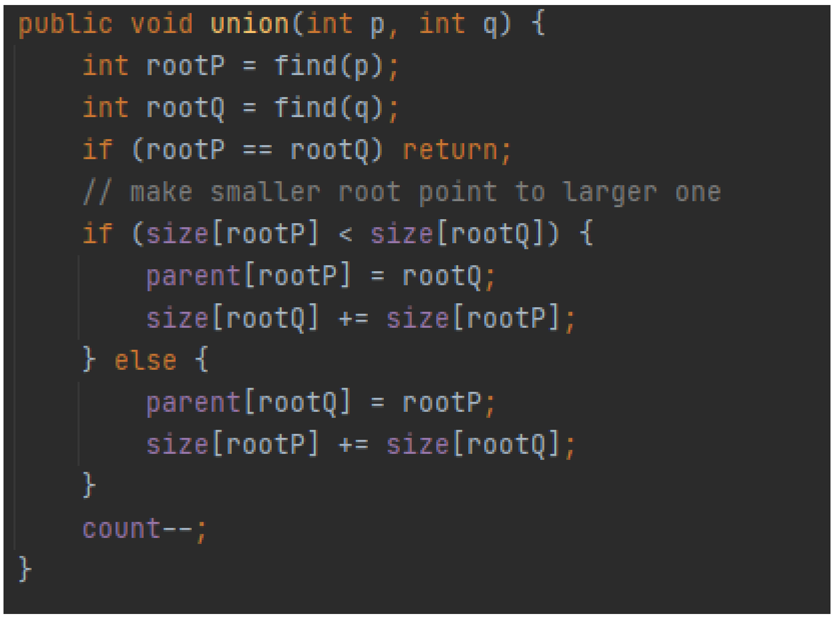
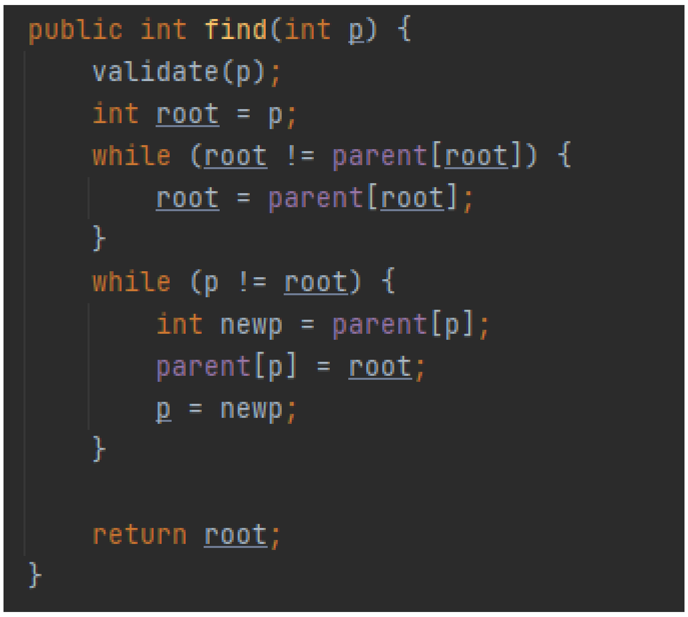
Student Name: **Jiachen Luo (001061582)**

INFO 6205
Program Structure & Algorithms
Spring 2021
Assignment4

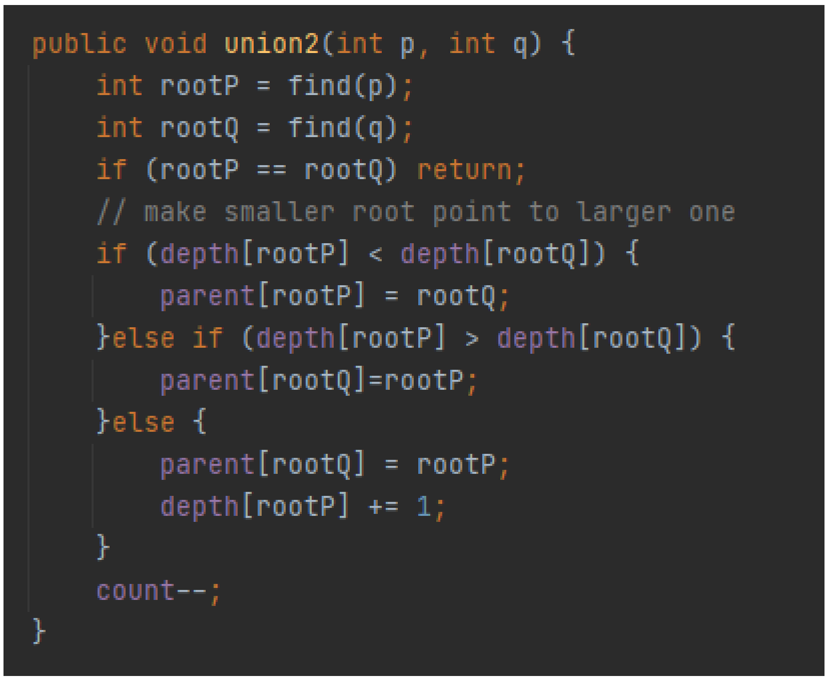
Task: For weighted quick union, store the depth rather than the size. For weight quick union with path compression, do two loops.



