Divide and Conquer Recursion + Merge Sort

Instructor: Krishna Venkatasubramanian

CSC 212

Based on slides from Prof. Carolina Ruiz's CS2223 at WPI

Announcements

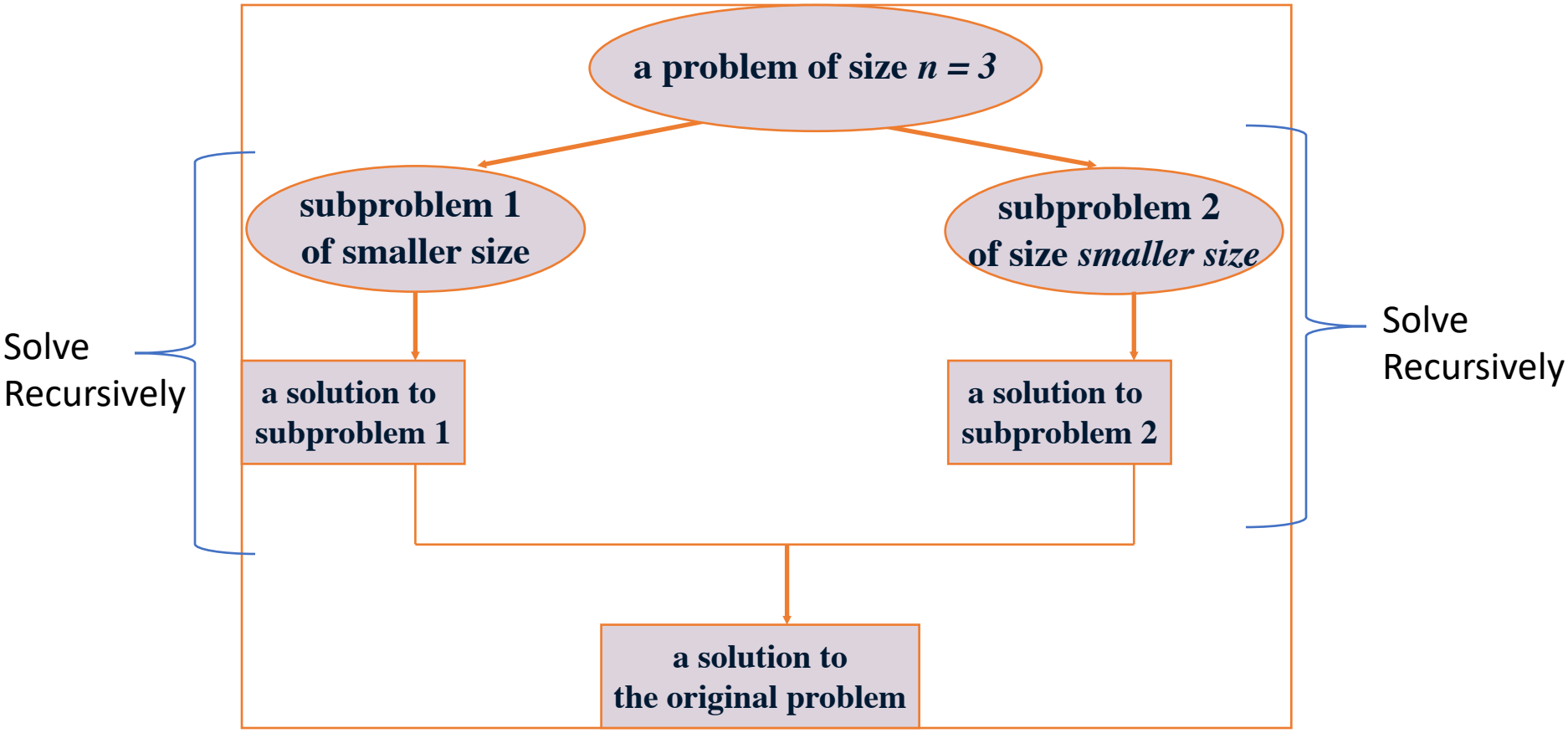
- Assignment 1 will be out on Thursday
- Quiz 2 will be next Tuesday (Oct 8)
- Slight change to office hours based on poll results and TA availability

Divide and Conquer Algorithms

- Before going to the details of today' lecture it is useful to under the concept of **Divide and Conquer Algorithms** and **Recursion**

- Divide and Conquer Algorithms
  - **Divide:** Break the larger problem into sub-problems that are smaller instances of the same problem
  - **Conquer:** the sub-problems are solved ***recursively***
    - If the sub-problem is really small, then solve in a straight-forward manner
  - **Combine:** combine the solutions of the sub-problems to find the solution of the original problem!

Visually Speaking!



Recursive Problem Solving

- What is Recursion?
  - A method of solving a problem where the solution depends on **solutions to smaller instances of the same problem**
- In effect
  - It is **basically a function calling itself** with a slightly smaller set of parameters
  - Can you see why Divide and Conquer Algorithms use it?
- **Recursive Algorithms MAY NOT be intuitive and need a lot of practice to work with!**

Example: Factorial

- Factorial of a positive number p (written as n!) is
  - p! = p*p-1*p-2*…*1
  - p! = 1, if p = 0

- Non-recursive Algorithm

```
def fact(n):
     for i in range(1,n+1):
         fact = fact * i
     return fact
```

- Recursive Algorithm

```
def fact(n):
     if n == 1:
        return 1
     else:
        return n*fact(n-1)
```

Example: Power of a Number

- Power of a number b^n (where n >=0) is defined as
  - b^n = b*b*b*b…*b (multiplied n times with itself)
  - b^n = 1, if n = 0

What will this be?

- Non-recursive Algorithm

```
def power(b,n):
      power = ___
      for i in range(n):
          power = power * b
      return power
```

- Recursive Algorithm

```
def power(b,n):
      if n == 0:
         return 1
      else:
         return b*power(b,n-1)
```

The Selection Sort Algorithm

```
Selection_Sort( A ):

   for i in range(len(A)-1):

    min_id = i
    for j in range(i+1, len(A)):
        if A[min_id] > A[j]:
            min_id = j

    swap(A[i], A[min_id])
```

Non-recursive Algorithm

i is the index of the array at which we want to "correctly" populate. Initially ZERO

Recursive Algorithm

```
Selection_Sort_Rec( A, n, i ):
   if i == n:
     return
   else:
     k = find_min_index(A, n, i)
     if k != i:
        A[i], A[k] = A[k], A[i]
     Selection_Sort_Rec(A,n,i+1)
```

find_min_index is a function to find the min element in an list A, starting at i and ending at n. It needs to be written separately

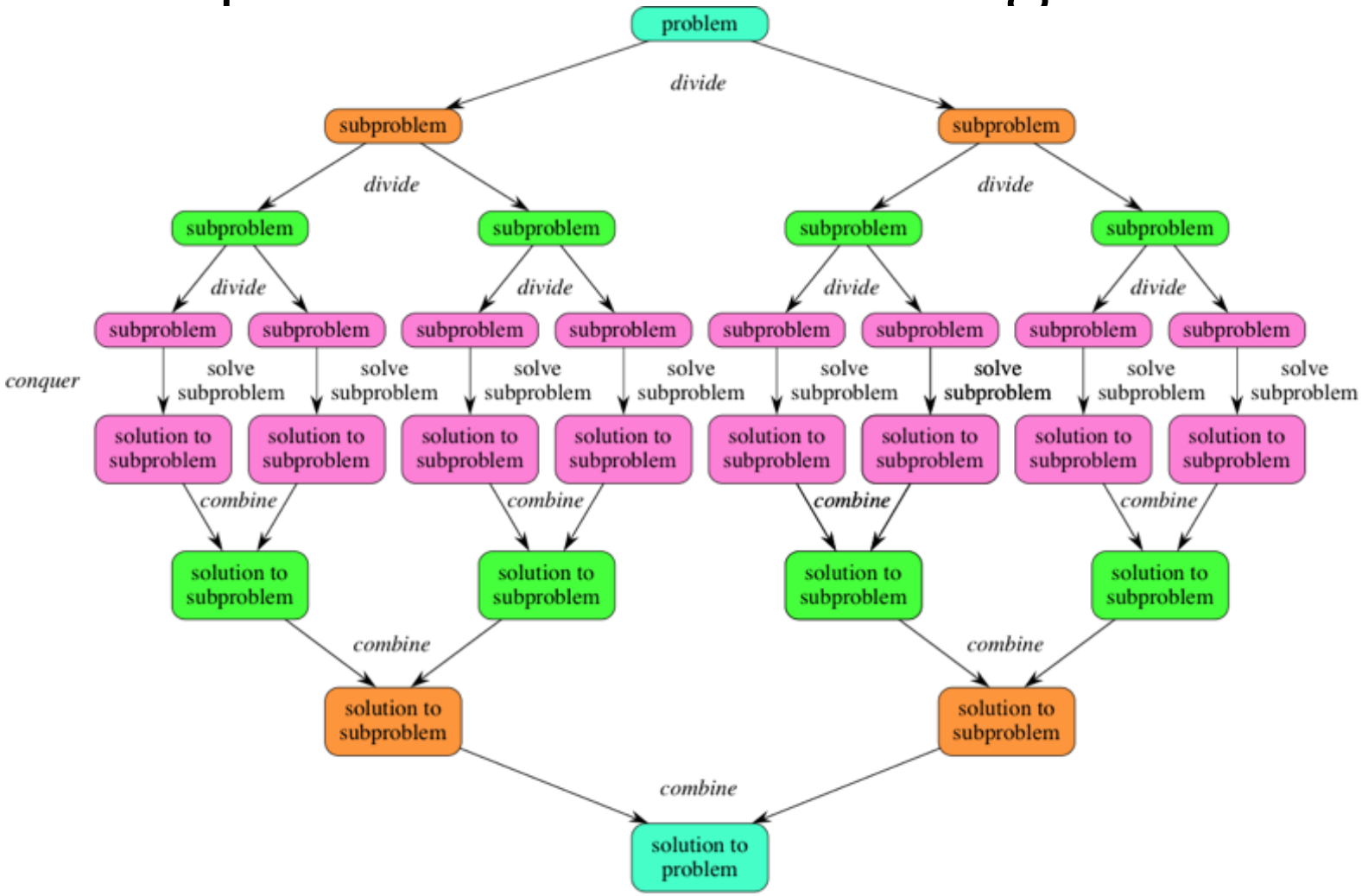
Practice: Convert a number to a String

- **Goal:** Write a function to take a number and converts it into a **string representation in any base** (i.e., binary, decimal, octal, hexadecimal
- So, the input is an integer
  - Example: 3, 17, 8, 19
- Produces an output is a **string representation of the number in a particular base**
  - Input – 3, 17, 8, 19
  - Output (Base 2) - 11, 10001, 1000, 10011 [in string format]
  - Output (Base 16) – 3, 11, 8, 13 [in string format]
- Of course do it with Recursion

Starter Code

```
def ToStr(n, base):
    str = "0123456789ABCDEF"
    if (n < base):
        return str[n]
    …
```

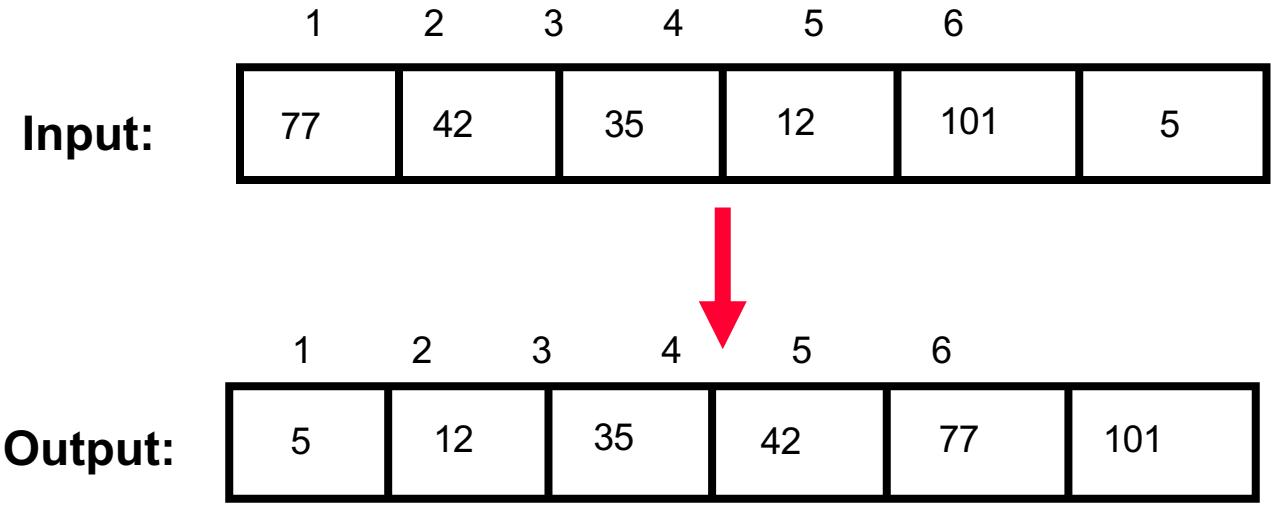
Coming back to Divide and Conquer Problem Solving



https://www.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/divide-and-conquer-algorithms

Sorting: Problem Definition

- **Sorting takes an unordered collection and makes it an ordered one.**



How can we sort an array using divide and conquer approach?

Sorting Algorithms

- *Insertion Sort --- covered already*
- *Bubble Sort --- covered already*
- *Selection Sort --- covered already*
- **Merge Sort**
- Quick Sort
- Heap Sort
- …

Merge Sort (Pseudocode)

```
MergeSort(A)
   if A's size > 1
     Divide array A in halves
     Call MergeSort on first half.
     Call MergeSort on second half.
     Merge two results (combine).
```

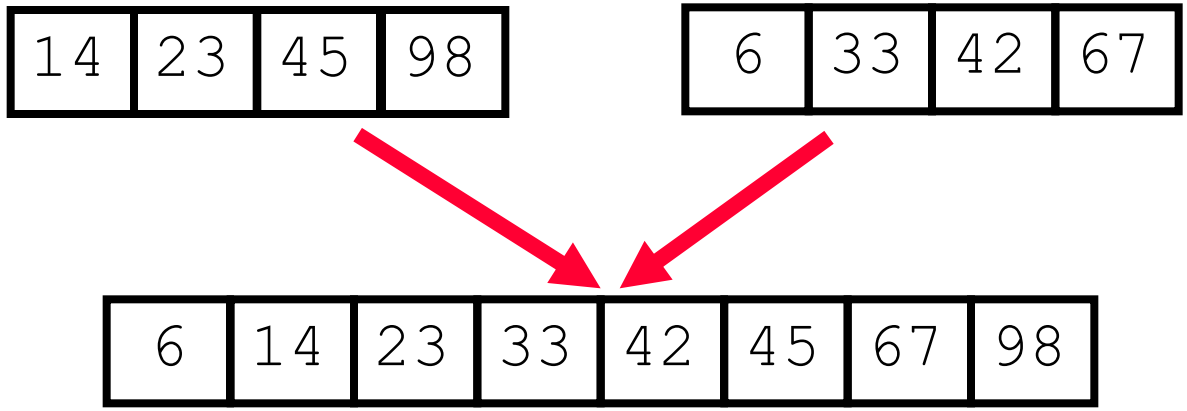
Recursion

Merge Sort Algorithm (Pseudocode)

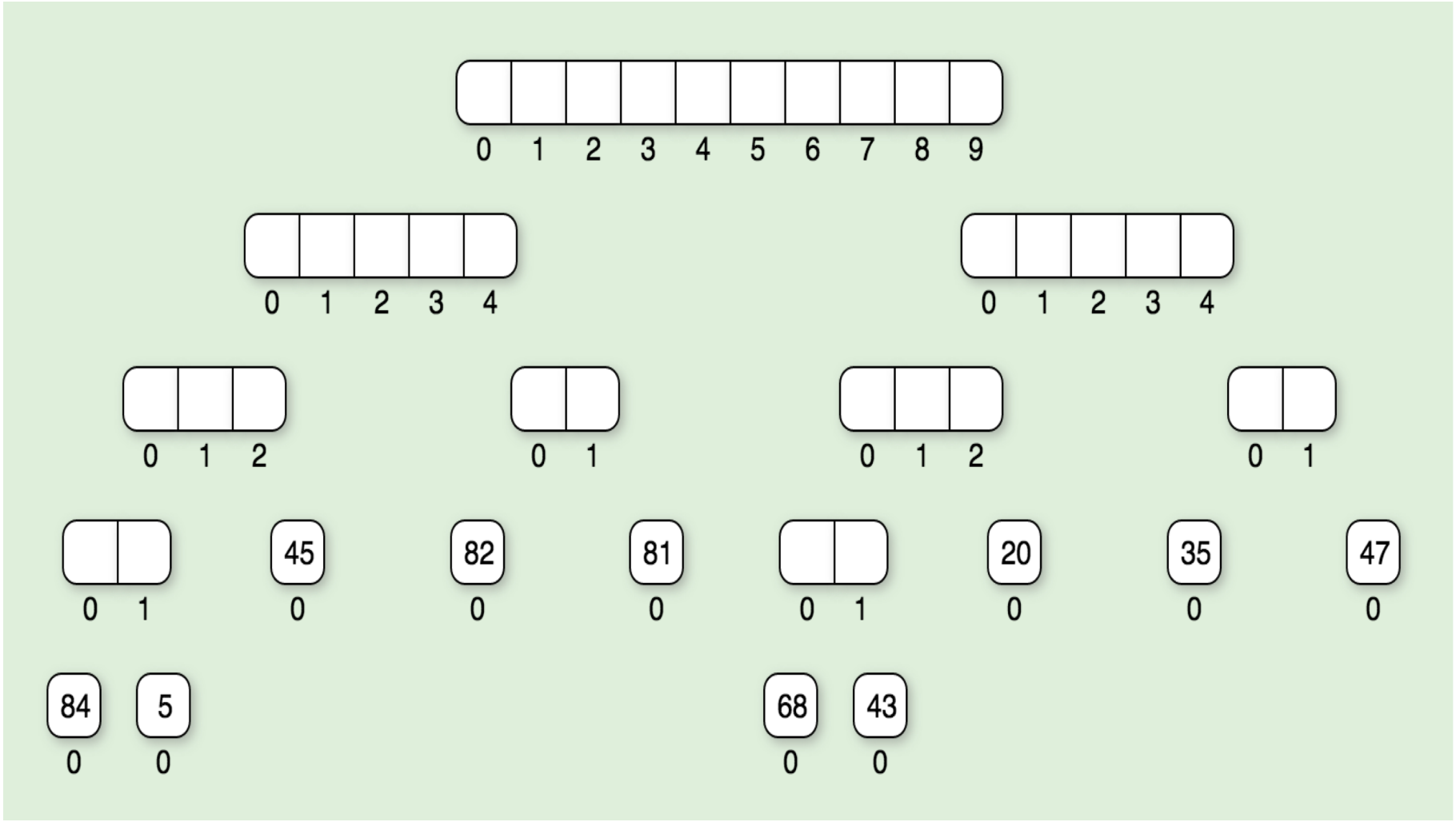
```
MergeSort(A, l, r)
   if l < r:
    m = (l + r)/2 //Divide
      MergeSort(A,l,m) //Conquer
      MergeSort(A,m+1,r)//Conquer
      Merge(A,l,m,r)// Combine
```

How to Merge?

