Compte Rendu Projet Algorithmique 3-Graphes
Maxime Dourlen et Théo Finot

Lors du semestre 5 de l'année scolaire 2021/2022 les étudiants en Licence 3 Informatique et Informatique parcours Science des Données devaient rendre un projet pour la matière Algorithmique 3-Graphes. Pour ce projet nous devions réaliser une étude (simplifiée) d'un réseau social (simplifié) de type Facebook. Pour mettre en place cette étude, nous avons utilisés quelques notions de théorie des graphes qui nous ont permis de représenter ces réseaux et de les analyser efficacement. Ce projet a été réalisé par Maxime Dourlen étudiant en Informatique TD3 TP5 et Théo Finot étudiant en Informatique parcours Science des Données TD4 TP7. Nous avons décider de faire ce projet ensemble car nous avions déjà réalisé des projets informatique lors de nos années précédentes ce qui nous a permis, lors de ce projet, d'être plus productif.

Nous voulions premièrement implanter notre code avec le langage C, car nous avions plus d'expérience avec ce langage notamment grâce aux anciens projets que nous avions du implanter en C. Cependant nous avons finalement décidé d'utiliser la langage orienté objet JAVA. Ce langage nous a paru plus simple car nous avons ainsi pu utiliser les bibliothèques JAVA et ensuite nous avons pu aussi faire une interface homme machine en JAVA.

Voici la modélisation de notre graphe, les utilisateurs de ce réseau peuvent créer des comptes de deux types, soit de type Utilisateur, soit de type Page. Les comptes sont représenté par des sommets dans notre graphe. Dans le sujet il nous ai informé qu'un compte de type Utilisateur peut suivre d'autres comptes de type Utilisateur et que les comptes de type Utilisateur peut aimer des comptes de type Page. Cependant les comptes de type Page ne peuvent ni suivre un compte de type Utilisateur ni aimer un compte de type Page. Nous avons donc les arcs qui symbolisent la relation x "suit" y, avec x et y des comptes Utilisateur dans le graphe et aussi la relation x aime y , avec x un compte Utilisateur et y un compte Page dans le graphe.

De plus les comptes Utilisateur sont identifiés par un nom et ont des informations supplémentaires comme le prénom et l'âge. Les comptes Page sont également identifiés par un nom et ont une liste d'administrateurs de type Utilisateur.

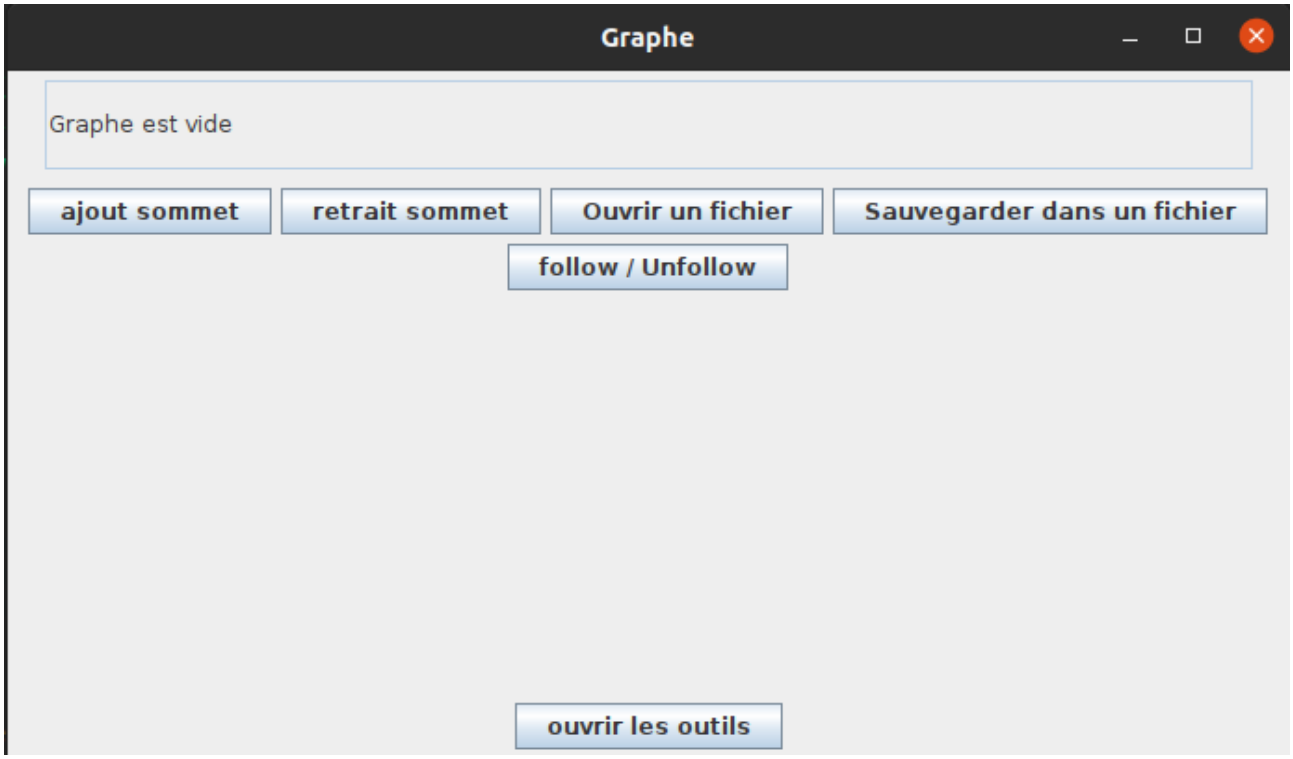
De ce fait comme demandé nous avons, afin de gérer le graphe, créer une classe abstraite Sommet (Sommet.java) , défini un constructeur, des accesseurs publics et chaque Sommet contiens la liste de ses voisins sortants. Nous avons aussi implanter les classes Utilisateur (Utilisateur.java) et Page (Page.java) et ces deux classes héritent de la classe Sommet. Et enfin nous avons coder la classe Graphe qui représente un graphe sous forme de liste d'adjacence et donc contiens un ensemble de Sommet.

Nous allons montrer ici comment utiliser l'interface graphique et comment elle fonctionne a travers des exemples. Tout d'abord voici la liste d'adjacence de notre graphe pour les exemples qui suivent :

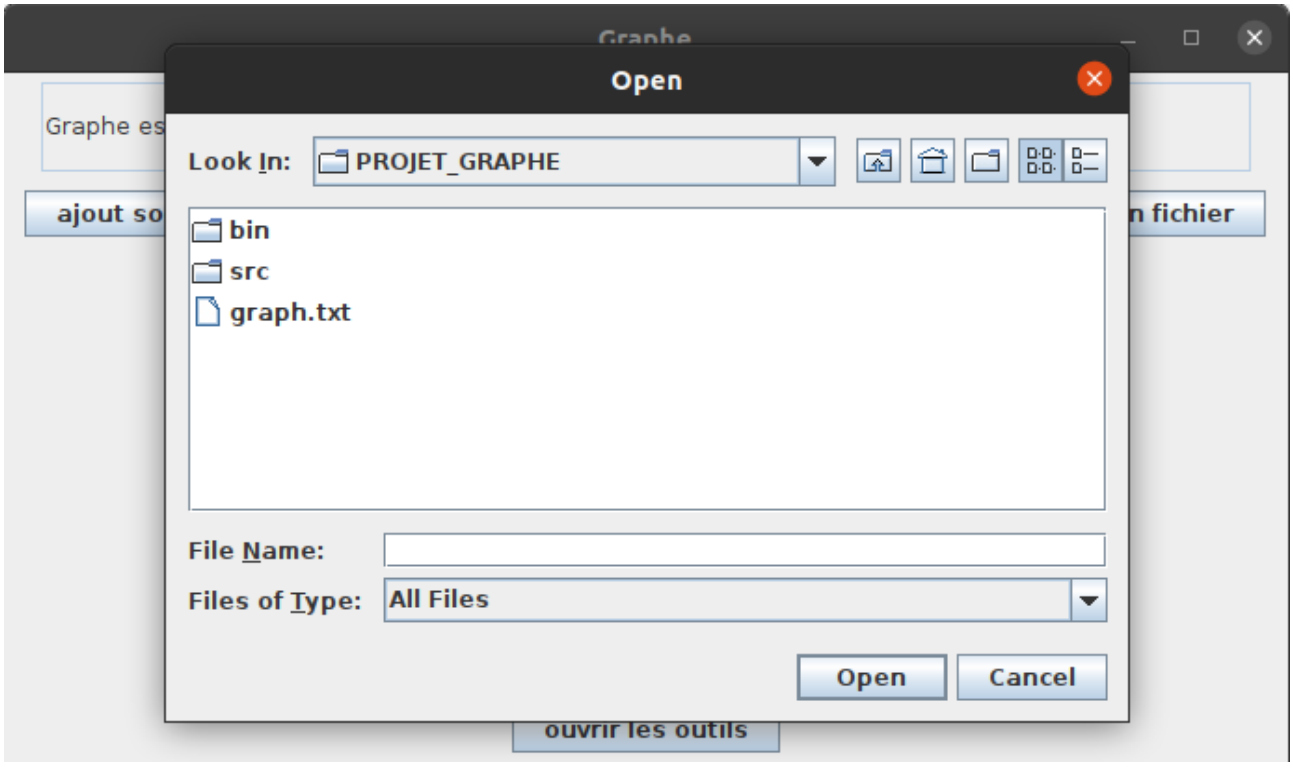
<user, 1>, <user, 2>, <page, 3>, <user, 4>, (1, 2),(2, 3),(2, 4),(4, 2)

