**TAREA\_44**

Notación-asintótica | Notación-Big-O | Algoritmos-de-ordenación

**Notación-asintótica**: En computación, la notación asintótica nos permite representar la complejidad, y por tanto la eficiencia, de un algoritmo, de tal manera que podemos inferir el aumento de operaciones requeridas al aumentar el tamaño de la entrada (input). Uno de los conjuntos de la notación asintótica es la notación O grande o Big-O, en este grupo entrarían las funciones que cumplen la condición de que existan dos constantes positivas (x0 y c), de manera que a partir de un valor x0, para todo x existe otra función cg(x) donde |cg(x)|>|f(x)|.

**Notación-Big-O**: Se trata de una herramienta para saber como de eficiente es un algoritmo. Para determinar esta eficiencia necesitamos saber como de complejo es dicho algoritmo, es decir, como se comporta (en términos de tiempo de ejecución y espacio de memoria requerido) a medida que el tamaño de los datos de entrada crece. Puede tener diferentes órdenes de complejidad:

**Orden constante**: aunque el tamaño del input aumente el tiempo de ejecución no aumenta. **O(1).**

**Orden lineal**: en este caso el tiempo de ejecución aumenta de manera lineal con respecto al tamaño del input -> **O(n).**

**Orden logarítmico**: la relación entre el crecimiento del número de datos y el del tiempo de ejecución es de tipo logarítmico ->**O(log n).**

**Orden exponencial**: el tiempo de ejecución aumenta de manera exponencial con respecto al aumento del tamaño del input-> **2n**

**Orden cuadrático**: el tiempo de complejidad aumenta de manera cuadrática con respecto al tamaño de los datos de entrada -> **O(n2)**, este tipo de algoritmos no son muy eficientes cuando el tamaño de los datos es muy grande.

**Algoritmo de búsqueda secuencial**: Se realiza la búsqueda de un elemento en un conjunto de elementos comparando el elemento a buscar con cada uno de los elementos del conjunto para determinar si es igual. En este tipo de búsqueda es necesario recorrer todos los elementos del conjunto para determinar si el elemento a buscar está presente o no en el conjunto.

**Algoritmo de búsqueda binaria**: En este tipo de algoritmos se utiliza la técnica divide y vencerás, se busca el elemento central del conjunto donde se va a realizar la búsqueda (pivote) y en base a él, el conjunto se divide en dos subconjuntos (parte inferior: desde el inicio al pivote, y parte superior: desde el pivote al final), se compara el elemento de búsqueda con el pivote y según sea mayor o menor, nos quedamos con la parte superior o inferior del conjunto. Esta operación se repite de manera iterativa hasta que el conjunto de búsqueda sea indivisible. Es imprescindible que el conjunto donde se realiza la búsqueda esté ordenado.

Algoritmos-de-ordenación:

Información extraída de: http://lwh.free.fr/pages/algo/tri/tri\_es.htm

**Ordenamiento por inserción**: es una manera muy natural de ordenar para un ser humano, y puede usarse fácilmente para ordenar un mazo de cartas numeradas en forma arbitraria.

La idea de este algoritmo de ordenación consiste en ir insertando un elemento de la lista ó un arreglo en la parte ordenada de la misma, asumiendo que el primer elemento es la parte ordenada, el algoritmo ira comparando un elemento de la parte desordenada de la lista con los elementos de la parte ordenada, insertando el elemento en la posición correcta dentro de la parte ordenada, y así sucesivamente hasta obtener la lista ordenada.

**Ordenación de burbuja**: funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posición si están en el orden equivocado. Es necesario revisar varias veces toda la lista hasta que no se necesiten más intercambios, lo cual significa que la lista está ordenada.

**Ordenamiento de burbuja bidireccional**: (también llamado "método de la sacudida" o "coctail sort" o **"shaker sort")** es un algoritmo de ordenamiento que surge como una mejora del algoritmo ordenamiento de burbuja.

Si ya habéis visto como funciona el algoritmo de ordenación por burbuja habréis observado que los números grandes se están moviendo rápidamente hasta al final de la lista (estas son las "liebres"), pero que los números pequeños (las "tortugas") se mueven sólo muy lentamente al inicio de la lista.

Una solución es de ordenar con el método de burbuja y cuando llegamos al final de la primera iteración, no volver a realizar el cálculo desde el principio sino que empezaremos desde el final hasta al inicio. De esta manera siempre se consigue que tanto los números pequeños como los números grandes se desplacen a los extremos de la lista lo más rápido posible.

**Ordenación "Gnome":** el algoritmo de ordenación conocido como **Gnome\_sort** fue inventada por Hamid Sarbazi-Azad, (profesor de la universidad de Sharif, una de las mayores universidades de Irán) quien lo desarrolló en el año 2000 y al que llamó Stupid sort (Ordenamiento estúpido).

