

Step: 1 Implementation of height-weighted Quick Union with Path Compression.

Test Case Screenshot:

The screenshot shows the Eclipse IDE interface. On the left, the 'JUnit' test runner window displays a list of tests that passed, including 'testIsConnected01' through 'testIsConnected05', 'testFind0' through 'testFind5', 'testToString', 'testConnect01', 'testConnect02', and 'testConnect03'. The main editor window shows the source code for 'UF_HWQUPC.java'. The code includes a package declaration, imports, and a test class 'UF_HWQUPC_Test' with a 'testToString()' method. The bottom status bar indicates the test run was terminated.

```
20 * Copyright (c) 2017. Phasmid Software[]
4
5 package edu.neu.coe.info6205.union_find;
6
7 import edu.neu.coe.info6205.util.PrivateMethodTester;[]
8
9
10
11
12 public class UF_HWQUPC_Test {
13
14     @Test
15     public void testToString() {
16         Connections h = new UF_HWQUPC(2);
17         assertEquals("UF_HWQUPC:\n" +
18             "    count: 2\n" +
19             "    path compression? true\n" +
20             "    parents: [0, 1]\n" +
21             "    heights: [1, 1]", h.toString());
22     }
23
24     /**
25      *
26      */
27     @Test
28     public void testIsConnected01() {
```

Step 2 : Implementation of UF_HWQUPC, develop a UnionFindClient file.

The screenshot shows the Eclipse IDE interface with the source code for 'UnionFindClient.java'. The code includes a package declaration, imports, and a class 'UnionFindClient' with a 'count()' method and a 'main()' method. The 'count()' method uses a 'UF_HWQUPC' instance to perform operations on a set of nodes. The 'main()' method generates a set of nodes and performs operations on them, printing the results.

```
1 package edu.neu.coe.info6205.union_find;
2 import java.util.Random;
3
4 public class UnionFindClient {
5     public static int count(int n)
6     {
7         int ufcon = 0;
8         UF_HWQUPC uf_hwqupc = new UF_HWQUPC(n);
9         Random ran = new Random();
10         while(uf_hwqupc.components() > 1)
11         {
12             int i = ran.nextInt(n);
13             int j = ran.nextInt(n);
14             uf_hwqupc.connect(i, j);
15             ufcon++;
16         }
17         return ufcon;
18     }
19
20     public static void main(String[] args) {
21         int n[] = new int[] {100,200,300,400,500,600,700,800,900,1000};
22         int m = 100;
23         for(int i = 0; i < n.length; i++)
24         {
25             double sum = 0;
26             for(int j=0; j < m; j++) {
27                 int operation = count(n[i]);
28                 sum += operation;
29             }
30             double average = sum/m;
31             System.out.println("Integer value: " + n[i] + ", number of sites: " + average + " expected value " + 0.53 * n[i] * Math.Log(n[i]) + " ratio " + average / (0.53 * n[i] * Math.Log(n[i])));
32         }
33     }
34 }
```

Result:

```
Problems Javadoc Declaration Console X Progress
<terminated> UnionFindClient [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (Mar 4, 2022, 7:15:46 PM - 7:15:49 PM)
Integer value: 100, number of sites: 263.88 expected value 244.07401985736885 ratio 1.0811474328738688
Integer value: 200, number of sites: 611.7 expected value 561.6216408540919 ratio 1.0891674314218929
Integer value: 300, number of sites: 915.22 expected value 906.9014134703359 ratio 1.0091725367345414
Integer value: 400, number of sites: 1363.05 expected value 1270.190483986892 ratio 1.0731067640513563
Integer value: 500, number of sites: 1717.76 expected value 1646.8711460818806 ratio 1.0430445661075387
Integer value: 600, number of sites: 2117.66 expected value 2034.2236303587345 ratio 1.0410163211143857
Integer value: 700, number of sites: 2463.12 expected value 2430.450804301103 ratio 1.0134416198184646
Integer value: 800, number of sites: 2935.13 expected value 2834.275372531201 ratio 1.0355839197723153
Integer value: 900, number of sites: 3326.64 expected value 3244.742302105696 ratio 1.0252401239510318
Integer value: 1000, number of sites: 3806.16 expected value 3661.1102978605327 ratio 1.0396190473213087
```

Conclusion:

The relationship between number of objects(n) and number of pairs(m):

$$m = 0.53 * n * \lg(n)$$

Below table and graph shows relationship between number of pairs and expected value. It depicts that the relationship between number of pairs and expected values is almost 1.

Number of pairs	Expected Value
263.88	244.074
611.7	561.621
915.22	906.901
1363.05	1270.19
1717.76	1646.871
2117.66	2034.223
2463.12	2430.45
2935.13	2834.275
3326.64	3244.742
3806.16	3661.11

