High-speed cryptography, part 1: elliptic-curve formulas

Daniel J. Bernstein
University of Illinois at Chicago &
Technische Universiteit Eindhoven

Crypto performance problems often lead users to reduce cryptographic security levels or give up on cryptography.

Example 1 (according to Firefox on Linux, 2013.06.24): Google SSL uses RSA-1024.

Security note:
Analyses in 2003 concluded that RSA-1024 was breakable; e.g., 2003 Shamir–Tromer estimated 1 year, $\approx 10^7$ USD.
RSA Labs and NIST response:
Move to RSA-2048 by 2010.

Example 2: Tor uses RSA-1024.

Example 3: DNSSEC uses RSA-1024: "tradeoff between the risk of key compromise and performance..."

Example 4: OpenSSL uses secret AES load addresses; dangerous!

Example 5: `https://sourceforge.net/account` is protected by SSL but `https://sourceforge.net/develop` redirects browser to `http://sourceforge.net/develop`, turning off the cryptography.

Extensive work on ECC speed
$\Rightarrow$ fast high-security ECC.
Example: Curve25519 ECDH in
460200 Cortex A8 cycles;
332304 Snapdragon S4 cycles;
182632 Ivy Bridge cycles.

Requires serious analysis
and optimization of algorithms.
Not just "polynomial time";
not just "quadratic time".

My topic today:
decomposing elliptic-curve
operations into field operations.

Eliminating divisions

Typical computation:
$P \mapsto nP$.

Decompose into additions:
$P, Q \mapsto P + Q$.

Addition $(x_1, y_1) + (x_2, y_2) =$
$((x_1 y_2 + y_1 x_2)/(1 + d x_1 x_2 y_1 y_2),$
$(y_1 y_2 - x_1 x_2)/(1 - d x_1 x_2 y_1 y_2))$
uses expensive divisions.

Better: postpone divisions
and work with fractions.
Represent (x, y) as
$(X : Y : Z)$ with $x = X/Z$ and
$y = Y/Z$ for $Z \neq 0$.

Addition now has to handle fractions as input:

$$\left(\frac{X_1}{Z_1}, \frac{Y_1}{Z_1}\right) + \left(\frac{X_2}{Z_2}, \frac{Y_2}{Z_2}\right) =$$

$$\left(\frac{\frac{X_1}{Z_1}\frac{Y_2}{Z_2} + \frac{Y_1}{Z_1}\frac{X_2}{Z_2}}{1 + d\frac{X_1}{Z_1}\frac{X_2}{Z_2}\frac{Y_1}{Z_1}\frac{Y_2}{Z_2}},\right.$$

$$\left.\frac{\frac{Y_1}{Z_1}\frac{Y_2}{Z_2} - \frac{X_1}{Z_1}\frac{X_2}{Z_2}}{1 - d\frac{X_1}{Z_1}\frac{X_2}{Z_2}\frac{Y_1}{Z_1}\frac{Y_2}{Z_2}}\right) =$$

$$\left(\frac{Z_1 Z_2(X_1 Y_2 + Y_1 X_2)}{Z_1^2 Z_2^2 + d X_1 X_2 Y_1 Y_2},\right.$$

$$\left.\frac{Z_1 Z_2(Y_1 Y_2 - X_1 X_2)}{Z_1^2 Z_2^2 - d X_1 X_2 Y_1 Y_2}\right)$$

i.e. $\left(\frac{X_1}{Z_1}, \frac{Y_1}{Z_1}\right) + \left(\frac{X_2}{Z_2}, \frac{Y_2}{Z_2}\right)$

$= \left(\frac{X_3}{Z_3}, \frac{Y_3}{Z_3}\right)$

where

$F = Z_1^2 Z_2^2 - dX_1X_2Y_1Y_2,$
$G = Z_1^2 Z_2^2 + dX_1X_2Y_1Y_2,$
$X_3 = Z_1Z_2(X_1Y_2 + Y_1X_2)F,$
$Y_3 = Z_1Z_2(Y_1Y_2 - X_1X_2)G,$
$Z_3 = FG.$

Input to addition algorithm:
$X_1, Y_1, Z_1, X_2, Y_2, Z_2.$
Output from addition algorithm:
$X_3, Y_3, Z_3.$ No divisions needed!

Save multiplications by eliminating common subexpressions:

$A = Z_1 \cdot Z_2$; $B = A^2$;
$C = X_1 \cdot X_2$;
$D = Y_1 \cdot Y_2$;
$E = d \cdot C \cdot D$;
$F = B - E$; $G = B + E$;
$X_3 = A \cdot F \cdot (X_1 \cdot Y_2 + Y_1 \cdot X_2)$;
$Y_3 = A \cdot G \cdot (D - C)$;
$Z_3 = F \cdot G$.

Cost: $11\mathbf{M} + 1\mathbf{S} + 1\mathbf{D}$.

Can do better: $10\mathbf{M} + 1\mathbf{S} + 1\mathbf{D}$.

Faster doubling

$(x_1, y_1) + (x_1, y_1) =$
$((x_1 y_1 + y_1 x_1)/(1 + d x_1 x_1 y_1 y_1),$
$(y_1 y_1 - x_1 x_1)/(1 - d x_1 x_1 y_1 y_1)) =$
$((2 x_1 y_1)/(1 + d x_1^2 y_1^2),$
$(y_1^2 - x_1^2)/(1 - d x_1^2 y_1^2)).$

$x_1^2 + y_1^2 = 1 + d x_1^2 y_1^2$ so
$(x_1, y_1) + (x_1, y_1) =$
$((2 x_1 y_1)/(x_1^2 + y_1^2),$
$(y_1^2 - x_1^2)/(2 - x_1^2 - y_1^2)).$

Again eliminate divisions using $\mathbf{P}^2$: only $3\mathbf{M} + 4\mathbf{S}$.

Much faster than addition.

Useful: many doublings in ECC.

More addition strategies

Dual addition formula:

$(x_1, y_1) + (x_2, y_2) =$

$((x_1y_1 + x_2y_2)/(x_1x_2 + y_1y_2),$

$(x_1y_1 - x_2y_2)/(x_1y_2 - x_2y_1)).$

Low degree, no need for d.

Warning: fails for doubling!

Is this really "addition"?

Most EC formulas have failures.

More addition strategies

Dual addition formula:

$(x_1, y_1) + (x_2, y_2) =$

$((x_1 y_1 + x_2 y_2)/(x_1 x_2 + y_1 y_2),$

$(x_1 y_1 - x_2 y_2)/(x_1 y_2 - x_2 y_1)).$

Low degree, no need for d.

Warning: fails for doubling!

Is this really "addition"?

Most EC formulas have failures.

More coordinate systems:

Inverted: $x = Z/X$, $y = Z/Y$.

Extended: $x = X/Z$, $y = Y/T$.

Completed: $x = X/Z$, $y = Y/Z$,

$xy = T/Z$.

