Merge Sort

SORTING ALGORITHM

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

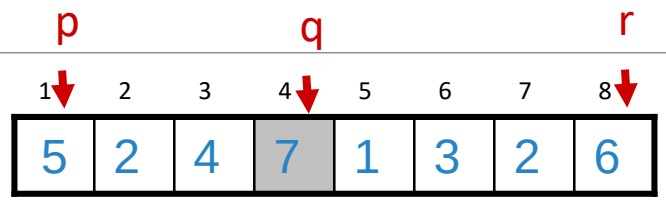
Conquer

- 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



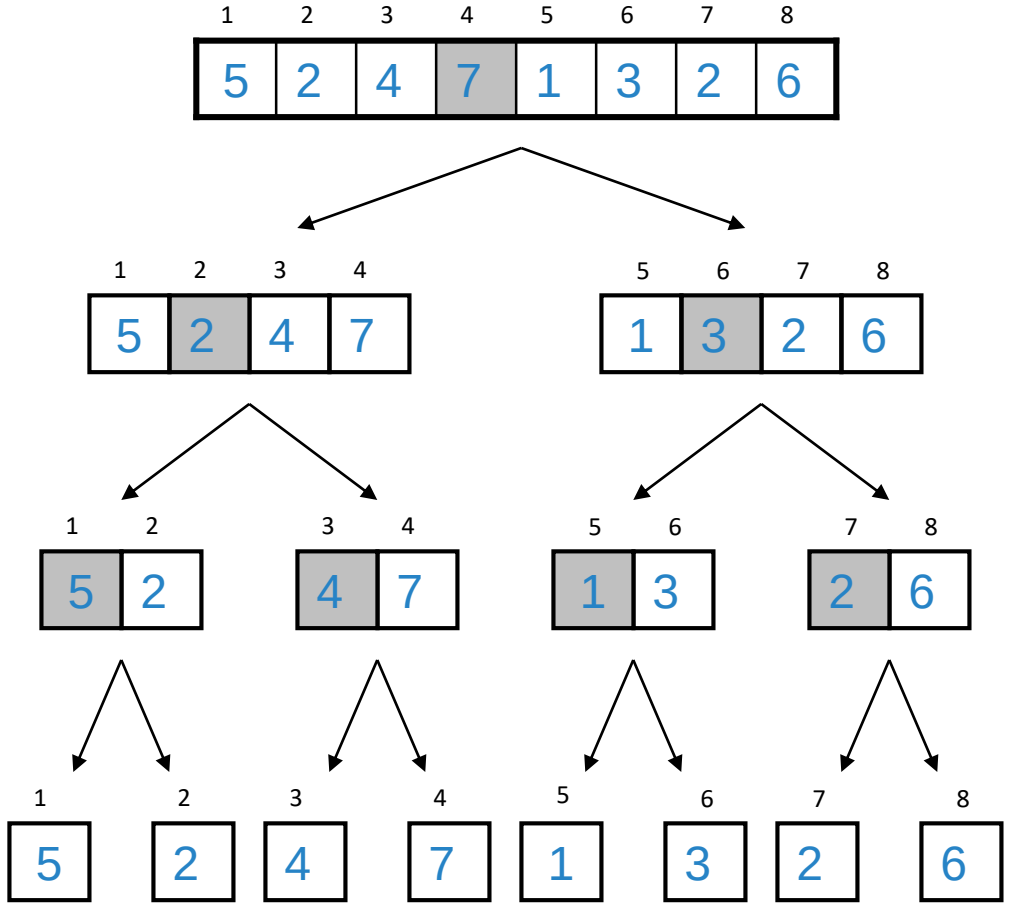
Alg.: MERGE-SORT(A, p, r)

```
if p < r                                  ▷ Check for base case
  then q ← ⌊(p + r)/2⌋                    ▷ 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)

Example – **n** Power of 2

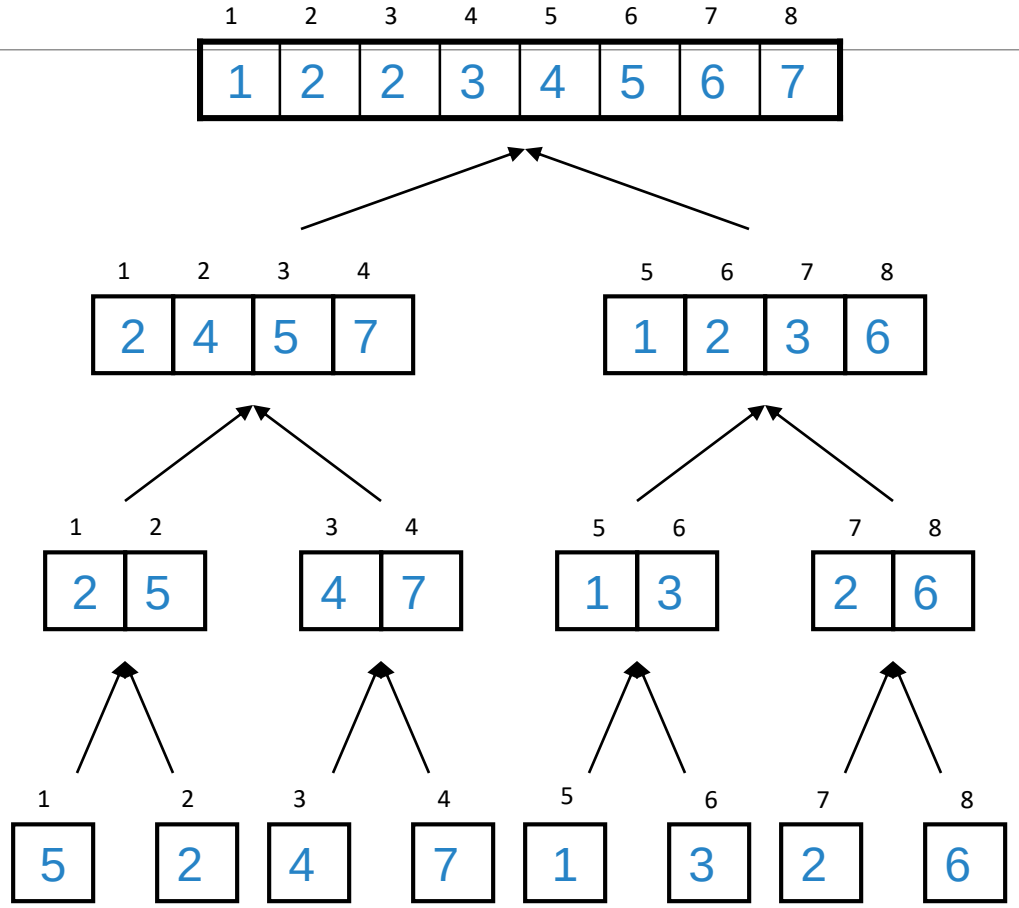
Divide



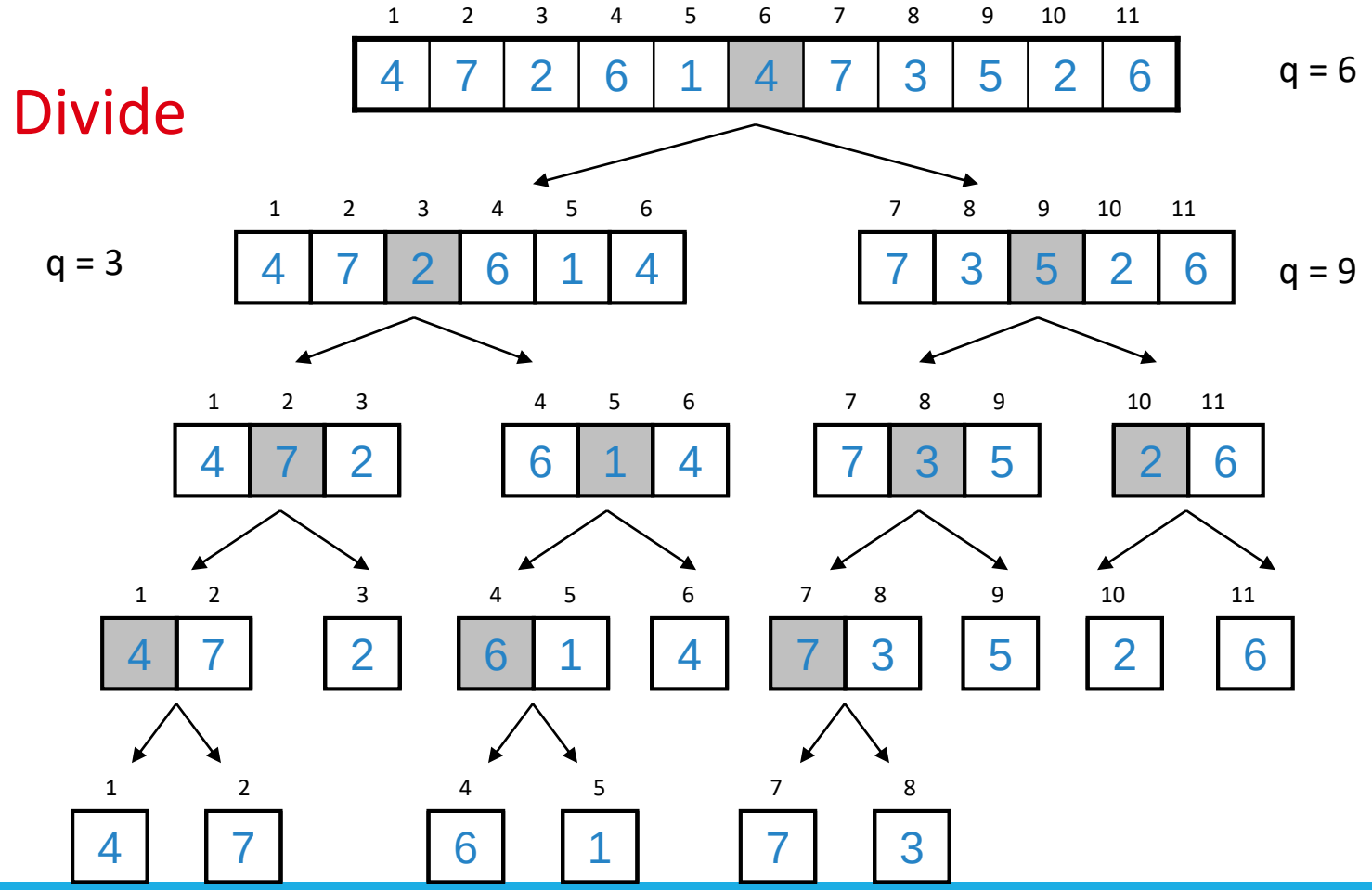
q = 4

Example – **n** Power of 2

Conquer and Merge

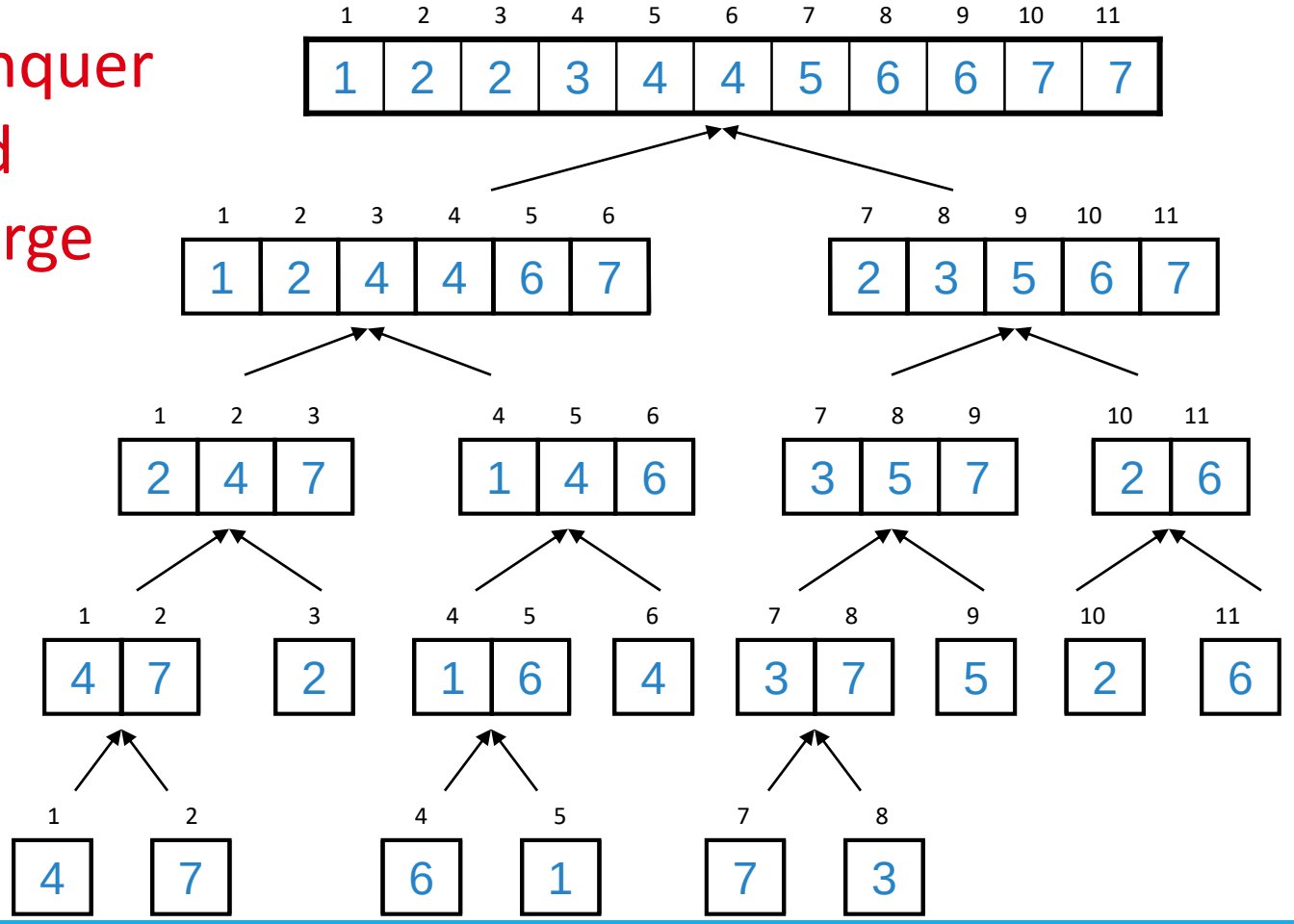


Example – **n** Not a Power of 2

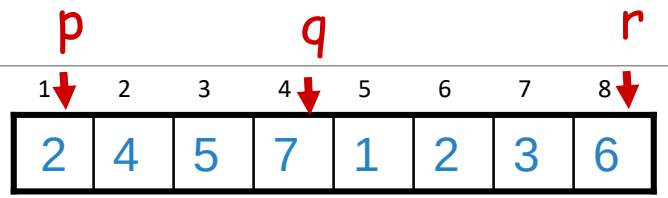


Example – *n* Not a Power of 2

Conquer and Merge



Merging

