Trevor Yokoyama

Prof. De Saint Germain

CS 2420

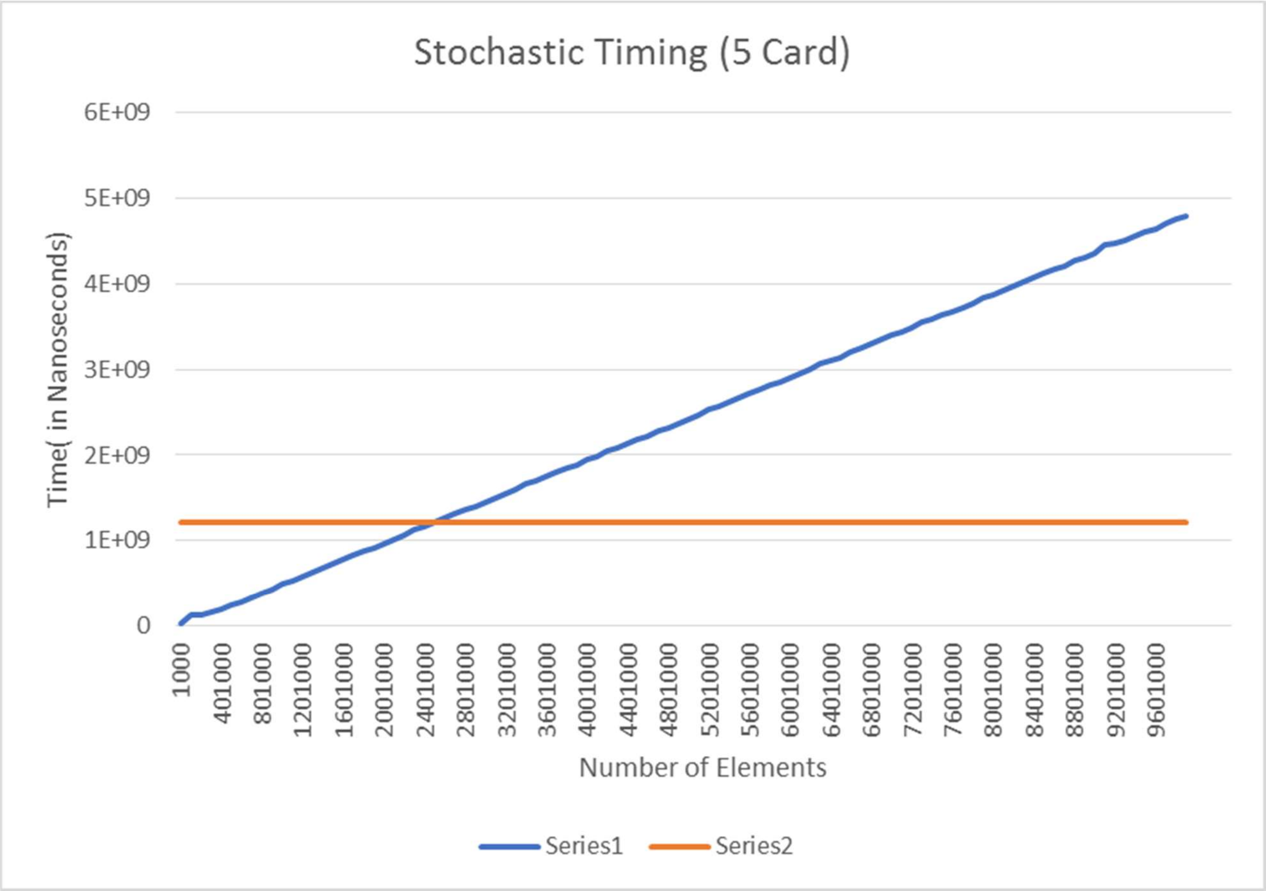
March 24, 2017

Assignment 8 Analysis

Overview

This project was centered on the analysis of 5 card and 7 card poker hands. The project used exhaustive and stochastic methods in order to create and rate different types of poker hands. The exhaustive method goes card by card and calculates all the possible combinations of a certain poker hand. The stochastic search creates random hands and then tests for the rank of each hand. Both of the methods are needed in order to truly understand the probability of certain poker hands. Furthermore, the random generation that was used to "randomly" create poker hands was java sort. However, we did create our own sort with a custom seed to see the impact that this had on the probability of certain poker hands.

