Sorting I: Mergesort

Instructor: Meng-Fen Chiang

COMPCSI220: WEEK 9

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND

Slides adapted from Mark Wilson, Georgy Gimel’farb, Simone Linz, Tanya Gvozdeva, and Kaiqi Zhao

Mergesort: Worst-case Running time of $\Theta(n \log n)$

A recursive divide-and-conquer approach to data sorting introduced by Professor John von Neumann in 1945!

- The best, worst, and average cases are similar.
- Particularly good for sorting data with slow access times, e.g., stored in external memory or linked lists.

- Basic ideas behind the algorithm
  - If the number of items is 1, return; otherwise:
    1. Separate the list into two lists of equal or nearly equal size.
    2. Recursively sort the first and the second halves separately.
  - Finally, merge the two sorted halves into one sorted list.

Almost all the work is performed in the merge steps.

Mergesort: Merge

Algorithm 1 Merge

function MERGE(list $a[0..n-1]$; indices l, s, r; list $t[0..n-1]$)

sorted sublists $a[l..s-1]$ and $a[s..r]$ into $a[l..r]$

  $i \leftarrow l; j \leftarrow s; k \leftarrow l$

  while $i \leq s-1$ and $j \leq r$ **do**

    if $a[i] \leq a[j]$ **then**

      $t[k] \leftarrow a[i]; k \leftarrow k+1; i \leftarrow i+1$

    else

      $t[k] \leftarrow a[j]; k \leftarrow k+1; j \leftarrow j+1$

  while $i \leq s-1$ **do** ▷ cope the rest of the 1st half

    $t[k] \leftarrow a[i]; k \leftarrow k+1; i \leftarrow i+1$

  while $j \leq r$ **do** ▷ cope the rest of the 2nd half

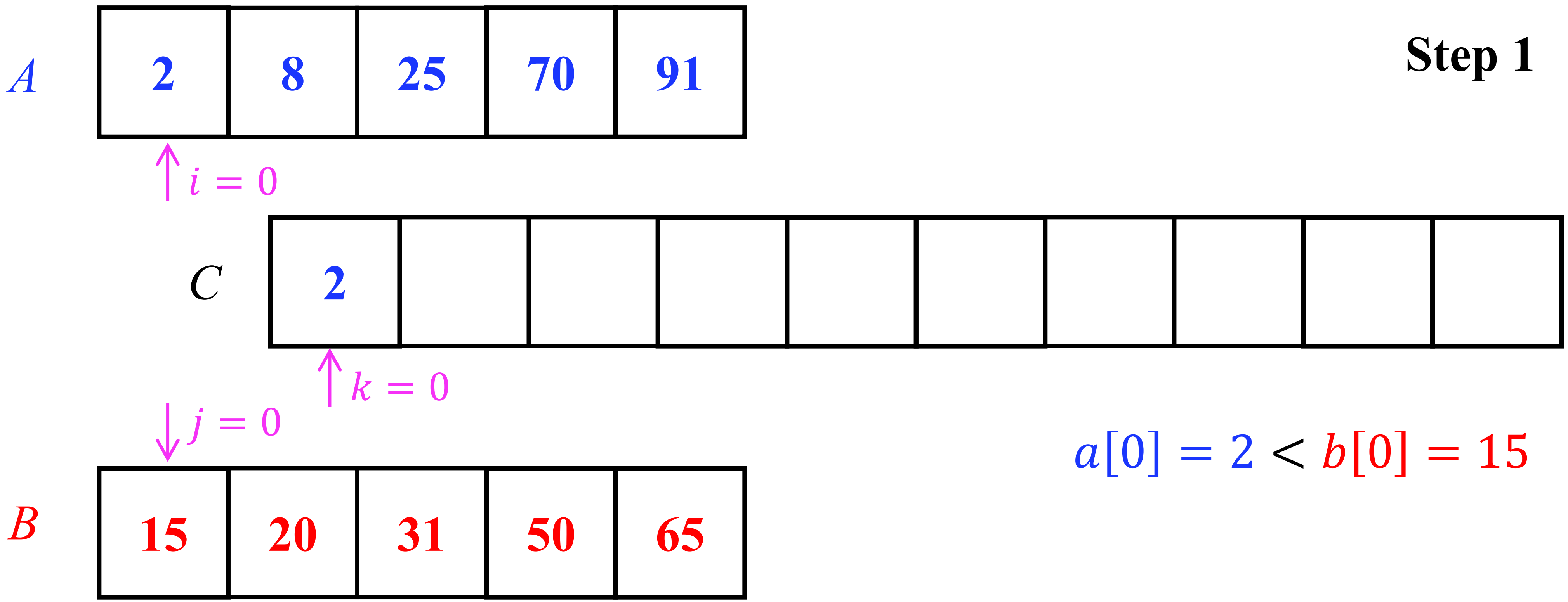
    $t[k] \leftarrow a[j]; k \leftarrow k+1; j \leftarrow j+1$

