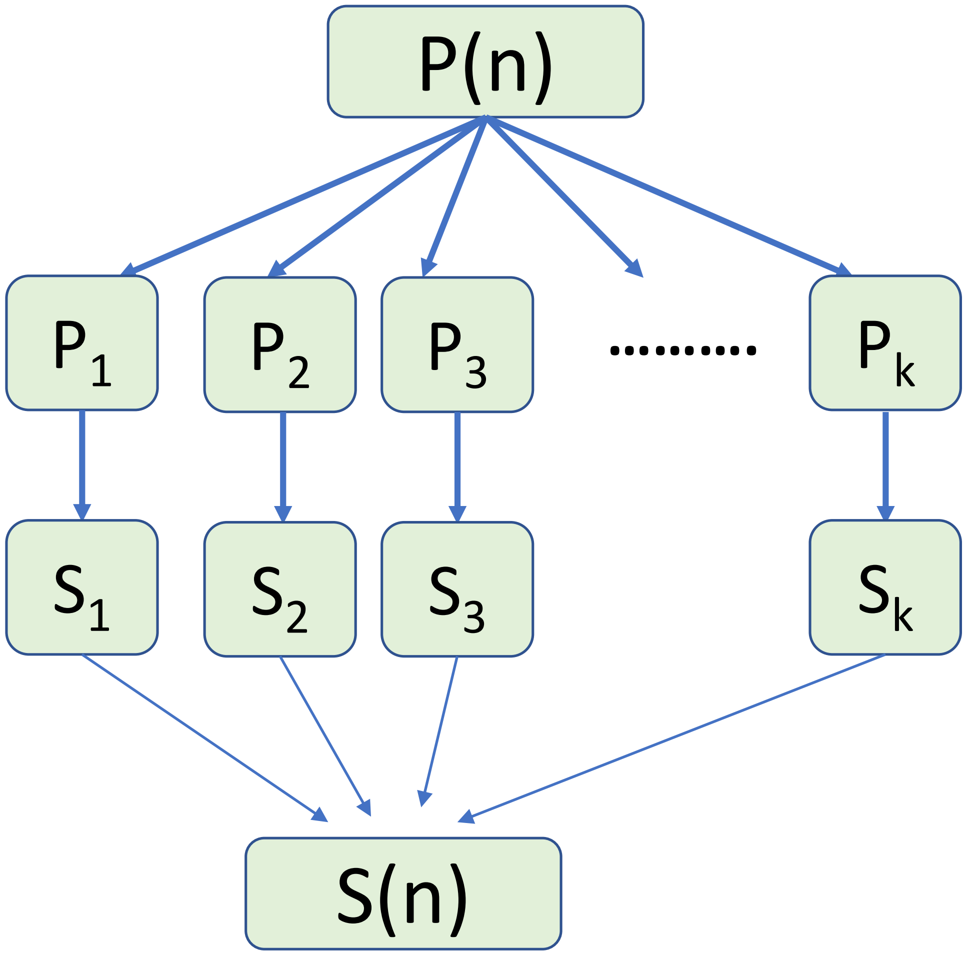
# CIS 675
# Recurrences

Dr. Asif Salekin

# Divide & Conquer




## Recursive in nature

- **Smaller sub-problems will be same as the problem P(n)**
  **You can not transform them into another problem.**

Example:

If P(n) is sorting an array of size n. The sub-problems can only be sorting an array of size m, where m<n.

- **Need a strategy to combine the solutions of the subproblems.**

# Warmup!

$(1+a+a^2+\ldots\ldots.+a^L)(a-1) \quad = a+a^2+a^3+\ldots\ldots +a^L +a^{L+1}$

$-1-a-a^2-a^3- \ldots\ldots -a^L$

$= a^{L+1}-1$

$(1+a+a^2+\ldots\ldots.+a^L) = \dfrac{a^{L+1-1}}{a-1}$

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

# Who has smallest SU ID in the first row?

When we can not memorize any number!


