Trabajo Práctico Especial
Programación 3
TUDAI
Segunda Entrega

Facultad de Ciencias Exactas
Universidad Nacional del Centro

Alumno: Sandoval, Fernando Daniel
Email: fdsandoval@gmail.com
Fecha de entrega: 6/6/2018

Introducción

La cátedra ahora nos solicita que implementamos una herramienta que realice un análisis de la utilización del buscador por parte de los usuarios, puntualmente la relación entre los géneros de las distintas búsquedas. La herramienta debe permitir el ingreso de categorías (géneros) a buscar, y la colección resultante deberá contener sólo los libros que cumplan con la categoría ingresada. Se nos solicita que la estructura a utilizar sea un grafo. Un grafo es un conjunto de objetos llamados vértices unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto.

Análisis

La catedra nos brinda inicialmente una serie de archivos de datos (datasets) de diverso tamaño, para comenzar a realizar nuestro trabajo. Los datos obtenidos de estos archivos deberán ser volcados en memoria en una estructura de tipo Grafo. Este puede ser dirigido o no dirigido. Esta estructura puede implementarse de dos maneras: como una matriz de adyacencias entre los vertices del grafo, o como una lista de adyacentes. Cada una tiene sus ventajas y desventajas. En el caso de la matriz de adyacencia tiene como ventaja un acceso directo con tiempo constante en caso de querer consultar si una arista se encuentra en el grafo, si queremos saber si una arista (i,j) se encuentra en el grafo con solo mirar en la matriz grafo [i,j] tendremos la respuesta. La desventaja es que ocupa mucho espacio en memoria, ya que si aun el grafo tiene pocas

aristas, el tamaño siempre es (número de vertices)^2. Además, si uno desea averiguar que vertices son adyacentes a un vertice dado *i* , hay que recorrer toda la fila *i* de la matriz incluso si es un número pequeño de adyacentes. Por otra parte, en una lista de adyacencia para averiguar si un arista *(i,j)* está en el grafo, se accede a la lista del vértice *i* en un tiempo constante y luego buscamos *j* en la lista de adyacencia. En el peor caso se tarda *O(n)* con *n* número de adyacentes del vertice *i.*

Toma de decisiones e implementación

Para decidir que estructuras utilizar, se tuvo en cuenta elegir el caso más eficiente y con menor consumo de memoria, por lo tanto nos hemos decidido por una representacion del grafo con listas de adyacencia. Este grafo será dirigido, ya que uno de los servicios solicitados asi lo requiere.

Resultados obtenidos

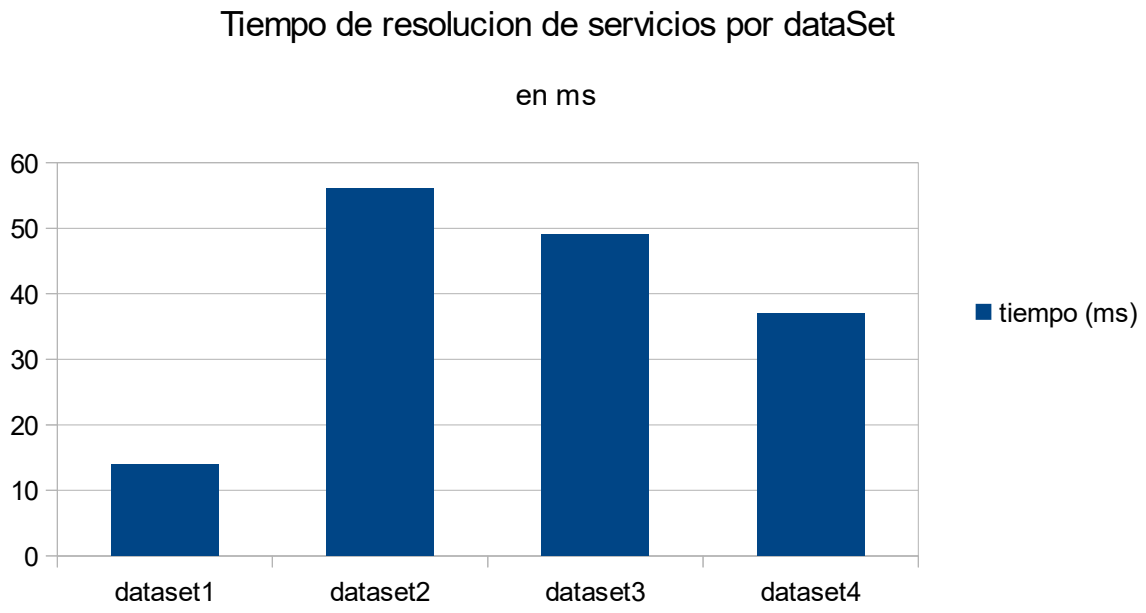
Se tuvo en cuenta para realizar las distintas mediciones el ingreso por teclado del género a ser buscado en el indice, ya que es lo único que puede hacer variar en mayor o menor medida el tiempo de ejecución. Los tiempos de lectura de los datos desde el archivo csv sólo varían dependiendo del tamaño del archivo csv. Tomamos diferentes muestras de tiempos y las comparamos a continuación:

	vertices	aristas	tiempo (ms)
dataset1	35	71	17
dataset2	40	1323	258
dataset3	40	1560	8611
dataset4	40	1560	77234

Como podemos ver, se generan 4 grafos, uno por cada dataset. El primer dataset nos genera un grafo pequeño en cantidad de aristas en comparación con los otros 3 datasets. También se puede observar un crecimiento lineal del tiempo de carga con respecto al tamaño del dataset, que no se ve reflejado en el tamaño del grafo, pero si en el peso de sus aristas. A continuación, los tiempos de resolución de los 3 servicios por cada dataset

	tiempo (ms)
dataset1	14
dataset2	56
dataset3	49
dataset4	37

En este caso no se observa un crecimiento lineal como pasaba en la carga de datos, sino que el tiempo tiende a estabilizarse. Solamente notamos diferencia en el primer grafo, que es el mas pequeño. Los otros tres grafos son similares en tamaño, y por lo tanto ejecutar los servicios sobre ellos nos devuelve tiempos no muy distintos entre sí.



Servicios

Primero se nos solicita obtener los N generos mas buscados luego de buscar por el género A. Para ello hacemos un recorrido en profundidad sobre el grafo para generar una lista de sus aristas y sus pesos. Esta lista se ordena en forma descendente para tener las aristas de mayor peso en los primeros nodos de la lista. Luego, con el N recibido como parámetro, se recorren los N primeros nodos de la lista, y se retornan como solución.
Para el segundo servicio, que es obtener todos los géneros que fueron buscados luego de buscar por un género A, se realiza sobre el grafo un recorrido en anchura iniciando por el género A, y los caminos recorridos son retornados como resultado.
Por ultimo, para el tercer servicio, que es obtener los géneros afines, se realiza dentro del grafo la busqueda de un ciclo que comienze con el género A. Una vez encontrado, el camino se guarda en un arrayList que se retorna como resultado. Luego de encontrado el primer camino, se siguen buscando otros ciclos dentro del grafo, y de encontrarse, sus elementos se

agregan al arrayList resullado en caso de que aún no se encuentren en el mismo (se descartan los elementos repetidos)

Conclusiones

En esta entrega utilizamos una estructura diferente (grafo vs lista) que nos muestra tiempos de carga lineales que dependen del número de elementos de carga, y tiempos de recorrido uniformes con respecto a otras estructuras. Además la resolución de los servicios nos permite apreciar en un ejemplo concreto la utilización práctica de los algoritmos de recorrido sobre grafos.