Week 4 Tutorial

By Yufei Tao's Teaching Team

Outline

- Review recursion principle
- Review merge sort
- A variant of binary search
- A variant of merge sort
- Closest pair problem

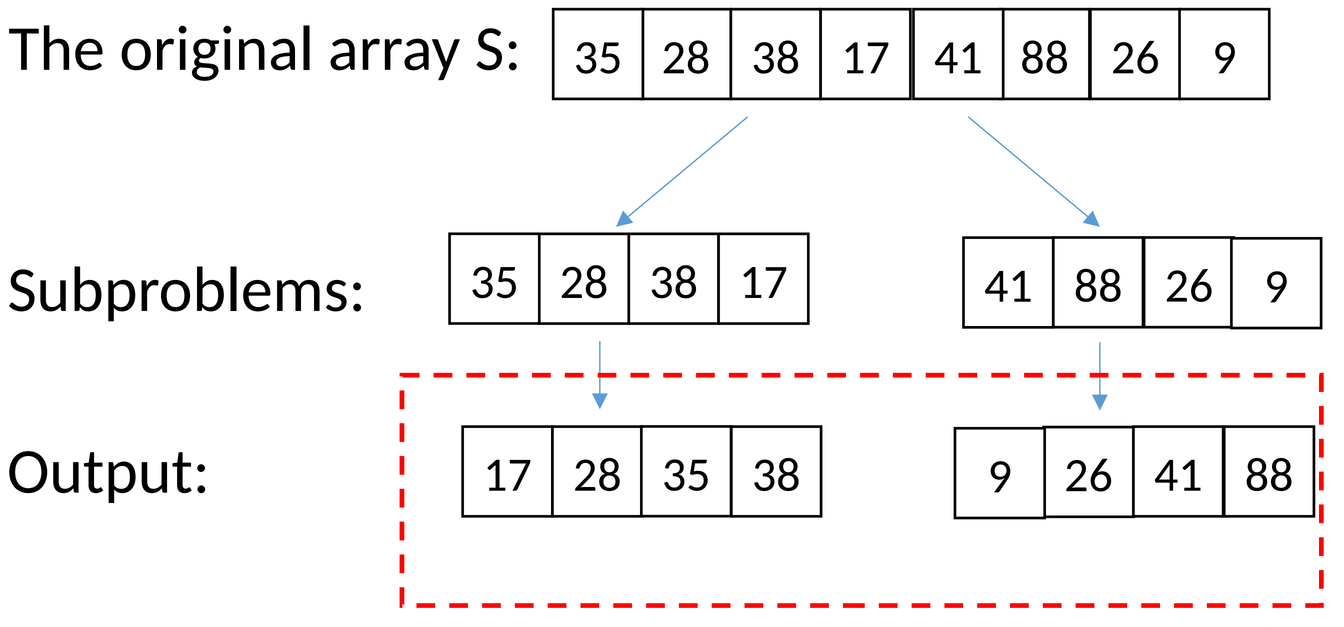
Review – Recursion Principle

- When dealing with a subproblem (same problem but with a smaller input)

1. Consider it solved;
2. Use its output to design the rest of the algorithm.

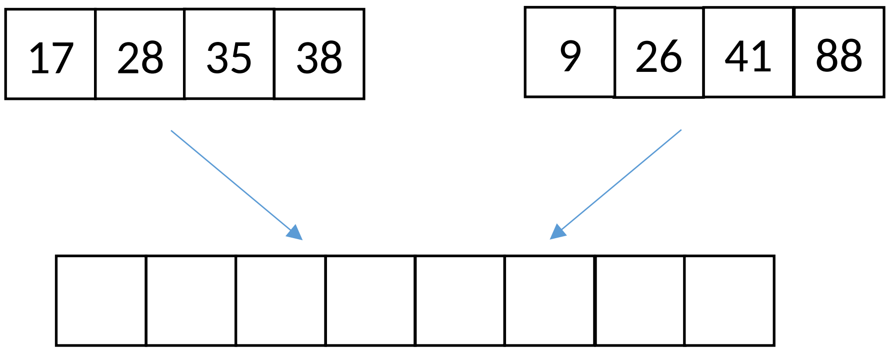
Review – Merge Sort

- Identify the subproblems:
  - Sort the first half of the array S.
  - Sort the second half of S.



