Métodos de ordenamiento

ESTRUCTURAS DE DATOS Y ALGORITMIA

Otero Cabrero Moises 22310402

4P

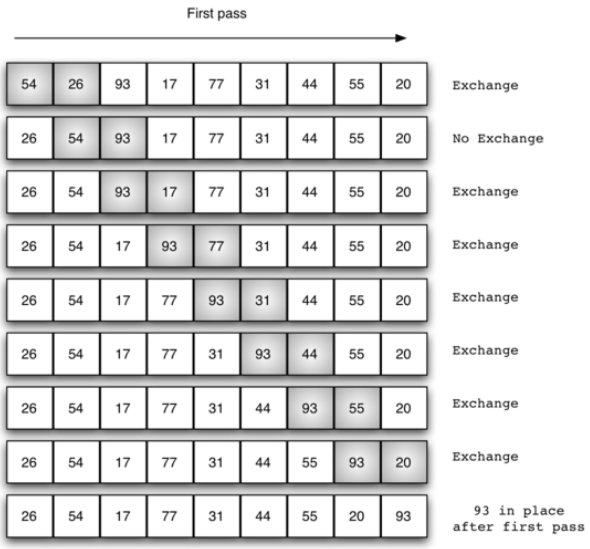
Contents

Método burbuja

El ordenamiento burbuja hace múltiples pasadas a lo largo de una lista. Compara los ítems adyacentes e intercambia los que no están en orden. Cada pasada a lo largo de la lista ubica el siguiente valor más grande en su lugar apropiado. En esencia, cada ítem "burbujea" hasta el lugar al que pertenece.

Si hay n ítems en la lista, entonces hay n−1 parejas de ítems que deben compararse en la primera pasada. Es importante tener en cuenta que, una vez que el valor más grande de la lista es parte de una pareja, éste avanzará continuamente hasta que la pasada se complete.

Al comienzo de la segunda pasada, el valor más grande ya está en su lugar. Quedan n−1 ítems por ordenar, lo que significa que habrá n−2 parejas. Puesto que cada pasada ubica al siguiente valor mayor en su lugar, el número total de pasadas necesarias será n−1. Después de completar la pasada n−1, el ítem más pequeño debe estar en la posición correcta sin requerir procesamiento adicional. La función recibe la lista como un parámetro, y lo modifica intercambiando ítems según sea necesario.



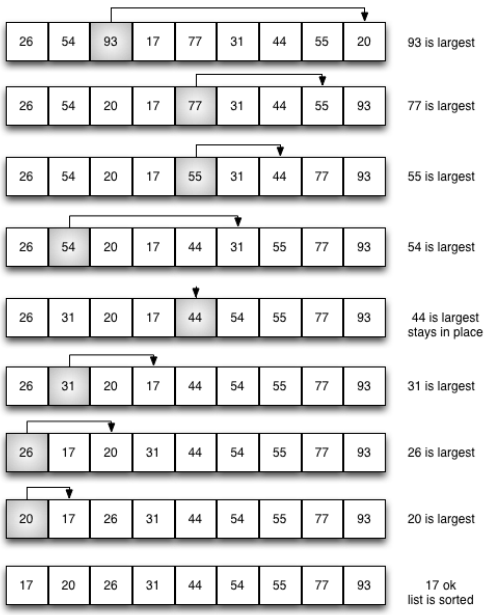
Ejemplo

```
// Repetimos el procesos n-1 veces para un arreglo de tamaño n.
        for(int i=0; i < vec.length-1; i++){
         // En cada iteración llegamos hasta n-1-i ya que hemos ordenado i enteros
         //en las i iteraciones pasadas.
               for(int j=0; j < (vec.length-1-i); j++){
                   //Comparamos e intercambiamos si se cumple la condición
                    if(vec[j] > vec[j+1]){
                            aux=vec[j];
                            vec[j]=vec[j+1];
                            vec[j+1]=aux;
                     }
               }
          }
```

Método selección

El algoritmo de ordenación por selección recorre todo el arreglo desde la primera posición buscando el menor elemento de todos. Cuando termina, lo reemplaza por el elemento de la primera posición y repite todo el proceso con el segundo elemento y así sucesivamente.