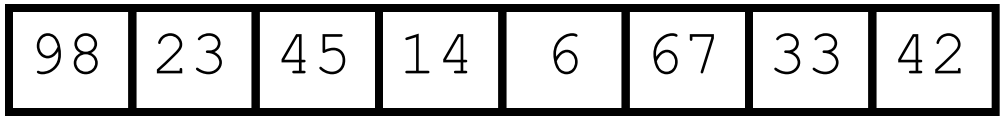
- **Merge the sub-problem solutions together:**
  - **Compare the sub-array's first elements**
  - **Remove the smallest element and put it into the result array**
  - **Continue the process until all elements have been put into the result array**



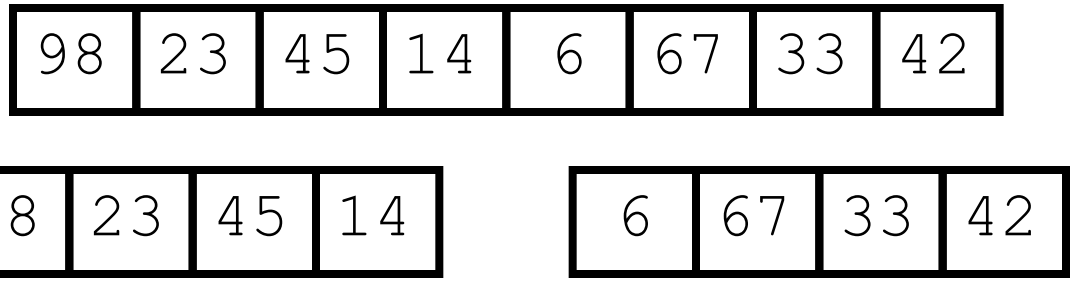
Practice -- Merging



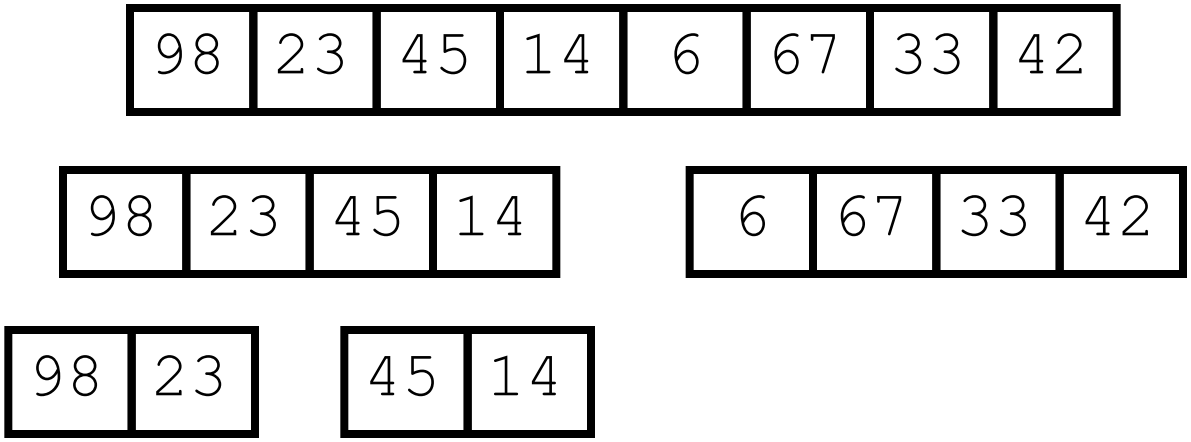
MERGE SORT EXAMPLE



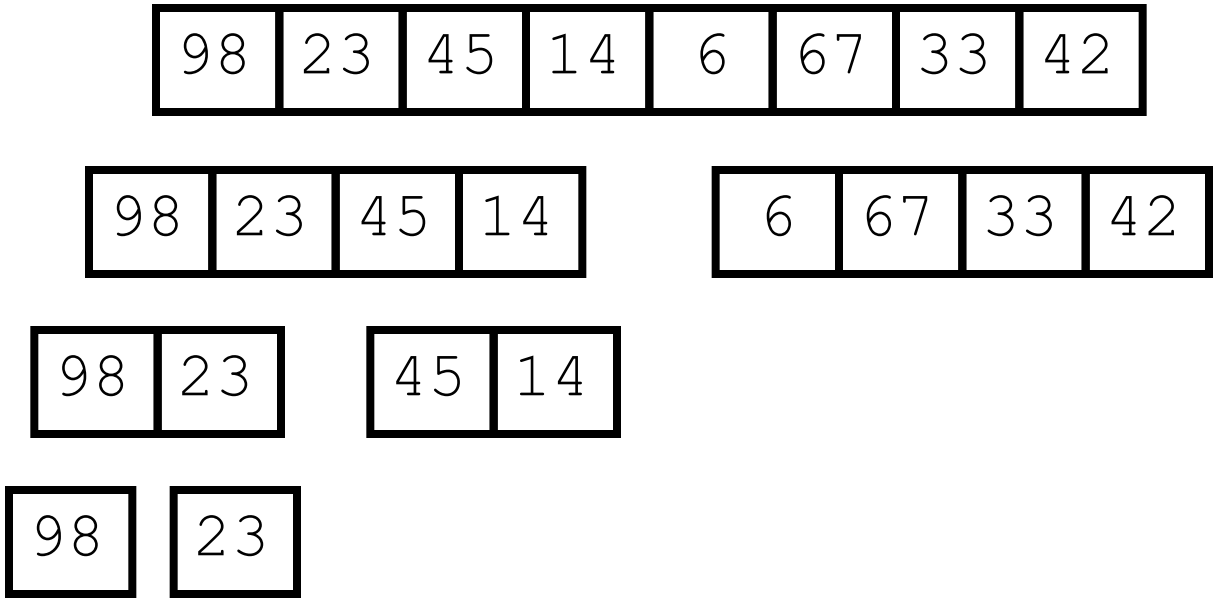
MERGE SORT EXAMPLE



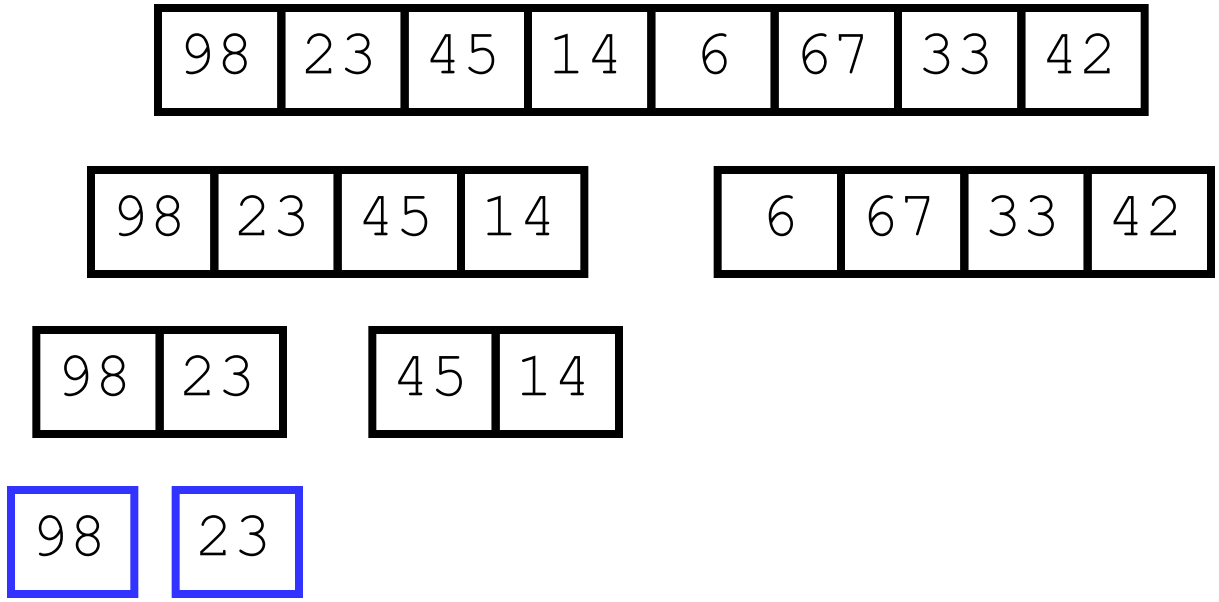
MERGE SORT EXAMPLE



