2017

TRABAJO PRÁCTICO

PROGRAMACION III

CIFUENTES/VÁZQUEZ

ETAPA 3

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

Introducción

1 GRAFO DE GUSTOS DE LOS USUARIOS.

Desde la cátedra se nos pide desarrollar un Grafo de gustos de los usuarios, este grafo tendrá relaciones entre los ids de usuario y los gustos pero estos ultimos no deberán de repetirse.
Para la implementación reutilizamos estructuras desarrolladas previamente durante el transcurso del cuatrimestre.

2 IMPLEMENTACIÓN

Decimos utilizar la estructura de un grafo no dirigido para que los nodos mantuviesen su relación bidireccional de “usuario-gusto”.
Este grafo se encuentra compuesto por un **Vector** de Vértices, dentro de los cuales se encuentra:

- Un **id** que lo identifica con su posición dentro del Vector.
- El **dato** que contiene.
- El **tipo** de dicho dato.
- Una **lista de vértices** adyacentes a él.

Decidimos que el Vértice debía de contener un id, debido a que en algunos de los servicios necesitábamos conocer la ubicación del vértice dentro del grafo.
El tipo de dato, por su parte, es un **Enum** utilizado para distinguir entre un *usuario* y un *gusto,* debido a que ambos son elementos de tipo **String** y al estar dentro del mismo grafo, su diferenciación fue necesaria;

3 INTERFACES

En el trabajo se nos pedía desarrollar una interfaz que funcionase con cualquier estructura que desee implementarla.
Métodos:

- **agregarUsuario (String dni):** Este método se encarga de agregar un usuario dentro del grafo, para ello hacemos uso del método privado **addVertice(Vertice v)** el cual se encarga de añadirlo al vector de Vértices.

- **agregarGusto(String gusto):** Este método se encarga de agregar un usuario dentro del grafo, para ello primero controlamos de que el gusto no haya sido ingresado previamente. Luego hacemos uso del método privado **addVertice(Vertice v)** el cual se encarga de añadirlo al vector de Vértices.
- **existe(String dniOgusto):** Con este método al recibir un String recorremos el vector de Vértices, en busca del dni o del gusto deseado, y se notifica su existencia con TRUE o FALSE.

4 SERVICIOS

- Dado un usuario, mostrarle las personas que tienen más de un gusto en común con él:
  El servicio se es brindado con la función **gustosSimilares (String u)**, en donde se recorre el vector de Vértices comparando los gustos en la lista de adyacencia del usuario recibido contra los gustos en la lista de adyacencia del resto de usuarios. Las coincidencias se van registrando en un contador de gustos que cuando sea > 1 se agrega el usuario comparado a un arreglo solución que será retornado al finalizar.
- Mostrar qué gusto le gusta a más cantidad de gente:
  El servicio se es brindado con la función **masGustado ()**, donde se recorre el vector de Vértices buscando los gustos y devolviendo el gusto que posea la lista de adyacencia más extensa.
- Dada una persona, mostrar aquella que tenga gustos más lejanos a él:
  El servicio se es brindado con la función **usuarioGustosMasLejanos ()**, en donde se implementó un algoritmo Dijkstra, el cual, dado un dni de usuario retorna un arreglo con los caminos más cortos del recorrido del grafo sin repetir gustos; luego en la función principal de este arreglo se toma el valor más grande, retornando así el usuario final del recorrido.

5 HASHING

¿Se podrá realizar el mapeo de alguna manera más eficiente utilizando técnicas de hashing?

Utilizando una técnica de Hashing abierta el mapeo podría realizarse de forma más eficiente, para esto deberíamos cambiar el vector de Vértices, para pasar a ser uno de Listas de adyacencia y realizar una relación o traducción del dni o el gusto con la posición a la que hace referencia en el Vector para poder agregarlo al Hashing.

Cuando se quiera realizar una búsqueda, solo sería necesario traducir el dni o gusto a número y al aplicar la función de hashing se conseguiría su valor en el vector de adyacencias dándonos el acceso a el mismo y su lista de adyacencias.