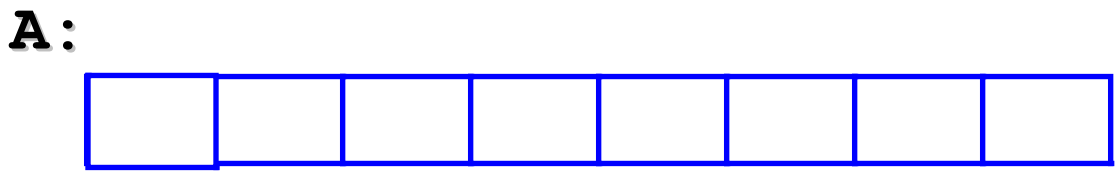
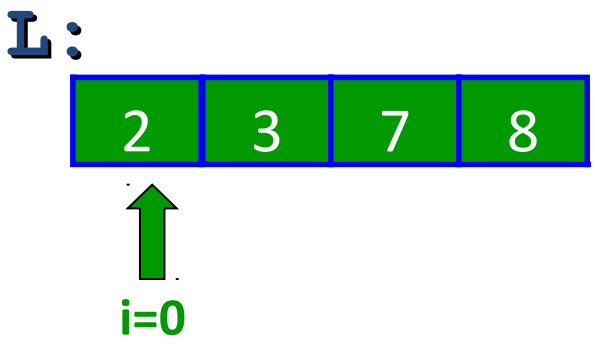
Merge Sort

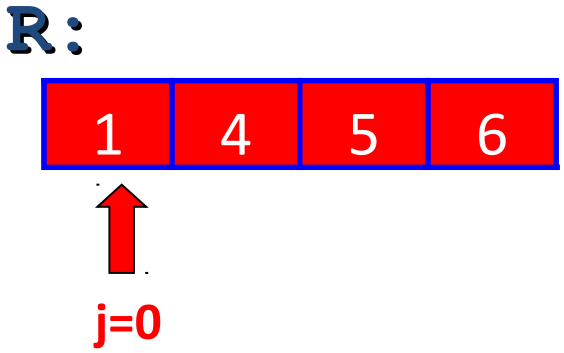
Merge sort Algorithm

- Is based on the divide and conquer approach
- Divides the list into two sublists of sizes as nearly equal as possible
- Sorts the two sublists separately by using merge sort
- Merges the sorted sublists into one single list

