Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Chair for
INFORMATICS I
Efficient Algorithms and
Knowledge-Based Systems

Institute for Informatics

Approximationsalgorithmen

Approximationsschemata und das Rucksack Problem

8. Vorlesung

Philipp Kindermann

Wintersemester 2019/20

Approximationsschema

Sei Π ein Optimierungsproblem.

Approximationsschema

Sei Π ein Optimierungsproblem. Ein Algorithmus $\mathcal{A}$ heißt **Polynomielles Approximationsschema** (PTAS), wenn dieser für jede Eingabe (I, ε) mit $I \in D_\Pi$ und $\varepsilon > 0$ eine Lösung $s \in S_\Pi(I)$ ausgibt, die Folgendes erfüllt:

Approximationsschema

Sei Π ein Optimierungsproblem. Ein Algorithmus $\mathcal{A}$ heißt **Polynomielles Approximationsschema** (PTAS), wenn dieser für jede Eingabe (I, ε) mit $I \in D_\Pi$ und $\varepsilon > 0$ eine Lösung $s \in S_\Pi(I)$ ausgibt, die Folgendes erfüllt:

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$, falls Π Min-Problem,

Approximationsschema

Sei Π ein Optimierungsproblem. Ein Algorithmus $\mathcal{A}$ heißt **Polynomielles Approximationsschema** (PTAS), wenn dieser für jede Eingabe (I, ε) mit $I \in D_\Pi$ und $\varepsilon > 0$ eine Lösung $s \in S_\Pi(I)$ ausgibt, die Folgendes erfüllt:

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$, falls Π Min-Problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$, falls Π Max-Problem.

Approximationsschema

Sei Π ein Optimierungsproblem. Ein Algorithmus $\mathcal{A}$ heißt **Polynomielles Approximationsschema** (PTAS), wenn dieser für jede Eingabe (I, ε) mit $I \in D_\Pi$ und $\varepsilon > 0$ eine Lösung $s \in S_\Pi(I)$ ausgibt, die Folgendes erfüllt:

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$, falls Π Min-Problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$, falls Π Max-Problem.

Die Laufzeit von $\mathcal{A}$ ist polynom. in $|I|$ für **jedes** feste $\varepsilon > 0$.

Approximationsschema

Sei Π ein Optimierungsproblem. Ein Algorithmus $\mathcal{A}$ heißt **Polynomielles Approximationsschema** (PTAS), wenn dieser für jede Eingabe (I, ε) mit $I \in D_\Pi$ und $\varepsilon > 0$ eine Lösung $s \in S_\Pi(I)$ ausgibt, die Folgendes erfüllt:

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$, falls Π Min-Problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$, falls Π Max-Problem.

Die Laufzeit von $\mathcal{A}$ ist polynom. in $|I|$ für **jedes** feste $\varepsilon > 0$.

$\mathcal{A}$ heißt **vollpolynomielles Approximationsschema** (FPTAS), falls seine Laufzeit polynomiell in $|I|$ und $1/\varepsilon$ ist.

Approximationsschema

Sei Π ein Optimierungsproblem. Ein Algorithmus $\mathcal{A}$ heißt **Polynomielles Approximationsschema** (PTAS), wenn dieser für jede Eingabe (I, ε) mit $I \in D_\Pi$ und $\varepsilon > 0$ eine Lösung $s \in S_\Pi(I)$ ausgibt, die Folgendes erfüllt:

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$, falls Π Min-Problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$, falls Π Max-Problem.

Die Laufzeit von $\mathcal{A}$ ist polynom. in $|I|$ für **jedes** feste $\varepsilon > 0$.

$\mathcal{A}$ heißt **vollpolynomielles Approximationsschema** (FPTAS), falls seine Laufzeit polynomiell in $|I|$ und $1/\varepsilon$ ist.

