More on Merge Sort and Binary Search

CSCI2100 Tutorial 3

Shangqi Lu

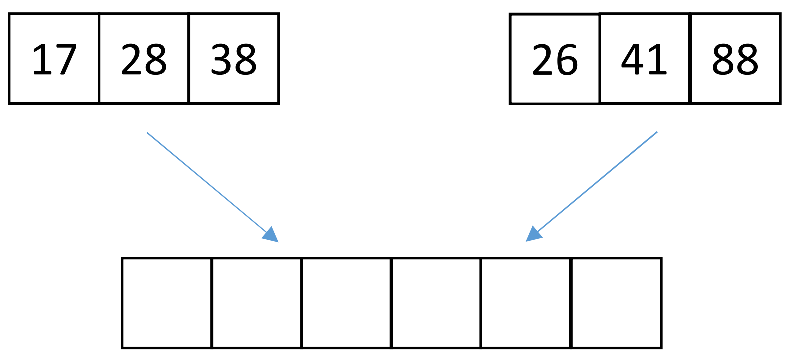
Adapted from the slides of the previous offerings of the course

Outline

- Review merge sort and its variant
- A variant of binary search

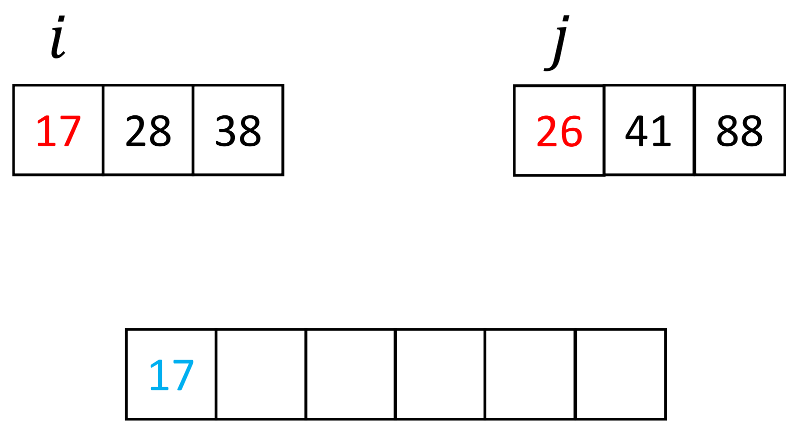
Review - Merge Operation

- Merge 2 sorted arrays into a single sorted array
- For example:



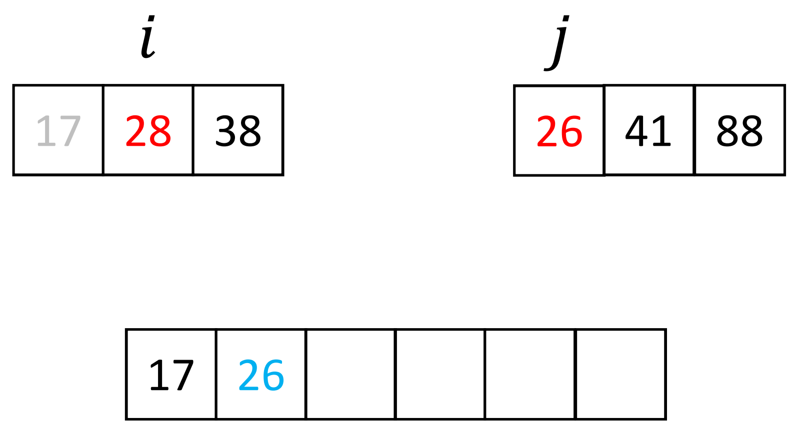
Review - Merge Operation

- Set i, j to be 1
- Compare 17 and 26
- 17 is smaller
- Place 17 into the new array and increase i by 1