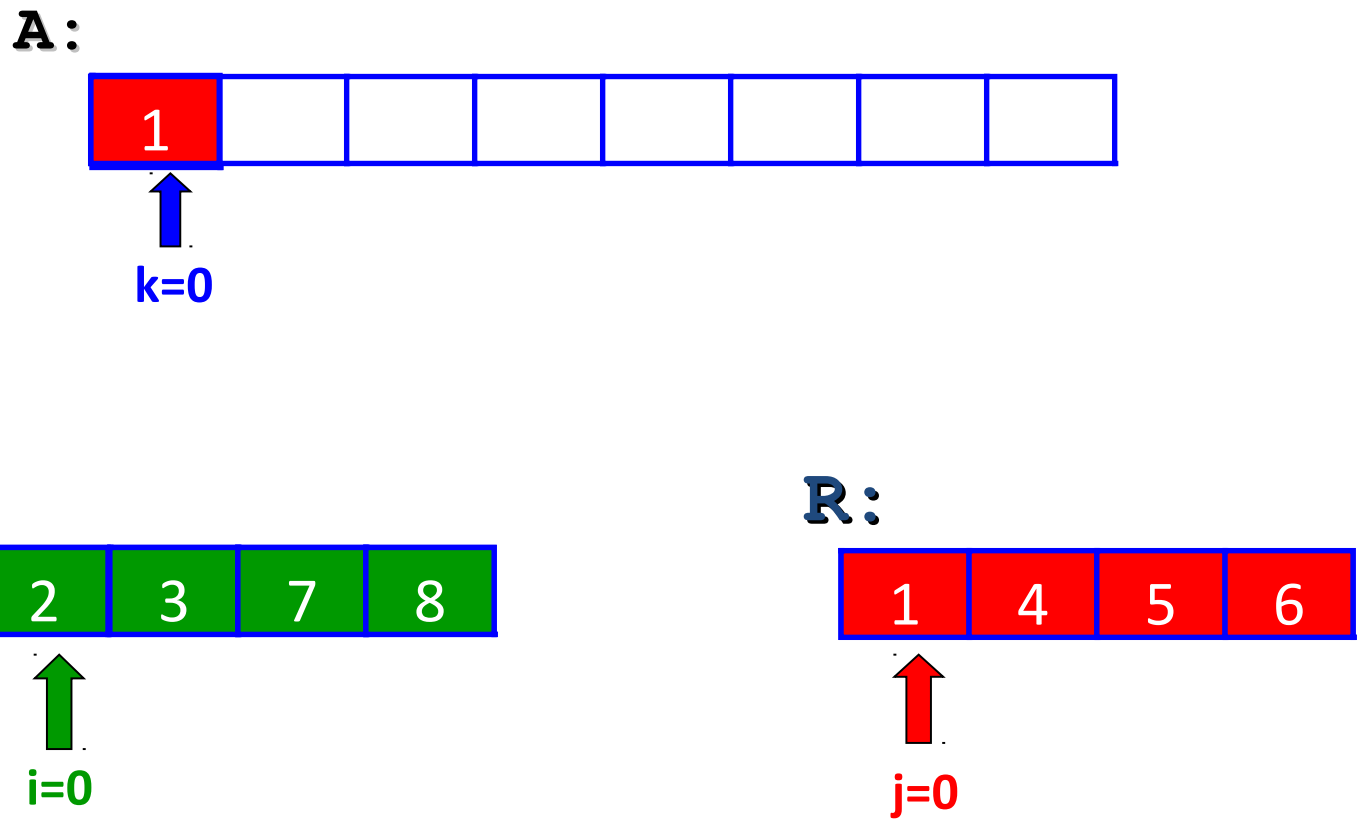
Merge-Sort: Merge Example



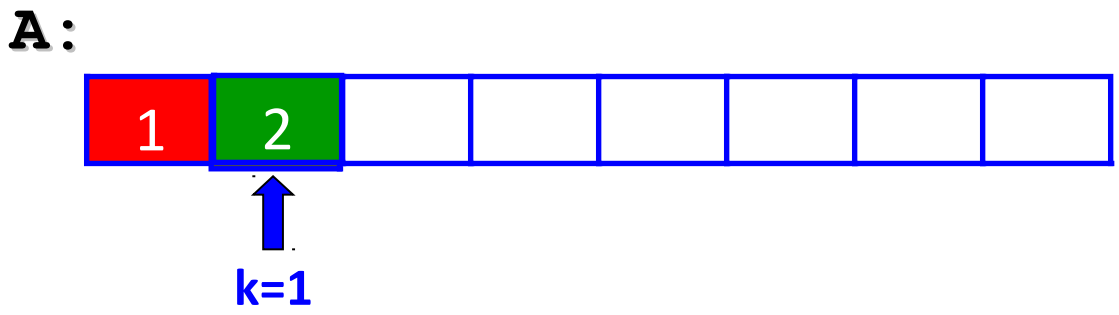


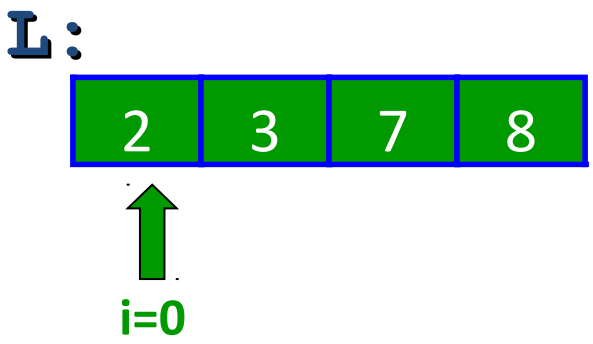


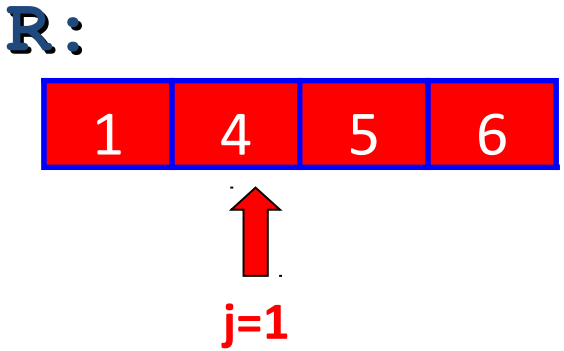
Merge-Sort: Merge Example



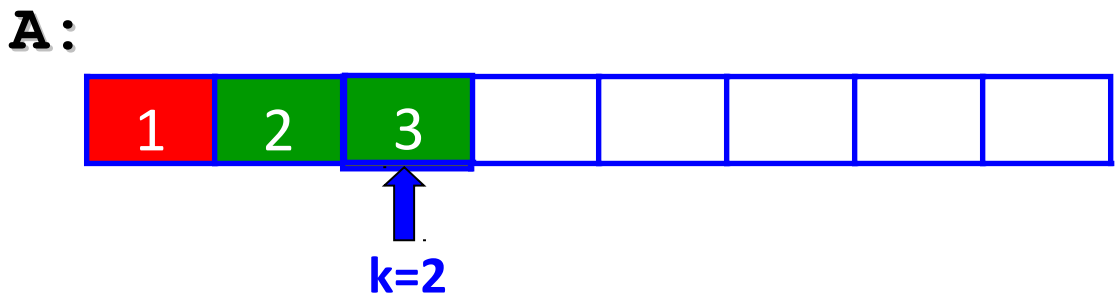
Merge-Sort: Merge Example

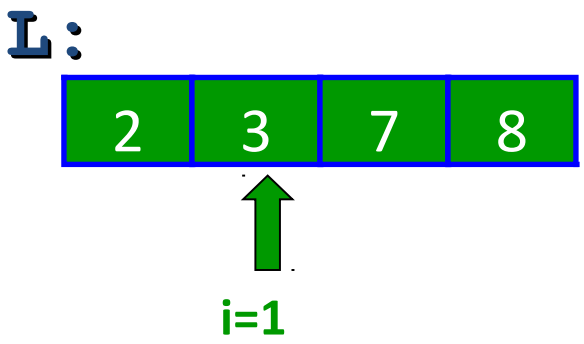


