**Software Testing Project**

*Student: Tung L. Nguyen*

*Instructor: Alex Groce*

*Course: CS567*

*Semester: Fall 2023*

**1. Introduction**

This project is an exploration into mutation testing, inspired by the insights gained from the CS567 course. The report acknowledges the personal nature of its findings and aims to shed light on mutation testing through a practical lens.

**1.1 Understanding Mutation Testing**

Mutation testing evaluates the effectiveness of a test suite by intentionally introducing faults into the code and assessing the suite's ability to detect these mutations. For example, altering an addition operation to subtraction in a simple function serves as a litmus test for the test suite's thoroughness.

**1.2 Tools Utilized**

The project employs mutpy and universalmutator as mutant generators, and tstl and unittest for testing. These tools offer diverse features and capabilities, facilitating a comprehensive exploration of mutation testing in Python.

**1.3 Objectives of the Study**

Rather than declaring a winner, the project aims to provide a nuanced comparison between mutpy and universalmutator. The report contributes valuable insights into the practical applications of mutation testing tools. The subsequent sections delve into the experimental process and results providing a comprehensive overview of the dynamic mutation testing landscape. All the source file is in my GitHub repo: <https://github.com/lamtung16/CS567_Project_Software_Testing>

**2. Experiments**

The program is used is a Python function to find the number with the highest frequency in a list. You can take a look at the source code at <https://github.com/lamtung16/CS567_Project_Software_Testing/blob/main/src_code/mutpy/myCode.py>

I chose this function because it’s easy to understand for any programmer and, it’s complicated enough for testing in term of introduction. The following subsections show you how to use tstl and mutpy to test the source Python code.

I used google colab to run my experiments because the it has the Linux environment and convenient to test, it’s a good choice for learning purpose.

* 1. **tstl:**
* Install tstl and universalmutator: pip install tstl universalmutator
* In the command line environment:
  + mkdir mutants # Create a directory to store mutant files.
  + tstl test\_myCode.tstl # Generate harness core for testing.
  + mutate myCode.py --mutantDir mutants # Generate mutants for myCode.py and store them in the 'mutants' directory.
  + analyze\_mutants myCode.py "tstl\_rt --timeout 120" --mutantDir mutants --verbose # Analyze all mutants from myCode.py

**2.2 mutpy:**

* Install mutpy: pip install mutpy
* In the command line environment:
* mkdir -p report # Create a directory to store .html reports.
* mut.py --target myCode --unit-test test\_myCode --show-mutants --report-html report --timeout-factor 30 --experimental-operators --colored-output # Analyze all mutants from myCode.py using test\_myCode.py

**3. Results**

**3.1. Number of mutants:**

- universalmutator: 53 mutants

- mutpy: 21 mutants

**3.2. Quality of mutants:**

Both universalmutator and mutpy produce similar mutants for my source code myCode.py. Most of the mutants are arithmetic operator replacement, assignment operator replacement, conditional operator deletion, one iteration loop, reverse iteration loop, statement deletion, zero iteration loop.

**3.3. Analyzing results:**

- Using unittest with mutpy’s mutants, I got mutation score is 82.6%

- Using tstl with universalmutator’s mutant, I got mutation score is 100% which is not what I expected.