

Algorithmen und Datenstrukturen

Wintersemester 2019/20

9. Vorlesung

Sortieren in Linearzeit

Anmeldung!

Melden Sie sich unbedingt unter WueCampus an zur

Teilnahme am 1. Kurztest am 21.11.

Anmeldung!

Melden Sie sich unbedingt unter WueCampus an zur

Teilnahme am 1. Kurztest am 21.11.

...sonst können Sie nicht teilnehmen!

Anmeldung!

Melden Sie sich unbedingt unter WueCampus an zur

Teilnahme am 1. Kurztest am 21.11.

...sonst können Sie nicht teilnehmen!

Anmeldefrist: Mo, 18.11., 23:59 Uhr

... aber besser gleich *jetzt!*

Event!

Auslandsaufenthalte für Studierende der Informatik

Infoveranstaltung, 14.11., Turing-HS

- 16:00 Partnerschaften und Fördermöglichkeiten der Uni
Würzburg
(Susanne Holzheimer, International Students Office)
- 16:30 Praktika und Abschlussarbeiten im Ausland
(Prof. Dr. Tobias Hoßfeld und Kathrin Borchert)
- 16:45 Partnerschaften in der Informatik: Konkrete Beispiele
(Prof. Dr. Alexander Wolff)
- 17:00 Einzelgespräche

Sortieren durch Vergleichen

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

Sortieren durch Vergleichen

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

Sortieren durch Vergleichen

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

vergleichsbasierter Sortieralg.

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* Sortieralg. charakterisiert durch seinen *Entscheidungsbaum*:

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* Sortieralg. charakterisiert durch seinen *Entscheidungsbaum*:

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

Sortieren durch Vergleichen

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

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



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

Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter* Sortieralg. 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 $\langle a_1, a_2, \dots, a_n \rangle$ $\xrightarrow[\text{Schlüsselvergleiche}]{\text{Sortieralg.}}$ Ausgabe: sortierte Eingabe

Für festes n ist ein *vergleichsbasierter* Sortieralg. 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

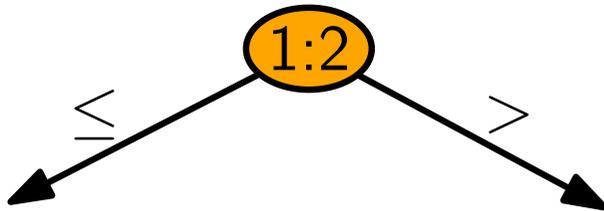
Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter* Sortieralg. 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



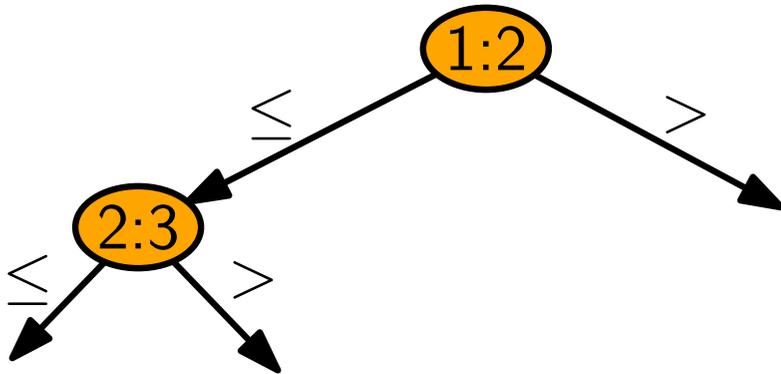
Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter Sortieralg.* 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



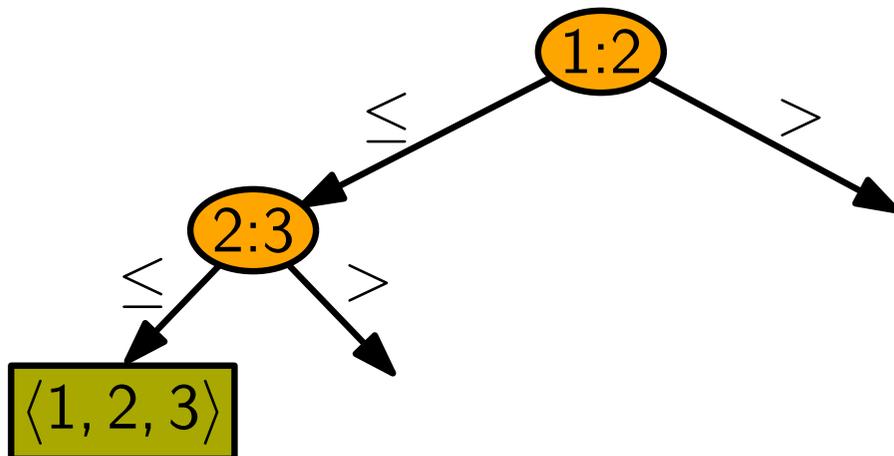
Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter Sortieralg.* 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



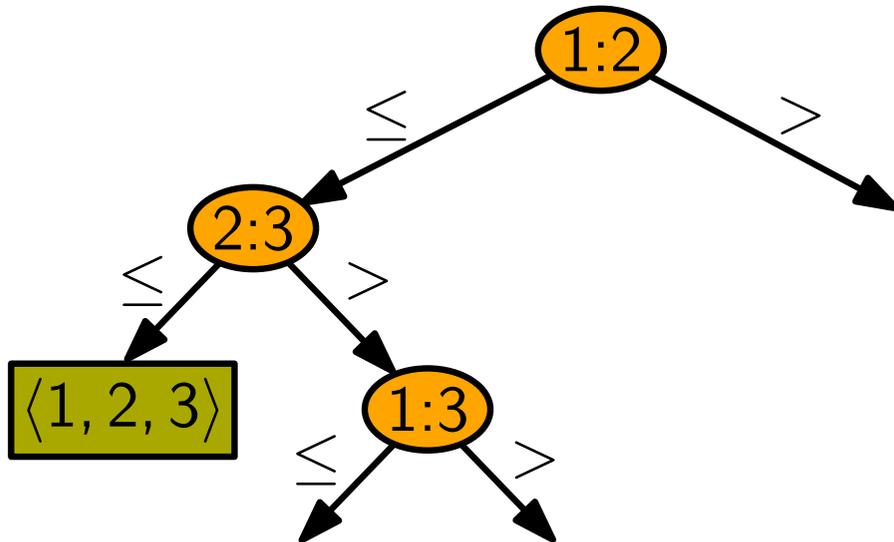
Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter* Sortieralg. 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



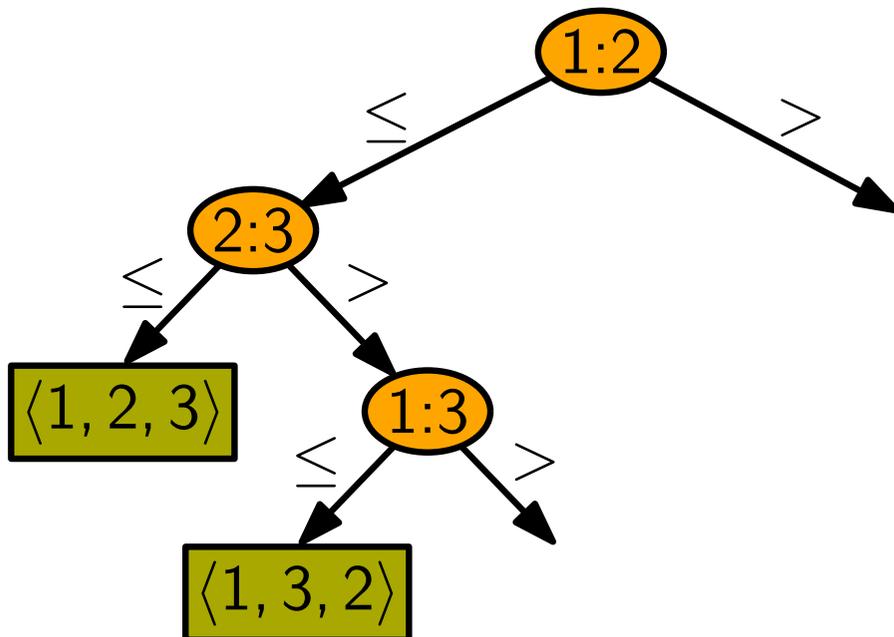
Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter Sortieralg.* 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 $n = 3$ [CLRS]

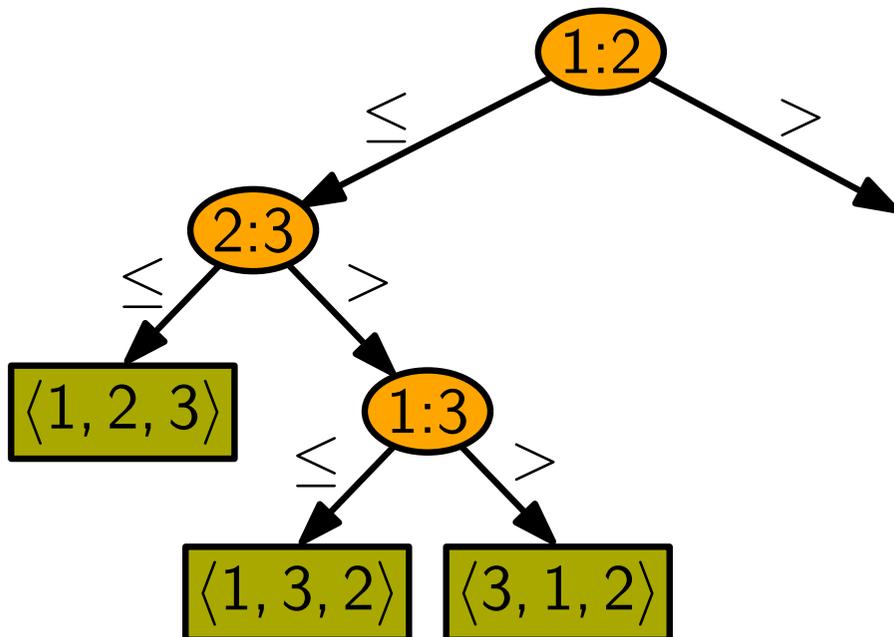
Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter Sortieralg.* 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 $n = 3$ [CLRS]

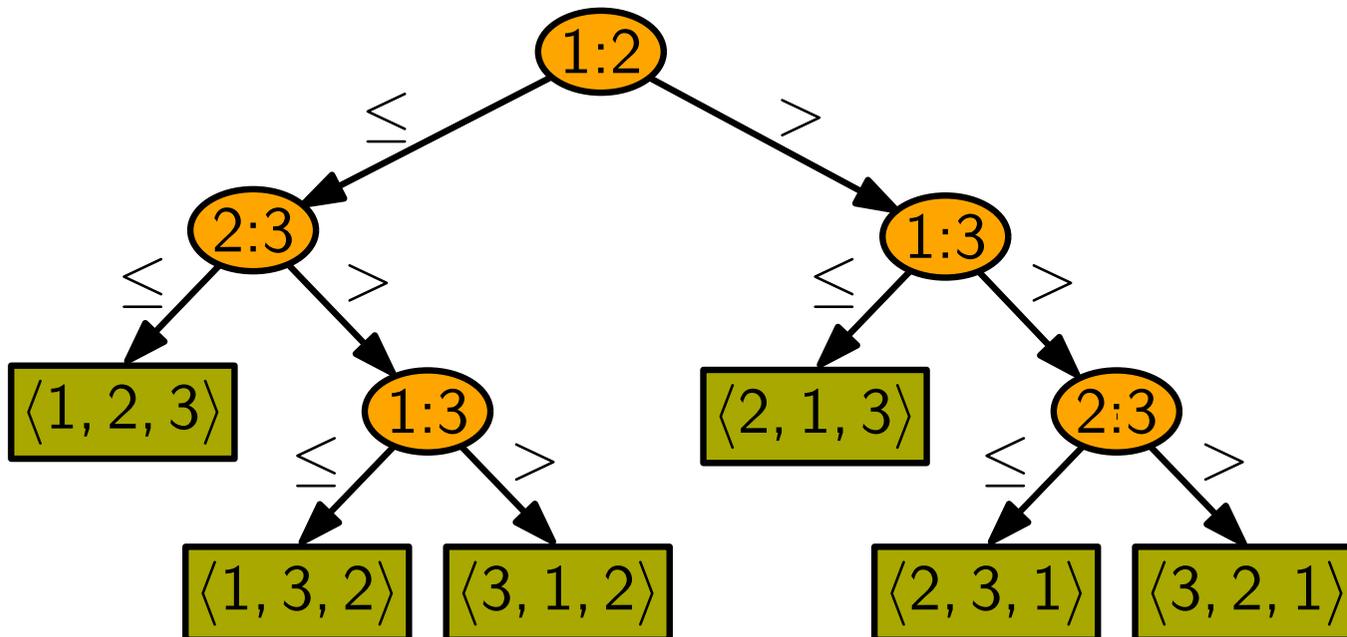
Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter Sortieralg.* 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 $n = 3$ [CLRS]

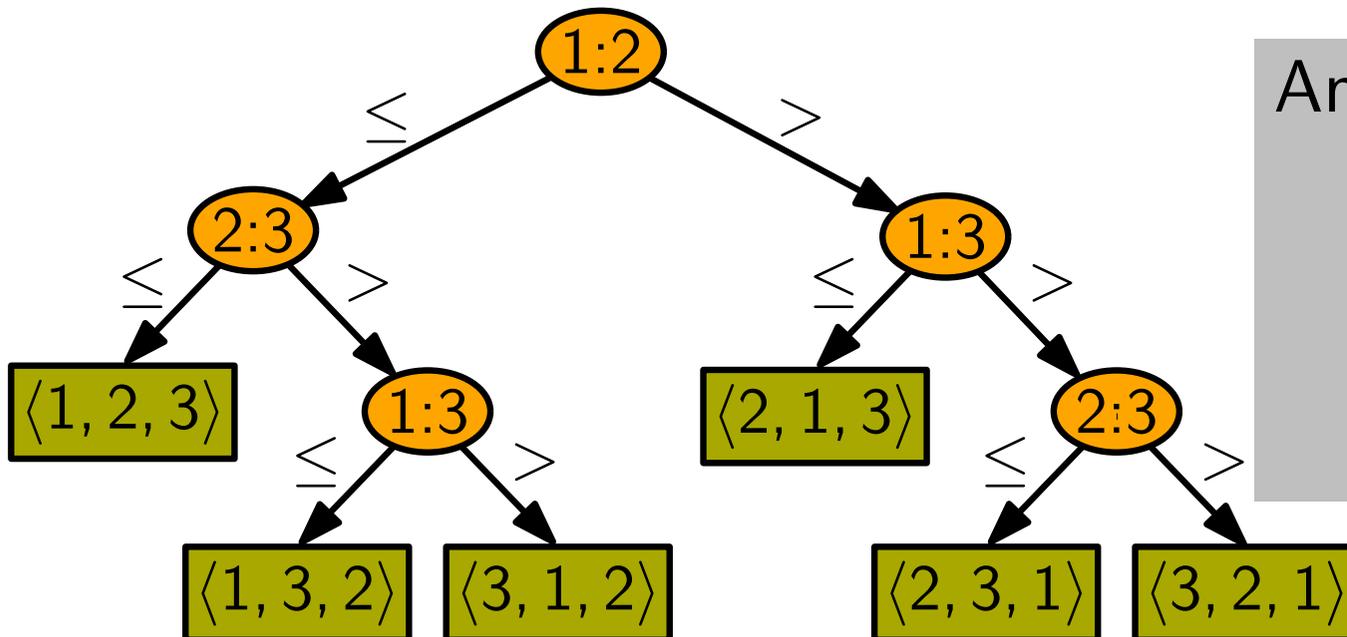
Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter Sortieralg.* 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



Anz. Vgl. im besten Fall

Entscheidungsbaum für InsertionSort und $n = 3$ [CLRS]

