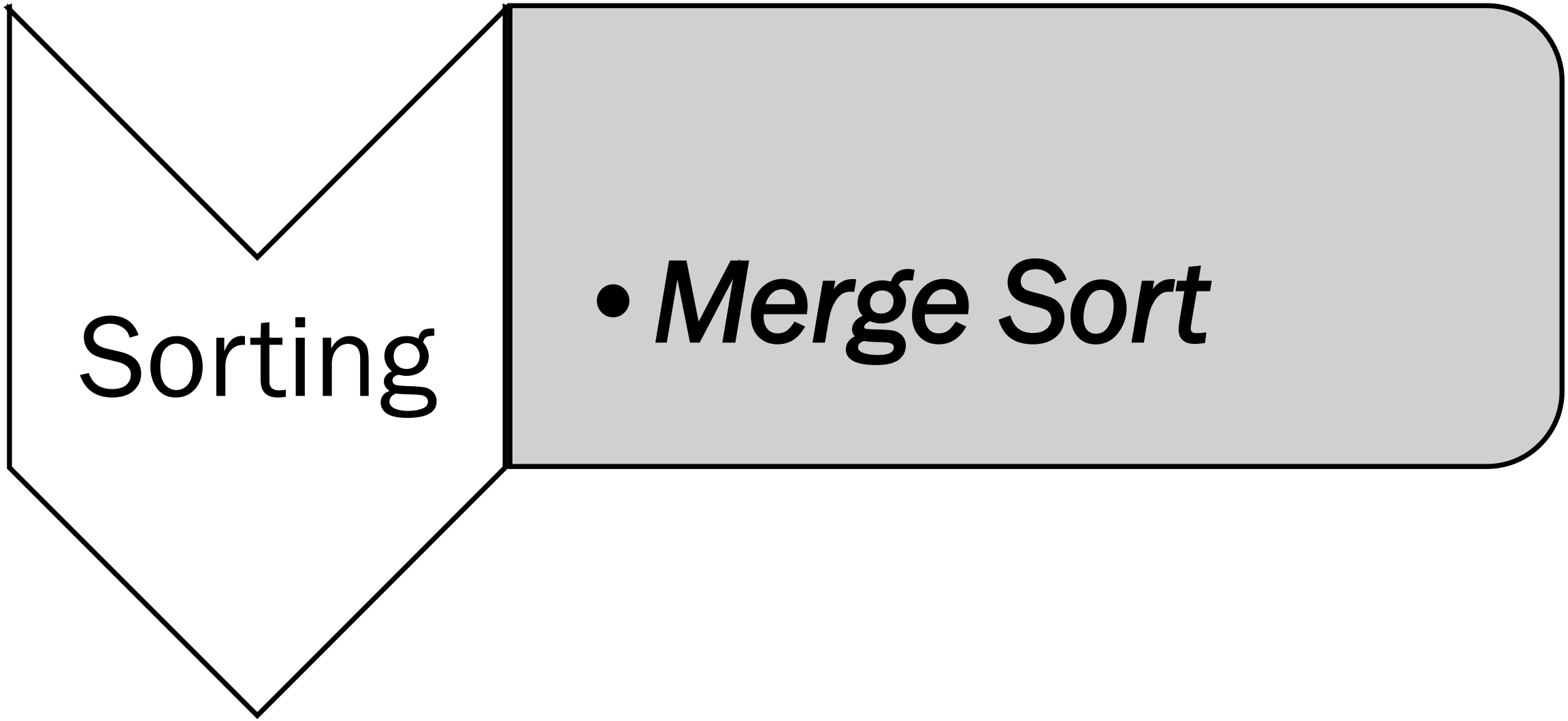
Todays discussion



Merge Sort

- Merge sort is a sorting technique based on divide and conquer technique. With worst-case time complexity being O(n log n), it is one of the most respected algorithms.

- Merge sort first divides the array into equal halves and then combines them in a sorted manner.

