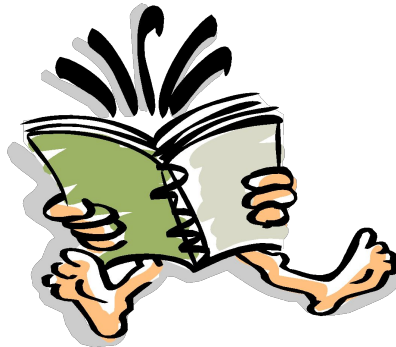


Analysis of Algorithms

CSE245/246



Sorting

- Insertion sort

- Design approach: incremental
- Sorts in place: Yes
- Best case: $\Theta(n)$
- Worst case: $\Theta(n^2)$

- Bubble Sort

- Design approach: incremental
- Sorts in place: Yes
- Running time: $\Theta(n^2)$

Sorting

- Selection sort

- Design approach: incremental
- Sorts in place: Yes
- Running time: $\Theta(n^2)$

- Merge Sort

- Design approach: divide and conquer
- Sorts in place: No
- Running time: *Let's see!!*

Divide-and-Conquer

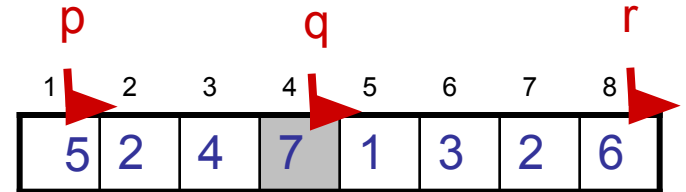
- **Divide** the problem into a number of sub-problems
 - Similar sub-problems of smaller size
- **Conquer** the sub-problems
 - Solve the sub-problems recursively
 - Sub-problem size small enough \Rightarrow solve the problems in straightforward manner
- **Combine** the solutions of the sub-problems
 - Obtain the solution for the original problem

Merge Sort Approach

- To sort an array $A[p \dots r]$:
- **Divide**
 - Divide the n -element sequence to be sorted into two subsequences of $n/2$ elements each
- **Conquer**
 - Sort the subsequences recursively using merge sort
 - When the size of the sequences is 1 there is nothing more to do
- **Combine**
 - Merge the two sorted subsequences

Merge Sort

Alg.: MERGE-SORT(A, p, r)



if $p < r$

Check for base case

then $q \leftarrow \lfloor (p + r)/2 \rfloor$

Divide

MERGE-SORT(A, p, q)

Conquer

MERGE-SORT($A, q + 1, r$)

Conquer

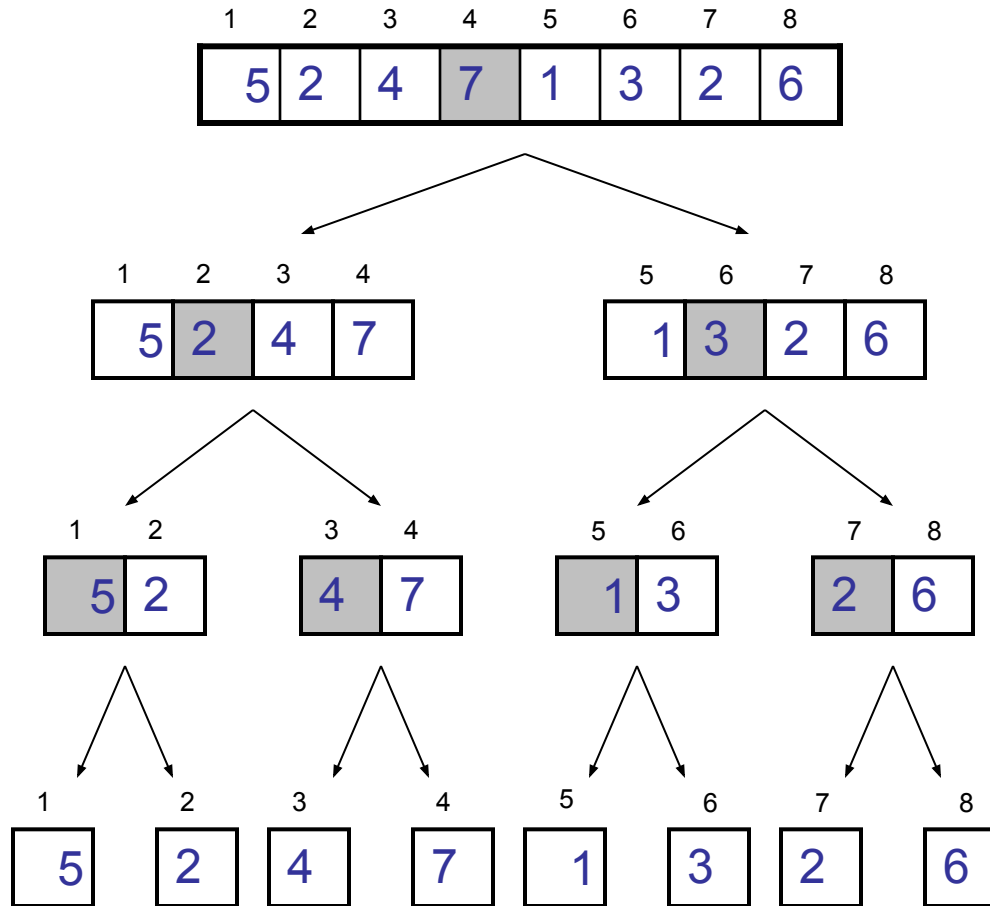
MERGE(A, p, q, r)

