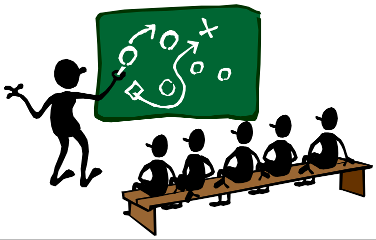
Great Theoretical Ideas In Computer Science

Victor Adamchik
Danny Sleator

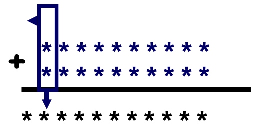
CS 15-251 Spring 2010

Lecture 15 Mar 02, 2010 Carnegie Mellon University

## Grade School Revisited: How To Multiply Two Numbers



## Time complexity of grade school addition

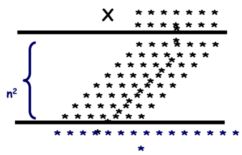


**T(n) = amount of time grade school addition uses to add two n-bit numbers**

T(n) is linear:

**T(n) = Θ(n)**

## Time complexity of grade school multiplication

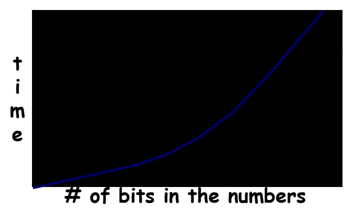



**T(n) = The amount of time grade school multiplication uses to multiply two n-bit numbers**

T(n) is quadratic:

**$T(n) = \Theta(n^2)$**

**Grade School Addition: Linear time**
**Grade School Multiplication: Quadratic time**




**No matter how dramatic the difference in the constants, the quadratic curve will eventually dominate the linear curve**

## Is there a sub-linear time method for addition?

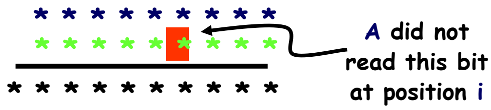
## Any addition algorithm takes Ω(n)

**Claim: Any algorithm for addition must read all of the input bits**

**Proof: Suppose there is a mystery algorithm A that does not examine each bit**

**Give A a pair of numbers. There must be some unexamined bit at position i in one of the numbers**

## Any addition algorithm takes $\Omega(n)$




If A is not correct on the inputs, we found a bug

If A is correct, flip the bit at position i
A gives the same answer as before, which is now wrong.

---

## Grade school addition can't be improved upon by more than a constant factor

---

Grade School Addition: $\Theta(n)$ time.
Furthermore, it is optimal

Grade School Multiplication: $\Theta(n^2)$ time

Is there a clever algorithm to multiply two numbers in linear time?

Despite years of research, no one knows! If you resolve this question, Carnegie Mellon will give you a PhD!

---

Can we even break the quadratic time barrier?

In other words, can we do something very different than grade school multiplication?

---

## Divide And Conquer

An approach to faster algorithms:

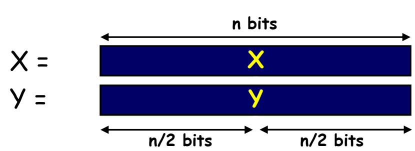
DIVIDE a problem into smaller subproblems

CONQUER them recursively

GLUE the answers together so as to obtain the answer to the larger problem

---

## Multiplication of 2 n-bit numbers




$X = a\ 2^{n/2} + b \qquad Y = c\ 2^{n/2} + d$

$X \times Y = ac\ 2^n + (ad + bc)\ 2^{n/2} + bd$

## Multiplication of 2 n-bit numbers

X = | a | b |
Y = | c | d |
n/2 bits   n/2 bits

$X \times Y = ac\ 2^n + (ad + bc)\ 2^{n/2} + bd$

MULT(X,Y):
If |X| = |Y| = 1 then return XY
else break X into a;b and Y into c;d
return MULT(a,c) $2^n$ + (MULT(a,d)
+ MULT(b,c)) $2^{n/2}$ + MULT(b,d)

