Pallak Singh (NUID: 001063095) - Section 02

# INFO 6205
# Program Structures & Algorithms
# Fall 2020
# Assignment No 4

- **Task**

  We were asked to perform benchmarking for alternatives for weighted quick union and weighted quick union with path compression. To summarize, our task was to do benchmarking and performance analysis for:

  1. Weighted quick union
  2. Height weighted quick union
  3. Weighted quick union with one pass path compression (grandparent fix)
  4. Weighted quick union with two pass path compression (connecting intermediate nodes to root)

- **Output**

  I created a new file called UnionFindAlternativesBenchmarking.java where I implemented count functions that generate random pair of elements within the range 0 – (n-1) and union them if they are not already connected until all nodes in the graph are connected (or number of components = 1). I did this for all four scenarios (weighted quick union, height weighted quick union, weighted quick union with one pass path compression (grandparent fix), and weighted quick union with two pass path compression (connecting intermediate nodes to root). I conducted benchmarking test for range n = 100 to n = 819,200 doubling n at each run, so a total of 14 values of n and 20 runs for each n. For each value of n, I also found the maximum depth of the tree at the end of connecting all components.

  ***Console Output for Benchmarking (for n = 100 to n = 819,200)***

```
2020-10-12 00:06:19 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 100 and weighted quick union, time taken to connect all nodes is: 0.05
2020-10-12 00:06:19 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 100 and height weighted quick union, time taken to connect all nodes is: 0.05
2020-10-12 00:06:19 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 100 and weighted quick union with one pass path compression, time taken to connect all nodes is:
0.1
2020-10-12 00:06:19 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 100 and weighted quick union with two pass path compression, time taken to connect all nodes is:
0.05

..........................................................................................

2020-10-12 00:06:19 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 200 and weighted quick union, time taken to connect all nodes is: 0.05
2020-10-12 00:06:19 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 200 and height weighted quick union, time taken to connect all nodes is: 0.1
2020-10-12 00:06:19 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 200 and weighted quick union with one pass path compression, time taken to connect all nodes is:
0.1
2020-10-12 00:06:19 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 200 and weighted quick union with two pass path compression, time taken to connect all nodes is:
0.1

..........................................................................................
…
…
```

```
…
…
.............................................................................................

2020-10-12 00:06:25 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 204800 and weighted quick union, time taken to connect all nodes is: 86.8
2020-10-12 00:06:27 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 204800 and height weighted quick union, time taken to connect all nodes is: 88.5
2020-10-12 00:06:29 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 204800 and weighted quick union with one pass path compression, time taken to connect all nodes
is: 56.65
2020-10-12 00:06:30 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 204800 and weighted quick union with two pass path compression, time taken to connect all nodes
is: 55.4

.............................................................................................

2020-10-12 00:06:31 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 409600 and weighted quick union, time taken to connect all nodes is: 206.5
2020-10-12 00:06:36 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 409600 and height weighted quick union, time taken to connect all nodes is: 210.75
2020-10-12 00:06:40 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 409600 and weighted quick union with one pass path compression, time taken to connect all nodes
is: 121.15
2020-10-12 00:06:43 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 409600 and weighted quick union with two pass path compression, time taken to connect all nodes
is: 119.65

.............................................................................................

2020-10-12 00:06:46 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 819200 and weighted quick union, time taken to connect all nodes is: 542.8
2020-10-12 00:06:58 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 819200 and height weighted quick union, time taken to connect all nodes is: 525.85
2020-10-12 00:07:09 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 819200 and weighted quick union with one pass path compression, time taken to connect all nodes
is: 357.55
2020-10-12 00:07:17 INFO  Benchmark_Timer - Begin run: Benchmark Test for Union Find Experiment with 20 runs
For nodes: 819200 and weighted quick union with two pass path compression, time taken to connect all nodes
is: 331.3
```

I have summarized the benchmarking results in the table below:

| n | Weighted Quick Union | Height Weighted Quick Union | Weighted Quick Union With One Pass Path Compression | Weighted Quick Union With Two Pass Path Compression |
|---|---|---|---|---|
| 100 | 0.05 ms | 0.05 ms | 0.1 ms | 0.05 ms |
| 200 | 0.05 ms | 0.1 ms | 0.1 ms | 0.1 ms |
| 400 | 0.2 ms | 0.5 ms | 0.2 ms | 0.25 ms |
| 800 | 1.05 ms | 0.3 ms | 0.85 ms | 0.4 ms |
| 1600 | 0.55 ms | 1.8 ms | 1.0 ms | 0.45 ms |
| 3200 | 0.85 ms | 1.55 ms | 1.1 ms | 1.2 ms |
| 6400 | 2.95 ms | 2.0  ms | 0.9 ms | 1.1 ms |
| 12,800 | 3.55 ms | 7.55 ms | 2.25 ms | 2.3 ms |
| 25,600 | 8.75 ms | 7.1 ms | 4.4 ms | 14.7 ms |
| 51,200 | 18.2 ms | 20.55 ms | 11.9 ms | 10.0 ms |
| 102,400 | 36.8 ms | 43.9 ms | 25.25 ms | 25.1 ms |
| 204,800 | 86.8 ms | 88.5 ms | 56.65 ms | 55.4 ms |
| 409,600 | 206.5 ms | 210.75 ms | 121.15 ms | 119.65 ms |

| | | | | |
|---|---|---|---|---|
| 819,200 | 542.8 ms | 525.85 ms | 357.55 ms | 331.3 ms |