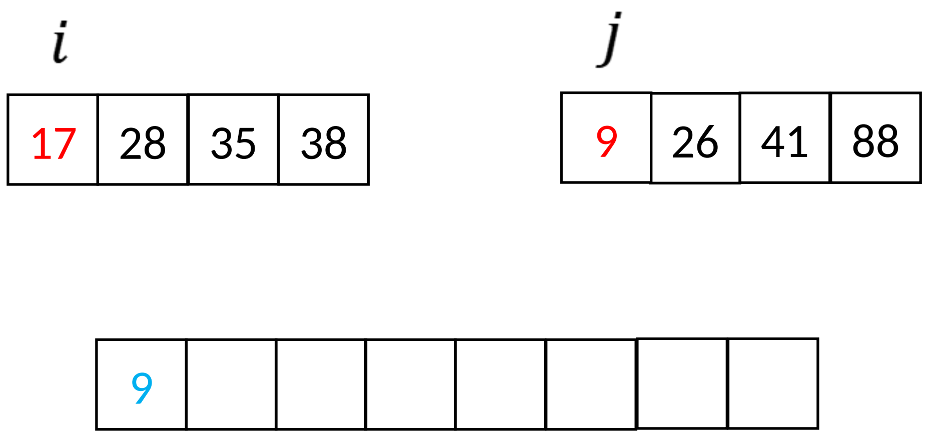
Review - Merge Operation

- Merge 2 sorted arrays into a single sorted array



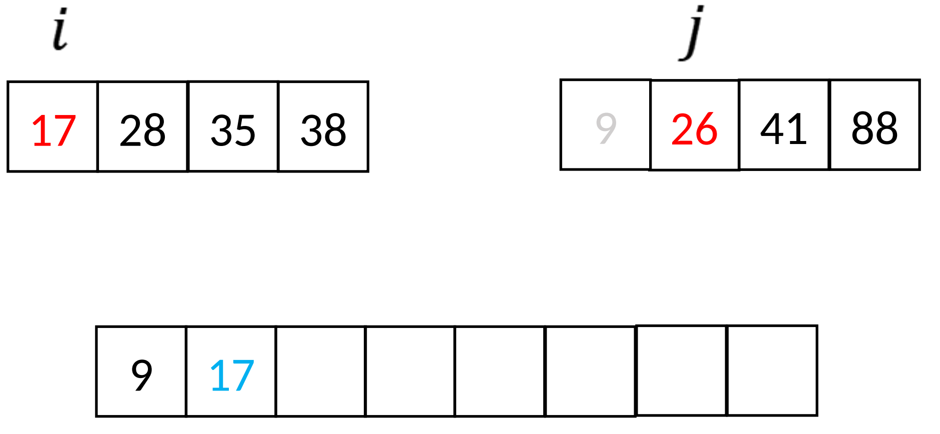
Review - Merge Operation

- Set *i*, *j* to 1
- Compare 17 and 9
- 9 is smaller
- Place 9 into the new array and increase *j* by 1



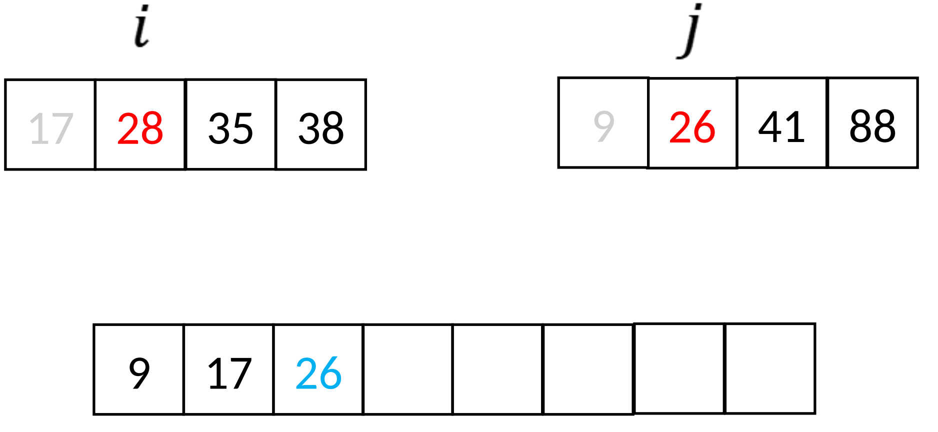
Review - Merge Operation

- Compare 17 and 26
- 17 is smaller
- Place 17 into the new array and increase *i* by 1



