TP test

L'objectif de ces TPs est la simulation d'une campagne de tests simplifiée et l'utilisation d'un outil pour suivre la couverture du code (EMMA ou un autre plugIn similaire pour Java)

Structure du document

0. Consignes sur le déroulement des TPs
I. Exemple utilisé
II. Scripts de test pour test fonctionnel et analyse de couverture sur un exemple simple
III. Annexe - Précisions sur l'utilisation du logiciel Emma en ligne de commande

Consignes sur le déroulement des TPs - partie tests

Le travail peut s'effectuer en groupe (stable) de 2-3 étudiants. En plus du travail explicitement demandé, vous devez rédiger un rapport de TP.

Un dépôt (par groupe de travail) est attendu (date précisée sous moodle). Ce dépôt contiendra le rapport de TP et les éléments demandés par la suite dans cet énoncé (suite de test, script, programme Java, captures d'écran avec l'outil de couverture de code, rapport)

Le logiciel utilisé pour suivre la couverture de code peut-être Emma. Ce logiciel se présente soit sur la forme d'un plug in Eclipse ou en ligne de commande. Vous pouvez choisir d'utiliser l'outil comme vous le souhaitez. Les expériences des années précédentes montrent que l'utilisation du plug in Eclipse est plus aisée.

I. Exemple utilisé

On s'intéresse à l'implementation de l'algorithme de Bellman-Ford, permettant de déterminer un plus court chemin dans un graphe non orienté étiqueté. Cet algorithme fonctionne avec des poids négatifs et permet la détection de cycles de poids négatif.

Le test peut se faire sur une librairie Java de votre choix qui implémentant cet algorithme (librairie que vous avez écrit / que vous écrivez pour l'occasion, ou une librairie disponible sur internet). Un exemple (avec des explications) est disponible ici

II. Scripts de test pour test fonctionnel et analyse de couverture sur un exemple simple

Question 0

a) Assurez-vous que vous disposez d'une implémentation en Java de l'algorithme Bellman-Ford, que vous avez bien identifié les entrées et les sorties du programme.
b) Mettez au propre la spécification de ce programme.

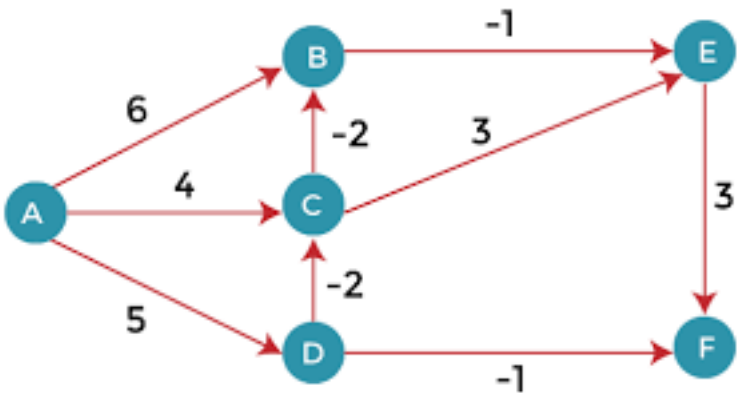
Assurez-vous que l'algorithme est bien spécifié, que les exigences sont clairement identifiées et listées, en suivant les préconisations vues en cours.

Question 1

Ecrire un script de test (en ce que vous voulez Java, C, shell, …) qui lit dans un fichier des jeux de test et des oracles, de façon à automatiser votre campagne de tests. Vous pouvez vous inspirer de l'exemple fournit en cours dans la partie processus de test.
Le script de test doit fournir un rapport d'analyse qui précise explicitement les tests qui ont été exécutés et le résultat de leur exécution (pass/fail), ainsi que quelques statistiques.

Question 2

En utilisant le logiciel EMMA, établir quelles **instructions** du programme permettent de trouver le plus court chemin dans le graphe suivant:



Question 3

Est-ce qu'on peut trouver une suite de tests qui couvre toutes les instructions?
Si oui, donnez la.
Sinon expliquez pourquoi?

