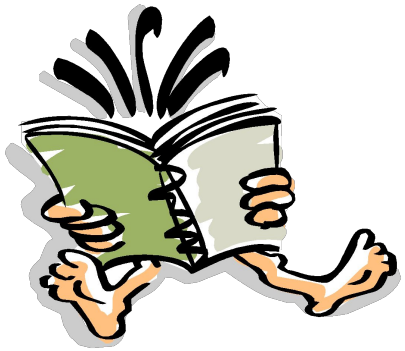
Divide and Conquer
Merge Sort

Ref Book: Computer Algorithms

By Sartaj Sahni



Nirvruti Patel ,IT Department,UVPCE

Divide & Conquer Design Technique

- Divide
  - Divide the problem into a number of sub problems.
- Conquer
  - Solve the sub problems recursively
- Combine
  - Combine the solutions of sub problems

Divide-and-Conquer

- **Divide** the problem into a number of sub-problems
  - Similar sub-problems of smaller size
- **Conquer** the sub-problems
  - Solve the sub-problems recursively
  - Sub-problem size small enough ⇒ solve the problems in straightforward manner
- **Combine** the solutions of the sub-problems
  - Obtain the solution for the original problem

Sorting

- Insertion sort
  - Design approach: incremental
  - Sorts in place: Yes
  - Best case: $\Theta(n)$
  - Worst case: $\Theta(n^2)$

- Bubble Sort
  - Design approach: incremental
  - Sorts in place: Yes
  - Best case $\Theta(n)$
  - Worst case: $\Theta(n^2)$

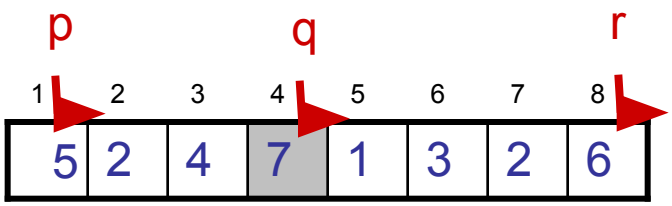
Sorting

- Selection sort
  - Design approach: incremental
  - Sorts in place: Yes
  - Running time: $\Theta(n^2)$
- Merge Sort
  - Design approach: divide and conquer
  - Sorts in place: No
  - Running time: Let's see!!

Merge Sort Approach

- To sort an array $A[p \,..\, r]$:
- **Divide**
  - Divide the n-element sequence to be sorted into two subsequences of $n/2$ elements each
- **Conquer**
  - Sort the subsequences recursively using merge sort
  - When the size of the sequences is 1 there is nothing more to do
- **Combine**
  - Merge the two sorted subsequences

Merge Sort



Alg.: MERGE-SORT(A, p, r)

if p < r Check for base case

then $q \leftarrow \lfloor (p + r)/2 \rfloor$ Divide

MERGE-SORT(A, p, q) Conquer

MERGE-SORT(A, q + 1, r) Conquer

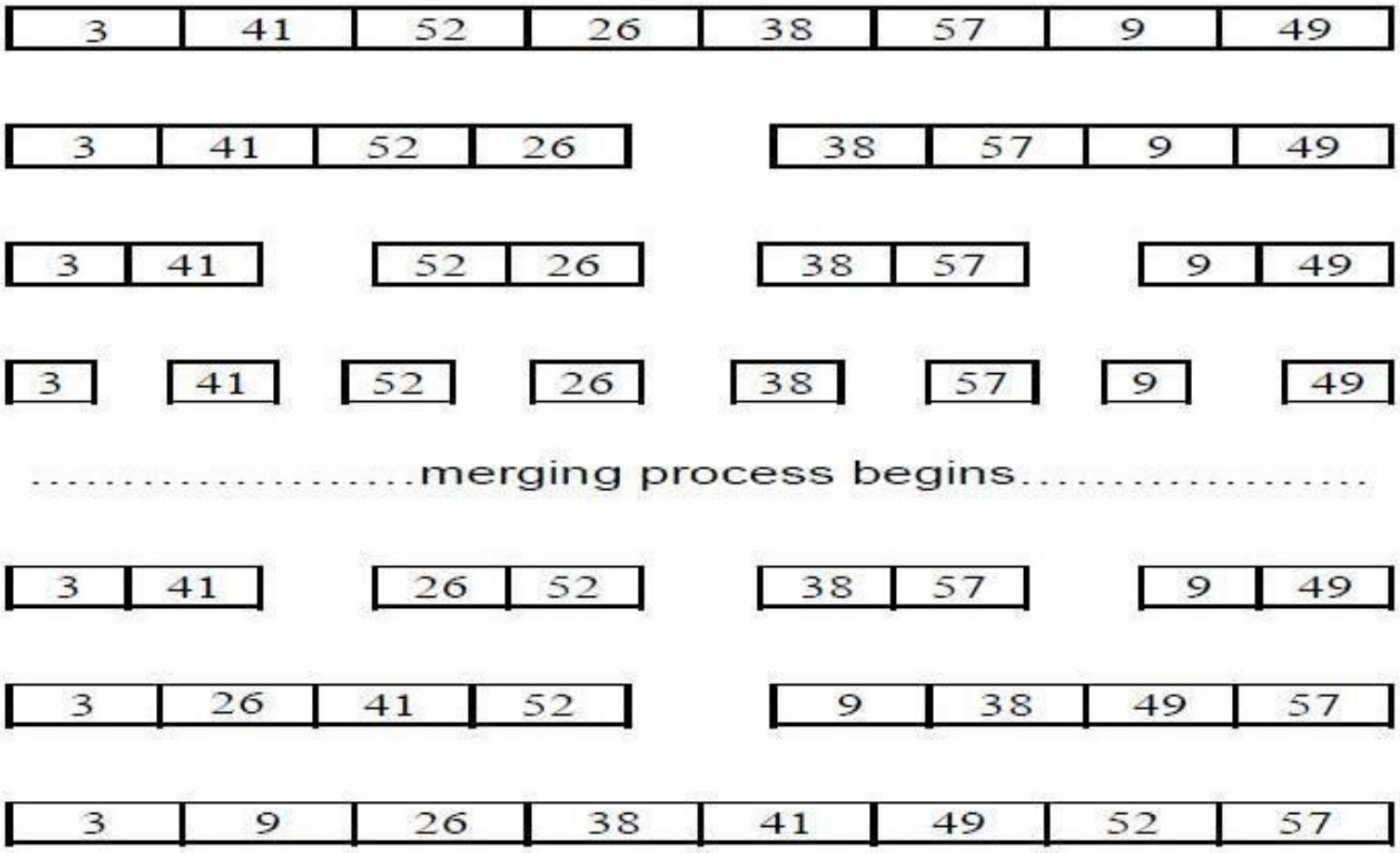
MERGE(A, p, q, r) Combine

- Initial call: MERGE-SORT(A, 1, n)

Merge Sort

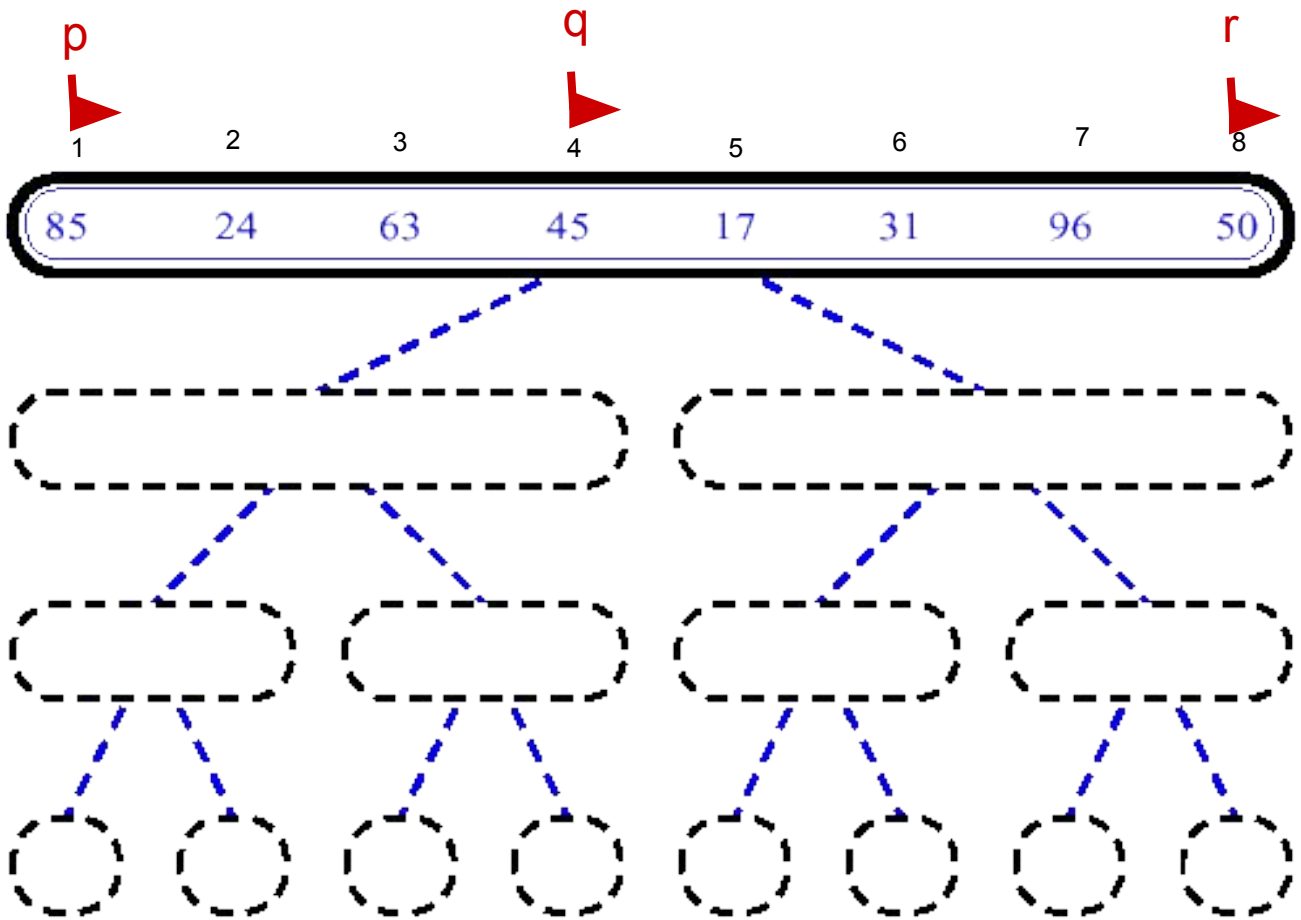
```
Algorithm MERGE(A , p , q , r ) {
Let i = p and j = q+1 and k=1
while( i <= q) and ( j <= r )
        do                  if A[ i ] <= A[ j ]
                        then B[ k ] = A[ i ]
                          i = i + 1,  k =  k + 1
                else B[ k ] =A[ j ]
                        j = j + 1,  k= k + 1
                  end of while
//here one of the subarray is in B
if  i  > q then
  for index = j  to r
do  B[ k ] = A[ index ]
                          k = k + 1
else for index = i  to q
do   B[ k ] = A[ index]
        k = k +1
x=1;
for index= p to r
 do  A[ index ]=B[ x++ ]
return
}
```

