ECE430.217 Data Structures

Merge Sort

Textbook: Weiss Chapter 7.6

Byoungyoung Lee
https://compsec.snu.ac.kr
byoungyoung@snu.ac.kr

Lecture slides were prepared based on Douglas W. Harder's notes (**dwharder@alumni.uwaterloo.ca).**

Outline

This topic covers merge sort

- A recursive divide-and-conquer algorithm
- Merging two lists
- The merge sort algorithm
- A run-time analysis

Merge Sort

The merge sort algorithm is defined recursively:

- If the list is of size 1, it is sorted—we are done;
- Otherwise:
  - Divide an unsorted list into two sub-lists,
  - Sort each sub-list recursively using merge sort, and
  - Merge the two sorted sub-lists into a single sorted list

This is the first significant **divide-and-conquer/recursive** algorithm

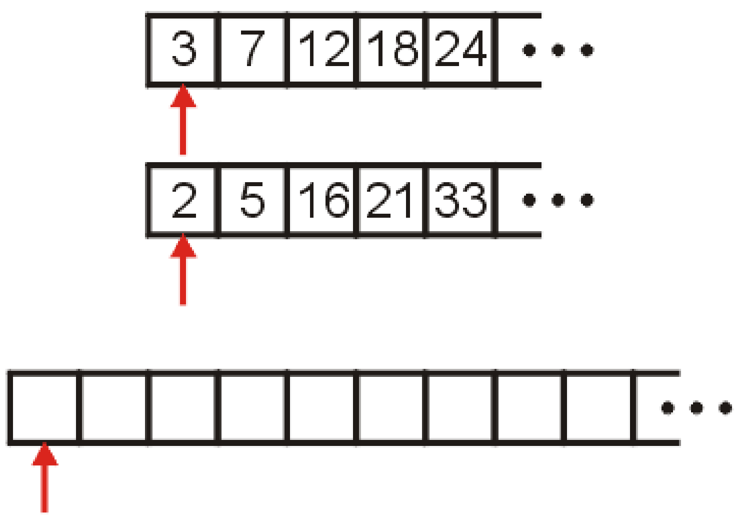
- Other algorithms include: backtracking, dynamic programming, greedy, brute force, randomized, …

Q: How quickly can we recombine the two sub-lists into a single sorted list?

Example: Merging Two Sorted Arrays

Consider the two sorted arrays and an empty array

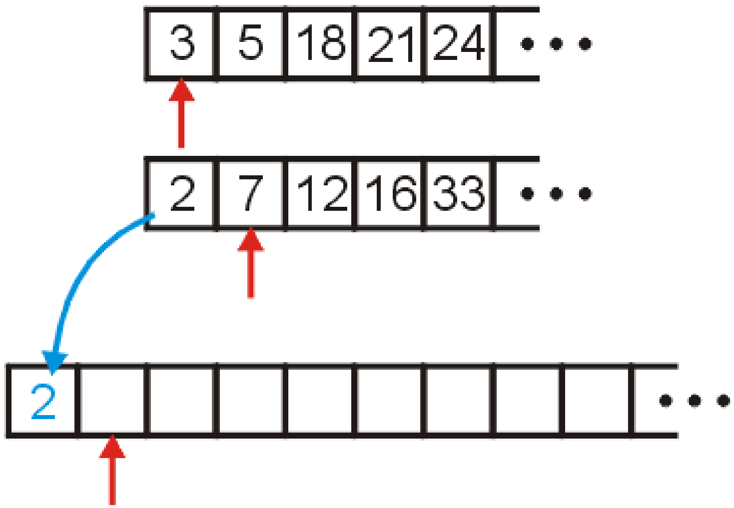
Define three indices, each points to each array's start