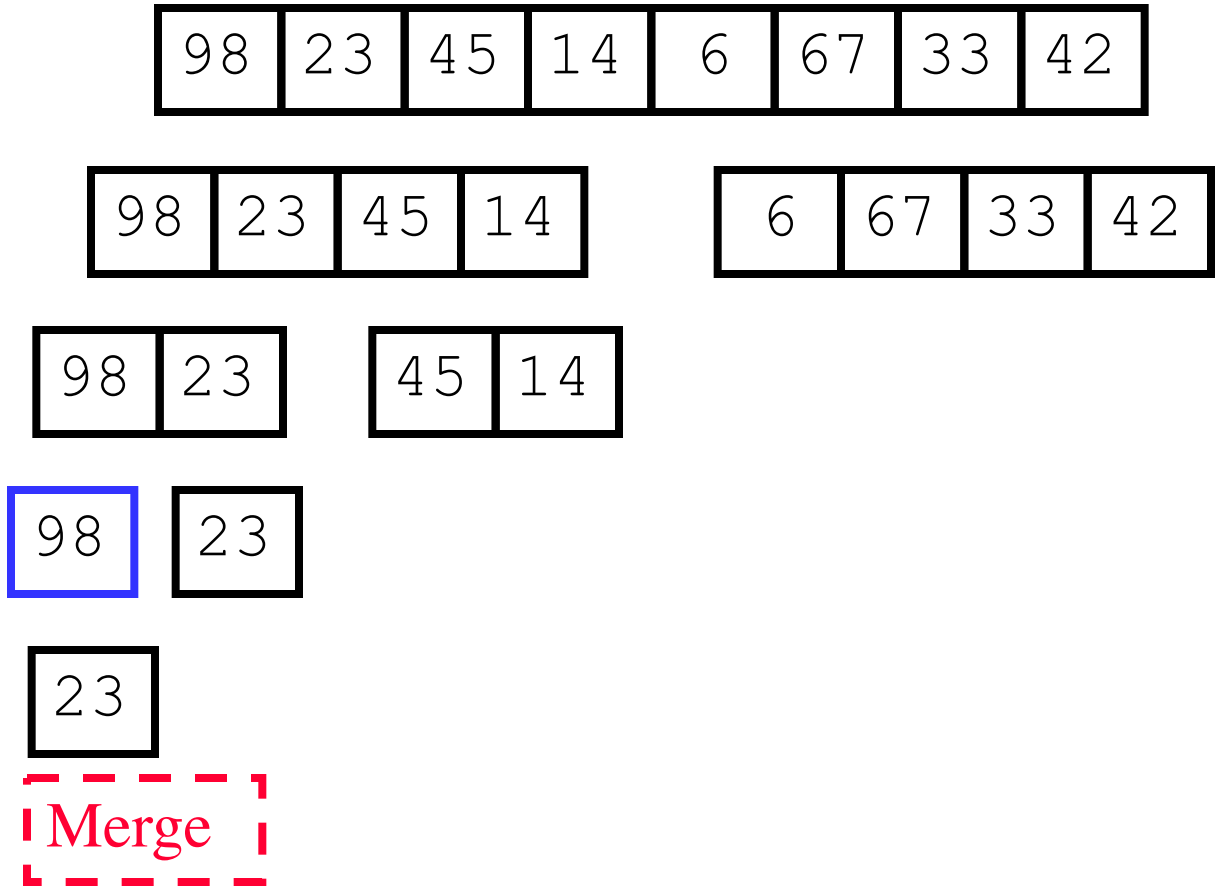
MERGE SORT EXAMPLE



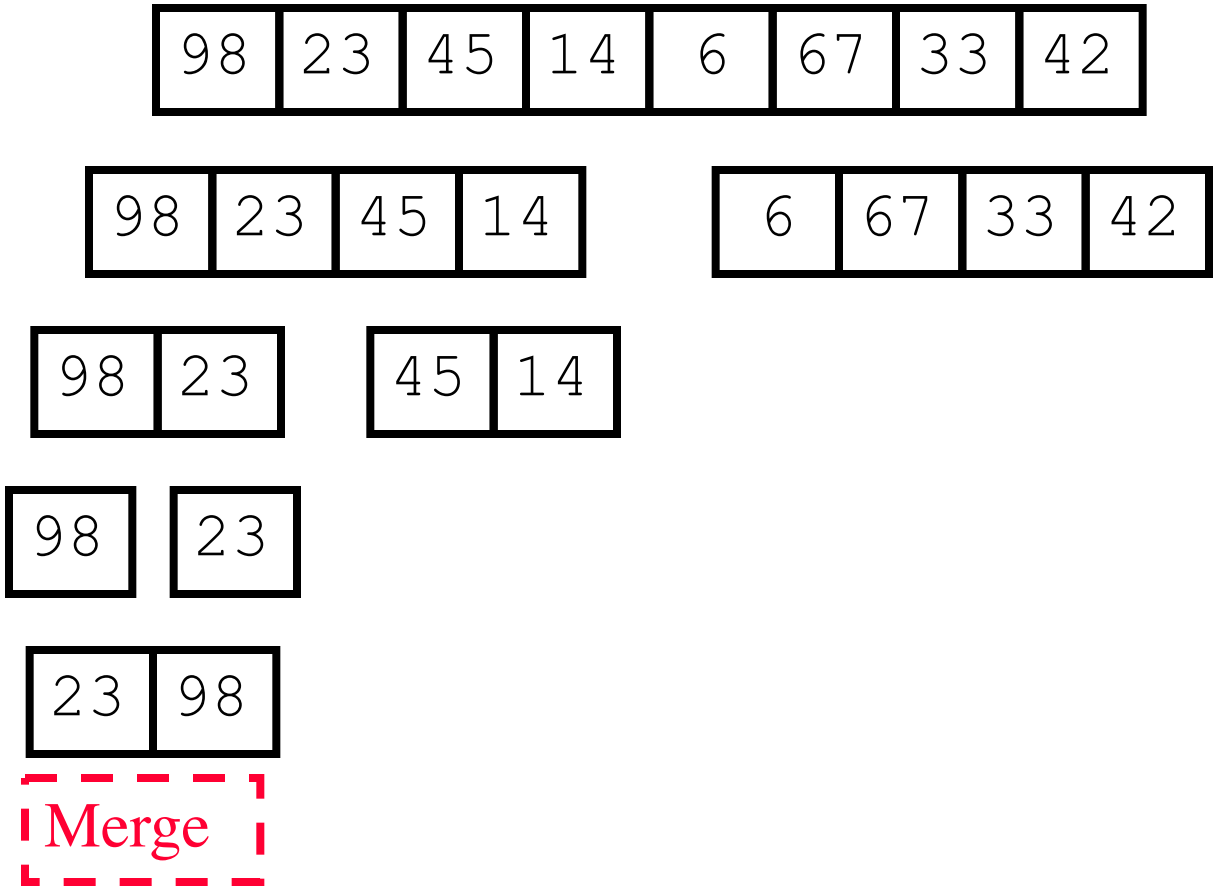
MERGE SORT EXAMPLE



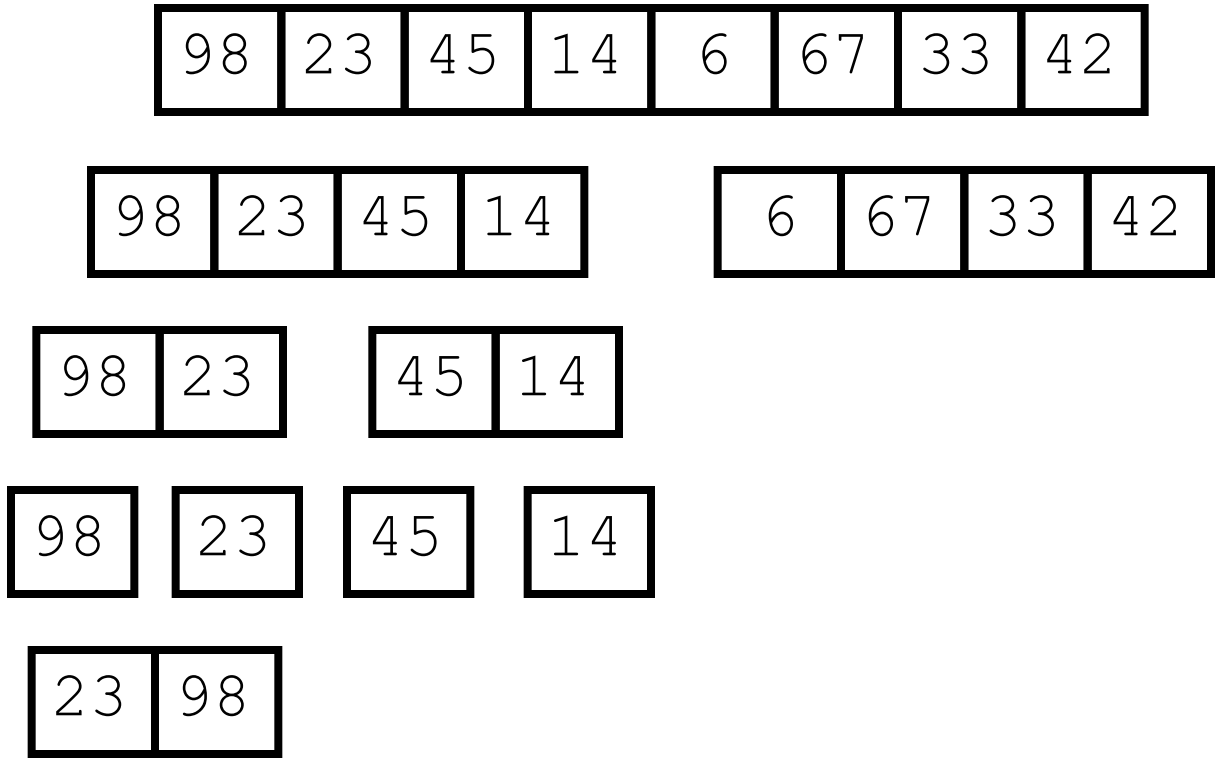
MERGE SORT EXAMPLE



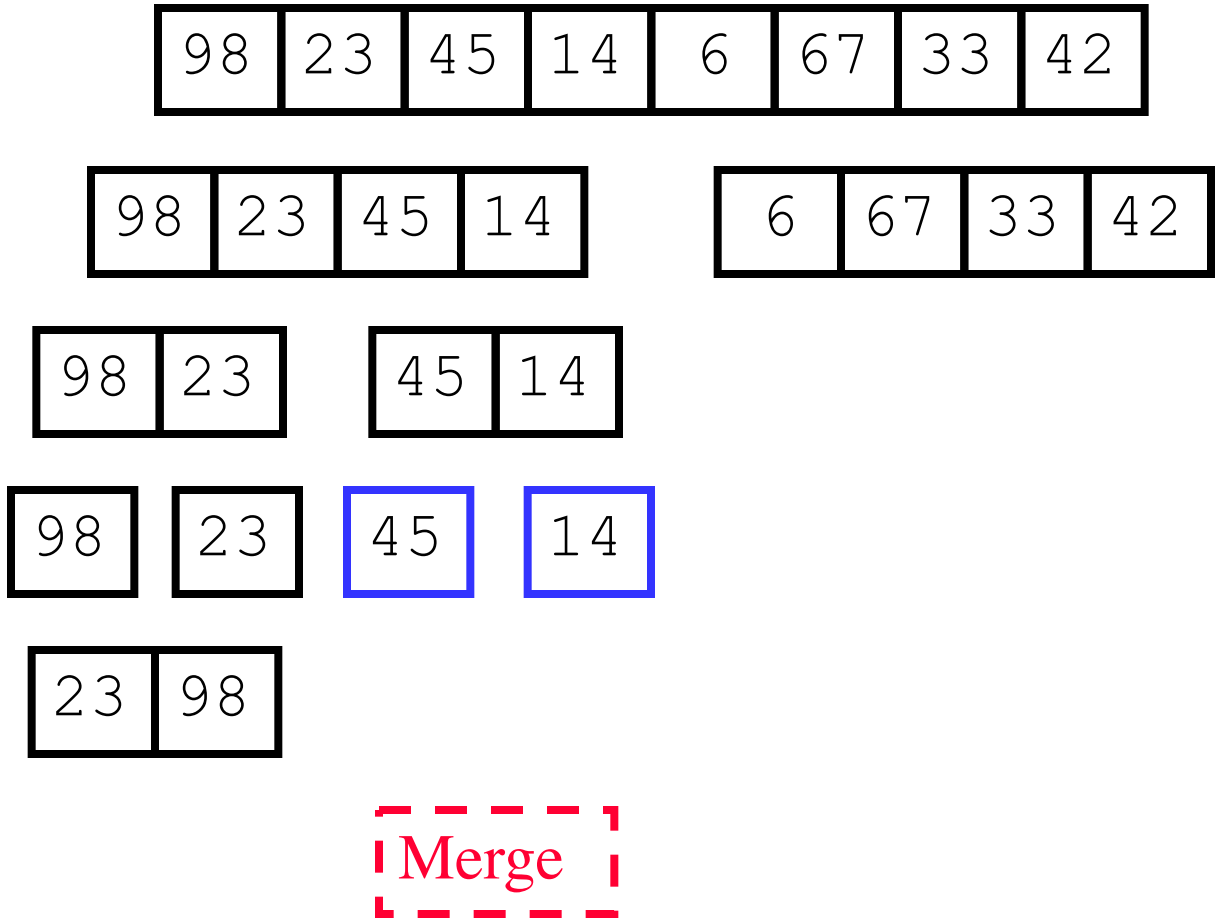
MERGE SORT EXAMPLE



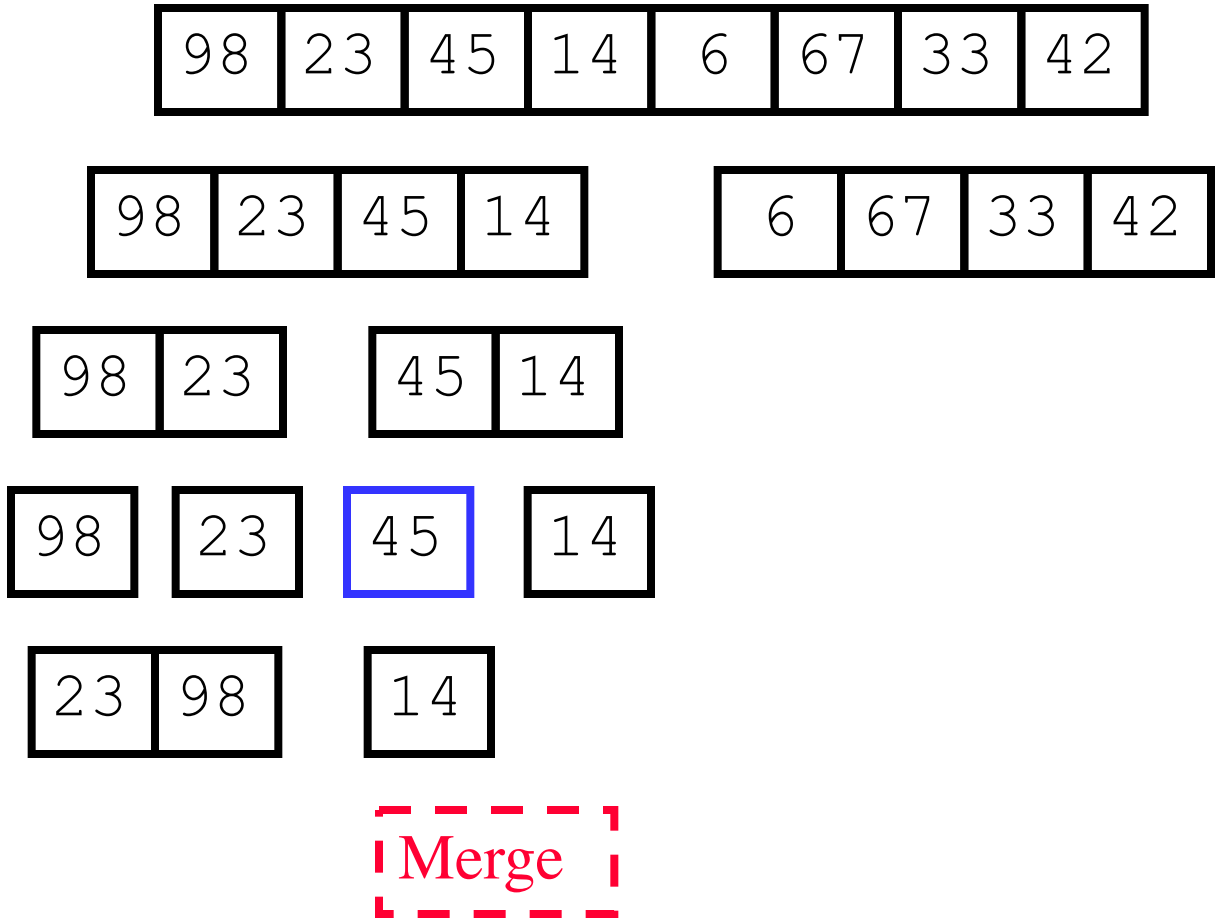
MERGE SORT EXAMPLE



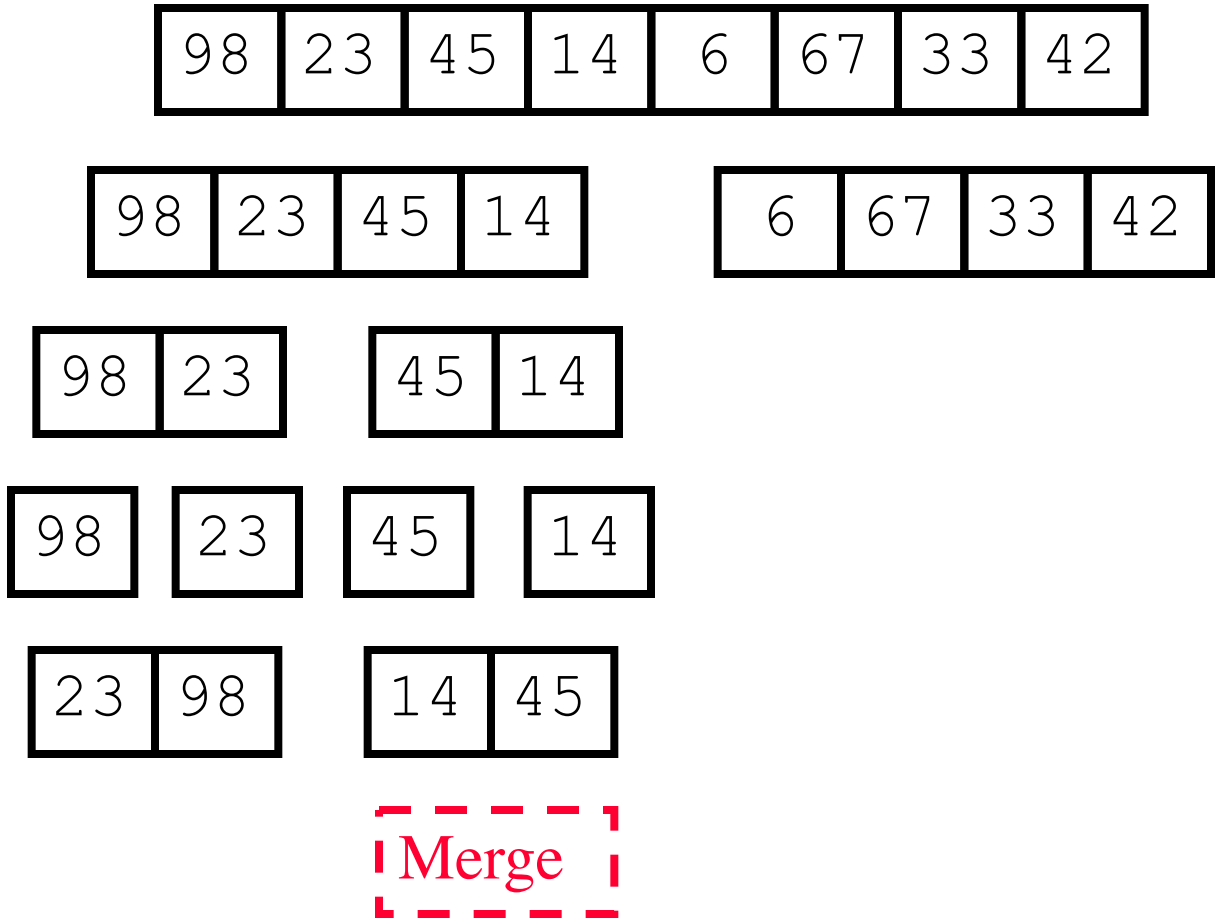
MERGE SORT EXAMPLE



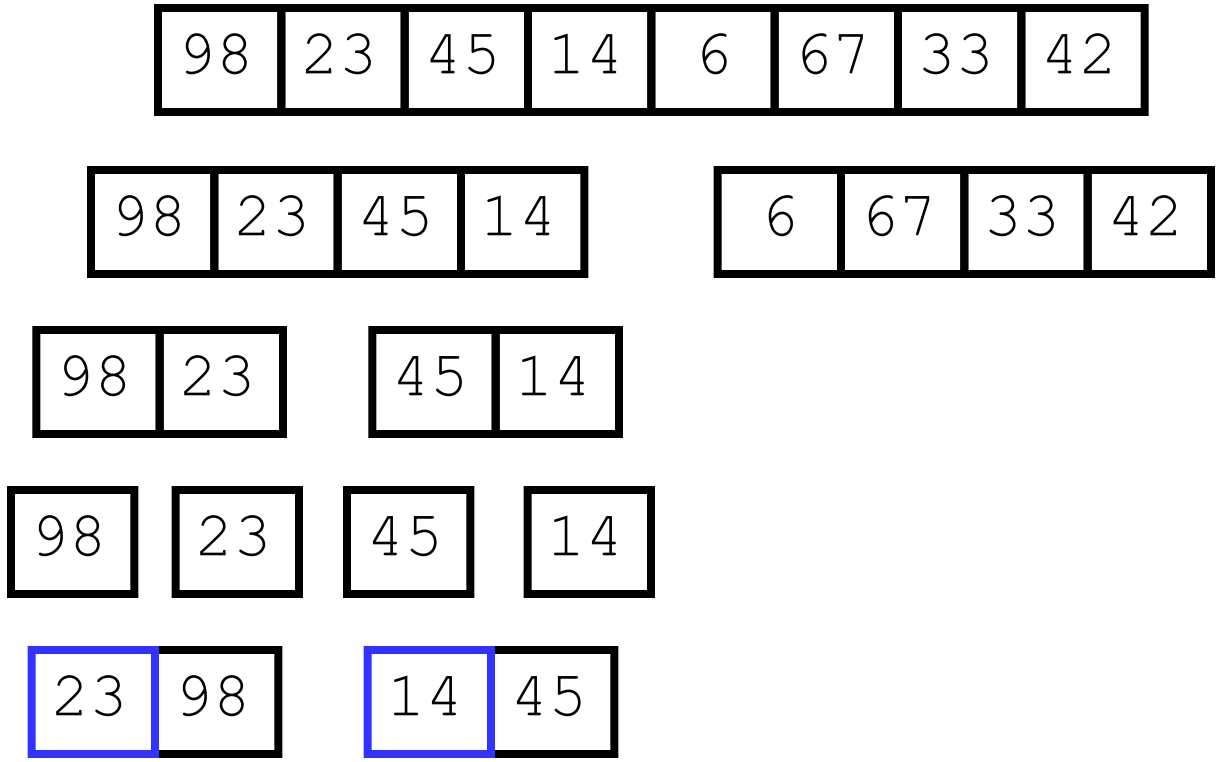
MERGE SORT EXAMPLE



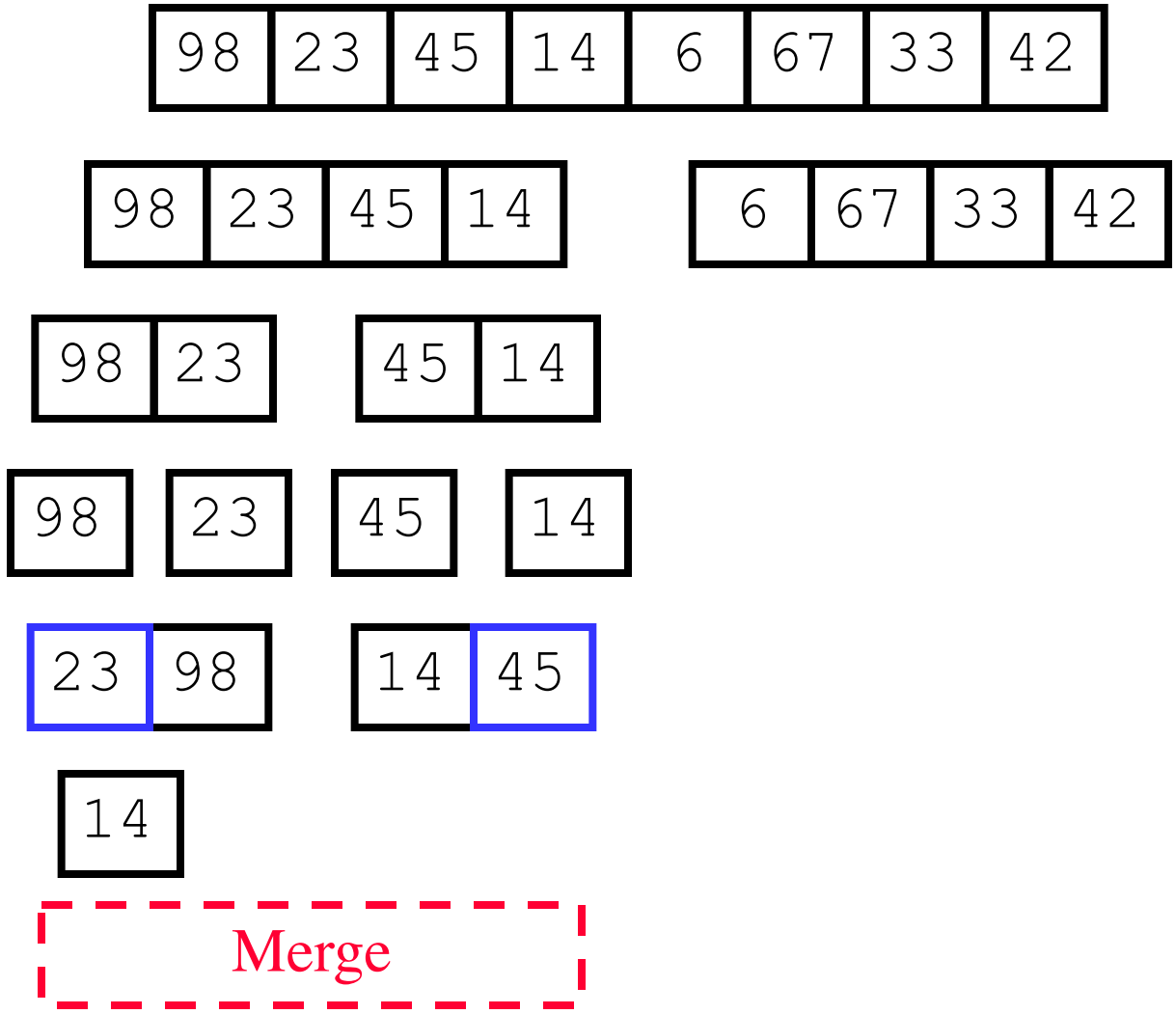
MERGE SORT EXAMPLE



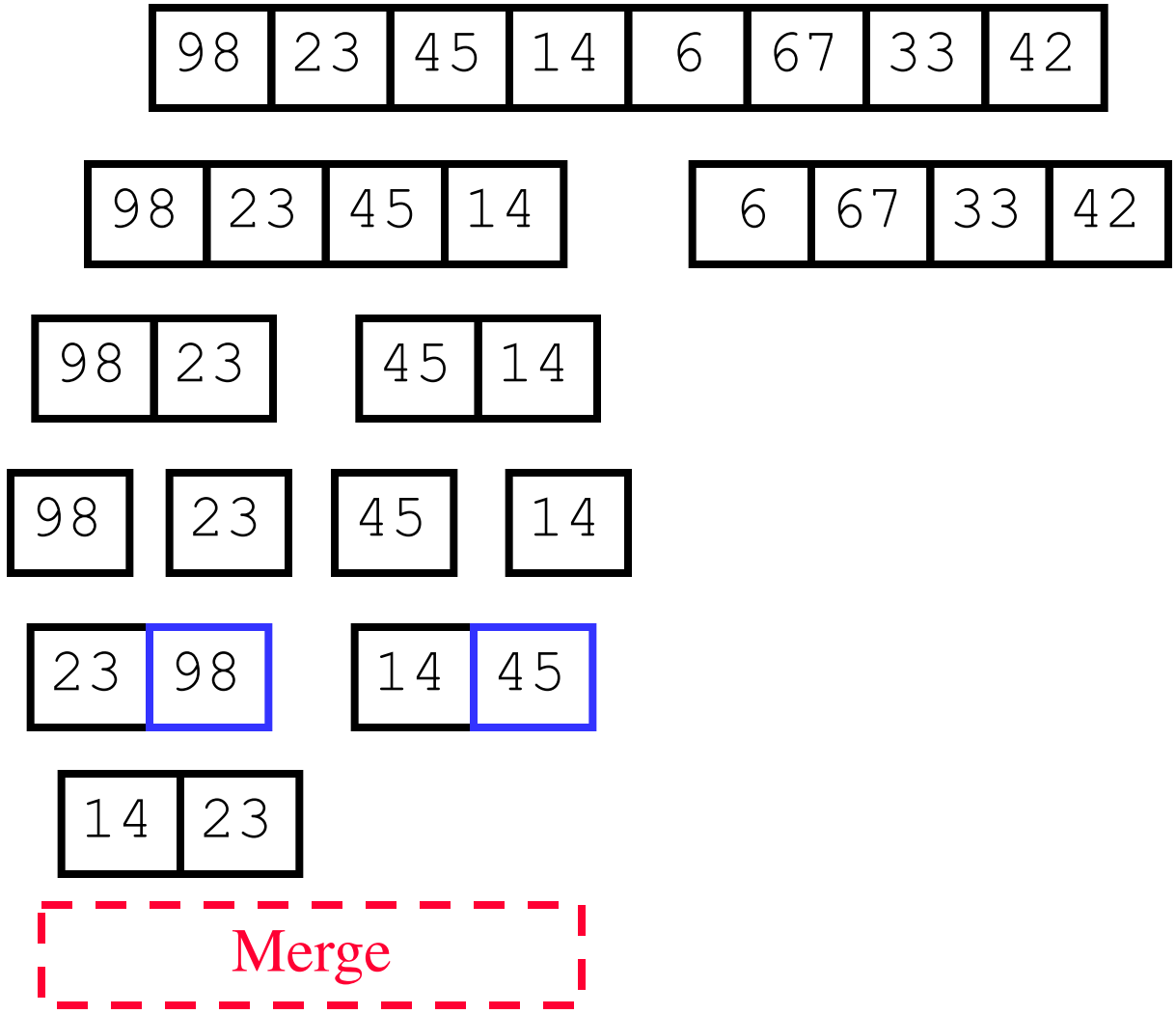
MERGE SORT EXAMPLE