Combine

- Initial call: MERGE-SORT($A, 1, n$)

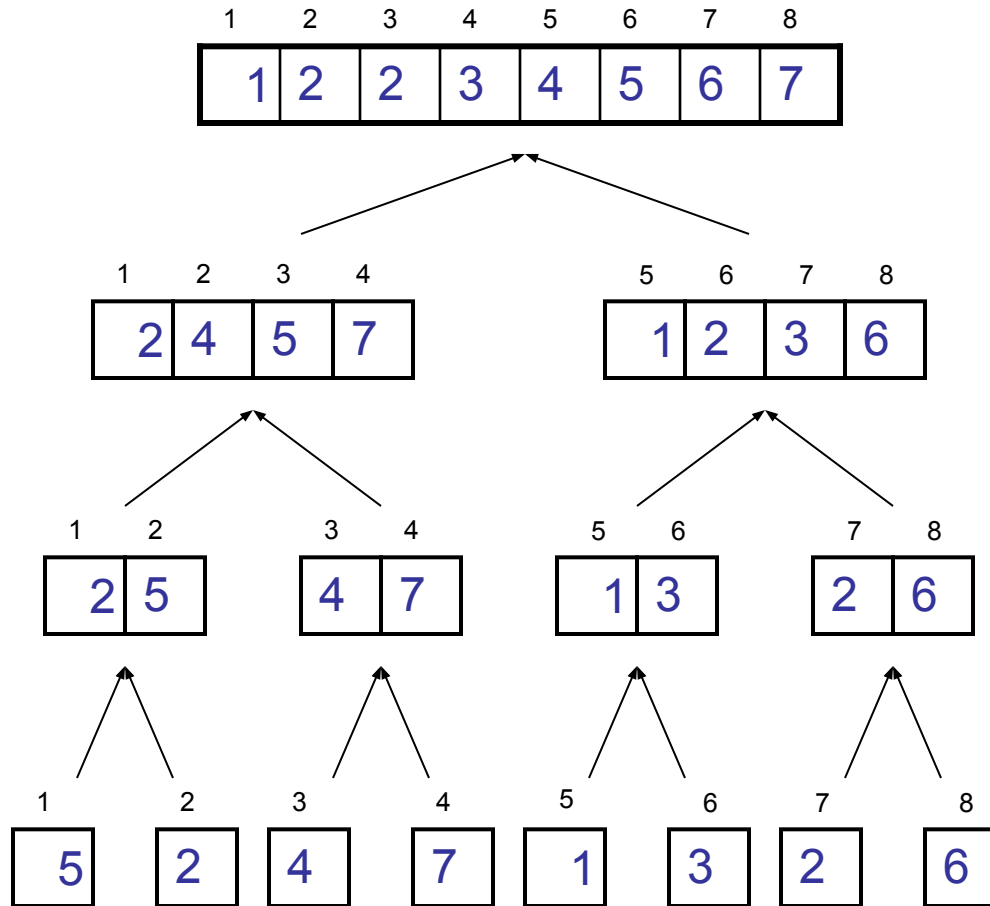
Example – n Power of 2

Divide



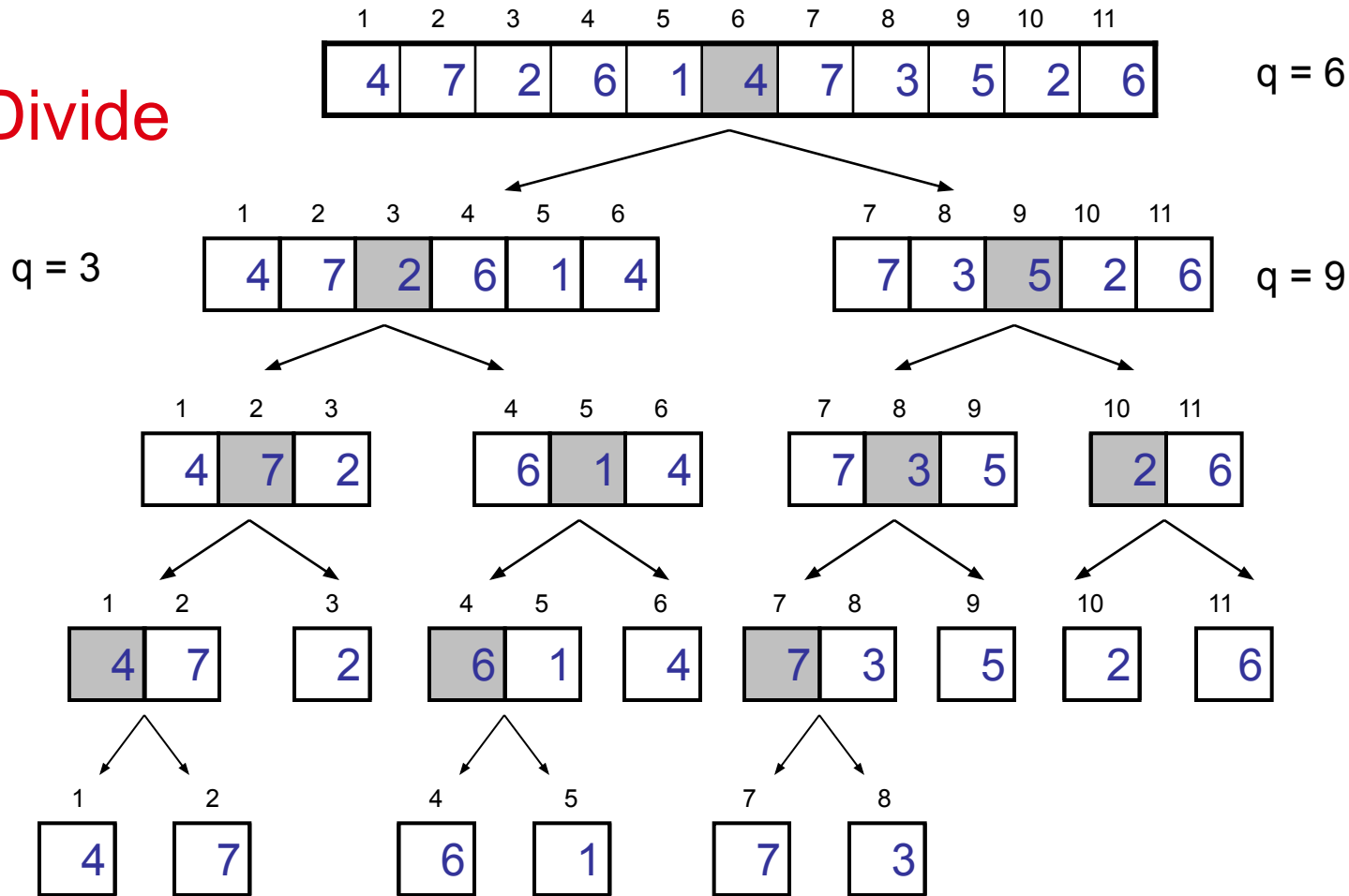
Example – n Power of 2

Conquer
and
Merge



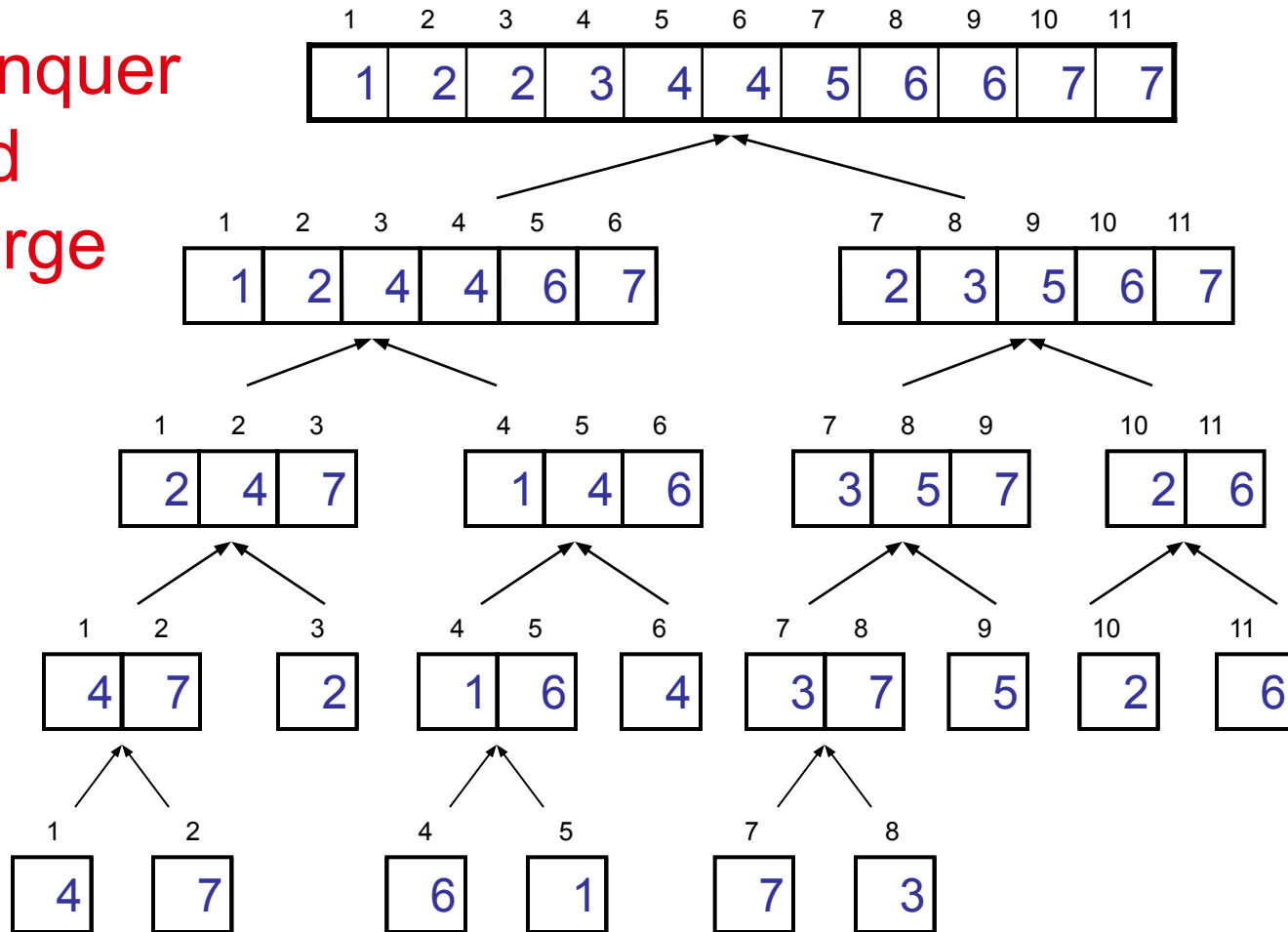
Example – n Not a Power of 2

Divide

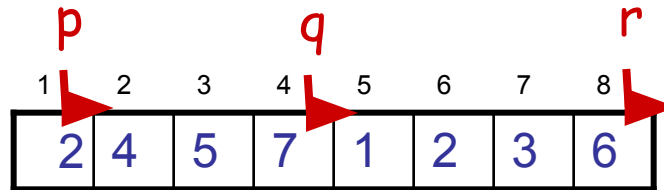


Example – n Not a Power of 2

Conquer
and
Merge



Merging



- **Input:** Array A and indices p, q, r such that $p \leq q < r$
 - Subarrays $A[p \dots q]$ and $A[q + 1 \dots r]$ are sorted
- **Output:** One single sorted subarray $A[p \dots r]$

Merging

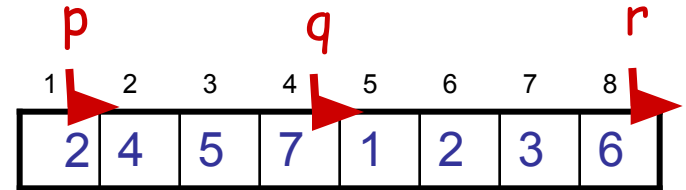
- Idea for merging:

- Two piles of sorted cards

- Choose the smaller of the two top cards
- Remove it and place it in the output pile