Comme vous pouvez le remarquer nous avons décider de représenter notre liste d'adjacence telle qu'entre les symboles : '<' et '>' soit représentés les sommets du graphe avec premièrement le type du sommet et ensuite son nom (séparés par une virgule), et entre les symboles '(' et ')' soit représentés les arcs du graphes. Donc d'après la liste d'adjacence de notre exemple on peut comprendre que notre graphe est composé de 4 sommets, 3 de type Utilisateur (les sommets '1', '2' et '4') et un sommet de type Page (le sommet '3'). Ensuite nous pouvons remarquer que notre

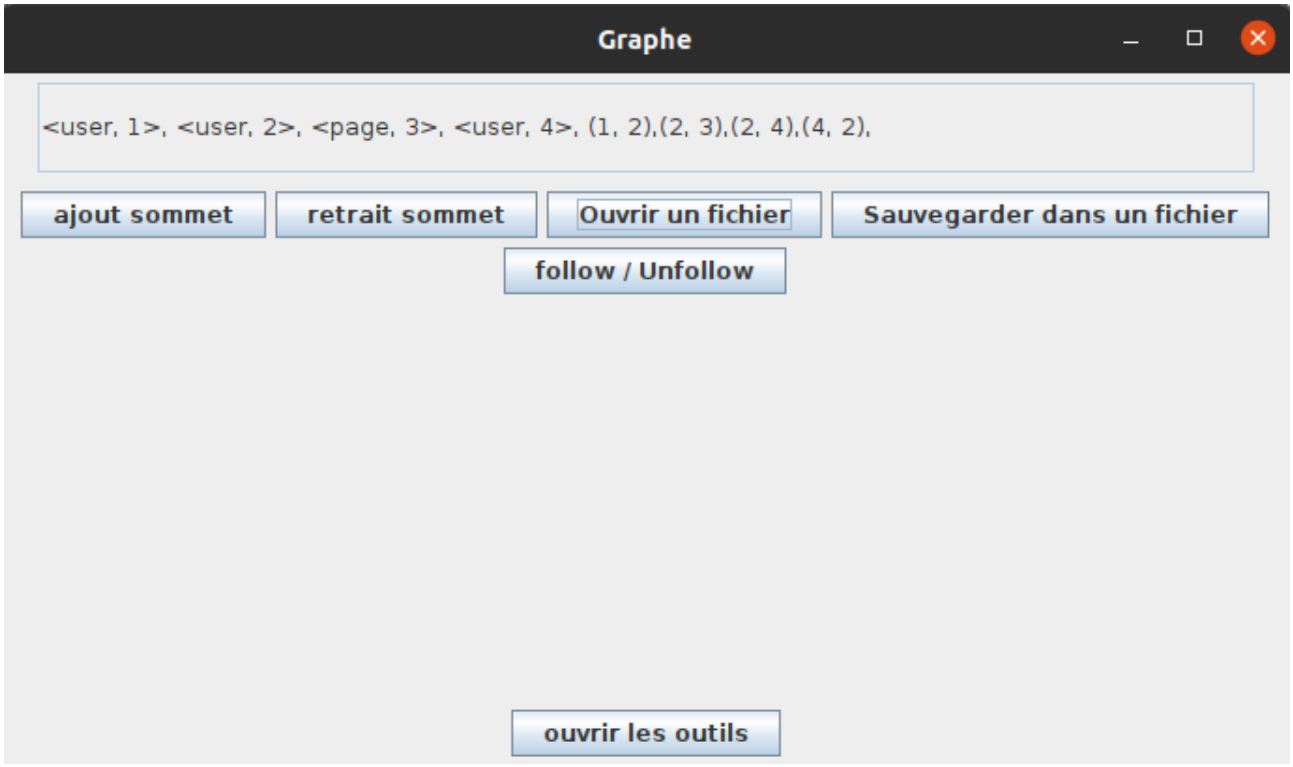
graphe est aussi composé de 4 arcs, un arc de ‘1’ vers ‘2’, un arc de ‘2’ vers ‘3’, un arc de ‘2’ vers ‘4’ et un arc de ‘4’ vers ‘2’.
Voici ici une première visualisation de la fenêtre principale de notre interface graphique :



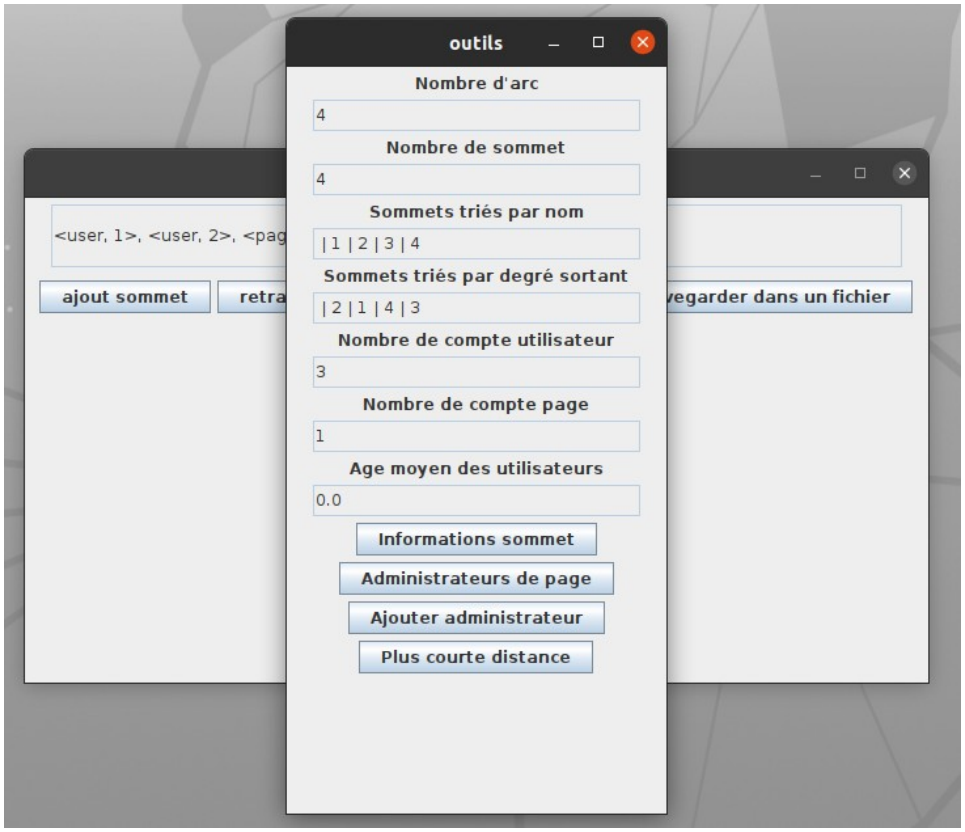
Nous remarquons alors que pour l’instant notre graphe est vide. Afin d’avoir le graphe correspondant à notre liste d’adjacence nous allons l’importer depuis un fichier texte. Pour cela il nous suffit de cliquer sur le bouton : ‘Ouvrir un fichier’ et ensuite de choisir le fichier voulu. Dans notre cas cela donne :



