Universidad Nacional Autónoma de México

Facultad de Ingeniería

(1317) Estructura de Datos y Algoritmos II

Profesor: M.I. Edgar Tista Garcia

Grupo: 05

Semestre 2024-2

Proyecto 1 - Complejidad Computacional en los Algoritmos de Ordenamiento.

Documentación

Equipo 02

Integrantes:

Cabrera Rojas Oscar

Chavez Marquez Sergio Antonio

Noyola Torres Pablo Sebastian

Lunes 25 de marzo del 2024

1. Material de apoyo

Para la investigación del algoritmo de ordenamiento adicional y la escritura de archivos en Java, consultó los siguientes sitios en internet en su búsqueda confiable de información.

1.1. API de Java

- Oracle. (2014). *Class FileWriter*. Java Platform Standard Edition 8 Documentation. Recuperado 22 de marzo de 2024, de https://docs.oracle.com/javase/8/docs/api/java/io/FileWriter.html
- Oracle. (2014). *Class IOException*. Java Platform Standard Edition 8 Documentation. Recuperado 22 de marzo de 2024, de https://docs.oracle.com/javase/8/docs/api/java/io/IOException.html

1.2. Sitios de internet consultados

- *Pancake sorting*. (2023, 11 abril). GeeksforGeeks. Recuperado 22 de marzo de 2024, de https://www.geeksforgeeks.org/pancake-sorting/
- Jindal, H. (2023, 12 octubre). *Ordenamiento de panqueques*. DelftStack. Recuperado 21 de marzo de 2024, de https://www.delftstack.com/es/tutorial/algorithm/pancake-sort/
- Mario Storti. (2009, 13 noviembre). *Pancake sort algorithm, visualization with VTK* [Vídeo]. YouTube. https://www.youtube.com/watch?v=kk-_DDgoXfk
- The Hexagon. (2023, 11 mayo). *Pancake sort* [Vídeo]. YouTube. https://www.youtube.com/watch?v=kXKkHRQmL6o
- *Calculadora gráfica - GeoGebra*. (s.f.). https://www.geogebra.org/graphing?lang=es

2. Propuesta de diseño de clases

Una vez se decidió realizar el programa principal utilizando el lenguaje de programación Java, el diseño y estructura de cada uno de los programas desarrollados resultó ser claro, pues se partió de la idea de implementar una clase individual para cada algoritmo de ordenamiento y, al ser nueve ordenamientos[1] se requerían entonces nueve archivos destinados a ordenamientos, dichos archivos una vez desarrollados no tienen más acción dentro del funcionamiento general del programa, por lo que lo sería conveniente anexarlos en una carpeta donde puedan guardarse sin contaminar la vista del resto de elementos del programa. Dado el contexto de estar trabajando en Java y retomando la idea de que, para separar física y lógicamente una parte del funcionamiento de un programa se debe conjuntar en un *package*, se decidió también entonces organizar el resto del funcionamiento del programa en paquetes acordes a la lógica de la función de cada programa.

[1] Los ordenamientos obligatorios en los requerimentos, el extra y ambos opcionales: *InsertionSort*, *SelectionSort*, *BubbleSort*, *PancakeSort*, *QuickSort*, *HeapSort*, *MergeSort*, *CountingSort* y *RadixSort*.

Cada uno de los algoritmos cuenta con un método de partida, generalmente se nombraron exacatamente igual que su clase, con la diferencia de tener la primera letra en minúscula; en el caso de aquellos que funcionan en base a la recursividad (*QuickSort*, *MergeSort* y *PancakeSort*), como estos métodos están guardados en su propia lógica para aplicar la recursividad, su método de inicio se llama *iniciarQuickSort*, *iniciarMergeSort* y *pancakeSortBegin* respectivamente. Estos métodos de inicio son los únicos que contienen un valor de retorno (el de conteo de operaciones)[2]. Absolutamente todos los métodos implicados en la lógica individual de cada ordenamiento son métodos estáticos, ésto con la idea de que no se van a instanciar objetos de estas clases, sino están destinados única y meramente a realizar su funcionamiento de ordenar un conjunto de datos a la vez.

El contador de operaciones en todos los casos se manejó como una variable de clase para que pueda ser accesible y global para cada parte del funcionamiento, no siendo relevante el lugar de procedencia del incremento.

