

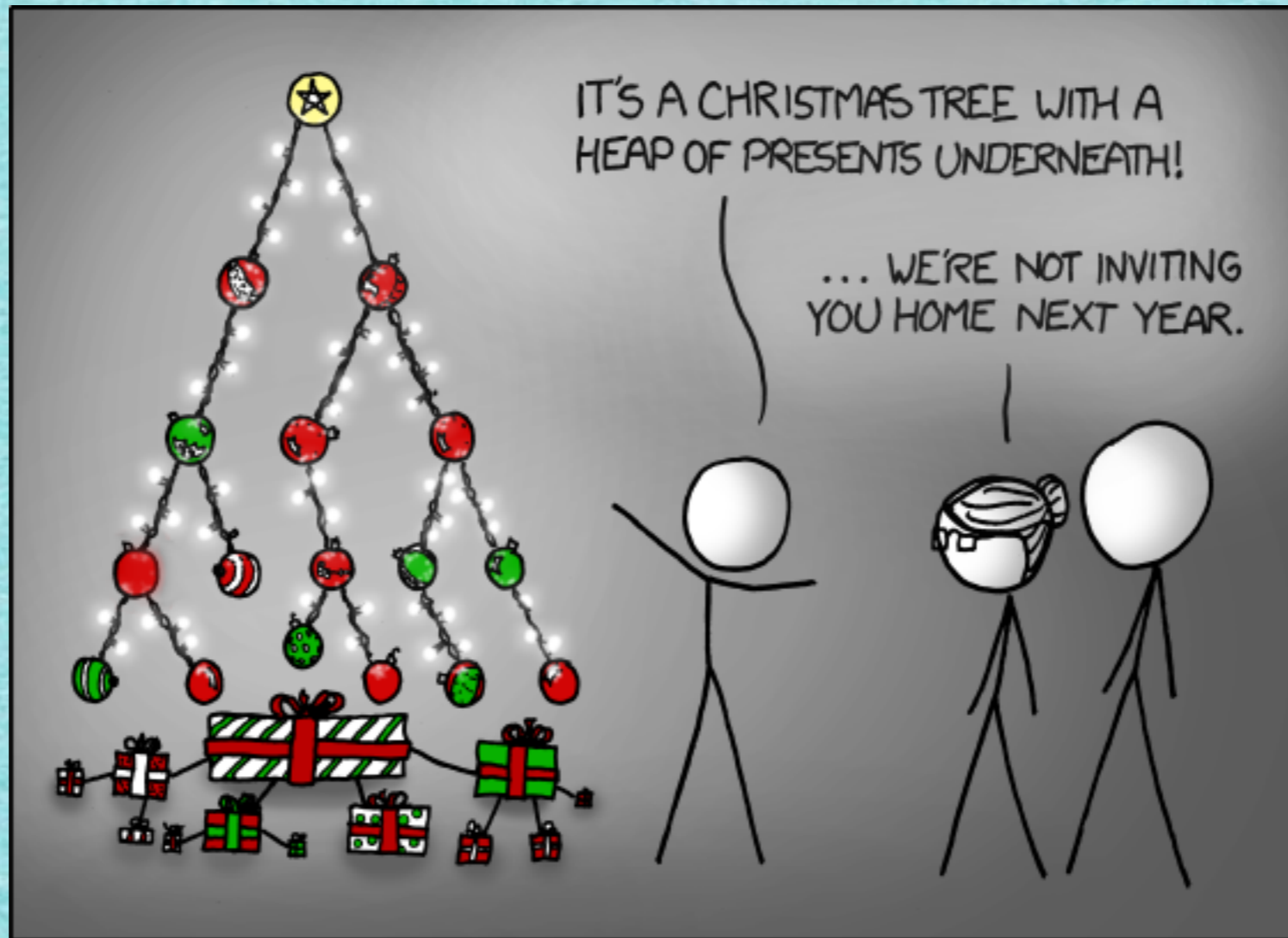


# Kapitel 5: Sortieren

*Algorithmen und Datenstrukturen  
WS 2021/22*

**Prof. Dr. Sándor Fekete**









## Wer ist eigentlich dieser knut?

Knut hat in Schweden eine fast 1.000 Jahre alte Tradition. Der **13.1.**, der St.-Knuts-Tag, hat seinen Namen von Knut IV., einem König, der von Weihnachten nicht genug bekommen konnte. Also hat er das **Fest auf 20 Tage verlängert**. Seitdem endet es jedes Jahr mit dem Knut-Tag, an dem wir **abschmücken und Platz für Neues schaffen**.



## Donald Knuth



Knuth in 2005

### Born

Donald Ervin Knuth  
January 10, 1938 (age 83)  
[Milwaukee, Wisconsin, U.S.](#)



THE CLASSIC WORK  
NEWLY UPDATED AND REVISED

# The Art of Computer Programming

VOLUME 3

Sorting and Searching  
Second Edition

DONALD E. KNUTH

**Donald Knuth**



Knuth in 2005

**Born**

Donald Ervin Knuth  
January 10, 1938 (age 83)  
Milwaukee, Wisconsin, U.S.



THE CLASSIC WORK  
NEWLY UPDATED AND REVISED

# The Art of Computer Programming

VOLUME 3

Sorting and Searching  
Second Edition

DONALD E. KNUTH

**Donald Knuth**



Knuth in 2005

**Born**

Donald Ervin Knuth  
January 10, 1938 (age 83)  
Milwaukee, Wisconsin, U.S.

# 5.1 Aufgabenstellung



# Sortieren von Objekten

# Sortieren von Objekten

**Gegeben:**

# Sortieren von Objekten

**Gegeben:**  $n$  Objekte unterschiedlicher Größe

# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

23 17 13 19 33 28 15

# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

23 17 13 19 33 28 15

**Gesucht:**



# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

23 17 13 19 33 28 15

**Gesucht:** Eine Sortierung nach Größe

# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

13 15 17 19 23 28 33

**Gesucht:** Eine Sortierung nach Größe

# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

23 17 13 19 33 28 15

**Gesucht:** Eine Sortierung nach Größe

# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

23 17 19 33 28 15  
13

**Gesucht:** Eine Sortierung nach Größe

# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

23 17 19 33 28  
13 15

**Gesucht:** Eine Sortierung nach Größe

# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

23                      19 33 28  
13 15 17

**Gesucht:** Eine Sortierung nach Größe



# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

33 28  
13 15 17 19 23

**Gesucht:** Eine Sortierung nach Größe



# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

33  
13 15 17 19 23 28

**Gesucht:** Eine Sortierung nach Größe

# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

13 15 17 19 23 28 33

**Gesucht:** Eine Sortierung nach Größe

# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

13 15 17 19 23 28 33

# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

13 15 17 19 23 28 33

Zahl der Vergleiche:

# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

13 15 17 19 23 28 33

Zahl der Vergleiche:

$O(n^2)$

# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

13 15 17 19 23 28 33

Zahl der Vergleiche:

$O(n^2)$

„Geht's nicht noch etwas schneller?“

# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

13 15 17 19 23 28 33

Zahl der Vergleiche:

„Geht's nicht noch etwas schneller?“

# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

13 15 17 19 23 28 33

Zahl der Vergleiche:



# Sortieren von Objekten

**Gegeben:** n Objekte unterschiedlicher Größe

13 15 17 19 23 28 33

Zahl der Vergleiche:

