COMP3251
Lecture 2: Merge Sort (Chapter 2.3)

Sorting

Input: A set of n integers — $x_1, x_2, \ldots, x_n$.

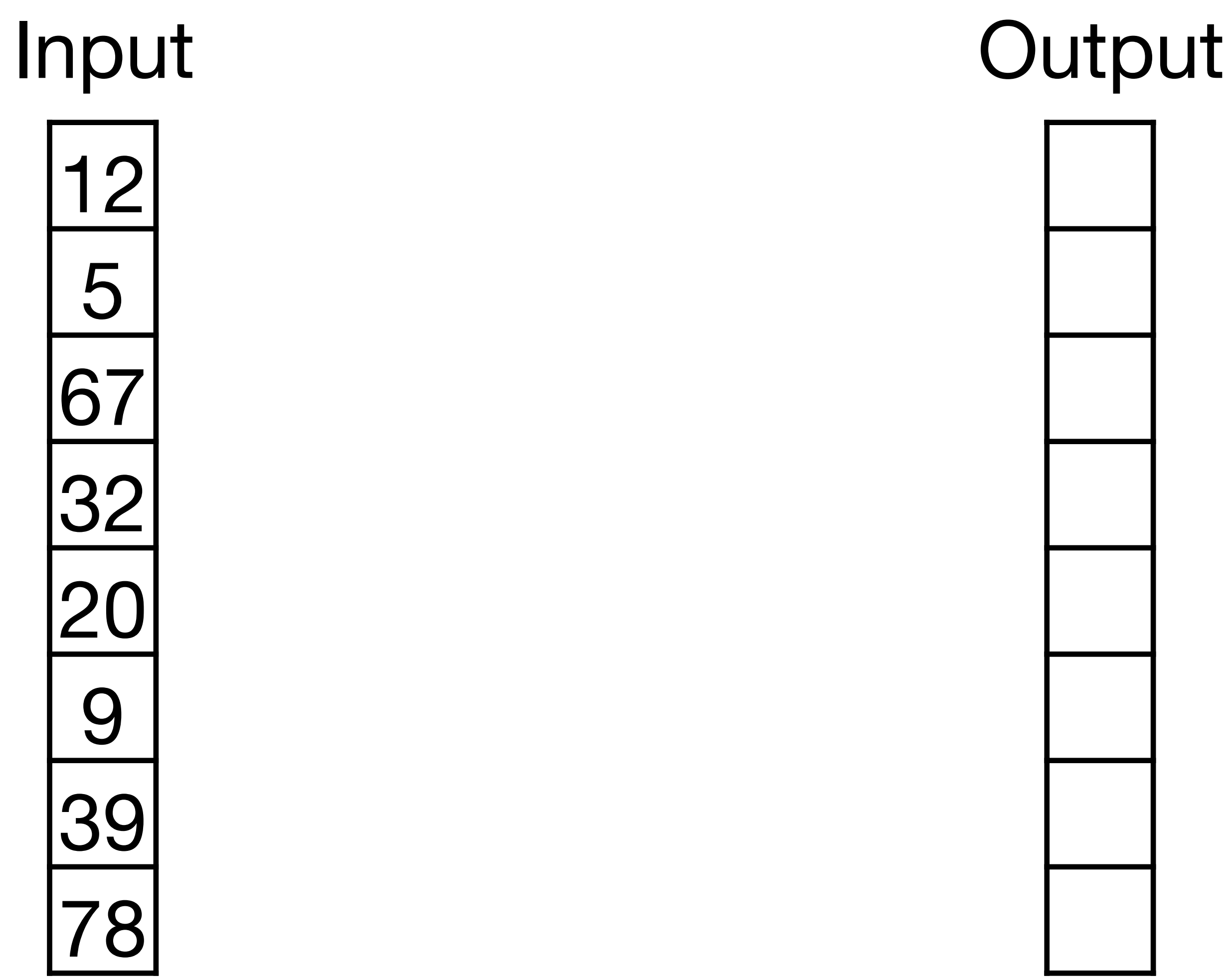
Output: The same set of n integers in ascending order.

Sorting

Input: A set of n integers — $x_1, x_2, \ldots, x_n$.

Output: The same set of n integers in ascending order.

Warm-up: The insertion sort algorithm.

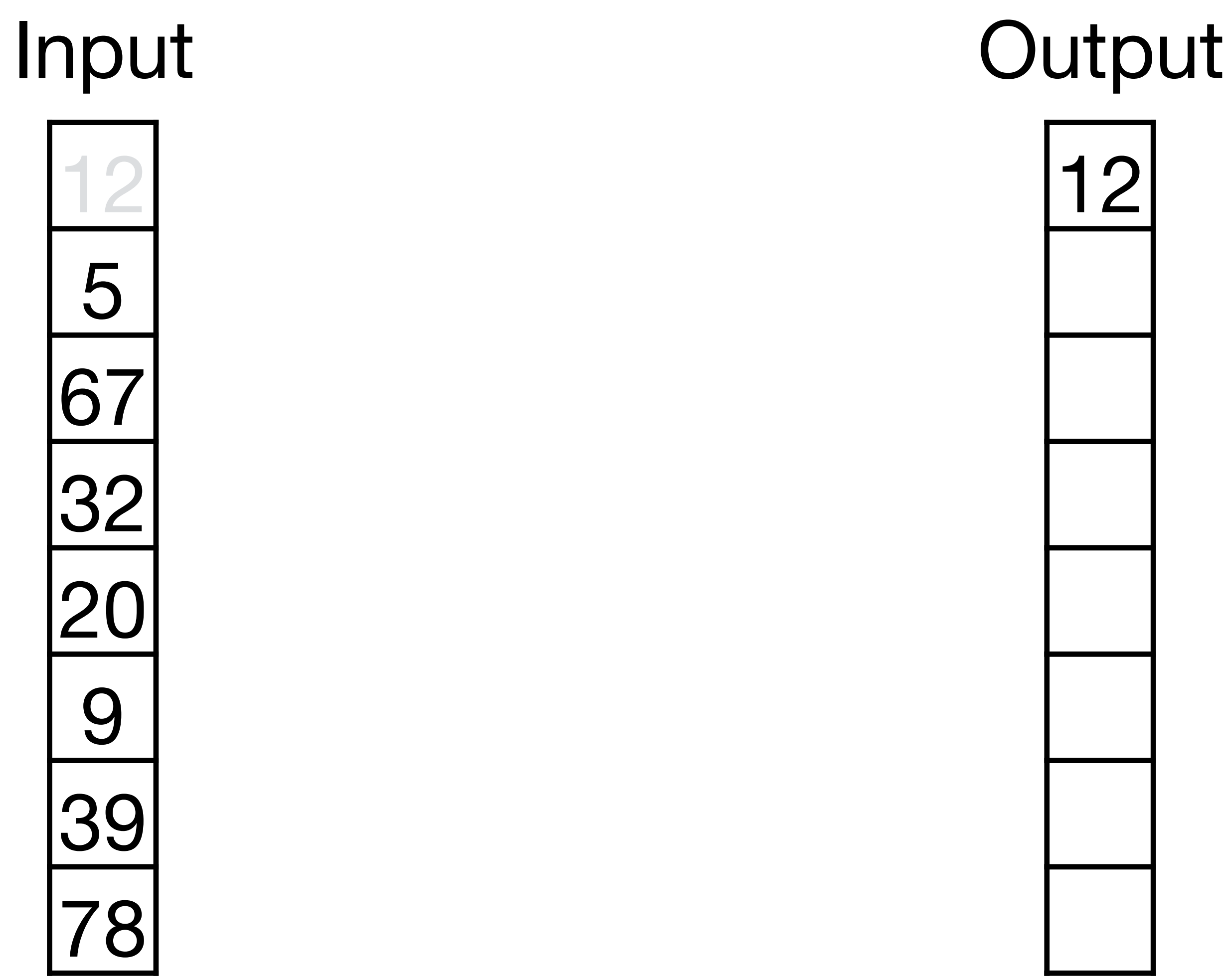


Sorting

Input: A set of n integers — $x_1, x_2, \ldots, x_n$.

Output: The same set of n integers in ascending order.

Warm-up: The insertion sort algorithm.

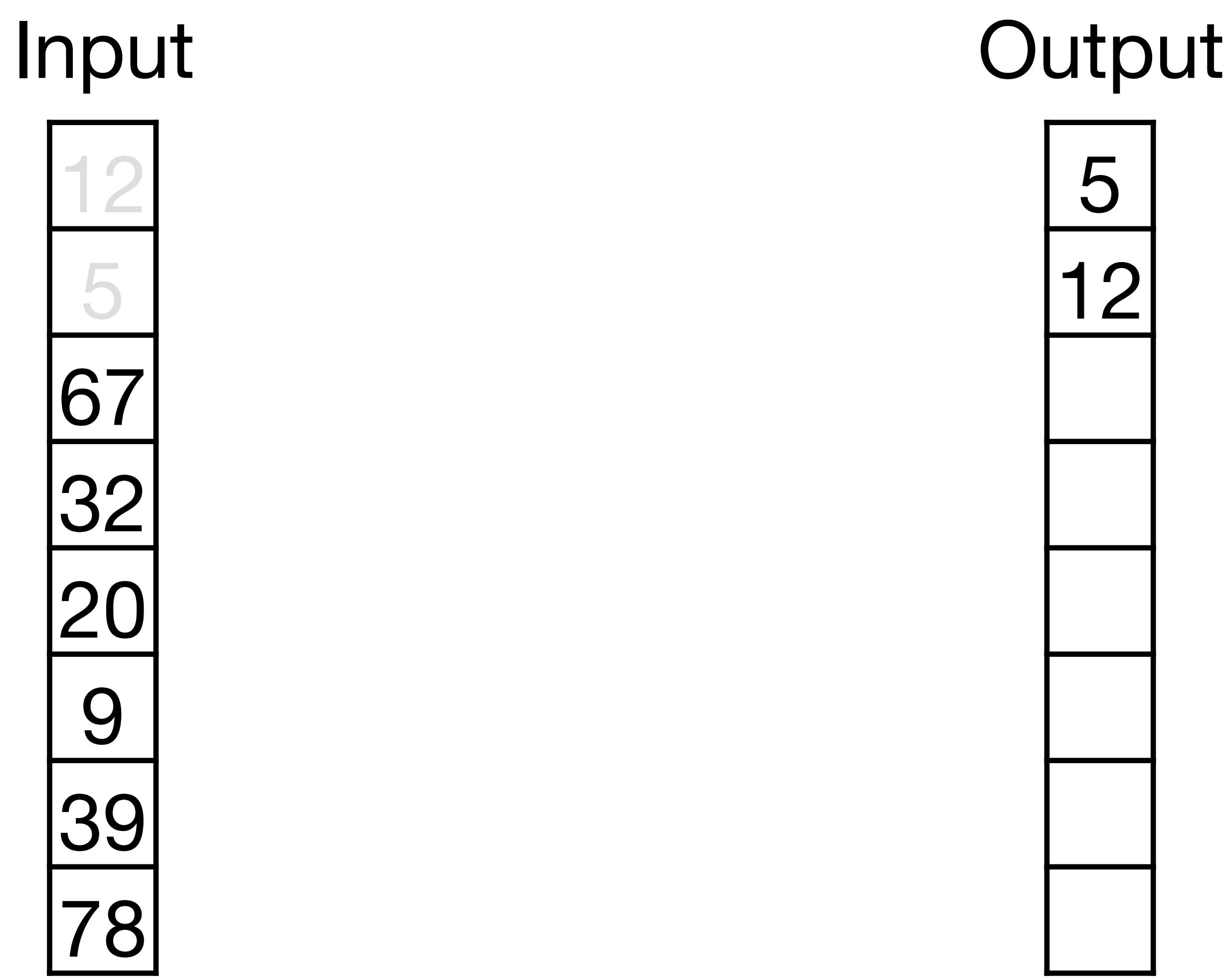


Sorting

Input: A set of n integers — $x_1, x_2, \ldots, x_n$.

Output: The same set of n integers in ascending order.

Warm-up: The insertion sort algorithm.



Sorting

Input: A set of n integers — $x_1, x_2, \ldots, x_n$.

Output: The same set of n integers in ascending order.

Warm-up: The insertion sort algorithm.

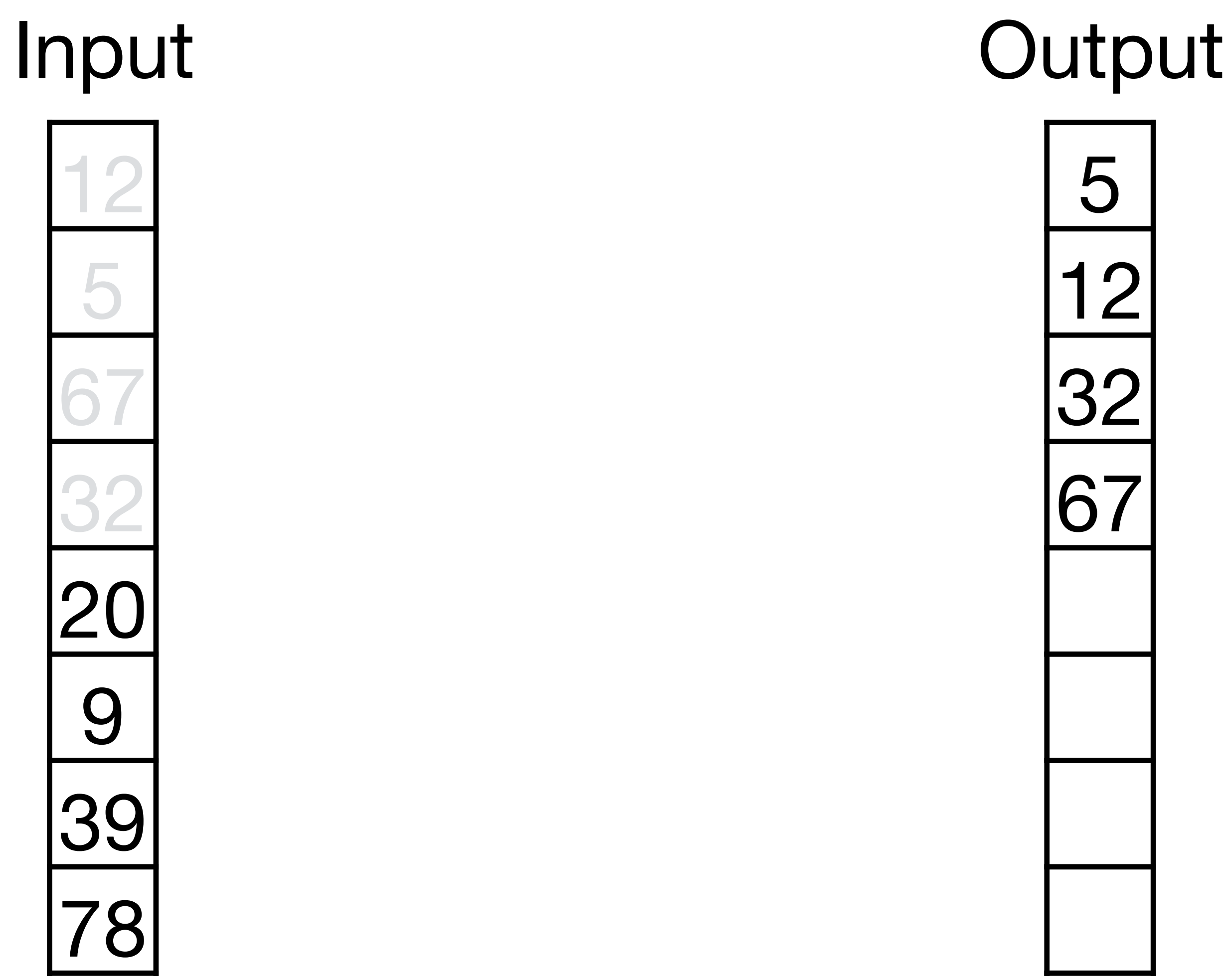
Input	Output
12	5
5	12
67	67
32	
20	
9	
39	
78	

Sorting

Input: A set of n integers — $x_1, x_2, \ldots, x_n$.

Output: The same set of n integers in ascending order.

Warm-up: The insertion sort algorithm.

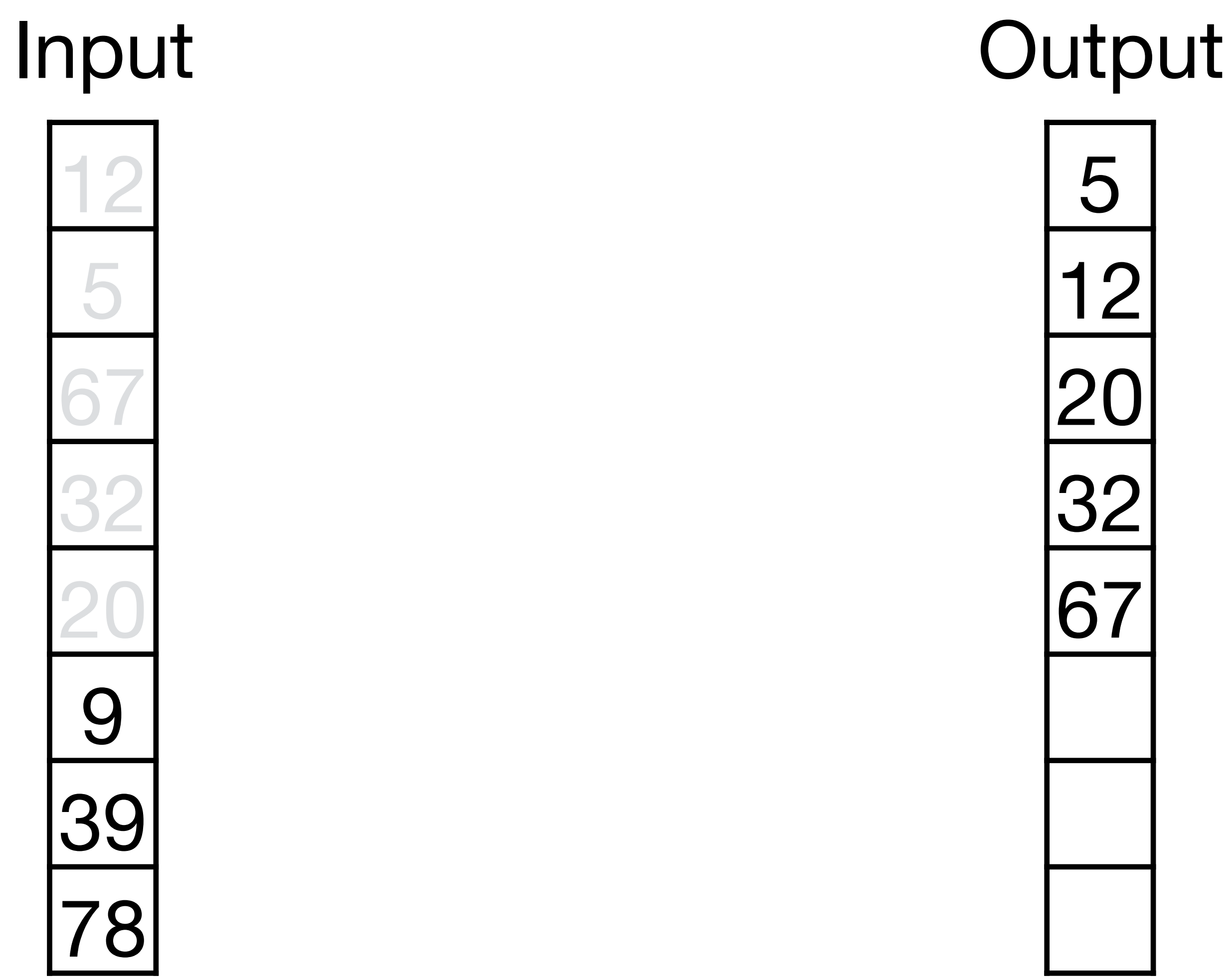


Sorting

Input: A set of n integers — $x_1, x_2, \ldots, x_n$.

