Design and Analysis of Algorithms

Lecture 1:

Introduction, and multiplication!

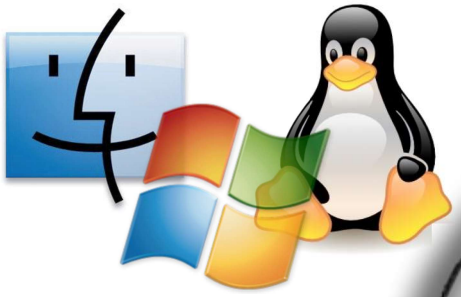
Why are you here?

- It is required.

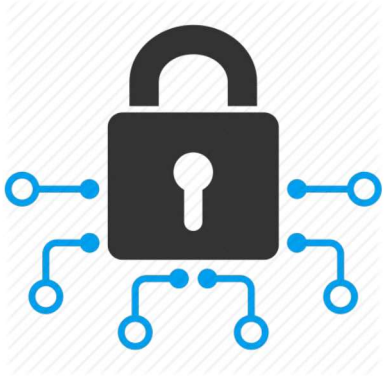
Why is this course required?

- Algorithms are fundamental to CS.
- Algorithms are useful.
- Algorithms are fun!

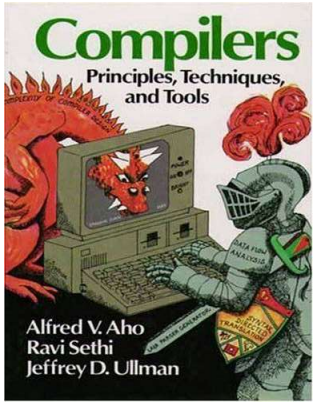
Algorithms are fundamental



Operating Systems (CS 1



Cryptography (CS 255)



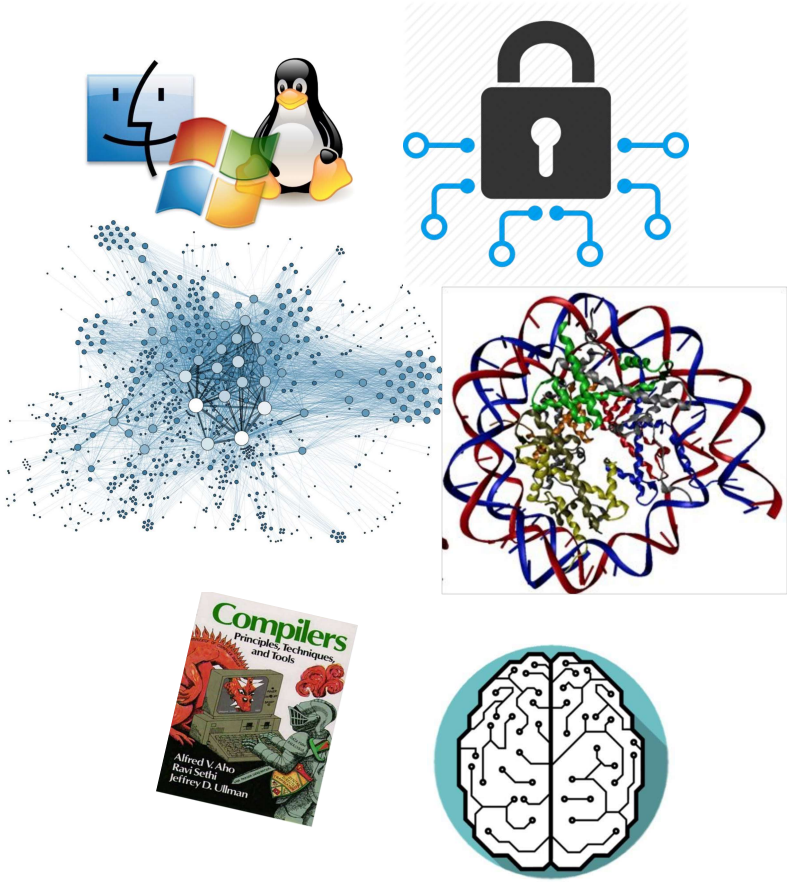
Compilers (CS 143)

Networking (CS 144)

Computational Biology (CS 262)

Algorithms are useful

- All those things
- As we get more and more data and problem sizes get bigger and bigger, algorithms become more and more important.
- Will help you get a job.



Algorithms are fun!

- Algorithm design is both an art and a science.
- Many surprises!
- A young field, lots of exciting research questions!

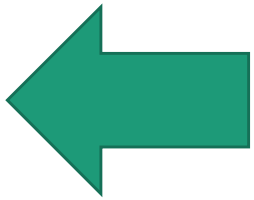
- (Will help you get a job you like!)

Course goals

- Build an "algorithmic toolkit"
- Learn to think "algorithmically"

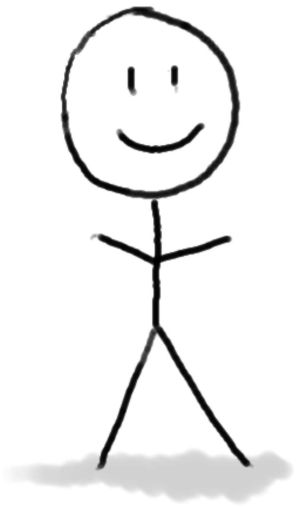
Today's goals

- Karatsuba Integer Multiplication
- Technique: Divide and conquer
- Meta points:
  - Algorithm designer's question
  - The role of rigor



The algorithm designer's question

Can I do better?



Algorithm designer

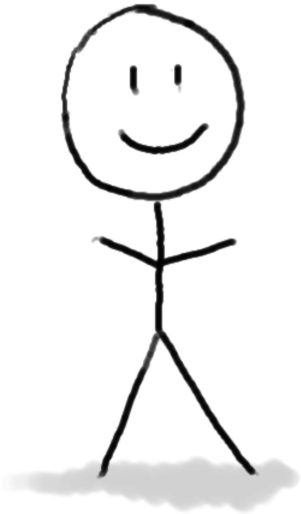
The algorithm designer's internal monologue…

What exactly do we mean by better? And what about that corner case? Shouldn't we be zero-indexing?

