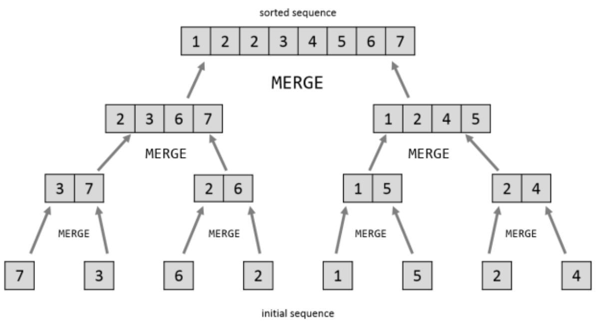
CO202 – Software Engineering – Algorithms

Divide and Conquer - Solutions

Exercise 1: Illustrate the Operations of Merge Sort

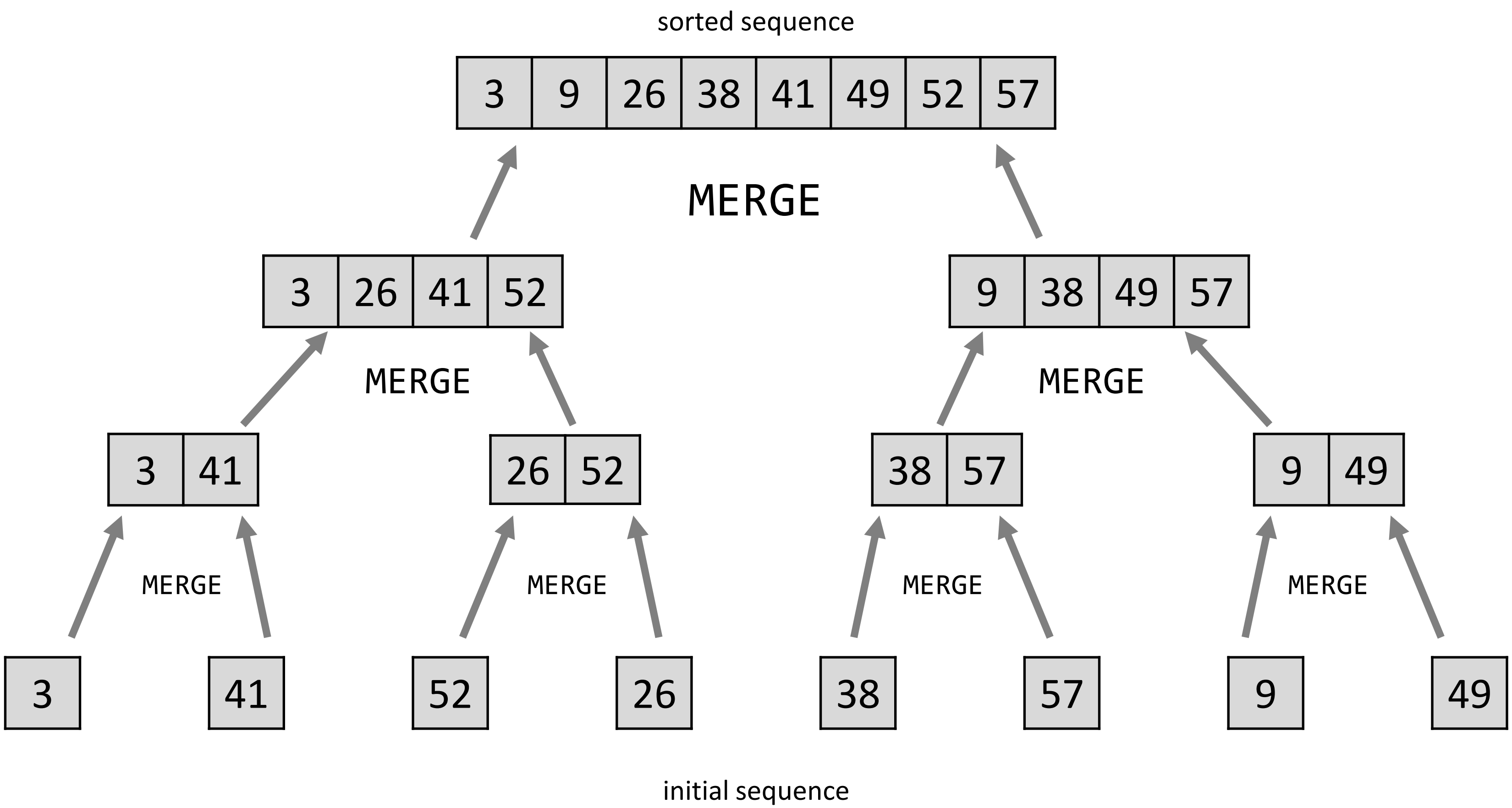
$$A = \langle 3, 41, 52, 26, 38, 57, 9, 49 \rangle$$



