GraphAlgorithms.java

```
    package com.softwareTesting;
    import java.util.*;



    public class GraphAlgorithms {
            int[][] floydWarshall(int dist[][], int V)
            {
            int i, j, k;

3           for (k = 0; k < V; k++)
            {
3                   for (i = 0; i < V; i++)
                    {
3                   for (j = 0; j < V; j++)
                    {
3                           if (dist[i][k] + dist[k][j] < dist[i][j])
                            {
1                                   dist[i][j] = dist[i][k] + dist[k][j];
                            }
                    }
                    }
            }

1           return dist;

            }


        int minKey(int key[], Boolean mstSet[], int V)
        {
            int min = Integer.MAX_VALUE, min_index = -1;

3           for (int v = 0; v < V; v++)
            {
3               if (mstSet[v] == false && key[v] < min)
                {
                    min = key[v];
                    min_index = v;
                }
            }

1           return min_index;
        }


        int totalMST(int key[], int V)
        {
            int sum = 0;
3           for (int i = 0; i < V; i++)
            {
1               sum += key[i];
            }
1           return sum;
        }


        int primMST(int graph[][], int V)
        {
            int parent[] = new int[V];


            int key[] = new int[V];


            Boolean mstSet[] = new Boolean[V];

3           for (int i = 0; i < V; i++)
            {
                key[i] = Integer.MAX_VALUE;
                mstSet[i] = false;
72              }
73
74
75              key[0] = 0;
76
77              parent[0] = -1;
78
79
80  4           for (int count = 0; count < V - 1; count++)
81              {
82
83                  int u = minKey(key, mstSet, V);
84
85                  mstSet[u] = true;
86
87
88  3               for (int v = 0; v < V; v++)
89                  {
90
91
92  4                   if (graph[u][v] != 0 && mstSet[v] == false && graph[u][v] < key[v])
93                      {
94                          parent[v] = u;
95                          key[v] = graph[u][v];
96                      }
97                  }
98              }
99
100 1           return totalMST(key, V);
101         }
102
103         int minDistance(int dist[], Boolean sptSet[], int V)
104         {
105             // Initialize min value
106             int min = Integer.MAX_VALUE, min_index = -1;
107
108 3           for (int v = 0; v < V; v++)
109             {
110 3               if (sptSet[v] == false && dist[v] <= min)
111                 {
112                     min = dist[v];
113                     min_index = v;
114                 }
115             }
116 1           return min_index;
117         }
118
119
120         int[] dijkstra(int graph[][], int src, int V)
121         {
122             int dist[] = new int[V];
123             Boolean sptSet[] = new Boolean[V];
124
125 3           for (int i = 0; i < V; i++)
126             {
127                 dist[i] = Integer.MAX_VALUE;
128                 sptSet[i] = false;
129             }
130
131             dist[src] = 0;
132 4           for (int count = 0; count < V - 1; count++)
133             {
134
135                 int u = minDistance(dist, sptSet,V);
136
137
138                 sptSet[u] = true;
139
140
141 3               for (int v = 0; v < V; v++)
142 2                   if (!sptSet[v] && graph[u][v] != 0
143 1                       && dist[u] != Integer.MAX_VALUE
144 3                       && dist[u] + graph[u][v] < dist[v])
145 1                       dist[v] = dist[u] + graph[u][v];
146             }
147 1           return dist;
```

```
148      }
149
150
151      void DFSUtil(int v, boolean[] visited, ArrayList<ArrayList<Integer> > adjListArray)
152          {
153                  // Mark the current node as visited and print it
154                  visited[v] = true;
155                  // Recur for all the vertices
156                  // adjacent to this vertex
157                  for (int x : adjListArray.get(v))
158                  {
159 1                        if (!visited[x])
160                          {
161 1                                DFSUtil(x, visited,adjListArray);
162                          }
163                  }
164          }
165          int connectedComponents(ArrayList<ArrayList<Integer> > adjListArray, int V)
166          {
167                  // Mark all the vertices as not visited
168                  boolean[] visited = new boolean[V];
169          int cnt =0 ;
170
171 3                for (int v = 0; v < V; ++v)
172                  {
173 1                        if (!visited[v])
174                          {
175                                  // print all reachable vertices
176                                  // from v
177 1                                DFSUtil(v, visited,adjListArray);
178 1                                cnt += 1;
179                                  // System.out.println();
180                          }
181                  }
182 1                return cnt;
183          }
184
185
186
187          boolean isBCUtil(int u, boolean visited[], int disc[],int low[],int parent[], LinkedList<Integer> adj[],int tim