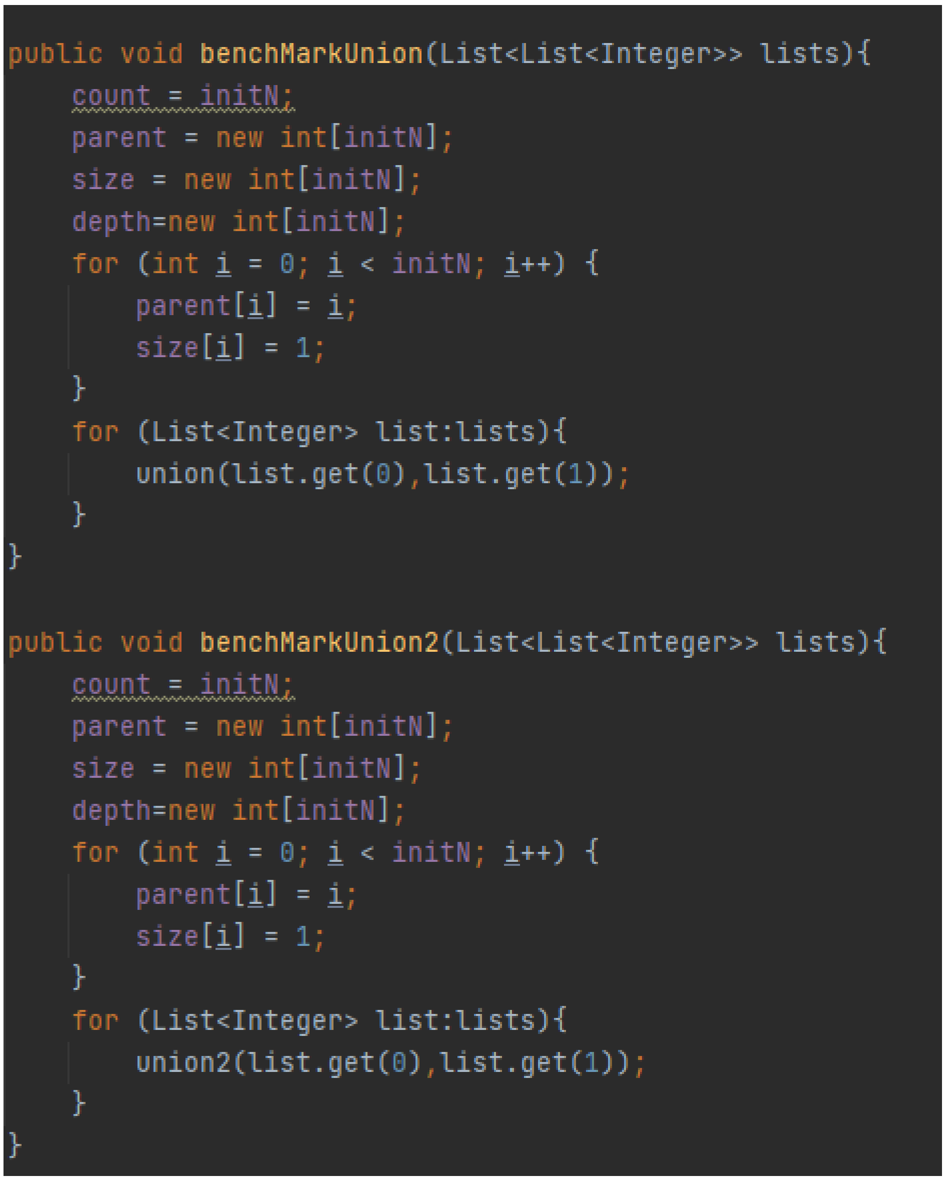
```java
public void union(int p, int q) {
    int rootP = find(p);
    int rootQ = find(q);
    if (rootP == rootQ) return;
    // make smaller root point to larger one
    if (size[rootP] < size[rootQ]) {
        parent[rootP] = rootQ;
        size[rootQ] += size[rootP];
    } else {
        parent[rootQ] = rootP;
        size[rootP] += size[rootQ];
    }
    count--;
}
```

