Aufgabe 1.1

T. Adam, M. ben Ahmed

Universität Osnabrück

Algorithm Engineering

November 8, 2020

Aufgabe 1.1 (a) - Externe Queue

Queue

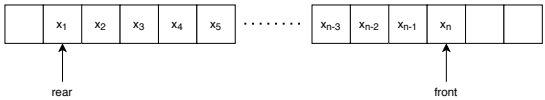
1. `void enqueue(type v)`
2. `type v dequeue()`

- Queue funktioniert nach dem First-In/First-Out Prinzip (**FIFO**)
- Array Implementation mit **front** und **rear** Zeiger

Aufgabe 1.1 (a) - Externe Queue

Standard Queue

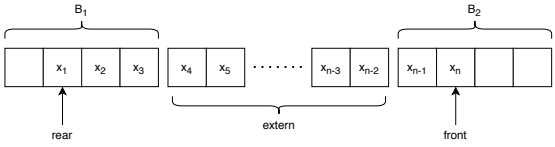
- *front*-Zeiger auf Anfang der Queue
- *rear*-Zeiger auf Ende der Queue
- `enqueue()` an *rear*
- `dequeue()` an *front*



Aufgabe 1.1 (a) - Externe Queue

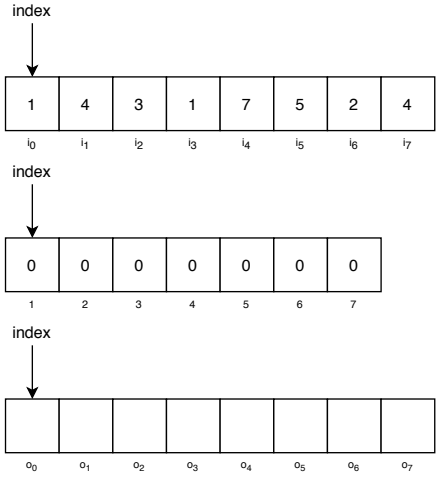
Externe Queue

- Falls $n > B$, halte nur ersten und letzten Block im Speicher
- Blöcke dazwischen $\rightarrow$ extern



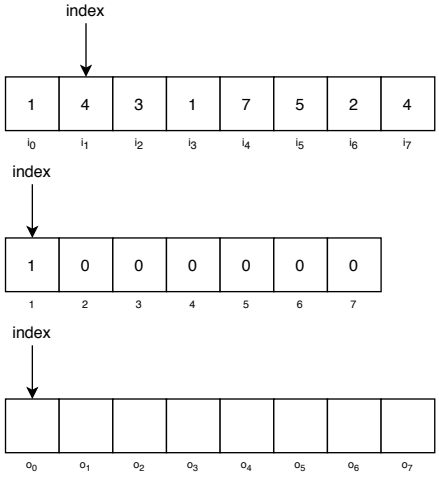
Aufgabe 1.1 (b) - Counting Sort

- array der Länge n
- Elemente in i müssen nicht eindeutig sein
- Temporäres array der Länge k
- k ist die Ziffer mit dem größten Wert



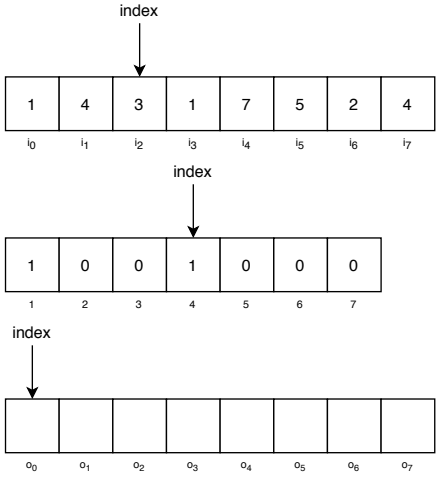
Aufgabe 1.1 (b) - Counting Sort

- Input array wird durchlaufen
- Im temp. array wird gezählt wie oft jede Ziffer vorkommt