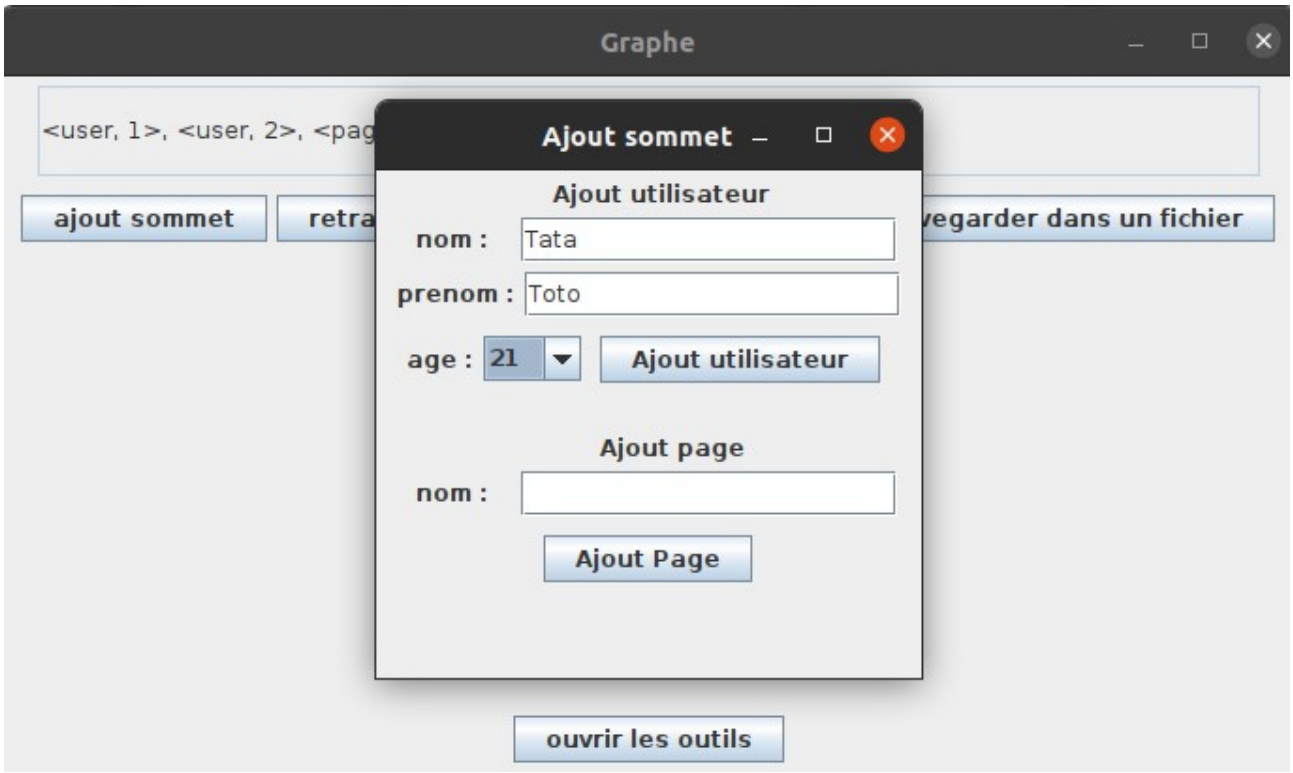
Après avoir ouvert le fichier ‘graph.txt’ qui comporte notre liste d’adjacence, nous nous apercevons que sur notre fenêtre principale s’affiche notre liste :



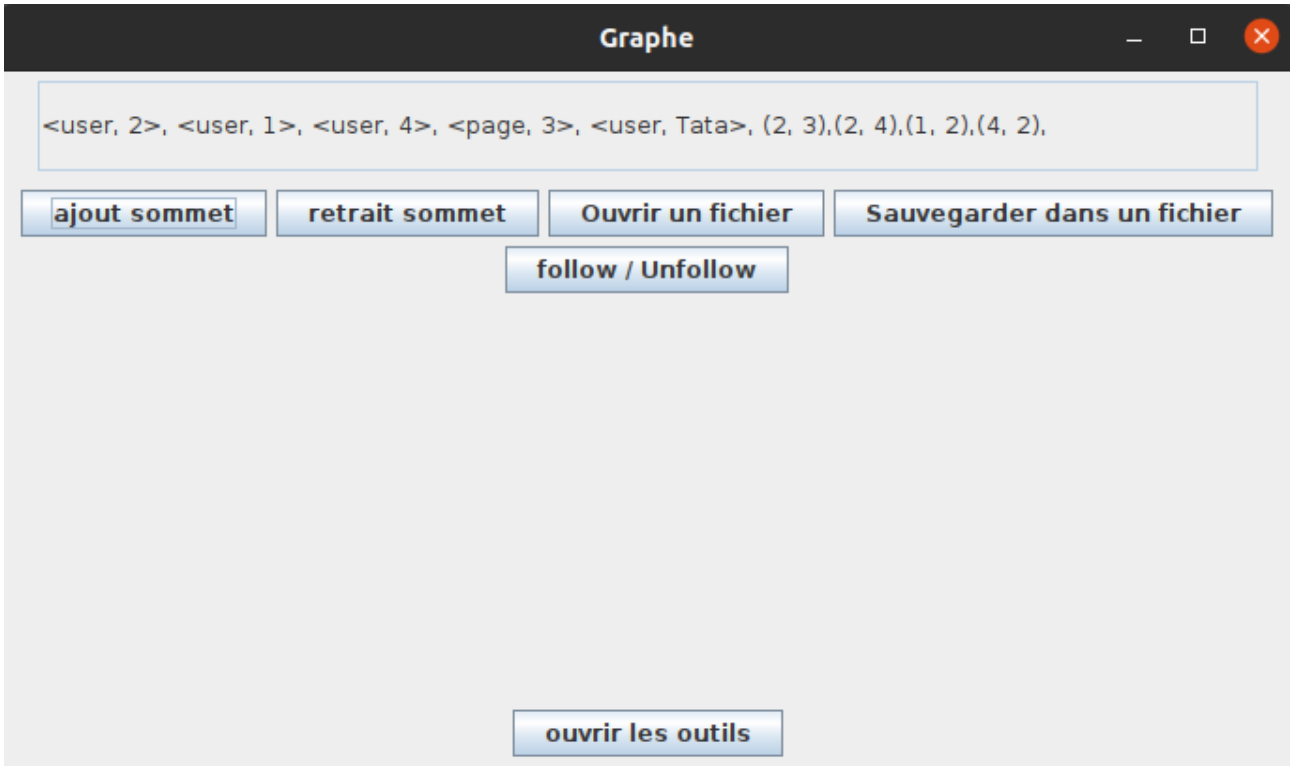
Afin de connaître le nombre de sommets et d’arcs, d’obtenir l’ensemble des sommets trié par nom, d’obtenir l’ensemble des sommets triées par degré sortant, d’obtenir le nombre de compte de type Utilisateur et de Page et d’obtenir l’âge moyen des utilisateurs, il est possible de faire afficher toutes ces informations en cliquant simplement sur le bouton : ‘ouvrir les outils’
Ce qui nous donne pour notre exemple :



