BITS Pilani

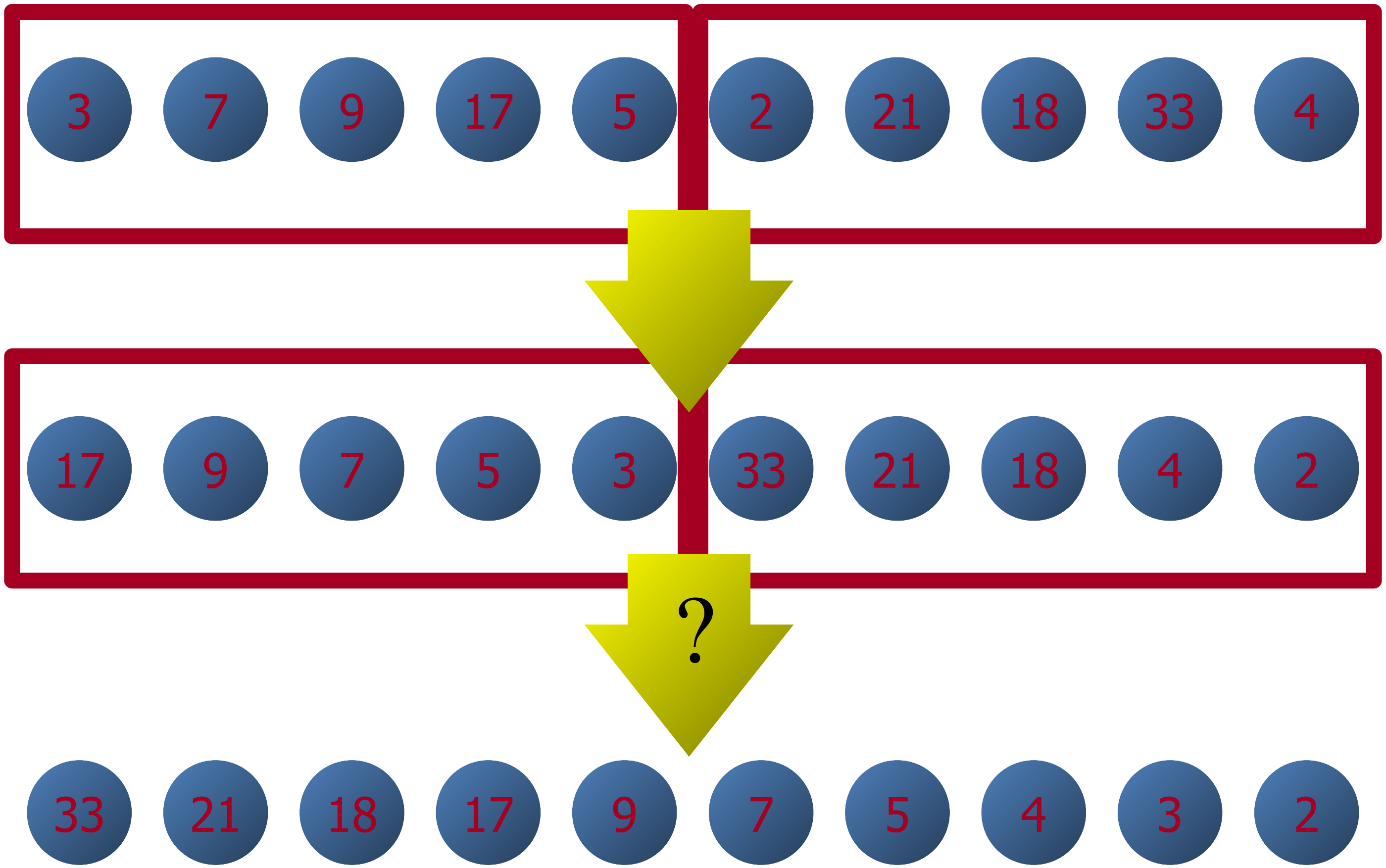
Hyderabad Campus

Data Structures and Algorithms (3)

(CS F211)

Dr.N.L.Bhanu Murthy

Sorting Algorithms



Comparison Based

- ✓ Bubble Sort
- ✓ Quick Sort
- ✓ Insertion Sort
- ✓ Merge Sort
- ✓ Heap Sort

Non-Comp Based

- ✓ Radix Sort
- ✓ Bucket Sort

Lower bound on comparison based algorithms

Sorting Applications

Obvious Applications:

- Sort a list of names.
- Organize an MP3 library.
- Display Google PageRank results.

Problems become easy once items are in sorted order

- Find the median.
- Find the closest pair.
- Binary search in a database.
- Identify statistical outliers.
- Find duplicates in a mailing list.

Non-obvious applications

- Data compression.
- Computer graphics.
- Computational biology.
- Supply chain management.
- Simulate a system of particles.
- Book recommendations on Amazon.
- Load balancing on a parallel computer.

Bubble Sort

Algorithm Bubble-Sort {

    int i, $done$;

    do {

        $done = 1$;

        for ($i = 1$; $i < n$; i++)

            if ($A[i] > A[i + 1]$) {

                exchange $A[i]$ and $A[i + 1]$;

                $done = 0$;

            }

    } while ($done == 0$);

}

Bubble Sort

Complexity is O(1) + f(n) { O(1) + O(n) O(1) + O(1) } = f(n) . O(n)

$$
\begin{array}{ll}
\text{Algorithm Bubble-Sort } \{ & \\
\quad \text{int } i,\ done; & O(1) \text{ time} \\
\quad \text{do } \{ & \\
\quad\quad done = 1; & O(1) \text{ time} \\
\quad\quad \text{for } (i = 1;\ i < n;\ i{+}{+}) & \text{O(1) time for operations like i = 1, i < n and i++)} \\
\quad\quad\quad \text{if } (A[i] > A[i+1])\ \{ & O(1) \text{ time} \\
\quad\quad\quad\quad \text{exchange } A[i] \text{ and } A[i+1]; & O(1) \text{ time} \\
\quad\quad\quad\quad done = 0; & O(1) \text{ time} \\
\quad\quad\quad \} & \\
\quad \} \text{ while } (done == 0); & O(1) \text{ time} \\
\} &
\end{array}
$$

$f(n)$ iterations

$O(n)$ iterations

$f(n) = O(n)$

Proof sketch One can prove by induction on $k = 1, 2, \ldots, n$ that after the k-th iteration of the while-loop, we have

$$A[n-k+1] > A[n-k+2] > \cdots > A[n-1] > A[n].$$

It follows that after the n-th iteration, all of the n input numbers are sorted in decreasing order. Therefore,

$$f(n) \leq n$$

and thus $f(n) = O(n)$.

Hence complexity is f(n) . O(n) = O(n) * O(n) = O(n^2)

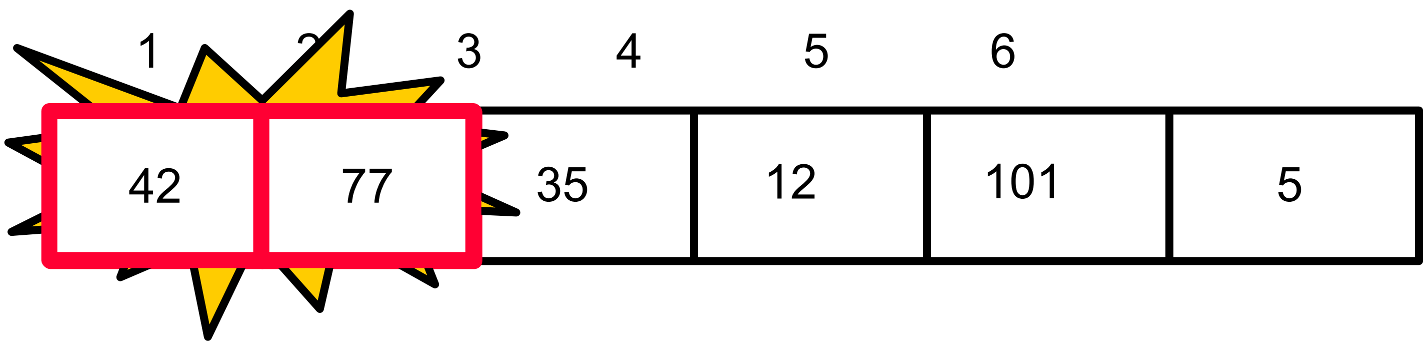
"Bubbling Up" the Largest Element

- Traverse a collection of elements
  - Move from the front to the end
  - "Bubble" the largest value to the end using pair-wise comparisons and swapping

1	2	3	4	5	6
77	42	35	12	101	5

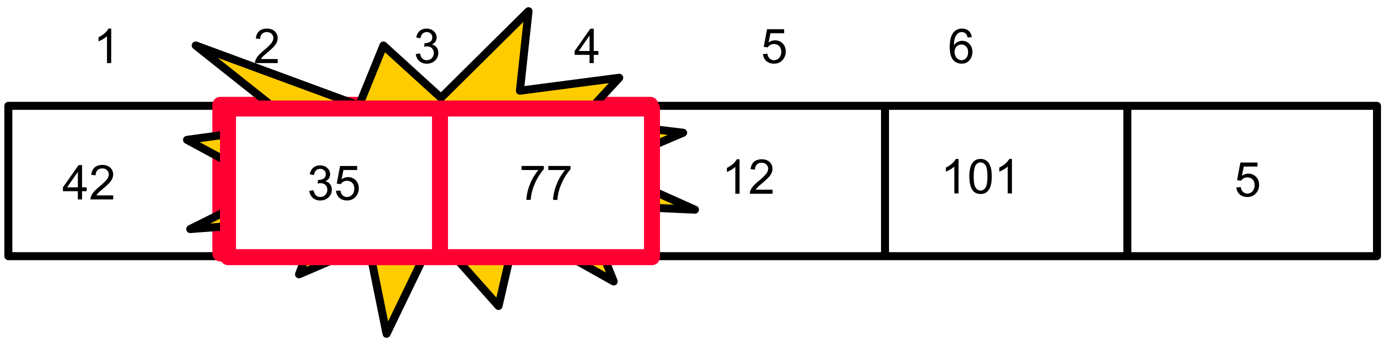
"Bubbling Up" the Largest Element

- Traverse a collection of elements
  - Move from the front to the end
  - “Bubble” the largest value to the end using pair-wise comparisons and swapping



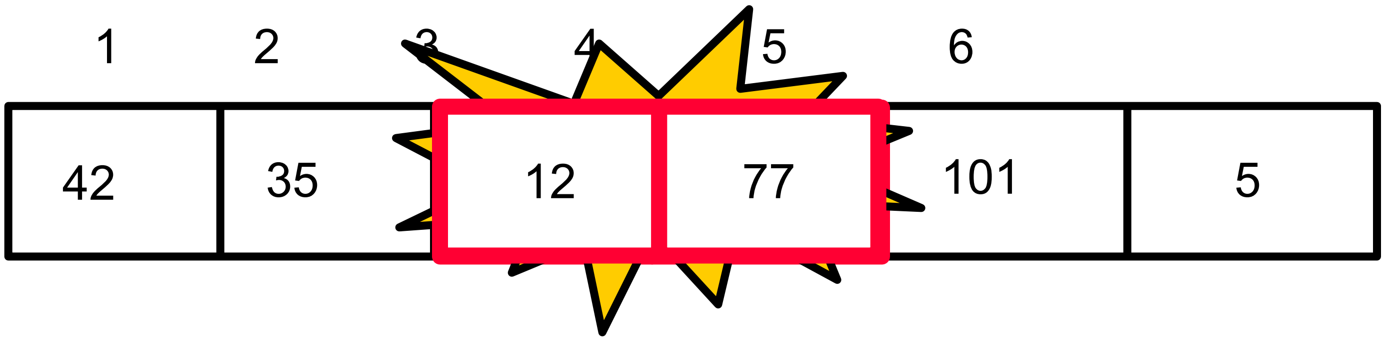
"Bubbling Up" the Largest Element

- Traverse a collection of elements
  - Move from the front to the end
  - "Bubble" the largest value to the end using pair-wise comparisons and swapping



"Bubbling Up" the Largest Element

- Traverse a collection of elements
  - Move from the front to the end
  - "Bubble" the largest value to the end using pair-wise comparisons and swapping



"Bubbling Up" the Largest Element

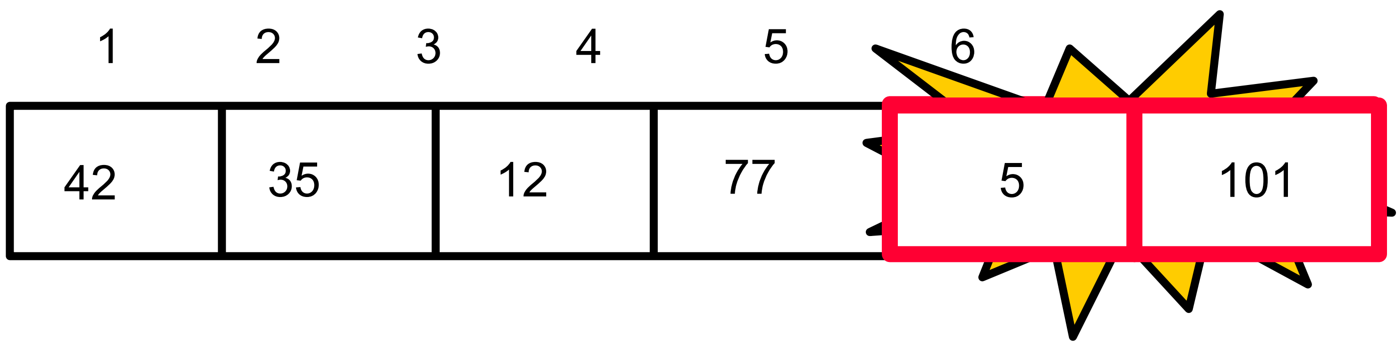
- Traverse a collection of elements
  - Move from the front to the end
  - "Bubble" the largest value to the end using pair-wise comparisons and swapping

1	2	3	4	5	6
42	35	12	77	101	5

No need to swap

"Bubbling Up" the Largest Element

- Traverse a collection of elements
  - Move from the front to the end
  - "Bubble" the largest value to the end using pair-wise comparisons and swapping



"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **"Bubble" the largest value to the end using pair-wise comparisons and swapping**

1	2	3	4	5	6	
42	35	12	77	5	**101**	

Largest value correctly placed

Items of Interest

- Notice that only the largest value is correctly placed
- All other values are still out of order
- So we need to repeat this process

1	2	3	4	5	6	
42	35	12	77	5		101

Largest value correctly placed

Repeat "Bubble Up" How Many Times?

- If we have N elements…
- And if each time we bubble an element, we place it in its correct location…
- Then we repeat the "bubble up" process N – 1 times.
- This guarantees we'll correctly place all N elements.

“Bubbling” All the Elements

N - 1

1	2	3	4	5	6	
42	35	12	77	5	101	

1	2	3	4	5	6	
35	12	42	5	77	101	

1	2	3	4	5	6	
12	35	5	42	77	101	

1	2	3	4	5	6	
12	5	35	42	77	101	

1	2	3	4	5	6	
5	12	35	42	77	101	


7 3 1 4 6 2 5
3 1 4 6 2 5 7
1 3 4 2 5 6 7
1 3 2 4 5 6 7

innovate achieve lead

Merge Sort

Divide and Conquer cuts the problem in half each time, but uses the result of both halves:

- ✓ cut the problem in half until the problem is trivial
- ✓ solve for both halves
- ✓ combine the solutions

Merge Sort, a sorting algorithm exploiting Divide and Conquer Technique

Merge Sort

John von Neumann
(1903-1957)

- ➢ Developed merge sort for EDVAC in 1945

Merge two sorted lists into a single sorted list

The **key to Merge Sort** is merging two sorted lists into one, such that if you have two lists X ($x_1 \le x_2 \le \cdots \le x_m$) and Y($y_1 \le y_2 \le \cdots \le y_n$) the resulting list is Z($z_1 \le z_2 \le \cdots \le z_{m+n}$)

Example: $L_1 = \{ 3\ 8\ 9 \}$

$L_2 = \{ 1\ 5\ 7 \}$

$merge(L_1, L_2) = \{ 1\ 3\ 5\ 7\ 8\ 9 \}$

What is complexity of the below mentioned algorithm?

Merge two sorted lists into a single sorted list

Algorithm M (*Two-way merge*). This algorithm merges the ordered files $x_1 \le x_2 \le \cdots \le x_m$ and $y_1 \le y_2 \le \cdots \le y_n$ into a single file $z_1 \le z_2 \le \cdots \le z_{m+n}$.

M1. [Initialize.] Set $i \leftarrow 1$, $j \leftarrow 1$, $k \leftarrow 1$.

M2. [Find smaller.] If $x_i \le y_j$, go to step M3, otherwise go to M5.

