Számítógépes számelmélet

Járai Antal

Ezek a programok csak szemléltetésre szolgálnak

▶ 1. A prímek eloszlása, szitálás

▶ 2. Egyszerű faktorizálási módszerek

▶ 3. Egyszerű prímtesztelési módszerek

▶ 4. Lucas-sorozatok

▶ 5. Alkalmazások

▶ 6. Számok és polinomok

▶ 7. Gyors Fourier-transzformáció

▶ 8. Elliptikus függvények

▶ 9. Számolás elliptikus görbéken

▶ 10. Faktorizálás elliptikus gürbékkel

▶ 11. Prímteszt elliptikus görbékkel

▼ 12. Polinomfaktorizálás

```
> restart; with(PolynomialTools);
```

[*CoefficientList*, *CoefficientVector*, *GcdFreeBasis*, *GreatestFactorialFactorization*, *Hurwitz*, *IsSelfReciprocal*, *MinimalPolynomial*, *PDEToPolynomial*, *PolynomialToPDE*, *ShiftEquivalent*, *ShiftlessDecomposition*, *Shorten*, *Shorter*, *Sort*, *Split*, *Splits*, *Translate*] (12.1)

▶ 12.1. Polinomfaktorizálás modulo egy prím.

▼ 12.2. Visszavezetés négyzetmentes esetre.

```
> SquareFree:=proc(a,x,p) local i,out,b,c,y,z,w;
  i:=1; out:=[]; b:=diff(a,x) mod p;
  if b=0 then error "zero derivative; substitute x^p with p";
  fi;
  c:=Gcd(a,b) mod p; w:=Quo(a,c,x) mod p;
  while degree(c)<>0 do
    y:=Gcd(w,c) mod p;
    z:=Quo(w,y,x) mod p;
    out:=[op(out),z];
    i:=i+1;
    w:=y; c:=Quo(c,y,x) mod p;
  od; out:=[c,op(out),w]; end;
```

SquareFree := **proc**(a, x, p) (12.2.1)

```
    local i, out, b, c, y, z, w;
    i := 1;
    out := [ ];
    b := mod(diff(a, x), p);
    if b = 0 then
        error "zero derivative; substitute x^p with p"
    end if;
    c := mod(Gcd(a, b), p);
    w := mod(Quo(a, c, x), p);
    while degree(c) <> 0 do
        y := mod(Gcd(w, c), p);
        z := mod(Quo(w, y, x), p);
        out := [op(out), z];
        i := i + 1;
        w := y;
        c := mod(Quo(c, y, x), p)
    end do;
    out := [c, op(out), w]
end proc
```

```
> `mod`:=mods; x:='x'; a:=x^15-1; debug(SquareFree); SquareFree
```

```
(a,x,5);
```

$$mod := mods$$

$$x := x$$

$$a := x^{15} - 1$$

$$SquareFree$$

```
{--> enter SquareFree, args = x^15-1, x, 5
```

$$i := 1$$

$$out := [\,]$$

$$b := 0$$

```
<-- ERROR in SquareFree (now at top level) = zero derivative;
substitute x^p with p}
Error, (in SquareFree) zero derivative; substitute x^p with p
> SquareFree(a,x,11);
{--> enter SquareFree, args = x^15-1, x, 11
```

$$i := 1$$

$$out := [\,]$$

$$b := 4\,x^{14}$$

$$c := 1$$

$$w := x^{15} - 1$$

$$out := \left[1,\, x^{15} - 1\right]$$

```
<-- exit SquareFree (now at top level) = [1, x^15-1]}
```

$$\left[1,\, x^{15} - 1\right] \tag{12.2.2}$$

```
> SquareFree(x^3+3*x^2+3*x+1,x,11);
{--> enter SquareFree, args = x^3+3*x^2+3*x+1, x, 11
```

$$i := 1$$

$$out := [\,]$$

$$b := 3\,x^2 - 5\,x + 3$$

$$c := x^2 + 2\,x + 1$$

$$w := x + 1$$

$$y := x + 1$$

$$z := 1$$

$$out := [1]$$

$$i := 2$$

$$w := x + 1$$

$$c := x + 1$$

$$y := x + 1$$

$$z := 1$$

$$out := [1, 1]$$

$$i := 3$$

$$w := x + 1$$

$$c := 1$$

$$out := [1, 1, 1, x + 1]$$

```
<-- exit SquareFree (now at top level) = [1, 1, 1, x+1]}
```

