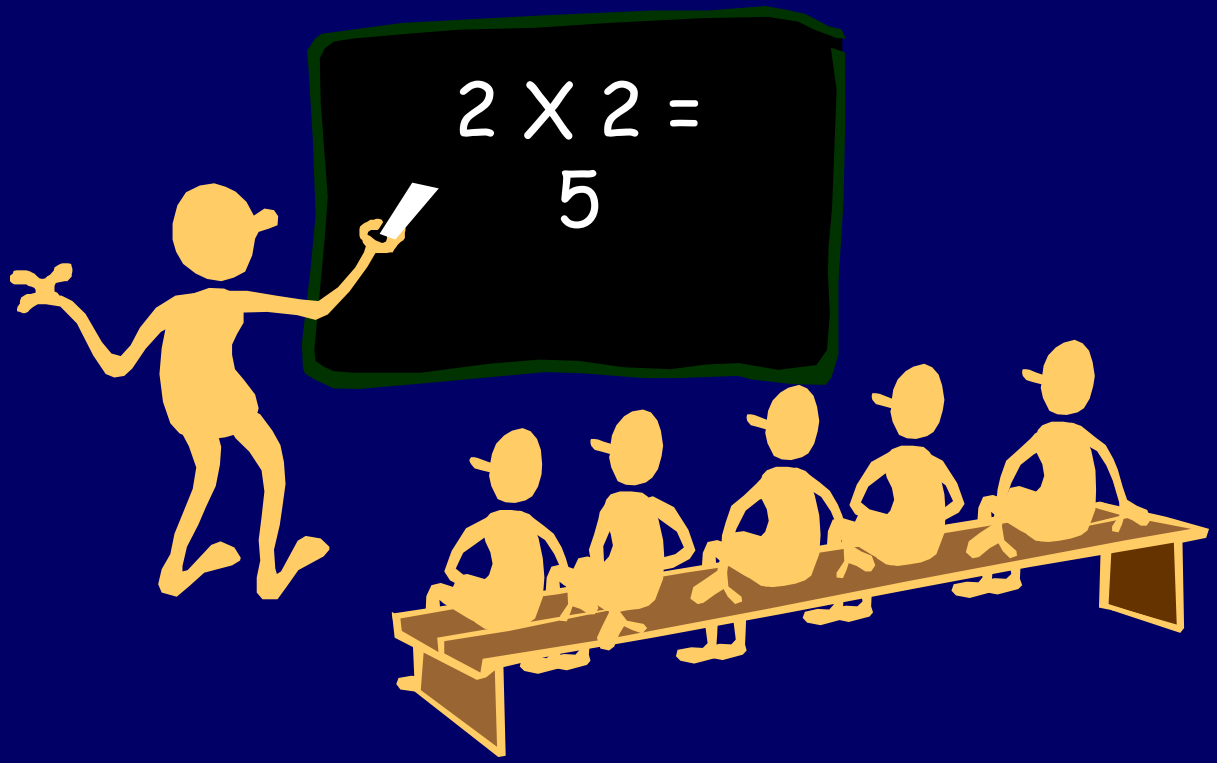
Great Theoretical Ideas In Computer Science

John Lafferty CS 15-251 Fall 2006

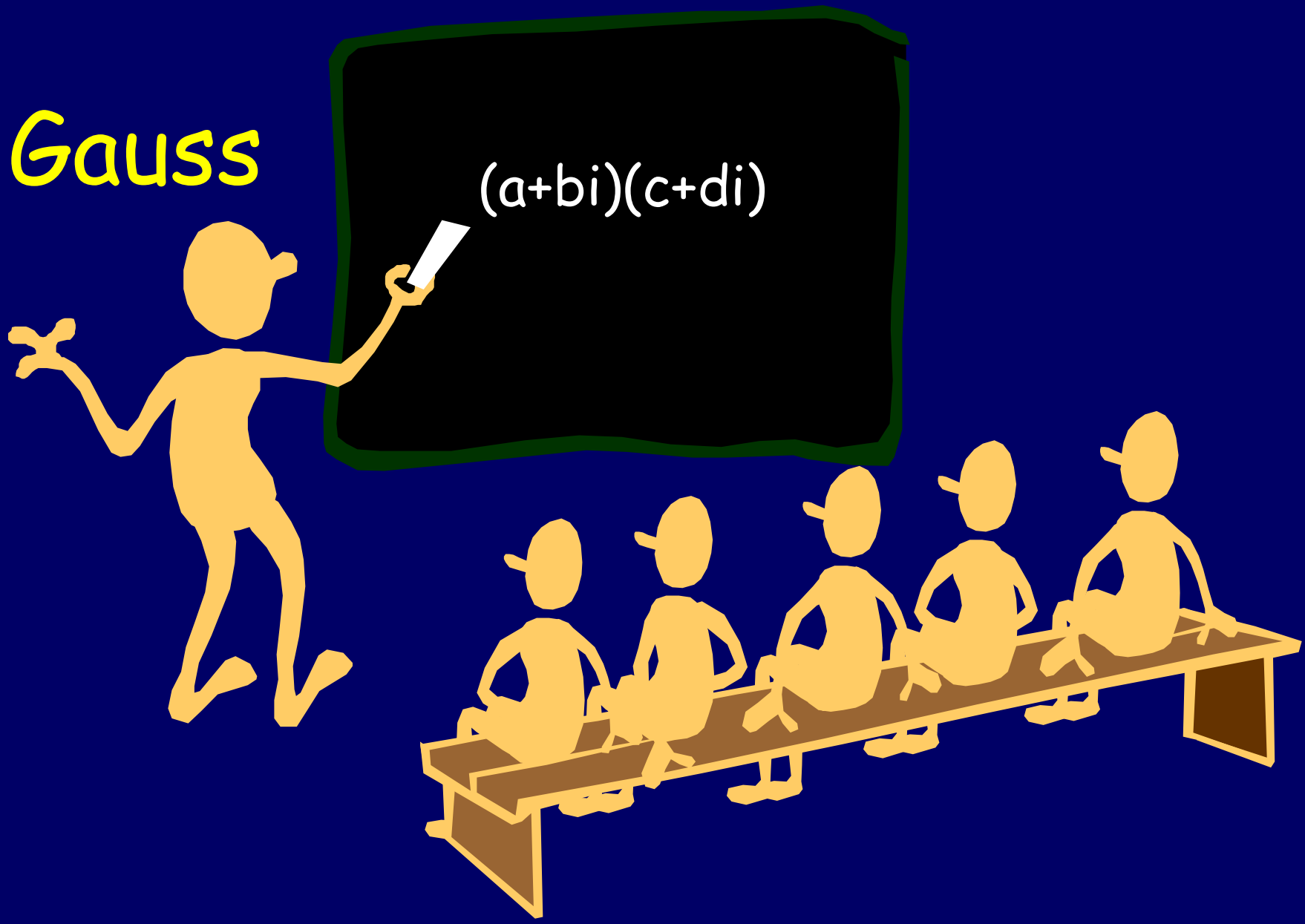
Lecture 24 November 16, 2006 Carnegie Mellon University

Grade School Revisited: How To Multiply Two Numbers




The best way is often far from obvious!

Gauss



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

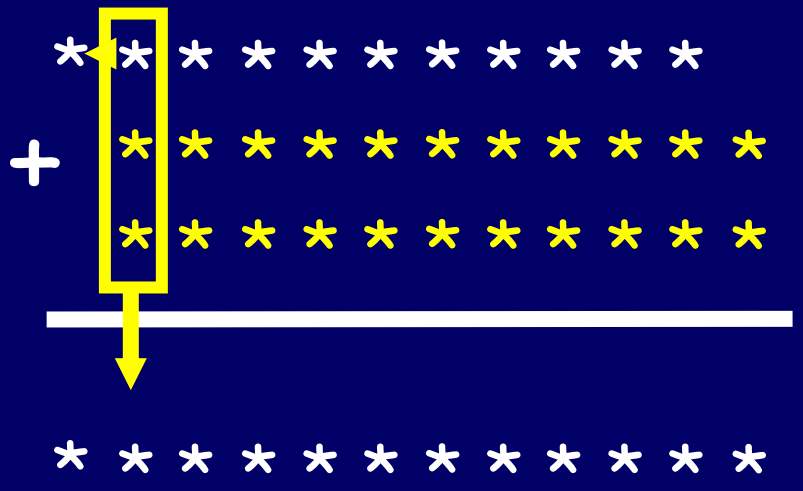
(a+bi)(c+di) =
[ac –bd] + [ad + bc] i

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

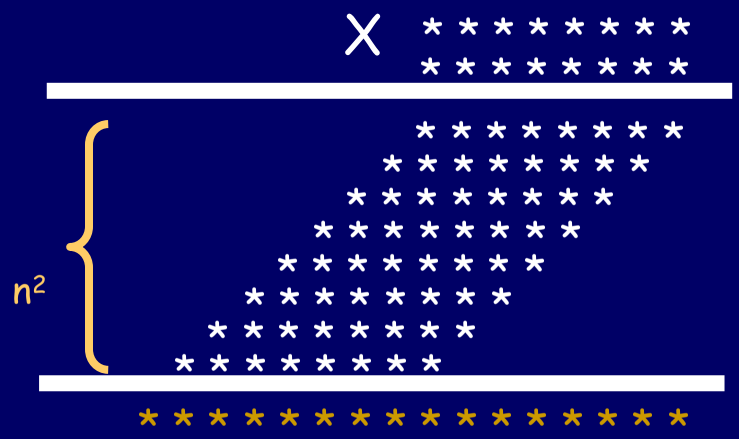


T(n) = The amount of time grade school addition uses to add two n-bit numbers

We saw that T(n) was linear.

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

Time complexity of grade school multiplication

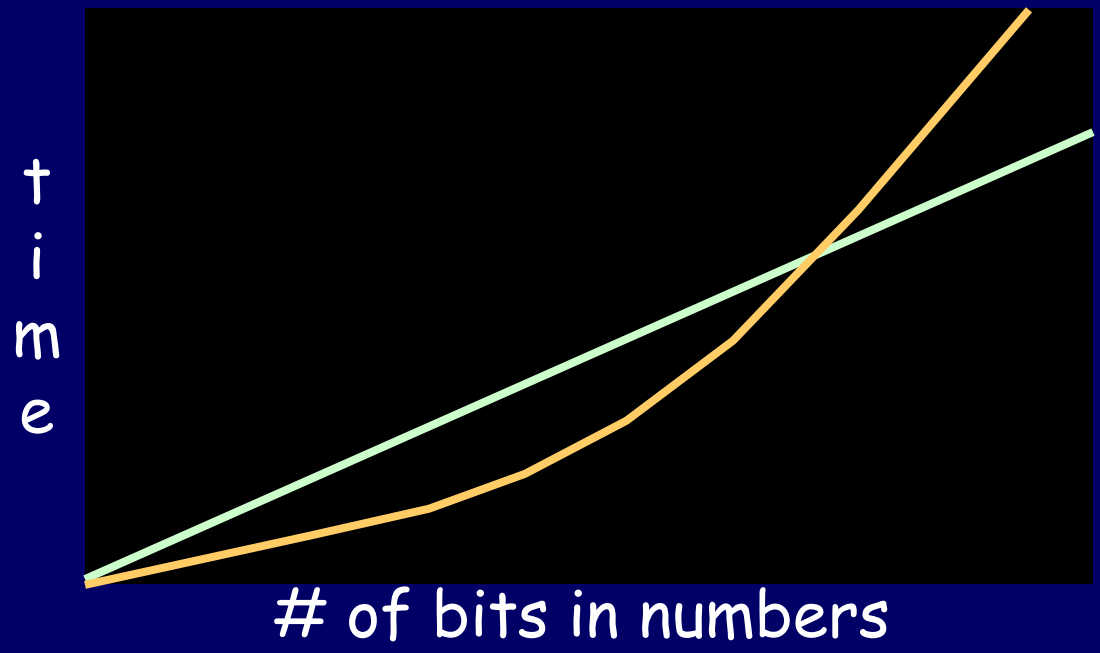


T(n) = The amount of time grade school multiplication uses to add two n-bit numbers

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

Grade school addition is linear time.

Is there a sub-linear time method for addition?

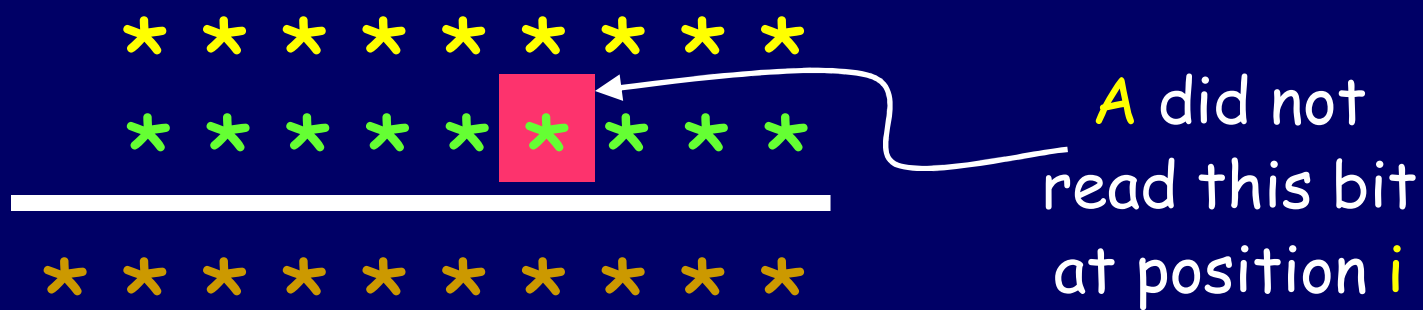
Any addition algorithm takes $\Omega(n)$ time

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

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

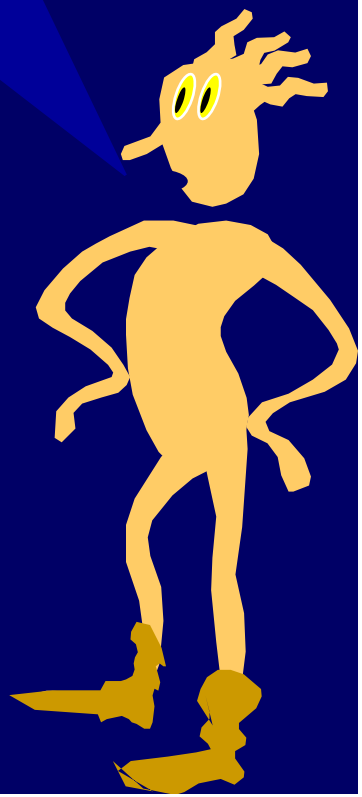
Any addition algorithm takes $\Omega(n)$ time



- 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.

So any algorithm for addition must use time at least linear in the size of the numbers.

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

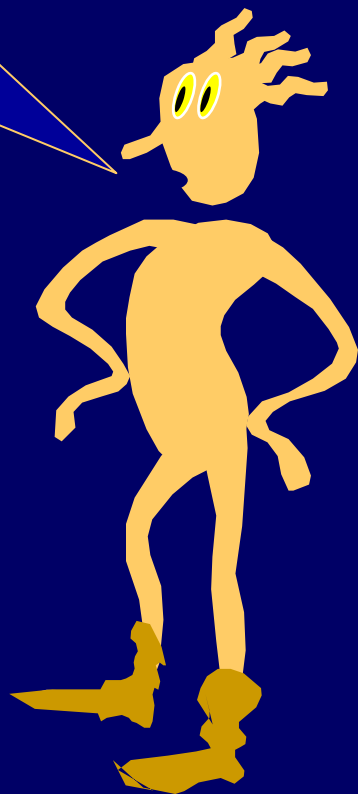


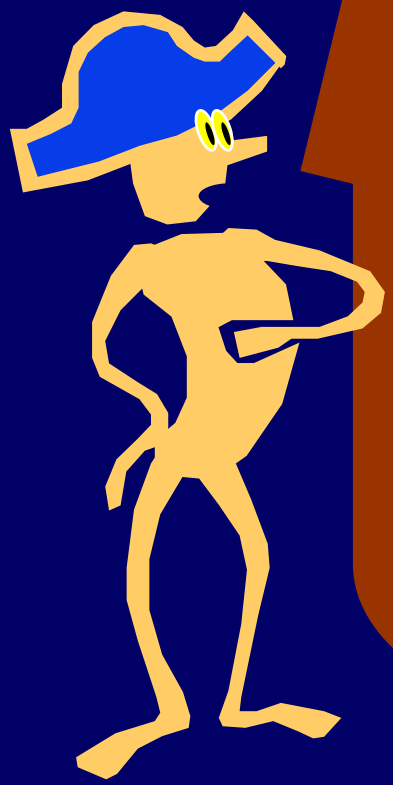
Grade School Addition: $\Theta(n)$ time
Furthermore, it is optimal

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!

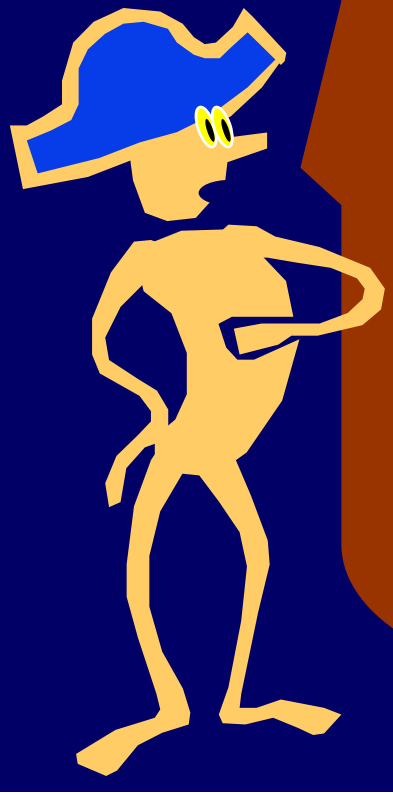




Can we even break the quadratic time barrier?

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

Grade School Multiplication: The Kissing Intuition



Intuition:
Let's say that each time an algorithm has to multiply a digit from one number with a digit from the other number, we call that a "kiss".

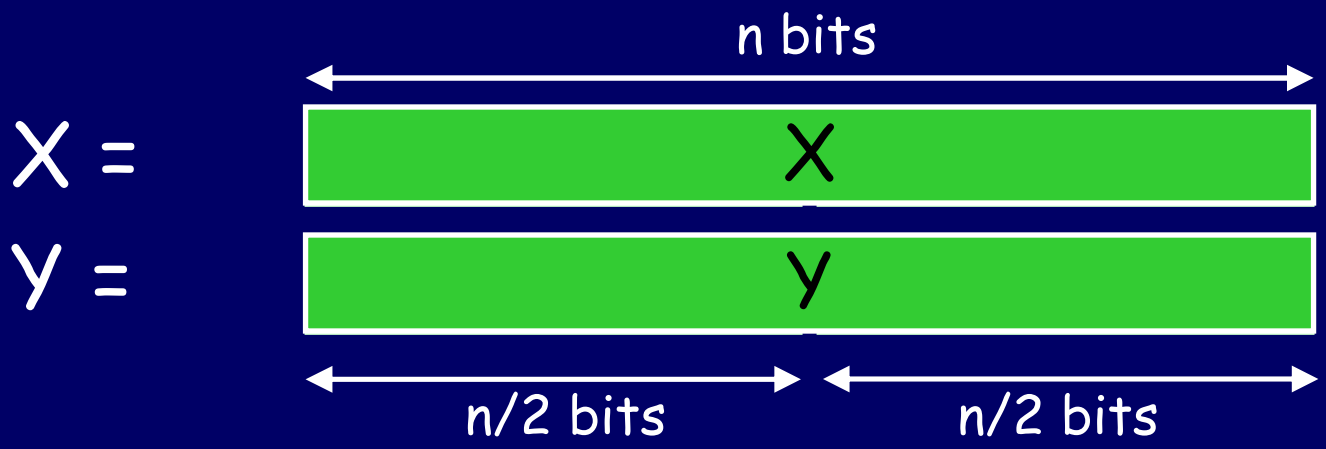
It seems as if any correct algorithm must kiss at least n^2 times.

