Primitive Polynomial Testing

Sean E. O'Connor

artifex@seanerikoconnor.freeservers.com

Abstract

We describe methods for testing whether a polynomial is primitive or irreducible using a *Mathematica script.* The straightforward test to determine whether a polynomial f(x) is primitive over GF(p) by checking if x is a generator of the field GF(p^n) is exponentially slow. We show a much faster test, due to Alanen and Knuth based on factoring $\frac{p^n-1}{p-1}$.

1. Preliminaries

First we have to set up a series of helper functions. Let's try them out in GF(5^4), using the primitive polynomial,

In[526]:=
```
p = 5; n = 4; f = x^4 + x^2 + 2 x + 3;
```

The number of elements in the field, which is the total number of polynomials is

In[527]:=
```
NumberOfPolynomials[p, n] := p^n
```

In[528]:=
```
NumberOfPolynomials[p, n]
```

Out[528]=
```
625
```

The number of distinct *primitive* polynomials is $\phi(\frac{p^n-1}{n})$

In[529]:=
```
NumberOfPrimitivePolynomials[ p_, n_ ] := EulerPhi[p^n - 1] / n
```

In[530]:=
```
NumberOfPrimitivePolynomials[p, n]
```

Out[530]=
```
48
```

The number of polynomials with linear factors is $\sum_{i=0}^{n} p^{n-i} (-1)^i \binom{p}{i}$

In[531]:=
```
NumberOfPolynomialsWithLinearFactors[ p_, n_ ] := Sum[p^(n-i) (-1)^i Binomial[p, i], {i, 0, n}]
```

In[532]:=
```
NumberOfPolynomialsWithLinearFactors[p, n]
```

Out[532]=
```
205
```

The probabilities of a random polynomial having no linear factors, and being primitive are,

In[533]:=
```
{1 - NumberOfPolynomialsWithLinearFactors[p, n] / NumberOfPolynomials[p, n],
  NumberOfPrimitivePolynomials[p, n] / NumberOfPolynomials[p, n]} // N
```

Out[533]=
```
{0.672, 0.0768}
```

Is **a** a primitive root of p, i.e. does **a** generate the multiplicative group GF(p)?

In[534]:=
```
IsPrimitiveRoot[ a_, p_ ] :=
  If[MultiplicativeOrder[a, p] == p - 1,
   isPrimitiveRoot = True, isPrimitiveRoot = False];
```

In[535]:=
```
IsPrimitiveRoot[4, p]
```

Out[535]=
```
False
```

In[536]:=
```
Table[Mod[4^k, p], {k, 1, p - 1}]
```

Out[536]=
```
{4, 1, 4, 1}
```

In[537]:=
```
IsPrimitiveRoot[3, p]
```

Out[537]=
```
True
```

In[538]:=
```
Table[Mod[3^k, p], {k, 1, p - 1}]
```

Out[538]=
```
{3, 4, 2, 1}
```

Extract a polynomial's constant term,

In[539]:=
```
ConstantTermOfPolynomial[f_] := CoefficientList[f, x][[1]];
```

In[540]:=
```
ConstantTermOfPolynomial[f]
```

Out[540]=
```
3
```

Test if a polynomial is constant,

In[541]:=
```
PolynomialIsConstant[f_] := If[Exponent[f, x] == 0, True, False]
```

In[542]:=
```
PolynomialIsConstant[3]
```

Out[542]=
```
True
```

Evaluate a polynomial at {0,...,p-1} to check if it has any zeros over GF(p),

In[543]:=
```
PolynomialHasLinearFactor[f_, p_] :=
 MemberQ[ Table[Mod[f, p] /. x → i, {i, 0, p - 1}], 0]
```

In[544]:=
```
PolynomialHasLinearFactor[x^4 + 2, p]
```

Out[544]=
```
False
```

In[545]:=
```
PolynomialHasLinearFactor[x + 2, p]
```

Out[545]=
```
True
```

Evaluate $x^{pk} \pmod{f(x), p}, 0 \le k \le n-1$

In[546]:=
```
PowersOfXModFP[f_, p_, n_] :=
 Table[ PolynomialMod[ x^(p k) , {f, p} ], {k, 0, n - 1}]
```

