Programming Problem Set 2 Paper

Evan Tyler

For the first problem we had to calculate the percentage of certain poker hands. Usually in poker there are 52 cards and the dealer draws five random cards from the deck and places them face-up on the table. Then the dealer will shuffle the cards and then draw another five random cards. The aim of the game is to have the best hand possible. Having the best hand can come down to the five random cards selected by a dealer. One of the hands is two of a kind which means two of the cards in the hand have the same integer value. Three of kind means that if 3 of the cards in a deck have the same integer value. Two pairs mean that 2 cards have identical integers as well as another pair of cards have identical integers. A straight means that the order of the cards are in sequential order. A full house means that there are three of the same integer value cards and the other two cards also have the same integer. A flush means that the suite of the 5 cards being, hearts, diamonds, clubs, and spades all match so 5 cards with all hearts suites for example. So for the cards themselves

To make poker hands in java I have a card class that is a blueprint for creating cards with a constructor that requires each card to have a numeric value and a suite. These cards are created in my CardSim class which runs my entire poker hand simulation. 52 cards are created 13 of each suite. Which adds it to an arraylist which stores the deck of cards. Then once the 52 cards are generated. My Shuffle method takes a random card from the old deck arraylist and repeats until the cards are all shuffled up. Once the cards are shuffled the five random card method runs and picks five random cards from the shuffled deck and makes sure to remove each one so they aren’t picked again. Once the five random cards are stored in a new five random card array list the isStraight() method will run and check for one pair, two pairs, three of a kind, four of a kind and full house. This works since the loop compares the cards in the five random card arraylist to one another and counts comparisons. Depending on the amount of comparisons at the end tells the program which certain hand there is. Then it adds a counter to the hand it found. Next the isFlush method gets called which does the same exact thing but checks for a flush. Finally the Stats method gets called and the percentages of hands print out for display.

Next I created a program to smooth a graph and the function I chose to graph was the function y = mx + b. First I created my tester which first generates x values for my y = mx +b function which I started at -3 and went to positive 3. Next I called the genyvalues function in which it passed the xvalues to the function y = mx +b. From there the function outputs values of y which I stored into an ArrayList full of y values. A separate ArrayList stored my x values. Next I called the writetographCSV function which uses Printwriter to write to a file. In this case I looped through my x and y values arraylist and wrote to the file. I made sure that each value was separated by a comma and that the x and y values were on separate lines. Then I opened the csv file in excel and then graphed the function.

Now that our original function was graphed it’s time now to salt the data. Next I called the saltthegraph function which uses File reader and buffered reader to read the original graphs csv file it then reads the file line by line and stops when a line is null. The program then loops through the y values and salts them adding to the first and then subtracting to the second number in each line until it gets to the end. Next it uses a printwrite again to write to a csv the x values of the original function and the now salted y values of the original function. Then the csv is imported into excel and made into a salted graph which adds a lot of noise to the y axis on the graph.

Next I created the smoothing function which gets called next and it takes the salted y value arraylist. Then it parses through the arryalist and takes the average of the first 3 values and then the next 3…. Etc until it reaches the index - 3 from the list. Then these new smoothed y values are put into a new arraylist full of smoothed y values. Next a print writer is used again and it takes the regular x values and the now smoothed y values and outputs them into a new csv. The csv is now moved into excel where the graph is now a smoothed out version of the salted graph. Taking the moving average of three data points helps to show a trend in this case that trend is to the moon.

For the Geometric distribution I made an input on the tester in which it accepts the success rate and the number a success is on. Then I plugged those values into the doGeometric function which plugs the values into the equation and spits out the percentage of hitting your success on the specific number of trials. This distribution is great for when there is a trial and you want to see what the probability of success is on a certain number in that trial.

For the Poisson distribution I also have a tester where a user can change the values and test the Poisson distribution's outputs. Then those values get passed through and the subsequent method is used to calculate the Poisson distribution using the formula. The Poisson distribution is best used if you need to calculate the probability of n events happening during an interval of time. For example how people can cross a road in an hour if the average is 1000 people per hour.

For the Hypergeometric distribution again I built a tester that accepts values and passes them through. These values get placed into the formula which calculates the number of times a particular event occurs in a certain number of trials. The difference between the Hypergeometric and the Binomial distribution has independent trials while the Hypergeometric trials are without replacement. They can be used for example with sampling problems like picking a defective part from a box for example.

For the Binomial distribution again a tester was made that I can pass values to. The values are put into the formula which can only be used when there are two outcomes, independent trials, number of trials are fixed, and prob of success stays the same for all trials. For example testing a new drug and the drug either works or it doesn’t.