Tarea 2

Benjamin Ignacio Parra Luman
Departamento de Ingeniería Informática
Universidad de Santiago de Chile, Santiago, Chile
benjamin.parra.l@usach.cl

I. Introducción

Debido a que la sociedad está retornando a la "normalidad" en diversas tiendas existe la necesidad de reinventarse y captar potenciales clientes, por lo que se realizó una investigación la cual concluyó que la música puede llamar la atención de los clientes. La tienda "Dj Simi" al enterarse de este estudio busca cautivar a los clientes con una lista de reproducción, pero que cada canción siga la misma línea de su género, no se encuentre duplicada y que estas vayan cambiando diariamente para atraer a diversos perfiles de posibles compradores. Esta nueva política de la empresa fue un rotundo éxito e incluso se empezaron a compartir la playlist a través de Spotify, pero diversos artistas presentaron reclamos por el uso de sus canciones en tiendas y es por esto que la playlist que se reproduce en el local debió eliminar dichas canciones reclamadas. Se procede a explicar el trabajo el cual tiene como objetivo aplicar los conceptos aprendidos en las clases, tales como cálculo de tiempo de ejecución de manera teórica y práctica, lectura y escritura de archivos, cálculo de complejidad de los algoritmos propuestos para el problema entregado y uso de estructuras. Para la realización de este trabajo se estableció el uso del IDE atom y el lenguaje de programación C con sus herramientas y bibliotecas, para este laboratorio se hará uso de las estructuras para poder almacenar datos y facilitará el acceso a ellos.

II. Solución propuesta

Las entradas del programa serán 3 archivos y un carácter, el primero llamado como "listado.in" contiene un listado de canciones en su primera línea contiene la cantidad de canciones las cuales tienen el formato: título de la canción, id del género, duración (en formato MM:SS) y artista. El segundo archivo "play.in" tiene información correspondiente a los géneros de las canciones que componen la lista del archivo anterior, además de detallar el nombre del género y su número identificador. El tercer archivo "reclamos.in" contiene en su primera línea la cantidad de artistas que han presentado un reclamo por derechos de autor y el listado de nombres del autor. Y el último parámetro de entrada en el archivo es un carácter para realizar un ordenamiento de manera ascendente de lista de reproducción. Si el carácter es una letra A se ordenará alfabéticamente los nombres de los artistas si la letra es C se ordenará alfabéticamente los nombres de canciones y finalmente, si el carácter es D se procederá a ordenar por la duración de las canciones.

Lo primero que se debió pensar y analizar cuál de los tda vistas en clases se utilizará para el almacenamiento de canciones fue, teníamos 3 opciones las cuales eran lista enlazada simple, cola y pila. Se decidió el uso del tda lista enlazada debido a que es el modelo más parecido a una lista de reproducción porque cada una de las canciones va enlazada con otra haciendo un símil a los nodos.
El tda lista enlazada será llamada "Lista" y estará formado por nodos, cada nodo (estructura llamada "Node") estará compuesto una estructura llamada "song" y un puntero al siguiente nodo. Dicha estructura llamada "song" estará compuesta por un arreglo de caracteres llamado "nombreCancion", un entero correspondiente al género (es un identificador que tiene su correspondencia en el archivo "play.in"), otro arreglo de caracteres llamado "duracionString" el cual corresponde a la duración de la canción en el formato "HH:MM:SS", arreglo de caracteres para el nombre del artista denominado "artista" y finalmente un entero para la duración de la canción en segundos. Una vez ya decidida la estructura podemos proceder a explicar el algoritmo para realizar la lectura del primer archivo.

```
leerCanciones(nombre_archivo, cantidad_de_canciones):
tdaLista... Es un tda de canciones (estructura explicada
... anteriormente)
      inicializar(tdaLista)  ...C
      read(nombre_archivo) ....C
      while (archivo no termine) do ... n*2C
            cancion =llenadoCancion()...se llena
                                        ...canción
            insertarNodo(tdaLista, cancion)
  close(nombre_archivo) ...C
  return(canciones)
```

Figura 1: Algoritmo de función leerCanciones con complejidad O(n)

Este algoritmo está descrito a grandes rasgos debido a que la solución final fue un poco más compleja, podrá variar con las cantidades de "C", pero el orden de complejidad sigue siendo el mismo. Una vez ya con los datos de las canciones almacenados en el arreglo de canciones (estructura definida con anterioridad) podemos empezar a hablar de la solución al problema planteado. Ahora debemos realizar la lectura del archivo "play.in" y "reclamos.in", con el primero archivo se logra saber el género de que tendrá la lista y el segundo archivo nos indica los artistas que presentan reclamos con el uso de sus canciones. Además se creará una nueva estructura llamada