Output: The same set of n integers in ascending order.

Warm-up: The insertion sort algorithm.

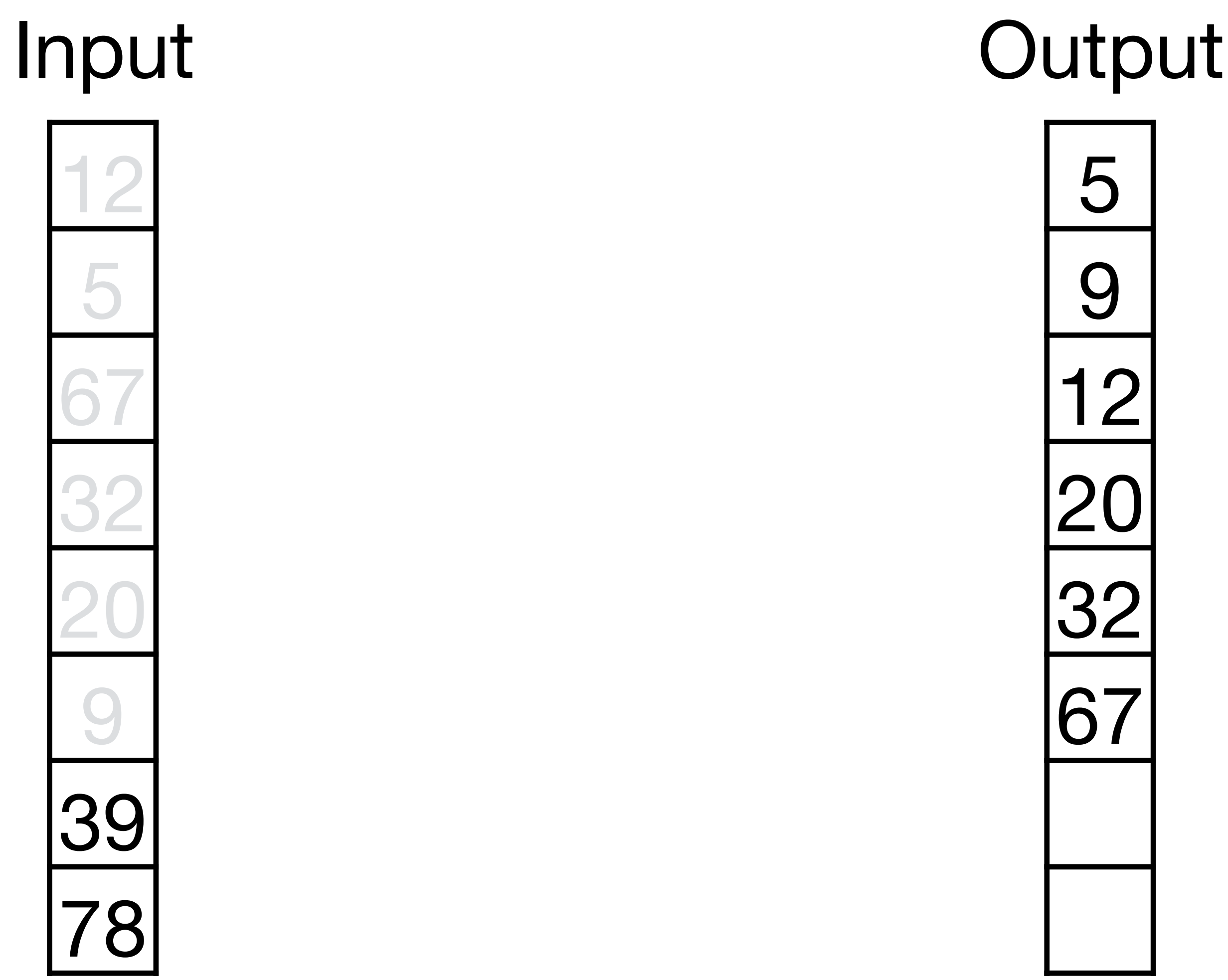


Sorting

Input: A set of n integers — $x_1, x_2, \ldots, x_n$.

Output: The same set of n integers in ascending order.

Warm-up: The insertion sort algorithm.

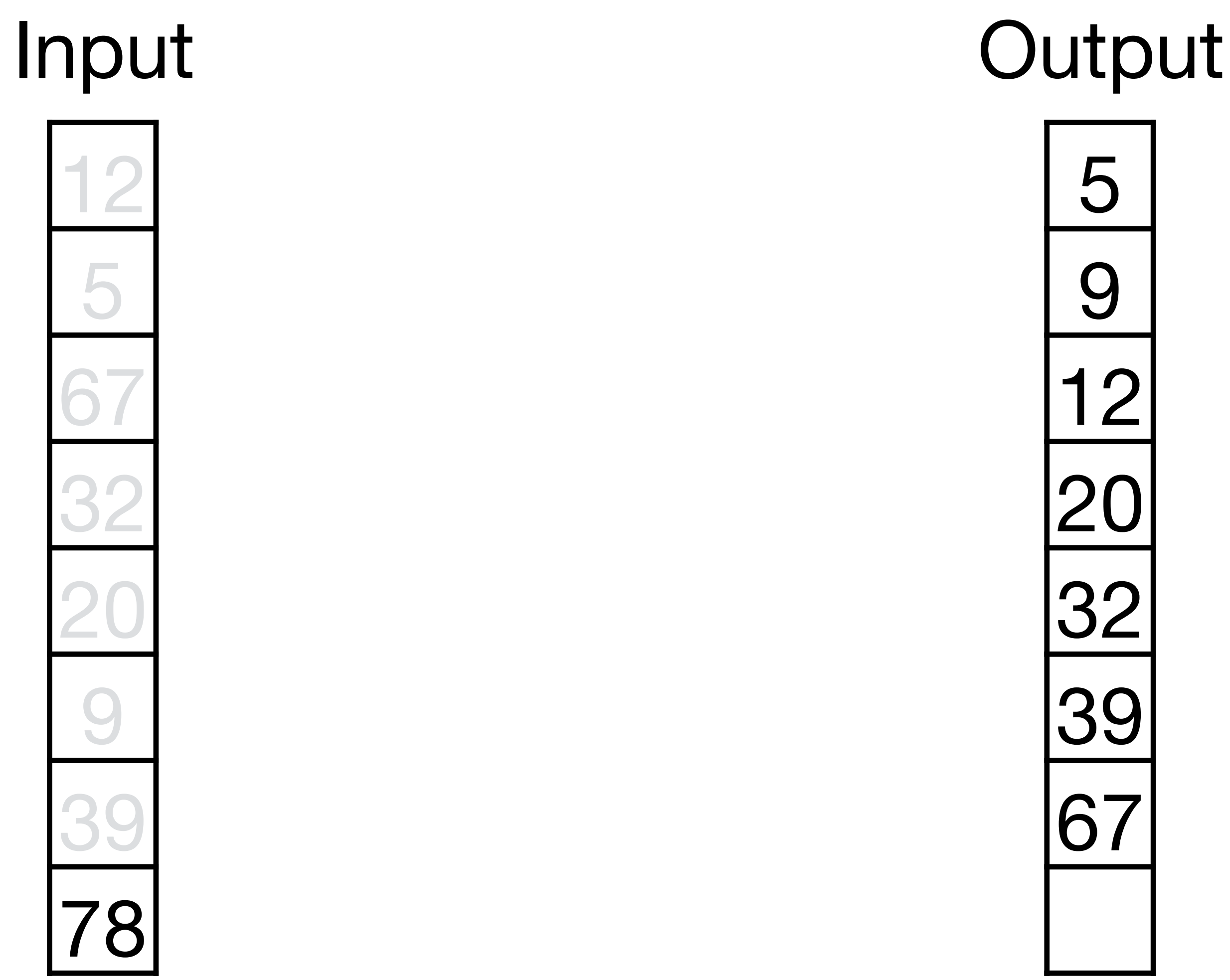


Sorting

Input: A set of n integers — $x_1, x_2, \ldots, x_n$.

Output: The same set of n integers in ascending order.

Warm-up: The insertion sort algorithm.



Sorting

Input: A set of n integers — $x_1, x_2, \ldots, x_n$.

Output: The same set of n integers in ascending order.

Warm-up: The insertion sort algorithm.

Input	Output
12	5
5	9
67	12
32	20
20	32
9	39
39	67
78	78

How good is insertion sort?

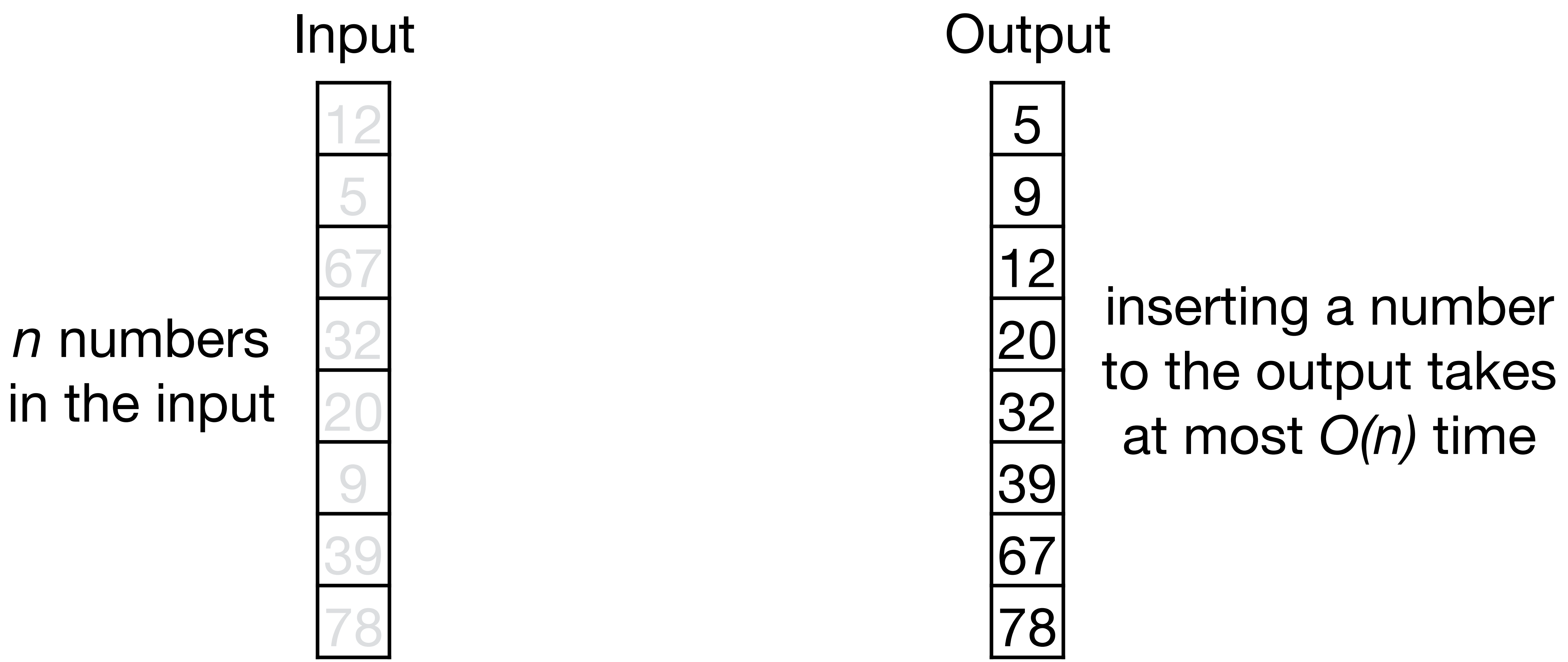
- **Worst-case analysis**
  - For any input length n, consider the maximum number of steps the algorithm needs on the **worst input** of length n.
  - Measuring performance instance by instance does not lead to reasonable algorithms (too specialized).
  - Average performance is a reasonable alternative, but **not** the focus of this course.
- **Asymptotic analysis**
  - Big-O notation
  - Focus on the performance when n gets really big
  - **Example:** $5n^2+2n+10 = O(n^2)$

Sorting

Input: A set of n integers — $x_1, x_2, \ldots, x_n$.

Output: The same set of n integers in ascending order.

Warm-up: The insertion sort algorithm takes $O(n^2)$ time.



Divide and Conquer (Ch. 2)

The divide-and-conquer algorithm design paradigm solves a problem as follows:

1) **Divide:** Breaking the problem into subproblems that are themselves smaller instances of the same type of problem;
2) **Recurse:** Recursively solving these subproblems;
3) **Combine:** Appropriately combining their answers to get an answer of the original problem.

Note: If the size of a subproblem is small enough, we will stop using the divide-and-conquer strategy; instead, we may solve the subproblem by brute-force.

Merge Sort

Input: A set of n integers — $x_1, x_2, \ldots, x_n$.

Output: The same set of n integers in ascending order.

Divide:	Divide the input integers into two subsets $x_1, \ldots, x_{n/2}$ and $x_{n/2+1}, \ldots, x_n$.
Recurse:	Sort the two subsets (n/2-size subproblems). Let $y_1, \ldots, y_{n/2}$ and $y_{n/2+1}, \ldots, y_n$ be the output.
Combine:	Merge two sorted list $y_1, \ldots, y_{n/2}$ and $y_{n/2+1}, \ldots, y_n$ into a single sorted lists of integers.

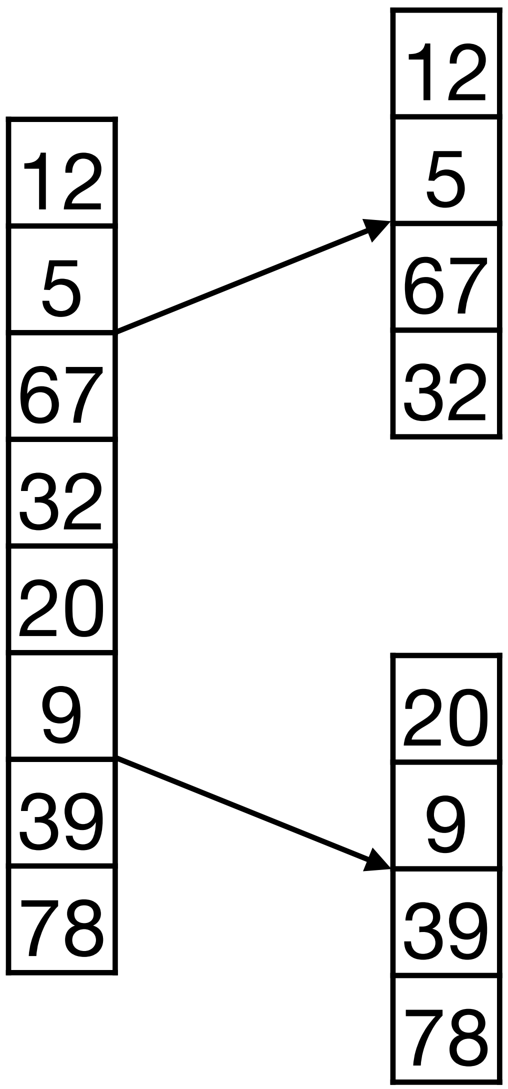
Running time: $T(n)$, total time for sorting a set of n integers.