Para hacer esto, un ordenamiento por selección busca el valor mayor a medida que hace una pasada y, después de completar la pasada, lo pone en la ubicación correcta. Al igual que con un ordenamiento burbuja, después de la primera pasada, el ítem mayor está en la ubicación correcta. Después de la segunda pasada, el siguiente mayor está en su ubicación. Este proceso continúa y requiere n−1 pasadas para ordenar los n ítems, ya que el ítem final debe estar en su lugar después de la (n−1) ésima pasada.



Ejemplo

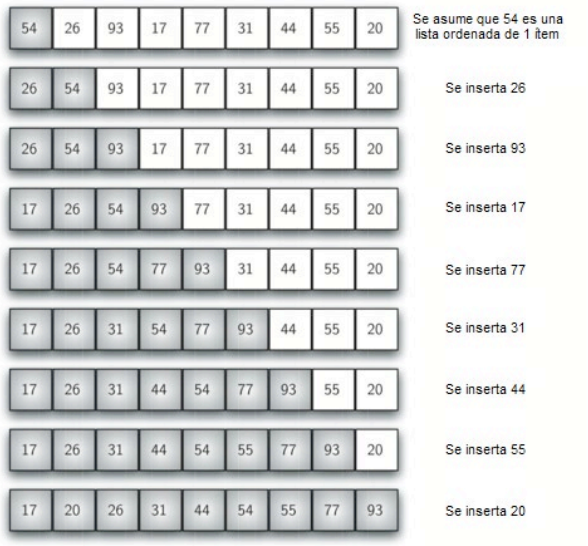
```
//se guardan los valores del arreglo
int N=vec.length;
        for(int i=0;i<N;i++){
//se declara una variable de la posición del numero actual
                int posMenor=i;

                for(int j=i+1;j<N;j++){
// se recorre el arreglo hasta que se encuenta un valor menor
                        if(vec[j]<vec[posMenor])
// se iguala la variable al valor menor
                                posMenor=j;
                }
//Cambia las posiones
        if(posMenor!=i){
// se declara variable temporal y se iguala a la posición del numero donde estamos
                int temp=vec[i];
// pone el numero de la posición menor en el lugar donde corresponde
                vec[i]=vec[posMenor];
// ponemos el numero de la posición original en la posición anterior
                vec[posMenor]=temp;
        }
}
```

Método inserción

Este algoritmo ordena los primeros dos elementos del vector, a continuación, toma el tercer elemento y lo ordena en la posición correspondiente. Cuando termina de ubicar un elemento en su posición correcta, continúa con el siguiente elemento, hasta hacer este proceso con todos los elementos del vector.

El ordenamiento por inserción, aunque sigue siendo O(n2), funciona de una manera ligeramente diferente. Siempre mantiene una sub lista ordenada en las posiciones inferiores de la lista. Cada ítem nuevo se "inserta" de vuelta en la sub lista previa de manera que la sub lista ordenada sea un ítem más largo.



Comenzamos asumiendo que una lista con un ítem (posición 0) ya está ordenada. En cada pasada, una para cada ítem desde 1 hasta n−1, el ítem actual se comparará contra los de la sub lista ya ordenada. A medida que revisamos en la sub lista ya ordenada, desplazamos a la derecha los ítems que sean mayores. Cuando llegamos a un ítem menor o al final de la sub lista, se puede insertar el ítem actual.

Ejemplo

```
//se declara el la variable del tamaño del arreglo total
final int N=vec.length;
for(int i=1;i<N;i++){
        //se crea una segunda variable que se iguala a la posicion
        int j=i;
// se crea un while que selecciona a las parejas de la posicion actual y la anterior
        while(j>0&&vec[j]<vec[j-1]){
                // crea la variablee temporal y lo iguala a la segunda posicion
                int temp=vec[j];
                // la variable de la posicion actual se cambia por la anterior
                vec[j]=vec[j-1];
                // la variable de la posicion anterior se cambia por la actual
                vec[j-1]=temp;
                //se resta un valor a j para seguir acomodando
                j--;
        }
}
```