## I. Bernoulli Trials and the Binomial Distribution

A. Definition: A Bernoulli trial is an experiment with only two possible outcomes.

B. Examples:

1. When a coin, biased or true, is flipped this is a Bernoulli trial since the outcome can be only a head or a tail.

2. When a single bit is generated by a random number generator, either true or biased, this is a Bernoulli trial since the outcome can be only a '0' or '1'.

3. If $p$ is the probability of a success ( a head or a bit of '1') then $q$ is the probability of a failure (a tail or a bit of '0') and we must have, according to our definition:

$$p + q = 1$$

## II. Example:

A. Problem:

A biased coin turns up a head with a probability of $\frac{2}{3}$. What is the probability that exactly four heads are the result of seven flips, assuming the event that a tail turns up is independent of the event that a head is flipped?

B. Solution:

1. Conditions:

a. $| E | = C(7, 4)$

b. $P(h) = \frac{2}{3}$ and $P(t) = \frac{1}{3}$

2. Therefore:

$$\begin{aligned} | E | &= C(7,4) \times p(h)^4 \times p(t)^3 \\ &= C(7,4) \times (\tfrac{2}{3})^4 \, (\tfrac{1}{3})^3 \\ &= [7!/\{4!(7-4)!] \times (\tfrac{2}{3})^4 (\tfrac{1}{3})^3 \\ &= \tfrac{560}{2187} \end{aligned}$$

**III. Theorem:** The probaility of exactly $k$ successes in $N$ independent Bernoulli trials with probability of success $P$ and probability of failure $Q = 1 - P$ is $C(N, k)P^kQ^{N-k}$

**Proof:**

1. Each experiment of $n$ Bernoulli trials can be represented as a bit string of $n$ bits.

2. We will let a success be represented by '1' and a failure by '0'.

4. The probability of $k$ bits being set to '1' is $P^k$ while the probability that the remaining bits be set to '0' is $Q^{n-k}$

5. Because the $N$ trials are independent the probability of each outcome of $N$ trials consisting of $k$ successes and $N - k$ failures (in any order) is $P^kQ^{N-k}$

6. There are $C(N, k)$ $N$-tuples of $k$ 1's (successes) and $N - k$ 0's (failures).

7. Therefore, we have: $| E | = C(N, k)$

8. Therefore: $P(E) = C(N, k)P^kQ^{N-k}$

**IV. Random Variables**

A. Definition A **random variable** is a function $f: E \rightarrow R$, where $E$ is the sample space of an experiment and $R$ is the set of real numbers.

1. The function $f$ assigns a real number to each possible outcome of the experiment.

2. Note that a random variable is a function, it is **not** a variable and it is not random.

B. Example 1: Choose as a random variable the number of heads that appear for every possible output of flipping a fair coin three times.

1. If we designate the output of each experiment by $t$ then we can designate the corresponding value of the random variable by $X(t)$.

2. $X(t)$ takes on the following values:

$$X(HHH) = 3$$

$$X(HHT) = X(HTH) = X(THH) = 2$$

$$X(HTT) = X(THT) = X(TTH) = 1$$

$$X(TTT) = 0$$

C. Definition: The ***distribution*** $D$ of a random variable $X$ on a sample space $S$ is the set of pairs $(r,\ P(X) = r))$ for all $r \in S$ where $P(X = r)$ is the probability that $X$ takes on the value $r$.

Then:

1. $S = \{HHH,\ HHT,\ HTH,\ THH,\ HTT,\ THT,\ TTH,\ TTT\}$

2. $D = \{\ (3,\ HHH),\ (2,\ HHT),\ (2,\ HTH),\ (2,\ THH),\ (1,\ HTT),\ (1,\ THT),\ (1,\ TTH)\ (0,\ TTT)\}$

D. Definition: The **expected value** of a random variable $X(t)$ on the sample space $S = \{s_1, s_2, ..., s_n\}$ is equal to $E(X) = \sum_{i=1}^{n} p(s_i)X(s_i)$

In the example above:

$$E(X) = \sum_{i=1}^{8} p(s_i)X(s_i) = \tfrac{1}{8}(3 + 2 + 2 + 2 + 1 + 1 + 1 + 0)$$

$$= \tfrac{12}{8} = \tfrac{3}{2}$$

## V. Example: Probability of a Collision in Hashing Functions

A. Hashing Function: A hashing function computes using some characteristic of a record the location of that record on a storage device.

