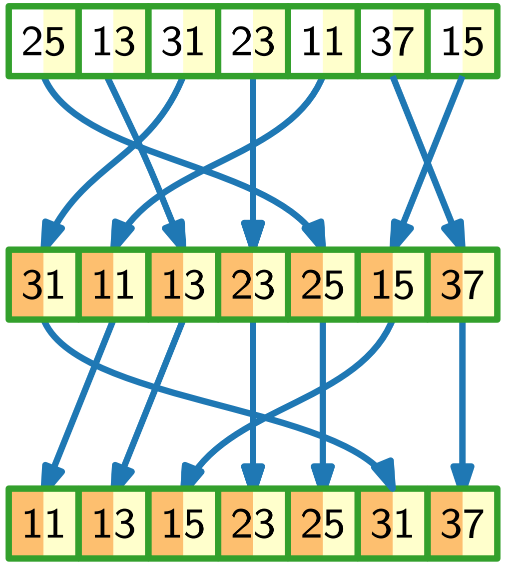
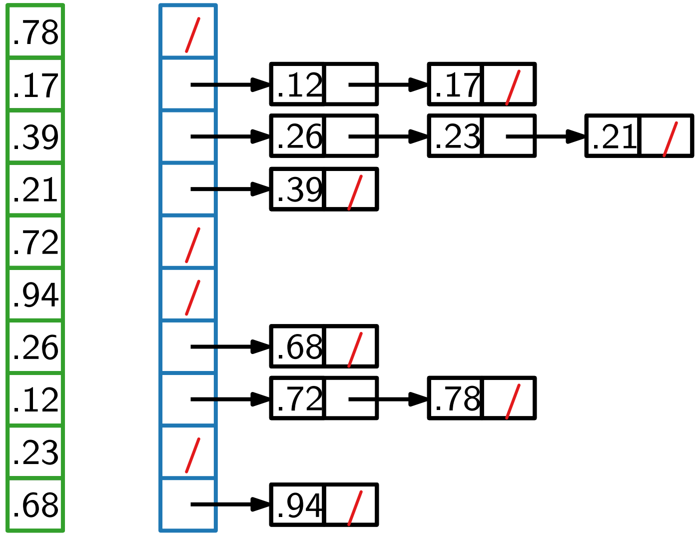
Algorithmen und Datenstrukturen



Vorlesung 9: Sortieren in Linearzeit



Alexander Wolff

Wintersemester 2024

Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle \xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle \xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

vergleichsbasierter Sortieralgorithmus

Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:

Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

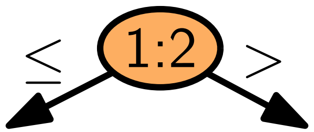
Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:

- innere Knoten $=$ Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)

Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:



- innere Knoten $=$ Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)

Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

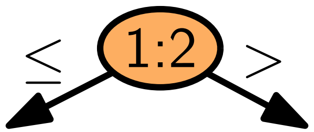
Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:

- innere Knoten $=$ Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter $=$ sortierte Permutationen der Eingabe

Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:

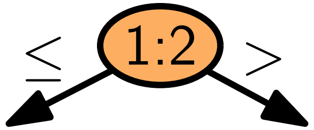


- innere Knoten $=$ Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter $=$ sortierte Permutationen der Eingabe
- Kanten $=$ Ergebnisse ($\leq$/$>$) eines Vergleichs

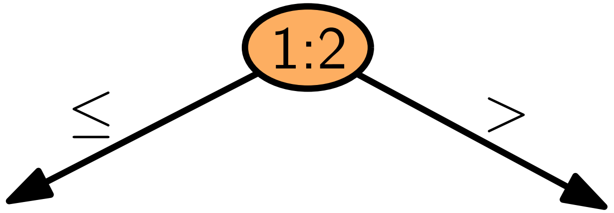
Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:



- innere Knoten = Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter = sortierte Permutationen der Eingabe
- Kanten = Ergebnisse ($\leq$/$>$) eines Vergleichs



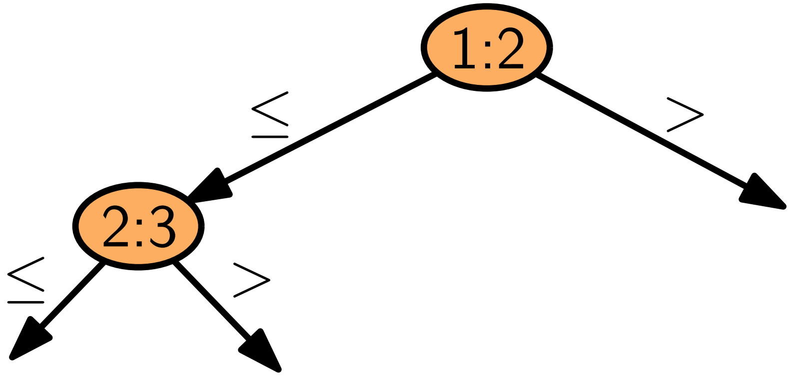
Entscheidungsbaum für InsertionSort und $n = 3$

Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:

- innere Knoten = Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter = sortierte Permutationen der Eingabe
- Kanten = Ergebnisse ($\leq$/$>$) eines Vergleichs

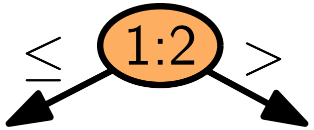


Entscheidungsbaum für InsertionSort und $n = 3$

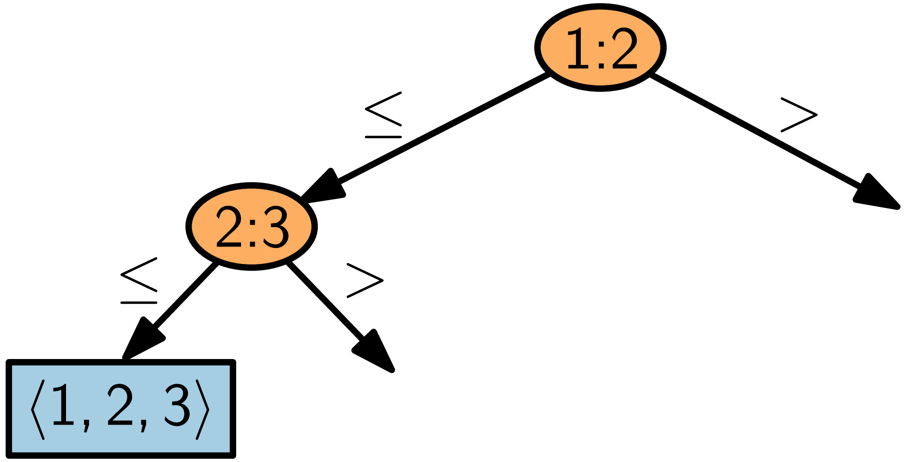
Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:



- innere Knoten $=$ Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter $=$ sortierte Permutationen der Eingabe
- Kanten $=$ Ergebnisse ($\leq$/$>$) eines Vergleichs



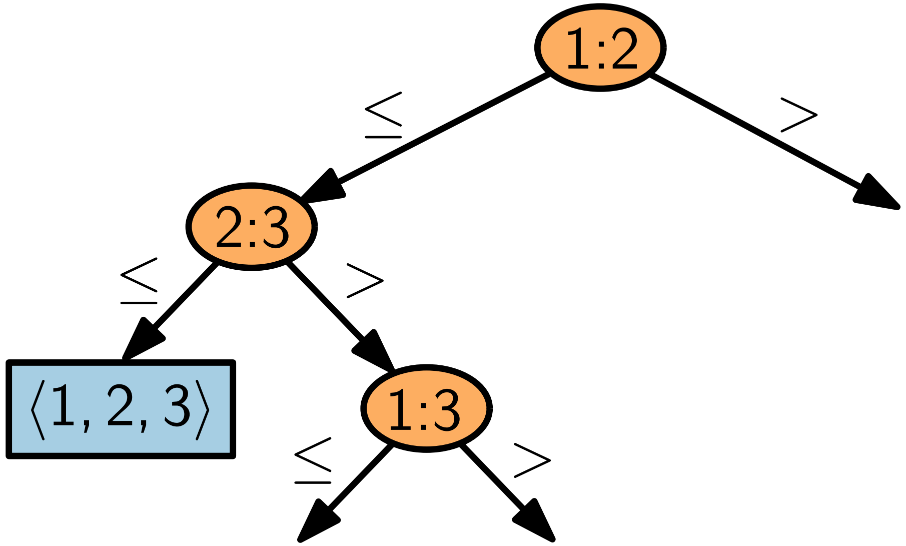
Entscheidungsbaum für InsertionSort und $n = 3$

Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:

- innere Knoten = Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter = sortierte Permutationen der Eingabe
- Kanten = Ergebnisse ($\leq$/$>$) eines Vergleichs



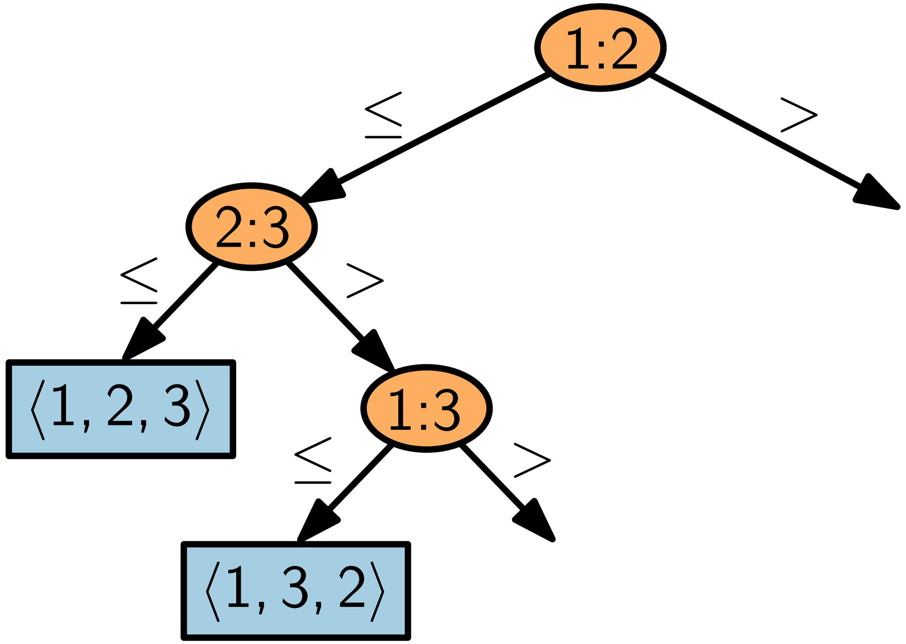
Entscheidungsbaum für InsertionSort und $n = 3$

Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle \xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:

- innere Knoten = Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter = sortierte Permutationen der Eingabe
- Kanten = Ergebnisse ($\leq / >$) eines Vergleichs



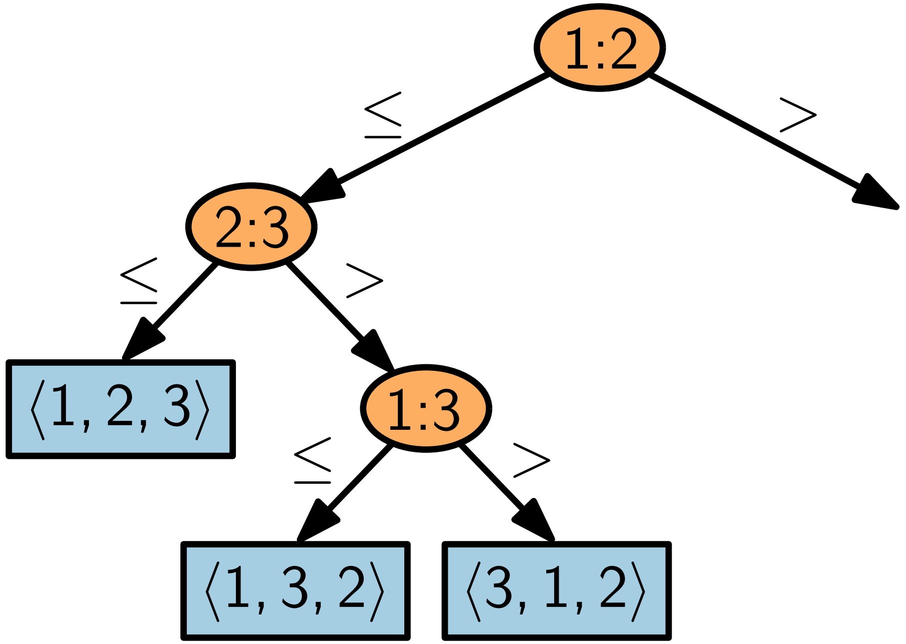
Entscheidungsbaum für InsertionSort und $n = 3$

Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \dots, a_n \rangle \xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:

- innere Knoten = Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter = sortierte Permutationen der Eingabe
- Kanten = Ergebnisse ($\leq$/$>$) eines Vergleichs

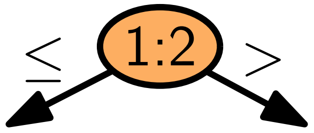


Entscheidungsbaum für InsertionSort und $n = 3$

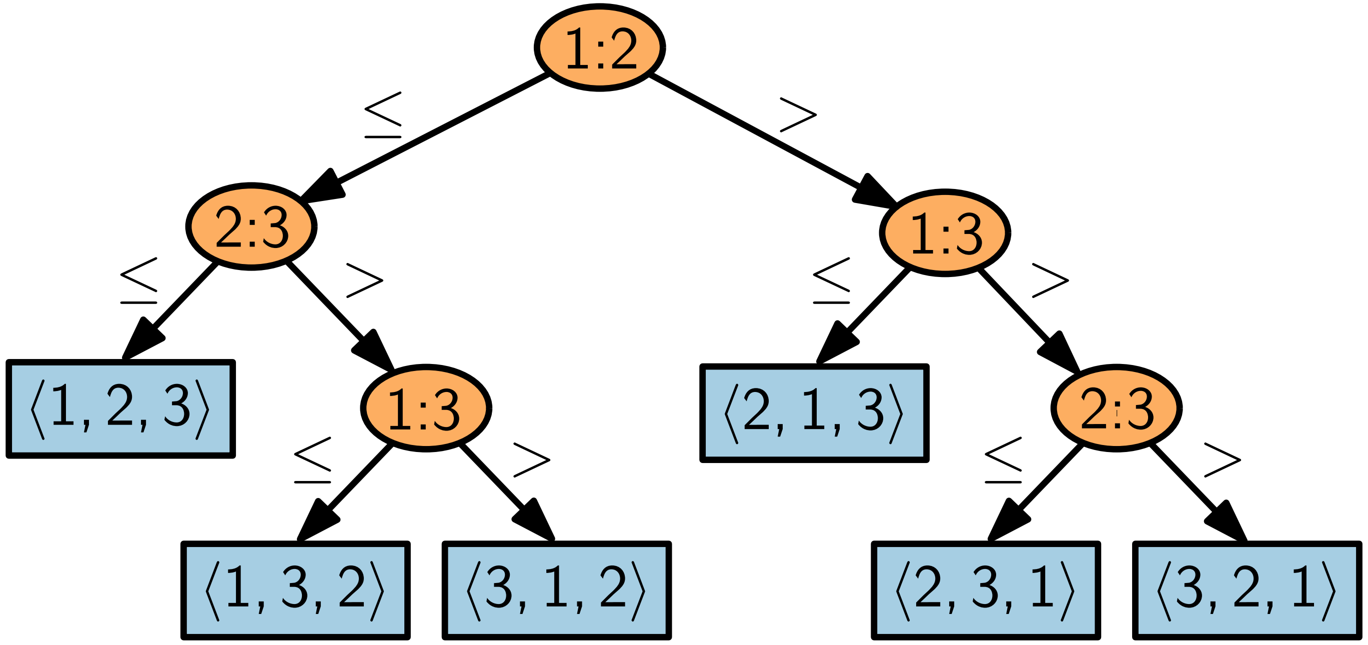
Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:



- innere Knoten $=$ Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter $=$ sortierte Permutationen der Eingabe
- Kanten $=$ Ergebnisse ($\leq$/$>$) eines Vergleichs



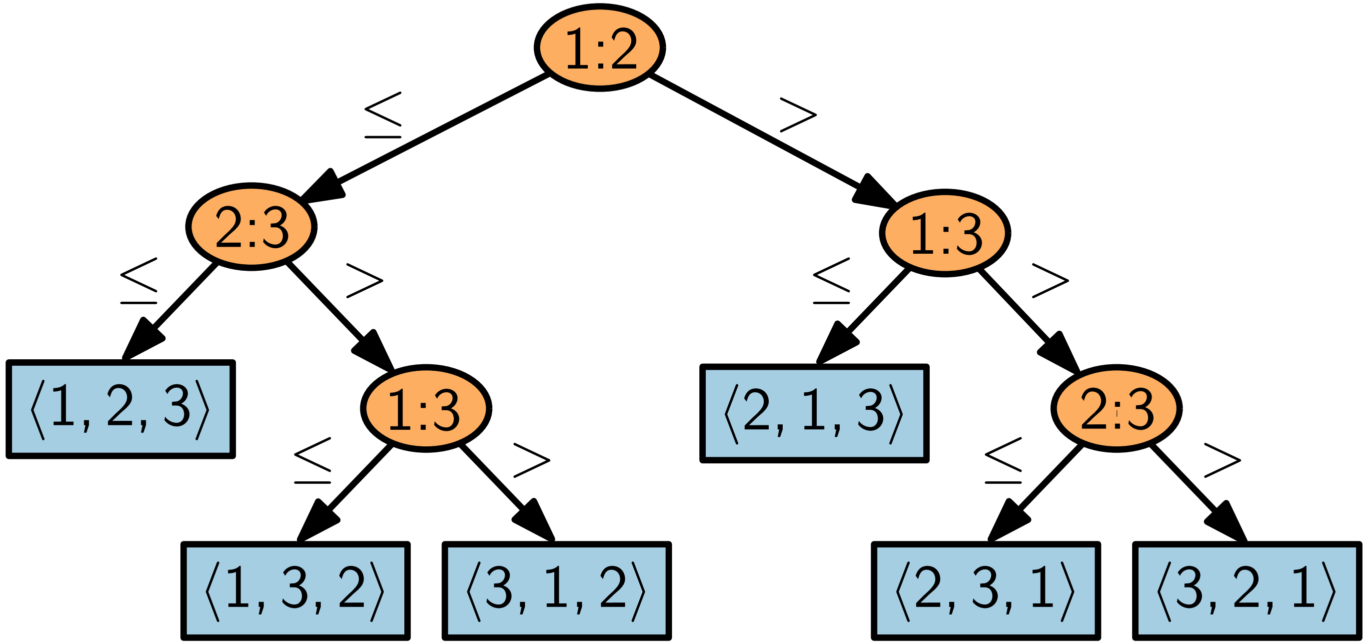
Entscheidungsbaum für InsertionSort und $n = 3$

Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle \xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:

- innere Knoten = Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter = sortierte Permutationen der Eingabe
- Kanten = Ergebnisse ($\leq$/$>$) eines Vergleichs



Entscheidungsbaum für InsertionSort und $n = 3$

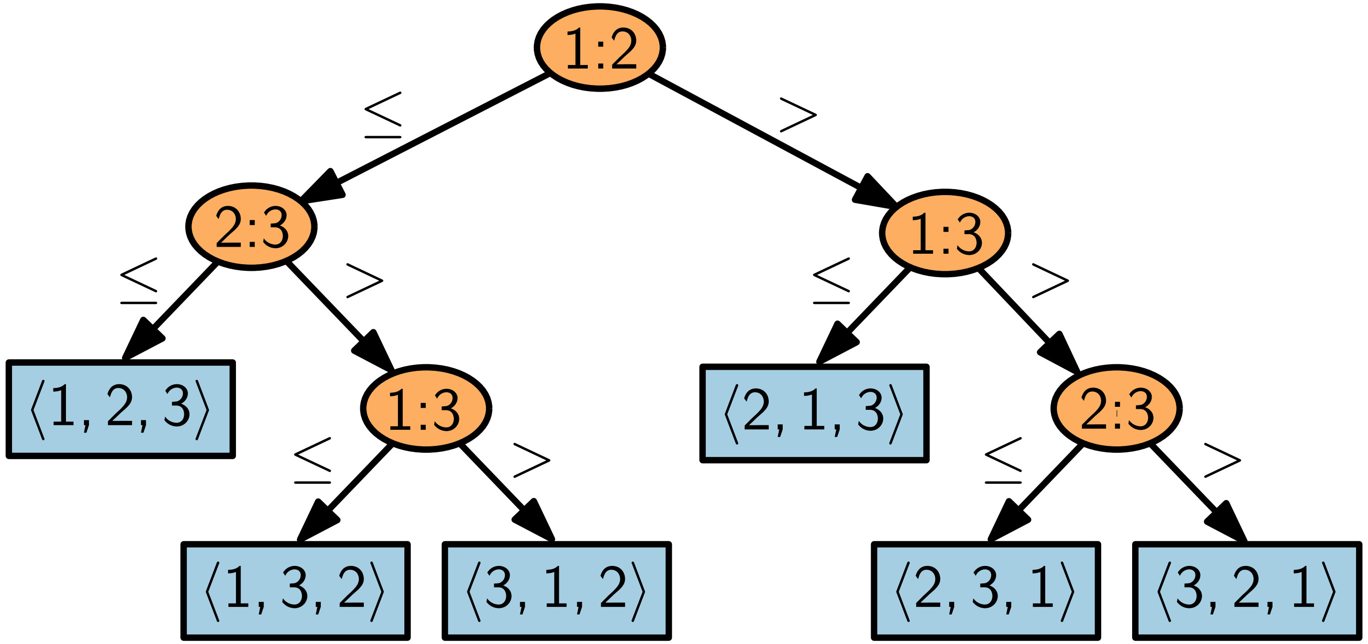
Anz. Vgl. im besten Fall

Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:

- innere Knoten = Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter = sortierte Permutationen der Eingabe
- Kanten = Ergebnisse ($\leq$/$>$) eines Vergleichs



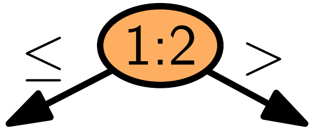
Entscheidungsbaum für InsertionSort und $n = 3$

Anz. Vgl. im besten Fall
= Länge eines kürzesten
Wurzel-Blatt-Pfads

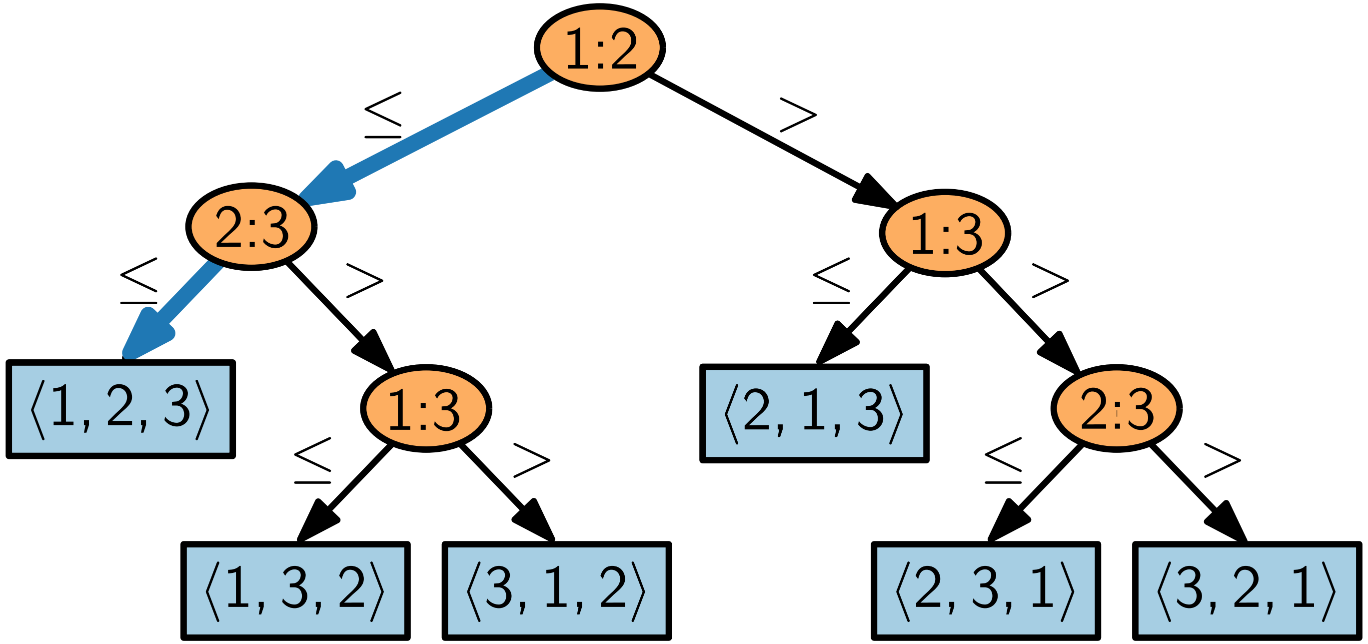
Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:



- innere Knoten = Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter = sortierte Permutationen der Eingabe
- Kanten = Ergebnisse ($\leq$/$>$) eines Vergleichs



Entscheidungsbaum für InsertionSort und $n = 3$

Anz. Vgl. im besten Fall

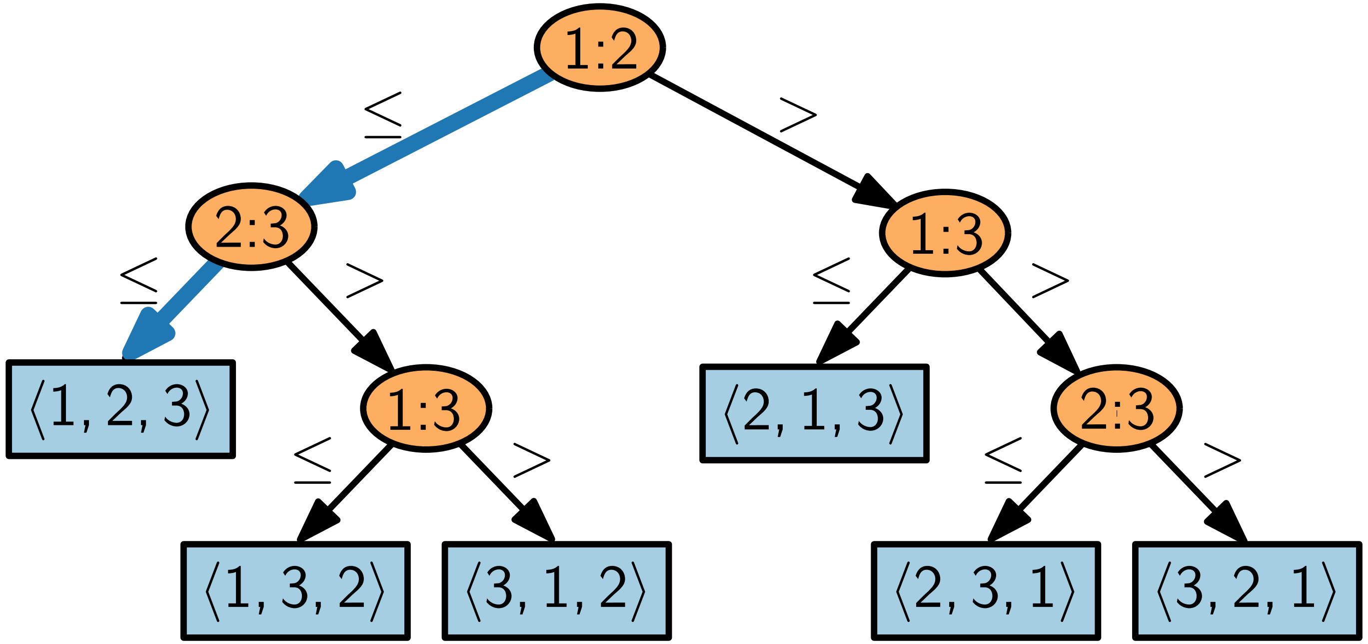
= Länge eines kürzesten Wurzel-Blatt-Pfads

Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:

- innere Knoten = Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter = sortierte Permutationen der Eingabe
- Kanten = Ergebnisse ($\leq$/$>$) eines Vergleichs



Entscheidungsbaum für InsertionSort und $n = 3$

Anz. Vgl. im besten Fall

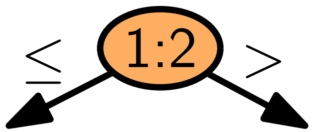
= Länge eines kürzesten Wurzel-Blatt-Pfads

$=_{\text{hier}}$ 2

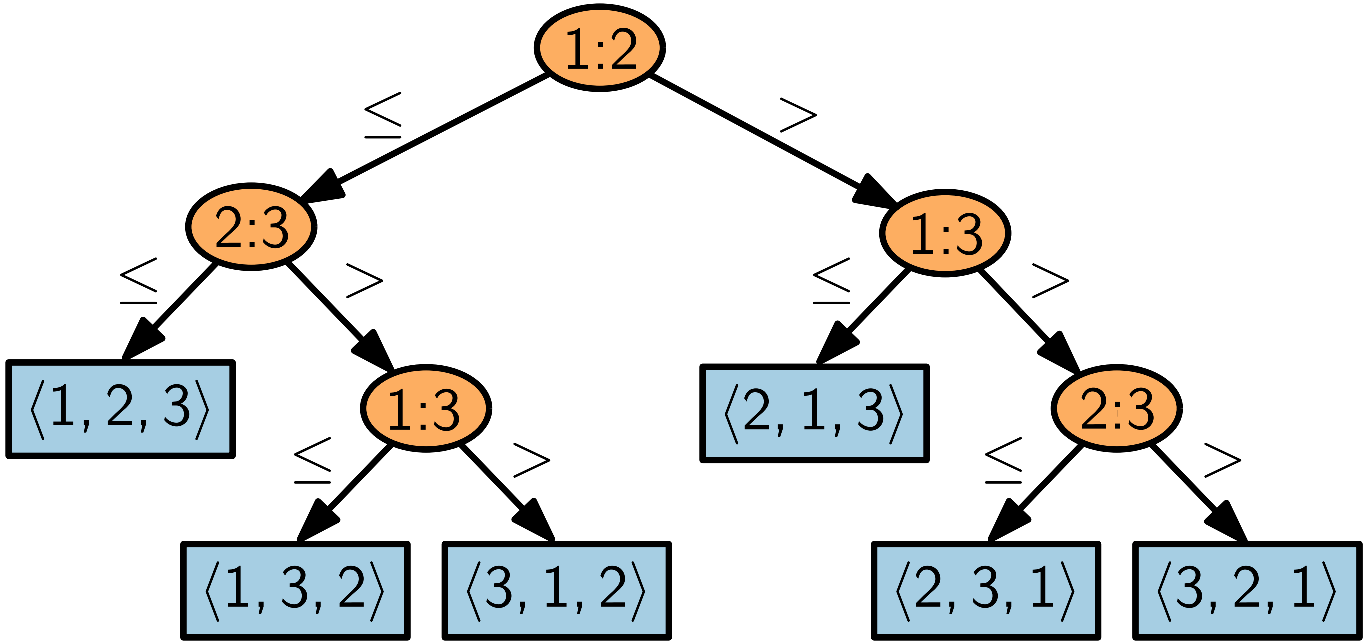
Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:



- innere Knoten = Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter = sortierte Permutationen der Eingabe
- Kanten = Ergebnisse ($\leq$/$>$) eines Vergleichs



Entscheidungsbaum für InsertionSort und $n = 3$

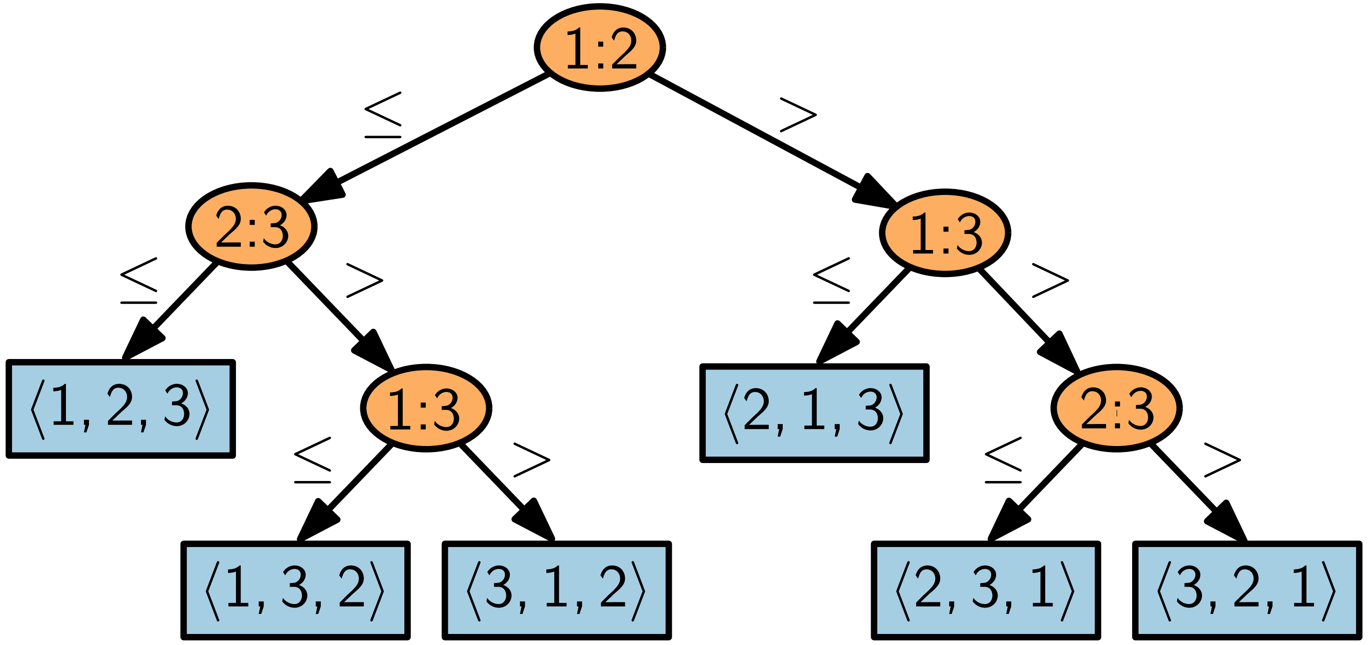
Anz. Vgl. im schlechtesten Fall

Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:

- innere Knoten = Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter = sortierte Permutationen der Eingabe
- Kanten = Ergebnisse ($\leq / >$) eines Vergleichs



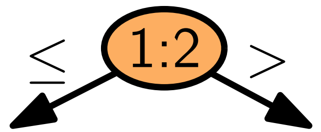
Entscheidungsbaum für InsertionSort und $n = 3$

Anz. Vgl. im schlechtesten Fall
= Länge eines **längsten** Wurzel-Blatt-Pfads

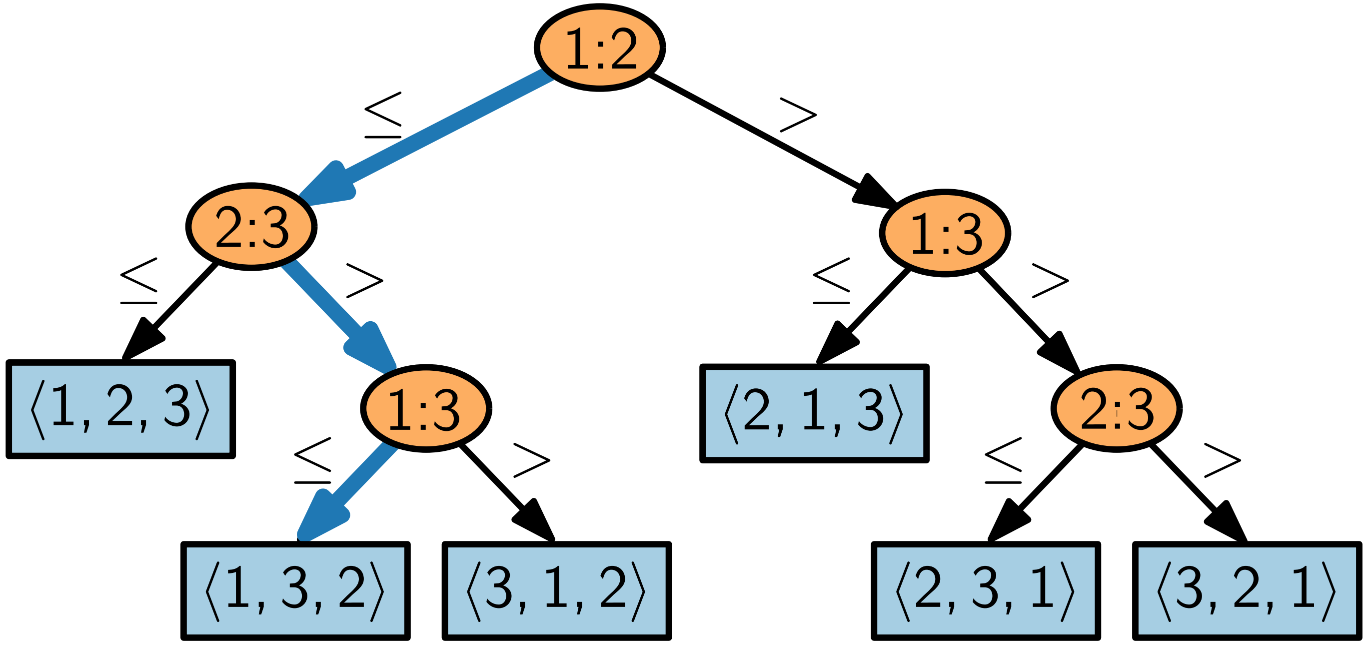
Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:



- innere Knoten = Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter = sortierte Permutationen der Eingabe
- Kanten = Ergebnisse ($\leq$/$>$) eines Vergleichs



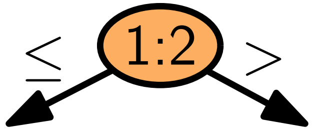
Entscheidungsbaum für InsertionSort und $n = 3$

Anz. Vgl. im schlechtesten Fall
= Länge eines **längsten** Wurzel-Blatt-Pfads

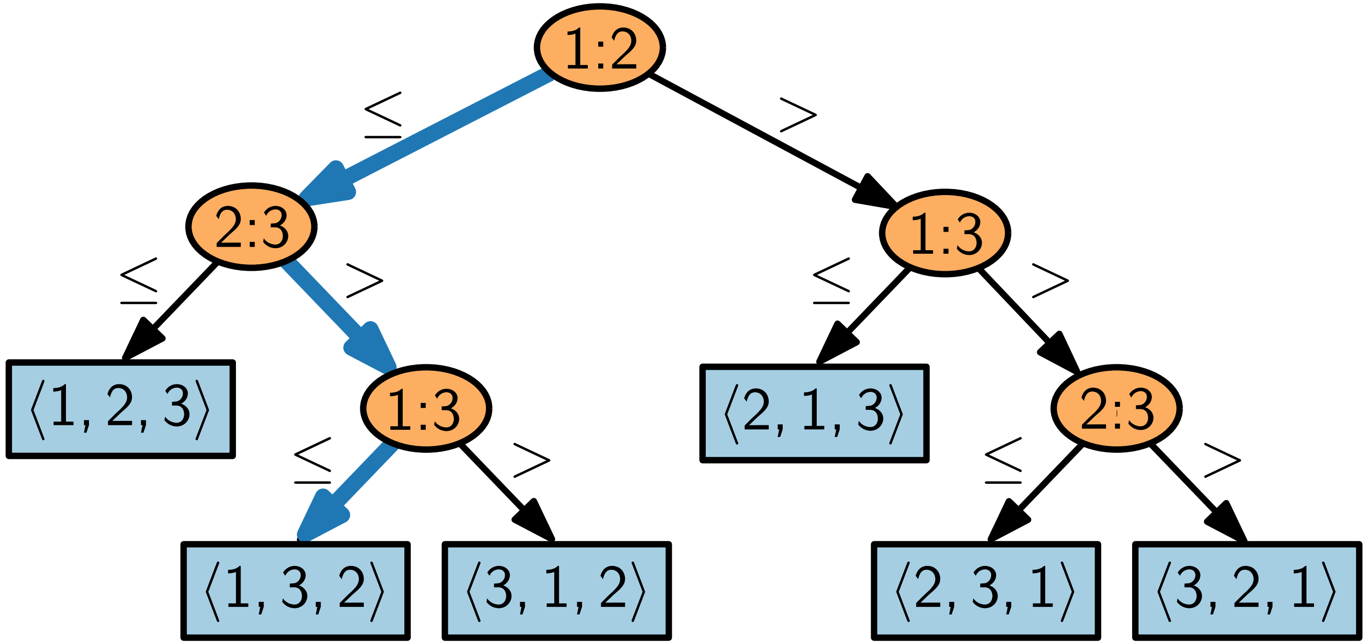
Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:



- innere Knoten = Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter = sortierte Permutationen der Eingabe
- Kanten = Ergebnisse ($\leq / >$) eines Vergleichs



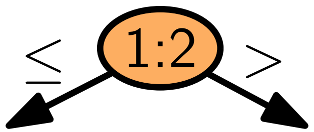
Entscheidungsbaum für InsertionSort und $n = 3$

Anz. Vgl. im schlechtesten Fall
= Länge eines **längsten** Wurzel-Blatt-Pfads
=: Höhe des Baums

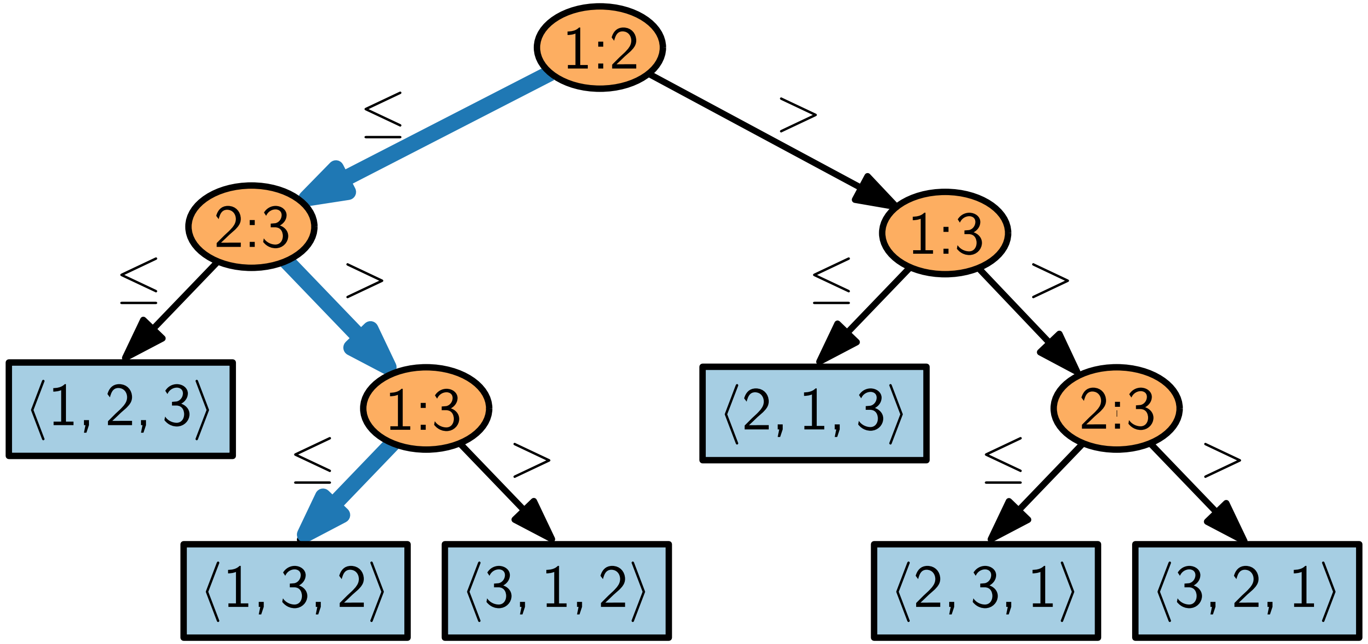
Sortieren durch Vergleichen

Eingabefolge $\langle a_1, a_2, \ldots, a_n \rangle \xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein **vergleichsbasierter** Sortieralgorithmus charakterisiert durch seinen **Entscheidungsbaum**:



- innere Knoten $=$ Vergleiche (o.B.d.A. immer $\leq$, z.B. „$a_1 \leq a_2$?“)
- Blätter $=$ sortierte Permutationen der Eingabe
- Kanten $=$ Ergebnisse ($\leq$/$>$) eines Vergleichs



Entscheidungsbaum für InsertionSort und $n = 3$

Anz. Vgl. im schlechtesten Fall
$=$ Länge eines **längsten** Wurzel-Blatt-Pfads
$=:$ Höhe des Baums
$=_{\text{hier}}$ 3

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

Anz. Blätter = Anz. Permutationen von n Obj. =

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

$$\text{Anz. Blätter} = \text{ Anz. Permutationen von } n \text{ Obj.} = n!$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

$$\text{Anz. Blätter} = \text{Anz. Permutationen von } n \text{ Obj.} = n!$$

$$\text{Höhe Binärbaum mit } B \text{ Blättern } \geq$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

$$\text{Anz. Blätter} = \text{Anz. Permutationen von } n \text{ Obj.} = n!$$

$$\text{Höhe Binärbaum mit } B \text{ Blättern} \geq \lceil \log_2 B \rceil$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

$$\text{Anz. Blätter} = \text{Anz. Permutationen von } n \text{ Obj.} = n!$$

$$\text{Höhe Binärbaum mit } B \text{ Blättern} \geq \lceil \log_2 B \rceil$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

$$\text{Anz. Blätter} = \text{Anz. Permutationen von } n \text{ Obj.} = n!$$

$$\text{Höhe Binärbaum mit } B \text{ Blättern} \geq \lceil \log_2 B \rceil$$

$$\text{Höhe Entscheidungsbaum} \geq$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

$$\text{Anz. Blätter} = \text{Anz. Permutationen von } n \text{ Obj.} = n!$$
$$\text{Höhe Binärbaum mit } B \text{ Blättern} \geq \lceil \log_2 B \rceil$$

$$\text{Höhe Entscheidungsbaum} \geq \log_2 n!$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

$$\text{Anz. Blätter} = \text{Anz. Permutationen von } n \text{ Obj.} = n!$$

$$\text{Höhe Binärbaum mit } B \text{ Blättern} \geq \lceil \log_2 B \rceil$$

$$\text{Höhe Entscheidungsbaum} \geq \log_2 n! = \log_2 \prod_{i=1}^{n} i$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

$$\text{Anz. Blätter} = \text{Anz. Permutationen von } n \text{ Obj.} = n!$$

$$\text{Höhe Binärbaum mit } B \text{ Blättern} \ \geq \ \lceil \log_2 B \rceil$$

$$\text{Höhe Entscheidungsbaum} \ \geq \ \log_2 n! \ = \ \log_2 \prod_{i=1}^{n} i \ = \ \sum_{i=1}^{n} \log_2 i$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

Anz. Blätter $=$ Anz. Permutationen von n Obj. $= n!$

Höhe Binärbaum mit B Blättern $\geq \lceil \log_2 B \rceil$

Höhe Entscheidungsbaum $\geq \log_2 n! = \log_2 \prod_{i=1}^{n} i = \sum_{i=1}^{n} \log_2 i$

$$\geq \int_1^n \log_2 x \, dx$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

$$\text{Anz. Blätter} = \text{Anz. Permutationen von } n \text{ Obj.} = n!$$
$$\text{Höhe Binärbaum mit } B \text{ Blättern} \geq \lceil \log_2 B \rceil$$

$$\text{Höhe Entscheidungsbaum} \geq \log_2 n! = \log_2 \prod_{i=1}^{n} i = \sum_{i=1}^{n} \log_2 i$$
$$\geq \int_1^n \log_2 x \, dx = \frac{1}{\ln 2} \int_1^n \ln x \, dx$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

$$\text{Anz. Blätter} = \text{ Anz. Permutationen von } n \text{ Obj.} = n!$$

$$\text{Höhe Binärbaum mit } B \text{ Blättern } \geq \lceil \log_2 B \rceil$$

$$\text{Höhe Entscheidungsbaum } \geq \log_2 n! = \log_2 \prod_{i=1}^{n} i = \sum_{i=1}^{n} \log_2 i$$

$$\geq \int_1^n \log_2 x \, dx = \frac{1}{\ln 2} \int_1^n \ln x \, dx = \frac{1}{\ln 2} \int_1^n 1 \cdot \ln x \, dx$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

Anz. Blätter = Anz. Permutationen von n Obj. = $n!$

Höhe Binärbaum mit B Blättern $\geq \lceil \log_2 B \rceil$

Höhe Entscheidungsbaum $\geq \log_2 n! = \log_2 \prod_{i=1}^{n} i = \sum_{i=1}^{n} \log_2 i$

$$\geq \int_1^n \log_2 x \, dx = \frac{1}{\ln 2} \int_1^n \ln x \, dx = \frac{1}{\ln 2} \int_1^n 1 \cdot \ln x \, dx$$

$$=$$

Partielle Integration. $\int_a^b u'v = [uv]_a^b - \int_a^b uv'$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

Anz. Blätter $=$ Anz. Permutationen von n Obj. $= n!$

Höhe Binärbaum mit B Blättern $\geq \lceil \log_2 B \rceil$

Höhe Entscheidungsbaum $\geq \log_2 n! = \log_2 \prod_{i=1}^{n} i = \sum_{i=1}^{n} \log_2 i$

$$\geq \int_1^n \log_2 x \, dx = \frac{1}{\ln 2} \int_1^n \ln x \, dx = \frac{1}{\ln 2} \int_1^n 1 \cdot \ln x \, dx$$

$$=$$

Partielle Integration. $\int_a^b u' v = [uv]_a^b - \int_a^b uv'$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

Anz. Blätter $=$ Anz. Permutationen von n Obj. $= n!$

Höhe Binärbaum mit B Blättern $\geq \lceil \log_2 B \rceil$

Höhe Entscheidungsbaum $\geq \log_2 n! = \log_2 \prod_{i=1}^{n} i = \sum_{i=1}^{n} \log_2 i$

$$\geq \int_1^n \log_2 x \, dx = \frac{1}{\ln 2} \int_1^n \ln x \, dx = \frac{1}{\ln 2} \int_1^n 1 \cdot \ln x \, dx$$

$$=$$

Partielle Integration. $\int_a^b u'v = [uv]_a^b - \int_a^b uv'$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

$$\text{Anz. Blätter} = \text{Anz. Permutationen von } n \text{ Obj.} = n!$$
$$\text{Höhe Binärbaum mit } B \text{ Blättern} \geq \lceil \log_2 B \rceil$$

$$\text{Höhe Entscheidungsbaum} \geq \log_2 n! = \log_2 \prod_{i=1}^{n} i = \sum_{i=1}^{n} \log_2 i$$
$$\geq \int_1^n \log_2 x \, dx = \frac{1}{\ln 2} \int_1^n \ln x \, dx = \frac{1}{\ln 2} \int_1^n 1 \cdot \ln x \, dx$$
$$=$$

Partielle Integration. $\int_a^b u' v = [uv]_a^b - \int_a^b u v'$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

Anz. Blätter = Anz. Permutationen von n Obj. $= n!$

Höhe Binärbaum mit B Blättern $\geq \lceil \log_2 B \rceil$

Höhe Entscheidungsbaum $\geq \log_2 n! = \log_2 \prod_{i=1}^{n} i = \sum_{i=1}^{n} \log_2 i$

$$\geq \int_1^n \log_2 x \, dx = \frac{1}{\ln 2} \int_1^n \ln x \, dx = \frac{1}{\ln 2} \int_1^n 1 \cdot \ln x \, dx$$

$$= \frac{1}{\ln 2}\left([\ \cdot\ ]_1^n - \int_1^n \ \cdot\ dx \right)$$

Partielle Integration. $\int_a^b u'v = [uv]_a^b - \int_a^b uv'$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

Anz. Blätter = Anz. Permutationen von n Obj. = $n!$

Höhe Binärbaum mit B Blättern $\geq \lceil \log_2 B \rceil$

Höhe Entscheidungsbaum $\geq \log_2 n! = \log_2 \prod_{i=1}^{n} i = \sum_{i=1}^{n} \log_2 i$

$$\geq \int_1^n \log_2 x \, dx = \frac{1}{\ln 2} \int_1^n \ln x \, dx = \frac{1}{\ln 2} \int_1^n 1 \cdot \ln x \, dx$$

$$= \frac{1}{\ln 2}\left([x \cdot \quad]_1^n - \int_1^n x \cdot \quad dx \right)$$

Partielle Integration. $\int_a^b u'v = [uv]_a^b - \int_a^b uv'$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

$$\text{Anz. Blätter} = \text{Anz. Permutationen von } n \text{ Obj.} = n!$$

$$\text{Höhe Binärbaum mit } B \text{ Blättern } \geq \lceil \log_2 B \rceil$$

$$\text{Höhe Entscheidungsbaum } \geq \log_2 n! = \log_2 \prod_{i=1}^{n} i = \sum_{i=1}^{n} \log_2 i$$

$$\geq \int_1^n \log_2 x \, dx = \frac{1}{\ln 2} \int_1^n \ln x \, dx = \frac{1}{\ln 2} \int_1^n 1 \cdot \ln x \, dx$$

$$= \frac{1}{\ln 2} \left([x \cdot \ln x]_1^n - \int_1^n x \cdot \quad dx \right)$$

Partielle Integration. $\int_a^b u'v = [uv]_a^b - \int_a^b uv'$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

$$\text{Anz. Blätter} = \text{Anz. Permutationen von } n \text{ Obj.} = n!$$

$$\text{Höhe Binärbaum mit } B \text{ Blättern} \geq \lceil \log_2 B \rceil$$

$$\text{Höhe Entscheidungsbaum} \geq \log_2 n! = \log_2 \prod_{i=1}^{n} i = \sum_{i=1}^{n} \log_2 i$$

$$\geq \int_1^n \log_2 x \, dx = \frac{1}{\ln 2} \int_1^n \ln x \, dx = \frac{1}{\ln 2} \int_1^n 1 \cdot \ln x \, dx$$

$$= \frac{1}{\ln 2} \left([x \cdot \ln x]_1^n - \int_1^n x \cdot \frac{1}{x} \, dx \right)$$

Partielle Integration. $\int_a^b u'v = [uv]_a^b - \int_a^b uv'$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

Anz. Blätter = Anz. Permutationen von n Obj. $= n!$

Höhe Binärbaum mit B Blättern $\geq \lceil \log_2 B \rceil$

Höhe Entscheidungsbaum $\geq \log_2 n! = \log_2 \prod_{i=1}^{n} i = \sum_{i=1}^{n} \log_2 i$

$$\geq \int_1^n \log_2 x \, dx = \frac{1}{\ln 2} \int_1^n \ln x \, dx = \frac{1}{\ln 2} \int_1^n 1 \cdot \ln x \, dx$$

$$= \frac{1}{\ln 2} \left([x \cdot \ln x]_1^n - \int_1^n \underbrace{x \cdot \frac{1}{x}}_{1} dx \right)$$

Partielle Integration. $\int_a^b u'v = [uv]_a^b - \int_a^b uv'$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

Anz. Blätter $=$ Anz. Permutationen von n Obj. $= n!$

Höhe Binärbaum mit B Blättern $\geq \lceil \log_2 B \rceil$

Höhe Entscheidungsbaum $\geq \log_2 n! = \log_2 \prod_{i=1}^{n} i = \sum_{i=1}^{n} \log_2 i$

$$\geq \int_1^n \log_2 x \, dx = \frac{1}{\ln 2} \int_1^n \ln x \, dx = \frac{1}{\ln 2} \int_1^n 1 \cdot \ln x \, dx$$

$$= \frac{1}{\ln 2}\left([x \cdot \ln x]_1^n - \int_1^n \underbrace{x \cdot \frac{1}{x}}_{1} dx \right) = \frac{(n \ln n - 0) - (n-1)}{\ln 2}$$

Partielle Integration. $\int_a^b u'v = [uv]_a^b - \int_a^b uv'$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht **jeder** vergleichsbasierte Sortieralgorithmus im schlechtesten Fall um n **verschiedene** Objekte zu sortieren?

M.a.W. Gegeben

- ein beliebiger vergleichsbasierter Sortieralgorithmus,
- eine Zahl n von verschiedenen Objekten, die man sortieren soll,

welche Höhe hat der Entscheidungsbaum **mindestens**?

Beob.: Die Höhe ist eine Funktion der Blätteranzahl.

Anz. Blätter $=$ Anz. Permutationen von n Obj. $= n!$

Höhe Binärbaum mit B Blättern $\geq \lceil \log_2 B \rceil$

Höhe Entscheidungsbaum $\geq \log_2 n! = \log_2 \prod_{i=1}^{n} i = \sum_{i=1}^{n} \log_2 i$

$$\geq \int_1^n \log_2 x \, dx = \frac{1}{\ln 2} \int_1^n \ln x \, dx = \frac{1}{\ln 2} \int_1^n 1 \cdot \ln x \, dx$$

$$= \frac{1}{\ln 2}\left([x \cdot \ln x]_1^n - \int_1^n \underbrace{x \cdot \frac{1}{x}}_{1} dx \right) = \frac{(n \ln n - 0) - (n-1)}{\ln 2} \in \Omega(n \log n)$$

Partielle Integration. $\int_a^b u'v = [uv]_a^b - \int_a^b uv'$

Resultat

Satz. Jeder vergleichsbasierte Sortieralg. benötigt im schlechtesten Fall $\Omega(n \log n)$ Vergleiche um n Objekte zu sortieren.

Resultat

Satz. Jeder vergleichsbasierte Sortieralg. benötigt im schlechtesten Fall $\Omega(n \log n)$ Vergleiche um n Objekte zu sortieren.

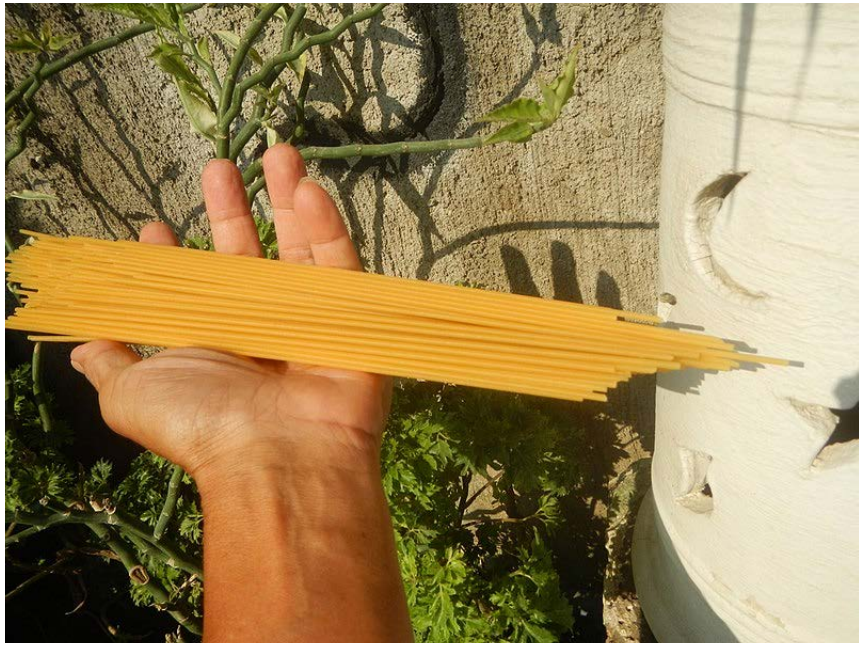
Korollar. MergeSort und HeapSort sind **asymptotisch worst-case optimale** vergleichsbasierte Sortieralgorithmen.

Wir durchbrechen die Schallmauer

[Ensign John Gay, U.S. Navy, Public domain, via Wikimedia Commons]

Wir durchbrechen die Schallmauer

- SPAGHETTISORT

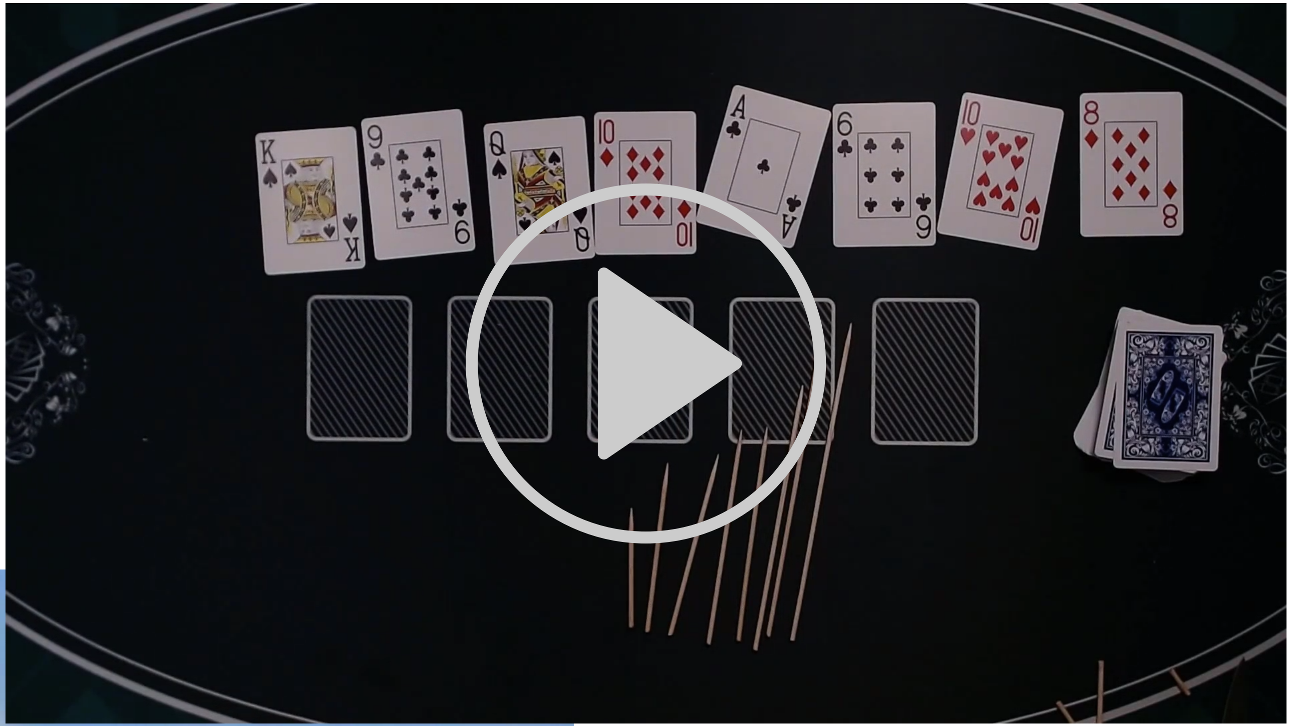


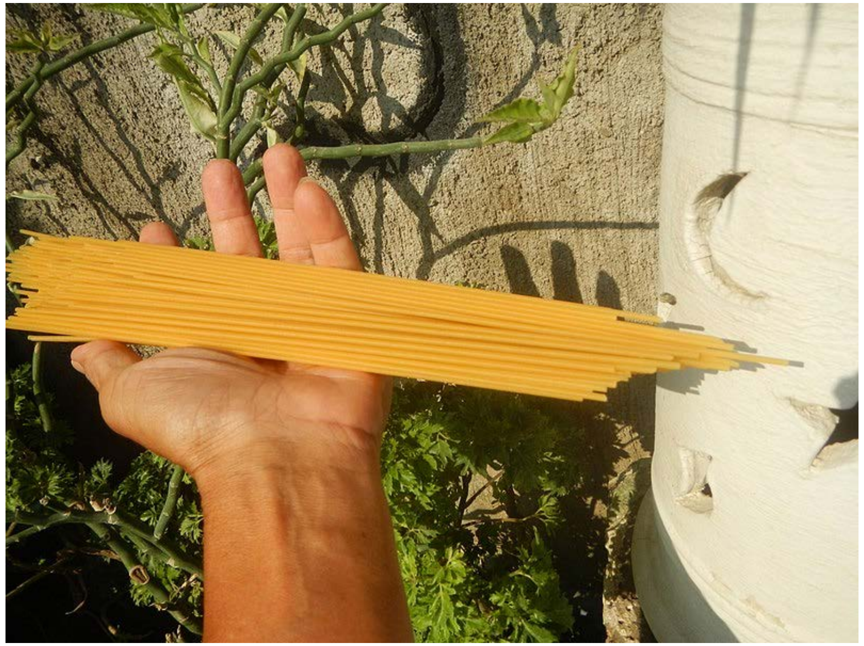
[JFVelasquez Floro, CC0, via Wikimedia Commons]

[Ensign John Gay, U.S. Navy, Public domain, via Wikimedia Commons]

Wir durchbrechen die Schallmauer

- SpaghettiSort



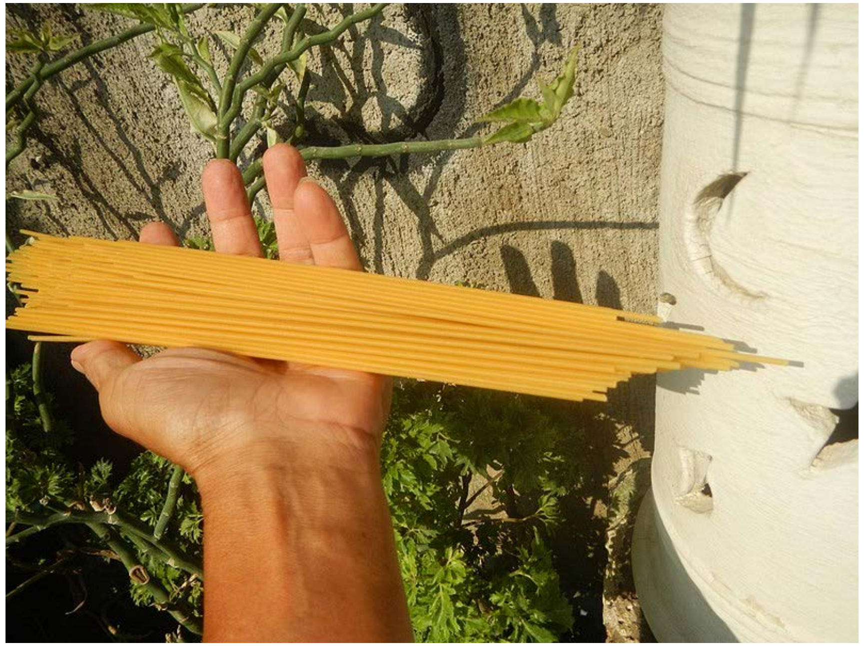


[JFVelasquez Floro, CC0, via Wikimedia Commons]

[Ensign John Gay, U.S. Navy, Public domain, via Wikimedia Commons]

Wir durchbrechen die Schallmauer

(■ SpaghettiSort sortiert Spaghetti nach Länge)



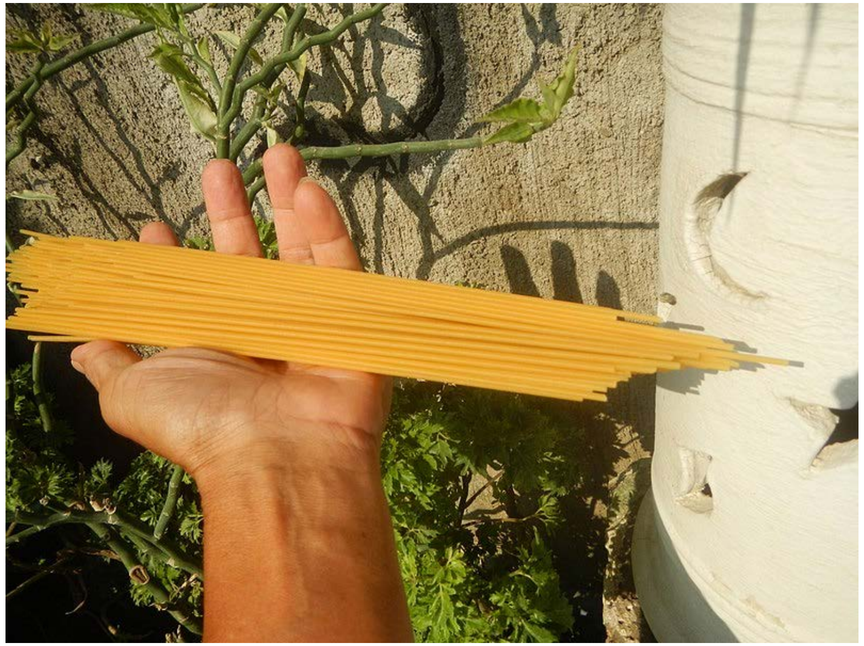
[JFVelasquez Floro, CC0, via Wikimedia Commons]

[Ensign John Gay, U.S. Navy, Public domain, via Wikimedia Commons]

Wir durchbrechen die Schallmauer

(■ SPAGHETTISORT sortiert Spaghetti nach Länge)

■ COUNTINGSORT



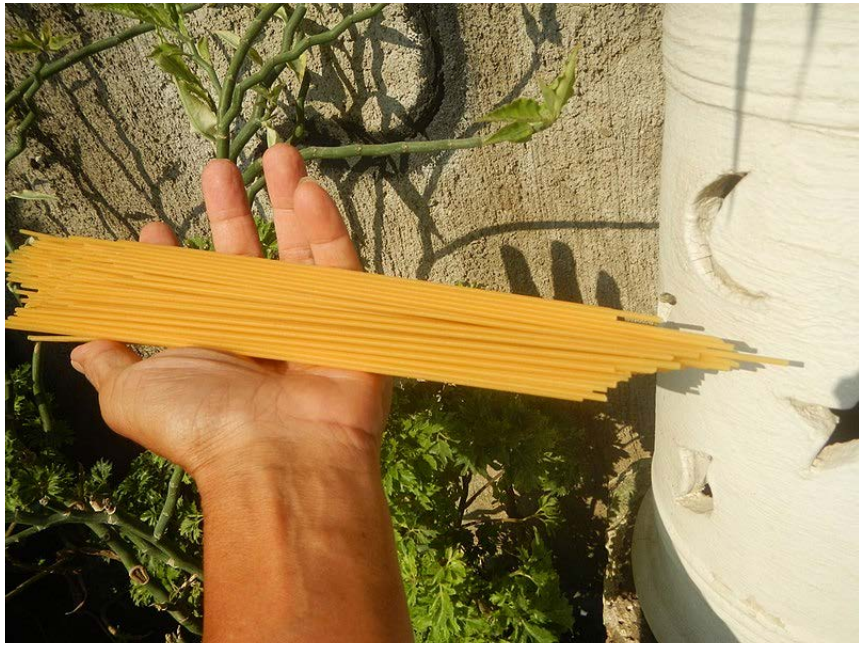
[JFVelasquez Floro, CC0, via Wikimedia Commons]

[Ensign John Gay, U.S. Navy, Public domain, via Wikimedia Commons]

Wir durchbrechen die Schallmauer

(■ SpaghettiSort sortiert Spaghetti nach Länge)

■ CountingSort sortiert Zahlen in $\{0, \ldots, k\}$

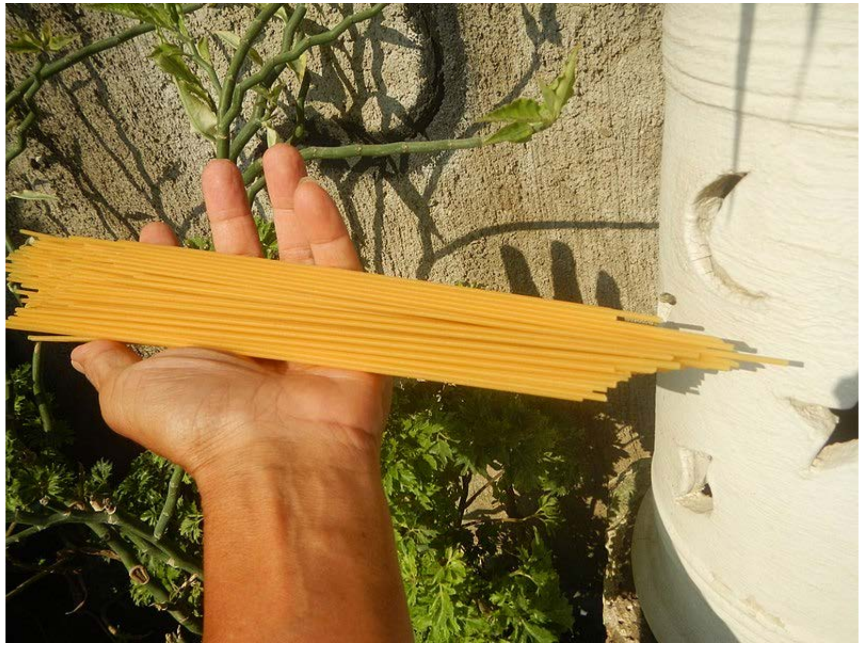


[JFVelasquez Floro, CC0, via Wikimedia Commons]

[Ensign John Gay, U.S. Navy, Public domain, via Wikimedia Commons]

Wir durchbrechen die Schallmauer

- (SpaghettiSort sortiert Spaghetti nach Länge)
- CountingSort sortiert Zahlen in $\{0, \ldots, k\}$
- RadixSort



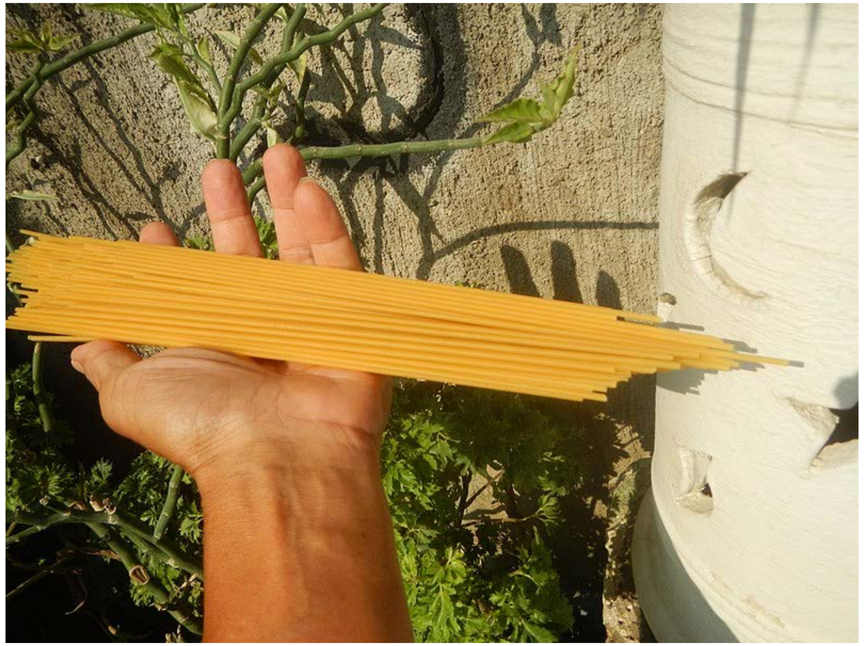
[JFVelasquez Floro, CC0, via Wikimedia Commons]

[Ensign John Gay, U.S. Navy, Public domain, via Wikimedia Commons]

Wir durchbrechen die Schallmauer

(■ SPAGHETTISORT sortiert Spaghetti nach Länge)

- COUNTINGSORT sortiert Zahlen in $\{0, \ldots, k\}$
- RADIXSORT sortiert s-stellige b-adische Zahlen



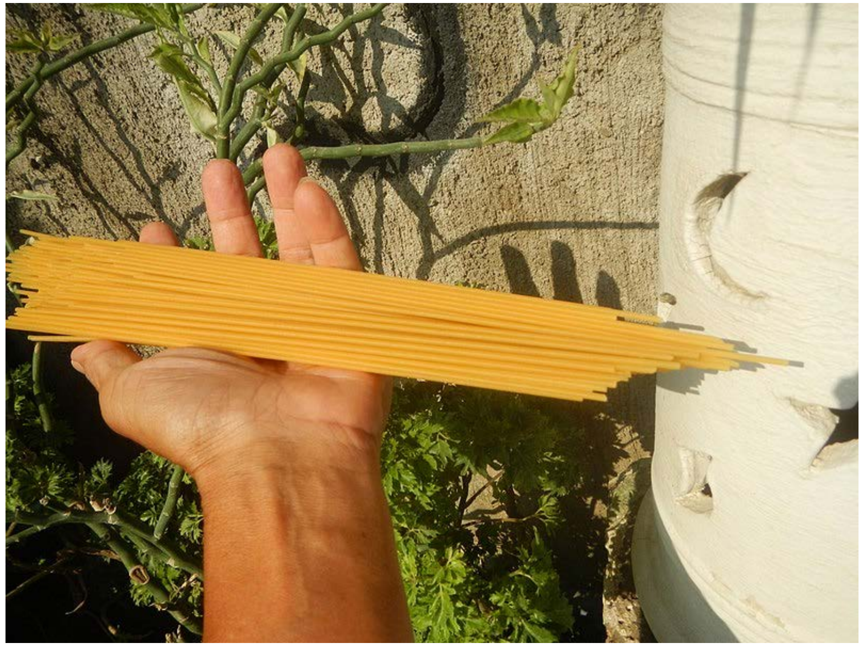
[JFVelasquez Floro, CC0, via Wikimedia Commons]

[Ensign John Gay, U.S. Navy, Public domain, via Wikimedia Commons]

Wir durchbrechen die Schallmauer

(■ SpaghettiSort sortiert Spaghetti nach Länge)

- CountingSort sortiert Zahlen in $\{0, \ldots, k\}$
- RadixSort sortiert s-stellige b-adische Zahlen
- BucketSort

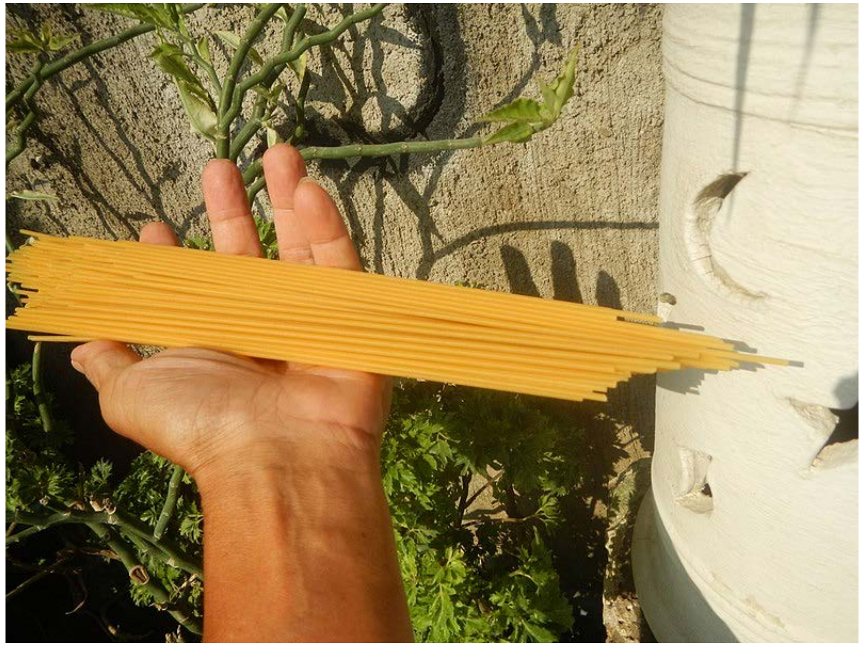


[JFVelasquez Floro, CC0, via Wikimedia Commons]