Plucky the Pedantic Penguin

Detail-oriented
Precise
Rigorous

Can I do better?



Algorithm designer

Dude, this is just like that other time. If you do the thing and the stuff like you did then, it'll totally work real fast!

Lucky the Lackadaisical Lemur

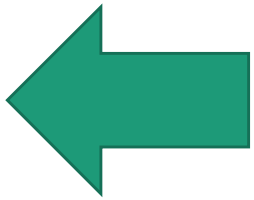
Big-picture
Intuitive
Hand-wavey

Course goals

- Build an "algorithmic toolkit"
- Learn to think "algorithmically"

Today's goals

- Karatsuba Integer Multiplication
- Technique: Divide and conquer
- Meta points:
  - Algorithm designer's question
  - The role of rigor



A problem you all know how to solve:

Integer Multiplication

$$\begin{array}{r} 12 \\ \times\ 34 \\ \hline \end{array}$$

A problem you all know how to solve:

Integer Multiplication

$$
\begin{array}{rr}
 & 12345678959314 13 \\
\times & 4563823520395533 \\
\hline
\end{array}
$$

A problem you all know how to solve:

Integer Multiplication

n

```
  12339257207527523846237642835683649183745238562 98
x 45623235823423952856234672350191307501353500137 53
```

???

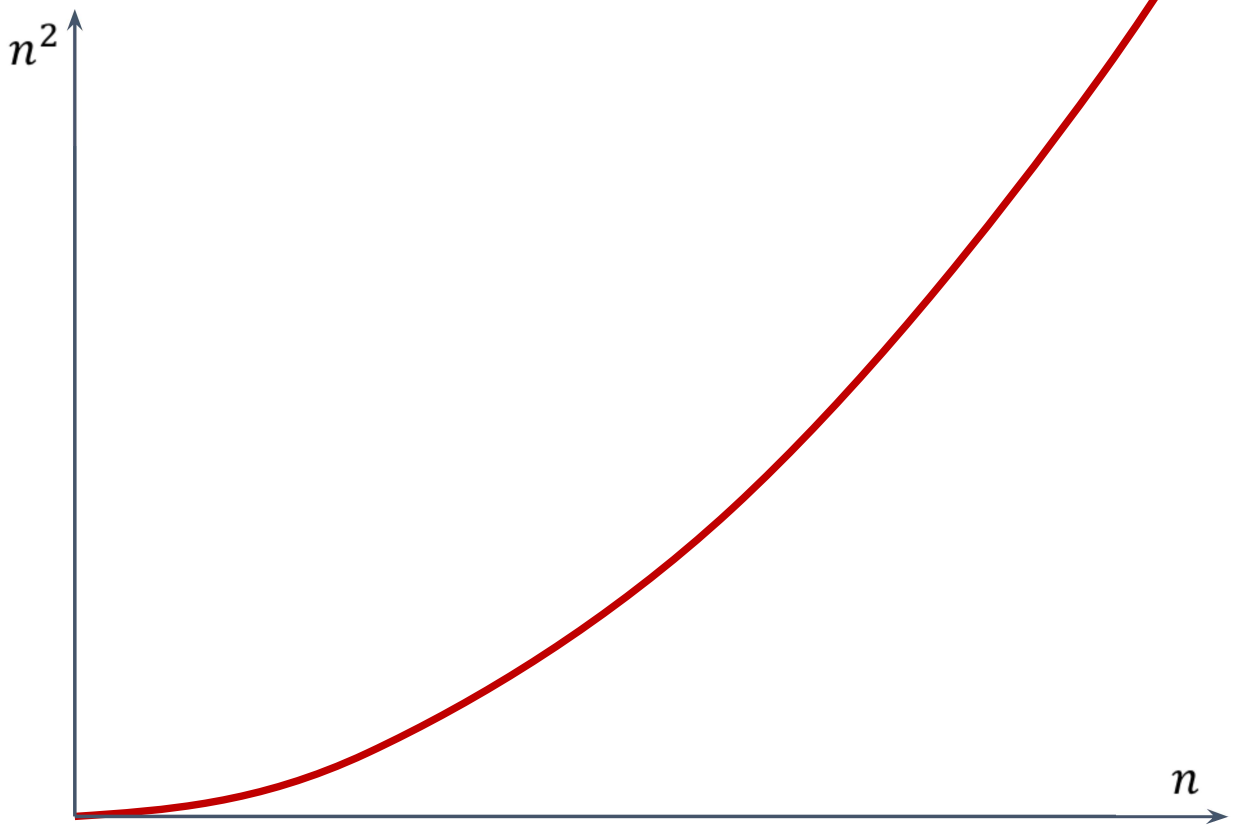
How would you solve this problem?
How long would it take you?