Merge Sort

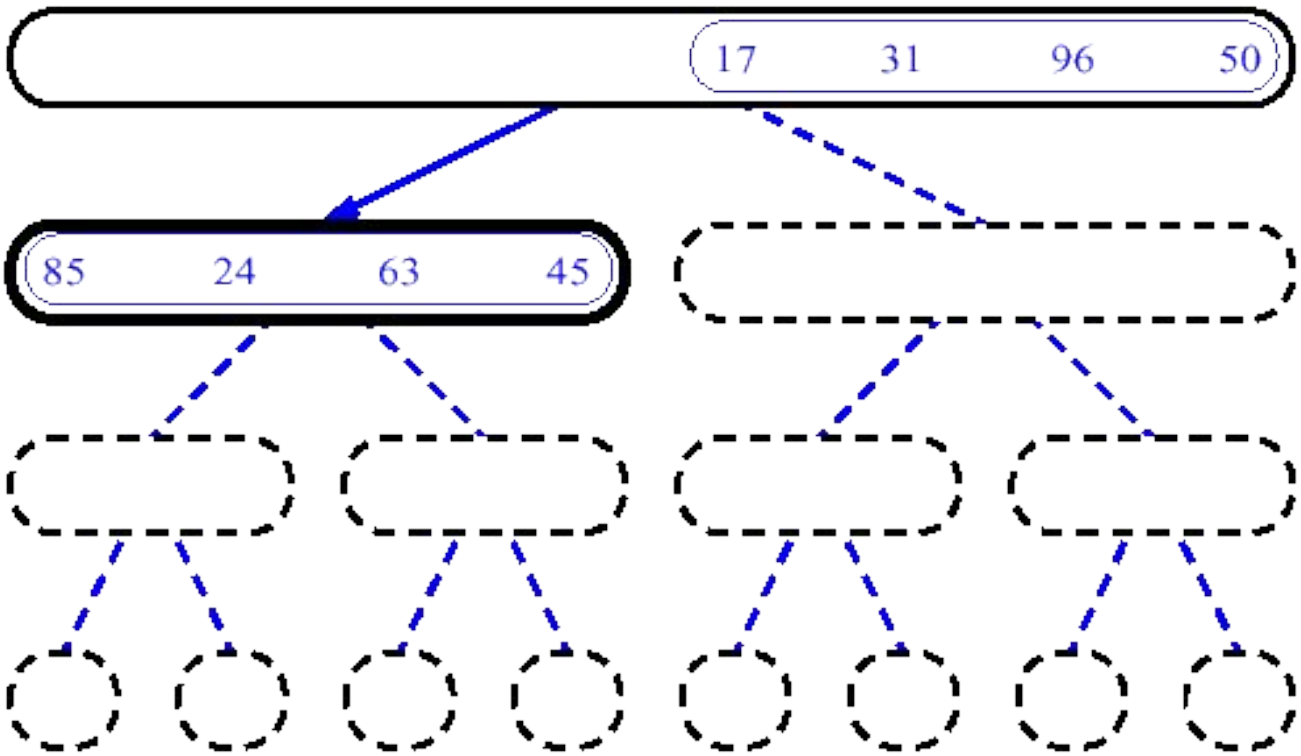

3 41 52 26 38 57 9 49
3 41 52 26 38 57 9 49
3 41 52 26 38 57 9 49
3 41 52 26 38 57 9 49
.....merging process begins.....
3 41 26 52 38 57 9 49
3 26 41 52 9 38 49 57
3 9 26 38 41 49 52 57