$$[1, 1, 1, x + 1] \quad (12.2.3)$$

```
> #
# This procedure find the factorization of the polynomial P
of
# the variable X modulo the prime number p in the form of
product
# P_i^i, where each P_i is square free. The list of P_i's
# is given back.
#

squarefreefactor:=proc(P,X,p) local Pm,D,PP,L,LL,Q,R,i,j;
Pm:=P mod p;
if degree(Pm)<=1 then RETURN([Pm]); fi;
PP:=diff(Pm,X) mod p;
if PP=0 then
  for i from 0 while p*i<=degree(Pm) do PP:=PP+coeff(Pm,X,i*
p)*X^i; od;
  LL:=squarefreefactor(PP,X,p); L:=[];
  for i to nops(LL) do L:=[op(L),0$j=1..p-1,LL[i]]; od;
  RETURN(L);
fi;
D:=Gcd(Pm,PP) mod p; if degree(D)=0 then RETURN([Pm]); fi;
Q:=Quo(Pm,D,X) mod p; PP:=Gcd(Q,D) mod p; Divide(Q,PP,'R')
mod p;
[R,op(squarefreefactor(D,X,p))];
end;
```

squarefreefactor := **proc**(*P*, *X*, *p*) (12.2.4)

  local *Pm*, D, *PP*, *L*, *LL*, *Q*, *R*, *i*, *j*;

  Pm := *mod*(*P*, *p*);

  if *degree*(*Pm*) <= 1 **then**

    RETURN([*Pm*])

  end if;

  PP := *mod*(*diff*(*Pm*, *X*), *p*);

```
    if PP = 0 then
        for i from 0 while p*i <= degree(Pm) do
            PP := PP + coeff(Pm, X, p*i)*X^i
        end do;
        LL := squarefreefactor(PP, X, p);
        L := [];
        for i to nops(LL) do
            L := [op(L), $(0, j = 1 ..p - 1), LL[i]]
        end do;
        RETURN(L)
    end if;
    D := mod(Gcd(Pm, PP), p);
    if degree(D) = 0 then
        RETURN([Pm])
    end if;
    Q := mod(Quo(Pm, D, X), p);
    PP := mod(Gcd(Q, D), p);
    mod(Divide(Q, PP, 'R'), p);
    [R, op(squarefreefactor(D, X, p))]
end proc
```

12.3. Véges testek.

```
> n:=8; RijndaelPoly:=Nextprime(Z^n,Z) mod 2; alpha:=Z;
```

$$n := 8$$

$$RijndaelPoly := Z^8 + Z^4 + Z^3 + Z + 1$$

$$\alpha := Z \qquad (12.3.1)$$

```
> x:=234; xx:=convert(x,base,2); xxx:=add(xx[i]*Z^(i-1),i=1..
  nops(xx));
```

$$x := 234$$

$$xx := [0, 1, 0, 1, 0, 1, 1, 1]$$

$$xxx := Z + Z^3 + Z^5 + Z^6 + Z^7 \qquad (12.3.2)$$

```
> y:=111; yy:=convert(y,base,2); yyy:=add(yy[i]*Z^(i-1),i=1..
```

```
nops(yy));
```

$$y := 111$$

$$yy := [1, 1, 1, 1, 0, 1, 1]$$

$$yyy := 1 + Z + Z^2 + Z^3 + Z^5 + Z^6 \tag{12.3.3}$$

```
> zzz:=modpol(xxx+yyy,RijndaelPoly,Z,2); zz:=CoefficientList
  (zzz,Z);
  z:=add(zz[i]*2^(i-1),i=1..nops(zz));
```

$$zzz := Z^7 + 1 + Z^2$$

$$zz := [1, 0, 1, 0, 0, 0, 0, 1]$$

$$z := 133 \tag{12.3.4}$$

```
> zzz:=modpol(xxx*yyy,RijndaelPoly,Z,2); zz:=CoefficientList
  (zzz,Z);
  z:=add(zz[i]*2^(i-1),i=1..nops(zz));
```

$$zzz := Z^6 + Z^5 + Z^4 + Z^3 + Z^2 + 1$$

$$zz := [1, 0, 1, 1, 1, 1, 1]$$

$$z := 125 \tag{12.3.5}$$

```
> zzz:=modpol(1/xxx,RijndaelPoly,Z,2); zz:=CoefficientList(zzz,
  Z);
  z:=add(zz[i]*2^(i-1),i=1..nops(zz));
```