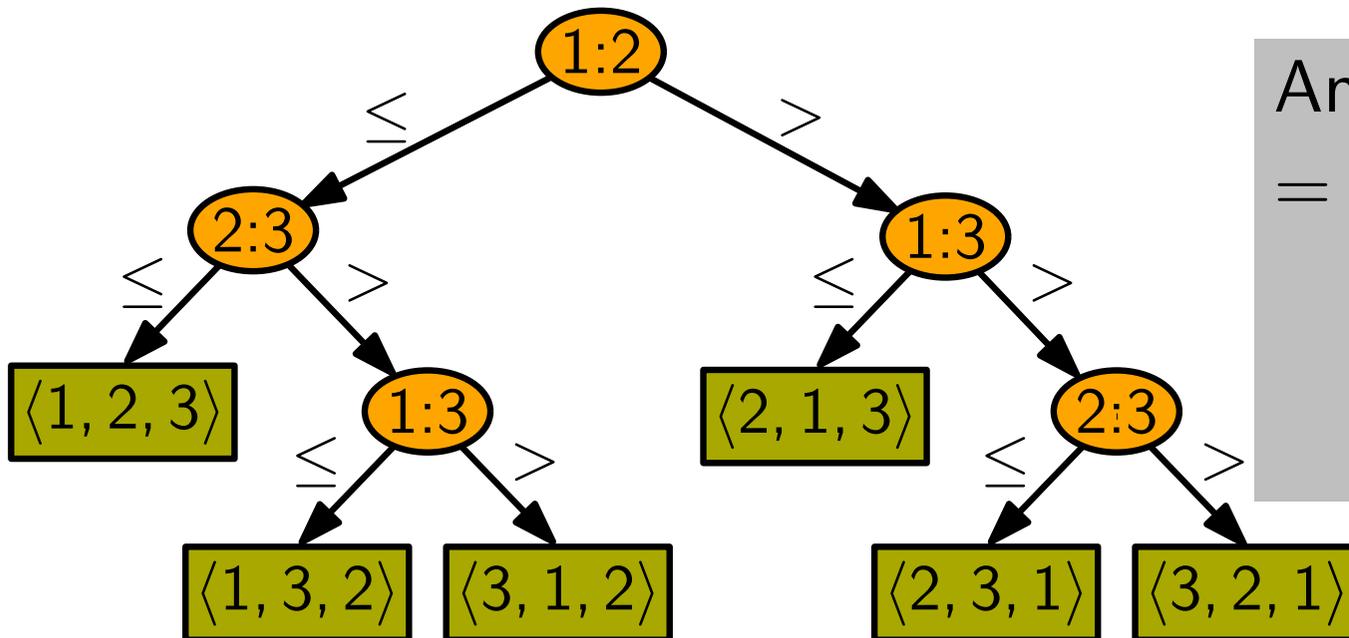
Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter Sortieralg.* 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



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

Entscheidungsbaum für InsertionSort und $n = 3$ [CLRS]

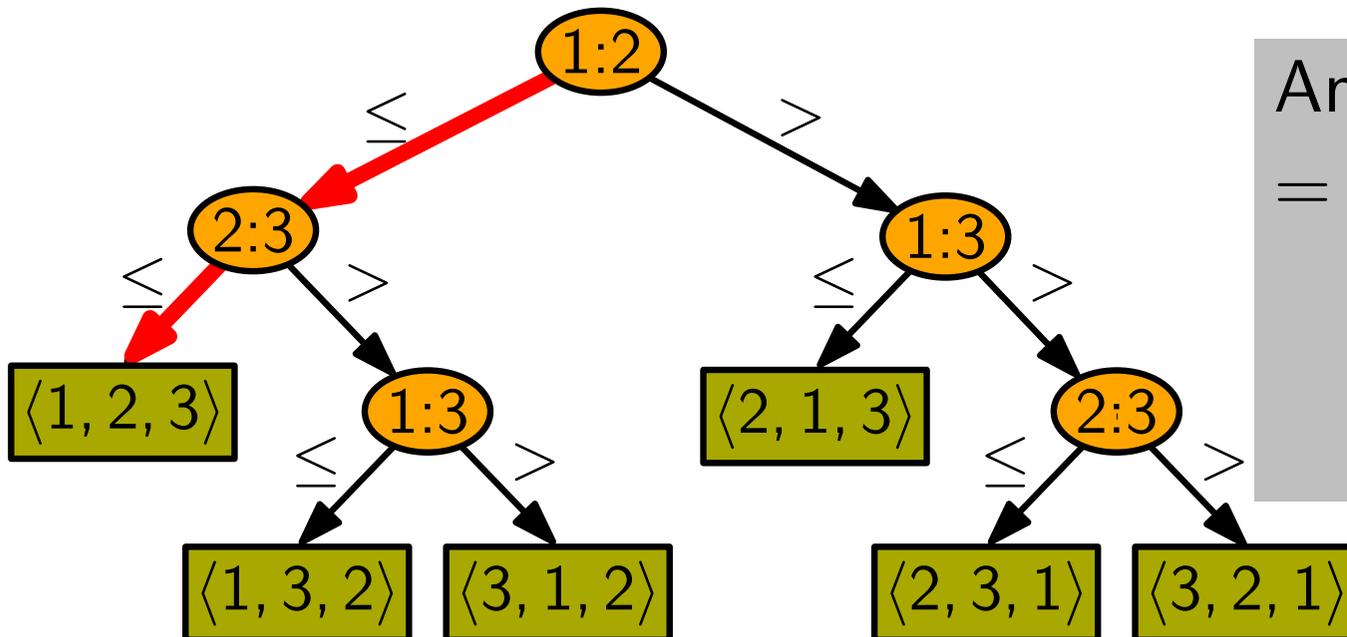
Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter Sortieralg.* 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



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

Entscheidungsbaum für InsertionSort und $n = 3$ [CLRS]

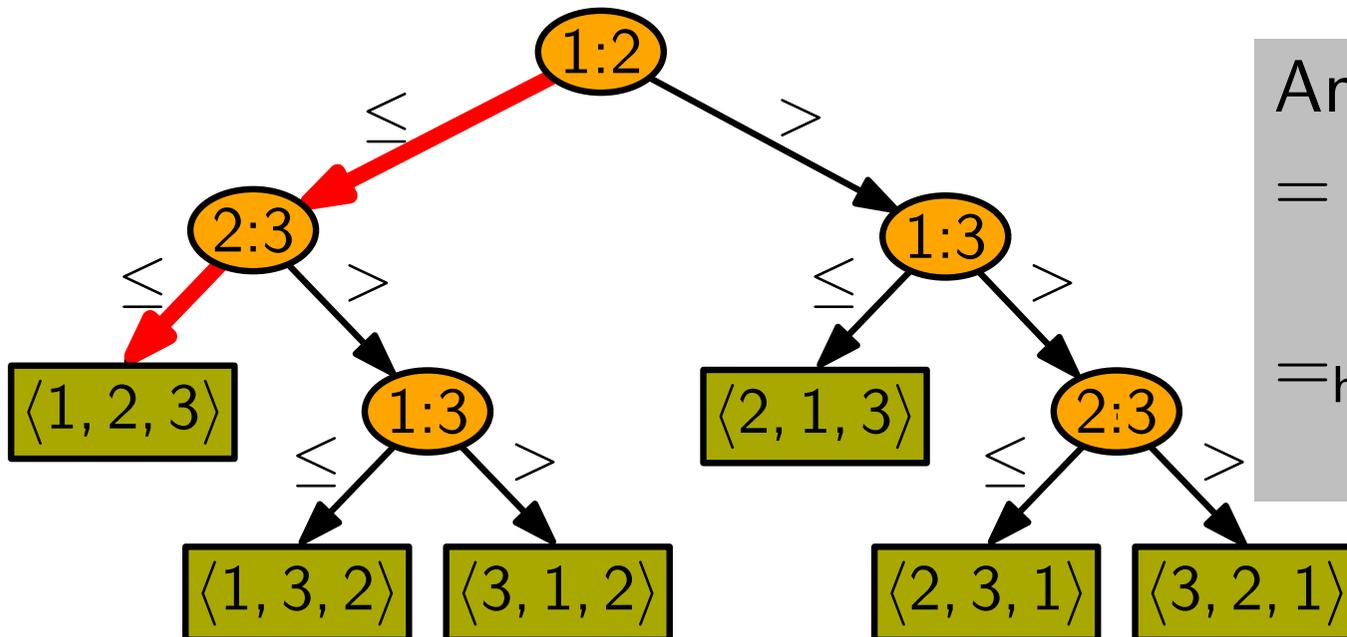
Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter Sortieralg.* 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



Anz. Vgl. im besten Fall
 = Länge eines kürzesten
 Wurzel-Blatt-Pfads
 = hier 2

Entscheidungsbaum für InsertionSort und $n = 3$ [CLRS]

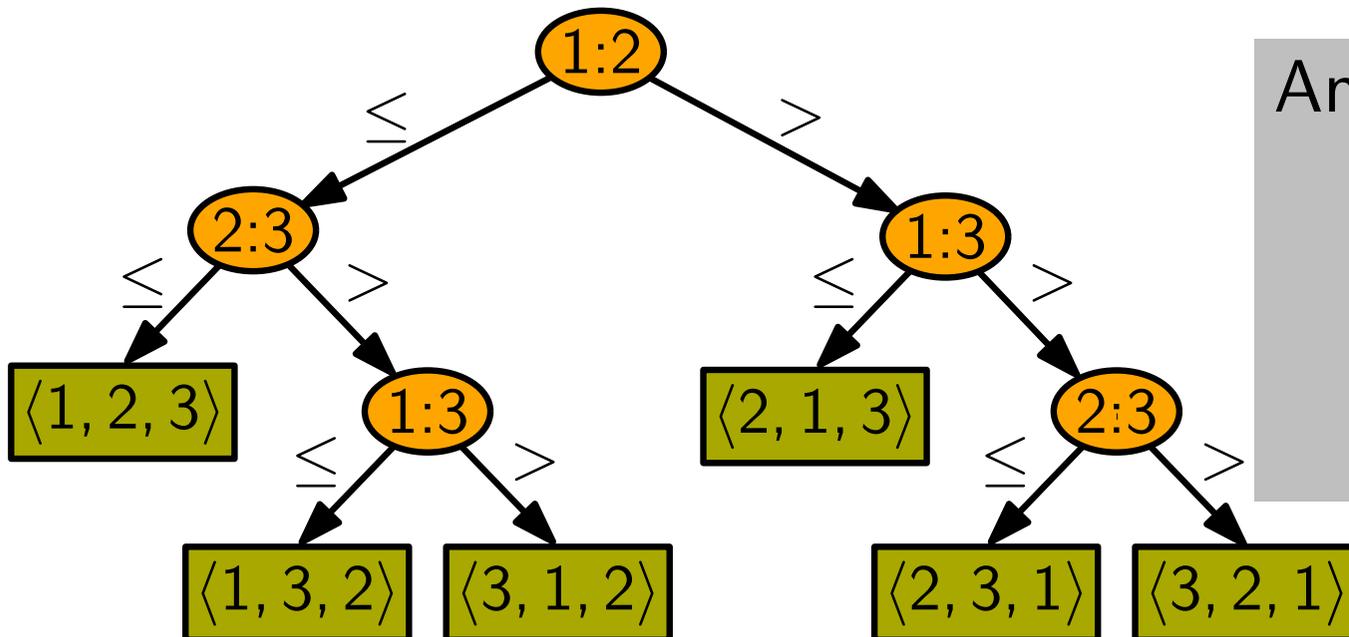
Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter Sortieralg.* 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



Anz. Vgl. im *worst case*

Entscheidungsbaum für InsertionSort und $n = 3$ [CLRS]

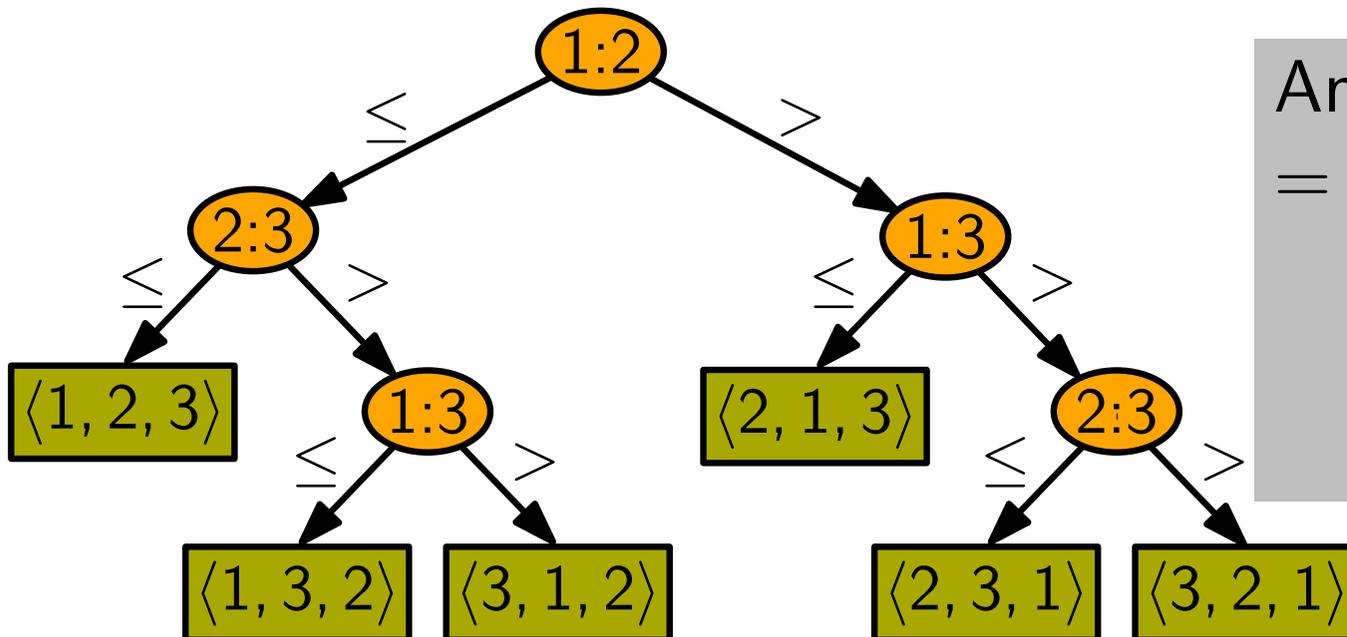
Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter Sortieralg.* 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



Anz. Vgl. im *worst case*
 = Länge eines *längsten*
 Wurzel-Blatt-Pfads

Entscheidungsbaum für InsertionSort und $n = 3$ [CLRS]

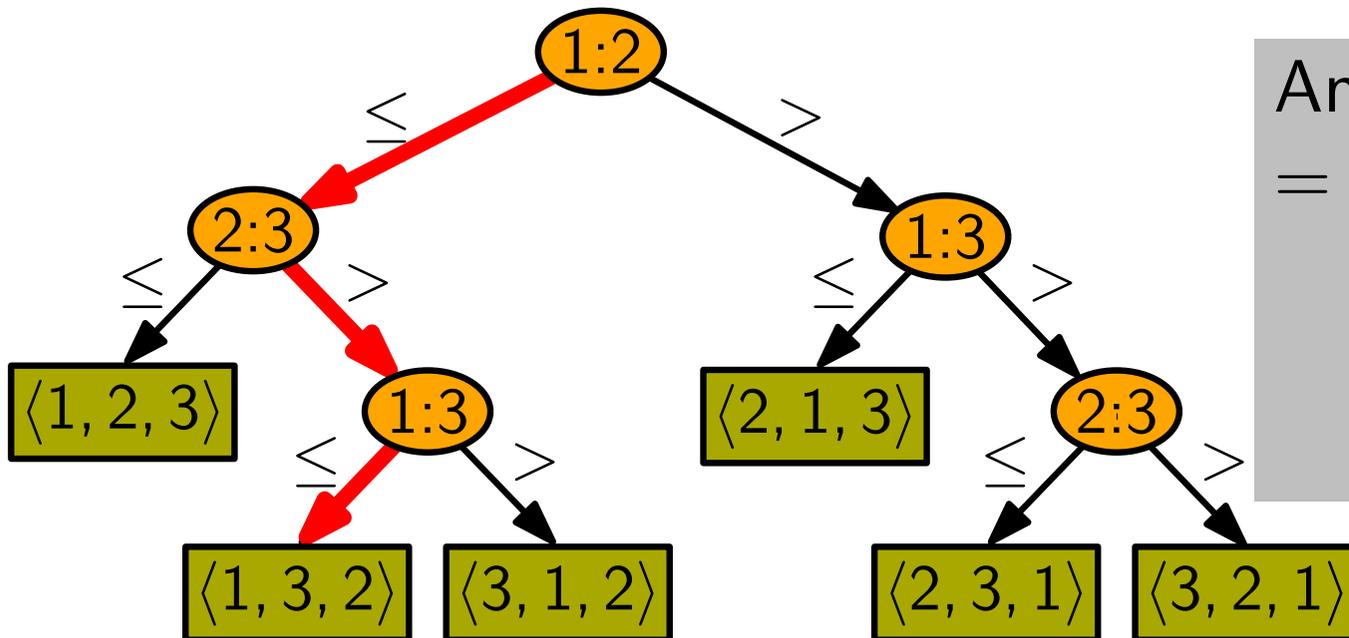
Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter Sortieralg.* 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



