

CSE 613: Parallel Programming

Lecture 7

(Analyzing Divide-and-Conquer Algorithms)

Rezaul A. Chowdhury

Department of Computer Science

SUNY Stony Brook

Spring 2017

A Useful Recurrence

Consider the following recurrence:

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise;} \end{cases}$$

where, $a \geq 1$ and $b > 1$.

Arises frequently in the analyses of *divide-and-conquer* algorithms.

Recall the following from the analyses of QSort (quicksort) in lecture 1.

Serial: $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$

Parallel (with serial partition): $T(n) = T\left(\frac{n}{2}\right) + \Theta(n)$

Parallel (with parallel partition): $T(n) = T\left(\frac{n}{2}\right) + \Theta(\log n)$

How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$

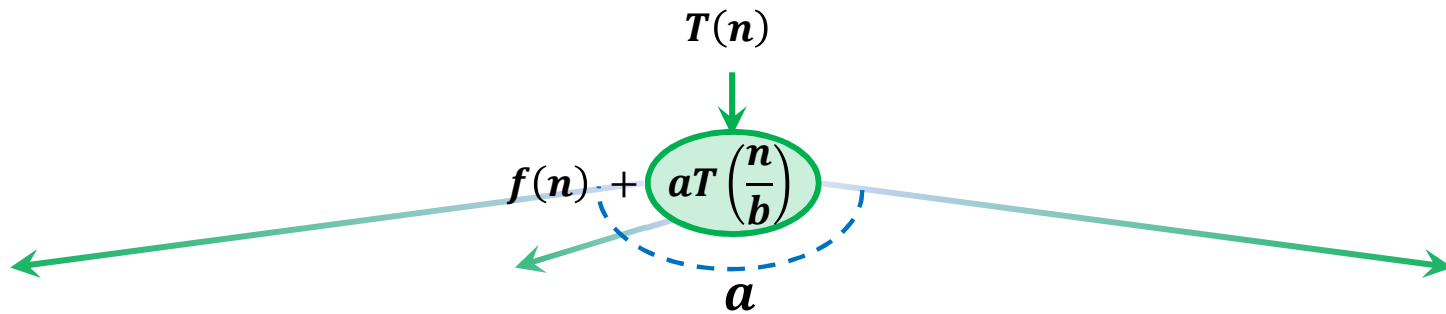
How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$

$$\begin{array}{c} T(n) \\ \downarrow \\ f(n) + aT\left(\frac{n}{b}\right) \end{array}$$

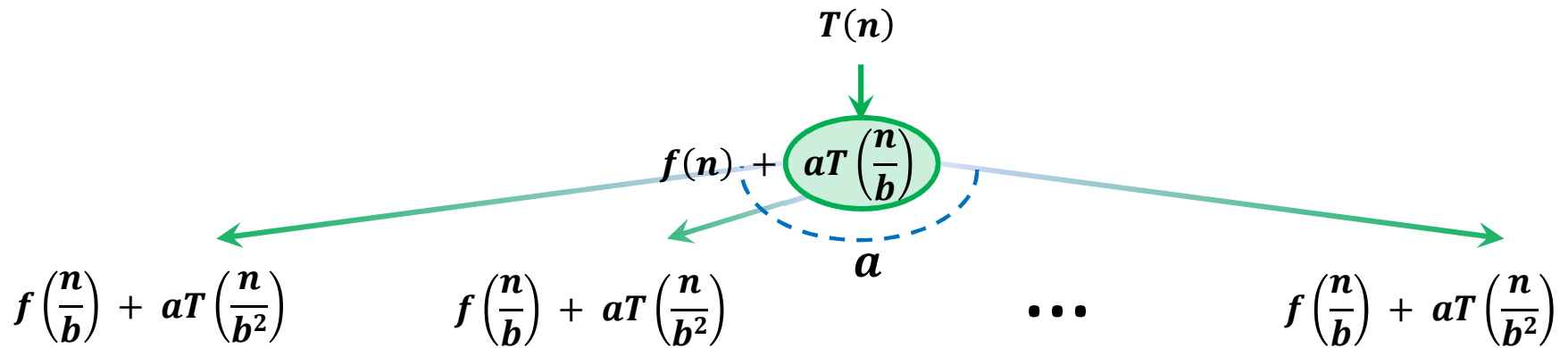
How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



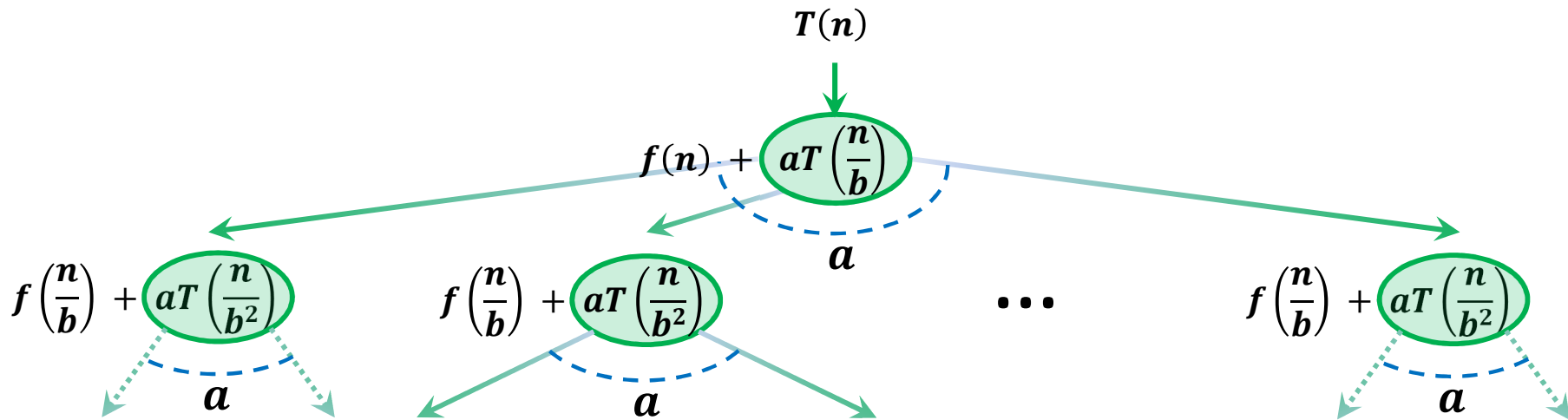
How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



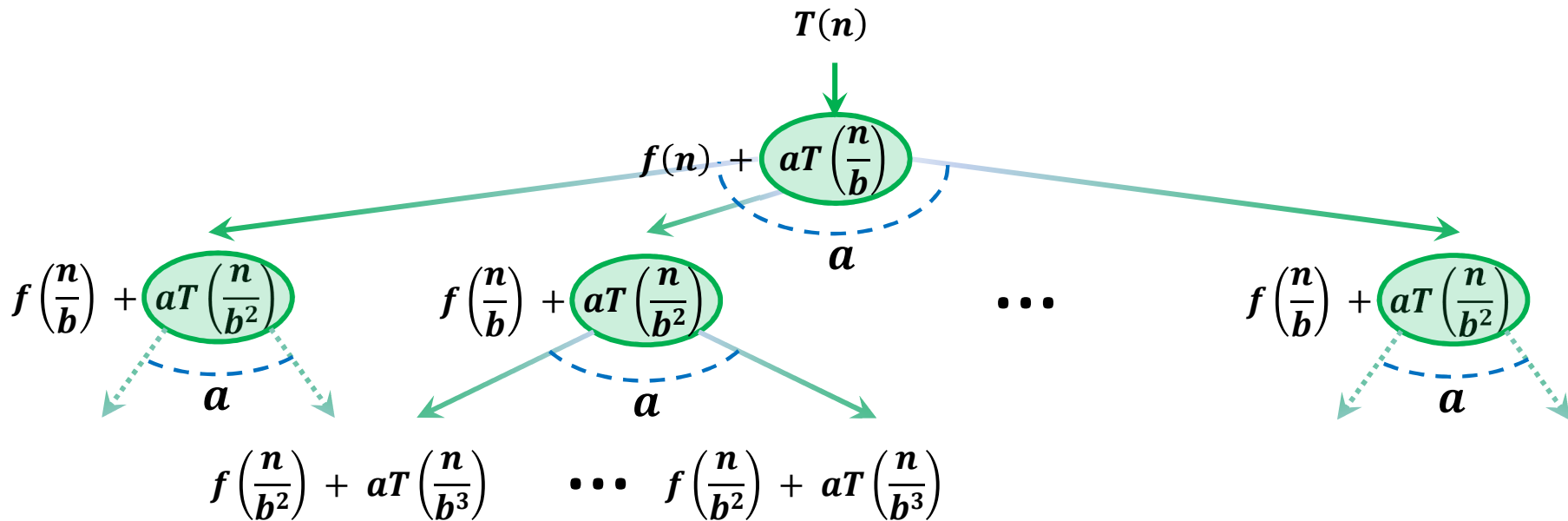
How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



