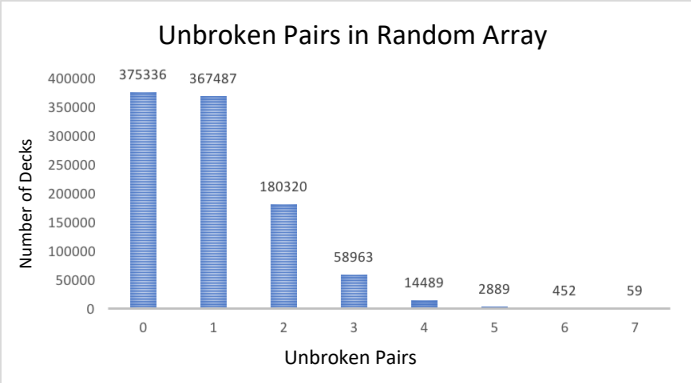
Overhand Shuffler

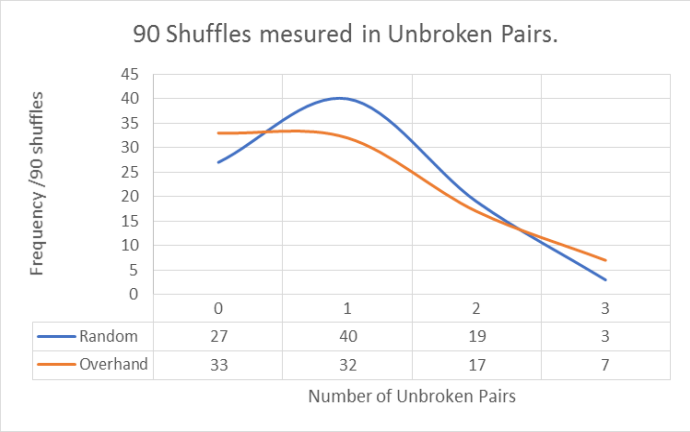
Efficiency of Overhand Shuffler.

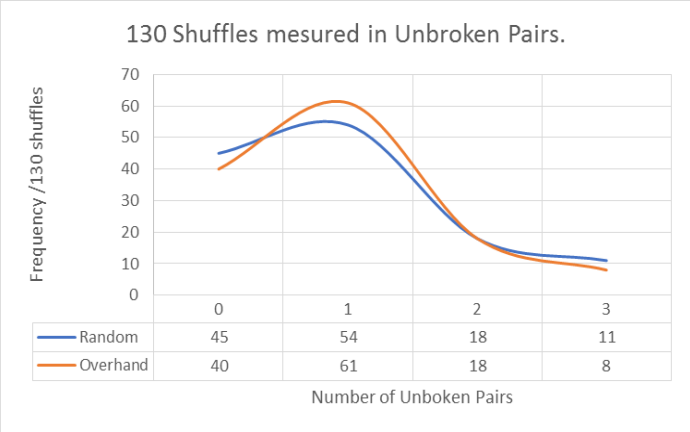
Disclaimer – We ran the test on this code through one million cycles, these are the results.



UNBROKEN PAIRS	SHUFFLES NEEDED
0	55
1	39
2	34

Using the top three most commonly found unbroken pairs in the randomly generated array from 0 to 50 (representing a deck of playing cards), we used our randomShuffle() and countShuffles() method to count how many shuffles were needed to achieve the same level of randomness in an array. What we found is that to achieve the same level of randomness, the deck of cards have to be shuffled a large number of times. In the case of a perfectly shuffled deck where there are no unbroken pairs, the number of shuffles needed was more than there are cards in the deck. Looking at this from a real life perspective where each overhand shuffle takes roughly a second, one minute spent shuffling a deck of cards. As you start to allow one or two unbroken pairs, then the number of shuffles needed to achieve to a reasonable amount.





Looking at the data from a different perspective, and comparing the number of unbroken pairs found in 90 to 130 instances of pseudo random deck of cards and 90 to 130 Overhand Shuffles. We saw that as you increase the number of Overhand Shuffles that were done, the frequency of unbroken pairs in Overhand Shuffler became closer those of the pseudo random deck. Implying that with enough Overhand Shuffles then the same level of randomness can be achieved. This also implies that it is not an effective method of making sure a deck of cards is perfectly random.