Partiendo de la lógica de lo que debe ser capaz de realizar el programa, se creó lo que es el punto de partida y programa principal, el archivo *Ordenamientos.java*, cuya función es la de ejecutar todos los ordenamientos el número requerido de veces y manejar dichos datos para las operaciones necesarias, éstas son la impresión en pantalla de las tablas correspondientes y la llamada al archivo *Archivos.java* para permitir la correcta graficación de los datos obtenidos y visualizar así el análisis de complejidad experimental para cada ordenamiento. Siguiendo la idea de utilizar distintos *package*, esta clase se anexó a *principal*.

Es pertinente mencionar que todos los métodos utilizados en este archivo son estáticos, puesto que no se está partiendo de realizar clases como representaciónes de objetos de la vida real, más bien las clases en el programa están hechas para delegar funciones y dividir la lógica de funcionamiento en varias partes.

Al trabajar con arreglos que se dan por entendidos como previamente llenados de elementos especificados[3] y listos para ser utilizados en los ordenamietnos, y por ciertas operaciones necesarias en los ordenamientos (intercambio de elementos), se decidió implementar la clase *Utilerias.java* como el lugar para contener estas operaciones recurrentes y necesarias para el programa, pero que no tienen cabida dentro de la lógica en otras partes. Al igual que el resto del programa, sus métodos son todos métodos estáticos, y al ser parte del funcionamiento general se agregó también al paquete de *principal* junto a *Ordenamientos.java*.

Finalmente, la última clase en Java *Archivos.java* pertenece a su propio paquete, *graficar*. A pesar de no formar parte de la resolución del problema inicial, ni de la lógica del funcionamiento del programa en sí, sino que se considera como un agregado por la necesidad de graficar los resultados, se puede considerar también dentro de este análisis de archivos y clases. La función de este archivo, es entonces, la de escribir los correspondientes archivos *.txt* de los resultados de cada ordenamiento obtenidos por *Ordenamientos.java* para que se puedan graficar.

[2]En el caso de los algoritmos con recursividad implicada se tuvo que hacer un método distinto al recursivo para el valor de retorno ya que, de funcionar como el resto, el análisis de complejidad se veia comprometido, no funcionando correctamente ni como era esperado.

[3]Arreglos llenos de números aleatorios, para el caso.

Este archivo nace de la necesidad de manejar grandes volúmenes de información eficientemente y de poder replicar nuestros resultados de manera cómoda para ver en cada ejecución el efecto de los cambios realizados. Haciendo énfasis en aquellos casos en los que el conteo del ordenamiento no fuera el esperado, que se vería al momento de graficar mediante los archivos en Python, este archivo cumple entonces en realizar la conexión de los datos obtenidos con el programa que debe procesarlos (graficarlos), la solución para hacer esta conexión son los archivos de texto, este programa alterno al principal encuentra su función en escribir los datos obtenidos en diez archivos de texto distintos.

Los programas de Python al no estar contenidos en la lógica de los programas en Java no están organizados en paquetes, sino que se encuentran fuera de cualquiera de éstos. *Graficar.py* es totalmente complementario a *Archivos.java*, pues lee todos los archivos creador por éste y grafica, en tres bloques, la información de los ordenamientos. En el primero los algoritmos cuya complejidad teórica es n^2, en el segundo los algoritmos cuya complejidad teórica es $nlog(n)$, y en el tercero los comprometidos espacialmente: nk y $n + k$.

