MATH4009 Project 1

September 27, 2021

This project 1 is a group work. You need to work in a group of 2 or 3. Submit your reports along with Python code through Moodle dropbox.

1. In this first part, we will investigate how good Python random number generator is. To do this, perform the following steps:

   (a) Generate 10,000 random numbers between -1000 and 1000.

   (b) Count how many times each number appeared from the first part.

   (c) Repeat first two parts for 100 times.

   (d) Investigate the distribution of random numbers you got and write a short report on whether or not the list really looks random.

   There are few things I want you to think about and answer in your report.

   i Why do we might want to repeat part a and b 100 times rather than performing part a 1,000,000 times in the first place?

   ii According to your answer to the question above, is it better to generate fewer random numbers and repeat it more times? What are ups and downs of doing so?

   iii If the random number generator gave you something that looks quite random in this simulation, can we conclude it is a safe random number generator? Why or why not?

2. In the second part, we will investigate the following question:
   "If you are trying to guess a number between 1 and 1000, is it better to start from 1 and increase by 1 each time or is it better to make a random guess each time?"
   To simulate this, do the followings:

   (a) Let Python choose a random number between 1 and 1000 by using its random number generator.

   (b) Start from 1 and see how many steps it takes to make a correct guess. (You don't need to write a separate code for this part as it is obvious)

   (c) Make a random guess using Python's random number generator and see how many steps it takes to make a correct guess.

(d) Repeat above 3 steps for 1000 many tries. Write a short report with results you have.

There are few things that you need to be careful about and few thing I want you to answer in the report.

i The same random number should be used for part *b* and part *c*. Why is this crucial?

ii Do you think one method is better than the other? Is this what you expected?

iii Even if there doesn't seems to be any difference, one method might be preferred over the other. Can you think of why?

3. In the third part, we will investigate what's known as *100 prisoner problem*. This is a famous computer science problem that raises few good points. The following is a version by *Phillipe Flajolet* and *Robert Sedgewick*.
The director of a prison offers 100 death row prisoners, who are numbered from 1 to 100, a last chance. A room contains a cupboard with 100 drawers. The director randomly puts one prisoner's number in each closed drawer. The prisoners enter the room, one after another. Each prisoner may open and look into 50 drawers in any order. The drawers are closed again afterwards. If, during this search, every prisoner finds his number in one of the drawers, all prisoners are pardoned. If just one prisoner does not find his number, all prisoners die. Before the first prisoner enters the room, the prisoners may discuss strategy, but may not communicate once the first prisoner enters to look in the drawers. What is the prisoners' best strategy?
If each prisoner makes a random guess, then the probability that any prisoner finds his or her number is 50%. Thus, the probability that all 100 prisoners find their numbers is $(\frac{1}{2})^{100} \approx 0.0000000000000000000000000000008$ which is nearly impossible. However, the following strategy can hugely improve the chance of success.

I Each prisoner first opens the drawer with his own number.

II If this drawer contains his number, he is done and was successful.

III Otherwise, the drawer contains the number of another prisoner and he next opens the drawer with this number.

IV The prisoner repeats steps II and III until he finds his own number within 50 tries.

You might wonder why this makes any difference and there's a really good mathematical explanation(combinatorial one) but I won't explain it here. You may read Wikipedia page yourself. We will run simulations to test this.

(a) Write a code that simulates a random guess case.

(b) Write a code that simulates the method described above.

(c) Run the simulation 1 million times and calculate the success rate. Write a short report.

Here are things I want you think about in this report.

i You need to make sure you perform step *a* and step *b* under the same setting(same randomly distributed drawers). Why is this crucial?

ii What's the success rate for each method? Can you tell one is better than the other? Do you think there's any possibility that this difference is a mere coincidence?

There is an important lesson from this simulation. Even if you think you can't do better than making a 50/50 guess, there might be a way to greatly improve the chance of success.

4. In this last part, you will break a simple substitution cipher using Python. The cipher text is provided in Moodle. I used numbers 0 to 25 to represent letters a to z respectively in plain text (hint: this is a very famous story).