ESTRUCTURAS DE DATOS

2024

PARCIAL 1

Profesores:
HECTOR REINAGA
FERNANDA DANIELA OYARZO
MIRTHA FABIANA MIRANDA

Alumno:
GONZALO ALEJANDRO ULLOA

Índice

Desarrollo

1.

a. inicializarMatriz

```
 void inicializarMatriz (int[][] matriz, int N) (
   int i, j;
   for (i=0; i < N; i++)                                  1+1+2
      for (j=0; j < N; j++)                               1+1+2
         if( (i+j) % 2 == 0)                                      3
            matriz[ i ][ j ] = i;                         3
         else
            matriz[ i ][ j ] = i + j;                     4
}
```

b. mostrarMatriz

```
void mostrarMatriz (int[][] matriz, int N) {
   int i, j;
   System.out.println("Matriz: ");                       1
   for (i=0; i < N; i++) {                               1+1+2
      for (j = 0; j < N; j++)                                     1+1+2
         System.out.print(matriz[ i l[ j ]);              3
      System.out.println(" ");                            1
   }
}
```

c. funcionMagica

```
void funcionMagica (int matriz [][10], int N) {
   int i, j;
   for (i = 1; i < N - 1; i++)                           1+2+2
      for (j = i; j < N - 1; j++)                        1+2+2
         matriz[ i ][ j ] = 0;                                    3
}
```

d. Main

```
 // programa principal
public static void main(String[] args){
   int[][] matriz=new int[10][10];                                3
   inicializarMatriz(matriz, 10);                      1
   mostrarMatriz(matriz, 10);                          1
   funcionMagica(matriz, 10);                          1
   mostrarMatriz(matriz, 10);                          1
}
```

a. inicializarMatriz

$$F_1(n) = 1 + 1 + \sum_{i=0}^{N-1}(1 + 1 + \sum_{j=0}^{N-1}(3 + 4 + 1 + 2) + 1 + 2)$$

$$F_1(n) = 2 + \sum_{i=0}^{N-1}(5 + \sum_{j=0}^{N-1}(10))$$

$$F_1(n) = 2 + \sum_{i=0}^{N-1}(5 + (N - 1 - 0 + 1) * (10))$$

$$F_1(n) = 2 + \sum_{i=0}^{N-1}(5 + 10N)$$

$$F_1(n) = 2 + (N - 1 - 0 + 1) * (5 + 10N)$$

$$F_1(n) = 2 + 10N^2 + 5N$$

$$F_1(n) = 10N^2 + 5N + 2 \Rightarrow O(N^2)$$

b. mostrarMatriz

$$F_2(n) = 1 + 1 + 1 + \sum_{i=0}^{N-1}(1 + 1 + \sum_{j=0}^{N-1}(3 + 1 + 2) + 1 + 2 + 1)$$