  $a[l..r] \leftarrow t[l..r]$

Linear Time $\Theta(n)$, Merge of Sorted Arrays

- if $a[i] \le b[j]$ then $c[k{+}{+}] = a[i{+}{+}]$ else $c[k{+}{+}] = b[j{+}{+}]$ (Example 2.10)

Step 1



$a[0] = 2 < b[0] = 15$

if $a[i] \leq b[j]$ then $c[k++] = a[i++]$ else $c[k++] = b[j++]$

Example 2.10

Step 1

A					
	2	**8**	**25**	**70**	**91**

$\uparrow i = 1$

C										
	2									

$\uparrow k = 1$

$\downarrow j = 0$

B					
	15	**20**	**31**	**50**	**65**

$i = 0 + 1; k = 0 + 1$

$$\text{if} \quad a[i] \le b[j] \quad \text{then} \quad c[k++] = a[i++] \quad \text{else} \quad c[k++] = b[j++]$$

Example 2.10

Step 2

A					
	2	**8**	**25**	**70**	**91**

$\uparrow i = 1$

C										
	2	**8**								

$\uparrow k = 1$

$\downarrow j = 0$

B					
	15	**20**	**31**	**50**	**65**

$$a[1] = 8 < b[0] = 15$$

$$\text{if}\ \ a[i] \le b[j]\ \ \text{then}\ \ c[k++] = a[i++]\ \ \text{else}\ \ c[k++] = b[j++]$$

Example 2.10

Step 2

A					
	2	**8**	**25**	**70**	**91**

$\uparrow i = 2$

C										
	2	**8**								

$\uparrow k = 2$

$\downarrow j = 0$

B					
	15	**20**	**31**	**50**	**65**

$i = 1 + 1; k = 1 + 1$

$$\text{if} \quad a[i] \le b[j] \quad \text{then} \quad c[k++] = a[i++] \quad \text{else} \quad c[k++] = b[j++]$$

Example 2.10

Step 3

A

2	**8**	**25**	**70**	**91**

$\uparrow i = 2$

C

2	**8**	**15**							

$\uparrow k = 2$

$\downarrow j = 0$

B

15	**20**	**31**	**50**	**65**

$$a[2] = 25 > b[0] = 15$$

if $a[i] \leq b[j]$ then $c[k++] = a[i++]$ else $c[k++] = b[j++]$

Example 2.10

Step 3

A

2	**8**	**25**	**70**	**91**

$\uparrow i = 2$

C

2	**8**	**15**							

$\uparrow k = 3$

$\downarrow j = 1$

B

15	**20**	**31**	**50**	**65**

$j = 0 + 1; k = 2 + 1$

if $a[i] \leq b[j]$ then $c[k++] = a[i++]$ else $c[k++] = b[j++]$

Example 2.10

Step 4

A

2	**8**	**25**	**70**	**91**

$\uparrow i = 2$

C

2	**8**	**15**	**20**						

$\uparrow k = 3$

$\downarrow j = 1$

B

15	**20**	**31**	**50**	**65**

$a[2] = 25 > b[1] = 20$

$\text{if} \quad a[i] \leq b[j] \quad \text{then} \quad c[k++] = a[i++] \quad \text{else} \quad c[k++] = b[j++]$

Example 2.10

Step 4

A

2	**8**	**25**	**70**	**91**

$\uparrow i = 2$

C

2	**8**	**15**	**20**						

$\uparrow k = 4$

$\downarrow j = 2$

B

15	**20**	**31**	**50**	**65**

$j = 1 + 1; k = 3 + 1$

if $a[i] \leq b[j]$ then $c[k++] = a[i++]$ else $c[k++] = b[j++]$

Example 2.10

Step 5

A

2	**8**	**25**	**70**	**91**

$\uparrow i = 2$

C

2	**8**	**15**	**20**	**25**					

$\uparrow k = 4$

$\downarrow j = 2$

B

15	**20**	**31**	**50**	**65**

$a[2] = 25 < b[2] = 31$

$$\text{if}\quad a[i] \le b[j] \quad \text{then} \quad c[k++] = a[i++] \quad \text{else} \quad c[k++] = b[j++]$$

