Assignment 4 - Program Structures and Algorithms Spring 2023(SEC - 1)

NAME: Paurush Batish **NUID: 002755631**

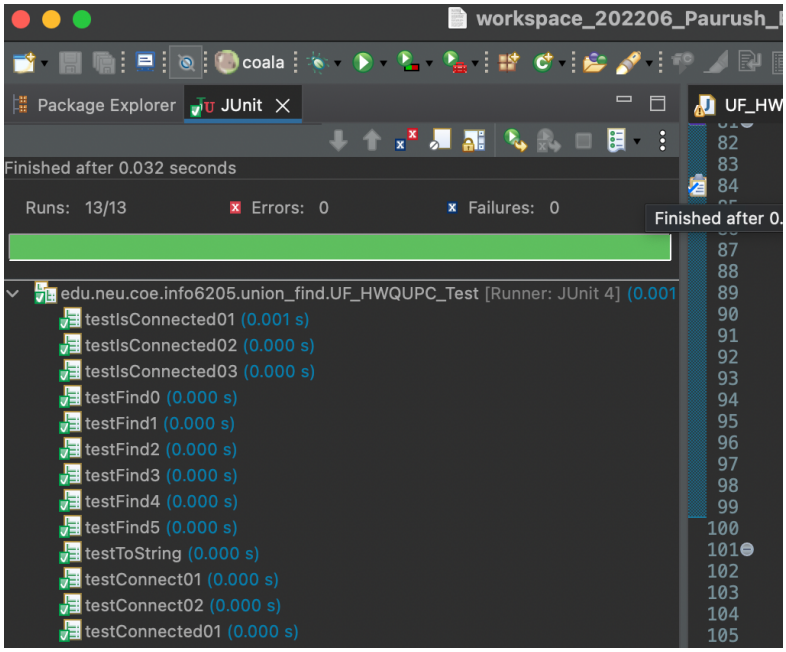
Your task is

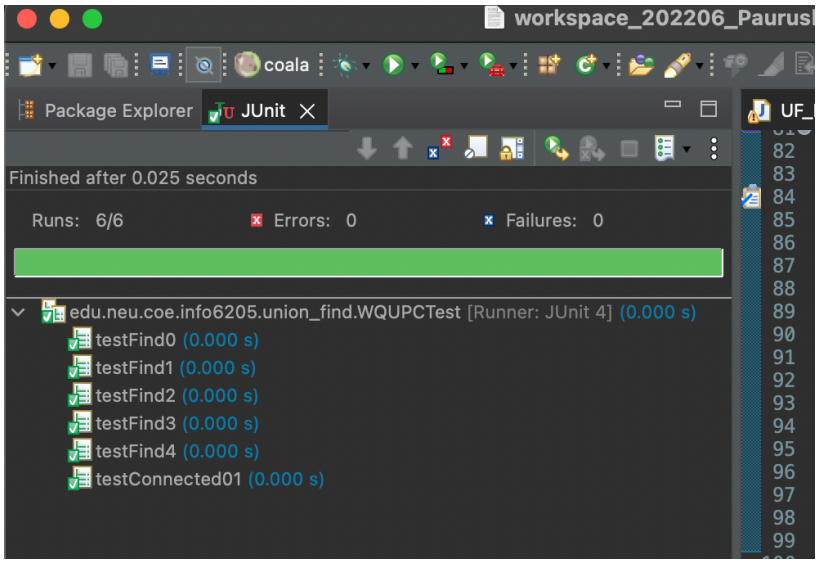
Step 1:
(a) Implement height-weighted Quick Union with Path Compression. For this, you will flesh out the class UF_HWQUPC. All you have to do is to fill in the sections marked with // TO BE IMPLEMENTED ... // ...END IMPLEMENTATION.

```
public int find(int p) {
    validate(p);
    int root = p;
    while (root != parent[root]) {
        root = parent[root];
    }
    if (pathCompression) {
        doPathCompression(p, root);
    }
    return root;
}
```