1. Traditionally hashing functions have employed very specialized logic to compute a record key.

2. Examples:

a. The third field of a product number divided by the third digit of the second field plus the product of the second and third digits of the third field.

b. The Social Security number modulus the street address.

B. Example: Many corporations assign a customer ID ($cID$) to each of their customers. This practice is the result of the increased reluctance of customers to divulge their Social Security Numer.

1. If the capacity of the storage device is $C$ records then a very simple hashing function $h$ might be

$$h(cID) = cID \bmod C$$

2. Then $record = h(cID)$ is the number, or key, of the record on a direct access storage device that contains that customer's data.

3. Criterion of goodness: The hashing function distributes the keys evenly, or with uniform density, among the available keys.

C. Problem: Two different ID's might return the same memory location, or a collision.

D. Calculation of the Probability of a Collision

1. Assume that the probability of the assignment of an ID to a record is:
   a. Uniform (A quality of good hashing functions, but one which is not guaranteed)

   b. Is equal to $\frac{1}{m}$, where $m$ is the number of available storage locations.

2. Suppose that the storage location addresses are given as $k_1, k_2, \ldots, k_m$

3. We can store the first record at any one of $m$ addresses.

4. The second recond can, then, stored at any one of $m - 1$ addresses.

5. The probability that the second record is not stored at the location of the first record, i.e., that $h(k_1) \neq h(k_2)$, is $p_2 = \frac{m-1}{m}$
   a. There are $m$ locations.
   b. The first record is occupying one of them.
   c. There are $m - 1$ free locations.

6. The probability that the third record is stored at any one of the $m - 2$ free locations is $p_3 = \frac{m-2}{m}$

7. Similarly, the probability that the *j*th record is stored at a free location is $\frac{m-(j-1)}{m}$ because $j - 1$ of the $m$ locations are occupied.

8. Because the assignment of any key is a task independent of the assignment of any other key the product rule applies.

9. Therefore, the probability that $n$ keys are mapped to $n$ separate locations is $P(n)$ where:

$$P(n) = \frac{m-1}{m} \times \frac{m-2}{m} \times ... \times \frac{m-(j-1)}{m} \times ..., \frac{m-(n-1)}{m}$$

$$= \prod_{i=1}^{n-1} \frac{m-i}{m}$$

10. The probability that one or more keys are not mapped to separate locations is, then:

$$1 - P(n) = 1 - \prod_{i=1}^{n-1} \frac{m-i}{m}$$

11. Note: The larger $n$ becomes:
    a. The fractions $\frac{m-i}{m}$ become smaller.
    b. The product $\prod_{i=1}^{n-1} \frac{m-i}{m}$ becomes smaller.
    c. The probability $1 - \prod_{i=1}^{n-1} \frac{m-i}{m}$ that one or more records are mapped to the same location approaches unity, or certainty.

## VI. Random Numbers

A. Random numbers generated by a recipe and, hence, are not truly random and are termed ***pseudorandom numbers***.

B. ***Linear Congruential Method*** The most commonly used procedure for generating such numbers.

1. Input:
    a. An integer ***modulus*** $M$
    b. An integer ***multiplier*** $A$
    c. An ***increment*** $C$
    d. A seed $S_0$ (often the current value of the system clock)

2. Requirements:
   a. $2 \leq A < M$
   b. $0 \leq C < M$
   c. $0 \leq S_0 < M$

3. Calculation: $X_{n+1} = (A \times X_n + C)\, mod M$

C. Warning: The numbers generated in this manner are not truly random.
Sequences of pseudorandom numbers have been compared to sequences of truly random numbers and have, in most cases, been found to be very close.

## VII. Application of Random Numbers: Monte Carlo Integration

A. Algorithms that make random choices for values of variable as part of the computation are known as ***probabalistic algorithms***.