Example: Merging Two Sorted Arrays

We compare 2 and 3: $2 < 3$

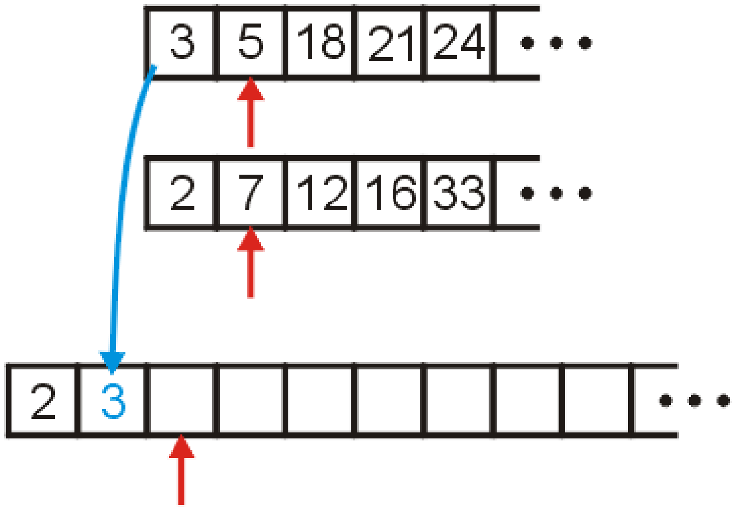
- Copy 2 down
- Increment the corresponding indices



Example: Merging Two Sorted Arrays

We compare 3 and 7

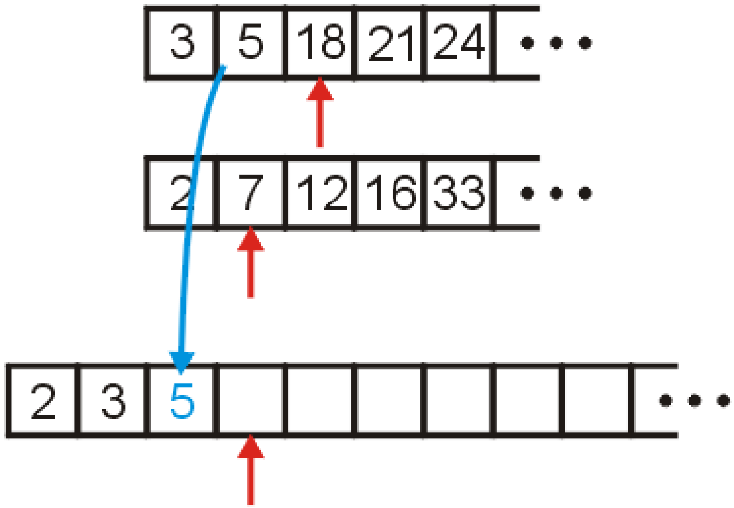
- Copy 3 down
- Increment the corresponding indices



Example: Merging Two Sorted Arrays

We compare 5 and 7

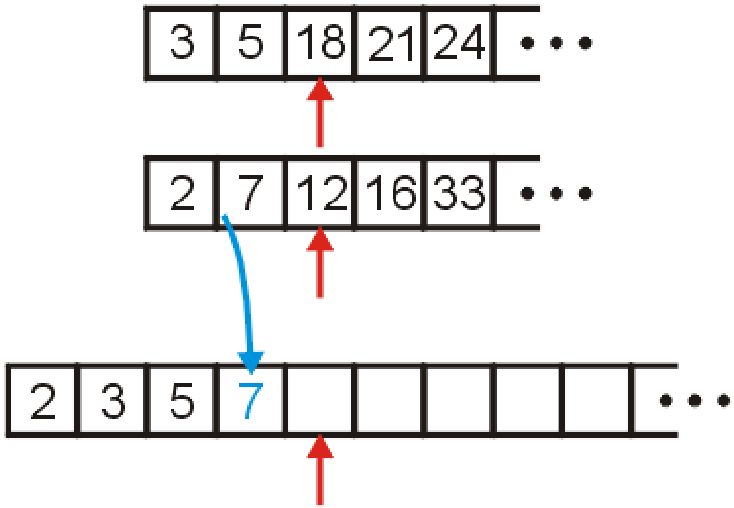
- Copy 5 down
- Increment the appropriate indices