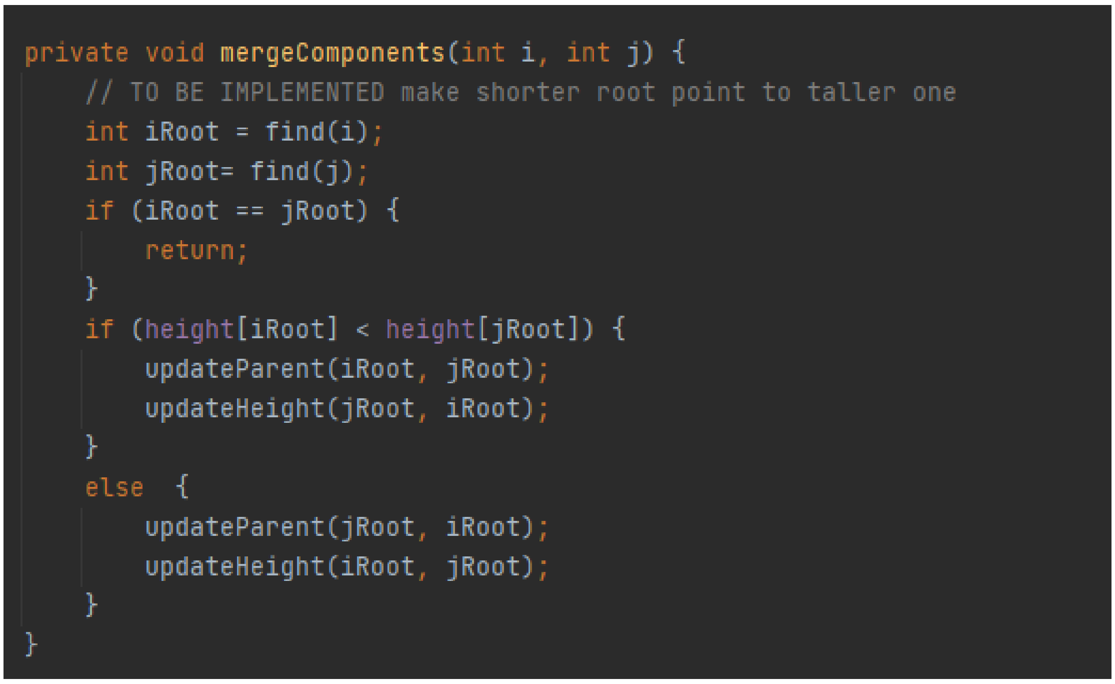


```java
public int find(int p) {
    validate(p);
    int root = p;
    while (root != parent[root]) {
        root = parent[root];
    }
    while (p != root) {
        int newp = parent[p];
        parent[p] = root;
        p = newp;
    }

    return root;
}
```

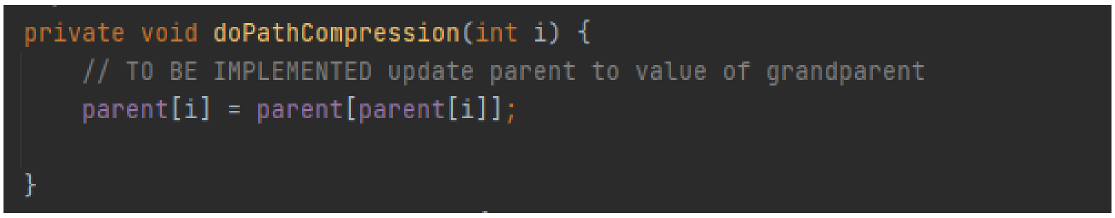


```java
public void union2(int p, int q) {
    int rootP = find(p);
    int rootQ = find(q);
    if (rootP == rootQ) return;
    // make smaller root point to larger one
    if (depth[rootP] < depth[rootQ]) {
        parent[rootP] = rootQ;
    }else if (depth[rootP] > depth[rootQ]) {
        parent[rootQ]=rootP;
    }else {
        parent[rootQ] = rootP;
        depth[rootP] += 1;
    }
    count--;
}
```


```java
public void benchMarkUnion(List<List<Integer>> lists){
    count = initN;
    parent = new int[initN];
    size = new int[initN];
    depth=new int[initN];
    for (int i = 0; i < initN; i++) {
        parent[i] = i;
        size[i] = 1;
    }
    for (List<Integer> list:lists){
        union(list.get(0),list.get(1));
    }
}

public void benchMarkUnion2(List<List<Integer>> lists){
    count = initN;
    parent = new int[initN];
    size = new int[initN];
    depth=new int[initN];
    for (int i = 0; i < initN; i++) {
        parent[i] = i;
        size[i] = 1;
    }
    for (List<Integer> list:lists){
        union2(list.get(0),list.get(1));
    }
}
```



