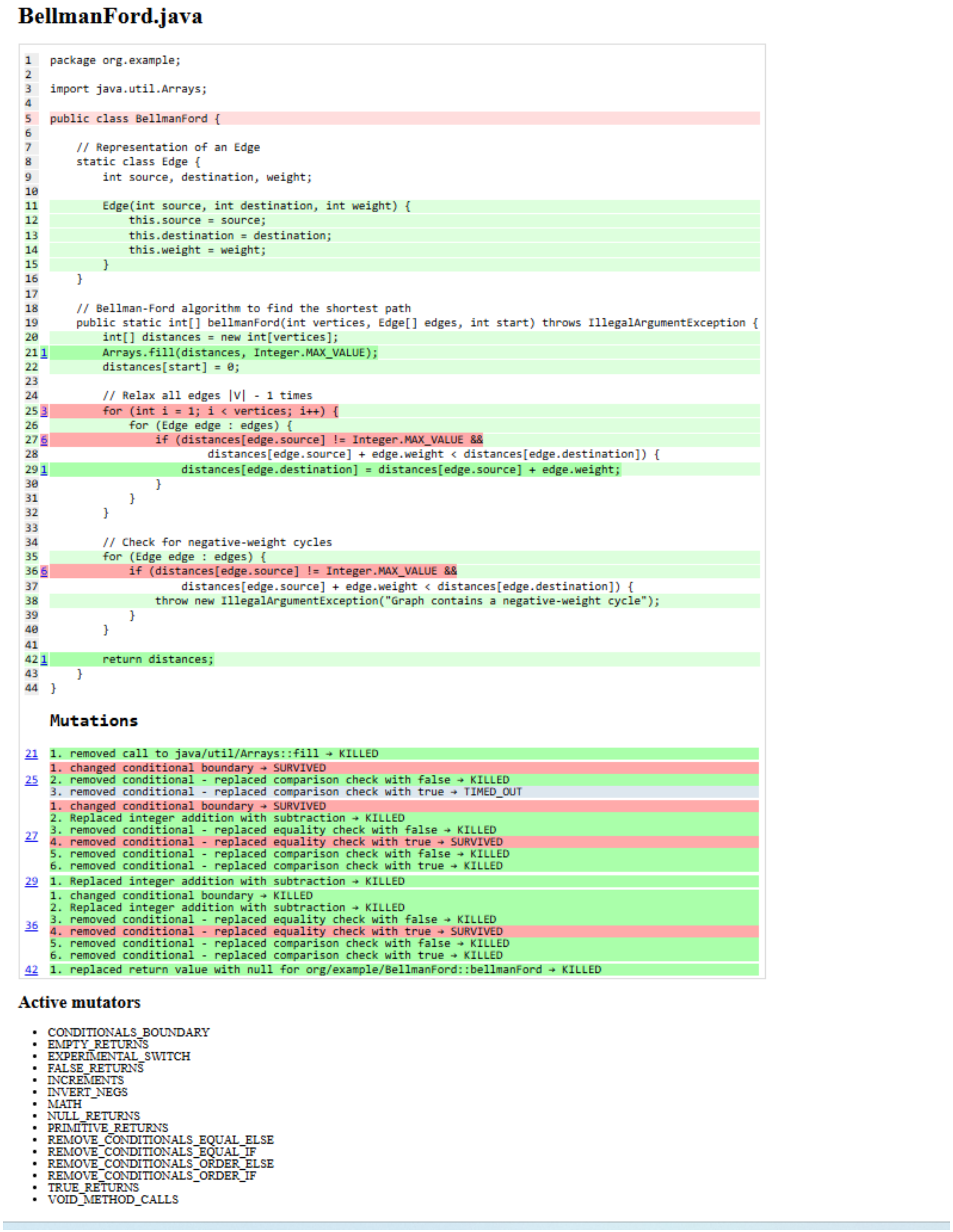
Mutation Testing Reports for all algorithms.

1.Bellman Ford Algorithm.



BellmanFord.java

```
1    package org.example;
2
3    import java.util.Arrays;
4
5    public class BellmanFord {
6
7        // Representation of an Edge
8        static class Edge {
9            int source, destination, weight;
10
11           Edge(int source, int destination, int weight) {
12               this.source = source;
13               this.destination = destination;
14               this.weight = weight;
15           }
16       }
17
18       // Bellman-Ford algorithm to find the shortest path
19       public static int[] bellmanFord(int vertices, Edge[] edges, int start) throws IllegalArgumentException {
20           int[] distances = new int[vertices];
21 1         Arrays.fill(distances, Integer.MAX_VALUE);
22           distances[start] = 0;
23
24           // Relax all edges |V| - 1 times
25 3         for (int i = 1; i < vertices; i++) {
26               for (Edge edge : edges) {
27 6                 if (distances[edge.source] != Integer.MAX_VALUE &&
28                           distances[edge.source] + edge.weight < distances[edge.destination]) {
29 1                     distances[edge.destination] = distances[edge.source] + edge.weight;
30                   }
31               }
32           }
33
34           // Check for negative-weight cycles
35           for (Edge edge : edges) {
36 6             if (distances[edge.source] != Integer.MAX_VALUE &&
37                       distances[edge.source] + edge.weight < distances[edge.destination]) {
38                   throw new IllegalArgumentException("Graph contains a negative-weight cycle");
39               }
40           }
41
42 1         return distances;
43       }
44   }
```

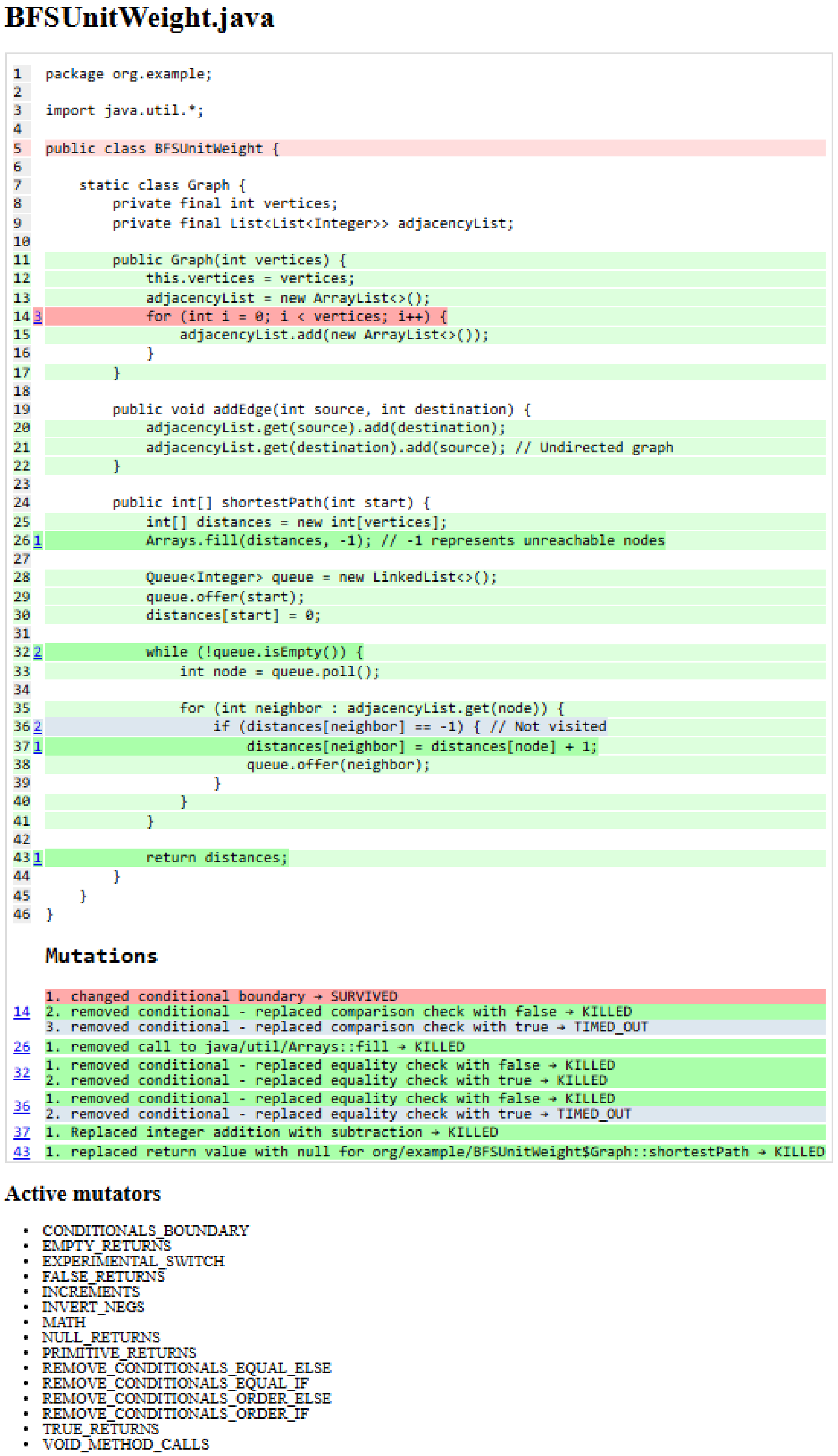
Mutations

```
21  1. removed call to java/util/Arrays::fill → KILLED
    1. changed conditional boundary → SURVIVED
25  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → TIMED_OUT
    1. changed conditional boundary → SURVIVED
    2. Replaced integer addition with subtraction → KILLED
    3. removed conditional - replaced equality check with false → KILLED
27  4. removed conditional - replaced equality check with true → SURVIVED
    5. removed conditional - replaced comparison check with false → KILLED
    6. removed conditional - replaced comparison check with true → KILLED
29  1. Replaced integer addition with subtraction → KILLED
    1. changed conditional boundary → KILLED
    2. Replaced integer addition with subtraction → KILLED
    3. removed conditional - replaced equality check with false → KILLED
36  4. removed conditional - replaced equality check with true → SURVIVED
    5. removed conditional - replaced comparison check with false → KILLED
    6. removed conditional - replaced comparison check with true → KILLED
42  1. replaced return value with null for org/example/BellmanFord::bellmanFord → KILLED
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- EXPERIMENTAL_SWITCH
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_EQUAL_IF
- REMOVE_CONDITIONALS_ORDER_ELSE
- REMOVE_CONDITIONALS_ORDER_IF
- TRUE_RETURNS
- VOID_METHOD_CALLS

2.BFSWithUnitWeight



BFSUnitWeight.java

```
1     package org.example;
2
3     import java.util.*;
4
5     public class BFSUnitWeight {
6
7         static class Graph {
8             private final int vertices;
9             private final List<List<Integer>> adjacencyList;
10
11            public Graph(int vertices) {
12                this.vertices = vertices;
13                adjacencyList = new ArrayList<>();
14 3              for (int i = 0; i < vertices; i++) {
15                    adjacencyList.add(new ArrayList<>());
16                }
17            }
18
19            public void addEdge(int source, int destination) {
20                adjacencyList.get(source).add(destination);
21                adjacencyList.get(destination).add(source); // Undirected graph
22            }
23
24            public int[] shortestPath(int start) {
25                int[] distances = new int[vertices];
26 1              Arrays.fill(distances, -1); // -1 represents unreachable nodes
27
28                Queue<Integer> queue = new LinkedList<>();
29                queue.offer(start);
30                distances[start] = 0;
31
32 2              while (!queue.isEmpty()) {
33                    int node = queue.poll();
34
35                    for (int neighbor : adjacencyList.get(node)) {
36 2                      if (distances[neighbor] == -1) { // Not visited
37 1                          distances[neighbor] = distances[node] + 1;
38                            queue.offer(neighbor);
39                        }
40                    }
41                }
42
43 1              return distances;
44            }
45        }
46    }
```

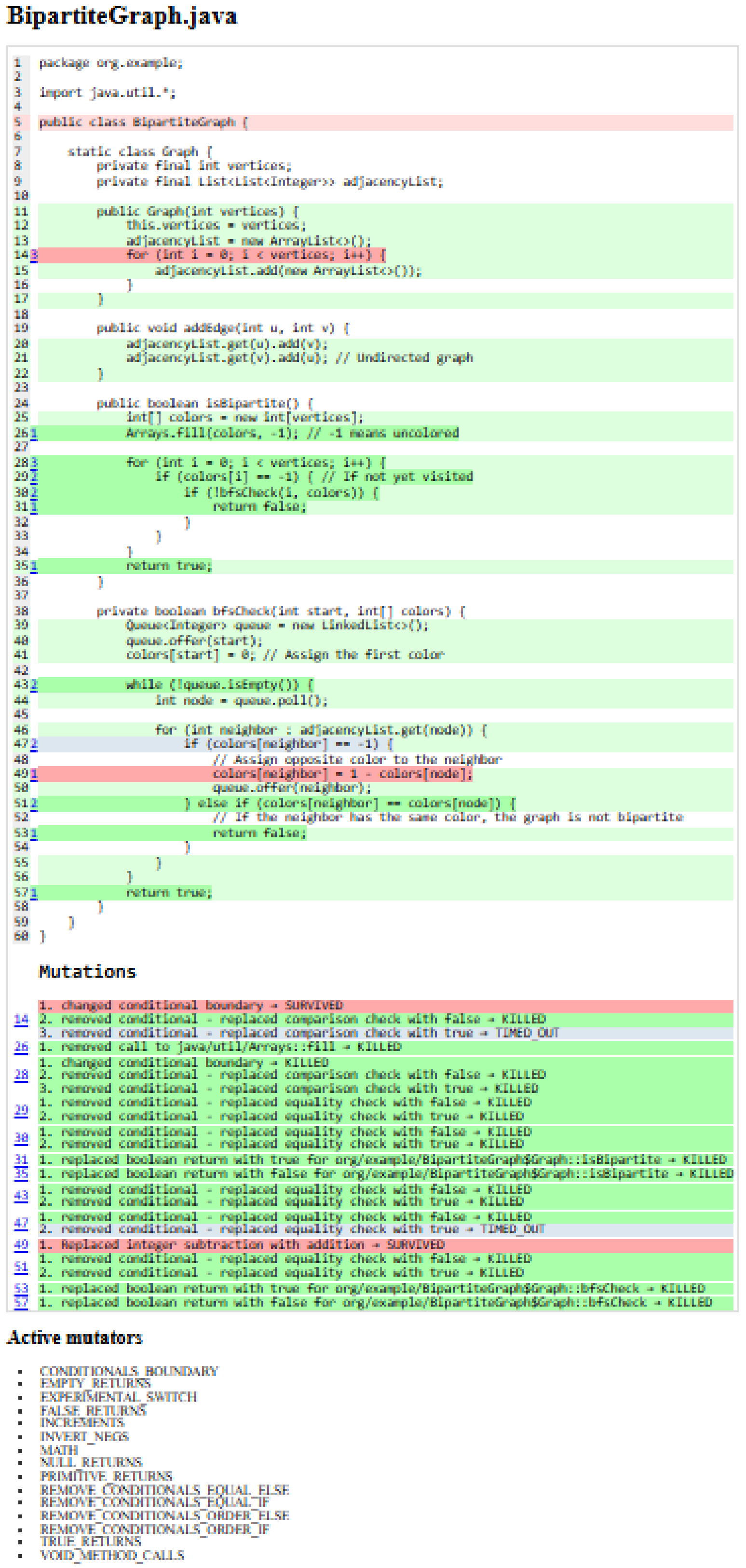
Mutations