MergeSort (Example) - 1

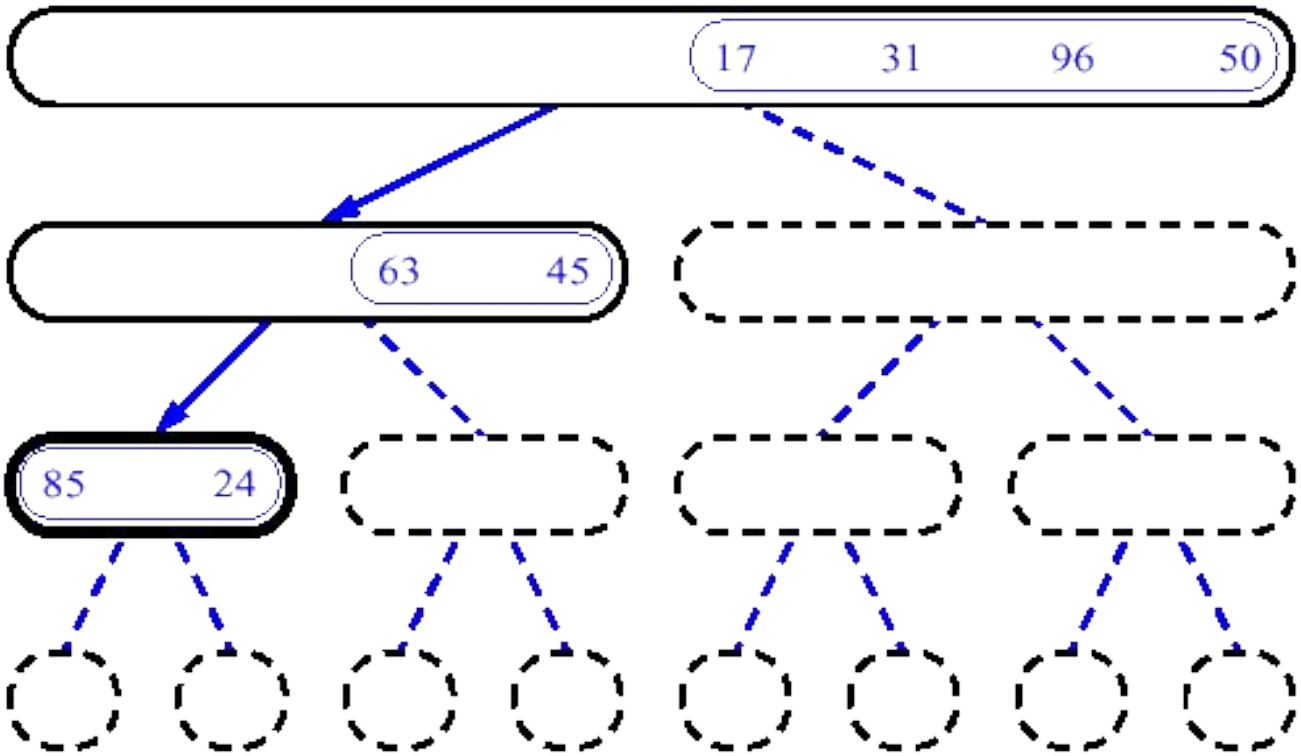
q = 4



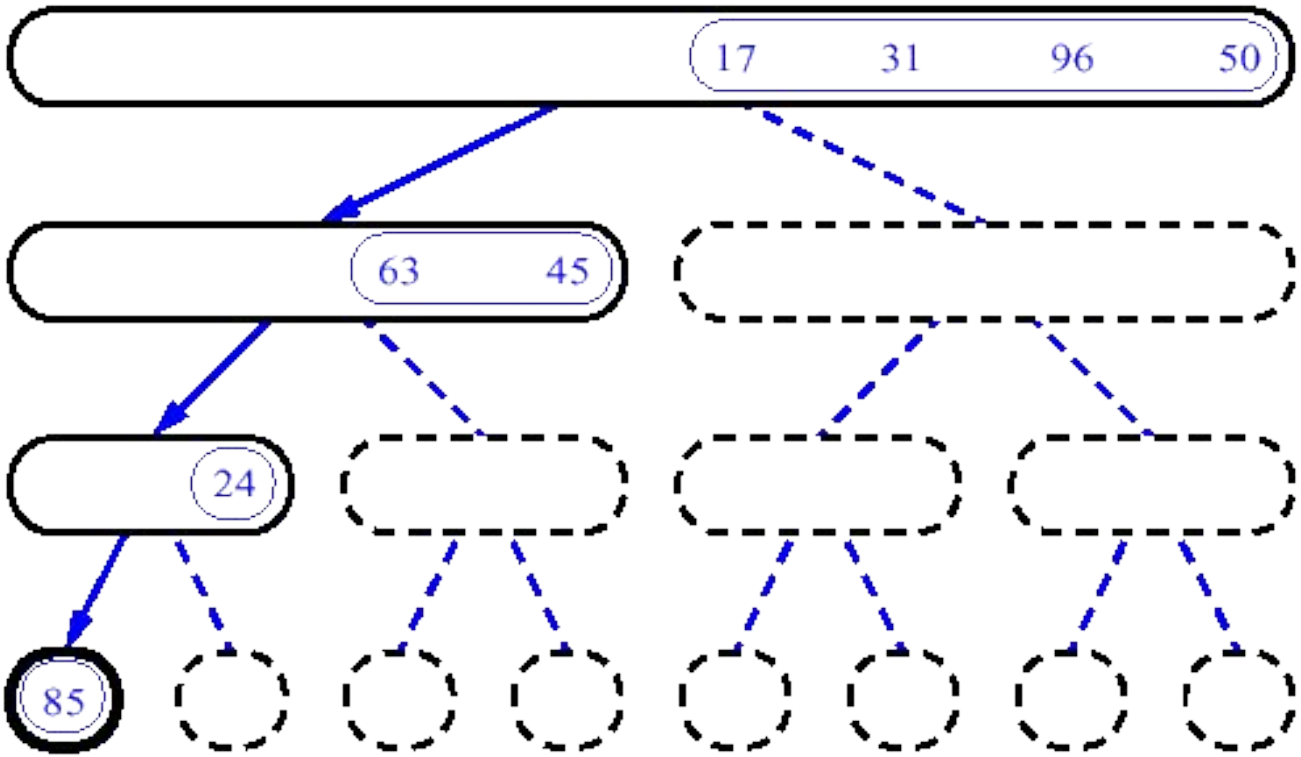
MergeSort (Example) - 2



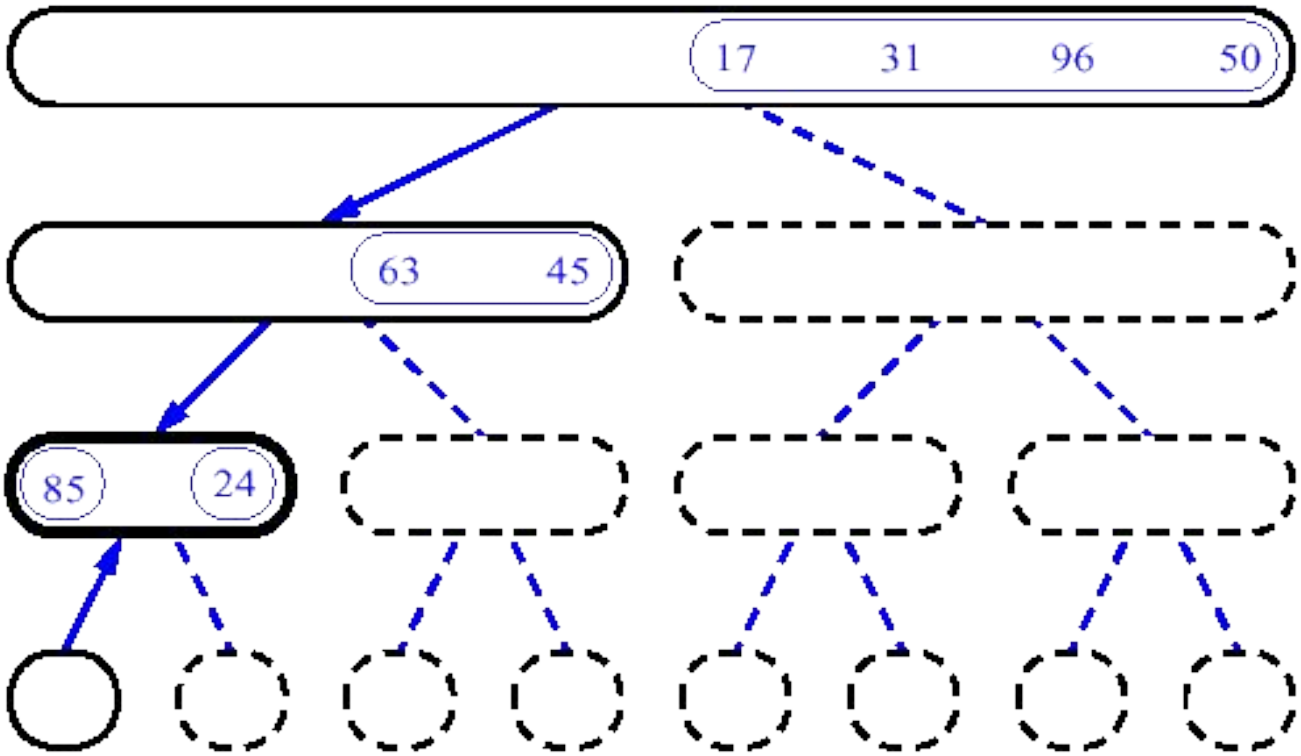
MergeSort (Example) - 3



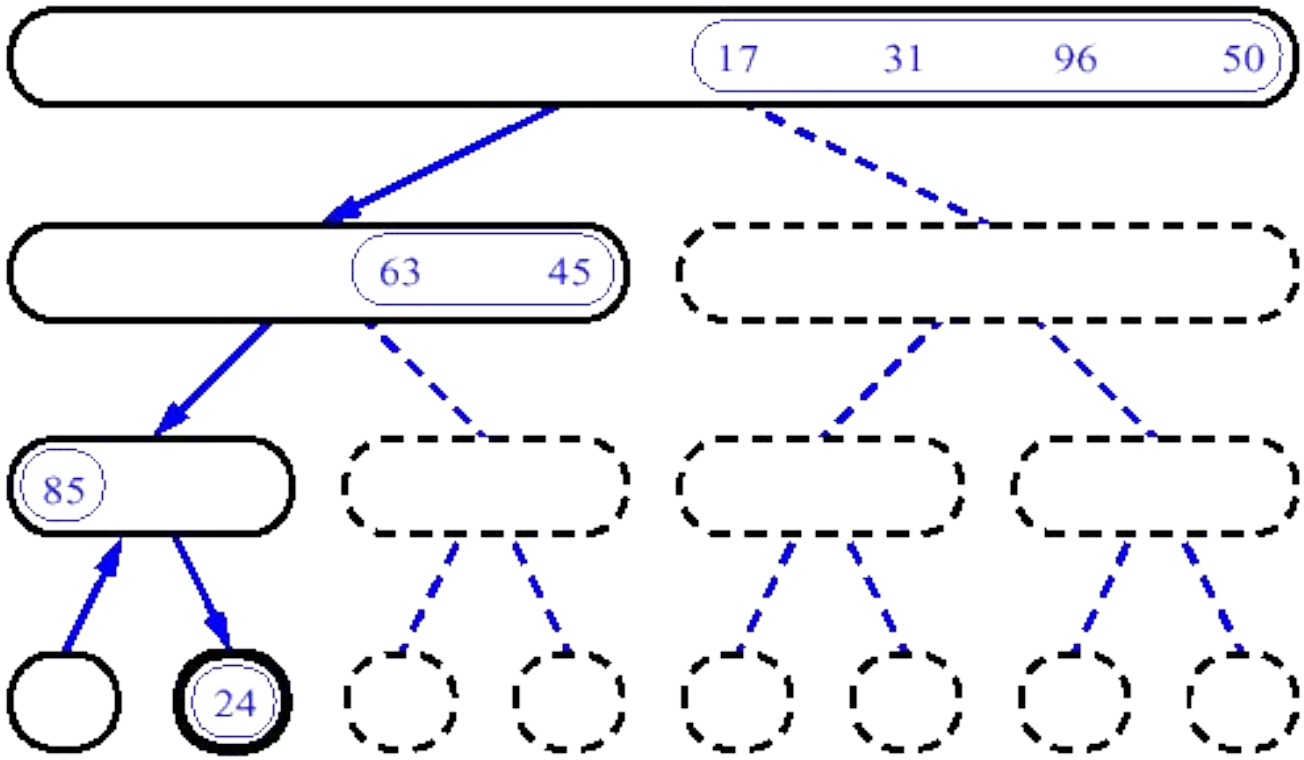
MergeSort (Example) - 4



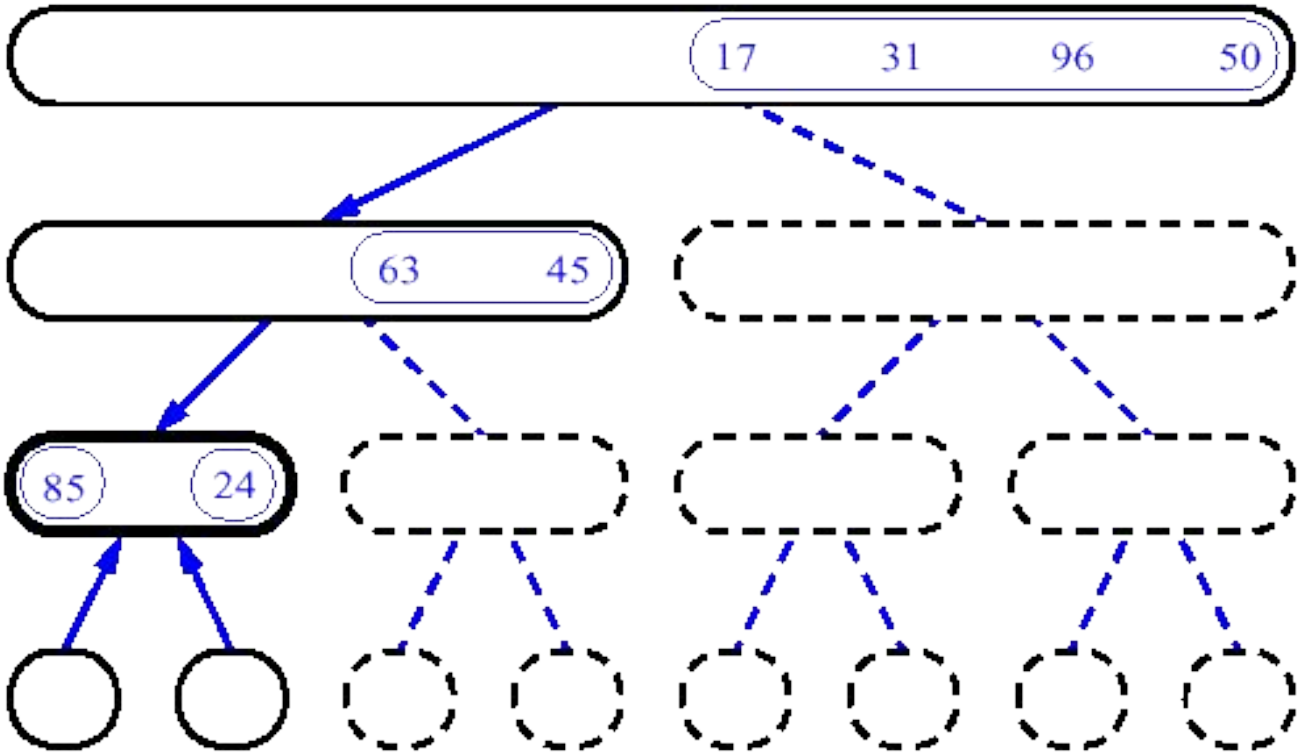
MergeSort (Example) - 5



