Program Structures and Algorithms
Spring 2023 (SEC –3)
Assignment-4: Union Find.

NAME: Venkatesha Matam
NUID: 002740702

Task:

- Implement the missing portions of UF_HWQUPC.java.
- Run the test cases in UF_HWQUPC_Test.java.
- Create a new class and main method to determine the number of sites.
- Create a count method inside the new class which takes in value n and prints the number of pairs produced.
- Draw conclusions from observations and evidence.

Code Change Snapshots:

1) **find() method in UF_HWQUPC.java –**

```
80      */
81      public int find(int p) {
82          validate(p);
83          int root = p;
84          // FIXME
85          if(!this.pathCompression) {
86              while(root != parent[root]) {
87                  root = parent[root];
88              }
89          } else {
90              doPathCompression(root);
91              root = parent[root];
92          }
93          // END
94          return root;
95      }
96
```

2) mergeComponents() :

```
179     private void mergeComponents(int i, int j) {
180         // FIXME make shorter root point to taller one
181         if(i == j) {
182             return;
183         } else if (height[i] == height[j]) {
184             parent[j] =i;
185             height[i] = height[i]+1;
186         } else if(height[i]<height[j]) {
187             parent[i] = j;
188         } else {
189             parent[j] =i;
190         }
191         // END
192     }
```

3) doPathCompression():

```
     * This implements the single-pass path-halving mechani
     */
    private void doPathCompression(int i) {
        // FIXME update parent to value of grandparent
        while(i!=parent[i]) {
            parent[i] = parent[parent[i]];
            i=parent[i];
        }
        // END
    }
}
```

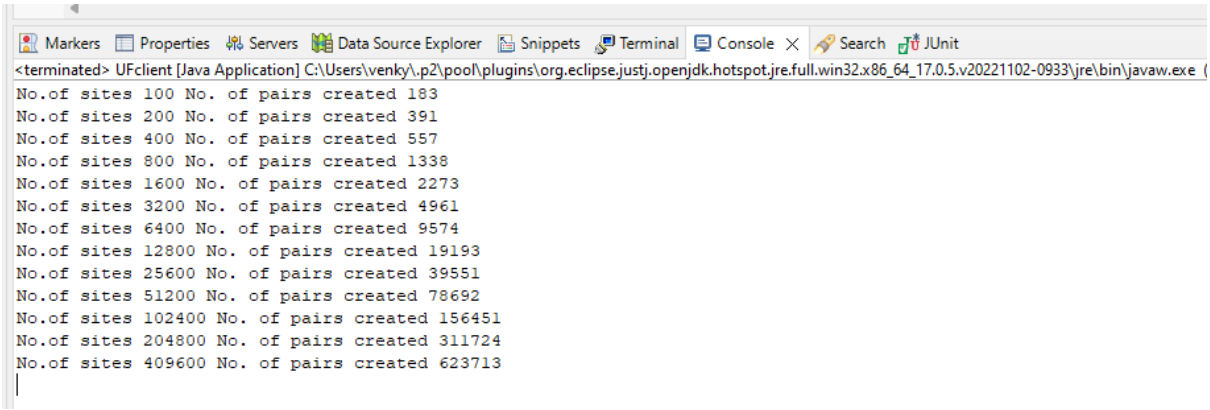
4) main() method of UFclient.java

```
    public static void main(String args[]) {
//      Scanner scanner = new Scanner(System.in);
//      System.out.println("Enter number of sites");
//      int n = scanner.nextInt();
        int n = 100;
        while (n < 409600) {
            System.out.println("No.of sites " + n + " No. of pairs created " + count(n));
            n = n * 2;
        }
    }
```

