An Exploration of Pseudorandomness

Liu Chengkai

518030910425

1 Introduction

As computer science students, we often hear a lot of "pseudorandom" in our studies and daily lives. It is often used to describe the random mechanisms in various computer programs and software. And it is also applied in some video games as well.

The terminology "pseudorandom" has been misused to now and has developed a variety of common mearnings that deviate from its original meaning. Besides, the concept of randomness itself is complex as well as has multiple mearnings, some people may get more confused about the clarifying the different meanings and usages of randomness and pseudorandomness.

So I find pseudorandomness is an interesting topic for me to explore and analyze. In this paper, an attempt is made to investigate and analyze the concept of "pseudorandom" from the perspective of probability theory with different meanings in various contexts in order to eliminate the misunderstandings. And with some examples, I try to explore and state the design purposes of various so-called "pseudorandom" mechanisms.

2 Randomness and Stochasticity

When we refer to "randomness" or "stochasticity" in probability theory, we may be pointing to one of two conecpts: stochastic process and random variable. A stochastic process is usually thought of as a set of random varables that depend on parameters, or a function that maps from parameters to random varables. And a random sequence is also a sequence of random variables obtained by taking the values of the parameters in a random process according to certain laws. These concepts are clear and unambiguous in the definition of probability theory. But when in the ream of practical application and philosophy of science specifically, their extensions are quite different: the contradiction is mainly directed at the word "random", which needs no much definition in probability theory but is crucial in real life applications.

Stochasticity in a stochastic process is usually embodied in the unpredictability of the process. Stochastic processes in nature, traced to their origin, are usually ensured by one of two mechanisms: one is the uncertainty principle of quantum mechanics, that is, the natural process cannot be accurately measured at a very small microscale; the other is the chaos theory at the mathematical level, that is, a small initial value change will cause a huge change in the output value of the system, thereby resulting in unpredictable results. Some points of view suggest that the existence of free will also lears to stochastic processes. For example, A cannot predict the behavior of B. For A, B is a random factor in this stochastic process.

Randomness in a random sequence, on the other hand, is often reflected in the irregularity, or incompressibility, of the sequence. Satisfying the statistical laws such as the large-sample nature of the corresponding distribution of random variables is the most basic requirement of a random sequence. The incompressibility here emphasizes the randomness of the order in which the individual samples of a random variable are arranged in the sequence. For example, if a sequence of random variables has circular nodes of length 10^{100}, such a random sequence would be compressible(there is an established pattern of cycling every certain distance), or not strictly speaking, "sufficiently" random; although this sequence does satisfy the statistical rule for samples of random variables. For quantitative analysis of randomness in this sense, Kolmogorov Complexity is often used as its measure.

In the practical application of stochasticity, these two seemingly distinct concepts have caused conflict and confusion. When we evaluate whether a stochastic process is random or not, we should expect the process to be unpredictable, but in most cases we do not have aceess to the mechanisms of this stochastic process, but can only get the random sequence output as the result. The process seems to be a black box for us. It is true that the two are causal inevitable and unified in a certain sense. Strictly speaking, unpredictable random processes must produce irregular random sequences, and the latter can only be generated by the former. But in practical applications, because it is difficult to research the processm, we get used to inferring whether the random sequence output has a certain rule to reverse the randomness of the stochastic process. Thus, the evaluation of randomness is fixed on such a system where the result determines everything.

3 Pseudorandomness

Now we would like to talk about pseudorandomness.

The term "pseudorandom" can be understood as the use of a fully predictable non-stochastic process, which generates highly irregular raondom sequences - approximating those sequences generated by real stochastic processes. Its original source should be around the advent of commercial computers. At that time, computer did not originally have a pseudorandom function, but used the sampling of air noise to generate random numbers. This is a truly stochastic process. However, such a process requires interaction between computer hardware and the surrounding environment. And the efficiency of generating random numbers are too low. Thus, "pseudorandom" came into being. Pseudorandom on the computer usually refers to the use of irregular but not random variables such as system time as seeds. After the algorithm iterates, a set of unrecognizable and irregular results are obtained, and treat them as random sequences. In a sense, this is also a relection of randomness caused by chaotic dynamics, but the artificially designed algorithm is far less complicated than nature. Therefore, such a random sequence generation mechanism is what we call "pseudorandom". The phrase "the program can only generate pseudorandom numbers", which is often mentioned on the Internet, applies only to this meaning of pseudorandom.