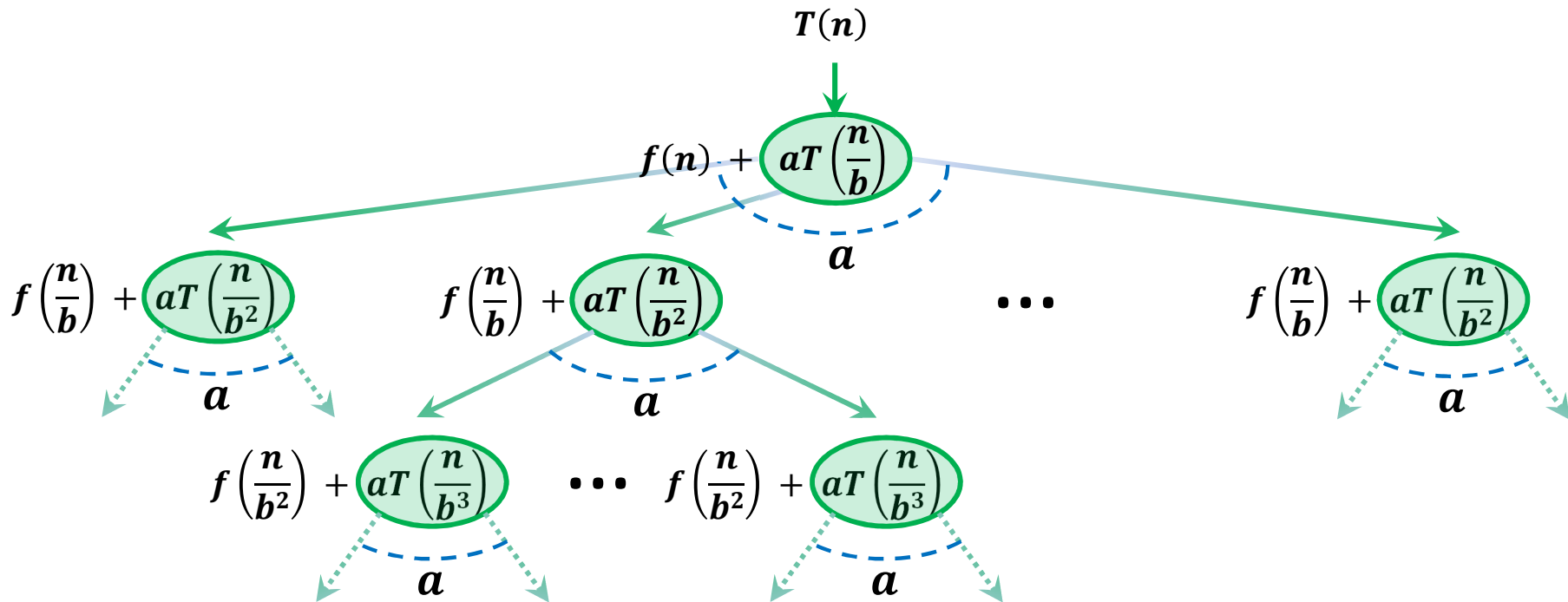
How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



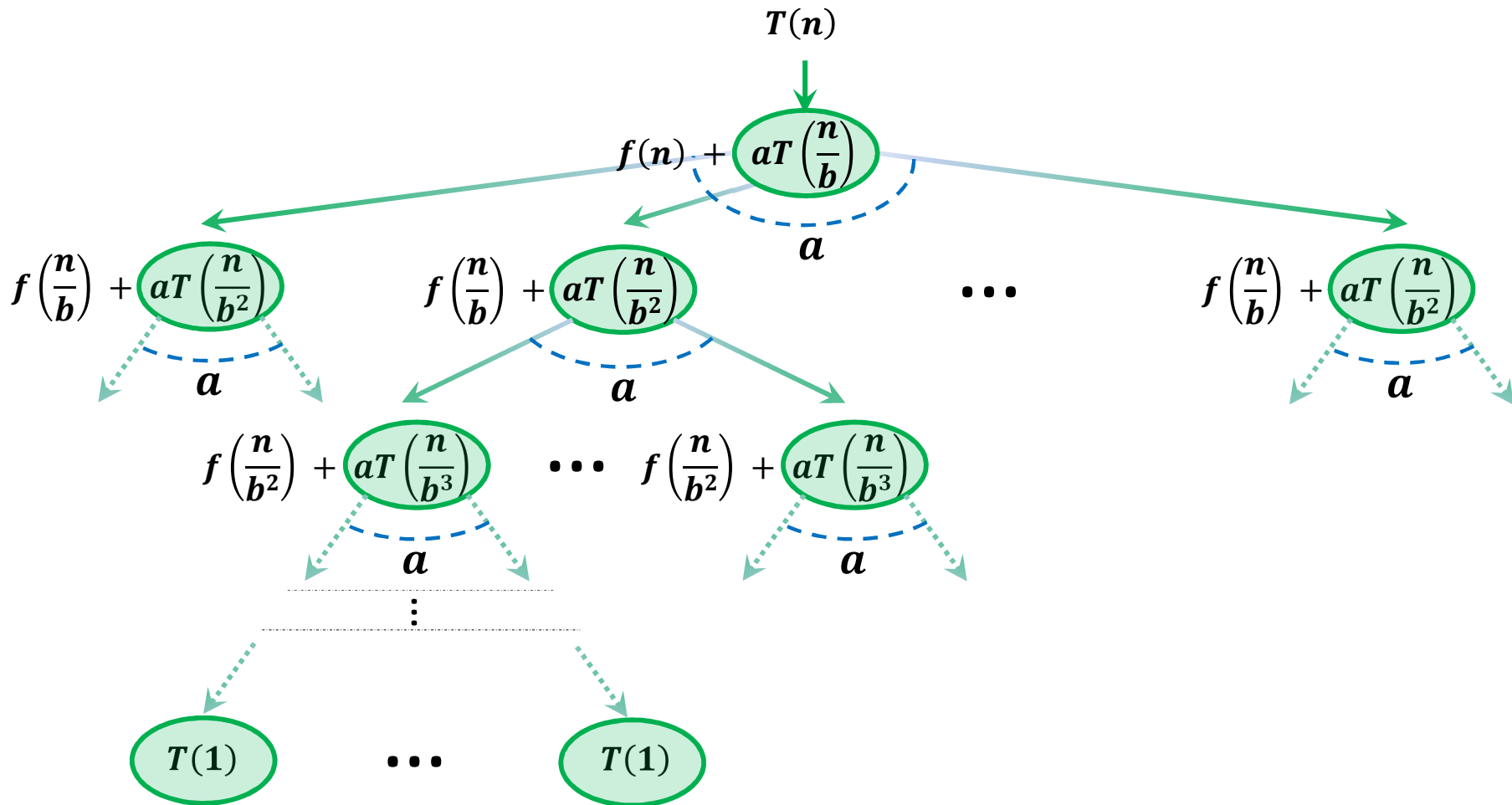
How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



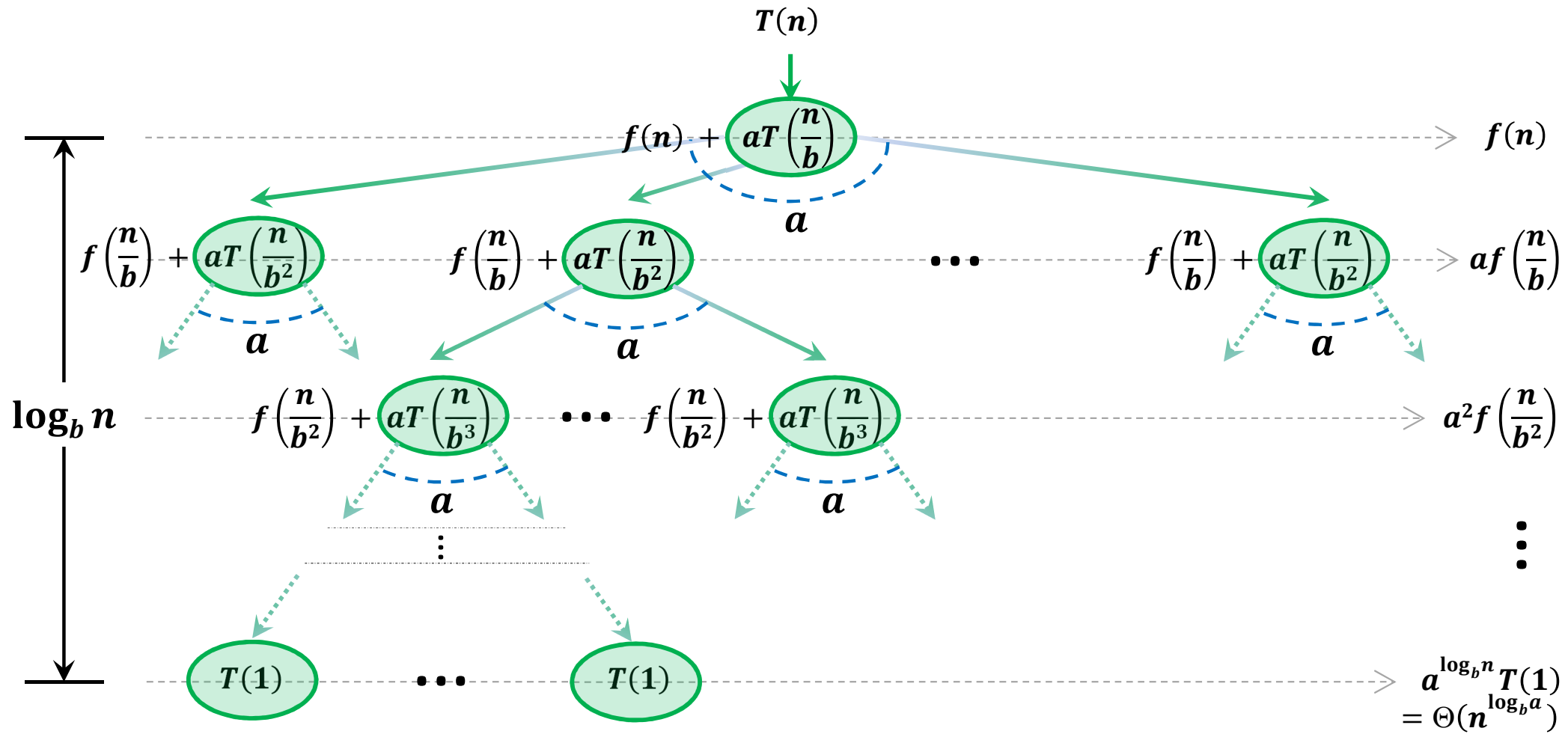
How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