$$F_2(n) = 3 + \sum_{i=0}^{N-1}(6 + \sum_{j=0}^{N-1}(6))$$

$$F_2(n) = 3 + \sum_{i=0}^{N-1}(6 + (N - 1 - 0 + 1) * (6))$$

$$F_2(n) = 3 + \sum_{i=0}^{N-1}(6 + 6N)$$

$$F_2(n) = 3 + (N - 1 - 0 + 1) * (6 + 6N)$$

$$F_2(n) = 6N^2 + 6N + 3 \Rightarrow O(N^2)$$

c. funcionMagica

$$F_3(n) = 1 + 2 + \sum_{i=1}^{N-1-1}(1 + 2 + \sum_{j=i}^{N-1-1}(3))$$

$$F_3(n) = 3 + \sum_{i=1}^{N-2}(3 + \sum_{j=i}^{N-2}(3))$$

$$F_3(n) = 3 + \sum_{i=1}^{N-2}(3 + (N - 2 - i + 1) * (3))$$

$$F_3(n) = 3 + \sum_{i=1}^{N-2}(3 + (3N - 3i - 3))$$

$$F_3(n) = 3 + \sum_{i=1}^{N-2}(3N - 3i)$$

$$F_3(n) = 3 + \sum_{i=1}^{N-2} 3N + \sum_{i=1}^{N-2} -3i$$

$$F_3(n) = 3 + 3\sum_{i=1}^{N-2} N - 3\sum_{i=1}^{N-2} i$$

$$F_3(n) = 3 + 3 * (N - 2 - 1 + 1) * N - 3 * (\frac{(N-2-1+1)*(N-2+1)}{2})$$

$$F_3(n) = 3 + 3 * (N - 2) * N - 3 * (\frac{N^2-3N+2}{2})$$

$$F_3(n) = 3 + 3 * (N^2 - 2N) - 3 * (\frac{1}{2}N^2 - \frac{3}{2}N + 1)$$

$$F_3(n) = 3 + 3N^2 - 6N - \frac{3}{2}N^2 + \frac{9}{2}N - 3$$

$$F_3(n) = \frac{3}{2}N^2 - \frac{3}{2}N$$

d. Main

$$F_4(n) = 3 + 1 + 1 + 1 + 1$$

$$F_4(n) = 7$$

Total

$$F(n) = F_1 + F_2 + F_3 + F_4$$

$$F(n) = 10N^2 + 5N + 2 + 6N^2 + 6N + 3 + \frac{3}{2}N^2 - \frac{3}{2}N + 7$$

$$F(n) = \frac{35}{2}N^2 + \frac{19}{2}N + 12 \Rightarrow O(N^2)$$

Complejidad cuadrática

2. Ordenamiento

Clase Quicksort

```java
public class QuickSort{
    private static int pasos = 1; // Variable para contar los
pasos
    public static void quickSort(int[] vec) {
        quickSort(vec, 0, vec.length - 1);
    }
    private static void quickSort(int[] vec, int first, int last)
{
        if (first < last) {
            int center = division(vec, first, last);
            System.out.print("Paso " + (pasos++) + ": ");
            printArray(vec);
            quickSort(vec, first, center - 1);
            quickSort(vec, center + 1, last);
```

```
        }
    }
    private static int division(int[] vec, int first, int last) {
        int left = first + 1;
        int right = last;
        int pivot = vec[first];
        System.out.println("Pivote seleccionado: " + pivot);
        while (left <= right) {
            while (left <= right && vec[left] < pivot) {
                System.out.println("Comparando " + vec[left] + " <
" + pivot + ": true");
                left++;
            }
            if (left <= right) {
                System.out.println("Comparando " + vec[left] + " <
" + pivot + ": false");
            }

            while (left <= right && vec[right] > pivot) {
                System.out.println("Comparando " + vec[right] + "
> " + pivot + ": true");
                right--;
            }
            if (left <= right) {
                System.out.println("Comparando " + vec[right] + "
> " + pivot + ": false");
            }
            if (left < right) {
                System.out.println("Intercambiando " + vec[left] +
" con " + vec[right]);
                interChange(vec, left, right);
                left++;
                right--;
            }
        }
        System.out.println("Colocando el pivote en la posición " +
right);
        interChange(vec, first, right);
        return right;
    }
```

```
    private static void interChange(int[] vec, int left, int
right) {
        int aux = vec[left];
        vec[left] = vec[right];
        vec[right] = aux;
    }
    private static void printArray(int[] vec) {
        for (int i : vec) {
            System.out.print(i + " ");
        }
        System.out.println();
    }
}
```

Main

```
public class Main {
    public static void main(String[] args) {
        int[] vec = {40, 5, 20, 45, 30, 15, 25, 10, 35, 0};
        System.out.print("Arreglo inicial:     ");
        printArray(vec);
        QuickSort.quickSort(vec);
        System.out.println("Arreglo final: ");
        printArray(vec);
    }
    private static void printArray(int[] vec) {
        for (int i : vec) {
            System.out.print(i + " ");
        }
        System.out.println();
    }
}
```

