L2

4102

Recurrences, Karatsuba

Jan 25 2016

shelat

warmup

Simplify $(1 + a + a^2 + \cdots + a^L)(a - 1) =$

$$a + a^2 + a^3 + \cdots + a^{L+1} - a - a^2 - a^3 \cdots - a^L - 1 = a^{L+1} - 1$$

$$\Rightarrow \sum_{i=0}^{L} a^i = \frac{a^{L+1} - 1}{a - 1}$$

warmup

$$\sum_{i=0}^{L} a^i = \frac{a^{L+1} - 1}{a - 1}$$

hw0 submission

https://church.cs.virginia.edu/16s-4102

abhi shelat   Bio   Teaching

16s 4102: Algorithms Submission Page

H0

Due Fri Jan 29, 5pm.
Please submit one file. The written answers should be submitted as a PDF. Any resubmission will overwrite the previous version.

Final pdf to upload:

Choose File no file selected   Submit

You can view your submission here.

512 Rice Hall
Charlottesville, VA 22902
434-243-2145
abhi@virginia.edu

abhvious
abhvious

academic homepage for abhi shelat, associate professor of computer science at u of virginia. I am also the co-founder of a small company Arqball.

1. stand
2. set your “number” to one
3. greet a neighbor (pause if odd person out)
4. if you are older, give your “number” to young and sit
if you are younger, add “numbers”
5. if you are standing & you have a neighbor, goto 3

1. stand
2. set
3. greet
4. sit/add
5. repeat

how fast does it work:

$$T(n)$$

steps to finish in a room of size n

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

$$T(1) = 3$$

1 stand
2 set

3 greet
4 sit/add
5 repeat

how fast does it work:

$$T(n) = 1 + 1 + T(\lceil n/2 \rceil)$$

how can we "solve"

recurrence?

$$\begin{cases} T(n) = T(\lceil n/2 \rceil) + 2 \\ T(1) = 3 \end{cases}$$

Closed form solution

① fine with an "asymptotic bound"

solve a simpler case when n is a power of 2.

$$T(2^k) = 2 + T(2^{k-1})$$

Consider a simpler case.

$= 2 + T(2^{k-2})$

$2 + T(2^{k-3})$

how many 2's involved??

$\ldots$

$2 + T(1)$, $1 = 2^0$, $T(1) = 3$

$= 2k + 3 \quad = \quad T(2^k) = 2\log_2(2^k) + 3$

$$
\begin{aligned}
T(2^k) &= 2 + T(2^{k-1}) \\
&= 2 + 2 + T(2^{k-2}) \\
&= 2 + \overbrace{2 + \cdots + 2}^{k-1} + T(2^0) \\
&= 2k + 3
\end{aligned}
$$

"intuition here"

