Evaluación extráulica 2

P2020_ESI1003D

EDUARDO ETHANDRAKE CASTILLO PULIDO
ANDRE SINSEL AYALA

1) Implementa el siguiente método que ordene el arreglo de productos de forma no descendente de acuerdo al precio del producto, utilizando el algoritmo BucketSort como se vio en clase.
static void ordenarPorPrecio(Producto[] productos)
Especificaciones del algoritmo BucketSort:
a) Se deberán crear N buckets (arreglo o lista).
b) Cada bucket almacenará una lista enlazada de productos.
c) El número de bucket que se asignará a un producto será de acuerdo a su precio, de forma que a un producto cuyo precio sea el mínimo posible se le asigne el bucket 0, y a un producto cuyo precio sea el máximo posible se le asigna el bucket N – 1.
d) Si el bucket no está vacío, el producto deberá insertarse en una posición de la lista enlazada tal que ésta se mantenga ordenada de forma no descendente por precio, y que garantice que el algoritmo sea estable.
e) Se sugieren las siguientes estructuras para crear los buckets:

```
class BucketNode {
    Producto producto;
    BucketNode next = null;
    public BucketNode(Producto producto) {
       this.producto = producto;
    }
}

class Bucket {
    BucketNode first = null;
    int size = 0;  // opcional
}
```

```
//EDUARDO ETHANDRAKE CASTILLO PULIDO
//ANDRE SINSEL AYALA
import java.util.Arrays;

class Producto {
   private String nombre;
   private double precio;
   private int existencias;
   public static final double PRECIO_MIN = 3;
   public static final double PRECIO_MAX = 2000;
   public static final int EXISTENCIAS_MIN = 5;
   public static final int EXISTENCIAS_MAX = 50;
   private static int productId = 0;

   public Producto(String n, double p, int e) {
      this.nombre = n;
      this.precio = p;
      this.existencias = e;
   }
   public Producto() {
      this.nombre = String.format("Producto %3d", ++ productId);
      this.precio = PRECIO_MIN + (PRECIO_MAX - PRECIO_MIN) * Math.random();
      this.existencias = EXISTENCIAS_MIN + (int) ((EXISTENCIAS_MAX - EXISTENCIAS_MIN + 1) * M
ath.random());
   }
   public String getNombre() {
      return nombre;
```

```
    }
    public double getPrecio() {
        return precio;
    }
    public int getExistencias() {
        return existencias;
    }
    public String toString() {
        return String.format("(%s, $%7.2f, %2d)", nombre, precio, existencias);
    }
}

public class Examen2a{
    static class BucketNode {
        Producto producto;
        BucketNode next = null;
        BucketNode previous = null;
        public BucketNode(Producto producto) {
            this.producto = producto;
        }

    }
    static class Bucket {
        BucketNode first = null;
        BucketNode last = null;
        int size = 0; // opcional
    }

    static void insertion(Bucket bucket){
        BucketNode pivotBucket = null;
        Producto pivotProduct = null;
        BucketNode bucketj = null;
        BucketNode bucketj_Plus1 = null;

        if(bucket != null){
            if(bucket.first != null){
                pivotBucket = bucket.first.next;
                pivotProduct = (pivotBucket != null) ? pivotBucket.producto : null;
            }
            while(pivotBucket != null){
                bucketj = pivotBucket.previous;
                bucketj_Plus1 = bucketj.next;
                while(bucketj != null && bucketj.producto.getPrecio() > pivotProduct.getPrecio()){

                    bucketj.next.producto = bucketj.producto;
```

```
                bucketj_Plus1 = bucketj;
                bucketj = bucketj.previous;
            }
            bucketj_Plus1.producto = pivotProduct;
            pivotBucket = pivotBucket.next;
            pivotProduct = (pivotBucket != null) ? pivotBucket.producto : null;
        }
    }
}

static void ordenarPorPrecio(Producto[] productos){
    Bucket bucketList[] = new Bucket[(int)(Producto.PRECIO_MAX-Producto.PRECIO_MIN+1)];

    for(int i = 0;i < bucketList.length; i++){
        bucketList[i] = new Bucket();
    }
    for(int i = 0; i< productos.length; i++){
        BucketNode nodo = new BucketNode(productos[i]);
        int code = (int)(nodo.producto.getPrecio()-Producto.PRECIO_MIN);
        if(bucketList[code].size == 0){
            bucketList[code].first = nodo;
            bucketList[code].last = nodo;
        }else{
            nodo.previous = bucketList[code].last;
            bucketList[code].last.next = nodo;
            bucketList[code].last = nodo;
        }
        bucketList[code].size++;
    }
    int index = 0;
    for(Bucket bucket : bucketList){
        insertion(bucket);
        while(bucket.first != null){
            productos[index] = bucket.first.producto;
            bucket.first = bucket.first.next;
            index++;
        }
    }
}

static boolean isProductSorted(Producto productos[]){
    if(productos != null){
        double pastPrice = productos[0].getPrecio();
        for(int i = 1; i<productos.length;i++){
            if(pastPrice > productos[i].getPrecio()){
```

```
                return false;
            }
        }
        return true;
    }
    return false;
}
public static void main(String[] args){
    /*Producto p1 = new Producto("Producto 1", 22, 32);
    Producto p2 = new Producto("Producto 2", 10, 6);
    Producto p3 = new Producto("Producto 3", 15, 6);
    Producto p4 = new Producto("Producto 4", 200, 30);
    Producto p5 = new Producto("Producto 5", 190, 13);
    Producto p6 = new Producto("Producto 6", 22, 25);
    Producto productList[] = {p1,p2,p3,p4,p5,p6};
    ordenarPorPrecio(productList);
    System.out.println(Arrays.toString(productList));
    System.out.println(isProductSorted(productList));
    */
    for(int n = 100_000; n<=500_000; n+=100_000){
        Producto productList[] = new Producto[n];
        for(int i = 0; i<n; i++){
            productList[i] = new Producto();
        }
        ordenarPorPrecio(productList);
        System.out.println(isProductSorted(productList));
    }

}
}
```

```
    Producto p1 = new Producto("Producto 1", 22, 32);
    Producto p2 = new Producto("Producto 2", 10, 6);
    Producto p3 = new Producto("Producto 3", 15, 6);
    Producto p4 = new Producto("Producto 4", 200, 30);
    Producto p5 = new Producto("Producto 5", 190, 13);
    Producto p6 = new Producto("Producto 6", 22, 25);
    Producto productList[] = {p1,p2,p3,p4,p5,p6};
    ordenarPorPrecio(productList);
    System.out.println(Arrays.toString(productList));
    System.out.println(isProductSorted(productList));
PS C:\Users\eduar>  & 'C:\Users\eduar\.vscode\extensions\vscjava.vscode-java-debug-0.25.1\scripts\launcher.bat' 'C:\Program Files\Java\jdk1.8.0_201\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,
server=n,suspend=y,address=localhost:61010' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\eduar\AppData\Local\Temp\vscodesws_a161d\jdt_ws\jdt.ls-java-project\bin' 'Examen2a'
[(Producto 2, $  10.00,  6), (Producto 3, $  15.00,  6), (Producto 1, $  22.00, 32), (Producto 6, $  22.00, 25), (Producto 5, $ 190.00, 13), (Producto 4, $ 200.00, 30)]
true
```

