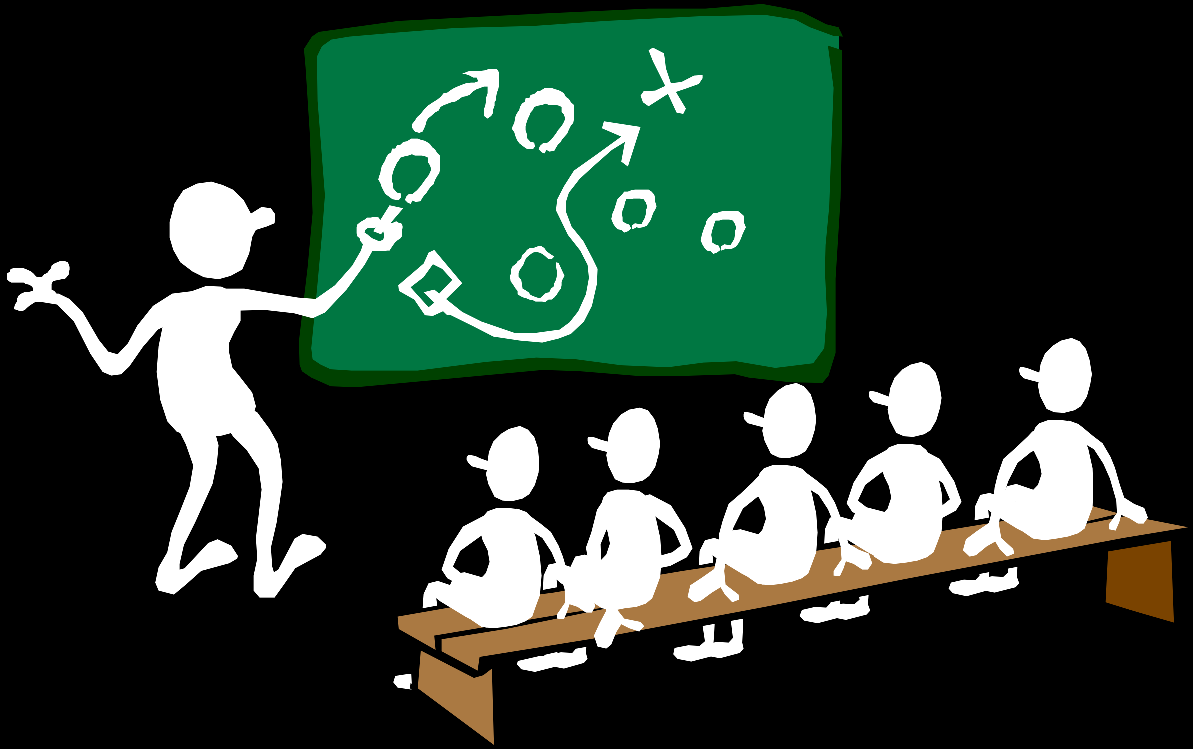
15-251

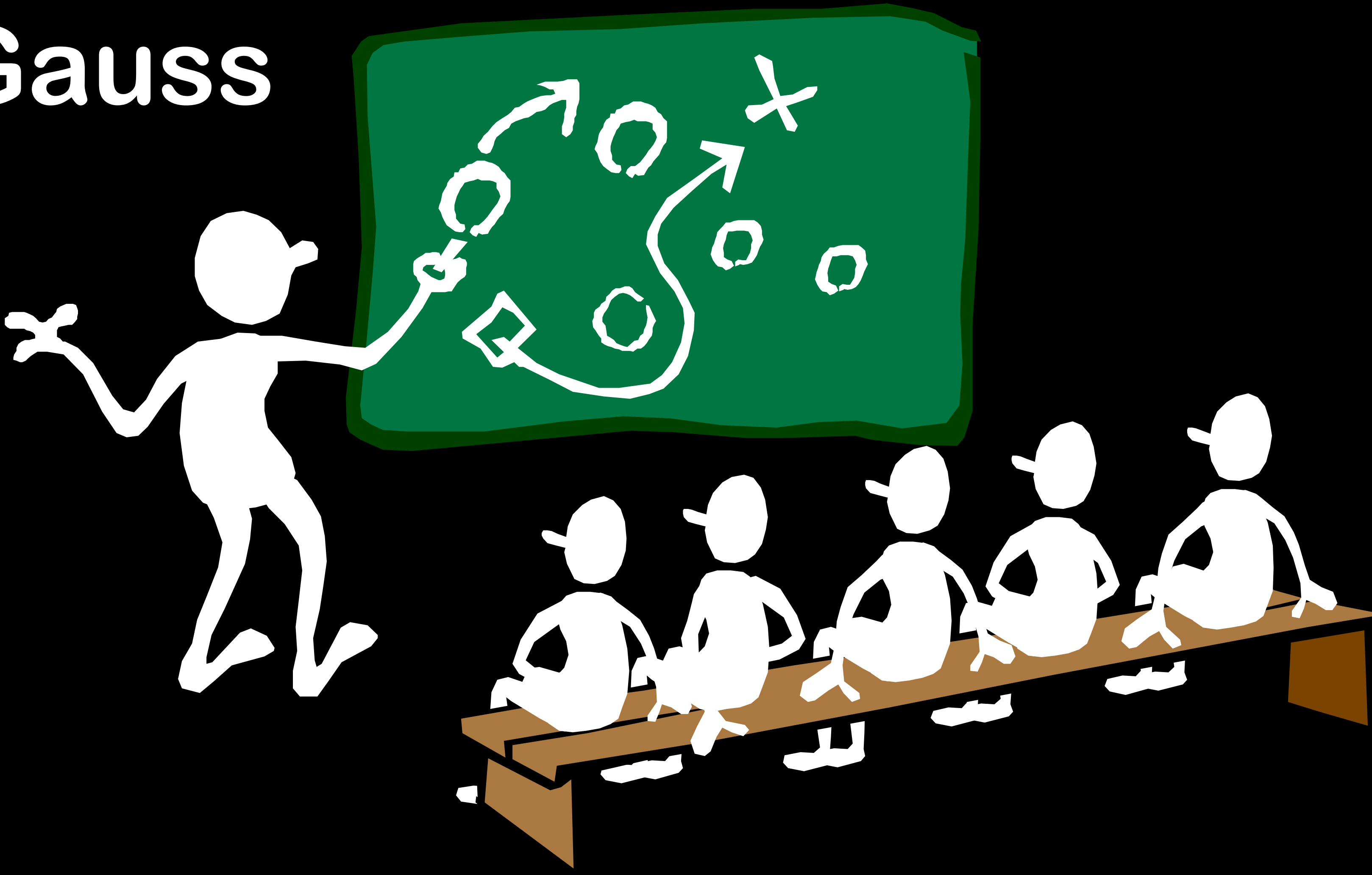
Great Theoretical Ideas in Computer Science

Grade School Revisited: How To Multiply Two Numbers

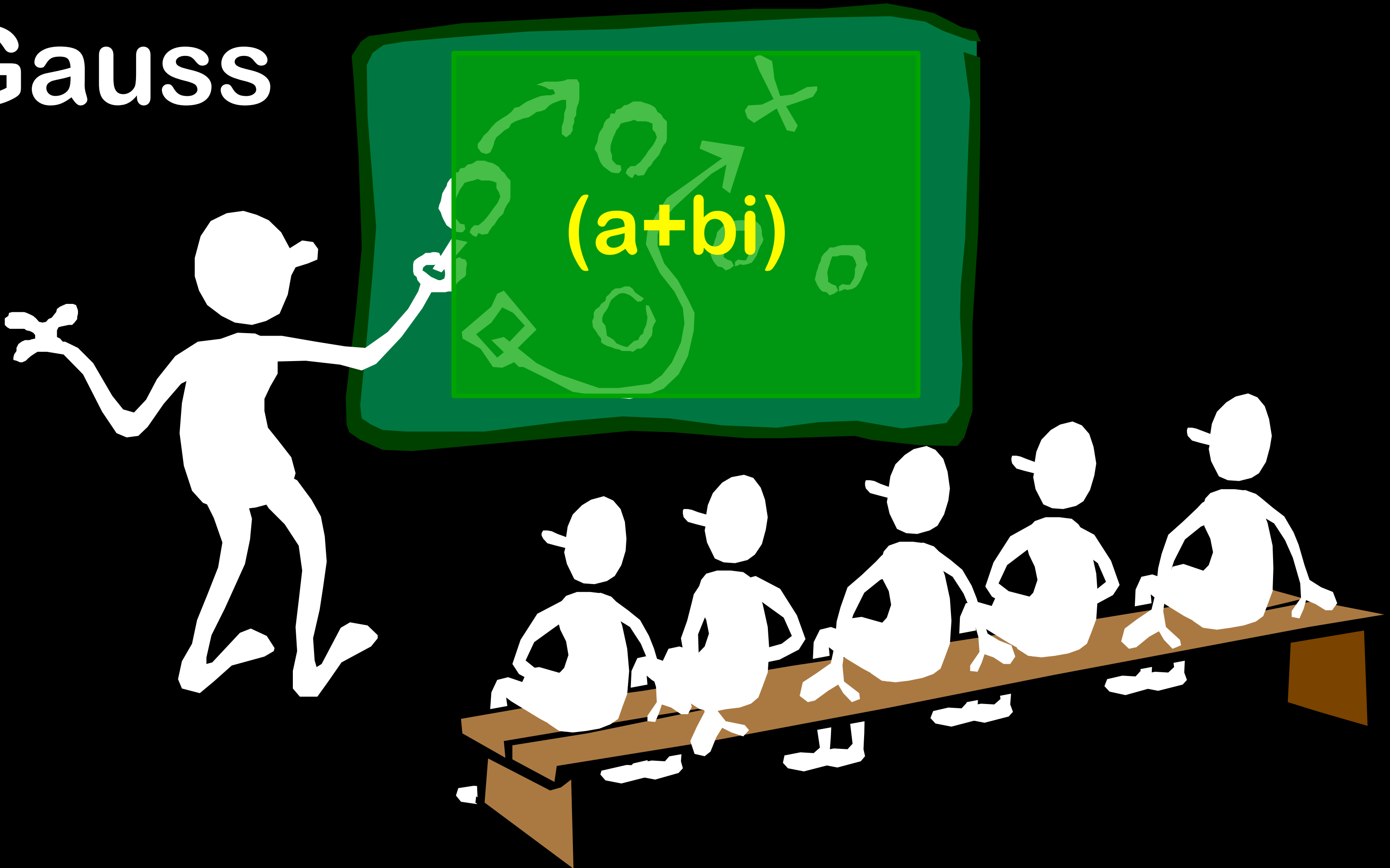
Lecture 22, November 6, 2008



Gauss



Gauss



Gauss' Complex Puzzle

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

Gauss' Complex Puzzle

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

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

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

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?

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 $4.02?

Gauss' $3.05 Method

Input: a,b,c,d

Output: ac-bd, ad+bc

Gauss' $3.05 Method

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

$X_1 = a + b$

Gauss' $3.05 Method

Input: a,b,c,d

Output: ac-bd, ad+bc

c $X_1 = a + b$

Gauss' $3.05 Method

Input: a,b,c,d

Output: ac-bd, ad+bc

c $X_1 = a + b$

$X_2 = c + d$

Gauss' $3.05 Method

Input: a,b,c,d

Output: ac-bd, ad+bc

c $X_1 = a + b$

c $X_2 = c + d$

Gauss' $3.05 Method

Input: a,b,c,d

Output: ac-bd, ad+bc

c $X_1 = a + b$

c $X_2 = c + d$

$X_3 = X_1 X_2 = ac + ad + bc + bd$

Gauss' $3.05 Method

Input: a,b,c,d

Output: ac-bd, ad+bc

c $X_1 = a + b$

c $X_2 = c + d$

\$ $X_3 = X_1 X_2$ $= ac + ad + bc + bd$

Gauss' \$3.05 Method

Input: a,b,c,d

Output: ac-bd, ad+bc

c $X_1 = a + b$

c $X_2 = c + d$

\$ $X_3 = X_1 X_2 = ac + ad + bc + bd$

$X_4 = ac$

Gauss' $3.05 Method

Input: a,b,c,d

Output: ac-bd, ad+bc

c $X_1 = a + b$

c $X_2 = c + d$

\$ $X_3 = X_1 X_2$ $= ac + ad + bc + bd$

\$ $X_4 = ac$

Gauss' $3.05 Method

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

c $X_1 = a + b$

c $X_2 = c + d$

\$ $X_3 = X_1 X_2 = ac + ad + bc + bd$

\$ $X_4 = ac$

$X_5 = bd$

Gauss' \$3.05 Method

Input: a,b,c,d

Output: ac-bd, ad+bc

c $X_1 = a + b$

c $X_2 = c + d$

\$ $X_3 = X_1 X_2 = ac + ad + bc + bd$

\$ $X_4 = ac$

\$ $X_5 = bd$

Gauss' $3.05 Method

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

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

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 $X_6 = X_4 – X_5$ $= ac - bd$

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	$X_6 = X_4 – X_5$	$= ac - bd$
	$X_7 = X_3 – X_4 – X_5$	$= bc + ad$

Gauss' $3.05 Method

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

c $X_1 = a + b$

c $X_2 = c + d$

\$ $X_3 = X_1 X_2 = ac + ad + bc + bd$

\$ $X_4 = ac$

\$ $X_5 = bd$

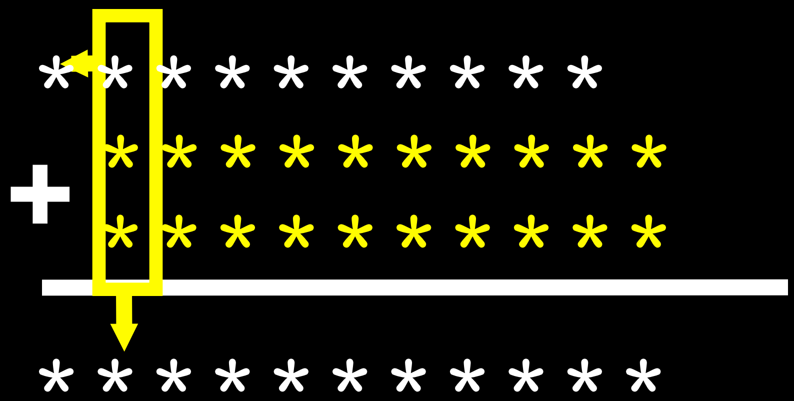
c $X_6 = X_4 – X_5 = ac - bd$

cc $X_7 = X_3 – X_4 – X_5 = bc + ad$

The Gauss optimization saves one multiplication out of four.

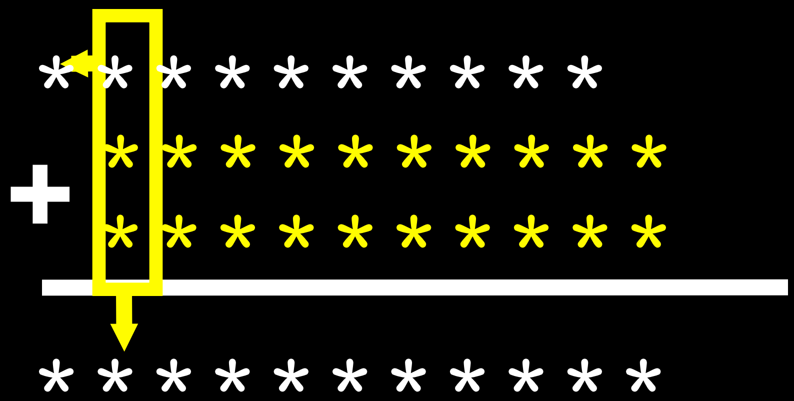
It requires 25% less work.