$T(n) = 2T(n/2) + (\textit{running time of merging two n/2-size sorted lists})$

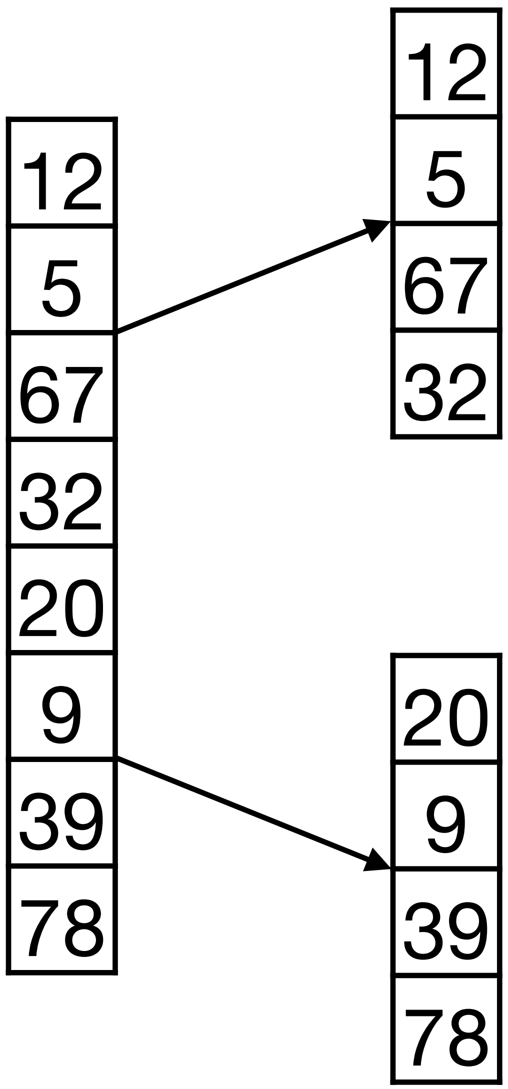
An Example of Merge Sort

12
5
67
32
20
9
39
78

An Example of Merge Sort

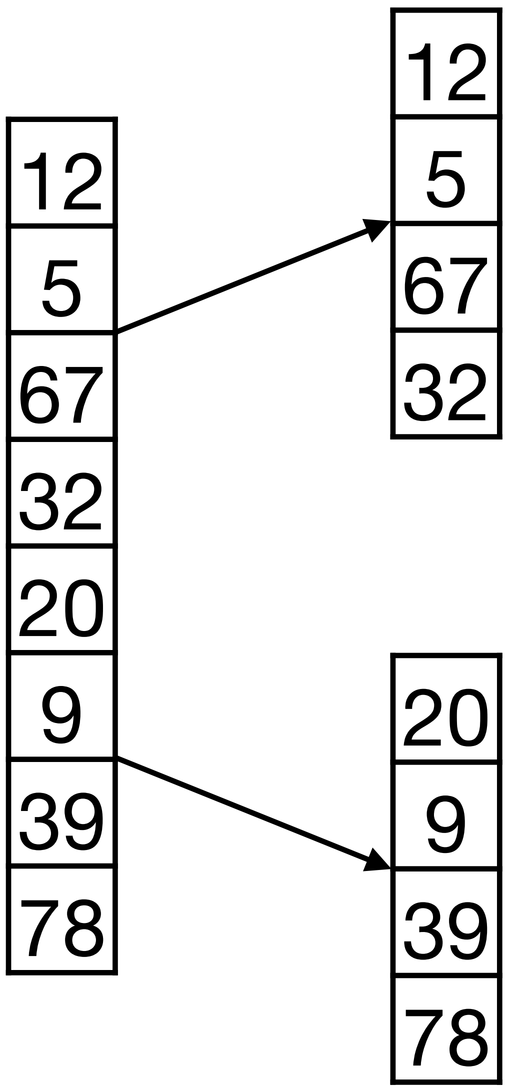


An Example of Merge Sort



1) Sort the top list recursively

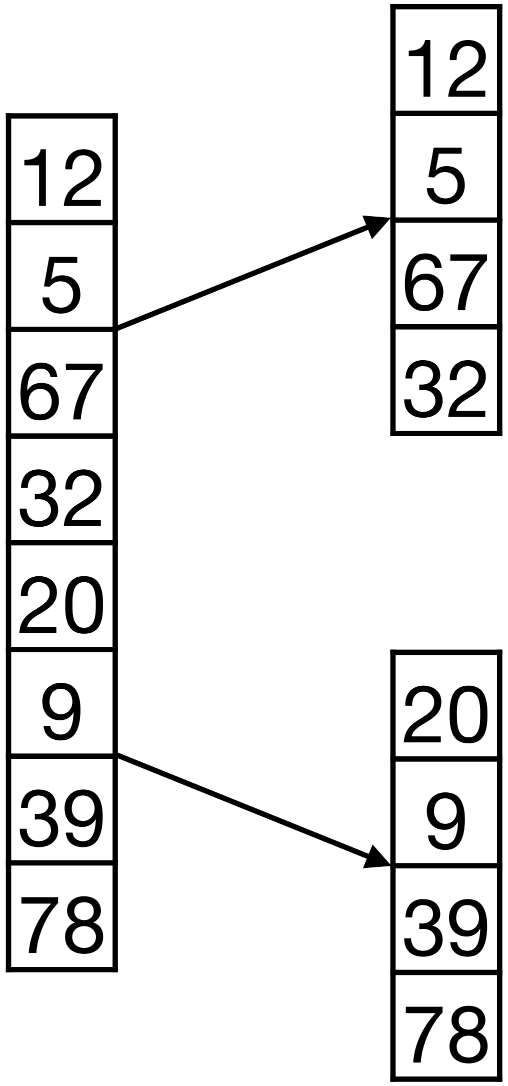
An Example of Merge Sort



1) Sort the top list recursively

2) Sort the bottom list recursively

An Example of Merge Sort

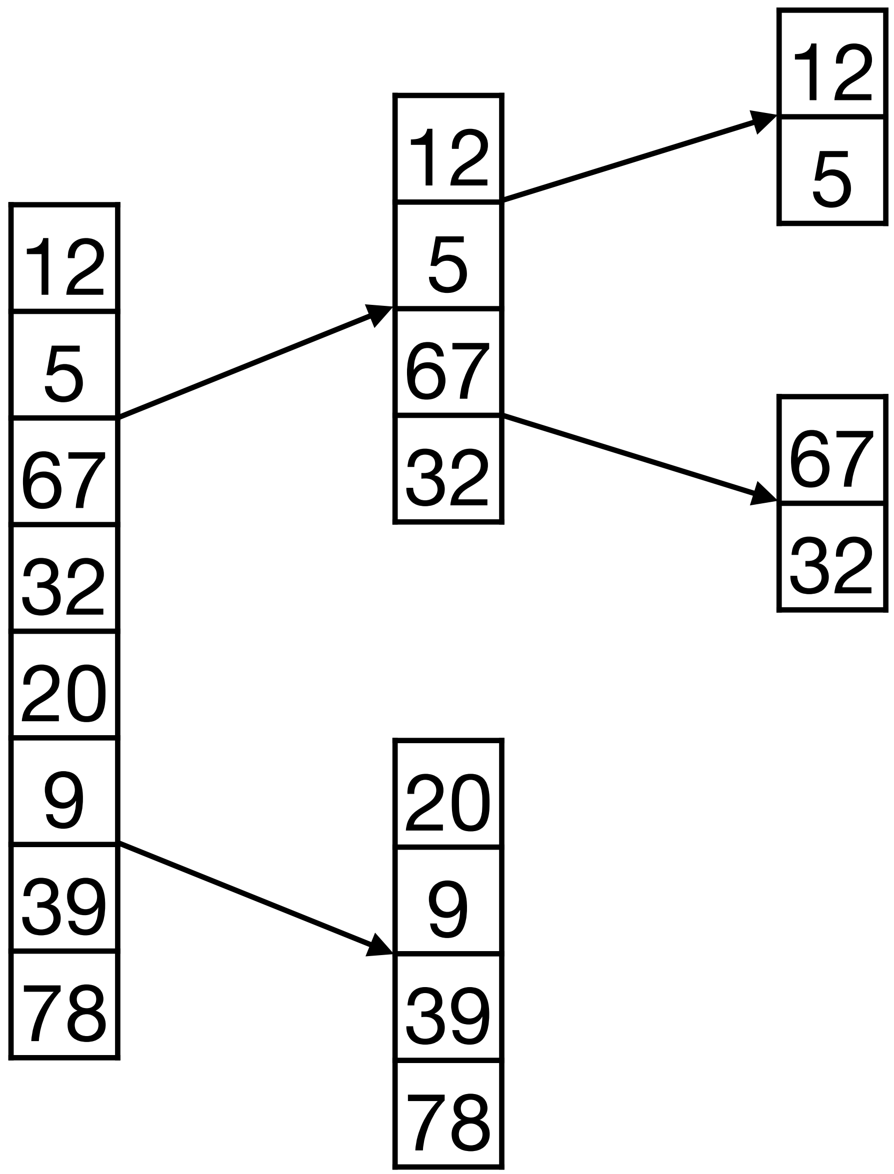


1) Sort the top list recursively

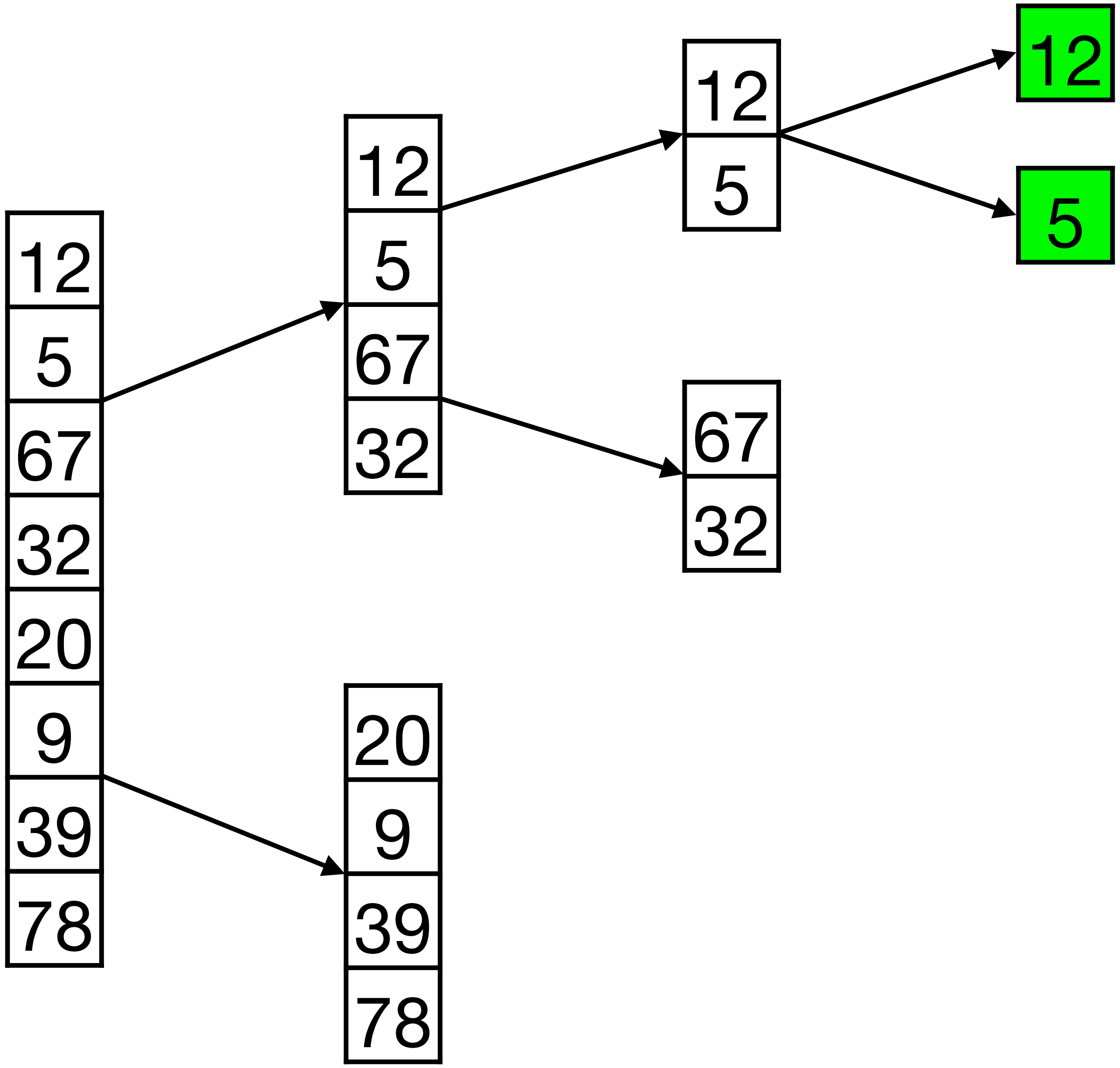
2) Sort the bottom list recursively

3) Merge the two sorted list output by the subproblems to get a sorted list of all integers

