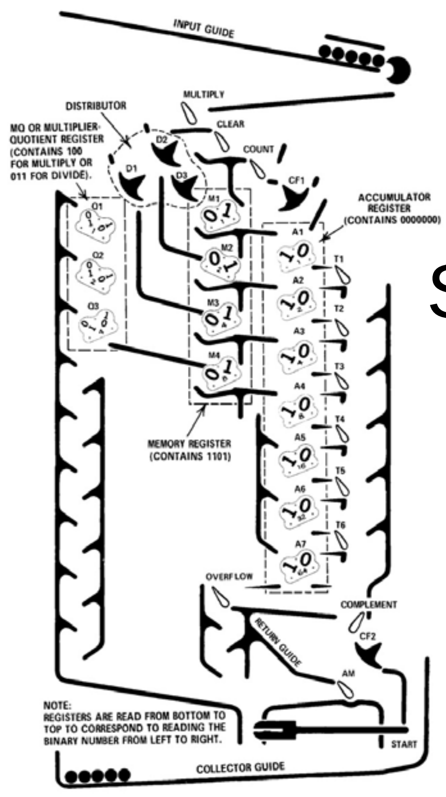

INPUT GUIDE
MULTIPLY
DISTRIBUTOR
MQ OR MULTIPLIER-QUOTIENT REGISTER (CONTAINS 100 FOR MULTIPLY OR 011 FOR DIVIDE).
CLEAR
COUNT
D1
D2
D3
CF1
ACCUMULATOR REGISTER (CONTAINS 0000000)
Q1
Q2
Q3
M1
M2
M3
M4
A1
A2
A3
A4
A5
A6
A7
T1
T2
T3
T4
T5
T6
MEMORY REGISTER (CONTAINS 1101)
OVERFLOW
COMPLEMENT
CF2
RETURN GUIDE
AM
NOTE: REGISTERS ARE READ FROM BOTTOM TO TOP TO CORRESPOND TO READING THE BINARY NUMBER FROM LEFT TO RIGHT.
START
COLLECTOR GUIDE

Speedy Maths

David McQuillan

Basic Arithmetic

What one needs to be able to do

- Addition and Subtraction
- Multiplication and Division
- Comparison

For a number of order 2^n

n ~ 100 is general multi precision arithmetic,
large *n* is 1000 for crypto applications
huge *n* is 100000 for maths investigations

The Plan

- Basic arithmetic on a computer
- Long multiply via three smaller ones
- Multiplying via polynomials
- Multiplying Matrices

- The Schönhage Strassen algorithm
- Use and future

Addition and Subtraction

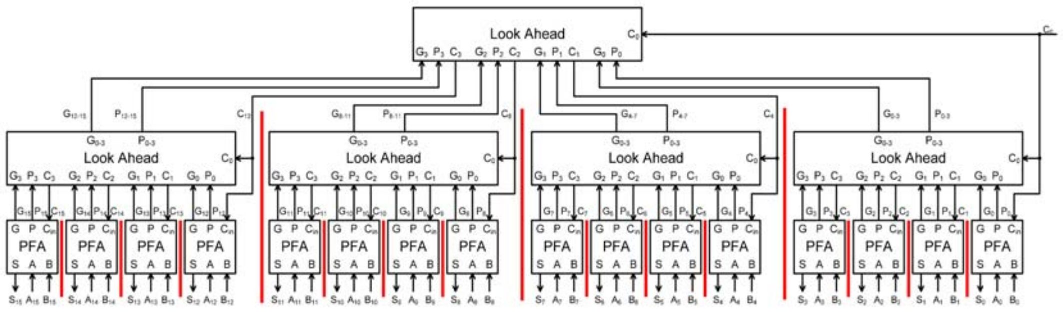
- Time to add two huge *n* bit numbers 'by hand'

  $\Theta(n)$

  Constant factors are ignored – though they can make all the difference in the real world.

- For small *n* hardware can calculate the sum in time $\Theta(\log n)$ using 'divide and conquer'.

Carry look ahead adder



Calculate the carry for both that there is a carry in and that there isn't. Total time $\Theta(\log n)$.

There are only $\log_2 n$ layers and each layer takes a constant time. Thus the fll add just need a signal to go up the hierarchy, the top one to make the decisions that go down and that sets the result, something like $2\log_2 n$ steps or of the order of log n.

Modulo Arithmetic

Addition and multiplication operations commute with the modulo operation

29 = 1 mod 7

13+16 = 6+2 = 8 = 1 mod 7

208 = 5 mod 7

13×16= 6×2 = 12 = 5 mod 7

Fast addition and multiplication

Use a number of different moduli

(5,6,7) are relatively prime

Accurate if result is less than $5\times6\times7 = 210$

$6 = (1,0,6)$, $7 = (2,1,0)$, $8 = (3,2,1)$, $9 = (4,3,2)$

$6\times9 = (4,0,5)$ $7\times8 = (1,2,0)$

But which is larger?

Expensive to do a comparison

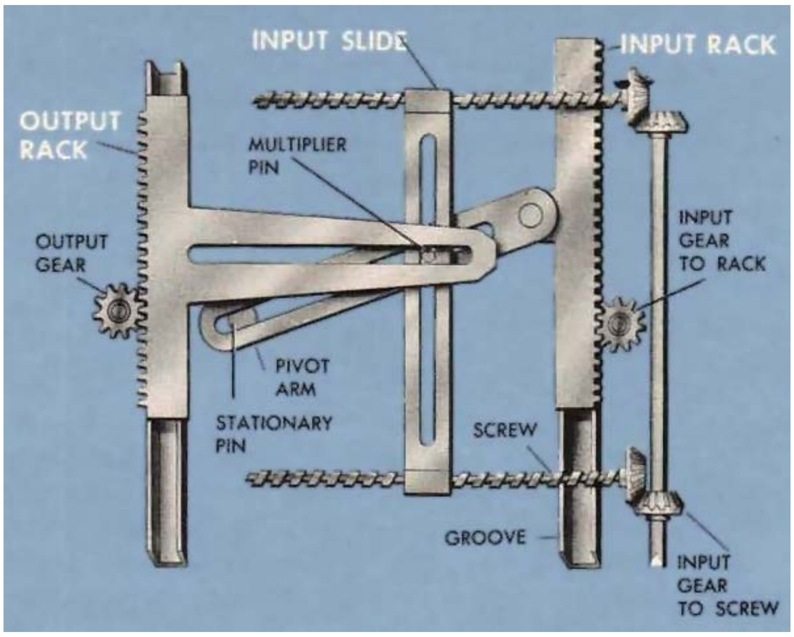
Chinese Remainder Theorem

Chinese Remainder theorem goes from the remainders to a number giving that remainder modulo the least common multiple of the moduli.

Can be implemented using divide and conquer.

Modular arithmetic used extensively even sometimes when division is required.

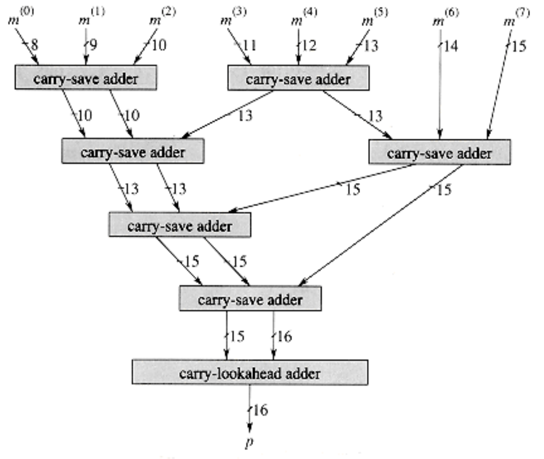
We want proper multiplication



From US Navy Ordnance Pamphlet 1140 “Basic Fire Control Mechanisms”. WW2 fire control computer for a ship.

