TreeAlgoritms.java

```
1    package com.softwareTesting;
2
3    import java.util.ArrayList;
4    import java.util.Arrays;
5    import java.util.Collections;
6    import java.util.Iterator;
7    import java.util.LinkedList;
8    import java.util.List;
9
10   import TreeAlgoritms.TreesTraverse.Node;
11   import TreeAlgoritms.TreesTraverse.Trie;
12   import TreeAlgoritms.TreesTraverse.KruskalAlgorithm.Edge;
13   import TreeAlgoritms.TreesTraverse.KruskalAlgorithm.Subset;
14
15   public class TreeAlgoritms {
16           void DFS(int vertex, boolean vis[], ArrayList<Integer> order, LinkedList<Integer> adj[])
17           {
18
19                   vis[vertex] = true;
20           order.add(vertex);
21           Iterator<Integer> it = adj[vertex].listIterator();
22
23 1         while (it.hasNext())
24           {
25             int n = it.next();
26
27 1           if (!vis[n])
28             {
29 1                 DFS(n, vis, order, adj);
30             }
31           }
32
33       }
34
35
36
37
38       ArrayList<Integer> BFS(int s, LinkedList<Integer> adj[],int V)
39       {
40 1         boolean visited[] = new boolean[V+1];
41
42           LinkedList<Integer> queue = new LinkedList<Integer>();
43           ArrayList<Integer> order = new ArrayList<Integer>();
44
45           visited[s]=true;
46           queue.add(s);
47
48 1         while (queue.size() != 0)
49           {
50
51               s = queue.poll();
52
53               order.add(s);
54
55               Iterator<Integer> i = adj[s].listIterator();
56 1            while (i.hasNext())
57               {
58                   int n = i.next();
59
60 1                if (!visited[n])
61                   {
62                       visited[n] = true;
63                       queue.add(n);
64                   }
65               }
66           }
67 1         return order;
68       }
69
70       int getMinWeight(int graph[][], int V)
71       {
```

```
72          int minEdge = Integer.MAX_VALUE;
73  3       for(int i=0;i<V;i++)
74          {
75  3          for(int j=0;j<V;j++)
76              {
77                  minEdge = Math.min(minEdge, graph[i][j]);
78              }
79          }
80  1       return minEdge;
81      }
82
83      int getMaxWeight(int graph[][], int V)
84      {
85          int maxEdge = Integer.MIN_VALUE;
86
87  3       for(int i=0;i<V;i++)
88          {
89  3          for(int j=0;j<V;j++)
90              {
91                  maxEdge = Math.max(maxEdge, graph[i][j]);
92              }
93          }
94  1       return maxEdge;
95      }
96
97      class Node
98      {
99          int data;
100         Node left, right;
101
102         public Node(int item)
103         {
104             data = item;
105             left = right = null;
106         }
107     }
108
109     class BinaryTree
110     {
111         Node root;
112
113         int diameter(Node root)
114         {
115 1           if (root == null)
116             {
117                 return 0;
118
119             }
120
121             // get the height of left and right sub-trees
122             int lheight = height(root.left);
123             int rheight = height(root.right);
124
125             // get the diameter of left and right sub-trees
126             int ldiameter = diameter(root.left);
127             int rdiameter = diameter(root.right);
128
129 3           return Math.max(lheight + rheight + 1,
130                             Math.max(ldiameter, rdiameter));
131         }
132
133         int diameter()
134         {
135 1               return diameter(root);
136         }
137
138         static int height(Node node)
139         {
140 1           if (node == null)
141                 return 0;
142
143 2           return (1
144                     + Math.max(height(node.left),
145                                height(node.right)));
146         }
147     }
```

```
148
149
150     public int largestRectangleArea(int[] heights) {
151         int n = heights.length;
152 1       if(n == 0)
153         {
154                 return 0;
155         }
156         int maxArea = 0;
157         int left[] = new int[n];
158         int right[] = new int[n];
159
160         left[0] = -1;
161 1       right[n - 1] = n;
162
163 3       for(int i = 1; i < n; i++)
164         {
