Trabajo Práctico Programación 3

Segunda Entrega
Materia: Programacion 3
Carrera T.U.D.A.I Correspondiente a la facultad de Ciencias Exactas(UNICEN)
Autores:
Matias Ezequiel Gonzalez
Mail: monatios@gmail.com
Belén Sofia Enemark
Mail: belenenemark@gmail.com

Introducción:

Partiendo de las búsquedas realizadas sobre los géneros que pueden contener los libros, se desea implementar una herramienta que sea capaz de analizar las mismas.

En esta entrega explicaremos cómo se construye dicha implementación teniendo en cuenta: La conformación de la estructura donde se guardaran las búsquedas, los algoritmos que nos van a permitir realizar los análisis pedidos y las pruebas que se realizaron sobre la estructura y sobre los algoritmos para evaluar la eficiencia y el costo de los mismos.
Por ello, vamos a dividir todo el contenido en dos secciones. La primera abordará la construcción de la estructura y la segunda se enfocará en el análisis de los algoritmos usados como así la eficiencia que tiene frente a otras opciones latentes que pudieran surgir.

Al final de la implementación, la herramienta responderá a los siguientes requerimientos:
a-Obtener los N generos mas buscados de un genero A
b- Obtener todos los generos que fueron buscados despues de buscar el genero A
c- Obtener el grafo únicamente con los géneros afines, es decir, que se vinculan entre sí.

Estructura principal del programa

Elegimos como estructura principal un grafo dirigido etiquetado, compuesto internamente por un arraylist de Vértices. Seleccionamos internamente la estructura arraylist porque:

- La cantidad de géneros no supera los 40, en dataset 1 tiene 35 y en los demás 40.
- Facilita tiempo de codificación(si implementas una List, tenes que crear un iterador por cada vez que lo recorras).
- Facilita las búsquedas de los géneros ya que están indexados.

A su vez en el vértice optamos por crear un arraylist de adyacencias porque:

- Ahorro memoria frente a usar una matriz de adyacencias: en los 4 datasets, comparamos la memoria que ocuparía cada uno(ver cuadro a continuación) , aquí vemos como conviene la lista en caso dataset1 y 2(si o si ocupa menos memoria) y en los casos de dataset 3 y 4 obtenemos que el espacio en memoria es el mismo.
- En segundo lugar, si quieres averiguar cuáles vértices son adyacentes a un vértice dado i. En una matriz tienes que mirar en todas las entradas $|V|$ en el renglón i, incluso si solo un pequeño número de vértices son adyacentes al vértice i. En Cambio en una lista son devolves aquellos vértices que son adyacentes sin recorrer aquellos que no lo son.

Aunque debemos aclarar que esta estructura es más costosa en tiempo de acceso en relación con la matriz de adyacencias. la mantenemos porque nuestra principal necesidad es recorrer sus adyacentes y usar estructuras que en lo posible ocupen menos espacio en memoria.

Cuadro memoria que ocuparía cada estructura: v= vértice, a= arista

Matriz: fórmula (O(v^2))	lista formula O(|v|+|a|)
O(35^2)=1225	O(35+71)
O(40^2)=1600	O(40+1323)
O(40^2)=1600	O(40+1560)
O(40^2)=1600	O(40+1560)

Resolucion de los servicios:

1-Obtener los N generos mas buscados:

Dado un vértice y un entero se desea obtener los N adyacentes de mayor “costo”, para este punto decidimos obtener la lista de adyacencias del vértice dado y recorrerla N veces devolviendo los adyacentes de mayor costo, dado el caso que el N deseado a devolver sea mayor que la cantidad de adyacentes se devuelve la lista de adyacencias entera.
Los casos en los que el N es mayor a la cantidad de adyacentes de un vértice, es un caso especial por que nos arroja el resultado más óptimo a nivel costos siendo este devolver la lista de adyacencias con costo directo, el caso de un N menor el problema más grave es tener que recorrer N veces una cantidad K de nodos adyacentes pero el caso de nuestro problema con nodos limitados el peor caso posible es que se recorran N veces 39 nodos (verticesMax -1)

En el gráfico se puede apreciar la cantidad máxima de adyacentes a recorrer para devolver una solución siempre y cuando el N sea menor a ese número. De estas representaciones concluimos que en los dataSet 3 y 4 crean una especie de grafo dirigido completo dado que todos sus nodos tienen relación entre si

	Dataset1	Dataset2	Dataset3	Dataset4
Arte	2	33	39	39
Cine	2	31	39	39
Viajes	4	33	39	39
Servicios	3	32	39	39

2-Obtener todos los géneros que fueron buscados luego de buscar por el género A.