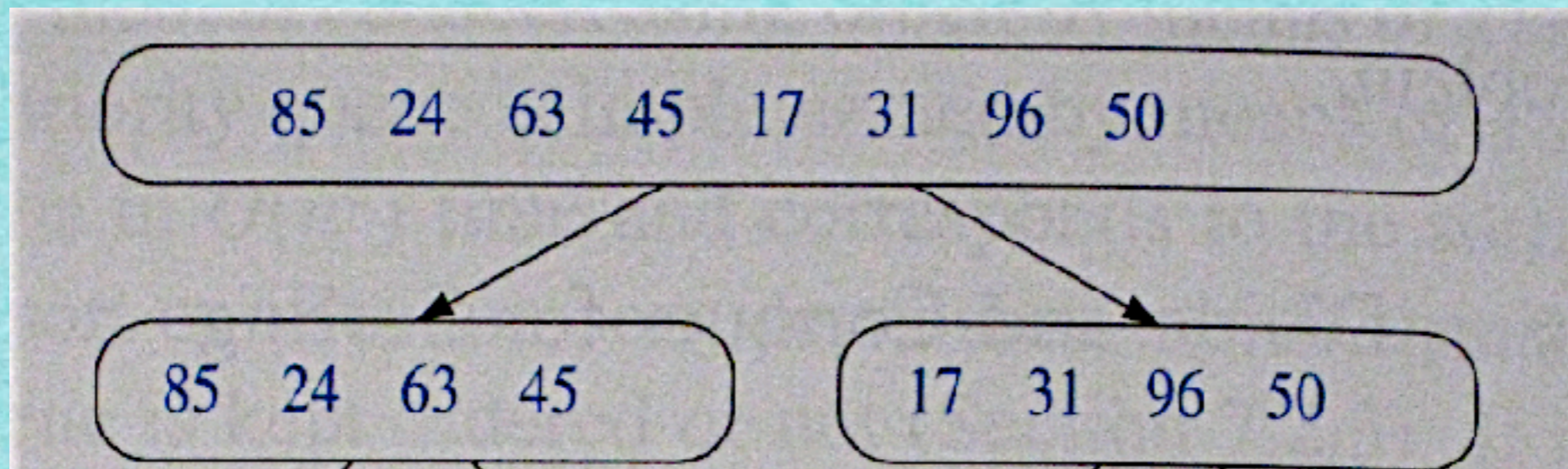
$O(n \log n)$

# 5.2 Mergesort

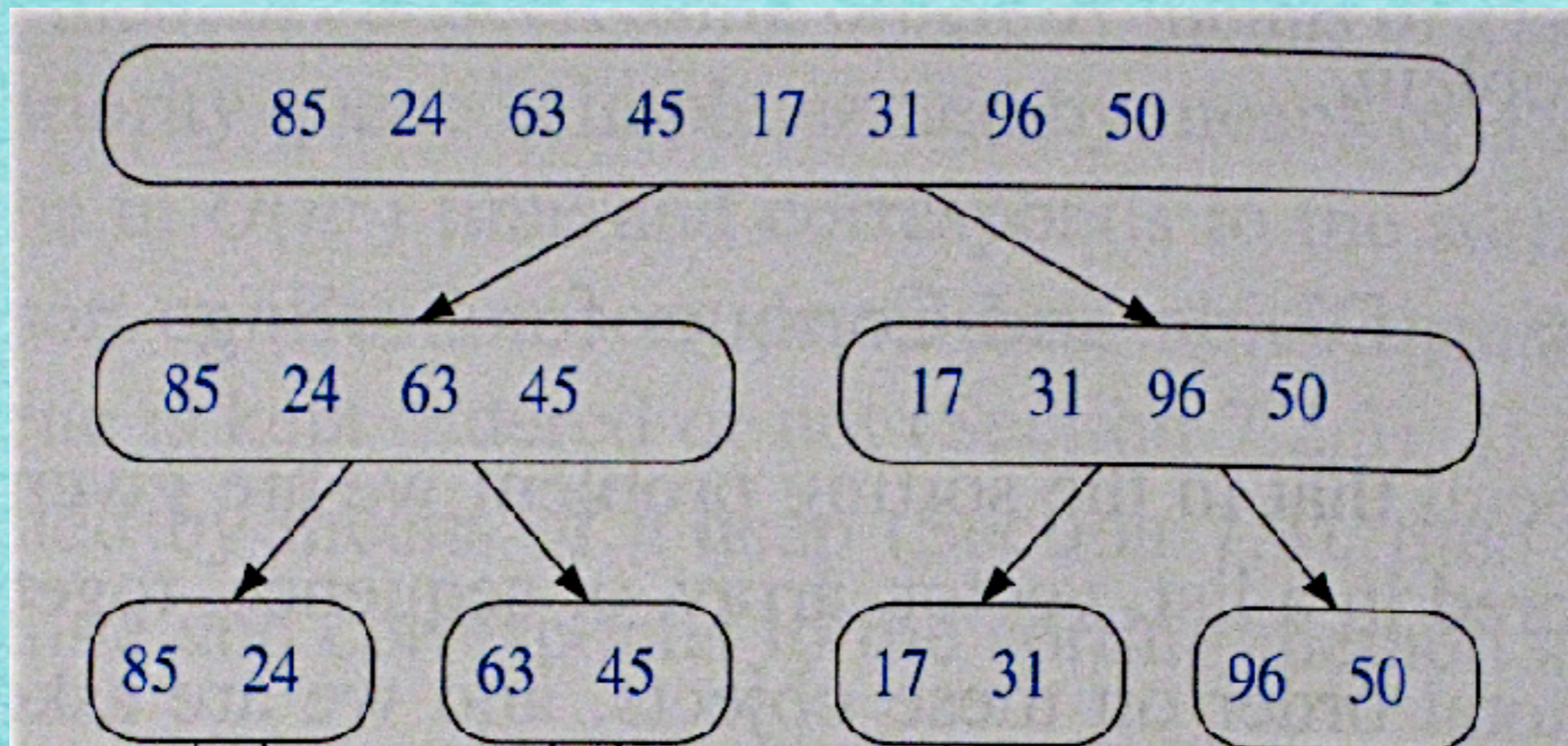
# 5.2 Mergesort

85 24 63 45 17 31 96 50

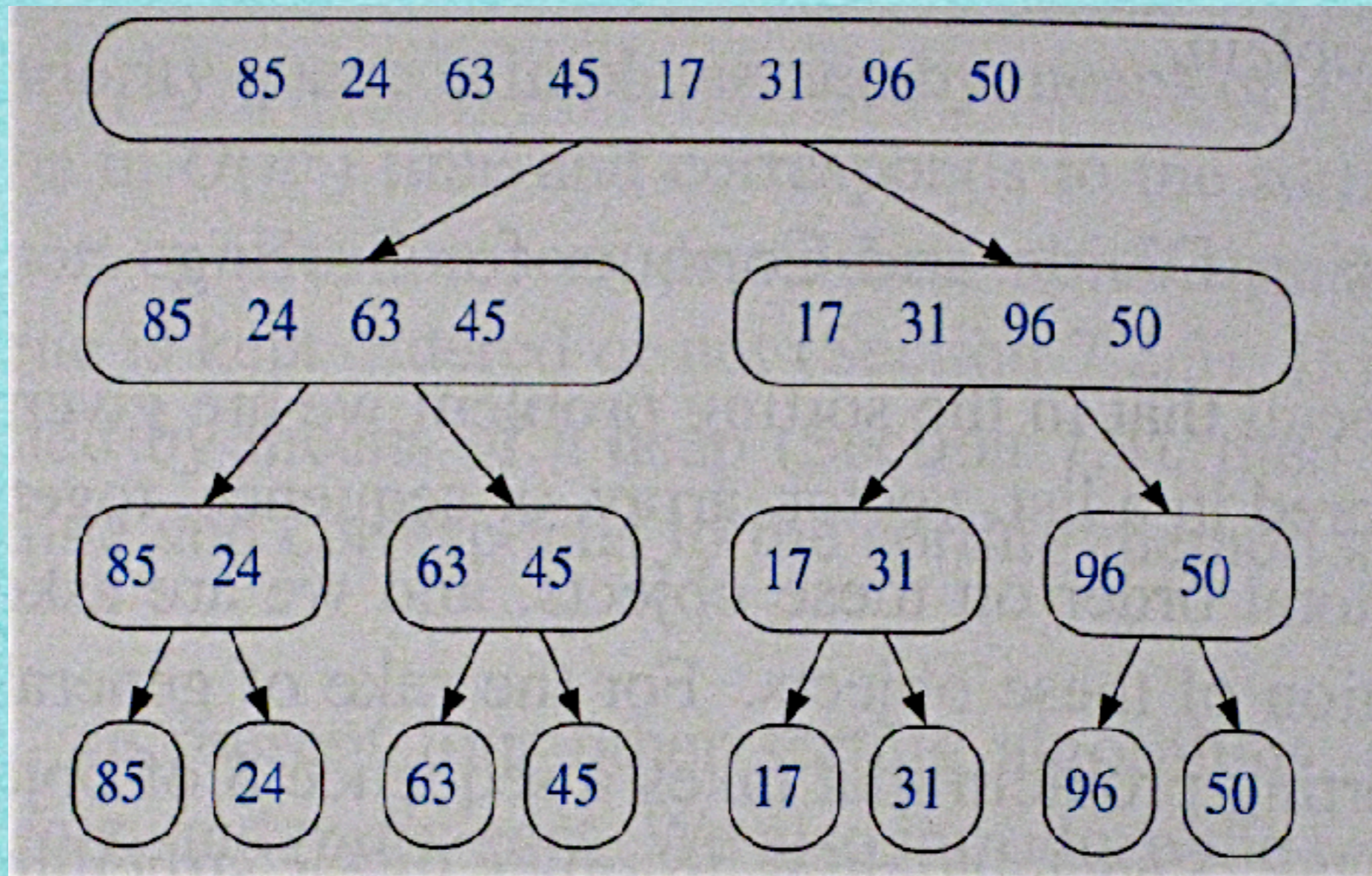
# 5.2 Mergesort



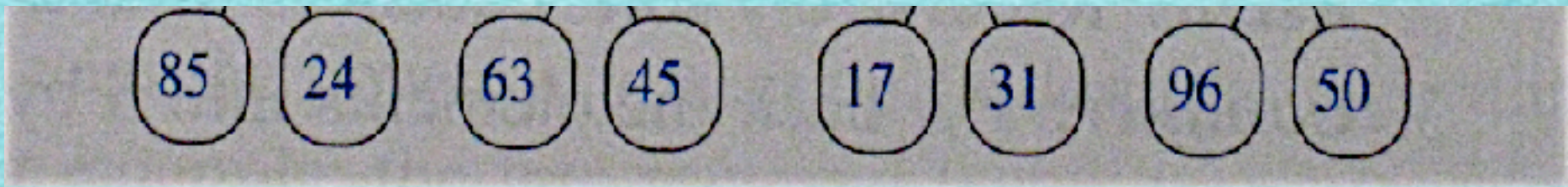
# 5.2 Mergesort



# 5.2 Mergesort

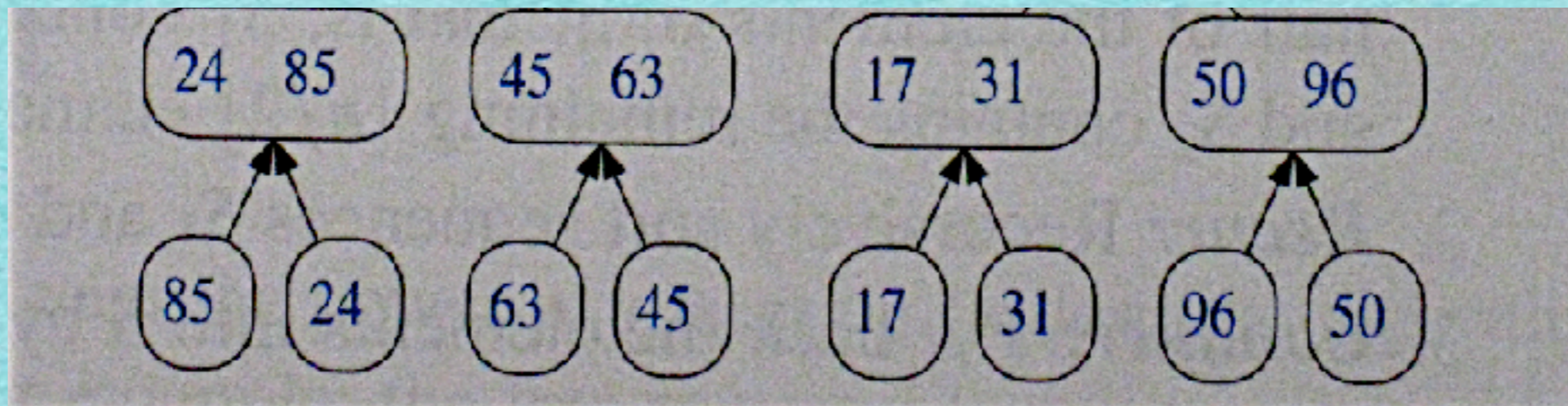


# 5.2 Mergesort



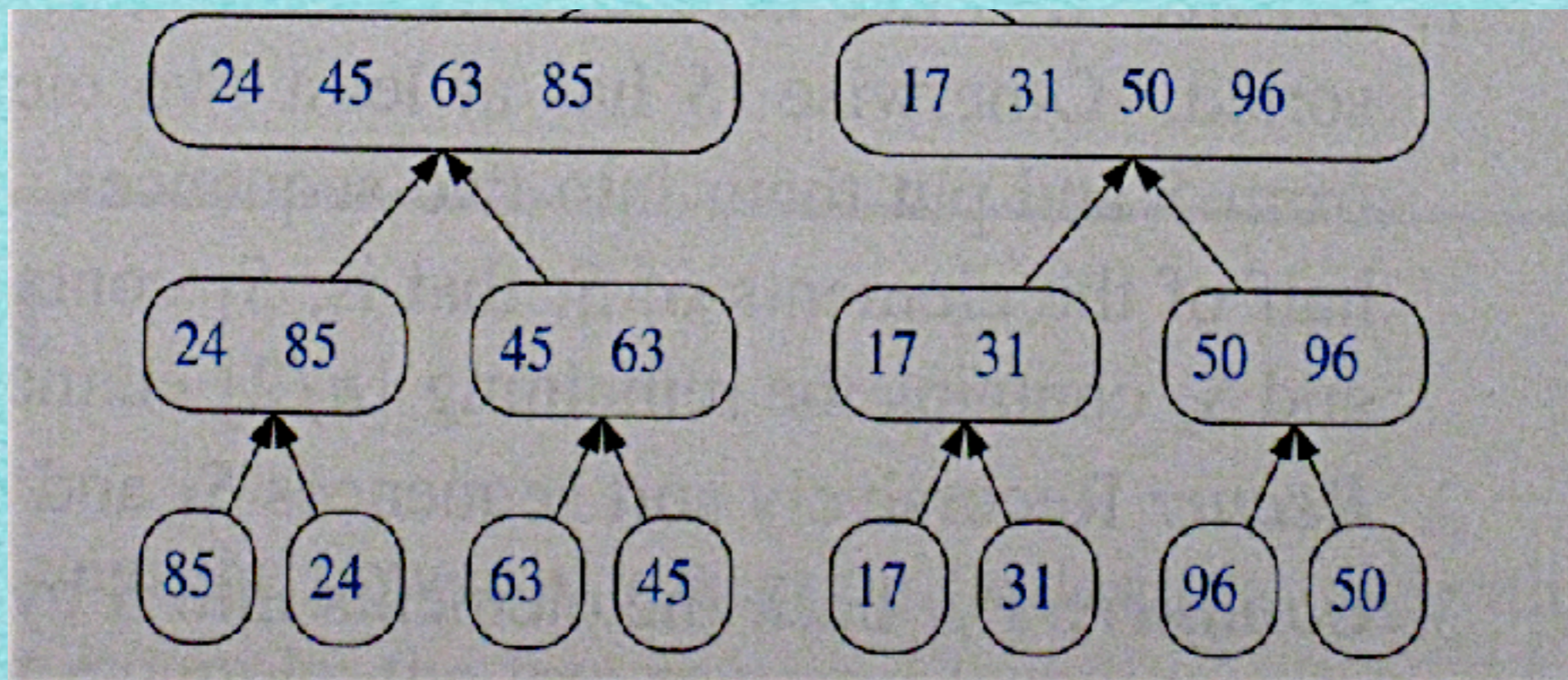
85 24 63 45 17 31 96 50

# 5.2 Mergesort

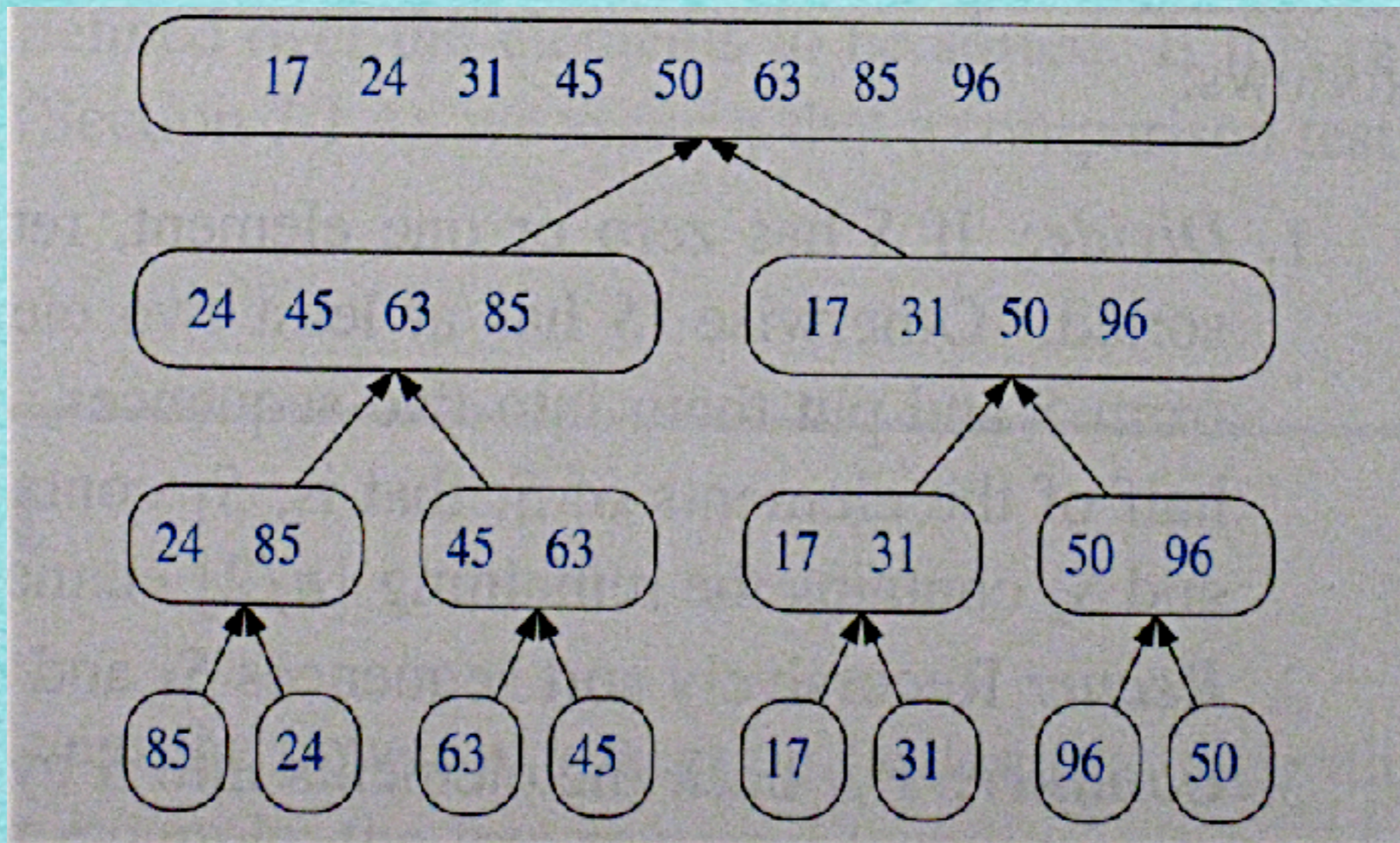




# 5.2 Mergesort



# 5.2 Mergesort



## 5.2.2 Algorithmische Beschreibung

### Algorithmus 5.1



## 5.2.2 Algorithmische Beschreibung

### Algorithmus 5.1

INPUT: Subarray von  $A=[1,\dots,n]$ ,

## 5.2.2 Algorithmische Beschreibung

### Algorithmus 5.1

INPUT: Subarray von  $A=[1,\dots,n]$ ,  
der bei Index  $p$  beginnt und bei Index  $r$  endet, d.h.  $A[p,\dots,r]$

## 5.2.2 Algorithmische Beschreibung

### Algorithmus 5.1

INPUT: Subarray von  $A=[1,\dots,n]$ ,  
der bei Index  $p$  beginnt und bei Index  $r$  endet, d.h.  $A[p,\dots,r]$

OUTPUT: Sortierter Subarray

## 5.2.2 Algorithmische Beschreibung

### Algorithmus 5.1

INPUT: Subarray von  $A=[1,\dots,n]$ ,  
der bei Index  $p$  beginnt und bei Index  $r$  endet, d.h.  $A[p,\dots,r]$

OUTPUT: Sortierter Subarray

MERGE-SORT( $A,p,r$ )

## 5.2.2 Algorithmische Beschreibung

### Algorithmus 5.1

INPUT: Subarray von  $A=[1,\dots,n]$ ,  
der bei Index  $p$  beginnt und bei Index  $r$  endet, d.h.  $A[p,\dots,r]$

OUTPUT: Sortierter Subarray

MERGE-SORT( $A,p,r$ )

```
1  if  $p < r$ 
2    then  $q \leftarrow \lfloor (p + r)/2 \rfloor$ 
3         MERGE-SORT( $A, p, q$ )
4         MERGE-SORT( $A, q + 1, r$ )
5         MERGE( $A, p, q, r$ )
```



## 5.2.2 Algorithmische Beschreibung

### Algorithmus 5.1

INPUT: Subarray von  $A=[1,\dots,n]$ ,  
der bei Index  $p$  beginnt und bei Index  $r$  endet, d.h.  $A[p,\dots,r]$

OUTPUT: Sortierter Subarray

MERGE-SORT( $A,p,r$ )

```
1  if  $p < r$ 
2    then  $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
3         MERGE-SORT( $A, p, q$ )
4         MERGE-SORT( $A, q + 1, r$ )
5         MERGE( $A, p, q, r$ )
```

## Subroutine 5.2



## Subroutine 5.2

INPUT: Zwei sortierte Subarrays von  $A=[1,\dots,n]$ , d.h.  $A[p,\dots,q]$  und  $A[q+1,\dots,r]$

OUTPUT: Sortierter Subarray  $A[p,\dots,r]$

## Subroutine 5.2

INPUT: Zwei sortierte Subarrays von  $A=[1,\dots,n]$ , d.h.  $A[p,\dots,q]$  und  $A[q+1,\dots,r]$

OUTPUT: Sortierter Subarray  $A[p,\dots,r]$

	8	9	10	11	12	13	14	15	16	17
A	...	2	4	5	7	1	2	3	6	...

## Subroutine 5.2

INPUT: Zwei sortierte Subarrays von  $A=[1,\dots,n]$ , d.h.  $A[p,\dots,q]$  und  $A[q+1,\dots,r]$

OUTPUT: Sortierter Subarray  $A[p,\dots,r]$

MERGE( $A, p, q, r$ )

	8	9	10	11	12	13	14	15	16	17
A	...	2	4	5	7	1	2	3	6	...

## Subroutine 5.2

INPUT: Zwei sortierte Subarrays von  $A=[1,\dots,n]$ , d.h.  $A[p,\dots,q]$  und  $A[q+1,\dots,r]$

OUTPUT: Sortierter Subarray  $A[p,\dots,r]$

MERGE( $A, p, q, r$ )

1  $n_1 \leftarrow q - p + 1$

2  $n_2 \leftarrow r - q$

	8	9	10	11	12	13	14	15	16	17
A	...	2	4	5	7	1	2	3	6	...

## Subroutine 5.2

INPUT: Zwei sortierte Subarrays von  $A=[1,\dots,n]$ , d.h.  $A[p,\dots,q]$  und  $A[q+1,\dots,r]$

OUTPUT: Sortierter Subarray  $A[p,\dots,r]$

MERGE( $A, p, q, r$ )

1  $n_1 \leftarrow q - p + 1$

2  $n_2 \leftarrow r - q$

	8	9	10	11	12	13	14	15	16	17
A	...	2	4	5	7	1	2	3	6	...

	8	9	10	11	12	13	14	15	16	17
A	...	2	4	5	7	1	2	3	6	...
					$k$					
L	1	2	3	4						
	2	4	5	7						
					$i$					
R						1	2	3	4	
						1	2	3	6	
										$j$

## Subroutine 5.2

INPUT: Zwei sortierte Subarrays von  $A=[1,\dots,n]$ , d.h.  $A[p,\dots,q]$  und  $A[q+1,\dots,r]$

OUTPUT: Sortierter Subarray  $A[p,\dots,r]$

MERGE( $A, p, q, r$ )

```
1  $n_1 \leftarrow q - p + 1$ 
2  $n_2 \leftarrow r - q$ 
3 erzeuge die Felder  $L[1..n_1 + 1]$  und  $R[1..n_2 + 1]$ 
4 for  $i \leftarrow 1$  to  $n_1$ 
5     do  $L[i] \leftarrow A[p + i - 1]$ 
6 for  $j \leftarrow 1$  to  $n_2$ 
7     do  $R[j] \leftarrow A[q + j]$ 
```

	8	9	10	11	12	13	14	15	16	17
A	...	2	4	5	7	1	2	3	6	...

