Calculo del O(n) de metodos usados en la practica

201930699 Erick Daniel Morales Xicará

1. Ingreso De Apuestas:

El ingreso fue utilizando una lista doblemente enlazada circular la cual permitio generar un O(1) para el ingreso de las apuestas ya que el algoritmo es constante y no depende de los datos que se ingresen

```
public void añadir(Apuesta valor) {                                    //O(1)
    if (empty() == false) {                                            //O(1)
        Nodo nuevo = new Nodo( siguiente: ultimoNodo,  anterior: inicioNode, valor);  //O(1)
        ultimoNodo.setSiguiente( siguiente: nuevo);                    //O(1)
        inicioNodo.setAnterior( anterior: nuevo);                      //O(1)
        ultimoNodo = nuevo;                                            //O(1)
    }
    if (empty()) {                                                     //O(1)
        Nodo nuevo = new Nodo( siguiente: ultimoNode,  anterior: inicioNode, valor);  //O(1)
        inicioNodo = nuevo;                                            //O(1)
        ultimoNodo = nuevo;                                            //O(1)
    }
    tamaño++;
    //                                                                 //O(1)
}                                                                      //RESULTADO: O(1)
```

2. Verificacion De Apuestas:

La validacion de apuestas esta constituido por tres metodos y se uso recursion para poder validar los datos. El primer metodo es un O(1) ya que sera contante y a su vez el metodo dos tiene un O(n) y el tercero no da tambien un O(n) por lo cual al sumar nos da como resultados un O(n).

3. Calculo de Resultados

El calculo de puntos se basa en dos metodos, tambien usando recursividad uno nos da un O(1) por ser un ciclo constante y el segundo nos da un O(n) ya que se tuvo que usar un while, como resultado de la multiplicacion nos da un O(n)

```
public void CalcularPuntos(int caballos[]){
    this.resultados= caballos;                                         //O(1)
    if(empty()==false){                                                //O(1)
        Nodo aux = inicioNodo;                                         //O(1)
        Nodo auxi= inicioNodo.getAnterior();                           //O(1)
            sumatoriaPuntos( apuesta: aux.getValor());                 //O(1)
            aux=aux.getSiguiente();                                    //O(1)
        while(auxi!=auxi){                                             //O(n)  ---->
            sumatoriaPuntos( apuesta: aux.getValor());                 //O(1)
            aux=aux.getSiguiente();                                    //O(1)
        }
      if(inicioNodo==ultimoNodo){

        } else{
            sumatoriaPuntos( apuesta: aux.getValor());
        }
    }                                                                  //-------------------------
   // this.OrdenarResultados();                                        //RESULTADO:   n
}
```

```
public void sumatoriaPuntos(Apuesta apuesta){
    for (int i = 0; i < 10; i++) {                                     //O(10)
        if(apuesta.getCaballos()[i]==this.resultados[i]){              //O(1)
            apuesta.setPuntos(apuesta.getPuntos()+(10-i));             //O(1)
        }                                                              //-----------------
    }                                                                  //RESULTADO 1
}
```