[Ensign John Gay, U.S. Navy, Public domain, via Wikimedia Commons]

Wir durchbrechen die Schallmauer

- (SpaghettiSort sortiert Spaghetti nach Länge)
- CountingSort sortiert Zahlen in $\{0, \ldots, k\}$
- RadixSort sortiert s-stellige b-adische Zahlen
- BucketSort sortiert gleichverteilte zufällige Zahlen



[JFVelasquez Floro, CC0, via Wikimedia Commons]

[Ensign John Gay, U.S. Navy, Public domain, via Wikimedia Commons]

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C					

COUNTINGSORT

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \boxed{3} & \boxed{0} & \boxed{4} & \boxed{1} & \boxed{3} & \boxed{4} & \boxed{1} & \boxed{4} \end{array} \Rightarrow C \; \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \boxed{3} & \boxed{0} & \boxed{4} & \boxed{1} & \boxed{3} & \boxed{4} & \boxed{1} & \boxed{4} \end{array} \Rightarrow C \; \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	0	0	0	0	0

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	0	0	0	1	0

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \boxed{3} & \boxed{0} & \boxed{4} & \boxed{1} & \boxed{3} & \boxed{4} & \boxed{1} & \boxed{4} \end{array} \Rightarrow C \; \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \boxed{0} & \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \boxed{3} & \boxed{0} & \boxed{4} & \boxed{1} & \boxed{3} & \boxed{4} & \boxed{1} & \boxed{4} \end{array} \Rightarrow C \; \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \boxed{0} & \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \begin{array}{|c|c|c|c|c|c|c|c|} \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} \\ \hline 3 & 0 & 4 & 1 & 3 & 4 & 1 & 4 \\ \hline \end{array} \Rightarrow C \begin{array}{|c|c|c|c|c|} \multicolumn{1}{c}{0} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} \\ \hline 0 & 0 & 0 & 1 & 0 \\ \hline \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

A

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

$\Rightarrow$ C

0	1	2	3	4
1	0	0	1	0

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	0	0	1	0

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	0	0	1	0

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	0	0	1	0

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	0	0	1	1

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \boxed{3} & \boxed{0} & \boxed{4} & \boxed{1} & \boxed{3} & \boxed{4} & \boxed{1} & \boxed{4} \end{array} \Rightarrow C \; \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \boxed{1} & \boxed{0} & \boxed{0} & \boxed{1} & \boxed{1} \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \boxed{3} & \boxed{0} & \boxed{4} & \boxed{1} & \boxed{3} & \boxed{4} & \boxed{1} & \boxed{4} \end{array} \Rightarrow C \; \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \boxed{1} & \boxed{0} & \boxed{0} & \boxed{1} & \boxed{1} \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	0	0	1	1

COUNTINGSORT

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \multicolumn{1}{|c|}{3} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{4} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{3} & \multicolumn{1}{c|}{4} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{4} \\ \hline \end{array} \Rightarrow C \; \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \multicolumn{1}{|c|}{1} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{1} \\ \hline \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	1	0	1	1

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \dots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \boxed{3} & \boxed{0} & \boxed{4} & \boxed{1} & \boxed{3} & \boxed{4} & \boxed{1} & \boxed{4} \end{array} \;\Rightarrow\; C \; \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \boxed{1} & \boxed{1} & \boxed{0} & \boxed{1} & \boxed{1} \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{|c|c|c|c|c|c|c|c|} \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} \\ \hline 3 & 0 & 4 & 1 & 3 & 4 & 1 & 4 \\ \hline \end{array} \Rightarrow C \; \begin{array}{|c|c|c|c|c|} \multicolumn{1}{c}{0} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} \\ \hline 1 & 1 & 0 & 1 & 1 \\ \hline \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{|c|c|c|c|c|c|c|c|} \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} \\ \hline 3 & 0 & 4 & 1 & 3 & 4 & 1 & 4 \\ \hline \end{array} \Rightarrow C \; \begin{array}{|c|c|c|c|c|} \multicolumn{1}{c}{0} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} \\ \hline 1 & 1 & 0 & 2 & 1 \\ \hline \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \boxed{3} & \boxed{0} & \boxed{4} & \boxed{1} & \boxed{3} & \boxed{4} & \boxed{1} & \boxed{4} \end{array} \;\Rightarrow\; C \; \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \boxed{1} & \boxed{1} & \boxed{0} & \boxed{2} & \boxed{1} \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \boxed{3} & \boxed{0} & \boxed{4} & \boxed{1} & \boxed{3} & \boxed{4} & \boxed{1} & \boxed{4} \end{array} \;\Rightarrow\; C \; \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \boxed{1} & \boxed{1} & \boxed{0} & \boxed{2} & \boxed{1} \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	1	0	2	1

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	1	0	2	2

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A\ \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \boxed{3} & \boxed{0} & \boxed{4} & \boxed{1} & \boxed{3} & \boxed{4} & \boxed{1} & \boxed{4} \end{array} \Rightarrow C\ \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \boxed{1} & \boxed{1} & \boxed{0} & \boxed{2} & \boxed{2} \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{|c|c|c|c|c|c|c|c|} \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} \\ \hline 3 & 0 & 4 & 1 & 3 & 4 & 1 & 4 \\ \hline \end{array} \Rightarrow C \; \begin{array}{|c|c|c|c|c|} \multicolumn{1}{c}{0} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} \\ \hline 1 & 1 & 0 & 2 & 2 \\ \hline \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

- A Eingabefeld
- B Ausgabefeld
- C Rechenfeld
- k begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	1	0	2	2

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen: A Eingabefeld | C Rechenfeld
B Ausgabefeld | k begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	2

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \boxed{3} & \boxed{0} & \boxed{4} & \boxed{1} & \boxed{3} & \boxed{4} & \boxed{1} & \boxed{4} \end{array} \Rightarrow C \; \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \boxed{1} & \boxed{2} & \boxed{0} & \boxed{2} & \boxed{2} \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

A

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

$\Rightarrow$ C

0	1	2	3	4
1	2	0	2	2

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen: A Eingabefeld | C Rechenfeld
B Ausgabefeld | k begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	2

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \boxed{3} & \boxed{0} & \boxed{4} & \boxed{1} & \boxed{3} & \boxed{4} & \boxed{1} & \boxed{4} \end{array} \Rightarrow C \; \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \boxed{1} & \boxed{2} & \boxed{0} & \boxed{2} & \boxed{3} \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \boxed{3} & \boxed{0} & \boxed{4} & \boxed{1} & \boxed{3} & \boxed{4} & \boxed{1} & \boxed{4} \end{array} \;\Rightarrow\; C \; \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \boxed{1} & \boxed{2} & \boxed{0} & \boxed{2} & \boxed{3} \end{array}$$

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

C					
	0	1	2	3	4

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C					

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{|c|c|c|c|c|c|c|c|} \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} \\ \hline 3 & 0 & 4 & 1 & 3 & 4 & 1 & 4 \\ \hline \end{array} \;\Rightarrow\; C \; \begin{array}{|c|c|c|c|c|} \multicolumn{1}{c}{0} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} \\ \hline 1 & 2 & 0 & 2 & 3 \\ \hline \end{array}$$

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

$$C \; \begin{array}{|c|c|c|c|c|} \hline 1 & & & & \\ \hline \multicolumn{1}{c}{0} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1				

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen: A Eingabefeld | C Rechenfeld
B Ausgabefeld | k begrenzt das **Universum:** $\{0, \dots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

C	1	3			
	0	1	2	3	4

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

A

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

$\Rightarrow$ C

0	1	2	3	4
1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

C

0	1	2	3	4
1	3			

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	3	3		

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen: A Eingabefeld | C Rechenfeld

B Ausgabefeld | k begrenzt das **Universum:** $\{0, \dots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	3	3		

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

A

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

$\Rightarrow$ C

0	1	2	3	4
1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

C

0	1	2	3	4
1	3	3	5	

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	3	3	5	

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \dots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	3	3	5	8

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \boxed{3} & \boxed{0} & \boxed{4} & \boxed{1} & \boxed{3} & \boxed{4} & \boxed{1} & \boxed{4} \end{array} \Rightarrow C \; \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \boxed{1} & \boxed{2} & \boxed{0} & \boxed{2} & \boxed{3} \end{array}$$

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

$$C \; \begin{array}{ccccc} \boxed{1} & \boxed{3} & \boxed{3} & \boxed{5} & \boxed{8} \\ \hline 0 & 1 & 2 & 3 & 4 \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \dots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \boxed{3} & \boxed{0} & \boxed{4} & \boxed{1} & \boxed{3} & \boxed{4} & \boxed{1} & \boxed{4} \end{array} \Rightarrow C \; \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \boxed{1} & \boxed{2} & \boxed{0} & \boxed{2} & \boxed{3} \end{array}$$

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

$$C \; \begin{array}{ccccc} \boxed{1} & \boxed{3} & \boxed{3} & \boxed{5} & \boxed{8} \\ \hline 0 & 1 & 2 & 3 & 4 \end{array}$$

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	3	3	5	8

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

	1	2	3	4	5	6	7	8
B								

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	3	3	5	8

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

	1	2	3	4	5	6	7	8
B								

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

$$A \; \begin{array}{c|c|c|c|c|c|c|c} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 3 & 0 & 4 & 1 & 3 & 4 & 1 & 4 \end{array} \;\Rightarrow\; C \; \begin{array}{c|c|c|c|c} 0 & 1 & 2 & 3 & 4 \\ \hline 1 & 2 & 0 & 2 & 3 \end{array}$$

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

$$C \; \begin{array}{c|c|c|c|c} 1 & 3 & 3 & 5 & 8 \\ \hline 0 & 1 & 2 & 3 & 4 \end{array}$$

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

$$B \; \begin{array}{c|c|c|c|c|c|c|c} & & & & & & & \\ \hline 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array}$$

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	3	3	5	8

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

	1	2	3	4	5	6	7	8
B								

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

- A Eingabefeld
- B Ausgabefeld
- C Rechenfeld
- k begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	3	3	5	8

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

	1	2	3	4	5	6	7	8
B								

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	3	3	5	8

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

	1	2	3	4	5	6	7	8
B								4

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	3	3	5	~~8~~

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

	1	2	3	4	5	6	7	8
B								4

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	3	3	5	7

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

	1	2	3	4	5	6	7	8
B								4

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	3	3	5	7

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

	1	2	3	4	5	6	7	8
B								4

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	3	3	5	7

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

	1	2	3	4	5	6	7	8
B								4

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen: A Eingabefeld | C Rechenfeld
B Ausgabefeld | k begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	3	3	5	7

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

	1	2	3	4	5	6	7	8
B								4

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	3	3	5	7

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

	1	2	3	4	5	6	7	8
B			1					4

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	~~3~~	3	5	7

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

	1	2	3	4	5	6	7	8
B			1					4

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	2	3	5	7

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

	1	2	3	4	5	6	7	8
B			1					4

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	2	3	5	7

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

	1	2	3	4	5	6	7	8
B			1					4

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

A:

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

$\Rightarrow$ C:

0	1	2	3	4
1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

C:

0	1	2	3	4
1	2	3	5	7

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

B:

1	2	3	4	5	6	7	8
		1					4

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen: A Eingabefeld | C Rechenfeld

B Ausgabefeld | k begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	2	3	5	7

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

	1	2	3	4	5	6	7	8
B			1					4

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

- A Eingabefeld
- B Ausgabefeld
- C Rechenfeld
- k begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	2	3	5	7

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	2	3	5	$\cancel{7}$

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	2	3	5	6

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	2	3	5	6

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	2	3	5	6

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	2	3	5	6

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	2	3	$\cancel{5}$	6

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	2	3	4	6

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

- A Eingabefeld
- B Ausgabefeld
- C Rechenfeld
- k begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	2	3	4	6

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	2	3	4	6

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

COUNTINGSORT

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	2	3	4	6

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	$\cancel{2}$	3	4	6

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	1	3	4	6

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	1	3	4	6

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	1	3	4	6

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	1	3	4	$\cancel{6}$

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	1	3	4	5

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

- A Eingabefeld
- B Ausgabefeld
- C Rechenfeld
- k begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	1	3	4	5

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen: A Eingabefeld | C Rechenfeld

B Ausgabefeld | k begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	1	1	3	4	5

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen: A Eingabefeld | C Rechenfeld

B Ausgabefeld | k begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	~~1~~	1	3	4	5

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	0	1	3	4	5

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

- A Eingabefeld
- B Ausgabefeld
- C Rechenfeld
- k begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	0	1	3	4	5

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	0	1	3	4	5

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	0	1	3	4	5

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	0	1	3	$\cancel{4}$	5

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	0	1	3	3	5

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	0	1	3	3	5

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

COUNTINGSORT

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	0	1	3	3	5

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

Idee: 1) für jedes x in der Eingabe: **zähle** die Anzahl der Zahlen $\leq x$

2) benutze diese Information um x im Ausgabefeld direkt an die richtige Position zu **schreiben**

Variablen:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das **Universum:** $\{0, \ldots, k\}$

Beispiel:

1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

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

$\Rightarrow$

	0	1	2	3	4
C	1	2	0	2	3

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

	0	1	2	3	4
C	0	1	3	3	5

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort ist stabil!

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld Ausgabefeld beschränkt Universum $\{0, \ldots, k\}$

CountingSort(int[] A, int[] B, int k)

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld
Ausgabefeld
beschränkt Universum $\{0, \ldots, k\}$

CountingSort(int[] A, int[] B, int k)

sei $C[0 \ldots k] = \langle 0, 0, \ldots, 0 \rangle$ ein neues Feld

Initialisierung

1a) Zählen

1b) Berechnen

2) Schreiben

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld | Ausgabefeld | beschränkt Universum $\{0, \ldots, k\}$

CountingSort(int[] A, int[] B, int k)

sei $C[0 \ldots k] = \langle 0, 0, \ldots, 0 \rangle$ ein neues Feld — Initialisierung

for $j = 1$ **to** $A.length$ **do** — 1a) Zählen

// $C[i]$ enthält jetzt die Anzahl der Elemente gleich i in A

1b) Berechnen

2) Schreiben

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld | Ausgabefeld | beschränkt Universum $\{0, \ldots, k\}$

CountingSort(int[] A, int[] B, int k)

sei $C[0 \ldots k] = \langle 0, 0, \ldots, 0 \rangle$ ein neues Feld — Initialisierung

for $j = 1$ **to** $A.length$ **do** — 1a) Zählen

// $C[i]$ enthält jetzt die Anzahl der Elemente gleich i in A

for $i = 1$ **to** k **do** — 1b) Berechnen

// $C[i]$ enthält jetzt die Anzahl der Elemente $\leq i$ in A

2) Schreiben

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld | Ausgabefeld | beschränkt Universum $\{0, \ldots, k\}$

CountingSort(int[] A, int[] B, int k)

sei $C[0 \ldots k] = \langle 0, 0, \ldots, 0 \rangle$ ein neues Feld — Initialisierung

for $j = 1$ **to** $A.length$ **do** — 1a) Zählen

// $C[i]$ enthält jetzt die Anzahl der Elemente gleich i in A

for $i = 1$ **to** k **do** — 1b) Berechnen

// $C[i]$ enthält jetzt die Anzahl der Elemente $\leq i$ in A

for $j = A.length$ **downto** 1 **do** — 2) Schreiben

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld | Ausgabefeld | beschränkt Universum $\{0, \ldots, k\}$

$\textsc{CountingSort}(\text{int}[]\ A, \text{int}[]\ B, \text{int}\ k)$

sei $C[0 \ldots k] = \langle 0, 0, \ldots, 0 \rangle$ ein neues Feld

for $j = 1$ **to** $A.\mathit{length}$ **do** — 1a) Zählen

// $C[i]$ enthält jetzt die Anzahl der Elemente gleich i in A

for $i = 1$ **to** k **do** — 1b) Berechnen

// $C[i]$ enthält jetzt die Anzahl der Elemente $\leq i$ in A

for $j = A.\mathit{length}$ **downto** 1 **do**

2) Schreiben

Aufgabe.

Füllen Sie die Felder mit Code, der obige Idee umsetzt!

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld | Ausgabefeld | beschränkt Universum $\{0, \ldots, k\}$

```
CountingSort(int[] A, int[] B, int k)
  sei C[0...k] = ⟨0, 0, ..., 0⟩ ein neues Feld
  for j = 1 to A.length do C[A[j]] = C[A[j]] + 1          1a) Zählen
  // C[i] enthält jetzt die Anzahl der Elemente gleich i in A
  for i = 1 to k do ____________                          1b) Berechnen
  // C[i] enthält jetzt die Anzahl der Elemente ≤ i in A
  for j = A.length downto 1 do
      ____________                                         2) Schreiben
```

Aufgabe.
Füllen Sie die Felder mit Code, der obige Idee umsetzt!

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld Ausgabefeld beschränkt Universum $\{0, \ldots, k\}$

CountingSort(int[] A, int[] B, int k)

sei $C[0 \ldots k] = \langle 0, 0, \ldots, 0 \rangle$ ein neues Feld

for $j = 1$ **to** $A.length$ **do** $C[A[j]] = C[A[j]] + 1$ — 1a) Zählen

// $C[i]$ enthält jetzt die Anzahl der Elemente gleich i in A

for $i = 1$ **to** k **do** $C[i] = C[i] + C[i-1]$ — 1b) Berechnen

// $C[i]$ enthält jetzt die Anzahl der Elemente $\leq i$ in A

for $j = A.length$ **downto** 1 **do**

2) Schreiben

Aufgabe.

Füllen Sie die Felder mit Code, der obige Idee umsetzt!

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld | Ausgabefeld | beschränkt Universum $\{0, \ldots, k\}$

CountingSort(int[] A, int[] B, int k)
- sei $C[0 \ldots k] = \langle 0, 0, \ldots, 0 \rangle$ ein neues Feld
- **for** $j = 1$ **to** $A.length$ **do** $C[A[j]] = C[A[j]] + 1$ — 1a) Zählen
- // $C[i]$ enthält jetzt die Anzahl der Elemente gleich i in A
- **for** $i = 1$ **to** k **do** $C[i] = C[i] + C[i-1]$ — 1b) Berechnen
- // $C[i]$ enthält jetzt die Anzahl der Elemente $\leq i$ in A
- **for** $j = A.length$ **downto** 1 **do**
  - $B[C[A[j]]] = A[j]$
  - 2) Schreiben

Aufgabe.
Füllen Sie die Felder mit Code, der obige Idee umsetzt!

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld

Ausgabefeld

beschränkt Universum $\{0, \ldots, k\}$

CountingSort(int[] A, int[] B, int k)

sei $C[0 \ldots k] = \langle 0, 0, \ldots, 0 \rangle$ ein neues Feld

for $j = 1$ **to** $A.length$ **do** $C[A[j]] = C[A[j]] + 1$ — 1a) Zählen

// $C[i]$ enthält jetzt die Anzahl der Elemente gleich i in A

for $i = 1$ **to** k **do** $C[i] = C[i] + C[i-1]$ — 1b) Berechnen

// $C[i]$ enthält jetzt die Anzahl der Elemente $\leq i$ in A

for $j = A.length$ **downto** 1 **do**

  $B[C[A[j]]] = A[j]$

  $C[A[j]] = C[A[j]] - 1$ — 2) Schreiben

Aufgabe.

Füllen Sie die Felder mit Code, der obige Idee umsetzt!

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld | Ausgabefeld | beschränkt Universum $\{0, \ldots, k\}$

CountingSort(int[] A, int[] B, int k)

sei $C[0 \ldots k] = \langle 0, 0, \ldots, 0 \rangle$ ein neues Feld

for $j = 1$ **to** $A.length$ **do** $C[A[j]] = C[A[j]] + 1$ — 1a) Zählen

// $C[i]$ enthält jetzt die Anzahl der Elemente gleich i in A

for $i = 1$ **to** k **do** $C[i] = C[i] + C[i-1]$ — 1b) Berechnen

// $C[i]$ enthält jetzt die Anzahl der Elemente $\leq i$ in A

for $j = A.length$ **downto** 1 **do**

$B[C[A[j]]] = A[j]$

$C[A[j]] = C[A[j]] - 1$

Aufgabe.

Füllen Sie die Felder mit Code, der obige Idee umsetzt!

Demo.

https://algo.uni-trier.de/demos/sort.html

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld | Ausgabefeld | beschränkt Universum $\{0, \ldots, k\}$

```
CountingSort(int[] A, int[] B, int k)
  sei C[0...k] = ⟨0, 0, ..., 0⟩ ein neues Feld                  Initialisierung
  for j = 1 to A.length do C[A[j]] = C[A[j]] + 1                1a) Zählen
  // C[i] enthält jetzt die Anzahl der Elemente gleich i in A
  for i = 1 to k do C[i] = C[i] + C[i − 1]                      1b) Berechnen
  // C[i] enthält jetzt die Anzahl der Elemente ≤ i in A
  for j = A.length downto 1 do
      B[C[A[j]]] = A[j]                                         2) Schreiben
      C[A[j]] = C[A[j]] − 1
```

Laufzeit:

$\mathcal{O}(\quad)$

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld | Ausgabefeld | beschränkt Universum $\{0, \dots, k\}$

```
CountingSort(int[] A, int[] B, int k)
  sei C[0...k] = ⟨0, 0, ..., 0⟩ ein neues Feld            Initialisierung
  for j = 1 to A.length do C[A[j]] = C[A[j]] + 1          1a) Zählen
  // C[i] enthält jetzt die Anzahl der Elemente gleich i in A
  for i = 1 to k do C[i] = C[i] + C[i − 1]                1b) Berechnen
  // C[i] enthält jetzt die Anzahl der Elemente ≤ i in A
  for j = A.length downto 1 do
      B[C[A[j]]] = A[j]                                   2) Schreiben
      C[A[j]] = C[A[j]] − 1
```

Laufzeit:
$\mathcal{O}(\quad)$

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld | Ausgabefeld | beschränkt Universum $\{0, \ldots, k\}$

```
CountingSort(int[] A, int[] B, int k)
  sei C[0...k] = ⟨0, 0, ..., 0⟩ ein neues Feld                  Initialisierung
  for j = 1 to A.length do C[A[j]] = C[A[j]] + 1                  1a) Zählen
  // C[i] enthält jetzt die Anzahl der Elemente gleich i in A
  for i = 1 to k do C[i] = C[i] + C[i − 1]                        1b) Berechnen
  // C[i] enthält jetzt die Anzahl der Elemente ≤ i in A
  for j = A.length downto 1 do
      B[C[A[j]]] = A[j]                                           2) Schreiben
      C[A[j]] = C[A[j]] − 1
```

Laufzeit:
$\mathcal{O}(\quad k\,)$

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld | Ausgabefeld | beschränkt Universum $\{0, \dots, k\}$

CountingSort(int[] A, int[] B, int k)

sei $C[0 \dots k] = \langle 0, 0, \dots, 0 \rangle$ ein neues Feld — Initialisierung

for $j = 1$ **to** $A.length$ **do** $C[A[j]] = C[A[j]] + 1$ — 1a) Zählen

// $C[i]$ enthält jetzt die Anzahl der Elemente gleich i in A

for $i = 1$ **to** k **do** $C[i] = C[i] + C[i-1]$ — 1b) Berechnen

// $C[i]$ enthält jetzt die Anzahl der Elemente $\leq i$ in A

for $j = A.length$ **downto** 1 **do**

- $B[C[A[j]]] = A[j]$
- $C[A[j]] = C[A[j]] - 1$

2) Schreiben

Laufzeit: $\mathcal{O}(\quad k\ )$

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld

Ausgabefeld

beschränkt Universum $\{0, \ldots, k\}$

CountingSort(int[] A, int[] B, int k)

sei $C[0 \ldots k] = \langle 0, 0, \ldots, 0 \rangle$ ein neues Feld — Initialisierung

for $j = 1$ **to** $A.length$ **do** $C[A[j]] = C[A[j]] + 1$ — 1a) Zählen

// $C[i]$ enthält jetzt die Anzahl der Elemente gleich i in A

for $i = 1$ **to** k **do** $C[i] = C[i] + C[i-1]$ — 1b) Berechnen

// $C[i]$ enthält jetzt die Anzahl der Elemente $\leq i$ in A

for $j = A.length$ **downto** 1 **do**

$B[C[A[j]]] = A[j]$

$C[A[j]] = C[A[j]] - 1$ — 2) Schreiben

Laufzeit:

$\mathcal{O}(n + k)$

COUNTINGSORT

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld | Ausgabefeld | beschränkt Universum $\{0, \ldots, k\}$

```
CountingSort(int[] A, int[] B, int k)
  sei C[0...k] = ⟨0, 0, ..., 0⟩ ein neues Feld                    Initialisierung
  for j = 1 to A.length do C[A[j]] = C[A[j]] + 1                   1a) Zählen
  // C[i] enthält jetzt die Anzahl der Elemente gleich i in A
  for i = 1 to k do C[i] = C[i] + C[i − 1]                         1b) Berechnen
  // C[i] enthält jetzt die Anzahl der Elemente ≤ i in A
  for j = A.length downto 1 do
      B[C[A[j]]] = A[j]
      C[A[j]] = C[A[j]] − 1                                        2) Schreiben
```

Laufzeit: $\mathcal{O}(n + k)$

CountingSort

Plan: 1a) **Zählen:** Für jedes x in A, zähle die Anzahl der Zahlen gleich x (in C)

1b) **Berechnen:** Für jedes x in A, berechne die Anzahl der Zahlen $\leq x$ (in C)

2) **Schreiben:** Schreibe jedes x in A direkt an die richtige Position in B

Eingabefeld

Ausgabefeld

beschränkt Universum $\{0, \ldots, k\}$

CountingSort(int[] A, int[] B, int k)

sei $C[0 \ldots k] = \langle 0, 0, \ldots, 0 \rangle$ ein neues Feld — Initialisierung

for $j = 1$ **to** $A.length$ **do** $C[A[j]] = C[A[j]] + 1$ — 1a) Zählen

// $C[i]$ enthält jetzt die Anzahl der Elemente gleich i in A

for $i = 1$ **to** k **do** $C[i] = C[i] + C[i-1]$ — 1b) Berechnen

// $C[i]$ enthält jetzt die Anzahl der Elemente $\leq i$ in A

for $j = A.length$ **downto** 1 **do** — 2) Schreiben

$B[C[A[j]]] = A[j]$

$C[A[j]] = C[A[j]] - 1$

Laufzeit: $\mathcal{O}(n + k)$

Zusammenfassung

- Jedes vergleichsbasierte Sortierverfahren braucht im Worst-Case $\Omega(n \log n)$ Vergleiche.

- CountingSort sortiert Zahlen in $\{0, \ldots, k\}$ **(stabil!)**
  Laufzeit für n Zahlen: $\mathcal{O}(n + k)$

- RadixSort sortiert s-stellige b-adische Zahlen

- BucketSort sortiert gleichverteilte zufällige Zahlen

Zusammenfassung

- Jedes vergleichsbasierte Sortierverfahren braucht im Worst-Case $\Omega(n \log n)$ Vergleiche.

- CountingSort sortiert Zahlen in $\{0, \ldots, k\}$ (**stabil!**)
  Laufzeit für n Zahlen: $\mathcal{O}(n + k)$

- RadixSort sortiert s-stellige b-adische Zahlen

- BucketSort sortiert gleichverteilte zufällige Zahlen

Zusammenfassung

- Jedes vergleichsbasierte Sortierverfahren braucht im Worst-Case $\Omega(n \log n)$ Vergleiche.

- CountingSort sortiert Zahlen in $\{0, \ldots, k\}$ (**stabil!**)
  Laufzeit für n Zahlen: $\mathcal{O}(n + k)$

- RadixSort sortiert s-stellige b-adische Zahlen
  max. s Stellen

- BucketSort sortiert gleichverteilte zufällige Zahlen

Zusammenfassung

- Jedes vergleichsbasierte Sortierverfahren braucht im Worst-Case $\Omega(n \log n)$ Vergleiche.

- CountingSort sortiert Zahlen in $\{0, \ldots, k\}$ (**stabil!**)
  Laufzeit für n Zahlen: $\mathcal{O}(n + k)$

- RadixSort sortiert s-stellige b-adische Zahlen
  max. s Stellen
  b mögliche unterschiedliche Ziffern

- BucketSort sortiert gleichverteilte zufällige Zahlen

Zusammenfassung

- Jedes vergleichsbasierte Sortierverfahren braucht im Worst-Case $\Omega(n \log n)$ Vergleiche.

- COUNTINGSORT sortiert Zahlen in $\{0, \ldots, k\}$ (**stabil!**)
  Laufzeit für n Zahlen: $\mathcal{O}(n + k)$

- RADIXSORT sortiert s-stellige b-adische Zahlen

  max. s Stellen

  b mögliche unterschiedliche Ziffern

  z.B. Dezimalzahl: $b = 10$

- BUCKETSORT sortiert gleichverteilte zufällige Zahlen

Zusammenfassung

- Jedes vergleichsbasierte Sortierverfahren braucht im Worst-Case $\Omega(n \log n)$ Vergleiche.

- CountingSort sortiert Zahlen in $\{0, \ldots, k\}$ (**stabil!**)
  Laufzeit für n Zahlen: $\mathcal{O}(n + k)$

- RadixSort sortiert s-stellige b-adische Zahlen
  max. s Stellen
  b mögliche unterschiedliche Ziffern
  z.B. Dezimalzahl: $b = 10$
  Binärzahl: $b = 2$

- BucketSort sortiert gleichverteilte zufällige Zahlen

Zusammenfassung

- Jedes vergleichsbasierte Sortierverfahren braucht im Worst-Case $\Omega(n \log n)$ Vergleiche.

- CountingSort sortiert Zahlen in $\{0, \ldots, k\}$ (**stabil!**)
  Laufzeit für n Zahlen: $\mathcal{O}(n + k)$

max. s Stellen

b mögliche unterschiedliche Ziffern

- RadixSort sortiert s-stellige b-adische Zahlen

z.B. Dezimalzahl: $b = 10$
Binärzahl: $b = 2$
Wörter: $b = 26$

- BucketSort sortiert gleichverteilte zufällige Zahlen

RadixSort

Frage: Gegeben Liste von Menschen mit deren Geburtstagen.
Wie würden Sie die Liste nach Alter sortieren?

RadixSort

(Jahr, Monat, Tag)

Frage: Gegeben Liste von Menschen mit deren Geburtstagen.
Wie würden Sie die Liste nach Alter sortieren?

RadixSort

(Jahr, Monat, Tag)

Frage: Gegeben Liste von Menschen mit deren Geburtstagen.
Wie würden Sie die Liste nach Alter sortieren?

Drei (?) Lösungen:

RadixSort

(Jahr, Monat, Tag)

Frage: Gegeben Liste von Menschen mit deren Geburtstagen.
Wie würden Sie die Liste nach Alter sortieren?

Drei (?) Lösungen:

- Geburtstage in Anzahl Tage seit 1.1.1970 umrechnen,
  dann vergleichsbasiertes Sortierverfahren verwenden.

RadixSort

(Jahr, Monat, Tag)

Frage: Gegeben Liste von Menschen mit deren Geburtstagen.
Wie würden Sie die Liste nach Alter sortieren?

Drei (?) Lösungen:

- Geburtstage in Anzahl Tage seit 1.1.1970 umrechnen, dann vergleichsbasiertes Sortierverfahren verwenden.
- Spezielle Vergleichsroutine schreiben und in vergleichsbasiertes Sortierverfahren einbauen.

RadixSort

(Jahr, Monat, Tag)

Frage: Gegeben Liste von Menschen mit deren Geburtstagen.
Wie würden Sie die Liste nach Alter sortieren?

Drei (?) Lösungen:

- Geburtstage in Anzahl Tage seit 1.1.1970 umrechnen, dann vergleichsbasiertes Sortierverfahren verwenden.
- Spezielle Vergleichsroutine schreiben und in vergleichsbasiertes Sortierverfahren einbauen.
- Liste $3\times$ sortieren: je $1\times$ nach Jahr, Monat, Tag.

RadixSort

(Jahr, Monat, Tag)

Frage: Gegeben Liste von Menschen mit deren Geburtstagen.
Wie würden Sie die Liste nach Alter sortieren?

Drei (?) Lösungen:

- Geburtstage in Anzahl Tage seit 1.1.1970 umrechnen, dann vergleichsbasiertes Sortierverfahren verwenden.
- Spezielle Vergleichsroutine schreiben und in vergleichsbasiertes Sortierverfahren einbauen.
- Liste $3\times$ sortieren: je $1\times$ nach Jahr, Monat, Tag.

RadixSort(A, s)

RadixSort

(Jahr, Monat, Tag)

Frage: Gegeben Liste von Menschen mit deren Geburtstagen.
Wie würden Sie die Liste nach Alter sortieren?

Drei (?) Lösungen:

- Geburtstage in Anzahl Tage seit 1.1.1970 umrechnen, dann vergleichsbasiertes Sortierverfahren verwenden.
- Spezielle Vergleichsroutine schreiben und in vergleichsbasiertes Sortierverfahren einbauen.
- Liste $3\times$ sortieren: je $1\times$ nach Jahr, Monat, Tag.

Aber in welcher Reihenfolge?

Anz. Stellen (hier: 3)

RadixSort(A, s)

RadixSort

(Jahr, Monat, Tag)

Frage: Gegeben Liste von Menschen mit deren Geburtstagen.
Wie würden Sie die Liste nach Alter sortieren?

Drei (?) Lösungen:

- Geburtstage in Anzahl Tage seit 1.1.1970 umrechnen,
dann vergleichsbasiertes Sortierverfahren verwenden.
- Spezielle Vergleichsroutine schreiben und
in vergleichsbasiertes Sortierverfahren einbauen.
- Liste $3\times$ sortieren: je $1\times$ nach Jahr, Monat, Tag.

Aber in welcher Reihenfolge?

Anz. Stellen (hier: 3)

RadixSort(A, s)

for $i = 1$ **to** s **do**
  sortiere A **stabil** nach der i-ten Stelle

RadixSort

(Jahr, Monat, Tag)

Frage: Gegeben Liste von Menschen mit deren Geburtstagen.
Wie würden Sie die Liste nach Alter sortieren?

Drei (?) Lösungen:

- Geburtstage in Anzahl Tage seit 1.1.1970 umrechnen, dann vergleichsbasiertes Sortierverfahren verwenden.
- Spezielle Vergleichsroutine schreiben und in vergleichsbasiertes Sortierverfahren einbauen.
- Liste $3\times$ sortieren: je $1\times$ nach Jahr, Monat, Tag.

Aber in welcher Reihenfolge?

Anz. Stellen (hier: 3)

RadixSort(A, s)

for $i = 1$ **to** s **do**
  sortiere A **stabil** nach der i-ten Stelle

1 = Index der **niederwertigsten** (!) Stelle

RadixSort

(Jahr, Monat, Tag)

Frage: Gegeben Liste von Menschen mit deren Geburtstagen.
Wie würden Sie die Liste nach Alter sortieren?

Drei (?) Lösungen:

- Geburtstage in Anzahl Tage seit 1.1.1970 umrechnen, dann vergleichsbasiertes Sortierverfahren verwenden.
- Spezielle Vergleichsroutine schreiben und in vergleichsbasiertes Sortierverfahren einbauen.
- Liste $3\times$ sortieren: je $1\times$ nach Jahr, Monat, Tag.

Aber in welcher Reihenfolge?

Anz. Stellen (hier: 3)

RadixSort(A, s)

for $i = 1$ **to** s **do**
 sortiere A **stabil** nach der i-ten Stelle

1 = Index der **niederwertigsten** (!) Stelle

z.B. mit CountingSort

RadixSort

(Jahr, Monat, Tag)

Frage: Gegeben Liste von Menschen mit deren Geburtstagen.
Wie würden Sie die Liste nach Alter sortieren?

Drei (?) Lösungen:

- Geburtstage in Anzahl Tage seit 1.1.1970 umrechnen, dann vergleichsbasiertes Sortierverfahren verwenden.
- Spezielle Vergleichsroutine schreiben und in vergleichsbasiertes Sortierverfahren einbauen.
- Liste $3\times$ sortieren: je $1\times$ nach Jahr, Monat, Tag.

Aber in welcher Reihenfolge?

Anz. Stellen (hier: 3)

RadixSort(A, s)

for $i = 1$ **to** s **do**
  sortiere A **stabil** nach der i-ten Stelle

1 = Index der **niederwertigsten** (!) Stelle

z.B. mit CountingSort

RadixSort

(Jahr, Monat, Tag)

Frage: Gegeben Liste von Menschen mit deren Geburtstagen.
Wie würden Sie die Liste nach Alter sortieren?

Drei (?) Lösungen:

- Geburtstage in Anzahl Tage seit 1.1.1970 umrechnen, dann vergleichsbasiertes Sortierverfahren verwenden.
- Spezielle Vergleichsroutine schreiben und in vergleichsbasiertes Sortierverfahren einbauen.
- Liste 3× sortieren: je 1× nach Jahr, Monat, Tag. Aber in welcher Reihenfolge?

Anz. Stellen (hier: 3)

RadixSort(A, s)

for $i = 1$ **to** s **do** 1 = Index der **niederwertigsten** (!) Stelle
  sortiere A **stabil** nach der i-ten Stelle

z.B. mit CountingSort

Laufzeit?

Beispiel

Sortiere A

25	13	31	23	11	37	15

$\textsc{RadixSort}(A, s)$

for $i = 1$ **to** s **do**
  sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere A

25	13	31	23	11	37	15

Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

RadixSort(A, s)

for $i = 1$ **to** s **do**
∟ sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere A

25	13	31	23	11	37	15

A_1

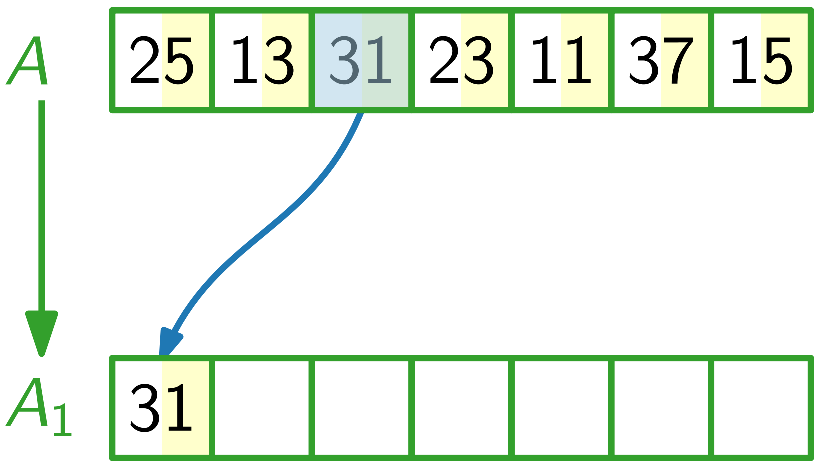
Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

RadixSort(A, s)

for $i = 1$ **to** s **do**
  sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



Gemäß RADIXSORT erst nach Einern, dann (stabil) nach Zehnern.