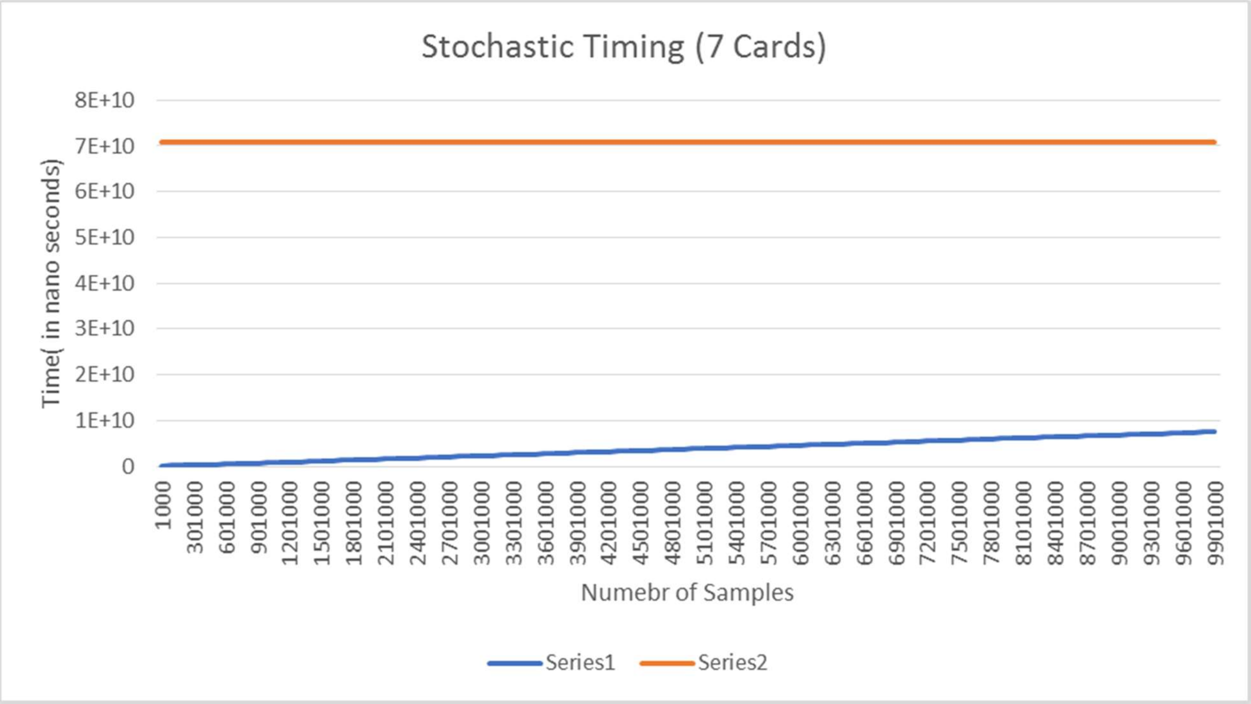
Graphs and Timing



Series 1 is the Stochastic and Series 2 is the Exhaustive

In this graph we can see that the exhaustive test is just a constant of time as the method has to always go through all the cards, thus adding more elements to the test does not change anything.

For the stochastic method, we can see that the series makes a linear line. As the number of elements increases, the time to test all of them also increases. I believe that this is because, the method has to run through more iterations as the number of elements increases, where the exhaustive method does not.

