LOG3430

MÉTHODES DE TEST & DE VALIDATION DU LOGICIEL

Laboratoire 2

Tests Combinatoires et Couverture de Code

Sanghyuk Lee 1589242

François-Xavier Legault 1876882

1 Octobre 2019

Introduction

L'objectif de ce travail pratique est de se familiariser avec les tests en boîte noire. On ne devait donc pas se baser sur l'implémentation des méthodes fournies. Il existe trois critères d'adéquation basés sur le choix d'utilisation afin d'effectuer des tests en boîte noire (tests combinatoires). Lors de ce laboratoire, nous allons nous concentrer sur les critères "all choices" et "each choice", mais normalement nous retrouverions le troisième critère "basic choice" qui s'agit d'un compromis entre les deux premiers critères. Le tout, en se familiarisant et utilisant les outils "unittest" et "coverage".

Première Partie (boîte noire)

Les méthodes que nous devions tester se trouvent dans le fichier "generators.py" et sont les suivantes:

- simple
- simple_with_probability
- bipartite
- bipartite_with_probability
- eurlerianCycle
- regular

La première étape de la première partie consistait à tester les méthodes de génération des graphes en utilisant la technique "each choice", puis de refaire le même travail en utilisant la technique "all choices".

Deuxième Partie (boîte blanche)

Nous nous sommes servis de l'outil "coverage" afin d'effectuer cette deuxième partie, vu qu'il fallait évaluer l'efficacité de nos tests. Nous avons remarqué quelque chose d'inquiétant; la couverture de nos tests n'atteignait jamais 100% peu importe la qualité et le nombre de nos tests. Après un peu recherche, nous avons remarqué que quelques fonctions dans le fichier "generators.py" n'étaient pas demandé d'être testées et ne se faisaient pas appelée par les fonctions qui étaient testées. En enlevant ces fonctions ("path", "cycle", "eulerianPath", "wheel" et "star"), notre couverture augmentait effectivement à 100%.

Après nos tests “each choice” et avoir enlevé les fonctions non-couverte par le TP (mentionnées plus haut) :

```
PS C:\Users\Efix\Documents\GitHub\3430\FX_Chris\3430\TP2\Graph App> coverage report
Name            Stmts   Miss  Cover
---------------------------------
EC_test.py        124     12    90%
generators.py      78      1    99%
models.py          52     20    62%
utils.py            5      1    80%
---------------------------------
TOTAL             259     34    87%
PS C:\Users\Efix\Documents\GitHub\3430\FX_Chris\3430\TP2\Graph App>
```

Après nos tests “all choice” et avoir enlevé les fonctions non-couverte par le TP (mentionnées plus haut) :

```
PS C:\Users\Efix\Documents\GitHub\3430\FX_Chris\3430\TP2\Graph App> coverage report
Name            Stmts   Miss  Cover
---------------------------------
AC_test.py        306     12    96%
generators.py      78      0   100%
models.py          52     19    63%
utils.py            5      1    80%
---------------------------------
TOTAL             441     32    93%
PS C:\Users\Efix\Documents\GitHub\3430\FX_Chris\3430\TP2\Graph App>
```

On voit alors une couverture de 100% pour “all choice” et 99% pour “each choice”. La prochaine étape était alors de trouver pourquoi nous perdions ce 1% dans la section “each choice”. Ce 1% manquant, nous l’avons trouvé facilement, et cela est dû à une des astuces de travail que nous avions en travaillant sur ce travail pratique. En effet, pour s’assurer d’une couverture de 100%, à chaque fois que nous travaillions sur une certaine fonction dans le fichier “genarators.py”, nous effacions toutes les autres et on s’assurait d’avoir une couverture de 100% sur la fonction présente.

Cela n’était pas trop difficile lorsque nous définissions bien nos catégories pour chacunes de nos variables. Finalement, ce 1% se trouvait dans un endroit inévitable pour les tests de type “each choice” qui est moins rigoureux que “all choice”, il s’agissait en fait d’une seule ligne (115) dans la fonction “eulerian Cycle”:

Ligne (115) initialement non-couverte par les tests “each choice”

```
104  def eulerianCycle(V, E):
105      '''
106      Returns an Eulerian cycle graph on V vertices.
107      @param  V the number of vertices in the cycle
108      @param  E the number of edges in the cycle
109      @return a graph that is an Eulerian cycle on V vertices and E edges
110      @raises ValueError if either V <= 0 or E <= 0
111      '''
112      if E <= 0:
113          raise ValueError("An Eulerian cycle must have at least one edge")
114      if V <= 0:
115          #prochaine ligne non couverte par tests EC
116          raise ValueError("An Eulerian cycle must have at least one vertex")
117      G = Graph(V)
118      vertices = [rand.randrange(V) for i in range(E)]
119      for i in range(E-1):
120          G.add_edge((vertices[i], vertices[i+1]))
121      G.add_edge((vertices[E-1], vertices[0]))
122      return G
```

Nous avons alors ajouté un test “each choice” supplémentaire pour arriver à une couverture de 100%

Après l’ajout d’un test EC supplémentaire lors de la 2ème étape (boite blanche):

```
PS C:\Users\Efix\Documents\GitHub\3430\FX_Chris\3430\TP2\Graph App> coverage report
Name            Stmts   Miss  Cover
---------------------------------------
EC_test.py        131     12    91%
generators.py      78      0   100%
models.py          52     20    62%
utils.py            5      1    80%
---------------------------------------
TOTAL             266     33    88%
PS C:\Users\Efix\Documents\GitHub\3430\FX_Chris\3430\TP2\Graph App>
```

100%! Nous tenons à le mentionner encore, ce 100% que nous obtenons, c’est en enlevant les fonctions (“path”, “cycle”, “eulerianPath”, “wheel” et “star”) qui n’étaient pas mentionnées dans l’énoncé du travail pratique (ni appelées par les fonctions à tester) et donc à ne pas couvrir par nos tests. Dans la remise de ce travail, nous avons bien sur enlevé ces fonctions dont nous n’avions pas à en évaluer la couverture par nos tests.

Question 5

Nous pensons que le code est conforme à sa spécification et ses commentaires pour la majorité du fichier. Il y a cependant quelques remarques à faire. En effet, certaines fonctions (mentionnées plus haut) n'étaient jamais utilisées ni appelées, et nous n'avions pas à les tester (selon l'énoncé du travail pratique) nous les avons donc supprimées. De plus, dans la plupart des fonctions, il était vérifié implicitement que les sommets ("vertices") n'étaient pas négatifs, puisque cette vérification était fait dans le fichier "models.py" dans le constructeur de la classe "Graph".

Conclusion

Nous nous sommes donc familiarisé lors de ce laboratoire avec les tests en boîte noire et les outils ("unittest" et "coverage") nous aidant à rédiger des tests et de s'assurer de leur couverture. Nous avons utilisé les critères d'adéquations "each choice" et "all choices" afin de réaliser ce travail pratique. Nous nous sommes finalement assuré d'obtenir une couverture de 100% du code, d'avoir rédigé des commentaires clairs et explicatifs ainsi que de comprendre pourquoi nous obtenions certains pourcentages de couverture pour nos tests.