	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: TISTA GARCÍA EDGAR

Asignatura: ESTRUCTURA DE DATOS Y ALGORITMOS II

Grupo: 5

No de Práctica(s): PRACTICA #3 – ALGORITMOS DE ORDENAMIENTO PARTE 3

Integrante(s): CARRILLO CERVANTES IVETTE ALEJANDRA

No. de Equipo de cómputo empleado: TRABAJO EN CASA

No. de Lista o Brigada: 09

Semestre: 2022 - 1

Fecha de entrega: 04 OCTUBRE 2021

Observaciones:

CALIFICACIÓN: _____

PRÁCTICA #3 – Algoritmos de Ordenamiento Parte 3

OBJETIVO: El estudiante identificará la estructura de los algoritmos de ordenamiento Counting Sort y Radix Sort

OBJETIVO DE CLASE: El alumno implementará casos particulares de estos algoritmos para entender mejor su funcionamiento a nivel algorítmico.

ACTIVIDADES PARA EL DESARROLLO DE LA PRÁCTICA

Esta práctica tiene como objetivo principal implementar los algoritmos Counting Sort y Radix Sort vistos en clase, en lenguaje de programación JAVA. Para comprobar el funcionamiento de cada algoritmo, se creó una clase "Principal" donde se encuentra el método main, dentro de este método se realizó un menú con las siguientes tres opciones:

- Counting Sort
- Radix Sort
- Salir

*Cada opción se describirá más adelante con su respectiva salida.

También se realizaron las siguientes clases auxiliares para hacer más eficiente su implementación:

- Ordenamiento.java

Contiene los métodos CountingSort y RadixSort.

- Utilerias.

Contiene un método sobrecargado de 2 diferentes formas para imprimir un arreglo de caracteres o de enteros según sea el caso. También, contiene el método para llenar un arreglo de enteros con ceros.

- Cola.

Tiene como objetivo crear un objeto de tipo First In First Out con la finalidad de almacenar cada digito o carácter de los datos a ordenar. Además del método constructor en el cual se especifica que el tamaño de la cada cola creada será 4 , el primer elemento será 1 y el último 0 y se guardarán los datos en un Array, también contiene un método llamado "encolar" el cual servirá para agregar a la cola un elemento dado por el usuario.

Primera opción "Counting Sort"

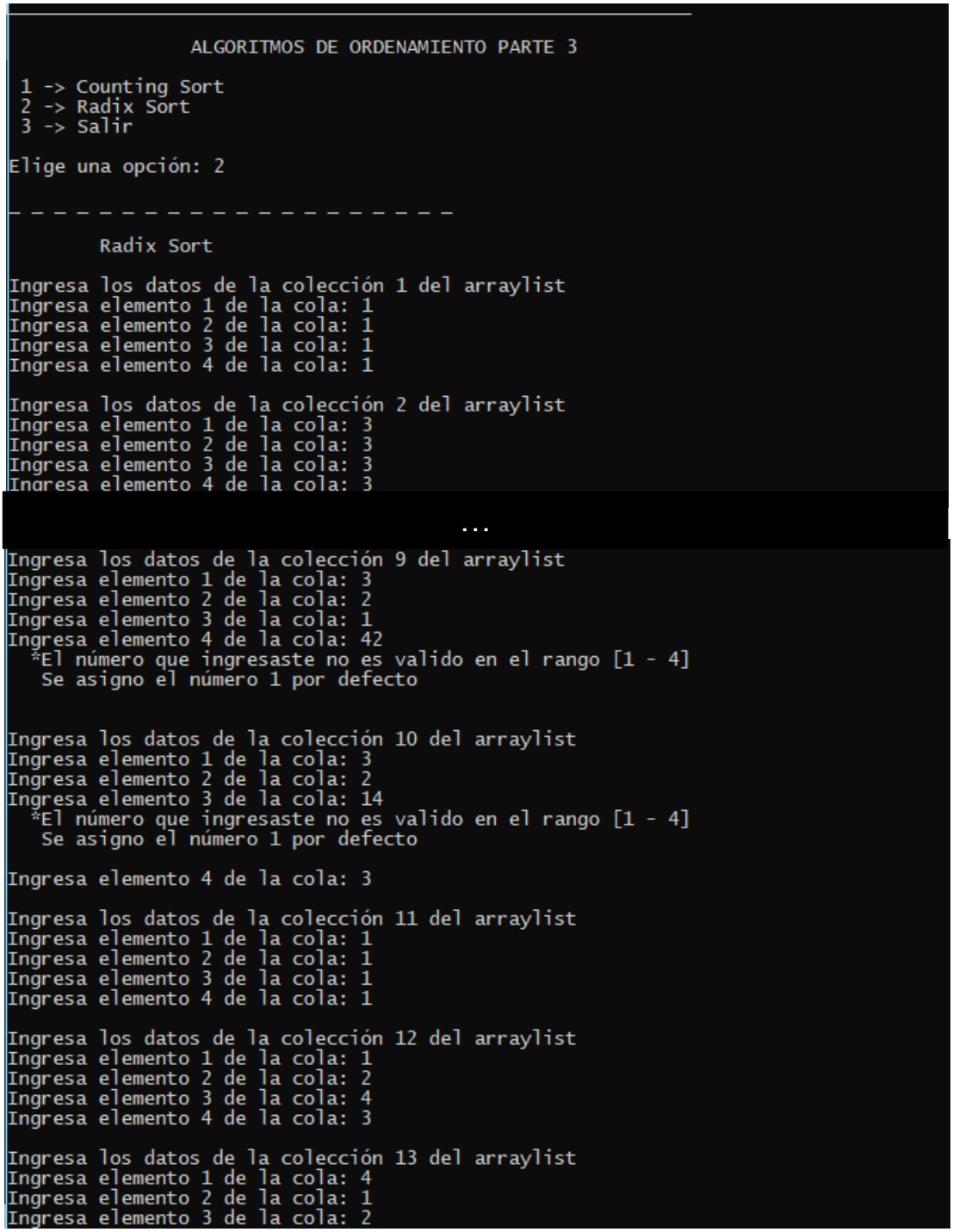
Para implementar este algoritmo se siguió el pseudocódigo visto en clase; sin embargo, se realizaron varias modificaciones. Primero, en el archivo Principal se le pidió al usuario ingresar los datos de cada indice del arreglo el cual se ordenaría, en un rango de letras de [A-B], en dado caso de que ingresará otra letra por error, se le asignaría por defecto la letra 'A' a el indice correspondiente; una vez ingresados los datos y guardados en el arreglo, se imprimió el arreglo a ordenar y después se llamo al método "countingSort" de la clase Ordenamiento, este método tiene como parámetro el arreglo a ordenar y tiene como función acomodar el arreglo según el algoritmo Counting Sort, a diferencia del pseudocódigo visto en clase, se generó un nuevo arreglo con 10 elementos inicializados y se mando a llamar al método de llenarCeros con el fin de que se llene dicho arreglo en puros ceros en cada uno de sus índices (cabe destacar que, este arreglo se utilizará para contar cuantas letras hay); luego, mediante una estructura condicional if-else dentro

