Laboratorio No. 8

Algoritmos de Ordenamiento (Elementales y Complejos)

Introducción

En este laboratorio el/la estudiante deberá trabajar con algoritmos elementales y complejos de ordenamiento sobre arreglos simples, como: burbuja, burbuja mejorada, selección lineal con intercambio (selección directa), selección línea con conteo (counting sort), quicksort, radixsort, mergesort, shellsort. Además, deberá utilizar algoritmos de búsqueda binaria (iterativa y recursiva) para localizar elementos dentro de los arreglos simples.

Objetivos

Al finalizar este laboratorio, el/la estudiante deberá ser capaz de:

a. Entender el funcionamiento de los diferentes tipos de algoritmos elementales y complejos de ordenamiento
b. Complementar el uso de algoritmos elementales de ordenamiento con los algoritmos de búsqueda sobre arreglos simples
c. Determinar los mejores algoritmos de ordenamiento
d. Aplicar conocimientos discutidos en clase

Contexto

1. Trabaje con un modelo de n capas (domain, controller, test, util).
   Cree un nuevo proyecto llamado “Laboratory8” utilizando la tecnología javaFX, la cual permitirá trabajar en un entorno gráfico.
2. Defina una clase llamada “Elementary” e implemente los siguientes algoritmos elementales de ordenamiento: bubbleSort, improvedBubbleSort, selectionSort, countingSort.
3. Defina una clase llamada “Complex” e implemente los siguientes algoritmos complejos de ordenamiento: quickSort, radixSort, mergeSort, shellSort.

4. Compruebe el funcionamiento de la clase “Elementary” a través de una testing class “ElementaryTest”, de la siguiente forma:
   a. Cree, instancie y llene 6 arreglos: a, b, c, d, e, f, y llénelos con 10 000 valores enteros positivos.
   b. Muestre el contenido de los primeros 200 elementos de cada arreglo por consola.
   c. Cree e instancie un objeto tipo Elementary llamado “elementary”.
   d. Cree un método llamado **isSorted(a[ ])** que devuelva true si el arreglo indicado está ordenado.
   e. Utilice el método anterior, compruebe y ordene cada arreglo indicado de acuerdo con los siguientes criterios:
      i. Si a no está ordenado, proceda a ordenarlo utilizando el algoritmo “improvedBubbleSort”.
      ii. Si b no está ordenado, proceda a ordenarlo utilizando el algoritmo “bubbleSort”.
      iii. Si c no está ordenado, proceda a ordenarlo utilizando el algoritmo “selectionSort”.
      iv. Si d no está ordenado, proceda a ordenarlo utilizando el algoritmo “improvedBubbleSort”.
      v. Si e y f no están ordenados, proceda a ordenarlos utilizando cualquier algoritmo elemental de ordenamiento.

f. Muestre el contenido de los primeros 200 elementos de cada arreglo ya ordenado por consola.

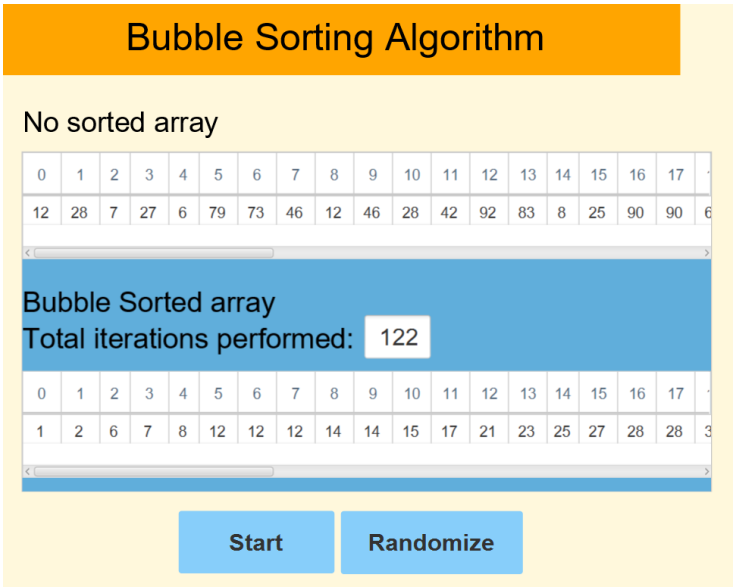
5. Compruebe el funcionamiento de la clase “Complex” a través de una testing class “ComplexTest”, de la siguiente forma:
   a. Cree, instancie y llene 6 arreglos: a, b, c, d, e, f, y llénelos con 20 000 valores enteros positivos.
   b. Muestre el contenido de los primeros 200 elementos de cada arreglo por consola.
   c. Cree e instancie un objeto tipo Complex llamado “complex” y ordene todos los arreglos anteriores, de la siguiente forma: a utilizando el algoritmo quickSort, b utilizando el algoritmo shellSort, c utilizando el algoritmo mergeSort, d utilizando el algoritmo shellSort, e utilizando el algoritmo radixSort, f utilizando el algoritmo mergesort.
   d. Muestre el contenido de cada arreglo ya ordenado por consola (los primeros 100 elementos).

