The range [1, n] is a discrete uniform distribution, so we firstly set out to recreate a uniform distribution of unique random numbers

**Creating Uniform Random Number Distributions & Verifying Uniformity**

We used a MCG (Multiplicative Congruential Generator) to generate (pseudo) random numbers, with

a = 16807,

m = 2^31 – 1, and

Seed = (Java-generated random number in the range [1, 57689])

We chose the range [1, 57689], because the seed cannot equal 0 ( so the lower bound was 1), and we chose 57689 (the upper bound) randomly.

Since the period (without repetition) is m - 1 (~ 2.15 \* 10^9), and we generated distributions with a maximum length of (1 \* 10^3).

(~2.15 \* 10^9) > (1 \* 10^3) ==

(m-1) > (n)

The period was larger than the sample size. Thus, we know that the numbers were distributed uniformly, and every number in the set was unique.

**Chi-Square**

Moreover, the results of our chi-square tests, were less than 79.96 for the corresponding value on the chi-square distribution table for n-1 degrees of freedom and α = 0.05

n = number of elements in distribution

d.f = 99, α = 0.05, x^2 = ?????????

d.f = 999, α = 0.05, x^2 = ?????????

Therefore, due to the period of the MCG being larger than the sample size, and scores that validate against the Chi-Square test, we believe that our random number distributions were uniform

**Strategy 1:**

Our first strategy is to conditionally select our number past a ‘point of desperation’ (POD).

In this strategy, we define a ‘point of desperation’ (POD). As we move through the uniform distribution of unique numbers, we record, but don’t select, the highest number we encounter before the POD – our ‘record highest’. Then, once we pass this POD, we select the first number that is higher than our ‘record highest’

So, if our POD is n/4, we record, but don’t select, the highest number from [1, n/4]. Then, we select the first number from [(n/4) +1, n] that is higher than our ‘record highest’.

**Strategy 2:**

Our second strategy is to randomly choose a ‘guess highest’, and if the current number in the sequence is higher than the ‘guess highest’, select that number.

**Strategy 3:**

Our third strategy is inspired by the counting-cards strategy in Blackjack.

In this strategy, we determine find the range which the highest numbers are found, through Monte Carlo simulations, which will give us a discrete random variable to develop a flexible strategy.

To find this range, we recorded the highest number in each of 1,000, 10,000 and 100,000 distributions/simulations (with n = 10, 100, 1000 for all), and plotted them into discrete random variables. In these distributions of highest numbers, we found the smallest and largest number, the expected value and the variance.

Smallest overall number that was highest in its distribution = Highest Floor

Largest overall number that was highest in its distribution = Highest Ceiling

Highest range = Highest Floor - Highest Ceiling

A screenshot of a white sheet

Description automatically generated

* One observation is that as you increase n by 10^x, the variance in highest numbers obtained decreases by 10^x
* This satisfies the empirical law of large numbers
* Therefore, it is in the player's best interest to play with a **larger** n (assuming that they are patient enough) in this strategy

The larger our sample size, the narrower our guess range of numbers, meaning we can more accurately guess the highest number.

With this information, we advise to play games with the highest n, to further your chances of winning

Now, we will run a modified version of strategy 1. We will define a ‘point of desperation’ (POD). A POD is, if we pass this point, and have not achieved a desirable outcome so far, we will select the next number in the range [highest\_floor, highest\_ceiling], that is higher than every number we’ve already come across so far.

We will test this strategy with PODs n/4, 2n/4, 3n/4 (ie, Quartile 1, Median and Quartile 3)

A desirable outcome is related to the POD. If the POD is n/4, the range for a desirable outcome increases by 4% (of the total distribution of highest numbers) with every increment towards n.

So, if POD = n/4, and we have taken 2 steps, we will select any number that appears in the range, [ (highest\_ceiling – (highest\_range\*(2\*(4/n)) ) ), highest\_ceiling]

k = number of steps taken

c = (inverted POD) (ie, if POD = n/2, c = (2/n),

if POD = 2n/4, c = (4/2n),

if POD = 3n/4, c = (4/3n)

Range of desirable outcome = [ (highest\_ceiling – (highest\_range\*(kc) ) ), highest\_ceiling]

This is based on the strategy of counting cards in Blackjack, where if you get a high number and not many cards have been dealt, you stick with that number. However, if the dealer has dealt many cards and your hand is still bad, you’re more likely to take a risk. Similarly, we are more likely to take a risk with a relatively low high number.