Elementary Mathematics for "Advanced Programming" - Concept to Code (Arithmetic & Basic Algebra)

Praseed Pai K.T.
Gadgeon Engineering Solutions
Kochi , Kerala

About the Presenter

- A Seasoned Software Engineering Professional with more than twenty five years of Exposure
- Author of Two books on Computer Programming
- Explorer in “Philosophical Tools for Software Engineering” (Has Presented on it, Written one university accredited paper, Designed a Pattern based on Advaita Vedanta to transition from OOP to FRP)
- An Expert level professional in Cross Cultural Encounters
- A Critique of Digital Technology Fads (Programmers will be better off , if they stick to Programming. Do not run after so called AI/ML, BlockChain etc) - “Plumbing is preferred over Painting!”
- I also help Programmers eliminate their “Math-Phobia”
- Currently, Designated as Sr. Solutions Architect @ Gadgeon

Source Code Available from

http://github.com/praseedpai/ElementaryMathForProgrammingSeries/tree/master/AlgebraNArith

Natural Number – Peano Way!

- Zero is a natural number.
- Every natural number has a successor in the natural numbers.
- Zero is not the successor of any natural number.
- If the successor of two natural numbers is the same, then the two original numbers are the same.
- If a set contains zero and the successor of every number is in the set, then the set contains the natural numbers.

Some notions about Equality

The expression "$x = y$" means that x and y are the same object. The symbol " $=$ " is called *equals*. "$x \neq y$" means that x and y are not the same object.

we assume the following:

I. For each x, $x = x$. In words, equals is *reflexive*.

II. For each x and for each y, if $x = y$, then $y = x$. (Equals is *symmetric*.)

III. For each x, for each y, and for each z, if $x = y$ and if $y = z$, then $x = z$. (Equals is *transitive*.)

Peano's Arithmetic in Semi Formal Terms

- 0 is a natural number.
- For every natural number x, $x = x$. That is, equality is **reflexive**.
- For all natural numbers x and y, if $x = y$, then $y = x$. That is, equality is symmetric.
- For all natural numbers x, y and z, if $x = y$ and $y = z$, then $x = z$. That is, equality is transitive.
- For all a and b, if b is a natural number and $a = b$, then a is also a natural number. That is, the natural numbers are closed under equality.
- For every natural number n, $S(n)$ is a natural number. That is, the natural numbers are closed under S.
- For all natural numbers m and n, $m = n$ if and only if $S(m) = S(n)$. That is, S is an injection.
- For every natural number n, $S(n) = 0$ is false. That is, there is no natural number whose successor is 0.

Peano's Axiom for Natural Number in Formal Notation

Peano Axioms for Natural Number Arithmetic

- Where s(x) stands for the successor of x:
  - $\forall x\ \forall y\ ((s(x) = s(y)) \rightarrow x = y)$
  - $\neg\exists x\ s(x) = 0$
  - $\forall x\ (x \neq 0 \rightarrow \exists y\ s(y) = x))$
  - $\forall x\ x + 0 = x$
  - $\forall x\ \forall y\ x + s(y) = s(x + y)$
  - $\forall x\ x * 0 =$ 0
  - $\forall x\ \forall y\ x * s(y) = x * y + x$

zero = Lf.Lx.x one = Lf.Lx.(f x) two = Lf.Lx.(f (f x)) three = Lf.Lx.(f (f (f x))) four = Lf.Lx.(F (f (f (f x))))

A C# Program to Implement Addition and Multiplication using Peano Arithmetic

FP to the Rescue!

Church Numerals

zero = Lf.Lx.x
one = Lf.Lx.(f x)
two = Lf.Lx.(f (f x))
three = Lf.Lx.(f (f (f x)))
four = Lf.Lx.(F (f (f (f x))))

```
public static INaturalNumber  Zero = new Zero();
public static INaturalNumber   One = new Successor(Zero);
public static INaturalNumber   Two = new Successor(One);
public static INaturalNumber Three = new Successor(Two);
public static INaturalNumber  Four = new Successor(Three);
public static INaturalNumber  Five = new Successor(Four);
public static INaturalNumber   Six = new Successor(Five);
public static INaturalNumber Seven = new Successor(Six);
public static INaturalNumber Eight = new Successor(Seven);
public static INaturalNumber  Nine = new Successor(Eight);
```

```
var two = new Successor(new Successor(new Zero()));
var three = new Successor(new Successor(new Successor(new Zero())));
var actual = two.Add(three);
Console.WriteLine(actual.Count());
var one = new Successor(new Zero());
two = new Successor(new Successor(new Zero()));
three = new Successor(new Successor(new Successor(new Zero())));
actual = one.Add(two).Add(three);
```

Operands can be any of the below in Arithematic

Adding More Power to Arithematic

- Natural Numbers (N)
- Whole Numbers (W)
- Integers (Z)
- Rationals (Q)
- Reals (R)
- Complex Numbers (C)
- Quaternions
- Octonions