Fast multiplication

- Hardware can multiply two *n* bit numbers in time $\Theta(\log n)$ using a 'Wallace tree multiplier'



8 partial products
Wallace tree

From 'Introduction to Algorithms' by Cormen, Leiserson and Rivest

Each box is composed of a number of independent full adders that sum three bits inputs and produce two bits – the sum and carry. The carries can all be treated as another numebr with a zero at the end. Thus three input numbers become two output numbers.
The three to two goes on till there are only two numbers and these are added using the carry lookahead adder. There are only a bit over $\log_2 3n$ layers which each take constant time plus the carry lookahead which is also of the order log n so the whole multiply can be done in time of order log n.

Complex multiplication

The first faster multiply was by Gauss multiplying two complex numbers

$$(a + bi) \cdot (c + di)$$

$$v_2 = a \cdot c$$

$$v_1 = (a + b) \cdot (c + d)$$

$$v_0 = b \cdot d$$

$$= (v_2 - v_0) + (v_1 - v_2 - v_0)i$$

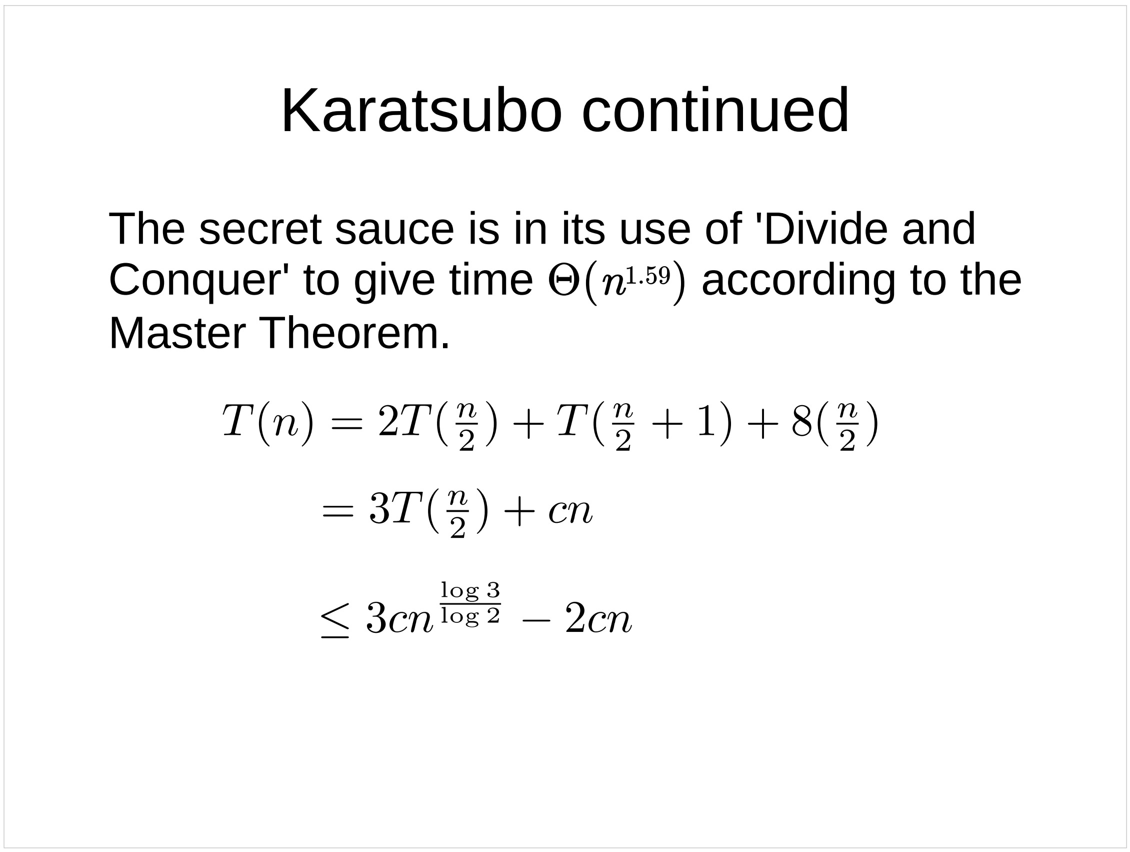
Karatsuba algorithm

- Discovered 1960, starts like Gauss's algorithm

$$(10^n p_1 + p_0) \cdot (10^n q_1 + q_0)$$

$$\begin{aligned} v_2 &= p_1 \cdot q_1 \\ v_1 &= (p_1 + p_0) \cdot (q_1 + q_0) \\ v_0 &= p_0 \cdot q_0 \end{aligned}$$

$$= 10^{2n} v_2 + 10^n (v_1 - v_2 - v_0) + v_0$$



Karatsubo continued

The secret sauce is in its use of 'Divide and Conquer' to give time $\Theta(n^{1.59})$ according to the Master Theorem.

$$T(n) = 2T(\tfrac{n}{2}) + T(\tfrac{n}{2} + 1) + 8(\tfrac{n}{2})$$

$$= 3T(\tfrac{n}{2}) + cn$$

$$\leq 3cn^{\frac{\log 3}{\log 2}} - 2cn$$

Rather a grand name for a theorem!

Can we do better?

Start on a strange path.....

A polynomial *p*(*x*) of degree *n* can be calculated from its value at *n*+1 points by using Lagrange's interpolation formula

$$p(x) = \sum_{j=0}^{n} p_j \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{(x - x_k)}{(x_j - x_k)}$$

- Gives a polynomial with *n*+1 coefficients

Multiplying polynomials

- A number can be considered as a polynomial evaluated at say 10^k, e.g. $12\times10^4+34\times10^2+56$
- The product of two polynomials $p(x)\cdot q(x)$ of degrees n and m is of degree $n+m$.
- Therefore the product polynomial is determined by the product of the values of $p(x)$ and $q(x)$ at $n+m+1$ points.