Exercise 1: Illustrate the Operations of Merge Sort

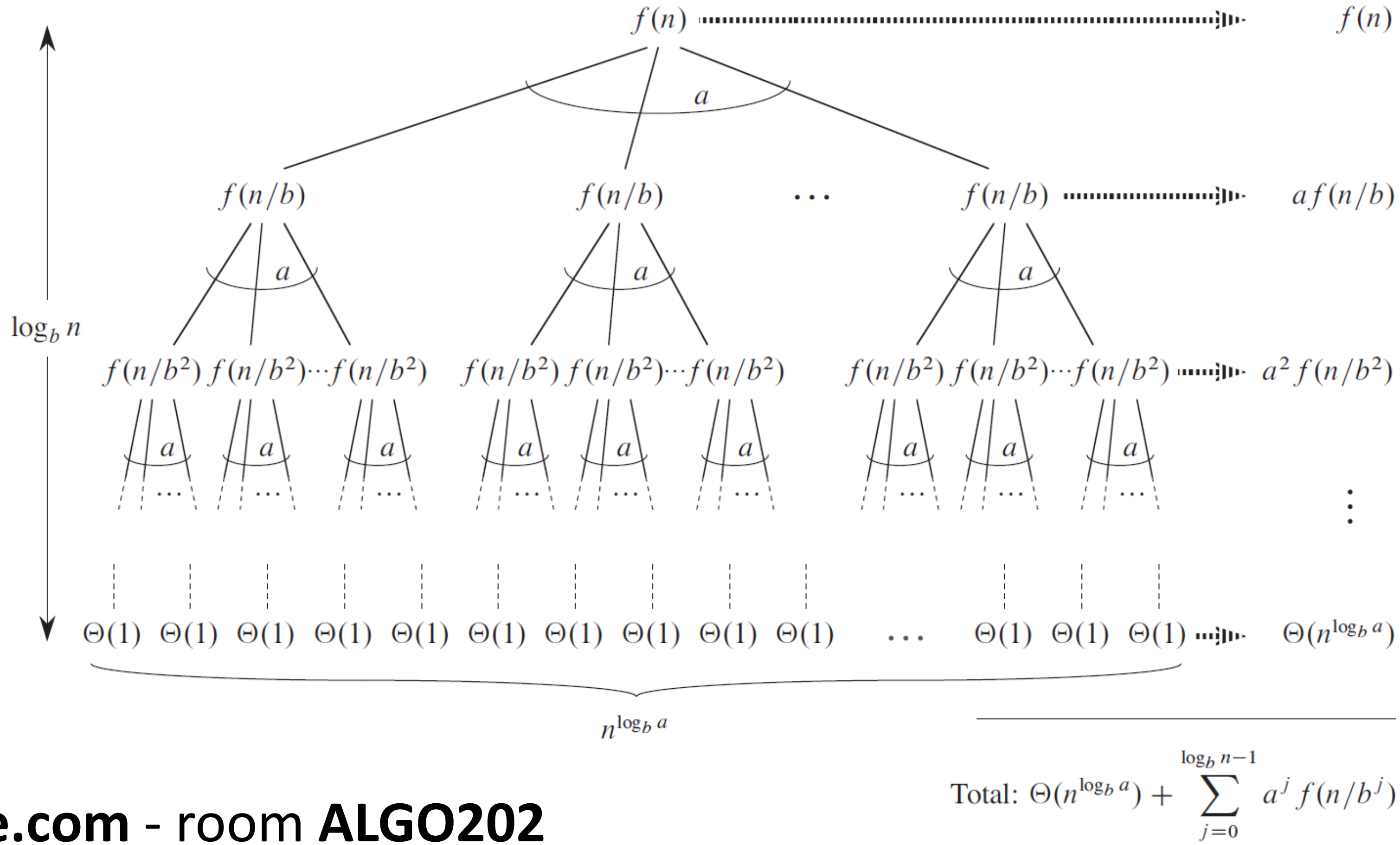
$A = \langle 3, 41, 52, 26, 38, 57, 9, 49 \rangle$