Time complexity of grade school addition



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

Time complexity of grade school addition

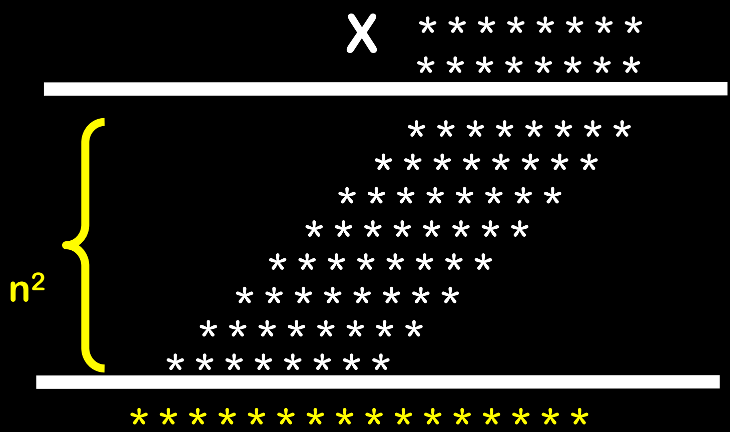


T(n) = amount of time grade school addition uses to add two n-bit numbers

We saw that T(n) was linear

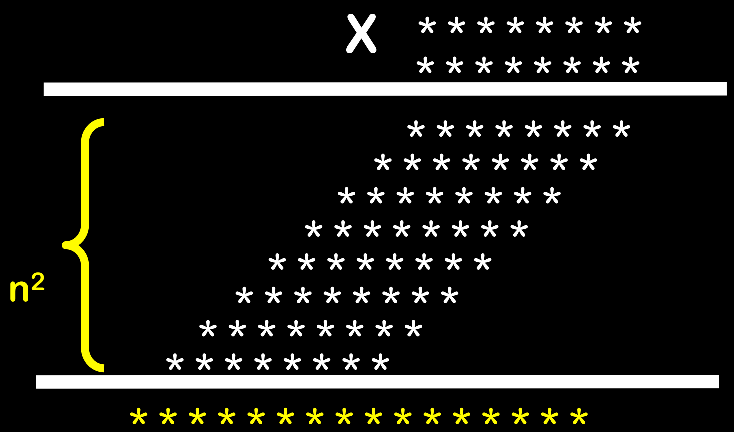
$T(n) = \Theta(n)$

Time complexity of grade school multiplication



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

Time complexity of grade school multiplication

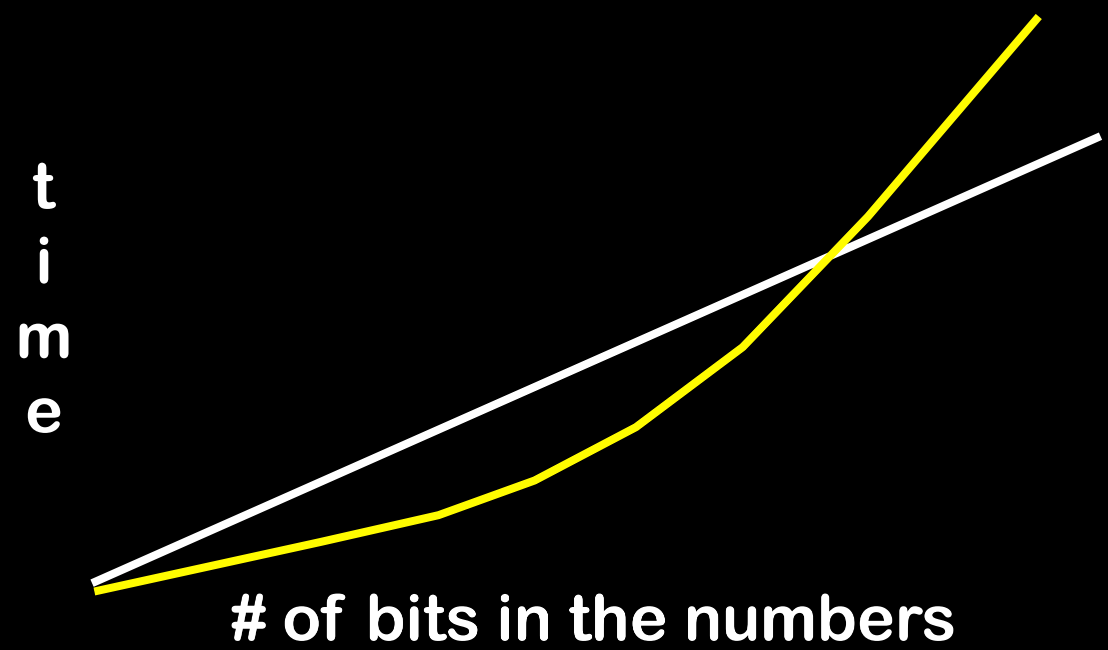


T(n) = The amount of time grade school multiplication uses to add two n-bit numbers

We saw that T(n) was 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 $\Omega(n)$ time

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

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

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

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

Any addition algorithm takes Ω(n) time

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 position i in one of the numbers

Any addition algorithm takes Ω(n) time

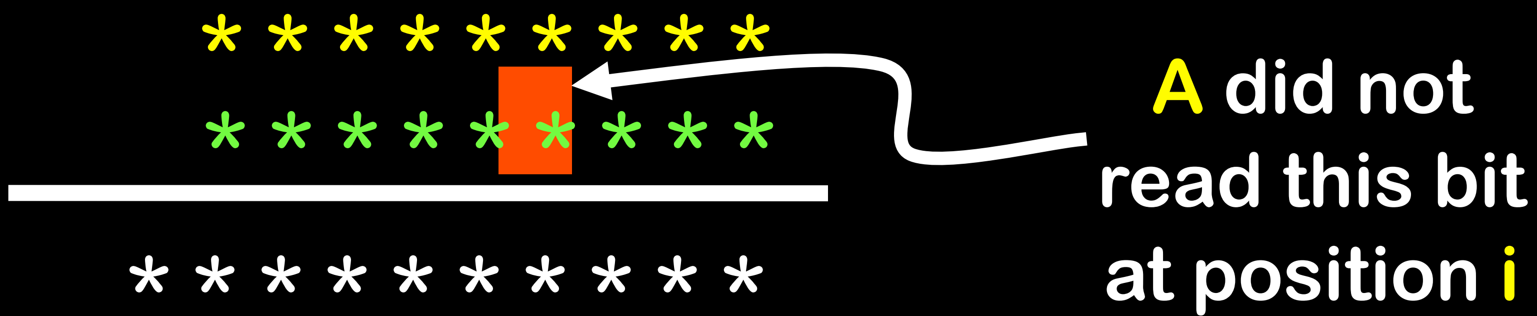
* * * * * * * * *

* * * * * * * * *

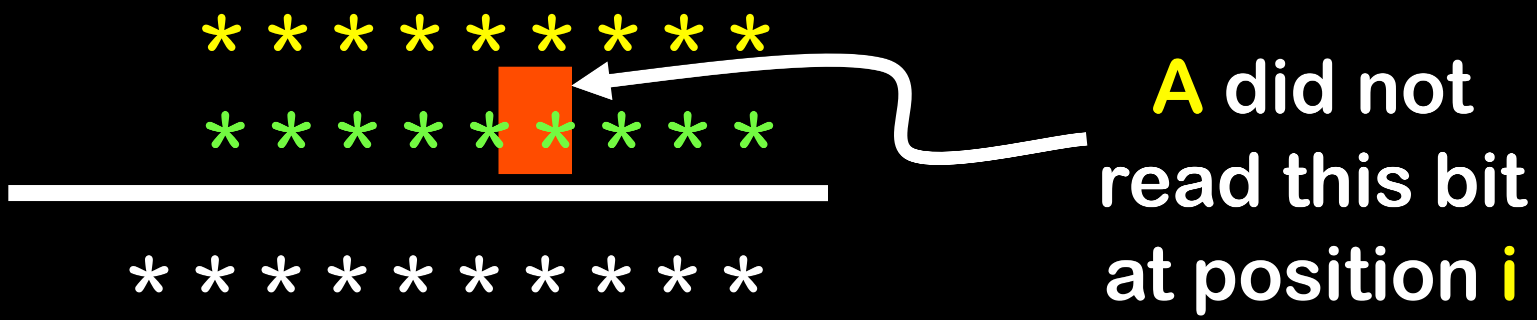
---

* * * * * * * * * *

Any addition algorithm takes Ω(n) time

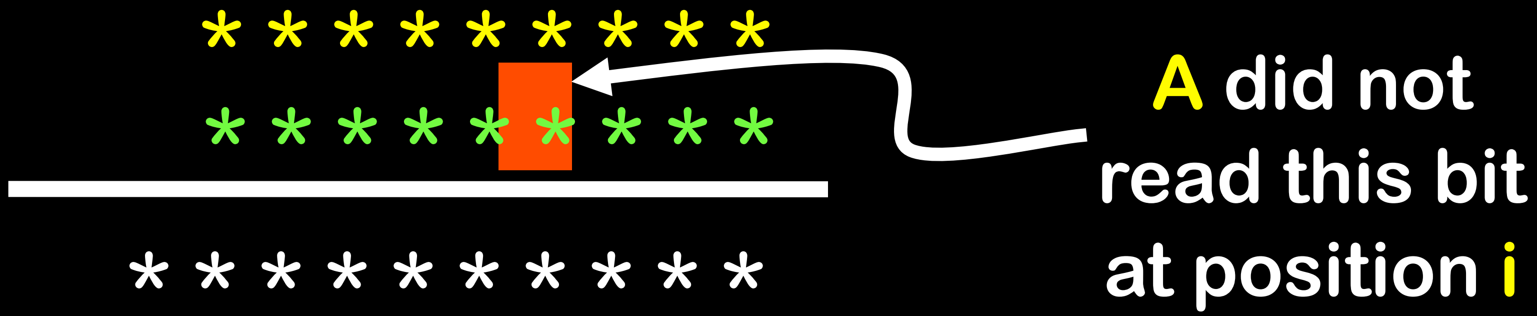


Any addition algorithm takes Ω(n) time



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

Any addition algorithm takes Ω(n) time



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

If A is correct, flip the bit at position i and give A the new pair of numbers. 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 Addition: $\Theta(n)$ time.
Furthermore, it is optimal

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

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?

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

Divide And Conquer

An approach to faster algorithms:

Divide And Conquer

An approach to faster algorithms:

DIVIDE a problem into smaller subproblems

Divide And Conquer

An approach to faster algorithms:

DIVIDE a problem into smaller subproblems

CONQUER them recursively

Divide And Conquer

An approach to faster algorithms:

DIVIDE a problem into smaller subproblems

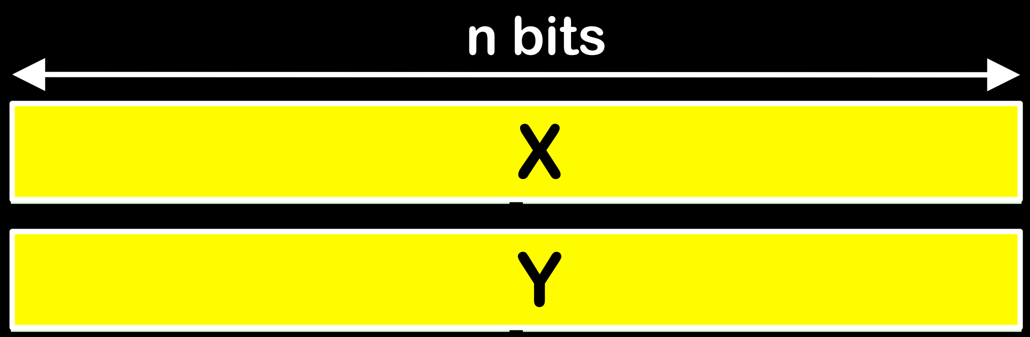
CONQUER them recursively

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

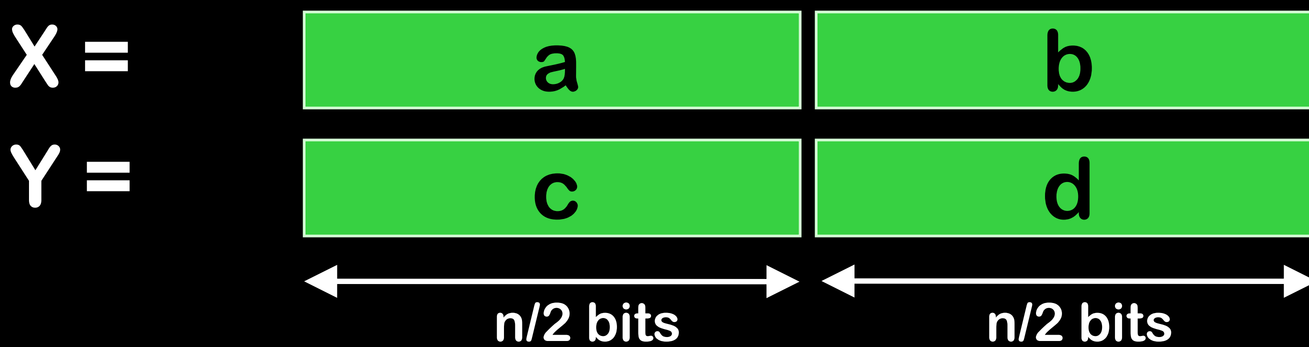
Multiplication of 2 n-bit numbers

X =

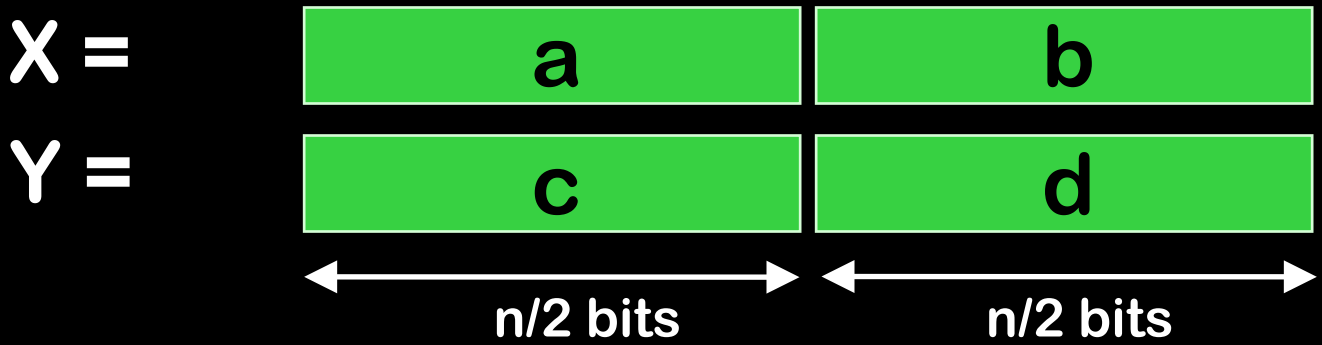
Y =



Multiplication of 2 n-bit numbers

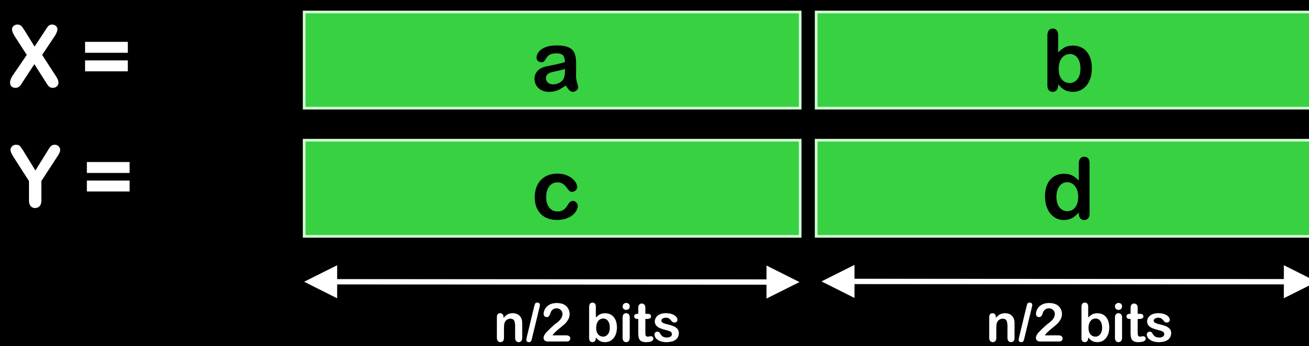


Multiplication of 2 n-bit numbers



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

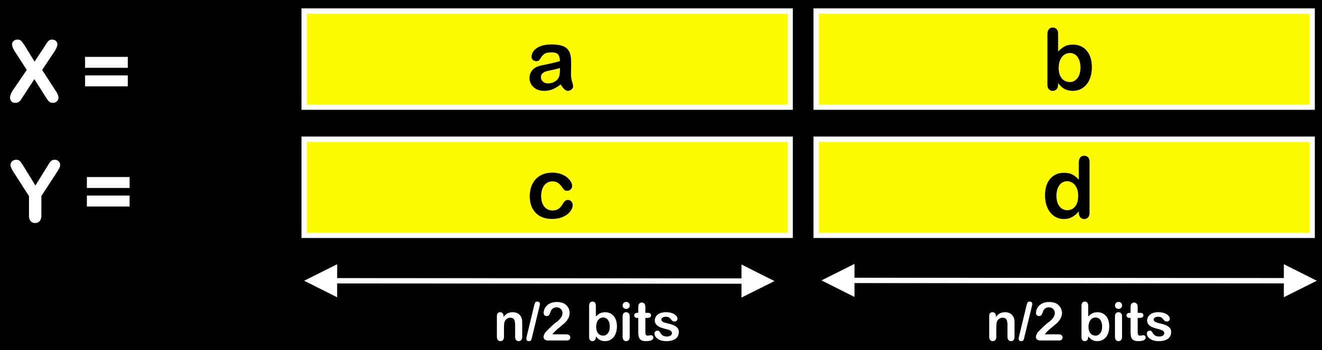
Multiplication of 2 n-bit numbers



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

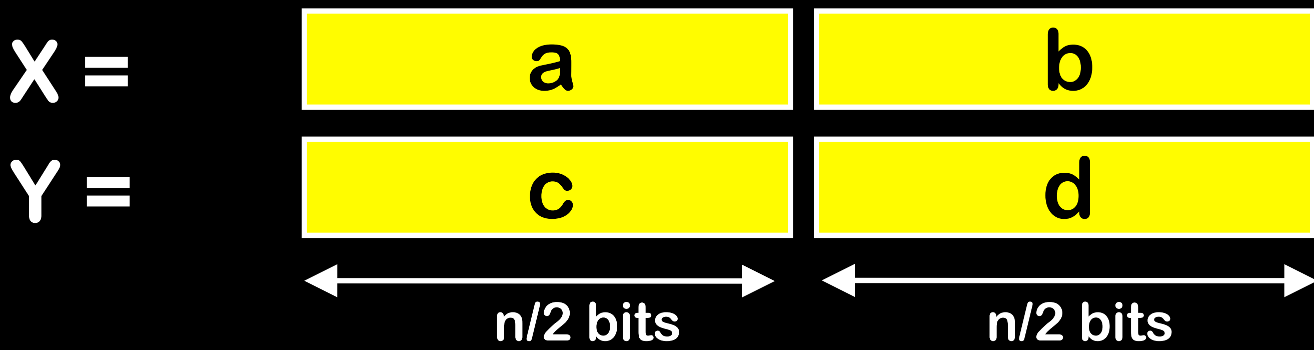
$$X \times Y = ac\,2^{n} + (ad + bc)\,2^{n/2} + bd$$