Resultado de consola

```
Arreglo inicial:     40 5 20 45 30 15 25 10 35 0
Pivote seleccionado: 40
Comparando 5 < 40: true
Comparando 20 < 40: true
Comparando 45 < 40: false
Comparando 0 > 40: false
Intercambiando 45 con 0
Comparando 30 < 40: true
Comparando 15 < 40: true
Comparando 25 < 40: true
Comparando 10 < 40: true
Comparando 35 < 40: true
Colocando el pivote en la posición 8
Paso 1: 35 5 20 0 30 15 25 10 40 45
Pivote seleccionado: 35
Comparando 5 < 35: true
Comparando 20 < 35: true
Comparando 0 < 35: true
Comparando 30 < 35: true
Comparando 15 < 35: true
Comparando 25 < 35: true
Comparando 10 < 35: true
Colocando el pivote en la posición 7
Paso 2: 10 5 20 0 30 15 25 35 40 45
Pivote seleccionado: 10
Comparando 5 < 10: true
Comparando 20 < 10: false
Comparando 25 > 10: true
Comparando 15 > 10: true
Comparando 30 > 10: true
Comparando 0 > 10: false
Intercambiando 20 con 0
Colocando el pivote en la posición 2
Paso 3: 0 5 10 20 30 15 25 35 40 45
Pivote seleccionado: 0
Comparando 5 < 0: false
Comparando 5 > 0: true
```

```
Colocando el pivote en la posición 0
Paso 4: 0 5 10 20 30 15 25 35 40 45
Pivote seleccionado: 20
Comparando 30 < 20: false
Comparando 25 > 20: true
Comparando 5 > 0: true
Colocando el pivote en la posición 0
Paso 4: 0 5 10 20 30 15 25 35 40 45
Pivote seleccionado: 20
Comparando 30 < 20: false
Comparando 25 > 20: true
Comparando 15 > 20: false
Intercambiando 30 con 15
Colocando el pivote en la posición 4
Paso 5: 0 5 10 15 20 30 25 35 40 45
Pivote seleccionado: 30
Comparando 25 < 30: true
Comparando 5 > 0: true
Colocando el pivote en la posición 0
Paso 4: 0 5 10 20 30 15 25 35 40 45
Pivote seleccionado: 20
Comparando 30 < 20: false
Comparando 25 > 20: true
Paso 4: 0 5 10 20 30 15 25 35 40 45
Pivote seleccionado: 20
Comparando 30 < 20: false
Comparando 25 > 20: true
Comparando 25 > 20: true
Comparando 15 > 20: false
Intercambiando 30 con 15
Colocando el pivote en la posición 4
Paso 5: 0 5 10 15 20 30 25 35 40 45
Pivote seleccionado: 30
Comparando 25 < 30: true
Colocando el pivote en la posición 6
Paso 6: 0 5 10 15 20 25 30 35 40 45
Arreglo final:
0 5 10 15 20 25 30 35 40 45
PS C:\Users\GonzaloUlloa\Desktop\gon\EDA>
```

3. Búsqueda

La función hash utilizada es la del módulo y para la resolución de colisiones utilizó el doble direccionamiento hash.

Clase Automovil

```
class Automovil {
    private String patente;
    private String marca;
    public Automovil(String patente, String marca) {
        this.patente = patente;
        this.marca = marca;
    }
    public String getPatente() {
        return this.patente;
    }
    public void setPatente(String patente) {
        this.patente = patente;
    }
    public String getMarca() {
        return this.marca;
    }
    public void setMarca(String marca) {
        this.marca = marca;
    }
    @Override
    public String toString() {
        return "Patente: " + patente + ", Marca: " + marca;
    }
}
```

Clase TablaHashAutomovil

```
class TablaHashAutomovil {
    private Automovil[] tabla;
    private int tam;
    public TablaHashAutomovil(int tam) {
        this.tam = tam;
        tabla = new Automovil[tam];
    }
    private int hash1(int clave){
        return clave % tam;
    }
    private int hash2(int clave){
        return (7 * clave + 1) % tam; //H1(K) = (7K + 1) % tam
    }
    public void agregar(Automovil auto) {
        int clave = Math.abs(auto.getPatente().hashCode());
        int indice = hash1(clave);  //hash1
        int rehash = hash2(clave); // hash 2

        while (tabla[indice] != null) {
            indice = (indice + rehash) % tam;
        }
        tabla[indice] = auto;
    }
    public void mostrarAutomoviles(){
        for (int i = 0; i < tam; i++){
            if (tabla[i] != null){
                System.out.println("Posicion " + i + ":\n" +
tabla[i]);
            }
        }
    }
}
```