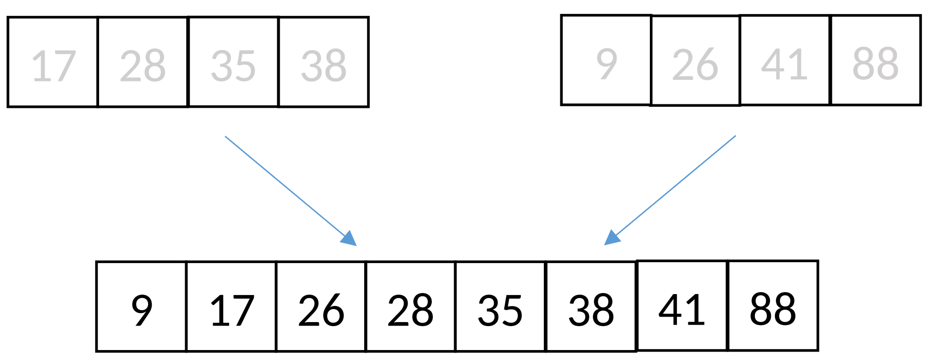
Review - Merge Operation

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



Review - Merge Operation

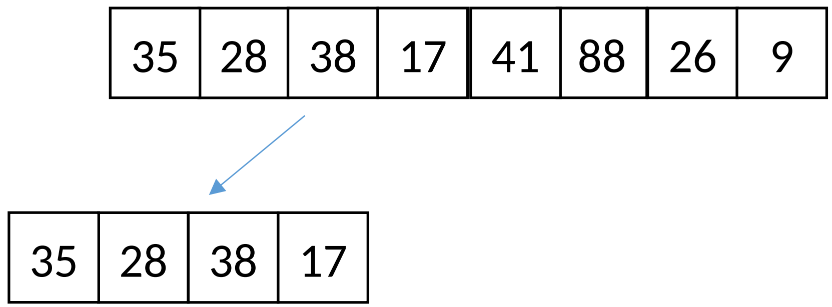
- Continue the above process until we have placed all elements into the new array
- Single pass over all the input elements
- Time complexity: $O(n)$