Multiplication of 2 n-bit numbers



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

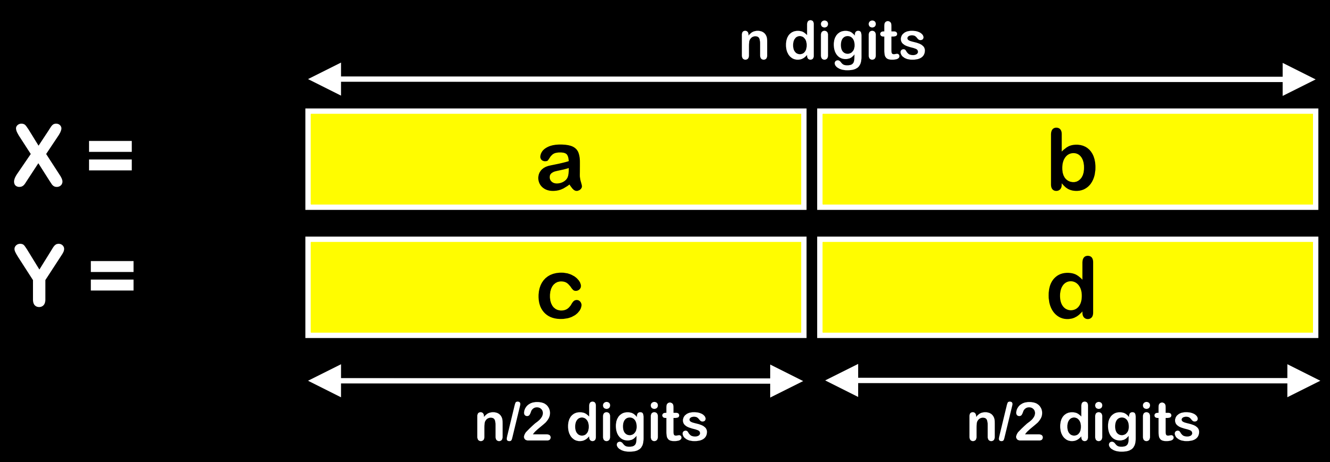
Multiplication of 2 n-bit numbers



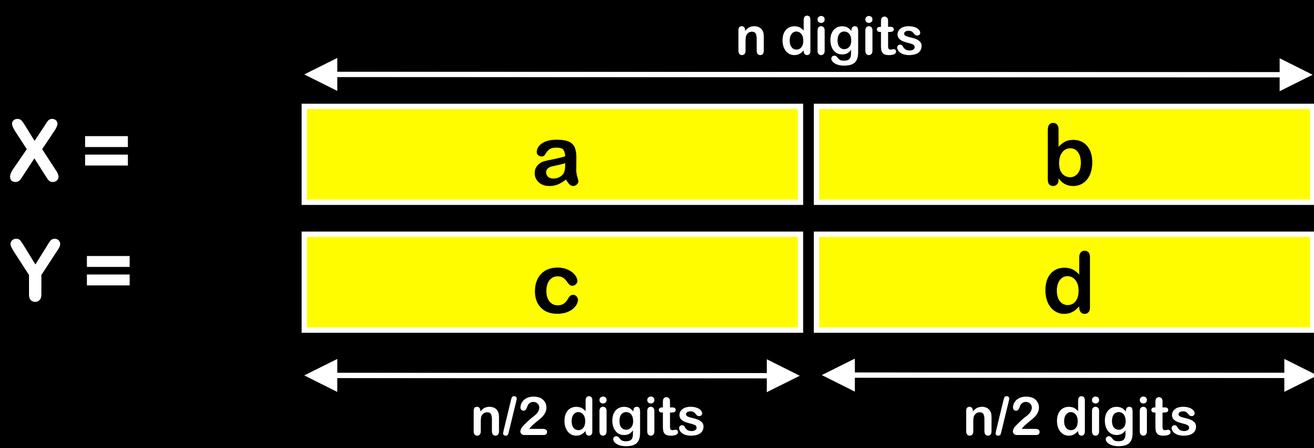
$$X \times Y = ac\, 2^n + (ad + bc)\, 2^{n/2} + bd$$

```
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!

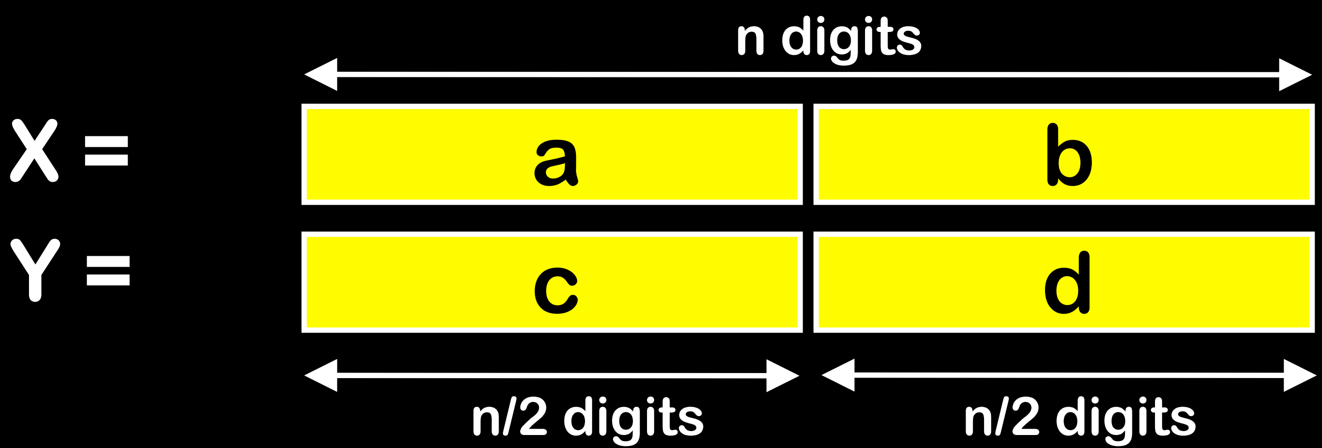


Same thing for numbers in decimal!



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

Same thing for numbers in decimal!



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

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

Multiplying (Divide & Conquer style)

X =	a	b
Y =	c	d

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

Multiplying (Divide & Conquer style)

12345678 * 21394276

X =	a	b
Y =	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

X =	a	b
Y =	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*21 12*39 34*21 34*39

X =	a	b
Y =	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*21   12*39   34*21   34*39

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

X =	a	b
Y =	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*21 12*39 34*21 34*39

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

2 1 4 2

X =	a	b
Y =	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*21 12*39 34*21 34*39

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

2 1 4 2

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

X =	a	b
Y =	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

2521 12*39 34*21 34*39

X =	a	b
Y =	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

2521 4689 34*21 34*39

X =	a	b
Y =	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

2521 4689 7141 34*39

X = | a | b

Y = | 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

2521   4689   7141   1326

X =	a	b
Y =	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

2521 4689 7141 1326

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

X =	a	b
Y =	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

2521 4689 7141 1326

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

X =	a	b
Y =	c	d

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

Multiplying (Divide & Conquer style)

12345678 * 21394276

2639526 9 1234*4276 5678*2139 5678*4276

X =	a	b
Y =	c	d

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

Multiplying (Divide & Conquer style)

12345678 * 21394276

26395269 52765846 5678*2139 5678*4276

X =	a	b
Y =	c	d

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