"generos" para almacenar el número identificador del género (entero) y el nombre del género (arreglo de caracteres). Estas funciones serán similares a la función de la figura 1 por ende la complejidad será la misma. Debido a que se solicita ordenar las canciones de manera ascendente dada una letra se creó un algoritmo que recibirá la lista y un carácter.

```
ordenarLista(lista, letra):
auxiliar <- lista->cabeza... C
siguiente ...Nodo  C
while auxiliar <> NULL do
   siguiente <- auxiliar->cabeza
    while  siguiente <> NULL do
        switch(letra)
            case 'A':
                if auxiliar->song->nombreCancion >
   siguiente->song->nombreCancion
                    cancion <- auxiliar->canción
                   auxiliar->canción <-siguiente->cancion
                  siguiente->canción <- cancion

   ... esto repetido para las 3 tipos de letra
```

Figura 2: Algoritmo de función ordenarLista con complejidad $O(n^2)$

Luego de ordenarla de manera ascendente se procede a la creación de las playlists, la primera playlist que va a spotify solo debe considerar canciones del mismo género y para la segunda playlist para el local se debe considerar que cada canción debe cumplir ciertos requisitos. El primero es que pertenezca al mismo género y segundo que el artista de la canción no se encuentre en el listado de reclamos. Una vez creada la lista de canciones para el spotify se realiza el filtro para no incluir canciones creadas por artistas que hayan presentado un reclamo y poder tenerla en el local.

```
creaPlaylist(lista, id_genero): tdaLista
    sublista <- creaLista()
    if !esListaVacia(lista)  then
       auxiliar <- lista->head
      while auxiliar <> NULL
         if auxiliar->song->idgenero == id_genero then
            cancion <- auxiliar->canción
            insertarNodo(sublista,cancion)
            auxiliar <- auxiliar->siguiente
         else
              auxiliar <- auxiliar->siguiente
    return sublista
```

Figura 3: Algoritmo de función creaPlaylist con complejidad O(n)

Luego de tener generadas las listas se procederá a crear los archivos de salida, el primero llamado "playlist.out" en su primera línea debe incluir la cantidad de canciones presentes en la lista, la duración en el formato HH:MM:SS y su género y el segundo llamado "playlistTienda.out" tiene el mismo formato, pero no incluye las canciones con reclamos.

III. Resultados y análisis

De acuerdo con los objetivos, herramientas y algoritmos anteriores, los resultados del programa cumplen con las expectativas, es decir, el funcionamiento correcto. Cabe destacar que el programa entrega los 2 archivos con el listado de canciones dentro de las condiciones establecidas.
En base a la complejidad del programa se calculó que es de orden cuadrático debido a que funcion con la complejidad más alta es la que ordena la lista de canciones y en base a al tiempo de ejecución se puede ver en la siguiente tabla que los tiempos no varían mucho cuando la cantidad de canciones ingresadas no es mayor, pero cuando se empieza ingresar cantidad del orden de los miles podemos apreciar una variación.

n canciones	tiempo en ms
52	0.001
208	0.001
624	0.004
1248	0.004
3744	0.015
4982	0.022

Figura 4: Tabla con el tiempo de los ciclos de reloj

Con respecto a la complejidad se puede apreciar que entre más canciones tenga el archivo la duración será mayor y esto lo podemos ver graficados en la siguiente figura

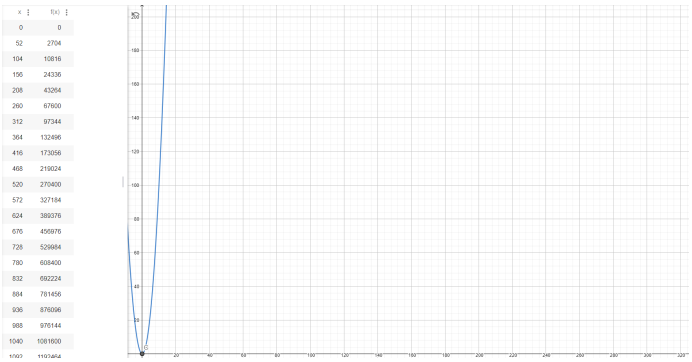

Figura 5: Gráfico de la complejidad del programa

IV. Conclusiones

A modo de conclusión, con respecto a los objetivos establecidos para este trabajo se puede señalar que, el programa entrega los 2 archivos esperados cumpliendo los requerimientos de que las canciones sean del mismo género, que la canción no se repita y que la duración de la playlist esté de acuerdo con los reclamos. Se lograron leer exitosamente los archivos de entradas, obtener sus contenidos

y almacenarlos en las estructuras definidas. Se logra realizar el filtro, ordenamiento y escritura del archivo de salida. Relacionado con la importancia de este laboratorio, se puede destacar que en el desarrollo del programa los conocimientos enlazados con lectura/escritura de archivos, manejo de arreglos, cálculo de complejidades y el uso de los tda.