6. Para las pruebas unitarias de las clases Elementary y Complex, cree una clase llamada “Search” e implemente algoritmos de búsqueda binaria (iterativa y recursiva) para buscar 30 elementos en cada uno de los arreglos. Si el elemento existe, deberá mostrar la siguiente leyenda: "The element ["+value+"] exists at position: "+pos. Si el elemento no existe, deberá mostrar la siguiente leyenda: "The element ["+value+"] does not exist in array

7. Utilice la tecnología “javaFX” para crear un entorno gráfico que muestre un menú y permita probar los algoritmos elementales y complejos de ordenamiento, de la siguiente forma:

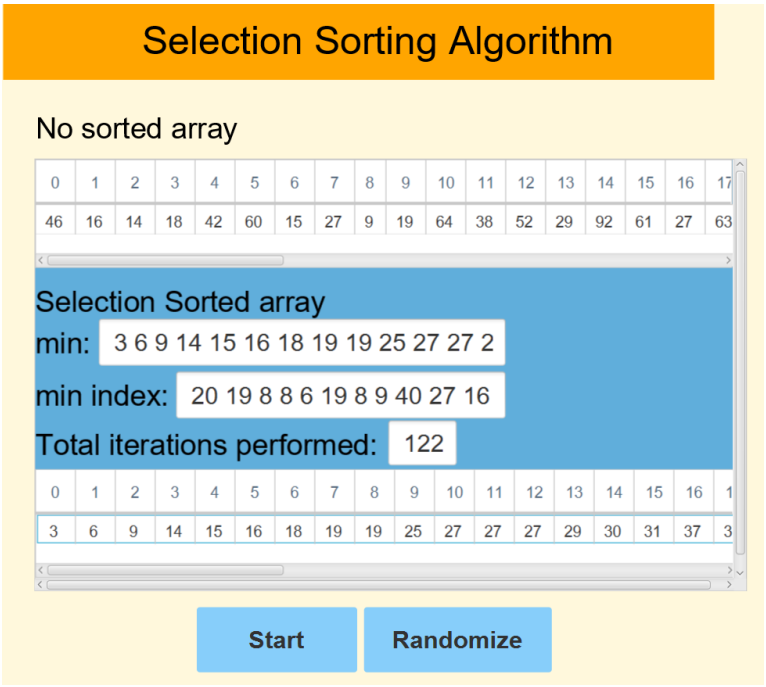
   Elementary Sorting

   a. BubbleSort: Utilice un objeto gráfico tipo tableview para mostrar por pantalla un arreglo de 200 valores numéricos generados en forma aleatoria. Luego, por medio de un botón haga un llamado al método “bubbleSort” para ordenar el arreglo y mostrarlo por pantalla ordenado. Adicionalmente, deberá mostrar el número de iteraciones realizadas durante el proceso de ordenamiento con bubbleSort.