$$A = \langle 3, 41, 52, 26, 38, 57, 9, 49 \rangle$$



Exercise 2: Draw a Recursion Tree, Guess, and Verify

$$T(n) = 3T(n/4) + cn^2$$

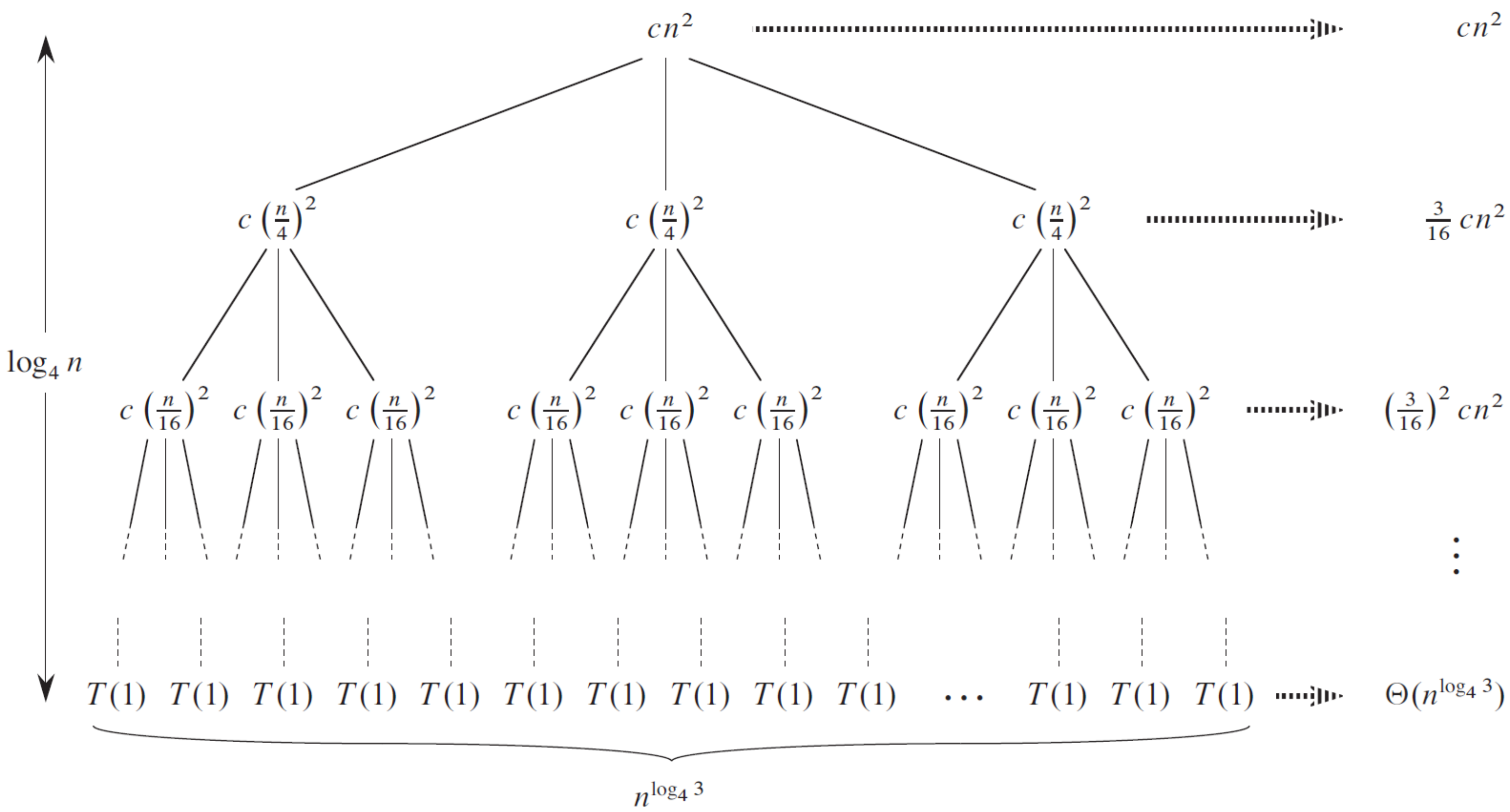


m.socrative.com - room **ALGO202**

A: $O(\lg n)$ **B**: $O(n)$ **C**: $O(n \lg n)$ **D**: $O(n^2)$ **E**: $O(2^n)$

Exercise 2: Draw a Recursion Tree, Guess, and Verify

$$T(n) = 3T(n/4) + cn^2$$



Exercise 2: Draw a Recursion Tree, Guess, and Verify

$$T(n) = 3T(n/4) + cn^2$$

Bottom level has $3^{\log_4 n} = n^{\log_4 3}$ nodes, each contributing $T(1)$, which makes $\Theta(n^{\log_4 3})$.

Adding up the costs over all levels

$$cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n-1} cn^2$$

$$= \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 < \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 = \frac{16}{13}cn^2 = O(n^2)$$

The cost for the entire tree is $O(n^2) + \Theta\left(n^{\log_4 3}\right) = \mathrm{O}(\mathrm{n}^2)$.