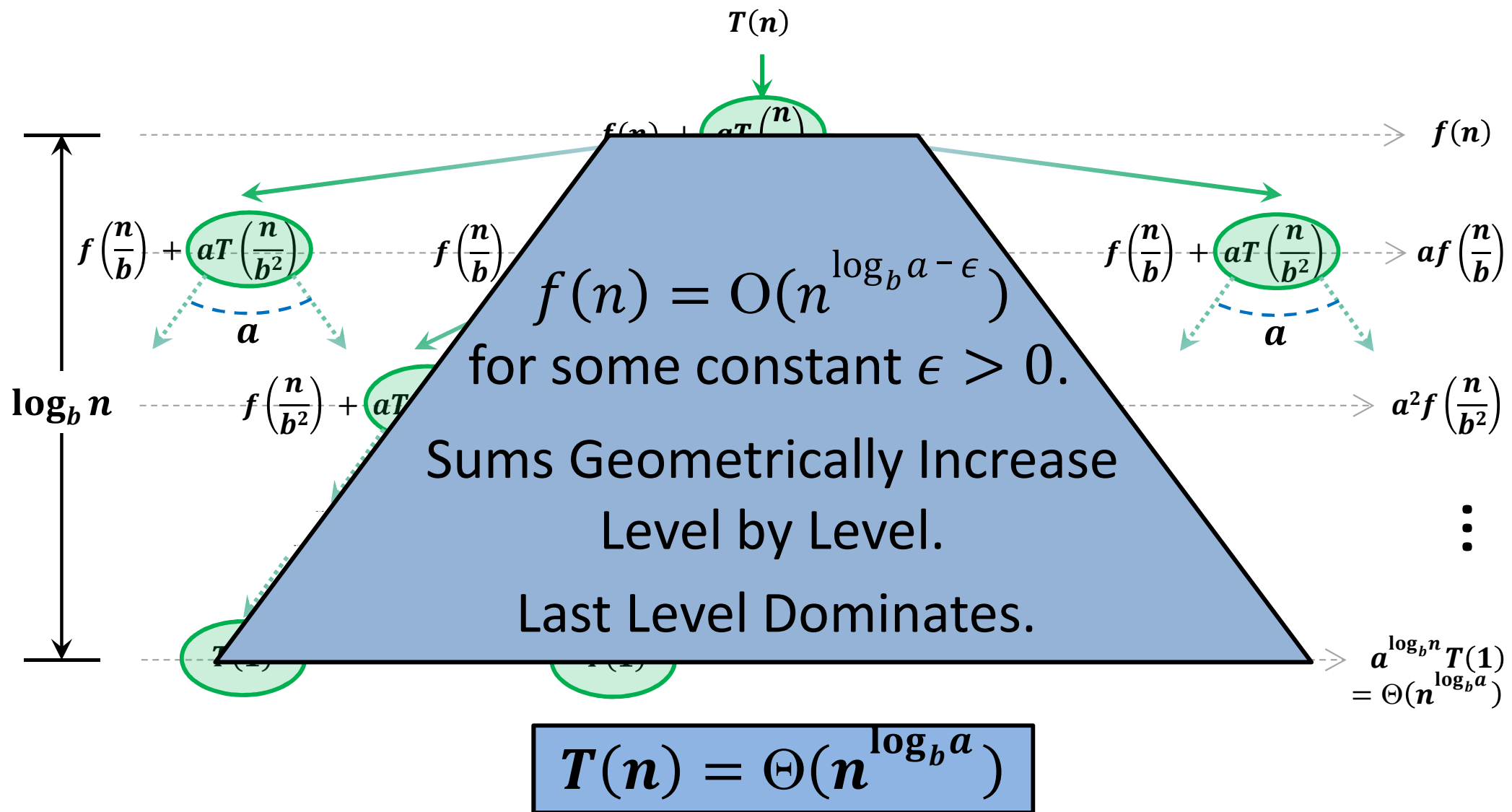
How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



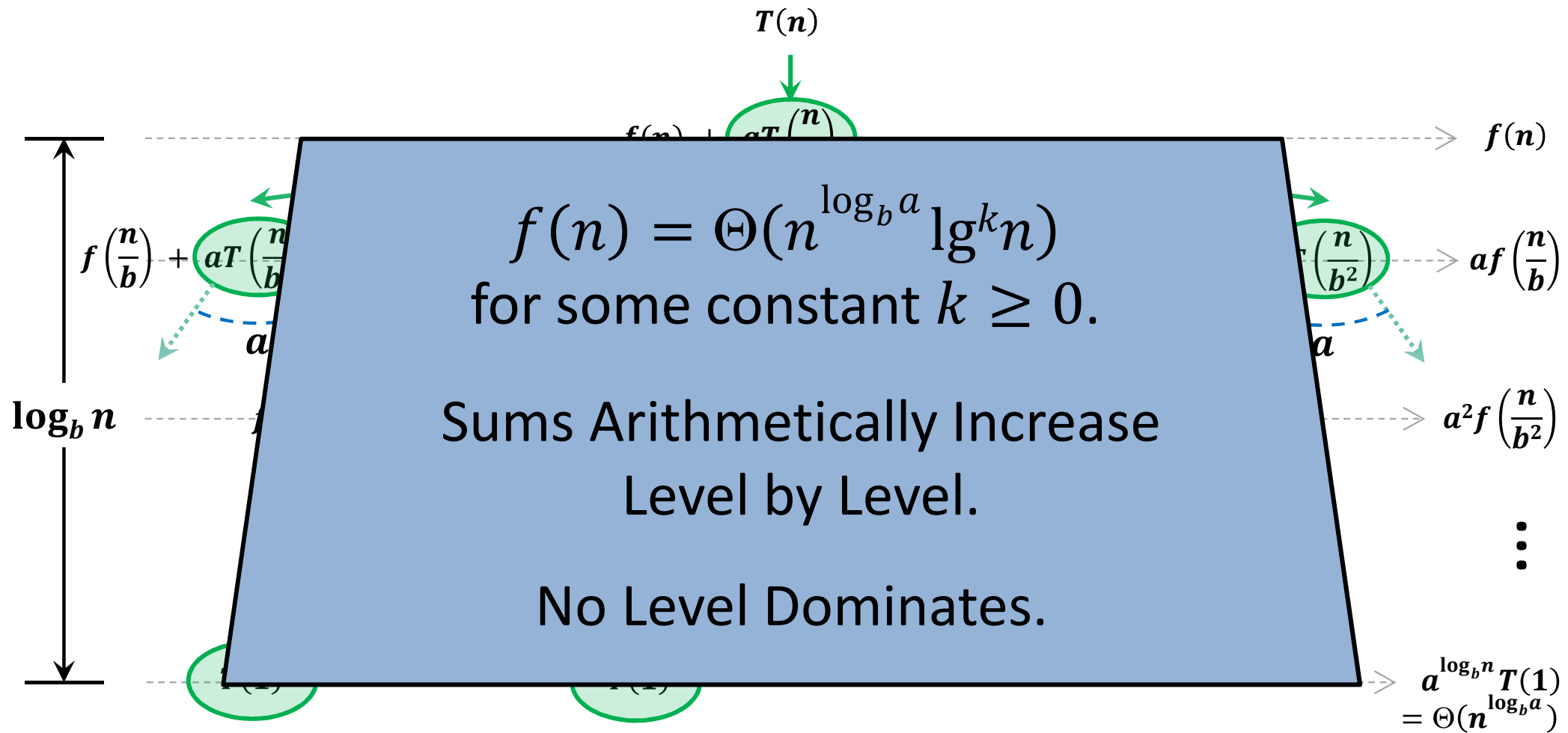
How the Recurrence Unfolds: Case 1

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



How the Recurrence Unfolds: Case 2

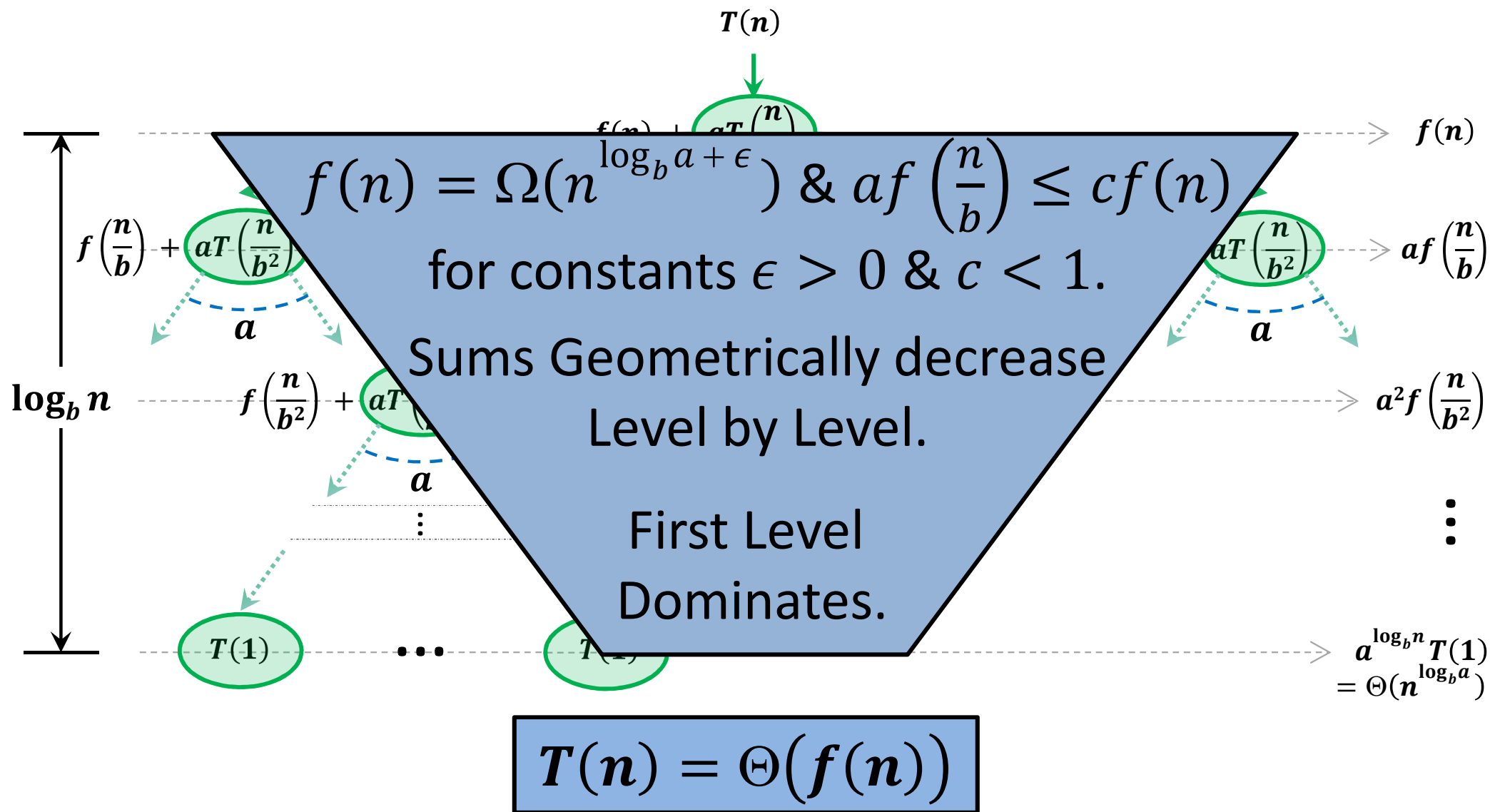
$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



$$T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$$

How the Recurrence Unfolds: Case 3

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



The Master Theorem

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise } (a \geq 1, b > 1). \end{cases}$$

Case 1: $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$

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

Case 2: $f(n) = \Theta(n^{\log_b a} \lg^k n)$ for some constant $k \geq 0$.

$$T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$$

Case 3: $f(n) = \Omega(n^{\log_b a + \epsilon})$ and $af\left(\frac{n}{b}\right) \leq cf(n)$
for constants $\epsilon > 0$ and $c < 1$.