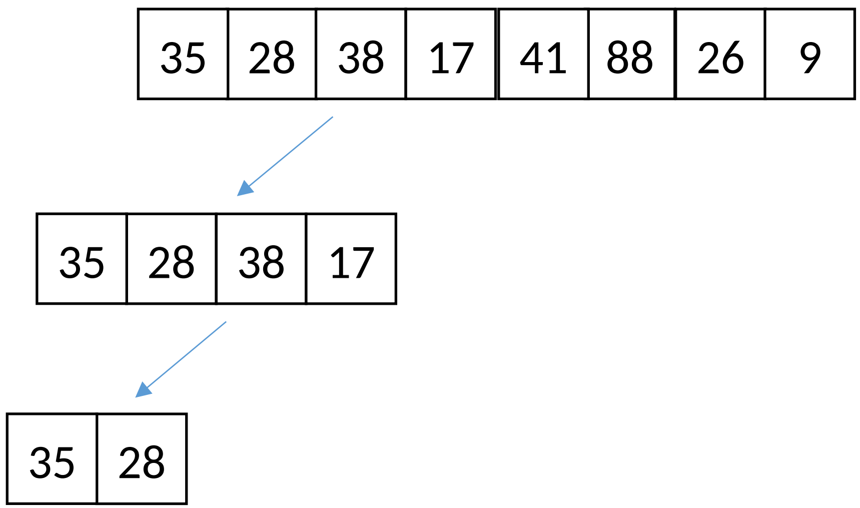
Execution Trace - Merge Sort

35	28	38	17	41	88	26	9

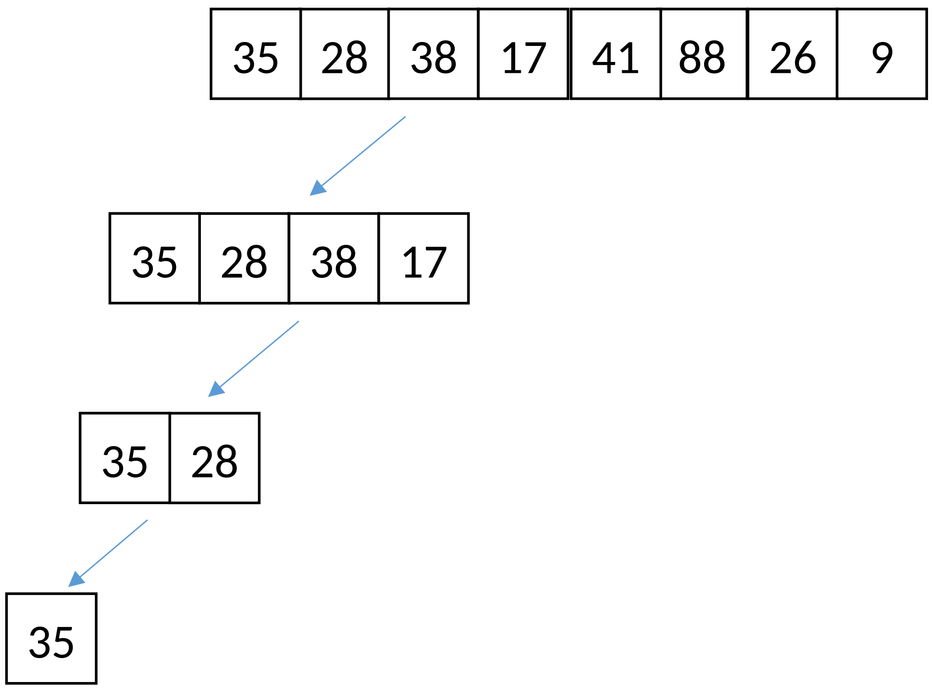
Execution Trace - Merge Sort



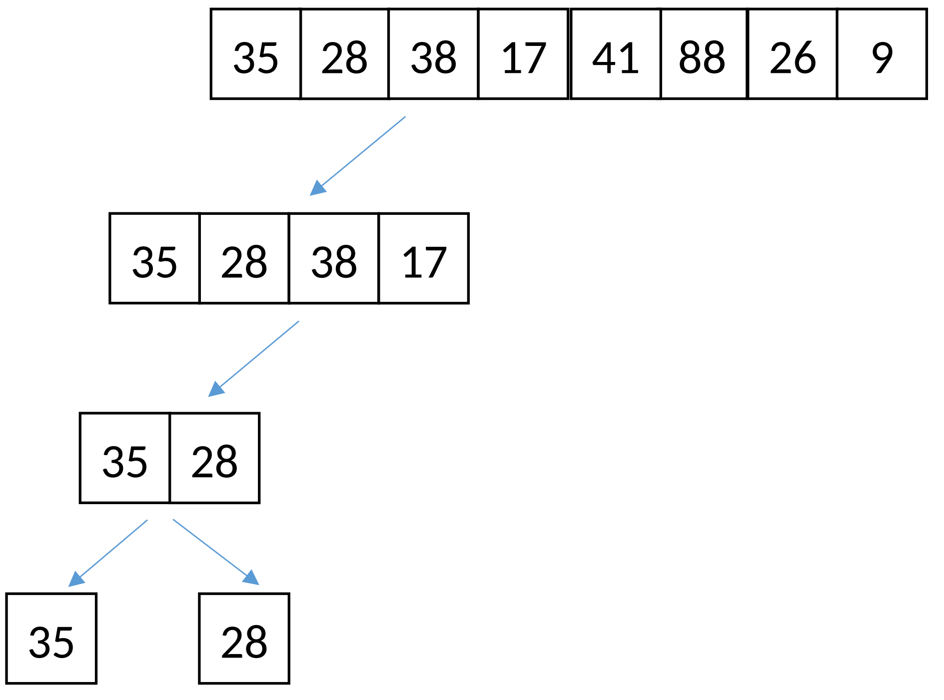
Execution Trace - Merge Sort



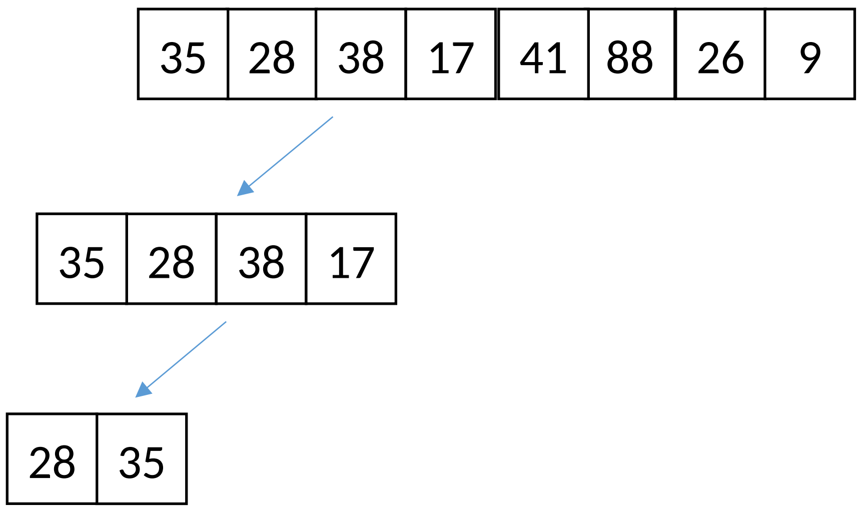
Execution Trace - Merge Sort



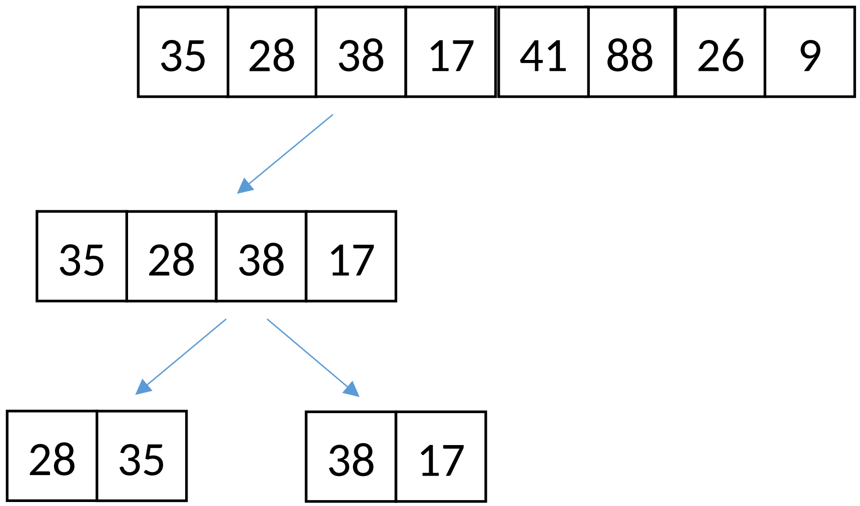
Execution Trace - Merge Sort



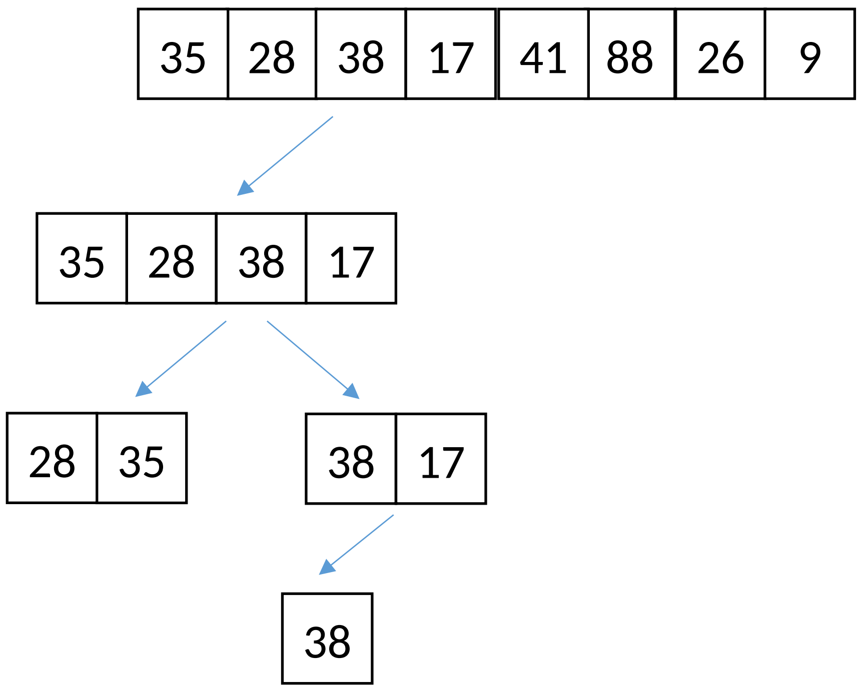