Como se puede observar en las capturas anteriores, el algoritmo funciona con arreglos pequeños y además es estable ya que conservó el previo orden entre el producto 1 y el producto 6.

```
for(int n = 100_000; n<=500_000; n+=100_000){
    Producto productList[] = new Producto[n];
    for(int i = 0; i<n; i++){
        productList[i] = new Producto();
    }
    ordenarPorPrecio(productList);
    System.out.println(isProductSorted(productList));
}
```

```
PS C:\Users\eduar>  & 'C:\Users\eduar\.vscode\extensions\vscjava.vscode-java-debug-0.25.1\scripts\launcher.bat' 'C:\Program Files\Java\jdk1.8.0_201\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,
server=n,suspend=y,address=localhost:59295' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\eduar\AppData\Local\Temp\vscodesws_a161d\jdt_ws\jdt.ls-java-project\bin' 'Examen2a'
true
true
true
true
true
```

Dentro de las capturas previas se observa que el algoritmo funciona con arreglos grandes de diferentes tamaños.

2) Implementa el siguiente método que ordene el arreglo de productos de forma no descendente de acuerdo a las existencias del producto, utilizando el algoritmo Conteo como se vio en clase.

static void ordenarPorExistencias(Producto[] productos)

Especificaciones del algoritmo Conteo:

a) El tamaño del arreglo de conteos estará en función de la diferencia entre el número mínimo y máximo de existencias posibles.

