

Algoritmos de ordenamiento

Los algoritmos de ordenamiento sirven para ordenar **listas** de manera rápida y eficiente. Este programa muestra la diferencia entre diferentes métodos.

Información general

En este proyecto se incluyen los siguientes algoritmos: *Selection Sort*, *Insertion Sort*, *Bubble Sort*, *Quick Sort* y *Merge Sort*. Esta solución está implementada en **Java**: lee n elementos y regresa tanto ese arreglo de elementos ordenado como el tiempo que le tomó ordenarlo. De esta manera, es fácil comparar eficiencia entre algoritmos.

Hay muchos recursos en internet donde se explican visualmente estos diferentes métodos; por ejemplo, [Sorting Algorithms Animations](#) de Toptal o [este](#) video de TEDEd en el que se ilustra a partir de cómo ordenar un librero.

En la siguiente figura se muestra la visualización de eficiencia de estos cinco métodos, y además de otros que tienen una mayor dificultad de programación.

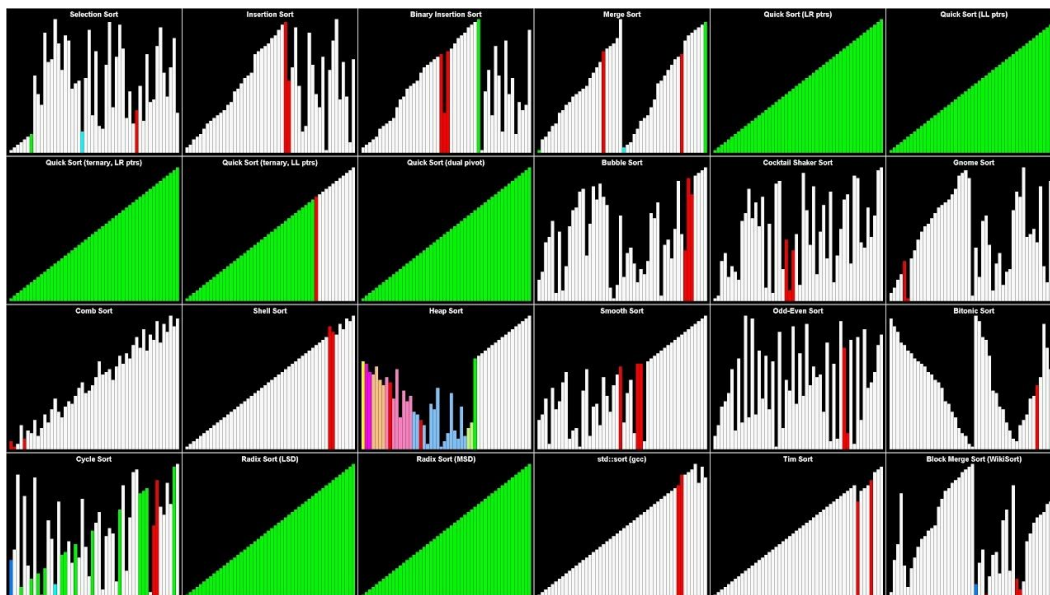


Figura 1: Visualización de 24 algoritmos de ordenamiento.

Setup

El proyecto es un archivo de Java, por lo que puede correrse en cualquier IDE que tenga soporte para Java (e.g. [NetBeans](#), [Eclipse](#)).

Para facilitar usar este programa, es necesario seguir los siguientes pasos:

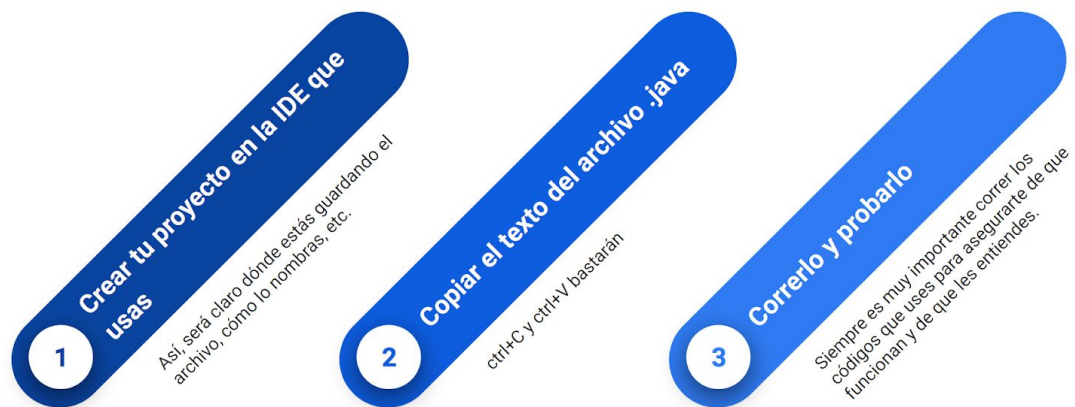


Figura 2: Diagrama de cómo se debe de copiar el programa.

