_INFO6205 –Assignment4

Manasa Bhat

Section 5

NU ID: 001032278

Union Find Alternatives

Main Task Solution

1) For weighted quick union, store the depth rather than the size;

Taking WQUPC class as the reference, I have created a new class WQUPC_ALT which performs weighted quick union for objects by Depth.

When a node is shorter in depth from other, the tree with shorter node is linked to tree with larger node.

In case of equal depth, tree with first parameter node is linked to other.

2)For weighted quick union with path compression, do two loops, so that all intermediate nodes point to the root, not just the alternates.

In the class WQUPC_ALT I have implemented Path compression such that every parent node points to root.

Note: class WQUPC_ALT can be run using the client **UF_ALT_CLIENT** class with main method.

I have created separate client program for Path compression and without it.

New classes UF_ALT_CLIENT and WQUPC_ALT are in the new package union_find_alternative.

3) Benchmarking the alternative against implementation in repository.

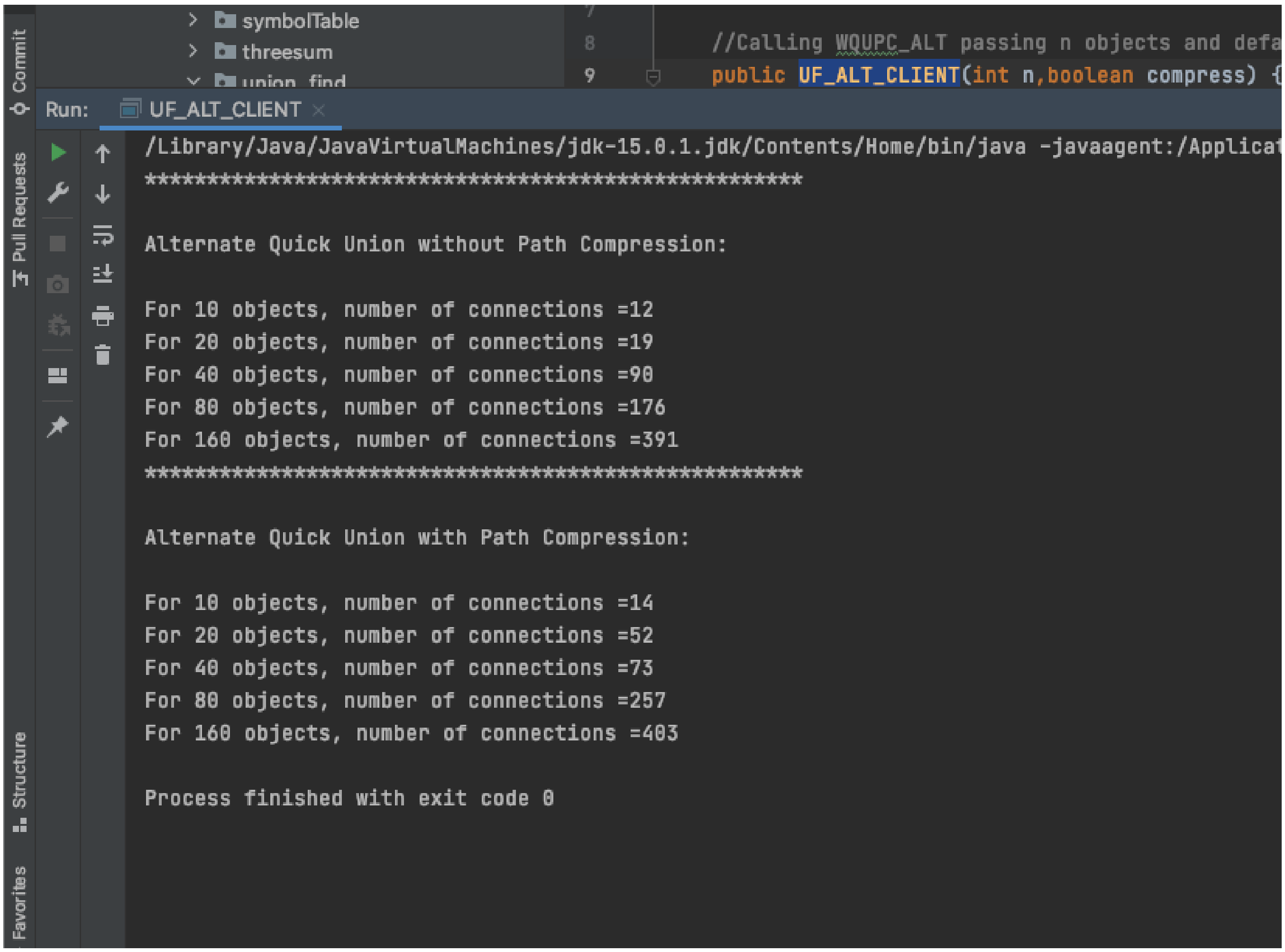
Created a test class called WQUPC_BENCHMARK_Test to benchmark Weighted quick union by size, height, depth with and without path compression. Compared for 5 different number of objects for each union.