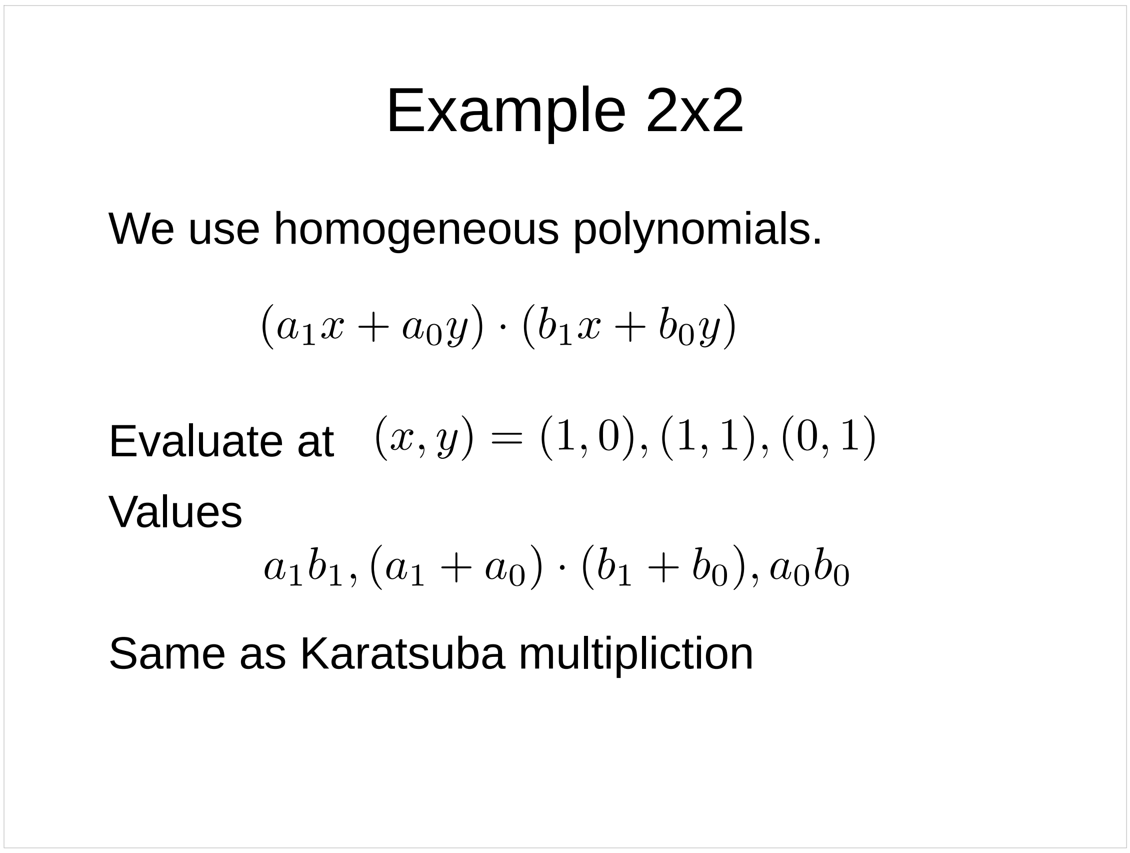
Toom Cook

Evaluate the numbers considered as polynomials at some small numbers instead of 10^k

Multiply the smaller numbers together

Use the interpolation formula to get the product polynomial

Stick in 10^k for the variable



Example 2x2

We use homogeneous polynomials.

$$(a_1 x + a_0 y) \cdot (b_1 x + b_0 y)$$

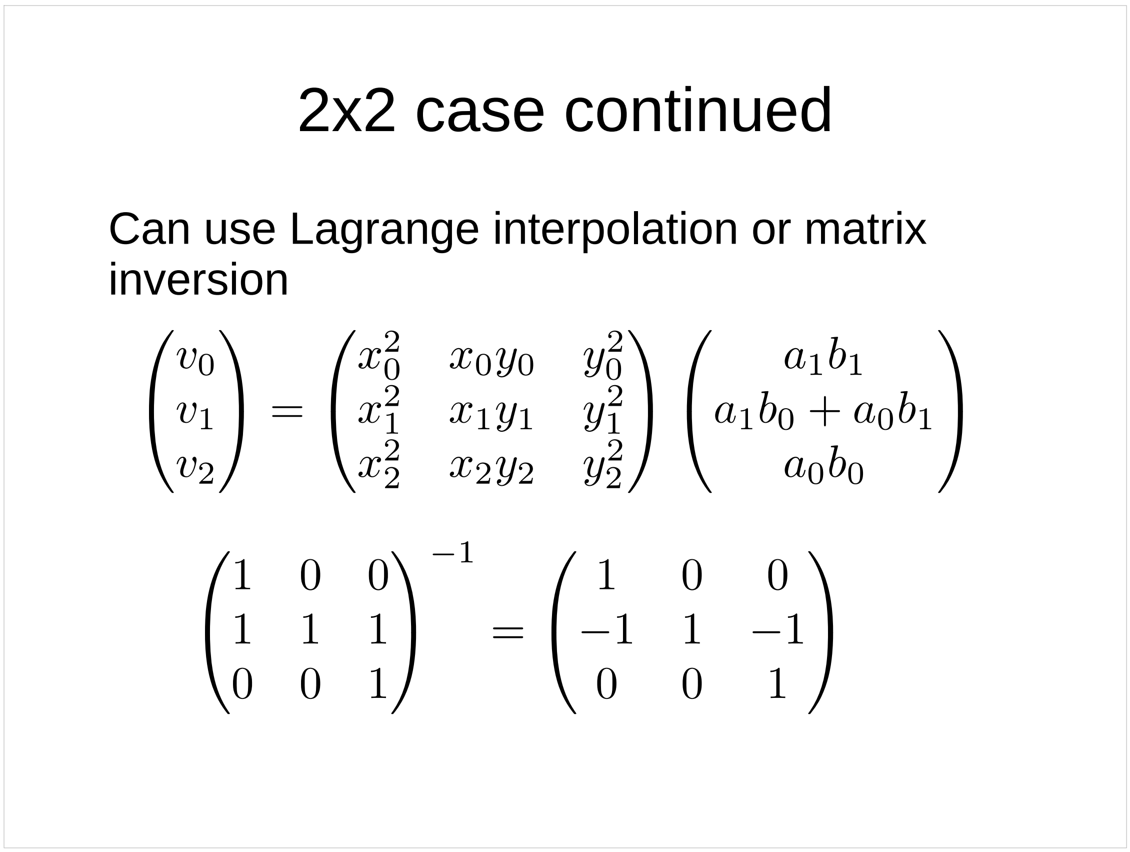
Evaluate at $(x, y) = (1, 0), (1, 1), (0, 1)$

Values

$$a_1 b_1, (a_1 + a_0) \cdot (b_1 + b_0), a_0 b_0$$

Same as Karatsuba multipliction

Using *x* and *y* so the total degree is alway constant. Useful so (1,0) can represent the pioint at infinity which is a convenient simple poinyt. Any three points could be used and only still only has 3 multiplications but the adsss and subtracts get more complicate.



2x2 case continued

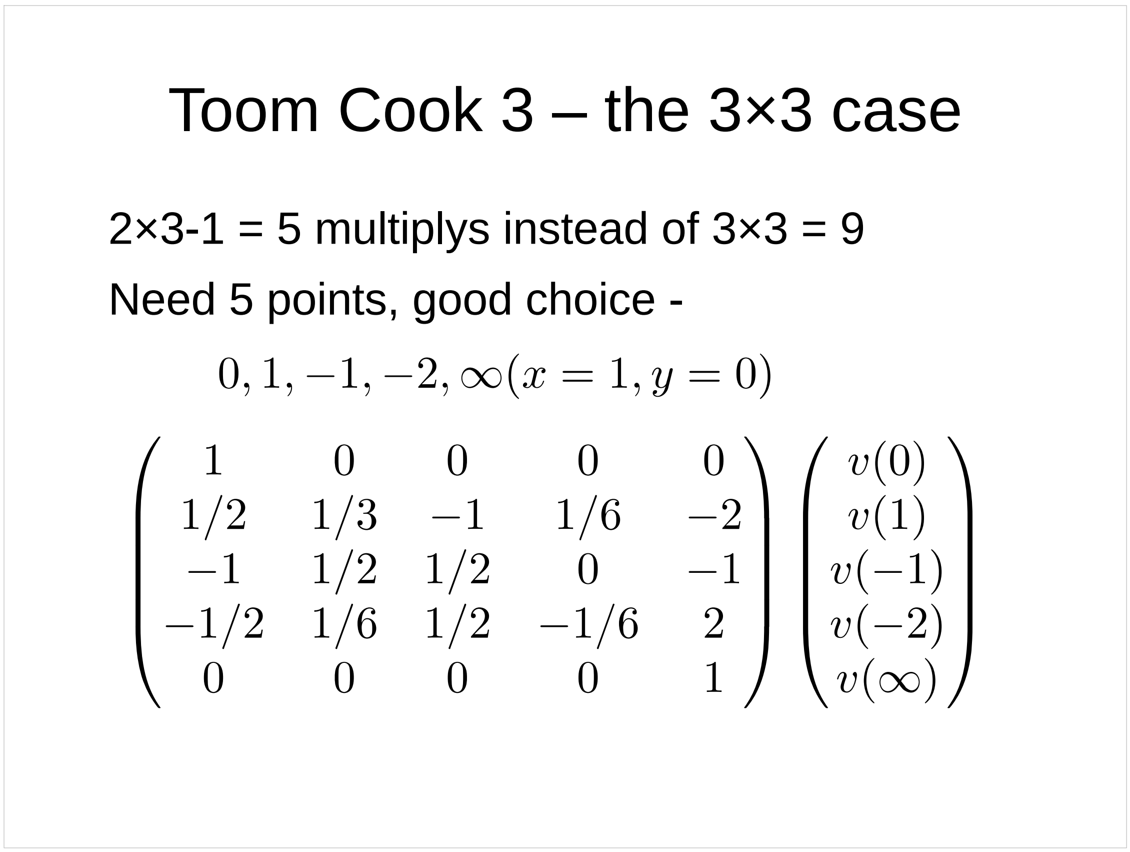
Can use Lagrange interpolation or matrix inversion

$$\begin{pmatrix} v_0 \\ v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} x_0^2 & x_0 y_0 & y_0^2 \\ x_1^2 & x_1 y_1 & y_1^2 \\ x_2^2 & x_2 y_2 & y_2^2 \end{pmatrix} \begin{pmatrix} a_1 b_1 \\ a_1 b_0 + a_0 b_1 \\ a_0 b_0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