```
    1. changed conditional boundary → SURVIVED
14  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → TIMED_OUT
26  1. removed call to java/util/Arrays::fill → KILLED
32  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
36  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → TIMED_OUT
37  1. Replaced integer addition with subtraction → KILLED
43  1. replaced return value with null for org/example/BFSUnitWeight$Graph::shortestPath → KILLED
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- EXPERIMENTAL_SWITCH
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_EQUAL_IF
- REMOVE_CONDITIONALS_ORDER_ELSE
- REMOVE_CONDITIONALS_ORDER_IF
- TRUE_RETURNS
- VOID_METHOD_CALLS

3.Bipartite Graph



BipartiteGraph.java

```
1    package org.example;
2
3    import java.util.*;
4
5    public class BipartiteGraph {
6
7        static class Graph {
8            private final int vertices;
9            private final List<List<Integer>> adjacencyList;
10
11           public Graph(int vertices) {
12               this.vertices = vertices;
13               adjacencyList = new ArrayList<>();
14 3             for (int i = 0; i < vertices; i++) {
15                   adjacencyList.add(new ArrayList<>());
16               }
17           }
18
19           public void addEdge(int u, int v) {
20               adjacencyList.get(u).add(v);
21               adjacencyList.get(v).add(u); // Undirected graph
22           }
23
24           public boolean isBipartite() {
25               int[] colors = new int[vertices];
26 1             Arrays.fill(colors, -1); // -1 means uncolored
27
28 3             for (int i = 0; i < vertices; i++) {
29 2                 if (colors[i] == -1) { // If not yet visited
30 2                     if (!bfsCheck(i, colors)) {
31 1                         return false;
32                       }
33                   }
34               }
35 1             return true;
36           }
37
38           private boolean bfsCheck(int start, int[] colors) {
39               Queue<Integer> queue = new LinkedList<>();
40               queue.offer(start);
41               colors[start] = 0; // Assign the first color
42
43 2             while (!queue.isEmpty()) {
44                   int node = queue.poll();
45
46                   for (int neighbor : adjacencyList.get(node)) {
47 2                     if (colors[neighbor] == -1) {
48                           // Assign opposite color to the neighbor
49 1                         colors[neighbor] = 1 - colors[node];
50                           queue.offer(neighbor);
51 2                     } else if (colors[neighbor] == colors[node]) {
52                           // If the neighbor has the same color, the graph is not bipartite
53 1                         return false;
54                       }
55                   }
56               }
57 1             return true;
58           }
59       }
60   }
```

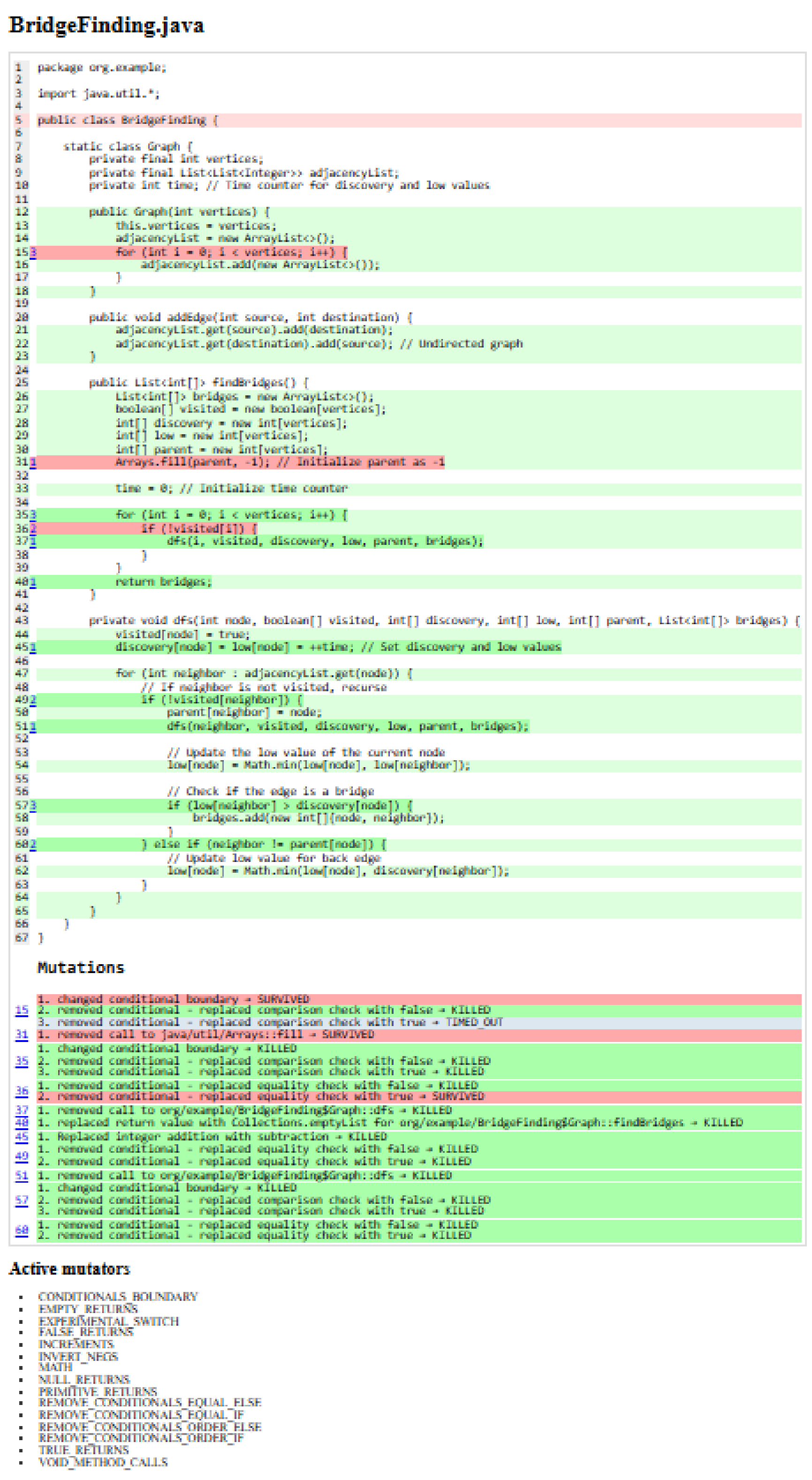
Mutations

```
    1. changed conditional boundary → SURVIVED
14  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → TIMED_OUT
26  1. removed call to java/util/Arrays::fill → KILLED
    1. changed conditional boundary → KILLED
28  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → KILLED
29  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
30  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
31  1. replaced boolean return with true for org/example/BipartiteGraph$Graph::isBipartite → KILLED
35  1. replaced boolean return with false for org/example/BipartiteGraph$Graph::isBipartite → KILLED
43  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
47  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → TIMED_OUT
49  1. Replaced integer subtraction with addition → SURVIVED
51  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
53  1. replaced boolean return with true for org/example/BipartiteGraph$Graph::bfsCheck → KILLED
57  1. replaced boolean return with false for org/example/BipartiteGraph$Graph::bfsCheck → KILLED
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- EXPERIMENTAL_SWITCH
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_EQUAL_IF
- REMOVE_CONDITIONALS_ORDER_ELSE
- REMOVE_CONDITIONALS_ORDER_IF
- TRUE_RETURNS
- VOID_METHOD_CALLS

4.BridgeFindingAlgorithm.



BridgeFinding.java

```
1    package org.example;
2
3    import java.util.*;
4
5    public class BridgeFinding {
6
7        static class Graph {
8            private final int vertices;
9            private final List<List<Integer>> adjacencyList;
10           private int time; // Time counter for discovery and low values
11
12           public Graph(int vertices) {
13               this.vertices = vertices;
14               adjacencyList = new ArrayList<>();
15 3             for (int i = 0; i < vertices; i++) {
16                   adjacencyList.add(new ArrayList<>());
17               }
18           }
19
20           public void addEdge(int source, int destination) {
21               adjacencyList.get(source).add(destination);
22               adjacencyList.get(destination).add(source); // Undirected graph
23           }
24
25           public List<int[]> findBridges() {
26               List<int[]> bridges = new ArrayList<>();
27               boolean[] visited = new boolean[vertices];
28               int[] discovery = new int[vertices];
29               int[] low = new int[vertices];
30               int[] parent = new int[vertices];
31 1             Arrays.fill(parent, -1); // Initialize parent as -1
32
33               time = 0; // Initialize time counter
34
35 3             for (int i = 0; i < vertices; i++) {
36 2                 if (!visited[i]) {
37 1                     dfs(i, visited, discovery, low, parent, bridges);
38                   }
39               }
40 1             return bridges;
41           }
42
43           private void dfs(int node, boolean[] visited, int[] discovery, int[] low, int[] parent, List<int[]> bridges) {
44               visited[node] = true;
45 1             discovery[node] = low[node] = ++time; // Set discovery and low values
46
47               for (int neighbor : adjacencyList.get(node)) {
48                   // If neighbor is not visited, recurse
49 2                 if (!visited[neighbor]) {
50                       parent[neighbor] = node;
51 1                     dfs(neighbor, visited, discovery, low, parent, bridges);
52
53                       // Update the low value of the current node
54                       low[node] = Math.min(low[node], low[neighbor]);
55
56                       // Check if the edge is a bridge
57 3                     if (low[neighbor] > discovery[node]) {
58                           bridges.add(new int[]{node, neighbor});
59                       }
60 2                 } else if (neighbor != parent[node]) {
61                       // Update low value for back edge
62                       low[node] = Math.min(low[node], discovery[neighbor]);
63                   }
64               }
65           }
66       }
67   }
```

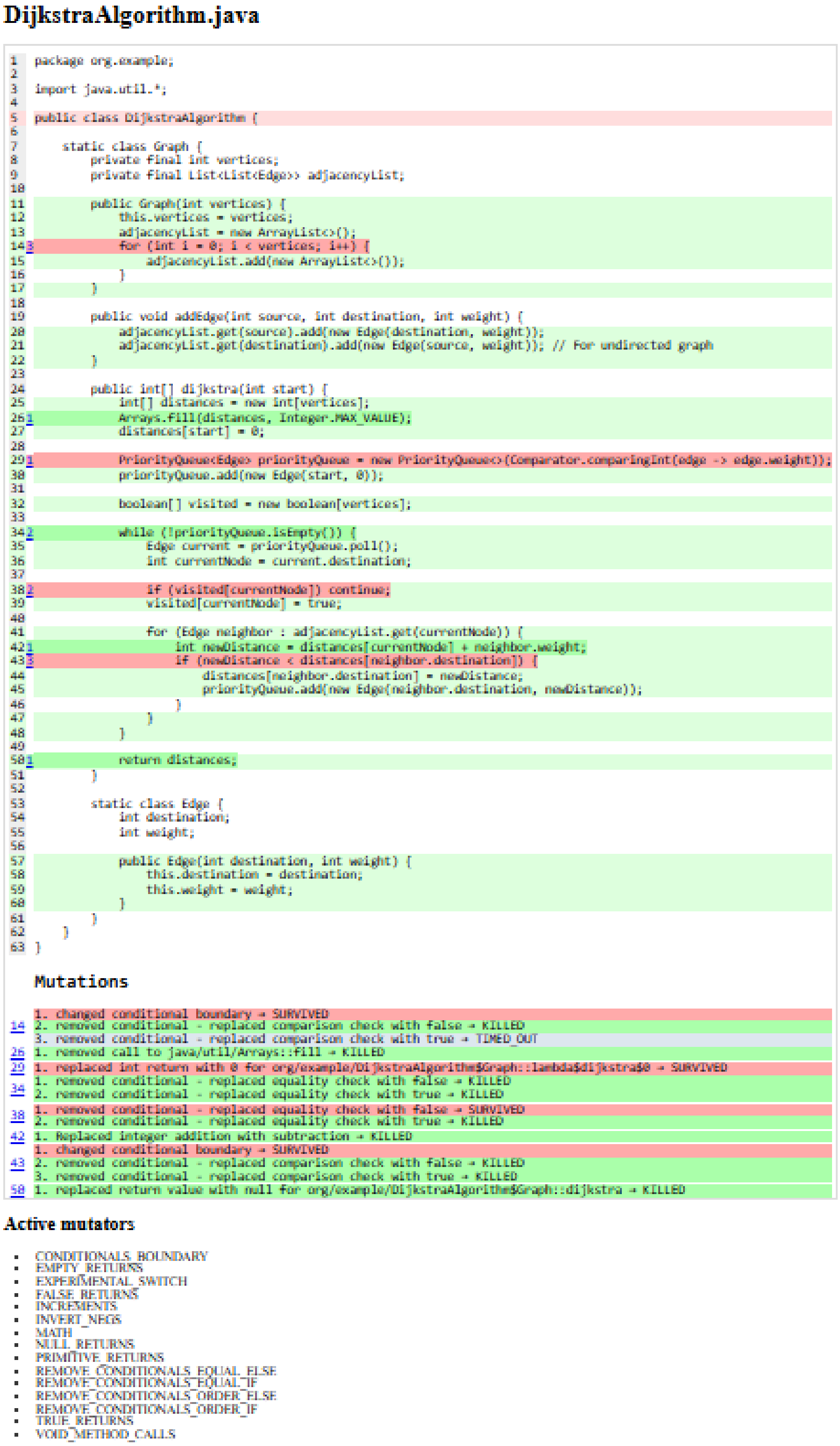
Mutations