WQUPC_BENCHMARK_Test is present in test folder under union_find_alternative package

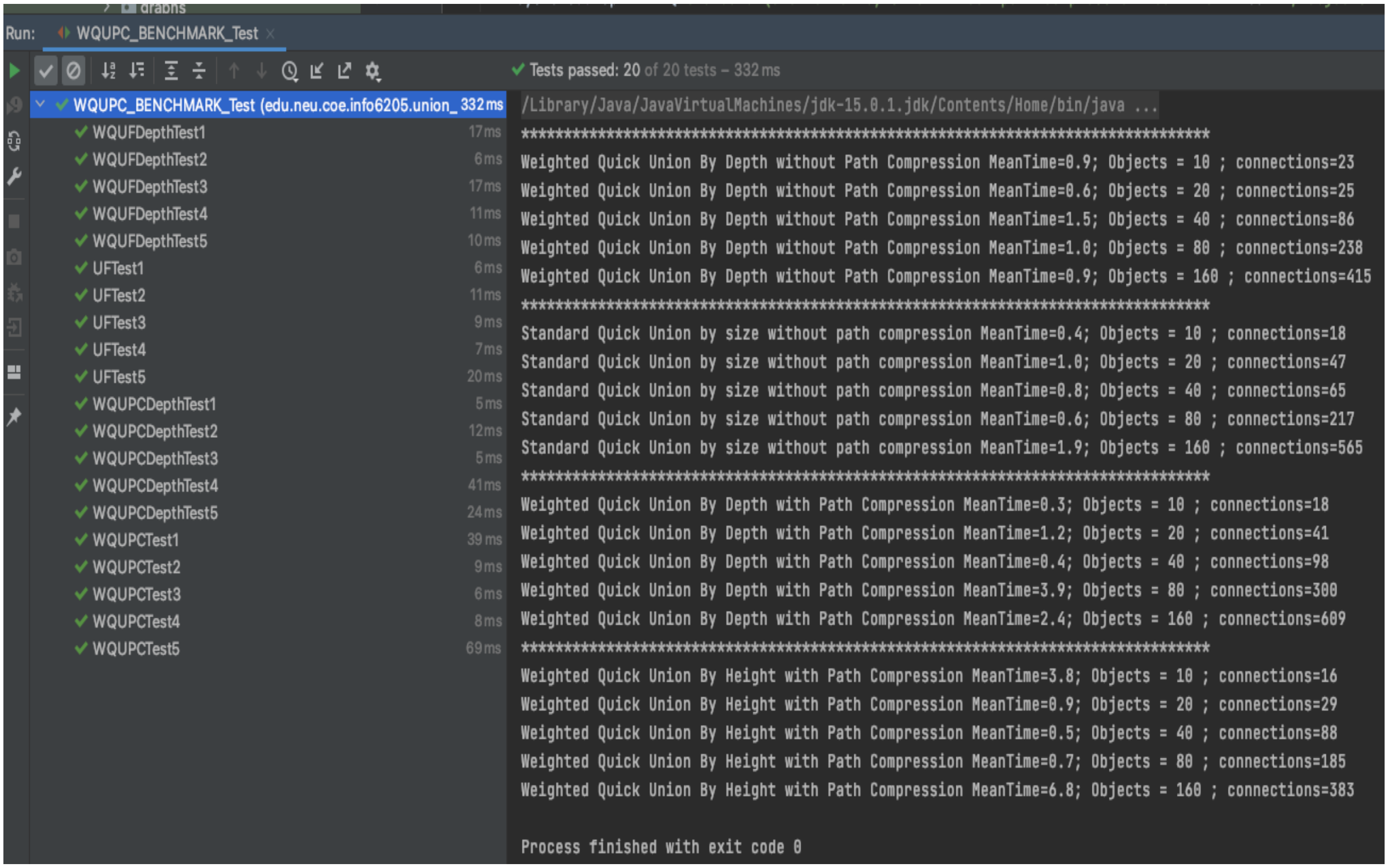
Evidence of runs:

CONSOLE OUTPUT

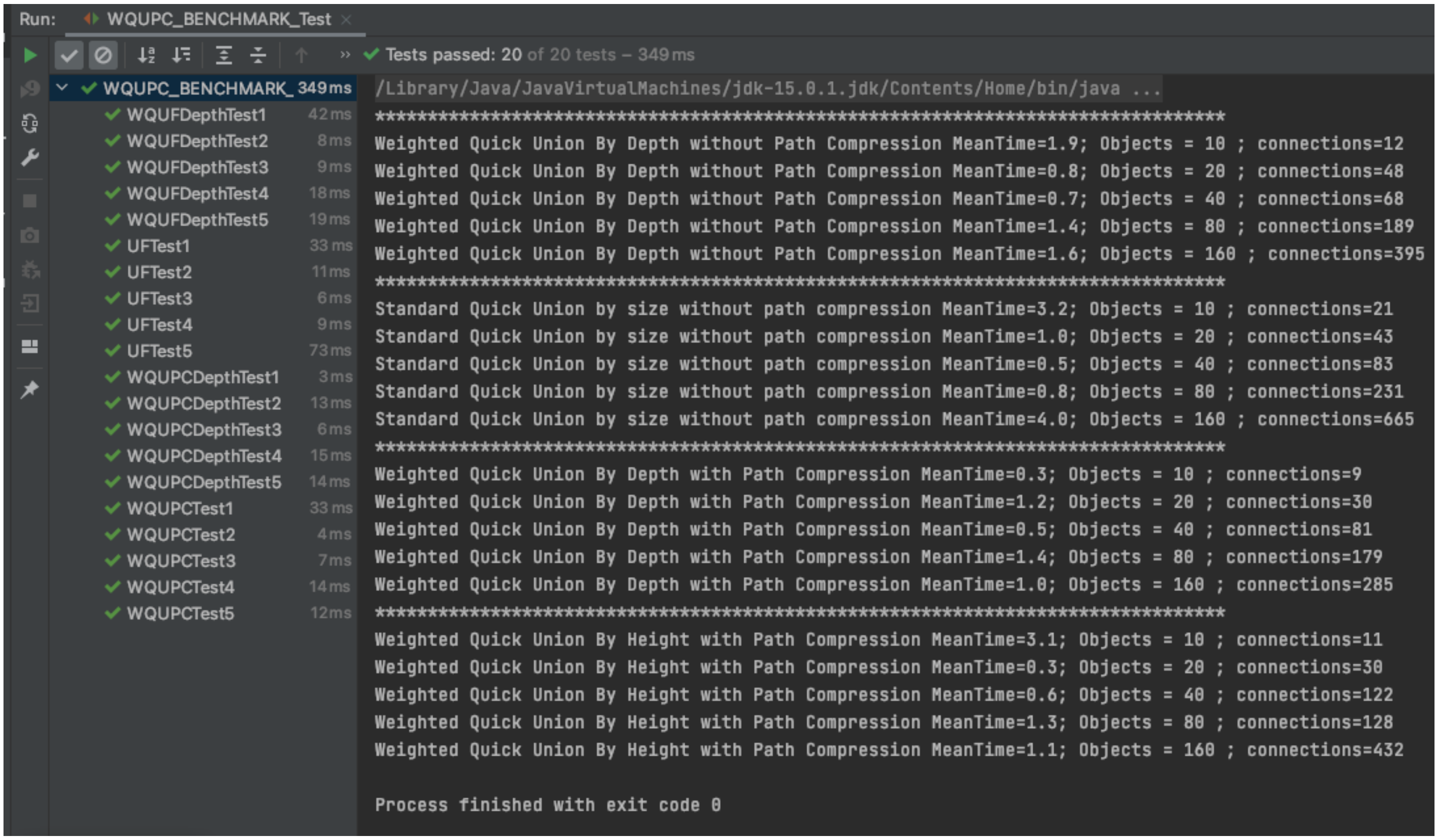
UF_ALT_CLIENT Output: Weighted Quick union by depth with and without Path compression.