188          {
189                  int children = 0;
190                  visited[u] = true;
191 1                disc[u] = low[u] = ++time;
192                  Iterator<Integer> i = adj[u].iterator();
193
194
195 1                while (i.hasNext())
196                  {
197                     int v = i.next();
198 1                   if (!visited[v])
199                     {
200 1                       children++;
201                         parent[v] = u;
202 1                       if (isBCUtil(v, visited, disc, low, parent,adj,time))
203                         {
204 1                           return true;
205                         }
206
207                         low[u]  = Math.min(low[u], low[v]);
208
209 3                       if (parent[u] == -1 && children > 1)
210                         {
211 1                           return true;
212                         }
213 3                       if (parent[u] != -1 && low[v] >= disc[u])
214                         {
215 1                           return true;
216                         }
217                     }
218 1                   else if (v != parent[u])
219                     {
220                         low[u]  = Math.min(low[u], disc[v]);
221                     }
222                  }
223 1                return false;
```

```
          }

          boolean isBC(int V, LinkedList<Integer> adj[])
          {
                  boolean visited[] = new boolean[V];
                  int disc[] = new int[V];
                  int low[] = new int[V];
                  int parent[] = new int[V];

3                 for (int i = 0; i < V; i++)
                  {
                     parent[i] = -1;
                     visited[i] = false;
                  }
                  int time = 0;

1                 if (isBCUtil(0, visited, disc, low, parent, adj,time) == true)
                  {
1                    return false;
                  }

3                 for (int i = 0; i < V; i++)
                  {
1                    if (visited[i] == false)
                     {
1                        return false;
                     }
                  }
1                 return true;
          }






          boolean bfs(int rGraph[][], int s, int t, int parent[], int V)
      {
          // Create a visited array and mark all vertices as
          // not visited
          boolean visited[] = new boolean[V];

3         for (int i = 0; i < V; ++i)
          {
              visited[i] = false;
          }
          // Create a queue, enqueue source vertex and mark
          // source vertex as visited
          LinkedList<Integer> queue = new LinkedList<Integer>();

          queue.add(s);

          visited[s] = true;
          parent[s] = -1;

          // Standard BFS Loop
1         while (queue.size() != 0)
          {
              int u = queue.poll();

3             for (int v = 0; v < V; v++)
              {
3                 if (visited[v] == false && rGraph[u][v] > 0)
                  {
1                     if (v == t)
                      {
                          parent[v] = u;
1                         return true;
                      }
                      queue.add(v);
                      parent[v] = u;
                      visited[v] = true;
                  }
              }
          }

```

```
300 1          return false;
301        }
302    
303        // Returns the maximum flow from s to t in the given
304        // graph
305        int fordFulkerson(int graph[][], int s, int t, int V)
306        {
307            int u, v;
308    
309            int rGraph[][] = new int[V][V];
310    
311 3          for (u = 0; u < V; u++)
312            {
313 3              for (v = 0; v < V; v++)
314                {
315                    rGraph[u][v] = graph[u][v];
316                }
317            }
318            // This array is filled by BFS and to store path
319            int parent[] = new int[V];
320    
321            int max_flow = 0; // There is no flow initially
322    
323            // Augment the flow while there is path from source
324            // to sink
325 1          while (bfs(rGraph, s, t, parent, V)) {
326    
327                int path_flow = Integer.MAX_VALUE;
328    
329 1              for (v = t; v != s; v = parent[v])
330                {
331                    u = parent[v];
332                    path_flow = Math.min(path_flow, rGraph[u][v]);
333                }
334    
335                // update residual capacities of the edges and
336                // reverse edges along the path
337 1              for (v = t; v != s; v = parent[v])
338                {
339                    u = parent[v];
340 1                  rGraph[u][v] -= path_flow;
341 1                  rGraph[v][u] += path_flow;
342                }
343    
344                // Add path flow to overall flow
345 1              max_flow += path_flow;
346            }
347    
348            // Return the overall flow
349 1          return max_flow;
350        }
351    
352        void topologicalSortUtil(int v, boolean visited[],
353                Stack<Integer> stack, ArrayList<ArrayList<Integer> > adj)
354            {
355                    // Mark the current node as visited.