Anz. Vgl. im *worst case*
 = Länge eines *längsten*
 Wurzel-Blatt-Pfads

Entscheidungsbaum für InsertionSort und $n = 3$ [CLRS]

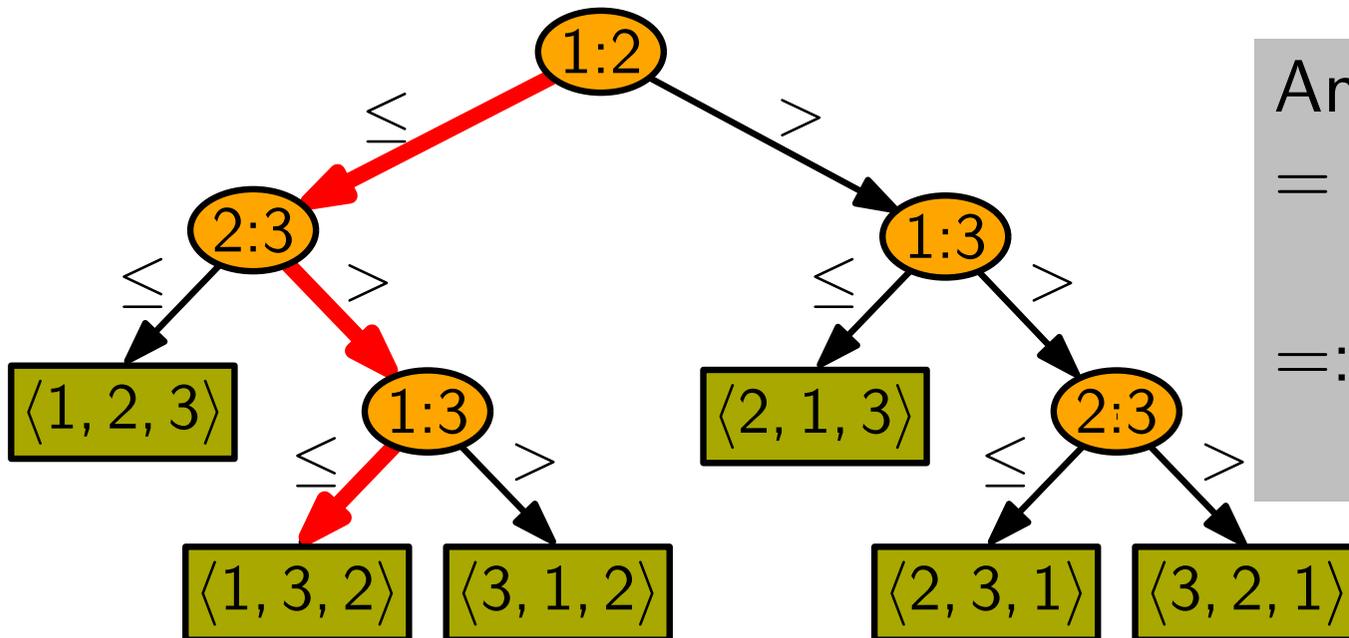
Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter Sortieralg.* 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



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

Entscheidungsbaum für InsertionSort und $n = 3$ [CLRS]

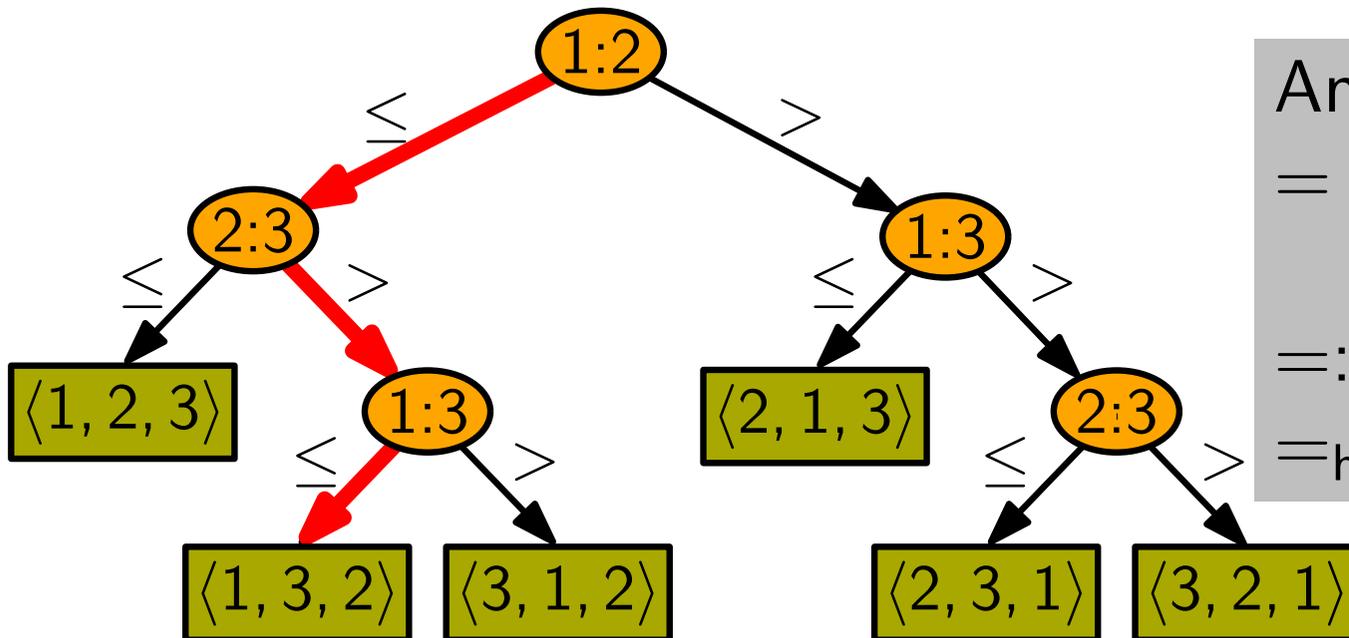
Sortieren durch Vergleichen

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

Für festes n ist ein *vergleichsbasierter Sortieralg.* 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



Anz. Vgl. im *worst case*
 = Länge eines *längsten*
 Wurzel-Blatt-Pfads
 =: Höhe des Baums
 = hier 3

Entscheidungsbaum für InsertionSort und $n = 3$ [CLRS]

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case um *n verschiedene* Objekte zu sortieren?

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 Sortieralg. im worst case 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 Sortieralg. im worst case 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 Sortieralg. im worst case 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. = $n!$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \lceil \log_2 B \rceil$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \lceil \log_2 B \rceil$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \log_2 n!$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \lceil \log_2 B \rceil$

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

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \lceil \log_2 B \rceil$

$$\begin{aligned} \text{Höhe Entscheidungsbaum} &\geq \log_2 n! = \sum_{i=1}^n \log_2 i \\ &\geq \int_1^n \log_2 x \, dx \end{aligned}$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \lceil \log_2 B \rceil$

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

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \lceil \log_2 B \rceil$

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

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \lceil \log_2 B \rceil$

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

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

partielle
Integration

=

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \lceil \log_2 B \rceil$

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

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

partielle
Integration

=

$$\int u' v = uv - \int uv'$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \lceil \log_2 B \rceil$

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

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

partielle
Integration

=

$$\int u'v = uv - \int uv'$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \lceil \log_2 B \rceil$

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

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

partielle
Integration

=

$$\int u'v = uv - \int uv'$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \lceil \log_2 B \rceil$

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

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

partielle
Integration

=

$$\int u'v = uv - \int uv'$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \lceil \log_2 B \rceil$

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

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

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

partielle
Integration

$$\int u'v = uv - \int uv'$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \lceil \log_2 B \rceil$

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

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

$$= \frac{1}{\ln 2} \left(\mathbf{x} \cdot \ln x \Big|_1^n - \int_1^n \mathbf{x} \cdot \mathbf{\quad} \, dx \right)$$

partielle
Integration

$$\int u'v = uv - \int uv'$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \lceil \log_2 B \rceil$

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

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

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

partielle
Integration

$$\int u'v = uv - \int uv'$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \lceil \log_2 B \rceil$

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

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

$$= \frac{1}{\ln 2} \left(\mathbf{x} \cdot \ln x \Big|_1^n - \int_1^n \mathbf{x} \cdot \frac{1}{x} \, dx \right) = \frac{(n \ln n - 0) - (n-1)}{\ln 2}$$

partielle
Integration

$$\int u' v = uv - \int uv'$$

Eine untere Schranke

Frage: Wie viele Vergleiche braucht *jeder* vergleichsbasierte Sortieralg. im worst case 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 $\geq \lceil \log_2 B \rceil$

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

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

$$= \frac{1}{\ln 2} \left(\mathbf{x} \cdot \ln x \Big|_1^n - \int_1^n \mathbf{x} \cdot \frac{1}{x} \, dx \right) = \frac{(n \ln n - 0) - (n-1)}{\ln 2}$$

$$\in \Omega(n \log n)$$

partielle
Integration

$$\int u' v = uv - \int uv'$$

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 Sortieralg.

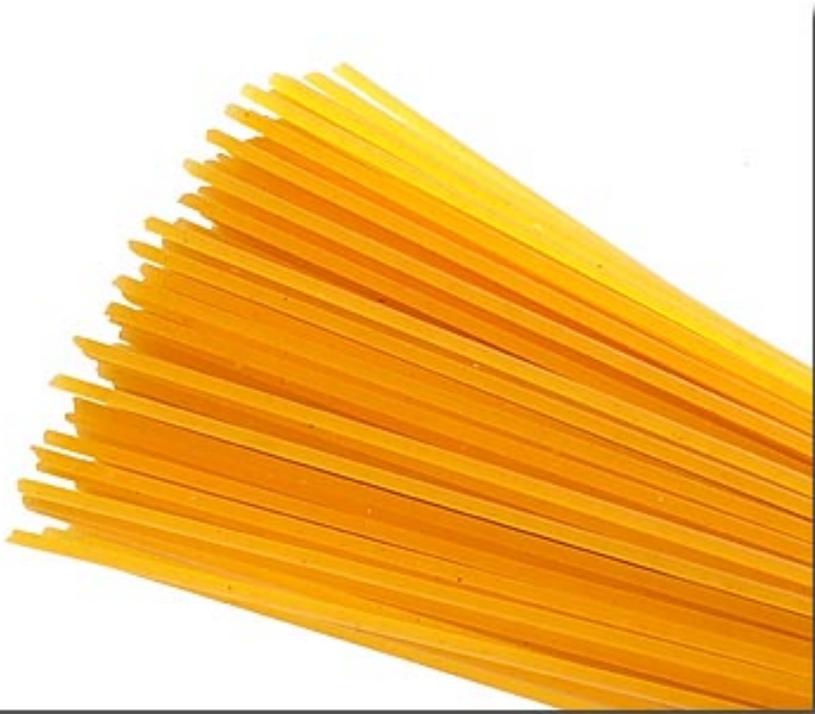
Wir durchbrechen die Schallmauer



By Eduard Marmet, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=5810282>

Wir durchbrechen die Schallmauer

- SpaghettiSort



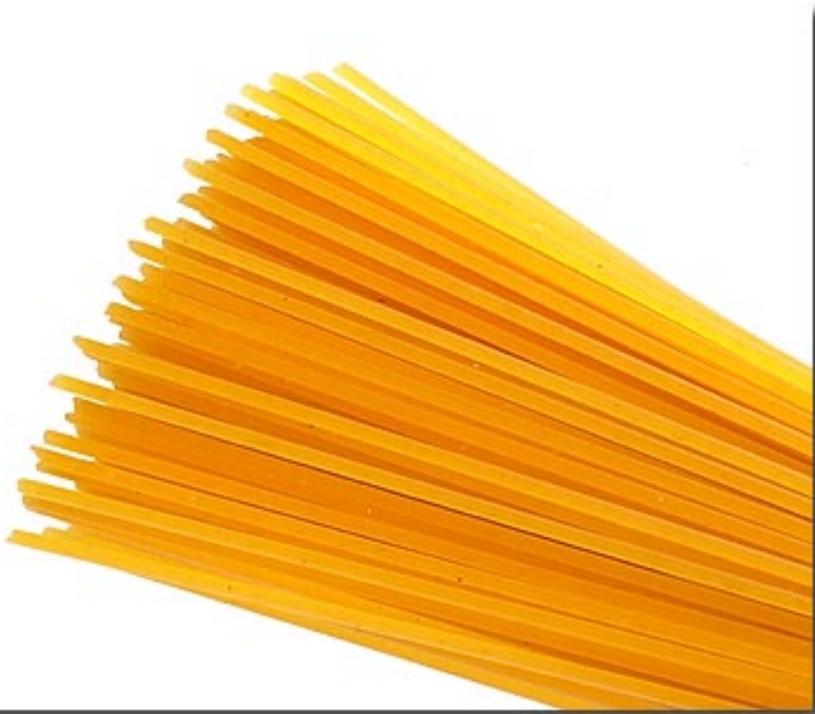
aus: www.marions-kochbuch.de



By Eduard Marmet, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=5810282>

Wir durchbrechen die Schallmauer

(● SpaghettiSort sortiert Spaghetti nach Länge ;-)



aus: www.marions-kochbuch.de



By Eduard Marmet, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=5810282>

Wir durchbrechen die Schallmauer

- (● SpaghettiSort sortiert Spaghetti nach Länge ;-)
- CountingSort



aus: www.marions-kochbuch.de



By Eduard Marmet, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=5810282>

Wir durchbrechen die Schallmauer

- SpaghettiSort sortiert Spaghetti nach Länge ;-)
- CountingSort sortiert Zahlen in $\{0, \dots, k\}$



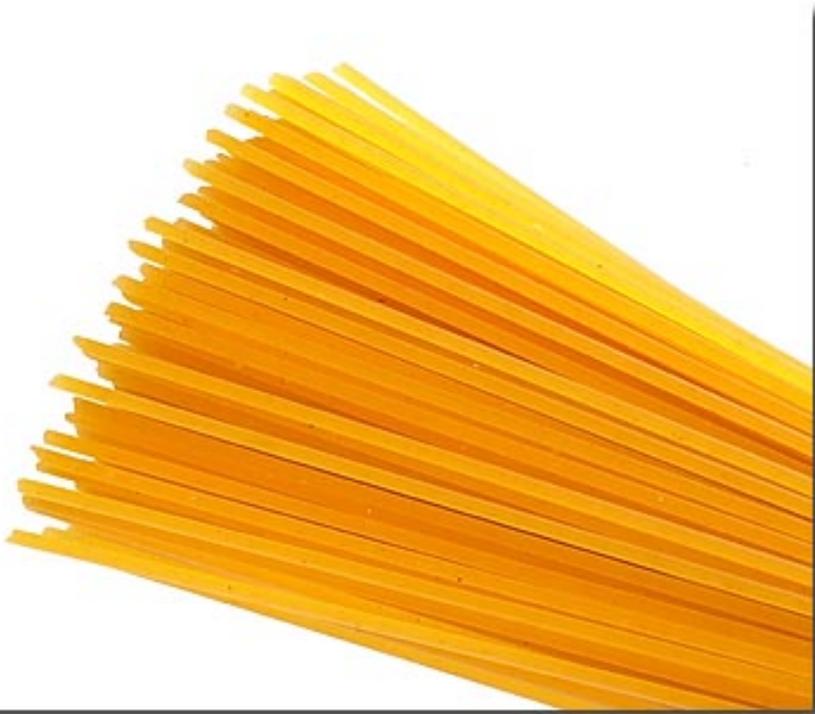
aus: www.marions-kochbuch.de



By Eduard Marmet, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=5810282>

Wir durchbrechen die Schallmauer

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



aus: www.marions-kochbuch.de



By Eduard Marmet, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=5810282>

Wir durchbrechen die Schallmauer

- SpaghettiSort sortiert Spaghetti nach Länge ;-)
- CountingSort sortiert Zahlen in $\{0, \dots, k\}$
- RadixSort sortiert s -stellige b -adische Zahlen



aus: www.marions-kochbuch.de



By Eduard Marmet, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=5810282>

Wir durchbrechen die Schallmauer

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



aus: www.marions-kochbuch.de



By Eduard Marmet, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=5810282>

Wir durchbrechen die Schallmauer

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



aus: www.marions-kochbuch.de



By Eduard Marmet, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=5810282>

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) benütze 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) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld		C	Rechenfeld
B	Ausgabefeld		k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld		C	Rechenfeld
B	Ausgabefeld		k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	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) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C					

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	0	0	0	0	0

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8	\Rightarrow	0	1	2	3	4	
A	3	0	4	1	3	4	1	4		C	0	0	0	0	0

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	0	0	0	0	0

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	0	0	0	1	0

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	0	0	0	1	0

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	0	0	0	1	0

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	0	0	0	1	0

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld		C	Rechenfeld
B	Ausgabefeld		k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	0	0	1	0

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	0	0	1	0

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	0	0	1	0

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	0	0	1	0

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	0	0	1	1

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	0	0	1	1

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	0	0	1	1

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8	\Rightarrow		0	1	2	3	4
A	3	0	4	1	3	4	1	4		C	1	0	0	1	1

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	1	0	1	1

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	1	0	1	1

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	1	0	1	1

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	1	0	2	1

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	1	0	2	1

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	1	0	2	1

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	1	0	2	2

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	1	0	2	2

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	1	0	2	2

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	1	0	2	2

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	2	0	2	2

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	2	0	2	2

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	2	0	2	2

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	2	0	2	3

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld		C	Rechenfeld
B	Ausgabefeld		k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

Bsp: 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	2	0	2	3

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C					
	0	1	2	3	4

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8	\Rightarrow		0	1	2	3	4
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

	0	1	2	3	4
C					



CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8	\Rightarrow		0	1	2	3	4
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

	0	1	2	3	4
C	1				
	0	1	2	3	4

↓

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8	\Rightarrow		0	1	2	3	4
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

	0	1	2	3	4
C	1				
	0	1	2	3	4

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

1	2	3	4	5	6	7	8	\Rightarrow	0	1	2	3	4		
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

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) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld		C	Rechenfeld
B	Ausgabefeld		k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

	0	1	2	3	4
C	1	3			
	0	1	2	3	4

Note: A red arrow points from the '0' in the second row of the previous table to the orange cell in the third row. A blue arrow points from the orange cell to the '2' in the third row of the current table.