Main

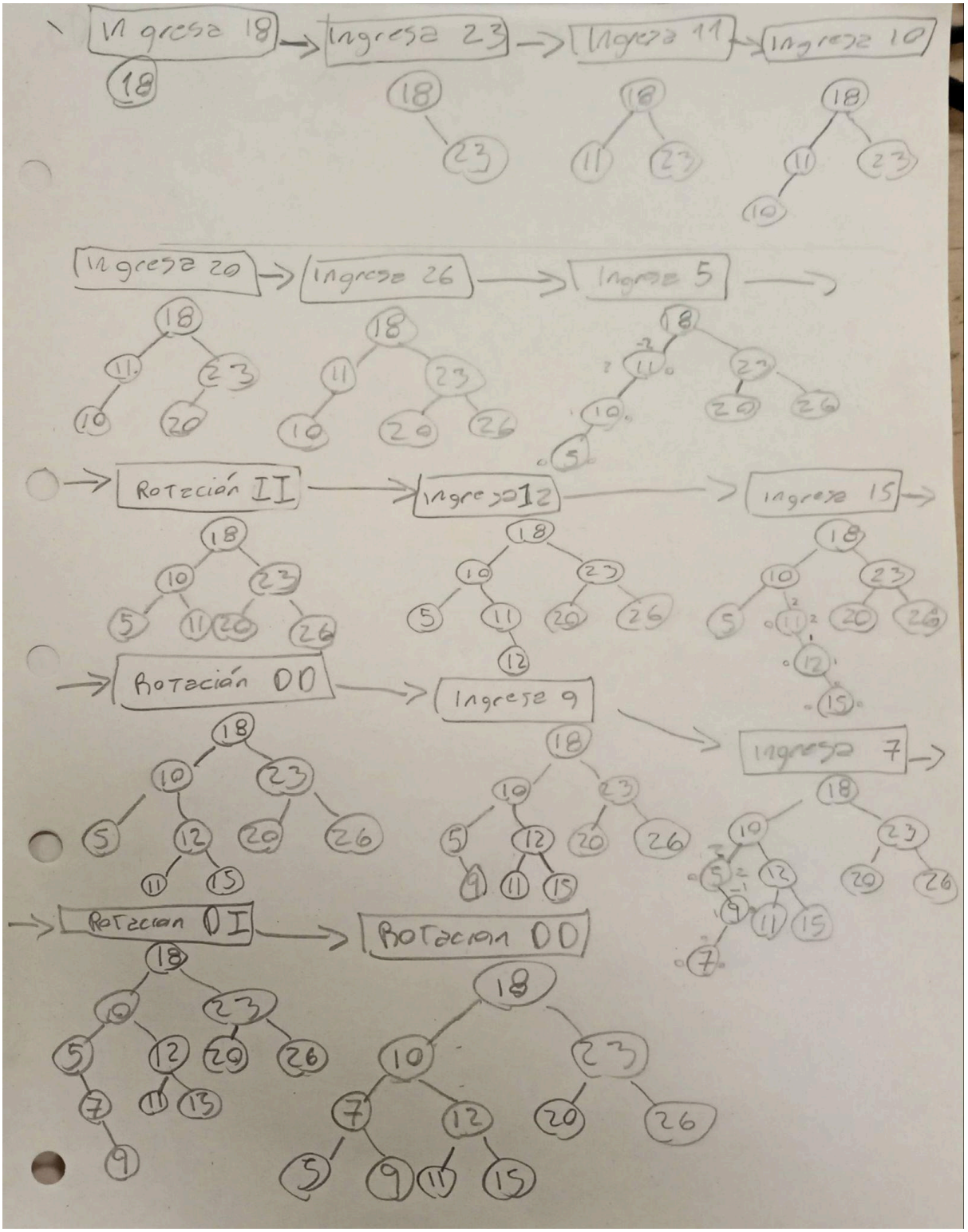
```
class Main {
    public static void main(String[] args) {
        TablaHashAutomovil tabla = new TablaHashAutomovil(10);
        Automovil auto1 = new Automovil("ABC123", "Toyota");
        Automovil auto2 = new Automovil("DEF456", "Chevrolet");
        Automovil auto3 = new Automovil("XYZ789", "Ford");
        Automovil auto4 = new Automovil("JKL999", "Fiat");
        tabla.agregar(auto1);
        tabla.agregar(auto2);
        tabla.agregar(auto3);
        tabla.agregar(auto4);
        tabla.mostrarAutomoviles();
    }
}
```