More elliptic curves

Edwards curves are elliptic.
Easiest way to understand elliptic curves is Edwards.

Geometrically, all elliptic curves are Edwards curves.

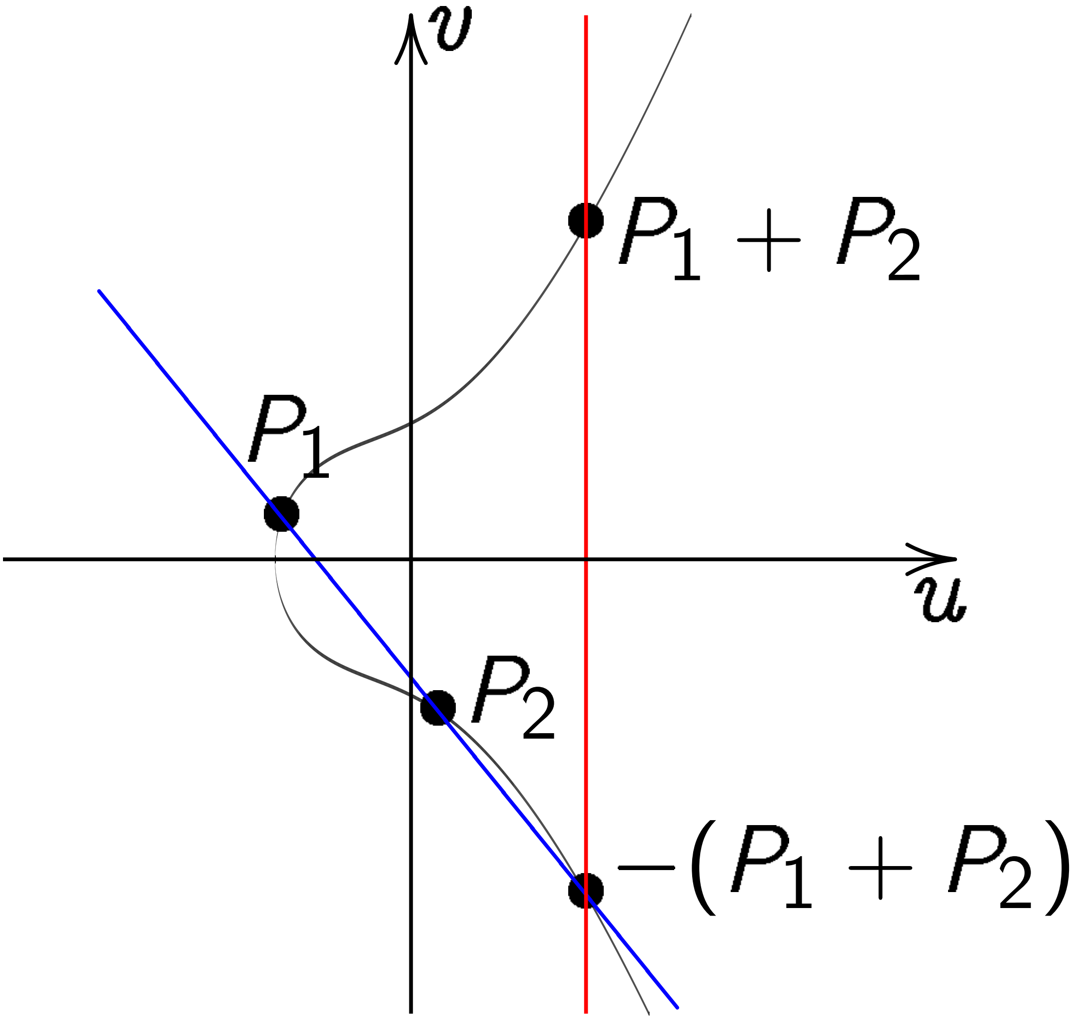
Algebraically,
more elliptic curves exist.

Every odd-char curve can be expressed as Weierstrass curve $v^2 = u^3 + a_2u^2 + a_4u + a_6$.

Warning: "Weierstrass" has different meaning in char 2.

Addition on Weierstrass curve

$$v^2 = u^3 + u^2 + u + 1$$

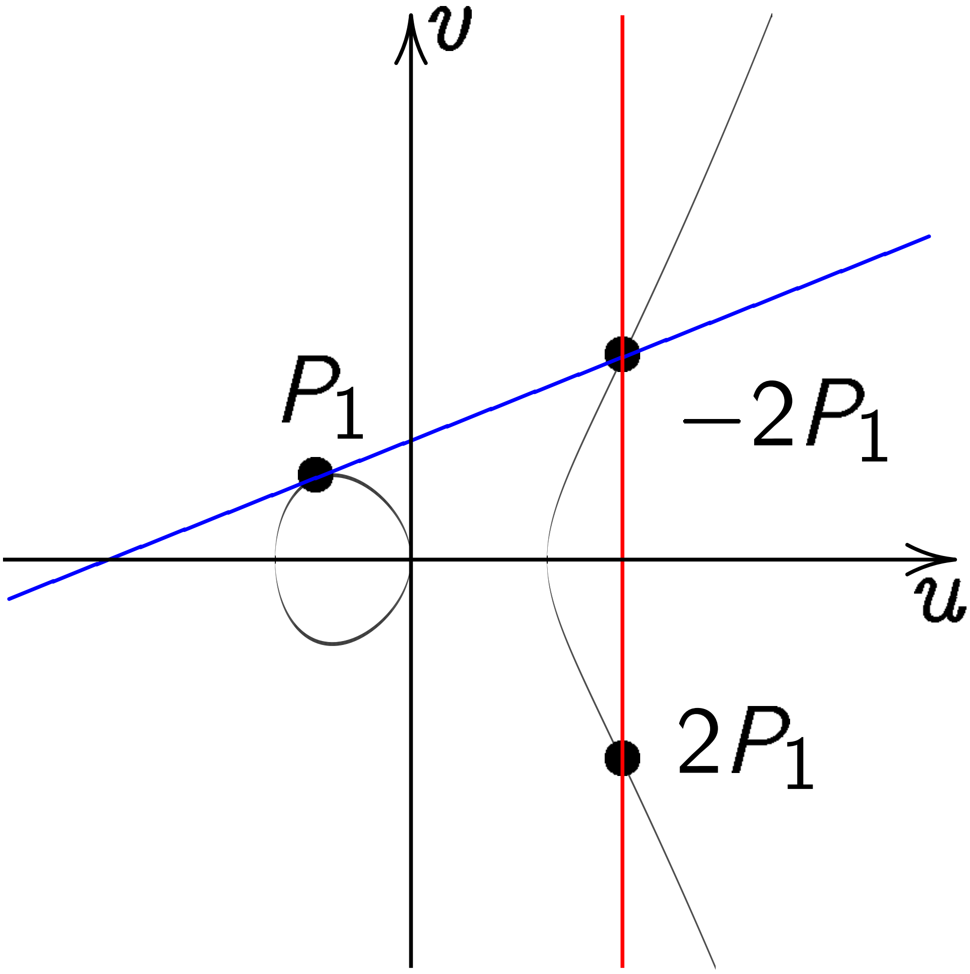


Slope $\lambda = (v_2 - v_1)/(u_2 - u_1)$.

