Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2019-2

17 de octubre de 2019
Actividad Formativa

Actividad 08

Estructuras de datos: Grafos

Introducción

Luego de las constantes luchas contra los alumnos, el *Chief* Enzini se ha dado cuenta que está perdiendo popularidad entre los alumnos. Debido a su gran necesidad de seguir siendo el *Chief*, decide hacer algo al respecto y buscar una forma para poder interactuar de manera directa con los alumnos. Es por esto, que le pide a su asistente, el *Sub Chief* Pinto, que le ayude con la creación de una red social para recuperar su popularidad con los alumnos. Sin embargo, el *Sub Chief* Pinto decide aprovecharse de la situación para lograr ascender al poder y decide usar la red social para su beneficio y les pide a ustedes, sus queridos ~~*súb*ditos~~ alumnos de IIC2233, que lo ayuden a crear **Pintogram**.

Pintogram

Pintogram es una red social que los usuarios usarán para compartir sus mejores y más hermosas líneas de código. El *Sub Chief* Pinto te encarga que hagas una primera versión de la red, donde aún no existen publicaciones, pero las distintas personas pueden crear su propia cuenta y seguirse mutuamente. Además, te pide que los usuarios puedan ver su "distancia social" a otro usuario junto con otras consultas explicadas más adelante.

Archivos

Los usuarios de Pintogram poseen un nombre y un ID único que los representa dentro de la red. Cuando un usuario sigue a otro, no es necesariamente mutuo. Por ejemplo, Enzini puede seguir a Pinto, pero esto no significa que Pinto siga a Enzini. Para que pruebes tu red, el *Sub Chief* Pinto te ha entregado archivos para que puedas poblarla de usuarios y testear su funcionamiento. Estos archivos poseen lineas con el siguiente formato:

```
ID_USUARIO,NOMBRE,ID_SEGUIDO1;ID_SEGUIDO2...
ID_USUARIO,NOMBRE,ID_SEGUIDO1;ID_SEGUIDO2...
.
.
.
ID_USUARIO,NOMBRE,ID_SEGUIDO1;ID_SEGUIDO2...
```

Cada línea posee el ID del usuario, su nombre y los ID de todos los usuarios que sigue. Recuerde que la relación de seguimiento en Pintogram no es necesariamente mutua. La función `cargar_archivos(path)` se encuentra implementada en el módulo `cargar.py` y retorna un generador de las lineas de el archivo que se encuentra en `path`.

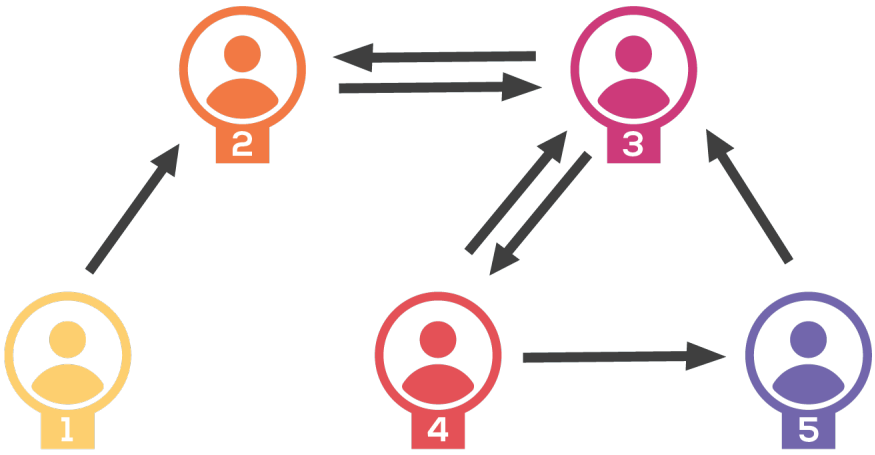

Figura 1: Ejemplo de relaciones en grafo de `simple.txt`

Los Usuarios

Debido a que Pintogram está en su etapa más temprana, tu primera tarea será cargar los datos de todos los usuarios y crear la red social. Para esto se encuentra dada la clase `Usuario`, en la cual deben almacenar los datos entregados en los archivos, para poder utilizarlos más adelante. Además, se encuentra la clase `Pintogram` que será la encargada de manejar las funcionalidades de la red.

