

Program Structures and Algorithms - Assignment 3
Name - Janhavi Patil | Section - 1 | NU ID - 001523317

Tasks:

Step 1:

(a) Implement height-weighted Quick Union with Path Compression. For this, you will flesh out the class UF_HWQUPC. All you have to do is to fill in the sections marked with

```
// TO BE IMPLEMENTED ... // ... END IMPLEMENTATION.
```

(b) Check that the unit tests for this class all work. You must show "green" test results in your submission (screenshot is OK).

Step 2:

Using your 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 your 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. If you prefer, you can create a main program that doesn't require any input and runs the experiment for a fixed set of n values. Show evidence of your run(s).

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.

Step 1:

Code Implementation Screenshots:

```
public int find(int p) {
    validate(p);
    int root = p;
    // FIXME

    while(root!=parent[root]){
        if(this.pathCompression){
            doPathCompression(root);
        }
        root = parent[root];
    }

    // END
    return root;
}
```

```
private void mergeComponents(int i, int j) {
    // FIXME make shorter root point to taller one

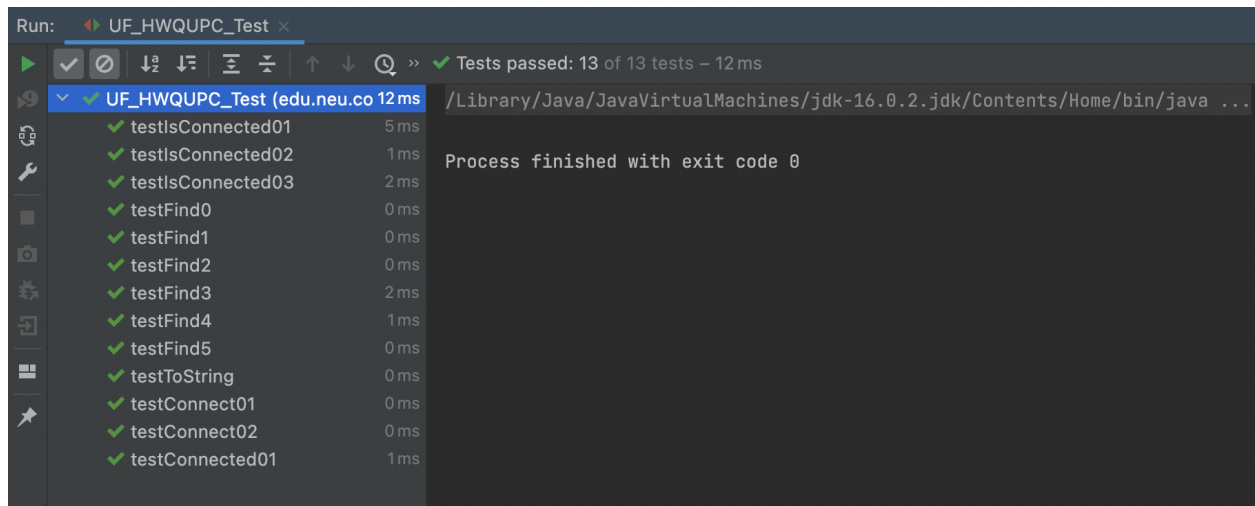
    if(i == j) return;

    if(height[i] < height[j]){
        updateParent(i,j);
        updateHeight(j,i);
    }else{
        updateParent(j,i);
        updateHeight(i,j);
    }

    // END
}

/**
 * This implements the single-pass path-halving mechanism of path compression
 */
private void doPathCompression(int i) {
    // FIXME update parent to value of grandparent
    updateParent(i, getParent(getParent(i)));
    // END
}
}
```

Testing Screenshots:



Step 2:

Code for UFClient.java

```
public class UFClient {  
  
    private static int count(int n){  
        int count = 0;  
        UF_HWQUPC uf = new UF_HWQUPC(n);  
        Random random = new Random();  
        while(uf.components()!=1){  
            int a = random.nextInt(n);  
            int b = random.nextInt(n);  
            uf.connect(a,b);  
            count++;  
        }  
        return count;  
    }  
  
    public static void main(String[] args) {  
        int times = 200;  
        for(int n = 1; n <= 100000; n*=2){  
            long sum = 0;  
            for(int i = 0; i < times; i++){  
                sum += count(n);  
            }  
            long connections = sum/times;  
            System.out.println("N Values: " + n + ", " + " Number of Connections Generated: " + connections);  
        }  
    }  
}
```

Output for UFClient.java

```
Run: UFClient x
/Library/Java/JavaVirtualMachines/jdk-16.0.2.jdk/Contents/Home/bin/java ...
N Values: 1, Number of Connections Generated: 0
N Values: 2, Number of Connections Generated: 1
N Values: 4, Number of Connections Generated: 5
N Values: 8, Number of Connections Generated: 11
N Values: 16, Number of Connections Generated: 29
N Values: 32, Number of Connections Generated: 68
N Values: 64, Number of Connections Generated: 155
N Values: 128, Number of Connections Generated: 349
N Values: 256, Number of Connections Generated: 775
N Values: 512, Number of Connections Generated: 1720
N Values: 1024, Number of Connections Generated: 3879
N Values: 2048, Number of Connections Generated: 8366
N Values: 4096, Number of Connections Generated: 18006
N Values: 8192, Number of Connections Generated: 38931
N Values: 16384, Number of Connections Generated: 85195
N Values: 32768, Number of Connections Generated: 178497
N Values: 65536, Number of Connections Generated: 385113

Process finished with exit code 0
```

Step 3:

