

Nitin Nagavel

HW #4 Software Testing

Com S/SE 417 Spring 2022

Handed out April 7th, 2022

Due at midnight, April 14th as a pdf uploaded to Canvas

Homework Policy

Homework Policy: The homework assignment should be done individually. You may talk to classmates about the problems in general, but you must complete the homework on your own. You are not permitted to use published answers from websites, etc. Assistance by others must be specifically credited in the solution to the problem that is turned in, describing what the contribution was (e.g., "Thanks to [name] for explaining X in Problem Y", not "Thanks to [name] for help with HW1."). The Dean of Students Office offers several good [resources](#) to build understanding of how to avoid plagiarism, such as Purdue's "[Safe Practices](#)" site.

The goal is for each of you to learn the material sufficiently well to use it productively, to think innovatively, and to develop confidence in your problem-solving abilities. Feel free to talk to me individually about this if you have any questions.

Late policy: 10% penalty per day for late homework. Assignments will not be accepted after April 18th, unless otherwise arranged/discussed with me.

Homework problems are adapted from the course textbook, "Introduction to Software Testing", 2nd edition, Ammann & Offutt, 2017. There is a Student Solution Manuals available online with answers to other practice questions <https://cs.gmu.edu/~offutt/softwaretest/exer-student.pdf>.

Q1. Answer the questions based on the following Graph: (35 points)

$N = \{1, 2, 3, 4, 5, 6, 7\}$

$N_0 = \{1\}$

$N_f = \{6\}$

$E = \{(1,2), (2,3), (2,4), (3,2), (4,5), (4,6), (5,6), (6,1), (7,6), (4,7), (5,7)\}$ And

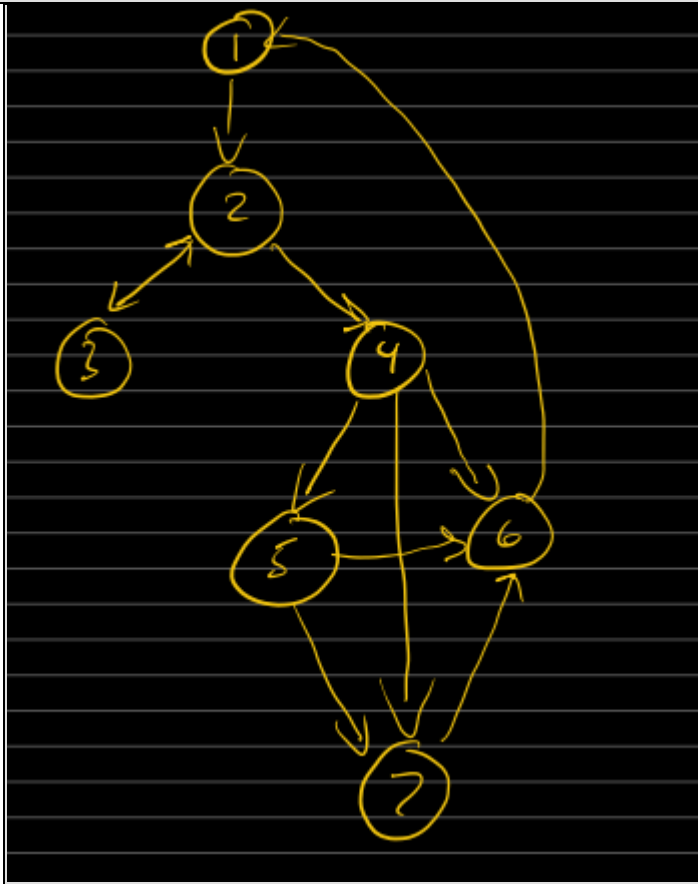
the following candidate test paths:

[1,2,4,5,6]

[1,2,3,2,4,7,6]

[1,2,4,6,1,2,4,7,6]

(a) **Draw the graph** (you can use the tool from the book website) and include a screenshot/picture



(b) **How many simple paths are there** (leave out length 0- single nodes)? 4 simple paths

(c) **List the prime paths**

1, 2, 4, 5, 7, 6

(d) **Does the given set of test paths satisfy** (1) Edge Coverage (2) Edge Pair Coverage (3) Node coverage? If not add test paths to complete those requirements (do not delete existing test paths)

(1) The given set of test paths didn't fully satisfy the edge coverage as the edge from 5 to 7 was not tested.