Beispiele Laufzeiten

- $O(n^{1/\varepsilon}) \rightsquigarrow$
- $O(2^{1/\varepsilon} n^4) \rightsquigarrow$
- $O(n^3/\varepsilon^2) \rightsquigarrow$

Approximationsschema

Sei Π ein Optimierungsproblem. Ein Algorithmus $\mathcal{A}$ heißt **Polynomielles Approximationsschema** (PTAS), wenn dieser für jede Eingabe (I, ε) mit $I \in D_\Pi$ und $\varepsilon > 0$ eine Lösung $s \in S_\Pi(I)$ ausgibt, die Folgendes erfüllt:

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$, falls Π Min-Problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$, falls Π Max-Problem.

Die Laufzeit von $\mathcal{A}$ ist polynom. in $|I|$ für **jedes** feste $\varepsilon > 0$.

$\mathcal{A}$ heißt **vollpolynomielles Approximationsschema** (FPTAS), falls seine Laufzeit polynomiell in $|I|$ und $1/\varepsilon$ ist.

Beispiele Laufzeiten

- $O(n^{1/\varepsilon}) \rightsquigarrow$ Polynomielles Approximationsschema
- $O(2^{1/\varepsilon} n^4) \rightsquigarrow$
- $O(n^3/\varepsilon^2) \rightsquigarrow$

Approximationsschema

Sei Π ein Optimierungsproblem. Ein Algorithmus $\mathcal{A}$ heißt **Polynomielles Approximationsschema** (PTAS), wenn dieser für jede Eingabe (I, ε) mit $I \in D_\Pi$ und $\varepsilon > 0$ eine Lösung $s \in S_\Pi(I)$ ausgibt, die Folgendes erfüllt:

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$, falls Π Min-Problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$, falls Π Max-Problem.

Die Laufzeit von $\mathcal{A}$ ist polynom. in $|I|$ für **jedes** feste $\varepsilon > 0$.

$\mathcal{A}$ heißt **vollpolynomielles Approximationsschema** (FPTAS), falls seine Laufzeit polynomiell in $|I|$ und $1/\varepsilon$ ist.

Beispiele Laufzeiten

- $O(n^{1/\varepsilon}) \rightsquigarrow$ Polynomielles Approximationsschema
- $O(2^{1/\varepsilon} n^4) \rightsquigarrow$ Polynomielles Approximationsschema
- $O(n^3/\varepsilon^2) \rightsquigarrow$

Approximationsschema

Sei Π ein Optimierungsproblem. Ein Algorithmus $\mathcal{A}$ heißt **Polynomielles Approximationsschema** (PTAS), wenn dieser für jede Eingabe (I, ε) mit $I \in D_\Pi$ und $\varepsilon > 0$ eine Lösung $s \in S_\Pi(I)$ ausgibt, die Folgendes erfüllt:

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$, falls Π Min-Problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$, falls Π Max-Problem.

Die Laufzeit von $\mathcal{A}$ ist polynom. in $|I|$ für **jedes** feste $\varepsilon > 0$.