Execution Trace - Merge Sort



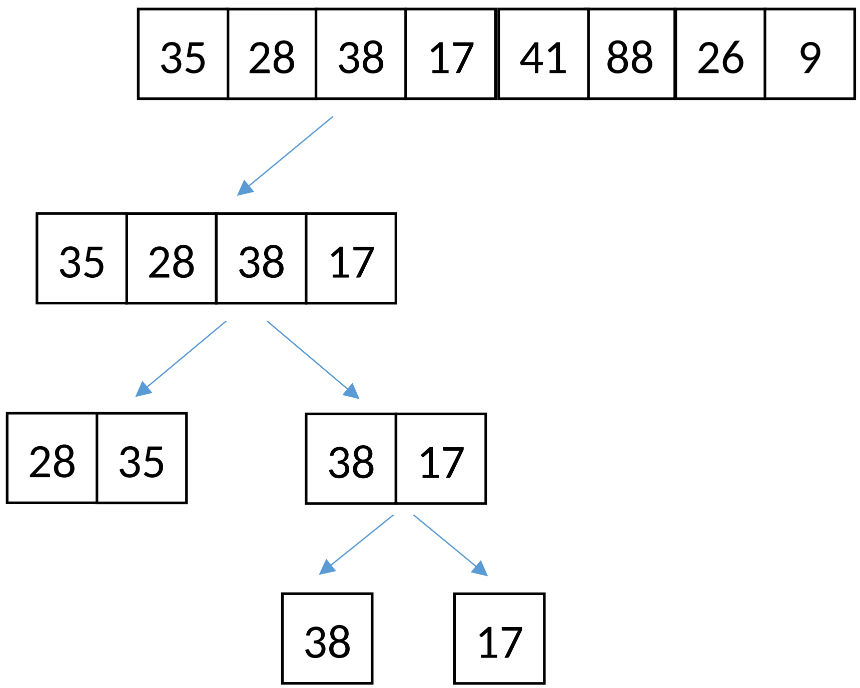
Execution Trace - Merge Sort



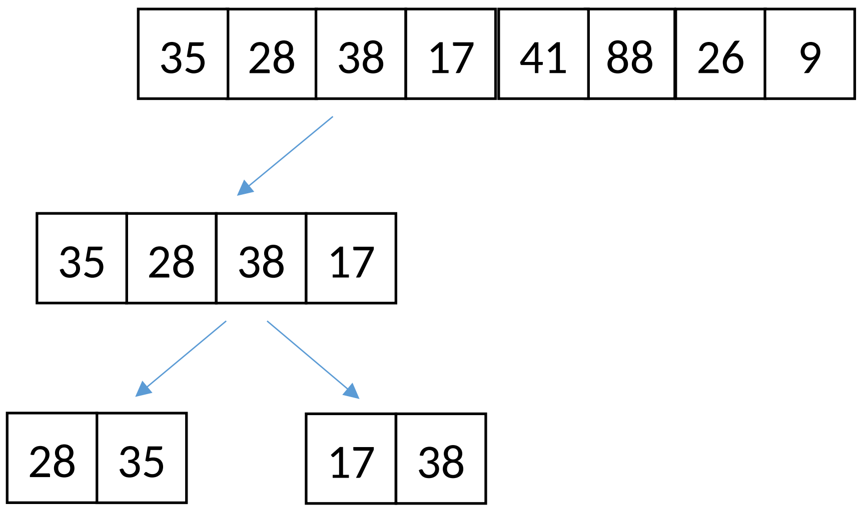
Execution Trace - Merge Sort



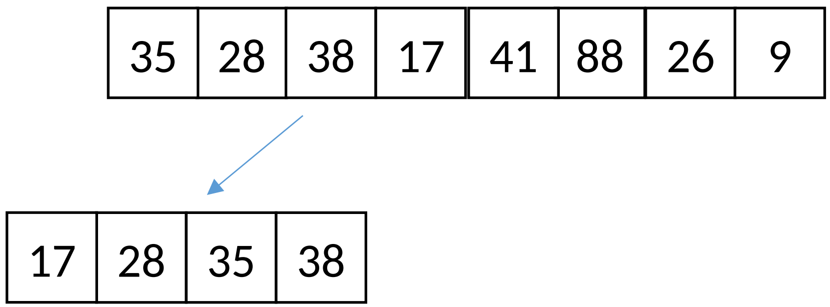
Execution Trace - Merge Sort



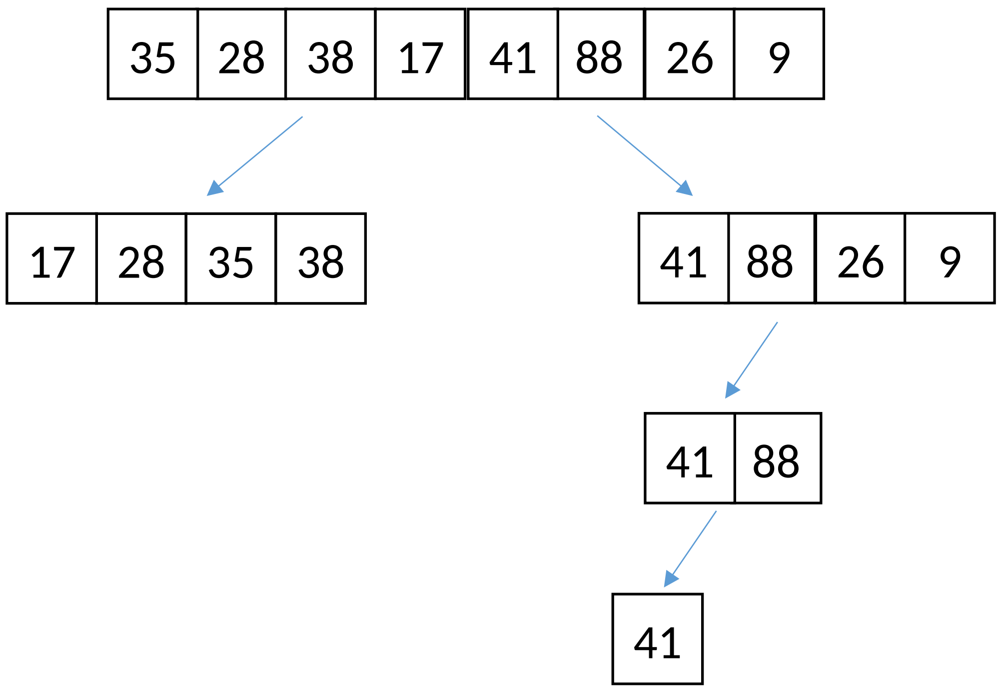
Execution Trace - Merge Sort



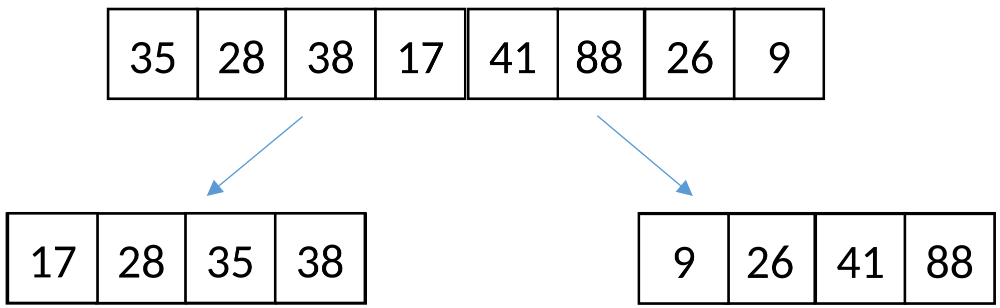
Execution Trace - Merge Sort



Execution Trace - Merge Sort



Execution Trace - Merge Sort