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8	\Rightarrow		0	1	2	3	4
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

	0	1	2	3	4
C	1	3	3		
	0	1	2	3	4

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

1	2	3	4	5	6	7	8	\Rightarrow	0	1	2	3	4		
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	3	3		
	0	1	2	3	4

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8	\Rightarrow		0	1	2	3	4
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

	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) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

1	2	3	4	5	6	7	8	\Rightarrow	0	1	2	3	4		
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	3	3	5	
	0	1	2	3	4

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8	\Rightarrow		0	1	2	3	4
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

	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) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8	\Rightarrow		0	1	2	3	4
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	3	3	5	8
	0	1	2	3	4

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	3	3	5	8
	0	1	2	3	4

- 2) 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) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8	\Rightarrow		0	1	2	3	4
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	3	3	5	8
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B								
	1	2	3	4	5	6	7	8

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	3	3	5	8
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B								
	1	2	3	4	5	6	7	8

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	3	3	5	8
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B								
	1	2	3	4	5	6	7	8

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8	\Rightarrow		0	1	2	3	4
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	3	3	5	8
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B								
	1	2	3	4	5	6	7	8

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	3	3	5	8
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B								
	1	2	3	4	5	6	7	8

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	3	3	5	8
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B								
	1	2	3	4	5	6	7	8

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	3	3	5	8
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B								4
	1	2	3	4	5	6	7	8

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

A	<table style="border-collapse: collapse;"> <tr> <td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td><td style="padding: 0 5px;">5</td><td style="padding: 0 5px;">6</td><td style="padding: 0 5px;">7</td><td style="padding: 0 5px;">8</td> </tr> <tr> <td style="border: 1px solid green; padding: 5px;">3</td><td style="border: 1px solid green; padding: 5px;">0</td><td style="border: 1px solid green; padding: 5px;">4</td><td style="border: 1px solid green; padding: 5px;">1</td><td style="border: 1px solid green; padding: 5px;">3</td><td style="border: 1px solid green; padding: 5px;">4</td><td style="border: 1px solid green; padding: 5px;">1</td><td style="border: 1px solid green; padding: 5px; background-color: #FFD700;">4</td> </tr> </table>	1	2	3	4	5	6	7	8	3	0	4	1	3	4	1	4	\Rightarrow	C	<table style="border-collapse: collapse;"> <tr> <td style="padding: 0 5px;">0</td><td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td> </tr> <tr> <td style="border: 1px solid green; padding: 5px;">1</td><td style="border: 1px solid green; padding: 5px;">2</td><td style="border: 1px solid green; padding: 5px;">0</td><td style="border: 1px solid green; padding: 5px;">2</td><td style="border: 1px solid green; padding: 5px;">3</td> </tr> </table>	0	1	2	3	4	1	2	0	2	3
1	2	3	4	5	6	7	8																							
3	0	4	1	3	4	1	4																							
0	1	2	3	4																										
1	2	0	2	3																										

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	<table style="border-collapse: collapse;"> <tr> <td style="padding: 0 5px;">0</td><td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td> </tr> <tr> <td style="border: 1px solid green; padding: 5px;">1</td><td style="border: 1px solid green; padding: 5px;">3</td><td style="border: 1px solid green; padding: 5px;">3</td><td style="border: 1px solid green; padding: 5px;">5</td><td style="border: 1px solid green; padding: 5px; background-color: #ADD8E6;">8</td> </tr> <tr> <td style="padding: 0 5px;">0</td><td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px; background-color: #FFD700;">4</td> </tr> </table>	0	1	2	3	4	1	3	3	5	8	0	1	2	3	4
0	1	2	3	4												
1	3	3	5	8												
0	1	2	3	4												

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B	<table style="border-collapse: collapse;"> <tr> <td style="border: 1px solid green; width: 30px; height: 30px;"></td><td style="border: 1px solid green; width: 30px; height: 30px; background-color: #FFD700;">4</td> </tr> <tr> <td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td><td style="padding: 0 5px;">5</td><td style="padding: 0 5px;">6</td><td style="padding: 0 5px;">7</td><td style="padding: 0 5px; background-color: #ADD8E6;">8</td> </tr> </table>								4	1	2	3	4	5	6	7	8
							4										
1	2	3	4	5	6	7	8										

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

A	1	2	3	4	5	6	7	8	\Rightarrow	C	0	1	2	3	4
	3	0	4	1	3	4	1	4		1	2	0	2	3	

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	3	3	5	8
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B								4
	1	2	3	4	5	6	7	8

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	3	3	5	7
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B								4
	1	2	3	4	5	6	7	8

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

1	2	3	4	5	6	7	8	\Rightarrow	0	1	2	3	4		
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	3	3	5	7
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B								4
	1	2	3	4	5	6	7	8

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	3	3	5	7
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B								4
	1	2	3	4	5	6	7	8

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8	\Rightarrow		0	1	2	3	4
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

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

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B								4
	1	2	3	4	5	6	7	8

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld		C	Rechenfeld
B	Ausgabefeld		k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x



- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$



- 2) 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) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x



- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$



- 2) 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) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld		C	Rechenfeld
B	Ausgabefeld		k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x



- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$



- 2) 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) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld		C	Rechenfeld
B	Ausgabefeld		k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	2	3	5	7
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B			1					4
	1	2	3	4	5	6	7	8

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8	\Rightarrow		0	1	2	3	4
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	2	3	5	7
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B			1					4
	1	2	3	4	5	6	7	8

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

A	1	2	3	4	5	6	7	8	\Rightarrow	C	0	1	2	3	4
	3	0	4	1	3	4	1	4			1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	2	3	5	7
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B			1					4
	1	2	3	4	5	6	7	8

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

A	1	2	3	4	5	6	7	8	\Rightarrow	C	0	1	2	3	4
	3	0	4	1	3	4	1	4			1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	2	3	5	7
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B			1					4
	1	2	3	4	5	6	7	8

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

A	<table style="border-collapse: collapse;"> <tr> <td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td><td style="padding: 0 5px;">5</td><td style="padding: 0 5px;">6</td><td style="padding: 0 5px;">7</td><td style="padding: 0 5px;">8</td> </tr> <tr> <td style="border: 1px solid green; padding: 5px;">3</td><td style="border: 1px solid green; padding: 5px;">0</td><td style="border: 1px solid green; padding: 5px;">4</td><td style="border: 1px solid green; padding: 5px;">1</td><td style="border: 1px solid green; padding: 5px;">3</td><td style="background-color: #FFD700; border: 1px solid green; padding: 5px;">4</td><td style="border: 1px solid green; padding: 5px;">1</td><td style="border: 1px solid green; padding: 5px;">4</td> </tr> </table>	1	2	3	4	5	6	7	8	3	0	4	1	3	4	1	4	\Rightarrow	<table style="border-collapse: collapse;"> <tr> <td style="padding: 0 5px;">0</td><td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td> </tr> <tr> <td style="border: 1px solid green; padding: 5px;">1</td><td style="border: 1px solid green; padding: 5px;">2</td><td style="border: 1px solid green; padding: 5px;">0</td><td style="border: 1px solid green; padding: 5px;">2</td><td style="border: 1px solid green; padding: 5px;">3</td> </tr> </table>	0	1	2	3	4	1	2	0	2	3
1	2	3	4	5	6	7	8																						
3	0	4	1	3	4	1	4																						
0	1	2	3	4																									
1	2	0	2	3																									

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	<table style="border-collapse: collapse;"> <tr> <td style="border: 1px solid green; padding: 5px;">1</td><td style="border: 1px solid green; padding: 5px;">2</td><td style="border: 1px solid green; padding: 5px;">3</td><td style="border: 1px solid green; padding: 5px;">5</td><td style="border: 1px solid green; padding: 5px;">7</td> </tr> <tr> <td style="padding: 0 5px;">0</td><td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td> </tr> </table>	1	2	3	5	7	0	1	2	3	4
1	2	3	5	7							
0	1	2	3	4							

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B	<table style="border-collapse: collapse;"> <tr> <td style="border: 1px solid green; padding: 5px;"> </td><td style="border: 1px solid green; padding: 5px;"> </td><td style="border: 1px solid green; padding: 5px;">1</td><td style="border: 1px solid green; padding: 5px;"> </td><td style="border: 1px solid green; padding: 5px;"> </td><td style="border: 1px solid green; padding: 5px;"> </td><td style="border: 1px solid green; padding: 5px;">4</td><td style="border: 1px solid green; padding: 5px;">4</td> </tr> <tr> <td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td><td style="padding: 0 5px;">5</td><td style="padding: 0 5px;">6</td><td style="padding: 0 5px;">7</td><td style="padding: 0 5px;">8</td> </tr> </table>			1				4	4	1	2	3	4	5	6	7	8
		1				4	4										
1	2	3	4	5	6	7	8										

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

A	<table style="border-collapse: collapse;"> <tr> <td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td><td style="padding: 0 5px;">5</td><td style="padding: 0 5px;">6</td><td style="padding: 0 5px;">7</td><td style="padding: 0 5px;">8</td> </tr> <tr> <td style="border: 1px solid green; padding: 2px 10px;">3</td><td style="border: 1px solid green; padding: 2px 10px;">0</td><td style="border: 1px solid green; padding: 2px 10px;">4</td><td style="border: 1px solid green; padding: 2px 10px;">1</td><td style="border: 1px solid green; padding: 2px 10px;">3</td><td style="background-color: #FFD700; border: 1px solid green; padding: 2px 10px;">4</td><td style="border: 1px solid green; padding: 2px 10px;">1</td><td style="border: 1px solid green; padding: 2px 10px;">4</td> </tr> </table>	1	2	3	4	5	6	7	8	3	0	4	1	3	4	1	4	\Rightarrow	C	<table style="border-collapse: collapse;"> <tr> <td style="padding: 0 5px;">0</td><td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td> </tr> <tr> <td style="border: 1px solid green; padding: 2px 10px;">1</td><td style="border: 1px solid green; padding: 2px 10px;">2</td><td style="border: 1px solid green; padding: 2px 10px;">0</td><td style="border: 1px solid green; padding: 2px 10px;">2</td><td style="border: 1px solid green; padding: 2px 10px;">3</td> </tr> </table>	0	1	2	3	4	1	2	0	2	3
1	2	3	4	5	6	7	8																							
3	0	4	1	3	4	1	4																							
0	1	2	3	4																										
1	2	0	2	3																										

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	<table style="border-collapse: collapse;"> <tr> <td style="padding: 0 5px;">0</td><td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td> </tr> <tr> <td style="border: 1px solid green; padding: 2px 10px;">1</td><td style="border: 1px solid green; padding: 2px 10px;">2</td><td style="border: 1px solid green; padding: 2px 10px;">3</td><td style="border: 1px solid green; padding: 2px 10px;">5</td><td style="border: 1px solid green; padding: 2px 10px; border-left: 2px solid red;">7</td> </tr> </table>	0	1	2	3	4	1	2	3	5	7
0	1	2	3	4							
1	2	3	5	7							

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B	<table style="border-collapse: collapse;"> <tr> <td style="border: 1px solid green; padding: 2px 10px;"> </td><td style="border: 1px solid green; padding: 2px 10px;"> </td><td style="border: 1px solid green; padding: 2px 10px;">1</td><td style="border: 1px solid green; padding: 2px 10px;"> </td><td style="border: 1px solid green; padding: 2px 10px;"> </td><td style="border: 1px solid green; padding: 2px 10px;"> </td><td style="border: 1px solid green; padding: 2px 10px;">4</td><td style="border: 1px solid green; padding: 2px 10px;">4</td> </tr> <tr> <td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td><td style="padding: 0 5px;">5</td><td style="padding: 0 5px;">6</td><td style="padding: 0 5px;">7</td><td style="padding: 0 5px;">8</td> </tr> </table>			1				4	4	1	2	3	4	5	6	7	8
		1				4	4										
1	2	3	4	5	6	7	8										

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

A	1	2	3	4	5	6	7	8	\Rightarrow	C	0	1	2	3	4
	3	0	4	1	3	4	1	4			1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	2	3	5	6
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8	\Rightarrow		0	1	2	3	4
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	2	3	5	6
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

A	<table style="border-collapse: collapse;"> <tr> <td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td><td style="padding: 0 5px;">5</td><td style="padding: 0 5px;">6</td><td style="padding: 0 5px;">7</td><td style="padding: 0 5px;">8</td> </tr> <tr> <td style="border: 1px solid green; padding: 2px 10px;">3</td><td style="border: 1px solid green; padding: 2px 10px;">0</td><td style="border: 1px solid green; padding: 2px 10px;">4</td><td style="border: 1px solid green; padding: 2px 10px;">1</td><td style="background-color: orange; border: 1px solid green; padding: 2px 10px;">3</td><td style="border: 1px solid green; padding: 2px 10px;">4</td><td style="border: 1px solid green; padding: 2px 10px;">1</td><td style="border: 1px solid green; padding: 2px 10px;">4</td> </tr> </table>	1	2	3	4	5	6	7	8	3	0	4	1	3	4	1	4	\Rightarrow	C	<table style="border-collapse: collapse;"> <tr> <td style="padding: 0 5px;">0</td><td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td> </tr> <tr> <td style="border: 1px solid green; padding: 2px 10px;">1</td><td style="border: 1px solid green; padding: 2px 10px;">2</td><td style="border: 1px solid green; padding: 2px 10px;">0</td><td style="border: 1px solid green; padding: 2px 10px;">2</td><td style="border: 1px solid green; padding: 2px 10px;">3</td> </tr> </table>	0	1	2	3	4	1	2	0	2	3
1	2	3	4	5	6	7	8																							
3	0	4	1	3	4	1	4																							
0	1	2	3	4																										
1	2	0	2	3																										

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	<table style="border-collapse: collapse;"> <tr> <td style="padding: 0 5px;">0</td><td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td> </tr> <tr> <td style="border: 1px solid green; padding: 2px 10px;">1</td><td style="border: 1px solid green; padding: 2px 10px;">2</td><td style="border: 1px solid green; padding: 2px 10px;">3</td><td style="border: 1px solid green; padding: 2px 10px;">5</td><td style="border: 1px solid green; padding: 2px 10px;">6</td> </tr> </table>	0	1	2	3	4	1	2	3	5	6
0	1	2	3	4							
1	2	3	5	6							

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B	<table style="border-collapse: collapse;"> <tr> <td style="border: 1px solid green; padding: 2px 10px;"></td><td style="border: 1px solid green; padding: 2px 10px;"></td><td style="border: 1px solid green; padding: 2px 10px;">1</td><td style="border: 1px solid green; padding: 2px 10px;"></td><td style="border: 1px solid green; padding: 2px 10px;"></td><td style="border: 1px solid green; padding: 2px 10px;"></td><td style="border: 1px solid green; padding: 2px 10px;">4</td><td style="border: 1px solid green; padding: 2px 10px;">4</td> </tr> <tr> <td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td><td style="padding: 0 5px;">5</td><td style="padding: 0 5px;">6</td><td style="padding: 0 5px;">7</td><td style="padding: 0 5px;">8</td> </tr> </table>			1				4	4	1	2	3	4	5	6	7	8
		1				4	4										
1	2	3	4	5	6	7	8										