*Table 2.1: Results recording benchmarking results for weighted quick union, height weighted quick union, weighted quick union with one pass and two pass path compression*

Plotting the results in the graph below:

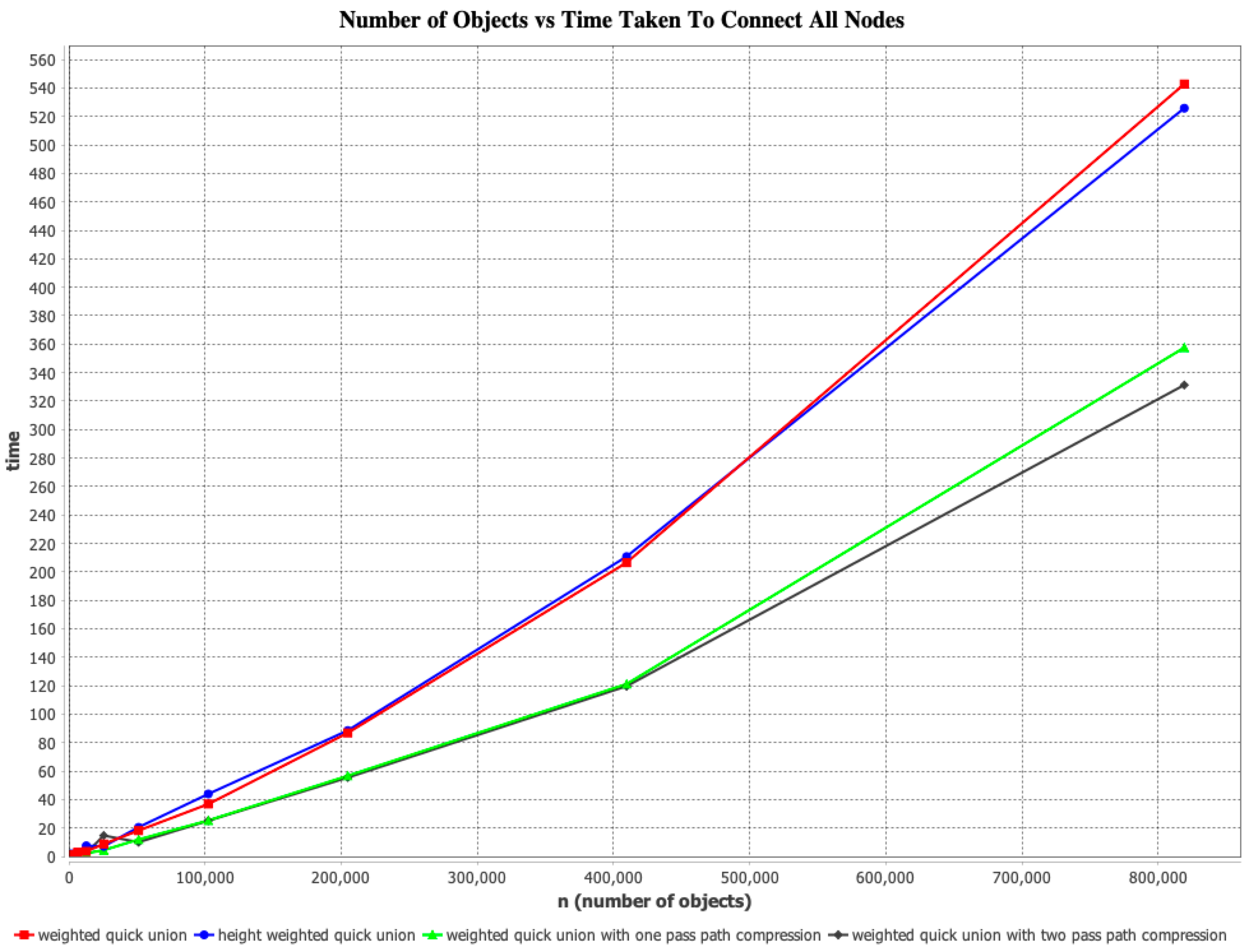

*Fig 2.1: Plotting the Number of Objects (n) vs Time Taken to Connect all Nodes for the range n = 100 to n = 819200 doubling n at each run*

