Lab 6 Worksheet

Answer the following questions by modifying this text file. Add your answer below each question.

1. List one property of a RNG that CheckRandomNumberGenerator measures and give a reason why you think that this is an important property.

Answer:

- Max number of times a number was generated

When evaluating a random number generator, we do not want to repetitively get the same number, which defeats the purpose of randomization and makes the number sequence very predictable.

2. It takes a lot longer to generate ten million random numbers using JavasRandomNumberGenerator than using PoorRandomNumberGenerator. Why?

Answer:

Regardless of how many times we want to generate the random number, the time used by PoorRandomNumberGenerator is almost always negligible. The reason is that calling nextInt() from PoorGenerator always yield constant time O(c) since it returns 1, whereas every time calling nextInt() from JavaGenerator, there is an algorithm running to return the next integer number.

3. The "Number of Zeros after 10000 tries" refers to the number of integers between 0 and 10000 (exclusive) that were not generated in 10000 tries. For PoorRandomNumberGenerator this value is 9999. Why?

Answer:

Because PoorGenerator always returns 1, therefore the rest of (1000 – 1) numbers in this range never gets generated.

4. What does the number of integers between 0 and 10000 (exclusive) that were not generated in 10000 tries tell us about the RNG? Does JavasRandomNumberGenerator generate every integer between 0 and 10000 (exclusive) in 10000 tries?

Answer:

It tells us that with the algorithm used by the RNG, how many values we can never get. It is similar to the y-range of a function, and the numbers the algorithm can not generate is out of this y-range. With Java RNG, there is always 3680 numbers we can not get within 10000 tries.

5. The "Number of odd_even pairs" refers to the number of times an odd number is followed by an even number in the generated sequence of integers. What is a good value for this number?

Answer:

Roughly 2500 would be a good value for this number. This is because we do not want our number sequence to follow any pattern. If the entire sequence follows Odd – Even – Odd, then it presents a predictable pattern. Ideally, on each run of the program, this value will change follows an unpredictable rule. However, the sequence an RNG generates is only pseudorandom. Therefore, based

on the law of large number, we want the probability of getting each of the four possibilities be equal. 10000/4 = 2500 values would be good in this case.

6. CheckRandomNumberGenerator checks that given the same seed, the RNG produces the same sequence of integers. What is an advantage of this behavior? What is a disadvantage?
Answer:
Advantage: The number sequence is reproducible. If we want to test some programs with a sequence of random number, we want this sequence to be the same giving the same seed every time we test it.
Disadvantage: The number sequence is predictable and fixed for the same seed. It is not actually random.

7. For what properties did your RNG perform better than PoorRandomNumberGenerator?
Answer:

- Number of Zeros after 10000 tries

It can generate roughly 100 unique numbers after 10000 tries

- Number of odd_even pairs

The probabilities of odd- even, even-even, even-odd, odd-odd are roughly 25% each, which resembles Java RNG.

8. Why are RNGs that use computational algorithms considered "pseudorandom"?

Answer:

Because it is not actually randomly generated. Each RNG follows an algorithm by which each number in the sequence is dependent on the previous number. For example, if we pass in the same seed and constants to a RNG, we will get the same sequence of numbers. It only seems random but is not random.