Divide and Conquer

Design and analysis of Algorithm

Tutorial 1

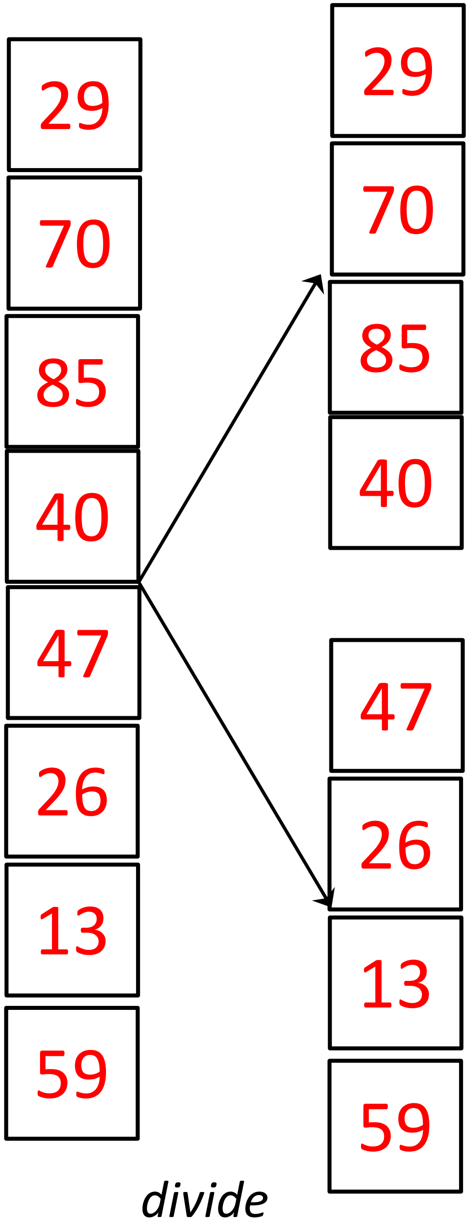
Sorting Problem

29	13
70	26
85	29
40	40
47	47
26	59
13	70
59	85

- Main Problem
  - Input: An unsorted array A[1..n]
  - Output: An sorted array A’[1..n]
- Sub-problem
  - Input: An unsorted array A[i,j] 1≤i,j ≤n
  - Output: An sorted array A’[i,j]

