**Informe TP3**

**75.41 Algoritmos y Programación II**

**Cátedra Rosa Wachenchauzer**

**Cuatrimestre II, 2016**

**Corrector: Javier Choque**

**Jamilis Netanel David – Padrón: 99093**

**Sinisi Fernando – Padrón: 99139**

**Análisis y diseño de la solución:**

## **Objetivo**

El objetivo de este Trabajo Práctico es usar el TDA grafo para modelar una red social y sus algoritmos para obtener información de ella. En este caso trabajaremos con la red social conformada por los personajes de Marvel Comics.

**Dificultades**

Luego de leer el enunciado del trabajo práctico nos encontramos con las

Siguientes dificultades:

- Implementar el TDA grafo con todas sus primitivas

- Leer el archivo con los personajes de marvel y cargar los datos necesarios al grafo

En los comandos a implementar:

Para el comando similares:

* Buscar la cantidad ingresada de personajes similares, los que más aparecen junto al ingresado y tienen la mayor cantidad de adyacentes compartidos

Para el comando recomendar:

* Buscar una cantidad de personajes que no sean adyacentes al personaje ingresado pero compartan gran cantidad de adyacentes

Para el comando camino:

* Buscar un camino desde el primer personaje al segundo pasando por medio de personajes adyacentes a ellos

Para el comando centralidad:

* Buscar la cantidad de personajes mas centrales de los comics, los que más aparecen conectados a los demas

Para el comando distancias:

* Recorrer los personajes adyacentes al personaje ingresado hasta que se hayan recorrido todos, contar la distancia entre el personaje ingresado y el resto

Para el comando estadísticas:

* Calcular la información necesaria para mostrársela al usuario

Para el comando comunidades:

* Por medio de las relaciones entre los personajes buscar las comunidades que se forman entre ellos.

**Resolución:**

Para poder implementar los comandos del trabajo practico utilizamos un TDA grafo en el cual modelamos a cada personaje con un vértice, las relaciones entre personajes con aristas y los pesos entre ambos son la cantidad de apariciones de ambos personajes juntos en los comics.

Analizando el comando similares:

Para obtener los personajes más similares al ingresado utilizamos la primitiva que nos da un camino aleatorio, el camino aleatorio lo realizamos varias veces y guardamos la cantidad de veces que aparecía cada personaje en el camino. Los que aparecían mas veces eran los mas similares al personaje ingresado. Luego volcamos los resultados en una cola de prioridad de mínimos la cual va a tener la cantidad de personajes igual a la de similares que hay que devolver va a contener solo los personajes que más aparecen, el mínimo de la cola va a aparecer más veces que el resto de los personajes que quedaron afuera de esta.

Luego vamos desencolando los personajes de la cola e imprimiéndolos.

Tuvimos en cuenta de no hacer muy largo el recorrido porque sino los similares darían siempre igual a los personajes centrales.

El orden de este algorimo depende del producto del largo del recorrido y de la cantidad de recorridos.

Analizando el comando recomendar:

Este algorimo es muy similar al similares, ya que lo que busca es recomendar a alguien que es similar a uno pero que aun no lo conoce. Para cumplir esa condición lo que se busca es que sea similar pero no este en los adyacentes de uno, entre los personajes que uno conoce.

El orden de este algoritmo es el mismo que el de similares.

Analizando el comando camino:

Como se pide que el camino sea mínimo desde el primer personaje al segundo, ya que los pesos entre los personajes son positivos, se utiliza el algoritmo de Dijkstra. Este algorimo sirve para encontrar el camino mínimo entre dos vértices de un grafo conexo.

El orden de este algorimo es  O(|A| log |V|) siendo A la cantidad de aristas y V la de vertices.

Analizando el comando centralidad:

Para implementar este comando se utiliza una estrategia muy similar a la del primer comando, la de similares. Lo que hacemos es obtener muchos recorridos aleatorios, del orden de la mitad de personajes que hay en el grafo, y de todos los recorridos guardamos la cantidad de veces que aparece cada personaje. Con esa información armamos armamos una cola de prioridad de mínimos, y mantenemos a los personajes mas centrales de marvel, la cantidad de personajes que pide el usuario.

Utilizamos la estrategia de caminos aleatorios porque aunque su orden sea peor que usando centralidad exacta, este es más fácil de programar.

El orden de este algoritmo es el mismo que el de similares, ya que lo único que varía es la cantidad de recorridos y que se empieza siempre por un personaje aleatorio.

Analizando el comando distancias:

Para implementar este comando utilizamos el recorrido BFS, búsqueda en amplitud, que lo que hace es, como podía la documentacion del comando, ir abriendo paso de a saltos. Es decir, buscar primero los que estan a distancia uno del personaje, luego a distancia dos, y así sucesivamente.

A medida que vamos haciendo el recorrido de los personajes lo vamos haciendo es contando la cantidad de personajes que estan a cada distancia del ingresado por el usuario, al cual le asignamos distancia cero.

Al finalizar el recorrido mostramos por pantalla la cantidad de personajes que se encuentran a cada distancia.

La complejidad de este algoritmo es la misma que la del recorrido BFS. Esta sería O(|V|+|E|) siendo V la cantidad de vertices y E la de aristas.

Analizando el comando estadísticas:

Este comando brinda información básica de como esta conformado el grafo al usuario.

La cantidad de vértices es la longitud del grafo, que es lo mismo que la longitud del diccionario que contiene a los vértices del grafo. El grafo mantiene este numero como una invariante. Lo devuelve en O(1).

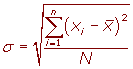
La cantidad de aristas se obtiene por medio de una primitiva con el mismo nombre. La estructura del grafo posee esta invariante, siempre mantiene actualizada en una variable la cantidad de aristas. Lo devuelve en O(1).

El promedio de grado se calcula como la suma de la cantidad de adyacentes a cada vértice del grafo dividido la cantidad de vértices del grafo. El orden de este algoritmo tiene un orden de O(V \* A) porque depende de la cantidad de vértices del grafo y de la cantidad de adyacentes que puede tener cada vértice.

La desviación estándar es la raíz cuadrada de la varianza.

Es decir, la raíz cuadrada de la media de los cuadrados de las puntuaciones de desviación.

La desviación estándar se representa por σ. Y se define por la siguiente formula matemática:



El orden de este algoritmo también depende de la cantidad de vértices y aristas, es O(V\*A).

La densidad de un grafo se calcula como dos veces la cantidad de aristas dividida la cantidad de de aristas máxima, que esta seria V\*(V-1). El orden de este algorimo es O(V+A).

El orden del algoritmo de estadísticas es el mayor de todos los ordenes, este sería O(V\*A).

Analizando el comando comunidades: