2018

.

TRABAJO PRÁCTICO PROGRAMACIÓN 3 - TUDAI

ETAPA 2

CARTON NELSON - VÁZQUEZ RODRIGO
NELSONCARTON@GMAIL.COM
RODRIGOVAZQUEZ420@GMAIL.COM

FACULTAD DE CIENCIAS EXACTAS
UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES

Introducción

1 PROBLEMA A RESOLVER

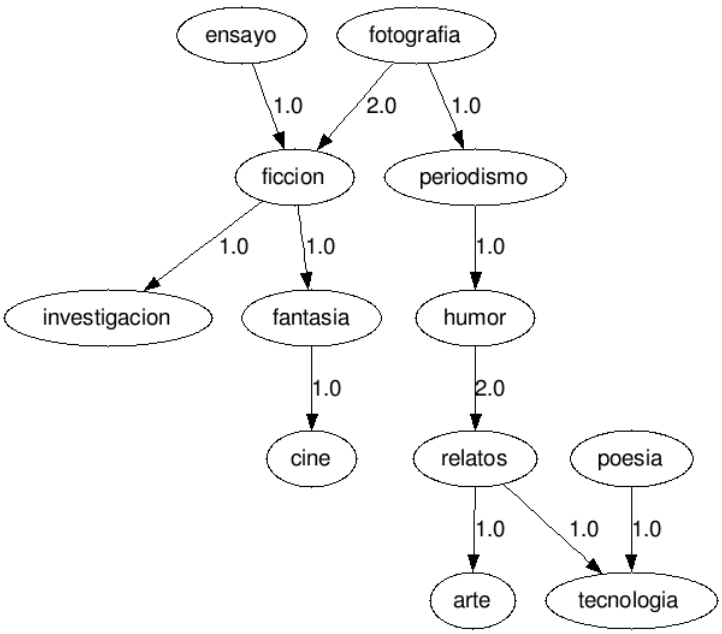
En esta etapa del trabajo se busca implementar los siguientes servicios:

1. Obtener los N géneros más buscados luego de buscar por el género A.
2. Obtener todos los géneros que fueron buscados luego de buscar por el género A.
3. Obtener el grafo únicamente con los géneros afines, es decir, que se vinculan entre sí (pasando o no por otros géneros).

La cátedra proveera 4 archivos csv del estilo:

```
ensayo,ficción,fantasía,cine
fotografía,periodismo,humor,relatos,tecnología
poesía,tecnología
fotografía,ficción
fotografía,ficción,investigación
humor,relatos,arte
```

A partir de los cuales se generará un grafo de busqueda de generos donde el peso de sus arcos indica la cantidad de veces que luego de buscar por un genero se busco directamente por el segundo.



1 Grafo del ejemplo de entrada

Desarrollo

2 DECISIONES DE DISEÑO

Para el grafo pensamos dos formas de resolver las relaciones entre los vértices, implementando una matriz de adyacencia o una lista de adyacentes en cada nodo. Optamos por escoger la segunda opción ya que es más eficiente a la hora de iterar sobre todos los adyacentes de un nodo a, diferencia de la matriz que cuando más grande es el grafo más ineficiente resulta el recorrido.

Luego para esta lista pensamos si sería óptimo utilizarla como una fila de prioridad ordenada por el grado de cada vértice, pero debido a que los servicios no necesitan de esta funcionalidad y además el hecho de que ralentizaría el añadido de nuevos vértices al grafo se optó por no mantener un orden en los nodos, y así tener una inserción directa.

En el primer servicio a implementar debe retornarse una cantidad n de géneros que más se buscaron inmediatamente luego de buscar un género dado, para esto usamos una función la cual chequea entre todos sus adyacentes y devuelve los mayores según dicha cantidad (n).

Para el segundo y tercer caso se decidió utilizar el algoritmo backtracking de búsqueda DFS (Deepth First Search) en lugar de BFS (*Breadth First Search*). Debido a que como en ambos servicios deben recorrerse **todos** los nodos adyacentes y esto puede lograrse con ambos algoritmos, la búsqueda por BFS no resultaría beneficiosa debido a que utilizara mas memoria al tener que almacenar los padres.

El grafo es un grafo dirigido, porque las búsquedas van en una dirección, es decir muestran que se buscó luego de un nodo N y etiquetado por que el peso de cada arista muestra la cantidad de veces que se realizó la misma secuencia de búsqueda.