```
     1. changed conditional boundary → SURVIVED
15   2. removed conditional - replaced comparison check with false → KILLED
     3. removed conditional - replaced comparison check with true → TIMED_OUT
31   1. removed call to java/util/Arrays::fill → SURVIVED
     1. changed conditional boundary → KILLED
35   2. removed conditional - replaced comparison check with false → KILLED
     3. removed conditional - replaced comparison check with true → KILLED
36   1. removed conditional - replaced equality check with false → KILLED
     2. removed conditional - replaced equality check with true → SURVIVED
37   1. removed call to org/example/BridgeFinding$Graph::dfs → KILLED
40   1. replaced return value with Collections.emptyList for org/example/BridgeFinding$Graph::findBridges → KILLED
45   1. Replaced integer addition with subtraction → KILLED
49   1. removed conditional - replaced equality check with false → KILLED
     2. removed conditional - replaced equality check with true → KILLED
51   1. removed call to org/example/BridgeFinding$Graph::dfs → KILLED
     1. changed conditional boundary → KILLED
57   2. removed conditional - replaced comparison check with false → KILLED
     3. removed conditional - replaced comparison check with true → KILLED
60   1. removed conditional - replaced equality check with false → KILLED
     2. removed conditional - replaced equality check with true → KILLED
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- EXPERIMENTAL_SWITCH
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_EQUAL_IF
- REMOVE_CONDITIONALS_ORDER_ELSE
- REMOVE_CONDITIONALS_ORDER_IF
- TRUE_RETURNS
- VOID_METHOD_CALLS

5.Dijkstra’s Algorithm



DijkstraAlgorithm.java

```
1    package org.example;
2
3    import java.util.*;
4
5    public class DijkstraAlgorithm {
6
7        static class Graph {
8            private final int vertices;
9            private final List<List<Edge>> adjacencyList;
10
11           public Graph(int vertices) {
12               this.vertices = vertices;
13               adjacencyList = new ArrayList<>();
14 3             for (int i = 0; i < vertices; i++) {
15                   adjacencyList.add(new ArrayList<>());
16               }
17           }
18
19           public void addEdge(int source, int destination, int weight) {
20               adjacencyList.get(source).add(new Edge(destination, weight));
21               adjacencyList.get(destination).add(new Edge(source, weight)); // For undirected graph
22           }
23
24           public int[] dijkstra(int start) {
25               int[] distances = new int[vertices];
26 1             Arrays.fill(distances, Integer.MAX_VALUE);
27               distances[start] = 0;
28
29 1             PriorityQueue<Edge> priorityQueue = new PriorityQueue<>(Comparator.comparingInt(edge -> edge.weight));
30               priorityQueue.add(new Edge(start, 0));
31
32               boolean[] visited = new boolean[vertices];
33
34 2             while (!priorityQueue.isEmpty()) {
35                   Edge current = priorityQueue.poll();
36                   int currentNode = current.destination;
37
38 2                 if (visited[currentNode]) continue;
39                   visited[currentNode] = true;
40
41                   for (Edge neighbor : adjacencyList.get(currentNode)) {
42 1                     int newDistance = distances[currentNode] + neighbor.weight;
43 3                     if (newDistance < distances[neighbor.destination]) {
44                           distances[neighbor.destination] = newDistance;
45                           priorityQueue.add(new Edge(neighbor.destination, newDistance));
46                       }
47                   }
48               }
49
50 1             return distances;
51           }
52
53           static class Edge {
54               int destination;
55               int weight;
56
57               public Edge(int destination, int weight) {
58                   this.destination = destination;
59                   this.weight = weight;
60               }
61           }
62       }
63   }
```

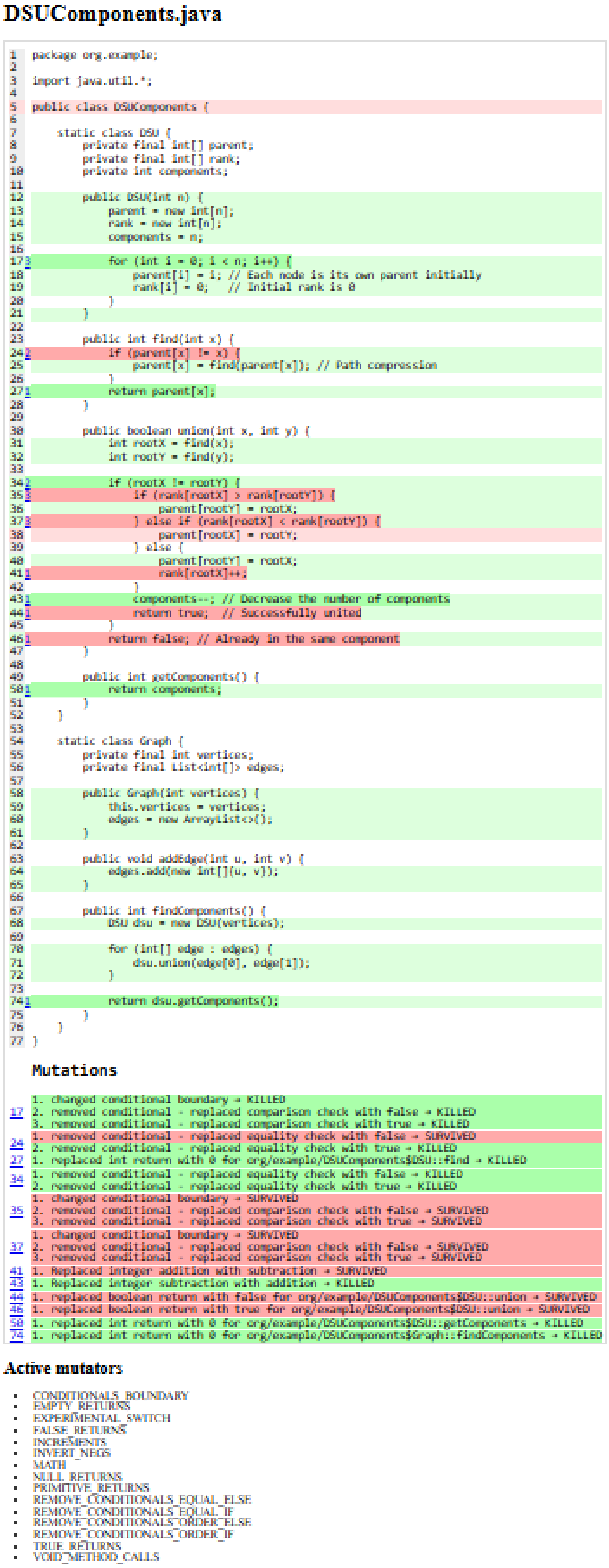
Mutations

```
    1. changed conditional boundary → SURVIVED
14  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → TIMED_OUT
26  1. removed call to java/util/Arrays::fill → KILLED
29  1. replaced int return with 0 for org/example/DijkstraAlgorithm$Graph::lambda$dijkstra$0 → SURVIVED
34  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
38  1. removed conditional - replaced equality check with false → SURVIVED
    2. removed conditional - replaced equality check with true → KILLED
42  1. Replaced integer addition with subtraction → KILLED
    1. changed conditional boundary → SURVIVED
43  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → KILLED
50  1. replaced return value with null for org/example/DijkstraAlgorithm$Graph::dijkstra → KILLED
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- EXPERIMENTAL_SWITCH
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_EQUAL_IF
- REMOVE_CONDITIONALS_ORDER_ELSE
- REMOVE_CONDITIONALS_ORDER_IF
- TRUE_RETURNS
- VOID_METHOD_CALLS

6.DSUComponents.



DSUComponents.java

```
1     package org.example;
2
3     import java.util.*;
4
5     public class DSUComponents {
6
7         static class DSU {
8             private final int[] parent;
9             private final int[] rank;
10            private int components;
11
12            public DSU(int n) {
13                parent = new int[n];
14                rank = new int[n];
15                components = n;
16
17 3              for (int i = 0; i < n; i++) {
18                    parent[i] = i; // Each node is its own parent initially
19                    rank[i] = 0;   // Initial rank is 0
20                }
21            }
22
23            public int find(int x) {
24 2              if (parent[x] != x) {
25                    parent[x] = find(parent[x]); // Path compression
26                }
27 1              return parent[x];
28            }
29
30            public boolean union(int x, int y) {
31                int rootX = find(x);
32                int rootY = find(y);
33
34 2              if (rootX != rootY) {
35 3                  if (rank[rootX] > rank[rootY]) {
36                        parent[rootY] = rootX;
37 3                  } else if (rank[rootX] < rank[rootY]) {
38                        parent[rootX] = rootY;
39                    } else {
40                        parent[rootY] = rootX;
41 1                      rank[rootX]++;
42                    }
43 1                  components--; // Decrease the number of components
44 1                  return true;  // Successfully united
45                }
46 1              return false; // Already in the same component
47            }
48
49            public int getComponents() {
50 1              return components;
51            }
52        }
53
54        static class Graph {
55            private final int vertices;
56            private final List<int[]> edges;
57
58            public Graph(int vertices) {
59                this.vertices = vertices;
60                edges = new ArrayList<>();
61            }
62
63            public void addEdge(int u, int v) {
64                edges.add(new int[]{u, v});
65            }
66
67            public int findComponents() {
68                DSU dsu = new DSU(vertices);
69
70                for (int[] edge : edges) {
71                    dsu.union(edge[0], edge[1]);
72                }
73
74 1              return dsu.getComponents();
75            }
76        }
77    }
```

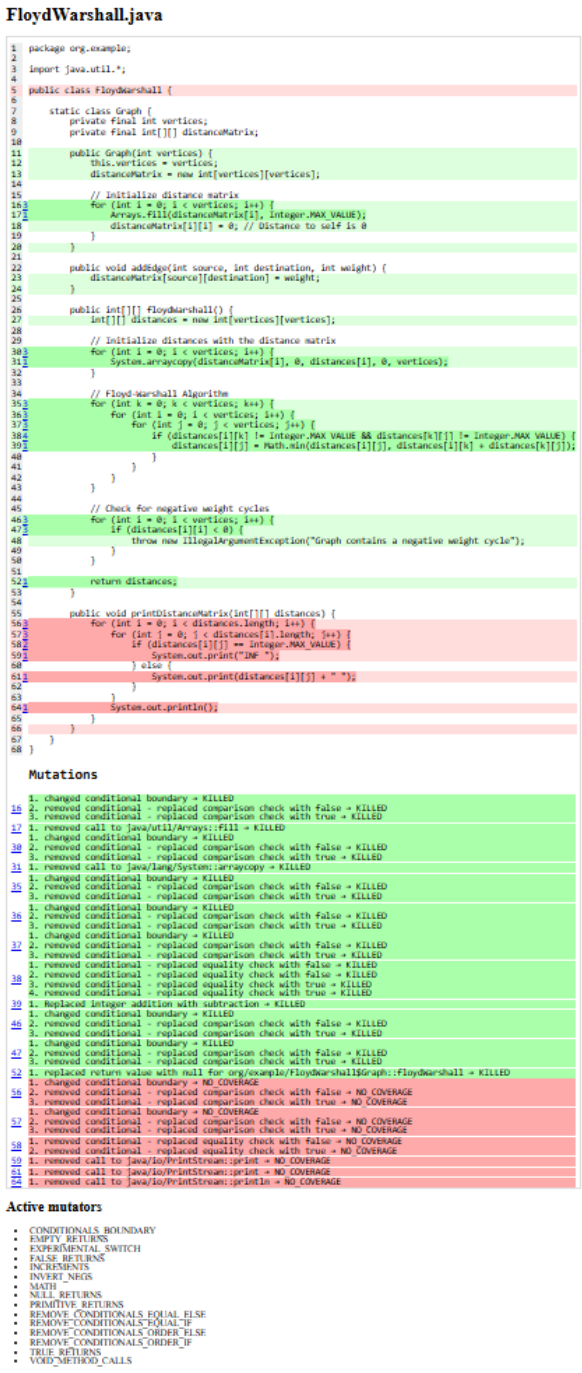
Mutations

```
17  1. changed conditional boundary → KILLED
    2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → KILLED
24  1. removed conditional - replaced equality check with false → SURVIVED
    2. removed conditional - replaced equality check with true → KILLED
27  1. replaced int return with 0 for org/example/DSUComponents$DSU::find → KILLED
34  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
35  1. changed conditional boundary → SURVIVED
    2. removed conditional - replaced comparison check with false → SURVIVED
    3. removed conditional - replaced comparison check with true → SURVIVED
37  1. changed conditional boundary → SURVIVED
    2. removed conditional - replaced comparison check with false → SURVIVED
    3. removed conditional - replaced comparison check with true → SURVIVED
41  1. Replaced integer addition with subtraction → SURVIVED
43  1. Replaced integer subtraction with addition → KILLED
44  1. replaced boolean return with false for org/example/DSUComponents$DSU::union → SURVIVED
46  1. replaced boolean return with true for org/example/DSUComponents$DSU::union → SURVIVED
50  1. replaced int return with 0 for org/example/DSUComponents$DSU::getComponents → KILLED
74  1. replaced int return with 0 for org/example/DSUComponents$Graph::findComponents → KILLED
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- EXPERIMENTAL_SWITCH
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_EQUAL_IF
- REMOVE_CONDITIONALS_ORDER_ELSE
- REMOVE_CONDITIONALS_ORDER_IF
- TRUE_RETURNS
- VOID_METHOD_CALLS

7.FloydWarshall.