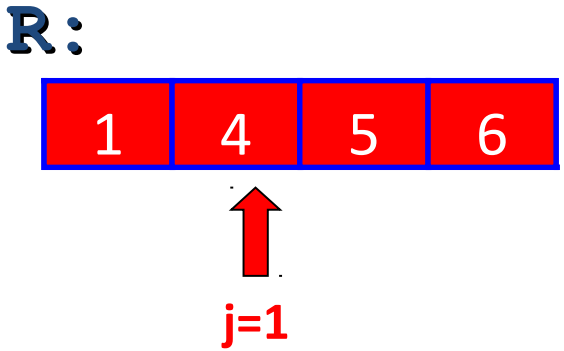




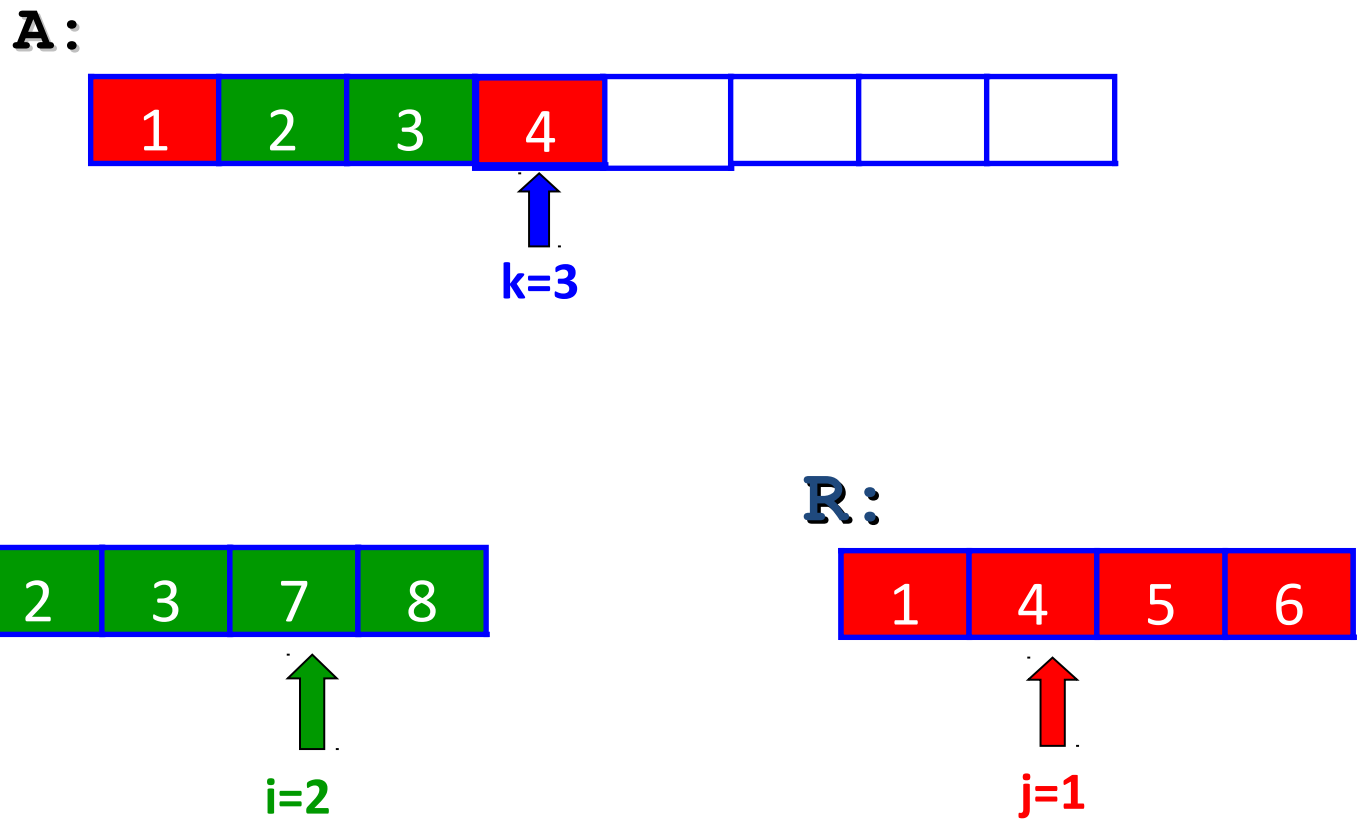
Merge-Sort: Merge Example



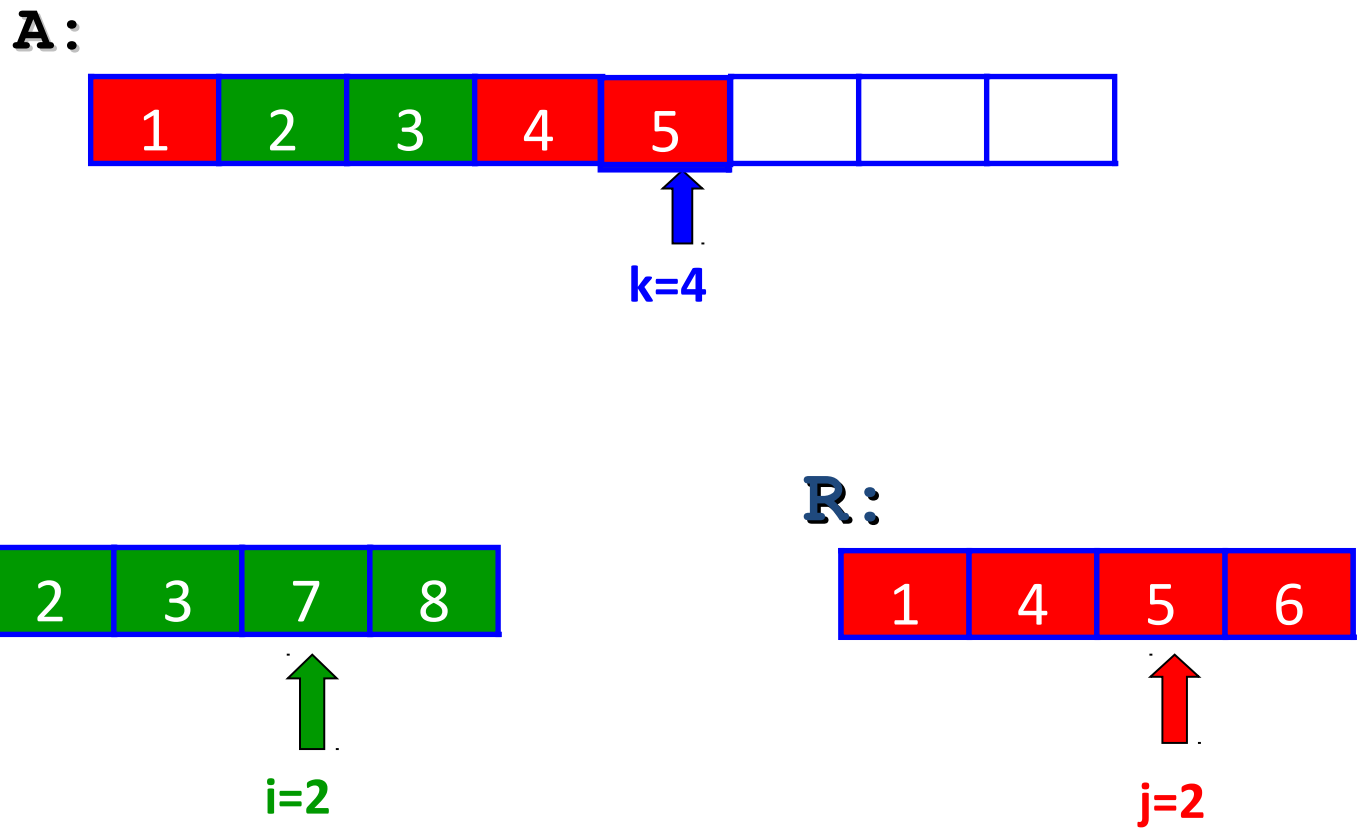




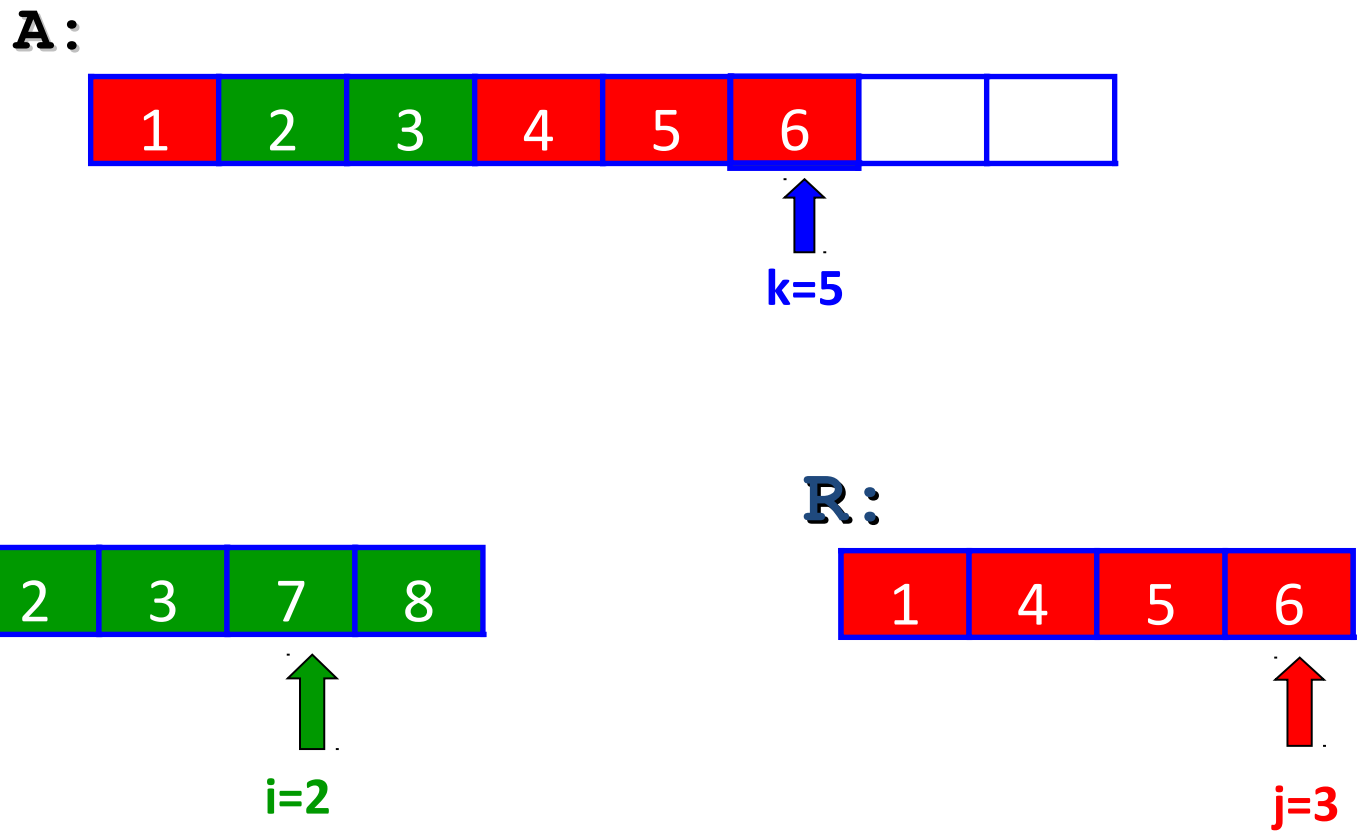
Merge-Sort: Merge Example



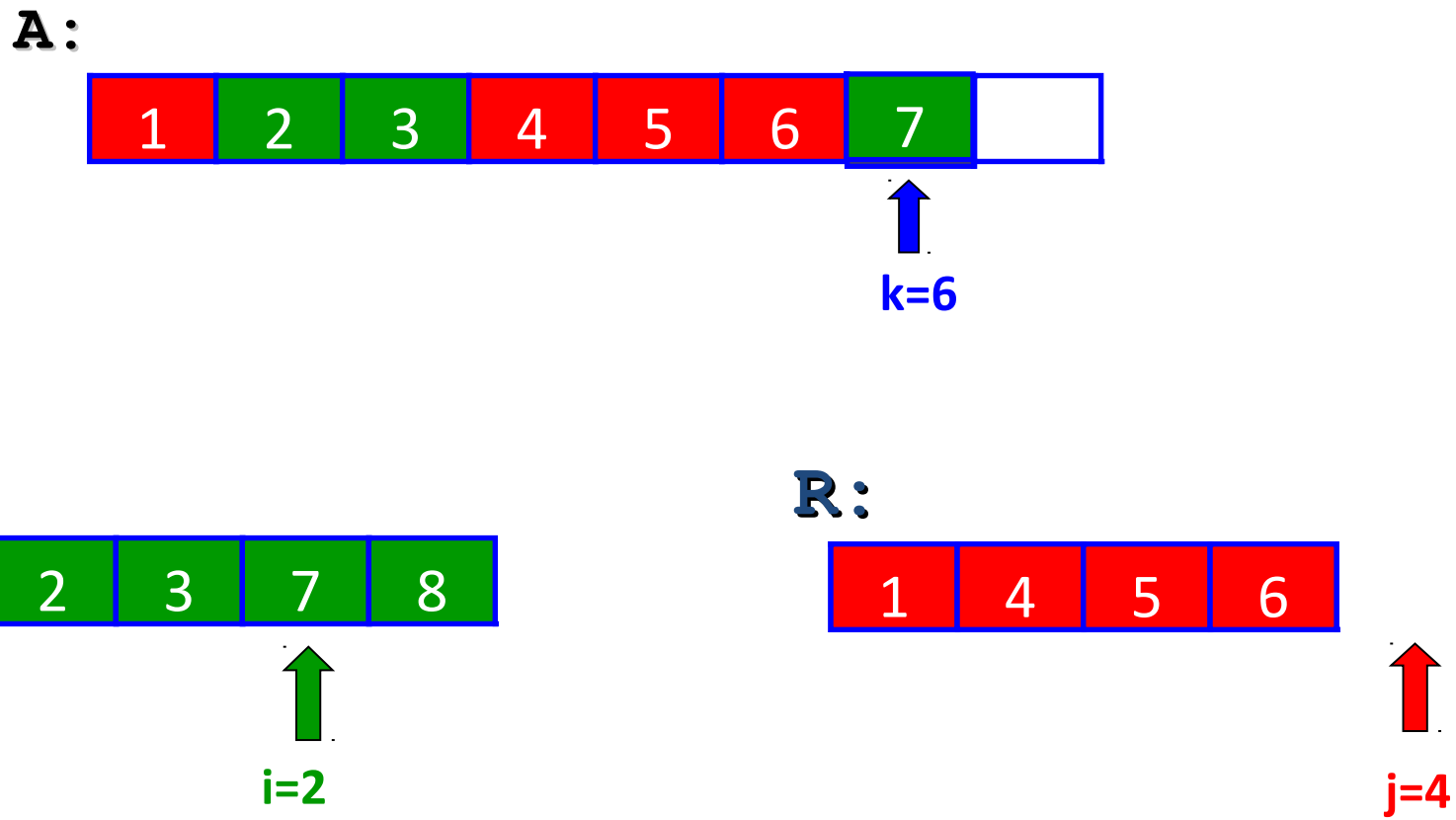
Merge-Sort: Merge Example



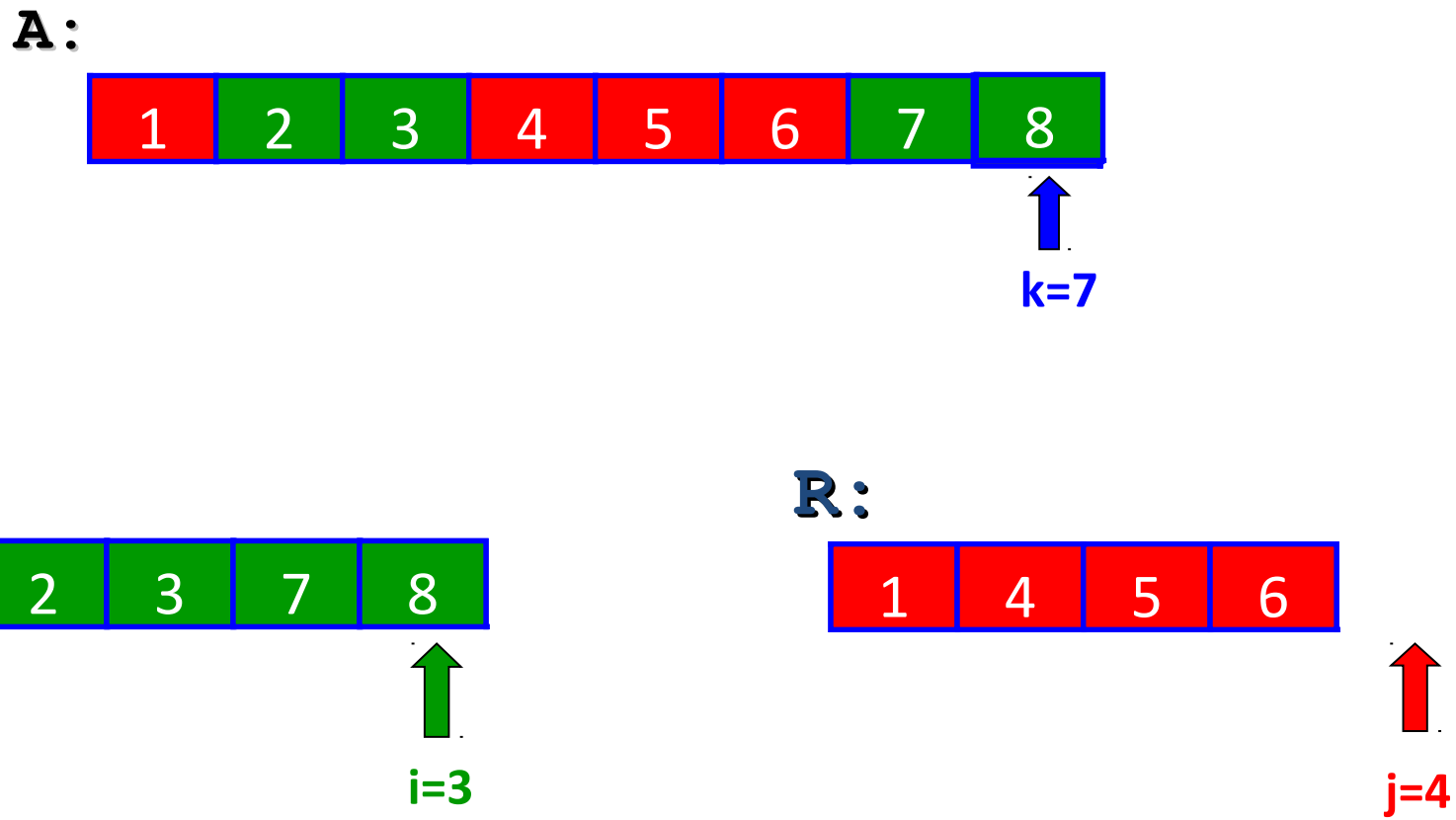