## Same thing for numbers in decimal

n digits

X = | a | b |
Y = | c | d |
n/2 digits   n/2 digits

$X = a\ 10^{n/2} + b \qquad Y = c\ 10^{n/2} + d$

$X \times Y = ac\ 10^n + (ad + bc)\ 10^{n/2} + bd$

## Multiplying (Divide & Conquer style)

12345678 * 21394276

1234*2139 1234*4276 5678*2139 5678*4276

12*21 12*39 34*21 34*39

1*2 1*1 2*2 2*1

2 1 4 2

Hence: 12*21 = $2*10^2 + (1 + 4)10^1 + 2 = 252$

| a | b |
| c | d |

$X \times Y = ac\ 10^n + (ad + bc)\ 10^{n/2} + bd$

## Multiplying (Divide & Conquer style)

12345678 * 21394276

1234*2139 1234*4276 5678*2139 5678*4276

12 252 468 714 1326 9

$*10^4 + *10^2 + *10^2 + *1 = 2639526$

| a | b |
| c | d |

$X \times Y = ac\ 10^n + (ad + bc)\ 10^{n/2} + bd$

## Multiplying (Divide & Conquer style)

12345678 * 21394276

1 2639526 5276584 12145242 24279128

$*10^8 + *10^4 + *10^4 +$

= 264126842539128

| a | b |
| c | d |

$X \times Y = ac\ 10^n + (ad + bc)\ 10^{n/2} + bd$

## Multiplying (Divide & Conquer style)

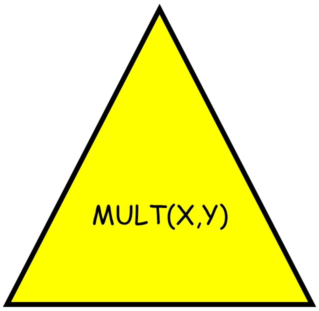
12345678 * 21394276

= 264126842539128

| a | b |
| c | d |

