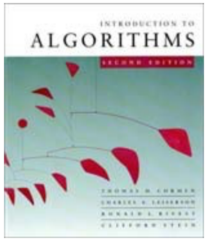
CS60020: Foundations of Algorithm Design and Machine Learning

Sourangshu Bhattacharya

DIVIDE AND CONQUER

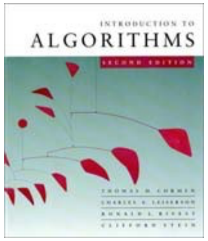


Merge sort

MERGE-SORT $A[1\,.\,.\,n]$

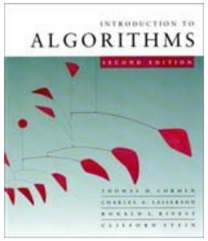
1. If $n = 1$, done.
2. Recursively sort $A[\,1\,.\,.\,\lceil n/2\rceil]$ and $A[\,\lceil n/2\rceil+1\,.\,.\,n\,]$.
3. "***Merge***" the 2 sorted lists.

Key subroutine: **MERGE**



Merging two sorted arrays

20	12
13	11
7	9
2	1



Merging two sorted arrays

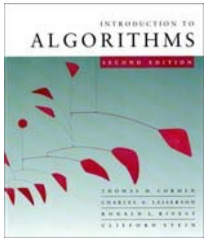
20 12

13 11

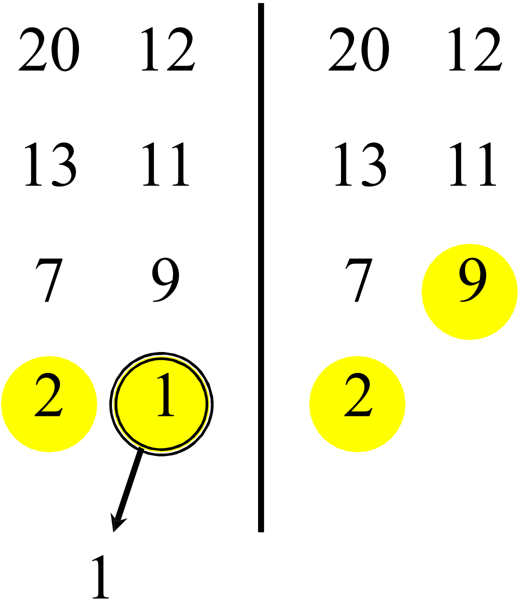
7 9

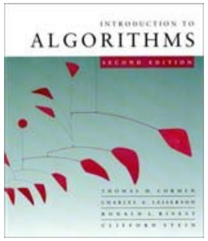
2 1

1

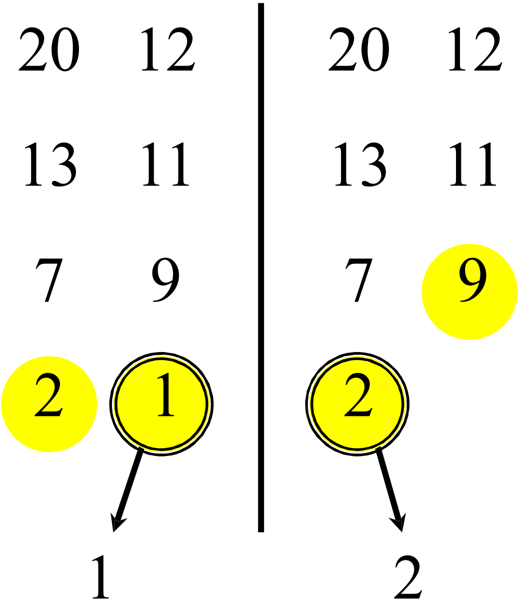