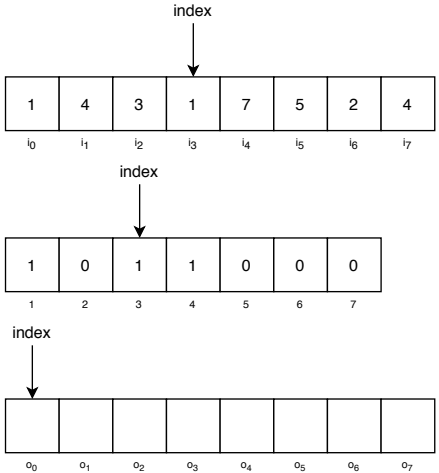
Aufgabe 1.1 (b) - Counting Sort

- Input array wird durchlaufen
- Im temp. array wird gezählt wie oft jede Ziffer vorkommt



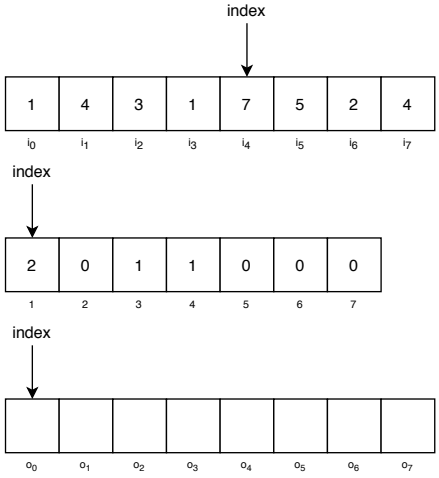
Aufgabe 1.1 (b) - Counting Sort

- Input array wird durchlaufen
- Im temp. array wird gezählt wie oft jede Ziffer vorkommt



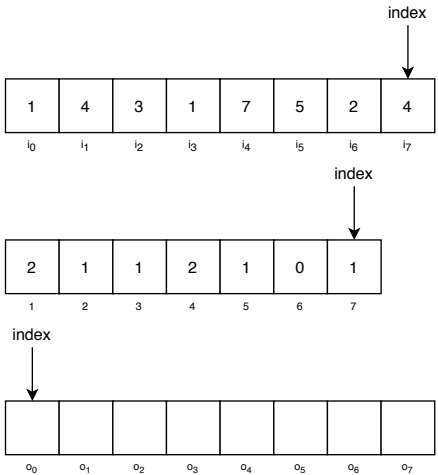
Aufgabe 1.1 (b) - Counting Sort

- Input array wird durchlaufen
- Im temp. array wird gezählt wie oft jede Ziffer vorkommt



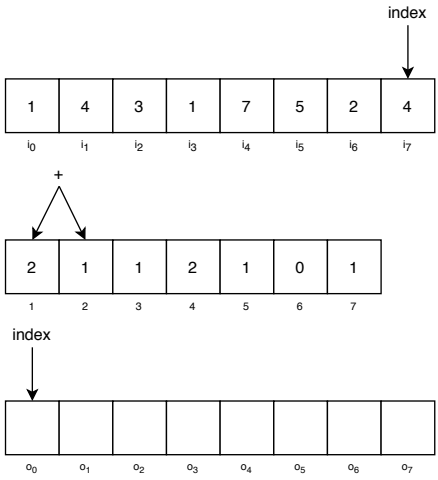
Aufgabe 1.1 (b) - Counting Sort

- Das temp. array enthält die Anzahl der gleichen Elemente



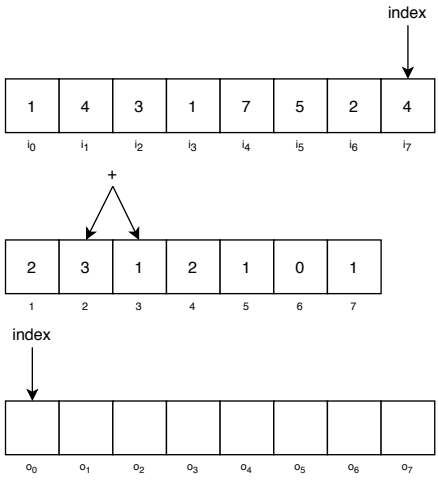
Aufgabe 1.1 (b) - Counting Sort

- Einträge des temp. array werden paarweise addiert
- Summe wird in die jeweils rechte Zelle eingetragen