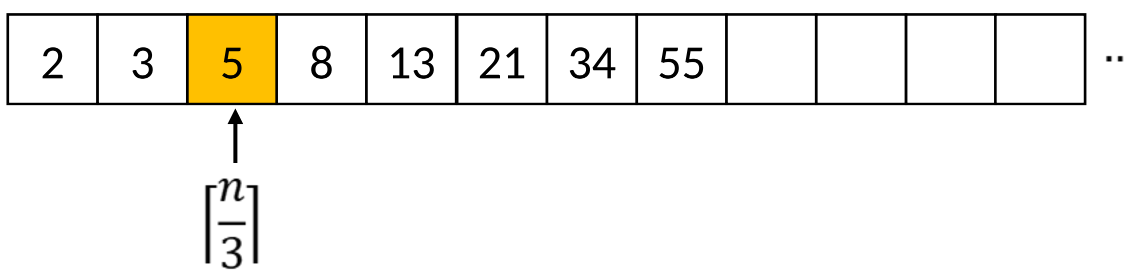


Execution Trace - Merge Sort

9	17	26	28	35	38	41	88

A Variant of Binary Search

- Instead of comparing the target value with the middle element, we compare the target with the $\lceil n/3 \rceil$-th element each time.



Time Complexity

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

Time Complexity

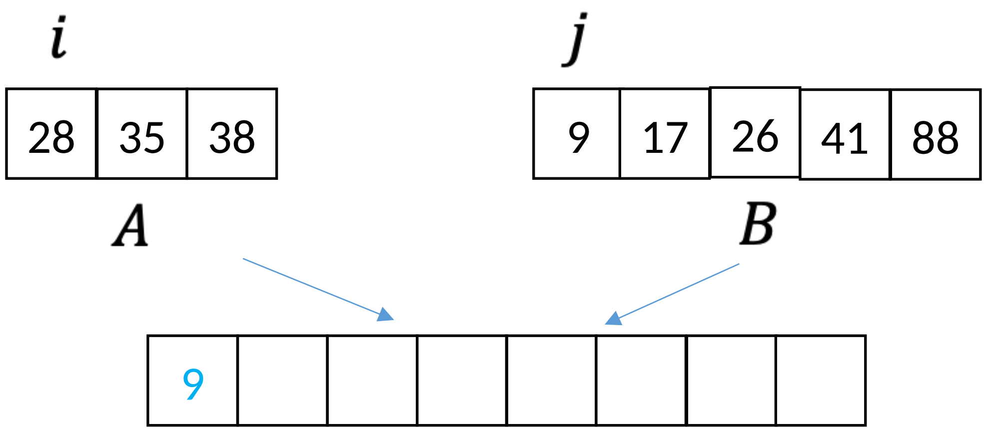