1. Stand

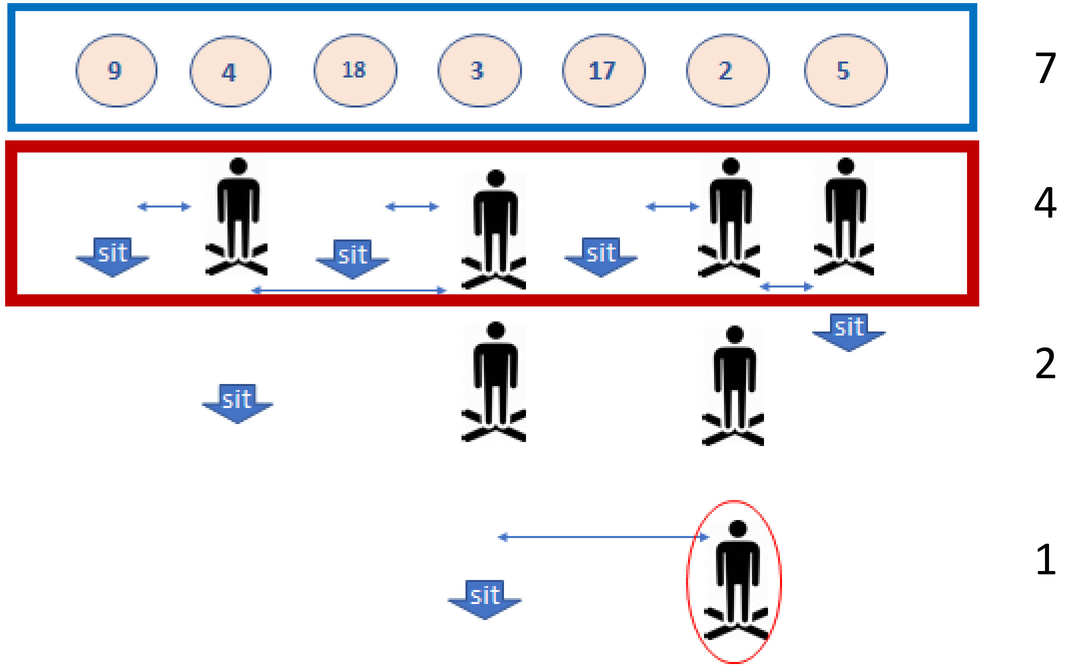
2. Greet a neighbor (stop if you are the only person standing) (do not communicate with more than one!)

3. If you have larger SU ID sit
If you have smaller SU ID compared to your neighbor's, remain standing

4. If you are standing & you have neighbor, go to 2

# How fast does it work?

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| | **Stand** | **Greet** | **Sit/stay** | **repeat** |
| T(n)= | | 1 | 1 | T ($\lceil n/2 \rceil$) |




**T(n) is a function: steps to finish in a room of n students**

**$T(n)= 1+ 1 +T( \lceil n/2 \rceil )$**
**& $T(1)=3$**

**How can we solve it?**

# Recurrence?

$T(n) = 2 + T(\lceil n/2 \rceil)$
& $T(1)=3$

- We are not going for exact solution!
- We will find an asymptotic bound!

# Solve a simpler case when n is power of 2!

$T(n) = 2 + T(\lceil n/2 \rceil)$
& $T(1)=3$

$$
\begin{aligned}
T(2^K) = 2 &+ T(2^{K-1}) \\
&+ 2 + T(2^{K-2}) \\
&+ 2 + T(2^{K-3}) \\
&+ 2 + T(2^{K-4}) \\
&\cdots \\
&+ 2 + T(1) \Rightarrow 3
\end{aligned}
$$

How many 2's involve?

$2^{K-K} = 2^0 = 1$

$T(2^K) = 2K+3 = 2\log_2(2^K)+3$

# Solve a simpler case when n is power of 2!

$T(n) = 2 + T(\lceil n/2 \rceil)$
$\&\ T(1)=3$

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

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

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

# Asymptotic notation

Set of functions

f(X)=O(g(X)) at most within cost of g for large n

$$\left\{ \begin{array}{l} \textbf{function f: there exist positive constants c, } n_0 \textbf{ such that} \\ \qquad \textbf{for all } n>n_0,\ 0 \le f(n) \le c \times g(n) \end{array} \right\}$$