Note that $u_1 \neq u_2$.

Doubling on Weierstrass curve

$v^2 = u^3 - u$



Slope $\lambda = (3u_1^2 - 1)/(2v_1)$.

In most cases

$(u_1, v_1) + (u_2, v_2) =$
(u_3, v_3) where $(u_3, v_3) =$
$(\lambda^2 - u_1 - u_2, \lambda(u_1 - u_3) - v_1)$.

$u_1 \neq u_2$, "addition" (alert!):
$\lambda = (v_2 - v_1)/(u_2 - u_1)$.
Total cost $1\mathbf{I} + 2\mathbf{M} + 1\mathbf{S}$.

$(u_1, v_1) = (u_2, v_2)$ and $v_1 \neq 0$,
"doubling" (alert!):
$\lambda = (3u_1^2 + 2a_2u_1 + a_4)/(2v_1)$.
Total cost $1\mathbf{I} + 2\mathbf{M} + 2\mathbf{S}$.

Also handle some exceptions:
$(u_1, v_1) = (u_2, -v_2)$;
inputs at ∞.

Birational equivalence

Starting from point (x, y)
on $x^2 + y^2 = 1 + dx^2y^2$:

Define $A = 2(1 + d)/(1 - d)$,
$B = 4/(1 - d)$;
$u = (1 + y)/(B(1 - y))$,
$v = u/x = (1 + y)/(Bx(1 - y))$.
(Skip a few exceptional points.)

$$v^2 = u^3 + (A/B)u^2 + (1/B^2)u.$$

Maps Edwards to Weierstrass.
Compatible with point addition!

Easily invert this map:
$x = u/v$, $y = (Bu - 1)/(Bu + 1)$.

Some history

There are many perspectives on elliptic-curve computations.

1984 (published 1987) Lenstra: ECM, the elliptic-curve method of factoring integers.

1984 (published 1985) Miller,
and independently
1984 (published 1987) Koblitz:
Elliptic-curve cryptography.

Bosma, Goldwasser–Kilian, Chudnovsky–Chudnovsky, Atkin: elliptic-curve primality proving.

The Edwards perspective is new!

1761 Euler, 1866 Gauss introduced an addition law for $x^2 + y^2 = 1 - x^2y^2$, the “lemniscatic elliptic curve.”

2007 Edwards generalized to many curves $x^2+y^2 = 1+c^4x^2y^2$. Theorem: have now obtained all elliptic curves over $\overline{\mathbf{Q}}$.

2007 Bernstein–Lange: Edwards addition law is complete for $x^2 + y^2 = 1 + dx^2y^2$ if $d \neq \blacksquare$; and gives new ECC speed records.

Representing curve points

Crypto 1985, Miller, "Use of elliptic curves in cryptography":

Given $n \in \mathbf{Z}$, $P \in E(\mathbf{F}_q)$, division-polynomial recurrence computes $nP \in E(\mathbf{F}_q)$ "in $26 \log_2 n$ multiplications"; but can do better!

"It appears to be best to represent the points on the curve in the following form:
Each point is represented by the triple (x, y, z) which corresponds to the point $(x/z^2, y/z^3)$."

1986 Chudnovsky–Chudnovsky, "Sequences of numbers generated by addition in formal groups and new primality and factorization tests":

"The crucial problem becomes the choice of the model of an algebraic group variety, where computations mod p are the least time consuming."

Most important computations:
ADD is $P, Q \mapsto P + Q$.
DBL is $P \mapsto 2P$.

"It is preferable to use models of elliptic curves lying in low-dimensional spaces, for otherwise the number of coordinates and operations is increasing. This limits us . . . to 4 basic models of elliptic curves."

Short Weierstrass:
$y^2 = x^3 + ax + b$.

Jacobi intersection:
$s^2 + c^2 = 1$, $as^2 + d^2 = 1$.

Jacobi quartic: $y^2 = x^4 + 2ax^2 + 1$.

Hessian: $x^3 + y^3 + 1 = 3dxy$.

Optimizing Jacobian coordinates

For “traditional” $(X/Z^2, Y/Z^3)$
on $y^2 = x^3 + ax + b$:
1986 Chudnovsky–Chudnovsky
state explicit formulas using
10**M** for DBL; 16**M** for ADD.

Consequence:
$$\approx \left(10 \lg n + 16 \frac{\lg n}{\lg \lg n}\right) \mathbf{M}$$
to compute $n, P \mapsto nP$
using sliding-windows method
of scalar multiplication.

Notation: $\lg = \log_2$.

Squaring is faster than **M**.

Here are the DBL formulas:

$$S = 4X_1 \cdot Y_1^2;$$
$$M = 3X_1^2 + aZ_1^4;$$
$$T = M^2 - 2S;$$
$$X_3 = T;$$
$$Y_3 = M \cdot (S - T) - 8Y_1^4;$$
$$Z_3 = 2Y_1 \cdot Z_1.$$

Total cost $3\mathbf{M} + 6\mathbf{S} + 1\mathbf{D}$ where **S** is the cost of squaring in $\mathbf{F}_q$, **D** is the cost of multiplying by a.

The squarings produce $X_1^2, Y_1^2, Y_1^4, Z_1^2, Z_1^4, M^2$.

Most ECC standards choose curves that make formulas faster.

Curve-choice advice from 1986 Chudnovsky–Chudnovsky:

Can eliminate the 1**D** by choosing curve with $a = 1$.

But "it is even smarter" to choose curve with $a = -3$.

If $a = -3$ then $M = 3(X_1^2 - Z_1^4)$
$= 3(X_1 - Z_1^2) \cdot (X_1 + Z_1^2)$.
Replace 2**S** with 1**M**.

Now DBL costs $4\mathbf{M} + 4\mathbf{S}$.

2001 Bernstein:
$3\mathbf{M} + 5\mathbf{S}$ for DBL.
$11\mathbf{M} + 5\mathbf{S}$ for ADD.

How? Easy $\mathbf{S} - \mathbf{M}$ tradeoff:
instead of computing $2Y_1 \cdot Z_1$,
compute $(Y_1 + Z_1)^2 - Y_1^2 - Z_1^2$.
DBL formulas were already
computing Y_1^2 and Z_1^2.

Same idea for the ADD formulas,
but have to scale X, Y, Z
to eliminate divisions by 2.

ADD for $y^2 = x^3 + ax + b$:
$U_1 = X_1 Z_2^2$, $U_2 = X_2 Z_1^2$,
$S_1 = Y_1 Z_2^3$, $S_2 = Y_2 Z_1^3$,
many more computations.