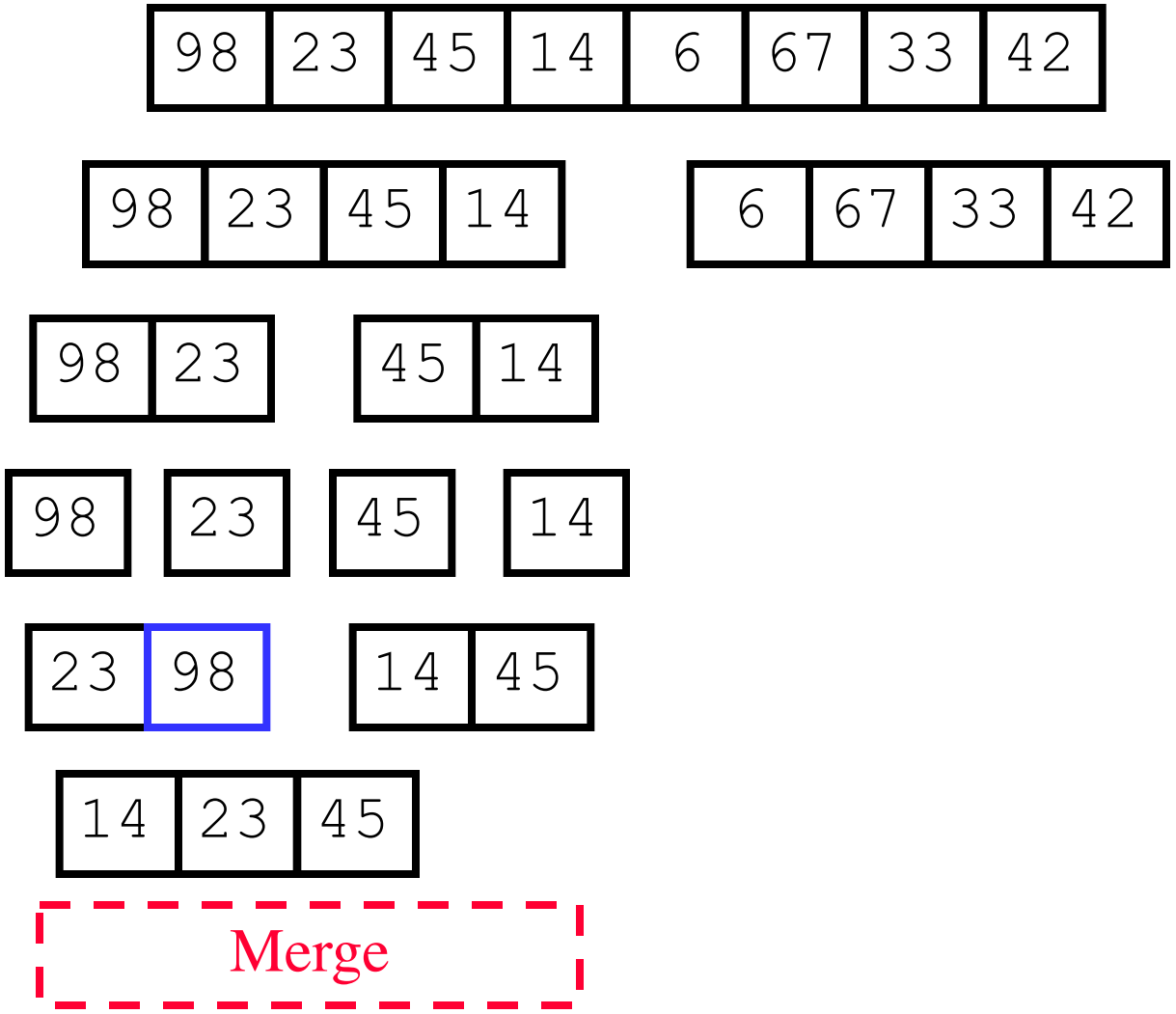
MERGE SORT EXAMPLE



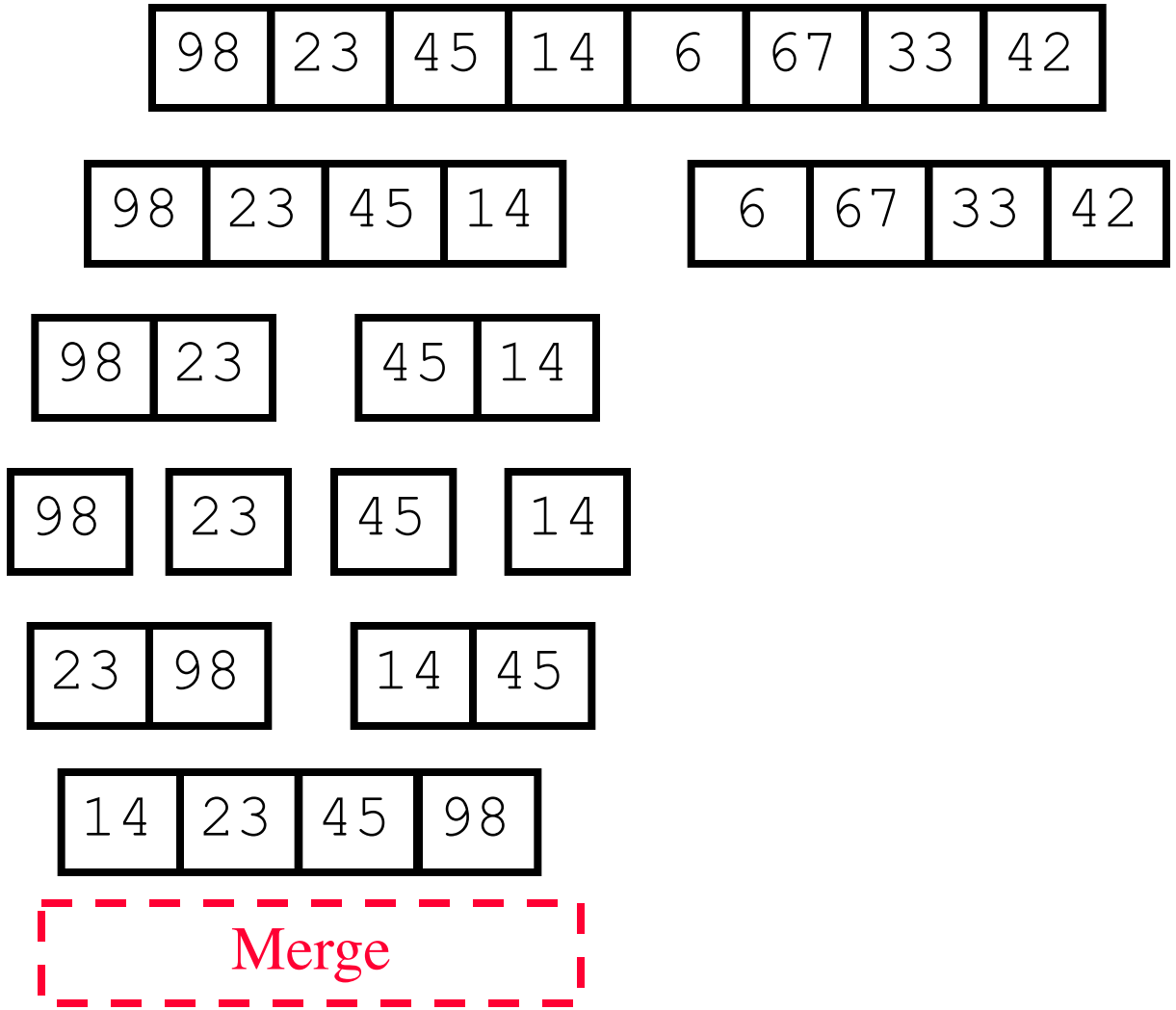
MERGE SORT EXAMPLE



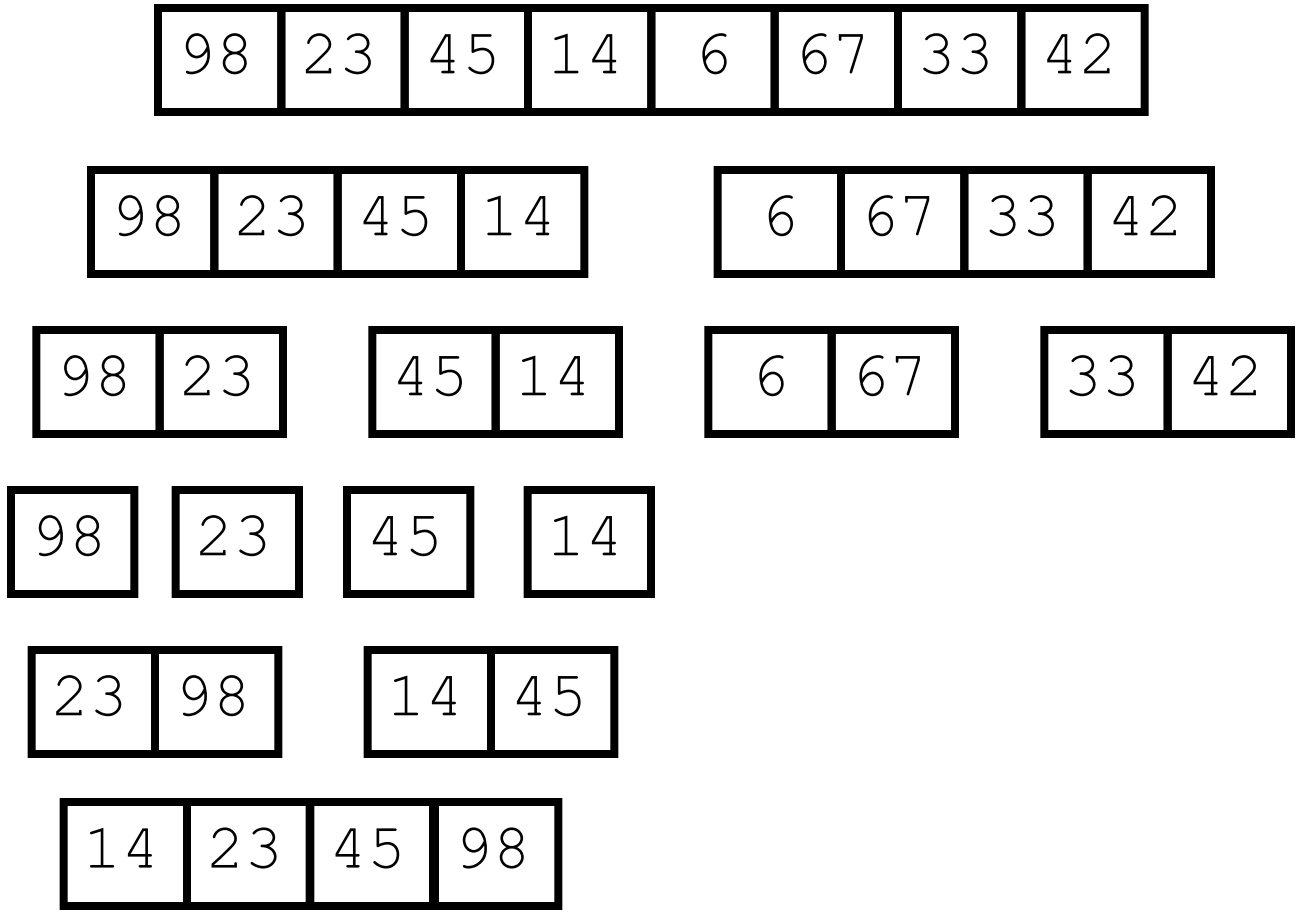
MERGE SORT EXAMPLE



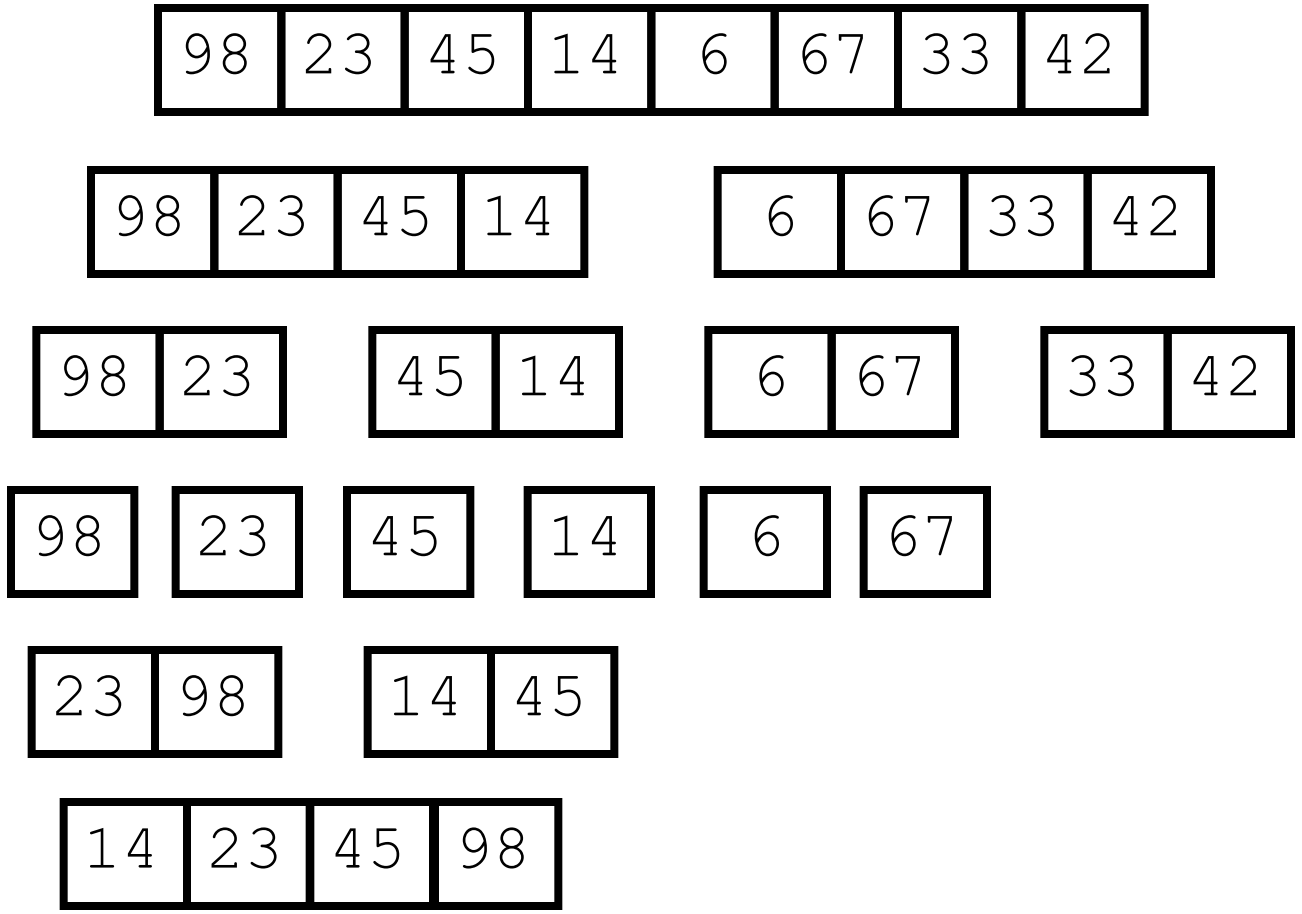
MERGE SORT EXAMPLE



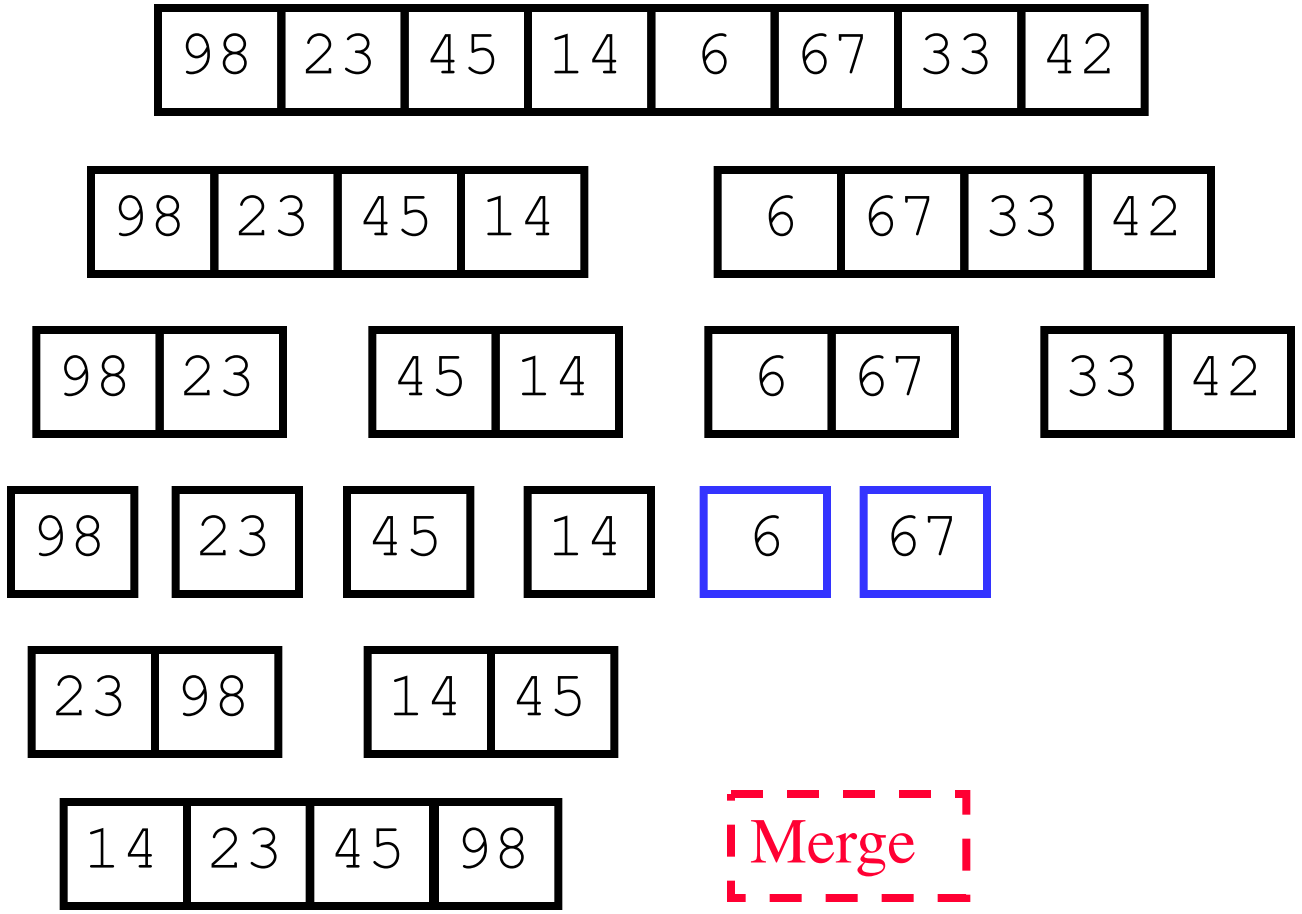
MERGE SORT EXAMPLE



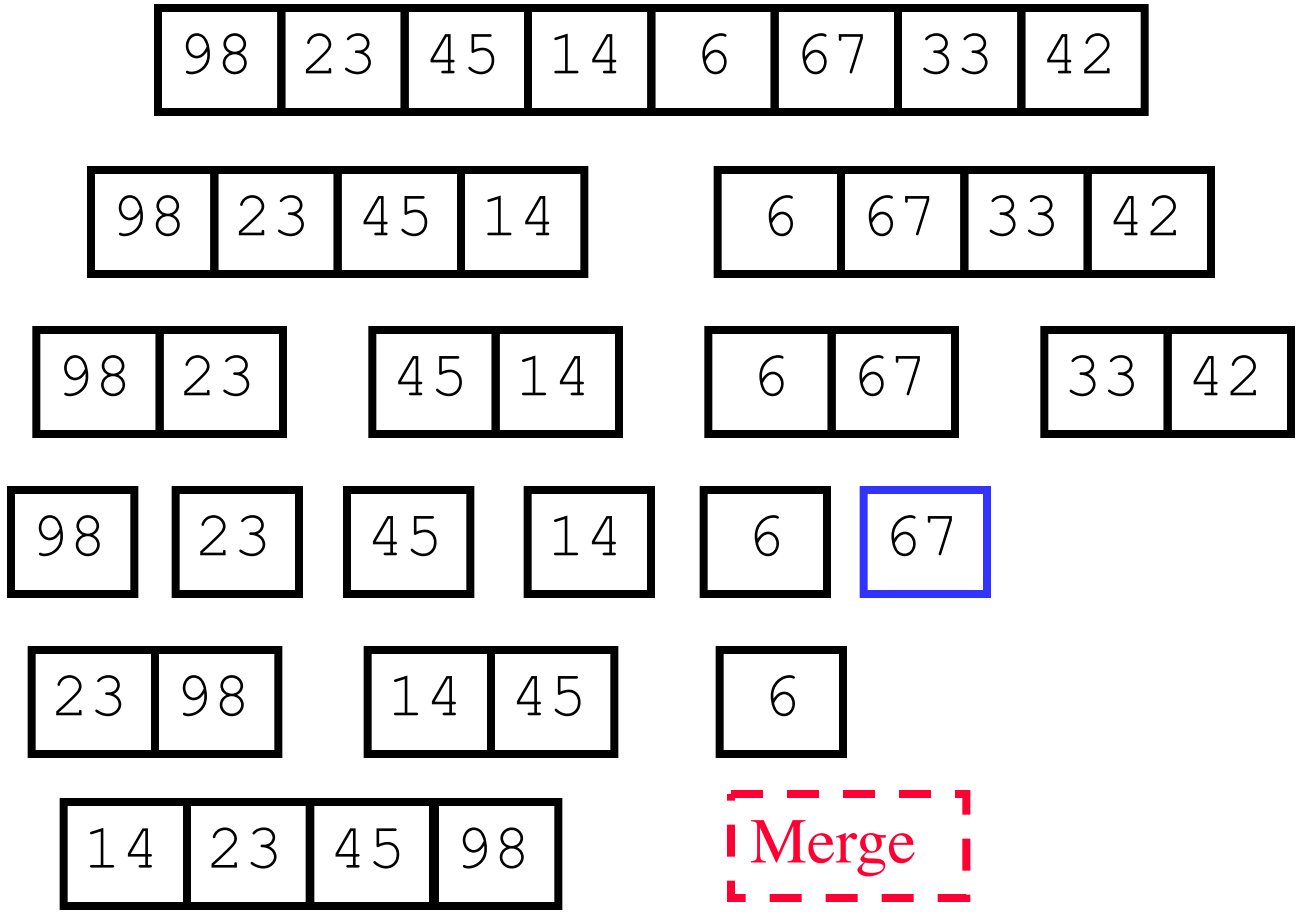
MERGE SORT EXAMPLE



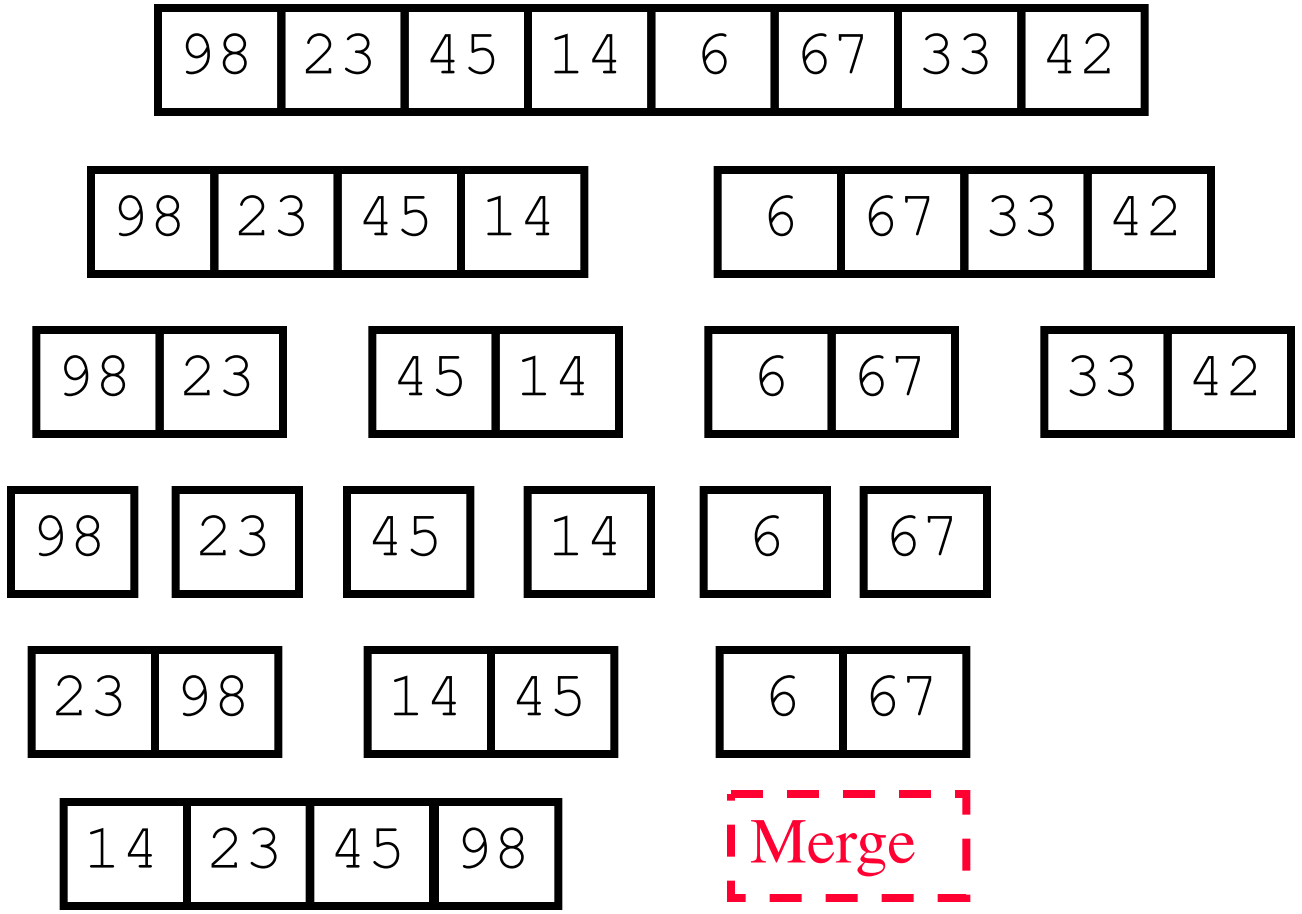
MERGE SORT EXAMPLE



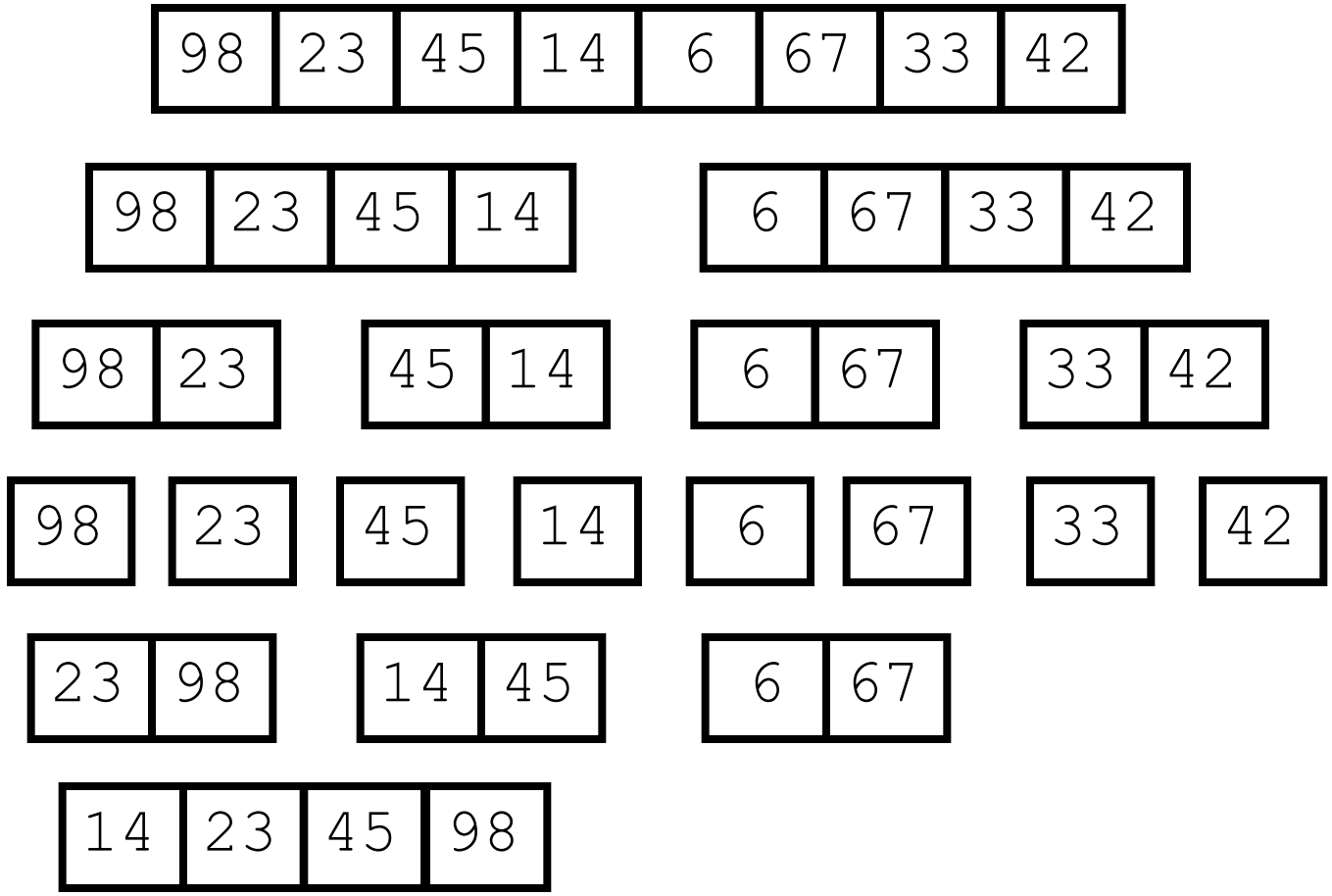
MERGE SORT EXAMPLE



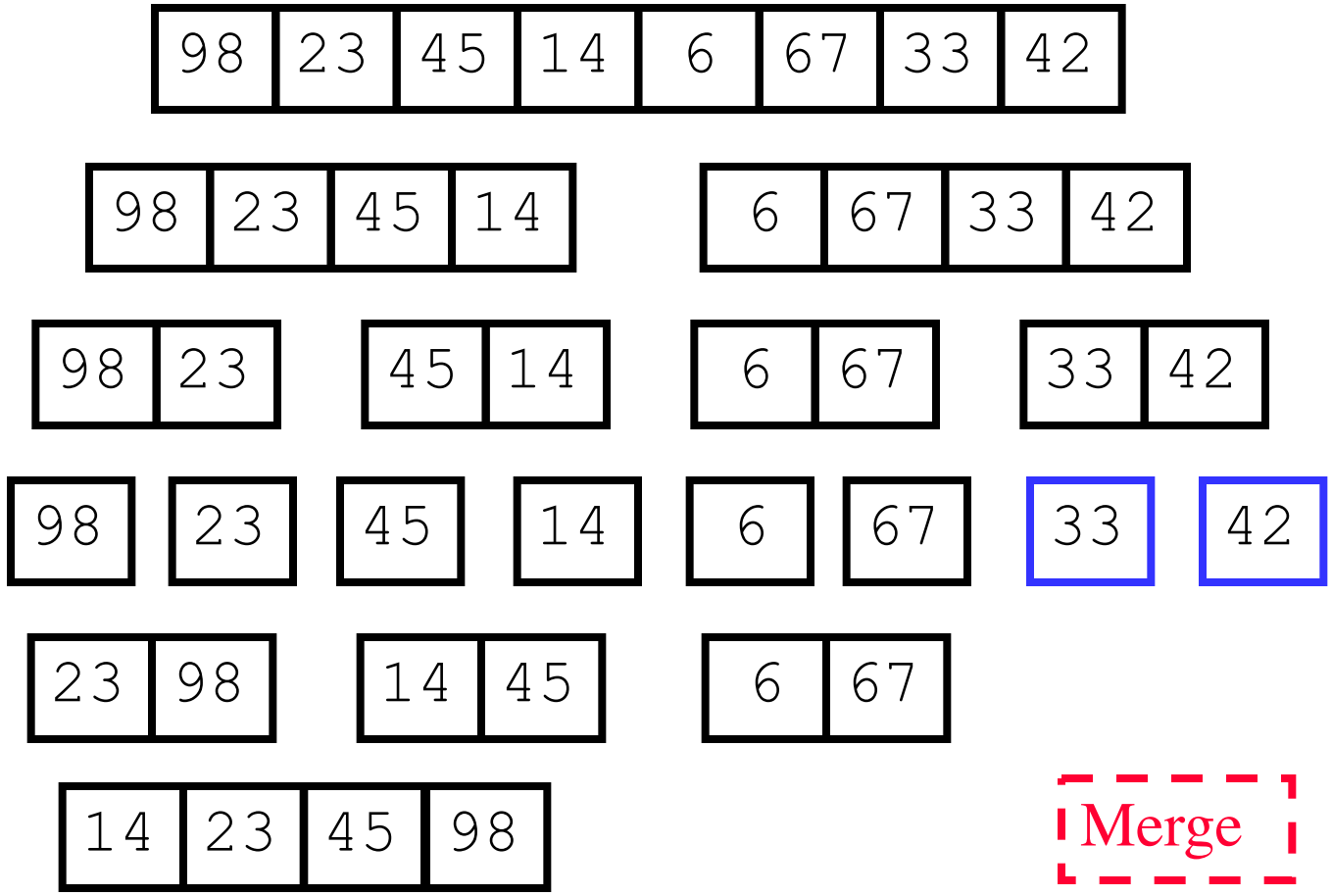
