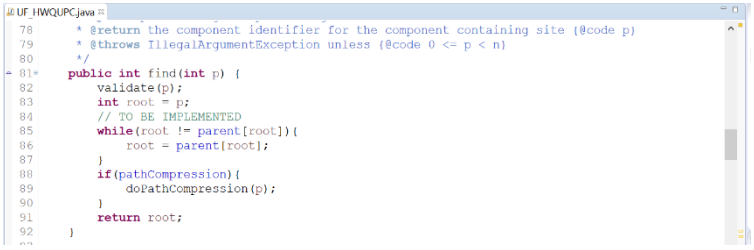
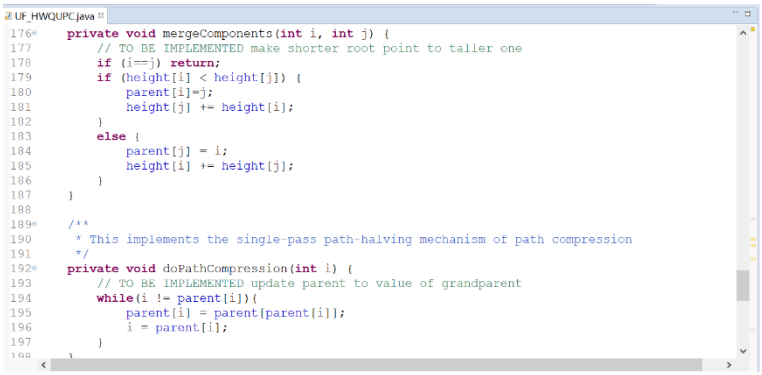
Program Structures & Algorithms
Fall 2021
Assignment No. 3

- **Task (List down the tasks performed in the Assignment)**
  - Step 1: Implement height-weighted Quick Union with Path Compression. For this, you will flesh out the class UF_HWQUPC. Check that the unit tests for this class all work.

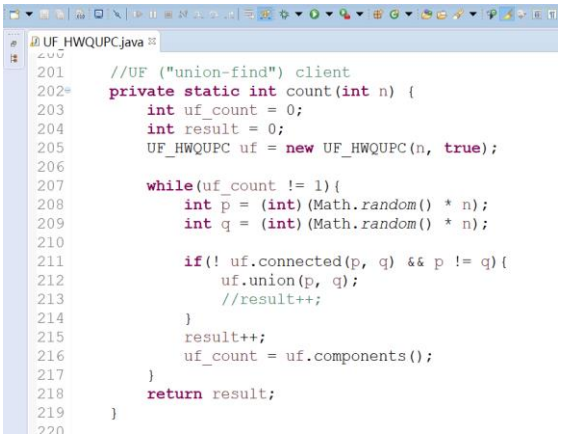


```
78      * @return the component identifier for the component containing site (@code p)
79      * @throws IllegalArgumentException unless (@code 0 <= p < n)
80      */
81     public int find(int p) {
82         validate(p);
83         int root = p;
84         // TO BE IMPLEMENTED
85         while(root != parent[root]){
86             root = parent[root];
87         }
88         if(pathCompression){
89             doPathCompression(p);
90         }
91         return root;
92     }
```

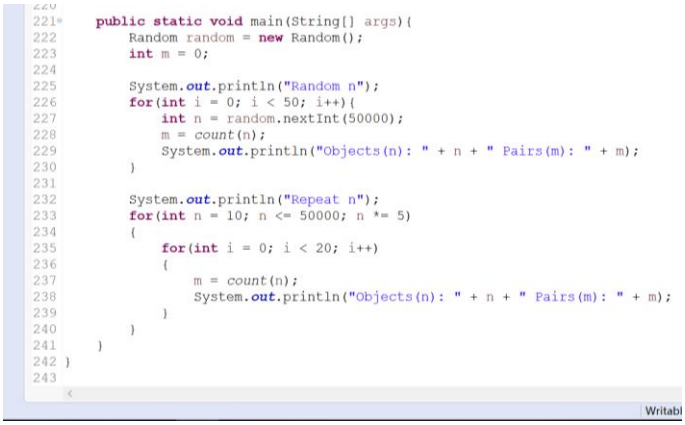


```
176     private void mergeComponents(int i, int j) {
177         // TO BE IMPLEMENTED make shorter root point to taller one
178         if (i==j) return;
179         if (height[i] < height[j]) {
180             parent[i]=j;
181             height[j] += height[i];
182         }
183         else {
184             parent[j] = i;
185             height[i] += height[j];
186         }
187     }
188
189     /**
190      * This implements the single-pass path-halving mechanism of path compression
191      */
192     private void doPathCompression(int i) {
193         // TO BE IMPLEMENTED update parent to value of grandparent
194         while(i != parent[i]){
195             parent[i] = parent[parent[i]];
196             i = parent[i];
197         }
```