Merge-Sort: Merge Example



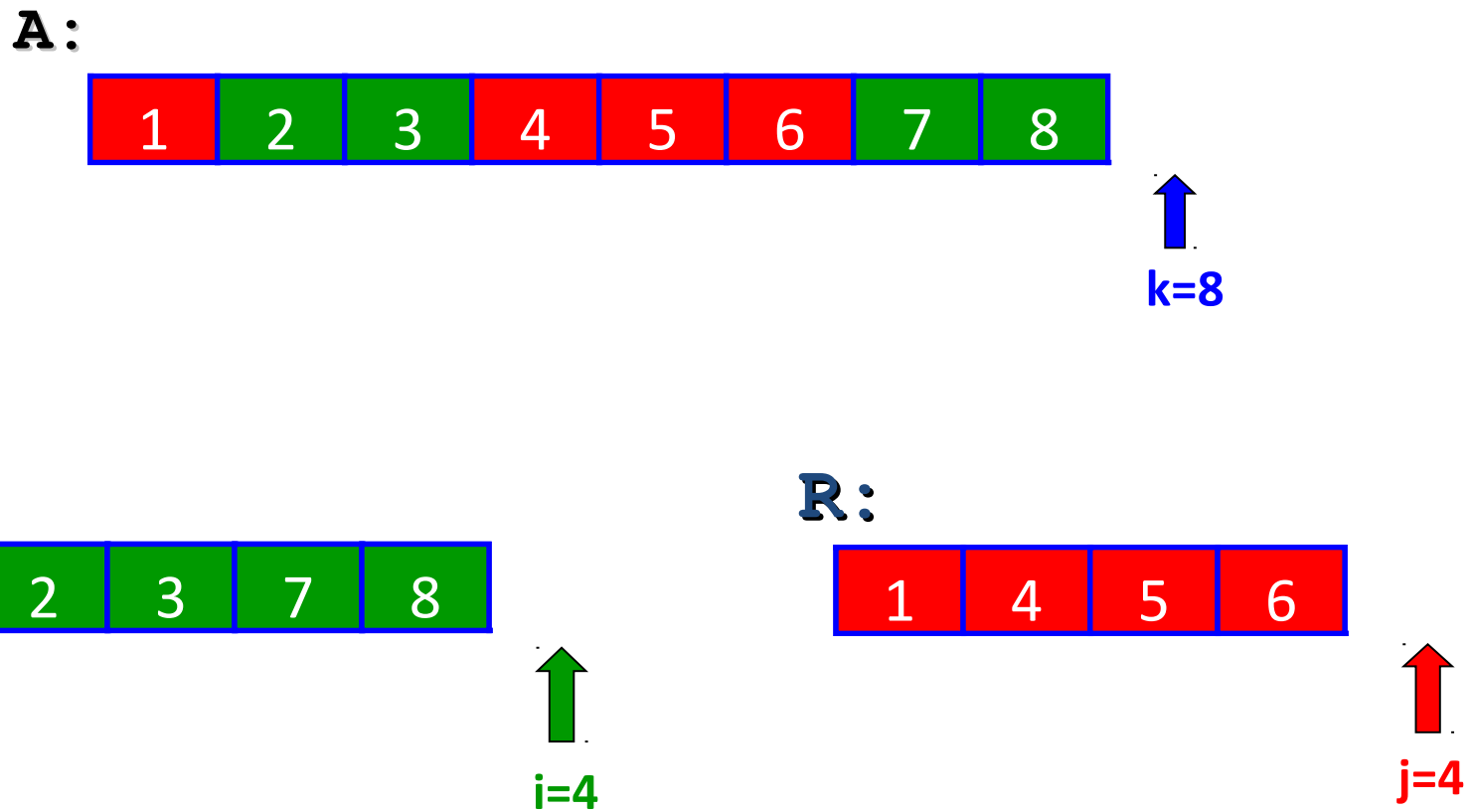
Merge-Sort: Merge Example



Merge-Sort: Merge Example



Merge-Sort: Merge Example



Algorithm:- Merge (A[], n, low, mid, high) //A is an array of n elements, low is lower bound of an array and high is upper bound.

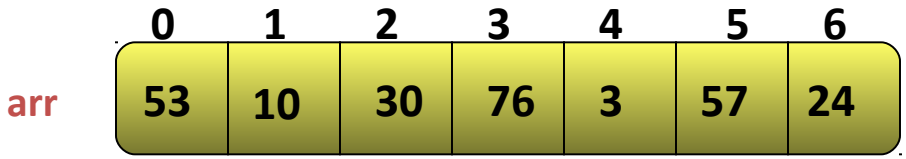
1. i = low 2. j = mid + 1 3. k = low
4. Do
    4.1 If(A [i] <= A [j])
        4.1.1 B [k] = A [i]
        4.1.2 i = i + 1
        4.1.3 k = k +1
    4.2 else
        4.2.1 B [k] = A [j]
        4.2.2 j = j +1
        4.2.3 k = k +1

while (i <= mid && j <= high);

5. While (j < = high)
    5.1 B [k] = A [j]
    5.2 j = j + 1
    5.3 k = k +1
6. While(i <= mid)
    6.1 B [k] = A [i]
    6.2 i = i + 1
    6.3 k = k +1
7. For (i=low to high)
    7.1 A [i] = B [i]

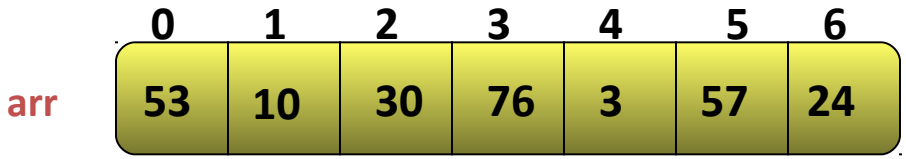
Implementing Merge Sort Algorithm

- To understand the implementation of merge sort algorithm, consider an unsorted list of numbers stored in an array.