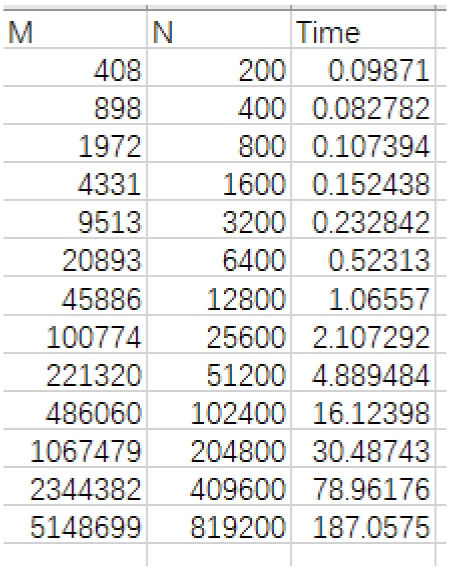
```java
private void mergeComponents(int i, int j) {
    // TO BE IMPLEMENTED make shorter root point to taller one
    int iRoot = find(i);
    int jRoot= find(j);
    if (iRoot == jRoot) {
        return;
    }
    if (height[iRoot] < height[jRoot]) {
        updateParent(iRoot, jRoot);
        updateHeight(jRoot, iRoot);
    }
    else  {
        updateParent(jRoot, iRoot);
        updateHeight(iRoot, jRoot);
    }
}
```



```java
private void doPathCompression(int i) {
    // TO BE IMPLEMENTED update parent to value of grandparent
    parent[i] = parent[parent[i]];

}
```

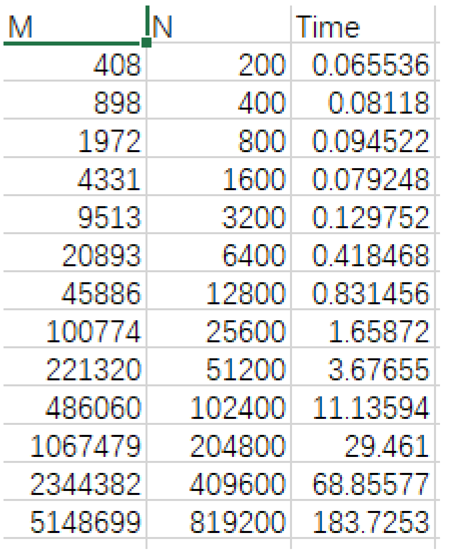
Output:

Depth2loop.csv



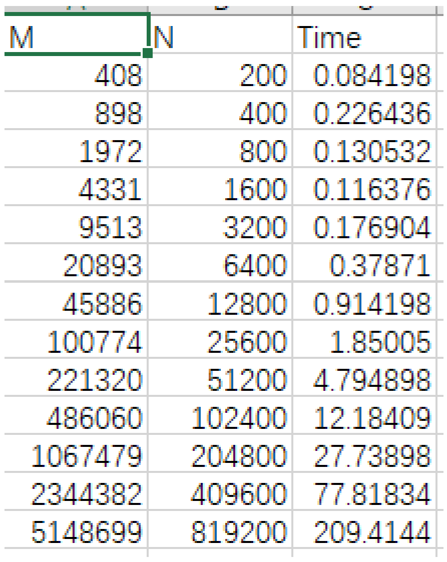
M	N	Time
408	200	0.09871
898	400	0.082782
1972	800	0.107394
4331	1600	0.152438
9513	3200	0.232842
20893	6400	0.52313
45886	12800	1.06557
100774	25600	2.107292
221320	51200	4.889484
486060	102400	16.12398
1067479	204800	30.48743
2344382	409600	78.96176
5148699	819200	187.0575