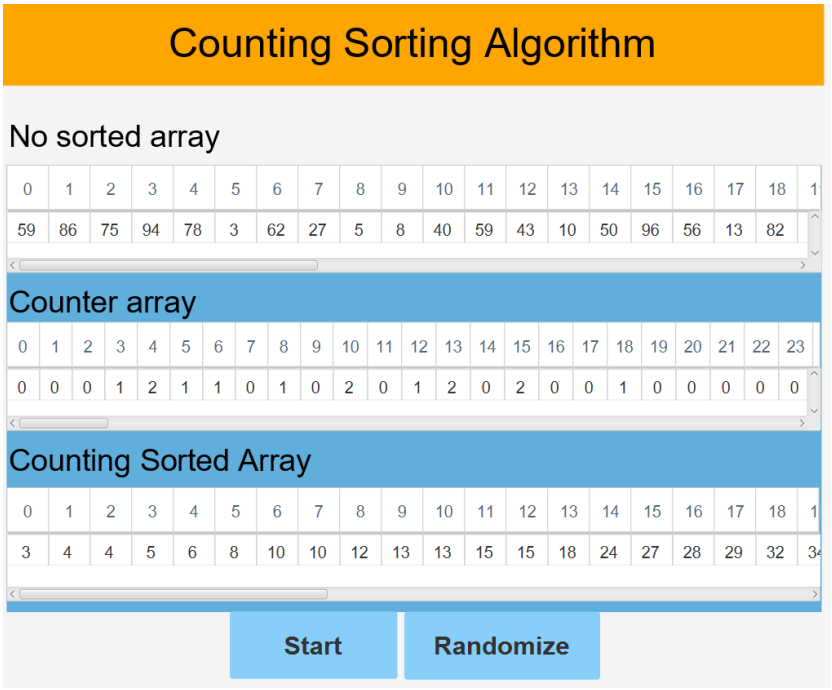


   b. Improved BubbleSort: Similar al caso anterior, utilizando el método “improvedBubbleSort”.

c. SelectionSort: Similar a los anteriores, utilizando el método “selectionSort”. También se deberá mostrar por pantalla todos los valores que van tomando las variables min y min index.

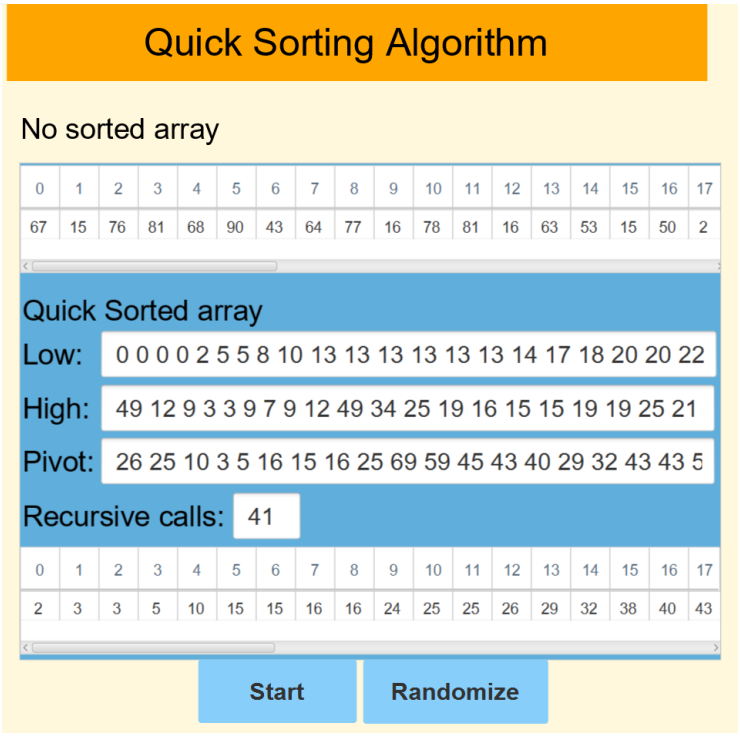


d. CountingSort: Similar al anterior, utilizando el método “countingSort”. En este caso deberá mostrar al arreglo original, el arreglo de conteo “counter” creado y el arreglo ordenado.