	8	9	10	11	12	13	14	15	16	17
A	...	2	4	5	7	1	2	3	6	...
		k								
L		1	2	3	4					
		2	4	5	7					
		i								
						1	2	3	4	
						1	2	3	6	
						j				



## Subroutine 5.2

INPUT: Zwei sortierte Subarrays von  $A=[1,\dots,n]$ , d.h.  $A[p,\dots,q]$  und  $A[q+1,\dots,r]$

OUTPUT: Sortierter Subarray  $A[p,\dots,r]$

MERGE( $A, p, q, r$ )

```
1  $n_1 \leftarrow q - p + 1$ 
2  $n_2 \leftarrow r - q$ 
3 erzeuge die Felder  $L[1..n_1 + 1]$  und  $R[1..n_2 + 1]$ 
4 for  $i \leftarrow 1$  to  $n_1$ 
5     do  $L[i] \leftarrow A[p + i - 1]$ 
6 for  $j \leftarrow 1$  to  $n_2$ 
7     do  $R[j] \leftarrow A[q + j]$ 
8  $L[n_1 + 1] \leftarrow \infty$ 
9  $R[n_2 + 1] \leftarrow \infty$ 
```

	8	9	10	11	12	13	14	15	16	17
A	...	2	4	5	7	1	2	3	6	...

	8	9	10	11	12	13	14	15	16	17
A	...	2	4	5	7	1	2	3	6	...
		$k$								
L		1	2	3	4					
		2	4	5	7					
		$i$								
R						1	2	3	4	
						1	2	3	6	
						$j$				



## Subroutine 5.2

INPUT: Zwei sortierte Subarrays von  $A=[1,\dots,n]$ , d.h.  $A[p,\dots,q]$  und  $A[q+1,\dots,r]$

OUTPUT: Sortierter Subarray  $A[p,\dots,r]$

MERGE( $A, p, q, r$ )

```
1   $n_1 \leftarrow q - p + 1$ 
2   $n_2 \leftarrow r - q$ 
3  erzeuge die Felder  $L[1..n_1 + 1]$  und  $R[1..n_2 + 1]$ 
4  for  $i \leftarrow 1$  to  $n_1$ 
5      do  $L[i] \leftarrow A[p + i - 1]$ 
6  for  $j \leftarrow 1$  to  $n_2$ 
7      do  $R[j] \leftarrow A[q + j]$ 
8   $L[n_1 + 1] \leftarrow \infty$ 
9   $R[n_2 + 1] \leftarrow \infty$ 
10  $i \leftarrow 1$ 
11  $j \leftarrow 1$ 
```

	8	9	10	11	12	13	14	15	16	17
A	...	2	4	5	7	1	2	3	6	...

	8	9	10	11	12	13	14	15	16	17	
A	...	2	4	5	7	1	2	3	6	...	
		k									
L		1	2	3	4	5					
		2	4	5	7	$\infty$					
		i									
R			1	2	3	4	5				
			1	2	3	6	$\infty$				
			j								

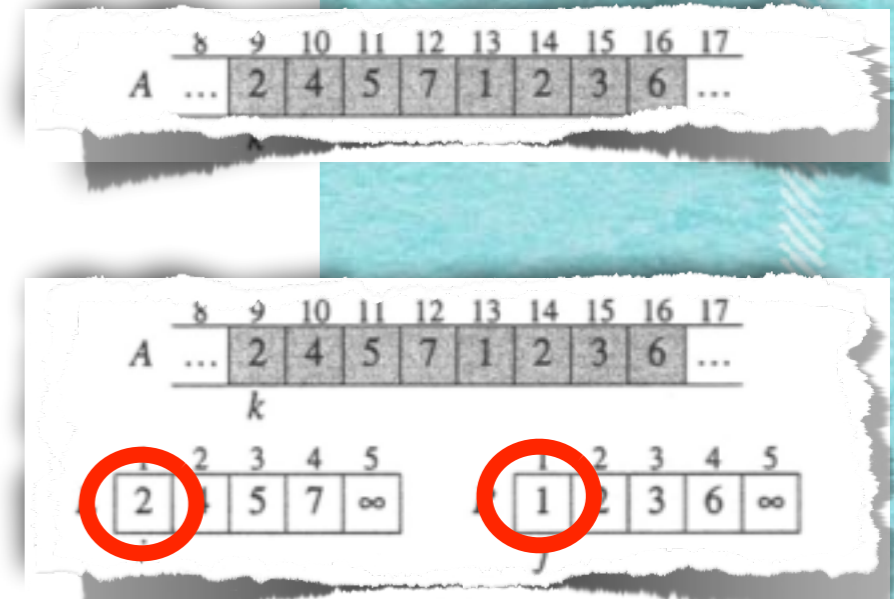
## Subroutine 5.2

INPUT: Zwei sortierte Subarrays von  $A=[1,\dots,n]$ , d.h.  $A[p,\dots,q]$  und  $A[q+1,\dots,r]$

OUTPUT: Sortierter Subarray  $A[p,\dots,r]$

MERGE( $A, p, q, r$ )

```
1  $n_1 \leftarrow q - p + 1$ 
2  $n_2 \leftarrow r - q$ 
3 erzeuge die Felder  $L[1..n_1 + 1]$  und  $R[1..n_2 + 1]$ 
4 for  $i \leftarrow 1$  to  $n_1$ 
5     do  $L[i] \leftarrow A[p + i - 1]$ 
6 for  $j \leftarrow 1$  to  $n_2$ 
7     do  $R[j] \leftarrow A[q + j]$ 
8  $L[n_1 + 1] \leftarrow \infty$ 
9  $R[n_2 + 1] \leftarrow \infty$ 
10  $i \leftarrow 1$ 
11  $j \leftarrow 1$ 
```



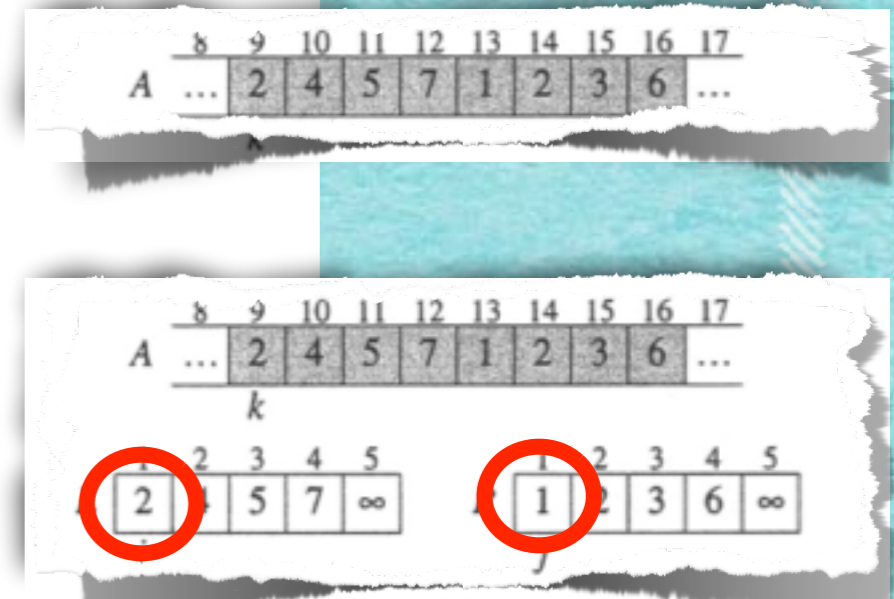
## Subroutine 5.2

INPUT: Zwei sortierte Subarrays von  $A=[1,\dots,n]$ , d.h.  $A[p,\dots,q]$  und  $A[q+1,\dots,r]$

OUTPUT: Sortierter Subarray  $A[p,\dots,r]$

MERGE( $A, p, q, r$ )

```
1   $n_1 \leftarrow q - p + 1$ 
2   $n_2 \leftarrow r - q$ 
3  erzeuge die Felder  $L[1..n_1 + 1]$  und  $R[1..n_2 + 1]$ 
4  for  $i \leftarrow 1$  to  $n_1$ 
5      do  $L[i] \leftarrow A[p + i - 1]$ 
6  for  $j \leftarrow 1$  to  $n_2$ 
7      do  $R[j] \leftarrow A[q + j]$ 
8   $L[n_1 + 1] \leftarrow \infty$ 
9   $R[n_2 + 1] \leftarrow \infty$ 
10  $i \leftarrow 1$ 
11  $j \leftarrow 1$ 
12 for  $k \leftarrow p$  to  $r$ 
13     do if  $L[i] \leq R[j]$ 
```



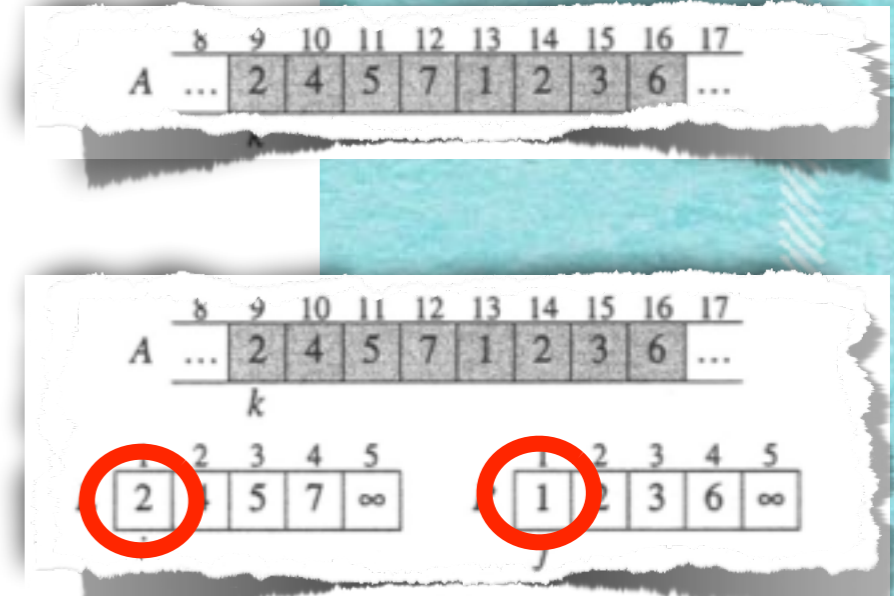
## Subroutine 5.2

INPUT: Zwei sortierte Subarrays von  $A=[1,\dots,n]$ , d.h.  $A[p,\dots,q]$  und  $A[q+1,\dots,r]$

OUTPUT: Sortierter Subarray  $A[p,\dots,r]$

MERGE( $A, p, q, r$ )

```
1   $n_1 \leftarrow q - p + 1$ 
2   $n_2 \leftarrow r - q$ 
3  erzeuge die Felder  $L[1..n_1 + 1]$  und  $R[1..n_2 + 1]$ 
4  for  $i \leftarrow 1$  to  $n_1$ 
5      do  $L[i] \leftarrow A[p + i - 1]$ 
6  for  $j \leftarrow 1$  to  $n_2$ 
7      do  $R[j] \leftarrow A[q + j]$ 
8   $L[n_1 + 1] \leftarrow \infty$ 
9   $R[n_2 + 1] \leftarrow \infty$ 
10  $i \leftarrow 1$ 
11  $j \leftarrow 1$ 
12 for  $k \leftarrow p$  to  $r$ 
13     do if  $L[i] \leq R[j]$ 
14         then  $A[k] \leftarrow L[i]$ 
15              $i \leftarrow i + 1$ 
```



## Subroutine 5.2

**INPUT:** Zwei sortierte Subarrays von  $A=[1,\dots,n]$ , d.h.  $A[p,\dots,q]$  und  $A[q+1,\dots,r]$

**OUTPUT:** Sortierter Subarray  $A[p,\dots,r]$

MERGE( $A, p, q, r$ )

```
1   $n_1 \leftarrow q - p + 1$ 
2   $n_2 \leftarrow r - q$ 
3  erzeuge die Felder  $L[1..n_1 + 1]$  und  $R[1..n_2 + 1]$ 
4  for  $i \leftarrow 1$  to  $n_1$ 
5      do  $L[i] \leftarrow A[p + i - 1]$ 
6  for  $j \leftarrow 1$  to  $n_2$ 
7      do  $R[j] \leftarrow A[q + j]$ 
8   $L[n_1 + 1] \leftarrow \infty$ 
9   $R[n_2 + 1] \leftarrow \infty$ 
10  $i \leftarrow 1$ 
11  $j \leftarrow 1$ 
12 for  $k \leftarrow p$  to  $r$ 
13     do if  $L[i] \leq R[j]$ 
14         then  $A[k] \leftarrow L[i]$ 
15              $i \leftarrow i + 1$ 
16     else  $A[k] \leftarrow R[j]$ 
17          $j \leftarrow j + 1$ 
```

	8	9	10	11	12	13	14	15	16	17
A	...	2	4	5	7	1	2	3	6	...

	8	9	10	11	12	13	14	15	16	17	
A	...	2	4	5	7	1	2	3	6	...	
		$k$									
		2	3	4	5		1	2	3	4	5
		2	5	7	$\infty$		1	2	3	6	$\infty$
							$j$				

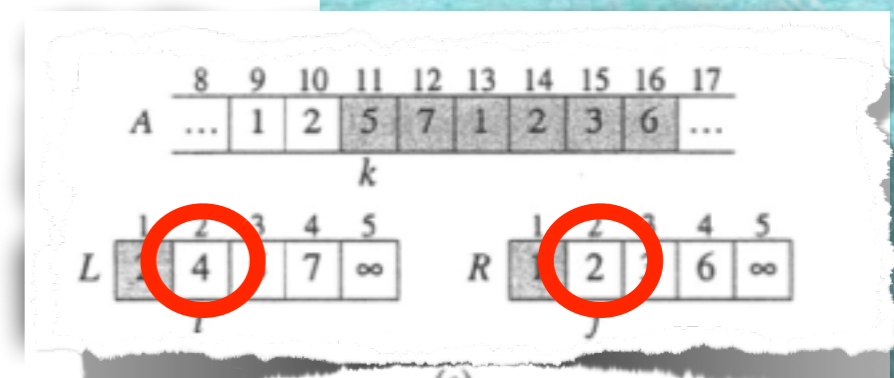
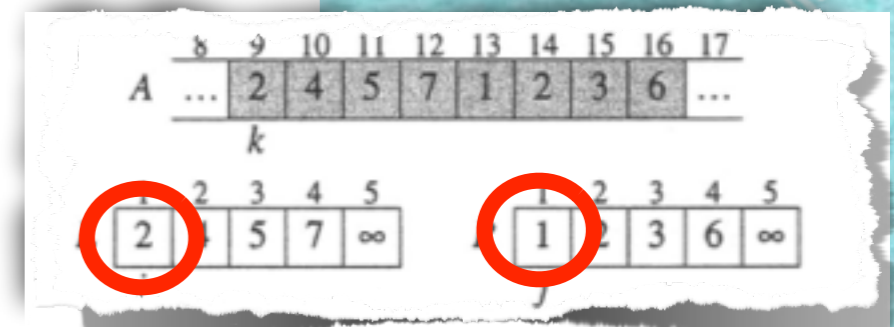
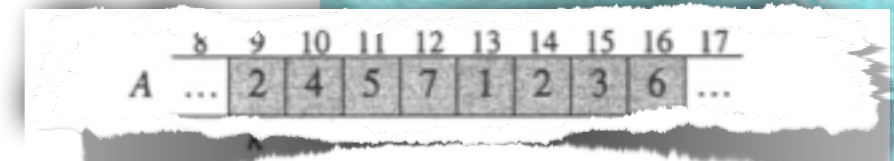
## Subroutine 5.2

INPUT: Zwei sortierte Subarrays von  $A=[1,\dots,n]$ , d.h.  $A[p,\dots,q]$  und  $A[q+1,\dots,r]$

OUTPUT: Sortierter Subarray  $A[p,\dots,r]$

MERGE( $A, p, q, r$ )

