Assignment – 4

Program Structure and Algorithms

Pranav Agarwal

NUID – 001099801

1. Weight by height vs size –

(a) When merging by height, the shortest tree will always be the child of the longer one. The only time the height of a node increases is when an attempt is made to join 2 trees with the same height. In this case, the parent has its height increased by 1.

This means that to get a tree with higher height, the number of nodes has to at least double. In other words, the maximum height of a tree with N nodes is at most lg(n)

$$h <= lg(n)$$

(b) When merging by size, we make the tree with less nodes the child of the one with greater. We also then increase the size of the parent by the size of the child. We can prove inductively that the size of a tree is always greater than or equal to 2 raised to its height (same intuition as in part (a)). This is because again, the only way a height of a tree increases is if 2 trees with the same height are merged. Even if we don't store the heights or increase them explicitly, the resulting tree will always follow this rule.

$$s >= 2^h$$

Since s is nothing but the total number of nodes in a tree, when all nodes are merged into a single tree, s will be equal to n. So,

$$n >= 2^h$$

$$\text{i.e} \quad h <= lg(n)$$

We find that the upper bound on height is same no matter if the weighting is done by height or size. Since upper bound on height directly decides upper bound on performance, we can safely conclude that the big O notation of time taken will be same for both weighting methods. So experimental benchmarking is not required.

2. Benchmarking to compare path halving, path compression, and no compression.

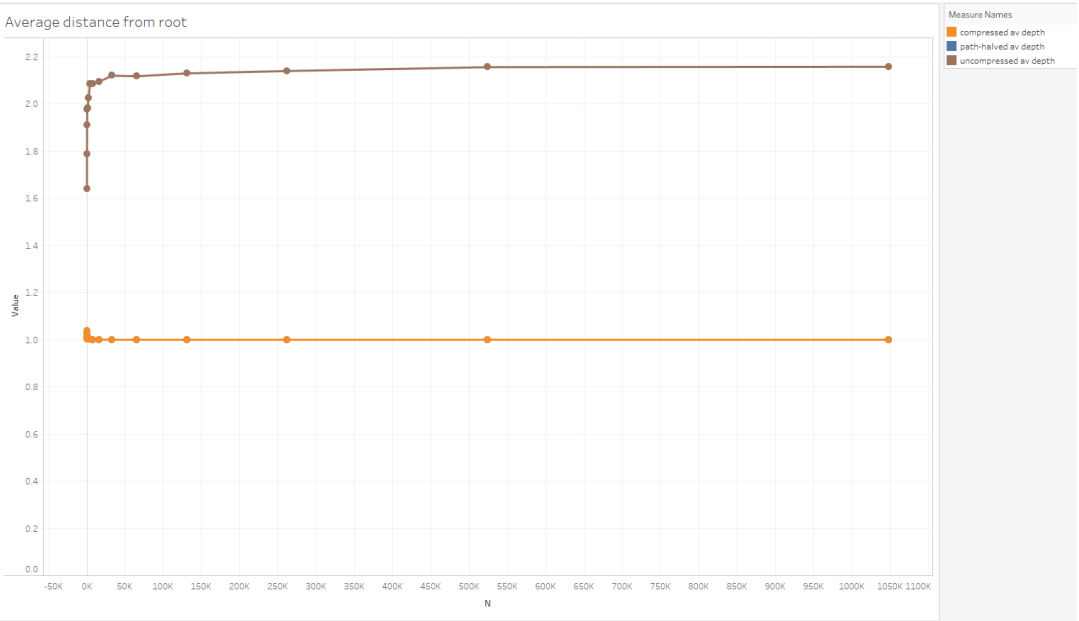
These 3 variations of UF have been compared on 3 metrics –

- Time taken for full connectivity in ms
- Average distance of node from root in final tree
- Number of random pairs generated for full connectivity