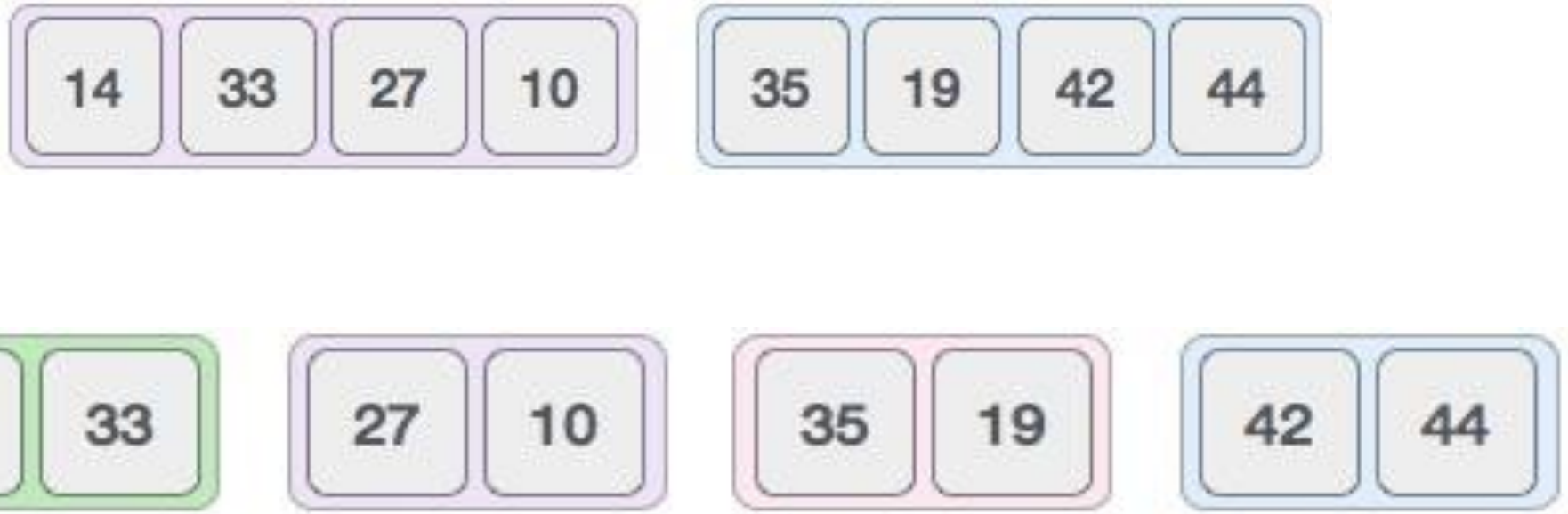
How Merge Sort Works?

- Take an unsorted array as the following

- Merge sort first divides the whole array iteratively into equal halves unless the atomic values are achieved.
  - *here that an array of 8 items is divided into two arrays of size 4*

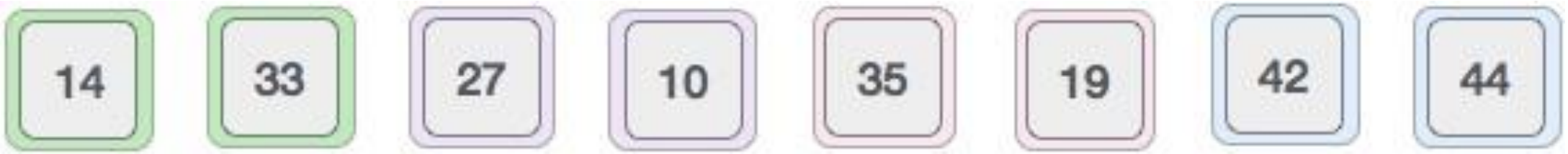
How Merge Sort Works?

- This does not change the sequence of appearance of items in the original. Now we divide these two arrays into halves.