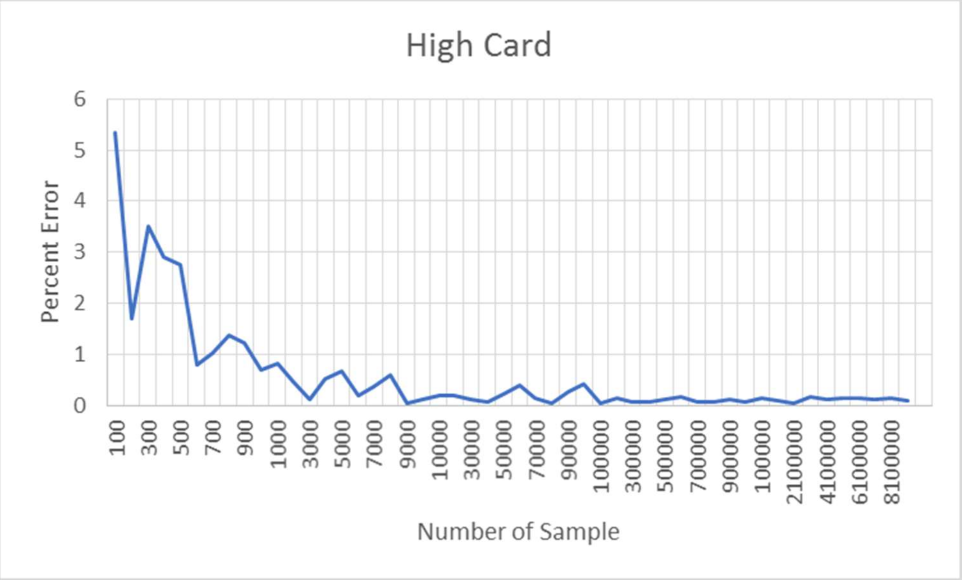


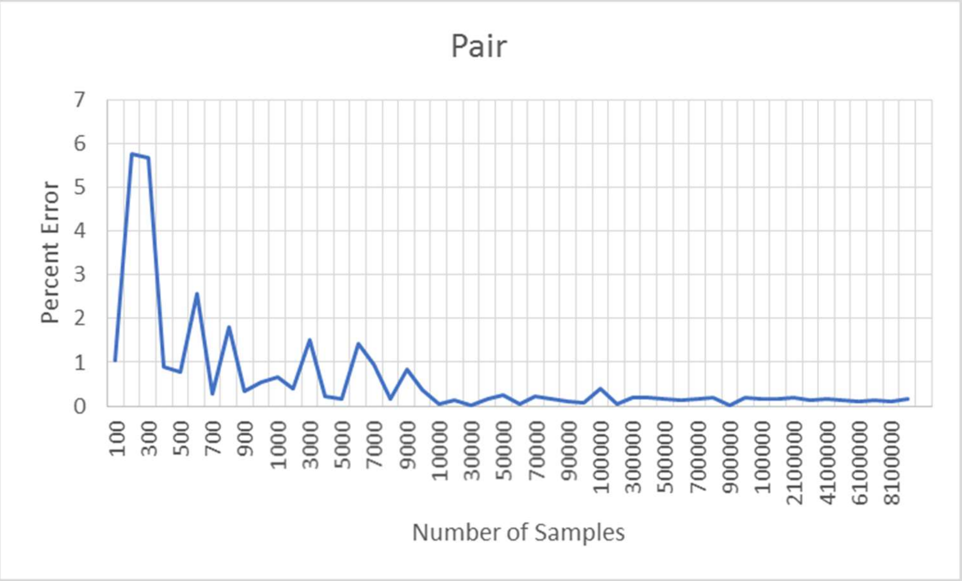
Series 1 is the Stochastic and Series 2 is the Exhaustive

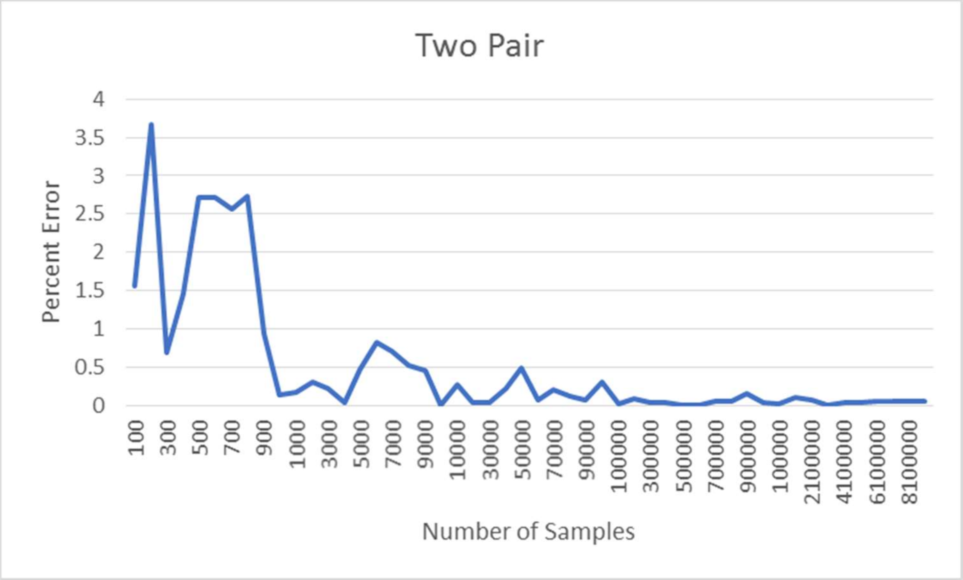
This graph shows both the stochastic search and the exhaustive search for 7 cards. Like the 5 card variation, we see that the exhaustive search is a constant line of time, where as the stochastic search is a positive linear line. However, the exhaustive search takes much longer to test everything as it now has to run through 7 cards instead of just 5 cards. This drastically increases the time in which the method takes to run. The stochastic method takes around the same time as the 5 card variation because the 2 added cards does not make an impact on the way in which the stochastic method tests variations.