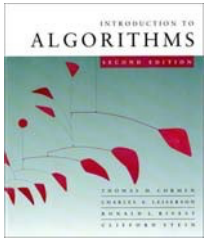
Merging two sorted arrays



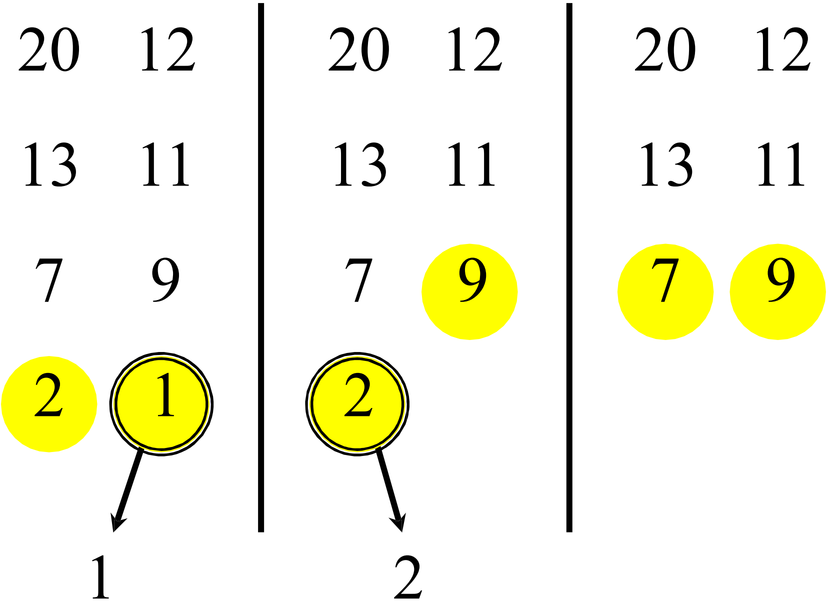


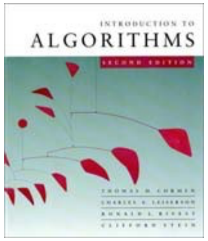
Merging two sorted arrays



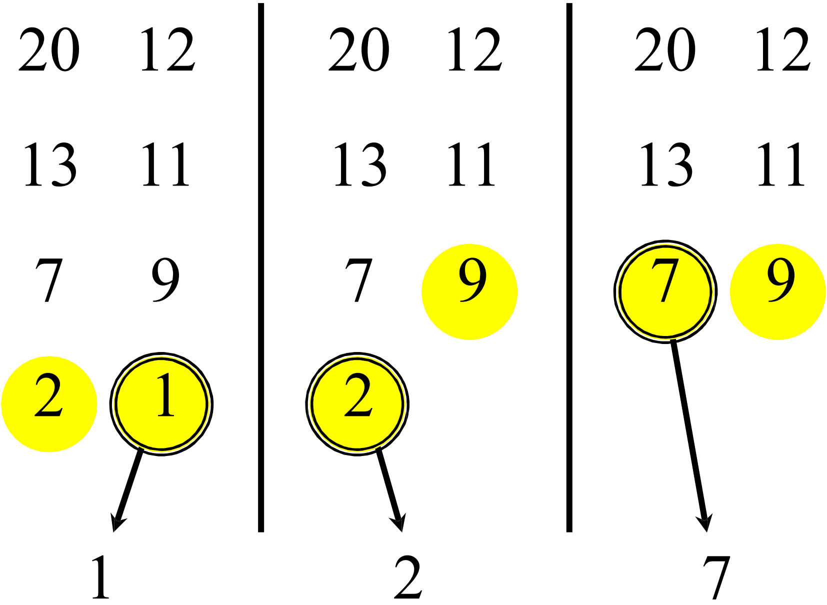


Merging two sorted arrays





Merging two sorted arrays



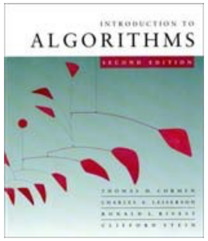
20 12 | 20 12 | 20 12

13 11 | 13 11 | 13 11

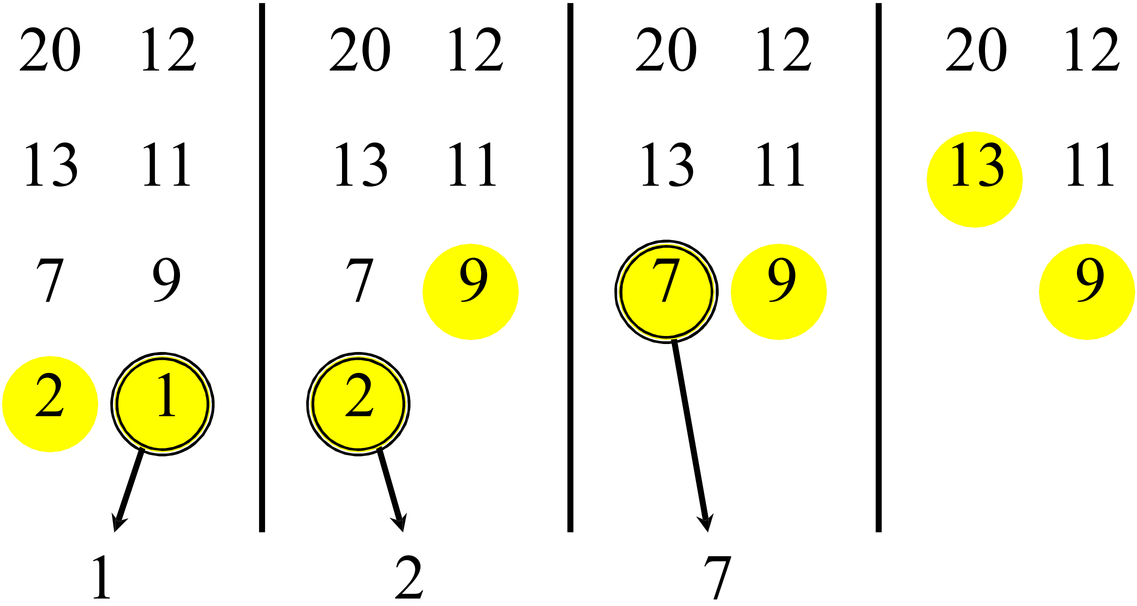
7 9 | 7 9 | 7 9

2 1 | 2

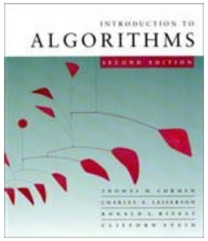
1 | 2 | 7



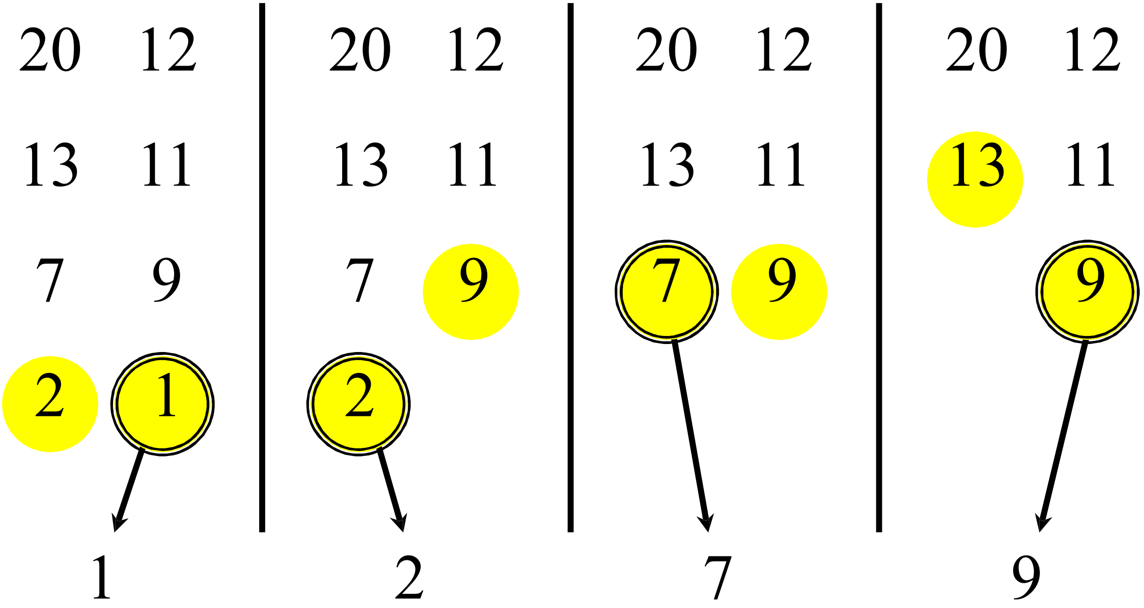
Merging two sorted arrays

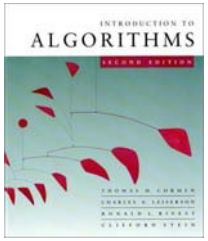


20	12	20	12	20	12	20	12
13	11	13	11	13	11	13	11
7	9	7	9	7	9		9
2	1	2					
1		2		7			