About n^2 one-digit operations

At most n^2 multiplications,
and then at most n^2 additions (for carries)
and then I have to add n different 2n-digit numbers…

And I take 1 second to multiply two one-digit numbers and .6 seconds to add, so…

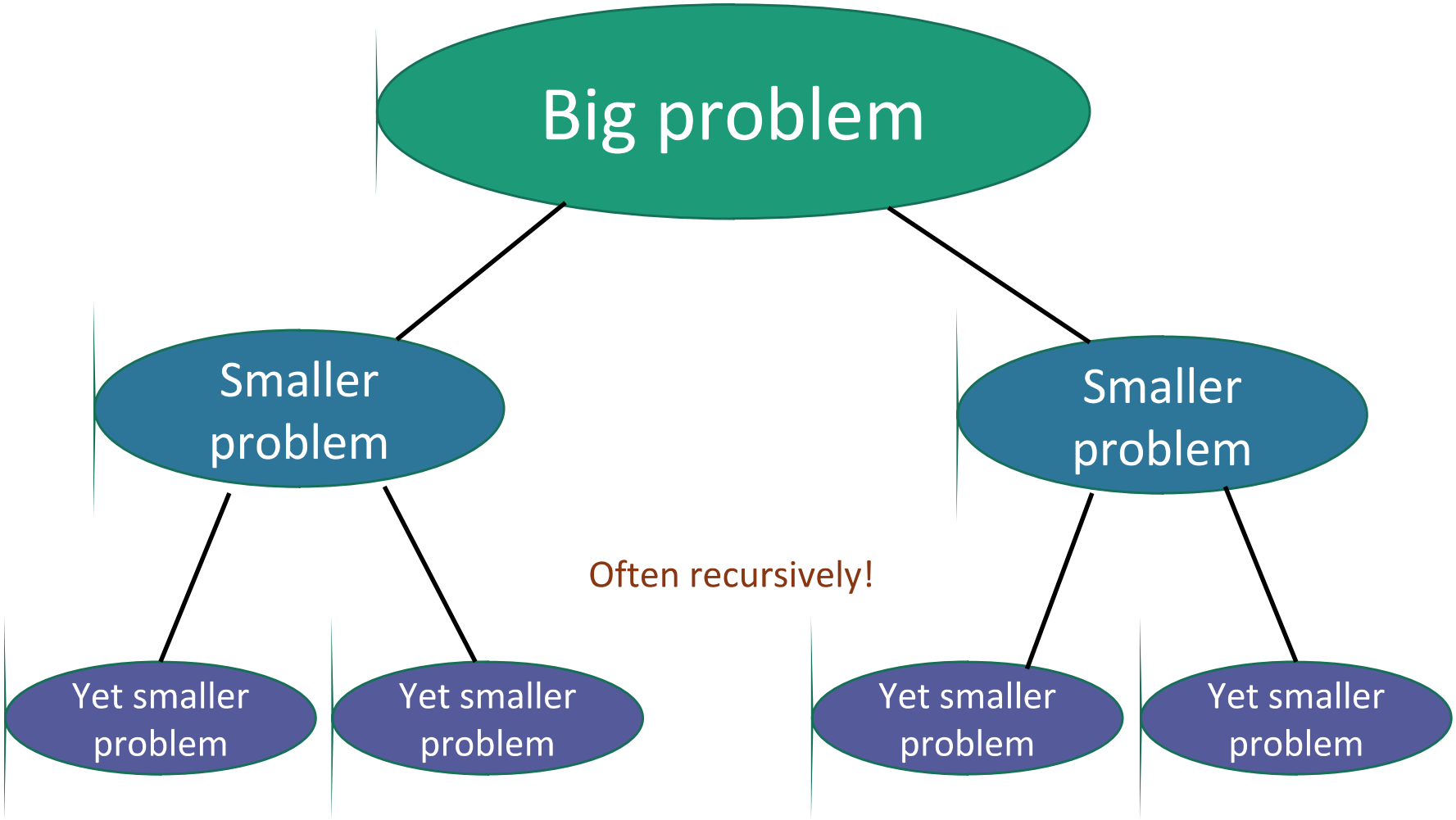
Can we do better?