4. Ordenamiento por Puntos

EL ordenamiento se utilizo el metodo de insecion consiste en recorrer todo el array comenzando desde el segundo elemento hasta el final. Para cada elemento, se trata de colocarlo en el lugar correcto entre todos los elementos anteriores a él osea entre los elementos a su izquierda en el array. y en el pero de los casos es O(n^2)

```
public void OrdenarResultados(){
    inicioNodo.setAnterior( anterior: null);                           //O(1)
    ultimoNodo.setSiguiente( siguiente: null);                         //O(1)
    Nodo aux=inicioNodo.getSiguiente();                                //O(1)
    Nodo auxi=inicioNodo.getAnterior();                                //O(1)
    while(aux!=null){                                                  //O(n)--->
        Nodo auxiE = aux.getAnterior();                                //O(1)
        while(auxiE!=null){                                            //O(n)--->
            if(auxiE.getValor().getPuntos() > auxiE.getSiguiente().getValor().getPuntos()){  //O(1)
                if (auxiE == inicioNodo) {                             //O(1)
                    Nodo pivot = auxiE.getSiguiente();                 //O(1)
                    Nodo pivotN= auxiE;                                //O(1)
                    auxiE.setSiguiente( siguiente: pivot.getSiguiente());  //O(1)
                    auxiE.setAnterior( anterior: pivot);               //O(1)
                    pivot.getSiguiente().setAnterior( anterior: auxiE);  //O(1)
                    pivot.setAnterior( anterior: null);                //O(1)
                    pivot.setSiguiente( siguiente: auxiE);             //O(1)
                    inicioNodo=auxiE;                                  //O(1)
                }else if(auxiE==ultimoNodo){                           //O(1)
                    Nodo pivot = auxiE.getSiguiente();                 //O(1)
                    Nodo pivotN= auxiE;                                //O(1)
                    auxiE.getAnterior().setSiguiente( siguiente: pivot);  //O(1)
                    auxiE.setSiguiente( siguiente: null);              //O(1)
                    auxiE.setAnterior( anterior: pivot);               //O(1)
                    pivot.setAnterior( anterior: pivotR.getAnterior());  //O(1)
                    pivot.setSiguiente( siguiente: auxiE);             //O(1)
                    ultimoNodo=auxiE;                                  //O(1)
                }else{                                                 //O(1)
                    Nodo pivot = auxiE.getSiguiente();                 //O(1)
                    Nodo pivotN= auxiE;                                //O(1)
                    auxiE.getAnterior().setSiguiente( siguiente: pivot);  //O(1)
                    auxiE.setSiguiente( siguiente: pivot.getSiguiente());  //O(1)
                    auxiE.setAnterior( anterior: pivot);               //O(1)
                    pivot.getSiguiente().setAnterior( anterior: auxiE);  //O(1)
                    pivot.setAnterior( anterior: pivotR.getAnterior());  //O(1)
                    pivot.setSiguiente( siguiente: auxiE);             //O(1)
                    auxiE=pivot.getAnterior();                         //O(1)
                }
            }else{
                break;                                                 //O(1)
            }
        }
        aux = aux.getSiguiente();                                      //O(1)
    }
    inicioNodo.setAnterior( anterior: ultimoNodo);                     //O(1)
    ultimoNodo.setSiguiente( siguiente: inicioNodo);                   //O(1)
                                                                       //-------------------------------
                                                                       //RESULTADO O(n^)
```

6.Exportar Archivo csv

Esta constituido de dos metodos, el primero genera un O(1) por ser constante y el segundo genera un O(n) por lo cual si sumamos y quitamos las constantes queda un O(n)

```
public void generarArchivo(ListaCircularDoble lista) throws IOException {
    FileDialog guardar = null;                                         //O(1)
    guardar = new FileDialog( parent: guardar,  title: "Guardar como",  mode: FileDialog.SAVE);  //O(1)
    guardar.setVisible( b: true);                                      //O(1)
    guardar.dispose();                                                 //O(1)
    if (guardar.getFile() != null && guardar.getDirectory() != null) {  //O(1)
        FileWriter writer = new FileWriter(guardar.getDirectory() + guardar.getFile() + ".csv");  //O(1)
        String resultado = "Nombre Apostador,Monto Apostado,Puntos Conseguidos,Primer Lugar"+   //O(1)
                ",Tercer Lugar,Cuarto Lugar, Quinto Lugar,Sexto Lugar,Septimo Lugar,Octavo Lugar"+
                ",Noveno Lugar,Decimo Lugar\n";
        resultado += escribirArchivo( nodo: lista.getInicioNodo());    //O(n)--->
        writer.write( str: resultado);                                 //O(1)
        writer.close();                                                //O(1)
        JOptionPane.showMessageDialog( parentComponent: null,  message: "Archive guardado con éxito");  //O(1)
    }                                                                  //-----------------------
}                                                                      //RESULTADO: O(n)
```

```
private String escribirArchivo(Nodo nodo) {                           //O(n)
    Nodo tmp = nodo;                                                   //O(1)
    String resultado= tmp.getValor().getPersona().getNombre() + "," + tmp.getValor().getMontoApostado()+","  //O(1)
            + tmp.getValor().getPuntos() + "," + tmp.getValor().getCaballos()[0] + "," + tmp.getValor().getCaballos()[1]  //O(1)
            + "," + tmp.getValor().getCaballos()[2] + "," + tmp.getValor().getCaballos()[3] + ","          //O(1)
            + tmp.getValor().getCaballos()[4] + "," + tmp.getValor().getCaballos()[5] + ","               //O(1)
            + tmp.getValor().getCaballos()[6] + "," + tmp.getValor().getCaballos()[7] + ","               //O(1)
            + tmp.getValor().getCaballos()[8] + "," + tmp.getValor().getCaballos()[9] + "\n";             //O(1)
    tmp = tmp.getSiguiente();                                          //O(1)

    while (tmp!=nodo.getAnterior()) {                                  //O(n)
        resultado += tmp.getValor().getPersona().getNombre() + "," + tmp.getValor().getMontoApostado()+","  //O(1)
                + tmp.getValor().getPuntos() + "," + tmp.getValor().getCaballos()[0] + "," + tmp.getValor().getCaballos()[1]  //O(1)
                + "," + tmp.getValor().getCaballos()[2] + "," + tmp.getValor().getCaballos()[3] + ","     //O(1)
                + tmp.getValor().getCaballos()[4] + "," + tmp.getValor().getCaballos()[5] + ","           //O(1)
                + tmp.getValor().getCaballos()[6] + "," + tmp.getValor().getCaballos()[7] + ","           //O(1)
                + tmp.getValor().getCaballos()[8] + "," + tmp.getValor().getCaballos()[9] + "\n";         //O(1)
        tmp = tmp.getSiguiente();                                      //O(1)
    }
    resultado += tmp.getValor().getPersona().getNombre() + "," + tmp.getValor().getMontoApostado()+","  //O(1)
            + tmp.getValor().getPuntos() + "," + tmp.getValor().getCaballos()[0] + "," + tmp.getValor().getCaballos()[1]  //O(1)
            + "," + tmp.getValor().getCaballos()[2] + "," + tmp.getValor().getCaballos()[3] + ","         //O(1)
            + tmp.getValor().getCaballos()[4] + "," + tmp.getValor().getCaballos()[5] + ","               //O(1)
            + tmp.getValor().getCaballos()[6] + "," + tmp.getValor().getCaballos()[7] + ","               //O(1)
            + tmp.getValor().getCaballos()[8] + "," + tmp.getValor().getCaballos()[9] + "\n";             //O(1)
    return resultado;                                                  //O(1)
}                                                                      //-----------------
                                                                       //RESULTADO O(n)
```