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

Back to QSort Complexities

Now let's try the QSort (quicksort) recurrences from lecture 1.

Serial: $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$

Master Theorem Case 2: $T(n) = \Theta(n \log n)$

Parallel (with serial partition): $T(n) = T\left(\frac{n}{2}\right) + \Theta(n)$

Master Theorem Case 3: $T(n) = \Theta(n)$

Parallel (with parallel partition): $T(n) = T\left(\frac{n}{2}\right) + \Theta(\log n)$

Master Theorem Case 2: $T(n) = \Theta(\log^2 n)$

More Example Applications of Master Theorem

Karatsuba's Algorithm: $T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n)$

Master Theorem Case 1: $T(n) = \Theta(n^{\log_2 3})$

Strassen's Matrix Multiplication: $T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2)$

Master Theorem Case 1: $T(n) = \Theta(n^{\log_2 7})$

Fast Fourier Transform: $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$

Master Theorem Case 2: $T(n) = \Theta(n \log n)$

Recurrences not Solvable using the Master Theorem

Example 1: $T(n) = \sqrt{n} T\left(\frac{n}{2}\right) + n$

$a = \sqrt{n}$ is not a constant

Example 2: $T(n) = 2T\left(\frac{n}{\log n}\right) + n^2$

$b = \log n$ is not a constant

Example 3: $T(n) = \frac{1}{2}T\left(\frac{n}{2}\right) + n^2$

$a = \frac{1}{2}$ is not ≥ 1

Example 4: $T(n) = 2T\left(\frac{4n}{3}\right) + n$

$b = \frac{3}{4}$ is not > 1 .

Recurrences not Solvable using the Master Theorem

Example 5: $T(n) = 3T\left(\frac{n}{2}\right) - n$

$f(n) = -n$ is not positive

Example 6: $T(n) = 2T\left(\frac{n}{2}\right) + n^2 \sin n$

violates regularity condition of case 3

Example 7: $T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$

$f(n) = O(n^{\log_b a})$, but $\neq O(n^{\log_b a - \epsilon})$ for any constant $\epsilon > 0$

Example 8: $T(n) = T\left(\frac{n}{2}\right) + 2T\left(\frac{n}{4}\right) + n$

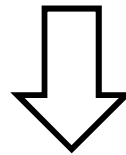
a and b are not fixed

Multithreaded Matrix Multiplication

Parallel Iterative MM

Iter-MM (Z, X, Y) { X, Y, Z are $n \times n$ matrices,
where n is a positive integer }

1. *for* $i \leftarrow 1$ *to* n *do*
2. *for* $j \leftarrow 1$ *to* n *do*
3. $Z[i][j] \leftarrow 0$
4. *for* $k \leftarrow 1$ *to* n *do*
5. $Z[i][j] \leftarrow Z[i][j] + X[i][k] \cdot Y[k][j]$



Par-Iter-MM (Z, X, Y) { X, Y, Z are $n \times n$ matrices,
where n is a positive integer }

1. *parallel for* $i \leftarrow 1$ *to* n *do*
2. *parallel for* $j \leftarrow 1$ *to* n *do*
3. $Z[i][j] \leftarrow 0$
4. *for* $k \leftarrow 1$ *to* n *do*
5. $Z[i][j] \leftarrow Z[i][j] + X[i][k] \cdot Y[k][j]$

Parallel Iterative MM

Par-Iter-MM (Z, X, Y) { X, Y, Z are $n \times n$ matrices,
where n is a positive integer }

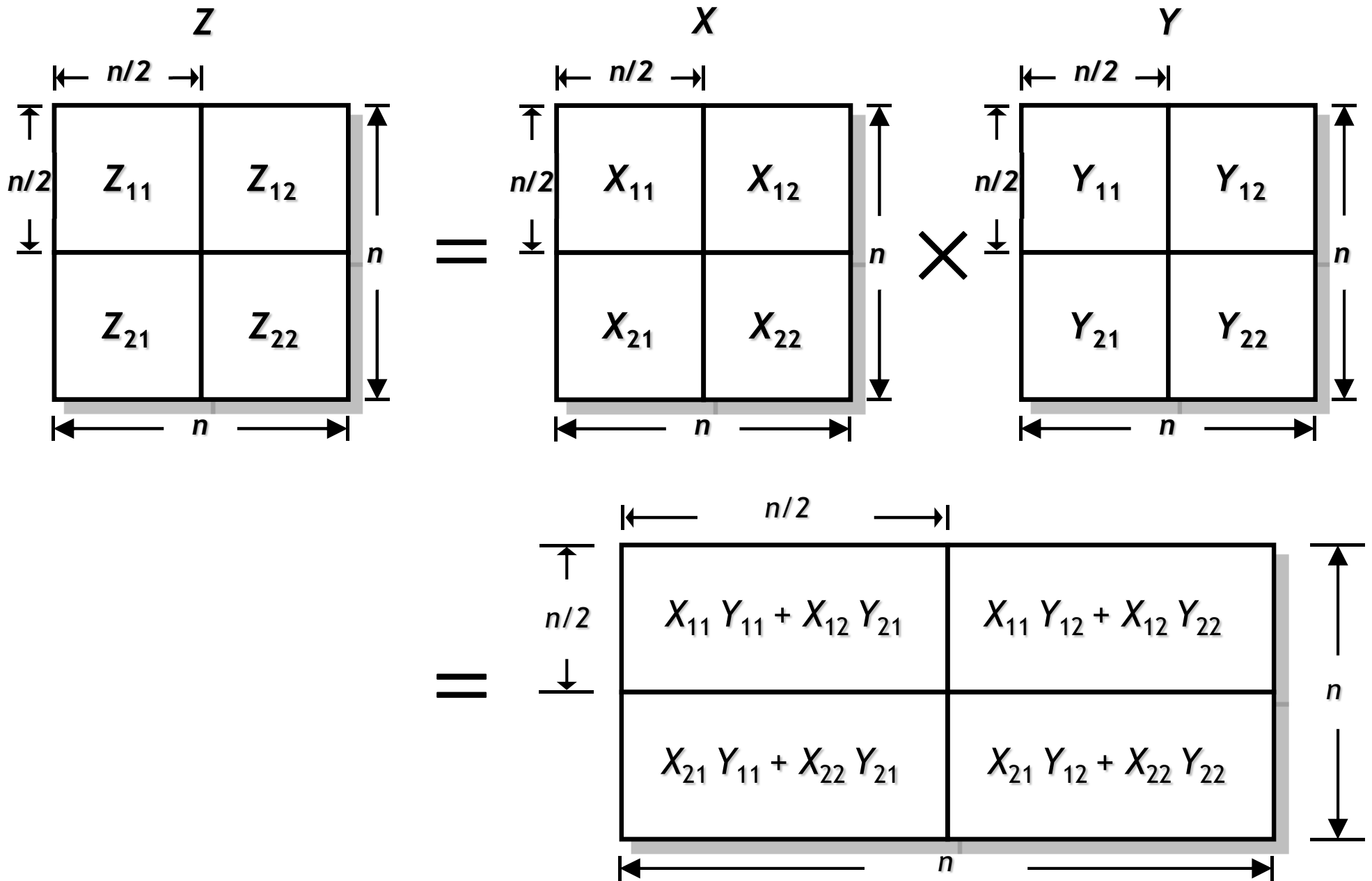
1. *parallel for* $i \leftarrow 1$ to n do
2. *parallel for* $j \leftarrow 1$ to n do
3. $Z[i][j] \leftarrow 0$
4. *for* $k \leftarrow 1$ to n do
5. $Z[i][j] \leftarrow Z[i][j] + X[i][k] \cdot Y[k][j]$

Work: $T_1(n) = \Theta(n^3)$

Span: $T_\infty(n) = \Theta(\log n + \log n + n) = \Theta(n)$

Parallelism: $\frac{T_1(n)}{T_\infty(n)} = \Theta(n^2)$

Parallel Recursive MM



Parallel Recursive MM

Par-Rec-MM (Z, X, Y) { X, Y, Z are $n \times n$ matrices,
where $n = 2^k$ for integer $k \geq 0$ }

1. *if* $n = 1$ *then*
2. $Z \leftarrow Z + X \cdot Y$
3. *else*
4. *spawn* *Par-Rec-MM* (Z_{11}, X_{11}, Y_{11})
5. *spawn* *Par-Rec-MM* (Z_{12}, X_{11}, Y_{12})
6. *spawn* *Par-Rec-MM* (Z_{21}, X_{21}, Y_{11})
7. *Par-Rec-MM* (Z_{22}, X_{21}, Y_{12})
8. *sync*
9. *spawn* *Par-Rec-MM* (Z_{11}, X_{12}, Y_{21})
10. *spawn* *Par-Rec-MM* (Z_{12}, X_{12}, Y_{22})
11. *spawn* *Par-Rec-MM* (Z_{21}, X_{22}, Y_{21})
12. *Par-Rec-MM* (Z_{22}, X_{22}, Y_{22})
13. *sync*
14. *endif*

Parallel Recursive MM

Par-Rec-MM (Z, X, Y) { X, Y, Z are $n \times n$ matrices,
where $n = 2^k$ for integer $k \geq 0$ }