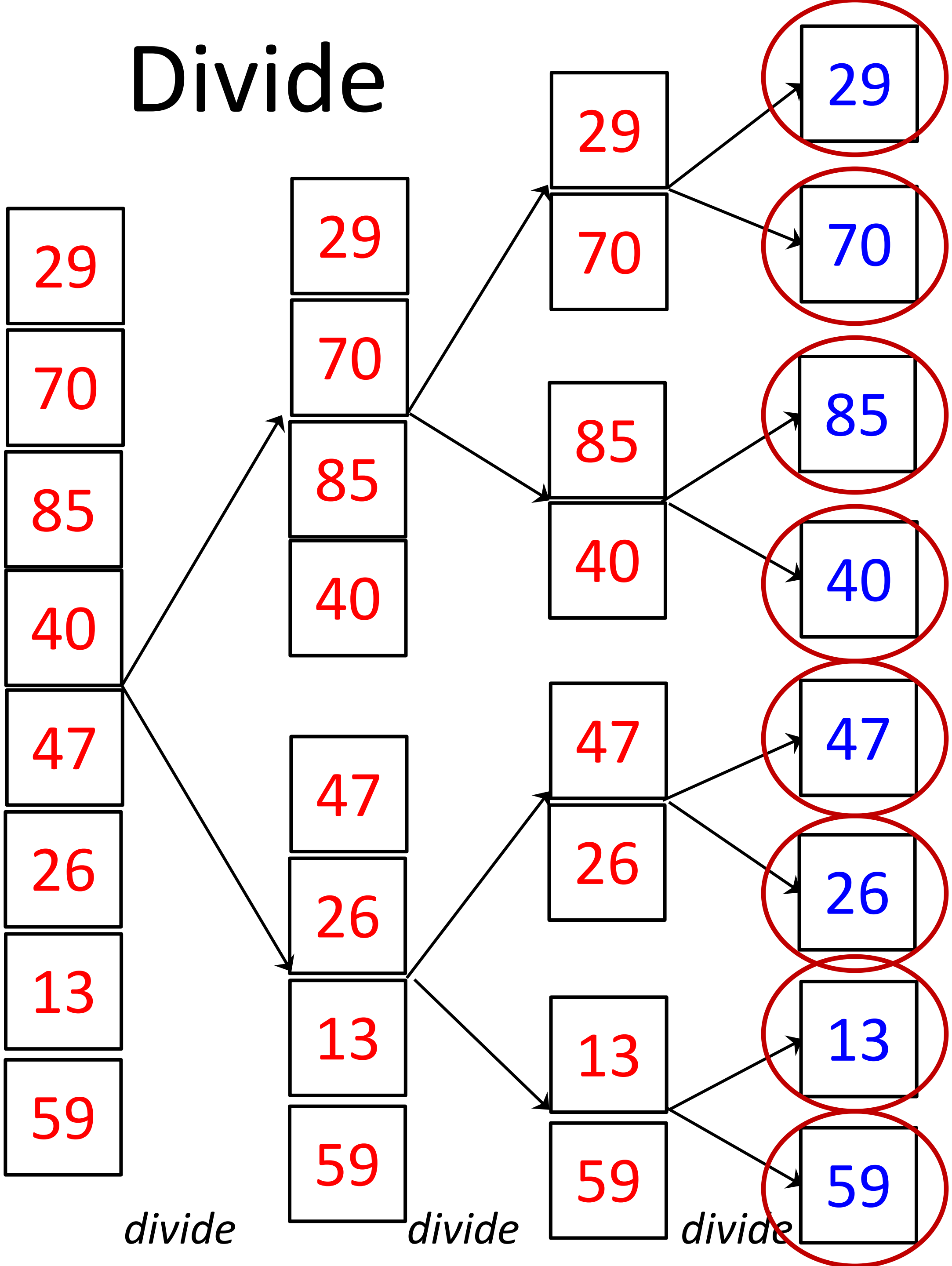
47	13
26	26
13	47
59	59

Divide



- Divide Process Concept
  1. Divide it recursively
- Problem:
  - Input: A[1..n]
  - Output: A[1..⌊n/2⌋], A[⌊n/2⌋+1 ..n]

Divide

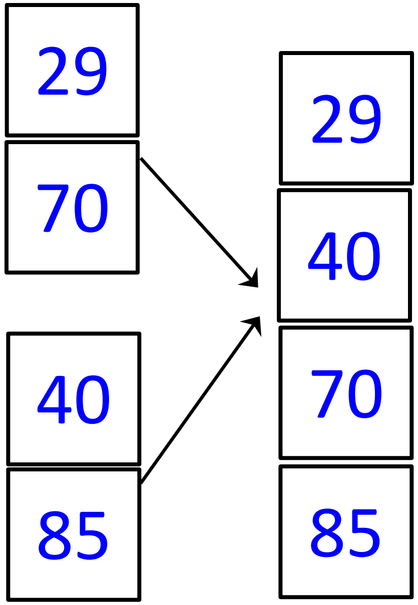


- Divide Process Concept
  1. Divide it recursively **until it is easy to conquer** (base case)