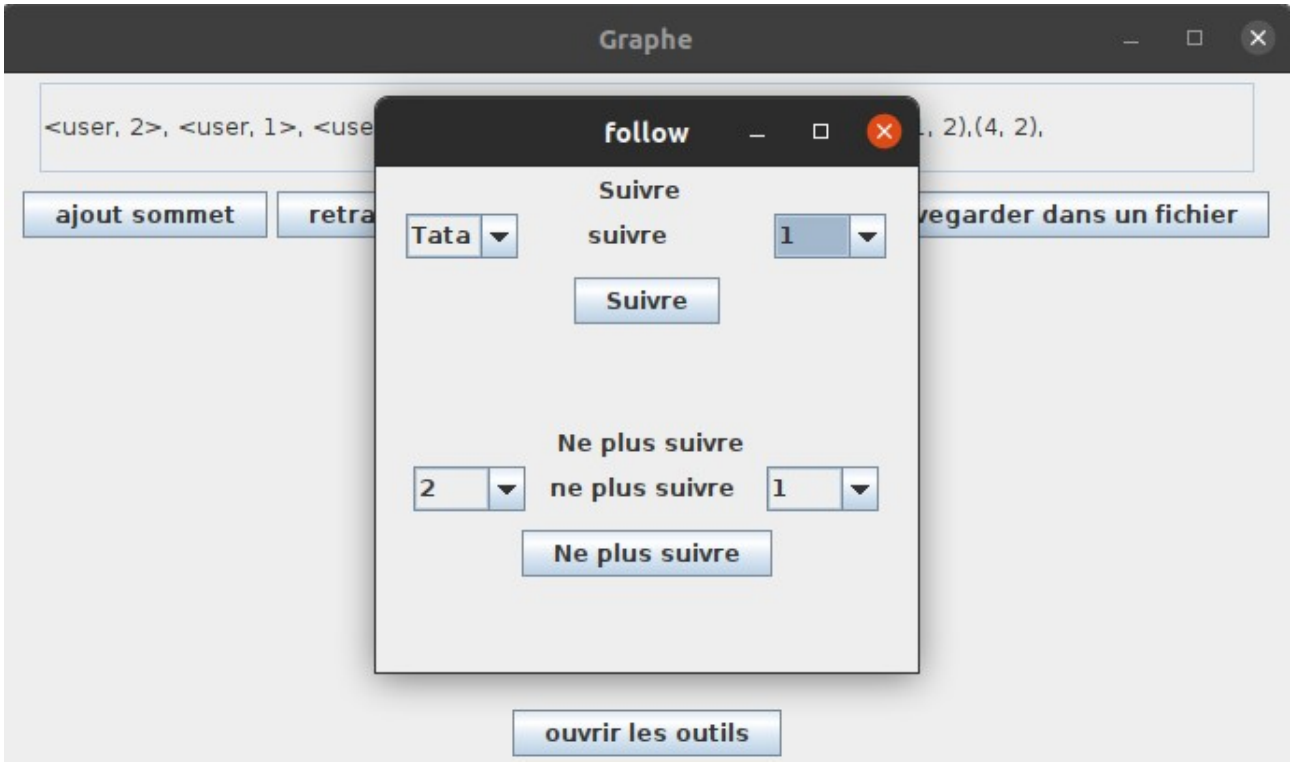
Il est aussi possible d'ajouter un nouveau sommet et de lui faire suivre des autres compte de types Utilisateur ou bien de Page. Par exemple ajoutons un sommet de type Utilisateur avec pour information, Toto comme prénom, Tata comme nom et pour âge 21 ans. Pour cela il suffit de cliquer sur la bouton : 'ajout sommet' dans la fenêtre principale. Ce qui nous donne :



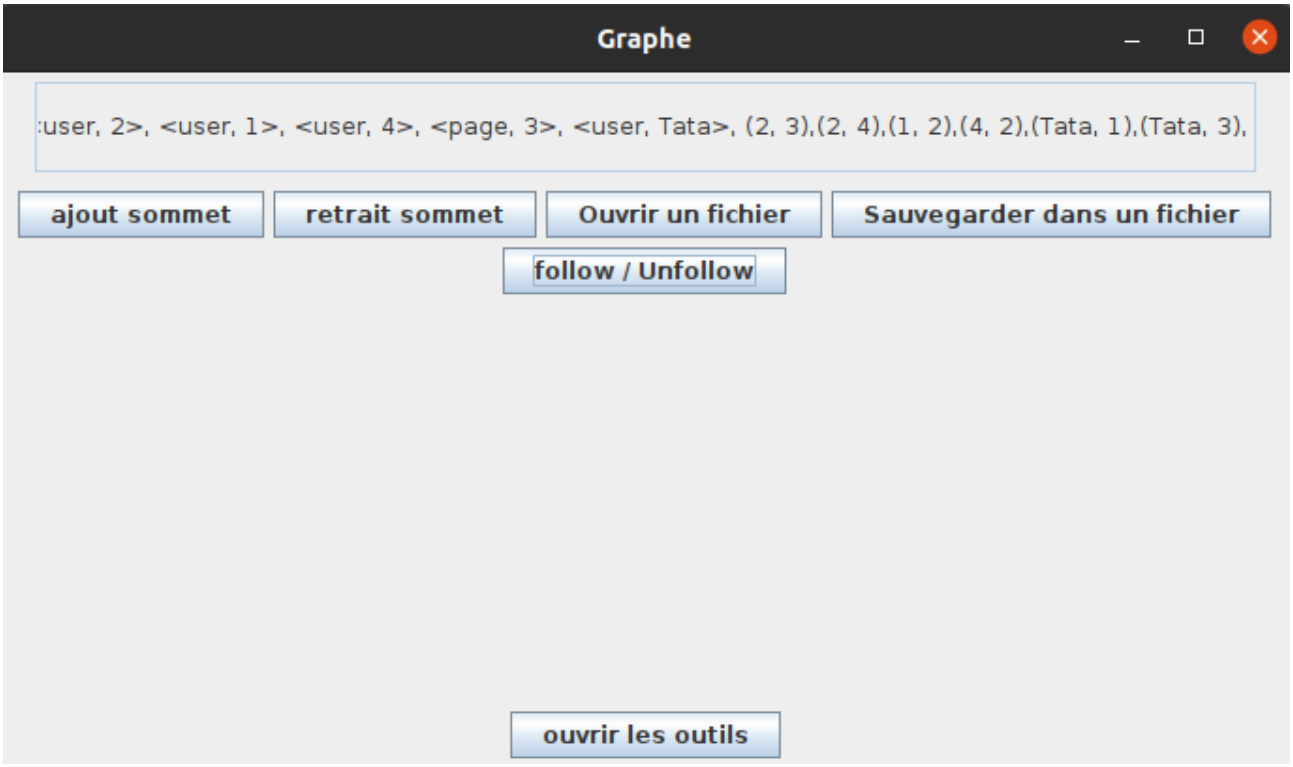
Nous remarquons alors que ce sommet créée s'ajoute dans l'affichage de notre liste d'adjacence :



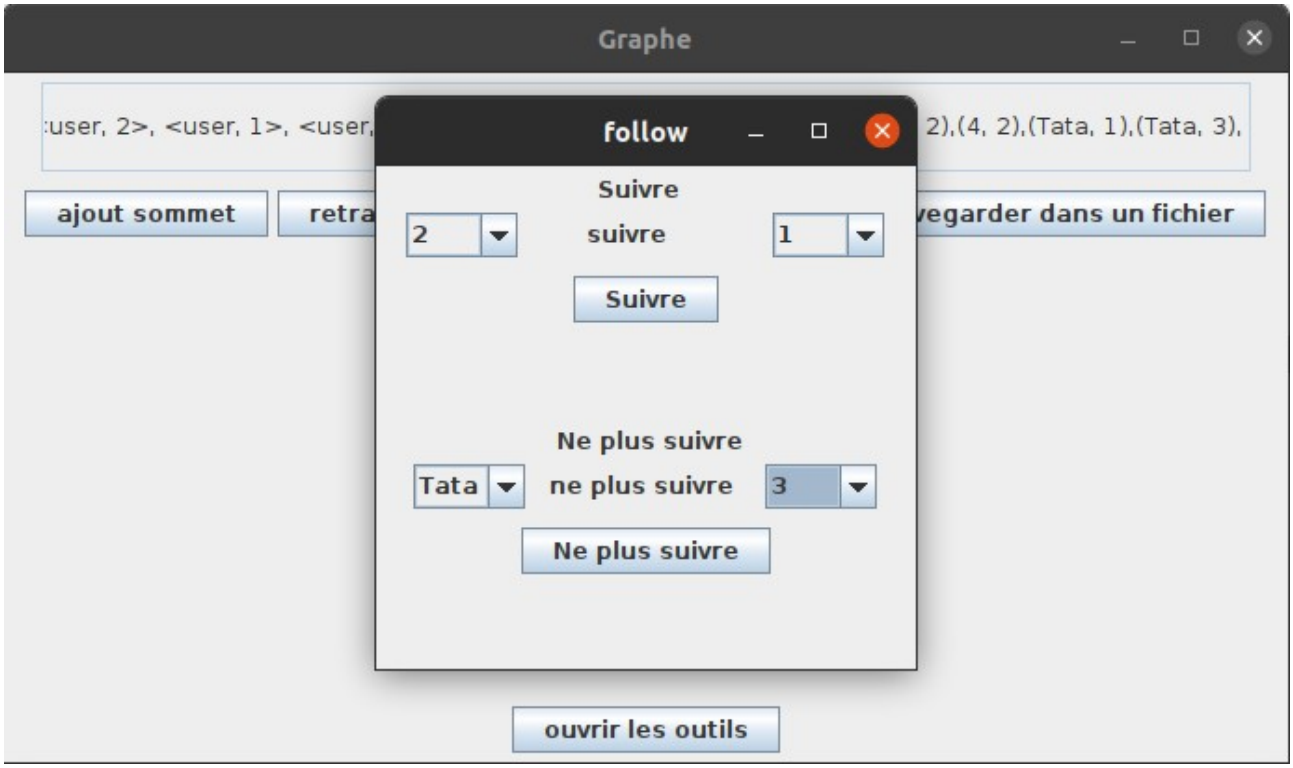
Maintenant nous allons faire suivre ce nouveau sommet ‘Tata’ au sommet Utilisateur ‘1’ et au sommet de type Page ‘3’. Pour ce faire il faut cliquer sur le bouton : ‘follow / Unfollow’ qui est présent dans la fenêtre principale. Sélectionner le sommet qui va suivre un autre sommet, puis le sommet qui va se faire suivre et enfin cliquer sur le bouton :’suivre’.
Voici les images de l’exemple :