1. *if* $n = 1$ *then*
2. $Z \leftarrow Z + X \cdot Y$
3. *else*
4. *spawn* *Par-Rec-MM* (Z_{11} , X_{11} , Y_{11})
5. *spawn* *Par-Rec-MM* (Z_{12} , X_{11} , Y_{12})
6. *spawn* *Par-Rec-MM* (Z_{21} , X_{21} , Y_{11})
7. *Par-Rec-MM* (Z_{22} , X_{21} , Y_{12})
8. *sync*
9. *spawn* *Par-Rec-MM* (Z_{11} , X_{12} , Y_{21})
10. *spawn* *Par-Rec-MM* (Z_{12} , X_{12} , Y_{22})
11. *spawn* *Par-Rec-MM* (Z_{21} , X_{22} , Y_{21})
12. *Par-Rec-MM* (Z_{22} , X_{22} , Y_{22})
13. *sync*
14. *endif*

Work:

$$T_1(n) = \begin{cases} \Theta(1), & \text{if } n = 1, \\ 8T_1\left(\frac{n}{2}\right) + \Theta(1), & \text{otherwise.} \end{cases}$$

$$= \Theta(n^3) \quad [\text{MT Case 1}]$$

Span:

$$T_\infty(n) = \begin{cases} \Theta(1), & \text{if } n = 1, \\ 2T_\infty\left(\frac{n}{2}\right) + \Theta(1), & \text{otherwise.} \end{cases}$$

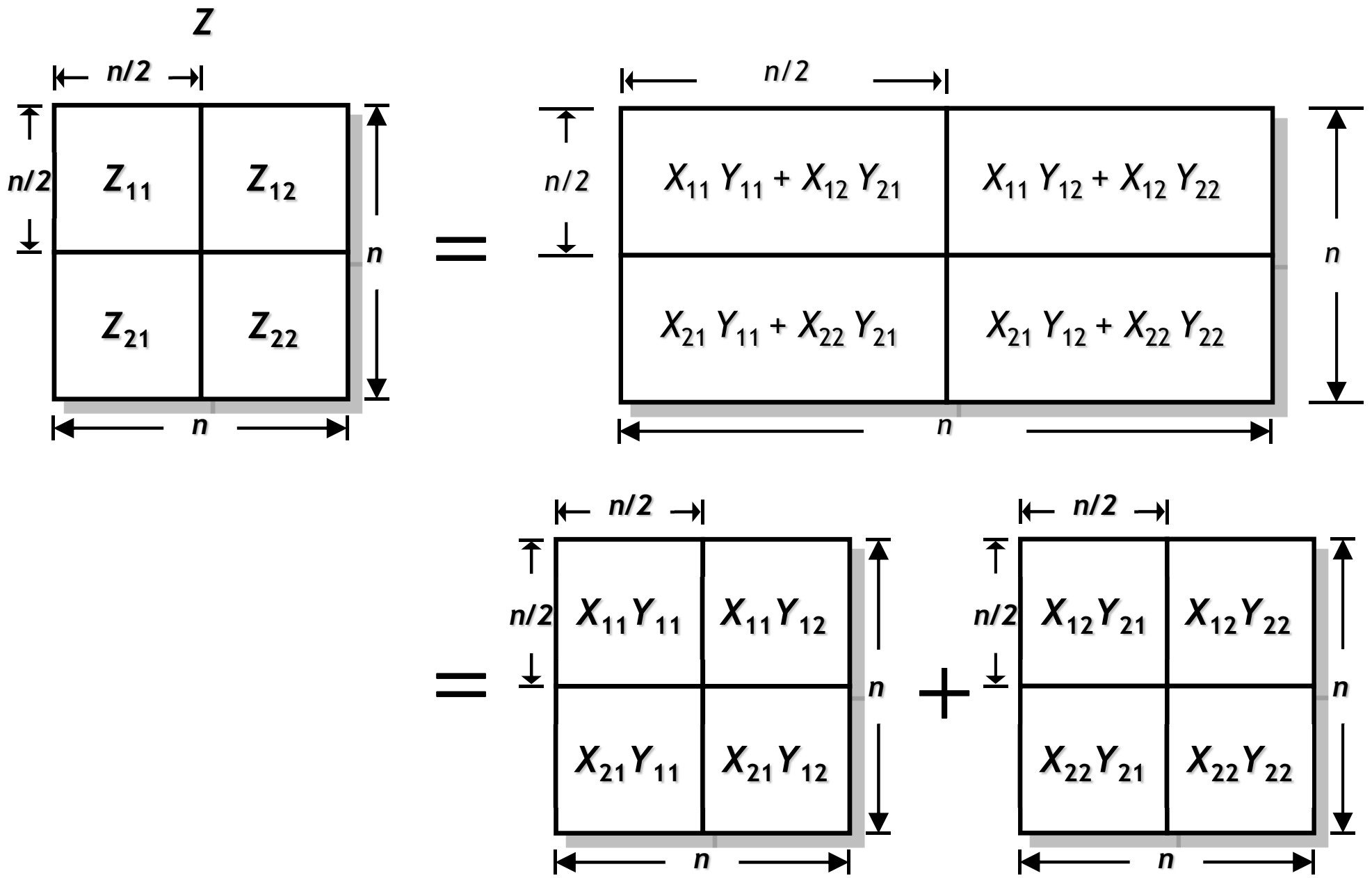
$$= \Theta(n) \quad [\text{MT Case 1}]$$

Parallelism: $\frac{T_1(n)}{T_\infty(n)} = \Theta(n^2)$

Additional Space:

$$s_\infty(n) = \Theta(1)$$

Recursive MM with More Parallelism



Recursive MM with More Parallelism

```
Par-Rec-MM2 ( Z, X, Y )   { X, Y, Z are  $n \times n$  matrices,  
                           where  $n = 2^k$  for integer  $k \geq 0$  }  
  
1. if  $n = 1$  then  
2.    $Z \leftarrow Z + X \cdot Y$   
3. else           {  $T$  is a temporary  $n \times n$  matrix }  
  
4.   spawn Par-Rec-MM2 (  $Z_{11}$ ,  $X_{11}$ ,  $Y_{11}$  )  
5.   spawn Par-Rec-MM2 (  $Z_{12}$ ,  $X_{11}$ ,  $Y_{12}$  )  
6.   spawn Par-Rec-MM2 (  $Z_{21}$ ,  $X_{21}$ ,  $Y_{11}$  )  
7.   spawn Par-Rec-MM2 (  $Z_{22}$ ,  $X_{21}$ ,  $Y_{12}$  )  
8.   spawn Par-Rec-MM2 (  $T_{11}$ ,  $X_{12}$ ,  $Y_{21}$  )  
9.   spawn Par-Rec-MM2 (  $T_{12}$ ,  $X_{12}$ ,  $Y_{22}$  )  
10.  spawn Par-Rec-MM2 (  $T_{21}$ ,  $X_{22}$ ,  $Y_{21}$  )  
11.      Par-Rec-MM2 (  $T_{22}$ ,  $X_{22}$ ,  $Y_{22}$  )  
12.  sync  
13.  parallel for  $i \leftarrow 1$  to  $n$  do  
14.    parallel for  $j \leftarrow 1$  to  $n$  do  
15.       $Z[i][j] \leftarrow Z[i][j] + T[i][j]$   
16.  endif
```

Recursive MM with More Parallelism

Par-Rec-MM2 (Z, X, Y) { X, Y, Z are n × n matrices,
where n = 2^k for integer k ≥ 0 }

1. *if* n = 1 *then*
2. Z ← Z + X · Y
3. *else* { T is a temporary n × n matrix }
4. *spawn* *Par-Rec-MM2* (Z₁₁, X₁₁, Y₁₁)
5. *spawn* *Par-Rec-MM2* (Z₁₂, X₁₁, Y₁₂)
6. *spawn* *Par-Rec-MM2* (Z₂₁, X₂₁, Y₁₁)
7. *spawn* *Par-Rec-MM2* (Z₂₂, X₂₁, Y₁₂)
8. *spawn* *Par-Rec-MM2* (T₁₁, X₁₂, Y₂₁)
9. *spawn* *Par-Rec-MM2* (T₁₂, X₁₂, Y₂₂)
10. *spawn* *Par-Rec-MM2* (T₂₁, X₂₂, Y₂₁)
11. *Par-Rec-MM2* (T₂₂, X₂₂, Y₂₂)
12. *sync*
13. *parallel for* i ← 1 *to* n *do*
14. *parallel for* j ← 1 *to* n *do*
15. Z[i][j] ← Z[i][j] + T[i][j]
16. *endif*

Work:

$$T_1(n) = \begin{cases} \Theta(1), & \text{if } n = 1, \\ 8T_1\left(\frac{n}{2}\right) + \Theta(n^2), & \text{otherwise.} \end{cases}$$

$$= \Theta(n^3) \quad [\text{MT Case 1}]$$

Span:

$$T_\infty(n) = \begin{cases} \Theta(1), & \text{if } n = 1, \\ T_\infty\left(\frac{n}{2}\right) + \Theta(\log n), & \text{otherwise.} \end{cases}$$

$$= \Theta(\log^2 n) \quad [\text{MT Case 2}]$$

Parallelism: $\frac{T_1(n)}{T_\infty(n)} = \Theta\left(\frac{n^3}{\log^2 n}\right)$

Additional Space:

$$s_\infty(n) = \begin{cases} \Theta(1), & \text{if } n = 1, \\ 8s_\infty\left(\frac{n}{2}\right) + \Theta(n^2), & \text{otherwise.} \end{cases}$$

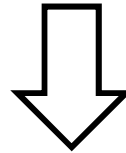
$$= \Theta(n^3) \quad [\text{MT Case 1}]$$

Multithreaded Merge Sort

Parallel Merge Sort

Merge-Sort (A, p, r) { sort the elements in $A[p \dots r]$ }

1. *if* $p < r$ *then*
2. $q \leftarrow \lfloor (p+r) / 2 \rfloor$
3. *Merge-Sort* (A, p, q)
4. *Merge-Sort* ($A, q+1, r$)
5. *Merge* (A, p, q, r)



Par-Merge-Sort (A, p, r) { sort the elements in $A[p \dots r]$ }

1. *if* $p < r$ *then*
2. $q \leftarrow \lfloor (p+r) / 2 \rfloor$
3. *spawn* *Merge-Sort* (A, p, q)
4. *Merge-Sort* ($A, q+1, r$)
5. *sync*
6. *Merge* (A, p, q, r)

