Ebenezer Ajay Williams Vincent Sankey Raj (002166250)

# Program Structures & Algorithms

## Fall 2021

## Assignment No. 3(WQUPC)

- **Task (List of tasks performed in the Assignment)**
    1) **Code added to the UF_HWQUPC.java class .**
    2) **New class called UnionFindAssignment3.java is created.**
    3) **Experiment is performed with nodes (n) starting from 1000 up to 64000 .**
    4) **Graph is plotted to deduce that the relation is linear.**
    5) **Unit tests have been run and are successful.**

- **Relationship Conclusion:**

    **The relationship between the nodes (n) and pairs (M) increases linearly.**

$$M \propto n$$

$$M = K * n$$

**Where ,**

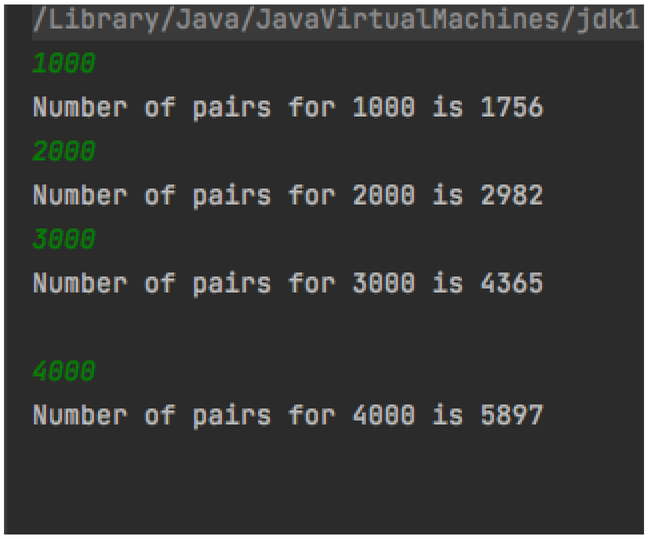
**n is the number of nodes**

**M is the number of pairs**

**K is a constant**

- **Evidence to support the conclusion:**

1. **Output (Snapshot of Code output in the terminal)**



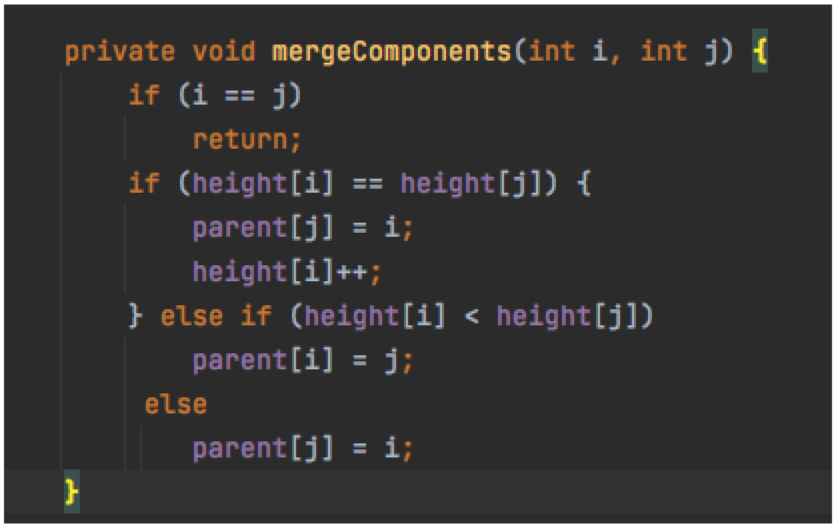
```
/Library/Java/JavaVirtualMachines/jdk1
1000
Number of pairs for 1000 is 1756
2000
Number of pairs for 2000 is 2982
3000
Number of pairs for 3000 is 4365

4000
Number of pairs for 4000 is 5897
```

Code modified to get the required output on the terminal

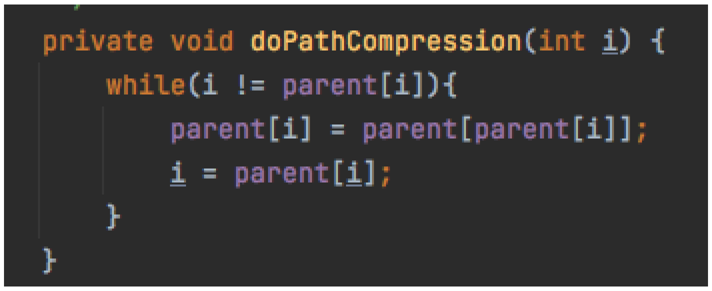
- UnionFindAssignment3.java has the main method used to call perform the experiment. (code at the end of this document)