MERGE SORT EXAMPLE



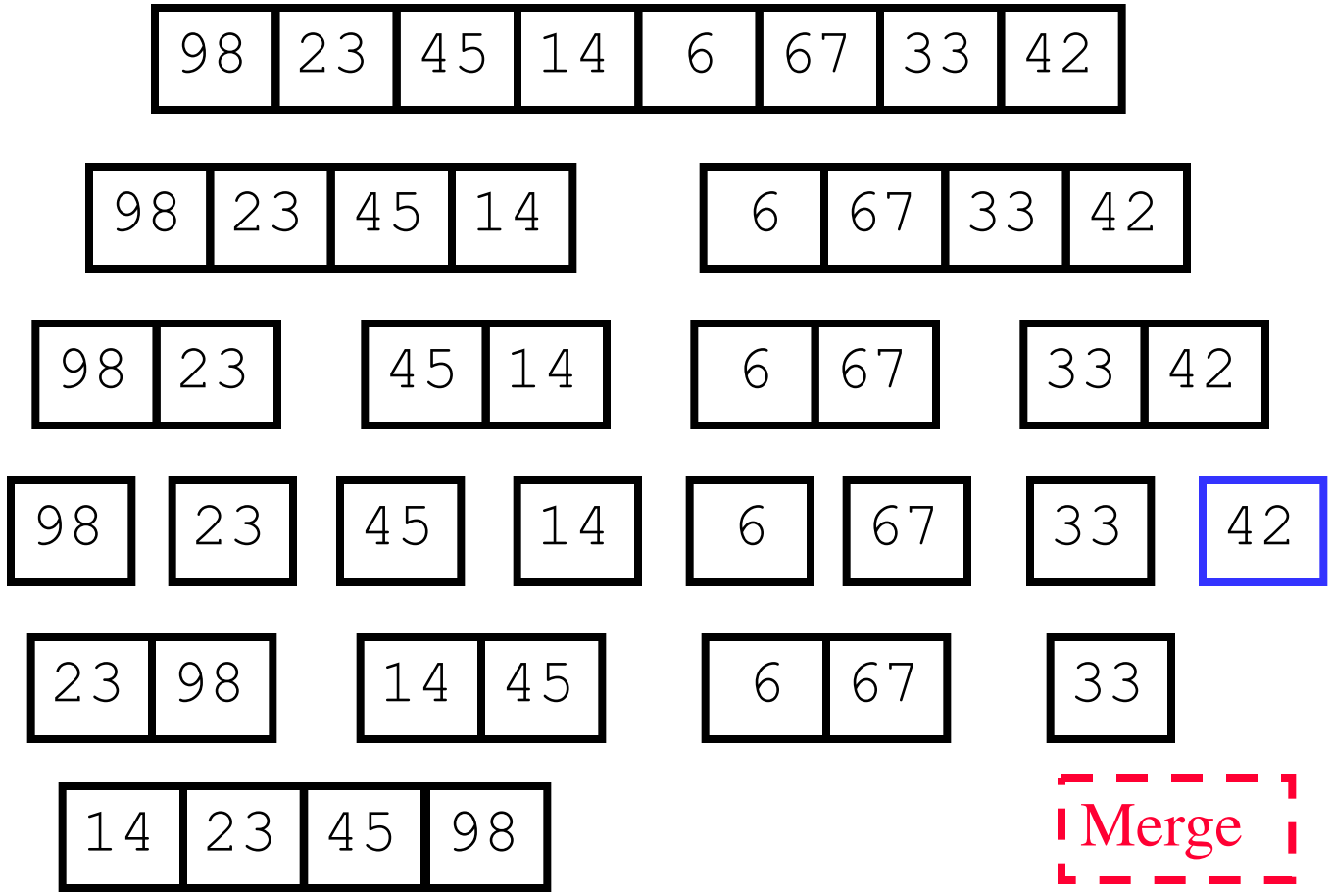
MERGE SORT EXAMPLE



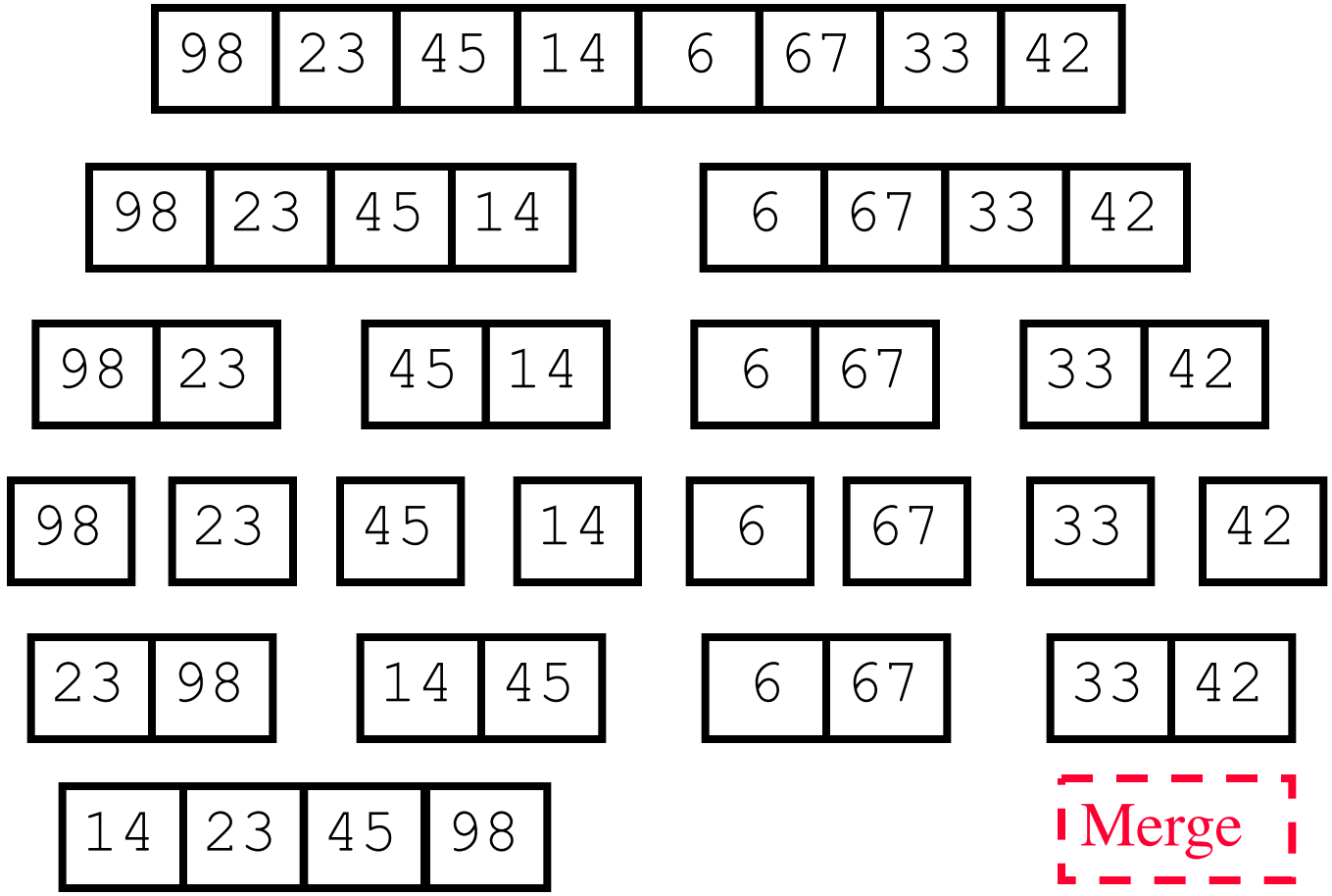
MERGE SORT EXAMPLE



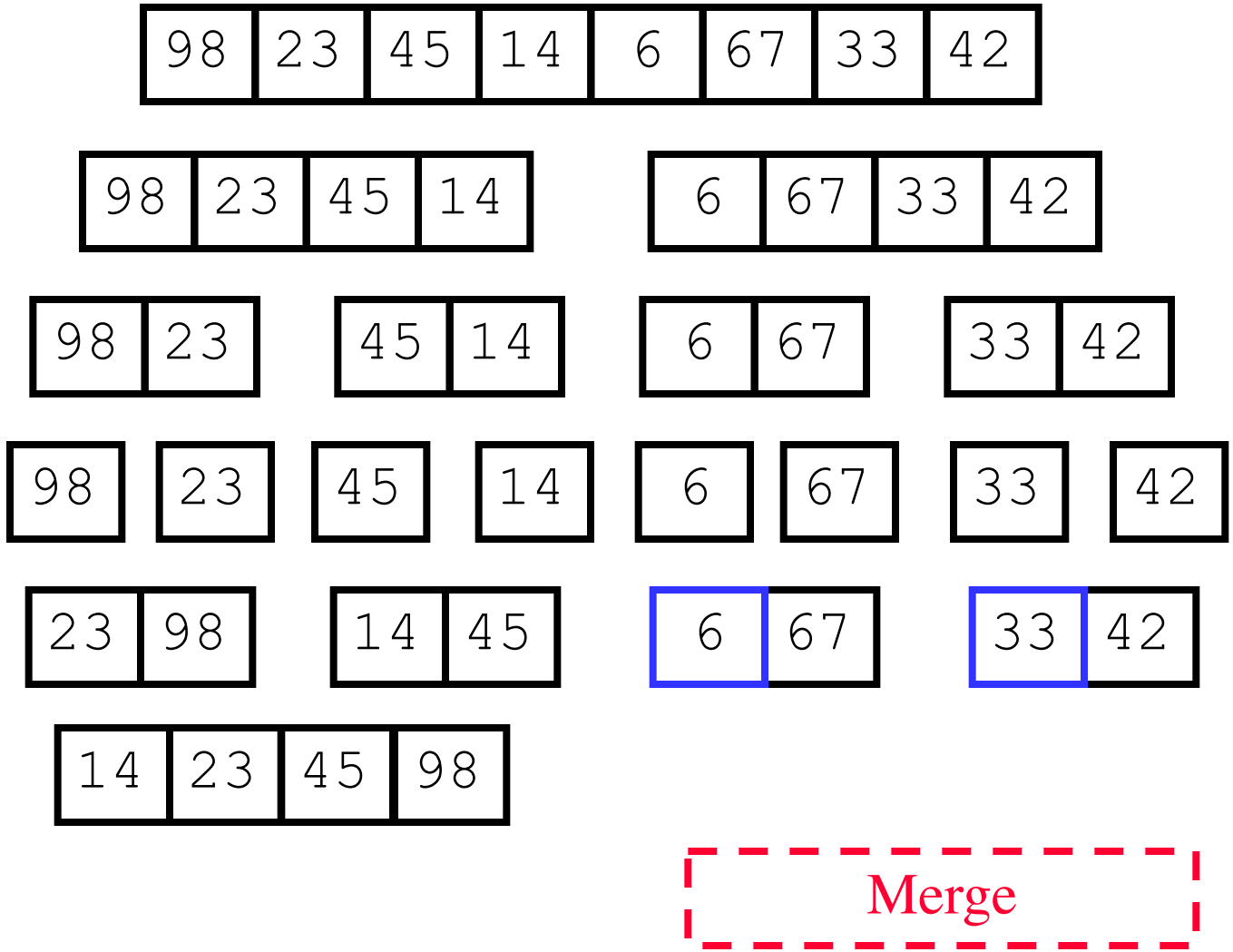
MERGE SORT EXAMPLE



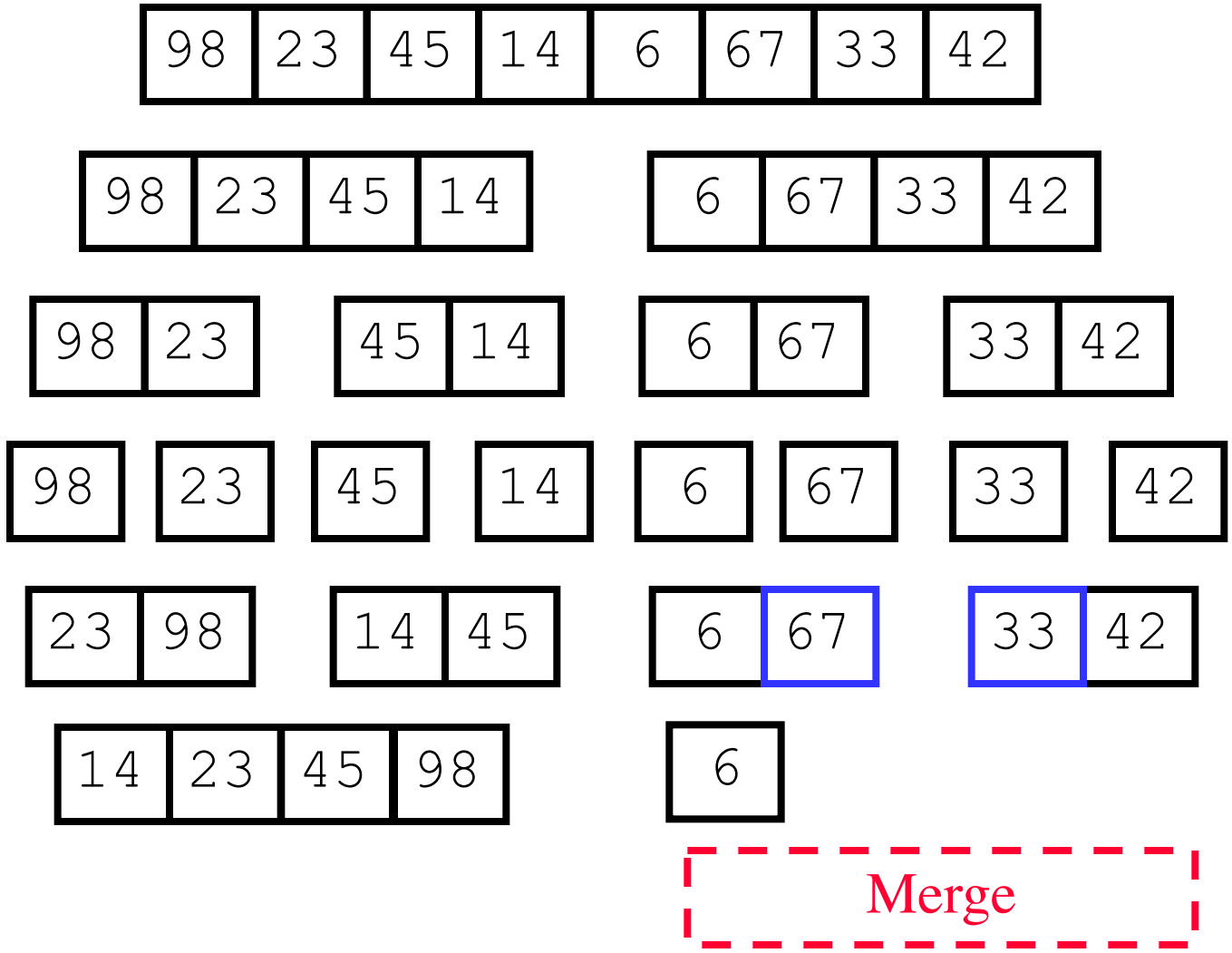
MERGE SORT EXAMPLE



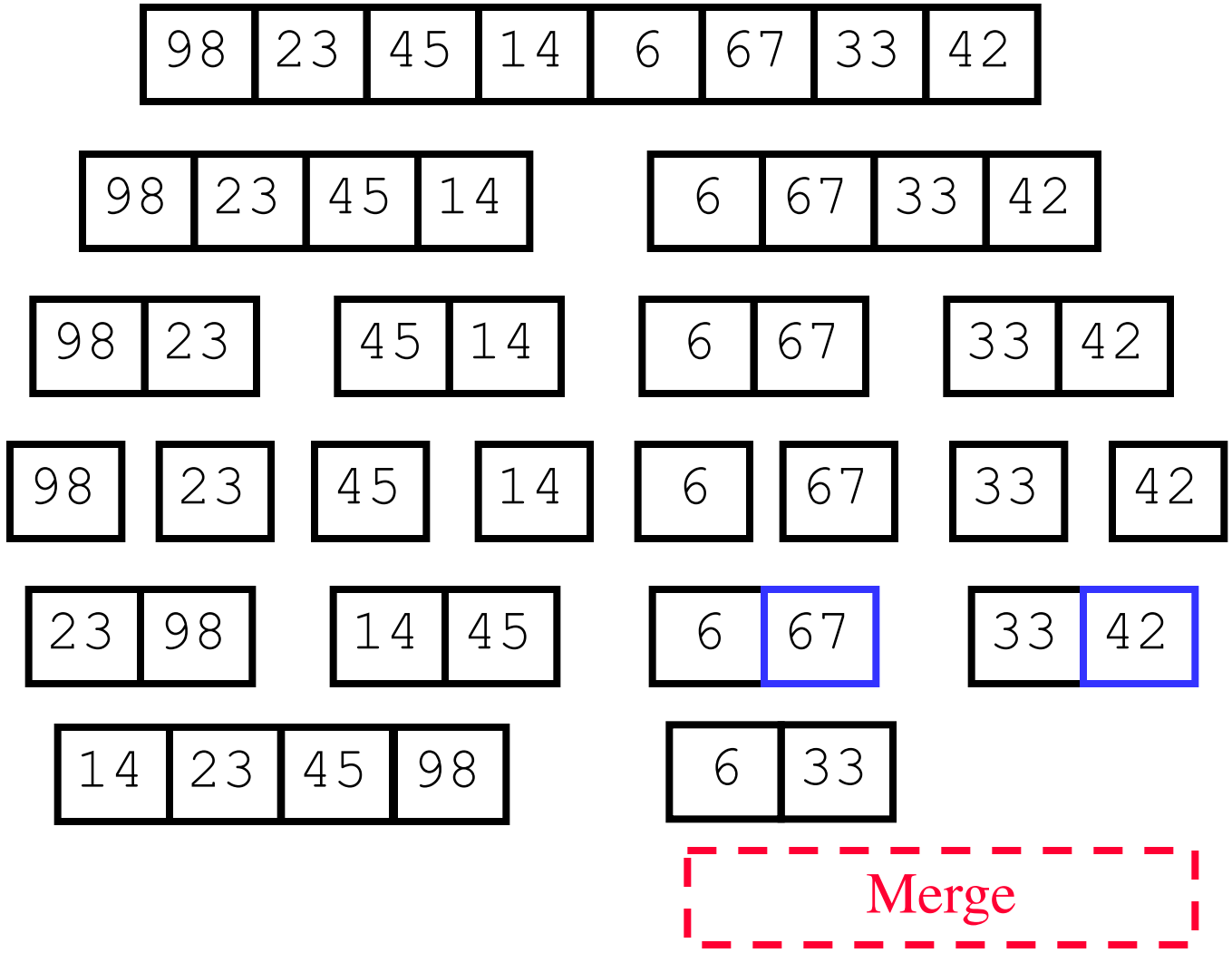
MERGE SORT EXAMPLE



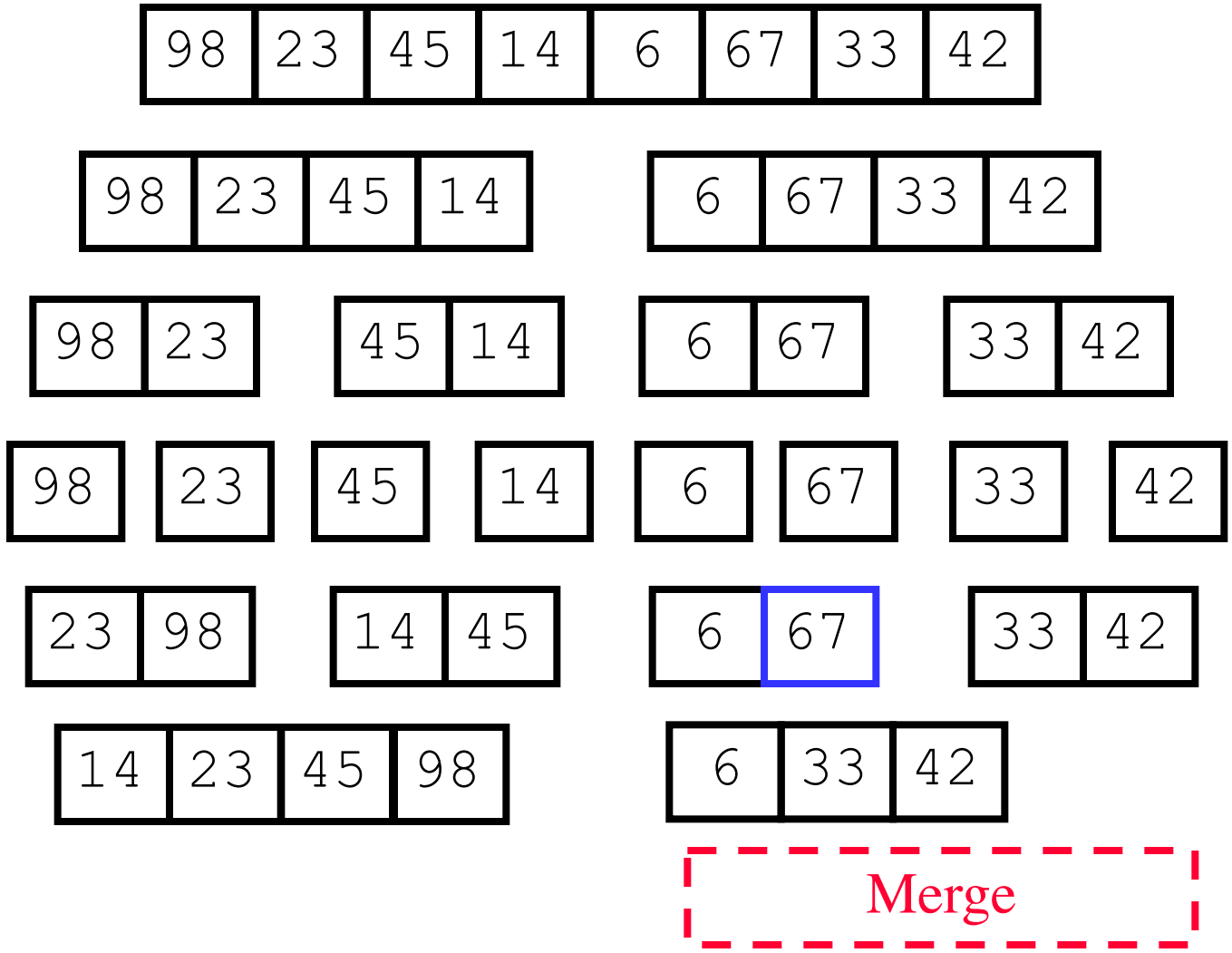
MERGE SORT EXAMPLE



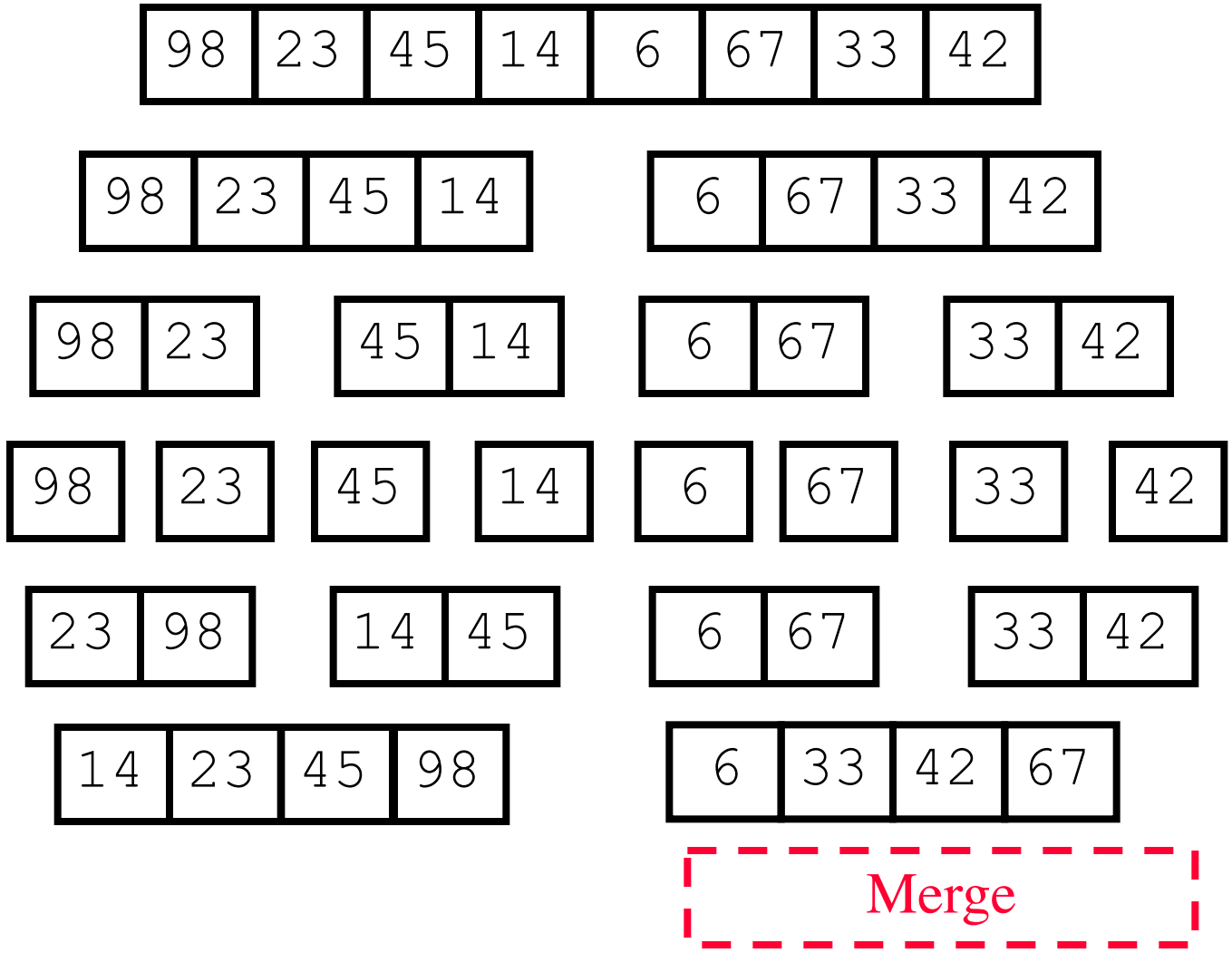
MERGE SORT EXAMPLE



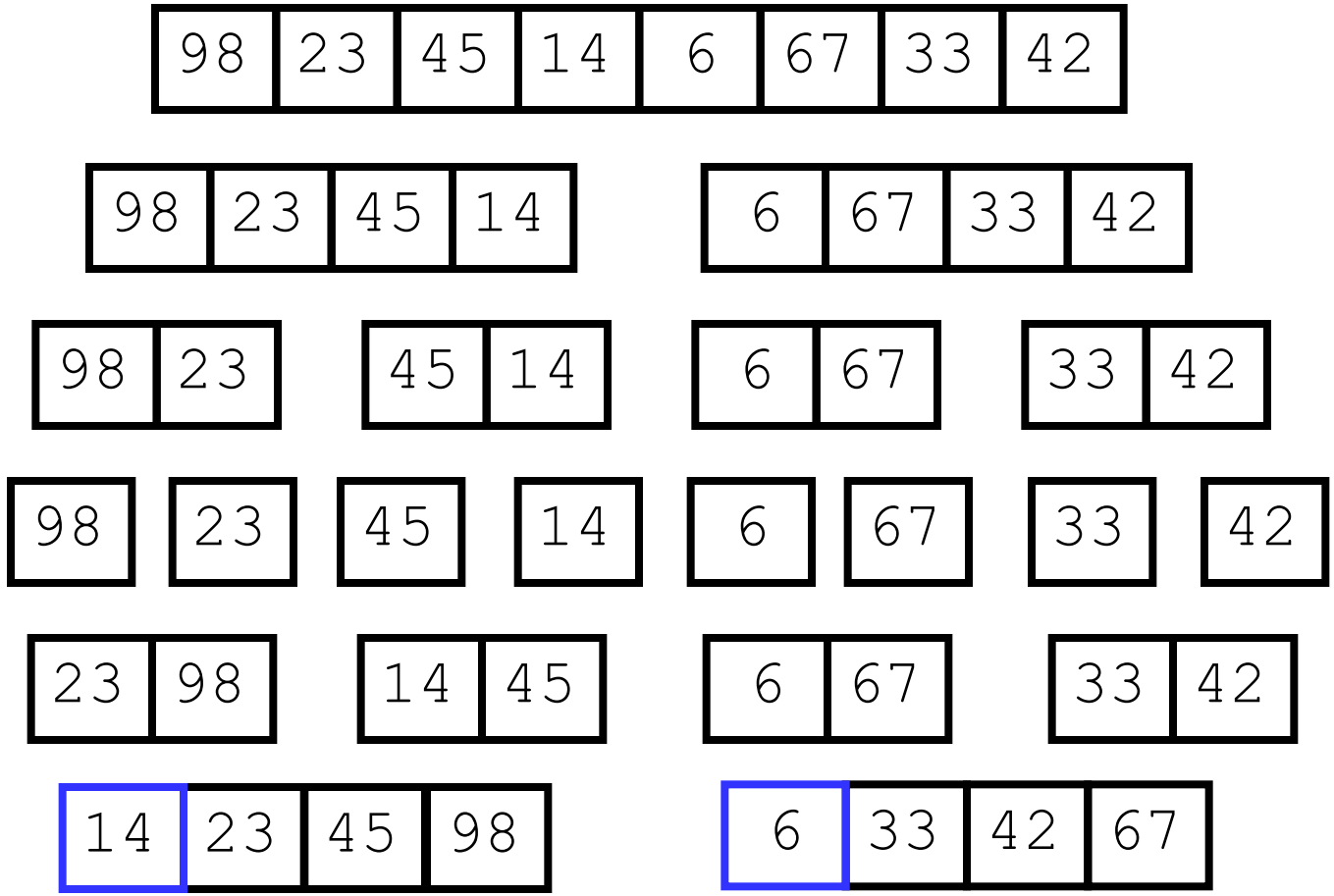
MERGE SORT EXAMPLE



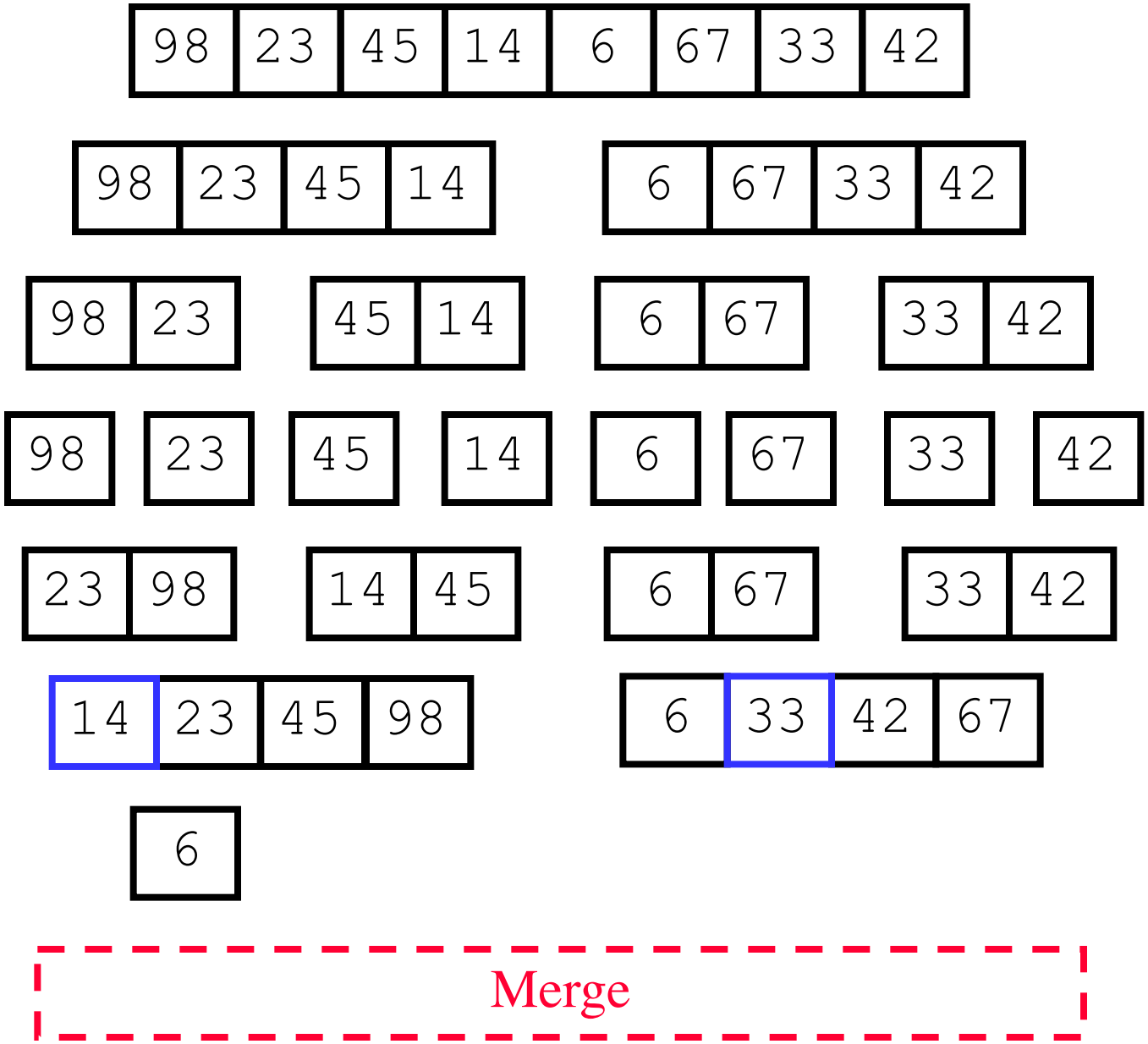