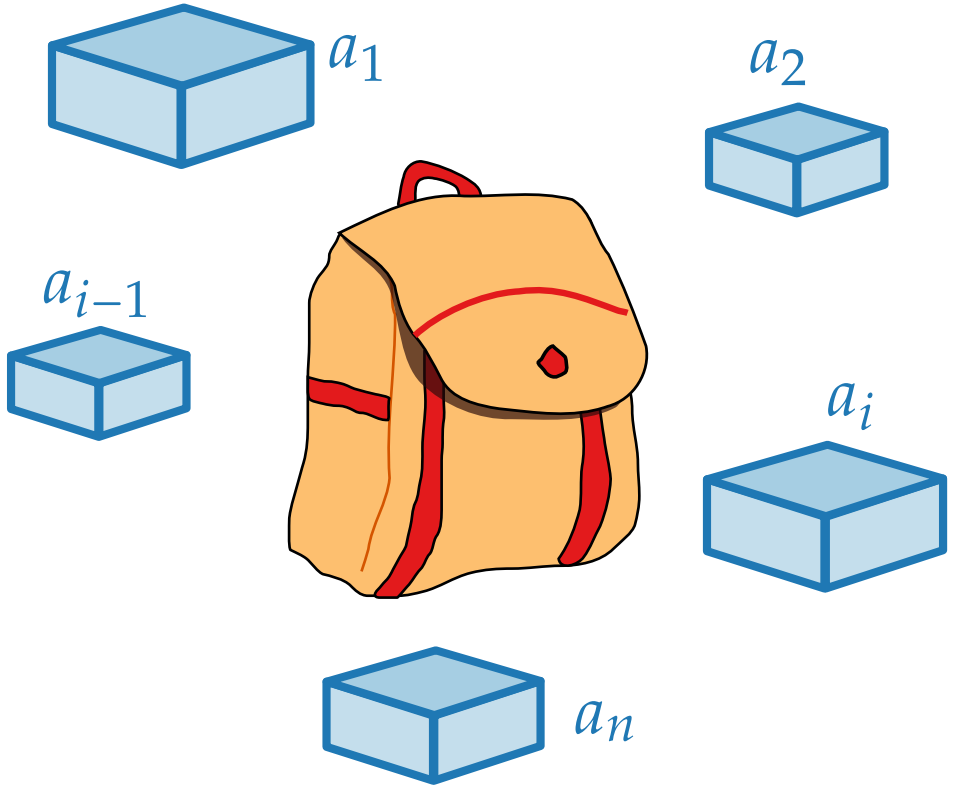
$\mathcal{A}$ heißt **vollpolynomielles Approximationsschema** (FPTAS), falls seine Laufzeit polynomiell in $|I|$ und $1/\varepsilon$ ist.

Beispiele Laufzeiten

- $O(n^{1/\varepsilon}) \rightsquigarrow$ Polynomielles Approximationsschema
- $O(2^{1/\varepsilon} n^4) \rightsquigarrow$ Polynomielles Approximationsschema
- $O(n^3/\varepsilon^2) \rightsquigarrow$ Vollpolynomielles Approximationsschema

Rucksack

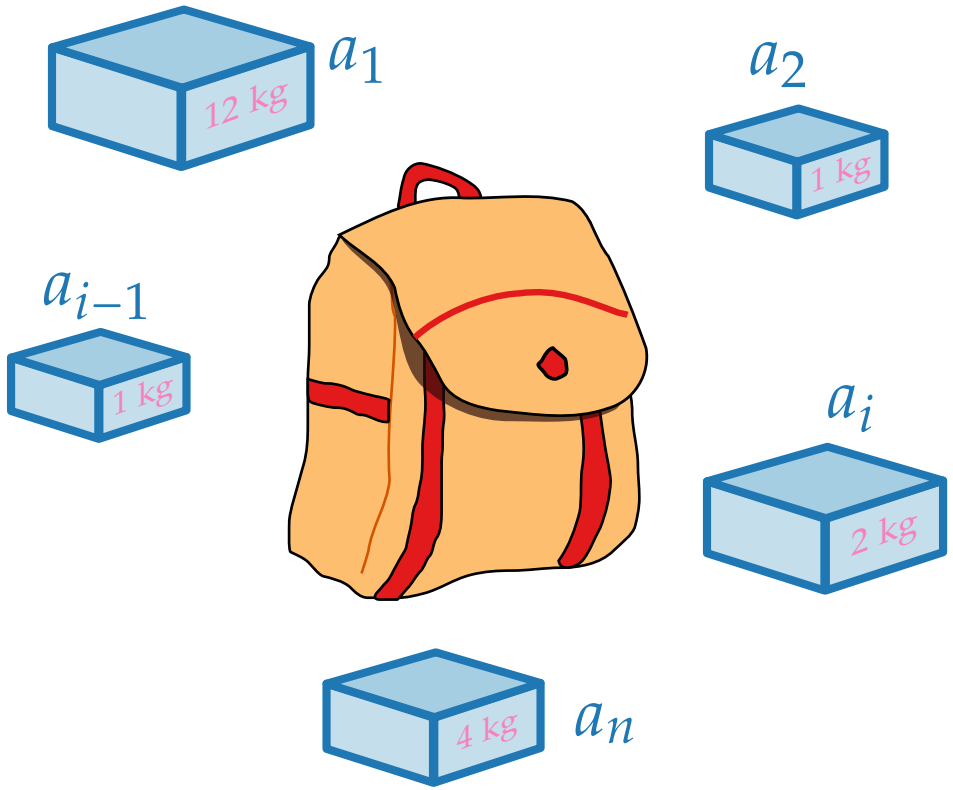
Gegeben: • Eine Menge $S = \{a_1, \ldots, a_n\}$ von **Objekten**.