$$zzz := Z^7 + Z^6 + Z^4 + Z^2 + Z + 1$$

$$zz := [1, 1, 1, 0, 1, 0, 1, 1]$$

$$z := 215 \tag{12.3.6}$$

12.4. Faktorizálás különböző fokú faktorokra.

```
> PartialFactorDD:=proc(a,x,p) local aa,L,aaa,w,i;
  i:=1; w:=x; aa:=a; L:=[];
  while i<=degree(aa)/2 do
    w:=Rem(w^p,aa,x) mod p;
    aaa:=Gcd(aa,w-x) mod p;
    L:=[op(L),aaa];
    if aaa<>1 then
      aa:=Quo(aa,aaa,x) mod p:
      w:=Rem(w,aa,x) mod p;
    fi; i:=i+1;
  od; L:=[op(L),aa]; end;
```

PartialFactorDD := **proc**(*a*, *x*, *p*) (12.4.1)

local *aa*, *L*, *aaa*, *w*, *i*;

i := 1;

```
    w := x;
    aa := a;
    L := [ ];
    while i <= 1 / 2 * degree(aa) do
        w := mod(Rem(w^p, aa, x), p);
        aaa := mod(Gcd(aa, w - x), p);
        L := [op(L), aaa];
        if aaa <> 1 then
            aa := mod(Quo(aa, aaa, x), p);
            w := mod(Rem(w, aa, x), p)
        end if;
        i := i + 1
    end do;
    L := [op(L), aa]
end proc
```

```
> `mod`:=mods; x:='x'; a:=x^15-1; debug(PartialFactorDD);
  PartialFactorDD(a,x,11);
```

$$mod := mods$$

$$x := x$$

$$a := x^{15} - 1$$

$$PartialFactorDD$$

```
{--> enter PartialFactorDD, args = x^15-1, x, 11
```

$$i := 1$$

$$w := x$$

$$aa := x^{15} - 1$$

$$L := [\,]$$

$$w := x^{11}$$

$$aaa := x^5 - 1$$

$$L := [x^5 - 1]$$

$$aa := x^{10} + x^5 + 1$$

$$w := -x^6 - x$$

$$i := 2$$

$$w := x$$

$$aaa := x^{10} + x^5 + 1$$

$$L := [x^5 - 1, x^{10} + x^5 + 1]$$

$$aa := 1$$

$$w := 0$$

$$i := 3$$

$$L := [x^5 - 1, x^{10} + x^5 + 1, 1]$$

```
<-- exit PartialFactorDD (now at top level) = [x^5-1,
x^10+x^5+1, 1]}
```

$$[x^5 - 1, x^{10} + x^5 + 1, 1] \tag{12.4.2}$$

```
> #
  # This procedure find the factorization of the square-free
  # polynomial P of the variable X modulo the prime number p in
  # the form of product P_i, i=0.. where each P_i has only
  degree
  # i irreducible factors. The list of P_i's is given back.
  #

  degreefactor:=proc(P,X,p) local V,W,D,PP,L,j,d;
  V:=P mod p; if degree(V)<1 then RETURN([V]); fi;
  L:=[1]; W:=X;
  for d while 2*d<=degree(V) do
    W:=Powmod(W,p,V,X) mod p; D:=Gcd(W-X,V) mod p;
    if D<>1 then V:=Quo(V,D,X) mod p; fi;
    L:=[op(L),D]; W:=Rem(W,V,X) mod p;
  od;
  if degree(V)=d then RETURN([op(L),V]); fi;
  if degree(V)>d then L:=[op(L),1$j=d..degree(V),V]; fi;
  L; end;
```

degreefactor := **proc**(*P*, *X*, *p*) (12.4.3)

local *V*, *W*, D, *PP*, *L*, *j*, *d*;

$V := mod(P, p)$;

if $degree(V) < 1$ **then**

$RETURN([V])$

end if;

$L := [1]$;

$W := X$;

for *d* **while** $2 * d <= degree(V)$ **do**

$W := mod(Powmod(W, p, V, X), p)$;

$D := mod(Gcd(W - X, V), p)$;

```
        if D <> 1 then
            V:= mod(Quo(V, D, X), p)
        end if;
        L:= [op(L), D];
        W:= mod(Rem(W, V, X), p)
    end do;
    if degree(V) = d then
        RETURN([op(L), V])
    end if;
    if d < degree(V) then
        L:= [op(L), $(1, j = d..degree(V)), V]
    end if;
    L
end proc
```