Example: Merging Two Sorted Arrays

We compare 18 and 7

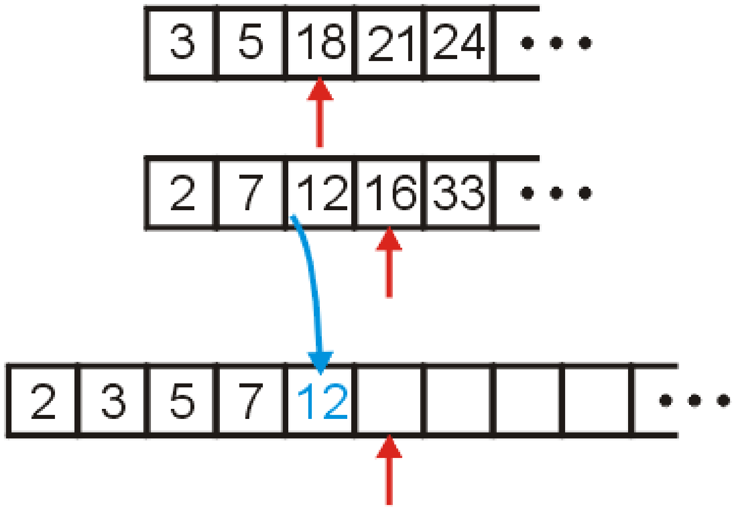
- Copy 7 down
- Increment...



Example: Merging Two Sorted Arrays

We compare 18 and 12

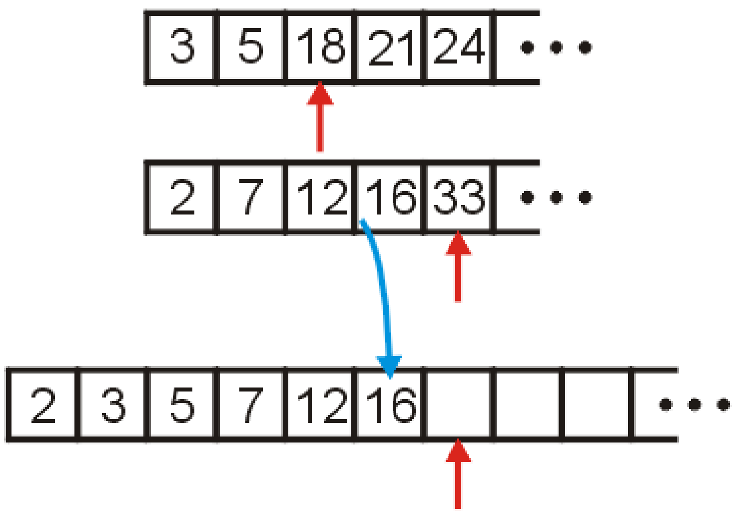
- Copy 12 down
- Increment...



Example: Merging Two Sorted Arrays

We compare 18 and 16

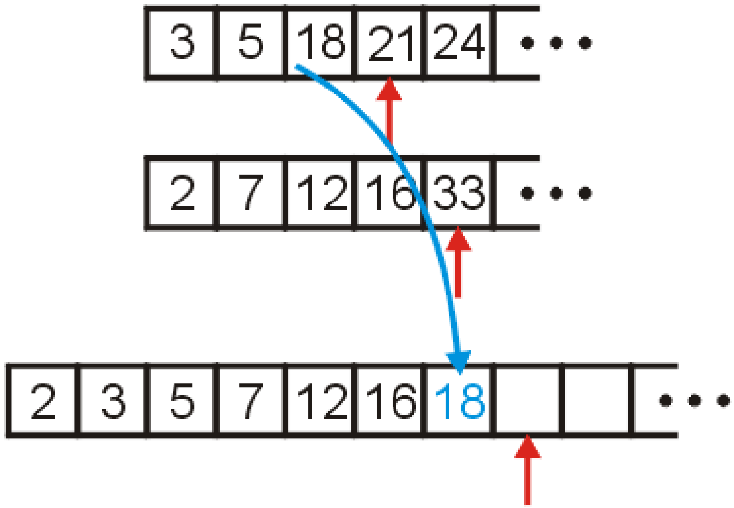
- Copy 16 down
- Increment...