B. A particular subset of probabalistic algorithms are known as Monte Carlo algorithms.
1. Always produce answers to problems.
2. A small probability exists that the answer is incorrect.
3. Probability that answer is incorrect decreases rapidly as the number of trials increases.

C. The process of integration consists of computing the area underneath a specified segment of a curve.
1. The segement of the curve is enclosed in a rectangle of known area.

2. Random numbers, all of which fall within the rectangle, are generated.

3. A count is maintained of:
   a. The total number of random numbers generated, $N_{tot}$.

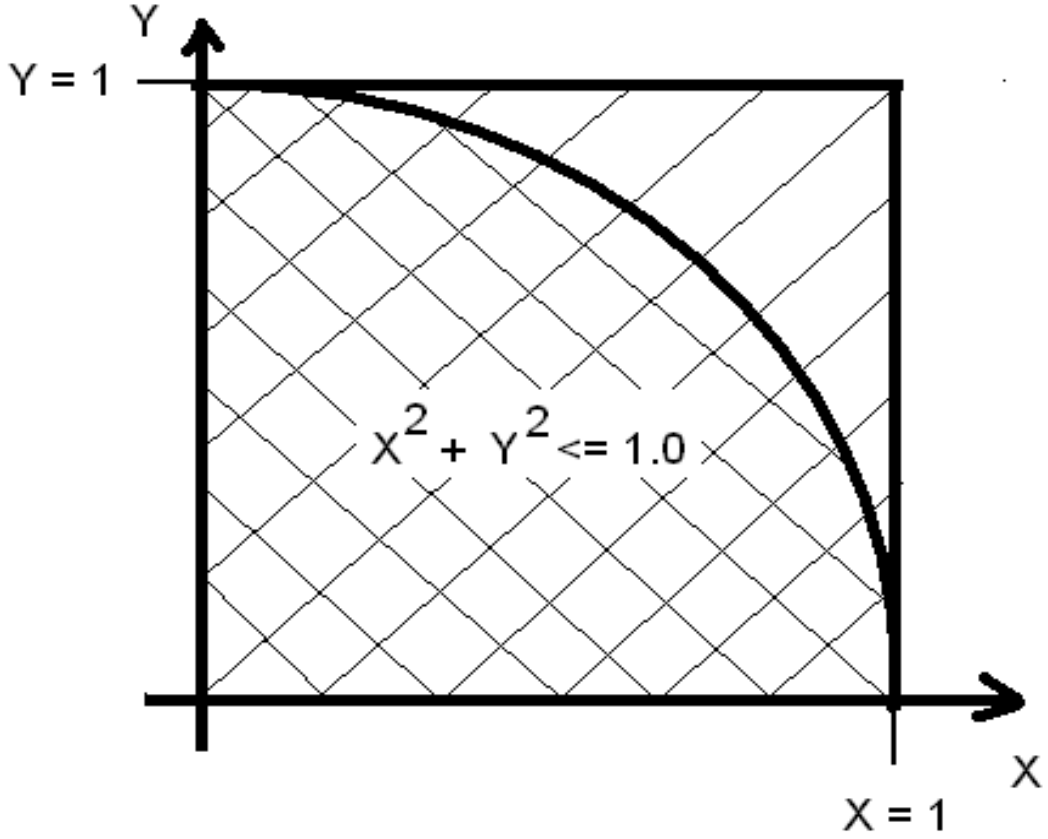
   b. Those random numbers that fall beneath the curve, $N_c$.

   c. The area $A$ beneath the curve is given by: $A = \frac{N_c}{N_{tot}}$

## VIII. Monte Carlo Integration Example: The Computation of $\pi$

A. Problem Description and Analysis

1. A ball is tossed onto a unit square with one quarter of a unit circle inscribed within it, as shown below.




2. If the sum of the squares of the coordinates of the landing point is less than or equal to unity, the ball has landed within the quarter unit circle