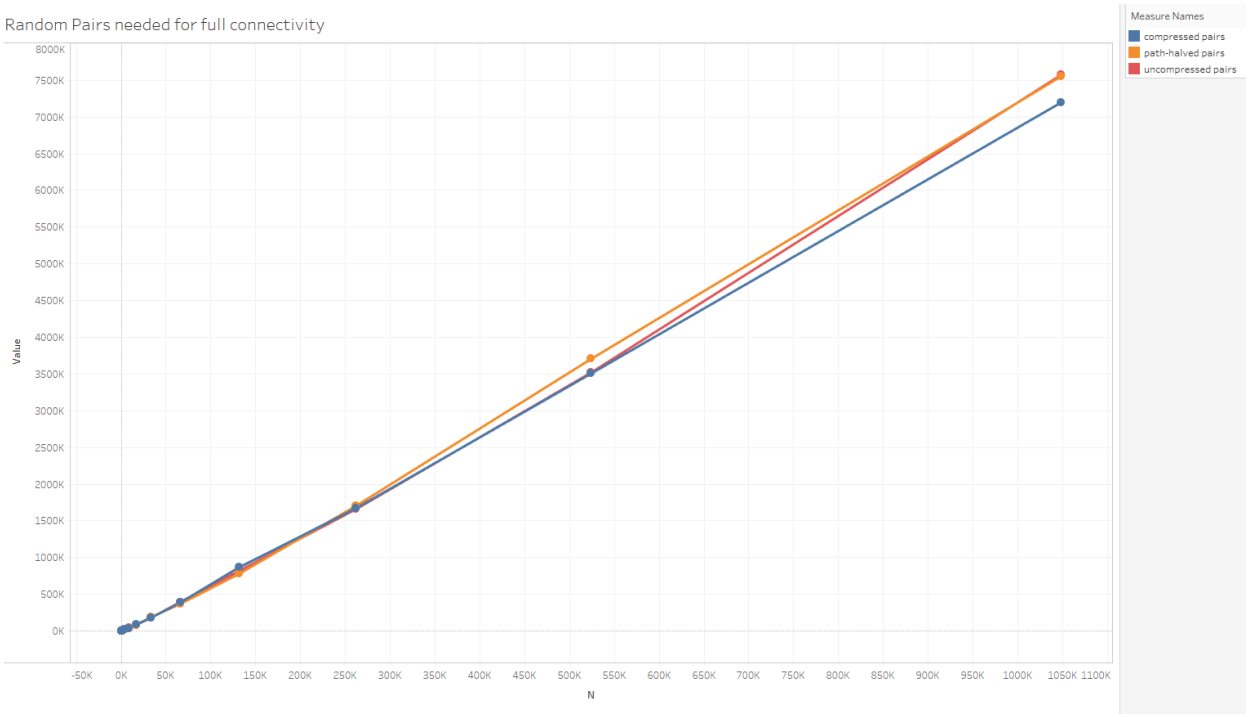
Results –

N	uncompressed time	uncompressed av depth	uncompressed pairs	path-halved time	path-halved av depth	path-halved pairs	compressed time	compressed av depth	compressed pairs
64	0.36	1.639063	134.40	0.07	1.0265625	155.60	0.26	1.0390625	143.30
128	0.09	1.786719	354.30	0.44	1.0101563	380.10	0.10	1.0242188	318.30
256	0.47	1.910156	752.80	0.30	1.0050781	741.20	0.29	1.0074219	753.10
512	0.29	1.977734	1,544.10	0.26	1.0041016	1,730.50	0.45	1.0029297	1,896.10
1,024	1.18	1.982520	3,671.70	0.85	1.0007813	3,836.10	2.23	1.0018555	4,006.10
2,048	2.50	2.023877	8,070.60	2.07	1.0014160	7,895.40	3.13	1.0003906	8,767.90
4,096	2.96	2.084009	18,859.60	1.81	1.0006592	18,883.00	1.79	1.0004150	17,428.30
8,192	4.45	2.085059	41,128.20	3.58	1.0002686	37,393.30	4.16	1.0002563	37,269.80
16,384	10.20	2.094678	78,961.50	10.37	1.0002625	76,497.30	7.58	1.0000427	88,548.20
32,768	24.39	2.120013	176,008.60	21.24	1.0000183	187,044.30	20.73	1.0000702	175,078.50
65,536	54.22	2.117342	392,380.30	44.00	1.0000412	366,211.10	44.45	1.0000259	384,705.40
131,072	129.50	2.129500	816,226.90	101.34	1.0000298	775,666.70	99.05	1.0000031	864,198.10
262,144	308.14	2.139569	1,661,343.00	269.89	1.0000111	1,705,277.60	252.87	1.0000092	1,672,618.40
524,288	858.65	2.155849	3,521,193.40	673.08	1.0000042	3,707,153.10	650.07	1.0000017	3,507,562.70
1,048,576	2,402.40	2.157471	7,576,534.00	1,744.27	1.0000014	7,555,869.30	1,700.08	1.0000024	7,199,970.50