1986 Chudnovsky–Chudnovsky:
"We suggest to write
addition formulas involving
(X, Y, Z, Z^2, Z^3)."

Disadvantages:
Allocate space for Z^2, Z^3.
Pay $1\mathbf{S} + 1\mathbf{M}$ in ADD and in DBL.

Advantages:
Save $2\mathbf{S} + 2\mathbf{M}$ at start of ADD.
Save $1\mathbf{S}$ at start of DBL.

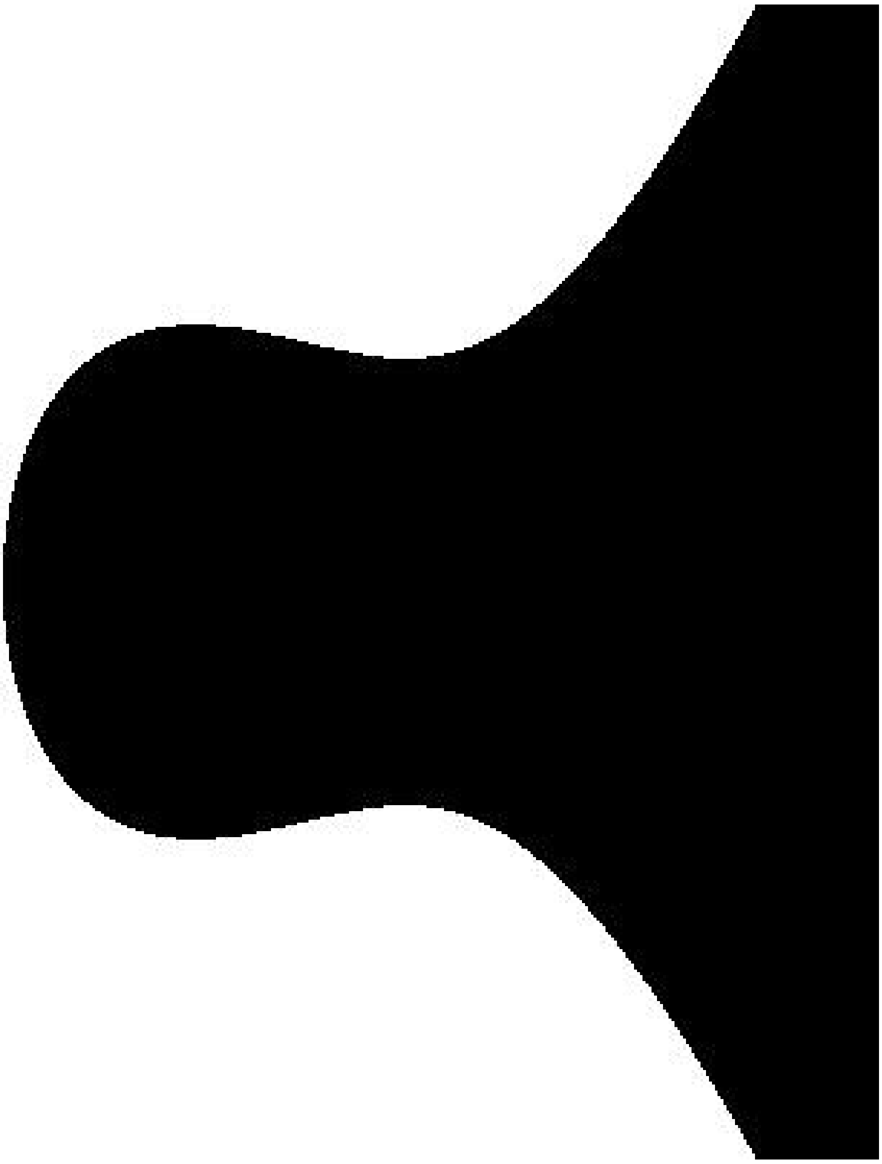
1998 Cohen–Miyaji–Ono:
Store point as $(X : Y : Z)$.
If point is input to ADD,
also cache Z^2 and Z^3.
No cost, aside from space.
If point is input to another ADD,
reuse Z^2, Z^3. Save $1\mathbf{S} + 1\mathbf{M}$!

Best Jacobian speeds today,
including $\mathbf{S} - \mathbf{M}$ tradeoffs:
$3\mathbf{M} + 5\mathbf{S}$ for DBL if $a = -3$.
$11\mathbf{M} + 5\mathbf{S}$ for ADD.
$10\mathbf{M} + 4\mathbf{S}$ for reADD.
$7\mathbf{M} + 4\mathbf{S}$ for mADD (i.e. $Z_2 = 1$).

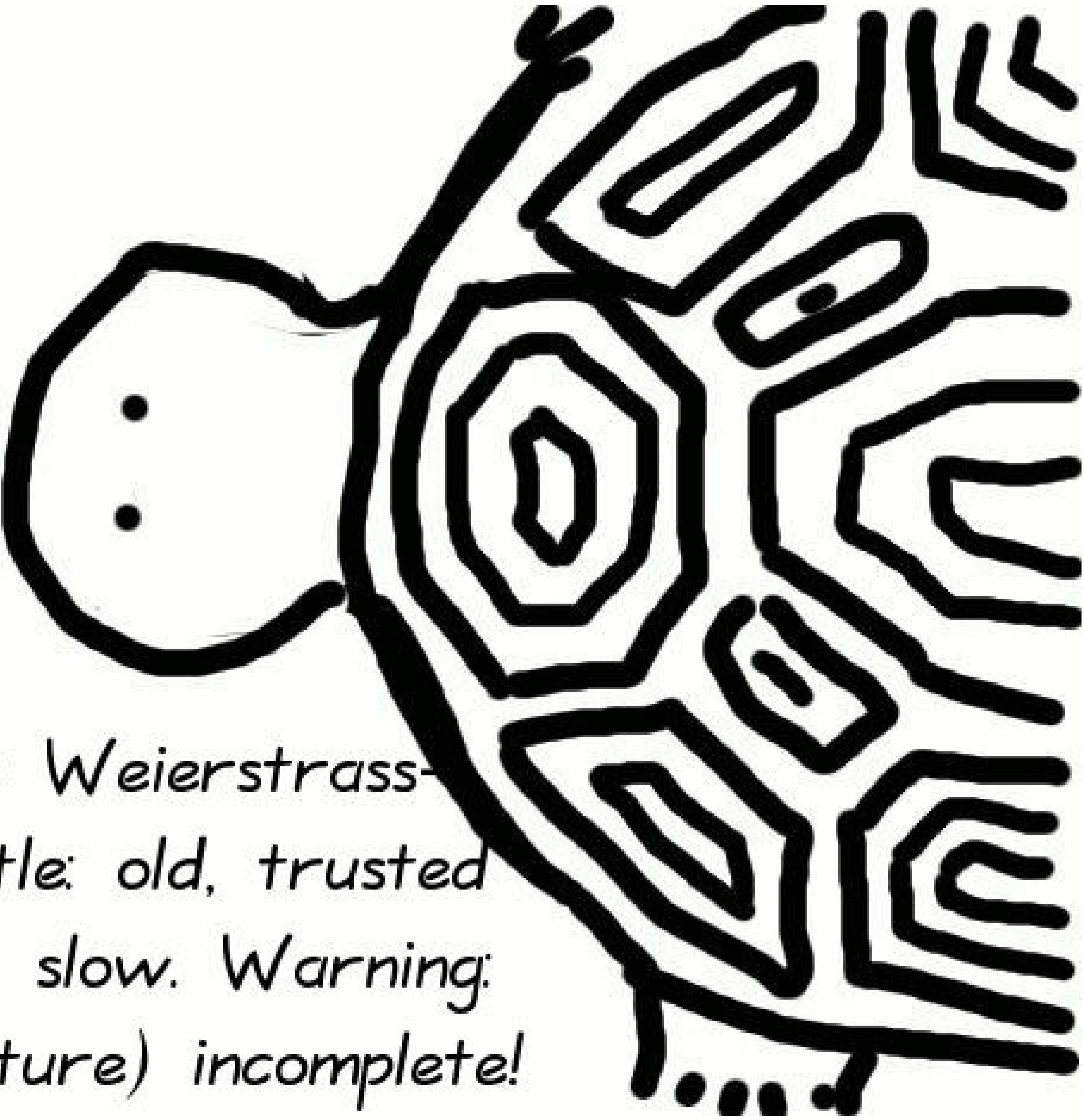
Compare to speeds for Edwards curves $x^2 + y^2 = 1 + dx^2y^2$ in projective coordinates (2007 Bernstein–Lange):
$3\mathbf{M} + 4\mathbf{S}$ for DBL.
$10\mathbf{M} + 1\mathbf{S} + 1\mathbf{D}$ for ADD.
$9\mathbf{M} + 1\mathbf{S} + 1\mathbf{D}$ for mADD.