  - Step 2: Using the implementation of UF_HWQUPC, develop a UF ("union-find") client that takes an integer value n from the command line to determine the number of "sites." Then generates random pairs of integers between 0 and n-1, calling connected() to determine if they are connected and union() if not. Loop until all sites are connected then print the number of connections generated. Package the program as a static method count() that takes n as the argument and returns the number of connections; and a main() that takes n from the command line, calls count() and prints the returned value.



```
201     //UF ("union-find") client
202     private static int count(int n) {
203         int uf_count = 0;
204         int result = 0;
205         UF_HWQUPC uf = new UF_HWQUPC(n, true);
206
207         while(uf_count != 1){
208             int p = (int)(Math.random() * n);
209             int q = (int)(Math.random() * n);
210
211             if(! uf.connected(p, q) && p != q){
212                 uf.union(p, q);
213                 //result++;
214             }
215             result++;
216             uf_count = uf.components();
217         }
218         return result;
219     }
```



```
221     public static void main(String[] args){
222         Random random = new Random();
223         int m = 0;
224
225         System.out.println("Random n");
226         for(int i = 0; i < 50; i++){
227             int n = random.nextInt(50000);
228             m = count(n);
229             System.out.println("Objects(n): " + n + " Pairs(m): " + m);
230         }
231
232         System.out.println("Repeat n");
233         for(int n = 10; n <= 50000; n *= 5)
234         {
235             for(int i = 0; i < 20; i++)
236             {
237                 m = count(n);
238                 System.out.println("Objects(n): " + n + " Pairs(m): " + m);
239             }
240         }
241     }
242 }
243
```

- Step 3: Determine the relationship between the number of objects (n) and the number of pairs (m) generated to accomplish this (i.e. to reduce the number of components from n to 1). Justify your conclusion in terms of your observations and what you think might be going on.

- **Relationship Conclusion: (For ex : z = a * b)**

m = 1/2*n*ln(n)

I did two ways to observe the relationship. Firstly, I got 50 random objects (n) a related pair (m). Then, I use 6 objects (n) with 20 related repeat pairs (m). By observing the experimental results, it can be concluded that the number of pairs (m) generated to accomplish the number of objects (n) is approximately equal to 0.5*n*ln(n).

- **Evidence to support the conclusion:**

1. **Output (Snapshot of Code output in the terminal)**

2. **Graphical Representation (Observations from experiments should be tabulated and analyzed by plotting graphs(usually in excel) to arrive on the relationship conclusion)**

```
Random n
Objects(n): 28918 Pairs(m): 166529
Objects(n): 36864 Pairs(m): 204178
Objects(n): 1226 Pairs(m): 3651
Objects(n): 5098 Pairs(m): 21053
Objects(n): 15453 Pairs(m): 80928
Objects(n): 781 Pairs(m): 3156
Objects(n): 30662 Pairs(m): 167207
Objects(n): 1300 Pairs(m): 4308
Objects(n): 15967 Pairs(m): 84320
Objects(n): 36491 Pairs(m): 231217
Objects(n): 41602 Pairs(m): 222027
Objects(n): 41981 Pairs(m): 251878
Objects(n): 27439 Pairs(m): 148302
Objects(n): 514 Pairs(m): 2293
Objects(n): 10980 Pairs(m): 48414
Objects(n): 39056 Pairs(m): 253018
Objects(n): 25562 Pairs(m): 140172
Objects(n): 44731 Pairs(m): 285229
Objects(n): 9657 Pairs(m): 41610
Objects(n): 577 Pairs(m): 2045
Objects(n): 44185 Pairs(m): 240579
Objects(n): 25863 Pairs(m): 156110
Objects(n): 15446 Pairs(m): 79946
Objects(n): 30501 Pairs(m): 188813
Objects(n): 13347 Pairs(m): 75503
Objects(n): 1229 Pairs(m): 6356
Objects(n): 13948 Pairs(m): 74981
```

```
Objects(n): 13948 Pairs(m): 74981
Objects(n): 23313 Pairs(m): 118557
Objects(n): 25887 Pairs(m): 145850
Objects(n): 15616 Pairs(m): 99570
Objects(n): 5018 Pairs(m): 18351
Objects(n): 44782 Pairs(m): 295084
Objects(n): 49815 Pairs(m): 311100
Objects(n): 29097 Pairs(m): 171788
Objects(n): 25276 Pairs(m): 128462
Objects(n): 11750 Pairs(m): 75607
Objects(n): 41881 Pairs(m): 236474
Objects(n): 46133 Pairs(m): 230646
Objects(n): 37362 Pairs(m): 267647
Objects(n): 12798 Pairs(m): 66296
Objects(n): 34471 Pairs(m): 201983
Objects(n): 1757 Pairs(m): 6627
Objects(n): 47579 Pairs(m): 247812
Objects(n): 17471 Pairs(m): 78356
Objects(n): 16206 Pairs(m): 83176
Objects(n): 42878 Pairs(m): 208128
Objects(n): 28640 Pairs(m): 139303
Objects(n): 26322 Pairs(m): 138685
Objects(n): 27011 Pairs(m): 153655
Objects(n): 27637 Pairs(m): 147272
Repeat n
```

