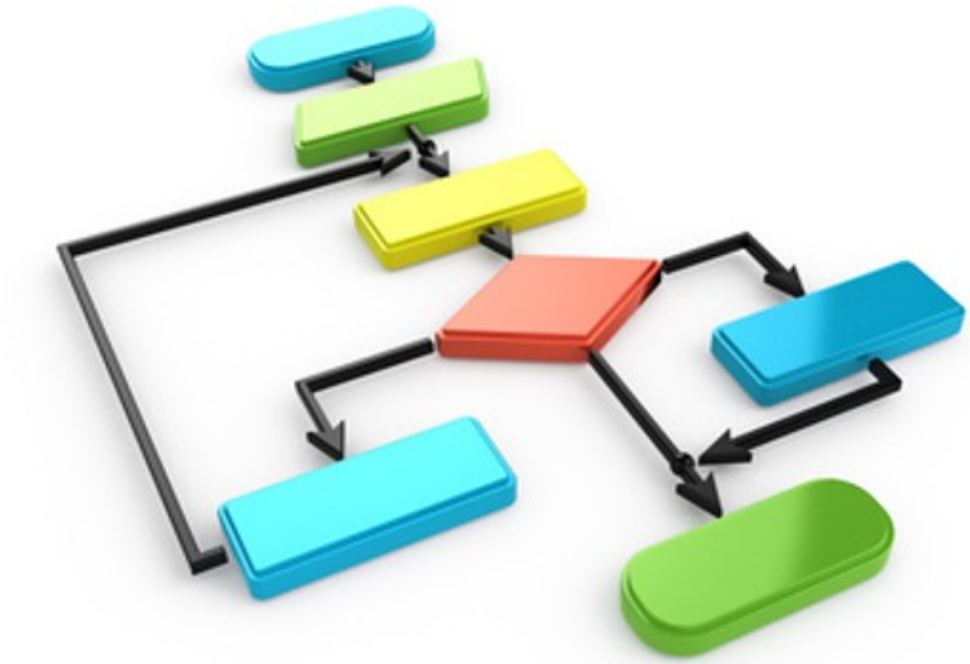
Algortimos y estructuras de datos avanzadas

Informe Métodos de Ordenación



Liam J. O'kelly Herrero

Durante el desarrollo de la Práctica 5 relacionada con los ***Métodos de Ordenación***, hemos trabajado con diferentes tipos de algortimos con el fin de ver cómo trabaja cada uno y su grado de optimalidad. Dichos algoritmos son:

-Inserción
-Quicksort
-Heapsort
-Burbuja
-Shellsort

A continuación, pasaremos a explicar el funcionamiento de cada uno de los algoritmos.

Inserción

El ordenamiento por inserción es una manera muy natural de ordenación. Este algortimo trabaja con conjuntos de valores. Es muy sencillo de entender, pero necesita muchas iteraciones

El algoritmo funciona de la siguiente manera.

1º De inicio tomamos el primer valor de la lista a ordenar, que llamaremos “k”.

2º Ahora comparamos ese valor con el valor que está en la posición k+1.

3º Si el valor en la posición k+1 es menor al que está en la posición k, lo movemos a la izquierda. Ahora comparamos con el valor en la posición k-1 y así sucesivamente hasta que el valor k+1 sea mayor que el valor con el que comparamos

4º Si el valor en la posición k+1 es mayor al que está en la posición k, lo dejamos donde está y lo añadimos al conjunto de valores en el que está k.

Quicksort

Al igual que el algortimo Mergesort, este algortimo también utiliza divide y vencerás, por lo que es un algoritmo recursivo. La forma en que quicksort usa divide y vencerás es un poco diferente de cómo lo hace el Mergesort. En Quicksort, todo el trabajo real ocurre en el paso de división. De hecho, el paso de fusión quicksort no hace absolutamente nada.

El Algortimo funciona de la siguiente manera.

1º De inicio establecemos el último elemento de la lista a ordenar, como el “pivote”.

2º Ahora ponemos el índice “izq” en el primer valor de la lista y el índice “der” en el último valor de la lista (sin contar el pivote, en realidad es el penúltimo).

3º Cuando el índice “izq” sea mayor que el pivote y el índice “der” sea menor que el pivote, se intercambian los elementos en esas posiciones. En otro caso, simplemente avanzamos los índices.