Rucksack

Gegeben:

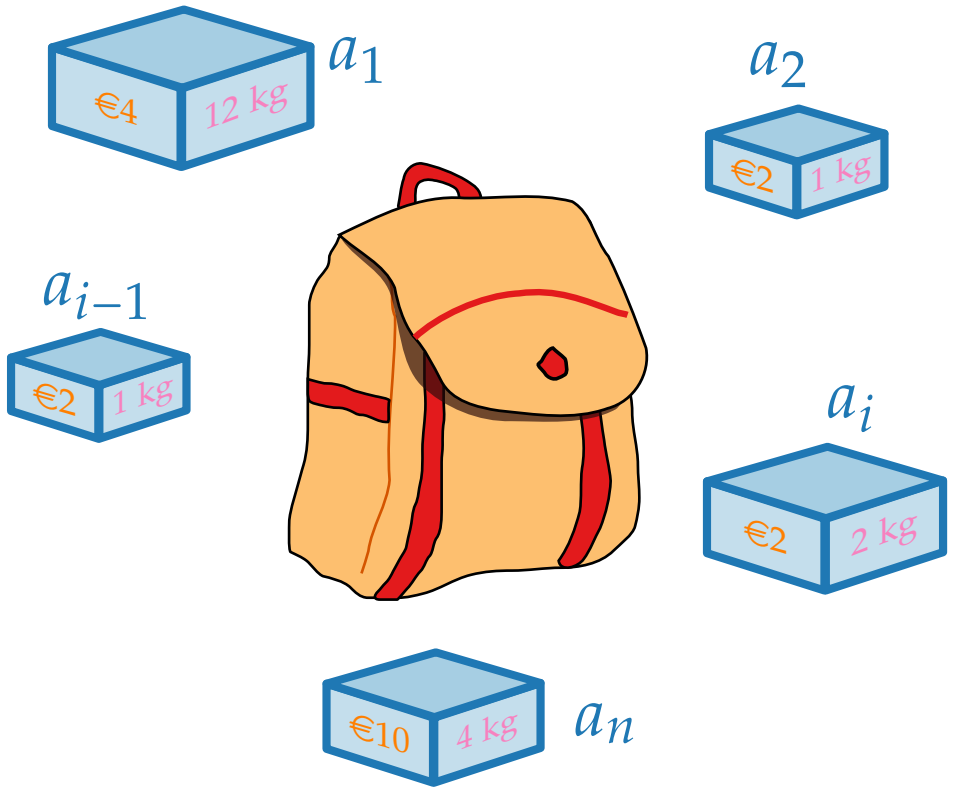
- Eine Menge $S = \{a_1, \ldots, a_n\}$ von **Objekten**.
- Für jedes Objekt a_i eine **Größe** $\mathrm{size}(a_i) \in \mathbb{N}^+$



Rucksack

Gegeben:

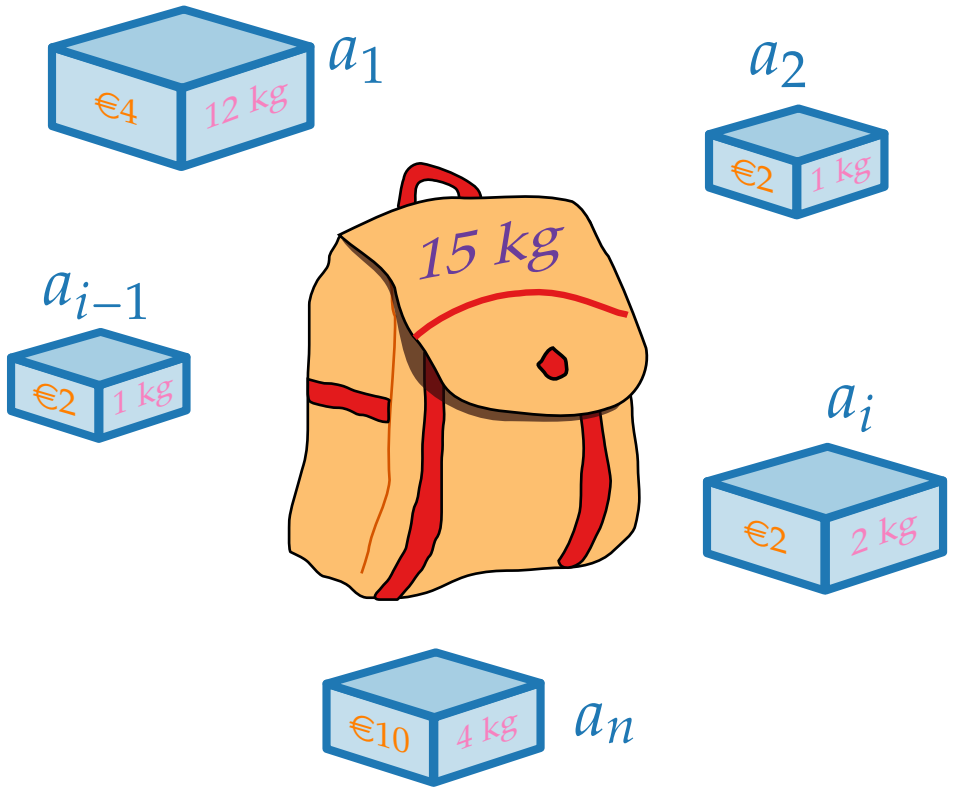
- Eine Menge $S = \{a_1, \ldots, a_n\}$ von **Objekten**.
- Für jedes Objekt a_i eine **Größe** $\text{size}(a_i) \in \mathbb{N}^+$
- Für jedes Objekt a_i ein **Profit** $\text{profit}(a_i) \in \mathbb{N}^+$



Rucksack

Gegeben:

- Eine Menge $S = \{a_1, \ldots, a_n\}$ von **Objekten**.
- Für jedes Objekt a_i eine **Größe** $\text{size}(a_i) \in \mathbb{N}^+$
- Für jedes Objekt a_i ein **Profit** $\text{profit}(a_i) \in \mathbb{N}^+$
- Eine Rucksack-**Kapazität** $B \in \mathbb{N}^+$