$$\forall 0 < n < m, T(n) \leq T(m)$$

$$T(m) \leq T(2^{\lceil \log(m) \rceil}) = 2\lceil \log(m) \rceil + 3$$

→ setting $c = 3$

Asymptotic notation

$O(g)$ — set of functions

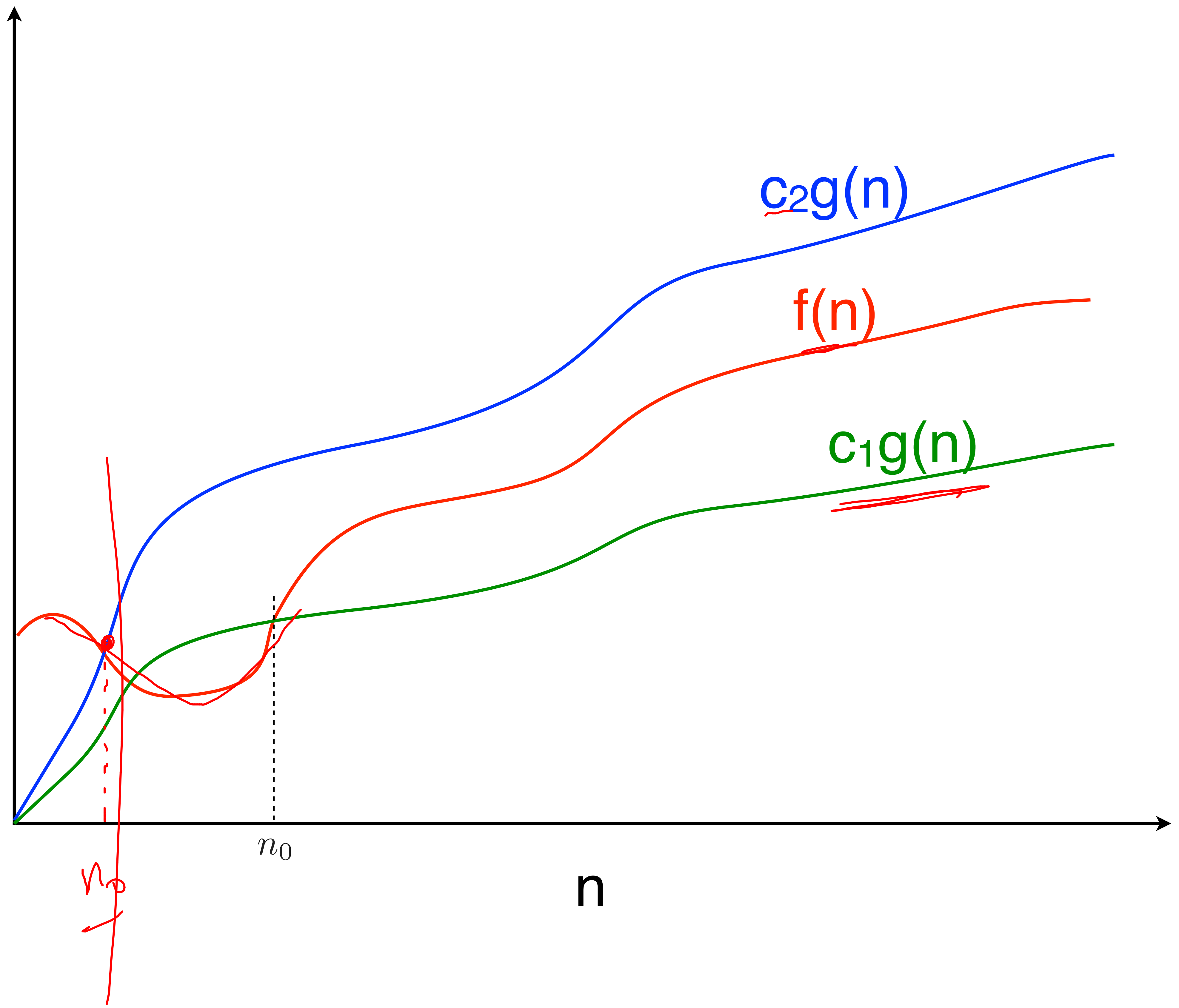
at most within const of g   for large n

$= \{$ functions f : there exist positive constants c, n_0 such that for all $n > n_0$, $0 \le f(n) \le c \cdot g(n) \}$

Asymptotic notation

$O(g)$	at most within const of g	for large n
$\Omega(g)$	at least within const of g (lower bound)	for large n (Omega)
$\Theta(g)$	within a const of g	for large n (Theta)

for all of our algorithms.



∈ "in the set"

$$f(n) = O(g(n))$$

$$f(n) = \Theta(g(n))$$

$$f(n) = \Omega(g(n))$$

$$T(m) \le T(2^{\lceil \log(m) \rceil}) = 2\lceil \log(m) \rceil + 3$$

upper-bound.

$T(m) = O(\log m)$

Ω

we can do this.

make sure you can @ home!!

Θ

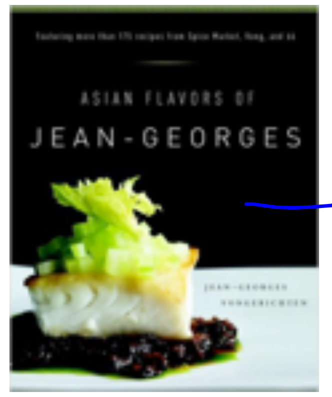
main ideas:

1. break large problem into a smaller one.
2. use recurrence relation to analyze the running time
3. we use asymptotic notation to simplify the analysis

How to solve recurrence relations

→ tree method.

?-✓ guess & check method
(induction)



→ cookbook method "Masters theorem"

http://www.drblank.com/law301.jpg

→ substitution technique.
"change of variable"