Multiplying (Divide & Conquer style)

12345678 * 21394276

26395269 52765846 12145242 5678*4276

X =	a	b
Y =	c	d

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

Multiplying (Divide & Conquer style)

12345678 * 21394276

26395269 52765846 12145242 24279128

X =	a	b
Y =	c	d

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

Multiplying (Divide & Conquer style)

12345678 * 21394276

2639526 9 5276584 6 12145242 24279128

$*10^8$ + $*10^4$ + $*10^4$ + $*1$

X =	a	b
Y =	c	d

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

Multiplying (Divide & Conquer style)

12345678 * 21394276

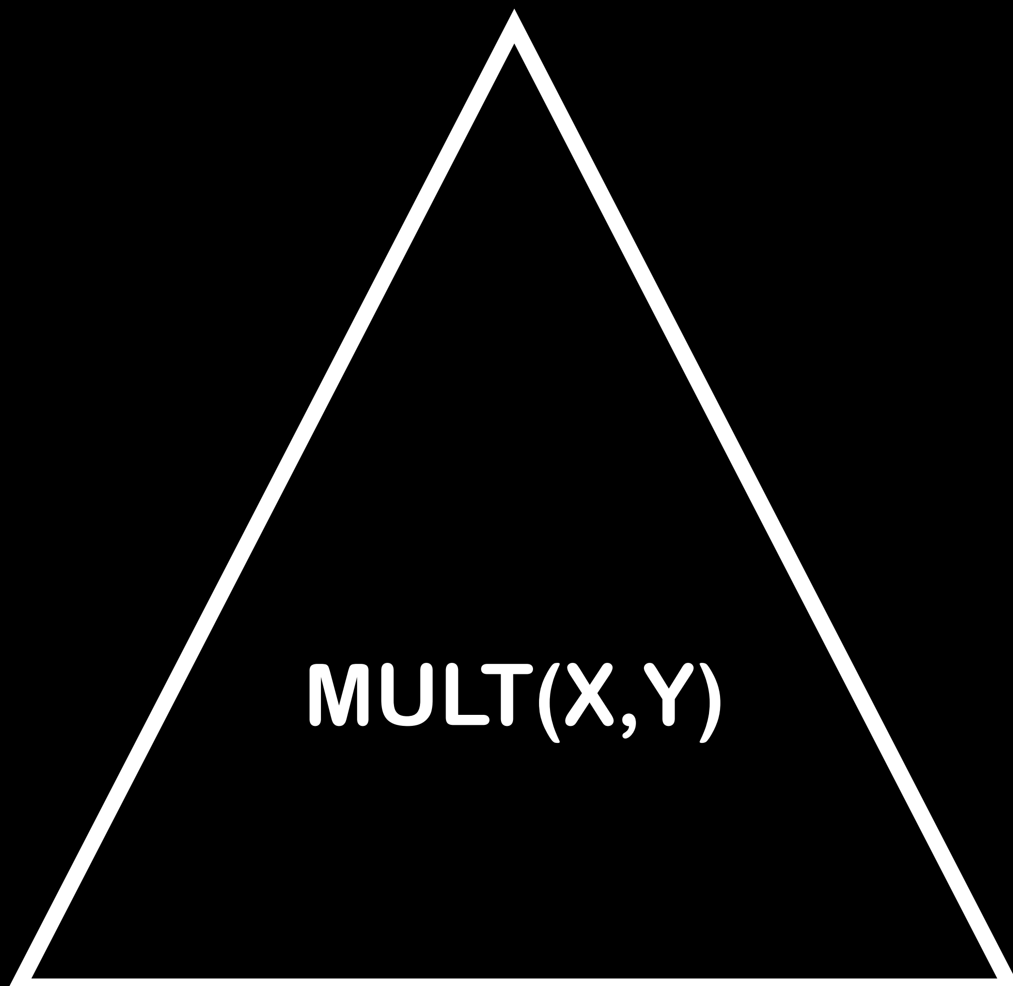
26395269 $*10^8$ + 52765846 $*10^4$ + 12145242 $*10^4$ + 24279128 $*1$

= 264126842539128

X =	a	b
Y =	c	d

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

Divide, Conquer, and Glue



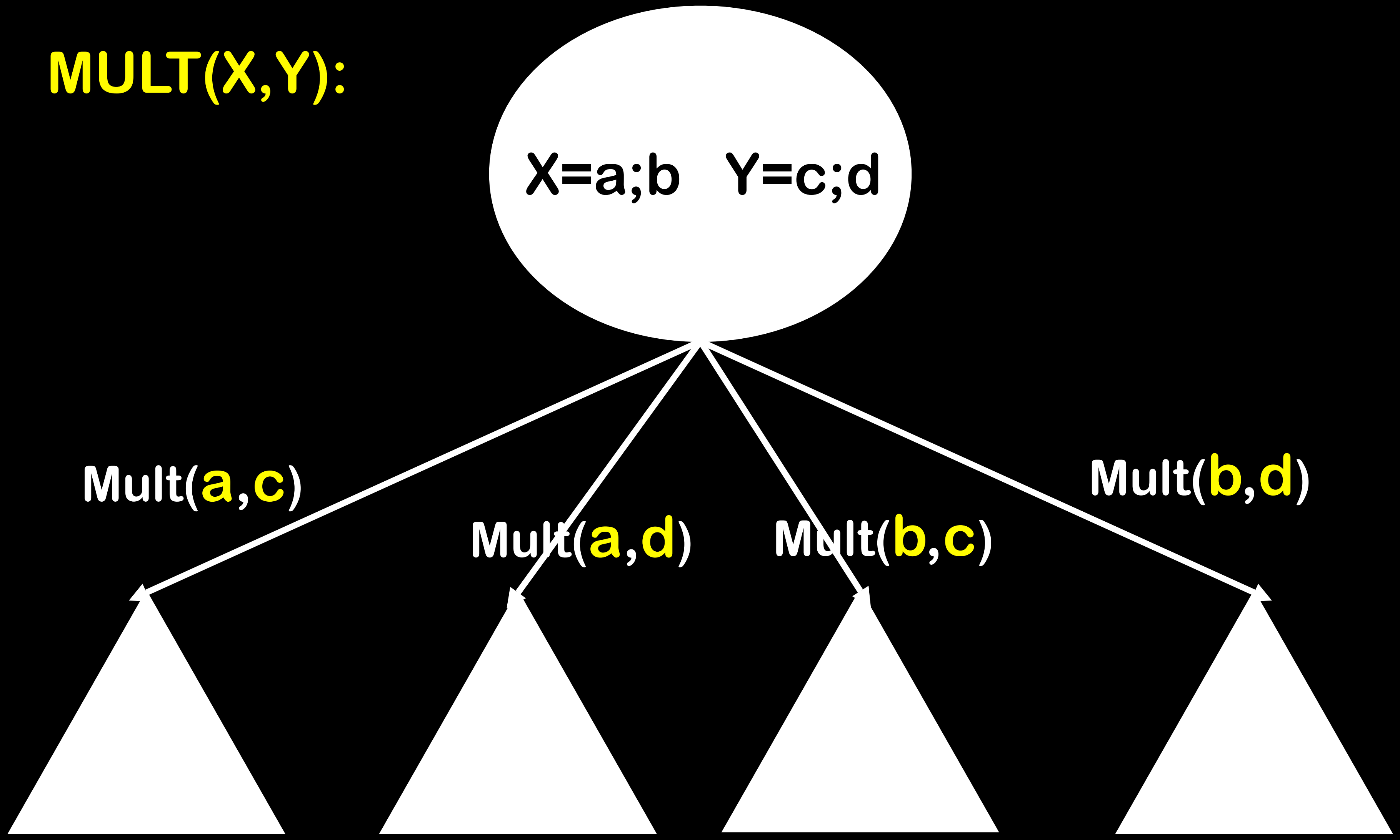
Divide, Conquer, and Glue

MULT(X,Y):

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

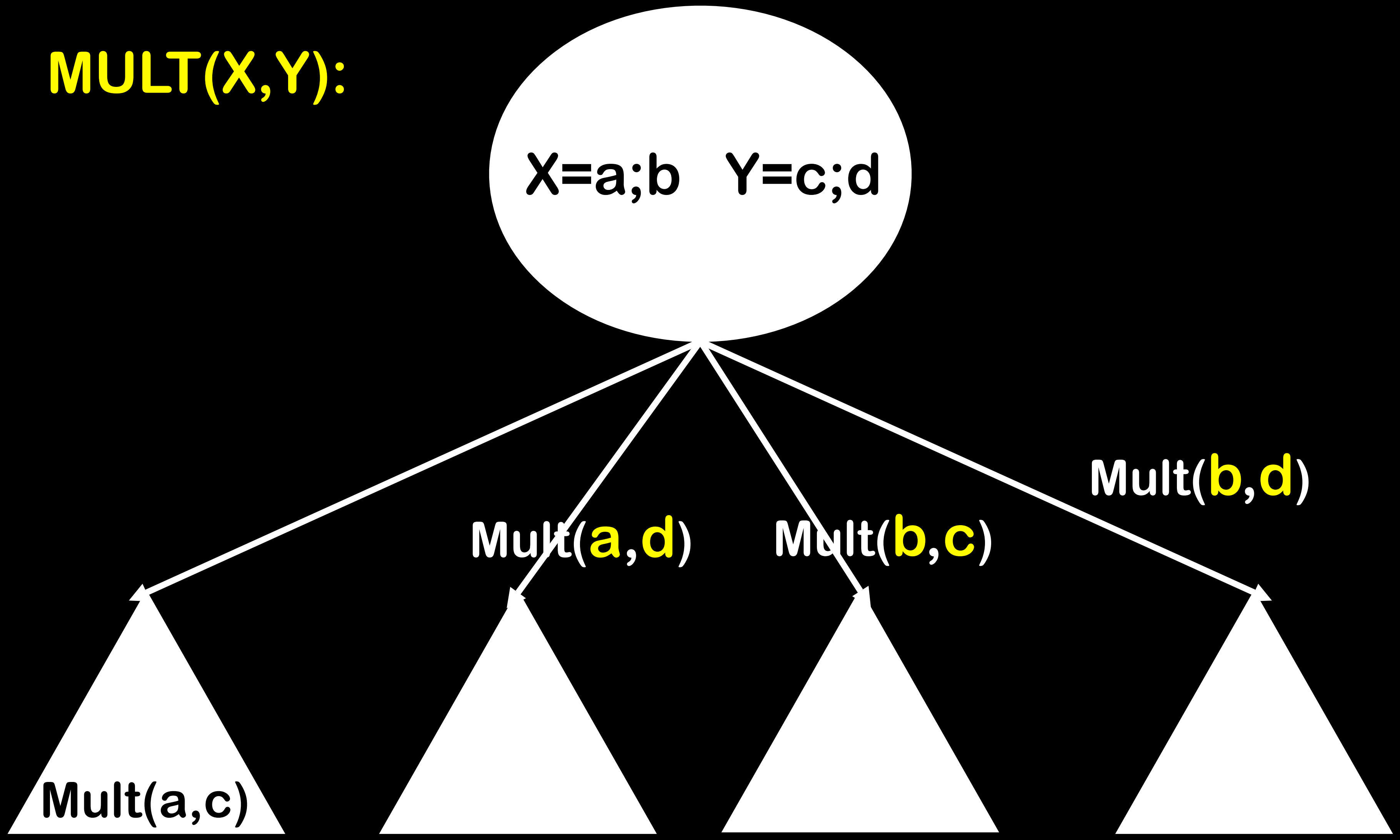
Divide, Conquer, and Glue

MULT(X,Y):



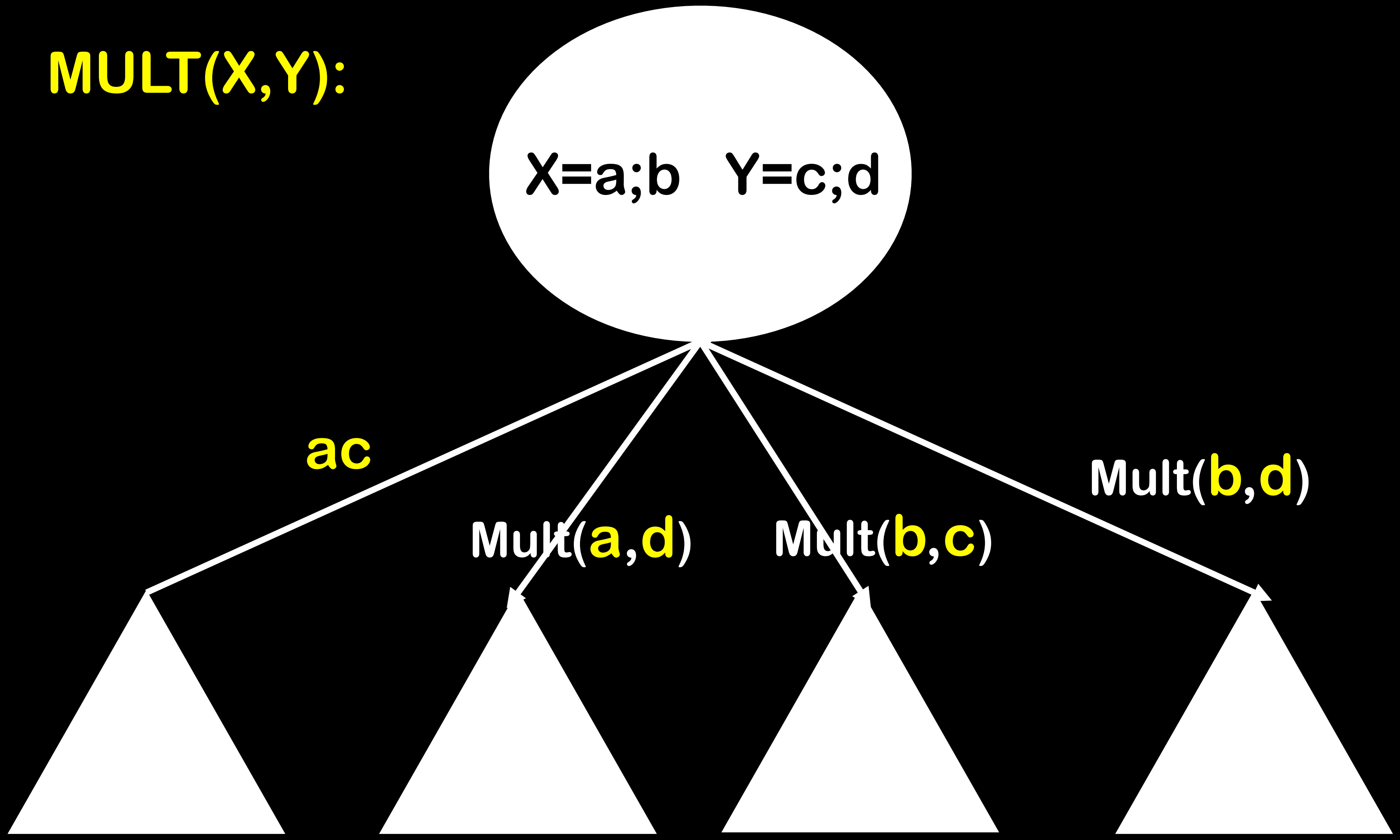
Divide, Conquer, and Glue

MULT(X,Y):



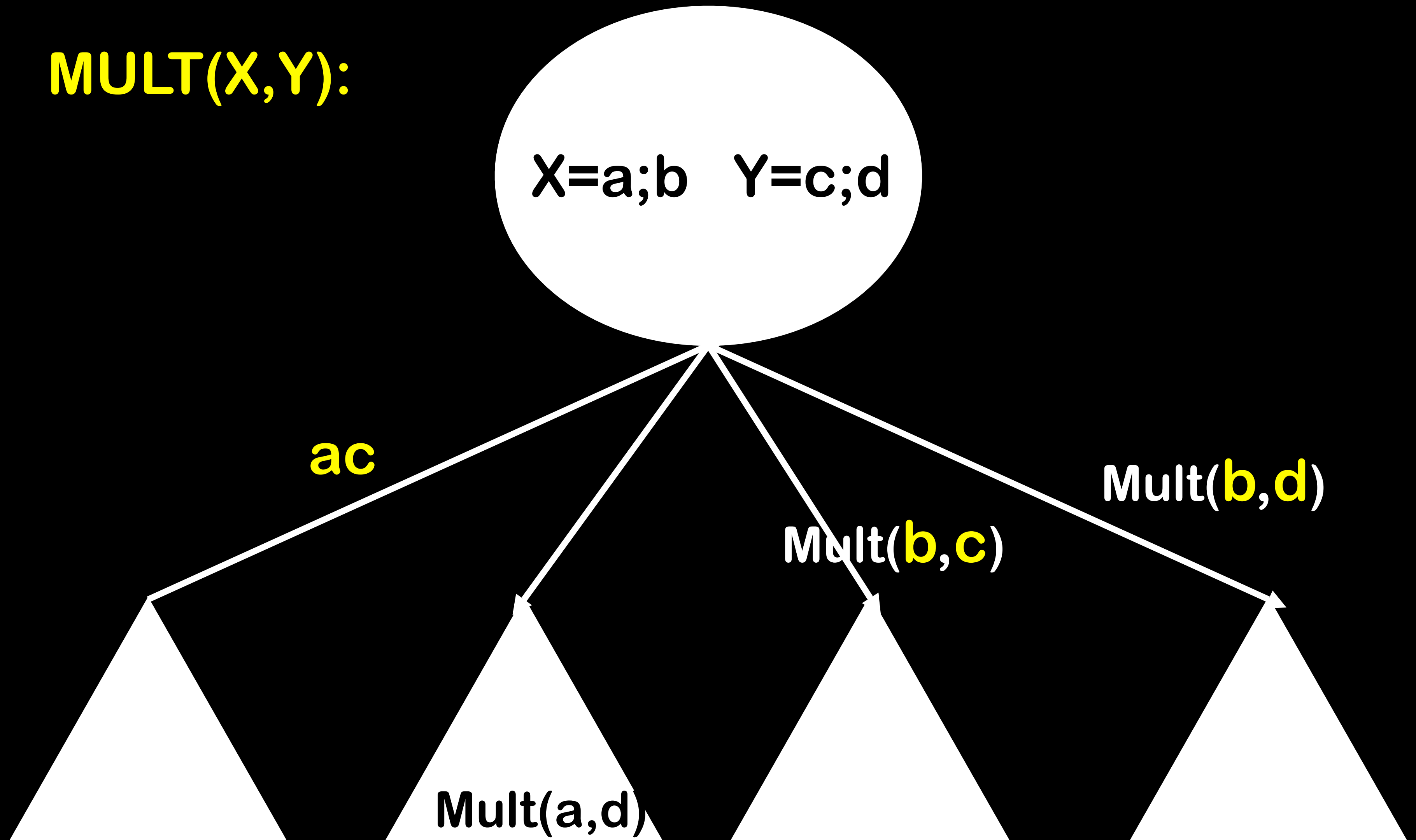
Divide, Conquer, and Glue

MULT(X,Y):



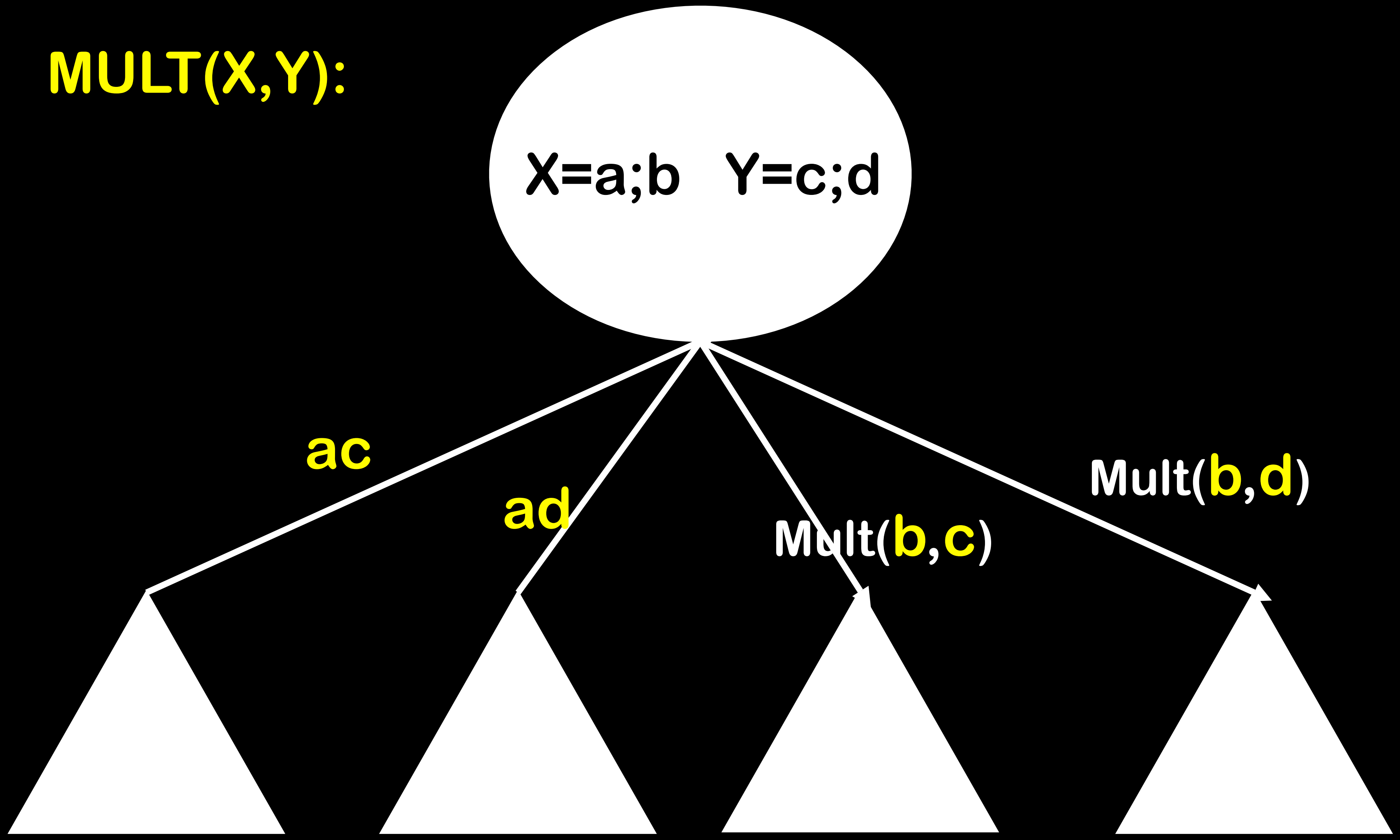
Divide, Conquer, and Glue

MULT(X,Y):



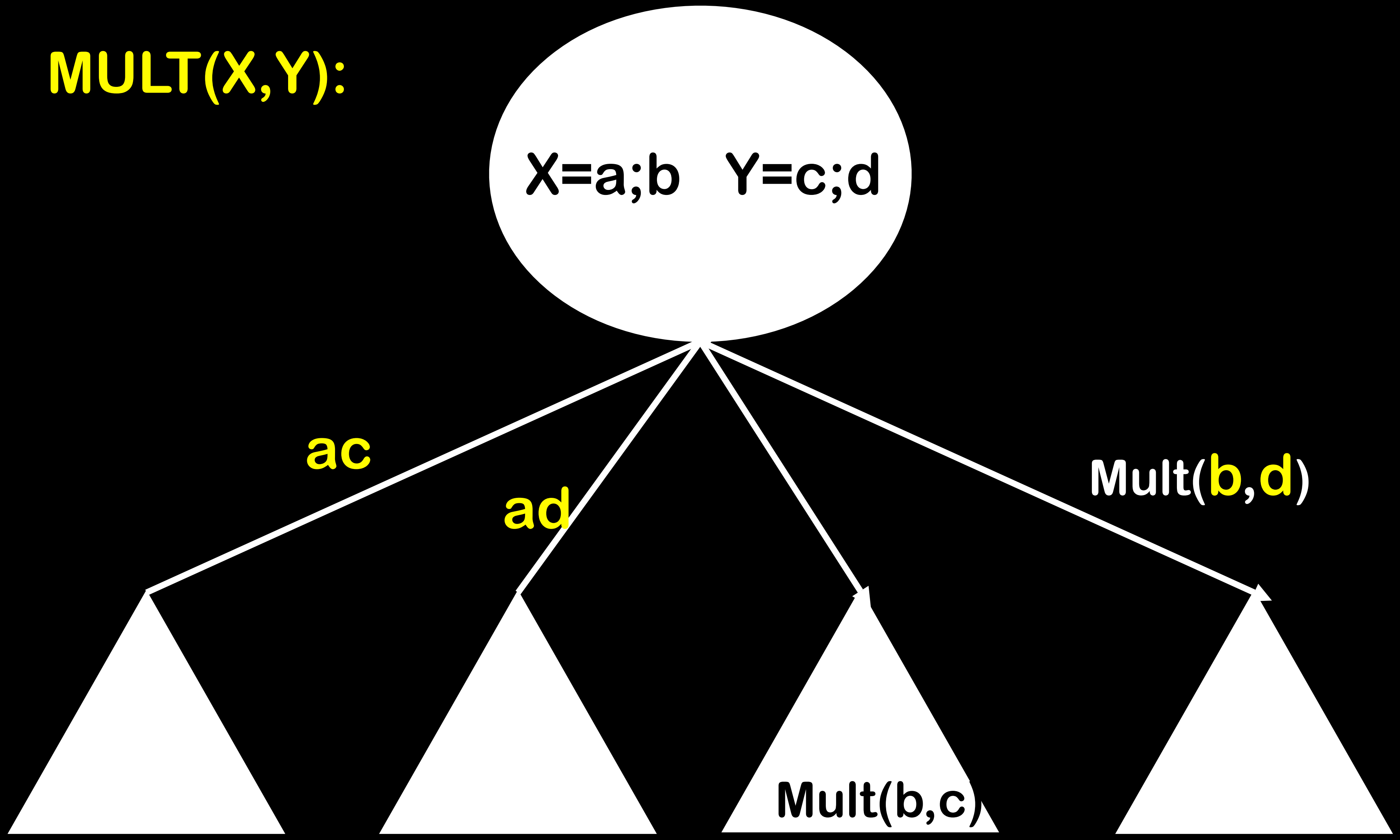
Divide, Conquer, and Glue

MULT(X,Y):



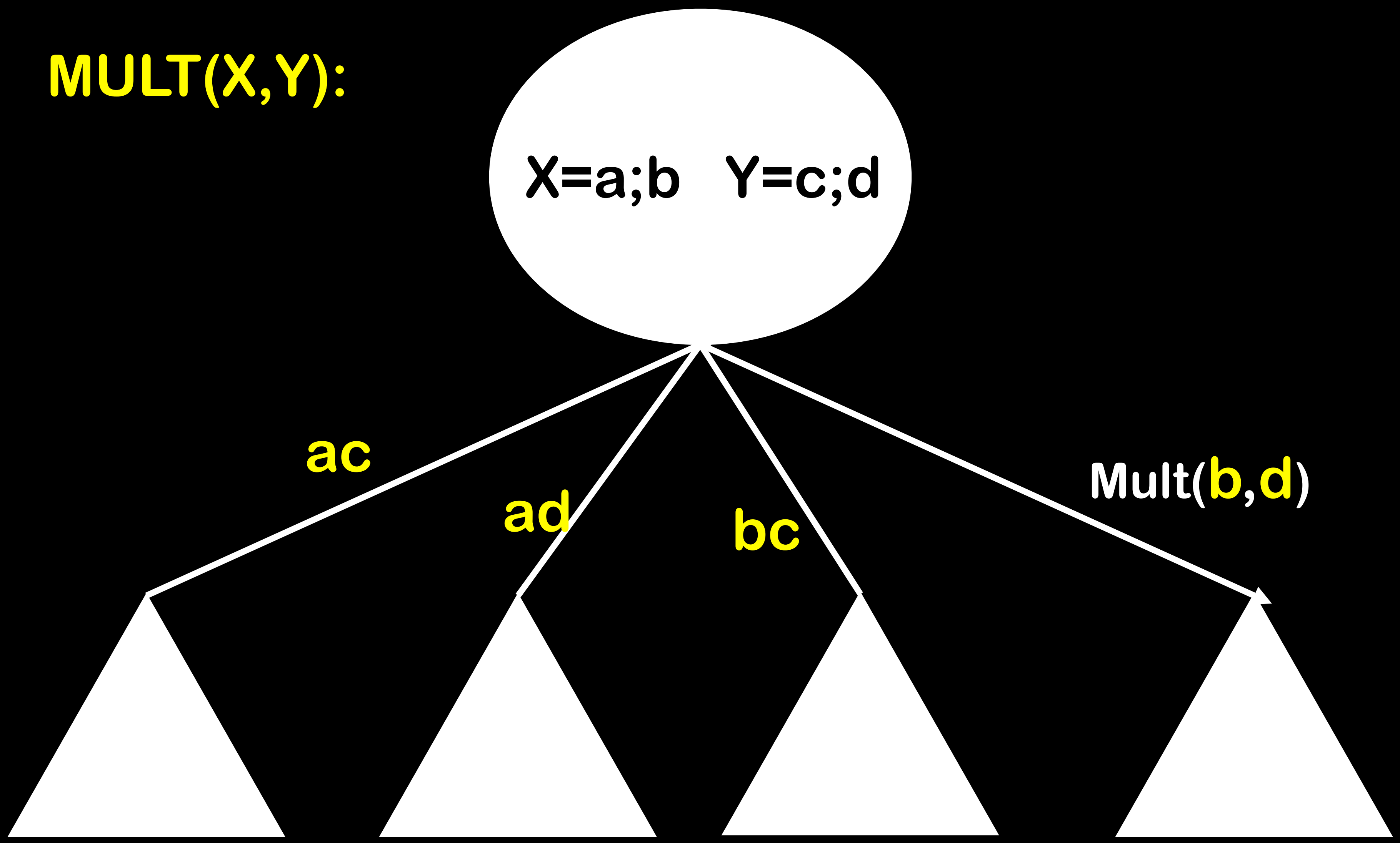
Divide, Conquer, and Glue

MULT(X,Y):



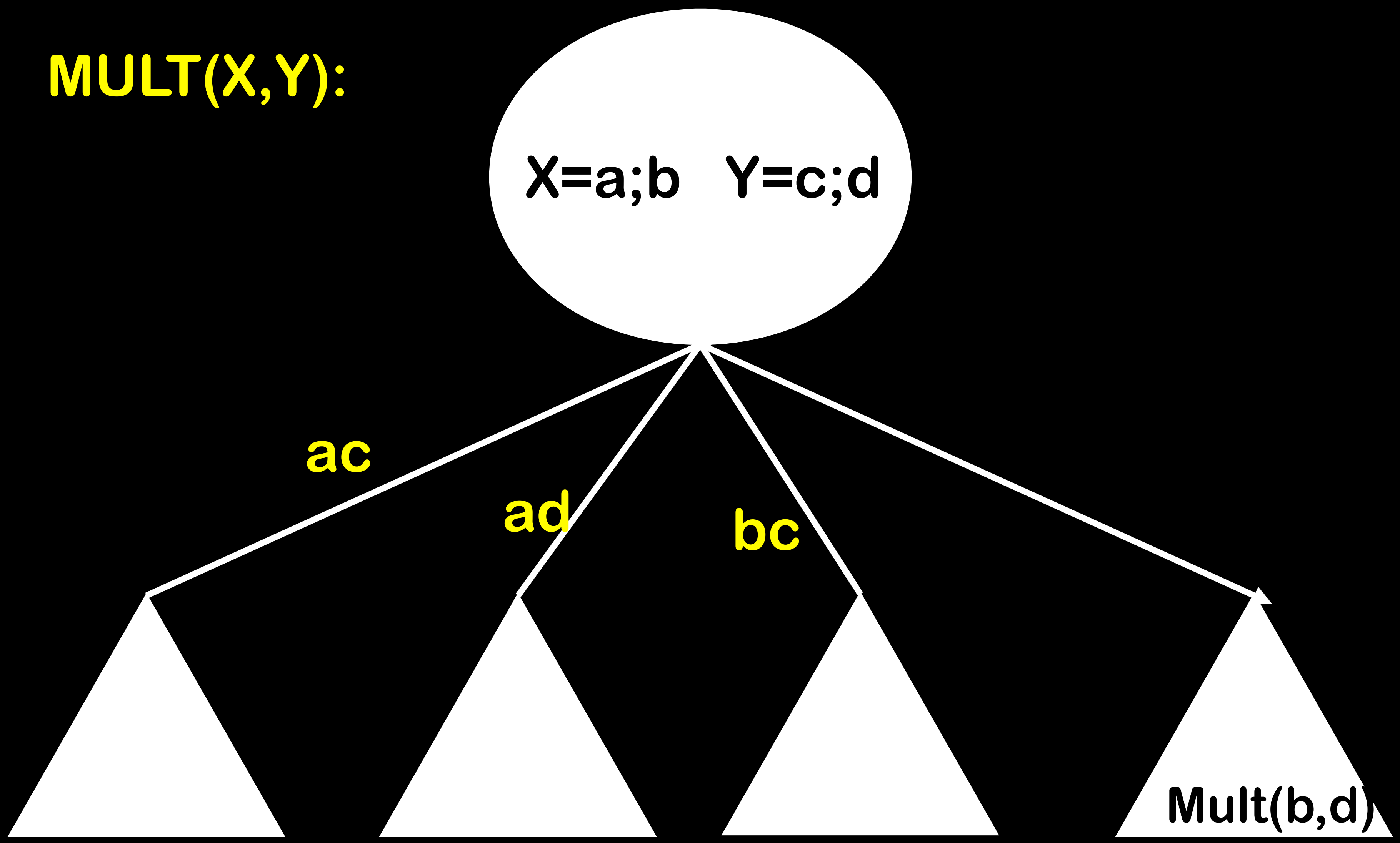
Divide, Conquer, and Glue

MULT(X,Y):



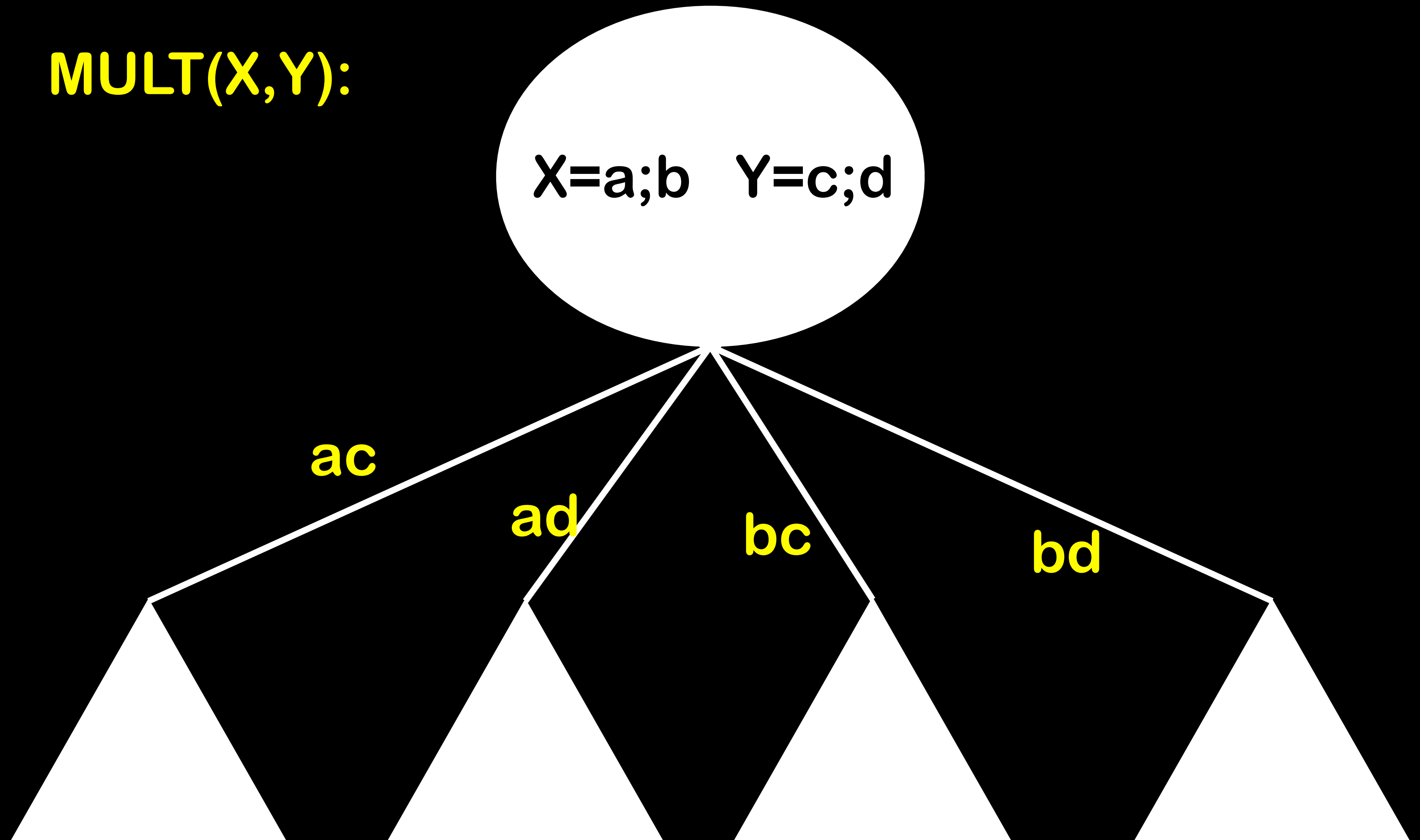
Divide, Conquer, and Glue

MULT(X,Y):



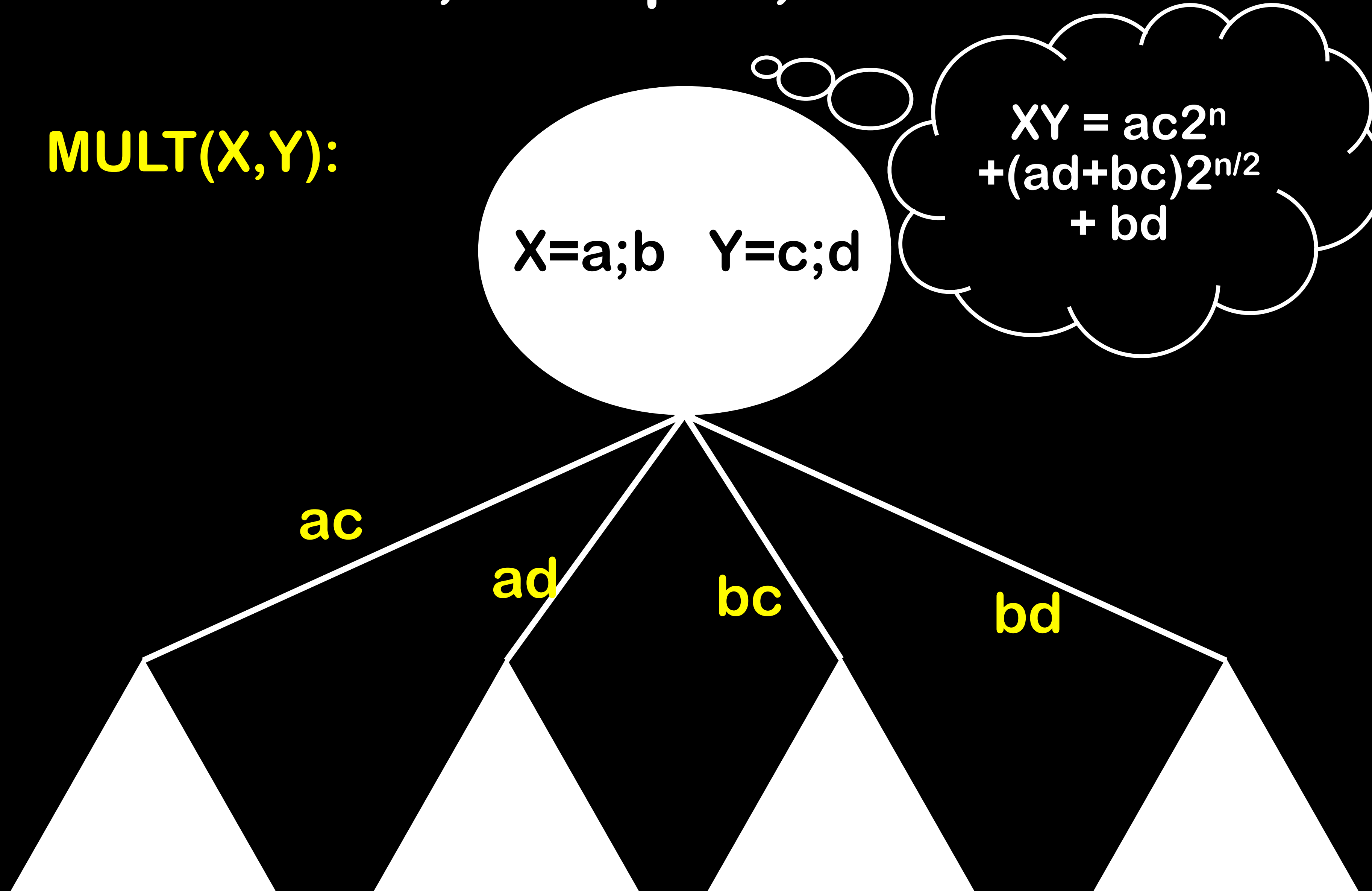
Divide, Conquer, and Glue

MULT(X,Y):



Divide, Conquer, and Glue

MULT(X,Y):



Time required by MULT

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

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 $\Theta(n^2)$?

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 $\Theta(n^2)$?

$$T(n) = 4\,T(n/2)$$

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 $\Theta(n^2)$?

$$T(n) = 4\,T(n/2) + (k'n + k'')$$

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 $\Theta(n^2)$?

$$T(n) = 4\, T(n/2) + (k'n + k'')$$

conquering time

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 $\Theta(n^2)$?

$$T(n) = 4\,T(n/2) + (k'n + k'')$$

conquering time

divide and glue

Recurrence Relation

T(1) = k   for some constant k

T(n) = 4 T(n/2) + k'n + k'' for constants k' and k''