**B. Analysis**

1. Geometry

a. The area $A$ of the quarter unit circle is given by:

$$A = \frac{\pi r^2}{4.0} = \frac{\pi}{4.0} \quad \text{since } r = 1.0$$

b. Hence: $\pi = 4.0 \times A$

2. Geometrical Probability
   a. The probability $P$ that the ball will land within the quarter unit circle is given by the ratio of the area of the circle to the area of the square.

   b. Hence:
   $$P = \frac{Area\,of\,Quarter\,Circle}{Area\,of\,Unit\,Square} = \frac{\pi r^2}{4.0} \Big/ 1.0^2 = \frac{\pi r^2}{4.0} = \frac{\pi}{4.0}$$

   c. Therefore: $\pi = 4.0 \times P$

3. Emperical Probability
   a. An emperical measure of the probabibility that the ball will land in in quarter unit circle can be determined by simply tossing the ball into the unit square, counting the total number of tosses $N$ and the number of times the ball lands in the quarter unit circle, $C$.

   b. Then: $P = \frac{C}{N}$

4. Computer Simulation
   a. Generate a random values for $X$ and $Y$ where
   $$0.0 \le X \le 1.0 \quad \text{and} \quad 0.0 \le Y \le 1.0$$

   b. If $X^2 + Y^2 \le 1.0$ then the point lies within the quarter unit circle

   c. Repeat steps a and b $N$ times where the count of the points which fall within the quarter unit circle is maintained in the variable $C$.

   d. At the conclusion of the program logic we have:
   $$\pi = 4.0 \times P = 4.0 \times \frac{C}{N}$$

5. Programming Details
   a. To enable the largest possible number of repetitions two nested loops are used.

   b. The inner loop repeats the maximum number of times allowed for a long integer loop counter.

   c. The outer loop repeats for a number of trials determined by the user.

   d. An average is computed at each termination of the inner loop.

   e. The total of all counts is maintained in a value of data type double .

C. Source Code

```
package randompi;

import java.util.Random;

public class RandomPi
        {
        final   static  long        One = 1;
        final   static  double      Four = 4.0;
        final   static  long        Divisor = 1000;

        public static void main(String[] args)
                {
                Random randNum = new Random();

                double  avgPi = 0.0;
                double  PiI   = 0.0;
                double  Max = 0.0;
                double  Min = 9.0;
```

```
        for (long i = 1; i <= Divisor; i++)
                {
                long    IN     = 0;
                long    Total = 0;
                for (int   j = 0; j < Integer.MAX_VALUE; j++)
                        {
                        double  X = randNum.nextDouble();
                        double  Y = randNum.nextDouble();

                        if ((X*X + Y*Y) < One) IN++;
                        Total++;
                        }

                double  Area = (double) IN / (double) Total;
                PiI = Four * Area;
                avgPi = avgPi + PiI;

                if (PiI > Max) Max = PiI;
                if (PiI < Min) Min = PiI;
                double  Avg = avgPi / (double) i;
                System.out.println(i + "   Probabalistic Pi = " + PiI
                                + "  and the current average Pi = " + Avg);
                }

                avgPi = avgPi / (double) Divisor;

                System.out.printf("Probabalistic Pi = %17.15f\n",
                                        avgPi);
                System.out.printf("Largest Probabalistic Pi = %17.15f\n",
                                        Max);
                System.out.printf("Smallest Probabalistic Pi = %17.15f\n",
                                        Min);
                System.out.printf("Math.Java Pi = %17.15f\n",
                                        Math.PI);
        }
}
```

D. Output

Probabalistic Pi = 3.141593542950973
Largest Probabalistic Pi = 3.141688242154982
Smallest Probabalistic Pi = 3.141484831991366
Math.Java Pi = 3.141592653589793
BUILD SUCCESSFUL (total time: 4,872 minutes 6 seconds)

E. Analysis:

1. The error of $8.9 \times 10^{-6}$ suggests that the pseudorandom number generator approximates a true random number generator quite closely.

2. The considerably lower accuracy of the values computed by some individual executions of the inner loop suggests the need for a large number of trials.

3. The large number of trials,

$$1000 \times (2^{32} - 1) = 2,147,483,647,000$$

clearly contributed to the accuracy.