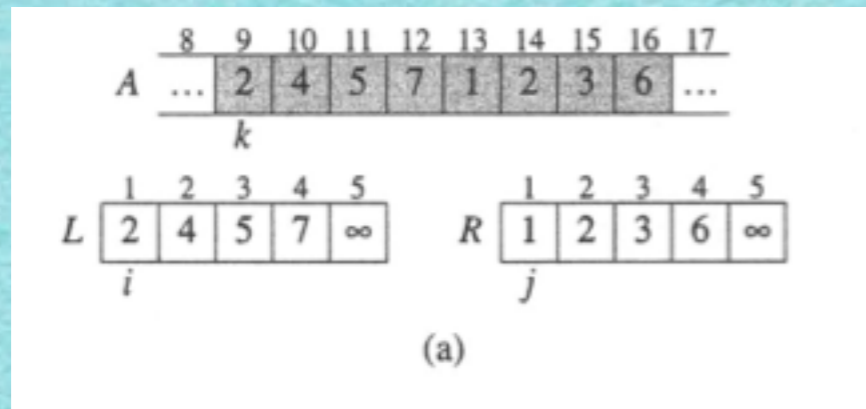
```
1   $n_1 \leftarrow q - p + 1$ 
2   $n_2 \leftarrow r - q$ 
3  erzeuge die Felder  $L[1..n_1 + 1]$  und  $R[1..n_2 + 1]$ 
4  for  $i \leftarrow 1$  to  $n_1$ 
5      do  $L[i] \leftarrow A[p + i - 1]$ 
6  for  $j \leftarrow 1$  to  $n_2$ 
7      do  $R[j] \leftarrow A[q + j]$ 
8   $L[n_1 + 1] \leftarrow \infty$ 
9   $R[n_2 + 1] \leftarrow \infty$ 
10  $i \leftarrow 1$ 
11  $j \leftarrow 1$ 
12 for  $k \leftarrow p$  to  $r$ 
13     do if  $L[i] \leq R[j]$ 
14         then  $A[k] \leftarrow L[i]$ 
15              $i \leftarrow i + 1$ 
16         else  $A[k] \leftarrow R[j]$ 
17              $j \leftarrow j + 1$ 
```



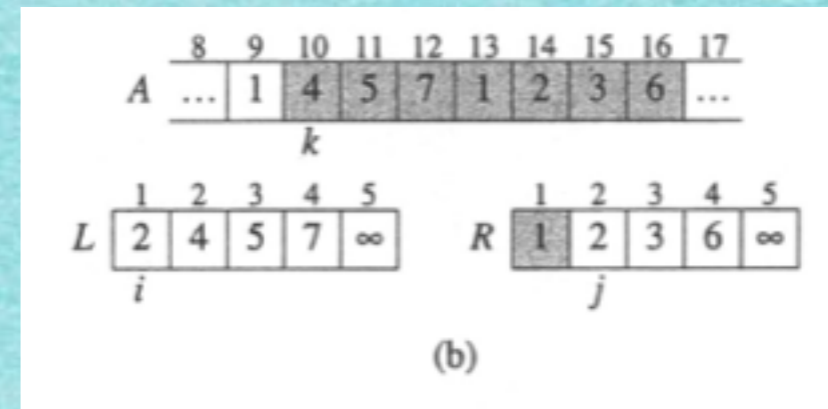
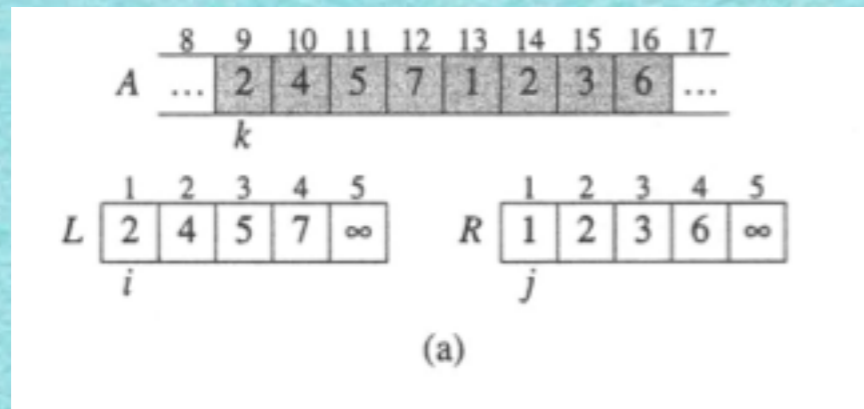


## 5.2 Mergesort

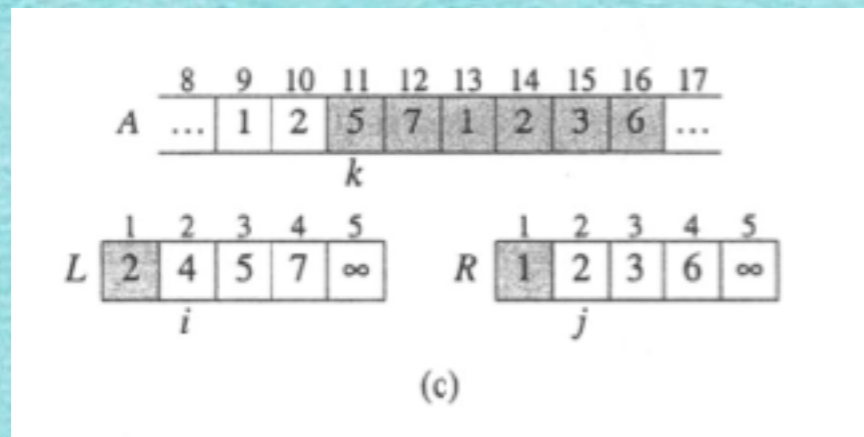
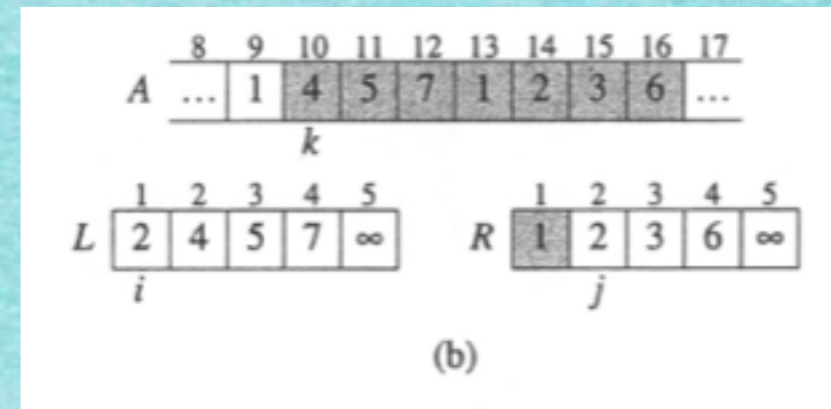
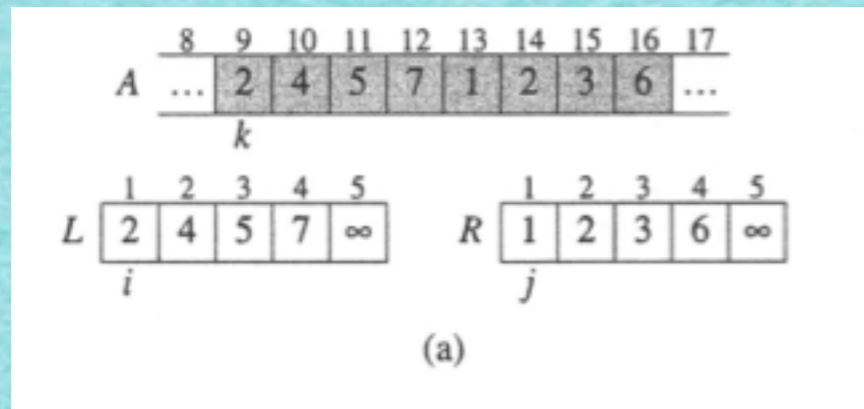
## 5.2 Mergesort



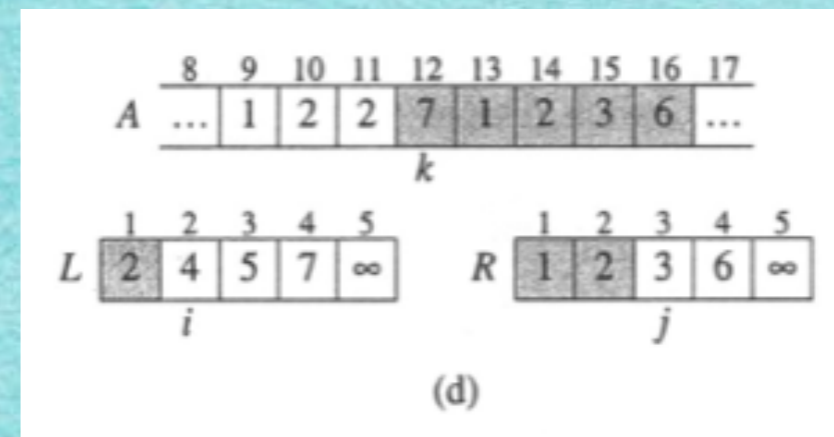
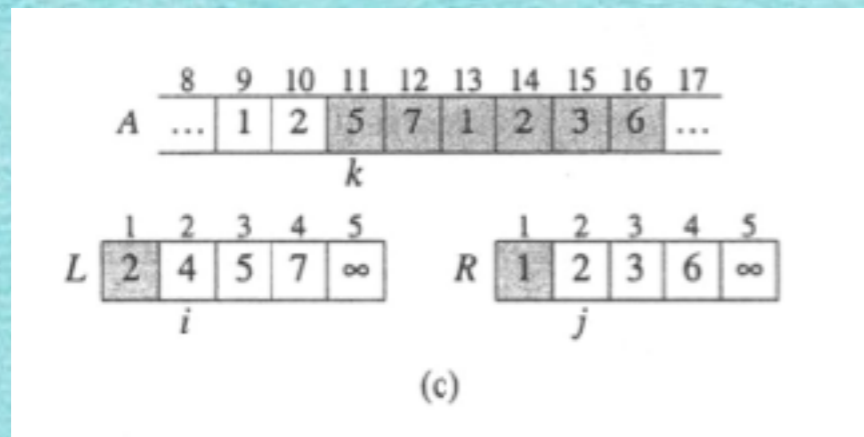
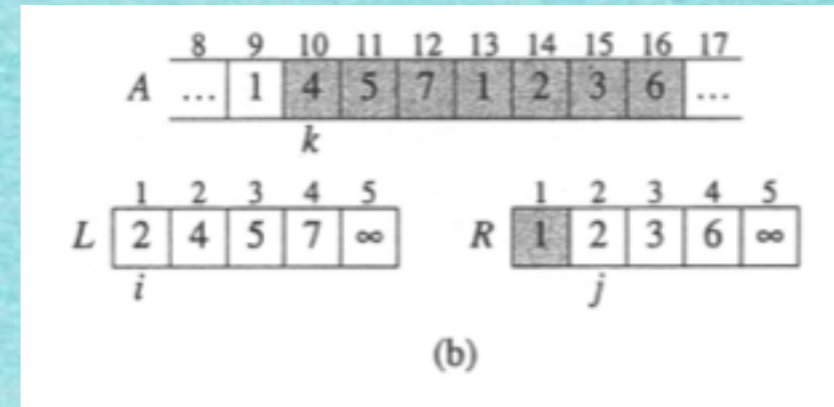
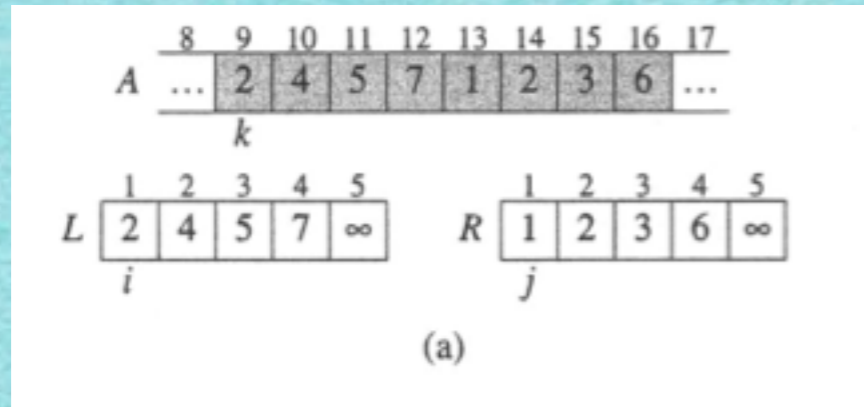
## 5.2 Mergesort



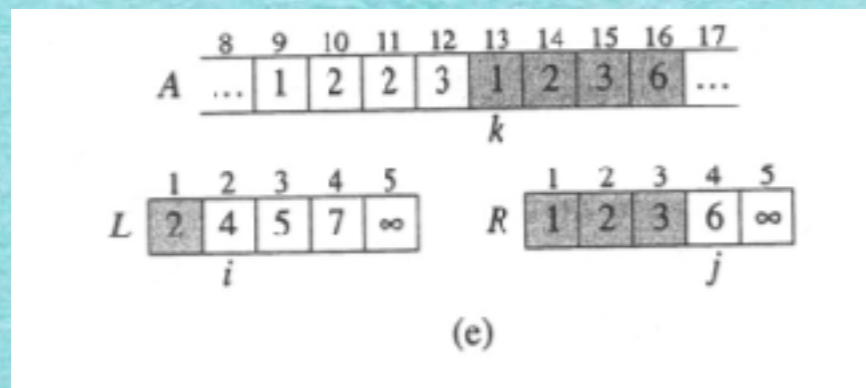
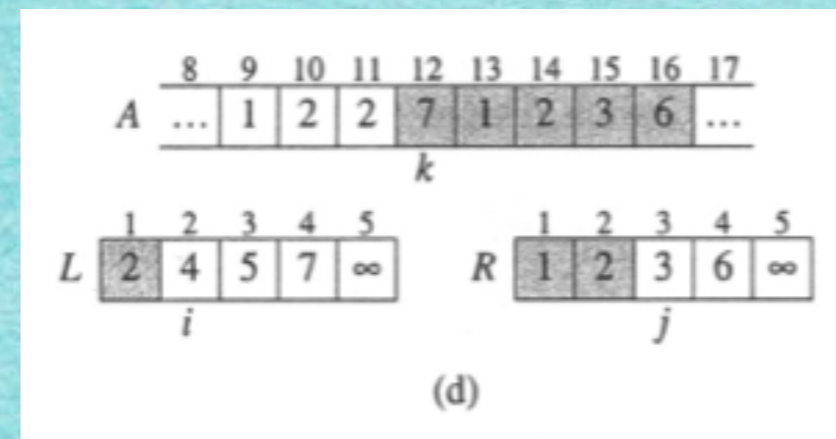
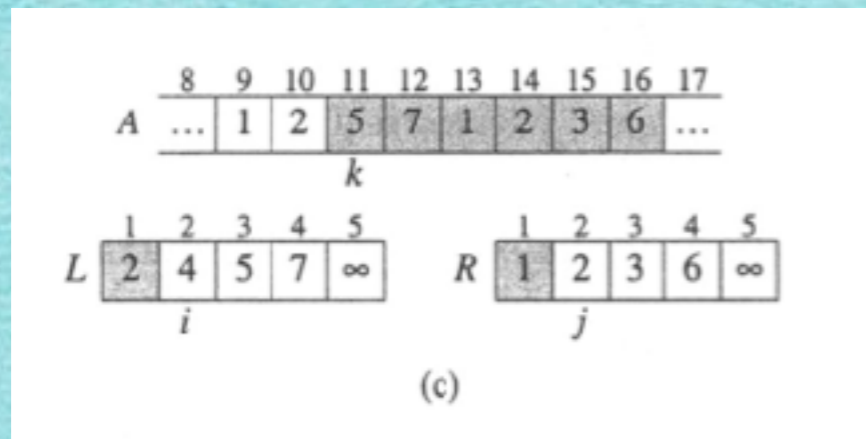
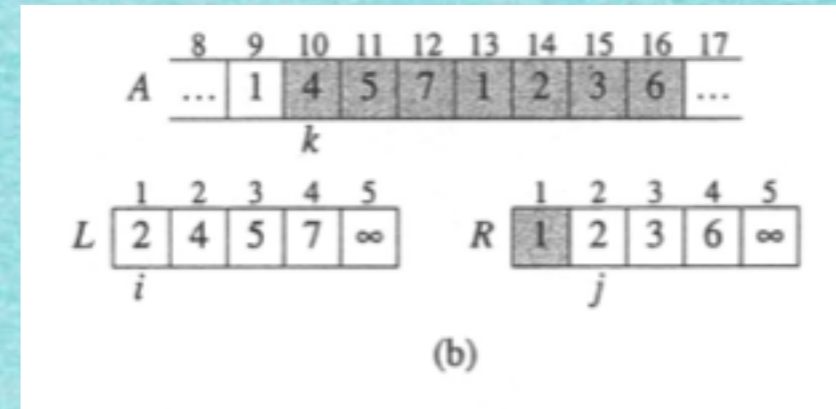
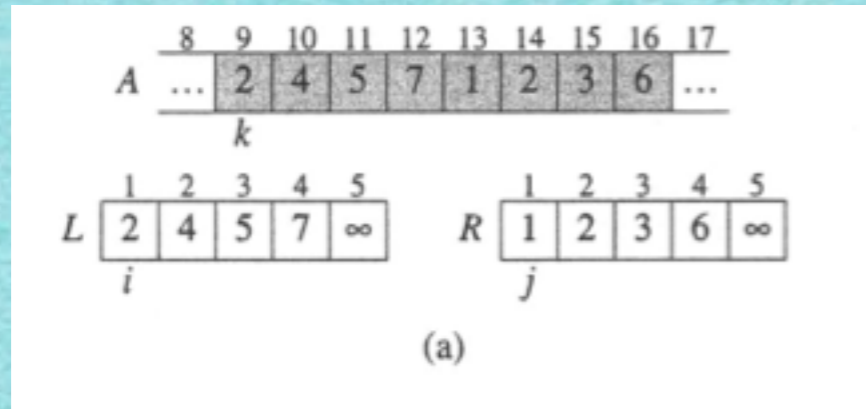
# 5.2 Mergesort



