Számítógépes számelmélet

Járai Antal

Ezek a programok csak szemléltetésre szolgálnak

1. A prímek eloszlása, szitálás

```
> restart; with(numtheory);
```

[*GIgcd*, *bigomega*, *cfrac*, *cfracpol*, *cyclotomic*, *divisors*, *factorEQ*, *factorset*, *fermat*, *imagunit*, *index*, *integral_basis*, *invcfrac*, *invphi*, *issqrfree*, *jacobi*, *kronecker*, λ, *legendre*, *mcombine*, *mersenne*, *migcdex*, *minkowski*, *mipolys*, *mlog*, *mobius*, *mroot*, *msqrt*, *nearestp*, *nthconver*, *nthdenom*, *nthnumer*, *nthpow*, *order*, *pdexpand*, ϕ, π, *pprimroot*, *primroot*, *quadres*, *rootsunity*, *safeprime*, σ, *sq2factor*, *sum2sqr*, τ, *thue*] (1.1)

1.1. A prímszámtétel.

```
> [i$i=1..20]; evalf(map(i->log[2](mersenne([i])+1),%));
```

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

[2., 3., 5., 7., 13., 17., 19., 31., 61., 89., 107., 127., 521., 607., 1279., 2203., 2281., 3217., 4253., 4423.] (1.1.1)

1.2. Kérdés: zeta gyökei.

1.3. Kérdés: π(x).

1.4. Ikerprímek.

1.5. Kérdés: π_2(x)

1.6. Kérdés: az ikerprímek reciprokainak összege.

1.7. Sejtés.

```
> #
  # This procedure approximate Cs calculating the product
```

```
# for primes below x.
#

Cs:=proc(s::posint,x::posint) local P,p;
P:=1.; p:=nextprime(s);
while p<x do P:=P*(1-s/p)/(1-1/p)^s; p:=nextprime(p) od;
P end;
```

```
Cs := proc(s::posint, x::posint)
    local P, p;
    P := 1.;
    p := nextprime(s);
    while p < x do
        P := P* (1 - s / p) / (1 - 1 / p)^s;
        p := nextprime(p)
    end do;
    P
end proc
```
(1.7.1)

```
> Cs(2,10); Cs(2,100); Cs(2,1000); Cs(2,10000); Cs(2,100000);
Cs(2,1000000);
```

0.6835937498

0.6613770846

0.6602457447

0.6601682974

0.6601623428

0.6601618366 (1.7.2)

1.8. Példa.

```
> f1:=h->(3.+30*h)*2.^38880.-1;
f2:=f1+2; f2(0);
g:=h->1/ln(f1(h))/ln(f2(h));
```

$$f1 := h \rightarrow (3. + 30\, h)\, 2.^{38880.} - 1$$

$$f2 := f1 + 2$$

$$3.336972828\ 10^{11704}$$

$$g := h \rightarrow \frac{1}{\ln(f1(h))\ \ln(f2(h))} \quad (1.8.1)$$

```
> 2.^27/6*(g(0)+4*g(2.^26)+g(2.^27));
```

(1.8.2)

$$0.1845532659 \tag{1.8.2}$$

```
> Cf1f2:=C2*(1-1/3)^2/(1-2/3)*(1-1/5)^2/(1-2/5)/(1-1/2)^2/(1
-1/3)^2/(1-1/5)^2;
```

$$Cf1f2 := 20\, C2 \tag{1.8.3}$$

```
> %%*20*0.66016;
```

$$2.436693680 \tag{1.8.4}$$

```
> f1:=h->(5775.+30030*h)*2.^171960.-1;
  f2:=f1+2; f2(0);
  g:=h->1/ln(f1(h))/ln(f2(h));
```

$$f1 := h \rightarrow (5775. + 30030\, h)\, 2.^{1.71960\, 10^5} - 1$$

$$f2 := f1 + 2$$

$$7.578903313\, 10^{51768}$$

$$g := h \rightarrow \frac{1}{\ln(f1(h))\, \ln(f2(h))} \tag{1.8.5}$$

```
> 2.^33/6*(g(0)+4*g(2.^32)+g(2.^33));
```