12.5. Hasítás.

```
> PartialFactorSplit:=proc(a,x,d,p) local t,i;
  t:=rand(); t:=convert(t,base,p); t:=add(t[i]*x^(i-1),i=1..
  nops(t));
  t:=modpol(t,a,x,p); t:=modpol(t^((p^d-1)/2)-1,a,x,p);
  t:=Gcd(t,a) mod p; [t,Quo(a,t,x) mod p]; end;
PartialFactorSplit:= proc(a, x, d, p)                          (12.5.1)
    local t, i;
    t:= rand();
    t:= convert(t, base, p);
    t:= add(t[i]*x^(i - 1), i = 1 ..nops(t));
    t:= modpol(t, a, x, p);
    t:= modpol(t^(1 / 2*p^d - 1 / 2) - 1, a, x, p);
    t:= mod(Gcd(t, a), p);
    [t, mod(Quo(a, t, x), p)]
end proc
> debug(PartialFactorSplit); PartialFactorSplit(x^5-1,x,1,11);
                        PartialFactorSplit
{--> enter PartialFactorSplit, args = x^5-1, x, 1, 11
```

$$t := 386408307450$$

$$t := [0, 4, 6, 3, 6, 4, 9, 6, 9, 9, 3, 1]$$

$$t := 4x + 6x^2 + 3x^3 + 6x^4 + 4x^5 + 9x^6 + 6x^7 + 9x^8 + 9x^9 + 3x^{10} + x^{11}$$

$$t := 3x + x^2 + x^3 + 4x^4 - 4$$

$$t := -2x^4 - 3x^3 + 3x^2 + 3x - 5$$

$$t := x^2 + 2x - 2$$

$$[x^2 + 2x - 2, x^3 - 2x^2 - 5x - 5]$$

```
<-- exit PartialFactorSplit (now at top level) = [x^2+2*x
-2, x^3-2*x^2-5*x-5]}
```

$$[x^2 + 2x - 2, x^3 - 2x^2 - 5x - 5] \tag{12.5.2}$$

```
> expand((x^2+2*x-2)*(x^3-2*x^2-5*x-5)) mod 11;
```

$$x^5 - 1 \tag{12.5.3}$$

```
> PartialFactorSplit(x^2+2*x-2,x,1,11);
  PartialFactorSplit(x^3-2*x^2-5*x-5,x,1,11);
{--> enter PartialFactorSplit, args = x^2+2*x-2, x, 1, 11
```

$$t := 694607189265$$

$$t := [0, 2, 1, 7, 1, 7, 3, 4, 6, 8, 4, 2]$$

$$t := 2x + x^2 + 7x^3 + x^4 + 7x^5 + 3x^6 + 4x^7 + 6x^8 + 8x^9 + 4x^{10} + 2x^{11}$$

$$t := -3x$$

$$t := -1 + 3x$$

$$t := x - 4$$

$$[x - 4, x - 5]$$

```
<-- exit PartialFactorSplit (now at top level) = [x-4, x
-5]}
```

$$[x - 4, x - 5]$$

```
{--> enter PartialFactorSplit, args = x^3-2*x^2-5*x-5, x,
1, 11
```

$$t := 773012980023$$

$$t := [9, 10, 1, 7, 4, 7, 8, 1, 9, 8, 7, 2]$$

$$t := 9 + 10x + x^2 + 7x^3 + 4x^4 + 7x^5 + 8x^6 + x^7 + 9x^8 + 8x^9 + 7x^{10} + 2x^{11}$$

$$t := 3 + 5x - x^2$$

$$t := 2x^2 + 3x - 2$$

$$t := x + 2$$

$$[x + 2, x^2 - 4x + 3]$$

```
<-- exit PartialFactorSplit (now at top level) = [x+2, x^2
-4*x+3]}
```

$$\left[x+2,\, x^2-4\,x+3\right] \tag{12.5.4}$$

```
> expand((x-4)*(x-5)) mod 11; expand((x+2)*(x^2-4*x+3)) mod 11;
```

$$x^2+2\,x-2$$

$$x^3-2\,x^2-5\,x-5 \tag{12.5.5}$$

```
> PartialFactorSplit(x^2-4*x+3,x,1,11);
{--> enter PartialFactorSplit, args = x^2-4*x+3, x, 1, 11
```