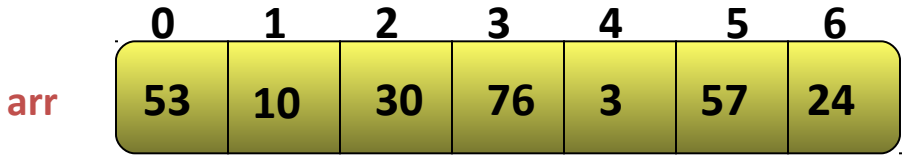
Implementing Merge Sort Algorithm (Contd.)

- Let us sort this unsorted list.



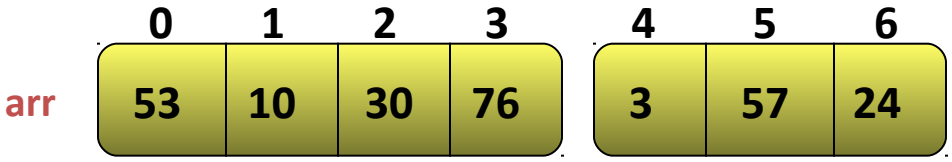
Implementing Merge Sort Algorithm (Contd.)

- The first step to sort data by using merge sort is to split the list into two parts.



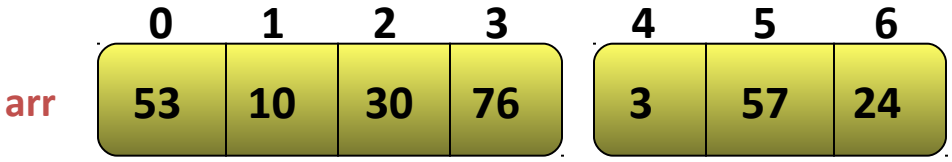
Implementing Merge Sort Algorithm (Contd.)

- The first step to sort data by using merge sort is to split the list into two parts.



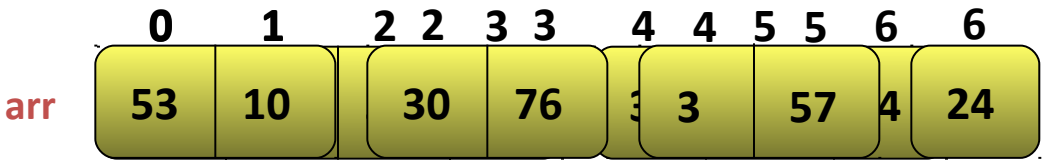
Implementing Merge Sort Algorithm (Contd.)

- The list has odd number of elements, therefore, the left sublist is longer than the right sublist by one entry.



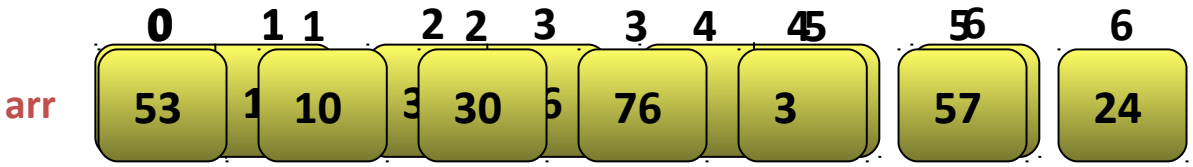
Implementing Merge Sort Algorithm (Contd.)

- Further divide the two sublists into nearly equal parts.



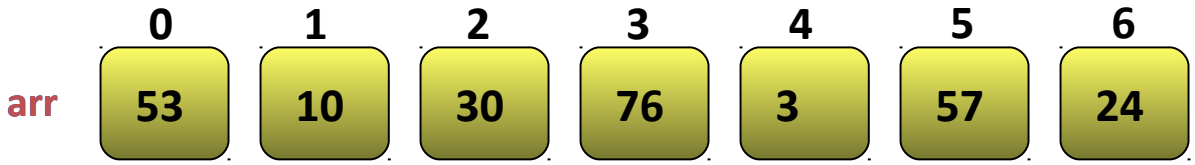
Implementing Merge Sort Algorithm (Contd.)

- Further divide the sublists.



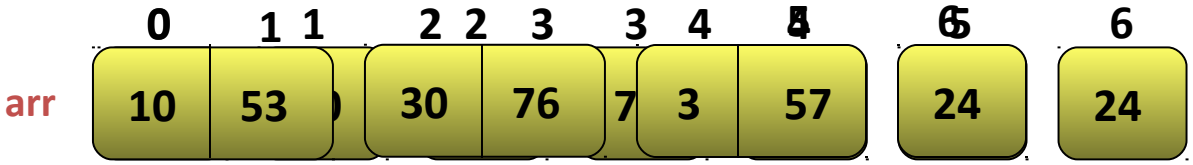
Implementing Merge Sort Algorithm (Contd.)

- There is a single element left in each sublist.
- Sublists with one element require no sorting.



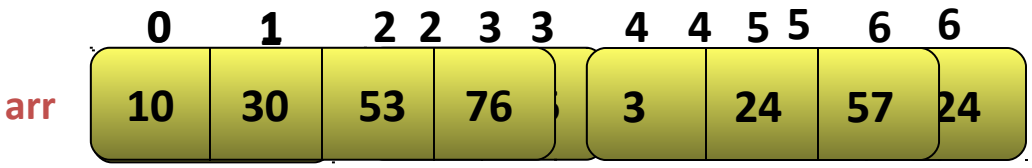
Implementing Merge Sort Algorithm (Contd.)

- Start merging the sublists to obtain a sorted list.