Base Case

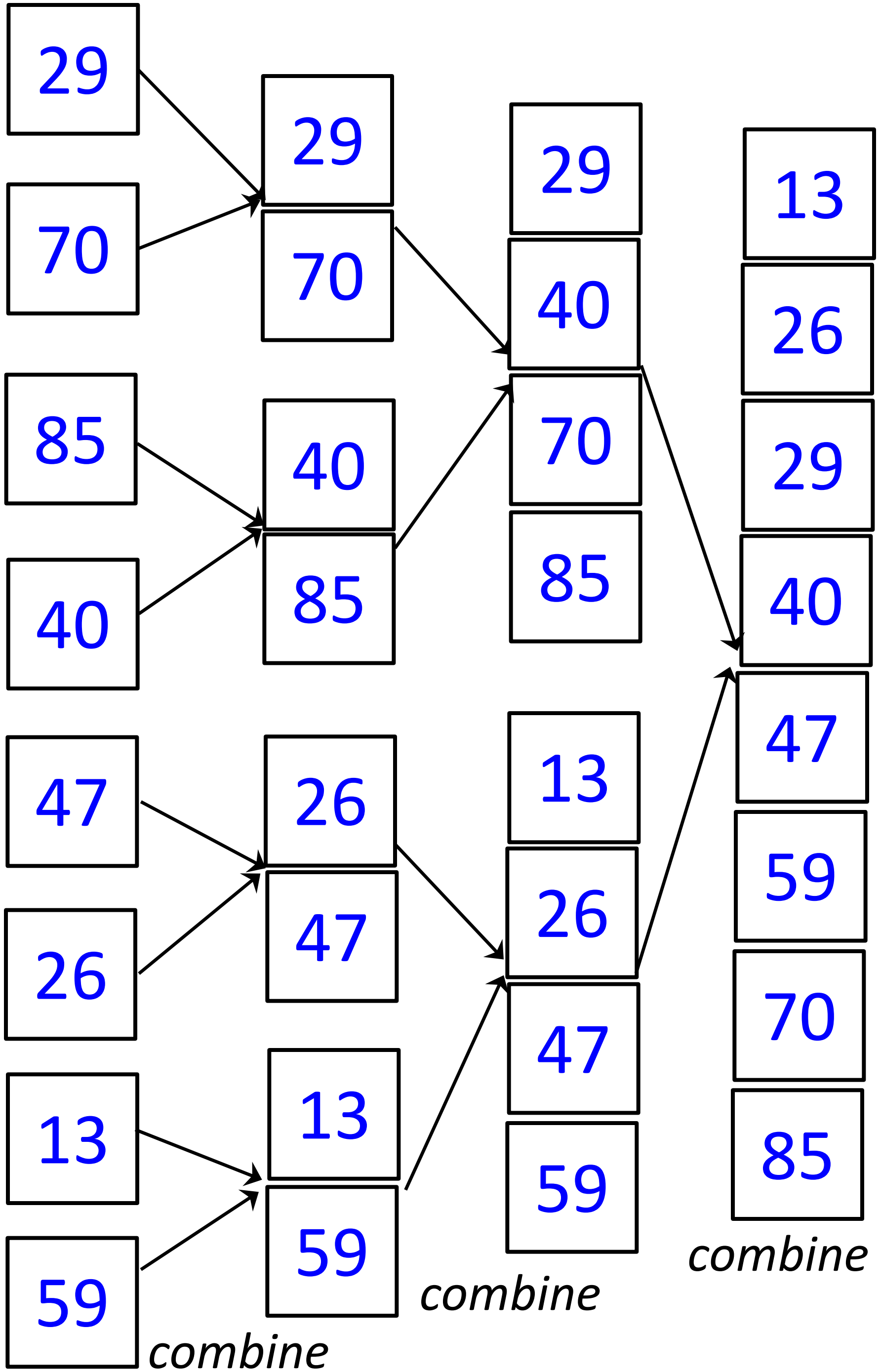
- Base Case Problem
  - Input: An unsorted array A[i]
  - Output: An sorted array A’[i]
- How to solve/conquer it?
  - It has been sorted →A’[i]=A[i] (easy) ☺

29



Combine

- Combine Problem:
  - Input: Given two sorted array B and C
  - **Output: A sorted array of elements including B and C**



Combine

- Combine the array recursively

Calculating $f(n)=a^n$

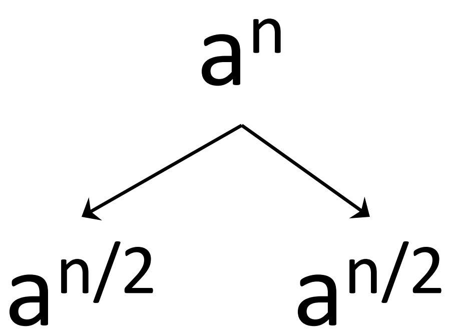
- Main Problem:
  - Input: value a, value n
  - Output: a^n
- Sub-problem:
  - Input: value a, value x, $1 \le x \le n$
  - Output: a^x

