# 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

```
> restart;
```

### ▼ 9.1. Elliptikus görbék.

```
> #
  # This routine randomly choose an elliptic "curve" modulo n,
  # where gcd(n,6)=1. The coordinates x,y are choosen
  # randomly, the parameter a too, and b is calculated.
  # The list [x,y,a,b] is given back or a divisor d of n.
  #

  ellrand:=proc(n) local x,y,a,b,r,d,f;
  r:=rand(n); d:=0;
  while d=0 do
    x:=r(n); y:=r(n); a:=r(n); b:=y^2-x^3-a*x mod n;
```

```
    d:=4*a^3+27*b^2 mod n; gcd(d,n);
  od;
  if %<n and %>1 then return % fi;
  [x,y,a,b]; end;
```

$$ellrand := \mathbf{proc}(n) \tag{9.1.1}$$

```
 local x, y, a, b, r, d, f;
 r:= rand(n);
 d:= 0;
 while d = 0 do
     x:= r(n);
     y:= r(n);
     a:= r(n);
     b:= mod(y^2 - x^3 - a*x, n);
     d:= mod(4*a^3 + 27*b^2, n);
     gcd(d, n)
 end do;
 if % < n and 1 < % then
     return %
 end if;
 [x, y, a, b]
end proc
```

```
> ellrand(97); ellrand(97); ellrand(97); ellrand(97); ellrand
  (97);
```

$$[5, 58, 43, 17]$$
$$[37, 68, 26, 54]$$
$$[95, 16, 89, 54]$$
$$[33, 17, 51, 14]$$
$$[55, 42, 82, 47] \tag{9.1.2}$$

## 9.2. Hasse tétele.

```
> #
  # This brute force procedure calculate the number of points
  # on an elliptic curve modulo p>3, a prime. The curve
  # parameters are a, b.
  #
```

```
ellcount:=proc(a,b,p) local x,c;
c:=1;
for x from 0 to p-1 do c:=c+numtheory[jacobi](x^3+a*x+b,p)+1;
od;
c; end;
```

$$ellcount := \mathbf{proc}(a, b, p) \tag{9.2.1}$$

$$\mathbf{local}\ x, c;$$

$$c := 1;$$

$$\mathbf{for}\ x\ \mathbf{from}\ 0\ \mathbf{to}\ p - 1\ \mathbf{do}$$

$$c := c + numtheory[jacobi](x\text{^}3 + a^{*}x + b, p) + 1$$

$$\mathbf{end\ do};$$

$$c$$

$$\mathbf{end\ proc}$$

```
> ellrand(97); ellcount(%[3],%[4],97);
  ellrand(97); ellcount(%[3],%[4],97);
  ellrand(97); ellcount(%[3],%[4],97);
  ellrand(97); ellcount(%[3],%[4],97);
  ellrand(97); ellcount(%[3],%[4],97);
```

$$[59, 92, 13, 4]$$

$$88$$

$$[49, 46, 7, 39]$$

$$115$$

$$[45, 43, 8, 89]$$

$$112$$

$$[76, 58, 15, 39]$$

$$105$$

$$[0, 69, 76, 8]$$

$$97 \tag{9.2.2}$$

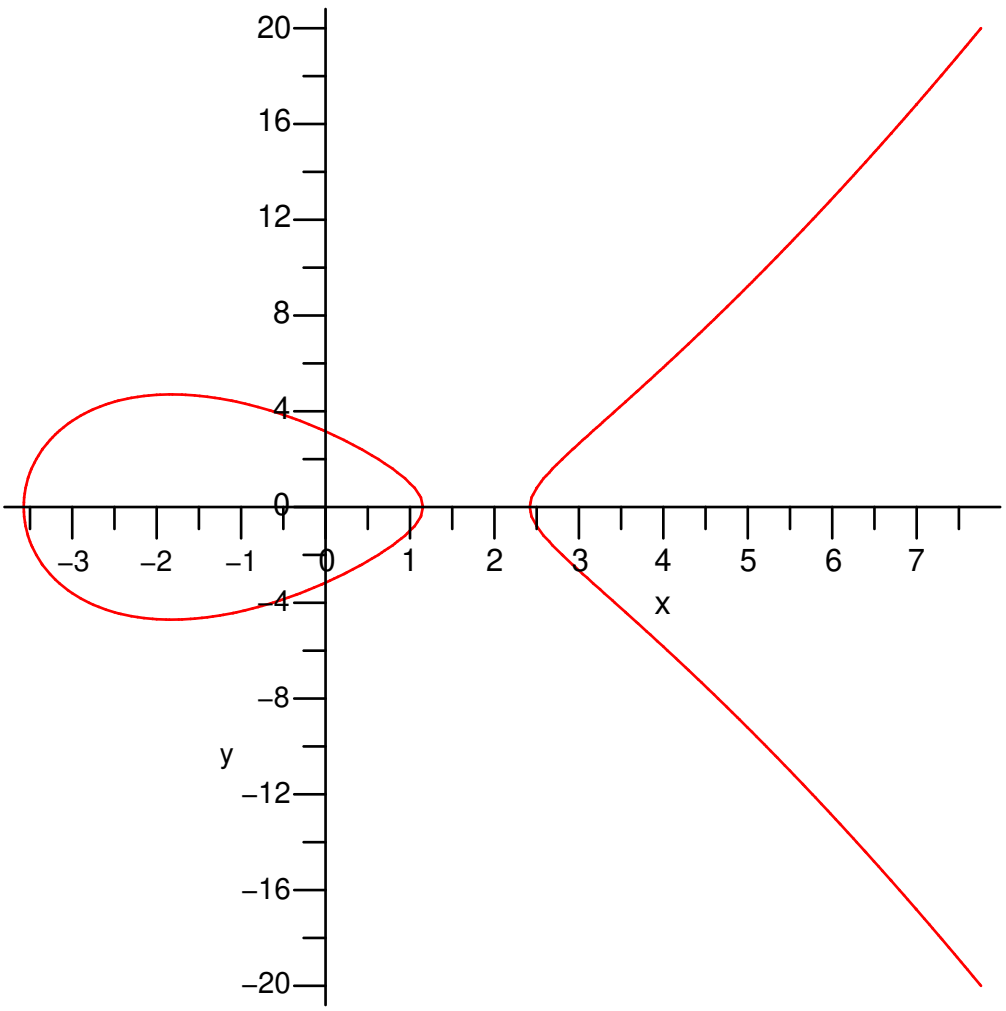
## 9.3. Gyakorlat.

```
> with(plots);
```