Let’s dig in to our algorithmic toolkit…

Divide and conquer

Break problem up into smaller (easier) sub-problems

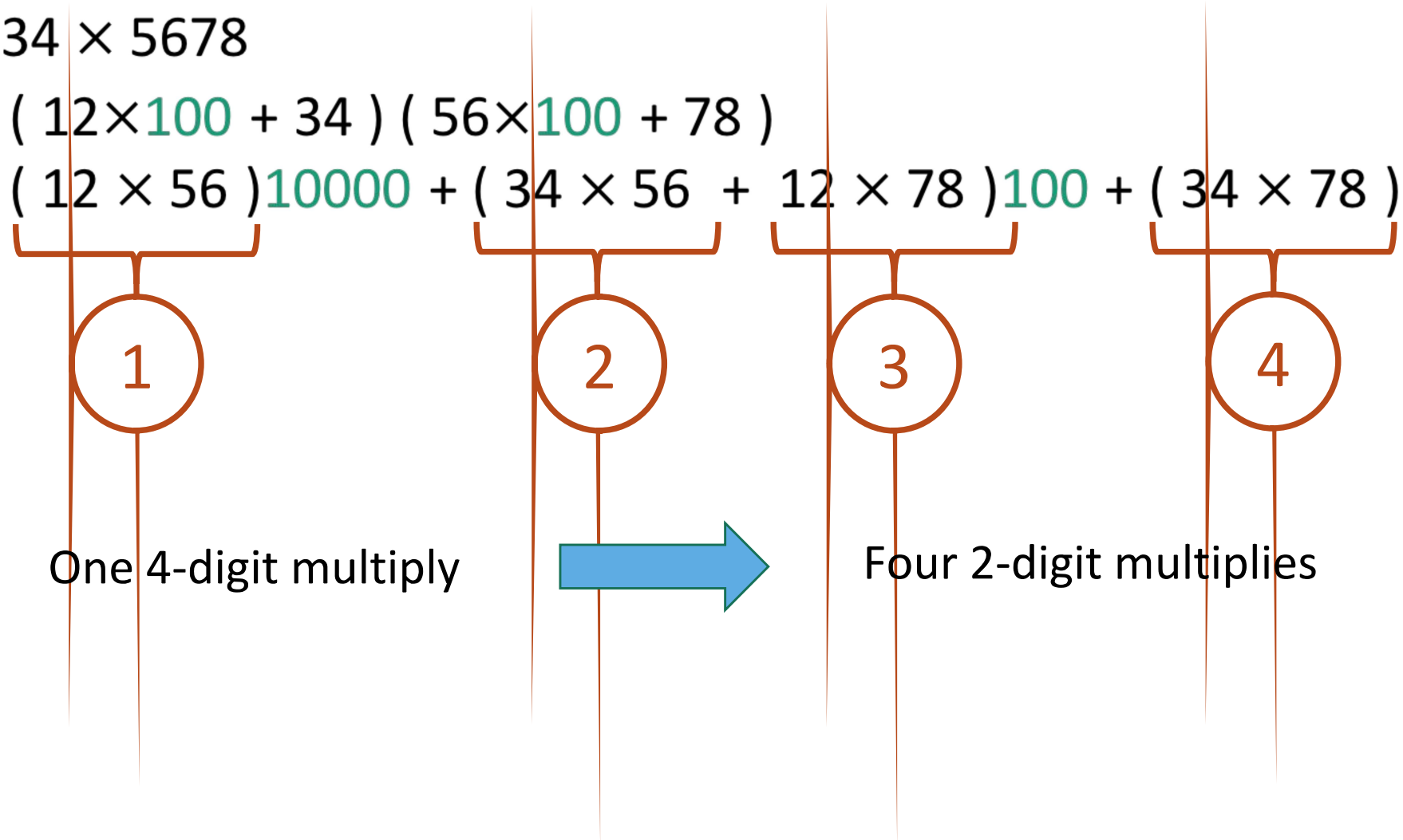


Divide and conquer for multiplication

Break up an integer:

$$1234 = 12 \times 100 + 34$$

$$1234 \times 5678$$
$$= (12 \times 100 + 34)(56 \times 100 + 78)$$
$$= (12 \times 56)10000 + (34 \times 56 + 12 \times 78)100 + (34 \times 78)$$



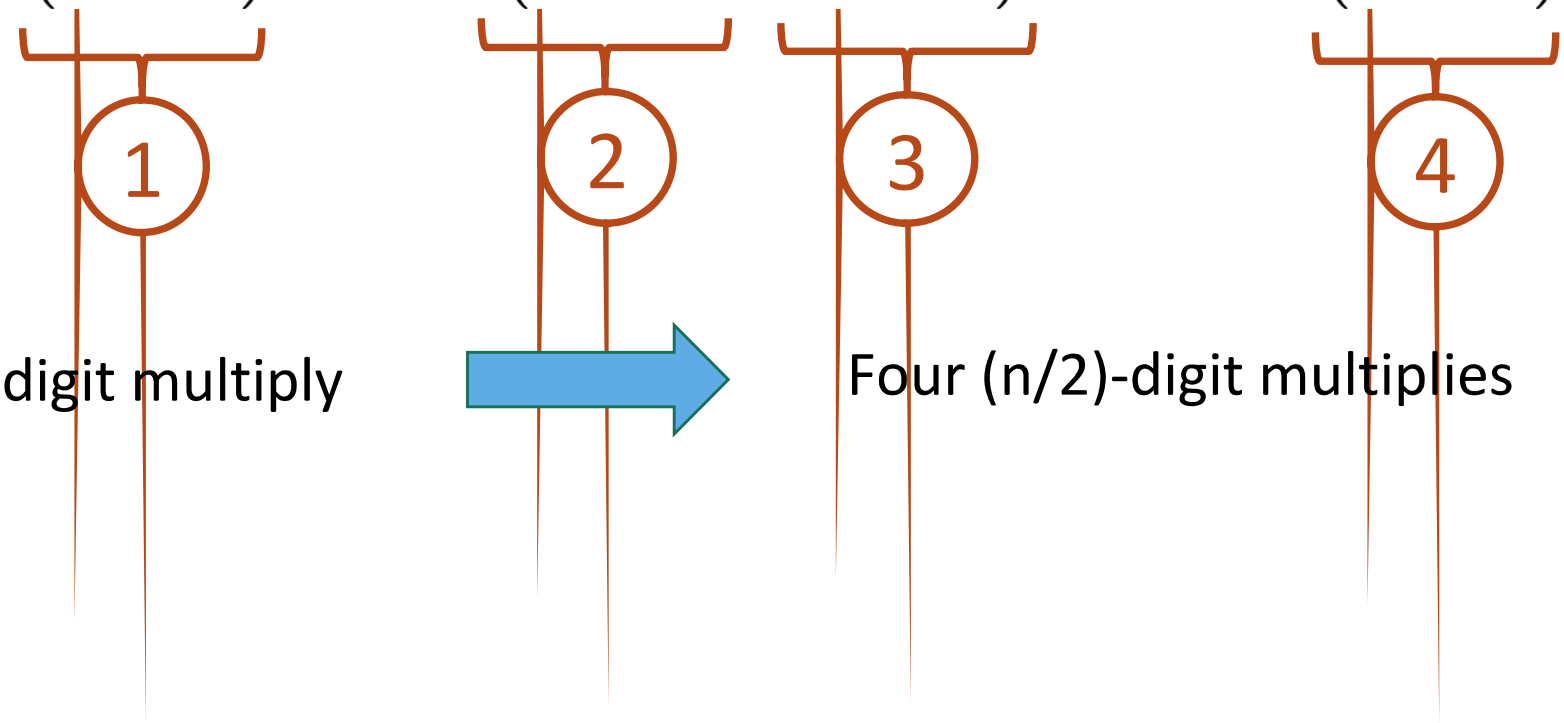
More generally

Break up an n-digit integer:

$$[x_1 x_2 \cdots x_n] = [x_1 x_2 \cdots x_{n/2}] \times 10^{n/2} + [x_{n/2+1} x_{n/2+2} \cdots x_n]$$

$$x \times y = (a \times 10^{n/2} + b)(c \times 10^{n/2} + d)$$

$$= (a \times c)10^n + (a \times d + c \times b)10^{n/2} + (b \times d)$$



One n-digit multiply

Four (n/2)-digit multiplies

Divide and conquer algorithm

x,y are n-digit numbers

Base case: I've memorized my 1-digit multiplication tables…

Multiply(x, y):