$X \times Y = ac\ 10^n + (ad + bc)\ 10^{n/2} + bd$

## Divide, Conquer, and Glue



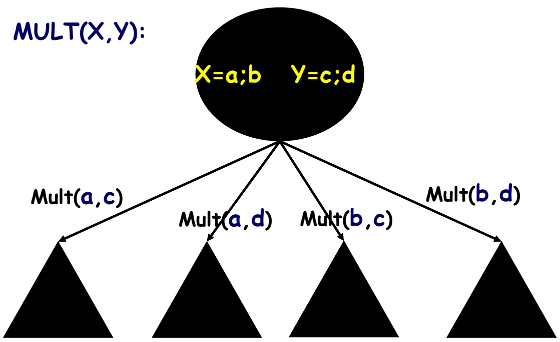

## Divide, Conquer, and Glue

**MULT(X,Y):**

**if |X| = |Y| = 1**
**then return XY,**
**else…**

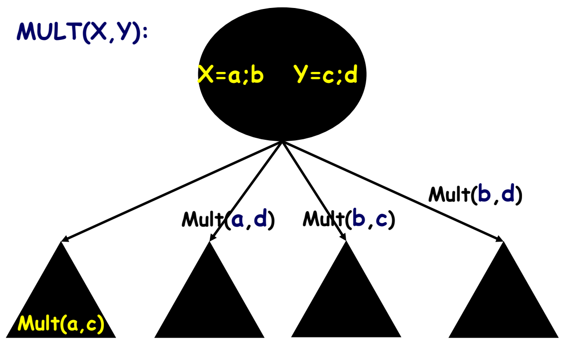
## Divide, Conquer, and Glue

**MULT(X,Y):**



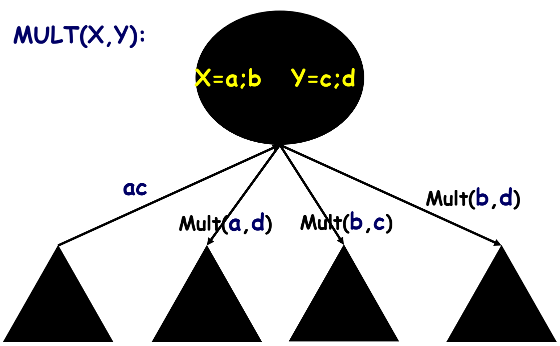

## Divide, Conquer, and Glue

**MULT(X,Y):**



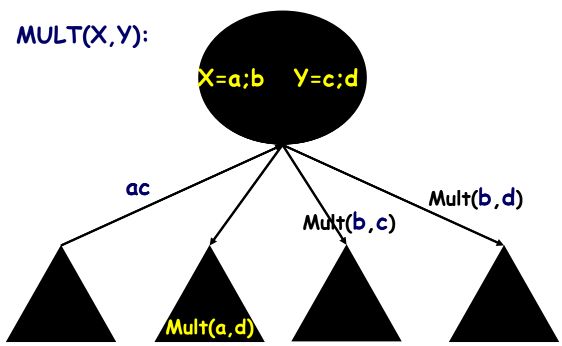

## Divide, Conquer, and Glue

**MULT(X,Y):**



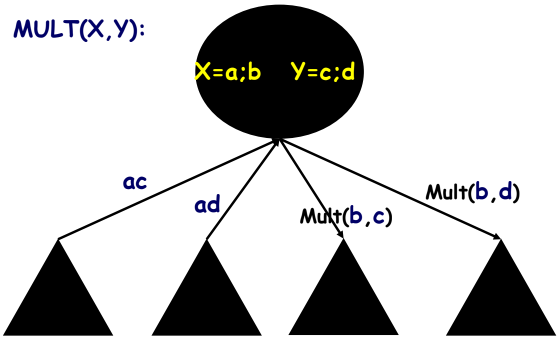

## Divide, Conquer, and Glue

**MULT(X,Y):**



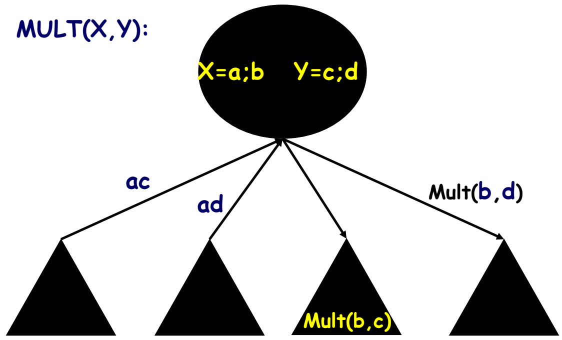
## Divide, Conquer, and Glue

MULT(X,Y):



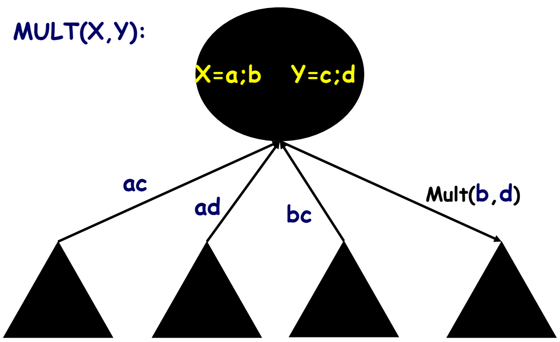

## Divide, Conquer, and Glue

MULT(X,Y):



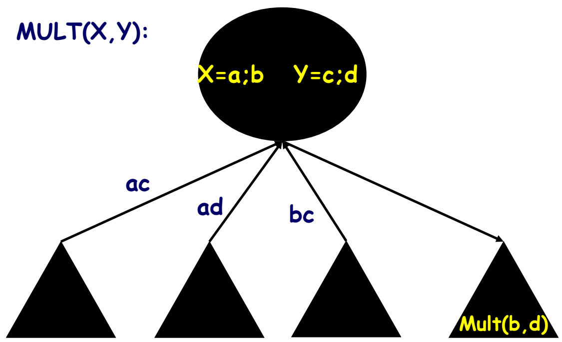

## Divide, Conquer, and Glue

MULT(X,Y):



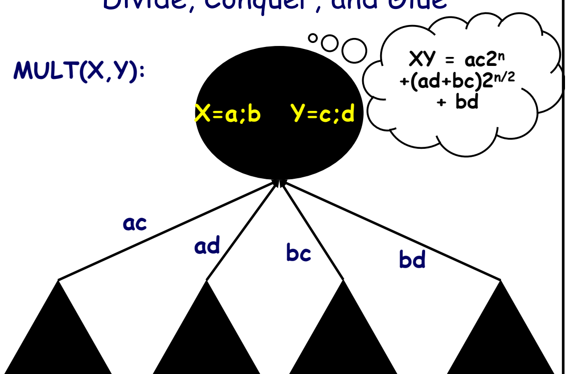

## Divide, Conquer, and Glue

MULT(X,Y):




## Divide, Conquer, and Glue

MULT(X,Y):




## Time required by MULT

T(n) = time taken by MULT on two n-bit numbers

What is T(n)? What is its growth rate?