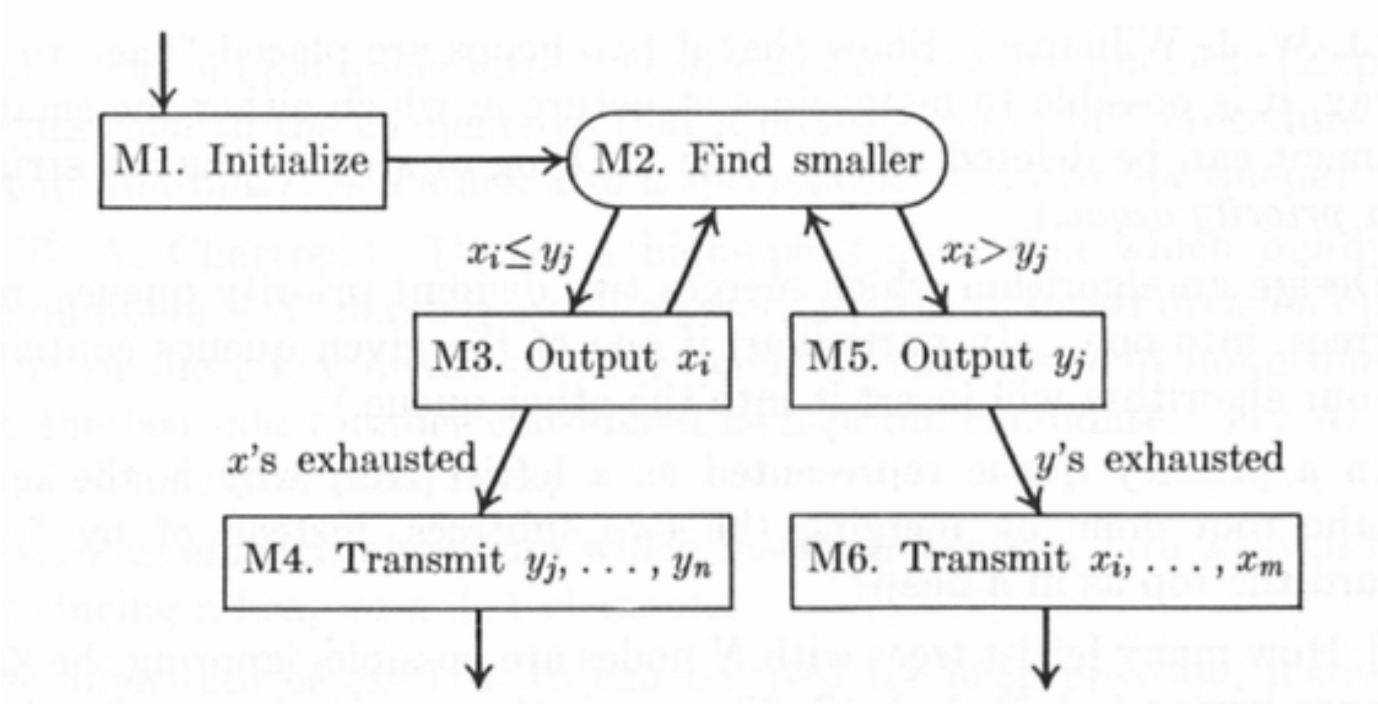
M3. [Output x_i.] Set $z_k \leftarrow x_i$, $k \leftarrow k+1$, $i \leftarrow i+1$. If $i \le m$, return to M2.

M4. [Transmit $y_j, \ldots, y_n$.] Set $(z_k, \ldots, z_{m+n}) \leftarrow (y_j, \ldots, y_n)$ and terminate the algorithm.

M5. [Output y_j.] Set $z_k \leftarrow y_j$, $k \leftarrow k+1$, $j \leftarrow j+1$. If $j \le n$, return to M2.

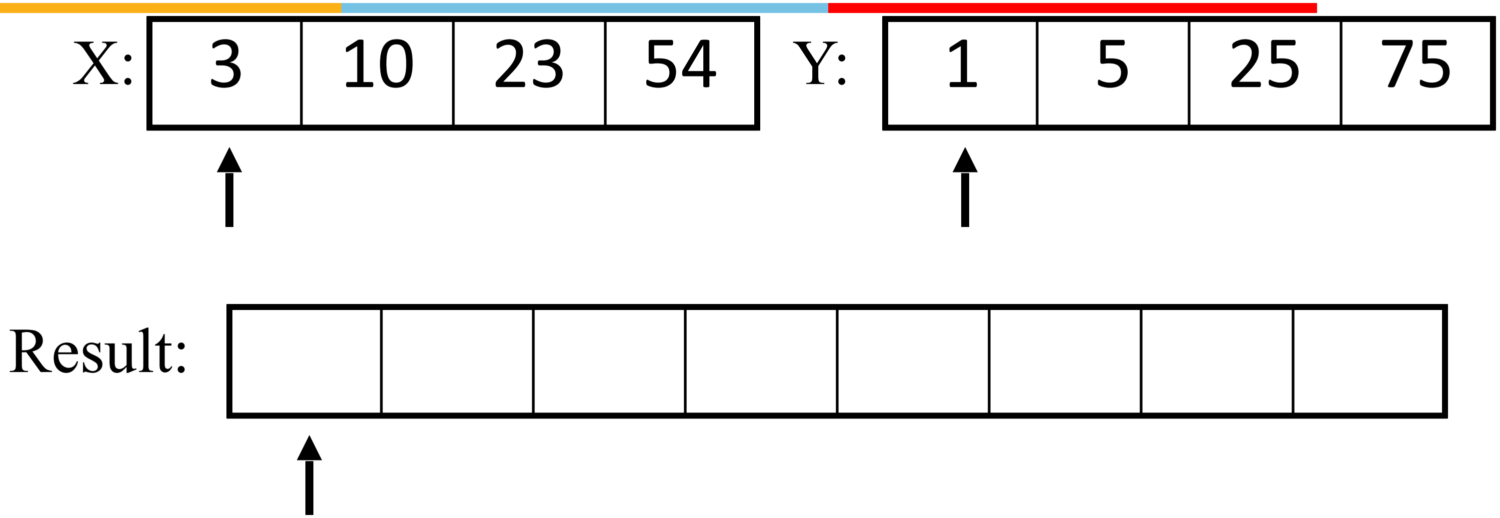
M6. [Transmit $x_i, \ldots, x_m$.] Set $(z_k, \ldots, z_{m+n}) \leftarrow (x_i, \ldots, x_m)$ and terminate the algorithm.

Merge two sorted lists into a single sorted list



What is complexity of the below mentioned algorithm?

Merge two sorted lists into a single sorted list



Merge two sorted lists into a single sorted list

X:

3	10	23	54

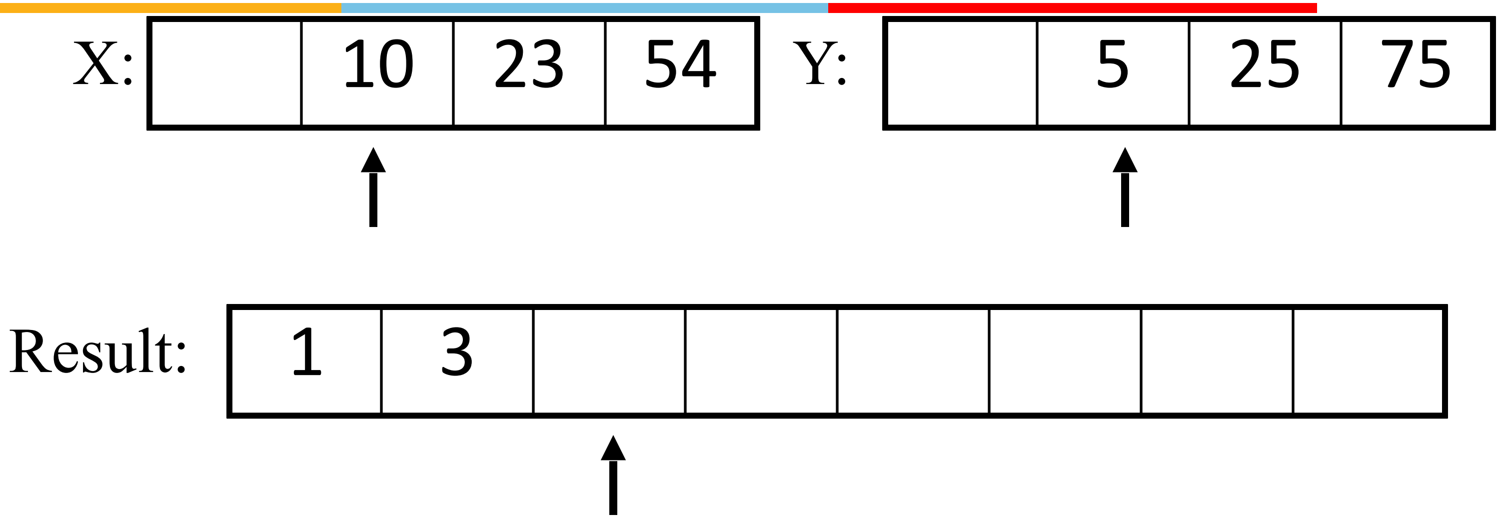
Y:

	5	25	75

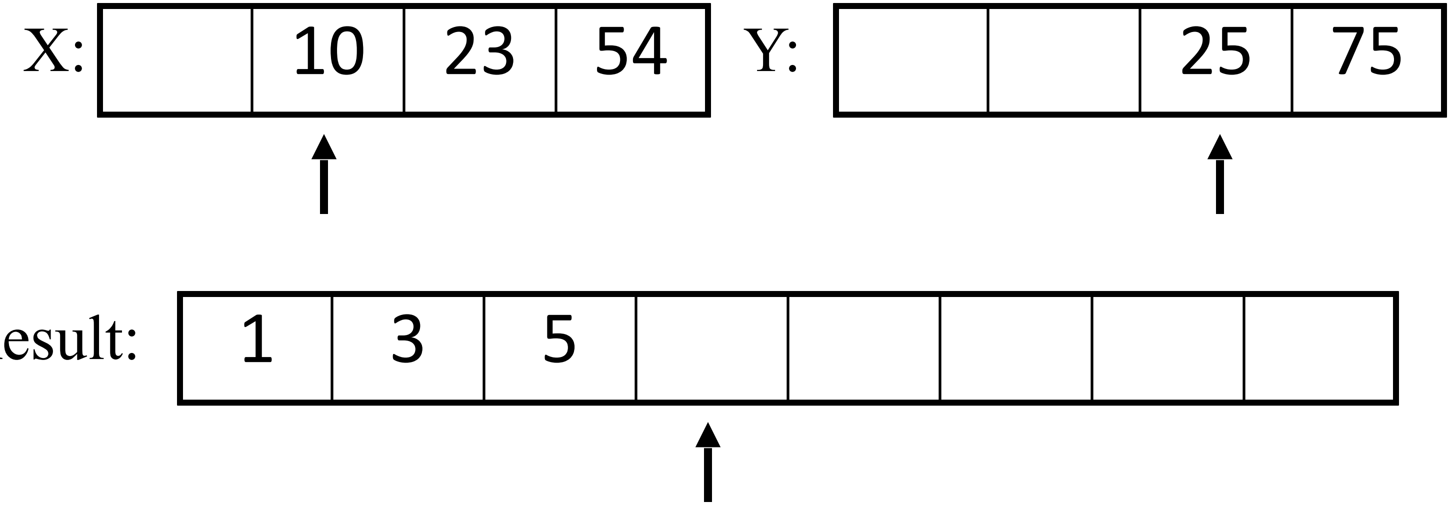
Result:

1							

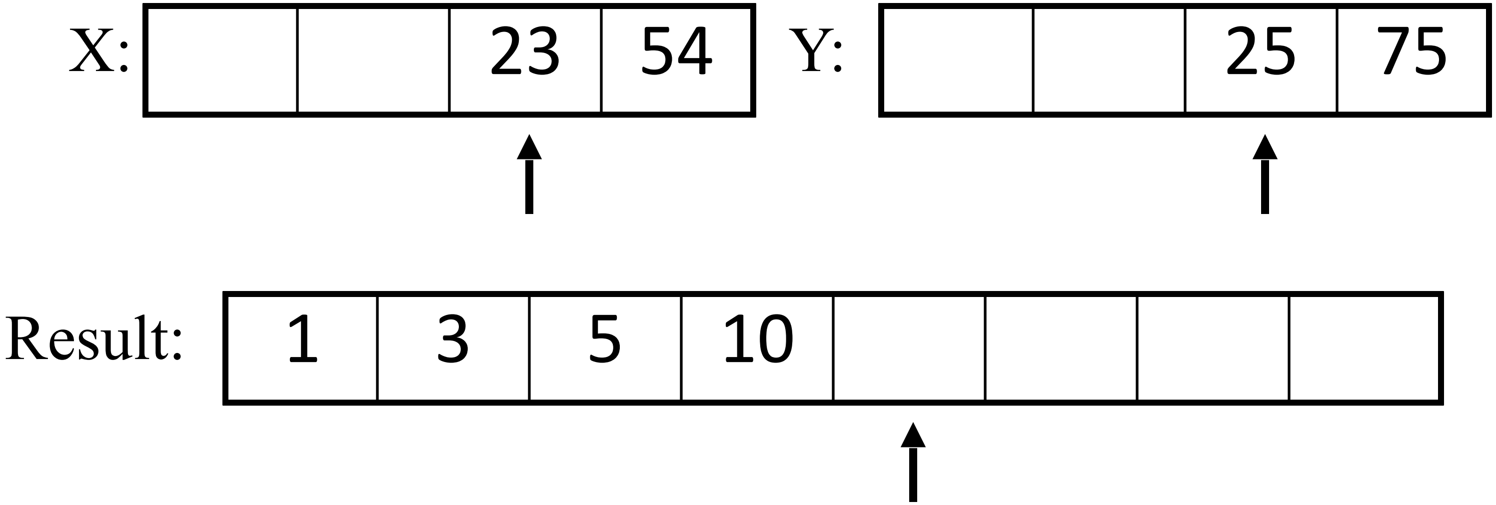
Merge two sorted lists into a single sorted list



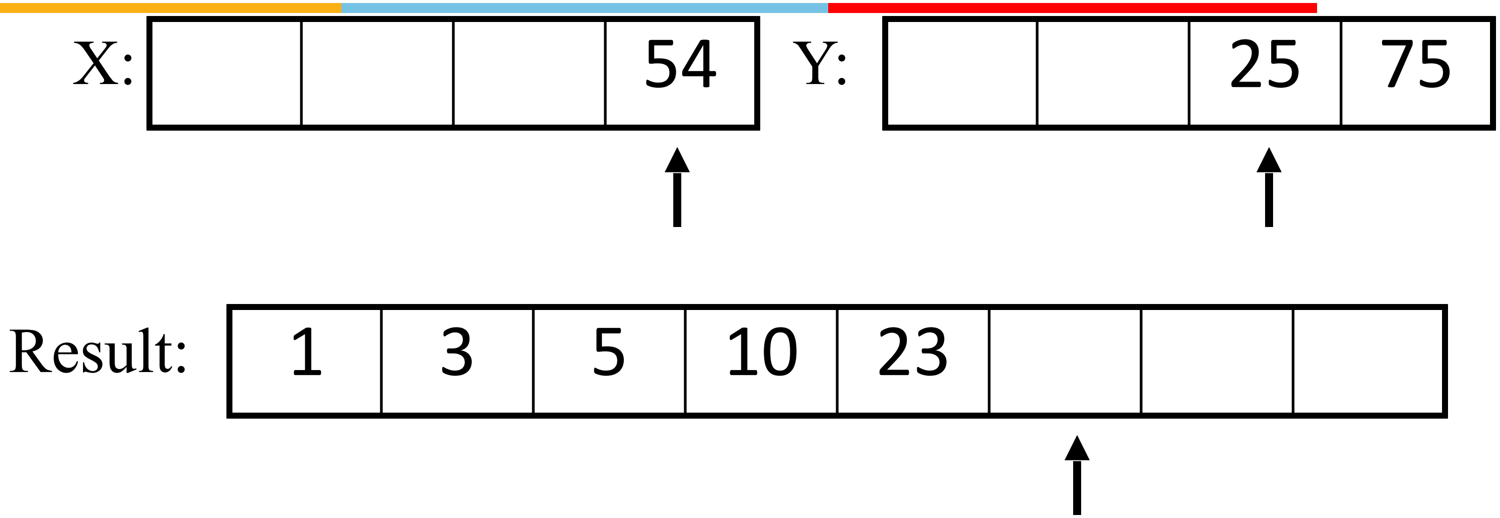
Merge two sorted lists into a single sorted list



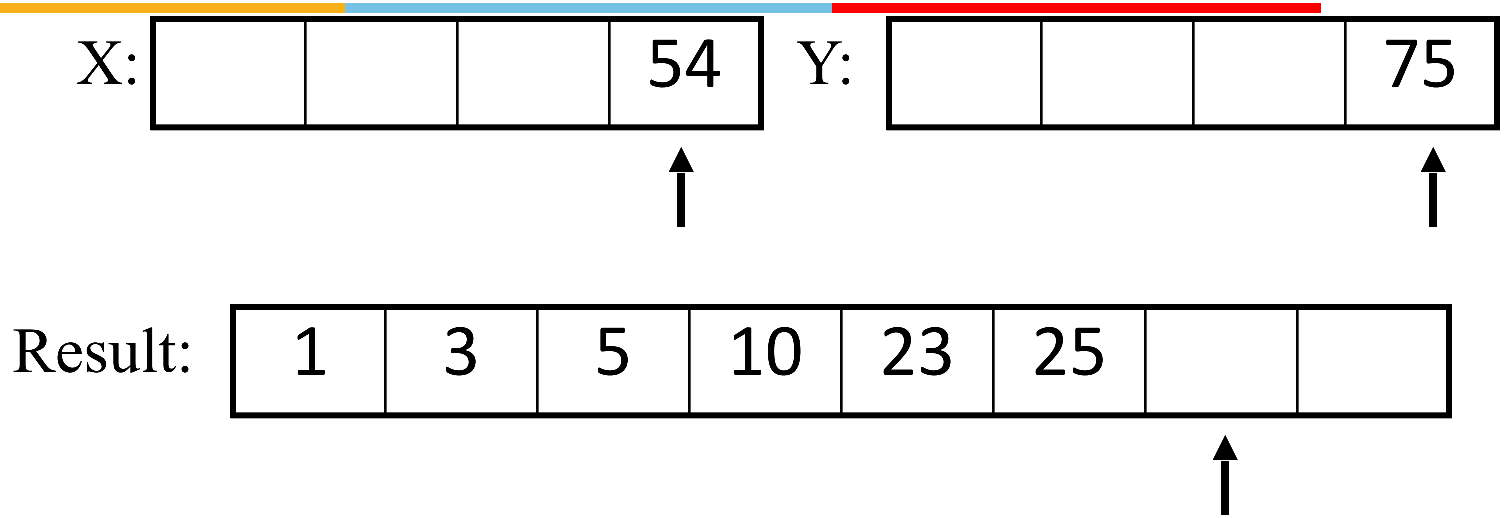
Merge two sorted lists into a single sorted list



Merge two sorted lists into a single sorted list



Merge two sorted lists into a single sorted list



Merge two sorted lists into a single sorted list

X:

Y:

			75

Result:

1	3	5	10	23	25	54	

Merge two sorted lists into a single sorted list

X:

Y:

Result:

1	3	5	10	23	25	54	75

Merge Sort

A divide-and-conquer algorithm:

- Divide the unsorted array into 2 halves until the sub-arrays only contain one element
- Merge the sub-problem solutions together:
  - Compare the sub-array's first elements
  - Remove the smallest element and put it into the result array
  - Continue the process until all elements have been put into the result array

37	23	6	89	15	12	2	19

Merge Sort Algorithm

Mergesort(Pass an array)

if array size > 1

Divide array in half

Call Mergesort on first half.