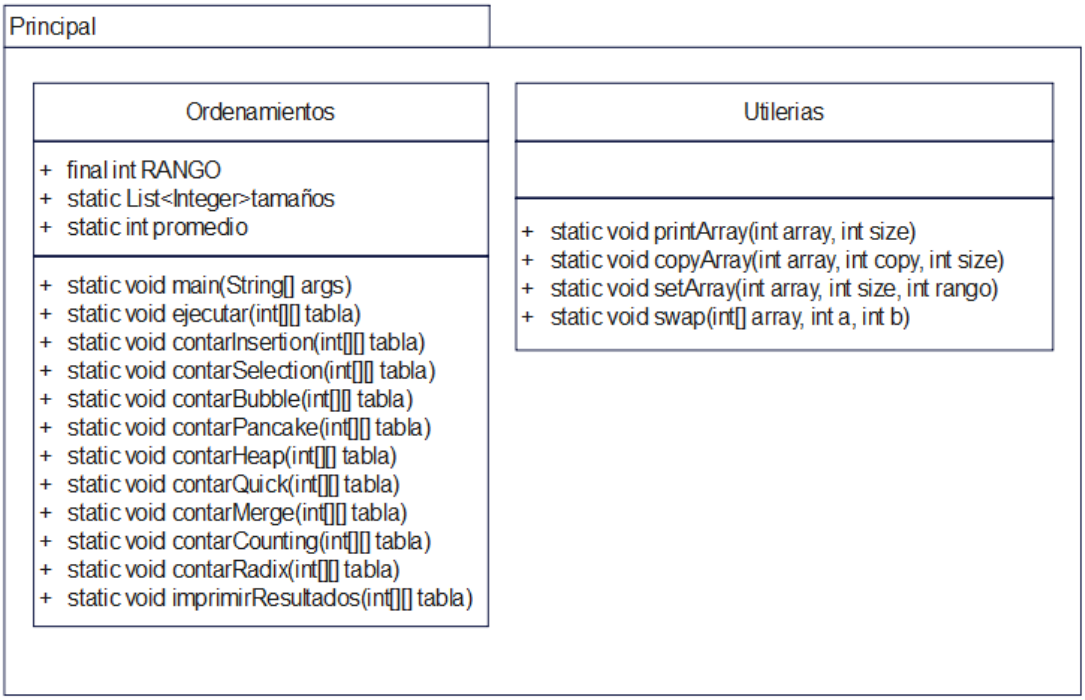
El programa *GraficarLogaritmo.py* es independiente a toda la lógica del programa y se utilizó como recurso para analizar los resultados obtenidos para los algoritmos de complejidad $nlog(n)$, comparando la gráfica teórica obtenida en Python de $nlog(n)$ cuando inicia en una centésima y en doscientos hasta cinco mil en cuatroscientos intervalos equidistantes. y contrastándola a la del programa principal.

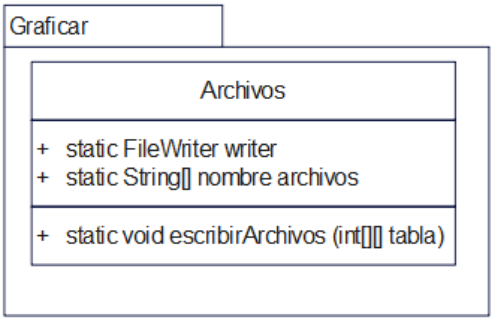
A continuación se muestra la organización por paquetes que se encuentra en el programa:

