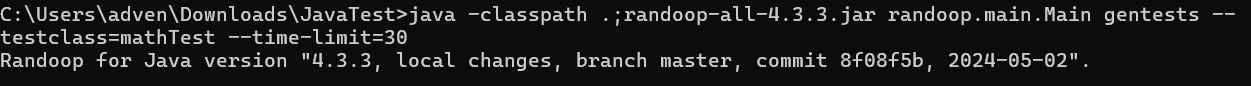
**Project Report**

Description: The main goal of this project was to reduce the test suite by getting rid of the redundant tests. By redundant tests, I meant tests the was covering the same branch of control flow graph. Control Flow Graph was an topic we learned in class that talked about how different inputs led to different branches of an program. In the class Example, there was a simple program with some branches and three different inputs which all led to different parts of the program. Interestingly enough, there were two different inputs that were pretty similar and one of the inputs followed a longer branch to end of program and one followed the shorter path but they covered the same parts. I thought at that time about why have multiple inputs going through same branch and how that wastes time. The goal was to reduce the test suites and still have the same coverage.

My solution was to have randomly generated test input for an program, convert the tests into sort of metrics and metadata that I could parse to get an score to be able to tell if a test was highly likely to be redundant.

Firstly, I used Randoop for my automatic test generation. The reason I used Randoop was because it could generate different kinds of tests (such as Regression and Unit Tests) and it could generate large quantities of it. 

Then I used a program called JaCoCo to get me metadata from the tests I generated. This can be used to compare different tests and analyze it to get an better understanding of what it is doing.



A black screen with white text

Description automatically generated

A black screen with white text

Description automatically generated

Then I developed a python program to parse the xml generated by JaCoCo and the parse the Test files itself and make it into better format to read. Then using the developed program in python, I ranked the two tests based on their metrics such as line numbers, and method numbers on an 1 through 10 score. Doing the same ranking for the test files based on parameters numbers and method amount, I was able to compare and average the score on 1-10 scale to get an score that told about how likely two tests are to be redundant to each other.

A screenshot of a computer

Description automatically generated

This solution can use the scores to guide developers to see if two tests are really redundant or not. In its current state, there are only manual evaluations but with that, I’ve tested some of the outputs given and most of them were accurate on being redundant. The important thing to understand is how this can be streamlined. The process can be automated with commands being put in a .bat file or some automation software to do this step, then go to next step and it will generate a file that will tell you how likely a test is redundant to each other. Of course, if an test is redundant, then the coverage will stay the same, so the issue is solved there.

A computer code with numbers and symbols

Description automatically generatedA computer code with text

Description automatically generated

Another benefit of this solution is that you can improve it using AI. What I mean is giving the AI the file containing the high possibility of two tests being redundant and the AI looking in the test files and using something like NLP to read and understand how similar two tests are and determine if they are redundant.

I was the only one working on this project, so all this contributed to me.