Student Name: **Hongyao Tao (001067209)**

INFO 6205
Program Structures & Algorithms
Spring 2021

Assignment No.3

- **Implement height-weighted Quick Union with Path Compression**

```java
    */
    public int find(int p) {
        validate(p);
//        int currentIndex=p;
//        int root=parent[p];
//        while (root!=currentIndex){
//            currentIndex=root;
//            root=parent[currentIndex];
//        }
//        return root;
        int currentNode=p;
        int parent=getParent(currentNode);
        while (parent!=currentNode) {
            if(pathCompression){
                doPathCompression(currentNode);
            }
            currentNode=getParent(currentNode);
            parent=getParent(currentNode);
        }
        return parent;
    }
```

```java
private void mergeComponents(int i, int j) {
    int iroot=find(i);
    int jroot=find(j);
    if(iroot==jroot) return;
    int iRootHeight=height[iroot];
    int jRootHeight=height[jroot];
    if(iRootHeight>=jRootHeight){
        updateParent(jroot,iroot);
        updateHeight(iroot,jroot);
    }else {
        updateParent(iroot,jroot);
        updateHeight(jroot,iroot);
    }


}

/**
 * This implements the single-pass path-halving mechanism of path compression
 */
private void doPathCompression(int i) {
    // TO BE IMPLEMENTED update parent to value of grandparent
    int grandparent=getParent(getParent(i));
    updateParent(i,grandparent);
}
```

Figure 1. The implement of UF_HWQUPC class

- **The implement of main and count function**

```java
public static int count(int n){
    UF_HWQUPC uf_hwqupc=new UF_HWQUPC(n);
    int connectCount=0;
    Random random=new Random();
    while (uf_hwqupc.components()!=1){
        int p=random.nextInt(n);
        int q=random.nextInt(n);
        uf_hwqupc.connect(p,q);
        connectCount++;
    }
    return connectCount;
}
```

```java
public static void main(String[] args) {
    String fileName="CSV/Assignment3/n_with_connection.csv";
    writeToCSV(fileName, line: "N,Connection");
    int n=10;
    int times=20;
    int size=100;
    for(int i=0;i<times;i++){
        long sum =0;
        for(int j=0; j<size; j++){
            sum +=count(n);
        }
        long average=sum/size;
        //System.out.println(sum+ " "+ n);
        System.out.println("Average times of connection are "+average+", n is "+n);
        writeToCSV(fileName, line: n+","+average);
        n*=2;
    }
}
```

Figure 2. The implement of count and main functions
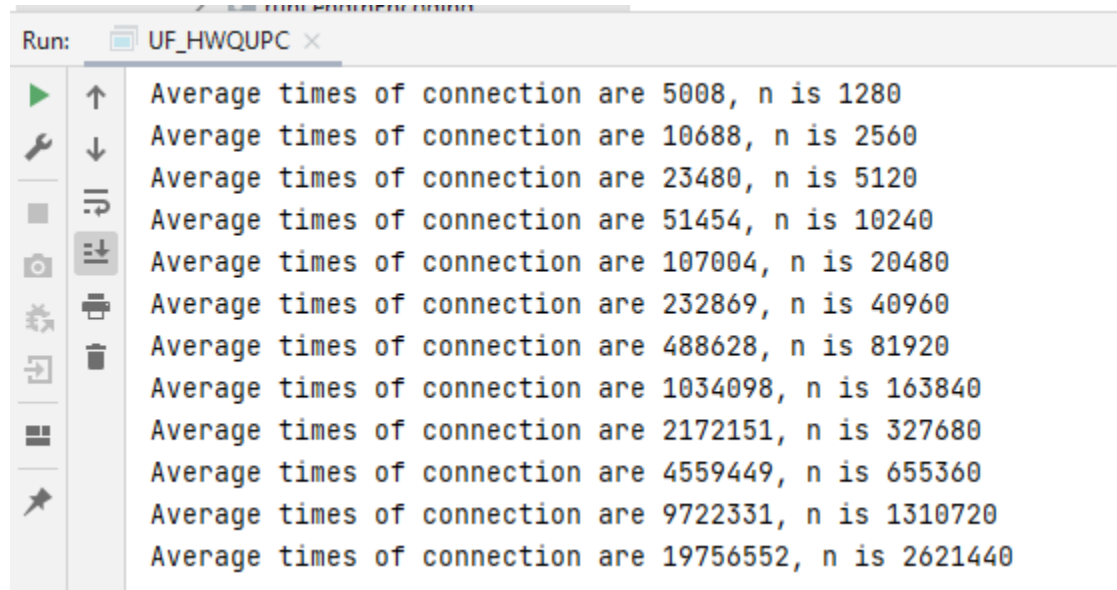
- **The Output**