Arithematic Expressions

- What is an Expression ?
- Expression is a chain of operations which are glued together
- An Expression consits of Terms , Factors and (Sub) Expressions!
- Terms are what you add and Factor is what you multiply
- Egs :- (2+3*4) = > 2 and 3*4 are terms
- 2 is a factor as it is 2*1
- 3 and 4 are factors as they are multiplied

- The Following Backus Naur Form can express an Expression

- <Expr> := <Term> (+ | *) <Expr>
- <Term> := <Factor> (* | /) <Term>
- <Factor> := + <Factor> | (<Expr>) | <Number> | -<Factor>

Mixed Mode Expressions in Arithematic

- We can mix number types in an expressions
- The necessity of Casting (Promotion or Coercion)
- E:= C + R => C + (C)R (0i + R)
  R + N => R + (R)N

Representing Expressions as Data (Abstract Syntax Trees)

```
// Abstract Syntax Tree (AST) for 5*10
Exp e = new BinaryExp(new NumericConstant(5),
            new NumericConstant(10),OPERATOR.MUL);
// Evaluate the Expression
Console.WriteLine(e.Evaluate(null));
// AST for  -(10 + (30 + 50 ) )
e = new UnaryExp(
       new BinaryExp(new NumericConstant(10),
       new BinaryExp(new NumericConstant(30),
             new NumericConstant(50),
                      OPERATOR.PLUS),
                OPERATOR.PLUS),
        OPERATOR.MINUS);
// Evaluate the Expression
Console.WriteLine(e.Evaluate(null));
```

Representing Numerical Value as AST

```
public class RUNTIME_CONTEXT{
    public RUNTIME_CONTEXT() {}
}
public enum OPERATOR{
    ILLEGAL = -1,PLUS,MINUS, DIV,MUL
}
public abstract class Exp {
    public abstract double Evaluate(RUNTIME_CONTEXT cont);
}
public class NumericConstant : Exp {
    private double _value;
    public NumericConstant(double value){_value = value;}
    public override double Evaluate(RUNTIME_CONTEXT cont) { return _value;}
}
```

Representing Binary Expressions as AST

```
public class BinaryExp : Exp
{
    private Exp _ex1, _ex2;
    private OPERATOR _op;

    public BinaryExp(Exp a, Exp b, OPERATOR op){
        _ex1 = a; _ex2 = b; _op = op;
    }
    public override double Evaluate(RUNTIME_CONTEXT cont){
        switch (_op){
            case OPERATOR.PLUS:
                return _ex1.Evaluate(cont) + _ex2.Evaluate(cont);
            case OPERATOR.MINUS:
                return _ex1.Evaluate(cont) - _ex2.Evaluate(cont);
            case OPERATOR.DIV:
                return _ex1.Evaluate(cont) / _ex2.Evaluate(cont);
            case OPERATOR.MUL:
                return _ex1.Evaluate(cont) * _ex2.Evaluate(cont);
         }
         return Double.NaN;
     }

}
```

Representing Unary Expressions as AST

```
public class UnaryExp : Exp
{
    private Exp _ex1;
    private OPERATOR _op;
    public UnaryExp(Exp a, OPERATOR op){
        _ex1 = a;
        _op = op;
    }
    public override double Evaluate(RUNTIME_CONTEXT cont){

        switch (_op){
            case OPERATOR.PLUS:
                return _ex1.Evaluate(cont);
            case OPERATOR.MINUS:
                return -_ex1.Evaluate(cont);
        }

        return Double.NaN;

    }

}
```

Representing Unary Expressions as AST

```
public class UnaryExp : Exp
{
    private BinaryExp  _ex1;
    public UnaryExp(Exp a, OPERATOR op)
    {

        _ex1 = new BinaryExp(new NumericConstant(0),a,_op);

    }
    public override double Evaluate(RUNTIME_CONTEXT cont)
    {
         return _ex1.Evaluate(cont);

    }

}
```

Mathematics has got Unary Operators and Binary Operators . How Operator Composition works?

+ N and – N are basically Binary Expressions

  +N == o + N

  -N == 0 - N

A op N and B op N are Binary Expressions

How do Ternary and Higher Order Expression Works ?

If an Operator has got Closure, Associativity and Identity

  Higher Order Expressions can work!

a + b + c + d is (((a + b) + c) + d)

Want to Learn Compiler Construction?