Code Examples

Lo que van a ordenar estos algoritmos son árboles que tienen la estructura que se muestra en el siguiente diagrama, donde cada rectángulo representa a un nodo con información y se quiere tener que cada nodo hijo es menor al nodo padre y a los nodos a su derecha.

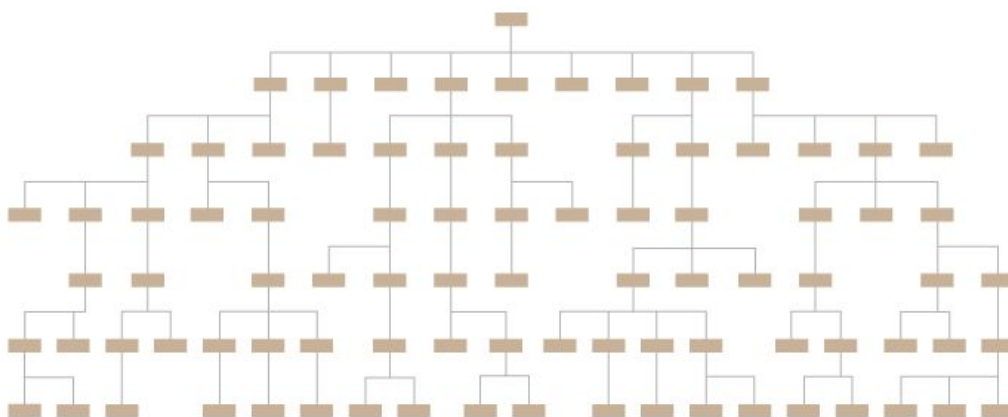


Figura 3: Diagrama de estructura de árbol

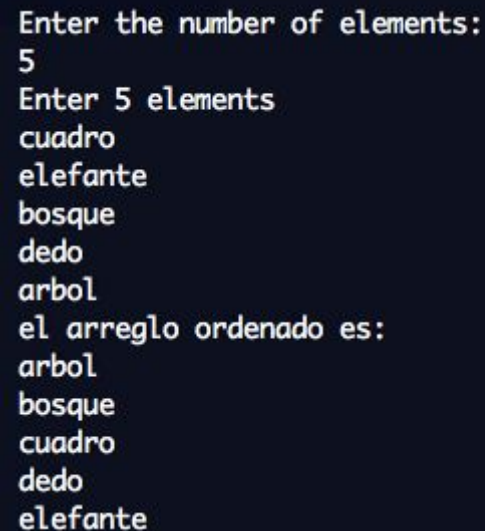
Para lograr representar esta estructura, se hace uso de arreglos en Java.

```
System.out.print("Enter the number of elements : ");
int n = sc.nextInt();
int arr[] = new int[n];
System.out.println("Enter " + n + " elements :");
for (int i = 0; i < n; i++)
    arr[i] = sc.nextInt();
```

después se llama a cada método:

```
selectionSort(arr, 0, n - 1);
insertionSort(arr, 0, n - 1);
bubbleSort(arr, 0, n - 1);
quickSort(arr, 0, n - 1);
mergeSort(arr, 0, n - 1);
```

Así, si, por ejemplo, metemos 5 palabras diferentes, la consola se vería algo así:



```
Enter the number of elements:
5
Enter 5 elements
cuadro
elefante
bosque
dedo
arbol
el arreglo ordenado es:
arbol
bosque
cuadro
dedo
elefante
```

Figura 4: Ejemplo de cómo se vería la consola de salida

Status

Como todo proyecto, nunca está realmente terminado; se pueden hacer mejoras a la eficiencia del uso de memoria. No obstante, el proyecto que aquí se presenta está completo para los 5 algoritmos ya antes mencionados.

Contacto

Código creado por [@salbertt](#) para un curso de Estructura de Datos en el ITAM.
¡Contáctame si tienes dudas o sugerencias!