Friend book

A Friends Recommendation system

—

Team 42

Girija

Nitin

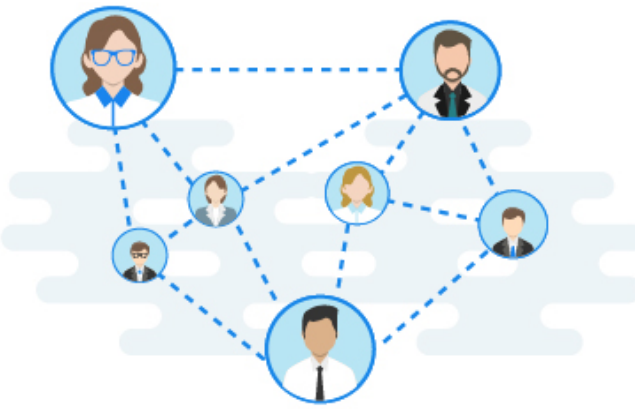
Sambasai

Vamshi

Vedant

Overview

Friend Book is a Friends Recommendation system similar to Facebook. It has basic functions such as adding users to a database and allocating IDs to them, adding friends to these users and recommending friends to these users based on common parameters and mutual friends.



Function Specifications

Register

Data Structures used: dynamic array/vector, Priority queue/min heap
Algorithms used:

1) Allocates memory for each user and stores its pointer in a dynamic array. **O(1)**
2) Allocates the least positive integer as user ID, obtained from a priority queue if it is not empty. **O(1)**

[The priority queue mentioned here stores the user ID of a person whenever they unregister]

Unregister

Data Structures used: Priority queue/min heap, hash tables (friends and anti friends adjacency list).
Algorithms used:

1) Frees memory allocated in "Register" function and pushes the user ID into the priority queue(mentioned previously). **O(log(number of unregistered users))**
2) Traverses through the anti friends list(a graph adjacency list which contains the opposite directed edge of each edge) of unregistered user and removes the unregistered user from each of their friends lists. **O(number of anti friends)**
3) Traverses through the friends list of unregistered user and removes the unregistered user from each of their anti friends list. **O(number of friends)**

NOTE: removing a user takes **O(1)** time since we need to find it in a hash table.

Recommend friends (has 2 variants)

Existing user

Data Structures used: hash tables (friends adjacency list), Queue.
Algorithms used:

1) BFS **O(total number of existing users)**

New user

Data Structures used: Array of linked lists
Algorithms used:

1) Traverse through the user list. **O(Number of registered users)**
2) For every user you encounter, compute the number of common parameters between the newly registered user and the user encountered during traversal. **O(1)**
3) Insert the encountered user in the linked list corresponding to the number of common parameters. **O(1)**
4) Iterate through the array of linked lists starting with the linked list with the most number of common parameters and recommend the top 10 users. **O(1)**

Check Friendship Status

Data Structures used: hash tables (friends adjacency list)
Algorithms used:

1) searching in hash table **O(1)**

Add Friends

Data Structures used: hash tables (friends adjacency list)
Algorithms used:

1) Inserting into hash table **O(1)**
2) If the hash table reaches its maximum capacity, a realloc function would be called **O(number of friends of user)**

Division of work

I. Girija

1) Implementation of Hash table (adjacency list for graph) and functions related to it Hash table
    a) Search (check friendship status)
    b) Insert
    c) Remove
    d) IsEmpty etc.

II. Nitin

1) Priority Queue Implementation
2) Register Function

III. Sambasai

1) Queue Implementation
2) Unregister Function
3) Debugging code

IV. Vamshi

1) Recommendation Function for existing Users
2) Implementation of userlist(dynamic array) and Add Friends functions
3) Merging and debugging code
4) Project planning, readme and report writing

V. Vedant

1) Recommendation Function for newly registering users
2) User Interface
3) Merging and debugging code
4) Project planning