# BST 234: Lab - 7

Divy Kangeyan

March 19, 2018

# Random numbers

- Efficiently generating independent random variables from the uniform distribution is important in permutation tests and many other useful instances.
- Random variable from uniform distributions can be transformed into other continuous random variables, i.e. Inverse Probability-Integral Transform
- Useful in simulations, transaction, cryptography etc.

# Properties of a good Pseudo Random Number Generators (PRNG)

- Uniformity
- Independence
- Passes all the diehard tests:
  - Birthday spacings test
  - Overlapping mutations test
  - Ranks of matrices test
  - Monkey test
  - Count the 1's test
  - Parking lot test
  - **Minimum distance test**
  - Random Sphere test
  - Squeeze test
  - **Overlapping Sums test**
  - Runs test
  - The craps test

# Properties of a good Pseudo Random Number Generators (PRNG) - Cont'd

- Replication (reason for generating pseudo random numbers instead of random numbers)
- Cycle length
- Speed
- Memory usage
- Parallel implementation
- cryptographically secure

# PRNG: Mid-square method

Algorithm:

- Start with a 4-digit seed ($z_0$)
- Square it to get 8 digit number, pad with zeros if necessary
- Take middle 4 digits from the 8-digit number generated
- Divide the 4-digit number by 10000 to generate Uniform RV

*Python Demonstration*

# Linear Congruential Generator

Produces a sequence of numbers between 0 and $m-1$

Algorithm:

- Start with the seed $z_0$
- $z_n = (az_{n-1} + c) \bmod m \;\; n = 1, 2, ...$
- To get Uniform RV $u_n = z_n/m$
- Choice of a, c and m are important

*Python Demonstration*

# Other Congruential Generator

- Multiplicative Congruential Generators ($z_n = az_{n-1}$)
  - Doesn't have full period
- Additive Congruential Generators ($z_n = z_{n-1} + z_{n-k}$)
  - Can have very long period upto $m^k$

# Mersenne Twister

- Current gold standard to generate PRN
- Invented by two Japanese scientists Makoto Matsumoto and Takuji Nishimura
- Passes all diehard tests
- Has very long period of $2^{19937} - 1$

*Python Demonstration for two diehard tests for PRN generated by Mersenne Twister*

# Truncation Error

- Introduced by algorithm via problem simplification, e.g. series truncation, iterative process truncation etc.
- For example several functions can be approximated by Taylor series expansion
- *Python demonstration*

# Relative Error

- Absolute error in basic arithmetic operation: $(\tilde{x} * \tilde{y}) - (x * y)$
  $* : +, -, x, /$
- Relative error in basic arithmetic operation: $\frac{(\tilde{x} * \tilde{y}) - (x * y)}{(x * y)}$ $* : +, -, x, /$
- *Python demonstration*