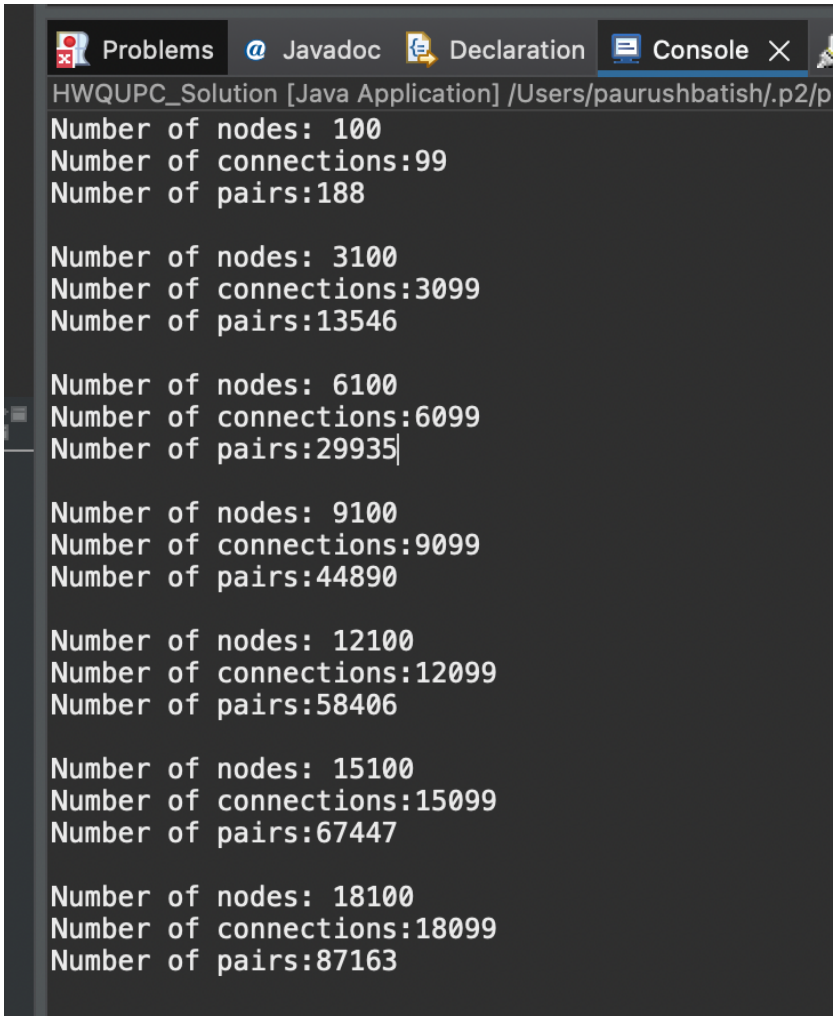
```
private void mergeComponents(int i, int j) {
    if (height[i] < height[j]) parent[i] = j;
    else if (height[i] > height[j]) parent[j] = i;
    else {
        parent[j] = i;
        height[i]++;
    }
}

/**
 * This implements the single-pass path-halving mechanism of path compression
 */
private void doPathCompression(int i, int root) {
    while (i != root) {
        int next = parent[i];
        parent[i] = root;
        i = next;
    }

}
```

(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.

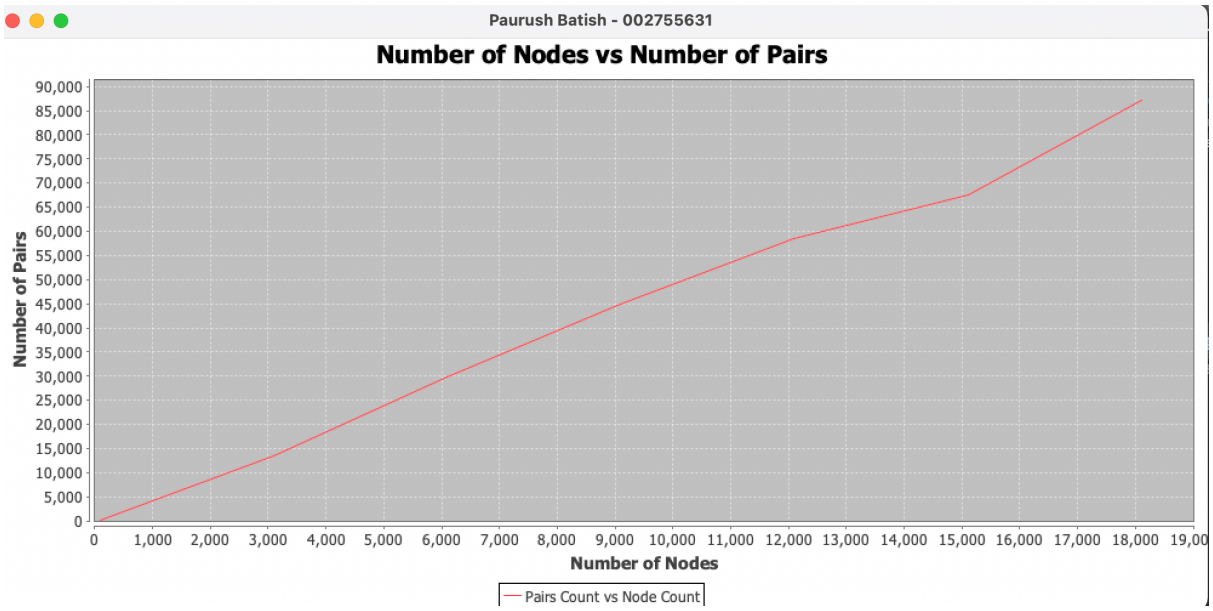
NOTE: although I'm not going to tell you in advance what the relationship is, I can assure you that it is a *simple* relationship.