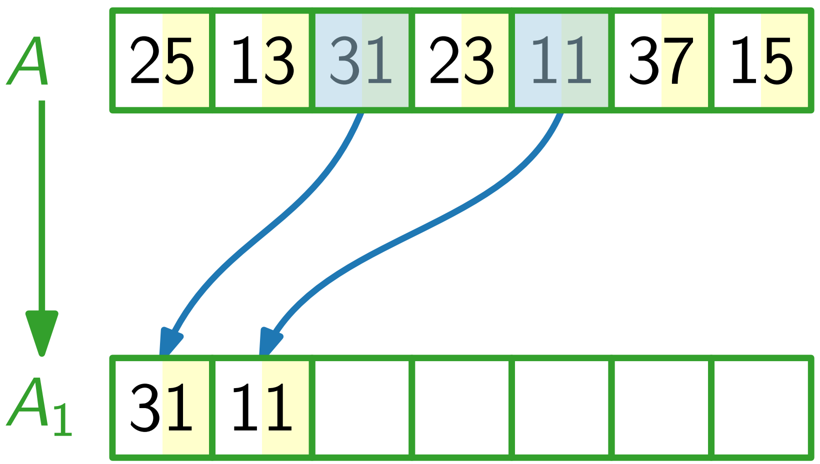
RADIXSORT(A, s)

for $i = 1$ **to** s **do**

sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



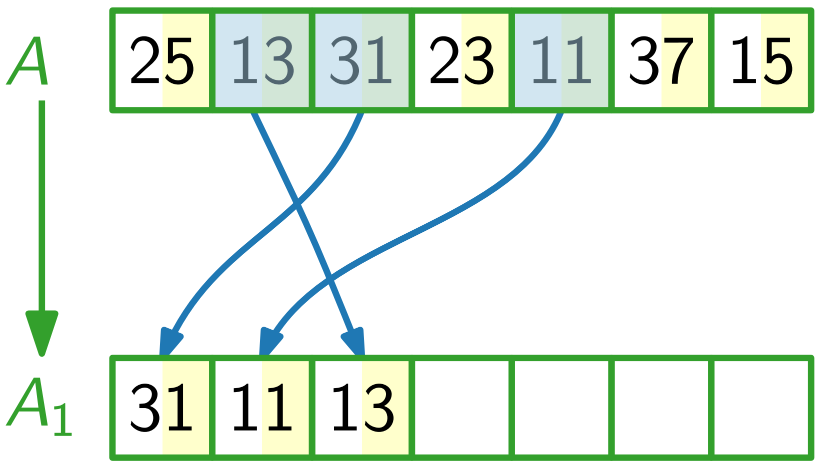
Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

RadixSort(A, s)

for $i = 1$ **to** s **do**
 └ sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



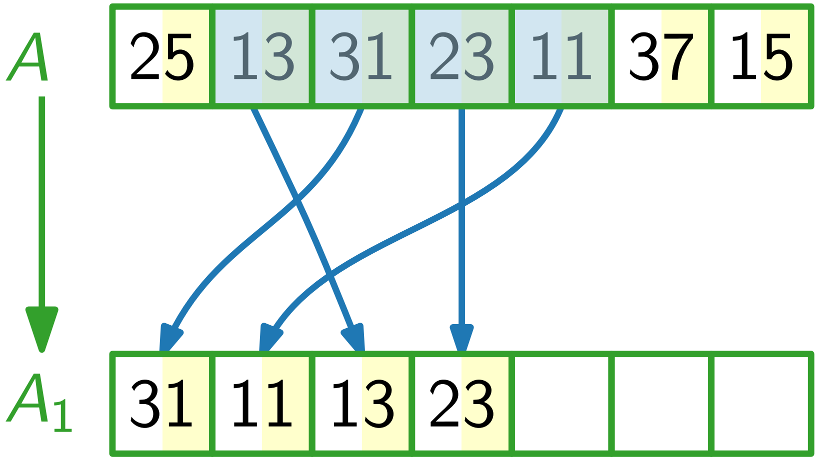
Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

RadixSort(A, s)

for $i = 1$ **to** s **do**
  sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



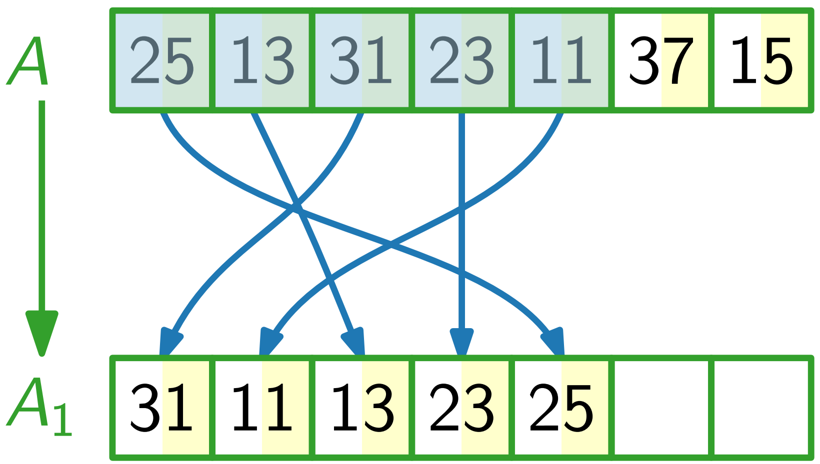
Gemäß RADIXSORT erst nach Einern, dann (stabil) nach Zehnern.

RADIXSORT(A, s)

for $i = 1$ **to** s **do**
  sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



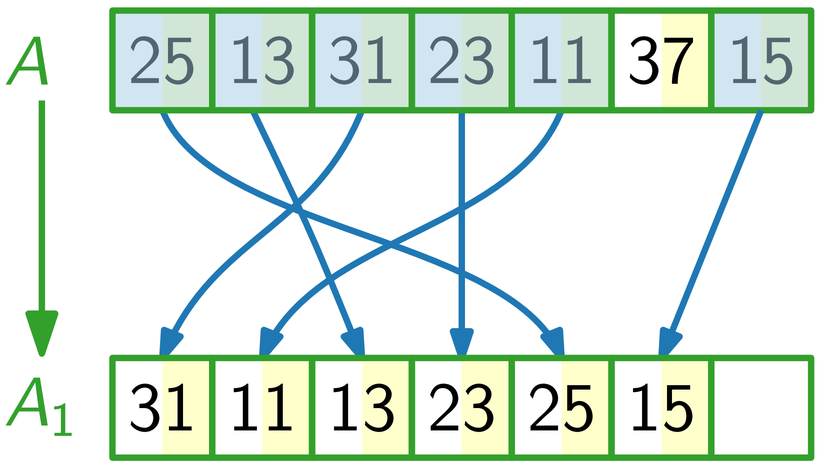
Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

RadixSort(A, s)

for $i = 1$ **to** s **do**
 sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



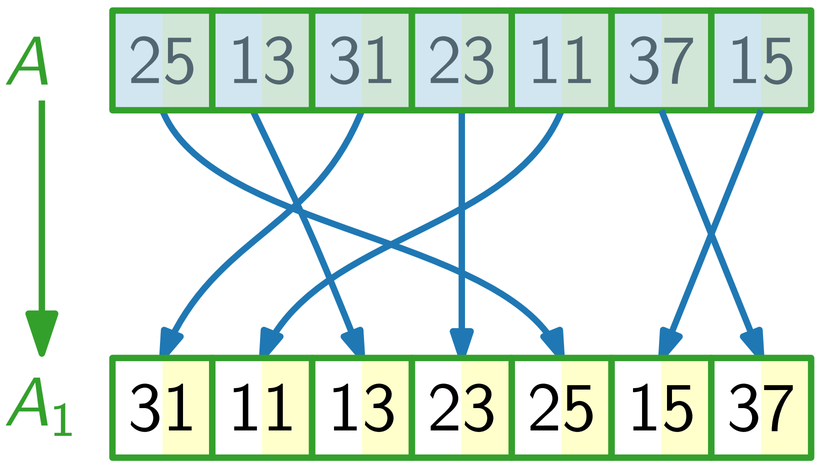
Gemäß RADIXSORT erst nach Einern, dann (stabil) nach Zehnern.

RADIXSORT(A, s)

for $i = 1$ **to** s **do**
⌊ sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

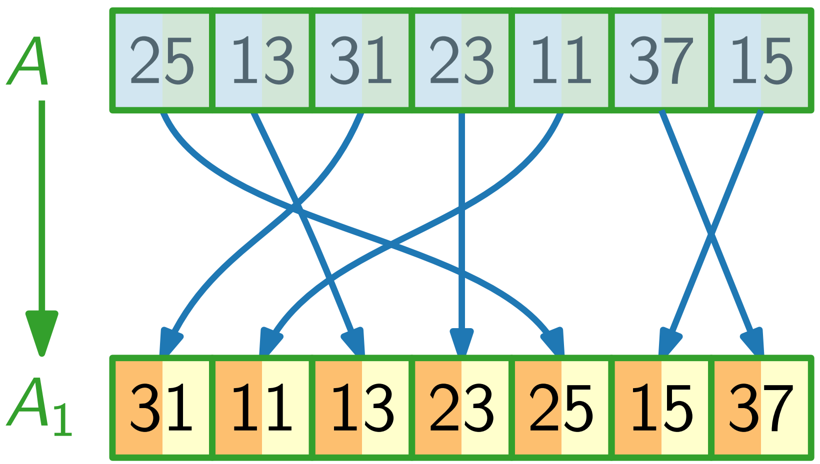
RadixSort(A, s)

for $i = 1$ **to** s **do**

└ sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



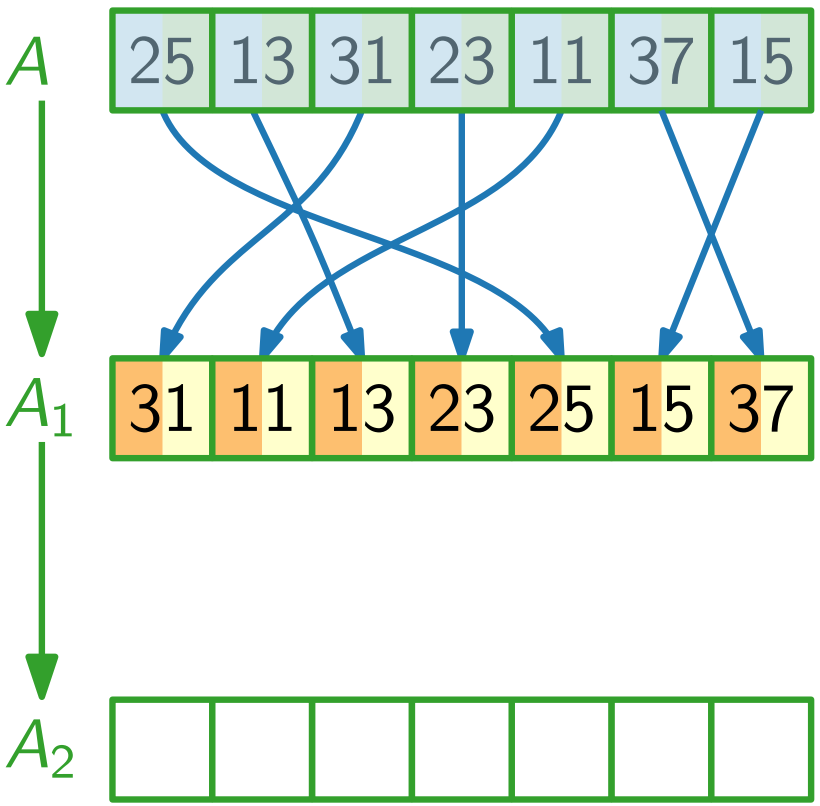
Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

RadixSort(A, s)

for $i = 1$ **to** s **do**
sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



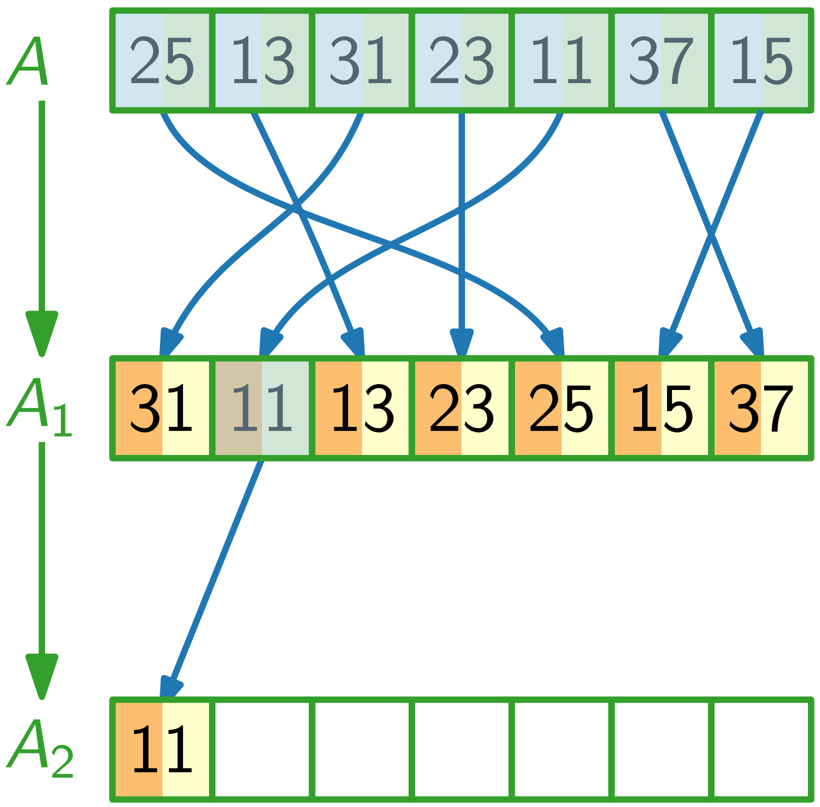
Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

RadixSort(A, s)

for $i = 1$ **to** s **do**
∟ sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



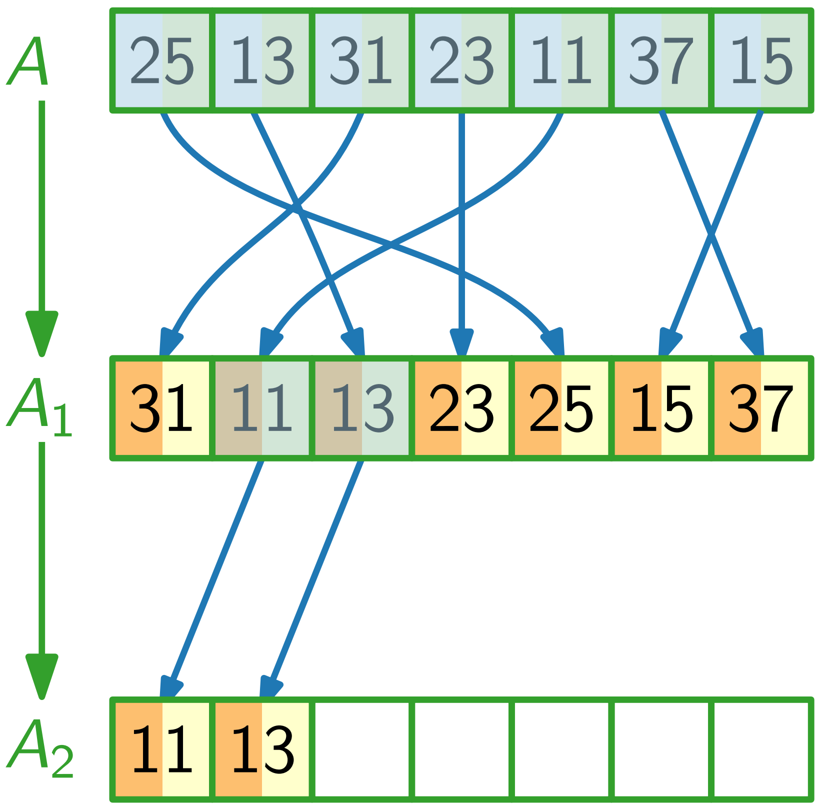
Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

RadixSort(A, s)

for $i = 1$ **to** s **do**
 ∟ sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



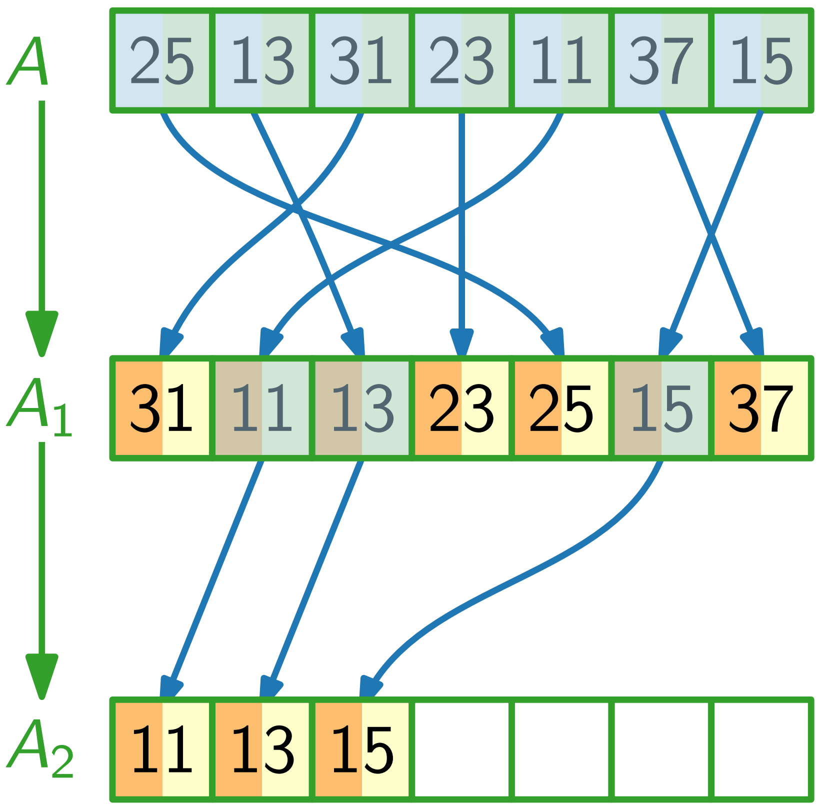
Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

RadixSort(A, s)

for $i = 1$ **to** s **do**
└ sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



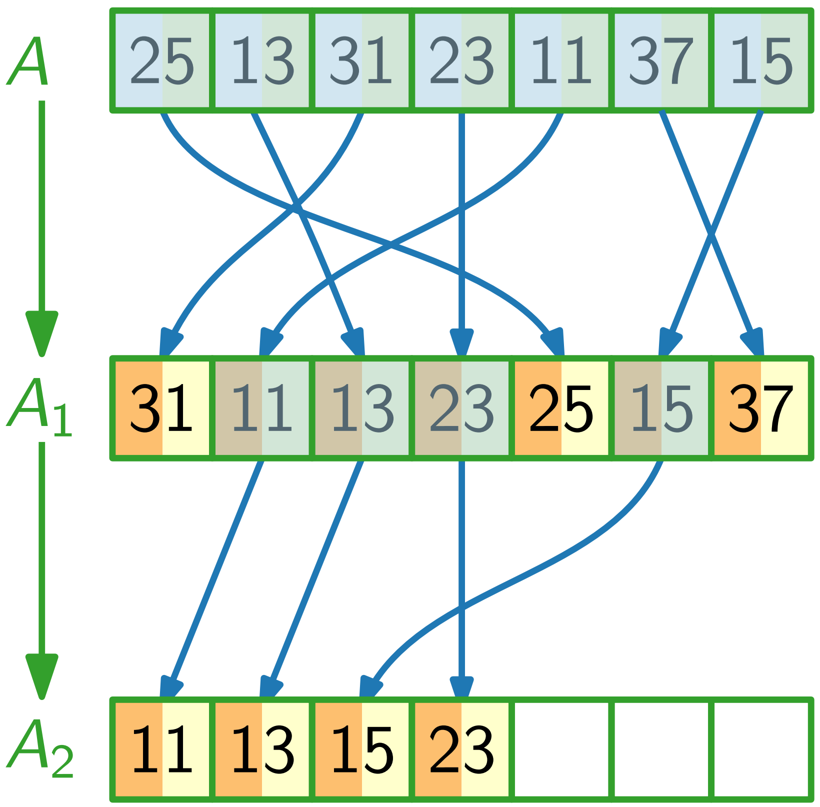
Gemäß RADIXSORT erst nach Einern, dann (stabil) nach Zehnern.

RADIXSORT(A, s)

for $i = 1$ **to** s **do**
∟ sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



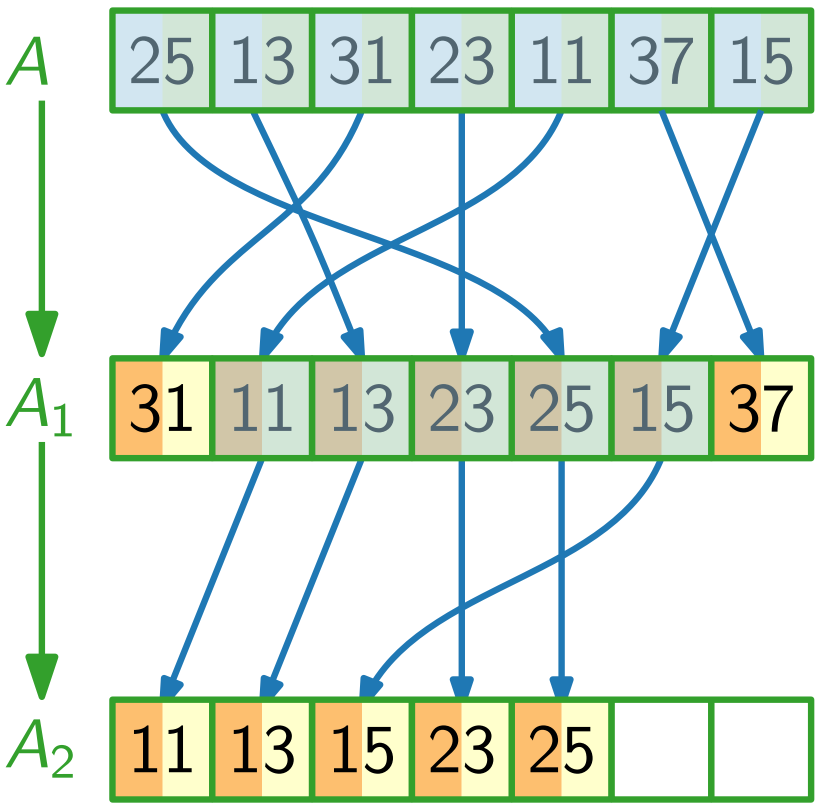
Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

RadixSort(A, s)

for $i = 1$ **to** s **do**
 sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



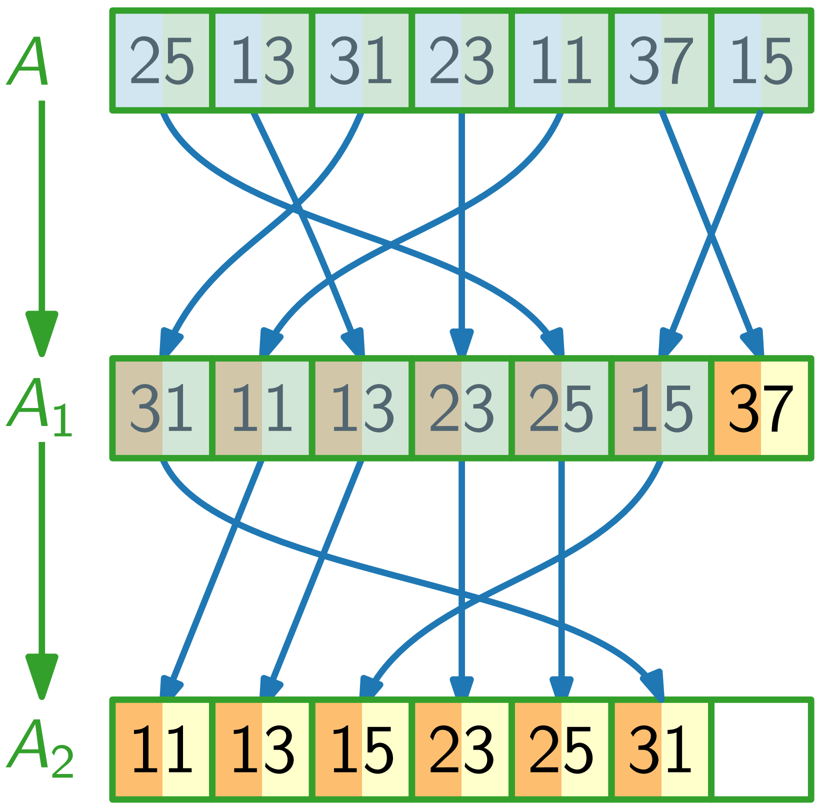
Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

RadixSort(A, s)

for $i = 1$ **to** s **do**
 sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



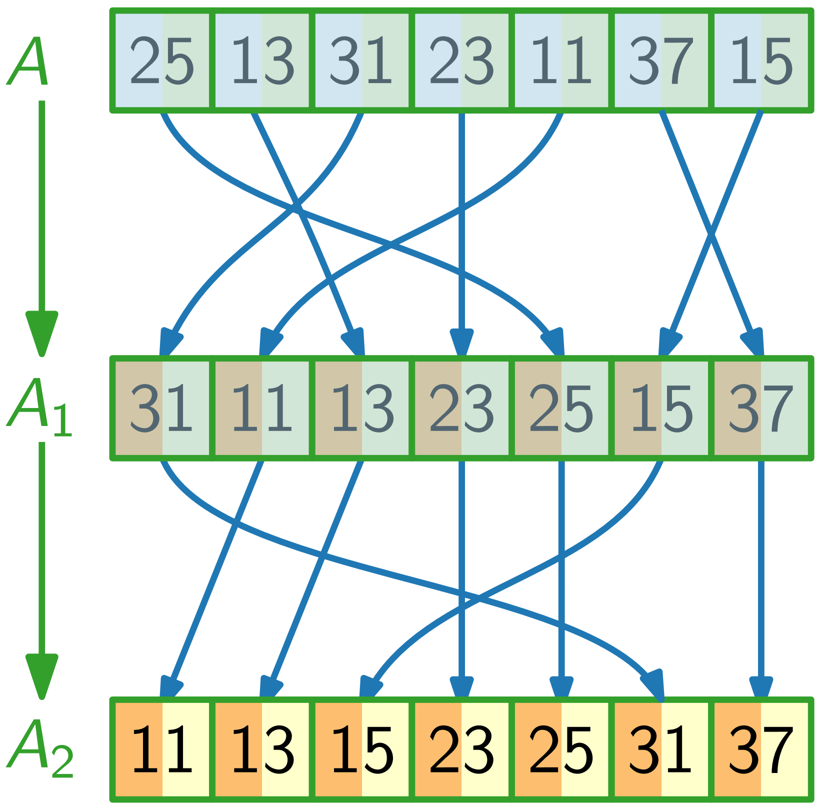
Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

RadixSort(A, s)

for $i = 1$ **to** s **do**
sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



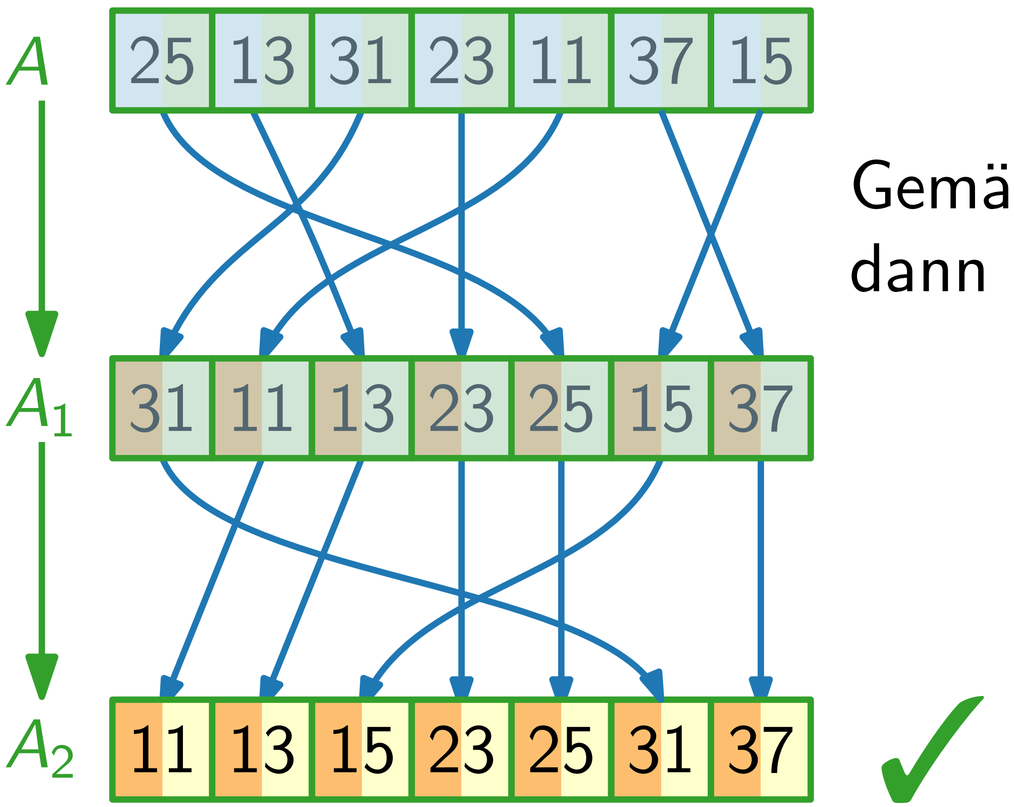
Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

RadixSort(A, s)

for $i = 1$ **to** s **do**
  sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



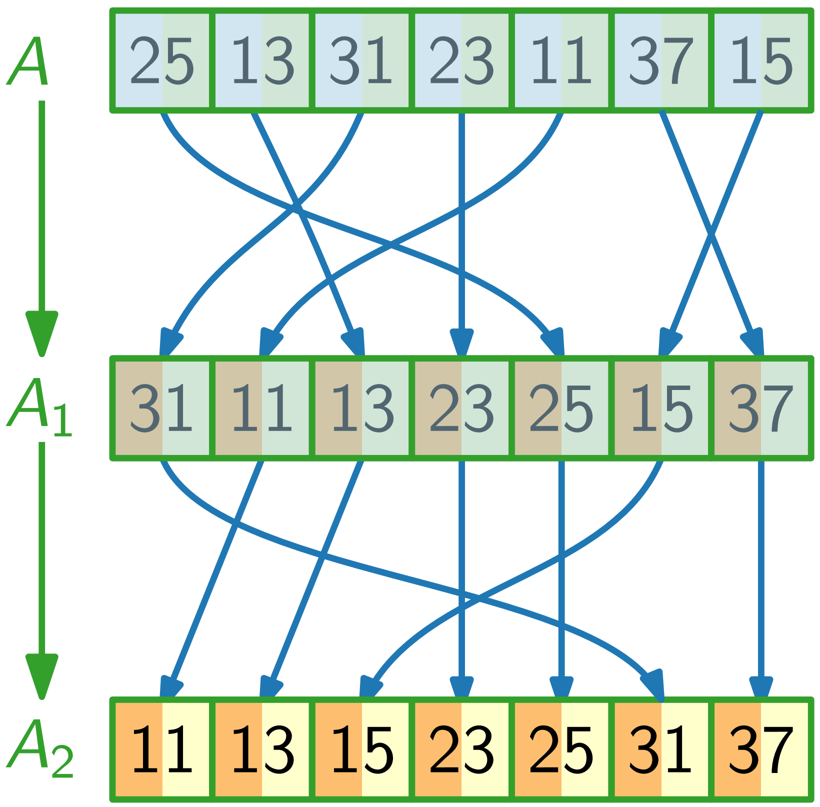
Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

RadixSort(A, s)

for $i = 1$ **to** s **do**
└ sortiere A **stabil** nach der i-ten Stelle

Beispiel

Sortiere



Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

Demo.
`https://algo.uni-trier.de/demos/sort.html`

✓

RadixSort(A, s)

for $i = 1$ **to** s **do**
∟ sortiere A **stabil** nach der i-ten Stelle

Zusammenfassung

- Jedes vergleichsbasierte Sortierverfahren braucht im schlechtesten Fall $\Omega(n \log n)$ Vergleiche.

- CountingSort sortiert Zahlen in $\{0, \dots, k\}$ **(stabil!)**

  Laufzeit für n Zahlen: $\mathcal{O}(n + k)$

- RadixSort sortiert s-stellige b-adische Zahlen

  max. s Stellen

  b mögliche unterschiedliche Ziffern

  Laufzeit für n Zahlen: $\mathcal{O}(s \cdot (n + b))$

  z.B. Dezimalzahl: $b = 10$
  Binärzahl: $b = 2$
  Wörter: $b = 26$

- BucketSort sortiert gleichverteilte zufällige Zahlen

Zusammenfassung

- Jedes vergleichsbasierte Sortierverfahren braucht im schlechtesten Fall $\Omega(n \log n)$ Vergleiche.

- CountingSort sortiert Zahlen in $\{0, \ldots, k\}$ **(stabil!)**

  Laufzeit für n Zahlen: $\mathcal{O}(n + k)$

- RadixSort sortiert s-stellige b-adische Zahlen

  max. s Stellen

  b mögliche unterschiedliche Ziffern

  Laufzeit für n Zahlen: $\mathcal{O}(s \cdot (n + b))$

  z.B. Dezimalzahl: $b = 10$
  Binärzahl: $b = 2$
  Wörter: $b = 26$

- BucketSort sortiert gleichverteilte zufällige Zahlen

BucketSort

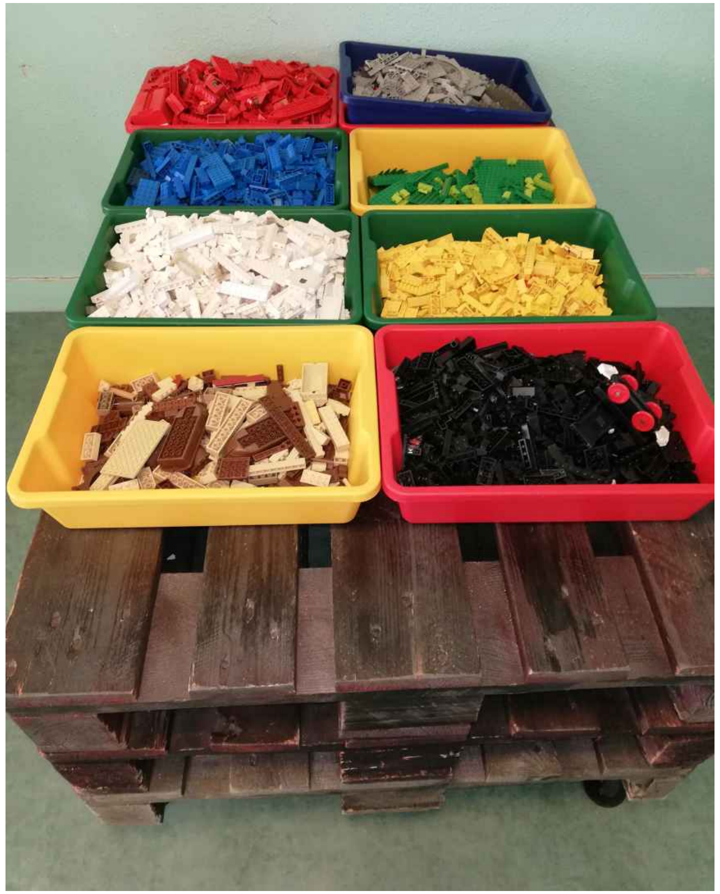


photo by JulienFou on reddit

BucketSort

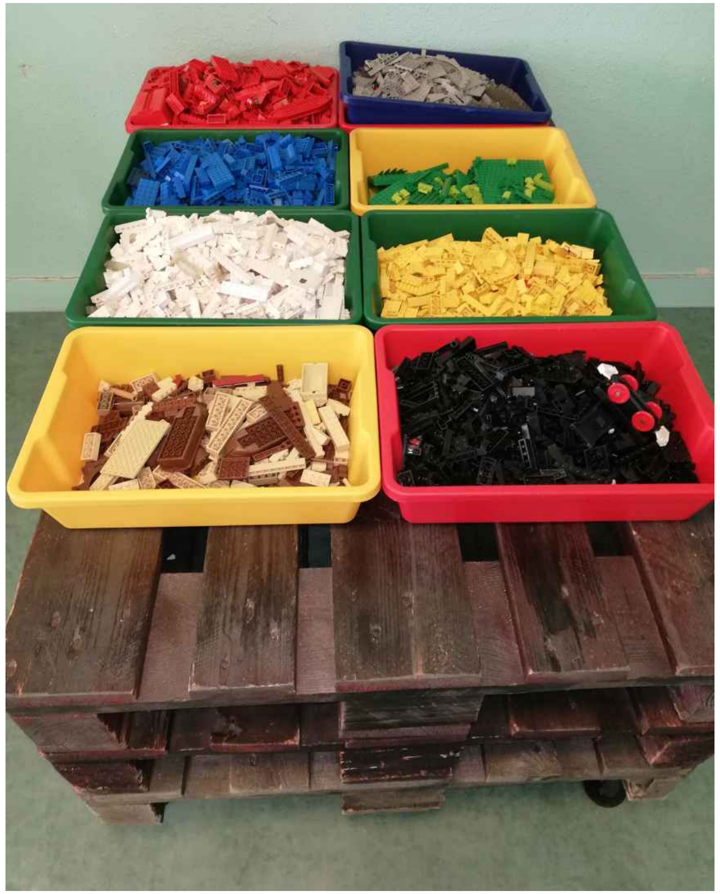


photo by JulienFou on reddit

BucketSort

	A
1	.78
2	.17
3	.39
4	.21
5	.72
6	.94
7	.26
8	.12
9	.23
10	.68

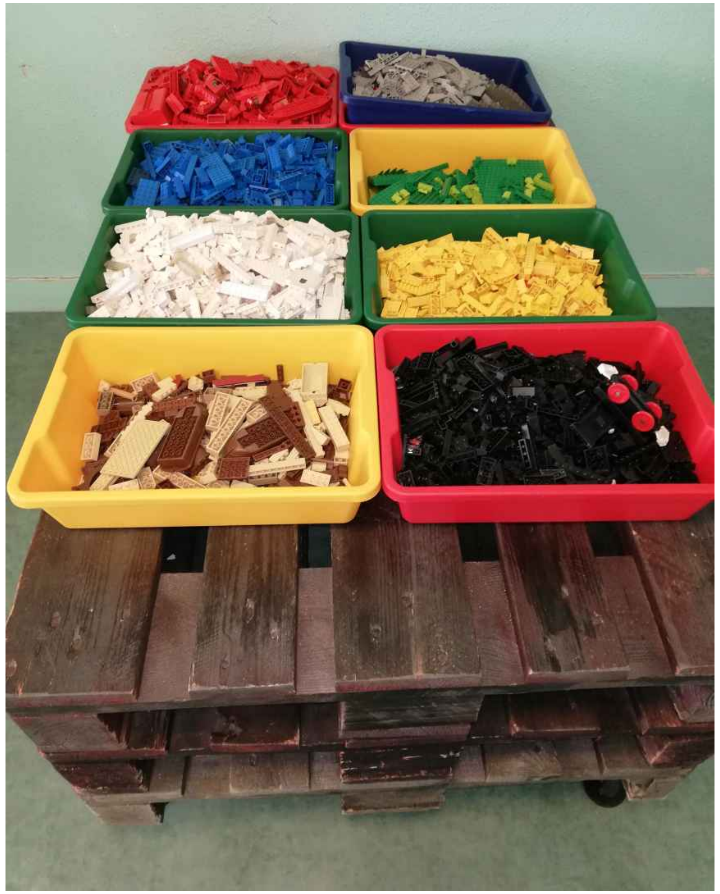


photo by JulienFou on reddit

BucketSort

A

1	.78
2	.17
3	.39
4	.21
5	.72
6	.94
7	.26
8	.12
9	.23
10	.68

Eingabefeld $A[1 \ldots n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

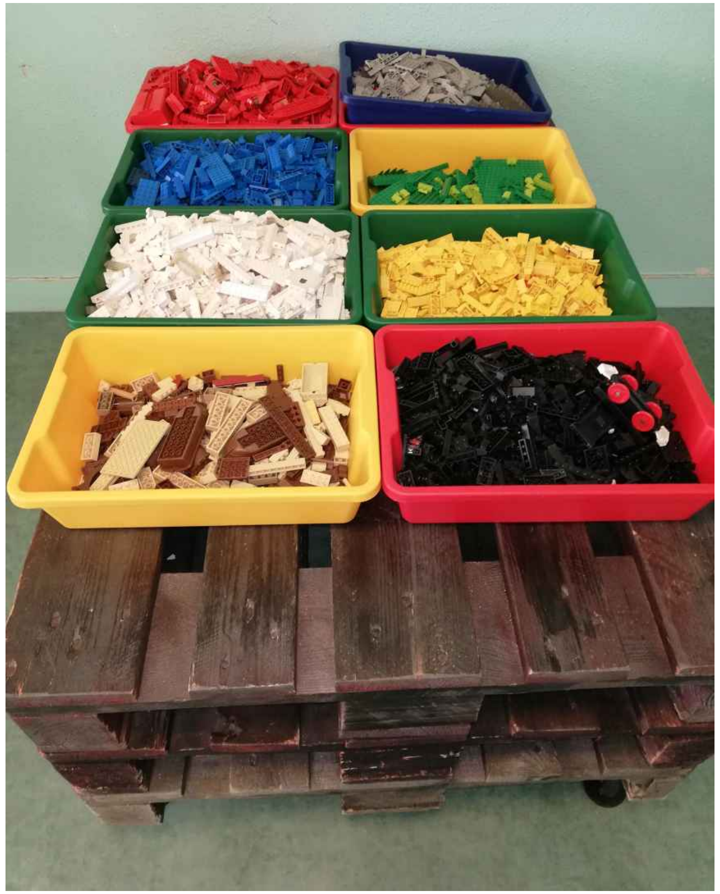


photo by JulienFou on reddit

BucketSort

A

1	.78
2	.17
3	.39
4	.21
5	.72
6	.94
7	.26
8	.12
9	.23
10	.68

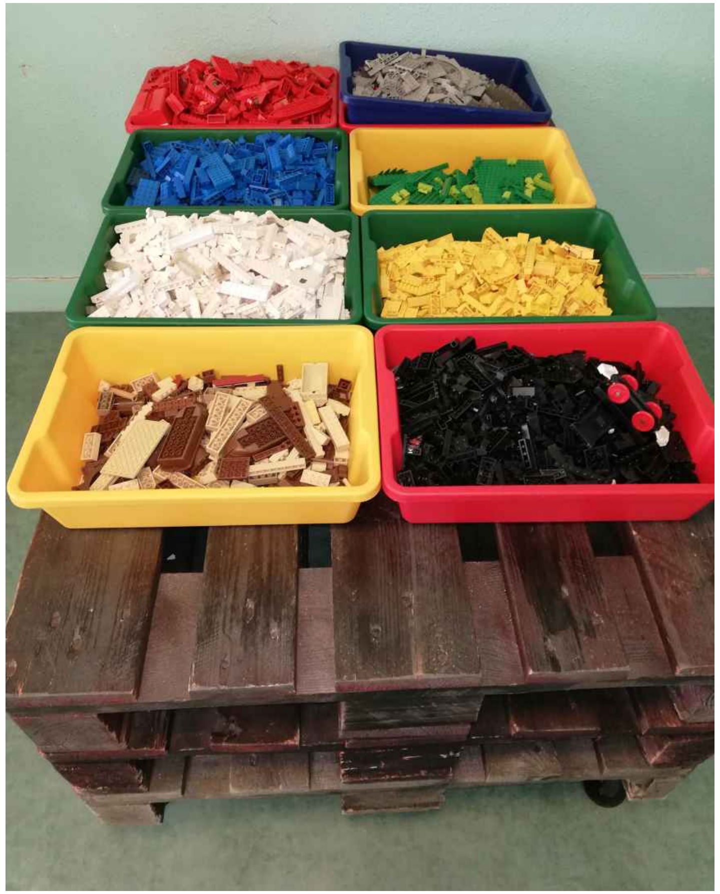


photo by JulienFou on reddit

Eingabefeld $A[1 \dots n]$ enthält Zahlen, zufällig und gleichverteilt aus $[0, 1)$ gezogen