- Check SlangForDotnet
  - **https://github.com/praseedpai/SlangForDotNet**
- An Electronic Book titled , “The Art of Compiler Construction” included as part of the resource kit
- A Seven Step Iterative Approach
- A One Pass Compiler with Support for if/else,while,recursive functions
- A Tree Walking Interpreter and IL Code Generator included
- Has been ported to Java, Python, C++ and JavaScript

Properties of Arithematic Operators

A Good Operator(s) should have

- Associativity
- Commutativity
- Closure (Type of Result does matter)
- Distributivity (Two ops in an Expression)
- Inverse Element (Additive/Multiplicative)
- Identity Element (0 | 1 | “”)

Associative Property

- ((A op B) op C) == (A op (B op C))
- 2 + 3 + 4 can be written as
  ((2 + 3) + 4) or (2 + (3 + 4))
- If a operator is associative, we can do parallel reduction (a long sequence of numbers can be chunked into small sequence to be reduced by different people, processors or mechanical devices)

Commutative Property

- (A op B) == (B op A)
- 2 * 3 can be written as 3 * 2
- If a operator is commutative, order in which one performs operation does not matter
- We can do Out of Order Execution (Relational DB Engine exploits this property to evaluate relational cross products)
- On top of Parallel reduction, we can perform Parallel shuffle as well

Closure

- When two Homogeneous Types of numbers are Operated Upon, if we get the same Type as result, it is called “Closure”
- Addition of Two natural numbers are closed
- So do Multiplication of Two Natural Numbers
- Closure helps us to Chain Operations without much “trouble”

Closure in Regular Expressions

Re(NULL) => NULL
Re(“”) => “”
Re([a-z]) => [a-z]
Re.Re => Re
(Re | Re) => Re
Re* => Re
The above stuff defines Re (Recursive definition)
What about R+?
Re+ = Re.Re*

Closure in SQL

Data is stored in a data structure called Relation
Relations can be combined using Rel Ops

CartesianProduct(Rel1,Rel2..Reln) => Rel
Restrict(Rel,Predicate) => Rel
Project(Rel,fieldlist) => Rel
Rename(Rel)=> Rel
SetOperators(Rel1..Reln) => Rel
Group(Rel,Pred) => Rel
And so on…

Infix/Prefix and Postfix notation

- Different Notations for Expressions
- Infix Notation (Mathematics and most Programming languages)
- Prefix Notation (LISP/Scheme uses it)
- PostFix Function (Stack based Evaluation , FORTH and Display PostScript)

```
2 + 3 * 4              => Infix
2 3 4 * +              => PostFix
( + 2  ( * 3 4 ))      => Prefix
```

How Lisp/Scheme Works ?

- The Code is stored as a List
- The First Element of List is denoted by Car(Lst)
- The Rest of the List is denoted by Cdr(Lst)
- Eval(Lst) is the Evaluation Function
- Apply(fn)

- Eval(Lst) :=> if (C := Car(Lst)) { return Value(C) }
  else if (V := Car(Lst)) { return Env.Lookup(V) }
  else { return Apply(Car(Lst),Eval(Cdr(Lst)))|

Apply(fn,Lst) :=> { return fn(Lst) }

How do I aggregate numbers from 1 to N ?

A For Loop and an accumulator is the obvious solutions. It is a Linear Solution (O(N)

Can we have a O(1) Solution for this problem ? ((n*(n+1))/2)

What about sum from K to N ?

```
// Returns Products of First  N
static BigInteger Product(int N) {
    BigInteger f = new BigInteger("1");
    for (int i = 2; i <= N; i++)
        f = f.multiply(BigInteger.valueOf(i));
    return f;
}
 // Returns Sigma of First  N
static BigInteger Sigma(int N) {
    BigInteger f = new BigInteger("0");
    for (int i = 1; i <= N; i++)
        f = f.add(BigInteger.valueOf(i));
    return f;
}

// Returns Sigma of First  N WithoutLoop
static BigInteger SigmaWithOutLoop(int N) {
    BigInteger f = BigInteger.valueOf(N);
    // N*(N+1)/2
    f = f.multiply(f.add( BigInteger.valueOf(1))).divide(new BigInteger("2"));
    return f;
}
```

```
// Driver method
public static void main(String args[]) throws Exception
{
     int N = 200;

     int len = args.length;
     if(len==0) {
                        System.out.println("No args");
                        return;
      }

      N = Integer.parseInt(args[0]);
      System.out.println(Product(N));
      System.out.println(Sigma(N));
      System.out.println(SigmaWithOutLoop(N));

}
```

A Challenge – What is the basis for this "cocksureness" ?

Sunday, December 01, 2013

A Rs. 1,00,00,000 Offer from me , If you are able to find a triplets (3 #'s satisfying a mathematical property)

When I learned to write computer programs,finding Pythagorean triplets was one of the first programs written by me. One can easily find lot of triplets which satisfy Pythagorean triangle equation ,
$x^2 + y^2 = z^2$.

Some examples are , $4^2 + 3^2 = 5^2$
$12^2 + 5^2 = 13^2$

The challenge is to find out triplets for any number n , provided n > 2 which satisfies the equation
$x^n + y^n = z^n$.

