Guía código Ordenación Radix

Nombre: **Juan David Jiménez Romero** **NRC:** **16137**

Código OrdenamientoRadix:

La primera línea es package ordenamientoradix nos indica que está en la libreria ordenaminetoRadix

La siguiente es public class OrdenamientoRadix que nos dice que es el inicio de la clase OrdenamientoRadix

public static void main(String args[]){ es un metodo que inicializa el programa

ordenRadix radix = new ordenRadix(); es la creacion de la instancia radix de la clase ordenRadix

int vec[]={45,17,23,67,21, 44, 12,34; En esta línea creacion del array de tipo entero que se usó de ejemplo con los valores dentro de las llaves

System.out.println("Vector original") esta línea hace que se muestra en pantalla un mensaje que dice Vector original

La línea radix.imprimirVector(vec) realiza la llamada al metodo de la clase ordenRadix que imprime los datos del array

System.out.println("\nVector ordenado") muestra en pantalla un mensaje que dice Vector ordenado

vec= ordenRadix.ordenacionRadix(vec)esta línea hace es crear una variable vec la cual va a tomar el valor devuelto del metodo ordenacionRadix()

radix.imprimirVector(vec) esta línea vuelve a llamar al metodo imprimirVector para mostrar los datos ordenados

Codigo de la clase Main:

```
package ordenamientoradix;//nos indica que esta en la libreria ordenaminetoRadix

/**

 *

 * @author jrome

 */

public class OrdenamientoRadix {//inicio de la clase OrdenamientoRadix


    public static void main(String args[]){//metodo que inicializa el programa

        ordenRadix radix = new ordenRadix();//creacion de la instancia radix de la clase ordenRadix
```

int vec[]={45,17,23,67,21, 44, 12,34};//creacion del vector de tipo entero que se uso de ejemplo con los valores dentro de las llaves

System.out.println("Vector original");//se muestra en pantalla un mensaje que dice Vector original

radix.imprimirVector(vec);//realiza la llamada al metodo de la clase ordenRadix que imprime los datos del array

System.out.println("\nVector ordenado");//muestra en pantalla un mensaje que dice Vectro ordenado

vec= ordenRadix.ordenacionRadix(vec);//una variable vec va a tomar el valor devuelto del metodo ordenacionRadix()

radix.imprimirVector(vec);//se vuelve a llamar al metodo imprimirVector para mostrar los datos ordenados

}//fin del método OrdenamientoRadix

}//Fin de la clase método ordenaminetoRadix

Clase ordenRadix:

Se empieza la clase con la línea package ordenamientoradix nos indica que está en el paquete ordenaminetoRadix

Por consiguiente las llamadas de las librerías import java.util.LinkedList nos dice que usa la librería de listasEnlazadas que nos permite ocupar las funciones de estas

Y la línea import java.util.Queue; que nos dice que usa la librería de colas y nos deja usar los métodos únicos para este tipos de TDA

public class ordenRadix esta línea da inicio de la clase ordenRadix

public static int[] ordenacionRadix(int[] vec) es el inicio del método estático entero ordenación Radix que contiene el valor vec que es un array

int rep = 1; esta línea nos dice que se declara la variable rep de tipo entero y que inicialize con el valor de 1

int numBytes = 4 esta línea declara una variable de numBytes que inicializa con el numero 4, número de bytes a desplazar

int numColas = (int) Math.pow(2, numBytes);

Queue[] cola = new LinkedList[numColas] esta linea se encarga de la creación de la cola con el nombre cola que tiene como tamaño la variable numColas

for (int i = 0; i < numColas; i++) esta línea es un for que nos dice que recorre la cola donde cada repetición realiza dentro de la cola una lista enlazada

cola[i] = new LinkedList();

} aquí se termina el for

for (int i = 0; i < rep; i++) { //for que recorre dependiendo de la variable rep de repeticiones que se realizara al programa