WQUPC_BENCHMARK_Test output1:



<u>WQUPC_BENCHMARK_Test output 2:</u>



Experiment Observations:

- Weighted Quick union by Depth

While performing experiments on this program for different number of objects, below were my observations:

1. The number of connections increases widely as the number of objects increase for union.
2. After all components get connected, I observed that the tree connection formed is not linear. i.e, inspite of weighting by depth, the tree formed is irregular and not near to flat.
3. Weighting by height for quick union was much more easier and effective than weighting by depth.
4. **Weighting by depth was complex as we had to increase the depth of every child node in recursion when the tree with smaller depth links to larger tree.** This takes lot of time and is inefficient and complicated when tree size is larger.
5. When comparing the trees with respect to depth, we compare the node depths and link the tree with smaller node depth. **There were scenarios where the tree with larger height was linked to tree of smaller height just because the nodes compared were of same level or depth.** This is inefficient practice as it will not weight the quick union to produce a stable or almost linear structure.
6. The maximum depth node was at most the height of the tree.

- Path compression of intermediate nodes

1. Weighted Quick union by path compression required lesser unions than weight quick union without path compression.
2. Path compression of every intermediate node makes the tree almost linear and efficient.
3. Path compression by intermediate node pointing to root is much more effective than pointing alternate to root as it will reduce complexity to find and to union.

- Benchmarking of Quick unions and comparison

1. Quick union takes O(n) time complexity for each union and O(1) for each find it does.
2. When there are M connections, Quick union without weighting takes O(MN) in the worst case, If M is proportional to N then it will be O(n^2).
3. For weighted quick union the time complexity based on experiments was found to be O(logN) where N is number of objects and log to the base of 2.
4. For weighted quick union along with path compression (whether it is weighted by height or depth) the time complexity is O(M+NlogN) where M is number of connections and N is number of Objects.

Below is the order of time complexity comparison for different quick union logics:

Quick Union without weighting(O(N^2))

Quick Union by Weighting (Depth or Height) (O(logN))

Quick Union by Path Compression O(logN)

Quick union weighted by path compression O(M+NlogN)

Where M is number of connections made, N is number of objects and log is to base of 2.