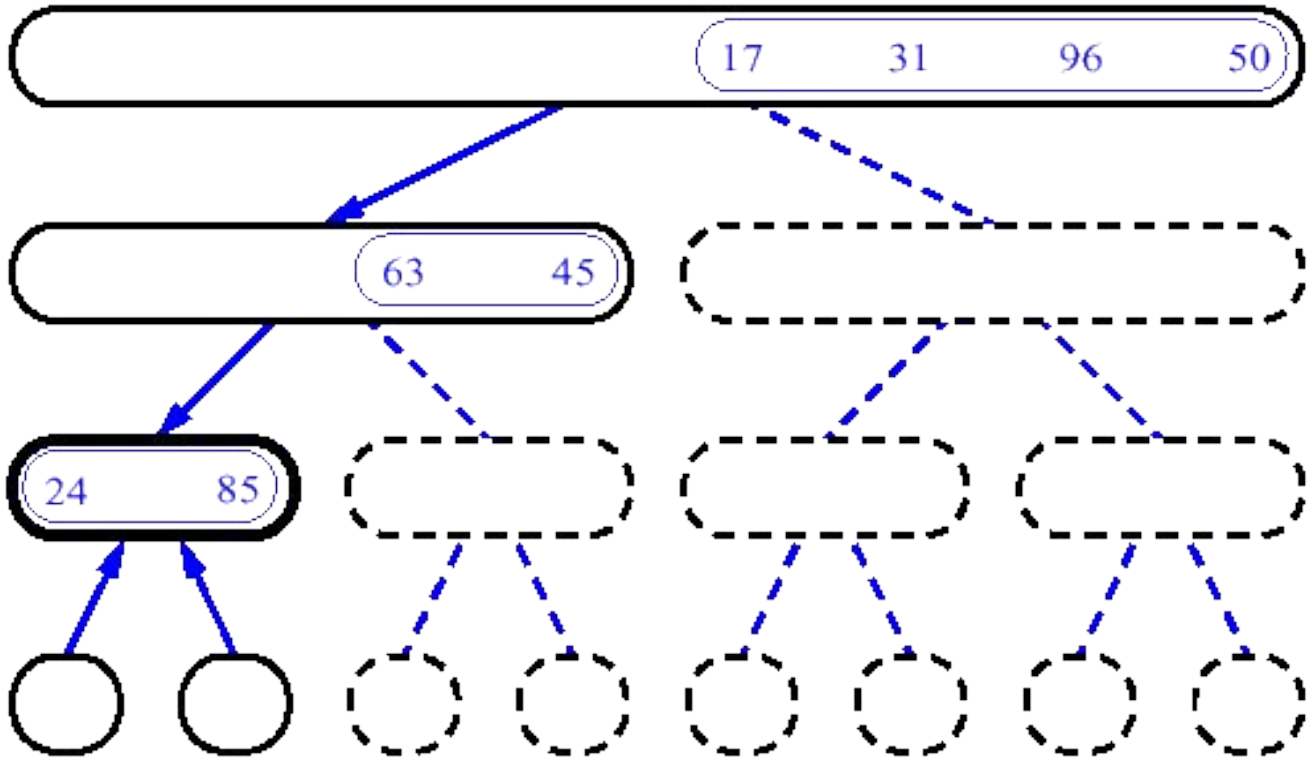
MergeSort (Example) - 6



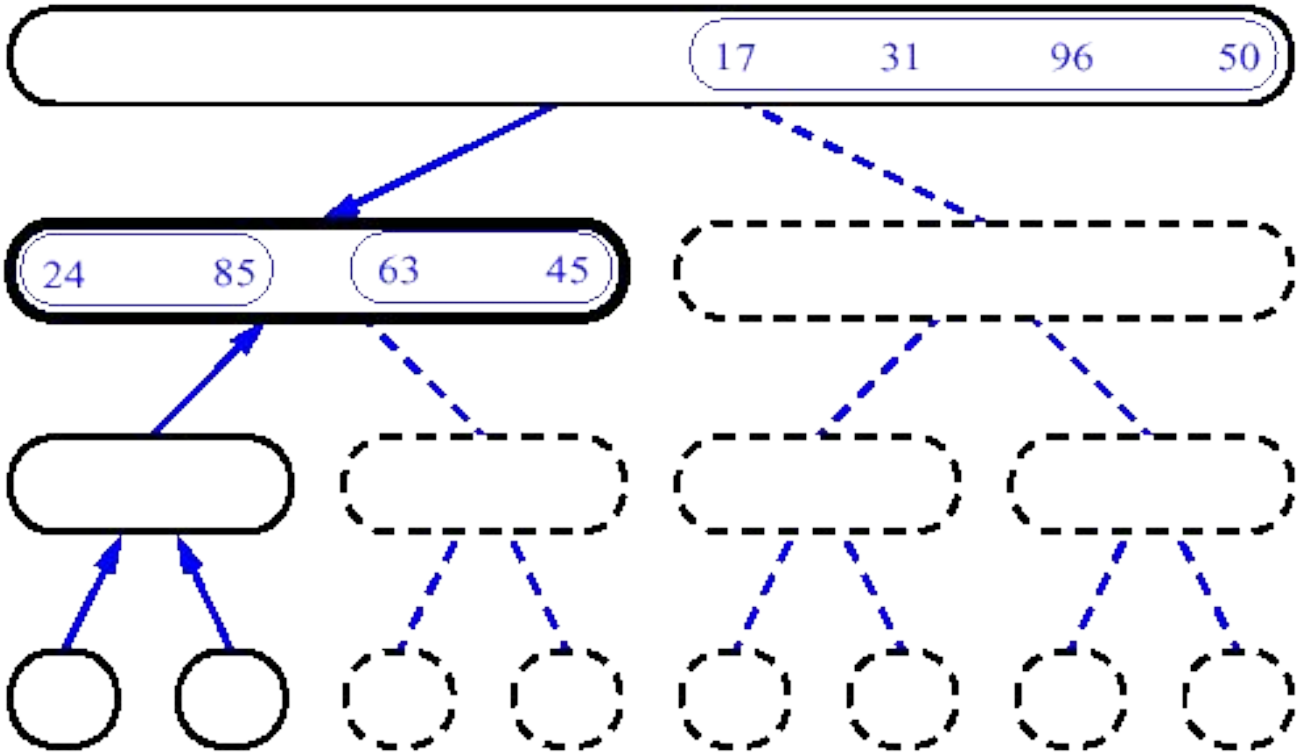
MergeSort (Example) - 7



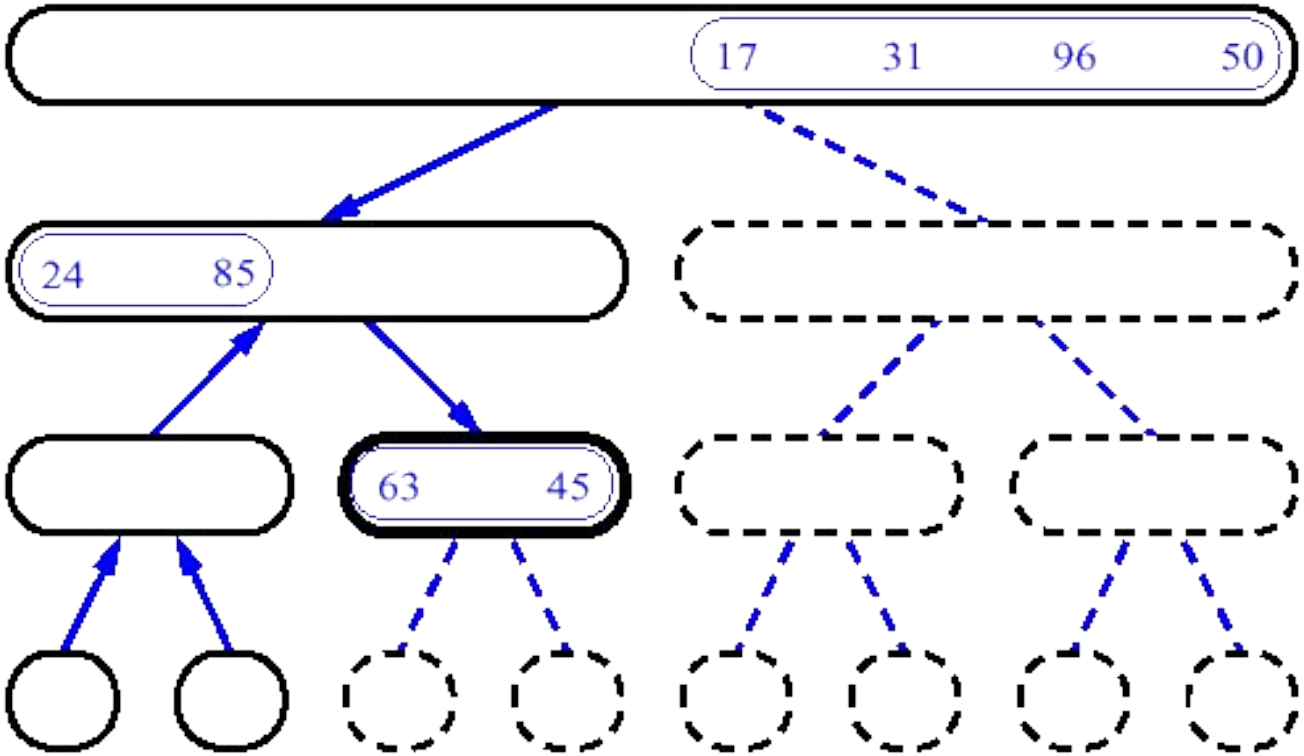
MergeSort (Example) - 8



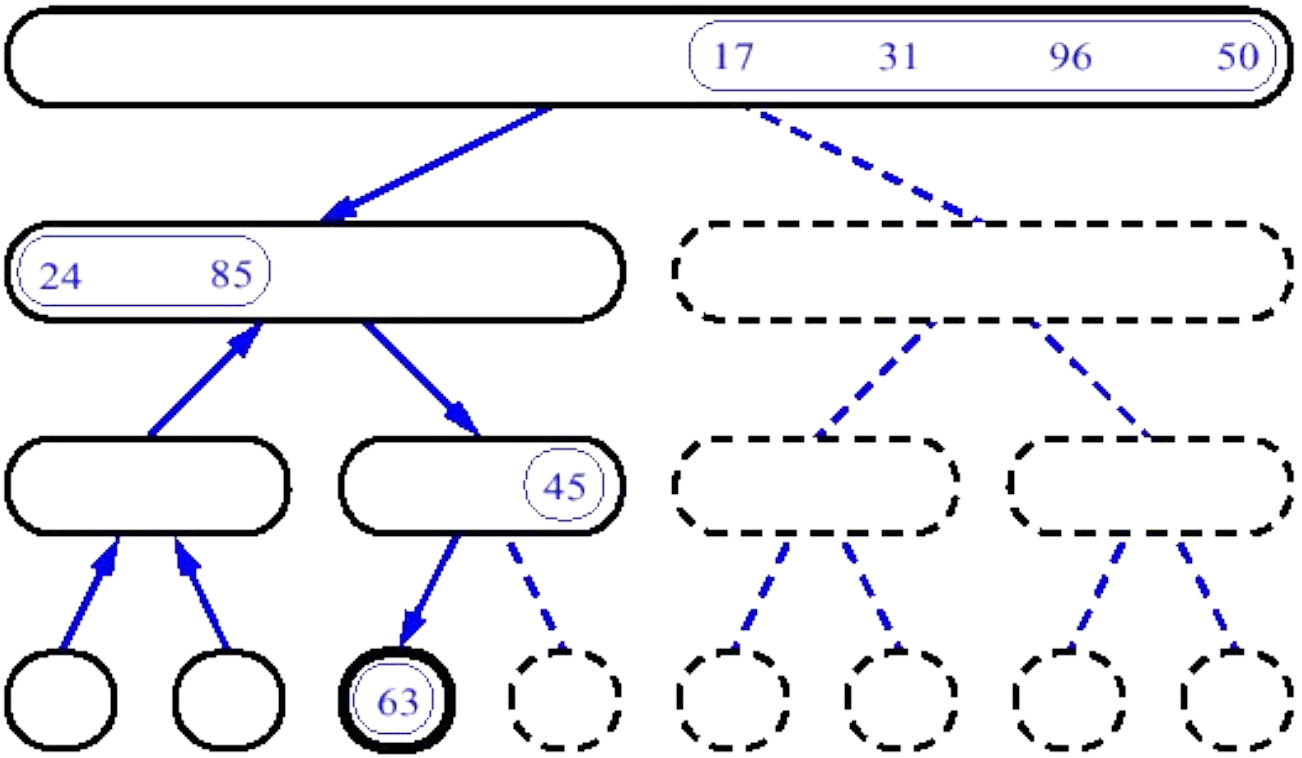
MergeSort (Example) - 9



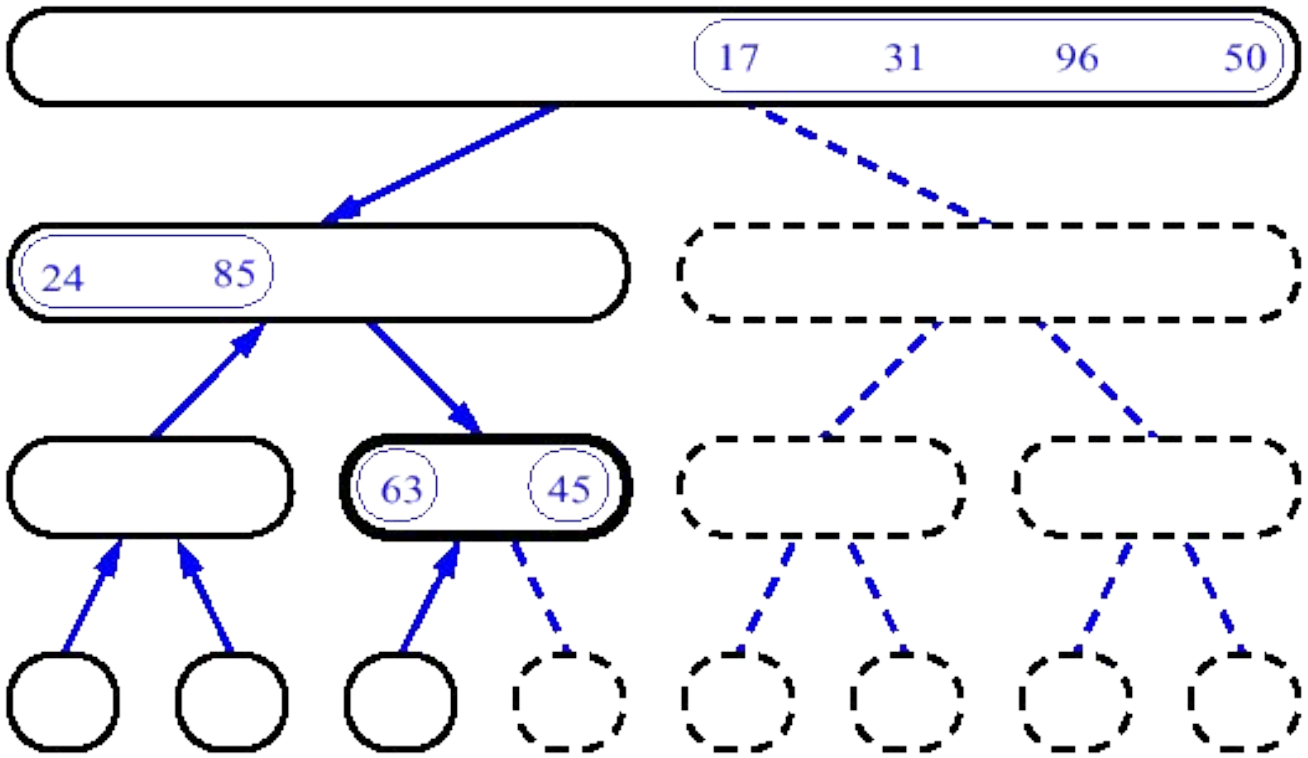
MergeSort (Example) - 10



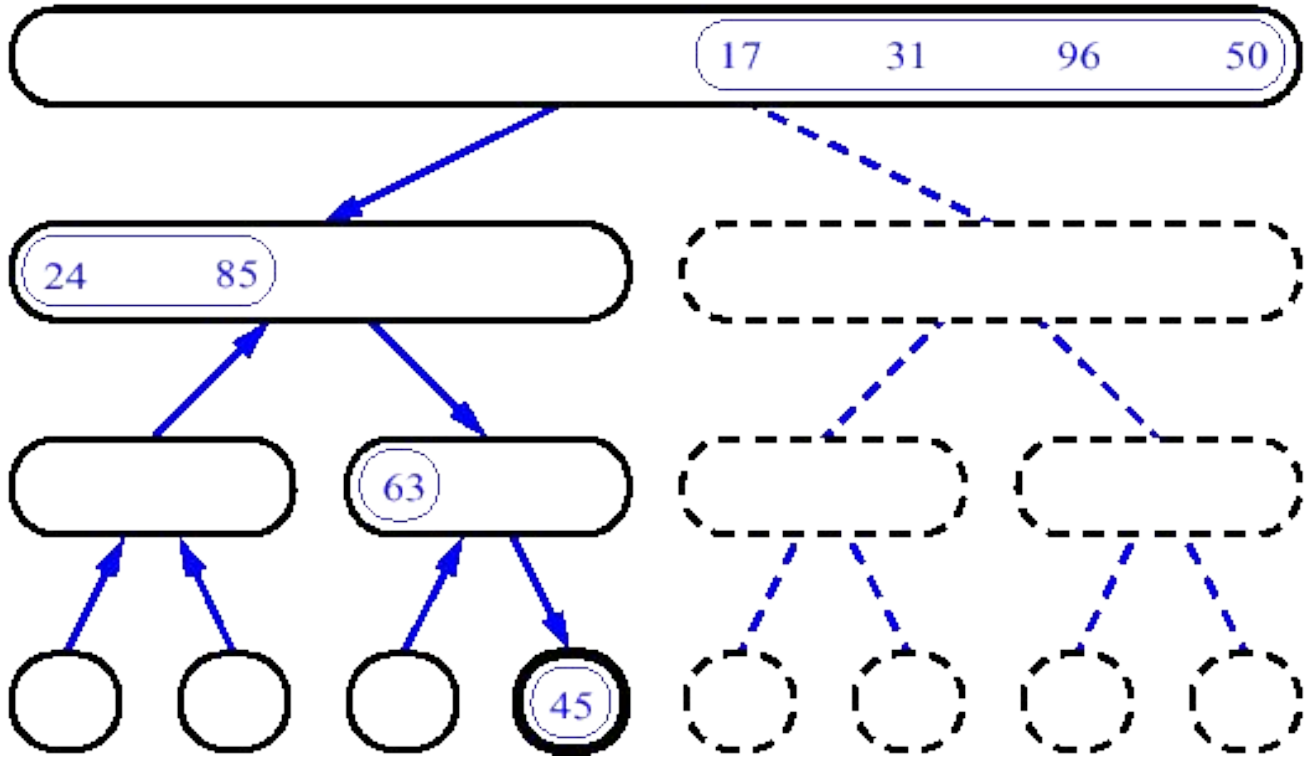
