RESUMEN ALGORITMOS EN ACCION

Preguntas sobre el tp (2do cuatri 2022): https://www.youtube.com/watch?v=a2dsOjENu0Q

EJ 1 nombresDeUsuarios:

* params: redSocial
* Qué hace?: Recibe una red social y devuelve los nombres de todos los usuarios utilizando recursion sobre listas.
* Como?: Recorre todos los usuarios de la red.
* Tests:
  -- a. Cuando no hay usuarios en la red debe devolver una lista vacia
  -- b. Hay un usuario solo en la red social debe devolver una lista con el nombre del unico usuario
  -- c. Hay mas de un usuario, debe devolver los nombres de todos esos usuarios en la red

---

EJ 2 amigosDe:

* params: redSocial, usuario
* Qué hace?: dada una red social y un usuario valido, devuelve una lista de los usuarios q son amigos de ese usuario dado.
* Cómo?:
  - recorre todas las relaciones y va "uniendo" las listas provinientes de procesar la relacion por la auxiliar "amigo".
  - usamos el operador (++) porque cuando recorremos las relaciones pueden haber algunas en las que el usuario no participa (union de listas)
* Auxiliar (amigo): Usamos como auxiliar la funcion amigo, que dada una relacion y un usuario, devuelve [] si el usuario no esta en la relacion, y el otro usuario en la relacion, si el usuario dado si pertenece a la relacion.
* Tests:
    -- a. La red no tiene relaciones
    -- b. La red tiene relaciones pero el usuario no tiene amigos,
    -- c. La red tiene relaciones, el usuario tiene 1 amigo y este aparece en la segunda coordenada de la relacion
    -- d. La red tiene relaciones, el usuario tiene 1 amigo y este aparece en la primera coordenada de la relacion
    -- e. La red tiene relaciones, el usuario tiene mas de 1 amigo y este aparece siempre en la primera coordenada de la relacion
    -- f. La red tiene relaciones, el usuario tiene mas de 1 amigo y este aparece siempre en la segunda coordenada de la relacion
    -- g. La red tiene relaciones, el usuario tiene mas de 1 amigo y este aparece tanto en la primera como en la segunda coordenada

OBS: Se podria haber hecho sin auxiliar con una guarda
| usuario in relacion = amigo : amigosDe (resto)
| otherwise = amigosDe (resto)

EJ 3 cantidadDeAmigos:

* params: redSocial, usuario
* Que hace?: dada una red social y un usuario te da la longitud de amigosDe(red,usuario)
* Cómo?: auxiliar longitud
* Tests:
    -- a. La red no tiene relaciones
    -- b. La red tiene relaciones y el usuario no tiene amigos:
    -- c. La red tiene relaciones y el usuario tiene 1 amigo:
    -- d. La red tiene relaciones y el usuario tiene mas de 1 amigo:

EJ 4 usuarioConMasAmigos:

* params: redSocial
* Que hace?: dada una RedSocial, devuelve el usuario con mas amigos, devolvemos el usuario que aparece más veces en la lista de relaciones.

* Cómo?: recorro los usuarios y comparo la cantidad de amigos del primero con el que tiene mas amigos del resto de la lista de usuarios.
* Test:
  -- a. La red tiene 1 solo usuario
  -- b. La red tiene mas de 1 usuario, pero no tiene relaciones
  -- c. La red tiene mas de 1 usuario y hay un unico usuario con mas amigos
  -- d. La red tiene mas de 1 usuario y hay mas de un usuario con la max cantidad de amigos

---

EJ 5 estaRobertoCarlos:

* params: redSocial
* Qué hace?: Dado una red social, devuelve true si algun usuario tiene mas de 10 amigos
* Cómo?: una auxiliar hayUnUsuarioConMasDeNAmigos(redsocial), se fija la cantidad de amigos del usuario con mas amigos de esa red social (usa las aux de los ejs anteriores)
* Tests:
  -- a. La red no tiene un Roberto Carlos
  -- b. La red tiene un Roberto Carlos
  -- c. La red tiene mas de un Roberto Carlos

EJ 6 publicacionesDe:

+ params: redSocial, user
* Que hace? dada una red social y un usuario te devuelve una lista con todas las publicaciones de ese usuario.
* Cómo? recorre las publicaciones, si el usuario de la publicacino es el usuario dado la agrega como cabeza de la lista de publicaciones del usuario en las publicaciones restantes
* Tests:
  -- a. La red no tiene publicaciones
  -- b. PublicacionesDe red con publicaciones, usuario sin posts
  -- c. PublicacionesDe red con publicaciones usuario con un post
  -- d. PublicacionesDe red con publicaciones usuario con mas de un post

---

EJ 7 publicacionesQueLeGustanA:

- params: redSocial, user
* Que hace? dada una red social y un usuario devuelve una lista con todas las publicaiones que likeo ese usuario
* Como? recorre las publicaciones, si el usuario esta en los likes de la publicacoin la agrega como cabeza de la lista de publicaciones que likeo el usuario las publicaciones restantes.

```
pertenece :: t -> [t] -> Bool
pertenece el [] = False
pertenece el (x:xs) | x == el = True
                    | otherwise = pertenece el xs
```

* Tests:
  -- a. La red no tiene publicaciones

-- b. El usuario no likeo ningun post
-- c. El usuario likeo 1 post
-- d. El usuario likeo mas de 1 post, resultado esperado = una lista con todas las publicaciones que gustaron al usuario (en cualquier orden)

Ej 8: lesGustanLasMismasPublicaciones.

-params: redSocial, user,user
* Que hace? dada una red social y dos users, devuelve True si le gustan las mismas publicaciones
* Como? compara la doble inclusion de las listas que da publicacionesQueLeGustan a ambos usuarios.
* tests:
-- a. A ambos usuarios le gustan las mismas publicaciones (1 sola)
-- b. A ambos usuarios le gustan las mismas publicaciones (mas de 1)
-- c. A los usuarios no les gustan las mismas publicaciones
-- d. No hay publicaciones

EJ 9: tieneUnSeguidorFiel

- params: redSocial, user
* Que hace? devuelve true si el usuario dado tiene posts y algun usuario (distinto a él) likeo todos sus posts
* Como? se fija el caso base (nadie likeo el post), recorre la lista de usuarios hasta que se vacie (devuelve False) o algun usuario haya likeado todos sus posts (True)
* Auxiliar likeoTodosLosPosts: dado un usuario y una lista de publicaciones, devuelve True <=> el usuario likeo a todas (recorre publicaciones).
* Tests:
-- a. Usuario sin publicaciones
-- b. Usuario con publicaciones, sin seguidor fiel
-- c. Usuario con publicaciones, con seguidor fiel
-- d. Usuario con publicaciones, con mas de 1 seguidor fiel

EJ 10: existeSecuenciaDeAmigos

- params: redSocial, usuario1, usuario2
* Que hace? Se fija si los usuarios estan en la misma particion determinada por la relacion amigo en la red social.

* Como? Hace una recursion indirecta, se fija si existe un camino entre los amigos del usuario y el usuario destino,
guardando registro de los usuarios que visito.

* Auxiliares:
-- existeSecuenciaDeAmigosSin: devuelve True cuando existe un camino entre el user1 y el user2, sin pasar por los nodos determinados en chequeados.

-- algunAmigoTieneSecuenciaA: dado una lista de amigos con al menos un elemento, un usuario final y los nodos registrados devuelve True <=> existe un camino de amigos entre algun amigo y el usuario final, sin pasar por los nodos ya chequeados.
-- relacionadosDirecto: dada una red y dos usuarios validos, devuelve true si son amigos, es decir, si existe una
tupla (u1,u2) o (u2,u1) en las relaciones de la red.

* Tests:
  -- a. Red sin relaciones
  -- b. Con relaciones, ambos usuarios en la misma particion, relacionados directamente
  -- c. Con relaciones, ambos usuarios en la misma particion, relacionados indirectamente
  -- d. Con relaciones, usuarios en distinta particion
  -- e. Con relaciones, usuarios con relacion indirecta por segundo camino