Semimagic generating functions and sequences

(general form, with cubic data)

Notation:

S: semimagic squares (all positive values).

s: normalized squares (symmetry types).

R: reduced squares (least element is 0).

r: reduced normalized squares (reduced symmetry types).

n: semimagic r.

gf: generating function in some form.

gfsum: generating function as a sum of simple terms.

c: Cubic (fixed strict upper bound; weak upper bound for reduced).

a: Affine (fixed magic sum).

p: Period of the quasipolynomial (known from geometry). (Period of the truncated quasipolynomial, in the affine count.)
d: Dimension of the geometry = degree of the quasipolynomials.
RtoSfactor: the rational function that multiplies Rgf to Sgf and rgf to sgf.

This is for **cubic**.

```
> d:=5; p:=60;
  RtoSfactor:=x^2/(1-x)^2;
```

The number of terms desired of each sequence is "enddegree". Comment: The slow part of the program is the series expansion (the following four commands).

```
> enddegree:=100;
```

We start by computing r_s=rsgf from the semimagic count. From the Latte results we get the closed Ehrhart g.f. of each flat, which depends on whether we're doing cubic or affine.

Set up the simplex data for the faces and intersection polytopes in the semimagic series. These are **cubic** data.

```
> simplexname[1]:="OABC": ehrgf[1]:= 1/((1-x)^3*(1-x^2)) : dimen[1]:=3:
  simplexname[2]:="OEE2": ehrgf[2]:= 1/((1-x)*(1-x^2)*(1-x^3)) :
  dimen[2]:=2:
  simplexname[3]:="OAE2": ehrgf[3]:= 1/((1-x)*(1-x^2)^2) : dimen[3]:=2:
  simplexname[4]:="ADE2": ehrgf[4]:= 1/((1-x^2)^3) : dimen[4]:=2:
  simplexname[5]:="DE1E2": ehrgf[5]:= 1/((1-x^2)^2*(1-x^3)) : dimen[5]:=2:
  simplexname[6]:="OCE": ehrgf[6]:= 1/((1-x)^2*(1-x^3)) : dimen[6]:=2:
  simplexname[7]:="BDE1": ehrgf[7]:= 1/((1-x)*(1-x^2)*(1-x^3)) :
  dimen[7]:=2:
  simplexname[8]:="ABD": ehrgf[8]:= 1/((1-x)*(1-x^2)^2) : dimen[8]:=2:
  simplexname[9]:="FG1": ehrgf[9]:= 1/((1-x^3)*(1-x^5)) : dimen[9]:=1:
  simplexname[10]:="EF": ehrgf[10]:= 1/((1-x^3)^2) : dimen[10]:=1:
  simplexname[11]:="OG": ehrgf[11]:= 1/((1-x)*(1-x^4)) : dimen[11]:=1:
  simplexname[12]:="FG": ehrgf[12]:= 1/((1-x^3)*(1-x^4)) : dimen[12]:=1:
  simplexname[13]:="AF": ehrgf[13]:= 1/((1-x^2)*(1-x^3)) : dimen[13]:=1:
  simplexname[14]:="DG": ehrgf[14]:= 1/((1-x^2)*(1-x^4)) : dimen[14]:=1:
```

```
simplexname[15]:="DG2": ehrgf[15]:= 1/((1-x^2)*(1-x^5)) : dimen[15]:=1:
simplexname[16]:="DE": ehrgf[16]:= 1/((1-x^2)*(1-x^3)) : dimen[16]:=1:
simplexname[17]:="H": ehrgf[17] := 1/(1-x^5) : dimen[17]:=0:
```

The closed E.g.f. is converted to the open E.g.f. The first step is to compute the Mobius function of the intersection poset.

```
> for n from 1 to 17 do
    mu[n]:=(-1)^(dimen[1]-dimen[n]):
  od:
  mu[14]:=2*mu[14]:
  for n from 1 to 17 do
    openehrgf[n]:=simplify(-(-1)^dimen[n]*subs(x=1/x,ehrgf[n])):
  od:
```

Get the g.f. of reduced, normalized squares.

```
> for n from 1 to 17 do
    rgfterm[n]:=openehrgf[n]:
  od:
  rgfsum:=sum(mu[nn]*rgfterm[nn],nn=1..17):
  rgf:=simplify(rgfsum):
```

Get the g.f. of reduced squares.

```
> Rgfsum:=72*rgfsum:
  Rgf:=simplify(Rgfsum):
```

Hence S, the g.f. of the number of semimagic squares, equals

```
> Sgf:=simplify(RtoSfactor*Rgf):
```

The g.f. of the total number of symmetry types, l_ml ("lgf"):

```
> sgf:=simplify(RtoSfactor*rgf):
```

Generate the labelled sequence of magilatin square numbers of all four kinds. The first step is to compute the degree of the first non-zero term.

```
> Sgfdegree:=ldegree(numer(Sgf),x);
  Rgfdegree:=ldegree(numer(Rgf),x);
  sgfdegree:=ldegree(numer(sgf),x);
  rgfdegree:=ldegree(numer(rgf),x);
```

Generate the series expansions of the g.f.'s.

Expressing the rational function with standard denominator gives an orders-of-magnitude speedup in the series expansion.

Standard denominator (1-x^p)^{d+1}.

```
> pdenom:=(1-x^p):
  standenom:=pdenom^(d+1);
```

G.f. as rational function with standard denominator.

```
> Sgfstandnum:=simplify(numer(Sgf)*standenom/denom(Sgf)):
  Sgf:=Sgfstandnum/standenom;
```

G.f. as rational function with standard denominator.

```
> Rgfstandnum:=simplify(numer(Rgf)*standenom/denom(Rgf)):
```

```
Rgf:=Rgfstandnum/standenom;
```

G.f. as rational function with standard denominator.

```
> sgfstandnum:=simplify(numer(sgf)*standenom/denom(sgf)):
  sgf:=sgfstandnum/standenom;
```

G.f. as rational function with standard denominator.

```
> rgfstandnum:=simplify(numer(rgf)*standenom/denom(rgf)):
  rgf:=rgfstandnum/standenom;
```

Expand as a series to find the first few values of the number of squares and symmetry types.

```
> Sseries:=series(Sgf,x=0,enddegree+1):
  print("Series computed.");
```

```
> sseries:=series(sgf,x=0,enddegree+1):
  print("Series computed.");
```

Expand as a series to find the first few values of the number of reduced squares and reduced symmetry types.

```
> Rseries:=series(Rgf,x=0,enddegree+1):
  print("Series computed.");
```

```
> rseries:=series(rgf,x=0,enddegree+1):
  print("Series computed.");
```

Find the counting sequences

List the coefficients of each series, i.e., the terms of the counting sequences.
The comment symbol # is for controlling the output. With large "enddegree" the output is huge so it's more convenient to run each sequence's output separately and copy it from the worksheet.

```
> for n from Sgfdegree to enddegree do
   co:=coeff(Sseries,x,n):
  # printf("%d  %d \n",n,co);
  od:
  print("Coefficients complete.",n,co);
```

```
> for n from Rgfdegree to enddegree do
   co:=coeff(Rseries,x,n):
  # printf("%d  %d \n",n,co);
  od:
  print("Coefficients complete.",n,co);
```

```
> for n from sgfdegree to enddegree do
   co:=coeff(sseries,x,n):
  # printf("%d  %d \n",n,co);
  od:
  print("Coefficients complete.",n,co);
```

```
> for n from rgfdegree to enddegree do
   co:=coeff(rseries,x,n):
  # printf("%d  %d \n",n,co);
  od:
  print("Coefficients complete.",n,co);
```