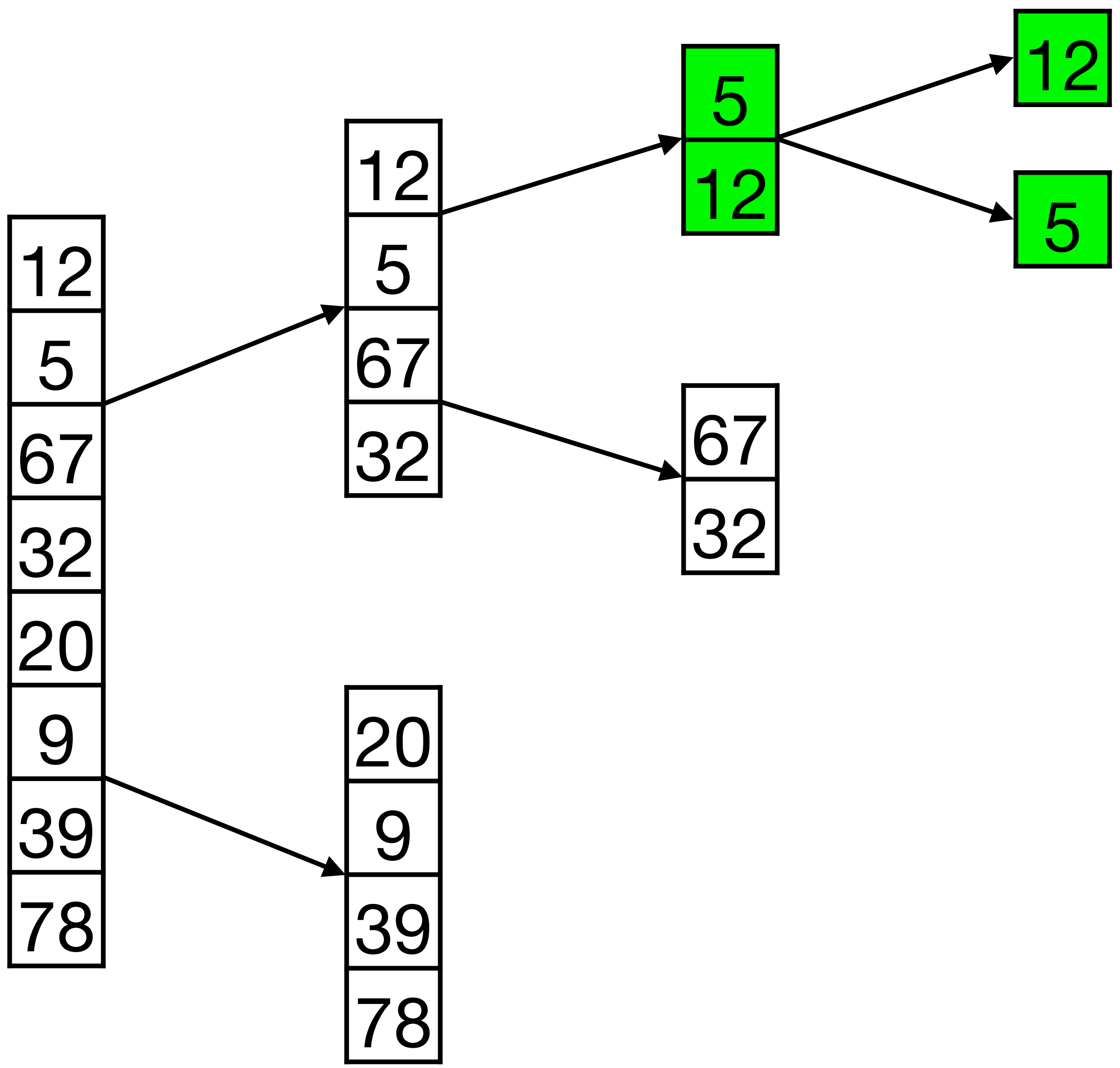
An Example of Merge Sort



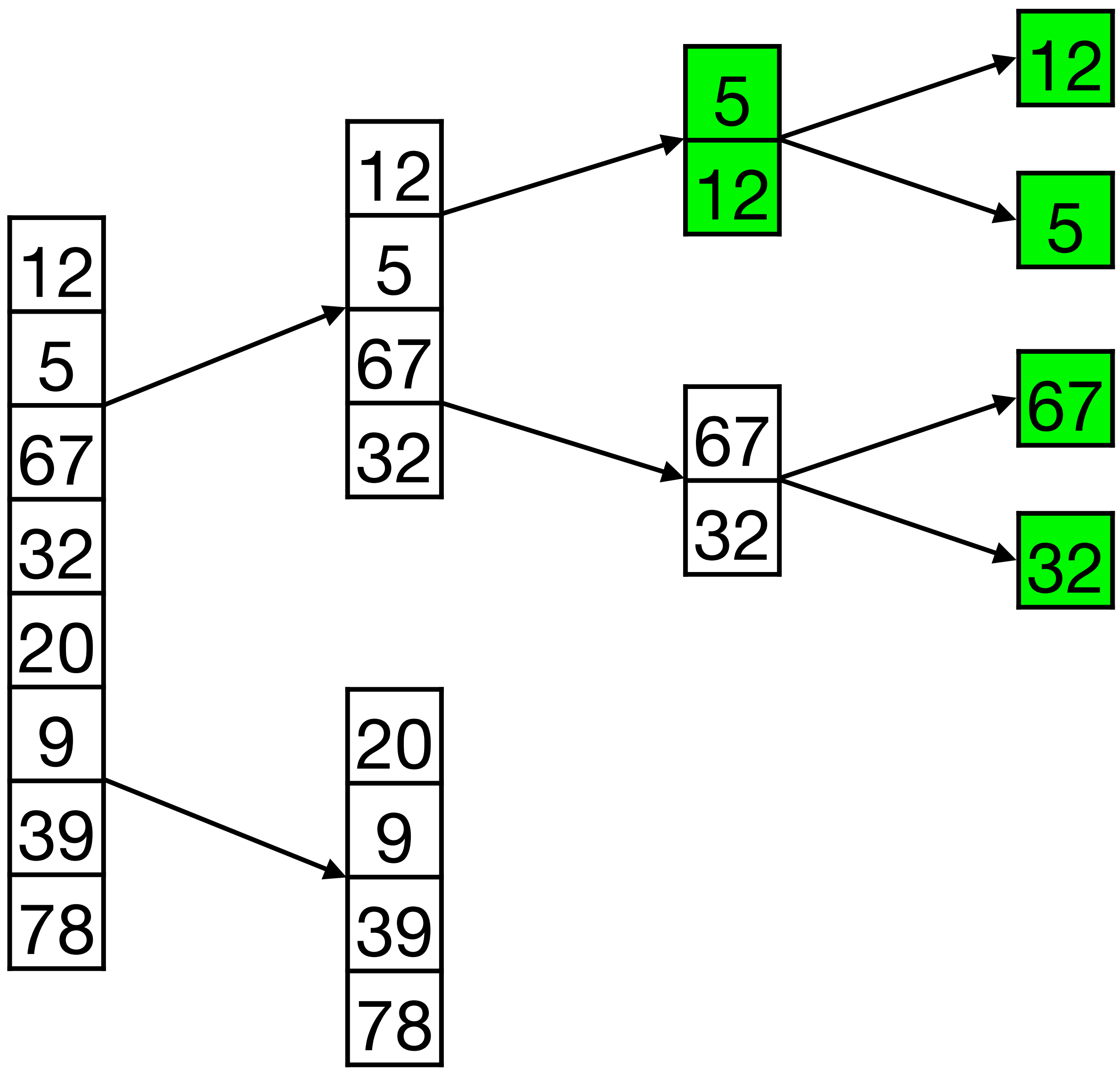
An Example of Merge Sort



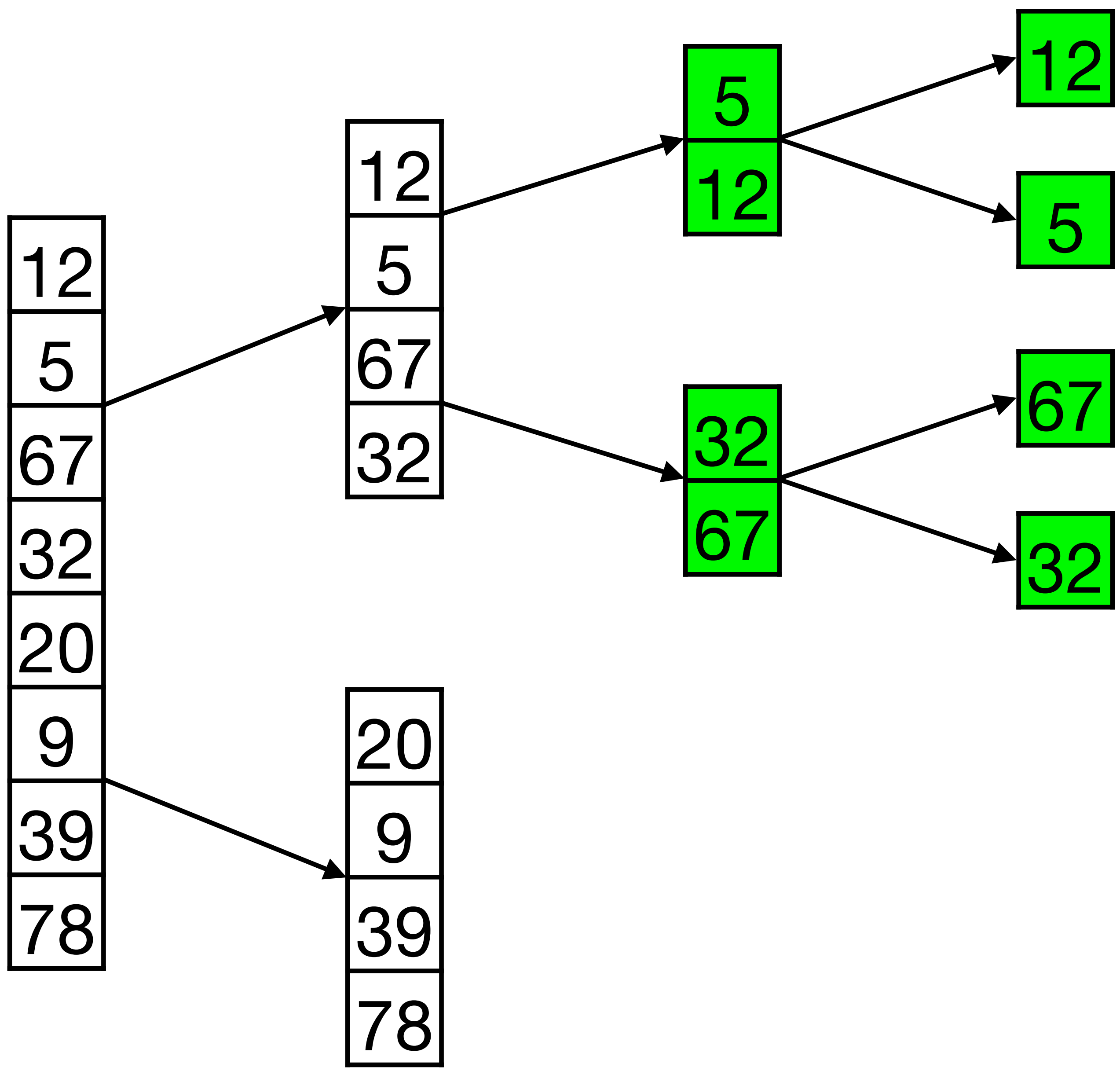
An Example of Merge Sort



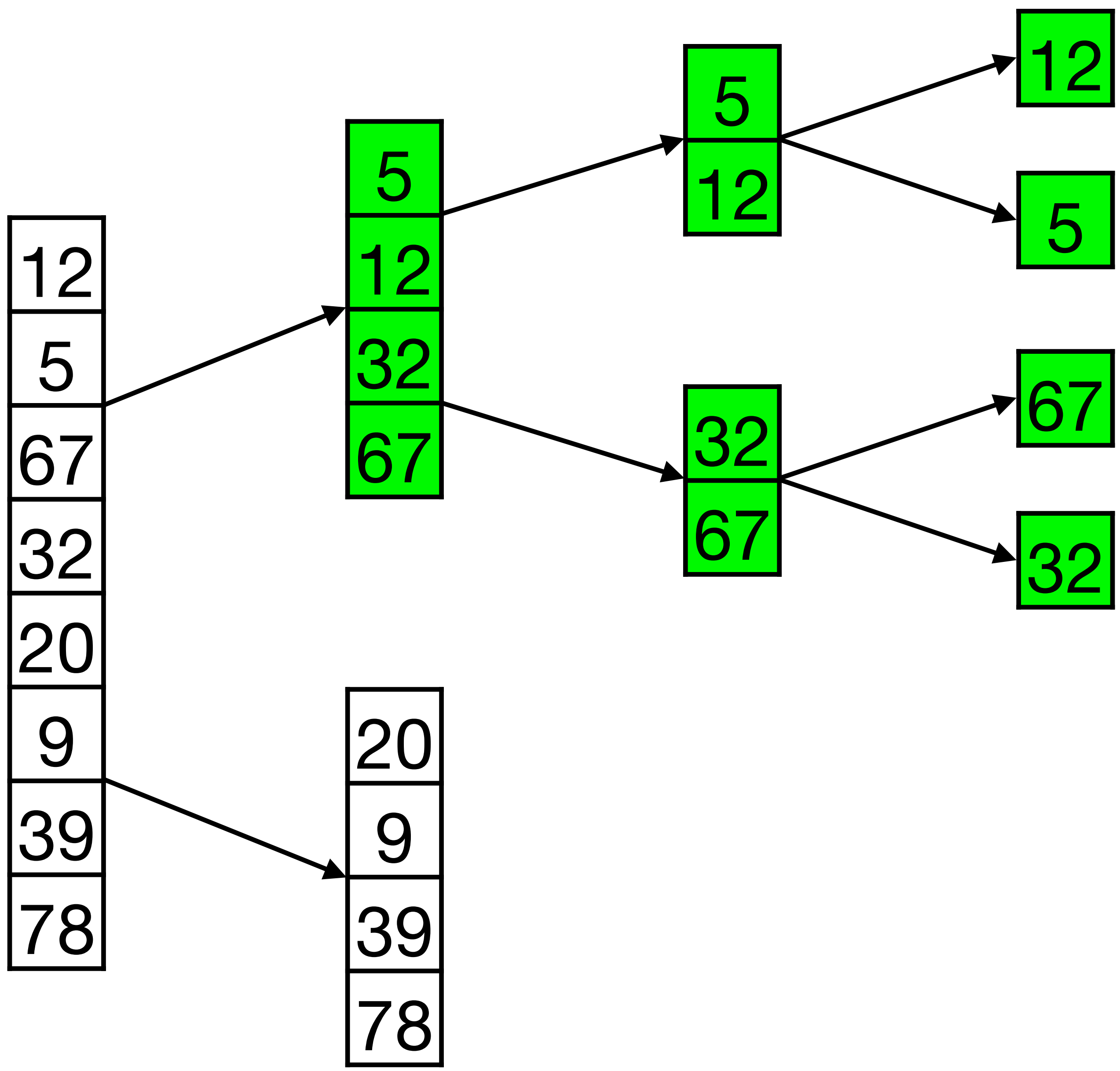
An Example of Merge Sort



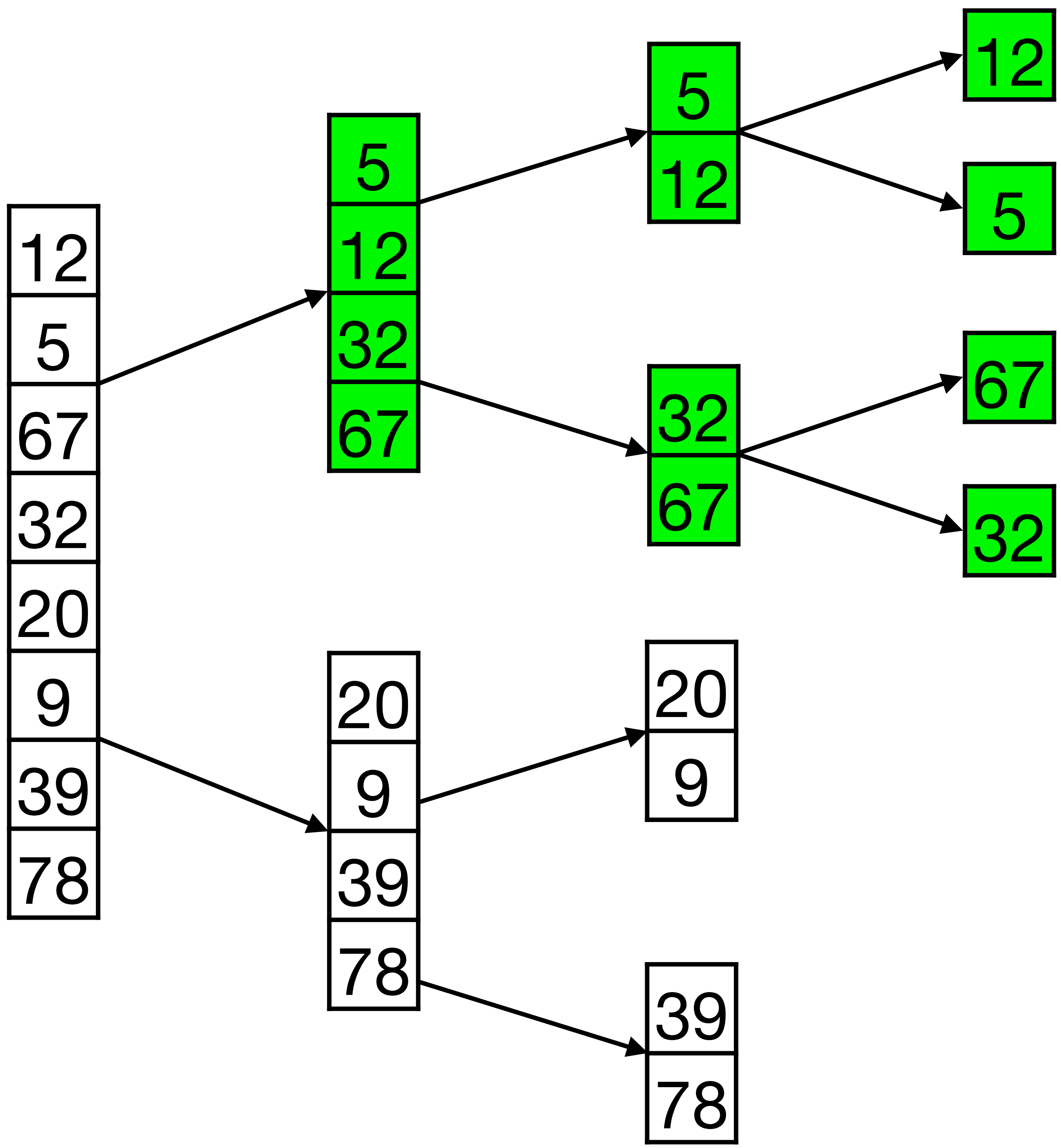
An Example of Merge Sort



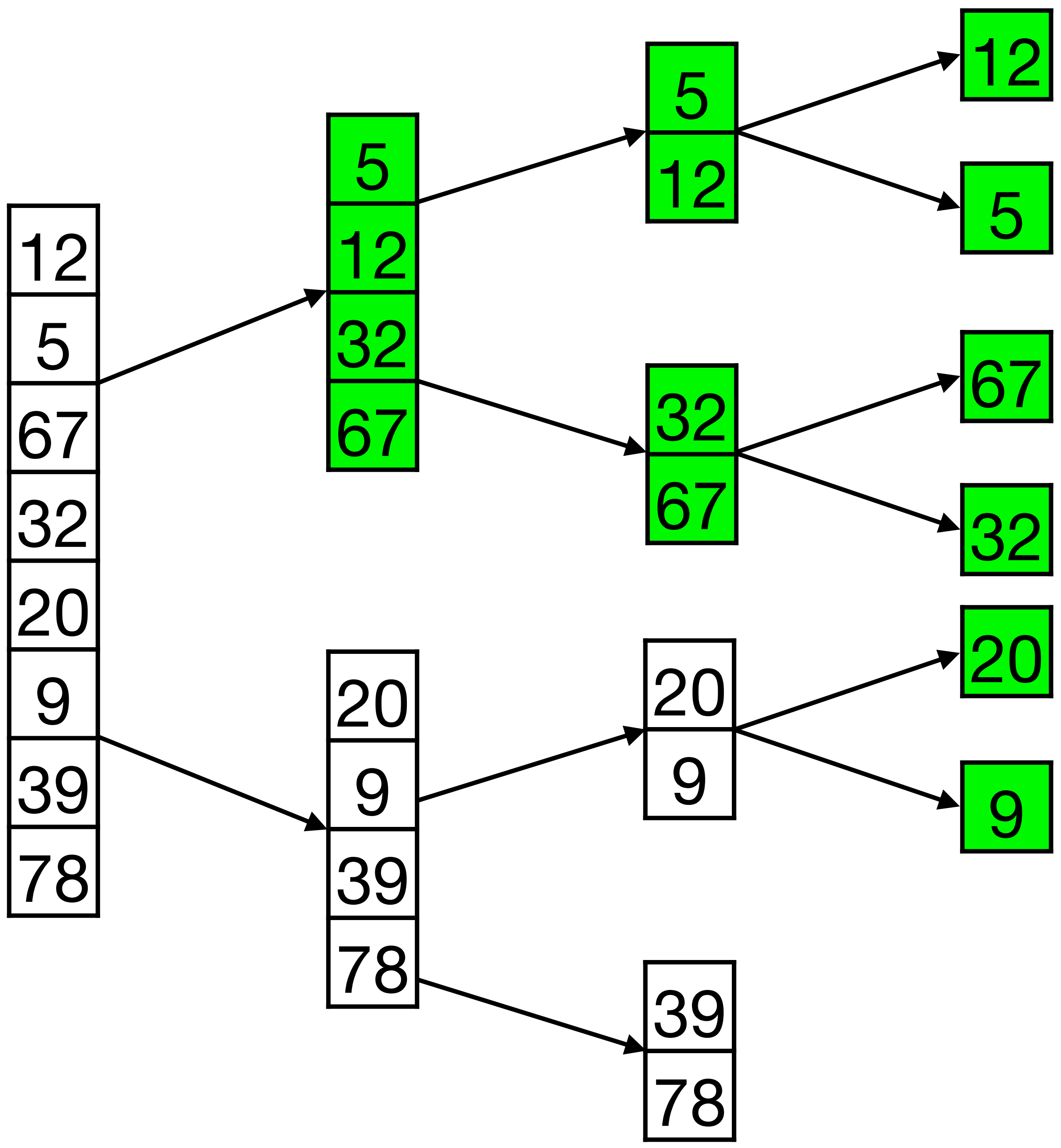
An Example of Merge Sort



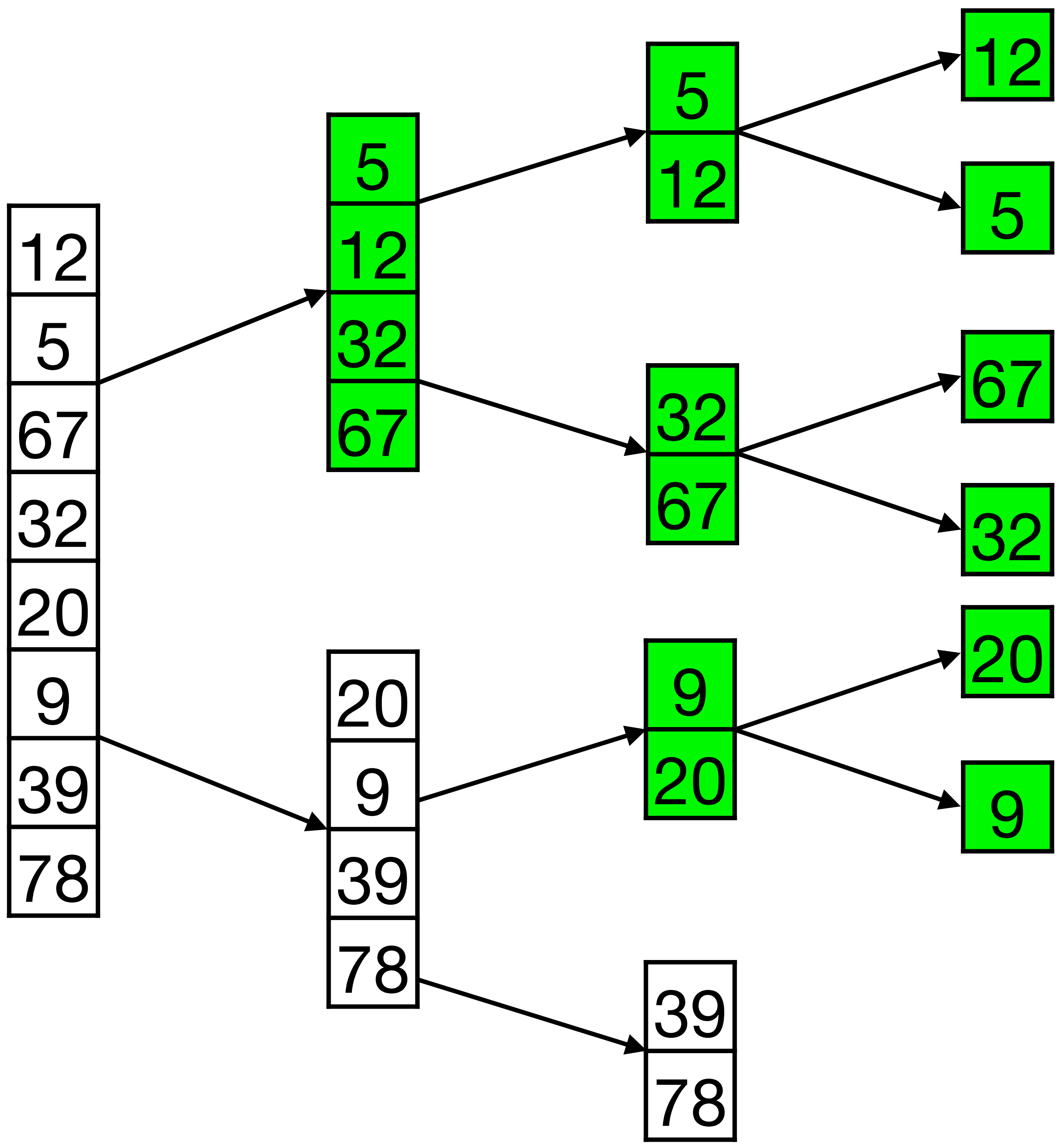
An Example of Merge Sort