MergeSort (Example) - 11



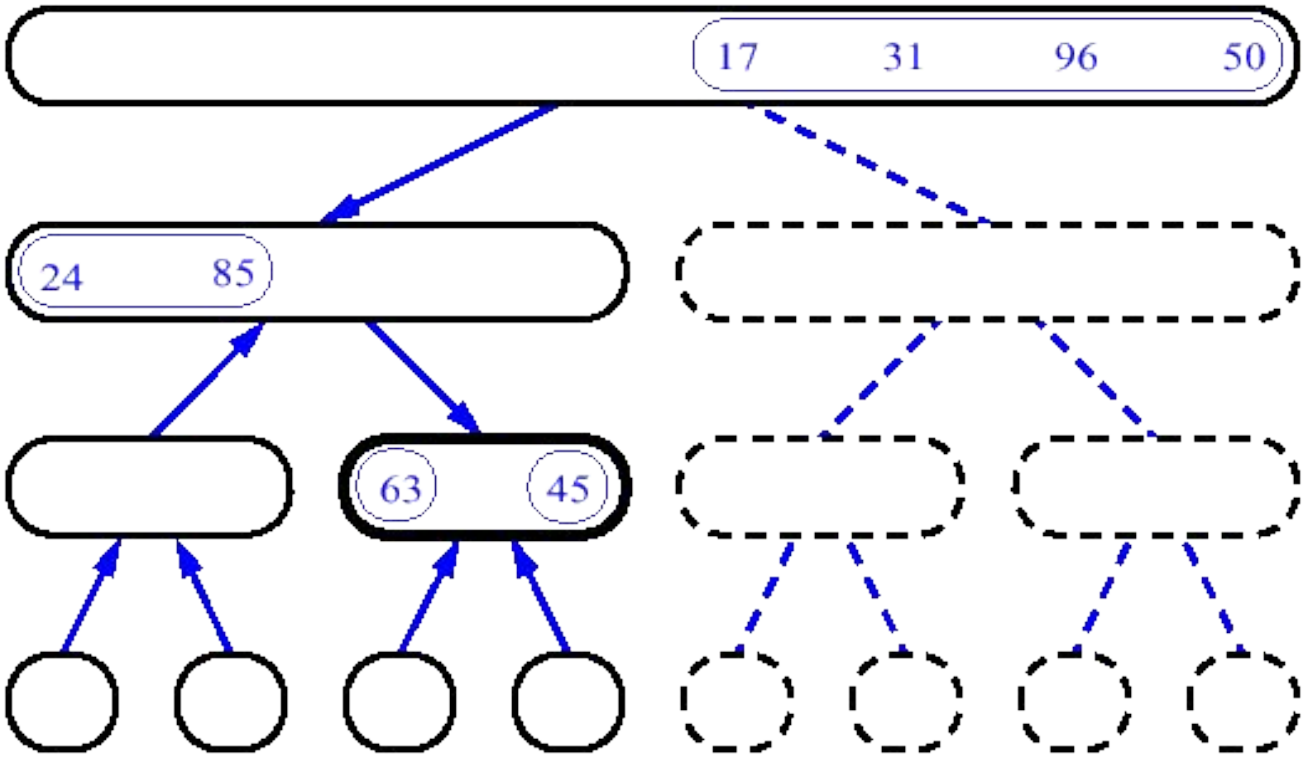
MergeSort (Example) - 12



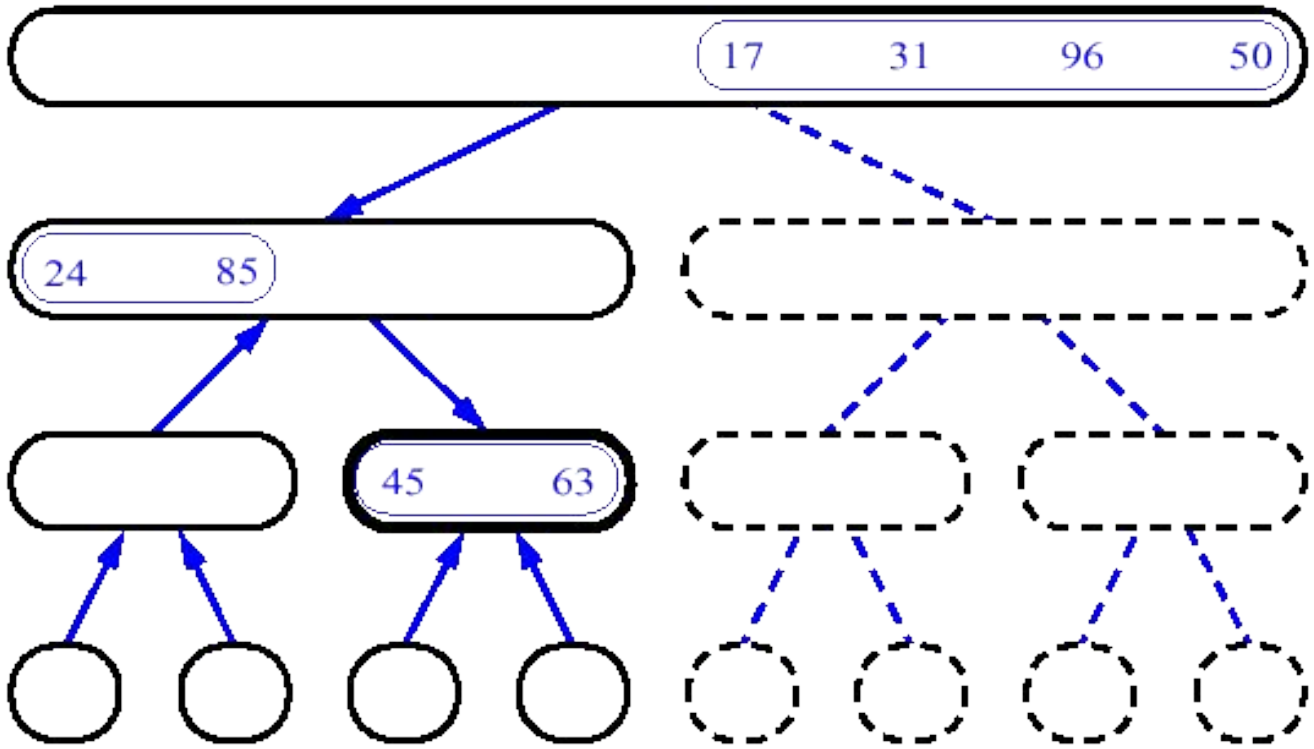
MergeSort (Example) - 13



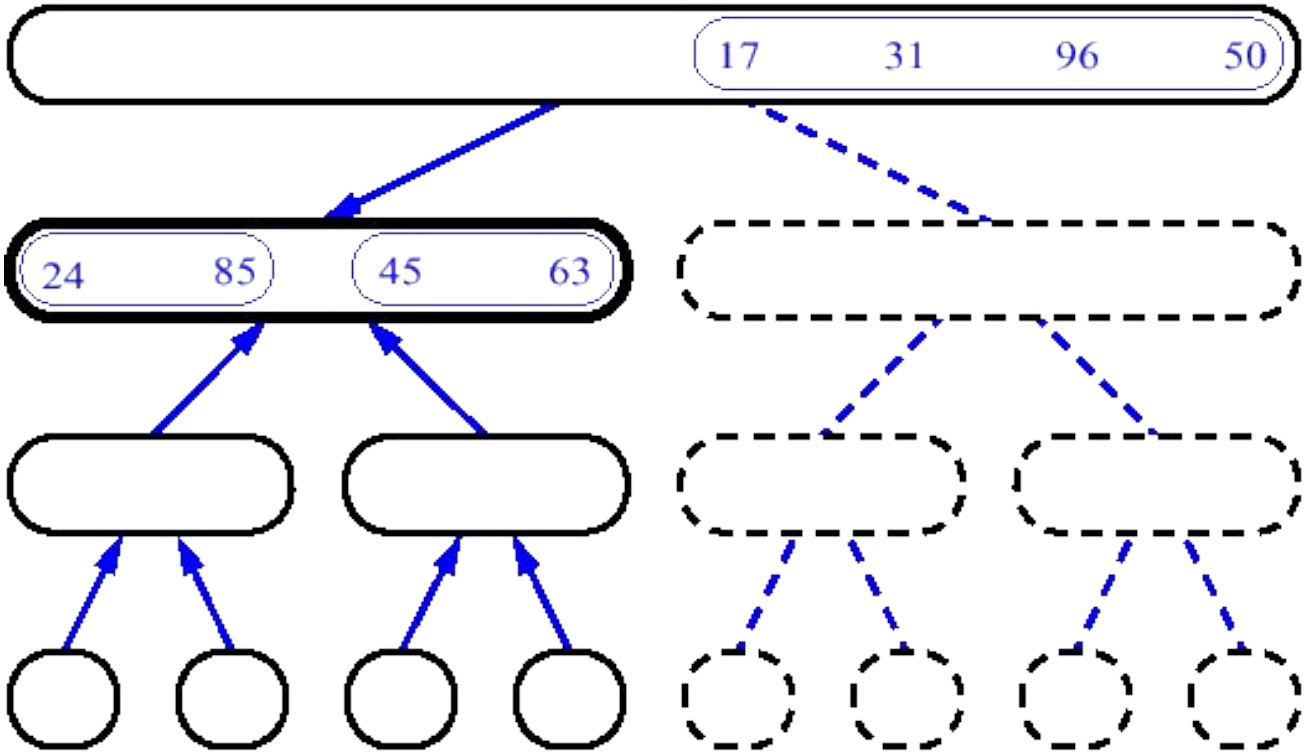
MergeSort (Example) - 14



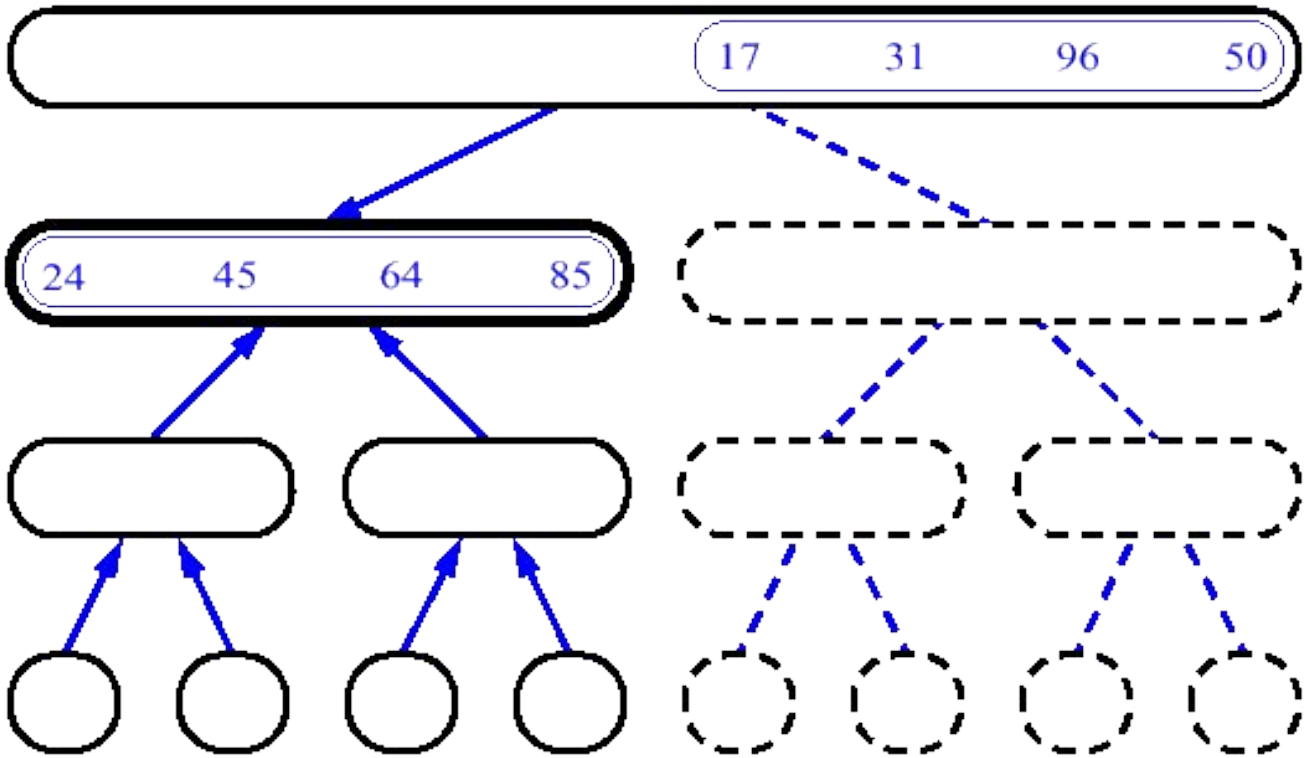
MergeSort (Example) - 15



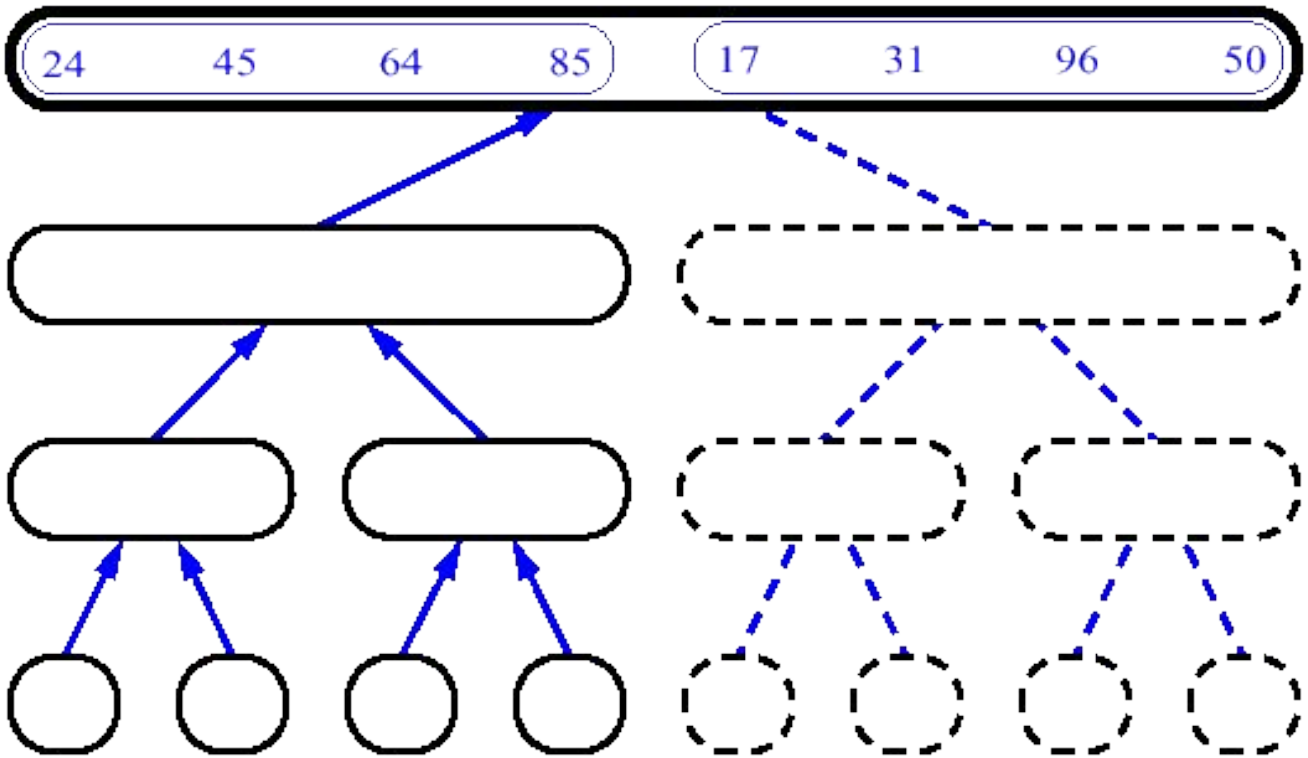
