Análisis de Complejidades

Pila.java

```
public class Pila implements Cloneable{
   private Nodo top;                                                    1
   private int cant, tiempo;                                            2
```

T(n) = 3

```
   public Pila(){
      top = null;                                                       1
      cant = 0;                                                         1
   }
```

T(n) = 5

```
   public int getCant() {
      return cant;                                                      1
   }

   public int getTiempo() {
      return tiempo;                                                    1
   }

   public void setTiempo(int t) {
      this.tiempo = t;                                                  1
   }
```

T(n) = 3

```
   public void push(Nodo pass){                                         1
      Nodo temp = new Nodo(pass.tel, pass.nombre, pass.inicio, pass.fin);   5
      if (temp == null) {
         System.out.print("\nHeap Overflow");
         return;                                                        1
      }
      temp.right = top;                                                 1
      top = temp;                                                       1
      cant++;                                                           n
   }
```

T(n) = n + 9

```
   public void pop(){
      if (top == null) {
         return;                                                        1
```

```
    }
    top = (top).right;                                                    1
  }
```

T(n) = 2

```
  public void print(JTextPane console){
    Nodo temp = top;                                                      2
    while(temp != null){
      //System.out.println(temp.nombre);
      console.setText(console.getText() + "\n" + temp.nombre + "     "
          + Conversion.timeToString(temp.inicio) + " - " + Conversion.timeToString(temp.fin));
      temp = temp.right;
    }
  }
```

T(n) = 2

```
  public Nodo getIndex(int i){                                            1
    Nodo temp = top; int index = 0;                                       4
    while(temp != null){
      //System.out.println(temp.nombre);
      if(index == i){
        return temp;                                                      1
      }
      index++;                                                            n
      temp = temp.right;                                                  1
    }
    return null;
  }
```

T(n) = n + 8

```
  public Pila clone() throws CloneNotSupportedException {
    return (Pila) super.clone();                                          1
  }
}
```

T(n) = 1

Nodo.java

```
public class Nodo implements Serializable {
   String tel;                                                    1
   String nombre;                                                 1
   int inicio, fin;                                               2
   Nodo left, right;                                              2
```

T(n) = 6

```
   Nodo(String tel, String nombre, int inicio, int fin){          4
      this.tel = tel;                                             1
      this.nombre = nombre;                                       1
      this.inicio = inicio;                                       1
      this.fin = fin;                                             1
      this.left = null;                                           1
      this.right = null;                                          1
   }
```

T(n) = 10

```
   public String getTel() {
      return tel;                                                 1
   }

   public String getNombre() {
      return nombre;                                              1
   }

   public int getInicio() {
      return inicio;                                              1
   }

   public int getFin() {
      return fin;                                                 1
   }

   public int getTiempo(){
      return fin-inicio;                                          1
   }

}
```

T(n) = 5

Conversion.java

```
public class Conversion {

   public static int stringToTime(String sTime){                    1
      int timeInMins  =   0;                                        2
      String[] strArr =   sTime.split(":");                         2
      int hour        =   Integer.parseInt(strArr[0]);              2
      int min         =   Integer.parseInt(strArr[1]);              2

      timeInMins = hour*60 + min;                                   1

      return timeInMins;                                            1
   }

   public static String timeToString(int sTime){                    1
      String time;                                                  1
      int hour = sTime/60, min = sTime%60;                          2
      String min2;                                                  1
      if(min == 0){
         time = hour + ":00";                                       1
      } else if(min < 10){
         time = hour + ":0" + min;                                  1
      } else {
         time = hour + ":" + min;                                   1
      }
      if(hour < 10){
         time = "0" + time;                                         1
      }
      return time;                                                  1
   }

}
```

Quicksort.java

```
public class QuickSort {
   private static void swap(Nodo[] arr, int i, int j){                3
      Nodo temp = arr[i];                                             2
      arr[i] = arr[j];                                                1
      arr[j] = temp;                                                  1
   }
```