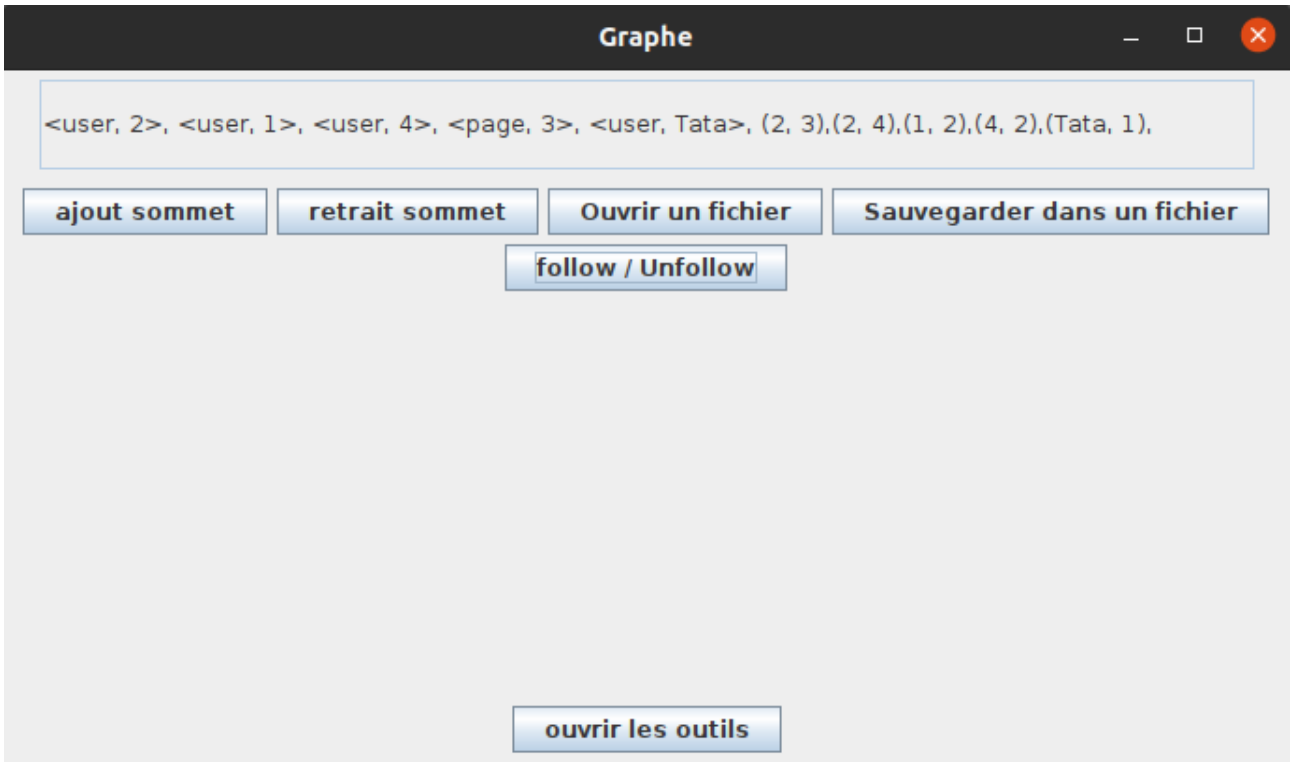
Ce qui nous donnes notre nouvelle liste d’adjacence :



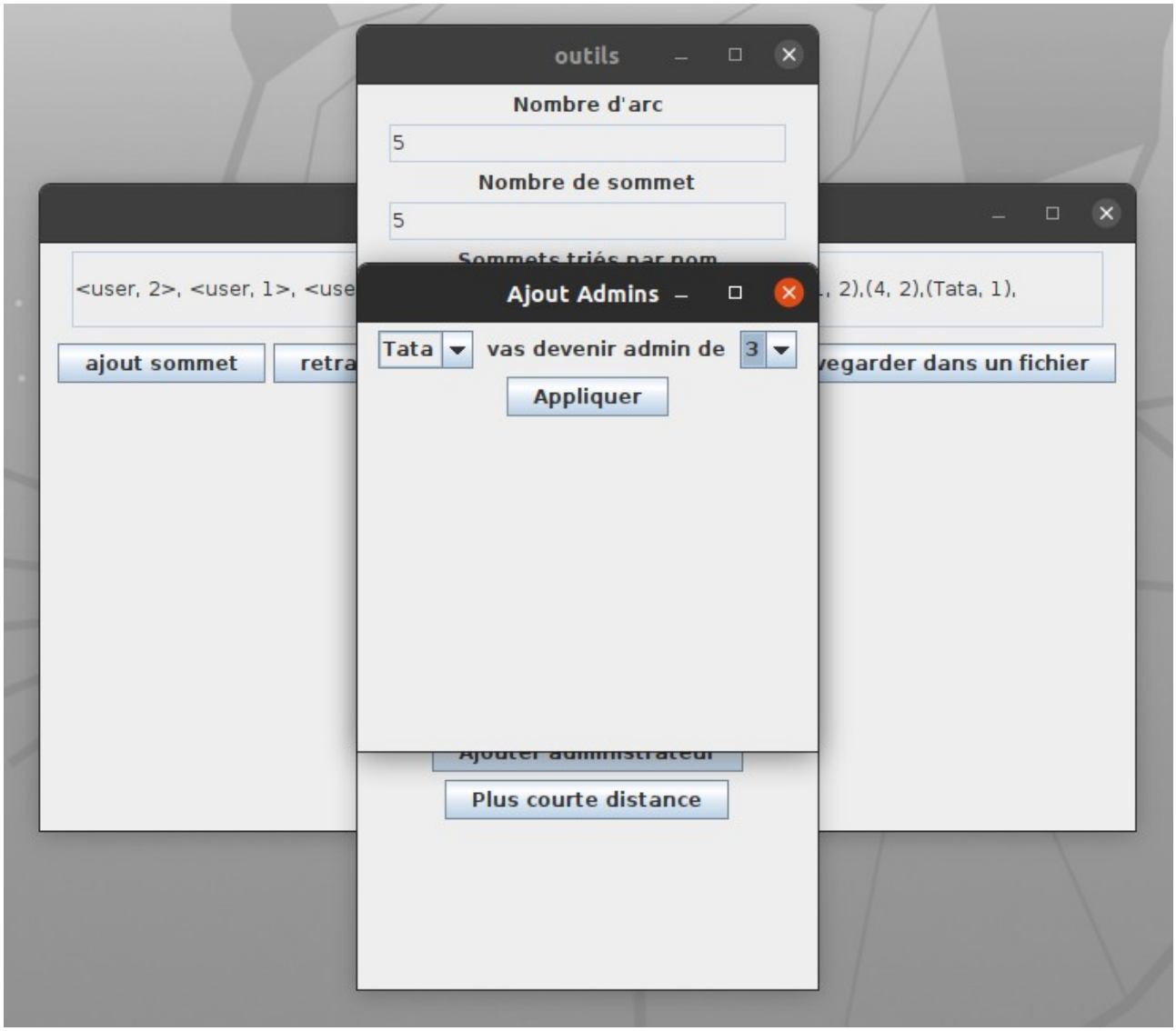
Pour notre exemple nous allons maintenant supprimer l'arc entre le sommet Utilisateur 'Tata' vers le sommets Page '3'. Il faut donc encore une fois appuyer sur le bouton : 'Follow / Unfollow' puis ensuite sélectionner premièrement le sommet 'Tata' puis le sommet '3' et clquer sur le bouton : 'ne plus suivre'.