Divide And Conquer

An approach to faster algorithms:

1. DIVIDE a problem into smaller subproblems
2. CONQUER them recursively
3. GLUE the answers together so as to obtain the answer to the larger problem

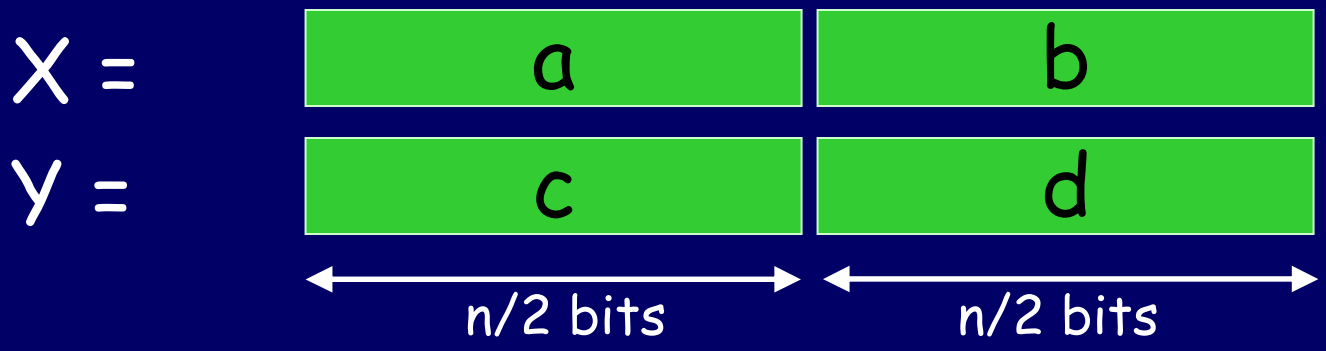
Multiplication of 2 n-bit numbers



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

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

Multiplication of 2 n-bit numbers



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

MULT(X,Y):

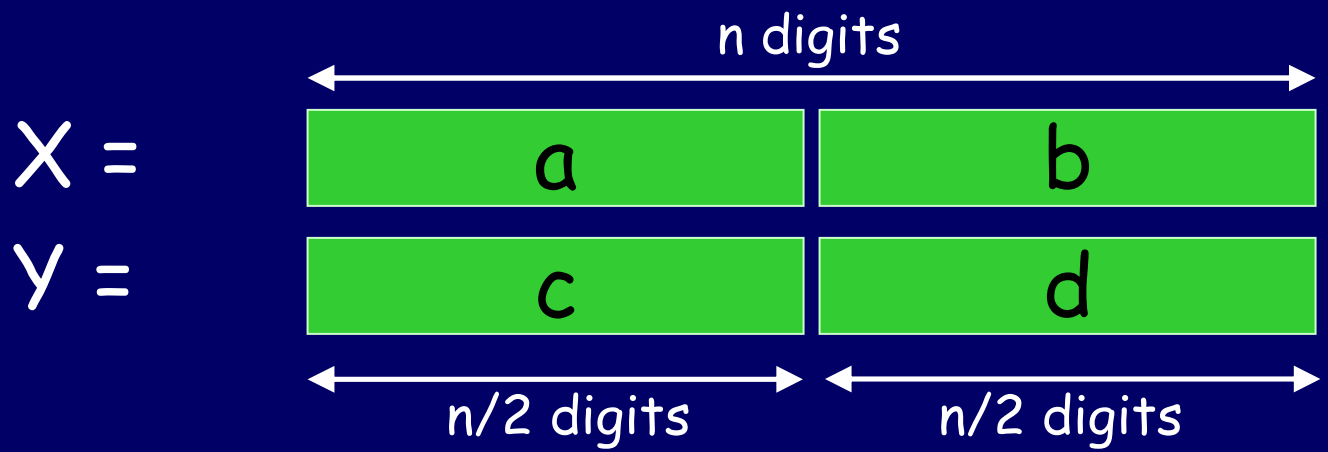
If |X| = |Y| = 1 then return XY

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!



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

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

Multiplying (Divide & Conquer style)

12345678 * 21394276

X = | a | b |

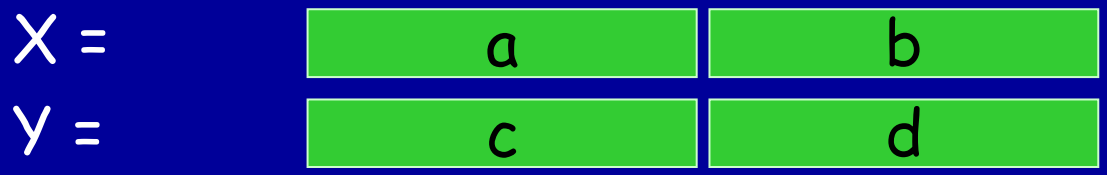
Y = | c | d |

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

Multiplying (Divide & Conquer style)

12345678 * 21394276

5678*4276

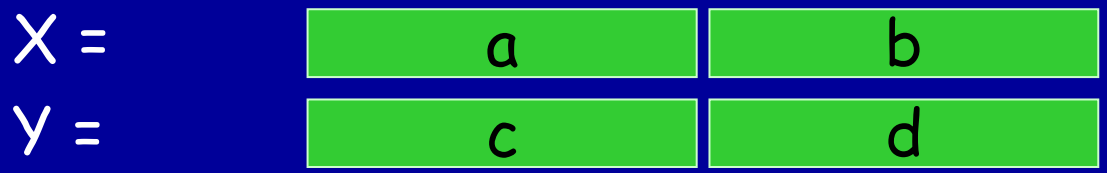


$X \times Y = ac\ 10^{n} + (ad + bc)\ 10^{n/2} + bd$

Multiplying (Divide & Conquer style)

12345678 * 21394276

5678*2139 5678*4276

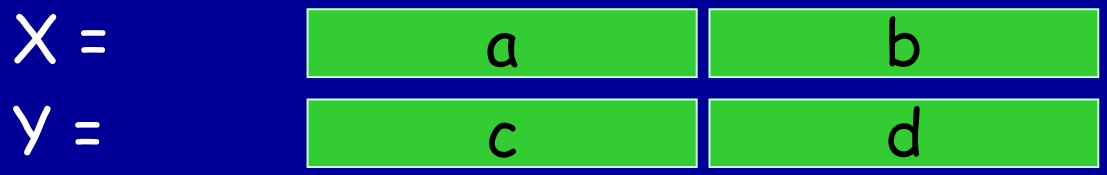


$X \times Y = ac\ 10^{n} + (ad + bc)\ 10^{n/2} + bd$

Multiplying (Divide & Conquer style)

12345678 * 21394276

1234*4276 5678*2139 5678*4276

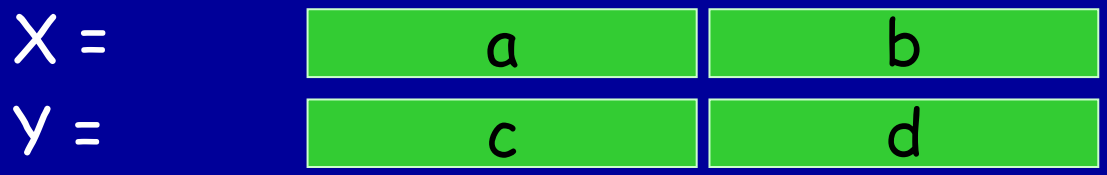


$X \times Y = ac\ 10^{n} + (ad + bc)\ 10^{n/2} + bd$

Multiplying (Divide & Conquer style)

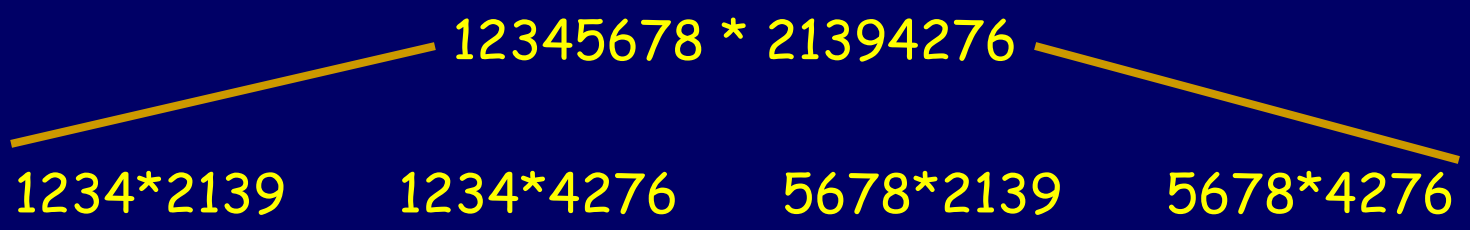
12345678 * 21394276

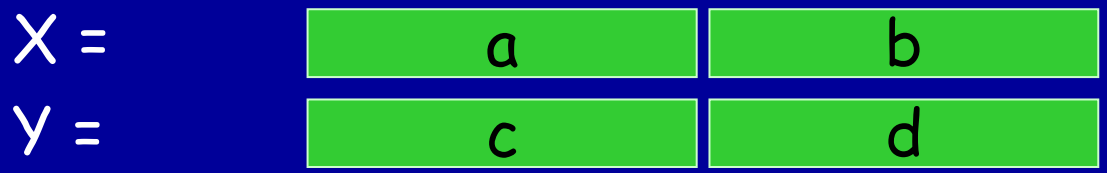
1234*2139	1234*4276	5678*2139	5678*4276



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

Multiplying (Divide & Conquer style)





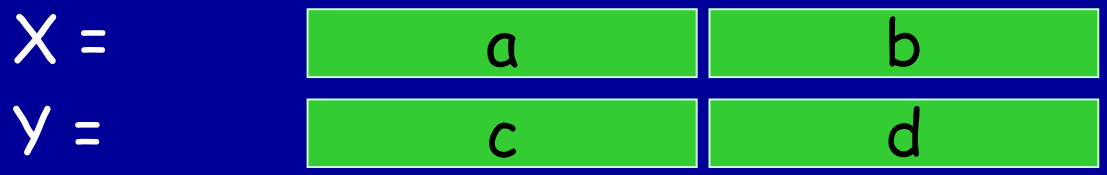
$$X \times Y = ac\ 10^{n} + (ad + bc)\ 10^{n/2} + bd$$

Multiplying (Divide & Conquer style)

12345678 * 21394276

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

34*39



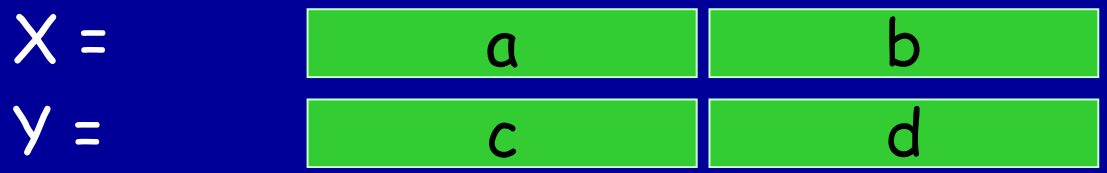
$X \times Y = ac\ 10^{n} + (ad + bc)\ 10^{n/2} + bd$

Multiplying (Divide & Conquer style)

12345678 * 21394276

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

34*21 34*39



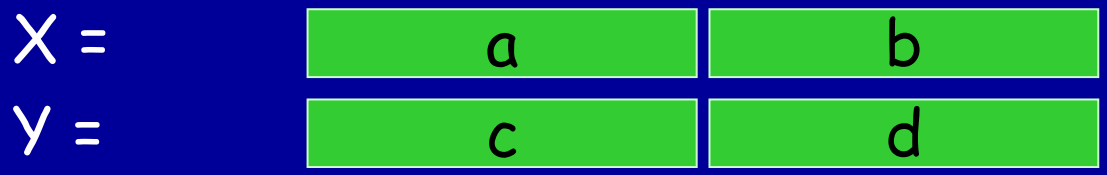
$X \times Y = ac\ 10^{n} + (ad + bc)\ 10^{n/2} + bd$

Multiplying (Divide & Conquer style)

12345678 * 21394276

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

12*39 34*21 34*39



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

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

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

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

252 12*39 34*21 34*39

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

252 12*39 34*21 34*39

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

252 12*39 34*21 34*39

1*3 1*9 2*3 2*9

X = a b

Y = c d

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

Multiplying (Divide & Conquer style)

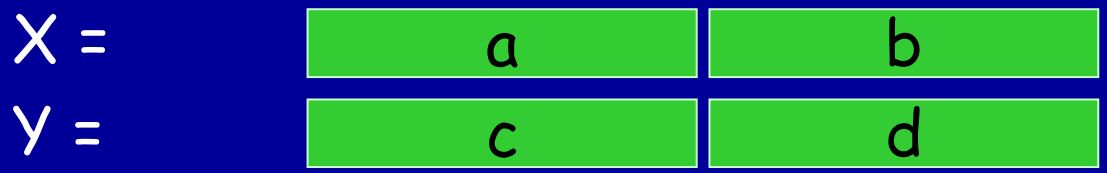
12345678 * 21394276

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

242 12*39 34*21 34*39

3 9 6 18

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



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

Multiplying (Divide & Conquer style)

12345678 * 21394276

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

242 468 34*21 34*39

3 9 6 18

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

X = a b

Y = c d

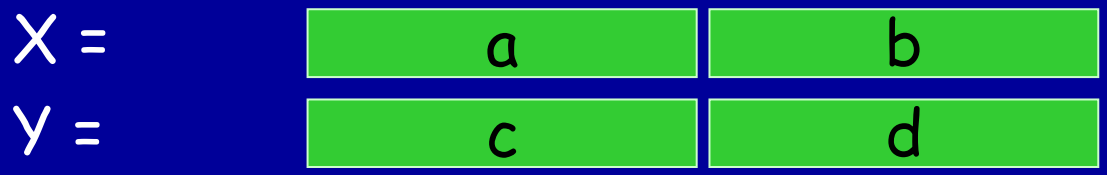
$X \times Y = ac\ 10^n + (ad + bc)\ 10^{n/2} + bd$

Multiplying (Divide & Conquer style)

12345678 * 21394276

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

252 468 714 1326



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

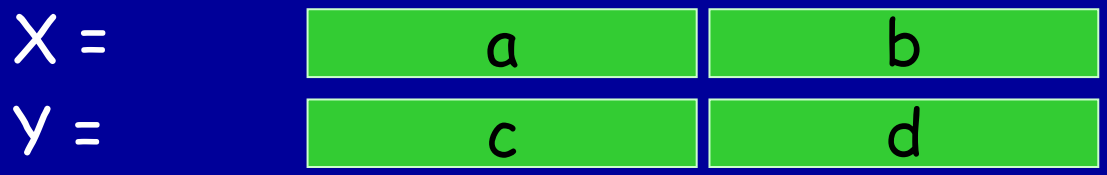
Multiplying (Divide & Conquer style)

12345678 * 21394276

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

252 468 714 1326

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



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

Multiplying (Divide & Conquer style)

12345678 * 21394276

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

252 468 714 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 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

2639526 5276584 12145242 24279128

X = a b

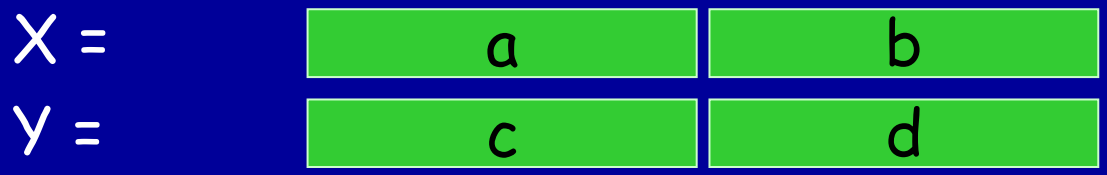
Y = c d

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

Multiplying (Divide & Conquer style)

12345678 * 21394276

2639526 *10^8 + 5276584 *10^4 + 12145242 *10^4 + 24279128 *1



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

Multiplying (Divide & Conquer style)

12345678 * 21394276

2639526 *10^8 + 5276584 *10^4 + 12145242 *10^4 + 24279128 *1

= 264126842539128

X = | a | b |

Y = | c | d |

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

Multiplying (Divide & Conquer style)

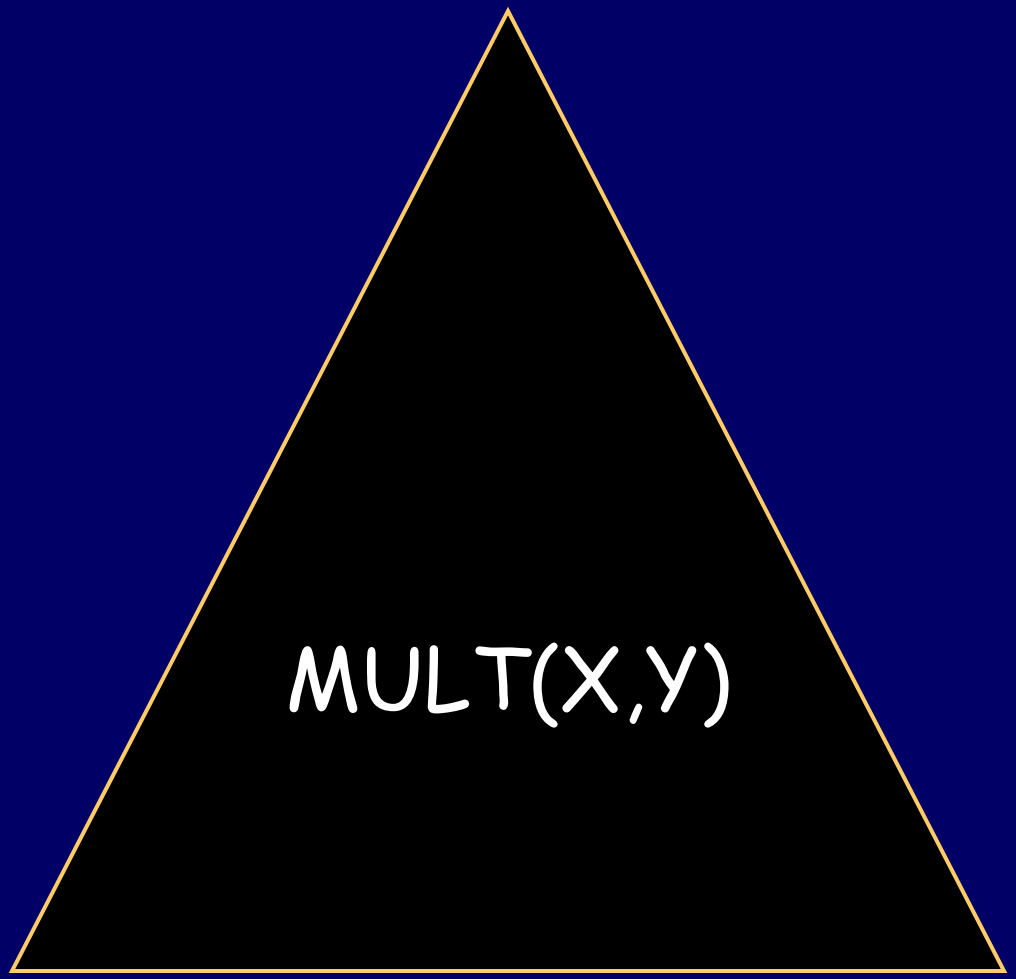
264126842539128

X = | a | b |

Y = | c | d |

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

Divide, Conquer, and Glue



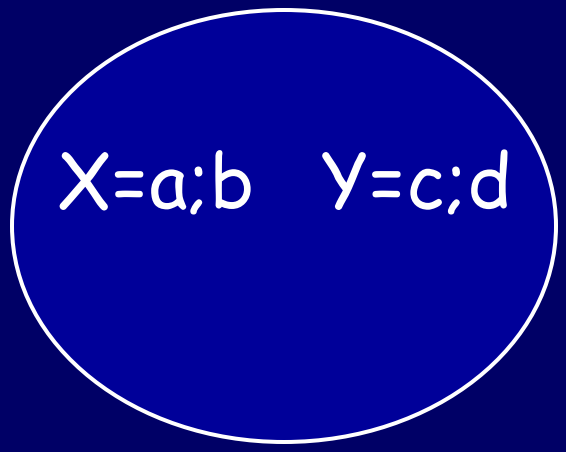
Divide, Conquer, and Glue

MULT(X,Y):