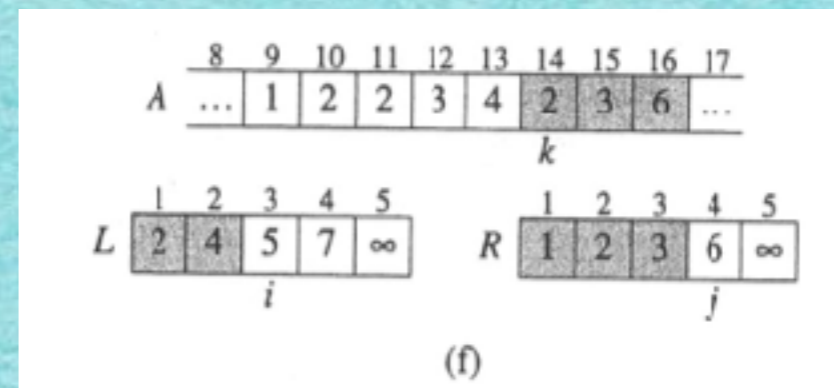
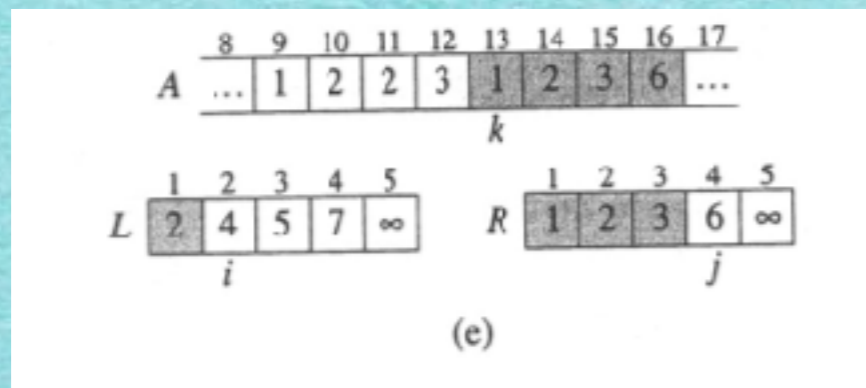
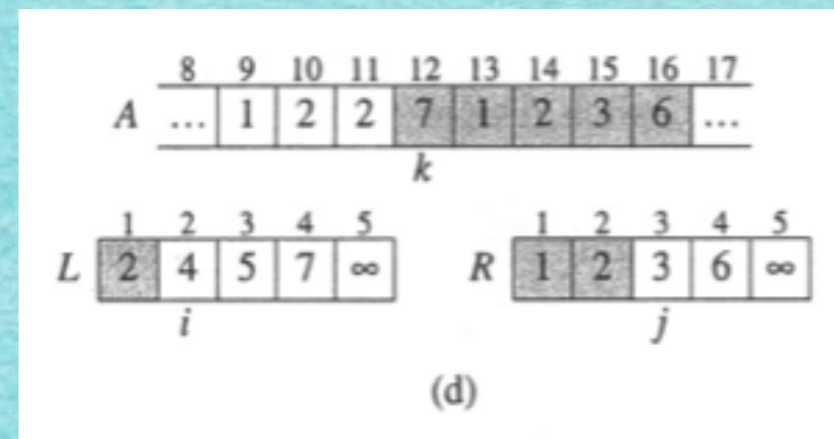
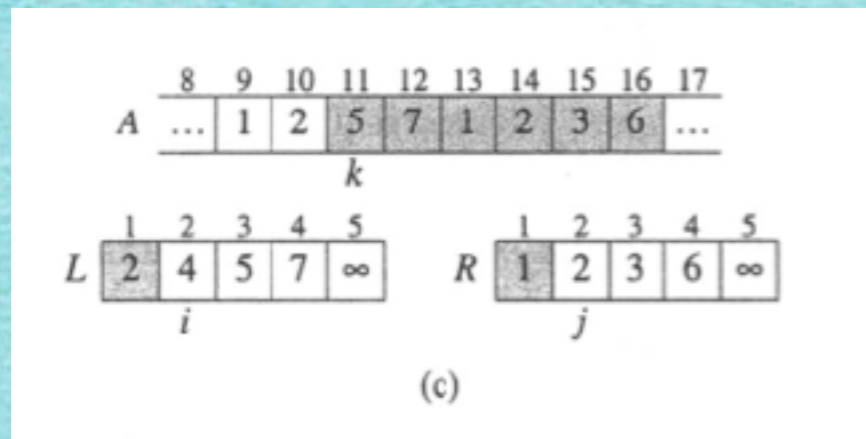
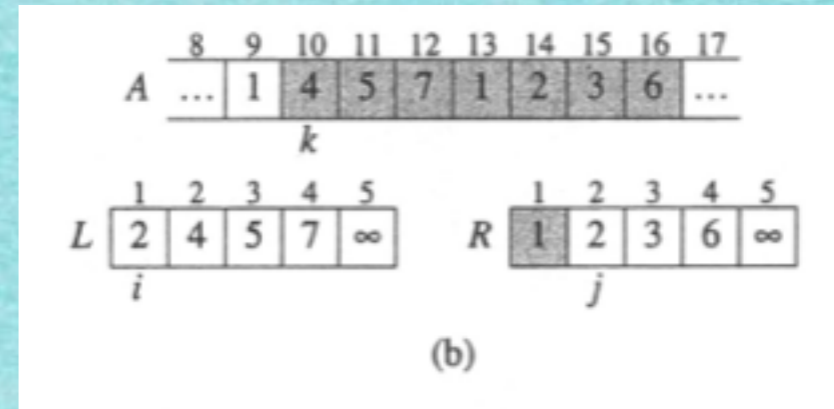
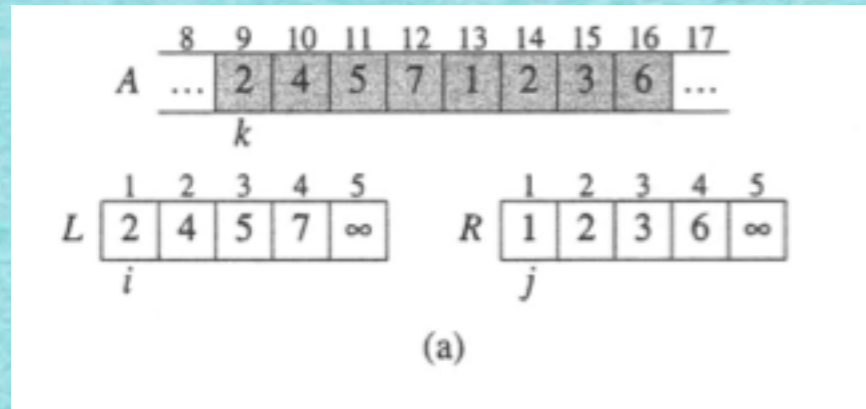
# 5.2 Mergesort



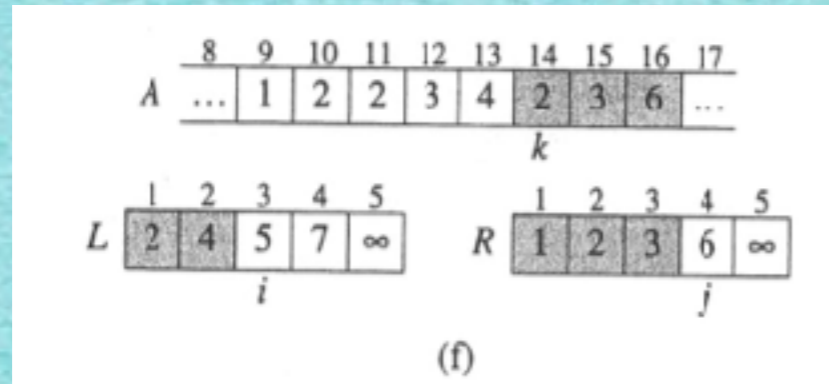
# 5.2 Mergesort



# 5.2 Mergesort

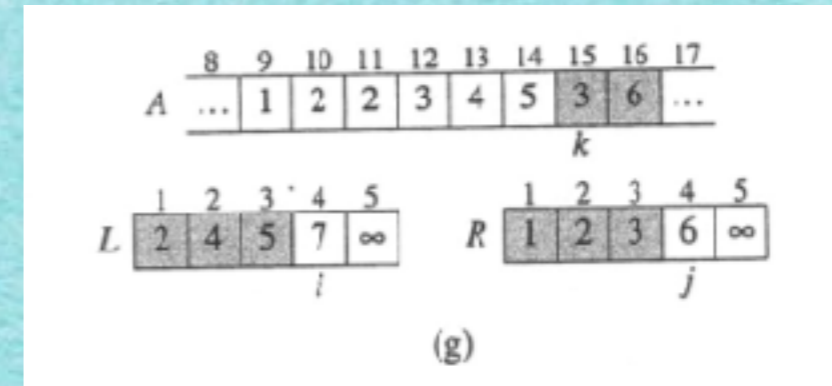
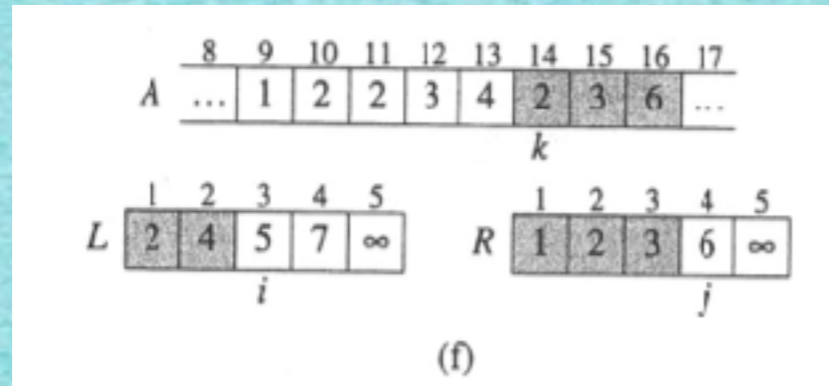


## 5.2 Mergesort

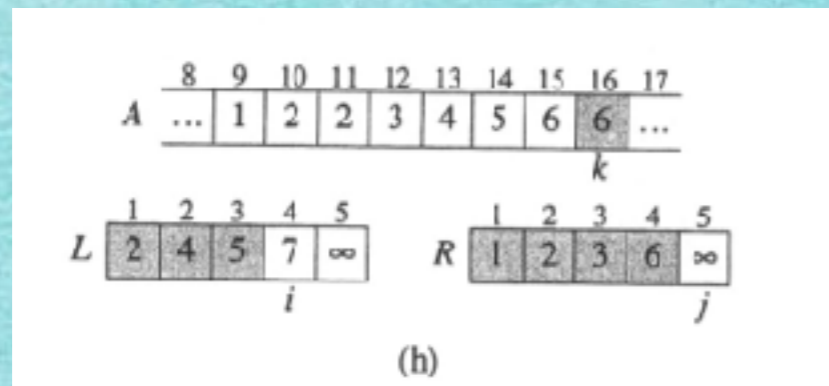
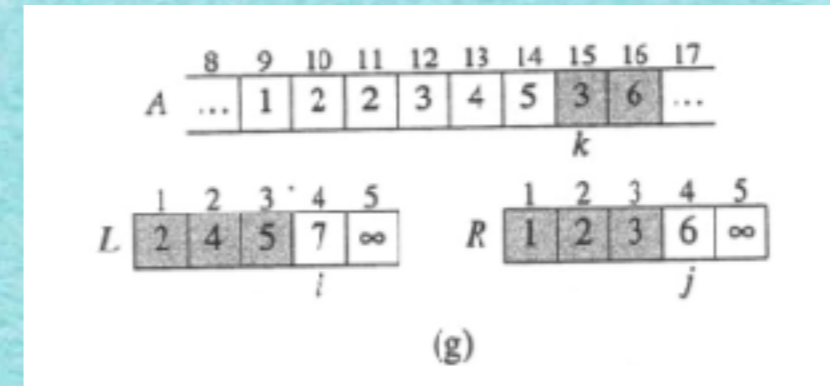
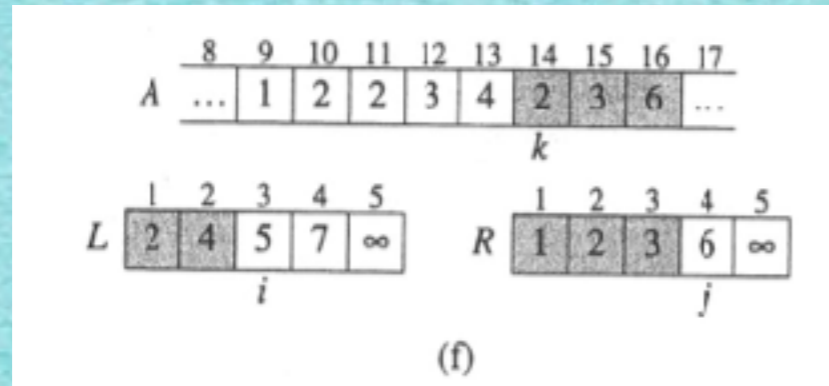