- Write $x = a\ 10^{\frac{n}{2}} + b$
- Write $y = c\ 10^{\frac{n}{2}} + d$

a, b, c, d are n/2-digit numbers

- Recursively compute ac, ad, bc, bd:
  - ac = **Multiply**(a, c), etc…
- Add them up (with shifts) to get xy

Not very precise pseudocode…

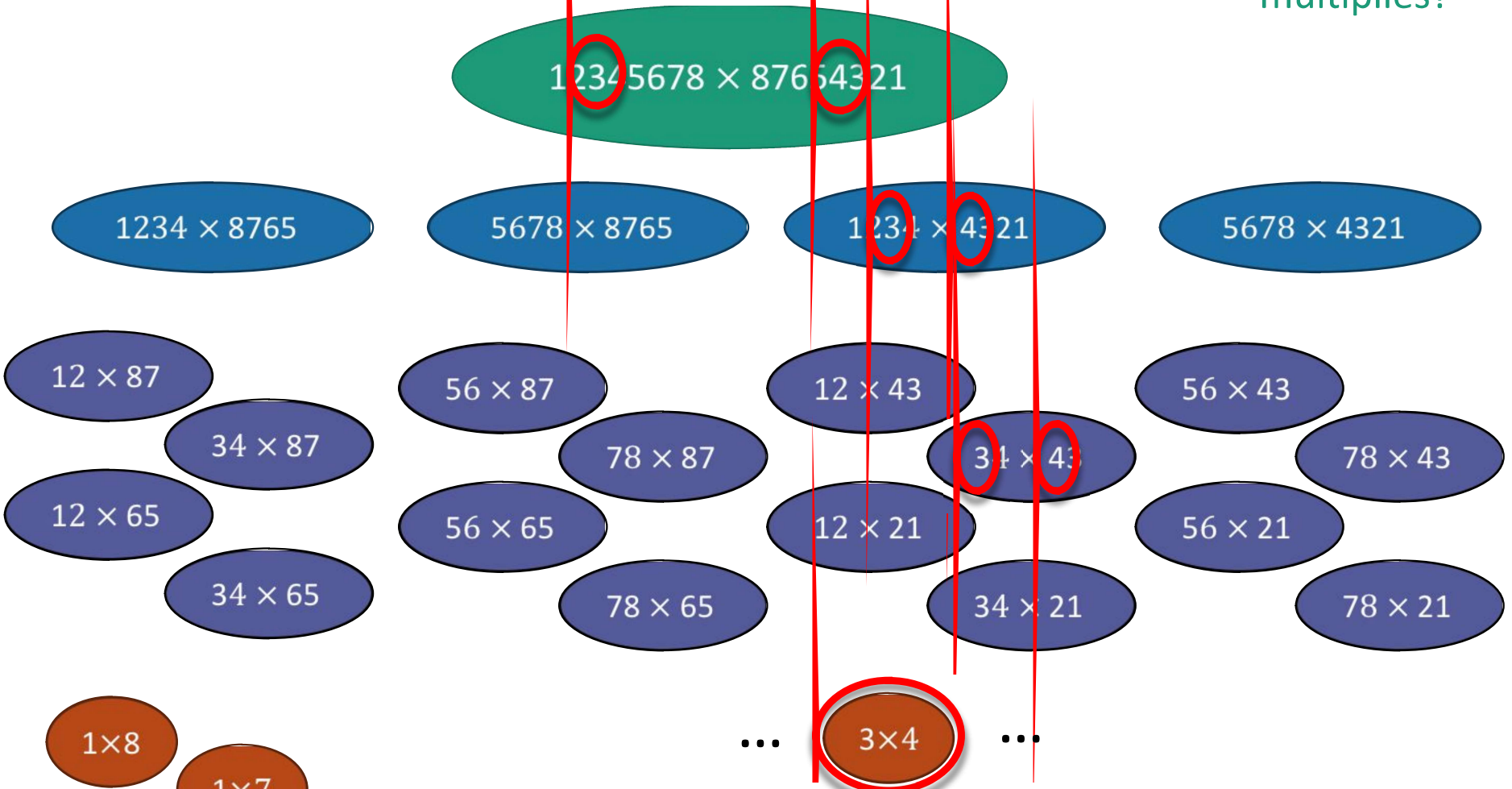
Questions about the algorithm?

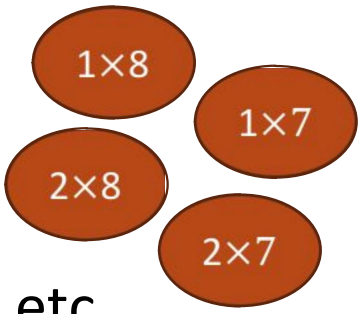
How long does this take?

Better or worse than the grade-school algorithm?

Let's do an example

How many one-digit multiplies?