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

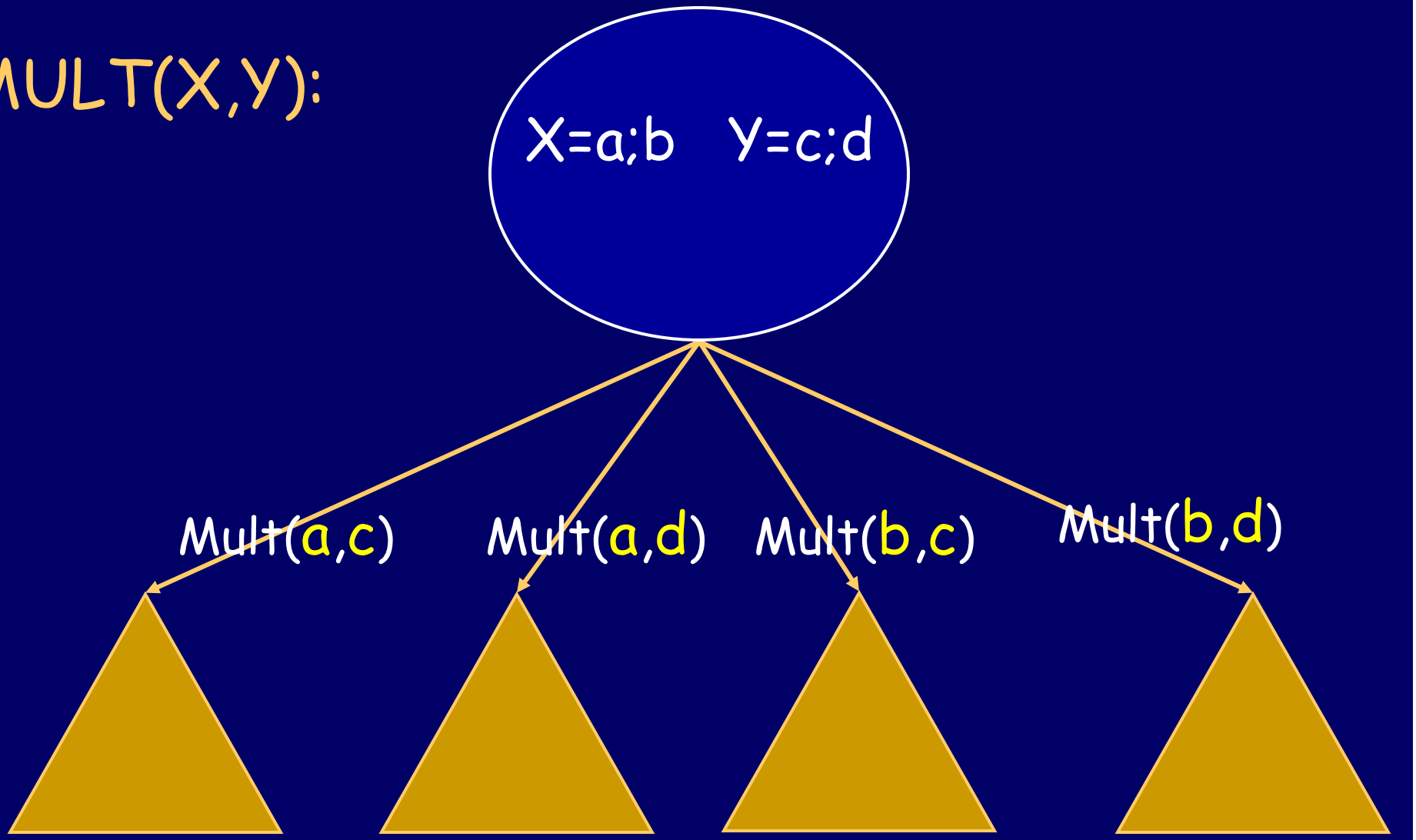
Divide, Conquer, and Glue

MULT(X,Y):

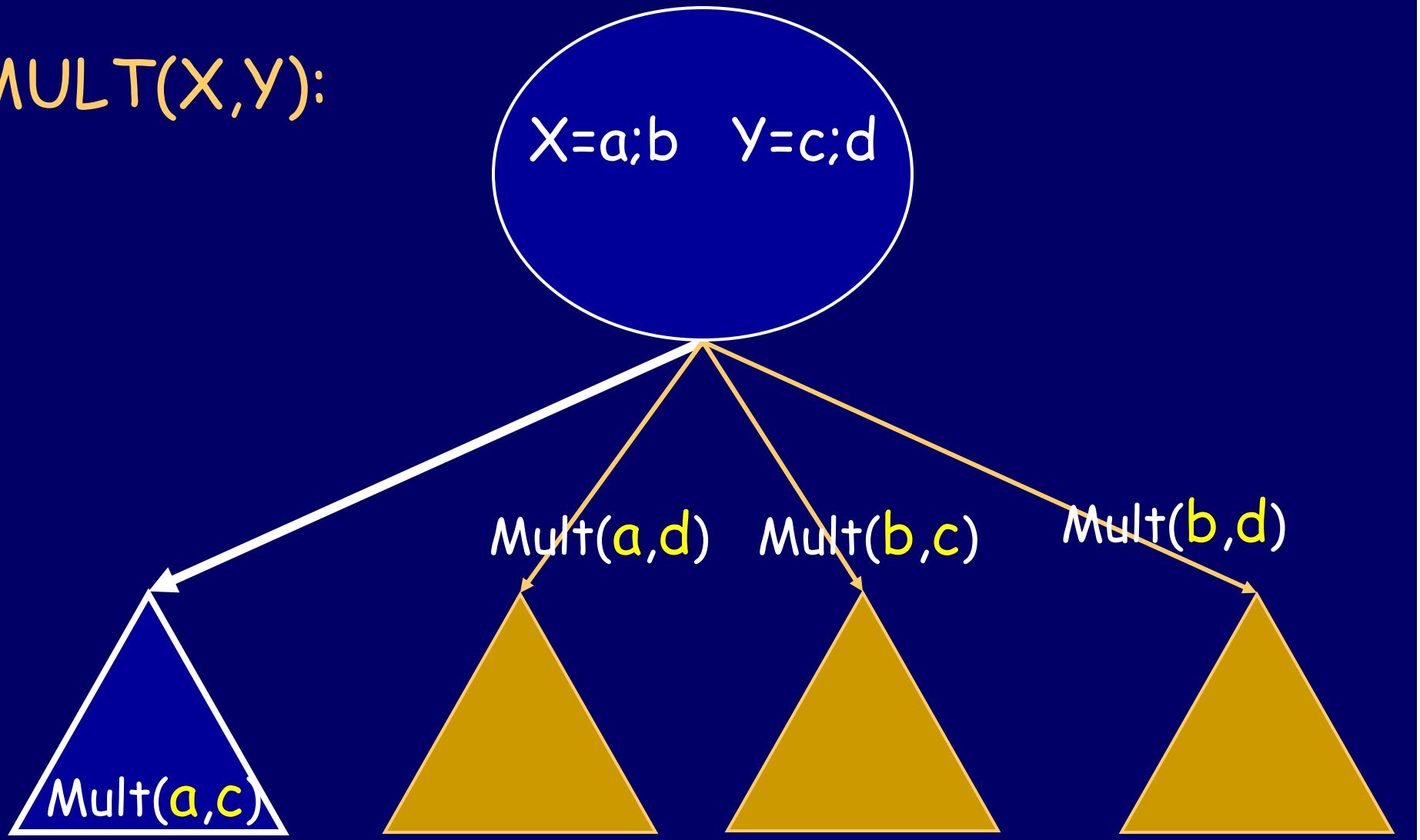


Divide, Conquer, and Glue

MULT(X,Y):

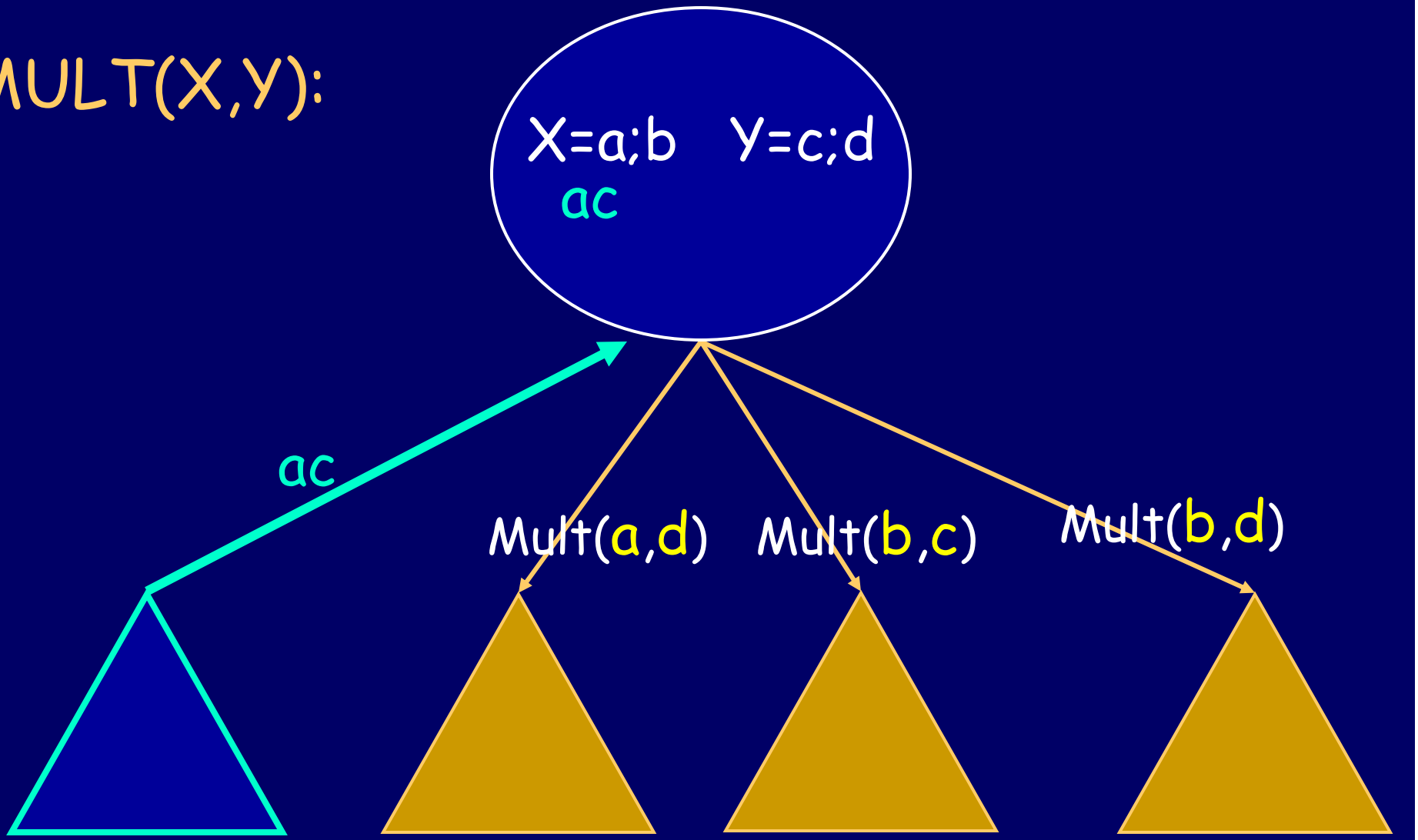


Divide, Conquer, and Glue

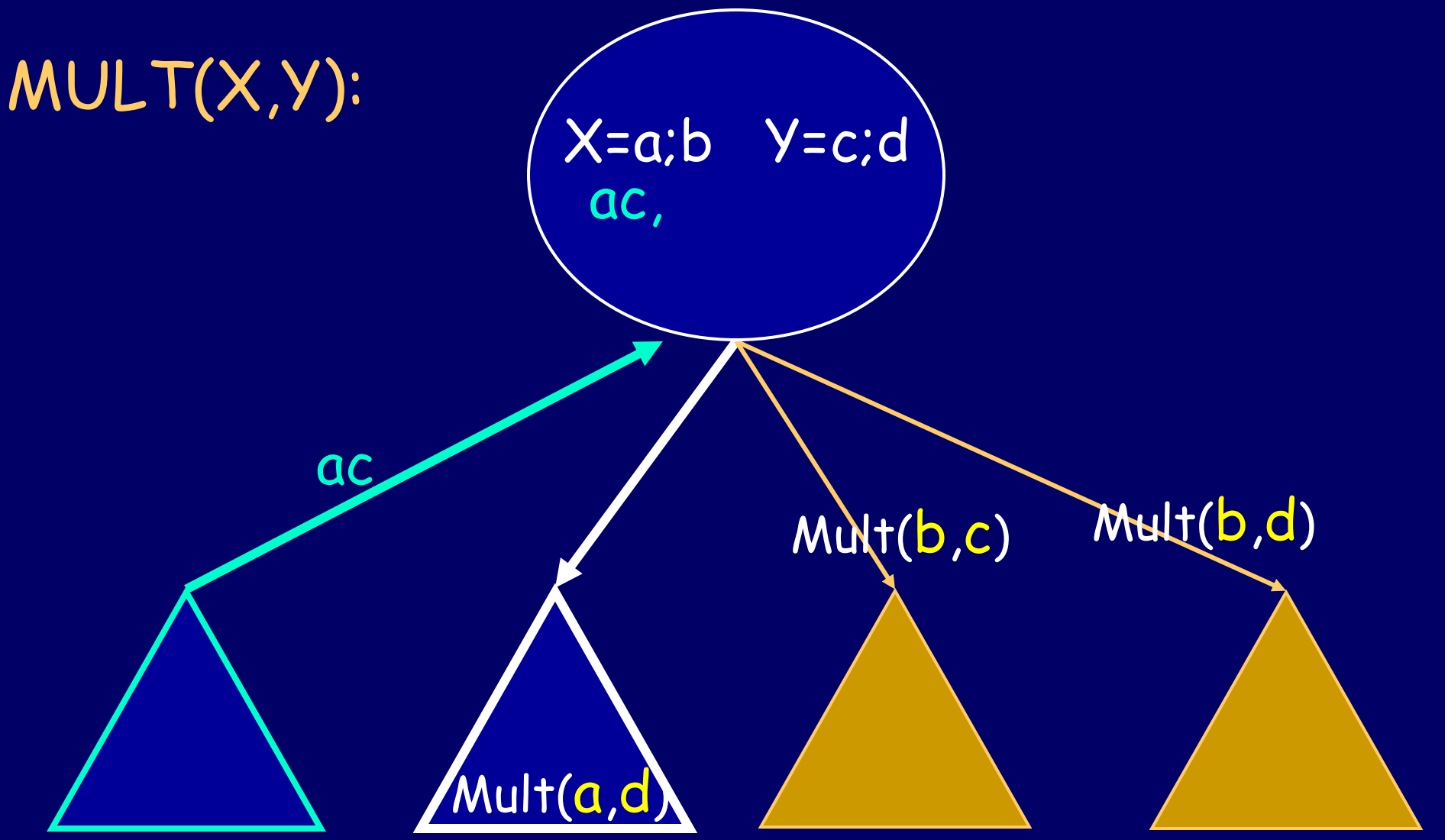


Divide, Conquer, and Glue

MULT(X,Y):

