582e -- minimal model

Coding up the Laska-Kraus-Connell algorithm from Cremona, Section 3.2

```
def bmod(a, n):
   """
   Balanced mod b with -n/2 < b <= n/2.
   INPUT: a, n -- integers
   OUTPUT: integer b with -n/2 < b <= n/2
   """
   b = a%n
   if b > n/2: b -= n
   return b

def reduced_minimal_model(c4, c6):
   """
   INPUT:
       c4, c6 -- integers; invariants of an elliptic curve E
   OUTPUT:
       u -- scaling factor
       a1, a2, a3, a4, a6 -- coefficients of a reduced model for
E
   EXAMPLES:
       sage: u, a = minimal_model(-44677734375,
-13080596923828125)
       sage: factor(u), a
       (5^3, (1, -1, 0, 4, 3))
       sage: E = EllipticCurve_from_c4c6(-44677734375,
-13080596923828125)
       sage: E.minimal_model().a_invariants()
       [1, -1, 0, 4, 3]
   """
   # type check and compute discriminant Delta
   c4, c6 = (ZZ(c4), ZZ(c6))
   Delta = ZZ( (c4^3 - c6^2)/1728 )

   # Compute scaling factor u
   u = 1
   g = gcd(c6^2, Delta)
   for p in prime_divisors(g):
       d = valuation(g, p) // 12
       if p == 2:
           a = bmod(c4 / (2^(4*d)), 16)
           b = bmod(c6 / (2^(6*d)), 32)
```

```
            if b%4 != 3 and not (a == 0 and (b==0 or b==8)):
                d -= 1
        elif p == 3 and valuation(c6, 3) == 6*d + 2:
            d -= 1
        u *= p^d
    # Transform to reduced minimal model
    c4 /= u^4; c6 /= u^6
    b2 = bmod(-c6, 12)
    b4 = (b2^2 - c4)/24
    b6 = (-b2^3 + 36*b2*b4 - c6)/216
    a1 = b2 % 2
    a2 = (b2 - a1)/4
    a3 = b6 % 2
    a4 = (b4 - a1*a3)/2
    a6 = (b6 - a3)/4
    return u, (ZZ(a1),ZZ(a2),ZZ(a3),ZZ(a4),ZZ(a6))
```

```
def mm2(c4,c6):
    return EllipticCurve_from_c4c6(c4,c6).minimal_model()
```

```
from sage.schemes.elliptic_curves.weierstrass_morphism import
baseWI
T = baseWI(5^(-3),1,2,3)
```

```
F = EllipticCurve(T([1,2,3,4,5])); F
```

```
    Elliptic Curve defined by y^2 + 625*x*y + 19531250*y = x^3 -
    15625*x^2 - 2929687500*x - 34332275390625 over Rational Field
```

```
factor(F.conductor())
```

```
    11 * 941
```

```
c4, c6 = F.c_invariants(); c4, c6
```

```
    (-44677734375, -13080596923828125)
```

```
u, a = reduced_minimal_model(c4, c6)
print factor(u), a
```

```
    5^3 (1, -1, 0, 4, 3)
```

```
F.minimal_model().a_invariants()
```

```
    [1, -1, 0, 4, 3]
```

```
# half this time is dominated by stupid overhead in call to
# pari to factor g (see below).
timeit('reduced_minimal_model(c4, c6)')
```

```
    625 loops, best of 3: 276 Âµs per loop
```

```
f = pari(F)
# I looked at the PARI source and this actually
# runs Tate's algorithm for all bad primes
timeit('f.ellminimalmodel()')
```

```
    625 loops, best of 3: 33.3 Âµs per loop
```

```
timeit('factor(145519152283668518066406 25)')
```

```
    625 loops, best of 3: 127 Âµs per loop
```

```
n = pari(145519152283668518066406 25)
timeit('n.factor()')
```

```
    625 loops, best of 3: 8.93 Âµs per loop
```

```
def testit2():
   """
   EXAMPLES:
       sage: for n in range(1000): testit2()
   """
   from sage.schemes.elliptic_curves.weierstrass_morphism import
baseWI
   try:
       T = baseWI(ZZ(randint(1,10))^(-randint(0,20)),
              randint(-50,50), randint(-50,50), randint(-50,50))
       E = EllipticCurve(T([randint(-100,100) for _ in
range(5)]))
   except ArithmeticError:
       return
   c4, c6 = E.c_invariants()
   try:
       u, (a1, a2, a3, a4, a6) = reduced_minimal_model(c4, c6)
   except Exception, msg:
       print msg
       print c4, c6
   F = EllipticCurve([a1,a2,a3,a4,a6])
   if F.minimal_model().a_invariants() != F.a_invariants():
       print c4, c6
```

```
%time
for n in range(1000): testit2()
```

```
    CPU time: 2.38 s,  Wall time: 2.39 s
```