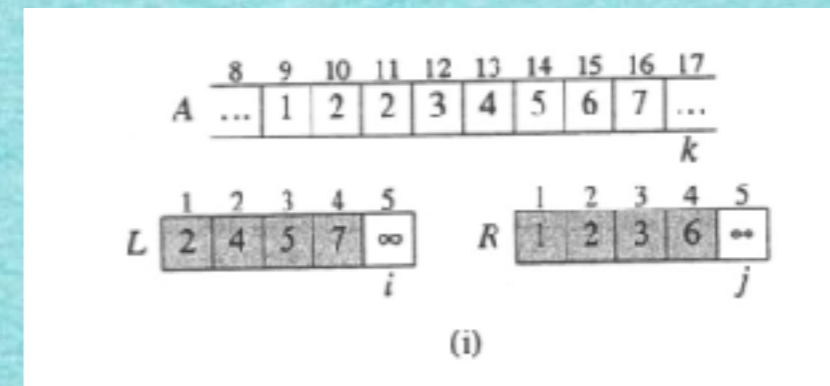
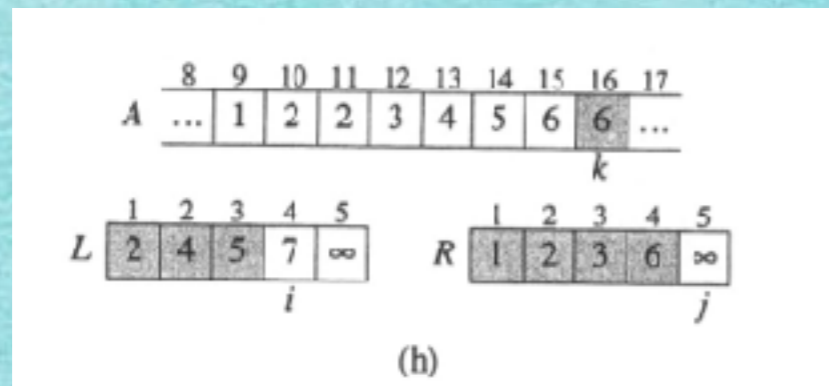
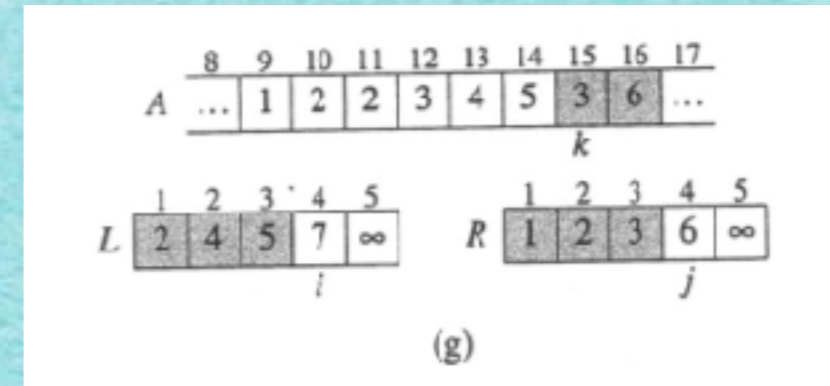
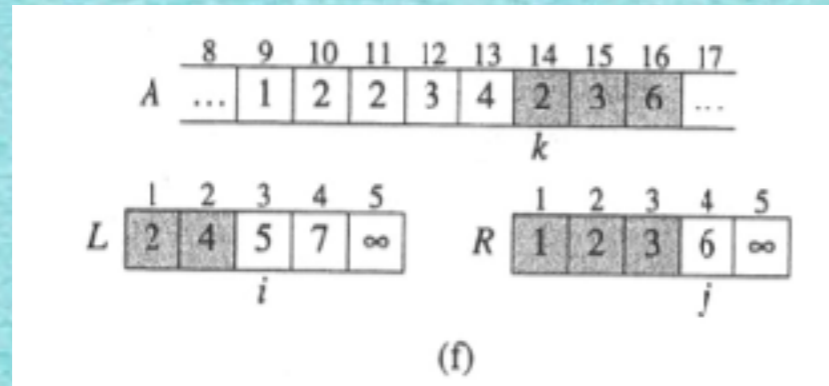
## 5.2 Mergesort



# 5.2 Mergesort



# 5.2 Mergesort



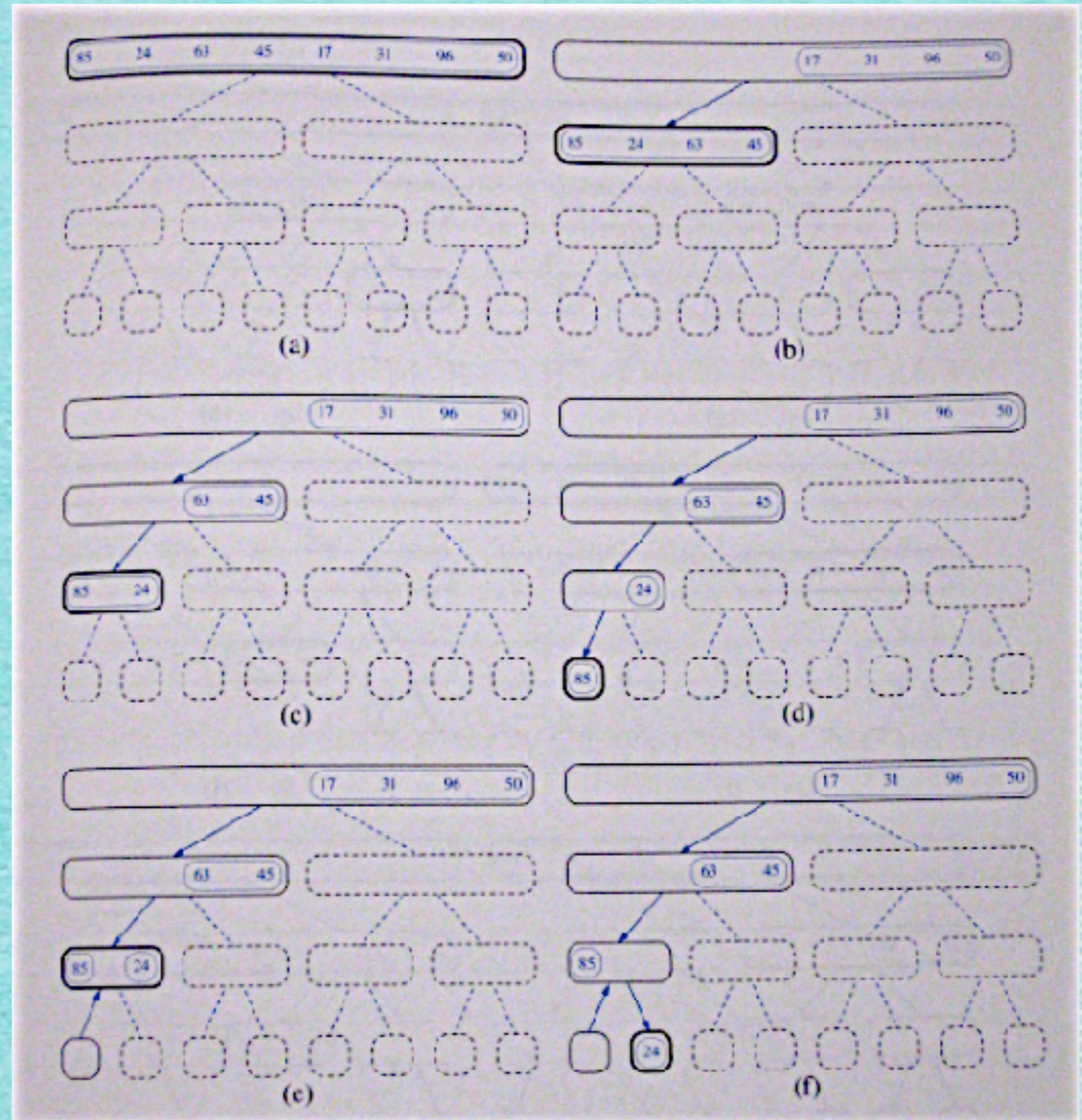
# Ablauf makroskopisch

# Ablauf makroskopisch

```
1  if  $p < r$ 
2    then  $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
3         MERGE-SORT( $A, p, q$ )
4         MERGE-SORT( $A, q + 1, r$ )
5         MERGE( $A, p, q, r$ )
```

# Ablauf makroskopisch

```
1  if  $p < r$   
2    then  $q \leftarrow \lfloor (p + r) / 2 \rfloor$   
3          MERGE-SORT( $A, p, q$ )  
4          MERGE-SORT( $A, q + 1, r$ )  
5          MERGE( $A, p, q, r$ )
```

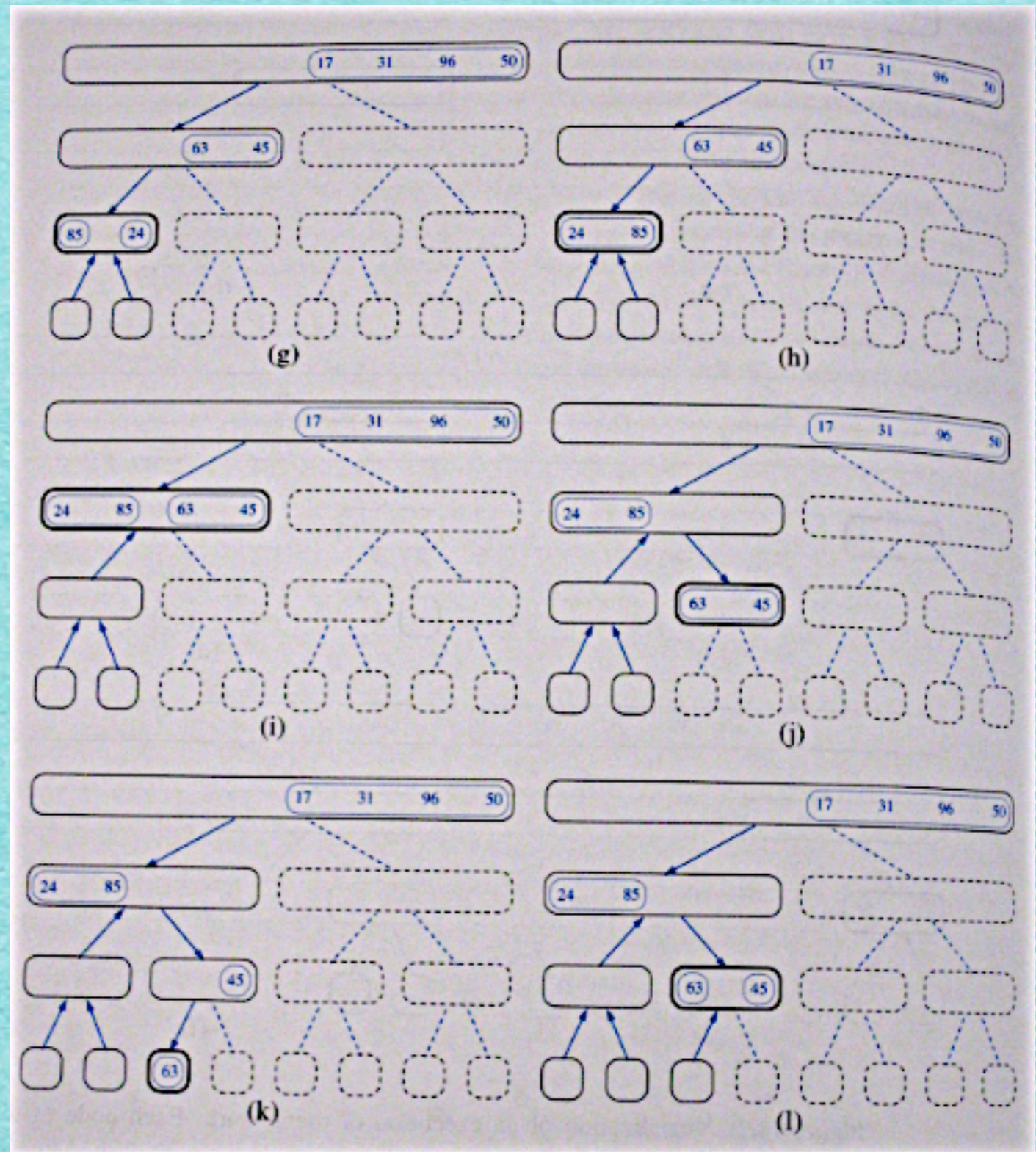


# Ablauf makroskopisch

```
1  if  $p < r$   
2    then  $q \leftarrow \lfloor (p + r) / 2 \rfloor$   
3        MERGE-SORT( $A, p, q$ )  
4        MERGE-SORT( $A, q + 1, r$ )  
5        MERGE( $A, p, q, r$ )
```

# Ablauf makroskopisch

```
1  if  $p < r$ 
2    then  $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
3         MERGE-SORT( $A, p, q$ )
4         MERGE-SORT( $A, q + 1, r$ )
5         MERGE( $A, p, q, r$ )
```





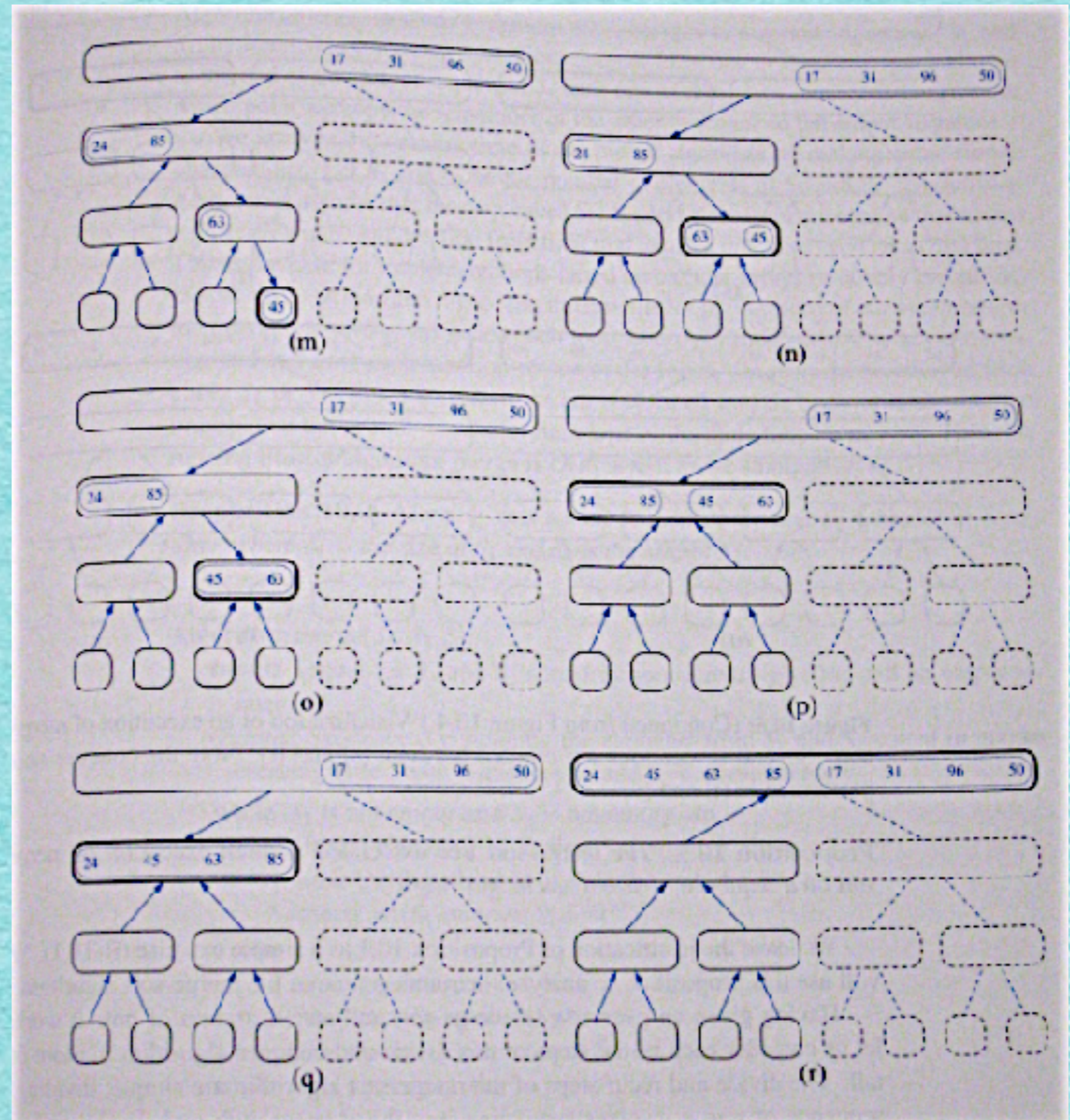
# Ablauf makroskopisch

```
1  if  $p < r$   
2    then  $q \leftarrow \lfloor (p + r) / 2 \rfloor$   
3        MERGE-SORT( $A, p, q$ )  
4        MERGE-SORT( $A, q + 1, r$ )  
5        MERGE( $A, p, q, r$ )
```

# Ablauf makroskopisch

```

1  if  $p < r$ 
2    then  $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
3         MERGE-SORT( $A, p, q$ )
4         MERGE-SORT( $A, q + 1, r$ )
5         MERGE( $A, p, q, r$ )
    
```

