Proyecto 1.b
Quick Sort

Algoritmos y Estructuras de Datos II - Laboratorio

Docentes: Natalia Bidart, Matías Bordese, Diego Dubois,
Leonardo Rodríguez, David Arch, Maximiliano Bustos.

Objetivos

- Implementar en el lenguaje de programación C el algoritmo de ordenación *Quick sort*.
- Contar la cantidad de comparaciones y swaps que este algoritmo requiere para ordenar un arreglo dado.
- Analizar y comparar los tres algoritmos implementados hasta el momento (*Selection sort*, *Insertion sort*, *Quick sort*), usando como base del análisis las mediciones hechas en cada caso.

Instrucciones generales

La tarea es extender lo hecho en la primera parte del proyecto, agregando una tercera implementación de procedimientos para ordernar un arreglo, esta vez usando el algoritmo de ordenación *quick sort*.

- En el archivo `sort.h` agregar la siguiente signatura para el nuevo procedimiento:

```
void quick_sort(int *a, int length);
/*
  Sort (ascendantly) the array 'a' using the Quick Sort algorithm.
  The array 'a' must have exactly 'length' elements.

*/
```

- Implementar el algoritmo en el archivo `sort.c`, de acuerdo a lo estudiado en el teórico.
- Modificar el menú principal de manera que el usuario pueda elegir el algoritmo *quick sort* (además de las opciones que tenía disponibles anteriormente).
- Imprimir en pantalla la cantidad de operaciones `swap` y de comparaciones que realiza el algoritmo.

- Analizar y comparar los tres algoritmos ya implementados, construyendo una tabla comparativa, para los siguientes casos (todos los archivos de entrada necesarios serán provistos por la cátedra):

  - Para cada algoritmo, recolectar datos de cantidad de comparaciones y swaps para arreglos de 3 largos distintos:
    - $N = 100$
    - $N = 1000$
    - $N = 10000$
  - Para cada N distinto, recolectar datos para arreglos con las siguientes características:
    - ordenado ascendentemente
    - ordenado descendentemente
    - desordenado

  Luego, se deberá presentar una tabla comparativa de la siguiente forma (la columna de *bubble sort* deberá ser llenada sólo si se hizo el punto estrella que lo pedía):

N	Array de entrada	Selection sort		Insertion sort		Quick sort		Bubble sort $\star$	
		Comps	Swaps	Comps	Swaps	Comps	Swaps	Comps	Swaps
100	Ordenado asc								
	Ordenado desc								
	Desordenado								
1000	Ordenado asc								
	Ordenado desc								
	Desordenado								
10000	Ordenado asc								
	Ordenado desc								
	Desordenado								

  Tener en cuenta que el día de la evaluación se les harán preguntas respecto de los datos arriba especificados, buscando conclusiones respecto de la eficiencia de cada algoritmo, para cada caso.

Punto $\star$ 1 *Permitir elegir entre ordenación ascendente y descendente para este nuevo algoritmo.*

Punto $\star$ 2 *La versión de* quick sort *que se vio en el teórico elige siempre la primera posición del arreglo como pivote. Sin embargo, el algoritmo da mejores resultados en la práctica cuando el pivote se elige (pseudo) aleatoriamente. Implementar esa modificación.*

Recordar

- Entrega y evaluación: Martes 3 de Abril.
- Comentar e identar el código apropiadamente, siguiendo el estilo de código ya provisto por la cátedra.