356                    visited[v] = true;
357                    Integer i;
358    
359                    // Recur for all the vertices adjacent
360                    // to thisvertex
361                    Iterator<Integer> it = adj.get(v).iterator();
362 1                  while (it.hasNext())
363                    {
364                            i = it.next();
365 1                          if (!visited[i])
366                            {
367 1                                  topologicalSortUtil(i, visited, stack, adj);
368                            }
369                    }
370    
371                    // Push current vertex to stack
372                    // which stores result
373                    stack.push(v);
374            }
375    
```

```
        // The function to do Topological Sort.
        // It uses recursive topologicalSortUtil()
        int[] topologicalSort(int V, ArrayList<ArrayList<Integer> > adj)
        {
                Stack<Integer> stack = new Stack<Integer>();

                // Mark all the vertices as not visited
                boolean visited[] = new boolean[V];

                for (int i = 0; i < V; i++)
                {
                        visited[i] = false;
                }

                // Call the recursive helper
                // function to store
                // Topological Sort starting
                // from all vertices one by one
                for (int i = 0; i < V; i++)
                {
                        if (visited[i] == false)
                        {
                                        topologicalSortUtil(i, visited, stack, adj);
                        }
                }

                int []a = new int[stack.size()];

                int i = 0;

                while (stack.empty() == false)
                {
                        a[i++] = stack.pop();
                }

                return a;
        }

        void bridgeUtil(int u, boolean visited[], int disc[],int low[], int parent[], int time, ArrayList<ArrayList<Int
        {

                // Mark the current node as visited
                visited[u] = true;

                // Initialize discovery time and low value
                disc[u] = low[u] = ++time;

                // Go through all vertices adjacent to this
                Iterator<Integer> i = adj[u].iterator();
                while (i.hasNext())
                {
                    int v = i.next();  // v is current adjacent of u

                    // If v is not visited yet, then make it a child
                    // of u in DFS tree and recur for it.
                    // If v is not visited yet, then recur for it
                    if (!visited[v])
                    {
                        parent[v] = u;
                        bridgeUtil(v, visited, disc, low, parent, time, bridges,adj);

                        // Check if the subtree rooted with v has a
                        // connection to one of the ancestors of u
                        low[u]  = Math.min(low[u], low[v]);

                        // If the lowest vertex reachable from subtree
                        // under v is below u in DFS tree, then u-v is
                        // a bridge
                        if (low[v] > disc[u])
                        {
                            ArrayList<Integer> inner = new ArrayList<Integer>();
                            inner.add(u);
                            inner.add(v);
                            bridges.add(inner);
                        }
                    }

                    // Update low value of u for parent function calls.