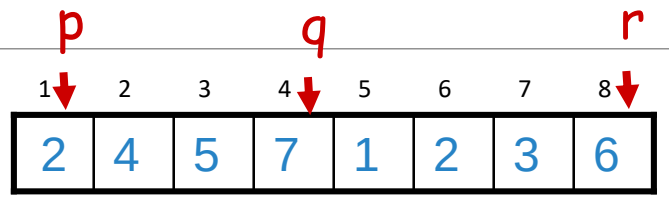


Input: Array A and indices p, q, r such that $p \le q < r$

- Subarrays $A[p\,..\,q]$ and $A[q+1\,..\,r]$ are sorted

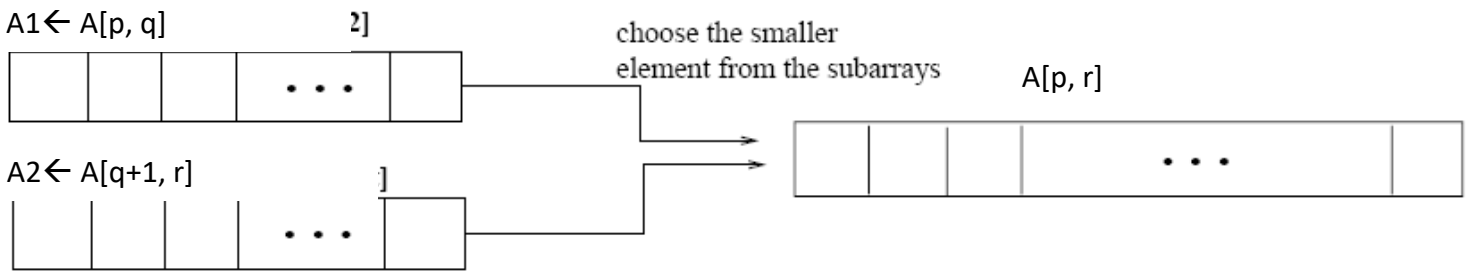
Output: One single sorted subarray $A[p\,..\,r]$

Merging



Idea for merging:

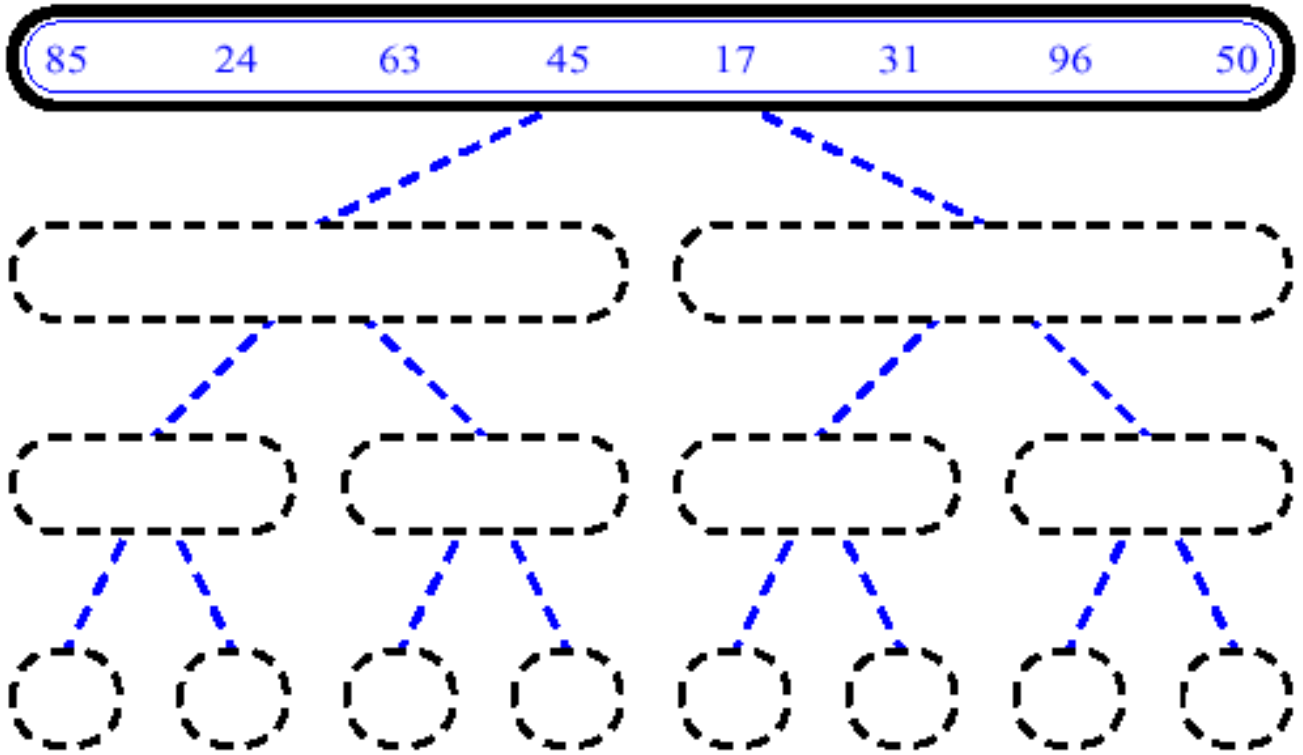
- Two piles of sorted cards
  - Choose the smaller of the two top cards
  - Remove it and place it in the output pile
- Repeat the process until one pile is empty
- Take the remaining input pile and place it face-down onto the output pile



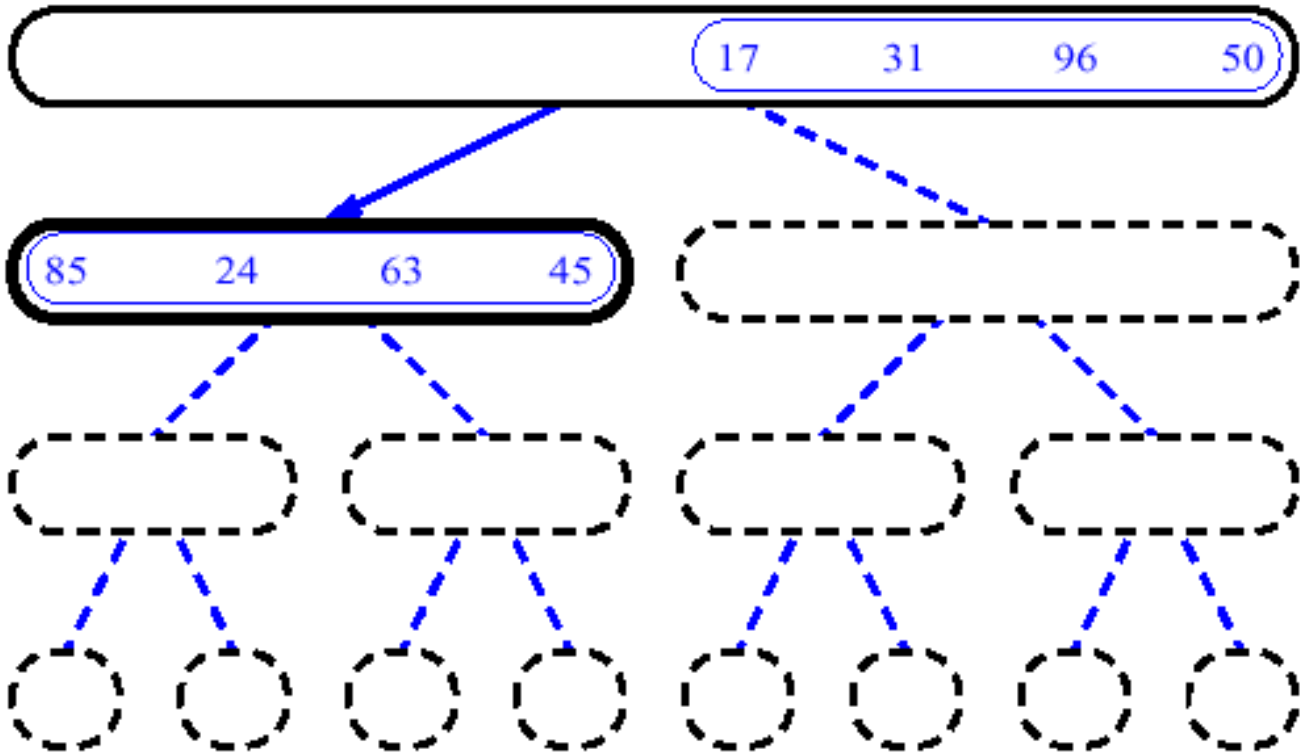
Merge Sort

EXAMPLE

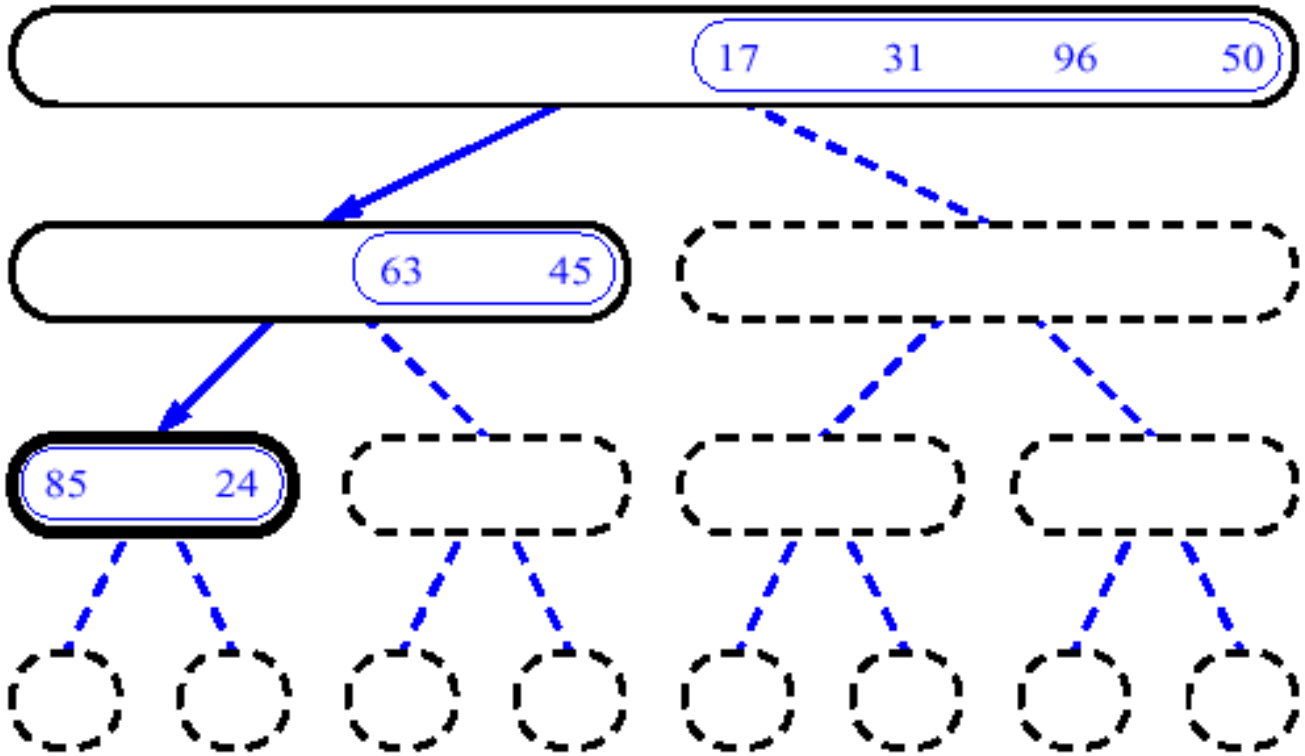
MergeSort (Example) - 1



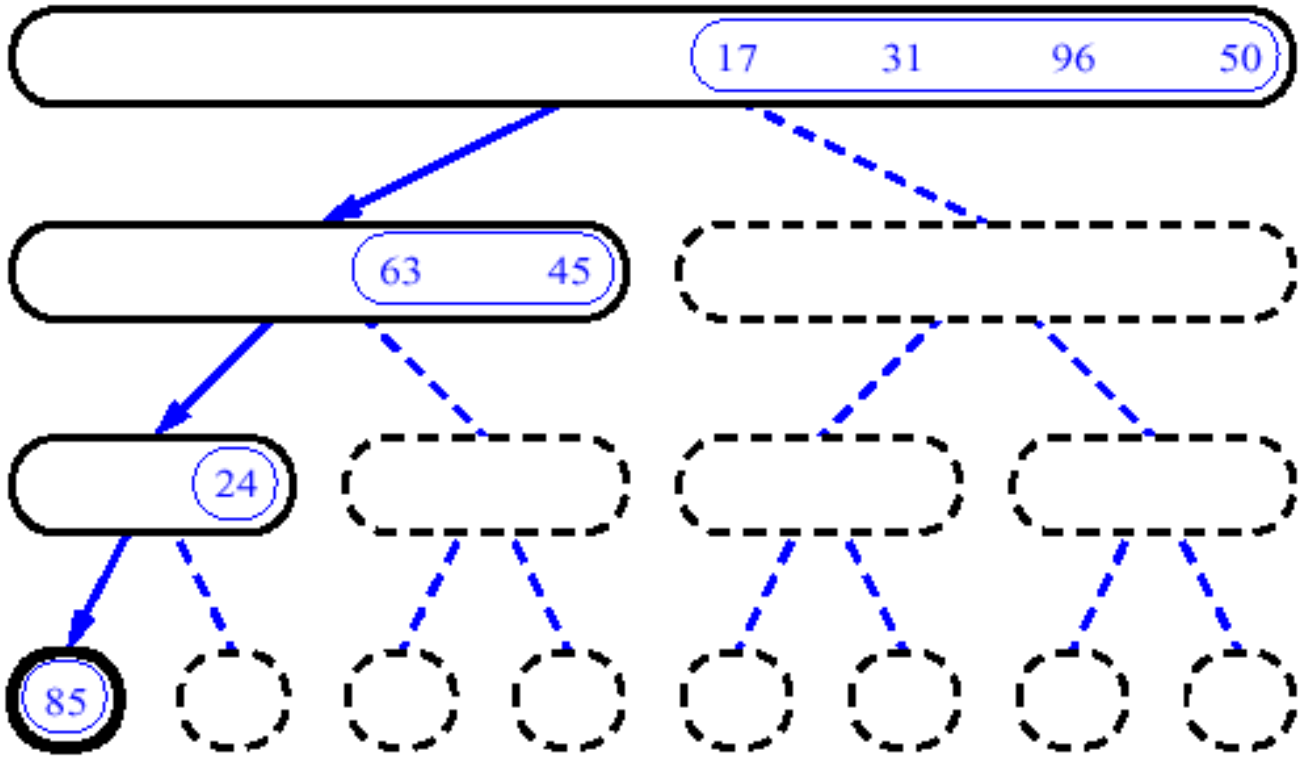
MergeSort (Example) - 2



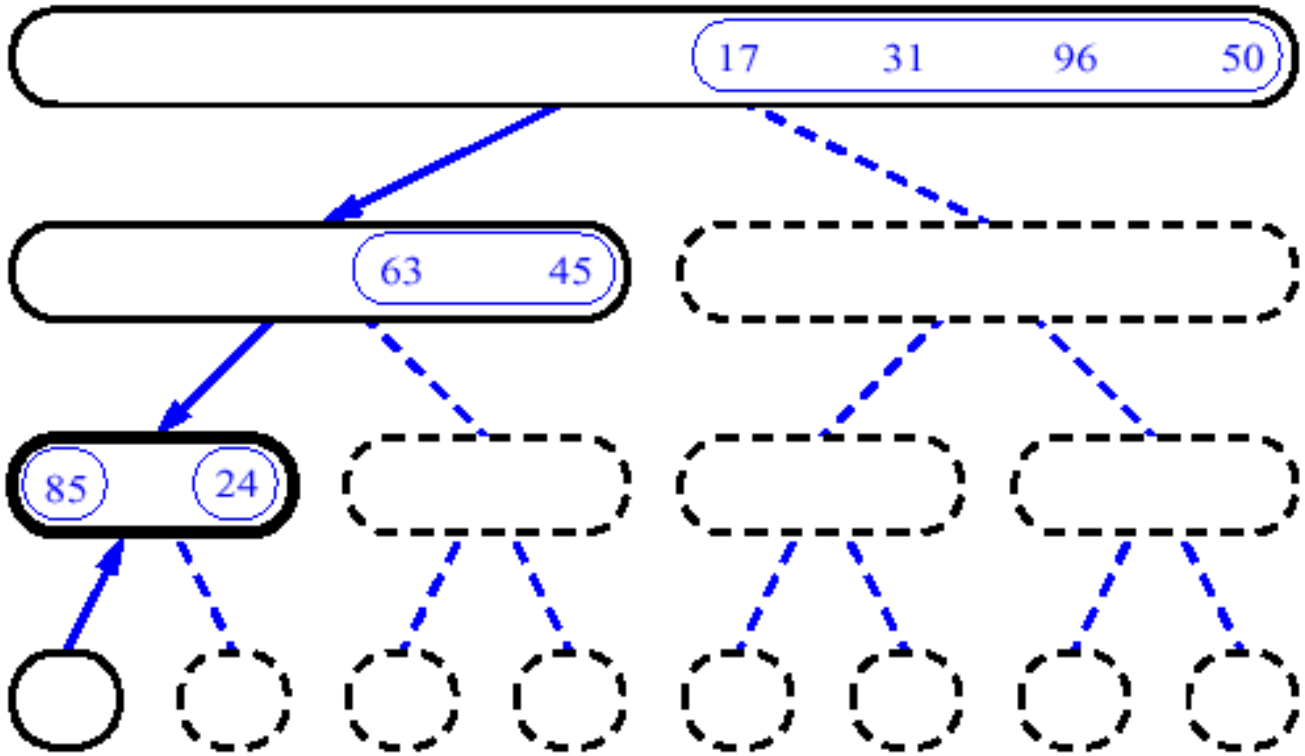