b) Recuerda que el algoritmo debe implementarse de manera que sea estable. Para ello, tendrás que crear un arreglo temporal de productos, pero el arreglo recibido deberá contener el ordenamiento definitivo.

```
//EDUARDO ETHANDRAKE CASTILLO PULIDO
//ANDRE SINSEL AYALA

import java.util.Arrays;

class Producto {
   private String nombre;
   private double precio;
   private int existencias;
   public static final double PRECIO_MIN = 3;
   public static final double PRECIO_MAX = 2000;
   public static final int EXISTENCIAS_MIN = 5;
   public static final int EXISTENCIAS_MAX = 50;
   private static int productId = 0;

   public Producto(String n, double p, int e) {
      this.nombre = n;
      this.precio = p;
      this.existencias = e;
   }
   public Producto() {
      this.nombre = String.format("Producto %3d", ++ productId);
      this.precio = PRECIO_MIN + (PRECIO_MAX - PRECIO_MIN) * Math.random();
```

```
        this.existencias = EXISTENCIAS_MIN + (int) ((EXISTENCIAS_MAX - EXISTENCIAS_MIN + 1) * M
ath.random());
    }
    public String getNombre() {
        return nombre;
    }
    public double getPrecio() {
        return precio;
    }
    public int getExistencias() {
        return existencias;
    }
    public String toString() {
        return String.format("(%s, $%7.2f, %2d)", nombre, precio, existencias);
    }
}

public class Examen2b {
    static void ordenarPorExistencias(Producto[] productos){
        int conteo[] = new int[Producto.EXISTENCIAS_MAX-Producto.EXISTENCIAS_MIN+1];
        for(Producto producto:productos){
            conteo[producto.getExistencias()-Producto.EXISTENCIAS_MIN]++;
        }
        for(int i = 1; i< conteo.length;i++){
            conteo[i] += conteo[i-1];
        }
        Producto productosTemp[] = new Producto[productos.length];
        for(int i = productos.length-1 ; i>=0 ; i--){
            Producto x = productos[i];
            int xnum = x.getExistencias()-Producto.EXISTENCIAS_MIN;
            int j = conteo[xnum] - 1;
            productosTemp[j] = x;
            conteo[xnum]--;
        }
        for(int i = 0; i<productos.length; i++){
            productos[i] = productosTemp[i];
        }
    }
    static boolean isProductSorted(Producto productos[]){
        if(productos != null){
            double pastExistence= productos[0].getExistencias();
            for(int i = 1; i<productos.length;i++){
                if(pastExistence > productos[i].getExistencias()){
                    return false;
                }
```

```
        }
        return true;
      }
      return false;
   }
   public static void main(String[] args){
      Producto p1 = new Producto("Producto 1", 22, 32);
      Producto p2 = new Producto("Producto 2", 10, 6);
      Producto p3 = new Producto("Producto 3", 15, 6);
      Producto p4 = new Producto("Producto 4", 200, 30);
      Producto p5 = new Producto("Producto 5", 190, 13);
      Producto p6 = new Producto("Producto 6", 22, 25);
      Producto productList[] = {p1,p2,p3,p4,p5,p6};
      ordenarPorExistencias(productList);
      System.out.println(Arrays.toString(productList));
      System.out.println(isProductSorted(productList));
      /*
      for(int n = 100_000; n<=500_000; n+=100_000){
         Producto productList[] = new Producto[n];
         for(int i = 0; i<n; i++){
            productList[i] = new Producto();
         }
         ordenarPorExistencias(productList);
         System.out.println(isProductSorted(productList));
      }*/
   }
}
```

```
   Producto p1 = new Producto("Producto 1", 22, 32);
   Producto p2 = new Producto("Producto 2", 10, 6);
   Producto p3 = new Producto("Producto 3", 15, 6);
   Producto p4 = new Producto("Producto 4", 200, 30);
   Producto p5 = new Producto("Producto 5", 190, 13);
   Producto p6 = new Producto("Producto 6", 22, 25);
   Producto productList[] = {p1,p2,p3,p4,p5,p6};
   ordenarPorExistencias(productList);
   System.out.println(Arrays.toString(productList));
   System.out.println(isProductSorted(productList));
PS C:\Users\eduar> & 'C:\Users\eduar\.vscode\extensions\vscjava.vscode-java-debug-0.25.1\scripts\launcher.bat' 'C:\Program Files\Java\jdk1.8.0_201\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,
server=n,suspend=y,address=localhost:61094' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\eduar\AppData\Local\Temp\vscodesws_a161d\jdt_ws\jdt.ls-java-project\bin' 'Examen2b'
[(Producto 2, $  10.00,  6), (Producto 3, $  15.00,  6), (Producto 5, $ 190.00, 13), (Producto 6, $  22.00, 25), (Producto 4, $ 200.00, 30), (Producto 1, $  22.00, 32)]
true
```

Como se puede observar dentro de las capturas anteriores, el algoritmo funciona para arreglos pequeños y además es estable ya que los productos 2 y 3 conservan su orden previo al algoritmo.

```java
for(int n = 100_000; n<=500_000; n+=100_000){
    Producto productList[] = new Producto[n];
    for(int i = 0; i<n; i++){
        productList[i] = new Producto();
    }
    ordenarPorExistencias(productList);
    System.out.println(isProductSorted(productList));
}
```

```
PS C:\Users\eduar>  & 'C:\Users\eduar\.vscode\extensions\vscjava.vscode-java-debug-0.25.1\scripts\launcher.bat' 'C:\Program Files\Java\jdk1.8.0_201\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:61120' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\eduar\AppData\Local\Temp\vscodesws_a161d\jdt_ws\jdt.ls-java-project\bin' 'Examen2b'
true
true
true
true
true
```

Dentro de las capturas anteriores se muestra como el algoritmo funciona para distintos arreglos de tamaño grande.

3) Implementa el siguiente método que ordene un arreglo de números enteros positivos de forma no descendente utilizando el algoritmo RadixSort. Deberás utilizar una cola por cada dígito diferente. Sea eficiente para obtener el dígito actual.

Ojo: convertir un número entero a cadena de texto no es nada eficiente.