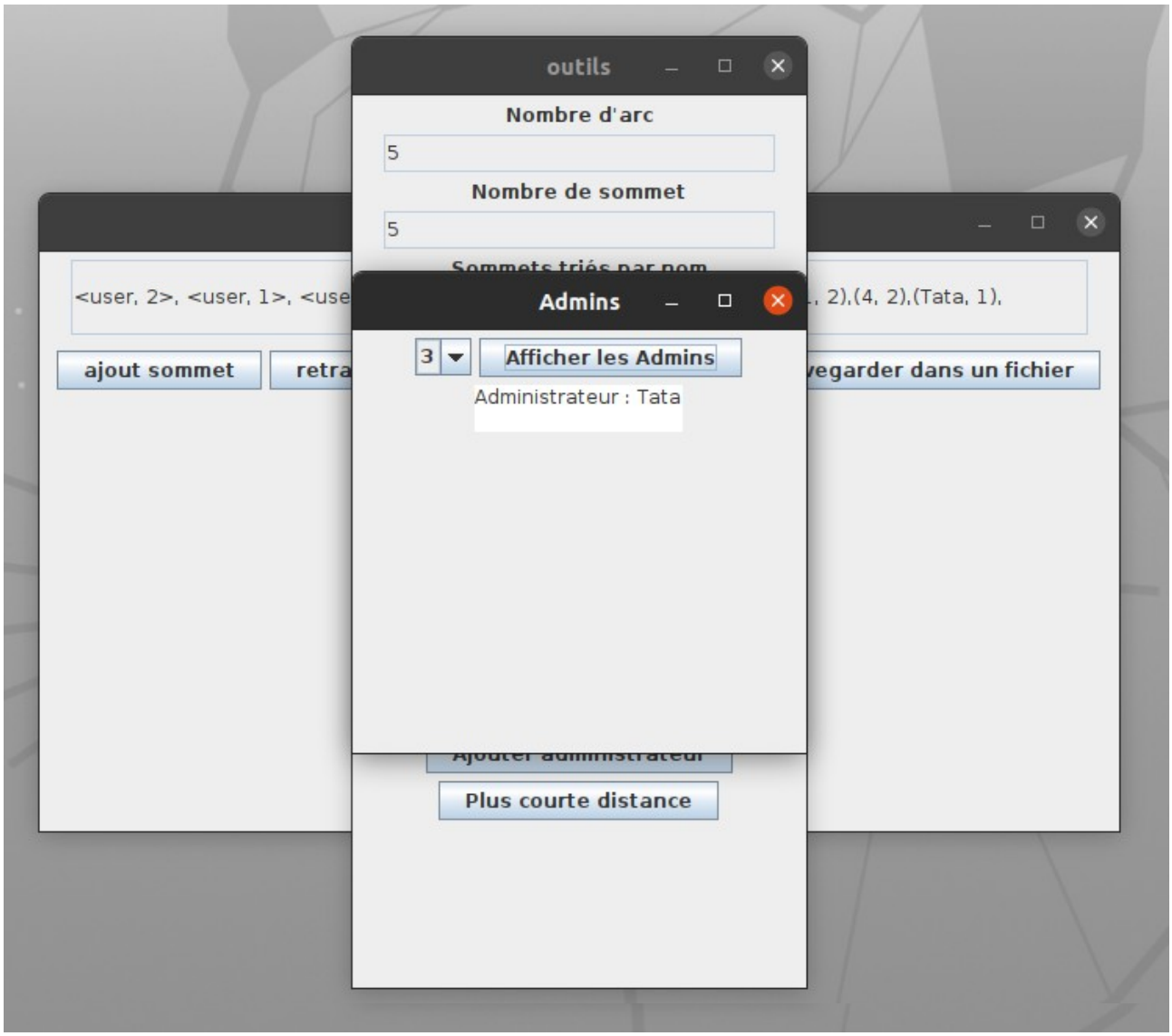
Voici notre liste d'adjacence mise a jour dans la fenêtre principale :



Nous voulons maintenant définir le sommet utilisateur ‘Tata’ comme administrateur de la Page ‘3’. Pour cela il faut cliquer sur le bouton : ‘ouvrir les outils’ puis cliquer sur le bouton : ‘Ajouter administrateur’. Sélectionner le sommet utilisateur qui deviendra l’administrateur de la Page, ensuite sélectionner la Page correspondante et enfin cliquer sur appliquer.

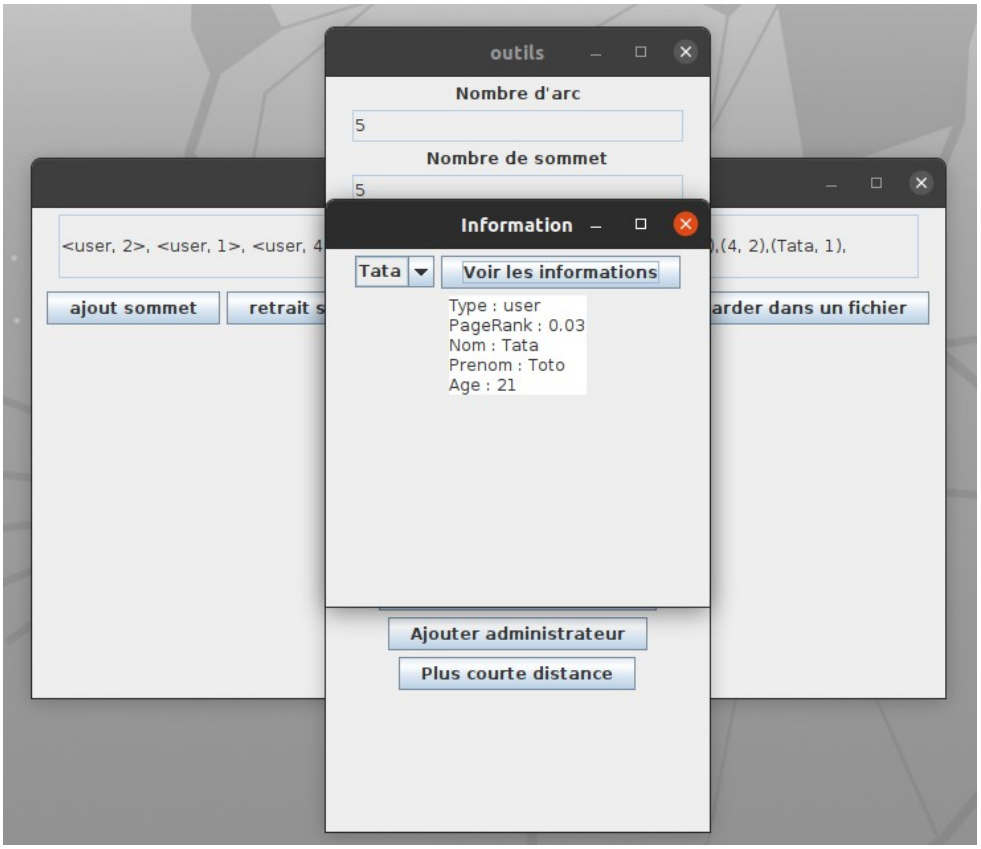


Pour afficher les administrateurs d’une page il faut cliquer sur le bouton : ‘ouvrir les outils’ puis cliquer sur le bouton : ‘Administrateur de page’, ensuite sélectionner la page voulue enfin cliquer sur le bouton : ‘Afficher les admins’. Dans notre exemple cela donne :