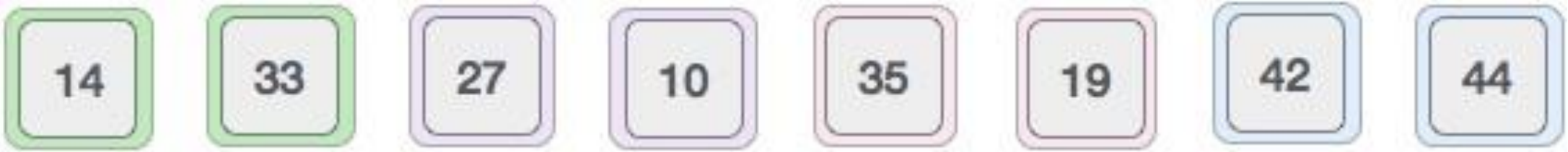
How Merge Sort Works?

- Further divide these arrays and we achieve atomic value which can no more be divided.



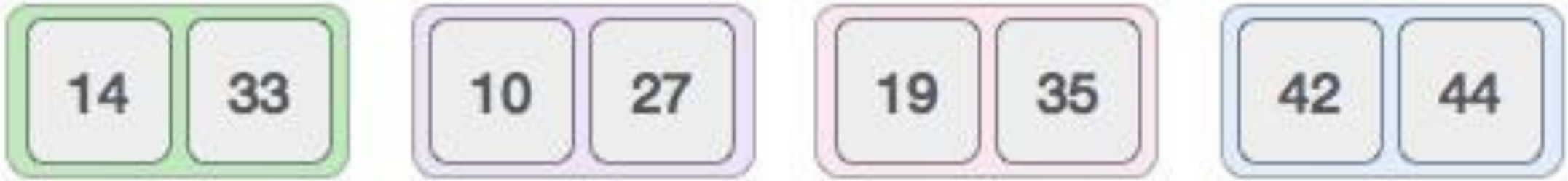
How Merge Sort Works?

- Now, we combine them in exactly the same manner as they were broken down. Please note the color codes given to these lists.



How Merge Sort Works?

- Compare the element for each list and then combine them into another list in a sorted manner.
- 14 and 33 are in sorted positions.
- 27 and 10 and in the target list of 2 values we put 10 first, followed by 27. We change the order of 19 and 35 whereas 42 and 44 are placed sequentially.