$$t := 730616292946$$

$$t := [1, 4, 7, 9, 3, 9, 1, 4, 9, 1, 6, 2]$$

$$t := 1+4\,x+7\,x^2+9\,x^3+3\,x^4+9\,x^5+x^6+4\,x^7+9\,x^8+x^9+6\,x^{10}+2\,x^{11}$$

$$t := -4+5\,x$$

$$t := 0$$

$$t := x^2-4\,x+3$$

$$\left[x^2-4\,x+3,\, 1\right]$$

```
<-- exit PartialFactorSplit (now at top level) = [x^2-4*
x+3, 1]}
```

$$\left[x^2-4\,x+3,\, 1\right] \tag{12.5.6}$$

```
> PartialFactorSplit(x^2-4*x+3,x,1,11);
{--> enter PartialFactorSplit, args = x^2-4*x+3, x, 1, 11
```

$$t := 106507053657$$

$$t := [4, 10, 8, 0, 0, 5, 5, 9, 1, 1, 4]$$

$$t := 4+10\,x+8\,x^2+5\,x^5+5\,x^6+9\,x^7+x^8+x^9+4\,x^{10}$$

$$t := -1+4\,x$$

$$t := -3+x$$

$$t := -3+x$$

$$\left[-3+x,\, x-1\right]$$

```
<-- exit PartialFactorSplit (now at top level) = [-3+x, x
-1]}
```

$$\left[-3+x,\, x-1\right] \tag{12.5.7}$$

```
> expand((x-3)*(x-1)) mod 11;
```

$$x^2-4\,x+3 \tag{12.5.8}$$

```
> PartialFactorSplit(x^10+x^5+1,x,2,11);
```

$$\left[x^6-2\,x^5+3\,x^4+x^3-2\,x^2+4\,x+5,\, x^4+2\,x^3+x^2-5\,x-2\right] \tag{12.5.9}$$

```
> expand((x^6-2*x^5+3*x^4+x^3-2*x^2+4*x+5)*(x^4+2*x^3+x^2-5*x
```

```
-2)) mod 11;
```

$$x^{10}+x^5+1 \quad (12.5.10)$$

```
> PartialFactorSplit(x^6-2*x^5+3*x^4+x^3-2*x^2+4*x+5,x,2,11);
  PartialFactorSplit(x^4+2*x^3+x^2-5*x-2,x,2,11);
```

$$[x^2+3\,x-2,\ x^4-5\,x^3-2\,x^2-3\,x+3]$$

$$[x^4+2\,x^3+x^2-5\,x-2,\ 1] \quad (12.5.11)$$

```
> expand((x^2+3*x-2)*(x^4-5*x^3-2*x^2-3*x+3)) mod 11;
```

$$x^6-2\,x^5+3\,x^4+x^3-2\,x^2+4\,x+5 \quad (12.5.12)$$

```
> PartialFactorSplit(x^4-5*x^3-2*x^2-3*x+3,x,2,11);
  PartialFactorSplit(x^4+2*x^3+x^2-5*x-2,x,2,11);
```

$$[x^4-5\,x^3-2\,x^2-3\,x+3,\ 1]$$

$$[x^2+4\,x+5,\ x^2-2\,x+4] \quad (12.5.13)$$

```
> expand((x^2+4*x+5)*(x^2-2*x+4)) mod 11;
```

$$x^4+2\,x^3+x^2-5\,x-2 \quad (12.5.14)$$

```
> PartialFactorSplit(x^4-5*x^3-2*x^2-3*x+3,x,2,11);
```

$$[1,\ x^4-5\,x^3-2\,x^2-3\,x+3] \quad (12.5.15)$$

```
> PartialFactorSplit(x^4-5*x^3-2*x^2-3*x+3,x,2,11);
```

$$[x^4-5\,x^3-2\,x^2-3\,x+3,\ 1] \quad (12.5.16)$$

```
> PartialFactorSplit(x^4-5*x^3-2*x^2-3*x+3,x,2,11);
```

$$[x^2+x+1,\ x^2+5\,x+3] \quad (12.5.17)$$

```
> expand((x^2+x+1)*(x^2+5*x+3)) mod 11;
```