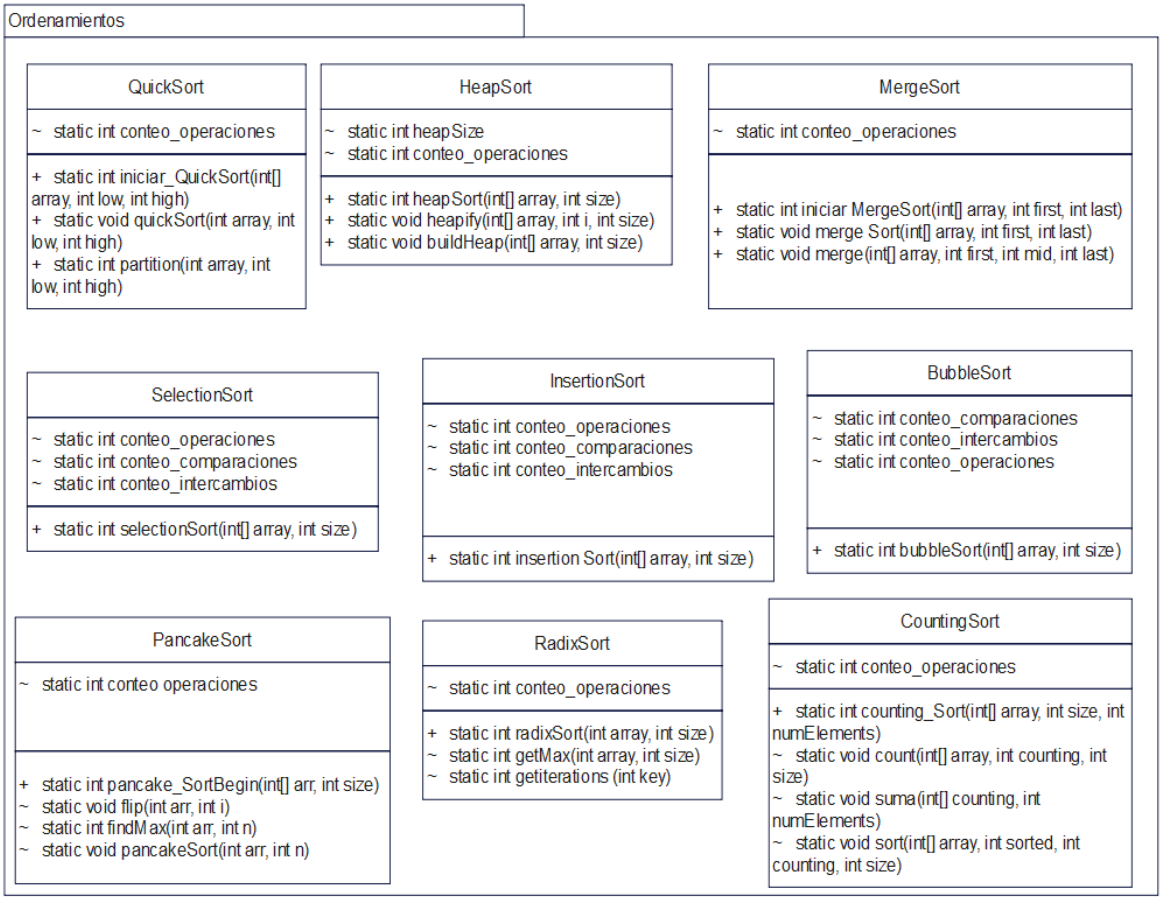
- graficar
  - Archivos.java
- ordenamientos
  - InsertionSort.java
  - SelectionSort.java
  - BubbleSort.java
  - PancakeSort.java
  - QuickSort.java
  - HeapSort.java
  - MergeSort.java
  - CountingSort.java
  - RadixSort.java
- principal
  - Ordenamientos.java
  - Utilerias.java
- Graficar.py
- GraficarLogaritmo.py

2.1. Diagrama UML

Se presentan los diagramas UML correspondientes a la lógica del programa principal en Java del proyecto. Se excluyen los programas en Python.







3. Bitácora del proyecto

Regisro de fecha, hora y lugar/medio de reunión de los integrantes para la organización, asignación de tareas y discución del proyecto.

- **Viernes 08 de marzo - 19:00 Hrs.**: Reunión en Zoom. Se estructuró de manera general lo que conformaría la realicación del proyecto, y se distribuyeron algunos puntos a trabajar.
  Encargado(s): Cabrera Rojas Oscar, Chavez Marquez Sergio Antonio, Noyola Torres Pablo Sebastian.

- **Sábado 09 de marzo - 20:00 Hrs.**: Creación del archivo colaborativo de java, en la plataforma de *Replit*.
  Encargado(s): Noyola Torres Pablo Sebastian

- **Martes 12 de marzo - 20:00 Hrs.**: Implemetación individual de los ocho algoritmos de ordenamiento tratados en clase.
  Prueba individual en crudo.
  Encargado(s): Cabrera Rojas Oscar

- **Martes 12 de marzo - 20:00 Hrs.**: Creación de los *Archivos.java* y *Graficar.py* para implementar la graficación de resultados posteriores a cada ejecución del programa principal.
  Encargado(s): Cabrera Rojas Oscar

- **Jueves 14 de marzo - 06:00 Hrs.**: Implementación del algoritmo adicional (*PancakeSort*).
  Encargado(s): Noyola Torres Pablo Sebastian

- **Jueves 14 de marzo - 17:30 Hrs.**: Inicialicación y estructuración del archivo escrito en LaTeX.
  Encargado(s): Chavez Marquez Sergio Antonio

- **Jueves 14 de marzo - 19:00 Hrs.**: Implementación de contadores de operaciones *BubbleSort*, *SelectionSort* e *InsertionSort*.
  Encargado(s): Noyola Torres Pablo Sebastian

- **Jueves 14 de marzo - 22:00 Hrs.**: Organización en paquetes de las clases del programa.
  Implementación del algoritmo adicional (*PancakeSort*) en la lógica de graficación.
  Planeación de la lógica del funcionamiento.
  Encargado(s): Cabrera Rojas Oscar.