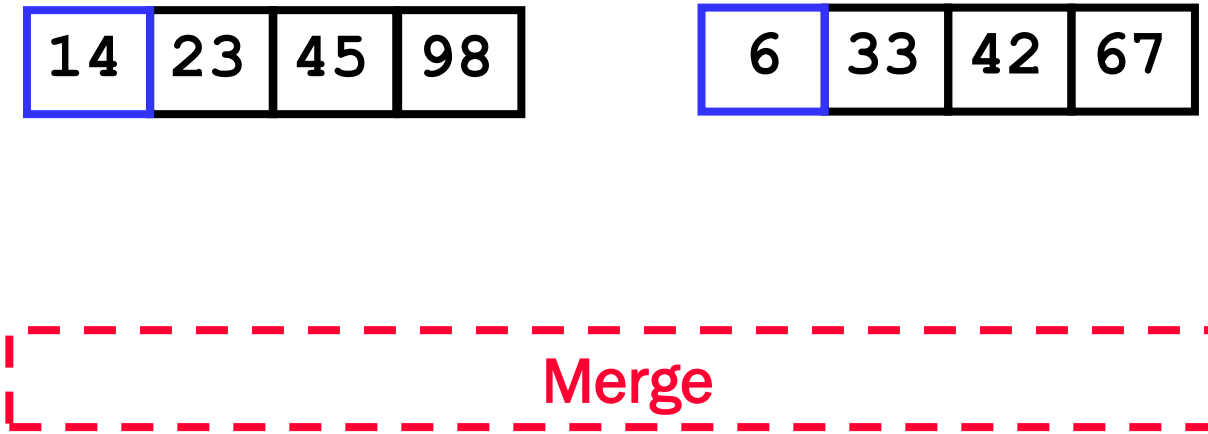


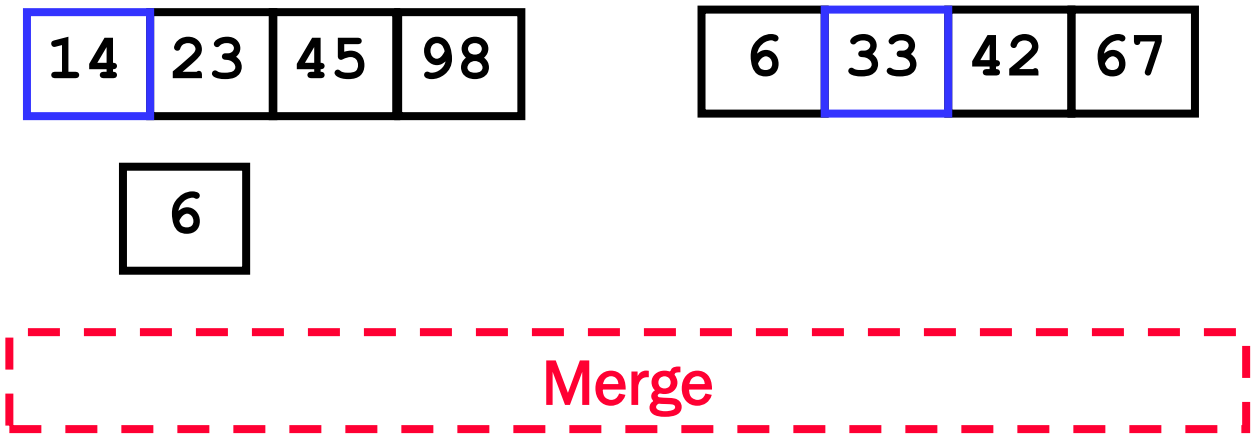
How Merge Sort Works?

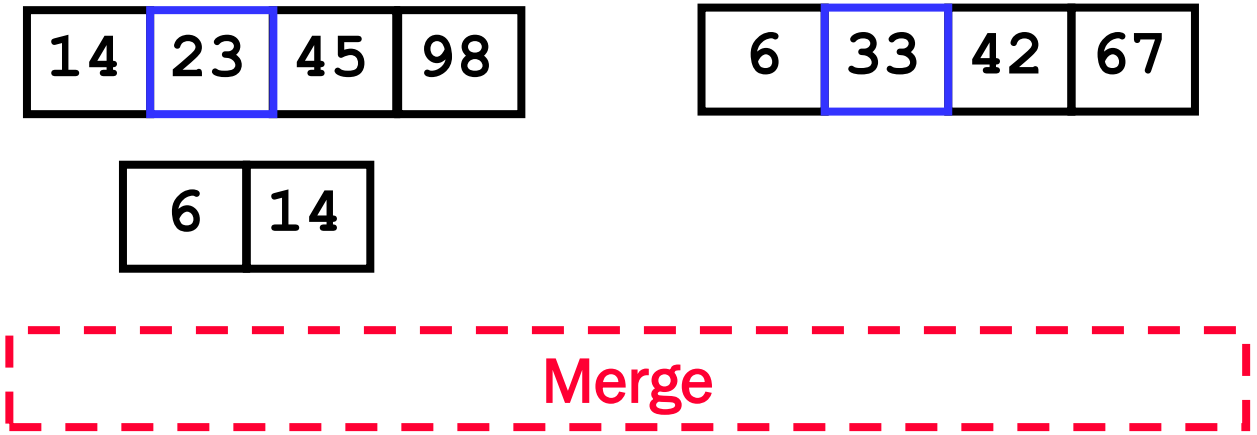
- In the next iteration of the combining phase, we compare lists of two data values, and merge them into a list of found data values placing all in a sorted order.