Question 4

Créer une suite de test en appliquant la méthode des partitions et des catégories. Expliquer votre démarche. « Passer » les tests de votre suite de tests sur le programme obtenu à la question 1. La suite de test doit être facilement extensible.

Question 5

Créer une suite de test en appliquant la méthode du test aux limites. Expliquer votre démarche. « Passer » les tests de votre suite de tests sur le programme obtenu à la question 1. La suite de test doit être facilement extensible.

Question 6

Définissez 3 *critères de mutation pertinents* et créer les mutants de votre programme de référence, en appliquant chacun de ces critères.
Passez la suite de tests obtenue aux questions précédentes sur chacun de ces mutants.
Analysez les résultats obtenus et commentez-les.

Annexe - Logiciel EMMA - installation et familiarisation avec l'outil en ligne de commande

EMMA est un outil open-source qui permet de mesurer et visualiser la couverture de code Java.
Le site de l'outil, pour explications, exemples commentaires est le suivant :
http://emma.sourceforge.net/
Le même site vous permet de télécharger le logiciel.

Le logiciel dispose d'une version plug-in Eclipse et d'une version indépendante. Vous avez l'entière liberté quant à l'utilisation d'une ou l'autre de ces versions, l'expérience montre que le plug-in est plus agréable à utiliser. Afin de faciliter l'utilisation d'Emma en ligne de commande la partie familiarisation concerne la version en ligne de commande. Si vous le souhaitez, vous pouvez effectuer des essais similaires avec le plug-in Eclipse.

Les consignes données ici pour Emma font référence à la version indépendante.Vous êtes encouragés à utiliser la version plug-in Eclipse, dont l'utilisation pour le reste du projet peut-être plus facile.

Pour utiliser le logiciel il faut que emma.jar se trouve dans votre répertoire de travail, ou qu'il soit visible à cet endroit.

En principe, EMMA est utilisé en complément d'une suite de test fonctionnel (e.g. JUnit), mais le logiciel peut aussi aider pour des tests, moins formels, effectués « à la volée ».

Pour calculer la couverture du code, EMMA peut prendre en compte soit une seule exécution, soit plusieurs, après une *instrumentation* du code source. Pour cela, EMMA a besoin d'une copie locale du code source à instrumenter, qu'elle va modifier en lui rajoutant des informations dont EMMA a besoin pour calculer les couvertures cumulés.
En plus de ça, EMMA garde, dans un fichier local coverage.em, des meta-informations sur le fichier à tester. Avec ces informations on peut tester en mode différé le logiciel instrumenté.

Exemple de démarche pour une prise en main d'Emma en ligne de commande

Nous allons considérer un des exemples les plus complexes de la distribution de java : SwingSet2.

1. Familiarisez-vous avec l'exemple. Assurez-vous d'avoir découvert au moins 7 des fonctionnalités différentes de l'outil.
   pour le lancer à partir de la ligne de commande vous tapez :
   java –jar SwingSet2.jar
2. Jouez avec l'exemple en indiquant à EMMA de suivre la couverture de code qui correspond à vos actions. Essayez-le également par rapport à 7 fonctionnalités
   java -cp emma.jar emmarun -jar SwingSet2.jar

3. Effectuez la même chose qu'au point 2, tout en générant un fichier .html
   java -cp emma.jar emmarun -r html -jar SwingSet2.jar
4. Consulter l'aide de l'outil pour découvrir qu'est-ce que vous pouvez paramétrer par rapport à la présentation du fichier .html
5. Effectuez la même chose qu'au point 3, tout en suivant la couverture sur le code source :

   option -sp .../jdk1.4.2/demo/jfc/SwingSet2/src
6. Initialiser l'instrumentation du code source (après avoir crée une copie locale)

   cp .../jdk1.4.2/demo/jfc/SwingSet2/SwingSet2.jar .
7. java -cp emma.jar emma instr -m overwrite -cp SwingSet2.jar
8. Lancez l'exécution de l'exemple en mode différé

   java -cp SwingSet2.jar:emma.jar SwingSet2
9. Générez le rapport de couverture qui utilise les données obtenues lors des exécutions antérieures :

   java -cp emma.jar emma report -r html -in coverage.em,coverage.ec
10. Suivez la couverture sur le code source.