MergeSort (Example) - 16



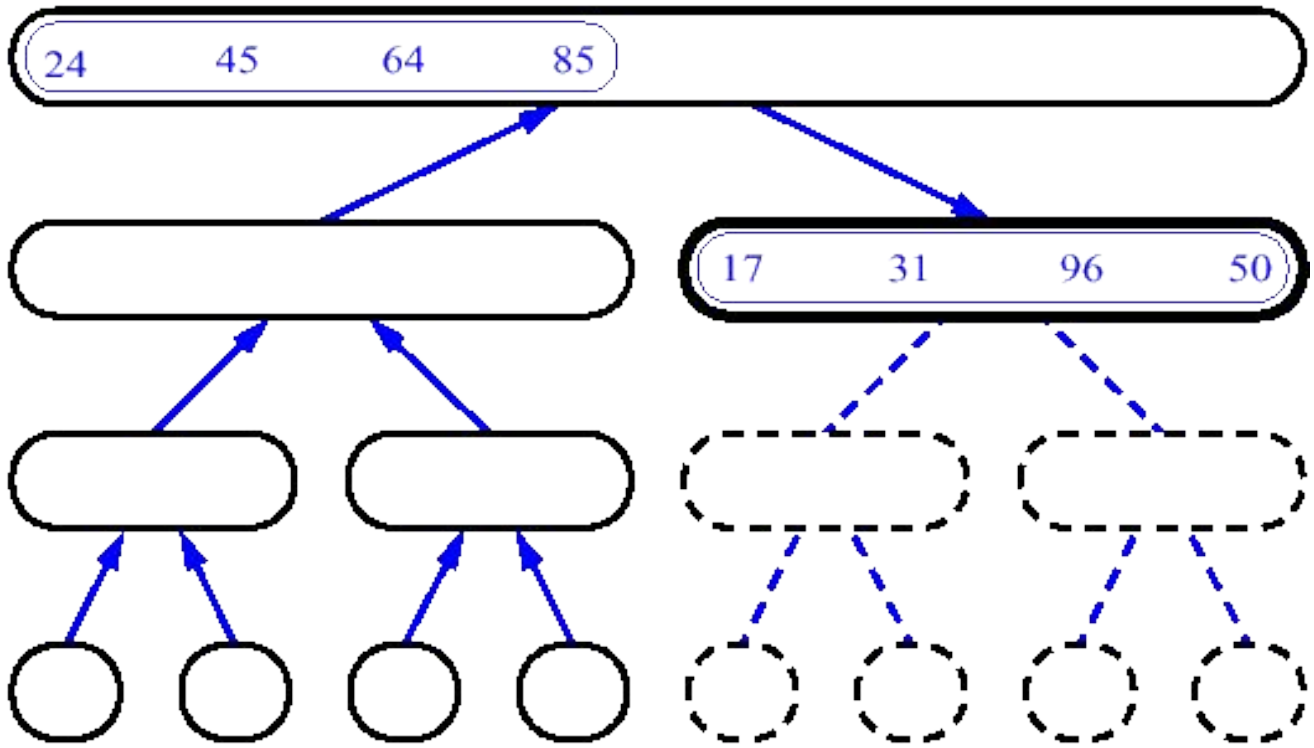
MergeSort (Example) - 17



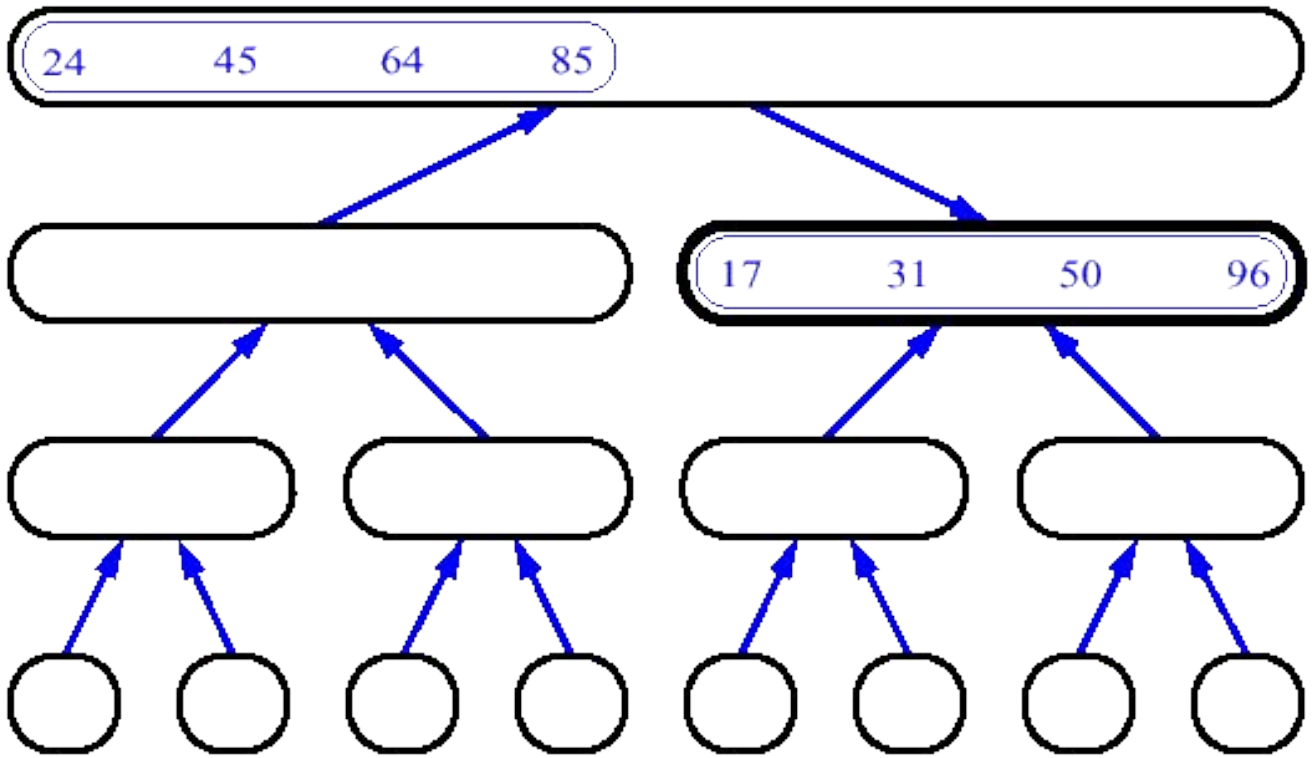
MergeSort (Example) - 18



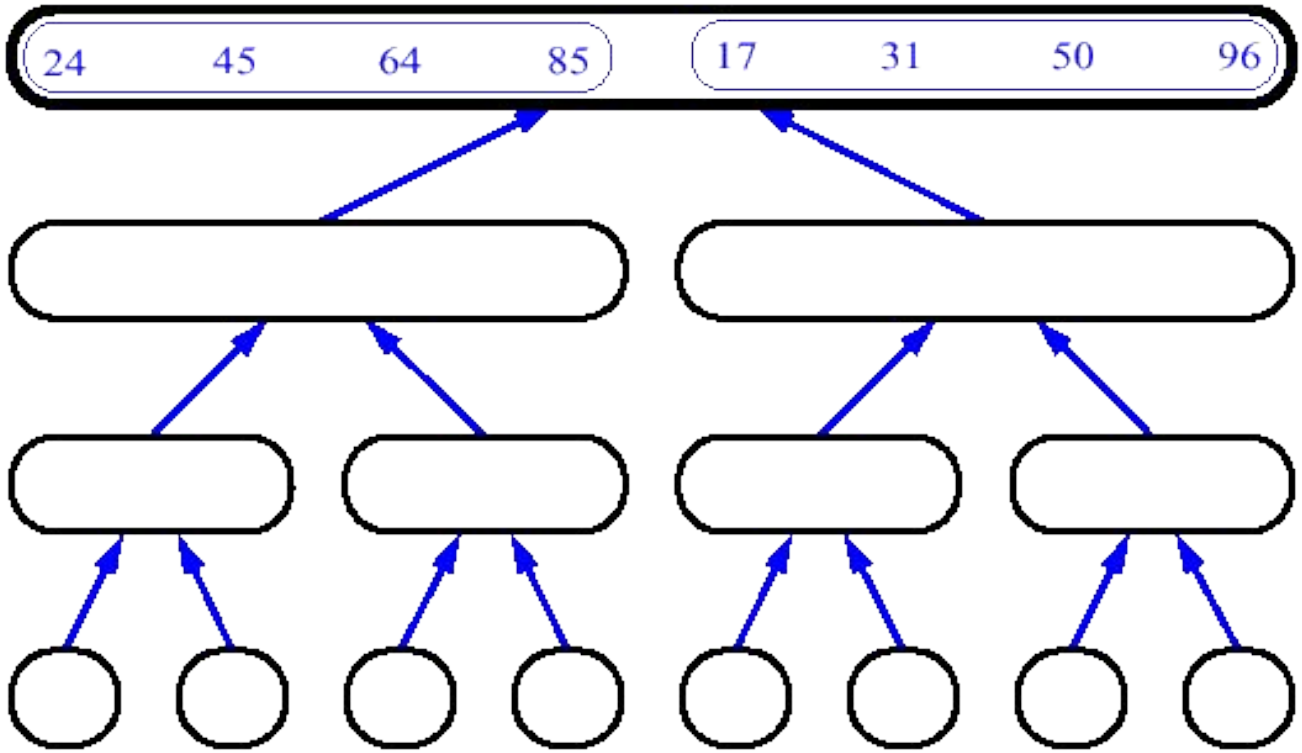
MergeSort (Example) - 19



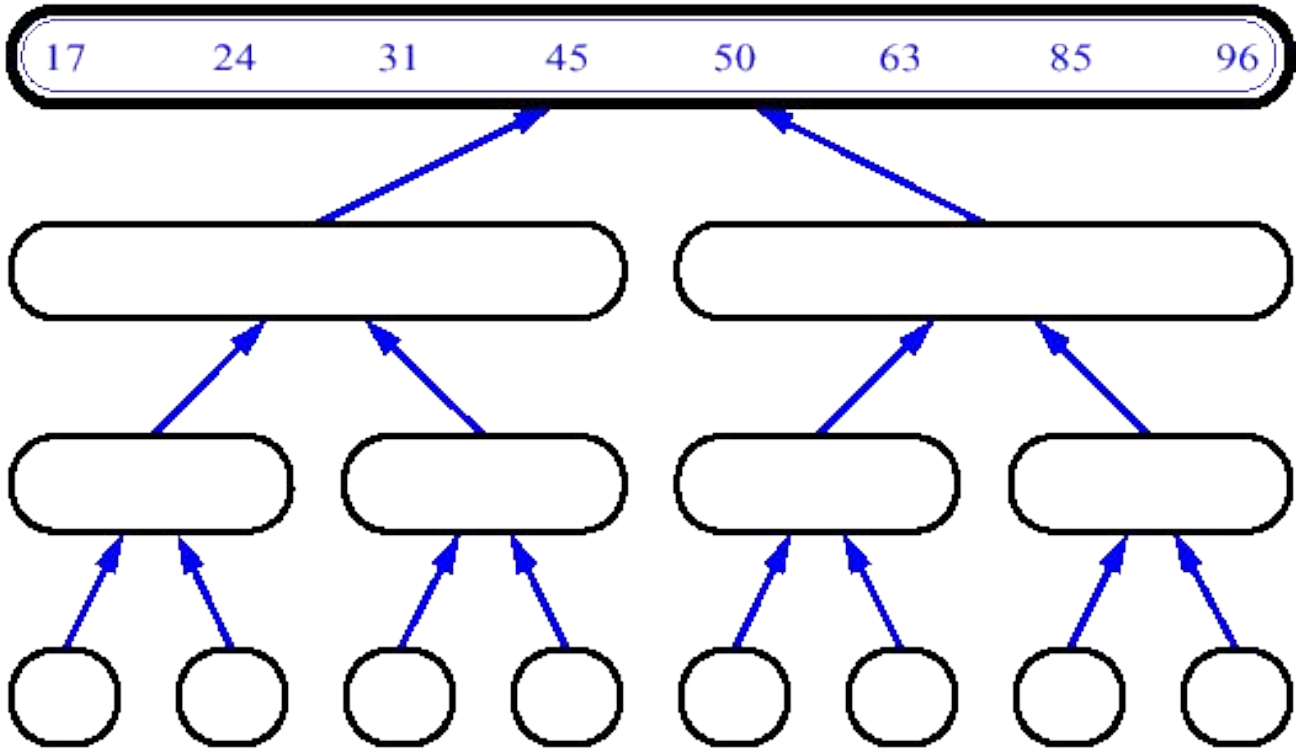
MergeSort (Example) - 20



MergeSort (Example) - 21

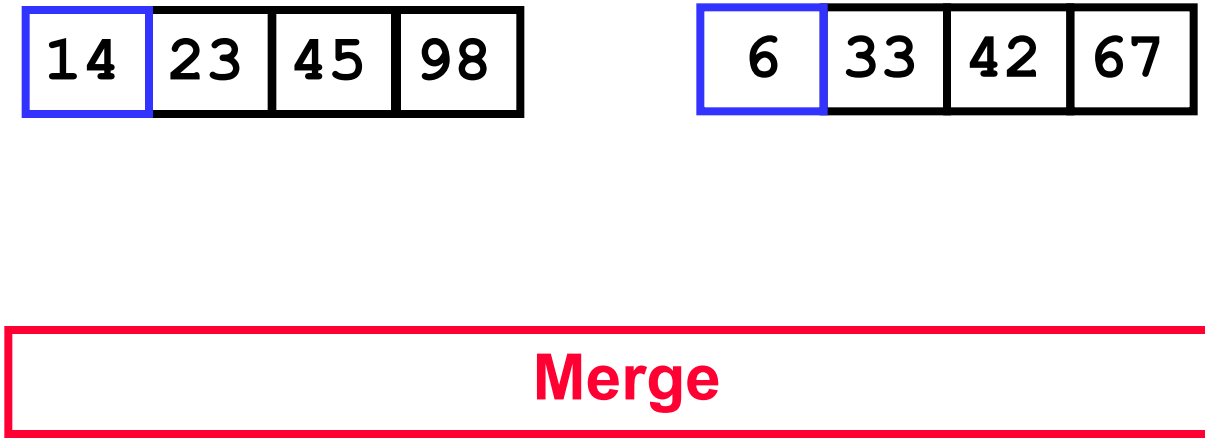