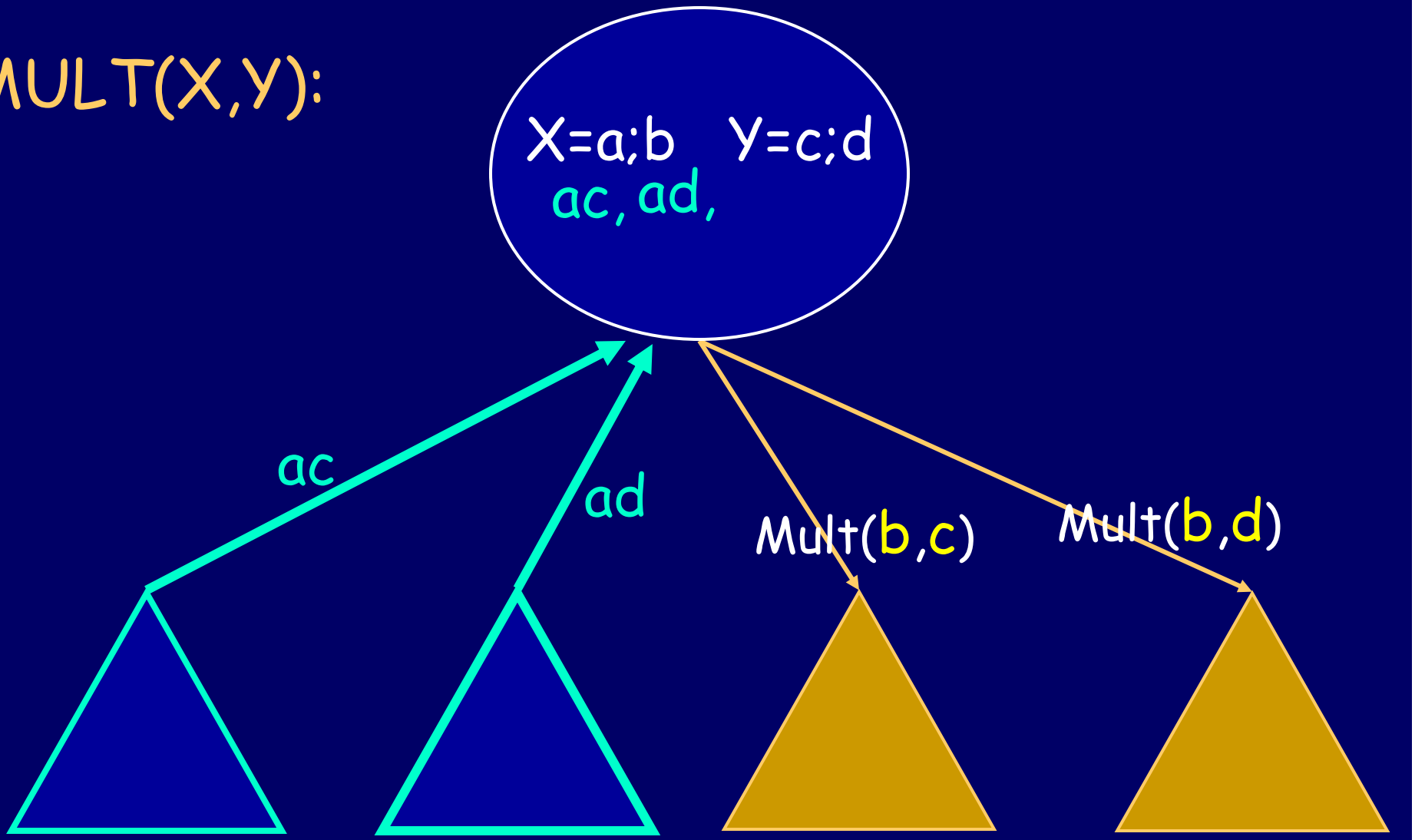


Divide, Conquer, and Glue



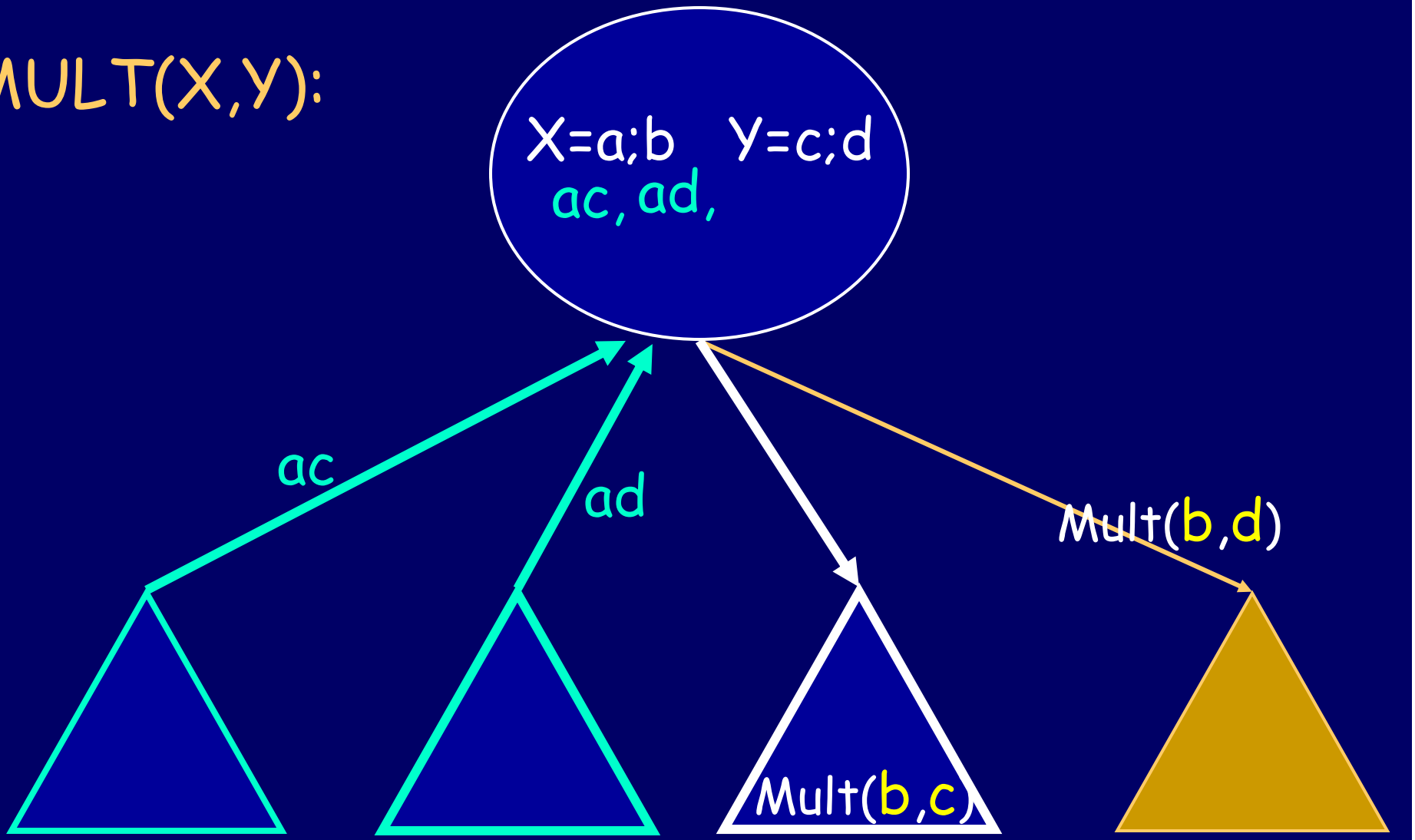
Divide, Conquer, and Glue

MULT(X,Y):



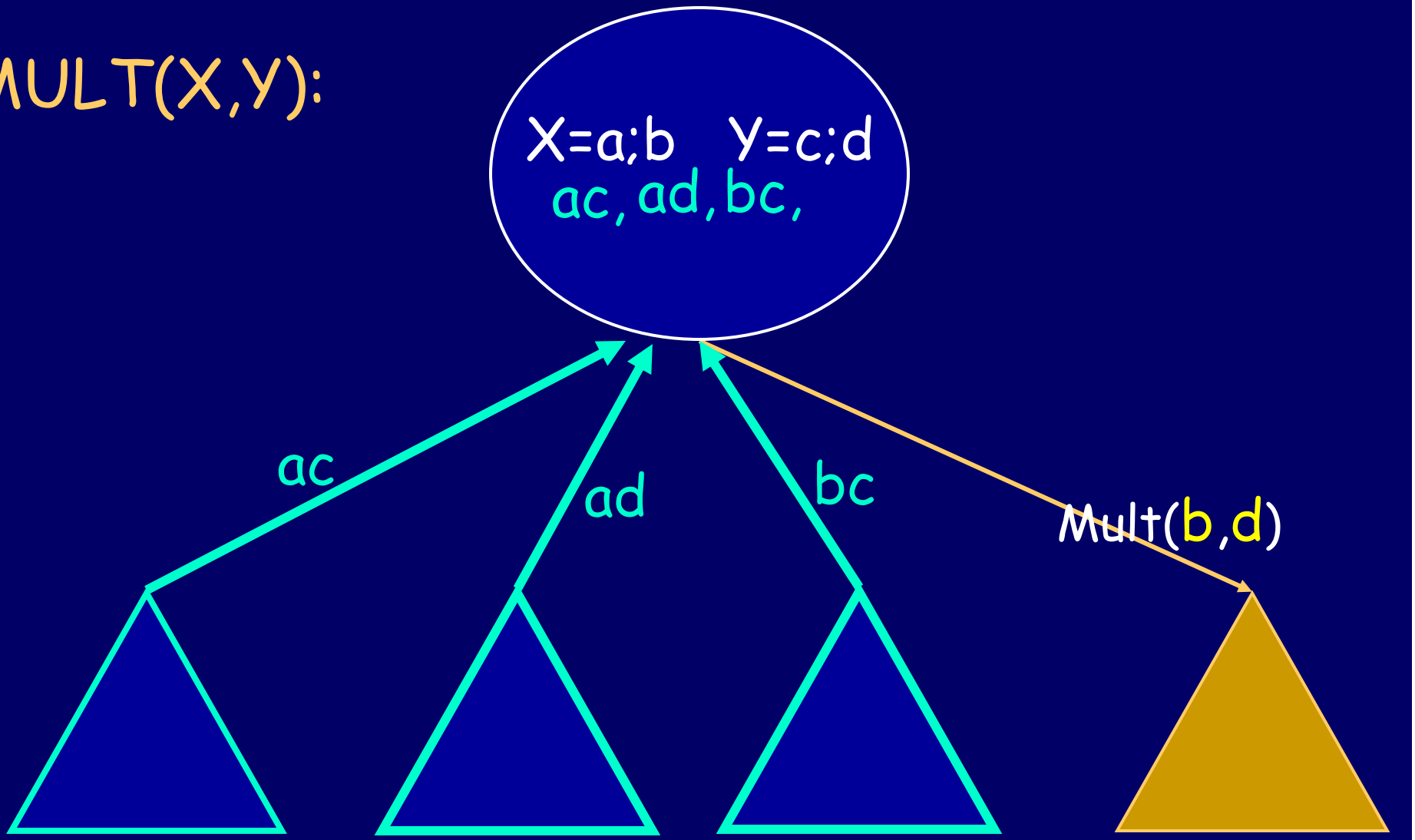
Divide, Conquer, and Glue

MULT(X,Y):



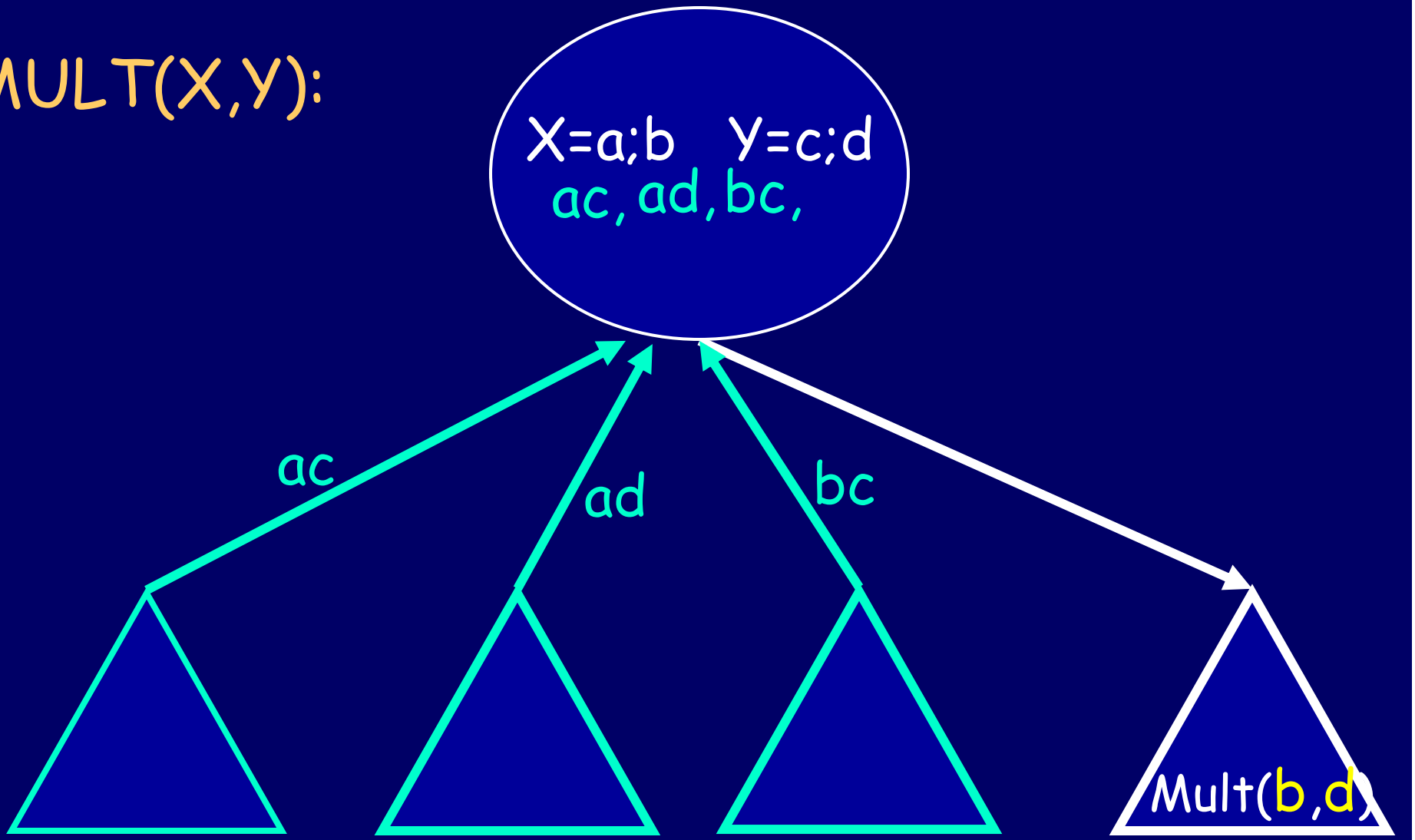
Divide, Conquer, and Glue

MULT(X,Y):



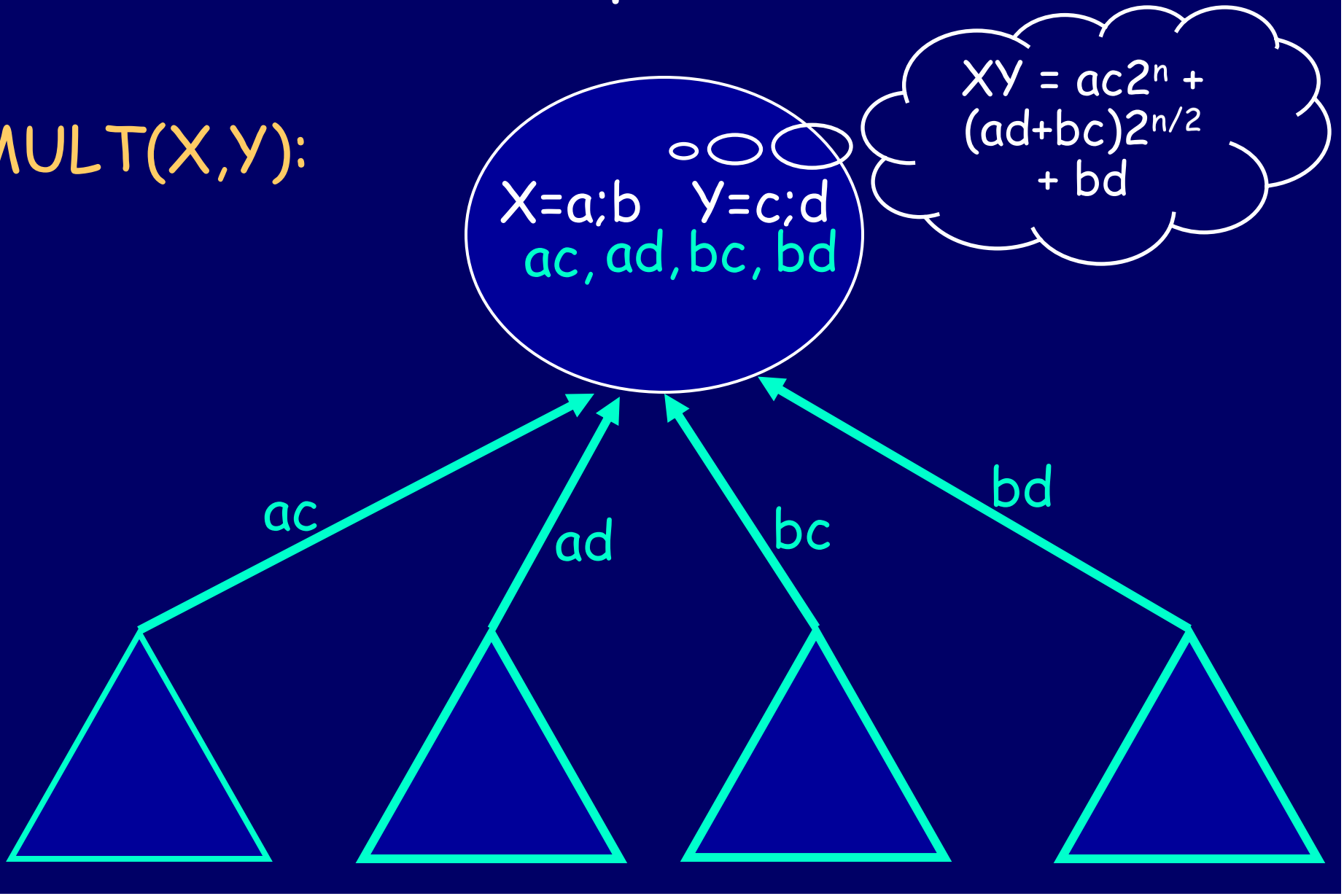
Divide, Conquer, and Glue

MULT(X,Y):



Divide, Conquer, and Glue

MULT(X,Y):

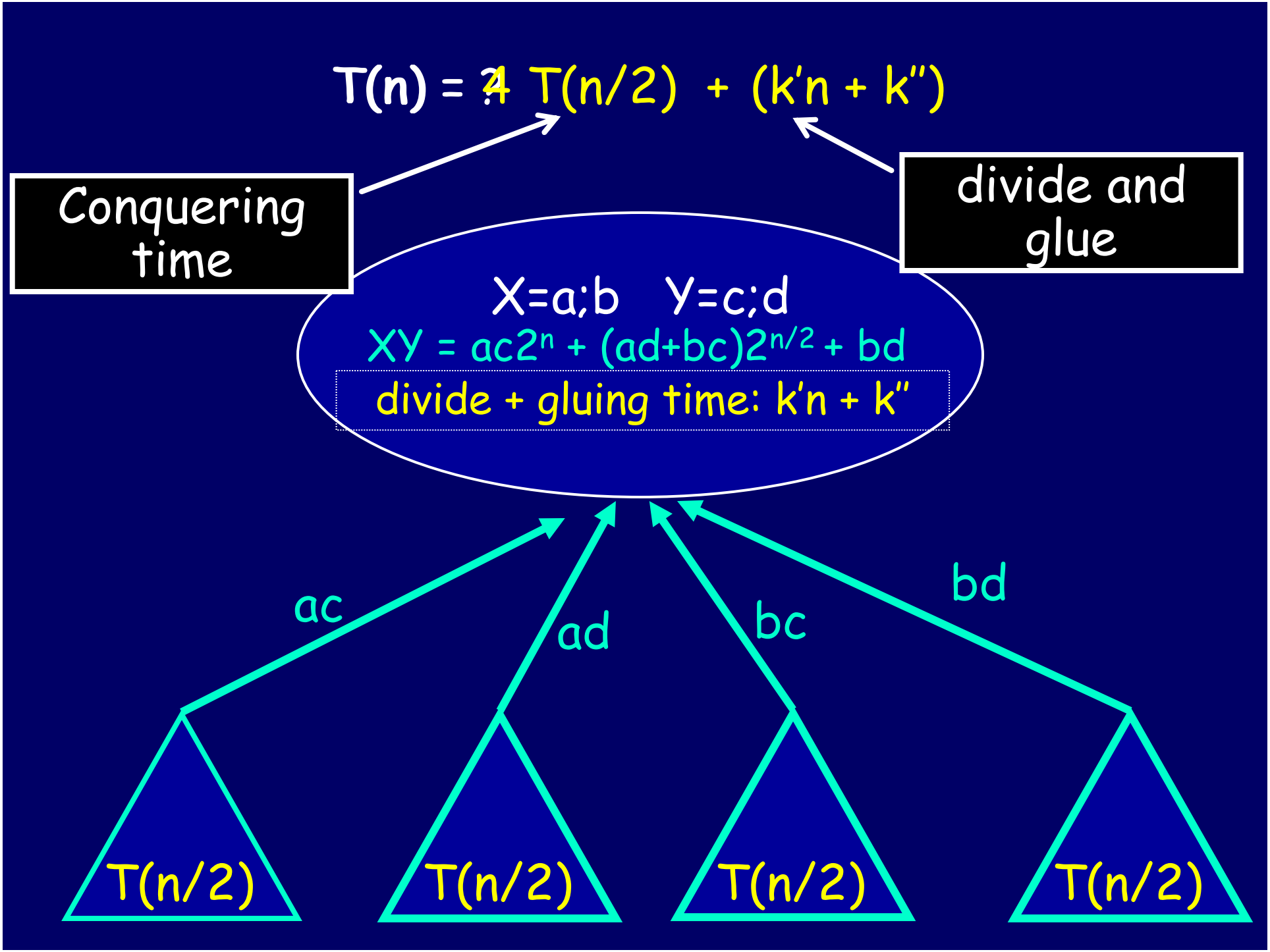


Time required by MULT

$T(n)$ = 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) + (k'n + k'')
Conquering time
divide and glue
X=a;b Y=c;d
XY = ac2^n + (ad+bc)2^n/2 + bd
divide + gluing time: k'n + k''
ac
ad
bc
bd
T(n/2)
T(n/2)
T(n/2)
T(n/2)

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

break X into a;b and Y into c;d

return

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

Let's be concrete and keep it simple

$T(1) = k \cdot 1$ 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

break X into a;b and Y into c;d

return

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

Let's be concrete and keep it simple

$T(1) = 1$

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

(Notice that T(n) is inductively defined only for positive powers of 2.)

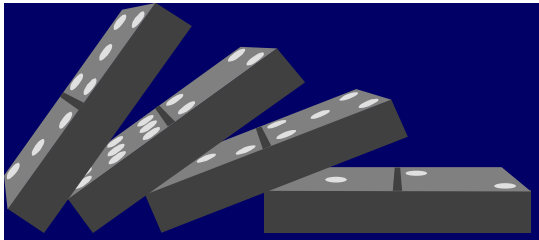
What is the growth rate of T(n)?