**Important note:**
**O(g(X)) is actually a set!**

**When we say "f(x)=O(g(x)), we are actually Saying that f(X) $\in$ O(g(x))**

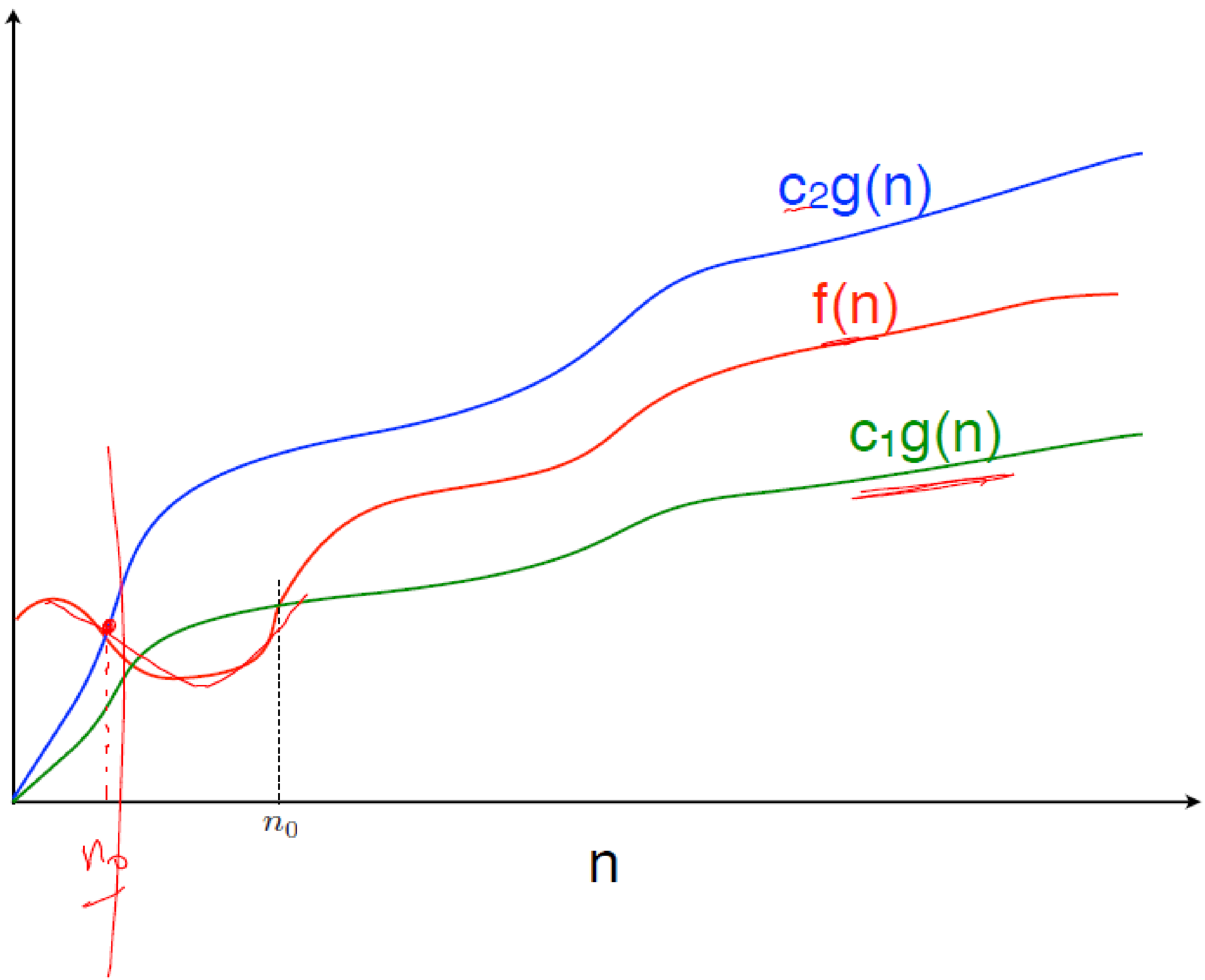
# Asymptotic notation

O(g) at most within const of g for large n

Ω(g) at least within const of g for large n

Θ(g) within const of g for large n

# Asymptotic notation




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

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

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

# Asymptotic notation

$\Theta(g)$ ***For all of our algorithms we want to proof the theta bound***

$\Theta(g(n)) = \{f(n)$: there exist positive constants $c_1$, $c_2$ and $n_0$

such that $0 \leq c_1 \times g(n) \leq f(n) \leq c_2 \times g(n)$ for all $n \geq n_0\}$

The above definition means, if f(n) is theta of g(n), then the value f(n) is always between $c_1 \times g(n)$ and $c_2 \times g(n)$ for large values of n ($n \geq n_0$).

# Asymptotic notation

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

**Upper bound**

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

**$\Omega(m)$ ?????**

# Main Ideas

- Break large problem into smaller ones.
- Use recurrence relation to analyze the running time.
- We use asymptotic notation to simplify the analysis.