Im Beispielt auf 2 Nachkommastellen gerundet

BucketSort

	A		B
1	.78	0	
2	.17	1	
3	.39	2	
4	.21	3	
5	.72	4	
6	.94	5	
7	.26	6	
8	.12	7	
9	.23	8	
10	.68	9	

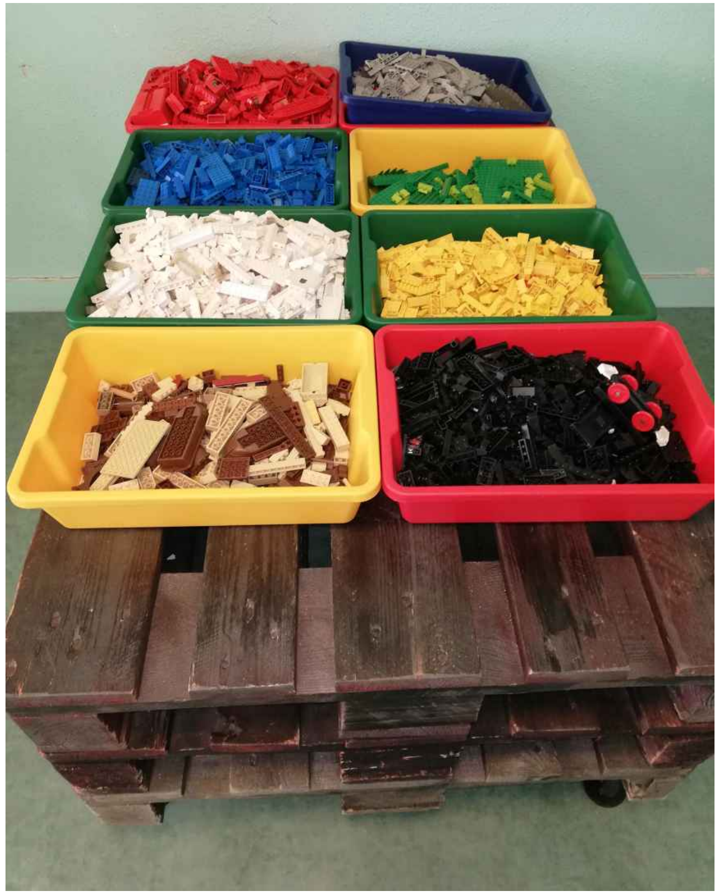


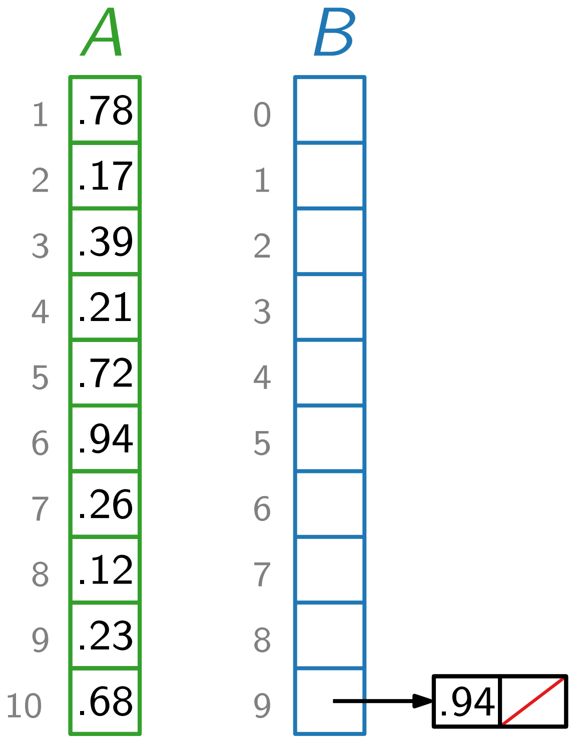
photo by JulienFou on reddit

Hilfsfeld $B[0 \ldots n-1]$;
jeder Eintrag entspricht einem „Eimer“ der Weite $1/n$

Eingabefeld $A[1 \ldots n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

Im Beispielt auf 2 Nach-
kommastellen gerundet

BucketSort



„Eimerinhalt“: **Verkettete Liste** von Elementen aus A.

Hilfsfeld $B[0 \ldots n-1]$;
jeder Eintrag entspricht einem „Eimer“ der Weite $1/n$

Eingabefeld $A[1 \ldots n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

Im Beispielt auf 2 Nachkommastellen gerundet

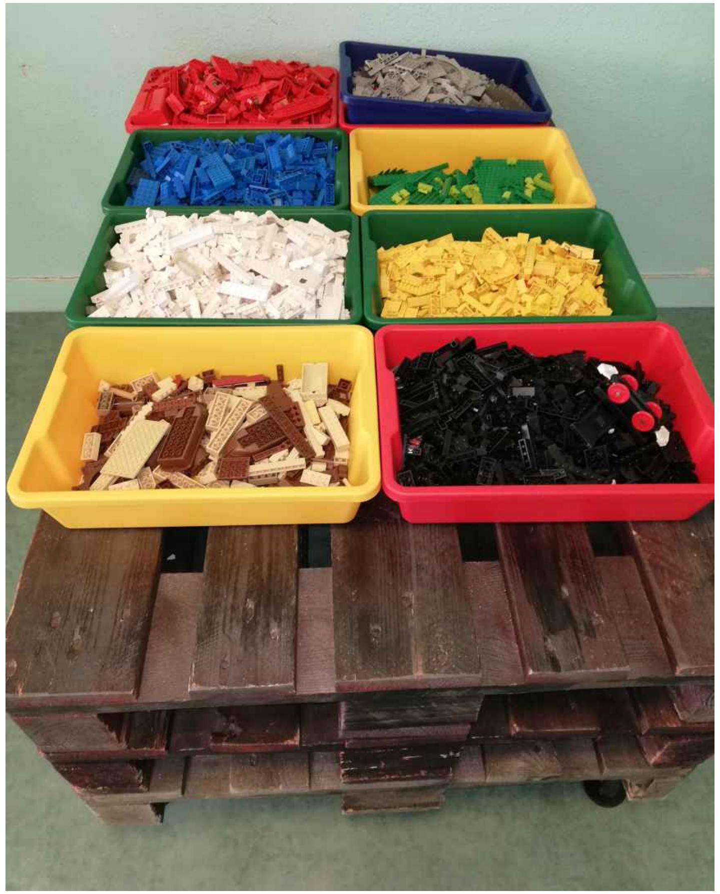
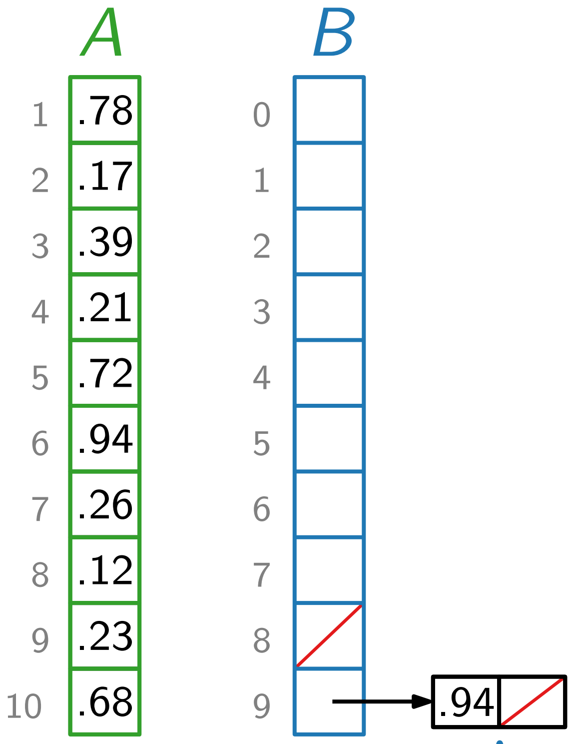


photo by JulienFou on reddit

BucketSort



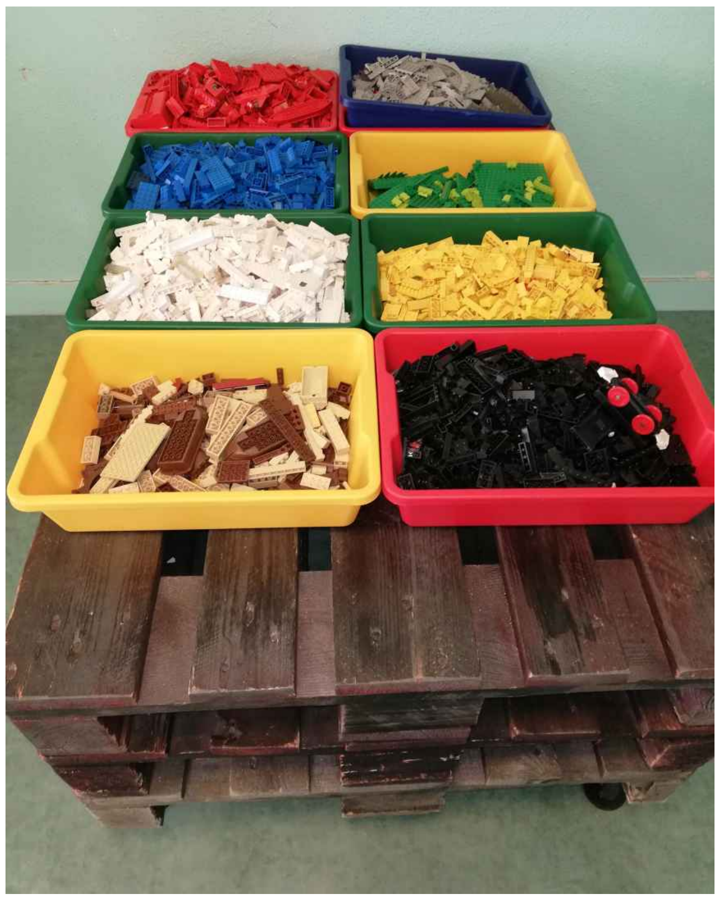


photo by JulienFou on reddit

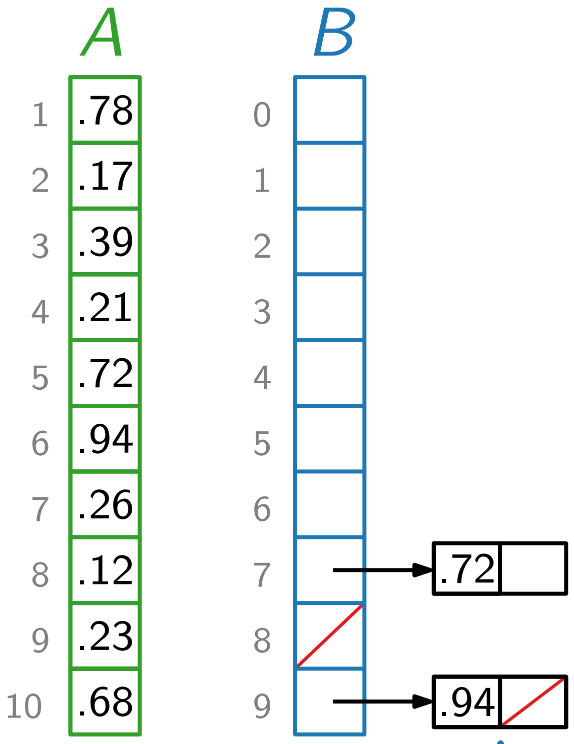
„Eimerinhalt“: **Verkettete Liste** von Elementen aus A.

Hilfsfeld $B[0 \dots n-1]$;
jeder Eintrag entspricht einem „Eimer“ der Weite $1/n$

Eingabefeld $A[1 \dots n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

Im Beispielt auf 2 Nachkommastellen gerundet

BucketSort



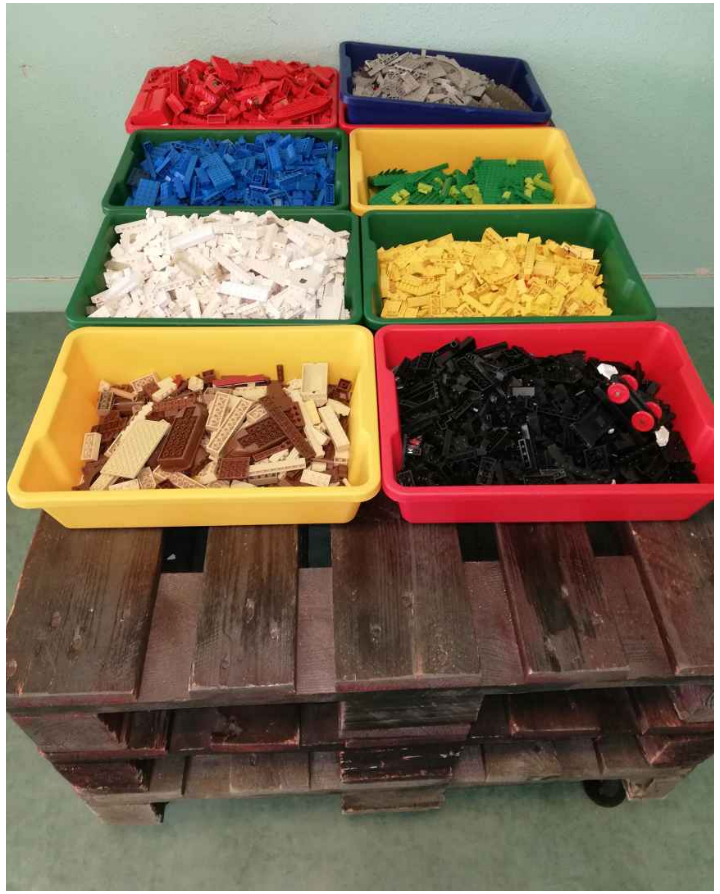


photo by JulienFou on reddit

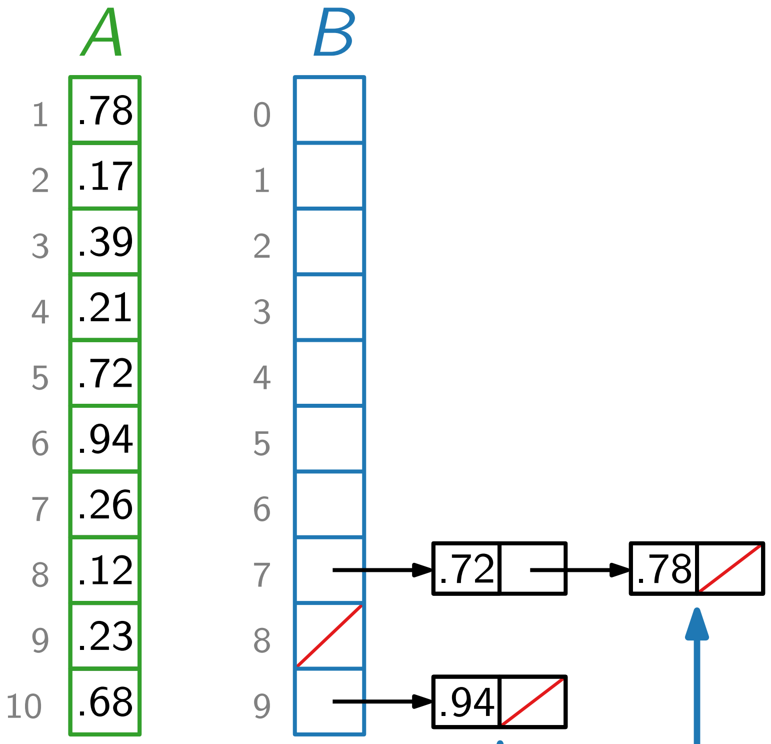
„Eimerinhalt“: **Verkettete Liste** von Elementen aus A.

Hilfsfeld $B[0 \ldots n-1]$;
jeder Eintrag entspricht einem „Eimer“ der Weite $1/n$

Eingabefeld $A[1 \ldots n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

Im Beispielt auf 2 Nachkommastellen gerundet

BucketSort



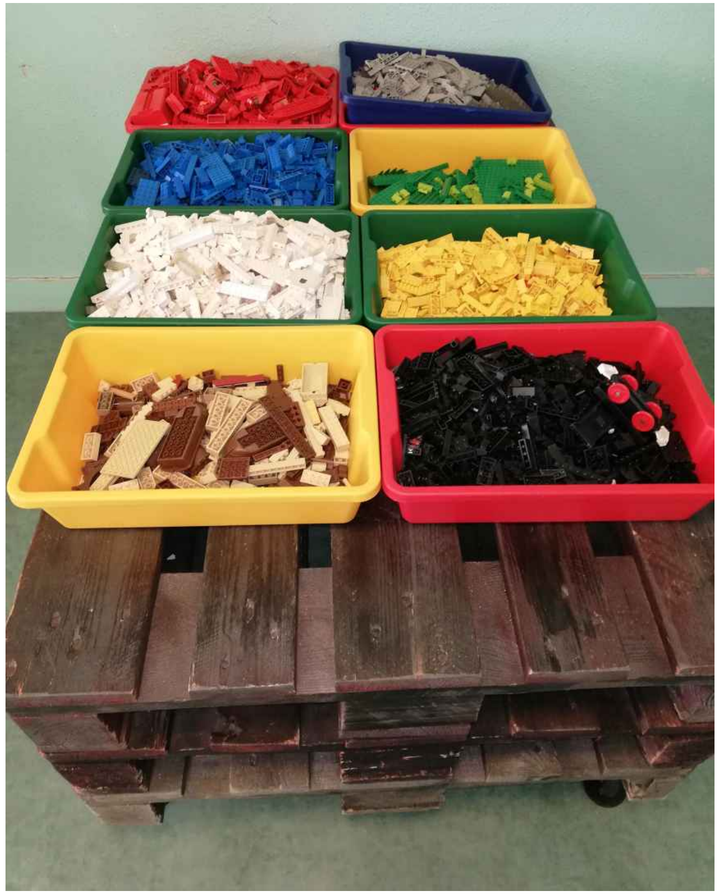


photo by JulienFou on reddit

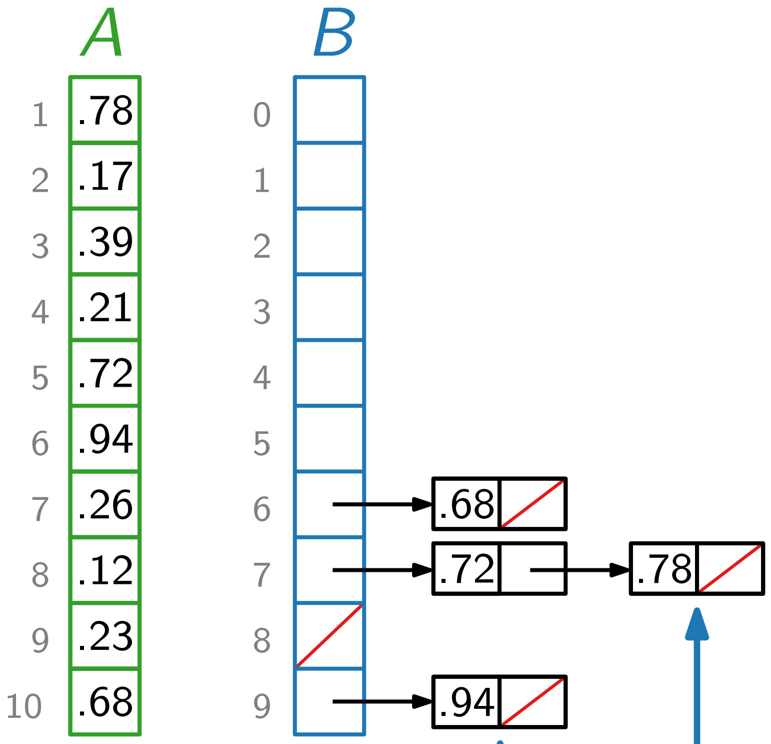
„Eimerinhalt“: **Verkettete Liste** von Elementen aus A.

Hilfsfeld $B[0 \ldots n-1]$;
jeder Eintrag entspricht einem „Eimer“ der Weite $1/n$

Eingabefeld $A[1 \ldots n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

Im Beispielt auf 2 Nachkommastellen gerundet

BUCKETSORT



„Eimerinhalt“: **Verkettete Liste** von Elementen aus A.

Hilfsfeld $B[0 \ldots n-1]$;
jeder Eintrag entspricht einem „Eimer“ der Weite $1/n$

Eingabefeld $A[1 \ldots n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0,1)$ gezogen

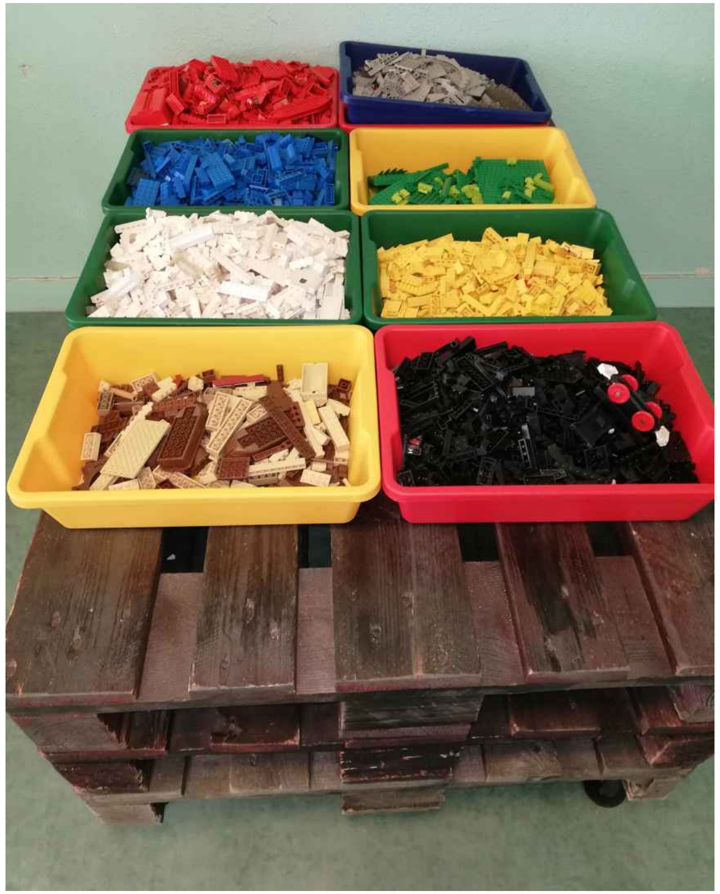
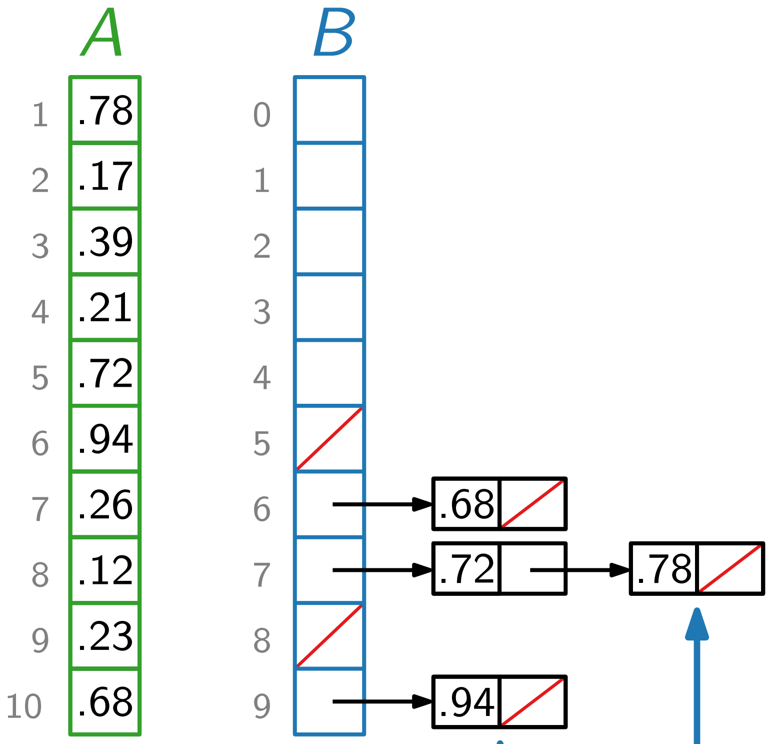


photo by JulienFou on reddit

Im Beispielt auf 2 Nachkommastellen gerundet

BucketSort



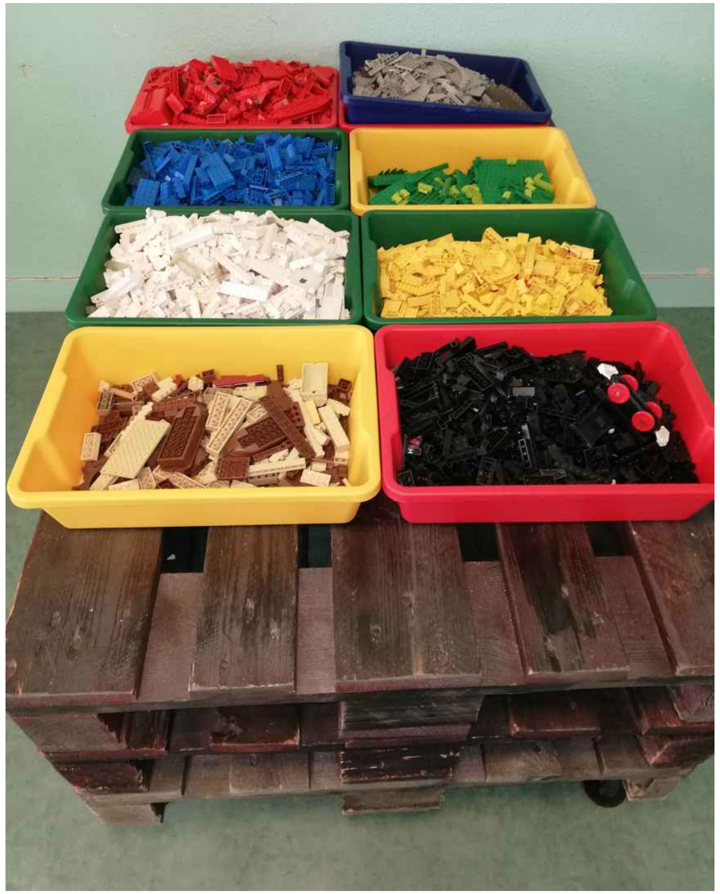


photo by JulienFou on reddit

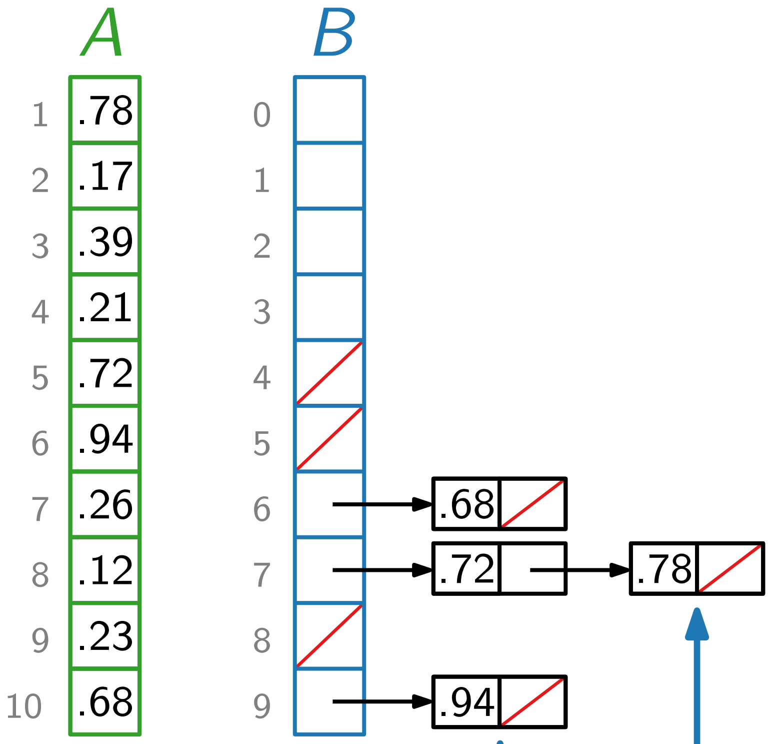
„Eimerinhalt“: **Verkettete Liste** von Elementen aus A.

Hilfsfeld $B[0 \ldots n-1]$;
jeder Eintrag entspricht einem „Eimer“ der Weite $1/n$

Eingabefeld $A[1 \ldots n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

Im Beispielt auf 2 Nachkommastellen gerundet

BucketSort



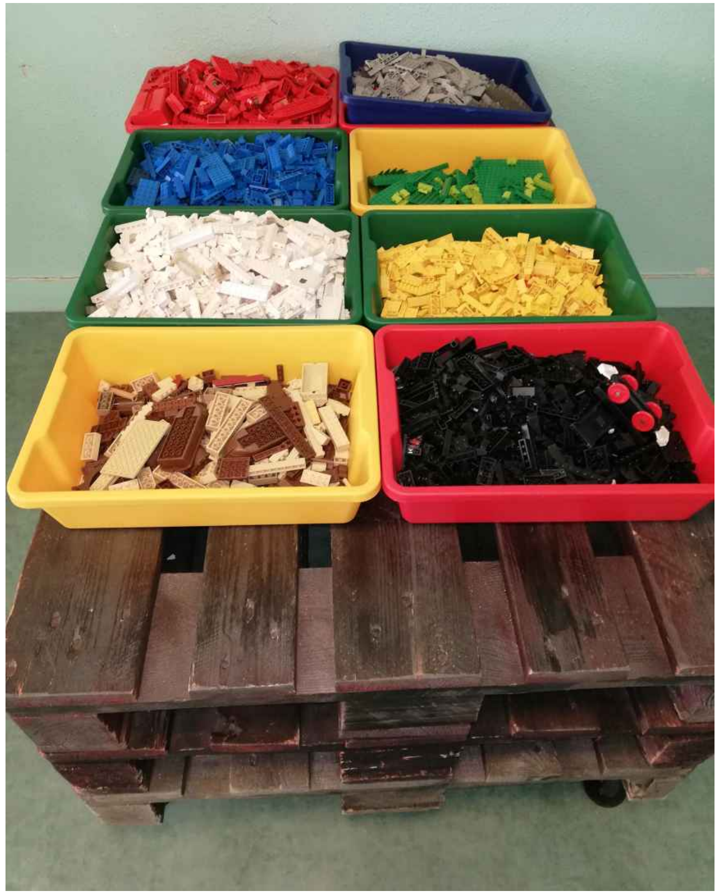


photo by JulienFou on reddit

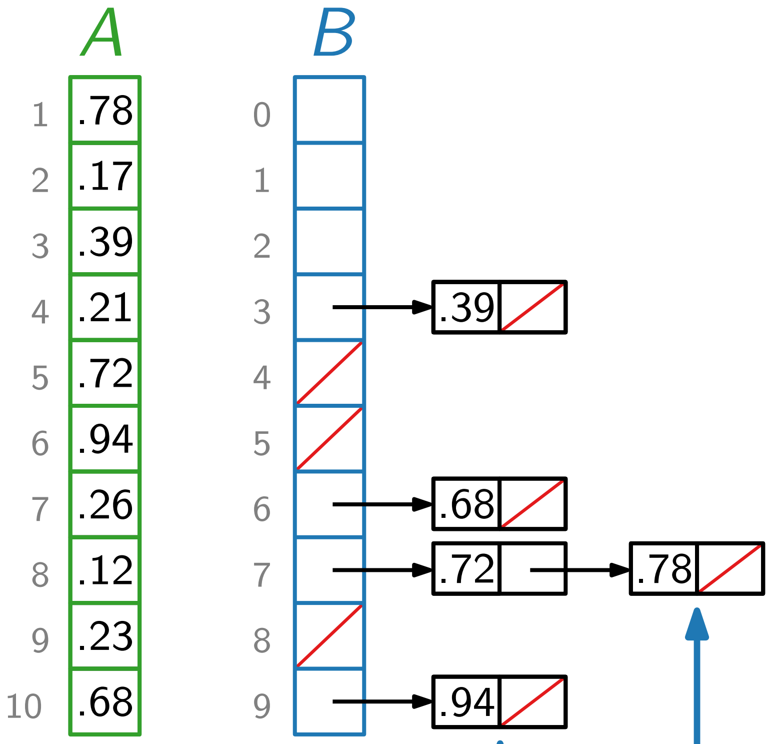
„Eimerinhalt“: **Verkettete Liste** von Elementen aus A.

Hilfsfeld $B[0 \ldots n-1]$;
jeder Eintrag entspricht einem „Eimer“ der Weite $1/n$

Eingabefeld $A[1 \ldots n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

Im Beispielt auf 2 Nachkommastellen gerundet

BucketSort



„Eimerinhalt“: **Verkettete Liste** von Elementen aus A.