Technique 1: Guess and Verify

Guess: $G(n) = 2n^2 - n$

Verify: $G(1) = 1$ and $G(n) = 4\,G(n/2) + n$

$$
\begin{aligned}
& 4\,[2(n/2)^2 - n/2] + n \\
= \;& 2n^2 - 2n + n \\
= \;& 2n^2 - n \\
= \;& G(n)
\end{aligned}
$$



Technique 1: Guess and Verify

Guess: $G(n) = 2n^2 - n$

Verify: $G(1) = 1$ and $G(n) = 4\ G(n/2) + n$

Similarly: $T(1) = 1$ and $T(n) = 4\ T(n/2) + n$

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

Technique 2:
Guess Form and Calculate Coefficients

Guess: $T(n) = an^2 + bn + c$ for some a,b,c

Calculate: $T(1) = 1 \Rightarrow a + b + c = 1$

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

$$\Rightarrow\ an^2 + bn + c = 4\ [a(n/2)^2 + b(n/2) + c] + n$$

$$= an^2 + 2bn + 4c + n$$

$$\Rightarrow\ (b+1)n + 3c = 0$$

Therefore: b=-1 c=0 a=2

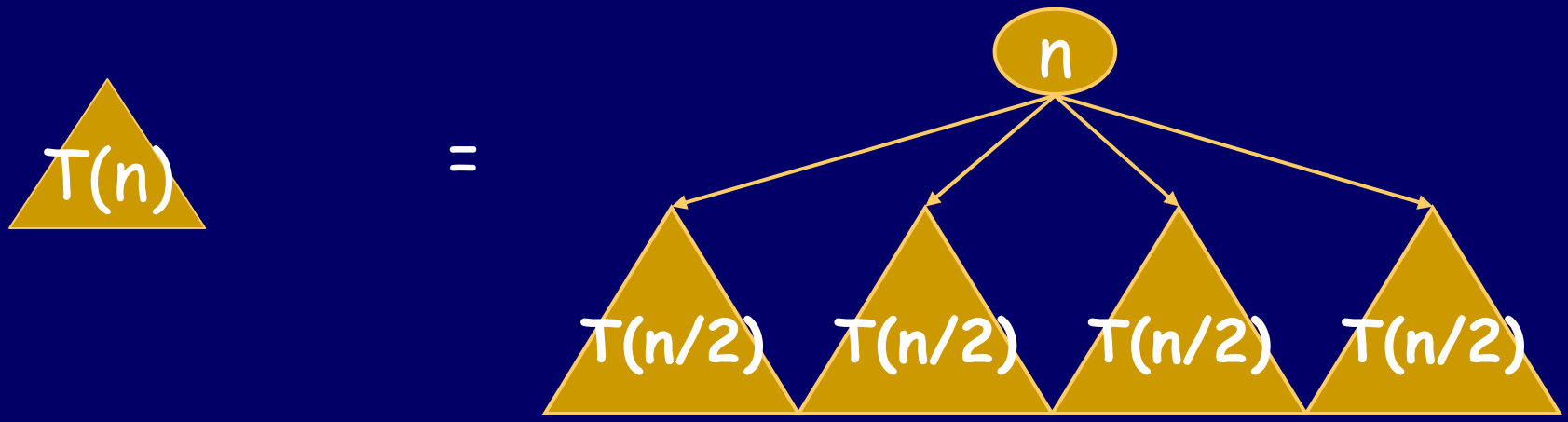
Technique 3: Labeled Tree Representation

Definition:
Labeled Tree

A tree node-labeled by S is a tree T = <V,E> with an associated function Label: V to S

Technique 3: Labeled Tree Representation

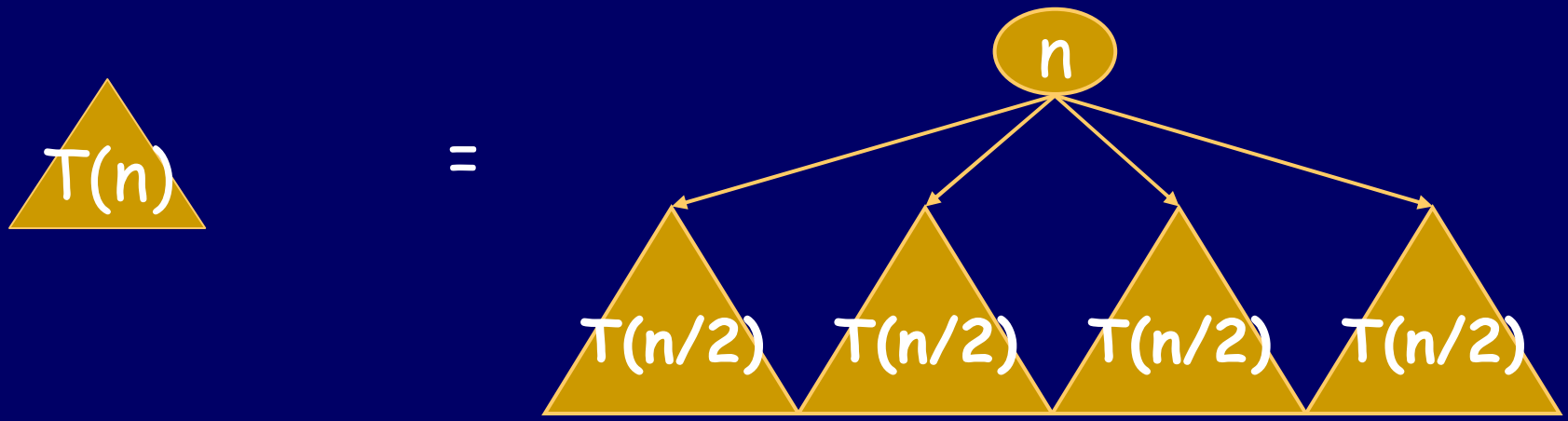
T(n)	=	n + 4 T(n/2)



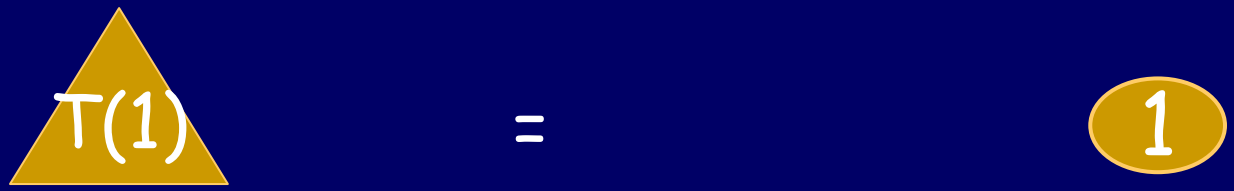
T(1)	=	1

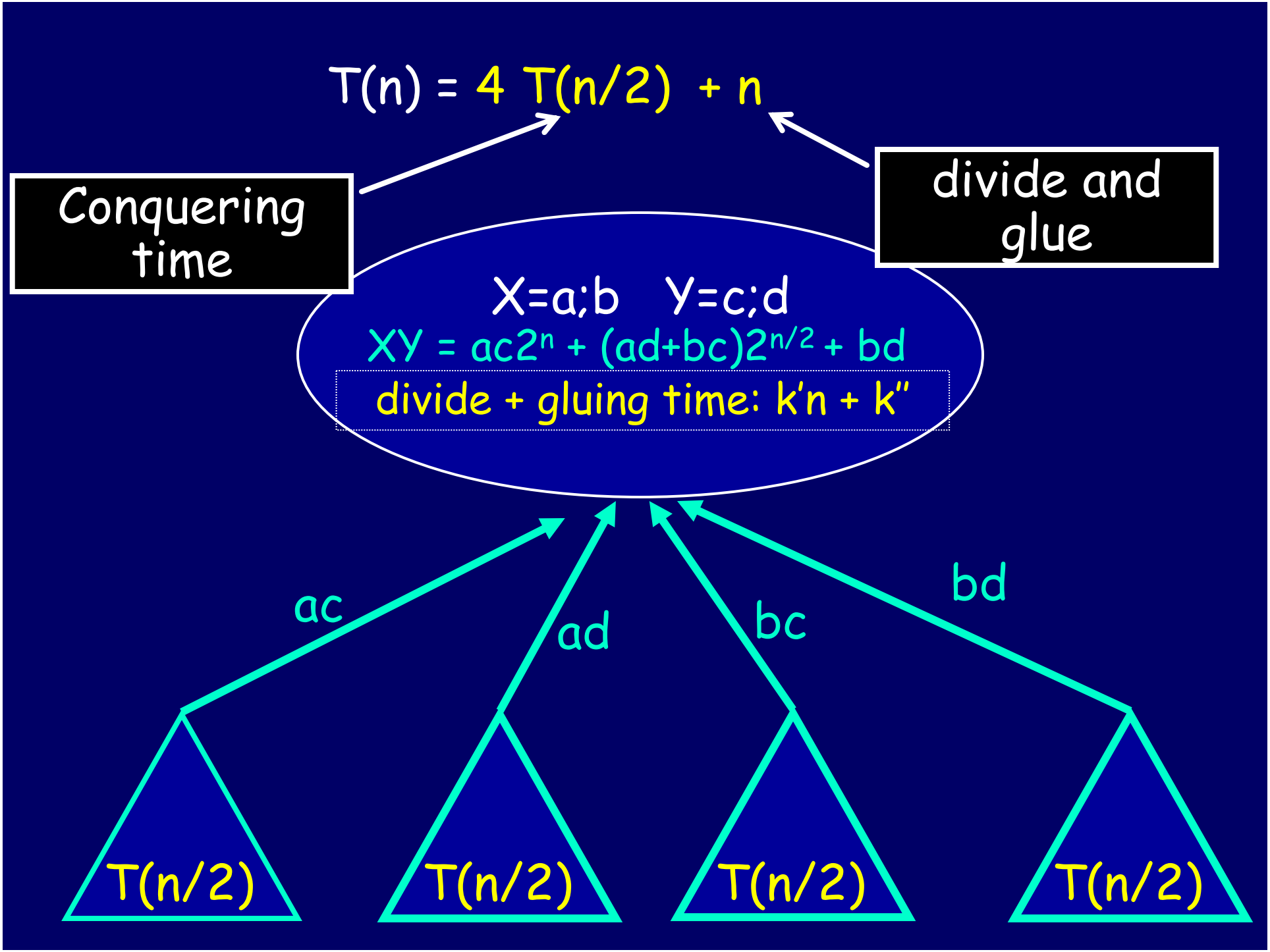
Node labels: time not spent conquering

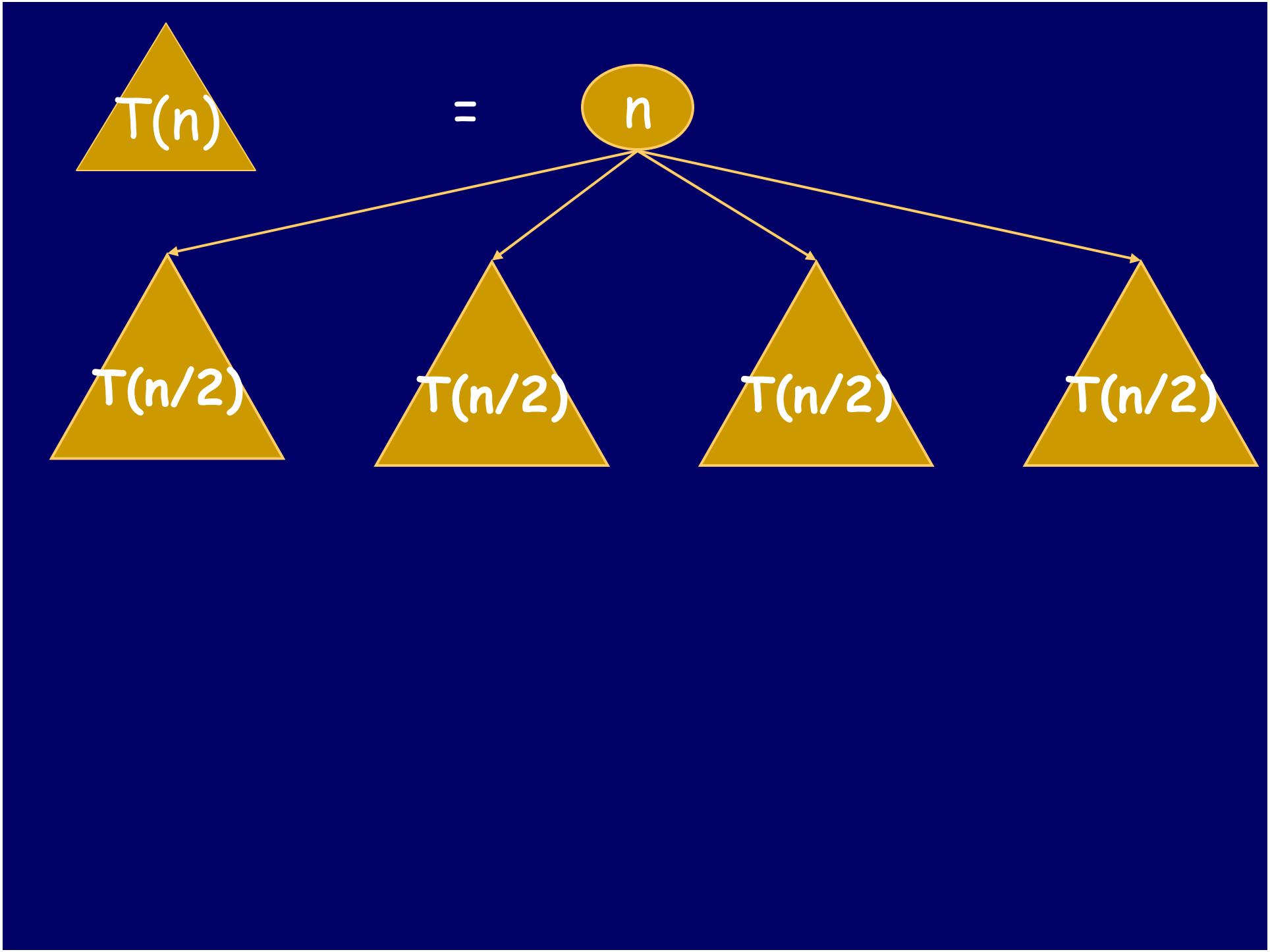
T(n)	=	n + 4 T(n/2)

