Divide & Conquer - 01

CS3026 – Analysis & Design of Algorithms

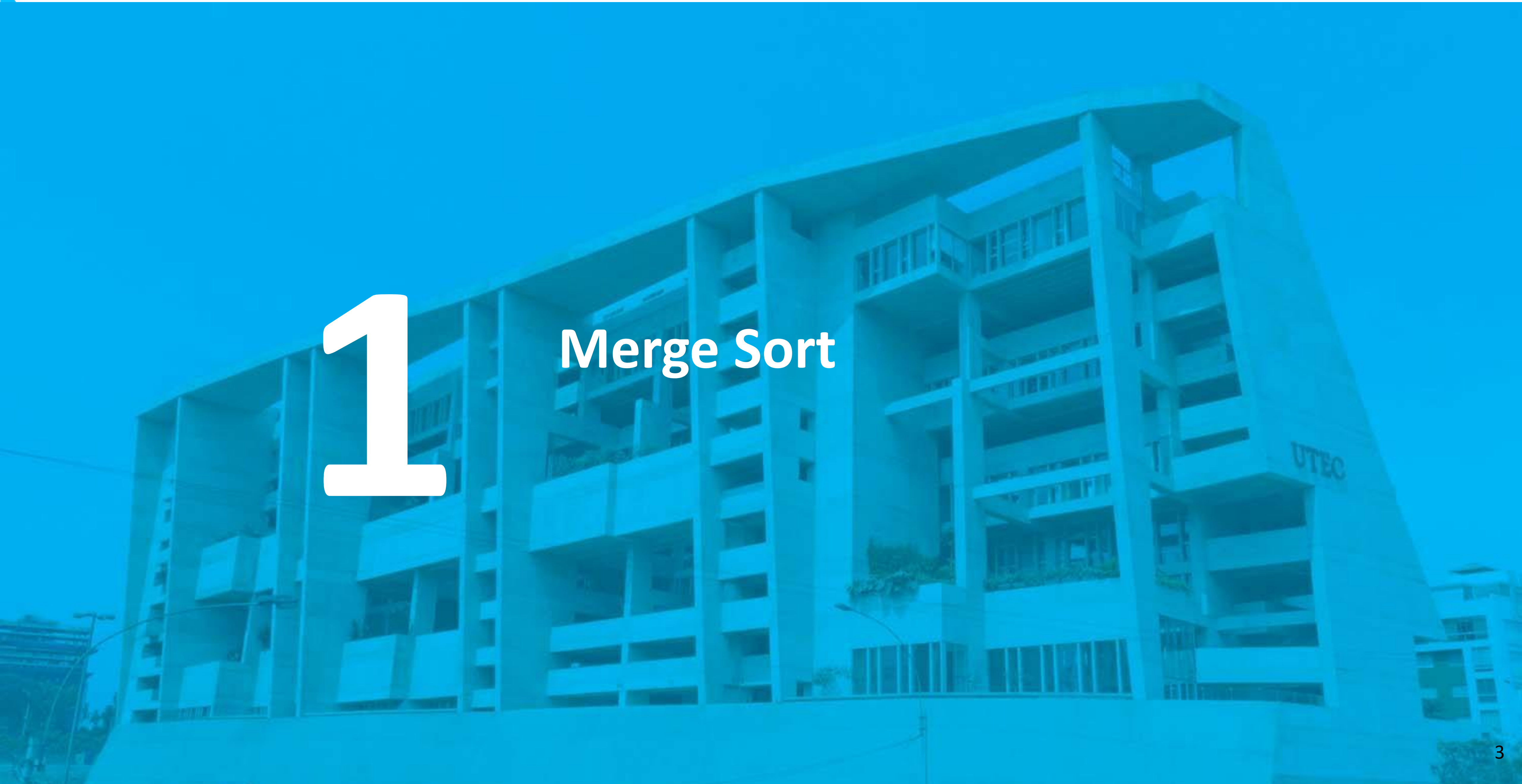
Angel Napa

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Índice

1. Merge Sort
2. Divide & Conquer Method
3. Running Time Analysis
4. Recurrences
5. Master Theorem

1 Merge Sort



Merge Sort

```
MERGE-SORT(A, p, r)
  if p ≥ r                              // zero or one element?
      return
  q = ⌊(p + r)/2⌋                        // midpoint of A[p : r]
  MERGE-SORT(A, p, q)                   // recursively sort A[p : q]
  MERGE-SORT(A, q + 1, r)               // recursively sort A[q + 1 : r]
  // Merge A[p : q] and A[q + 1 : r] into A[p : r].
  MERGE(A, p, q, r)
```

