SFU CMPT 473 SPR2019 Assignment 3

Individual Analysis

Method 1 `org.jgrapht.generate.CompleteGraphGenerator@<init>(int)`

> 6 Mutants: 2 Killed (6351,6354), 4 Live (6349,6350,6352,6353), 0 Uncovered

- `6349:LVR:0:POS:org.jgrapht.generate.CompleteGraphGenerator@<init>(int):76:0 |==> 1` . Literal Value Replacement. `0` was changed to `1` , transforming `if (size < 0)` to `if (size < 1)` . **Mutant lived**. Originaly, this condition prevents generation of graphs with negative size **but allows graphs of size 0**. However, the mutant **prevents empty graphs to be generated!** This could have been caught by testing the generation of 0-size graphs.

- `6350:LVR:0:NEG:org.jgrapht.generate.CompleteGraphGenerator@<init>(int):76:0 |==> -1` . Literal Value Replacement. `0` was changed to `-1` , transforming `if (size < 0)` to `if (size < -1)` . **Mutant lived**. The original condition was supposed to **prevent the creation of negative size graphs**. However, this mutant **allows graphs of `size = -1` to be generated** which breaks the business logic. This could be fixed with a negative testcase that attempts to generate a negative size graph.

- `6351:ROR:<(int,int):!=(int,int):org.jgrapht.generate.CompleteGraphGenerator@<init>(int):76:size < 0 |==> size != 0` . Relational Operator Replacement. `<` was changed to `!=` , transforming `if (size < 0)` to `if (size != 0)` . **Mutant threw an exception**. Originally, the function was written to throw an exception for negative numbers, but this mutant makes it throw an exception for any number except 0. **This was a good catch by the test.**

- `6352:ROR:<(int,int):<=(int,int):org.jgrapht.generate.CompleteGraphGenerator@<init>(int):76:size < 0 |==> size <= 0` . Relational Operator Replacement. `<` was changed to `<=` , transforming `if (size < 0)` to `if (size <= 0)` . **Mutant Lived.** Orginally the function allows generation of 0-size graphs, but the mutant prevents generation of 0-size graphs. This mutant could have been killed by a testcase that generates a 0-size graph.

- `6353:ROR:<(int,int):FALSE(int,int):org.jgrapht.generate.CompleteGraphGenerator@<init>(int):76:size < 0 |==> false` . Relational Operator Replacement. `<` was changed to `false` , transforming `if (size < 0)` to `if (false)` . **Mutant Lived.** This mutant would have been killed with a negative testcase that checks generating a graph with negative size.

- `6354:STD:<ASSIGN>:<NO-OP>:org.jgrapht.generate.CompleteGraphGenerator@<init>(int):80:this.size = size |==> <NO-OP>` . Statement Deletion. Assignment was changed to *NO-OP*, deleteing the statement `this.size = size` . **Mutant killed.**

Method 2 `org.jgrapht.generate.CompleteGraphGenerator@generateGraph(..)`

> 20 Mutants: 12 Killed (6357,6361,6362,6364,6365,6366,6367,6368,6369,6370,6371,6372), 7 Live (6355,6356,6358,6359,6363,6373,6374), 1 Uncovered (6360)

- `6355:LVR:POS:0:org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.VertexFactory<V>,java.util.Ma` Literal Value Replacement. `1` was changed to `0` transforming `if (size < 1)` to `if (size < 0)` . **Mutant Lived.** Originally, this condition prevents generation of 0-sized graphs. This mutant allows generating a 0-sized graph. Mutant could have been killed by a testcase checking for a 0-sized graph.

- `6356:LVR:POS:NEG:org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.VertexFactory<V>,java.util.` Literal Value Replacement. `1` was changed to `-1` , transforming `if (size < 1)` to `if (size < -1)` . **Mutant Lived.** Originally, this condition prevents generation of graphs with 0 or negative size. This mutant allows generating a graph or 0 or -1 size!. This could have been caught by a negative testcase checking for graph of negative size.

- `6357:ROR:<(int,int):!=(int,int):org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.VertexFactor` Relational Operator Replacement. `<` was changed to `!=` i.e `if (size < 1)` becomes `if (size != 1)` . **Mutant Killed.** This mutant prevents the generation of any graph of size other than 1. This was easily caught by the testsuite because there is a testcase checking for a size != 1.

- `6358:ROR:<(int,int):<=(int,int):org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.VertexFactor` Relational Operator Replacement. `<` was changed to `<=` i.e `if (size < 1)` becomes `if (size <= 1)` . **Mutant Lived.** Mutant could have been killed by a testcase using a graph of size 1.

- `6359:ROR:<(int,int):FALSE(int,int):org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.VertexFac` Relational Operator Replacement. `<` was changed to `false` transforming `if (size < 1)` to `if (false)` . **Mutant Lived.** Mutant could have been killed by a testcase with a graph of size 0 or negative *which expects a failure.*

- `6360:STD:<RETURN>:<NO-OP>:org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.VertexFactory<V>,j` Statement Deletion. The `return` statement was simply deleted. **Mutant Uncovered.** Unsure why this mutant was uncovered by the testsuite, it should have lived.

- `6361:LVR:0:POS:org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.VertexFactory<V>,java.util.Ma` Literal Value Replacement. `0` was replaced with `1` , transforming `for (int i = 0; i < size; i++)` to `for (int i = 1; i < size; i++)` . In effect an *off-by-one error.* **Mutant Killed.** The testsuite correctly detects this error as the final produced graph will have a size less than expected.

- `6362:LVR:0:NEG:org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.VertexFactory<V>,java.util.Ma`

Literal Value Replacement. `0` was replaced with `-1`, transforming `for (int i = 0; i < size; i++)` to `for (int i = -1; i < size; i++)`. **Mutant Killed.** In effect, the loop runs 1 more than the required number of times, hence the generated graph has 1 more vertex, and the testsuite correctly detects this.