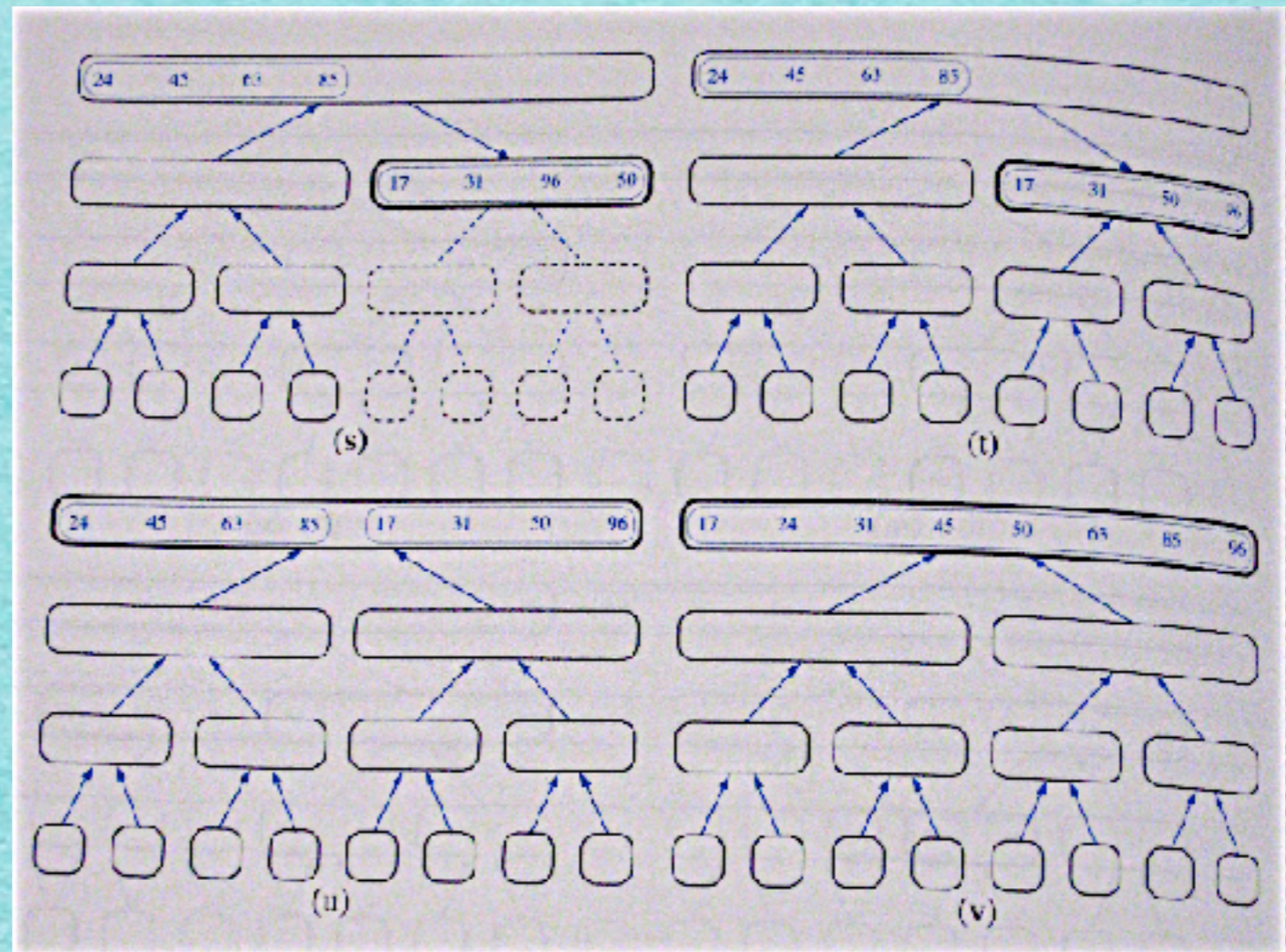


# Ablauf makroskopisch

```
1  if  $p < r$   
2    then  $q \leftarrow \lfloor (p + r) / 2 \rfloor$   
3        MERGE-SORT( $A, p, q$ )  
4        MERGE-SORT( $A, q + 1, r$ )  
5        MERGE( $A, p, q, r$ )
```

# Ablauf makroskopisch

```
1 if  $p < r$   
2   then  $q \leftarrow \lfloor (p + r) / 2 \rfloor$   
3         MERGE-SORT( $A, p, q$ )  
4         MERGE-SORT( $A, q + 1, r$ )  
5         MERGE( $A, p, q, r$ )
```



## 5.2 Mergesort

## 5.2 Mergesort



## 5.2 Mergesort

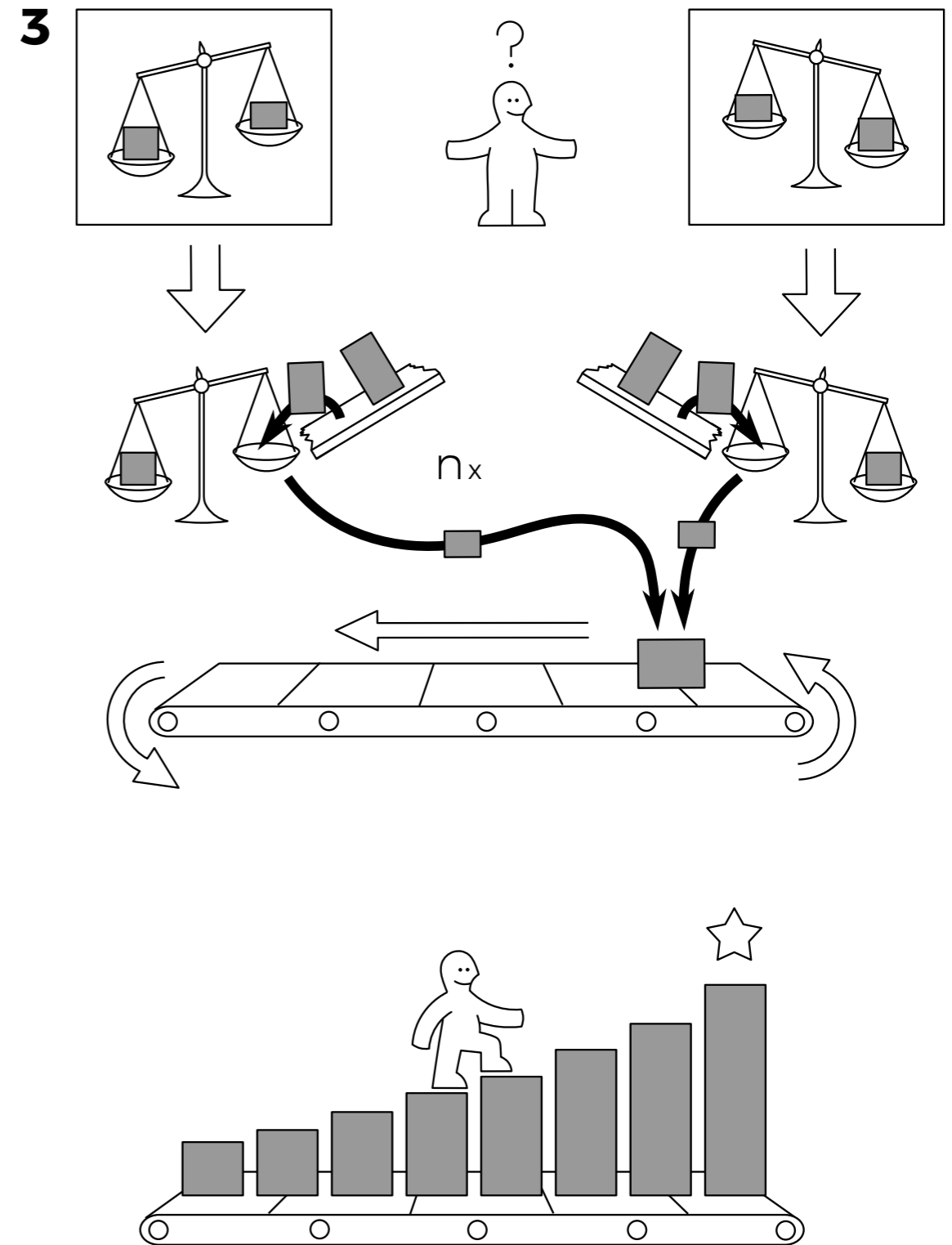
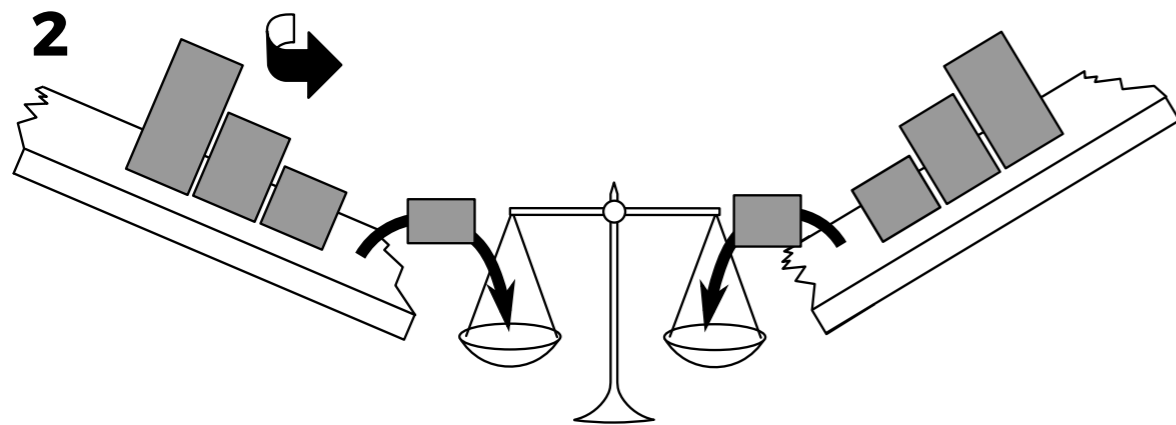
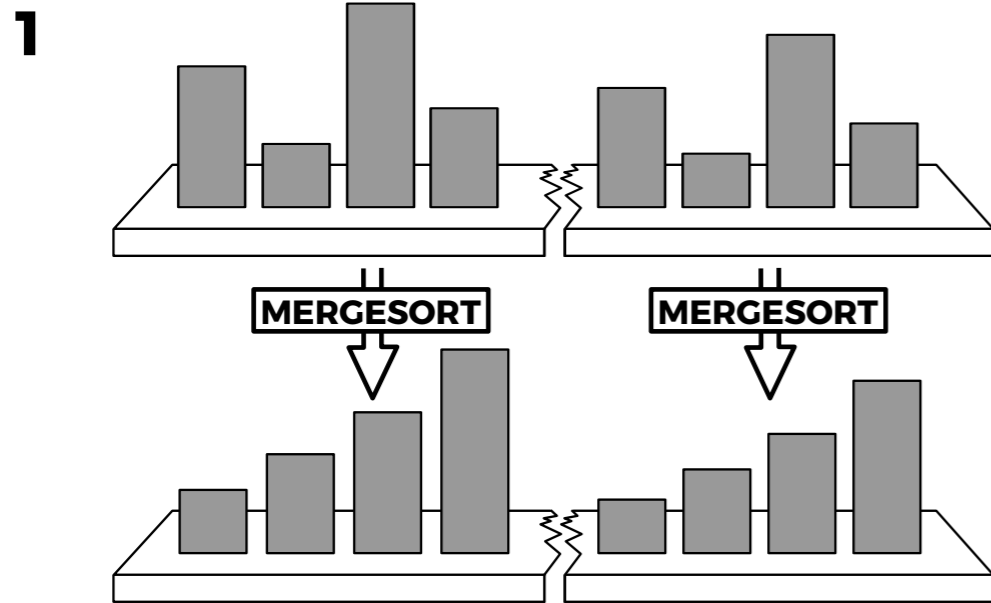
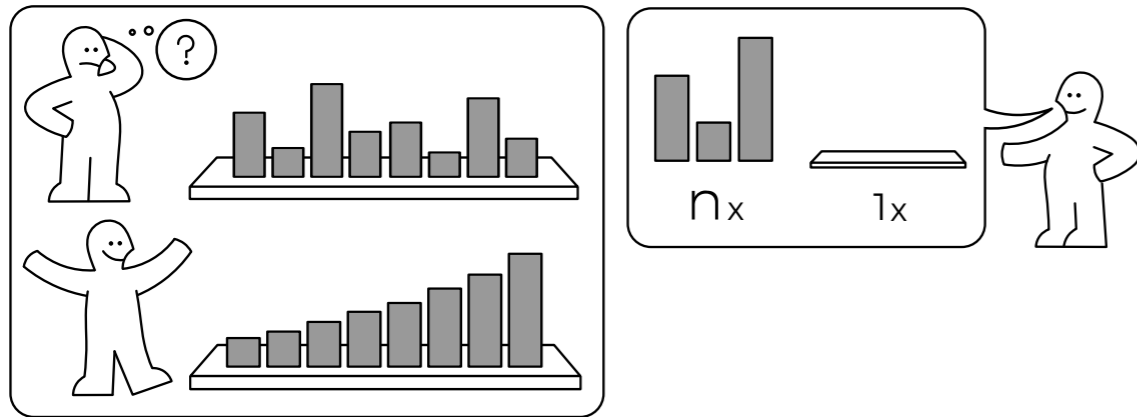


