

9. Sortieren in linearer Zeit

- Es gibt Algorithmen, die die $\Omega(n \log(n))$ untere Schranke für Vergleichssortierer schlagen.
- Diese Algorithmen benutzen neben Vergleichen und Kontrolloperationen noch andere Operationen wie z.B. arithmetische Operationen auf den zu sortierenden Zahlen.
- Allerdings benötigen diese Algorithmen gewisse Einschränkungen, um eine Laufzeit von $o(n \log n)$ zu erzielen.
- Zu diesen Algorithmen gehören **Counting-Sort**, **Bucket-Sort** und **Radix-Sort**.

Sortieren durch Abzählen (1)

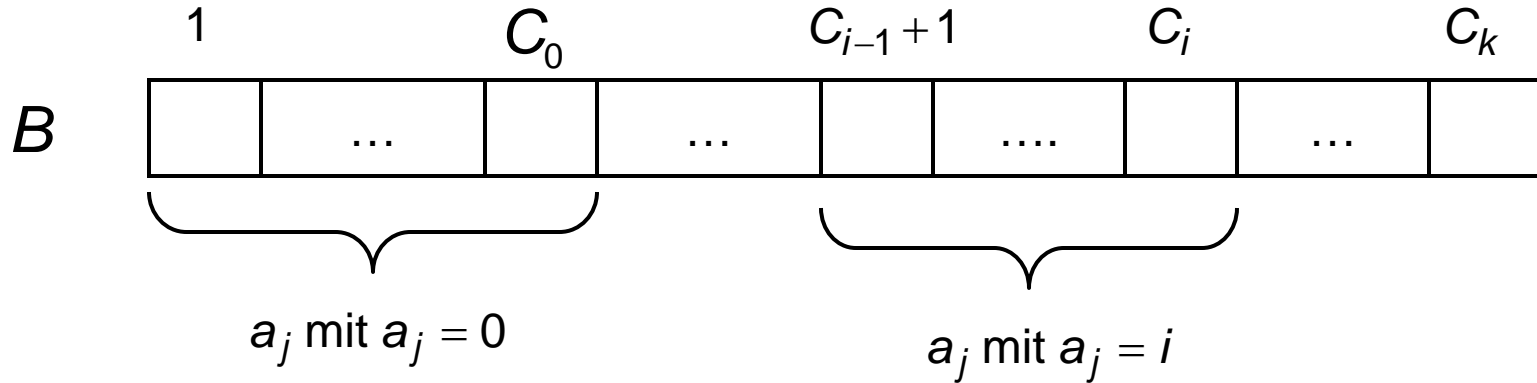
Annahme: Es gibt einen dem Algorithmus Counting-Sort bekannten Parameter k , so dass für die Eingabefolge (a_1, \dots, a_n) gilt:

$$0 \leq a_i \leq k \text{ für alle } 1 \leq i \leq n$$

Algorithmusidee:

1. Für alle j , $0 \leq j \leq k$, bestimme Anzahl C_j der a_i mit $a_i \leq j$. (D.h. $C_j - C_{j-1}$ ist die Anzahl der a_i mit $a_i = j$.)
2. Kopiere die a_i mit $a_i = j$ in Felder $B[C_{j-1} + 1], \dots, B[C_j]$ eines Arrays B mit $\text{length}(B) = n$. Dabei gilt $C_{-1} = 0$.

Sortieren durch Abzählen (2)



Counting-Sort

Counting-Sort(A,B,k)

1. for $i \leftarrow 0$ to k do
2. $C[i] \leftarrow 0$
3. for $j \leftarrow 1$ to $\text{length}(A)$ do
4. $C[A[j]] \leftarrow C[A[j]] + 1$
5. ▷ $C[i]$ enthält Anzahl der Elemente in A mit Wert i
6. for $i \leftarrow 1$ to k do
7. $C[i] \leftarrow C[i] + C[i-1]$
8. ▷ $C[i]$ enthält Anzahl der Elemente in A mit Wert $\leq i$
9. for $j \leftarrow \text{length}(A)$ downto 1 do
10. $B[C[A[j]]] \leftarrow A[j]$
11. $C[A[j]] \leftarrow C[A[j]] - 1$