4º Repetir esto hasta que se crucen los índices.

5º El punto en que se cruzan los índices es la posición adecuada para colocar el pivote, porque sabemos que a un lado los elementos son todos menores y al otro son todos mayores (o habrían sido intercambiados).

Heapsort

Es un método de ordenamiento basado con comparación. Usa el montículo (Heap) como estructura de datos, el cual representa un arbol. Mas lento que otros métodos, pero mas eficaz en escenarios mas rigurosos. Se defino como No Recursivo y No Estable.

Funcionamiento paso a paso:

1º Nos colocamos en la primera posición de la lista y este será nuestro nodo padre.

2º Los siguientes dos valores serán sus dós hijos.

3º Los hijos serán a su vez padres de otros dos valores cada uno. Esos dos nuevos hijos serán tomados por orden de la lista de valores, sin contar los elementos que ya son hijos de otros.

4º Cuando tengamos todos los valores colocados en el árbol, tenemos que comprobar que todos los padres sean mayores que sus dos hijos. Para ello, empezaremos por los últimos hijos que hemos ido creando e iremos subiendo en órden inverso. En caso de que algún hijo sea mayor que el padre, los intercambiamos.

5º Una vez hecho eso, el padre de todo el árbol, será el mayor elemento de la lista.

6º Ahora intercambiaremos las posiciones del último valor (el último hijo del árbol) con el padre del árbol, de tal forma que el padre pasará a estar al final de la lista (en otras palabras, el mayor valor ya está al final)

7º El siguiente paso es reducir el tamaño de la lista en 1, ya que ese último valor ya está colocado en su sitio.

8º Volvemos al paso 4

9º Cuando el tamaño de la lista sea igual a 1, la lista ya estará ordenada

Burbuja

La **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.

El funcionamiento paso a paso es el siguiente:

1º Nos colocamos en la primera posición

2º Se compara el valor de nuestra posición con el siguiente.

3º Si nuestro valor es menor que el segundo, los dejamos como están. Si es mayor al segundo, los intercambiamos.

4º Ahora nos movemos a la siguiente posición y repetimos el paso 3.

5º Cuando lleguemos al final de la lista en el primer ciclo, tendremos ya posicionado el mayor valor en la última posición. Ese valor ya no lo recorreremos más ya que está en si sitio.

6º Ahora volvemos a empezar desde el principio de la lista de valores a ordenar y volvemos al paso 1.

Shellsort

La implementación original del Shellsort, requiere $O(n^2)$ comparaciones e intercambios en el peor caso. La mejora presentada en el libro de V. Pratt tiene un rendimiento de $O(n \log^2 n)$ en el peor caso. Esto es mejor que las $O(n^2)$ comparaciones requeridas por algoritmos simples pero peor que el óptimo $O(n \log n)$.

El funcionamiento paso por paso es el siguiente:

1º Lo primero que es necesario es establecer un valor gap, que será la mitad del tamaño.

2º Nos posicionamos en una posición a distancia gap del primer valor de la lista de elementos a ordenar.

3º Compararemos en bucle el valor que está a la distancia “gap” de nuestra posición, por la izquierda, con el valor donde estamos. Si el valor a distancia “gap” es mayor al de la posición en la que nos encontramos, los intercambiamos. Esto lo haremos hasta que el elemento a distancia “gap” sea menor que en el que nos encontramos.

4º Ahora nos movemos una posición a la derecha y volvemos a repetir el paso 3 hasta que hayamos comparado con el último valor de la lista a elementos a ordenar.

5º Ahora que ya hemos llegado a la última posición, dividimos el valor “gap” a la mitad y vamos al paso 2.

6º Cuando el valor “gap” sea 1 y lleguemos a comparar con el último valor de la lista a ordenar, ya estará ordenada

Conclusión final

Después de haber acabado la práctica y leyendo información en internet, he llegado a la conclusión obvia de que el algortimo más óptimo es el Quicksort. El resto de los algortimos consumen más iteraciones para ordenar listas de valores del mismo tamaño. Eso es debido a que ordenan de una forma más “humana”. Es decir, vemos por ejemplo el método de ordenamiento Burbuja y nos damos cuenta en seguida del mótivo de que no sea óptimo. Está comparando cada valor con todos los demás para ordenarlos.