1                   else if (v != parent[u])
                    {
                        low[u]  = Math.min(low[u], disc[v]);
                    }
                }
        }


        void bridge(LinkedList<Integer> adj[], int V, ArrayList<ArrayList<Integer>> bridges)
        {
                boolean visited[] = new boolean[V];
                int disc[] = new int[V];
                int low[] = new int[V];
                int parent[] = new int[V];


3               for (int i = 0; i < V; i++)
                {
                    parent[i] = -1;
                    visited[i] = false;
                }

3               for (int i = 0; i < V; i++)
                {
1                   if (visited[i] == false)
                    {
1                       bridgeUtil(i, visited, disc, low, parent,0,bridges,adj);
                    }
                }
        }

        public class Graph
        {
            private int V;   // No. of vertices
            private LinkedList<Integer> adj[]; //Adjacency List

            //Constructor
            Graph(int v)
            {
                V = v;
                adj = new LinkedList[v];
3               for (int i=0; i<v; ++i)
                {
                    adj[i] = new LinkedList();
                }
            }

            //Function to add an edge into the graph
  //          void addEdge(int v, int w)  { adj[v].add(w); }

            // A recursive function to print DFS starting from v
            void DFSUtil(int v,boolean visited[],LinkedList<Integer> adj[])
            {
                // Mark the current node as visited and print it
                visited[v] = true;

                int n;

                // Recur for all the vertices adjacent to this vertex
                Iterator<Integer> i =adj[v].iterator();
1               while (i.hasNext())
                {
                    n = i.next();
1                   if (!visited[n])
                    {
1                       DFSUtil(n,visited,adj);
                    }
                }
            }

            // Function that returns reverse (or transpose) of this graph
            Graph getTranspose(LinkedList<Integer> adj[])
            {
                Graph g = new Graph(V);
```

```
3             for (int v = 0; v < V; v++)
              {
                  // Recur for all the vertices adjacent to this vertex
                  Iterator<Integer> i =adj[v].listIterator();
1                 while(i.hasNext())
                  {
                      g.adj[i.next()].add(v);
                  }
              }
1             return g;
          }

          void fillOrder(int v, boolean visited[], Stack stack,LinkedList<Integer> adj[])
          {
              // Mark the current node as visited and print it
              visited[v] = true;

              // Recur for all the vertices adjacent to this vertex
              Iterator<Integer> i = adj[v].iterator();
1             while (i.hasNext())
              {
                  int n = i.next();
1                 if(!visited[n])
                  {
1                     fillOrder(n, visited, stack,adj);
                  }
              }

              stack.push(v);
          }

          int getSCCsCount(int V,LinkedList<Integer> adj[])
          {
              Stack stack = new Stack();

              boolean visited[] = new boolean[V];

2             for(int i = 0; i < V; i++)
              {
                      visited[i] = false;
              }

2             for (int i = 0; i < V; i++)
              {
1                 if (visited[i] == false)
                  {
1                     fillOrder(i, visited, stack,adj);
                  }
              }
              Graph gr = getTranspose(adj);

2             for (int i = 0; i < V; i++)
              {
                  visited[i] = false;
              }

              int answer = 0;

1             while (stack.empty() == false)
              {
                  int v = (int)stack.pop();

1                 if (visited[v] == false)
                  {
1                     gr.DFSUtil(v, visited,gr.adj);
1                     answer++;
                  }
              }
1             return answer;
          }


      }


      public class Find_Centroid
```

```
604           {
605               static final int MAXN=100_005;
606               ArrayList<Integer>[] graph;
607               static int[] depth,parent;  // Step 2
608               static int N;
609
610               public Find_Centroid(ArrayList<Integer>[] graph,int N)
611               {
612                   Find_Centroid.N = N;
613                   this.graph =  graph;
614               }
615
616
617               static int[] queue=new int[MAXN],leftOver;
618                                   // Step 3
619
620               static int findCentroid(int r,ArrayList<Integer>[] graph)
621               {
622                   leftOver=new int[N];
623 1                 int i,target=N/2,ach=-1;
624
625 1                 bfs(r,graph);     // Step 4
626 4                 for(i=N-1;i>=0;--i)
627 1                     if(queue[i]!=r)
628 2                         leftOver[parent[queue[i]]] += leftOver[queue[i]] +1;
629                               // Step 5
630 3                 for(i=0;i<N;++i)
631 2                     leftOver[i] = N-1 -leftOver[i];
632                               // Step 6
633 3                 for(i=0;i<N;++i)
634 4                     if(leftOver[i]<=target && leftOver[i]>ach)
635                               // Closest to target(=N/2) but does not exceed it.
636                       {
637                           r=i;    ach=leftOver[i];
638                       }
639                               // Step 7
640 1                 return r;
641               }
642               static void bfs(int root,ArrayList<Integer>[] graph)   // Iterative
643               {
644                   parent=new int[N];  depth=new int[N];
645                   int st=0,end=0;
646                   parent[root]=-1;    depth[root]=1;
647                           // Parent of root is obviously undefined. Hence -1.