Big Question: Is it $\Theta(n^2)$?

$$T(n) = 4\ T(n/2) + O(n)$$

conquering time

divide and glue

## Recurrence Relation

T(1) = 1

T(n) = 4 T(n/2) + O(n)

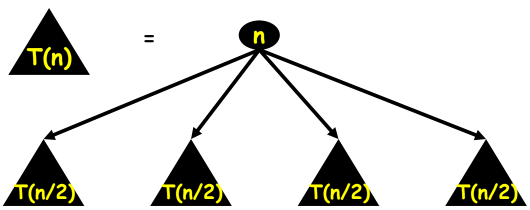
## Simplified Recurrence Relation

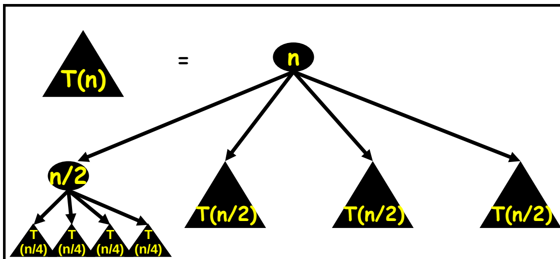
T(1) = 1

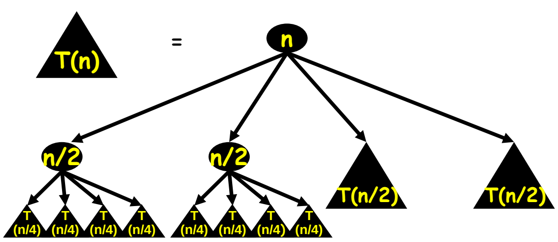
T(n) = 4 T(n/2) + n

conquering time

divide and glue




| | |
|---|---|
| 0 | n |
| 1 | n/2 + n/2 + n/2 + n/2 |
| 2 | |
| | |
| i | Level i is the sum of $4^i$ copies of $n/2^i$ |
| | |
| | . . . . . . . . . . . . . . . . . . . . . . . . . . |
| | 1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1 |

| | |
|---|---|
| $1n$ = | n |
| $2n$ = | n/2 + n/2 + n/2 + n/2 |
| $4n$ = | |
| | |
| $2^i n$ = | Level i is the sum of $4^i$ copies of $n/2^i$ |
| | |
| | . . . . . . . . . . . . . . . . . . . . . . . . . . |
| $(n)n$ = | 1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1 |

$$n(1+2+4+8+ \ . \ . \ . \ +n) = n(2n-1) = 2n^2-n$$

Divide and Conquer MULT: $\Theta(n^2)$ time
Grade School Multiplication: $\Theta(n^2)$ time

**Bummer!**

**MULT calls itself 4 times. Can you see a way to reduce the number of calls?**

## Gauss' Complex Puzzle

**Remember how to multiply two complex numbers a + bi and c + di?**

**$(a+bi)(c+di) = [ac - bd] + [ad + bc]\ i$**

**Input: a,b,c,d**
**Output: ac-bd, ad+bc**

**If multiplying two real numbers costs $1 and adding them costs a penny, what is the cheapest way to obtain the output from the input?**

**Can you do better than $4.03?**

## Gauss' $3.05 Method

**Input: a,b,c,d**
**Output: ac-bd, ad+bc**

| | | | |
|---|---|---|---|
| c | $X_1 = a + b$ | | |
| c | $X_2 = c + d$ | | |
| $ | $X_3 = X_1 X_2$ | | $= ac + ad + bc + bd$ |
| $ | $X_4 = ac$ | | |
| $ | $X_5 = bd$ | | |
| c | $X_6 = X_4 - X_5$ | | $= ac - bd$ |
| cc | $X_7 = X_3 - X_4 - X_5$ | | $= bc + ad$ |

The Gauss optimization saves one multiplication out of four.
It requires 25% less work.