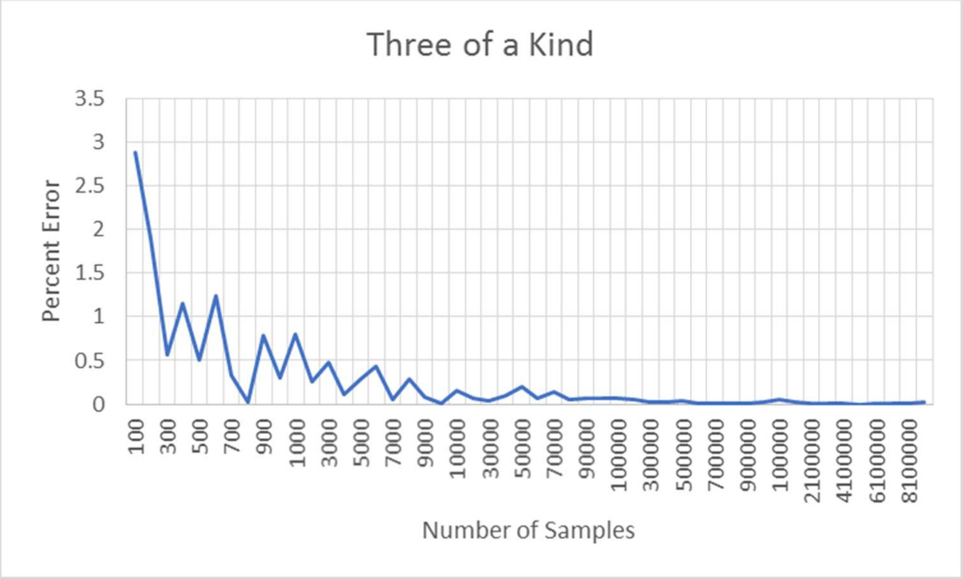
Because the stochastic sample is random, it can better reflect the odds that a poker player may receive certain odds, while it may not be the “correct” odds, it does give a sense to real life situations. Furthermore, the stochastic method works better for hands that contain fewer cards. For example, the stochastic method would work much more efficiently when computing a 9 card hand as opposed to the exhaustive method.

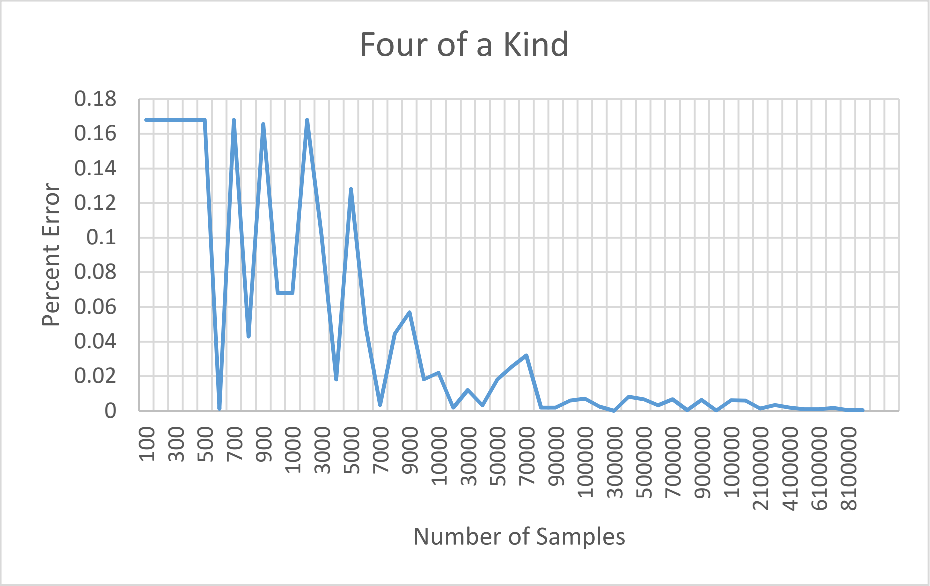
Graphs that show the difference between the stochastic and exhaustive methods for each of the different ranks

