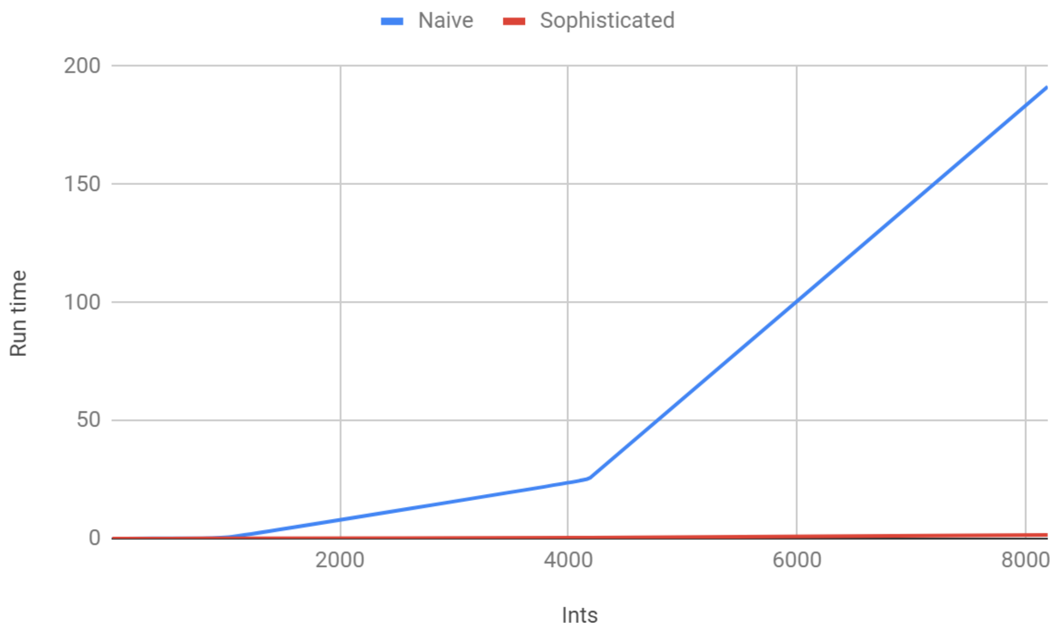
ECE 573 Spring 2019 HW#1 Report
Yuhang Zhou
yz853
2/17/2019

Q1.
Implementation in C++ is in the attachment 3sum.cpp file
Result

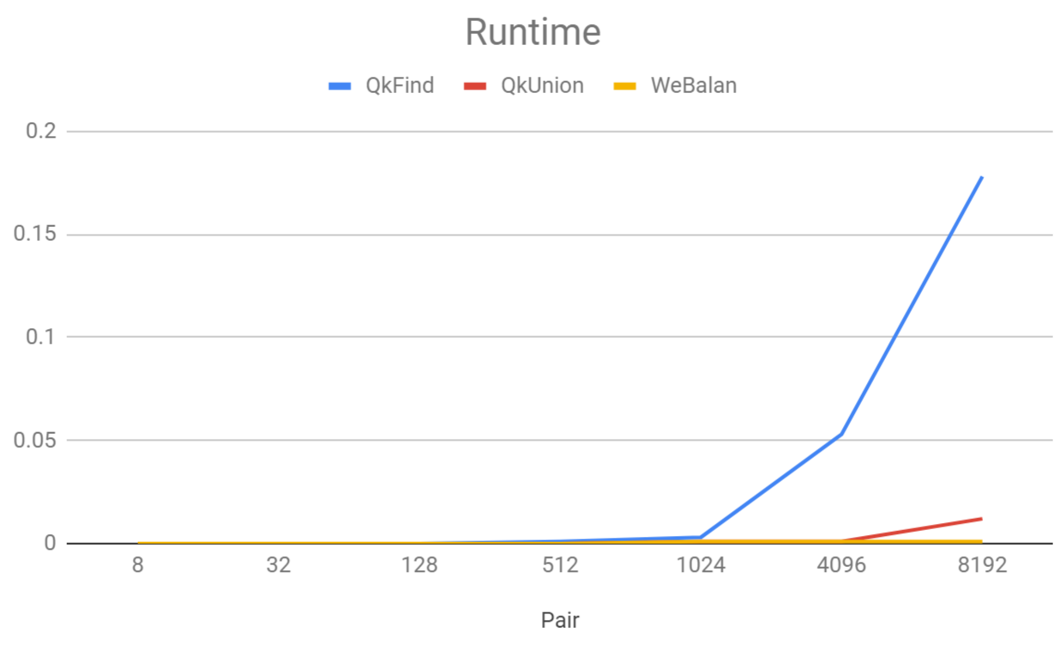
Int	8	32	128	512	1024	4096	4192	8192
Naive	0	0.001	0.002	0.065	0.538	24.459	25.813	191.426
Sophisticated	0	0	0	0.007	0.021	0.349	0.373	1.554



The run time of the naïve algorithm increases much faster than that of the sophisticated one. For the naïve algorithm, there is a very obvious increment of run time when the numbers of data get more than 1000, the run time is easy to perceive, while the run time of the sophisticated algorithm is still less than 1 second.
There is actually no different between these two algorithms when the number of data is less than 1000, the naïve algorithm even has an advantage compares to the other one that it can avoid the problem of repeated data. But the sophisticated algorithm works much faster when the number of data comes over 1000.

Q2
Implementation in C++ is in the attachment FindUnion.cpp file
Result

Pair	QkFind	QkUnion	WeBalan
8	0	0	0
32	0	0	0
128	0	0	0
512	0.001	0	0
1024	0.003	0.001	0.001
4096	0.053	0.001	0.001
8192	0.178	0.012	0.001



For the Find Union program, all the three algorithms run fast when the number of data pairs is below 1000. With the increasing of the pair numbers, run time of the Quick Find increase much faster than the others. The Quick Union with Weigh Balancing is the best one.

Q3

Nc for Q1 3 sum problem

In the naïve algorithm, the Nc should be 3, because this algorithm is basically made up by 3 loops to traverse all the pairs, and every data read at one time shouldn't be repeated. So for example, the first pair will be a[0], a[1] and a[2]. So there is at least three data in this array, so the Nc should be 3.

In the Sophisticated algorithm, the Nc may be 5. This algorithm is basically made up by 2 loops and a binary search. The first 2 loops are the same and they need at least two data. Then for the binary search, it needs "low", "mid" and "high", which means 3 data. So Nc should be 2+3=5.

Nc for Q2 Find Union problem

In the quick find algorithm, the Nc could be 2, because both the union and the find operation

needs 2 data to realize that.
In the quick union algorithm, the Nc could be 4. Based on the quick find algorithm, each data needs at least one root to reflect the Big Oh complexity.
In the quick union with weigh balancing algorithm, the Nc could be 5. Based on the quick union algorithm, the roots of each data should be different so that the weigh balancing works. So let's say one data has the at least one root, then the other one data should have at least two roots.

Q4

Farthest Pair implementation is in the attachment Farthest Pair.cpp file

Q5

Faster-est-ist 3-sum implementation is in the attachment Faster-est-ist 3-sum.cpp file