Multiplication

$O(n^2)$ operations

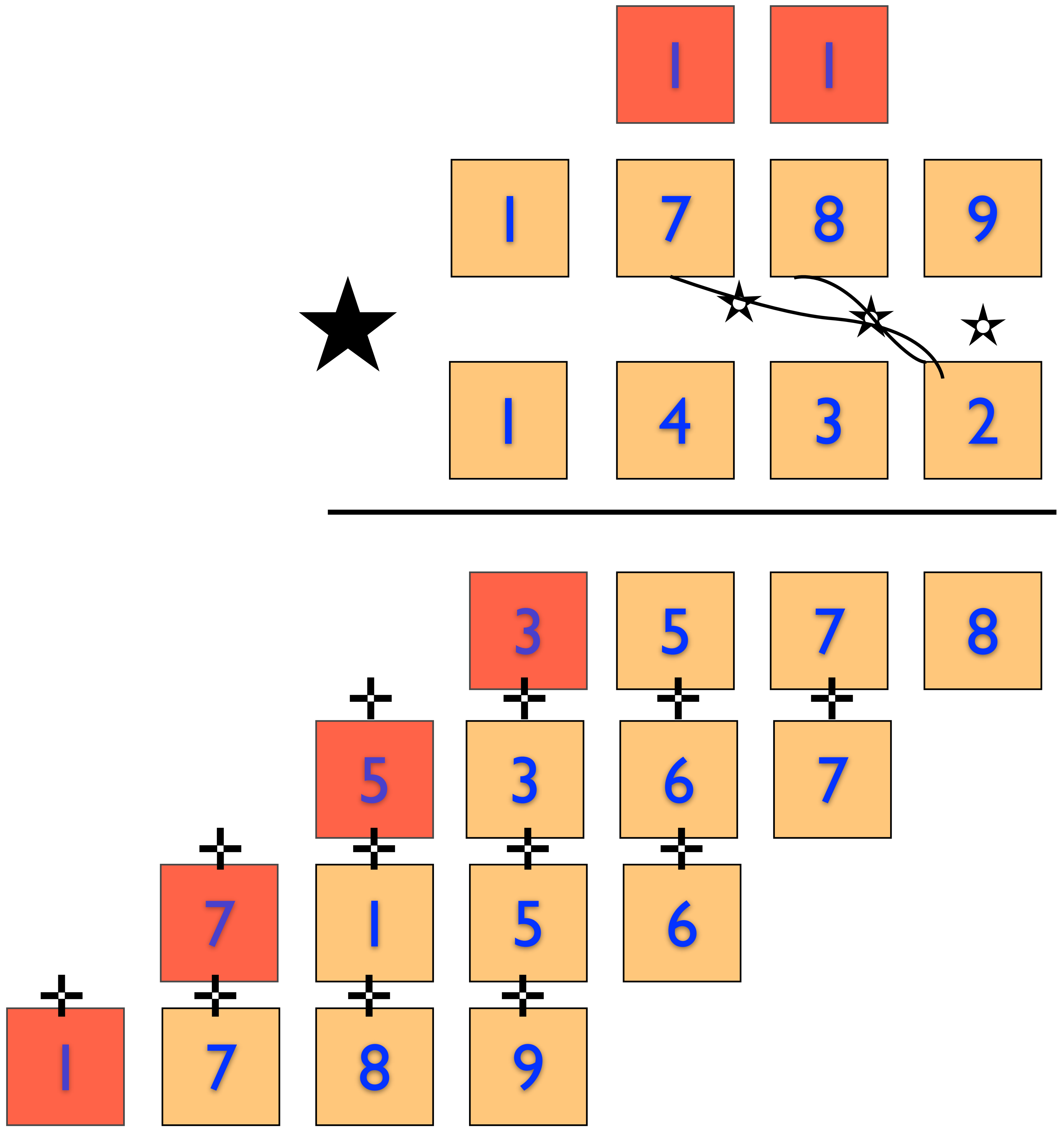
```
     1 7 8 9   ← n-digit numbers
 ×   1 4 3 2
 -----------
     3 5 7 8
   5 3 6 7
 7 1 5 6
1 7 8 9
```