Parallel Merge Sort

Par-Merge-Sort (A, p, r) { sort the elements in $A[p \dots r]$ }

1. *if* $p < r$ *then*
2. $q \leftarrow \lfloor (p+r) / 2 \rfloor$
3. *spawn* *Merge-Sort* (A, p, q)
4. *Merge-Sort* ($A, q+1, r$)
5. *sync*
6. *Merge* (A, p, q, r)

$$\text{Work: } T_1(n) = \begin{cases} \Theta(1), & \text{if } n = 1, \\ 2T_1\left(\frac{n}{2}\right) + \Theta(n), & \text{otherwise.} \end{cases}$$

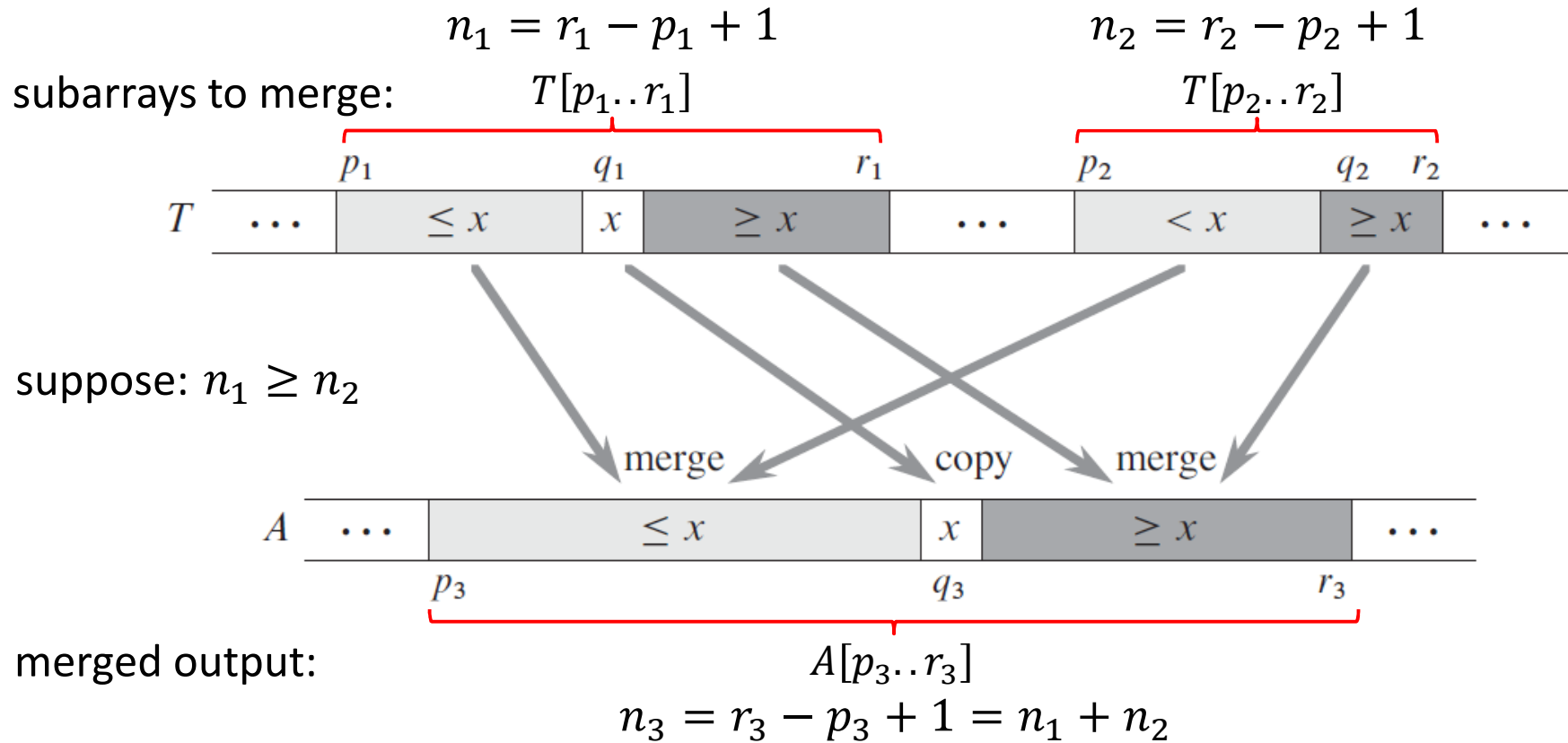
$$= \Theta(n \log n) \quad [\text{MT Case 2}]$$

$$\text{Span: } T_\infty(n) = \begin{cases} \Theta(1), & \text{if } n = 1, \\ T_\infty\left(\frac{n}{2}\right) + \Theta(n), & \text{otherwise.} \end{cases}$$

$$= \Theta(n) \quad [\text{MT Case 3}]$$

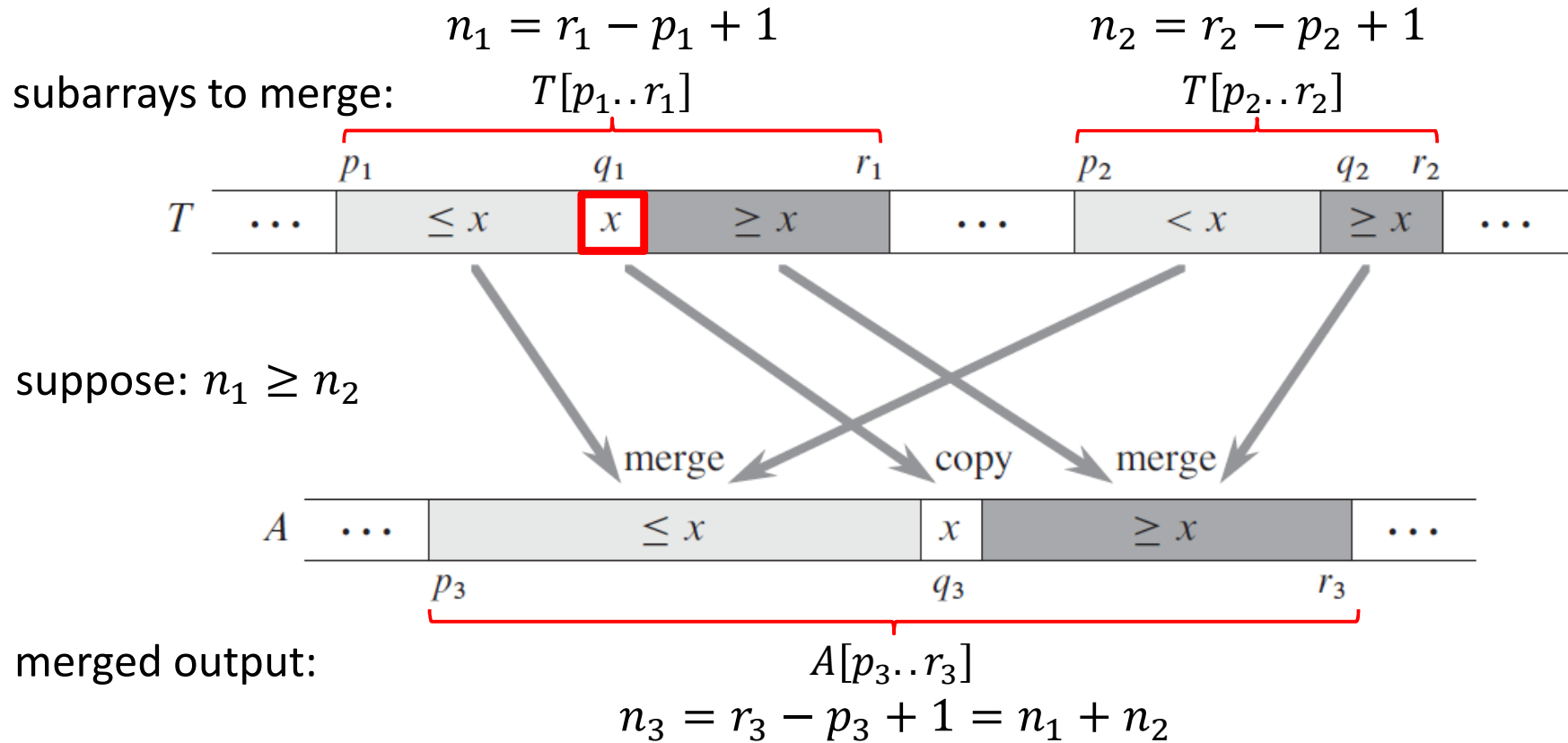
$$\text{Parallelism: } \frac{T_1(n)}{T_\infty(n)} = \Theta(\log n)$$