Call Mergesort on second half.

Merge two halves.

Merge(Pass two arrays)

Compare leading element in each array

Select lower and place in new array.

(If one input array is empty then place remainder of other array in output array)

Merge Sort Algorithm

```
MergeSort(A, left, right) {
   if (left < right) {
       mid = floor((left + right) / 2);
       MergeSort(A, left, mid);
       MergeSort(A, mid+1, right);
       Merge(A, left, mid, right);
   }
}

// Merge() takes two sorted subarrays of A and
// merges them into a single sorted subarray of A
//    (how long should this take?)
```

Analysis of Merge Sort

Statement	Effort
MergeSort(A, left, right) {	$T(n)$
if (left < right) {	$\Theta(1)$
mid = floor((left + right) / 2);	$\Theta(1)$
MergeSort(A, left, mid);	$T(n/2)$
MergeSort(A, mid+1, right);	$T(n/2)$
Merge(A, left, mid, right);	$\Theta(n)$
}	
}	

So $T(n) =$ $\Theta(1)$ when $n = 1$, and

$2T(n/2) + \Theta(n)$ when $n > 1$

✓ So what (more succinctly) is $T(n)$?

98	23	45	14	6	67	33	42

98	23	45	14	6	67	33	42

98	23	45	14

6	67	33	42

98	23	45	14	6	67	33	42

98	23	45	14

6	67	33	42

98	23

45	14

98	23	45	14	6	67	33	42

98	23	45	14

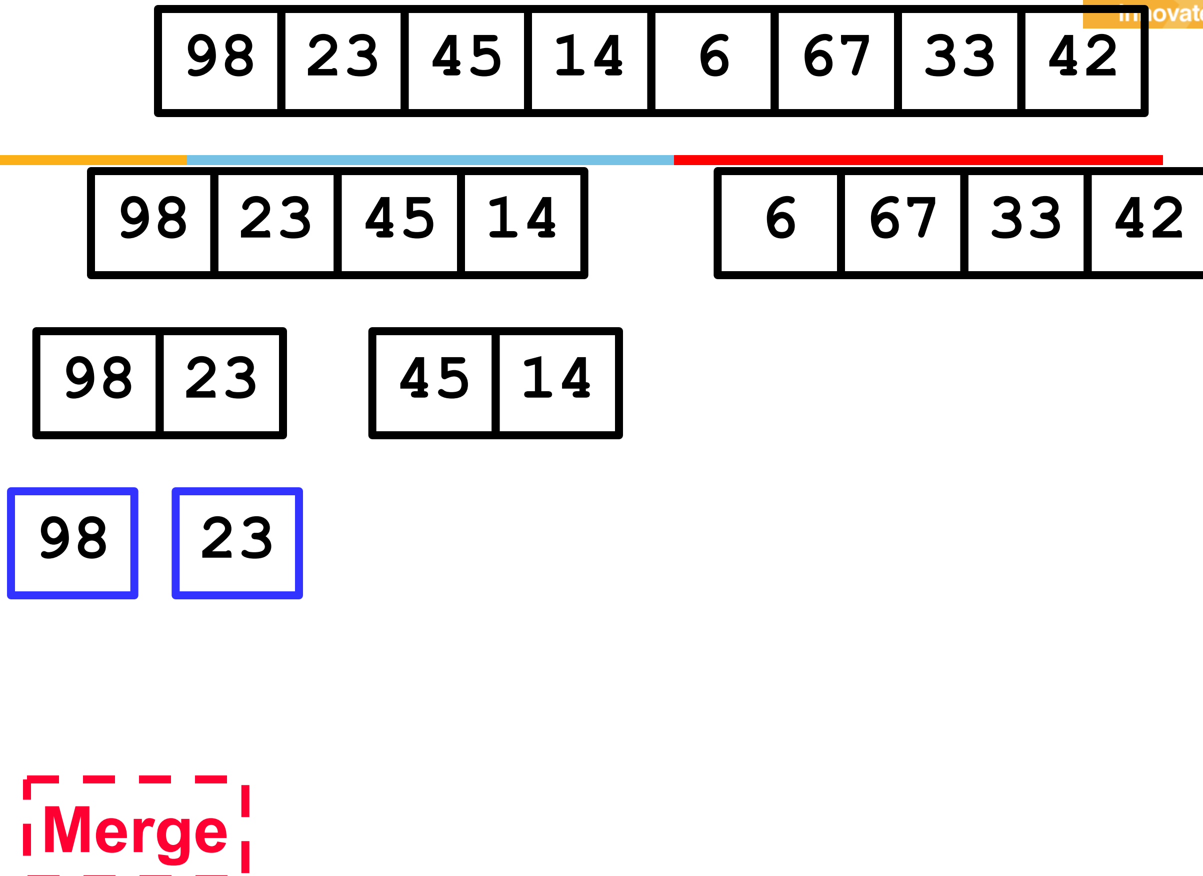
6	67	33	42

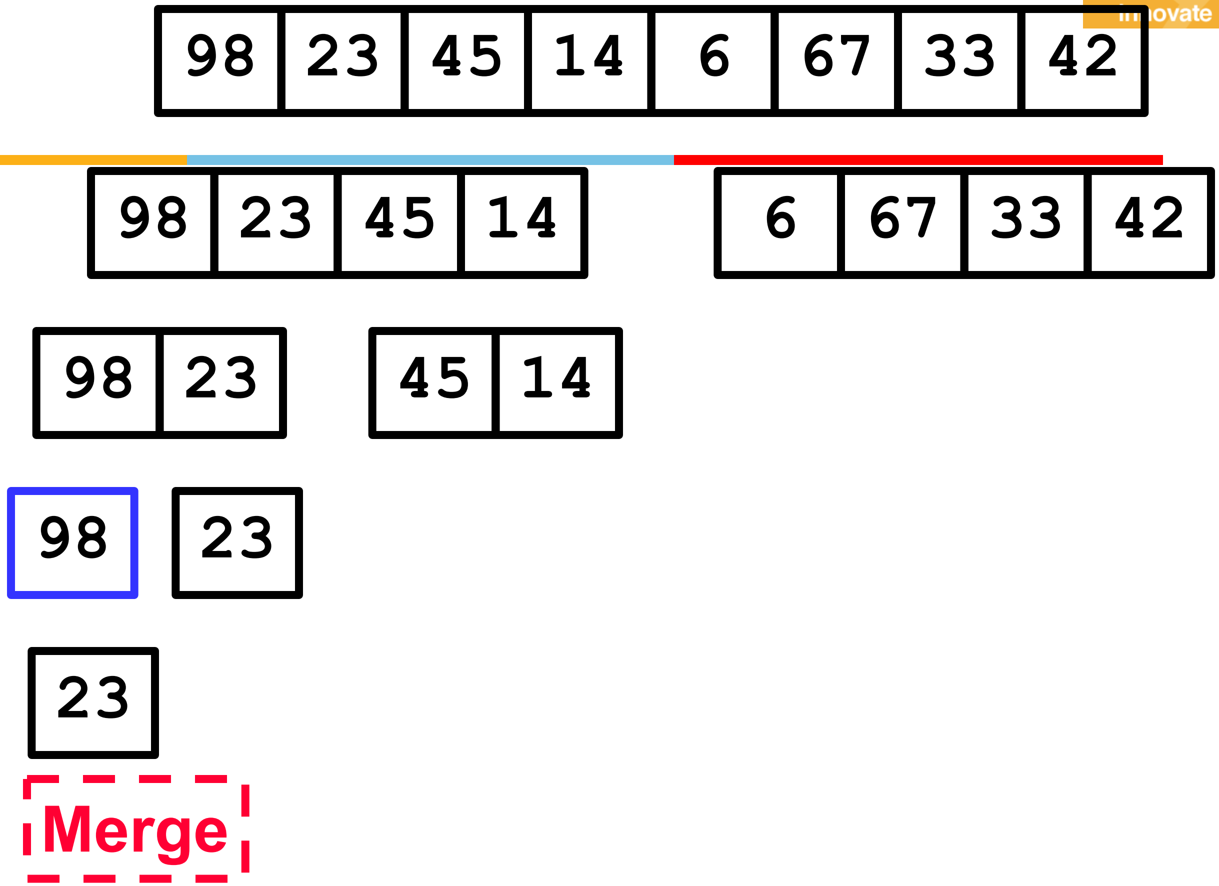
98	23

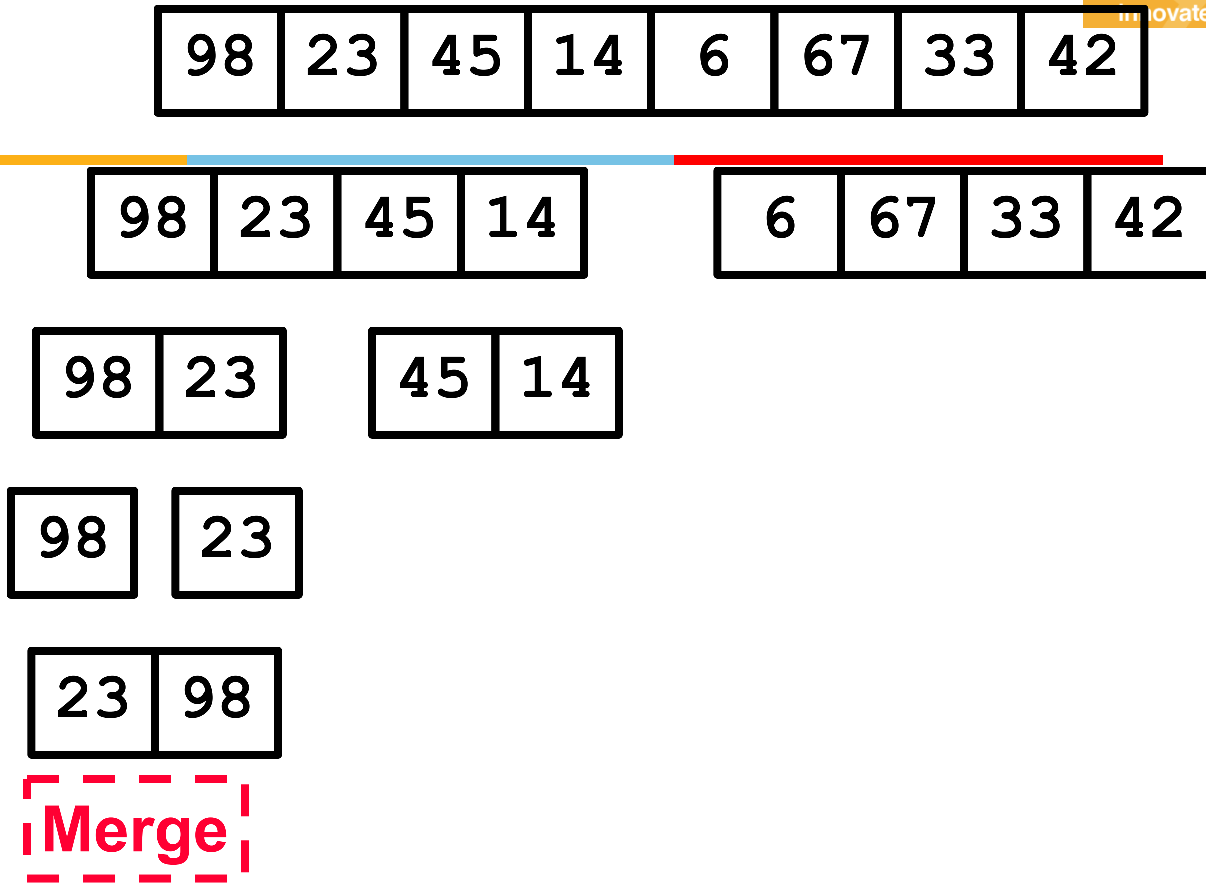
45	14

98

23


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
23
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
23 98
Merge