etc…

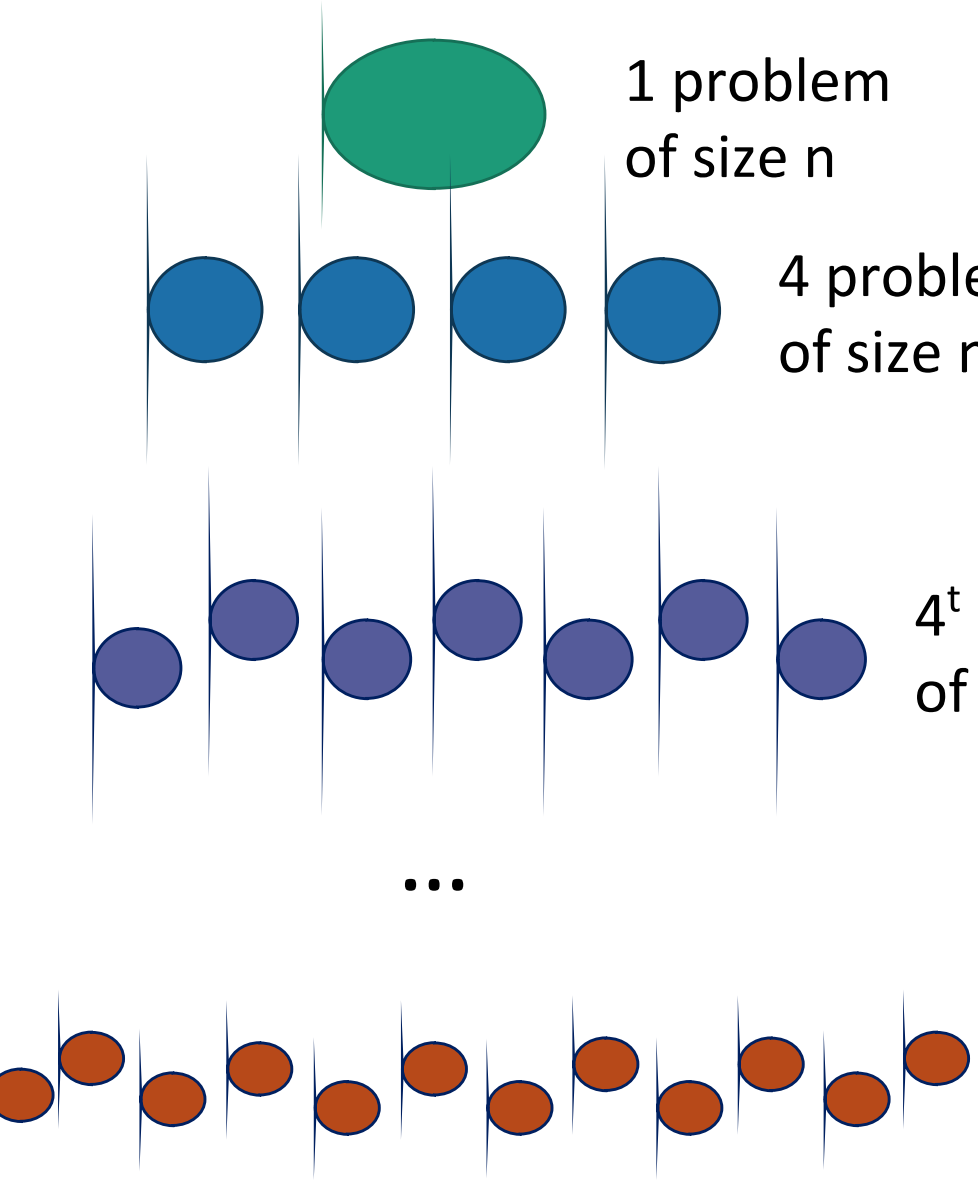
Claim: there are n^2 one-digit problems.

Every pair of digits still gets multiplied together separately.

So the running time is still at least n^2.

Another way to see this*

*we will come back to this sort of analysis later and still more rigorously.



1 problem of size n

4 problems of size n/2

4^t problems of size $n/2^t$

...

$\frac{n^2}{\_\_\_}$ problems of size 1

- If you cut n in half $\log_2(n)$ times, you get down to 1.
- So we do this $\log_2(n)$ times and get…

$4^{\log\_2(n)} = n^2$ problems of size 1.

What about the work you actually do in the problems?

Yet another way to see this*

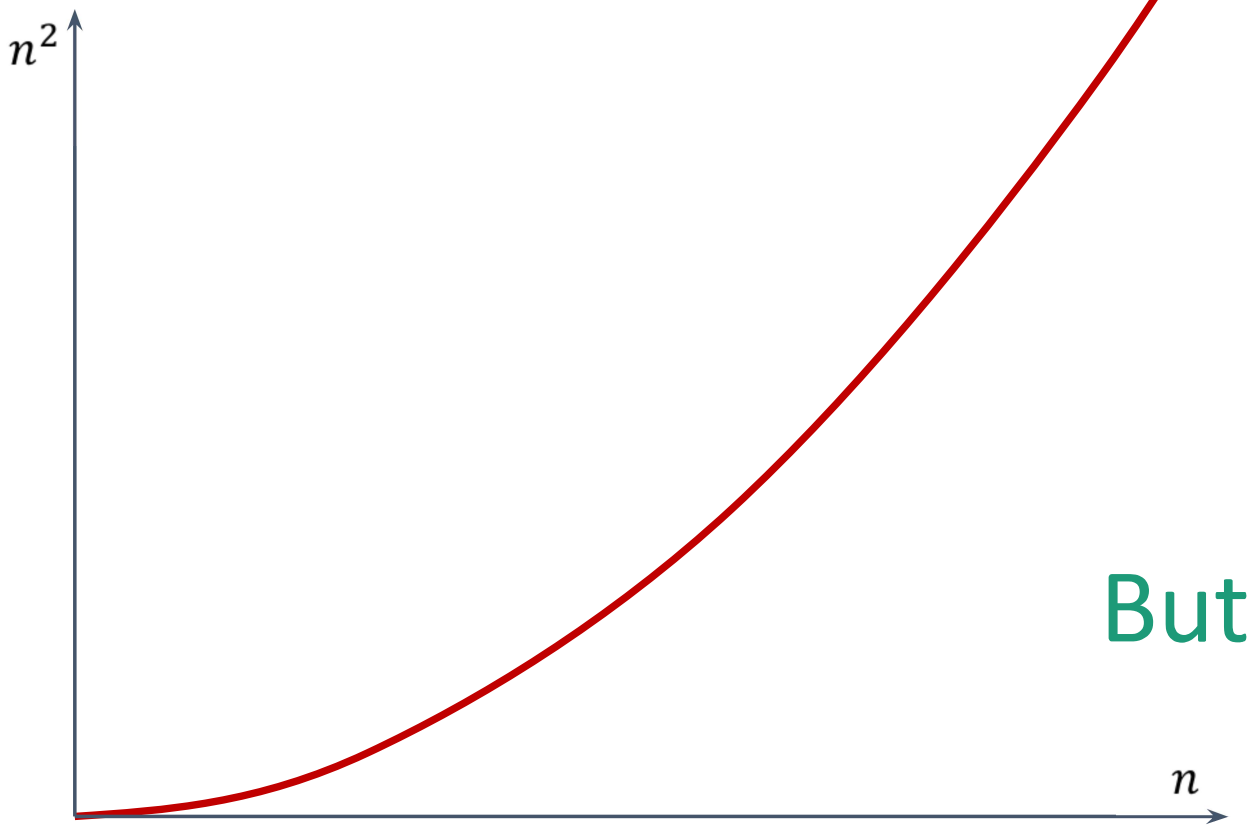
- Let T(n) be the time to multiply two n-digit numbers.
- Recurrence relation:
  - $T(n) = 4 \cdot T\left(\frac{n}{2}\right) +$ (about n to add stuff up)