Example 2.10

Step 5

A

2	**8**	**25**	**70**	**91**

$\uparrow i = 3$

C

2	**8**	**15**	**20**	**25**					

$\uparrow k = 5$

$\downarrow j = 2$

B

15	**20**	**31**	**50**	**65**

$$i = 2 + 1; k = 4 + 1$$

$$\text{if} \quad a[i] \leq b[j] \quad \text{then} \quad c[k++] = a[i++] \quad \text{else} \quad c[k++] = b[j++]$$

Example 2.10

Step 6

A

2	**8**	**25**	**70**	**91**

$\uparrow i = 3$

C

2	**8**	**15**	**20**	**25**	**31**				

$\uparrow k = 5$

$\downarrow j = 2$

B

15	**20**	**31**	**50**	**65**

$$a[3] = 70 > b[2] = 31$$

$$\text{if} \quad a[i] \leq b[j] \quad \text{then} \quad c[k++] = a[i++] \quad \text{else} \quad c[k++] = b[j++]$$

Example 2.10

Step 6

A					
	2	**8**	**25**	**70**	**91**

$\uparrow i = 3$

C										
	2	**8**	**15**	**20**	**25**	**31**				

$\uparrow k = 6$

$\downarrow j = 3$

B					
	15	**20**	**31**	**50**	**65**

$$j = 2 + 1; k = 5 + 1$$

$$\text{if } \ a[i] \le b[j] \ \text{ then } \ c[k++] = a[i++] \ \text{ else } \ c[k++] = b[j++]$$

Example 2.10

Step 7

A

2	**8**	**25**	**70**	**91**

$\uparrow i = 3$

C

2	**8**	**15**	**20**	**25**	**31**	**50**			

$\uparrow k = 6$

$\downarrow j = 3$

B

15	**20**	**31**	**50**	**65**

$$a[3] = 70 > b[3] = 50$$

$$\text{if } \; a[i] \le b[j] \; \text{ then } \; c[k++] = a[i++] \; \text{ else } \; c[k++] = b[j++]$$

