemento por elemento, del listado, para encontrar al elemento de nuestro interés. Indudablemente, y sin importar si el listado está ordenado o no, la busqueda líneal siempre encontrará al elemento en cuestión; pues, revisa uno por uno a cada uno de ellos. Sin embargo, este mecanismo de busqueda resultaría demasiado engorroso si estamos ante listados de elementos contados por millones. Toca, como dice el dicho, *“dividir para vencer”*. Pero, para “dividir y vencer”; es decir, no tener que hacer la engorrosa tarea de revisar uno por uno, los elementos, para encontrar alguno de nuestro interés, tenemos que estar necesariamente ante un listado de elementos YA ***ordenados.*** Y este es el costo de oportunidad, precisamente, de la busqueda del tipo binaria, la **busqueda binaria** sólo funciona si se enfrenta a un listado de elementos ya ordenados de menor a mayor: esto para poder definir patrones de busqueda mucho más eficientes según el orden de los elementos y no tener que revisar uno por uno; sino que, hacer divisiones para acercarse más rápido al elemento en cuestión. En caso que el listado no esté ordenado, debe ordenarlos primeros, esto se hace por medio de: ***Algoritmos de clasificación.*** Hay, principalemente, tres tipos de algoritmos de clasificación:

***1. Ordenación por selección (Selection sort): la que se tarda con regularidad en listados desordenados.***

En la ordenación por selección la forma en que funciona el algoritmo es tomando una matriz de números y repitiendo algún proceso: el proceso de mirar a través de todo el conjunto de números para intentar encontrar el elemento más pequeño de toda la matriz y luego llevar ese elemento más pequeño al principio de la matriz. Posteriormente, buscamos en el resto de la matriz al siguiente elemento más pequeño y, cuando lo encontremos, traemos el siguiente elemento más pequeño al principio de la matriz (en la segunda posición); y repetimos ese proceso, mirando continuamente a través de la parte restante desordenada de la matriz, encontrando al elemento más pequeño, e intercambiándolo con el elemento en la siguiente posición... y así sucesivamente hasta dar con una matriz ordenada. Dato: en matrices ordenadas, suele ser el más lento de todos en la revisión.

***2. Ordenación por burbujas (Bubble sort): la que se tarda más en listados desordenados.***

Mientras tanto, la clasificación de burbujas funciona de manera un poco diferente. También realiza pasadas a través de la matriz, pero compara dos valores a la vez. Mira cada par de valores, sin volarse ningún par, y analiza si ese par de número analizados en cuestión se encuentran en el orden correcto. Es decir, intenta ordenar de menor a mayor cada par, asegurandose de que, para cualquier par de números observados, el número más pequeño esté a la izquierda del número mayor; si no lo están, los cambia de posición. Así es el proceso de ordenación por burbujas, pasando continuamente por este proceso de “considerar pares de valores e intercambiarlos si es necesario”... hasta que llega al final de la matriz. Dato: en matrices ordenadas, suele ser muy rápido su proceso de revisión ya que se da cuenta inmediatamente de que el listado ya se encuentra ordenado en sí y que no necesita hacer más nada.

***3. Ordenación por combinación (Merge sort): la que se tarda menos en listados desordenados.***

Finalmente, la ordenación por combinación funciona de manera fundamentalmente diferente. Funciona tomando una matriz y dividiendola en una mitad izquierda y una mitad derecha, y ordenando cada una de esas mitades primero; y, después de que forma recursiva, clasificamos cada una de esas mitades, las fusionamos nuevamente. Repitiendo este proceso de forma recursiva una y otra vez, podemos construir una matriz ordenada muy rápidamente (es el tipo de ordenación más rapida para matrices desordenadas). *La dinamica es más o menos así:* Luego de dividir por la mitad a la matriz completa, si al contar los elementos del par de matrices restantes de dicha división da un número par, debe organizar matrices de dos en cada una de ellas y ordenarlas de menor a mayor (por cada sub-matriz); pero, si al contar los elementos del par de matrices restantes, de la división inicial, da un número impar en cada una de ellas: debe primero bajar el primer elemento y organizar los demás elementos en sub-matrices de pares o de dos en dos (como se hace en el primer caso) y ordenarlas, uno por uno, de menor a mayor por cada sub-matriz.