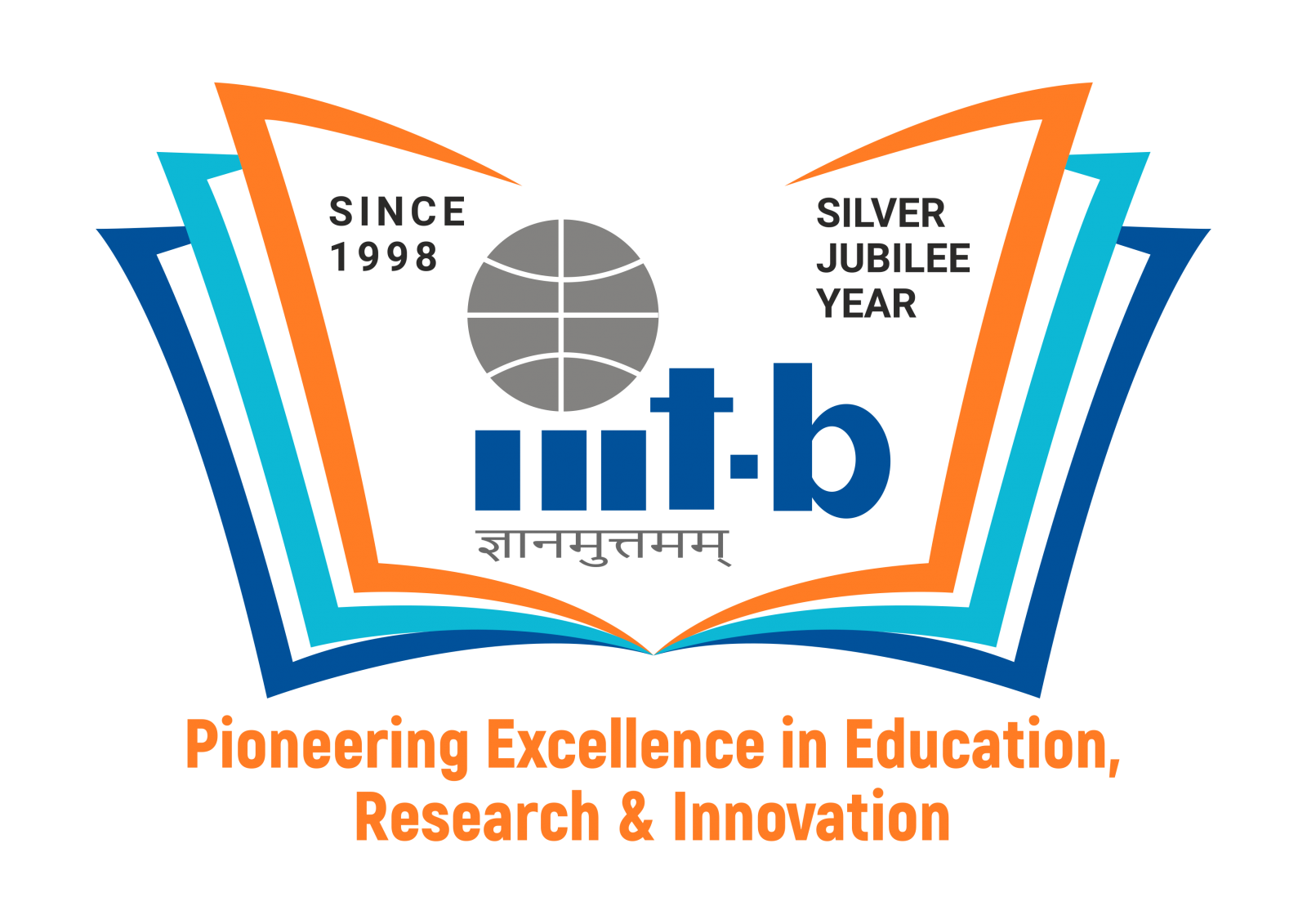
**CS731 SOFTWARE TESTING**

**PROJECT REPORT**

****

**MUTATION TESTING**

**ON**

**SOURCE CODE OF STANDARD ALGORITHMS**

**SUBMITTED BY:**

**ABHIRUP CHAKRABORTY (MT2022003)**

**ANUJKUMAR JAIN (MT2022017)**

*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/AbhirupChakraborty/Mutation-Testing-On-Standard-Algorithms/](https://github.com/AbhirupChakraborty/Mutation-Testing-On-Standard-Algorithms)

*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](https://code.visualstudio.com/): Visual Studio Code is a code editor redefined and optimized for building and debugging web and cloud applications.
2. [PIT Mutation Testing Tool](https://pitest.org/): 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:

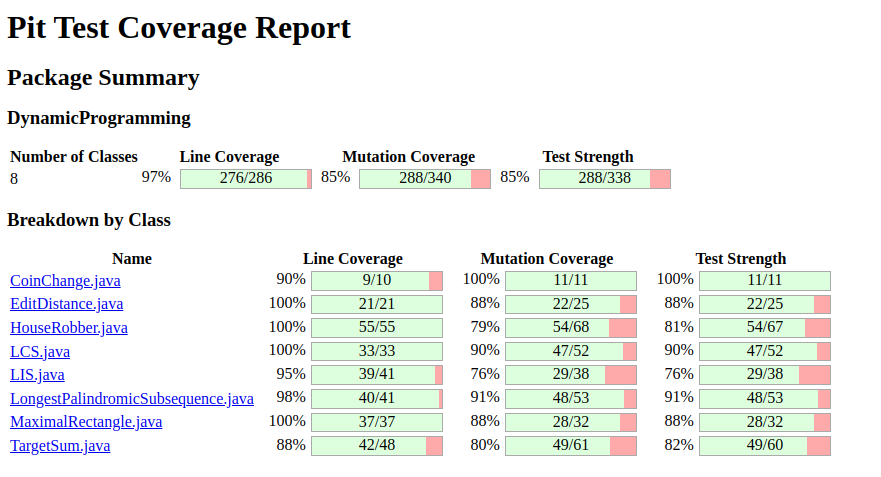
|  |  |
| --- | --- |
| BOOLEAN\_FALSE\_RETURN | INCREMENTS\_MUTATOR |
| BOOLEAN\_TRUE\_RETURN | INVERT\_NEGS\_MUTATOR |
| CONDITIONALS\_BOUNDARY\_MUTATOR | MATH\_MUTATOR |
| EMPTY\_RETURN\_VALUES | NEGATE\_CONDITIONALS\_MUTATOR |
| PRIMITIVE\_RETURN\_VALS\_MUTATOR | VOID\_METHOD\_CALL\_MUTATOR |
| NULL\_RETURN\_VALUES | |