648                           // Assuming depth of root = 1
649 1                 queue[end++]=root;
650 2                 while(st<end)
651                   {
652 2                     int node = queue[st++], h = depth[node]+1;
653                       Iterator<Integer> itr=graph[node].iterator();
654 1                     while(itr.hasNext())
655                       {
656                           int ch=itr.next();
657 2                         if(depth[ch]>0)     // 'ch' is parent of 'node'
658                               continue;
659                           depth[ch]=h;   parent[ch]=node;
660 1                         queue[end++]=ch;    // Recording the Traversal sequence
661                       }
662                   }
663               }
664           }
665
666
667       }
668
```

Mutations

```
      1. changed conditional boundary → KILLED
11    2. Changed increment from 1 to -1 → KILLED
      3. negated conditional → KILLED
      1. changed conditional boundary → KILLED
13    2. Changed increment from 1 to -1 → KILLED
      3. negated conditional → KILLED
15    1. changed conditional boundary → KILLED
      2. Changed increment from 1 to -1 → KILLED
      3. negated conditional → KILLED
```

```
    1. changed conditional boundary → SURVIVED
17  2. Replaced integer addition with subtraction → KILLED
    3. negated conditional → KILLED
19  1. Replaced integer addition with subtraction → KILLED
25  1. replaced return value with null for com/softwareTesting/GraphAlgorithms::floydWarshall → KILLED
    1. changed conditional boundary → KILLED
34  2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → KILLED
    1. changed conditional boundary → SURVIVED
36  2. negated conditional → KILLED
    3. negated conditional → KILLED
43  1. replaced int return with 0 for com/softwareTesting/GraphAlgorithms::minKey → KILLED
    1. changed conditional boundary → KILLED
50  2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → KILLED
52  1. Replaced integer addition with subtraction → KILLED
54  1. replaced int return with 0 for com/softwareTesting/GraphAlgorithms::totalMST → KILLED
    1. changed conditional boundary → KILLED
68  2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → KILLED
    1. changed conditional boundary → SURVIVED
80  2. Changed increment from 1 to -1 → KILLED
    3. Replaced integer subtraction with addition → KILLED
    4. negated conditional → KILLED
    1. changed conditional boundary → KILLED
88  2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → KILLED
    1. changed conditional boundary → SURVIVED
92  2. negated conditional → KILLED
    3. negated conditional → KILLED
    4. negated conditional → KILLED
100 1. replaced int return with 0 for com/softwareTesting/GraphAlgorithms::primMST → KILLED
    1. changed conditional boundary → KILLED
108 2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → KILLED
    1. changed conditional boundary → SURVIVED
110 2. negated conditional → KILLED
    3. negated conditional → KILLED
116 1. replaced int return with 0 for com/softwareTesting/GraphAlgorithms::minDistance → KILLED
    1. changed conditional boundary → KILLED
125 2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → KILLED
    1. changed conditional boundary → SURVIVED
132 2. Changed increment from 1 to -1 → KILLED
    3. Replaced integer subtraction with addition → KILLED
    4. negated conditional → KILLED
    1. changed conditional boundary → KILLED
141 2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → KILLED
142 1. negated conditional → KILLED
    2. negated conditional → KILLED
143 1. negated conditional → KILLED
    1. changed conditional boundary → SURVIVED
144 2. Replaced integer addition with subtraction → KILLED
    3. negated conditional → KILLED
145 1. Replaced integer addition with subtraction → KILLED
147 1. replaced return value with null for com/softwareTesting/GraphAlgorithms::dijkstra → KILLED
159 1. negated conditional → KILLED
161 1. removed call to com/softwareTesting/GraphAlgorithms::DFSUtil → KILLED
    1. changed conditional boundary → KILLED
171 2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → KILLED
173 1. negated conditional → KILLED
177 1. removed call to com/softwareTesting/GraphAlgorithms::DFSUtil → KILLED
178 1. Changed increment from 1 to -1 → KILLED