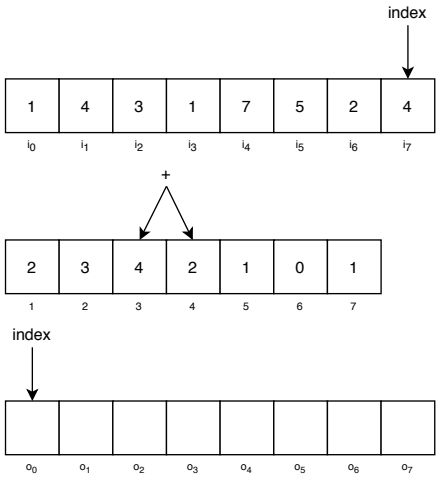
Aufgabe 1.1 (b) - Counting Sort

- Einträge des temp. array werden paarweise addiert
- Summe wird in die jeweils rechte Zelle eingetragen



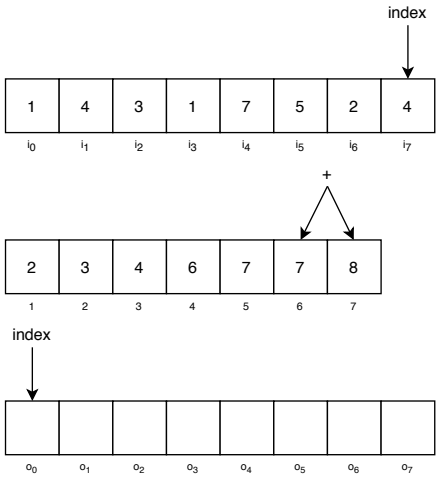
Aufgabe 1.1 (b) - Counting Sort

- Einträge des temp. array werden paarweise addiert
- Summe wird in die jeweils rechte Zelle eingetragen



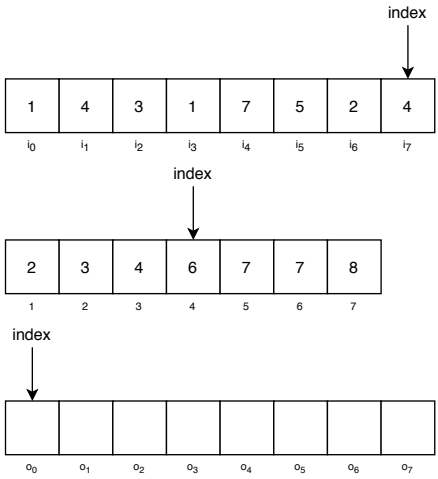
Aufgabe 1.1 (b) - Counting Sort

- Alle Einträge wurden aufsummiert
- Es ergibt sich eine aufsteigende Reihenfolge



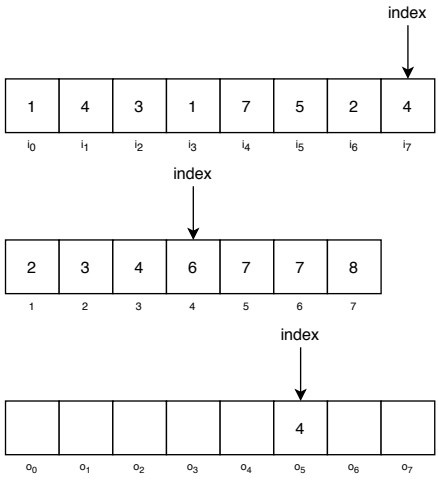
Aufgabe 1.1 (b) - Counting Sort

- Input array wird rückwärts durchlaufen
- Es wird an der i-ten Stelle des temp. arrays nach dem output Index geschaut