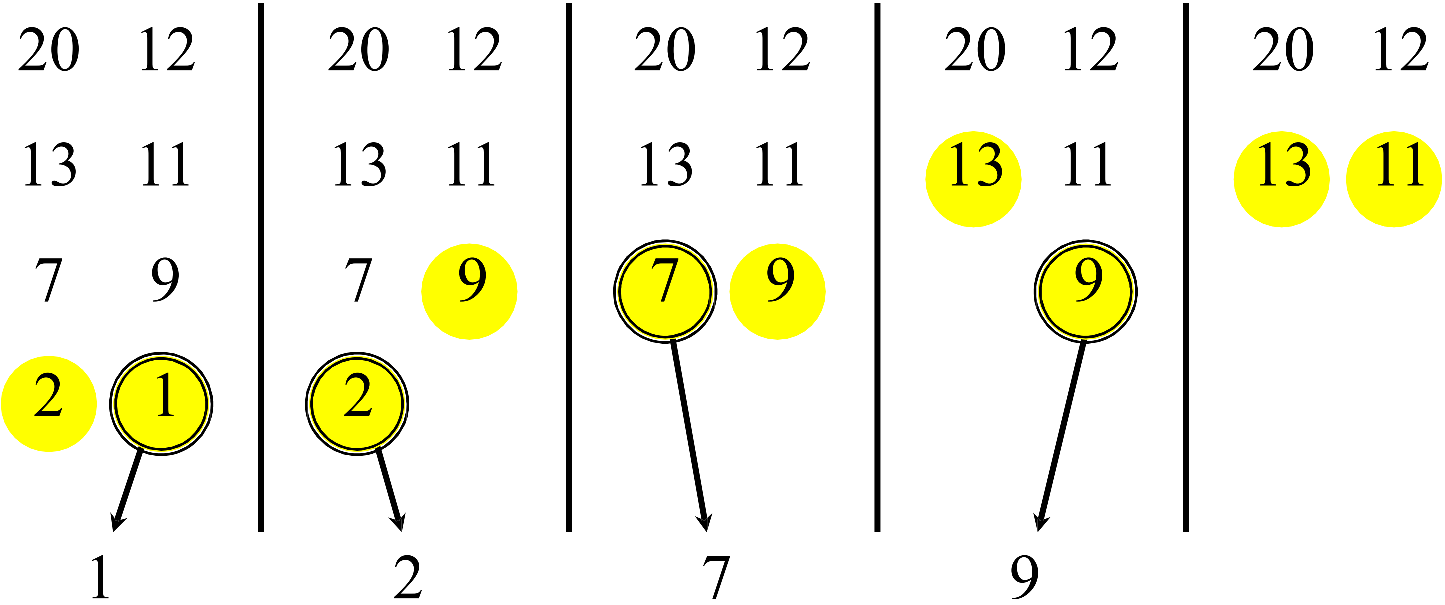


Merging two sorted arrays

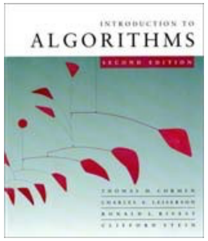




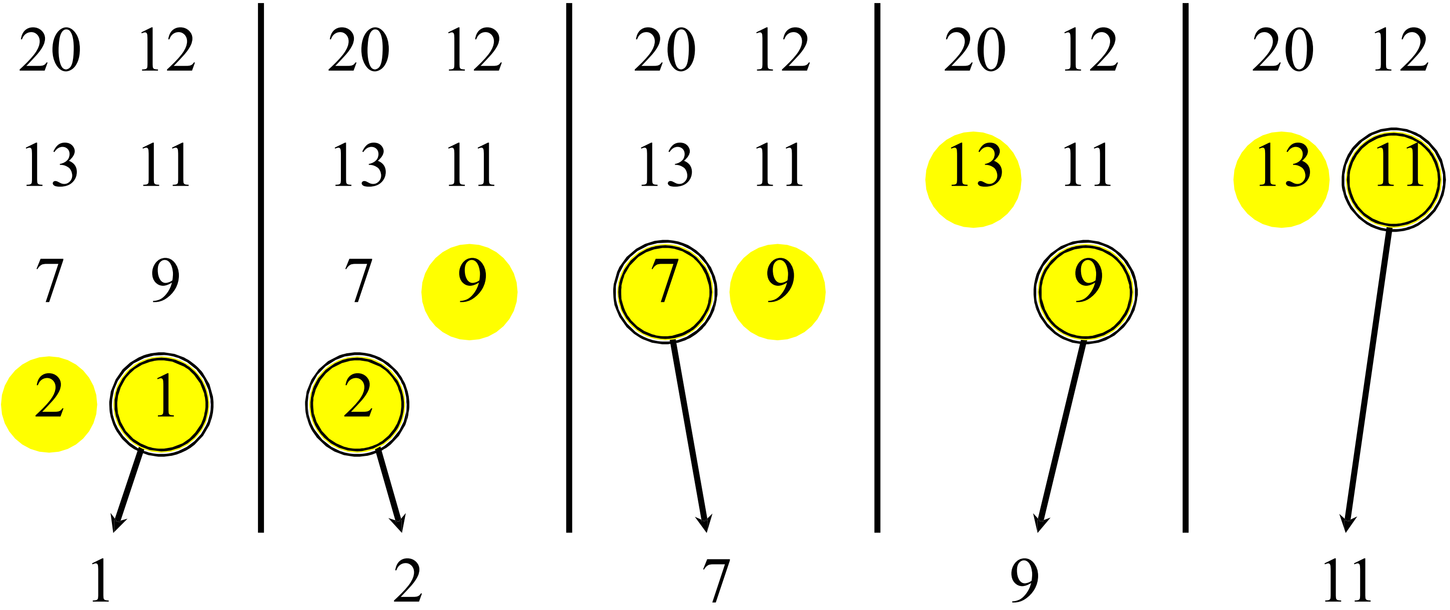
Merging two sorted arrays



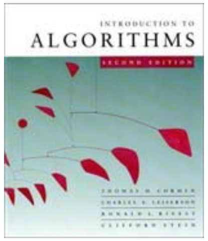
20	12	20	12	20	12	20	12	20	12
13	11	13	11	13	11	13	11	13	11
7	9	7	9	7	9		9		
2	1	2							
1		2		7		9			



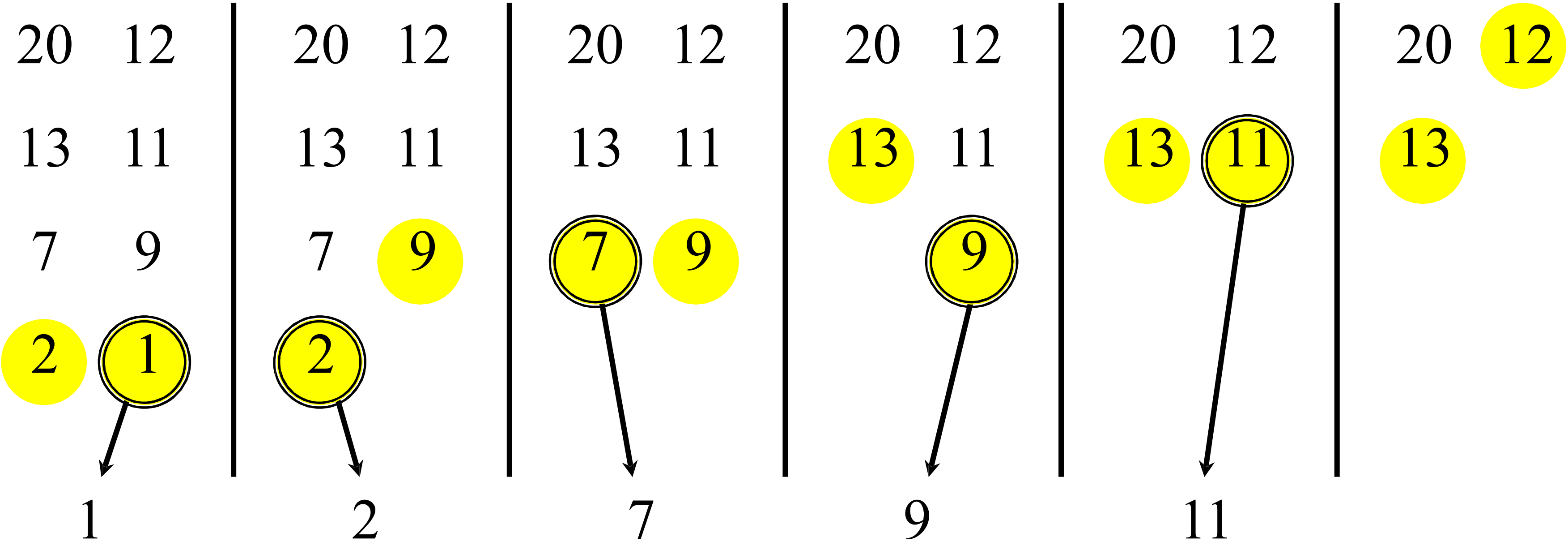
Merging two sorted arrays



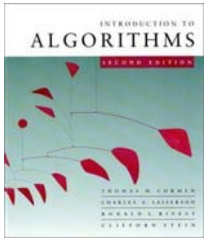
20	12	20	12	20	12	20	12	20	12
13	11	13	11	13	11	13	11	13	11
7	9	7	9	7	9		9		
2	1	2							
1		2		7		9		11	



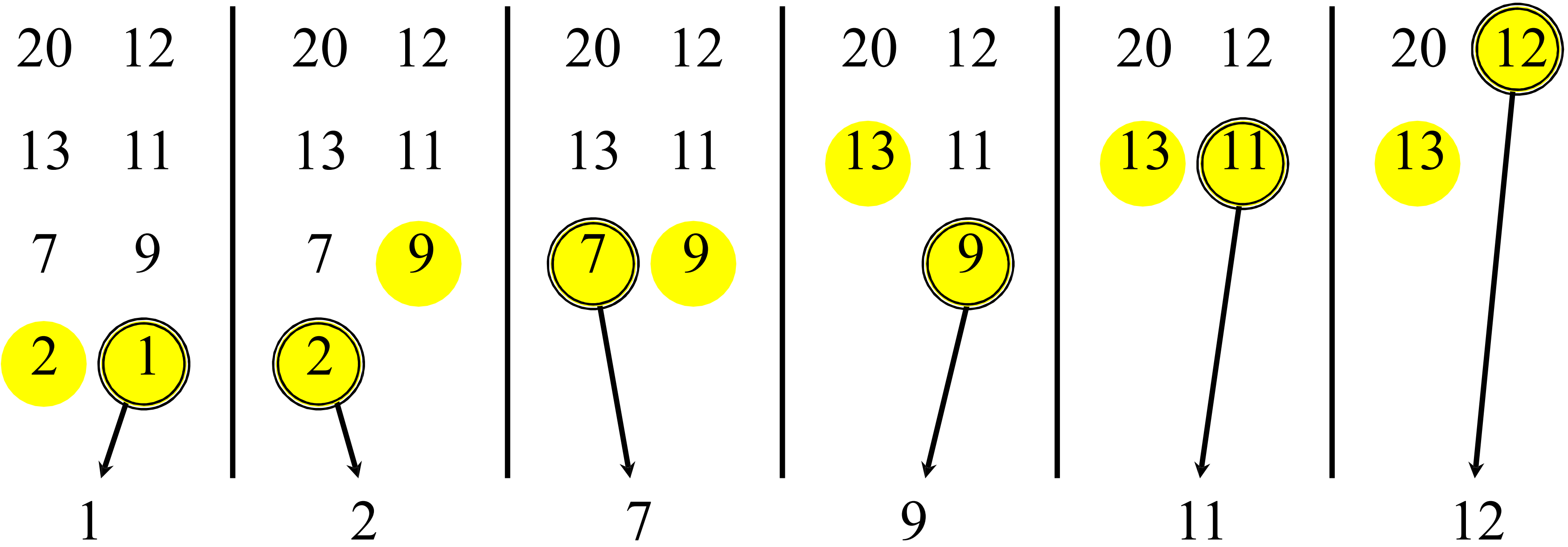
Merging two sorted arrays



20	12	20	12	20	12	20	12	20	12	20	12
13	11	13	11	13	11	13	11	13	11	13	
7	9	7	9	7	9		9				
2	1	2									
1		2		7		9		11			