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

A	1	2	3	4	5	6	7	8	\Rightarrow	C	0	1	2	3	4
	3	0	4	1	3	4	1	4			1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	2	3	5	6
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

A	1	2	3	4	5	6	7	8	\Rightarrow	C	0	1	2	3	4
	3	0	4	1	3	4	1	4			1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	2	3	5	6
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

A	1 2 3 4 5 6 7 8	⇒	0 1 2 3 4	C													
	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>3</td><td>0</td><td>4</td><td>1</td><td style="background-color: #ffcc00;">3</td><td>4</td><td>1</td><td>4</td></tr> </table>	3	0	4	1	3	4	1	4		<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>1</td><td>2</td><td>0</td><td>2</td><td>3</td></tr> </table>	1	2	0	2	3	
3	0	4	1	3	4	1	4										
1	2	0	2	3													

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>1</td><td>2</td><td>3</td><td>4</td><td>6</td></tr> </table>	1	2	3	4	6
1	2	3	4	6		
	0 1 2 3 4					

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td></td><td></td><td>1</td><td></td><td style="background-color: #ffcc00;">3</td><td></td><td>4</td><td>4</td></tr> </table>			1		3		4	4
		1		3		4	4		
	1 2 3 4 5 6 7 8								

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	2	3	4	6
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

0	1	2	3	4
1	2	3	4	6
0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	2	3	4	6
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld		C	Rechenfeld
B	Ausgabefeld		k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	2	3	4	6
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	1	3	4	6
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

A	1	2	3	4	5	6	7	8	\Rightarrow	C	0	1	2	3	4
	3	0	4	1	3	4	1	4			1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	1	3	4	6
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	1	3	4	6
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	1	3	4	6
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	1	3	4	5
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	1	3	4	5
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld		C	Rechenfeld
B	Ausgabefeld		k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	1	3	4	5
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	1	1	3	4	5
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

A	1 2 3 4 5 6 7 8	\Rightarrow	0 1 2 3 4	C													
	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>3</td><td>0</td><td>4</td><td>1</td><td>3</td><td>4</td><td>1</td><td>4</td></tr> </table>	3	0	4	1	3	4	1	4		<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>1</td><td>2</td><td>0</td><td>2</td><td>3</td></tr> </table>	1	2	0	2	3	
3	0	4	1	3	4	1	4										
1	2	0	2	3													

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>0</td><td>1</td><td>3</td><td>4</td><td>5</td></tr> </table>	0	1	3	4	5
0	1	3	4	5		
	<table style="border: none;"> <tr><td style="padding: 0 5px;">0</td><td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td></tr> </table>	0	1	2	3	4
0	1	2	3	4		

- 2) Schreibe jedes x in A direkt an die richtige Position in B

B	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>0</td><td>1</td><td>1</td><td></td><td>3</td><td>4</td><td>4</td><td>4</td></tr> </table>	0	1	1		3	4	4	4
0	1	1		3	4	4	4		
	<table style="border: none;"> <tr><td style="padding: 0 5px;">1</td><td style="padding: 0 5px;">2</td><td style="padding: 0 5px;">3</td><td style="padding: 0 5px;">4</td><td style="padding: 0 5px;">5</td><td style="padding: 0 5px;">6</td><td style="padding: 0 5px;">7</td><td style="padding: 0 5px;">8</td></tr> </table>	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8		

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A	Eingabefeld	C	Rechenfeld
B	Ausgabefeld	k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	0	1	3	4	5
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	0	1	3	4	5
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	0	1	3	4	5
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	0	1	3	4	5
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

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

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	0	1	3	3	5
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8	\Rightarrow		0	1	2	3	4
A	3	0	4	1	3	4	1	4		C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	0	1	3	3	5
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x

	1	2	3	4	5	6	7	8		0	1	2	3	4	
A	3	0	4	1	3	4	1	4	\Rightarrow	C	1	2	0	2	3

- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$

C	0	1	3	3	5
	0	1	2	3	4

- 2) Schreibe jedes x in A direkt an die richtige Position in B

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

CountingSort

- Idee:**
- 1) für jedes x in der Eingabe: zähle die Anzahl der Zahlen $\leq x$
 - 2) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

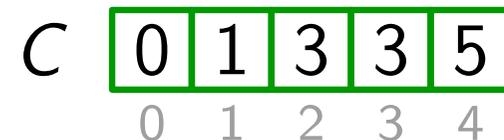
Variable:

A	Eingabefeld		C	Rechenfeld
B	Ausgabefeld		k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x



- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$



- 2) 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) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

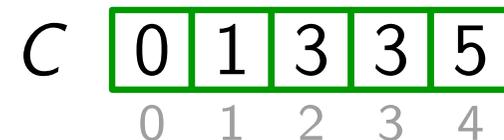
Variable:

A Eingabefeld	C Rechenfeld
B Ausgabefeld	k begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

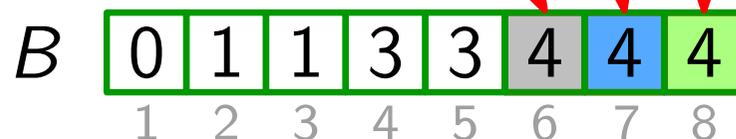
- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x



- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$



- 2) 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) benütze diese Information um x im Ausgabefeld direkt an die richtige Position zu schreiben

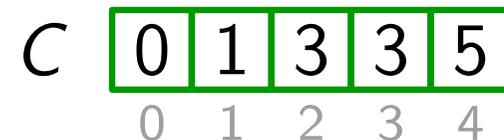
Variable:

A	Eingabefeld		C	Rechenfeld
B	Ausgabefeld		k	begrenzt das <i>Universum</i> : $\{0, \dots, k\}$

- Bsp:** 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x



- 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$



- 2) Schreibe jedes x in A direkt an die richtige Position in B



CountingSort ist *stabil!*

CountingSort

- Plan:**
- 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x
 - 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$
 - 2) Schreibe jedes x in A direkt an die richtige Position in B

CountingSort

- Plan:**
- 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x
 - 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$
 - 2) Schreibe jedes x in A direkt an die richtige Position in B

CountingSort(int[] A , int[] B , int k)

Eingabefeld
Ausgabefeld
beschränkt Universum $\{0, \dots, k\}$

CountingSort

- Plan:**
- 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x
 - 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$
 - 2) Schreibe jedes x in A direkt an die richtige Position in B

CountingSort(int[] *A*, int[] *B*, int *k*)

sei $C[0..k] = \langle 0, 0, \dots, 0 \rangle$ ein neues Feld

for $j = 1$ **to** $A.length$ **do** // (1a)

// $C[i]$ enthält jetzt die Anz. der Elem. gleich i in A

for $i = 1$ **to** k **do** // (1b)

// $C[i]$ enthält jetzt die Anz. der Elem. $\leq i$ in A

for $j = A.length$ **downto** 1 **do**

┌ // (2)

Aufgabe:

Fülle die Felder mit Code, der obige Idee umsetzt!

CountingSort

- Plan:**
- 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x
 - 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$
 - 2) Schreibe jedes x in A direkt an die richtige Position in B

CountingSort(int[] *A*, int[] *B*, int *k*)



sei $C[0..k] = \langle 0, 0, \dots, 0 \rangle$ ein neues Feld

for $j = 1$ **to** $A.length$ **do** $C[A[j]] = C[A[j]] + 1$ // (1a)

// $C[i]$ enthält jetzt die Anz. der Elem. gleich i in A

for $i = 1$ **to** k **do** $C[i] = C[i] + C[i - 1]$ // (1b)

// $C[i]$ enthält jetzt die Anz. der Elem. $\leq i$ in A

for $j = A.length$ **downto** 1 **do**

 // (2)

Aufgabe:

Fülle die Felder mit Code, der obige Idee umsetzt!

CountingSort

- Plan:**
- 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x
 - 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$
 - 2) Schreibe jedes x in A direkt an die richtige Position in B

CountingSort(int[] *A*, int[] *B*, int *k*)

sei $C[0..k] = \langle 0, 0, \dots, 0 \rangle$ ein neues Feld

for $j = 1$ **to** $A.length$ **do** $C[A[j]] = C[A[j]] + 1$ // (1a)

// $C[i]$ enthält jetzt die Anz. der Elem. gleich i in A

for $i = 1$ **to** k **do** $C[i] = C[i] + C[i - 1]$ // (1b)

// $C[i]$ enthält jetzt die Anz. der Elem. $\leq i$ in A

for $j = A.length$ **downto** 1 **do**

$B[C[A[j]]] = A[j]$ // (2)

Aufgabe:

Fülle die Felder mit Code, der obige Idee umsetzt!

CountingSort

- Plan:**
- 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x
 - 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$
 - 2) Schreibe jedes x in A direkt an die richtige Position in B

CountingSort(int[] A , int[] B , int k)

Eingabefeld
Ausgabefeld
beschränkt Universum $\{0, \dots, k\}$

sei $C[0..k] = \langle 0, 0, \dots, 0 \rangle$ ein neues Feld

for $j = 1$ **to** $A.length$ **do** $C[A[j]] = C[A[j]] + 1$ // (1a)

// $C[i]$ enthält jetzt die Anz. der Elem. gleich i in A

for $i = 1$ **to** k **do** $C[i] = C[i] + C[i - 1]$ // (1b)

// $C[i]$ enthält jetzt die Anz. der Elem. $\leq i$ in A

for $j = A.length$ **downto** 1 **do**

$B[C[A[j]]] = A[j]$
 $C[A[j]] = C[A[j]] - 1$ // (2)

Aufgabe:

Fülle die Felder mit Code, der obige Idee umsetzt!

CountingSort

Laufzeit:
 $O(\quad)$

- Plan:**
- 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x
 - 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$
 - 2) Schreibe jedes x in A direkt an die richtige Position in B

CountingSort(int[] A , int[] B , int k)

Eingabefeld
Ausgabefeld
beschränkt Universum $\{0, \dots, k\}$

sei $C[0..k] = \langle 0, 0, \dots, 0 \rangle$ ein neues Feld

for $j = 1$ **to** $A.length$ **do** $C[A[j]] = C[A[j]] + 1$ // (1a)

// $C[i]$ enthält jetzt die Anz. der Elem. gleich i in A

for $i = 1$ **to** k **do** $C[i] = C[i] + C[i - 1]$ // (1b)

// $C[i]$ enthält jetzt die Anz. der Elem. $\leq i$ in A

for $j = A.length$ **downto** 1 **do**

$\left[\begin{array}{l} B[C[A[j]]] = A[j] \\ C[A[j]] = C[A[j]] - 1 \end{array} \right.$ // (2)

CountingSort

Laufzeit:
 $O(\blacksquare + \blacksquare)$

- Plan:**
- 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x
 - 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$
 - 2) Schreibe jedes x in A direkt an die richtige Position in B

CountingSort(int[] A , int[] B , int k)

sei $C[0..k] = \langle 0, 0, \dots, 0 \rangle$ ein neues Feld

for $j = 1$ **to** $A.length$ **do** $C[A[j]] = C[A[j]] + 1$ // (1a)

// $C[i]$ enthält jetzt die Anz. der Elem. gleich i in A

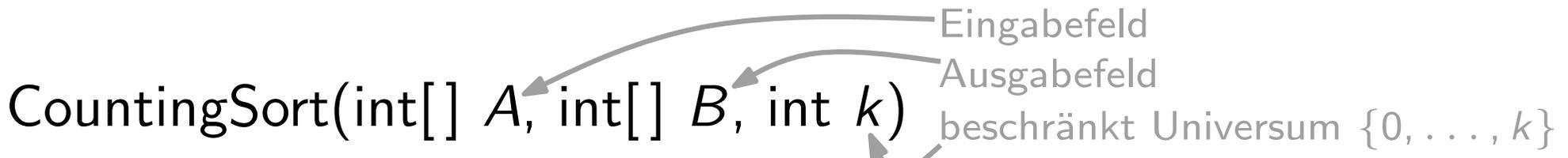
for $i = 1$ **to** k **do** $C[i] = C[i] + C[i - 1]$ // (1b)

// $C[i]$ enthält jetzt die Anz. der Elem. $\leq i$ in A

for $j = A.length$ **downto** 1 **do**

$B[C[A[j]]] = A[j]$

$C[A[j]] = C[A[j]] - 1$ // (2)



CountingSort

Laufzeit:
 $O(n + k)$

- Plan:**
- 1a) Für jedes x in A , zähle die Anz. der Zahlen gleich x
 - 1b) Für jedes x in A , berechne die Anz. der Zahlen $\leq x$
 - 2) Schreibe jedes x in A direkt an die richtige Position in B

CountingSort(int[] A , int[] B , int k)

Eingabefeld
Ausgabefeld
beschränkt Universum $\{0, \dots, k\}$

sei $C[0..k] = \langle 0, 0, \dots, 0 \rangle$ ein neues Feld

for $j = 1$ **to** $A.length$ **do** $C[A[j]] = C[A[j]] + 1$ // (1a)

// $C[i]$ enthält jetzt die Anz. der Elem. gleich i in A

for $i = 1$ **to** k **do** $C[i] = C[i] + C[i - 1]$ // (1b)

// $C[i]$ enthält jetzt die Anz. der Elem. $\leq i$ in A

for $j = A.length$ **downto** 1 **do**

$\left[\begin{array}{l} B[C[A[j]]] = A[j] \\ C[A[j]] = C[A[j]] - 1 \end{array} \right.$ // (2)

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: $O(n + k)$
- RadixSort sortiert s -stellige b -adische Zahlen
- BucketSort sortiert gleichverteilte zufällige Zahlen

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: $O(n + k)$
- RadixSort sortiert s -stellige b -adische Zahlen
- 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 Anz. 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 Anz. 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 Anz. 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 Anz. 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??

RadixSort

(Jahr, Monat, Tag)
↑

Frage: Gegeben Liste von Menschen mit deren Geburtstagen.
Wie würden Sie die Liste nach Alter sortieren?

Drei (?) Lösungen:

- Geburtstage in Anz. 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??

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 Anz. 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??

RadixSort(A, s)

Anz. Stellen (hier: 3)



RadixSort

(Jahr, Monat, Tag)



Frage: Gegeben Liste von Menschen mit deren Geburtstagen.
Wie würden Sie die Liste nach Alter sortieren?

Drei (?) Lösungen:

- Geburtstage in Anz. 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??

RadixSort(A, s) Anz. Stellen (hier: 3)

for $i = 1$ **to** s **do**

\perp 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 Anz. 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??

RadixSort(A, s) Anz. Stellen (hier: 3)

for $i = 1$ **to** s **do** [1 = Index der *niederwertigsten (!)* Stelle]
 └ 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 Anz. 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??

RadixSort(A, s) Anz. Stellen (hier: 3)

for $i = 1$ **to** s **do** [1 = Index der *niederwertigsten (!)* Stelle]
 └ sortiere A *stabil* nach der i -ten 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 Anz. 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??

RadixSort(A, s)

Anz. Stellen (hier: 3)

Laufzeit?

for $i = 1$ **to** s **do** [1 = Index der *niederwertigsten (!)* Stelle]

└ sortiere A *stabil* nach der i -ten Stelle

└ z.B. mit CountingSort

Beispiel

Sortiere $A = \langle 25, 13, 31, 23, 11, 37, 15 \rangle$:

Gemäß RadixSort erst nach Einern, dann (stabil) nach Zehnern.

RadixSort(A, s)

for $i = 1$ **to** s **do**

\perp 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: $O(n + k)$
- RadixSort sortiert s -stellige b -adische Zahlen
Laufzeit für n Zahlen: $O(s \cdot (n + b))$
- BucketSort sortiert gleichverteilte zufällige Zahlen

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: $O(n + k)$
- RadixSort sortiert s -stellige b -adische Zahlen
Laufzeit für n Zahlen: $O(s \cdot (n + b))$
- BucketSort sortiert gleichverteilte zufällige Zahlen

BucketSort



(c) www.seafish.org

BucketSort

[CLRS]

A	
1	.78
2	.17
3	.39
4	.21
5	.72
6	.94
7	.26
8	.12
9	.23
10	.68



(c) www.seafish.org

BucketSort

[CLRS]

A	
1	.78
2	.17
3	.39
4	.21
5	.72
6	.94
7	.26
8	.12
9	.23
10	.68



(c) www.seafish.org

↑
Eingabefeld $A[1..n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

BucketSort

[CLRS]

A	
1	.78
2	.17
3	.39
4	.21
5	.72
6	.94
7	.26
8	.12
9	.23
10	.68



(c) www.seafish.org

Eingabefeld $A[1..n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

[Im Bsp. auf 2 Nach-
kommastellen gerundet!]

