We define our elliptic curve, which has rank 1:

```
E = EllipticCurve('37a')
show(E)
show(E.plot(hue=0.8, thickness=3))
```

$$y^2 + y = x^3 - x$$

We compute a list of the first 20 fundamental discriminants that satisfy the Heegner hypothesis.

```
discs = E.heegner_discriminants_list(20); discs
        [-7, -11, -40, -47, -67, -71, -83, -84, -95, -104, -107, -115, -12
        -132, -136, -139, -151, -152]
```

## Table

Here is a nice table that gives each discriminant, the class number, the index of the Heegner point in $E(K)$, and the actual Heegner point (up to sign). The Heegner point is computed by cheating -- we compute the height of the Heegner point using the Gross-Zagier formula, and simply take the corresponding multiple of a known generator of $E$.

```
P = E.gens()[0]
print '<html><table cellpadding=5 cellspacing=1 border=2 bgcolor="#fafafa'
print '<tr><td>Discriminant</td><td>Class Number</td><td>Index</td><td>Poi
for D in discs:
    ind = E.heegner_index(D)
    if ind > 0:
        ind = ind.sqrt().simplest_rational()
        y = ZZ(ind)*P
    else:
        ind = infinity
        y = E(0)
    hK = QuadraticField(D,'sqrtD').class_number()
    print '<tr><td>%s</td><td>%s</td><td>%s</td><td>%s</td></tr>'%(D,hK,ir
print '</table></html>'
```

| Discriminant | Class Number | Index | Point |
|---|---|---|---|
| -7 | 1 | 1 | (0 : 0 : 1) |
| -11 | 1 | 1 | (0 : 0 : 1) |
| -40 | 2 | 2 | (1 : 0 : 1) |
| -47 | 5 | 1 | (0 : 0 : 1) |
| -67 | 1 | 6 | (6 : 14 : 1) |
| -71 | 7 | 1 | (0 : 0 : 1) |
| -83 | 3 | 1 | (0 : 0 : 1) |
| -84 | 4 | 1 | (0 : 0 : 1) |
| -95 | 8 | +Infinity | (0 : 1 : 0) |
| -104 | 6 | +Infinity | (0 : 1 : 0) |
| -107 | 3 | +Infinity | (0 : 1 : 0) |
| -115 | 2 | 6 | (6 : 14 : 1) |
| -120 | 4 | 2 | (1 : 0 : 1) |
| -123 | 2 | 3 | (-1 : -1 : 1) |
| -127 | 5 | 1 | (0 : 0 : 1) |
| -132 | 4 | 3 | (-1 : -1 : 1) |
| -136 | 4 | 4 | (2 : -3 : 1) |
| -139 | 3 | +Infinity | (0 : 1 : 0) |
| -151 | 7 | 2 | (1 : 0 : 1) |
| -152 | 6 | 2 | (1 : 0 : 1) |

## Now we directly compute the Heegner point for discriminant -7

```
K.<a> = QuadraticField(-7); K
        Number Field in a with defining polynomial x^2 + 7
D = K.discriminant(); D
        -7
K.class_group()
        Trivial Abelian Group
N = 37
```

We compute the square roots of D modulo $4N$.

```
srs = Mod(D, 4*N).sqrt(all=True); srs
        [17, 57, 91, 131]
```

Then make a list of their reductions modulo $2N$:

```
S = list(set([Mod(a, 2*N) for a in srs])); S
        [17, 57]
```

For fun we list all representative primitive binary quadratic forms of discriminant -7 -- we will not need this later.

```
C = BinaryQF_reduced_representatives(-7); C
        [x^2  + xy  + 2y^2 ]
```

The following is useful for making the computations more efficient; we will not use it yet.

```
def minimal_form(f, N):
    """
    Given a positive definite form f = (A,B,C) of discriminant D < 0
    with N | A, this function returns a form f' = (A',B',C') equivalent
    to f with N | A' and B' = B (mod 2*N) with A' minimal.
    """
    A,B,C = f
    u = ZZ(-B/(2*A/N))
    v2 = ZZ( abs(D) / (2*A / N)^2 )
    raise NotImplementedError
```