n operations

4 mult.	4 additions
4 mult	4 adds.
4	4
4	4

n rows

several mults



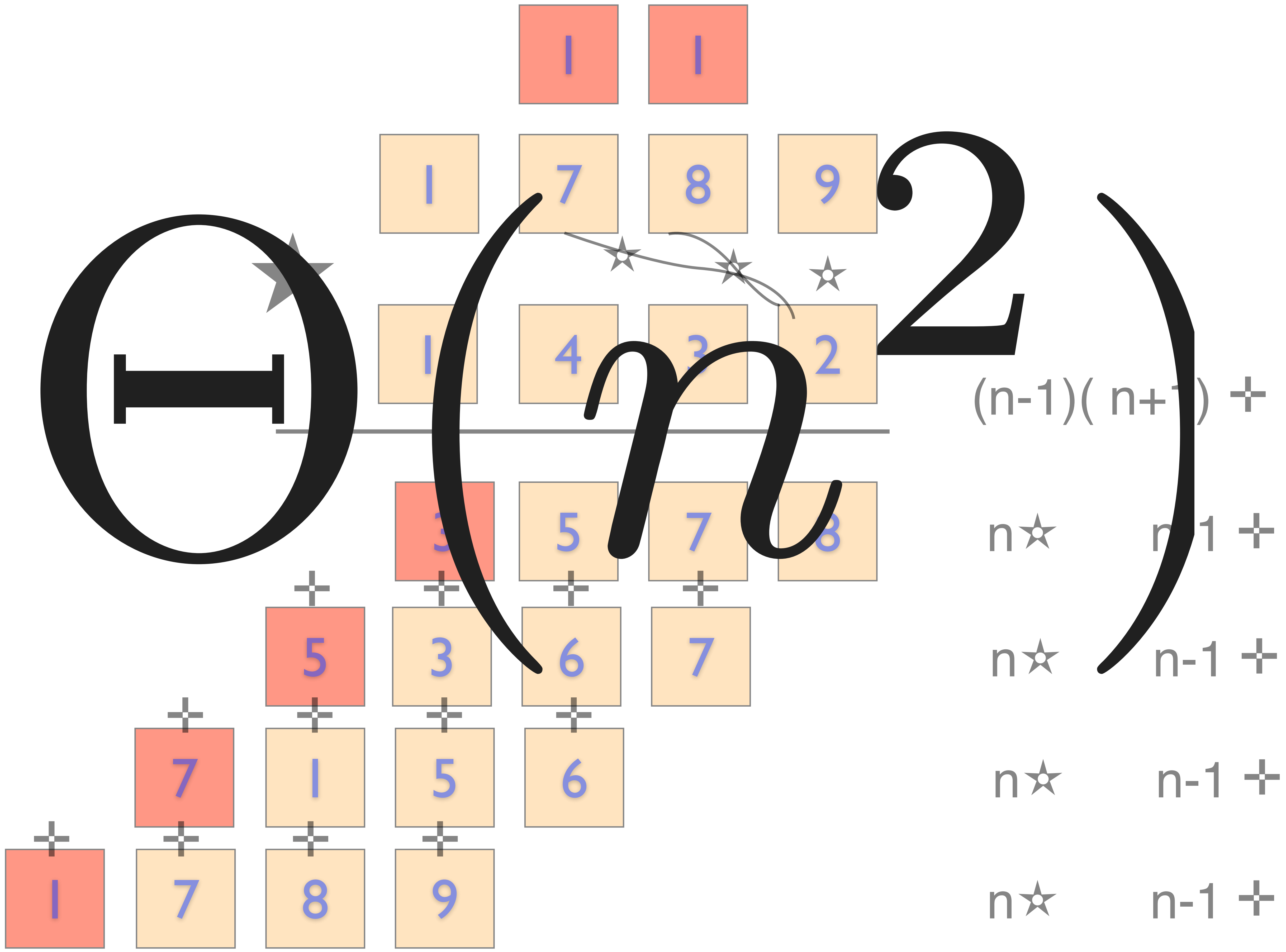
(n-1)(n+1) ✢

n☆ n-1 ✢

n☆ n-1 ✢

n☆ n-1 ✢

n☆ n-1 ✢



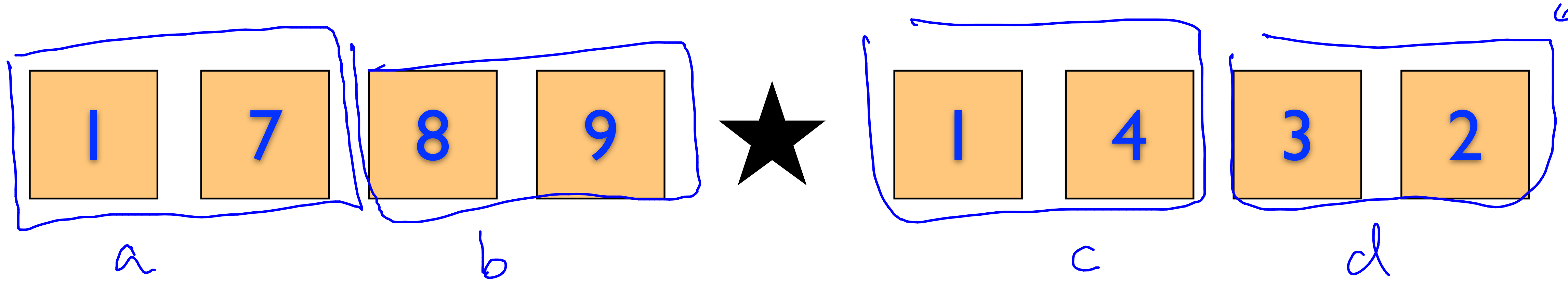
$\Theta(n^2)$
(n-1)(n+1) +
n☆ n-1 +
n☆ n-1 +
n☆ n-1 +
n☆ n-1 +