98	23	45	14	6	67	33	42

98	23	45	14

6	67	33	42

98	23

45	14

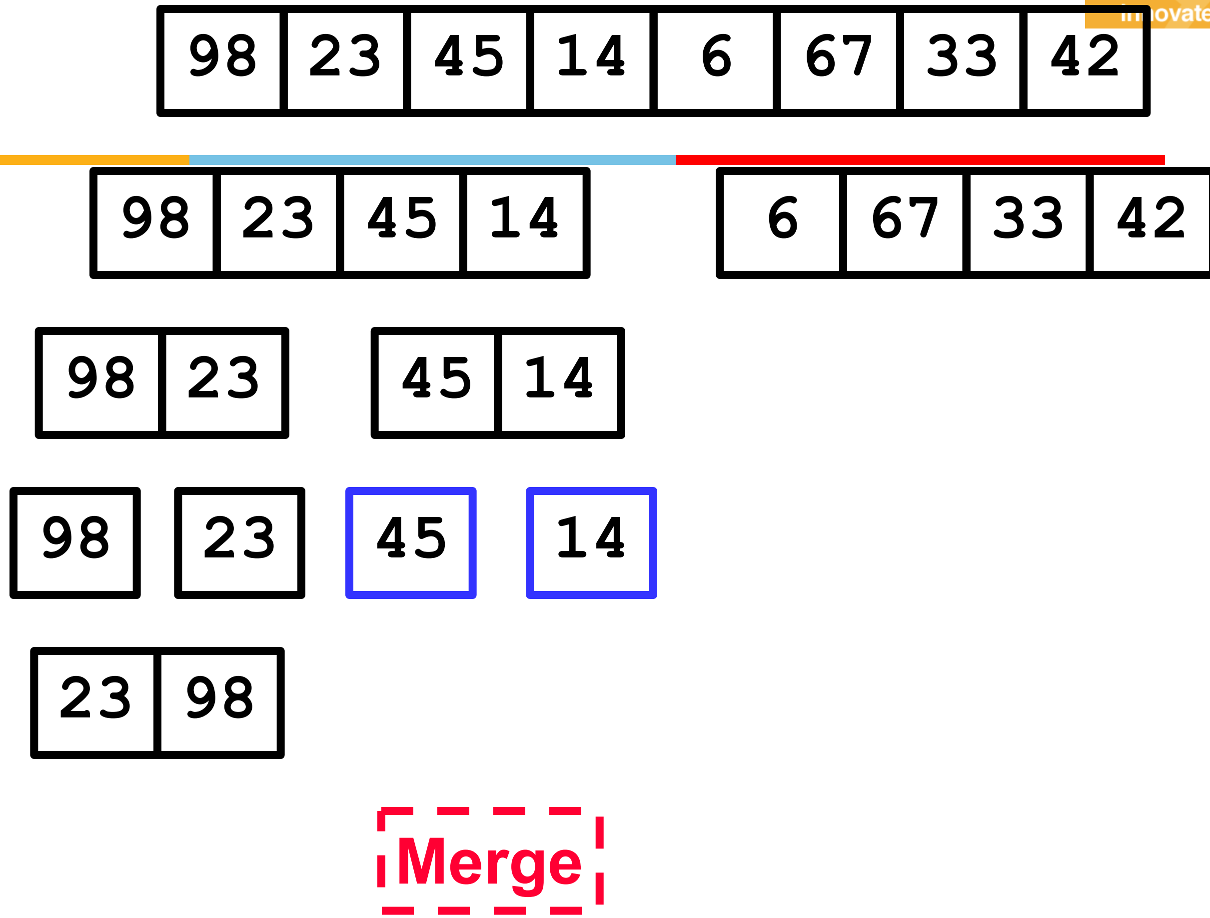
98

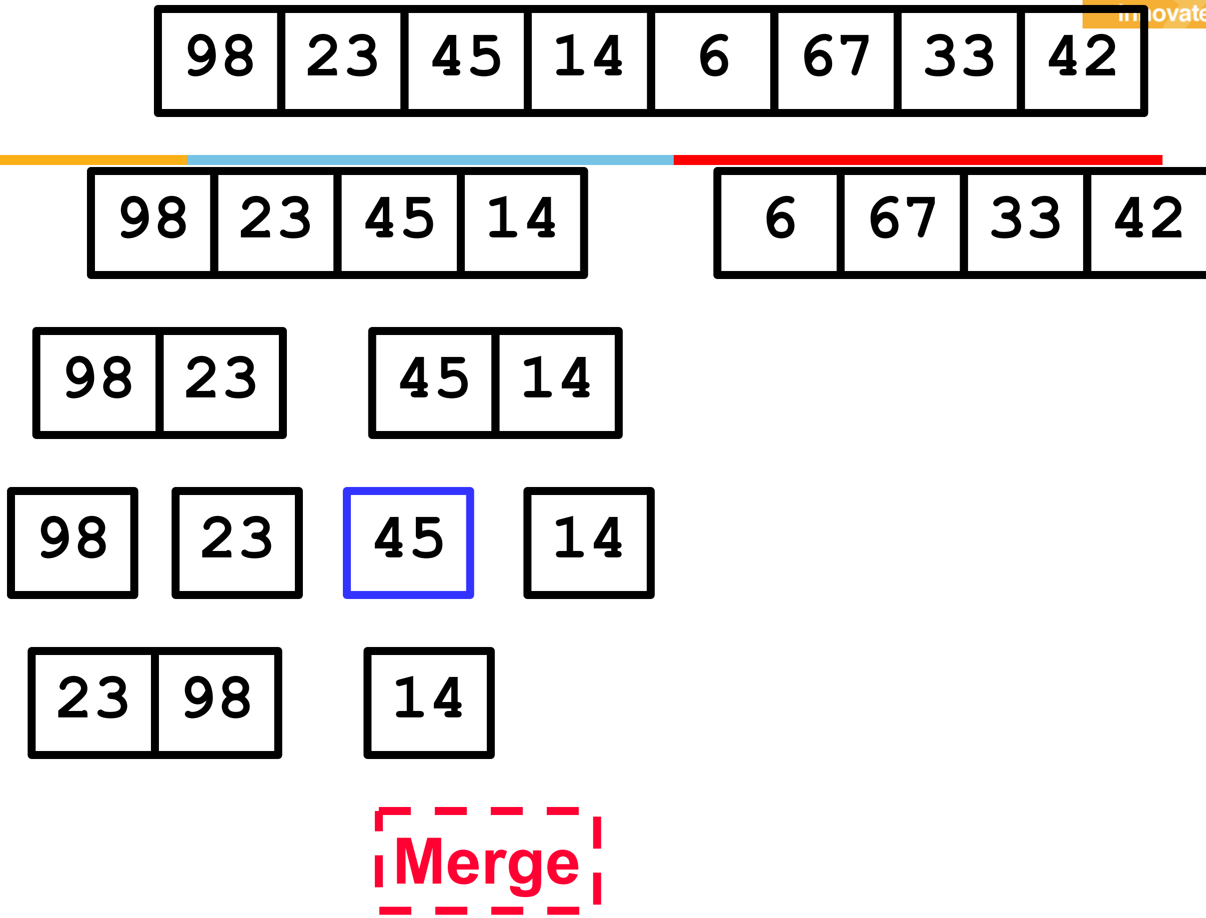
23

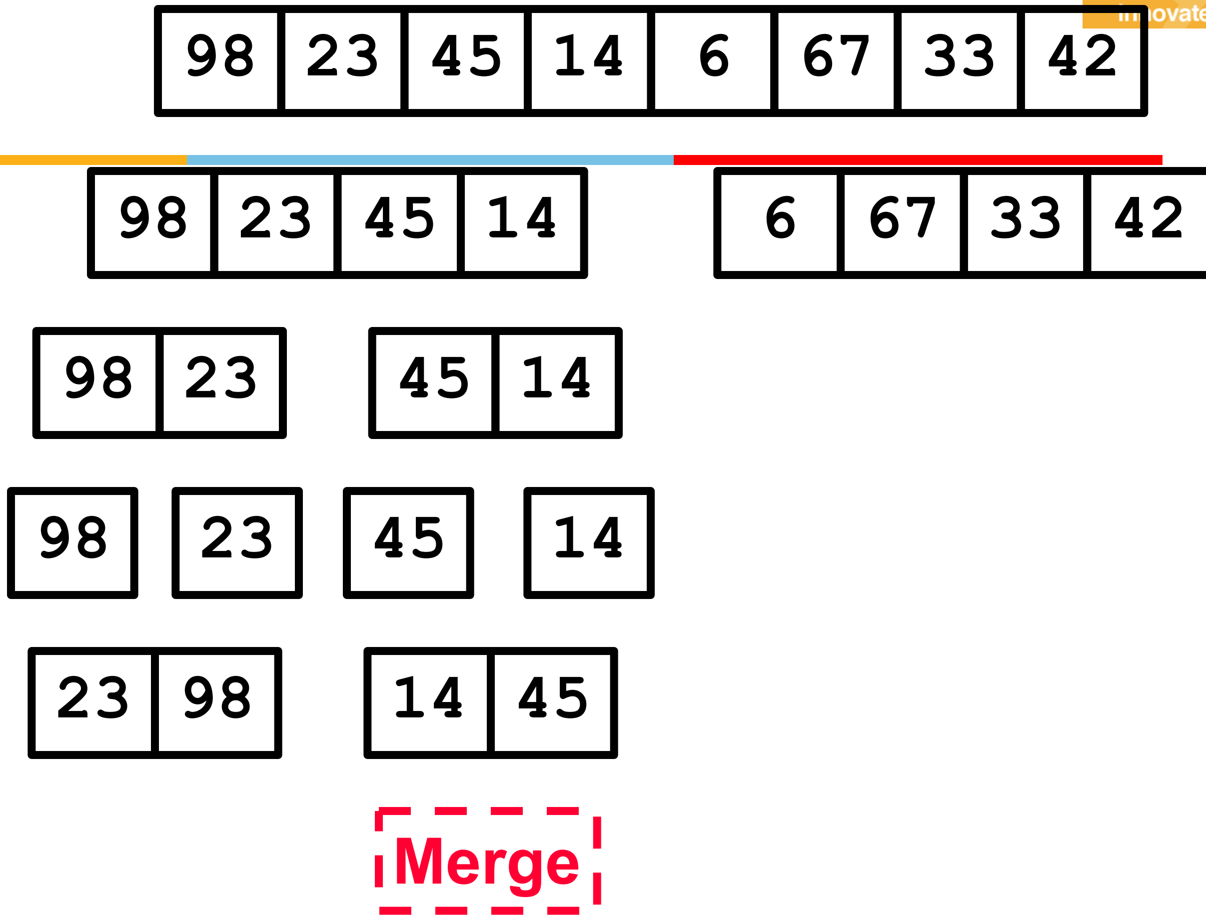
45

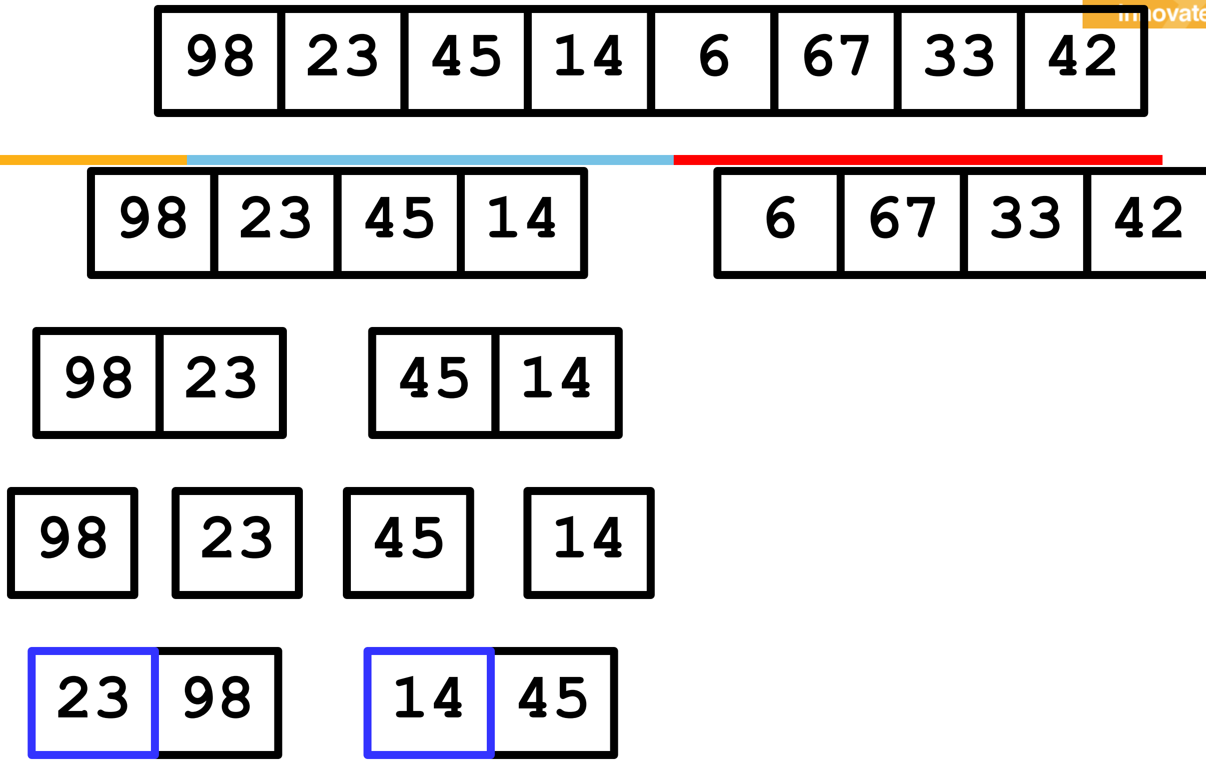
14

23	98

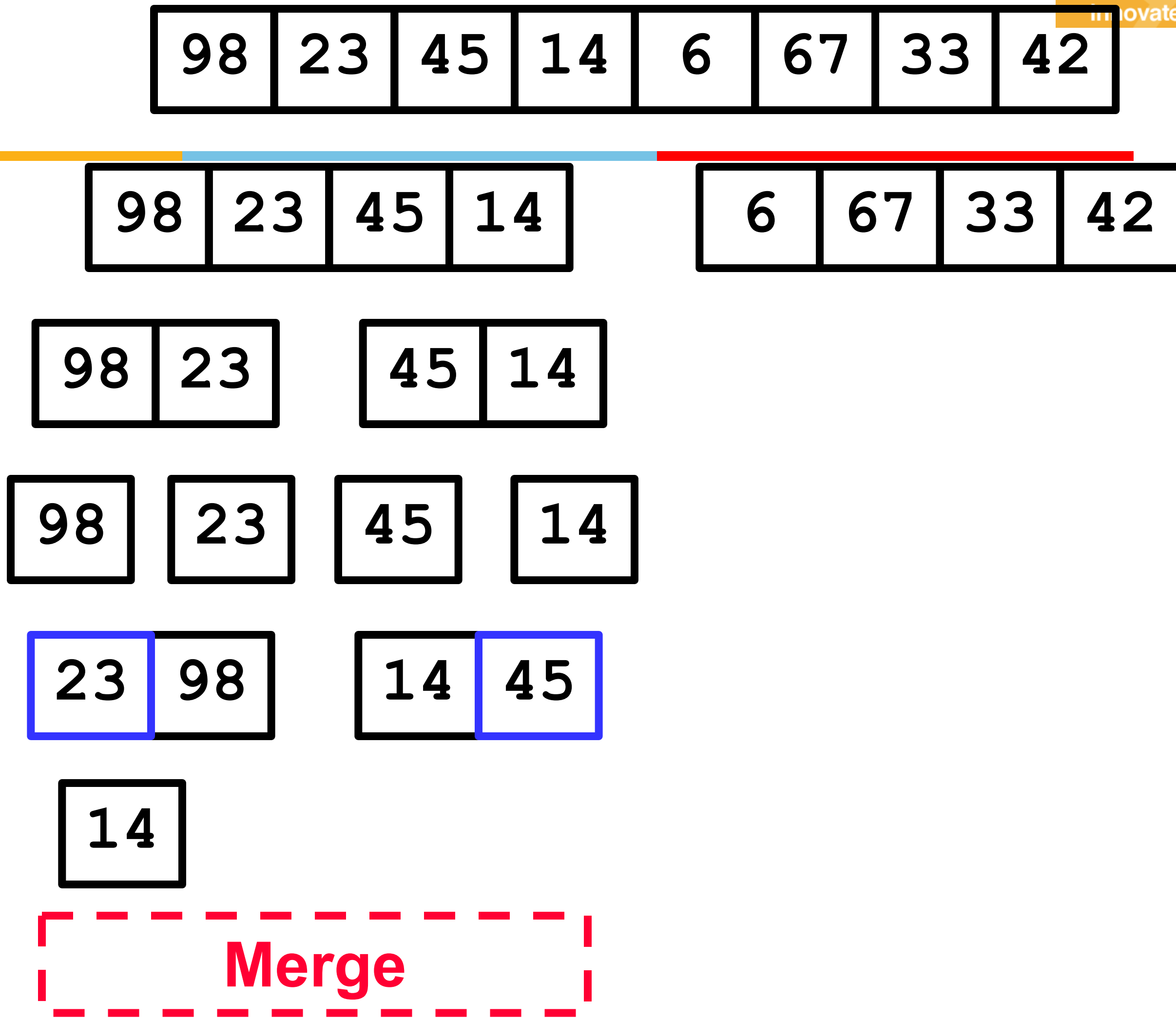

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
Merge

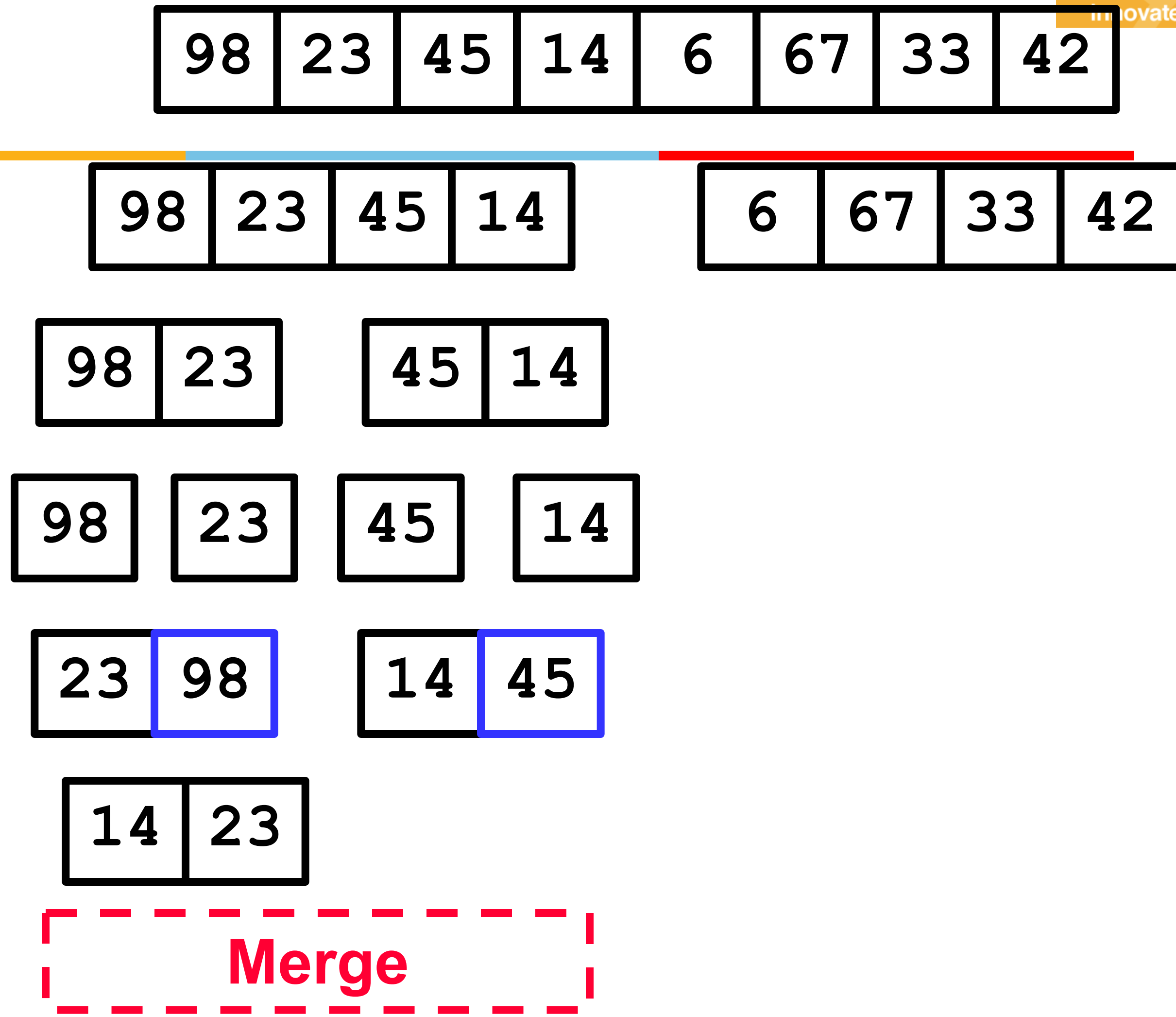

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14
Merge

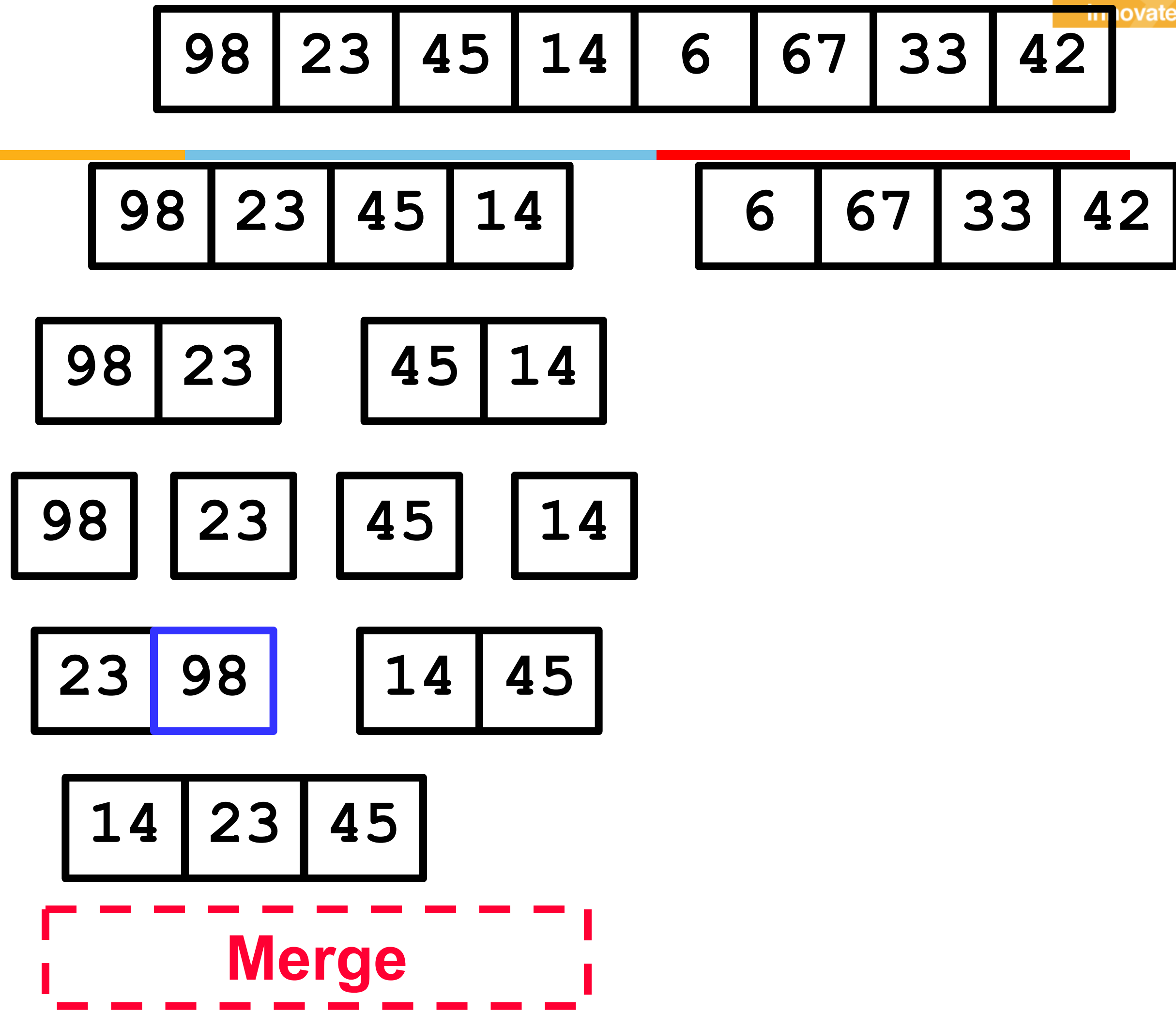

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
Merge