We make a list of the binary quadratic forms of discriminant $D$ that have first entry divisible by $N$.

```
def fill_list(D, N, b):
    b = ZZ(b)
    R = ZZ((b^2 - D)/(4*N))
    L = []
    Lprime = []
    for d in divisors(R):
       f = BinaryQF([d*N,b,R/d])
       fr = f.reduced()
       if not fr in Lprime:
           Lprime.append(fr)
           L.append(f)
    return L

b = S[0]; print b
L = fill_list(D, N, b); L
        17
        [37x^2  + 17xy  + 2y^2 ]
def tau(f):
   A,B,C = f
   return (-B + sqrt(D))/(2*A)

v = [CDF(tau(f)) for f in L]
v
        [-0.22972972973 + 0.0357533960955*I]
```

```
e = gp(E)

E2 = E.change_ring(CDF)

def phi(E, t, prec):
    """
    Map a point t in the upper half plan to a point on the complex torus
    representation of the elliptic curve, using prec terms of the
    modular parametrization.
    """
    q = CDF(exp(2*pi*I*t))
    a = E.anlist(prec)
    return sum( (a[n]/n) * q^n for n in xrange(1,prec))

phi(E, v[0], 100)
       0.929592715295 - 1.22569469101*I
phi(E, v[0], 200)
       0.929592715279 - 1.225694691*I
def topoint(E, z):
    """
    Given a point in the complex lattice representation of E, return
    a point on the Weierstrass model of E.  The curve E should be defined
    over a complex floating point field.
    """
    e = gp(E)
    w = list(e.ellztopoint(z))
    K = E.base_field()
    return E.point([K(repr(w[0])), K(repr(w[1])), K(1)], check=False)

topoint(E2, phi(E, v[0], 200))
       (-6.3956744052e-12 - 6.60496957297e-12*I : 6.3956744052e-12 +
       6.60496957289e-12*I : 1.0)
```

## Next we try discriminant -67

In this example, the class number is still 1, but the Gross-Zagier formula predicts an index of 6.

```
D = -67
K.<a> = QuadraticField(D); K
       Number Field in a with defining polynomial x^2 + 67
K.class_number()
       1
srs = Mod(D, 4*N).sqrt(all=True); srs
       [9, 65, 83, 139]
S = list(set([Mod(a, 2*N) for a in srs])); S
       [9, 65]
b = S[0]; print b
L = fill_list(D, N, b); L
       9
       [37x^2  + 9xy  + y^2 ]
v = [CDF(tau(f)) for f in L]; v
       [-0.121621621622 + 0.110612875296*I]
phi(E, v[0], 100)
       0.409361000751 - 2.44021449129e-12*I
topoint(E2, phi(E, v[0], 100))
       (6.00000000002 + 7.07662202477e-11*I : -15.0000000001 - 2.61102950
       1.0)
```

```
-topoint(E2, phi(E, v[0], 100))
        (6.00000000002 + 7.07662202477e-11*I : 14.0000000001 + 2.611029505
b = S[1]; print b
L = fill_list(D, N, b); L
v = [CDF(tau(f)) for f in L]; v
        65
        [-0.878378378378 + 0.110612875296*I]
topoint(E2, phi(E, v[0], 100))
        (6.00000000002 - 7.07662202477e-11*I : -15.0000000001 + 2.61102950
        1.0)
-topoint(E2, phi(E, v[0], 100))
        (6.00000000002 - 7.07662202477e-11*I : 14.0000000001 - 2.611029505
```

## Next we consider discriminant -40, which has class number 2 and Heegner index 2

```
D = -40
K.<a> = QuadraticField(D); K
        Number Field in a with defining polynomial x^2 + 40
K.class_number()
        2
srs = Mod(D, 4*N).sqrt(all=True); srs
        [16, 58, 90, 132]
S = list(set([Mod(a, 2*N) for a in srs])); S
        [16, 58]
```

We use the first $b$, as usual. Note that there are two representatives since the class group has order 2.

```
b = S[0]; print b
L = fill_list(D, N, b); L
        16
        [37x^2  + 16xy  + 2y^2 , 74x^2  + 16xy  + y^2 ]
```

These are the corresponding Heegner points in the upper half plane.

```
v = [CDF(tau(f)) for f in L]
v
        [-0.216216216216 + 0.0854669637883*I, -0.108108108108 + 0.04273348
```

We compute their images in the complex torus representation of the elliptic curve.

```
e = [phi(E, z, 100) for z in v]
e
        [0.567136607832 - 0.572182654882*I, 0.567136607828 + 0.57218265488
```

Their sum in the torus is a point that must map to the Heegner point $y_K$ on the Weierstrass equation, which is defined over $\mathbf{Q}$.

```
w = e[0] + e[1]
```