Theme 1

Break n-digit mult into smaller problems

$(17 \cdot 100 + 89)$ $(14 \cdot 100 + 32)$

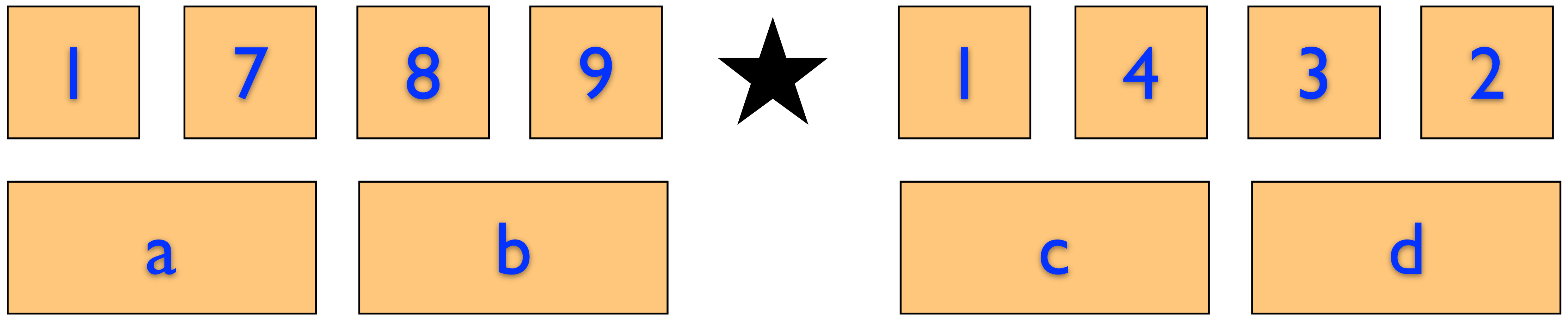
n digit number



a, b, c, d are $\frac{n}{2}$ digit

$$(a \cdot c)(100^2) + (ad + bc)(100) + bd$$

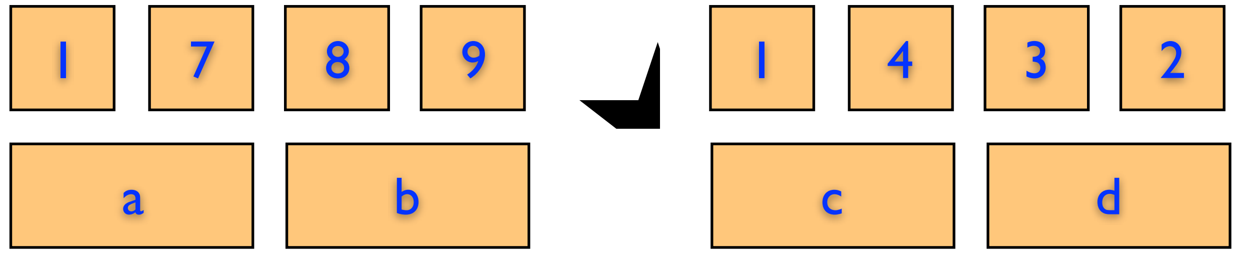
Let's analyze how well this works —



$$ac100^2 + (ad + bc)100 + bd$$

n-digit inputs

Mult(ab, cd)



$$ac100^2 + (ad + bc)100 + bd$$

Base case: return b*d if inputs are 1-digit

else

ac = mult(a, c)

bd = mult(b, d)

ad = mult(a, d)

bc = mult(b, c)

Return $ac \cdot 100^2 + (ad + bc)100 + bd$

Mult(ab, cd) → $T(n)$ running time of Mult on n-digit input

Base case: return b*d if inputs are 1-digit

Compute x = Mult(a,c)
Compute y = Mult(a,d)
Compute z = Mult(b,c)
Compute w = Mult(b,d)

→ $4T(\frac{n}{2})$

Return r = x*100^2 + (y+z)100 + w → $3 \cdot n$ steps (approx)