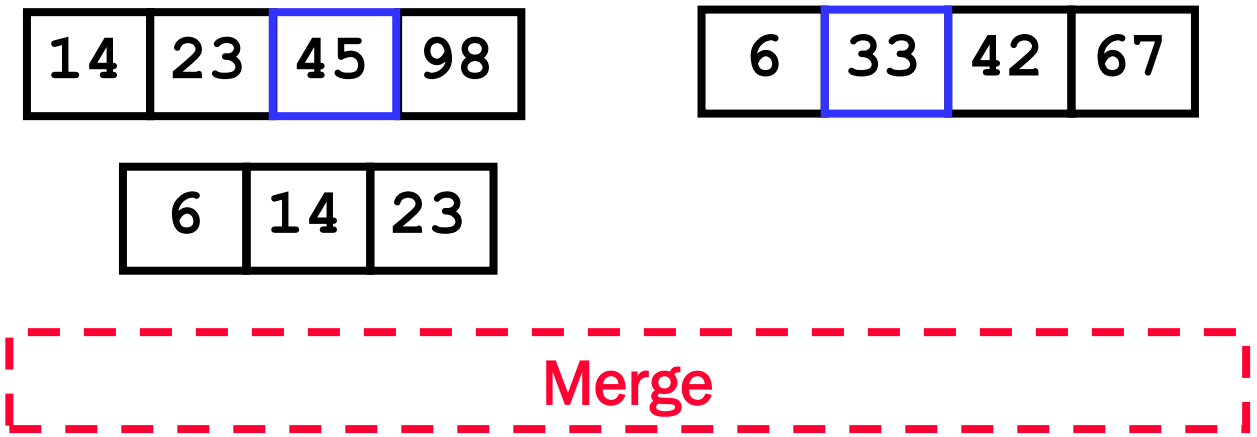
- After the final merging, the list should look like this