If you are able to find a integer solution, you will get this reward !

Fermat's Last Theorom

Words which gave Mathematicians Sleepless nights for Centurites

The Celebrated French Mathematician wrote the following in one of his book.

"It is impossible to separate a cube into two cubes, or a fourth power into two fourth powers, or in general, any power higher than the second into two like powers. I have discovered a truly marvelous proof of this, which this margin is too narrow to contain."

What essentially, he told was

$$x^3 <> y^3 + y^3 \text{ or } x^4 <> y^4 + z^4,$$
$$\text{in general, } x^n + y^n <> z^n \text{ for } n > 2$$

We are familiar with Pythagores Therom, whcih states that,

"In a right triangle, hypotenuse (z) squared is equivalent to base (x) squared added to opposite side(y)"
Algebraically, It can be written as,

$$z^2 = x^2 + y^2.$$

Diophantine equations are generalization of Pythagoras theorem where factors (x,y,z) are integral numbers. What Fermat essentially told us was , **For a Diophantine equation with powers greater than 2, there is no solutions.**

Tony Wiles, British mathematician proved it finally, in 1994. Counless mathematicians have attempted a proof and Finally, a solution was found in 1994 (proposed in 1993), 300 years after Fermat's death.

Pythagorean Triplets (Square exists, Cube does not!)

```
for( x = 1.0 ; x <= 100; x=x+1.0 )
     for( y = 1.0 ; y <= 100 ; y=y+1.)
            for( z = 1.0 ; z <=100; z=z+1. )  {
                 if ( x*x + y*y == z*z )
                          printf(" triplets %d\t\t%d\t\t%d\n",(long)x,(long)y,(long)z);

                   if ( x*x*x + y*y*y == z*z*z )
                        printf("cube =%d\t\t%d\t\t%d\n",(long)x,(long)y,(long)z);
           }
```

Two case Studies of Formal Verification

- The Verification of “COM Component”
- The Verification of a Custom Type which mimics the semantics of int, double , float

Java/C# has builtin Interface Verification

```
interface IA { void SayHello(String s); }
interface IB { void SayHello2(String s); }
interface IC { void SayHello3(String s); }
class UberComponent implements IA,IB,IC{
   public void SayHello(String s) { System.out.println("IA"); }
   public  void SayHello2(String s) {System.out.println("IB");}
   public  void SayHello3(String s) {System.out.println("IC");}
}
public class FormalVerify {
    static public void main(String[] args){
         //--------------- Check For Reflexivity IA = IA
         IA ia = new UberComponent(); IA ia2 = (IA) ia; ia2.SayHello(",");
         //--------------------- Check For Symmetry
         IB ia3 = (IB) ia; ia2 = (IA) ia3; ia2.SayHello(","); ia3.SayHello2(",");
         //----------- Check For Transitivity
         IC ia4 = (IC) ia3; IA ia5 = (IA) ia4; ia5.SayHello(","); a4.SayHello3("m");
    }
}
```

Calendar Calculation

- Compute Leap Year
- Gregorian Calendar
- Modulo 7 Arithmetic
- 0 – 6 (Sunday To Saturday)

How to Compute CDOW ?

```
//--- Take the Previous Year , Take Modulo 400
tempcomp = year-1;
tempcomp = tempcomp%400;
//--- Depending upon reminder, odd days has to be assigned
if (tempcomp >= 300 ) { tempcomp = tempcomp%300; oddday = 1;}
else if ( tempcomp >= 200 ) { tempcomp = tempcomp%200; oddday = 3; }
else if ( tempcomp >= 100 ) { tempcomp = tempcomp%100; oddday = 3;}
else  { oddday = 0; }

//---- Adjust the Number of Year and Number of Leap Years
oddday = ( oddday + ( tempcomp + (long)(tempcomp/4) ));
//------- Adjust for the month
oddday = ( oddday + accum[ month- 1 ]);
//---- Adjust for the Leap Year
if ( month > 2 && isleap(year) ) { oddday +=1; }
//------------ Adjust the Odd days for DAYs and take modulo 7
oddday = ( oddday + day ) % 7;
//-------------- Look up in the days array to find CDOW
printf("The day is %s\n" , days[oddday]);
return 0;
```

MRF SpeedCheck @ Sharjah and It's Math

```
#include <stdio.h>


void main()
{
    double fmiles=0.0;
    float miles=0.0;
    double kmph;
    printf("Enter the Speed at which bowler bowls \n");
    scanf("%f",&miles);
    fmiles =(double) miles;
    kmph = fmiles*1.6;
    kmph = kmph * 1000/3600;
    kmph = 20.12/kmph;
    printf("The time which batsmen will get is %f seconds\n",kmph);
}
```

Russian Multiplication

~~57~~	~~86~~
114	43
228	21
~~456~~	~~10~~
912	5
~~1824~~	~~2~~
+ 3648	1
4902	

```
public static long Calc(int x,int y) {
    int a = x; int b = y; long sum = 0;
    while(a >= 1) {
      if( a % 2 !=0)
        sum = sum+b;
      a = a >> 1;
      b = b<<1;
    }
    return sum;
  }
```

A Mathematical Trivia

Arvind Kejriwal & A "nerdy" joke re-visited

Years ago, from a book, "Secured Programming Cookbook", I encountered the following sentence,
"**there are 10 types of people in this world, those who understand binary and those who dont**". It took some time for me to understand the joke inside. **The number 10, If interpreted as binary, is decimal two**.

