INFO 6205 – Section 01

Giang Vu – NUID: 001537937

Assignment 4: UNION-FIND Alternative

1. Task

   We mentioned two alternatives for implementing Union-Find:

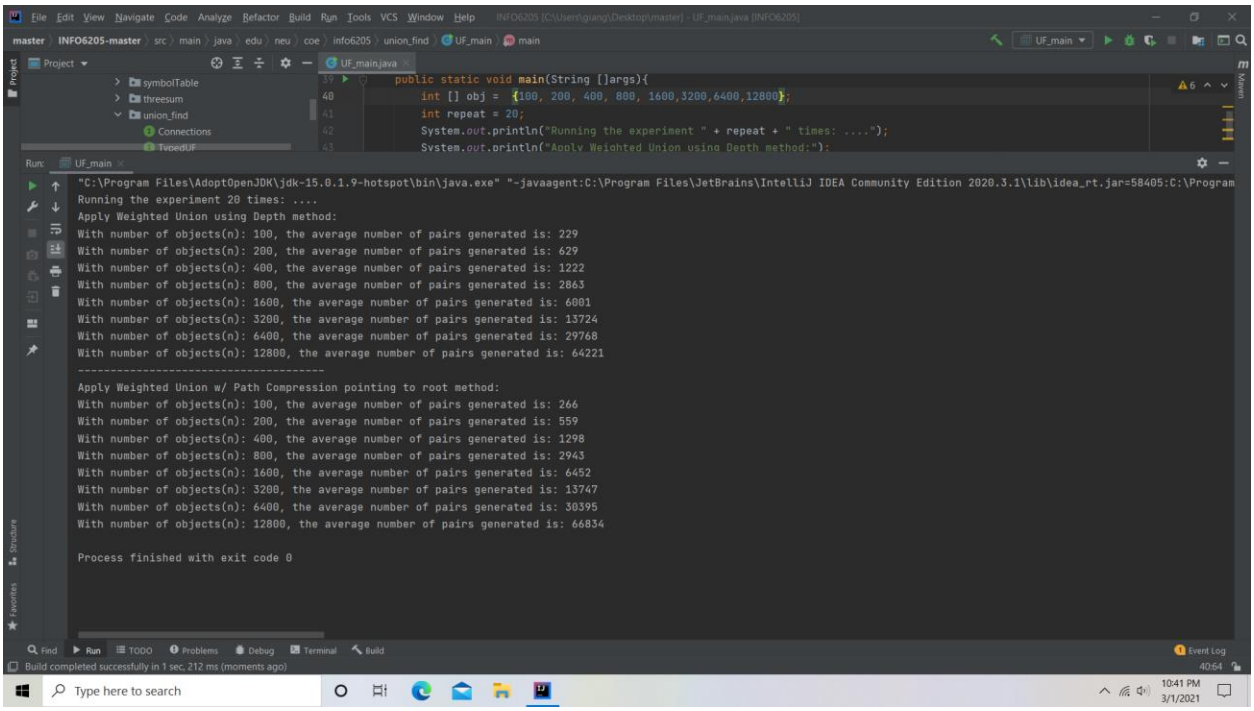
   - For weighted quick union, store the depth rather than the size.
   - For weighted quick union with path compression, do two loops, so that all intermediate nodes point to the root, not just the alternates.

   For both of these, code the alternative and benchmark it against the implementation in the repository. You have all of that available from a previous assignment.

   If you can explain why alternative #1 is unnecessary to be benchmarked, you may skip benchmarking that one.