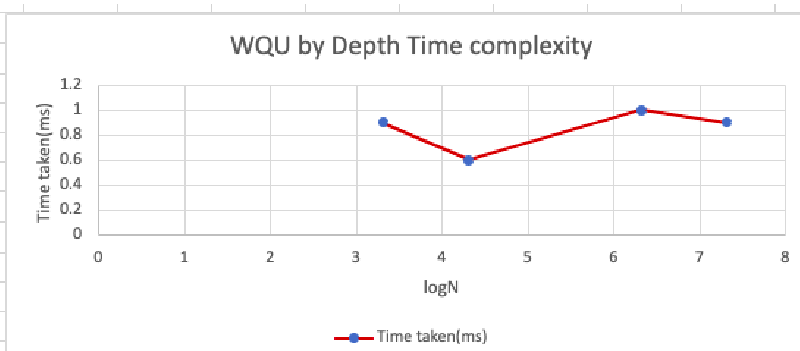
Supporting Evidence –Plots

Tabulations:

A	B	C	D	E	F
Weighted QuickUnion by Depth without Path Compression					
Number of objects(N)	Numbers of connections(M)	Time taken(ms)	logN		
10	23	0.9	3.321928		
20	25	0.6	4.321928		
40	86	1.5	5.321928		
80	238	1	6.321928		
160	415	0.9	7.321928		
Weighted QuickUnion by Depth with Path Compression					
Number of objects(N)	Numbers of connections(M)	Time taken(ms)	logn	nlogn	m+nlogn
10	18	0.4	3.321928	33.2192809	51.21928
20	41	1	4.321928	86.4385619	127.4386
40	98	0.8	5.321928	212.877124	310.8771
80	300	0.6	6.321928	505.754248	805.7542
160	609	1.9	7.321928	1171.5085	1780.508
Weighted QuickUnion by Size without Path Compression					
Number of objects(N)	Numbers of connections(M)	Time taken(ms)	MN	logN	
10	18	0.4	180	3.32192809	
20	47	0.6	940	4.32192809	
40	65	0.8	2600	5.32192809	
80	217	0.6	17360	6.32192809	
160	565	1.9	90400	7.32192809	
Weighted QuickUnion by Height with Path Compression					
Number of objects(N)	Numbers of connections(M)	Time taken(ms)	logn	nlogn	m+nlogn
10	16	3.8	3.321928	33.2192809	49.21928
20	29	0.9	4.321928	86.4385619	115.4386
40	88	0.5	5.321928	212.877124	300.8771
80	185	0.7	6.321928	505.754248	690.7542
160	383	6.8	7.321928	1171.5085	1554.508

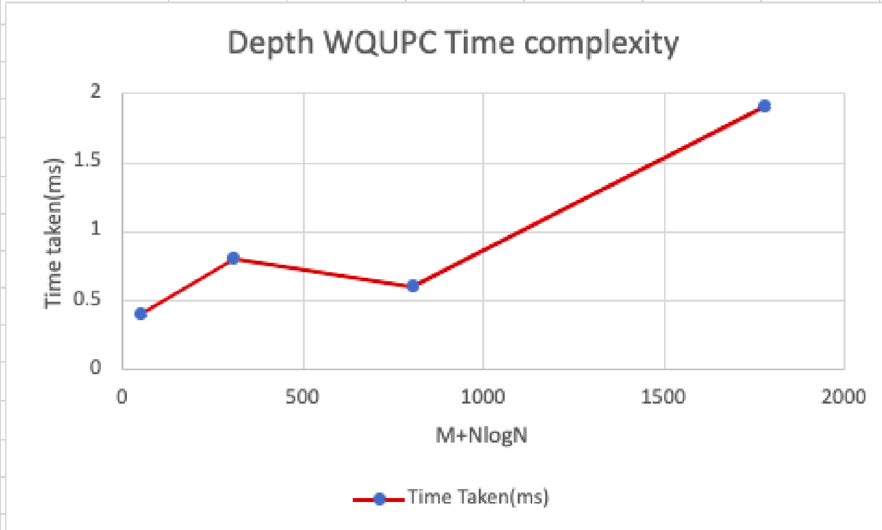
Weighted Quick Union by Depth without Pathcompression