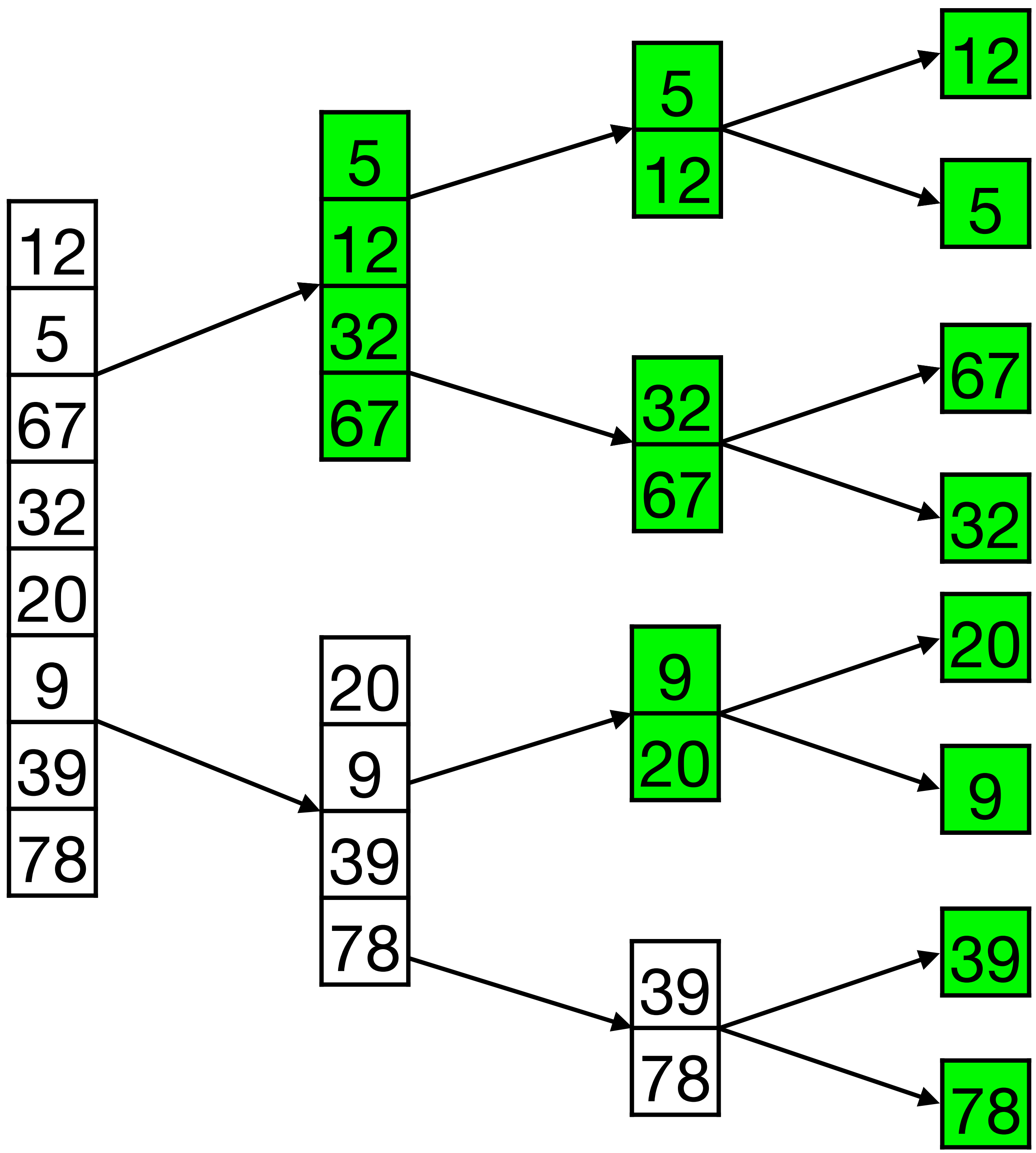
An Example of Merge Sort



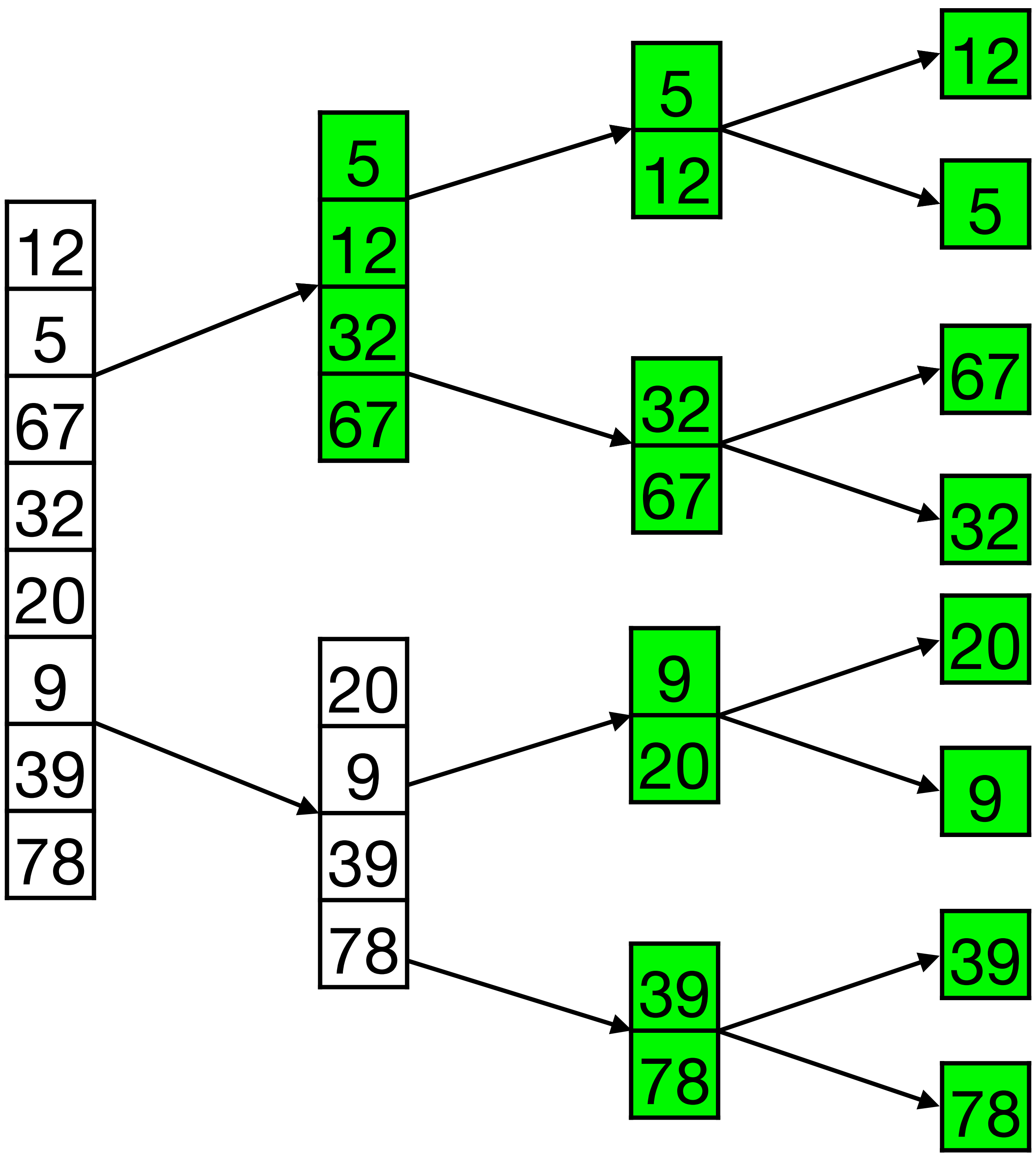
An Example of Merge Sort



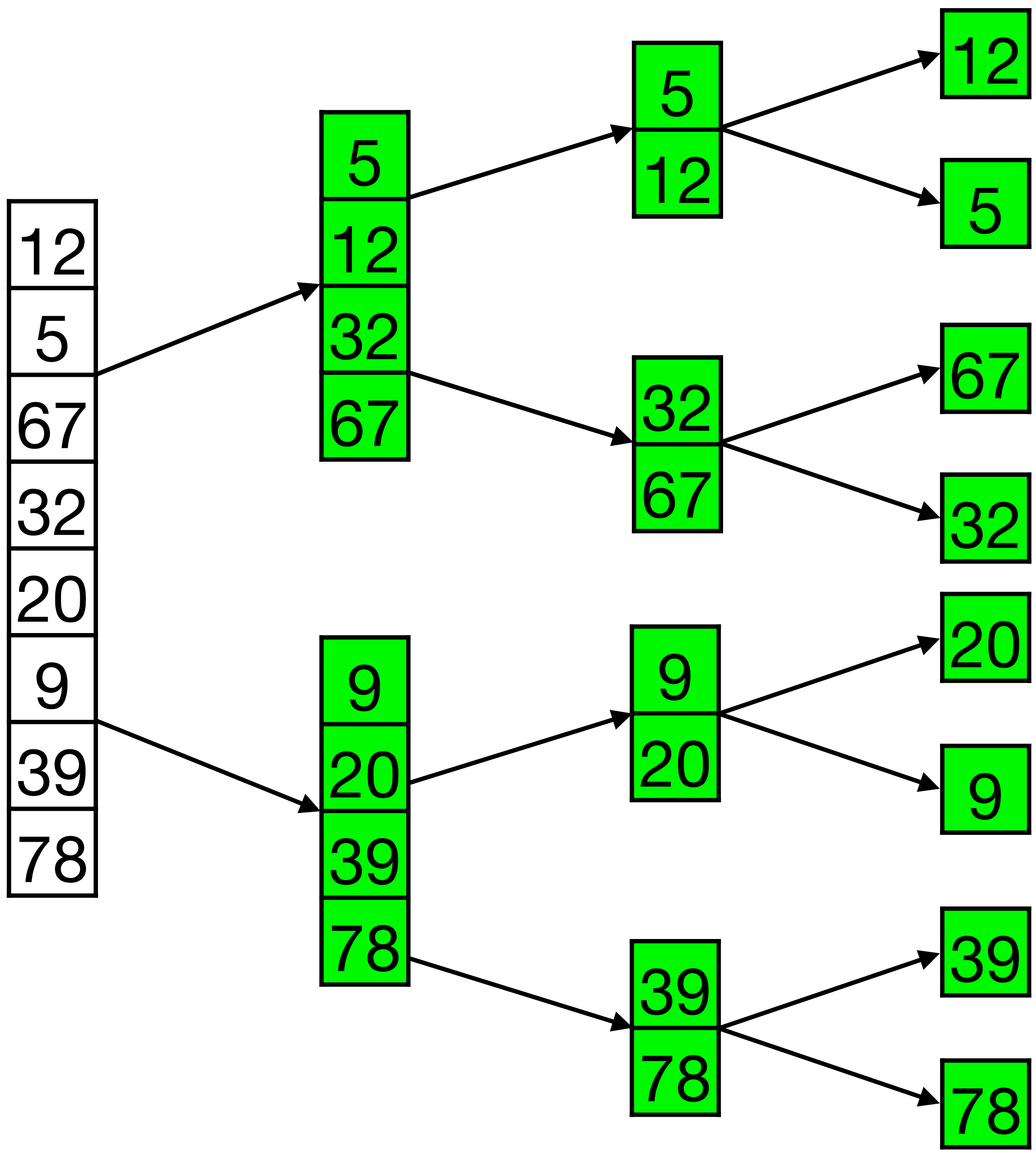
An Example of Merge Sort



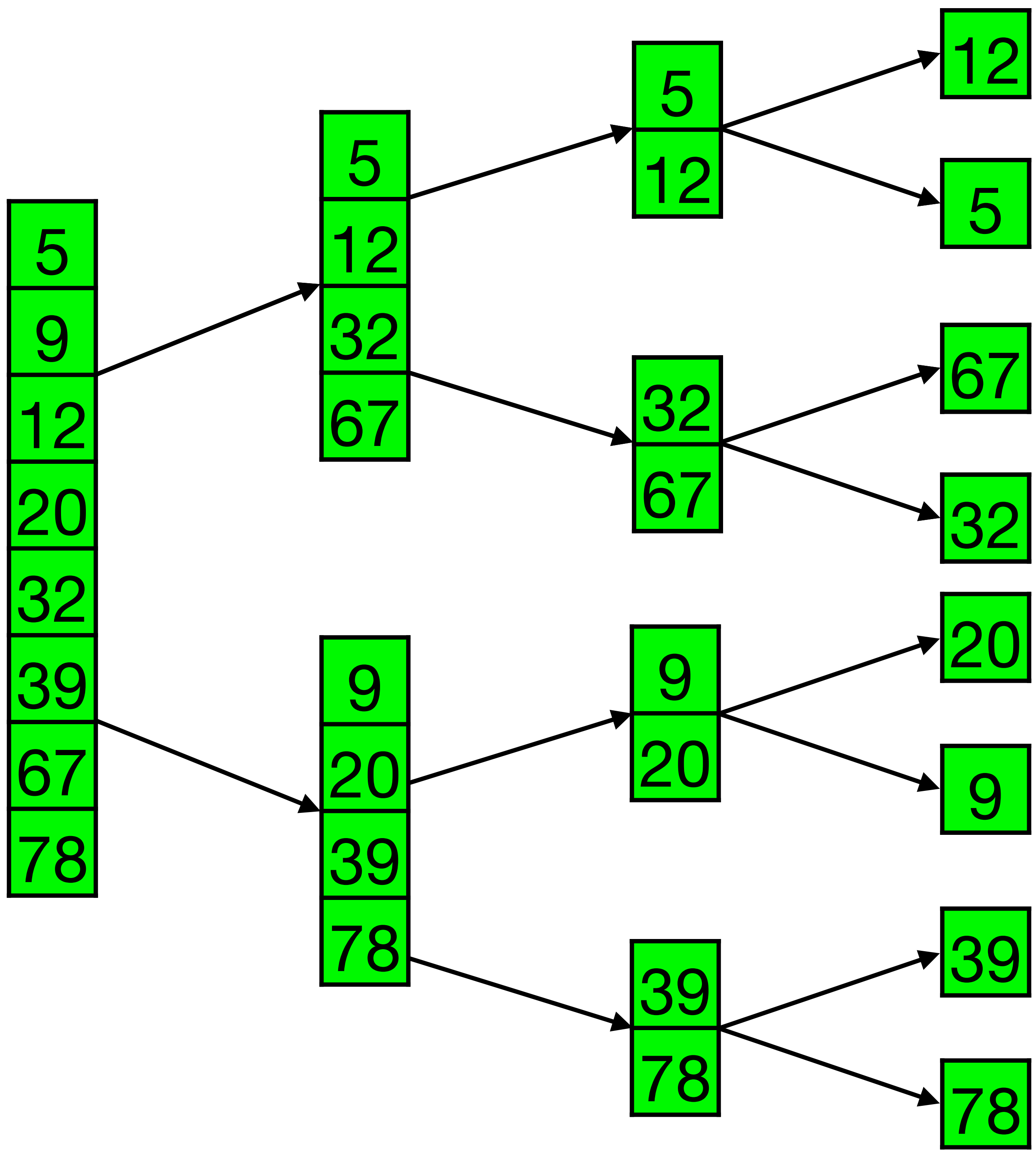
An Example of Merge Sort



An Example of Merge Sort



An Example of Merge Sort



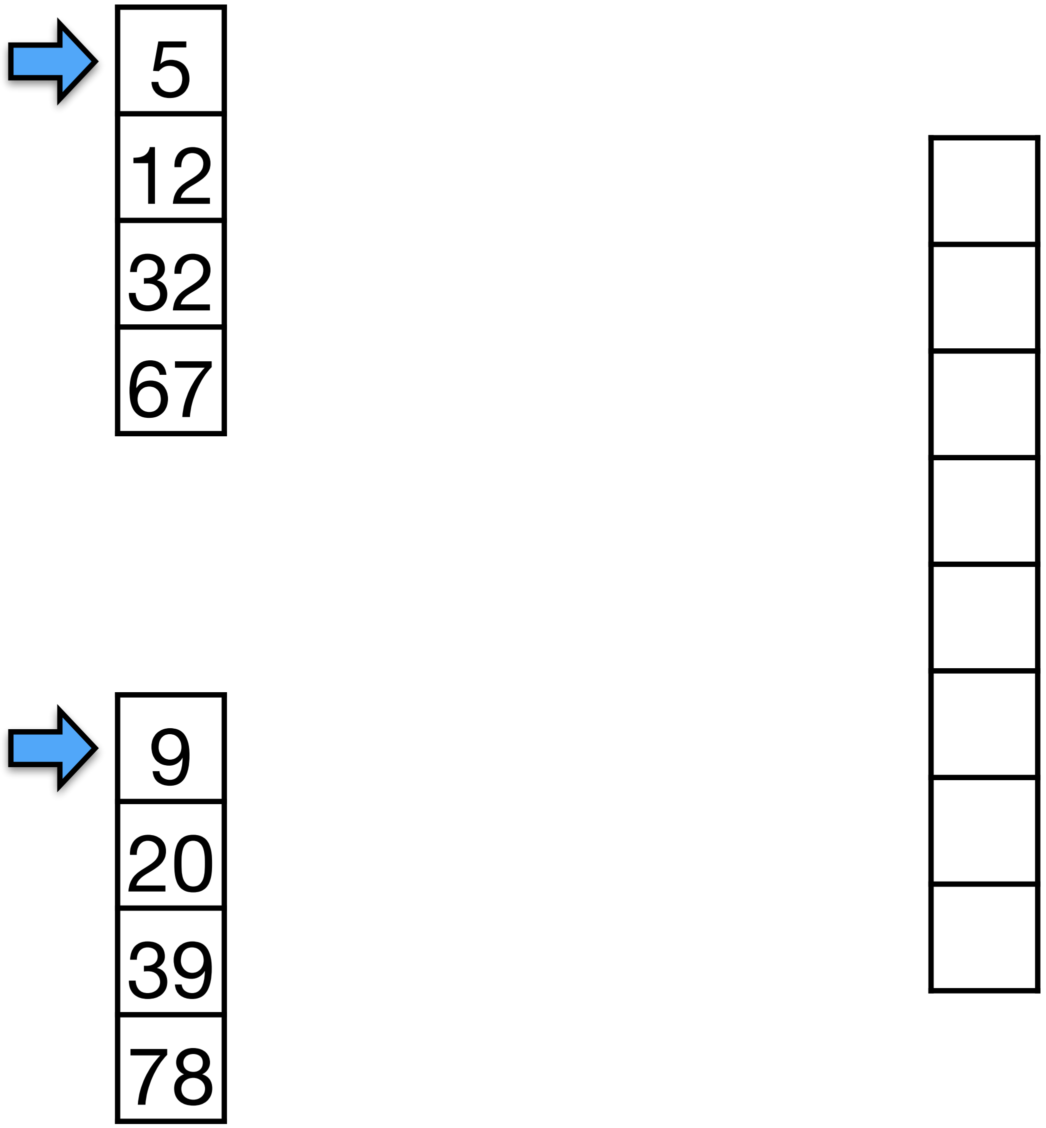
Rest of this lecture:

1) How to merge two sorted lists of integers?
2) Solving $T(n)$.