- mergeComponents() method



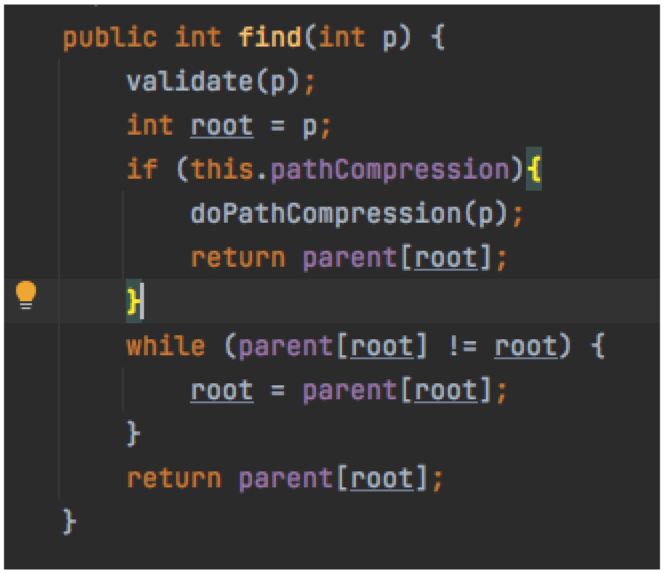
```java
private void mergeComponents(int i, int j) {
    if (i == j)
        return;
    if (height[i] == height[j]) {
        parent[j] = i;
        height[i]++;
    } else if (height[i] < height[j])
        parent[i] = j;
     else
        parent[j] = i;
}
```

- doPathCompression() method



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

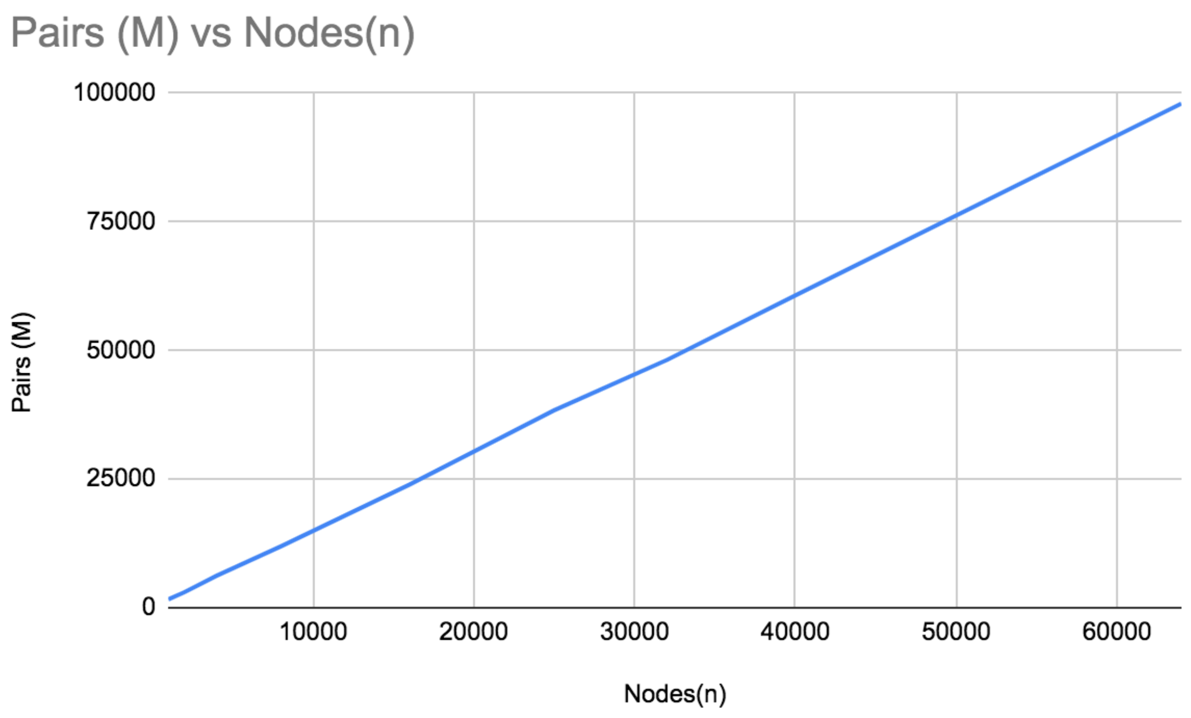
- **find() method in InsertionSort.java**