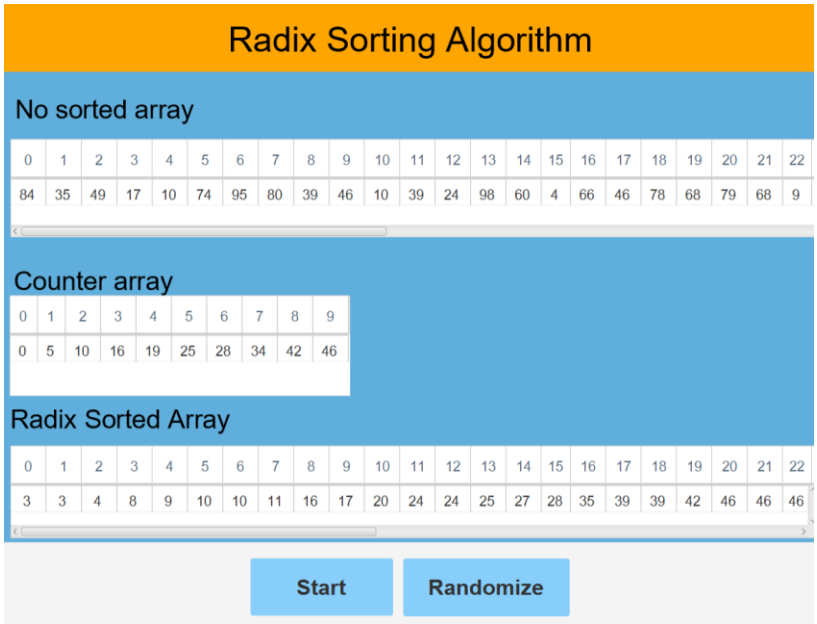


Complex Sorting

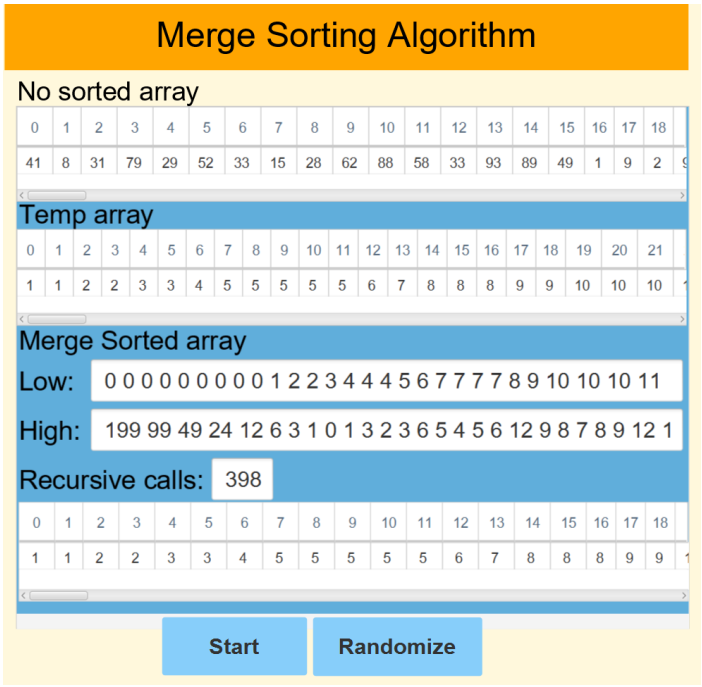
a. QuickSort: Utilice un objeto gráfico tipo tableview para mostrar por pantalla un arreglo de 200 valores numéricos generados en forma aleatoria. Luego, por medio de un botón haga un llamado al método “quickSort” para ordenar el arreglo y mostrarlo por pantalla ordenado. Adicionalmente, deberá mostrar todos los valores que van tomando las variables low, high y pivot, durante el proceso de ordenamiento. Por último, se deberá mostrar el número de llamadas recursivas que realiza el método.