Inverted Edwards coordinates (2007 Bernstein–Lange):
$3\mathbf{M} + 4\mathbf{S} + 1\mathbf{D}$ for DBL.
$9\mathbf{M} + 1\mathbf{S} + 1\mathbf{D}$ for ADD.
$8\mathbf{M} + 1\mathbf{S} + 1\mathbf{D}$ for mADD.

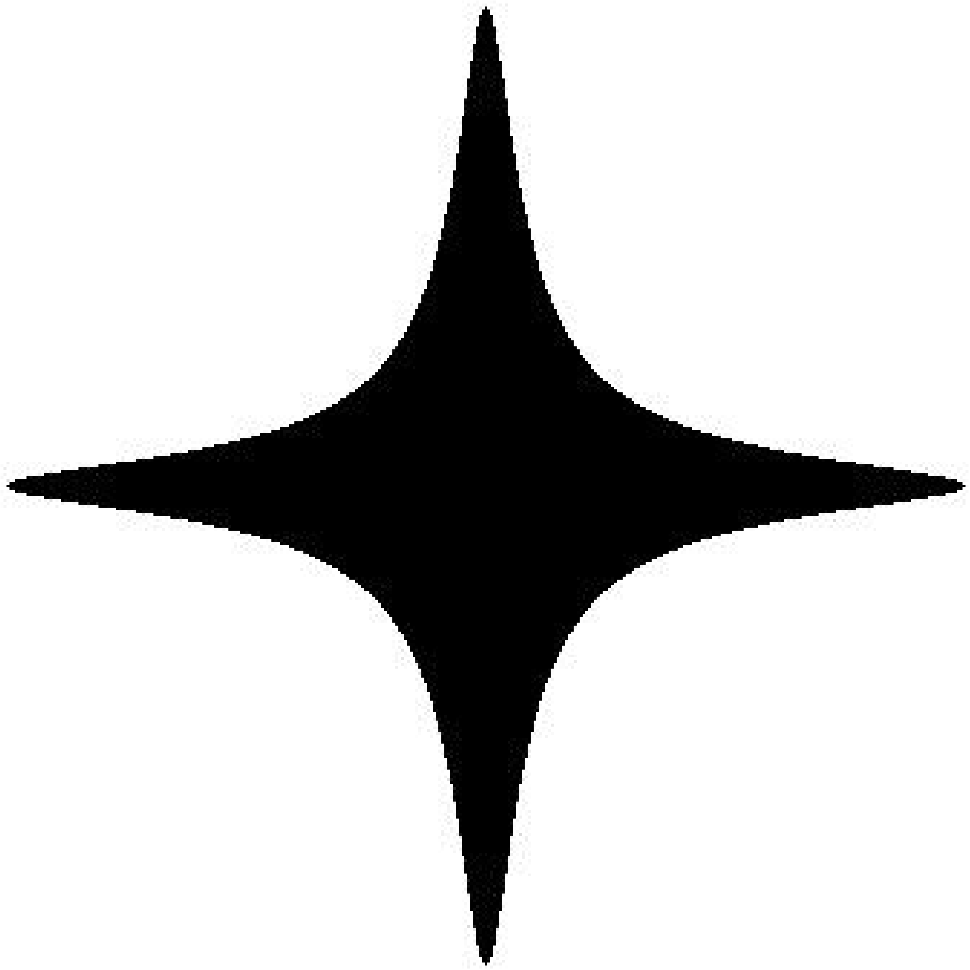
Even better speeds from extended/completed coordinates (2008 Hisil–Wong–Carter–Dawson).



