Trabajo Practico Nº2

Programación II

Segundo Semestre 2016

Ejercicio 2: Polimorfismo

Estrategia:

Uno de los primeros problemas que se me presento fue el pensar en cómo englobar correctamente ambos tipos de red social, ya que de hacerlo de forma incorrecta sería necesario repetir mucho código y la consigna no se cumpliría. Lo primero en lo que pensé es en cómo se comporta cada una y sobre todo que tienen en común, y así tratar de poner todo lo similar dentro de una clase abstracta llamada RedSocial. Pero lo que mayormente tenían en común eran todas las funciones de RedAsimétrica, así que decidí utilizar a RedSocial como una interface para que englobe ambos tipos de red. Aquí me surgió una duda ya que con esta implementación venia otra en serie que es la de RedSimétrica que sería una extensión de RedAsimétrica y por lo tanto heredaría todas sus funcionalidades, pero de este modo RedAsimétrica estaría englobando también a RedSimétrica, por lo que se podría declarar a una red simétrica como una asimétrica. Como se me aclaro en clase esta contradicción llevaría a que los algoritmos que son exclusivos para redes asimétricas provocarían un error en una red simétrica declarada como tal. Pero como la consigna no especifica esta funcionalidad me maneje con este esquema.

Complejidad:

El segundo problema era mantener las complejidades de los algoritmos en O(1), para esto fue de gran ayuda el uso de diccionarios. El único inconveniente que no me quedo muy en claro es un conflicto de funcionalidad entre obtenerRelaciones(Persona a) y agregarRelación(Persona a, Persona b). Para agregar una relación y poder mantener el conteo correcto de las relaciones en la red es necesario el uso de diccionarios "enlazados" para que se pueda consultar si una persona ya está relacionada con otra con una complejidad O(1), pero esto trae el inconveniente de que a la hora de obtener las relaciones de una persona esta devuelve un diccionario que requiere el uso de iteradores para recorrerlo y no es tan cómodo como una lista o arreglo. Y como no conozco la funcionalidad de la librería de iteradores de java, no puedo garantizar que se recorra con una complejidad de $O(m_a)$ como lo pide en el apartado 2.4 (siendo m_a el número de relaciones de esta persona). Me gustaría se me aclare esta duda.

Como un punto fuerte de esta implementación me parece que cabe destacar que al conectar los diccionarios en la implementación esto hace que en futuras funcionalidades para las redes sea posible acceder a las relaciones de una persona, desde las relaciones de otra (esto se ve muy claro en redes como Facebook o Twitter, en las que se puede ver los amigos de una persona y acceder a sus perfiles y amigos, como así también a sus seguidores).