T(n) = 7

```
   private static int partition(Nodo[] arr, int low, int high){       3

      Nodo pivot = arr[high];                                         2
      int i = (low - 1);                                              1

      for (int j = low; j <= high - 1; j++) {                         2 + n + 1 + n
         if (arr[j].getFin() < pivot.getFin()) {
            i++;                                                      n
            swap(arr, i, j);                                          1
         }
      }
      swap(arr, i + 1, high);                                         1
      return (i + 1);                                                 1
   }
```

T(n) = 3n + 12 = O(n)

```
   public static void quickSort(Nodo[] arr, int low, int high){       3
      if (low < high) {
         int pi = partition(arr, low, high);                          1 + n

         quickSort(arr, low, pi - 1);                                 n/2
         quickSort(arr, pi + 1, high);                                n/2
      }
   }
```

T(n) = 4 + n + n/2 + n/2 =n + 2(n/2) + 4

T(n) = T(n) + T(n/2) + c

T(n) = n.logn + c

T(n) = O(n.logn)

```
   public static void quickSort(Nodo[] arr){                          1
      int n = arr.length;                                             2
      quickSort(arr, 0, n-1);                                         n.logn
   }
}
```

Árbol.java

```
public class Arbol implements Serializable {
   private Nodo raiz;                                                    1
   private int cant;                                                     1
```

T(n) = 2

```
   public Arbol(){
      raiz = null;                                                       1
      cant = 0;                                                          1
   }
```

T(n) = 2

```
   public int getCant() {
      return cant;                                                       1
   }

   public int comparar(String s1, String s2){
      return s1.compareTo(s2);                                           1
   }
```

T(n) =2

```
   private Nodo addR(Nodo now, String tel, String nombre, int inicio, int fin){   5
      if(now == null){
         return new Nodo(tel, nombre, inicio, fin);                      1
      }
      if(comparar(nombre, now.nombre) < 0){
         now.left = addR(now.left, tel, nombre, inicio, fin);            1
      } else if(comparar(nombre, now.nombre) > 0){
         now.right = addR(now.right, tel, nombre, inicio, fin);          1
      } else {
         return now;                                                     1
      }
      return now;                                                        1
   }
```

T(n) = 10

```
   public void add(String tel, String nombre, int inicio, int fin){
      raiz = addR(raiz, tel, nombre, inicio, fin);                       1
      cant++;                                                            n
```

```
    }
```

T(n) = n + 1

```
    private Nodo buscarR(Nodo now, String nombre){                    2
        if(now == null){
            return null;                                              1
        }
        if(comparar(nombre, now.nombre) == 0){
            return now;                                               1
        }
        if(comparar(nombre, now.nombre)<0){
            return buscarR(now.left, nombre);                         1
        } else {
            return buscarR(now.right, nombre);                        1
        }
    }
```

T(n) = 6

```
    public Nodo buscar(String nombre){
        return buscarR(raiz, nombre);                                 1
    }
```

T(n) = 1

```
    private Nodo borrarR(Nodo now, String nombre){                    2
        if(now == null){
            return null;                                              1
        }
        if(comparar(nombre, now.nombre) == 0){
            if(now.left == null && now.right == null){
                return null;                                          1
            }
            if(now.right == null){
                return now.left;                                      1
            }
            if(now.left == null){
                return now.right;                                     1
            }
            String menor = findMin(now.right);                        2
            now.nombre = menor;                                       1
            now.right = borrarR(now.right, menor);                    1
            return now;                                               1
```

```
      }
      if(comparar(nombre, now.nombre) < 0){
         now.left = borrarR(now.left, nombre);                    1
      }
      now.right = borrarR(now.right, nombre);                     1
      return now;                                                 1
   }
```

T(n) = 14

```
   public void borrar(String nombre){                             1
      raiz = borrarR(raiz, nombre);                               1
      cant--;                                                     n
   }
```