182 1. replaced int return with 0 for com/softwareTesting/GraphAlgorithms::connectedComponents → KILLED
191 1. Changed increment from 1 to -1 → SURVIVED
195 1. negated conditional → KILLED
198 1. negated conditional → KILLED
200 1. Changed increment from 1 to -1 → SURVIVED
202 1. negated conditional → KILLED
204 1. replaced boolean return with false for com/softwareTesting/GraphAlgorithms::isBCUtil → KILLED
    1. changed conditional boundary → KILLED
209 2. negated conditional → SURVIVED
    3. negated conditional → KILLED
211 1. replaced boolean return with false for com/softwareTesting/GraphAlgorithms::isBCUtil → NO_COVERAGE
    1. changed conditional boundary → SURVIVED
213 2. negated conditional → KILLED
    3. negated conditional → KILLED
215 1. replaced boolean return with false for com/softwareTesting/GraphAlgorithms::isBCUtil → SURVIVED
218 1. negated conditional → KILLED
223 1. replaced boolean return with true for com/softwareTesting/GraphAlgorithms::isBCUtil → KILLED
233 1. changed conditional boundary → KILLED
```

```
    2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → KILLED
240 1. negated conditional → KILLED
242 1. replaced boolean return with true for com/softwareTesting/GraphAlgorithms::isBC → KILLED
    1. changed conditional boundary → KILLED
245 2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → SURVIVED
247 1. negated conditional → KILLED
249 1. replaced boolean return with true for com/softwareTesting/GraphAlgorithms::isBC → NO_COVERAGE
252 1. replaced boolean return with false for com/softwareTesting/GraphAlgorithms::isBC → KILLED
    1. changed conditional boundary → KILLED
266 2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → SURVIVED
280 1. negated conditional → KILLED
    1. changed conditional boundary → KILLED
284 2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → KILLED
    1. changed conditional boundary → TIMED_OUT
286 2. negated conditional → KILLED
    3. negated conditional → TIMED_OUT
288 1. negated conditional → TIMED_OUT
291 1. replaced boolean return with false for com/softwareTesting/GraphAlgorithms::bfs → KILLED
300 1. replaced boolean return with true for com/softwareTesting/GraphAlgorithms::bfs → TIMED_OUT
    1. changed conditional boundary → KILLED
311 2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → KILLED
    1. changed conditional boundary → KILLED
313 2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → KILLED
325 1. negated conditional → KILLED
329 1. negated conditional → KILLED
337 1. negated conditional → TIMED_OUT
340 1. Replaced integer subtraction with addition → KILLED
341 1. Replaced integer addition with subtraction → SURVIVED
345 1. Replaced integer addition with subtraction → KILLED
349 1. replaced int return with 0 for com/softwareTesting/GraphAlgorithms::fordFulkerson → KILLED
362 1. negated conditional → KILLED
365 1. negated conditional → KILLED
367 1. removed call to com/softwareTesting/GraphAlgorithms::topologicalSortUtil → KILLED
385 1. changed conditional boundary → KILLED
    2. negated conditional → SURVIVED
394 1. changed conditional boundary → KILLED
    2. negated conditional → KILLED
396 1. negated conditional → KILLED
398 1. removed call to com/softwareTesting/GraphAlgorithms::topologicalSortUtil → KILLED
406 1. negated conditional → KILLED
408 1. Changed increment from 1 to -1 → KILLED
411 1. replaced return value with null for com/softwareTesting/GraphAlgorithms::topologicalSort → KILLED
421 1. Changed increment from 1 to -1 → KILLED
425 1. negated conditional → KILLED
432 1. negated conditional → KILLED
435 1. removed call to com/softwareTesting/GraphAlgorithms::bridgeUtil → KILLED
444 1. changed conditional boundary → KILLED
    2. negated conditional → KILLED
454 1. negated conditional → KILLED
    1. changed conditional boundary → KILLED
470 2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → SURVIVED
    1. changed conditional boundary → KILLED
476 2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → KILLED
478 1. negated conditional → KILLED
480 1. removed call to com/softwareTesting/GraphAlgorithms::bridgeUtil → KILLED
    1. changed conditional boundary → KILLED
495 2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → KILLED
514 1. negated conditional → KILLED
517 1. negated conditional → KILLED