Exercise 2: Draw a Recursion Tree, Guess, and Verify

$T(n) = 3T(n/4) + cn^2$

Guess: $O(n^2)$

Prove that $T(n) \leq dn^2$

c is taken, so we use d

Assume this holds for all positive $m < n$,
in particular for $m \leq n/4$, yielding $T(n/4) \leq d\left((n/4)^2\right)$

By substitution:

$$
\begin{aligned}
T(n) &\leq 3(d(n/4)^2) + cn^2 \\
&= 3d\, n^2/16 + cn^2 \\
&= \frac{3}{16} dn^2 + cn^2 \\
&\leq dn^2
\end{aligned}
$$

holds for $d \geq \frac{16}{13} c$

■

Skipping the base case here.

Exercise 3: Master Method

- $T(n) = 3T(n/4) + cn^2$

1. If $d < \log_b a$, then $T(n) = \Theta\left(n^{\log_b a}\right)$.
2. If $d = \log_b a$, then $T(n) = \Theta\left(n^d \lg n\right)$.
3. If $d > \log_b a$, then $T(n) = \Theta\left(n^d\right)$.

- $T(n) = 2T\left({}^{n}/_{4}\right) + \sqrt{n}$

- $T(n) = 8T(n/2) + n^2$

- $T(n) = T(n/2) + 1$

Exercise 3: Master Method

- $T(n) = 3T(n/4) + cn^2$ $\log_4 3 \approx 0.7925$

Case 3: $d = 2 > 0.8$ so $T(n) = \Theta(n^2)$.

- $T(n) = 2T({}^{n}/_{4}) + \sqrt{n}$ $\log_4 2 = 0.5$

Case 2: $d = 0.5 = 0.5$ so $T(n) = \Theta(\sqrt{n} \lg n)$.

- $T(n) = 8T(n/2) + n^2$ $\log_2 8 = 3$

Case 1: $d = 2 < 3$ so $T(n) = \Theta(n^3)$.

- $T(n) = T(n/2) + 1$ $\log_2 1 = 0$

Case 2: $d = 0 = 0$ so $T(n) = \Theta(\lg n)$.

Exercise 4: Substitution Method

$$T(n) = 2T(n/2) + 1$$

1) Obtain the running time using the Master Method
2) Confirm with the Substitution Method

Exercise 4: Substitution Method

$T(n) = 2T(n/2) + 1$

Case 1: $d < \log_b a$ with $a = 2, b = 2, d = 0$, so $T(n) = \Theta(n)$

Substitution method will fail:

$$T(n) \leq 2c\frac{n}{2} + 1 = cn + 1 \nleq cn$$

Need to subtract a lower order term d and show $T(n) \leq cn - d$

$$T(n) \leq 2\left(c\frac{n}{2} - d\right) + 1 = cn - 2d + 1 \leq cn - d$$

which holds for $d \geq 1$

Exercise 5: Divide and Conquer

1) Write in pseudo-code a recursive function $f(x, n) = x^n$ for powering a number using divide-and-conquer.

Hint:

- $x^n = x^{n/2} \cdot x^{n/2}$ for even n
- $x^n = x^{(n-1)/2} \cdot x^{(n-1)/2} \cdot x$ for odd n

2) Show that the running time complexity is $O(\lg n)$.

Exercise 5: Divide and Conquer

1) Write in pseudo-code a recursive function $f(x, n) = x^n$ for powering a number using divide-and-conquer.

Hint:

- $x^n = x^{n/2} \cdot x^{n/2}$ for even n
- $x^n = x^{(n-1)/2} \cdot x^{(n-1)/2} \cdot x$ for odd n

```
RECURSIVE-POWER(x, n)
if n == 1
    return x
if n is even
    return RECURSIVE-POWER(x, n/2) * RECURSIVE-POWER(x, n/2)
else
    return RECURSIVE-POWER(x, (n-1)/2) * RECURSIVE-POWER(x, (n-1)/2) * x
```

Exercise 5: Divide and Conquer

1) Write in pseudo-code a recursive function $f(x, n) = x^n$ for powering a number using divide-and-conquer.

```
RECURSIVE-POWER(x, n)
if n == 1
    return x
if n is even
    y = RECURSIVE-POWER(x, n/2)
    return y*y
else
    y = RECURSIVE-POWER(x, (n-1)/2)
    return y*y*x
```

Exercise 5: Divide and Conquer

2) Show that the running time complexity is $O(\lg n)$.

$$T(n) = T(n/2) + c$$

Case 2: $d = \log_b a$ with $a = 1, b = 2, d = 0$, so $T(n) = \Theta(\lg n)$