Arvind Kerjriwal, in April,2014 told the following

"**AAP will Win 100 LS seats, No question of tie-ups**"

A joke appeared on the Facebook, to interpret the numbers in binary. 100 - in binary, is 4 decimal. Exactly the number of seats won by AAP in 2014.

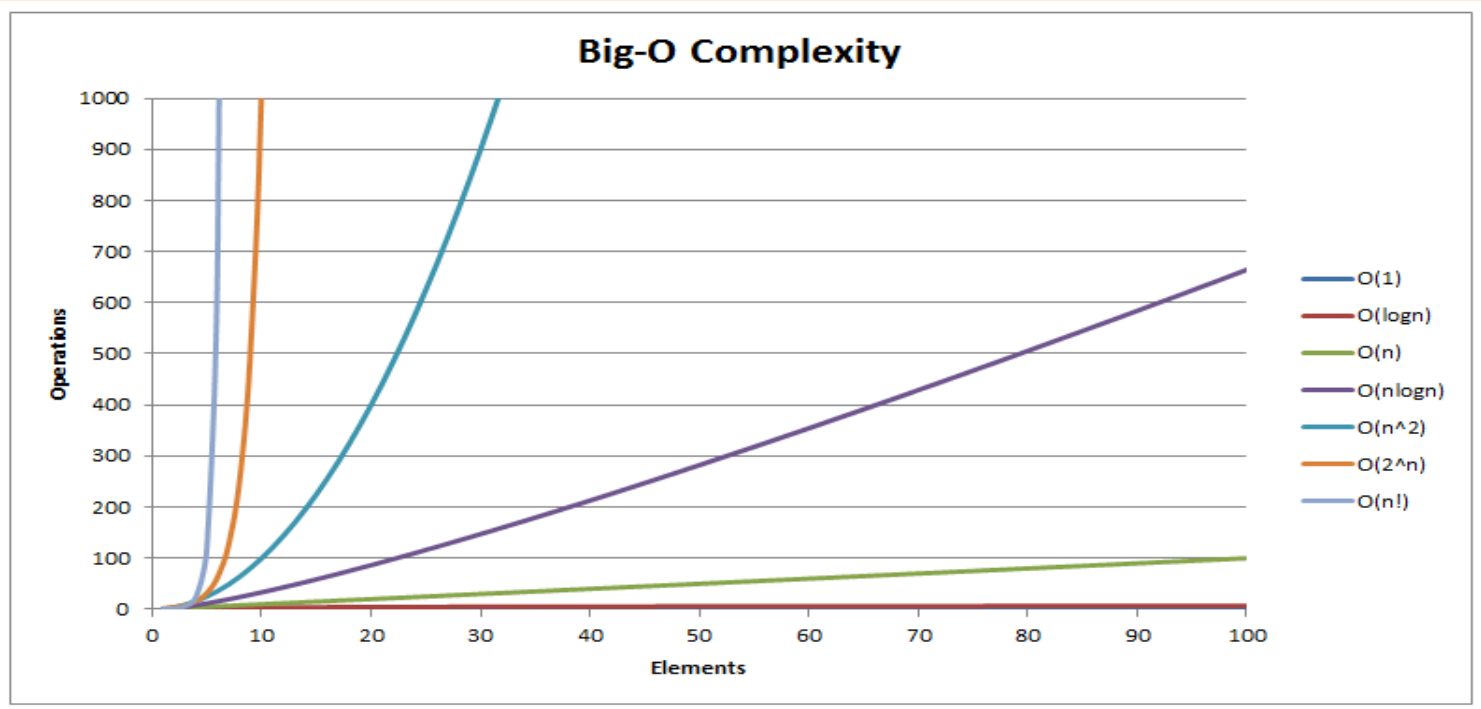
How To Permute a Sequence ?

123	(	ABC	)
132	(	ACB	)
213	(	BAC	)
231	(	BCA	)
312	(	CAB	)
321	(	CBA	)

How To generate a Subset?