Illustration für Counting-Sort (1)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

C

(nach Zeilen 3-4)

0	1	2	3	4	5
0	0	0	0	0	0

C

(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

	0	1	2	3	4	5
	0	0	0	0	0	0

C
(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

	0	1	2	3	4	5
	0	0	1	0	0	0

C
(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

C

(nach Zeilen 3-4)

0	1	2	3	4	5
0	0	1	0	0	0

C

(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

C

(nach Zeilen 3-4)

0	1	2	3	4	5
0	0	1	0	0	1

C

(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

C

(nach Zeilen 3-4)

0	1	2	3	4	5
0	0	1	0	0	1

C

(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

0	1	2	3	4	5
0	0	1	1	0	1

C
(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

0	1	2	3	4	5
0	0	1	1	0	1

C
(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

0	1	2	3	4	5
1	0	1	1	0	1

C
(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

0	1	2	3	4	5
1	0	1	1	0	1

C
(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

C

(nach Zeilen 3-4)

0	1	2	3	4	5
1	0	2	1	0	1

C

(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

	0	1	2	3	4	5
	1	0	2	1	0	1

C
(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

0	1	2	3	4	5
1	0	2	2	0	1

C
(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

0	1	2	3	4	5
1	0	2	2	0	1

C
(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

0	1	2	3	4	5
2	0	2	2	0	1

C
(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

0	1	2	3	4	5
2	0	2	2	0	1

C
(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

0	1	2	3	4	5
2	0	2	3	0	1

C
(nach Zeilen 6-7)

Illustration für Counting-Sort (1)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

0	1	2	3	4	5
2	0	2	3	0	1

C
(nach Zeilen 6-7)

0	1	2	3	4	5
2	0	2	3	0	1

Illustration für Counting-Sort (1)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

0	1	2	3	4	5
2	0	2	3	0	1

C
(nach Zeilen 6-7)

0	1	2	3	4	5
2	2	2	3	0	1

Illustration für Counting-Sort (1)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

0	1	2	3	4	5
2	0	2	3	0	1

C
(nach Zeilen 6-7)

0	1	2	3	4	5
2	2	4	3	0	1

Illustration für Counting-Sort (1)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

0	1	2	3	4	5
2	0	2	3	0	1

C
(nach Zeilen 6-7)

0	1	2	3	4	5
2	2	4	7	0	1

Illustration für Counting-Sort (1)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

0	1	2	3	4	5
2	0	2	3	0	1

C
(nach Zeilen 6-7)

0	1	2	3	4	5
2	2	4	7	7	1

Illustration für Counting-Sort (1)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

	0	1	2	3	4	5
	2	0	2	3	0	1

C
(nach Zeilen 6-7)

	0	1	2	3	4	5
	2	2	4	7	7	8

Illustration für Counting-Sort (1)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

C
(nach Zeilen 3-4)

	0	1	2	3	4	5
	2	0	2	3	0	1

C
(nach Zeilen 6-7)

	0	1	2	3	4	5
	2	2	4	7	7	8

Illustration für Counting-Sort (2)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

C

0	1	2	3	4	5
2	2	4	7	7	8

B

1	2	3	4	5	6	7	8

C

0	1	2	3	4	5
2	2	4	7	7	8

Illustration für Counting-Sort (2)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

C

0	1	2	3	4	5
2	2	4	7	7	8

B

1	2	3	4	5	6	7	8

C

0	1	2	3	4	5
2	2	4	7	7	8

Illustration für Counting-Sort (2)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

C

0	1	2	3	4	5
2	2	4	7	7	8

B

1	2	3	4	5	6	7	8
						3	

C

0	1	2	3	4	5
2	2	4	6	7	8

Illustration für Counting-Sort (2)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