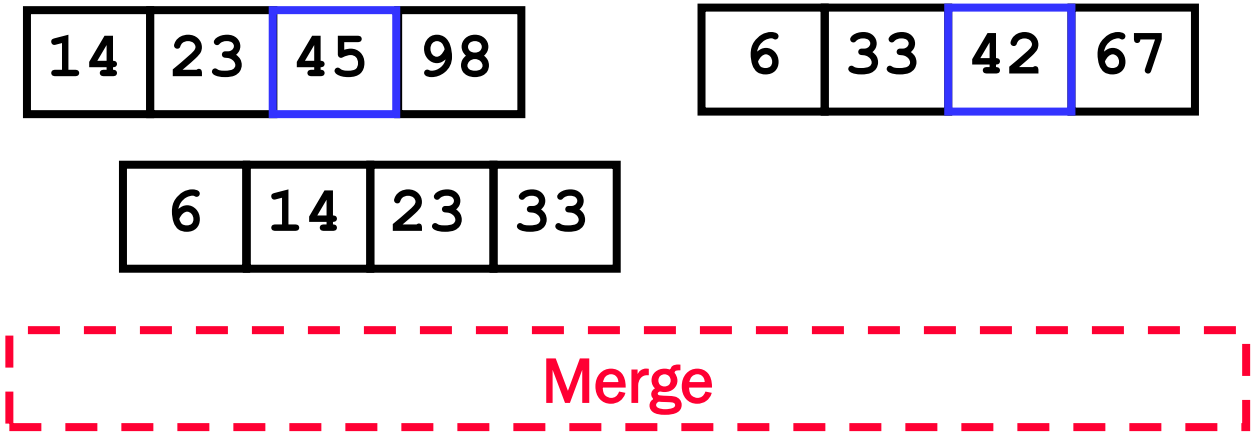

14
23
45
98
6
33
42
67

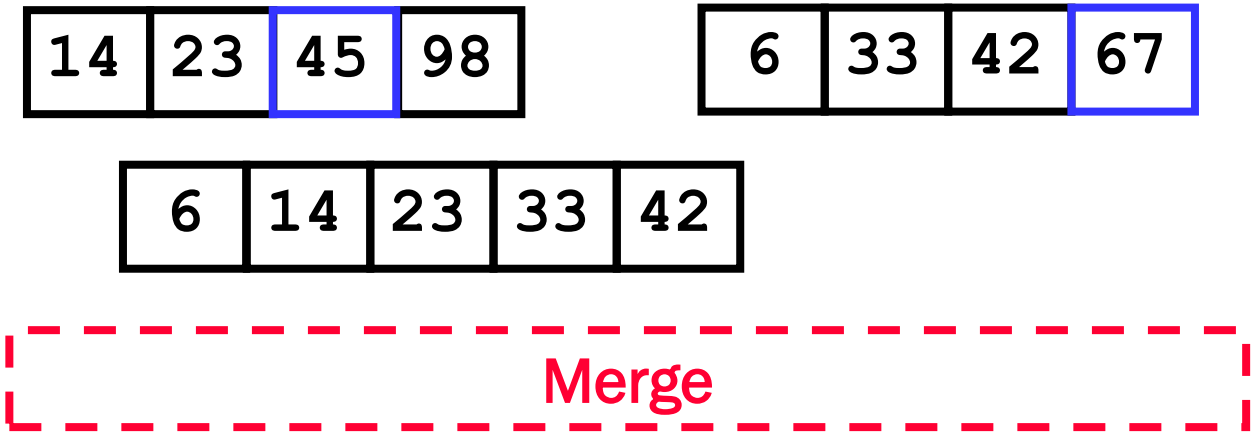

14
23
45
98
6
33
42
67
Merge

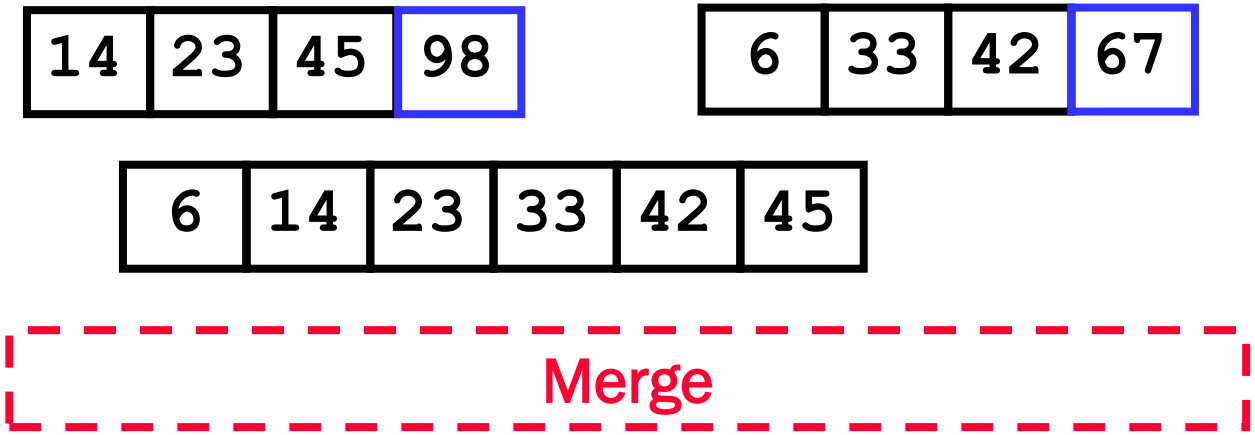

14
23
45
98
6
33
42
67
6
Merge

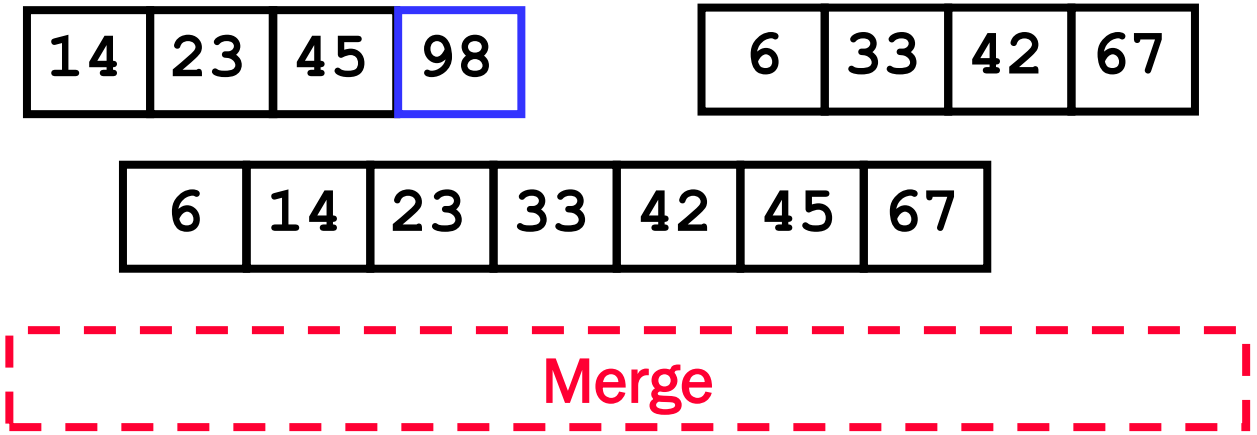

14
23
45
98
6
33
42
67
6
14
Merge

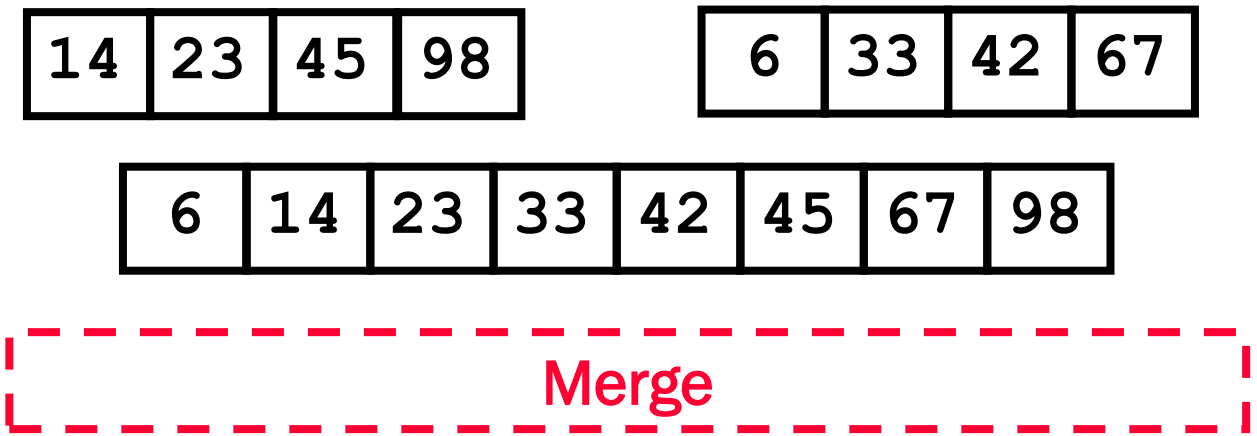

14 23 45 98
6 33 42 67
6 14 23
Merge


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


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


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


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


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

Merge Sort

- Merge sort keeps on dividing the list into equal halves until it can no more be divided.
- By definition, if it is only one element in the list, it is sorted. Then, merge sort combines the smaller sorted lists keeping the new list sorted too.

Algorithm

- Step 1 – if it is only one element in the list it is already sorted, return.
- Step 2 – divide the list recursively into two halves until it can no more be divided.
- Step 3 – merge the smaller lists into new list in sorted order.