Example: Merging Two Sorted Arrays

We compare 18 and 33

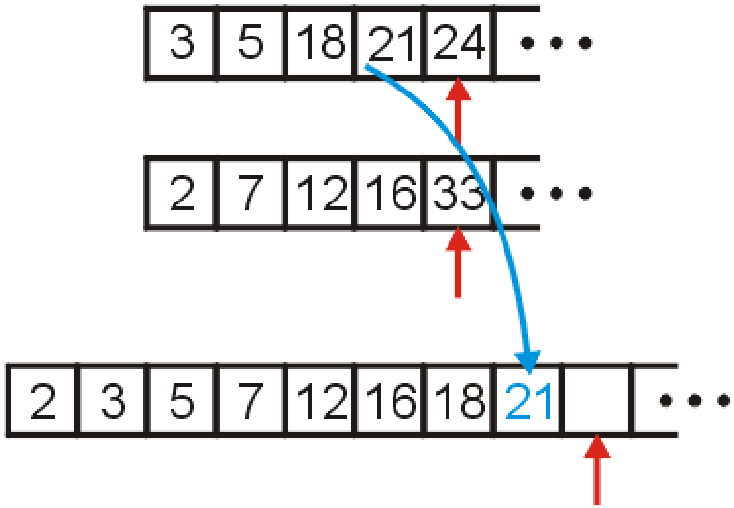
- Copy 18 down
- Increment...



Example: Merging Two Sorted Arrays

We compare 21 and 33

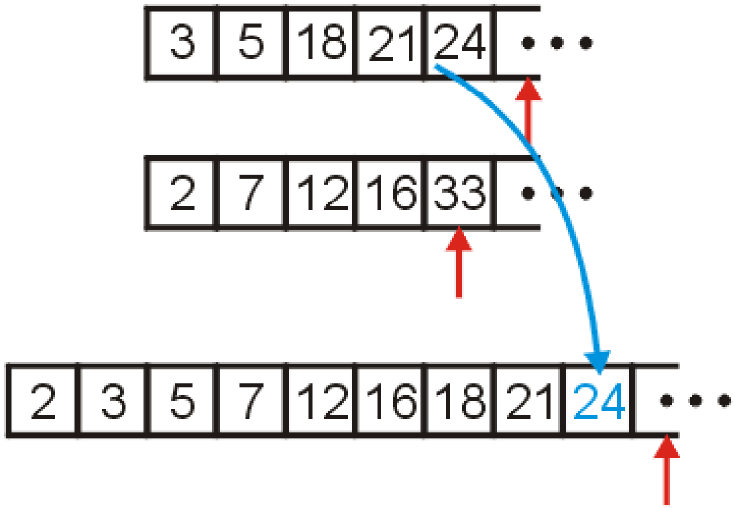
- Copy 21 down
- Increment...



Example: Merging Two Sorted Arrays

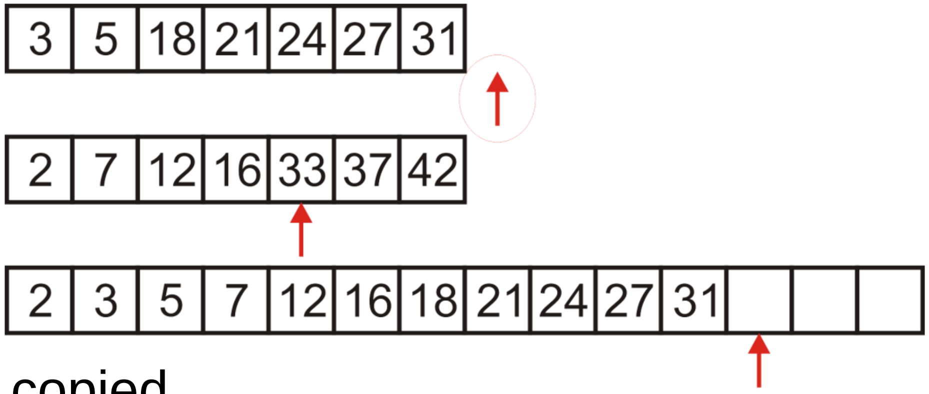
We compare 24 and 33

- Copy 24 down
- Increment...