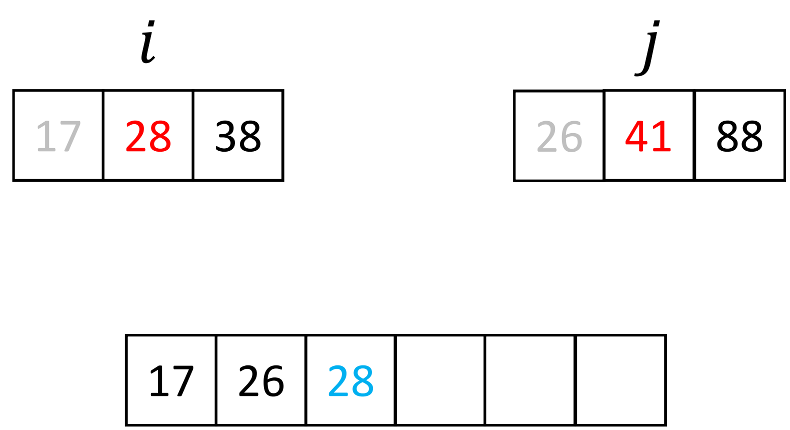
Review - Merge Operation

- Compare 28 and 26
- 26 is smaller
- Place 26 into the new array and increase j by 1



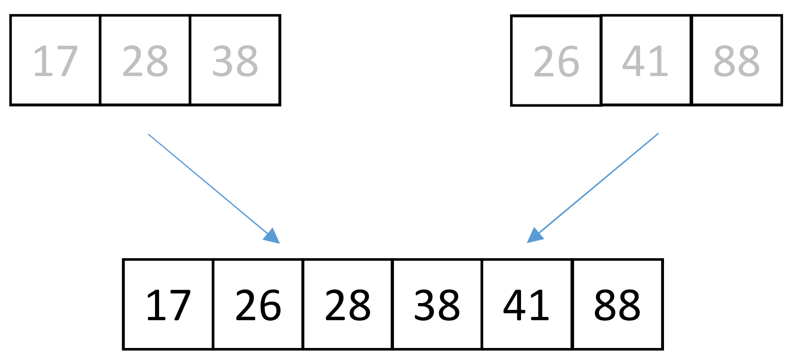
Review - Merge Operation

- Compare 28 and 41
- 28 is smaller
- Place 28 into the new array and increase i by 1