165 1           int prev = i - 1;
166 4           while(prev >= 0 && heights[prev] >= heights[i])
167             {
168                 prev = left[prev];
169             }
170             left[i] = prev;
171         }
172
173 4       for(int i = n - 2; i >= 0; i--)
174         {
175 1           int prev = i + 1;
176 4           while(prev < n && heights[prev] >= heights[i])
177             {
178                 prev = right[prev];
179             }
180             right[i] = prev;
181         }
182
183 3       for(int i = 0; i < n; i++)
184         {
185 2           int width = right[i] - left[i] - 1;
186 1           maxArea = Math.max(maxArea, heights[i] * width);
187         }
188 1       return maxArea;
189
190     }
191
192
193         public int findMedianSortedArrays(int[] nums1, int[] nums2) {
194           int len1 = 0, len2 = 0;
195
196 2         if(nums1 == null && nums2 == null)
197           {
198               return 0;
199           }
200 1         else if(nums1 == null)
201           {
202               len2 = nums2.length;
203           }
204 1         else if(nums2 == null)
205           {
206               len1 = nums1.length;
207           }
208           else
209           {
210               len1 = nums1.length;
211               len2 = nums2.length;
212           }
213
214 3         if((len1 + len2) % 2 == 0)
215           {
216 8             return (int)((findKth(nums1, 0, len1, nums2, 0, len2, (len1 + len2)/2) + findKth(nums1, 0, len1, nums2, 0
217           }
218           else
219           {
220 4             return (int)(findKth(nums1, 0, len1, nums2, 0, len2, (len1 + len2)/2 + 1));
221           }
222         }
223
```

```
224         public int findKth(int[] n1, int start1, int len1, int[] n2, int start2, int len2, int k) {
225 2         if(len1 > len2)
226         {
227 1               return findKth(n2, start2, len2, n1, start1, len1, k);
228         }
229
230 1         if(len1 == 0)
231         {
232 3               return n2[start2 + k - 1];
233         }
234 1         if(k == 1)
235         {
236 1               return Math.min(n1[start1], n2[start2]);
237         }
238
239 2         int p1 = Math.min(len1, k / 2), p2 = k - p1;
240 4         int num1 = n1[start1 + p1 - 1], num2 = n2[start2 + p2 - 1];
241
242 1         if(num1 == num2)
243         {
244 1             return num1;
245         }
246 2         else if(num1 < num2)
247         {
248 4             return findKth(n1, start1 + p1, len1 - p1, n2, start2, len2, k - p1);
249         }
250         else
251         {
252 4             return findKth(n1, start1, len1, n2, start2 + p2, len2 - p2, k - p2);
253         }
254       }
255
256
257     //Kruskals
258
259         public class KruskalAlgorithm {
260             class Edge implements Comparable<Edge>
261             {
262                 int source, destination, weight;
263
264                 public int compareTo(Edge edgeToCompare)
265                 {
266 2                   return this.weight - edgeToCompare.weight;
267                 }
268             };
269
270             class Subset
271             {
272                 int parent, value;
273             };
274
275             int vertices, edges;
276             Edge edgeArray[];
277
278             KruskalAlgorithm(int vertices, int edges)
279             {
280                 this.vertices = vertices;
281                 this.edges = edges;
282                 edgeArray = new Edge[this.edges];
283 3               for (int i = 0; i < edges; ++i)
284                 {
285                     edgeArray[i] = new Edge();
286                 }
287             }
288
289             public int applyKruskal() {
290
291                 Edge finalResult[] = new Edge[vertices];
292                 int newEdge = 0;
293                 int index = 0;
294 3               for (index = 0; index < vertices; ++index)
295                 {
296                     finalResult[index] = new Edge();
297                 }
298
299 1               Arrays.sort(edgeArray);
```

```
300
301                 Subset subsetArray[] = new Subset[vertices];
302
303 3               for (index = 0; index < vertices; ++index)
304                 {
305                     subsetArray[index] = new Subset();
306                 }
307