$$a^n = \underbrace{a * a * .... * a}_{n}$$

$$= \underbrace{(a * a * .... * a)}_{x}\underbrace{(a * a * .... * a)}_{n-x}$$

Divide Process

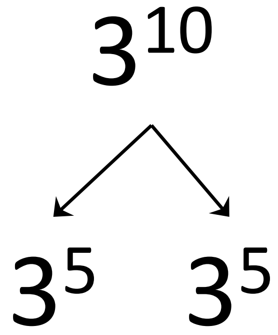
Even Case



a^n

$a^{n/2}$ $a^{n/2}$

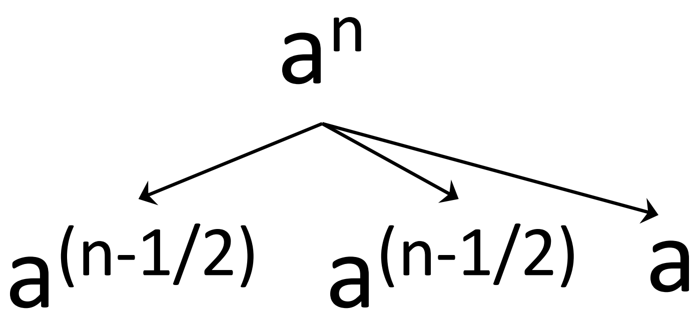
Example



3^{10}

3^5 3^5

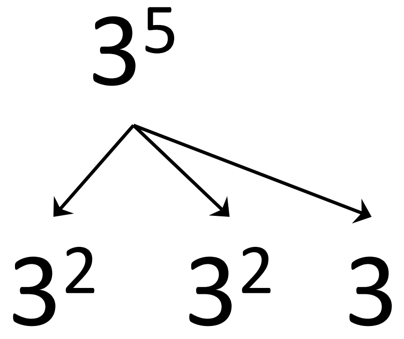
Odd Case



a^n

$a^{(n-1/2)}$ $a^{(n-1/2)}$ a

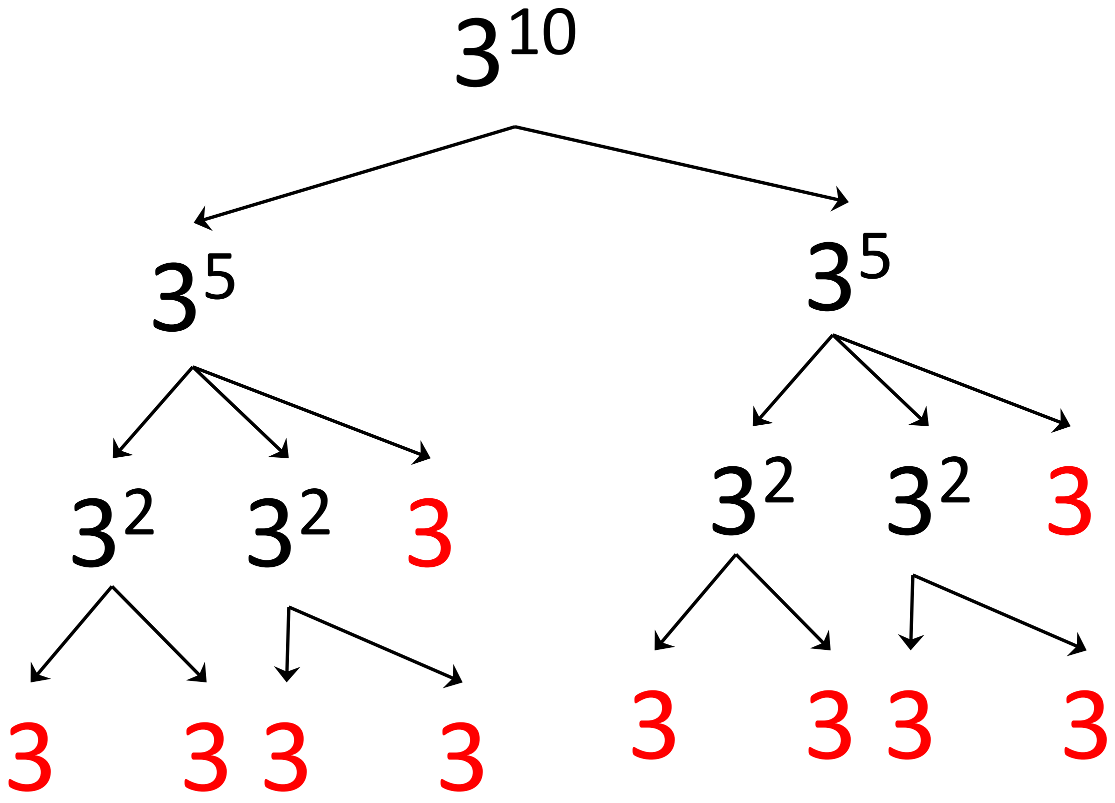
Example



3^5

3^2 3^2 3

