Karlsruher Institut für Technologie

# Algorithmen II
# Vorlesung am 08.01.2013

## Approximierende Algorithmen: Multiprozessor-Scheduling, Bin Packing

INSTITUT FÜR THEORETISCHE INFORMATIK · PROF. DR. DOROTHEA WAGNER

# Grundlagen



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Approximationsalgorithmen

**Ziel:** Entwerfe polynomielle Algorithmen für $\mathcal{NP}$-schwere Probleme, die nicht die optimale, aber immer eine beweisbar gute Lösung finden.

**Definition: Approximationsalgorithmus**
Sei $\Pi$ ein Minimierungsproblem, sowie $\mathcal{A}$ ein polynomieller Algorithmus, der für jede Instanz $I$ von $\Pi$ eine Lösung mit Wert $\mathcal{A}(I)$ liefert, sodass $\frac{\mathcal{A}(I)}{\text{OPT}(I)} \leq K$ gilt, wobei $K$ eine Konstante und $\text{OPT}(I)$ der Wert einer optimalen Lösung ist.
Dann heißt $\mathcal{A}$ *Approximationsalgorithmus mit relativer Gütegarantie*.

**Definition: Gütegarantie**
Sei $\mathcal{R}_{\mathcal{A}}(I) = \frac{\mathcal{A}(I)}{\text{OPT}(I)}$. Dann ist die *Approximationsgüte* $\mathcal{R}_{\mathcal{A}}$ definiert als

$$\mathcal{R}_{\mathcal{A}} = \inf\{r \geq 1 \mid \mathcal{R}_{\mathcal{A}}(I) \leq r \text{ für alle Instanzen } I \text{ von } \Pi\}.$$

$\mathcal{A}$ heißt $\varepsilon$-*approximierend*, falls $\mathcal{R}_{\mathcal{A}} \leq 1 + \varepsilon$.

**Maximierungsprobleme:**

- Für ein Maximierungsproblem betrachte $\frac{\text{OPT}(I)}{\mathcal{A}(I)}$ statt $\frac{\mathcal{A}(I)}{\text{OPT}(I)}$.

# Approximationsschemata

**Definition: PAS** **(Definition 7.10)**

Ein *(polynomielles) Approximationsschema (PAS)* für ein Optimierungsproblem $\Pi$ ist eine Familie von Algorithmen $\{\mathcal{A}_\varepsilon \mid \varepsilon > 0\}$, sodass $\mathcal{A}_\varepsilon$ ein $\varepsilon$-approximierender Algorithmus ist.

(häufig auch PTAS genannt)

**Beachte:** Die Laufzeit jedes Algorithmus $\mathcal{A}_\varepsilon$ ist nur polynomiell in der Eingabegröße und kann stark von $\varepsilon$ abhängen.

**Definition: FPAS** **(Definition 7.10)**

Ein Approximationsschema $\{\mathcal{A}_\varepsilon \mid \varepsilon > 0\}$ heißt *vollpolynomiell (FPAS)*, falls die Laufzeit jedes $\mathcal{A}_\varepsilon$ zusätzlich polynomiell in $\frac{1}{\varepsilon}$ ist.

(häufig auch FPTAS genannt)

**Beispiel:**

- Laufzeit $O\left(n^2 + n^{\frac{1}{\varepsilon}}\right)$ für $A_\varepsilon : \rightarrow$ PAS aber kein FPAS
- Laufzeit $O\left(\sqrt{n} \cdot 3^{\frac{1}{\varepsilon}}\right)$ für $A_\varepsilon : \rightarrow$ PAS aber kein FPAS
- Laufzeit $O\left(n^4 \cdot \left(\frac{1}{\varepsilon}\right)^2\right)$ für $A_\varepsilon : \rightarrow$ FPAS

Karlsruher Institut für Technologie

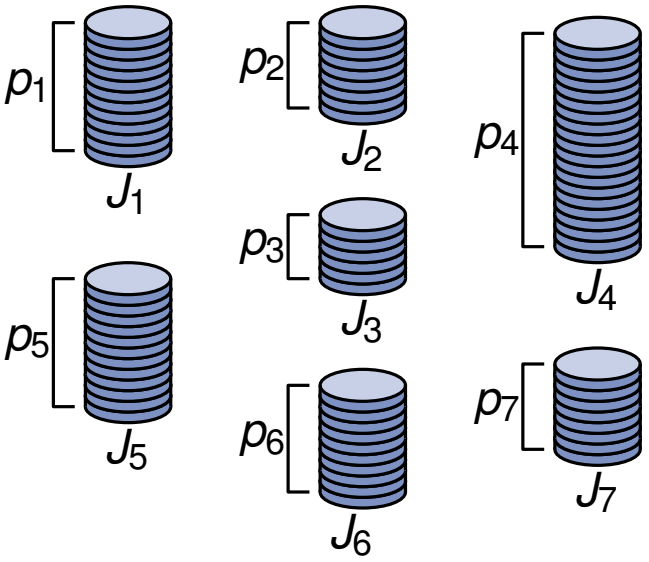
# Multiprozessor-Scheduling



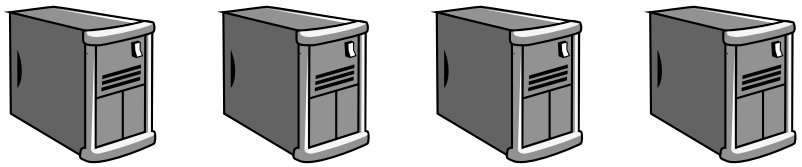
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Multiprozessor-Scheduling – Definition

$n$ Jobs $J_1, \ldots, J_n$ (in bel. Reihenfolge)
mit Bearbeitungsdauer $p_1, \ldots, p_n$.




$m < n$



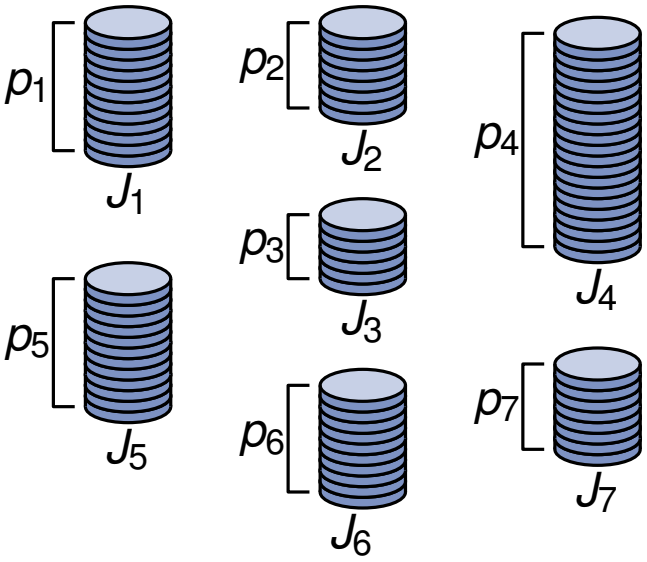
$m$ identische Maschinen

**Problem: MULTIPROZESSOR SCHEDULING**
Finde eine Zuweisung der Jobs auf die Maschinen, sodass der Zeitpunkt, zu dem alle Jobs abgearbeitet sind, minimal ist.
Dieser Zeitpunkt heißt auch MAKESPAN einer Zuweisung.

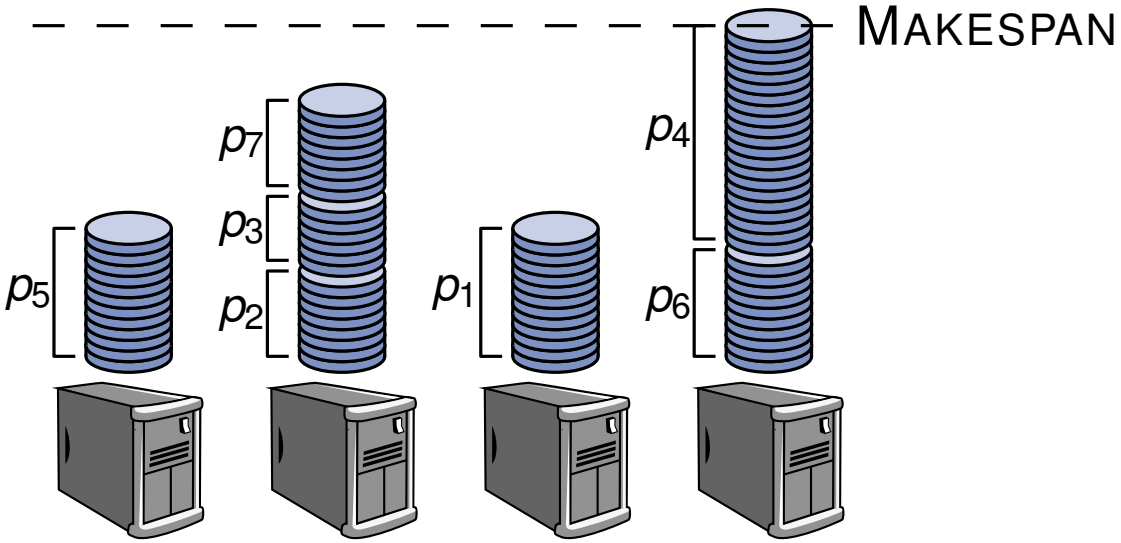
MULTIPROZESSOR SCHEDULING ist $\mathcal{NP}$-schwer.

# Multiprozessor-Scheduling – Definition

$n$ Jobs $J_1, \ldots, J_n$ (in bel. Reihenfolge)
mit Bearbeitungsdauer $p_1, \ldots, p_n$.




$m < n$



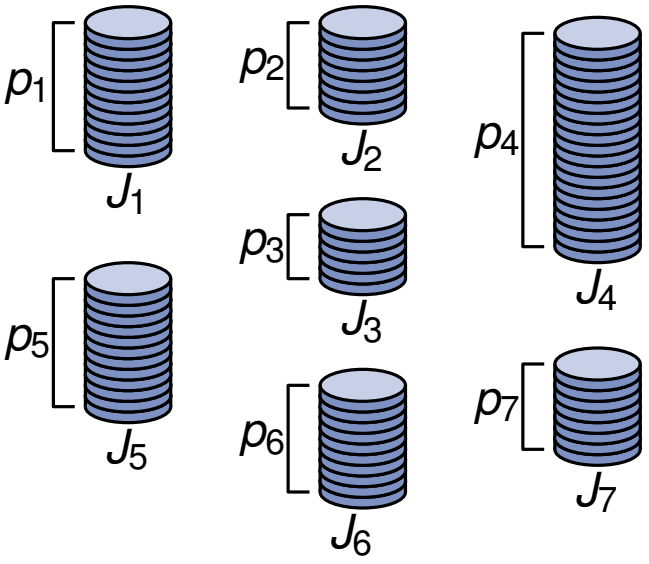

$m$ identische Maschinen

**Problem: MULTIPROZESSOR SCHEDULING**
Finde eine Zuweisung der Jobs auf die Maschinen, sodass der Zeitpunkt, zu dem alle Jobs abgearbeitet sind, minimal ist.
Dieser Zeitpunkt heißt auch MAKESPAN einer Zuweisung.

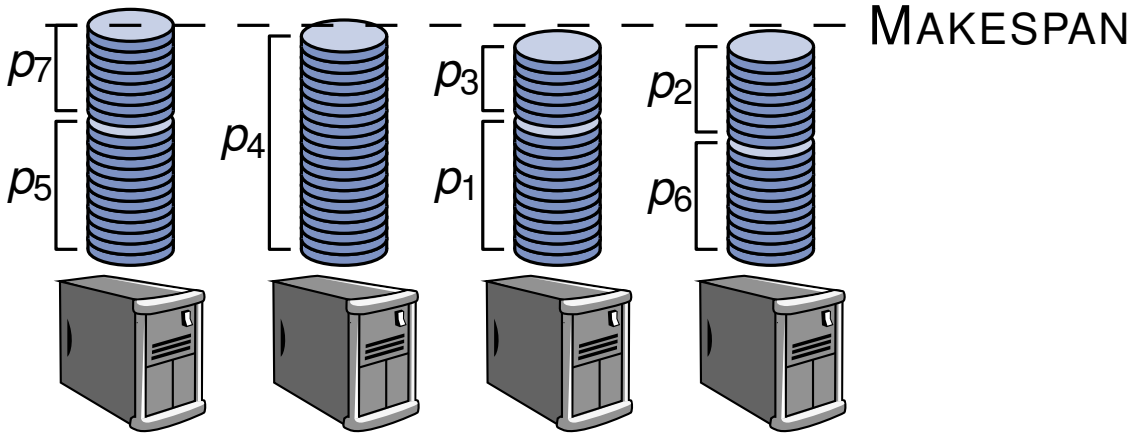
MULTIPROZESSOR SCHEDULING ist $\mathcal{NP}$-schwer.