Figure 1: Cormen, Introduction to Algorithms

Merge Sort

```
MERGE(A, p, q, r)
nL = q - p + 1            // length of A[p:q]
nR = r - q                // length of A[q+1:r]
let L[0:nL - 1] and R[0:nR - 1] be new arrays
for i = 0 to nL - 1       // copy A[p:q] into L[0:nL - 1]
    L[i] = A[p + i]
for j = 0 to nR - 1       // copy A[q+1:r] into R[0:nR - 1]
    R[j] = A[q + j + 1]
i = 0                     // i indexes the smallest remaining element in L
j = 0                     // j indexes the smallest remaining element in R
k = p                     // k indexes the location in A to fill
// As long as each of the arrays L and R contains an unmerged element,
//      copy the smallest unmerged element back into A[p:r].
while i < nL and j < nR
    if L[i] <= R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
    k = k + 1
// Having gone through one of L and R entirely, copy the
//      remainder of the other to the end of A[p:r].
while i < nL
    A[k] = L[i]
    i = i + 1
    k = k + 1
while j < nR
    A[k] = R[j]
    j = j + 1
    k = k + 1
```

Figure 2, 3: Cormen, Introduction to Algorithms

Merge Sort

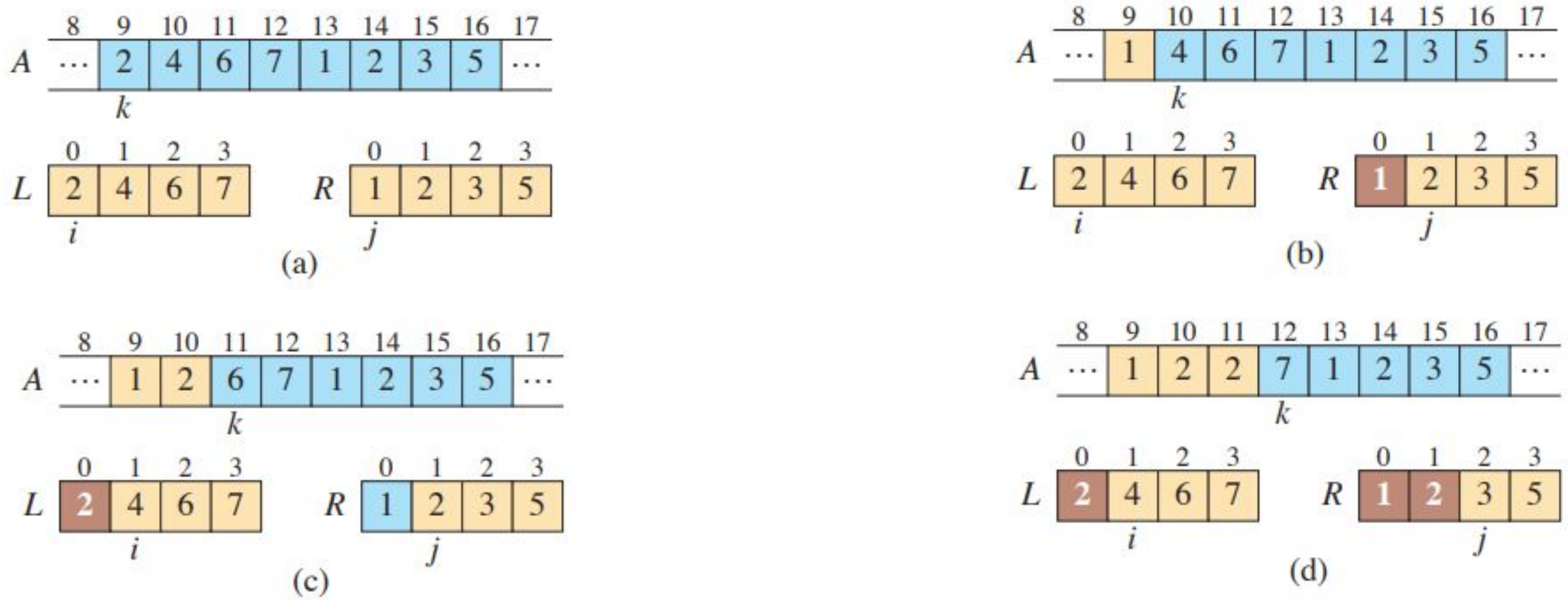


Figure 1: Cormen, Introduction to Algorithms

Merge Sort

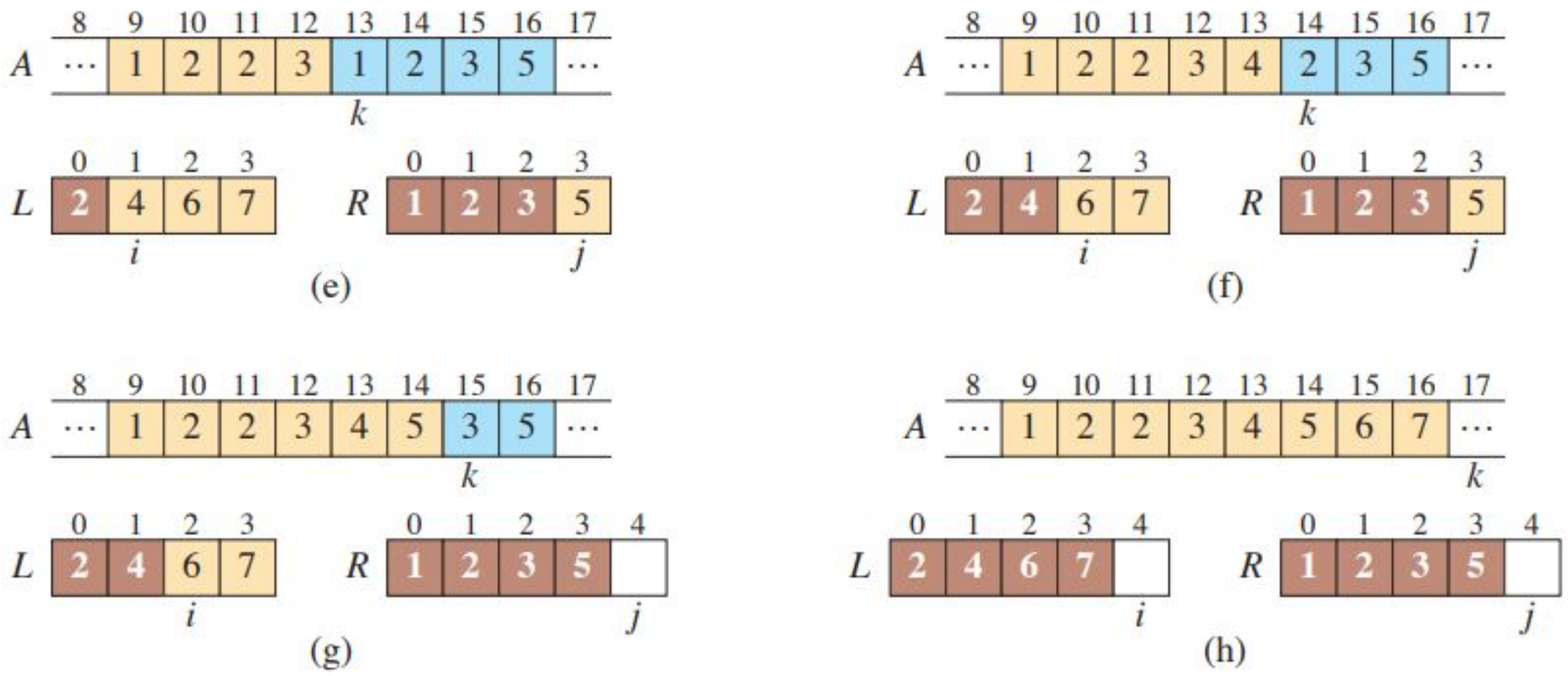


Figure 1: Cormen, Introduction to Algorithms

Merge Sort

Invariant: At the start of each iteration of the first **while** bucle (lines 12–18), the subarray $A[p \ldots k-1]$ contains the $k-p$ smallest elements between $L[0 \ldots n_L - 1]$ and $R[0 \ldots n_R - 1]$, sorted. Also, $L[i]$ y $R[j]$ are the smallest elements of each array that are not in that subarray.