static void radixSort(int[] array, int M) {

El argumento M representa el número de dígitos del número más grande del arreglo.

```java
//EDUARDO ETHANDRAKE CASTILLO PULIDO
//ANDRE SINSEL AYALA
import java.util.Arrays;
import java.util.LinkedList;

public class Examen2c {
   static void radixSort(int[] array, int M) {
      LinkedList<Integer> digitList[] = new LinkedList[10];
      for(int i = 0; i<10 ; i++){
         digitList[i] = new LinkedList<Integer>();
      }
      //acomodar los elementos por primera vez

      for(int datum : array){
         int digit = datum%10;
         digitList[digit].offer(datum);
      }

      for(int i=1; i<=M ; i++){
         for(LinkedList<Integer> list: digitList) list.offer(-1);

         double exp = 1;
         for(int e=0;e<i;e++){
            exp *= 10;
         }
         for(LinkedList<Integer> list:digitList){
            int datum = list.poll();
            while(datum != -1){
```

```
                int digit = (int)(datum/exp)%10;
                digitList[digit].offer(datum);
                datum = list.poll();
            }
        }
    }
    int index=0;
    for(LinkedList<Integer> list: digitList){
        while(!list.isEmpty()){
            array[index++] = list.poll();
        }
    }
}
static boolean isSorted(int array[]){
    if(array != null){
        int pastDatum = array[0];
        for(int i = 1; i<array.length;i++){
            if(pastDatum > array[i]){
                return false;
            }
        }
        return true;
    }
    return false;
}
static int random(int min, int max) {
    return min + (int) ((max - min + 1) * Math.random());
}

static int[] randomArray(int N, int min, int max) {
    int[] a = new int[N];
    for(int i = 0; i < N; i ++) {
        a[i] = random(min, max);
    }
    return a;
}
public static void main(String[] args){
    /*int array[]={124,1,2,3151,36,58,46,95,26541,235,6984};
    radixSort(array, 5);
    System.out.println(Arrays.toString(array));
    System.out.println(isSorted(array));*/
    for(int n = 100_000; n<=500_000; n+=100_000){
        int array[] = randomArray(n, 0, n);
        radixSort(array, 6);
        System.out.println(isSorted(array));
```

```
    }
  }
}
```

```
int array[]={124,1,2,3151,36,58,46,95,26541,235,6984};
radixSort(array, 5);
System.out.println(Arrays.toString(array));
System.out.println(isSorted(array));
```

```
PS C:\Users\eduar>  & 'C:\Users\eduar\.vscode\extensions\vscjava.vscode-java-debug-0.25.1\scripts\launcher.bat' 'C:\Program Files\Java\jdk1.8.0_201\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,
server=n,suspend=y,address=localhost:62594' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\eduar\AppData\Local\Temp\vscodesws_a161d\jdt_ws\jdt.ls-java-project\bin' 'Examen2c'
[[], [1, 3151, 26541], [2], [], [124, 6984], [95, 235], [36, 46], [], [58], []]
[[1, 2], [], [124], [235, 36], [26541, 46], [3151, 58], [], [], [6984], [95]]
[[1, 2, 36, 46, 58, 95], [124, 3151], [235], [], [], [26541], [], [], [], [6984]]
[[1, 2, 36, 46, 58, 95, 124, 235], [], [], [3151], [], [], [26541, 6984], [], [], []]
[[1, 2, 36, 46, 58, 95, 124, 235, 3151, 6984], [], [26541], [], [], [], [], [], [], []]
[[1, 2, 36, 46, 58, 95, 124, 235, 3151, 6984, 26541], [], [], [], [], [], [], [], [], []]
[1, 2, 36, 46, 58, 95, 124, 235, 3151, 6984, 26541]
true
```

Dentro de las capturas anteriores se puede observar como el algoritmo funciona paso a paso de manera correcta con un arreglo pequeño como entrada.

```
for(int n = 100_000; n<=500_000; n+=100_000){
    int array[] = randomArray(n, 0, n);
    radixSort(array, 6);
    System.out.println(isSorted(array));
}
```

```
PS C:\Users\eduar>  & 'C:\Users\eduar\.vscode\extensions\vscjava.vscode-java-debug-0.25.1\scripts\launcher.bat' 'C:\Program Files\Java\jdk1.8.0_201\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,
server=n,suspend=y,address=localhost:62648' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\eduar\AppData\Local\Temp\vscodesws_a161d\jdt_ws\jdt.ls-java-project\bin' 'Examen2c'
true
true
true
true
true
```

En las capturas previas se muestra que el algoritmo funciona de manera correcta con arreglos grandes por medio de la función isSorted.

4) Implementa el siguiente método que calcule el código hash de una frase que puede estar compuesta por letras minúsculas no acentuadas, el punto, letras mayúsculas no acentuadas y la coma. La eñe no está incluida. Utiliza el método de Horner y aritmética modular.

static void hashCode(String phrase, int M) {

El argumento M representa el número de frases que se esperan almacenar en la tabla hash. Es decir, el código hash deberá estar en el rango [0… M – 1].

El código hash de “A”, “B”, “BA”, “BAA”, “aAA”, “…” deben ser, respectivamente, 0, 1, 54, 2916, 75816, 154492, sin considerar aún el valor de M

```
//EDUARDO ETHANDRAKE CASTILLO PULIDO
//ANDRE SINSEL AYALA
public class Examen2d {
   static int valueOf(char c){
      if(c >= 'A' && c<='Z'){
         return c-'A';
      }else if(c >= 'a' && c<='z'){
         return c-'a'+26;
      }else if(c == '.'){
         return 52;
      }else if(c == ','){
         return 53;
      }
      return 0;
   }
```

```java
    static void hashCode(String phrase, int M) {
        //Base = 26+26+1+1=54
        int base = 54;
        int hash = valueOf(phrase.charAt(0));
        for(int i = 1;i<phrase.length();i++){
            hash = (hash * base + valueOf(phrase.charAt(i))) % M;
        }
        System.out.println(hash);
    }
    public static void main(String[] args){
        String phrase = "Prueba,numero.5";
        int M = 55;
        hashCode(phrase, M);
        phrase="MoScAs";
        M=20;
        hashCode(phrase, M);
        phrase="ElectroEncefAlografista.";
        M=560;
        hashCode(phrase, M);
        phrase="Si,conseguiTrabajo.";
        M=201;
        hashCode(phrase, M);
        phrase="StarWars";
        M=30;
        hashCode(phrase, M);
    }
}
```

```java
String phrase = "Prueba,numero.5";
int M = 55;
hashCode(phrase, M);
phrase="MoScAs";
M=20;
hashCode(phrase, M);
phrase="ElectroEncefAlografista.";
M=560;
hashCode(phrase, M);
phrase="Si,conseguiTrabajo.";
M=201;
hashCode(phrase, M);
phrase="StarWars";
M=30;
hashCode(phrase, M);
```

```
PS C:\Users\eduar>  & 'C:\Users\eduar\.vscode\extensions\vscjava.vscode-java-debug-0.25.1\scripts\launcher.bat' 'C:\Program Files\Java\jdk1.8.0_201\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,
server=n,suspend=y,address=localhost:63155' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\eduar\AppData\Local\Temp\vscodesws_a161d\jdt_ws\jdt.ls-java-project\bin' 'Examen2d'
41
12
20
133
14
```

Dentro de las capturas previas se observa la funcionalidad del algoritmo al utilizarse diferentes cadenas de texto con distinta longitud y valores de M como entradas.