The -1 1 -1 corresponds to v_1-v_0-v_2

Exactly the same as Karatsuba mutiplication when these three piints use.



Toom Cook 3 – the 3×3 case

2×3-1 = 5 multiplys instead of 3×3 = 9

Need 5 points, good choice -

$$0, 1, -1, -2, \infty(x = 1, y = 0)$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1/2 & 1/3 & -1 & 1/6 & -2 \\ -1 & 1/2 & 1/2 & 0 & -1 \\ -1/2 & 1/6 & 1/2 & -1/6 & 2 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v(0) \\ v(1) \\ v(-1) \\ v(-2) \\ v(\infty) \end{pmatrix}$$

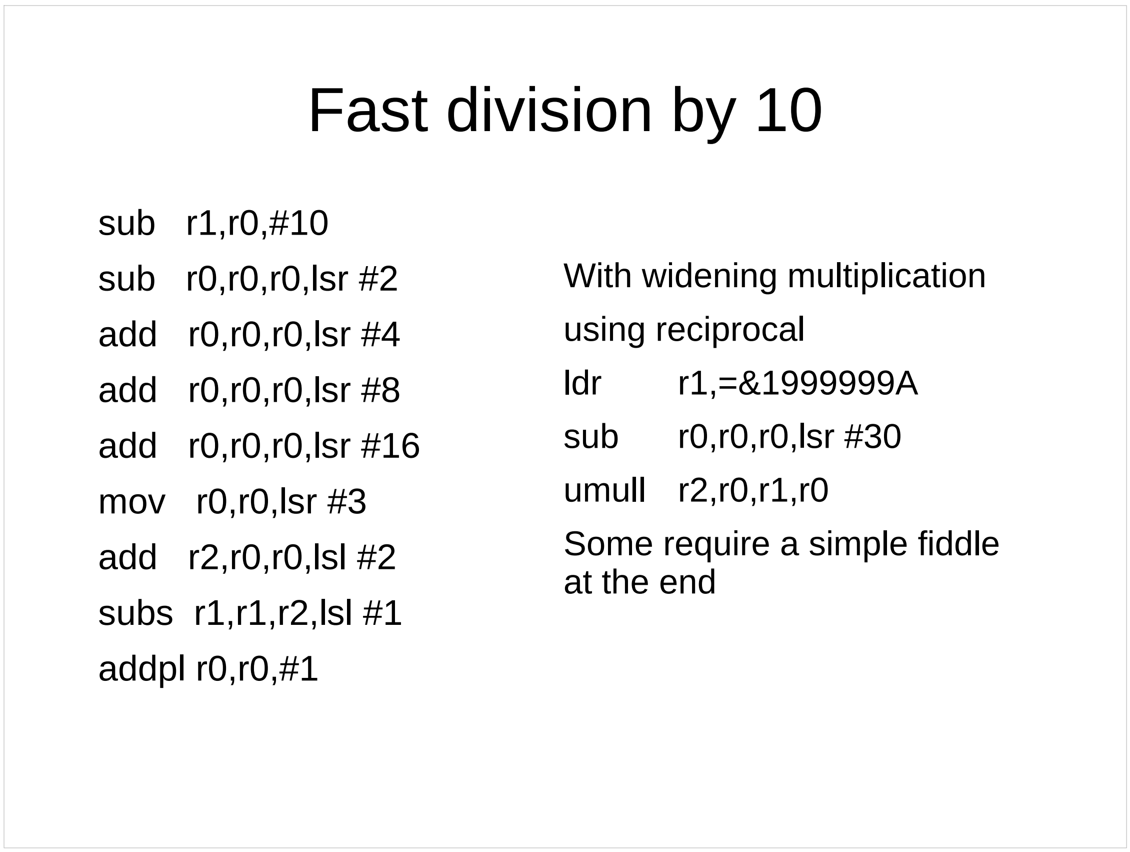
Rather a bit more complicated as factors 1/3 and 1/6 need to be applied. These are quite easy to evaluate though even without a hardware divide.

No free lunch

Need to do lots of adds and subtracts plus multiplys and divides by small integers.

In fact divides by small constants can be done quite efficiently.

Toom Cook gets unwieldy fairly quickly and recursion as for Karatsubo is used to split the problem into manageable chunks.



Fast division by 10

```
sub   r1,r0,#10
sub   r0,r0,r0,lsr #2
add   r0,r0,r0,lsr #4
add   r0,r0,r0,lsr #8
add   r0,r0,r0,lsr #16
mov   r0,r0,lsr #3
add   r2,r0,r0,lsl #2
subs  r1,r1,r2,lsl #1
addpl r0,r0,#1
```

With widening multiplication

using reciprocal

```
ldr     r1,=&1999999A
sub     r0,r0,r0,lsr #30
umull   r2,r0,r1,r0
```

Some require a simple fiddle at the end

Sort of thing I used to do at one time in the past. First column just uses shifts and add or subtract to divide by 10. The other version uses a 'reciprocal' of 10, in fact $2^{32}/10+1$ in this case.

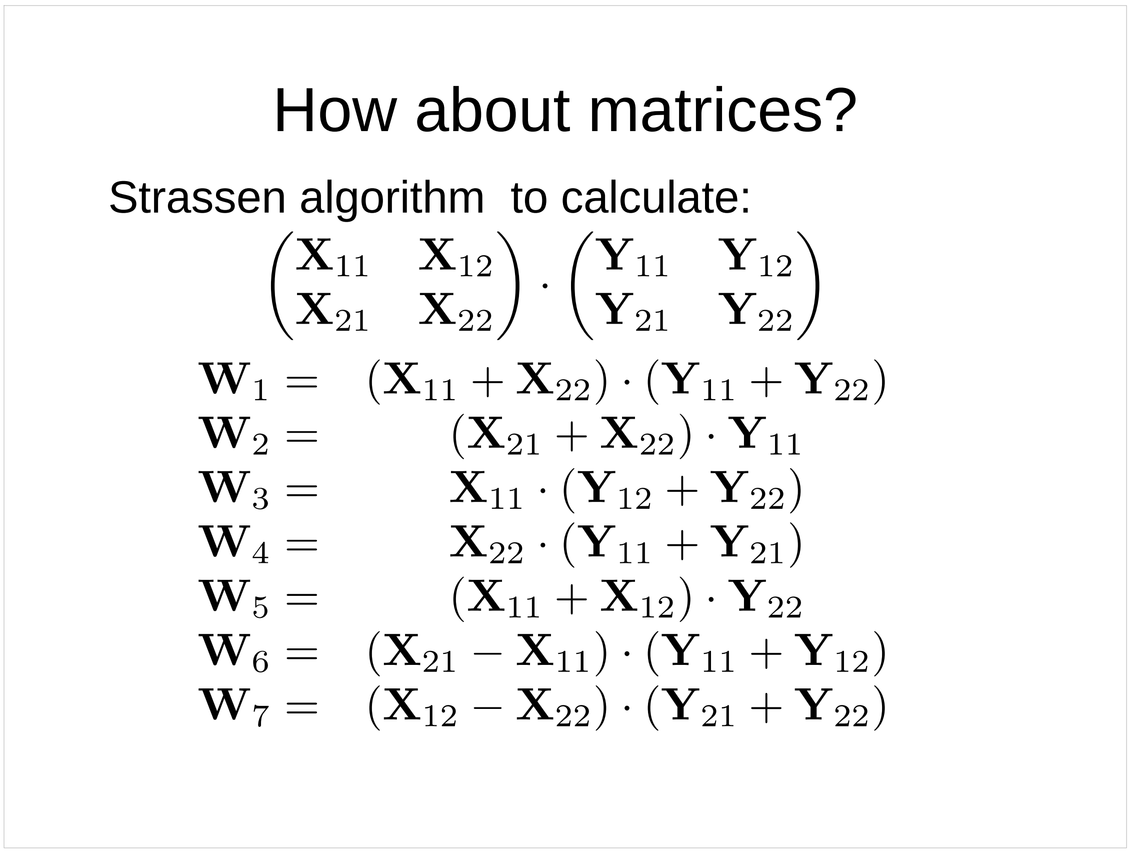
So where are we?

