Mutation Testing

Sualeh Fatehi

What Software Testing Does

- *Unit and functional testing* validates software **works as designed**
- *Regression testing* validates that software **still works after changes**

But how do we ensure that our tests are sufficient?

Ensuring That Tests are Sufficient

- Creating test plans with the team
- Measuring code coverage
  to see how much of the source code is executed
  during test runs
- Mutation testing

What is Mutation Testing?

- Measures the quality of the software tests themselves
- Idea proposed by Richard Lipton in 1971
- Requires computational power, and is catching on in recent times

Theory

- *Competent programmer hypothesis*
  software faults are due to small syntactic errors
- *Coupling effect*
  simple faults can cascade to form other emergent faults

How Does it Work?

- Introduce small random code changes called mutants
- Run all the tests that cover that area of code
- If any test fails, the mutant is killed
- If all tests pass, the mutant lives
- If a mutant lives, you have insufficient testing

Types of Mutations

- **Value Mutations**: Change values of constants to detect errors
- **Decision Mutations**: Change decisions or conditions (Boolean and arithmetic operations) to check for design errors
- **Statement Mutations**: Delete or duplicate statements, like copied and pasted code

Tools

Awesome Mutation Testing has a list of resources

- Stryker Mutator for C# and JavaScript
- PIT for Java