Merge Sort

- **Inicialization** $k = p$, luego $A[p \dots k-1] = \emptyset$
- **Maintenance**

  Case 1: $L[i] \leq R[j]$. Then, we execute line 14. Since $A[p \dots k-1]$ was sorted with the smallest elements, then $A[p \dots k]$ will have the $k-p+1$ smallest elements. Case 2: $L[i] > R[j]$: simmilar.

- **Termination**

  Let $k = pos + 1$. Then $A[p \dots k-1] = A[p \dots pos]$ contains the $k - p = pos - p + 1$ smallest elements of $L[0 \dots n_L - 1]$ y $R[0 \dots n_R - 1]$. If $pos = r$,

2 Divide & Conquer:

Divide & Conquer Method

- **Divide** the problem in one or more subproblems

- **Conquer**: Solving the subproblems recursively. If the size is small enough, solve it directly

- **Combine** the subproblem solutions to form a solution to the original problem.

Merge Sort

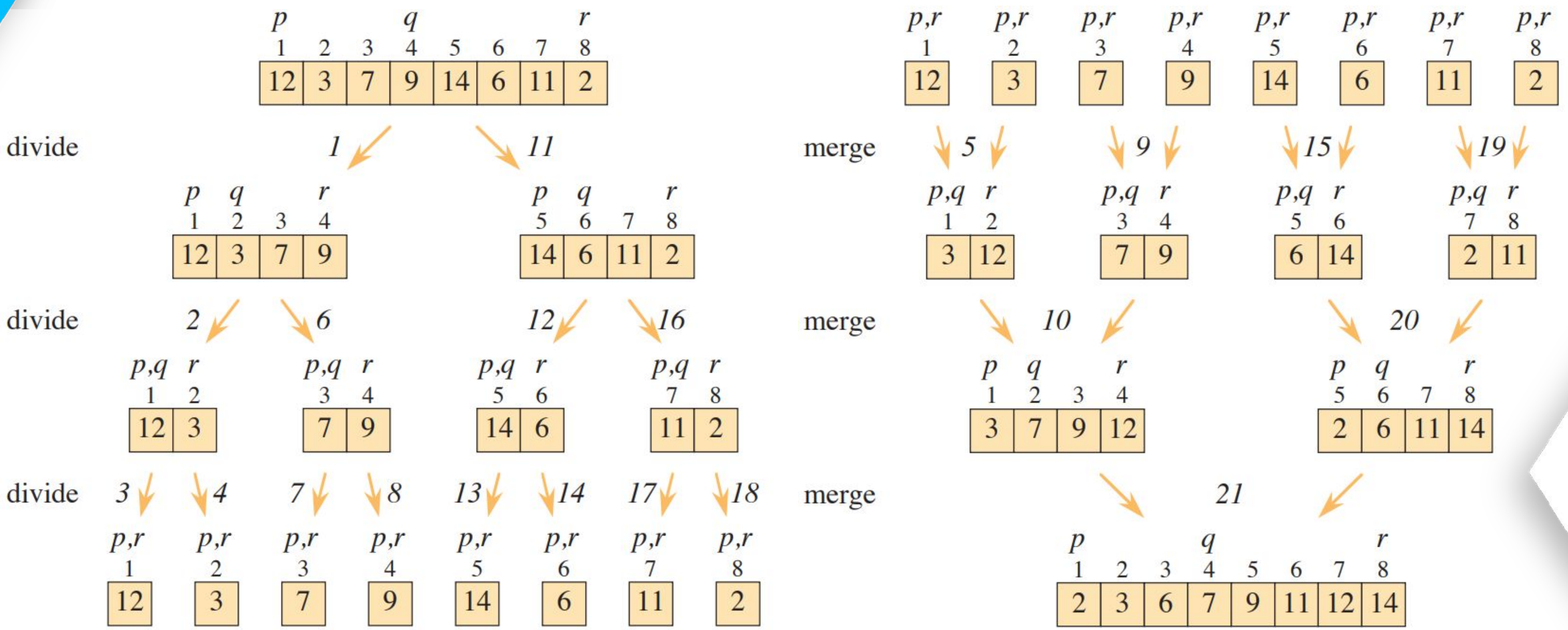


Figure 1: Cormen, Introduction to Algorithms

3 Running Time Analysis

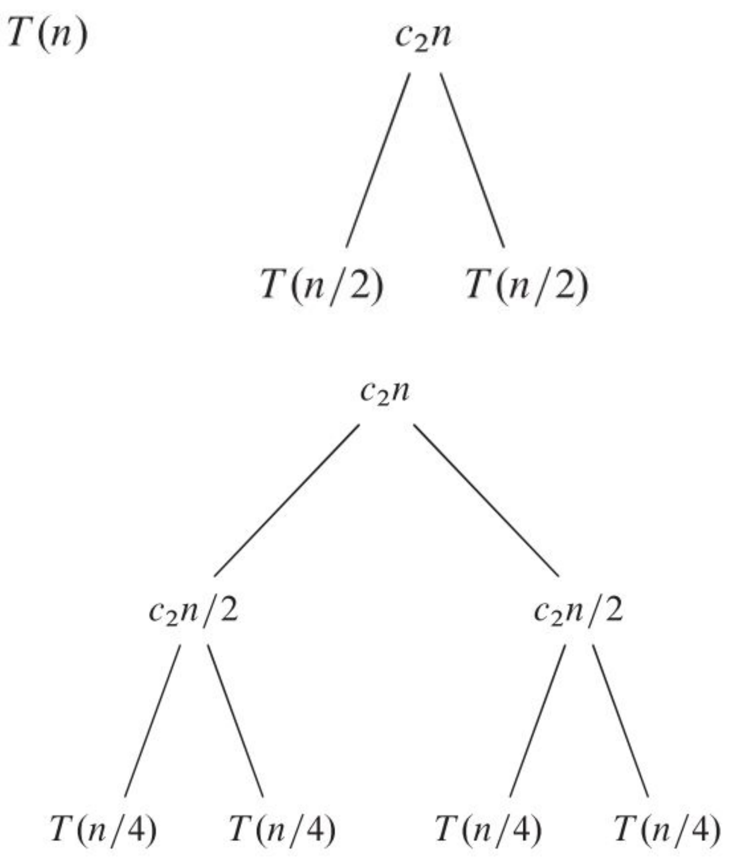
General Formula:

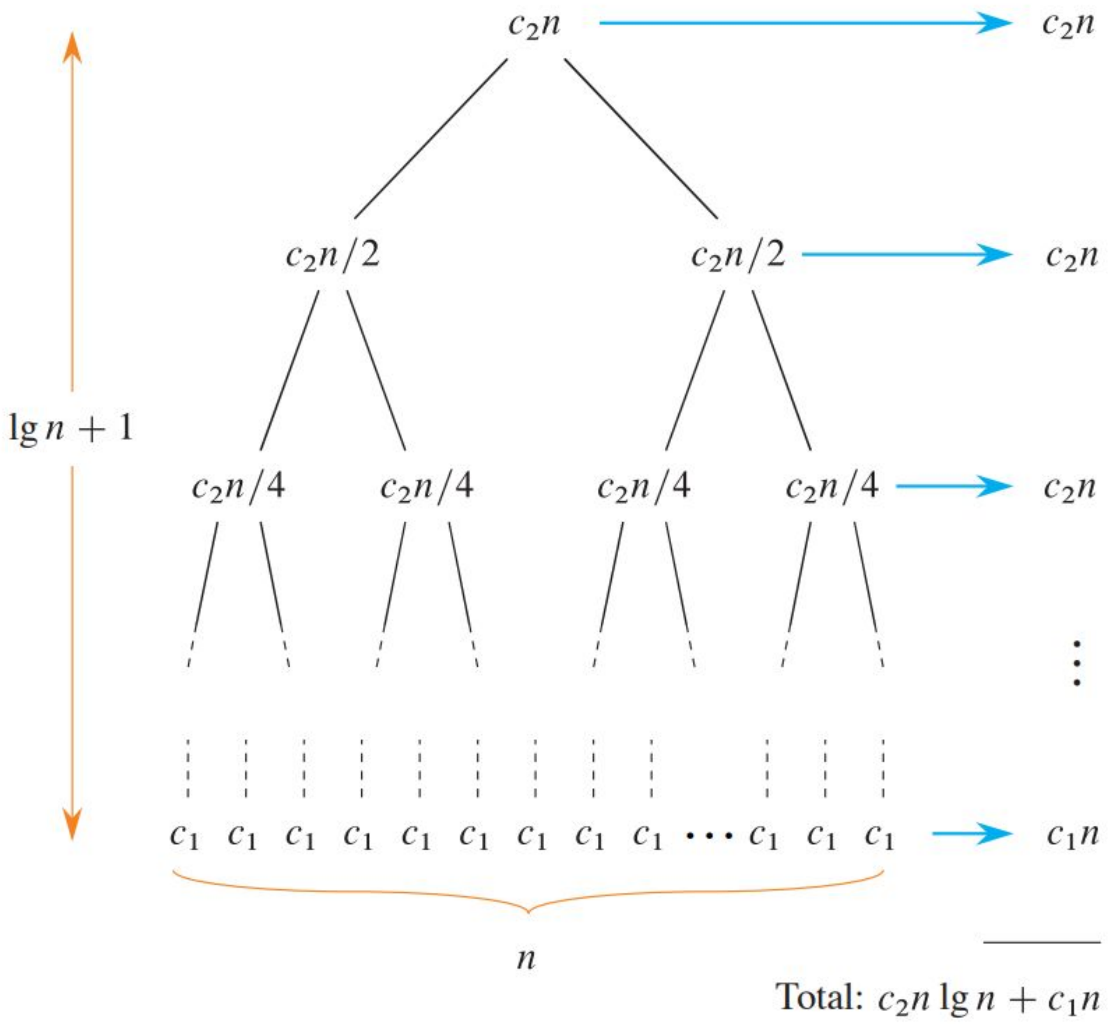
$$T(n) = aT(n/b) + D(n) + C(n),$$

For MERGE-SORT:

$$T(n) = 2T(n/2) + k_2 n + k_1 = 2T(n/2) + cn.$$

General Formula:





4 Recurrence

Explicit proof

Example 3.1. *Let* $F : \mathbb{N} \to \mathbb{R}^+$ *defined as*

$$F(n) = \begin{cases} 1 & n = 1 \\ 2F(n-1) + 1 & otherwise \end{cases}$$

For example: $F(2) = 2F(1)+1 = 3, F(3) = 2F(2)+1 = 7, F(4) = 15$. *What is the value of* $F(n)$*?*

Explicit proof

Example 3.2. *Let* $F : \mathbb{N} \to \mathbb{R}^+$ *define as*

$$F(n) = \begin{cases} 1 & : n = 1 \\ F(n-1) + n & : \textit{otherwise} \end{cases}$$

For example $F(2) = F(1) + 2 = 3, F(3) = F(2) + 3 = 6$. *What is the value of* $F(n)$*?*

Explicit proof

Example 3.3. *Let* $F : \mathbb{N} \to \mathbb{R}^+$ *defined as*

$$F(n) = \begin{cases} 1 & : n = 1 \\ 2F(\lfloor n/2 \rfloor) + n & : \textit{otherwise} \end{cases}$$

For example $F(2) = 2F(1) + 2 = 4, F(3) = 2F(1) + 3 = 5$. *How can we find bounds of* $F(n)$*?*

Proof by induction

Example 3.4. *Let* $T : \mathbb{N} \to \mathbb{R}^+$ *defined as*