- What if we compare the target with the $\left\lceil \frac{n}{300} \right\rceil$-th element?
- The time complexity is still $O(\log n)$!
  - Try verifying this by yourself.

- In general, if the comparison is made to the $\left\lceil \frac{n}{k} \right\rceil$-th element for some constant $k > 1$, the time complexity is still $O(\log n)$.

Generalized Merge Operation

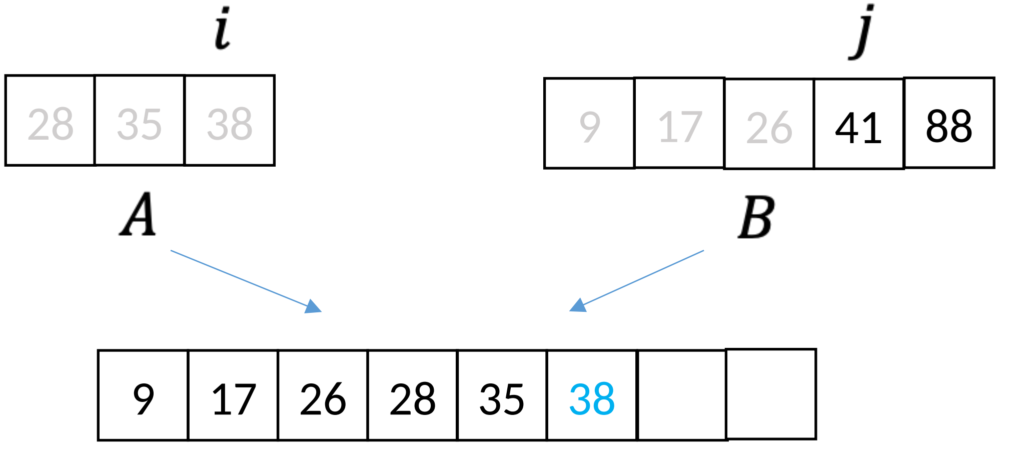
Merge 2 sorted arrays A and B, of lengths m and n.