FloydWarshall.java

```
1     package org.example;
2
3     import java.util.*;
4
5     public class FloydWarshall {
6
7         static class Graph {
8             private final int vertices;
9             private final int[][] distanceMatrix;
10
11            public Graph(int vertices) {
12                this.vertices = vertices;
13                distanceMatrix = new int[vertices][vertices];
14
15                // Initialize distance matrix
16 3              for (int i = 0; i < vertices; i++) {
17 1                  Arrays.fill(distanceMatrix[i], Integer.MAX_VALUE);
18                    distanceMatrix[i][i] = 0; // Distance to self is 0
19                }
20            }
21
22            public void addEdge(int source, int destination, int weight) {
23                distanceMatrix[source][destination] = weight;
24            }
25
26            public int[][] floydWarshall() {
27                int[][] distances = new int[vertices][vertices];
28
29                // Initialize distances with the distance matrix
30 3              for (int i = 0; i < vertices; i++) {
31 1                  System.arraycopy(distanceMatrix[i], 0, distances[i], 0, vertices);
32                }
33
34                // Floyd-Warshall Algorithm
35 3              for (int k = 0; k < vertices; k++) {
36 3                  for (int i = 0; i < vertices; i++) {
37 3                      for (int j = 0; j < vertices; j++) {
38 4                          if (distances[i][k] != Integer.MAX VALUE && distances[k][j] != Integer.MAX VALUE) {
39 1                              distances[i][j] = Math.min(distances[i][j], distances[i][k] + distances[k][j]);
40                            }
41                        }
42                    }
43                }
44
45                // Check for negative weight cycles
46 3              for (int i = 0; i < vertices; i++) {
47 3                  if (distances[i][i] < 0) {
48                        throw new IllegalArgumentException("Graph contains a negative weight cycle");
49                    }
50                }
51
52 1              return distances;
53            }
54
55            public void printDistanceMatrix(int[][] distances) {
56 3              for (int i = 0; i < distances.length; i++) {
57 3                  for (int j = 0; j < distances[i].length; j++) {
58 2                      if (distances[i][j] == Integer.MAX_VALUE) {
59 1                          System.out.print("INF ");
60                        } else {
61 1                          System.out.print(distances[i][j] + " ");
62                        }
63                    }
64 1                  System.out.println();
65                }
66            }
67        }
68    }
```

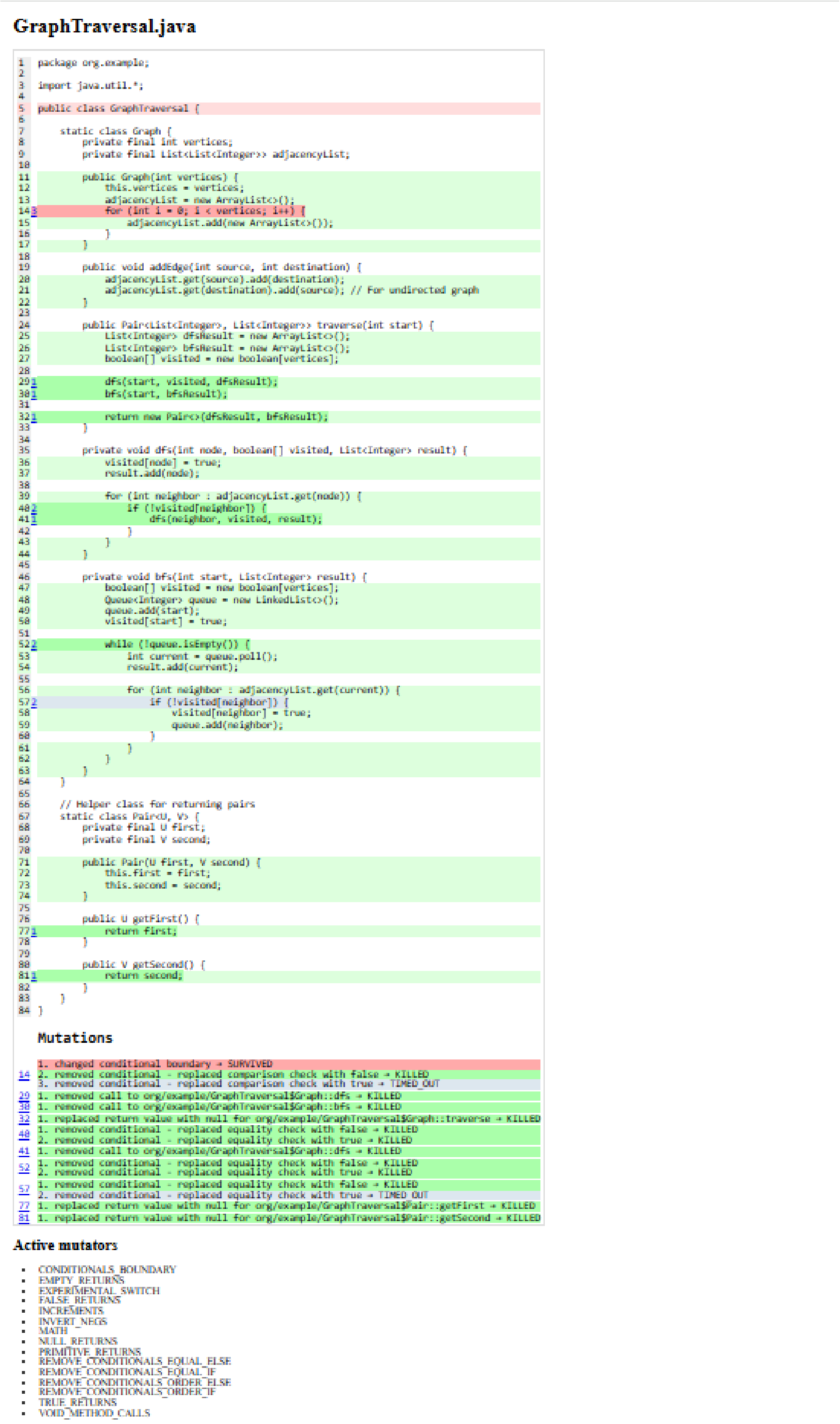
Mutations

```
    1. changed conditional boundary → KILLED
16  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → KILLED
17  1. removed call to java/util/Arrays::fill → KILLED
    1. changed conditional boundary → KILLED
30  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → KILLED
31  1. removed call to java/lang/System::arraycopy → KILLED
    1. changed conditional boundary → KILLED
35  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → KILLED
    1. changed conditional boundary → KILLED
36  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → KILLED
    1. changed conditional boundary → KILLED
37  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → KILLED
    1. removed conditional - replaced equality check with false → KILLED
38  2. removed conditional - replaced equality check with false → KILLED
    3. removed conditional - replaced equality check with true → KILLED
    4. removed conditional - replaced equality check with true → KILLED
39  1. Replaced integer addition with subtraction → KILLED
    1. changed conditional boundary → KILLED
46  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → KILLED
    1. changed conditional boundary → KILLED
47  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → KILLED
52  1. replaced return value with null for org/example/FloydWarshall$Graph::floydWarshall → KILLED
    1. changed conditional boundary → NO_COVERAGE
56  2. removed conditional - replaced comparison check with false → NO_COVERAGE
    3. removed conditional - replaced comparison check with true → NO_COVERAGE
    1. changed conditional boundary → NO_COVERAGE
57  2. removed conditional - replaced comparison check with false → NO_COVERAGE
    3. removed conditional - replaced comparison check with true → NO_COVERAGE
58  1. removed conditional - replaced equality check with false → NO_COVERAGE
    2. removed conditional - replaced equality check with true → NO_COVERAGE
59  1. removed call to java/io/PrintStream::print → NO_COVERAGE
61  1. removed call to java/io/PrintStream::print → NO_COVERAGE
64  1. removed call to java/io/PrintStream::println → NO_COVERAGE
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- EXPERIMENTAL_SWITCH
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_EQUAL_IF
- REMOVE_CONDITIONALS_ORDER_ELSE
- REMOVE_CONDITIONALS_ORDER_IF
- TRUE_RETURNS
- VOID_METHOD_CALLS

8.GraphTraversal.



GraphTraversal.java

```
1    package org.example;
2
3    import java.util.*;
4
5    public class GraphTraversal {
6
7        static class Graph {
8            private final int vertices;
9            private final List<List<Integer>> adjacencyList;
10
11           public Graph(int vertices) {
12               this.vertices = vertices;
13               adjacencyList = new ArrayList<>();
14 3             for (int i = 0; i < vertices; i++) {
15                   adjacencyList.add(new ArrayList<>());
16               }
17           }
18
19           public void addEdge(int source, int destination) {
20               adjacencyList.get(source).add(destination);
21               adjacencyList.get(destination).add(source); // For undirected graph
22           }
23
24           public Pair<List<Integer>, List<Integer>> traverse(int start) {
25               List<Integer> dfsResult = new ArrayList<>();
26               List<Integer> bfsResult = new ArrayList<>();
27               boolean[] visited = new boolean[vertices];
28
29 1             dfs(start, visited, dfsResult);
30 1             bfs(start, bfsResult);
31
32 1             return new Pair<>(dfsResult, bfsResult);
33           }
34
35           private void dfs(int node, boolean[] visited, List<Integer> result) {
36               visited[node] = true;
37               result.add(node);
38
39               for (int neighbor : adjacencyList.get(node)) {
40 2                 if (!visited[neighbor]) {
41 1                     dfs(neighbor, visited, result);
42                   }
43               }
44           }
45
46           private void bfs(int start, List<Integer> result) {
47               boolean[] visited = new boolean[vertices];
48               Queue<Integer> queue = new LinkedList<>();
49               queue.add(start);
50               visited[start] = true;
51
52 2             while (!queue.isEmpty()) {
53                   int current = queue.poll();
54                   result.add(current);
55
56                   for (int neighbor : adjacencyList.get(current)) {
57 2                     if (!visited[neighbor]) {
58                           visited[neighbor] = true;
59                           queue.add(neighbor);
60                       }
61                   }
62               }
63           }
64       }
65
66       // Helper class for returning pairs
67       static class Pair<U, V> {
68           private final U first;
69           private final V second;
70
71           public Pair(U first, V second) {
72               this.first = first;
73               this.second = second;
74           }
75
76           public U getFirst() {
77 1             return first;
78           }
79
80           public V getSecond() {
81 1             return second;
82           }
83       }
84   }
```

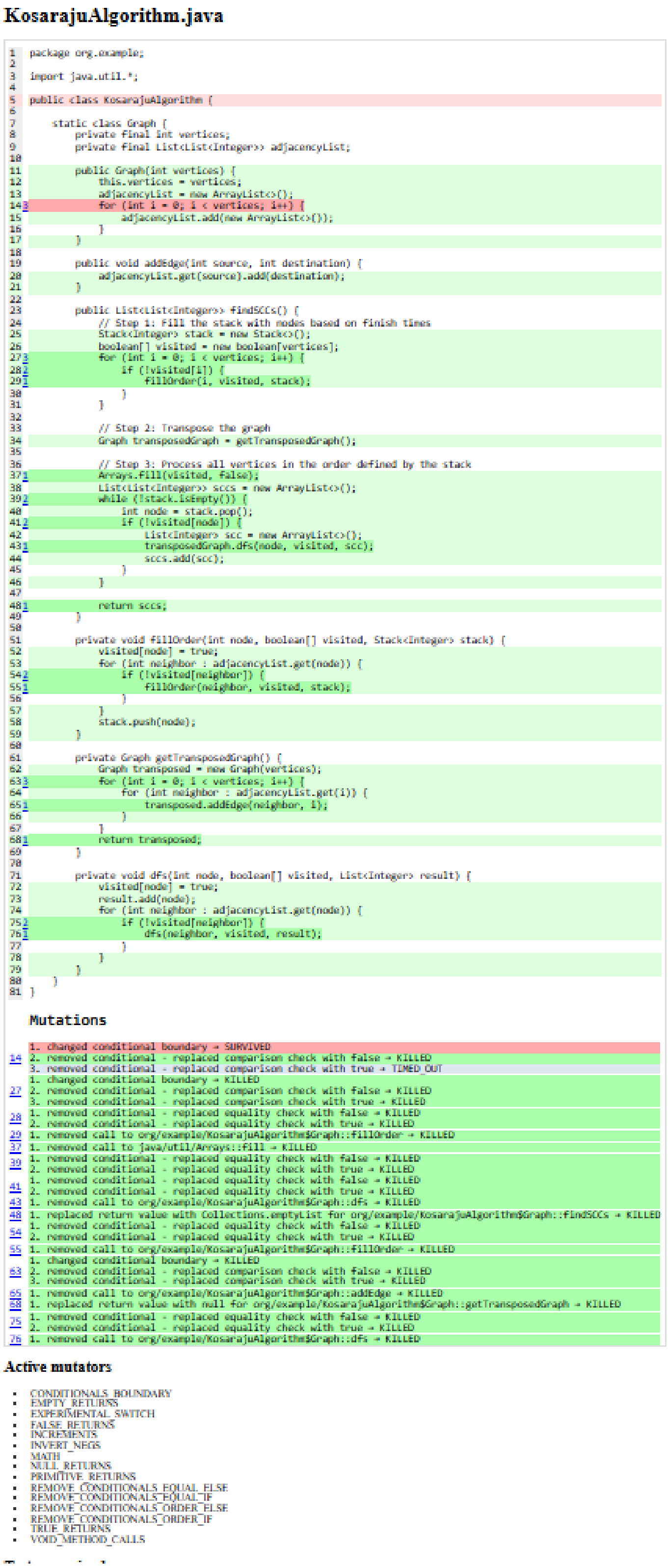
Mutations