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(\lfloor \frac{n}{2} \rfloor) + n & \textit{otherwise} \end{cases}$$

Prove by induction that $T(n) = O(n^2)$. *Prove by induction that* $T(n) = \Omega(n \lg n)$.

Proof by induction

Example 3.5. *Let* $T : \mathbb{N} \to \mathbb{R}^+$ *defined as*

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(\lfloor \frac{n}{2} \rfloor) + 1 & otherwise \end{cases}$$

Prove by induction that $T(n) = O(n)$.

5 Master Theorem

Master Theorem

Let $a \geq 1$, $b \geq 2$, $k \geq 0$, $n_0 \geq 1 \in \mathbb{N}$; $c \in \mathbb{R}^+$. Let $F : \mathbb{N} \to \mathbb{R}^+$ a nondecreasing function such that

$$F(n) = aF(n/b) + cn^k$$

for $n = n_0 b^1, n_0 b^2, n_0 b^3, \ldots$.

It holds that

- If $\lg a / \lg b > k$ then $F(n) = \Theta(n^{\lg a / \lg b})$.
- If $\lg a / \lg b = k$ then $F(n) = \Theta(n^k \lg n)$.
- If $\lg a / \lg b < k$ then $F(n) = \Theta(n^k)$.

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

> Reinventa el mundo <

Thank you!

Let’s keep finding mistakes!

TRANSFORMATEC