- Repeat the process until one pile is empty

- Take the remaining input pile and place it face-down onto the output pile



$A_1 \sqsubset A[p, q]$



$A_2 \sqsubset A[q+1, r]$

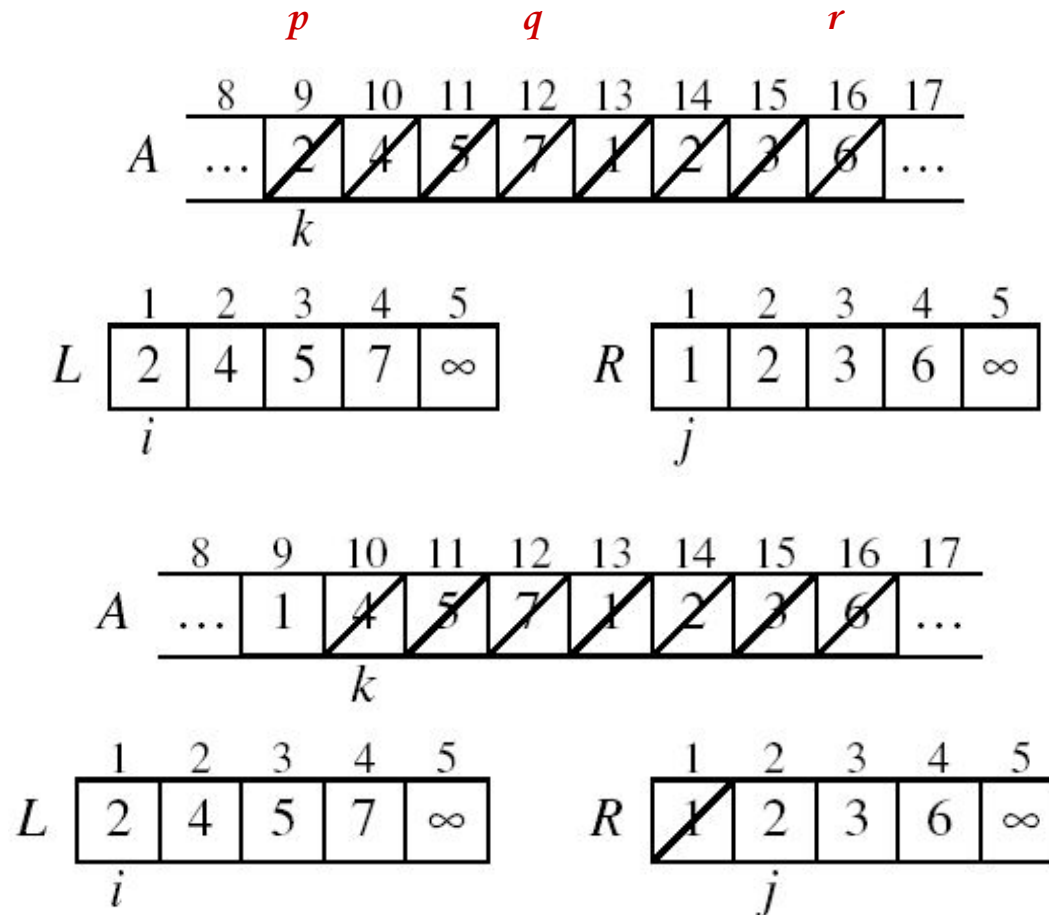


choose the smaller
element from the subarrays

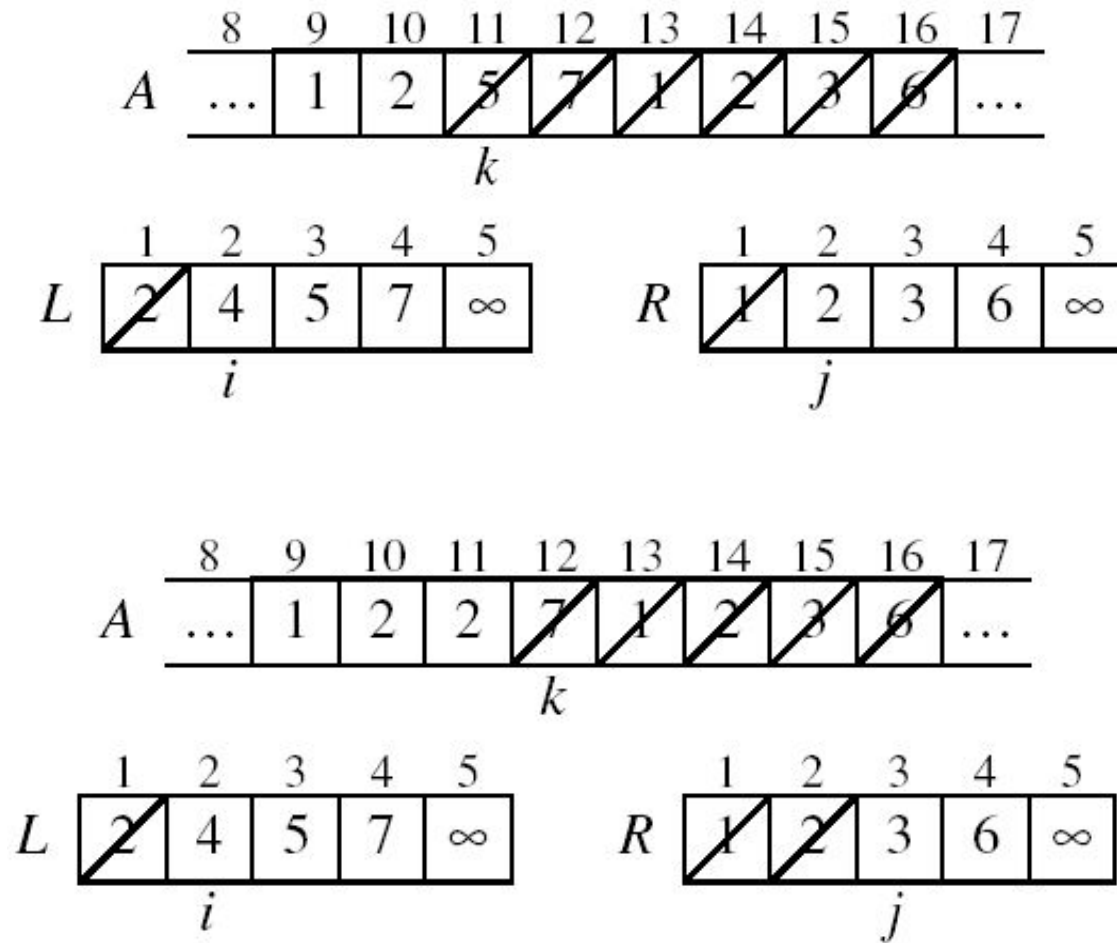
$A[p, r]$



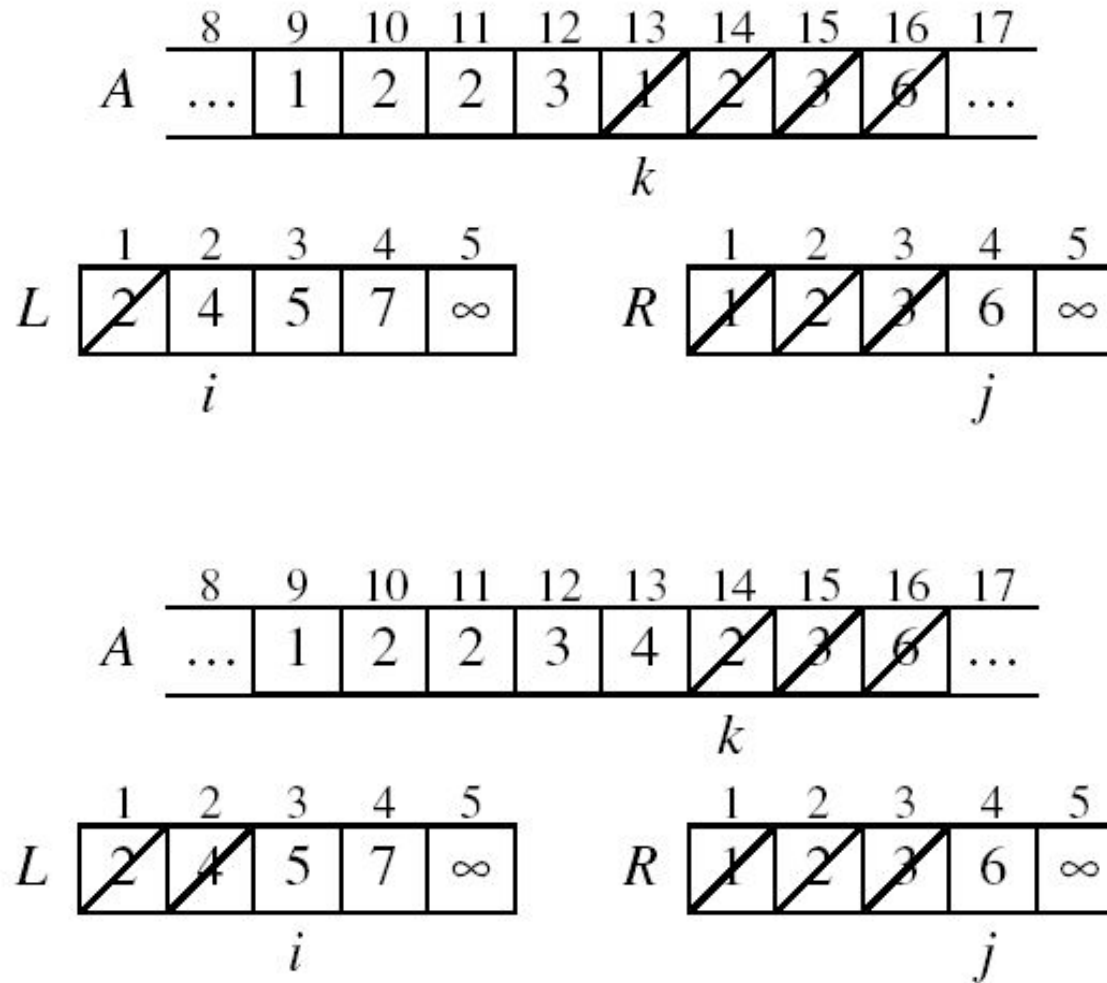
Example: MERGE(A, 9, 12, 16)



