More on Merge Sort and Binary Search

CSCI2100 Tutorial 3

Adapted from the slides of the previous offerings of the course

Outline

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

Review – Recursion Principle

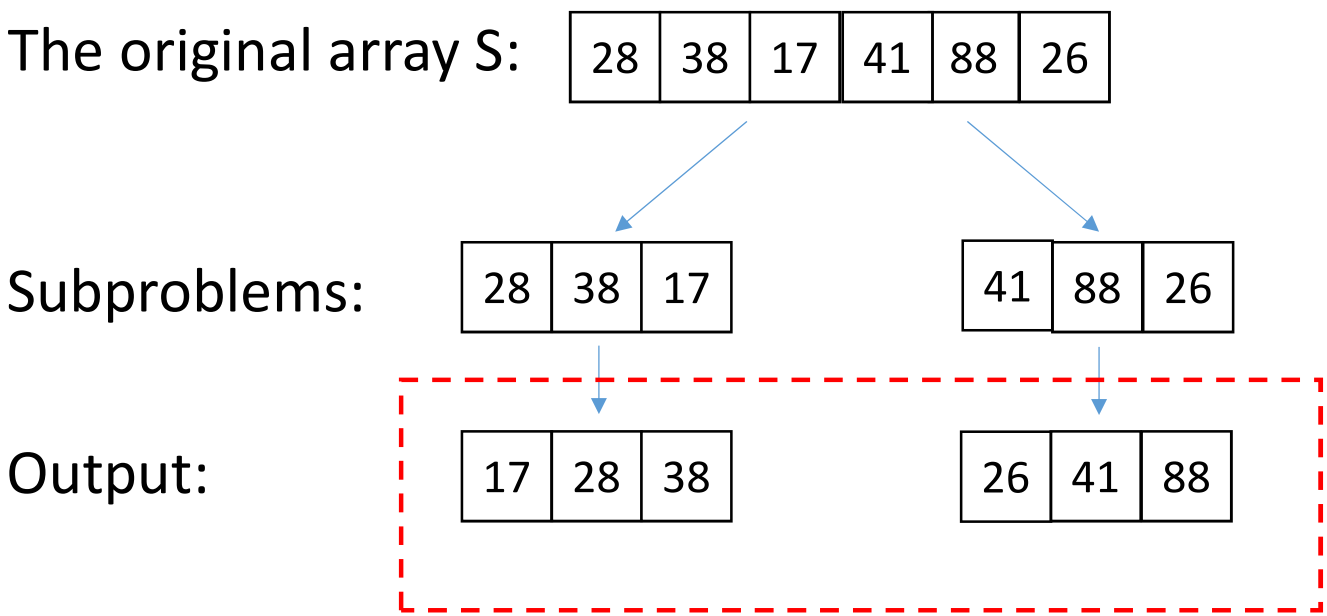
- When dealing with a subproblem (same problem but with a smaller input), consider it solved.