- Set *i*, *j* to 1.
- Compare 9 with 25.
- Place 9 into the new array and increase i by 1.



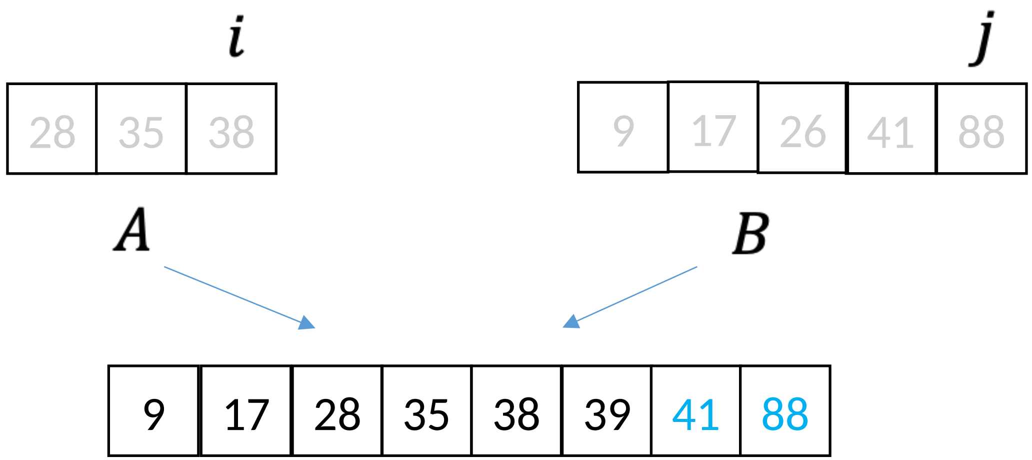
Generalized Merge Operation

- Repeat the process until we have put all the elements of one input array into the new array.



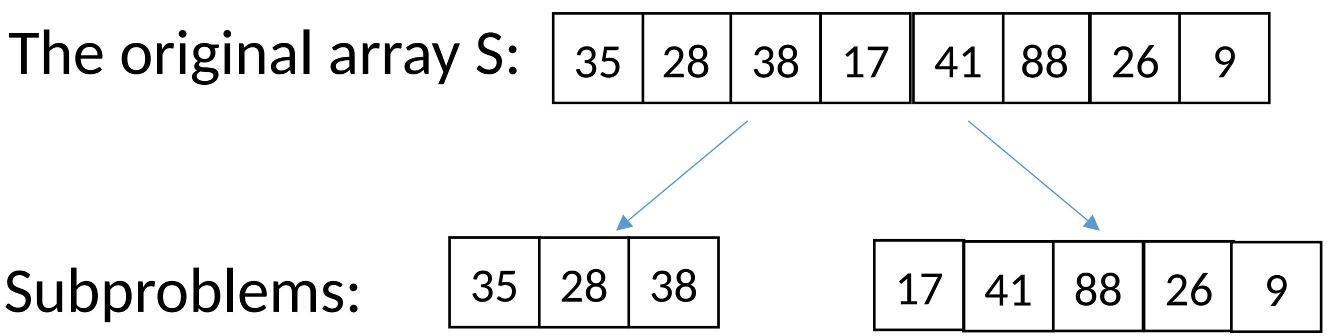
Generalized Merge Operation

- Append the remaining elements to the new array.
- Time complexity: $O(\mathrm{m}+n)$.



A Variant of Merge Sort

- Solve the subproblems:
  - Sort the first $\left\lceil \frac{n}{3} \right\rceil$ elements of the array S.
  - Sort the rest of S.
- Merge the 2 sorted arrays of different lengths.



Time Complexity

- The merging takes $O\left(\left\lceil\frac{2n}{3}\right\rceil + \left\lceil\frac{n}{3}\right\rceil\right) = O(n)$ time.
- Recurrence
  - $T(1) = O(1)$
  - $T(n) \leq T\left(\left\lceil\frac{2n}{3}\right\rceil\right) + T\left(\left\lceil\frac{n}{3}\right\rceil\right) + O(\mathrm{n})$
  - Solving the recurrence gives $T(n) = O(n \log n)$.
  - The recurrence can be solved with the substitution method (a regular exercise).

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 $O(m \log n)$ time.
  - 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

Sorted Sequence B	7	8	11	12	13