308 3               for (int vertex = 0; vertex < vertices; ++vertex)
309                 {
310                     subsetArray[vertex].parent = vertex;
311                     subsetArray[vertex].value = 0;
312                 }
313                 index = 0;
314
315 3               while (newEdge < vertices - 1)
316                 {
317                     Edge nextEdge = new Edge();
318 1                   nextEdge = edgeArray[index++];
319
320                     int nextSource = findSetOfElement(subsetArray, nextEdge.source);
321                     int nextDestination = findSetOfElement(subsetArray, nextEdge.destination);
322
323 1                   if (nextSource != nextDestination)
324                     {
325 1                       finalResult[newEdge++] = nextEdge;
326 1                       performUnion(subsetArray, nextSource, nextDestination);
327                     }
328                 }
329                 int answer = 0;
330
331 3               for (index = 0; index < newEdge; ++index)
332                 {
333 1                       answer += finalResult[index].weight;
334                 }
335
336 1               return answer;
337
338             }
339
340             int findSetOfElement(Subset subsetArray[], int i)
341             {
342 1               if (subsetArray[i].parent != i)
343                 {
344                     subsetArray[i].parent = findSetOfElement(subsetArray, subsetArray[i].parent);
345                 }
346 1               return subsetArray[i].parent;
347             }
348
349             void performUnion(Subset subsetArray[], int sourceRoot, int destinationRoot) {
350
351                 int nextSourceRoot = findSetOfElement(subsetArray, sourceRoot);
352                 int nextDestinationRoot = findSetOfElement(subsetArray, destinationRoot);
353
354 2               if (subsetArray[nextSourceRoot].value < subsetArray[nextDestinationRoot].value)
355                 {
356                     subsetArray[nextSourceRoot].parent = nextDestinationRoot;
357                 }
358 2               else if (subsetArray[nextSourceRoot].value > subsetArray[nextDestinationRoot].value)
359                 {
360                     subsetArray[nextDestinationRoot].parent = nextSourceRoot;
361                 }
362                 else
363                 {
364                     subsetArray[nextDestinationRoot].parent = nextSourceRoot;
365 1                   subsetArray[nextSourceRoot].value++;
366                 }
367             }
368         }
369
370
371         public class Trie
372         {
373             private static final int CHAR_SIZE = 26;
374
375             private boolean isLeaf;
```

```
376             private List<Trie> children = null;
377
378             Trie()
379             {
380                 isLeaf = false;
381                 children = new ArrayList<>(Collections.nCopies(CHAR_SIZE, null));
382             }
383
384             public void insert(String key)
385             {
386
387                 Trie curr = this;
388
389                 for (char c: key.toCharArray())
390                 {
391 2                   if (curr.children.get(c - 'a') == null)
392                     {
393 1                       curr.children.set(c - 'a', new Trie());
394                     }
395
396 1                   curr = curr.children.get(c - 'a');
397                 }
398
399                 curr.isLeaf = true;
400             }
401
402             public boolean search(String key)
403             {
404
405                 Trie curr = this;
406
407                 for (char c: key.toCharArray())
408                 {
409 1                   curr = curr.children.get(c - 'a');
410
411 1                   if (curr == null)
412                     {
413 1                       return false;
414                     }
415                 }
416
417 2               return curr.isLeaf;
418             }
419         }
420 }
```

Mutations

23 1. negated conditional → KILLED
27 1. negated conditional → KILLED
29 1. removed call to com/softwareTesting/TreeAlgoritms::DFS → KILLED
40 1. Replaced integer addition with subtraction → KILLED
48 1. negated conditional → KILLED
56 1. negated conditional → KILLED
60 1. negated conditional → KILLED
67 1. replaced return value with null for com/softwareTesting/TreeAlgoritms::BFS → KILLED