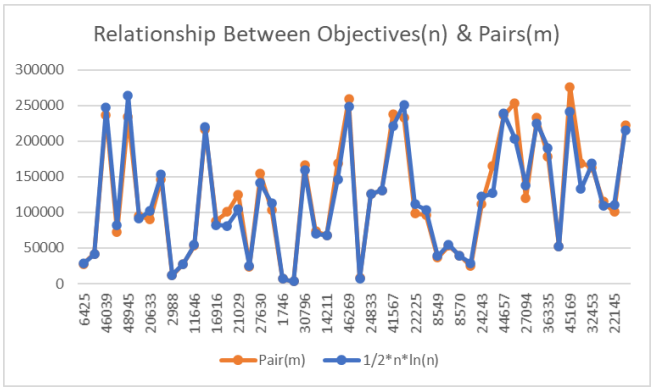
```
Objects(n): 10 Pairs(m): 13 Objects(n): 50 Pairs(m): 108
Objects(n): 10 Pairs(m): 24 Objects(n): 50 Pairs(m): 99
Objects(n): 10 Pairs(m): 18 Objects(n): 50 Pairs(m): 104
Objects(n): 10 Pairs(m): 16 Objects(n): 50 Pairs(m): 109
Objects(n): 10 Pairs(m): 17 Objects(n): 50 Pairs(m): 119
Objects(n): 10 Pairs(m): 15 Objects(n): 50 Pairs(m): 142
Objects(n): 10 Pairs(m): 21 Objects(n): 50 Pairs(m): 82
Objects(n): 10 Pairs(m): 13 Objects(n): 50 Pairs(m): 119
Objects(n): 10 Pairs(m): 18 Objects(n): 50 Pairs(m): 119
Objects(n): 10 Pairs(m): 18 Objects(n): 50 Pairs(m): 103
Objects(n): 10 Pairs(m): 12 Objects(n): 50 Pairs(m): 105
Objects(n): 10 Pairs(m): 15 Objects(n): 50 Pairs(m): 131
Objects(n): 10 Pairs(m): 17 Objects(n): 50 Pairs(m): 85
Objects(n): 10 Pairs(m): 12 Objects(n): 250 Pairs(m): 871
Objects(n): 10 Pairs(m): 31 Objects(n): 250 Pairs(m): 789
Objects(n): 10 Pairs(m): 16 Objects(n): 250 Pairs(m): 643
Objects(n): 10 Pairs(m): 9  Objects(n): 250 Pairs(m): 931
Objects(n): 10 Pairs(m): 11 Objects(n): 250 Pairs(m): 831
Objects(n): 10 Pairs(m): 15 Objects(n): 250 Pairs(m): 1213
Objects(n): 10 Pairs(m): 14 Objects(n): 250 Pairs(m): 875
Objects(n): 50 Pairs(m): 115Objects(n): 250 Pairs(m): 573
Objects(n): 50 Pairs(m): 88 Objects(n): 250 Pairs(m): 747
Objects(n): 50 Pairs(m): 95 Objects(n): 250 Pairs(m): 808
Objects(n): 50 Pairs(m): 70 Objects(n): 250 Pairs(m): 646
Objects(n): 50 Pairs(m): 133Objects(n): 250 Pairs(m): 827
Objects(n): 50 Pairs(m): 143Objects(n): 250 Pairs(m): 659
Objects(n): 50 Pairs(m): 117Objects(n): 250 Pairs(m): 576
Objects(n): 50 Pairs(m): 108Objects(n): 250 Pairs(m): 876
```

```
Objects(n): 250 Pairs(m): 876
Objects(n): 250 Pairs(m): 512
Objects(n): 250 Pairs(m): 937
Objects(n): 250 Pairs(m): 738
Objects(n): 250 Pairs(m): 1270
Objects(n): 250 Pairs(m): 915
Objects(n): 1250 Pairs(m): 4307
Objects(n): 1250 Pairs(m): 4924
Objects(n): 1250 Pairs(m): 3907
Objects(n): 1250 Pairs(m): 5811
Objects(n): 1250 Pairs(m): 5367
Objects(n): 1250 Pairs(m): 5536
Objects(n): 1250 Pairs(m): 4167
Objects(n): 1250 Pairs(m): 4565
Objects(n): 1250 Pairs(m): 5586
Objects(n): 1250 Pairs(m): 4153
Objects(n): 1250 Pairs(m): 5084
Objects(n): 1250 Pairs(m): 4668
Objects(n): 1250 Pairs(m): 4341
Objects(n): 1250 Pairs(m): 6157
Objects(n): 1250 Pairs(m): 4274
Objects(n): 1250 Pairs(m): 7187
Objects(n): 1250 Pairs(m): 3848
Objects(n): 1250 Pairs(m): 5306
Objects(n): 1250 Pairs(m): 4184
Objects(n): 1250 Pairs(m): 5833
Objects(n): 6250 Pairs(m): 32887
Objects(n): 6250 Pairs(m): 25958
```

