Question 3:

Introduction:

In this question, the range [1, n] is a discrete uniform distribution, so we firstly set out to recreate a uniform distribution of unique random numbers.

Afterwards, we developed 3 strategies to determine the highest number among *n* positive numbers, to win €1000. Strategy 1 is designed for someone who has no data on previous outcomes of the game. Strategy 2 demonstrates lack of value in random selection. Strategy 3 is intended for a player who has data on the past outcomes of the game's random number generator.

Creating Uniform Random Number Distributions

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 the upper bound randomly. That is, we had a different upper limit for every seed in every simulation.

Code: 'in highest range', lines 117->131

Verifying Uniformity

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.

Therefore, due to the period of the MCG being larger than the sample size, we know that our random number distributions were uniform.

Strategy 1 'Point Of Desperation':

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'.

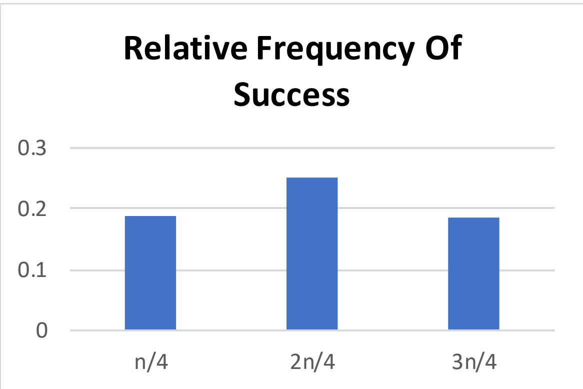
We will test this with distributions of n = 10, 100, 1000 (with 1 000, 10 000, 100 000 simulations each) at PODs n/4, 2n/4, 3n/4

Code: ‘funcs’, lines 5->64

These are the raw results;

		Points Of Desperation - Relative Frequencies Of Success		
n	Number Of Simulations	n/4	2n/4	3n/4
10	1000	0.223	0.283	0.153
10	10000	0.144	0.234	0.176
10	100000	0.176	0.246	0.165
100	1000	0.23	0.261	0.191
100	10000	0.237	0.242	0.174
100	100000	0.241	0.245	0.186
1000	1000	0.172	0.266	0.186
1000	10000	0.186	0.253	0.193
1000	100000	0.188	0.25	0.186

Bar-chart:



We can see that fn(A) (relative frequency of success) is highest for a POD of n/2 (with distributions of n = 1000)

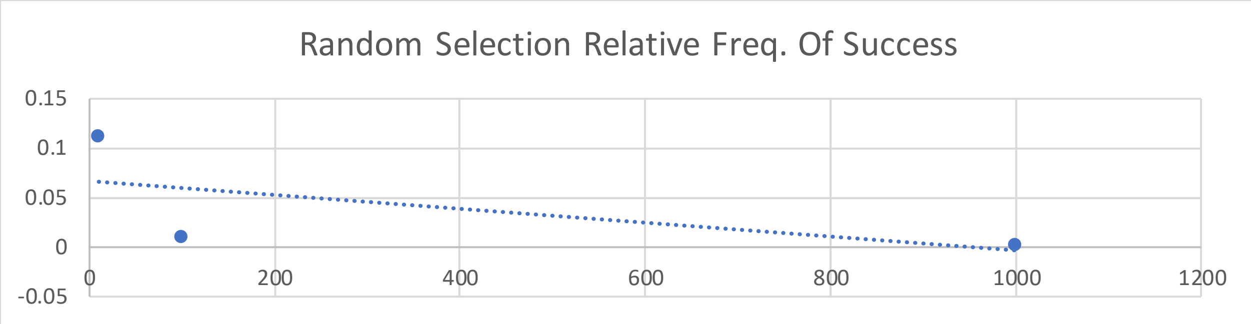
Strategy 1 ‘Point Of Desperation’ Conclusion

Therefore, the player should choose a POD of n/2 (the median) to maximise their chances of winning with this strategy.

Strategy 2 ‘Random Selection’:

Our second strategy is to choose a random number/piece of paper. While it is a rudimentary strategy, with a probability of success of 1/n, we feel it will prove the advantages of other strategies. Code: ‘method 2’, lines 2->14

And these are the relative frequencies of success with 100,000 trials for n = 10, 100, 1000, in that order



We can see that there is a negative linear relationship with the size of n and the relative frequency of winning – as n increases, the player's chances of winning decrease.