// En esta parte recorre el vector para guardar cada valor en la cola

for (int numero : vec) {

// Busca el mayor número del vector

if (i == 0) { //Si i es igual a 0 se realizara la siguiente función

if (numero > rep) {//si número es mayor a rep este realizara lo siguiente

rep = numero; // rep toma el valor de numero

// Calcula en que cola debe ir cada número

int numCola = (numero >> div) & 0xf;

cola[numCola].add(numero);

}

div = div + numBytes;

// Recorre cada cola para colocar cada elemento en el vector

int j = 0 esta línea se declara una variable j de tipo entero que inicializa en 0

for (Queue c : cola) for para pasar los datos a la cola cola a la cola c

while (!c.isEmpty()) esta linea nos dice que la variable c mientras este no este vacia realizara lo siguiente

vec[j++] = (int) c.remove() esta línea recorre el array vec y le va asignando los datos de la cola c y remueve el primer valor de este haciendo que el segundo valor de la cola c pase a ser primera hasta que esta se quede vacia

// La primera vez se actualiza el número de veces que debe ejecutar el proceso

if (i == 0)

rep = (int) (Math.log(rep) / Math.log(numColas)) + 1;

return vec regresa el valor del array vec de tipo entero

public static void imprimirVector(int vec[]) este metodo sirve para imprimir en consola los datos del array

for(int i=0;i<vec.length;i++) esta linea dice que el for va a repetirse hasta que i deje de ser menor al tamaño del arreglo este para que imprima todos los valores q estén dentro del array

System.out.print(vec[i]+" "); esta línea muestra en pantalla el valor dentro del array vec[i] y lo que este dentro de la posicion i

Código clase ordenRadix:

```
package ordenamientoradix;//nos indica que esta en la libreria ordenaminetoRadix


import java.util.LinkedList;//dice que usa la libreria de listasEnlazadas que nos permite ocupar las funciones de estas

import java.util.Queue;//dice que usa la libreria de colas y nos deja usar los metodos unicos para este tipos de TDA


public class ordenRadix {//Inicio de la clase ordenRadix

    public static int[] ordenacionRadix(int[] vec) {//inicio del metodo statico entero ordenacion Radix que contiene el valor vec que es un array

        int rep = 1; // cantidad de repeticiones

        int numBytes = 4; // número de bytes a desplazar

        int numColas = (int) Math.pow(2, numBytes);


        // Creación de las colas

        Queue[] cola = new LinkedList[numColas];

        for (int i = 0; i < numColas; i++) {//for que nos dice que recorre la cola donde cada repeticion realiza dentro de la cola una lista enlazada

            cola[i] = new LinkedList();

        }

        int div = 0;//declara un valor denominado div de tipo entero y lo inicializa con 0

        for (int i = 0; i < rep; i++) {

            // En esta parte recorre el vector para guardar cada valor en la cola

            for (int numero : vec) {

                // Busca el mayor número del vector

                if (i == 0) {

                    if (numero > rep) {

                        rep = numero;

                    }

                }
```

```
            // Calcula en que cola debe ir cada número
            int numCola = (numero >> div) & 0xf;
            cola[numCola].add(numero);
        }
        div = div + numBytes;
        // Recorre cada cola para colocar cada elemento en el vector
        int j = 0;
        for (Queue c : cola) {
            while (!c.isEmpty()) {
                vec[j++] = (int) c.remove();
            }
        }
        // La primera vez se actualiza el número de veces que debe ejecutar el proceso
        if (i == 0) {
            rep = (int) (Math.log(rep) / Math.log(numColas)) + 1;
        }
    }

    return vec;//regresa el valor de vec
  }//fin del metodo ordenacionRadix

  public static void imprimirVector(int vec[]){
    //for que repite hasta que i recorra el vector hasta que el vector no durante i no sea igual o mayor al tamaño del array vec[]
    for(int i=0;i<vec.length;i++){
      System.out.print(vec[i]+" ");//muestra en pantalla el valor dentro del array vec[i] y lo que este dentro de la posicion i
    }
  }//fin del metodo imprimirVector
}//fin de la clase ordenRadix
```