```
    1. changed conditional boundary → SURVIVED
14  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → TIMED_OUT
29  1. removed call to org/example/GraphTraversal$Graph::dfs → KILLED
30  1. removed call to org/example/GraphTraversal$Graph::bfs → KILLED
32  1. replaced return value with null for org/example/GraphTraversal$Graph::traverse → KILLED
40  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
41  1. removed call to org/example/GraphTraversal$Graph::dfs → KILLED
52  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
57  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → TIMED_OUT
77  1. replaced return value with null for org/example/GraphTraversal$Pair::getFirst → KILLED
81  1. replaced return value with null for org/example/GraphTraversal$Pair::getSecond → KILLED
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- EXPERIMENTAL_SWITCH
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_EQUAL_IF
- REMOVE_CONDITIONALS_ORDER_ELSE
- REMOVE_CONDITIONALS_ORDER_IF
- TRUE_RETURNS
- VOID_METHOD_CALLS

9.KosarajuAlgorithm.



KosarajuAlgorithm.java

```
1   package org.example;
2
3   import java.util.*;
4
5   public class KosarajuAlgorithm {
6
7       static class Graph {
8           private final int vertices;
9           private final List<List<Integer>> adjacencyList;
10
11          public Graph(int vertices) {
12              this.vertices = vertices;
13              adjacencyList = new ArrayList<>();
14 3            for (int i = 0; i < vertices; i++) {
15                  adjacencyList.add(new ArrayList<>());
16              }
17          }
18
19          public void addEdge(int source, int destination) {
20              adjacencyList.get(source).add(destination);
21          }
22
23          public List<List<Integer>> findSCCs() {
24              // Step 1: Fill the stack with nodes based on finish times
25              Stack<Integer> stack = new Stack<>();
26              boolean[] visited = new boolean[vertices];
27 3            for (int i = 0; i < vertices; i++) {
28 2                if (!visited[i]) {
29 1                    fillOrder(i, visited, stack);
30                  }
31              }
32
33              // Step 2: Transpose the graph
34              Graph transposedGraph = getTransposedGraph();
35
36              // Step 3: Process all vertices in the order defined by the stack
37 1            Arrays.fill(visited, false);
38              List<List<Integer>> sccs = new ArrayList<>();
39 2            while (!stack.isEmpty()) {
40                  int node = stack.pop();
41 2                if (!visited[node]) {
42                      List<Integer> scc = new ArrayList<>();
43 1                    transposedGraph.dfs(node, visited, scc);
44                      sccs.add(scc);
45                  }
46              }
47
48 1            return sccs;
49          }
50
51          private void fillOrder(int node, boolean[] visited, Stack<Integer> stack) {
52              visited[node] = true;
53              for (int neighbor : adjacencyList.get(node)) {
54 2                if (!visited[neighbor]) {
55 1                    fillOrder(neighbor, visited, stack);
56                  }
57              }
58              stack.push(node);
59          }
60
61          private Graph getTransposedGraph() {
62              Graph transposed = new Graph(vertices);
63 3            for (int i = 0; i < vertices; i++) {
64                  for (int neighbor : adjacencyList.get(i)) {
65 1                    transposed.addEdge(neighbor, i);
66                  }
67              }
68 1            return transposed;
69          }
70
71          private void dfs(int node, boolean[] visited, List<Integer> result) {
72              visited[node] = true;
73              result.add(node);
74              for (int neighbor : adjacencyList.get(node)) {
75 2                if (!visited[neighbor]) {
76 1                    dfs(neighbor, visited, result);
77                  }
78              }
79          }
80      }
81  }
```

Mutations

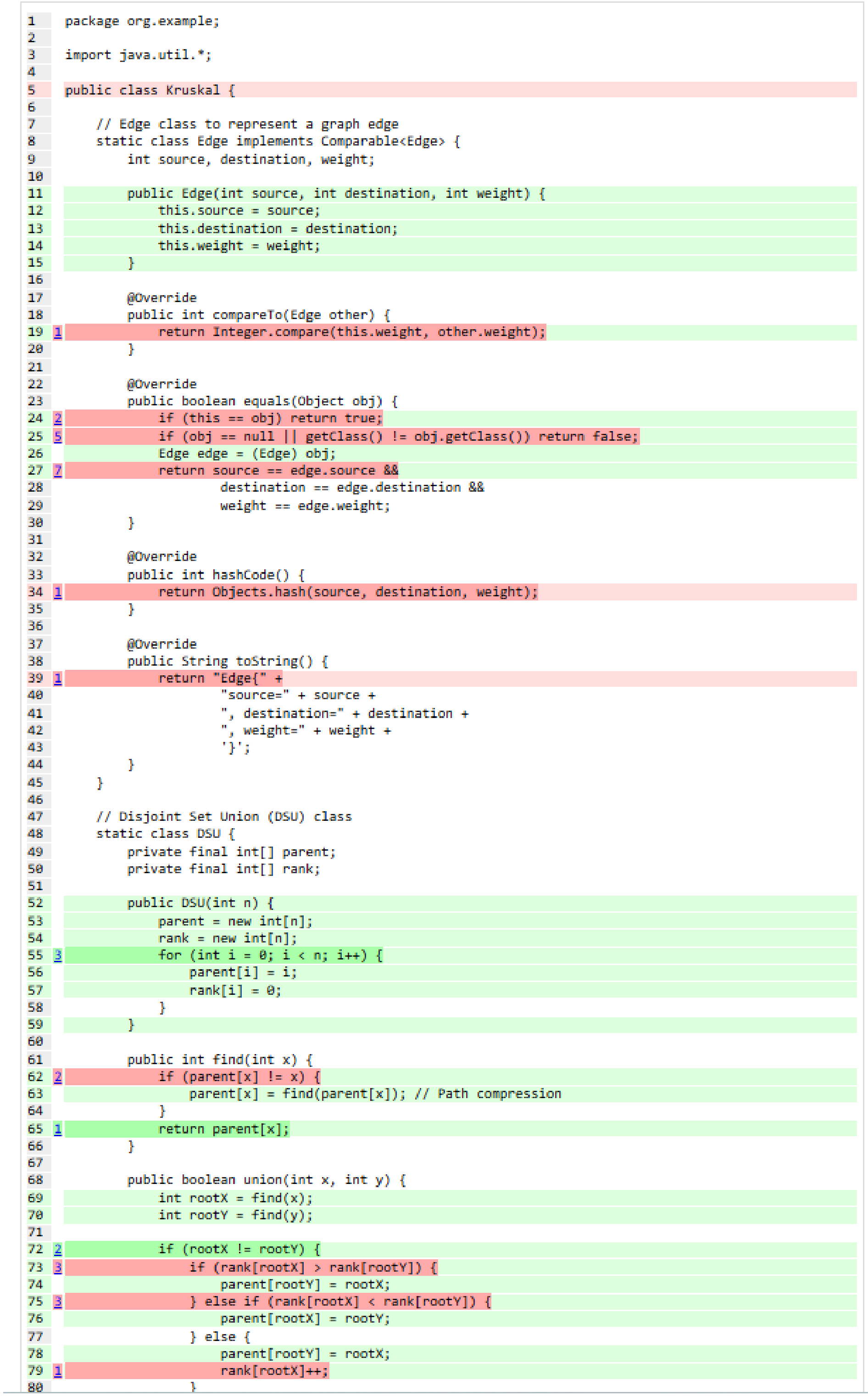
```
    1. changed conditional boundary → SURVIVED
14  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → TIMED_OUT
    1. changed conditional boundary → KILLED
27  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → KILLED
28  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
29  1. removed call to org/example/KosarajuAlgorithm$Graph::fillOrder → KILLED
37  1. removed call to java/util/Arrays::fill → KILLED
39  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
41  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
43  1. removed call to org/example/KosarajuAlgorithm$Graph::dfs → KILLED
48  1. replaced return value with Collections.emptyList for org/example/KosarajuAlgorithm$Graph::findSCCs → KILLED
54  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
55  1. removed call to org/example/KosarajuAlgorithm$Graph::fillOrder → KILLED
    1. changed conditional boundary → KILLED
63  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → KILLED
65  1. removed call to org/example/KosarajuAlgorithm$Graph::addEdge → KILLED
68  1. replaced return value with null for org/example/KosarajuAlgorithm$Graph::getTransposedGraph → KILLED
75  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
76  1. removed call to org/example/KosarajuAlgorithm$Graph::dfs → KILLED
```

Active mutators

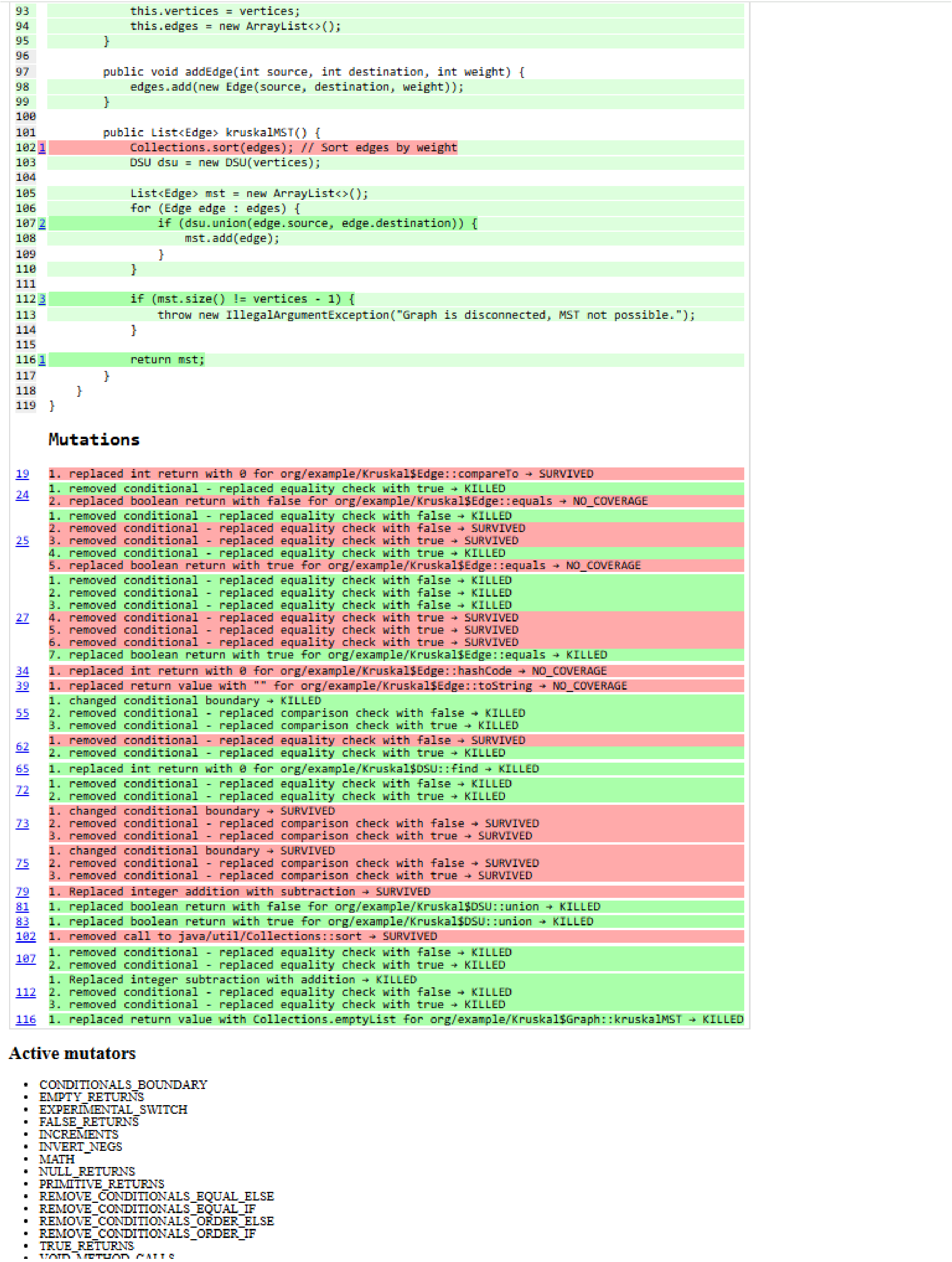
- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- EXPERIMENTAL_SWITCH
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_EQUAL_IF
- REMOVE_CONDITIONALS_ORDER_ELSE
- REMOVE_CONDITIONALS_ORDER_IF
- TRUE_RETURNS
- VOID_METHOD_CALLS

