Manual Técnico

Calculo de Algoritmos:

1) **Ingreso de las apuestas antes del inicio de la carrera**

```
public void IngresoApuestaOrden(String texto, String textoNombre, String textoMonto, JTextArea textArea){

  try {
        String[] datos = texto.split(",");

  String textoGeneral = "";
        String nombre = textoNombre;
                    // ingreso de nuevas apuestas en base al texto

        double monto = Double.parseDouble(textoMonto);
           textoGeneral = textoGeneral + nombre + ",";
         textoGeneral = textoGeneral + monto + ",";
         textoGeneral = textoGeneral   + datos[0] + ",";
         textoGeneral = textoGeneral   + datos[1] + ",";
         textoGeneral = textoGeneral   + datos[2] + ",";
         textoGeneral = textoGeneral   + datos[3] + ",";
         textoGeneral = textoGeneral   + datos[4] + ",";
         textoGeneral = textoGeneral   + datos[5] + ",";
         textoGeneral = textoGeneral   + datos[6] + ",";
         textoGeneral = textoGeneral   + datos[7] + ",";
         textoGeneral = textoGeneral   + datos[8] + ",";
         textoGeneral = textoGeneral   + datos[9] + ",";


         textArea.append(textoGeneral);
         textArea.append("\n");
```

```
String nombre = textoNombre;
                    // ingreso de nuevas apuestas en base al texto

        double monto = Double.parseDouble(textoMonto);
           textoGeneral = textoGeneral + nombre + ",";
         textoGeneral = textoGeneral + monto + ",";
         textoGeneral = textoGeneral   + datos[0] + ",";
         textoGeneral = textoGeneral   + datos[1] + ",";
         textoGeneral = textoGeneral   + datos[2] + ",";
         textoGeneral = textoGeneral   + datos[3] + ",";
         textoGeneral = textoGeneral   + datos[4] + ",";
         textoGeneral = textoGeneral   + datos[5] + ",";
         textoGeneral = textoGeneral   + datos[6] + ",";
         textoGeneral = textoGeneral   + datos[7] + ",";
         textoGeneral = textoGeneral   + datos[8] + ",";
         textoGeneral = textoGeneral   + datos[9] + ",";
```

Al final el ingreso por partes del area de texto, genera un 0(1).

```java
// metodo de ingreso de las apuestas EL CUAL DEBE DE CUMPLIR CON EL O(1)
public void addBets(String line, int cantidadLineas, Apuesta[] apuetas) {

    String[] datosStringOrden = line.split(",");
    int[] orden = new int[10];
    Apuesta nuevaApuesta = new Apuesta();
    try {

        String nombre = datosStringOrden[0];
        double monto = Double.parseDouble(datosStringOrden[1]);
        // ahora el ingreso del orden
        for (int i = 0; i < 10; i++) {
            orden[i] = Integer.parseInt(datosStringOrden[i + 2]);
        }
        nuevaApuesta.setNombre(nombre);
        nuevaApuesta.setMonto(monto);
        nuevaApuesta.setApuestaCaballosOrden(orden);
        nuevaApuesta.setValida(true);
    } catch (Exception e) {
        e.printStackTrace();
        nuevaApuesta.setValida(false);

    }
    apuetas[cantidadLineas] = nuevaApuesta;

}
```

En este caso lo único que varia en cuanto a complejidad con el algoritmo anterior es que, este tiene un ciclo, pero solo realiza 10 ciclos, así que no se convierte en una función o(n), sino en una función o(10n) → lo cual al final seria constante.

```
    if (jTextArea1.equals("")) {
        JOptionPane.showMessageDialog(null, "NO EXISTE ARCHIVO INGRESE NUEVAMENTE");
    } else {
        try {
            lectureArchive ingresoArchivo = new lectureArchive();
            int cantidaddeApuestas = this.cantidadApuestas(jTextArea1);
            int numero = 0;
            boolean esVacio = false;
            apuestas = new Apuesta[cantidaddeApuestas];
            String[] cadenass = ingresoArchivo.obtencionLineas(jTextArea1.getText());

            for (int i = 0; i < cadenass.length; i++) {
                if ((cadenass[i].isBlank()) || cadenass[i].isEmpty()) {
                    JOptionPane.showMessageDialog(null, "ERROR");
                    esVacio = true;
                    break;
                } else {
                    addBets(cadenass[i], numero, apuestas);
                    numero++;
                }
            }

            if (esVacio != true) {
                betsVerify verificacion = new betsVerify();
                verificacion.verificar(apuestas);

            }
            int prueba = 0;


            for (Apuesta ver : apuestas) {
                System.out.println(prueba + " " + ver.toString() + " ");
                prueba++;
```

