Trabajo Práctico 2

- Alumnos:
  - Garcia Sanchez Julian – 10 45 90
  - Zimbimbakis Francisco Manuel – 10 32 95

- Materia: Algoritmos Y Programacion 2 – curso Buchwald
- Corrector: Federico Brasburg

Introduccion:

Se nos pidió implementar una nueva red social llamada AlgoGram. Tuvimos que realizar la implementación del sistema de posteos y que esto le llegue a los demás usuarios, y que cada usuario pueda tener un feed que priorice la relación con otros usuarios, poder likear un post, entre otras opciones. Esta red social debe funcionar como una aplicación de consola.

La complejidad de los comandos fue la mayor prioridad al momento de decidir el camino a utilizar para la forma de desarrollo del programa implementado.

Estructura:

Esta red social esta estructurada con por dos pilares: las estructuras donde se guarda la informacion y el mapa donde se encuentran los comandos desarrollados.
El primer pilar consta de 2 estructuras tipo slice (ya que está implementado en Go), una de usuarios donde la posición de cada uno corresponde con el id propio y unico del usuario, y el otro de publicaciones con la misma idea respecto su id. Mas adelante se desarrollará respecto a estos tipos de dato.
El segundo pilar es un diccionario de funciones (los comandos pedidos por el tp) que linkea cada comando escrito por el usuario en el stdin a su respectiva función para ser ejecurtada con los parametros recibidos por el mismo medio.
Ademas de eso, el programa fue modularizado en dos principales tipos de datos abstractos (TDAs), usuario y post, que se encuentran en las interfaces con esos nombres.

Tda usuario:

Se desarrolló este TDA con el fin de obtener una abstracción respecto del comportamiento del feed de cada usuario y la afinidad entre estos.
Dentro de su comportamiento podemos: crear un usuario nuevo, agregar un post a su feed, ver el siguiente post y obtener sus datos. Todo esto metido dentro de una interface Usuario.
Mirando un poco su estructura interna, lo mas llamativo es el uso de un heap para el desarrollo del feed de publicaciones. Este se utilizó con el fin de lograr la complejidad O (log (p)) (siendo p la cantidad de posts creados), ya que el desencolar y encolar de heap tiene esa complejidad, logrando que ver proximo post y publicar post tenga esa complejidad para cada usuario (en el caso de publicar el programa lo hará en O(u log(p)), siendo u la cantidad de usuarios).

Tda post:

Con la misma idea del TDA anterior respecto a la abstracción y comportamientos respecto a los posts, se desarrolló este tipo de dato.
Dentro de su comportamiento podemos: crear un post nuevo, likear un post, ver el mensaje del post y ver sus likes. Todo nuevamente metido en la interface Post.
Mirando su estructura interna, se utilizó un diccionario ordenado para el manejo de los likes recibidos en cada post. Este se utilizó con el fin de llegar a una complejidad de

$O(u_p)$ (siendo u_p la cantidad de usuarios que likearon el post) al mostrar los likes en orden alfabetico y $O(\log(u_p))$ al agregar un like al post. Todo esto gracias a las propiedades del diccionario al guardar e iterar inorder.

Login y logout:

El comando de inicio de sesión y fin de la sesión no requirió de un TDA nuevo. Para estos se utilizó el TDA hash implementado durante la cursada, en el cual todas sus funciones tienen una complejidad de O(1).
Login y logout utilizan un diccionario de posiciones y busca y encuentra, si es que existe, el usuario en O(1), ubicado en algún lugar del slice de usuarios mencionado anteriormente.

Procesamiento y errores:

Como se explicó en la parte de estructuras, los post estan ubicados en un slice donde sus ids corresponde con su ubicación, por lo tanto se encuentra cada post en O(1).

Los tipos de errores del programa estan en un archivo específico para hacer más estructurado y legible el codigo. Funcionan como un tipo error clasico de Go.

En el archivo llamado procesamiento se encuentran la mayoria de las funciones auxiliares que validan los datos obtenidos por medio del stdin y contruyen partes de la estructura para guardar y obtener los distintos tipos de datos, tales como el diccionario de usuarios y el slice de los mismos.

Conclusión:

Se obtuvo un programa robusto, eficiente y con las complejidades pedidas en cada uno de los comandos posibles. Se utilizaron muchos de los TDAs desarrollados a lo largo de la cursada y se perfeccionaron las tecnicas de modularización y abstracción de la programación. Tambien cabe destacar la legibilidad del codigo, asi como tambien su simplicidad y explicación comentada.