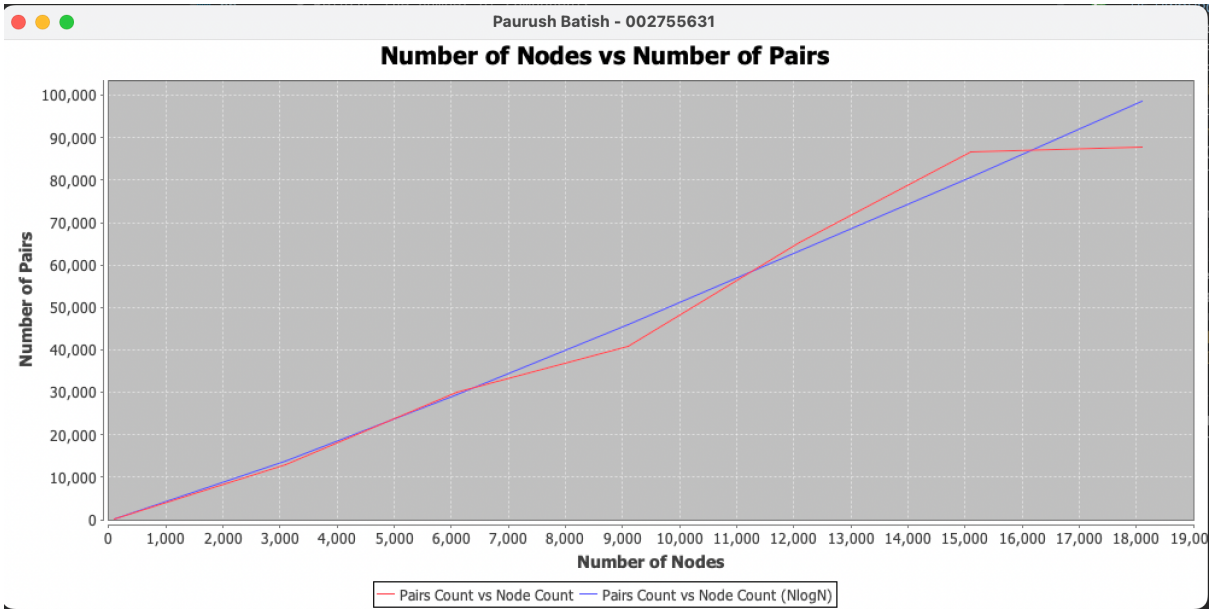
Don't forget to follow the submission guidelines. And to use sufficient (and sufficiently large) different values of n.

Conclusion –

The number of objects (n) is increased by 3000 in each iteration of the loop, and for each value of n, the createHWQUPC method is called until the number of components is reduced to 1. The number of pairs (m) generated to reduce the number of components from n to 1 is measured and plotted against the number of objects (n). The number of pairs required to reduce the number of components to 1 using the Quick-Union algorithm without path compression is proportional to n^2. With path compression, the complexity is reduced to n*log(n).

Based on the mathematical relations and the graph values the relationship is appearing to be having a complexity of $n\log(\mathrm{n})$.





Hence we can say that since the graphs coincide for this relation. Hence the equation can be said as

$$Y = m * (Nlog(N))$$

Therefore, we can conclude that the given program is using the Union-Find data structure with path compression, as the graph generated shows an approximately logarithmic relationship between the number of pairs and the number of objects.