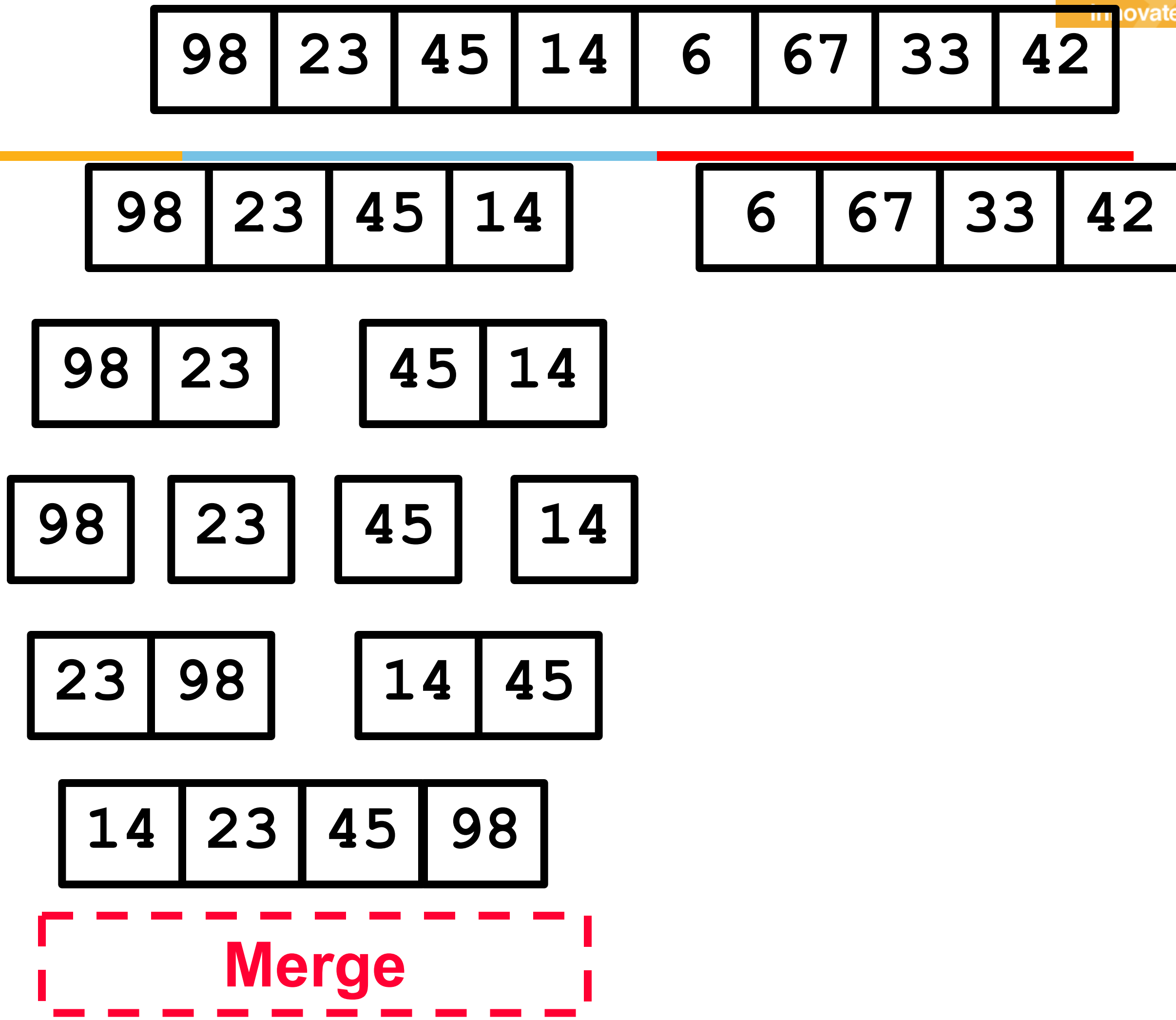

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45

Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
14
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
14 23
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
14 23 45
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
14 23 45 98
Merge

98	23	45	14	6	67	33	42

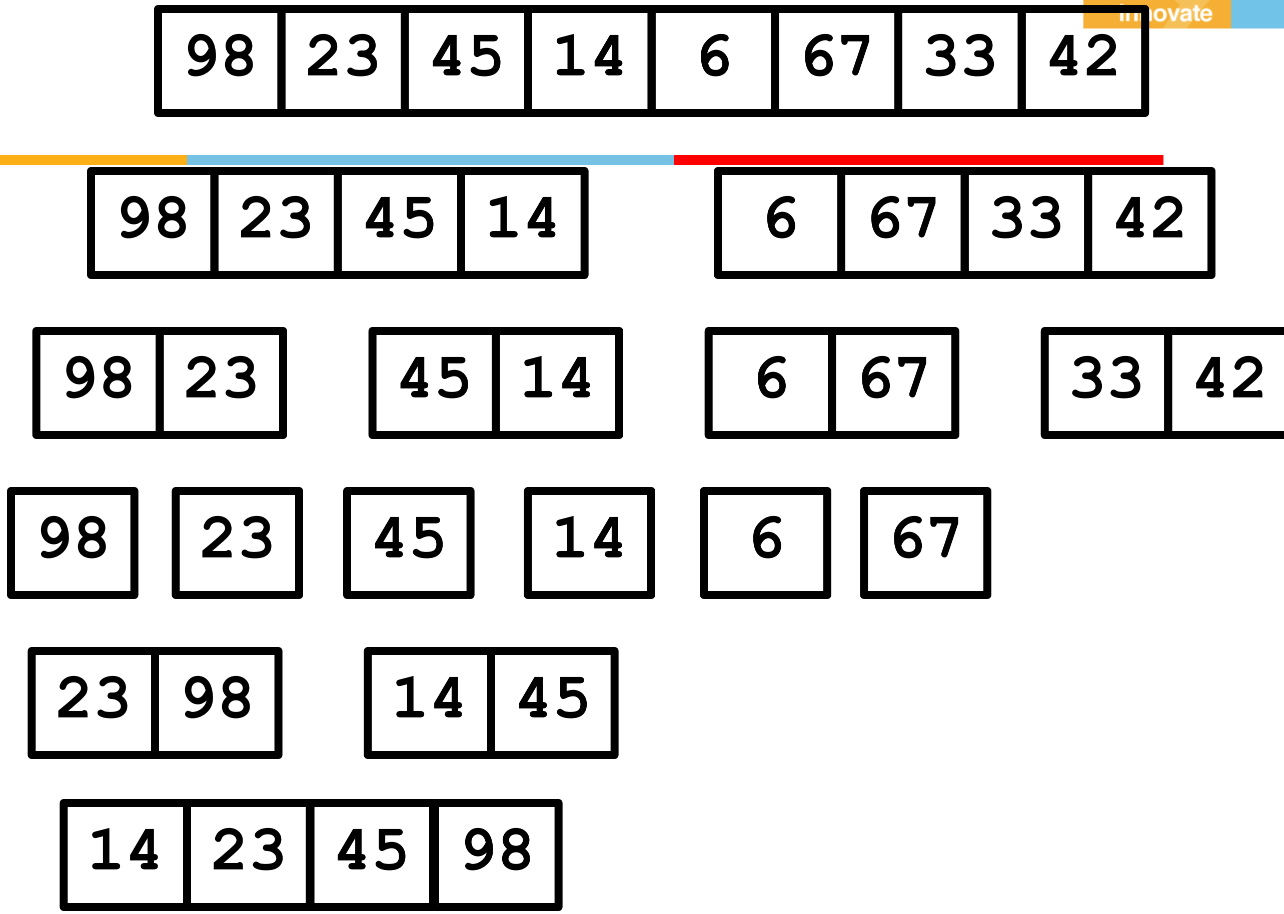
[98 | 23 | 45 | 14]   [6 | 67 | 33 | 42]

[98 | 23]   [45 | 14]   [6 | 67]   [33 | 42]

[98]   [23]   [45]   [14]

[23 | 98]   [14 | 45]

[14 | 23 | 45 | 98]