logN	Time taken(ms)
3.32	0.9
4.32	0.6
6.32	1
7.32	0.9



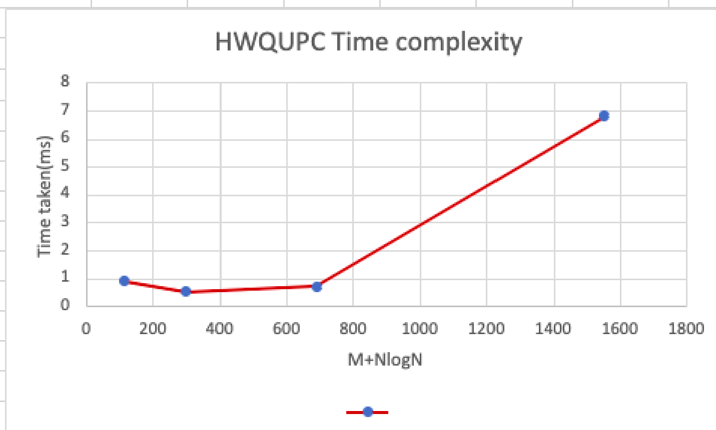
Weighted Quick Union by Depth with Pathcompression

M+NlogN	Time Taken(ms)
51.22	0.4
310.9	0.8
805.75	0.6
1780.5	1.9



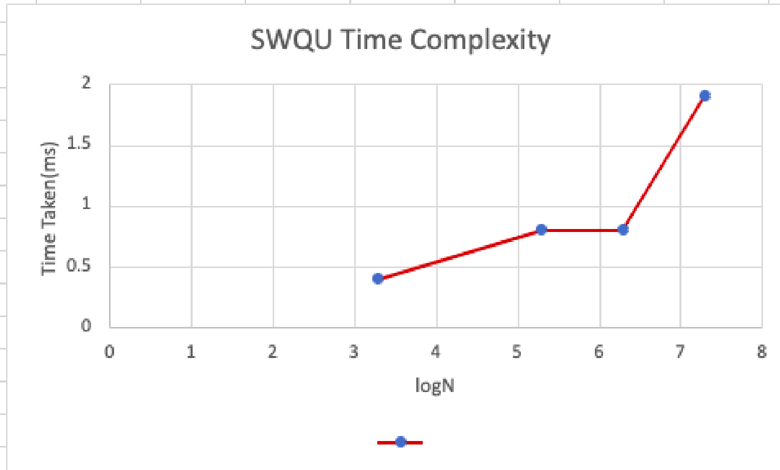
Weighted Quick Union by Height with Pathcompression

M+NlogN	Time Taken
115.44	0.9
300.9	0.5
690.75	0.7
1554.5	6.8



Weighted Quick Union by Size without Pathcompression

logN	Time Taken
3.3	0.4
5.3	0.8
6.3	0.8
7.3	1.9



As seen in the plots, Weighted quick union by height and depth is almost similar time complexity, but I could see weighted quick union by height being more faster than by depth.

Weighted quick union by height has a more linear time complexity compared to that of depth.

As stated before and based on the tabulations, Weighted quick union by Depth with path compression has time complexity of O(M+NlogN).

Conclusion

From the test results and observations, it can be concluded that weighted quick union by depth alternate will optimize the union time complexity compared to that of size or no weights, but it is inefficient when compared with weighted quick union by height.

Benchmarking of Weighted quick union by depth is inefficient in giving a linear tree or improving time complexity because :

1. On weighting by depth, when a smaller tree links to larger, all the child nodes need to be looped to increase the depth which will take longer times when objects are large.
2. We check the depth of nodes for connecting and link the trees – in this logic there will be tree of larger height being linked to tree of smaller height because of nodes being compared being equal. This will disrupt the linearity of the tree and will take longer times for union. This method would be inefficient from optimization point of view.
3. Depth of any node in the tree is at most the height of the tree which is worst case logN where N is the number of objects.
4. Quick union by path compression of intermediate node is a very good optimization method as it makes the tree almost linear and improves the time complexity.

Weighted quick union by Depth with path compression has time complexity of O(M+NlogN).

Where n is number of objects , m is connection and log is taken to base of 2.