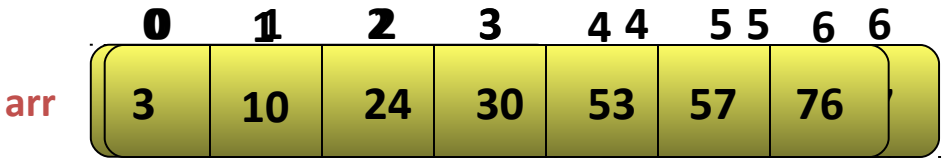
Implementing Merge Sort Algorithm (Contd.)

- Further merge the sublists.



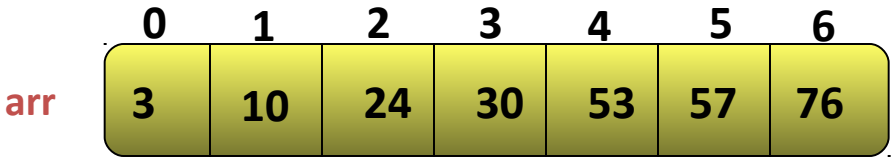
Implementing Merge Sort Algorithm (Contd.)

- Again, merge the sublists.



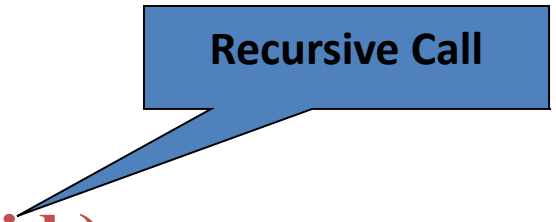
Implementing Merge Sort Algorithm (Contd.)

- The list is now sorted.



Write an algorithm to implement Merge sort:

```
Merge_sort ( A[n], low, high )
   {
    //  A is an array of n elements,low is lower bound
       of an array and high is upper bound.
    1. if (low = = high)
         1.1 Return
    2. else
         2.1 mid = (low + high)/2
         2.2 Merge_sort ( A[], low , mid )
         2.3 Merge_sort (A[], mid+1 , high )
         2.4 Merge ( A[], low, mid, high )
```

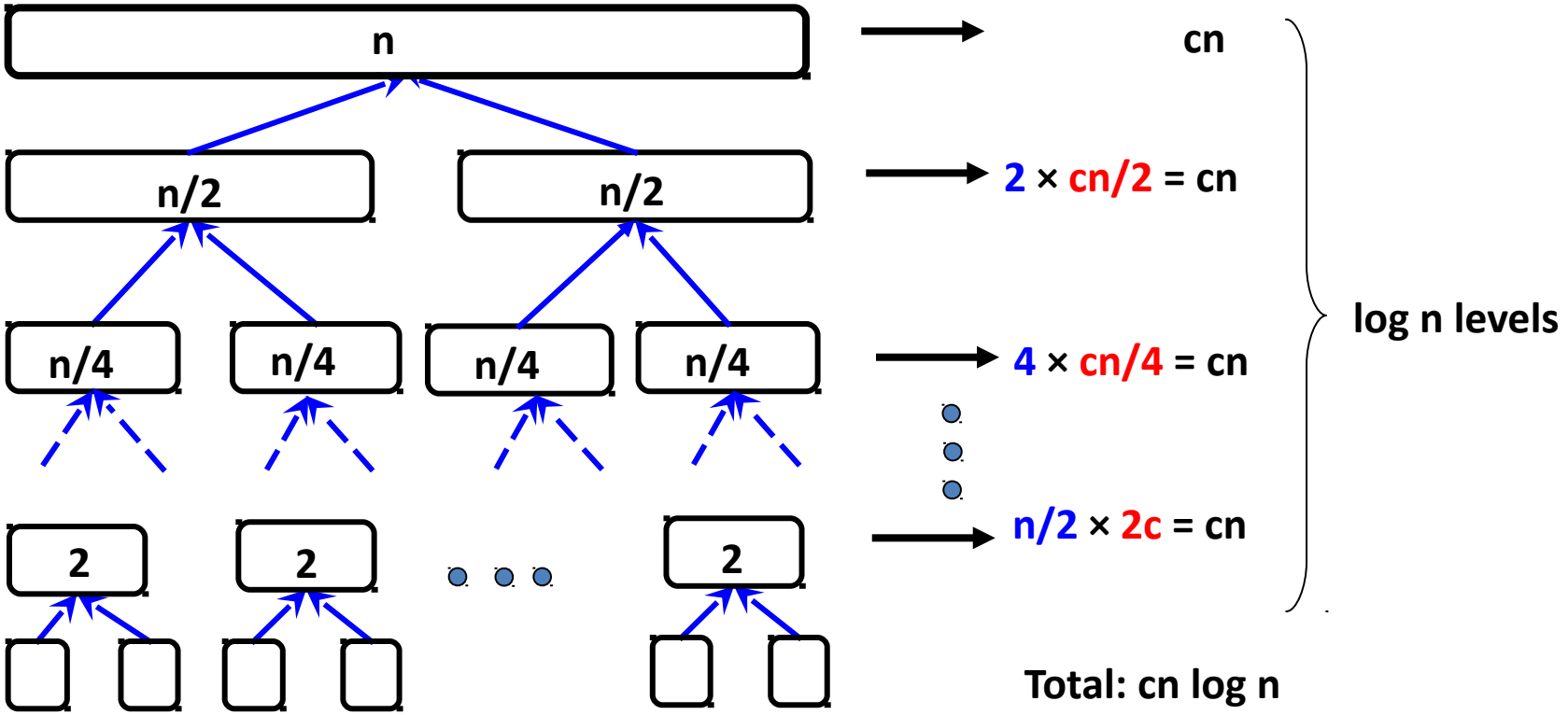


Analysis of Merge Sort

- Divide the List into two sub-list
- Sort each sub-list
- Merge the two sub-list

The efficiency of merge sort is equal to O(n log n)

Merge-Sort Analysis



- Total running time: O (n logn)
-