*Tabulated Testing Results*

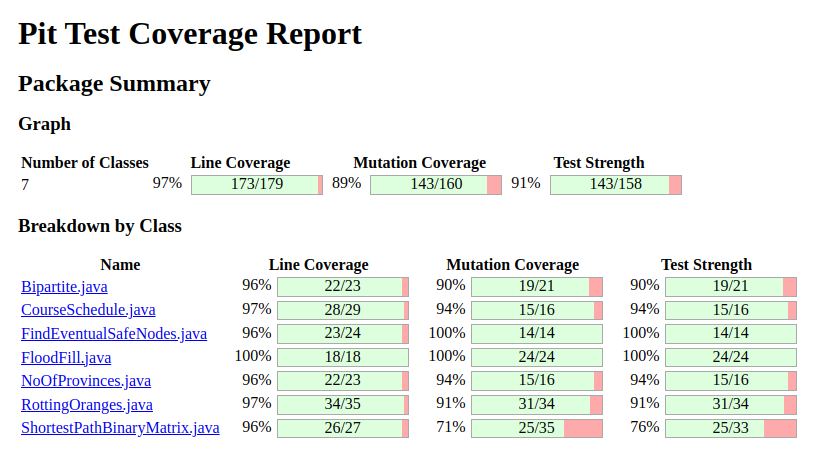
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Contributor** | **Algorithm  Domain** | **Algorithm Name** | **Mutations  Killed** | **%  Mutations  Killed** | **Lines of Code  in Algorithm  (approx.) \*\*** |
| Abhirup  (MT2022003) | Dynamic  Programming | [Coin Change](https://github.com/TheAlgorithms/Java/blob/master/src/main/java/com/thealgorithms/dynamicprogramming/CoinChange.java) | 11/11 | 100 | 10 |
| [Edit Distance](https://github.com/TheAlgorithms/Java/blob/master/src/main/java/com/thealgorithms/dynamicprogramming/EditDistance.java) | 22/25 | 88 | 21 |
| [H](https://github.com/TheAlgorithms/Java/blob/master/src/main/java/com/thealgorithms/dynamicprogramming/Fibonacci.java)ouse Robber | 54/68 | 79 | 55 |
| [Longest Common Subsequence](https://github.com/TheAlgorithms/Java/blob/master/src/main/java/com/thealgorithms/dynamicprogramming/LongestCommonSubsequence.java) | 47/52 | 90 | 33 |
| [Longest Increasing Subsequence](https://github.com/TheAlgorithms/Java/blob/master/src/main/java/com/thealgorithms/dynamicprogramming/LongestIncreasingSubsequence.java) | 29/38 | 76 | 41 |
| [Longest Palindromic Subsequence](https://github.com/TheAlgorithms/Java/blob/master/src/main/java/com/thealgorithms/dynamicprogramming/LongestPalindromicSubsequence.java) | 48/53 | 91 | 41 |
| Maximal Rectangle | 28/32 | 88 | 48 |
| [T](https://github.com/TheAlgorithms/Java/blob/master/src/main/java/com/thealgorithms/dynamicprogramming/LongestValidParentheses.java)arget Sum | 49/61 | 80 | 35 |
| Abhirup  (MT2022003) | Binary Search | [Binary Search](https://www.geeksforgeeks.org/binary-search/) | 12/13 | 92 | 12 |
| [F](https://www.geeksforgeeks.org/closest-pair-of-points-using-divide-and-conquer-algorithm/)irst Position Finder | 12/13 | 92 | 13 |
| [L](https://www.geeksforgeeks.org/rabin-karp-algorithm-for-pattern-searching/)ast Position Finder | 12/13 | 92 | 13 |
| [M](https://www.geeksforgeeks.org/strassens-matrix-multiplication/)edian Of Two Sorted Array | 58/77 | 75 | 59 |
| [Binary Search in 2D Array](https://www.geeksforgeeks.org/search-element-sorted-matrix/) | 36/44 | 82 | 33 |
| Abhirup  (MT2022003) | Graph | [B](https://www.geeksforgeeks.org/floor-and-ceil-from-a-bst/)ipartite | 19/21 | 90 | 23 |
| [C](https://www.geeksforgeeks.org/symmetric-tree-tree-which-is-mirror-image-of-itself/)ourse Schedule | 15/16 | 94 | 29 |
| [F](https://www.geeksforgeeks.org/k-dimensional-tree/)ind Eventual Safe Nodes | 14/14 | 100 | 24 |
| [F](https://www.geeksforgeeks.org/lazy-propagation-in-segment-tree/)lood Fill | 24/24 | 100 | 18 |
| No of provinces | 15/16 | 94 | 23 |
| Rotting Oranges | 31/34 | 91 | 35 |
| Shortest Path Binary Matrix | 25/35 | 71 | 27 |
| Anuj Jain  (MT2022017 | String Algo. | [Letter Combinations of  Phone Number](https://www.geeksforgeeks.org/iterative-letter-combinations-of-a-phone-number/) | 6/6 | 100 | 21 |
| [L](https://www.geeksforgeeks.org/java-program-to-write-your-own-atoi/)ongest Non Repeating Substring | 12/19 | 63 | 22 |
| [S](https://www.geeksforgeeks.org/java-program-to-write-your-own-atoi/)tring Compression | 21/23 | 91 | 23 |
| [V](https://www.geeksforgeeks.org/java-program-to-write-your-own-atoi/)alid Parentheses | 13/15 | 87 | 11 |
| [W](https://www.geeksforgeeks.org/c-program-check-given-string-palindrome/)ord Ladder | 14/15 | 93 | 28 |
| Anuj Jain  (MT2022017) | Sliding Window | Character Replacement | 42/44 | 95 | 31 |
| [C](https://www.geeksforgeeks.org/hamiltonian-cycle-backtracking-6/)ount of Nice SubArray | 36/38 | 95 | 31 |
| Length of Longest Substring | 25/26 | 96 | 36 |
| [M](https://www.geeksforgeeks.org/hamiltonian-cycle-backtracking-6/)ax Consecutive Ones | 22/24 | 92 | 31 |
| [S](https://www.geeksforgeeks.org/bipartite-graph/)ubstring Containing Three | 21/22 | 95 | 23 |