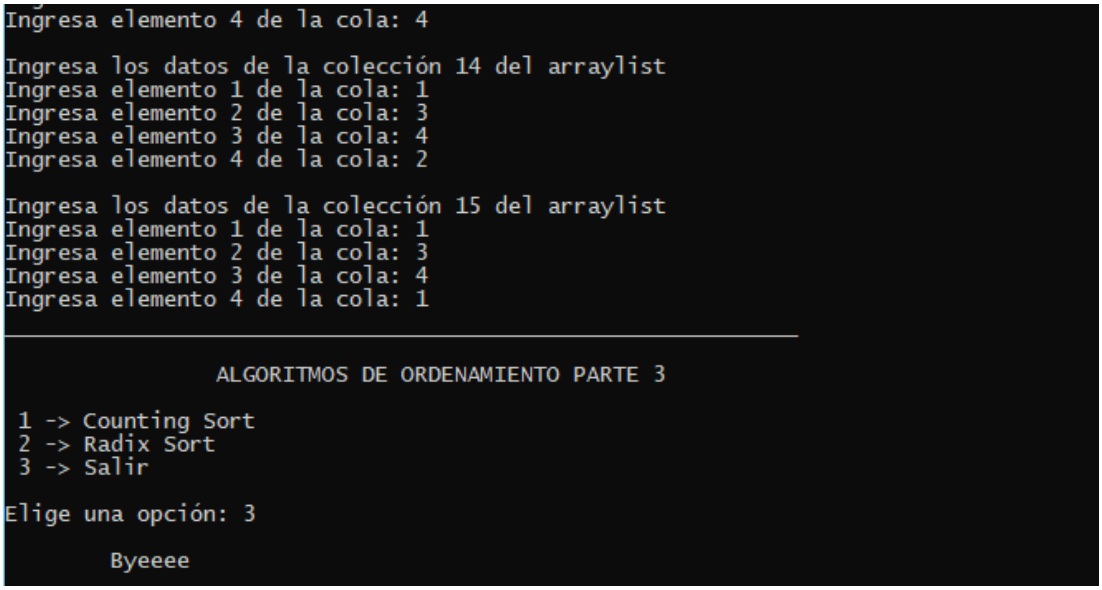
de un ciclo for, se fue comparando el indice de cada elemento del arreglo a ordenar, con cada letra en un rango de [A-J], si estos eran iguales se sumaba un elemento al indice correspondiente del arreglo previamente inicializado con ceros "*Count*". Con ayuda de otro ciclo de repetición for se fue sumando cada indice del arreglo "count" con su indice anterior para saber en que posición se tendrá que agregar cada letra ordenada en un arreglo nuevo "finalArr" (diferente a los anteriores), para esto último, se realizó un ciclo for el cual fue recorriendo el arreglo a ordenar de derecha a izquierda y se fue comparando con cada letra en el rango [A-J], si estas eran iguales, se asignaba una nueva variable al valor del indice que le corresponde en el arreglo Count y en esa posición (el valor del índice) en el arreglo "finalArr" se le asigno el carácter correspondiente . Finalmente, se imprimió la lista ordenada.