MergeSort (Example) - 3



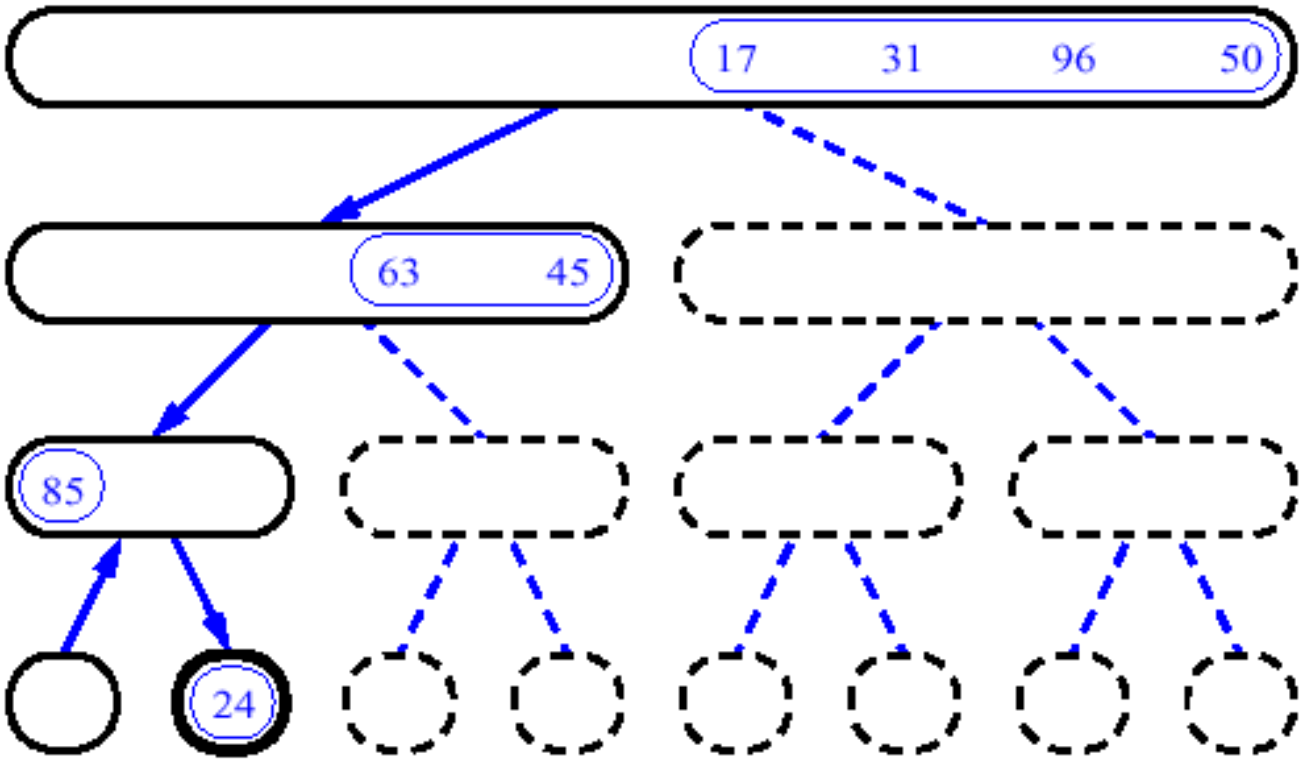
MergeSort (Example) - 4



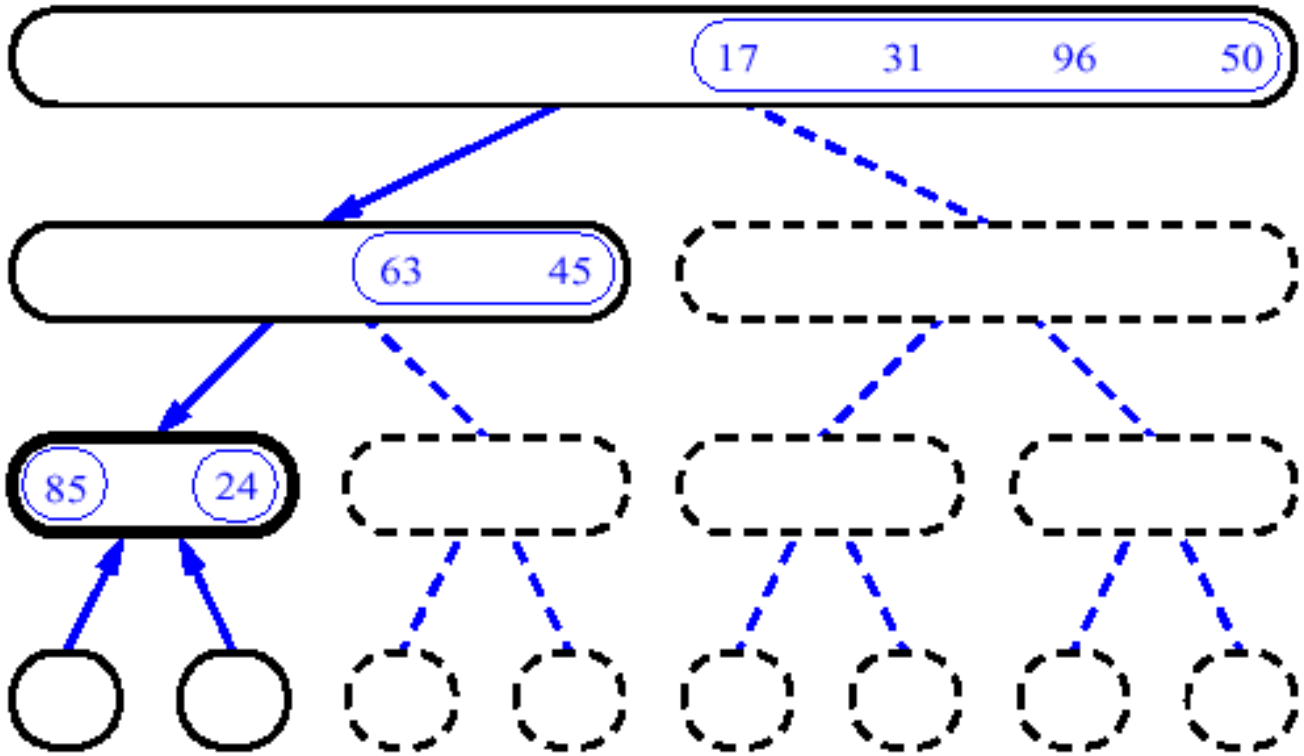
MergeSort (Example) - 5



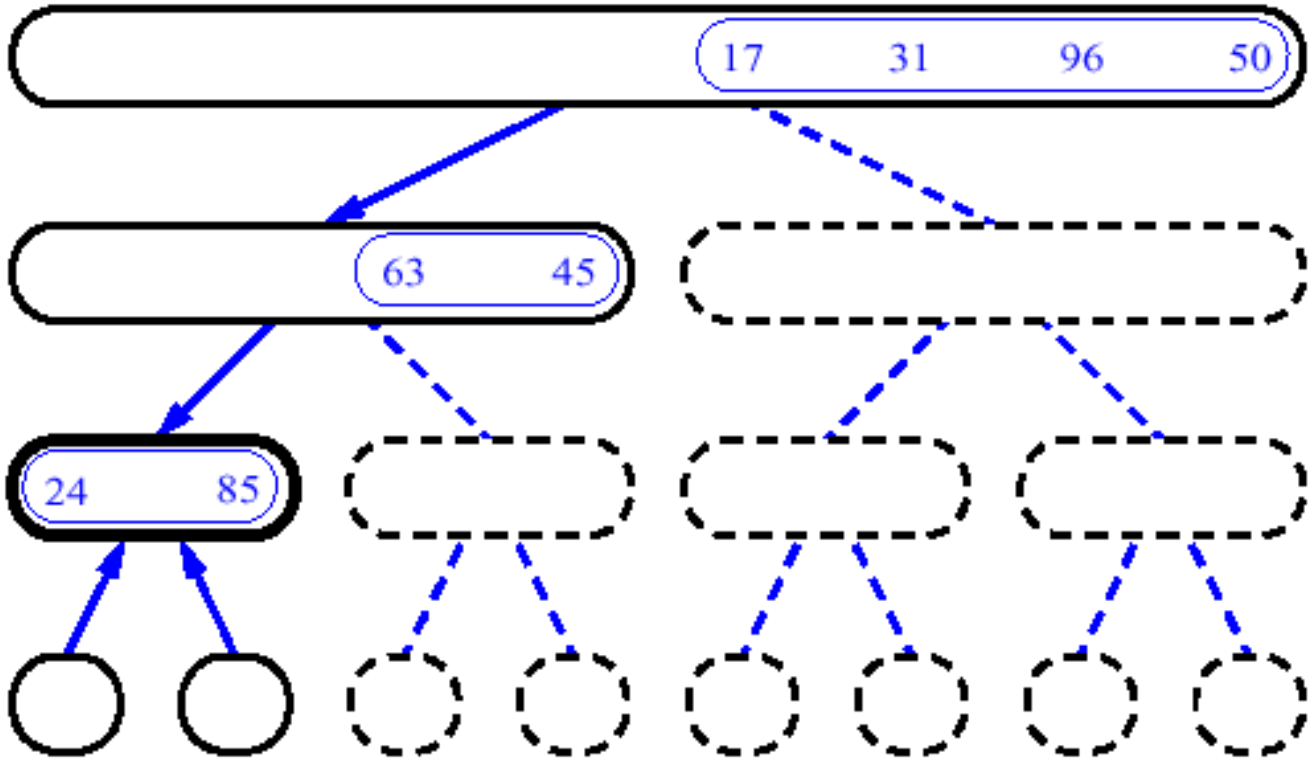
MergeSort (Example) - 6



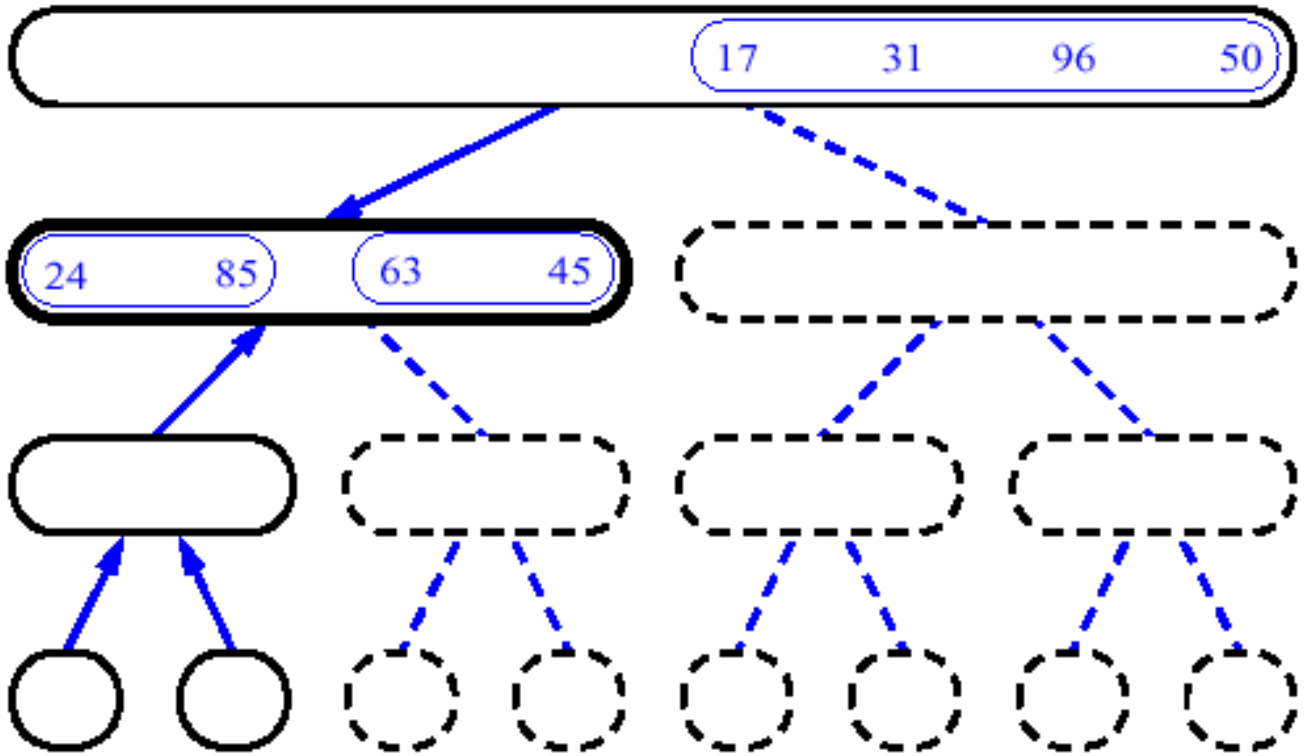
MergeSort (Example) - 7



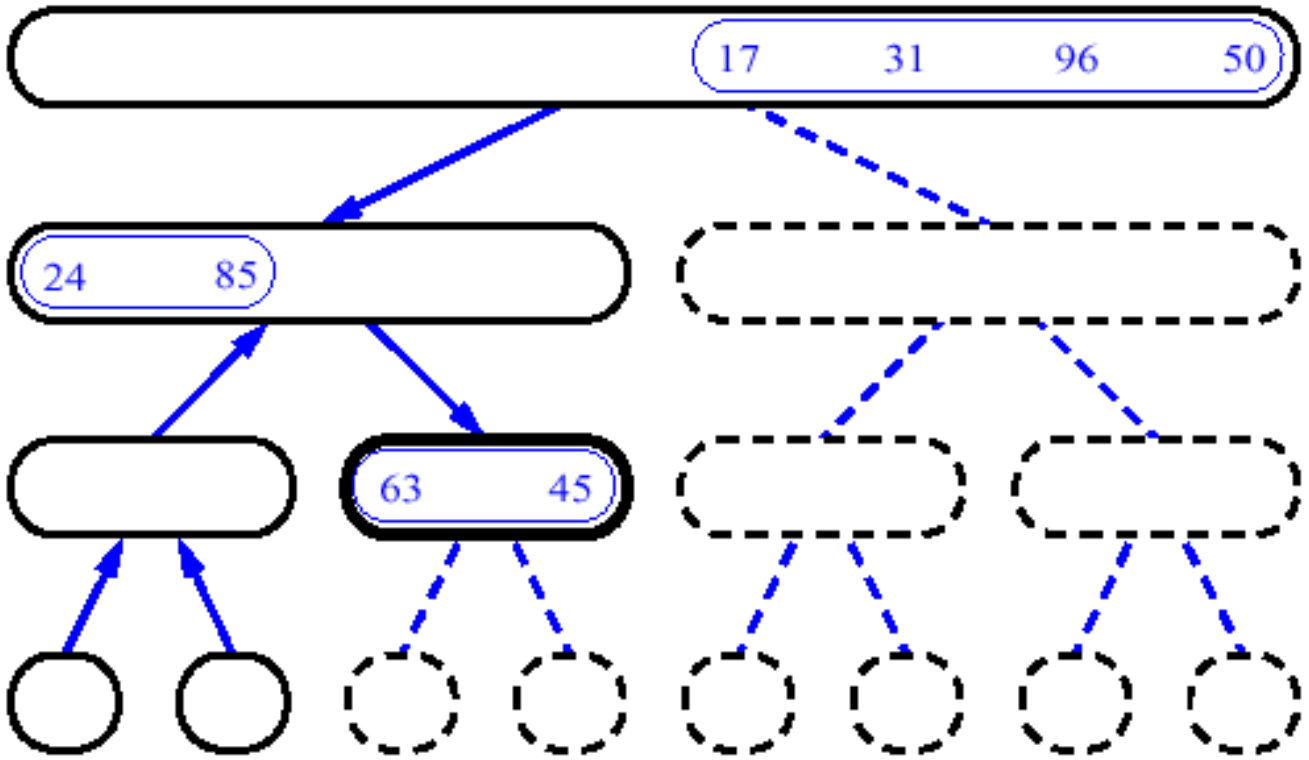
MergeSort (Example) - 8



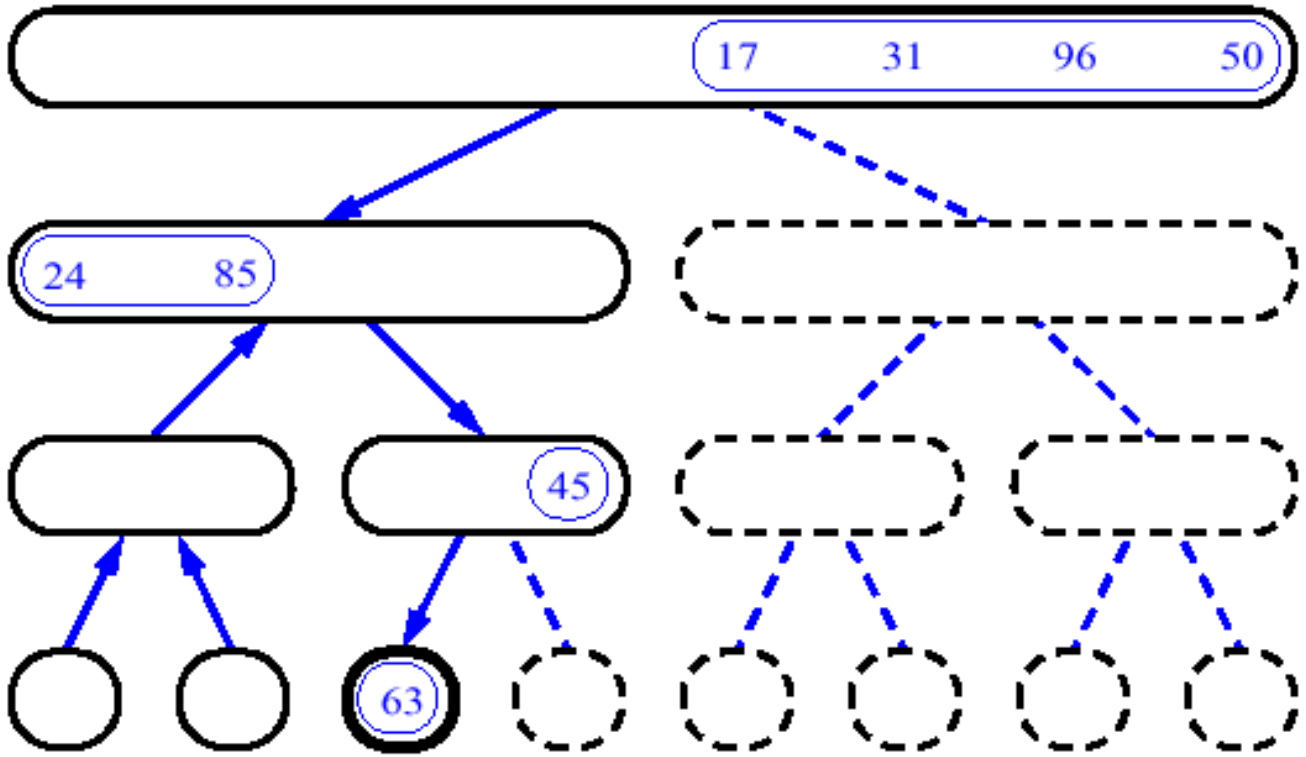
MergeSort (Example) - 9



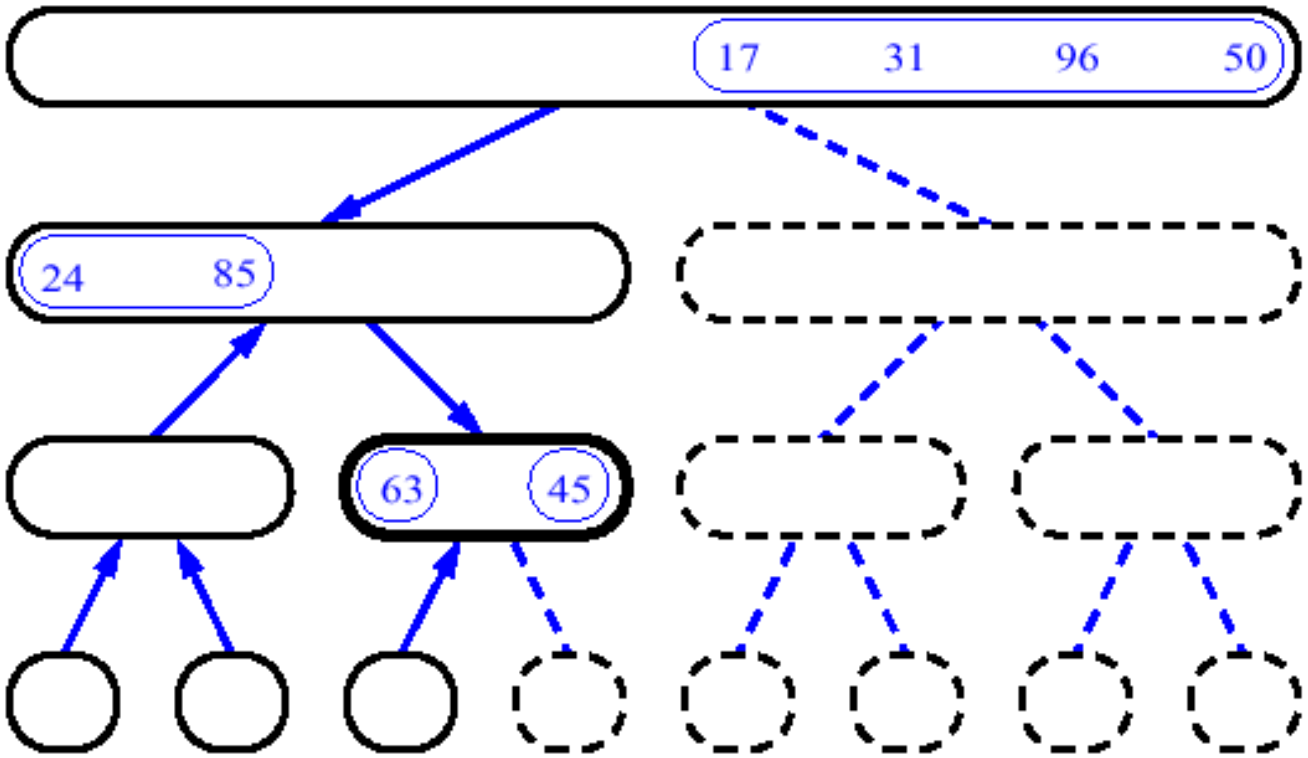
MergeSort (Example) - 10



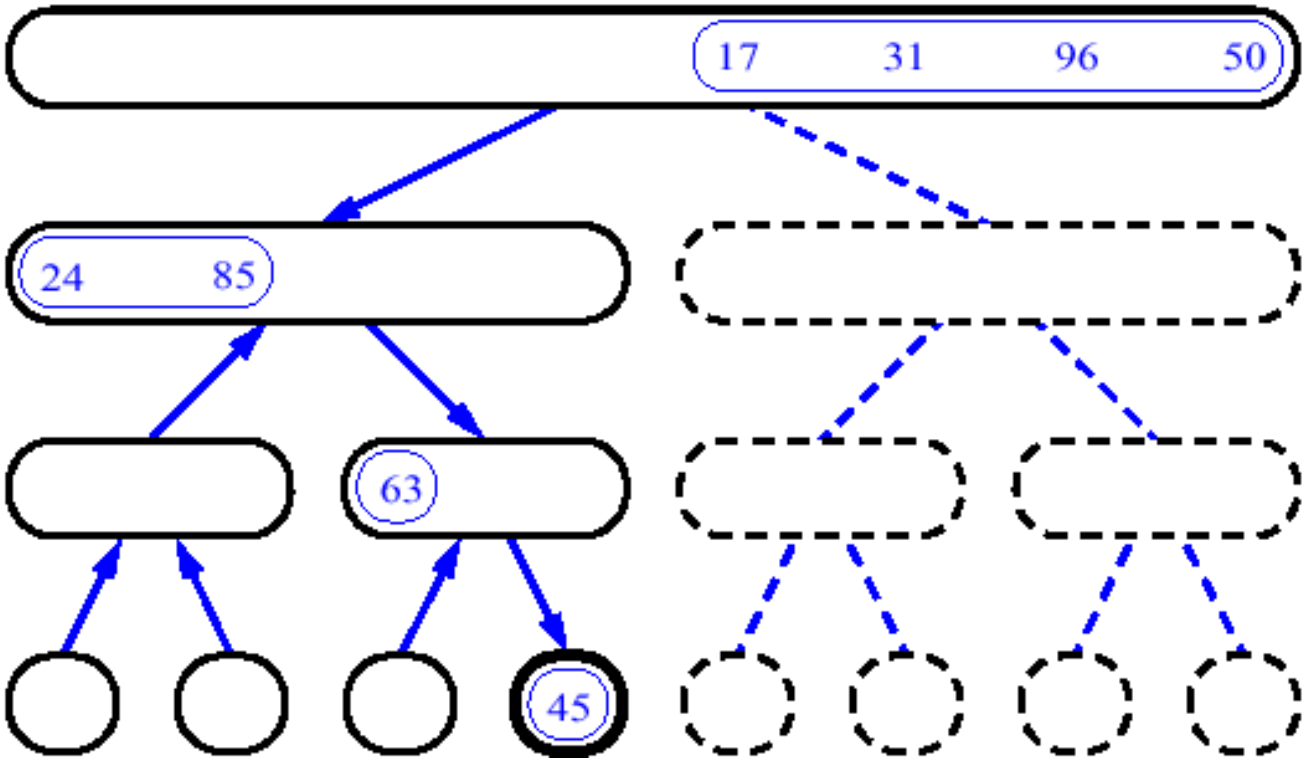
MergeSort (Example) - 11



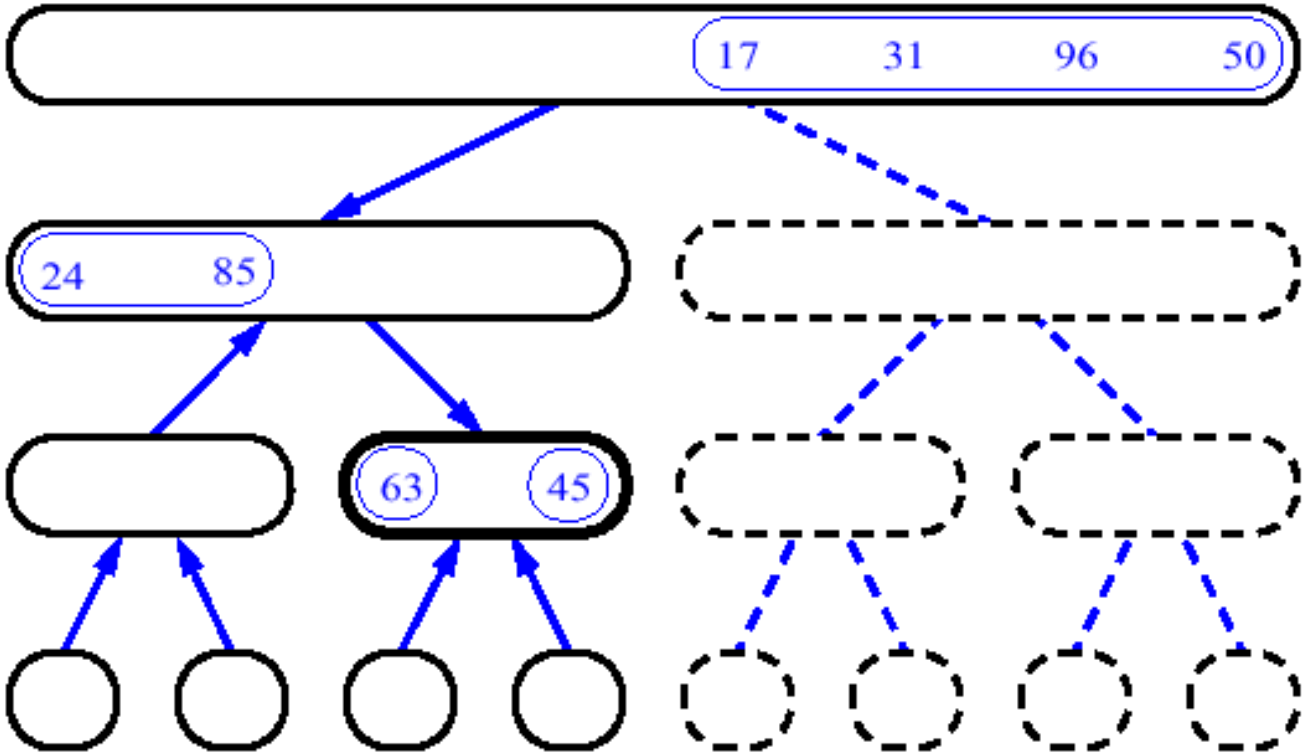
MergeSort (Example) - 12



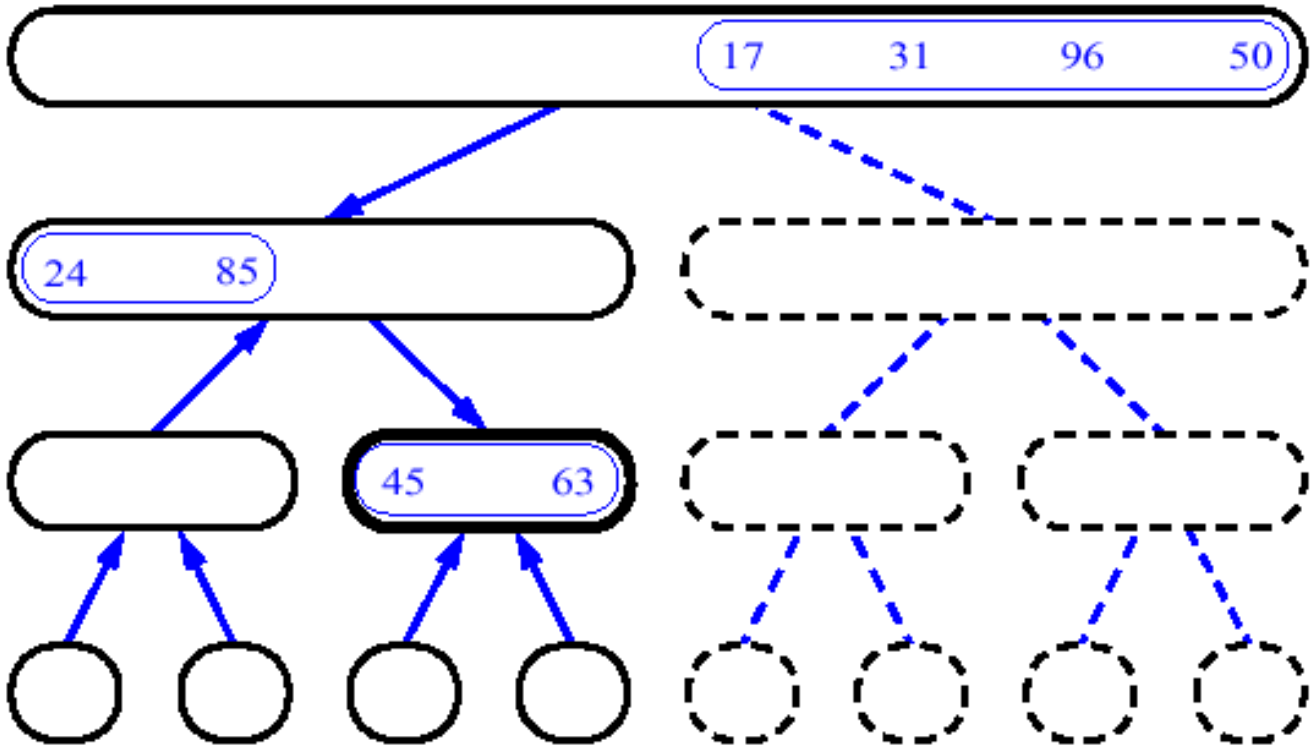
MergeSort (Example) - 13



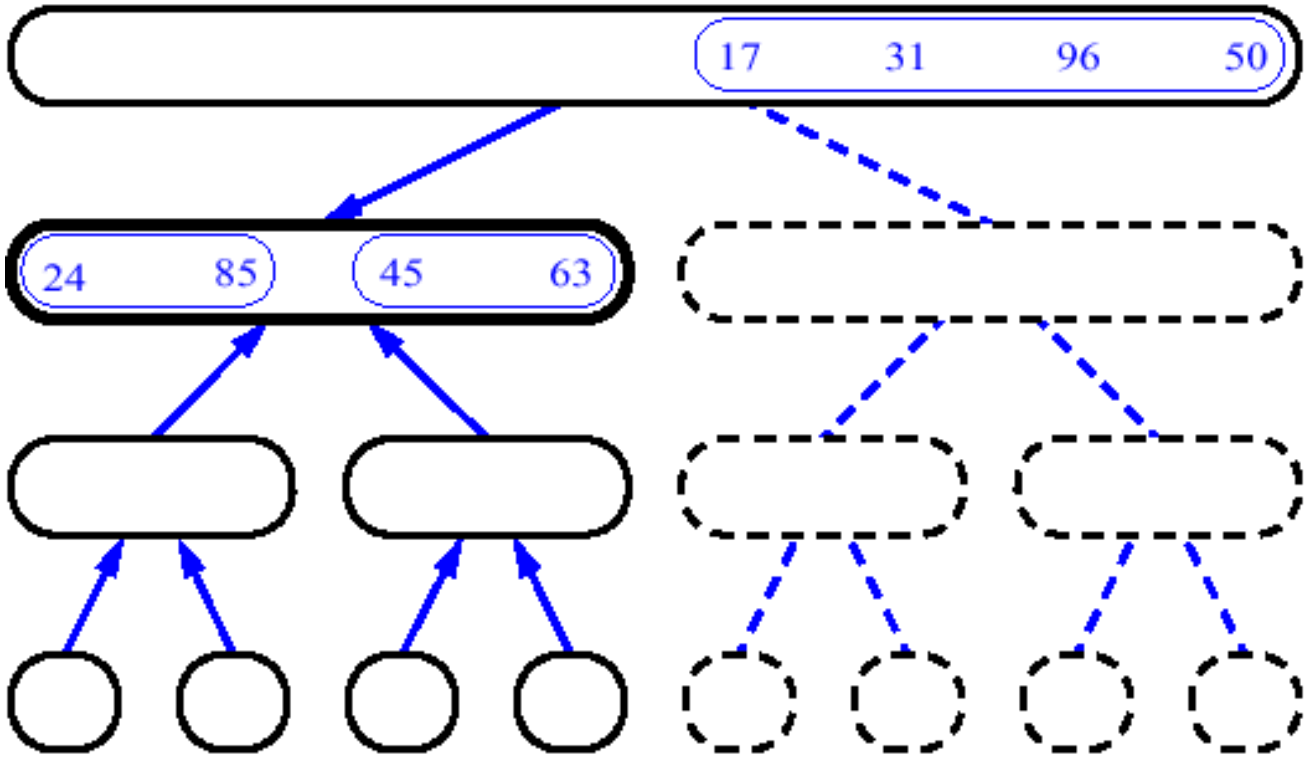
MergeSort (Example) - 14



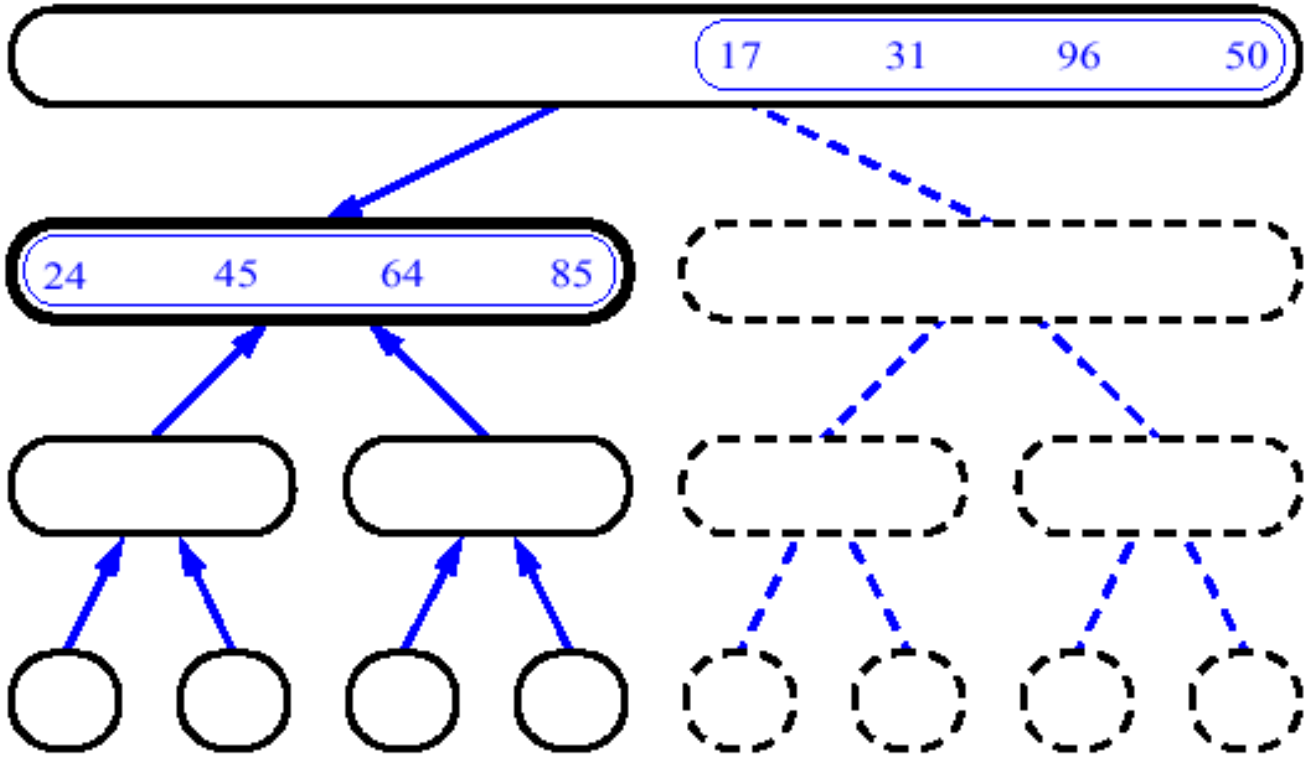