ABC	000	{}
ABC	001	{C}
ABC	010	{B}
ABC	011	{B,C}
ABC	100	{A}
ABC	101	{A,C}
ABC	110	{A,B}
ABC	111	{A,B,C}

Algorithmic Complexity



What is “e” ?

```
// e => ( n + (1/n))^n
//          as
//      lim n => INFINITY
//
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main( int argc , char **argv ) {

   if ( argc == 1 ) { return 0; }
   int n = atoi(argv[1]);
   double nucleus = 1.0 + (1.0  /(double)n);
   double exp_one   = exp(1.0);
   double exp_brute = pow(nucleus,(double)n);
   printf("CRT exp = %g\t BRUTE exp = %g\n",exp_one,exp_brute);
}
```

Rule of 70

$$
\begin{aligned}
(e^r)^p &= 2 \\
e^{rp} &= 2 \\
\ln e^{rp} &= \ln 2 \\
rp &= \ln 2 \\
p &= \frac{\ln 2}{r} \\
\\
p &\approx \frac{0.693147}{r}
\end{aligned}
$$

Logarithm Soup

$$\log_{10}(x) = \frac{\ln(x)}{\ln(10)} \qquad \text{or} \qquad \log_{10}(x) = \frac{\log_2(x)}{\log_2(10)}$$

Nth Root

```
public static double NthRoot(double num, double  n)
{

    return Math.Exp((1 / n) * Math.Log(num));

}
```

Arithmetic Mean

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

$$\bar{x} = \frac{\sum x}{n}$$

$$\mu = \frac{\sum x}{n}$$

$$\mathrm{E}[X] = \sum_{i=1}^{k} x_i\, p_i = x_1 p_1 + x_2 p_2 + \cdots + x_k p_k.$$

$$\mathrm{E}[X] = 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 4 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} + 6 \cdot \frac{1}{6} = 3.5.$$

$$\mathrm{E}[X] = \mu.$$

```
private static double Aggregate(  double [] p ,
                double init,Func<double,double,double> fn) {
        double temp = init;
        foreach (var n in p)
            temp =  fn(n,temp);
        return temp;
}

private static double AMEAN(double[] p) {
        return Aggregate(p, 0, (double a, double b) =>
                        { return b += a; }) / p.Length;
}
```

Geometric Mean

$$\left(\prod_{i=1}^{n} a_i\right)^{\frac{1}{n}} = \sqrt[n]{a_1 a_2 \cdots a_n}.$$

When $a_1, a_2, \ldots, a_n > 0$

$$\left(\prod_{i=1}^{n} a_i\right)^{\frac{1}{n}} = \exp\left[\frac{1}{n}\sum_{i=1}^{n} \ln a_i\right]$$

additionally,

$$\left(\prod_{i=1}^{n} a_i\right)^{\frac{1}{n}} = (-1)^m \exp\left[\frac{1}{n}\sum_{i=1}^{n} \ln|a_i|\right]$$

where m is the number of negative numbers.

```
private static double Aggregate(  double [] p ,
                double init,Func<double,double,double> fn) {
        double temp = init;
        foreach (var n in p)
            temp =  fn(n,temp);
        return temp;
}

private static double GMEAN(double[] p){
     double pi = Aggregate(p, 1,
         (double a, double accum) => { return accum *= a; });
     return Math.Exp(Math.Log(pi)*(1 / p.Length));
}
```

Harmonic Mean

The harmonic mean *H* of the positive real numbers $x_1, x_2, \ldots, x_n$ is defined to be

$$H = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}} = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}} = \left(\frac{\sum_{i=1}^{n} x_i^{-1}}{n} \right)^{-1}.$$

$$1/H(1/x_1 \ldots 1/x_n) = A(x_1 \ldots x_n)$$

Arun Travelled from A to B , at 60 km per hour. While returning, he travelled at 20 km per hour. What is the Average Speed?

Average speed

$$= \frac{\text{Total distance traveled}}{\text{Total time taken}}$$

$$= \frac{2d}{\frac{d}{x} + \frac{d}{y}} = \frac{2d}{\frac{yd + xd}{xy}} = \frac{2dxy}{d(x + y)}$$

$$= \frac{2xy}{x + y}$$ (harmonic mean of *x* and *y*)

Standard Deviation

$$\sigma = \sqrt{\frac{1}{N}\left[(x_1-\mu)^2 + (x_2-\mu)^2 + \cdots + (x_N-\mu)^2\right]}, \text{ where } \mu = \frac{1}{N}(x_1 + \cdots + x_N),$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2}, \text{ where } \mu = \frac{1}{N}\sum_{i=1}^{N} x_i.$$

The "**Population** Standard Deviation": $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2}$