# Multiprozessor-Scheduling – Definition

$n$ Jobs $J_1, \ldots, J_n$ (in bel. Reihenfolge)
mit Bearbeitungsdauer $p_1, \ldots, p_n$.




$m < n$




$m$ identische Maschinen

**Problem: MULTIPROZESSOR SCHEDULING**
Finde eine Zuweisung der Jobs auf die Maschinen, sodass der Zeitpunkt, zu dem alle Jobs abgearbeitet sind, minimal ist.
Dieser Zeitpunkt heißt auch MAKESPAN einer Zuweisung.

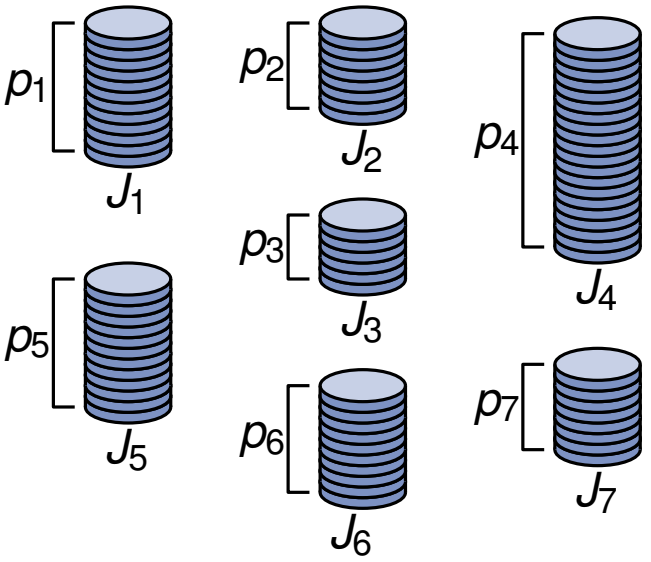
MULTIPROZESSOR SCHEDULING ist $\mathcal{NP}$-schwer.

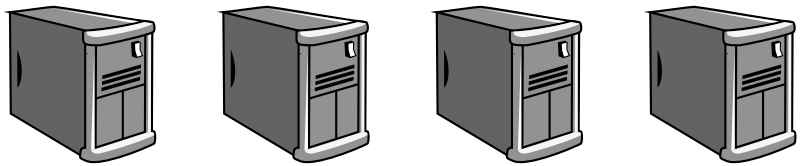
# Ein einfacher Algorithmus – LIST SCHEDULING

LIST SCHEDULING($J_1, \ldots, J_n, m$)

Lege die ersten $m$ Jobs auf die $m$ Maschinen.

Sobald eine Maschine frei ist, ordne ihr den nächsten der restlichen Jobs zu.





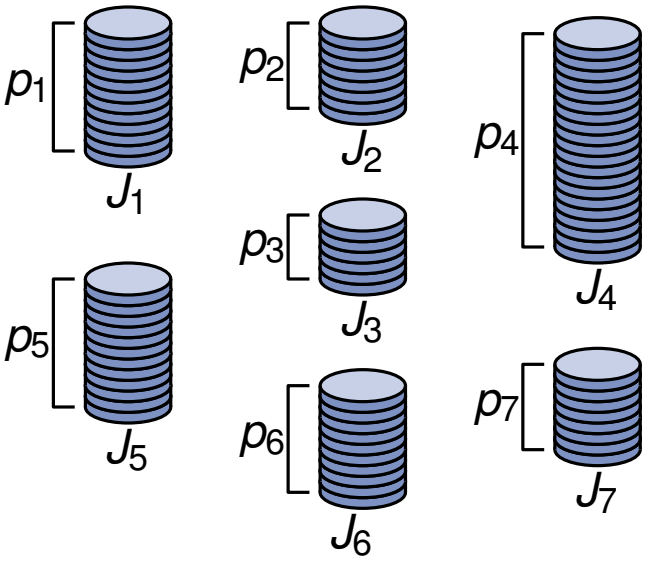
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

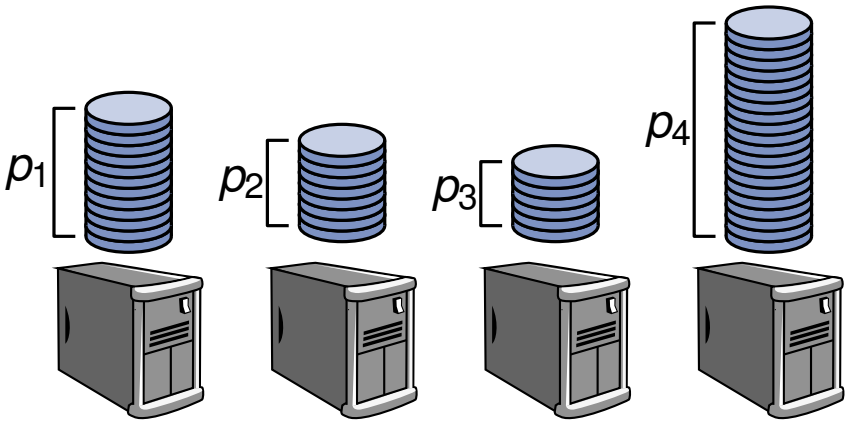
# Ein einfacher Algorithmus – LIST SCHEDULING

LIST SCHEDULING($J_1, \ldots, J_n, m$)

Lege die ersten $m$ Jobs auf die $m$ Maschinen.

Sobald eine Maschine frei ist, ordne ihr den nächsten der restlichen Jobs zu.





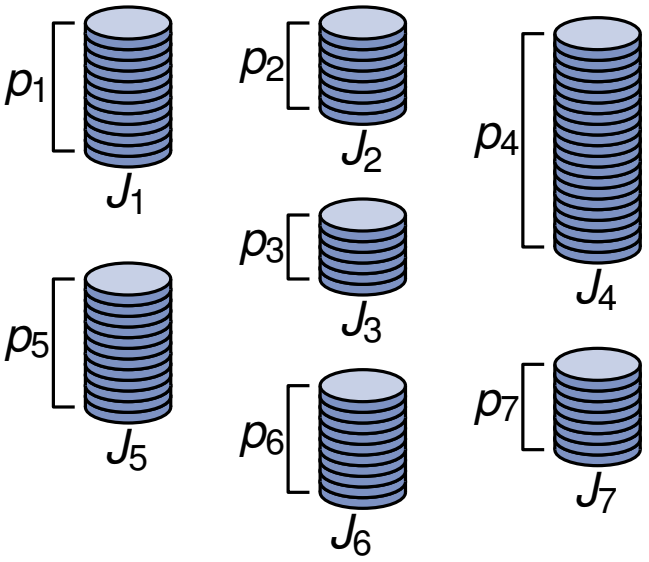

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

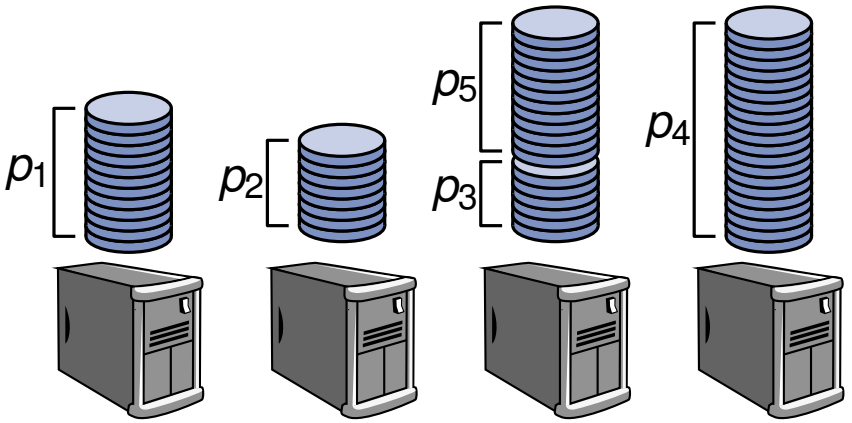
# Ein einfacher Algorithmus – LIST SCHEDULING

LIST SCHEDULING($J_1, \ldots, J_n, m$)

Lege die ersten $m$ Jobs auf die $m$ Maschinen.

Sobald eine Maschine frei ist, ordne ihr den nächsten der restlichen Jobs zu.





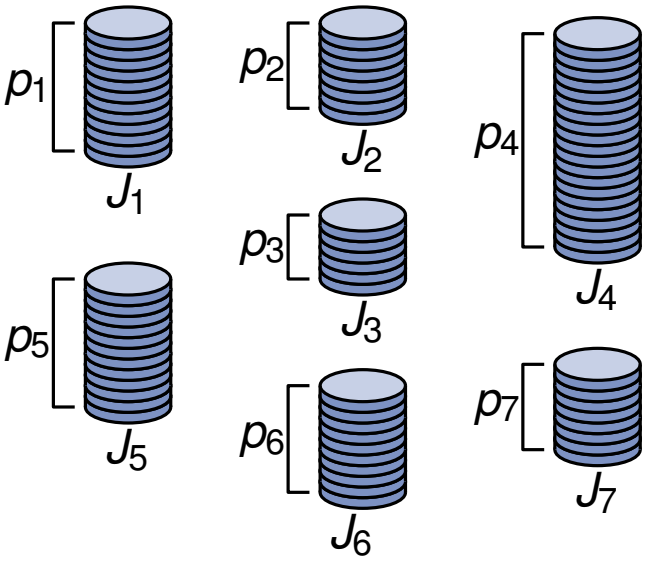

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

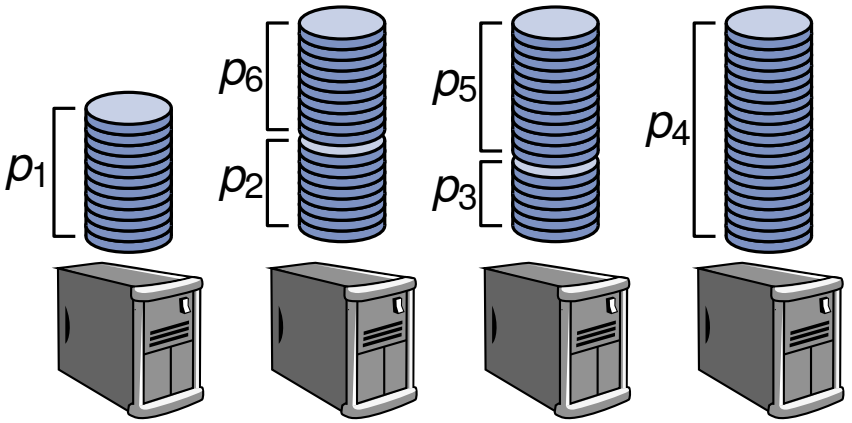
# Ein einfacher Algorithmus – LIST SCHEDULING

LIST SCHEDULING($J_1, \ldots, J_n, m$)

Lege die ersten $m$ Jobs auf die $m$ Maschinen.

Sobald eine Maschine frei ist, ordne ihr den nächsten der restlichen Jobs zu.





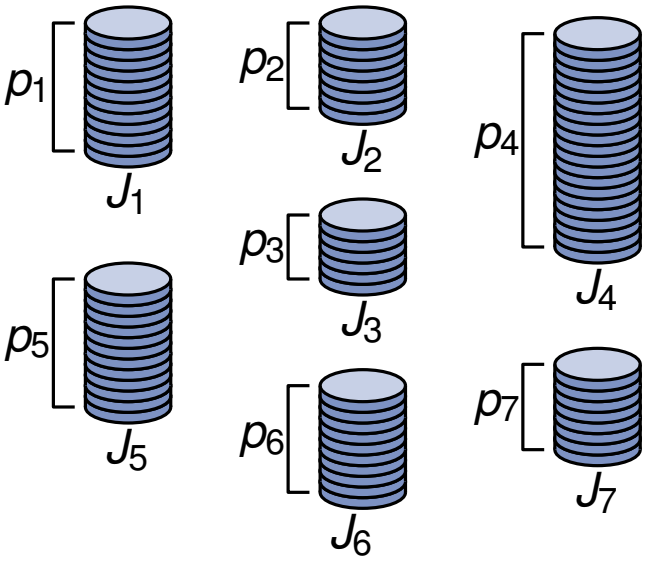

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

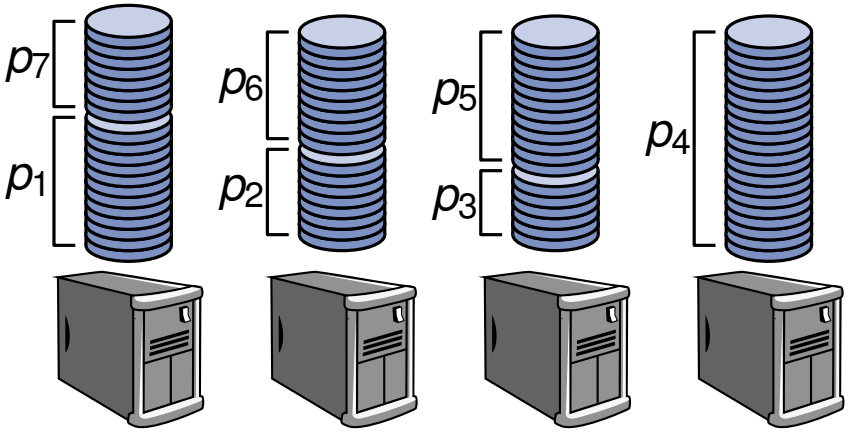
# Ein einfacher Algorithmus – LIST SCHEDULING

LIST SCHEDULING($J_1, \ldots, J_n, m$)

Lege die ersten $m$ Jobs auf die $m$ Maschinen.

