

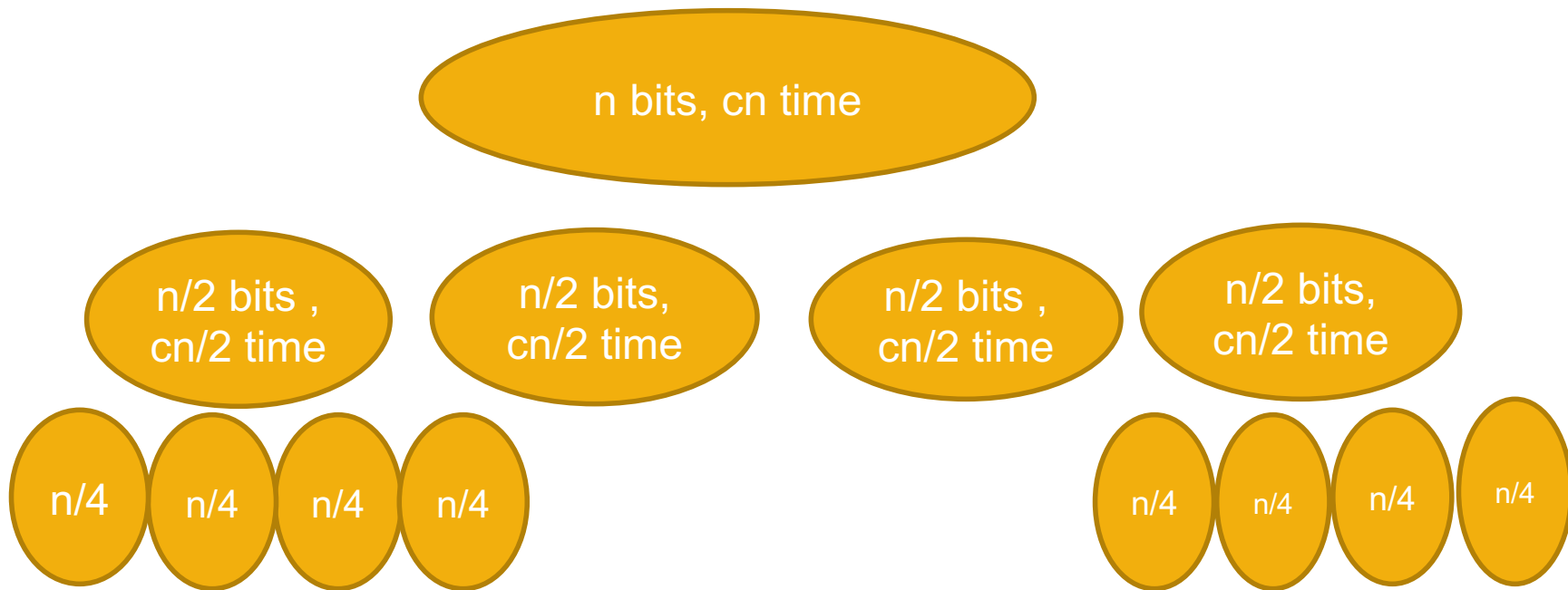
Algorithm multiply

- function **multiply** (x,y):
- *Input*: n-bit integers x and y
- *Output*: the product xy
 - If $n=1$: return xy
 - x_L, x_R and y_L, y_R are the left-most and right-most $n/2$ bits of x and y, respectively.
 - $P_1 = \mathbf{multiply}(x_L, y_L)$
 - $P_2 = \mathbf{multiply}(x_L, y_R)$
 - $P_3 = \mathbf{multiply}(x_R, y_L)$
 - $P_4 = \mathbf{multiply}(x_R, y_R)$
 - return $(P_1 * 2^n + (P_2 + P_3) * 2^{\frac{n}{2}} + P_4)$

Algorithm

- Runtime analysis:
 - Let $T(n)$ be the runtime of the multiply algorithm.
 - Then $T(n) = 4T\left(\frac{n}{2}\right) + O(n)$

Total time



Total

- One top level : cn
- 4 depth 1: $cn/2 * 4$
- 16 depth 2: $cn/4 * 16$
- 64 depth 3: $cn/8 * 64$

....

- 4^t depth t : $\frac{cn}{2^t} * 4^t = cn * 2^t$

....

- Max level : $t = \log n$, $(cn/2^{\log n}) * 4^{\log n} = c * 2^{\log n} * 2^{\log n} = cn^2$

Total time

- $cn (1+ 2 +4 +8+\dots 2^{\log n}) = O(cn^2)$
- Because in a geometric series with ratio other than 1, largest term dominates order.

Multiplication



Andrey Kolmogorov 1903 - 1987



Anatoly Karatsuba 1937 - 2008

**Insight: replace
one (of the 4)
multiplications by
(linear time)
subtraction**

Algorithm multiply KS

- function **multiplyKS** (x,y)
- *Input*: n-bit integers x and y
- *Output*: the product xy
 - If $n=1$: return xy
 - x_L, x_R and y_L, y_R are the left-most and right-most $n/2$ bits of x and y, respectively.
 - $R_1 = \text{multiplyKS}(x_L, y_L)$
 - $R_2 = \text{multiplyKS}(x_R, y_R)$
 - $R_3 = \text{multiplyKS}((x_L + x_R), (y_L + y_R))$
 - return $(R_1 * 2^n + (R_3 - R_1 - R_2) * 2^{\frac{n}{2}} + R_2)$

Correctness multiply KS

- Correctness: by strong induction on n , the number of bits of x and y .
- Base Case: $n = 1$ then return xy (could make a table of possibilities.)
- Inductive hypothesis:

Correctness multiply KS

- Correctness: by strong induction on n , the number of bits of x and y .
- Base Case: $n = 1$ then return xy (could make a table of possibilities.)
- Inductive hypothesis: For some $n > 1$, assume that **multiplyKS**(x, y) returns the correct product xy whenever x has k digits and y has k digits for any $1 \leq k < n$.
- Then by the IH: $R_1 = x_L y_L$, $R_2 = x_R y_R$, $R_3 = (x_L + x_R)(y_L + y_R)$

Correctness multiply KS

- Then by the IH:
 - $R_1 = x_L y_L$, $R_2 = x_R y_R$, $R_3 = (x_L + x_R)(y_L + y_R) = x_L y_L + x_R y_R + x_L y_R + x_R y_L$

- And the algorithm returns: $R_1 * 2^n + (R_3 - R_1 - R_2) * 2^{\frac{n}{2}} + R_2$

$$R_1 * 2^n + (R_3 - R_1 - R_2) * 2^{\frac{n}{2}} + R_2 =$$

$$x_L y_L * 2^n + (x_L y_R + x_R y_L) * 2^{\frac{n}{2}} + x_R y_R =$$

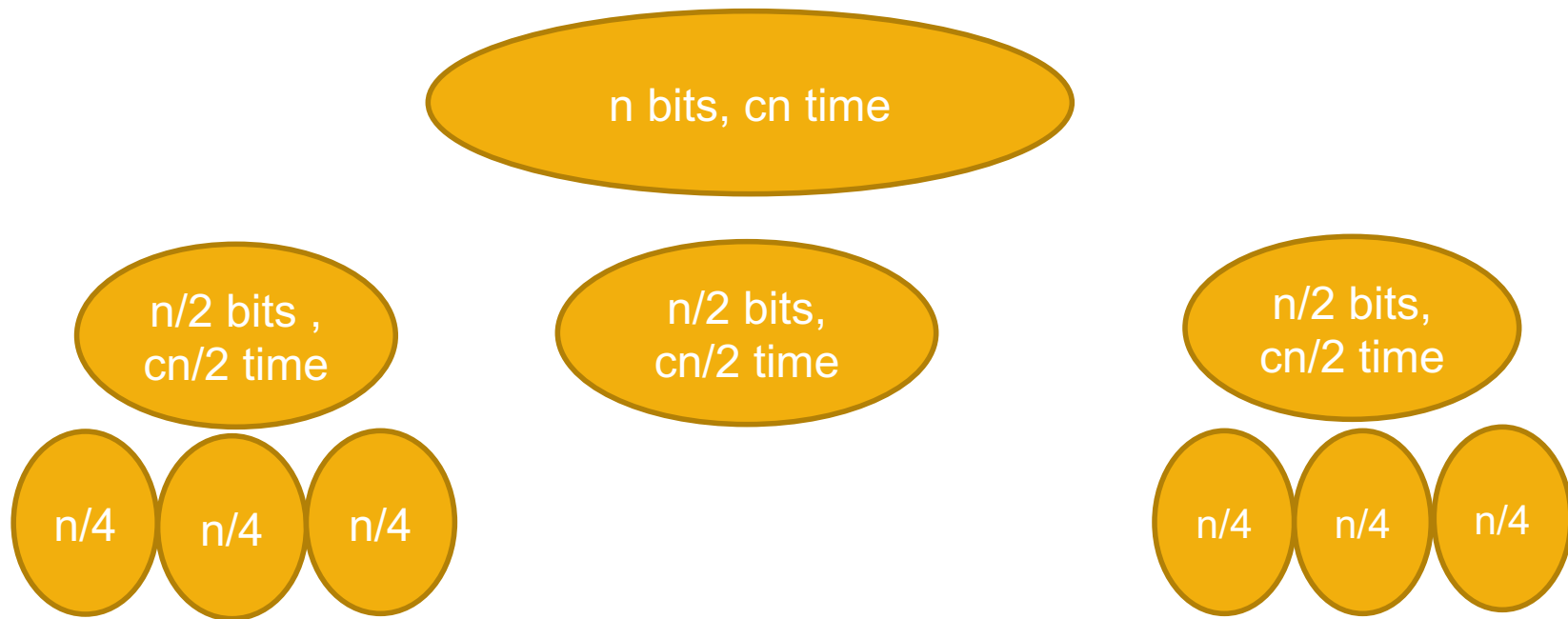
$$\left(x_L * 2^{\frac{n}{2}} + x_R \right) \left(y_L * 2^{\frac{n}{2}} + y_R \right) =$$

$$xy$$

Algorithm multiplyKS

- Runtime
- Let $T(n)$ be the runtime of the multiply algorithm.
- Then
- $T(n) = 3T\left(\frac{n}{2}\right) + O(n)$

Total time



3 vs 4

- Since we are pruning the tree recursively, replacing 4 recursive calls instead of 3 reduces the size of the tree more than a constant factor.

Total

- One top level : cn
- 4 depth 1: $cn/2 * 3$
- 16 depth 2: $cn/4 * 9$
- 64 depth 3: $cn/8 * 27$

....

- 4^t depth t : $\frac{cn}{2^t} * 3^t = cn * (1.5)^t$

....

- Max level : $t = \log n$

Total time

- $cn (1+ 1.5 +2.25 + \dots (1.5)^{\log n}) = O (3^{\log n})$
- Because in a geometric series with ratio other than 1, largest term dominates order.
- But what is $3^{\log n}$?

Simplifying

- $3^{\log n} = (2^{\log 3})^{\log n} = 2^{\{\log n * \log 3\}} = (2^{\{\log n\}})^{\log 3} = n^{\log 3} = n^{\{1.58...\}}$
- So total time is $O(n^{\log 3})$