10.Kruskal

Kruskal.java



```
   package org.example;

   import java.util.*;

   public class Kruskal {

       // Edge class to represent a graph edge
       static class Edge implements Comparable<Edge> {
           int source, destination, weight;

           public Edge(int source, int destination, int weight) {
               this.source = source;
               this.destination = destination;
               this.weight = weight;
           }

           @Override
           public int compareTo(Edge other) {
1              return Integer.compare(this.weight, other.weight);
           }

           @Override
           public boolean equals(Object obj) {
2              if (this == obj) return true;
5              if (obj == null || getClass() != obj.getClass()) return false;
               Edge edge = (Edge) obj;
7              return source == edge.source &&
                       destination == edge.destination &&
                       weight == edge.weight;
           }

           @Override
           public int hashCode() {
1              return Objects.hash(source, destination, weight);
           }

           @Override
           public String toString() {
1              return "Edge{" +
                       "source=" + source +
                       ", destination=" + destination +
                       ", weight=" + weight +
                       '}';
           }
       }

       // Disjoint Set Union (DSU) class
       static class DSU {
           private final int[] parent;
           private final int[] rank;

           public DSU(int n) {
               parent = new int[n];
               rank = new int[n];
3              for (int i = 0; i < n; i++) {
                   parent[i] = i;
                   rank[i] = 0;
               }
           }

           public int find(int x) {
2              if (parent[x] != x) {
                   parent[x] = find(parent[x]); // Path compression
               }
1              return parent[x];
           }

           public boolean union(int x, int y) {
               int rootX = find(x);
               int rootY = find(y);

2              if (rootX != rootY) {
3                  if (rank[rootX] > rank[rootY]) {
                       parent[rootY] = rootX;
3                  } else if (rank[rootX] < rank[rootY]) {
                       parent[rootX] = rootY;
                   } else {
                       parent[rootY] = rootX;
1                      rank[rootX]++;
                   }
```



```
93          this.vertices = vertices;
94          this.edges = new ArrayList<>();
95      }
96
97      public void addEdge(int source, int destination, int weight) {
98          edges.add(new Edge(source, destination, weight));
99      }
100
101     public List<Edge> kruskalMST() {
102 1       Collections.sort(edges); // Sort edges by weight
103         DSU dsu = new DSU(vertices);
104
105         List<Edge> mst = new ArrayList<>();
106         for (Edge edge : edges) {
107 2           if (dsu.union(edge.source, edge.destination)) {
108                 mst.add(edge);
109             }
110         }
111
112 3       if (mst.size() != vertices - 1) {
113             throw new IllegalArgumentException("Graph is disconnected, MST not possible.");
114         }
115
116 1       return mst;
117     }
118   }
119 }
```

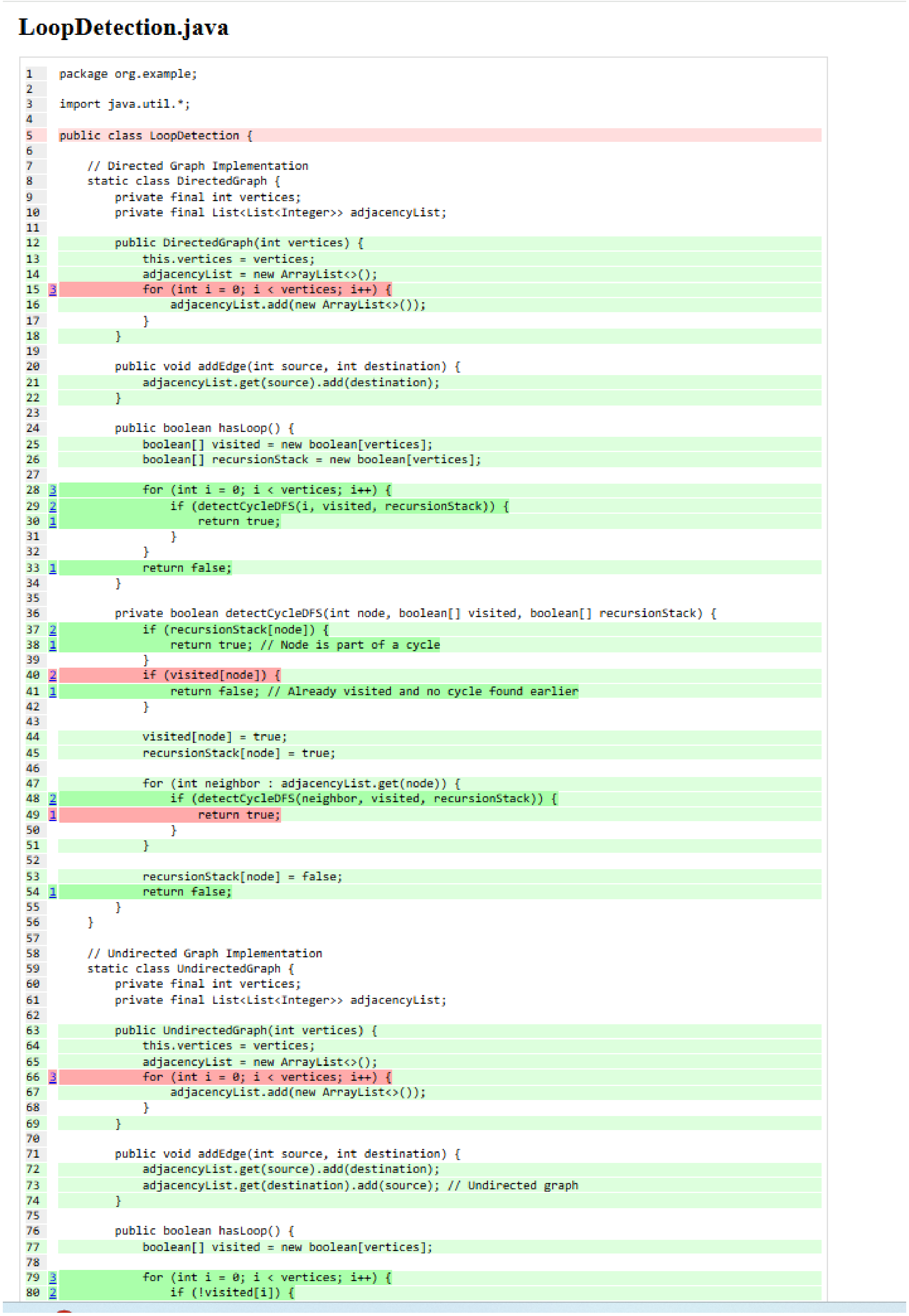
Mutations

```
19   1. replaced int return with 0 for org/example/Kruskal$Edge::compareTo → SURVIVED
24   1. removed conditional - replaced equality check with true → KILLED
     2. replaced boolean return with false for org/example/Kruskal$Edge::equals → NO_COVERAGE
     1. removed conditional - replaced equality check with false → KILLED
     2. removed conditional - replaced equality check with false → SURVIVED
25   3. removed conditional - replaced equality check with true → SURVIVED
     4. removed conditional - replaced equality check with true → KILLED
     5. replaced boolean return with true for org/example/Kruskal$Edge::equals → NO_COVERAGE
     1. removed conditional - replaced equality check with false → KILLED
     2. removed conditional - replaced equality check with false → KILLED
     3. removed conditional - replaced equality check with false → KILLED
27   4. removed conditional - replaced equality check with true → SURVIVED
     5. removed conditional - replaced equality check with true → SURVIVED
     6. removed conditional - replaced equality check with true → SURVIVED
     7. replaced boolean return with true for org/example/Kruskal$Edge::equals → KILLED
34   1. replaced int return with 0 for org/example/Kruskal$Edge::hashCode → NO_COVERAGE
39   1. replaced return value with "" for org/example/Kruskal$Edge::toString → NO_COVERAGE
     1. changed conditional boundary → KILLED
55   2. removed conditional - replaced comparison check with false → KILLED
     3. removed conditional - replaced comparison check with true → KILLED
62   1. removed conditional - replaced equality check with false → SURVIVED
     2. removed conditional - replaced equality check with true → KILLED
65   1. replaced int return with 0 for org/example/Kruskal$DSU::find → KILLED
72   1. removed conditional - replaced equality check with false → KILLED
     2. removed conditional - replaced equality check with true → KILLED
     1. changed conditional boundary → SURVIVED
73   2. removed conditional - replaced comparison check with false → SURVIVED
     3. removed conditional - replaced comparison check with true → SURVIVED
     1. changed conditional boundary → SURVIVED
75   2. removed conditional - replaced comparison check with false → SURVIVED
     3. removed conditional - replaced comparison check with true → SURVIVED
79   1. Replaced integer addition with subtraction → SURVIVED
81   1. replaced boolean return with false for org/example/Kruskal$DSU::union → KILLED
83   1. replaced boolean return with true for org/example/Kruskal$DSU::union → KILLED
102  1. removed call to java/util/Collections::sort → SURVIVED
107  1. removed conditional - replaced equality check with false → KILLED
     2. removed conditional - replaced equality check with true → KILLED
     1. Replaced integer subtraction with addition → KILLED
112  2. removed conditional - replaced equality check with false → KILLED
     3. removed conditional - replaced equality check with true → KILLED
116  1. replaced return value with Collections.emptyList for org/example/Kruskal$Graph::kruskalMST → KILLED
```

Active mutators

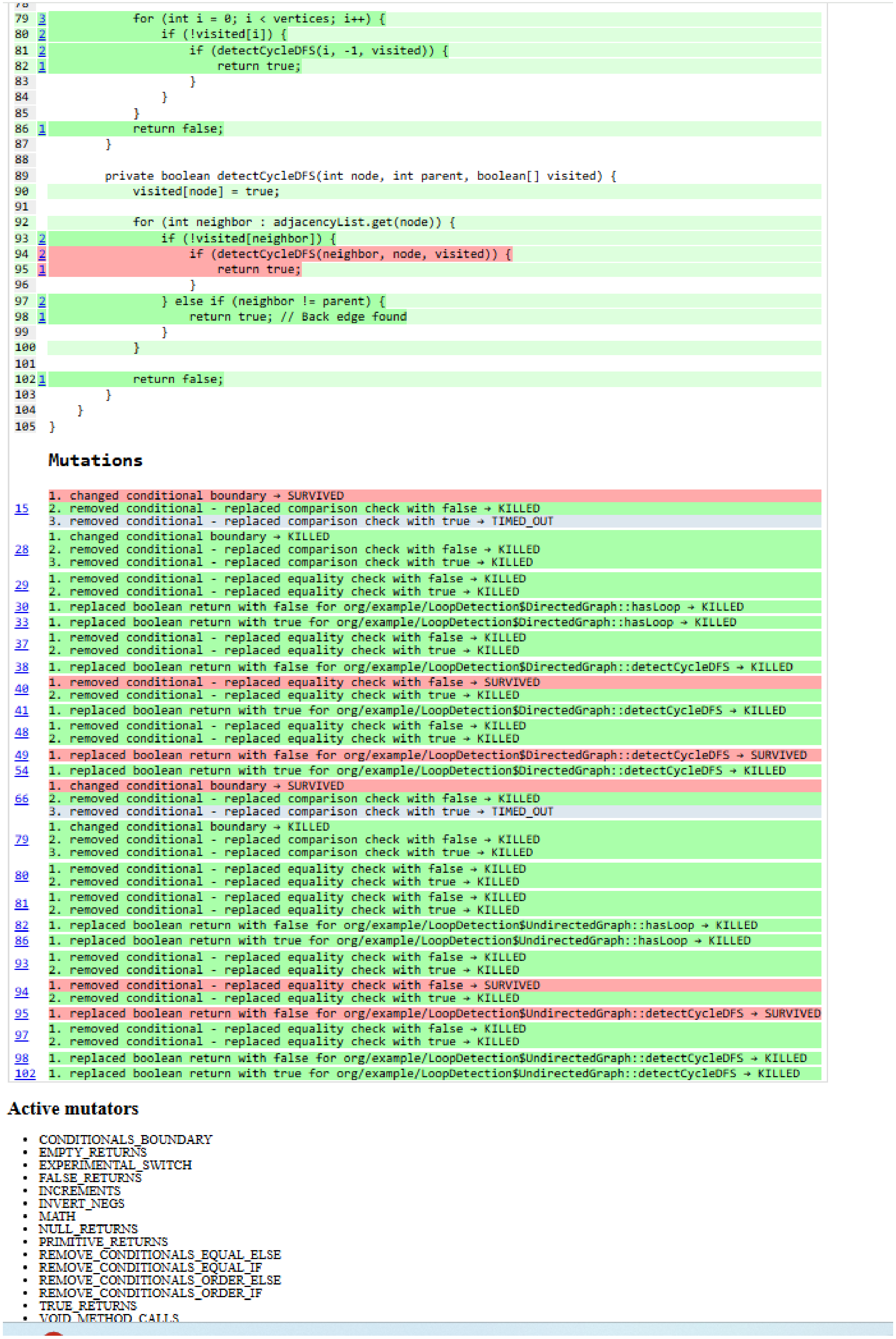
- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- EXPERIMENTAL_SWITCH
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_EQUAL_IF
- REMOVE_CONDITIONALS_ORDER_ELSE
- REMOVE_CONDITIONALS_ORDER_IF
- TRUE_RETURNS
- VOID_METHOD_CALLS

11.LoopDetection.