$$0.6043317724 \tag{1.8.6}$$

```
> C2*(1-1/3)^2/(1-2/3)*(1-1/5)^2/(1-2/5)*(1-1/7)^2/(1-2/7)*(1
-1/11)^2/(1-2/11)*(1-1/13)^2/(1-2/13);
```

$$\frac{16384}{11011}\, C2 \tag{1.8.7}$$

```
> Cf1f2:=%/(1-1/2)^2/(1-1/3)^2/(1-1/5)^2/(1-1/7)^2/(1-1/11)^2/
(1-1/13)^2;
```

$$Cf1f2 := \frac{364}{9}\, C2 \tag{1.8.8}$$

```
> %*%%%; subs(C2=0.66016,%);
```

$$24.44186279\, C2$$

$$16.13554014 \tag{1.8.9}$$

▶ 1.9. Kérdés.

▼ 1.10. Eratosztenész szitája.

```
> sieve:=proc(N::posint) local n,B,i,j;
  n:=floor((N-1)/2);
  B:=Array(0..n-1);
  for j from 0 to n-1 do B[j]:=1 od;
  j:=0;
  while j<n do
    while B[j]=0 do j:=j+1 od;
    i:=2*j^2+6*j+3;
```

```
    if i>=n then break fi;
    while i<n do B[i]:=0; i:=i+2*j+3 od;
    j:=j+1;
  od; B; end;
```

$sieve := \textbf{proc}(N::posint)$ (1.10.1)

```
    local n, B, i, j;
    n := floor(1/2*N - 1/2);
    B := Array(0..n - 1);
    for j from 0 to n - 1 do
        B[j] := 1
    end do;
    j := 0;
    while j < n do
        while B[j] = 0 do
            j := j + 1
        end do;
        i := 2*j^2 + 6*j + 3;
        if n <= i then
            break
        end if;
        while i < n do
            B[i] := 0;
            i := i + 2*j + 3
        end do;
        j := j + 1
    end do;
    B
end proc
```

```
> debug(sieve); sieve(21);
```

sieve

```
{--> enter sieve, args = 21
```

$n := 10$

$B := Array(0..9, \{\}, datatype = anything, storage = rectangular, order = Fortran\_order)$

$$B_0 := 1$$
$$B_1 := 1$$
$$B_2 := 1$$
$$B_3 := 1$$
$$B_4 := 1$$
$$B_5 := 1$$
$$B_6 := 1$$
$$B_7 := 1$$
$$B_8 := 1$$
$$B_9 := 1$$
$$j := 0$$
$$i := 3$$
$$B_3 := 0$$
$$i := 6$$
$$B_6 := 0$$
$$i := 9$$
$$B_9 := 0$$
$$i := 12$$
$$j := 1$$
$$i := 11$$

$Array(0..9, \{0 = 1, 1 = 1, 2 = 1, 4 = 1, 5 = 1, 7 = 1, 8 = 1\},$
$datatype = anything, storage = rectangular, order = Fortran\_order)$

```
<-- exit sieve (now at top level) = Array(0..9, {(1) = 1,
(2) = 1, (3) = 1, (4) = 0, (5) = 1, (6) = 1, (7) = 0, (8)
= 1, (9) = 1})}
```

$Array(0..9, \{0 = 1, 1 = 1, 2 = 1, 4 = 1, 5 = 1, 7 = 1, 8 = 1\},$ (1.10.2)
$datatype = anything, storage = rectangular, order = Fortran\_order)$

```
> undebug(sieve); sieve(10000);
```

sieve

(1.10.3)

$$\begin{bmatrix} 0 \,..\, 4998\ Array \\ Data\ Type\text{: } anything \\ Storage\text{: } rectangular \\ Order\text{: } Fortran\_order \end{bmatrix} \qquad (1.10.3)$$