Figure 3. The output result of **main function**

- **Relationship Conclusion:**

$$Connection = N^{1.135}$$

- **Evidence to support the conclusion:**

N	Connection
10	17
20	37
40	85
80	210
160	451
320	1042
640	2237
1280	5008
2560	10688
5120	23480
10240	51454
20480	107004
40960	232869
81920	488628
163840	1034098
327680	2172151
655360	4559449
1310720	9722331
2621440	19756552
5242880	42804257

Figure 4. The data of **examination**

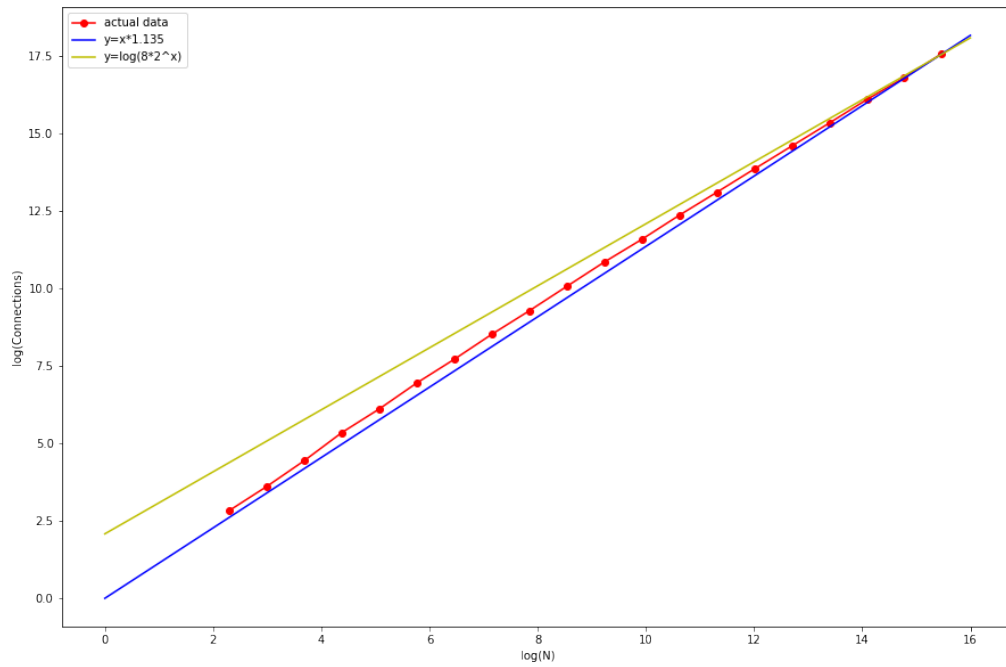- **Graphical representation:**

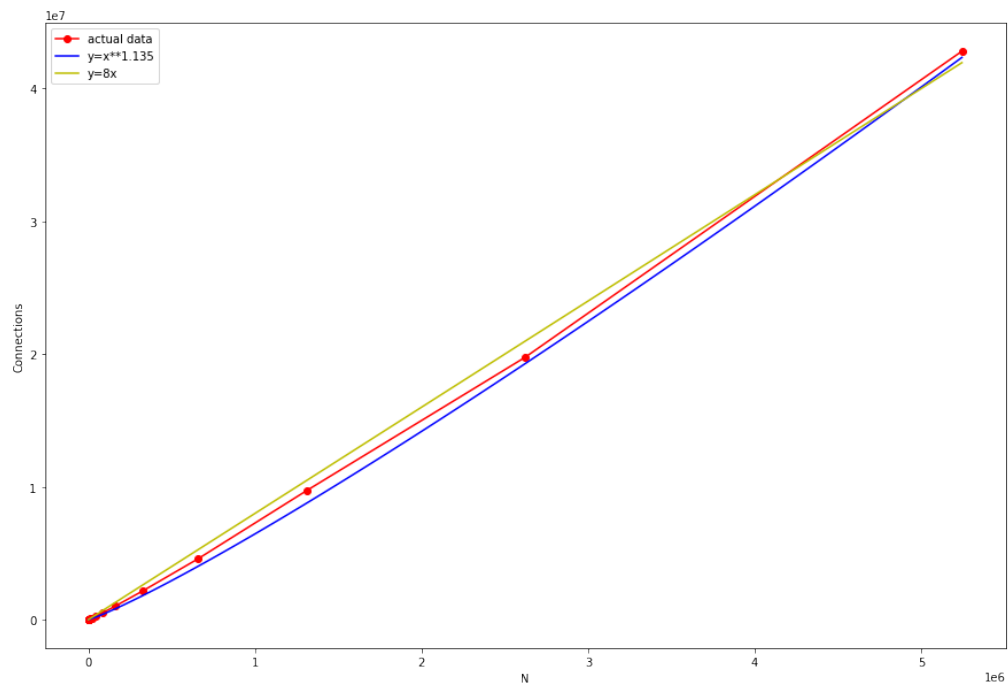**Figure 5. The Connection/N grow logarithm graph**
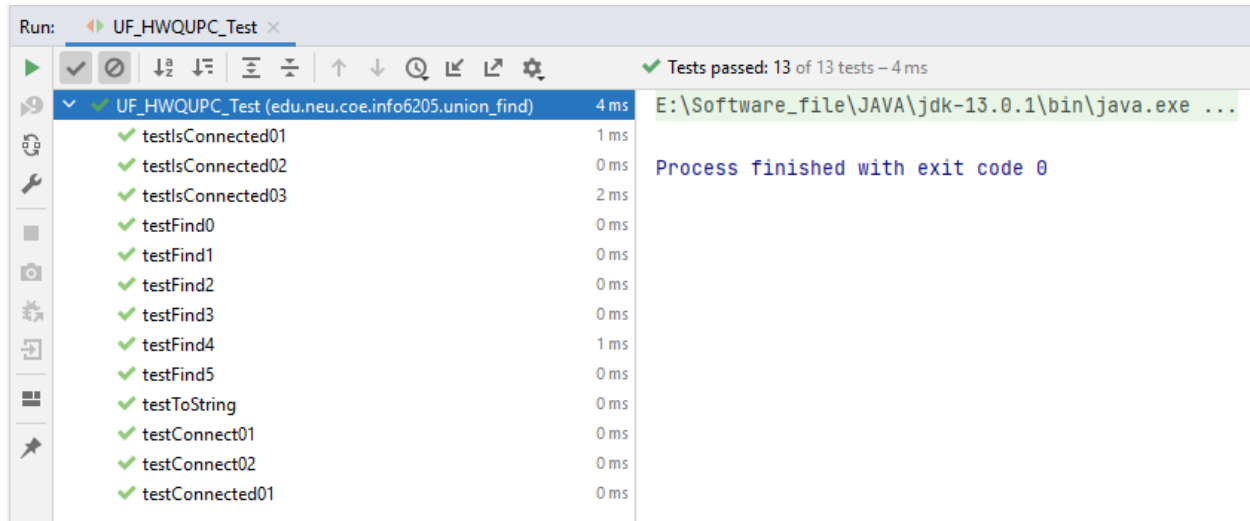

Figure 6. The Connection/N line graph

- **Unit tests result:**

Figure 7. The unit tests of UF_HWQUPC class