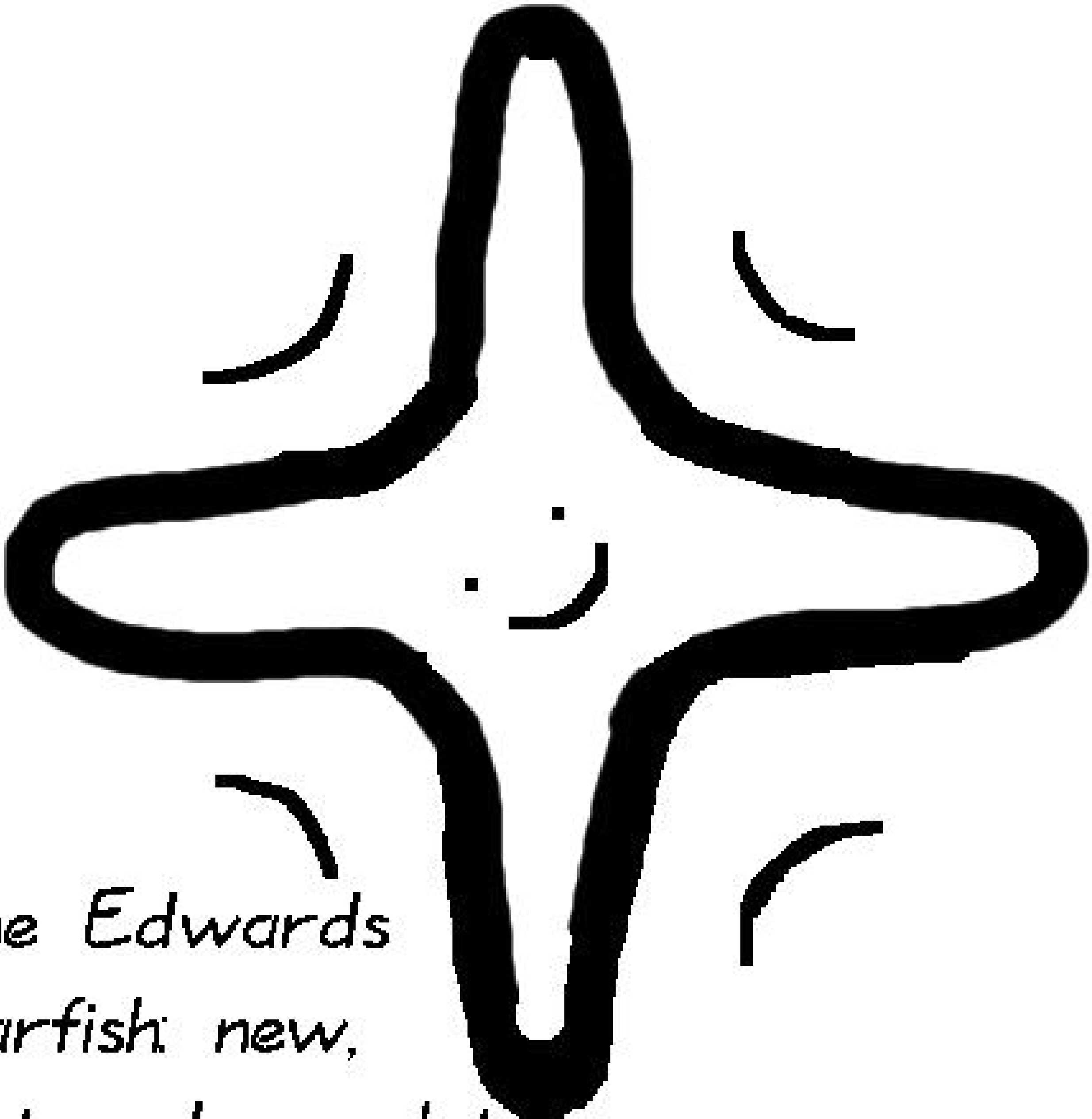
$$y^2 = x^3 - 0.4x + 0.7$$



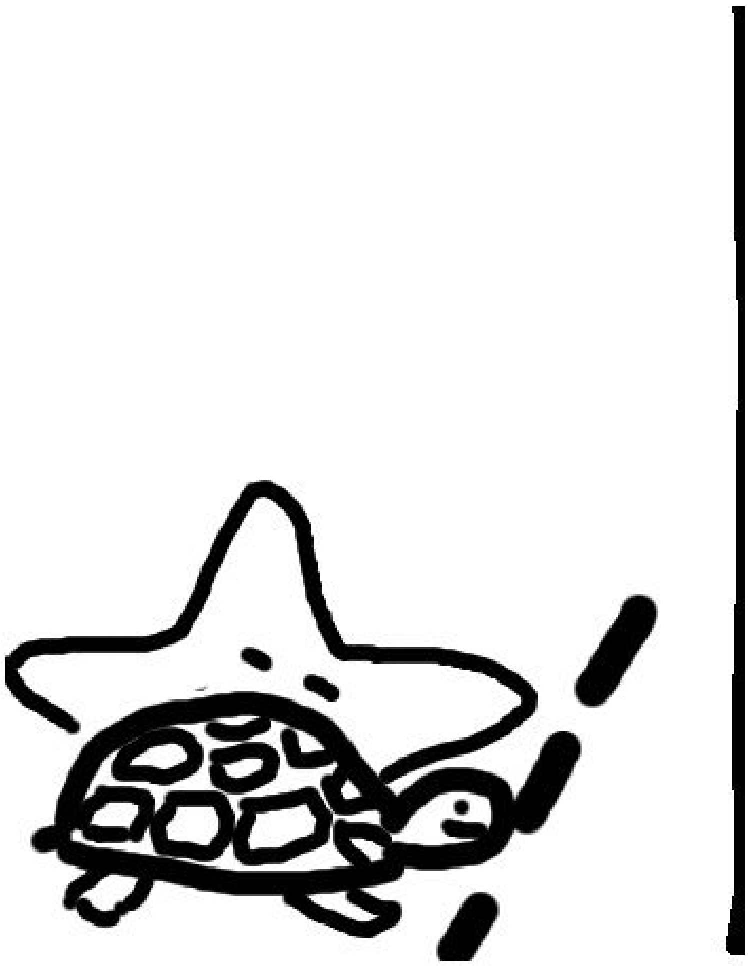
The Weierstrass-turtle: old, trusted and slow. Warning: (picture) incomplete!