7. Entrada De Datos

Esta constituido de dos metodos, los cuales uno tiene un O(1) y el otro un O(n) por lo cual al sumar tenemos un O(n)

```
public String[] SubirArchivo(MainFrame in) {
    try{
        JFileChooser archivosubir = new JFileChooser();                //O(1)
    int seleccion = archivosubir.showOpenDialog( parent: in);          //O(1)
    if (seleccion == JFileChooser.APPROVE_OPTION) {                    //O(1)
        File fichero = archivosubir.getSelectedFile();                 //O(1)
        String path=fichero.getAbsolutePath();                         //O(1)
        try ( FileReader fr = new FileReader( file: fichero)) {        //O(1)
            String cadena = "";                                        //O(1)
            int valor = fr.read();                                     //O(1)
            while (valor != -1) {                                      //O(n)------>
                cadena = cadena + (char)valor;                         //O(1)
                valor = fr.read();                                     //O(1)
            }
            return splitDatos( cadena: cadena.replace( target: "\n", replacement: ","));  //O(1)
        } catch (IOException e1) {
            JOptionPane.showMessageDialog( parentComponent: null,  message: "NO SE PUDO CARGAR EL ARCHIVO");  //O(1)
        }
        return null;                                                   //O(1)
    }
    }catch(Exception e){
        JOptionPane.showMessageDialog( parentComponent: null,  message: "NO SE PUDO CARGAR EL ARCHIVO");  //O(1)
    }
    return null;                                                       //O(1)
}

public String[] splitDatos(String cadena){
    String [] resultado = cadena.split( regex: ",");                   //O(1)
    return resultado;                                                  //O(1)
}                                                                      //-------------------------------
                                                                       //RESULTADO O(n)
```

Metodos Extras

5. Tamaño Arreglo:

Este metodo tiene un O(n) ya que al final dependera de los n datos

```
public int tamañoArreglo() {
    boolean leer = true;                                               //O(1)
    int contador = 1;                                                  //O(1)
    if (empty() == false) { // VERIFICO QUE NO ESTE VACIA MI LISTA     //O(1)
        Nodo aux = inicioNodo.getSiguiente();                          //O(1)
        Nodo auxi = inicioNodo.getAnterior();                          //O(1)
        if (inicioNodo.getAnterior() == ultimoNodo.getSiguiente()) {   //O(1)
            leer=false;
        } else {                                                       //O(1)
            while (leer) {                                             //O(n)--------->
                if (aux != auxi) {                                     //O(1)
                  //   System.out.println("NODO " + aux.getValor());     //O(1)
                    aux = aux.getSiguiente();                          //O(1)
                    contador++;                                        //O(1)
                } else {                                               //O(1)
                  //  System.out.println("NODO " + aux.getValor());     //O(1)
                    leer = false;                                      //O(1)
                    contador++;                                        //O(1)
                }                                                      //----------------
                                                                       //Resultado final: O(n)
            }
        }

    }
```