BucketSort

[CLRS]

	A	B
1	.78	
2	.17	
3	.39	
4	.21	
5	.72	
6	.94	
7	.26	
8	.12	
9	.23	
10	.68	



(c) www.seafish.org

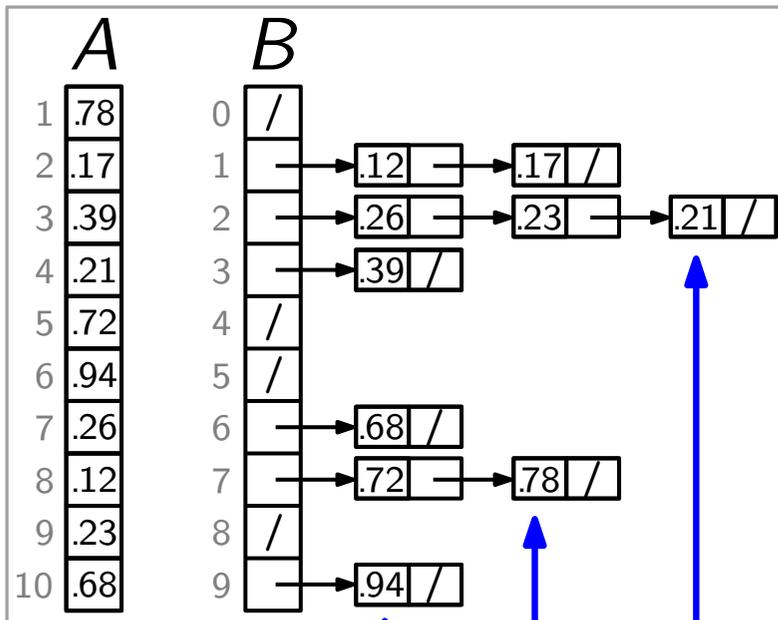
Hilfsfeld $B[0..n - 1]$;
jeder Eintrag entspricht einem „Eimer“ der Weite $1/n$

Eingabefeld $A[1..n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

[Im Bsp. auf 2 Nach-
kommastellen gerundet!]

BucketSort

[CLRS]



(c) www.seafish.org

„Eimerinhalt“: Verkettete Liste von Elementen aus A.

Hilfsfeld $B[0..n - 1]$;

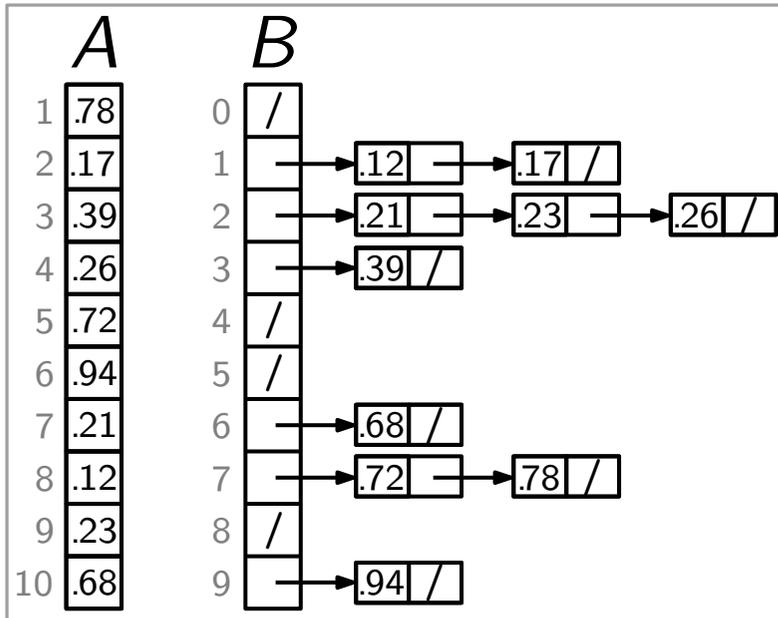
jeder Eintrag entspricht einem „Eimer“ der Weite $1/n$

Eingabefeld $A[1..n]$ enthält Zahlen,
zufällig und gleichverteilt aus $[0, 1)$ gezogen

[Im Bsp. auf 2 Nach-
kommastellen gerundet!]

BucketSort

[CLRS]



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n - 1]$ von Listen an

for $j = 1$ **to** n **do**

└ füge $A[j]$ in Liste $B[\text{]}$ ein

for $i = 0$ **to** $n - 1$ **do**

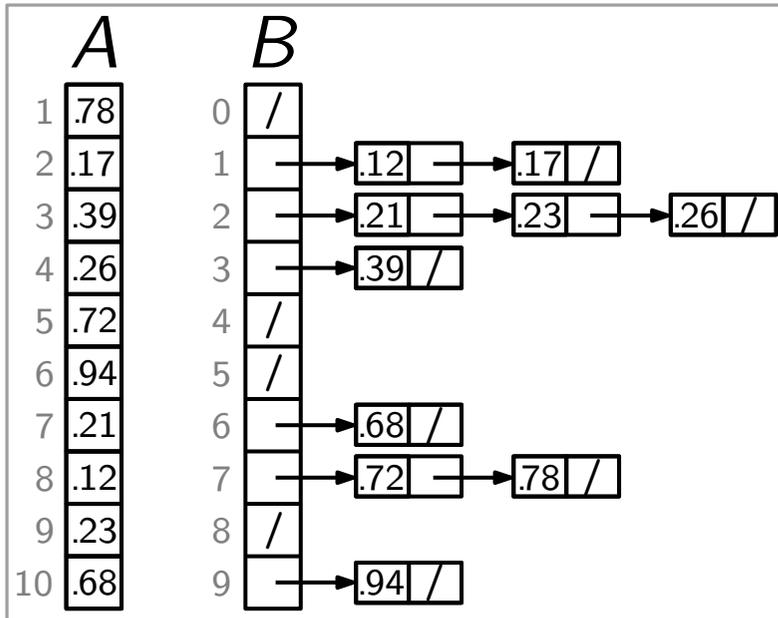
└ sortiere Liste $B[i]$

Aufgabe:

Fülle die Felder mit Code, der BucketSort umsetzt!

BucketSort

[CLRS]



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n - 1]$ 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**

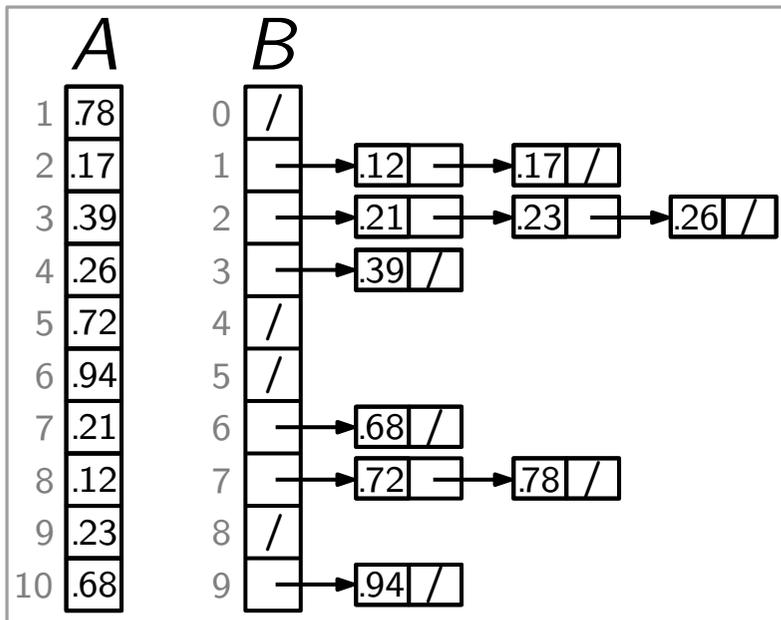
 sortiere Liste $B[i]$

Aufgabe:

Fülle die Felder mit Code, der BucketSort umsetzt!

BucketSort

[CLRS]



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n - 1]$ 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**

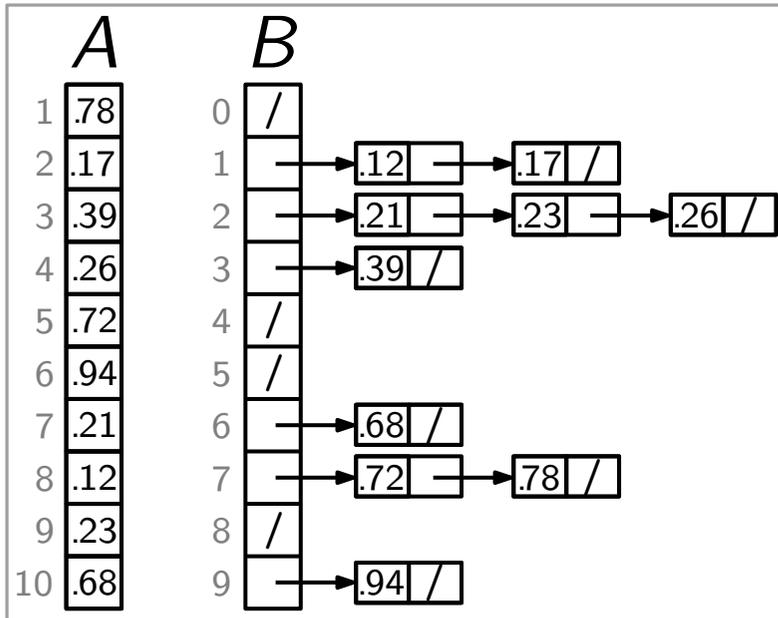
└ sortiere Liste $B[i]$

hänge $B[0], \dots, B[n - 1]$ aneinander

kopiere das Ergebnis nach $A[1..n]$

BucketSort

[CLRS]



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n - 1]$ 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**



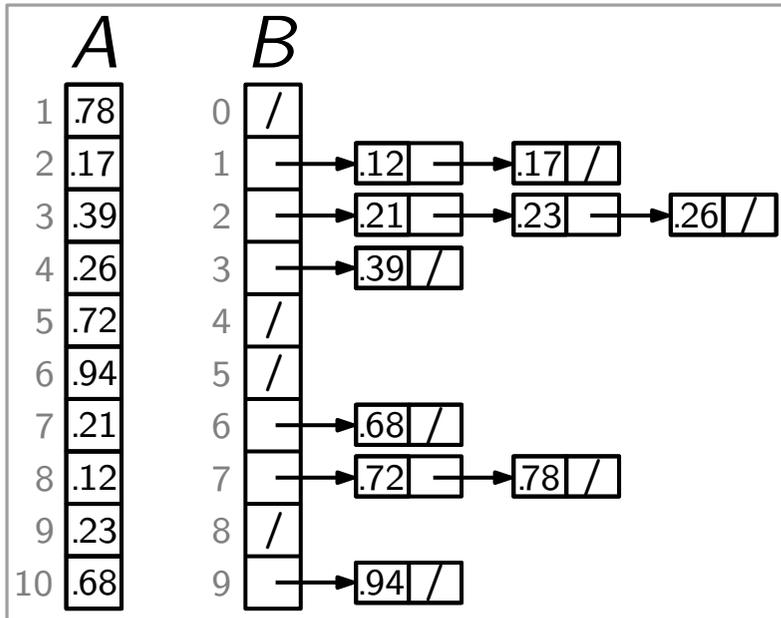
└ sortiere Liste $B[i]$

hänge $B[0], \dots, B[n - 1]$ aneinander

kopiere das Ergebnis nach $A[1..n]$

BucketSort

[CLRS]



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n - 1]$ 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**

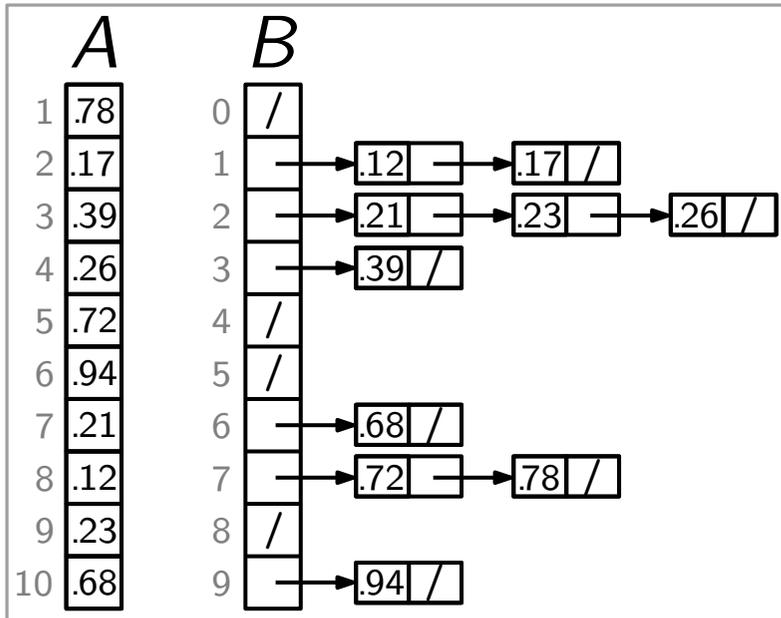
└ sortiere Liste $B[i] = \left[\frac{i}{n}, \frac{i+1}{n} \right) \cap A$

hänge $B[0], \dots, B[n - 1]$ aneinander

kopiere das Ergebnis nach $A[1..n]$

BucketSort

[CLRS]



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n - 1]$ 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 $B[i] = \left[\frac{i}{n}, \frac{i+1}{n} \right) \cap A$

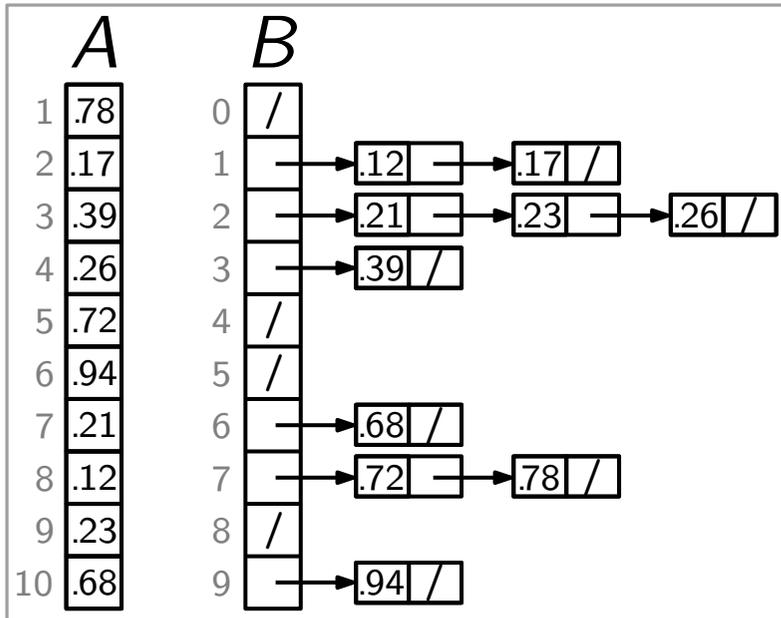
hänge $B[0], \dots, B[n - 1]$ aneinander

kopiere das Ergebnis nach $A[1..n]$

Korrektheit? 2 Fälle:

BucketSort

[CLRS]



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n - 1]$ 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 $B[i] = \left[\frac{i}{n}, \frac{i+1}{n} \right) \cap A$

hänge $B[0], \dots, B[n - 1]$ aneinander

kopiere das Ergebnis nach $A[1..n]$

Korrektheit?

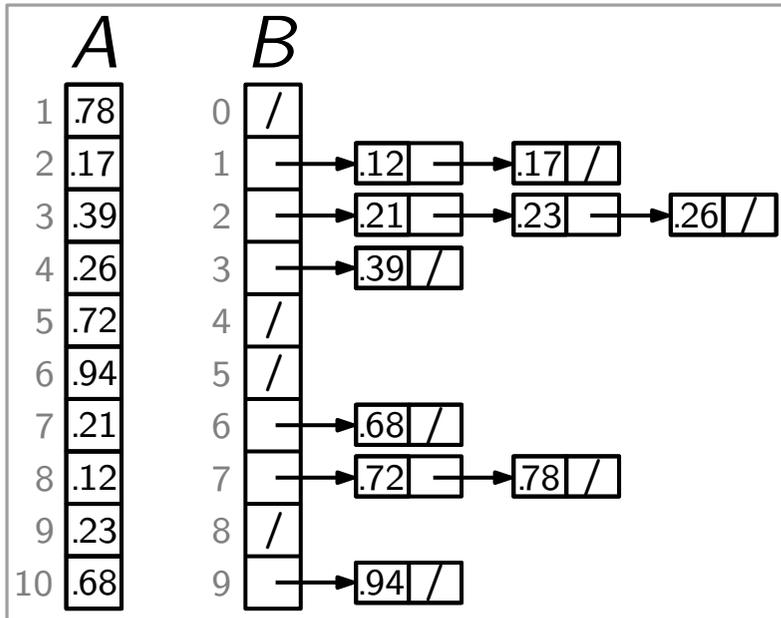
2 Fälle:

– $A[i]$ und $A[j]$ in der gleichen Liste

– $A[i]$ und $A[j]$ in verschiedenen Listen

BucketSort

[CLRS]



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n - 1]$ 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 $B[i] = \left[\frac{i}{n}, \frac{i+1}{n} \right) \cap A$

hänge $B[0], \dots, B[n - 1]$ aneinander

kopiere das Ergebnis nach $A[1..n]$

Korrektheit?