1.11. Feladat.

1.12. Moduláris inverz euklidészi algoritmussal.

```
> #
  # Calculation of the greatest common
  # divisor by the Euclidean algorithm.
  #

  eucgcd:=proc(x::integer,y::integer) local u,v,r;
  u:=abs(x); v:=abs(y);
  while v<>0 do r:=irem(u,v); u:=v; v:=r od;
  u end;
```

eucgcd := **proc**(*x*::*integer*, *y*::*integer*) (1.12.1)

```
    local u, v, r;
    u := abs(x);
    v := abs(y);
    while v <> 0 do
        r := irem(u, v);
        u := v;
        v := r
    end do;
    u
end proc
```

```
> modinvdiv:=proc(a::integer,m::integer) local x1,x2,x3,d1,d2,
  d3,q,p;
  x1:=1; d1:=a; x2:=0; d2:=m; p:=0;
  do
    if d2=0 then
      if p=0 then return [x1,d1]
      elif x1=0 then return [x1,d1]
      else return [m-x1,d1]
      fi;
```

```
    fi;
    q:=iquo(d1,d2); d3:=d1-q*d2; x3:=x1+q*x2; p:=1-p;
    x1:=x2; x2:=x3; d1:=d2; d2:=d3;
  od; end;
```

$modinvdiv := \mathbf{proc}(a::integer, m::integer)$ (1.12.2)

```
    local x1, x2, x3, d1, d2, d3, q, p;
    x1 := 1;
    d1 := a;
    x2 := 0;
    d2 := m;
    p := 0;
    do
        if d2 = 0 then
            if p = 0 then
                return [x1, d1]
            elif x1 = 0 then
                return [x1, d1]
            else
                return [m − x1, d1]
            end if
        end if;
        q := iquo(d1, d2);
        d3 := d1 − q * d2;
        x3 := x1 + q * x2;
        p := 1 − p;
        x1 := x2;
        x2 := x3;
        d1 := d2;
        d2 := d3
    end do
end proc
```

```
> modinvdiv(13874,15543);
```

$$[8903, 1] \quad (1.12.3)$$

1.13. Feladat.

1.14. Moduláris inverz bináris lnko algoritmussal.

```
> #
  # Calculation of the greatest common
  # divisor by the binary algorithm.
  #

  bingcd:=proc(x::integer,y::integer) local u,v,k,t;
  u:=abs(x); v:=abs(y);
  if u=0 then RETURN(v) fi;
  if v=0 then RETURN(u) fi;
  k:=0;
  while type(u,even) and type(v,even) do k:=k+1; u:=u/2; v:=v/2
  od;
  if type(u,odd) then t:=-v else t:=u fi;
  while t<>0 do
    while type(t,even) do t:=t/2 od;
    if t>0 then u:=t else v:=-t fi;
    t:=u-v;
  od; u*2^k end;
```

bingcd := **proc**(*x*::*integer*, *y*::*integer*) (1.14.1)

local *u*, *v*, *k*, *t*;

$u := \mathrm{abs}(x)$;

$v := \mathrm{abs}(y)$;

if $u = 0$ **then**

RETURN(*v*)

end if;

if $v = 0$ **then**

RETURN(*u*)

end if;

$k := 0$;

while *type*(*u*, *even*) **and** *type*(*v*, *even*) **do**

$k := k + 1$;

$u := 1 / 2 * u$;

$v := 1 / 2 * v$

end do;

```
if type(u, odd) then
    t := -v
else
    t := u
end if;
while t <> 0 do
    while type(t, even) do
        t := 1 / 2 * t
    end do;
    if 0 < t then
        u := t
    else
        v := -t
    end if;
    t := u - v
end do;
u * 2^k
end proc
```

1.15. Feladat.

```
> oddmodinvbin:=proc(a::nonnegint,m::posint)
  local x1,x2,x3,d1,d2,d3,p;
  if not type(m,odd) then return FAIL fi;
  if m=1 then return [0,1] fi;
  x1:=1; d1:=a mod m; x2:=m; d2:=m;
  if type(d1,even) then x3:=0; d3:=m; p:=1 else x3:=1; d3:=d1;
  p:=0 fi;
  while d3<>0 do
    while type(d3,even) do d3:=d3/2;
      if type(x3,even) then x3:=x3/2 else x3:=(x3+m)/2 fi;
    od;
    if p=0 then x1:=x3; d1:=d3 else x2:=m-x3; d2:=d3 fi;
    if x1>=x2 then x3:=x1-x2 else x3:=m+x1-x2 fi;
    if d1>=d2 then d3:=d1-d2; p:=0 else d3:=d2-d1; p:=1 fi;
  od; [x1,d1] end;
```