5)Implementa el siguiente método que mezcle el contenido de dos arreglos ordenados Ay Ben A, de forma que A almacene el contenido de ambos de forma ordenada. Puedes suponer que A tiene el espacio necesario y exacto para almacenar los datos de B. El algoritmo debe ser lineal.
static voidmerge(int[]A, int[]B,int[]N)
El argumento N representa el número de elementos de A que tienen datos válidos.
Ejemplo de uso:
int[] A= { 3, 5, 7, 8, 9, 10, 0, 0, 0, 0, 0 };
int[] B= { 1, 2, 4, 5, 6 };
merge(A, B, 6);
// A= {1, 2, 3, 4, 5, 5, 6, 7, 8, 9, 10};

```java
//EDUARDO ETHANDRAKE CASTILLO PULIDO
//ANDRE SINSEL AYALA

import java.util.Arrays;

public class Examen2e {

    public static void merge(int[] A, int[] B, int N) {
        int []aux =new int[N];
        for(int i = 0; i < N ; i++) //Theta(N)
            aux[i] = A[i];

        int auxindex = 0;
        int bindex = 0;

        for(int i = 0; i<A.length ; i++)  //Theta(A.length)
        {
            if(bindex == B .length || (auxindex < N && aux[auxindex] < B[bindex])) {
                A[i] = aux[auxindex];
                auxindex ++ ;
            }
            else {
                A[i] = B[bindex];
                bindex ++ ;
            }

        }

    }

    public static int random(int min, int max) {
        return min + (int) ((max - min + 1) * Math.random());
    }
```

```java
    public static int[] randomArray(int N, int min, int max) {
        int[] a = new int[N];
        for(int i = 0; i < N; i ++) {
            a[i] = random(min, max);
        }
        return a;
    }

    public static boolean isSorted(int[] a) {
        for(int i = 0; i < a.length - 1; i ++) {
            if(a[i] > a[i + 1]) return false;
        }
        return true;
    }

    public static void main(String[] args) {
        int[] A = { 3, 5, 7, 8, 9, 10, 0, 0, 0, 0, 0, 0 };
        int[] B = { 1, 2, 4, 5, 6 };
        merge(A, B, 6);
        System.out.println(Arrays.toString(A));
        System.out.println(isSorted(A));

        /*for(int n = 100_000; n<=500_000; n+=100_000){
            int A[] = randomArray(n, 0, n);
            int B[] = randomArray(n/2,0,n);
            Arrays.sort(A);
            Arrays.sort(B);
            merge(A,B,n/2);
            System.out.println(isSorted(A));
        }*/

    }


}
```

```java
int[] A = { 3, 5, 7, 8, 9, 10, 0, 0, 0, 0, 0, 0 };
int[] B = { 1, 2, 4, 5, 6 };
merge(A, B, 6);
System.out.println(Arrays.toString(A));
System.out.println(isSorted(A));
```

```
PS C:\Users\eduar>  & 'C:\Users\eduar\.vscode\extensions\vscjava.vscode-java-debug-0.25.1\scripts\launcher.bat' 'C:\Program Files\Java\jdk1.8.0_201\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,
server=n,suspend=y,address=localhost:50671' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\eduar\AppData\Local\Temp\vscodesws_79458\jdt_ws\jdt.ls-java-project\bin' 'Examen2e'
[1, 2, 3, 4, 5, 5, 6, 7, 8, 9, 10]
true
```

Como se puede observar en las capturas anteriores, el algoritmo entrega el resultado esperado cuando las entradas corresponden a arreglos pequeños.

```java
for(int n = 100_000; n<=500_000; n+=100_000){
    int A[] = randomArray(n, 0, n);
    int B[] = randomArray(n/2,0,n);
    Arrays.sort(A);
    Arrays.sort(B);
    merge(A,B,n/2);
    System.out.println(isSorted(A));
}
```

```
PS C:\Users\eduar>  & 'C:\Users\eduar\.vscode\extensions\vscjava.vscode-java-debug-0.25.1\scripts\launcher.bat' 'C:\Program Files\Java\jdk1.8.0_201\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,
server=n,suspend=y,address=localhost:50738' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\eduar\AppData\Local\Temp\vscodesws_79458\jdt_ws\jdt.ls-java-project\bin' 'Examen2e'
true
true
true
true
true
```

En las ilustraciones previas se muestra que el algoritmo funciona con el uso de arreglos grandes como entradas gracias a la función isSorted.