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2022-1\EDAII\Prácticas\P3 EDA\Carrillo
Cervantes Ivette Alejandra G5 P3 V1>java Principal
_____________________________________________________

                ALGORITMOS DE ORDENAMIENTO PARTE 3

 1 -> Counting Sort
 2 -> Radix Sort
 3 -> Salir

Elige una opción: 1

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

        Counting Sort

Ingresa los elementos en el array en un rango de [A-J]

Elemento [1] del array: A
Elemento [2] del array: J
Elemento [3] del array: E
Elemento [4] del array: D
Elemento [5] del array: L
 *La letra que ingresaste no es valida en el rango [A-J]
  Se asigno la letra 'A' por defecto

Elemento [6] del array: C
Elemento [7] del array: D
Elemento [8] del array: B
Elemento [9] del array: E
Elemento [10] del array: F
Elemento [11] del array: H
Elemento [12] del array: Q
 *La letra que ingresaste no es valida en el rango [A-J]
  Se asigno la letra 'A' por defecto

Elemento [13] del array: J
Elemento [14] del array: A
Elemento [15] del array: E
Elemento [16] del array: C
Elemento [17] del array: B
Elemento [18] del array: I
Elemento [19] del array: D
Elemento [20] del array: Z
 *La letra que ingresaste no es valida en el rango [A-J]
  Se asigno la letra 'A' por defecto


 -> El arreglo a ordenar es el siguiente:
 { A, J, E, D, A, C, D, B, E, F, H, A, J, A, E, C, B, I, D, A}

 -> Cuenta de los elementos:
 { 5, 2, 2, 3, 3, 1, 0, 1, 1, 2}
   A  B  C  D  E  F  G  H  I  J

 -> Suma de los elementos:
 { 5, 7, 9, 12, 15, 16, 16, 17, 18, 20}
   A  B  C  D  E  F  G  H  I  J

 -> El array ordenado, es el siguiente:
 { A, A, A, A, A, B, B, C, C, D, D, D, E, E, E, F, H, I, J, J}
```

Segunda opción "Radix Sort"

Para implementar este código, lo primero que se hizo fue crear una clase llamada "Cola" la cual crearía objetos tipo First In First Out, en cada objeto se le asignaría dígitos o caracteres los cuales serían datos a odenar, cada objeto se guardaría dentro de un ArrayList previamente inicializado. Para obtener los datos de cada cola, se realizó un ciclo for en el archivo Principal, se le fue pidiendo al usuario que ingresará dato por dato y posteriormente con el método "encolar" se agregó a la cola creada previamente. Este programa solo acepta números de un rango de [1-4], si el usuario ingresa otro número el cual esta fuera del rango, se le asignará como defecto el elemento 1.
La salida de el programa al pedir los datos es la siguiente:





./Principal.java

No se pudo implementar el algoritmo de ordenamiento Radix Sort, ya que al intentarlo, marcaba varios errores con respecto a las colecciones que se usaron, considero que si se hubieran utilizado otras colecciones como LinkedList hubiera sido más sencilla su implementación, así como su eficiencia, debido a algunos métodos que trae esta biblioteca.

Conclusiones

Se cumplió con la mayoría de los objetivos de esta práctica ya que se realizó el algoritmo de ordenamiento Counting Sort, y a pesar de que no se pudo complementar el algoritmo Radix Sort se observaron varias formas de implementar este.

Personalmente, me costó un poco de trabajo implementar ambos algoritmos, ya que a Counting Sort se le tuvo que cambiar muchas cosas y considero que se pudo tener una solución más eficiente de la planteada en esta práctica, pues fueron muchas lineas de código para un algoritmo donde se supone que su complejidad es de O(n). Mientras que, para Radix Sort, considero que, como se mencionó anteriormente, tal vez se hubiera completado la actividad si se hubiera usado otra colección.

En mi opinión, considero que las actividades de esta práctica fueron adecuadas para completar el tema 1 de la materia de Estructura de Datos y Algoritmos; además, fue buena idea poder elegir el lenguaje a usar, ya sea Java o C, pues es una buena forma de conocer un poco más y practicar el lenguaje que estamos aprendiendo en Programación Orientada a Objetos, el cual es Java. (: