Program structures and algorithms
Spring 2023 (Section-01)

BY: PRANAV KAPOOR – NUID: 002998253

Assignment 4 (WQUPC)

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.

Solution:

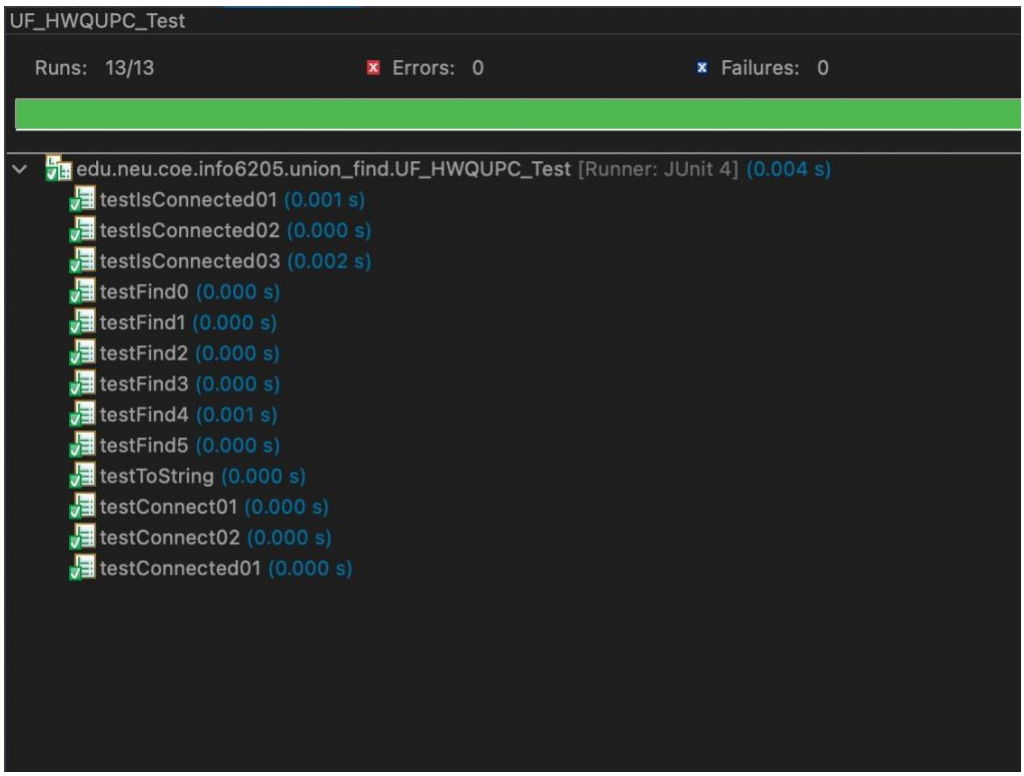
```
79      * @throws IllegalArgumentException unless {@code 0 <= p < n}
80      */
81      public int find(int p) {
82          validate(p);
83          int root = p;
84          // FIXME
85          // END
86
87          while(root!=parent[root]) {
88              root=parent[root];
89          }
90          if(pathCompression) {
91              while(p!=root) {
92                  int next=parent[p];
93                  parent[p]=root;
94                  p=next;
95              }
96          }
97          return root;
98      }
99
```

```
     */
    private int getParent(int i) {
        return parent[i];
    }

    private final int[] parent;   // parent[i] = parent of i
    private final int[] height;   // height[i] = height of subtree rooted at i
    private int count;  // number of components
    private boolean pathCompression;

    private void mergeComponents(int i, int j) {
        // FIXME make shorter root point to taller one
        // END
        if(height[i]<height[j]) parent[i]=j;
        else if(height[i]>height[j]) parent[j]=i;
        else {
            parent[j]=i;
            height[i]++;
        }
    }

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

        int node=find(i);
        while(i!=node) {
            int next=parent[i];
            parent[i]=node;
            i=next;
        }
    }
}
```