Parallel Merge



Source: Cormen et al.,
 "Introduction to Algorithms",
 3rd Edition

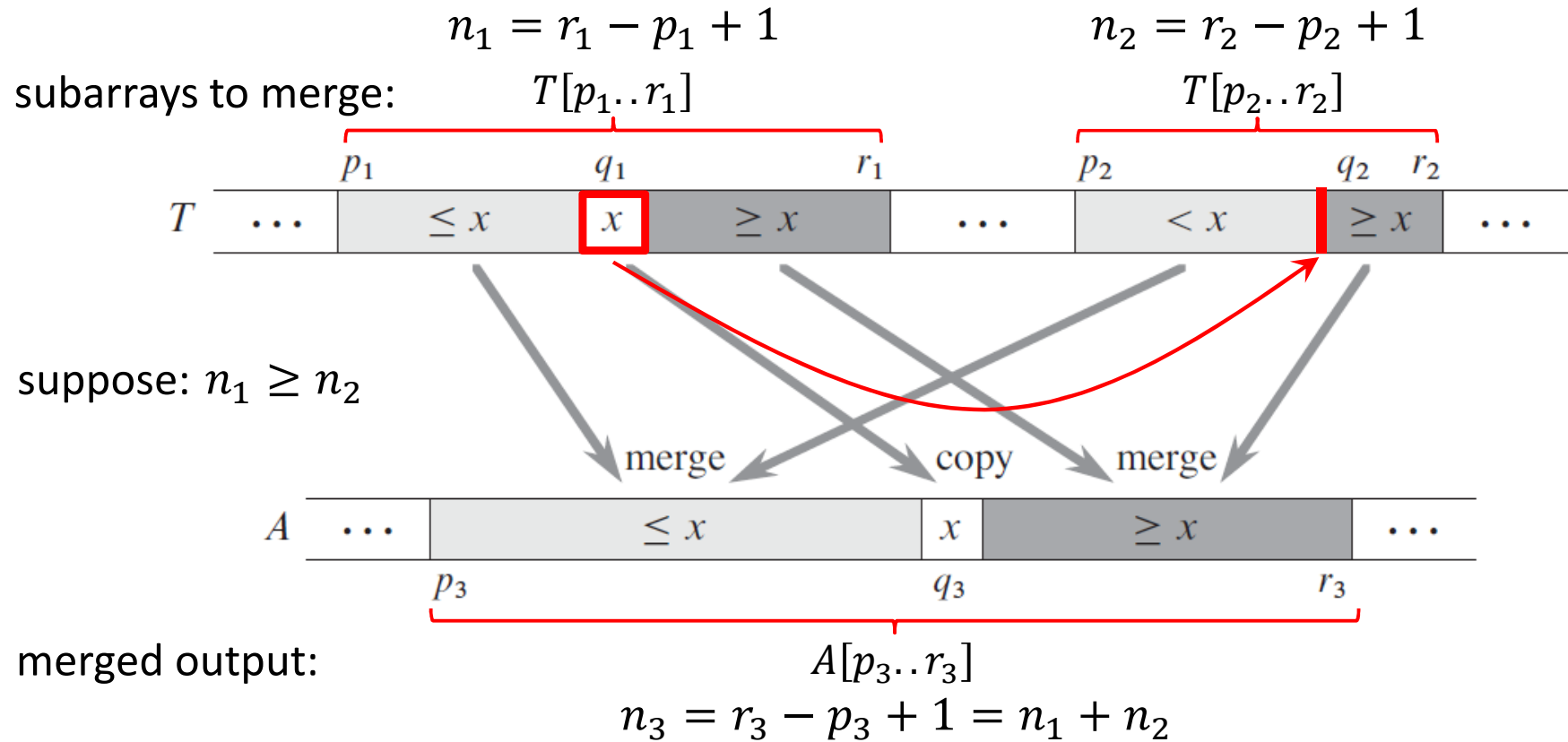
Parallel Merge



Source: Cormen et al.,
 "Introduction to Algorithms",
 3rd Edition

Step 1: Find $x = T[q_1]$, where q_1 is the midpoint of $T[p_1..r_1]$

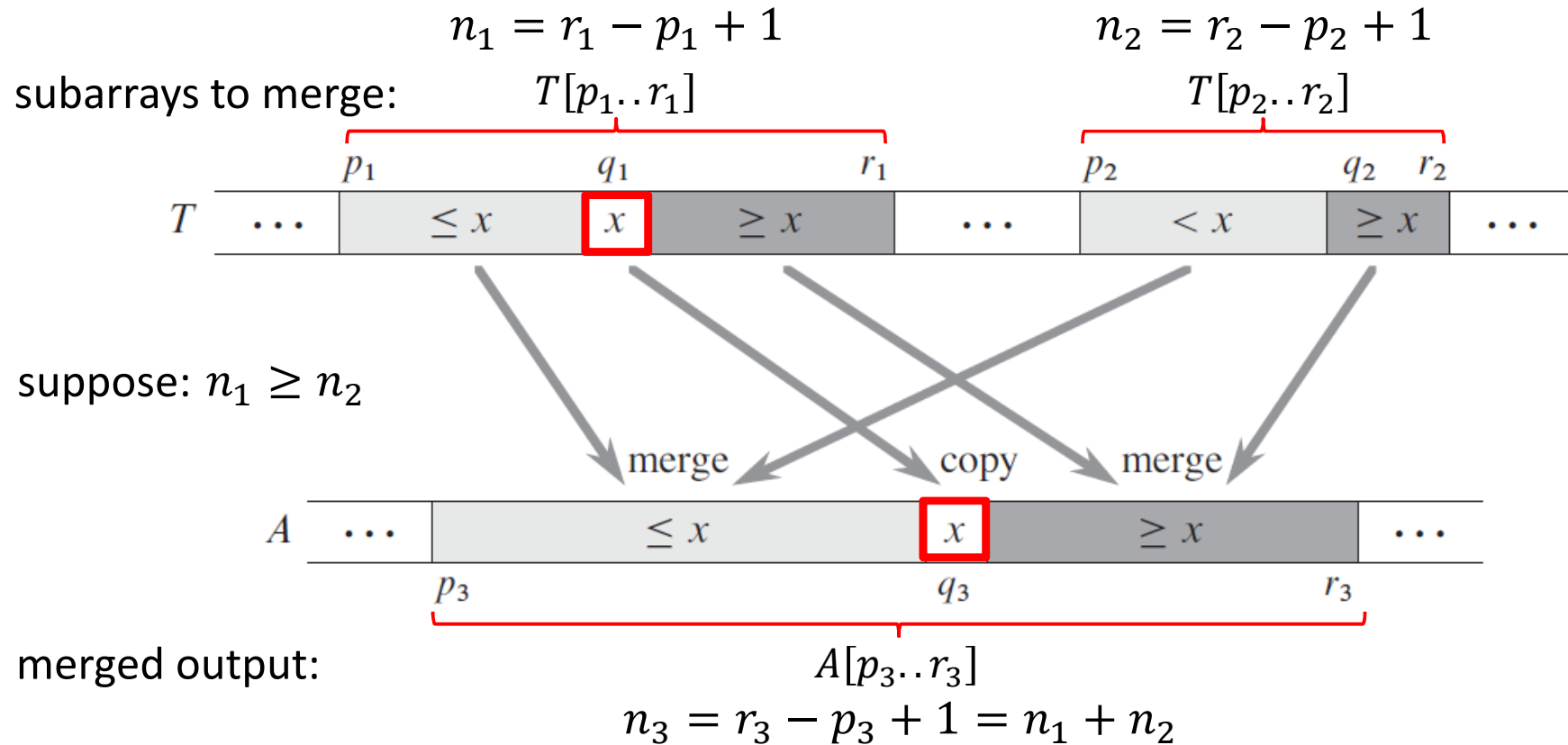
Parallel Merge



Source: Cormen et al.,
 "Introduction to Algorithms",
 3rd Edition

Step 2: Use binary search to find the index q_2 in subarray $T[p_2..r_2]$ so that the subarray would still be sorted if we insert x between $T[q_2 - 1]$ and $T[q_2]$