Sobald eine Maschine frei ist, ordne ihr den nächsten der restlichen Jobs zu.




LIST SCHEDULING hat offensichtlich Laufzeit $O(n)$.

**Satz: Approximation** **(Satz 7.12)**

Sei $\mathcal{A}$ der Algorithmus LIST SCHEDULING. Dann gilt $\mathcal{R}_{\mathcal{A}} = 2 - \frac{1}{m}$.

# Ein einfacher Algorithmus – LIST SCHEDULING

LIST SCHEDULING($J_1, \ldots, J_n, m$)

Lege die ersten $m$ Jobs auf die $m$ Maschinen

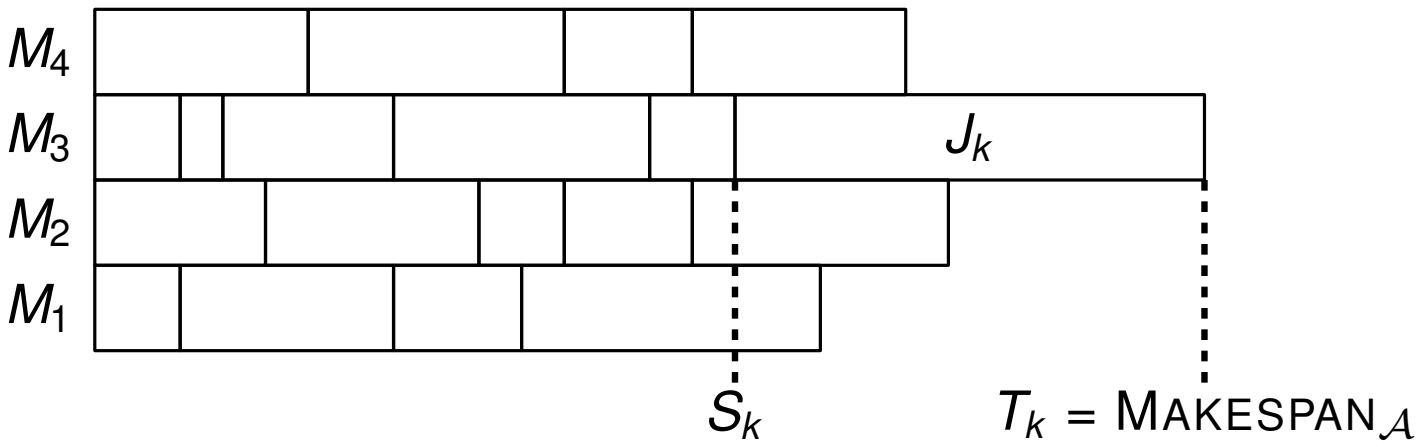
Sobald eine Maschine frei ist, ordne ihr den nächsten der restlichen $n - m$ Jobs zu

**Satz: Approximation** **(Satz 7.12)**

Sei $\mathcal{A}$ der Algorithmus LIST SCHEDULING. Dann gilt $\mathcal{R}_{\mathcal{A}} = 2 - \frac{1}{m}$.

**Beweisidee:**

- Sei $S_i$ die Startzeit und $T_i$ die Abschlusszeit von Job $J_i$.
- Falls $J_k$ der zuletzt beendete Job ist, so ist $T_k = \text{MAKESPAN}_{\mathcal{A}}$.
- Keine Maschine ist zu einem Zeitpunkt vor $S_k$ untätig.



# Ein PAS für Multiprozessor-Scheduling

Für eine Konstante $\ell$ ($1 \leq \ell \leq n$) definiere den Algorithmus $\mathcal{A}_\ell$ wie folgt.

$\mathcal{A}_\ell(J_1, \ldots, J_n, m)$

Sortiere die Jobs absteigend entsprechend ihrer Laufzeit.

Bestimme optimalen Schedule für die $\ell$ größten Jobs $J_1, \ldots, J_\ell$.

Ordne die restlichen Jobs $J_{\ell+1}, \ldots, J_n$ den Maschinen gemäß LIST SCHEDULING zu.

# Ein PAS für Multiprozessor-Scheduling

Für eine Konstante $\ell$ ($1 \leq \ell \leq n$) definiere den Algorithmus $\mathcal{A}_\ell$ wie folgt.

$\mathcal{A}_\ell(J_1, \ldots, J_n, m)$ — polynomiell für konstantes $\ell \rightarrow$ $O(m^\ell + n \log n)$

- Sortiere die Jobs absteigend entsprechend ihrer Laufzeit. — $O(n \log n)$
- Bestimme optimalen Schedule für die $\ell$ größten Jobs $J_1, \ldots, J_\ell$. — $O(m^\ell)$
- Ordne die restlichen Jobs $J_{\ell+1}, \ldots, J_n$ den Maschinen gemäß LIST SCHEDULING zu.

# Ein PAS für Multiprozessor-Scheduling

Für eine Konstante $\ell$ ($1 \leq \ell \leq n$) definiere den Algorithmus $\mathcal{A}_\ell$ wie folgt.

$\mathcal{A}_\ell(J_1, \ldots, J_n, m)$ — polynomiell für konstantes $\ell \rightarrow$ $O(m^\ell + n \log n)$

- Sortiere die Jobs absteigend entsprechend ihrer Laufzeit. — $O(n \log n)$
- Bestimme optimalen Schedule für die $\ell$ größten Jobs $J_1, \ldots, J_\ell$. — $O(m^\ell)$
- Ordne die restlichen Jobs $J_{\ell+1}, \ldots, J_n$ den Maschinen gemäß LIST SCHEDULING zu.

**Satz: Approximation** **(Satz 7.13)**

Für die Approximationsrate von $\mathcal{A}_\ell$ mit $1 \leq \ell \leq n$ konstant gilt:

$$\mathcal{R}_{\mathcal{A}_\ell} \leq 1 + \frac{1 - \frac{1}{m}}{1 + \left\lfloor \frac{\ell}{m} \right\rfloor}$$

# Ein PAS für Multiprozessor-Scheduling

Für eine Konstante $\ell$ ($1 \leq \ell \leq n$) definiere den Algorithmus $\mathcal{A}_\ell$ wie folgt.

$\mathcal{A}_\ell(J_1, \ldots, J_n, m)$ — polynomiell für konstantes $\ell \rightarrow$ $O(m^\ell + n \log n)$

- Sortiere die Jobs absteigend entsprechend ihrer Laufzeit. — $O(n \log n)$
- Bestimme optimalen Schedule für die $\ell$ größten Jobs $J_1, \ldots, J_\ell$. — $O(m^\ell)$
- Ordne die restlichen Jobs $J_{\ell+1}, \ldots, J_n$ den Maschinen gemäß LIST SCHEDULING zu.

**Satz: Approximation** **(Satz 7.13)**

Für die Approximationsrate von $\mathcal{A}_\ell$ mit $1 \leq \ell \leq n$ konstant gilt:

$$\mathcal{R}_{\mathcal{A}_\ell} \leq 1 + \frac{1 - \frac{1}{m}}{1 + \left\lfloor \frac{\ell}{m} \right\rfloor}$$

Zu $\varepsilon > 0$ sei $\mathcal{A}_\varepsilon$ ein Algorithmus $\mathcal{A}_\ell$ mit $\ell$ so gewählt, dass $\mathcal{R}_{\mathcal{A}_\ell} \leq 1 + \varepsilon$.

**Folgerung: PAS** **(Folgerung 7.14)**

Für konstant viele Maschinen ist $\{\mathcal{A}_\varepsilon \mid \varepsilon > 0\}$ ein PAS.

$\{\mathcal{A}_\varepsilon \mid \varepsilon > 0\}$ ist kein FPAS, da die Laufzeit nicht polynomiell in $\frac{1}{\varepsilon}$ ist.

Karlsruher Institut für Technologie

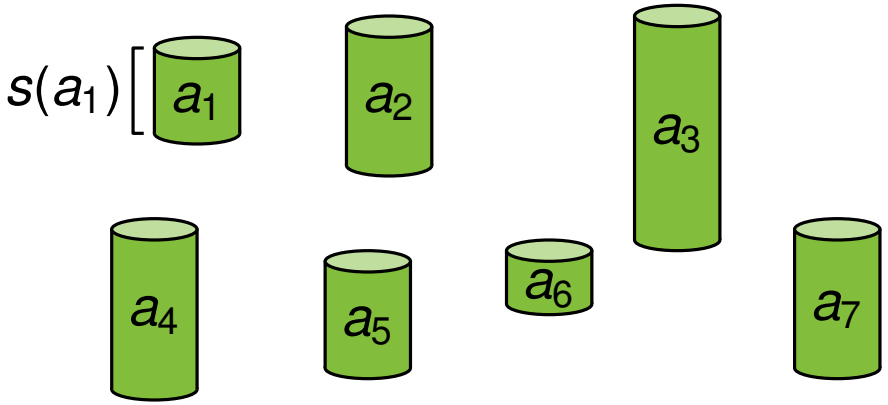
# Bin Packing

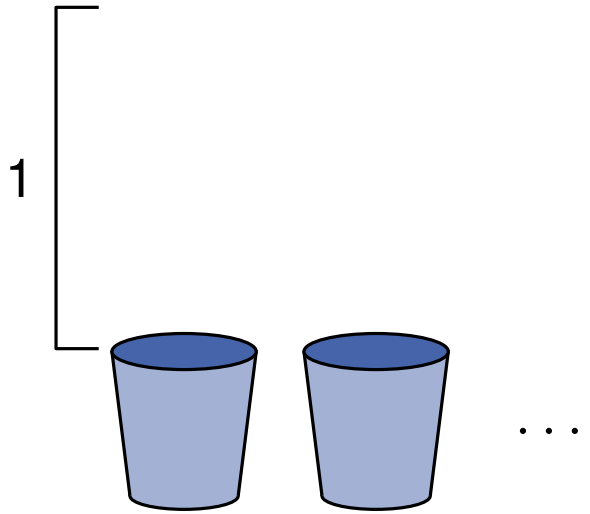


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Bin Packing – Definition

endliche Menge $M = \{a_1, \ldots, a_n\}$
mit Gewichtsfunktion $s\colon M \longrightarrow (0, 1]$




Eimer (Bins) mit Fassungsvermögen 1

**Problem: BIN PACKING**
Weise die Elemente in $M$ einer minimalen Anzahl an Bins $B_1, \ldots, B_m$ zu, sodass für jeden Bin $B$ gilt:

$$\sum_{a_i \in B} s(a_i) \leq 1$$

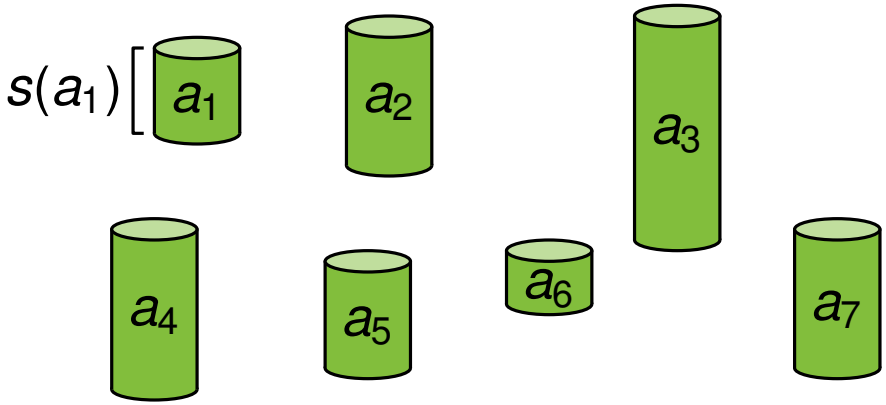
BIN PACKING ist $\mathcal{NP}$-schwer.