Hilfsfeld $B[0 \ldots n-1]$;
jeder Eintrag entspricht einem „Eimer“ der Weite $1/n$

Eingabefeld $A[1 \ldots n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

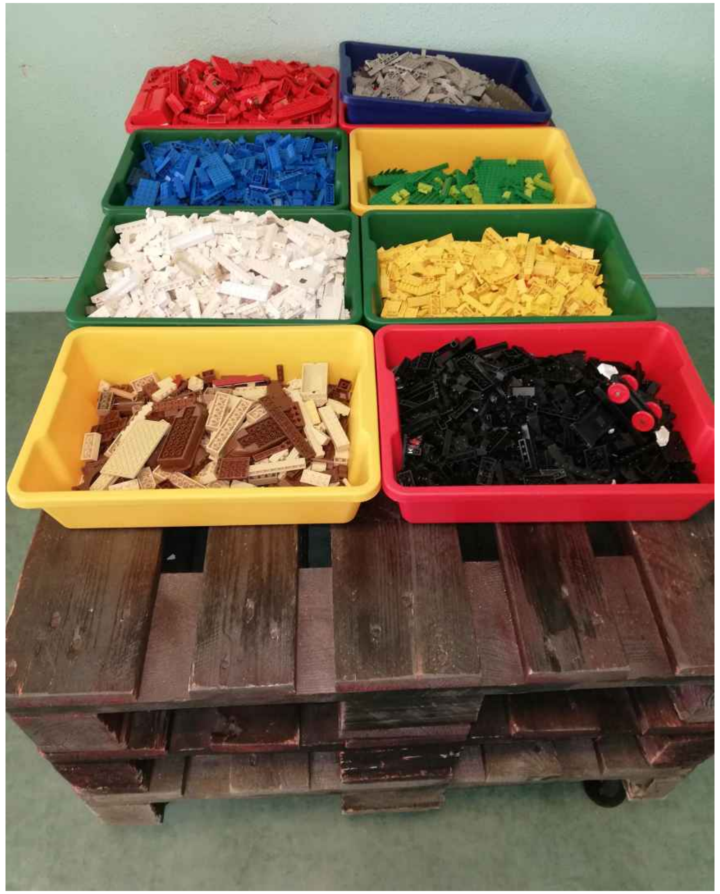
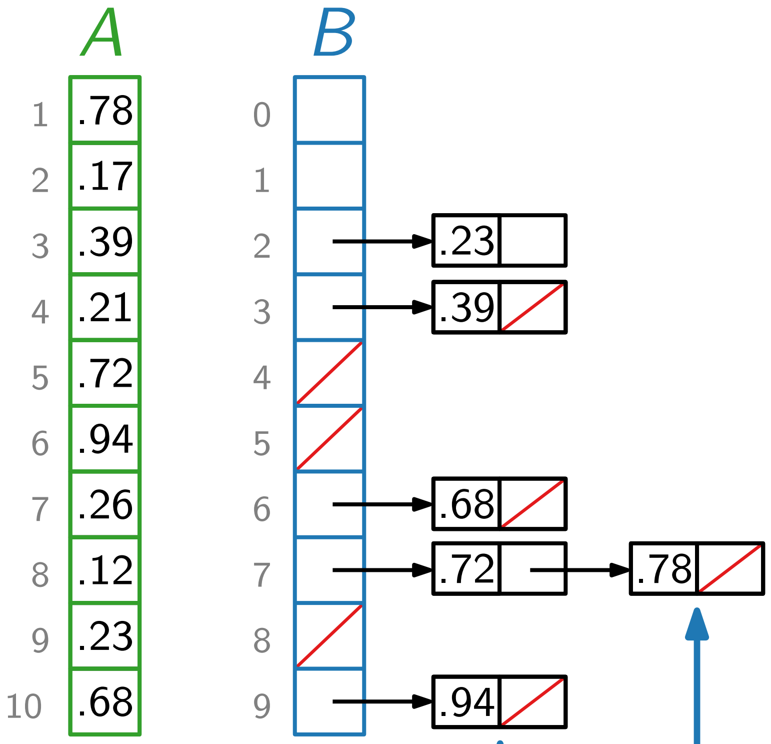


photo by JulienFou on reddit

Im Beispielt auf 2 Nachkommastellen gerundet

BucketSort



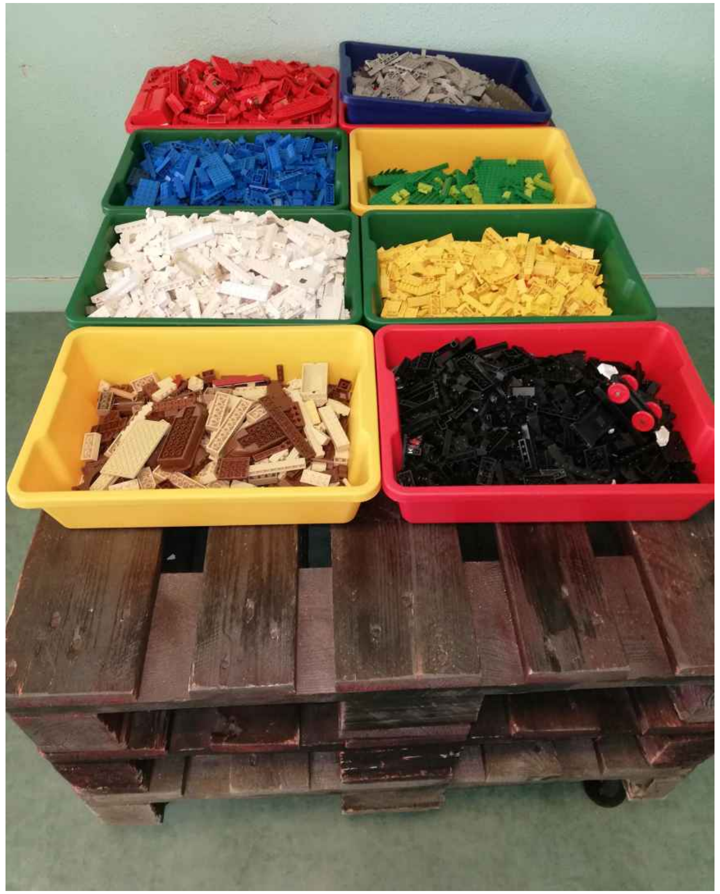


photo by JulienFou on reddit

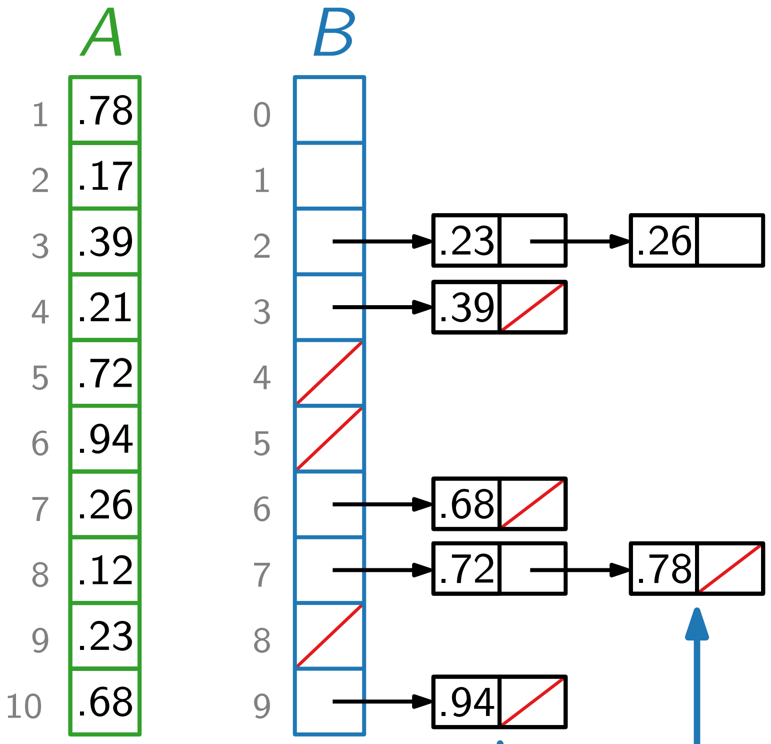
„Eimerinhalt“: **Verkettete Liste** von Elementen aus A.

Hilfsfeld $B[0 \dots n-1]$;
jeder Eintrag entspricht einem „Eimer“ der Weite $1/n$

Eingabefeld $A[1 \dots n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

Im Beispielt auf 2 Nachkommastellen gerundet

BucketSort



„Eimerinhalt“: **Verkettete Liste** von Elementen aus A.

Hilfsfeld $B[0 \dots n-1]$;
jeder Eintrag entspricht einem „Eimer“ der Weite $1/n$

Eingabefeld $A[1 \dots n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0,1)$ gezogen

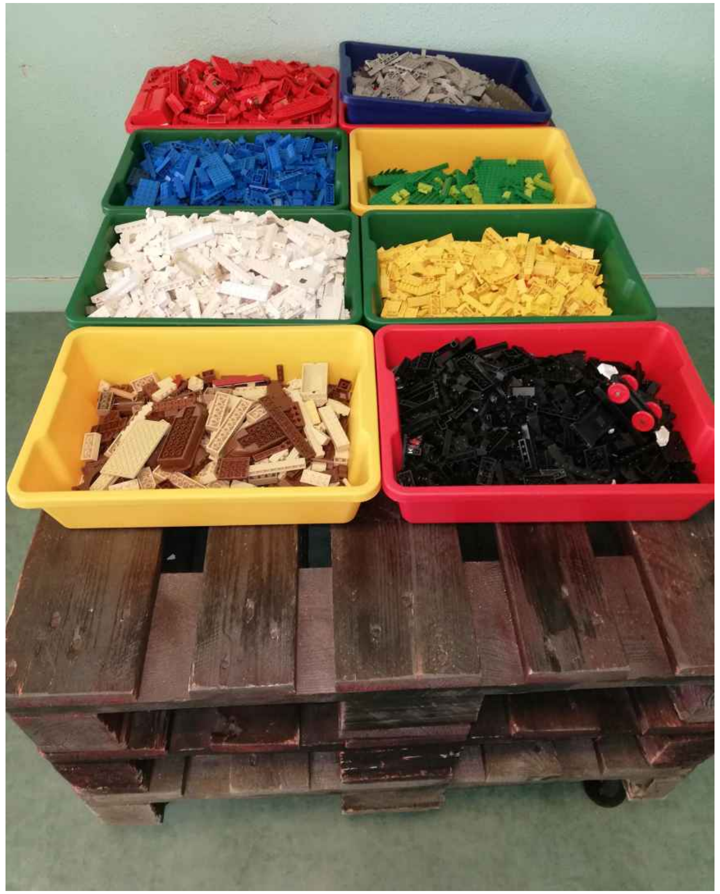
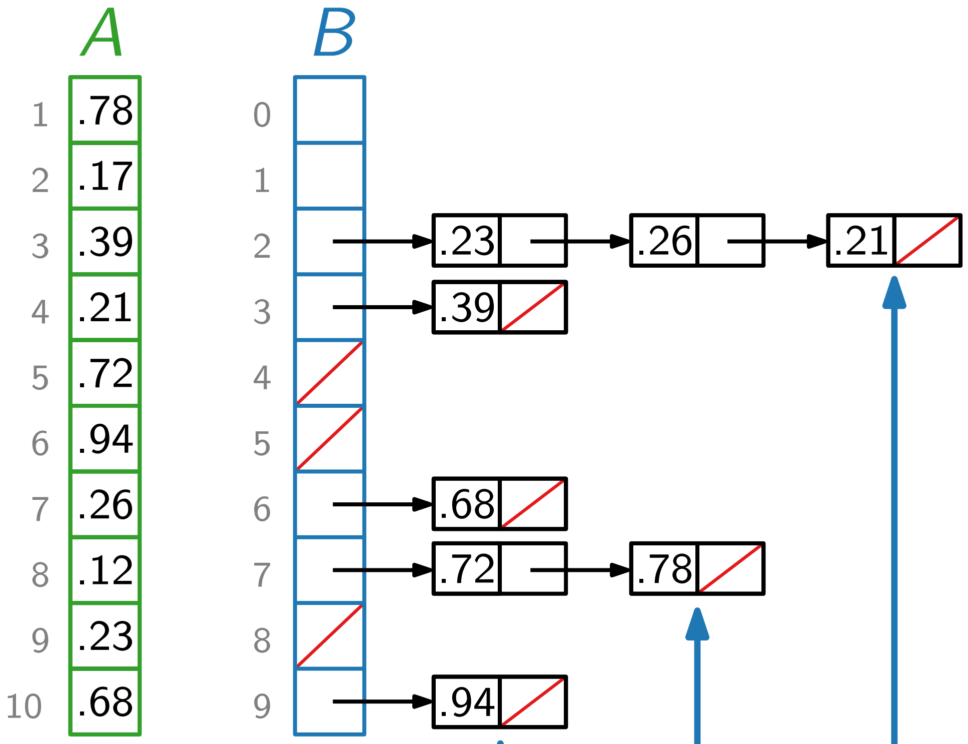


photo by JulienFou on reddit

Im Beispielt auf 2 Nachkommastellen gerundet

BucketSort



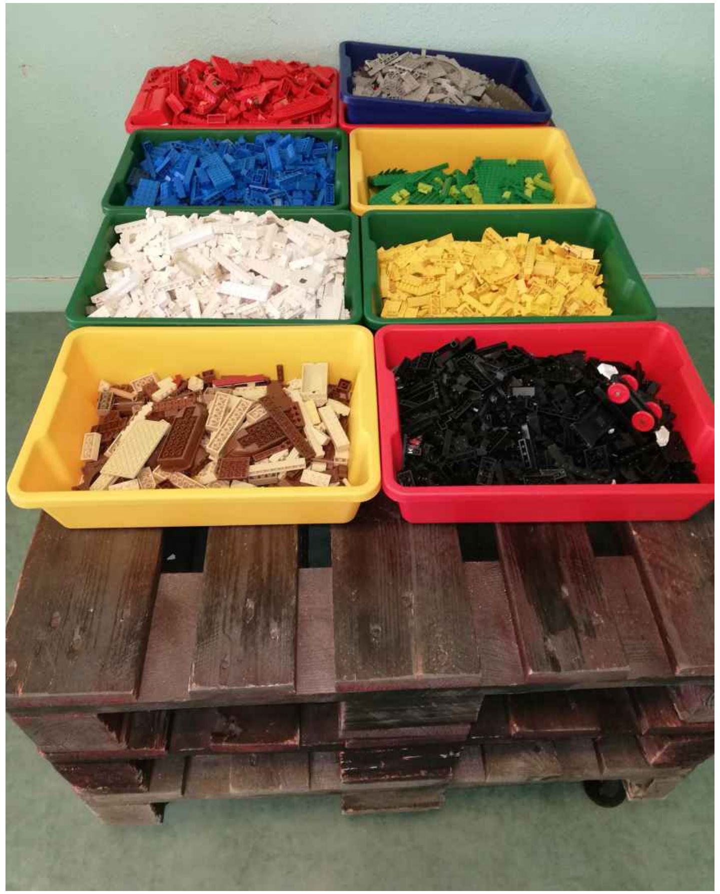


photo by JulienFou on reddit

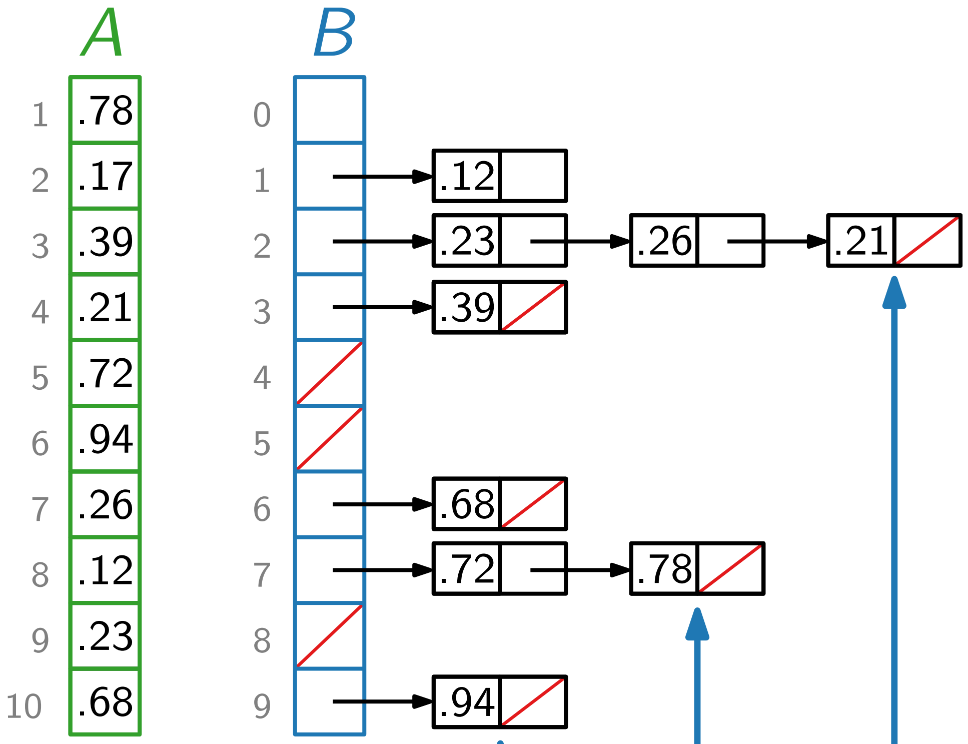
„Eimerinhalt“: **Verkettete Liste** von Elementen aus A.

Hilfsfeld $B[0 \ldots n-1]$;
jeder Eintrag entspricht einem „Eimer“ der Weite $1/n$

Eingabefeld $A[1 \ldots n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

Im Beispielt auf 2 Nachkommastellen gerundet

BucketSort



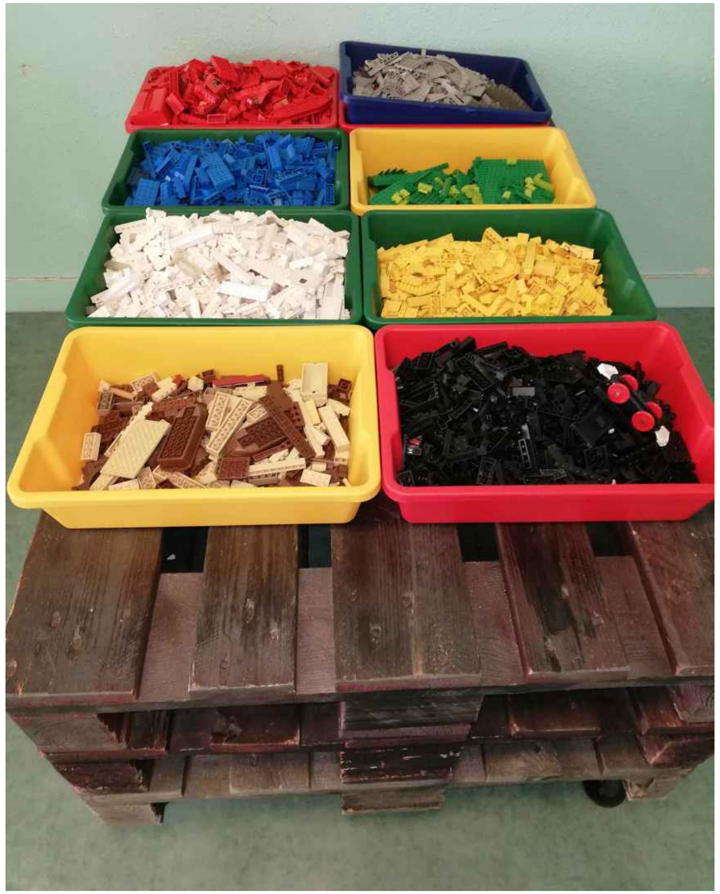


photo by JulienFou on reddit

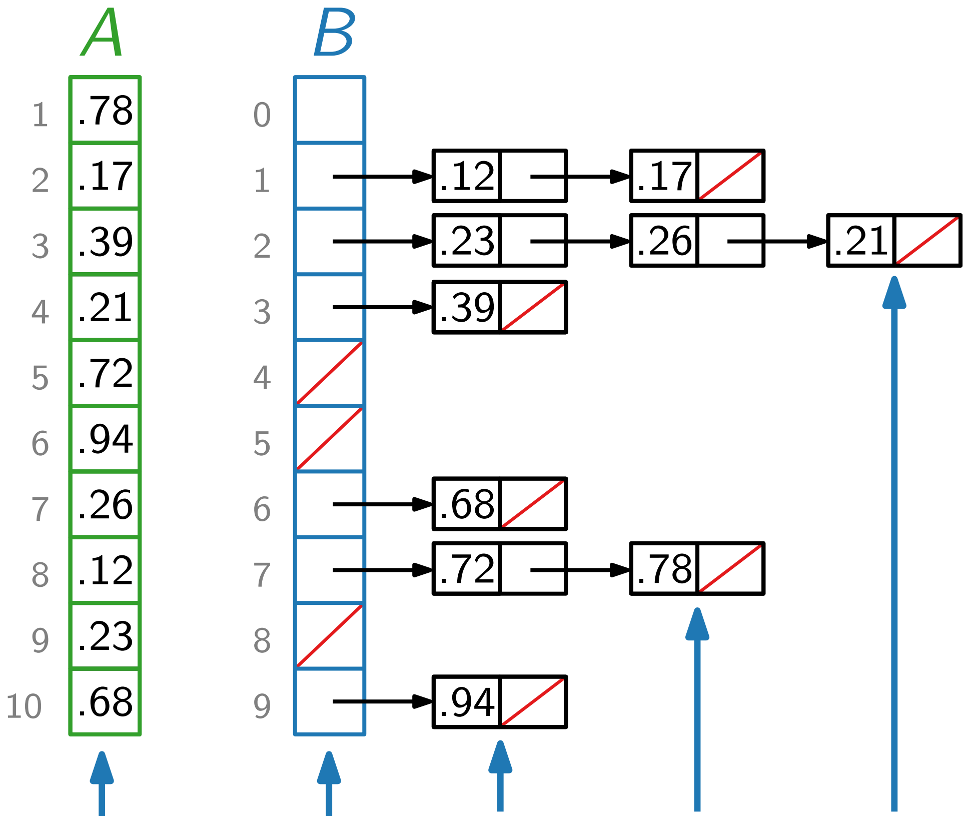
„Eimerinhalt“: **Verkettete Liste** von Elementen aus A.

Hilfsfeld $B[0 \dots n-1]$;
jeder Eintrag entspricht einem „Eimer“ der Weite $1/n$

Eingabefeld $A[1 \dots n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

Im Beispielt auf 2 Nachkommastellen gerundet

BucketSort



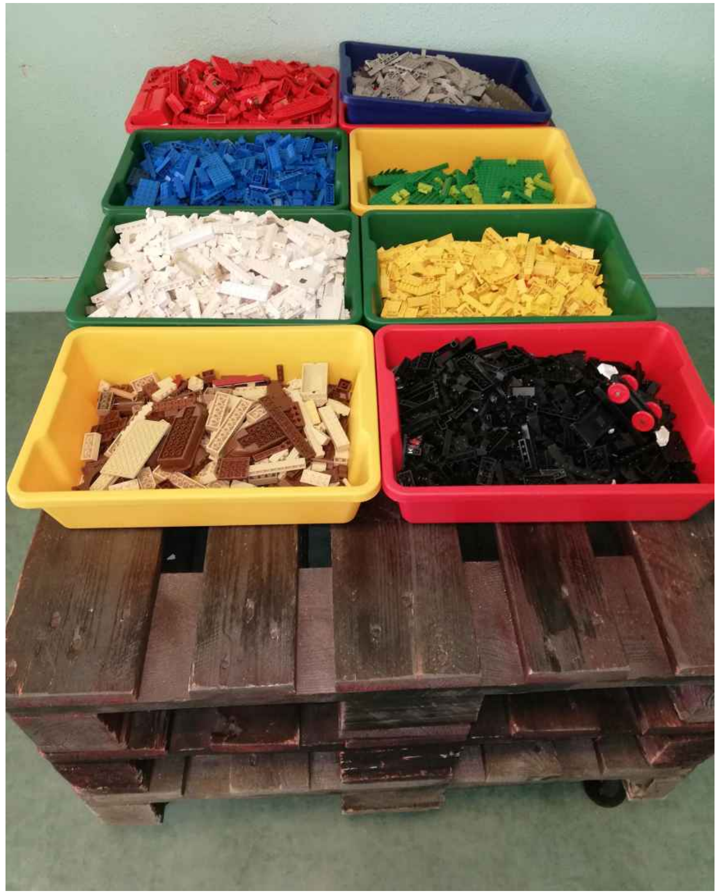


photo by JulienFou on reddit

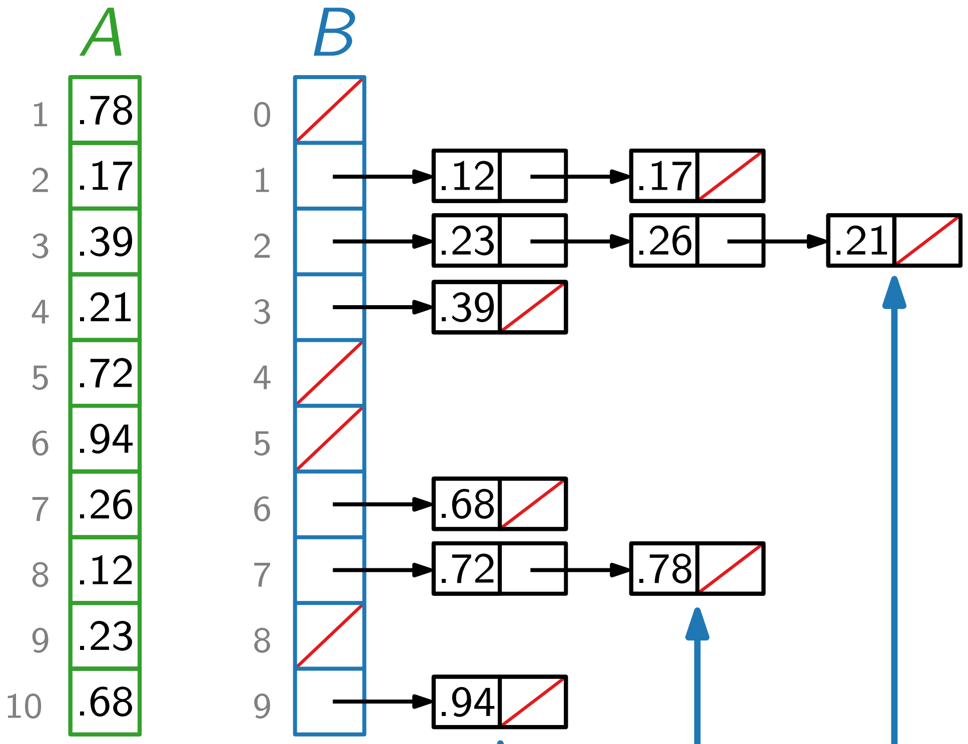
„Eimerinhalt“: **Verkettete Liste** von Elementen aus A.

Hilfsfeld $B[0 \ldots n-1]$;
jeder Eintrag entspricht einem „Eimer“ der Weite $1/n$

Eingabefeld $A[1 \ldots n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

Im Beispielt auf 2 Nachkommastellen gerundet

BucketSort



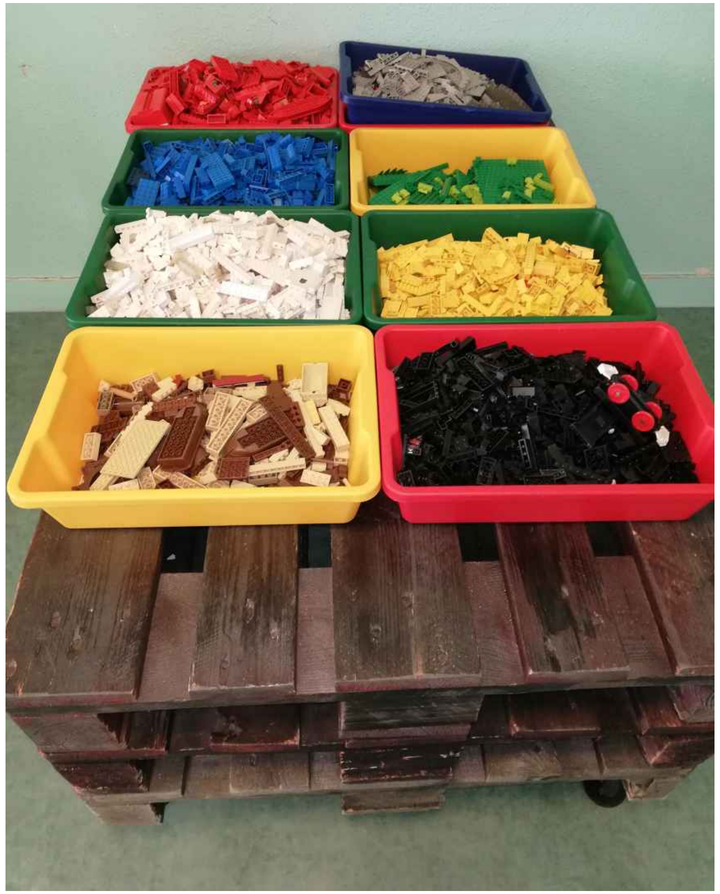


photo by JulienFou on reddit

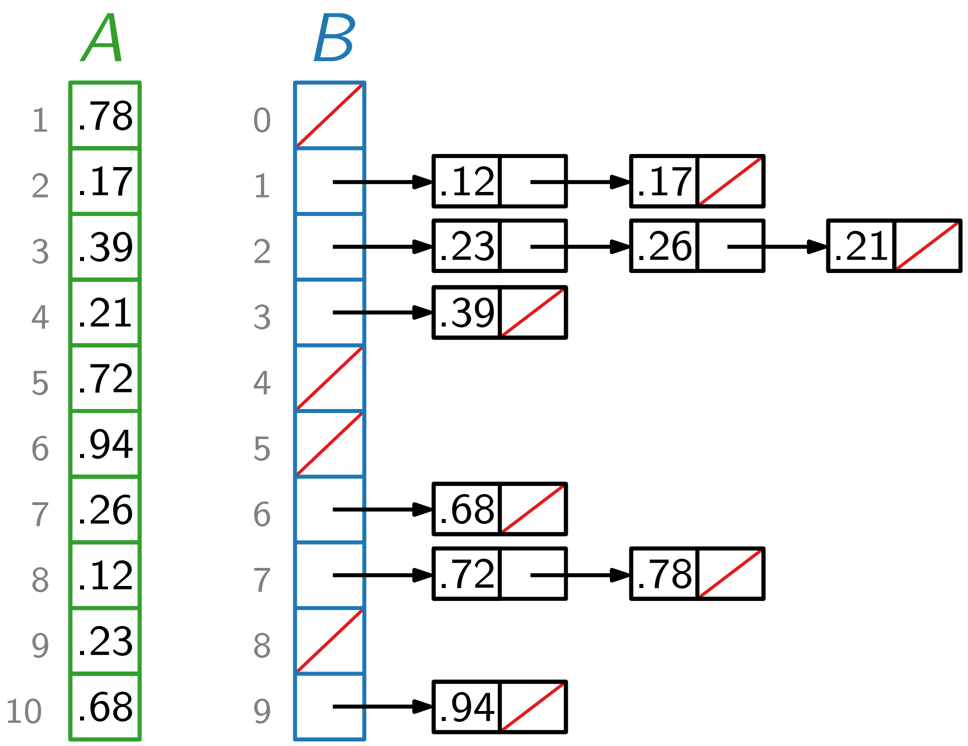
„Eimerinhalt": **Verkettete Liste** von Elementen aus A.

Hilfsfeld $B[0 \ldots n-1]$;
jeder Eintrag entspricht einem „Eimer" der Weite $1/n$

Eingabefeld $A[1 \ldots n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

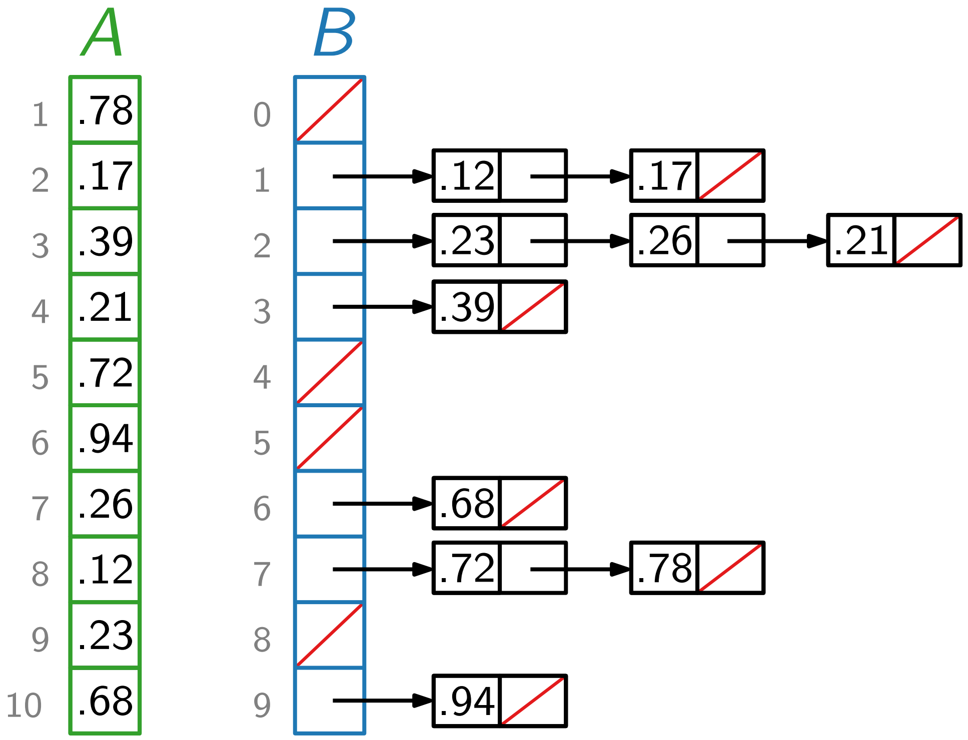
Im Beispielt auf 2 Nachkommastellen gerundet

BucketSort



BucketSort(Feld A von Zahlen in $[0,1)$)

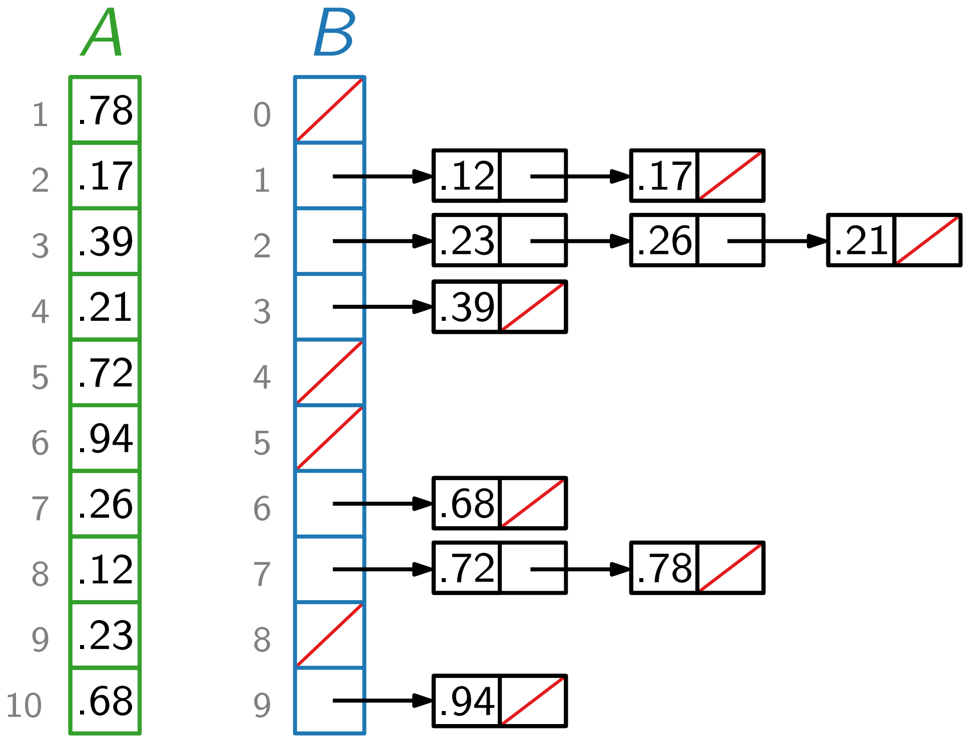
BUCKETSORT



BUCKETSORT(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

BucketSort

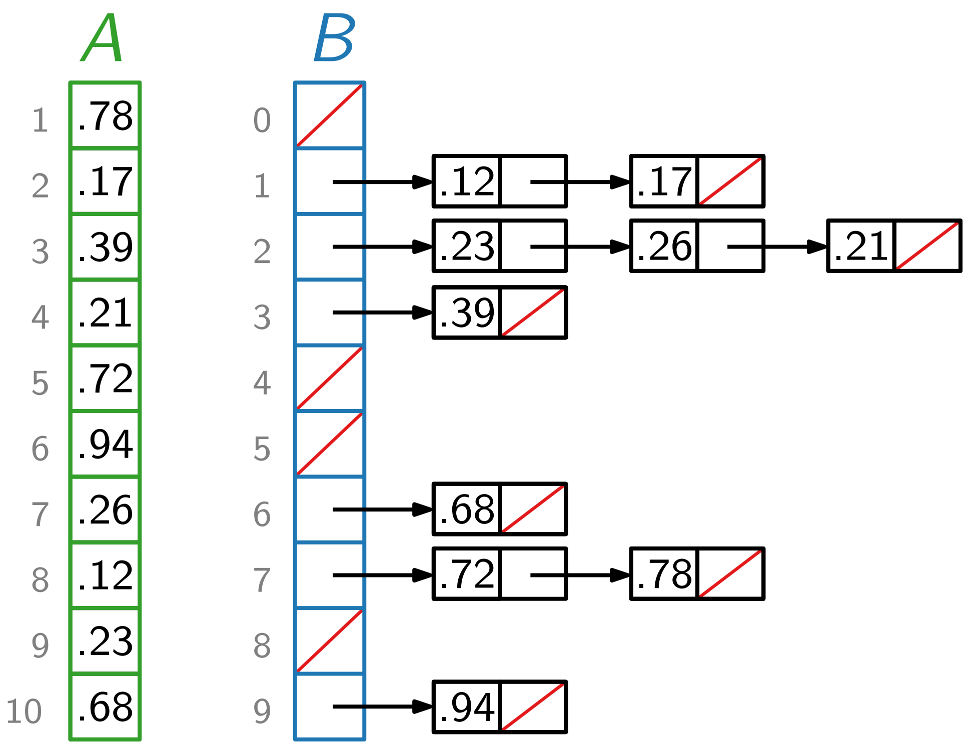


BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.\mathit{length}$

lege Feld $B[0 \ldots n-1]$ von Listen an

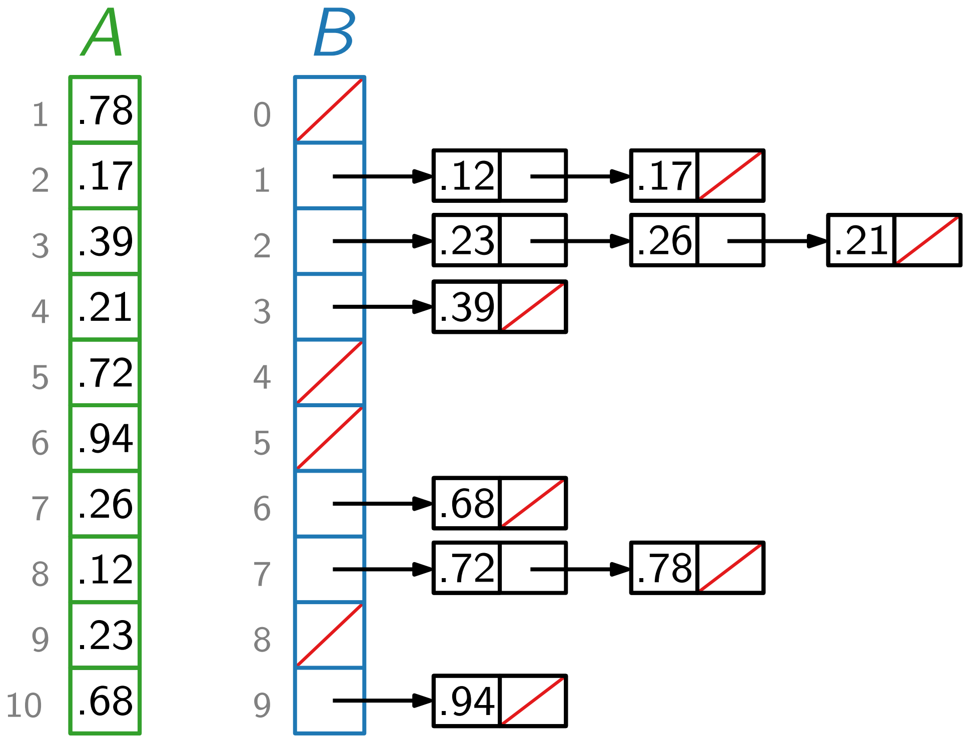
BucketSort



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$
lege Feld $B[0 \ldots n-1]$ von Listen an
for $j = 1$ **to** n **do**

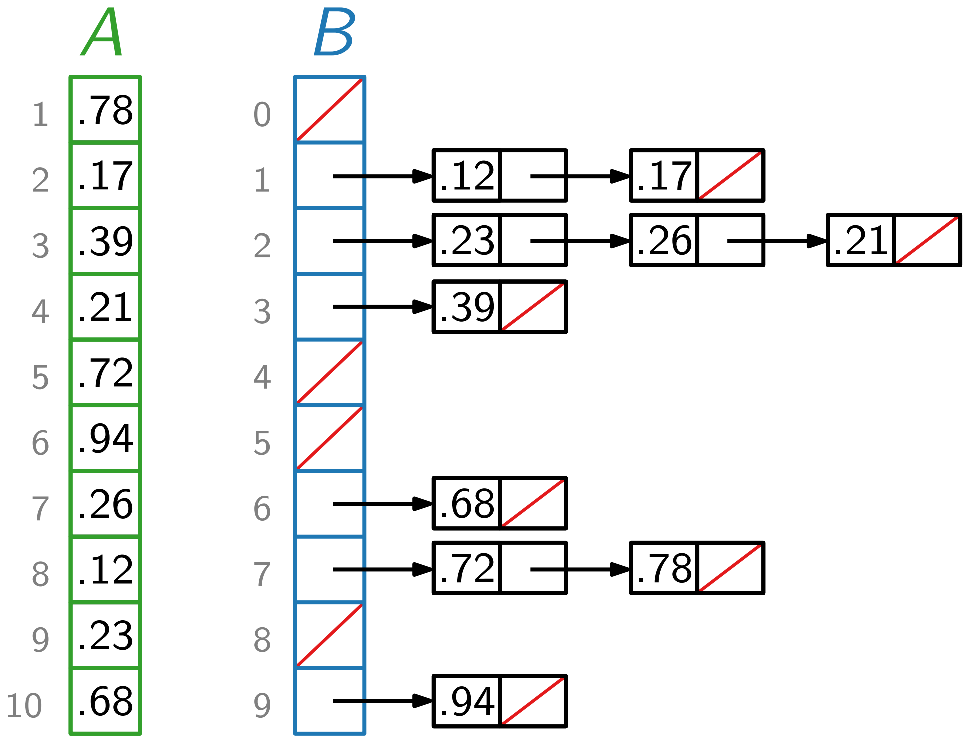
BucketSort



BucketSort(Feld A von Zahlen in $[0,1)$)