Ignore this term for now…

$$
\begin{aligned}
T(n) &= 4 \cdot T(n/2) \\
&= 4 \cdot (4 \cdot T(n/4)) && \text{---------- } 4^2 \cdot T(n/2^2) \\
&= 4 \cdot (4 \cdot (4 \cdot T(n/8))) && \text{---------- } 4^3 \cdot T(n/2^3) \\
&\vdots \\
&= 2^{2t} \cdot T(n/2^t) && \text{---------- } 4^t \cdot T(n/2^t) \\
&\vdots \\
&= n^2 \cdot T(1). && \text{---------- } 4^{\log_2(n)} \cdot T(n/2^{\log_2(n)})
\end{aligned}
$$

That’s a bit dissappointing

All that work and still n^2…

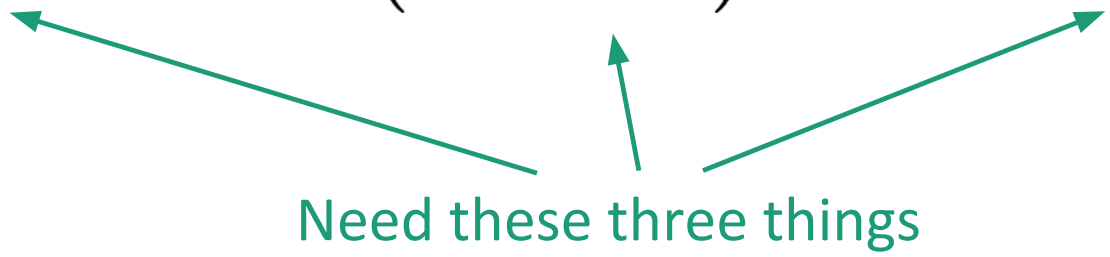


But wait!!

Divide and conquer can actually make progress

- Karatsuba figured out how to do this better!

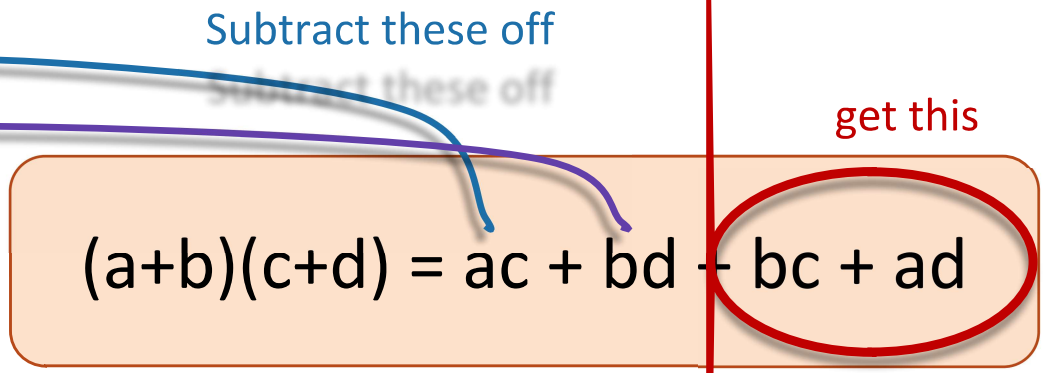
$$xy = (a \cdot 10^{n/2} + b)(c \cdot 10^{n/2} + d)$$
$$= ac \cdot 10^{n} + (ad + bc)10^{n/2} + bd$$



- If only we recurse three times instead of four…

Karatsuba integer multiplication

- Recursively compute
  - ac
  - bd
  - (a+b)(c+d)