73 1. changed conditional boundary → KILLED
2. Changed increment from 1 to -1 → KILLED
3. negated conditional → KILLED
75 1. changed conditional boundary → KILLED
2. Changed increment from 1 to -1 → KILLED
3. negated conditional → KILLED
80 1. replaced int return with 0 for com/softwareTesting/TreeAlgoritms::getMinWeight → KILLED
87 1. changed conditional boundary → KILLED
2. Changed increment from 1 to -1 → KILLED
3. negated conditional → KILLED
89 1. changed conditional boundary → KILLED
2. Changed increment from 1 to -1 → KILLED
3. negated conditional → KILLED
94 1. replaced int return with 0 for com/softwareTesting/TreeAlgoritms::getMaxWeight → KILLED
115 1. negated conditional → NO_COVERAGE
129 1. Replaced integer addition with subtraction → NO_COVERAGE
2. Replaced integer addition with subtraction → NO_COVERAGE
3. replaced int return with 0 for com/softwareTesting/TreeAlgoritms$BinaryTree::diameter → NO_COVERAGE
135 1. replaced int return with 0 for com/softwareTesting/TreeAlgoritms$BinaryTree::diameter → NO_COVERAGE
140 1. negated conditional → NO_COVERAGE
143 1. Replaced integer addition with subtraction → NO_COVERAGE
2. replaced int return with 0 for com/softwareTesting/TreeAlgoritms$BinaryTree::height → NO_COVERAGE

152	1. negated conditional → KILLED
161	1. Replaced integer subtraction with addition → KILLED
163	1. changed conditional boundary → KILLED 2. Changed increment from 1 to -1 → KILLED 3. negated conditional → KILLED
165	1. Replaced integer subtraction with addition → KILLED
166	1. changed conditional boundary → SURVIVED 2. changed conditional boundary → SURVIVED 3. negated conditional → KILLED 4. negated conditional → KILLED
173	1. changed conditional boundary → SURVIVED 2. Changed increment from -1 to 1 → KILLED 3. Replaced integer subtraction with addition → KILLED 4. negated conditional → KILLED
175	1. Replaced integer addition with subtraction → TIMED_OUT
176	1. changed conditional boundary → KILLED 2. changed conditional boundary → SURVIVED 3. negated conditional → KILLED 4. negated conditional → KILLED
183	1. changed conditional boundary → KILLED 2. Changed increment from 1 to -1 → KILLED 3. negated conditional → KILLED
185	1. Replaced integer subtraction with addition → KILLED 2. Replaced integer subtraction with addition → KILLED
186	1. Replaced integer multiplication with division → KILLED
188	1. replaced int return with 0 for com/softwareTesting/TreeAlgoritms::largestRectangleArea → KILLED
196	1. negated conditional → SURVIVED 2. negated conditional → NO_COVERAGE
200	1. negated conditional → KILLED
204	1. negated conditional → KILLED
214	1. Replaced integer addition with subtraction → SURVIVED 2. Replaced integer modulus with multiplication → KILLED 3. negated conditional → KILLED
216	1. Replaced integer addition with subtraction → KILLED 2. Replaced integer division with multiplication → KILLED 3. Replaced integer addition with subtraction → KILLED 4. Replaced integer division with multiplication → KILLED 5. Replaced integer addition with subtraction → KILLED 6. Replaced integer addition with subtraction → KILLED 7. Replaced integer division with multiplication → KILLED 8. replaced int return with 0 for com/softwareTesting/TreeAlgoritms::findMedianSortedArrays → KILLED
220	1. Replaced integer addition with subtraction → KILLED 2. Replaced integer division with multiplication → KILLED 3. Replaced integer addition with subtraction → KILLED 4. replaced int return with 0 for com/softwareTesting/TreeAlgoritms::findMedianSortedArrays → KILLED
225	1. changed conditional boundary → SURVIVED 2. negated conditional → KILLED
227	1. replaced int return with 0 for com/softwareTesting/TreeAlgoritms::findKth → KILLED
230	1. negated conditional → KILLED
232	1. Replaced integer addition with subtraction → KILLED 2. Replaced integer subtraction with addition → KILLED 3. replaced int return with 0 for com/softwareTesting/TreeAlgoritms::findKth → KILLED
234	1. negated conditional → KILLED
236	1. replaced int return with 0 for com/softwareTesting/TreeAlgoritms::findKth → KILLED
239	1. Replaced integer division with multiplication → KILLED 2. Replaced integer subtraction with addition → KILLED