Size1loop.csv



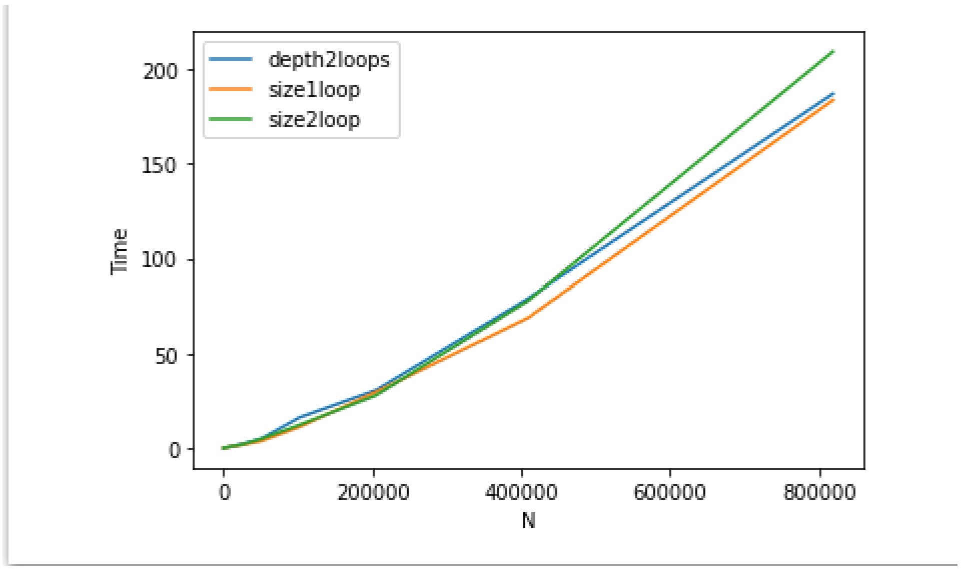
M	N	Time
408	200	0.065536
898	400	0.08118
1972	800	0.094522
4331	1600	0.079248
9513	3200	0.129752
20893	6400	0.418468
45886	12800	0.831456
100774	25600	1.65872
221320	51200	3.67655
486060	102400	11.13594
1067479	204800	29.461
2344382	409600	68.85577
5148699	819200	183.7253

Size2loop.csv:



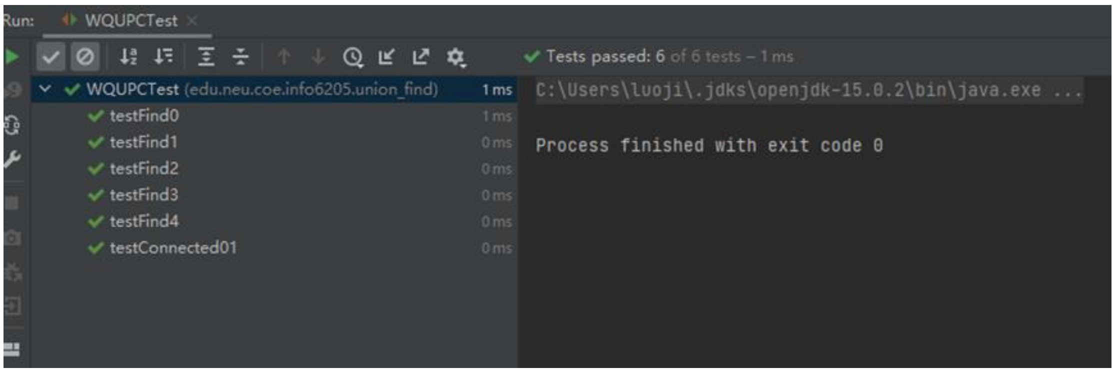
M	N	Time
408	200	0.084198
898	400	0.226436
1972	800	0.130532
4331	1600	0.116376
9513	3200	0.176904
20893	6400	0.37871
45886	12800	0.914198
100774	25600	1.85005
221320	51200	4.794898
486060	102400	12.18409
1067479	204800	27.73898
2344382	409600	77.81834
5148699	819200	209.4144

Graphical representation:



Unit tests result:

UF_HWQUPC_test


Run: WQUPCTest

Tests passed: 6 of 6 tests – 1 ms

```
WQUPCTest (edu.neu.coe.info6205.union_find)   1 ms
  testFind0                                   1 ms
  testFind1                                   0 ms
  testFind2                                   0 ms
  testFind3                                   0 ms
  testFind4                                   0 ms
  testConnected01                             0 ms
```

```
C:\Users\luoji\.jdks\openjdk-15.0.2\bin\java.exe ...

Process finished with exit code 0
```

Conclusion: As can be seen from figure, the benchmark time of the size1loop algorithm is smaller than the nested loop (depth2loop and size2loop) and as the value of N increases, the benchmark time of size2loop should be gradually smaller than that of depth2loop. In short, the benchmark time of a single-layer loop is always less than multiple cycles