Cuando Dick Grune lo reinventó y documentó, no halló evidencias de que existiera y en palabras suyas, dijo de él "the simplest sort algorithm" (es el algoritmo más simple) y quizás tenga razón, pues lo describió en sólo cuatro líneas de código. Dick Grune se basó en los gnomos de jardín holandés, en cómo se colocan en los maceteros y de ahí también el nombre que le dio.

El algoritmo es similar a la ordenación por inserción, excepto que, en lugar de insertar directamente el elemento a su lugar apropiado, el algoritmo realiza una serie de permutaciones , como en el ordenamiento de burbuja.

**Ordenamiento por selección**: consiste en encontrar el menor de todos los elementos del vector e intercambiarlo con el que está en la primera posición. Luego el segundo más pequeño, y así sucesivamente hasta ordenarlo todo.

**Ordenación de peine o de Dobosiewicz**: en 1980, Wlodzimierz Dobosiewicz propuso este algoritmo en su breve artículo "An Efficient Variation of Bubble Sort", Information Processing Letters, vol. 11, num. 1, 1980. En él escribió literalmente: "Bubble sort can be improved in yet another way, which is similar to Shell’s version of the insertion sort." ("La ordenación por burbuja se puede mejorar de otra manera adicional, que es similar a la versión de Shell de la ordenación por inserción").

Posteriormente fue redescubierto y popularizado por Stephen Lacey y Richard Box en un artículo publicado por la revista Byte en abril de 1991.

En el ordenamiento de burbuja, cuando dos elementos cualesquiera se comparan, siempre tienen un espacio (distancia entre ellos) de 1. La idea básica del algoritmo **CombSort** es que el espacio pueda ser mucho mayor de uno.

El ordenamiento Shell también se basa en esta idea, pero es una modificación del algoritmo de ordenamiento por inserción más que del algoritmo de ordenamiento de burbuja.

**Ordenación por mezcla**: también conocido como “MergeSort”, este método es otro ejemplo del paradigma “divide y vencerás”. El principio de clasificación es el siguiente:

1. Dividimos la lista para ordenarla en dos mitades (tomando, por ejemplo, un elemento de dos para cada una de las listas).
2. Clasificamos cada uno de ellos.
3. Fusionamos las dos mitades obtenidas para reconstituir la lista ordenada.

**Ordenamiento Shell**: la ordenación de Shell pertenece a los métodos de clasificación avanzados, nombrado así en honor del ingeniero y matemático estadounidense Donald Shell que la propuso en 1959.

Este método utiliza una segmentación entre los datos. Funciona comparando elementos que estén distantes; la distancia entre comparaciones decrece conforme el algoritmo se ejecuta hasta la última fase, en la cual se comparan los elementos adyacentes, por esta razón se le llama ordenación por disminución de incrementos.

La ordenación de Shell usa una secuencia, h1, h2, . . ., ht, conocida como la secuencia de incrementos. Al principio de todo proceso, se fija una secuencia decreciente de incrementos. Cualquier secuencia funcionará en tanto que empiece con un incremento grande, pero menor al tamaño del arreglo de los datos a ordenar, y que el último valor de dicha secuencia sea 1. Clasificación máxima

**Ordenación por montículos**: este método de clasificación (también llamado “heapsort” u "ordenación de Williams") se basa en la noción de montón. Un montón es un árbol binario perfecto, de modo que la información contenida en cualquier nodo es mayor que la contenida en sus hijos. El método de ordenación del montón tiene dos pasos:

1. Construcción por inserciones sucesivas de un montón del vector a clasificar.
2. Reemplazamos la raíz, que es necesariamente el elemento más grande de la matriz, por su hijo más grande. La raíz se traslada al lugar de la última hoja del árbol y se reposiciona. Repetimos con la nueva raíz tantas veces como nodos haya en el árbol.

**Ordenamiento rápido o Quicksort**: El ordenamiento rápido (también llamado ordenamiento de Hoare o quicksort en inglés) es un algoritmo creado por el científico británico en computación Tony Hoare y basado en la técnica de divide y vencerás.

El algoritmo fundamental es el siguiente:

1. Elegir un elemento de la lista de elementos a ordenar, al que llamaremos pivote.
2. Resituar los demás elementos de la lista a cada lado del pivote, de manera que a un lado queden todos los menores que él, y al otro los mayores. En este momento, el pivote ocupa exactamente el lugar que le corresponderá en la lista ordenada.
3. La lista queda separada en dos sublistas, una formada por los elementos a la izquierda del pivote, y otra por los elementos a su derecha.
4. Repetir este proceso de forma recursiva para cada sublista mientras éstas contengan más de un elemento. Una vez terminado este proceso todos los elementos estarán ordenados.

Como se puede suponer, la eficiencia del algoritmo depende de la posición en la que termine el pivote elegido.

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