***Console Output for average depth at the end of quick union (for n = 100 to n = 819,200)***

```
For nodes: 100 and weighted quick union, the average depth is: 3
For nodes: 100 and height weighted quick union, the average depth is: 5
For nodes: 100 and weighted quick union with one pass path compression, the average depth is: 2
For nodes: 100 and weighted quick union wuth two pass path compression, the average depth is: 2

...........................................................................................
```

```
For nodes: 200 and weighted quick union, the average depth is: 4
For nodes: 200 and height weighted quick union, the average depth is: 5
For nodes: 200 and weighted quick union with one pass path compression, the average depth is: 2
For nodes: 200 and weighted quick union wuth two pass path compression, the average depth is: 1

…
…
…

For nodes: 204800 and weighted quick union, the average depth is: 8
For nodes: 204800 and height weighted quick union, the average depth is: 9
For nodes: 204800 and weighted quick union with one pass path compression, the average depth is: 2
For nodes: 204800 and weighted quick union wuth two pass path compression, the average depth is: 2

.........................................................................................................

For nodes: 409600 and weighted quick union, the average depth is: 8
For nodes: 409600 and height weighted quick union, the average depth is: 8
For nodes: 409600 and weighted quick union with one pass path compression, the average depth is: 2
For nodes: 409600 and weighted quick union wuth two pass path compression, the average depth is: 2

.........................................................................................................

For nodes: 819200 and weighted quick union, the average depth is: 8
For nodes: 819200 and height weighted quick union, the average depth is: 9
For nodes: 819200 and weighted quick union with one pass path compression, the average depth is: 2
For nodes: 819200 and weighted quick union wuth two pass path compression, the average depth is: 1
```

I have summarized the maximum depth results in the table below:

| n | Weighted Quick Union | Height Weighted Quick Union | Weighted Quick Union With One Pass Path Compression | Weighted Quick Union With Two Pass Path Compression |
|---|---|---|---|---|
| 100 | 3 | 5 | 2 | 2 |
| 200 | 4 | 5 | 2 | 1 |
| 400 | 4 | 5 | 1 | 1 |
| 800 | 5 | 6 | 2 | 2 |
| 1600 | 5 | 6 | 2 | 2 |
| 3200 | 6 | 6 | 1 | 2 |
| 6400 | 6 | 6 | 1 | 2 |
| 12,800 | 6 | 7 | 2 | 2 |
| 25,600 | 6 | 7 | 1 | 2 |
| 51,200 | 7 | 8 | 2 | 2 |
| 102,400 | 7 | 7 | 2 | 2 |
| 204,800 | 8 | 9 | 2 | 2 |
| 409,600 | 8 | 8 | 2 | 2 |
| 819,200 | 8 | 9 | 2 | 1 |

*Table 2.2: Results recording final depth of tree at the end for weighted quick union, height weighted quick union, weighted quick union with one pass and two pass path compression*

- **Relationship Observation and Conclusion**

  We observe that the benchmarking results for height weighted quick union and weighted quick union are almost similar. This was expected because weighted quick union and height weighted quick union have same upper bounds. This is because of a direct relationship between the size and the height. For instance, in quick union weighted according to the size, if a tree has a height of 1, it must have at least two nodes in it because to get a height of one, two nodes must be joined together. Similarly, for a height of two, it must have at least 4 nodes because to get a height of 2, two trees of height 2 must be joined together. We can generalize this by saying a tree of height n must have at least $2^n$ nodes in it. Or, a tree with n number of nodes has a height of at most log n. Since the height is log n in both types of weighted unions, the find and union operation have a performance of O(log (n)) because during the find operation, we are traversing the height which is at most log n in both cases. For union of n nodes, the performance is O(n log(n)).

  We also observe that the performance of weighted quick union with path compression is significantly better than only weighted quick union. This is because in both one pass and two pass path compression, we are reducing our height. In the one pass path compression or the grandparent fix, we set the parent of a node to its grandparent. While in two pass path compression, we make each of the intermediate node during the find operation point to the root. The performance for path compression while joining n nodes is O(n $\log_*$ n) where $\log_*$ refers to the iterated log function. We can show this using the inverse of Ackerman equation:

$$\log * n = \begin{cases} 0 & n \le 1 \\ 1 + log * (logn) & n > 1 \end{cases}$$

  However, for both types of path compression there is no significant difference in their performance. Theoretically, we would expect our one pass solution (grandparent fix) to be better in terms of performance as it only takes one loop to implement during the find operation. However, looking at the depths in both types of path compression, we see that the tree becomes almost flat in both cases and hence, the performance for both are very similar as well. Observing the depth of the tree for all four instances, we see that path compression maintains a tree which is almost flattened while this is not the case for weighted or height weighted quick union, hence, the significantly better performance.

- **Files used in this assignment**
  - The benchmarking test was performed using Benchmark_Timer.java present in /util folder.
  - The experiment was conducted in the UnionFindAlternativesBenchmarking.java in /union_find folder
  - The weighted quick union is performed using WeightedQuickUnion.java in /union_find folder by setting path compression to true or false and setting typePathCompression to 1 or 2 referring to point to root fix (two pass) and grandparent fix (one pass) respectively.
  - The height weighted quick union is performed using UF_HWQUPC.java in /union_find folder
  - The graphs were plotted using UFClientPlotter.java which uses JFreeChart package.