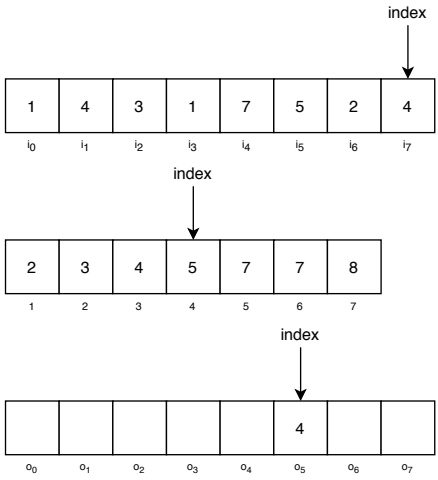
Aufgabe 1.1 (b) - Counting Sort

- Element aus dem input array wird an die entsprechende Stelle im output array geschrieben



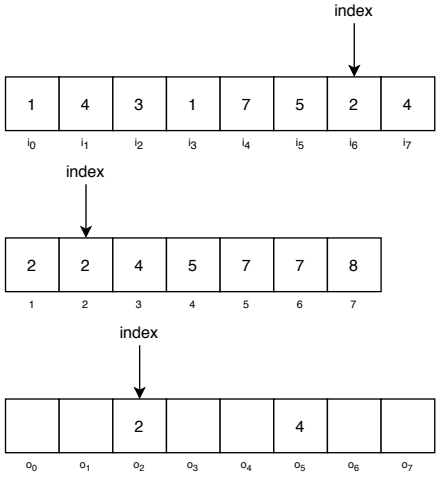
Aufgabe 1.1 (b) - Counting Sort

- Wert im temp. array wird um 1 dekrementiert



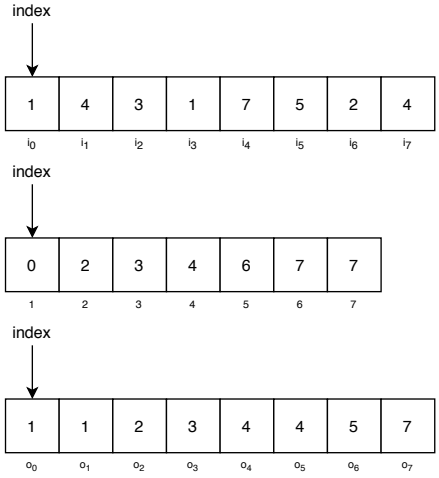
Aufgabe 1.1 (b) - Counting Sort

- Element aus dem input array wird an die entsprechende Stelle im output array geschrieben



Aufgabe 1.1 (b) - Counting Sort

- Im output array befinden sich die sortierten Werte des input arrays
- Insbesondere bleibt die Reihenfolge gleicher Elemente erhalten



Aufgabe 1.1 (b) - Counting Sort

- Iterieren über Input Array (inkl. zählen): $\mathcal{O}(n/B)$

I/O Fallunterscheidung

1. $k = \alpha \cdot M$ mit $\alpha = 0,5$

Aufgabe 1.1 (b) - Counting Sort

I/O Fallunterscheidung

1. $k = \alpha \cdot M$ mit $\alpha = 0,5$

- Iterieren über Input Array (inkl. zählen): $\mathcal{O}(n/B)$
- Aufsummieren in temp. Array: kein I/O

Aufgabe 1.1 (b) - Counting Sort

I/O Fallunterscheidung

1. $k = \alpha \cdot M$ mit $\alpha = 0,5$

- Iterieren über Input Array (inkl. zählen): $\mathcal{O}(n/B)$
- Aufsummieren in temp. Array: kein I/O
- Zurück iterieren über Inputarray: $\mathcal{O}(n/B)$

Aufgabe 1.1 (b) - Counting Sort

I/O Fallunterscheidung

1. $k = \alpha \cdot M$ mit $\alpha = 0,5$

- Iterieren über Input Array (inkl. zählen): $\mathcal{O}(n/B)$
- Aufsummieren in temp. Array: kein I/O
- Zurück iterieren über Inputarray: $\mathcal{O}(n/B)$
- Verweis finden in temp. Array: kein I/O