LoopDetection.java

```java
package org.example;

import java.util.*;

public class LoopDetection {

    // Directed Graph Implementation
    static class DirectedGraph {
        private final int vertices;
        private final List<List<Integer>> adjacencyList;

        public DirectedGraph(int vertices) {
            this.vertices = vertices;
            adjacencyList = new ArrayList<>();
            for (int i = 0; i < vertices; i++) {
                adjacencyList.add(new ArrayList<>());
            }
        }

        public void addEdge(int source, int destination) {
            adjacencyList.get(source).add(destination);
        }

        public boolean hasLoop() {
            boolean[] visited = new boolean[vertices];
            boolean[] recursionStack = new boolean[vertices];

            for (int i = 0; i < vertices; i++) {
                if (detectCycleDFS(i, visited, recursionStack)) {
                    return true;
                }
            }
            return false;
        }

        private boolean detectCycleDFS(int node, boolean[] visited, boolean[] recursionStack) {
            if (recursionStack[node]) {
                return true; // Node is part of a cycle
            }
            if (visited[node]) {
                return false; // Already visited and no cycle found earlier
            }

            visited[node] = true;
            recursionStack[node] = true;

            for (int neighbor : adjacencyList.get(node)) {
                if (detectCycleDFS(neighbor, visited, recursionStack)) {
                    return true;
                }
            }

            recursionStack[node] = false;
            return false;
        }
    }

    // Undirected Graph Implementation
    static class UndirectedGraph {
        private final int vertices;
        private final List<List<Integer>> adjacencyList;

        public UndirectedGraph(int vertices) {
            this.vertices = vertices;
            adjacencyList = new ArrayList<>();
            for (int i = 0; i < vertices; i++) {
                adjacencyList.add(new ArrayList<>());
            }
        }

        public void addEdge(int source, int destination) {
            adjacencyList.get(source).add(destination);
            adjacencyList.get(destination).add(source); // Undirected graph
        }

        public boolean hasLoop() {
            boolean[] visited = new boolean[vertices];

            for (int i = 0; i < vertices; i++) {
                if (!visited[i]) {
```



```
78
79  3            for (int i = 0; i < vertices; i++) {
80  2                if (!visited[i]) {
81  2                    if (detectCycleDFS(i, -1, visited)) {
82  1                        return true;
83                       }
84                   }
85               }
86  1            return false;
87           }
88
89           private boolean detectCycleDFS(int node, int parent, boolean[] visited) {
90               visited[node] = true;
91
92               for (int neighbor : adjacencyList.get(node)) {
93  2                if (!visited[neighbor]) {
94  2                    if (detectCycleDFS(neighbor, node, visited)) {
95  1                        return true;
96                       }
97  2                } else if (neighbor != parent) {
98  1                    return true; // Back edge found
99                   }
100              }
101
102 1            return false;
103          }
104      }
105  }
```

Mutations

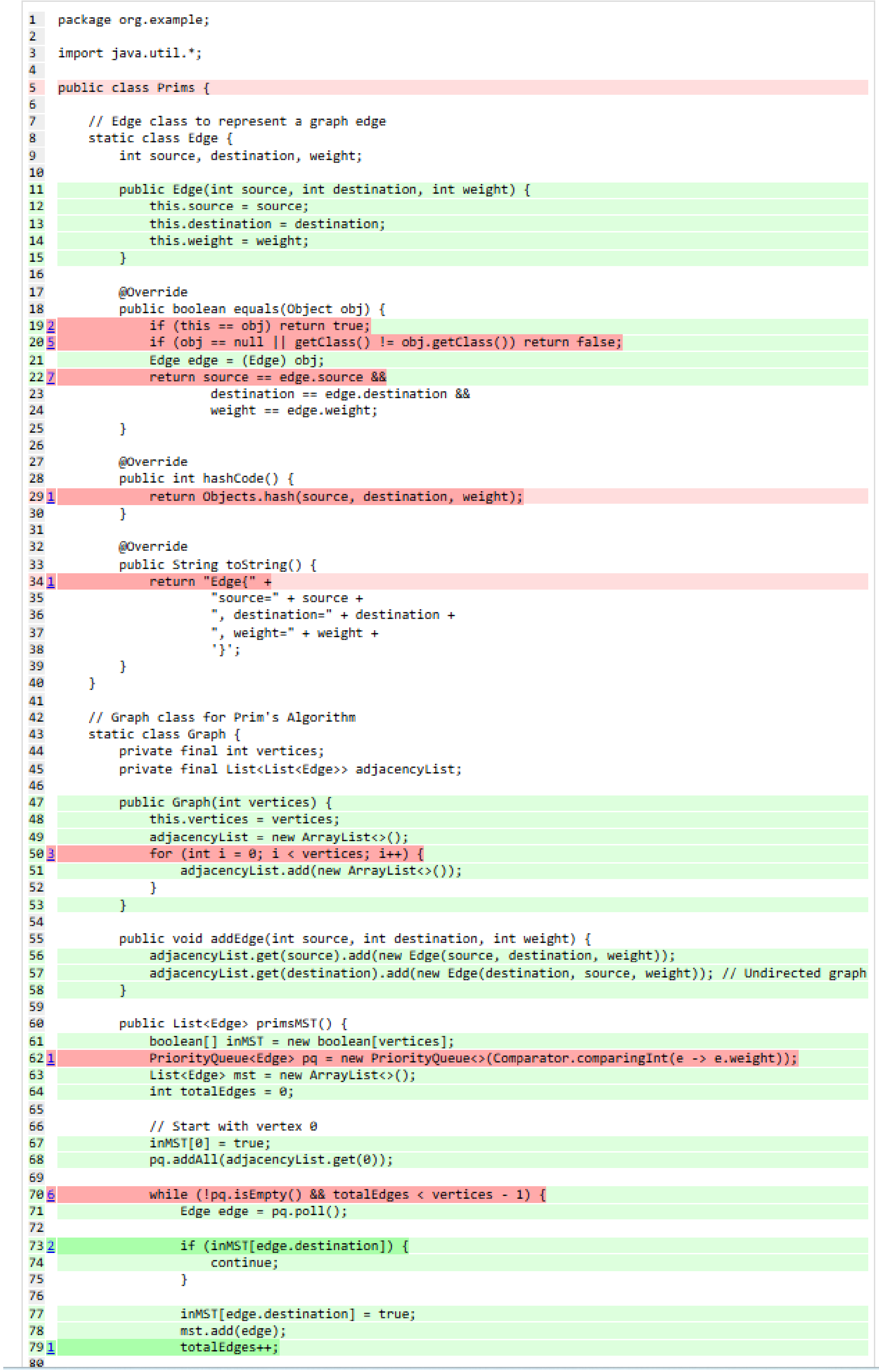
```
    1. changed conditional boundary → SURVIVED
15  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → TIMED_OUT
    1. changed conditional boundary → KILLED
28  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → KILLED
29  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
30  1. replaced boolean return with false for org/example/LoopDetection$DirectedGraph::hasLoop → KILLED
33  1. replaced boolean return with true for org/example/LoopDetection$DirectedGraph::hasLoop → KILLED
37  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
38  1. replaced boolean return with false for org/example/LoopDetection$DirectedGraph::detectCycleDFS → KILLED
40  1. removed conditional - replaced equality check with false → SURVIVED
    2. removed conditional - replaced equality check with true → KILLED
41  1. replaced boolean return with true for org/example/LoopDetection$DirectedGraph::detectCycleDFS → KILLED
48  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
49  1. replaced boolean return with false for org/example/LoopDetection$DirectedGraph::detectCycleDFS → SURVIVED
54  1. replaced boolean return with true for org/example/LoopDetection$DirectedGraph::detectCycleDFS → KILLED
    1. changed conditional boundary → SURVIVED
66  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → TIMED_OUT
    1. changed conditional boundary → KILLED
79  2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → KILLED
80  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
81  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
82  1. replaced boolean return with false for org/example/LoopDetection$UndirectedGraph::hasLoop → KILLED
86  1. replaced boolean return with true for org/example/LoopDetection$UndirectedGraph::hasLoop → KILLED
93  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
94  1. removed conditional - replaced equality check with false → SURVIVED
    2. removed conditional - replaced equality check with true → KILLED
95  1. replaced boolean return with false for org/example/LoopDetection$UndirectedGraph::detectCycleDFS → SURVIVED
97  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
98  1. replaced boolean return with false for org/example/LoopDetection$UndirectedGraph::detectCycleDFS → KILLED
102 1. replaced boolean return with true for org/example/LoopDetection$UndirectedGraph::detectCycleDFS → KILLED
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- EXPERIMENTAL_SWITCH
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_EQUAL_IF
- REMOVE_CONDITIONALS_ORDER_ELSE
- REMOVE_CONDITIONALS_ORDER_IF
- TRUE_RETURNS
- VOID_METHOD_CALLS

12 Prims

Prims.java



```java
package org.example;

import java.util.*;

public class Prims {

    // Edge class to represent a graph edge
    static class Edge {
        int source, destination, weight;

        public Edge(int source, int destination, int weight) {
            this.source = source;
            this.destination = destination;
            this.weight = weight;
        }

        @Override
        public boolean equals(Object obj) {
            if (this == obj) return true;
            if (obj == null || getClass() != obj.getClass()) return false;
            Edge edge = (Edge) obj;
            return source == edge.source &&
                    destination == edge.destination &&
                    weight == edge.weight;
        }

        @Override
        public int hashCode() {
            return Objects.hash(source, destination, weight);
        }

        @Override
        public String toString() {
            return "Edge{" +
                    "source=" + source +
                    ", destination=" + destination +
                    ", weight=" + weight +
                    '}';
        }
    }

    // Graph class for Prim's Algorithm
    static class Graph {
        private final int vertices;
        private final List<List<Edge>> adjacencyList;

        public Graph(int vertices) {
            this.vertices = vertices;
            adjacencyList = new ArrayList<>();
            for (int i = 0; i < vertices; i++) {
                adjacencyList.add(new ArrayList<>());
            }
        }

        public void addEdge(int source, int destination, int weight) {
            adjacencyList.get(source).add(new Edge(source, destination, weight));
            adjacencyList.get(destination).add(new Edge(destination, source, weight)); // Undirected graph
        }

        public List<Edge> primsMST() {
            boolean[] inMST = new boolean[vertices];
            PriorityQueue<Edge> pq = new PriorityQueue<>(Comparator.comparingInt(e -> e.weight));
            List<Edge> mst = new ArrayList<>();
            int totalEdges = 0;

            // Start with vertex 0
            inMST[0] = true;
            pq.addAll(adjacencyList.get(0));

            while (!pq.isEmpty() && totalEdges < vertices - 1) {
                Edge edge = pq.poll();

                if (inMST[edge.destination]) {
                    continue;
                }

                inMST[edge.destination] = true;
                mst.add(edge);
                totalEdges++;
74                      continue;
75                  }
76
77                  inMST[edge.destination] = true;
78                  mst.add(edge);
79 1                totalEdges++;
80
81                  // Add all edges from the new vertex to the priority queue
82                  for (Edge nextEdge : adjacencyList.get(edge.destination)) {
83 2                    if (!inMST[nextEdge.destination]) {
84                          pq.offer(nextEdge);
85                      }
86                  }
87              }
88
89 3            if (totalEdges != vertices - 1) {
90                  throw new IllegalArgumentException("Graph is disconnected, MST not possible.");
91              }
92
93 1            return mst;
94          }
95      }
96  }
```

Mutations