$n = A.length$
lege Feld $B[0 \ldots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\quad]$ ein

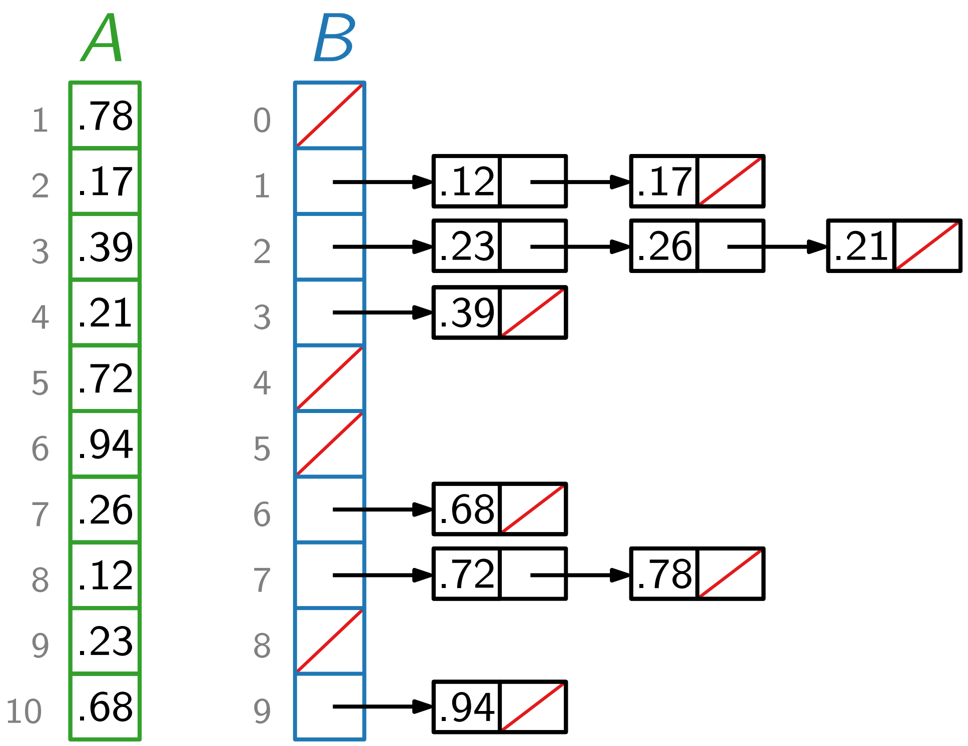
BUCKETSORT



BUCKETSORT(Feld A von Zahlen in $[0,1)$)

$n = A.length$
lege Feld $B[0 \ldots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\quad]$ ein
for $i = 0$ **to** $n-1$ **do**

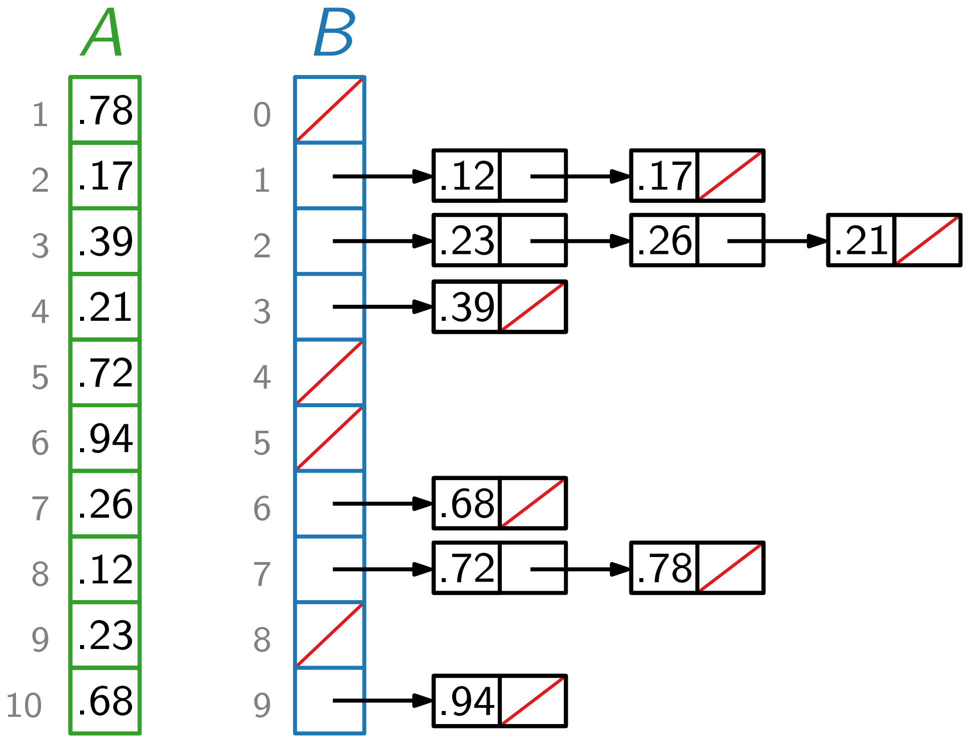
BucketSort



BucketSort(Feld A von Zahlen in $[0,1)$)

$n = A.length$
lege Feld $B[0 \dots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\quad]$ ein
for $i = 0$ **to** $n-1$ **do**
  sortiere Liste $B[i]$

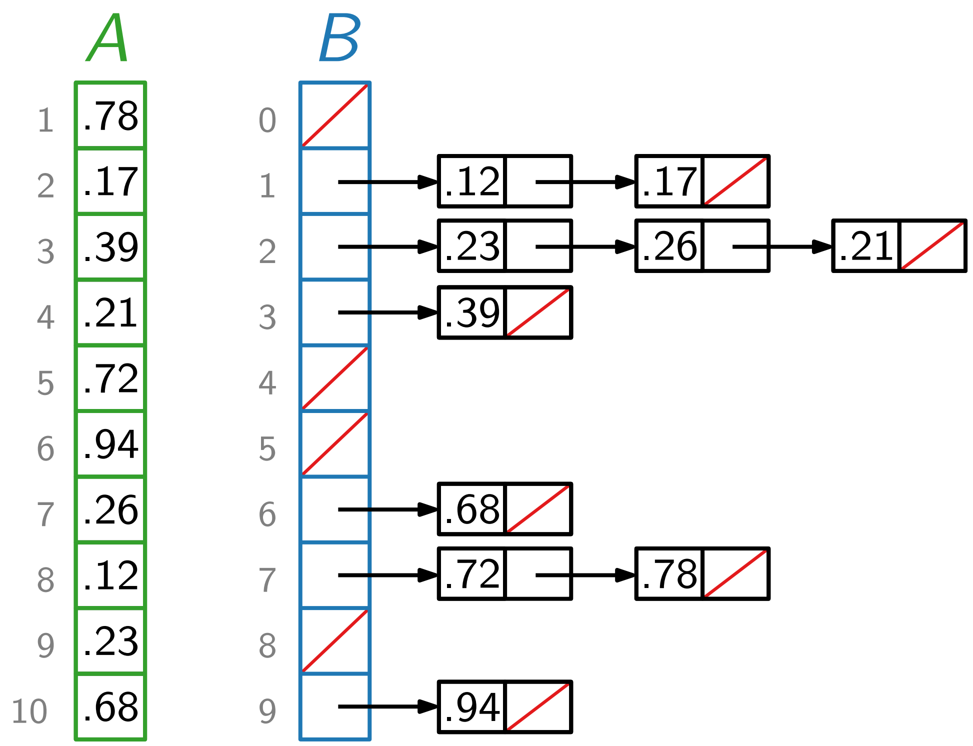
BucketSort



BucketSort(Feld A von Zahlen in $[0,1)$)

$n = A.length$
lege Feld $B[0 \dots n-1]$ von Listen an
for $j = 1$ **to** n **do**
 füge $A[j]$ in Liste $B[\quad]$ ein
for $i = 0$ **to** $n-1$ **do**
 sortiere Liste $B[i]$

BucketSort



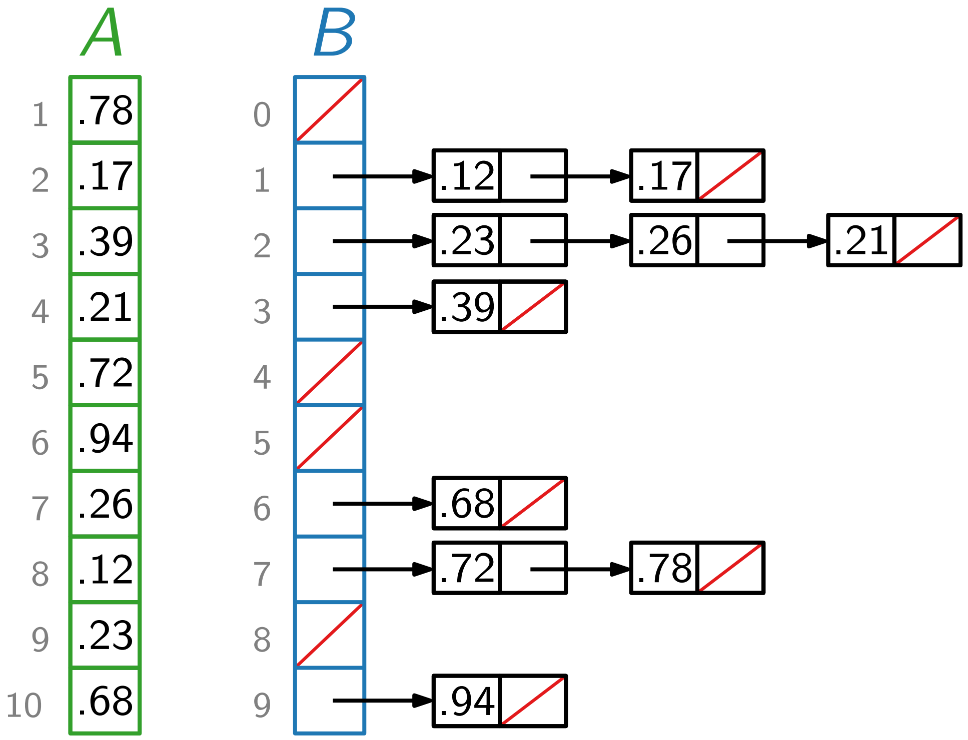
BucketSort(Feld A von Zahlen in $[0,1)$)

$n = A.length$
lege Feld $B[0 \dots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\quad]$ ein
for $i = 0$ **to** $n-1$ **do**
  sortiere Liste $B[i]$

Aufgabe:

Füllen Sie die Felder mit Code, der BucketSort umsetzt!

BucketSort

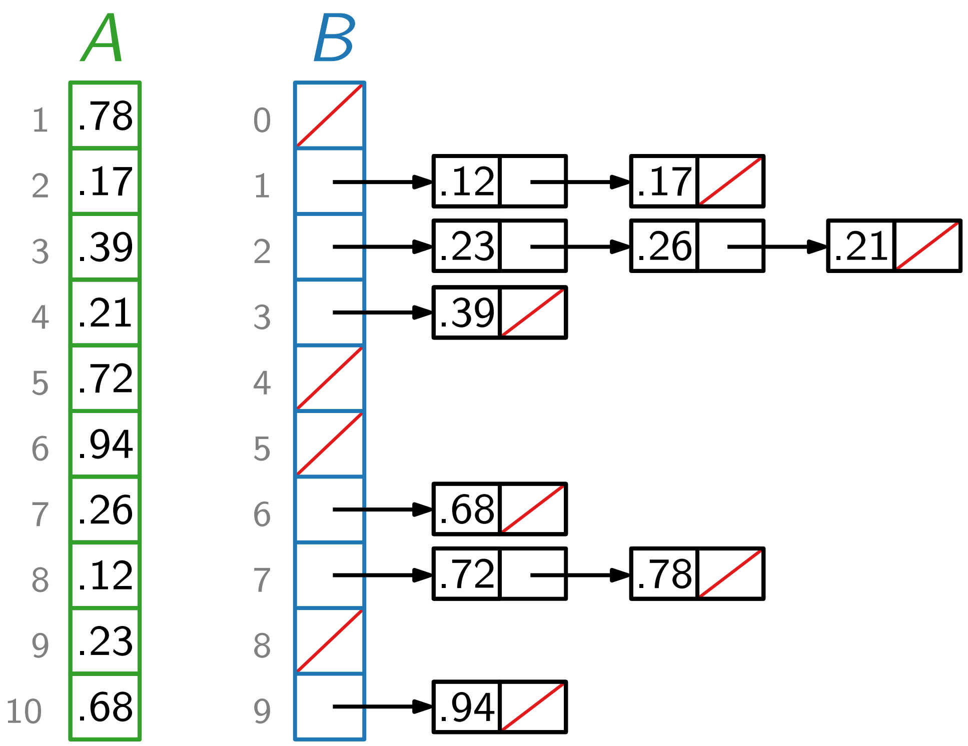


BucketSort(Feld A von Zahlen in $[0,1)$)

$n = A.length$
lege Feld $B[0 \dots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein
for $i = 0$ **to** $n-1$ **do**
  sortiere Liste $B[i]$

Aufgabe:
Füllen Sie die Felder mit Code, der BucketSort umsetzt!

BucketSort

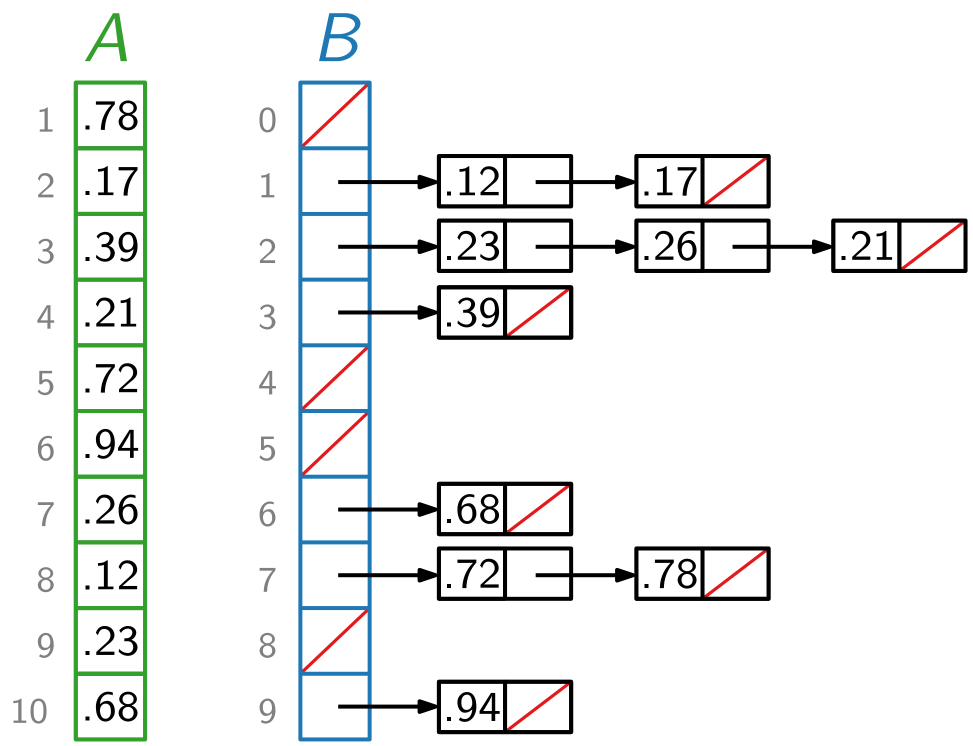


BucketSort(Feld A von Zahlen in $[0,1)$)

$n = A.length$
lege Feld $B[0 \dots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein
for $i = 0$ **to** $n-1$ **do**
  sortiere Liste $B[i]$
hänge $B[0], \dots, B[n-1]$ aneinander

Aufgabe:
Füllen Sie die Felder mit Code, der BucketSort umsetzt!

BucketSort

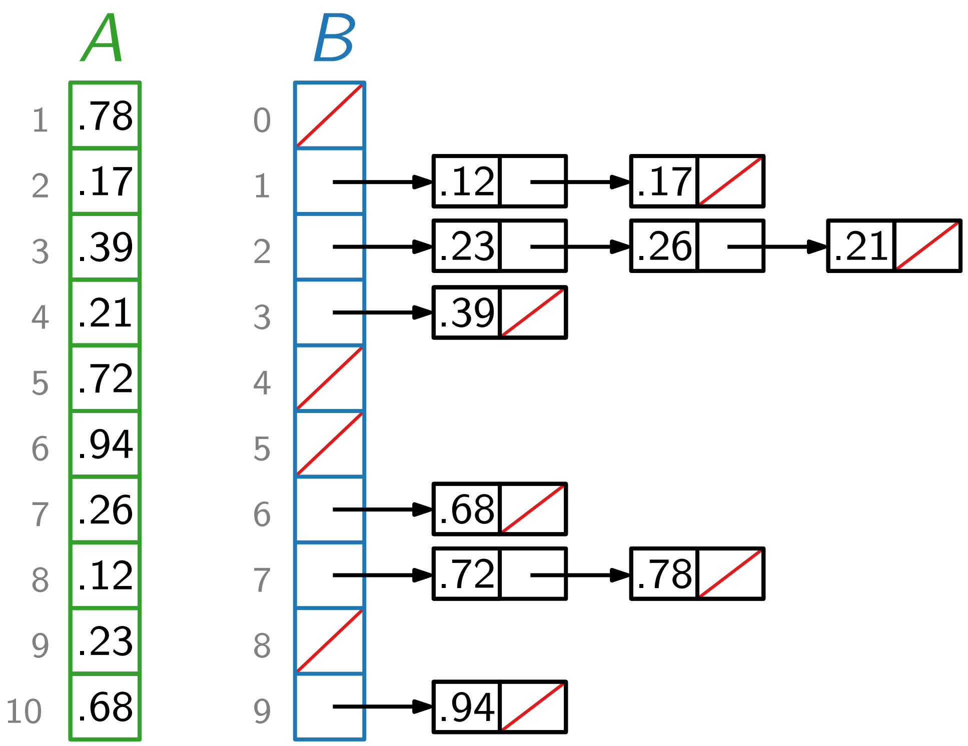


BucketSort(Feld A von Zahlen in $[0,1)$)

$n = A.length$
lege Feld $B[0 \dots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein
for $i = 0$ **to** $n-1$ **do**
  sortiere Liste $B[i]$
hänge $B[0], \dots, B[n-1]$ aneinander
kopiere das Ergebnis nach $A[1 \dots n]$

Aufgabe:
Füllen Sie die Felder mit Code, der BucketSort umsetzt!

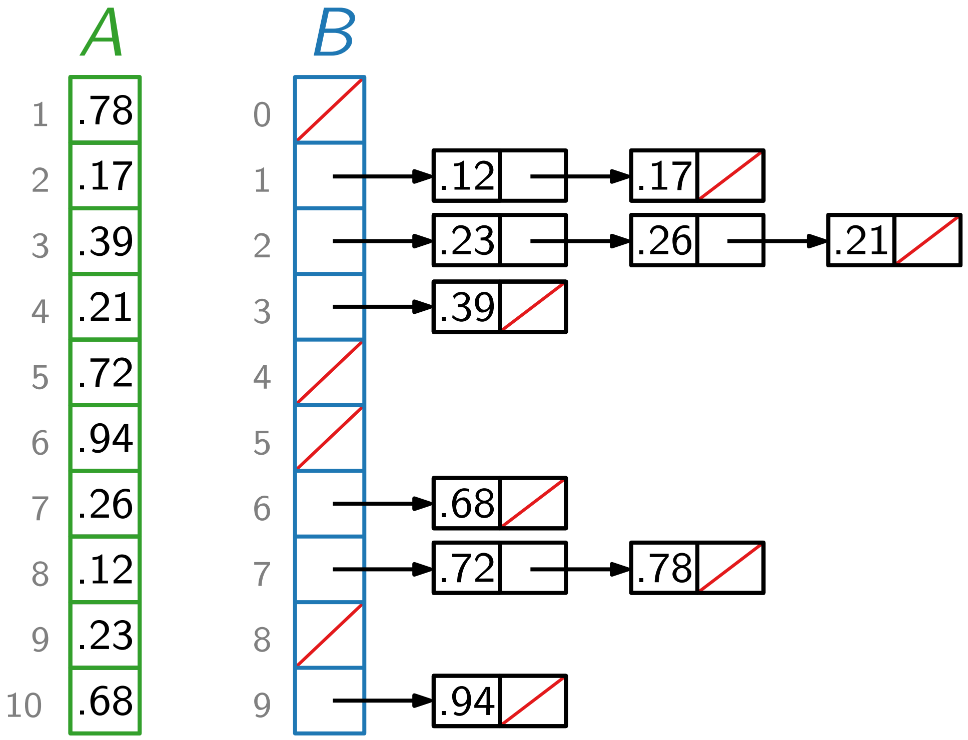
BucketSort



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$
lege Feld $B[0 \dots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein
for $i = 0$ **to** $n - 1$ **do**
  sortiere Liste $B[i]$
hänge $B[0], \dots, B[n-1]$ aneinander
kopiere das Ergebnis nach $A[1 \dots n]$

BucketSort

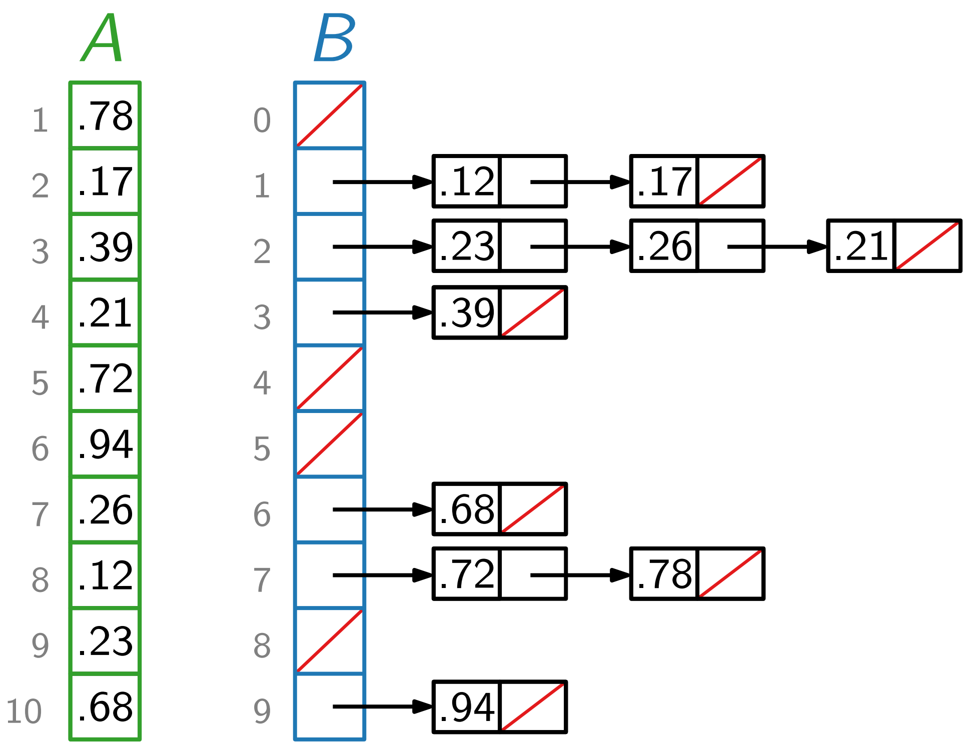


BucketSort(Feld A von Zahlen in $[0,1)$)

$n = A.\mathit{length}$
lege Feld $B[0 \ldots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein
for $i = 0$ **to** $n-1$ **do**
  sortiere Liste $B[i]$
hänge $B[0], \ldots, B[n-1]$ aneinander
kopiere das Ergebnis nach $A[1 \ldots n]$

$= [\frac{i}{n}, \frac{i+1}{n}) \cap A$

BucketSort



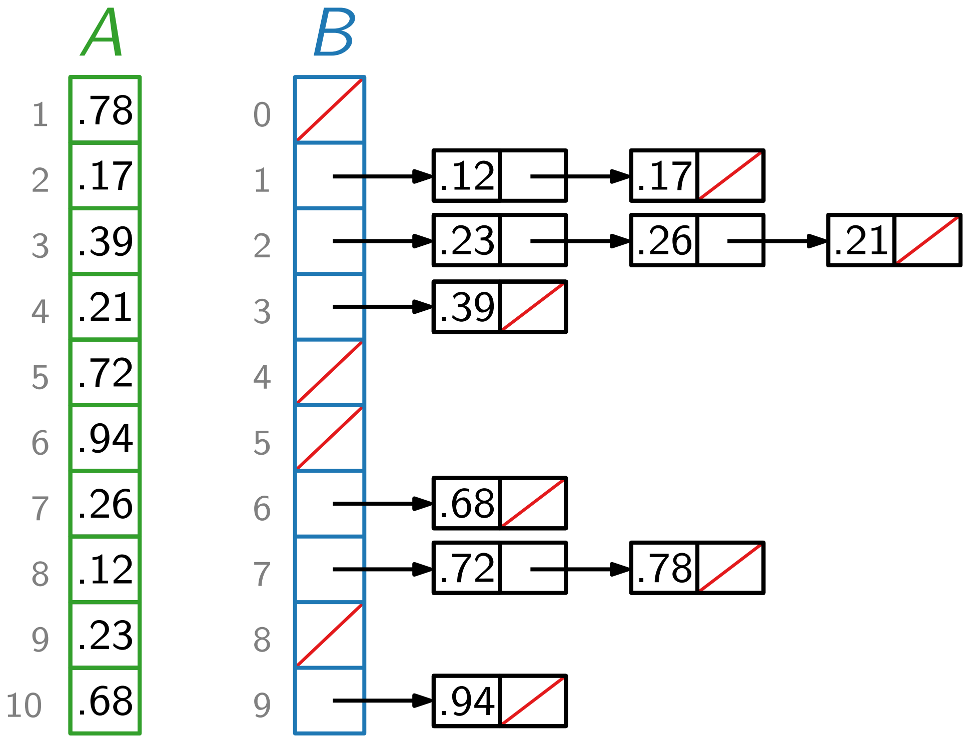
BucketSort(Feld A von Zahlen in $[0,1)$)

$n = A.length$
lege Feld $B[0 \dots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein
for $i = 0$ **to** $n-1$ **do**
  sortiere Liste $B[i]$
hänge $B[0], \dots, B[n-1]$ aneinander
kopiere das Ergebnis nach $A[1 \dots n]$

$= [\frac{i}{n}, \frac{i+1}{n}) \cap A$

Korrektheit? 2 Fälle:

BucketSort



BucketSort(Feld A von Zahlen in $[0, 1)$)

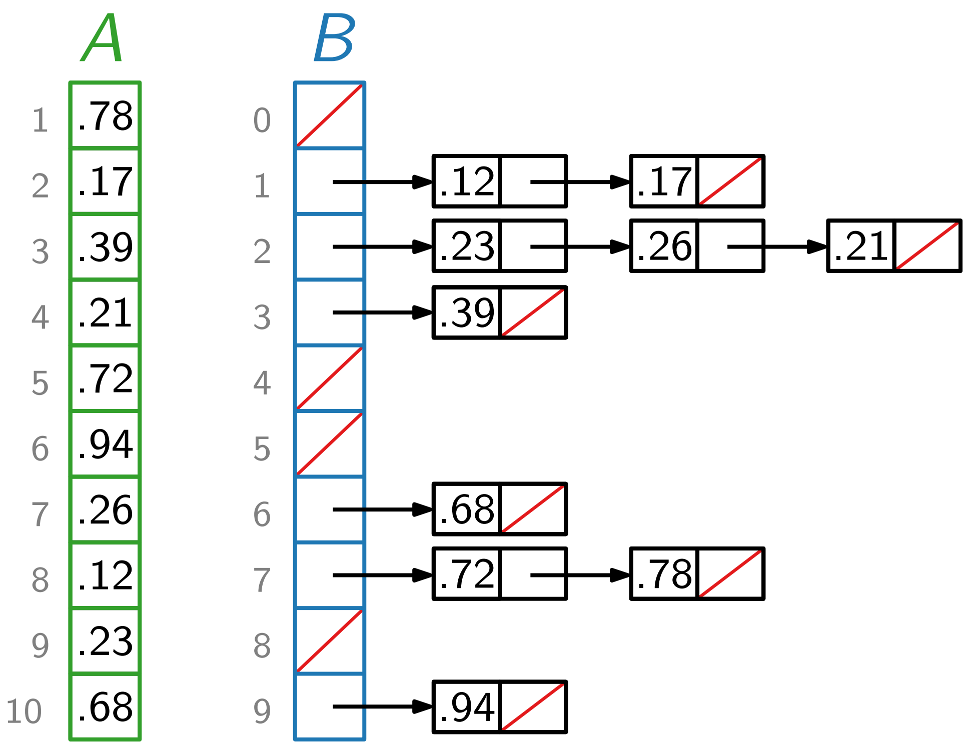
$n = A.length$
lege Feld $B[0 \dots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein
for $i = 0$ **to** $n - 1$ **do**
  sortiere Liste $B[i]$
hänge $B[0], \dots, B[n-1]$ aneinander
kopiere das Ergebnis nach $A[1 \dots n]$

$= [\frac{i}{n}, \frac{i+1}{n}) \cap A$

Korrektheit? 2 Fälle:

- $A[i]$ und $A[j]$ in der gleichen Liste
- $A[i]$ und $A[j]$ in verschiedenen Listen

BucketSort



BucketSort(Feld A von Zahlen in $[0,1)$)

$n = A.length$
lege Feld $B[0 \ldots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein
for $i = 0$ **to** $n-1$ **do**
  sortiere Liste $B[i]$
hänge $B[0], \ldots, B[n-1]$ aneinander
kopiere das Ergebnis nach $A[1 \ldots n]$

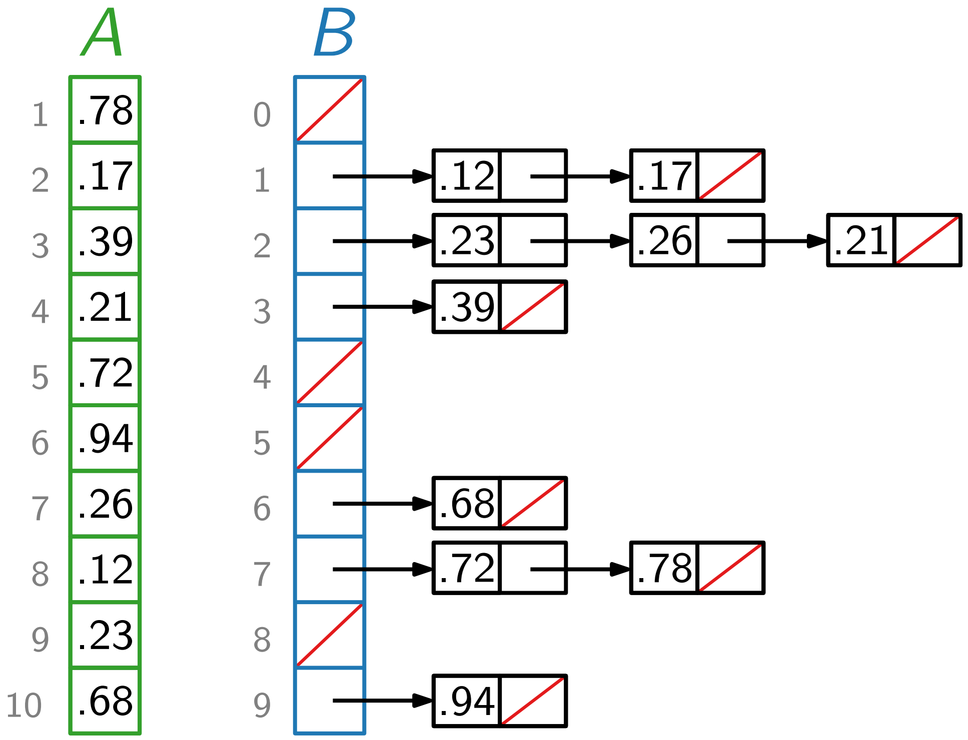
$= [\frac{i}{n}, \frac{i+1}{n}) \cap A$

Korrektheit? 2 Fälle:

- $A[i]$ und $A[j]$ in der gleichen Liste
- $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit?

BucketSort



BucketSort(Feld A von Zahlen in $[0,1)$)

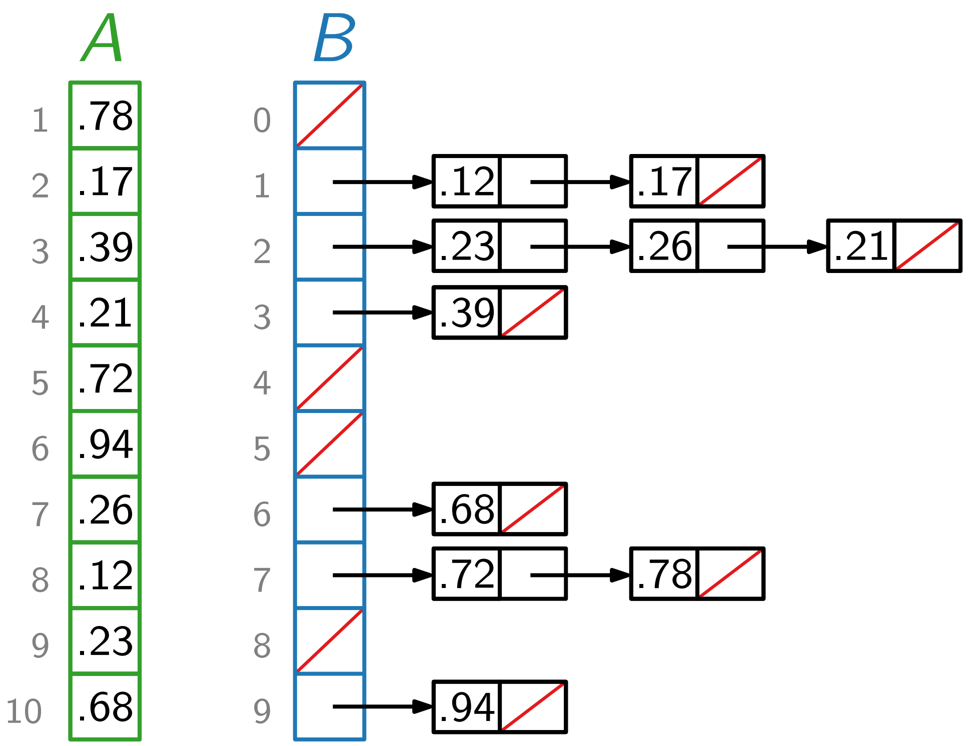
$n = A.length$
lege Feld $B[0\dots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein
for $i = 0$ **to** $n-1$ **do**
  sortiere Liste $B[i]$
hänge $B[0], \dots, B[n-1]$ aneinander
kopiere das Ergebnis nach $A[1 \dots n]$

$= [\frac{i}{n}, \frac{i+1}{n}) \cap A$

Korrektheit? 2 Fälle:
- $A[i]$ und $A[j]$ in der gleichen Liste
- $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit?
- **erwartet**, hängt von den zufälligen Zahlen in A ab

BucketSort



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$
lege Feld $B[0 \ldots n-1]$ von Listen an
for $j = 1$ **to** n **do**
 füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein
for $i = 0$ **to** $n-1$ **do**
 sortiere Liste $B[i]$
hänge $B[0], \ldots, B[n-1]$ aneinander
kopiere das Ergebnis nach $A[1 \ldots n]$

$= [\frac{i}{n}, \frac{i+1}{n}) \cap A$

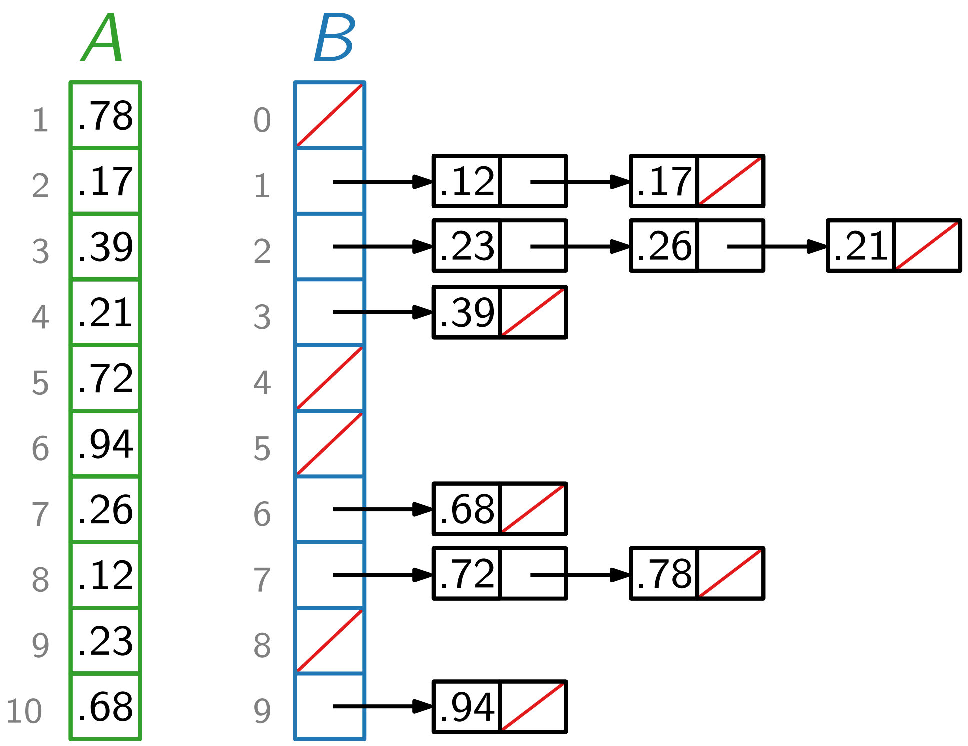
Korrektheit? 2 Fälle:

- $A[i]$ und $A[j]$ in der gleichen Liste
- $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit?

- **erwartet**, hängt von den zufälligen Zahlen in A ab
- hängt vom Sortieralgorithmus in Zeile 6 ab;

BucketSort



BucketSort(Feld A von Zahlen in $[0,1)$)

- $n = A.length$
- lege Feld $B[0 \ldots n-1]$ von Listen an
- **for** $j = 1$ **to** n **do**
  - füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein
- **for** $i = 0$ **to** $n-1$ **do**
  - sortiere Liste $B[i]$
- hänge $B[0], \ldots, B[n-1]$ aneinander
- kopiere das Ergebnis nach $A[1 \ldots n]$

$= [\frac{i}{n}, \frac{i+1}{n}) \cap A$

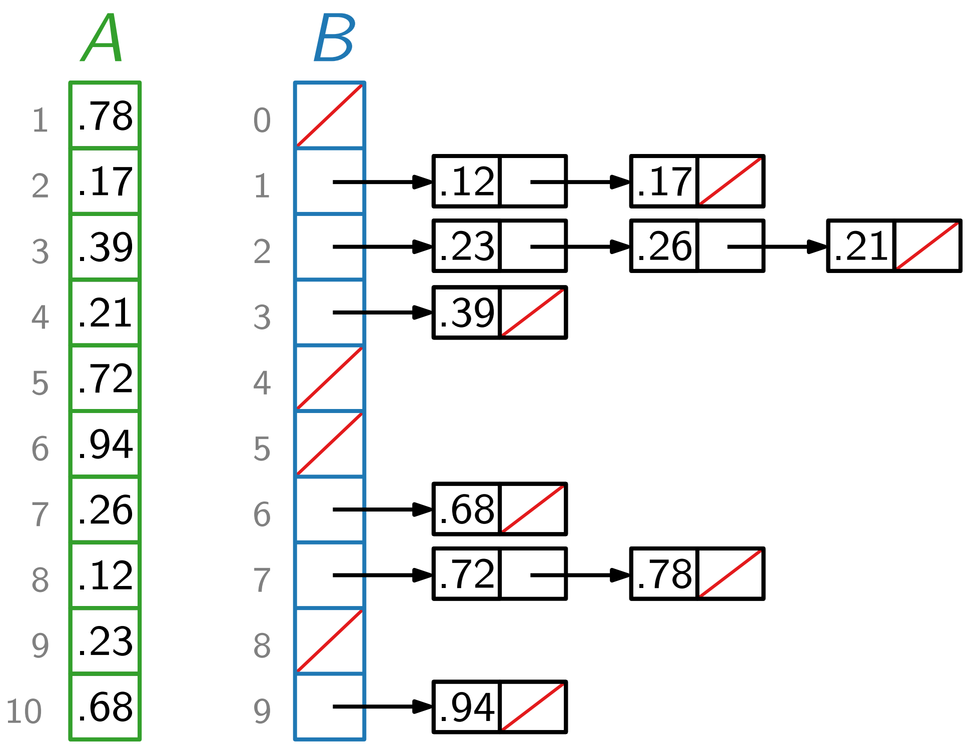
Korrektheit? 2 Fälle:

- $A[i]$ und $A[j]$ in der gleichen Liste
- $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit?

