Dayu Jia (NUID: 001569081)

INFO 6205

Program Structures & Algorithms

Fall 2020

Assignment No4

1. Depth rather than size:

I will solve this problem after the Question2.

2. Path compression

At first, I wrote the new find() method:

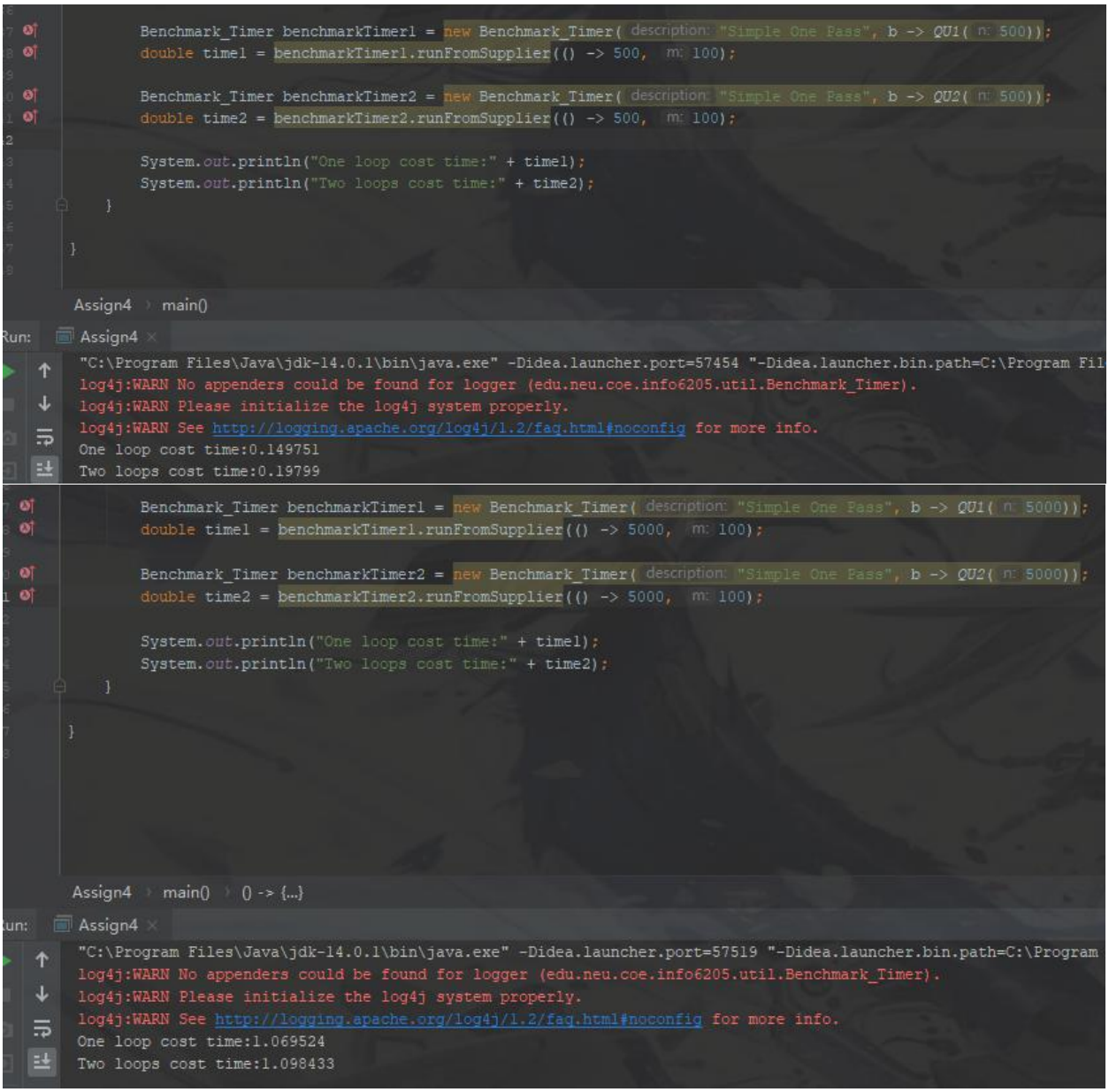
```
public int find2(int p) {
    validate(p);
    int root = p;
    int init = p;
    while (root != parent[root]){
        root = parent[root];
    }

    while(root != init) {
        int x = parent[init];
        parent[init] = root;
        init = x;
    }

    return root;
}
```

And added the Assign4.java to test:
I tried these two ways with 500, 5000 and 50000 elements many times, and there are some outputs:

Dayu Jia (NUID: 001569081)



Dayu Jia (NUID: 001569081)

```
public static void main(String[] args) {

    Benchmark_Timer benchmarkTimer1 = new Benchmark_Timer( description: "Simple One Pass", b -> QU1( n: 50000));
    double time1 = benchmarkTimer1.runFromSupplier(() -> 50000,  m: 100);

    Benchmark_Timer benchmarkTimer2 = new Benchmark_Timer( description: "Simple One Pass", b -> QU2( n: 50000));
    double time2 = benchmarkTimer2.runFromSupplier(() -> 50000,  m: 100);


    System.out.println("One loop cost time:" + time1);
    System.out.println("Two loops cost time:" + time2);
}
```

```
Assign4 › main()
Assign4 ×
"C:\Program Files\Java\jdk-14.0.1\bin\java.exe" -Didea.launcher.port=57837 "-Didea.launcher.bin.path=C:\Program File
log4j:WARN No appenders could be found for logger (edu.neu.coe.info6205.util.Benchmark_Timer).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
One loop cost time:13.858095
Two loops cost time:14.967535

Process finished with exit code 0
```

Apparently, the former find() method we used is better. But the difference between these two methods is not so much, especially when the number of elements is small enough.

Considering our Question1:
I changed the union() method:

```
private void mergeComponents(int i, int j) {
    // TO BE IMPLEMENTED make shorter root point to taller one

    /*if (height[i] < height[j]) {
        updateParent(i, j);
        updateHeight(j, i);
    }
    else {
        updateParent(j, i);
        updateHeight(i, j);
    }*/

    if (height[i] < height[j]) {
        updateParent(i, j);
    } else if (height[i] > height[j]) {
        updateParent(j, i);
    } else {
        updateParent(i, j);
        height[j]++;
    }
}
```

And run it with 500, 5000, 50000 elements:

```
One loop cost time:1.062608
Two loops cost time:1.128996
```

```
One loop cost time:0.170242
Two loops cost time:0.17549700000000001
```

```
log4j:WARN See http://logging.apache.org/l
One loop cost time:12.967683
Two loops cost time:14.730618999999999
```

We can find out that there seems no difference between these two methods, and it still helps us to confirm our conclusion about the Path Compression.