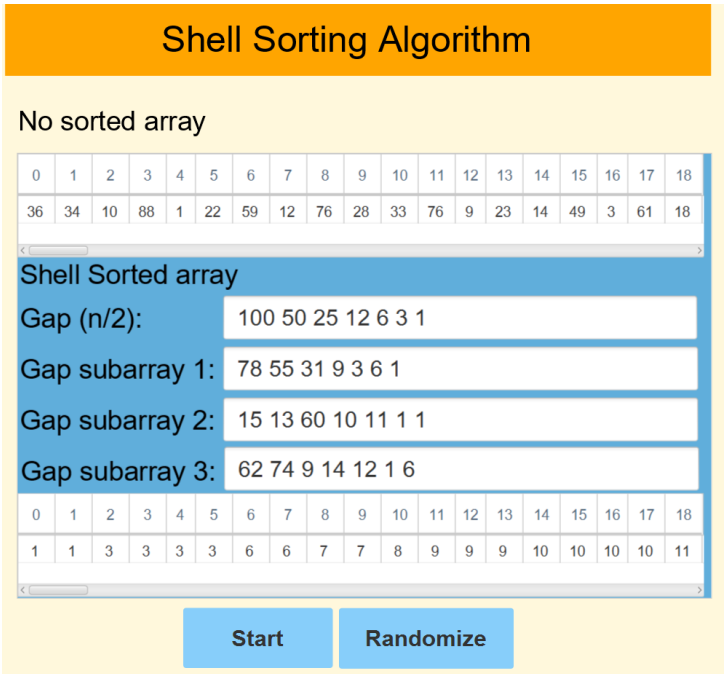
b. RadixSort: Similar al caso anterior, utilizando el método “radixSort”. En este caso deberá mostrar el arreglo original, el arreglo de conteo “count” en la última iteración y el arreglo ordenado.



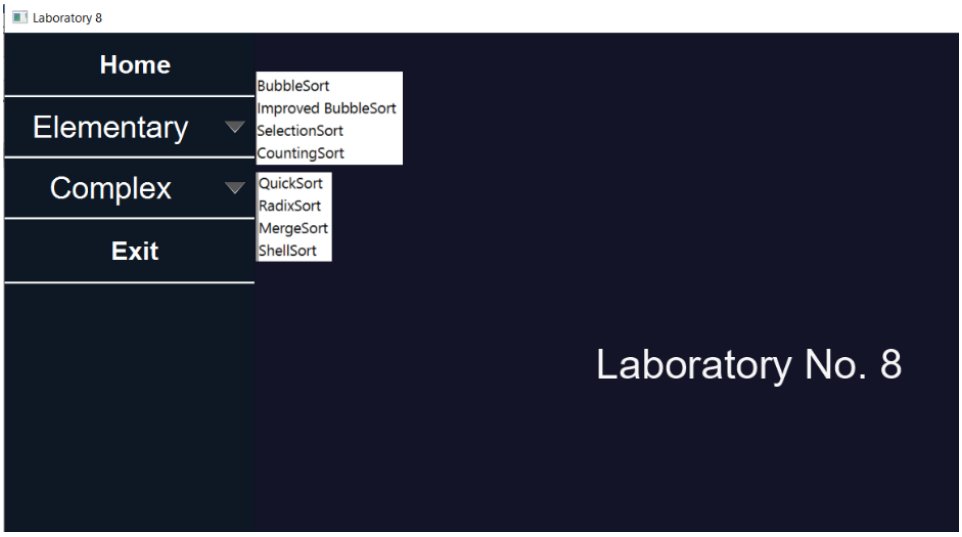
c. MergeSort: Similar a los anteriores, utilizando el método “mergeSort”. También deberá mostrar el arreglo temporal y todos los valores que van tomando las variables low y high, durante el proceso de ordenamiento. Por último, se deberá mostrar el número de llamadas recursivas que realiza el método.



d. ShellSort: Similar a los anteriores, utilizando el método “shellSort”. En este caso deberá mostrar el arreglo original, los valores de gap (k=n/2), para cada gap los valores de los subarreglos y el arreglo ordenado.



Un ejemplo del menú principal es el siguiente:



Resuelva y publique el laboratorio en el entorno del curso de la plataforma de mediación virtual (METICS). Verifique la fecha límite para el envío del informe.
URL: https://mv1.mediacionvirtual.ucr.ac.cr/course/view.php?id=7513