Example 2.10

Step 7

A	**2**	**8**	**25**	**70**	**91**

$\uparrow i = 3$

C	**2**	**8**	**15**	**20**	**25**	**31**	**50**			

$\uparrow k = 7$

$\downarrow j = 4$

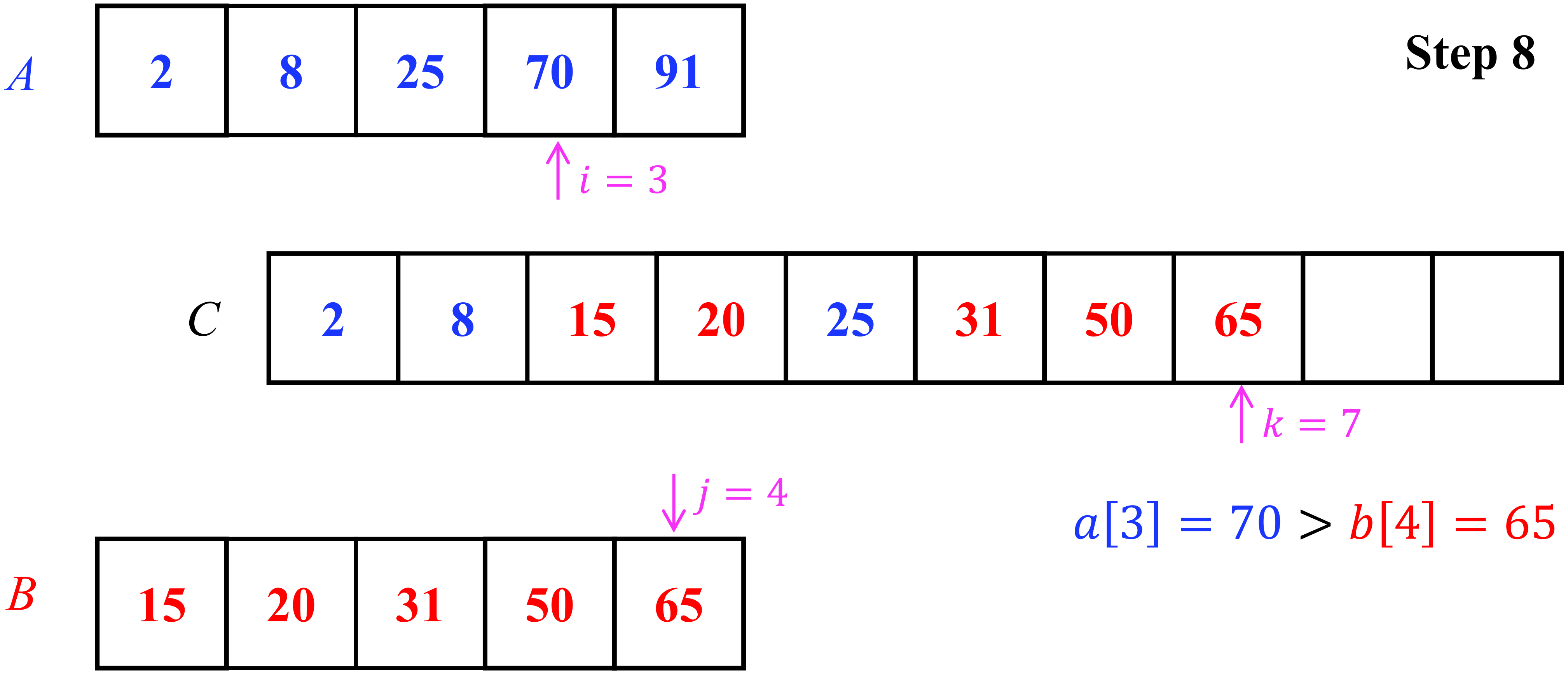
B	**15**	**20**	**31**	**50**	**65**

$j = 3 + 1; k = 6 + 1$

if $a[i] \leq b[j]$ then $c[k++] = a[i++]$ else $c[k++] = b[j++]$

Example 2.10

Step 8



$a[3] = 70 > b[4] = 65$

if $a[i] \leq b[j]$ then $c[k++] = a[i++]$ else $c[k++] = b[j++]$

Example 2.10

Step 8

A					
	2	**8**	**25**	**70**	**91**

$\uparrow i = 3$

C										
	2	**8**	**15**	**20**	**25**	**31**	**50**	**65**		

$\uparrow k = 8$

$\downarrow j = 4$

B					
	15	**20**	**31**	**50**	**65**

B exhausted; $k = 7 + 1$

if $a[i] \le b[j]$ then $c[k++] = a[i++]$ else $c[k++] = b[j++]$

Example 2.10

Step 9, 10

A				
2	**8**	**25**	**70**	**91**

$\uparrow i = 4$

C									
2	**8**	**15**	**20**	**25**	**31**	**50**	**65**	**70**	**91**

$\downarrow j = 4$

B				
15	**20**	**31**	**50**	**65**

A copied; $k = 8$ and 9

Recursive Mergesort for Arrays

- Easier than for linked lists: a constant time for splitting an array in the middle.

Algorithm 2 Mergesort

1: **function** MERGESORT(list $a[0..n-1]$; list indices l, r; list $t[0..n-1]$)
2: **if** $l < r$ **then** sorts the subarray $a[l..r]$
3: $m \leftarrow \lfloor (l+r)/2 \rfloor$
4: MERGESORT(a, l, m, t)
5: MERGESORT$(a, m+1, r, t)$
6: MERGE$(a, l, m+1, r, t)$

- The recursive version simply divides the list until it reaches lists of size 1, then merges these repeatedly.

Time Complexity Analysis

- **Theorem**: The running time of mergesort on an input list of size n is $\Theta(n \log n)$ in the best, worst, and average case.

- **Proof**: The number of comparisons used by mergesort on an input of size n satisfies a recurrence of the form:

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + cn$$

- For simplicity when $n = 2^k$ we have shown that $T(n)$ is $\Theta(n \log n)$.
  - The other elementary operations in the divide and combine steps depend on the implementation of the list, but in each case their number is $\Theta(n)$.
  - Thus these operations satisfy a similar recurrence and do not affect the $\Theta(n \log n)$ answer.

Time Complexity Analysis (Contd.)

- Advantage: The $\Theta(n\ log\ n)$ best-, average-, and worst-case complexity because the merging is always linear.
  - Recall the basic recurrence: $T(n) = 2T\left(\frac{n}{2}\right) + cn \quad \Rightarrow \quad T(n) = cn \lg n$
- Disadvantage:
  - Extra $\Theta(n)$ temporary array for merging data.
  - Extra copying to the temporary array and back.
- Algorithm mergesort is useful only for external sorting.
- For internal sorting: quicksort and heapsort are much better.

SUMMARY

- Mergesort Illustration
- Recursive Mergesort
- Time Complexity Analysis