# How to solve recurrence relations?

- Tree method
- Guess & check method (induction)
- Cookbook method “Master Theorem”
- Substitution Technique

# Multiplication

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | 1 | 7 | 8 | 9 |
| | | × | 1 | 4 | 3 | 2 |
| | | | 3 | 5 | 7 | 8 |
| | | 5 | 3 | 6 | 7 | |
| | 7 | 1 | 5 | 6 | | |
| 1 | 7 | 8 | 9 | | | |

$O(n^2)$

**N distinct numbers**

**2n operations**

**4 multiplication** **4 addition**

**4 multiplication** **4 addition**

**4 multiplication** **4 addition**

**4 multiplication** **4 addition**

**n rows**

# Main Ideas

- Break large problem into smaller ones.
- Use recurrence relation to analyze the running time.
- We use asymptotic notation to simplify the analysis.

# Multiplication

$17 \times 100 + 89$ $\qquad$ $14 \times 100 + 32$

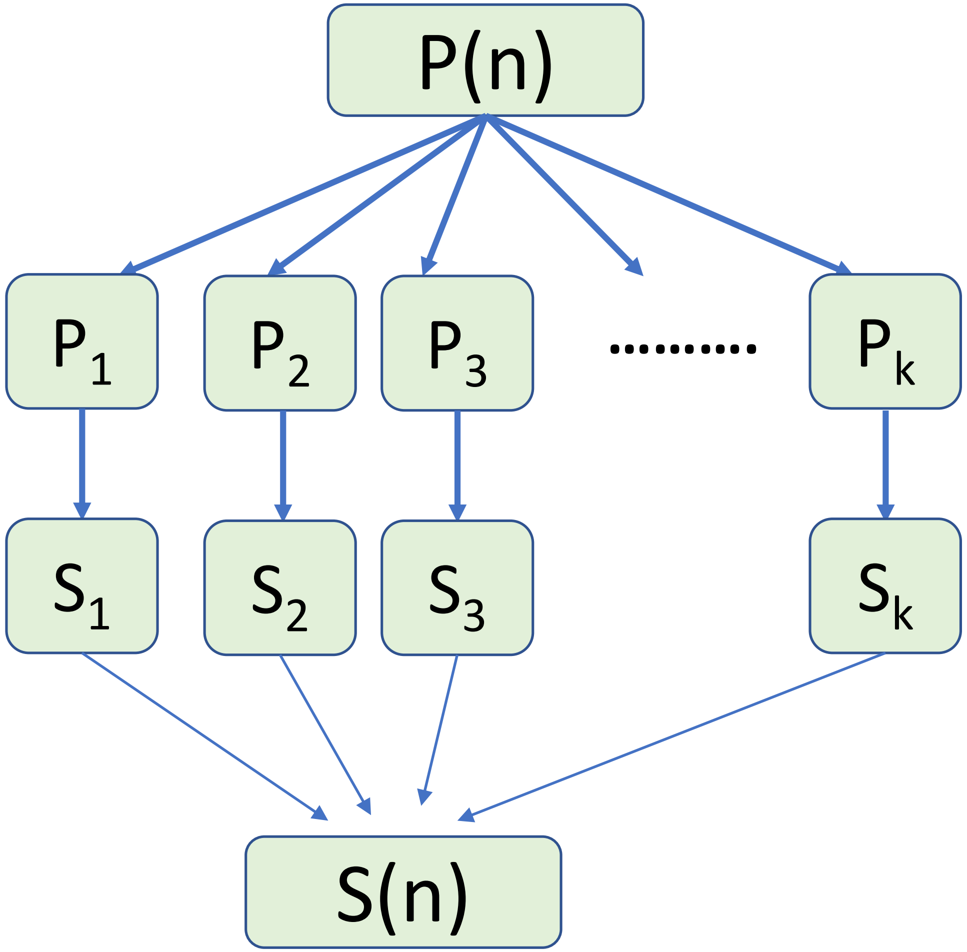
$$\underbrace{1\;7}_{a}\;\underbrace{8\;9}_{b} \times \underbrace{1\;4}_{c}\;\underbrace{3\;2}_{d}$$

**n digit**

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

$$(a\times c)(100^2) + (a\times d + b\times c)(100) + b\times d$$

# Divide & Conquer




## Recursive in nature

- **Smaller sub-problems will be same as the problem P(n)**
  **You can not transform them into another problem.**

Example:
If P(n) is sorting an array of size n. The sub-problems can only be sorting an array of size m, where m<n.