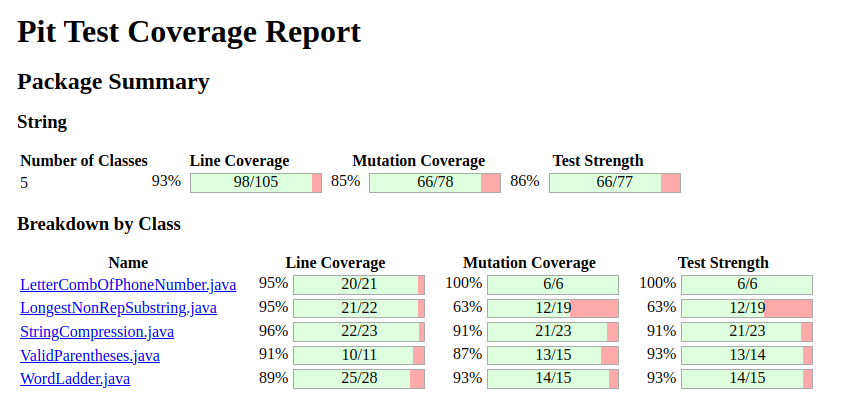
*Testing Summaries*

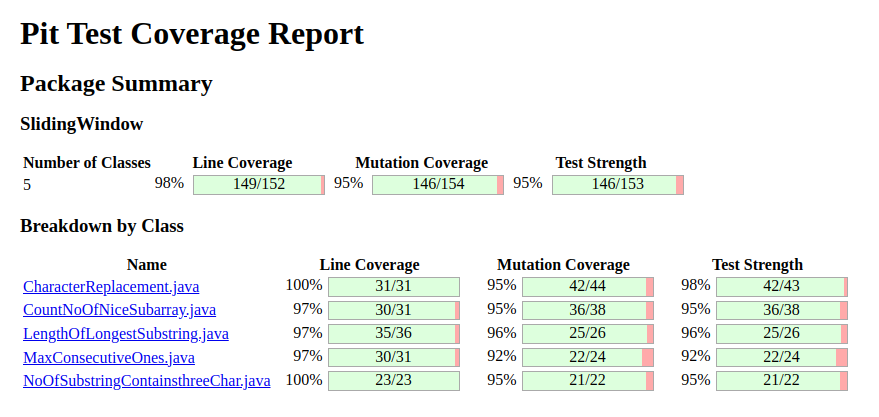
1. *Dynamic Programming*



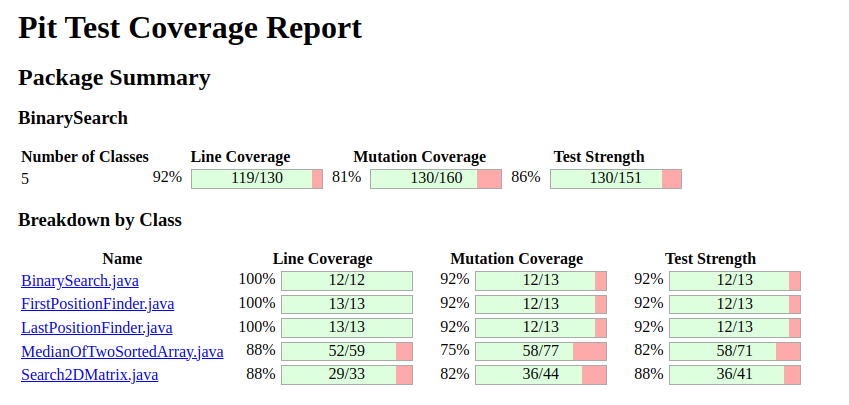
1. *Graph*

**

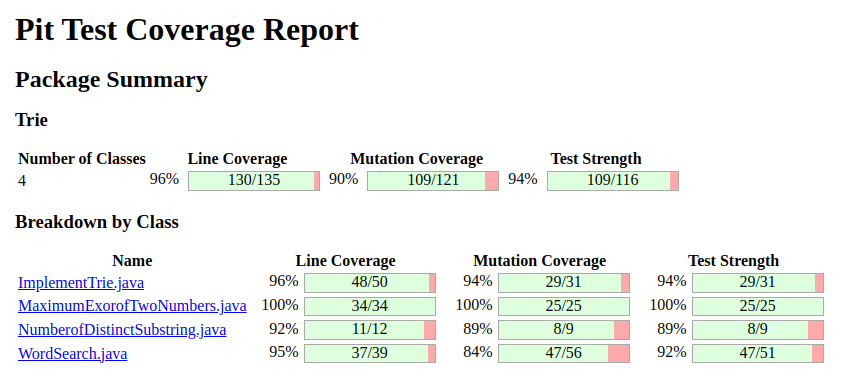
1. *Strings*
2. *Sliding Window*



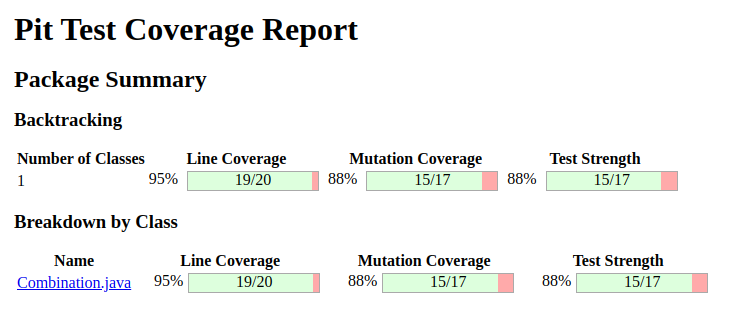
1. *BINARY SEARCH*

**

1. *TRIE*



1. *Backtracking*



*References*

1. Mutation Testing Theory:
   1. <https://www.geeksforgeeks.org/software-testing-mutation-testing/>
   2. <https://www.guru99.com/mutation-testing.html>
2. Mutation Testing Tutorial:
   1. https://www.youtube.com/watch?v=fiVma2syvoo
   2. https://www.youtube.com/watch?v=wZeZMtqVmck&t=256s
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>