MergeSort (Example) - 22

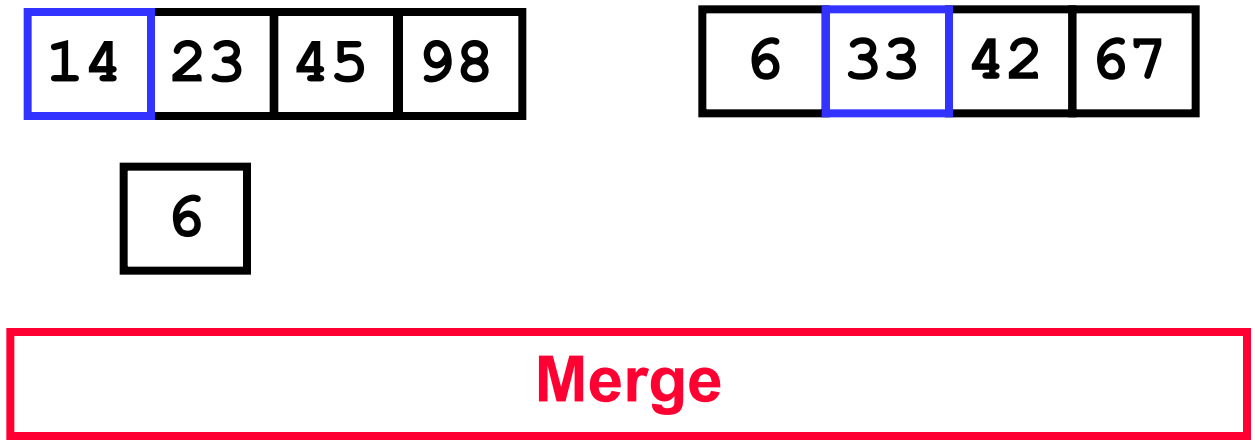


Merging Process

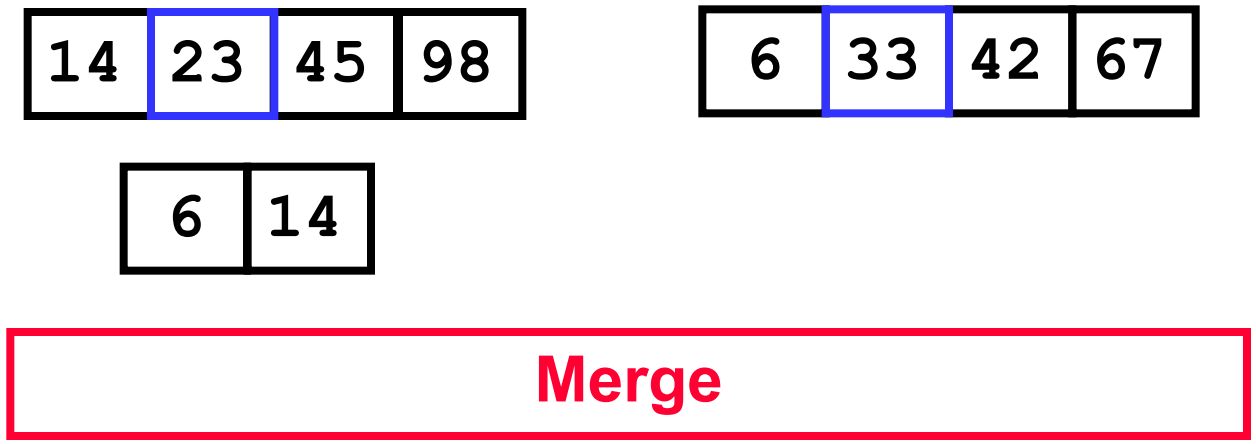
Merging Process



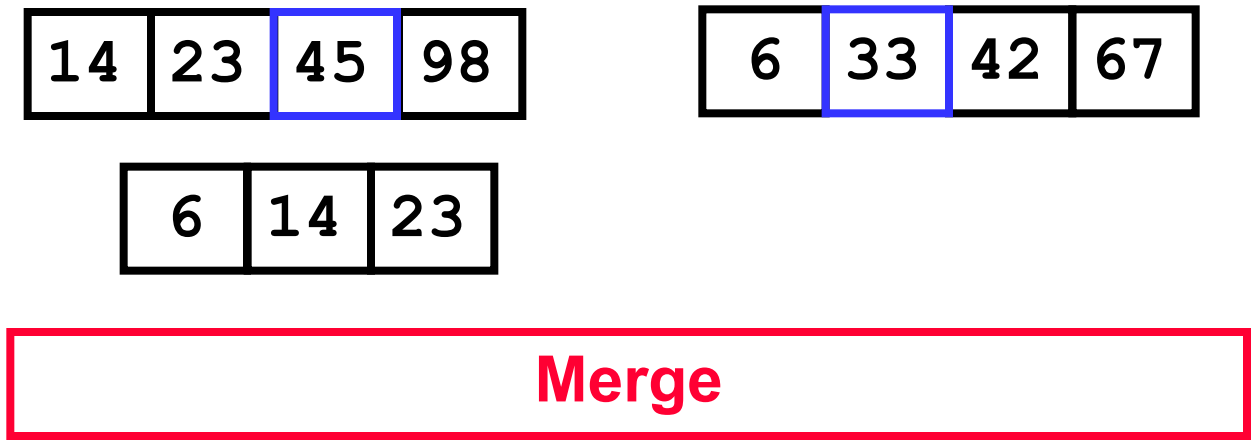
Merging Process



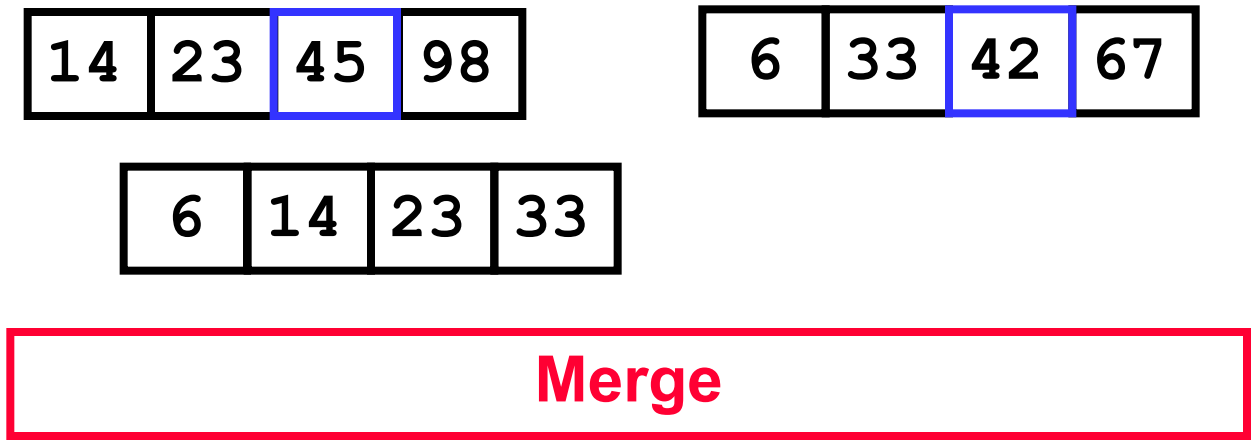
Merging Process



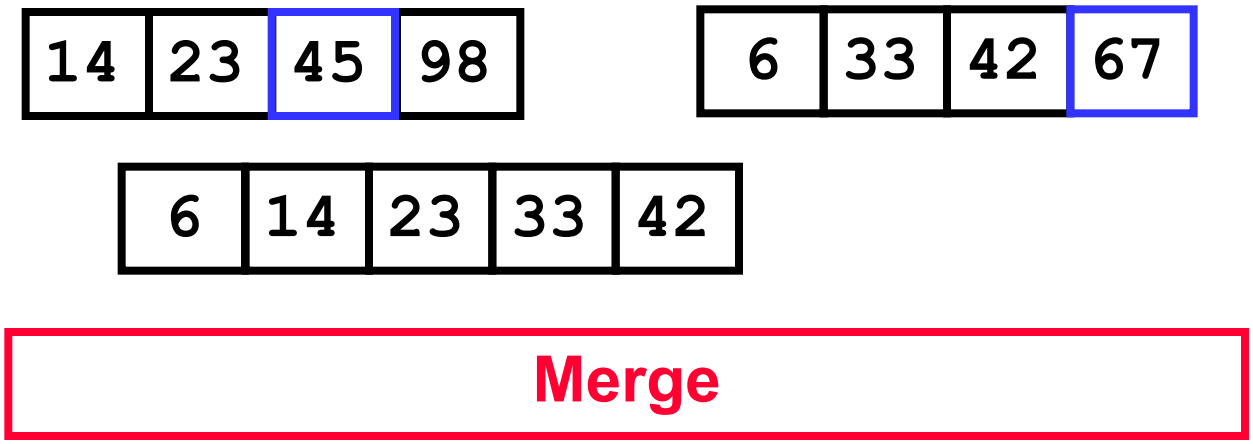
Merging Process



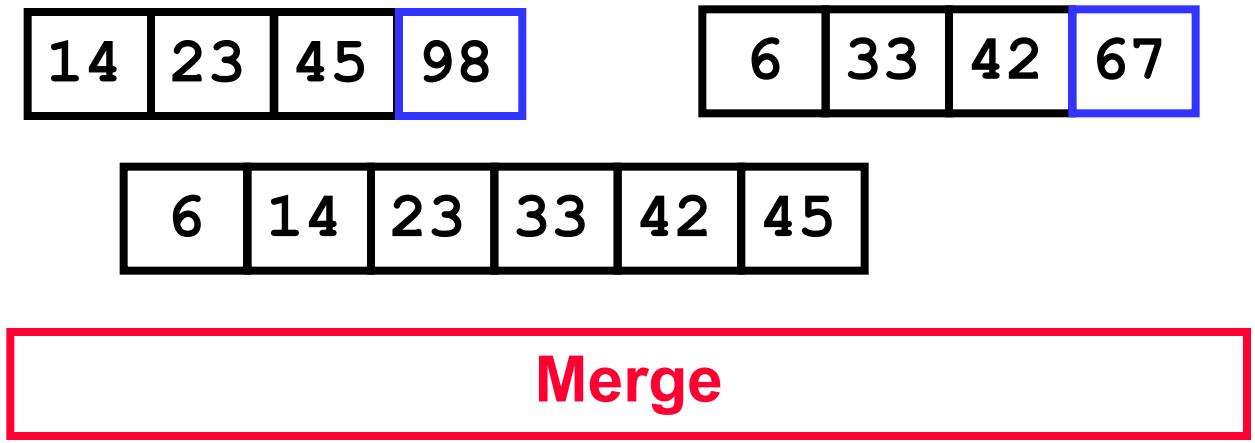