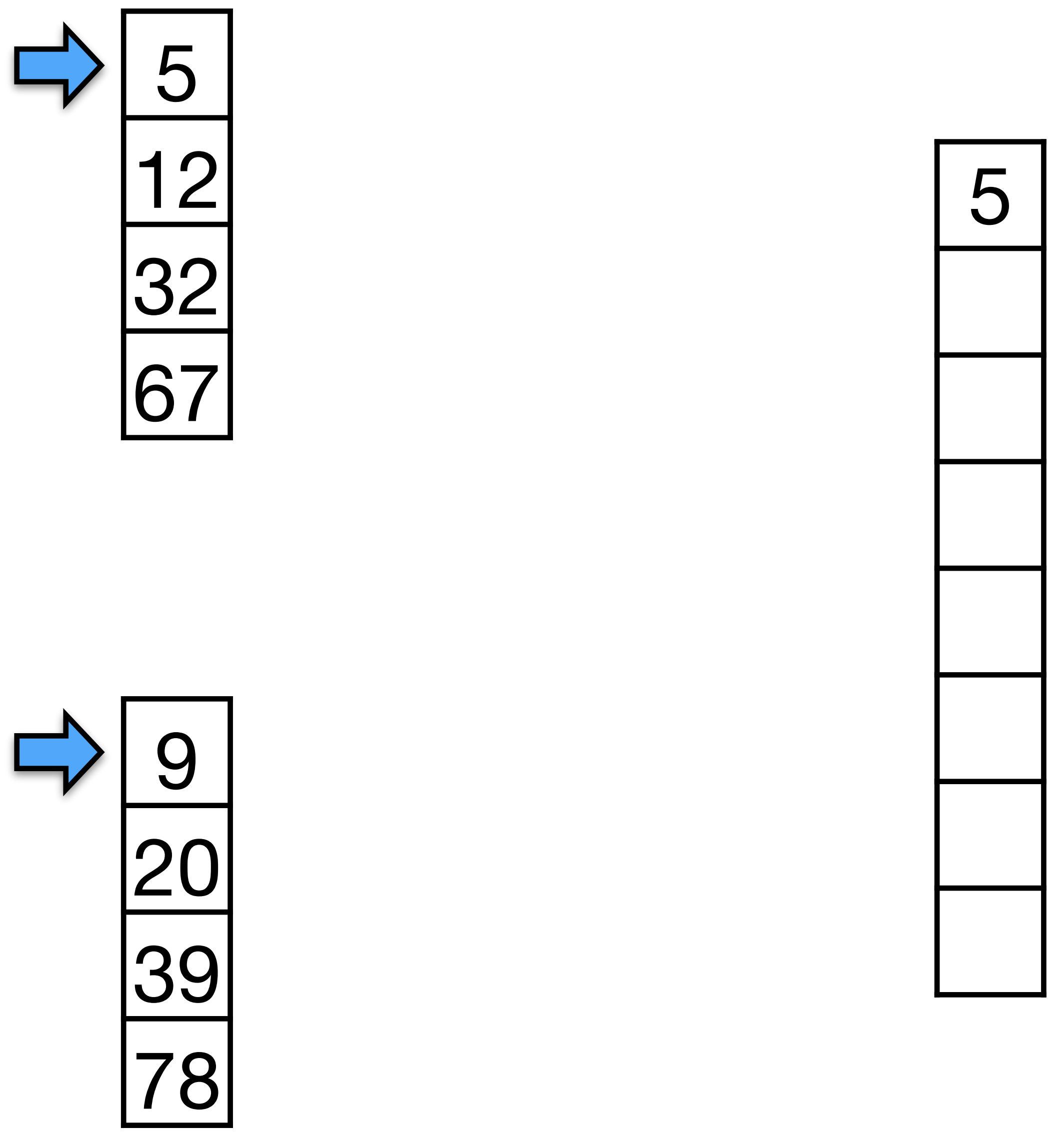
Merging Two Sorted Lists of Numbers

```
To merge sorted lists A = a_1,...,a_n and B = b_1,...,b_n:
  Maintain a Current pointer into each list, initialized to
    point to the front elements
  While both lists are nonempty:
    Let a_i and b_j be the elements pointed to by the Current pointer
    Append the smaller of these two to the output list
    Advance the Current pointer in the list from which the
      smaller element was selected
  EndWhile
  Once one list is empty, append the remainder of the other list
      to the output
```

