Sribalaji Prabhakar (001517537)

Program Structures & Algorithms

Spring 2021

Assignment No. 4

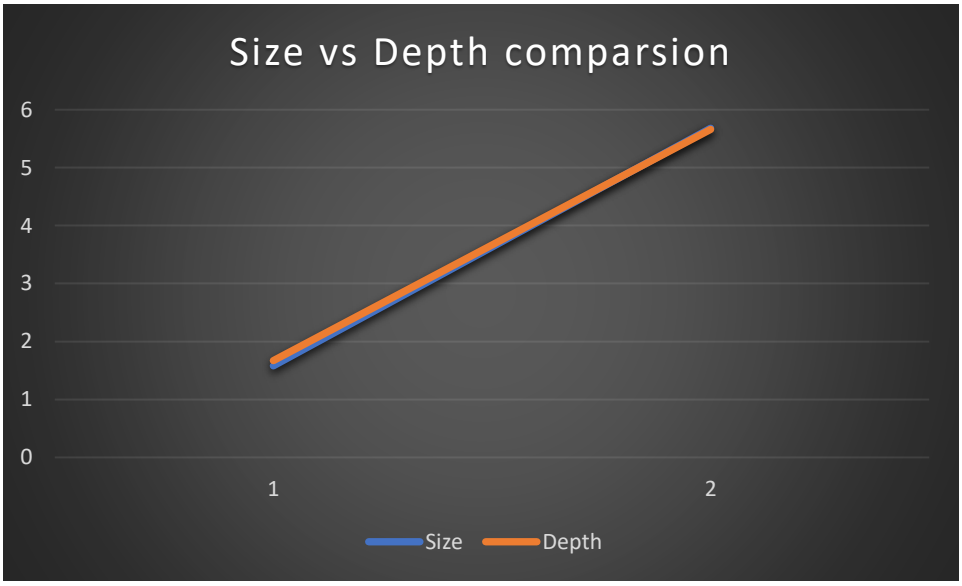
Task:

Our task is to Implement a weighted quick union and store the depth instead of size. Then to implement double path compression so that all the intermediate nodes points to the root.

Part 1:

Added two new Java classes UF_Alternate_1 and WQU_Size. The benchmark comparison for Size and Depth is below. To connect 10000 sites into a single component, the difference is significantly less after replacing the depth by size.

```
"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe" ...
2021-03-02 03:54:20 INFO  Benchmark_Timer - Begin run: WeightedUnionFindRun 10000 with 100 runs
Weighted QuickUnion by storing the size. To connect 10000 sites takes 1.58 milliseconds
2021-03-02 03:54:20 INFO  Benchmark_Timer - Begin run: WeightedUnionFindRun 10000 with 100 runs
Alternate method weighted QuickUnion by storing the depth of the tree. TO connect 10000 sites takes 1.67 milliseconds
```



Part 2:

Added a new class UF_Alternate_2_PC and added the double path compression to point all the intermediate nodes to the root and used the existing class WQUPC to test the benchmark result.

To connect 100000 sites the path compression took only 17.5 milliseconds. As expected, the double path compression took much less time because all the intermediate nodes will be close to the root, thus flattening the tree.

```
"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe" ...
2021-03-02 04:01:11 INFO  Benchmark_Timer - Begin run: WeightedUnionFindRun 100000 with 200 runs
Weighted QuickUnion by storing the size. To connect 100000 sites it takes 26.39 milliseconds
2021-03-02 04:01:17 INFO  Benchmark_Timer - Begin run: WeightedUnionFindRun 100000 with 200 runs
Weighted quick union with double path compression. TO connect 100000 sites it takes 17.50 milliseconds

Process finished with exit code 0
```

The lower value is better.