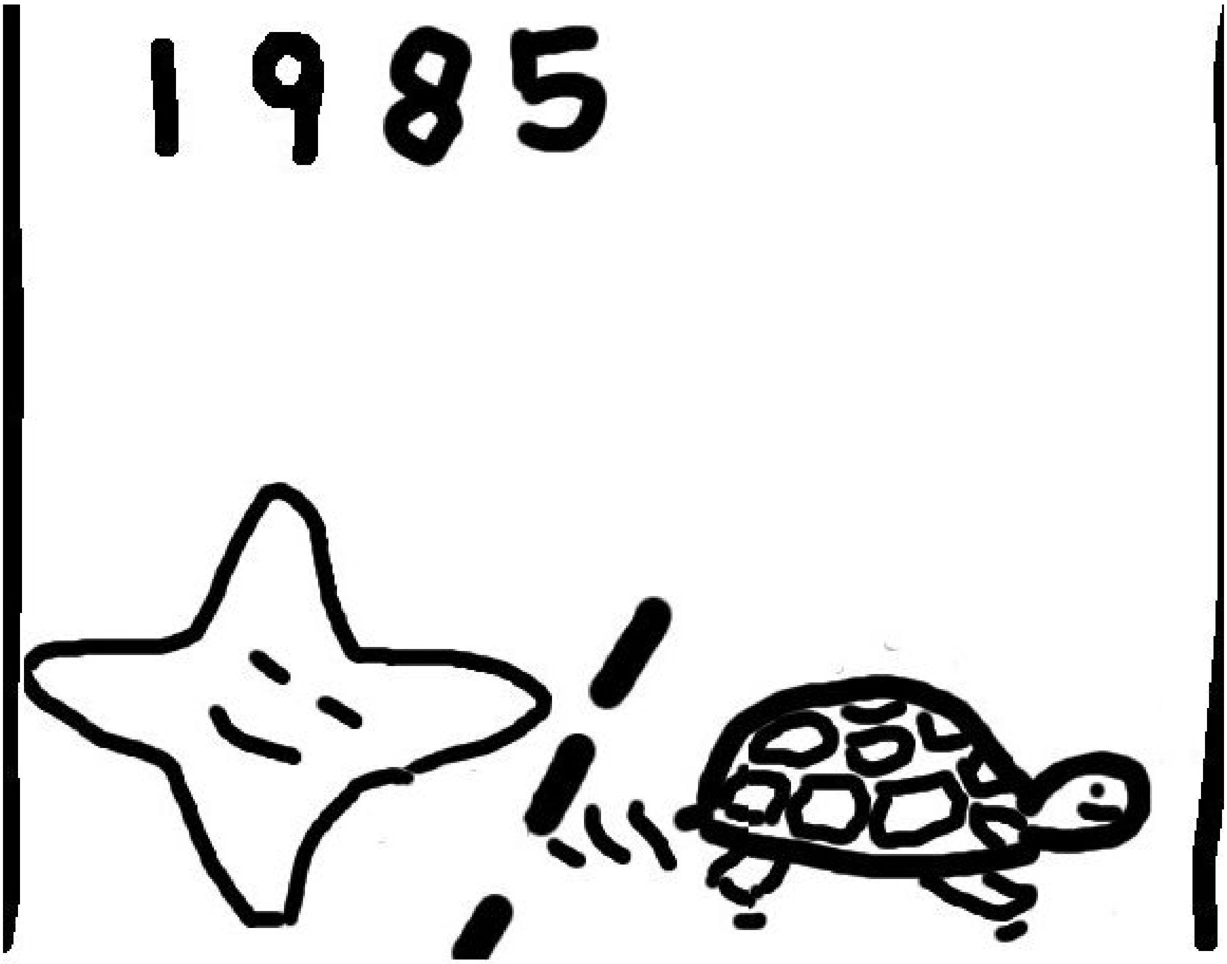
$$x^2 + y^2 = 1 - 300x^2y^2$$



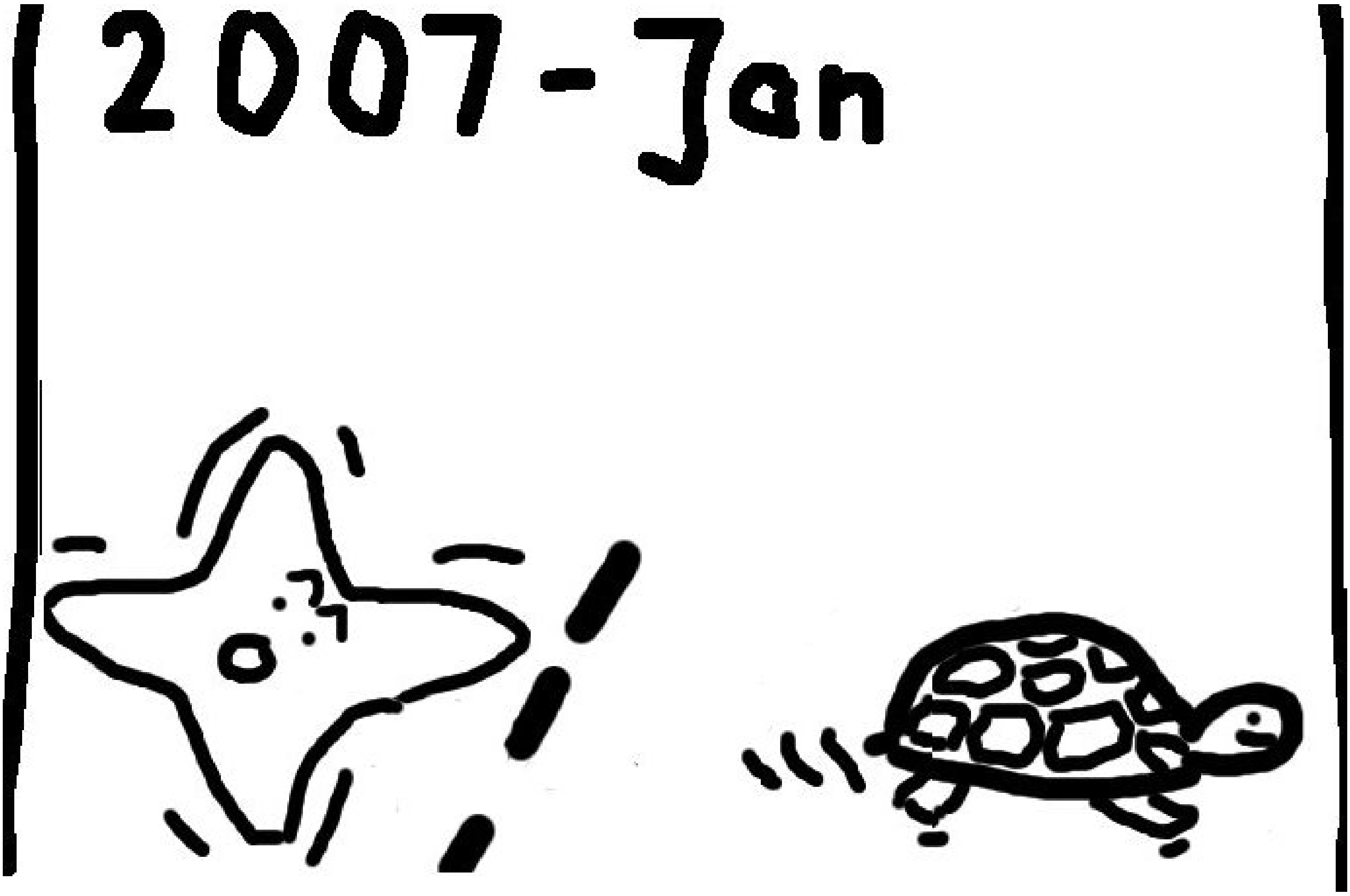
The Edwards starfish: new, fast and complete!