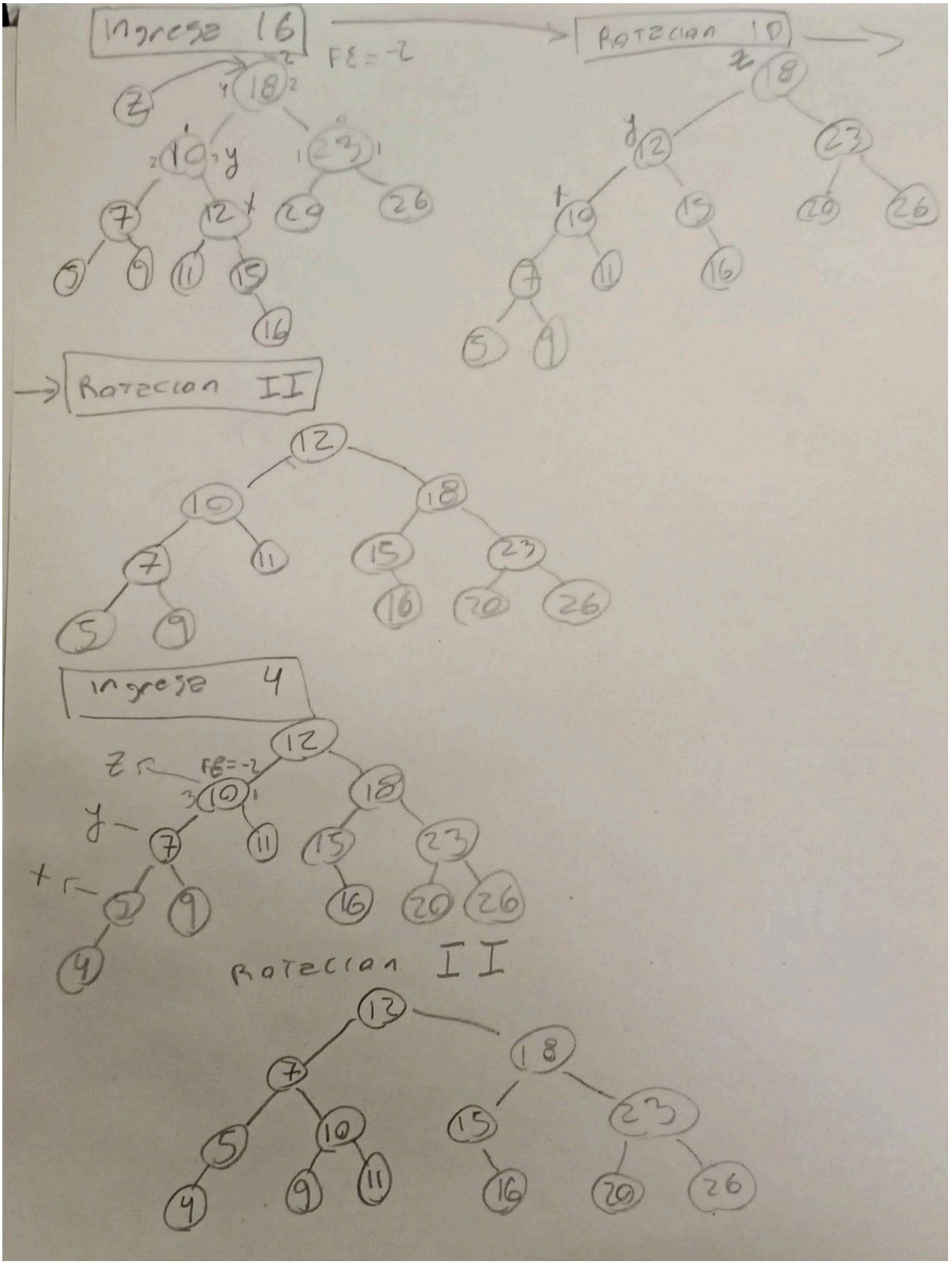
Resultado de consola:

```
Posicion 0:
Patente: JKL999, Marca: Fiat
Posicion 1:
Patente: XYZ789, Marca: Ford
Posicion 6:
Patente: DEF456, Marca: Chevrolet
Posicion 8:
Patente: ABC123, Marca: Toyota
PS C:\Users\GonzaloUlloa\Desktop\gon\EDA>
```