Merging two sorted arrays

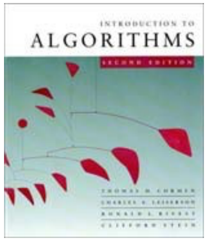

20 12 | 20 12 | 20 12 | 20 12 | 20 12 | 20 12

13 11 | 13 11 | 13 11 | 13 11 | 13 11 | 13

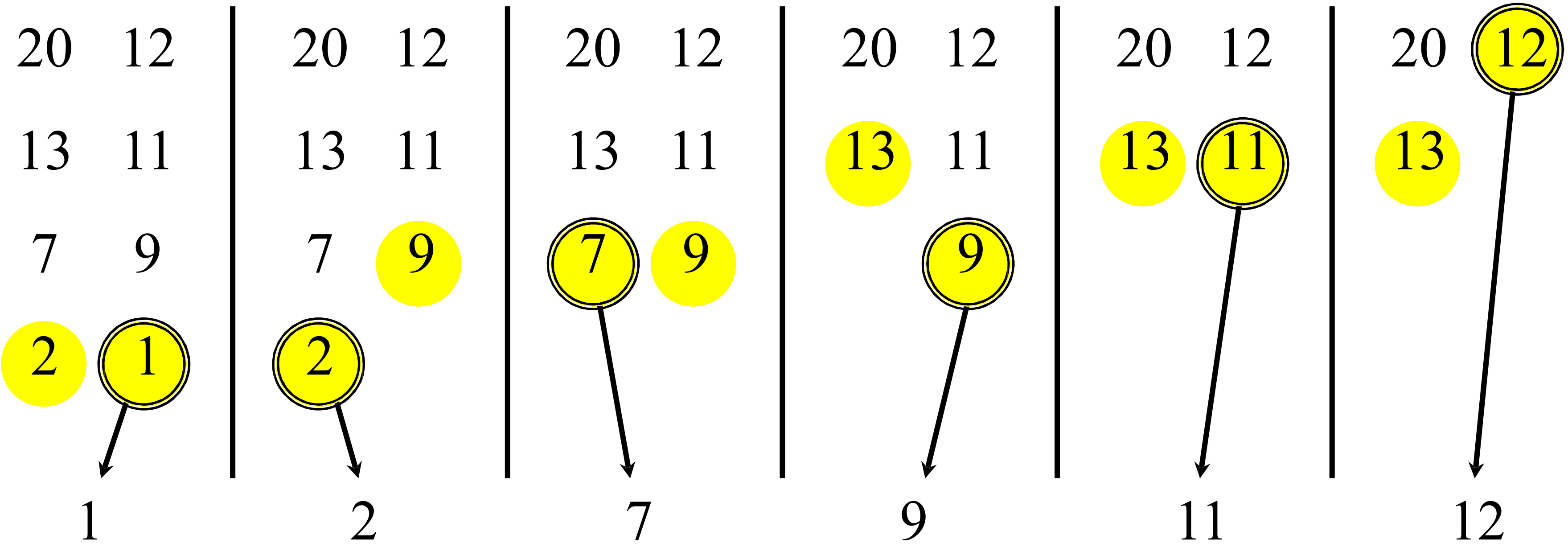
7 9 | 7 9 | 7 9 | 9

2 1 | 2

1 2 7 9 11 12

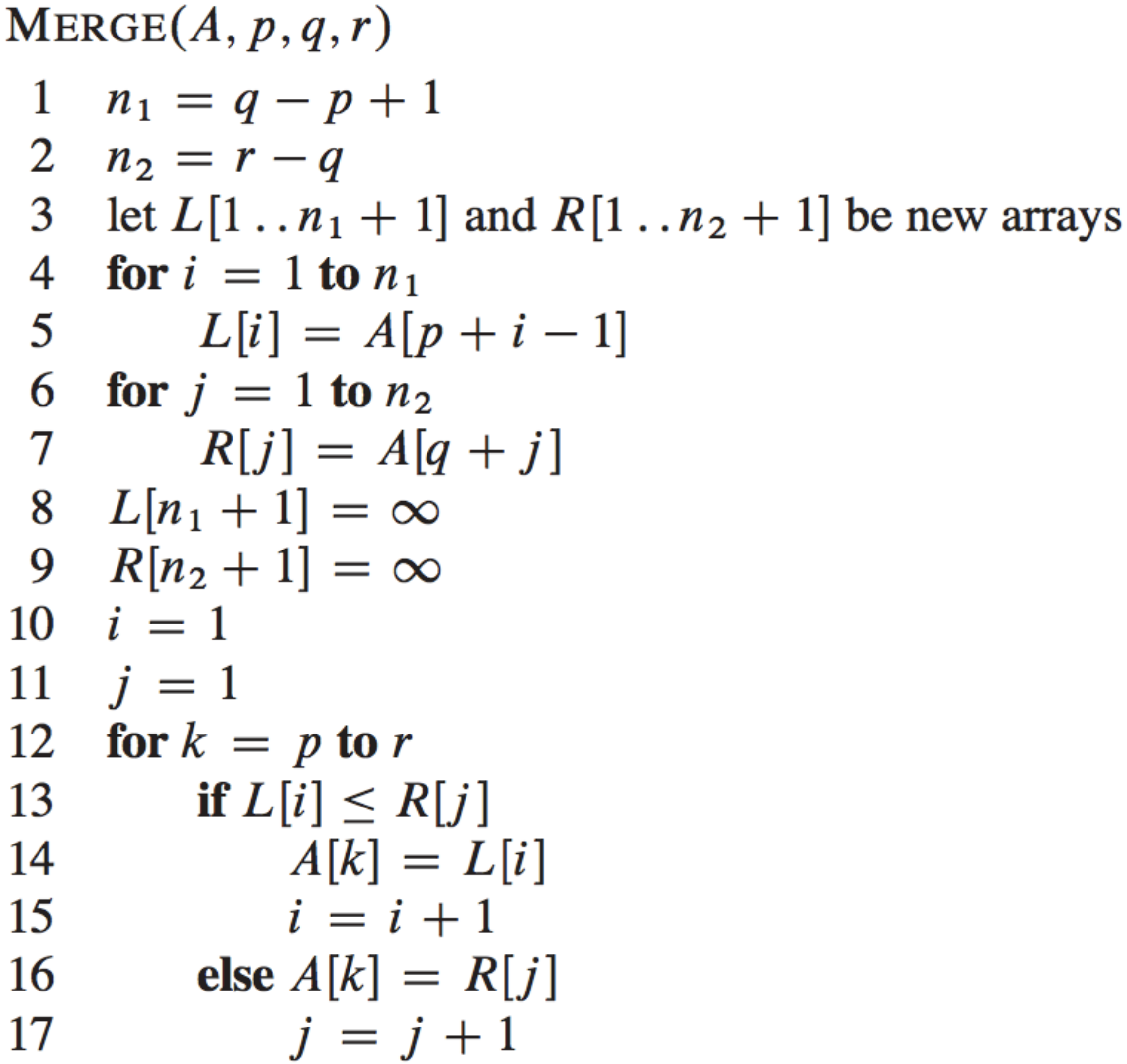


Merging two sorted arrays

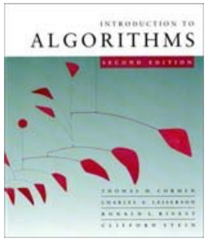


Time $= \Theta(n)$ to merge a total of n elements (linear time).