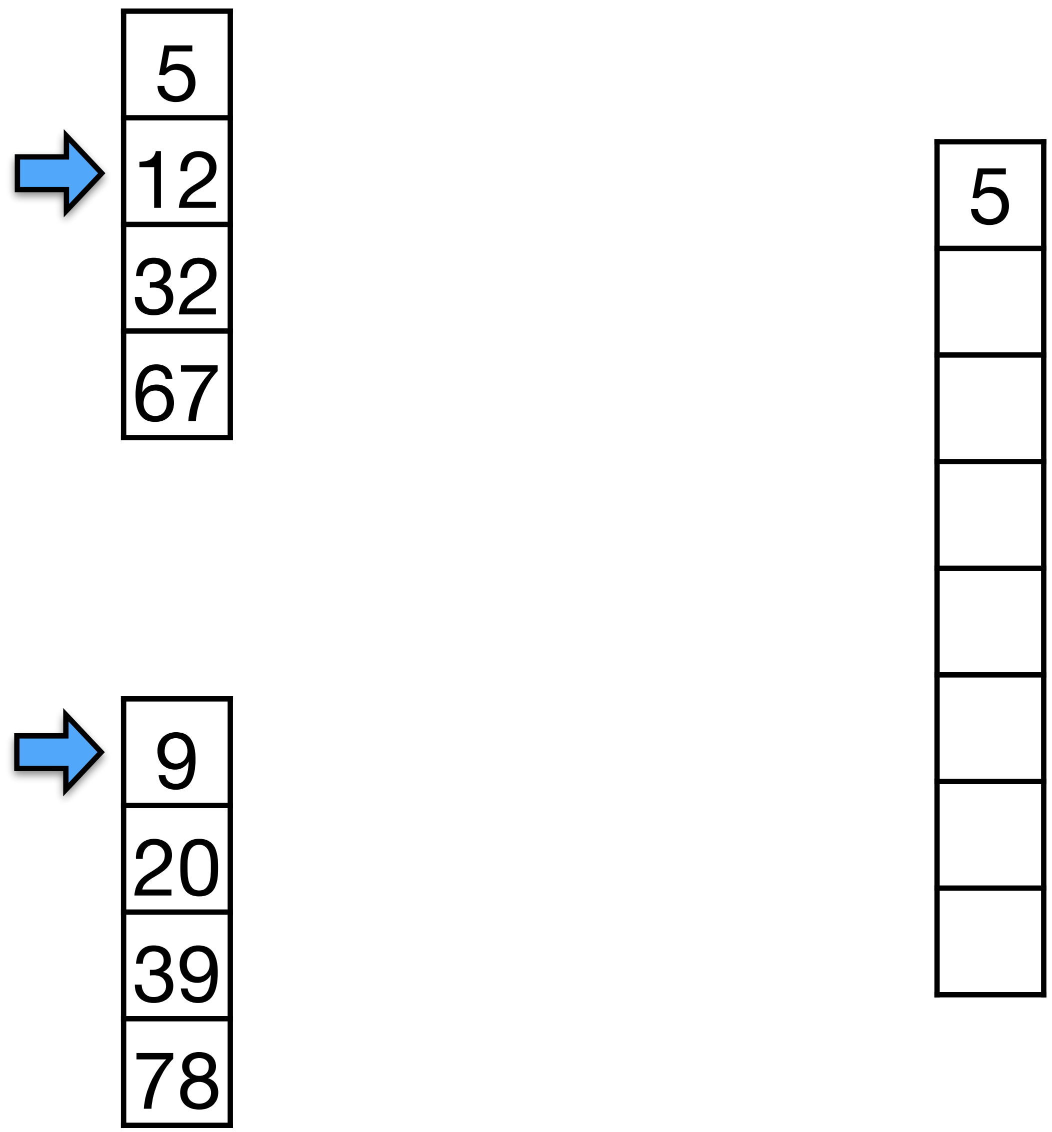
An Example of Merging Two Sorted Lists



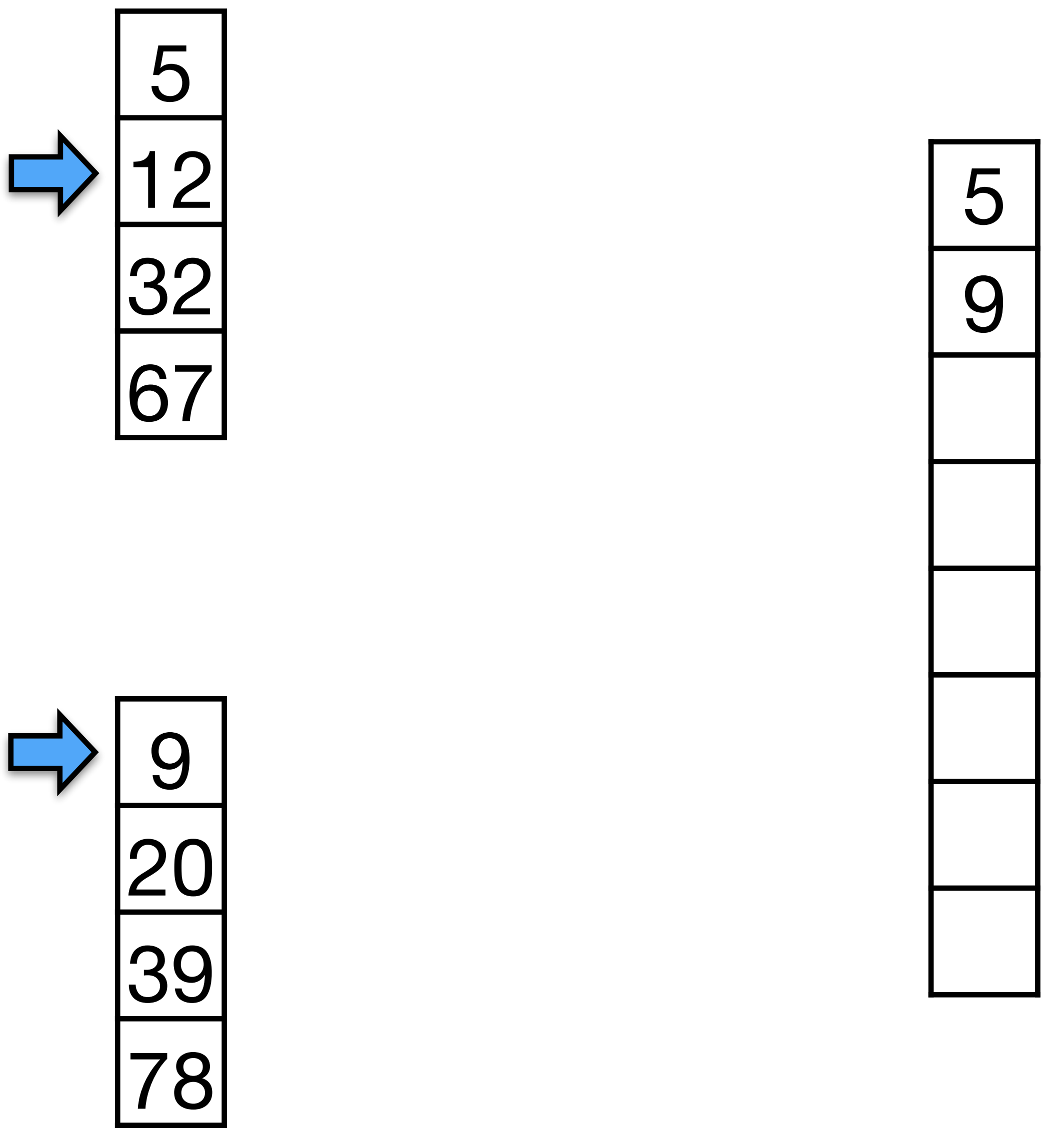
An Example of Merging Two Sorted Lists



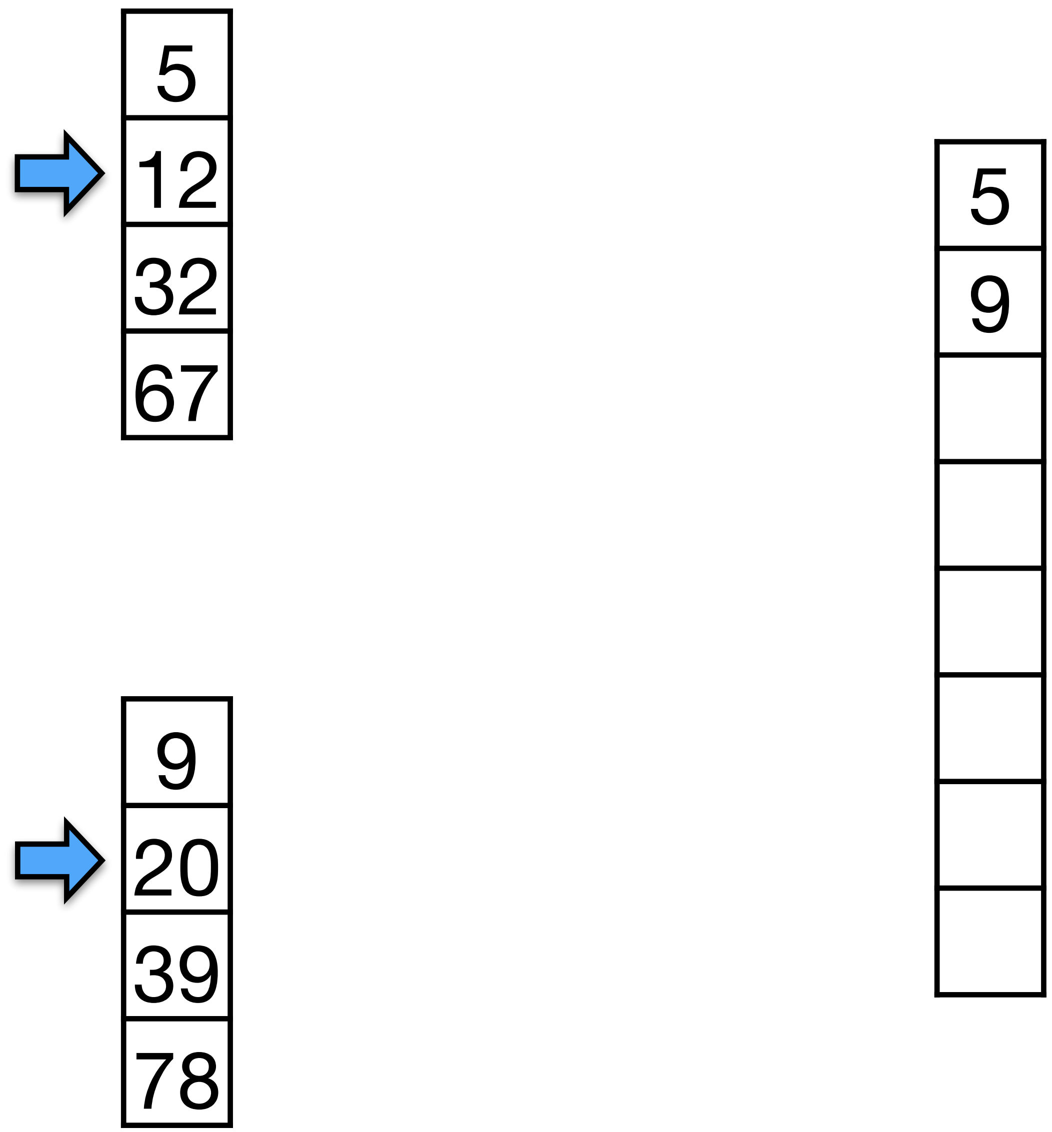
An Example of Merging Two Sorted Lists



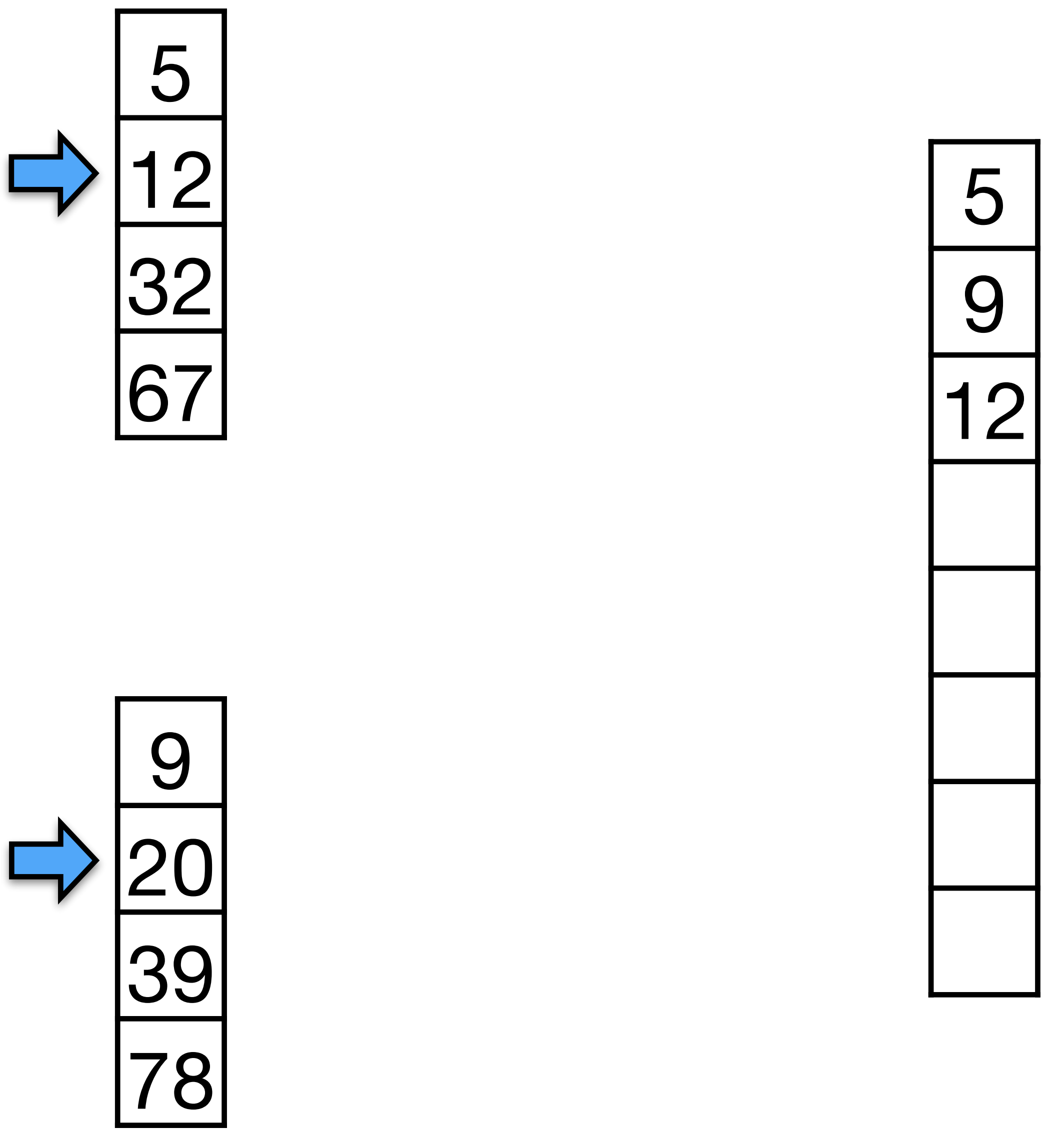
An Example of Merging Two Sorted Lists



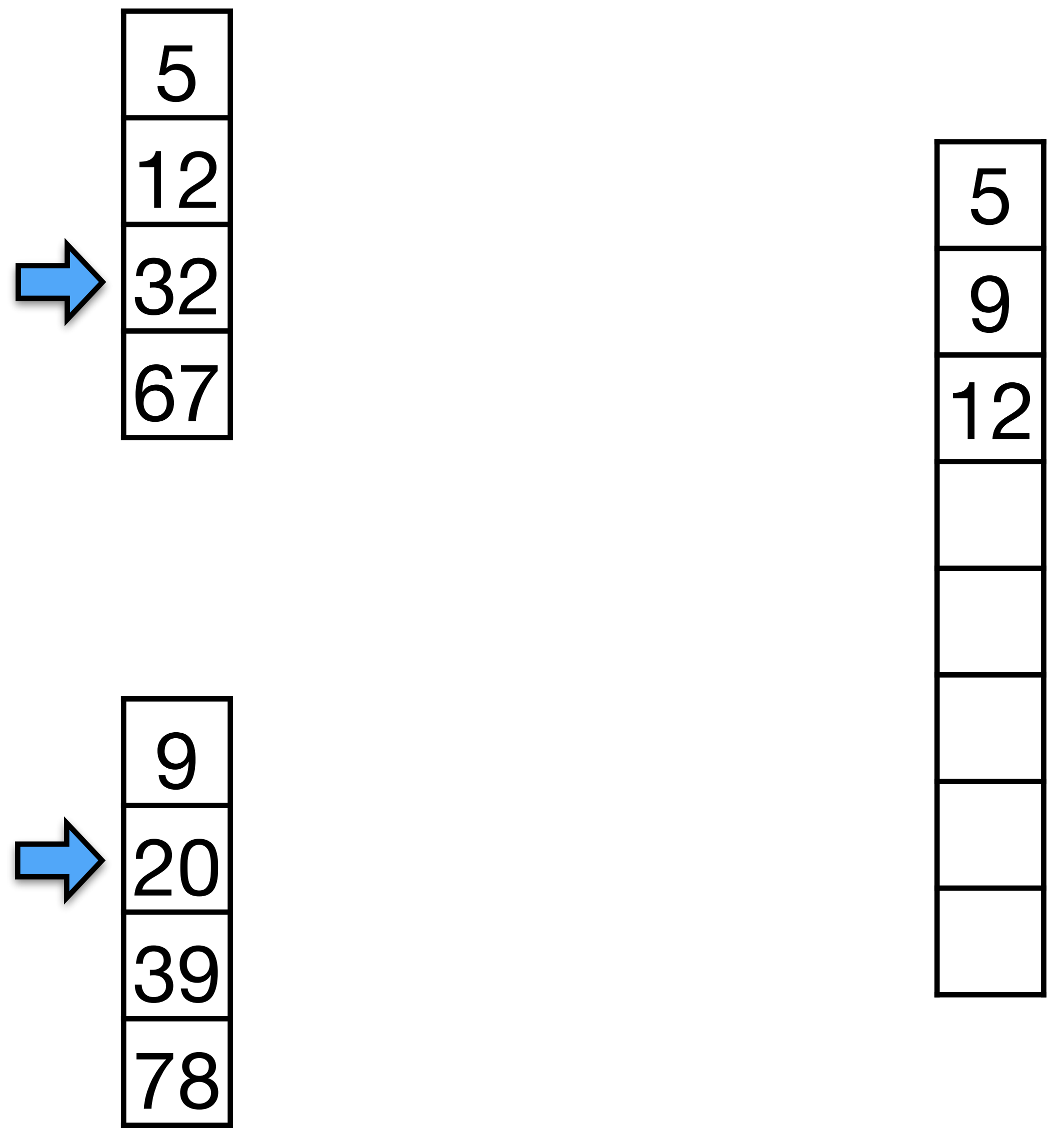
An Example of Merging Two Sorted Lists



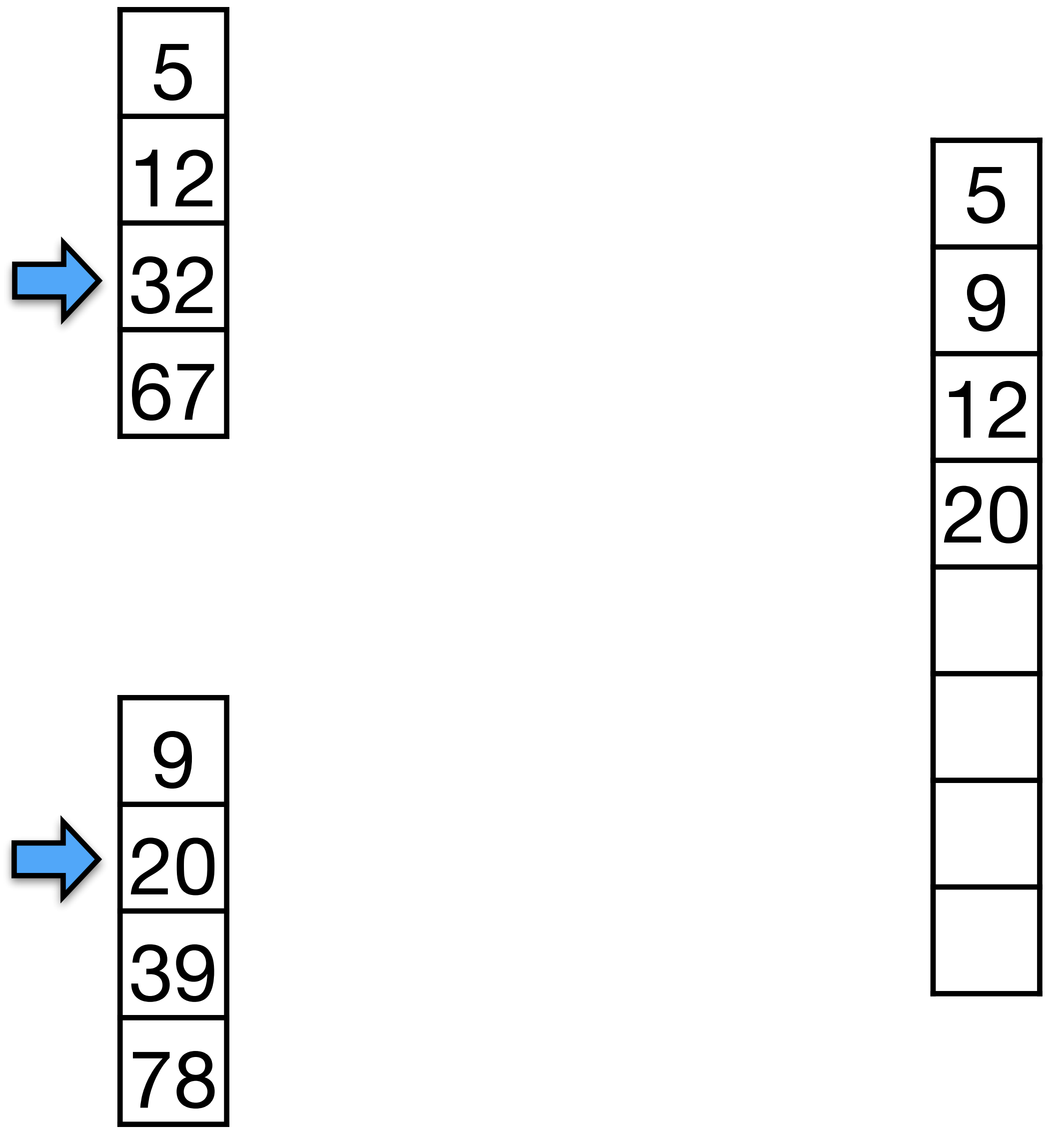
An Example of Merging Two Sorted Lists



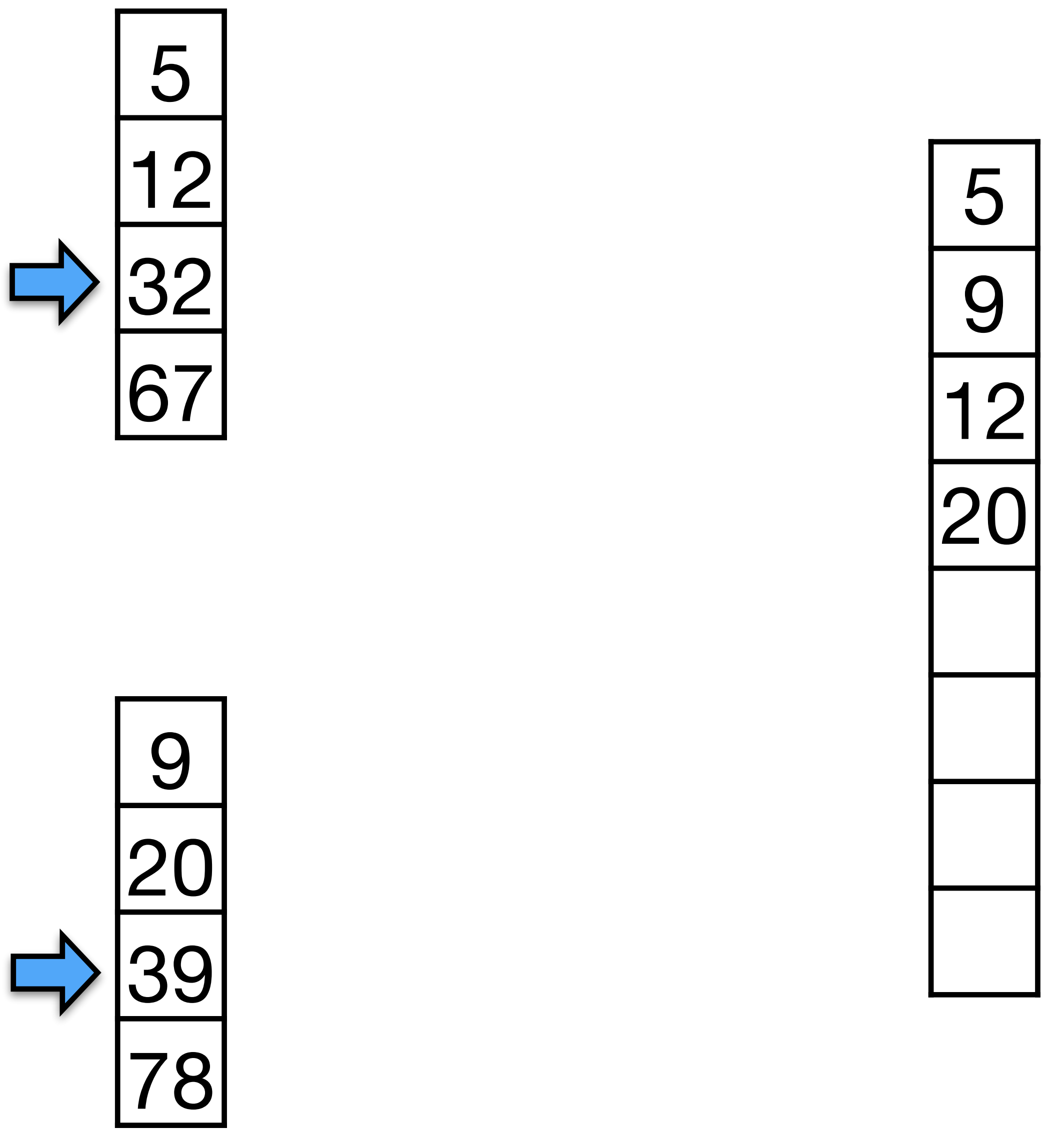
An Example of Merging Two Sorted Lists



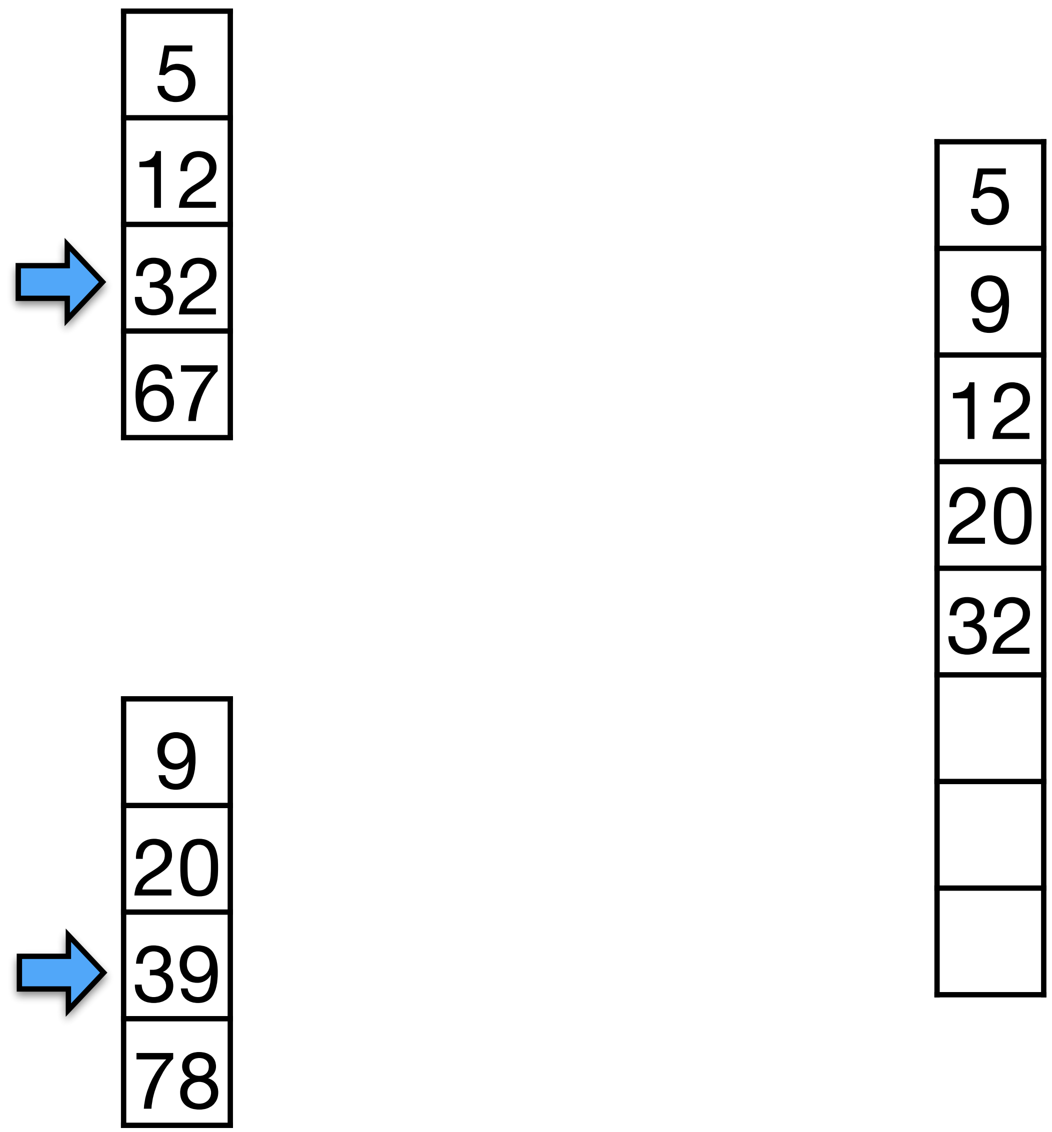
An Example of Merging Two Sorted Lists



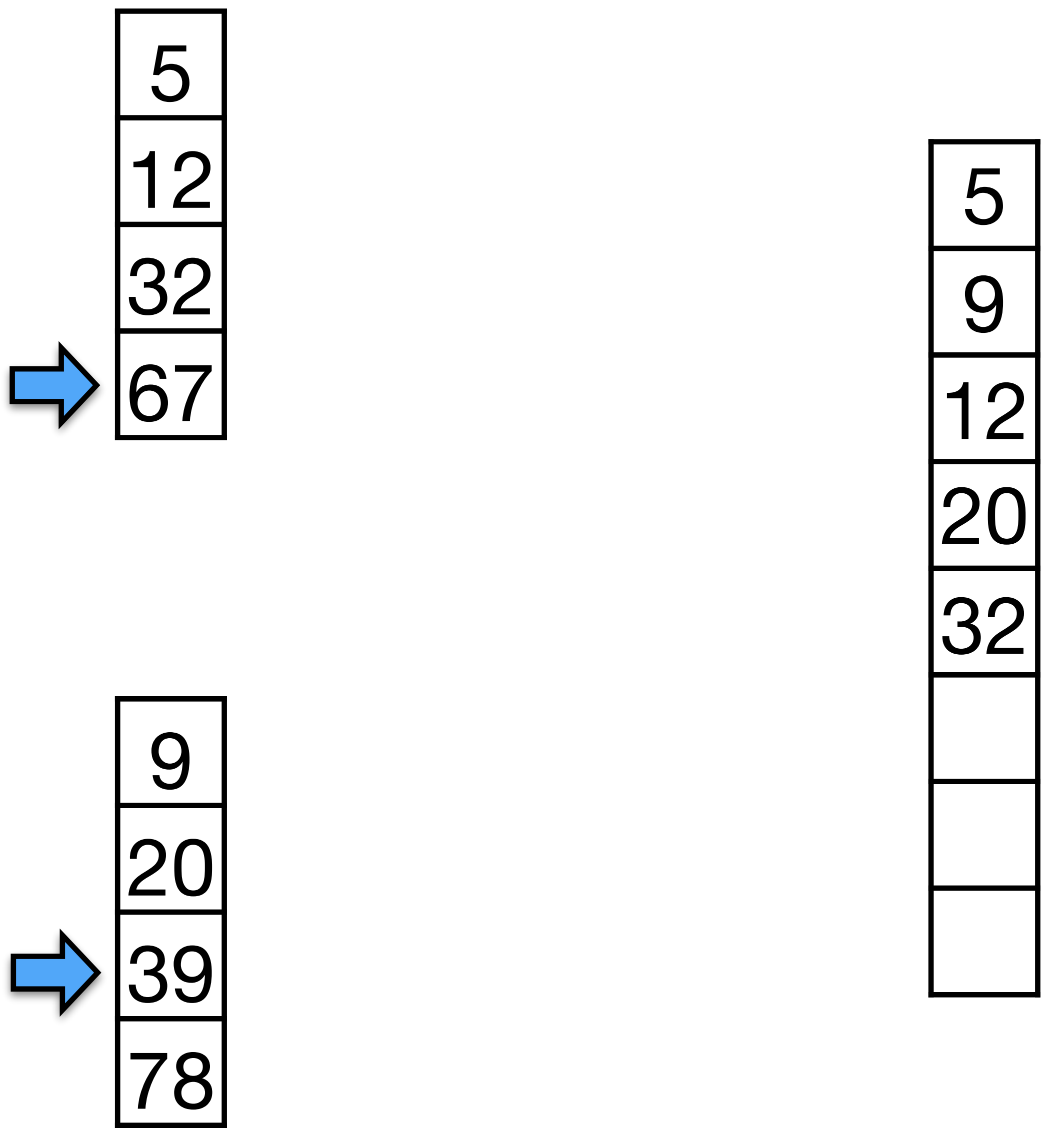
An Example of Merging Two Sorted Lists



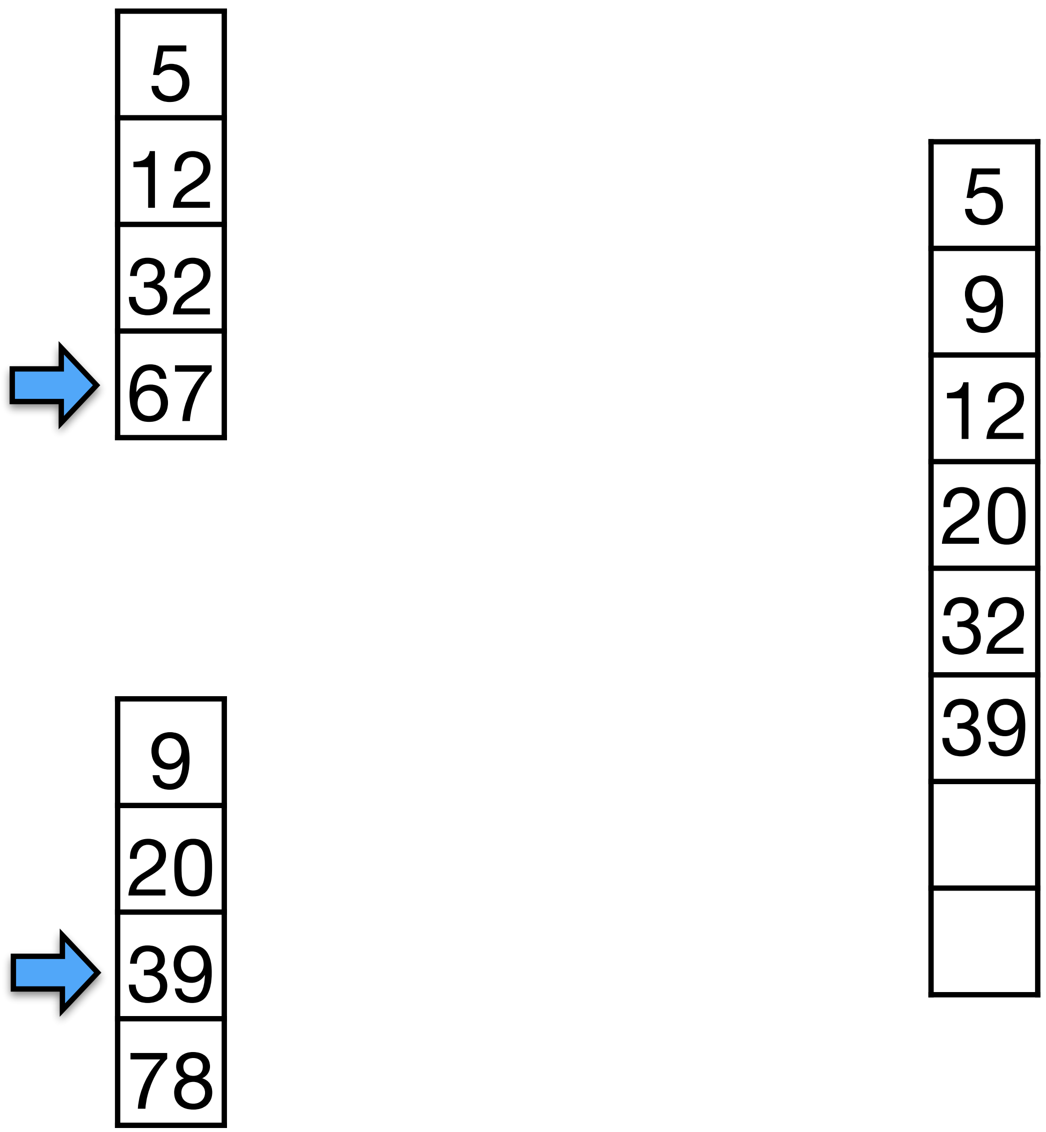
An Example of Merging Two Sorted Lists



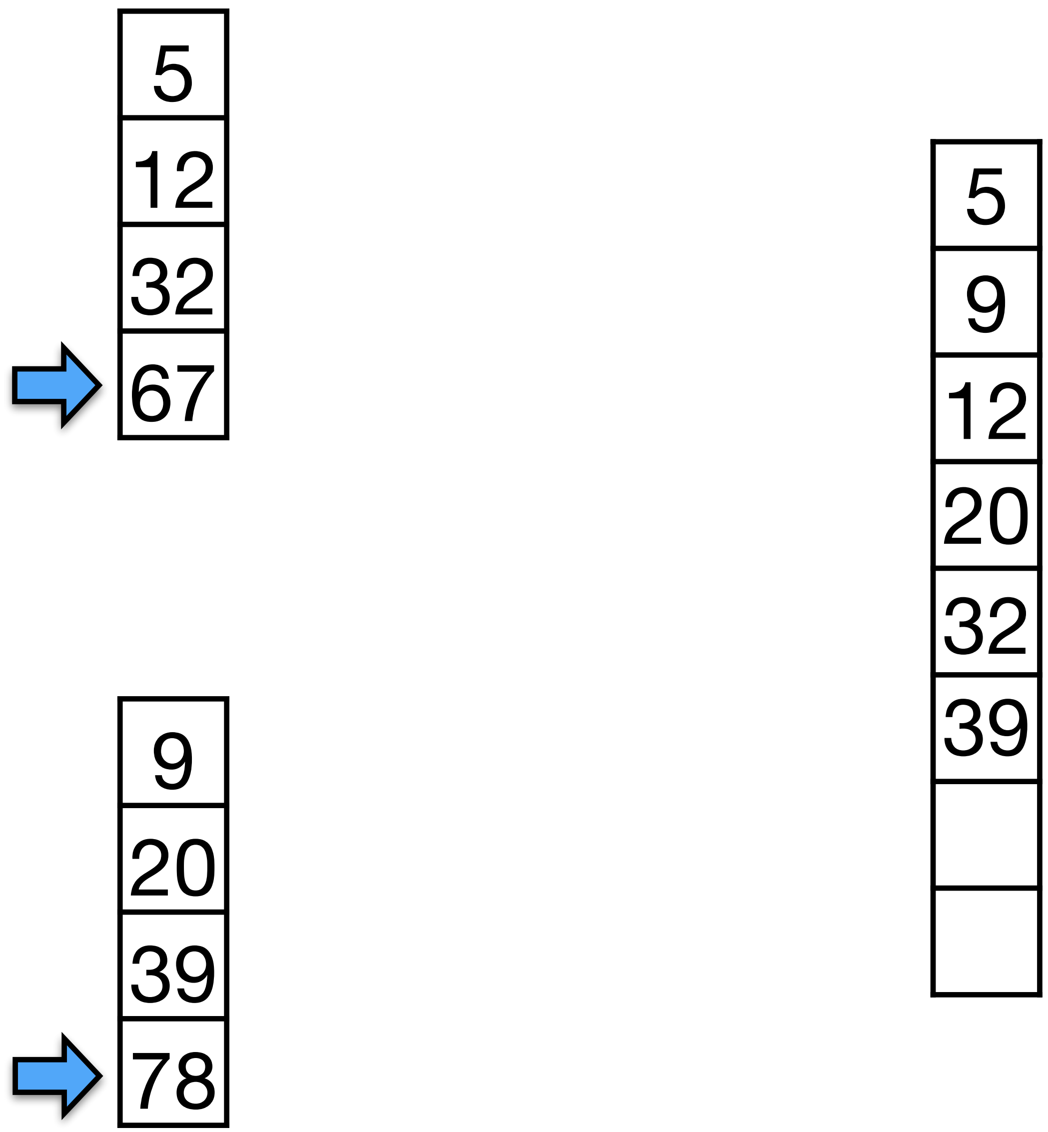
An Example of Merging Two Sorted Lists



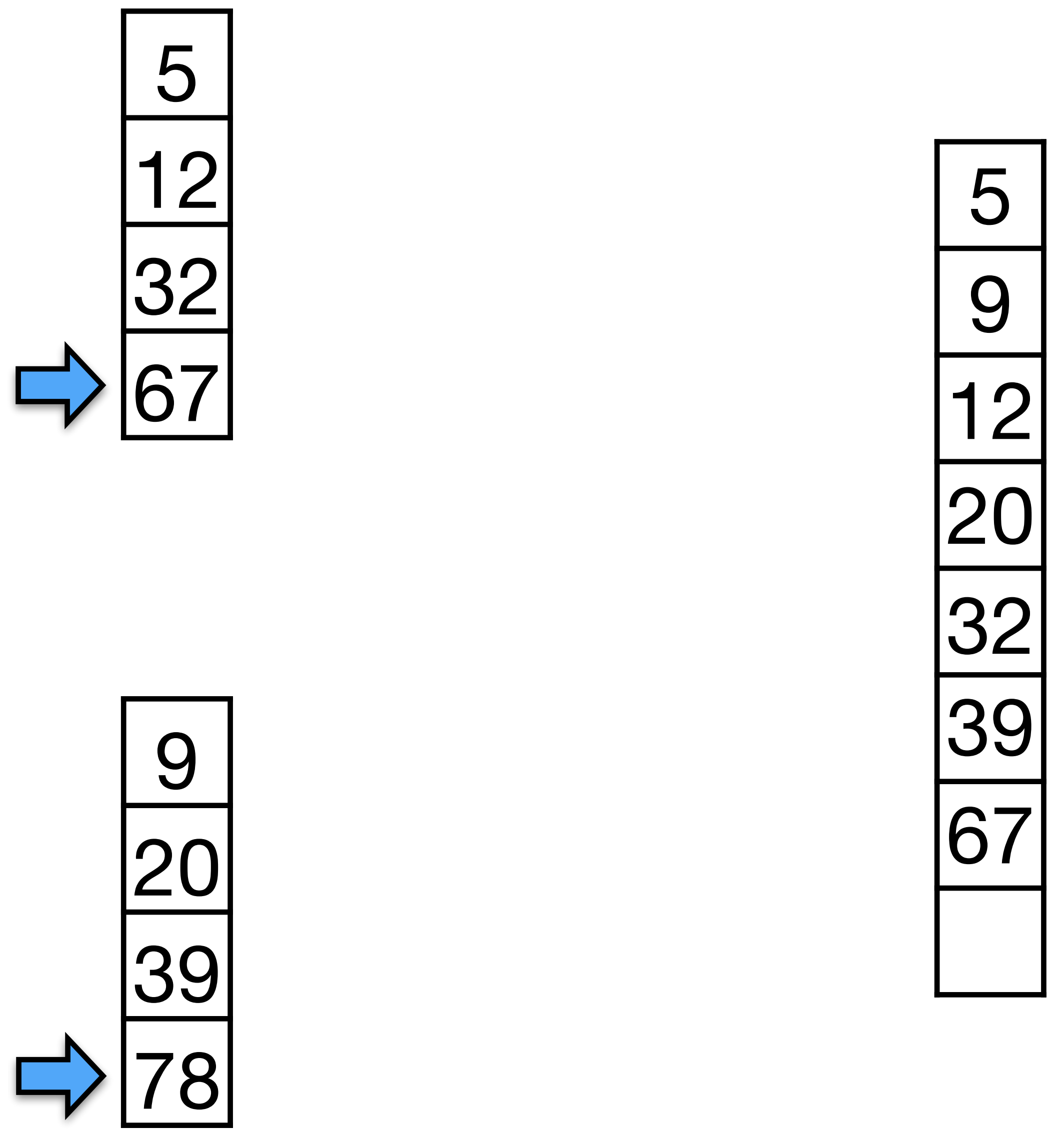
An Example of Merging Two Sorted Lists



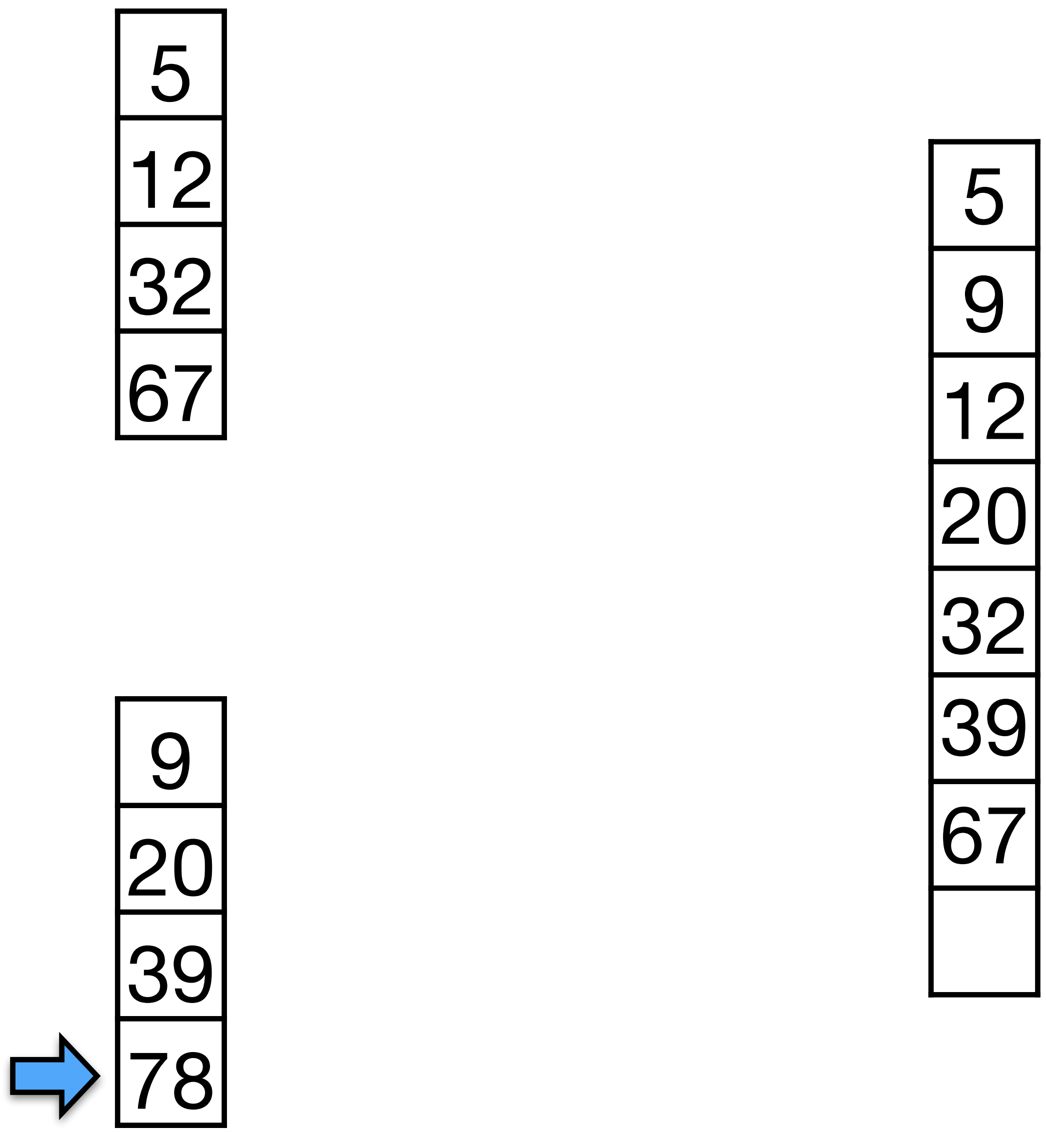
An Example of Merging Two Sorted Lists



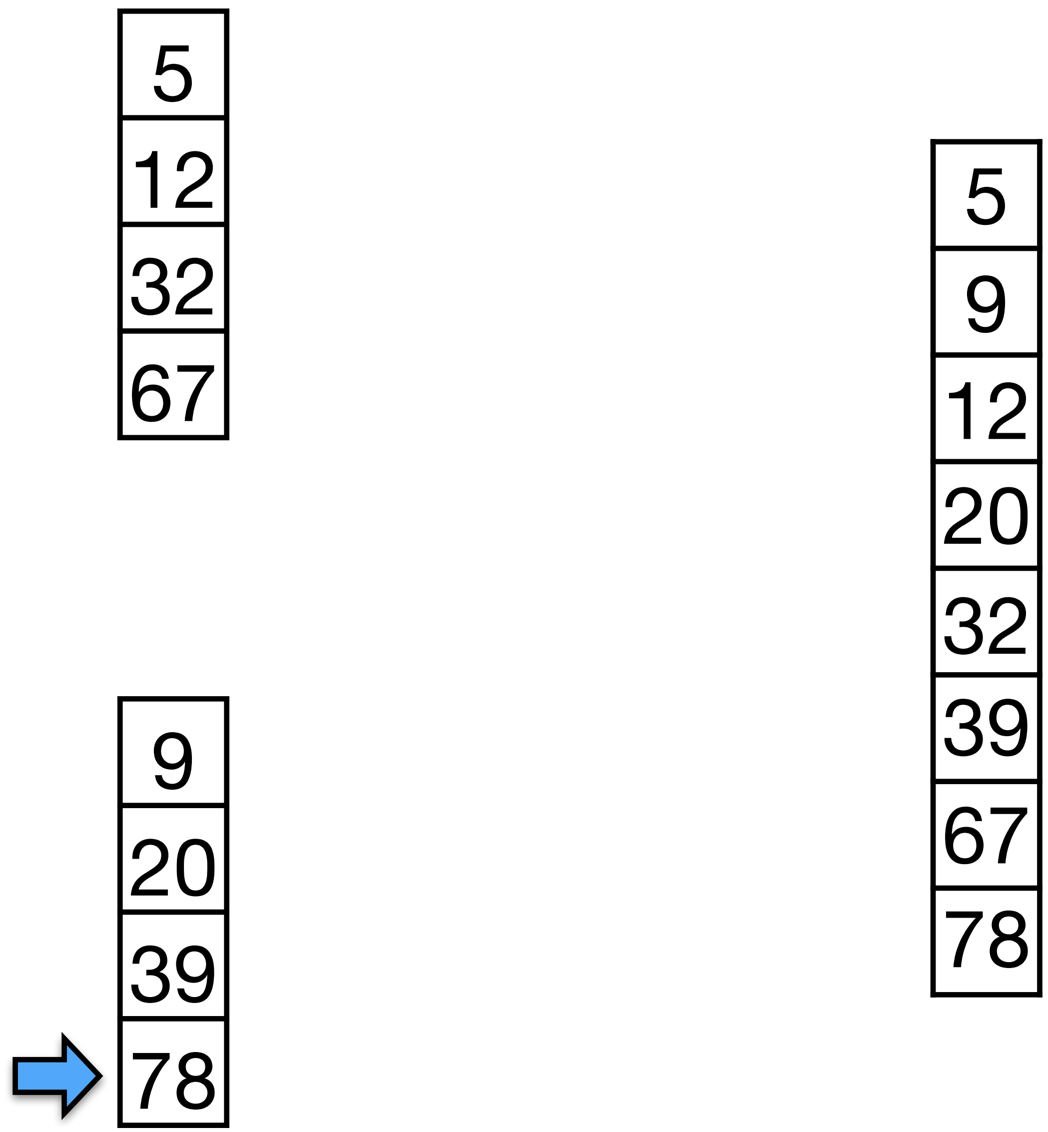
An Example of Merging Two Sorted Lists



An Example of Merging Two Sorted Lists



An Example of Merging Two Sorted Lists



What is the running time?

- To analyze the running time, we measure progress by the number of integers that have been put into the output list.

What is the running time?

- To analyze the running time, we measure progress by the number of integers that have been put into the output list.
  - After a comparison, the algorithm will put an integer into the output list.

What is the running time?

- To analyze the running time, we measure progress by the number of integers that have been put into the output list.
  - After a comparison, the algorithm will put an integer into the output list.
  - Initially, there is no integer in the output list.

What is the running time?

- To analyze the running time, we measure progress by the number of integers that have been put into the output list.
  - After a comparison, the algorithm will put an integer into the output list.