Pseudocode

```
procedure mergesort( var a as array )
   if ( n == 1 ) return a

   var l1 as array = a[0] ... a[n/2]
   var l2 as array = a[n/2+1] ... a[n]

   l1 = mergesort( l1 )
   l2 = mergesort( l2 )

   return merge( l1, l2 )
end procedure
```

Pseudocode...

```
procedure merge( var a as array, var b as array )

  var c as array

  while ( a and b have elements )
    if ( a[0] > b[0] )
      add b[0] to the end of c
      remove b[0] from b
    else
      add a[0] to the end of c
      remove a[0] from a
    end if
  end while

  while ( a has elements )
    add a[0] to the end of c
    remove a[0] from a
  end while

  while ( b has elements )
    add b[0] to the end of c
    remove b[0] from b
  end while

  return c

end procedure
```

Recap

- Three intuitive sorting algorithms

Merge Sort

- Divide array in two equal parts
- Separately sort left and right half
- Combine the two sorted halves to get the full array sorted
- Given two sorted lists A and B, combine into a sorted list C
  - *Compare first element of A and B*
  - *Move it into C*
  - *Repeat until all elements in A and B are over*
- Merging A and B

Array Stack

- 32
- 74
- 89
- 21
- 55
- 64

Recap Merge Sort

- Sort A[0] to A[n/2-1]
- Sort A[n/2] to A[n-1]
- Merge sorted halves into B[0..n-1]
- How do we sort the halves?
  - *Recursively, using the same strategy!*

Divide and conquer

- Break up problem into disjoint parts
- Solve each part separately
- Combine the solutions efficiently

Merge Sort

- To sort A[0..n-1] into B[0..n-1]
- If n is 1, nothing to be done
- Otherwise
  - *Sort A[0..n/2-1] into L (left)*
  - *Sort A[n/2..n-1] into R (right)*
  - *Merge L and R into B*

Merging sorted lists

- Combine two sorted lists L and R into B
  - *compare first element of L and R and*
  - *move the smaller of the two into B*
  - *Repeat until all elements in L and R have been moved*
- Otherwise,
  - *If L is empty, copy R into C*
  - *If R is empty, copy L into C*

```
functionMergeSort(A,left,right,B)
// Sort the segment A[left..right-1] into B
if (right - left == 1) // Base case
B[0] = A[left]
if (right - left > 1) // Recursive call
mid = (left+right)/2
MergeSort(A,left,mid-1,L)
MergeSort(A,mid,right-1,R)
Merge(L,mid-left,R,right-mid,B)
```

```
function Merge(A,m,B,n,C)
 // Merge A[0..m-1], B[0..n-1] into C[0..m+n-1]
i = 0; j = 0; k = 0;
// Current positions in A,B,C respectively
while (k < m+n)
// Case 1: Move head of A into C
 if (A[i] <= B[j])
 { C[k] = A[i]; i++; k++;}
// Case 2: Move head of B into C
 if (A[i] > B[j])
{C[k] = B[j]; j++; k++;
// Case 0: One of the two lists is empty
 if (i==m) {j++; k++;}
 if (j==n) {i++; k++;}
}
```

Variations on merge

- Union of two sorted lists (discard duplicates)
- [1,2,3] [2,3,6]
- If A[i] == B[j], copy A[i] to C[k] and increment i,j,k
- Intersection of two sorted lists
- If A[i] < B[j], increment i
- If B[j] < A[i], increment j
- If A[i] == B[j], copy A[i] to C[k] and increment i,j,k
- Exercise: List difference: elements in A but not in B
- [1,2,3] [3,4,6] = [1,2]

Merge Sort: Shortcomings

- Merging A and B creates a new array C
  - *No obvious way to efficiently merge in place*
- Extra storage can be costly
- Inherently recursive
  - *Recursive call and return are expensive*

Alternative approach

- Extra space is required to merge
- Merging happens because elements in left half must move right and vice versa
- [2,4,6] [1,3,5]
- Can we divide so that everything to the left is smaller than everything to the right?
  - *No need to merge!*
  - *[2,1,3] [6,4,5]*