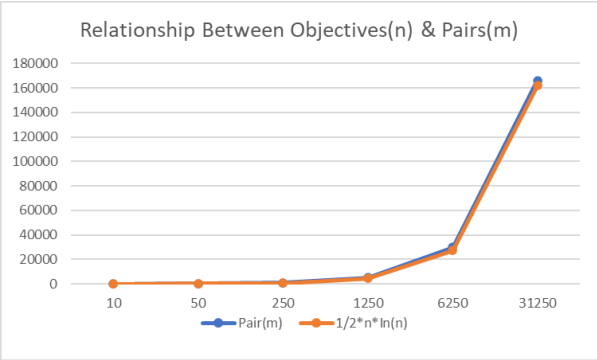
```
Objects(n): 6250 Pairs(m): 25958
Objects(n): 6250 Pairs(m): 26670
Objects(n): 6250 Pairs(m): 28598
Objects(n): 6250 Pairs(m): 25925
Objects(n): 6250 Pairs(m): 28911
Objects(n): 6250 Pairs(m): 29129
Objects(n): 6250 Pairs(m): 29054
Objects(n): 6250 Pairs(m): 26907
Objects(n): 6250 Pairs(m): 34088
Objects(n): 6250 Pairs(m): 29955
Objects(n): 6250 Pairs(m): 29033
Objects(n): 6250 Pairs(m): 31622
Objects(n): 6250 Pairs(m): 28514
Objects(n): 6250 Pairs(m): 36334
Objects(n): 6250 Pairs(m): 39365
Objects(n): 6250 Pairs(m): 29073
Objects(n): 6250 Pairs(m): 26786
Objects(n): 6250 Pairs(m): 28548
Objects(n): 6250 Pairs(m): 32526
Objects(n): 31250 Pairs(m): 164067
Objects(n): 31250 Pairs(m): 157593
Objects(n): 31250 Pairs(m): 163970
Objects(n): 31250 Pairs(m): 147844
Objects(n): 31250 Pairs(m): 179947
Objects(n): 31250 Pairs(m): 185996
Objects(n): 31250 Pairs(m): 204701
Objects(n): 31250 Pairs(m): 158892
Objects(n): 31250 Pairs(m): 181321
```

```
Objects(n): 6250 Pairs(m): 28514
Objects(n): 6250 Pairs(m): 36334
Objects(n): 6250 Pairs(m): 39365
Objects(n): 6250 Pairs(m): 29073
Objects(n): 6250 Pairs(m): 26786
Objects(n): 6250 Pairs(m): 28548
Objects(n): 6250 Pairs(m): 32526
Objects(n): 31250 Pairs(m): 164067
Objects(n): 31250 Pairs(m): 157593
Objects(n): 31250 Pairs(m): 163970
Objects(n): 31250 Pairs(m): 147844
Objects(n): 31250 Pairs(m): 179947
Objects(n): 31250 Pairs(m): 185996
Objects(n): 31250 Pairs(m): 204701
Objects(n): 31250 Pairs(m): 158892
Objects(n): 31250 Pairs(m): 181321
Objects(n): 31250 Pairs(m): 159784
Objects(n): 31250 Pairs(m): 149697
Objects(n): 31250 Pairs(m): 174633
Objects(n): 31250 Pairs(m): 178582
Objects(n): 31250 Pairs(m): 161131
Objects(n): 31250 Pairs(m): 149418
Objects(n): 31250 Pairs(m): 149735
Objects(n): 31250 Pairs(m): 163971
Objects(n): 31250 Pairs(m): 165952
Objects(n): 31250 Pairs(m): 173898
Objects(n): 31250 Pairs(m): 152466
```

I put all the output into the Excel, then averaged the twenty number of pairs (m) with the related of objects (n), and then calculated the error of expected pairs (m) of each object (n).



average					
16.25	109.3	811.85	4978.7	29994.15	166179.9
n					
10	50	250	1250	6250	31250
1/2*n*ln(n)					
11.51293	97.80058	690.1826	4456.812	27313.55	161715.2
Error					
41.15%	11.76%	17.63%	11.71%	9.81%	2.76%



- **Unit tests result:(Snapshot of successful unit test run)**