MERGE SORT EXAMPLE



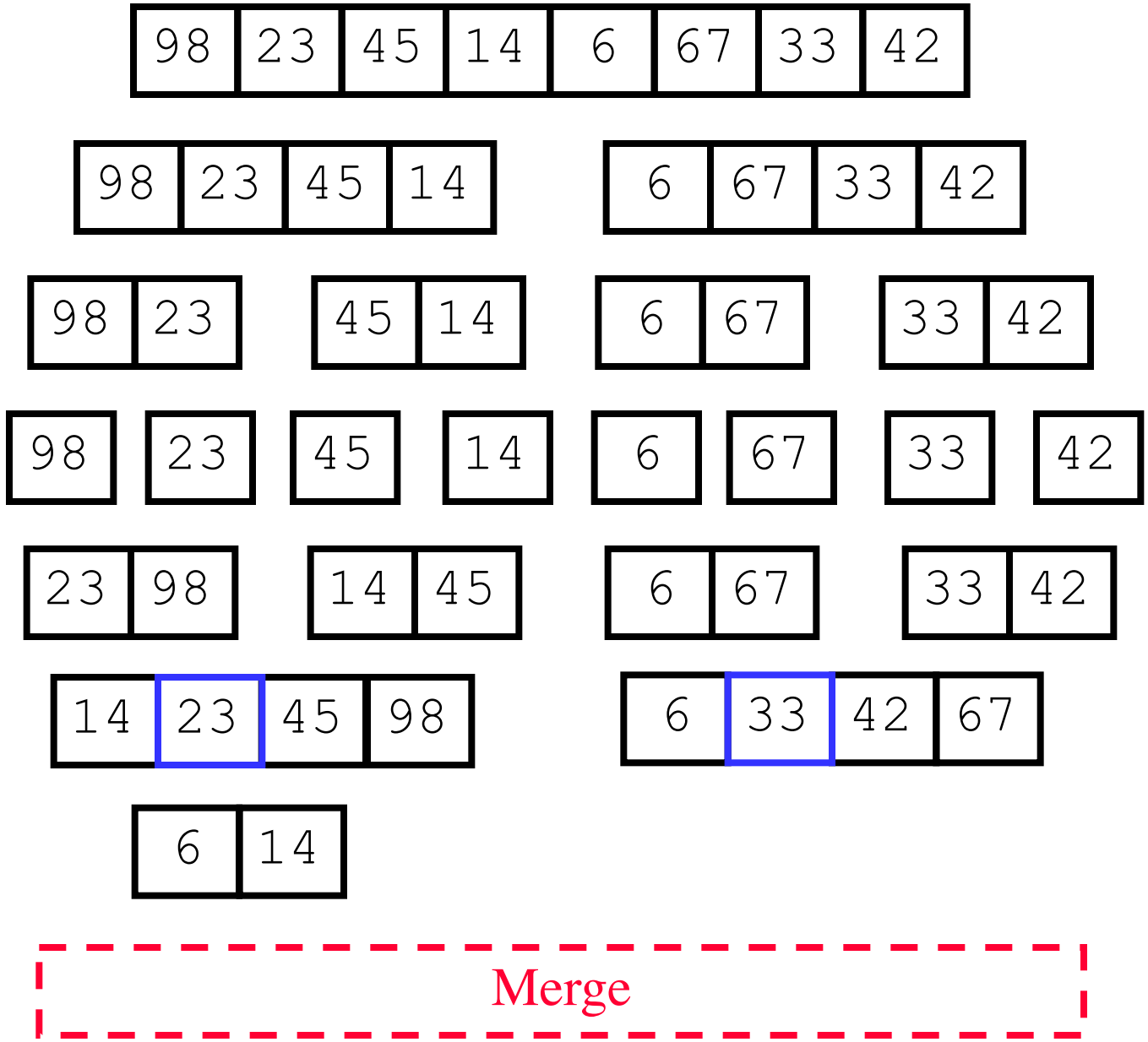
MERGE SORT EXAMPLE



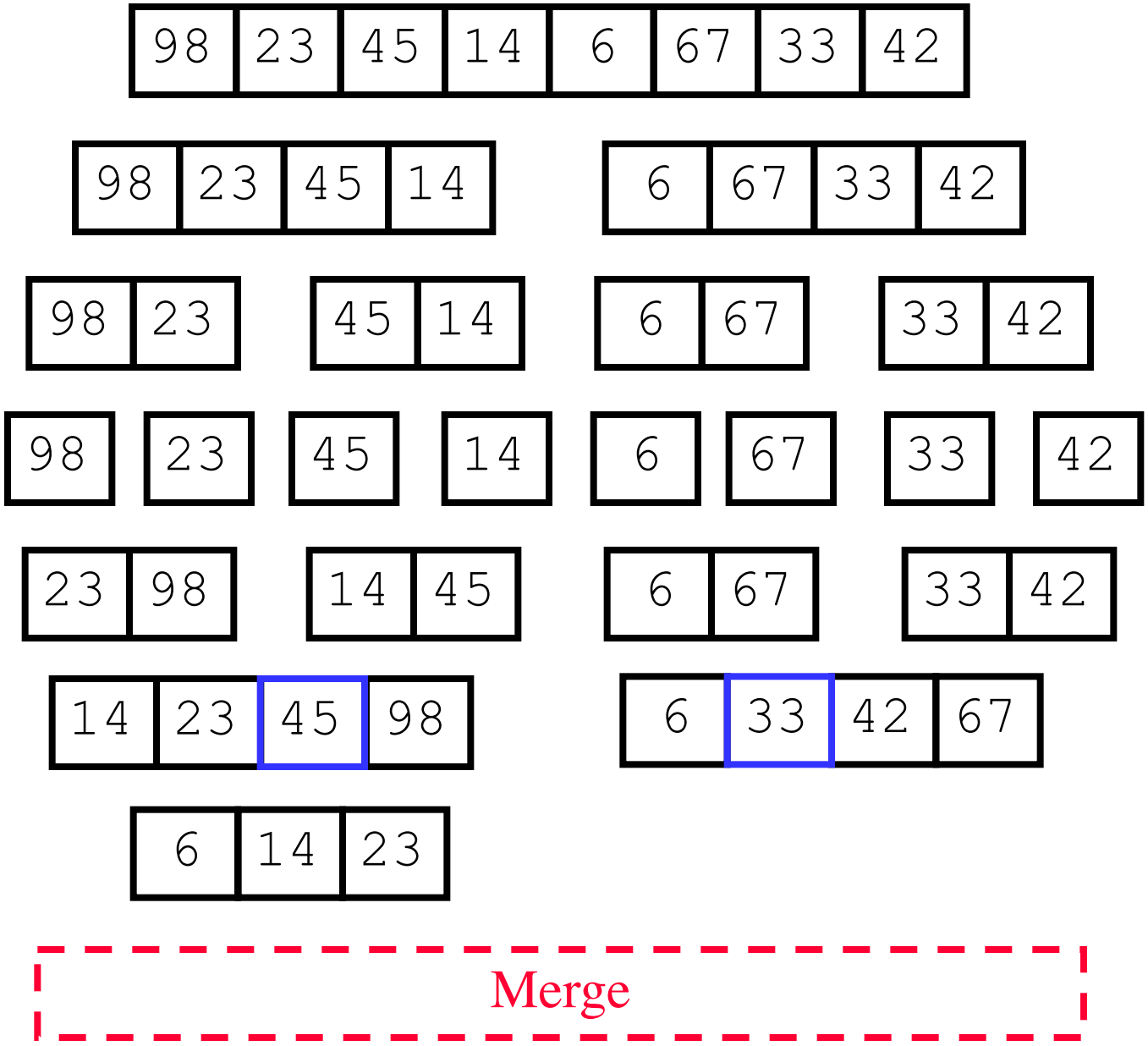
MERGE SORT EXAMPLE



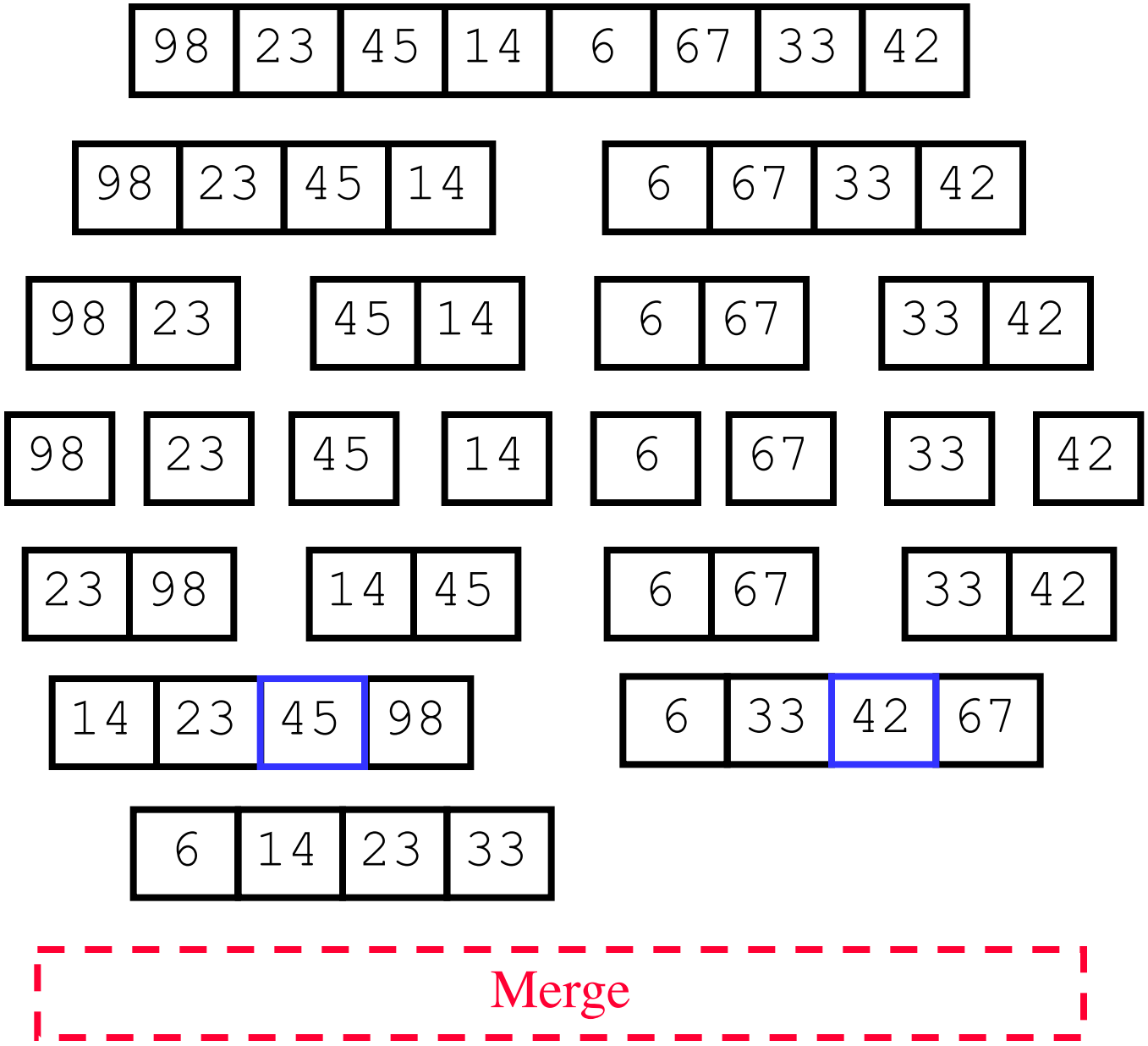
MERGE SORT EXAMPLE



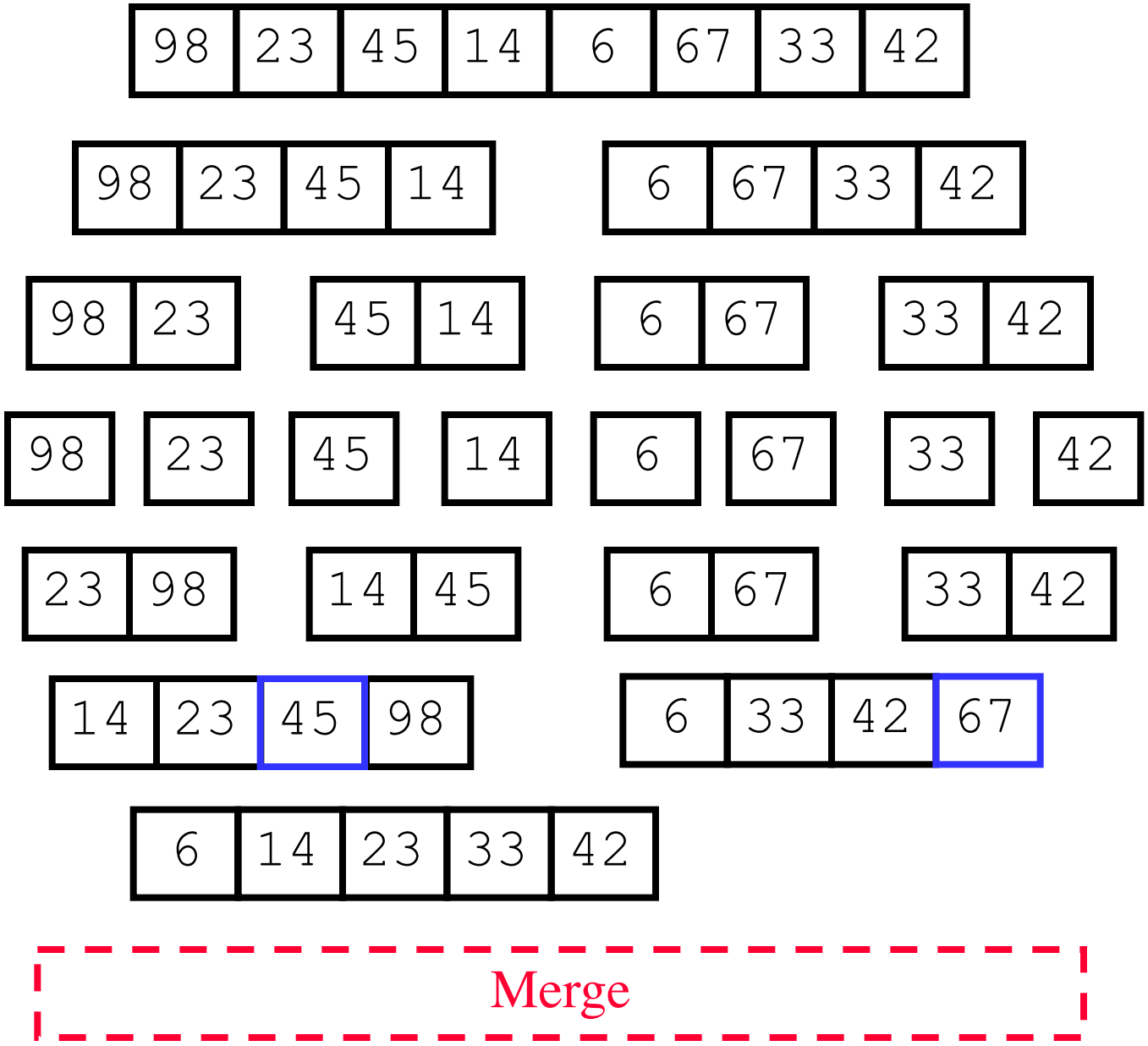
MERGE SORT EXAMPLE



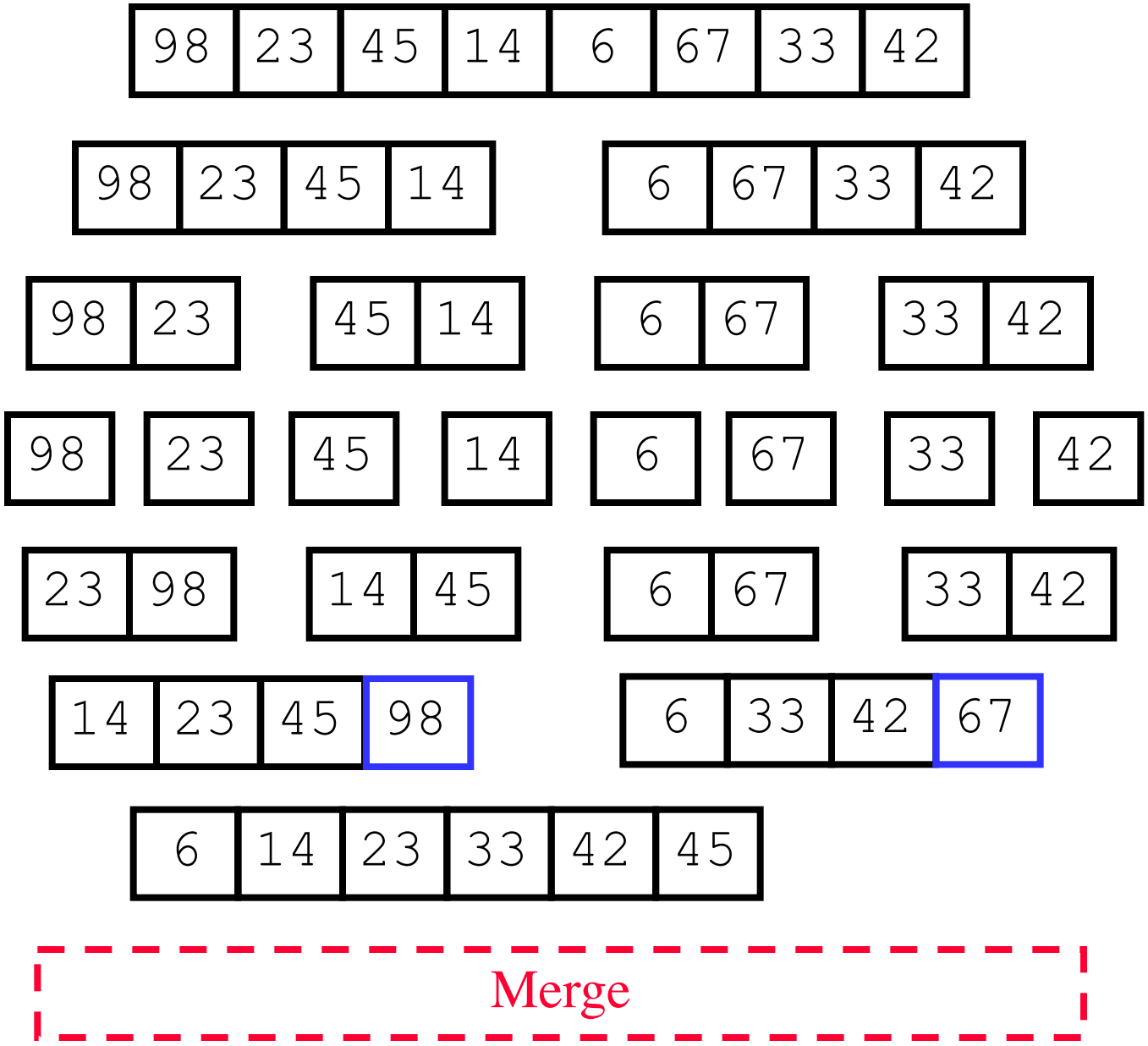
MERGE SORT EXAMPLE



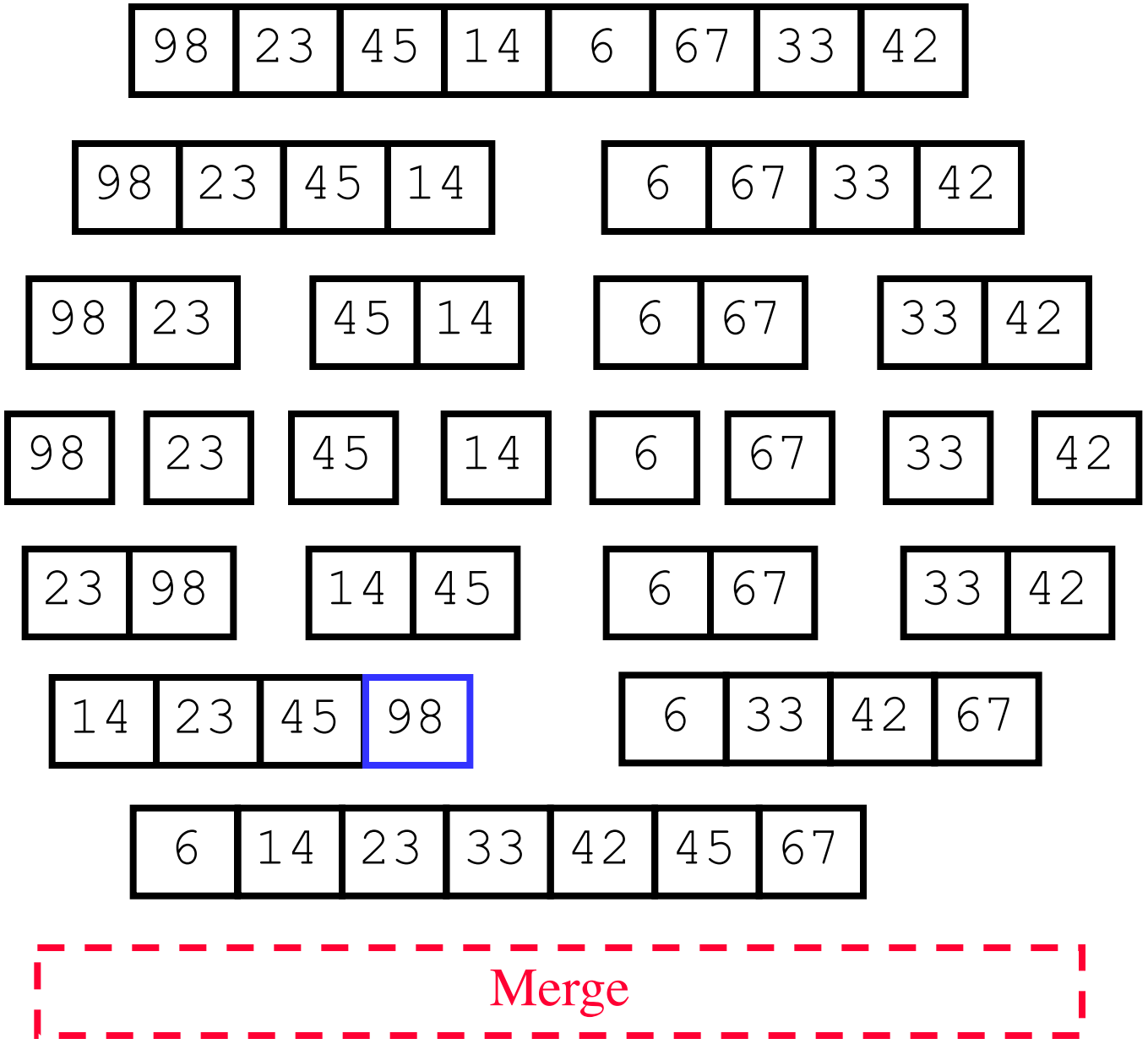
MERGE SORT EXAMPLE



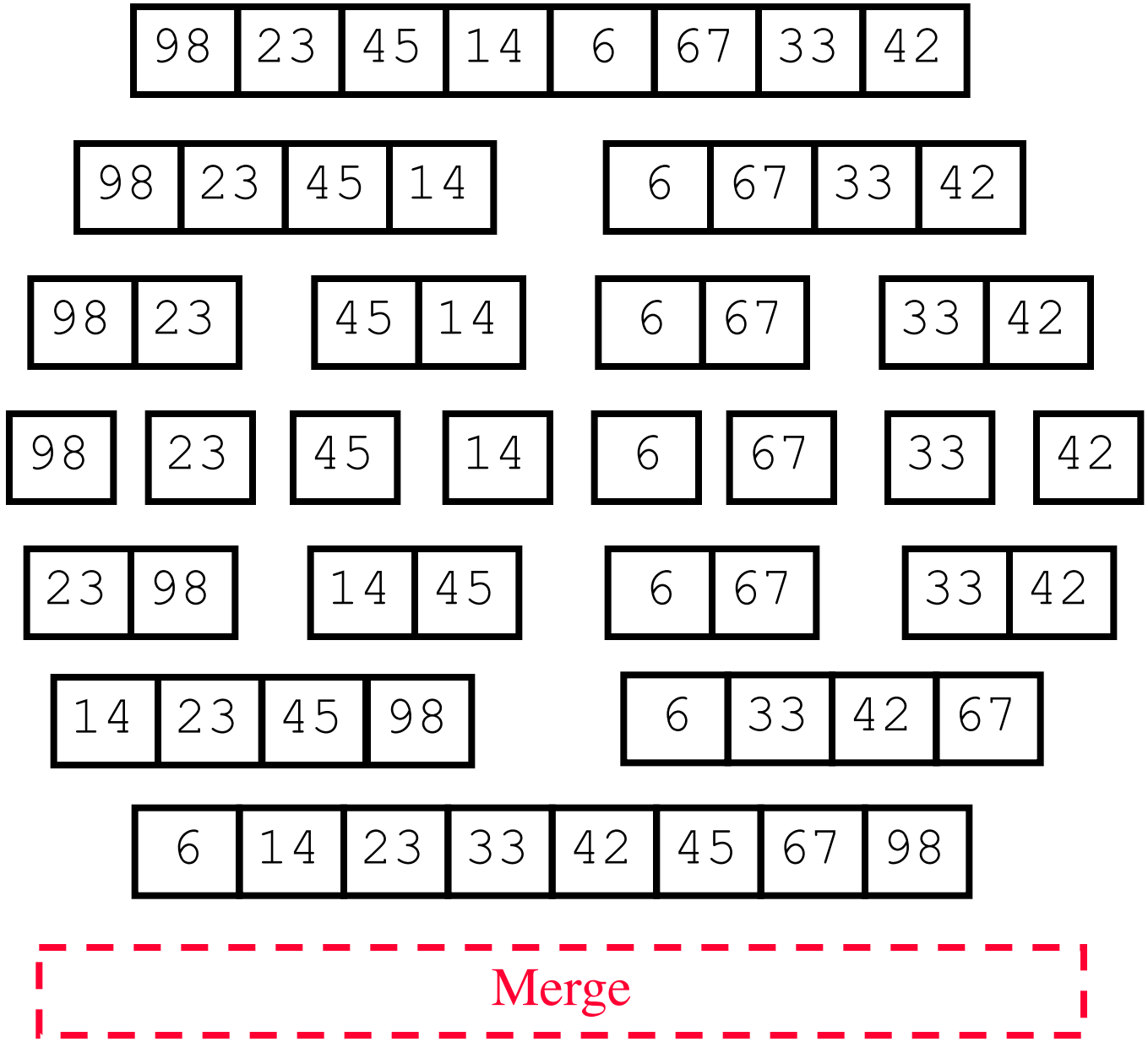
MERGE SORT EXAMPLE



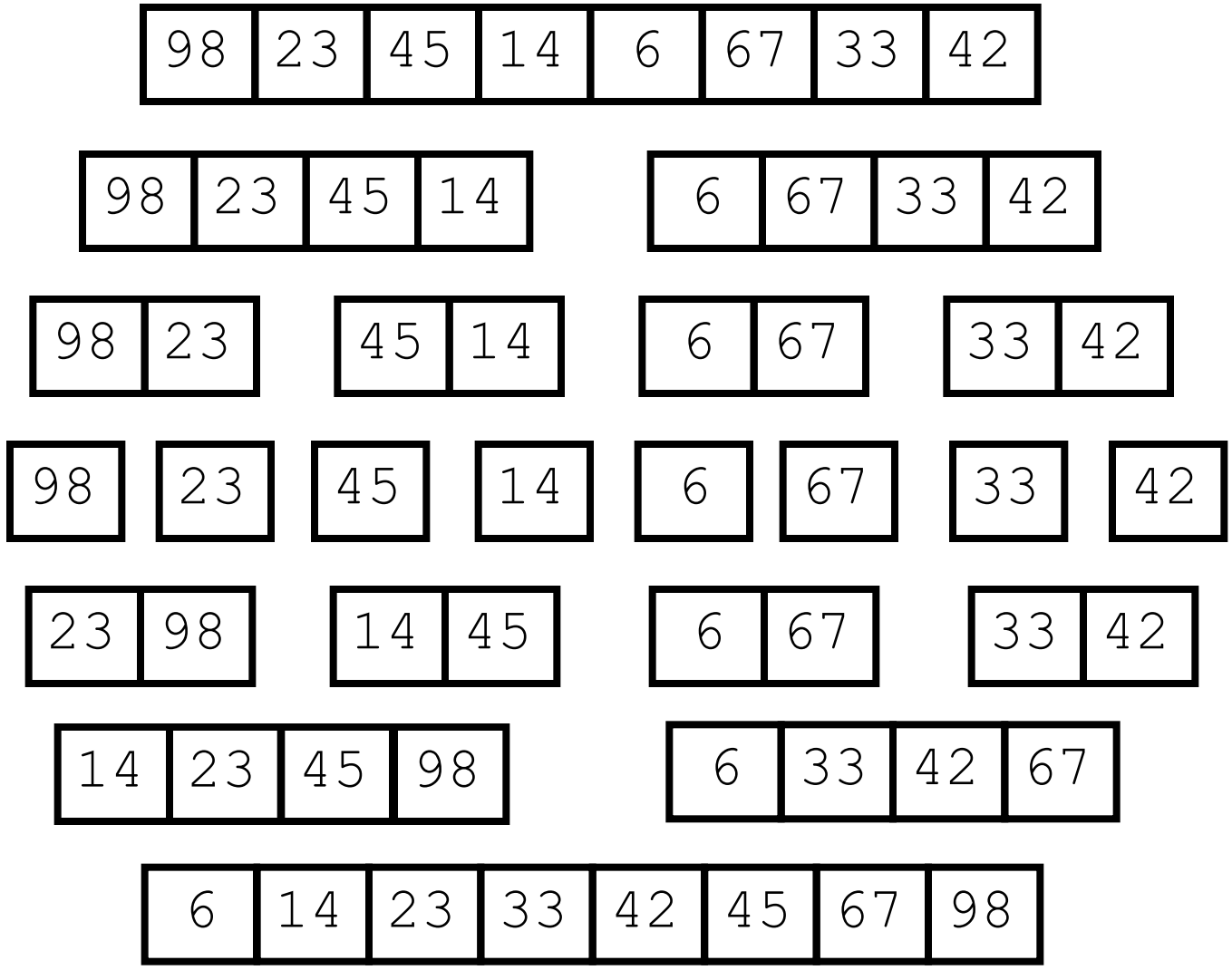
MERGE SORT EXAMPLE



