INTEGER POINTS ON ELLIPTIC CURVES AND RADII OF CIRCLES

ANDREW REITER

1. Idea

First off we know from Siegel that set of integer points of elliptic curves is finite for a given curve (I hope I remember that right :D).

In looking at integer points of $y^2 = x^3$ you can notice that the first integral points are

$$\{x_k, y_k\} = \{(0,0), (1, \pm 1), (4, \pm 8), (9, \pm 27), (16, \pm 64), (25, \pm 125) \ldots\}$$

If you let $(0,0)$ be the center of a circle with radius $(x_{j=i\geq 1}, y_j)$ then the progression of radii is

$$\{r_k\} = \{0, 1^2\sqrt{2}, 2^2\sqrt{5}, 3^2\sqrt{10}, 4^2\sqrt{17}, 5^2\sqrt{26}, \ldots\}$$

for the points shown above. We can split this up

$$r_k = f(k)\sqrt{s(k, s(k-1))}$$

Where $f(k) = k^2$ and

$$\begin{aligned} s(0, s(-1)) &= 0 \\ s(1, s(0)) &= s(0) + 2(i) - 1 = 1 + 1 = 2 \\ s(2, s(1)) &= 2 + (2(2) - 1) = 5 \\ s(3, s(2)) &= 5 + (2(3) - 1) = 10 \end{aligned}$$

and so on. I did not look at the progression of θ, the angle from $y = 0$ and $x > 0$, as we would really just be looking at $\frac{y}{x}$.

2. code?

To see if I could quickly find integral points of other ECs, I wrote python script to help look at $x, y \in [0, N]$ and let $N = 1500$. I generate solution sets for y and x independently and then take the intersection to be the range values we are looking for from each solution set. This is memory usage is painful since I store the inverse maps, so we can quickly find the x, y values from the set produced by the intersection. I also don't know how bad the intersection operation is in python, but whatever.

```
import sys
imap_sq = {}
imap_rhs = {}
A = 0
B = 0
```

```
def y2(x):
        xx = x*x
        imap_sq[xx] = x
        return xx
def x3(x):
        global A, B
        xx = x*x*x + A*x + B
        imap_rhs[xx] = x
        return xx
def main():
        global A
        global B
        if len(sys.argv) == 3:
                A = int(sys.argv[1])
                B = int(sys.argv[2])
        elif len(sys.argv) == 2:
                A = int(sys.argv[1])
        ysq = map(y2, range(0, 1500))
        rhs = map(x3, range(0, 1500))
        intersex = set(ysq) & set(rhs)
        for k in intersex:
                print "(x, y) = (%d, %d)" % (imap_rhs[k], imap_sq[k])
        return
if __name__ == '__main__':
        main()
```

2.1. **modif = 3.** find points $(0,0),(1,2),(12,42)$ in the first thousand which leads to

$$\{r_i\} = \{0, \sqrt{5}, 6\sqrt{53}, \ldots\}$$