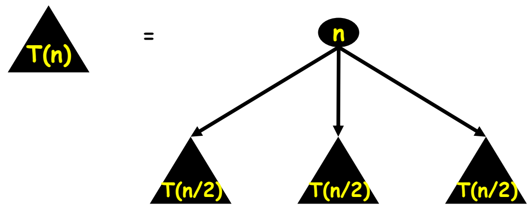
Karatsuba, Anatolii Alexeevich
(1937-2008)

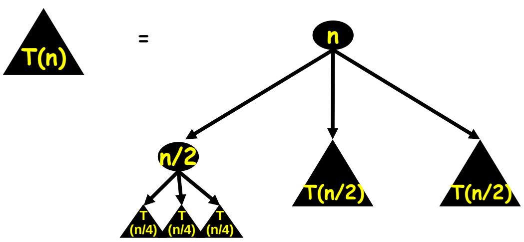
In 1962 Karatsuba had formulated the first algorithm to break the $n^2$ barrier!

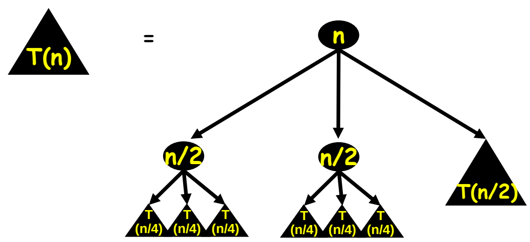
# Gaussified MULT
# (Karatsuba 1962)

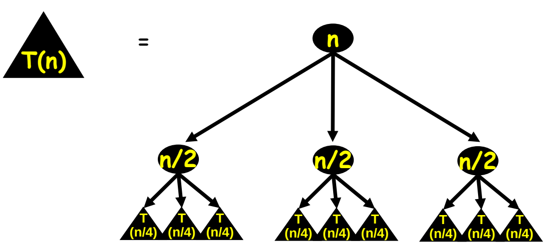
```
MULT(X,Y):
  If |X| = |Y| = 1 then return XY
  else break X into a;b and Y into c;d
        e : = MULT(a,c)
        f  := MULT(b,d)
  return
   e 2^n + (MULT(a+b,c+d) – e – f) 2^{n/2} + f
```

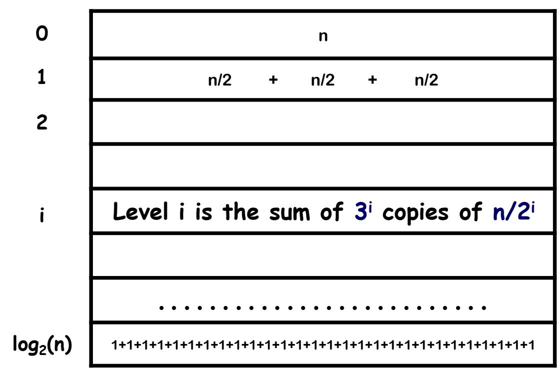
$T(n) = 3\,T(n/2) + n$



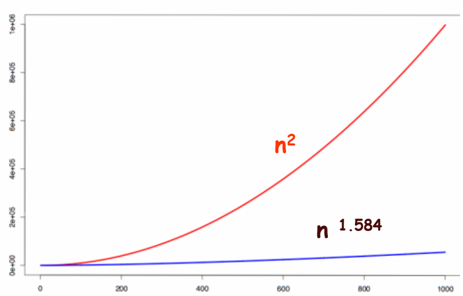
| | |
|---|---|
| $1n$ = | $n$ |
| $3/2n$ = | $n/2 \quad + \quad n/2 \quad + \quad n/2$ |
| $9/4n$ = | |
| | |
| $(3/2)^i n$ = | Level i is the sum of $3^i$ copies of $n/2^i$ |
| | |
| | . . . . . . . . . . . . . . . . . . . . . . . . . . |
| $(3/2)^{\log n} n$ = | 1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1 |

$$n(1+3/2+(3/2)^2+ \ . \ . \ . \ + (3/2)^{\log_2 n}) = 3n^{1.58\ldots} - 2n$$

## Dramatic Improvement for Large n

$$T(n) = 3n^{\log_2 3} - 2n$$
$$= \Theta(n^{\log_2 3})$$
$$= \Theta(n^{1.58\ldots})$$

A huge savings over $\Theta(n^2)$ when $n$ gets large.