- **erwartet**, hängt von den zufälligen Zahlen in A ab
- hängt vom Sortieralgorithmus in Zeile 6 ab; wir nehmen InsertionSort:

BucketSort



BucketSort(Feld A von Zahlen in $[0,1)$)

$n = A.length$
lege Feld $B[0 \dots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein $= [\frac{i}{n}, \frac{i+1}{n}) \cap A$
for $i = 0$ **to** $n-1$ **do**
  sortiere Liste $B[i]$
hänge $B[0], \dots, B[n-1]$ aneinander
kopiere das Ergebnis nach $A[1 \dots n]$

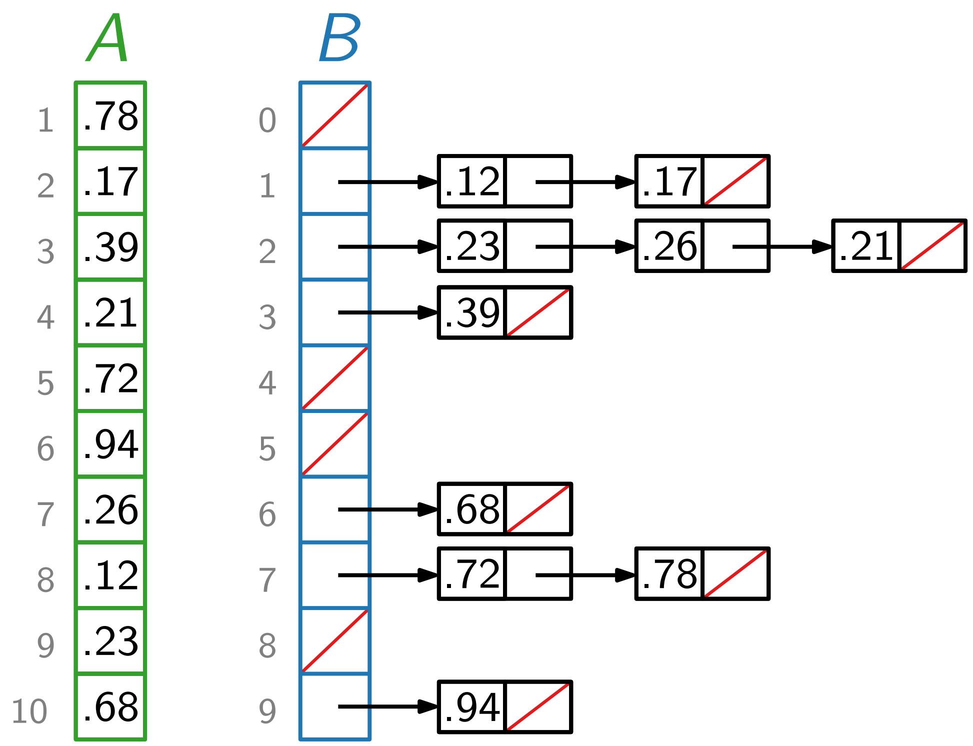
Korrektheit? 2 Fälle:

- $A[i]$ und $A[j]$ in der gleichen Liste
- $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit?

- **erwartet**, hängt von den zufälligen Zahlen in A ab
- hängt vom Sortieralgorithmus in Zeile 6 ab;
  wir nehmen InsertionSort: schnell auf kurzen Listen!

BucketSort



BucketSort(Feld A von Zahlen in $[0,1)$)

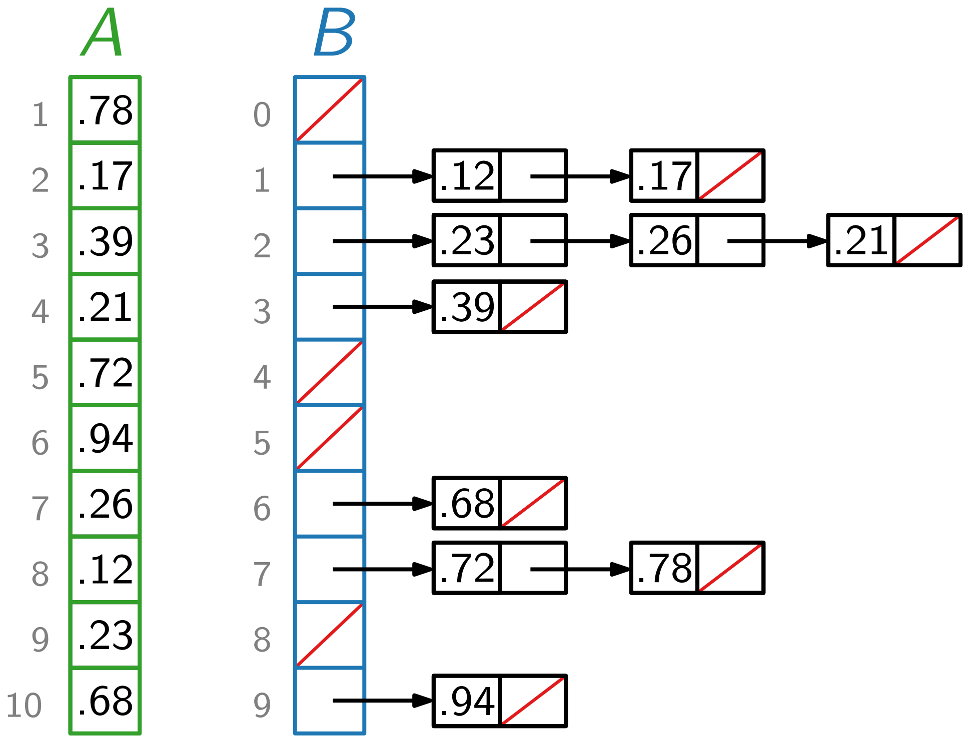
$n = A.\mathit{length}$
lege Feld $B[0 \ldots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein
for $i = 0$ **to** $n-1$ **do**
  sortiere Liste $B[i]$
hänge $B[0], \ldots, B[n-1]$ aneinander
kopiere das Ergebnis nach $A[1 \ldots n]$

Korrektheit? 2 Fälle:

- $A[i]$ und $A[j]$ in der gleichen Liste
- $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit? $T_{\mathrm{BS}}(n) =$

BucketSort



BucketSort(Feld A von Zahlen in $[0,1)$)

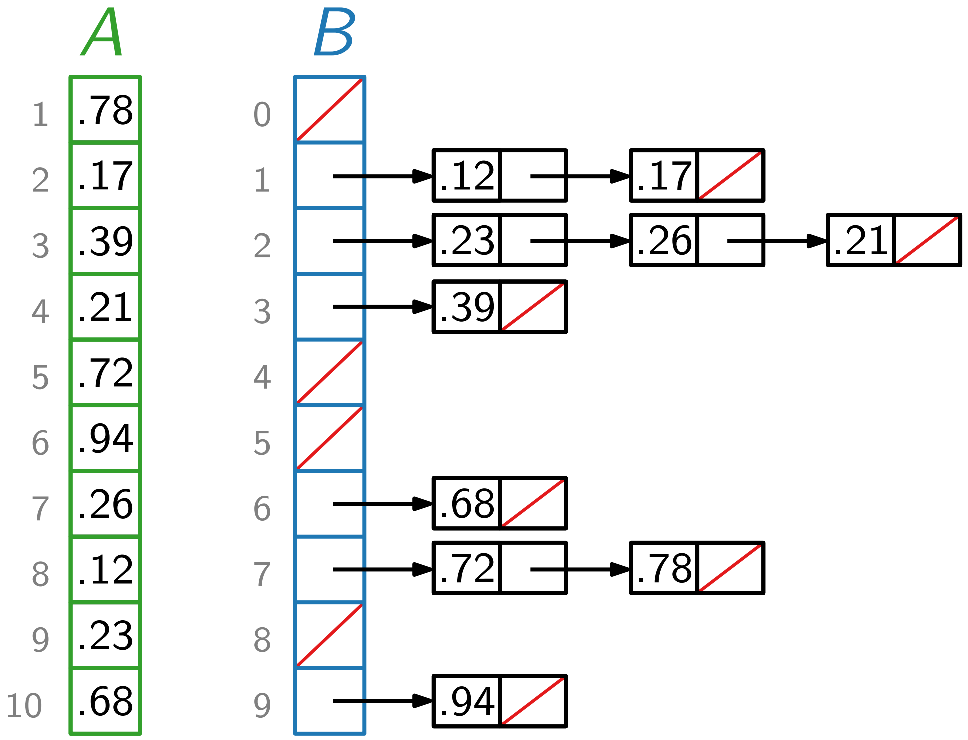
$n = A.length$
lege Feld $B[0 \ldots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein
for $i = 0$ **to** $n-1$ **do**
  sortiere Liste $B[i]$
hänge $B[0], \ldots, B[n-1]$ aneinander
kopiere das Ergebnis nach $A[1 \ldots n]$

Korrektheit? 2 Fälle:

- $A[i]$ und $A[j]$ in der gleichen Liste
- $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit? $T_{\mathrm{BS}}(n) =$

BucketSort



BucketSort(Feld A von Zahlen in $[0,1)$)

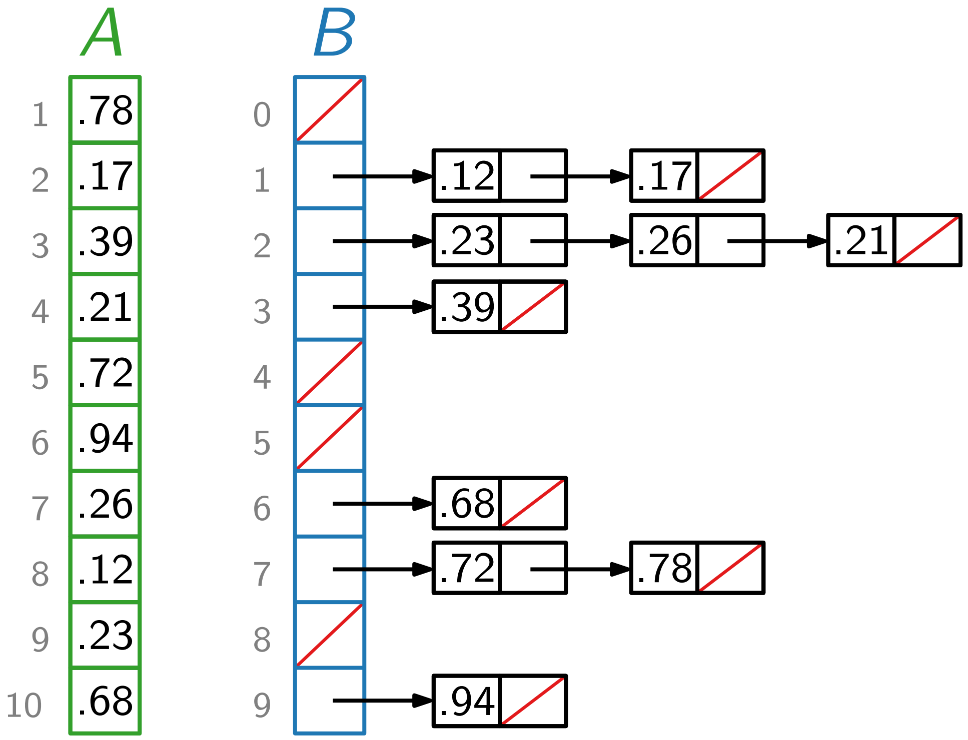
$n = A.\mathit{length}$
lege Feld $B[0\dots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein
for $i = 0$ **to** $n-1$ **do**
  sortiere Liste $B[i]$
hänge $B[0], \dots, B[n-1]$ aneinander
kopiere das Ergebnis nach $A[1\dots n]$

Korrektheit? 2 Fälle:

- $A[i]$ und $A[j]$ in der gleichen Liste
- $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit? $T_{\mathrm{BS}}(n) = \Theta(n) +$

BucketSort



BucketSort(Feld A von Zahlen in $[0,1)$)

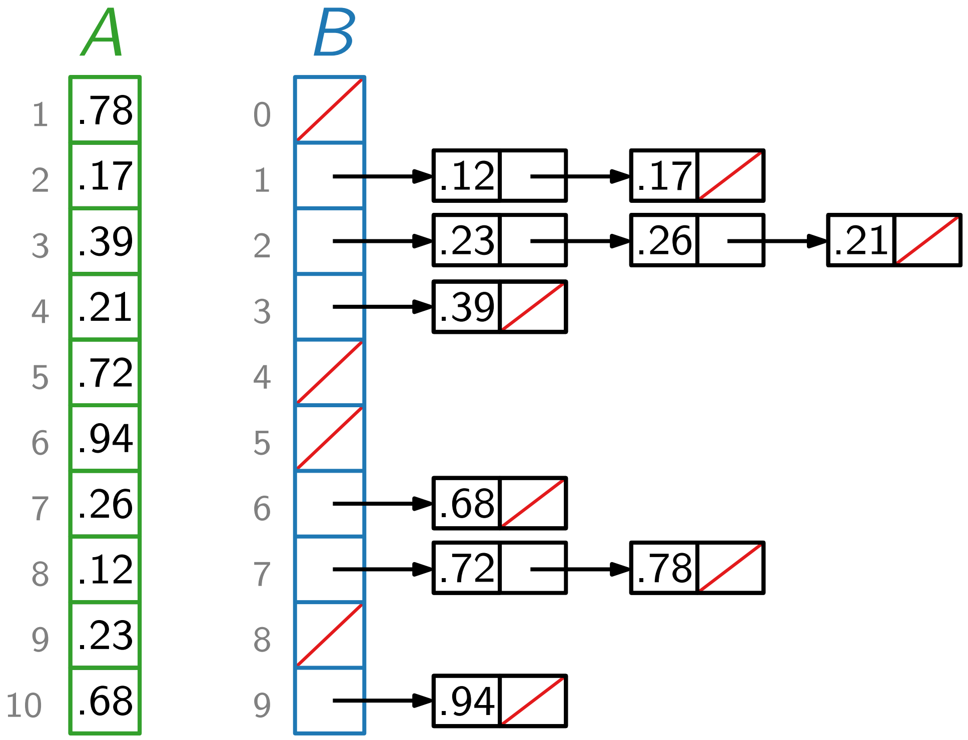
$n = A.\mathit{length}$
lege Feld $B[0 \ldots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein
for $i = 0$ **to** $n-1$ **do**
  sortiere Liste $B[i]$
hänge $B[0], \ldots, B[n-1]$ aneinander
kopiere das Ergebnis nach $A[1 \ldots n]$

Korrektheit? 2 Fälle:

- $A[i]$ und $A[j]$ in der gleichen Liste
- $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit? $T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i)$,

BUCKETSORT



BUCKETSORT(Feld A von Zahlen in $[0, 1)$)

$n = A.length$
lege Feld $B[0 \dots n-1]$ von Listen an
for $j = 1$ **to** n **do**
  füge $A[j]$ in Liste $B[\lfloor n \cdot A[j] \rfloor]$ ein
for $i = 0$ **to** $n-1$ **do**
  sortiere Liste $B[i]$
hänge $B[0], \dots, B[n-1]$ aneinander
kopiere das Ergebnis nach $A[1 \dots n]$

Korrektheit? 2 Fälle:

- $A[i]$ und $A[j]$ in der gleichen Liste
- $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit? $T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i)$,

wobei $T_{\mathrm{IS}}(\cdot)$ Laufzeit von INSERTIONSORT
n_i Zufallsvariable für Länge der Liste $B[i]$

Erwartete Laufzeit von BucketSort

$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i)$

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \;\; = \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \ = \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\mathrm{BS}}(n)] =$$

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \;= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\mathrm{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \; = \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\mathrm{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X + Y] = E[X] + E[Y]$

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \;= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\mathrm{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X+Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \;= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\mathrm{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X+Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Für $a \in \mathbb{R}:\; E[aX] = a \cdot E[X]$

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \;= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\mathrm{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X+Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Für $a \in \mathbb{R}:\ E[aX] = a \cdot E[X]$

$$= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(E[n_i^2])$$

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \;\; = \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\mathrm{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X+Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Für $a \in \mathbb{R}$: $E[aX] = a \cdot E[X]$

$$= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(E[n_i^2]) \;\; =$$

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \;=\; \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\mathrm{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X+Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Für $a \in \mathbb{R}$: $E[aX] = a \cdot E[X]$

$$= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(E[n_i^2]) \;=$$

Behauptung: $E[n_i^2] \;\leq\; 2 - \frac{1}{n}$

Erwartete Laufzeit von BucketSort

$$T_{BS}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{IS}(n_i) \;=\; \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{BS}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X+Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Für $a \in \mathbb{R}$: $E[aX] = a \cdot E[X]$

$$= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(E[n_i^2]) \;=\; \Theta(n)$$

Behauptung: $E[n_i^2] \;\leq\; 2 - \frac{1}{n}$

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \;=\; \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\mathrm{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X+Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Für $a \in \mathbb{R}$: $E[aX] = a \cdot E[X]$

$$= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(E[n_i^2]) \;=\; \Theta(n)$$

Behauptung: $E[n_i^2] \;\leq\; 2 - \frac{1}{n}$

Beweis.

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \;= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\mathrm{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X+Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Für $a \in \mathbb{R}:\; E[aX] = a \cdot E[X]$

$$= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(E[n_i^2]) \;=\; \Theta(n)$$

Behauptung: $E[n_i^2] \;\leq\; 2 - \frac{1}{n}$

Beweis. Definiere Indikator-Zufallsvariable $X_j := 1$, falls $A[j]$ in Eimer i fällt.

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \;=\; \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\mathrm{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X+Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Für $a \in \mathbb{R}$: $E[aX] = a \cdot E[X]$

$$= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(E[n_i^2]) \;=\; \Theta(n)$$

Behauptung: $E[n_i^2] \;\leq\; 2 - \frac{1}{n}$

Beweis. Definiere Indikator-Zufallsvariable $X_j := 1$, falls $A[j]$ in Eimer i fällt. fest!

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \;=\; \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\mathrm{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X+Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Für $a \in \mathbb{R}$: $E[aX] = a \cdot E[X]$

$$= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(E[n_i^2]) \;=\; \Theta(n)$$

Behauptung: $E[n_i^2] \;\leq\; 2 - \frac{1}{n}$

fest!

Beweis. Definiere Indikator-Zufallsvariable $X_j := 1$, falls $A[j]$ in Eimer i fällt.

$\Rightarrow\; n_i =$

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \;=\; \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\mathrm{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X+Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Für $a \in \mathbb{R}: \; E[aX] = a \cdot E[X]$

$$= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(E[n_i^2]) \;=\; \Theta(n)$$

Behauptung: $E[n_i^2] \;\leq\; 2 - \frac{1}{n}$

fest!

Beweis. Definiere Indikator-Zufallsvariable $X_j := 1$, falls $A[j]$ in Eimer i fällt.

$$\Rightarrow \; n_i \;=\; \sum_{j=1}^{n} X_j$$

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \;=\; \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\mathrm{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X+Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Für $a \in \mathbb{R}$: $E[aX] = a \cdot E[X]$

$$= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(E[n_i^2]) \;=\; \Theta(n)$$

Behauptung: $E[n_i^2] \;\leq\; 2 - \frac{1}{n}$

Beweis. Definiere Indikator-Zufallsvariable $X_j := 1$, falls $A[j]$ in Eimer i fällt. fest!

$$\Rightarrow \; n_i \;=\; \sum_{j=1}^{n} X_j \qquad E[X_j] =$$

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \;=\; \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\mathrm{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X+Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Für $a \in \mathbb{R}$: $E[aX] = a \cdot E[X]$

$$= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(E[n_i^2]) \;=\; \Theta(n)$$

Behauptung: $E[n_i^2] \;\leq\; 2 - \frac{1}{n}$

fest!

Beweis. Definiere Indikator-Zufallsvariable $X_j := 1$, falls $A[j]$ in Eimer i fällt.

$$\Rightarrow \; n_i \;=\; \sum_{j=1}^{n} X_j \qquad E[X_j] = \Pr[X_j = 1] =$$

Erwartete Laufzeit von BucketSort

$$T_{\mathrm{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\mathrm{IS}}(n_i) \; = \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\mathrm{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X+Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Für $a \in \mathbb{R}$: $E[aX] = a \cdot E[X]$

$$= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(E[n_i^2]) \;=\; \Theta(n)$$

Behauptung: $E[n_i^2] \;\leq\; 2 - \frac{1}{n}$

Beweis. Definiere Indikator-Zufallsvariable $X_j := 1$, falls $A[j]$ in Eimer i fällt.

fest!

$$\Rightarrow \; n_i \;=\; \sum_{j=1}^{n} X_j \qquad E[X_j] = \Pr[X_j = 1] = 1/n$$

Erwartete Laufzeit von BucketSort

$$T_{BS}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{IS}(n_i) \;= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{BS}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X+Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Für $a \in \mathbb{R}$: $E[aX] = a \cdot E[X]$

$$= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(E[n_i^2]) \;=\; \Theta(n)$$

Behauptung: $E[n_i^2] \;\leq\; 2 - \frac{1}{n}$

fest!

Beweis. Definiere Indikator-Zufallsvariable $X_j := 1$, falls $A[j]$ in Eimer i fällt.

$$\Rightarrow\; n_i \;=\; \sum_{j=1}^{n} X_j \qquad E[X_j] = \Pr[X_j = 1] = 1/n$$

$$\Rightarrow\; n_i^2 = \left(\sum_{j=1}^{n} X_j\right)^2 =$$

Erwartete Laufzeit von BucketSort

$$T_{BS}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{IS}(n_i) \;= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{BS}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X+Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Für $a \in \mathbb{R}$: $E[aX] = a \cdot E[X]$

$$= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(E[n_i^2]) \;=\; \Theta(n)$$

Behauptung: $E[n_i^2] \;\leq\; 2 - \frac{1}{n}$

fest!

Beweis. Definiere Indikator-Zufallsvariable $X_j := 1$, falls $A[j]$ in Eimer i fällt.

$$\Rightarrow\; n_i \;=\; \sum_{j=1}^{n} X_j \qquad E[X_j] = \Pr[X_j = 1] = 1/n$$

$$\Rightarrow\; n_i^2 = \left(\sum_{j=1}^{n} X_j\right)^2 = \sum_{j=1}^{n}\sum_{k=1}^{n} X_j X_k$$

Erwartete Laufzeit von BucketSort

$$T_{\text{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\text{IS}}(n_i) \;= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)$$

$$E[T_{\text{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(n_i^2)]$$

Linearität des Erwartungswerts: $E[X + Y] = E[X] + E[Y]$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[\mathcal{O}(n_i^2)]$$

Für $a \in \mathbb{R}: \; E[aX] = a \cdot E[X]$

$$= \Theta(n) + \sum_{i=0}^{n-1} \mathcal{O}(E[n_i^2]) \;=\; \Theta(n)$$

Behauptung: $E[n_i^2] \;\leq\; 2 - \frac{1}{n}$

Beweis. Definiere Indikator-Zufallsvariable $X_j := 1$, falls $A[j]$ in Eimer i fällt. fest!

$$\Rightarrow\; n_i \;=\; \sum_{j=1}^{n} X_j \qquad E[X_j] = \Pr[X_j = 1] = 1/n$$

$$\Rightarrow\; n_i^2 = \left(\sum_{j=1}^{n} X_j\right)^2 = \sum_{j=1}^{n} \sum_{k=1}^{n} X_j X_k$$

$$= \sum_{j=1}^{n} X_j^2 + \sum_{j=1}^{n} \sum_{k \neq j} X_j X_k$$

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^{n} X_j^2 + \sum_{j=1}^{n} \sum_{k \neq j} X_j X_k$

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^{n} X_j^2 + \sum_{j=1}^{n} \sum_{k \neq j} X_j X_k$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^{n} X_j^2 + \sum_{j=1}^{n} \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] =$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Behandle die beiden Typen von Erwartungswerten getrennt:

Erwartete Laufzeit von BucketSort

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Es gilt $n_i^2 = \sum_{j=1}^{n} X_j^2 + \sum_{j=1}^{n} \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^{n} E[X_j^2] + \sum_{j=1}^{n} \sum_{k \neq j} E[X_j X_k]$

Behandle die beiden Typen von Erwartungswerten getrennt:

$E[X_j^2] \quad =$

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^{n} X_j^2 + \sum_{j=1}^{n} \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^{n} E[X_j^2] + \sum_{j=1}^{n} \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Behandle die beiden Typen von Erwartungswerten getrennt:

$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^{n} X_j^2 + \sum_{j=1}^{n} \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^{n} E[X_j^2] + \sum_{j=1}^{n} \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$
$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0]$$

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^{n} X_j^2 + \sum_{j=1}^{n} \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^{n} E[X_j^2] + \sum_{j=1}^{n} \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$

$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n}$$

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n} \end{aligned}$$

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^{n} X_j^2 + \sum_{j=1}^{n} \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^{n} E[X_j^2] + \sum_{j=1}^{n} \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\boxed{E[X_j^2]} = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$

$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \boxed{\frac{1}{n}}$$

unabhängig von j

Erwartete Laufzeit von BucketSort

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Es gilt $n_i^2 = \sum_{j=1}^{n} X_j^2 + \sum_{j=1}^{n} \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^{n} E[X_j^2] + \sum_{j=1}^{n} \sum_{k \neq j} E[X_j X_k]$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\boxed{E[X_j^2]} = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$

$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \boxed{\frac{1}{n}}$$

unabhängig von j

$E[X_j X_k] =$

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^{n} X_j^2 + \sum_{j=1}^{n} \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^{n} E[X_j^2] + \sum_{j=1}^{n} \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\boxed{E[X_j^2]} = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$

$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \boxed{\frac{1}{n}}$$

unabhängig von j

$E[X_j X_k] =$

für $j \neq k$ sind X_j und X_k **unabhängig**

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\boxed{E[X_j^2]} = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$

$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \boxed{\frac{1}{n}}$$

unabhängig von j

$$E[X_j X_k] = E[X_j] \cdot E[X_k]$$

für $j \neq k$ sind X_j und X_k **unabhängig**

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\boxed{E[X_j^2]} = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$

$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \boxed{\frac{1}{n}}$$

unabhängig von j

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

für $j \neq k$ sind X_j und X_k **unabhängig**

Erwartete Laufzeit von BUCKETSORT

Es gilt $n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\boxed{E[X_j^2]} = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$

unabhängig von j

$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \boxed{\frac{1}{n}}$$

$$\boxed{E[X_j X_k]} = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \boxed{\frac{1}{n^2}}$$

für $j \neq k$ sind X_j und X_k **unabhängig**

Erwartete Laufzeit von BucketSort

Behauptung:
$E[n_i^2] \le 2 - \frac{1}{n}$

Es gilt $n_i^2 = \sum_{j=1}^{n} X_j^2 + \sum_{j=1}^{n} \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^{n} E[X_j^2] + \sum_{j=1}^{n} \sum_{k \neq j} E[X_j X_k]$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$
$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n}$$

unabhängig von j

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

unabhängig von j und k

für $j \neq k$ sind X_j und X_k **unabhängig**

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^{n} X_j^2 + \sum_{j=1}^{n} \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^{n} E[X_j^2] + \sum_{j=1}^{n} \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\boxed{E[X_j^2]} = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$

$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \boxed{\frac{1}{n}}$$

unabhängig von j

$$\boxed{E[X_j X_k]} = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \boxed{\frac{1}{n^2}}$$

unabhängig von j und k

für $j \neq k$ sind X_j und X_k **unabhängig**

Fasse die Zwischenergebnisse zusammen:

Erwartete Laufzeit von BucketSort

Behauptung:
$E[n_i^2] \le 2 - \frac{1}{n}$

Es gilt $n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$
$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n}$$

unabhängig von j

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

unabhängig von j und k

für $j \neq k$ sind X_j und X_k **unabhängig**

Fasse die Zwischenergebnisse zusammen:

$$E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^{n} X_j^2 + \sum_{j=1}^{n} \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^{n} E[X_j^2] + \sum_{j=1}^{n} \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$
$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n}$$

unabhängig von j

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

unabhängig von j und k

für $j \neq k$ sind X_j und X_k **unabhängig**

Fasse die Zwischenergebnisse zusammen:

$$E[n_i^2] = \sum_{j=1}^{n} E[X_j^2] + \sum_{j=1}^{n} \sum_{k \neq j} E[X_j X_k]$$

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$
$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n}$$

unabhängig von j

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

unabhängig von j und k

für $j \neq k$ sind X_j und X_k **unabhängig**

Fasse die Zwischenergebnisse zusammen:

$$E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$
$$=$$

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$
$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n}$$

unabhängig von j

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

unabhängig von j und k

für $j \neq k$ sind X_j und X_k **unabhängig**

Fasse die Zwischenergebnisse zusammen:

$$E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$
$$= \quad \cdot \frac{1}{n} + \quad \cdot \quad \cdot \frac{1}{n^2}$$

Erwartete Laufzeit von BucketSort

Behauptung:
$E[n_i^2] \le 2 - \frac{1}{n}$

Es gilt $n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n} \end{aligned}$$

unabhängig von j

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

unabhängig von j und k

für $j \neq k$ sind X_j und X_k **unabhängig**

Fasse die Zwischenergebnisse zusammen:

$$\begin{aligned} E[n_i^2] &= \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k] \\ &= n \cdot \frac{1}{n} + n \cdot \qquad \cdot \frac{1}{n^2} \end{aligned}$$

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^{n} X_j^2 + \sum_{j=1}^{n} \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^{n} E[X_j^2] + \sum_{j=1}^{n} \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$
$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n}$$

unabhängig von j

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

unabhängig von j und k

für $j \neq k$ sind X_j und X_k **unabhängig**

Fasse die Zwischenergebnisse zusammen:

$$E[n_i^2] = \sum_{j=1}^{n} E[X_j^2] + \sum_{j=1}^{n} \sum_{k \neq j} E[X_j X_k]$$
$$= n \cdot \frac{1}{n} + n \cdot (n-1) \cdot \frac{1}{n^2} =$$

Erwartete Laufzeit von BucketSort

Behauptung:
$E[n_i^2] \le 2 - \frac{1}{n}$

Es gilt $\quad n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$

$$\Rightarrow \quad E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$
$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n}$$

unabhängig von j

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

unabhängig von j und k

für $j \neq k$ sind X_j und X_k **unabhängig**

Fasse die Zwischenergebnisse zusammen:

$$E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$
$$= n \cdot \frac{1}{n} + n \cdot (n-1) \cdot \frac{1}{n^2} = 1 + \frac{n-1}{n} =$$

Erwartete Laufzeit von BucketSort

Es gilt $n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$

$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$

Behauptung:
$E[n_i^2] \leq 2 - \frac{1}{n}$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$
$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n}$$

unabhängig von j

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

unabhängig von j und k

für $j \neq k$ sind X_j und X_k **unabhängig**

Fasse die Zwischenergebnisse zusammen:

$$E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$
$$= n \cdot \frac{1}{n} + n \cdot (n-1) \cdot \frac{1}{n^2} = 1 + \frac{n-1}{n} = 2 - \frac{1}{n} \qquad \square$$

Zusammenfassung

- Jedes vergleichsbasierte Sortierverfahren braucht im Worst-Case $\Omega(n \log n)$ Vergleiche.

- CountingSort sortiert Zahlen in $\{0, \ldots, k\}$. (**stabil!**)
  Laufzeit für n Zahlen: $\mathcal{O}(n + k)$

- RadixSort sortiert s-stellige b-adische Zahlen.
  Laufzeit für n Zahlen: $\mathcal{O}(s \cdot (n + b))$

- BucketSort sortiert gleichverteilte zufällige Zahlen.
  Erwartete Laufzeit für n Zahlen: $\mathcal{O}(n)$

Zusammenfassung

- Jedes vergleichsbasierte Sortierverfahren braucht im Worst-Case $\Omega(n \log n)$ Vergleiche.

- COUNTINGSORT sortiert Zahlen in $\{0, \ldots, k\}$. (**stabil!**)
  Laufzeit für n Zahlen: $\mathcal{O}(n + k)$

- RADIXSORT sortiert s-stellige b-adische Zahlen.
  Laufzeit für n Zahlen: $\mathcal{O}(s \cdot (n + b))$

- BUCKETSORT sortiert gleichverteilte zufällige Zahlen.
  Erwartete Laufzeit für n Zahlen: $\mathcal{O}(n)$

Bemerkung. Die Idee mit den (gleichgroßen) Eimern ist natürlich nicht nur auf Zufallszahlen beschränkt, aber hier lässt sie sich hübsch analysieren.

Vergleich Sortieralgorithmen

	Bester Fall	Erw. Fall	Schl. Fall	in-situ	stabil
InsertionSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
SelectionSort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✗
BubbleSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
MergeSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✗	✓
HeapSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✓	✗
QuickSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$	✓	✗

Vergleich Sortieralgorithmen

	Bester Fall	Erw. Fall	Schl. Fall	in-situ	stabil
InsertionSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
SelectionSort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✗
BubbleSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
MergeSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✗	✓
HeapSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✓	✗
QuickSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$	✓	✗
CountingSort					
RadixSort					
BucketSort					

Vergleich Sortieralgorithmen

	Bester Fall	Erw. Fall	Schl. Fall	in-situ	stabil
InsertionSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
SelectionSort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✗
BubbleSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
MergeSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✗	✓
HeapSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✓	✗
QuickSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$	✓	✗
CountingSort	$\mathcal{O}(n+k)$	$\mathcal{O}(n+k)$	$\mathcal{O}(n+k)$		
RadixSort					
BucketSort					

Vergleich Sortieralgorithmen

	Bester Fall	Erw. Fall	Schl. Fall	in-situ	stabil
InsertionSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
SelectionSort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✗
BubbleSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
MergeSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✗	✓
HeapSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✓	✗
QuickSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$	✓	✗
CountingSort	$\mathcal{O}(n + k)$	$\mathcal{O}(n + k)$	$\mathcal{O}(n + k)$		
RadixSort	$\mathcal{O}(s \cdot (n + b))$	$\mathcal{O}(s \cdot (n + b))$	$\mathcal{O}(s \cdot (n + b))$		
BucketSort					

Vergleich Sortieralgorithmen

	Bester Fall	Erw. Fall	Schl. Fall	in-situ	stabil
InsertionSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
SelectionSort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✗
BubbleSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
MergeSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✗	✓
HeapSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✓	✗
QuickSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$	✓	✗
CountingSort	$\mathcal{O}(n + k)$	$\mathcal{O}(n + k)$	$\mathcal{O}(n + k)$		
RadixSort	$\mathcal{O}(s \cdot (n + b))$	$\mathcal{O}(s \cdot (n + b))$	$\mathcal{O}(s \cdot (n + b))$		
BucketSort	$\mathcal{O}(n)$	$\mathcal{O}(n)$			

Vergleich Sortieralgorithmen

	Bester Fall	Erw. Fall	Schl. Fall	in-situ	stabil
InsertionSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
SelectionSort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✗
BubbleSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
MergeSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✗	✓
HeapSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✓	✗
QuickSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$	✓	✗
CountingSort	$\mathcal{O}(n + k)$	$\mathcal{O}(n + k)$	$\mathcal{O}(n + k)$		
RadixSort	$\mathcal{O}(s \cdot (n + b))$	$\mathcal{O}(s \cdot (n + b))$	$\mathcal{O}(s \cdot (n + b))$		
BucketSort	$\mathcal{O}(n)$	$\mathcal{O}(n)$ wenn Eingabe zufällig und gleichverteilt			

Vergleich Sortieralgorithmen

	Bester Fall	Erw. Fall	Schl. Fall	in-situ	stabil
InsertionSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
SelectionSort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✗
BubbleSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
MergeSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✗	✓
HeapSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✓	✗
QuickSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$	✓	✗
CountingSort	$\mathcal{O}(n + k)$	$\mathcal{O}(n + k)$	$\mathcal{O}(n + k)$		
RadixSort	$\mathcal{O}(s \cdot (n + b))$	$\mathcal{O}(s \cdot (n + b))$	$\mathcal{O}(s \cdot (n + b))$		
BucketSort	$\mathcal{O}(n)$	$\mathcal{O}(n)$ wenn Eingabe zufällig und gleichverteilt	$\mathcal{O}(n^2)$		

Vergleich Sortieralgorithmen

	Bester Fall	Erw. Fall	Schl. Fall	in-situ	stabil
InsertionSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
SelectionSort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✗
BubbleSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
MergeSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✗	✓
HeapSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✓	✗
QuickSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$	✓	✗
CountingSort	$\mathcal{O}(n+k)$	$\mathcal{O}(n+k)$	$\mathcal{O}(n+k)$	✗	
RadixSort	$\mathcal{O}(s \cdot (n+b))$	$\mathcal{O}(s \cdot (n+b))$	$\mathcal{O}(s \cdot (n+b))$	✗	
BucketSort	$\mathcal{O}(n)$	$\mathcal{O}(n)$ wenn Eingabe zufällig und gleichverteilt	$\mathcal{O}(n^2)$	✗	

Vergleich Sortieralgorithmen

	Bester Fall	Erw. Fall	Schl. Fall	in-situ	stabil
InsertionSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
SelectionSort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✗
BubbleSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
MergeSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✗	✓
HeapSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✓	✗
QuickSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$	✓	✗
CountingSort	$\mathcal{O}(n + k)$	$\mathcal{O}(n + k)$	$\mathcal{O}(n + k)$	✗	✓
RadixSort	$\mathcal{O}(s \cdot (n + b))$	$\mathcal{O}(s \cdot (n + b))$	$\mathcal{O}(s \cdot (n + b))$	✗	✓
BucketSort	$\mathcal{O}(n)$	$\mathcal{O}(n)$ wenn Eingabe zufällig und gleichverteilt	$\mathcal{O}(n^2)$	✗	

Vergleich Sortieralgorithmen

	Bester Fall	Erw. Fall	Schl. Fall	in-situ	stabil
InsertionSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
SelectionSort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✗
BubbleSort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	✓	✓
MergeSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✗	✓
HeapSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	✓	✗
QuickSort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$	✓	✗
CountingSort	$\mathcal{O}(n+k)$	$\mathcal{O}(n+k)$	$\mathcal{O}(n+k)$	✗	✓
RadixSort	$\mathcal{O}(s \cdot (n+b))$	$\mathcal{O}(s \cdot (n+b))$	$\mathcal{O}(s \cdot (n+b))$	✗	✓
BucketSort	$\mathcal{O}(n)$	$\mathcal{O}(n)$ wenn Eingabe zufällig und gleichverteilt	$\mathcal{O}(n^2)$	✗	✓ wenn verwendeter Sortieralg. stabil