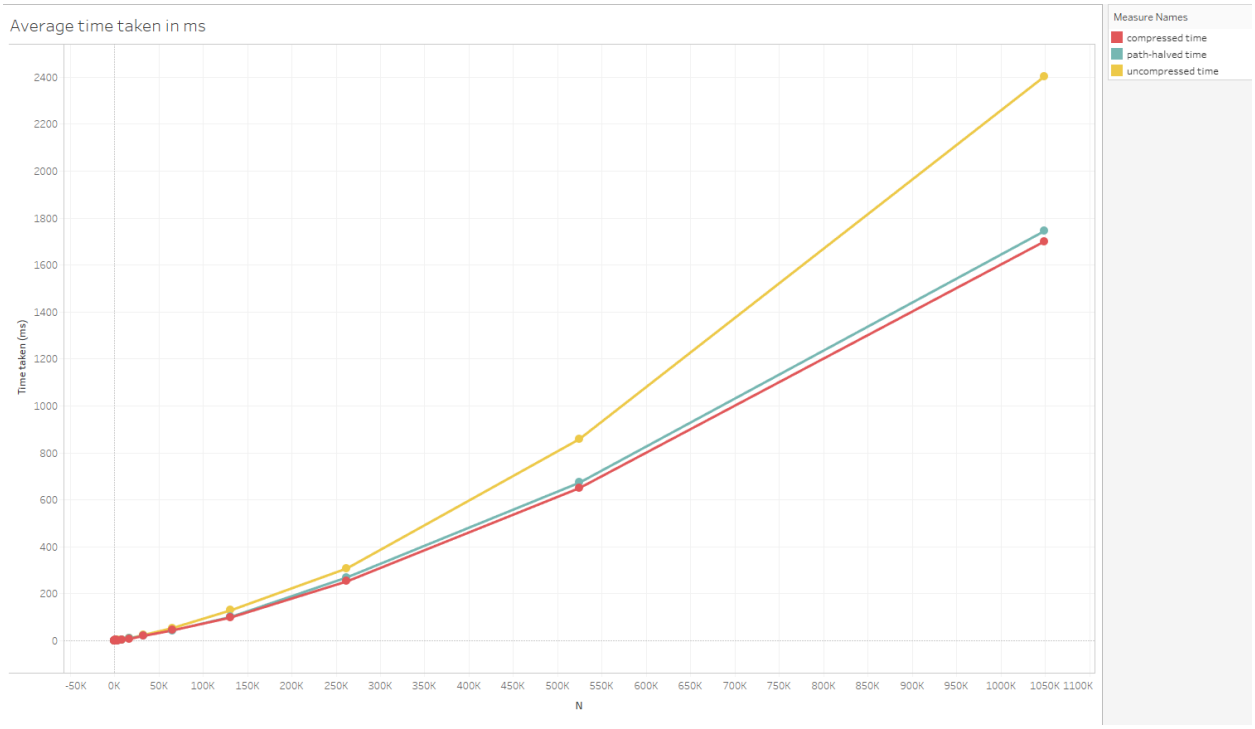
***1** - Raw Data*



2 - Average distance from root



3 - Random pairs count



4 - Average time taken (ms)

3. Observations from table and graphs–

- The average distance from the root is around 2.2 for uncompressed union find and very nearly 1 for compressed union find. The difference between path halving and full 2 loop compression is negligible and, in the graph, they overlap almost perfectly.
- The compression type chosen does not have any bearing on how many pairs will be needed to connect all components. The graph for all 3 types is almost the same and any variation can be attributed to the random nature of the input.
- The time taken by uncompressed UF is significantly more, approximately 1.3-1.4 times that of compressed UF. Time taken by path halving is slightly but consistently more than full compression.

4. Notes regarding assignment –

- The simulation can be run using Assignment4.java in the union find folder.
- The simulation data is stored in results/union_find/data1.csv
- All time benchmarking was done using the benchmark class from assignment 2.
- The method for finding average depth is averageDepth(), in both the UF_HWQUPC and the WQUPC classes.
- Count of pairs was found using the same count method as in assignment 3.
- Uncompressed UF was done using a UF_HWQUPC object with pathCompression set to false.
- Path Halving UF was done using a UF_HWQUPC object with pathCompression set to true.
- Compressed UF was done using a WQUPC object from the repository, since the implementation of 2 loop compression was already there. The WQUPC class was modified slightly to implement the UF interface, to make the benchmark easier and more consistent.
- Although there are no benchmarks for the different types of weights, the UF_HWQUPC class uses height, and the WQUPC class uses size, so the code and implementation for both is present.