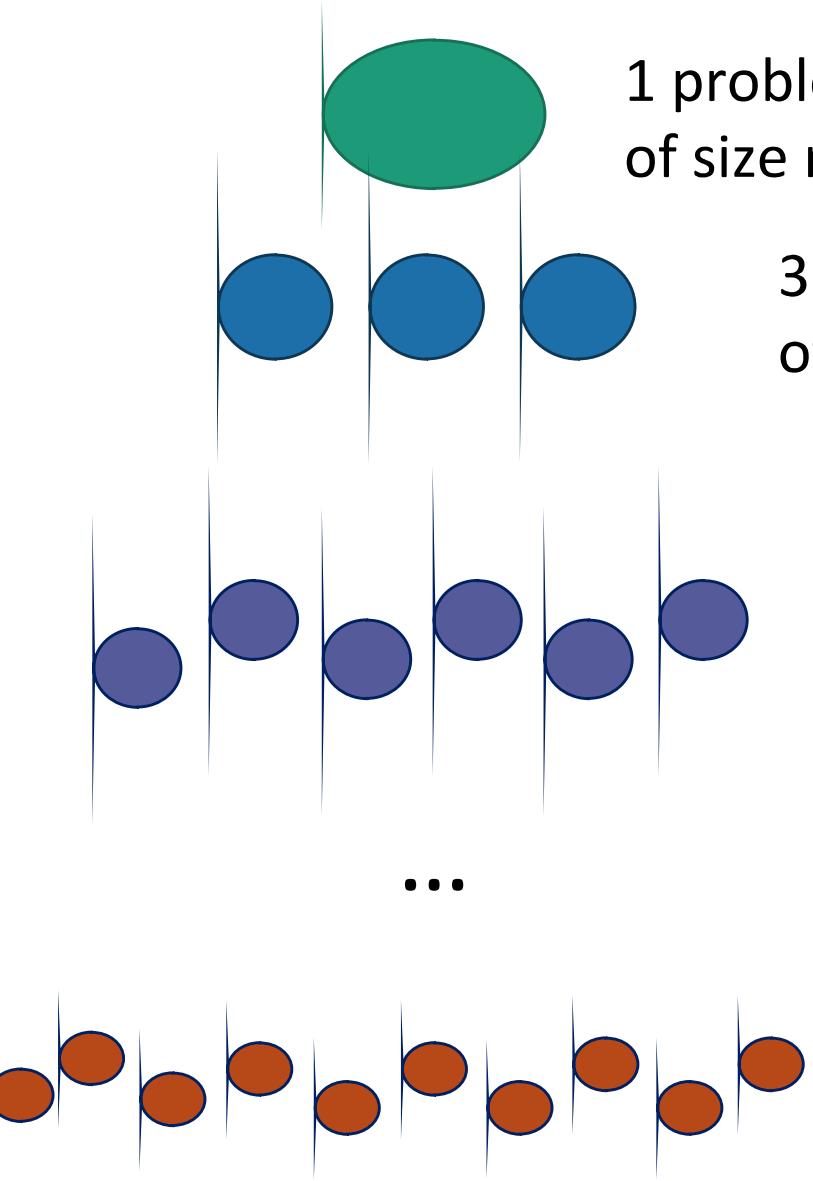


- Assemble the product:

$$xy = (a \cdot 10^{n/2} + b)(c \cdot 10^{n/2} + d)$$
$$= ac \cdot 10^{n} + (ad + bc)10^{n/2} + bd$$

✓ ✓ ✓

What's the running time?



1 problem
of size n

3 problems
of size n/2

3^t problems
of size $n/2^t$

...

$\frac{n^{1.6}}{}$ problems
of size 1

- If you cut n in half $\log_2(n)$ times, you get down to 1.
- So we do this $\log_2(n)$ times and get…

$3^{\log\_2(n)} = n^{\log\_2(3)} = n^{1.6}$
problems of size 1.

This is much better!

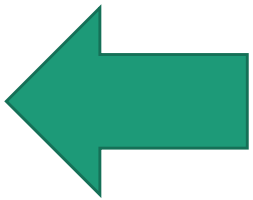
(Can we do better still?)

Course goals

- Build an "algorithmic toolkit"
- Learn to think "algorithmically"

Today's goals

- Karatsuba Integer Multiplication
- Technique: Divide and conquer
- Meta points:
  - Algorithm designer's question
  - The role of rigor
- End on a historical note…



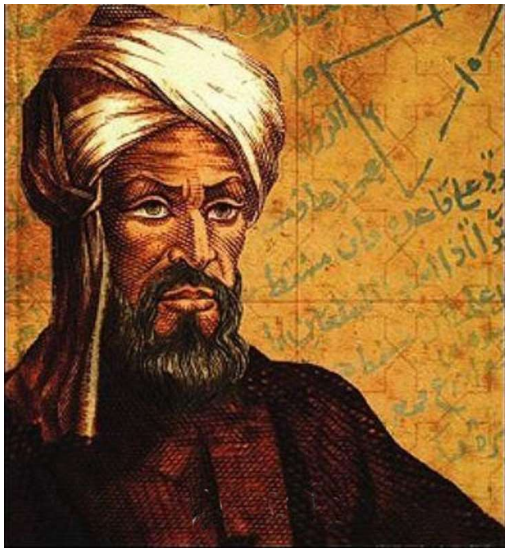
It’s actually pretty amazing that you can big multiply numbers quickly at all

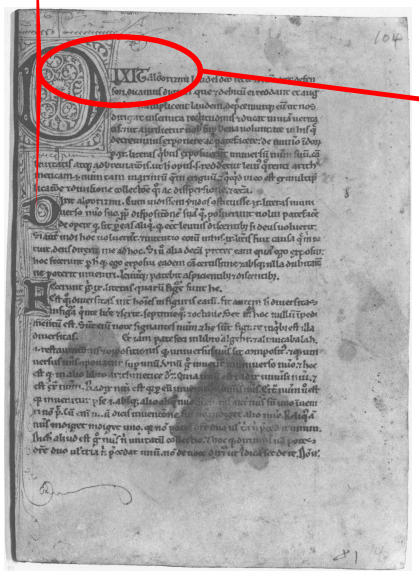
- You could do this when you were 8.
- It wasn’t always so easy!

LXXXIX × CM = ?

Etymology of “Algorithm”

- Al-Khwarizmi (Persian mathematician, lived around 800AD) wrote a book about how to multiply with Arabic numerals.
- His ideas came to Europe in the 12th century.





Dixit algorizmi
(so says Al-Khwarizmi)

- Originally, “Algorisme” [old French] referred to just the Arabic number system, but eventually it came to mean “Algorithm” as we know today.

Wrap up

- Algorithms are:
  - Fundamental, useful, and fun!
- In this course, we will develop both algorithmic intuition and algorithmic technical chops
- Karatsuba Integer Multiplication:
  - You can do better than grade school multiplication!
  - Example of divide-and-conquer in action

Next time

- Divide-and-conquer again
- Asymptotics and big-O notation