- **Need a strategy to combine the solutions of the subproblems.**

# Multiplication

$17 \times 100 + 89$ $\quad$ $14 \times 100 + 32$

**n digit**

$$\underbrace{\boxed{\mathbf{1}}\ \boxed{\mathbf{7}}}_{\mathbf{a}}\ \underbrace{\boxed{\mathbf{8}}\ \boxed{\mathbf{9}}}_{\mathbf{b}}\ \times\ \underbrace{\boxed{\mathbf{1}}\ \boxed{\mathbf{4}}}_{\mathbf{c}}\ \underbrace{\boxed{\mathbf{3}}\ \boxed{\mathbf{2}}}_{\mathbf{d}}$$

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

$$(a\times c)(100^2) + (a\times d + b\times c)\ (100) + b\times d$$

Let’s analyze how well it works!

# Multiplication

## Mult(ab,cd)

| 1 | 7 | 8 | 9 | × | 1 | 4 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|

$\underbrace{17}_{a}\ \underbrace{89}_{b} \times \underbrace{14}_{c}\ \underbrace{32}_{d}$

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

**BASE CASE:**

**return b × d if inputs are 1 digit**

**ELSE:**

**Compute X=mult(a,c)** → $T(\frac{n}{2})$

**Compute Y=mult(a,d)** → $T(\frac{n}{2})$

**Compute Z=mult(b,c)** → $T(\frac{n}{2})$

**Compute W=mult(b,d)** → $T(\frac{n}{2})$

$4 \times T(\frac{n}{2})$

$T(n) = 4T(\frac{n}{2}) + 6n$

**Return X × ((10)$^{(\text{number of digit of a})}$)$^2$ +(Y+Z)×(10)$^{(\text{number of digit of a})}$ +W**

6n steps

Why 6n?

- In the calculation of $X \times ((10)^{(\text{number of digit of a})})^2$ I am considering two big-wise shifts. That's why there are 2n. But, you can consider it as one bit-wise shift, in that case it will be n operations.
- (Y+Z) is n operation.
- $(Y+Z)\times(10)^{(\text{number of digit of a})}$ Now, multiplying with the $(10)^{(\text{number of digit of a})}$ term is one bit-wise shift. So n operations.
- At the end two addition operation left. Two addition means 2n operation.
- That's why there are 6n operation.
- Now, we are more concern about asymptotic analysis of big O, right? So, 5n, 6n or 7n, it does not matter in the proof. We will still end up with $O(n^2)$
- Note: these are all approximation, we are ignoring carry too, so actually term will not be 6n or 5n.

| 1 | 7 | 8 | 9 | × | 1 | 4 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|

$$\underbrace{17}_{a}\ \underbrace{89}_{b} \times \underbrace{14}_{c}\ \underbrace{32}_{d}$$