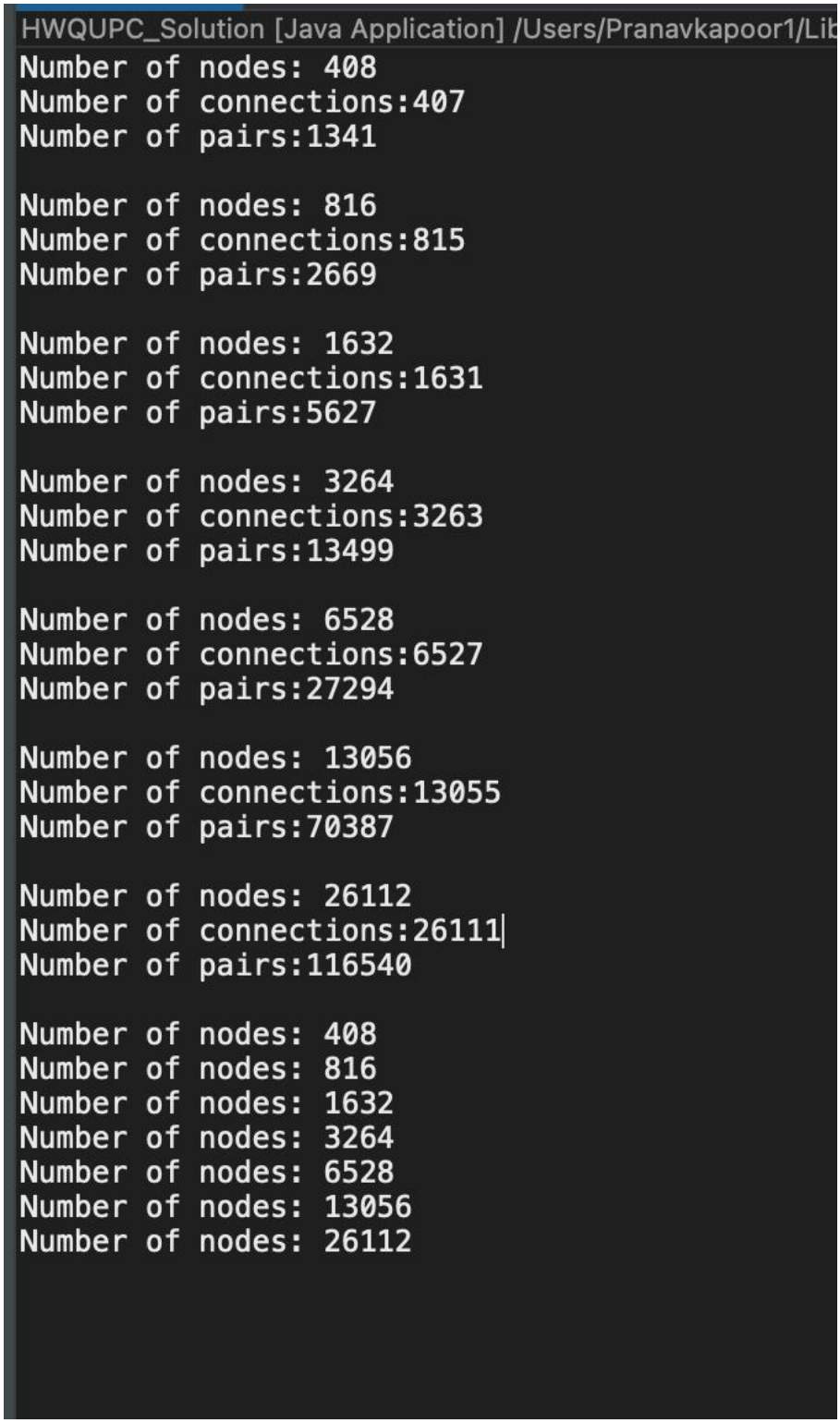
```
    private static int createHWQUPC(int n) {

        int no_connections = 0;
        int no_pairs = 0;

        UF_HWQUPC uf = new UF_HWQUPC(n);
        while (uf.components() != 1) {
            int x = (int) (Math.random() * (n));
            int y = (int) (Math.random() * (n));
            no_pairs += 1;
            if (uf.connected(x, y) == false) {
                uf.union(x, y);
                no_connections++;
            }
        }
        System.out.println("Number of connections:" + no_connections);
        System.out.println("Number of pairs:" + no_pairs + "\n");
        return no_pairs;
    }

```

```
public static void main (String arg[]) {
    int n = 408;

    XYSeries series = new XYSeries("Pairs Count vs Node Count");
    XYSeries series2 = new XYSeries("Pairs Count vs Node Count");
    double ratio = 0.0;
    for (int i = 0; i < 7; i++) {
        System.out.println("Number of nodes: " + n);
        int no_pairs = createHWQUPC(n);
        series.add(n, no_pairs);

        ratio = ratio + (no_pairs/(n * Math.log(n)));

        n = n * 2;
    }

    ratio = ratio / 7;
    n = 408;
    for (int i = 0; i < 7; i++) {
        System.out.println("Number of nodes: " + n);

        series2.add(n, ratio * (n * Math.log(n)));

        n = n * 2;
    }


    XYSeriesCollection dataset = new XYSeriesCollection();
    dataset.addSeries(series);
    dataset.addSeries(series2);

    JFreeChart chart = ChartFactory.createXYLineChart("Number of Nodes vs Pairs", "Number of Nodes",
            "Number of Pairs", dataset, PlotOrientation.VERTICAL, true, true, false);

    JFrame frame = new JFrame("Pranav Kapoor - 002998253");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.add(new ChartPanel(chart));
    frame.setSize(1000, 500);
    frame.setVisible(true);
}
```