Clase `Pintogram`

Una vez creada la red, el *Sub Chief* Pinto te ha dado la misión de darle algunas funcionalidades a Pintogram para que los usuarios puedan obtener información de su cuenta y otras. Para poder realizar las siguientes funcionalidades y consultas correspondientes a la red social, debes implementar una clase `Pintogram` que pueda almacenar a los usuarios creados anteriormente y además, posea los siguientes métodos:

- `def nuevo_usuario(self, id, nombre):` Esta función recibe como parámetro el ID de un nuevo usuario, el cual debes incluir en la red.
- `def follow(self, id_seguidor, id_seguido):` Esta función recibe como parámetro el ID de dos usuarios y debe permitir que el correspondiente al seguidor pueda seguir al seguido. Cómo no se puede seguir dos veces al mismo usuario, debes evitar que esta situación suceda.

- `def cargar_red(self, ruta_red)`: Esta función debe encargarse de cargar la red social. Para esto debes leer los datos de los usuarios desde el `path` dado y almacenar cada uno de ellos en la estructura que implementaste anteriormente. A medida que cargas los usuarios, debes asegurarte de ir realizando las conexiones entre ellos. Por ultimo, es necesario almacenar a cada uno de los usuarios dentro de la red Pintogram.
- `def unfollow(self, id_seguidor, id_seguido)`: Esta función recibe como parámetro el ID de dos usuarios y debe permitir que el correspondiente al seguidor pueda dejar de seguir al usuario seguido. Si es que intento hacer *unfollow* a un usuario que no está en mis seguidos, el programa no debe permitirlo.
- `def mis_seguidos(self, id_usuario)`: Los usuarios de Pintogram necesitan saber cuánta gente siguen ellos. Esta función recibe como parámetro el ID de un usuario y debe retornar un número con la cantidad de personas que el usuario sigue.
- `def distancia_social(self, id_usuario_1, id_usuario_2)`: Para sus planes *Chief* Enzini quiere saber la **distancia social** entre dos usuarios, dados sus IDs. La distancia social es la **cantidad de conexiones que separan a dos usuarios**.

  Por ejemplo, si vemos la **Figura 1**, el *Usuario_1* no sigue al *Usuario_3*, pero si sigue al *Usuario_2*. Además, el *Usuario_2* sigue a *Usuario_3*. Podemos ver entonces, que existe una conexión indirecta entre el *Usuario_1* y *Usuario_3*, a través del *Usuario_2*. Para este caso, la distancia social entre los usuarios *Usuario_1* y *Usuario_3* tomaría el valor de 2, debido a que es necesario hacer dos conexiones para llegar desde el *Usuario_1* al *Usuario_3*. Si es que no existe una posible conexión entre dos usuarios, se dice que su distancia social es infinita.[1] También es necesario notar, que la distancia social considera la dirección entre las conexiones: la distancia social entre *Usuario_1* y *Usuario_2* es 1 debido a que estan conectados directamente; pero la distancia social entre *Usuario_2* y *Usuario_1* es infinita, ya que no hay forma de llegar **desde** *Usuario_2* hacia *Usuario_1*.

Requerimientos

- (3.00 pts) Construir Red
  - (0.50 pts) Leer el archivo y modelar correctamente a los usuarios y a la red social.
  - (0.50 pts) Función `nuevo_usuario` implementada correctamente.
  - (0.50 pts) Función `follow` implementada correctamente.
  - (0.50 pts) Función `unfollow` implementada correctamente
  - (1.00 pts) Función `cargar_red` implementada correctamente.
- (3.00 pts) Consultas
  - (1.00 pts) Función `mis_seguidos` implementada correctamente.
  - (2.00 pts) Función `distancia_social` implementada correctamente.

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AC08/`

[1] Para representar el infinito en Pyhton puedes hacer `float("inf")`

- **Hora del *push*:** 16:40

Auto-evaluación

Como esta corresponde a una actividad formativa, te extendemos la instancia de responder, después de terminada la actividad, una auto-evaluación de tu desempeño. Esta se habilitará a las **16:50 de jueves 17 de octubre** y tendrás plazo para responderla hasta las **23:59 del día siguiente**. Puedes acceder al formulario mediante el siguiente enlace:

`https://forms.gle/S7FdRiA3GA5e1TzSA`

El asistir, realizar la actividad y responder la auto-evaluación otorgará como bonificación al alumno 2 décimas para sumar en su mejor actividad sumativa del semestre.