$$x^4-5\,x^3-2\,x^2-3\,x+3 \quad (12.5.18)$$

```
> #
  # This procedure try to factor a product P of different
  irreducible
  # polynomials of X modulo p each known to be the same degree
  d.
  # The parameter K is the limit for the random tries. The list
  of
  # the found factors is given back.
  #

  fixdegreefactor:=proc(P,X,p,d,K) local T,D,Q,j,i,r;
  if degree(P)<=d then RETURN([P]); fi; r:=rand(p);
  for i to K do T:=0;
    for j to 2*degree(P) do T:=expand(T*X+r()); od;
    T:=Powmod(T,(p^d-1)/2,P,X) mod p;
    D:=Gcd(P,T+1) mod p;
    if degree(D)>0 and degree(D)<degree(P) then
      Q:=Quo(P,D,X) mod p;
```

```
      Q:=fixdegreefactor(Q,X,p,d,K);
      D:=fixdegreefactor(D,X,p,d,K);
      RETURN([op(D),op(Q)]);
    fi;
  od; [P]; end;
```

$$fixdegreefactor := \textbf{proc}(P, X, p, d, K) \tag{12.5.19}$$

local T, D, Q, j, i, r;

if $degree(P) <= d$ **then**

$RETURN([P])$

end if;

$r := rand(p)$;

for i **to** K **do**

$T := 0$;

for j **to** $2 * degree(P)$ **do**

$T := expand(T^{*} X + r(\,))$

end do;

$T := mod(Powmod(T, 1 / 2 * p^{\wedge}d - 1 / 2, P, X), p)$;

$\mathrm{D} := mod(Gcd(P, T + 1), p)$;

if $0 < degree(\mathrm{D})$ **and** $degree(\mathrm{D}) < degree(P)$ **then**

$Q := mod(Quo(P, \mathrm{D}, X), p)$;

$Q := fixdegreefactor(Q, X, p, d, K)$;

$\mathrm{D} := fixdegreefactor(\mathrm{D}, X, p, d, K)$;

$RETURN([op(\mathrm{D}), op(Q)])$

end if

end do;

$[P]$

end proc

```
> #
  # This procedure try to factor a polynomial P of X modulo p.
  # The parameter K is the limit for the random tries.
  # The list of the found factors is given back.
  #

  modfactor:=proc(P,X,p,K) local L,LL,LLL,LLLL,D,j,i,x,y,c,r,
  PP;
  L:=[]; c:=1; r:=rand(p); LL:=squarefreefactor(P,X,p);
  for i to nops(LL) do
```

```
    PP:=LL[i];
    if degree(PP)=0 then c:=c*PP mod p; next; fi;
    LLL:=degreefactor(PP,X,p);
    for j to nops(LLL) do
      PP:=LLL[j];
      if degree(PP)=0 then c:=c*PP mod p; next; fi;
      LLLL:=fixdegreefactor(PP,X,p,j-1,K);
      D:=map(x->degree(x),LLLL);
      if max(op(D))>j-1 then RETURN(FAIL) fi;
      D:=map(proc(x,y) [x,y] end,LLLL,i);
      L:=[op(L),op(D)];
    od;
  od; [c,op(L)]; end;
```

modfactor := **proc**(*P*, *X*, *p*, *K*) (12.5.20)

local *L*, *LL*, *LLL*, *LLLL*, D, *j*, *i*, *x*, *y*, *c*, *r*, *PP*;

L := [ ];

c := 1;

r := *rand*(*p*);

LL := *squarefreefactor*(*P*, *X*, *p*);

for *i* **to** *nops*(*LL*) **do**

PP := *LL*[*i*];

if *degree*(*PP*) = 0 **then**

c := *mod*(*c* * *PP*, *p*);

next

end if;

LLL := *degreefactor*(*PP*, *X*, *p*);

for *j* **to** *nops*(*LLL*) **do**

PP := *LLL*[*j*];

if *degree*(*PP*) = 0 **then**

c := *mod*(*c* * *PP*, *p*);

next

end if;

LLLL := *fixdegreefactor*(*PP*, *X*, *p*, *j* − 1, *K*);

D := *map*(**proc**(*x*)

option *operator*, *arrow*;

degree(*x*)

end proc, *LLLL*);

```
            if j - 1 < max(op(D)) then
                RETURN(FAIL)
            end if;
            D := map(proc(x, y)
                [x, y]
            end proc, LLLL, i);
            L := [op(L), op(D)]
        end do
    end do;
    [c, op(L)]
end proc
```

▶ 13. Az AKS-teszt

▶ 14. A szita módszerek alapjai

▶ 15. Számtest szita

▶ 16. Vegyes problémák