- `6363:ROR:<(int,int):!=(int,int):org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.VertexFactor`
  Relational Operator Replacement. `<` was replaced with `!=`, transforming `for (int i = 0; i < size; i++)` to `for (int i = 1; i != size; i++)`. **Mutant Lived**. This is an **Equivalent Mutant**, because in both cases, the loop quits when `i == size`.

- `6364:ROR:<(int,int):<=(int,int):org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.VertexFactor`
  Relational Operator Replacement. `<` was replaced with `<=`, transforming `for (int i = 0; i < size; i++)` to `for (int i = 1; i <= size; i++)`. **Mutant Killed**. The mutant runs the loop 1 extra time, creating 1 extra vertex for the graph. The testsuite correctly detects this.

- `6365:ROR:<(int,int):FALSE(int,int):org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.VertexFac`
  Relational Operator Replacement. `<` was replaced with `false`, transforming `for (int i = 0; i < size; i++)` to `for (int i = 1; false; i++)`. **Mutant Killed.** The loop is not run in the mutant, and the testsuite correctly detects this.

- `6366:EVR:<METHOD_INVOCATION(V)>:<DEFAULT>:org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.Ve`
  Expression Value Replacement. A method invocation was replaced with a default value, transforming `V newVertex = vertexFactory.createVertex();` to `V newVertex = null;`. **Mutant threw an exception.** This mutant was caught because the value is expected to be non-null later down the code.

- `6367:STD:<CALL>:<NO-OP>:org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.VertexFactory<V>,jav`
  Statement Deletion. A method invocation was changed to *NO-OP*, deleting the statement `target.addVertex(newVertex);`. **Mutant Killed.** The mutant effectively does not create any vertices for the graph, and this is detected by the testsuite.

- `6368:EVR:<METHOD_INVOCATION(java.util.Iterator<V>)>:<DEFAULT>:org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph`
  Expression Value Replacement. A method invocation was replaced with a default value, transforming `Iterator<V> slowI = target.vertexSet().iterator();` to `Iterator<V> slowI = null;`. **Mutant threw an exception.** The mutant results in a null-pointer exception when the variable is later dereferenced.

- `6369:EVR:<METHOD_INVOCATION(V)>:<DEFAULT>:org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.Ve`
  Expression Value Replacement. A method invocation was replaced with a default value, transforming `V latestVertex = slowI.next();` to `V latestVertex = null;`. **Mutant threw an exception.** The mutant results in a null-pointer exception down the road when this variable is assigned to another variable that ends up being dereferenced.

- `6370:EVR:<METHOD_INVOCATION(java.util.Iterator<V>)>:<DEFAULT>:org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph`
  Expression Value Replacement. A method invocation was replaced with a default value, transforming `fastI = target.vertexSet().iterator();` to `fastI = null;`. **Mutant threw an exception.** The mutant results in a null-pointer exception down the road when this variable is dereferenced.

- `6371:ROR:!=(java.lang.Object,java.lang.Object):TRUE(java.lang.Object,java.lang.Object):org.jgrapht.generate.CompleteGraphGenerator@genera`
  Relational Operator Replacement. `!=` was replaced with `true`, transforming `while (fastI.next() != latestVertex)` to `while (true)`. **Mutant Killed.** The mutant effectively makes the program loop infinitely. I'm curious why the mutant did not timeout.

- `6372:EVR:<METHOD_INVOCATION(V)>:<DEFAULT>:org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.Ve`
  Expression Value Replacement. A method invocation was replaced with a default value, transforming `temp = fastI.next();` to `temp = null;`. **Mutant threw an exception.** The mutant results in a null-pointer exception when the variable is later dereferenced.

- `6373:STD:<CALL>:<NO-OP>:org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.VertexFactory<V>,jav`
  Statement Deletion. The statement `target.addEdge(latestVertex, temp);` was deleted. **Mutant Lived.** The mutant does not add one end of a graph edge. This should have been killed by a testcase which verifies the number of edges in a generated graph.

- `6374:STD:<CALL>:<NO-OP>:org.jgrapht.generate.CompleteGraphGenerator@generateGraph(org.jgrapht.Graph<V,E>,org.jgrapht.VertexFactory<V>,jav`
  Statement Deletion. The statement `target.addEdge(temp, latestVertex);` was deleted. **Mutant Lived.** The mutant does not add one end of a graph edge. This should have been killed by a testcase which verifies the number of edges in a generated graph.

Summary

14 Killed + 11 Lived = 25. This is 1 less than the total number of mutants generated for both functions (i.e 26). The 1 unaccounted mutant was not covered by the testsuite.

Individual Methods Metrics

```
Method 1 i.e CompleteGraphGenerator@<init>(int)
    Mutation Score
        = MutantsKilled / MutantsGenerated
        = 2 / 6
        = 33.3%
    TestSuiteEffectiveness
        = MutantsKilled / MutantsCovered
        = 2 / 6
        = 33.3%

Method 2 i.e CompleteGraphGenerator@generateGraph()
    Mutation Score
        = MutantsKilled / MutantsGenerated
        = 12 / 20
        = 60%
    TestSuiteEffectiveness
        = MutantsKilled / MutantsCovered
        = 12 / 19
        = 63.2%
```

Deduction

The results show that the testsuite is ineffectively testing method 1. Although it covers all generated mutants, it is very ineffective in killing these mutants. On the other hand, the testsuite does not cover 1 mutant of method 2, but it is more effective in killing the covered mutants. In sum, the testsuite to test more varied configurations of method 1.