Merging Process



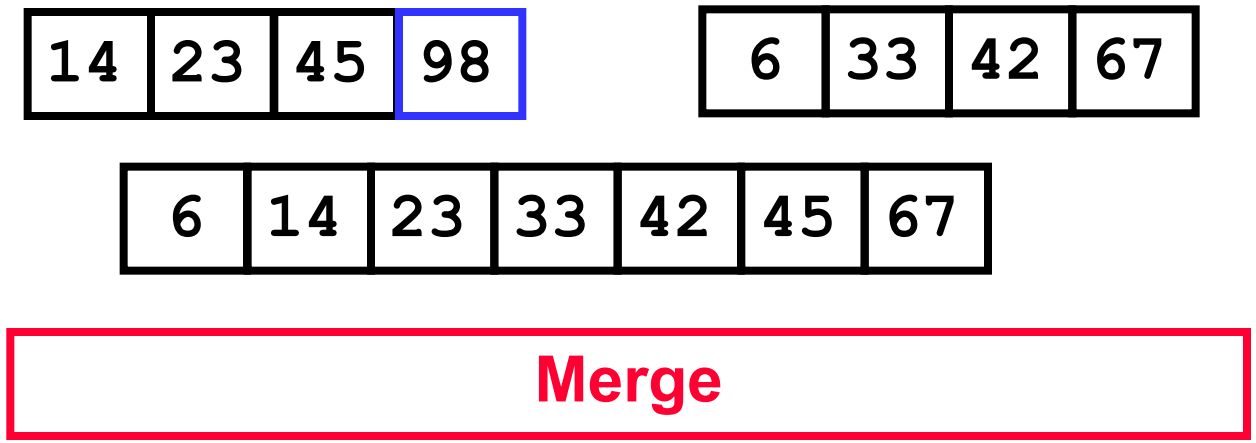
Merging Process



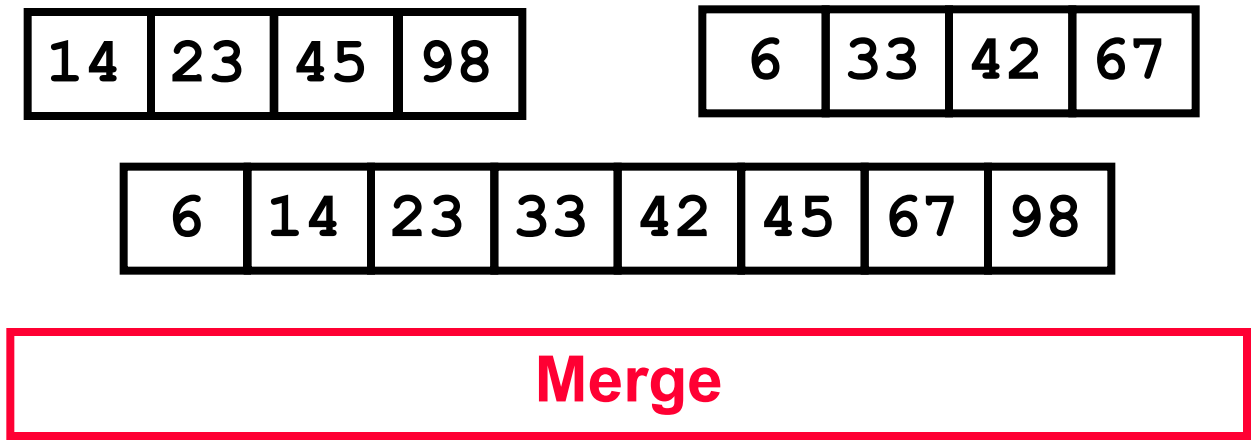
Merging Process



Merging Process



Merging Process



Merge Sort - Discussion

- Running time insensitive of the input
- Advantages:
  - Guaranteed to run in $\Theta(n \lg n)$
- Disadvantage
  - Requires extra space ≈N