T(n) = n + 2

```
   private String findMin(Nodo raiz){                             1
      if(raiz.left == null){
         return raiz.nombre;                                      1
      } else {
         return findMin(raiz.left);                               1
      }
   }
```

T(n) = 3

```
   private int mostrarR(Nodo raiz, int cont, JTextPane instruccion){   3
      if(raiz.left != null){
         cont = mostrarR(raiz.left, cont, instruccion);           1
      }
      if(raiz.right != null){
         cont = mostrarR(raiz.right, cont, instruccion);          1
      }
      instruccion.setText( instruccion.getText() + cont + ") " + raiz.nombre + "\t   " + raiz.tel +
"\t\t"
            + Conversion.timeToString(raiz.inicio) + "  -  " + Conversion.timeToString(raiz.fin) +
"\n" );                                                           1
      return cont+1;                                              1
   }
```

T(n) = 7

```
   public void mostrar(JTextPane instruccion){                    1
      Nodo rar = this.raiz;                                       1
```

```
      //System.out.println("#\tnombre\t\ttelefono\t\tinicio\t\tfin\n");
      instruccion.setText("");                                          1
      mostrarR(rar, 1, instruccion);                                    1
   }
```

T(n) = 4

```
   private Nodo editarR(int opc, Nodo now, String nombre, String nuevo, String tel, int inicio,
int fin){                                                               7
      if(now == null){
         return null;                                                   1
      }
      if(comparar(nombre, now.nombre) == 0){
         switch (opc) {
            case 2 -> now.tel = tel;                                    1
            case 3 -> now.inicio = inicio;                              1
            case 4 -> now.fin = fin;                                    1
         }
         return now;
      }
      if(comparar(nombre, now.nombre) < 0){
         now.left = editarR(opc, now.left, nombre, nuevo, tel, inicio, fin);  1
      }
      now.right = editarR(opc, now.right, nombre, nuevo, tel, inicio, fin);   1
      return now;                                                       1
   }
```

T(n) = 14

```
   public void editar(int opc, String nombre, String nuevo, String tel, int inicio, int fin){    6
      if (opc==1 && !nombre.equals(nuevo)) {
         Nodo temp = buscar(nombre);                                    2
         add(temp.tel, nuevo, temp.inicio, temp.fin);                   n
         borrar(nombre);                                                n
         return;                                                        1
      } else if (opc == 1){
         return;                                                        1
      }
      raiz = editarR(opc, raiz, nombre, nuevo, tel, inicio, fin);       1
   }
```

T(n) = 2n + 5

```
   private static int toArrayR(Nodo raiz, int cont, Nodo[] arr){       3
      if(raiz.left != null){
```

```
        cont = toArrayR(raiz.left, cont, arr);                    n
      }
      if(raiz.right != null){
        cont = toArrayR(raiz.right, cont, arr);                   n
      }
      arr[cont] = raiz;                                           1
      return cont+1;                                              1
    }
```

T(n) = 2n + 5

```
    public Nodo[] toArray(){
      Nodo rar = this.raiz;                                       2
      Nodo[] arr = new Nodo[cant];                                2
      toArrayR(rar, 0, arr);                                      n
      return arr;                                                 1
    }
```

T(n) = n + 5

```
    private int setCant(Nodo raiz, int cont){                     2
      if(raiz != null){
        cont = setCant(raiz.left, cont);                          n
        cont++;                                                   n
        cont = setCant(raiz.right, cont);                         n
      }
      return cont;                                                1
    }
```

T(n) = 3n + 3

```
    public void guardar() throws IOException, ClassNotFoundException {
      Nodo temp = raiz;                                           2
      cargar(1);                                                  n
      File f = new File("contactos.txt");                         2
      ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream("contactos.txt"));                               2
      out.writeObject(raiz);                                      1
      out.close();                                                1
      raiz = temp;                                                1
      JOptionPane.showConfirmDialog(null, "Los contactos fueron exportados",
          "Contactos", JOptionPane.DEFAULT_OPTION);               1
    }
```