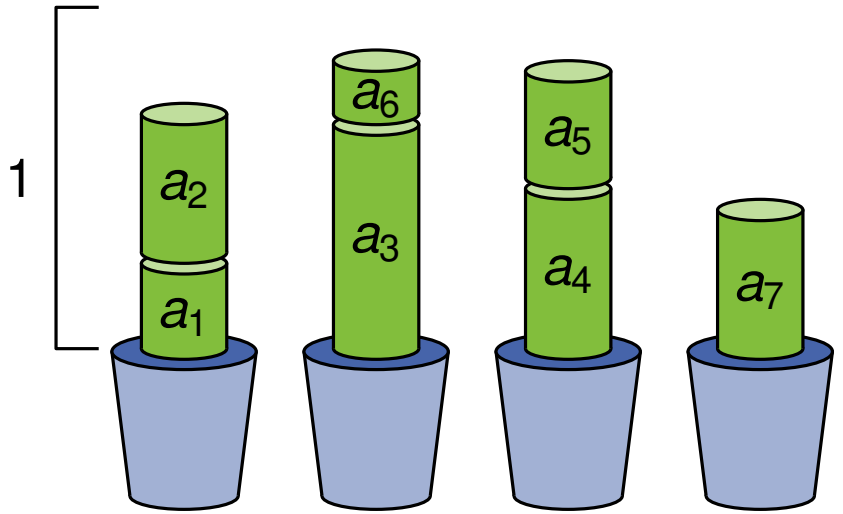


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Bin Packing – Definition

endliche Menge $M = \{a_1, \ldots, a_n\}$
mit Gewichtsfunktion $s\colon M \longrightarrow (0, 1]$




Eimer (Bins) mit Fassungsvermögen 1

4 Bins

**Problem: BIN PACKING**
Weise die Elemente in $M$ einer minimalen Anzahl an Bins $B_1, \ldots, B_m$ zu, sodass für jeden Bin $B$ gilt:

$$\sum_{a_i \in B} s(a_i) \leq 1$$

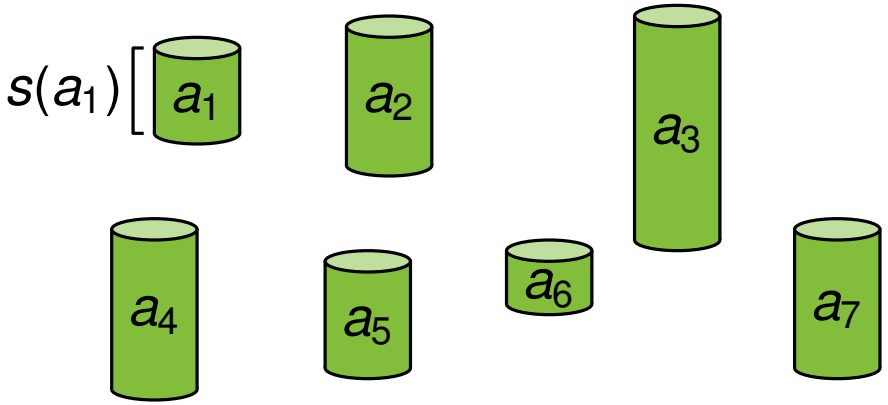
BIN PACKING ist $\mathcal{NP}$-schwer.

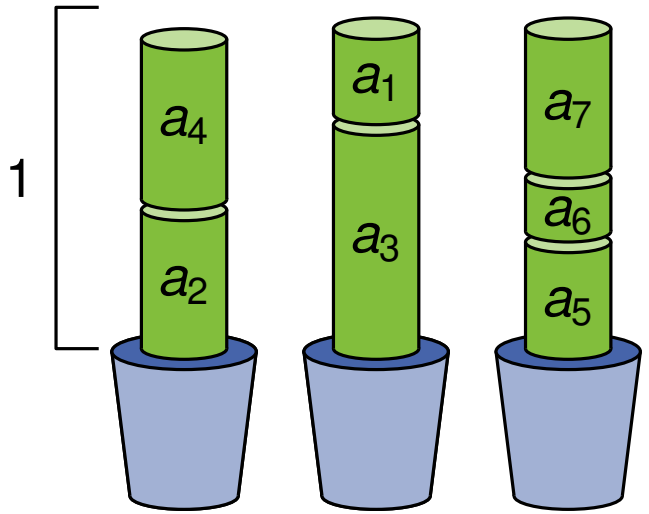


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Bin Packing – Definition

endliche Menge $M = \{a_1, \ldots, a_n\}$
mit Gewichtsfunktion $s\colon M \longrightarrow (0, 1]$




Eimer (Bins) mit Fassungsvermögen 1

3 Bins

**Problem: BIN PACKING**
Weise die Elemente in $M$ einer minimalen Anzahl an Bins $B_1, \ldots, B_m$ zu, sodass für jeden Bin $B$ gilt:

$$\sum_{a_i \in B} s(a_i) \leq 1$$

BIN PACKING ist $\mathcal{NP}$-schwer.



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Lösungsstrategie 1 – Next Fit

**Strategie:**

Füge Elemente nacheinander in den aktuellen Bin ein. Wenn ein Element nicht mehr passt, schließe den Bin ab und nimm einen neuen.

NEXT FIT (NF)$(M, s)$ — Laufzeit: $O(n)$

Füge $a_1$ in $B_1$ ein

**for** $a_\ell \in \{a_2, \ldots, a_n\}$ **do**

  $B_j \leftarrow$ letzter nicht-leerer Bin

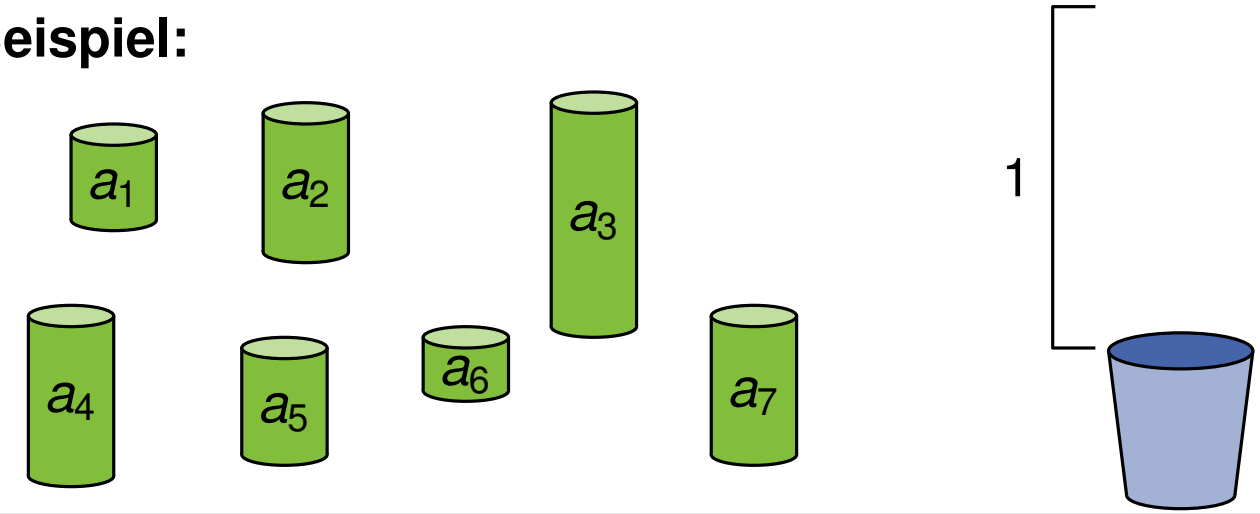
  **if** $s(a_\ell) \leq 1 - \sum_{a_i \in B_j} s(a_i)$ **then**

    Füge $a_\ell$ in $B_j$ ein

  **else**

    Füge $a_\ell$ in $B_{j+1}$ ein

**Beispiel:**



# Lösungsstrategie 1 – Next Fit

**Strategie:**

Füge Elemente nacheinander in den aktuellen Bin ein. Wenn ein Element nicht mehr passt, schließe den Bin ab und nimm einen neuen.

NEXT FIT (NF)$(M, s)$ — Laufzeit: $O(n)$

Füge $a_1$ in $B_1$ ein

**for** $a_\ell \in \{a_2, \ldots, a_n\}$ **do**

  $B_j \leftarrow$ letzter nicht-leerer Bin

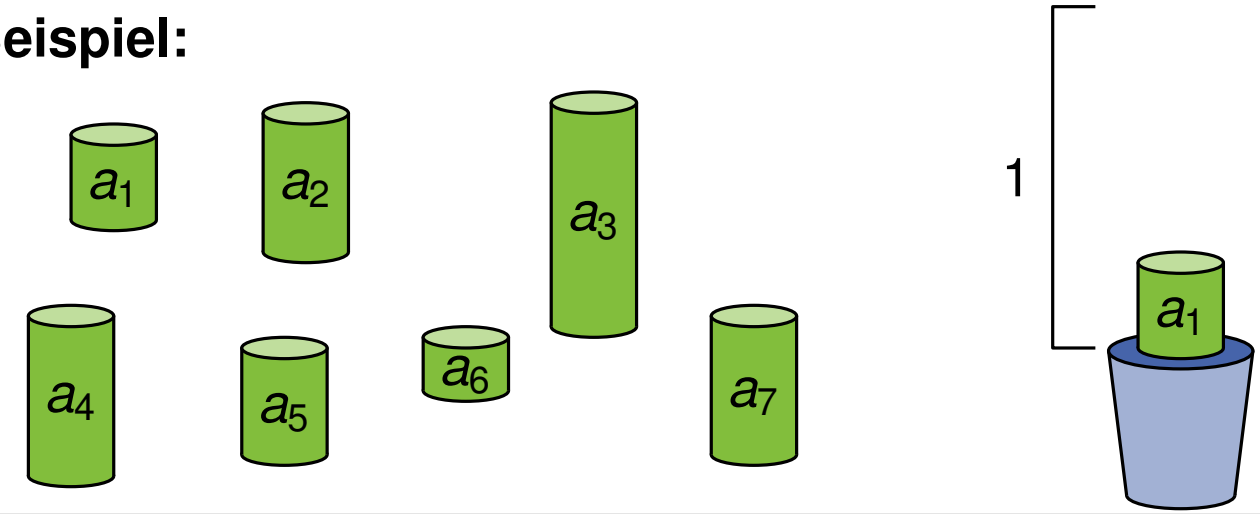
  **if** $s(a_\ell) \leq 1 - \sum_{a_i \in B_j} s(a_i)$ **then**

    Füge $a_\ell$ in $B_j$ ein

  **else**

    Füge $a_\ell$ in $B_{j+1}$ ein

**Beispiel:**






Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Lösungsstrategie 1 – Next Fit

**Strategie:**

Füge Elemente nacheinander in den aktuellen Bin ein. Wenn ein Element nicht mehr passt, schließe den Bin ab und nimm einen neuen.