  - Initially, there is no integer in the output list.
  - To merge two sorted lists of length m and n, the final output list has $m+n$ integers.

What is the running time?

- To analyze the running time, we measure progress by the number of integers that have been put into the output list.
  - After a comparison, the algorithm will put an integer into the output list.
  - Initially, there is no integer in the output list.
  - To merge two sorted lists of length m and n, the final output list has $m+n$ integers.
  - The algorithm makes at most $m+n$ comparisons.

What is the running time?

- To analyze the running time, we measure progress by the number of integers that have been put into the output list.
  - After a comparison, the algorithm will put an integer into the output list.
  - Initially, there is no integer in the output list.
  - To merge two sorted lists of length m and n, the final output list has $m+n$ integers.
  - The algorithm makes at most $m+n$ comparisons.
- Note that the size of the input is $m+n$, and the merging algorithm runs in $O(m+n)$ time. So it is a linear-time algorithm.

What is the running time?

- To analyze the running time, we measure progress by the number of integers that have been put into the output list.
  - After a comparison, the algorithm will put an integer into the output list.
  - Initially, there is no integer in the output list.
  - To merge two sorted lists of length m and n, the final output list has $m+n$ integers.
  - The algorithm makes at most $m+n$ comparisons.
- Note that the size of the input is $m+n$, and the merging algorithm runs in $O(m+n)$ time. So it is a linear-time algorithm.
- Thus, we have $T(n) = 2T(n/2) + O(n)$

Solving the Recursion

- Suppose $T(n) = 2T(n/2) + O(n) \leq 2T(n/2) + cn$
- Keep expanding the RHS until it is a function of *n* and *T(1)*:

$$
\begin{aligned}
T(n) &\leq 2T(n/2) + cn \\
&\leq 2\big(2T(n/4) + c(n/2)\big) + cn \\
&= 4T(n/4) + 2cn \\
&\leq 4\big(2T(n/8) + c(n/4)\big) + 2cn \\
&= 8T(n/8) + 3cn \\
&\qquad \cdots \\
&\leq 2^i T(n/2^i) + icn \\
&\qquad \cdots \\
&= 2^{\log n} T(1) + cn \log n = O(n \log n)
\end{aligned}
$$