MergeSort (Example) - 15



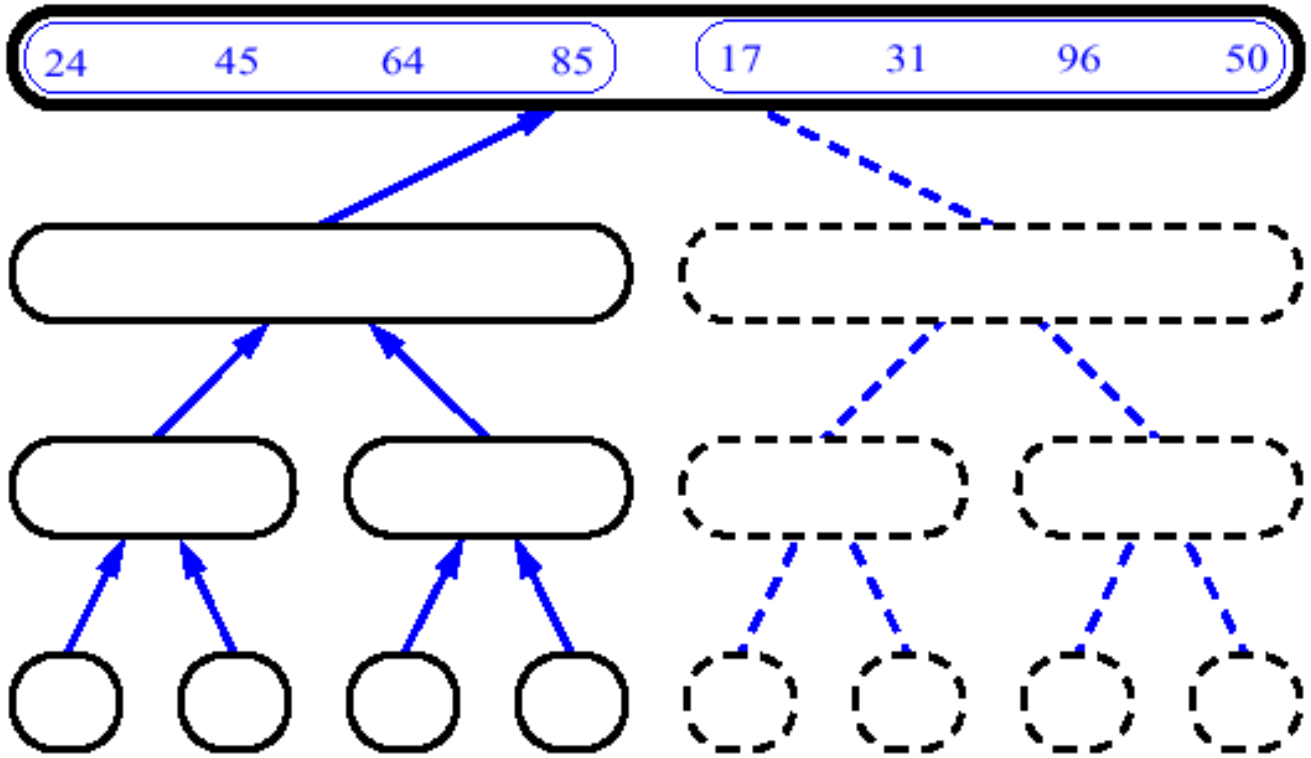
MergeSort (Example) - 16



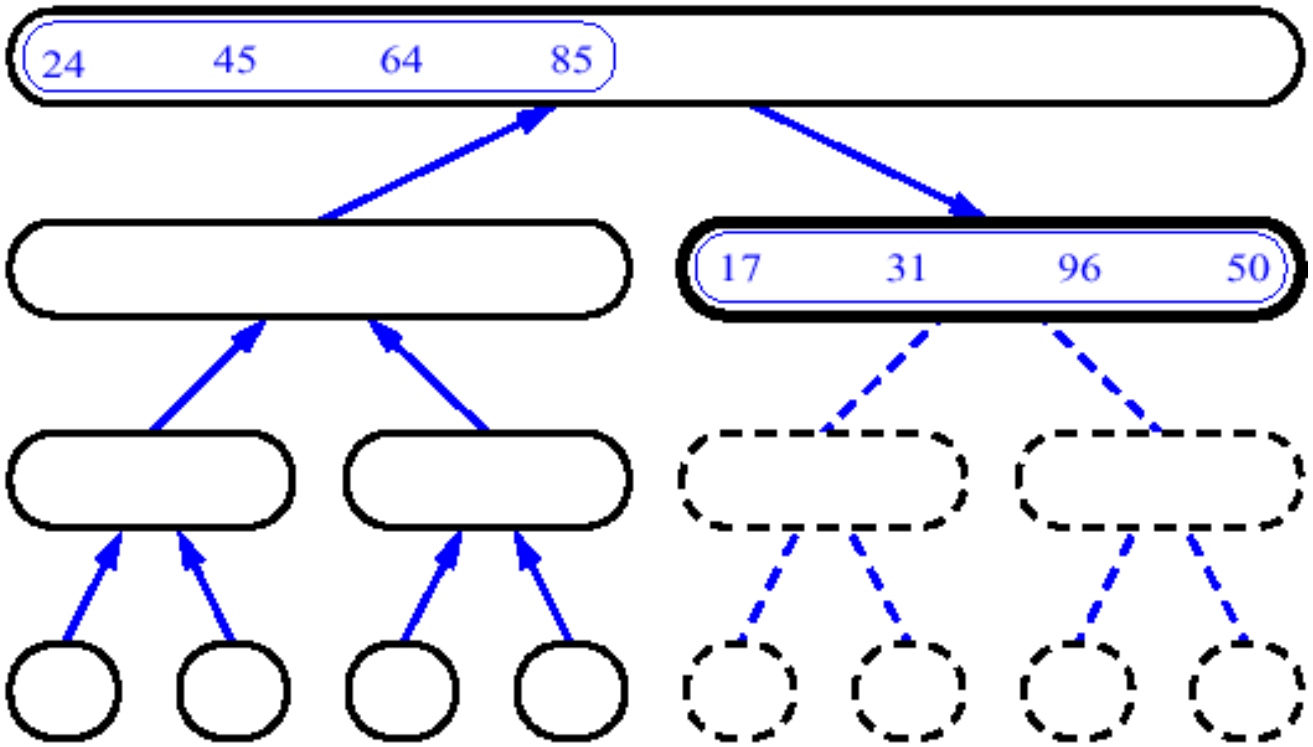
MergeSort (Example) - 17



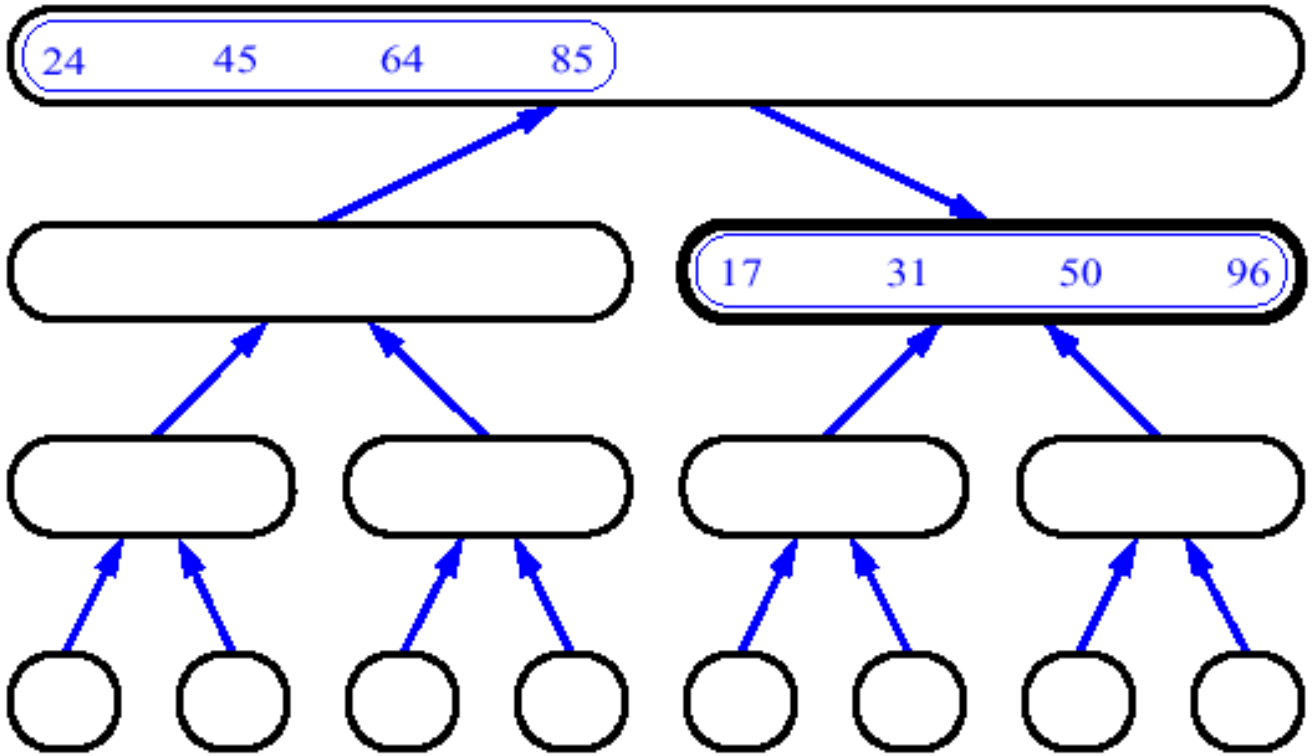
MergeSort (Example) - 18



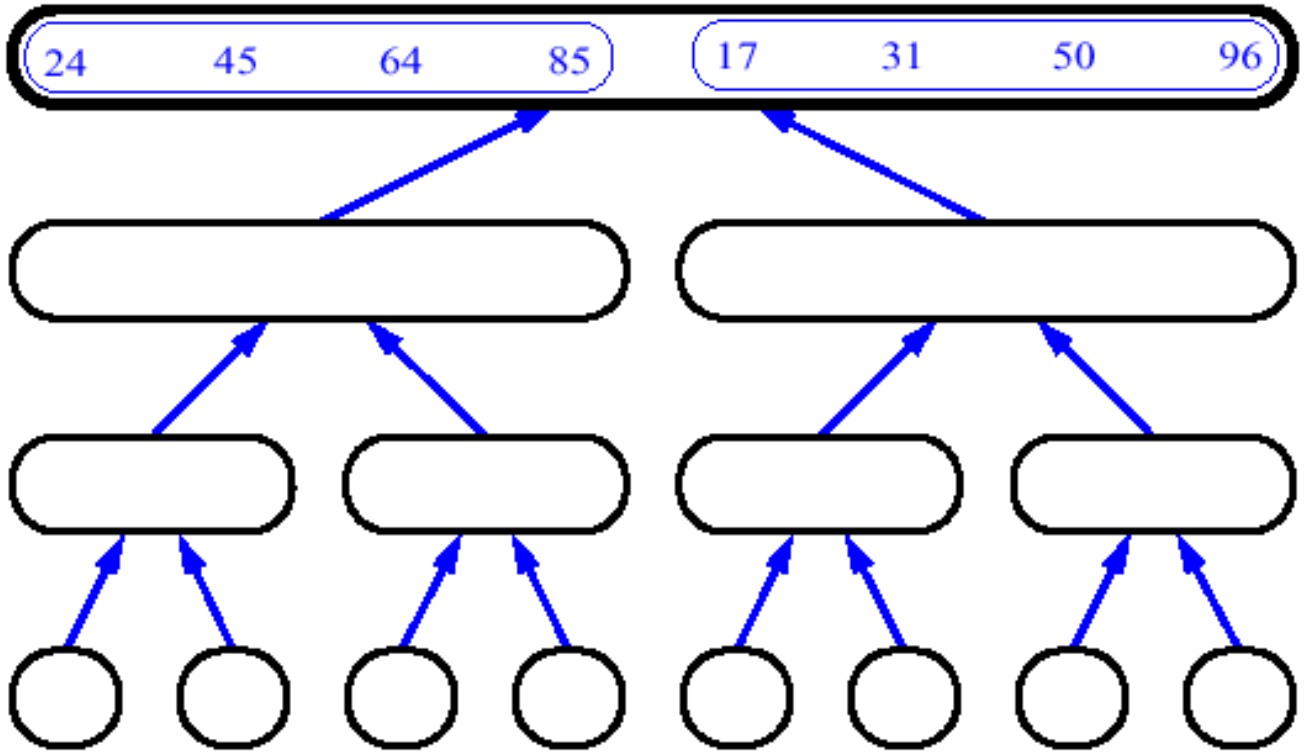
MergeSort (Example) - 19



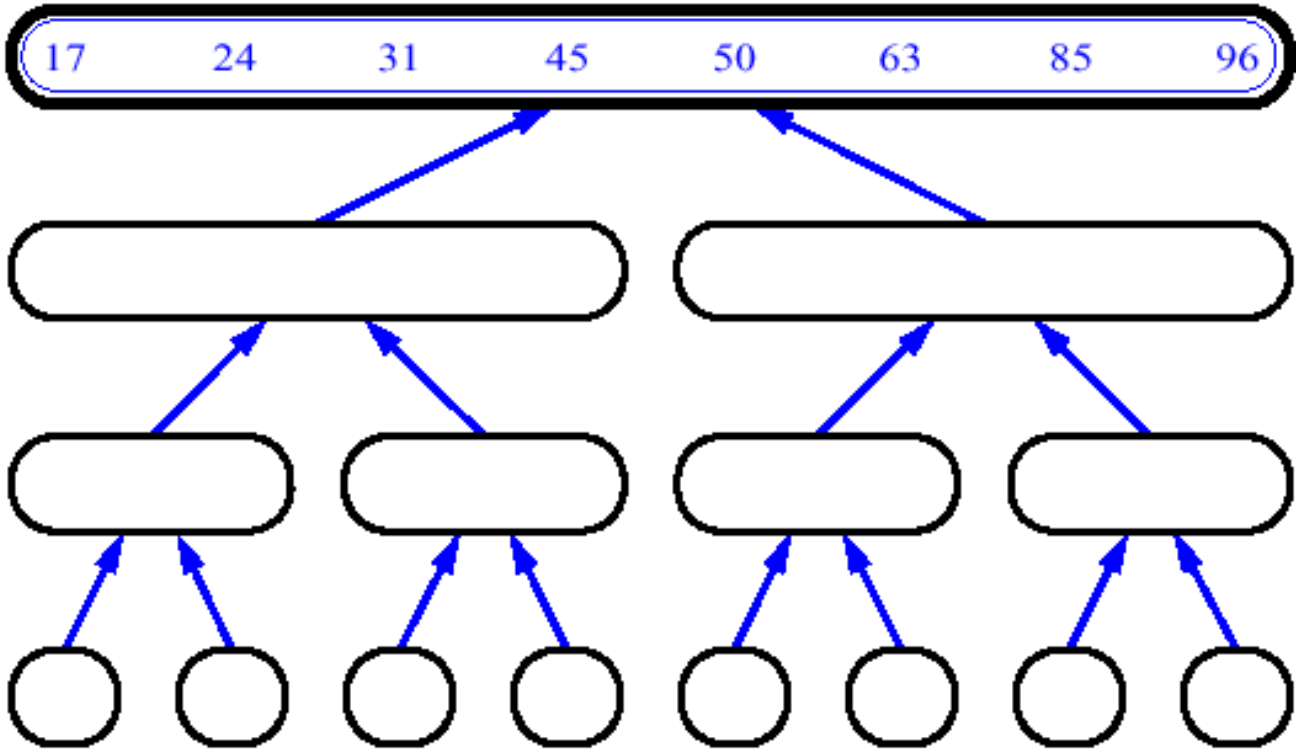
MergeSort (Example) - 20

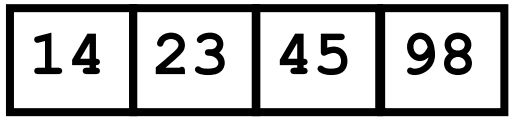


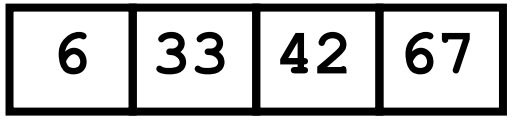
MergeSort (Example) - 21

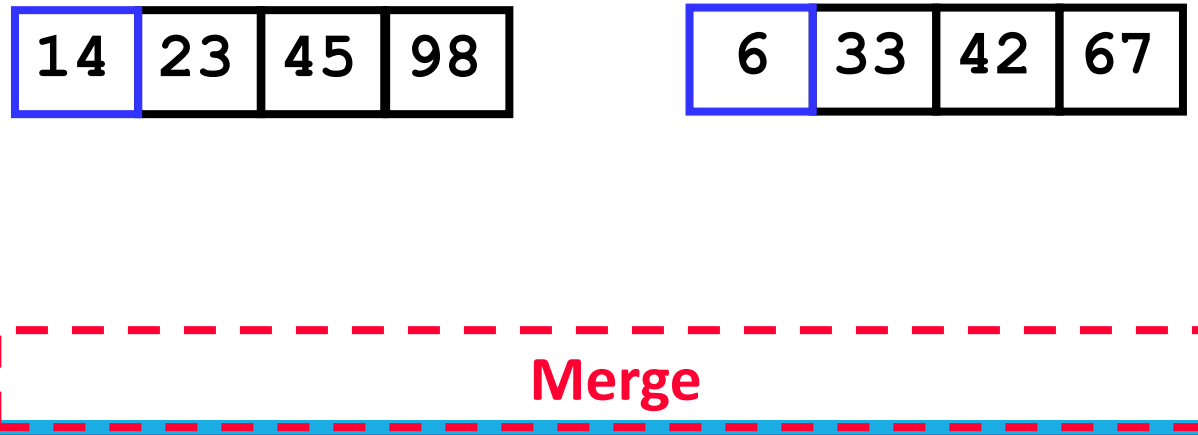


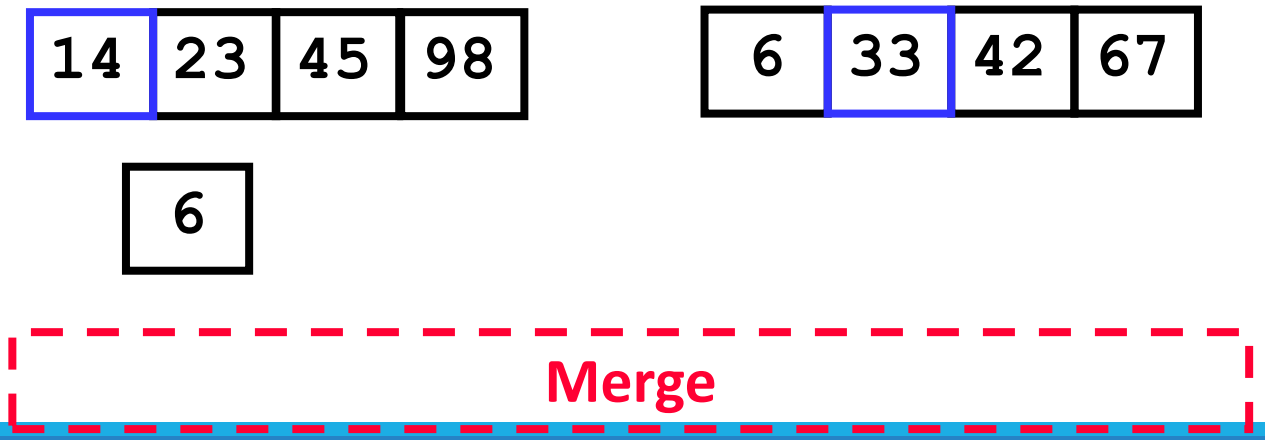
MergeSort (Example) - 22

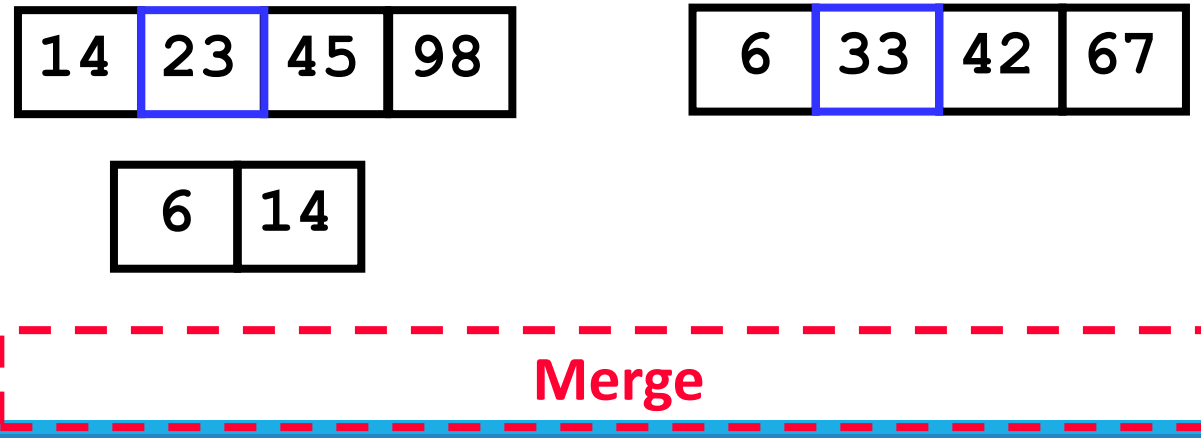


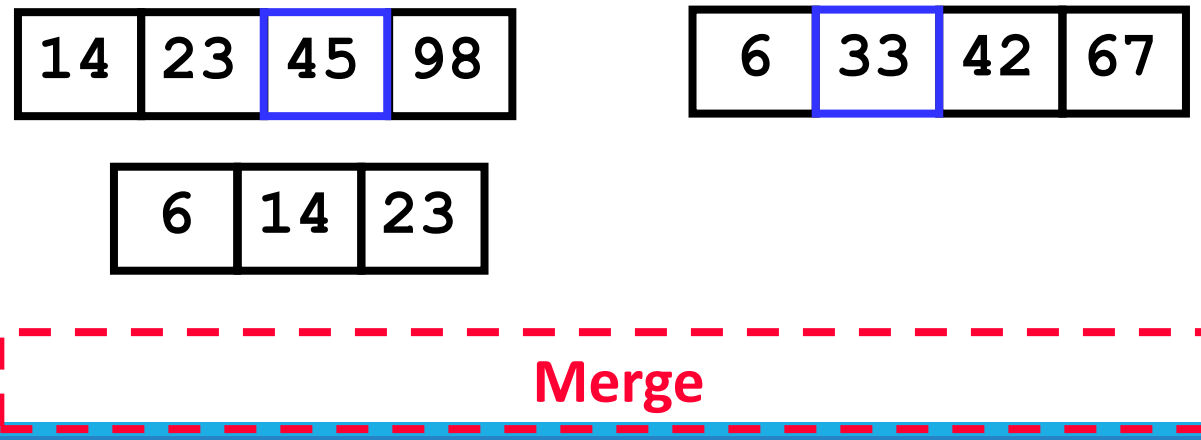

14
23
45
98

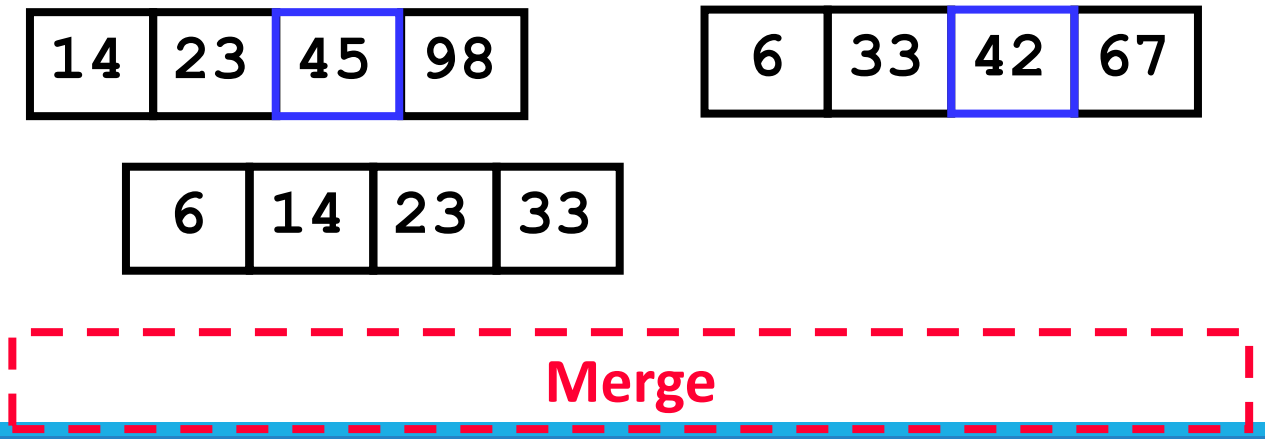

6
33
42
67

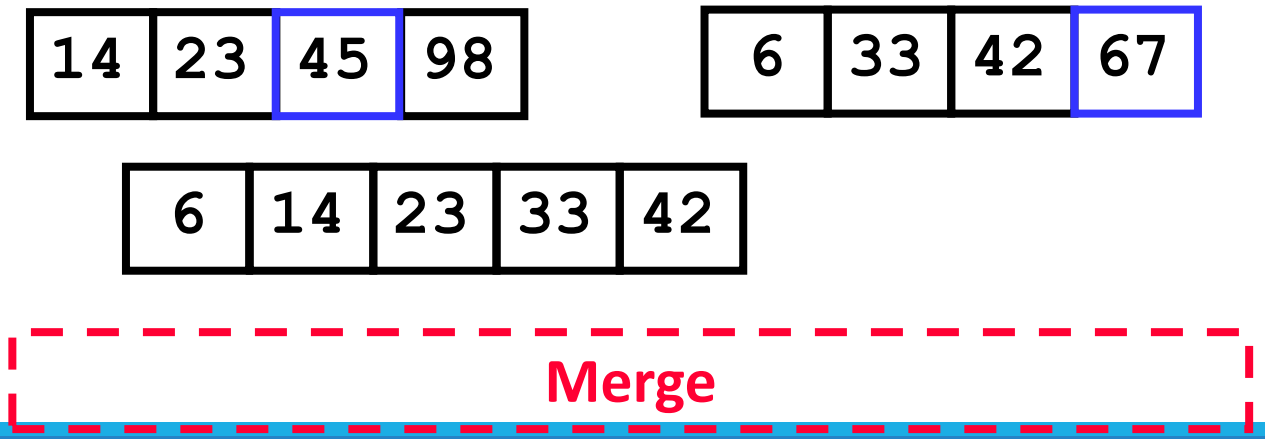