NEXT FIT (NF)$(M, s)$ — Laufzeit: $O(n)$

Füge $a_1$ in $B_1$ ein

**for** $a_\ell \in \{a_2, \ldots, a_n\}$ **do**

  $B_j \leftarrow$ letzter nicht-leerer Bin

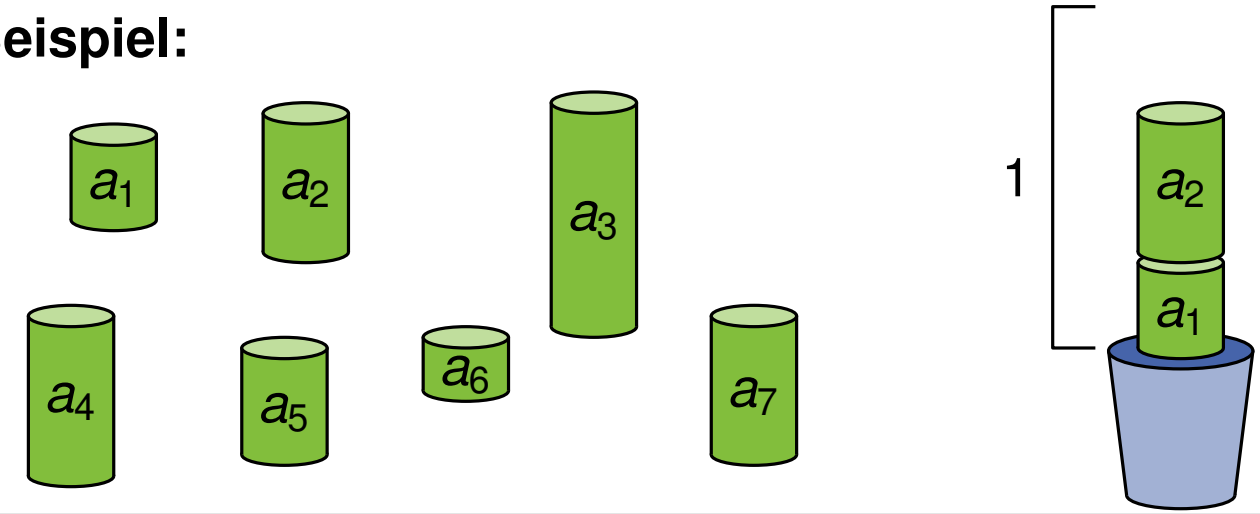
  **if** $s(a_\ell) \leq 1 - \sum_{a_i \in B_j} s(a_i)$ **then**

    Füge $a_\ell$ in $B_j$ ein

  **else**

    Füge $a_\ell$ in $B_{j+1}$ ein

**Beispiel:**






Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Lösungsstrategie 1 – Next Fit

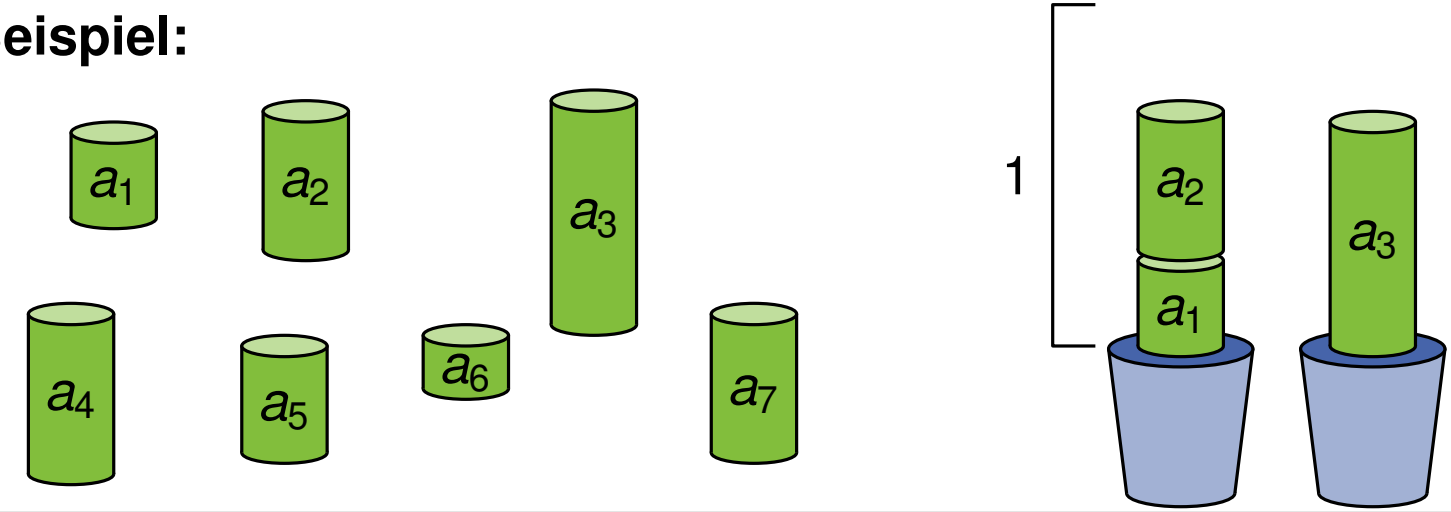
**Strategie:**

Füge Elemente nacheinander in den aktuellen Bin ein. Wenn ein Element nicht mehr passt, schließe den Bin ab und nimm einen neuen.

NEXT FIT (NF)$(M, s)$ — Laufzeit: $O(n)$

```
Füge a_1 in B_1 ein
for a_ℓ ∈ {a_2, ..., a_n} do
    B_j ← letzter nicht-leerer Bin
    if s(a_ℓ) ≤ 1 − Σ_{a_i ∈ B_j} s(a_i) then
        Füge a_ℓ in B_j ein
    else
        Füge a_ℓ in B_{j+1} ein
```

**Beispiel:**






Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Lösungsstrategie 1 – Next Fit

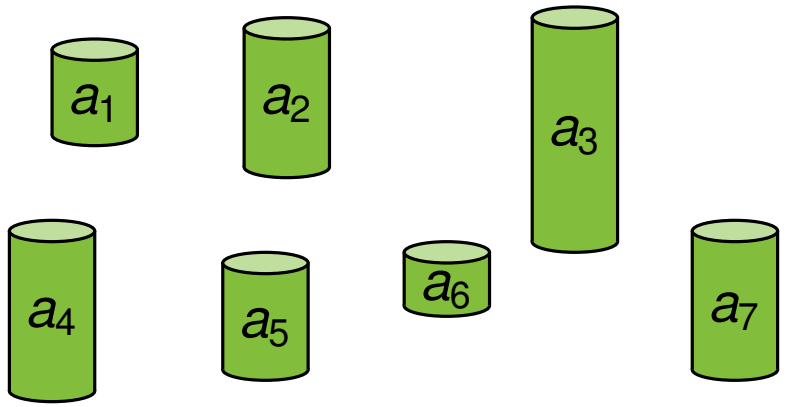
**Strategie:**

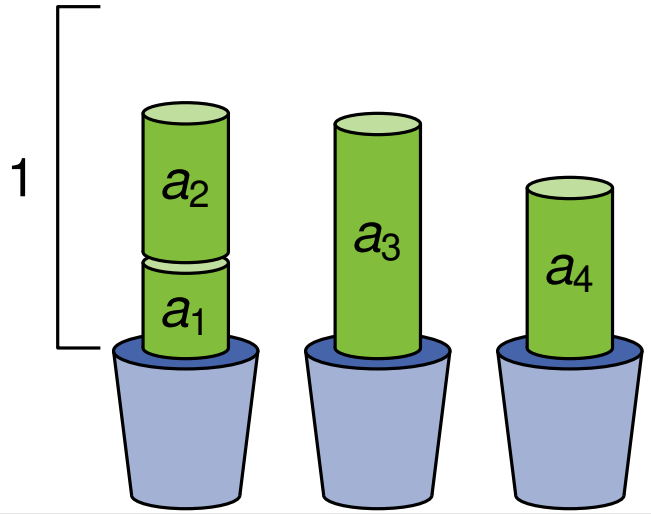
Füge Elemente nacheinander in den aktuellen Bin ein. Wenn ein Element nicht mehr passt, schließe den Bin ab und nimm einen neuen.

NEXT FIT (NF)$(M, s)$ Laufzeit: $O(n)$

```
Füge a_1 in B_1 ein
for a_ℓ ∈ {a_2, ..., a_n} do
    B_j ← letzter nicht-leerer Bin
    if s(a_ℓ) ≤ 1 − Σ_{a_i ∈ B_j} s(a_i) then
        Füge a_ℓ in B_j ein
    else
        Füge a_ℓ in B_{j+1} ein
```

**Beispiel:**






Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Lösungsstrategie 1 – Next Fit

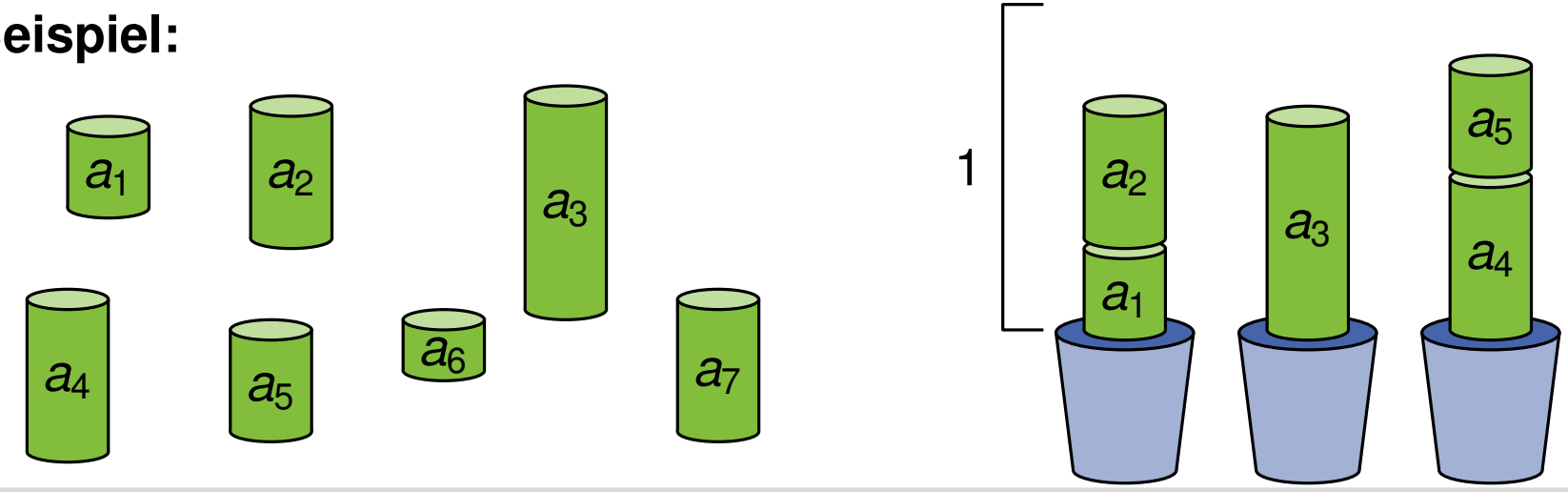
**Strategie:**

Füge Elemente nacheinander in den aktuellen Bin ein. Wenn ein Element nicht mehr passt, schließe den Bin ab und nimm einen neuen.

NEXT FIT (NF)$(M, s)$ Laufzeit: $O(n)$

```
Füge a_1 in B_1 ein
for a_ℓ ∈ {a_2, ..., a_n} do
    B_j ← letzter nicht-leerer Bin
    if s(a_ℓ) ≤ 1 − Σ_{a_i ∈ B_j} s(a_i) then
        Füge a_ℓ in B_j ein
    else
        Füge a_ℓ in B_{j+1} ein
```

**Beispiel:**






Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Lösungsstrategie 1 – Next Fit

**Strategie:**

Füge Elemente nacheinander in den aktuellen Bin ein. Wenn ein Element nicht mehr passt, schließe den Bin ab und nimm einen neuen.

NEXT FIT (NF)$(M, s)$ Laufzeit: $O(n)$

Füge $a_1$ in $B_1$ ein

**for** $a_\ell \in \{a_2, \ldots, a_n\}$ **do**

  $B_j \leftarrow$ letzter nicht-leerer Bin

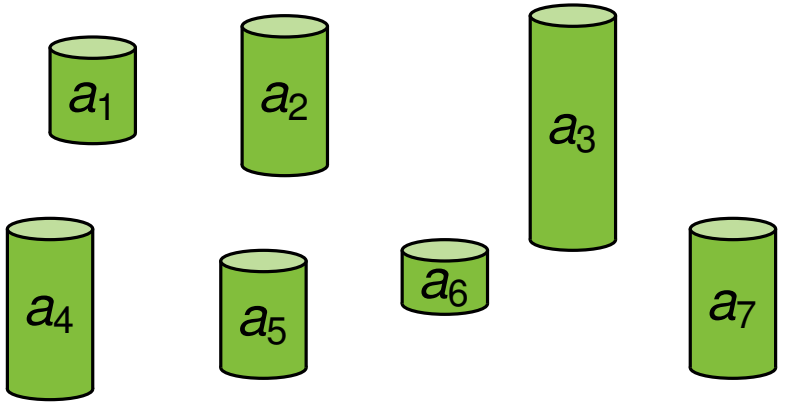
  **if** $s(a_\ell) \leq 1 - \sum_{a_i \in B_j} s(a_i)$ **then**

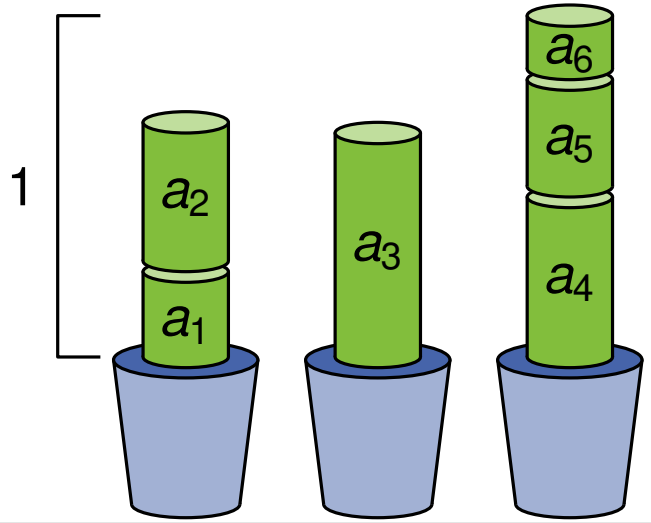
    Füge $a_\ell$ in $B_j$ ein

  **else**

    Füge $a_\ell$ in $B_{j+1}$ ein

**Beispiel:**






Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Lösungsstrategie 1 – Next Fit

**Strategie:**

Füge Elemente nacheinander in den aktuellen Bin ein. Wenn ein Element nicht mehr passt, schließe den Bin ab und nimm einen neuen.