Unit Tests



N=408
M=408


HWQUPC_Solution [Java Application] /Users/Pranavkapoor1/Lib
Number of nodes: 408
Number of connections:407
Number of pairs:1341
Number of nodes: 816
Number of connections:815
Number of pairs:2669
Number of nodes: 1632
Number of connections:1631
Number of pairs:5627
Number of nodes: 3264
Number of connections:3263
Number of pairs:13499
Number of nodes: 6528
Number of connections:6527
Number of pairs:27294
Number of nodes: 13056
Number of connections:13055
Number of pairs:70387
Number of nodes: 26112
Number of connections:26111
Number of pairs:116540
Number of nodes: 408
Number of nodes: 816
Number of nodes: 1632
Number of nodes: 3264
Number of nodes: 6528
Number of nodes: 13056
Number of nodes: 26112

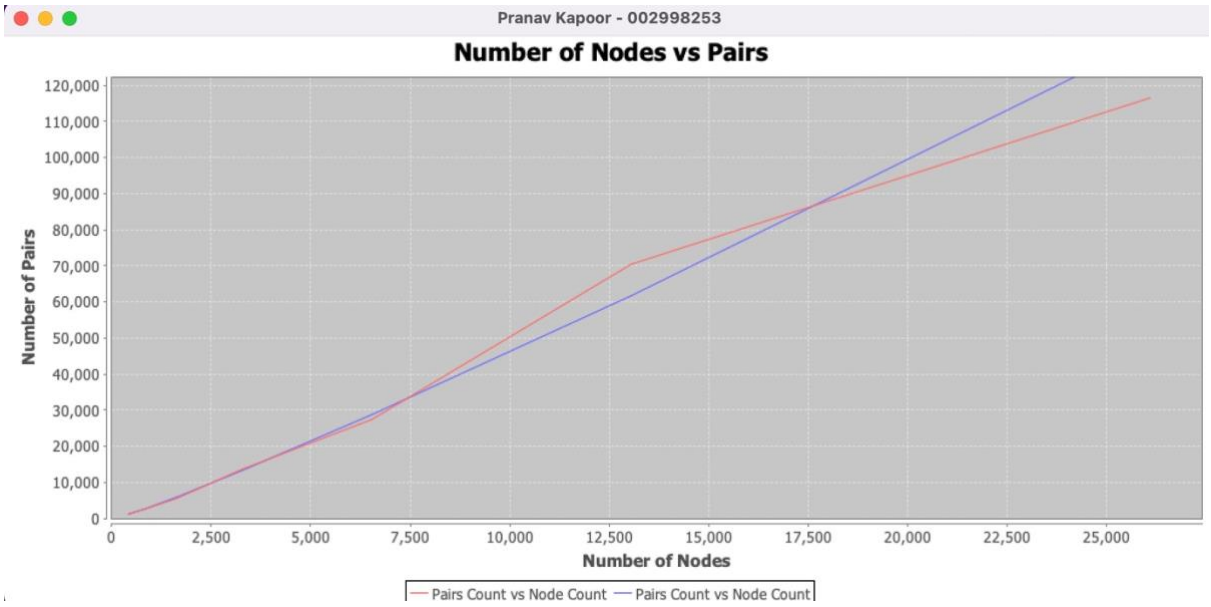
N=208
M=308

```
HWQUPC_Solution [Java Application] /Users/Pranavkapoor1/Libra
Number of nodes: 208
Number of connections:207
Number of pairs:643

Number of nodes: 416
Number of connections:415
Number of pairs:1242

Number of nodes: 832
Number of connections:831
Number of pairs:3064

Number of nodes: 1664
Number of connections:1663
Number of pairs:7475

Number of nodes: 3328
Number of connections:3327
Number of pairs:15623

Number of nodes: 6656
Number of connections:6655
Number of pairs:27335

Number of nodes: 13312
Number of connections:13311
Number of pairs:67336

Number of nodes: 308
Number of nodes: 616
Number of nodes: 1232
Number of nodes: 2464
Number of nodes: 4928
Number of nodes: 9856
Number of nodes: 19712
```

GRAPH:

We can see that the number of nodes increase in the slope of nlogn with number of pairs.

N=408
M=408



N=208
M=308