6)Implementa el siguiente método que encuentra el elemento menor en un arreglo que estuvo ordenado de forma no decreciente y luego tuvo una rotación. El tiempo de ejecución del algoritmo debe ser logarítmico. Siga la estrategia divide y vencerás de la búsqueda binaria.

```java
static void findMin(int[] array)
```

Ejemplos de arreglos rotados. En todos los casos el mínimo es 1.

```
{ 15, 16, 19, 20, 25,  1,  3,  4,  5,  7, 10, 13 }
{ 25,  1,  3,  4,  5,  7, 10, 14, 15, 16, 19, 20 }
{  4,  5,  7, 10, 14, 15, 16, 19, 20, 23, 25,  1 }
```

```java
//EDUARDO ETHANDRAKE CASTILLO PULIDO
//ANDRE SINSEL AYALA

public class Examen2f {

   public static int findMin(int[] array) {
      return  findMin(array,0,array.length -1);
   }

   static int findMin (int[] array, int left, int right) {
      int m = (left + right) / 2;
      if(m == 0 && array[m] > array[right])
         return array[right];
      if(m == array.length-1 && array[m] > array[left])
         return array[left];
      if(m == right || array[m] < array[m-1] && (array[m] < array[m+1])) {
         return array[m];
      }

      if((array[left]< array[m] && array[m]<array[right]) ||
            (array[left]>array[m] && array[left]>array[m]))
         return findMin(array, left, m - 1);
      else
         return findMin(array, m + 1, right);
```

```
    }

    public static void main(String[] args) {
        int[] arraya = {15, 16, 19, 20, 25, 1, 3, 4, 5, 7, 10, 13};
        System.out.println(findMin(arraya));
        int[] arrayb = {25, 1, 3, 4, 5, 7, 10, 14, 15, 16, 19, 20};
        System.out.println(findMin(arrayb));
        int[] arrayc = {4, 5, 7, 10, 14, 15, 16, 19, 20, 23, 25, 1};
        System.out.println(findMin(arrayc));
    }
}
```

```
int[] arraya = {15, 16, 19, 20, 25, 1, 3, 4, 5, 7, 10, 13};
System.out.println(findMin(arraya));
int[] arrayb = {25, 1, 3, 4, 5, 7, 10, 14, 15, 16, 19, 20};
System.out.println(findMin(arrayb));
int[] arrayc = {4, 5, 7, 10, 14, 15, 16, 19, 20, 23, 25, 1};
System.out.println(findMin(arrayc));
```

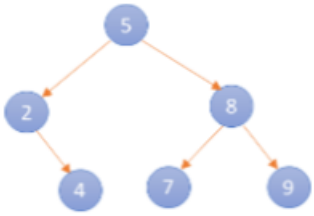
```
PS C:\Users\eduar>  & 'C:\Users\eduar\.vscode\extensions\vscjava.vscode-java-debug-0.25.1\scripts\launcher.bat' 'C:\Program Files\Java\jdk1.8.0_201\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,
server=n,suspend=y,address=localhost:52426' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\eduar\AppData\Local\Temp\vscodesws_c79db\jdt_ws\jdt.ls-java-project\bin' 'Examen2f'
1
1
1
```

Como se puede observar dentro de las capturas anteriores el algoritmo cumple su función, mostrándose así en tres escenarios distintos de arreglos rotados diferentemente contando con el valor mínimo en medio, al inicio y al final.

Para el siguiente ejercicio, hay que crear la clase IntNode:

```
static class IntNode {
   int key;
   int height;
   IntNode left, right;
   public IntNode(int key) {
      this.key = key;
   }
}
```

7)Implementa el siguiente método que determina si existe una ruta de la raíz a alguna hoja, cuya suma de las claves de los nodos visitados es igual a un argumento L.
static boolean existsPath (IntNode node, int L)
Supongamos que no de refiere a la raíz del siguiente árbol binario(5). El método devuelve true para L = 11, 20 y 22; devuelve false para cualquier otro valor de L.




```
//EDUARDO ETHANDRAKE CASTILLO PULIDO
//ANDRE SINSEL AYALA
```

```java
public class BinarySearchTreeInt {

   private class IntNode {
      int key;
      int height;
      IntNode left, right;
      public IntNode(int key) {
         this.key = key;
      }
   }

   private IntNode root = null;

   private void print(IntNode node, String spaces) {
      if(node != null) {
         System.out.println(spaces + "" + node.key);
         print(node.left,  spaces + " ");
         print(node.right, spaces + " ");
      }
   }

   public void print() {
      print(this.root, "");
      System.out.println("-------");
   }

   public boolean add(int key) {
      if(this.root == null) {
         this.root = new IntNode(key);
         return true;
      } else {
         IntNode ite = this.root;
         boolean added = false, found = false;
         while(!found && !added) {
            int res = key-ite.key;
            if(res == 0) {
               found = true;
            } else if(res < 0) {
               if(ite.left != null) {
                  ite = ite.left;
               } else {
                  ite.left = new IntNode(key);
                  added = true;
               }
            } else {
```

```
                if(ite.right != null) {
                    ite = ite.right;
                } else {
                    ite.right = new IntNode(key);
                    added = true;
                }
            }
        }
        return added;
    }
  }
  public boolean existsPath(int L) {
    return existsPath(root, L);
  }
  private static boolean existsPath(IntNode node, int L) {
    IntNode left = node.left;
    IntNode right = node.right;
    boolean leftb = false;
    boolean rightb = false;
    L -= node.key;
    if(left == null & right == null) {
        if(L == 0)
            return true;
    }
    if(L < 0)
        return false;
    if(left != null) {
        leftb = existsPath(left,L);
        if(leftb == true)
            return true;
    }
    if(right != null) {
        rightb = existsPath(right,L);
        if(rightb == true)
            return true;
    }

    return false;

  }
}
```

```java
//EDUARDO ETHANDRAKE CASTILLO PULIDO
//ANDRE SINSEL AYALA

public class Examen2g {

    public static void main(String[] args) {
        BinarySearchTreeInt intBST    = new BinarySearchTreeInt();
        intBST.add(5);
        intBST.add(2);
        intBST.add(4);
        intBST.add(8);
        intBST.add(7);
        intBST.add(9);
        intBST.add(3);
        intBST.add(1);
        intBST.add(11);
        intBST.print();
        System.out.println(intBST.existsPath(11));
        System.out.println(intBST.existsPath(20));
        System.out.println(intBST.existsPath(22));
        System.out.println(intBST.existsPath(33));
        System.out.println(intBST.existsPath(8));
    }
}
```

```
PS C:\Users\eduar>  & 'C:\Users\eduar\.vscode\extensions\vscjava.vscode-java-debug-0.25.1\scripts\launcher.bat' 'C:\Program Files\Java\jdk1.8.0_201\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,
server=n,suspend=y,address=localhost:52929' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\eduar\AppData\Local\Temp\vscodesws_c79db\jdt_ws\jdt.ls-java-project\bin' 'Examen2g'
5
 2
  1
  4
   3
 8
  7
  9
   11
-------
false
true
false
true
true
```