2. Conclusion
   - I ran the experiments for each case of objects (n) 20 times to get the average number of pair (m) generated, which I will show in the table result below. I believe the pair generated are approximately equal among assignment 3 implementation compare to our 2 alternative implementations of this assignment. Thus, I do believe they will lead to the same running time/performance.
   - I believe alternative# (using depth) is unnecessary to benchmarked since:
     + If we weighted a tree by size(n), when tree has 1 node (n =1) -> height (h) = 1. We learnt that h only increases if tree is merged with another tree has size >= n. Thus, we can see: h =2 , n >=2 | h = 3, n > = 4 | so on. So, the upper bound of h will be lg(n) +1.
     + If we weighted a tree by depth, when h =2, tree would need at least 2 nodes. When h = 3, n >=4 (since 2 tree with h =2 merged with minimum nodes of each tree). We can see the upper bound of h using this method will also be lg(n) +1. So in this case, no matter we using height or size, the depth <= lg(n) +1.
3. Evidence

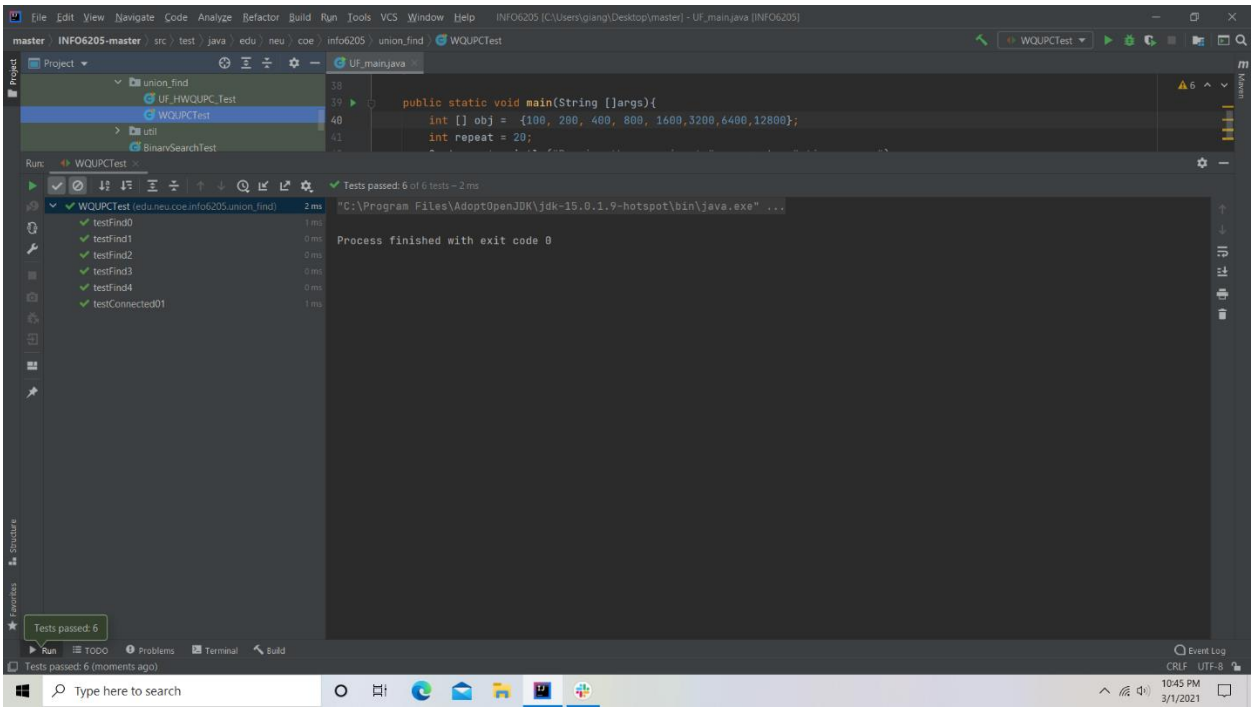
Objects (n)	Pairs generated (m)		
	Assignment 3 Implement	UF_using_depth Implement	WQUPC point to root Implement
100	258	229	266
200	568	629	559
400	1341	1222	1298
800	2926	2863	2943
1600	6179	6001	6452
3200	13984	13724	13747
6400	29584	29867	30395
12800	68767	64221	66834



4. Code

   File modified including UF_HWQUPC.java (for using depth implementation), WQUPC.java (for point to root implementation) and UF_main.java for benchmarking. If you have trouble running it, please download the master folder on github repo for running/testing.

5. Unit test
   (Test Point to Root implementation)



(Test using Depth implementation)