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
23 98
14 45
14 23 45 98

98	23	45	14	6	67	33	42

98 23 45 14   6 67 33 42

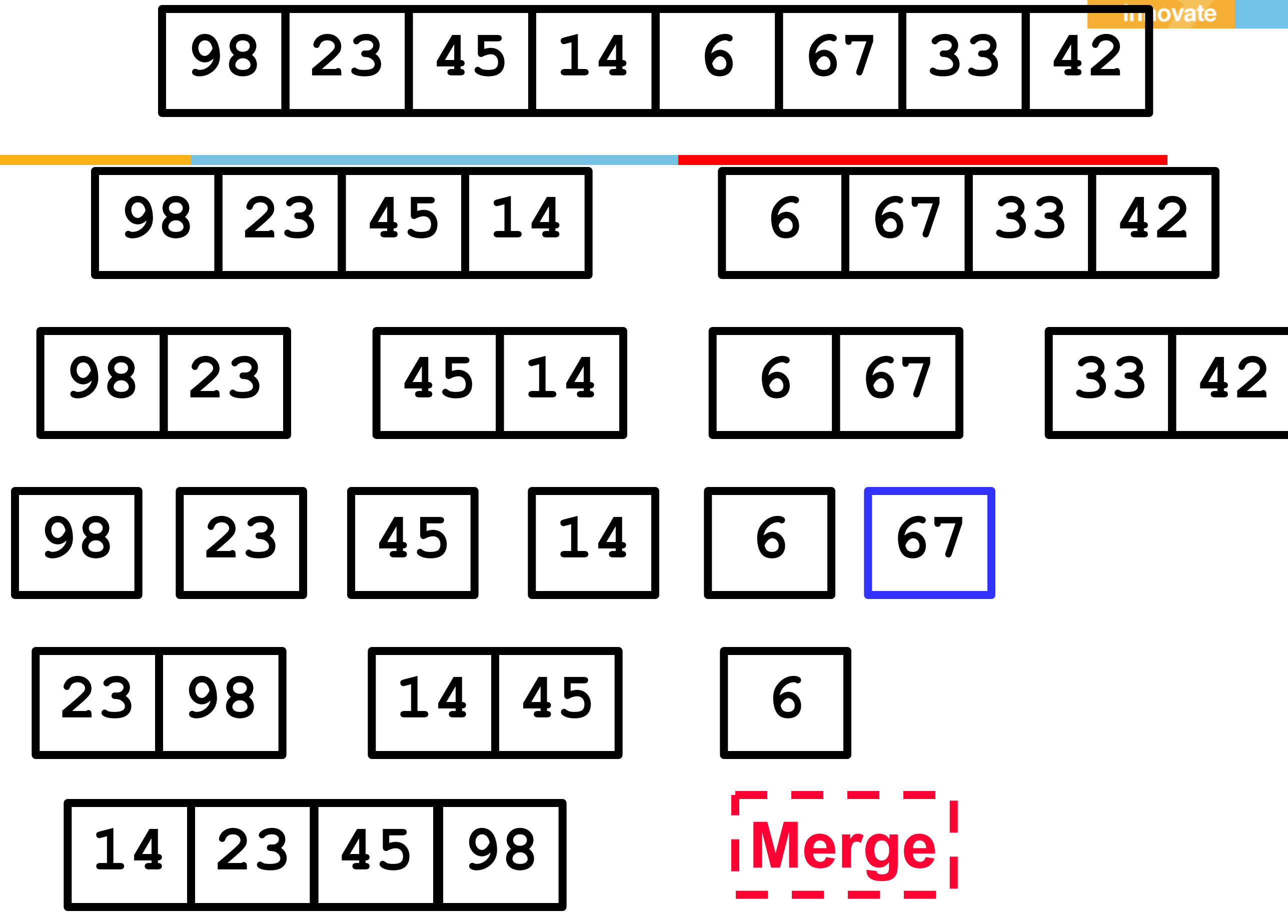
98 23   45 14   6 67   33 42

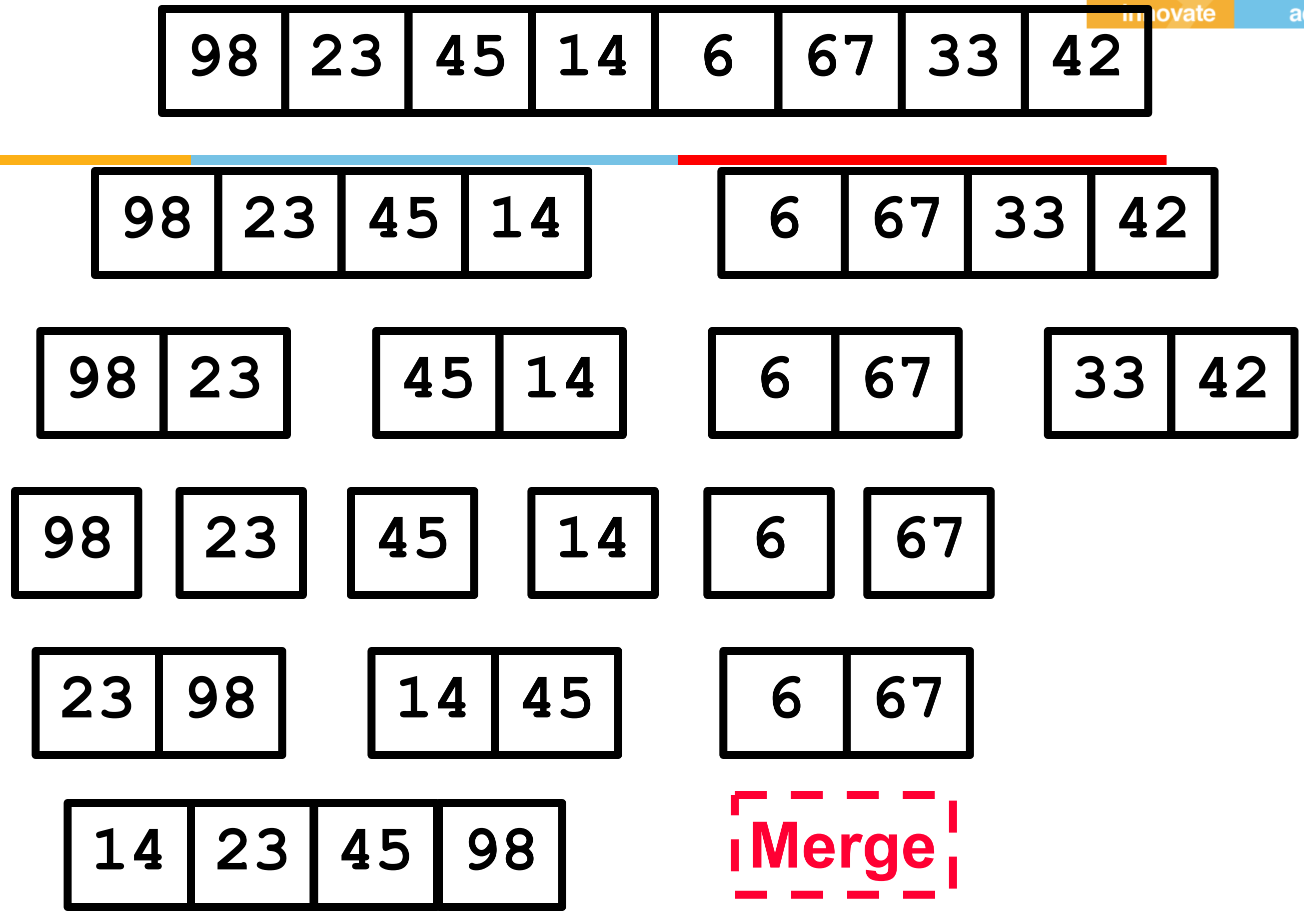
98   23   45   14   6   67

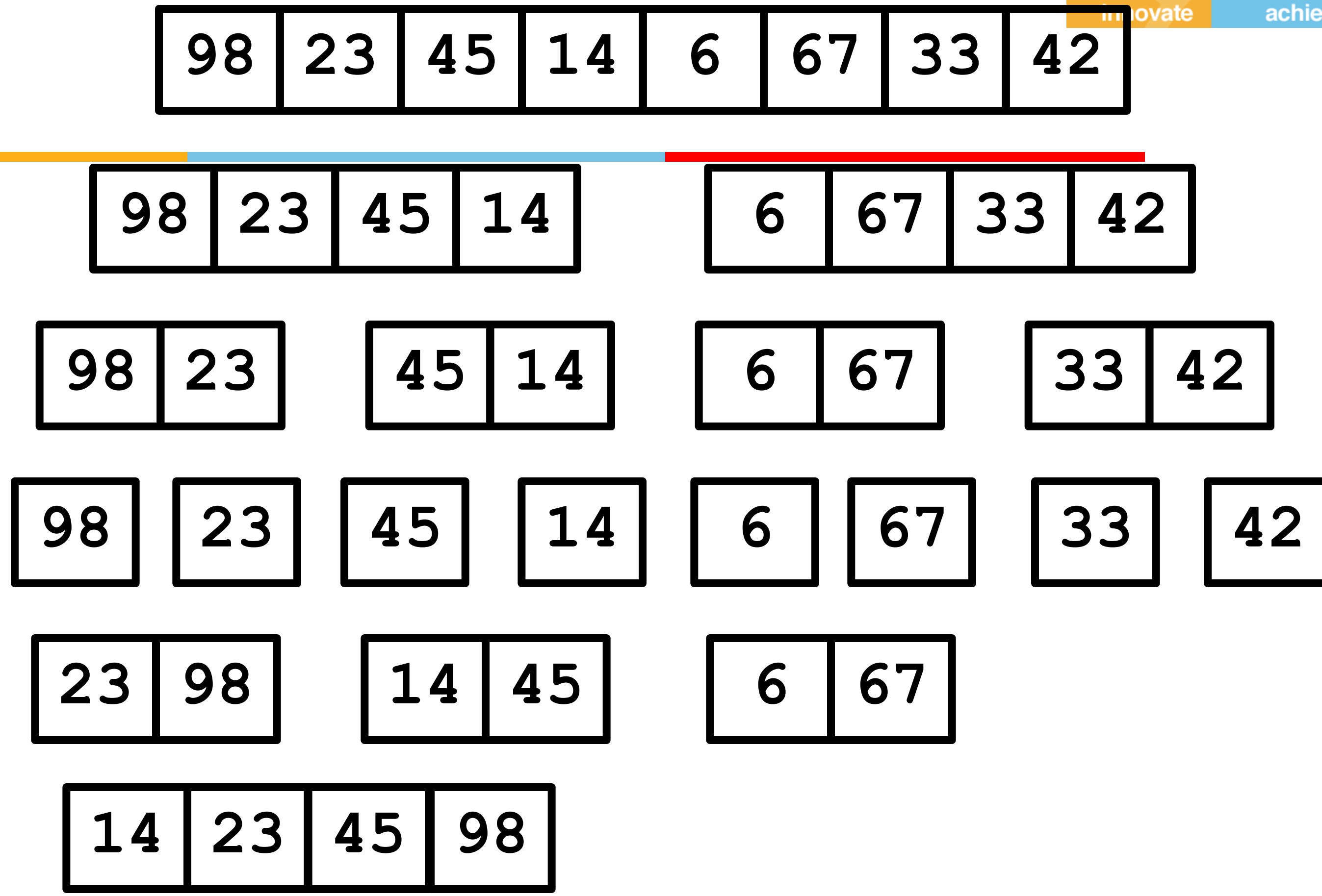
23 98   14 45

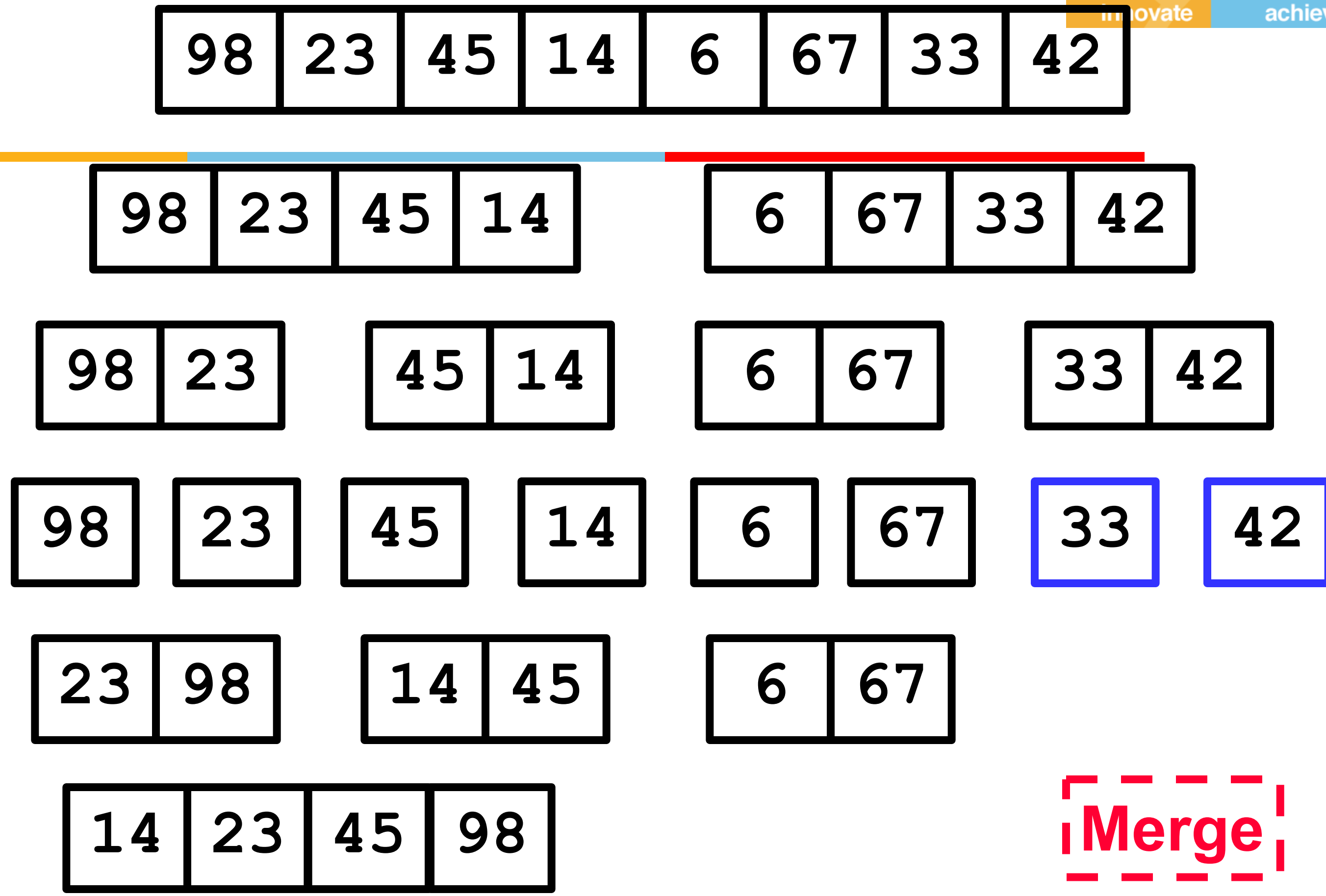
14 23 45 98

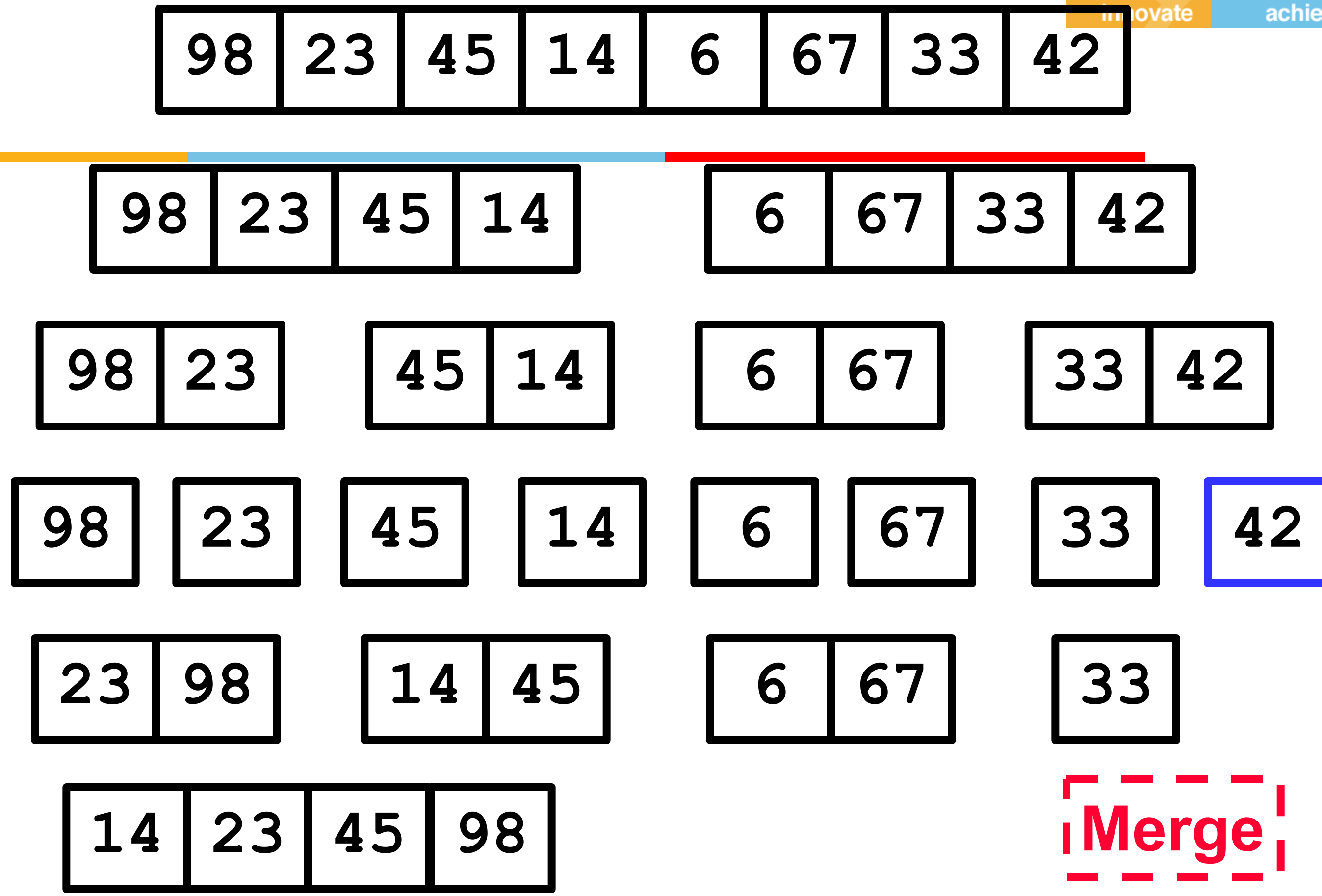
Merge

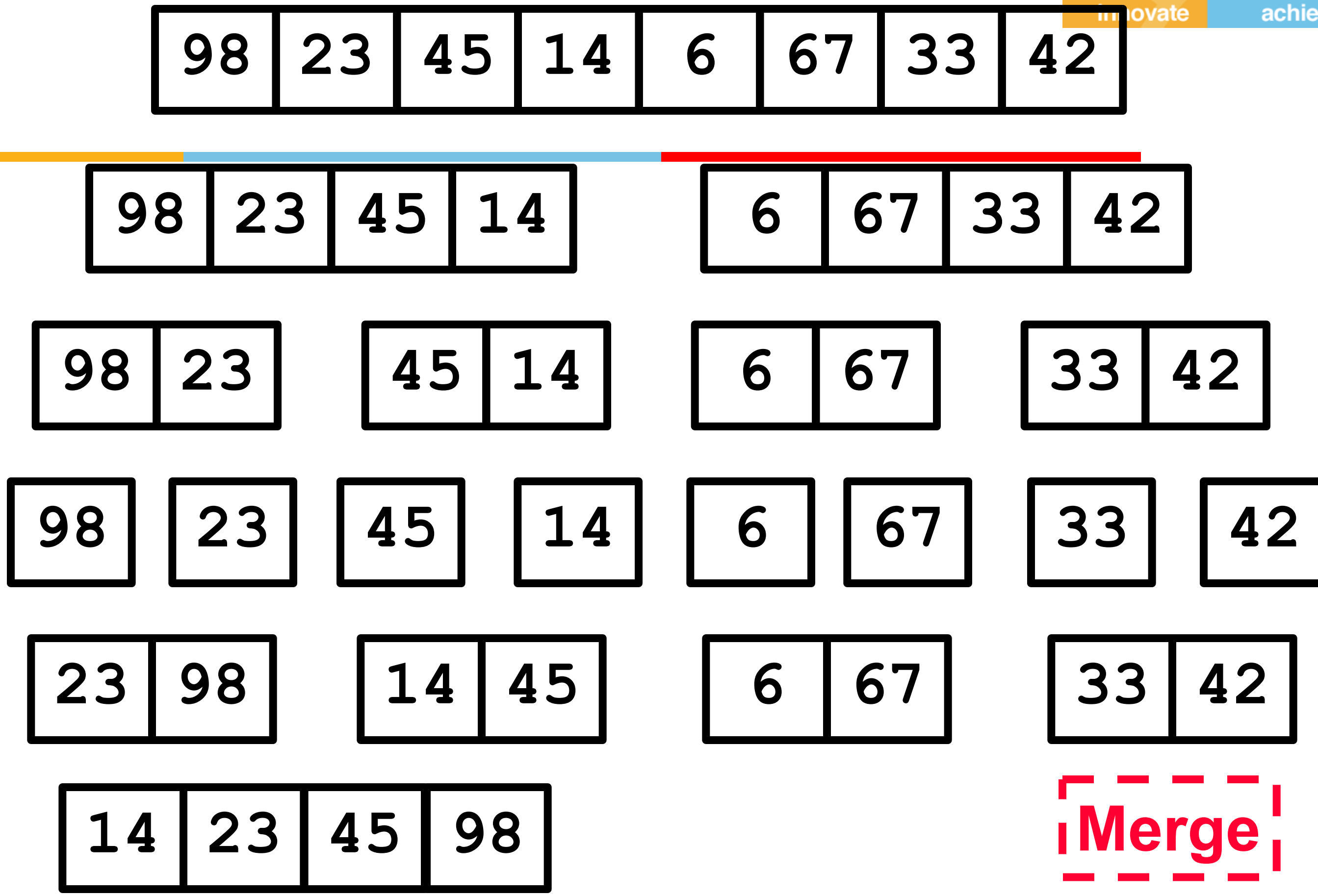

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
23 98
14 45
6
14 23 45 98
Merge

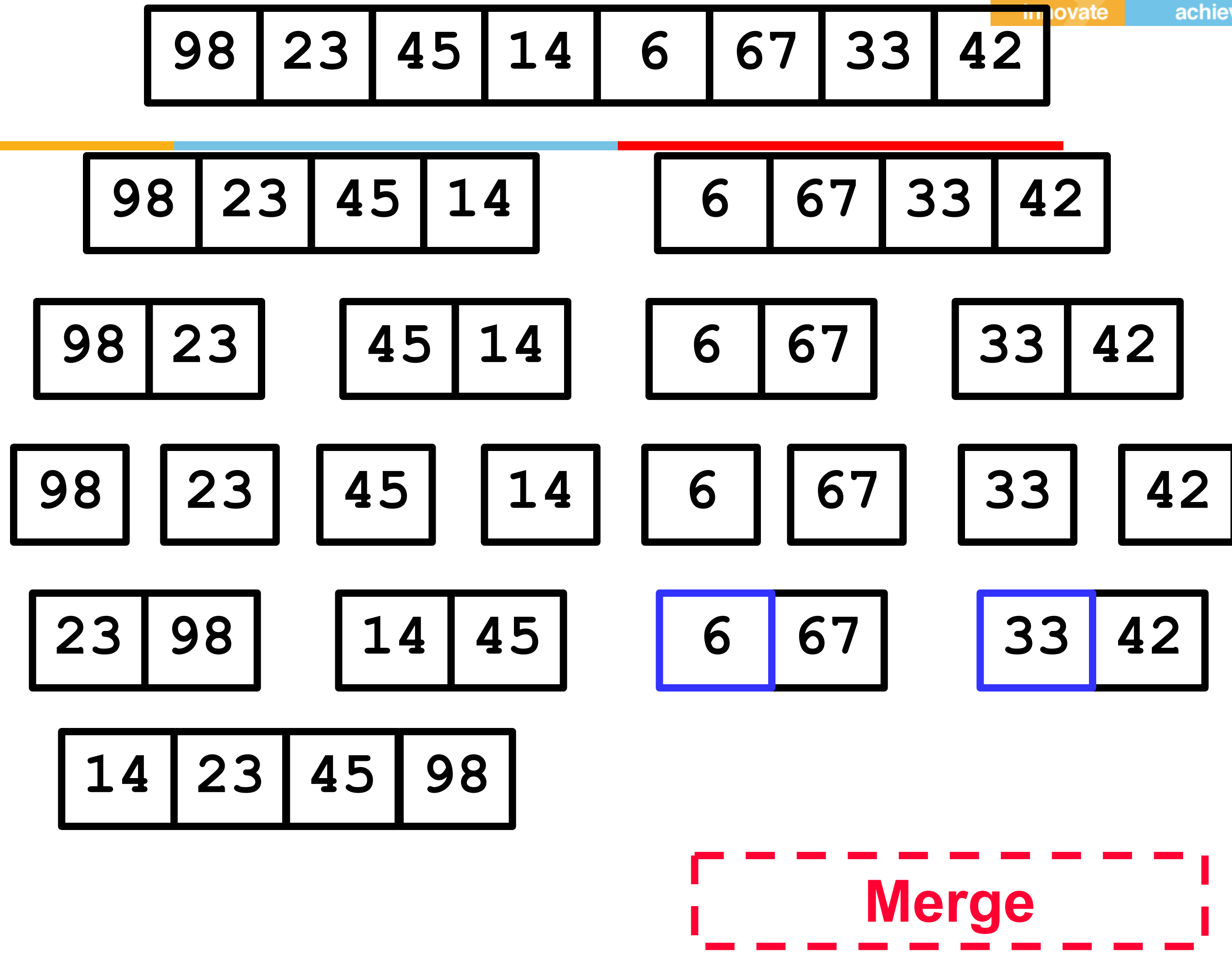

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
23 98
14 45
6 67
14 23 45 98
Merge