The "**Sample** Standard Deviation": $s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i-\overline{x})^2}$

$$\mathrm{E}[X] = \mu.$$

$$\begin{aligned}
\sigma &= \sqrt{\mathrm{E}[(X-\mu)^2]} \\
&= \sqrt{\mathrm{E}[X^2] + \mathrm{E}[-2\mu X] + \mathrm{E}[\mu^2]} \\
&= \sqrt{\mathrm{E}[X^2] - 2\mu\,\mathrm{E}[X] + \mu^2} \\
&= \sqrt{\mathrm{E}[X^2] - 2\mu^2 + \mu^2} \\
&= \sqrt{\mathrm{E}[X^2] - \mu^2} \\
&= \sqrt{\mathrm{E}[X^2] - (\mathrm{E}[X])^2}
\end{aligned}$$

Standard Deviation (STD)

```
private static double Aggregate(  double [] p ,
              double init,Func<double,double,double> fn) {
      double temp = init;
      foreach (var n in p)
          temp =  fn(n,temp);
      return temp;
}

private static double STD(double[] p){
     double avg = Aggregate(p, 0,
                  (double a, double b) => { return b += a; }) / p.Length;
     double var = Aggregate(p, 0,
                  (double a, double b) => { return b += ((a - avg)*(a-avg)); }) / p.Length;
     return Math.Sqrt(var);
}
```

Mean Absolute Deviation

The mean absolute deviation of a set $\{x_1, x_2, ..., x_n\}$ is

$$\frac{1}{n}\sum_{i=1}^{n}|x_i - m(X)|.$$

Covariance

$$\operatorname{cov}(X,Y) = \frac{1}{n}\sum^{n}(x_i - E(X))(y_i - E(Y)).$$

$$\operatorname{cov}(X,Y) = \operatorname{E}[(X - \operatorname{E}[X])\,(Y - \operatorname{E}[Y])]$$

$$\begin{aligned}
\operatorname{cov}(X,Y) &= \operatorname{E}[(X - \operatorname{E}[X])\,(Y - \operatorname{E}[Y])] \\
&= \operatorname{E}[XY - X\operatorname{E}[Y] - \operatorname{E}[X]Y + \operatorname{E}[X]\operatorname{E}[Y]] \\
&= \operatorname{E}[XY] - \operatorname{E}[X]\operatorname{E}[Y] - \operatorname{E}[X]\operatorname{E}[Y] + \operatorname{E}[X]\operatorname{E}[Y] \\
&= \operatorname{E}[XY] - \operatorname{E}[X]\operatorname{E}[Y],
\end{aligned}$$

Correlation

$$\rho_{X,Y} = \operatorname{corr}(X,Y) = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\operatorname{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

$$\rho_{X,Y} = \frac{\operatorname{E}(XY) - \operatorname{E}(X)\operatorname{E}(Y)}{\sqrt{\operatorname{E}(X^2) - \operatorname{E}(X)^2} \cdot \sqrt{\operatorname{E}(Y^2) - \operatorname{E}(Y)^2}}$$

BETA (Finance)

$$\beta = \frac{\operatorname{Cov}(r_a, r_b)}{\operatorname{Var}(r_b)},$$

$$\sigma_a = \sqrt{\operatorname{Var}(r_a)},\ \sigma_b = \sqrt{\operatorname{Var}(r_b)},\ \rho_{a,b} = \operatorname{Cov}(r_a, r_b) / \sqrt{\operatorname{Var}(r_a)\operatorname{Var}(r_b)},$$

$$\beta = \rho_{a,b} \frac{\sigma_a}{\sigma_b}$$

Moment

Moment ordinal	Moment		
	Raw	Central	Standardized
1	Mean	0	0
2	–	Variance	1
3	–	–	Skewness
4	–	–	(Non-excess or historical) kurtosis

EMI

$$P = A \cdot \frac{1 - (1 + r)^{-n}}{r}$$

$$A = P \cdot \frac{r(1 + r)^{n}}{(1 + r)^{n} - 1}$$

Derivation of EMI

$P_1 = P \times (1 + r) - E$

$P_2 = P_1 \times (1 + r) - E$

$P_2 = (P \times (1 + r) - E) \times (1 + r) - E$

$P_2 = P \times (1 + r)^2 - E \times ((1 + r) + 1)$

$P_2 = P \times t^2 - E \times (1 + t)$

$P_i = P \times t^i - E \times (1 + t + t^2 + \ldots + t^{i-1})$

$P_n = P \times t^n - E \times (1 + t + t^2 + \ldots + t^{n-1}) = 0$

$P \times t^n = E \times (1 + t + t^2 + \ldots + t^{n-1})$

$P \times t^n = E \times (t^n - 1)/(t - 1)$

$E = P \times t^n \times (t - 1)/(t^n - 1)$

$E = P \times r \times (1 + r)^n/((1 + r)^n - 1)$

$$E = P \cdot r \cdot \frac{(1 + r)^n}{((1 + r)^n - 1)}$$

EMI Computation

```
import java.lang.*;


public class Emi {
  public static double Get_Emi( double principal , double rate , long n ){
       double percent_rate = rate/(12.0*100.0);
       return  principal/(( 1 - Math.pow(1+percent_rate,-n*12.0))/percent_rate);
  }
}
```