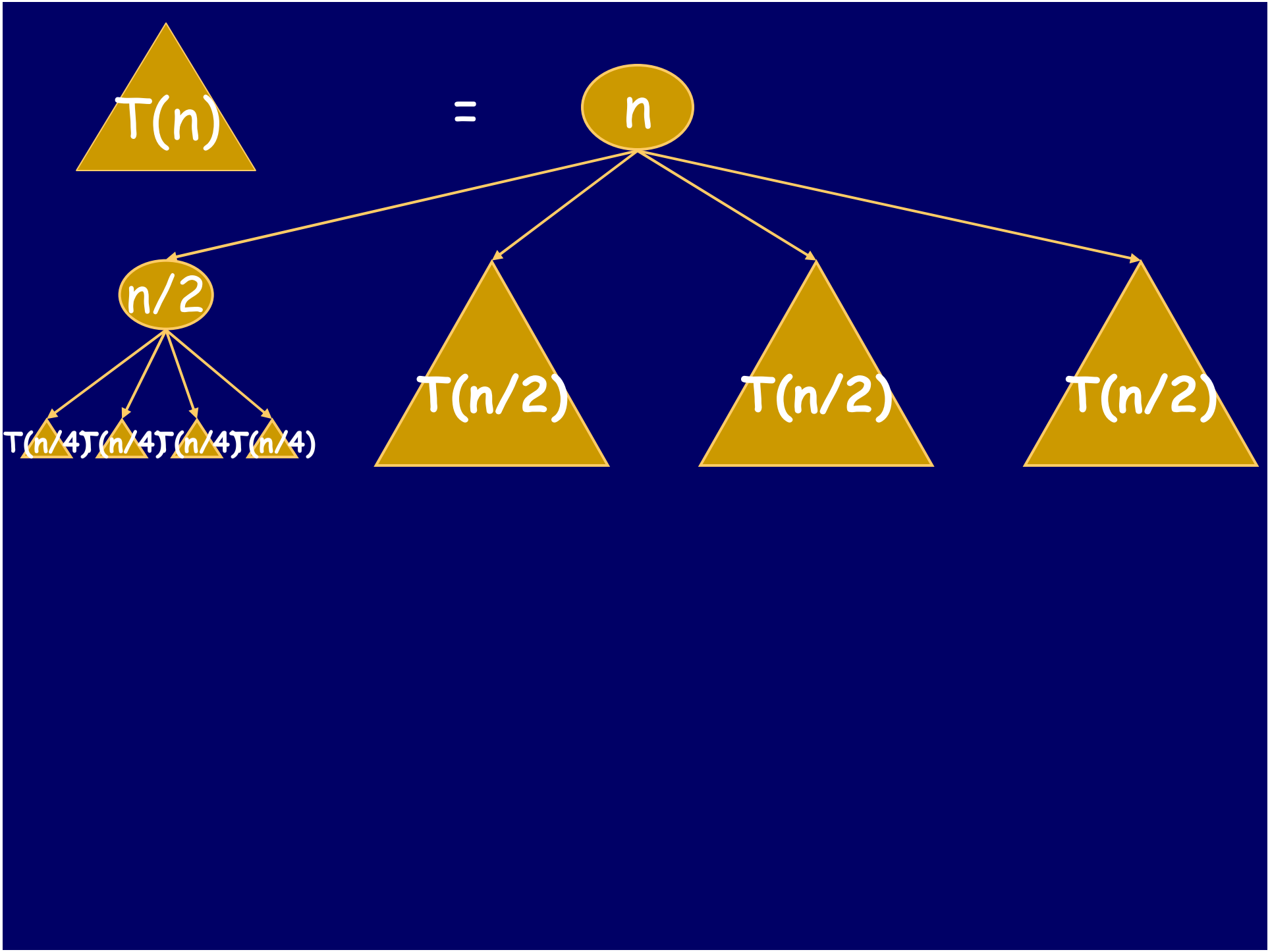


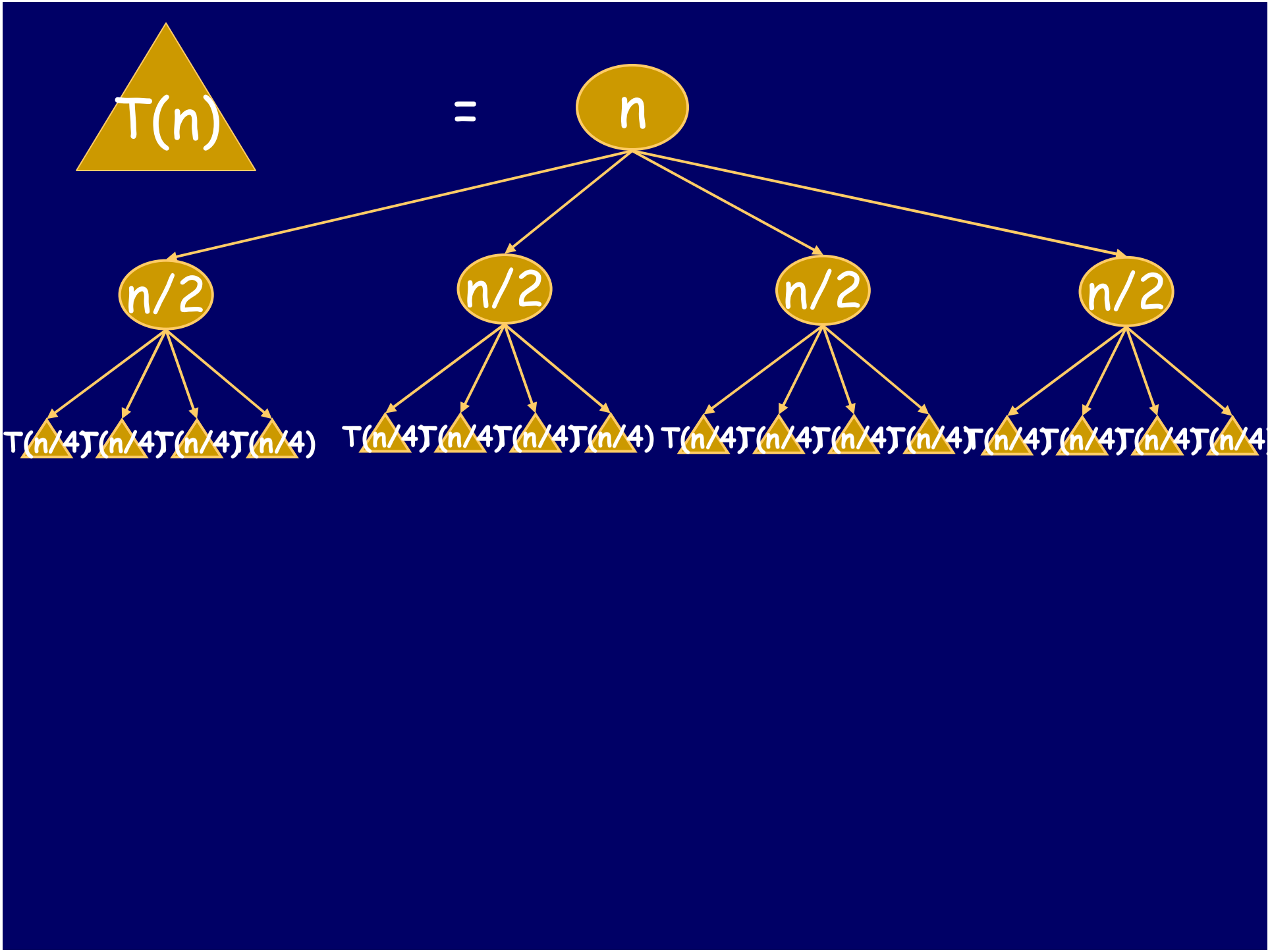
T(1)	=	1

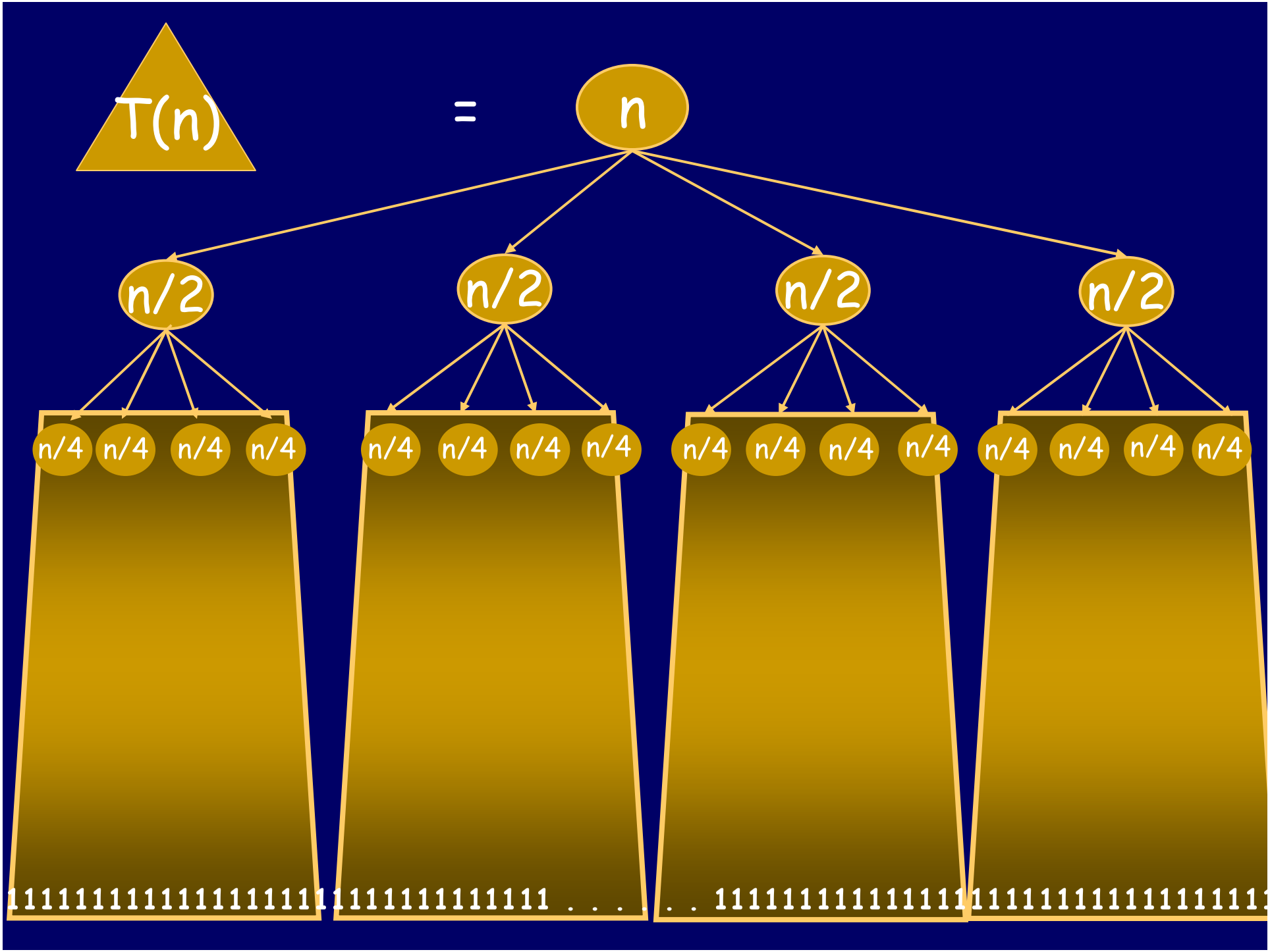


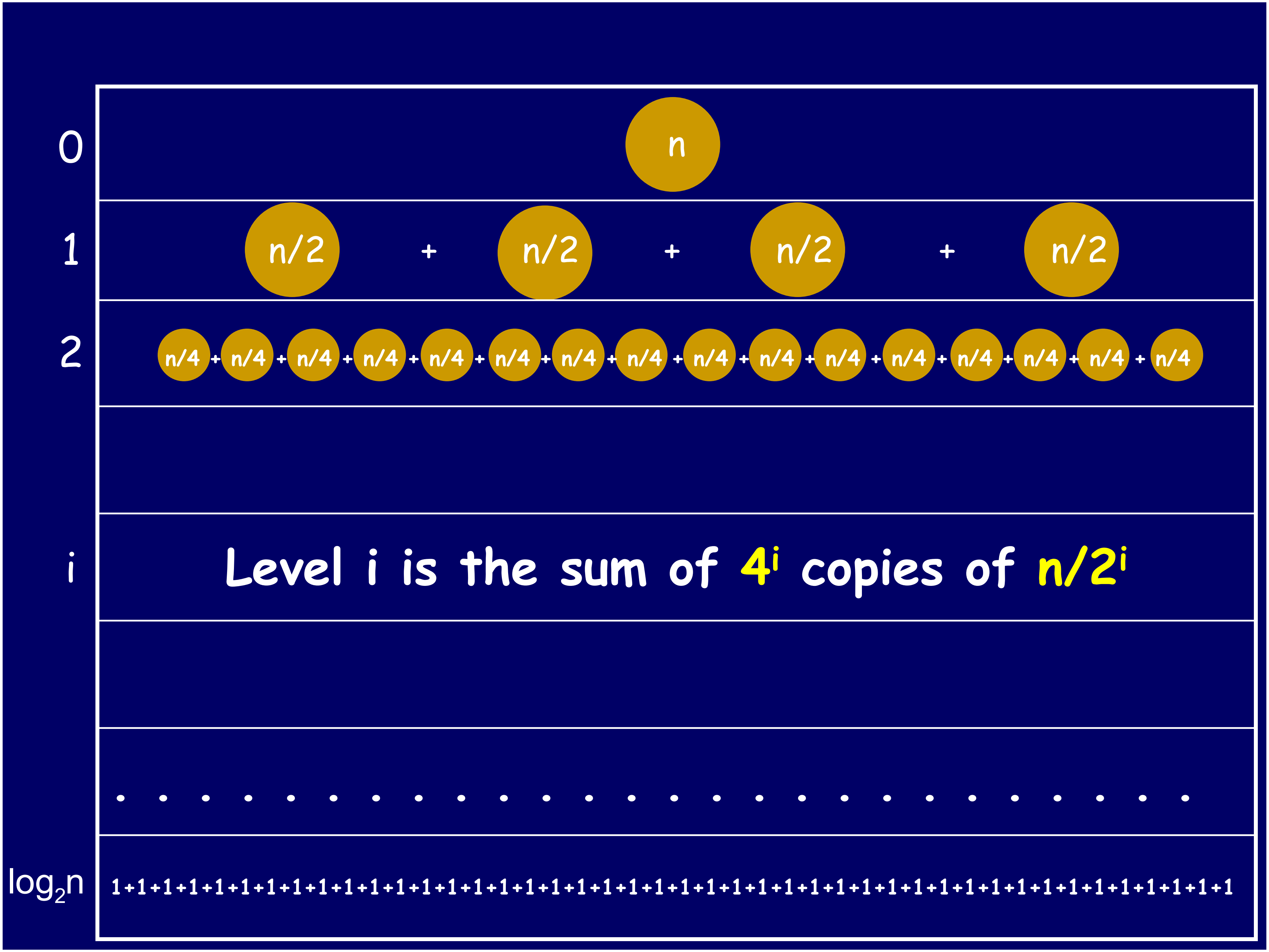

T(n) = 4 T(n/2) + n
Conquering time
divide and glue
X=a;b Y=c;d
XY = ac2^n + (ad+bc)2^n/2 + bd
divide + gluing time: k'n + k''
ac
ad
bc
bd
T(n/2)
T(n/2)
T(n/2)
T(n/2)

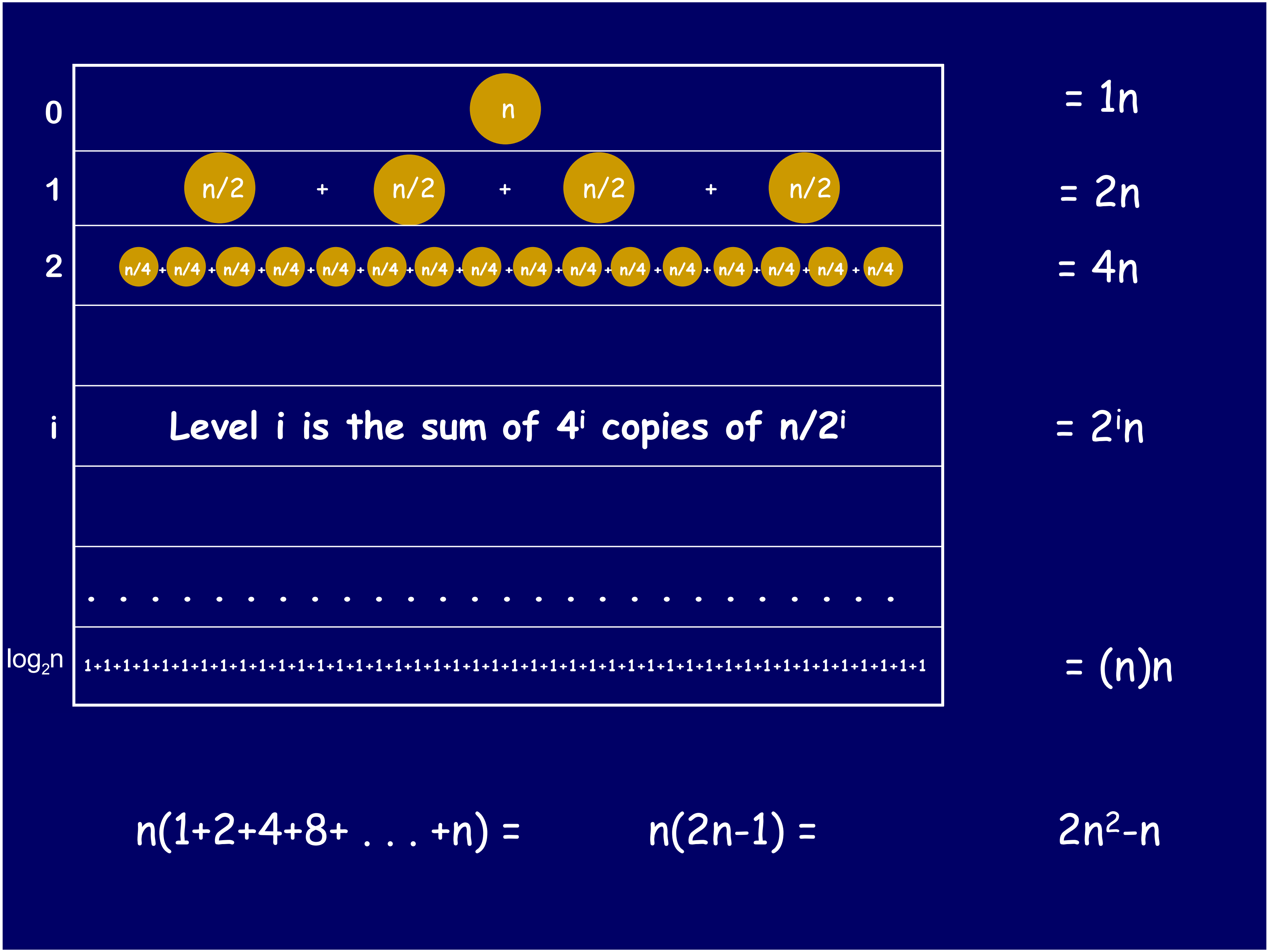

T(n)
=
n
T(n/2)
T(n/2)
T(n/2)
T(n/2)


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

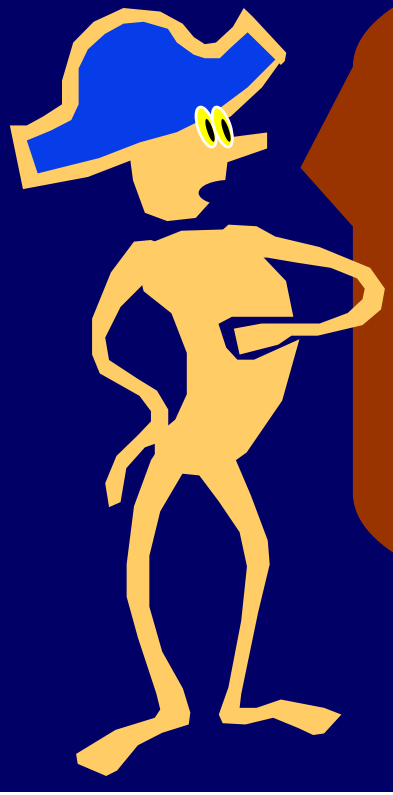

T(n)
=
n
n/2
n/2
n/2
n/2
n/4
n/4
n/4
n/4
n/4
n/4
n/4
n/4
n/4
n/4
n/4
n/4
n/4
n/4
n/4
n/4


0
1
2
i
log2n
n
n/2 + n/2 + n/2 + n/2
n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4
Level i is the sum of 4i copies of n/2i
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+1+1+1+1+1+1+1+1+1+1+1


0
n
= 1n
1
n/2 + n/2 + n/2 + n/2
= 2n
2
n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4
= 4n
i
Level i is the sum of 4^i copies of n/2^i
= 2^i n
. . . .
log2n
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+1+1+1+1+1+1+1+1+1+1+1
= (n)n
n(1+2+4+8+ . . . +n) =
n(2n-1) =
2n2-n

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

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



In retrospect, it is obvious that the kissing number for Divide and Conquer MULT is n^2, since the leaves are in correspondence with the kisses.

MULT revisited

MULT(X,Y):

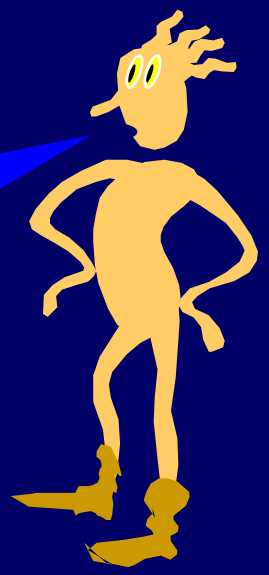
If |X| = |Y| = 1 then return XY

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

(a+bi)(c+di) =
[ac −bd] + [ad + bc] i

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

Karatsuba, Anatolii Alexeevich (1937-)

Sometime in the late 1950's Karatsuba had formulated the first algorithm to break the kissing barrier!

Gaussified MULT (Karatsuba 1962)

MULT(X,Y):

If |X| = |Y| = 1 then return XY

break X into a;b and Y into c;d

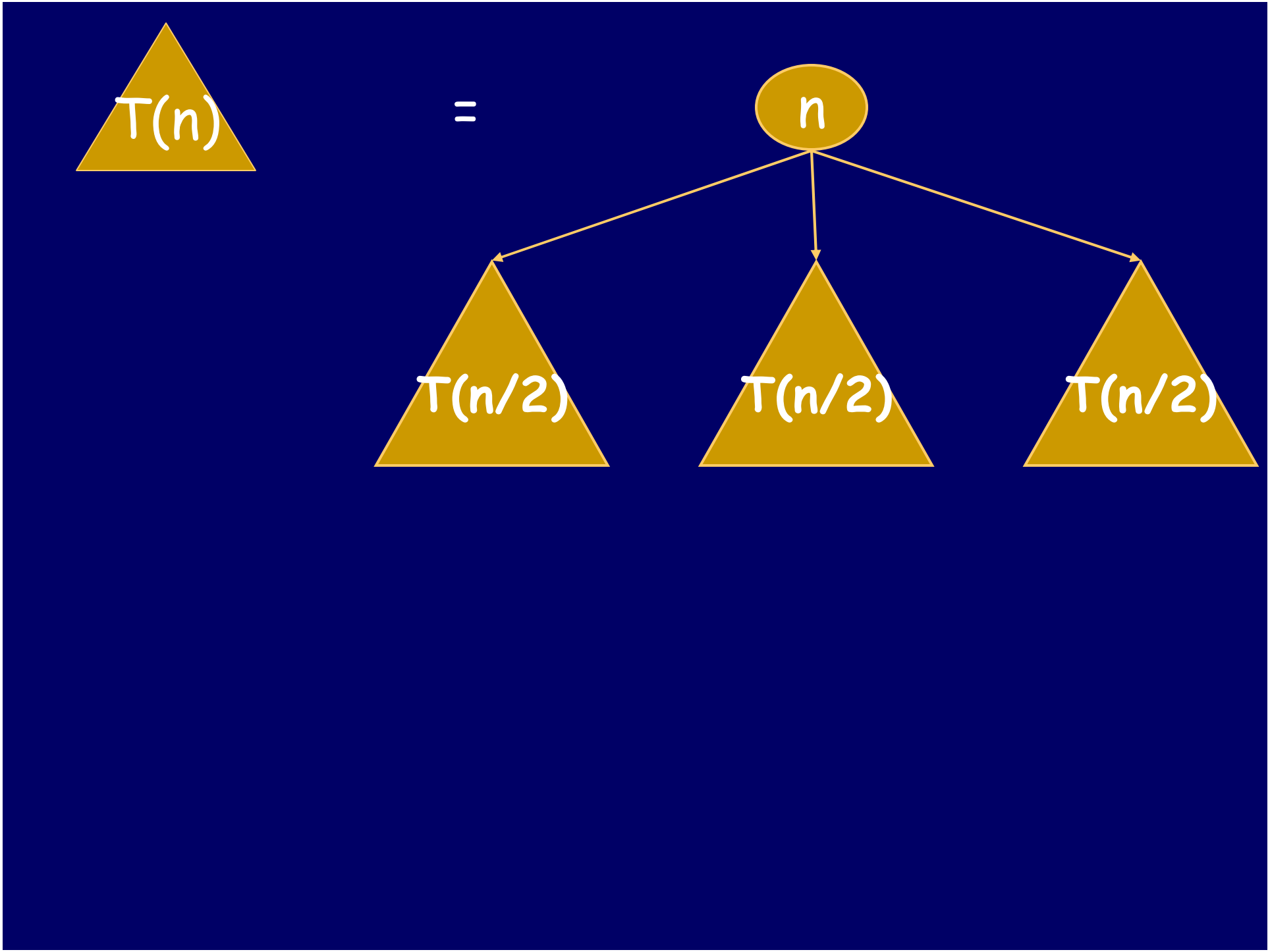
e = MULT(a,c) and f = MULT(b,d)