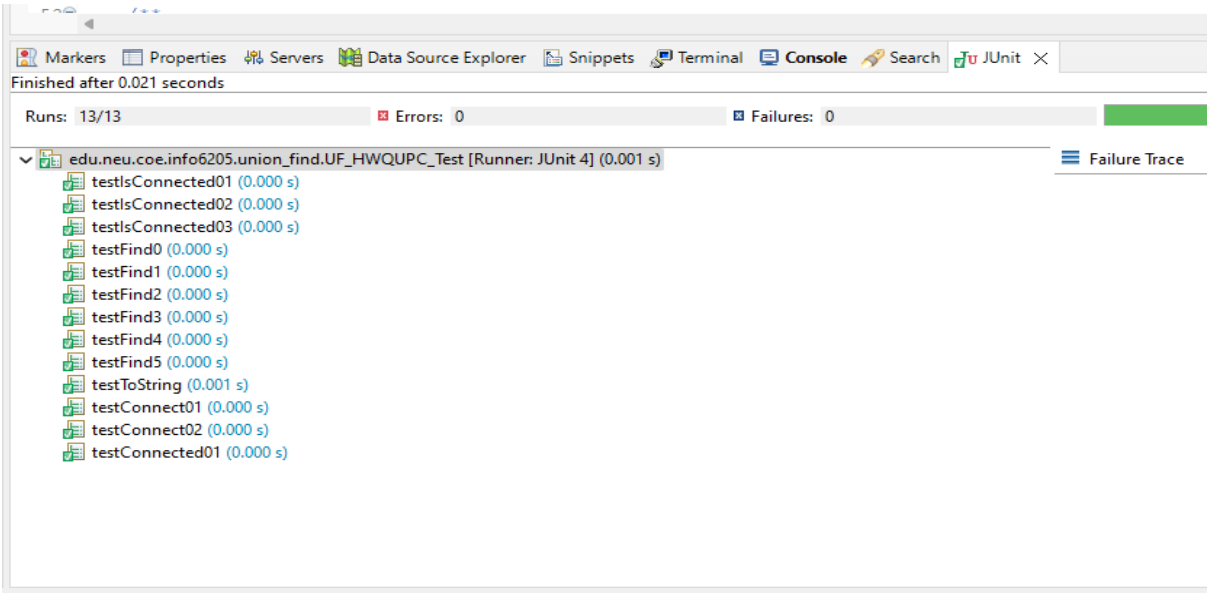
5) **count() method of UFClient.java:**

```
public static int count(int n) {
        UF_HWQUPC uf = new UF_HWQUPC(n);
        Random rand = new Random();
        int count = 0;
        while (uf.components() != 1) {
            int i = rand.nextInt(n);
            int j = rand.nextInt(n);
            count++;
            if (!uf.isConnected(i, j)) {
                uf.union(i, j);
            }
        }
        return count;
    }
```

Output Snapshots:

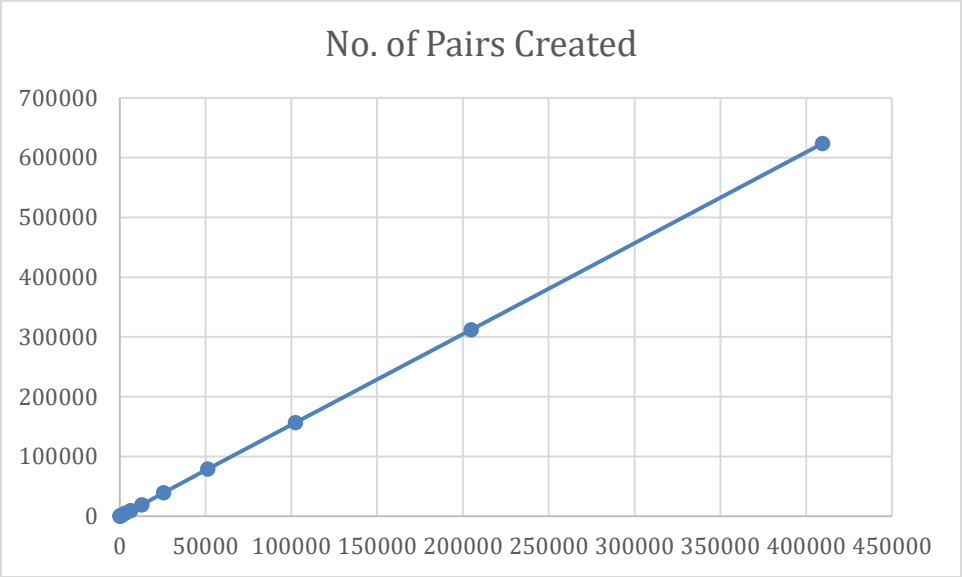


Unit Test Screenshots:



Observations and Analysis:

No. of Sites (N)	No. of Pairs Created (M)	Slope
100	183	2.08
200	391	0.83
400	557	1.9525
800	1338	1.16875
1600	2273	1.68
3200	4961	1.4415625
6400	9574	1.50296875
12800	19193	1.59046875
25600	39551	1.528945313
51200	78692	1.518730469
102400	156451	1.516337891
204800	311724	1.523383789
409600	623713	1.522736816



Conclusion:

As the number of sites increases, the number of pairs will increase proportionally, meaning that the relationship between the two is linear.

Number of Pairs is directly proportional to the Number of sites entered.

Upon analysing the values and the slope, I conclude that -

Number of pairs created = C * number of sites where C is a constant value (~1.5) and the slope of the line --- **M = C*N**.