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

```
> restart;
```

6.1. Összehasonlítás, összeadás, kivonás.

```
> #
  # We fix a base. For demo purposes:
  #

  B:=10;
```

$$B := 10 \tag{6.1.1}$$

```
> `mod`:=modp;
```

$$mod := modp \tag{6.1.2}$$

```
> #
  # Carry addition to long number s
  #

  cadd:=proc(s,c) local i,r,cc,x; global B;
  r:=[]; cc:=c;
  for i from 1 to nops(s) do
    x:=s[i]+cc mod B;
    cc:=(s[i]+cc-x)/B;
    r:=[op(r),x]
```

```
   od; [r,cc] end;
```

$$cadd := \mathbf{proc}(s, c) \tag{6.1.3}$$

local *i, r, cc, x*;
global *B*;
$r := [\,]$;
$cc := c$;
for *i* **to** $nops(s)$ **do**
$x := mod(s[i] + cc, B)$;
$cc := (s[i] + cc - x) / B$;
$r := [op(r), x]$
end do;
$[r, cc]$
end proc

```
> #
  # Carry subtraction from long number s
  #

  csub:=proc(s,c) local i,r,cc,x; global B;
  r:=[]; cc:=c;
  for i from 1 to nops(s) do
    x:=s[i]-cc mod B;
    cc:=-(s[i]-cc-x)/B;
    r:=[op(r),x]
  od; [r,cc] end;
```

$$csub := \mathbf{proc}(s, c) \tag{6.1.4}$$

local *i, r, cc, x*;
global *B*;
$r := [\,]$;
$cc := c$;
for *i* **to** $nops(s)$ **do**
$x := mod(s[i] - cc, B)$;
$cc := -(s[i] - cc - x) / B$;
$r := [op(r), x]$
end do;
$[r, cc]$
end proc

```
> ss:=convert(100,base,B); csub(ss,2);
```

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

$$[[8, 9, 0], 0] \qquad (6.1.5)$$

```
> #
  # Comparision of long numbers
  #

  cmp:=proc(s1,s2) local i;
  if nops(s1)>nops(s2) then RETURN(1) fi;
  if nops(s1)<nops(s2) then RETURN(-1) fi;
  for i from nops(s1) by -1 to 1 do
    if s1[i]>s2[i] then RETURN(1) fi;
    if s1[i]<s2[i] then RETURN(-1) fi;
  od; 0 end;
```

cmp := **proc**(s1, s2) (6.1.6)

```
    local i;
    if nops(s2) < nops(s1) then
        RETURN(1)
    end if;
    if nops(s1) < nops(s2) then
        RETURN(-1)
    end if;
    for i from nops(s1) by -1 to 1 do
        if s2[i] < s1[i] then
            RETURN(1)
        end if;
        if s1[i] < s2[i] then
            RETURN(-1)
        end if
    end do;
    0
end proc
```

```
> s1:=convert(123,base,B); s2:=convert(321,base,B); cmp(s1,s2);
```

$$s1 := [3, 2, 1]$$

$$s2 := [1, 2, 3]$$

$$-1 \qquad (6.1.7)$$

```
> #
  # Addition with carry, n digit
  #

  addc:=proc(s1,s2,c,n) local i,r,cc,x; global B;
  r:=[]; cc:=c;
  for i from 1 to n do
    x:=s1[i]+s2[i]+cc mod B;
    cc:=(s1[i]+s2[i]+cc-x)/B;
    r:=[op(r),x]
  od; [r,cc]; end;
```

addc := **proc**(*s1*, *s2*, *c*, *n*) (6.1.8)

```
    local i, r, cc, x;
    global B;
    r := [];
    cc := c;
    for i to n do
        x := mod(s1[i] + s2[i] + cc, B);
        cc := (s1[i] + s2[i] + cc - x) / B;
        r := [op(r), x]
    end do;
    [r, cc]
end proc
```

```
> addc(s,s1,2,3);
```