$$ac\times100^2+(ad+bc)\times100+bd$$

$$T(n)=4T(\tfrac{n}{2}) +6n$$

$$T(\tfrac{n}{2})$$

$$T(\tfrac{n}{2})$$

$$T(\tfrac{n}{2})$$

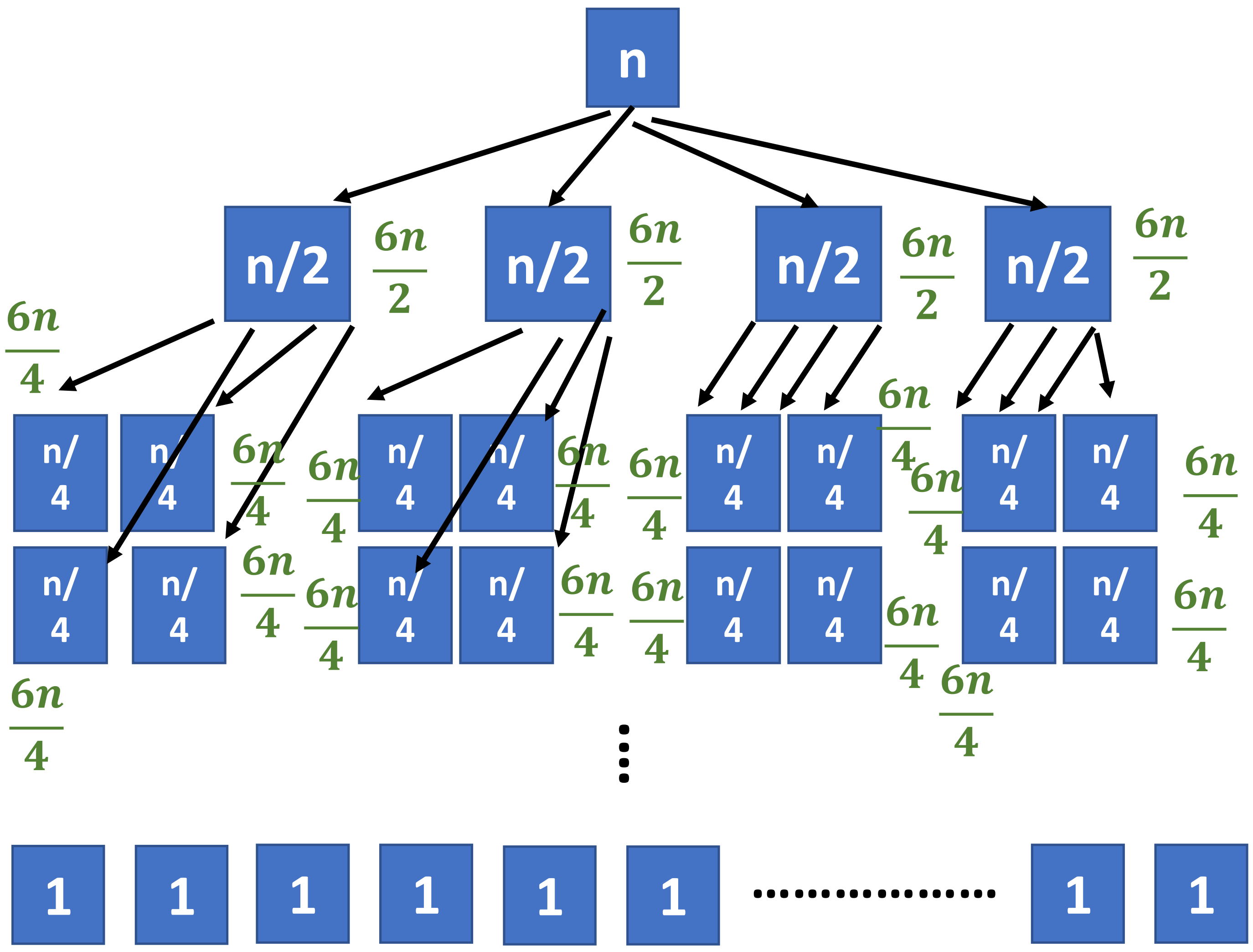
$$T(\tfrac{n}{2})$$

$$4 \times T(\tfrac{n}{2})$$

6n steps

**Return** $X \times ((10)^{(\text{number of digit of a})})^2 + (Y+Z)\times(10)^{(\text{number of digit of a})} + W$

# $T(n)=4T(\frac{n}{2})+6n$ , base case: T(1)=1




| | |
|---|---|
| 0 | 6n |
| 1 | $4\times\frac{6n}{2} = 2^1\times 6n$ |
| 2 | $16\times\frac{6n}{4} = 2^2\times 6n$ |
| 3 | $64\times\frac{6n}{8} = 2^3\times 6n$ |
| $\lceil \log n \rceil$ | $2^{\lceil \log n \rceil}\times 6n$ |

**Calculations:**

$$(1+a+a^2+\ldots\ldots+a^L) = \frac{a^{L+1-1}}{a-1}$$

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

$$T(n) = 6n + 6n\times 2 + 6n\times 2^2 + \ldots\ldots\ldots 6n\times 2^{\lceil \log n \rceil}$$

$$= 6n \times (1 + 2 + 2^2 + \ldots\ldots\ldots + 2^{\lceil \log n \rceil})$$

**n**

$$= \mathbf{6n} \times \left[ \frac{2^{1+\lceil \log n \rceil} - 1}{2-1} \right] = 6n \times [2 \times 2^{\lceil \log n \rceil} - 1]$$

$$= 6n[2n-1]$$

$$= 12n^2 - 6n$$

$$= O(n^2)$$

Now what we have concluded in slide 31 of lecture 3?