$$[Interactive, animate, animate3d, animatecurve, arrow, changecoords, \tag{9.3.1}$$
$$complexplot, complexplot3d, conformal, conformal3d, contourplot,$$
$$contourplot3d, coordplot, coordplot3d, cylinderplot, densityplot,$$
$$display, display3d, fieldplot, fieldplot3d, gradplot, gradplot3d,$$
$$graphplot3d, implicitplot, implicitplot3d, inequal, interactive,$$

*interactiveparams, listcontplot, listcontplot3d, listdensityplot, listplot, listplot3d, loglogplot, logplot, matrixplot, multiple, odeplot, pareto, plotcompare, pointplot, pointplot3d, polarplot, polygonplot, polygonplot3d, polyhedra_supported, polyhedraplot, replot, rootlocus, semilogplot, setoptions, setoptions3d, spacecurve, sparsematrixplot, sphereplot, surfdata, textplot, textplot3d, tubeplot*]

```
> implicitplot(y^2=x^3-10*x+10,x=-5..10,y=-20..20,numpoints=
  10000);
```




## 9.4. Gyakorlat.

```
> #
  # Doubling on an elliptic "curve" modulo n, where gcd(n,6)=1.
  # P is the point to double and a, b are the parameters.
  # The return value is the double of the point P or
  # a divisor d of n.
```

```
#

elldou:=proc(P,a,b,n) local l,d;
if P[3]=0 then return P fi;
if P[2]=0 then return [0,1,0] fi;
d:=igcdex(2*P[2],n,'l');
if 1<d and d<n then return d fi;
l:=(3*P[1]^2+a)*l mod n;
l^2-2*P[1] mod n;
[%,l*(P[1]-%)-P[2] mod n,1];
end;
```

$$elldou := \mathbf{proc}(P, a, b, n) \tag{9.4.1}$$

```
local l, d;
if P[3] = 0 then
    return P
end if;
if P[2] = 0 then
    return [0, 1, 0]
end if;
d := igcdex(2 * P[2], n, 'l');
if 1 < d and d < n then
    return d
end if;
l := mod((3 * P[1]^2 + a) * l, n);
mod(l^2 - 2 * P[1], n);
[%, mod(l * (P[1] - %) - P[2], n), 1]
end proc
```

```
> #
  # Addition on an elliptic "curve" modulo n, where gcd(n,6)=1.
  # P and Q are the points to add and a,b are the parameters.
  # The return value is the sum of the points or a divisor d of
  n.
  #

  elladd:=proc(P,Q,a,b,n) local l,d;
  if P[3]=0 then return Q fi;
  if Q[3]=0 then return P fi;
  if P=Q then return elldou(P,a,b,n) fi;
  if P[1]=Q[1] then return [0,1,0] fi;
  d:=igcdex(P[1]-Q[1],n,'l');
```

```
   if 1<d and d<n then return d fi;
   l:=(P[2]-Q[2])*l mod n;
   l^2-P[1]-Q[1] mod n;
   [%,l*(P[1]-%)-P[2] mod n,1];
   end;
```

$$elladd := \mathbf{proc}(P, Q, a, b, n) \tag{9.4.2}$$

```
local l, d;
if P[3] = 0 then
    return Q
end if;
if Q[3] = 0 then
    return P
end if;
if P = Q then
    return elldou(P, a, b, n)
end if;
if P[1] = Q[1] then
    return [0, 1, 0]
end if;
d := igcdex(P[1] - Q[1], n, 'l');
if 1 < d and d < n then
    return d
end if;
l := mod((P[2] - Q[2]) * l, n);
mod(l^2 - P[1] - Q[1], n);
[%, mod(l * (P[1] - %) - P[2], n), 1]
end proc
```

```
> n:= 97; P:=[59,92,1]; a:=13; b:=4;
```

$$n := 97$$

$$P := [59, 92, 1]$$

$$a := 13$$

$$b := 4 \tag{9.4.3}$$

```
> elldou(P,a,b,n); elldou(%,a,b,n); elldou(%,a,b,n);
```

$$[80, 60, 1]$$

$$[77, 59, 1]$$

$$[67, 29, 1] \tag{9.4.4}$$

```
> elladd(P,P,a,b,n); elladd(%,P,a,b,n); elladd(%,P,a,b,n);
  elladd(%,P,a,b,n);
```

$$[80, 60, 1]$$
$$[67, 68, 1]$$
$$[77, 59, 1]$$
$$[91, 96, 1] \tag{9.4.5}$$

## ▶ 10. Faktorizálás elliptikus görbékkel

## ▶ 11. Prímteszt elliptikus görbékkel

## ▶ 12. Polinomfaktorizálás

## ▶ 13. Az AKS-teszt

## ▶ 14. A szita módszerek alapjai