NEXT FIT (NF)$(M, s)$ Laufzeit: $O(n)$

Füge $a_1$ in $B_1$ ein

**for** $a_\ell \in \{a_2, \ldots, a_n\}$ **do**

  $B_j \leftarrow$ letzter nicht-leerer Bin

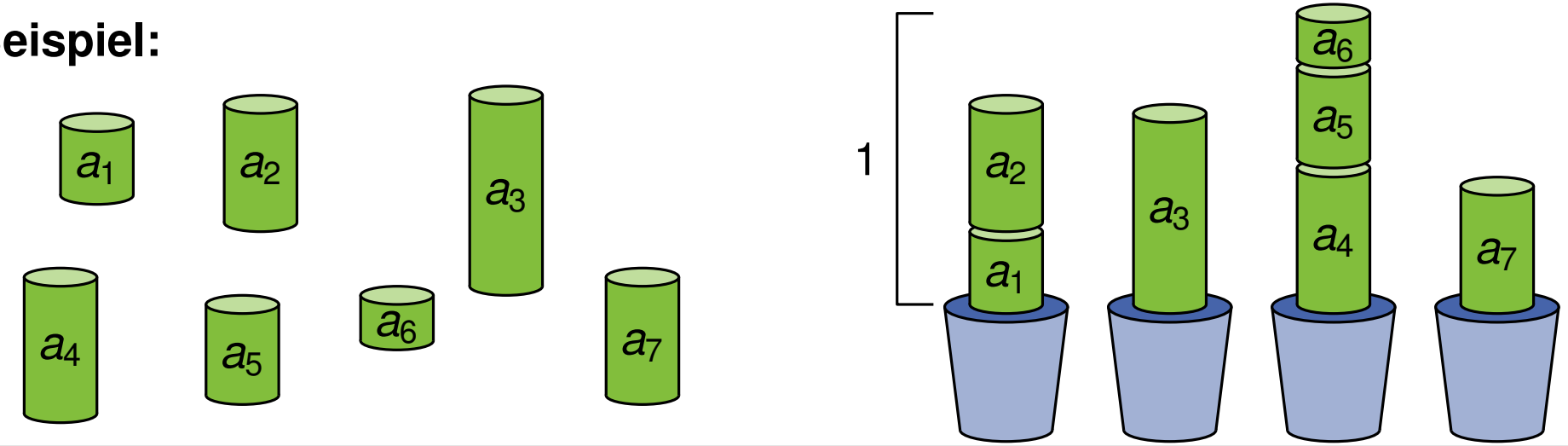
  **if** $s(a_\ell) \leq 1 - \sum_{a_i \in B_j} s(a_i)$ **then**

    Füge $a_\ell$ in $B_j$ ein

  **else**

    Füge $a_\ell$ in $B_{j+1}$ ein

**Beispiel:**






Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Next Fit – Approximation

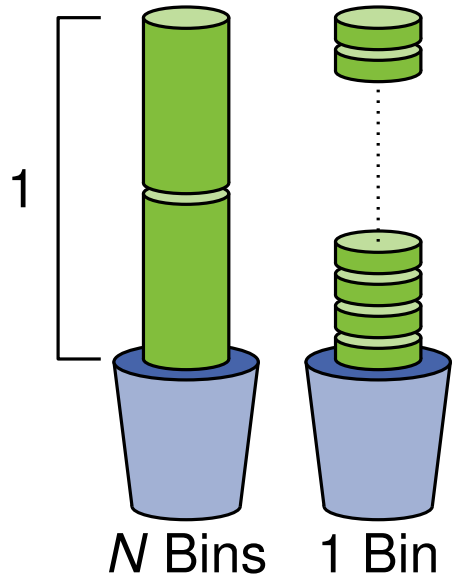
## Schlechtes Beispiel

**(Beispiel 7.4)**

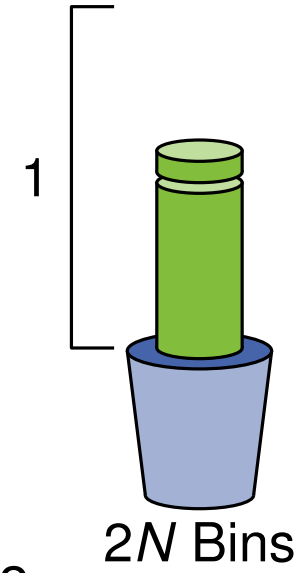
Betrachte die Menge $\{a_1, \ldots, a_n\}$ für $n = 4 \cdot N$ mit $N \in \mathbb{N}$. Sei weiterhin

$$s(a_i) = \begin{cases} \frac{1}{2} & \text{falls } i \text{ ungerade} \\ \frac{1}{2 \cdot N} & \text{sonst} \end{cases}$$

**Optimale Lösung OPT($I$)**




**Lösung von NEXT FIT NF($I$)**




$$\Rightarrow \text{NF}(I) = 2 \cdot \text{OPT}(I) - 2$$

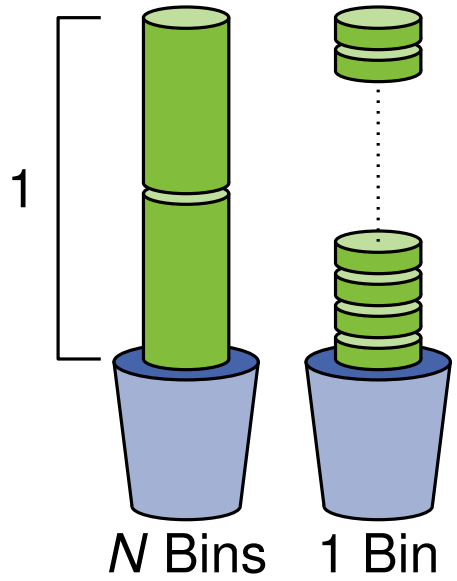
# Next Fit – Approximation

**Schlechtes Beispiel** **(Beispiel 7.4)**

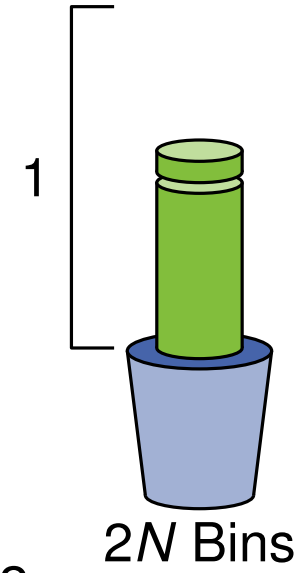
Betrachte die Menge $\{a_1, \ldots, a_n\}$ für $n = 4 \cdot N$ mit $N \in \mathbb{N}$. Sei weiterhin

$$s(a_i) = \begin{cases} \frac{1}{2} & \text{falls } i \text{ ungerade} \\ \frac{1}{2 \cdot N} & \text{sonst} \end{cases}$$

**Optimale Lösung OPT($I$)**




**Lösung von NEXT FIT NF($I$)**




$$\Rightarrow \text{NF}(I) = 2 \cdot \text{OPT}(I) - 2$$

**Satz: Approximation** **(Satz 7.5)**

NEXT FIT erfüllt $\mathcal{R}_{\text{NF}} = 2$.



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Lösungsstrategie 2 – First Fit

**Strategie:**

Füge aktuelles Element immer in den ersten Bin ein, in den es passt.

FIRST FIT (FF)$(M, s)$ — Laufzeit: $O(n^2)$

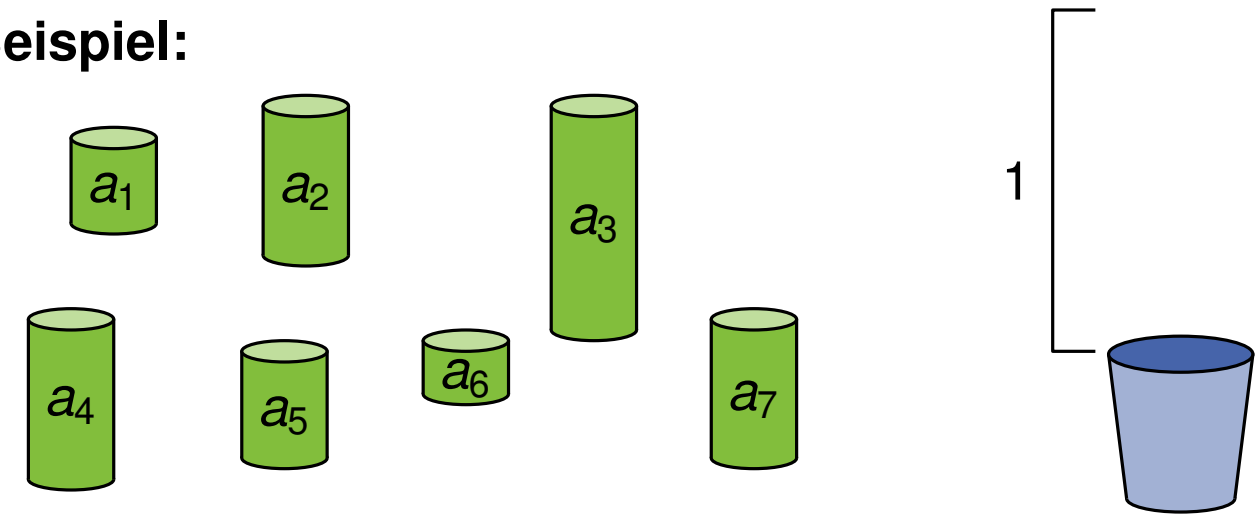
Füge $a_1$ in $B_1$ ein

**for** $a_\ell \in \{a_2, \ldots, a_n\}$ **do**

$B_j \leftarrow$ erster Bin mit $s(a_\ell) \leq 1 - \sum_{a_i \in B_j} s(a_i)$

Füge $a_\ell$ in $B_j$ ein

**Beispiel:**



# Lösungsstrategie 2 – First Fit

**Strategie:**

Füge aktuelles Element immer in den ersten Bin ein, in den es passt.

FIRST FIT (FF)$(M, s)$ — Laufzeit: $O(n^2)$

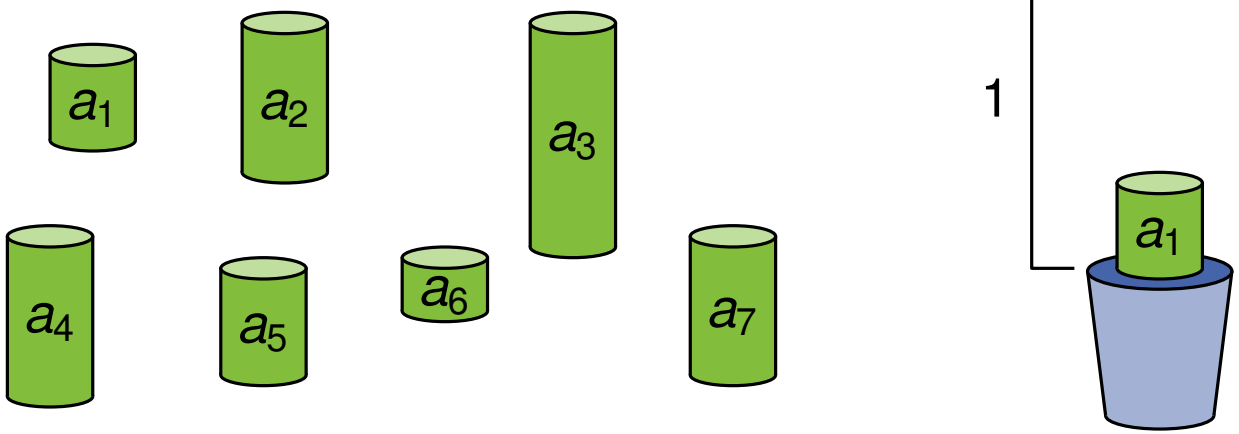
Füge $a_1$ in $B_1$ ein

**for** $a_\ell \in \{a_2, \ldots, a_n\}$ **do**

$B_j \leftarrow$ erster Bin mit $s(a_\ell) \leq 1 - \sum_{a_i \in B_j} s(a_i)$

Füge $a_\ell$ in $B_j$ ein

**Beispiel:**



# Lösungsstrategie 2 – First Fit

**Strategie:**

Füge aktuelles Element immer in den ersten Bin ein, in den es passt.

