Estructuras de datos y diseño de algoritmos
Laboratorio 1:

Profesora: Daniela Moreno
Autores: Lucas Abello

1. Introducción

Este informe tiene como objetivo analizar, corregir y mejorar el código "platform.java", este codigo permite leer un archivo csv donde estan contenidos distintos videos de youtube con sus parametros, este se va leyendo linea por linea y almacenando los videos a la plataforma (dentro de este codigo esta la clase video y platform es una lista enlazada).

2. Depuración y mejora del código

Este código implementa una plataforma de videos que puede leer datos de un archivo CSV y manejar una lista enlazada de videos. Las principales funciones incluyen: la inserción de videos desde un archivo CSV en la lista enlazada y la impresión iterativa de todos los videos de la lista. Se han detectado algunos errores en el código que se deben abordar, tales como:

- En la funcion de la linea 68 del código entregado (funcion arrayToVideo) en los 3 ultimos Array.get() no se permite ingresar, esto debido a que las 3 ultimas variables que han sido ingresadas son de tipo string, y en el constructor de la clase video estas son de tipo int.

  Agregado a esto al solucionar este error surge nuevamente otro debido a que un string tiene un numero muy grande este no puede ser te tipo int, obligando a cambiar los int del constructor por unos long y a los Array.get() que daban problemas hacerles un Long.parseLong(Array.get()).

  Luego de solucionar esos dos errores surge el 3er y ultimo error en ese bloque y es que algunos strings estan vacios () negandonos la transformacion a Long, esto lo arreglamos creando 3 variables temporales y se hicieron unas condiciones con un if:

  - si Array.get(nro x) == entonces tempx = 0, si no, temp = Long.parseLong(Array.get())

  Con todos esos procesos y arreglos el bloque de arrayToVideo queda corregido.

- La funcion insertAtEnd no funciona por ende fue reemplazado completamente, esto debido a que igualmente fue encontrado un error en el iterativeLast. Este error fue arreglado de la siguiente manera:

  - Al inicio de la clase platform fue creada la variable de tipo Video llamada auxiliar y esta fue igualada a la variable head.
  - en la funcion insertAtEnd esta el siguiente if:
    - si auxiliar == null, entonces head = Video ingresado y auxiliar igual, si no, auxiliar.next = Video ingresado y auxiliar = Video ingresado.

  Como auxiliar hace referencia a head todos los videos que se van ingresando se van "linkeando.[en] la lista quedando al final de esta, de esta manera no nececito iterar la lista entera para ir agregando al final reduciendo bastante la complejidad algoritmica. (de O(n) a O(1))

- Fue corregida completamente la funcion iterativePrint reemplazandolo simplemente por un while(mientras Nodo ingresado != null) se va imprimiendo el ID de ese nodo y el titulo asignado a ese nodo reduciendo nuevamente la complejidad algoritmica de la funcion del codigo original (bajando de O(n a la 2) a O(N).

- fueron eliminadas del codigo las funciones:
  - recursivePrint, debido a poca optimización
  - iterativeLast, debido a que no se termino usando
  - recursiveLast, debido a que no se termino usando
  - isNumericInt, debido a que se sustituyeron los int por Long y se parseó
  - isNUmericLong, debido a que se uso parseLong y se hicieron condiciones para

2.1. Calcular e Incluir la popularidad de cada video

Para lograr esto fue agregado el parametro long popularidad a la clase Video, dentro del constructor de esta fue creado un if que ve si los likes son 0, si estos son 0 la popularidad es 0, si no, la popularidad son las visitas / likes, quedandonos:

```
1
2       (todos los otros parametros)
3       long popularidad;
4
5       Video(todos los otros parametros){
6           this.videoID = videoID;
7           this.videoTitle = videoTitle;
8           this.channelID = channelID;
9           this.channelTitle = channelTitle;
10          this.publishedAt = publishedAt;
11          this.viewCount = viewCount;
12          this.likeCount = likeCount;
13          this.commentCount = commentCount;
14
15          //agregado el calculo y el parametro de popularidad
16          if(likeCount == 0){
17              this.popularidad = 0;
18          }else{
19              this.popularidad = viewCount/likeCount;
20          }
21      }
22
```