C

0	1	2	3	4	5
2	2	4	6	7	8

B

1	2	3	4	5	6	7	8
						3	

C

0	1	2	3	4	5
2	2	4	6	7	8

Illustration für Counting-Sort (2)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

C

0	1	2	3	4	5
2	2	4	6	7	8

B

1	2	3	4	5	6	7	8
	0					3	

C

0	1	2	3	4	5
1	2	4	6	7	8

Illustration für Counting-Sort (2)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

B

1	2	3	4	5	6	7	8
	0					3	

C

0	1	2	3	4	5
1	2	4	6	7	8

Illustration für Counting-Sort (2)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

B

1	2	3	4	5	6	7	8
	0				3	3	

C

0	1	2	3	4	5
1	2	4	5	7	8

Illustration für Counting-Sort (2)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

B

1	2	3	4	5	6	7	8
	0				3	3	

C

0	1	2	3	4	5
1	2	4	5	7	8

Illustration für Counting-Sort (2)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

B

1	2	3	4	5	6	7	8
	0		2		3	3	

C

0	1	2	3	4	5
1	2	3	5	7	8

Illustration für Counting-Sort (2)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

B

1	2	3	4	5	6	7	8
	0		2		3	3	

C

0	1	2	3	4	5
1	2	3	5	7	8

Illustration für Counting-Sort (2)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

B

1	2	3	4	5	6	7	8
0	0		2		3	3	

C

0	1	2	3	4	5
0	2	3	5	7	8

Illustration für Counting-Sort (2)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

B

1	2	3	4	5	6	7	8
0	0		2		3	3	

C

0	1	2	3	4	5
0	2	3	5	7	8

Illustration für Counting-Sort (2)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

B

1	2	3	4	5	6	7	8
0	0		2	3	3	3	

C

0	1	2	3	4	5
0	2	3	4	7	8

Illustration für Counting-Sort (2)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

B

	1	2	3	4	5	6	7	8
	0	0		2	3	3	3	

C

	0	1	2	3	4	5
	0	2	3	4	7	8

Illustration für Counting-Sort (2)

A

	1	2	3	4	5	6	7	8
	2	5	3	0	2	3	0	3

B

	1	2	3	4	5	6	7	8
	0	0		2	3	3	3	5

C

	0	1	2	3	4	5
	0	2	3	4	7	7

Illustration für Counting-Sort (2)

	1	2	3	4	5	6	7	8
<i>A</i>	2	5	3	0	2	3	0	3

	1	2	3	4	5	6	7	8
<i>B</i>	0	0		2	3	3	3	5

	0	1	2	3	4	5
<i>C</i>	0	2	3	4	7	7

Illustration für Counting-Sort (2)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

B

1	2	3	4	5	6	7	8
0	0	2	2	3	3	3	5

C

0	1	2	3	4	5
0	2	2	4	7	7

Illustration für Counting-Sort (2)

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

B

1	2	3	4	5	6	7	8
0	0	2	2	3	3	3	5

C

0	1	2	3	4	5
0	2	2	4	7	7

Laufzeit von Counting-Sort

Counting-Sort(A,B,k)

1. for $i \leftarrow 0$ to k do
2. $C[i] \leftarrow 0$
3. for $j \leftarrow 1$ to $\text{length}(A)$ do
4. $C[A[j]] \leftarrow C[A[j]] + 1$
5. ▷ $C[i]$ enthält Anzahl der Elemente in A mit Wert i
6. for $i \leftarrow 1$ to k do
7. $C[i] \leftarrow C[i] + C[i-1]$
8. ▷ $C[i]$ enthält Anzahl der Elemente in A mit Wert $\leq i$
9. for $j \leftarrow \text{length}(A)$ downto 1 do
10. $B[C[A[j]]] \leftarrow A[j]$
11. $C[A[j]] \leftarrow C[A[j]] - 1$

➤ Zeilen 1-2, 6-7: jeweils $\mathbf{O}(k)$

➤ Zeilen 3-4, 9-11: jeweils $\mathbf{O}(n)$

Laufzeit von Counting-Sort

Satz 9.1: Counting-Sort besitzt Laufzeit $\mathbf{O}(n+k)$.