3 IMPLEMENTACIÓN DE LAS CLASES

Para la clase **Grafo** empezamos con declarar un arrayList con todos los vértices(géneros) que se encuentran en el archivo .csv. Al agregar un nuevo género se comprueba que no exista en el grafo y se agrega, se relaciona con el anteriormente agregado y se le setea el peso en 1 (es decir, luego de una búsqueda X se buscó una vez Y) si el género ya está en el grafo se comprueba si existe la relación de adyacencia con el género anteriormente buscado y si es así se le aumenta el peso.

La clase **Nodo** está compuesta de un String el cual lleva el nombre del género y una arrayList<adyacencte> para la lista de vértices adyacentes a él.

La clase **Adyacente** seria la representante de las aristas, poseen un vértice destino y un peso, el cual representara la cantidad de veces que se busca llegar a dicho destino.

4 IMPLEMENTACIÓN DE LOS SERVICIOS

- **Servicio 1**: Para este servicio primero buscamos el género dentro del grafo con el método `buscarVertice(genero)`. Y luego de este vértice utilizamos: `getAdyacentesMasBuscados(int n)` el cual (siempre y cuando n sea menor que la cantidad de adyacentes que posee) colocará todos sus adyacentes en una lista auxiliar de la cual se le removerá n veces el Adyacente con el mayor peso. Por lo que el tiempo de búsqueda es directamente proporcional a las n veces que se deba recorrer toda lista.

- **Servicio 2**: Como dijimos con anterioridad en el apartado “decisiones de diseño” implementamos una búsqueda del tipo DFS, para poder devolver los adyacentes del vértice en cuestión y los adyacentes de estos últimos de modo recursivo. Cada vértice visitado es marcado en una tabla de hash para evitar volver a navegarlo y es añadido a un ArrayList<Nodo> que se retornara al final.

- **Servicio 3**: De nuevo aquí, implementamos una búsqueda DFS con una tabla de hash donde marcamos los nodos visitados. Dado un vértice inicial recorremos sus hijos y le preguntamos si alguno de ellos es parte del camino de afín. La recursividad ira avanzando hasta volver a encontrarse (o no) con el nodo inicial, cuando esto ocurra significará que desde el inicio volvió a llegar a si mismo por lo que se agregaran los vértices a un nuevo grafo.
  En una primera instancia obteníamos estos vértices de manera correcta en un Set de nodos. Luego como pedía el ejercicio decidimos que se devolviese un nuevo grafo, la pega de esto es que, si bien funciona para el Dataset 1, para los Datasets del 2 al 4 se queda ejecutando el código. No sabemos si se queda en un loop infinito o si tarda demasiado tiempo en compilar. Debido a esto **no se pudo realizar análisis comparativos para este servicio**.

5 RESULTADOS OBTENIDOS EN LAS PRUEBAS

Para poder apreciar las diferencias entre los distintos servicios incluimos en el codigo un timer que nos retorna el tiempo que tarda en ejecutarse cada metodo. Los tiempos obtenidos fueron los siguientes:

DATASET 1		
Tiempo / Genero	Servicio 1	Servicio 2
arte	69531	1433644
ciencia	20517	1443617
juegos	15958	1464134
marketing	76085	1883314
romance	85204	1590658
terror	22512	2532457

DATASET 2		
Tiempo / Genero	Servicio 1	Servicio 2
arte	100592	2599993
ciencia	135642	2225838
juegos	73806	2367464
marketing	153595	2146903
romance	217996	2170556
terror	134502	2652996

DATASET 3		
Tiempo / Genero	Servicio 1	Servicio 2
arte	111136	2277132
ciencia	155304	2114988
juegos	176676	1849974
marketing	181806	1877044
romance	147326	2207316
terror	316878	2450673

DATASET 4		
Tiempo / Genero	Servicio 1	Servicio 2
arte	162713	2022375
ciencia	81785	2042607
juegos	259316	1913235
marketing	151885	1986470
romance	261880	2649292
terror	137637	2187653