```
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)
```

Recurrence Relation

$T(1) = 1$

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

```
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)
```

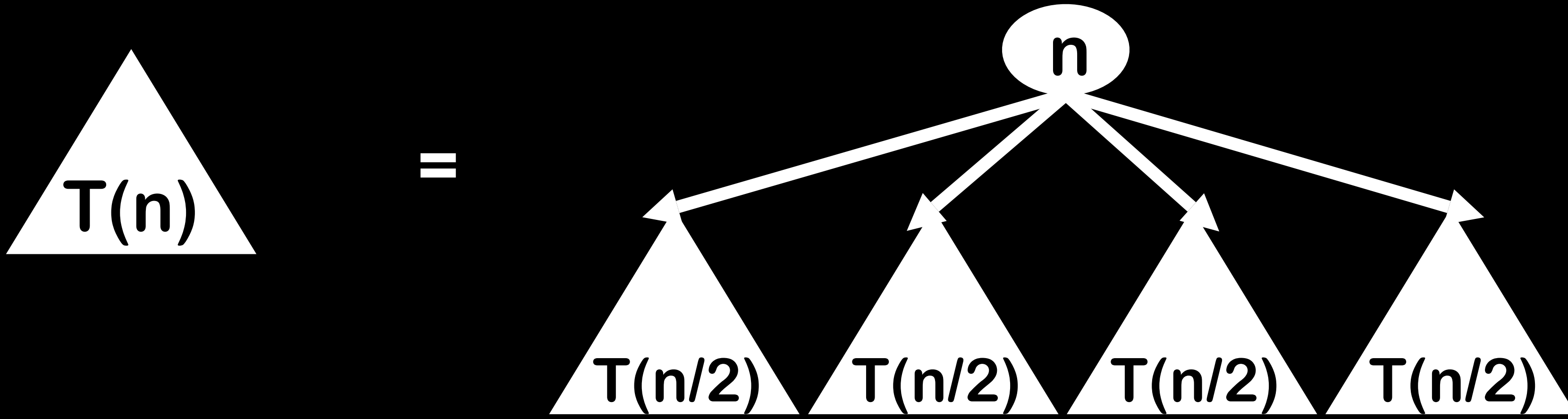
Technique: Labeled Tree Representation

Technique: Labeled Tree Representation

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

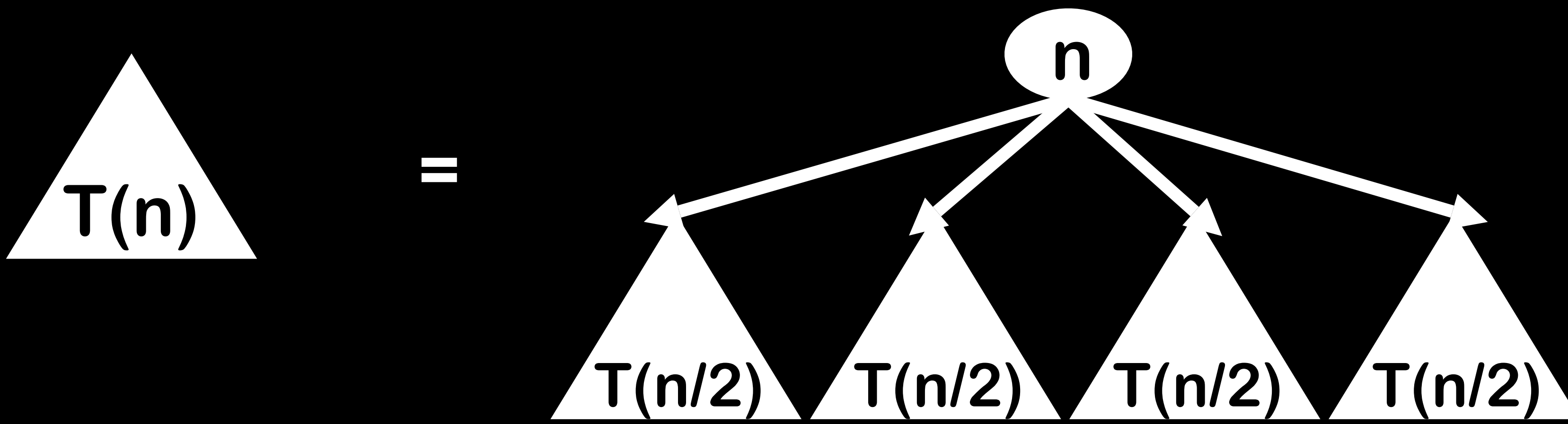
Technique: Labeled Tree Representation

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



Technique: Labeled Tree Representation

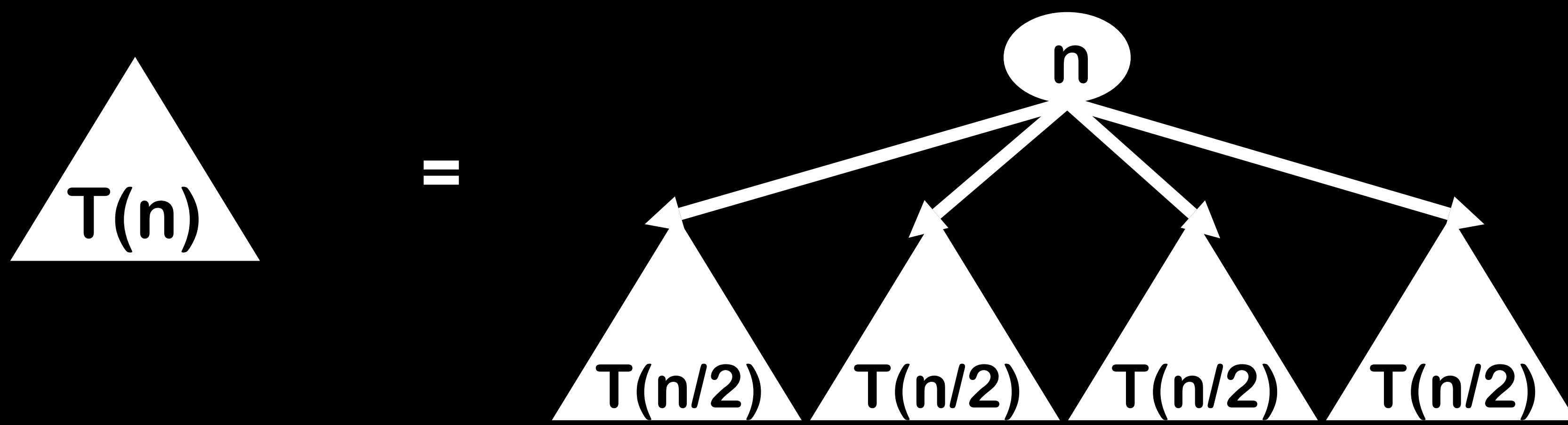
$$T(n) = n + 4\,T(n/2)$$



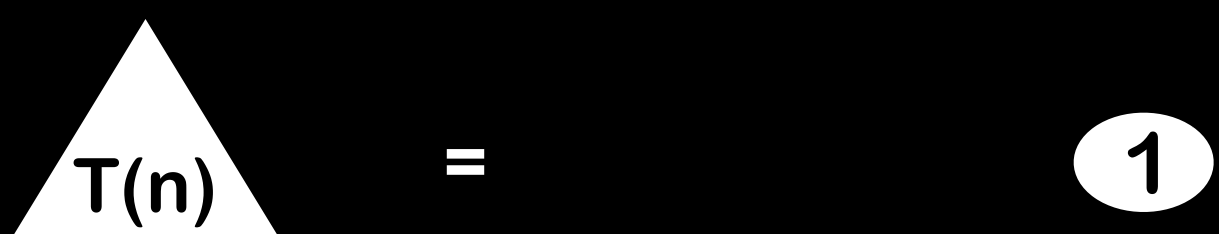
$$T(1) = 1$$

Technique: Labeled Tree Representation

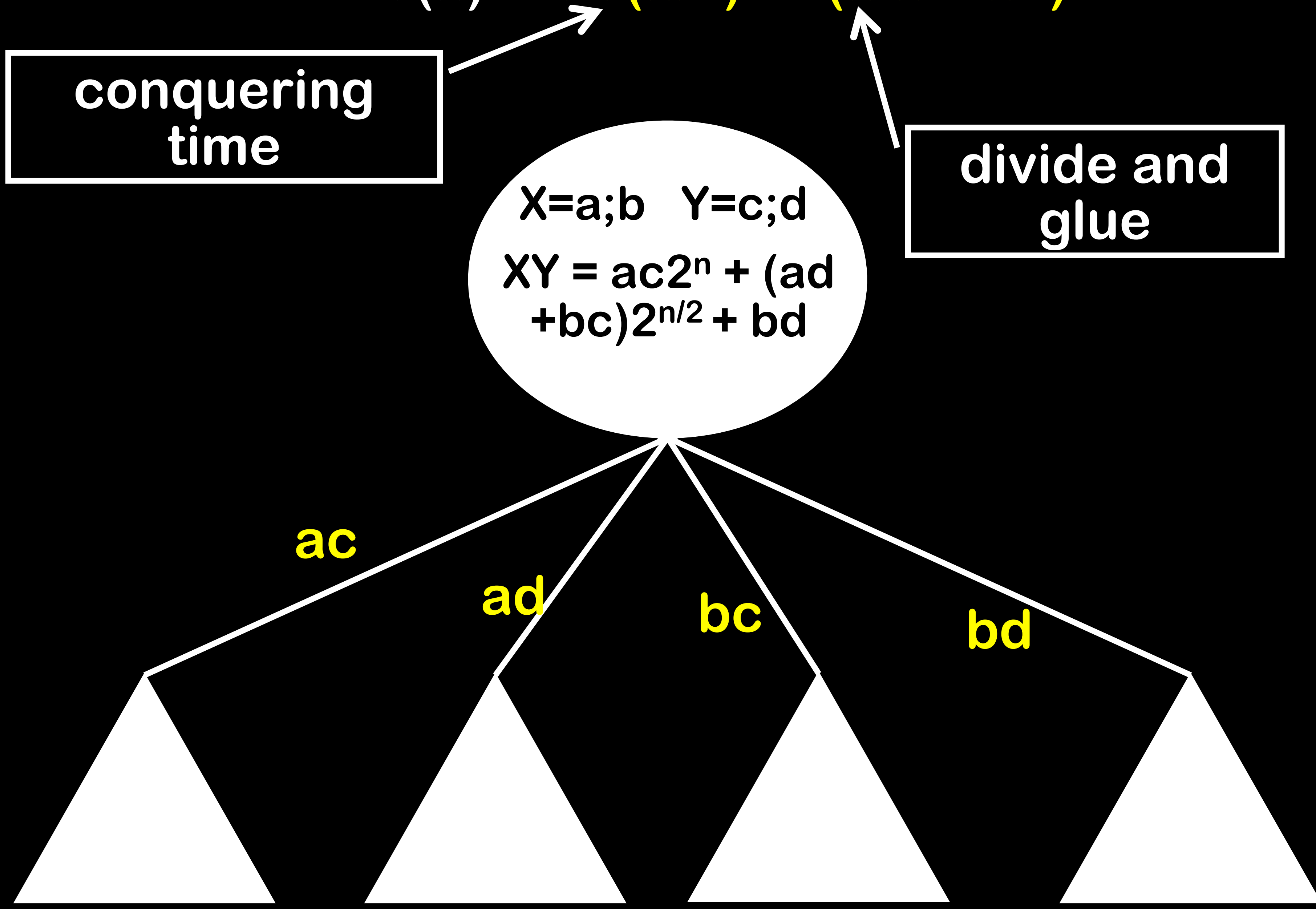
T(n)	=	n + 4 T(n/2)



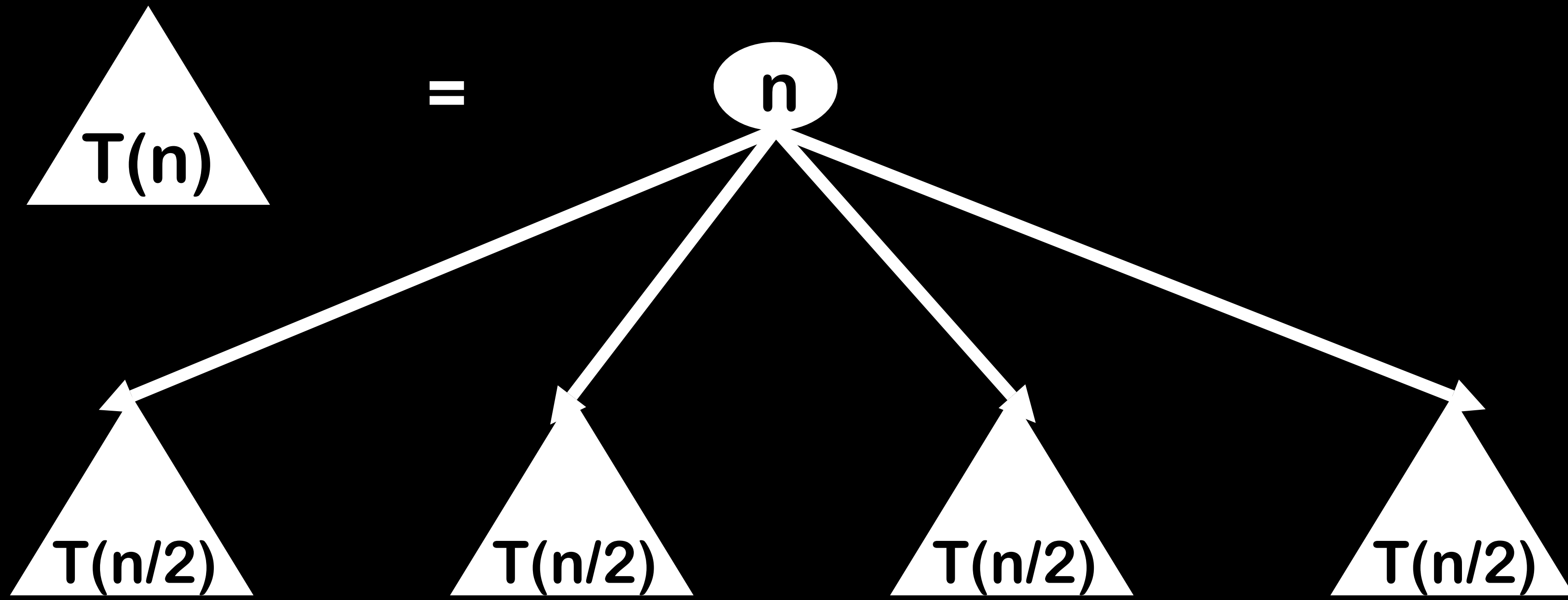
T(1)	=	1



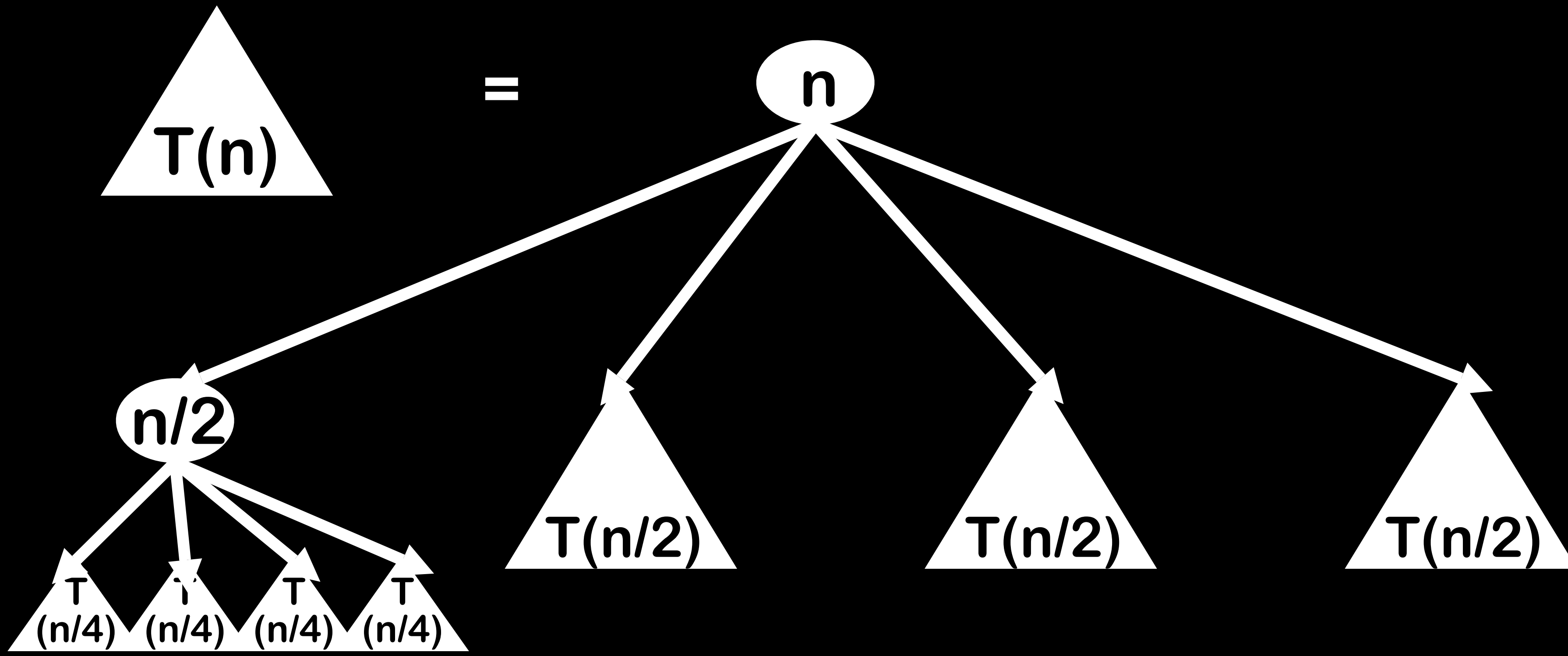
$$T(n) = 4\,T(n/2) + (k'n + k'')$$



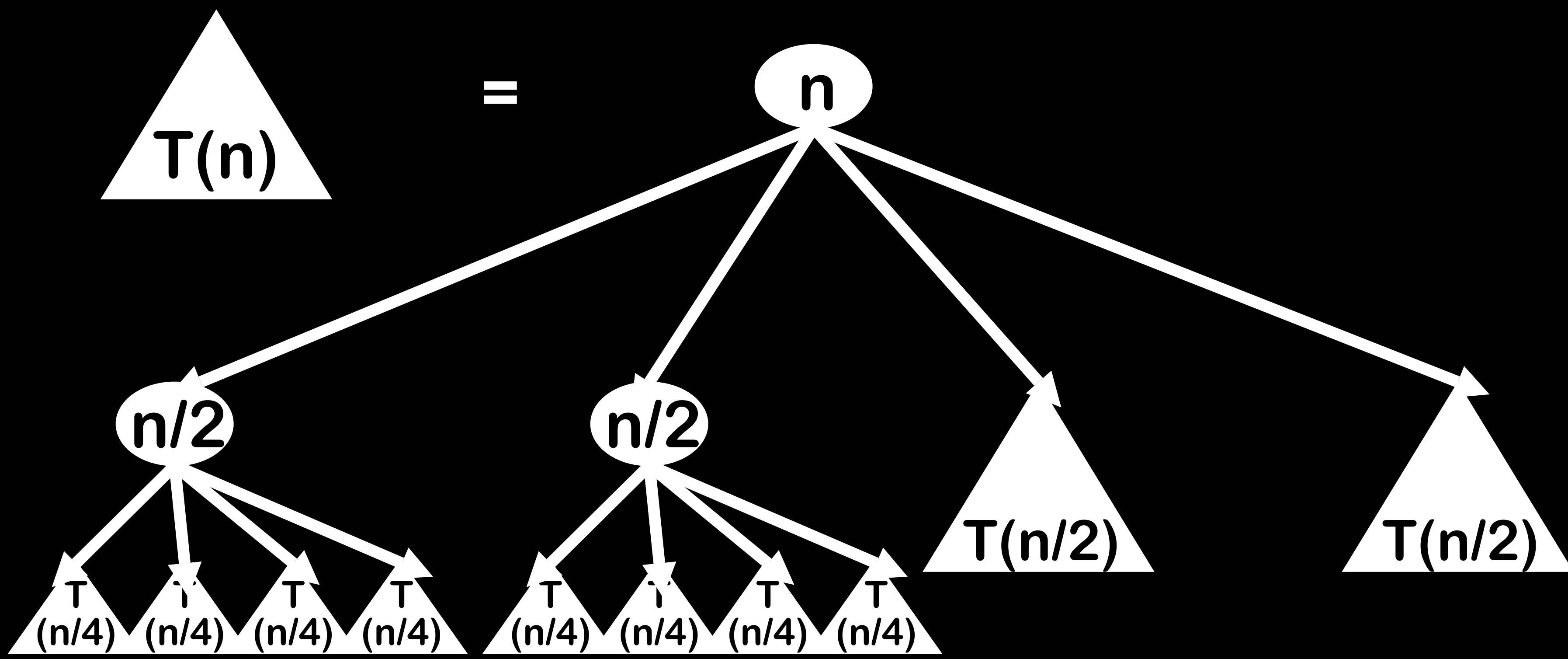
conquering time
divide and glue
X=a;b Y=c;d
$XY = ac2^n + (ad+bc)2^{n/2} + bd$
ac
ad
bc
bd



T(n)
=
n
T(n/2)
T(n/2)
T(n/2)
T(n/2)



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


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

0	n
1	n/2 + n/2 + n/2 + n/2
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

0	n
1	n/2 + n/2 + n/2 + n/2
2	
i	
	. . . .
$\log_2(n)$	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

0	n
1	n/2 + n/2 + n/2 + n/2
2	
i	Level i is the sum of 4^i copies of $n/2^i$
	. . . .
$\log_2(n)$	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

n/2 + n/2 + n/2 + n/2

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
n/2 + n/2 + n/2 + n/2
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
	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 =	
	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
$2n =$	$n/2 \quad + \quad n/2 \quad + \quad n/2 \quad + \quad n/2$
$4n =$	
$2^i n =$	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
$2n =$	$n/2 \quad + \quad n/2 \quad + \quad n/2 \quad + \quad n/2$
$4n =$	
$2^i n =$	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

1n =	n
2n =	n/2 + n/2 + n/2 + n/2
4n =	
2^in =	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

MULT revisited