Toom Cook best for 'large' crypto type numbers.

Toom-3 is $\Theta(n^{\log 5/\log 3}) \sim \Theta(n^{1.465})$

Toom-5 is the normal biggest used but up to Toom-8 has been used which gives a time $\Theta(n^{\log 15/\log 8}) \sim \Theta(n^{1.302})$

- But can do much better for huge numbers



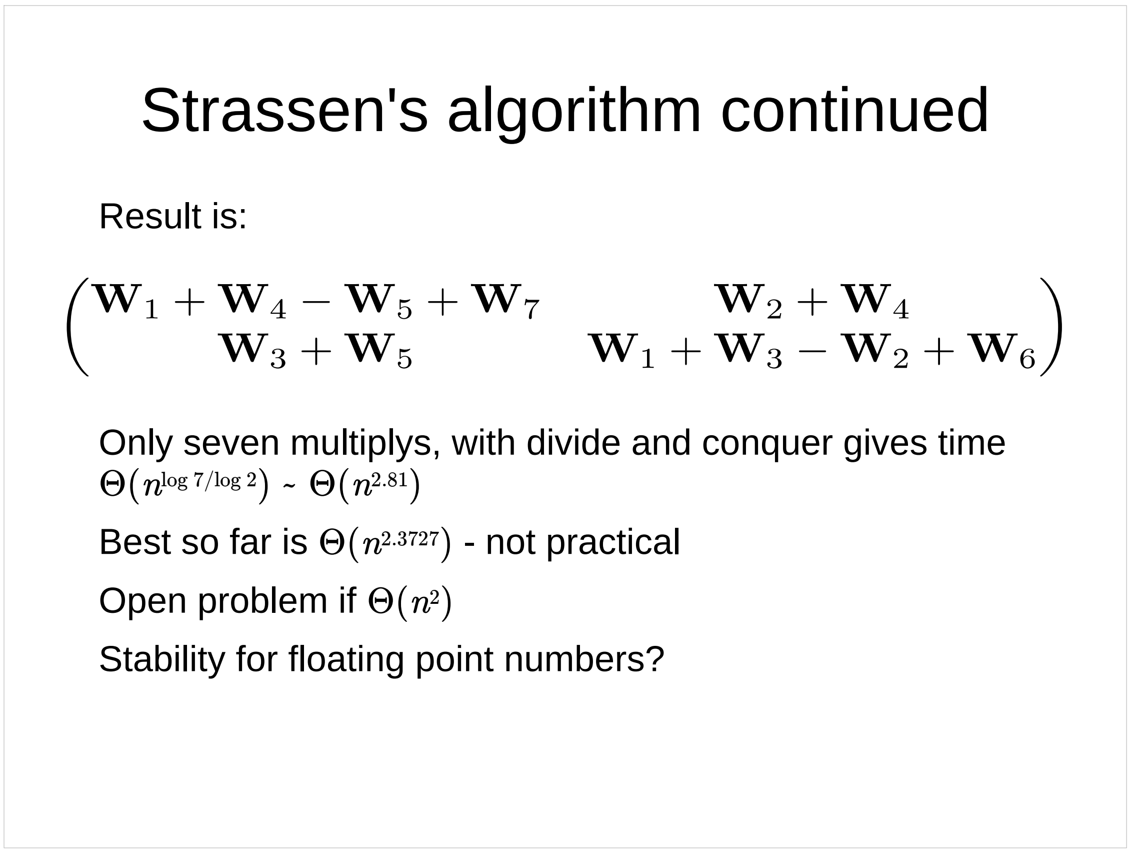
How about matrices?

Strassen algorithm to calculate:

$$\begin{pmatrix} \mathbf{X}_{11} & \mathbf{X}_{12} \\ \mathbf{X}_{21} & \mathbf{X}_{22} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \\ \mathbf{Y}_{21} & \mathbf{Y}_{22} \end{pmatrix}$$

$$\begin{aligned}
\mathbf{W}_1 &= (\mathbf{X}_{11} + \mathbf{X}_{22}) \cdot (\mathbf{Y}_{11} + \mathbf{Y}_{22}) \\
\mathbf{W}_2 &= (\mathbf{X}_{21} + \mathbf{X}_{22}) \cdot \mathbf{Y}_{11} \\
\mathbf{W}_3 &= \mathbf{X}_{11} \cdot (\mathbf{Y}_{12} + \mathbf{Y}_{22}) \\
\mathbf{W}_4 &= \mathbf{X}_{22} \cdot (\mathbf{Y}_{11} + \mathbf{Y}_{21}) \\
\mathbf{W}_5 &= (\mathbf{X}_{11} + \mathbf{X}_{12}) \cdot \mathbf{Y}_{22} \\
\mathbf{W}_6 &= (\mathbf{X}_{21} - \mathbf{X}_{11}) \cdot (\mathbf{Y}_{11} + \mathbf{Y}_{12}) \\
\mathbf{W}_7 &= (\mathbf{X}_{12} - \mathbf{X}_{22}) \cdot (\mathbf{Y}_{21} + \mathbf{Y}_{22})
\end{aligned}$$

A small interlude to look at matrices
The various X,Y,W are matrices of half the order of the original.



