

Rapport de Synthèse – Projet C/C++

*Projet C/C++*

Sujet : « Réseaux sociaux : Recherche de composantes fortement connexes »

DREYER Quentin

JAMBET Pierre

NGUYEN Michael

Table des matières

[Introduction 3](#_Toc261728815)

[1. Sujet 3](#_Toc261728816)

[2. Organisation du travail 3](#_Toc261728817)

[Travail réalisé 4](#_Toc261728818)

[1. Outils utilisés 4](#_Toc261728819)

[2. Représentation 5](#_Toc261728820)

[3. Les classes 6](#_Toc261728821)

[4. Les algorithmes 7](#_Toc261728822)

[Conclusion 8](#_Toc261728823)

[Annexes 9](#_Toc261728824)

[1. Remerciements 9](#_Toc261728825)

[2. Bibliographie 9](#_Toc261728826)

[3. Webographie 9](#_Toc261728827)

# **Introduction**

## **Sujet**

## **Organisation du travail**

Le choix du langage fut la première question auquel nous avons du faire face, en effet le projet pouvait être réalisé en C ou C++, ayant tous trois des connaissances en dans ces deux langages nous avons choisi le C++. Ce choix est justifié par coté « Orienté Objet » du C++.

Ce choix peut paraitre surprenant car nous n’avons utilisé pour ce projet qu’une infime partie des possibilités du C++, nous n’avons pas utilisé d’héritages ni de polymorphismes ou autres techniques avancées. La « simple » utilisation des classes ainsi que de la Bibliothèque standard STL (Standard Template Library), nous ont été très utiles. De plus nous avons pu aussi utiliser des fonctions du langage C très utiles : atoi ou fscanf par exemple.

# **Travail réalisé**

## **Outils utilisés**

#### Développement

Le projet a été développé conjointement sur Windows ainsi que sur Gnu/Linux, pour plusieurs raisons. Nous avions tous des machines fonctionnant sous Windows chez nous, cependant, certains outils n’étant disponibles que sur Gnu/Linux, comme Valgrind par exemple et comme les ordinateurs de Polytech’ fonctionnent en grande partie sur Debian, nous avons utilisé les deux systèmes d’exploitation. Ceci nous a permis d’obtenir un code portable.

#### Contrôle de versions

Afin de gérer au mieux le travail de groupe, et pour optimiser le travail collaboratif nous avons choisi d’utiliser un gestionnaire de version. Notre choix s’est porté sur SubVersion(SVN), car l’outil TortoiseSVN pour Windows est extrêmement pratique. Nous avons donc hébergé notre code sur un serveur fourni gratuitement par Google (URL : <http://code.google.com/>).

#### Editeurs

Les outils que nous avons utilisées pour le développement étaient soit un simple éditeur de texte, NotePad++ sur Windows ou Gedit sur Ubuntu, ainsi qu’un IDE tel que CodeBlocks que nous avons retenu pour sa portabilité ainsi que sa gratuité.

#### Tests unitaires

Afin de créer un code robuste et facile a modifier, nous avons choisi d’utiliser un framework permettant de mettre en place des tests unitaires. Nous avons choisi le framework Google Test, pour sa simplicité d’utilisation. Ainsi a chaque modification du code, nous pouvions utiliser les tests pour vérifier que chaque fonction se comportait correctement, indépendamment du reste du programme.

Un test s’écrit simplement de la manière suivante :

*Sur cet exemple, on teste premièrement que la condition cond est vraie et on teste ensuite l’égalité des valeurs de val1 et val2.*

TEST(nom\_categorie\_test, nom\_test) {  
 ASSERT\_TRUE(cond);  
 EXPECT\_EQ(val1, val2);  
}

On a donc pu écrire une succession d’assertions vérifiant les propriétés de nos fonctions.

## **Représentation**

Nous avons choisi de représenter les graphes de deux manières différentes, qui sont les deux façons les plus classiques, la matrice d’adjacence, et la liste d’adjacence. Voici un exemple illustrant ces deux représentations. On notera l’intérêt de la liste en termes de stockage mémoire car on ne stocke pas les sommets « non voisins » d’un sommet donné.

## E:\Mes documents\Bureau\ig3projetc\trunk\doc\Fichier6n-graph2.png

## Figure 1 : Graphe exemple

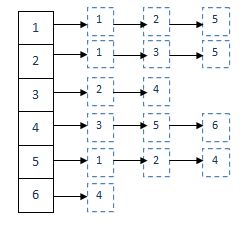
\begin{pmatrix}
1 & 1 & 0 & 0 & 1 & 0\\
1 & 0 & 1 & 0 & 1 & 0\\
0 & 1 & 0 & 1 & 0 & 0\\
0 & 0 & 1 & 0 & 1 & 1\\
1 & 1 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 1 & 0 & 0\\
\end{pmatrix}.

Figure 2 : Matrice d'adjacence

Figure 3 : Liste d'adjacence

## **Les classes**

### Graph

### AdjMat

|  |
| --- |
|  |
|  |

### AdjList

|  |
| --- |
|  |
|  |

### Generator

|  |
| --- |
|  |
|  |

### Summit

|  |
| --- |
|  |
|  |

## **Les algorithmes**

# **Conclusion**

# **Annexes**

## **Remerciements**

Nous tenons à remercier Polytech‘ Montpellier, plus particulièrement l’ensemble de nos professeurs, dont les enseignements nous ont permis de mener a bien ce projet. Cependant nous pensons qu’il est important de remercier aussi les auteurs des outils qui nous ont été indispensables pendant le déroulement du projet.

Merci à :

* Google pour : Google Code, Google Test et Google
* Richard Stallman pour GCC
* Mathieu Triay !

## **Bibliographie**

Support de cours

## **Webographie**

* <http://www.siteduzero.com/>
* <http://www.cplusplus.com/>
* <http://www.developpez.com/>
* et en particulier :
* <http://gl.developpez.com/tutoriel/outil/makefile/>
* <http://fr.wikipedia.org/>
* <http://code.google.com/p/googletest/>