**Devoir #1**

PIF1005

Scott Le Clair

Université du Québec à Trois-Rivières

Hiver 2023

Table des matières

[Introduction 1](#_Toc131962957)

[Les structures des données 2](#_Toc131962958)

[Les algorithmes 3](#_Toc131962959)

## Introduction

Ce document contient des descriptions et des explications pour les choix de conception prises pour le programme Java de devoir 1, qui évalue des graphes simples. Ce document présentera au lecteur :

1. Les structures des données
2. Les algorithmes
3. Les inputs
4. Les résultats des traitements
5. La comparaison entre une matrice d’adjacence et une matrice d’incidence pour représenter un graphe

Tout pour mettre en contexte le programme relié à ce document.

## Les structures des données

Pour commencer, le programme fait le traitement d’un graphe à l’aide d’un des deux classes : un pour les graphes représentés par un matrice d’adjacence, et l’autre pour les graphes représentés par un matrice d’incidence. Le développeur a fait ce choix, parce que le traitement des matrices diffèrent d’une manière importante.

Le développeur a choisi un tableau à deux dimensions pour représenter une matrice, puisque la matrice ne changera pas pendant le traitement d’un graphe et le développeur voulait mesurer exactement la quantité d’itérations et affectations fait pendant le programme traverse un graphe.

Au début, les données étaient tous structurées sous forme de tableaux, ce qui a marchais assez bien, mais le manque des méthodes de ArrayList a augmenté considérément le temps de développement. Plus important, il y avait une situation où le développeur n’a pas pu identifier la taille potentielle d’un tableau.

Cette situation a été quand le développeur voulais représenter les arcs que l’algorithme « parcourir() » (recherche en profondeur) a pris lors de son recherche (« arcsD », informellement appelés arcs directs) et les arcs de redondance (« arcsA », informellement appelés arcs arrières). Donc, ce fait a motivé le développeur à changer tous les tableaux en ArrayLists (sauf que les matrices).

Une chose importante à mentionner est le fait que dans « GrapheA.java » presque tous les ArrayLists concernant les arcs, entreposent des indices pour un ArrayList « Maitre » (retourné par la méthode GrapheA.identifieArcs()) qui contient les arcs sous forme d’adjacence unique (il n’y a pas de doublon comme [1,0] et [0,1]). Le développeur a créé ce méthode parce qu’il faut déterminer les arcs pour plusieurs traitements dans le programme. Plus que la moitié des méthodes dans GrapheA font référence a identifieArcs().

Finalement, un HashMap<String, Integer> a été utilisé pour représenter le nom de chaque sommet et son propre degré. Un HashMap est utile pour associer deux valeurs des types différents, et le développeur voulait utiliser une chaîne (e.g. « S0 », « S1 », « S2 ») pour représenter chaque sommet, qui a aussi un degré associé (sous forme d’entier).

## Les algorithmes

Le programme contient 3 fichiers :

* Main.java – le fichier principale qui contient le méthode pilot et l’entrée des données de graphe sous forme de matrice
* GrapheI.java – le ficher qui représente et traite un graphe par sa matrice d’incidence
* GrapheA.java – le ficher qui représente et traite un graphe par sa matrice d’adjacence

### Les algorithmes identiques entre GrapheA.java et GrapheI.java

#### evaluation ()