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

2. **Graphical Representation**

**Line graph showing the relationship between the number of nodes (n) vs the connections made is plotted.**



We can see that the line is linear, hence as the value of n increases the number of connections also increases linearly

**Tabulated values:**

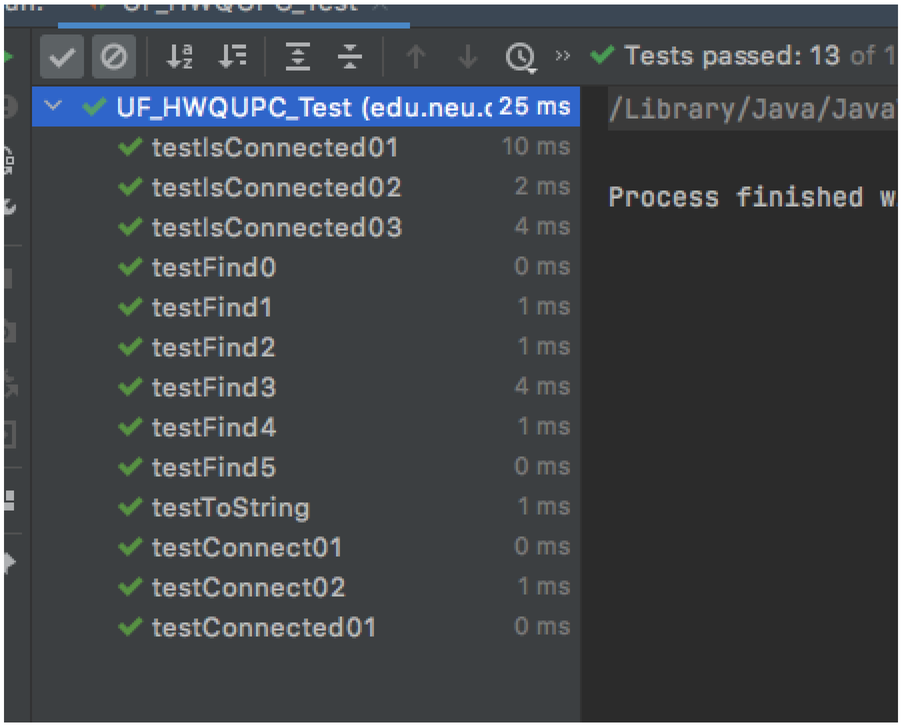
| **Nodes(n)** | **Pairs (M)** |
|---:|---:|
| 1000 | 1613 |
| 2000 | 3010 |
| 4000 | 6215 |
| 8000 | 11914 |
| 16000 | 23944 |
| 25000 | 38390 |
| 32000 | 48150 |
| 40000 | 60705 |
| 64000 | 97994 |

Analysis & Proof of Relation

Since the number of nodes increases the number of connections that can be made also increases since the maximum number of edges is given by n *(n-1) /2.

Since n-1 unions are made and is in random, the number of pairs also increases in linear

- **Unit tests result:(Snapshot of successful unit test run)**



```
Tests passed: 13 of 1
UF_HWQUPC_Test (edu.neu.c 25 ms
  testIsConnected01   10 ms
  testIsConnected02    2 ms
  testIsConnected03    4 ms
  testFind0            0 ms
  testFind1            1 ms
  testFind2            1 ms
  testFind3            4 ms
  testFind4            1 ms
  testFind5            0 ms
  testToString         1 ms
  testConnect01        0 ms
  testConnect02        1 ms
  testConnected01      0 ms

/Library/Java/Java

Process finished w
```

**Final code of UnionFindAssignment3.java**

```java
package edu.neu.coe.info6205.assignment;

import edu.neu.coe.info6205.union_find.UF_HWQUPC;

import java.util.Random;
import java.util.Scanner;

public class UnionFindAssignment3 {

   UF_HWQUPC uf ;
   Random random;

   public int count(int n){
       this.uf = new UF_HWQUPC(n);
       this.random = new Random();
       int result =0;

       while(uf.components()!=1)
       {
           int node1= random.nextInt(n);
           int node2= random.nextInt(n);
           result++;
           if(!uf.connected(node1,node2))
               uf.union(node1,node2);
```

```
        }
        return  result;
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        UnionFindAssignment3 ufa = new UnionFindAssignment3();
        int input = 0;
        while(true){
            input = scn.nextInt();
            if(input ==-1)
                break;
            System.out.println("Number of pairs for " + input + " is " +
ufa.count(input));
        }
    }
}
```