Strassen's algorithm continued

Result is:

$$\begin{pmatrix} \mathbf{W}_1 + \mathbf{W}_4 - \mathbf{W}_5 + \mathbf{W}_7 & \mathbf{W}_2 + \mathbf{W}_4 \\ \mathbf{W}_3 + \mathbf{W}_5 & \mathbf{W}_1 + \mathbf{W}_3 - \mathbf{W}_2 + \mathbf{W}_6 \end{pmatrix}$$

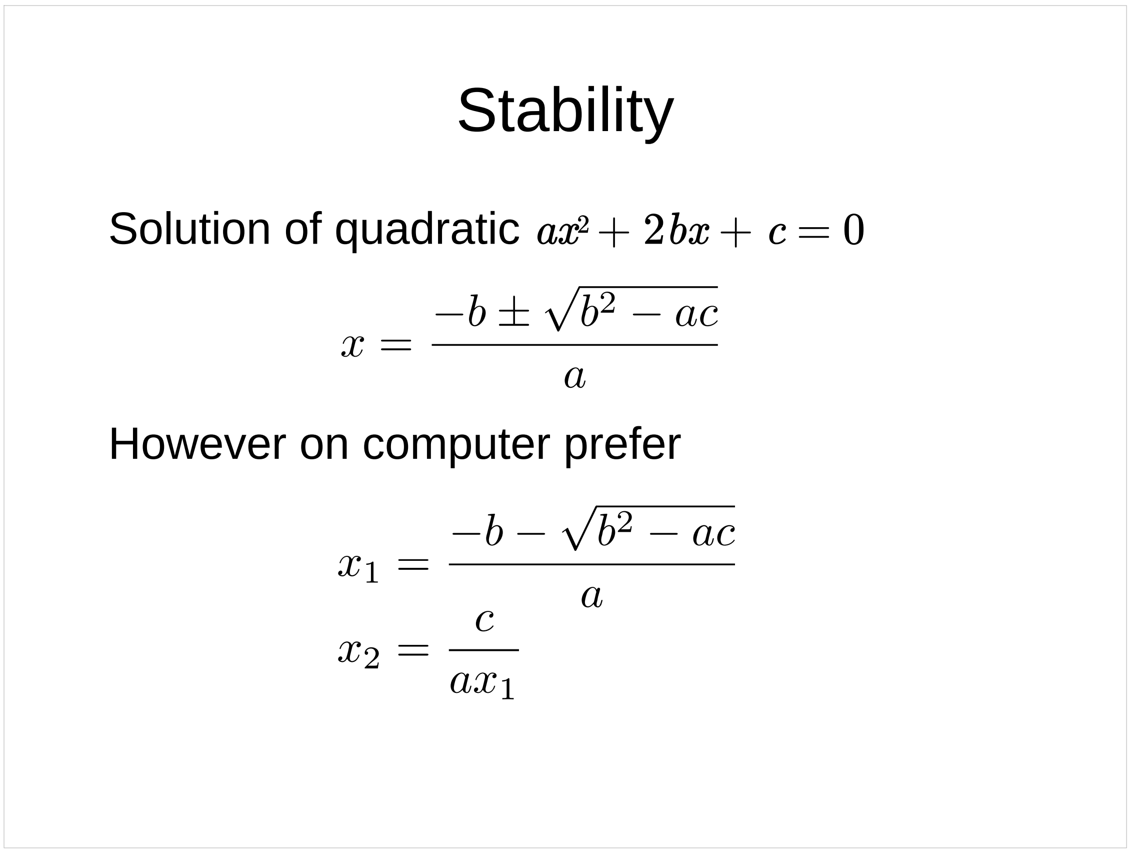
Only seven multiplys, with divide and conquer gives time $\Theta(n^{\log 7/\log 2})$ ~ $\Theta(n^{2.81})$

Best so far is $\Theta(n^{2.3727})$ - not practical

Open problem if $\Theta(n^2)$

Stability for floating point numbers?

There are some very plausible conjectures which if true imply order of n^2. There are some other quite plausible conjectures which imply the n^2 result can't be achieved!



Stability

Solution of quadratic $ax^2 + 2bx + c = 0$

$$x = \frac{-b \pm \sqrt{b^2 - ac}}{a}$$

However on computer prefer

$$x_1 = \frac{-b - \sqrt{b^2 - ac}}{a}$$

$$x_2 = \frac{c}{ax_1}$$

Show what stability is about. If ac is much smaller than b^2 then most significant figures disappear if the + sign is used in the normal formula.

Perhaps not this!

Digicomp II multiplying 3×13

C:\Users\David\Desktop\partition\digicomp II.webm

Have the break here

.

The video shows the Digicomp II doing a binary multiply of 3 by 13 to produce 39

Discrete Fourier Transform

The matrix multiply and its inverse in the Toom Cook system become much simpler at the *n*th roots of unity.

$$1, e^{\frac{2\pi i}{n}}, e^{\frac{4\pi i}{n}}, \ldots, e^{\frac{2(n-1)\pi i}{n}}$$

In 1965 Cooley and Turkey found a fast way to compute the values at these points

Fast Fourier Transform

From v_0, v_1, v_2, v_3

Calculate

$$v_0 + v_2, v_0 - v_2, v_1 + v_3, v_1 - v_3$$

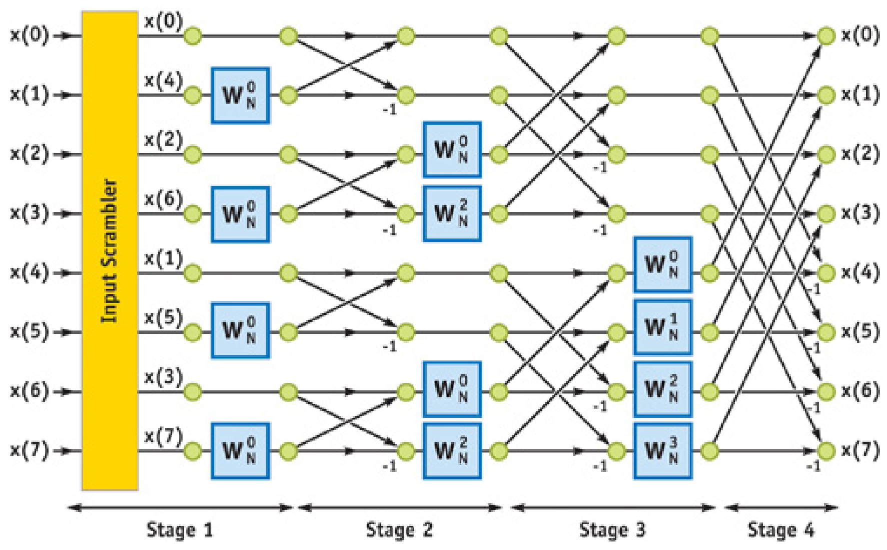
Then

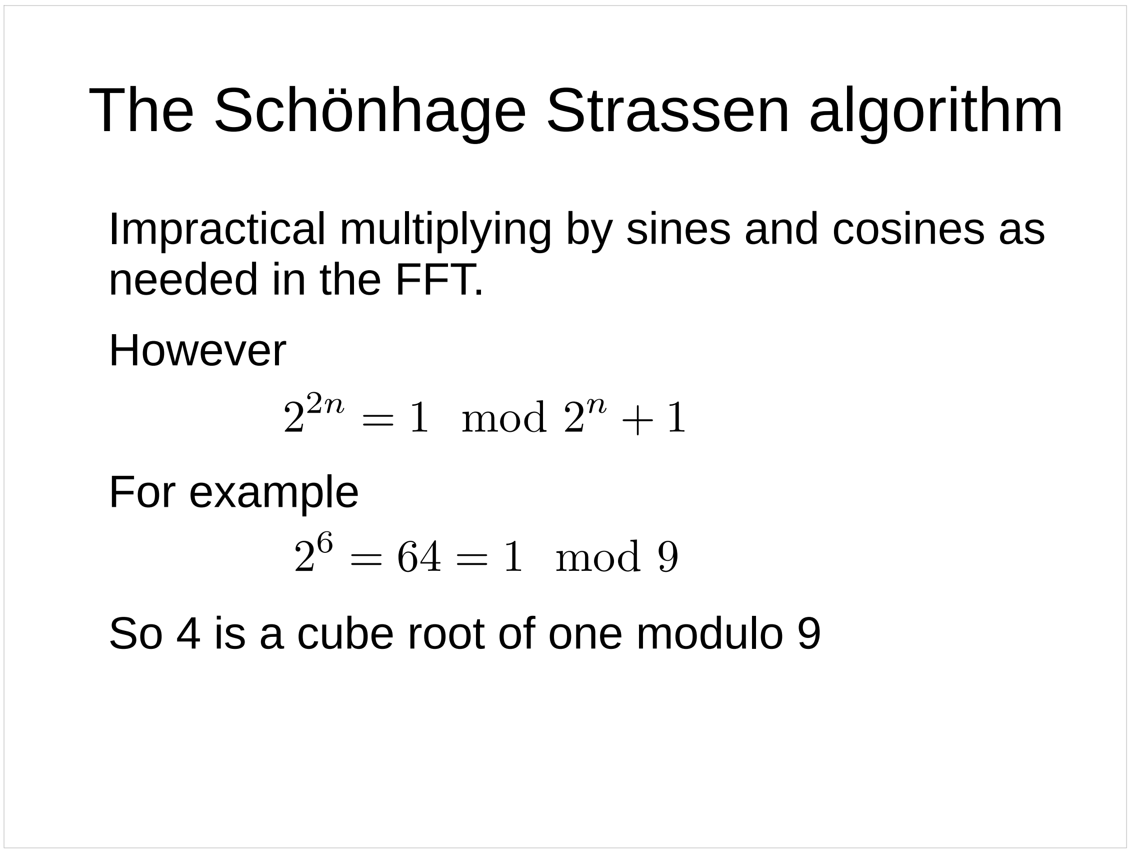
$$(v_0 + v_2) + (v_1 + v_3), (v_0 + v_2) - (v_1 + v_3),$$
$$(v_0 - v_2) + i(v_1 - v_3), (v_0 - v_2) - i(v_1 - v_3)$$