- Only pay attention to the dominant terms ($n^2$ more important than n)
- Don't include constants in your big-O expression.
- That's why we are ignoring the term 6n. According to the definition of big O, constant term c=12. and our big O is $O(n^2)$

# Karatsuba

$$\underbrace{1\ 7}_{a}\ \underbrace{8\ 9}_{b} \times \underbrace{1\ 4}_{c}\ \underbrace{3\ 2}_{d}$$

$$(a\times c)(100^2) + (a\times d + b\times c)(100) + b\times d$$

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

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

# Karatsuba

$(a\times c)(100^2) + (a\times d + b\times c)(100) + b\times d$

$$\underbrace{1\ 7}_{a}\ \underbrace{8\ 9}_{b} \times \underbrace{1\ 4}_{c}\ \underbrace{3\ 2}_{d}$$

n/2 n/2 n/2 n/2

**Recursively compute:**

1. ac,bd,(a+b)(c+d)
2. ad+bc=(a+b)(c+d) –ac-bd
3. ac×100$^2$ + (ad+bc) ×100 +bd

$3T(\frac{n}{2})$

2 addition,

4n subtraction

4n addition

**Approximately, Not exactly**

# Karatsuba (ab,cd)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 8 | 9 | × | 1 | 4 | 3 | 2 |

$\underbrace{1\ 7}_{a}\ \underbrace{8\ 9}_{b} \times \underbrace{1\ 4}_{c}\ \underbrace{3\ 2}_{d}$

**BASE CASE:**

**return b × d if inputs are 1 digit**

**ELSE:**

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

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

**Compute t=karatsuba((a+b),(c+d))** → $T(\frac{n}{2}) + 2n$

**mid=t-ac-bd** → 3n

$T(n)=3T(\frac{n}{2}) + 9n$

**Ignoring issue of carries**

**Return ac × ((10)$^{\text{(number of digit of a)}}$)$^2$ +mid×(10)$^{\text{(number of digit of a)}}$ +bd** $\Big\}$ **4n steps**

# Karatsuba (ab,cd)

1 7 8 9 × 1 4 3 2

a b c d

In lecture 3, we consider 2n operations.
**Why?**
There are 2 subtractions. Hence 2n operation.
**Now, why I changed 9n to 8n for the same proof, in Lecture 3?**
To emphasize and show that, the constant operations are approximations. Your assumption on the constant number of arithmetic operations, can be different, but in asymptotic analysis you will still end up with same big O, or theta or omega.

$T(\frac{n}{2})$

$T(\frac{n}{2})$

))