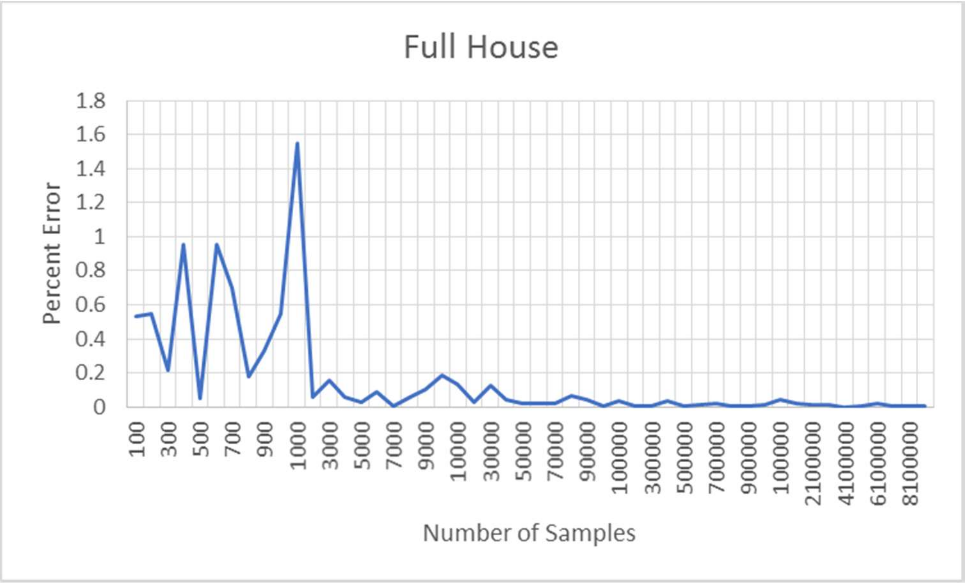


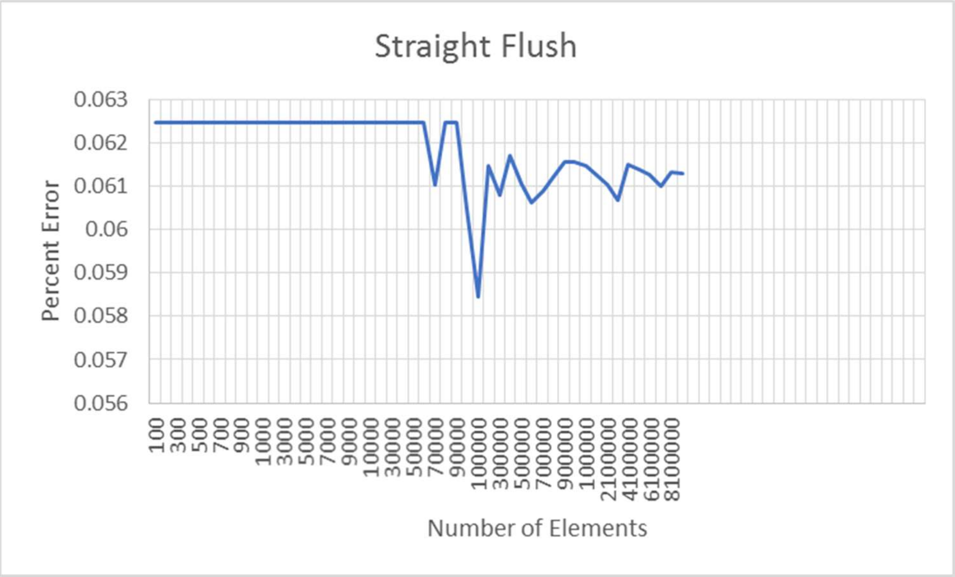


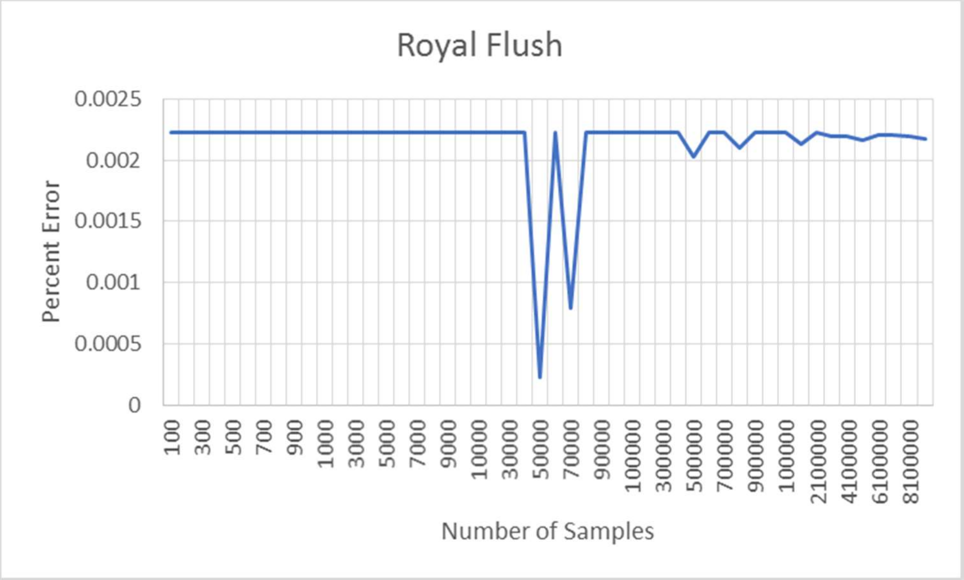

Two Pair
Percent Error
4
3.5
3
2.5
2
1.5
1
0.5
0
100
300
500
700
900
1000
3000
5000
7000
9000
10000
30000
50000
70000
90000
100000
300000
500000
700000
900000
1000000
2100000
4100000
6100000
8100000
Number of Samples


Three of a Kind
Percent Error
3.5
3
2.5
2
1.5
1
0.5
0
100
300
500
700
900
1000
3000
5000
7000
9000
10000
30000
50000
70000
90000
100000
300000
500000
700000
900000
1000000
2100000
4100000
6100000
8100000
Number of Samples


Four of a Kind
Percent Error
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
100
300
500
700
900
1000
3000
5000
7000
9000
10000
30000
50000
70000
90000
100000
300000
500000
700000
900000
100000
2100000
4100000
6100000
8100000
Number of Samples


Full House
Percent Error
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
100
300
500
700
900
1000
3000
5000
7000
9000
10000
30000
50000
70000
90000
100000
300000
500000
700000
900000
100000
2100000
4100000
6100000
8100000
Number of Samples





In looking at these graphs, we can see that as more samples are created, the difference between the stochastic and exhaustive samples becomes less and less. This is because more information and data will yield better results that are closer to the actual data. As the better ranks of the poker hands were reached, the percent error for the stochastic sample was less than the staring percent error for the lower ranks. This is because of the rarity of these hands, it is both difficult for the exhaustive and stochastic samples to achieve these ranks, leaving less of a margin of error.

Software Development Log

This project took around 8 hours in total. I found that the most difficult part of the problem was figuring out the correct syntax for the enums and how to correctly use them. But once I found out how to use them, I found the project to be much easier. I feel that I had adequate time for the project, however tests and papers in other classes prevented me from implementing the Texas hold em methods. The hardest method to write was the getRank method. The reason for this was that I tried to find a general solution for the rank. However, I found that it would be easier to just write out each of the rank cases. Writing tests also helped me to correct and tweak my ranking method as I kept having to go back to fix broken code.

Thought Problems

A good random number generator is very important for the stochastic experiment. Because the stochastic approach relies on truly "random" hands, the performance of the experiment will only go so far as the random number generator allows. The poor random generator was the worst as the cards picked were not random at all. The Better random generator's performance depended on the seed that was used. A better seed would yield better results. We used the code that we created in the lab, therefore, our generator and the java generator were very similar as the seeds were practically the same.