V. Anexo Manual de usuario

Este manual de usuario está hecho con la finalidad de explicar las funcionalidades, sus casos de uso, cómo hacer funcionar el programa y posibles errores detectados. Este programa necesita de la entrada de 2 nombres de archivos en el cual el primero contiene el listado de canciones y el segundo las especificaciones de la playlist de salida. Este programa funcionará siempre y cuando se respeten los formatos establecidos en los archivos.

V-A. Windows

Se debe ingresar al icono de inicio de Windows haciendo clic en en la esquina inferior izquierda y buscar “cmd” hasta encontrar el panel de “símbolos del sistema” conocida como consola o presionar las teclas “windows + r” y escribir “cmd”. Una vez con la consola abierta nos dirigiremos a la carpeta en la cual estén el programa con extensión “.c” y los 2 archivos extensión “.in” y en la parte superior de la carpeta donde está la dirección con el click derecho copiaremos la dirección como texto

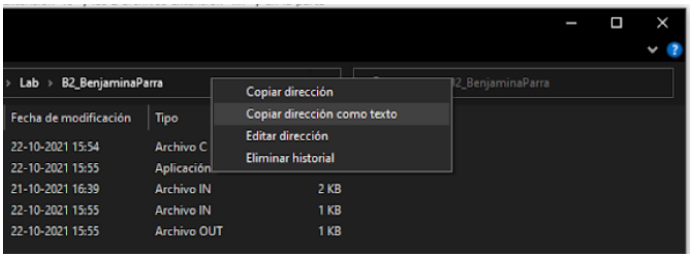

Figura 6: Cómo copiar dirección de la ubicación del programa

Con la dirección ya copiada se debe ingresar la siguiente línea de comando: cd “dirección copiada”

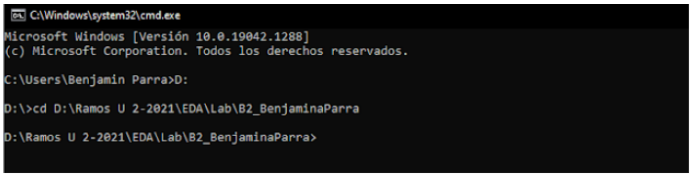

Figura 7: Cómo accede a la carpeta a través de la consola

Ahora para poder compilar el programa con extensión “.c” se debe ingresar la siguiente línea de comando: gcc nombre de archivo.c -o nombre del ejecutable sin extensión

Una vez compilado se debe hacer la ejecución del programa con la siguiente línea de comando: (nombre del ejecutable sin extensión nombre archivo canciones).in (especificaciones playlist).in (lista de reclamos).in (carácter)

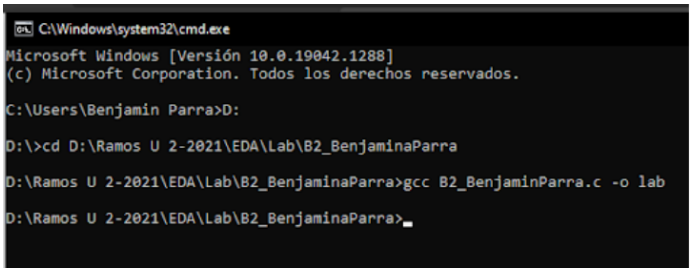

Figura 8: Cómo compilar el código fuente

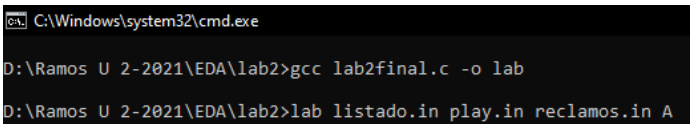

Figura 9: Cómo ejecutar el programa

V-B. Linux

En el sistema operativo Linux la única diferencia radica al momento de realizar la ejecución del programa se debe anteponer “/.” de este modo: ./(nombre del ejecutable sin extensión nombre archivo canciones).in (especificaciones playlist).in (lista de reclamos).in (carácter)

V-C. Posibles errores

Este programa funcionará siempre y cuando se respete el orden de los archivos de entrada, es decir, primero el archivo con el listado de canciones, segundo el archivo con las especificaciones de la playlist, tercero el archivo con los reclamos y un carácter.