Социална мрежа
Автор: Данаил Димитров

В този забързан и изпълнен с Интернет свят, всички участват в различни виртуални социални мрежи. Нека опитаме и ние да се впишем и да напишем библиотека за част от функционалността на една социална мрежа.

В нашата социална мрежа ще искаме да са налице следните функционалности:

- ***Добавяне на потребител (име, години)*** - добавя потребител в нашето множество от потребители, като допускаме само **уникални имена**
- ***Създаване на приятелство/връзка между потребители (име1, име2[, тип_приятелство])*** - създава връзка на приятелство между потребители с име1 и име2
    - където, типовете приятелства биват: *bestie*, *relative*, *normal*
    - по подразбиране приятелството е от тип "normal"
- **Търсене на потребител (име)** - намираме потребител с зададеното име; За този потрибител получаваме информация за:
    - Има и години
    - Списък от приятели
- ***Забрана (ban) на потребител (име1, име2)*** - потребител с име1 не желае да има каквото и да е общо с потребител с име2
- ***Премахване на потребител (име)*** - премахва потребител от нашето множество от потребители. *Все пак винаги можем да решим да избягаме от виртуалния живот*
- ***Премахване на приятелство (име1, име2)*** - премахва връзката между двамата потребители

Би било добре да не губим създадените потребители и връзки в случай на проблем с машината ни, за целта ще е добре **да съхраняваме данните по подходящ начин във файл.** Вие преценете как ще е най-удобно това да се случи

Също така, би било добре да помагаме на хората да откриват нови "приятели". За целта ще се нуждаем от функционалност за предлагане на хора, които може би се познават. Тя би включвала множество от функционалности, описани по-долу.

Предложения за потребител (име1) - дава списък с предложение и идеи за нови приятели за потребител с *име1*. И малко детайли:

- Колкото повече общи приятели има нашият потребител с друг потребител, то вероятността да се познават е по-голяма
- Също така колкото по-близки негови приятели са приятели с този потребител, толкова вероятността те да се познават е по-голяма - например, ако общият им приятел е от тип *bestie* шансът би бил 3 пъти повече и съответно: за *family* - 2x, а за *normal* - 1x
    - Можем да изчисляваме коефициенти на базата на тази идея

- Ако потребителят няма никакви приятели, то нека му предлагаме най-дружелюбните хора (тези с най-много приятели)
- Никога не предлагаме забранени (banned) от име1 потребители;
- Бихме желали да не предлагаме повече от 30 души (количество, което може да се разгледа от потребителя)

Нашата Задача

Крайната ни цел е да постигнем едно функционално и използваемо решение на гореописаните функционалностти. За целта:

Създайте RELP (Read eval print loop), който да поддържа следните команди, отражение на елементите изброени горе:

- CREATE <name> <e-mail> <age>
    - Съобщение за успех, ако е създаден
    - Съобщение за неуспех, ако такъв потребител съществува
- DELETE <e-mail>
- LINK <name-1> <name-2> <type>
    - Съобщение за успех, ако е създадена
    - Съобщение за неуспех, при неуспех - banned потребител
- FIND <name>
    - None - не е намерен потребител
- BAN <name-1> <name-2>
- DELINK <name-1> <name-2>
- RECOMMEND <name-1>

Примерен вход и изход (повече от десетки думи...):

> CREATE Dido 23
User Dido created
> CREATE Dido 24
FAIL: Dido already exists
> CREATE Stamat 23
User Stamat created
> LINK Dido Stamat family
Users Stamat and Dido are family now
> FIND Dido
User

Name: Dido
Age: 23
Friends: Stamat

> DELETE Stamat
User Stamat has been deleted
> FIND Stamat
None
> FIND Dido
User

Name: Dido
Age: 23
Friends: none

> CREATE Danny 25
User Dido created
> CREATE Evlogii 24
User Evlogii created
> BAN Evlogii Danny
User Danny is now banned by Evlogii
> LINK Danny Evlogii
FAIL: User Danny is banned

За ентусиасти: Работа с огромни данни - какво ще трябва да промените, ако не можете да заредите всичко в паметта? Как бихте оптимизирали максимално, така че да не се налага да се използва само бавното четене от файл?