2) **Verificación de apuesta**

```java
    private int CantidadErrores;
// verifica la cantidad de repetidos segun las apuestas y si hay tira error
    public void verificar( Apuesta[] apuestas) {

        for (int i = 0; i < apuestas.length; i++) {

            int[] numeros = apuestas[i].getApuestaCaballosOrden();
            if (repetidos(numeros) == true) {
                apuestas[i].setValida(false);
            } else {
                apuestas[i].setValida(true);
            }
        }

        setErrores(apuestas);

    }


// determina si hay repetidos

    public boolean repetidos(int[] valores) {
        for (int i = 0; i < valores.length; i++) {
            for (int j = i + 1; j < valores.length; j++) {
                if (valores[i] == valores[j]) {
                    return true;
                }
            }
        }
        return false;
    }
```

Para la verificación de errores y apuestas, genere dos algoritmos, los cuales se ejecutan por separado, pero ambos son ciclos, los cuales están a parámetros de otros variables es decir que son funciones O(n), ademas que al usar estos dos, se genera un O(Cn) => O(2n).

3) **Cálculo de los resultados al finalizar la carrera**

```
public void calculo(){
   for (int i = 0; i < 10; i++) {
      for (int j = 0; j < apuestasOrden.length; j++) {
         if (apuestasOrden[j].getValida()== true) {
            int[] ordenCalculo = apuestasOrden[j].getApuestaCaballosOrden();
            if (orden[i] == ordenCalculo[i]) {
               switch(i){
                  case 0:apuestasOrden[j].incrementoPunteo(10);
                     break;
                    case 1:apuestasOrden[j].incrementoPunteo(9);
                     break;
                     case 2:apuestasOrden[j].incrementoPunteo(8);
                     break;
                     case 3:apuestasOrden[j].incrementoPunteo(7);
                     break;
                     case 4:apuestasOrden[j].incrementoPunteo(6);
                     break;
                     case 5:apuestasOrden[j].incrementoPunteo(5);
                     break;
                     case 6:apuestasOrden[j].incrementoPunteo(4);
                     break;
                     case 7:apuestasOrden[j].incrementoPunteo(3);
                     break;
                     case 8:apuestasOrden[j].incrementoPunteo(2);
                     break;
                     case 9:apuestasOrden[j].incrementoPunteo(1);
                     break;
               }

            }
         }
      }
```

Para el calculo utilice dos ciclos, pero que al final uno sera constante y otro sera variable, entonces, se formaría → C * n => O(Cn).

4 **Ordenamiento de los resultados**

```
public void ordenNombre(Apuesta[] apuestasOrden, JTextArea textArea){
  textArea.setText(null);
  //ordenamiento burbuja
  for (int i = 0; i < apuestasOrden.length; i++) {
     for (int j = 0; j < apuestasOrden.length - 1; j++) {
        //comparacion entre valor actual y siguiente
        if (apuestasOrden[j].getNombre().compareTo(apuestasOrden[j + 1].getNombre()) > 0) {
           Apuesta apuestaAuxiliar;

           //ordenamiento burbuja
           apuestaAuxiliar = apuestasOrden[j];
           apuestasOrden[j] = apuestasOrden[j + 1];
           apuestasOrden[j + 1] = apuestaAuxiliar;
        }
     }
  }

        //impresion
  for (int i = 0; i < apuestasOrden.length; i++) {
     if (apuestasOrden[i].getValida()== true) {
        String texto = "";
        texto = apuestasOrden[i].getNombre() + " Punteo: " + apuestasOrden[i].getPunteo();
        textArea.append(texto+"\n");
     }
  }
```

Para el ordenamiento, realice el método burbuja, en el cual se utilizan dos ciclos variables, los cuales serán
→ n*n <=> n^2 <=> **O(n^2).**