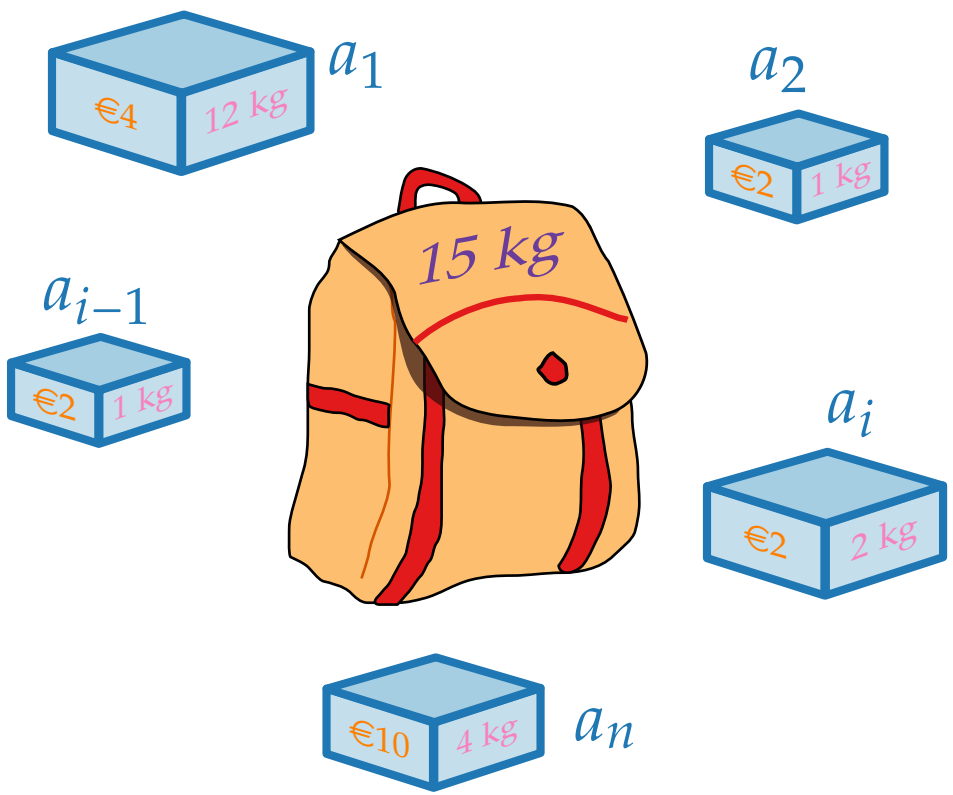


Rucksack

Gegeben:

- Eine Menge $S = \{a_1, \ldots, a_n\}$ von **Objekten**.
- Für jedes Objekt a_i eine **Größe** $\text{size}(a_i) \in \mathbb{N}^+$
- Für jedes Objekt a_i ein **Profit** $\text{profit}(a_i) \in \mathbb{N}^+$
- Eine Rucksack-**Kapazität** $B \in \mathbb{N}^+$

Gesucht: Eine Teilmenge von Objekten deren **Gesamtgröße** höchstens B und deren **Gesamtprofit** maximal ist.