Para la resolución de este punto decidimos utilizar un DFS adaptado a nuestras necesidades, a partir de un nodo dado exploramos en profundidad, sin preguntar a la salida de ese recorrido si quedan nodos no visitados de este modo solo recorremos los nodos que tienen de algun modo relacion con el nodo solicitado

Dado este análisis nos dimos cuenta de la fuerte relación que tenían los géneros entre sí, considerando la cantidad limitada de nodos con respecto a la cantidad total de aristas que genera el grafo:
-Dataset1 71
-Dataset2 1323
-Dataset3 1560
-Dataset4 1560

Esto datos nos muestran lo fuertemente relacionado que estan los nodos entre si, por eso no es de extrañar que a la hora de hacer la búsqueda en profundidad desde cualquier nodo, el grafo sea recorrido en su totalidad.

	Dataset1	Dataset2	Dataset3	Dataset4
Arte	33	40	40	40
Cine	33	40	40	40
Viajes	33	40	40	40
Servicios	33	40	40	40

3-Obtener el grafo únicamente con los géneros afines, es decir, que se vinculan entre sí (pasando o no por otros géneros).

En este inciso decidimos reciclar el algoritmo de recorrido DFS para mostrar aquellos nodos que se relacionan entre si, los nodos que se relacionan entre sí son aquellos nodos que partiendo desde cualquier nodo de ese subgrafo pueda volver al estado inicial. En definitiva, se quiere obtener el ciclo del grafo. Para ello, fue necesario la creación de una estructura que dijera si el nodo en el que estabamos parados era afin al vertice analizado y guardarlo, por esta causa usamos un HashTable que nos permitiera guardar el nodo y si era afin.

Con el analisis de los generos dados, obtuvimos que en el dataset1 los ciclos obtenidos eran menores a la cantidad total de géneros en ese dataset, en los siguientes dataset vemos que todos los nodos están conectados.

	Dataset1	Dataset2	Dataset3	Dataset4
Arte	22	40	40	40
Cine	25	40	40	40
Viajes	22	40	40	40
Servicios	21	40	40	40

Forma de utilizar el Servicio

La forma de acceder a los servicios es mediante un menú en el cual se le pide al cliente que servicio desea usar.
En el inciso a, le pedimos al usuario que ingrese el generoBuscado y la cantidad de generos mas buscados posteriores a ese generoBuscado. Desde alli generamos una impresión del arrayList obtenido.
En el inciso b, se le pide al usuario el ingreso de un “generoBuscado” para obtener todos los generos buscados despues de ese mismo. Lo mostraemos en otro arraylist.
En el inciso c, tambien se pide el ingreso del Genero a buscar pero en este caso solo mostrara en el arraylist los generos por los cuales se llega nuevamente al Genero buscado.

Conclusiones

Llegando al final de este trabajo, podemos determinar, que apartir del analisis realizado en el inciso 3, que los grafos obtenidos de los dataset 2,3 y 4 son grafos completos(es decir que desde cualquier genero que quieras obtener los ciclos vas a poder llegar a todo el grafo. En cambio en el dataset1 vemos que es un grafo que se conecta pero que no esta completo.

Cuando llegamos al dataset 4 vemos que el grafo tarda mas tiempo en formarse que en los anteriores deducimos que es por la cantidad de líneas que tiene que leer pero eso no incrementa la cantidad de aristas que va a tener al final de la relación.

Bibliografia:

Acerca de “Representacion de grafos” Autor Desconocido
“es.khanacademy.org/computing/computer-science/algorithms/graph-representation/a/representing-graphs”

Autores: Julián Pérez Porto y María Merino. Publicado: 2008. Actualizado: 2012. Definicion.de: Definición de grafos (https://definicion.de/grafos/)