2 Fälle:

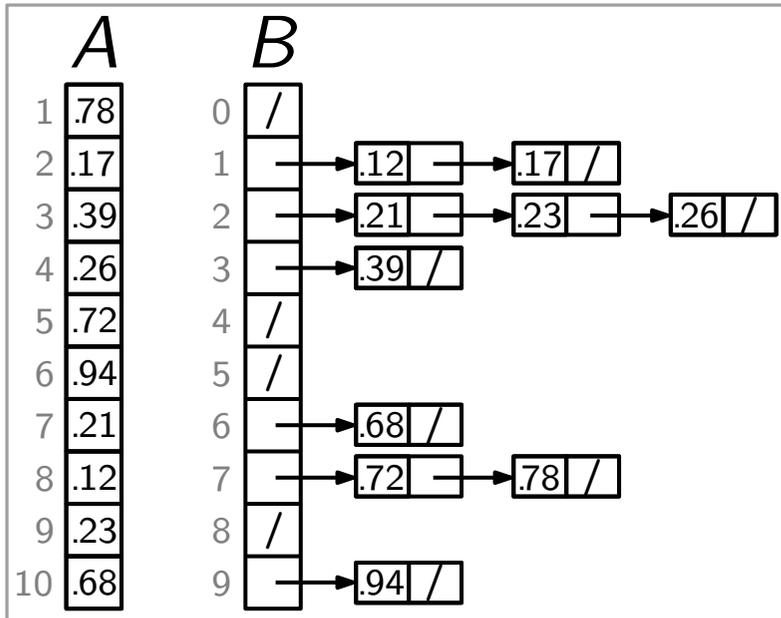
– $A[i]$ und $A[j]$ in der gleichen Liste

– $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit?

BucketSort

[CLRS]



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n - 1]$ 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 $B[i] = \left[\frac{i}{n}, \frac{i+1}{n} \right) \cap A$

hänge $B[0], \dots, B[n - 1]$ aneinander

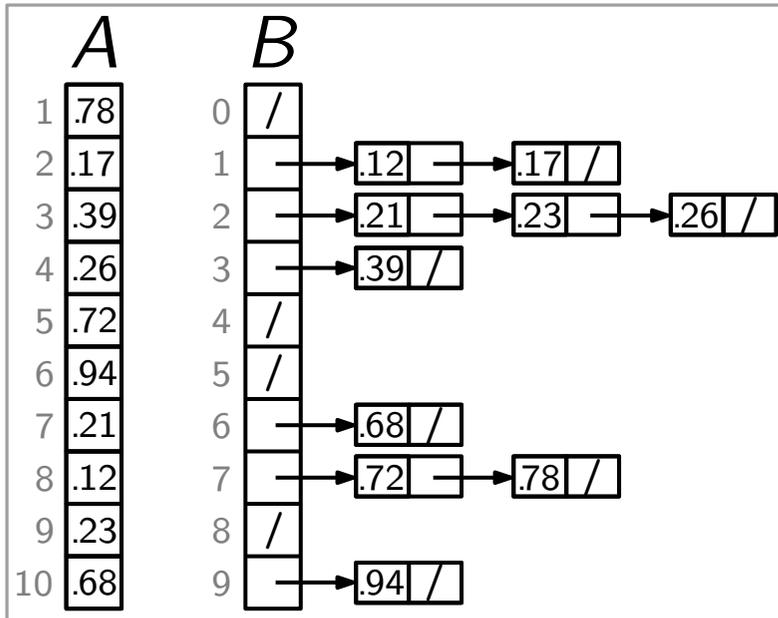
kopiere das Ergebnis nach $A[1..n]$

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

BucketSort

[CLRS]



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n - 1]$ 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 $B[i] = \left[\frac{i}{n}, \frac{i+1}{n} \right) \cap A$

hänge $B[0], \dots, B[n - 1]$ aneinander

kopiere das Ergebnis nach $A[1..n]$

Korrektheit?

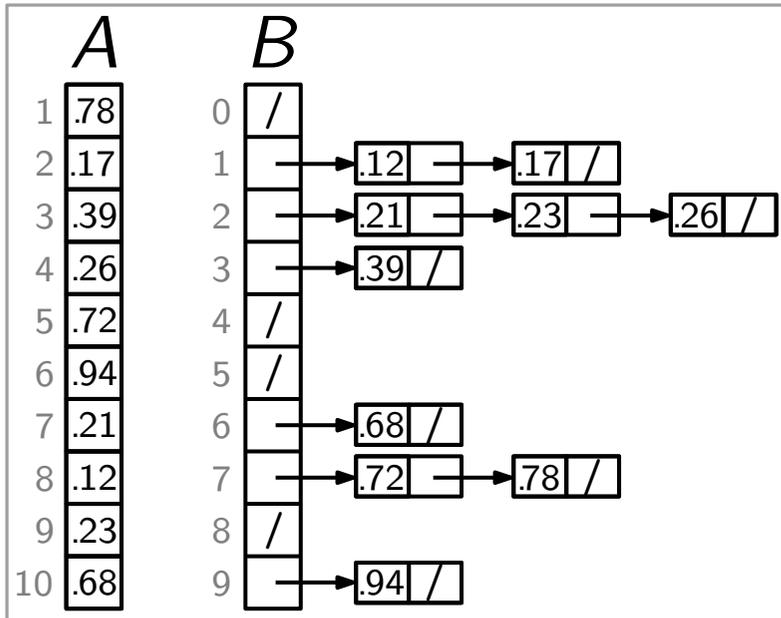
- 2 Fälle:
- $A[i]$ und $A[j]$ in der gleichen Liste
 - $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit?

- *erwartet*, hängt von den zufälligen Zahlen in A ab
- hängt vom Sortieralgorithmus in Zeile 6 ab;

BucketSort

[CLRS]



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n - 1]$ 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 $B[i] = \left[\frac{i}{n}, \frac{i+1}{n} \right) \cap A$

hänge $B[0], \dots, B[n - 1]$ aneinander
kopiere das Ergebnis nach $A[1..n]$

Korrektheit?

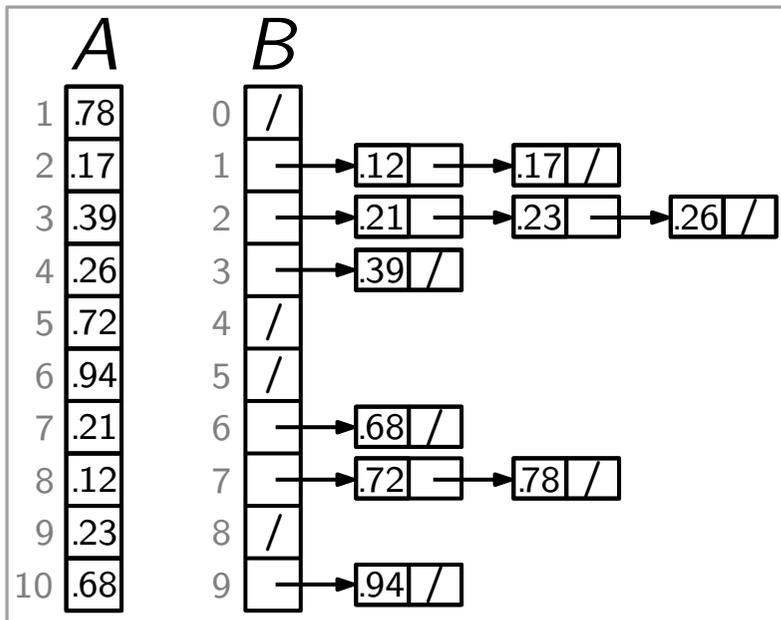
- 2 Fälle:
- $A[i]$ und $A[j]$ in der gleichen Liste
 - $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit?

- *erwartet*, hängt von den zufälligen Zahlen in A ab
- hängt vom Sortieralgorithmus in Zeile 6 ab;
wir nehmen InsertionSort:

BucketSort

[CLRS]



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n - 1]$ 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 $B[i] = \left[\frac{i}{n}, \frac{i+1}{n} \right) \cap A$

hänge $B[0], \dots, B[n - 1]$ aneinander

kopiere das Ergebnis nach $A[1..n]$

Korrektheit?

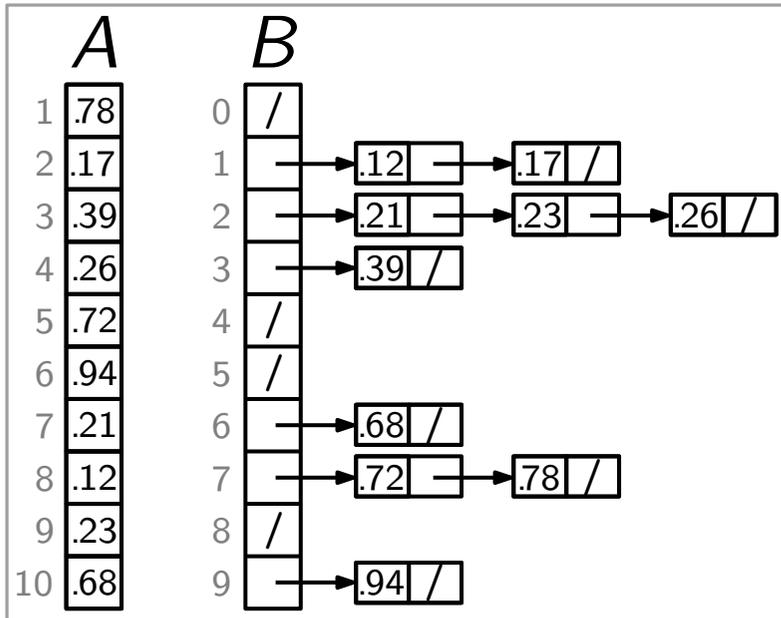
- 2 Fälle:
- $A[i]$ und $A[j]$ in der gleichen Liste
 - $A[i]$ und $A[j]$ in verschiedenen Listen

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

[CLRS]



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n - 1]$ 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 $A[1..n]$

Korrektheit?

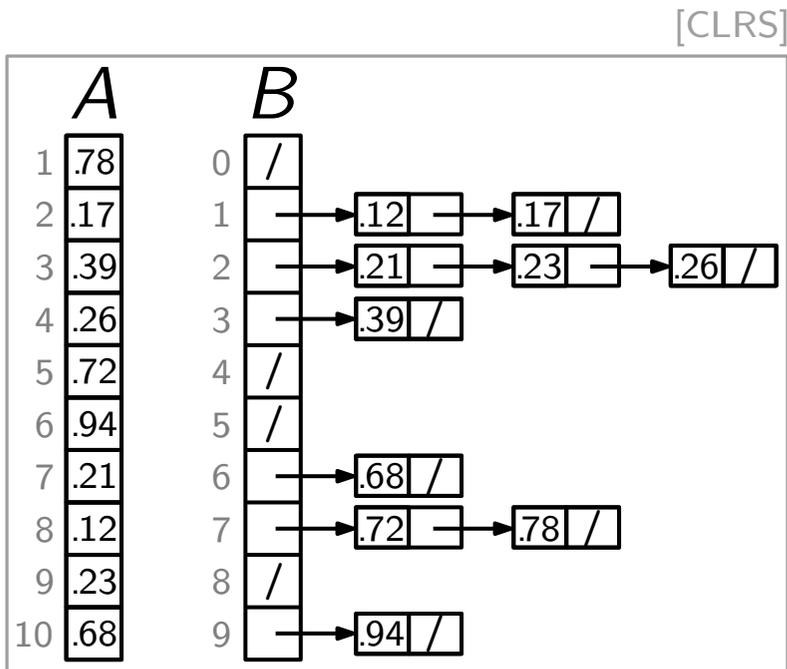
2 Fälle: – $A[i]$ und $A[j]$ in der gleichen Liste

– $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit?

$T_{BS}(n) =$

BucketSort



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n - 1]$ 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 $A[1..n]$

Korrektheit?

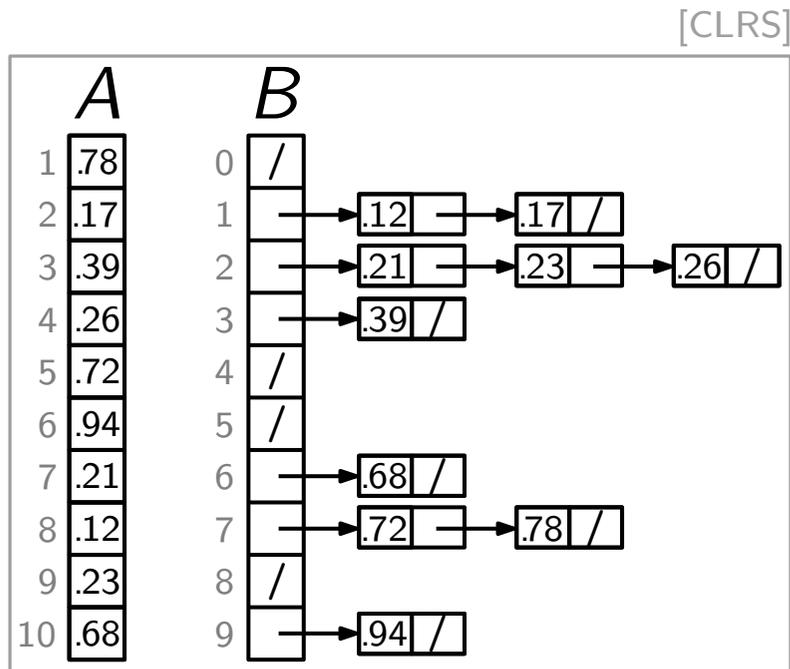
2 Fälle: – $A[i]$ und $A[j]$ in der gleichen Liste
– $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit?

$T_{BS}(n) =$



BucketSort



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n - 1]$ 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 $A[1..n]$

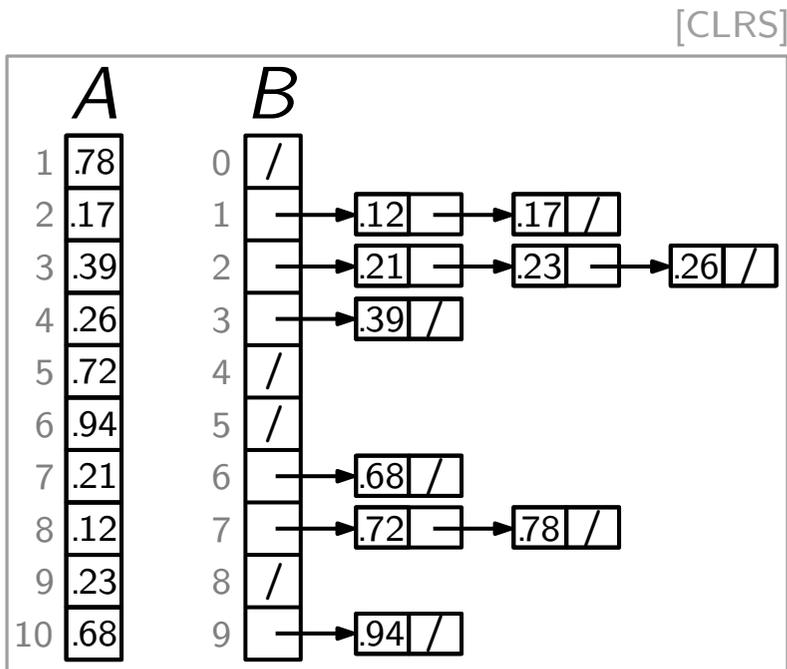
Korrektheit?