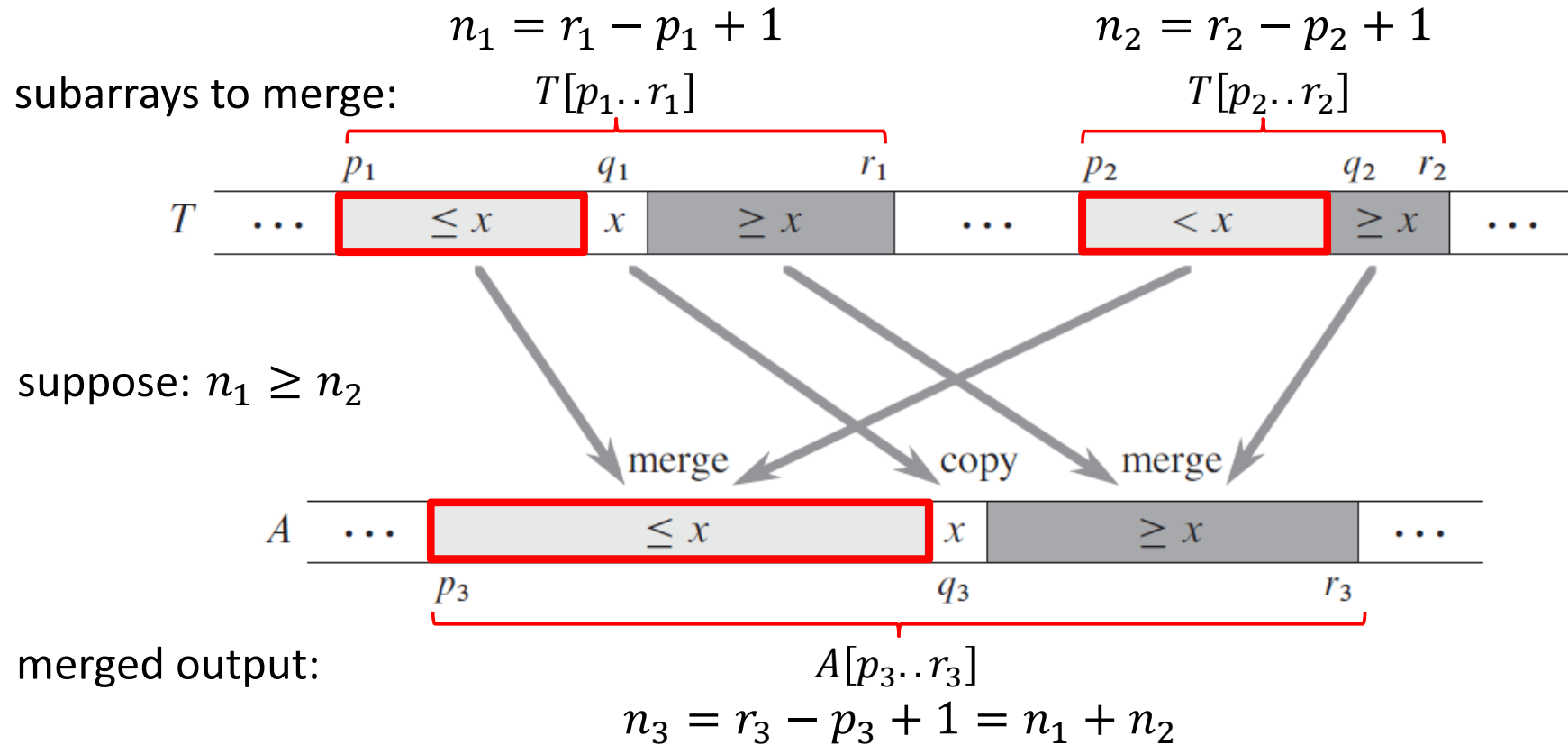
Parallel Merge



Source: Cormen et al.,
 "Introduction to Algorithms",
 3rd Edition

Step 3: Copy x to $A[q_3]$, where $q_3 = p_3 + (q_1 - p_1) + (q_2 - p_2)$

Parallel Merge

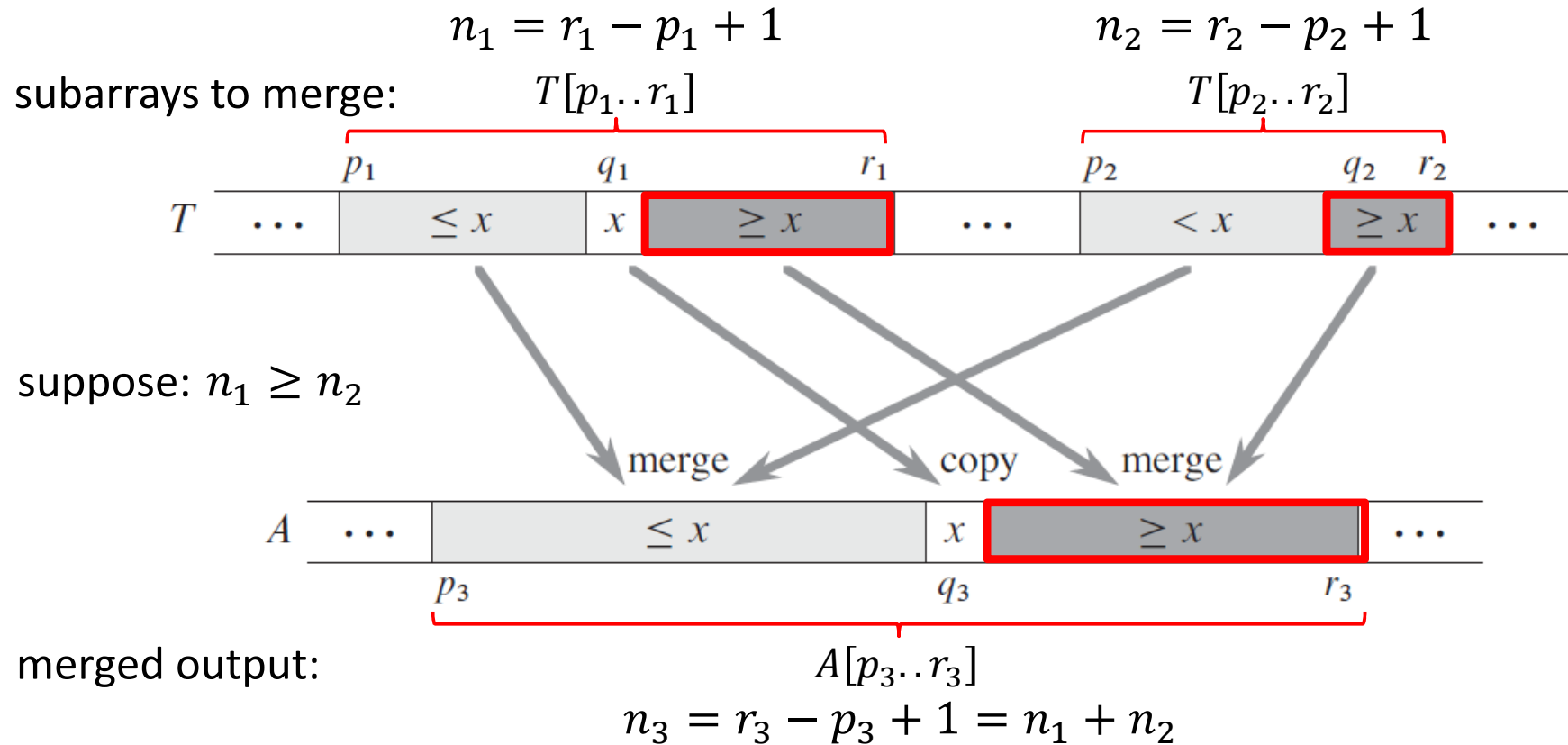


Source: Cormen et al.,
 "Introduction to Algorithms",
 3rd Edition

Perform the following two steps in parallel.

Step 4(a): Recursively merge $T[p_1..q_1 - 1]$ with $T[p_2..q_2 - 1]$,
 and place the result into $A[p_3..q_3 - 1]$

Parallel Merge



Source: Cormen et al.,
 "Introduction to Algorithms",
 3rd Edition

Perform the following two steps in parallel.

Step 4(a): Recursively merge $T[p_1..q_1 - 1]$ with $T[p_2..q_2 - 1]$, and place the result into $A[p_3..q_3 - 1]$

Step 4(b): Recursively merge $T[q_1 + 1..r_1]$ with $T[q_2 + 1..r_2]$, and place the result into $A[q_3 + 1..r_3]$

Parallel Merge

Par-Merge ($T, p_1, r_1, p_2, r_2, A, p_3$)

1. $n_1 \leftarrow r_1 - p_1 + 1, n_2 \leftarrow r_2 - p_2 + 1$
2. *if* $n_1 < n_2$ *then*
3. $p_1 \leftrightarrow p_2, r_1 \leftrightarrow r_2, n_1 \leftrightarrow n_2$
4. *if* $n_1 = 0$ *then return*
5. *else*
6. $q_1 \leftarrow \lfloor (p_1 + r_1) / 2 \rfloor$
7. $q_2 \leftarrow \text{Binary-Search} (T[q_1], T, p_2, r_2)$
8. $q_3 \leftarrow p_3 + (q_1 - p_1) + (q_2 - p_2)$
9. $A[q_3] \leftarrow T[q_1]$
10. *spawn* *Par-Merge* ($T, p_1, q_1-1, p_2, q_2-1, A, p_3$)
11. *Par-Merge* ($T, q_1+1, r_1, q_2+1, r_2, A, q_3+1$)
12. *sync*

Parallel Merge

Par-Merge ($T, p_1, r_1, p_2, r_2, A, p_3$)

1. $n_1 \leftarrow r_1 - p_1 + 1, n_2 \leftarrow r_2 - p_2 + 1$
2. *if* $n_1 < n_2$ *then*
3. $p_1 \leftrightarrow p_2, r_1 \leftrightarrow r_2, n_1 \leftrightarrow n_2$
4. *if* $n_1 = 0$ *then return*
5. *else*
6. $q_1 \leftarrow \lfloor (p_1 + r_1) / 2 \rfloor$
7. $q_2 \leftarrow \text{Binary-Search} (T[q_1], T, p_2, r_2)$
8. $q_3 \leftarrow p_3 + (q_1 - p_1) + (q_2 - p_2)$
9. $A[q_3] \leftarrow T[q_1]$
10. *spawn* *Par-Merge* ($T, p_1, q_1-1, p_2, q_2-1, A, p_3$)
11. *Par-Merge* ($T, q_1+1, r_1, q_2+1, r_2, A, q_3+1$)
12. *sync*

We have,

$$n_2 \leq n_1 \Rightarrow 2n_2 \leq n_1 + n_2 = n$$

In the worst case, a recursive call in lines 9-10 merges half the elements of $T[p_1..r_1]$ with all elements of $T[p_2..r_2]$.

Hence, #elements involved in such a call:

$$\left\lfloor \frac{n_1}{2} \right\rfloor + n_2 \leq \frac{n_1}{2} + \frac{n_2}{2} + \frac{n_2}{2} = \frac{n_1 + n_2}{2} + \frac{2n_2}{4} \leq \frac{n}{2} + \frac{n}{4} = \frac{3n}{4}$$

Parallel Merge

Par-Merge ($T, p_1, r_1, p_2, r_2, A, p_3$)

1. $n_1 \leftarrow r_1 - p_1 + 1, \quad n_2 \leftarrow r_2 - p_2 + 1$
2. *if* $n_1 < n_2$ *then*
3. $p_1 \leftrightarrow p_2, \quad r_1 \leftrightarrow r_2, \quad n_1 \leftrightarrow n_2$
4. *if* $n_1 = 0$ *then return*
5. *else*
6. $q_1 \leftarrow \lfloor (p_1 + r_1) / 2 \rfloor$
7. $q_2 \leftarrow \text{Binary-Search} (T[q_1], T, p_2, r_2)$
8. $q_3 \leftarrow p_3 + (q_1 - p_1) + (q_2 - p_2)$
9. $A[q_3] \leftarrow T[q_1]$
10. *spawn* *Par-Merge* ($T, p_1, q_1-1, p_2, q_2-1, A, p_3$)
11. *Par-Merge* ($T, q_1+1, r_1, q_2+1, r_2, A, q_3+1$)
12. *sync*

Span:

$$T_{\infty}(n) = \begin{cases} \Theta(1), & \text{if } n = 1, \\ T_{\infty}\left(\frac{3n}{4}\right) + \Theta(\log n), & \text{otherwise.} \end{cases}$$
$$= \Theta(\log^2 n) \quad [\text{MT Case 2}]$$

Work:

Clearly, $T_1(n) = \Omega(n)$

We show below that, $T_1(n) = O(n)$

For some $\alpha \in \left[\frac{1}{4}, \frac{3}{4}\right]$, we have the following recurrence,

$$T_1(n) = T_1(\alpha n) + T_1((1 - \alpha)n) + O(\log n)$$

Assuming $T_1(n) \leq c_1 n - c_2 \log n$ for positive constants c_1 and c_2 , and substituting on the right hand side of the above recurrence gives us: $T_1(n) \leq c_1 n - c_2 \log n = O(n)$.

Hence, $T_1(n) = \Theta(n)$.

Parallel Merge Sort with Parallel Merge

Par-Merge-Sort (A, p, r) { sort the elements in $A[p \dots r]$ }

1. *if* $p < r$ *then*
2. $q \leftarrow \lfloor (p+r) / 2 \rfloor$
3. *spawn* *Merge-Sort* (A, p, q)
4. *Merge-Sort* ($A, q+1, r$)
5. *sync*
6. *Par-Merge* (A, p, q, r)

$$\text{Work: } T_1(n) = \begin{cases} \Theta(1), & \text{if } n = 1, \\ 2T_1\left(\frac{n}{2}\right) + \Theta(n), & \text{otherwise.} \end{cases}$$

$$= \Theta(n \log n) \quad [\text{MT Case 2}]$$

$$\text{Span: } T_\infty(n) = \begin{cases} \Theta(1), & \text{if } n = 1, \\ T_\infty\left(\frac{n}{2}\right) + \Theta(\log^2 n), & \text{otherwise.} \end{cases}$$

$$= \Theta(\log^3 n) \quad [\text{MT Case 2}]$$

$$\text{Parallelism: } \frac{T_1(n)}{T_\infty(n)} = \Theta\left(\frac{n}{\log^2 n}\right)$$