Como se puede observar dentro de la captura anterior, el algoritmo funciona con un árbol binario de 3 niveles y con diferentes valores de L, en este caso se tomaron 5 valores diferentes de L como entradas al algoritmo.

8)Añade los siguientes métodos a la clase BinarySearchTree (disponible en Canvas) que permitan eliminar una clave de un árbol binario de búsqueda, siguiendo el algoritmo visto en clase (compuesto de tres casos).Si la clave no existe, devuelve false; en caso contrario, devuelve true. No olvides actualizar la raíz(this.root), en los casos que sean pertinentes

```java
public class BinarySearchTree<T extends Comparable<T>> {
```

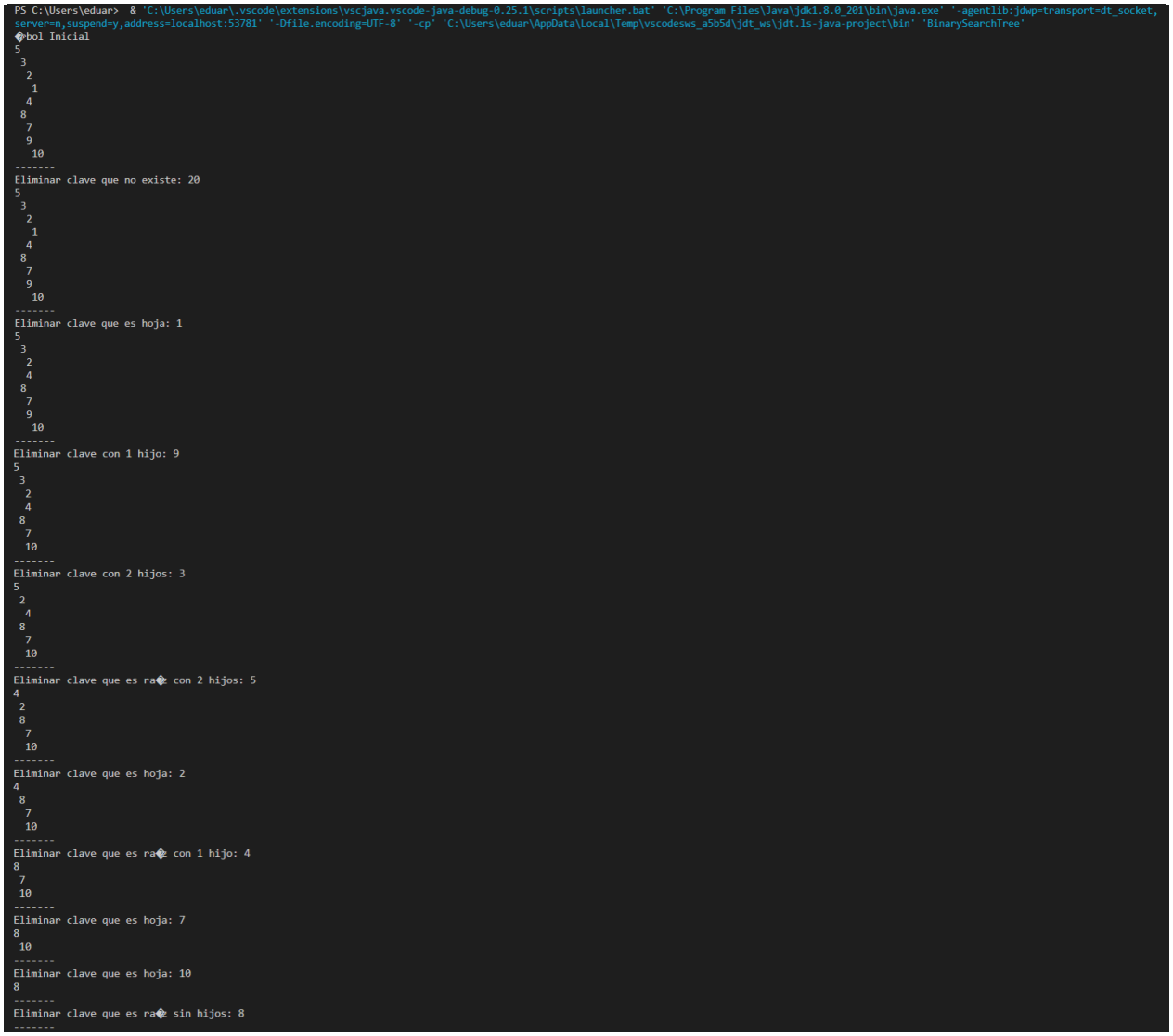
```
    private class TreeNode {
        T key;
        TreeNode left, right, parent;
        public TreeNode(T key) {
            this.key = key;
        }
    }

    private TreeNode root = null;

    private void print(TreeNode node, String spaces) {
        if(node != null) {
            System.out.println(spaces + "" + node.key);
            print(node.left,  spaces + " ");
            print(node.right, spaces + " ");
        }
    }

    public void print() {
        print(this.root, "");
        System.out.println("-------");
    }

    public boolean add(T key) {
        if(this.root == null) {
            this.root = new TreeNode(key);
            return true;
        } else {
            TreeNode ite = this.root;
            boolean added = false, found = false;
            while(!found && !added) {
                int res = key.compareTo(ite.key);
                if(res == 0) {
                    found = true;
                } else if(res < 0) {
                    if(ite.left != null) {
                        ite = ite.left;
                    } else {
                        ite.left = new TreeNode(key);
                        ite.left.parent = ite;
                        added = true;
                    }
                } else {
                    if(ite.right != null) {
                        ite = ite.right;
```

```
                } else {
                    ite.right = new TreeNode(key);
                    ite.right.parent = ite;
                    added = true;
                }
            }
        }
        return added;
    }
}

private TreeNode search(T key, TreeNode current) {
    if(current == null) return null;
    int res = key.compareTo(current.key);
    if(res == 0) {          // key = current.key
        return current;
    } else if(res < 0) {    // key < current.key
        return search(key, current.left);
    } else {                // key > current.key
        return search(key, current.right);
    }
}

public boolean contains(T key) {
    return search(key, this.root) != null;
}

private TreeNode minimum(TreeNode current) {
    if(current.left == null) {  // current contiene a la clave m□nima
        return current;
    } else {                    // current no contiene a la clave m□nima
        return minimum(current.left);
    }
}

public T minimum() {
    if(this.root == null) return null;
    return minimum(this.root).key;
}

private TreeNode maximum(TreeNode current) {
    if(current.right == null) {
        return current;
    } else {
        return maximum(current.right);
```

```
        }
    }

    public T maximum() {
        if(this.root == null) return null;
        return maximum(this.root).key;
    }

    private TreeNode predecessor(TreeNode current) {
        if(current.left != null) {
            return maximum(current.left);
        } else {
            TreeNode parent = current.parent;
            while(parent != null && current == parent.left) {
                current = parent;
                parent  = parent.parent;