## 3-Way Multiplication

The key idea of the algorithm is to divide a large integer into 3 parts (rather than 2) of size approximately n/3 and then multiply those parts.

$$154517766 = 154 * 10^6 + 517 * 10^3 + 766$$

## 3-Way Multiplication

Let

$$X = x_2\ 10^{2p} + x_1\ 10^p + x_0$$
$$Y = y_2\ 10^{2p} + y_1\ 10^p + y_0$$

Then

$$X*Y = 10^{4p}\ x_2y_2 + 10^{3p}\ (x_2y_1 + x_1y_2) + 10^{2p}\ (x_2y_0 + x_1y_1 + x_0y_2) + 10^p\ (x_1y_0 + x_0y_1) + x_0y_0$$

$$T(n) = 9\ T(n/3) + \Theta(n)$$

$$T(n) = \Theta(n^2)$$

## 3-Way Multiplication

Consider the equation in general form $p > 3$

$$T(n) = p\ T(n/3) + O(n)$$

Its solution is given by

$$T(n) = O(n^{\log_3 p})$$

Thus, this is faster if $p = 5$ or less

$$T(n) = O(n^{\log_3 5}) = O(n^{1.46\ldots})$$

## Is it possible to reduce the number of multiplications to 5?

Here is the system of new variables:

$x_0\, y_0 = Z_0$
$12\,(x_1y_0+x_0y_1) = 8\, Z_1 - Z_2 - 8\, Z_3 + Z_4$
$24\,(x_2y_0+x_1y_1+x_0y_2) = -30\, Z_0 + 16\, Z_1 - Z_2 + 16\, Z_3 - Z_4$
$12\,(x_2y_1+x_1y_2) = -2\, Z_1 + Z_2 + 2\, Z_3 - Z_4$
$24\, x_2y_2 = 6\, Z_0 - 4\, Z_1 + Z_2 - 4\, Z_3 + Z_4$

## Is it possible to reduce the number of multiplications to 5?

We rewrite all multiplications in terms of $Z_k$

$Z_0 = x_0\, y_0$
$Z_1 = (x_0+x_1+x_2)\,(y_0+y_1+y_2)$
$Z_2 = (x_0+2\, x_1+4\, x_2)\,(y_0+2\, y_1+4\, y_2)$
$Z_3 = (x_0-x_1+x_2)\,(y_0-y_1+y_2)$
$Z_4 = (x_0-2\, x_1+4\, x_2)\,(y_0-2\, y_1+4\, y_2)$

## Further Generalizations

It is possible to develop a faster algorithm by increasing the number of splits.

A 4-way splitting:

$T(n) = 7\, T(n/4) + O(n)$

$T(n) = O(n^{1.403\ldots})$

## Further Generalizations

Intuitively, the k-way split requires 2 k - 1 multiplications.

A k-way splitting:

$T(n) = (2k-1)\, T(n/k) + O(n)$

$T(n) = O(n^{\log_k(2k-1)})$

$n^{1.58}$, $n^{1.46}$, $n^{1.40}$, $n^{1.36}$, $n^{1.33}$

Note, we will never get a linear performance

## Is it always possible to find such 2k-1 multiplications?

Consider two polynomials of k-1 degree

$polyn_1 = a_{k-1}\, x^{k-1} + a_{k-2}\, x^{k-2} + \ldots + a_1\, x + a_0$
$polyn_2 = b_{k-1}\, x^{k-1} + b_{k-2}\, x^{k-2} + \ldots + b_1\, x + b_0$

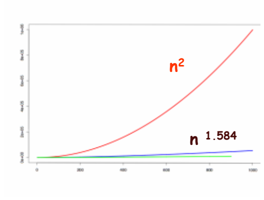
Each polynomial is defined by k coefficients.

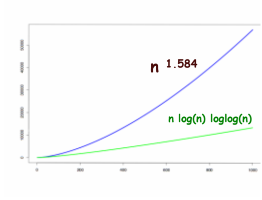
When we multiply $polyn_1 * polyn_2$ we get a polynomial of 2k-2 degree

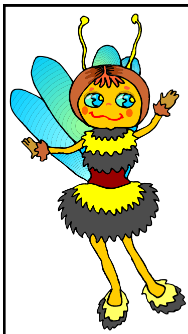
that has exactly 2k-1 coefficients. Therefore, it's uniquely defined by 2k-1 values.

## Multiplication Algorithms

| | |
|---|---|
| Grade School | $n^2$ |
| Karatsuba | $n^{1.58\ldots}$ |
| Fast Fourier Transform | n logn loglogn |



Study Bee

- Divide and Conquer
- Karatsuba Multiplication
- Solving Recurrences