Total operations $4 \log_2 4$

$$\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & i & i^2 = -1 & i^3 = -i \\
1 & i^2 = -1 & i^4 = 1 & i^6 = -1 \\
1 & i^3 = -i & i^6 = -1 & i^9 = i
\end{array}$$

A Butterfly Diagram





The Schönhage Strassen algorithm

Impractical multiplying by sines and cosines as needed in the FFT.

However

$$2^{2n} = 1 \mod 2^n + 1$$

For example

$$2^6 = 64 = 1 \mod 9$$

So 4 is a cube root of one modulo 9

Getting back to modulo multiplication

Plus a couple of tricks

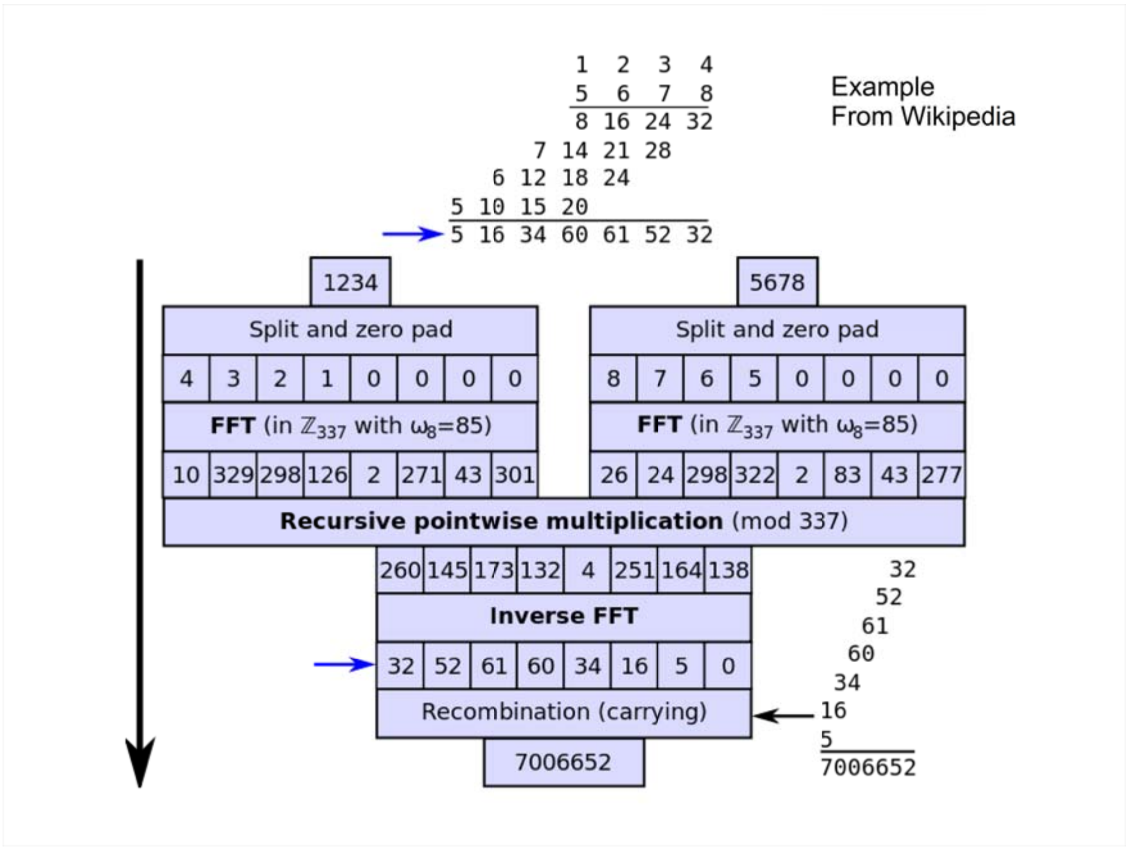
The number should be split into sections of square root of number of bits long, then recurse on the smaller bits. Leads to time

$$\Theta(n \cdot \log n \cdot \log \log n)$$

The smaller parts don't need to be doubled in length since only required modulo a number.

Multiplying or modulo 2^k+1 just requires shift+add/subtract

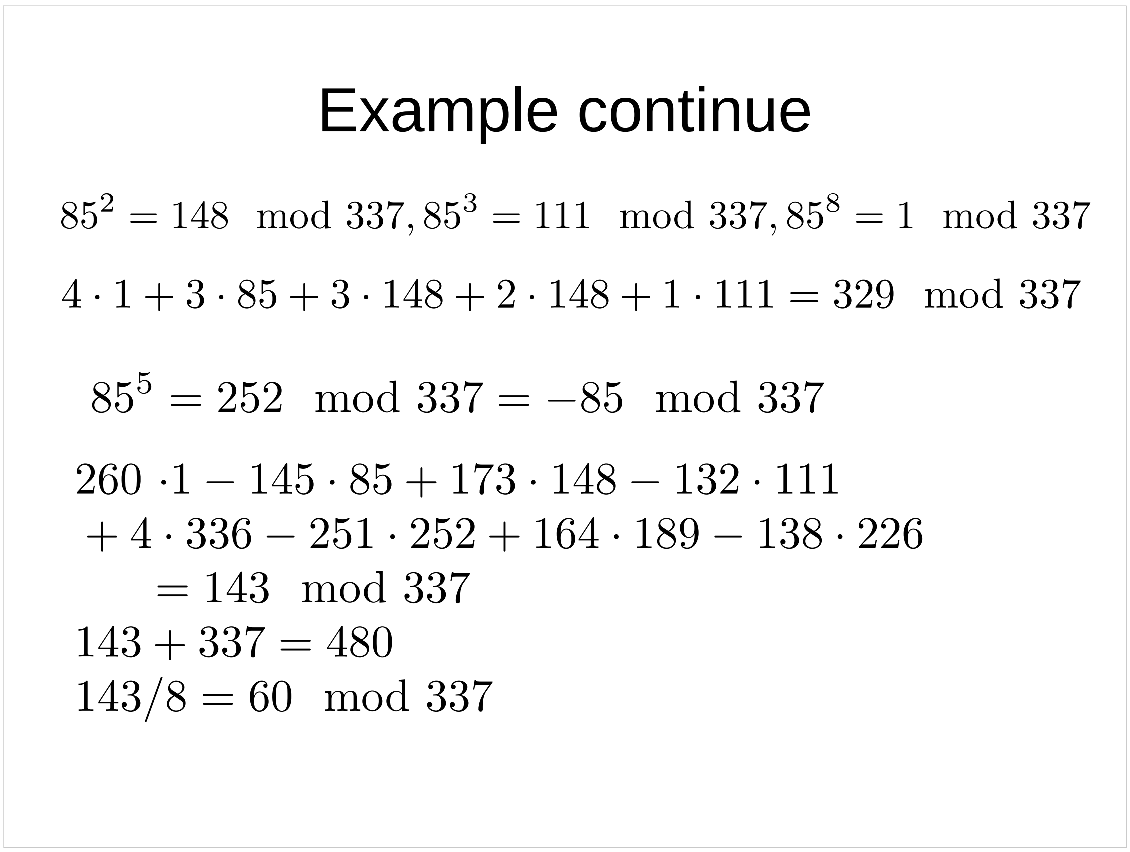
- An area that real effort has been spent on.