            }
            return parent;
        }
    }

    public T predecessor(T key) {
        TreeNode node = search(key, this.root);
        if(node == null) return null;
        TreeNode preNode = predecessor(node);
        if(preNode == null) return null;
        else return preNode.key;
    }
    public boolean delete(T key) {
        TreeNode node;
        node =  search(key, this.root);
        if(node == null)
            return false;
        else {
            delete(node);
            return true;
        }
    }

    private void delete(TreeNode node) {

        if(node.left == null & node.right == null) {
            if(node == this.root)
                this.root = null;
            else if(node.parent.left == node)
```

```
                node.parent.left = null;
            else
                node.parent.right = null;
        }

        else if(node.left != null && node.right == null) {
            if(node == this.root) {
                this.root = node.left;
                this.root.parent = null;
            }
            else if(node.parent.left == node) {
                node.parent.left = node.left;
                node.left.parent = node.parent;
            }
            else {
                node.parent.right = node.left;
                node.left.parent = node.parent;
            }
        }

        else if(node.left == null && node.right != null) {
            if(node == this.root) {
                this.root = node.right;
                this.root.parent = null;
            }
            else if(node.parent.left == node) {
                node.parent.left = node.right;
                node.right.parent = node.parent;
            }
            else {
                node.parent.right = node.right;
                node.right.parent = node.parent;
            }
        }

        else {
            TreeNode pre = predecessor(node);
            node.key = pre.key;
            delete(pre);
        }
    }
    public static void main(String[] args) {
        BinarySearchTree<Integer> intBST    = new BinarySearchTree<>();
        intBST.add(5);
        intBST.add(3);  intBST.add(8);
```

```
        intBST.add(4);  intBST.add(7);  intBST.add(2);  intBST.add(9);
        intBST.add(1);  intBST.add(10);
        System.out.println("�rbol Inicial");
        intBST.print();
        System.out.println("Eliminar clave que no existe: 20");
        intBST.delete(20);
        intBST.print();
        System.out.println("Eliminar clave que es hoja: 1");
        intBST.delete(1);
        intBST.print();
        System.out.println("Eliminar clave con 1 hijo: 9");
        intBST.delete(9);
        intBST.print();
        System.out.println("Eliminar clave con 2 hijos: 3");
        intBST.delete(3);
        intBST.print();
        System.out.println("Eliminar clave que es ra�z con 2 hijos: 5");
        intBST.delete(5);
        intBST.print();
        System.out.println("Eliminar clave que es hoja: 2");
        intBST.delete(2);
        intBST.print();
        System.out.println("Eliminar clave que es ra�z con 1 hijo: 4");
        intBST.delete(4);
        intBST.print();
        System.out.println("Eliminar clave que es hoja: 7");
        intBST.delete(7);
        intBST.print();
        System.out.println("Eliminar clave que es hoja: 10");
        intBST.delete(10);
        intBST.print();
        System.out.println("Eliminar clave que es ra�z sin hijos: 8");
        intBST.delete(8);
        intBST.print();

    }

}
```



Como se muestra dentro de las capturas anteriores, el algoritmo se prueba con un árbol binario de mínimo 3 niveles, 7 nodos y 7 diferentes claves a eliminar con las características marcadas dentro de las rubricas del documento.