An example test would be [1, 2, 4, 5, 7, 6]

(2) So edge pair coverage was not fully achieved but it can be accomplished by adding the test previously [1, 2, 4, 5, 7, 6] in conjunction with:

[1, 2, 4, 6]

[1, 2, 4, 7, 6]

[1, 2, 4, 5, 7, 6]

(3) Node coverage is achieved

Q2. Draw a control flow graph for the following program fragment. Make sure to include the statements and conditionals for the nodes/edges. (35 points)

```
a = b          if
```

```
(a > 0) {
```

```
    a=a-1
```

```
}
```

```
else if (a==0){
```

```
    a=a+1
```

```
}
```

```
else{
```

```
    a=a*2
```

```
}
```

```
while (a > 0){
```

```
    print "loop"
```

```
    a -- if(a
```

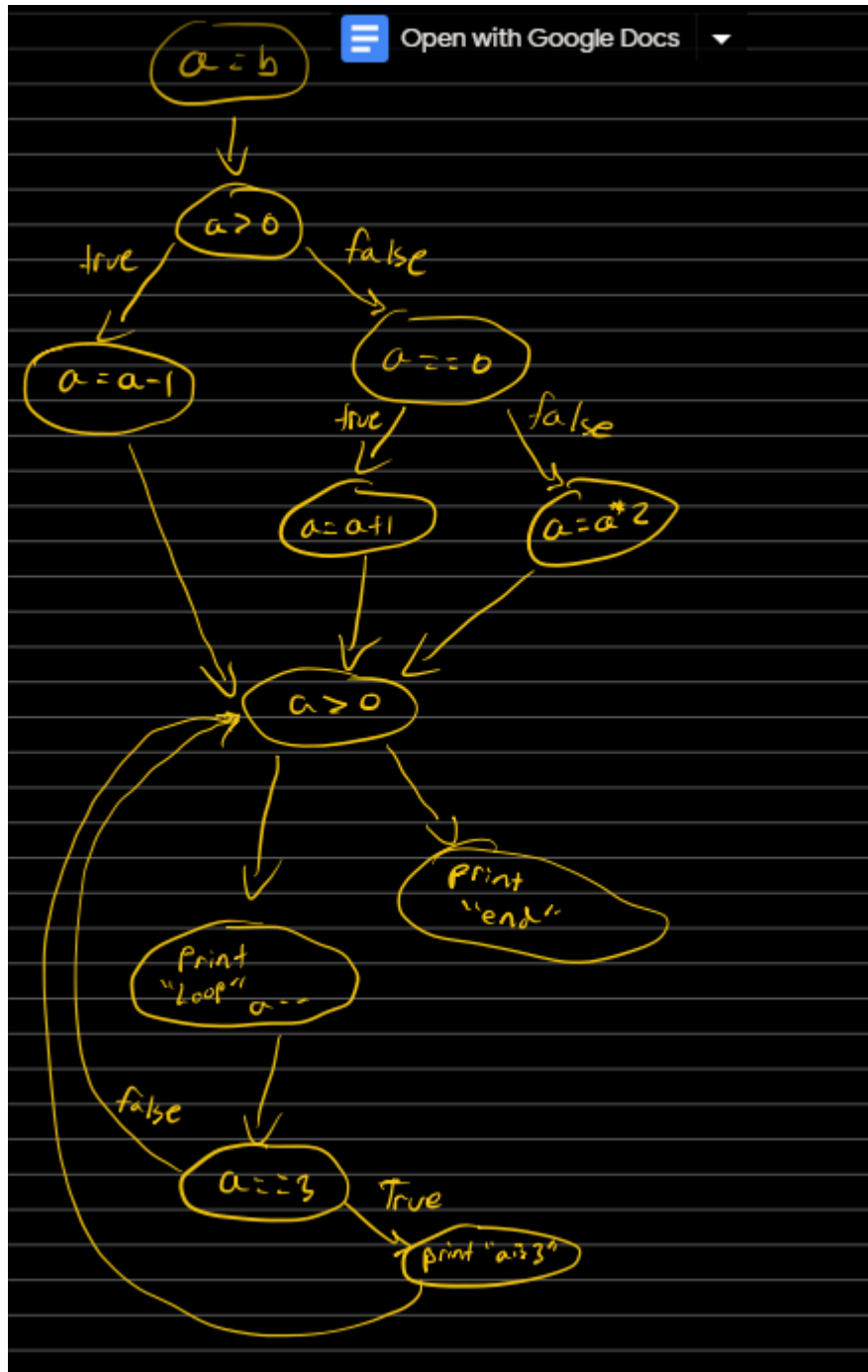
```
==3) {    print
```

```
    "a is 3"
```

```
    }
```

```
}
```

```
print "end"
```



