Program Structures & Algorithms
Spring 2022
Assignment No. 3 (WQUPC)

Name: Sri Vyshnavi Kotha
(NUID): 002985810

Task

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).

- Implemented height-weighted Quick Union with Path Compression.
- Attached the screenshot of all testcases working.
- Created a TestClient file which 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.
- Attached the evidence of the runs on the TestClient File
- Determine the relationship between the number of objects (*n*) and the number of pairs (*m*) and justified with a graph

Output screenshot

```
n:  UF_HWQUPC_TestCustom
/Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home/bin/java ...
Provide an Integer value to determine the number of sites :
1001
Number of Sites : 891 Number of Connections : 3111
Number of Sites : 516 Number of Connections : 1942
Number of Sites : 322 Number of Connections : 837
Number of Sites : 475 Number of Connections : 1527
Number of Sites : 470 Number of Connections : 1887
Number of Sites : 77 Number of Connections : 145
Number of Sites : 636 Number of Connections : 2367
Number of Sites : 280 Number of Connections : 774
Number of Sites : 717 Number of Connections : 2723
Number of Sites : 881 Number of Connections : 3824
Number of Sites : 487 Number of Connections : 1393
Number of Sites : 46 Number of Connections : 94
Number of Sites : 155 Number of Connections : 433
Number of Sites : 830 Number of Connections : 2541
Number of Sites : 368 Number of Connections : 1313
Number of Sites : 439 Number of Connections : 1603
Number of Sites : 149 Number of Connections : 341
Number of Sites : 126 Number of Connections : 291
Number of Sites : 483 Number of Connections : 1748
Number of Sites : 320 Number of Connections : 1092
Number of Sites : 793 Number of Connections : 2855
Number of Sites : 793 Number of Connections : 3317
Number of Sites : 826 Number of Connections : 3064
Number of Sites : 336 Number of Connections : 1022
Number of Sites : 13 Number of Connections : 18
Git  Run  TODO  Problems  Terminal  Build  Dependencies
```

Relationship Conclusion

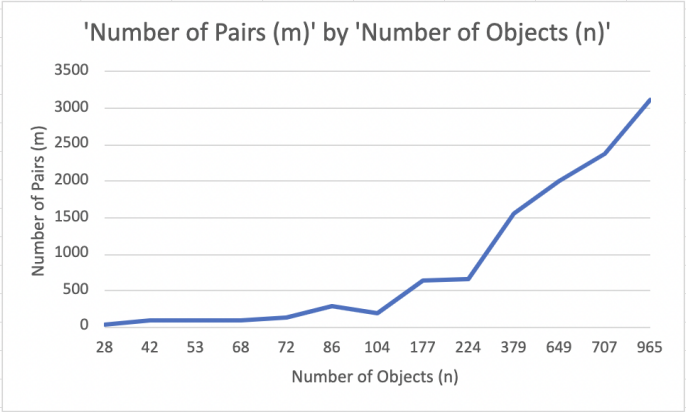
If ‘n’ is the number of objects and ‘m’ is the number of pairs generated to reduce the number of components from n to 1 then **m = nlogn/2** Is the relationship deduced.

- The value of ‘m’ is approximately equal to the value of n*Log(n)/2 and the relationship mentioned above holds good with approximation.
- The conclusion has been made based on the series of tests.

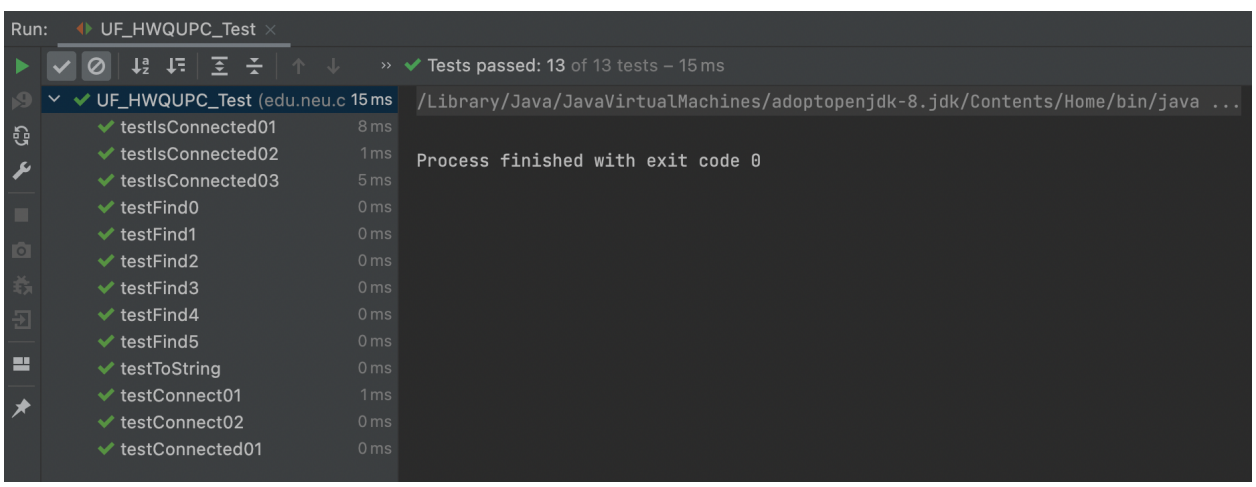
Evidence / Graph

Below graph depicts the relation between the number of Objects/Sites (n) and Number of Pairs/Connections (m)

Number of Objects (n)	Number of Pairs (m)	nlogn/2
28	43	67.30296891
42	90	113.2386659
53	98	151.789892
68	95	206.9737366
72	137	222.1173001
86	295	276.3293845
104	190	348.4228653
177	648	660.8830912
224	668	874.4237513
379	1551	1623.26724
649	2007	3031.50323
707	2374	3346.077724
965	3111	4783.690826



Unit tests result



Code

UF_HWQUPC.java

```
public int find(int p) {
    validate(p);
    int root = p;
    while (root != getParent(root)) {
        if (this.pathCompression){
            doPathCompression(root);
        }
        root = getParent(root);
    }
    return root;
}

private void mergeComponents(int i, int j) {
    int rootX = find(i);
    int rootY = find(j);
    if (rootX != rootY) {
        if (height[rootX] > height[rootY]) {
            updateParent(rootY, rootX);
        } else if (height[rootX] < height[rootY]) {
            updateParent(rootX, rootY);

        } else {
            updateParent(rootY, rootX);
            updateHeight(rootX,rootY);
        }
    }
}

private void doPathCompression(int i) {
    parent[i] =  getParent(getParent(i));
}
```

UF_HWQUPC_TestClient.java

```
public class UF_HWQUPC_TestClient {
    public static int Count(UF_HWQUPC uf, int numberOfSites) {
        int numberOfPairs = 0;
        Random random = new Random();
        while (uf.components() != 1) {
            uf.connect(random.ints(0, numberOfSites).findFirst().getAsInt(),
random.ints(0, numberOfSites).findFirst().getAsInt());
            numberOfPairs++;
        }
        return numberOfPairs;
    }

    public static void main(String[] args){
        System.out.println("Provide an Integer value to determine the number of
sites :");
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        Random random = new Random();
        for (int i = 1; i < 100; i++) {
            int numberOfSites = random.ints(0, n-1).findFirst().getAsInt();
            UF_HWQUPC uf = new UF_HWQUPC(numberOfSites);
            int numberOfConnections = Count(uf, numberOfSites);
            System.out.println("Number of Sites : " + numberOfSites + " Number
of Connections : "+ numberOfConnections);
        }
    }
}
```