```
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)
```

MULT revisited

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)

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

Gauss' optimization

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	$X_7 = X_3 - X_4 - X_5$	$= bc + ad$

Karatsuba, Anatolii Alexeevich (1937-)

Sometime in the late 1950's 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
```

$$e\,2^n + (\text{MULT}(a+b,c+d) - e - f)\,2^{n/2} + f$$

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)

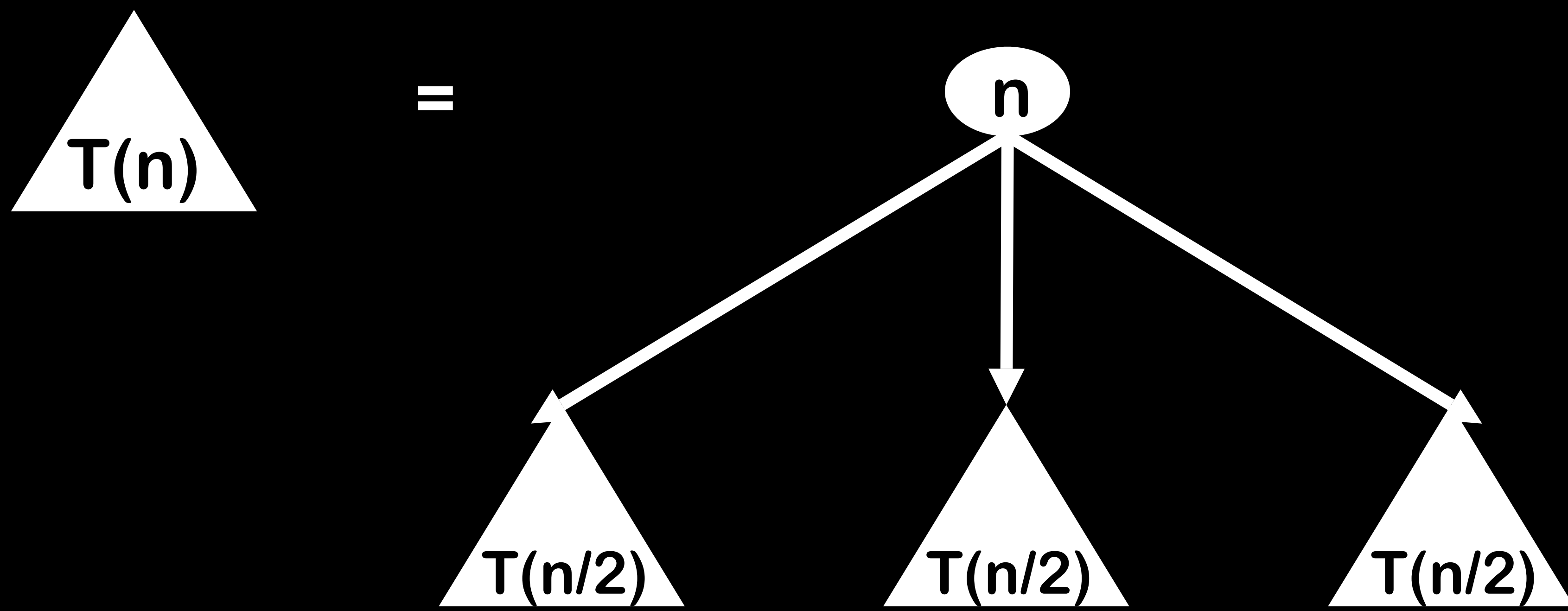
f := MULT(b,d)