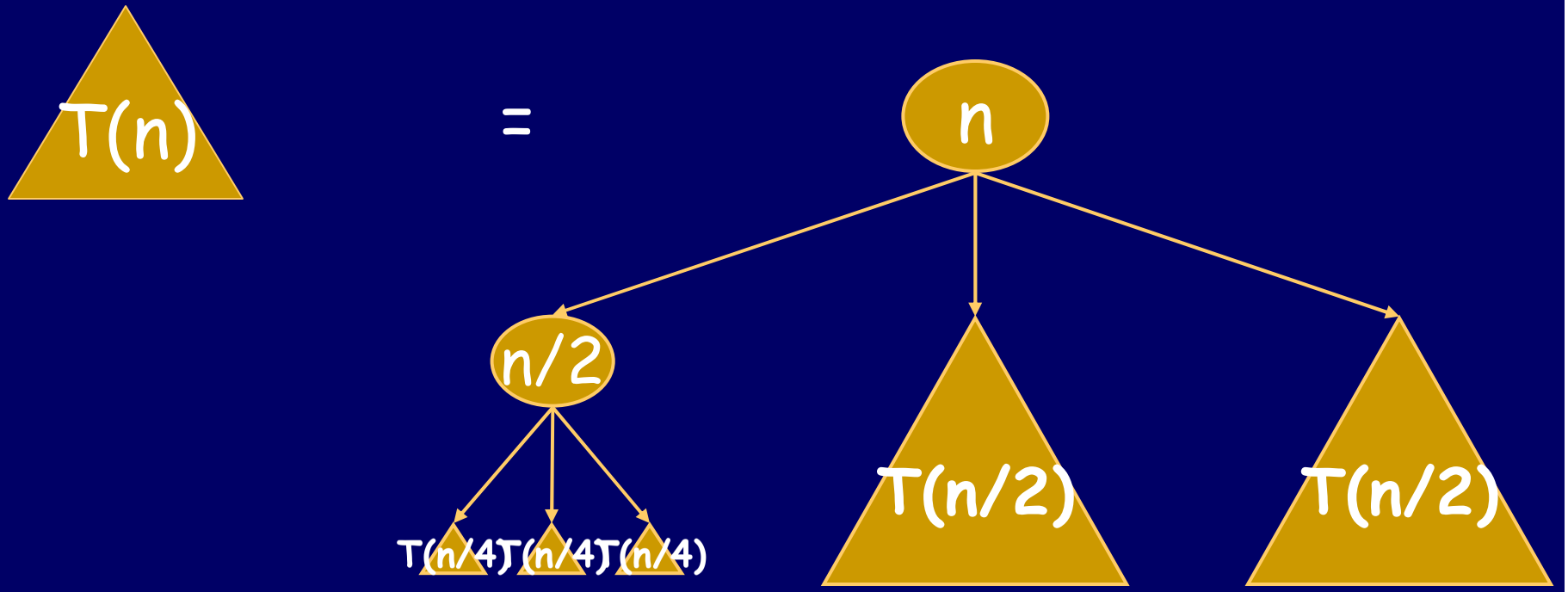
return

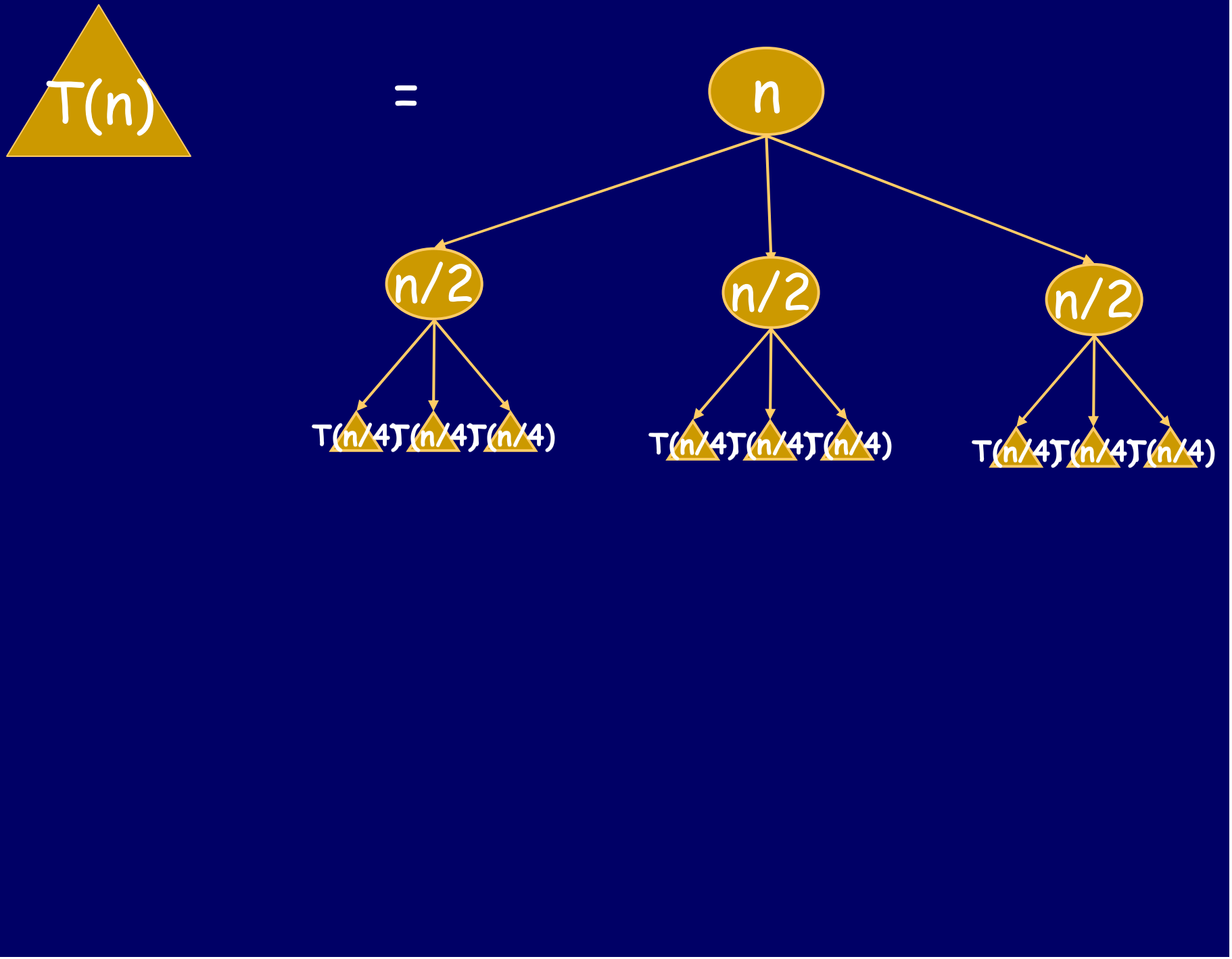
$e\ 2^n + (MULT(a+b,c+d) - e - f)\ 2^{n/2} + f$

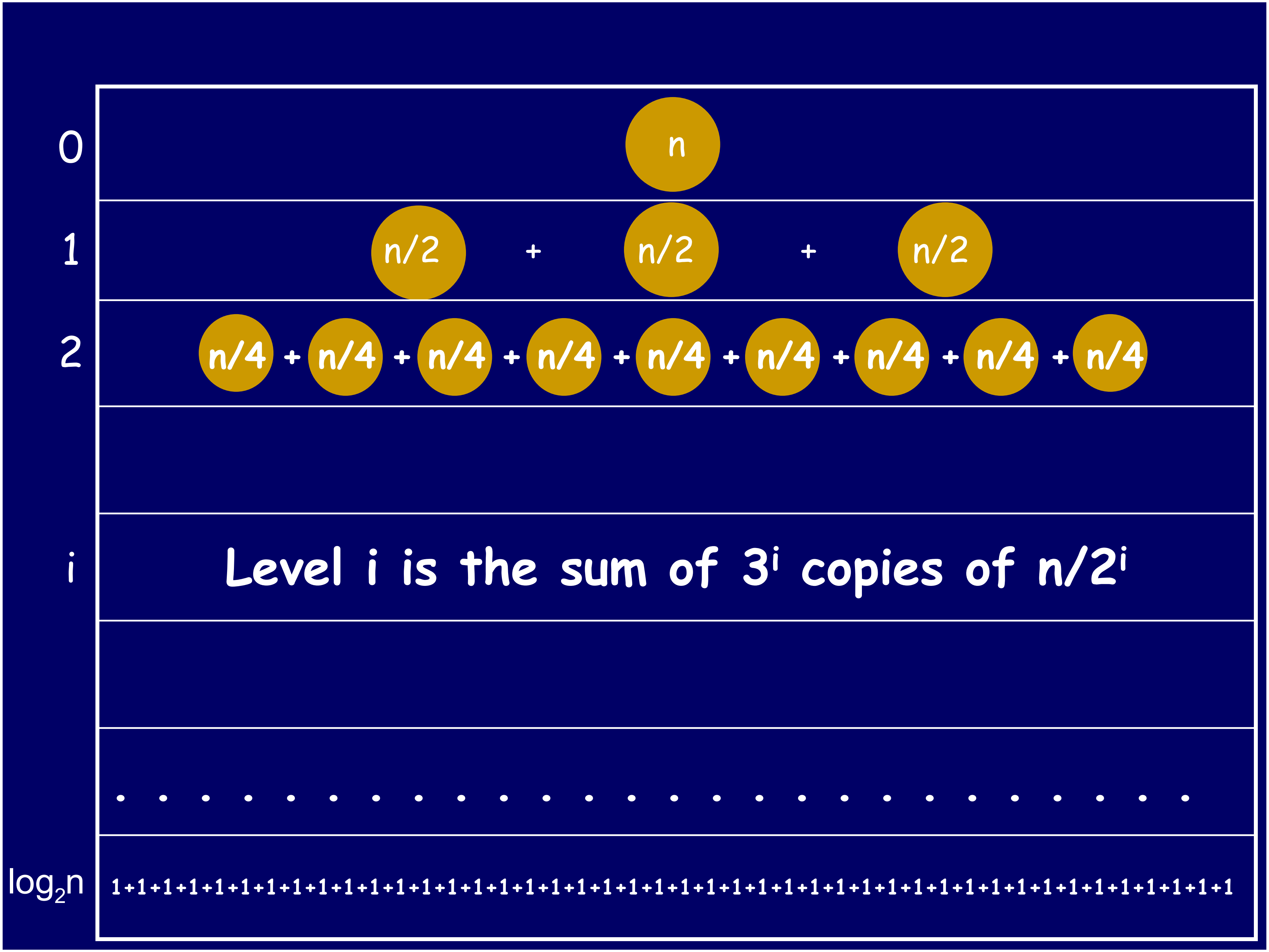
$$T(n) = 3\ T(n/2) + n$$

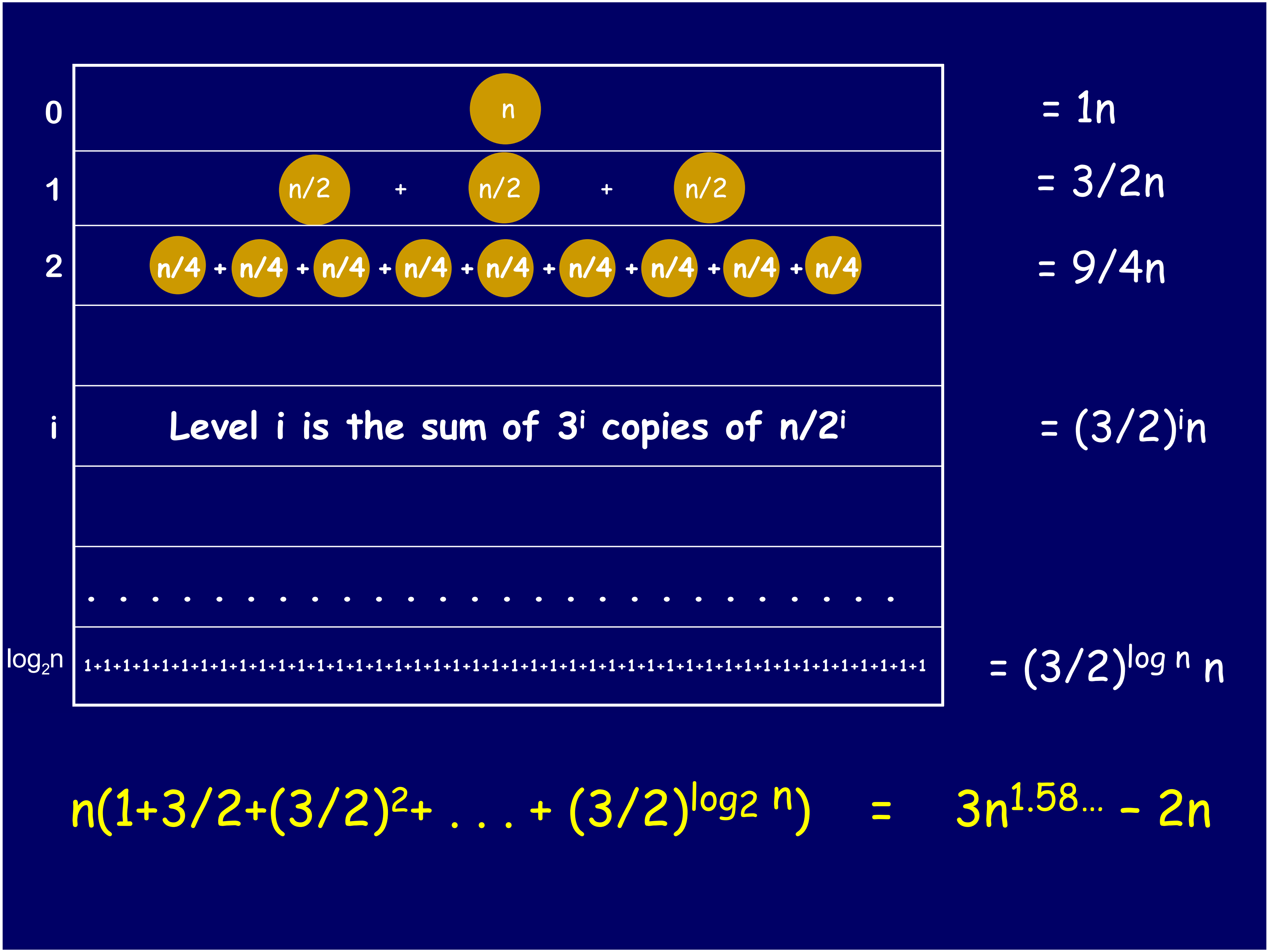
Actually: $T(n) = 2\ T(n/2) + T(n/2 + 1) + kn$


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

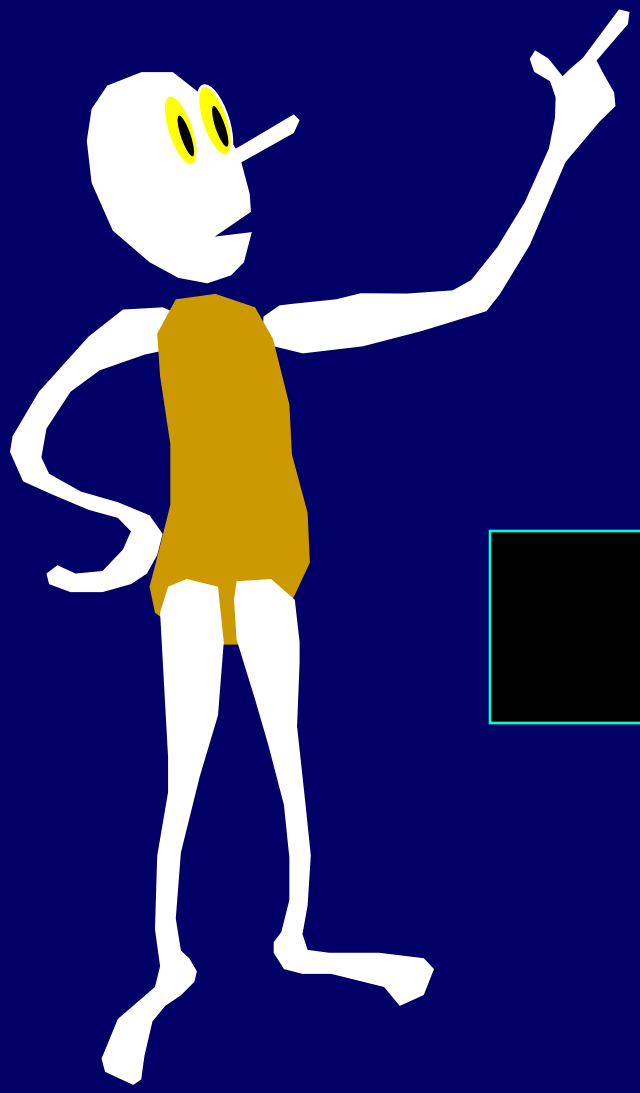

T(n)
=
n
n/2
T(n/4)
T(n/4)
T(n/4)
T(n/2)
T(n/2)


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


0
n
1
n/2
+
n/2
+
n/2
2
n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4
i
Level i is the sum of 3^i copies of n/2^i
. . . .
log2n
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+1+1+1+1+1+1+1+1+1+1+1


0
n
= 1n
1
n/2 + n/2 + n/2
= 3/2n
2
n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4 + n/4
= 9/4n
i
Level i is the sum of 3^i copies of $n/2^i$
$= (3/2)^i n$
. . . .
$\log_2 n$
$= (3/2)^{\log n} n$
$n(1+3/2+(3/2)^2+ \dots + (3/2)^{\log_2 n}) = 3n^{1.58\dots} - 2n$

$$1 + X^1 + X^2 + X^3 + \ldots + X^{n-1} + X^n = \frac{X^{n+1} - 1}{X - 1}$$



The Geometric Series

Substituting into our formula….