Merge



```
MERGE($A, p, q, r$)
$n_1 = q - p + 1$
$n_2 = r - q$
let $L[1 \,..\, n_1 + 1]$ and $R[1 \,..\, n_2 + 1]$ be new arrays
for $i = 1$ to $n_1$
    $L[i] = A[p + i - 1]$
for $j = 1$ to $n_2$
    $R[j] = A[q + j]$
$L[n_1 + 1] = \infty$
$R[n_2 + 1] = \infty$
$i = 1$
$j = 1$
for $k = p$ to $r$
    if $L[i] \le R[j]$
        $A[k] = L[i]$
        $i = i + 1$
    else $A[k] = R[j]$
        $j = j + 1$
```

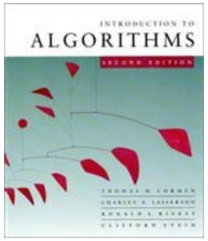


Analyzing merge sort

$T(n)$	**MERGE-SORT** $A[1\,.\,.\,n]$
$\Theta(1)$	1. If $n = 1$, done.
$2T(n/2)$	2. Recursively sort $A[\,1\,.\,.\,\lceil n/2\rceil]$ and $A[\,\lceil n/2\rceil+1\,.\,.\,n\,]$.
$\Theta(n)$	3. ***"Merge"*** the 2 sorted lists

Abuse

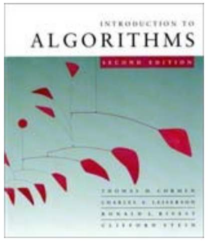
Sloppiness: Should be $T(\lceil n/2\rceil) + T(\lfloor n/2\rfloor)$, but it turns out not to matter asymptotically.



Recurrence for merge sort

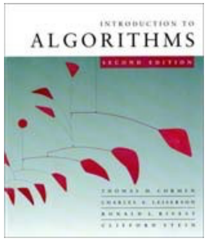
$$T(n) = \begin{cases} \Theta(1) \text{ if } n = 1; \\ 2T(n/2) + \Theta(n) \text{ if } n > 1. \end{cases}$$

- We shall usually omit stating the base case when $T(n) = \Theta(1)$ for sufficiently small n, but only when it has no effect on the asymptotic solution to the recurrence.
- CLRS and Lecture 2 provide several ways to find a good upper bound on $T(n)$.



Recursion tree

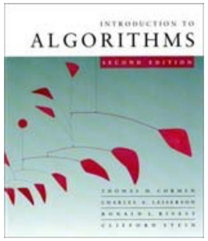
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.



Recursion tree

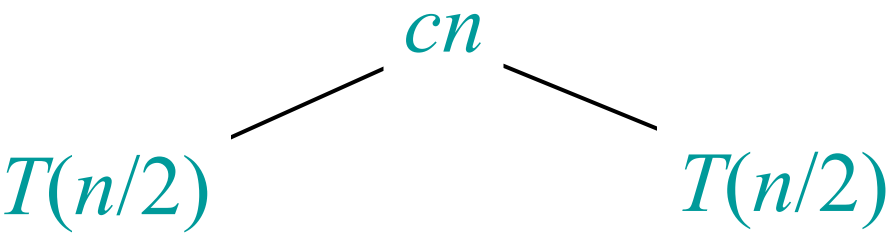
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.

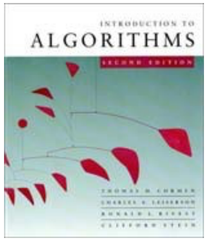
$$T(n)$$



Recursion tree

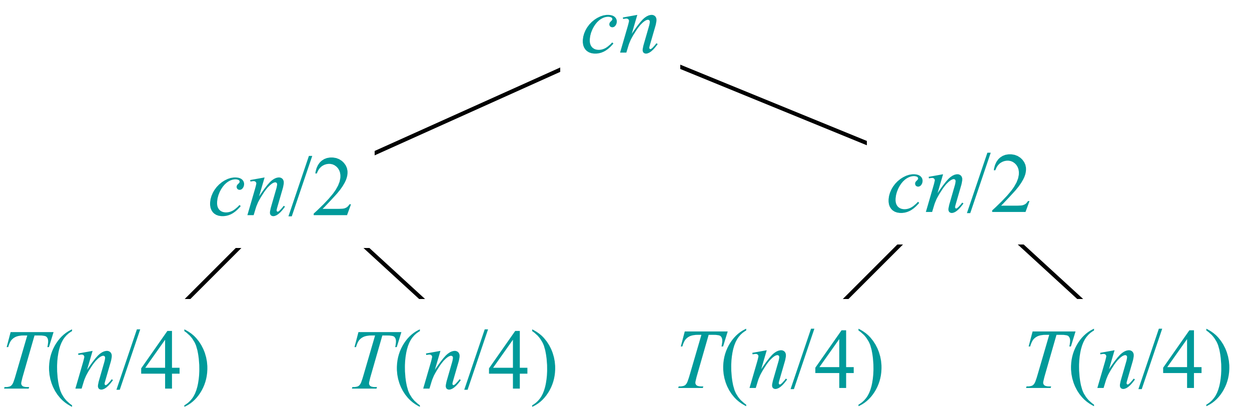
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.





Recursion tree

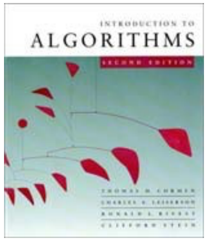
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.



cn

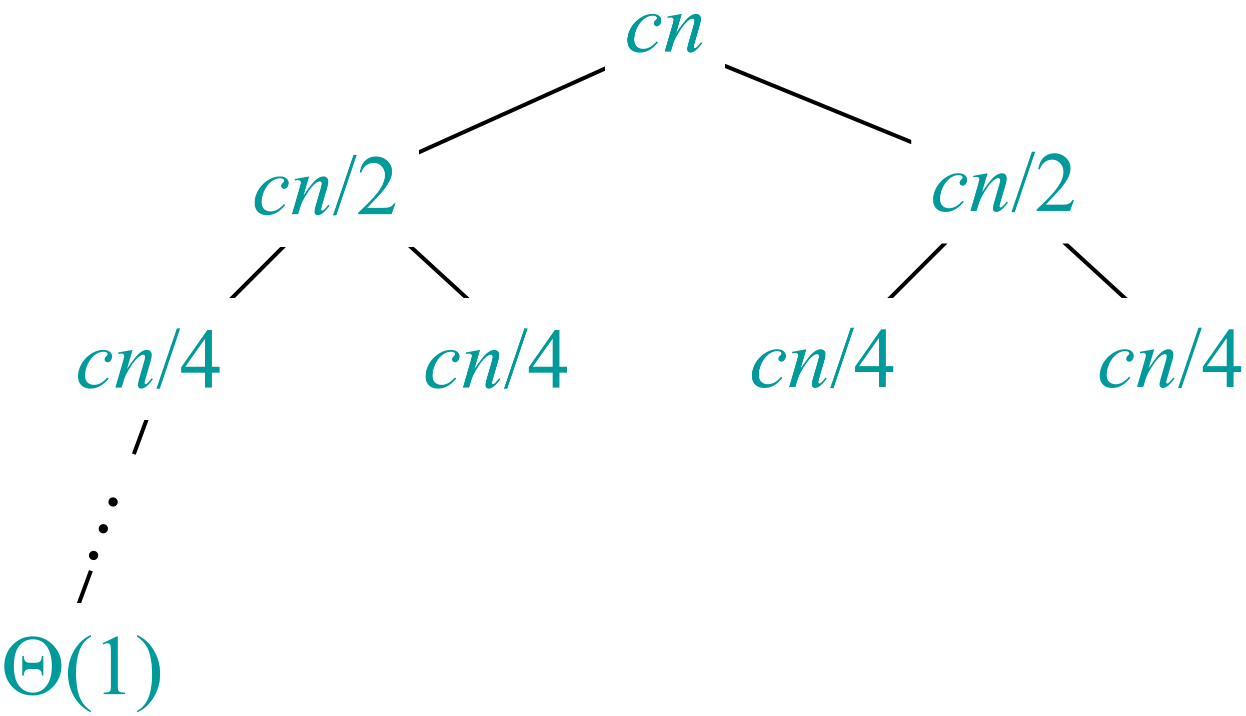
$cn/2$ $cn/2$

$T(n/4)$ $T(n/4)$ $T(n/4)$ $T(n/4)$



Recursion tree

Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.

