Rapport du projet: Algo avancé

Réalisé par Rihem Garrouch
Bejaoui Ilhem

Présentation du projet :

Le projet consiste à développer les structures de données et les méthodes nécessaires pour réaliser un réseau social en java.

1) Présentation des structures de données:

membre=
enreg

Nom : chaine de caractères
Prénom : chaine de caractères
Mur : pile de messages
Amis : LS1 de membres
Demandes : file de membres
PagesAimé : LS1 de pages
PagesCrées: LS1 de pages
GroupesCrées : LS1 de groupes
GroupesRejoints: LS1 de groupes
Suggestion: LS1 de membres // les membres suggérés par le réseau

Finenreg

Page=
enreg
date : Date
nomPage: chaine de caractéres
Genre : chaine de caractère
Créateur : membre
Admins : tableau de membres // les membres qui gerent la page
Likers :LS1 de membres // les membres qui aiment la page.
Finenreg

Goupe=
enreg
date : Date
nomGroupe: chaine de caractéres
Genre : chaine de caractère
Créateur : membre
Admins : tableau de membres
Membres_groupe : LS1 de membres
Finenreg

Message= //un enregistrement qui représente un message empilé dans la pile mur de membre
enreg
Contenu : chaine de caractères

Auteur : membre
DateCréation : Date
Finenreg

Graphe=
enreg
sommet: tableau[1..N] de membres
amitié: tableau[1..N, 1..N] d entiers.
finenreg

2) Présentation des méthodes utilisées:

envoyer_amitié(membre a, membre b)
// insérer la personne a dans la file de demandes de b
début
enfiler(b.demandes,a)
fin

accepter_amitié(membre a)
// defiler la file de demandes et insérer la personne accepté

var p :membre
choix :entier
tant que(a.demandes.file_vide=faux) faire
p=defiler(a.demandes)
écrire(‘’taper 1 si vous acceptez l’amitié de ‘’ p.nom p.prenom ‘’ taper 0 si vous refusez’’)
lire(choix)
si(choix=1)
insérer_tete(a.amis,p)
sinon
écrire(‘’la personne’’ p.nom p.prenom ‘’ est refusé’’)
fsi
fin

CréerPage(membre a; nom, genre: chaine de caracteres; date: Date)
// creer une page et l ajouter à la liste de pagescrees du membre a et affecter le membre a au créateur de la page
var page: Page
debut
page.nomPage:=nom
page.Genre:=genre
page.date:=date
page.Créateur:=a
insere_tete(a.PagesCrées,page)
fin

CréerGroupe(membre a; nom,genre:chaine de caracteres, date: Date)
// creer un groupe et l ajouter à la liste de groupescrees du membre a et affecter le membre a au créateur du groupe
var groupe: Groupe

debut
groupe.nomGroupe:=nom
groupe.Genre:=genre
groupe.date:=date
groupe.Créateur:=a
insere_tete(a.GroupesCrées,groupe)
fin

aimerPage(membre a, page b)
// inserer la personne a dans la liste de likers de b et ajouter la page b aux pages_aimés de membre a

début
insére_tete(b.likers,a)
insére_tete(a.PagesAimés,b)
fin

RejoindreGroupe(membre a, groupe g)
// inserer la personne a dans la liste de membres_groupe de g et ajouter g aux groupesRejoint de a
debut
insere_tete(g.membres_groupe, a)
insere_tete(a.groupesRejoints, g);
fin.

poster_message(a : membre, b : membre, contenu : chaine de caractère)
// il faut verifier que a et b sont amis. a poste un message qui sera empilé dans la pile mur de b.
Var msg : message
Début
msg.contenu=contenu
msg.auteur=b
msg.date=dateSystème
empiler(mur,message)
fin.

remplir graphe(G: graphe, N:entier)
var i,j: entier
debut
pour i de 1 à N faire
pour j de 1 à N faire
si amis(sommet[i],sommet[j]) alors
amitié[i,j]=1
sinon
amitié[i,j]=0
finsi
fin pour
fin pour
fin

//passons maintenant aux methode suggérés par le réseau et basé sur le graphe

suggestion_amis:
// cette methode consiste à parcourir le graphe afin de trouver un membre y qui n est pas ami avec un membre a donné. ensuite calculer le nombre d amis en commun entre le membre y et le membre a. Si ce nombre est >0, recupérer le membre associé à partir de tableau de sommet grace à son indice et l ajouter à la liste de suggestion du membre a.

suggestion_amis(g : Graphe,a :membre ,N entier) :membre
Var
i,j,n :entier
Début

Pour i de 1 à N faire
n:=0
Si(a.id != i) alors
Si g.amitié[a.id,i]=0 alors
Pour j de 1 à N faire
Si (g.amitié[j,i]=1) et (g.amitié[j,a.id]=1) alors // j est ami en commun avec a et avec i
n :=n+1
Fsi
Fin pour
finsi
finsi
si n>0 alors
insere_tete(Suggestion,sommet[i])
finsi
Fin pour
fin

suggestion_courtChemin
// cette methode consiste à parcourir le graphe pour trouver des membres qui ne sont pas amis avec le membre a donné mais qui sont amis à ses amis.

suggestion_courtChemin(g : Graphe,a :membre,N : entier) :LS1
var
j,k :entier

début
pour j de 1 à N faire
si G.amitié[[a.id,j]=1 alors
pour k de 1 à n faire
si (g.amitié[k,j]=1 et g.amitié[k,a.id]=0) alors // k est ami avec j mais qui n'est pas ami avec a
insere_tete(a.Suggestion,sommet[k]) // sommet(k) donne le membre d indice k
fin si
fin pour
fin pour
fin

suggestion_amis_PagesCommunes:
//cette methode consiste à parcourir le graphe et calculer le nombre de pages en communs qu avait le membre a avec les autres membres qui ne sont pas amis avec lui. si le nombre est supérieur strictement à 0, récupérer ce membre à partir du

tableau sommeil grace à son indice et l inserer à liste de suggestion de membre a.

suggestion_amis_PagesCommunes(g : Graphe,a :membre,N : entier) :LS1
Var
i,k,n :entier

Début

Pour i de 1 à N faire
n:=0
Si(a.id != i) alors
Si (g.amitié[a.id,i]=0) alors
si contenir(g.sommet[i].PageAimés,a.PageAimés) alors
// s il y a au moins une page en commun entre PagesAimés de membre a et les pages aimés du membre d indice i
n :=n+1
Fsi
finsi
finsi
si n>0 alors
insere_tete(Suggestion,sommet[i])
finsi
Fin pour
fin

methode de recherche en utilisant le backtracking
// on va commencer la recherche d un membre à partir du sommet d indice i
Recherche(g :Graphe, nom : chaine de caractère, prenom : chaine de caractère, N :entier,i :entier) : membre
var q: booléen

début

k :=0
q :=faux
répéter
k :=k+1
si (g.sommet[i].nom=nom et g.sommet[i].prenom=prenom) alors
Recherche :=g.sommet[i] // la personne recherchée est trouvée.
Sinon si(amitié[i,k]=1) alors // on teste si le sommet k est adjacent à i cad le candidat est acceptable
Si(k<N) alors
Recherche(g,nom,prenom,N,k+1)
Sinon
q :=vrai
fsi
fsi
jusqu'à (i=N ou k=N ou q)

fin