InsertionSort.

InsertionSort (ordenamiento por inserción) es la manera natural que el humano ordena, usado comúnmente en el ordenamiento de datos cuantitativos y cualitativos prioritarios.

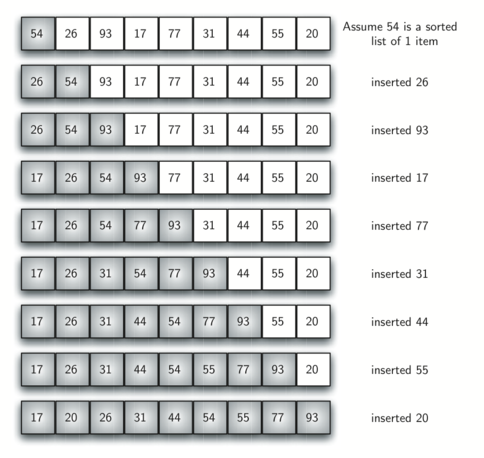
Análisis.

Ordenamiento, que requiere O(n2) operaciones, para cumplir su objetivo en listas o arreglos de n-elementos. Basando en la instrucción básica de:

* El tener una lista de j números ordenados, se toma el elemento j+1(el elemento nuevo a insertar)
* Se compara iterativamente con los elementos contenidos en la lista si j+1 es menor
  + Si el elemento es menor, el elemento consultado se desplaza a la derecha
    - Si el elemento no es menor se inserta en la posición +1 de la que se consulta (con los mayores desplazados a la derecha si es el caso)
    - Si no se encuentra menor, el elemento j+1 se inserta al principio, dado que el elemento es el menor a los que ya están en la lista desplazados

Diseño.

Con el análisis del funcionamiento del método de ordenamientos por inserción, se diseñó el funcionamiento de la inserción y su ordenamiento. El cual consistió en separar la inserción del ordenamiento principalmente, dado que para este tipo de ordenamiento es válido la inserción de objetos ya sean primitivos o no, de manera individual como en un arreglo y/o lista no ordenado.



Forma de ordenamiento general en base a comparaciones de cualidades mayores o menores en importancia, siendo en el ordenamiento de números el orden natural de menos a mayor.

Orden ejecutado en listas, para su mejor adaptabilidad y uso de recursos.

Usando ArrayList<E>() y sus métodos de consulta y modificación, además del método compareTo() para comparar cualidades de una clase, definida en Alumno para esta actividad.

Implementación.

Luego de conocer el funcionamiento y cualidades que se le otorgaría al algoritmo, se implementó la clase API de “***InsertionSort.java***”, que contenía en si los métodos de inserción de primitivos enteros “***InsertarListaEnteros(ArrayList<Integer> lista, int valor)***”, la cual cumple la función de insertar directamente en la lista entregada para su ordenamiento posterior, llamado dentro del método.

Código del método de insercion:

if(!lista.contains(valor)) {

lista.add(valor);

}else {

System.out.print("(el valor existe) ");

}

if(lista.size() > 1) {

ordenarListaEnteros(lista);

}

El método de ordenamiento posterior, es de uso público, es decir que su función es para ordenar en la inserción individual de datos, como inserción masiva a través de lista, método llamado “***ordenarListaEnteros(ArrayList<Integer> lista)***”, que iteraba la lista entregada, comparando y desplazando el contenido en esta para la inserción o remplazo en la posición óptima.

Código del método de orden:

int tam = lista.size();

for(int i = 1;i < tam;i++) {

int aux = lista.get(i);

for(int j = i;j > 0;j--) {

if(aux < lista.get(j-1)) {

lista.set(j, lista.get(j-1));

lista.set(j-1,aux);

imprimirListaEnteros(lista);

}else {

break;

}

}

}

Y el ultimo método implementado fue la Impresión de datos de las lista, tanto de forma ordenada luego de transcurrido los procesos anteriores de ordenamiento, pero además implementada para la muestra de la condición de la lista mientras se ordenaba iterativamente y mostrar su función de forma visual.

Código del método de impresión de enteros:

Public static void imprimirListaEnteros(ArrayList<Integer> lista) {

System.out.println(lista);

}

Codigo metodo de impresión de Clase Alumno:

public static void imprimirListaAlumno(ArrayList<Alumno> lista) {

int tam = lista.size();

for(int i = 0;i < tam;i++) {

System.out.print(" Nombre: "+lista.get(i).getNombre());

System.out.print(" Apellido: "+lista.get(i).getApellido());

System.out.print(" Edad: "+lista.get(i).getEdad());

System.out.println(" Nota: "+lista.get(i).getNota());

}

}

Estos ordenamientos básicos de enteros, luego fueron usados para los ordenamientos particulares de la clase alumno, con respecto a sus distintas variables, implementando de forma particular cada ordenamiento para su uso único en la opción requerida para ordenar, cualidades como nombre, apellidos, edad y notas, variables de la clase “***Alumno.java***”