In[547]:=
```
PowersOfXModFP[f, p, n]
```

Out[547]=
```
{1, 2 x + 3 x^2 + 4 x^3, 3 + 4 x^2 + x^3, 4 x + 4 x^2 + 3 x^3}
```

Form the Q matrix from the rows of coefficients of the powers
$x^{pk} \pmod{f(x), p}, 0 \le k \le n-1$

In[548]:=
```
PadRightRows[L_, n_] := Map[ Function[ x, PadRight[x, n]], L]
```

In[549]:=
```
PadRightRows[{{1}, {0, 0, 1}, {1, 0, 1}, {1}}, n]
```

Out[549]=
```
{{1, 0, 0, 0}, {0, 0, 1, 0}, {1, 0, 1, 0}, {1, 0, 0, 0}}
```

In[550]:=
```
QMatrix[ f_, p_, n_ ] :=
 PadRightRows[CoefficientList[ PowersOfXModFP[f, p, n], x ], n]
```

In[551]:=
```
QMatrix[f, p, n] // MatrixForm
```

Out[551]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 3 & 4 \\ 3 & 0 & 4 & 1 \\ 0 & 4 & 4 & 3 \end{pmatrix}$$

In[552]:=
```
QIMatrix[f_, p_, n_] := Mod[ QMatrix[f, p, n] - IdentityMatrix[n], p]
```

In[553]:=
```
QI = QIMatrix[f, p, n]
```

Out[553]=
```
{{0, 0, 0, 0}, {0, 1, 3, 4}, {3, 0, 3, 1}, {0, 4, 4, 2}}
```

Determine the left nullspace of Q-I and its nullity.

In[554]:=
```
LeftNullspace[QI_, p_] := NullSpace[ Transpose[ QI], Modulus → p]
```

In[555]:=
```
LeftNullspace[QI, p]
```

Out[555]=
```
{{1, 0, 0, 0}}
```

In[556]:=
```
LeftNullity[QI_, p_] := Length[LeftNullspace[QI, p]]
```

In[557]:=
```
LeftNullity[QI, p]
```

Out[557]=
```
1
```

If polynomial f(x) = $u(x)^e$, for irreducible polynomial u(x), then GCD($\frac{\partial f}{\partial x}$, f) = GCD(e $u(x)^{e-1}$u'(x),$u(x)^e$) = u(x) ≥ 1 for irreducible u(x) whenever e ≥2, and 1 otherwise.

In[558]:=
```
IrreducibleFactorExponent[f_, p_] := PolynomialGCD[f, ∂x f, Modulus → p]
```

Our example polynomial f(x) is irreducible, so its exponent = 1.

In[559]:=
```
IrreducibleFactorExponent[f, p]
```

Out[559]=
```
1
```

2. Fast Algorithm for Finding a Primitive Polynomial

Here's a reasonably fast method for testing a polynomial modulo p for primitivity.

In[560]:=
```
IsPrimitive[f_, p_, n_] := Module[
  {r, factors, isPrimitiveRoot = False, a0, a},
  Catch[

   Print["Testing polynomial ", f,
    " factored into irreducibles = ", Factor[f, Modulus → p] ] ;

   (* Factor r into primes. *)

   r = (p^n - 1)/(p - 1);
   factors = FactorInteger[r];
   Print["r = ", r, " number of distinct prime factors = ",
    Length[ factors ], " factors to powers = ", factors ];


   (* Check if (-1)^n a0  is a primitive root of p,
   where a0 is the constant term of f(x). *)

   a0 = ConstantTermOfPolynomial[f];
   Print["a0 = ", a0];

   If[ IsPrimitiveRoot[(-1)^n a0, p],
       Print[ ">>>Const coeff a0 passes primitive root test" ],
       Throw["Not primitive"]];


   (* Check that f(x) has no linear factors. *)

   If[ PolynomialHasLinearFactor[f, p],
       Return[],
```

```
    Print[">>>No linear factors"]
  ];


(* Check that f(x) is irreducible or a power of an irreducible. *)

 Print["Powers of x in Q matrix rows =", PowersOfXModFP[f, p, n]];

 QI = QIMatrix[f, p, n];
 Print["Q - I = ", QI // MatrixForm];

 null = LeftNullity[QI, p];
 Print["Nullity = ", null];

If[null ≤ 1,
   Print[">>>One (possibly repeated) distinct irreducible factor"],
   Throw["Not primitive"]];

If[IrreducibleFactorExponent[f, p] == 1,
 Print[">>>Polynomial is irreducible"],
 Throw["Not primitive"]];


(* Check that x^r (mod f(x), p) = a is an integer. *)

a = PolynomialMod[x^r, {f, p}];

If[PolynomialIsConstant[a],
   Print[">>>Pass test for x^r = a = integer ", a], Return[]];


(* Check if a=(-1)^n a0 (mod p) *)

If[Mod[(-1)^n a0 - a, p] == 0,
    Print[">>>Const coeff test passes"], Return[]];


(* Check x^m≠integer (mod f(x),p) for m∈{r/p_i},
but skip the test for m = r/p_i if p_i | (p-1) *)

numPrimeFactors = Length[factors];

For[i = 1, i ≤ numPrimeFactors, ++i,
```

```
    prime = factors[[i]][[1]];

    m = r / prime;

    If[ Mod[p - 1, prime] == 0,
        Print[ "Skipping test for m = ",
      m, " since p_i = ", prime, " | p-1 = ", p - 1 ],

       Print[ ">>>Testing m = ", m, " prime factor = ", prime ];
        a = PolynomialMod[x^m, {f, p}] ;
       If[ PolynomialIsConstant[a],
          Print[ "Failed the test:  ", a, " is an integer"];
          Throw["Not primitive"],

          Print[ ">>>Pass test for x^m = a = ",
        a, " ≠ integer for m = ", m, " p_i = ", prime];
         ]
      ]
     ]
   Print[">>>Polynomial is primitive." ];
  ]
 ]
```