```
19  1. removed conditional - replaced equality check with true → KILLED
    2. replaced boolean return with false for org/example/Prims$Edge::equals → NO_COVERAGE
20  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with false → SURVIVED
    3. removed conditional - replaced equality check with true → SURVIVED
    4. removed conditional - replaced equality check with true → KILLED
    5. replaced boolean return with true for org/example/Prims$Edge::equals → NO_COVERAGE
22  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with false → KILLED
    3. removed conditional - replaced equality check with false → KILLED
    4. removed conditional - replaced equality check with true → SURVIVED
    5. removed conditional - replaced equality check with true → SURVIVED
    6. removed conditional - replaced equality check with true → SURVIVED
    7. replaced boolean return with true for org/example/Prims$Edge::equals → KILLED
29  1. replaced int return with 0 for org/example/Prims$Edge::hashCode → NO_COVERAGE
34  1. replaced return value with "" for org/example/Prims$Edge::toString → NO_COVERAGE
50  1. changed conditional boundary → SURVIVED
    2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → TIMED_OUT
62  1. replaced int return with 0 for org/example/Prims$Graph::lambda$primsMST$0 → SURVIVED
70  1. changed conditional boundary → SURVIVED
    2. Replaced integer subtraction with addition → SURVIVED
    3. removed conditional - replaced equality check with false → KILLED
    4. removed conditional - replaced equality check with true → KILLED
    5. removed conditional - replaced comparison check with false → KILLED
    6. removed conditional - replaced comparison check with true → SURVIVED
73  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
79  1. Changed increment from 1 to -1 → KILLED
83  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → SURVIVED
89  1. Replaced integer subtraction with addition → KILLED
    2. removed conditional - replaced equality check with false → KILLED
    3. removed conditional - replaced equality check with true → KILLED
93  1. replaced return value with Collections.emptyList for org/example/Prims$Graph::primsMST → KILLED
```

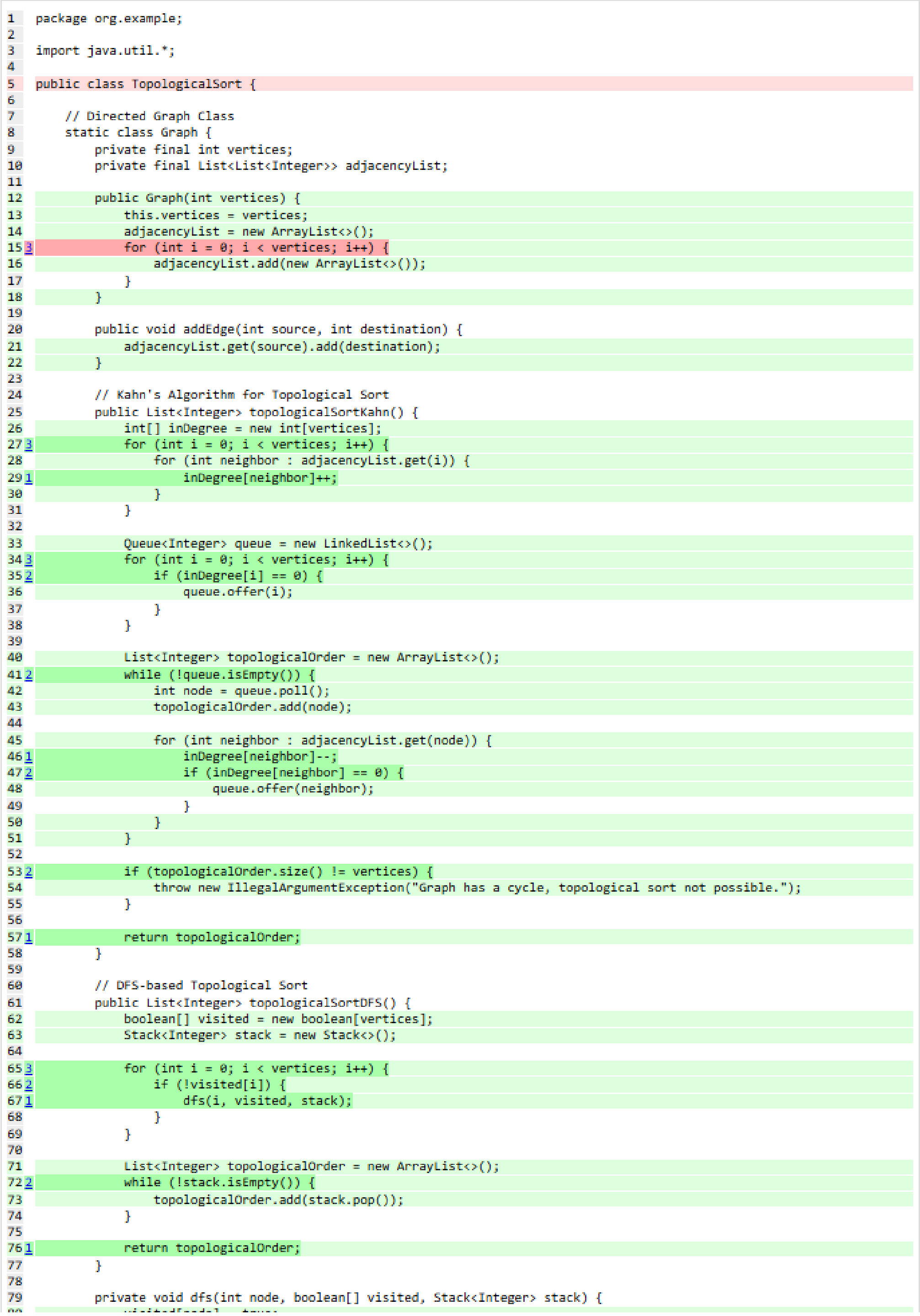
Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- EXPERIMENTAL_SWITCH
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_EQUAL_IF
- REMOVE_CONDITIONALS_ORDER_ELSE
- REMOVE_CONDITIONALS_ORDER_IF
- TRUE_RETURNS
- VOID_METHOD_CALLS

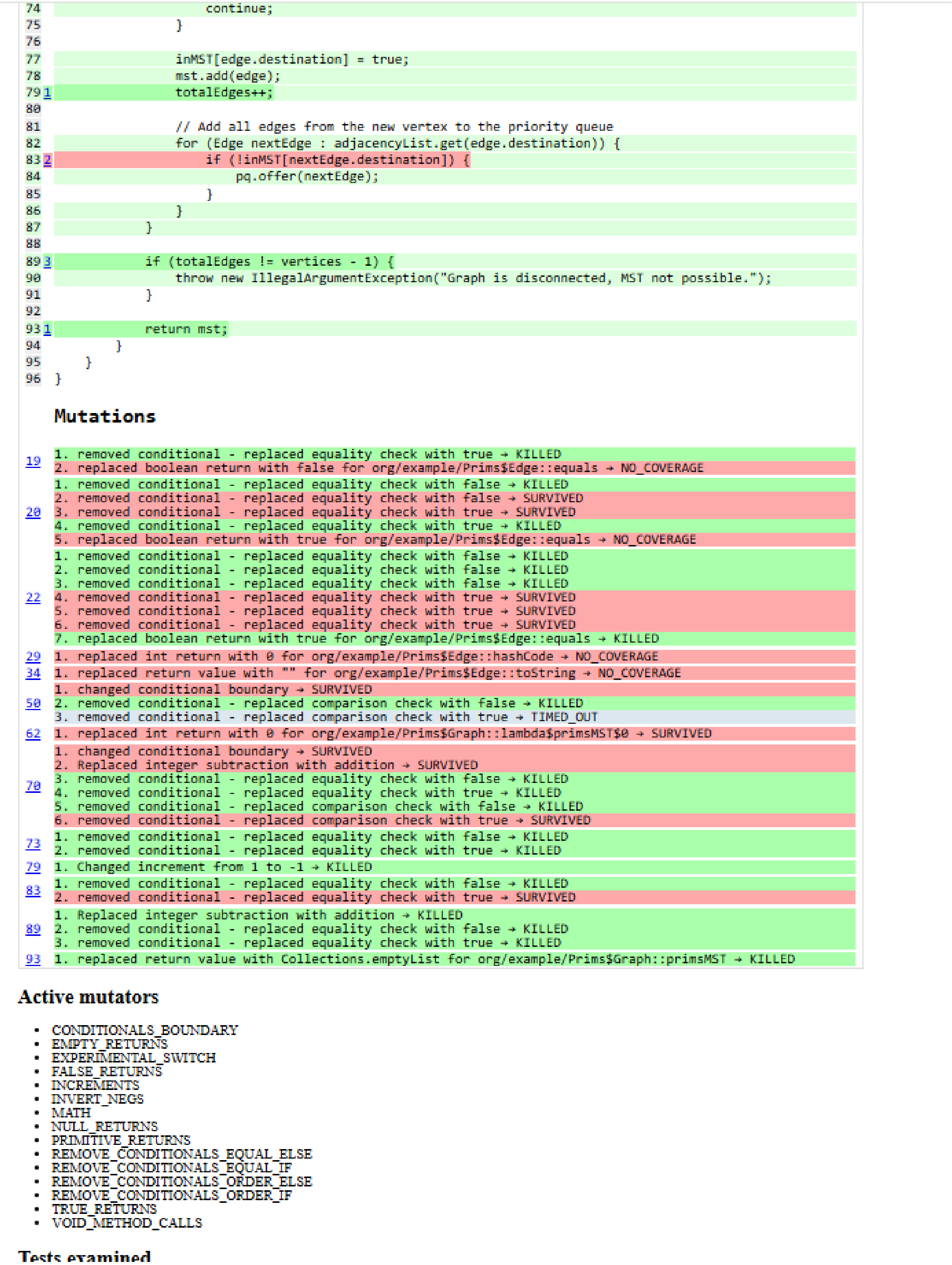
Tests examined

13.TopologicalSort

TopologicalSort.java



```java
package org.example;

import java.util.*;

public class TopologicalSort {

    // Directed Graph Class
    static class Graph {
        private final int vertices;
        private final List<List<Integer>> adjacencyList;

        public Graph(int vertices) {
            this.vertices = vertices;
            adjacencyList = new ArrayList<>();
            for (int i = 0; i < vertices; i++) {
                adjacencyList.add(new ArrayList<>());
            }
        }

        public void addEdge(int source, int destination) {
            adjacencyList.get(source).add(destination);
        }

        // Kahn's Algorithm for Topological Sort
        public List<Integer> topologicalSortKahn() {
            int[] inDegree = new int[vertices];
            for (int i = 0; i < vertices; i++) {
                for (int neighbor : adjacencyList.get(i)) {
                    inDegree[neighbor]++;
                }
            }

            Queue<Integer> queue = new LinkedList<>();
            for (int i = 0; i < vertices; i++) {
                if (inDegree[i] == 0) {
                    queue.offer(i);
                }
            }

            List<Integer> topologicalOrder = new ArrayList<>();
            while (!queue.isEmpty()) {
                int node = queue.poll();
                topologicalOrder.add(node);

                for (int neighbor : adjacencyList.get(node)) {
                    inDegree[neighbor]--;
                    if (inDegree[neighbor] == 0) {
                        queue.offer(neighbor);
                    }
                }
            }

            if (topologicalOrder.size() != vertices) {
                throw new IllegalArgumentException("Graph has a cycle, topological sort not possible.");
            }

            return topologicalOrder;
        }

        // DFS-based Topological Sort
        public List<Integer> topologicalSortDFS() {
            boolean[] visited = new boolean[vertices];
            Stack<Integer> stack = new Stack<>();

            for (int i = 0; i < vertices; i++) {
                if (!visited[i]) {
                    dfs(i, visited, stack);
                }
            }

            List<Integer> topologicalOrder = new ArrayList<>();
            while (!stack.isEmpty()) {
                topologicalOrder.add(stack.pop());
            }

            return topologicalOrder;
        }

        private void dfs(int node, boolean[] visited, Stack<Integer> stack) {
            visited[node] = true;
```



```
74                      continue;
75                  }
76
77                  inMST[edge.destination] = true;
78                  mst.add(edge);
79 1                totalEdges++;
80
81                  // Add all edges from the new vertex to the priority queue
82                  for (Edge nextEdge : adjacencyList.get(edge.destination)) {
83 2                    if (!inMST[nextEdge.destination]) {
84                          pq.offer(nextEdge);
85                      }
86                  }
87              }
88
89 3            if (totalEdges != vertices - 1) {
90                  throw new IllegalArgumentException("Graph is disconnected, MST not possible.");
91              }
92
93 1            return mst;
94          }
95      }
96  }
```

Mutations

```
19  1. removed conditional - replaced equality check with true → KILLED
    2. replaced boolean return with false for org/example/Prims$Edge::equals → NO_COVERAGE
20  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with false → SURVIVED
    3. removed conditional - replaced equality check with true → SURVIVED
    4. removed conditional - replaced equality check with true → KILLED
    5. replaced boolean return with true for org/example/Prims$Edge::equals → NO_COVERAGE
22  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with false → KILLED
    3. removed conditional - replaced equality check with false → KILLED
    4. removed conditional - replaced equality check with true → SURVIVED
    5. removed conditional - replaced equality check with true → SURVIVED
    6. removed conditional - replaced equality check with true → SURVIVED
    7. replaced boolean return with true for org/example/Prims$Edge::equals → KILLED
29  1. replaced int return with 0 for org/example/Prims$Edge::hashCode → NO_COVERAGE
34  1. replaced return value with "" for org/example/Prims$Edge::toString → NO_COVERAGE
50  1. changed conditional boundary → SURVIVED
    2. removed conditional - replaced comparison check with false → KILLED
    3. removed conditional - replaced comparison check with true → TIMED_OUT
62  1. replaced int return with 0 for org/example/Prims$Graph::lambda$primsMST$0 → SURVIVED
70  1. changed conditional boundary → SURVIVED
    2. Replaced integer subtraction with addition → SURVIVED
    3. removed conditional - replaced equality check with false → KILLED
    4. removed conditional - replaced equality check with true → KILLED
    5. removed conditional - replaced comparison check with false → KILLED
    6. removed conditional - replaced comparison check with true → SURVIVED
73  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → KILLED
79  1. Changed increment from 1 to -1 → KILLED
83  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with true → SURVIVED
89  1. Replaced integer subtraction with addition → KILLED
    2. removed conditional - replaced equality check with false → KILLED
    3. removed conditional - replaced equality check with true → KILLED
93  1. replaced return value with Collections.emptyList for org/example/Prims$Graph::primsMST → KILLED
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- EXPERIMENTAL_SWITCH
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_EQUAL_IF
- REMOVE_CONDITIONALS_ORDER_ELSE
- REMOVE_CONDITIONALS_ORDER_IF
- TRUE_RETURNS
- VOID_METHOD_CALLS

Tests examined