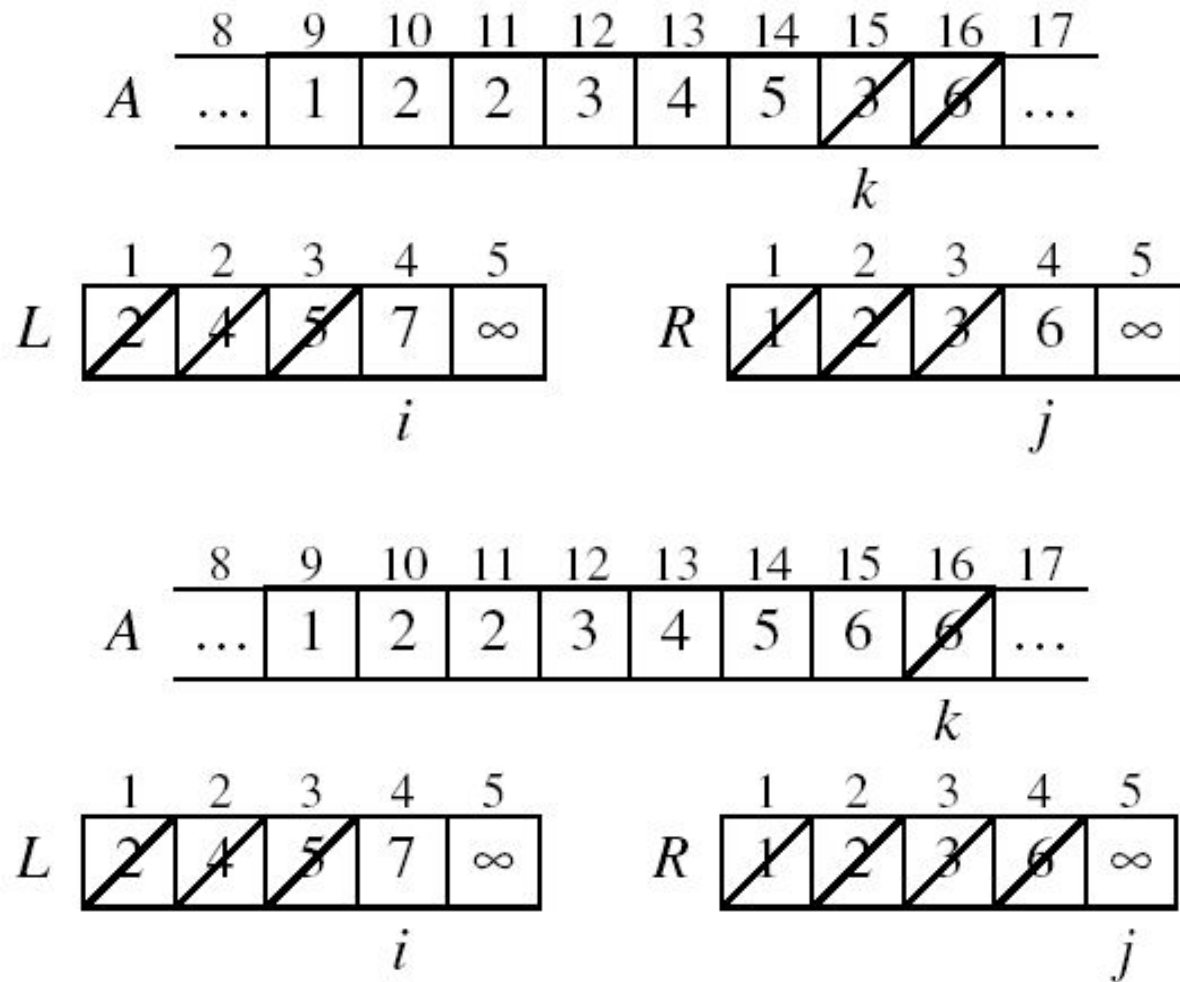
Example: MERGE(A, 9, 12, 16)



Example (cont.)



Example (cont.)



Example (cont.)