Doing a decimal version here rather than the binary one done in computers

337 is a big enough number so the partial products won't overflow and causes aliases – remember the 5x6x7 210 example where numbers less than that exact.

Max partial product and sum is 4 lots of 9 times 9 which is 324, 337 is the next number up with an integer eight root. 337 s a prime and 336 = 8x42 so has an eight root. In fact 85 is an eight root modulo 337.



Example continue

$$85^2 = 148 \mod 337, 85^3 = 111 \mod 337, 85^8 = 1 \mod 337$$

$$4 \cdot 1 + 3 \cdot 85 + 3 \cdot 148 + 2 \cdot 148 + 1 \cdot 111 = 329 \mod 337$$

$$85^5 = 252 \mod 337 = -85 \mod 337$$

$$\begin{aligned} &260 \cdot 1 - 145 \cdot 85 + 173 \cdot 148 - 132 \cdot 111 \\ &+ 4 \cdot 336 - 251 \cdot 252 + 164 \cdot 189 - 138 \cdot 226 \\ &\quad = 143 \mod 337 \\ &143 + 337 = 480 \\ &143/8 = 60 \mod 337 \end{aligned}$$

Calculation of one of the entres of the fourier transform. The entries are multiplied together in pairs mod 337. Then calculation of one of the inverse fourier transform entries.
Need to divide by 8 in the inverse transform (in general by N for an N entry one)

M_{67} not a prime

“No factor was found until a famous talk by Cole in 1903. Without speaking a word, he went to a blackboard and raised 2 to the 67th power, then subtracted one. On the other side of the board, he multiplied 193,707,721 × 761,838,257,287 and got the same number, then returned to his seat (to applause) without speaking.”

So how does it all work out?

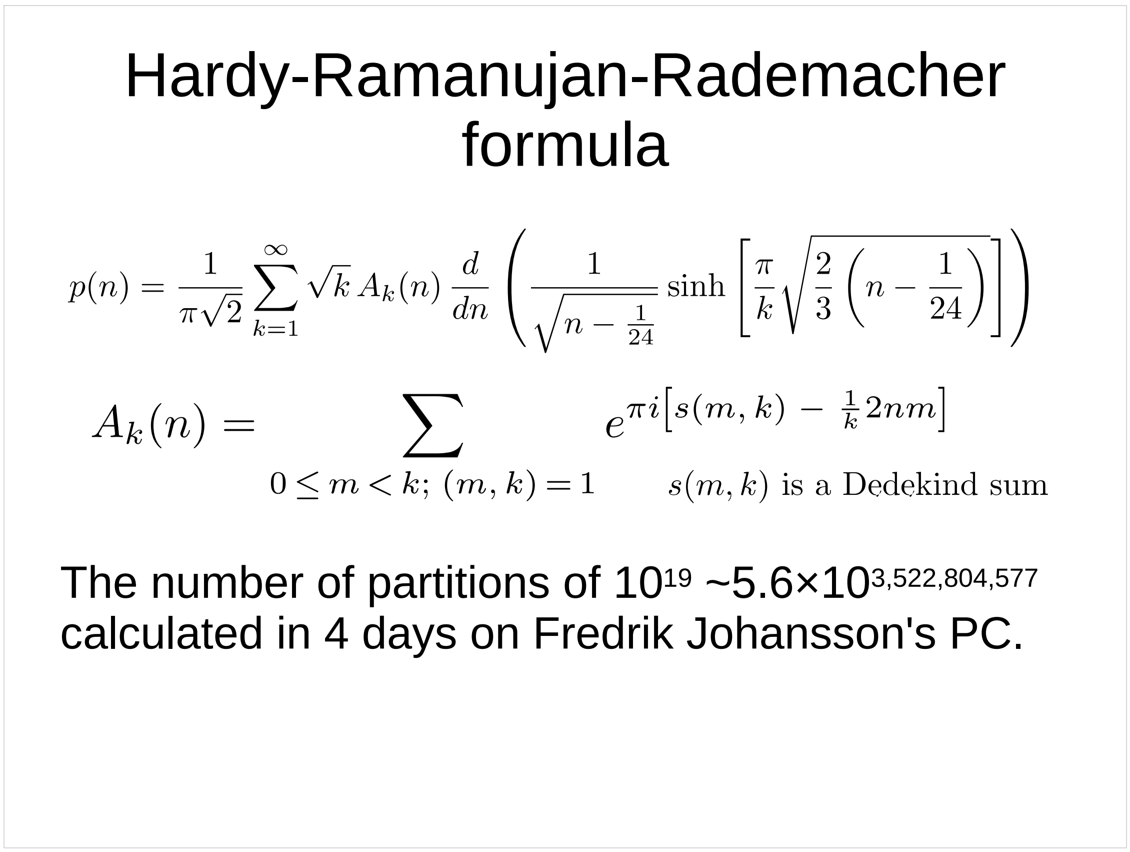
Partitions of a number: 7 for 5

5, 4 + l , 3 + 2 , 3 + l + l , 2 + 2 + 1,
2 + l + l , l + l + 1+ 1+ 1

People have been interested in large values because of extensions of a strange relation the first 3 of which were found by Ramanujan

$$\begin{aligned} p(5k+4) &= 0 \mod 5 \\ p(7k+5) &= 0 \mod 7 \\ p(11k+6) &= 0 \mod 11 \\ p(11^2 \cdot 13 \cdot k + 237) &= 0 \mod 13 \end{aligned}$$

$s(m,k)$ is a Dedekind sum



Hardy-Ramanujan-Rademacher formula

$$p(n) = \frac{1}{\pi\sqrt{2}} \sum_{k=1}^{\infty} \sqrt{k}\, A_k(n)\, \frac{d}{dn} \left(\frac{1}{\sqrt{n-\frac{1}{24}}} \sinh\left[\frac{\pi}{k} \sqrt{\frac{2}{3}\left(n - \frac{1}{24}\right)} \right] \right)$$

$$A_k(n) = \sum_{0 \le m < k;\ (m,k)=1} e^{\pi i \left[s(m,k) \ - \ \frac{1}{k} 2nm \right]}$$

$s(m,k)$ is a Dedekind sum

The number of partitions of 10^{19} ~$5.6 \times 10^{3,522,804,577}$ calculated in 4 days on Fredrik Johansson's PC.

A formula just to cause a Gah! Reaction. Evaluated till sure the result will converge to a particular number. Here about 500 million terms were used.

5646928403 . . .3674631046 but we'll leave out the middle 3 thousand million digits!

A single million digit by million digit multiplication can be done in a hundreth of a second.

Have found numerous other congruences like the Rmanaujan ones – 'Experimental maths'.

Future

Arithmetic on two numbers on a single machine ✔

- Efficiently use parallel computers
- A faster way to multiply matrices?
- Even more use of modular arithmetic and Chinese Remainder Theorem
- Speed up series evaluation - get better trig functions for example