240	1. Replaced integer addition with subtraction → KILLED 2. Replaced integer subtraction with addition → KILLED 3. Replaced integer addition with subtraction → KILLED 4. Replaced integer subtraction with addition → KILLED
242	1. negated conditional → KILLED
244	1. replaced int return with 0 for com/softwareTesting/TreeAlgoritms::findKth → NO_COVERAGE
246	1. changed conditional boundary → SURVIVED 2. negated conditional → KILLED
248	1. Replaced integer addition with subtraction → KILLED 2. Replaced integer subtraction with addition → KILLED 3. Replaced integer subtraction with addition → KILLED 4. replaced int return with 0 for com/softwareTesting/TreeAlgoritms::findKth → KILLED
252	1. Replaced integer addition with subtraction → KILLED 2. Replaced integer subtraction with addition → SURVIVED 3. Replaced integer subtraction with addition → KILLED 4. replaced int return with 0 for com/softwareTesting/TreeAlgoritms::findKth → KILLED
266	1. Replaced integer subtraction with addition → KILLED 2. replaced int return with 0 for com/softwareTesting/TreeAlgoritms$KruskalAlgorithm$Edge::compareTo → KILLED
283	1. changed conditional boundary → KILLED 2. Changed increment from 1 to -1 → KILLED 3. negated conditional → KILLED
294	1. changed conditional boundary → KILLED 2. Changed increment from 1 to -1 → KILLED 3. negated conditional → SURVIVED
299	1. removed call to java/util/Arrays::sort → KILLED
303	1. changed conditional boundary → KILLED 2. Changed increment from 1 to -1 → KILLED 3. negated conditional → KILLED
308	1. changed conditional boundary → KILLED

	2. Changed increment from 1 to -1 → KILLED 3. negated conditional → KILLED
	1. changed conditional boundary → KILLED
315	2. Replaced integer subtraction with addition → KILLED 3. negated conditional → KILLED
318	1. Changed increment from 1 to -1 → KILLED
323	1. negated conditional → KILLED
325	1. Changed increment from 1 to -1 → KILLED
326	1. removed call to com/softwareTesting/TreeAlgoritms$KruskalAlgorithm::performUnion → SURVIVED
	1. changed conditional boundary → SURVIVED
331	2. Changed increment from 1 to -1 → KILLED 3. negated conditional → KILLED
333	1. Replaced integer addition with subtraction → KILLED
336	1. replaced int return with 0 for com/softwareTesting/TreeAlgoritms$KruskalAlgorithm::applyKruskal → KILLED
342	1. negated conditional → KILLED
346	1. replaced int return with 0 for com/softwareTesting/TreeAlgoritms$KruskalAlgorithm::findSetOfElement → KILLED
354	1. changed conditional boundary → SURVIVED 2. negated conditional → SURVIVED
358	1. changed conditional boundary → SURVIVED 2. negated conditional → SURVIVED
365	1. Replaced integer addition with subtraction → SURVIVED
391	1. Replaced integer subtraction with addition → KILLED 2. negated conditional → KILLED
393	1. Replaced integer subtraction with addition → KILLED
396	1. Replaced integer subtraction with addition → KILLED
409	1. Replaced integer subtraction with addition → KILLED
411	1. negated conditional → KILLED
413	1. replaced boolean return with true for com/softwareTesting/TreeAlgoritms$Trie::search → KILLED
417	1. replaced boolean return with false for com/softwareTesting/TreeAlgoritms$Trie::search → KILLED 2. replaced boolean return with true for com/softwareTesting/TreeAlgoritms$Trie::search → SURVIVED

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:TestfindMedianSortedArrays1()] (21 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:getMaxWeight2()] (18 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestTrie()] (23 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestKrushkalsAlgorithm2()] (19 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestDFS1()] (14 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:getMinWeight2()] (11 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:TestfindMedianSortedArrays2()] (17 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:getMaxWeight1()] (22 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestlargestRectangleArea()] (35 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestKrushkalsAlgorithm()] (81 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:getMinWeight1()] (23 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestBFS2()] (17 ms)
- com.softwareTesting.AlgoritmsTest.[engine:junit-jupiter]/[class:com.softwareTesting.AlgoritmsTest]/[method:TestDFS2()] (19 ms)

Report generated by PIT 1.6.8