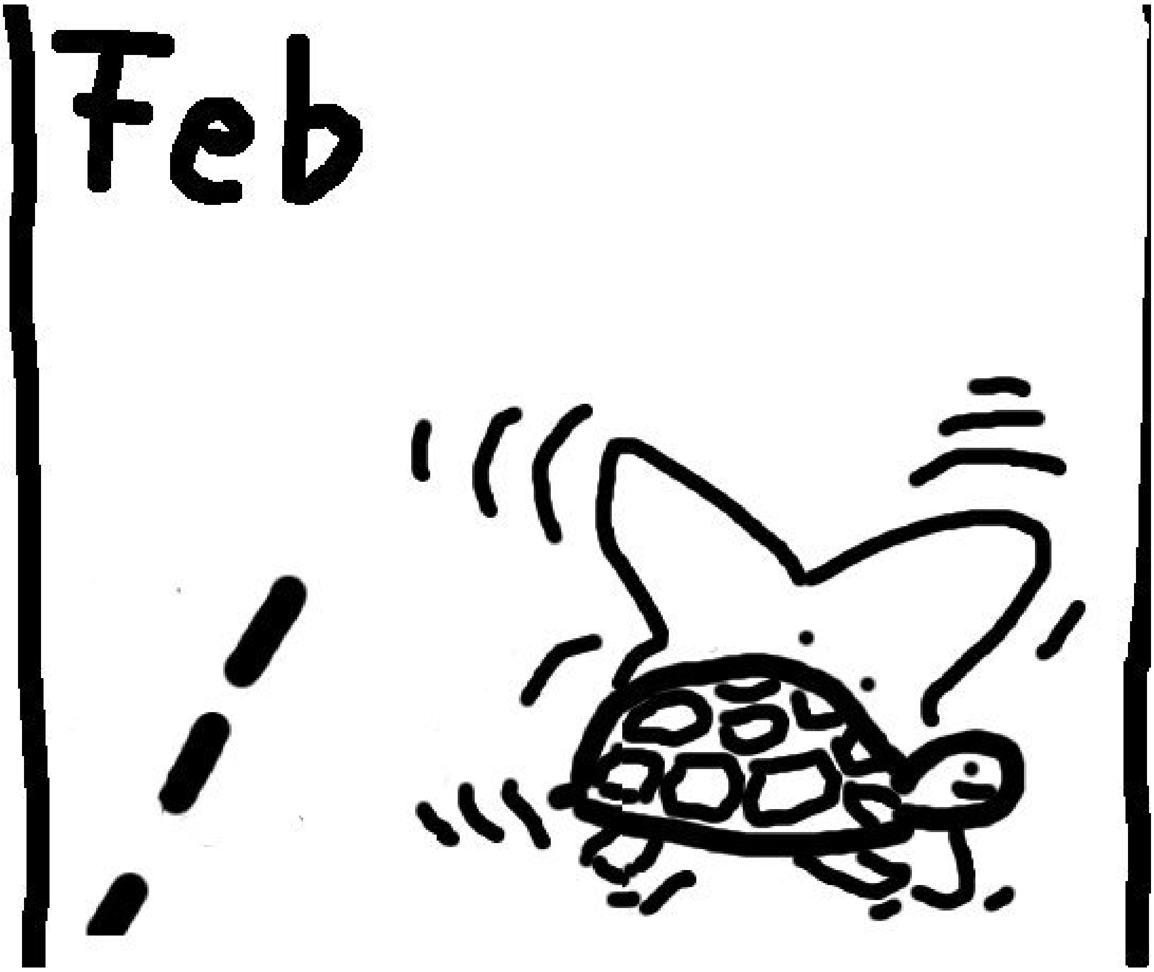
Start!



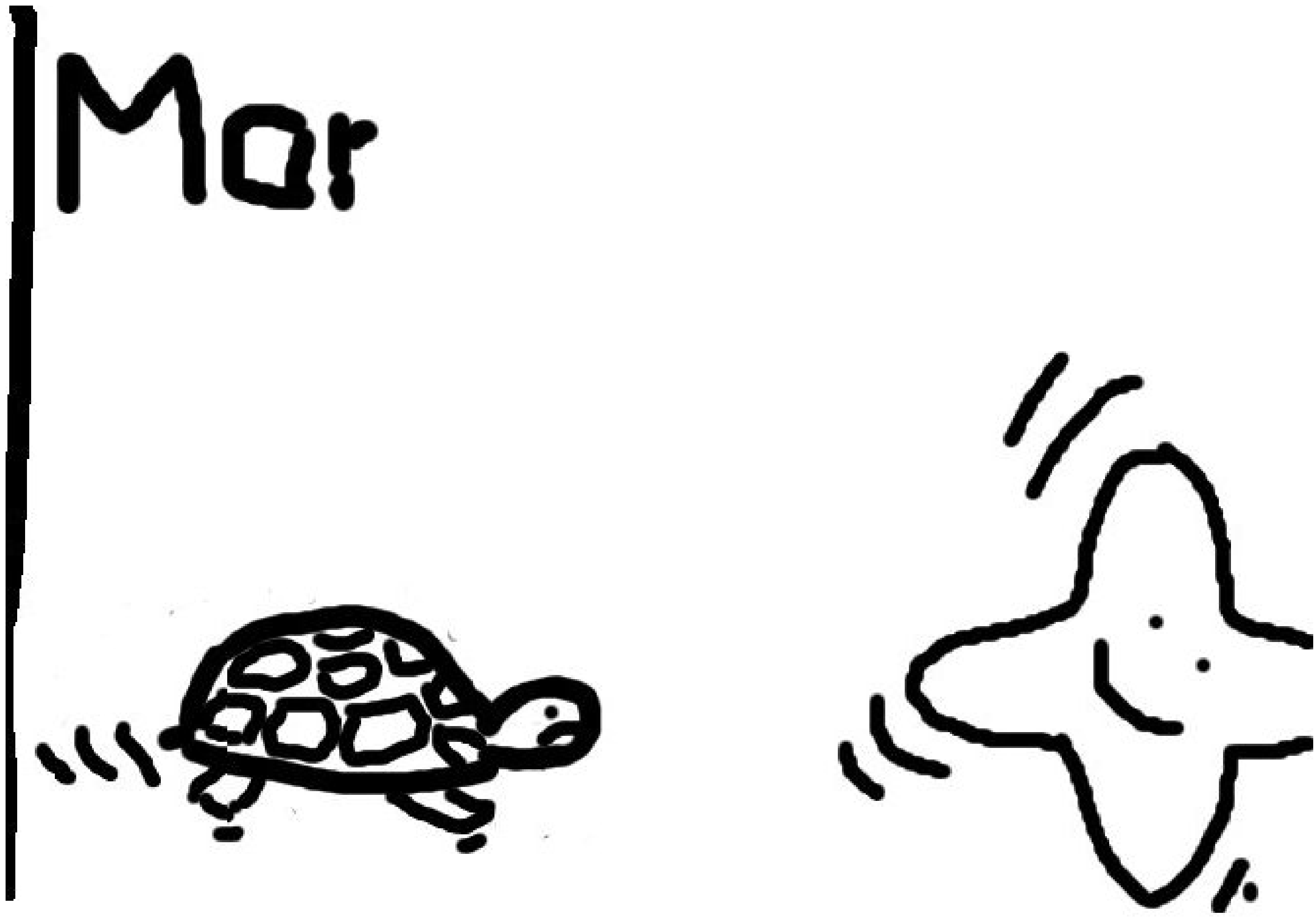
Weierstrass sets off, Edwards left behind sleeping



Weierstrass has made some progress ...
finally Edwards wakes up.



Exciting progress: Edwards about to overtake!!



And the winner is: Edwards!