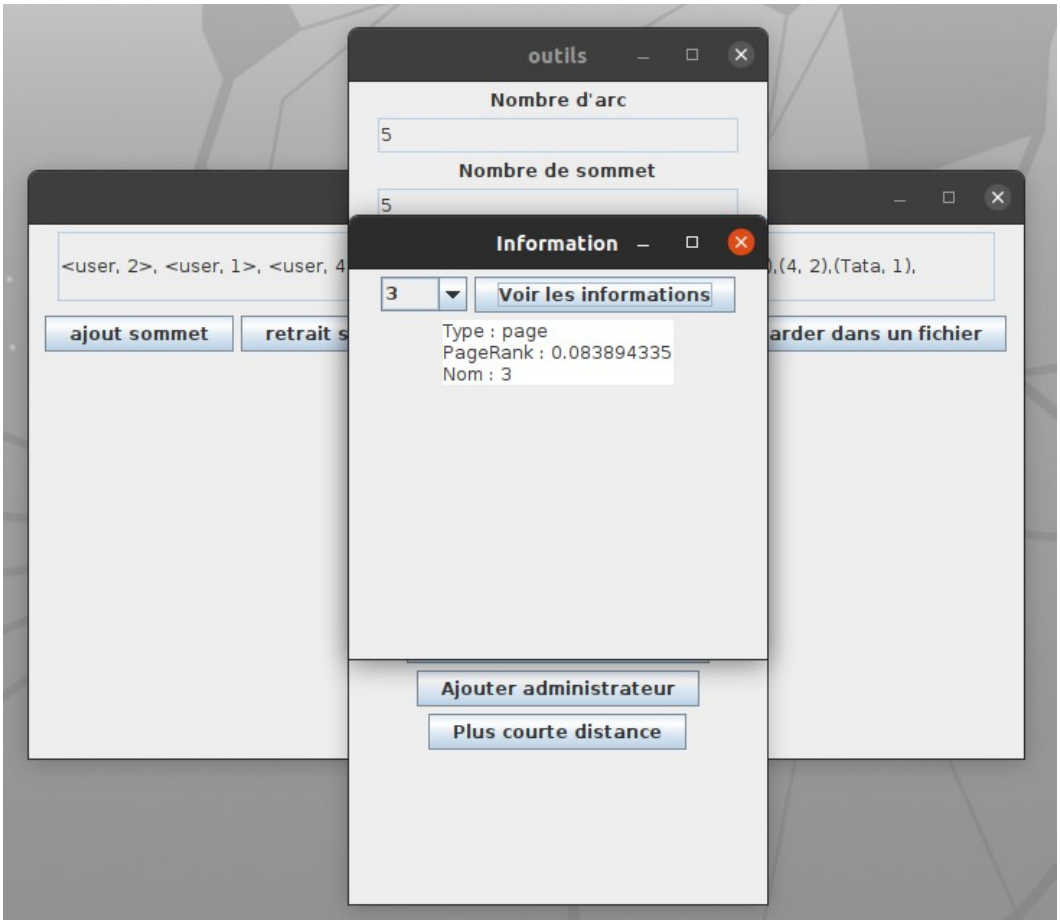


Afin d’afficher les informations d’un sommet telle que si ce sommets est de type Utilisateurs alors les informations affichées seront : son type, son PageRank, son nom, son prénom, son âge et si ce sommet est de type Page alors les informations affichées seront : son type, son PageRank et son nom. Pour obtenir ces information il faut cliquer sur le bouton : ‘ouvrir les outils’ puis cliquer sur le bouton : ‘Information sommet’, ensuite sélectionner le sommet voulue et enfin cliquer sur le bouton : ‘Voir les informations ’.

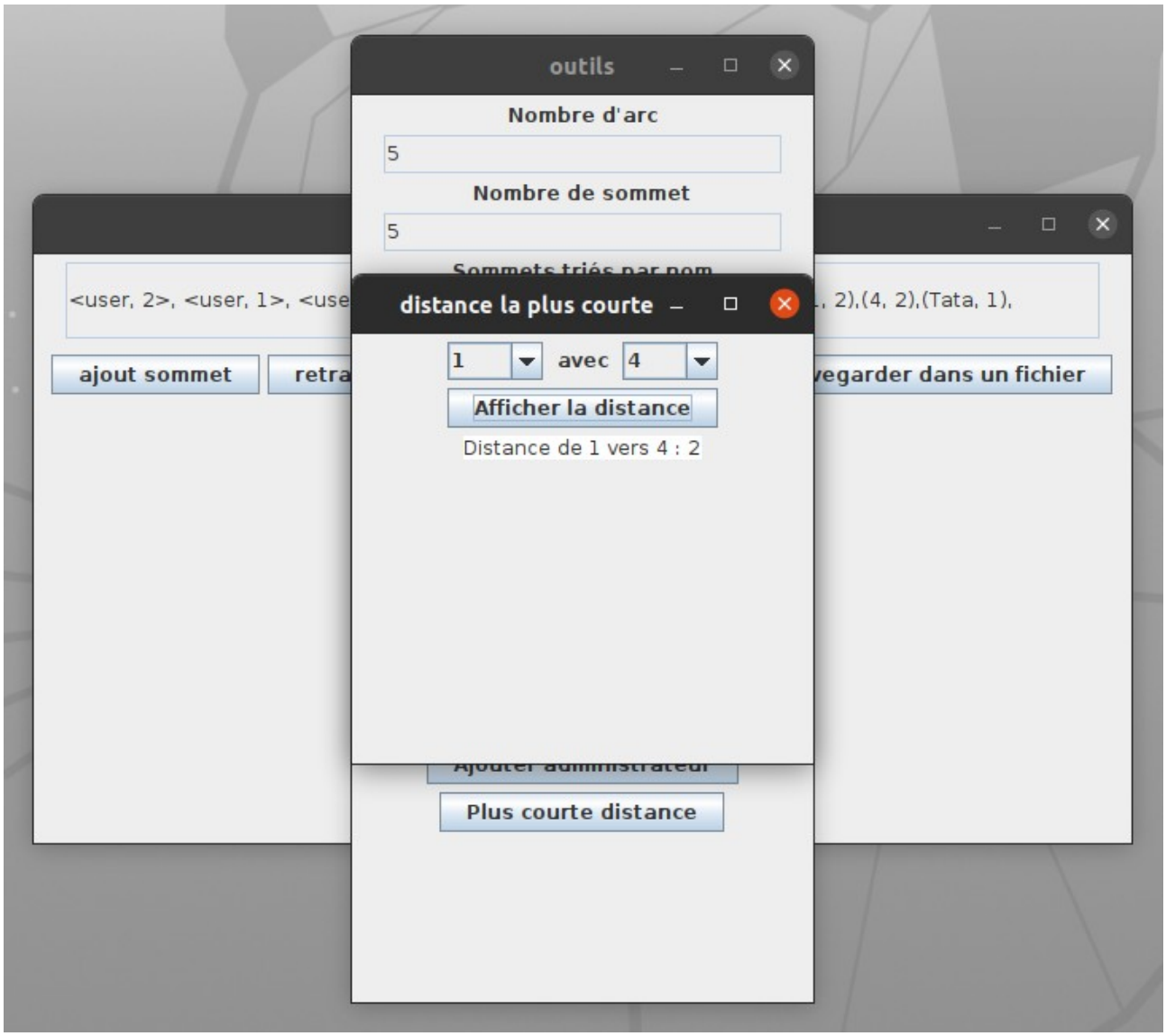
Dans notre exemple le sommet Utilisateur ‘Tata’ nous affiche :



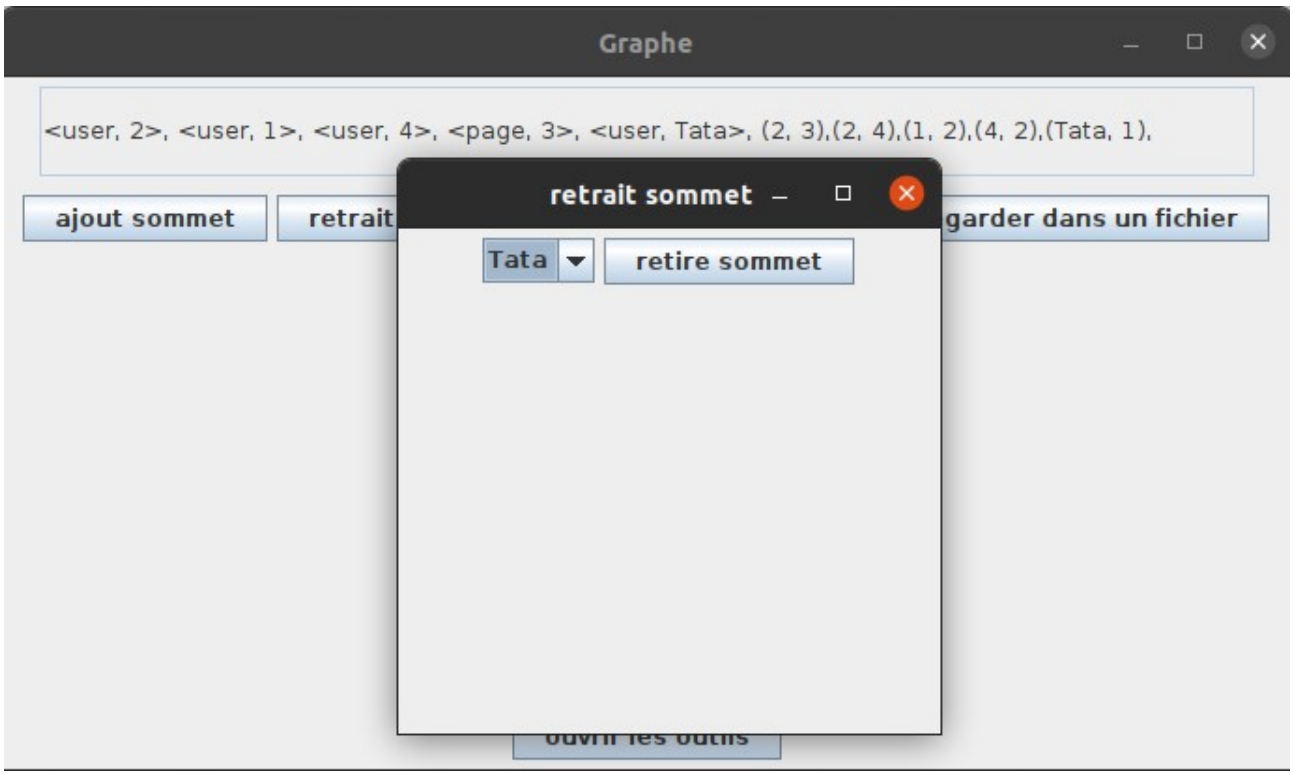
Et pour le sommet de type Page ‘3’ :