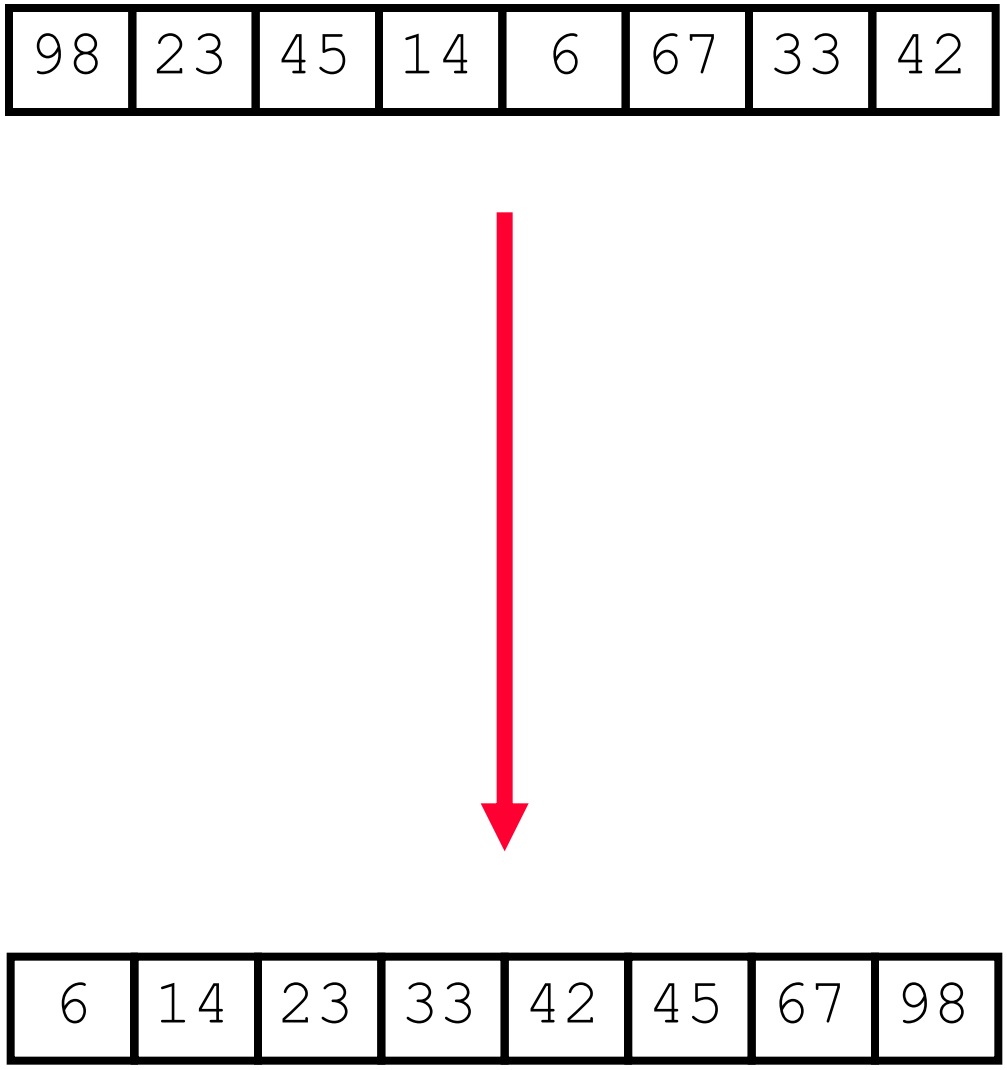
MERGE SORT EXAMPLE



MERGE SORT EXAMPLE

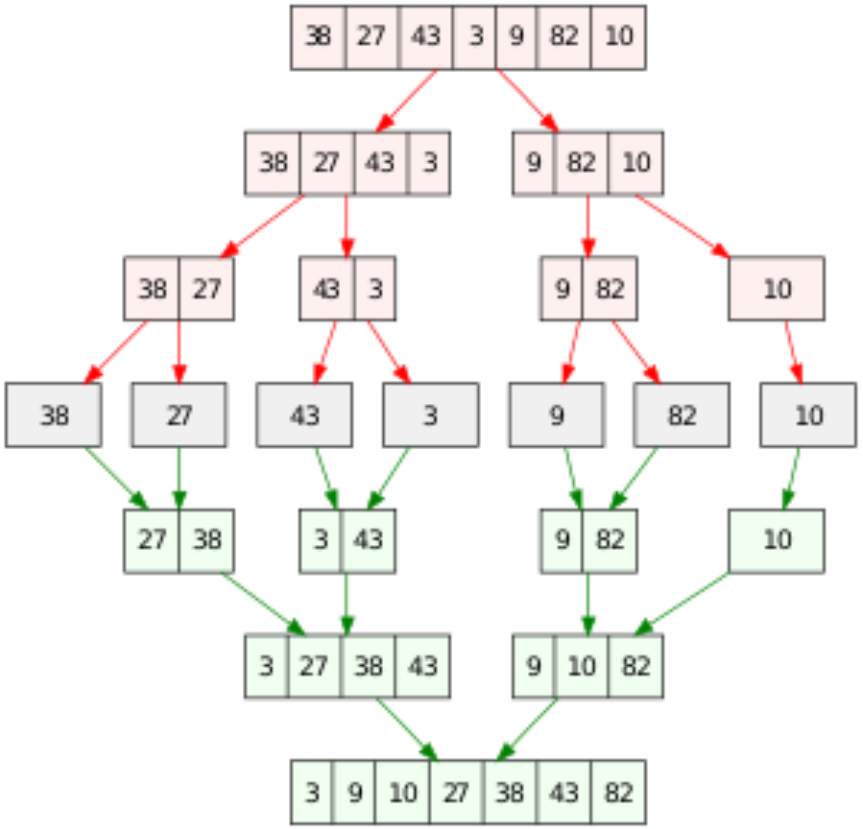


MERGE SORT EXAMPLE



Merge-Sort Memory Requirements

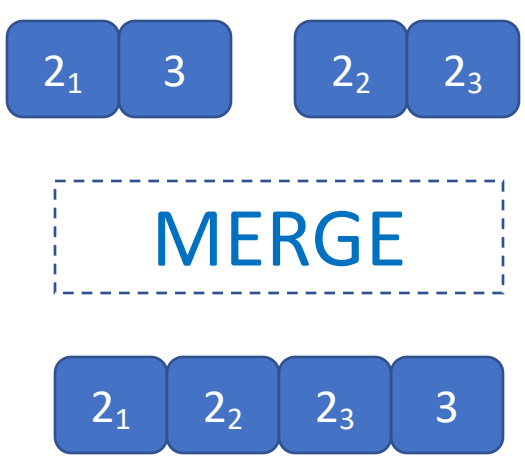
- The memory size of the input must be allocated for the sorted output to be stored.
  - **O(n)** space complexity
- In-place merge sort algorithm exist (quite cumbersome)



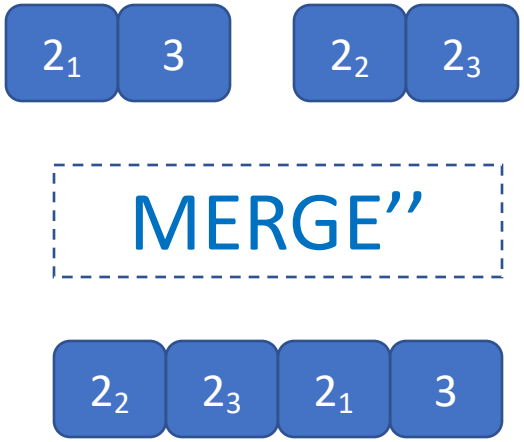
https://en.wikipedia.org/wiki/Merge_sort

Is Merge Sort Stable?

Example



STABILITY OF MERGE SORT?
It depends on the
Merge algorithm
used



Consider:
left array elements
before
right-array elements
if values equal
(the usual)

Consider:
right-array elements
before
left-array elements
if values equal
(alternate version)

Summary

- **Merge Sort** example of **Divide and Conquer Algorithms**
  - **Divide** the unsorted array **into two**
  - Until the sub-arrays only **contain one element**
  - Then **merge the sub-problem solutions** together
- It taken **O(n)** extra memory to sort
- Stability **depends** on the **merge** algorithm


That's all Folks!
Any Question ?