In[561]:=
```
IsPrimitive[f, p, n]
```

```
Testing polynomial 3 + 2 x + x^2 + x^4
  factored into irreducibles = 3 + 2 x + x^2 + x^4
```

```
r = 156 number of distinct prime factors =
 3 factors to powers = {{2, 2}, {3, 1}, {13, 1}}
```

```
a0 = 3
```

```
>>>Const coeff a0 passes primitive root test
```

```
>>>No linear factors
```

Powers of x in Q matrix rows =
$\{1, 2x + 3x^2 + 4x^3, 3 + 4x^2 + x^3, 4x + 4x^2 + 3x^3\}$

$$Q - I = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 4 \\ 3 & 0 & 3 & 1 \\ 0 & 4 & 4 & 2 \end{pmatrix}$$

Nullity = 1

>>>One (possibly repeated) distinct irreducible factor

>>>Polynomial is irreducible

>>>Pass test for $x^r = a$ = integer 3

>>>Const coeff test passes

Skipping test for m = 78 since $p_i = 2 \mid p-1 = 4$

>>>Testing m = 52 prime factor = 3

>>>Pass test for $x^m = a = 4x + x^2 + 2x^3 \neq$ integer for m = 52 $p_i = 3$

>>>Testing m = 12 prime factor = 13

>>>Pass test for $x^m = a = 3 + 4x + 2x^2 + 4x^3 \neq$ integer for m = 12 $p_i = 13$

>>>Polynomial is primitive.

3. Slow Algorithm for Finding a Primitive Polynomial

The exponentially slow (brute force) method is to try x^k (mod f(x), p) and check we don't get 1 until k = p^n-1. Here are the first few powers,

In[562]:=
```
Table[ PolynomialMod[x^k, {f, p}], {k, 1, 10}]
```

Out[562]=
$$\{x,\ x^2,\ x^3,\ 2+3x+4x^2,\ 2x+3x^2+4x^3,\ 3+2x+3x^2+3x^3,\ 1+2x+4x^2+3x^3,\ 1+4x^2+4x^3,\ 3+3x+x^2+4x^3,\ 3+4x^2+x^3\}$$

Scan through all the powers, and make sure 1 is the last one!

In[563]:=
```
IsPrimitiveSlow[f_, p_, n_] :=
 Module[ {allPowers},
  allPowers = Table[ PolynomialMod[x^k, {f, p}], {k, 1, p^n - 1}];
  For[ k = 1, k ≤ p^n - 1, ++k,
       If[ allPowers[[k]] == 1,
           If[ k < p^n - 1,

      Print[ ">>>Polynomial is NOT primitive x^k=1 for k = ", k ]; Return[],
             Print[ ">>>Polynomial is primitive!" ]]]]
```

In[564]:=
```
IsPrimitiveSlow[f, p, n]
```

```
>>>Polynomial is primitive!
```