Korollar 9.2: Gilt $k = \mathbf{O}(n)$, so besitzt Counting-Sort Laufzeit $\mathbf{O}(n)$.

Korrektheit von Counting-Sort

Counting-Sort(A,B,k)

1. for $i \leftarrow 0$ to k do
2. $C[i] \leftarrow 0$
 $I_1(j): \forall i \in \{0, \dots, k\}: C[i] = |\{ 1 \leq l \leq j-1 \mid A[l]=i \}|$
3. for $j \leftarrow 1$ to $\text{length}(A)$ do
4. $C[A[j]] \leftarrow C[A[j]] + 1$
5. ▷ $C[i]$ enthält Anzahl der Elemente in A mit Wert i
 $I_2(i): \forall j \in \{0, \dots, i-1\}: C[j] = \# \leq j \text{ in } A \text{ und } \forall j \in \{i, \dots, k\}: C[j] = \# = j \text{ in } A$
6. for $i \leftarrow 1$ to k do
7. $C[i] \leftarrow C[i] + C[i-1]$
8. ▷ $C[i]$ enthält Anzahl der Elemente in A mit Wert $\leq i$
 $I_3(j): \forall i \in \{j+1, \dots, \text{length}(A)\}: \text{Pos. von } A[i] \text{ in } B \text{ ist gleich}$
 $|\{ 1 \leq l \leq \text{length}(A) \mid A[l] < A[i] \}| + |\{ 1 \leq l < i \mid A[l] = A[i] \}| + 1$ und
 $\forall i \in \{0, \dots, k\}: C[i] = |\{ 1 \leq l \leq \text{length}(A) \mid A[l] < i \}| + |\{ 1 \leq l \leq j \mid A[l] = i \}|$
9. for $j \leftarrow \text{length}(A)$ downto 1 do
10. $B[C[A[j]]] \leftarrow A[j]$
11. $C[A[j]] \leftarrow C[A[j]] - 1$

Korrektheit von Counting-Sort

Counting-Sort(A,B,k)

1. for $i \leftarrow 0$ to k do
2. $C[i] \leftarrow 0$
3. for $j \leftarrow 1$ to $\text{length}(A)$ do
4. $C[A[j]] \leftarrow C[A[j]] + 1$
5. ▷ $C[i]$ enthält Anzahl der Elemente in A mit Wert i
6. for $i \leftarrow 1$ to k do
7. $C[i] \leftarrow C[i] + C[i-1]$
8. ▷ $C[i]$ enthält Anzahl der Elemente in A mit Wert $\leq i$
9. for $j \leftarrow \text{length}(A)$ downto 1 do
10. $B[C[A[j]]] \leftarrow A[j]$
11. $C[A[j]] \leftarrow C[A[j]] - 1$

Am Ende gilt: $\forall i \in \{1, \dots, \text{length}(A)\}$: Pos. von $A[i]$ in B ist gleich

$|\{ 1 \leq l \leq \text{length}(A) \mid A[l] < A[i] \}| + |\{ 1 \leq l < i \mid A[l] = A[i] \}| + 1$