14
23
45
98
6
33
42
67
Merge

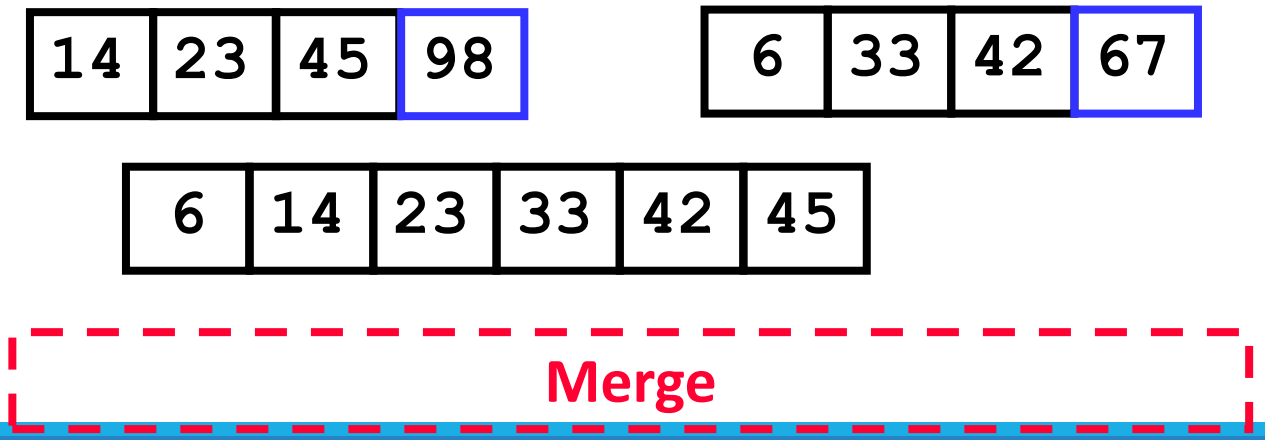

14
23
45
98
6
33
42
67
6
Merge

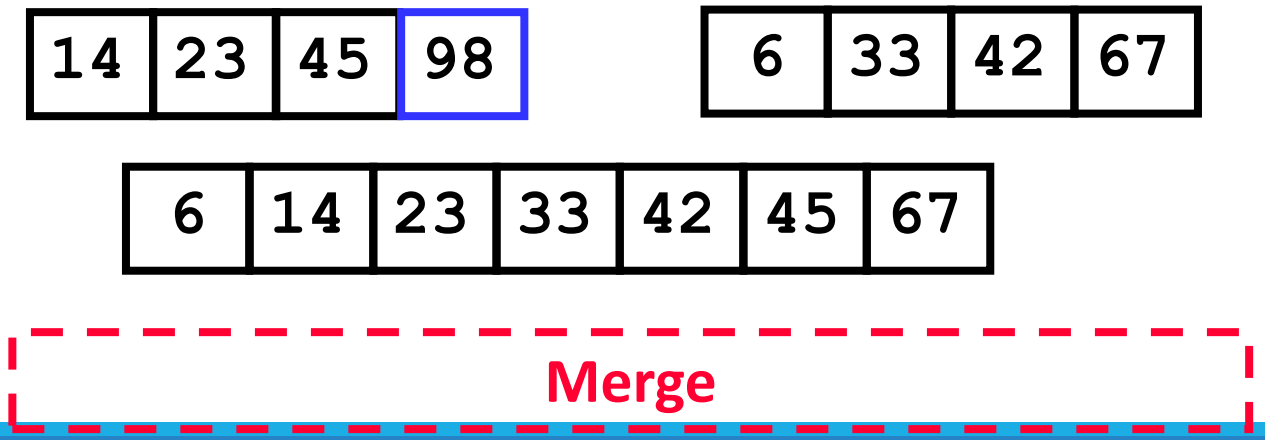

14 23 45 98
6 33 42 67
6 14
Merge

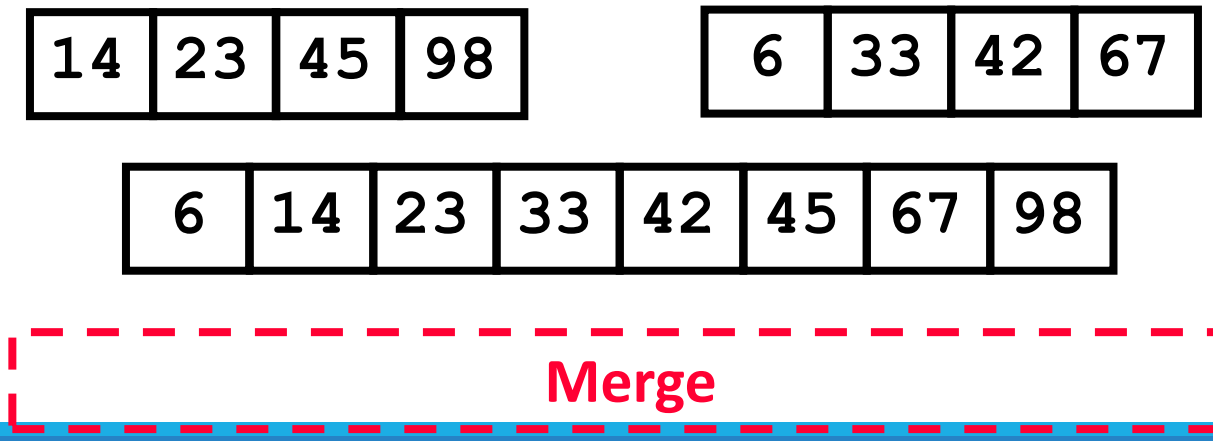

14 23 45 98
6 33 42 67
6 14 23
Merge


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

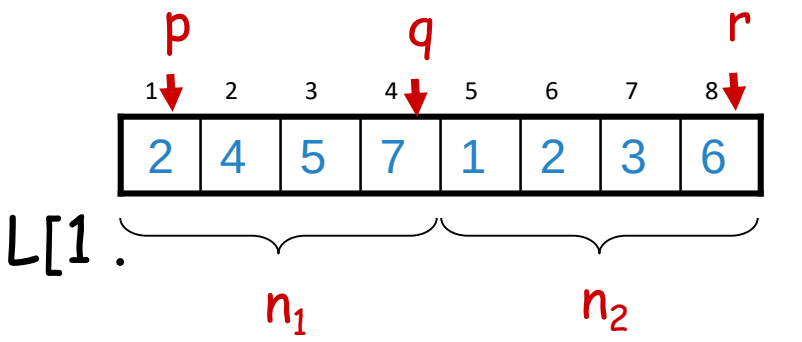

14 23 45 98
6 33 42 67
6 14 23 33 42 45
Merge

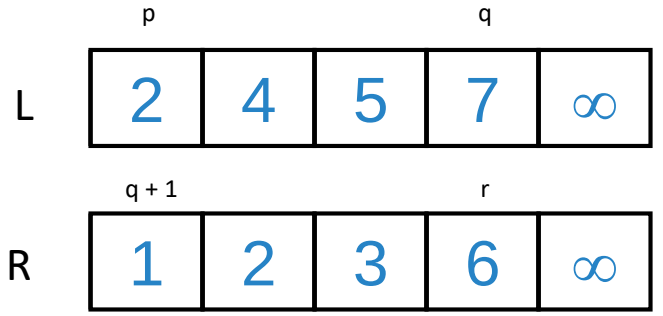

14 23 45 98
6 33 42 67
6 14 23 33 42 45 67
Merge


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

Alg.: MERGE(A, p, q, r)

1. Compute n_1 and n_2
2. Copy the first n_1 elements into $L[1 \,..\, n_1 + 1]$ and the next n_2 elements into $R[1 \,..\, n_2 + 1]$
3. $L[n_1 + 1] \leftarrow \infty$; $R[n_2 + 1] \leftarrow \infty$
4. $i \leftarrow 1$; $j \leftarrow 1$