oddmodinvbin := proc(a::nonnegint, m::posint) (1.15.1)

```
local x1, x2, x3, d1, d2, d3, p;
```

```
if not type(m, odd) then
    return FAIL
end if;
if m = 1 then
    return [0, 1]
end if;
x1 := 1;
d1 := mod(a, m);
x2 := m;
d2 := m;
if type(d1, even) then
    x3 := 0;
    d3 := m;
    p := 1
else
    x3 := 1;
    d3 := d1;
    p := 0
end if;
while d3 <> 0 do
    while type(d3, even) do
        d3 := 1 / 2 * d3;
        if type(x3, even) then
            x3 := 1 / 2 * x3
        else
            x3 := 1 / 2 * x3 + 1 / 2 * m
        end if
    end do;
    if p = 0 then
        x1 := x3;
        d1 := d3
    else
```

```
        x2 := m - x3;
        d2 := d3
      end if;
      if x2 <= x1 then
        x3 := x1 - x2
      else
        x3 := m + x1 - x2
      end if;
      if d2 <= d1 then
        d3 := d1 - d2;
        p := 0
      else
        d3 := d2 - d1;
        p := 1
      end if
    end do;
    [x1, d1]
  end proc
```

```
> oddmodinvbin(13874,15543);
```

$$[8903, 1] \tag{1.15.2}$$

1.16. Általános szita.

1.17. Programozási problémák.

```
> #
  # This procedure calculate the sum of the reciprocal
  # of primes up to x and compare with ln(ln(x)).
  #

  sumprimerec:=proc(x) local s,p,i;
  s:=0.; p:=2;
  while p<x do
    s:=evalf(s+1/p); p:=nextprime(p)
  od; [s,evalf(s-ln(ln(x)))] end;
```

$$sumprimerec := \mathbf{proc}(x) \tag{1.17.1}$$

```
    local s, p, i;
```

```
    s := 0.;
    p := 2;
    while p < x do
        s := evalf(s + 1 / p);
        p := nextprime(p)
    end do;
    [s, evalf(s - ln(ln(x)))]
end proc
```

```
> sumprimerec(10); sumprimerec(100); sumprimerec(1000);
  sumprimerec(10000); sumprimerec(100000); sumprimerec(1000000)
  ;
```

[1.176190476, 0.3421580307]
[1.802817203, 0.275637577]
[2.198080131, 0.265435397]
[2.483059958, 0.262733152]
[2.705272178, 0.261801821]
[2.887328140, 0.261536225] (1.17.2)

1.18. A szitálás dúsító hatása.

```
> #
  # This procedure calculate the factor qsAB.
  #

  qsAB:=proc(s::posint,A::posint,B::posint) local P,p;
  P:=1.; p:=nextprime(A-1);
  while p<B do P:=P*(1-s/p); p:=nextprime(p) od;
  P end;
```

```
qsAB := proc(s::posint, A::posint, B::posint)          (1.18.1)
    local P, p;
    P := 1.;
    p := nextprime(A - 1);
    while p < B do
        P := P* (1 - s / p);
        p := nextprime(p)
    end do;
```

```
    P
end proc
> qsAB(1,1,100);
```

$$0.1203172905 \tag{1.18.2}$$

```
> B:=10: qsAB(1,1,B);
  B:=100: qsAB(1,1,B);
  B:=1000: qsAB(1,1,B);
  B:=10000: qsAB(1,1,B);
  B:=100000: qsAB(1,1,B);
  B:=1000000: qsAB(1,1,B);
```

$$0.2285714285$$

$$0.1203172905$$

$$0.08096526349$$

$$0.06088469238$$

$$0.04875291757$$

$$0.04063820997 \tag{1.18.3}$$

▼ 1.19. Példa.

```
> qsAB(2,7,1000000);
```

$$0.02180467265 \tag{1.19.1}$$

```
> %*(ln(1000000.)/ln(44000.*2^25))^2;
```

$$0.005300634160 \tag{1.19.2}$$

▶ 1.20. Kérdés.

▶ 1.21. Kérdés.

▶ 1.22. Kérdés.

▶ 1.23. Kérdés.

▶ 1.24. Kérdés.

▶ 1.25. Kérdés.

▶ 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 görbékkel
- 11. Prímteszt elliptikus görbékkel
- 12. Polinomfaktorizálás
- 13. Az AKS teszt
- 14. A szita módszerek alapjai