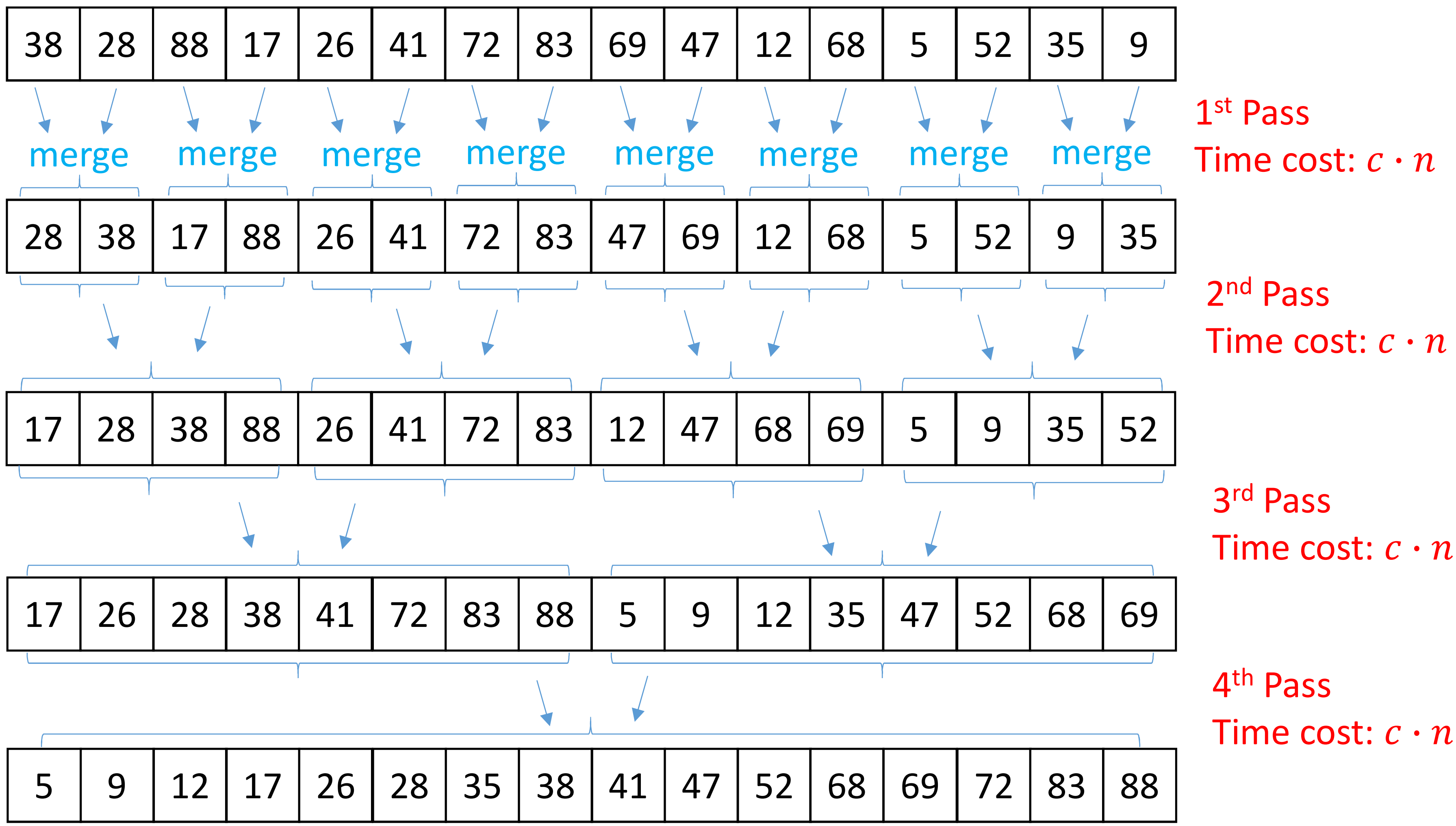


Review - Merge Operation

- Continue the above process until we've placed all elements into the new array
- Single pass over all the input elements
- Time complexity: $O(n)$



Bottom-up Merge Sort



$\log_2 n$ pass in total, cost $c \cdot n$ for each pass, time complexity is $O(n \log n)$

Exercise: Modified Merge Sort

- Regular Exercise 3 Problem 6*
- A variant of merge sort
  - If $n = 1$ then return immediately
  - Otherwise set $k = \lceil n/3 \rceil$
  - Recursively sort $A[1 \ldots k]$ and $A[k + 1 \ldots n]$, respectively
  - Merge $A[1 \ldots k]$ and $A[k + 1 \ldots n]$ into one sorted array
- Prove the time complexity is $O(n \log n)$

Example of Modified Merge Sort

38	28	88	17	26	41	72	83	69	47	12	68	5	52	35	9	

$n/3$

17	26	28	38	41	88	72	83	69	47	12	68	5	52	35	9	

sort recursively

38	28	88	17	26	41	5	9	12	35	47	52	68	69	72	83	

sort recursively

Merge $A[1 \ldots k]$ and $A[k+1 \ldots n]$

5	9	12	17	26	28	35	38	41	47	52	68	69	72	83	88	

Solution

- Let $f(n)$ be the worst case time
- $f(1) = O(1)$
- $f(n) \leq f\left(\left\lceil \frac{n}{3} \right\rceil\right) + f\left(\left\lceil \frac{2n}{3} \right\rceil\right) + O(n)$
- Want to prove $f(n) = O(n \log n)$
- This can be done using the substitution method – see the course website for solution (reg ex list 3).

A Variant of Binary Search

- Instead of comparing the target value with the middle element, we compare the target with the $\left\lceil \frac{n}{3} \right\rceil$th element each time.
  - For example, we want to find the value 13 from the following sorted sequence

2	3	5	8	13	21	34	55					…
		$\uparrow$ $\left\lceil \frac{n}{3} \right\rceil$										

Time Complexity

- In the worst case, after each comparison, two-third of the active elements are left.
- Solution
  - $T(1) = O(1)$
  - $T(n) \leq T\left(\frac{2n}{3}\right) + O(1)$
  - Solving the recurrence gives $T(n) = O(\log n)$.

Time Complexity

- What if we compare the target with the $\left\lceil \frac{n}{300} \right\rceil$ th element?
- The time complexity is also $O(\log n)$!
  - Try verifying this by yourself.

A Bonus Problem: Closest Pair

- Problem Input:
  - Two unsorted sequences A and B with m and n integers
  - n < m
- Goal: Find a pair (x, y), x from A and y from B, with the minimum $|x - y|$.

Sequence A

1	20	9	23	2	20

Sequence B

11	8	7	12	13

A Bonus Problem: Closest Pair

- This problem can be solved in $O(m \log n)$.
  - Sort the shorter sequence.
  - Then use elements of the longer sequence to perform binary searches.
- Note: $O(m \log n)$ is better than $O(m \log m)$ when n << m.

Sequence A	1	20	9	23	2	20

Sequence B	11	8	7	12	13