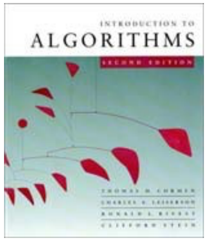


cn

$cn/2$ $cn/2$

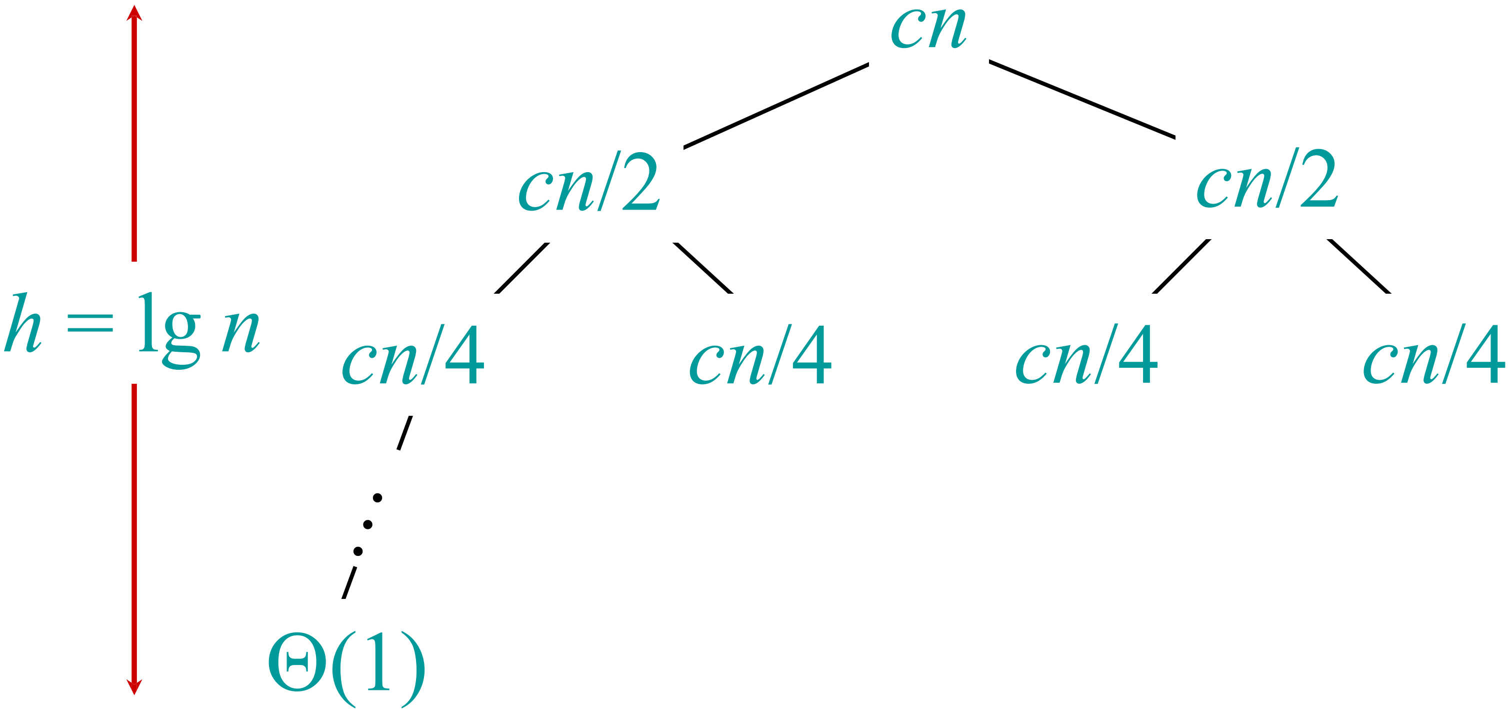
$cn/4$ $cn/4$ $cn/4$ $cn/4$

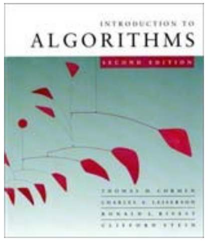
$\Theta(1)$



Recursion tree

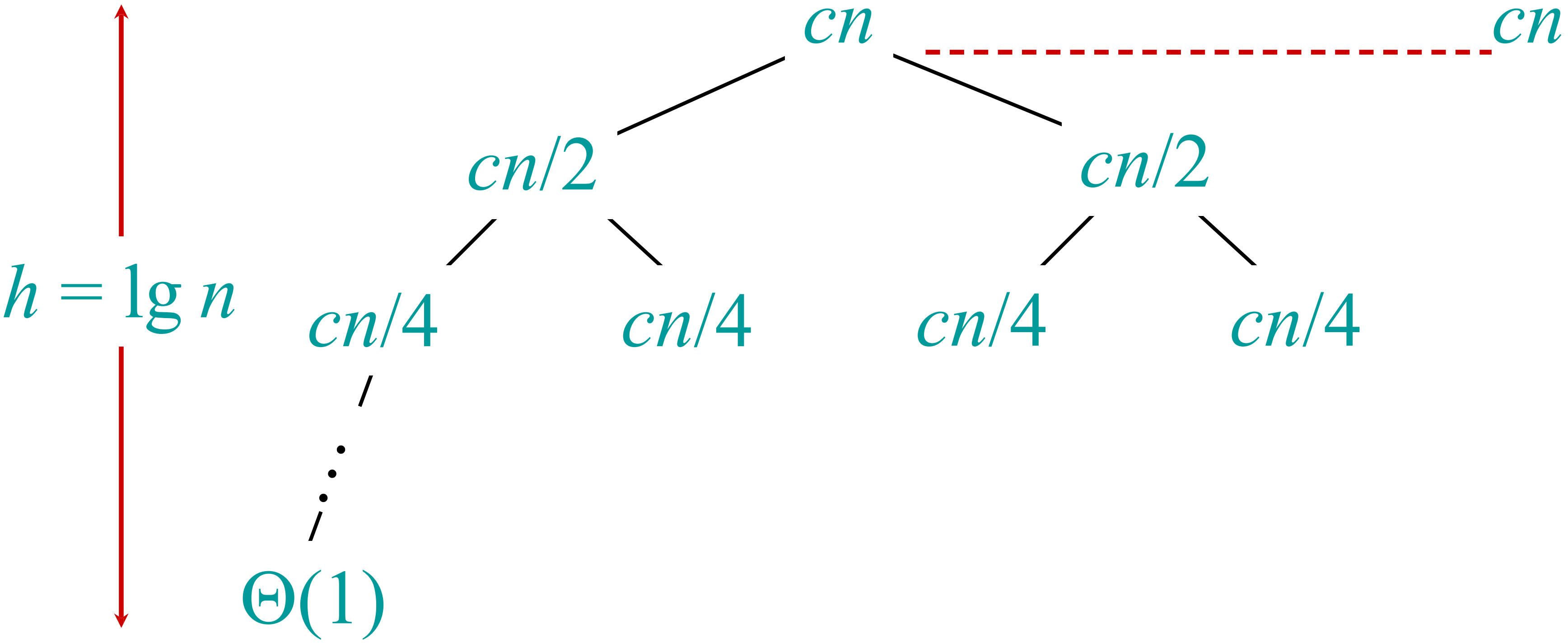
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.