T(n) = n + 10

```
    public void cargar(int op) throws ClassNotFoundException, IOException {      1
```

```
    ObjectInputStream in = new ObjectInputStream(new FileInputStream("contactos.txt"));   2
    Nodo temp = raiz;                                                                     2
    raiz = (Nodo)in.readObject();                                                         1
    if(temp != null){
      cargarR(temp);                                                                      n
    }
    in.close();                                                                           1
    Nodo r = raiz;                                                                        2
    cant = setCant(r, 0);                                                                 1
    if(op == 0){
      JOptionPane.showConfirmDialog(null, "Los contactos fueron importados",
          "Contactos", JOptionPane.DEFAULT_OPTION);                                       1
    }
  }
```

T(n) = n + 9

```
  private void cargarR(Nodo temp){                                                        1
    if(temp.left != null){
      add(temp.tel, temp.nombre, temp.inicio, temp.fin);                                  n
      cargarR(temp.left);                                                                 n
    }
    if(temp.right != null){
      add(temp.tel, temp.nombre, temp.inicio, temp.fin);                                  n
      cargarR(temp.right);                                                                n
    }
    add(temp.tel, temp.nombre, temp.inicio, temp.fin);                                    n
  }

}
```

T(n) = 5n + 1

MaxContactos.java

```
public class MaxContactos {
   private static int busquedaBin(Nodo[] arr, int i){                    2
      int min = 0, max = i-1;                                            4
      while(min <= max){                                                 n
         int mid = (min + max)/2;                                        2
         if(arr[mid].getFin() <= arr[i].getInicio()){
            if(arr[mid+1].getFin() <= arr[i].getInicio()){
               min = mid+1;                                              1
            } else {
               return mid;                                               1
            }
         } else {
            max = mid-1;                                                 1
         }
      }
      return -1;                                                         1
   }
T(n) = n + 12

   static class Aux{
      Pila aux;                                                          1
      int tiempo;                                                        1

      public Aux(){
         aux = new Pila();                                               1
         tiempo = 0;                                                     1
      }

      public Pila getAux() {
         return aux;                                                     1
      }

      public void setAux(Pila aux) {
         this.aux = aux;                                                 1
      }

      public int getTiempo() {
         return tiempo;                                                  1
      }

      public void setTiempo(int tiempo) {
         this.tiempo = tiempo;                                           1
      }
   }
T(n) = 8
```

```
   public static void maxContactos(Nodo[] arr, JTextPane console) throws
CloneNotSupportedException {                                              2
      QuickSort.quickSort(arr);                                           n.logn
      int n = arr.length;                                                 1

      LinkedList<Aux> lis = new LinkedList<>();                           1
      for(int i=0; i<n; i++){                                             5
         lis.add(new Aux());                                              n
      }

      lis.get(0).setTiempo(arr[0].getTiempo());                           1
      lis.get(0).aux.push(arr[0]);                                        n

      for(int i=1; i<n; i++){                                             5
         int tiempo = arr[i].getTiempo();                                 2
         int l = busquedaBin(arr, i);                                     2
         if(l != -1){
            tiempo += lis.get(l).getTiempo();                             1
         }
         if(tiempo > lis.get(i-1).tiempo){
            lis.get(i).setTiempo(tiempo);                                 1
            lis.get(i).setAux(lis.get(l).getAux().clone());               1
            lis.get(i).getAux().push(arr[i]);                             n

         } else {
            lis.get(i).setTiempo(lis.get(i-1).getTiempo());               1
            lis.get(i).setAux(lis.get(i-1).getAux().clone());             1
         }
      }
      console.setText(console.getText() + "\nMaximo tiempo = "+
Conversion.timeToString(lis.get(n-1).getTiempo())+" horas.\n");
      lis.get(n-1).getAux().print(console);                               4
      console.setText(console.getText() + "\n");                          1

   }

}
```

T(n) = n.logn + n + 29

T(n) = O(n.logn)