Q3. Using a mutation testing tool: (25 points)

1. Download the tar file from canvas (mutation-pitest-example.tar) and extract on pyrite (this may work on your local machine, but it has been tested on pyrite). This is a an example program that uses pitest
2. Move to the triangle-example directory and type ➤ mvn verify
(you may see some warnings but you can ignore those)

1. Once the program has run **copy the output report (on the screen)** starting with:

```
=====
- Timings
=====
- Timings
=====
> scan classpath : < 1 second
> coverage and dependency analysis : 15 seconds
> build mutation tests : < 1 second
> run mutation analysis : 19 seconds
=====
> Total   : 35 seconds
=====
- Statistics
=====
>> Generated 51 mutations Killed 35 (69%)
>> Ran 130 tests (2.55 tests per mutation)
=====
- Mutators
=====
> org.pitest.mutationtest.engine.gregor.mutators.ConditionalsBoundaryMutator
>> Generated 10 Killed 0 (0%)
> KILLED 0 SURVIVED 9 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 1
=====
> org.pitest.mutationtest.engine.gregor.mutators.RemoveConditionalMutator_EQUAL_ELSE
>> Generated 7 Killed 6 (86%)
> KILLED 6 SURVIVED 0 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 1
=====
> org.pitest.mutationtest.engine.gregor.mutators.ReturnValsMutator
>> Generated 8 Killed 6 (75%)
> KILLED 6 SURVIVED 0 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 2
=====
> org.pitest.mutationtest.engine.gregor.mutators.MathMutator
>> Generated 9 Killed 8 (89%)
> KILLED 8 SURVIVED 0 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 1
=====
> org.pitest.mutationtest.engine.gregor.mutators.NegateConditionalsMutator
>> Generated 17 Killed 15 (88%)
> KILLED 15 SURVIVED 0 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 2
=====
```

2. Highlight or provide a summary at the end that states the overall statistics (number of total mutants generated and killed)

```

- Statistics
=====
>> Generated 51 mutations Killed 35 (69%)
>> Ran 130 tests (2.55 tests per mutation)
=====

```

- Go to the target/pit-reports directory and copy the report directory to your local machine. Open up the index.html file. And take a screenshot (handin)

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage	Mutation Coverage
1	83% <div><div>19/23</div></div>	69% <div><div>35/51</div></div>

Breakdown by Package

Name	Number of Classes	Line Coverage	Mutation Coverage
com.example 1	1	83% <div><div>19/23</div></div>	69% <div><div>35/51</div></div>

Report generated by [PIT](#) 1.1.12-SNAPSHOT

- Click on com.example and then on triangle.java to show the source code and mutations. Take a screenshot of the mutations for lines 8-22

```

8 6 if ((a <= 0) || (b <= 0) || (c <= 0)) {
9 1 return TriangleType.INVALID;
10 }
11 trian = 0;
12 2 if (a == b) {
13 1 trian = trian + 1;
14 }
15 2 if (a == c) {
16 1 trian = trian + 2;
17 }
18 2 if (b == c) {
19 1 trian = trian + 3;
20 }
21 2 if (trian == 0) {
22 9 if (((a + b) < c) || ((a + c) < b) || ((b + c) < a)) {

```

- If you hover over the mutants it will tell you what they are and if they survived or were killed. **Pick ONE mutation that survived** and explain why. Make sure to state which line/mutation it is.

- a On line 22, Mutation 1 survived because the changed boundary was not strong enough to affect the rest of the program. Also, the input given by the test didn't have to rely on the conditional boundary that was changed.

Q4. Briefly Identify your team project and list your team members:(5 points):

Automated Web Testing Using Selenium:

Nitin Nagavel, Adam Banwell, Evan Danforth