INTRODUCTION TO
ALGORITHMS

Recursion tree

Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.


cn

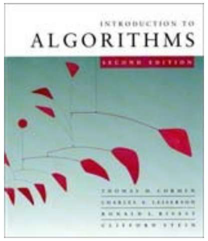
cn

$cn/2$ $cn/2$

$cn/4$ $cn/4$ $cn/4$ $cn/4$

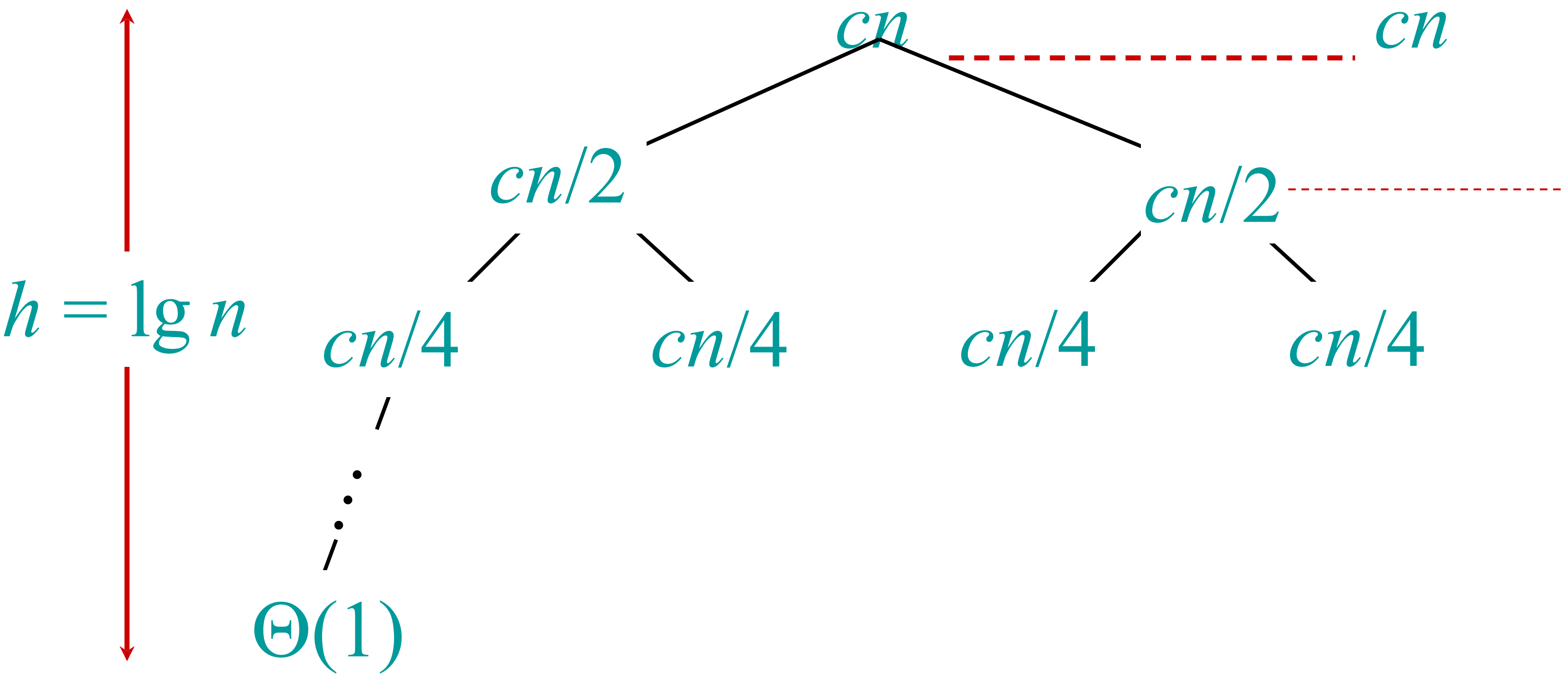
$h = \lg n$

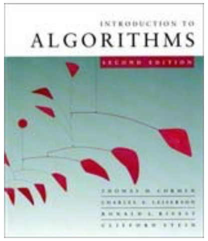
$\Theta(1)$



Recursion tree

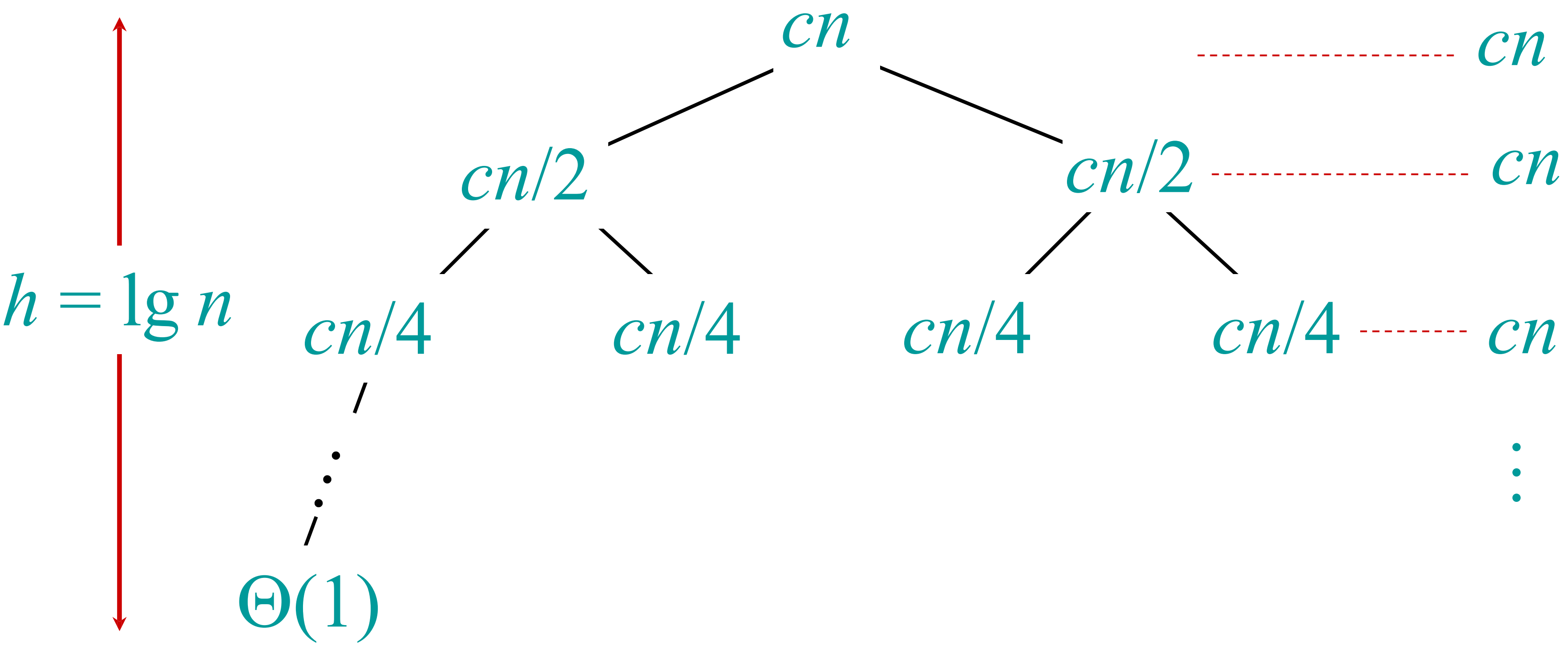
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.

