CS206 Software Testing

Prutha Patel (862393798)
Nikhil Anand Mahendrakar (862464249)

Experimental Results:

Benchmark	Coverage Criteria	Test Case Prioritization	No. of test cases	No. of faults
tcas	Statement and Branch Coverage	Random	12	41
		Additional	11	41
		Total	14	41
totinfo	Statement and Branch Coverage	Random	8	
		Additional	5	
		Total	5	
schedule	Statement and Branch Coverage	Random	10	9
		Additional	7	9
		Total	8	9
schedule2	Statement and Branch	Random	12	9

	Coverage	Additional	5	9
		Total	7	9
printtokens	Statement and Branch Coverage	Random	13	7
		Additional	6	7
		Total	7	7
pronttokens2	Statement and Branch Coverage	Random	13	9
		Additional	4	9
		Total	6	9
replace	Statement and Branch Coverage	Random	10	
		Additional	12	
		Total	21	

Observations:

How small are your test suites as compared to the original number of available test cases?

- The test suites appear to be significantly smaller than the original number of available test cases for each program. For example, for the program "tcas," there were initially 1590 available test cases, but the total number of test cases in the benchmark is only 14.
- Test cases are often designed to achieve specific coverage criteria, such as statement coverage and branch coverage. Instead of having numerous test cases that might redundantly cover the same code paths, a smaller set of test cases is selected to achieve the desired coverage criteria.This way we can ensures that critical parts of the code are adequately tested while minimizing redundancy.
- Overall, the smaller test suites are the result of a deliberate effort to optimize testing resources while still ensuring adequate coverage and effectiveness in detecting faults and errors in the software.

How do the suite sizes change according to different coverage criteria?

- The suite sizes will vary depending on the specific coverage criteria used and the complexity of the code being tested. More comprehensive coverage criteria tend to require larger test suites to achieve adequate coverage, while simpler criteria may result in smaller suites. Additionally, other factors such as prioritization techniques and resource constraints may also influence suite sizes across different coverage criteria.

How many faults are exposed by your test suites as compared to the total number of available faults, and as compared to the original test suite?

- As per the results we observed that fewer faults are been exposed by our test suits as compared to total number of available faults. The discrepancy between the faults exposed by the test suites and the total number of available faults, as well as the original test suite, is a result of various factors related to test suite size, effectiveness, coverage criteria, fault complexity, and testing limitations. While test suites aim to identify and mitigate software faults, they may not be able to uncover all potential issues due to inherent limitations in testing processes and resources.

Which coverage criteria seem to be the least and most effective at being able to expose faults?

- Branch coverage is typically considered the most effective at exposing faults due to their thorough exploration of code paths and critical decision points. However, all coverage criteria play important roles in the testing process, and a combination of different criteria is often used to achieve comprehensive test coverage and maximize fault exposure.

What other conclusions can you draw from your observations?

- It's noted that some lines in the printtokens program remain uncovered by the test suite, indicating potential gaps in coverage. Additionally, the critical use of the optimization flag (-O0) with the GCC compiler is emphasized for better code analysis, as it preserves the original code structure. However, random coverage may encounter performance issues due to its indiscriminate selection of test cases, potentially leading to inefficient test case generation. These insights underscore the importance of comprehensive coverage analysis, careful compiler settings, and optimization considerations in ensuring effective and efficient software testing practices.