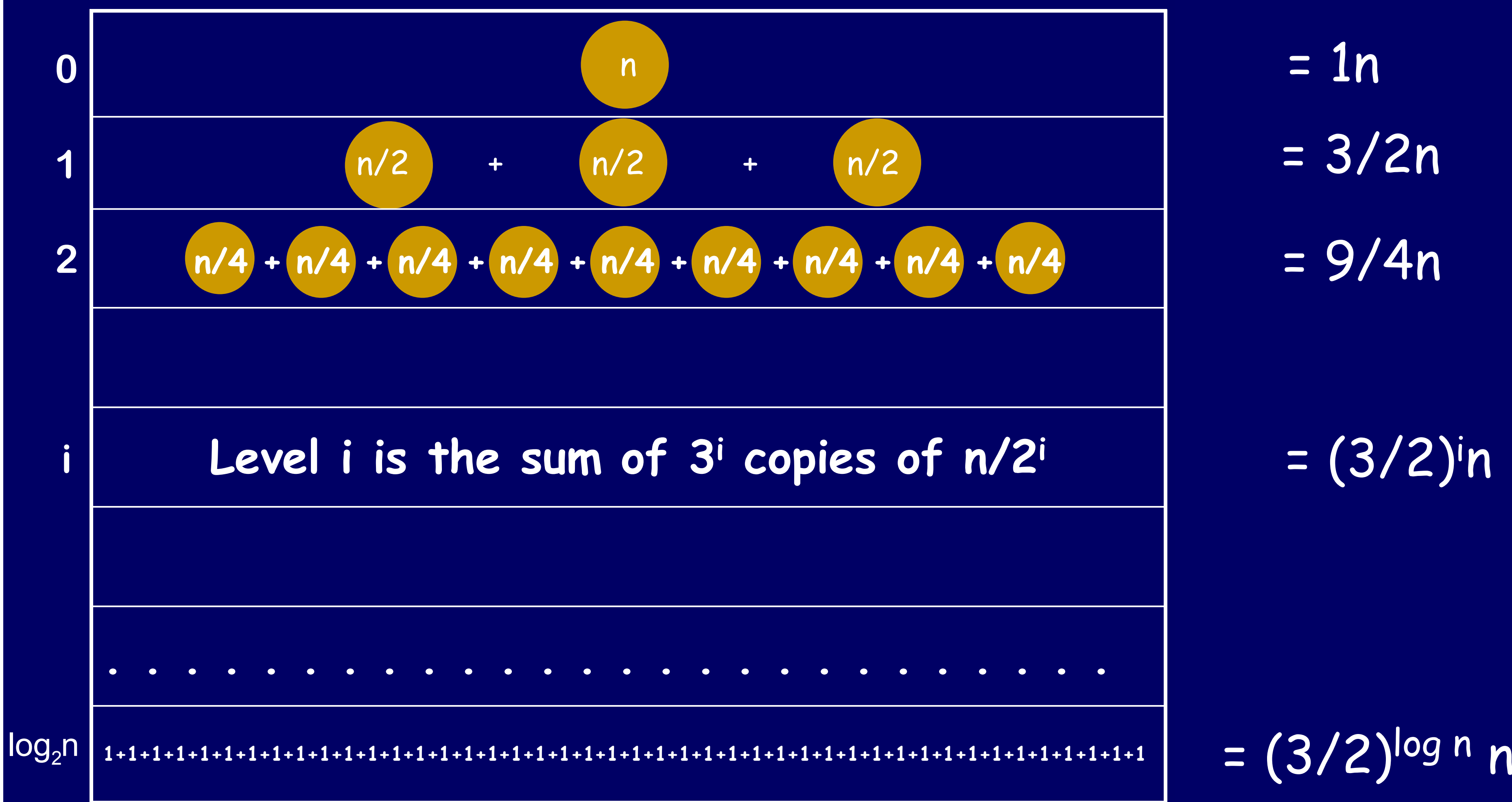
$$1 + X^1 + X^2 + X^3 + \ldots + X^{k-1} + X^k = \frac{X^{k+1} - 1}{X - 1}$$

We have: $X = 3/2$ $\qquad\qquad$ $k = \log_2 n$

$$\frac{(3/2) \times (3/2)^{\log_2 n} - 1}{\frac{1}{2}} = 3 \times (3^{\log_2 n}/2^{\log_2 n}) - 2$$

$$= 3 \times (3^{\log_2 n}/n) - 2$$

$$= \frac{3\, n^{(\log_2 3)}}{n} - 2$$

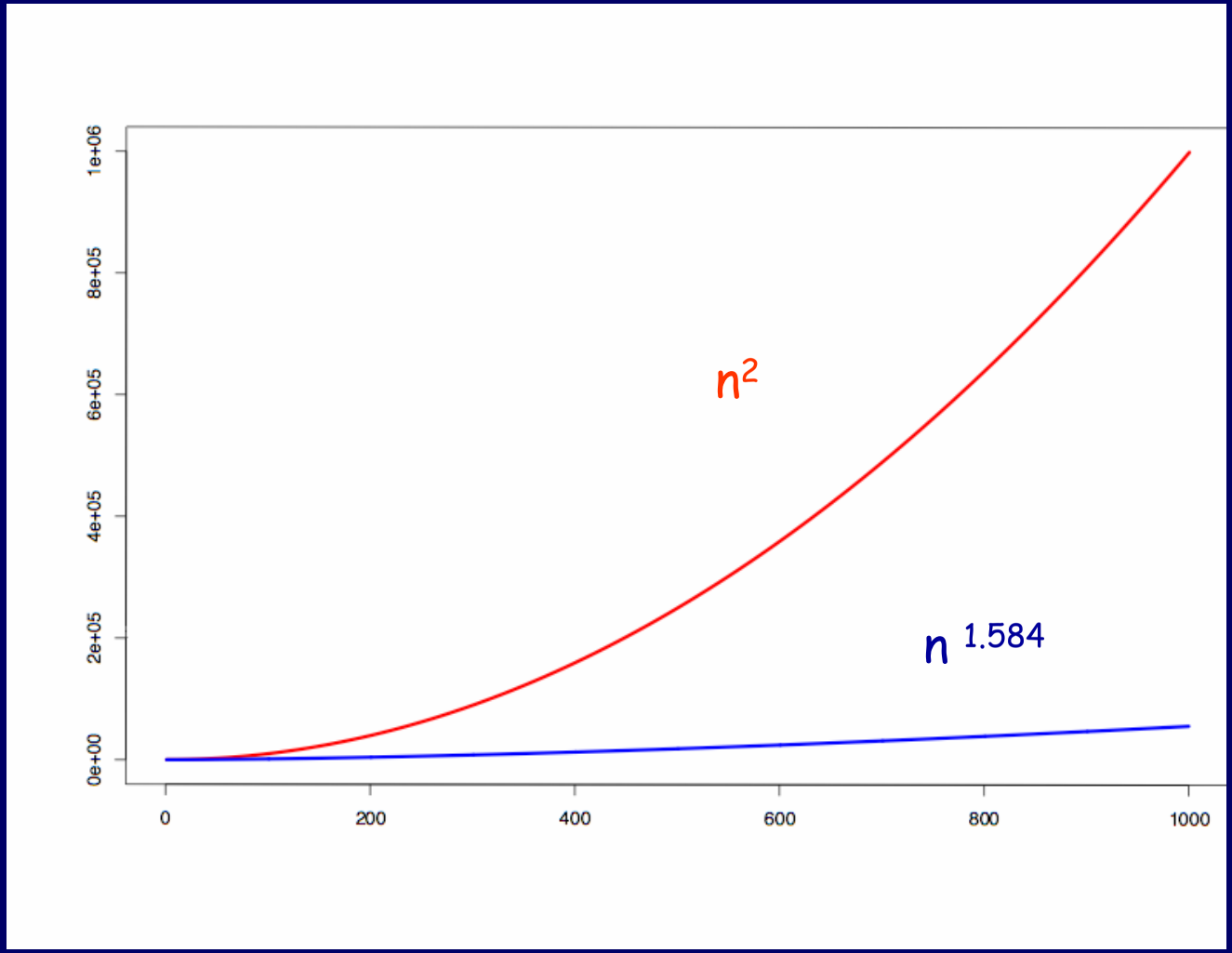


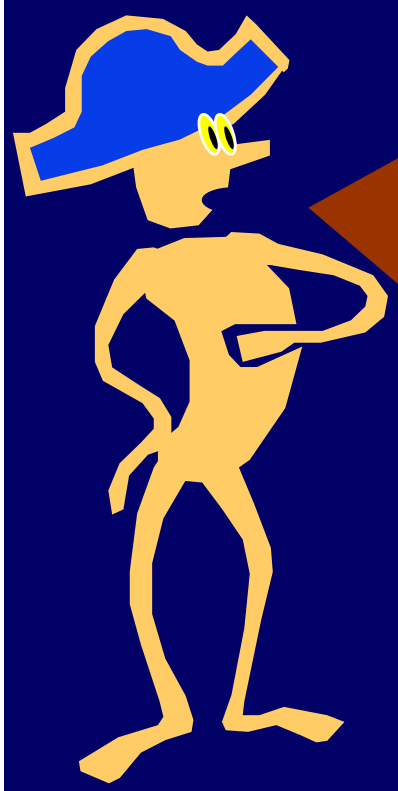
$$n\left(\frac{3n^{\log_2 3}}{n} - 2\right) = 3n^{\log_2 3} - 2n$$

Dramatic improvement for large n

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

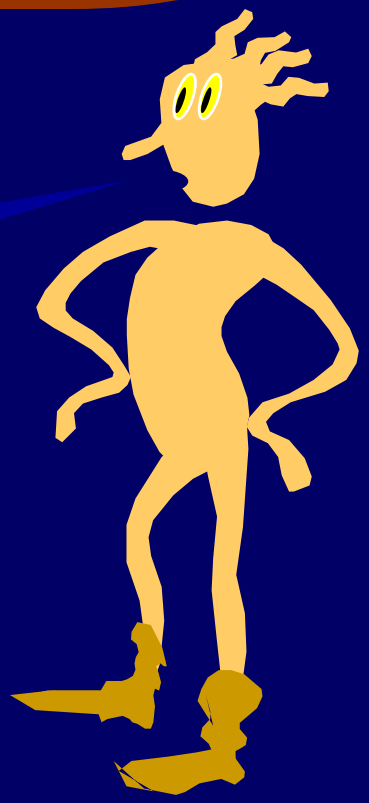
A huge savings over $\Theta(n^2)$ when n gets large.


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

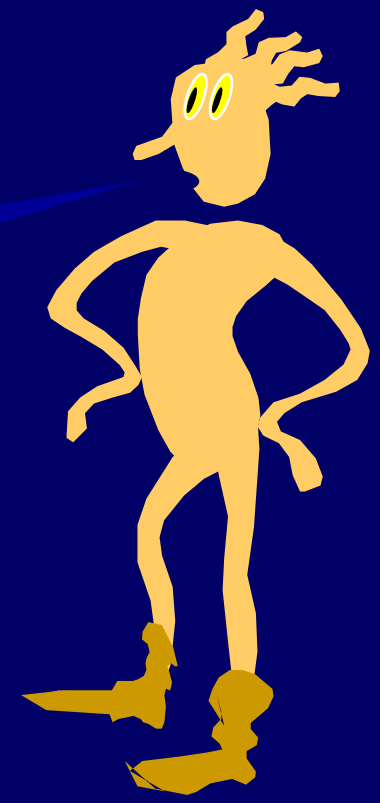


Strange! The Gauss optimization seems to only be worth a 25% speedup. It simply replaces every 4 multiplications with 3. Where did the power come from?

We applied the Gauss optimization RECURSIVELY!



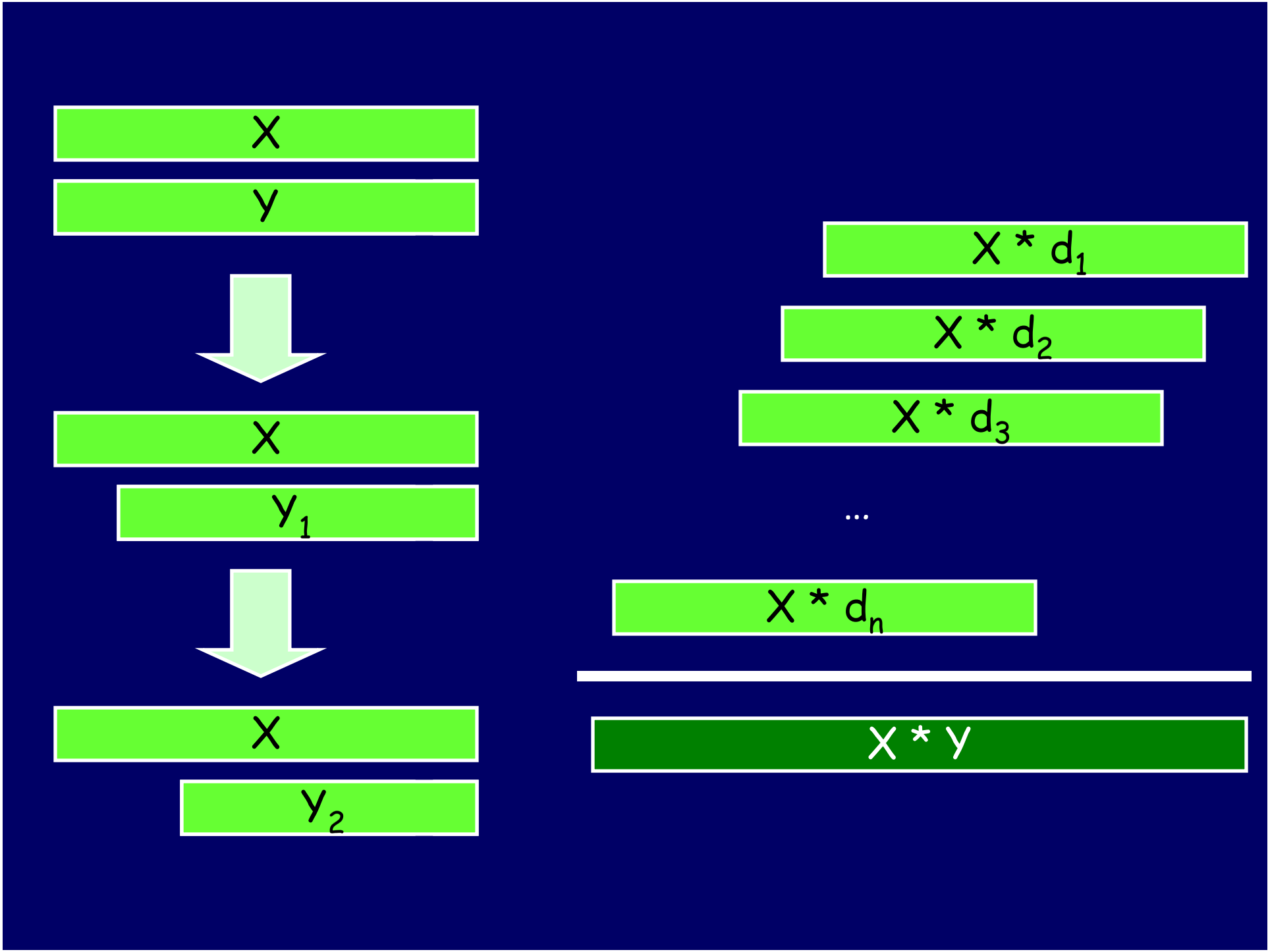

So what is the kissing number of Karatsuba's algorithm?


It's not clear that we kiss even once…

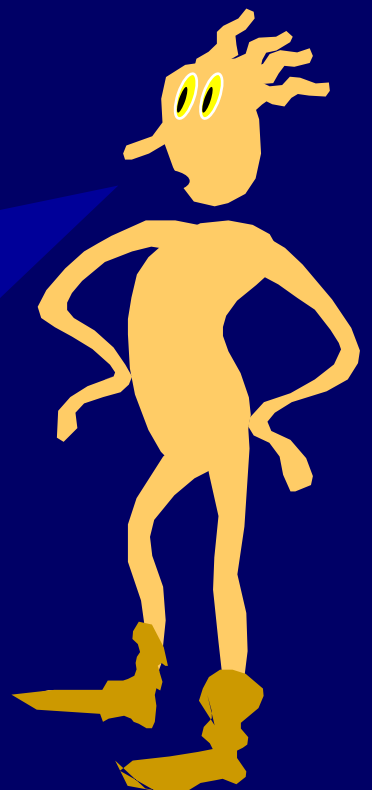
Mystery MULT

```
Mys-MULT(X,Y):
If |Y| = 1 then return X × Y
break Y into c;d where |d| = 1
return
  2[ Mys-MULT(X,c) ] + Mys-MULT(X, d)
```

What's going on here?
Is this an even better way?

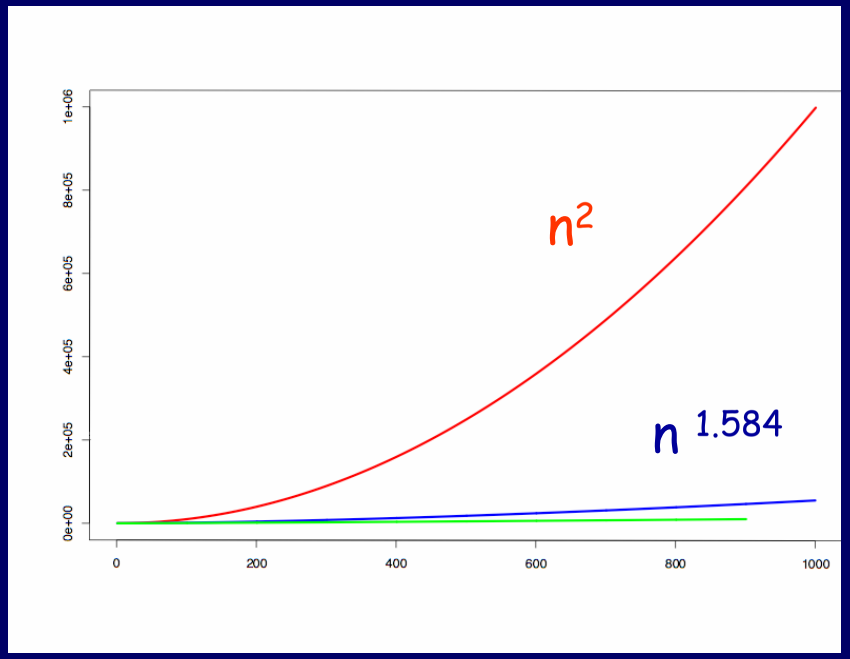

X
y
X
y_1
X
y_2
X * d_1
X * d_2
X * d_3
...
X * d_n
X * y

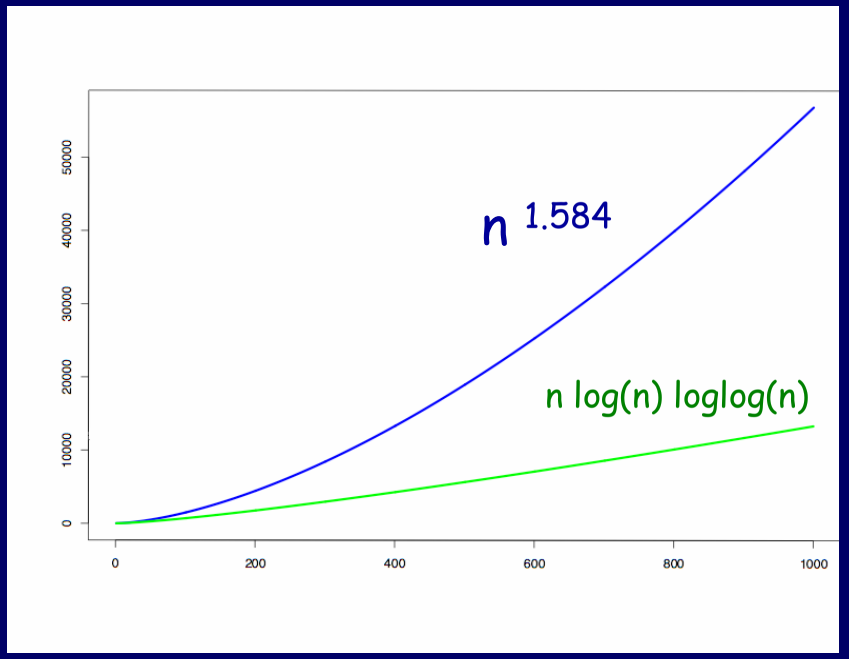
Mys-MULT is the Egyptian method stated in recursive language.



Multiplication Algorithms

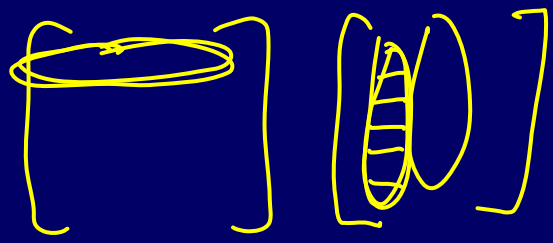
Kindergarten	$n2^n$
Grade School	n^2
Karatsuba	$n^{1.58\ldots}$
Fastest Known	n logn loglogn


0e+00
2e+05
4e+05
6e+05
8e+05
1e+06
0
200
400
600
800
1000
n^2
$n^{1.584}$


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

Matrix multiplication.

$A_{n \times n} \; B_{n \times n} = C_{n \times n}$



$O(n^3)$

$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} \circ & \circ \\ & \end{bmatrix}$

Strassen's algorithm $O(n^{\log_2 7}) \approx O(n^{2.8})$

Coppersmith & Winograd: $O(n^{2.38})$

$O(n^{2+\varepsilon})$

REFERENCES

Karatsuba, A., and Ofman, Y. *Multiplication of multidigit numbers on automata*. Sov. Phys. Dokl. 7 (1962), 595--596.