In contrast, the term "pseudorandom", in other contexts, is often used to mean "pseudorandom sequence". The difference between the two is obvious: pseudorandom seeks to generate random sequences that are close to natural stochastic processes, while pseudorandom sequences are the opposite. It attempts to reduce the randomness of the sequence generated by pseudorandom, changing the original probability distribution of the random variables and correcting it to be compressible sequence. There are some examples below.

4 Examples

I would like to show some specific examples of pseudorandomness in several classic video games.

4.1 Shuffle Algorithm in *Final Fantasy*

In the first game of the *Final Fantasy* series, the ancestor of Japanese Role Play Games(shortened RPG), the map encounter mechanism uses the so-called "shuffle algorithm". Due to the extremly limited read-only memory(shortened ROM) capacity of the host, the programmers of *Final Fantasy* skillfully uses a fixed sequence of 1-256 to complete the random number generation required for the game. Every time the player takes a step in the game, this sequence will automatically read the next item, and then compare the number of reads with the pre-set enemy encounter rate parameter at this position to determine wheter an enemy is encountered.

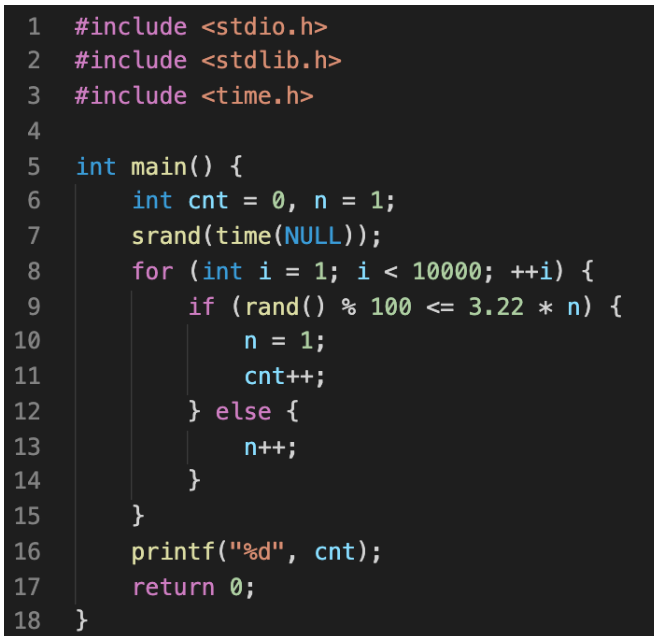
If the outcome is not corrected, the problem of this algorithm is very obvious: the entire game process is fixed, and players at different times and different locations can reproduce the exact same "random" encounter process by saving and reading files, thereby making the pseudorandom mechanism lose its purpose. So the programmers made corrections during the battle: from the beginning to the end of each battle, the sequence was read to the next item every few frams. Considering that the time required for each battle generally not noticed by the player, the result of this correction is that the first enemy encounter after each file reading is always the same, but the progress of reading the sequence after a battle is almost unpredictable and replicable, equivalent to taking a random sample again.

This pseudorandom mechanism is quite simple and lightweight. It has to save ROM because of the extremely limited ROM in that era. It does not even use a pseudorandom generator. Just by simply introducing a strong random factor of “battle time” in the player’s game into a fixed sequence, the problem of random enemy encounters is solved.

4.2 Probability Correction Mechanism in *Warcraft III*

There is a special probability mechanism in *Warcraft iii.* Take the critical strike of the blademaster in the game as an example. Although the skill desciption is “There is a 15% chance that the blademaster can cause more damage when attacking”. In fact, each attack does not obey 15% triggers a critical strike, 85% does not trigger a two-point distribution. The reality is that the game has a built-in counter with an initial value of 1. Every time an attack without a critical strike will increment the count by 1, and every time a critical strike is triggered, the count is reset to 0; the probability of each attack triggering a critical strike is nC = 3.22% multiplied by counting.

Pushing backwards from the apparent probaility to derive the value of C is a more complex process, so here we start with C = 3.22% and simulate 10,000 attacks to verify the final large sample probability of 15%.