## 5.2 Mergesort





# MERGE SÖRT



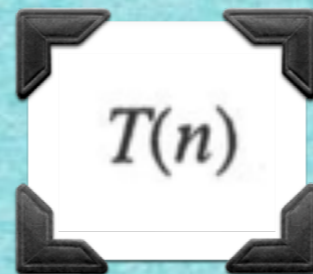
## 5.2.3 Laufzeit von Mergesort

## 5.2.3 Laufzeit von Mergesort

**Wie viele Schritte benötigt Merge-Sort für einen Array der Länge  $n$ ?**

## 5.2.3 Laufzeit von Mergesort

**Wie viele Schritte benötigt Merge-Sort für einen Array der Länge  $n$ ?**



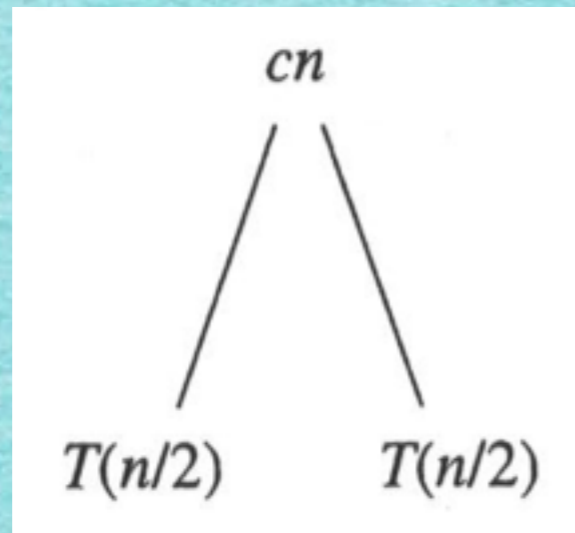
## 5.2.3 Laufzeit von Mergesort

## 5.2.3 Laufzeit von Mergesort

$T(n)$

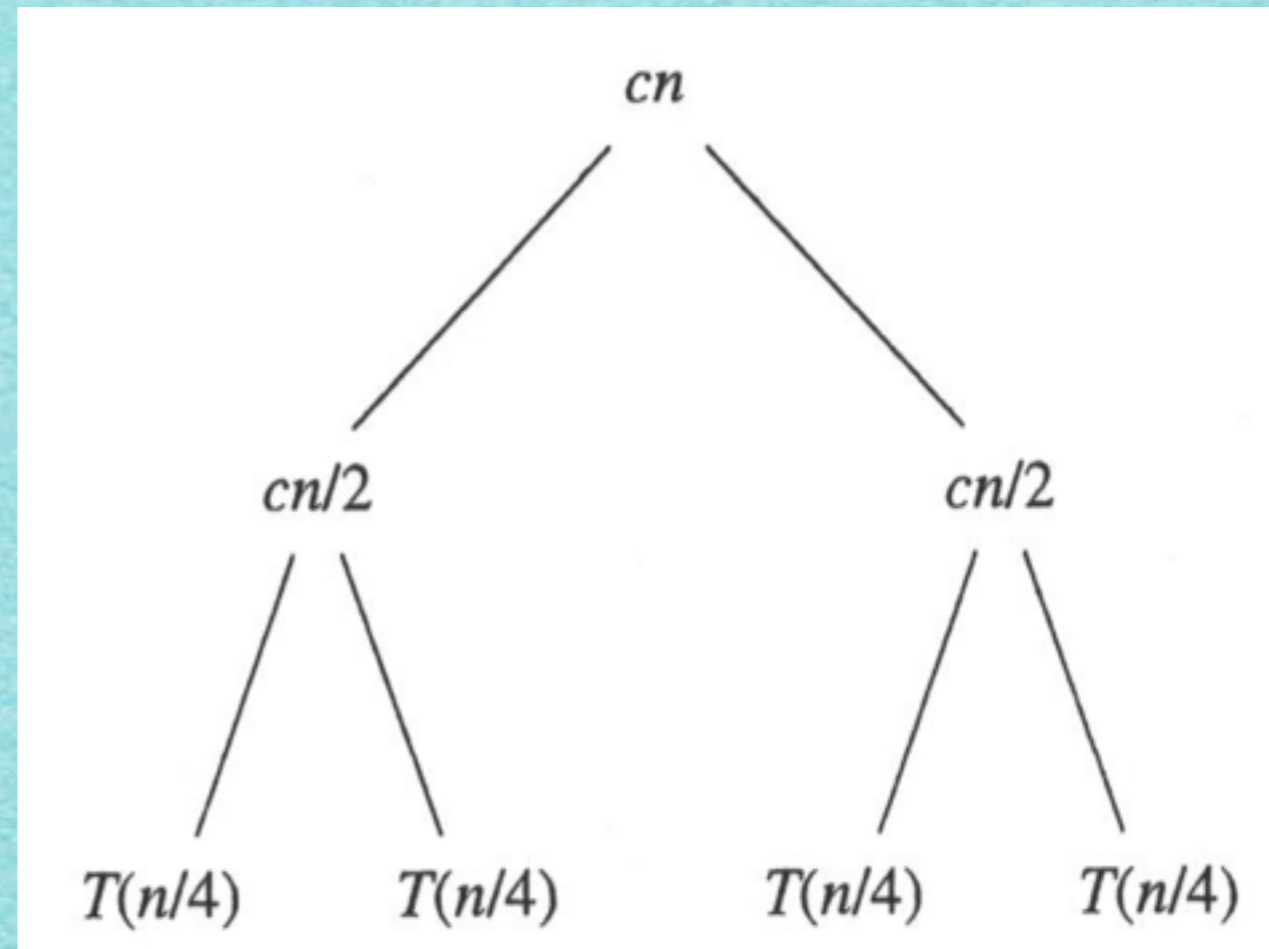
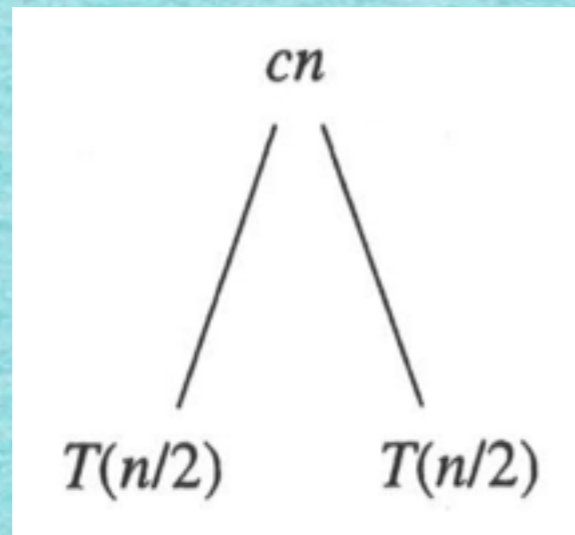
## 5.2.3 Laufzeit von Mergesort

$T(n)$



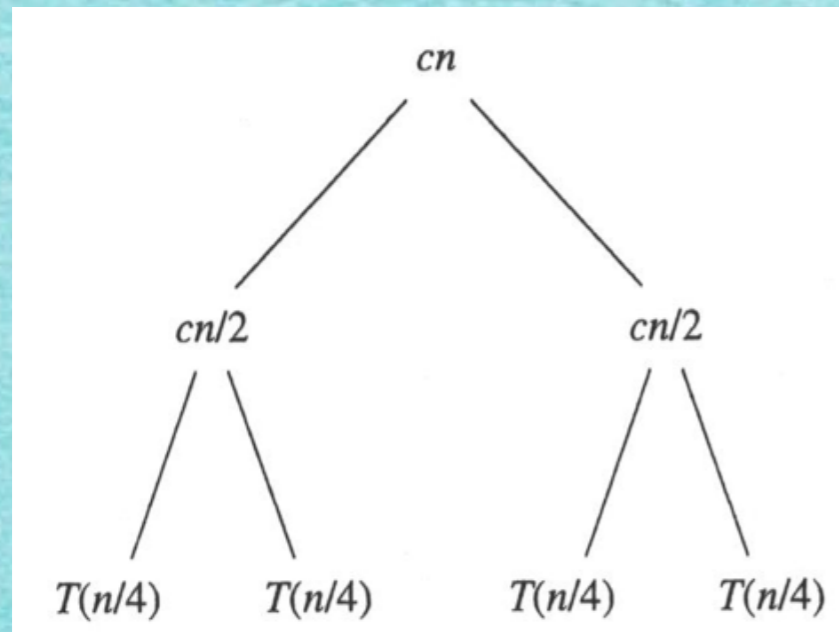
## 5.2.3 Laufzeit von Mergesort

$T(n)$

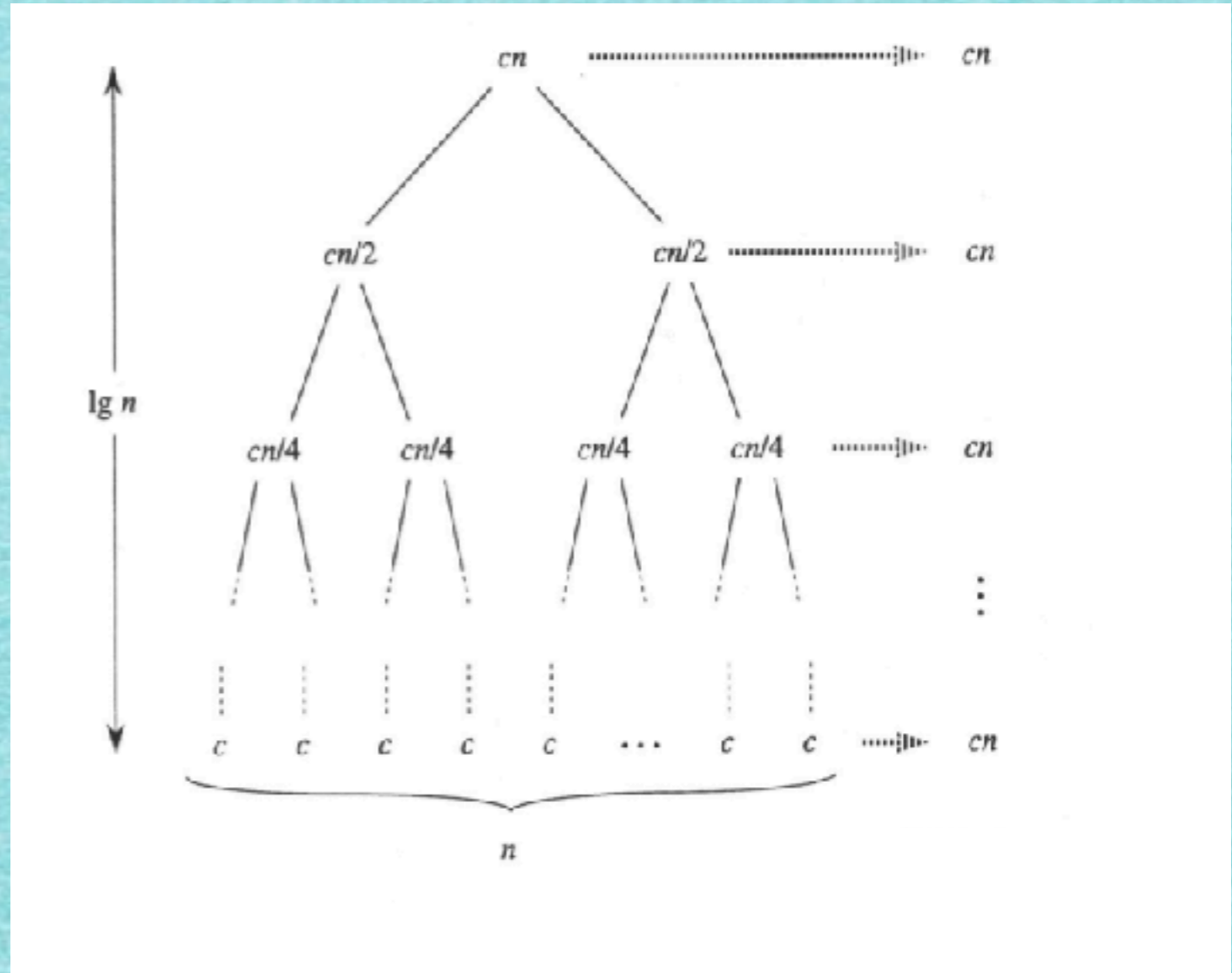
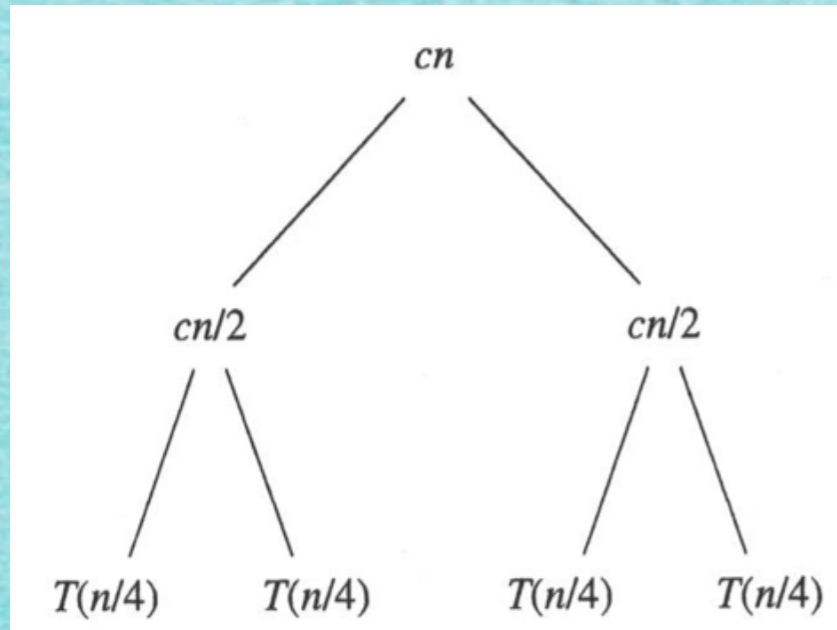




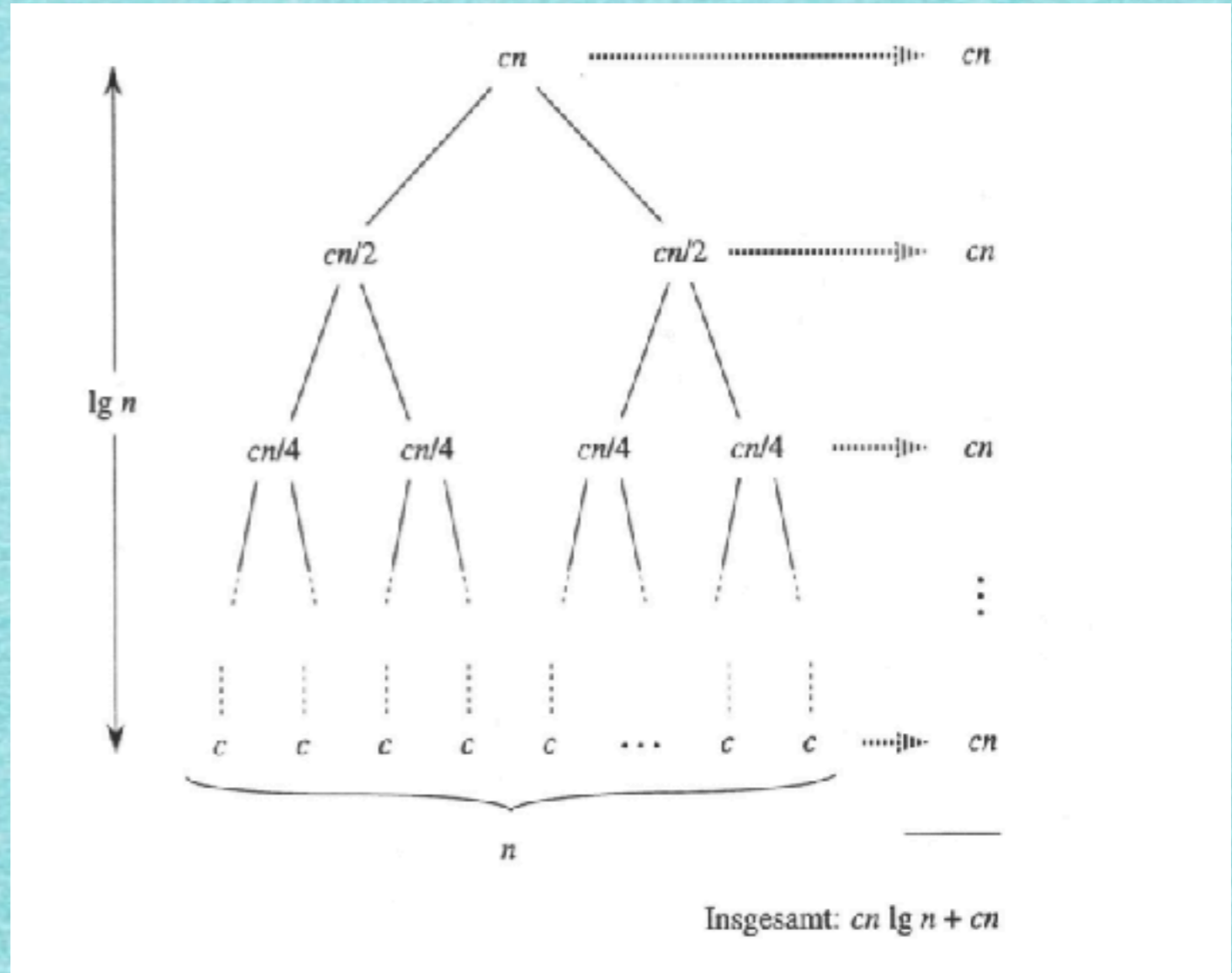
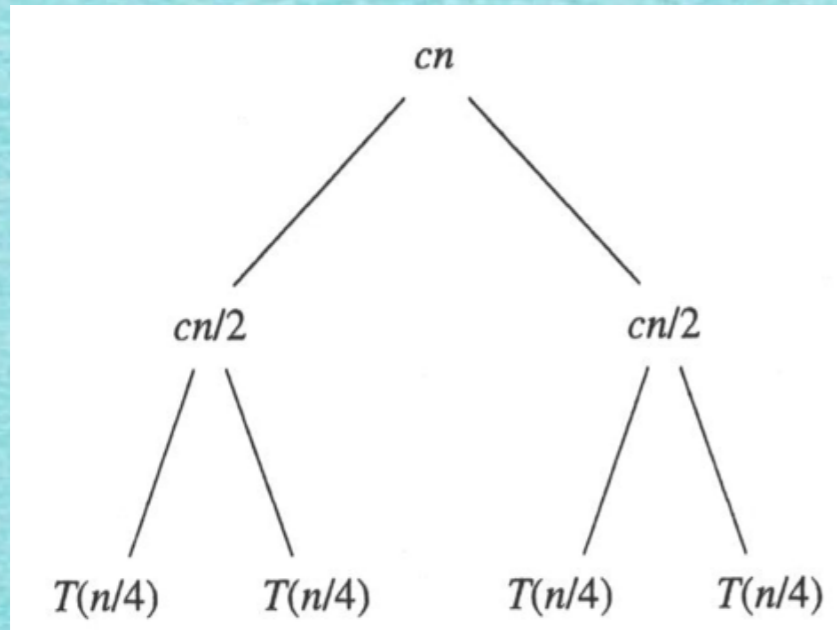
## 5.2.3 Laufzeit von Mergesort



## 5.2.3 Laufzeit von Mergesort



## 5.2.3 Laufzeit von Mergesort



## 5.2.3 Laufzeit von Mergesort

## 5.2.3 Laufzeit von Mergesort

### Satz 5.3 (Komplexität von Mergesort)

## 5.2.3 Laufzeit von Mergesort

**Satz 5.3 (Komplexität von Mergesort)**  
Für einen  $n$ -elementigen Array  $A$  hat Mergesort eine Laufzeit von  $O(n \log n)$ .

# *Mehr an der Tafel!*

*s.fekete@tu-bs.de*