$$[[4, 2, 1], 1] \quad (6.1.9)$$

```
> #
  # Subtraction with carry, n digit
  #

  subc:=proc(s1,s2,c,n) local i,r,cc,x;
  r:=[]; cc:=c;
  for i from 1 to n do
    x:=s1[i]-s2[i]-cc mod B;
    cc:=-(s1[i]-s2[i]-cc-x)/B;
    r:=[op(r),x]
  od; [r,cc] end;
```

subc := **proc**(*s1*, *s2*, *c*, *n*) (6.1.10)

```
    local i, r, cc, x;
    r := [];
    cc := c;
```

for i **to** n **do**

$x := mod(s1[i] - s2[i] - cc, B);$

$cc := -(s1[i] - s2[i] - cc - x) / B;$

$r := [op(r), x]$

end do;

$[r, cc]$

end proc

```
> subc(s1,s2,2,3);
```

$$[[0, 0, 8], 1] \quad (6.1.11)$$

6.2. Szorzás és polinomszorzás.

```
> x:='x'; i:='i';
s1:=convert(123,base,B); p1:=add(s1[i]*x^(i-1),i=1..nops(s1))
;
s2:=convert(321,base,B); p2:=add(s2[i]*x^(i-1),i=1..nops(s2))
;
convert(123*321,base,B); expand(p1*p2);
convert(1002003*3002001,base,B^3);
```

$$x := x$$

$$i := i$$

$$s1 := [3, 2, 1]$$

$$p1 := 3 + 2\,x + x^2$$

$$s2 := [1, 2, 3]$$

$$p2 := 1 + 2\,x + 3\,x^2$$

$$[3, 8, 4, 9, 3]$$

$$3 + 8\,x + 14\,x^2 + 8\,x^3 + 3\,x^4$$

$$[3, 8, 14, 8, 3] \quad (6.2.1)$$

6.3. Klasszikus algoritmusok a szorzásra és az osztásra.

```
>
```

6.4. Karacuba szorzás.

```
> #
# Single digit multiplication
#
```

```
mul11:=proc(x,y) global B;
convert(x[1]*y[1],base,B);
[%[1],%[2]] end;
```

mul11 := **proc**(x, y) (6.4.1)

global B;

convert(x[1]*y[1], base, B);

[%[1], %[2]]

end proc

```
> mul11([2],[7]);
```

[4, 1] (6.4.2)

```
> #
  # Karatsuba's method; x and y are lists of nonnegative
  # digits with lenght 2^n.
  #

  kara:=proc(x,y,n)
  local m0,m00,m01,m1,m10,m11,m2,m20,m21,f,x0,x1,y0,y1,z0,z1,
  c0,c1;
  if n=0 then RETURN(mul11(x,y)) fi;
  x0:=[x[1..2^(n-1)]];
  x1:=[x[1+2^(n-1)..2^n]];
  y0:=[y[1..2^(n-1)]];
  y1:=[y[1+2^(n-1)..2^n]];
  m0:=kara(x0,y0,n-1);
  m00:=[m0[1..2^(n-1)]];
  m01:=[m0[1+2^(n-1)..2^n]];
  m2:=kara(x1,y1,n-1);
  m20:=[m2[1..2^(n-1)]];
  m21:=[m2[1+2^(n-1)..2^n]];
  f:=true;
  if cmp(x1,x0,2^(n-1))>=0 then
    z0:=subc(x1,x0,0,2^(n-1))[1]
  else
    z0:=subc(x0,x1,0,2^(n-1))[1];
    f:=not f;
  fi;
  if cmp(y1,y0,2^(n-1))>=0 then
    z1:=subc(y1,y0,0,2^(n-1))[1];
  else
    z1:=subc(y0,y1,0,2^(n-1))[1];
    f:=not f;
  fi;
    m1:=kara(z0,z1,n-1);
    m10:=[m1[1..2^(n-1)]];
    m11:=[m1[1+2^(n-1)..2^n]];
```

```
    z0:=addc(m01,m20,0,2^(n-1));
    c0:=z0[2];
    z1:=addc(z0[1],m00,0,2^(n-1));
    c1:=c0+z1[2];
    z0:=addc(z0[1],m21,c1,2^(n-1));
    c0:=c0+z2[2];
    m21:=cadd(z0[1],m21,c0,2^(n-1));
  if f then
    z1:=subc(z1[1],m10,0,2^(n-1));
    c1:=z1[2];
    z0:=subc(z0[1],m11,c1,2^(n-1));
    c0:=z0[2];
    m21:=csub(m21[2],c2);
  else
    z1:=addc(z1[1],m10,0,2^(n-1));
    c1:=z1[2];
    z0:=addc(z0[1],m11,c1,2^(n-1));
    c0:=z2[2];
    cadd(m21[1],c0);
  fi; [op(m00),op(z1[1]),op(z0[1]),op(m21[1])] end;
```