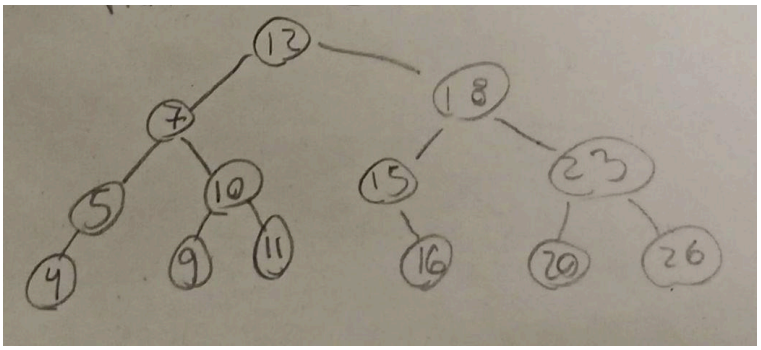
4. AVL

Inserciones



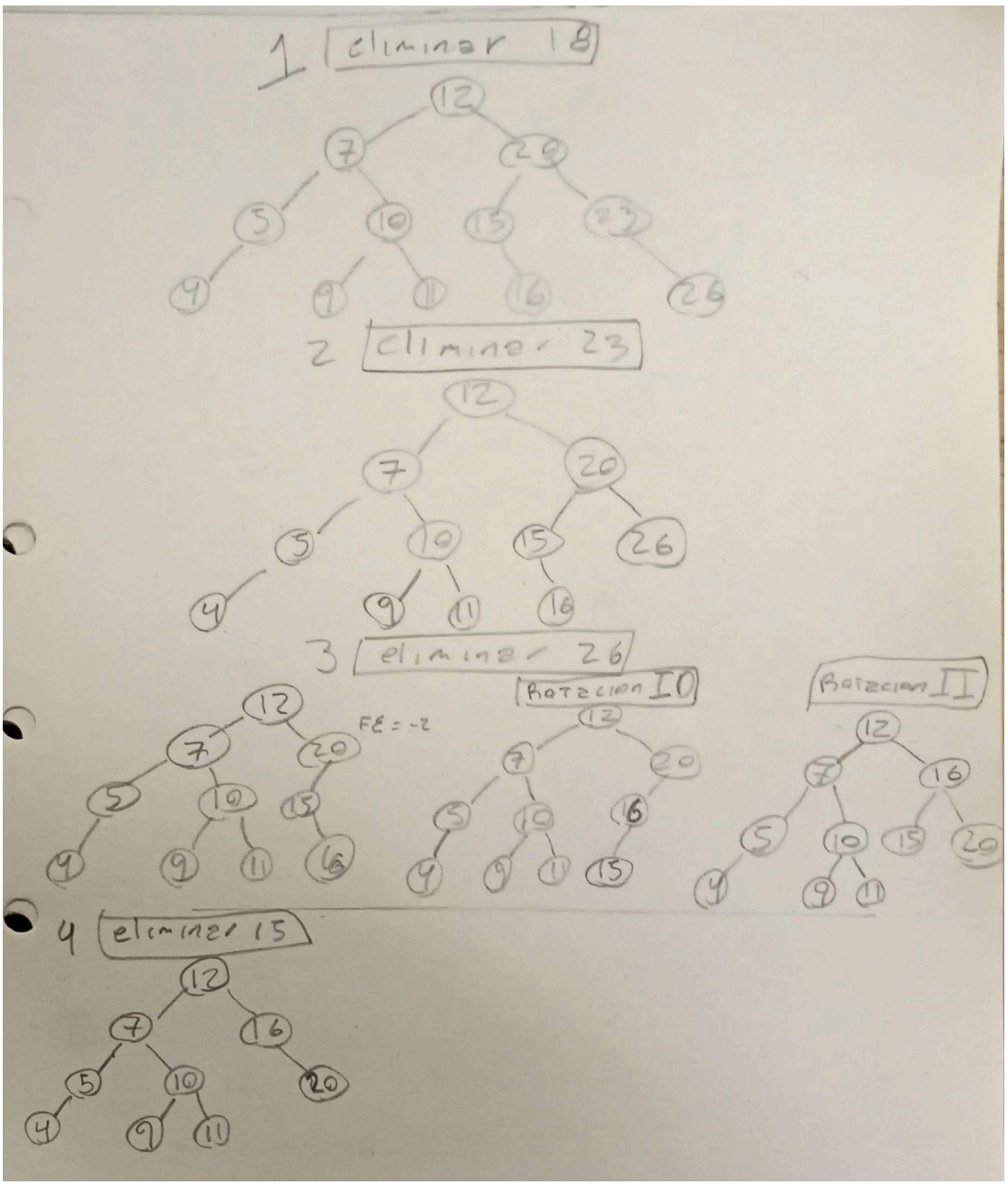

Ingrese 16
FE=-2
Rotacion ID
Rotacion II
Ingrese 4
Rotacion II

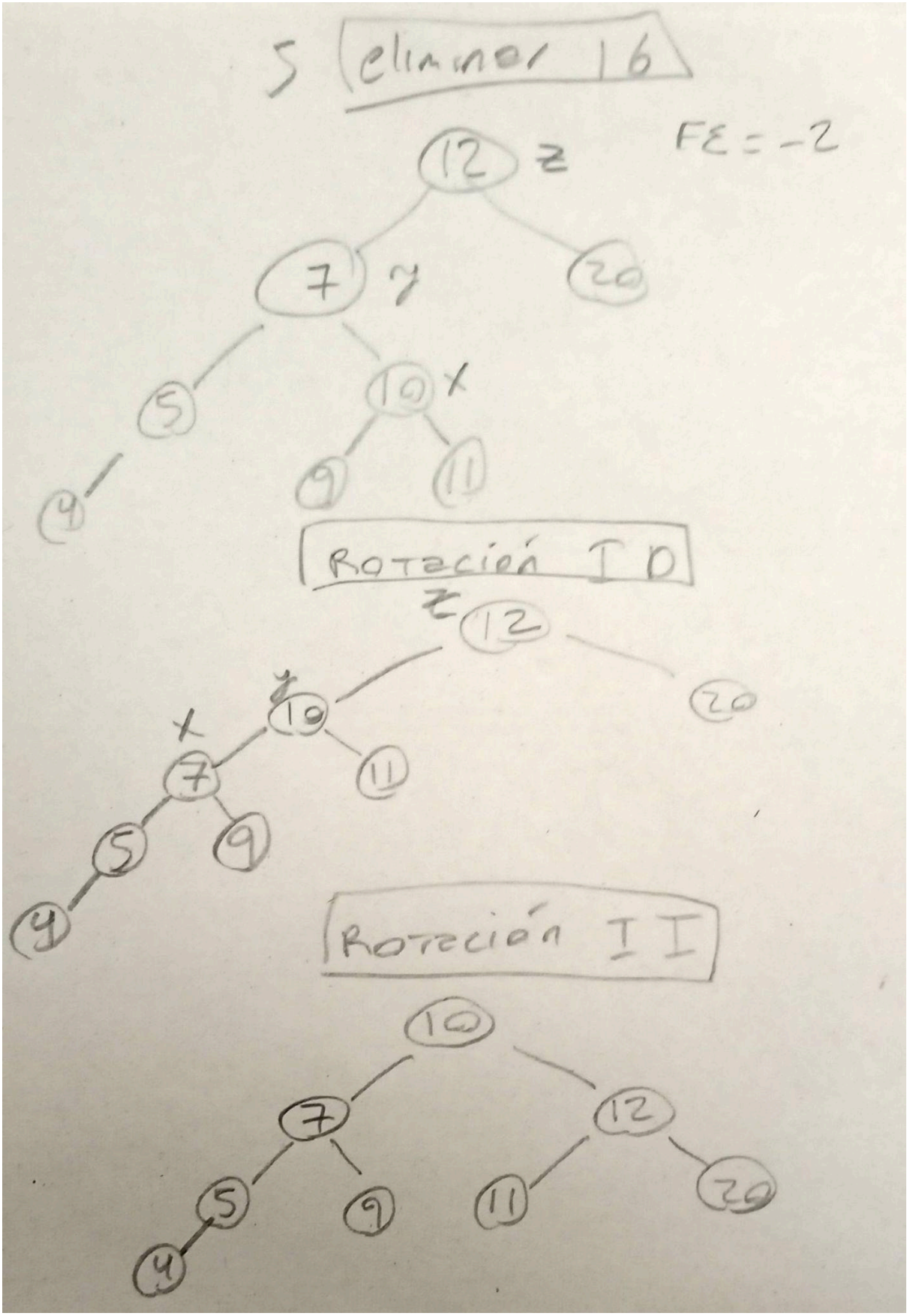
Resultado



Eliminación

(Para la eliminación de un nodo con dos hijos utilizo de reemplazo el elemento más izquierdo del subarbol derecho)




5 eliminar 16
FE = -2
12 z
7 y
20
5
10 x
4
9
11
ROTACIÓN I D
z
12
20
x
10
7
11
5
9
4
ROTACIÓN I I
10
7
12
5
9
11
20
4