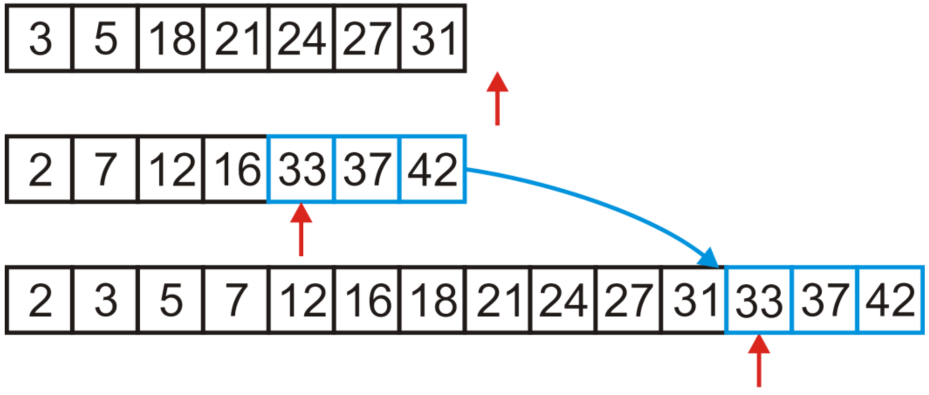


Example: Merging Two Sorted Arrays

We would continue until we have passed beyond the limit of one of the two arrays



After this, the rest can be simply copied



Analysis of merging

Suppose the sorted arrays, `array1` and `array2`, are of size `n1` and `n2`, respectively

- Then merging would be performed in $\Theta(n_1 + n_2)$ time

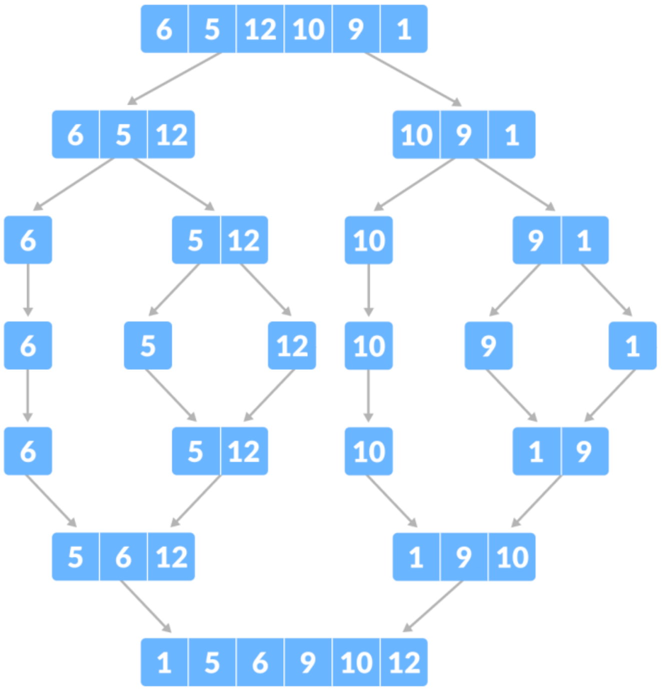
Problem: Merge sorting is out-of-place sorting

- This algorithm always required the allocation of a new array
- Therefore, the memory requirements are also $\Theta(n)$

The Algorithm

The algorithm:

- Split the list into two approximately equal sub-lists
- Recursively call merge sort on both sub lists
- Merge the resulting sorted lists

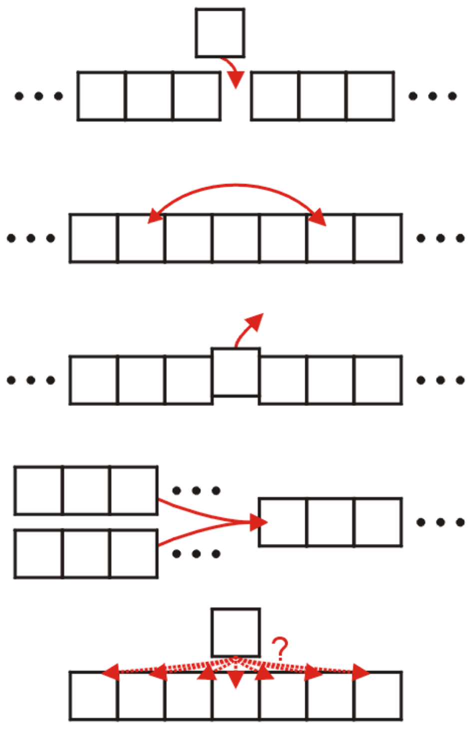


Ref: https://www.programiz.com/dsa/merge-sort

The Algorithm

Recall the five sorting techniques:

- Insertion
- Exchange
- Selection
- Merging
- Distribution



Clearly merge sort falls into the fourth category

Run-time Analysis of Merge Sort

Thus, the time required to sort an array of size $n > 1$ is:

- the time required to sort the first half,
- the time required to sort the second half, and
- the time required to merge the two lists

Representing these with the recurrence relation:

$$\mathrm{T}(n) = \begin{cases} 1 & n = 1 \\ 2\,\mathrm{T}\left(\frac{n}{2}\right) + n & n > 1 \end{cases}$$

Divide by n,

$$\frac{T(n)}{n} = \frac{T(n/2)}{n/2} + 1$$

The equation holds for any n, which is power of 2

$$\frac{T(n/2)}{n/2} = \frac{T(n/4)}{n/4} + 1$$

$$\frac{T(n/4)}{n/4} = \frac{T(n/8)}{n/8} + 1$$

$$\vdots$$

$$\frac{T(2)}{2} = \frac{T(1)}{1} + 1$$

Then telescope a sum of all equations,

$$\frac{T(n)}{n} = \frac{T(1)}{1} + \log n$$

Finally we get

$$T(n) = n \log n + n = O(n \log n)$$

Run-time Summary

The following table summarizes the run-times of merge sort

Case	Run Time	Comments
Worst	$\mathbf{O}(n \ln(n))$	No worst case
Average	$\mathbf{O}(n \ln(n))$	
Best	$\mathbf{O}(n \ln(n))$	No best case

Comments

Merge sort requires an additional array

- Heap sort does not require

Next we see quick sort

- Faster, on average, than either heap or quick sort
- Requires $\mathbf{o}(n)$ additional memory

Divide and Conquer and Recursive Algorithm

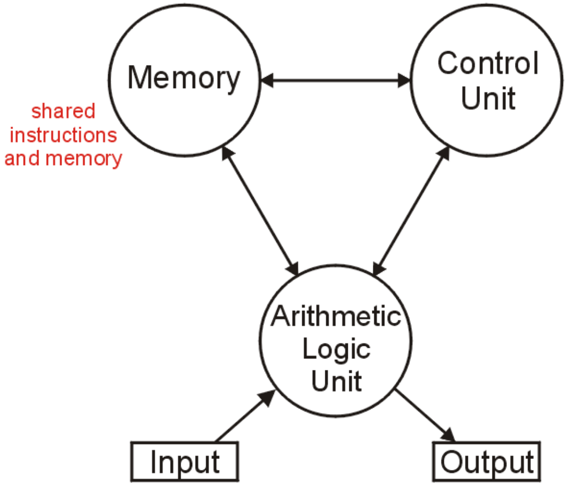
- **Divide and conquer** algorithms have three stages
  - **#1. Divide**
  - **#2. Conquer**
  - **#3. Combine**

- **In the case of merge sort,**
  - **#1. Divide:** Split the array into two sub-arrays
  - **#2. Conquer:** Sort the resulting sub-arrays **recursively** (using the same merge sort)
  - **#3. Combine:** Merge two sorted sub-arrays into a single sorted array

Merge Sort

The (likely) first implementation of merge sort was on the ENIAC in 1945 by John von Neumann

- The creator of the *von Neumann architecture* used by all modern computers:



http://en.wikipedia.org/wiki/Von_Neumann

Summary

This topic covered merge sort:

- Divide an unsorted list into two equal or nearly equal sub lists,
- Sorts each of the sub lists by calling itself recursively, and then
- Merges the two sub lists together to form a sorted list

References

Wikipedia, http://en.wikipedia.org/wiki/Sorting_algorithm
http://en.wikipedia.org/wiki/Sorting_algorithm#Inefficient.2Fhumorous_sorts

[1] Donald E. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching*, 2nd Ed., Addison Wesley, 1998, §5.1, 2, 3.

[2] Cormen, Leiserson, and Rivest, *Introduction to Algorithms*, McGraw Hill, 1990, p.137-9 and §9.1.

[3] Weiss, *Data Structures and Algorithm Analysis in C*++, *3rd Ed.*, Addison Wesley, §7.1, p.261-2.

[4] Gruber, Holzer, and Ruepp, *Sorting the Slow Way: An Analysis of Perversely Awful Randomized Sorting Algorithms*, 4th International Conference on Fun with Algorithms, Castiglioncello, Italy, 2007.