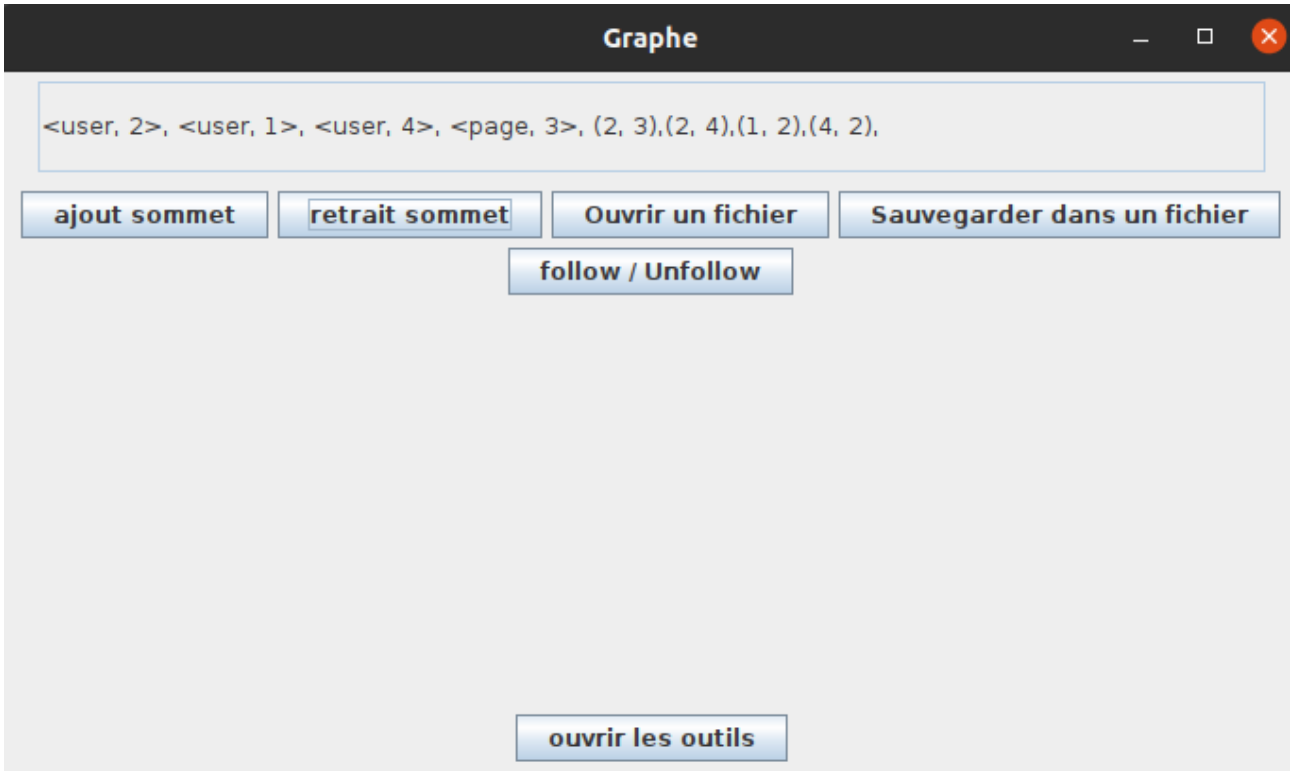
Il est aussi possible pour cette interface graphique d'afficher la plus courte distance entre deux sommets. Pour cela il faut cliquer sur le bouton : 'ouvrir les outils' puis cliquer sur le bouton : 'Plus courte distance', ensuite sélectionner le premier sommet voulue, puis le second sommet et enfin cliquer sur le bouton : 'Afficher la distance'.
Ce qui nous donne dans notre exemple entre le sommet 1 et 4 :



Maintenant supprimons le sommet Utilisateur ‘Tata’, pour cela dans la fenêtre principale cliquer sur le bouton : ‘retrait sommet’ puis sélectionner le sommet a supprimer et cliquer sur le bouton :’retire sommet’. Pour notre exemple nous avons donc sélectionner le sommet ‘Tata’ afin de le supprimer.



Ensuite voici notre liste d'adjacence mise a jour dans la fenêtre principale car le sommet 'Tata' a était supprimé mais aussi les arcs qui le comportaient :



Pour enregistrer une liste adjacente dans un fichier texte, il suffit de cliquer sur le bouton : 'Sauvegarder dans un fichier'.

Voici une image du code de l'implantation de l'algorithme du plus cours chemin que nous avons réalisé. Cet algorithme a pour complexité 2n.

```
public Map shortest(Sommet s){
    //graph doit pas etre vide !!!!!!!!!!!!!!!!!!!!!!!!
    Map dist = new HashMap<Sommet, Integer>();
    List v = getSommets();
    for(int n = 0; n < v.size(); n++){
        Sommet current = (Sommet)v.get(n);
        if(current != s){
            dist.put(current, new Integer(10000000));
        } else {
            dist.put(current, new Integer(0));
        }
    }
    List p = new ArrayList(v);
    while(p.size() > 0){
        Sommet u = plusCourt(dist, p);
        p.remove(u);
        List f = u.getFollower();
        for(int i = 0; i < f.size(); i++){
            Sommet vb = (Sommet)f.get(i);
            int alt = ((Integer)dist.get(u)).intValue() + 1;
            if(alt <= ((Integer)dist.get(vb)).intValue()){
                dist.put(vb, new Integer(alt));
            }
        }
    }
    return dist;
}
```

Et ici le code de l'implantation de l'algorithme PageRank. Nous faisons appelle a la méthode 'calPR' qui permet de calculer la somme des PR de chaque sommets du graphe. Cet algorithme est aussi de complexité 2n

```
public Map pageRank(){
    Map pr = new HashMap<Sommet, Float>();
    List v = getSommets();
    for(int n = 0; n < v.size(); n++){
        pr.put((Sommet)v.get(n), new Float(1));
    }
    int i = 0;
    while(i < 100){
        for(int n = 0; n < v.size(); n++){
            float pagerank = (float)(0.15/v.size());
            pagerank += calPR(((Sommet)v.get(n)).getFollower(), pr);
            pr.put((Sommet)v.get(n), new Float(pagerank));
            i++;
        }
    }
    return pr;
}
private float calPR(List follower, Map pr){
    float result = 0;
    for(int u = 0; u < follower.size(); u++){
        float c = (float)((Float) pr.get(follower.get(u))).floatValue();
        float d = (float)((Utilisateur)follower.get(u)).nbVoisin();
        result += (float)c/d;
    }
    return (float)0.85*result;
}
```

Pour conclure, notre projet implémente tout ce qu'il y a été demandé avec les contraintes demandés. De plus nous dirions que nous n'avons pas rencontrée de grande difficultés lors de la réalisation de ce projet. Néanmoins il nous a fallut beaucoup de réflexion afin d'optimiser au mieux ce projet et nous nous sommes aussi beaucoup aider de la Javadoc pour simplifier les lignes de codes.