Divide Process



3^{10}

3^5 3^5

3^2 3^2 3 3^2 3^2 3

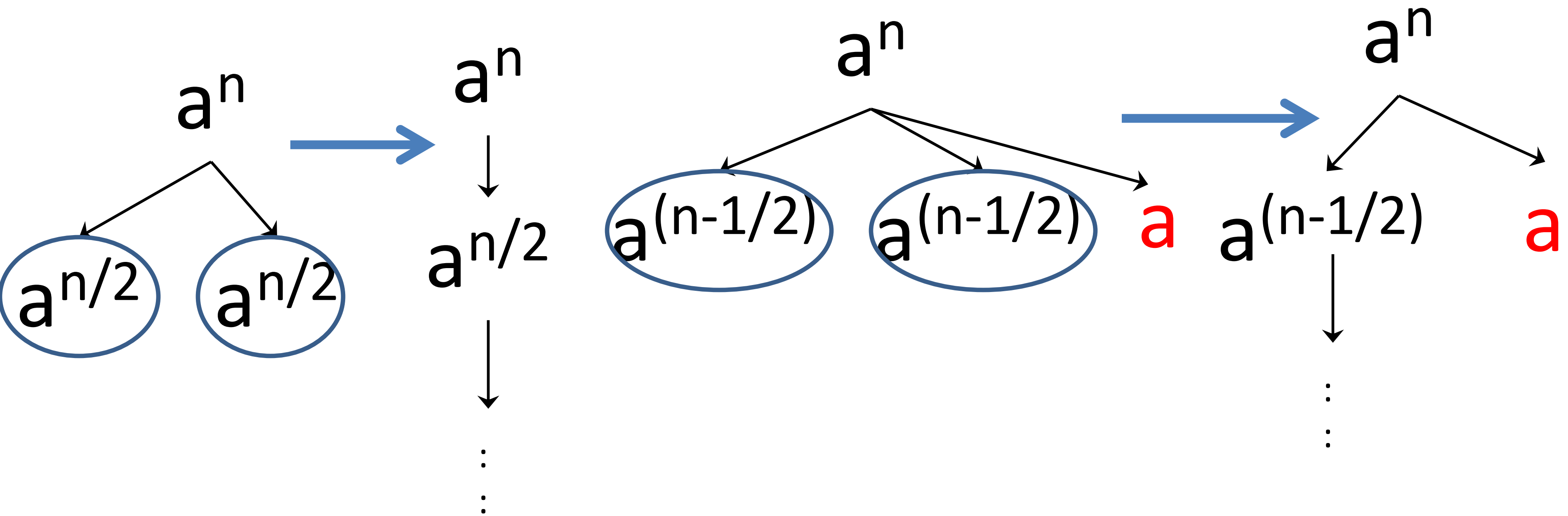
3 3 3 3 3 3 3 3

- **Divide until it is easy to conquer (Base Case)**

Divide Process

Even Case

Odd Case



a^n

$a^{n/2}$ $a^{n/2}$

a^n

$a^{n/2}$

a^n

$a^{(n-1/2)}$ $a^{(n-1/2)}$ a

a^n

$a^{(n-1/2)}$ a

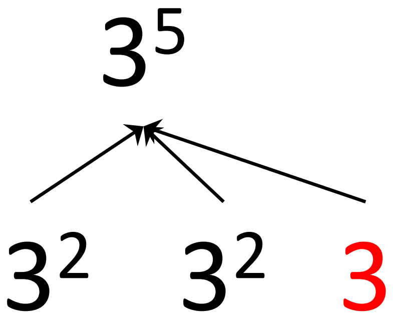
- More sub-problems can increase time complexity
- The sub-problem are the same →combine them to one