$kara := \mathbf{proc}(x, y, n)$ (6.4.3)

$\mathbf{local}\ m0, m00, m01, m1, m10, m11, m2, m20, m21, f, x0, x1, y0, y1, z0, z1, c0, c1;$

$\mathbf{if}\ n = 0\ \mathbf{then}$

$RETURN(mul1(x, y))$

$\mathbf{end\ if};$

$x0 := [x[1..2\wedge(n-1)]];$

$x1 := [x[1+2\wedge(n-1)..2\wedge n]];$

$y0 := [y[1..2\wedge(n-1)]];$

$y1 := [y[1+2\wedge(n-1)..2\wedge n]];$

$m0 := kara(x0, y0, n-1);$

$m00 := [m0[1..2\wedge(n-1)]];$

$m01 := [m0[1+2\wedge(n-1)..2\wedge n]];$

$m2 := kara(x1, y1, n-1);$

$m20 := [m2[1..2\wedge(n-1)]];$

$m21 := [m2[1+2\wedge(n-1)..2\wedge n]];$

$f := true;$

$\mathbf{if}\ 0 <= cmp(x1, x0, 2\wedge(n-1))\ \mathbf{then}$

$z0 := subc(x1, x0, 0, 2\wedge(n-1))[1]$

```
else
    z0 := subc(x0, x1, 0, 2^(n-1))[1];
    f := not f
end if;
if 0 <= cmp(y1, y0, 2^(n-1)) then
    z1 := subc(y1, y0, 0, 2^(n-1))[1]
else
    z1 := subc(y0, y1, 0, 2^(n-1))[1];
    f := not f
end if;
m1 := kara(z0, z1, n-1);
m10 := [m1[1..2^(n-1)]];
m11 := [m1[1+2^(n-1)..2^n]];
z0 := addc(m01, m20, 0, 2^(n-1));
c0 := z0[2];
z1 := addc(z0[1], m00, 0, 2^(n-1));
c1 := c0 + z1[2];
z0 := addc(z0[1], m21, c1, 2^(n-1));
c0 := c0 + z2[2];
m21 := cadd(z0[1], m21, c0, 2^(n-1));
if f then
    z1 := subc(z1[1], m10, 0, 2^(n-1));
    c1 := z1[2];
    z0 := subc(z0[1], m11, c1, 2^(n-1));
    c0 := z0[2];
    m21 := csub(m21[2], c2)
else
    z1 := addc(z1[1], m10, 0, 2^(n-1));
    c1 := z1[2];
    z0 := addc(z0[1], m11, c1, 2^(n-1));
    c0 := z2[2];
    cadd(m21[1], c0)
```

```
        end if;
        [op(m00), op(z1[1]), op(z0[1]), op(m21[1])]
    end proc
```

▶ 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

▶ 15. Számtest szita

▶ 16. Vegyes problémák