This point is not exactly the point in the table above. In fact it is negative of that point.

```
topoint(E2, w)
        (1.0 - 1.51800794158e-12*I : -1.0 + 3.03601588315e-12*I : 1.0)
-topoint(E2, w)
        (1.0 - 1.51800794158e-12*I : 2.33657537763e-12 - 3.03601588315e-12
```

## Hilbert Class Field

We could also find the image of $y_1 \in E(H)$ point on the Weierstrass equation, which is a point defined over the Hilber class field of $K$, which is a quadratic extension of the field $K$.

```
y1 = topoint(E2, e[0])

y1
        (9.04203384991e-12 + 1.41421356237*I : 0.99999999999 - 1.414213562
CDF(sqrt(2))
        1.41421356237
```

As you can see, the point $y_1$ is defined over $\mathbf{Q}(\sqrt{-2})$. We have ``magically'' constructed the Hilbert class field of $K$ -- it's $K(\sqrt{-2})$.

## Finally we consider discriminant -71, which has class number 7 and Heegner index 1

```
D = -71
K.<a> = QuadraticField(D); K
        Number Field in a with defining polynomial x^2 + 71
K.class_number()
        7
srs = Mod(D, 4*N).sqrt(all=True); srs
        [15, 59, 89, 133]
S = list(set([Mod(a, 2*N) for a in srs])); S
        [59, 15]
```

We use the first $b$, as usual. Note that there are two representatives since the class group has order 2.

```
b = S[0]; print b
L = fill_list(D, N, b); L
        59
        [37x^2  + 59xy  + 24y^2 , 74x^2  + 59xy  + 12y^2 , 111x^2  + 59xy
        148x^2  + 59xy  + 6y^2 , 222x^2  + 59xy  + 4y^2 , 444x^2  + 59xy
        888x^2  + 59xy  + y^2 ]
```

These are the corresponding Heegner points in the upper half plane.

```
v = [CDF(tau(f)) for f in L]; v
        [-0.797297297297 + 0.113866888827*I, -0.398648648649 + 0.056933444
        -0.265765765766 + 0.0379556296089*I, -0.199324324324 + 0.028466722
        -0.132882882883 + 0.0189778148045*I, -0.0664414414414 + 0.00948890
        -0.0332207207207 + 0.00474445370111*I]
```

We compute their images in the complex torus representation of the elliptic curve.

```
e = [phi(E, z, 1000) for z in v]; e
        [0.436338629693 + 0.391421370491*I, -1.08506665883 - 0.56388900859
        0.302913505562 - 1.22569469099*I, 1.52920424182 - 0.558966284143*I
        1.52920424182 + 0.558966284147*I, -1.08506665882 + 0.563889008594*
        0.436338629696 - 0.391421370488*I]
```

Their sum in the torus is a point that must map to the Heegner point $y_K$ on the Weierstrass equation, which is defined over $\mathbf{Q}$.

```
w = sum(e); w
        2.06386593094 - 1.22569469098*I
```

This point is not exactly the point in the table above. In fact it is negative of that point.

```
topoint(E2, w)
        (6.56395542695e-12 - 1.33950304269e-11*I : -0.999999999993 - 1.339
        : 1.0)
-topoint(E2, w)
        (6.56395542695e-12 - 1.33950304269e-11*I : -6.56397158849e-12 +
        1.33950304271e-11*I : 1)
y1 = topoint(E2, e[0]); y1
        (0.326282033345 - 2.82634364479*I : 2.48918514468 + 4.09829358564*
RR.<x> = PolynomialRing(CDF)
f = prod(x - topoint(E2, e[i])[0] for i in xrange(len(e)))
f
        1.0*x^7 + (-2.00000000003 - 9.62474544508e-12*I)*x^6 + (9.00000000
        1.00197627972e-10*I)*x^5 + (-10.0000000003 + 1.69921854365e-11*I)*
        (-0.999999999939 + 1.66228364407e-10*I)*x^3 + (8.00000000023 -
        1.35994326911e-10*I)*x^2 + (-5.0000000002 + 3.67623709252e-11*I)*x
        1.00000000005 - 2.89157586764e-12*I
SS.<x> = PolynomialRing(QQ)
f = x^7 - 2*x^6 + 9*x^5 - 10*x^4 -x^3 + 8*x^2 -5*x + 1
factor(f.discriminant())
        -1 * 71^3
```