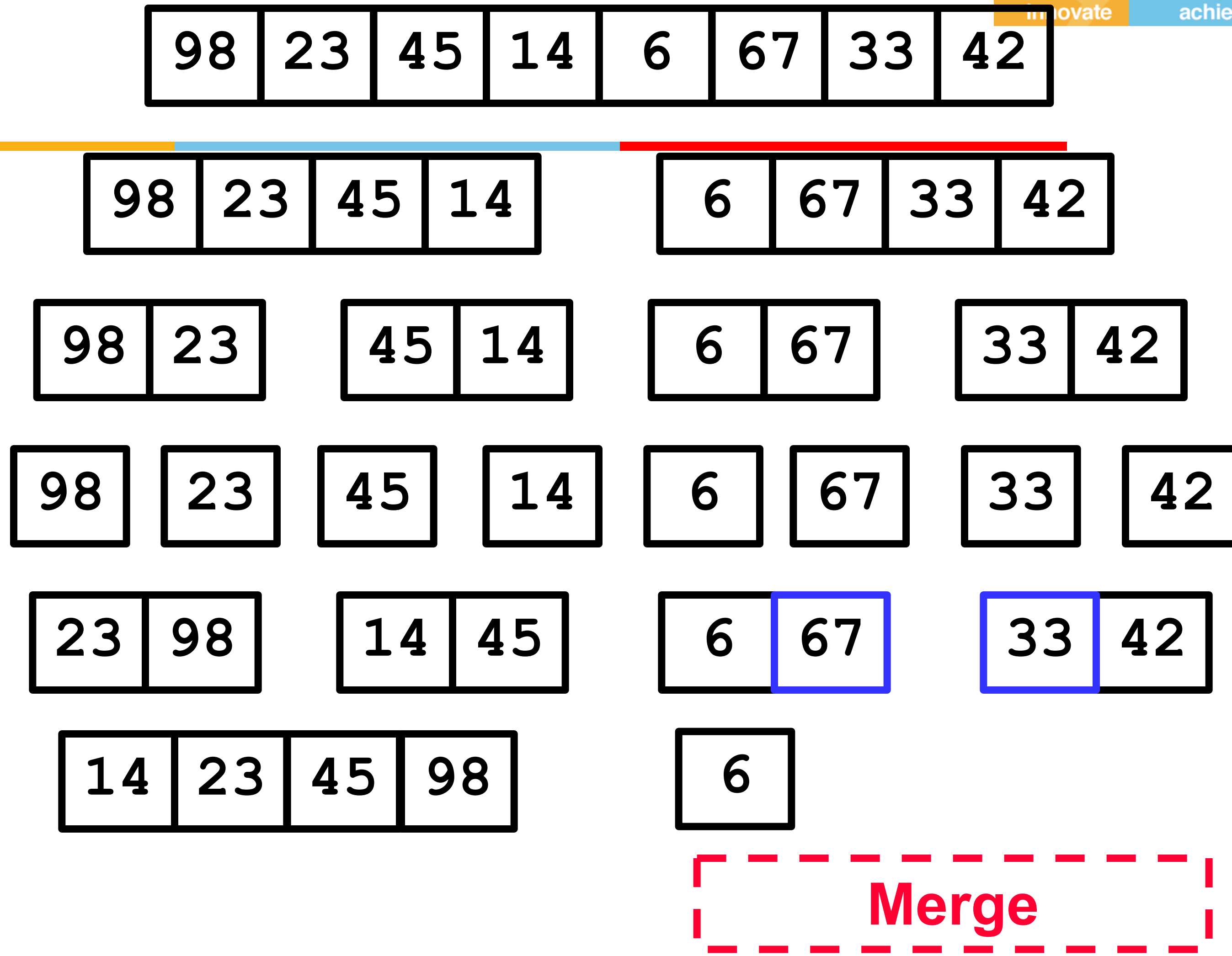

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
14 23 45 98

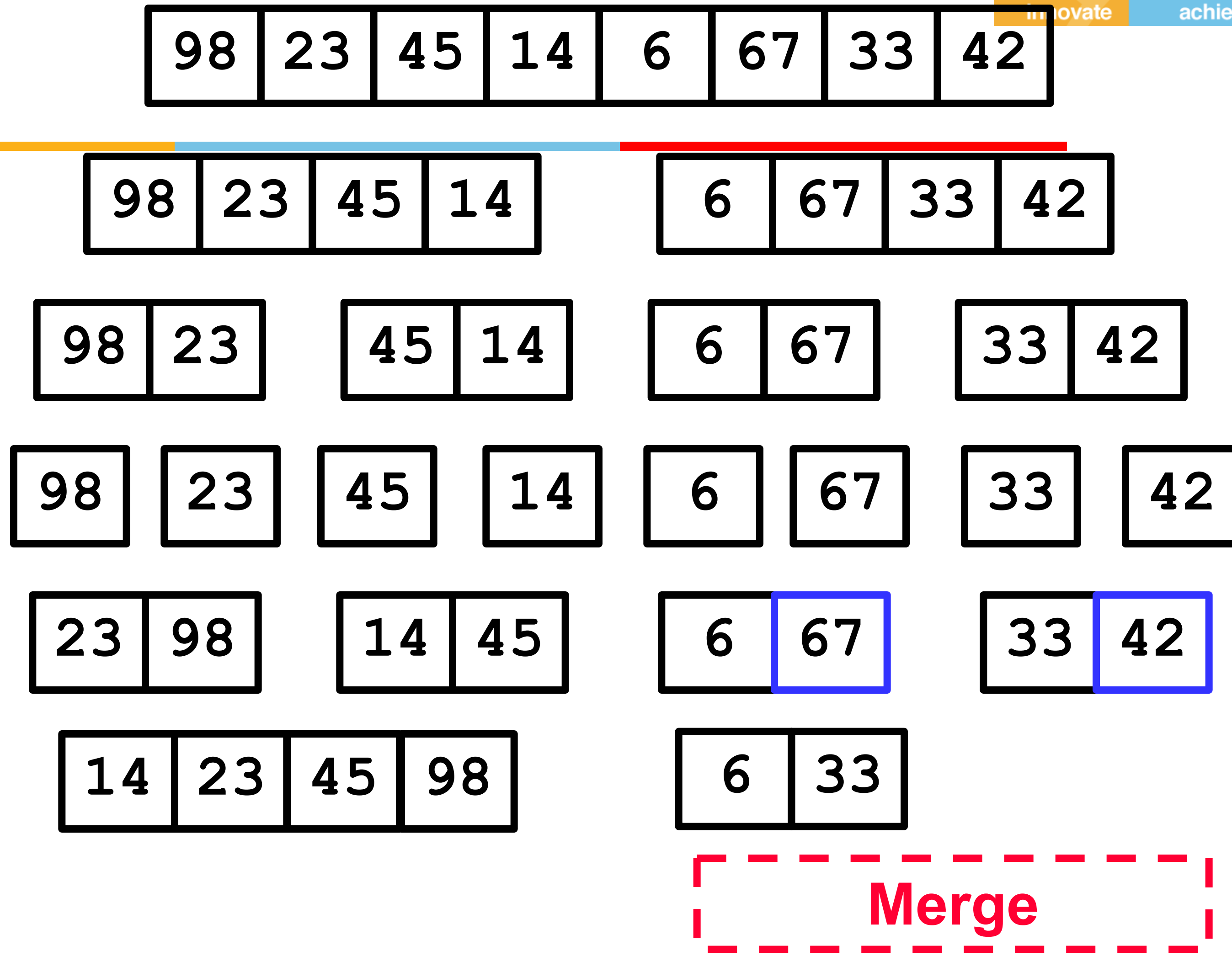

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
14 23 45 98
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33
14 23 45 98
Merge