Strategy 2 'Random Selection' Conclusion

We advise to pursue strategy 1, '*Point of Desperation'* over strategy 2, '*Random Selection'*, when n > 4.

Strategy 3 'Highest Range':

Our third strategy is inspired by the counting-cards strategy in Blackjack. It is intended for real-life scenarios, where random number generators (RNGs) are deterministic, and thus have deterministic upper bounds.

In this strategy, we determine the range which the highest numbers are found, prove that RNGs have a determined range of upper bounds, 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) 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
Code: 'in highest range', lines 8->87

n	Number of Simulations	Highest Floor	Highest Ceiling	Expected Value	Variance	Highest Range
10	1000	977085980	2.15E+09	1.9E+09	4.77E+16	1.17E+09
10	10000	685638244	2.15E+09	1.91E+09	4.63E+16	1.46E+09
10	100000	685638244	2.15E+09	1.91E+09	4.51E+16	1.46E+09
100	1000	1987302196	2.15E+09	2.13E+09	4.65E+14	1.6E+08
100	10000	1953843470	2.15E+09	2.13E+09	4.62E+14	1.94E+08
100	100000	1938750646	2.15E+09	2.13E+09	4.59E+14	2.09E+08
1000	1000	2131159213	2.15E+09	2.15E+09	4.27E+12	16321964
1000	10000	2125702795	2.15E+09	2.15E+09	4.43E+12	21780776
1000	100000	2124629467	2.15E+09	2.15E+09	4.67E+12	22854141

- We observe 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 interests to play with an exponentially **larger** n (assuming that they are patient enough) in this strategy, so we will use n = 1000 from now on

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

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 larger than our ‘record highest’.

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/n% (of the total distribution of highest numbers) with every increment towards n.

So, if POD = n/4, n = 100, and we have taken 2 steps, we will select any number that appears in the range,

[(highest_ceiling – (highest_range*(2*(4/100)))), 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]
Code: ‘in highest range’, lines 158->211

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.

Finding the highest ceilings, floors and range, with different parameters to the MCG than in the first picture:

n	Number of Simulations	Highest Floor	Highest Ceiling	Highest Range
10	1000	39977441	134216516	94239075
				0
				0
				0
				0
				0
				0
				0
1000	100000	132840659	134217720	1377061

Here, you can see that highest_floor and highest_ceiling have large differences in magnitude compared to the earlier first picture, despite them both being RNGs. This proves that the MCG, while providing a uniform distribution, has an upper limit, regardless of the seed.

Since it is not a truly random distribution, our strategy to determine the lower and upper limits of the game’s random number generator is valid. Here are the results, when we know the range of upper limits of the RNG:

	Points Of Desperation, n = 1000			
	n/4	**2n/4**	**3n/4**	**9n/10**
Relative Frequencies Of Success (100,000 simulations)	0.191	0.265	0.334	0.358

We can see that Strategy 3 is more successful than the previous two strategies.

Strategy 3 'Highest Range' Conclusion

It is in the players' best interests to play with a larger *n*, and a POD of 9*n*/10. However, it requires a lot of prior data on the previous outcomes of the random number generator, and hinges on the probability that the game owners do not change their RNG's algorithm whatsoever. **If these prerequisites aren't met, do not pursue this strategy**.

Conclusion:

Strategy 2, '*Random Selection*', scales very poorly, so do not choose Strategy 2 (unless n < 5,). This has a probability of success of 1/n –> unacceptable.

Strategy 1, '*Point Of Desperation*', with a POD of n/2, is the best strategy for a player who has no prior data on the results. This has a probability of success of 0.25 at best (making it better than Strategy 2 for n > 4).

Strategy 3, '*Highest Range'* has the highest probability of success of any strategy, however, it requires that the game owners never change their RNG, and that the player has the results of over 1000 games of size 1000. If these prerequisites aren't met, the strategy falls apart.

We conclude that the theoretically best strategy is Strategy 1, '*Point Of Desperation*', however, the practically best strategy is Strategy 3, '*Highest Range*'

Code: https://github.com/bobAnthonyVarley/Applied-Probability-1-Assignment