2.2. Mejoras del código

Como fue mencionado en la parte de depuración hay muchas funciones que fueron eliminadas, y las funciones las cuales se redujo la complejidad algoritmica son:

- insertAtEnd, la cual fue mejorada creando la variable auxiliar de tipo Video asociada a la cabecera de la lista enlazada, al llamar la funcion pide como parametro un video, esta verifica si auxiliar esta vacia (o sea si head es null) y de ser asi le asigna el Video dado como parametro a head, si no esta vacia auxiliar entonces auxiliar.next pasa a ser el Video ingresado y auxiliar pasa a ser el Video ingresado estando auxiliar siempre en el ultimo nodo de la lista platform.

  Con este metodo se redujo la complejidad algoritmica de O(N) a O(1) de esta funcion, esto debido a que la funcion original iteraba toda la lista hasta llegar a la cola de esta y con el arreglo es una sola iteracion la que debe hacer.

- iterativePrint, esta funcion originalmente tenia 2 while anidados (que la verdad eran más la funcion search que piden crear que un iterative print). Esto fue reemplazado por un solo while que va recorriendo la lista platform y ejecutando la funcion .play() que va imprimiendo en consola los ID de el video actual y el tirulo de este. Este cambio permitió una reduccion de complejidad bastante grande pasando de O(n a la 2) a O(N).

2.3. Incluir el método para buscar un video por el campo videoID

Esta funcion fue incluida de la siguiente manera:

```
    Video search(Video v, String videoID){
            Video aux = v;
            while (aux != null){
                if (aux.videoID.equals(videoID)){
                    System.out.print("search: ");
                    aux.play();
                    return aux;
                }
                aux = aux.next;
           }
            return null;
     }

```

Esta va recorriendo iterativamente la lista desde el Video entregado como parametro comparando el string entregado con el videoID del Nodo actual, si estos coinciden entonces imprimira en pantalla el titulo de este junto a su Id, en caso contrario retorna null.

2.4. Incluir el m étodo para invertir la lista de orden

Esta funcion fue incluida de la siguiente manera:

```
    void reverse(Video v){
        if (head == null ) {
            return;
        }

        Video prev = null;
        Video aux = head;
        Video next= null;

        while (aux != null) {
            next = aux.next;
            aux.next = prev;
            prev = aux;
            aux = next;
        }
        head = prev;
    }

```

2.5. Probar el funcionamiento de la CLI ubicado en el archivo Client.java

Este codigo no tiene ningun error alguno, este implementa la clase platform del codigo platform.java permitiendonos mediante un menu invertir, imprimir o buscar un video en la consola.

3. Item 2

3.1. Comentarios sobre depuracion

En este proceso no hubo muchos problemas, los errores eran bastante faciles de corregir y al eliminar algunas funciones el código quedó mas legible y menos confuso de entender.

3.2. Hubo una mejoria en la complejidad temporal?

No, no hubo una mejora en esta debido a la funcion insertFromFile, esta es O(N a la 2) por los dos bucles anidados y no fue encontrada una forma optimizada de reducir la complejidad de esta.

3.3. diferencias entre funciones recursivas e iteradas de la API

En este caso particular, las funciones recursivas proporcionadas no se utilizaron debido a que no siempre son más rápidas que las iterativas. De hecho, en algunas situaciones, las funciones recursivas pueden ser más lentas debido al exceso de carga en la pila de llamadas y la necesidad de crear y destruir múltiples marcos de pila. En cambio, una función iterativa bien escrita puede ser más eficiente en términos de memoria y tiempo de ejecución, como se demostró en este código (ya que en algunas funciones se redujo la complejidad).