$$P = A \cdot \frac{1 - (1 + r)^{-n}}{r}$$

$$A = P \cdot \frac{r(1 + r)^{n}}{(1 + r)^{n} - 1}$$

IRR – Internal Rate of Return

Quadratic Equation

$$ax^2 + bx + c = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
public static double Quadratic(double a, double b, double c,
                               ref double rt1 , ref double rt2 )
{
    double disc = b*b - 4*a*c;

    if (disc  < 0.0 )
        return Double.NaN;


    double root1 = (-b + Math.Sqrt(disc)) / (2*a) ;
    double root2 = (-b - Math.Sqrt(disc)) / (2*a) ;
    rt1 = root1;
    rt2 = root2;
    return 0.0;
 }
```

For investing 100, if u get 60 rs each for two years, what is IRR

$$100 = \frac{60}{1+r} + \frac{60}{(1+r)^2}.$$

$$60x^2 + 60x - 100 = 0,$$

$$x = \frac{-60 \pm \sqrt{60^2 + 4(60)(100)}}{120}.$$

$$x = \frac{\sqrt{27{,}600} - 60}{120} \approx .8844.$$

$$1 + r^* \approx \frac{1}{.8844} \approx 1.131.$$

IRR

$$\mathrm{NPV} = \sum_{n=0}^{N} \frac{C_n}{(1+r)^n} = 0$$

If an investment may be given by the sequence of cash flows

Year (n)	Cash flow (C_n)
0	-123400
1	36200
2	54800
3	48100

then the IRR r is given by

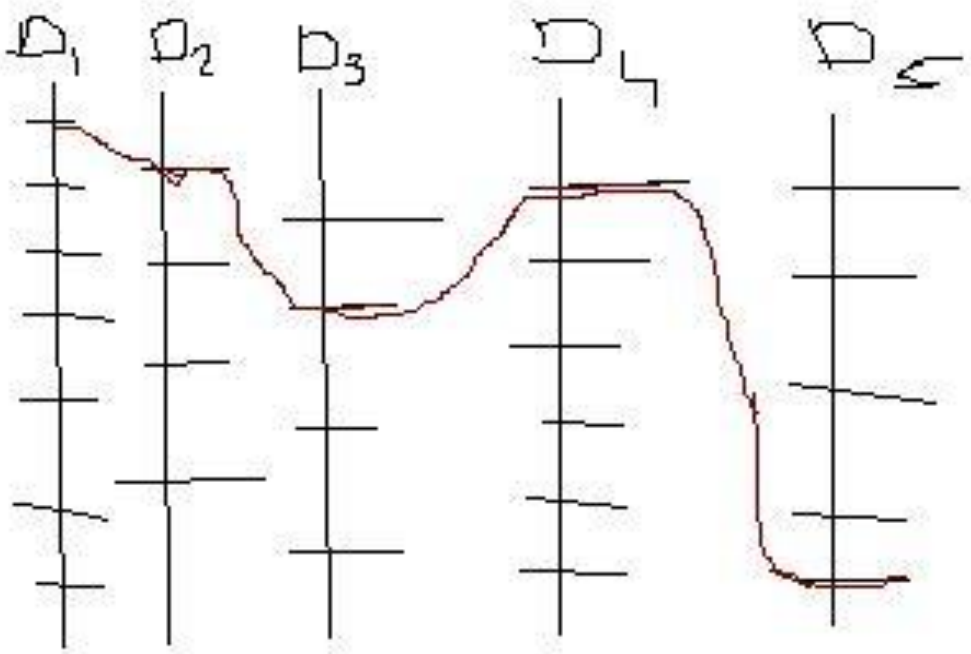
$$\mathrm{NPV} = -123400 + \frac{36200}{(1+r)^1} + \frac{54800}{(1+r)^2} + \frac{48100}{(1+r)^3} = 0.$$

In this case, the answer is 5.96% (in the calculation, that is, r = .0596).

Algebra To Modern Algebra

- Why Limit Algebra to Number Types?
- Algebra of Sets (Operations on Sets)
- Algebra of Relations
- Algebra of Matrices
- Algebra of Vectors
- Algebra on Groups, Rings, Fields, Lattices
- We Create Hierarchy of Mathematical Structures with Varying Property
- Mathematical Properties like Associativity,Commutativity, Closure nicely extrapolates to Modern Algebra

How to Visualize more than three dimensions



A point can represent one dimension (number line) , a line can represent 2 dimension (a plane) and a cube three dimension (space) . when it comes to four and above , cartesian geometric depiction of dimension fails .

2.161 "There must be something identical in a picture and what it depicts , to enable one to be a picture of the other at all"

Any Questions?!