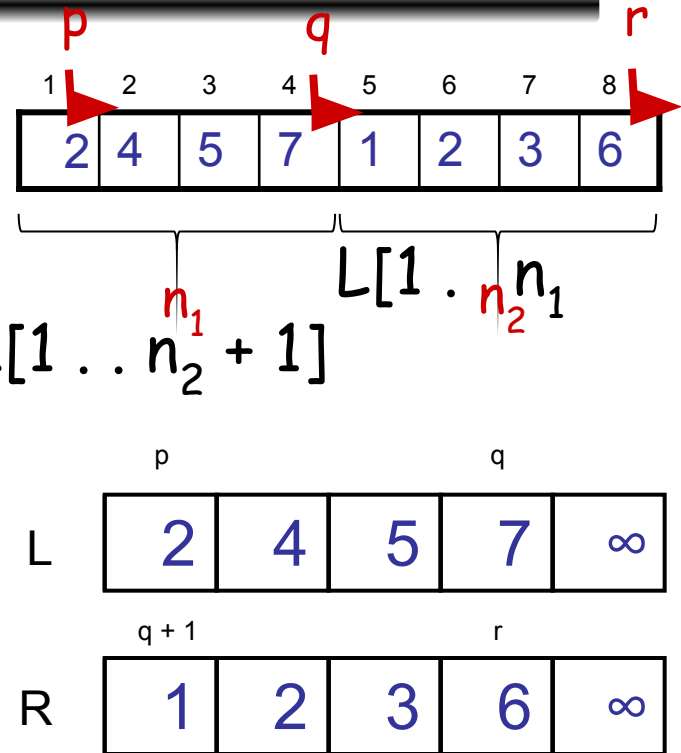
	8	9	10	11	12	13	14	15	16	17	
A	...	1	2	2	3	4	5	6	7	...	
										k	
	1	2	3	4	5						
L	2	4	5	7	∞						
					i						
	1	2	3	4	5						
R	1	2	3	6	∞						
					j						

Done!

Merge - Pseudocode

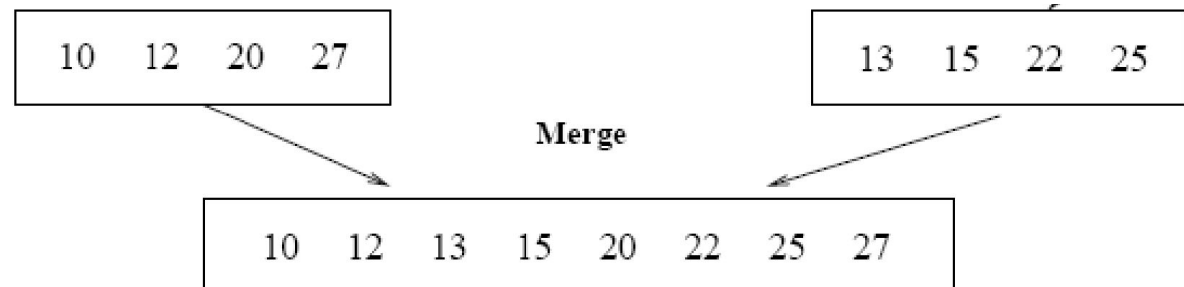
Alg.: MERGE(A, p, q, r)

1. Compute n_1 and n_2
2. Copy the first n_1 elements into $L[1 \dots n_1]$ and the next n_2 elements into $R[1 \dots n_2]$
3. $L[n_1 + 1] \leftarrow \infty$; $R[n_2 + 1] \leftarrow \infty$
4. $i \leftarrow 1$; $j \leftarrow 1$
5. **for** $k \leftarrow p$ **to** r
6. **do if** $L[i] \leq R[j]$
7. **then** $A[k] \leftarrow L[i]$
8. $i \leftarrow i + 1$
9. **else** $A[k] \leftarrow R[j]$
10. $j \leftarrow j + 1$



Running Time of Merge (assume last **for** loop)

- Initialization (copying into temporary arrays):
 - $\Theta(n_1 + n_2) = \Theta(n)$
- Adding the elements to the final array:
 - n iterations, each taking constant time $\Rightarrow \Theta(n)$
- Total time for Merge:
 - $\Theta(n)$



Analyzing Divide-and Conquer Algorithms

- The recurrence is based on the three steps of the paradigm:
 - $T(n)$ – running time on a problem of size n
 - **Divide** the problem into a subproblems, each of size n/b : takes $D(n)$
 - **Conquer** (solve) the subproblems $aT(n/b)$
 - **Combine** the solutions $C(n)$

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ aT(n/b) + D(n) + C(n) & \text{otherwise} \end{cases}$$

MERGE-SORT Running Time

- **Divide:**

- compute q as the average of p and r : $D(n) = \Theta(1)$

- **Conquer:**

- recursively solve 2 subproblems, each of size $n/2$
 $\Rightarrow 2T(n/2)$

- **Combine:**

- MERGE on an n -element subarray takes $\Theta(n)$ time
 $\Rightarrow C(n) = \Theta(n)$

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

Solve the Recurrence

$$T(n) = \begin{cases} c & \text{if } n = 1 \\ 2T(n/2) + cn & \text{if } n > 1 \end{cases}$$

Use Master's Theorem:

Compare n with $f(n) = cn$

Case 2: $T(n) = \Theta(n \lg n)$