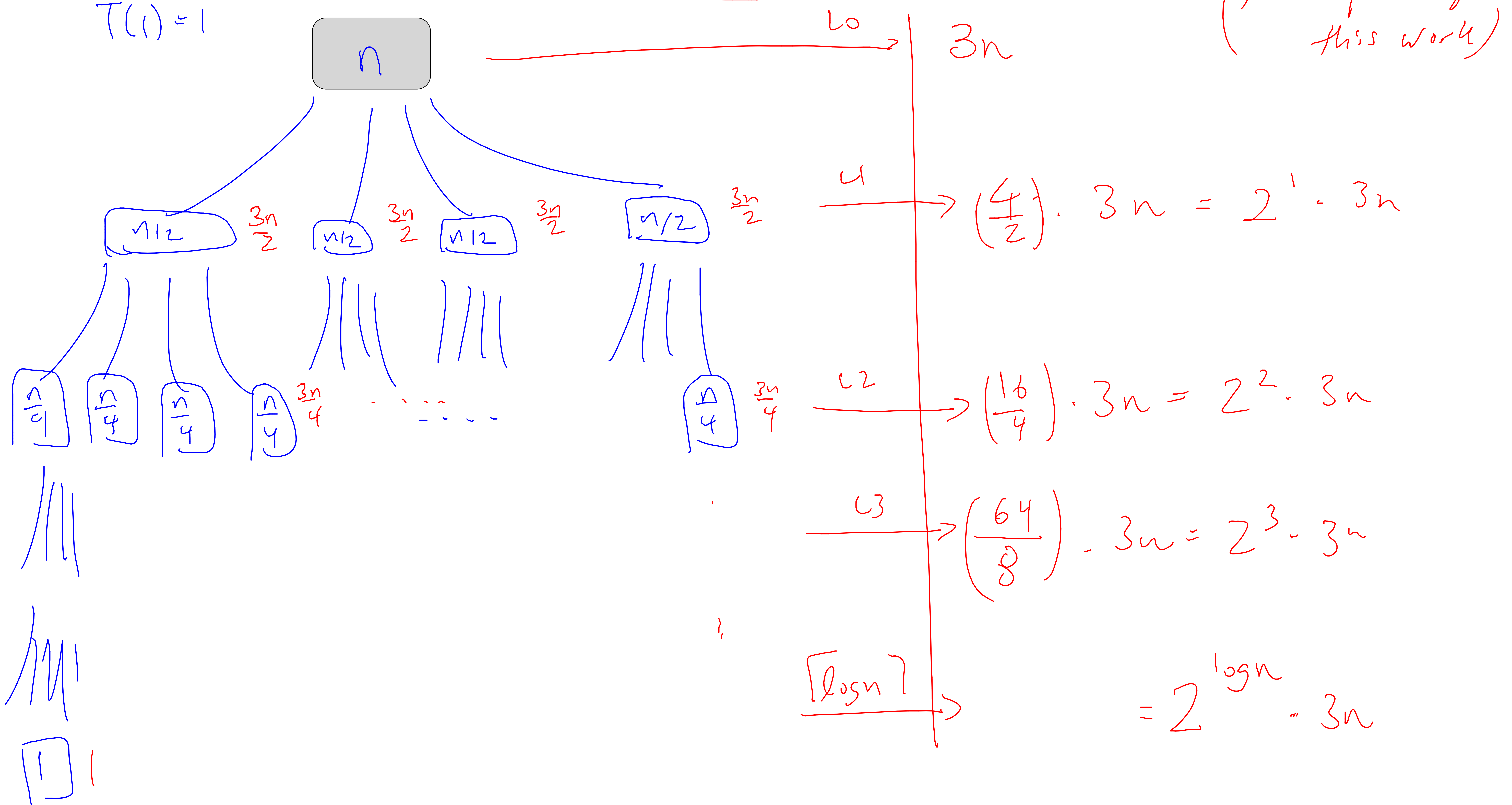
↑ n digits

$$T(n) = 4T(\tfrac{n}{2}) + 3n$$

$$T(n) = 4T(n/2) + 3\Theta(n)$$

tree

(Add up all of this work)



calculations:

$$T(n) = 3n(1 + 2 + 2^2 + 2^3 + \dots 2^{\log n})$$

$$= 3n \cdot \sum_{i=0}^{\log n} 2^i = 3n\left(\frac{2^{\log n + 1} - 1}{1}\right)$$

$$= 3n(2n - 1) = O(n^2)$$

Karatsuba

[a][b] × [c][d]

$(ac)\,100^2 + (ad+bc)\cdot 100 + bd$ ← need this

$a\cdot c$ $\quad$ $b\cdot d$

$(a+b)\cdot(c+d) = ac + ad + bc + bd$ → extra

— But we can subtract these terms off!

Karatsuba

1	7	8	9	★	1	4	3	2
a		b			c		d	

$$ac100^2 + (ad + bc)100 + bd$$

$$(a + b)(c + d) = ac + ad + bc + bd$$

$$ad + bc = (a + b)(c + d) - ac - bd$$

Karatsuba algorithm

1	7	8	9	★	1	4	3	2

a	b		c	d

Recursively compute

❶ $ac, bd, (a+b)(c+d)$

❷ $ad + bc = (a+b)(c+d) - ac - bd$

❸ $ac100^2 + (ad+bc)100 + bd$

Karatsuba(ab, cd)

$T(n)$ = running time of K on n-digits

Base case: return b*d if inputs are 1-digit

ac = Karatsuba(a,c) → $T(\frac{n}{2})$

bd = Karatsuba(b,d) → $T(\frac{n}{2})$

t = Karatsuba((a+b),(c+d)) → $T(\frac{n}{2}+1) \sim T(\frac{n}{2}) + 2n$ (for additions)

mid = t - ac - bd → $2n$

RETURN ac*100^2 + mid*100 + bd → $4n$ work

$$T(n) = 3T(\frac{n}{2}) + 8n$$

Karatsuba(ab, cd)

Base case: return b*d if inputs are 1-digit

ac = Karatsuba(a,c)

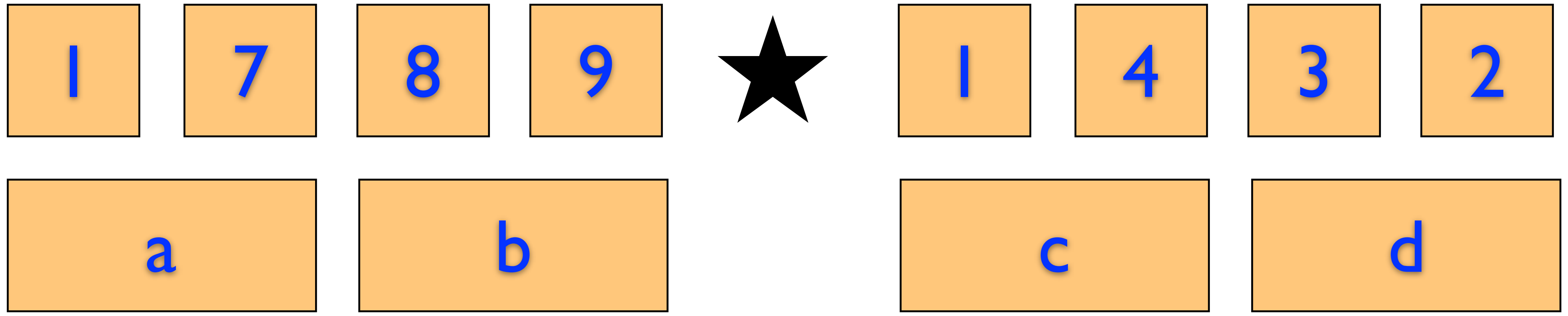
bd = Karatsuba(b,d)

