SOLUCIONES ALGORÍTMICAS QUE REQUIERAN LA UTILIZACIÓN DE LOS MÉTODOS DE ORDENAMIENTO Y BÚSQUEDA

Desarrollo temático

Tabla de contenido

De clic en los siguientes enlaces para acceder a un contenido específico.

Métodos de ordenación

En el siguiente audio encontrarás la explicación del concepto de métodos de ordenación. Te invitamos a escuchar atentamente y a tomar notas de los aspectos más importantes que allí se exponen: métodos ordenación. Una vez finalizada la audición, puedes iniciar con el estudio de los cuatro métodos que se explican a continuación.

Ordenación por burbuja

Es el más conocido y popular entre estudiantes y aprendices de programación, por su facilidad de comprender y programar. Así mismo, es importante aprender que es el menos eficiente y por ello, normalmente, se aprende su técnica pero no suele utilizarse.

Algoritmo

En el caso de un array (lista) con n elementos, la ordenación por burbuja requiere hasta n-1 pasadas. Por cada pasada se comparan elementos adyacentes y se intercambian sus valores cuando el primer elemento es mayor que el segundo elemento. Al final de cada pasada, el elemento mayor ha "burbujeado" hasta la cima de la sub-lista actual

- En la pasada 0 se comparan elementos adyacentes
  (A[0],A[1]),(A[1],A[2]),(A[2],A[3]),...(A[n-2],A[n-1])
  Se realizan n-1 comparaciones, por cada pareja (A[i],A[i+1]) se intercambian los valores si
  A[i+1] < A[i]. Al final de la pasada, el elemento mayor de la lista está situado en A[n-1].
- En la pasada 1 se realizan las mismas comparaciones e intercambios, terminando con el elemento de segundo mayor valor en A[n-2].
- El proceso termina con la pasada n-1, en la que el elemento más pequeño se almacena en A[0] .
- El algoritmo tiene una mejora inmediata, el proceso de ordenación puede terminar en la pasada n-1, o bien antes. Si en un una pasada no se produce intercambio
  Veamos un ejemplo para esta ordenación en el sitio web de Algoritmia, en el encontramos definición y uso: http://www.algoritmia.net/articles.php?id=31

Ordenación por inserción directa

El método de ordenación por inserción es muy parecido al que se utiliza normalmente para ordenar cartas (una baraja) por orden alfabético. Consiste en insertar un nombre en su posición correcta dentro de una lista o archivo que ya está ordenado. Veamos un ejemplo

A = 60, 25, 45,70, 30.

- Comienzo con 60.
- Insertamos 25, el cual toma la posición 0 y 60 se mueve a posición 1; tenemos A = 25, 60
- Ahora insertamos 45, el cual toma la posición 1 y se mueve 60 a posición 2, tenemos A=25,45,60
- Insertamos 70, este elemento está bien ordenado A=25,45,60,70
- Tomamos 30, se ubica en posición 1, y los números de la sub-lista se corre hacia la derecha en un lugar quedando así A=25,30,45,60,70

Algoritmo

1. El primer elemento A[0] está ordenado, la lista inicial tiene un elemento.

2. Se inserta A[1] en la posición correcta delante o detrás de A[0], dependiendo de que sea menor o mayor.

3. Por cada bucle (desde i = 1 hasta n-1) se explora la sublista A[i-1] . . A[0]
buscando la posición correcta de inserción; a la vez se mueve hacia abajo (a la derecha en la sublista) una posición todos los elementos mayores que el elemento a insertar A[i], para dejar vacía esa posición.

4. Insertar el elemento en la posición correcta.

Tomado de: http://youtu.be/o5Zc37absvg

Ordenación por selección

Consiste en la búsqueda del menor elemento del arreglo y en colocarlo en la primera posicion. Luego se busca el segundo elemento más pequeño del arreglo y se coloca en la segunda posición. El proceso continua hasta que todos los elementos del arreglo son ordenados. El método se basa en:

1) Seleccionar el menor elemento del arreglo
2) Intercambiar dicho elemento con el primero
3) Repetir los pasos anteriores con los (n-1),(n-2) elementos y así sucesivamente, realizando de manera adecuada el intercambio, hasta que solo quede el elemento mayor.

Veamos un ejemplo: A=50,30,40,70,33

Paso 0: Seleccionar el menor valor = 30 colocarlo en la posición a[0], tenemos
A=30, 50,40,70,33

Paso 1: Seleccionamos el siguiente valor más pequeño=33, colocamos en la posición a[1], tenemos: A=30, 33,40,70,50

Paso 2: Seleccionamos el siguiente valor mas pequeño=40, colocamos en la posición a[2],
tenemos: A=30, 33,40, 70,50

Paso 3: Seleccionamos 50, lo colocamos en la posición a[3], tenemos: A=30, 33,40, 50,70
La lista se encuentra ordenada

Veamos un ejemplo para esta ordenación nuevamente con la ayuda de una baraja:

Tomado de: http://youtu.be/NqjLlSEKldI

Ordenación Quicksort (divide y vencerás)

Se basa en la división en particiones de la lista a ordenar. El método es, posiblemente, el más pequeño en código, más rápido, más elegante y más interesante y eficiente de los algoritmos conocidos de ordenación.

El método se basa en dividir los n elementos de la lista a ordenar en dos partes o particiones separadas por un elemento (pivote)m tenemos entonces: una partición izquierda, un elemento central denominado pivote o elemento de partición y una partición derecha. La partición o división se hace de tal forma que todos los elementos de la primera sub-lista (partición izquierda) son menores que todos los elementos de la segunda sub-lista (partición derecha).

Las dos sub-listas se ordenan independientemente. Para dividir la lista en particiones (sub-listas) se elige uno de los elementos de la lista y se utiliza como pivote o elemento de partición. Si se elige una lista cualquiera con los elementos en orden aleatorio, se puede elegir cualquier elemento de la lista como pivote, por ejemplo el primer elemento de la lista.

Si la lista tiene algún orden parcial, que se conoce, se puede tomar otra decisión para el pivote. Idealmente, el pivote se debe elegir de modo que divida la lista exactamente por la mitad, de acuerdo al tamaño relativo de las claves. Por ejemplo, si se tiene una lista de enteros de 1 a 10, 5 o 6 serían pivotes ideales, mientras que 1 o 10 serían elecciones "pobres" de pivotes.

Una vez que el pivote ha sido elegido, se utiliza para ordenar el resto de la lista en dos sub-listas: una tiene todas las claves menores que el pivote y la otra en la que todos los

elementos (claves) son mayores o iguales que el pivote (o al revés). Estas dos listas parciales se ordenan recursivamente utilizando el mismo algoritmo; es decir, se llama sucesivamente al propio algoritmo quicksort. La lista final ordenada se consigue concatenando la primera sub.lista, el pivote y la segunda lista, en ese orden, en una única lista. La primera etapa de quicksort es la división o "particionado" recursivo de la lista hasta que todas las sub-listas constan de sólo un elemento. 1

Ahora veamos un ejemplo para esta ordenación teniendo en cuenta valores de diferentes billetes:

Tomado de: http://youtu.be/2yPorr5WwKo

---

1. Tomado de Joyanes, L (2001), *Programación en C Metodología, algoritmos y estructura de datos.* Capítulo 10: Algoritmos de ordenación y búsqueda. Programación en C: Metodología, algoritmos y estructura de datos. Editorial Mc Graw Hill.