d.h. die Folge ist sortiert **bei Beibehaltung der Reihenfolge gleicher El.**

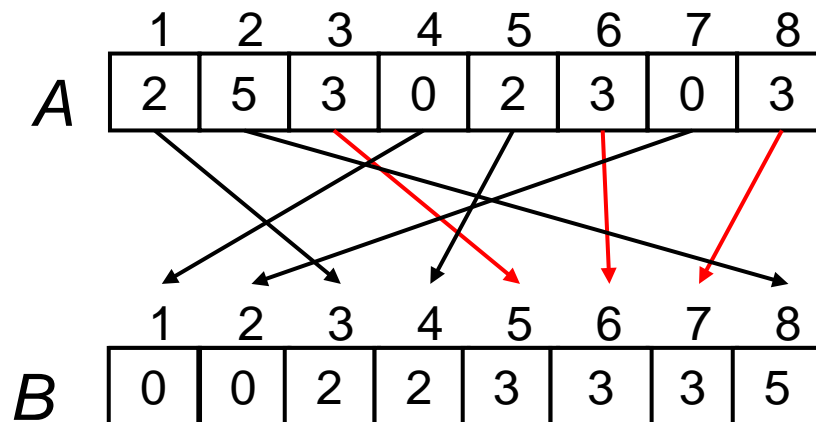
Korrektheit von Counting-Sort

Am Ende gilt: $\forall i \in \{1, \dots, \text{length}(A)\}$: Pos. von $A[i]$ in B ist gleich

$|\{ 1 \leq l \leq \text{length}(A) \mid A[l] < A[i] \}| + |\{ 1 \leq l < i \mid A[l] = A[i] \}| + 1$

d.h. die Folge ist sortiert **bei Beibehaltung der Reihenfolge gleicher El.**

Betrachte unser Beispiel:



Radixsort

Ideen:

- verwende k -adische Darstellung der Schlüssel
- Sortiere Ziffer für Ziffer gemäß **Counting-Sort**
- Behalte bei Betrachtung der i -ten Ziffer die **Reihenfolge** der Elemente mit **gleicher** i -ter Ziffer bei (s. Countingsort)

Annahme: für alle Zahlen z : $0 \leq z < k^d$

Radixsort

Radixsort(A,k)

1. for $i \leftarrow 1$ to d do
2. Counting-Sort(A,B,i,k-1)
3. ▷ sortiere gemäß der $A_i[j]$ -Werte, wobei $A_i[j]$ die i -te Ziffer (von rechts) in $A[j]$ angibt

Laufzeit: $O(d(n+k))$

$k=n$ und d konstant: Laufzeit $O(n)$.

Radixsort

Counting-Sort(A,B,z,k)

1. for $i \leftarrow 0$ to k do
2. $C[i] \leftarrow 0$
3. for $j \leftarrow 0$ to $\text{length}(A)$ do
4. $C[A_z[j]] \leftarrow C[A_z[j]] + 1$
5. ▷ $C[i]$ enthält Anzahl der Elemente in A mit Wert i
6. for $i \leftarrow 1$ to k do
7. $C[i] \leftarrow C[i] + C[i-1]$
8. ▷ $C[i]$ enthält Anzahl der Elemente in A mit Wert $\leq i$
9. for $j \leftarrow \text{length}(A)$ downto 1 do
10. $B[C[A_z[j]]] \leftarrow A[j]$
11. $C[A_z[j]] \leftarrow C[A_z[j]] - 1$

Radixsort

Beispiel:

12 203 3 74 24 17 112

Nach Counting-Sort(A,B,1,9):

12 112 203 3 74 24 17

Radixsort

Beispiel:

12 112 203 3 74 24 17

Nach Counting-Sort(A,B,2,9):

203 03 12 112 17 24 74

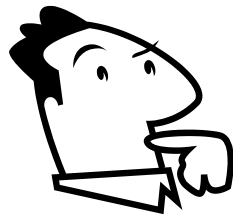
Radixsort

Beispiel:

203 3 12 112 17 24 74

Nach Counting-Sort(A,B,3,9):

003 012 017 024 074 112 203



Zauberei???

Radixsort

Korrektheit:

- Für jedes Paar $a < b$ in A gilt:
es existiert i mit $a_i < b_i$ und $a_j = b_j$ für alle $j > i$
- Schleifendurchlauf für i : $\text{pos}_B(a) < \text{pos}_B(b)$
($\text{pos}_B(a)$: Position von a in Feld B)
- Schleifendurchlauf für $j > i$: Ordnung wird **beibehalten** wegen Counting-Sort, d.h. auch danach gilt $\text{pos}_B(a) < \text{pos}_B(b)$
- Da das für alle Zahlenpaare gilt, ist die Folge am Ende sortiert.