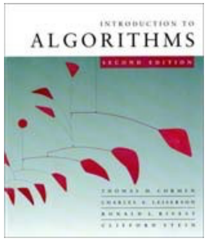




Recursion tree

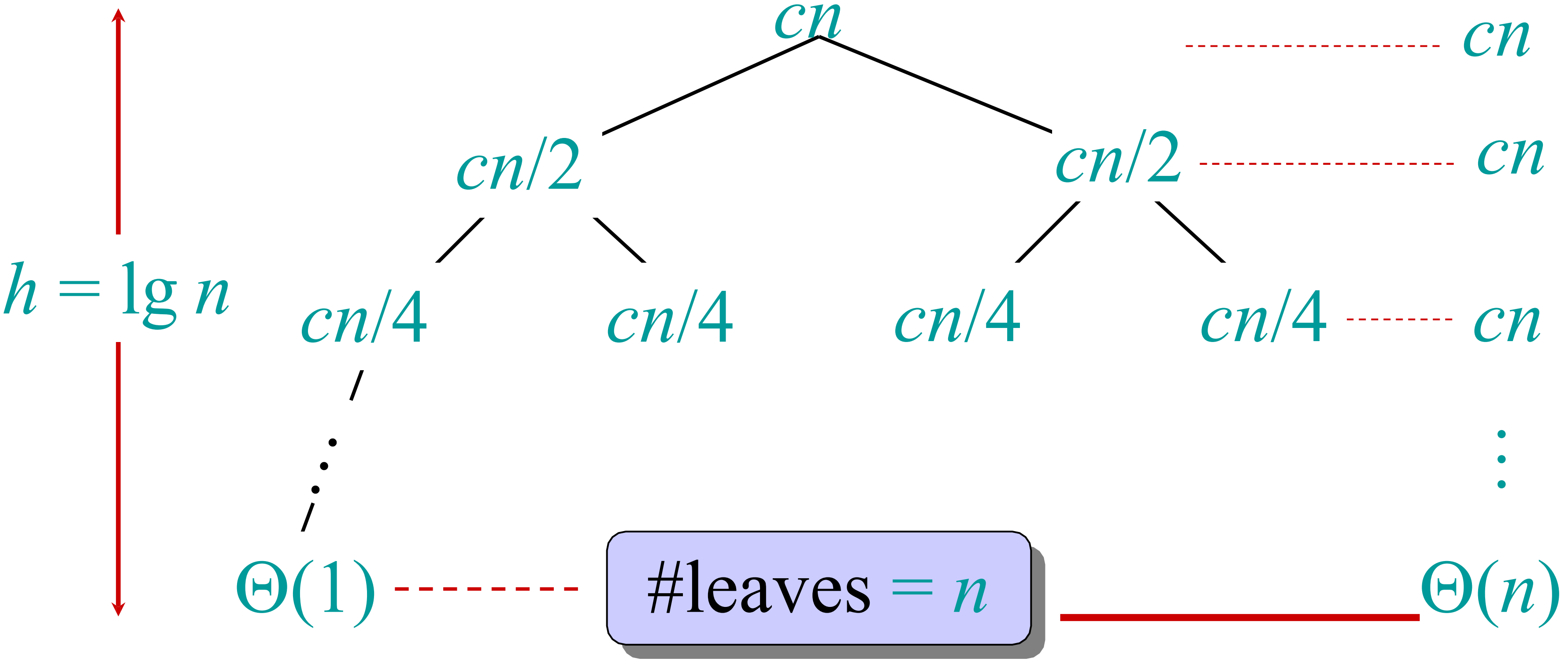
Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.





Recursion tree

Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.



cn $\quad$ cn

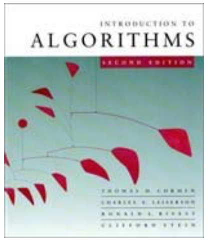
$cn/2$ $\quad$ $cn/2$ $\quad$ cn

$h = \lg n$

$cn/4$ $\quad$ $cn/4$ $\quad$ $cn/4$ $\quad$ $cn/4$ $\quad$ cn

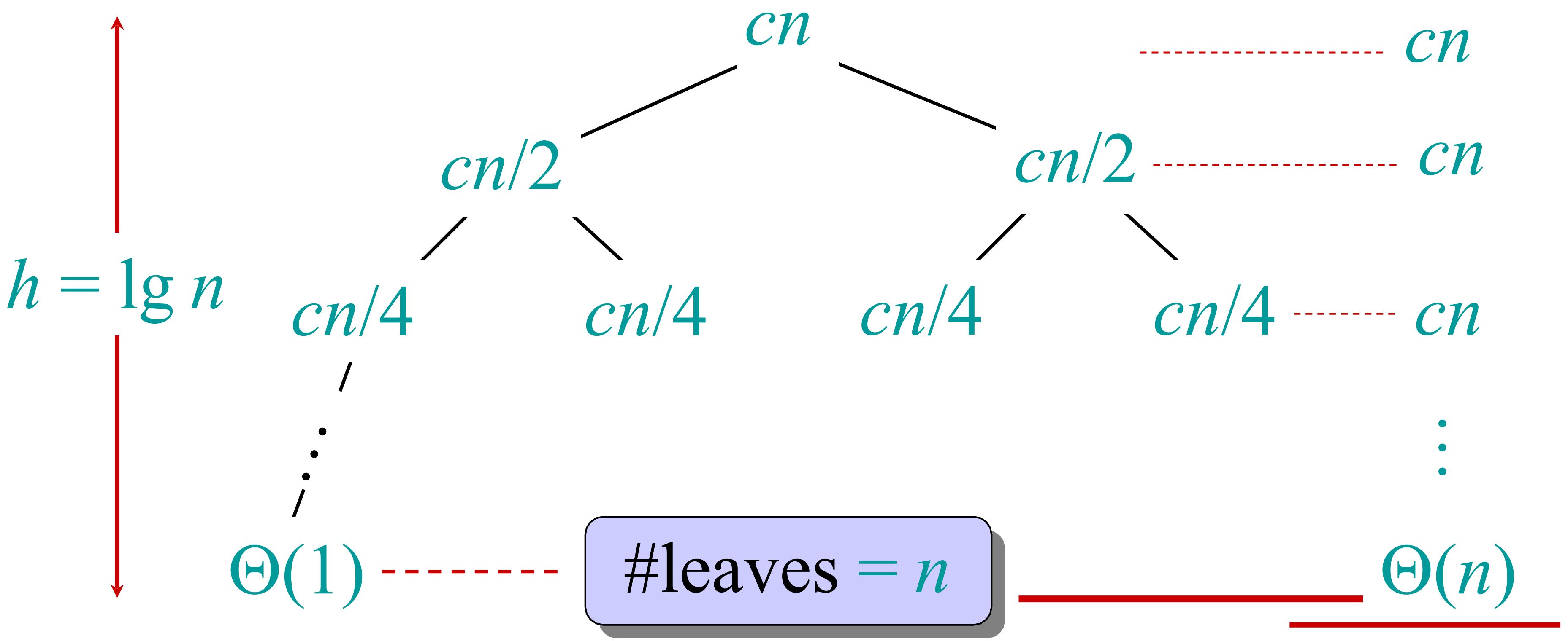
$\vdots$

$\Theta(1)$ $\quad$ #leaves $= n$ $\quad$ $\Theta(n)$



Recursion tree

Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.



cn ---------- cn

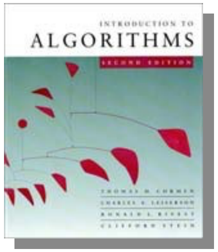
$cn/2$ $\quad$ $cn/2$ ---------- cn

$cn/4$ $\quad$ $cn/4$ $\quad$ $cn/4$ $\quad$ $cn/4$ ---------- cn

$h = \lg n$

$\Theta(1)$ ---------- #leaves $= n$ $\qquad$ $\Theta(n)$

Total $= \Theta(n \lg n)$



Merge sort

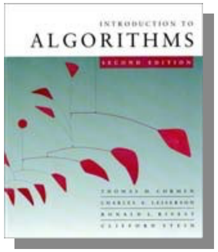
1. ***Divide:*** Trivial.
2. ***Conquer:*** Recursively sort 2 subarrays.
3. ***Combine:*** Linear-time merge.

$$T(n) = 2\,T(n/2) + \Theta(n)$$

 # subproblems

subproblem size

work dividing and combining



Master theorem

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

CASE 1: $f(n) = O(n^{\log_b a - \varepsilon})$, constant $\varepsilon > 0$

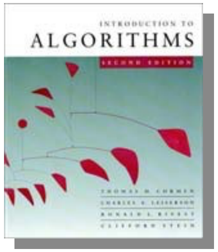
$\Rightarrow T(n) = \Theta(n^{\log_b a})$.

CASE 2: $f(n) = \Theta(n^{\log_b a})$

$\Rightarrow T(n) = \Theta(n^{\log_b a} \lg n)$.

CASE 3: $f(n) = \Omega(n^{\log_b a + \varepsilon})$, constant $\varepsilon > 0$, and regularity condition

$\Rightarrow T(n) = \Theta(f(n))$.



Master theorem

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

CASE 1: $f(n) = O(n^{\log_b a - \varepsilon})$, constant $\varepsilon > 0$

$\Rightarrow T(n) = \Theta(n^{\log_b a})$.

CASE 2: $f(n) = \Theta(n^{\log_b a})$

$\Rightarrow T(n) = \Theta(n^{\log_b a} \lg n)$.

CASE 3: $f(n) = \Omega(n^{\log_b a + \varepsilon})$, constant $\varepsilon > 0$, and regularity condition

$\Rightarrow T(n) = \Theta(f(n))$.

Merge sort: $a = 2$, $b = 2$ $\Rightarrow$ $n^{\log_b a} = n^{\log_2 2} = n$

$\Rightarrow$ CASE 2 $\Rightarrow$ $T(n) = \Theta(n \lg n)$.