1. We consider that the subproblem has already been solved.

2. We can directly use the output of the subproblem in the rest algorithm design.

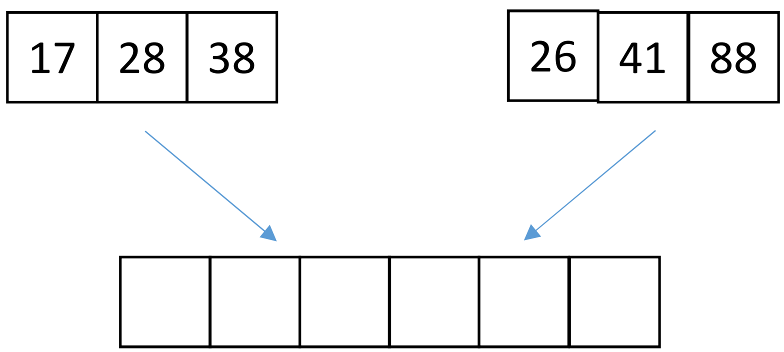
Review – Merge Sort

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



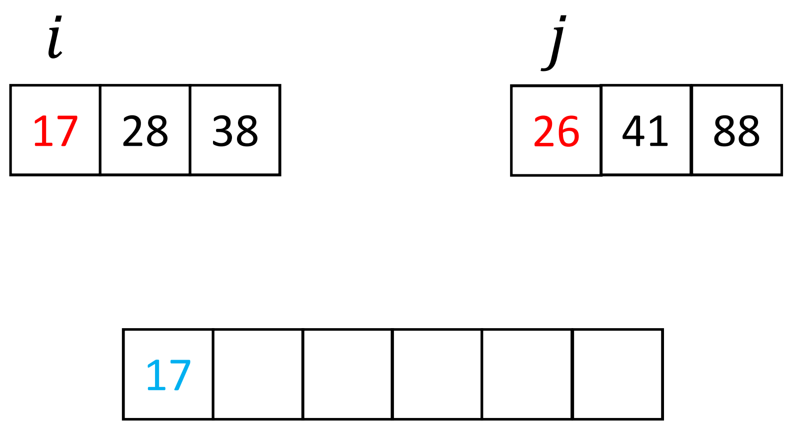
Review - Merge Operation

- Merge 2 sorted arrays into a single sorted array



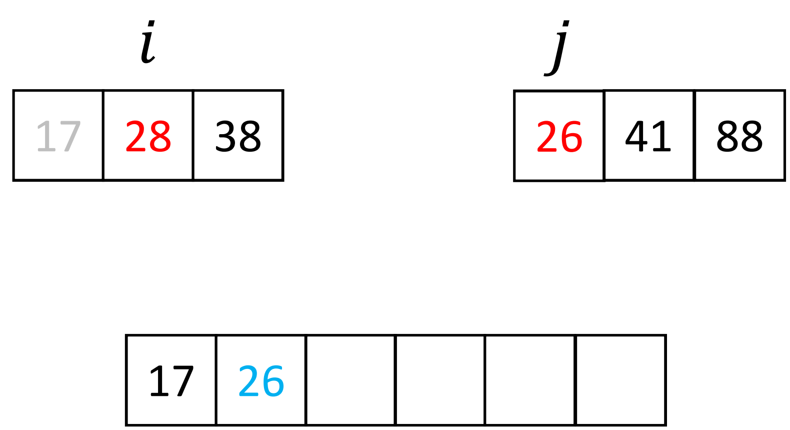
Review - Merge Operation

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



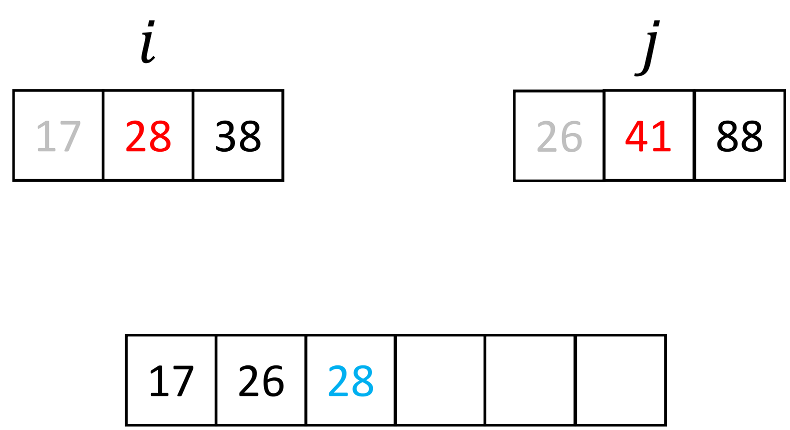
Review - Merge Operation

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



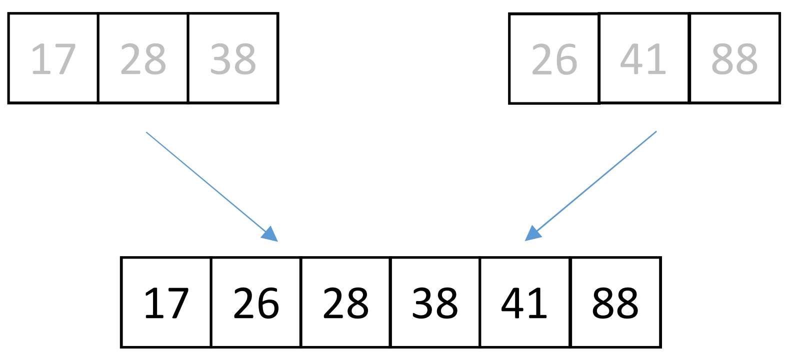
Review - Merge Operation

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



Review - Merge Sort Time Complexity

- Let $f(n)$ be the worst case time
- $f(n) \leq 2f\left(\left\lceil \frac{n}{2} \right\rceil\right) + O(n)$

- By Master theorem we can get $f(n) = O(n \log n)$

- Note that it suffices to analyze only one level of the algorithm due to recursion.

Exercise: Modified Merge Sort

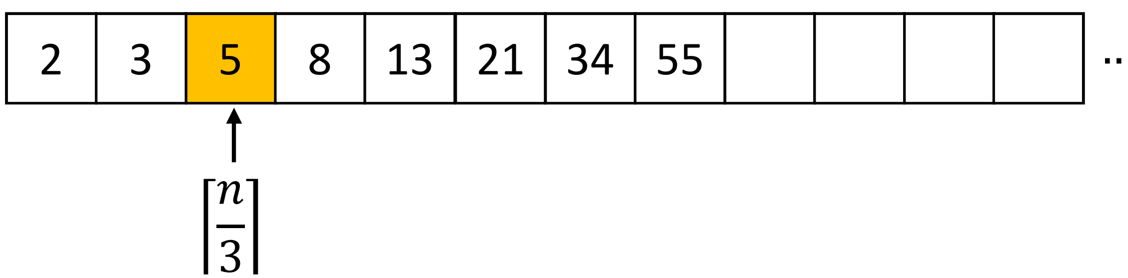
- Regular Exercise 3 Problem 6
- A variant of merge sort
  - If $n = 1$ then return immediately
  - Otherwise set $k = \lceil n/3 \rceil$
  - Recursively sort $A[1 \dots k]$ and $A[k+1 \dots n]$, respectively
  - Merge $A[1 \dots k]$ and $A[k+1 \dots n]$ into one sorted array
- Prove the time complexity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

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.



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 also $O(\log n)$!
  - 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 $O(\log n)$.

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

Sequence B	11	**8**	7	12	13