5. **for** $k \leftarrow p$ **to** r
6. **do if** $L[\, i \,] \leq R[\, j \,]$
7. **then** $A[k] \leftarrow L[\, i \,]$
8. $i \leftarrow i + 1$
9. **else** $A[k] \leftarrow R[\, j \,]$
10. $j \leftarrow j + 1$





PSEUDOCODE

QUICK SORT

SORTING ALGORITHM

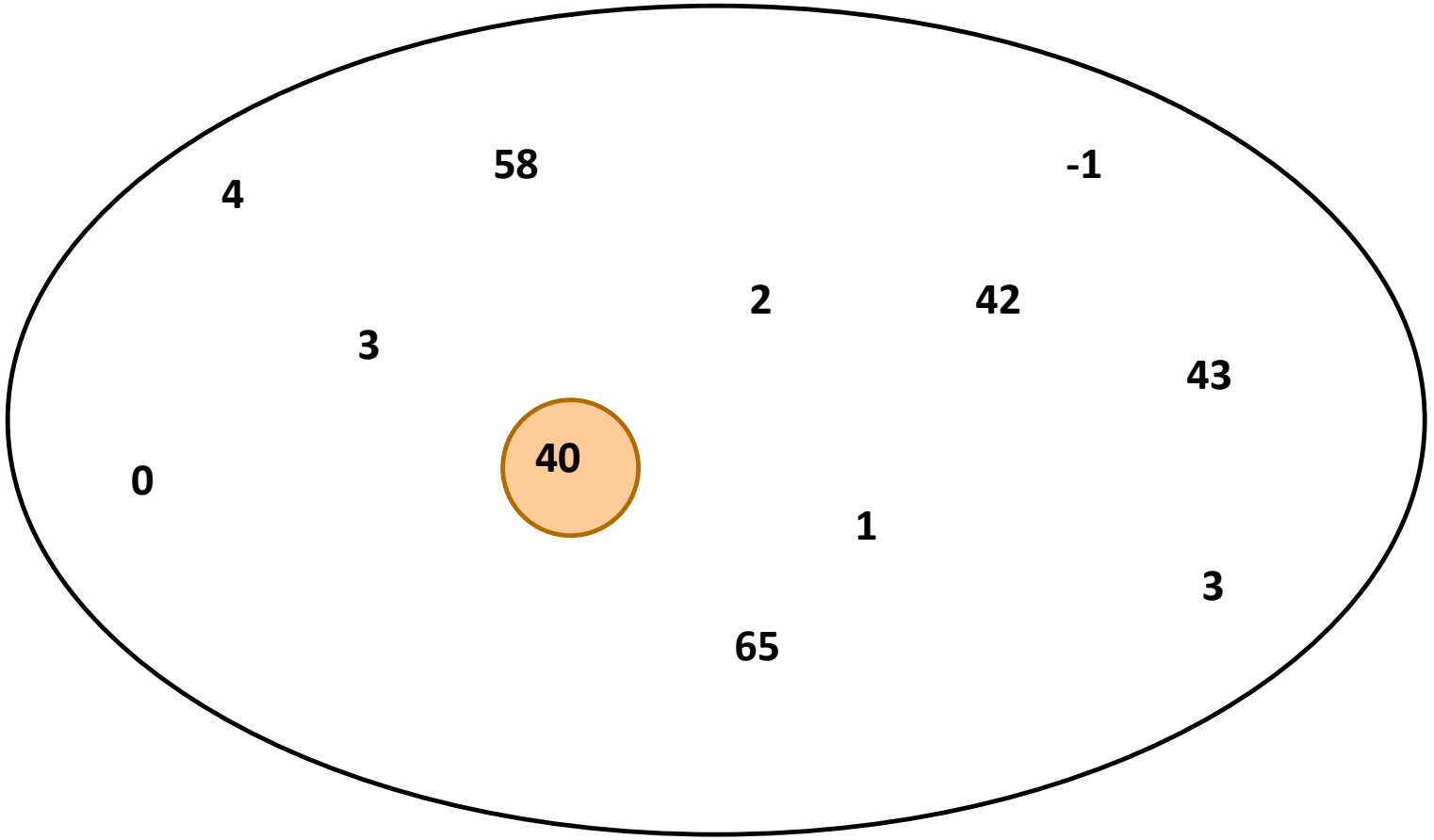
Quick sort: Ide dasar

Divide and conquer approach

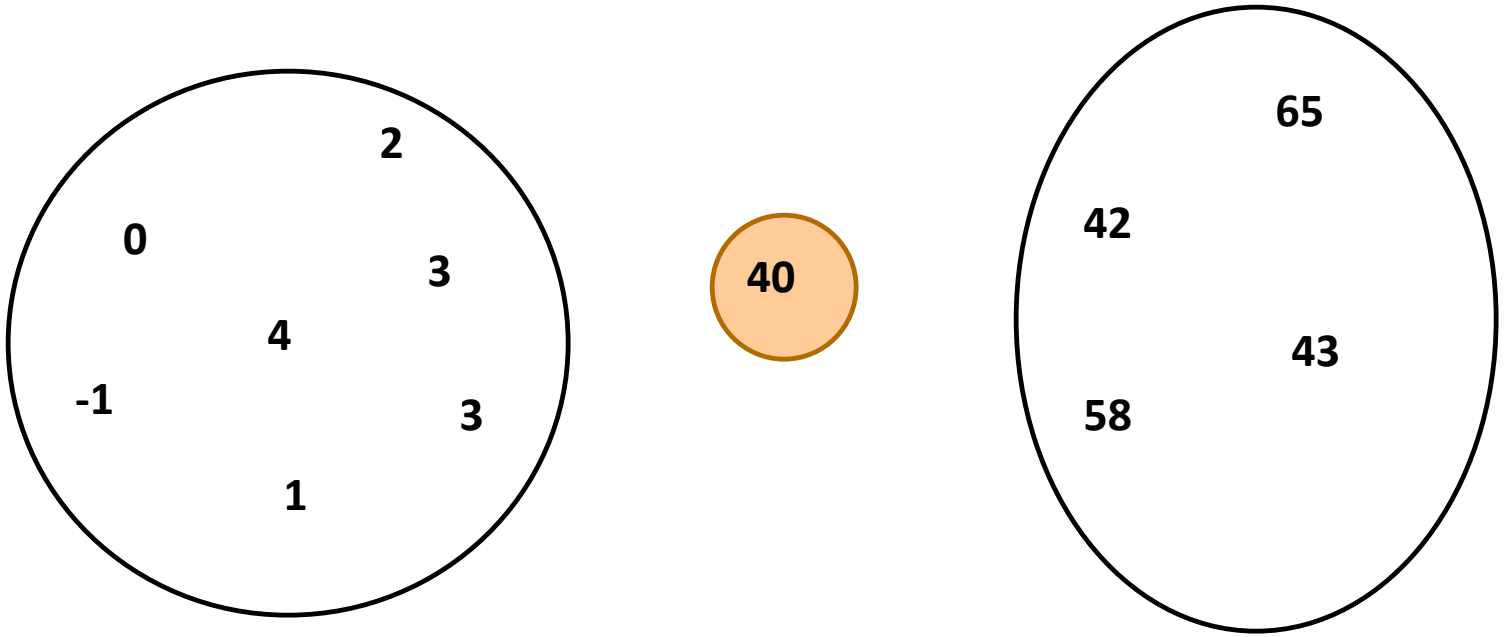
Algoritma *quickSort(S)*:

- Jika jumlah elemen dalam S = 0 atau 1, return.
- Pilih sembarang elemen $v \in S$ – sebutlah pivot.
- Partisi S – {v} ke dalam 2 bagian:
  - $L = \{x \in S - \{v\} \mid x \leq v\}$
  - $R = \{x \in S - \{v\} \mid x \geq v\}$
- Kembalikan nilai *quickSort(S)*, diikuti *v*, diikuti *quickSort(S)*.

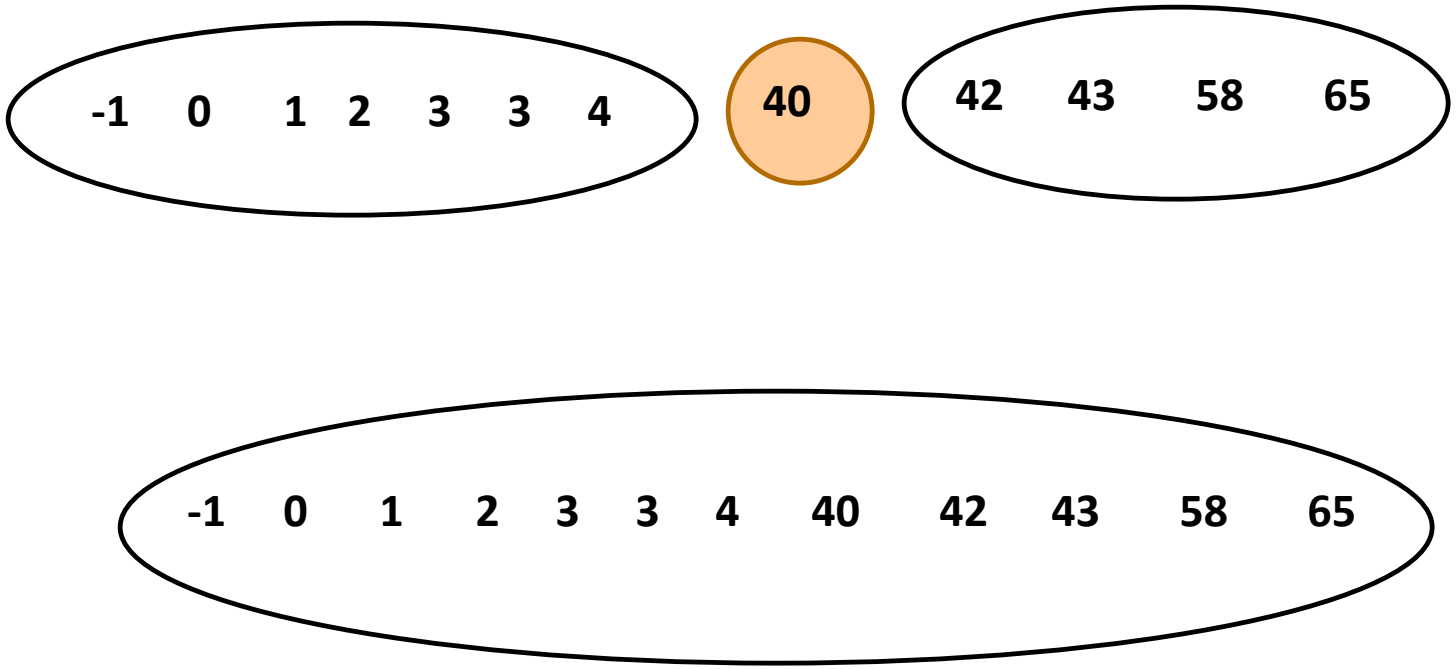
Quick sort: Pilih elemen *pivot*