4. CODIGO CORREGIDO

```java
//Java code
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.Objects;

public class Platform {
    Video head;
    Video auxiliar = head;

    Platform(){
        head = null;
    }

    static class Video {
        String videoID;
        String videoTitle;
        String channelID;
        String channelTitle;
        String publishedAt;
        long viewCount;
        long likeCount;
        long popularidad;
        long commentCount;


        Video next;
        Video(String videoID, String videoTitle, String channelID, String channelTitle, String publishedAt, long viewCount, long likeCount, long commentCount){
            this.videoID = videoID;
            this.videoTitle = videoTitle;
            this.channelID = channelID;
            this.channelTitle = channelTitle;
            this.publishedAt = publishedAt;
            this.viewCount = viewCount;
            this.likeCount = likeCount;
            this.commentCount = commentCount;

            //agregado el calculo y el parametro de popularidad
            if(likeCount == 0){
                this.popularidad = 0;
            }else{
                this.popularidad = viewCount/likeCount;
            }

        }

        void play(){
            System.out.println(videoID +" "+videoTitle);
        }
    }

    Video begin(){
        return head;
    }
```

```
55
56      Video arrayToVideo(ArrayList<String> array){
57
58          long temp, temp2, temp3;
59
60          if(array.get(5) == ""){
61              temp = 0;
62          }else{
63              temp = Long.parseLong(array.get(5));
64          }
65
66          if(array.get(6) == ""){
67              temp2 = 0;
68          }else{
69              temp2 = Long.parseLong(array.get(6));
70          }
71
72          if(array.get(7) == ""){
73              temp3 = 0;
74          }else{
75              temp3 = Long.parseLong(array.get(7));
76          }
77
78          Video v = new Video(array.get(0), array.get(1), array.get(2), array.get(3), array.get(4), temp,
     temp2, temp3);
79
80          return v;
81      }
82
83      void insertFromFile(String file){
84          String string;
85          //boolean first = false;
86          try (BufferedReader br = new BufferedReader(new FileReader(file));){
87              //Skip first
88              br.readLine();
89              while((string = br.readLine()) != null){
90                  boolean inQuotes = false;
91                  int start = 0;
92                  ArrayList<String> newLines = new ArrayList<>();
93                  for (int i = 0; i < string.length(); i++) {
94                      if (string.charAt(i) == '\"') {
95                          inQuotes = !inQuotes;
96                      } else if (string.charAt(i) == ',' && !inQuotes) {
97                          newLines.add(string.substring(start, i));
98                          start = i + 1;
99                      }
100                 }
101                 newLines.add(string.substring(start));
102
103                 Video newVideo = arrayToVideo(newLines);
104
105                 insertAtEnd(newVideo);
106
107             }
108         } catch (Exception e){
109             e.printStackTrace();
110         }
111     }
112
113     void iterativePrint(Video v){
114         Video actual = v;
115         //Dentro del while saque el v.next
116         while (actual != null){
117             actual.play();
118             actual = actual.next;
119         }
120     }
```

```
121
122     void insertAtEnd(Video v){
123         if(auxiliar == null){
124             head = v;
125             auxiliar = v;
126         }else{
127             auxiliar.next = v;
128             auxiliar = v;
129         }
130     }
131
132     Video search(Video v, String videoID){
133         Video aux = v;
134         while (aux != null){
135             if (aux.videoID.equals(videoID)){
136                 System.out.print("search: ");
137                 aux.play();
138                 return aux;
139             }
140             aux = aux.next;
141         }
142         return null;
143     }
144
145     void reverse(Video v){
146         if (head == null ) {
147             return;
148         }
149
150         Video prev = null;
151         Video aux = head;
152         Video next= null;
153
154         while (aux != null) {
155             next = aux.next;
156             aux.next = prev;
157             prev = aux;
158             aux = next;
159         }
160         head = prev;
161     }
162
163     public static void main(String[] args) {
164         //pruebas para la API
165         Platform platform = new Platform();
166         String file = "YoutubeDTSV2.csv";
167         platform.insertFromFile(file);
168         platform.reverse(platform.begin());
169         platform.iterativePrint(platform.begin());
170         platform.search(platform.begin(), "y83x7MgzWOA");
171     }
172 }
```

Listing 1: codigo Platform.java