2 Fälle: – $A[i]$ und $A[j]$ in der gleichen Liste
– $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit?

$$T_{BS}(n) = \Theta(n) +$$

BucketSort



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n-1]$ 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 $A[1..n]$

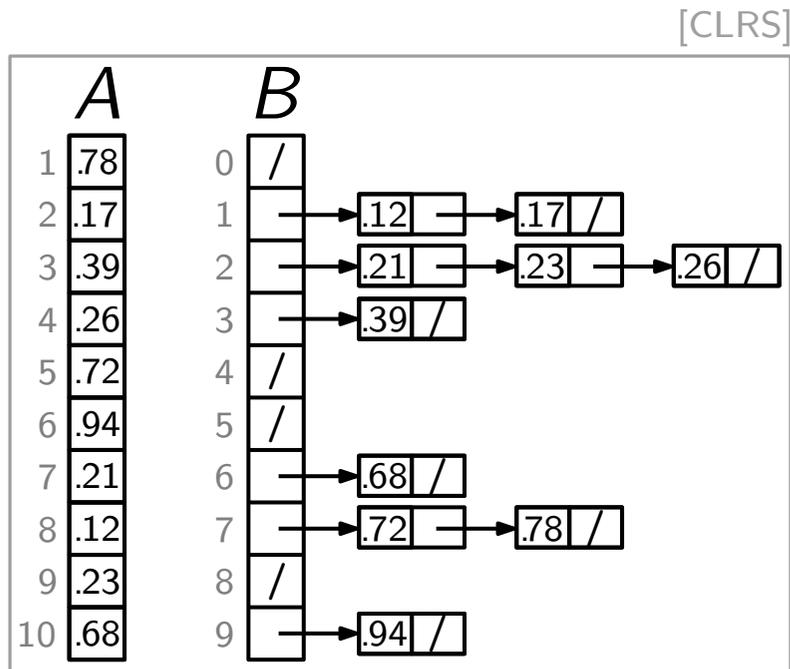
Korrektheit?

2 Fälle: – $A[i]$ und $A[j]$ in der gleichen Liste
– $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit?

$$T_{BS}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{IS}(n_i),$$

BucketSort



BucketSort(Feld A von Zahlen in $[0, 1)$)

$n = A.length$

lege Feld $B[0..n - 1]$ 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 $A[1..n]$

Korrektheit?

2 Fälle: – $A[i]$ und $A[j]$ in der gleichen Liste
– $A[i]$ und $A[j]$ in verschiedenen Listen

Laufzeit?

$$T_{BS}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{IS}(n_i),$$

wobei $T_{IS}(\cdot)$ Laufzeit von InsertionSort
 n_i ZV für Länge der Liste $B[i]$

Erwartete Laufzeit von BucketSort

$$T_{BS}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{IS}(n_i)$$

Erwartete Laufzeit von BucketSort

$$T_{BS}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{IS}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

Erwartete Laufzeit von BucketSort

$$T_{BS}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{IS}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$E[T_{BS}(n)] =$$

Erwartete Laufzeit von BucketSort

$$T_{\text{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\text{IS}}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$E[T_{\text{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)]$$

Erwartete Laufzeit von BucketSort

$$T_{BS}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{IS}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$\begin{aligned} E[T_{BS}(n)] &= E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)] \\ &= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)] \end{aligned}$$


Erwartete Laufzeit von BucketSort

$$T_{\text{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\text{IS}}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$\begin{aligned} E[T_{\text{BS}}(n)] &= E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)] \\ &= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)] \end{aligned}$$

Linearität des Erwartungswerts
 $E[X + Y] = E[X] + E[Y]$

Erwartete Laufzeit von BucketSort

$$T_{BS}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{IS}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$E[T_{BS}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2])$$

Linearität des Erwartungswerts

Erwartete Laufzeit von BucketSort

$$T_{\text{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\text{IS}}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$E[T_{\text{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2])$$

Linearität des Erwartungswerts

Für $a \in \mathbb{R}$: $E[aX] = a \cdot E[X]$

Erwartete Laufzeit von BucketSort

$$T_{\text{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\text{IS}}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$\begin{aligned} E[T_{\text{BS}}(n)] &= E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)] \\ &= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)] \\ &= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2]) = \end{aligned}$$

Erwartete Laufzeit von BucketSort

$$T_{BS}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{IS}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$E[T_{BS}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2]) =$$

Behauptung: $E[n_i^2] \leq 2 - \frac{1}{n}$

Erwartete Laufzeit von BucketSort

$$T_{\text{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\text{IS}}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$E[T_{\text{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2]) = \Theta(n)$$

Behauptung: $E[n_i^2] \leq 2 - \frac{1}{n}$

Erwartete Laufzeit von BucketSort

$$T_{\text{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\text{IS}}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$E[T_{\text{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2]) = \Theta(n)$$

Behauptung: $E[n_i^2] \leq 2 - \frac{1}{n}$

Beweis.

Erwartete Laufzeit von BucketSort

$$T_{\text{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\text{IS}}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$E[T_{\text{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2]) = \Theta(n)$$

Behauptung: $E[n_i^2] \leq 2 - \frac{1}{n}$

Beweis. Def. Indikator-ZV $X_j := 1$, falls $A[j]$ in Eimer i fällt. *fest!*

Erwartete Laufzeit von BucketSort

$$T_{\text{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\text{IS}}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$E[T_{\text{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2]) = \Theta(n)$$

Behauptung: $E[n_i^2] \leq 2 - \frac{1}{n}$

Beweis. Def. Indikator-ZV $X_j := 1$, falls $A[j]$ in Eimer i fällt. *fest!*

$$\Rightarrow n_i =$$

Erwartete Laufzeit von BucketSort

$$T_{\text{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\text{IS}}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$E[T_{\text{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2]) = \Theta(n)$$

Behauptung: $E[n_i^2] \leq 2 - \frac{1}{n}$

Beweis. Def. Indikator-ZV $X_j := 1$, falls $A[j]$ in Eimer i fällt. *fest!*

$$\Rightarrow n_i = \sum_{j=1}^n X_j$$

Erwartete Laufzeit von BucketSort

$$T_{\text{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\text{IS}}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$E[T_{\text{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2]) = \Theta(n)$$

Behauptung: $E[n_i^2] \leq 2 - \frac{1}{n}$

Beweis. Def. Indikator-ZV $X_j := 1$, falls $A[j]$ in Eimer i fällt. *fest!*

$$\Rightarrow n_i = \sum_{j=1}^n X_j \quad E[X_j] =$$

Erwartete Laufzeit von BucketSort

$$T_{\text{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\text{IS}}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$E[T_{\text{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2]) = \Theta(n)$$

Behauptung: $E[n_i^2] \leq 2 - \frac{1}{n}$

Beweis. Def. Indikator-ZV $X_j := 1$, falls $A[j]$ in Eimer i fällt. *fest!*

$$\Rightarrow n_i = \sum_{j=1}^n X_j \quad E[X_j] = \Pr[X_j = 1] =$$

Erwartete Laufzeit von BucketSort

$$T_{\text{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\text{IS}}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$E[T_{\text{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2]) = \Theta(n)$$

Behauptung: $E[n_i^2] \leq 2 - \frac{1}{n}$

Beweis. Def. Indikator-ZV $X_j := 1$, falls $A[j]$ in Eimer i fällt. *fest!*

$$\Rightarrow n_i = \sum_{j=1}^n X_j \quad E[X_j] = \Pr[X_j = 1] = 1/n$$

Erwartete Laufzeit von BucketSort

$$T_{\text{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\text{IS}}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$E[T_{\text{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2]) = \Theta(n)$$

Behauptung: $E[n_i^2] \leq 2 - \frac{1}{n}$

Beweis. Def. Indikator-ZV $X_j := 1$, falls $A[j]$ in Eimer i fällt. *fest!*

$$\Rightarrow n_i = \sum_{j=1}^n X_j \quad E[X_j] = \Pr[X_j = 1] = 1/n$$

$$\Rightarrow n_i^2 = \left(\sum_{j=1}^n X_j \right)^2 =$$

Erwartete Laufzeit von BucketSort

$$T_{\text{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\text{IS}}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$E[T_{\text{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2]) = \Theta(n)$$

Behauptung: $E[n_i^2] \leq 2 - \frac{1}{n}$

Beweis. Def. Indikator-ZV $X_j := 1$, falls $A[j]$ in Eimer i fällt. *fest!*

$$\Rightarrow n_i = \sum_{j=1}^n X_j \quad E[X_j] = \Pr[X_j = 1] = 1/n$$

$$\Rightarrow n_i^2 = \left(\sum_{j=1}^n X_j \right)^2 = \sum_{j=1}^n \sum_{k=1}^n X_j X_k$$

Erwartete Laufzeit von BucketSort

$$T_{\text{BS}}(n) = \Theta(n) + \sum_{i=0}^{n-1} T_{\text{IS}}(n_i) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

$$E[T_{\text{BS}}(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2]) = \Theta(n)$$

Behauptung: $E[n_i^2] \leq 2 - \frac{1}{n}$

Beweis. Def. Indikator-ZV $X_j := 1$, falls $A[j]$ in Eimer i fällt. *fest!*

$$\Rightarrow n_i = \sum_{j=1}^n X_j \quad E[X_j] = \Pr[X_j = 1] = 1/n$$

$$\Rightarrow n_i^2 = \left(\sum_{j=1}^n X_j \right)^2 = \sum_{j=1}^n \sum_{k=1}^n X_j X_k$$

$$= \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

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

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

Erwartete Laufzeit von BucketSort

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

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:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$E[X_j^2] =$$

Erwartete Laufzeit von BucketSort

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$

Erwartete Laufzeit von BucketSort

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] \end{aligned}$$

Erwartete Laufzeit von BucketSort

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n} \end{aligned}$$

Erwartete Laufzeit von BucketSort

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n} \end{aligned}$$

Erwartete Laufzeit von BucketSort

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] && \text{unabhängig von } j! \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n} \end{aligned}$$

Erwartete Laufzeit von BucketSort

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$

unabhängig von j!

$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n}$$

$$E[X_j X_k] =$$

Erwartete Laufzeit von BucketSort

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$

unabhängig von j!

$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n}$$

$$E[X_j X_k] =$$

für $j \neq k$ sind X_j und X_k *unabhängig*

Erwartete Laufzeit von BucketSort

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] && \text{unabhängig von } j! \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n} \end{aligned}$$

$$E[X_j X_k] = E[X_j] \cdot E[X_k]$$

für $j \neq k$ sind X_j und X_k unabhängig

Erwartete Laufzeit von BucketSort

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0]$$

unabhängig von j!

$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n}$$

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

für $j \neq k$ sind X_j und X_k *unabhängig*

Erwartete Laufzeit von BucketSort

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] && \text{unabhängig von } j! \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n} \end{aligned}$$

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

für $j \neq k$ sind X_j und X_k unabhängig

Erwartete Laufzeit von BucketSort

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] && \text{unabhängig von } j! \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n} \end{aligned}$$

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

für $j \neq k$ sind X_j und X_k unabhängig

unabh. von j und k !

Erwartete Laufzeit von BucketSort

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] && \text{unabhängig von } j! \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n} \end{aligned}$$

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

für $j \neq k$ sind X_j und X_k unabhängig

unabh. von j und k !

Fasse die Zwischenergebnisse zusammen:

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:

$$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] \quad \text{unabhängig von } j!$$

$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n}$$

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2} \quad \text{unabh. von } j \text{ und } k!$$

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

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] && \text{unabhängig von } j! \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n} \end{aligned}$$

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

für $j \neq k$ sind X_j und X_k unabhängig

unabh. von j und k !

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

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] && \text{unabhängig von } j! \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n} \end{aligned}$$

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

für $j \neq k$ sind X_j und X_k unabhängig

unabh. von j und k !

Fasse die Zwischenergebnisse zusammen:

$$\begin{aligned} E[n_i^2] &= \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k] \\ &= \end{aligned}$$

Erwartete Laufzeit von BucketSort

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] && \text{unabhängig von } j! \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n} \end{aligned}$$

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2} \quad \leftarrow \text{unabh. von } j \text{ und } k!$$

für $j \neq k$ sind X_j und X_k unabhängig

Fasse die Zwischenergebnisse zusammen:

$$\begin{aligned} E[n_i^2] &= \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k] \\ &= \sum_{j=1}^n \frac{1}{n} + \sum_{j=1}^n (n-1) \cdot \frac{1}{n^2} \end{aligned}$$

Erwartete Laufzeit von BucketSort

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] && \text{unabhängig von } j! \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n} \end{aligned}$$

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

für $j \neq k$ sind X_j und X_k unabhängig

unabh. von j und k !

Fasse die Zwischenergebnisse zusammen:

$$\begin{aligned} E[n_i^2] &= \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k] \\ &= n \cdot \frac{1}{n} + n \cdot (n-1) \cdot \frac{1}{n^2} \end{aligned}$$

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:

$$E[X_j^2] = 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] \quad \text{unabhängig von } j!$$

$$= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n}$$

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2} \quad \text{unabh. von } j \text{ und } k!$$

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:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] && \text{unabhängig von } j! \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n} \end{aligned}$$

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2} \quad \leftarrow \text{unabh. von } j \text{ und } k!$$

für $j \neq k$ sind X_j und X_k unabhängig

Fasse die Zwischenergebnisse zusammen:

$$\begin{aligned} E[n_i^2] &= \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k] \\ &= n \cdot \frac{1}{n} + n \cdot (n-1) \cdot \frac{1}{n^2} = 1 + \frac{n-1}{n} = \end{aligned}$$

Erwartete Laufzeit von BucketSort

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] && \text{unabhängig von } j! \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n} \end{aligned}$$

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2} \quad \text{unabh. von } j \text{ und } k!$$

für $j \neq k$ sind X_j und X_k unabhängig

Fasse die Zwischenergebnisse zusammen:

$$\begin{aligned} E[n_i^2] &= \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k] \\ &= n \cdot \frac{1}{n} + n \cdot (n-1) \cdot \frac{1}{n^2} = 1 + \frac{n-1}{n} = 2 - \frac{1}{n} \end{aligned}$$

Erwartete Laufzeit von BucketSort

Behauptung:
 $E[n_i^2] \leq 2 - \frac{1}{n}$

$$\text{Es gilt } n_i^2 = \sum_{j=1}^n X_j^2 + \sum_{j=1}^n \sum_{k \neq j} X_j X_k$$

$$\Rightarrow E[n_i^2] = \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k]$$

Behandle die beiden Typen von Erwartungswerten getrennt:

$$\begin{aligned} E[X_j^2] &= 1 \cdot \Pr[X_j^2 = 1] + 0 \cdot \Pr[X_j^2 = 0] && \text{unabhängig von } j! \\ &= 1 \cdot \Pr[X_j = 1] + 0 \cdot \Pr[X_j = 0] = 1 \cdot \frac{1}{n} + 0 = \frac{1}{n} \end{aligned}$$

$$E[X_j X_k] = E[X_j] \cdot E[X_k] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

für $j \neq k$ sind X_j und X_k unabhängig

unabh. von j und k !

Fasse die Zwischenergebnisse zusammen:

$$\begin{aligned} E[n_i^2] &= \sum_{j=1}^n E[X_j^2] + \sum_{j=1}^n \sum_{k \neq j} E[X_j X_k] \\ &= n \cdot \frac{1}{n} + n \cdot (n-1) \cdot \frac{1}{n^2} = 1 + \frac{n-1}{n} = 2 - \frac{1}{n} \quad \square \end{aligned}$$

Zusammenfassung

- Jedes *vergleichsbasierte* Sortierverfahren braucht im schlechtesten Fall $\Omega(n \log n)$ Vergleiche für n Zahlen.
- **CountingSort** sortiert Zahlen in $\{0, \dots, k\}$ (*stabil!*)
Laufzeit für n Zahlen: $O(n + k)$
- **RadixSort** sortiert s -stellige b -adische Zahlen
Laufzeit für n Zahlen: $O(s \cdot (n + b))$
- **BucketSort** sortiert gleichverteilte zufällige Zahlen
erwartete Laufzeit für n Zahlen: $O(n)$

Zusammenfassung

- Jedes *vergleichsbasierte* Sortierverfahren braucht im schlechtesten Fall $\Omega(n \log n)$ Vergleiche für n Zahlen.
 - **CountingSort** sortiert Zahlen in $\{0, \dots, k\}$ (*stabil!*)
Laufzeit für n Zahlen: $O(n + k)$
 - **RadixSort** sortiert s -stellige b -adische Zahlen
Laufzeit für n Zahlen: $O(s \cdot (n + b))$
 - **BucketSort** sortiert gleichverteilte zufällige Zahlen
erwartete Laufzeit für n Zahlen: $O(n)$
- Bem.** Die Idee mit den (gleichgroßen) Eimern ist natürlich nicht nur auf Zufallszahlen beschränkt, aber hier lässt sie sich hübsch analysieren.