FIRST FIT (FF)$(M, s)$ — Laufzeit: $O(n^2)$

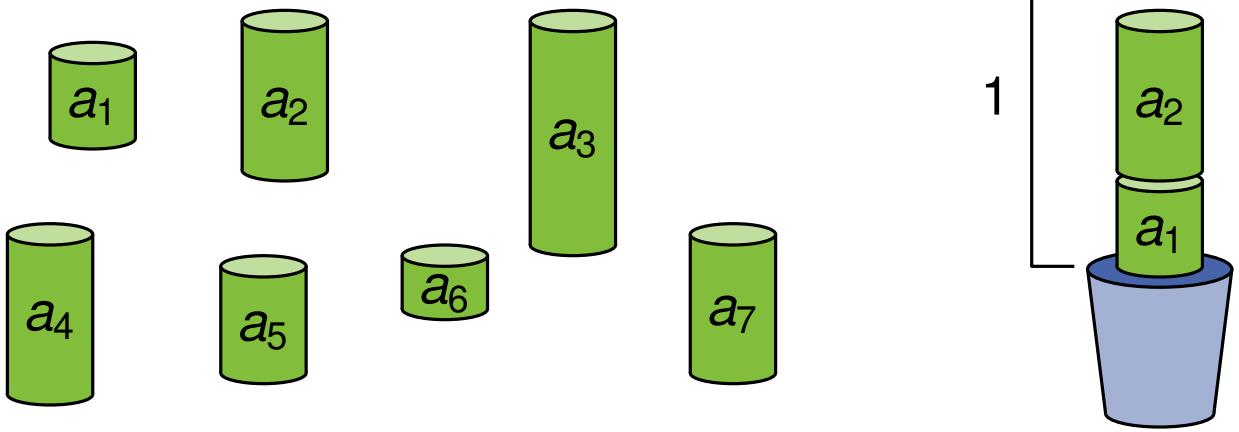
Füge $a_1$ in $B_1$ ein

**for** $a_\ell \in \{a_2, \ldots, a_n\}$ **do**

$B_j \leftarrow$ erster Bin mit $s(a_\ell) \leq 1 - \sum_{a_i \in B_j} s(a_i)$

Füge $a_\ell$ in $B_j$ ein

**Beispiel:**






Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Lösungsstrategie 2 – First Fit

**Strategie:**

Füge aktuelles Element immer in den ersten Bin ein, in den es passt.

FIRST FIT (FF)$(M, s)$ — Laufzeit: $O(n^2)$

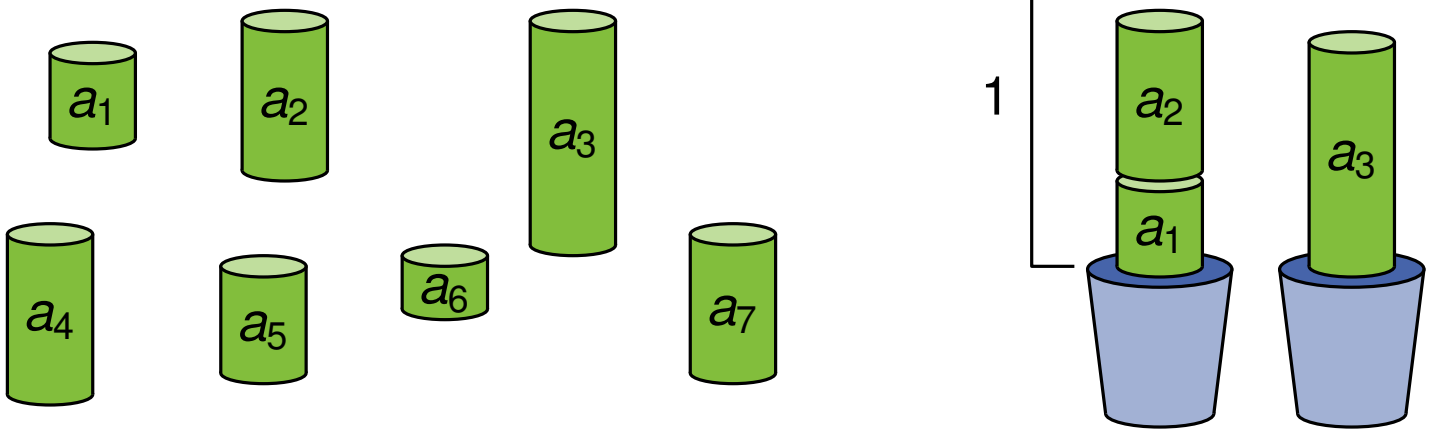
Füge $a_1$ in $B_1$ ein

**for** $a_\ell \in \{a_2, \ldots, a_n\}$ **do**

$B_j \leftarrow$ erster Bin mit $s(a_\ell) \leq 1 - \sum_{a_i \in B_j} s(a_i)$

Füge $a_\ell$ in $B_j$ ein

**Beispiel:**






Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# Lösungsstrategie 2 – First Fit

**Strategie:**

Füge aktuelles Element immer in den ersten Bin ein, in den es passt.

FIRST FIT (FF)$(M, s)$ — Laufzeit: $O(n^2)$

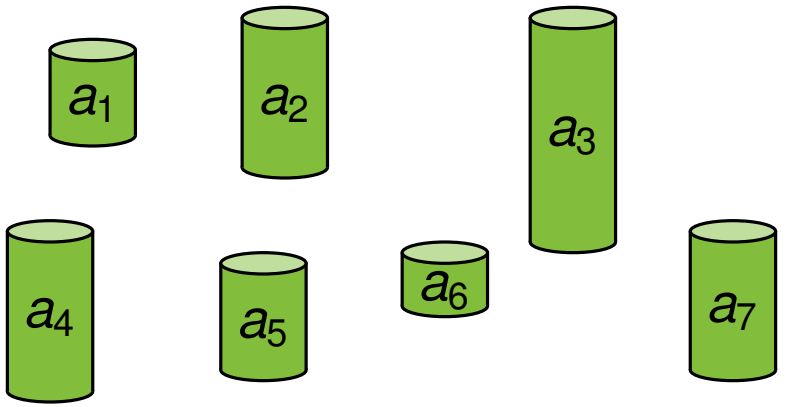
Füge $a_1$ in $B_1$ ein

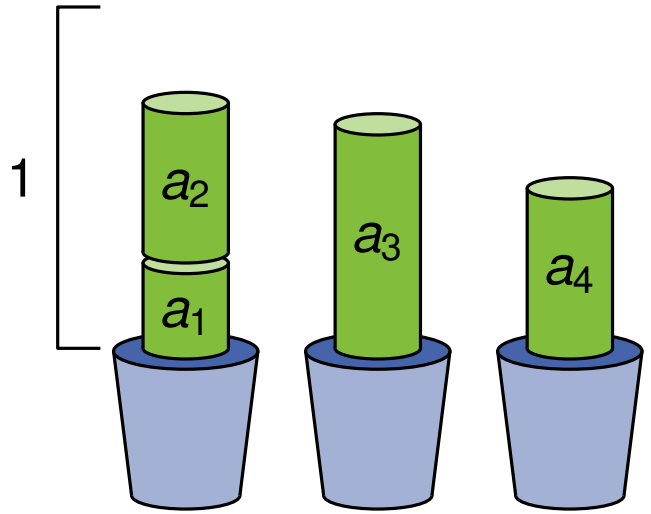
**for** $a_\ell \in \{a_2, \ldots, a_n\}$ **do**

$B_j \leftarrow$ erster Bin mit $s(a_\ell) \leq 1 - \sum_{a_i \in B_j} s(a_i)$

Füge $a_\ell$ in $B_j$ ein

**Beispiel:**



# Lösungsstrategie 2 – First Fit

**Strategie:**

Füge aktuelles Element immer in den ersten Bin ein, in den es passt.

FIRST FIT (FF)$(M, s)$ — Laufzeit: $O(n^2)$

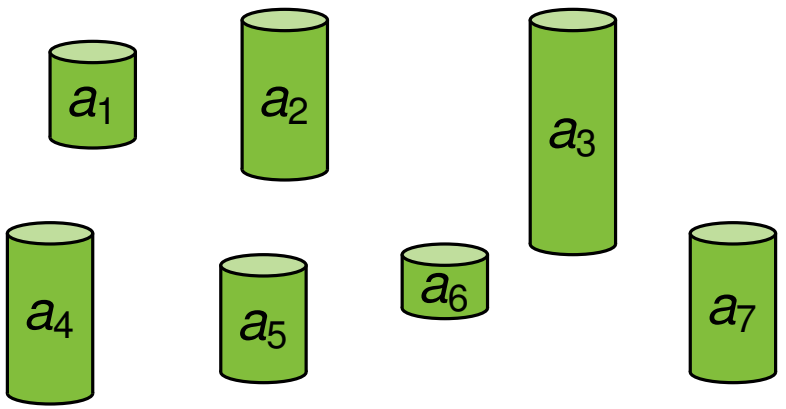
Füge $a_1$ in $B_1$ ein

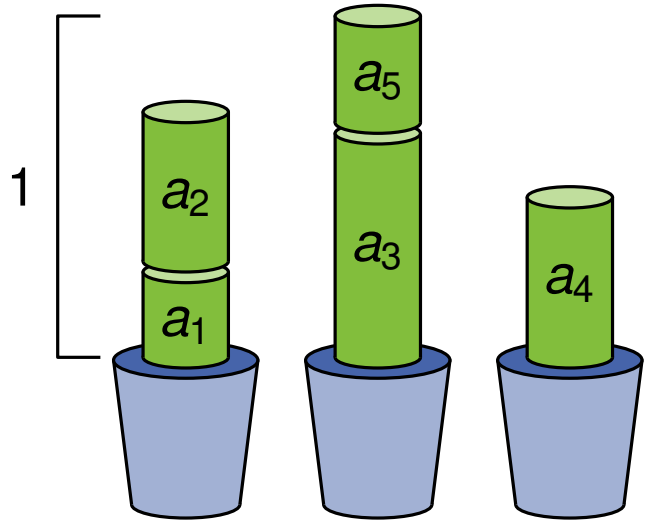
**for** $a_\ell \in \{a_2, \ldots, a_n\}$ **do**

$B_j \leftarrow$ erster Bin mit $s(a_\ell) \leq 1 - \sum_{a_i \in B_j} s(a_i)$

Füge $a_\ell$ in $B_j$ ein

**Beispiel:**



# Lösungsstrategie 2 – First Fit

**Strategie:**

Füge aktuelles Element immer in den ersten Bin ein, in den es passt.