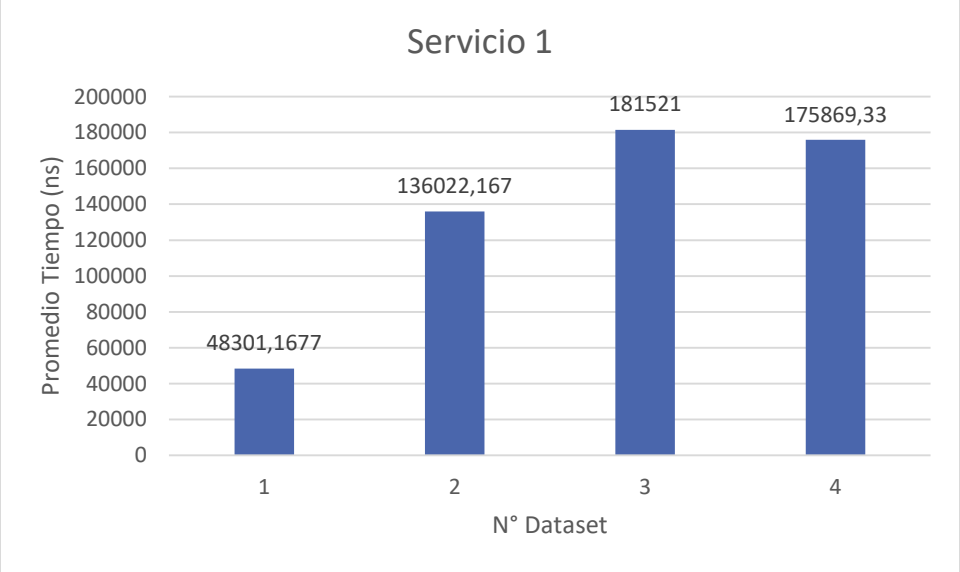
Entonces en promedio seria para el Servicio 1:

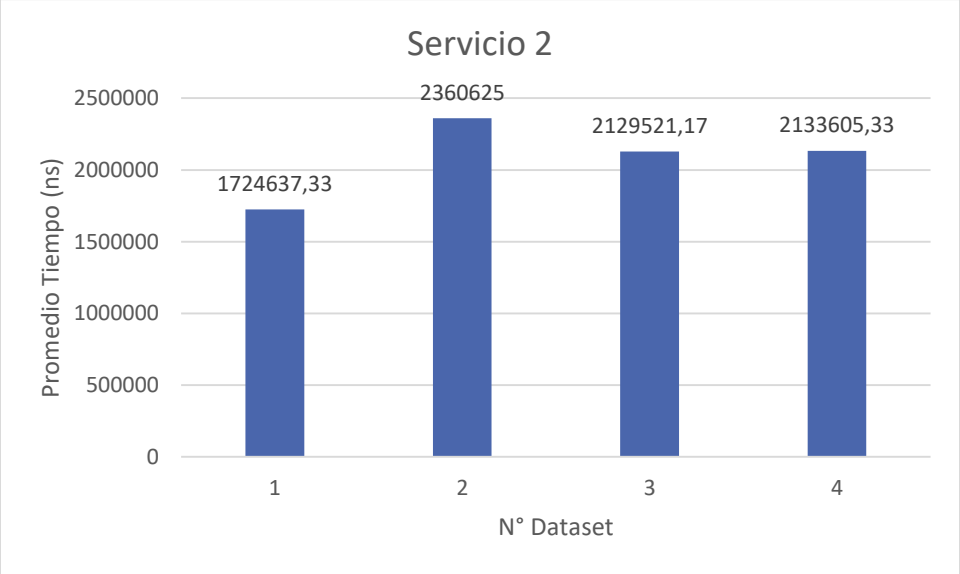
- El Dataset 1 tarda un promedio de 48301,1677
- El Dataset 2 tarda un promedio de 136022,167
- El Dataset 3 tarda un promedio de 181521
- El Dataset 4 tarda un promedio de 175869,33

Promedio para el Servicio 2:

- El Dataset 1 tarda un promedio de 1724637,33
- El Dataset 2 tarda un promedio de 2360625
- El Dataset 3 tarda un promedio de 2129521,17
- El Dataset 4 tarda un promedio de 2133605,33

6 GRÁFICOS





7 CONCLUSIÓN

Analizando los resultados y los gráficos podemos decir que:
Para el primer servicio en un Dataset de pocos valores la cantidad de relaciones entre nodos es pequeña, para el primer Dataset cada vértice tiene una media de dos adyacentes directos por lo que la búsqueda tarda menos en este. En Datasets mas grandes la cantidad de adyacentes por Nodo se va normalizando debido a que la cantidad de géneros distintos no crece demasiado, por lo que los tiempos también se van normalizando.

Para el segundo servicio debido a que deben recorrerse todas las relaciones de adyacencia la cantidad de Vértices a explorar es generalmente la de todos los ellos. Y como no hay una gran diferencia numérica de géneros distintos los resultados son muy similares.