Base Case

- Base Case Problem:
  - Input: value a and n=1
  - Output: a^n
- How to solve it?
  - Directly return $a^1 = a$ (Easy ☺)

Combine

- Combine Problem:
  - Input: a^x
  - Output: a^{2x} if x is even a^{2x+1} if x is odd
- How to solve it?
  - Just do at most 2 multiplication



3^5

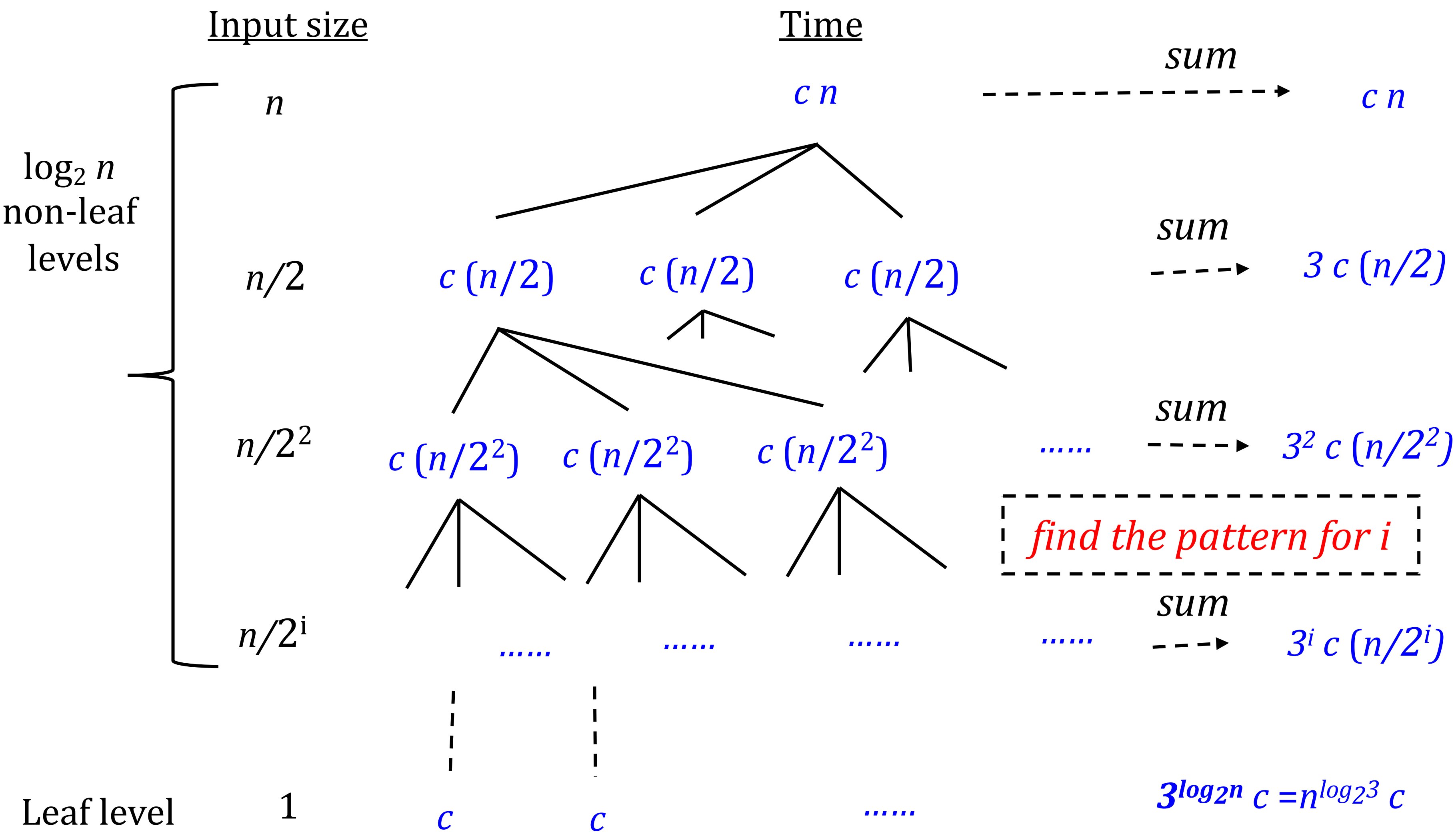
3^2 3^2 3

How to write Recurrence for D/C Algorithm?