FIRST FIT (FF)$(M, s)$ — Laufzeit: $O(n^2)$

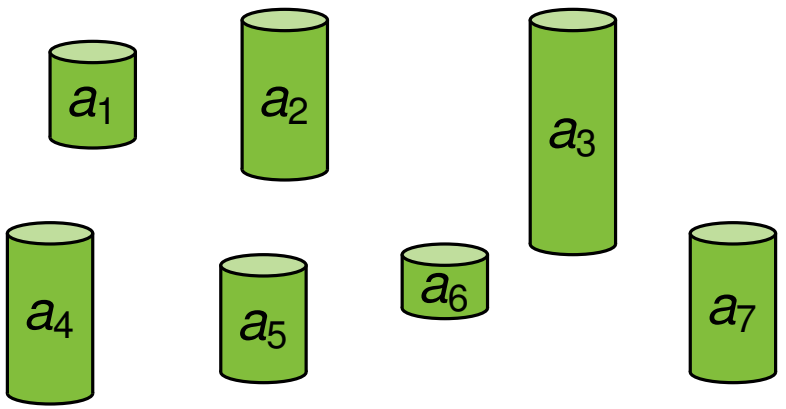
Füge $a_1$ in $B_1$ ein

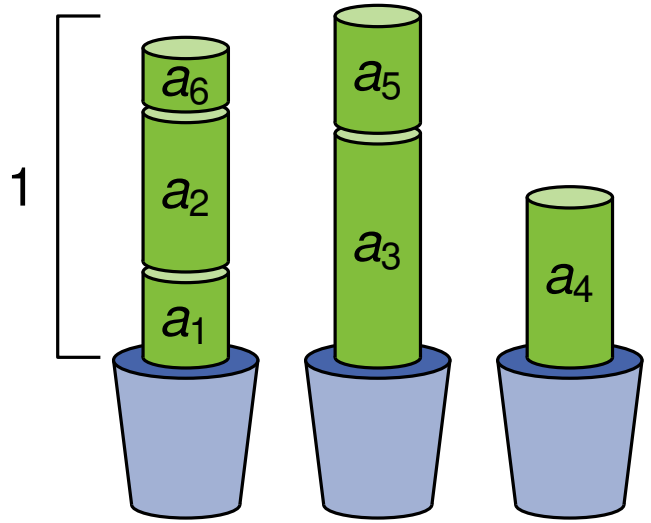
**for** $a_\ell \in \{a_2, \ldots, a_n\}$ **do**

$B_j \leftarrow$ erster Bin mit $s(a_\ell) \leq 1 - \sum_{a_i \in B_j} s(a_i)$

Füge $a_\ell$ in $B_j$ ein

**Beispiel:**



# Lösungsstrategie 2 – First Fit

**Strategie:**

Füge aktuelles Element immer in den ersten Bin ein, in den es passt.

FIRST FIT (FF)$(M, s)$ — Laufzeit: $O(n^2)$

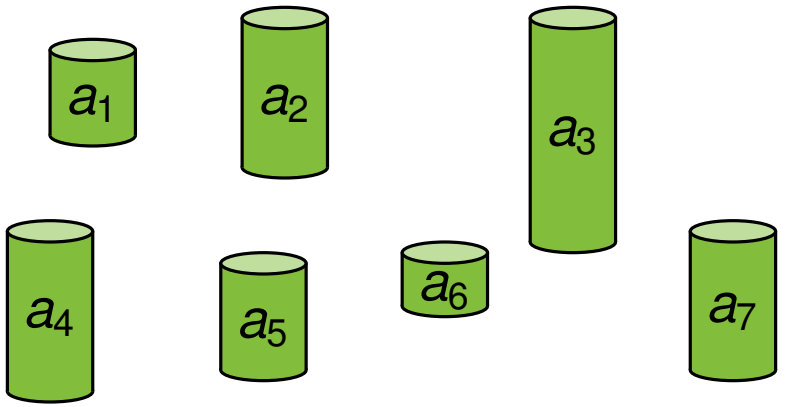
Füge $a_1$ in $B_1$ ein

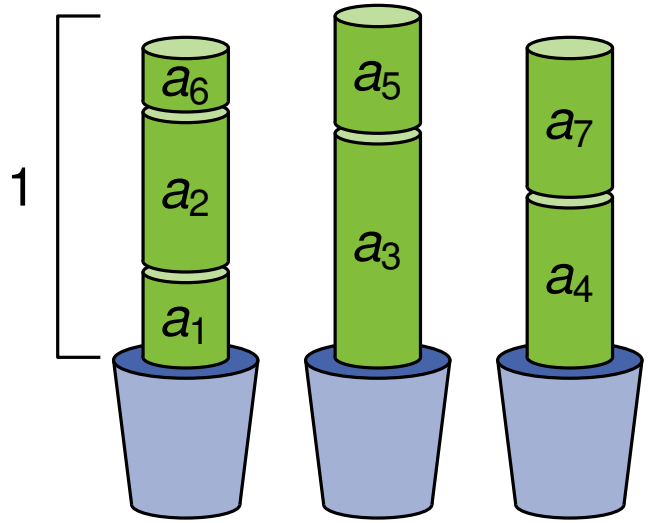
**for** $a_\ell \in \{a_2, \ldots, a_n\}$ **do**

$B_j \leftarrow$ erster Bin mit $s(a_\ell) \leq 1 - \sum_{a_i \in B_j} s(a_i)$

Füge $a_\ell$ in $B_j$ ein

**Beispiel:**




Für jede Instanz $I$ gilt $FF(I) < 2 \cdot \text{OPT}(I)$

Kann man das besser abschätzen?



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

# First Fit – Approximation

**Schlechtes Beispiel** **(Beispiel 7.7)**

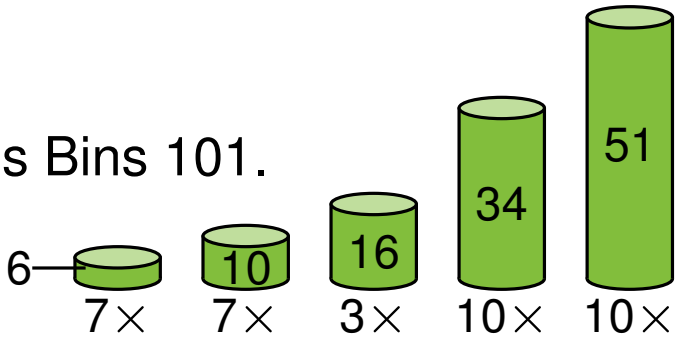
Sei das Fassungsvermögen eines Bins 101 (statt 1) und sei $n = 37$. Sei weiterhin

$$s(a_i) = \begin{cases} 6 & \text{für } 1 \leq i \leq 7 \\ 10 & \text{für } 8 \leq i \leq 14 \\ 16 & \text{für } 15 \leq i \leq 17 \\ 34 & \text{für } 18 \leq i \leq 27 \\ 51 & \text{für } 28 \leq i \leq 37 \end{cases}$$

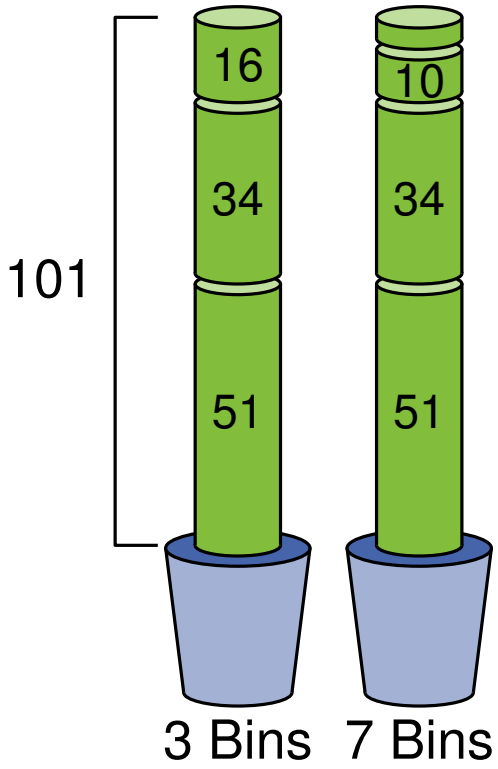
# First Fit – Approximation

**Schlechtes Beispiel** **(Beispiel 7.7)**

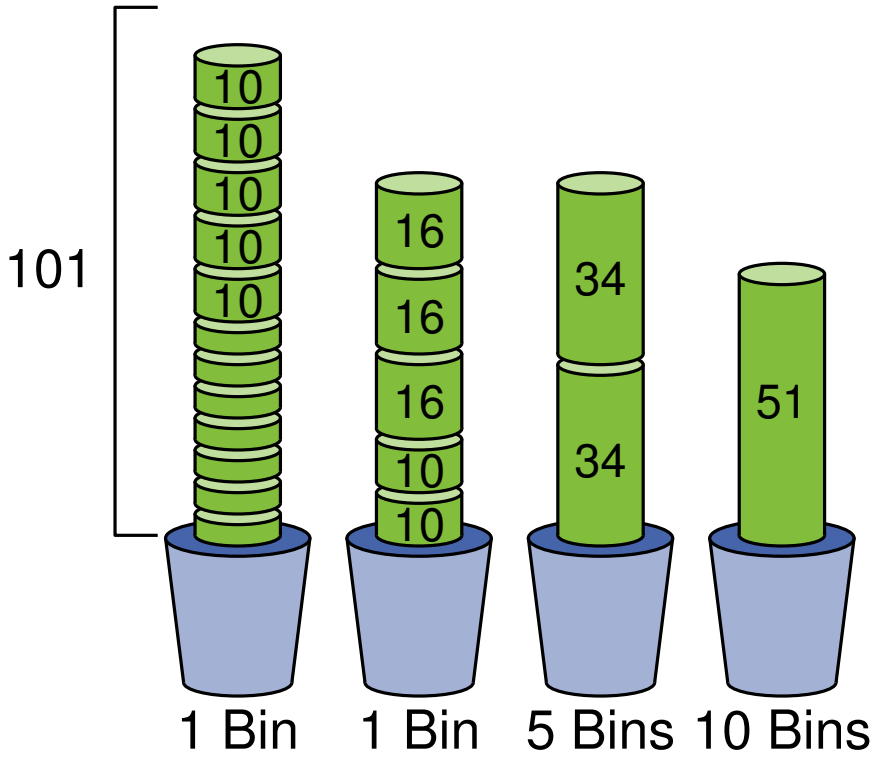
Sei das Fassungsvermögen eines Bins 101.
Betrachte folgende Elemente:




**Optimale Lösung OPT(*I*)**




**Lösung von FIRST FIT FF(*I*)**




$$\Rightarrow \mathrm{FF}(I) = 17 \text{ und } \mathrm{OPT}(I) = 10 \Rightarrow \mathcal{R}_{\mathrm{FF}}(I) = \frac{17}{10}$$

# First Fit – Approximation

**Gerade gesehen:** Es gibt eine Instanz $I$, sodass $\mathcal{R}_{\mathrm{FF}}(I) = \frac{17}{10}$.

Umgekehrt kann es kein viel schlechteres Beispiel geben:

**Satz: Approximation** **(Satz 7.8)**

Für jede Instanz $I$ von BIN PACKING gilt: $\mathrm{FF}(I) < \frac{17}{10} \cdot \mathrm{OPT}(I) + 1$

# First Fit – Approximation

**Gerade gesehen:** Es gibt eine Instanz $I$, sodass $\mathcal{R}_{\text{FF}}(I) = \frac{17}{10}$.

Umgekehrt kann es kein viel schlechteres Beispiel geben:

**Satz: Approximation** **(Satz 7.8)**

Für jede Instanz $I$ von BIN PACKING gilt: $\text{FF}(I) < \frac{17}{10} \cdot \text{OPT}(I) + 1$

**Bemerkung: Asymptotische Approximation** **(Bemerkung 7.9)**

Definiere die *asymptotische Gütegarantie* $\mathcal{R}_{\mathcal{A}}^{\infty}$ wie folgt:

$$\mathcal{R}_{\mathcal{A}}^{\infty} := \inf \{ r \geq 1 \mid \text{Es gibt } N > 0, \text{sodass } \mathcal{R}_{\mathcal{A}}(I) \leq r \text{ für alle } I \text{ mit } \text{OPT}(I) \geq N \}$$

Dann ist $\mathcal{R}_{\text{FF}}^{\infty} = \frac{17}{10}$ (der Summand 1 ist vernachlässigbar).

# First Fit – Approximation

**Gerade gesehen:** Es gibt eine Instanz $I$, sodass $\mathcal{R}_{\text{FF}}(I) = \frac{17}{10}$.

Umgekehrt kann es kein viel schlechteres Beispiel geben:

**Satz: Approximation** **(Satz 7.8)**

Für jede Instanz $I$ von BIN PACKING gilt: $\text{FF}(I) < \frac{17}{10} \cdot \text{OPT}(I) + 1$

**Bemerkung: Asymptotische Approximation** **(Bemerkung 7.9)**

Definiere die *asymptotische Gütegarantie* $\mathcal{R}_{\mathcal{A}}^{\infty}$ wie folgt:

$$\mathcal{R}_{\mathcal{A}}^{\infty} := \inf \{ r \geq 1 \mid \text{Es gibt } N > 0, \text{sodass } \mathcal{R}_{\mathcal{A}}(I) \leq r \text{ für alle } I \text{ mit } \text{OPT}(I) \geq N \}$$

Dann ist $\mathcal{R}_{\text{FF}}^{\infty} = \frac{17}{10}$ (der Summand 1 ist vernachlässigbar).

Es gibt weitere Strategien mit teilweise besseren Approximationsgüten.

**Beispiel:** FIRST FIT DECREASING (FFD) sortiert die Elemente zunächst absteigend und fügt sie dann mittels FF ein. $\rightarrow$ asymptotische Gütegarantie: $\frac{11}{9}$