519 1. removed call to com/softwareTesting/GraphAlgorithms$Graph::DFSUtil → KILLED
    1. changed conditional boundary → KILLED
528 2. Changed increment from 1 to -1 → KILLED
    3. negated conditional → KILLED
532 1. negated conditional → KILLED
537 1. replaced return value with null for com/softwareTesting/GraphAlgorithms$Graph::getTranspose → KILLED
547 1. negated conditional → KILLED
550 1. negated conditional → KILLED
552 1. removed call to com/softwareTesting/GraphAlgorithms$Graph::fillOrder → KILLED
565 1. changed conditional boundary → KILLED
    2. negated conditional → SURVIVED
570 1. changed conditional boundary → KILLED
    2. negated conditional → KILLED
```

```
572  1. negated conditional → KILLED
574  1. removed call to com/softwareTesting/GraphAlgorithms$Graph::fillOrder → KILLED
579  1. changed conditional boundary → KILLED
     2. negated conditional → KILLED
586  1. negated conditional → KILLED
590  1. negated conditional → KILLED
592  1. removed call to com/softwareTesting/GraphAlgorithms$Graph::DFSUtil → KILLED
593  1. Changed increment from 1 to -1 → KILLED
596  1. replaced int return with 0 for com/softwareTesting/GraphAlgorithms$Graph::getSCCsCount → KILLED
623  1. Replaced integer division with multiplication → KILLED
625  1. removed call to com/softwareTesting/GraphAlgorithms$Find_Centroid::bfs → SURVIVED
     1. changed conditional boundary → SURVIVED
626  2. Changed increment from -1 to 1 → KILLED
     3. Replaced integer subtraction with addition → SURVIVED
     4. negated conditional → KILLED
627  1. negated conditional → KILLED
628  1. Replaced integer addition with subtraction → KILLED
     2. Replaced integer addition with subtraction → KILLED
     1. changed conditional boundary → KILLED
630  2. Changed increment from 1 to -1 → KILLED
     3. negated conditional → SURVIVED
631  1. Replaced integer subtraction with addition → KILLED
     2. Replaced integer subtraction with addition → KILLED
     1. changed conditional boundary → KILLED
633  2. Changed increment from 1 to -1 → KILLED
     3. negated conditional → SURVIVED
     1. changed conditional boundary → SURVIVED
634  2. changed conditional boundary → SURVIVED
     3. negated conditional → KILLED
     4. negated conditional → KILLED
640  1. replaced int return with 0 for com/softwareTesting/GraphAlgorithms$Find_Centroid::findCentroid → KILLED
649  1. Changed increment from 1 to -1 → KILLED
650  1. changed conditional boundary → SURVIVED
     2. negated conditional → KILLED
652  1. Changed increment from 1 to -1 → KILLED
     2. Replaced integer addition with subtraction → KILLED
654  1. negated conditional → KILLED
657  1. changed conditional boundary → KILLED
     2. negated conditional → KILLED
660  1. Changed increment from 1 to -1 → KILLED
```

Active mutators

- BOOLEAN_FALSE_RETURN
- BOOLEAN_TRUE_RETURN
- CONDITIONALS_BOUNDARY_MUTATOR
- EMPTY_RETURN_VALUES
- INCREMENTS_MUTATOR
- INVERT_NEGS_MUTATOR
- MATH_MUTATOR
- NEGATE_CONDITIONALS_MUTATOR
- NULL_RETURN_VALUES
- PRIMITIVE_RETURN_VALS_MUTATOR
- VOID_METHOD_CALL_MUTATOR

Tests examined

- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestSCC()] (21 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestTopologicalSort2()] (36 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:testConnectedComp()] (16 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestBridge1()] (15 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestPrims1()] (27 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestisBc2()] (23 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestBFS1()] (30 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestDijsktra1()] (31 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestDijsktra2()] (23 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestBridge2()] (15 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestFindCentroid()] (23 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestisBc()] (43 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestTopologicalSort1()] (23 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:testMaxFlow()] (13 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestPrims2()] (42 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestFlloyd()] (13 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:testConnectedComp2()] (25 ms)

Report generated by PIT 1.6.8