$T(\frac{n}{2}) + 2n$

**mid=t-ac-bd**

3n

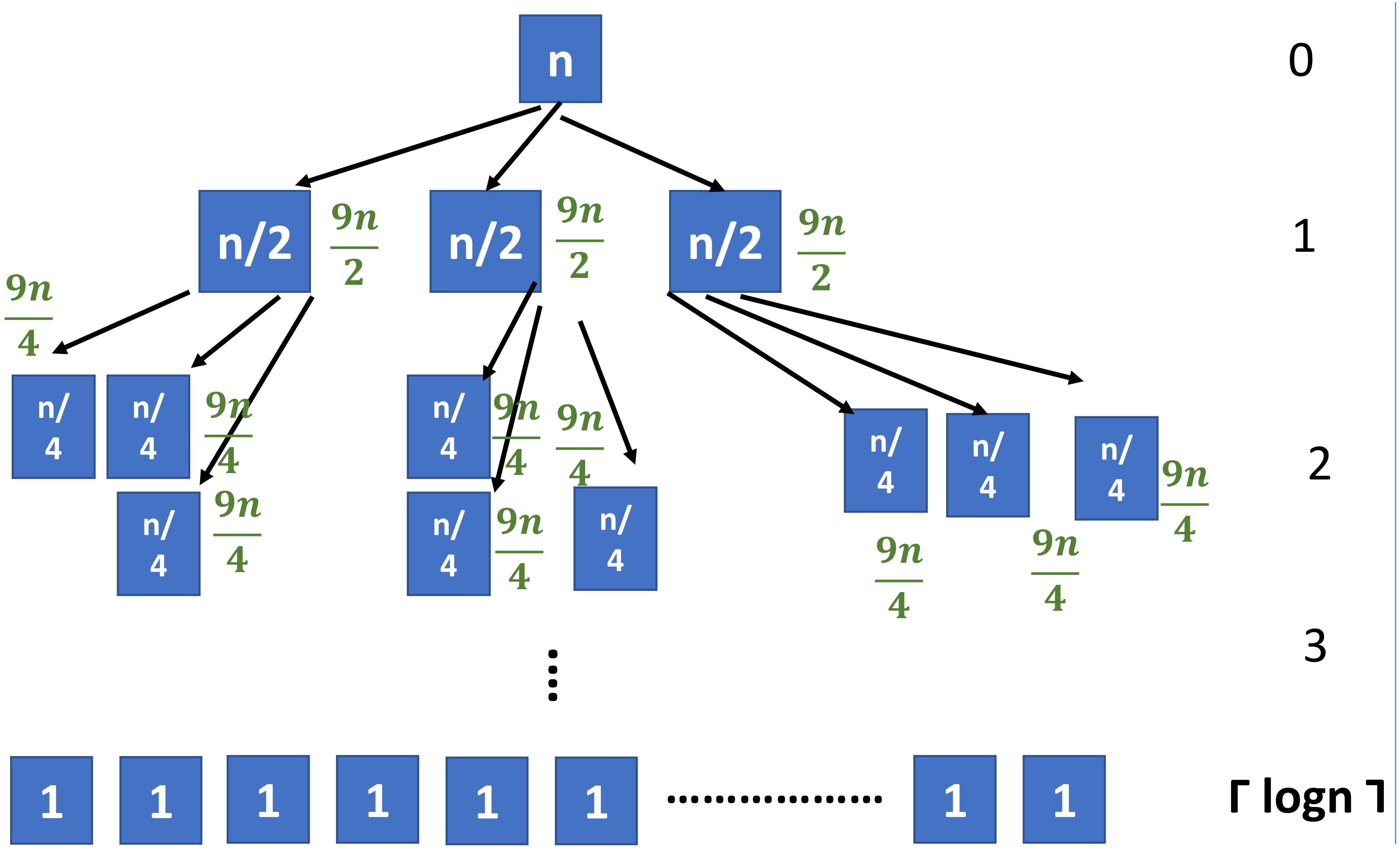
**$T(n)=3T(\frac{n}{2}) + 9n$**

**Ignoring issue of carries**

**4n steps**

**Return ac × $((10)^{(\text{number of digit of a})})^2$ +mid×$(10)^{(\text{number of digit of a})}$ +bd**

# $T(n) = 3T\left(\frac{n}{2}\right) + 9n$




$9n$

$$3\times\frac{9n}{2} = \left(\frac{3}{2}\right)^{1} \times 9n$$

$$9\times\frac{9n}{4} = \left(\frac{3}{2}\right)^{2} \times 9n$$

$$27\times\frac{9n}{8} = \left(\frac{3}{2}\right)^{3} \times 9n$$

$$\left(\frac{3}{2}\right)^{\lceil \log n \rceil} \times 9n$$

**Calculations:**

$$(1+a+a^2+\ldots\ldots+a^L) = \frac{a^{L+1-1}}{a-1}$$

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

$$T(n) = 9n + \frac{3}{2}\times 9n + \left(\frac{3}{2}\right)^2 \times 9n + \ldots\ldots + \left(\frac{3}{2}\right)^{\lceil \log n \rceil} \times 9n$$

$$= 9n \times \left(1 + \frac{3}{2} + \left(\frac{3}{2}\right)^2 + \ldots\ldots + \left(\frac{3}{2}\right)^{\lceil \log n \rceil}\right)$$

$$= \mathbf{9n} \times \left[\frac{\left(\frac{3}{2}\right)^{\lceil \log n \rceil + 1} - 1}{\frac{3}{2} - 1}\right] = 9n \times (2) \times \left[\left(\frac{3}{2}\right)^{\lceil \log n \rceil + 1} - 1\right]$$

$$= 2\times 9n\left[2^{\log_2 \frac{3}{2}}\right]^{\log n + 1} - 18n = 18n\left[2^{(\log_2 3\ -1)}\right]^{(\log n + 1)} - 18$$

$$= 18n\left[2^{((\log_2 n)^\wedge \log_2 3\ - \log_2 n + \log_2 3\ -1)}\right] - 18n = 18n\left[\frac{n^{\log_2 3} \times 2^{\log_2 3\ -1}}{n}\right] - 18n$$

$$= 18\times 2^{(\log_2 3\ -1)} \times n^{\log_2 3} - 18n = o(n^{\log_2 3})$$