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    int cnt = 0, n = 1;
    srand(time(NULL));
    for (int i = 1; i < 10000; ++i) {
        if (rand() % 100 <= 3.22 * n) {
            n = 1;
            cnt++;
        } else {
            n++;
        }
    }
    printf("%d", cnt);
    return 0;
}
```

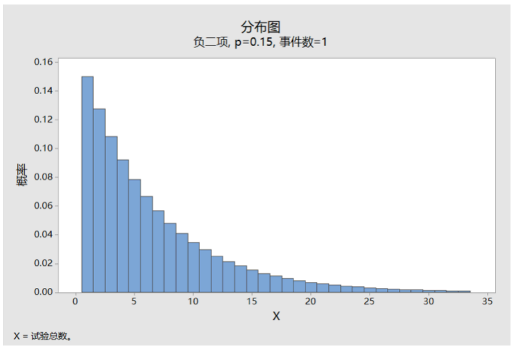
Figure 1: the code of the simulator

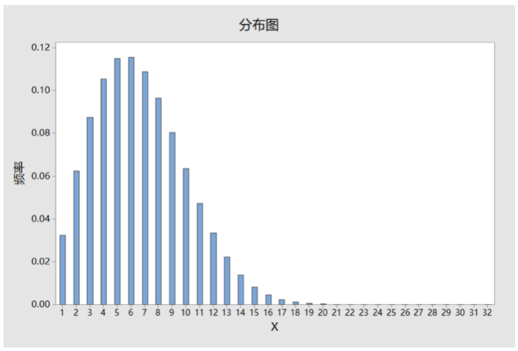
The program was run 30 times and the number of critical strike outputted is shown in the following table.

1498	1491	1466	1510	1522	1499	1479	1531	1512	1493
1482	1537	1496	1518	1491	1532	1471	1495	1500	1503
1494	1500	1530	1492	1517	1495	1496	1514	1507	1482

Considering the original hypothesis "expected value $\mu = 1500$. The distribution of the overall obedience is unknown, and the central limit theorem is used to test the large sample. When the original hypothesis is valid and n tends to infinity, the statistics $\frac{\sqrt{n}(\bar{X}-\mu)}{S}$ tends to the standard normal distribution. By calculation, we get $\frac{\sqrt{n}(\bar{X}-\mu)}{S} = 0.54$. Therefore, the P value of the two-sided hypothesis test is $P(\frac{\sqrt{n}(\bar{X}-\mu)}{S} \geq 0.54) = 0.59$, and the original hypothesis cannot be rejected at the confidence level $\alpha = 0.05$.

A more detailed analysis of this correction mechanism is also possible. Set the number of attacks required for each critical strike as a random variable X(which actually takes every two 1s' interval in a binary sequence of 0, 1 as a random variable to study). Then X obeys a special discrete distribution, taking values in the range 0 to 32. The distribution is as follows.





1	2	3	4	5
0.0322	0.06232632	0.087468757	0.105359034	0.114735988
6	7	8	9	10
0.114735988	0.108731542	0.096255374	0.080392488	0.063438606
11	12	13	14	15
0.047312512	0.033332095	0.022156955	0.013872981	0.008163258
16	17	18	19	20
0.004501765	0.002318859	0.001111252	0.000493124	0.000201506
21	22	23	24	25
$7.5323E-05$	$2.5551E-05$	$7.78934E-06$	$2.1084E-06$	$4.98989E-07$
26	27	28	29	30
$1.01195E-07$	$1.71082E-08$	$2.31708E-09$	$2.36144E-10$	$1.61718E-11$
31	32			
$5.682E-13$	$1.1E-15$			

Compare this distribution with the case where the probability of triggering each attack is the same, that is, the negative binomial distribution. And it is clear that this distribution reduces the overall of variance, allowing the number of attacks required per critical strike to be concentrated around 5 to 6.

This mechanism has a name called pseudo-random distribution(shortened PRD). On a practical level in the game, this design makes the player's game experience more stable. Interestingly, this mechanism is exploited by some experienced players to attack the enemy heroes or some other important targets if they fail to make a critical strike for multiple times, at which point the conditional probability of a critical strike will highly increase, allowing the critical strike to be made where it is needed.

4.3 Guaranteed Mechanism in Lottery and Draws

There are many draws in video games. Players can draw some cards or game equipments in lottery or draws.

For the sake of some unlucky players, a so-called "guaranteed mechanism" are added in lottery and draws. This mechanism is also widely used in games, such as the famous card game *Hearthstone*.

There is an example illustrating this mechanism. A player wants to draw a rare golded card in draws. Each time the golded card is not drawn by the player, a certain amount of points will be added to the counter. Each time the player get the golden card, the counterwill be reset to 0. Once the counter reaches a certain value, the result of the next draw will be the golden card. The number of draws corresponding to the certain value of the counter is slightly higher than the expected number of times requiredRto draw a golden card (subject to negative binomial distribtion), so that the player's overall draw experience is relatively close and there will be no player who has a terribly bad luck in the draws.

5 Conclusion

There is a famous saying that there is no true randomness in the programs. Randomness and pseudorandomness have different definitions.

So I talk about pseudorandomness, especially from the prospective of computer softwares and video games. It is a truth that seudorandomness is widely used in computer softwares and video games, which is worth studying.