t = Karatsuba((a+b),(c+d)) $3T(n/2) + 2O(n)$

mid = t - ac - bd $2O(n)$

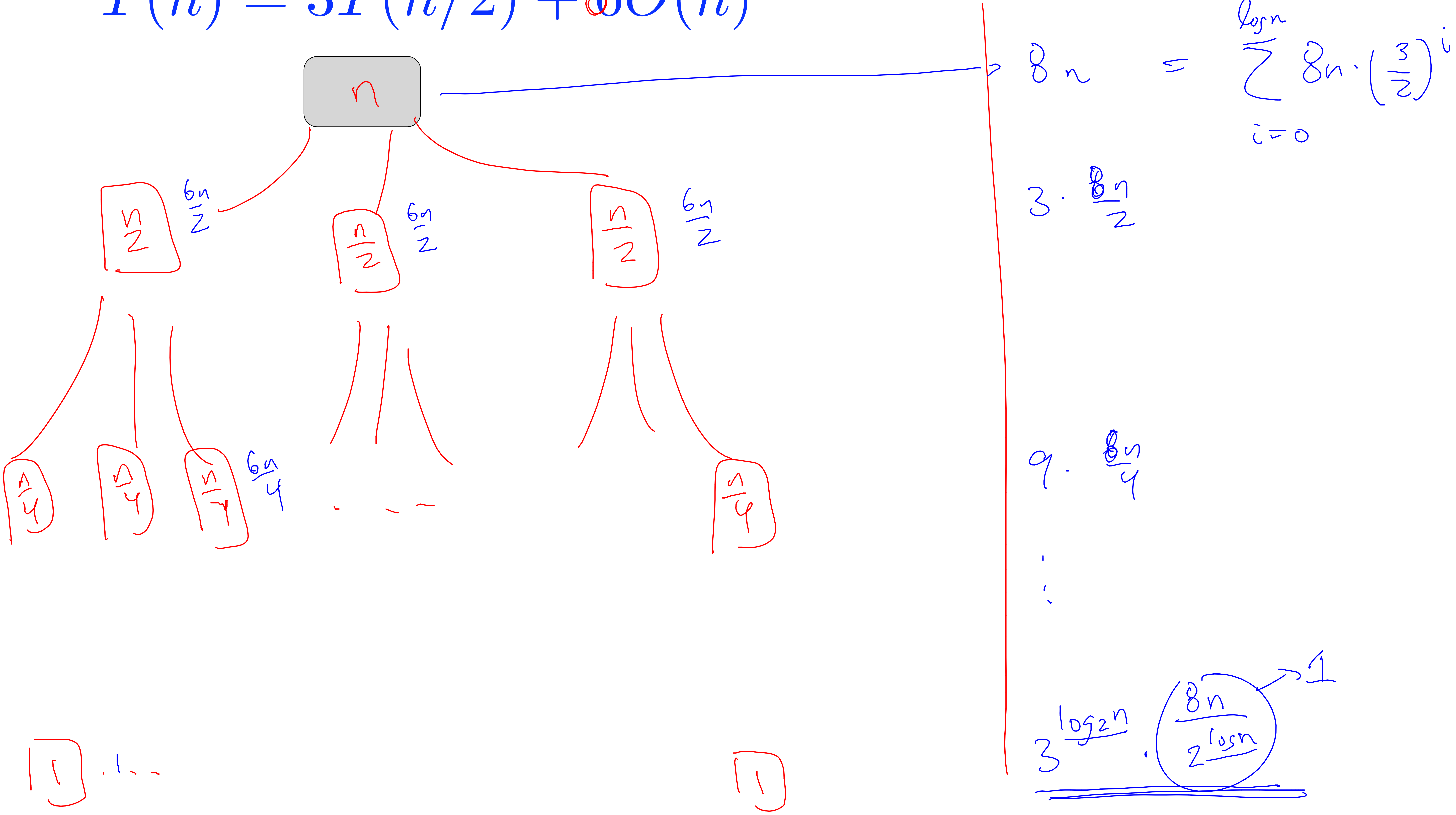
RETURN ac*100^2 + mid*100 + bd $2O(n)$

Karatsuba algorithm



$$T(n) = 3T(n/2) + 6O(n)$$

$$T(n) = 3T(n/2) + 6O(n)$$



calculations:

$$T(n) = 8n \sum_{i=0}^{\log n} \left(\frac{3}{2}\right)^i$$

$$= 8n \cdot \left[\frac{\left(\frac{3}{2}\right)^{\log_2 n + 1} - 1}{\frac{3}{2} - 1} \right]$$

(change of base for logarithm)