Aufgabe 1.1 (b) - Counting Sort

I/O Fallunterscheidung

1. $k = \alpha \cdot M$ mit $\alpha = 0,5$

- Iterieren über Input Array (inkl. zählen): $\mathcal{O}(n/B)$
- Aufsummieren in temp. Array: kein I/O
- Zurück iterieren über Inputarray: $\mathcal{O}(n/B)$
- Verweis finden in temp. Array: kein I/O
- Schreiben in Output Array: $\mathcal{O}(n)$
- $\Rightarrow \mathcal{O}(n)$

Aufgabe 1.1 (b) - Counting Sort

- Iterieren über Input Array: $\mathcal{O}(n/B)$

I/O Fallunterscheidung

1. $k = \alpha \cdot M$ mit $\alpha > 1$

Aufgabe 1.1 (b) - Counting Sort

- Iterieren über Input Array: $\mathcal{O}(n/B)$
- Zählen in temp. Array: $\mathcal{O}(n)$

I/O Fallunterscheidung

1. $k = \alpha \cdot M$ mit $\alpha > 1$

Aufgabe 1.1 (b) - Counting Sort

I/O Fallunterscheidung

1. $k = \alpha \cdot M$ mit $\alpha > 1$

- Iterieren über Input Array: $\mathcal{O}(n/B)$
- Zählen in temp. Array: $\mathcal{O}(n)$
- Aufsummieren in temp. Array: $\mathcal{O}(k/B)$

Aufgabe 1.1 (b) - Counting Sort

I/O Fallunterscheidung

1. $k = \alpha \cdot M$ mit $\alpha > 1$

- Iterieren über Input Array: $\mathcal{O}(n/B)$
- Zählen in temp. Array: $\mathcal{O}(n)$
- Aufsummieren in temp. Array: $\mathcal{O}(k/B)$
- Zurück iterieren über Inputarray: $\mathcal{O}(n/B)$

Aufgabe 1.1 (b) - Counting Sort

I/O Fallunterscheidung

1. $k = \alpha \cdot M$ mit $\alpha > 1$

- Iterieren über Input Array: $\mathcal{O}(n/B)$
- Zählen in temp. Array: $\mathcal{O}(n)$
- Aufsummieren in temp. Array: $\mathcal{O}(k/B)$
- Zurück iterieren über Inputarray: $\mathcal{O}(n/B)$
- Verweis finden in temp. Array: $\mathcal{O}(n)$

Aufgabe 1.1 (b) - Counting Sort

I/O Fallunterscheidung

1. $k = \alpha \cdot M$ mit $\alpha > 1$

- Iterieren über Input Array: $\mathcal{O}(n/B)$
- Zählen in temp. Array: $\mathcal{O}(n)$
- Aufsummieren in temp. Array: $\mathcal{O}(k/B)$
- Zurück iterieren über Inputarray: $\mathcal{O}(n/B)$
- Verweis finden in temp. Array: $\mathcal{O}(n)$
- Schreiben in Output Array: $\mathcal{O}(n)$

Aufgabe 1.1 (b) - Counting Sort

I/O Fallunterscheidung

1. $k = \alpha \cdot M$ mit $\alpha > 1$

- Iterieren über Input Array: $\mathcal{O}(n/B)$
- Zählen in temp. Array: $\mathcal{O}(n)$
- Aufsummieren in temp. Array: $\mathcal{O}(k/B)$
- Zurück iterieren über Inputarray: $\mathcal{O}(n/B)$
- Verweis finden in temp. Array: $\mathcal{O}(n)$
- Schreiben in Output Array: $\mathcal{O}(n)$
- $\Rightarrow \mathcal{O}(n + k/B)$

Aufgabe 1.1 (b) - Counting Sort

Lässt sich Counting Sort **einfach** verbessern?