CS731 SOFTWARE TESTING

PROJECT REPORT

MUTATION TESTING

ON

SOURCE CODE OF STANDARD ALGORITHMS

SUBMITTED BY:

DHRUV KHARKWAL (MT2022158)

TUSHAR GUNWARE (MT2022160)

PROJECT AIM

The aim of the project is to use **Mutation Testing** to test a real-world software project with the help of open-source tools.

GIT REPOSITORY

GitHub Repo: https://github.com/Dhruv-Kharkwal/MutationTesting

TESTING STRATEGY AND TOOLS USED

Mutation testing is a form of testing where small modifications are made to the source code (and each modification is called a mutant). The aim of mutation testing is to "kill" these mutants – that is show that making small changes to the code can alter the execution of the program and hence the result that it produces. There are 2 ways to "kill" a mutant:

1. Kill a mutant **weakly**: Here, the memory state of the program after the execution of the mutated statement is different from the memory state of the program when the statement was not mutated and executed. Notice that in this case, the output of the program on a test case can remain the same, irrespective of whether a program statement was mutated or not.
2. Kill a mutant **strongly**: Here, the output of the program on a test case, when a statement was mutated and not mutated, must change. Notice that when we kill a mutant strongly, the error propagates through the program and we notice this by seeing different outputs in the presence and absence of the mutant.

We chose **mutation testing** as our testing strategy, with the aim to kill the mutants **strongly**.

The tools that we used for mutation testing are as follows:

1. VS CODE: Visual Studio Code is a code editor redefined and optimized for building and debugging web and cloud applications.
2.
3. PIT Mutation Testing Tool: An easy-to-use mutation testing tool, that works for Java. We used the org.pitest plugin for VS Code to integrate this tool into the VS Code.

MUTATIONS USED

PIT, by default provides a set of mutation operators. These operators are listed below:

BOOLEAN_FALSE_RETURN	INCREMENTS_MUTATOR
BOOLEAN_TRUE_RETURN	INVERT_NEGS_MUTATOR
CONDITIONALS_BOUNDARY_MUTATOR	MATH_MUTATOR
EMPTY_RETURN_VALUES	NEGATE_CONDITIONALS_MUTATOR
PRIMITIVE_RETURN_VALS_MUTATOR	VOID_METHOD_CALL_MUTATOR
NULL_RETURN_VALUES	

TESTING SUMMARIES

ALL SECTIONS:

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
37	97%	1011/1046	87%	850/982	88%	850/964

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
com.example.SortingAlgos	4	100%	88/88	84%	73/87	84%	73/87
com.example.array	8	95%	241/254	82%	222/270	85%	222/262
com.example.binarySearch	1	100%	12/12	92%	11/12	92%	11/12
com.example.dynamicProgramming	4	95%	145/152	88%	176/201	88%	176/199
com.example.graph	8	98%	273/279	86%	158/184	86%	158/183
com.example.string	8	96%	118/123	94%	101/107	96%	101/105
com.example.trie	4	97%	134/138	90%	109/121	94%	109/116

Report generated by PIT 1.15.0

Enhanced functionality available at arcmutate.com

1. *DYNAMIC PROGRAMMING*

Pit Test Coverage Report

Package Summary

com.example.dynamicProgramming

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
4	95%	145/152	88%	176/201	88%	176/199

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
KnapSack.java	97%	37/38	93%	70/75	95%	70/74
LIS.java	100%	48/48	82%	32/39	82%	32/39
LongestValidParentheses.java	100%	18/18	96%	25/26	96%	25/26
TargetSum.java	88%	42/48	80%	49/61	82%	49/60

Report generated by PIT 1.15.0

2. *GRAPH*

Pit Test Coverage Report

Package Summary

com.example.graph

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
8	98%	273/279	86%	158/184	86%	158/183

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
AccountsMerge.java	93%	53/57	80%	20/25	80%	20/25
Bipartite.java	100%	24/24	90%	19/21	90%	19/21
CriticalConnections.java	100%	42/42	100%	19/19	100%	19/19
DetectCycle.java	100%	41/41	92%	23/25	92%	23/25
Kruskal.java	97%	35/36	79%	23/29	79%	23/29
NoOfProvinces.java	100%	23/23	94%	15/16	94%	15/16
RemoveStone.java	100%	26/26	86%	12/14	86%	12/14
ShortestPathBinaryMatrix.java	97%	29/30	77%	27/35	79%	27/34

Report generated by PIT 1.15.0

3. *STRING*

Pit Test Coverage Report

Package Summary

com.example.string

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
8	96%	118/123	94%	101/107	96%	101/105

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
CheckKAnagrams.java	100%	15/15	89%	16/18	89%	16/18
CountDistinctSubsequences.java	100%	19/19	92%	12/13	92%	12/13
IsValidIP.java	83%	19/23	88%	14/16	93%	14/15
Isomorphic.java	93%	13/14	88%	7/8	100%	7/7
LeftRotateByK.java	100%	12/12	100%	5/5	100%	5/5
LongestCommonPrefix.java	100%	9/9	100%	6/6	100%	6/6
MultiplyString.java	100%	22/22	100%	31/31	100%	31/31
Pangram.java	100%	9/9	100%	10/10	100%	10/10

Report generated by PIT 1.15.0

4. *TRIE*

Pit Test Coverage Report

Package Summary

com.example.trie

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
4	97%	134/138	90%	109/121	94%	109/116

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
ImplementTrie.java	96%	48/50	94%	29/31	94%	29/31
MaximumXorOfTwo.java	100%	34/34	100%	25/25	100%	25/25
NumberofDistinctSubstring.java	100%	12/12	89%	8/9	89%	8/9
WordSearch.java	95%	40/42	84%	47/56	92%	47/51

Report generated by PIT 1.15.0

5 SORT

Pit Test Coverage Report

Package Summary

com.example.SortingAlgos

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
4	100%	88/88	84%	73/87	84%	73/87

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
BubbleSort.java	100%	14/14	79%	11/14	79%	11/14
HeapSort.java	100%	23/23	79%	19/24	79%	19/24
MergeSort.java	100%	26/26	91%	21/23	91%	21/23
QuickSort.java	100%	25/25	85%	22/26	85%	22/26

Report generated by PIT 1.15.0

REFERENCES

1. Mutation Testing Theory:
   a. https://www.geeksforgeeks.org/software-testing-mutation-testing/

2. Code References: Geeks for Geeks

3. JUnit Assert: https://junit.org/junit4/javadoc/4.13/org/junit/Assert.html

4. PITest: https://medium.com/geekculture/mutation-testing-for-maven-project-using-pitest-f9b8fef03a05