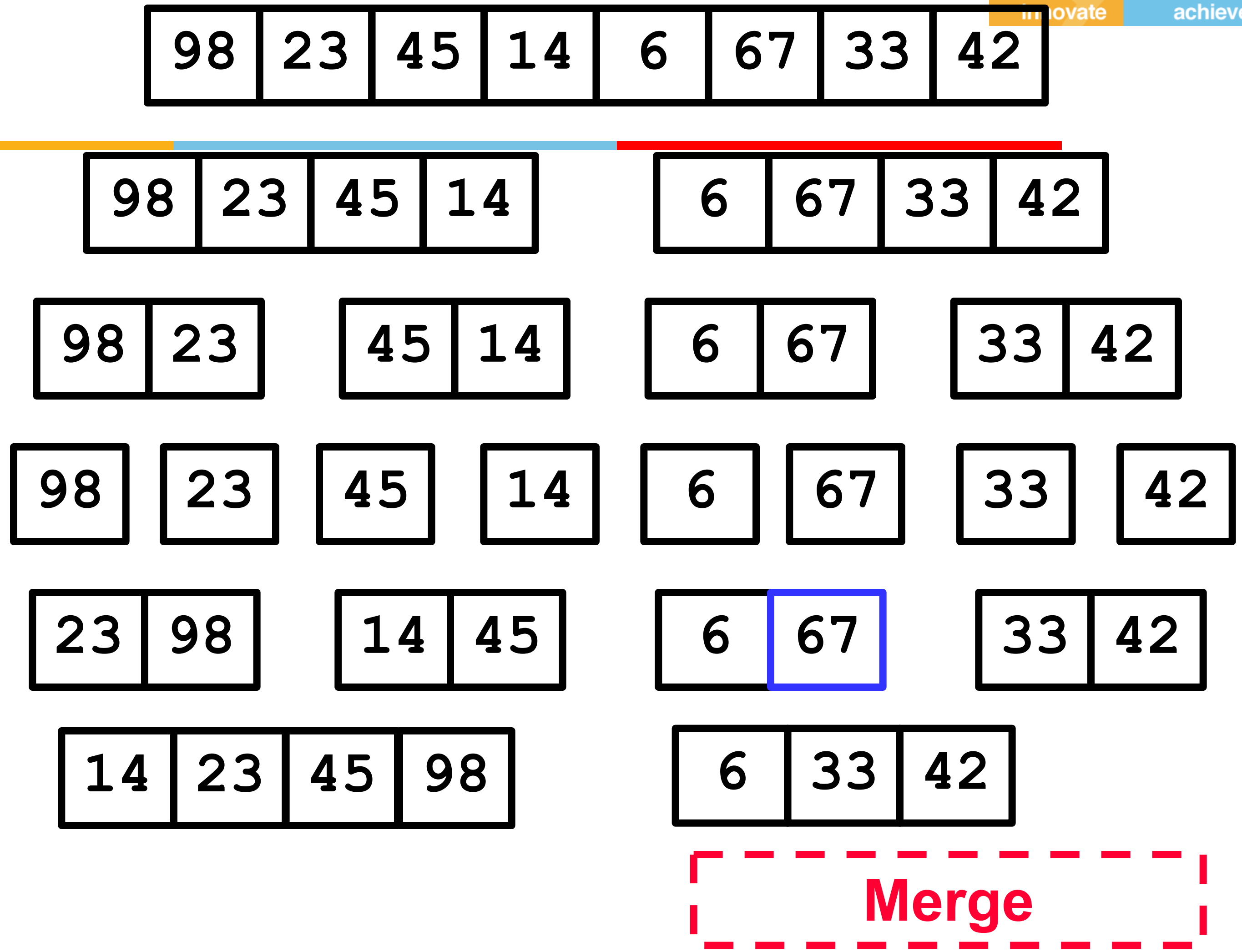

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6
Merge



98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42
Merge

98 23 45 14 6 67 33 42

98 23 45 14 | 6 67 33 42

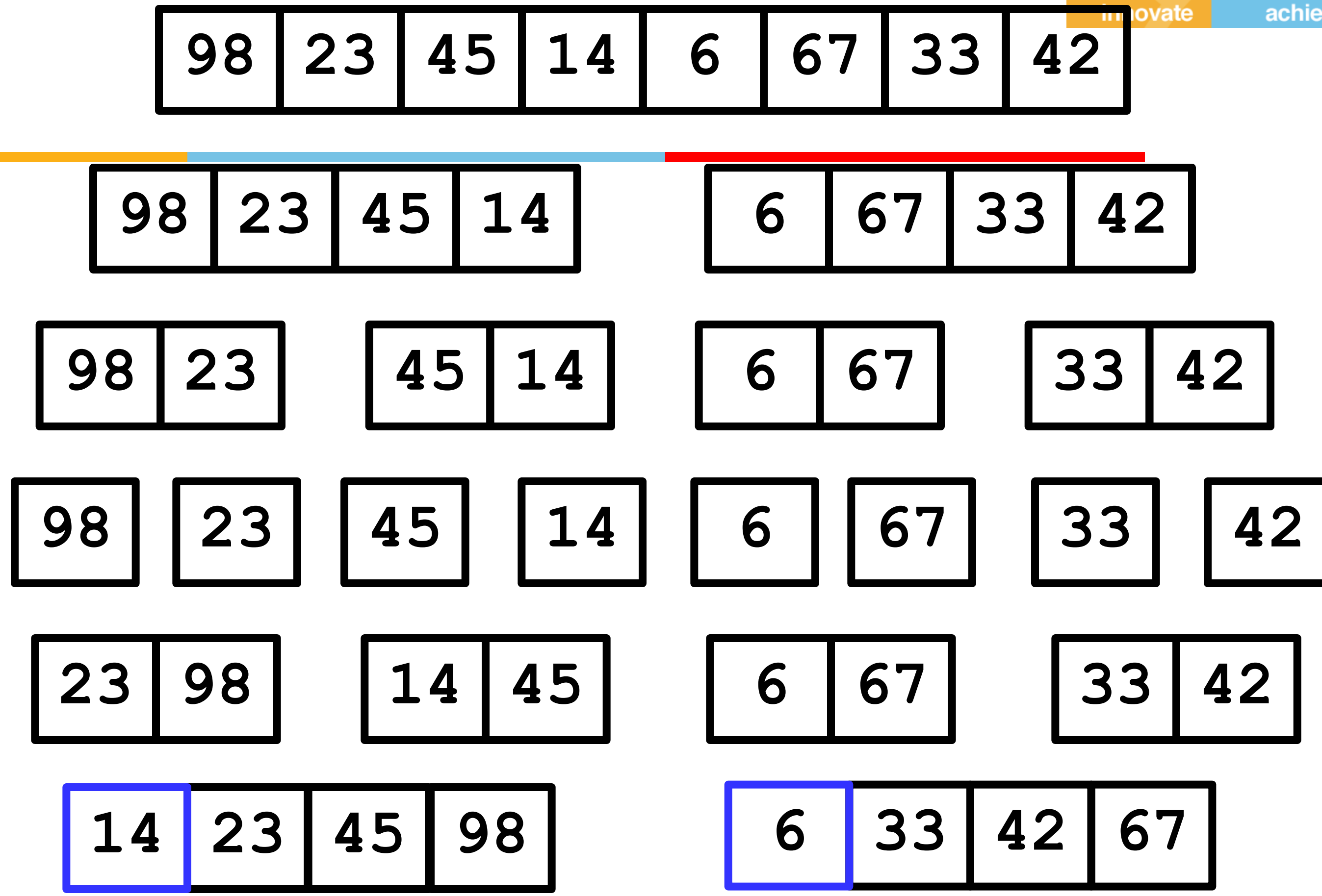
98 23 | 45 14 | 6 67 | 33 42

98 | 23 | 45 | 14 | 6 | 67 | 33 | 42

23 98 | 14 45 | 6 67 | 33 42

14 23 45 98 | 6 33 42 67

Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67

Merge

98	23	45	14	6	67	33	42

98	23	45	14

6	67	33	42

98 23   45 14   6 67   33 42

98   23   45   14   6   67   33   42

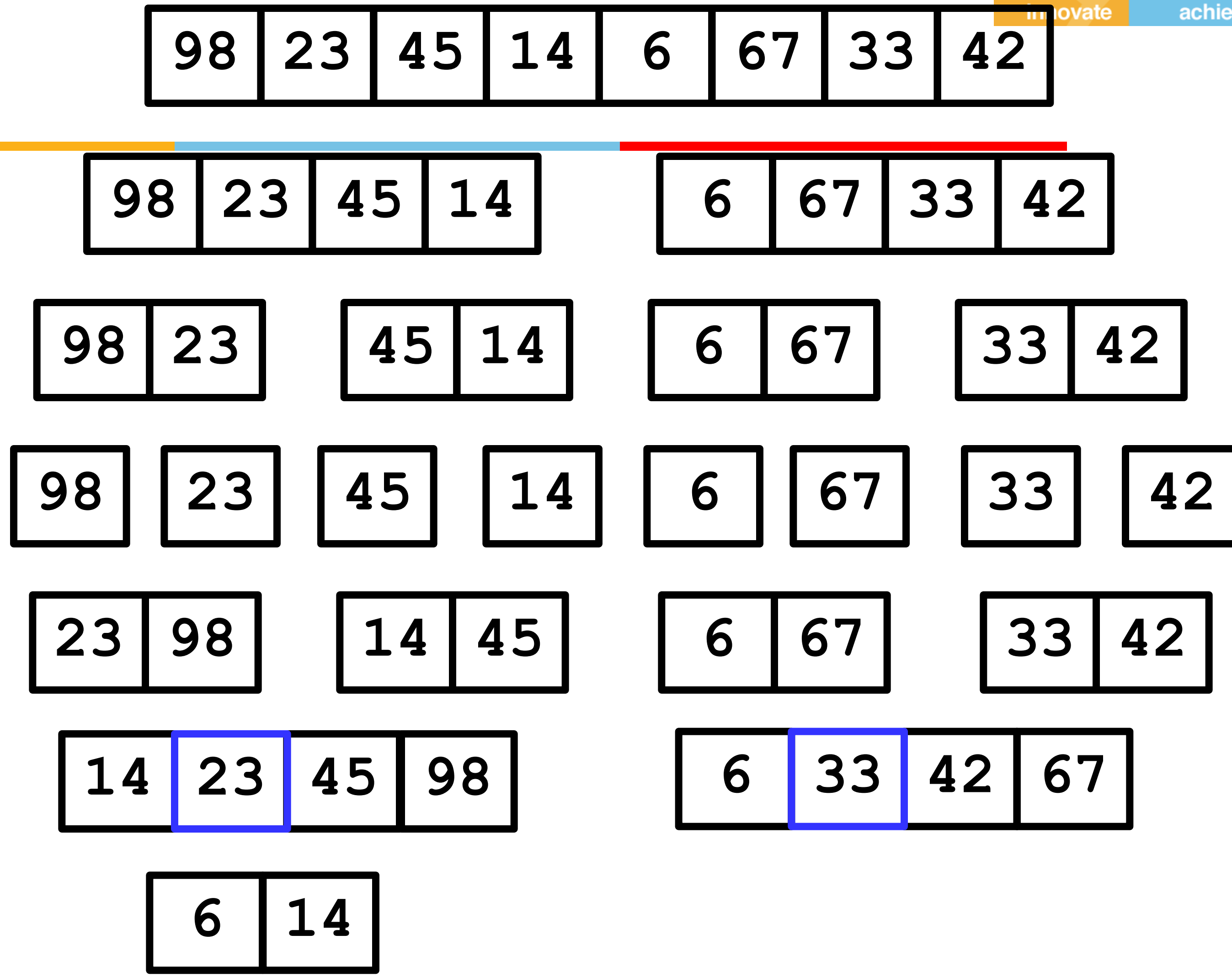
23 98   14 45   6 67   33 42

14	23	45	98

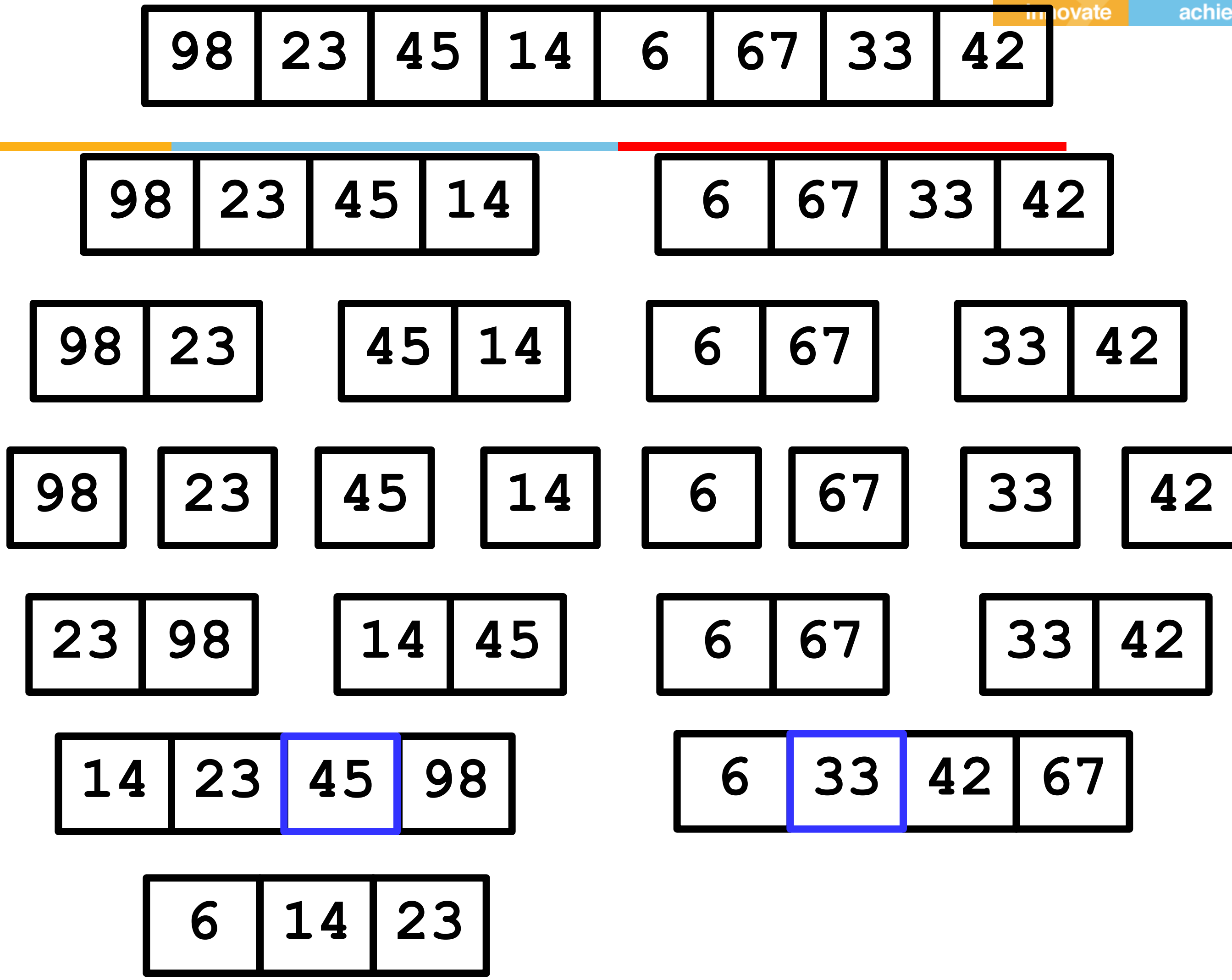
6	33	42	67

6

Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14

Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23

Merge

innovate achieve lead

98	23	45	14	6	67	33	42

98	23	45	14

6	67	33	42

98 23   45 14   6 67   33 42

98   23   45   14   6   67   33   42

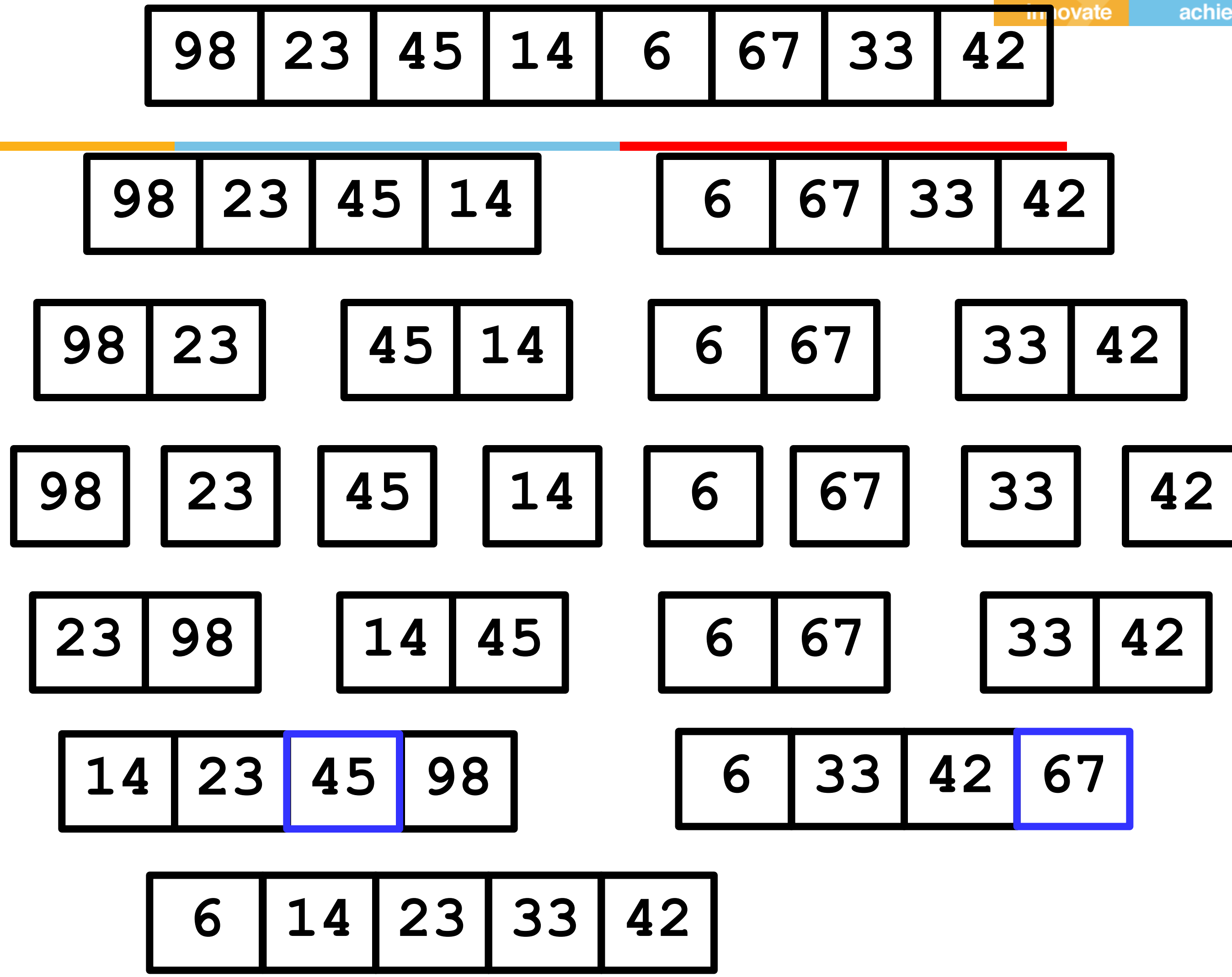
23 98   14 45   6 67   33 42

14	23	45	98

6	33	42	67

6	14	23	33

Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33 42

Merge

98	23	45	14	6	67	33	42

98 23 45 14   6 67 33 42

98 23   45 14   6 67   33 42

98   23   45   14   6   67   33   42

23 98   14 45   6 67   33 42

14 23 45 98   6 33 42 67

6 14 23 33 42 45

Merge

98	23	45	14	6	67	33	42

98	23	45	14

6	67	33	42

98	23

45	14

6	67

33	42

98   23   45   14   6   67   33   42

23	98

14	45

6	67

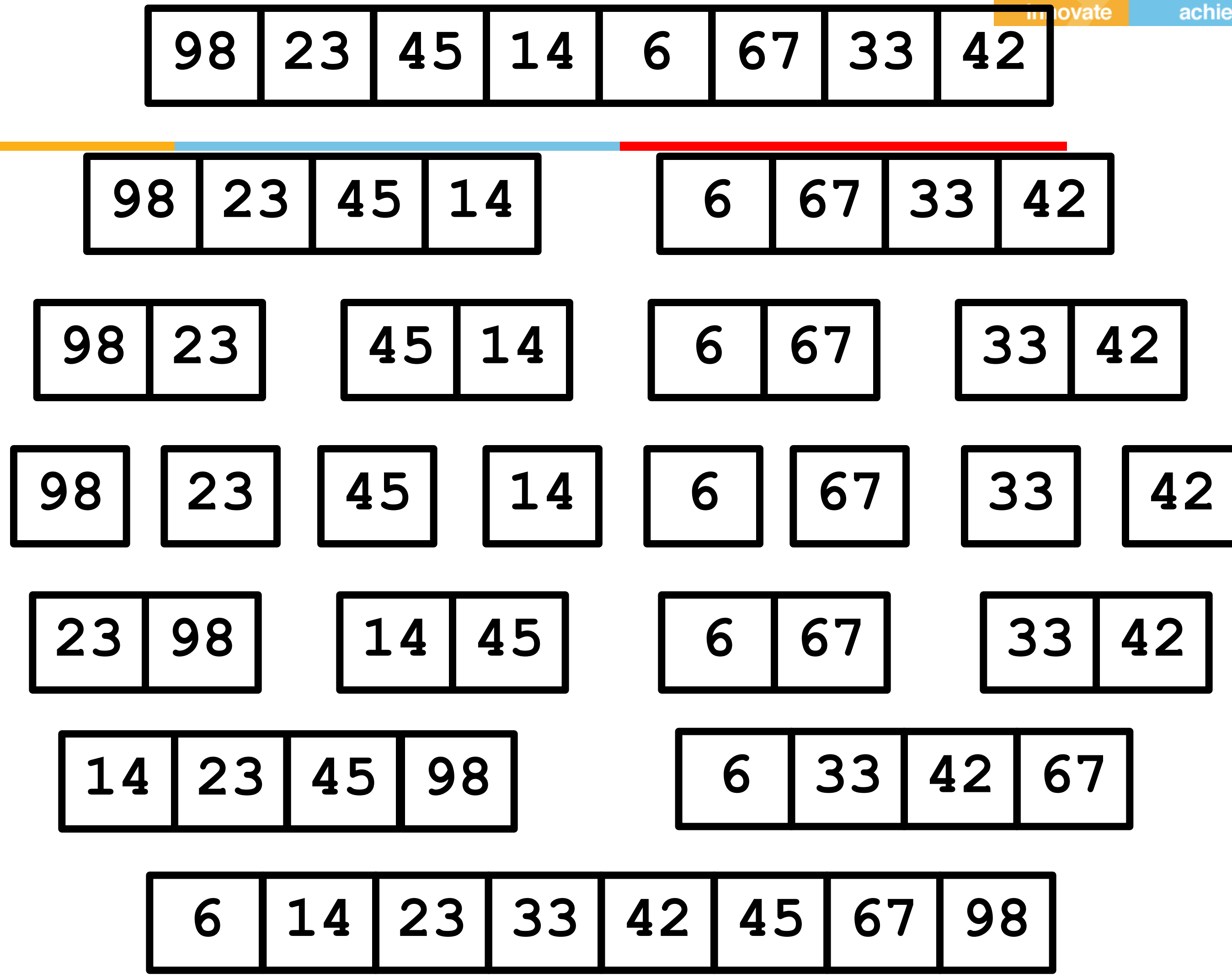
33	42

14	23	45	98

6	33	42	67

6	14	23	33	42	45	67

Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33 42 45 67 98

Merge

98	23	45	14	6	67	33	42

98	23	45	14		6	67	33	42

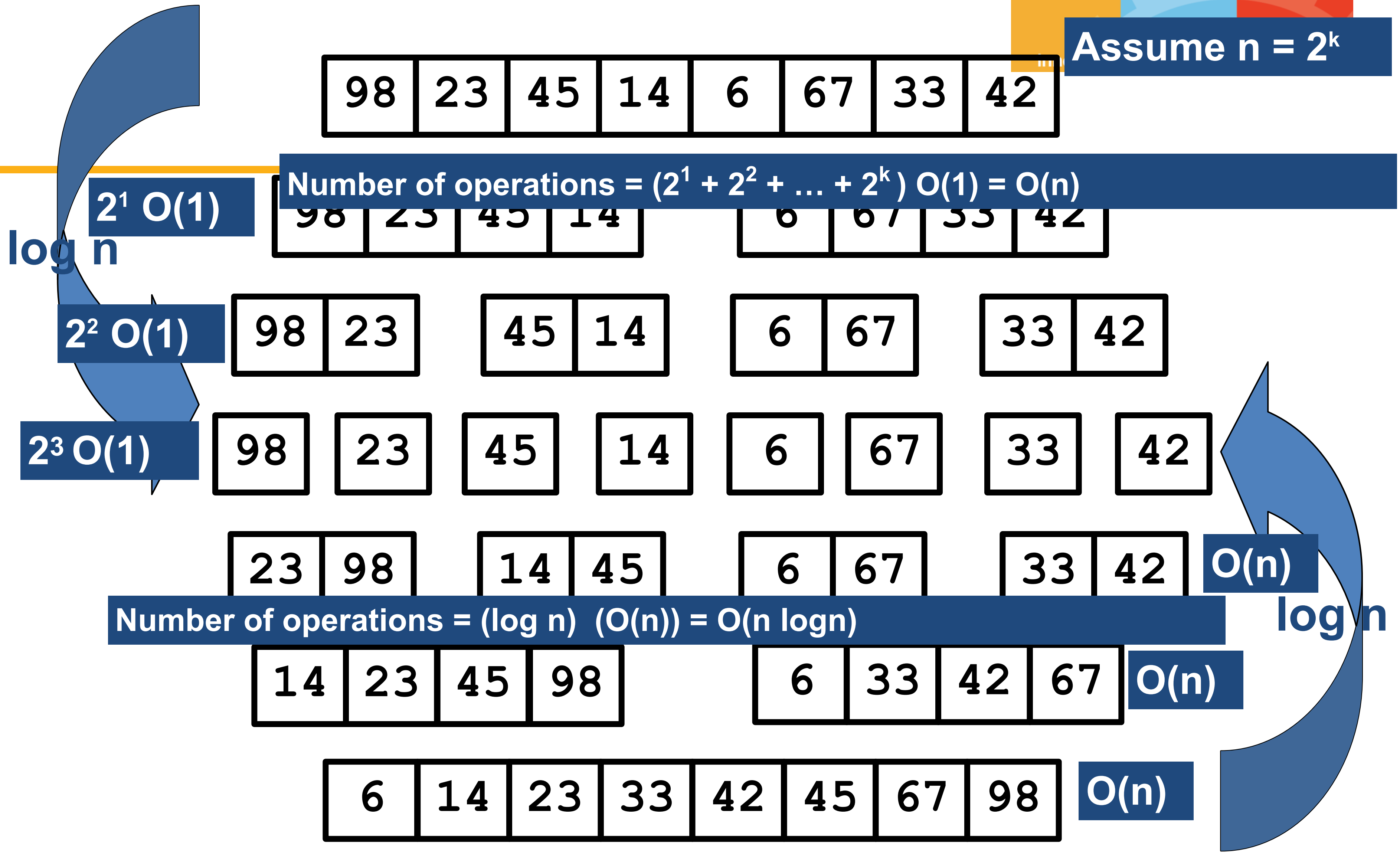
98	23		45	14		6	67		33	42

98		23		45		14		6		67		33		42

23	98		14	45		6	67		33	42

14	23	45	98		6	33	42	67

6	14	23	33	42	45	67	98


Assume n = 2^k
98 23 45 14 6 67 33 42
Number of operations = $(2^1 + 2^2 + \dots + 2^k)$ O(1) = O(n)
2^1 O(1)
98 23 45 14
6 67 33 42
log n
2^2 O(1)
98 23
45 14
6 67
33 42
2^3 O(1)
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
O(n)
Number of operations = (log n) (O(n)) = O(n logn)
log n
14 23 45 98
6 33 42 67
O(n)
6 14 23 33 42 45 67 98
O(n)

98	23	45	14	6	67	33	42

6	14	23	33	42	45	67	98

Merge Sort, another example

179 254 285 310 351 423 450 520 652 861

179 285 310 351 652 254 423 450 520 861

285 310 179 351 423 861 254 450 520

652 351 450 520

Thank You!!