Triangle Coding

Our system has found a correspondence between a pair of non-negative integers (x_a, x_b) where $x_a \geq x_b$, and a single non-negative integer x by enumerating the following sequence of pairs (x_a, x_b) with non-negative integers:

$$(0,0),(1,0),(1,1),(2,0),(2,1),(2,2),(3,0),\dots.$$

In one direction, it computes $x = \frac{x_a \times (x_a+1)}{2} + x_b$ in order to "encode" the pair (x_a, x_b). Decoding of a single non-negative integer x can be calculated as $(x_a, x_b) = (f_0(x) - f_1(x), f_0(x))$, where the functions f_0, f_1 can be implemented in Python for example as follows (an actual code invented by the program generator). Note that the function f_0 computes x_b, and the function f_1 computes $x_b - x_a$ (a non-positive integer).

```
def f0(X):
    x = X
    for y in range (1,(2 + (X // (1 + (2 + 2)))) + 1):
        x = x - (y if (y - x) <= 0 else 0)
    return x

def f1(X):
    x = X
    for y in range (1,(2 + (X // (1 + (2 + 2)))) + 1):
        x = x - (0 if x <= 0 else (1 + y))
    return x
```

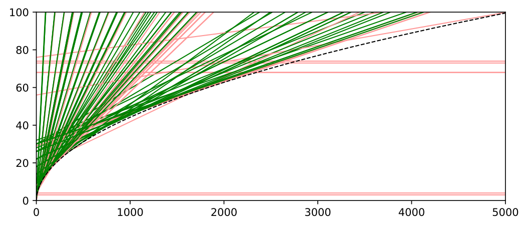
This representation was initially discovered when our system found solutions for the sequences A2262,[1] and A25581.[2] Indeed, f_0 is a solution for A2262 invented in the first generation and $-f_1$ is a solution for A25581 invented during the 9^{th} generation.

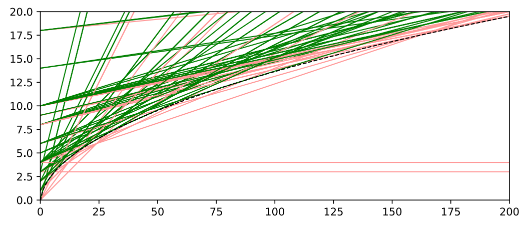
However the triangle coding turned out to be useful in many other cases. The designed language for our programs only supports a maximum of two variables but using the triangle coding, a pair of variables can be temporarily packed into one. A simple example is sequence A279364[3] – sum of 5th powers of proper divisors of n. A human programmer could implement this sequence as:

```
def f(X):
    res = 0
    for y in range(1,X+1):
        res = res + (y**5 if (X+1) % y <= 0 else 0)
    return res
```

There are three variables used in the body of the loop, in particular `res`, `y`, and `X`. Since the native language supports only two variables available at every moment, this Python code cannot be straightforwardly translated into the language of our programs. Nevertheless, the program generator has found a workaround using the triangle coding – it packs `X` and `y` into a single natural

[1] https://oeis.org/A002262
[2] https://oeis.org/A025581
[3] https://oeis.org/A279364





number, so it can perform the summing loop adding to `res`, and in order to decode what to add to `res`, it unpacks the pair, and checks whether `y` divides `X+1`.

Number of steps

In order to calculate f_0 and f_1, one must perform approximatelly $\sqrt{2x+\frac{1}{4}}-\frac{1}{2}$ steps in a subtracting loop. Since the programming language is based on `for` loops, the programs are approximating the number of suptracting steps with a function which can be easily obtained using the basic arithmetical operations. In particular, the code above uses approximations with $2 + x/5$ where 5 is written as $1 + (2 + 2)$ (our basic language only supports constants 0,1,2). The program generator has experimented with many such bounds, vast majority of such bounds are of the shape $a + x/b$ or $a + 2(x/b)$ where a, b are constants. We have collected all such subprograms used in the generated programs, and plotted the bounds in the pictures above. The approximated function $\sqrt{2x+\frac{1}{4}}-\frac{1}{2}$ is plotted with a black dashed line. All the found valid bounds are plotted green, all the invalid bound attempts are plotted red.