- **Jueves 15 de marzo - 22:00 Hrs.**: Organización y seleccionamiento de los contenidos textuales y de los archivos escritos en el procesador de LaTeXmediante *OverLeaf* correspondiente: trabajo_escrito.tex y documentacion.text.
  Encargado(s): Cabrera Rojas Oscar

- **Viernes 22 de marzo - 20:00 Hrs.**: Implementación de los contadpres en los algoritmos restantes, así como la rectificaión de los ya existentes.
  Encargado(s): Chavez Marquez Sergio Antonio, Noyola Torres Pablo Sebastian.

- **Viernes 22 de marzo - 20:00 Hrs.**: Desarrollo final del programa principal *Ordenamientos.java* y lógica de su funcionamiento terminada.
  Encargado(s): Cabrera Rojas Oscar.

- **Viernes 22 de marzo - 22:00 Hrs.**: Desarrollo final del sistema de escritura y lectura de archivos de texto, cambios en *Archivos.java* y *Graficar.py*.
  Encargado(s): Cabrera Rojas Oscar.

- **Sábado 23 de marzo - 00:30 Hrs.**: Reunión en Zoom. Primeras pruebas preliminares del funcionamiento del programa.
  Encargado(s): Cabrera Rojas Oscar, Chavez Marquez Sergio Antonio, Noyola Torres Pablo Sebastian.

- **Sábado 23 de marzo - 10:00 Hrs.**: Reunión en Zoom. Corrección de variables contadoras de operaciones en *QuickSort* y *MergeSort*. Resolución de problema de retorno *int* con recursividad.
  Encargado(s): Cabrera Rojas Oscar, Chavez Marquez Sergio Antonio, Noyola Torres Pablo Sebastian.

- **Sábado 23 de marzo - 15:00 Hrs.**: Creación del programa *GraficarLogaritmo.py* para realizar pruebas sobre el funcionamiento de graficación con Python y visualización de la forma de la gráfica $n\log(n)$
  Encargado(s): Cabrera Rojas Oscar.

- **Sábado 23 de marzo - 22:00 Hrs.**: Cambio no planeado a *Ordenamientos.java* para que la lista (antes arreglo) de tamaños sea dinámica y creada a partir de la entrada del usuario.
  Encargado(s): Cabrera Rojas Oscar.

- **Domingo 24 de marzo - 20:00 Hrs.**: Reunión en Zoom. Organización para la grabación del video, y realización de pruebas.
  Encargado(s): Cabrera Rojas Oscar, Chavez Marquez Sergio Antonio, Noyola Torres Pablo Sebastian.

- **Domingo 24 de marzo - 21:00 Hrs.**: Reunión en Zoom. Grabación conjunta del video explicativo sobre el funcionamiento del programa.
  Encargado(s): Cabrera Rojas Oscar, Chavez Marquez Sergio Antonio, Noyola Torres Pablo Sebastian.

- **Lunes 25 de marzo - 02:00 Hrs.**: Edición del video explicativo sobre el funcionamiento del programa y adición de las pistas de audio individuales.
  Encargado(s): Cabrera Rojas Oscar, Chavez Marquez Sergio Antonio, Noyola Torres Pablo Sebastian.

- **Lunes 25 de marzo - 22:00 Hrs.**: Terminamos el ingreso de la información en los archivos de texto correspondientes al "Reporte Escrito" y "Documentación".
  Encargado(s): Cabrera Rojas Oscar, Chavez Marquez Sergio Antonio, Noyola Torres Pablo Sebastian.

4. Calendarización del proyecto

Aquí se muestra un calendario del mes de marzo marcando los aspectos más relevantes de la cosntrucción de este proyecto.

DOM	LUN	MAR	MIE	JUE	VIE	SAB
					01	02
03	04	05	06	07	08 **PRIMERA REUNIÓN**	09 CREACIÓN DE LOS ARCHIVOS DE LOS CÓDIGOS
10	11	12 IMPLEMENTACIÓN DE 08 ALGORITMOS	13	14 IMPLEMENTACIÓN DEL ALGORITMO PANCAKE **ENTREGA DE AVANCES**	15	16
17	18	19	20	21	22 **SEGUNDA REUNIÓN**	23
24 **GRABACIÓN DEL VIDEO** CÓDIGOS CONCLUIDOS	25 TRABAJO ESCRITO CONLCUIDO **ENTREGA DEL PROYECTO**	26	27	28	29	30