1. Un algoritmo de ordenación se dice que es estable cuando no altera el orden relativo de las posiciones de los elementos repetidos,

(4, 1) (3, 7) (3, 1) (5, 6)

En este caso, dos resultados diferentes son posibles, uno de los cuales mantiene un orden relativo de registros con claves iguales, y una en la que no:

**(3, 7) (3, 1)** (4, 1) (5, 6) (orden mantenido)

**(3, 1) (3, 7)** (4, 1) (5, 6) (orden cambiado)

Su utilidad no puede verse si se ordena listas con una sola “key”, el problema ya esta cuando se tiene mas de una “key”, donde en el ejemplo de arriba no estaría totalmente ordenado, es decir, su utilidad es para listas donde se dan mas de una clave por el cual ordenar



|  |  |
| --- | --- |
| InsertSort | Estable |
| MergeSort | Estable, pero depende de la implementación |
| HeapSort | Inestable |
| QuickSort | Inestable |
| BubbleSort | Estable |
| ShellSort | Inestable |
| RadixSort | Estable,pero depende de la implementacion |

1. El algoritmo es inestable porque modifica el orden relativo de la lista, al hacer el intercambio (swap) ocurre este mal

-Una forma de volverlo estable es haciendo que en lugar de intercambiar las posiciones, el arreglo antes de la posición vacia a “intercambiar” se corra por la derecha, de esta forma se mantendrá el orden relativo ya que no estamos cambiando el espacio de una variable al intercambiarlo, al hacerlo correr un espacio hacia la derecha los ordenes se mantendrán.

**Observación**

* Trabajo hecho con la participacion y cooperacion de G03