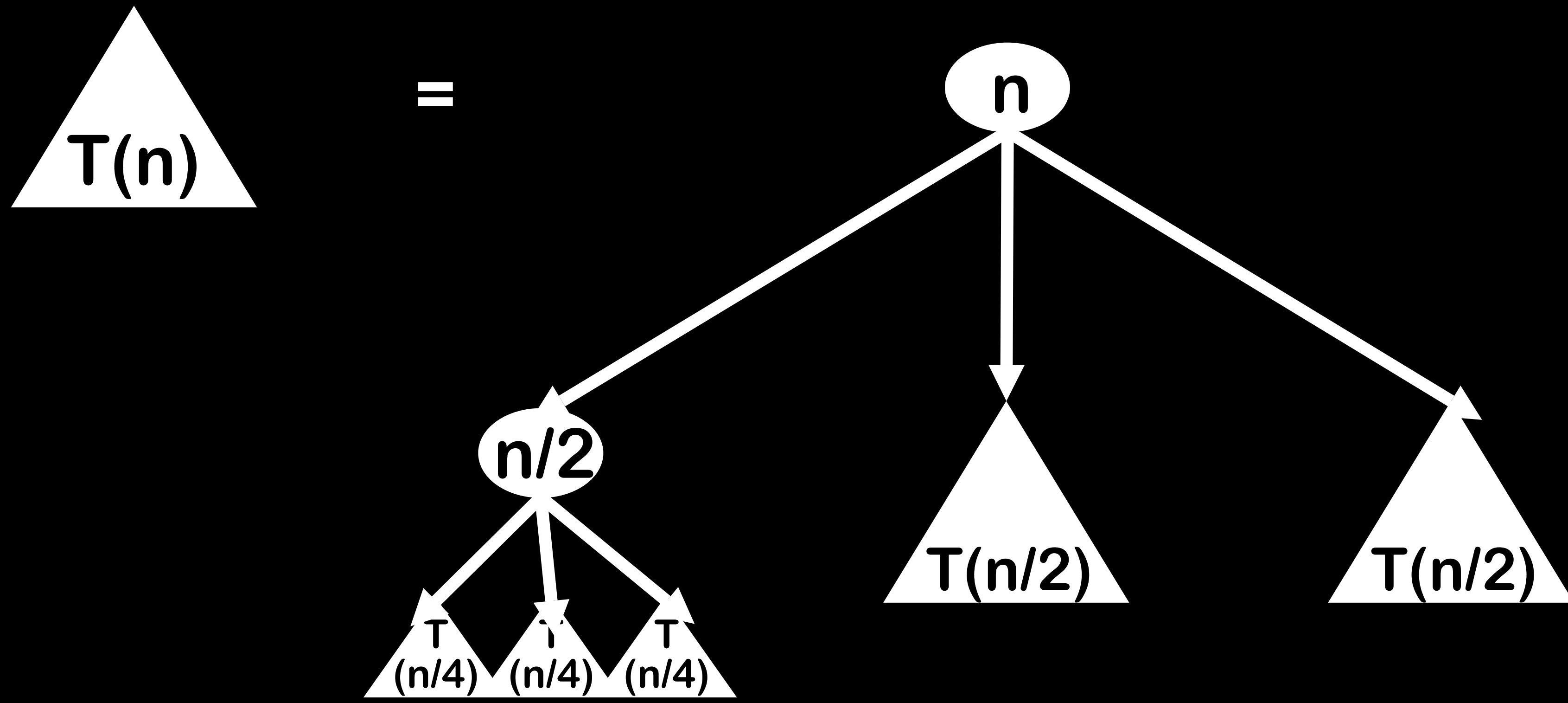
return

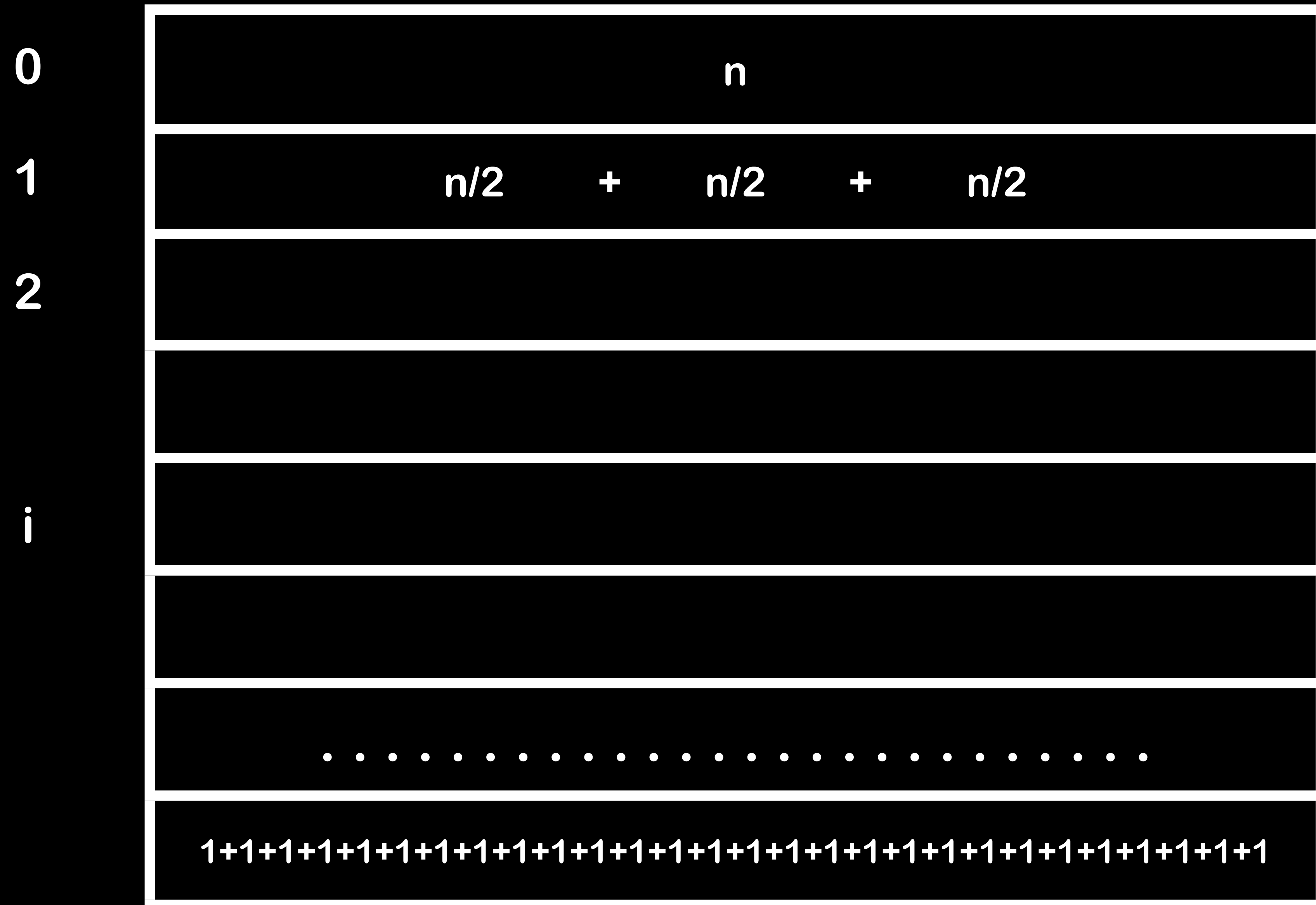
$e\ 2^n + (\text{MULT}(a+b,c+d) - e - f)\ 2^{n/2} + f$

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

Actually: $T(n) = 2\ T(n/2) + T(n/2 + 1) + kn$


T(n)
=
n
T(n/2)
T(n/2)
T(n/2)


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


0
n
1
n/2 + n/2 + n/2
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

0	n
1	n/2 + n/2 + n/2
2	
i	
	. . . .
$\log_2(n)$	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

0	n
1	n/2 + n/2 + n/2
2	
i	Level i is the sum of 3^i copies of $n/2^i$
	. . . .
$\log_2(n)$	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

n/2 + n/2 + n/2

Level i is the sum of 3^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
$n/2 \quad + \quad n/2 \quad + \quad n/2$
Level i is the sum of 3^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
3/2n =	n/2 + n/2 + n/2
	Level i is the sum of 3^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
3/2n =	n/2 + n/2 + n/2
9/4n =	
	Level i is the sum of 3^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
3/2n =	n/2 + n/2 + n/2
9/4n =	
$(3/2)^i n$ =	Level i is the sum of 3^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
$3/2n =$	n/2 + n/2 + 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

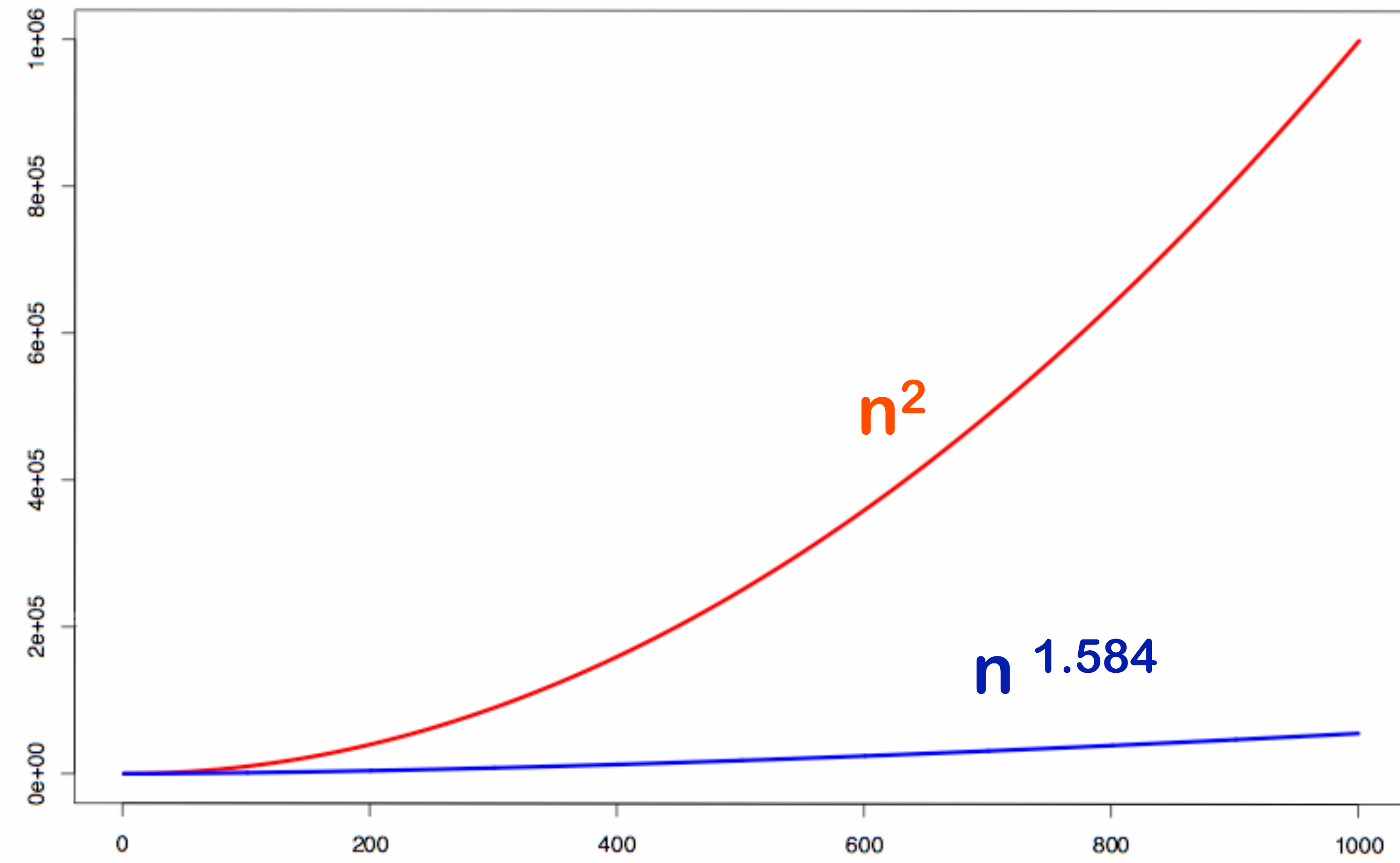
$1n =$	n
$3/2n =$	$n/2 \quad + \quad n/2 \quad + \quad n/2$
$9/4n =$	
$(3/2)^i n =$	Level i is the sum of 3^i copies of $n/2^i$
	. . . .
$(3/2)^{\log n} n =$	1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1

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

Dramatic Improvement for Large n

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

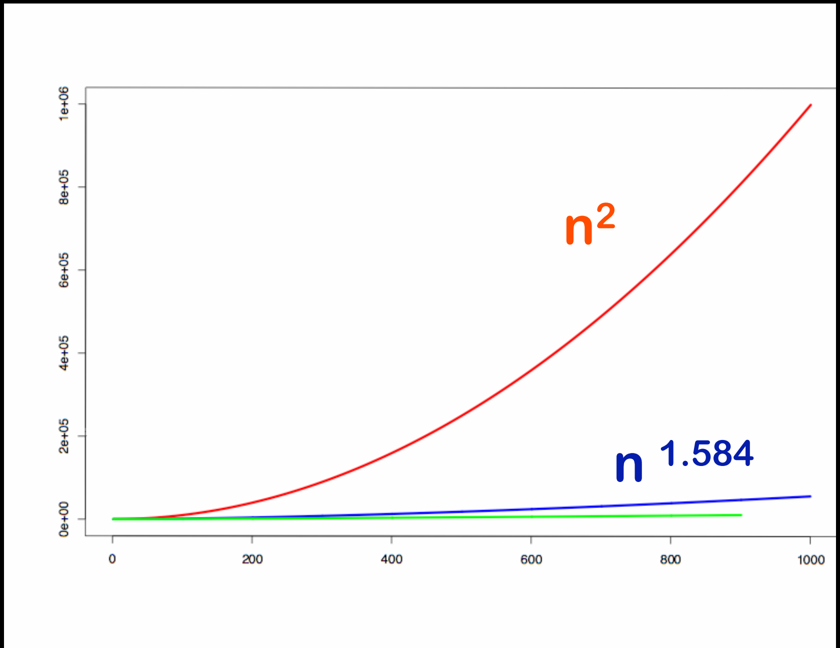
A huge savings over $\Theta(n^2)$ when n gets large.



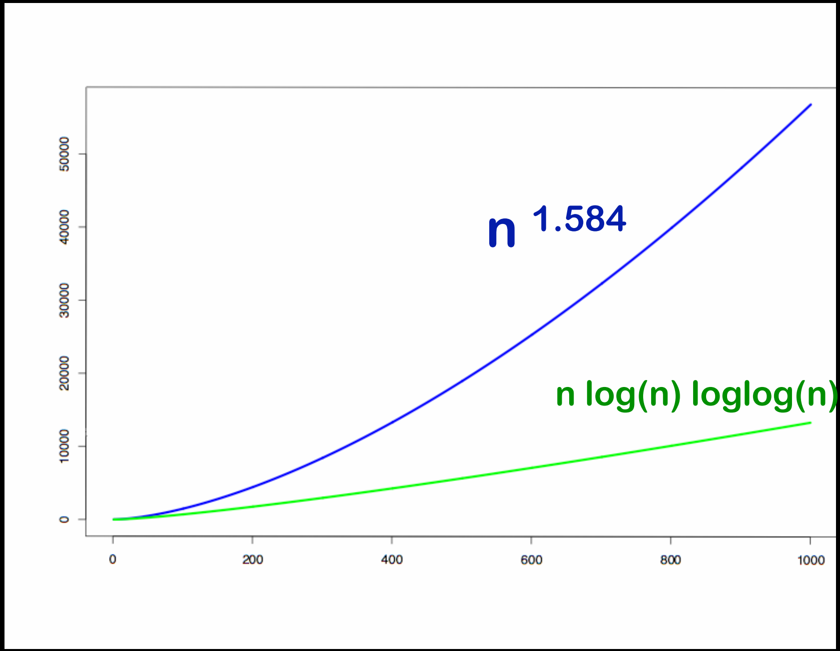
1e+06
8e+05
6e+05
4e+05
2e+05
0e+00
0
200
400
600
800
1000
n2
n 1.584

Multiplication Algorithms

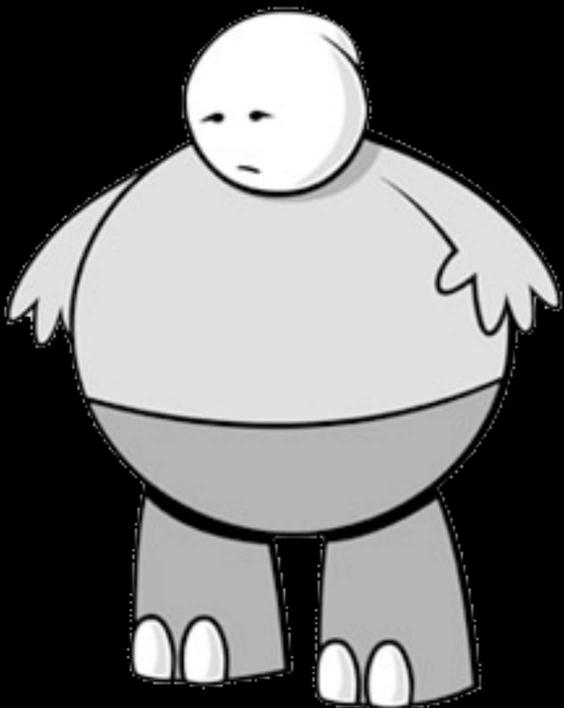
Kindergarten	$n2^n$
Grade School	n^2
Karatsuba	$n^{1.58...}$
Fastest Known	n logn loglogn



n2
n 1.584
0e+00
2e+05
4e+05
6e+05
8e+05
1e+06
0
200
400
600
800
1000



n 1.584
n log(n) loglog(n)
0
10000
20000
30000
40000
50000
0
200
400
600
800
1000



Here's What You Need to Know…

- Gauss's Multiplication Trick
- Proof of Lower bound for addition
- Divide and Conquer
- Solving Recurrences
- Karatsuba Multiplication