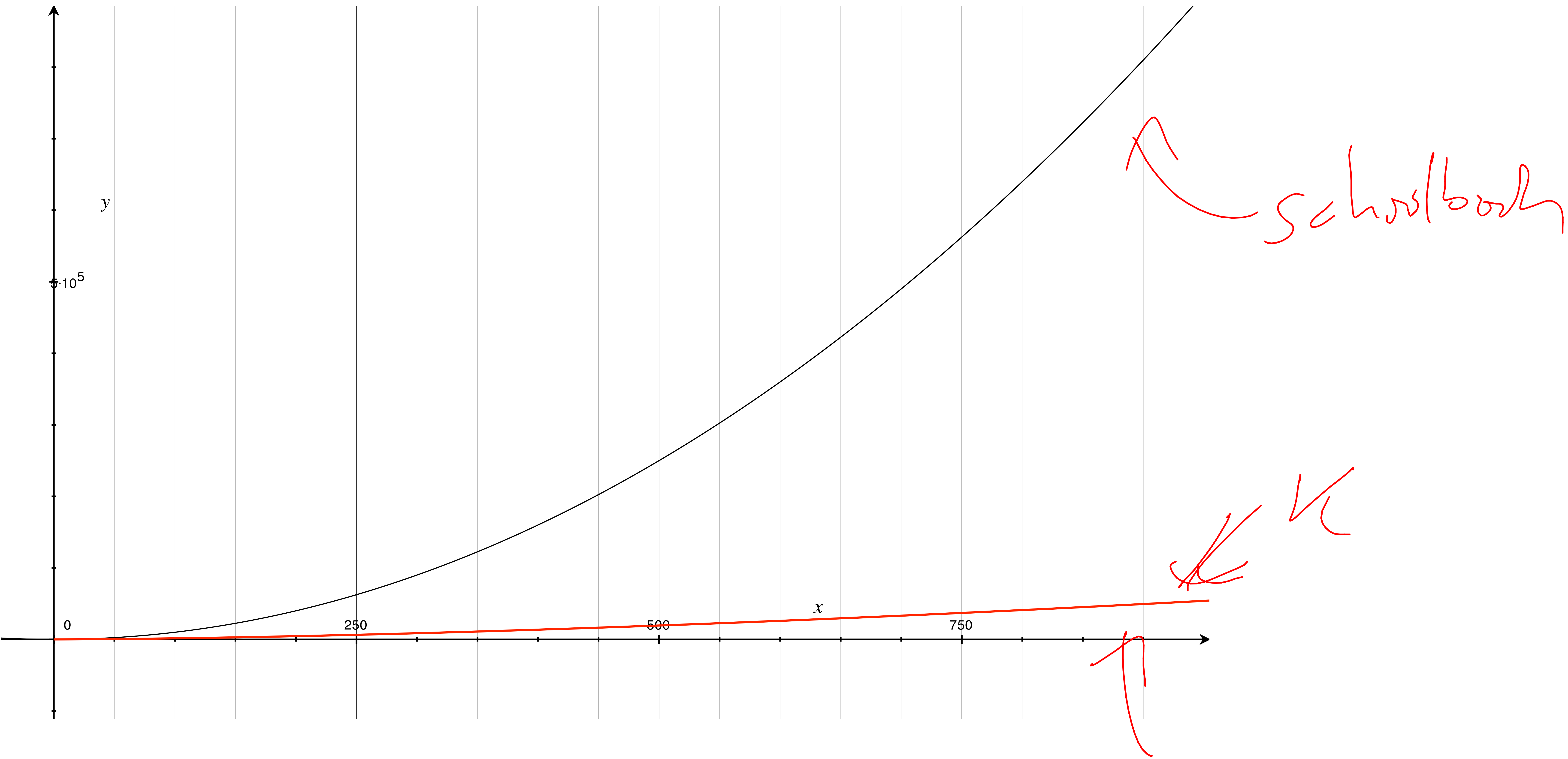
simplify

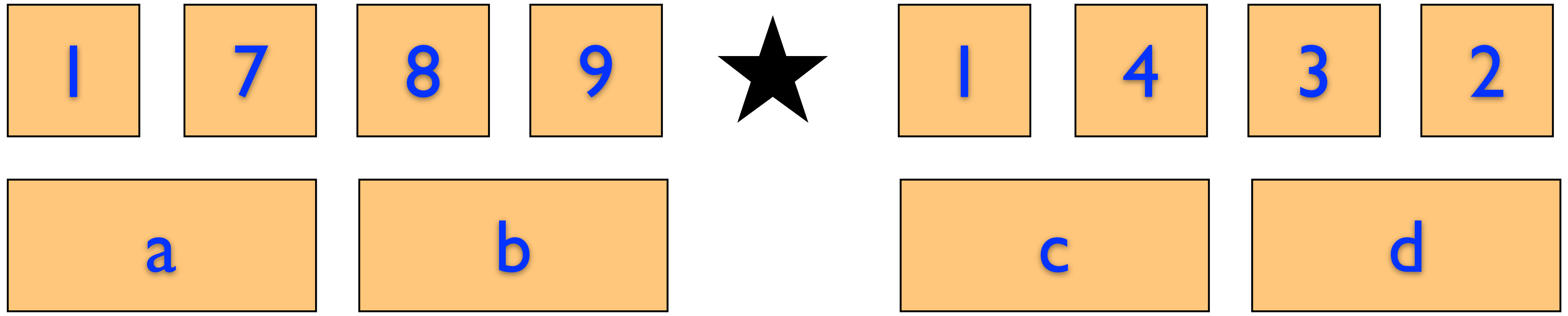
$$= O\left(n^{\log_2 3}\right)$$

calculations:

$$T(n) = 3T(n/2) + 6O(n)$$

$$O(n^{\log_2(3)}) \quad O(n^{1.589})$$


y
5·10^5
0
250
500
750
x
Schulbuch



$$T(n) = 3T(n/2) + 6O(n)$$

$$T(n) = 4T(n/2) + 3O(n)$$