$$\mathrm{T}(n) = a\,\mathrm{T}(n/b) + f(\mathrm{n})$$

- n: data input size
- a: number of sub-problems division
- n/b: the sub-problem sizes
- f(n): time to combine sub-problems

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



Input size | Time

$\log_2 n$ non-leaf levels

n — $c\,n$ — sum — $c\,n$

$n/2$ — $c\,(n/2)$ $c\,(n/2)$ $c\,(n/2)$ — sum — $3\,c\,(n/2)$

$n/2^2$ — $c\,(n/2^2)$ $c\,(n/2^2)$ $c\,(n/2^2)$ …… — sum — $3^2\,c\,(n/2^2)$

find the pattern for i

$n/2^i$ — …… …… …… …… — sum — $3^i\,c\,(n/2^i)$

Leaf level — 1 — c c …… — $3^{\log_2 n}\,c = n^{\log_2 3}\,c$

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

- Time at the **leaf** level: $c\, n^{\log_2 3}$
- Number of **non-leaf** levels L: $\log_2 n$
- Consider the level i (level 0 is the top level)
  - Input size of a subproblem: $n/2^i$
  - Number of subproblems: 3^i
  - Combine time of a subproblem: $c\,(n/2^i)$
  - Total time at this level: $3^i\, c\,(n/2^i)$
- Total time T(n):