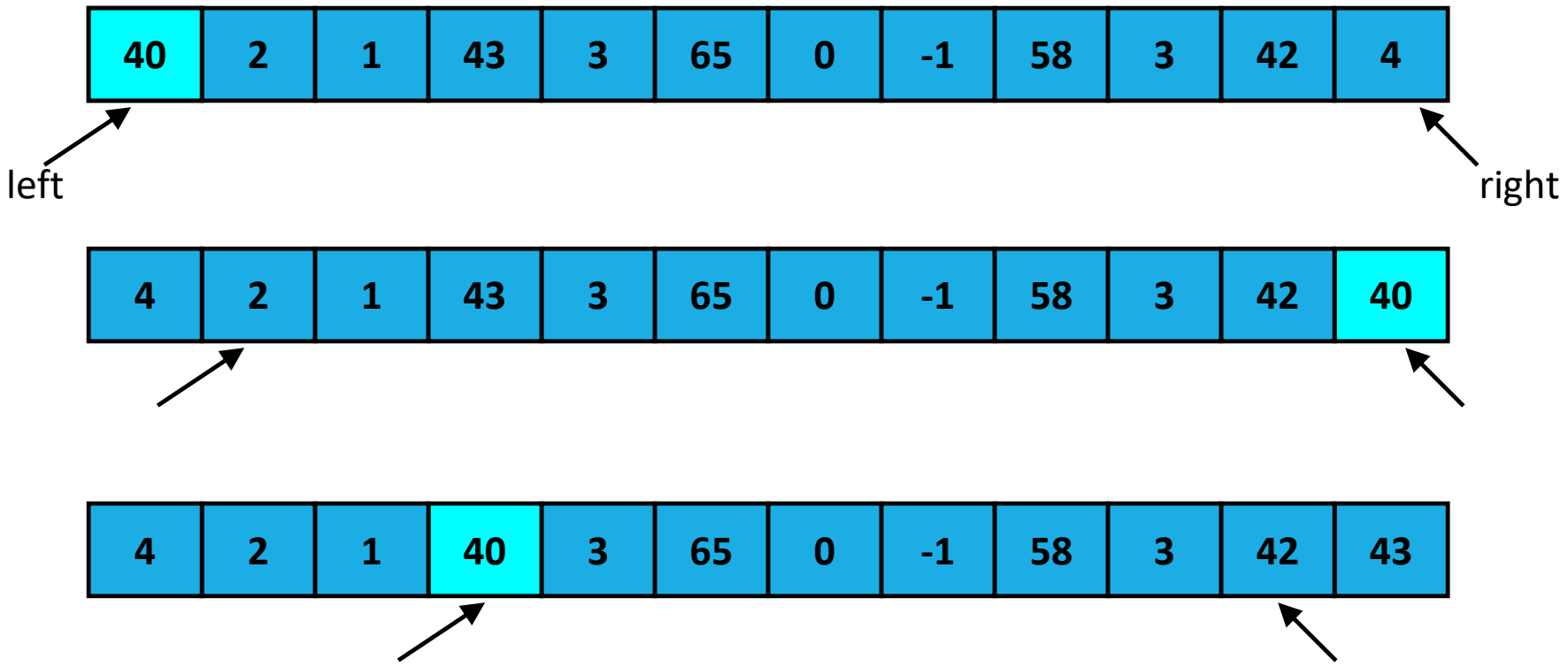
Quick sort: Partition



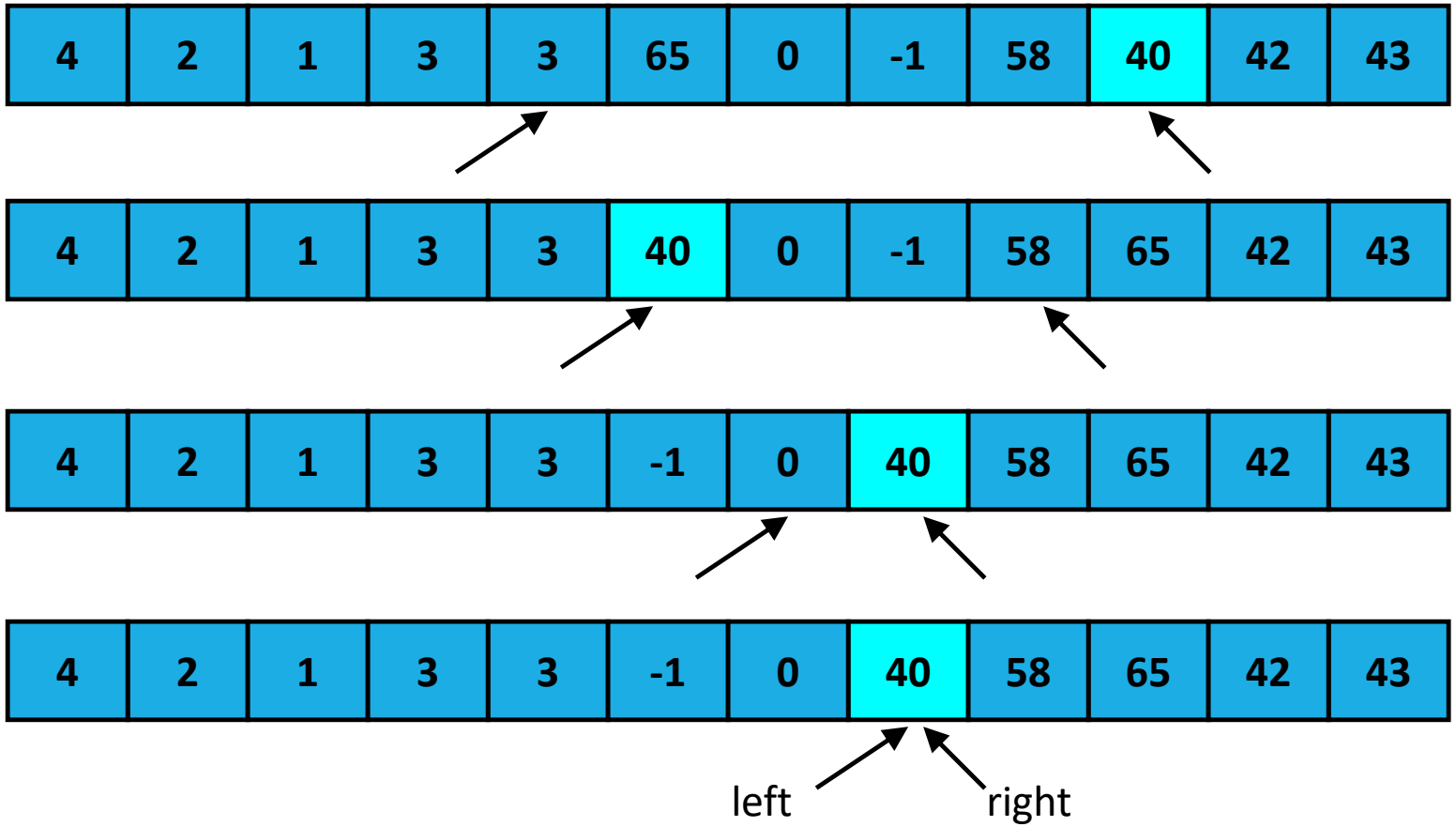
Quick sort: Sort scr. rekursif, gabungkan



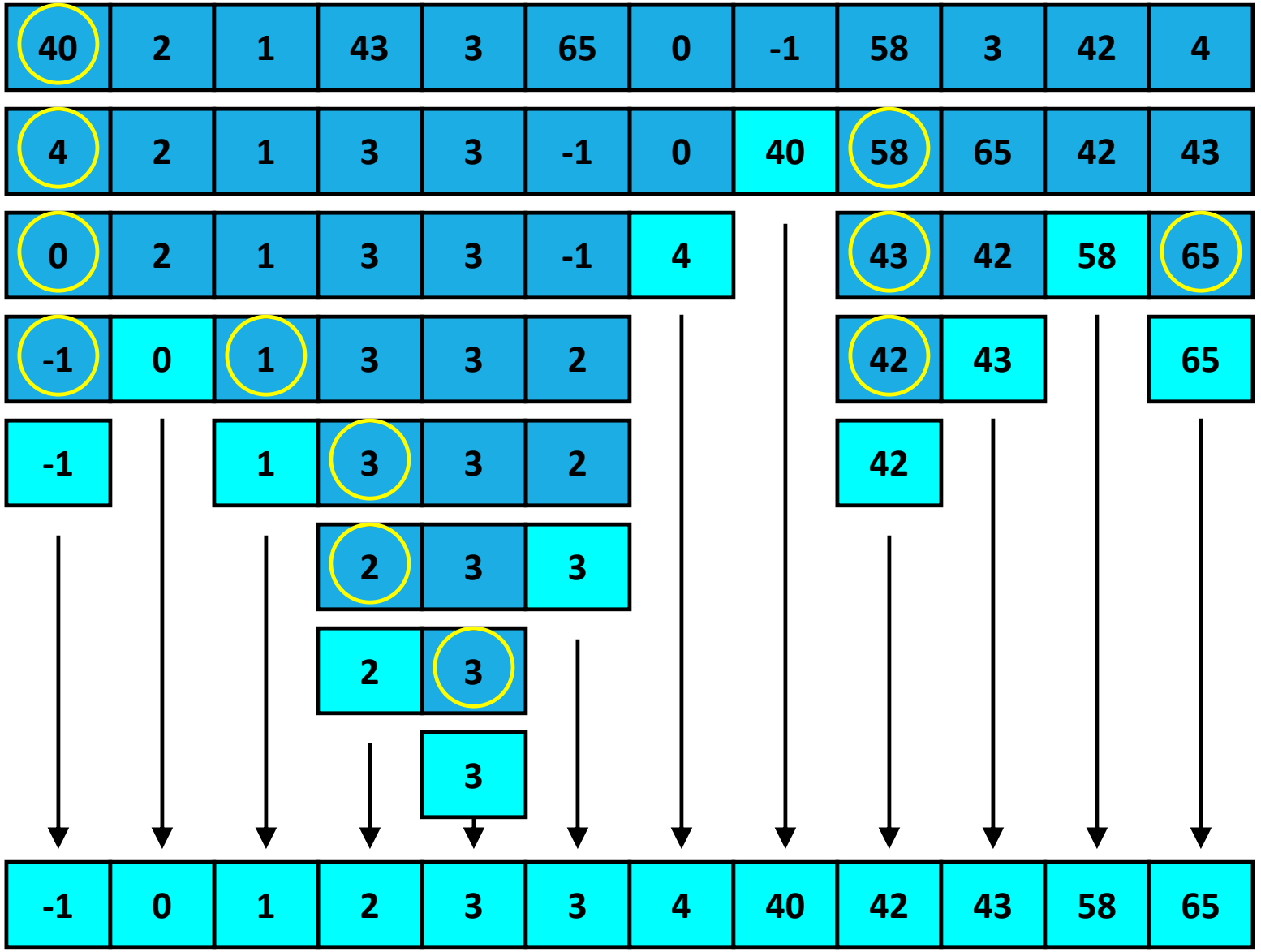
Quick sort: Partition algorithm 1



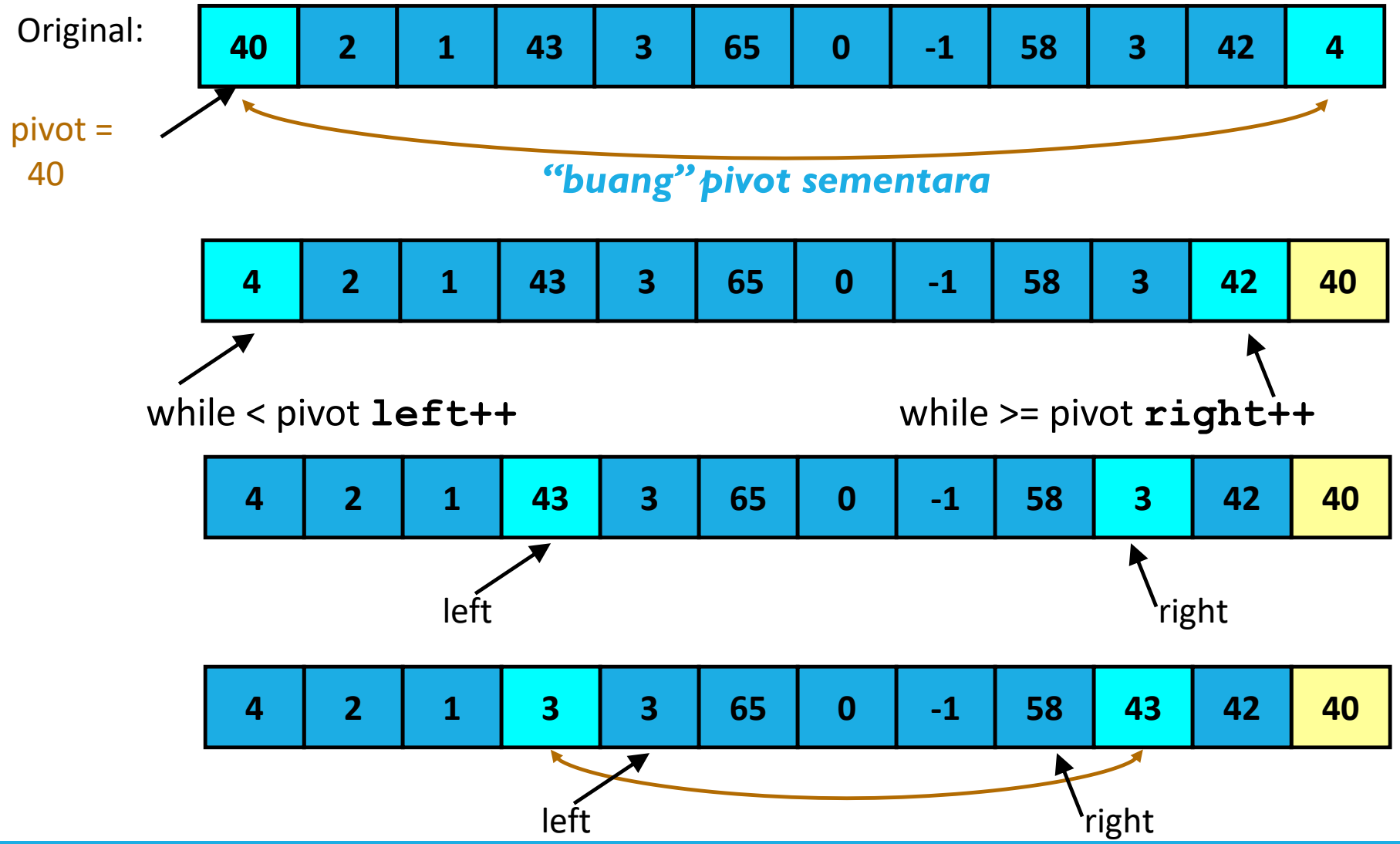
Quick sort: Partition algorithm 1



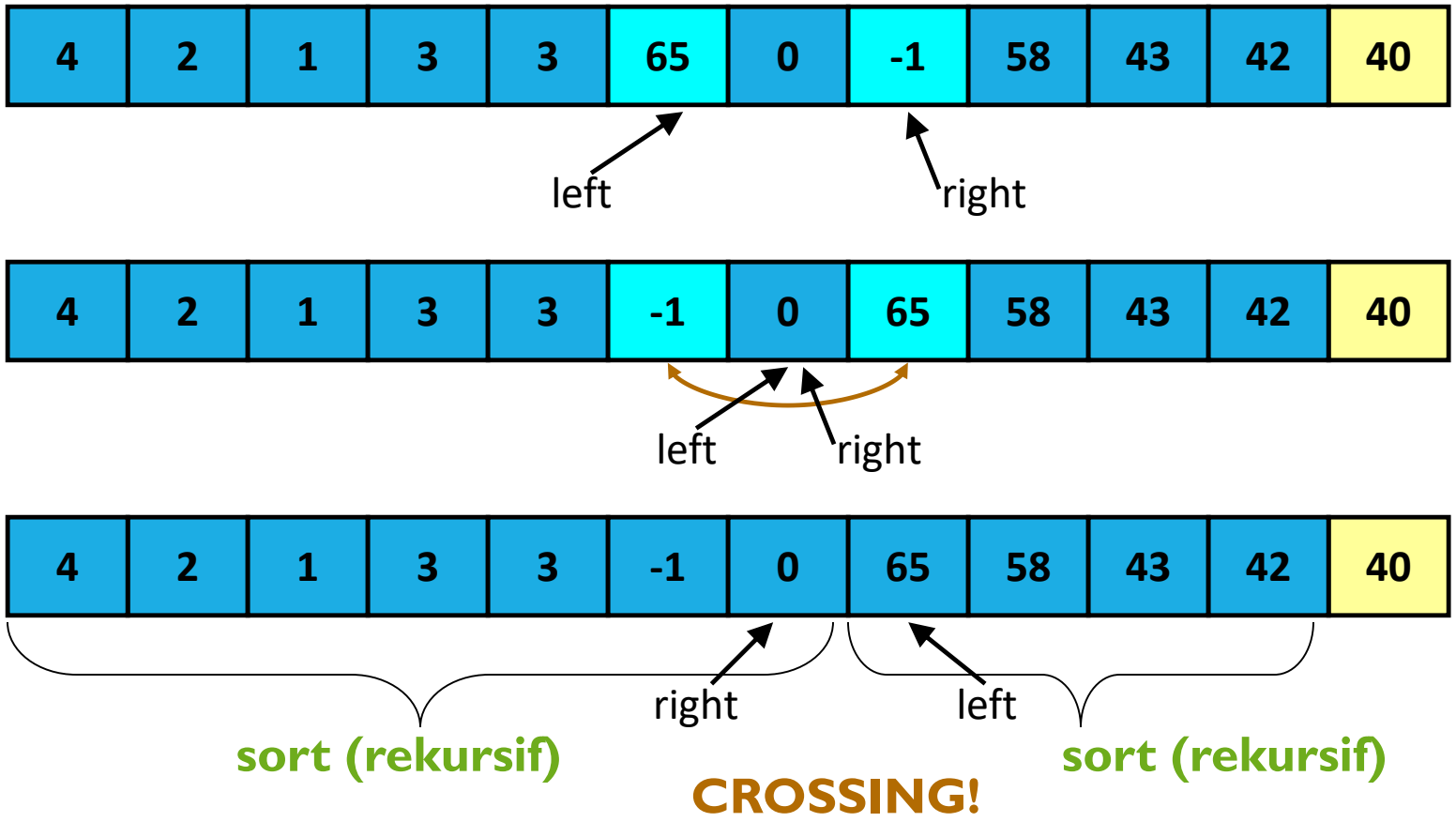
QUICK SORT : PARTITION ALGORITM (1)



Quick sort: Partition algorithm 2



Quick sort: Partition algorithm 2



Quick sort: Implementasi

```
static void quickSort(int a[], int low, int high)
{
    if(high <= low) return; // base case
    pivot = choosePivot(a); // select "best" pivot

    int i=low, j=high-1;
    swap(a,pivot,a[j]);     // move pivot out of the way
    while(i <= j)
    {
        // find large element starting from left
        while(i<high && a[i]<pivot) i++;

        // find small element starting from right
        while(j>low && a[j]>=pivot) j--;

        // if the indexes have not crossed, swap
        if(i>j) swap(a, i, j);
    }
    swap(a,i,high-1);       // restore pivot
    quickSort(a,low,i-1);   // sort small elements
    quickSort(a,i+1,high);  // sort large elements
}
```

Quick sort: Memilih pivot

Pivot ideal:

- Elemen media

Yang biasa dijadikan calon pivot:

- Elemen pertama
- Elemen terakhir
- Elemen di tengah-tengah
- Median dari ketiga elemen tsb.

Source

http://www.cs.sfu.ca/CourseCentral/307/jmanuch/lec/MergeSort.ppt

http://www.cse.unr.edu/~bebis/CS477/Lect/MergesortQuickSort.ppt

Slide Kuliah Fasilkom UI

Next....

- Bucket Sort
- Shell Sort
- Radix Sort
- External Sort