UNIVERSIDAD TECNOLÓGICA DE PANAMÁ CENTRO REGIONAL DE CHIRIQUÍ
FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

CARRERA:
Gestión y Desarrollo de Software

ACTIVIDAD No. 17

LABORATORIO No. 10

"Laboratorio 10"

ASIGNATURA: Estructura de Datos II

DOCENTE:
Profa. Nunehar Mondul

ESTUDIANTE/s:
Jorge Jiménez (4-826-874)
-
-

I SEMESTRE 2025

FECHA:
07/01/2025

Desarrollo

1. **“Ordenar por Quicksort la siguiente lista de números paso a paso.”**

[8, 1, 5, 14, 4, 15, 12, 6, 2, 11, 10, 7, 9]

8	1	5	14	4	15	12	6	2	11	10	7	9
Pivote: 9												
8	1	5	4	6	2	7	9	15	12	10	14	11
Pivote: 7												
1	5	4	6	2	7	8	9	15	12	10	14	11
Pivote: 2												
1	2	4	6	5	7	8	9	15	12	10	14	11
Pivote: 5												
1	2	4	5	6	7	8	9	15	12	10	14	11
Pivote: 11												
1	2	4	5	6	7	8	9	10	11	12	14	15
Pivote: 15												

Esta es la forma más corta que me salió. Otra manera que estaba intentando estaba tomando 3 tablas. Es lo que mayormente me agarro tiempo…

2. **"Realizar la 1 con el método de inserción también. ¿Cuál es la diferencia en tiempo de ordenamiento? ¿Cuál es la ventaja que tiene Quicksort?"**

8	1	5	14	4	15	12	6	2	11	10	7	9
1	8	5	14	4	15	12	6	2	11	10	7	9
1	5	8	14	4	15	12	6	2	11	10	7	9
1	5	4	14	8	15	12	6	2	11	10	7	9
1	5	4	14	6	15	12	8	2	11	10	7	9
1	5	4	14	6	15	12	2	8	11	10	7	9
1	5	4	14	6	15	12	2	7	11	10	8	9
1	4	5	14	6	15	12	2	7	11	10	8	9
1	2	4	5	14	6	15	12	7	11	10	8	9
1	2	4	5	6	14	15	12	7	11	10	8	9
1	2	4	5	6	12	14	15	7	11	10	8	9
1	2	4	5	6	7	12	14	15	11	10	8	9
1	2	4	5	6	7	11	12	14	15	10	8	9
1	2	4	5	6	7	10	11	12	14	15	8	9
1	2	4	5	6	7	8	10	11	12	14	15	9
1	2	4	5	6	7	8	9	10	11	12	14	15

Significado de los colores
Comparador.
Con el que se compara.
Ya comparado.
Fin de la comparación.
Ya completado.

Comparación:
-Quicksort funciona bastante bien con listas grandes *(aunque es la que más me tomo el tiempo)* Usando O(n log n). Mientras que la de inserción funciona bien solo si la lista esta ya casi ordenada.

Ventaja de Quicksort:
-Mucho más rápido con listas grandes y desordenadas, mientras que la de inserción hace muchas comparaciones y movimientos.

3. **"Ordenar por heapsort paso a paso la siguiente lista."**

[10, 5, 7, 9, 4, 11, 45, 17, 60]

-[2:24 P.M.]

4. **"Clasificar los diferentes métodos de ordenamiento estudiados si funcionan con O(0), O(n), O(log n), O(nlogn)."**

Burbuja:
-Funciona mejor con O(n) cuando el arreglo ya está ordenado, porque solo recorre una vez. No funciona con O(0) ni con O(log n) porque siempre requiere comparar elementos. No trabaja con O(nlogn), ya que no divide ni aplica estructuras más eficientes.

Inserción:
-Puede alcanzar O(n) si el arreglo está casi ordenado, ya que solo hace pequeños desplazamientos. No entra en O(nlogn) porque compara secuencialmente sin dividir ni aplicar recursión, y no tiene casos con los otros.

Selección:
-No se adapta a O(n) porque siempre realiza las mismas comparaciones sin importar el orden. No entra en O(nlogn) porque siempre recorre todo el arreglo de forma lineal y repetitiva, y los otros no se le aplican.

Quicksort:
-Funciona muy bien con O(nlogn) cuando los pivotes dividen equilibradamente. No entra en O(0) ni en O(n), ya que siempre hay partición. Tampoco encaja en O(log n) porque recorre todo el arreglo.

Heapsort:
-Su estructura permite operar de forma eficiente en O(nlogn) sin importar el orden inicial. No es O(0), O(n) ni O(log n) por el trabajo de construcción y extracción del heap.

5. **"Investigar los algoritmos de KPM y de Boyer Moore. Anotar ejemplos. ¿Cuál es mejor para encontrar cadenas en un párrafo?"**

KMP (Knuth-Morris-Pratt)

Este algoritmo evita retrocesos innecesarios usando la tabla LPS (Longest Prefix Suffix), que indica cuánto avanzar tras un fallo. Su complejidad es O(n + m), donde n es largo del texto y m del patrón.

Ejemplo:
Texto: "aaaaaaab"
Patrón: "aaaab"
KMP precalcula cuánto retroceder en el patrón cuando falla, evitando repetir comparaciones de prefijos.

Boyer-Moore

Este busca de derecha a izquierda en el patrón, y al fallar aplica dos heurísticas: caracter malo y sufijo bueno, para saltos grandes. En promedio es sublineal, típico O(n/m), aunque el peor caso puede ser O(n·m).

Ejemplo:
Texto: "bananas"
Patrón: "nana"
Compara desde el final "a", si no coincide, salta según la heurística. Se aprovechan saltos grandes al detectar caracteres que no aparecen en el patrón.

¿Cuál es mejor para buscar en un párrafo?

Si el objetivo es la velocidad en la mayoría de los textos grandes, Boyer-Moore parece ser mejor. Si la predictibilidad es más importante, con patrones repetitivos o críticas en tiempo, KMP es una opción segura y eficiente.

6. **"Para el siguiente código documentar e imprimir ejecución"**

Lab10.java

```java
public static void quickSort(int[] array, int low, int high) {
    if (low < high) {
        int partitionIndex = partition(array, low, high); // Obtener índice de partición
        quickSort(array, low, partitionIndex - 1); // Lado izquierdo
        quickSort(array, partitionIndex + 1, high); // Lado derecho
    }
}

// Partición: reorganiza el arreglo y coloca el pivote en su lugar
private static int partition(int[] array, int low, int high) {
    int pivot = array[high];
    int i = low - 1;

    for (int j = low; j < high; j++) {
        if (array[j] <= pivot) {
            i++;
            int temp = array[i]; array[i] = array[j]; array[j] = temp;
        }
    }
    int temp = array[i + 1]; array[i + 1] = array[high]; array[high] = temp;
    return i + 1;
}

// El método main para probar el quicksort
public static void main(String[] args) {
    int[] array = {10, 7, 8, 9, 1, 5};
    System.out.println("Array original:");
    printArray(array);
    quickSort(array, 0, array.length - 1);
    System.out.println("Array ordenado por QuickSort:");
    printArray(array);
}

// Y el método para imprimir arreglo
private static void printArray(int[] array) {
    for (int num : array) {
        System.out.print(num + " ");
    }
    System.out.println();
}
```

```
Output - lab10 (run)
run:
Array original:
10 7 8 9 1 5
Array ordenado por QuickSort:
1 5 7 8 9 10
BUILD SUCCESSFUL (total time: 0 seconds)
```