$c\, n^{\log 23} + \Sigma_{i=0..L-1}\, 3^i\, c\,(n/2^i)$

$= c\, n^{\log 23} + c\, n\ \ (\mathbf{1.5}^L - 1) / (1.5 - 1)$

$= O\,(c\, n^{\log 23} + c\, n\, n^{\log 21.5})$

$= O\,(n^{1.585})$

geometric sum equation

log base equation

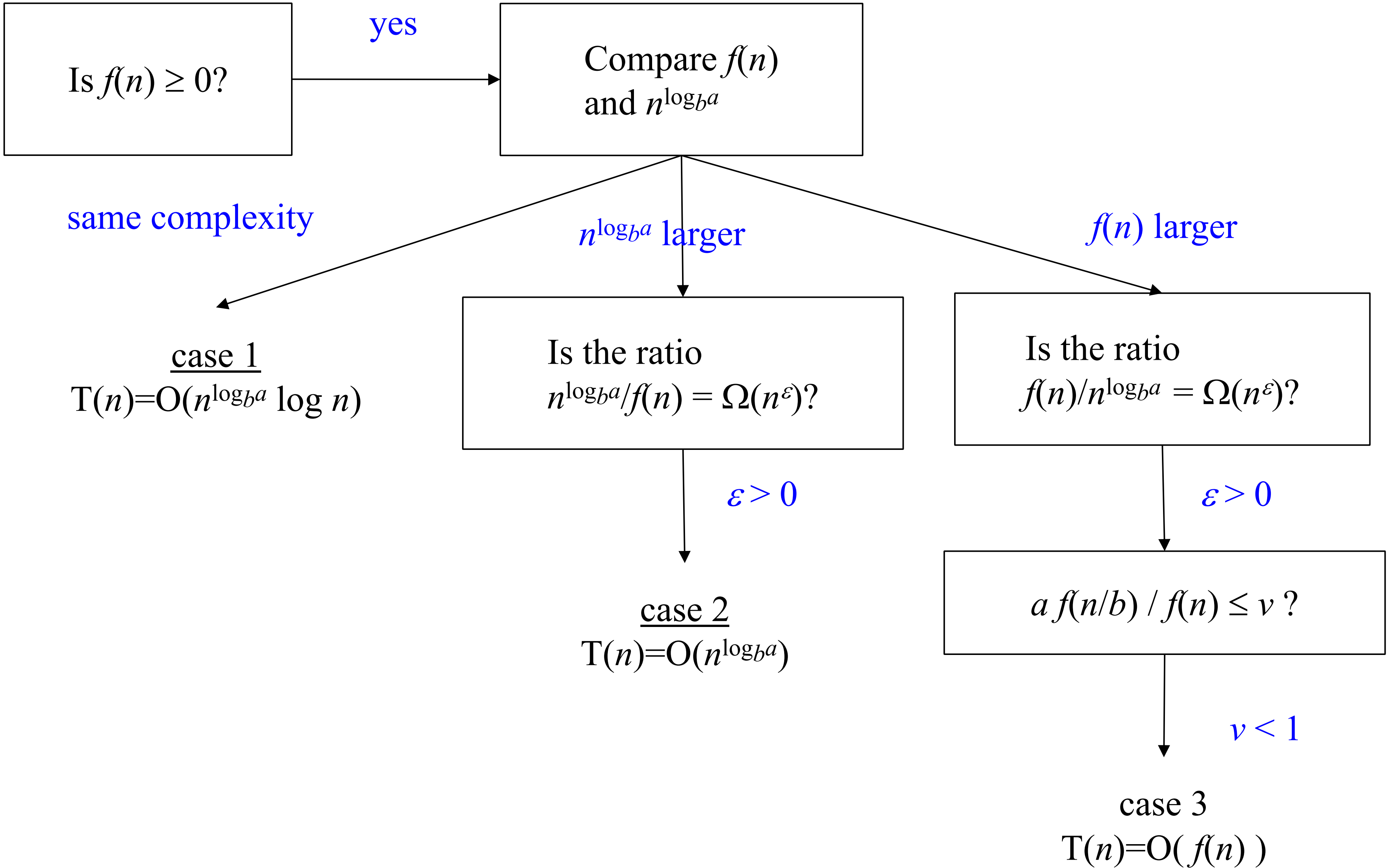
How do we get 1.585 ?

Method: Master Method

- Given the recurrence: $T(n) = a\,T(n/b) + f(\mathrm{n})$
  - $f(n)$ must be ≥ 0
- Compare $f(n)$ and $n^{\log_b a}$
  - $\boldsymbol{\varepsilon}$, $\boldsymbol{v}$ are some constants such that $\varepsilon > 0$, $v < 1$

$$T(n) = \begin{cases} O(n^{\log_b a} \log n) & \text{if } f(n) = \Theta(n^{\log_b a}) \\ & \text{polynomially larger} \\ O(n^{\log_b a}) & \text{if } n^{\log_b a} / f(n) = \Omega(n^{\boldsymbol{\varepsilon}}) \\ & \text{polynomially larger} \\ O(\,f(n)\,) & \text{if } f(n) \,/\, n^{\log_b a} = \Omega(n^{\boldsymbol{\varepsilon}}) \\ & \text{and } a\, f(n/b) \,/\, f(n) \leq \boldsymbol{v} \\ & \text{regularity condition} \end{cases}$$

$$T(n) = a\,T(n/b) + f(n)$$



Is $f(n) \geq 0$?

yes

Compare $f(n)$ and $n^{\log_b a}$

same complexity

$n^{\log_b a}$ larger

$f(n)$ larger

case 1

$T(n)=O(n^{\log_b a} \log n)$

Is the ratio $n^{\log_b a}/f(n) = \Omega(n^{\varepsilon})$?

$\varepsilon > 0$

case 2

$T(n)=O(n^{\log_b a})$

Is the ratio $f(n)/n^{\log_b a} = \Omega(n^{\varepsilon})$?

$\varepsilon > 0$

$a\,f(n/b)\,/\,f(n) \leq v$?

$v < 1$

case 3

$T(n)=O(\,f(n)\,)$

Finally

- Practice makes perfect (Active learner)
- Reference Exercises:
  1. Introduction to Algorithm
  2. https://www.cs.berkeley.edu/~vazirani/algorithms/chap2.pdf
- Divide and Conquer is the foundation concept for you to learn Dynamic Programming