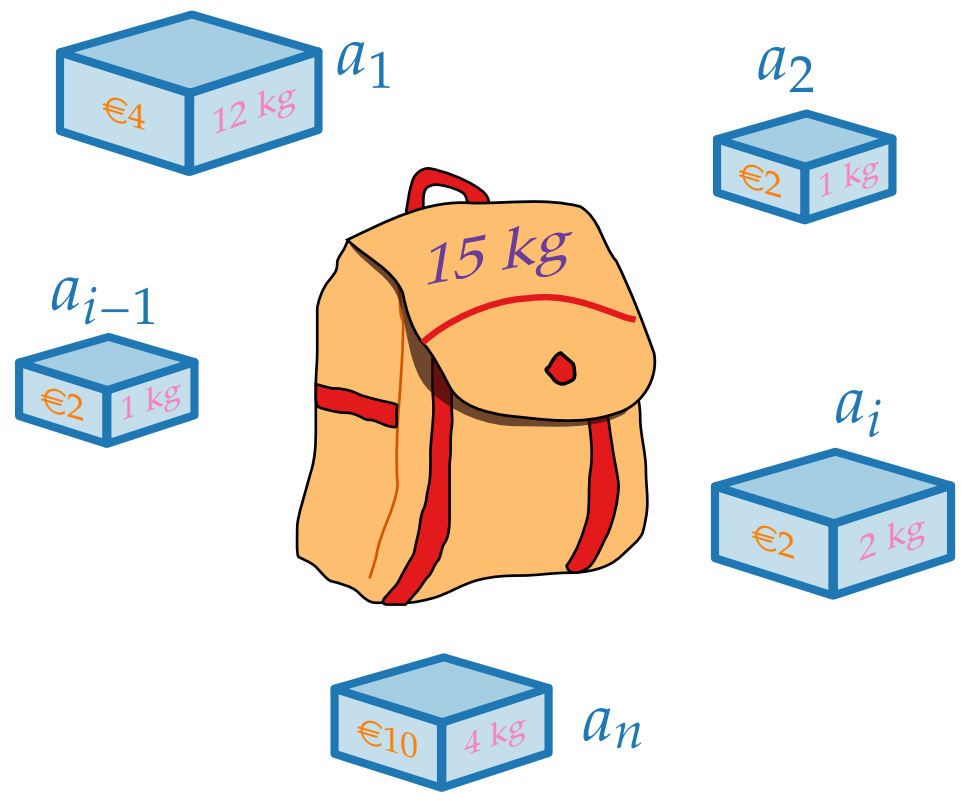


Rucksack

Gegeben:
- Eine Menge $S = \{a_1, \ldots, a_n\}$ von **Objekten**.
- Für jedes Objekt a_i eine **Größe** $\text{size}(a_i) \in \mathbb{N}^+$
- Für jedes Objekt a_i ein **Profit** $\text{profit}(a_i) \in \mathbb{N}^+$
- Eine Rucksack-**Kapazität** $B \in \mathbb{N}^+$

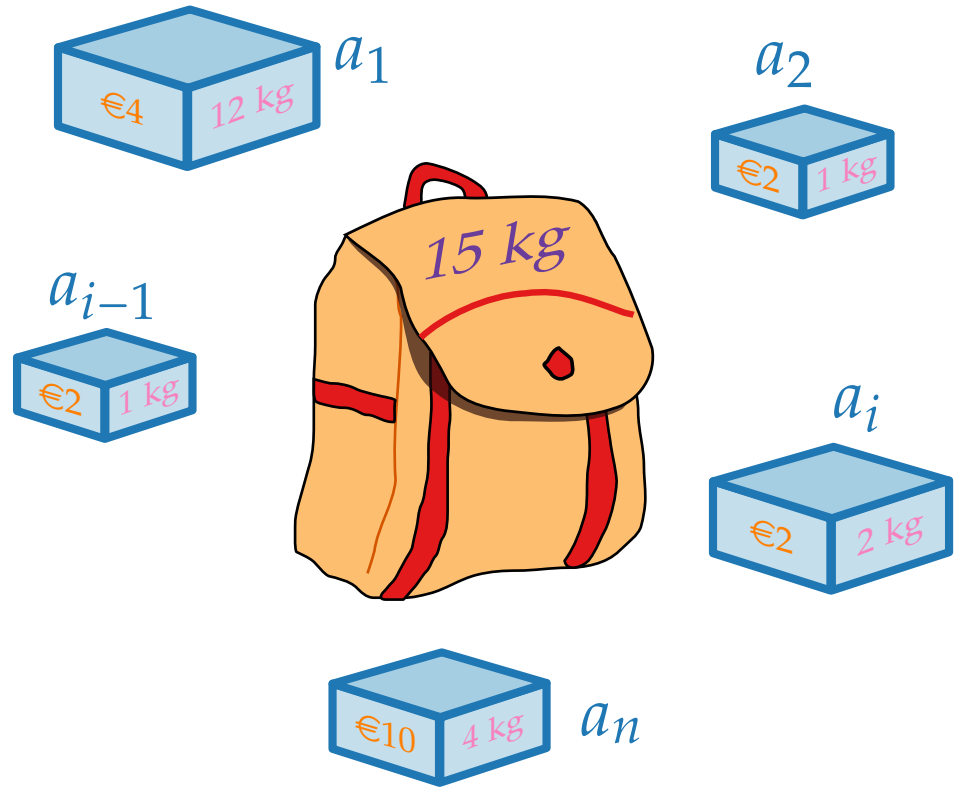
Gesucht: Eine Teilmenge von Objekten deren **Gesamtgröße** höchstens B und deren **Gesamtprofit** maximal ist.

NP-schwer



Pseudopolynomieller Algorithmus

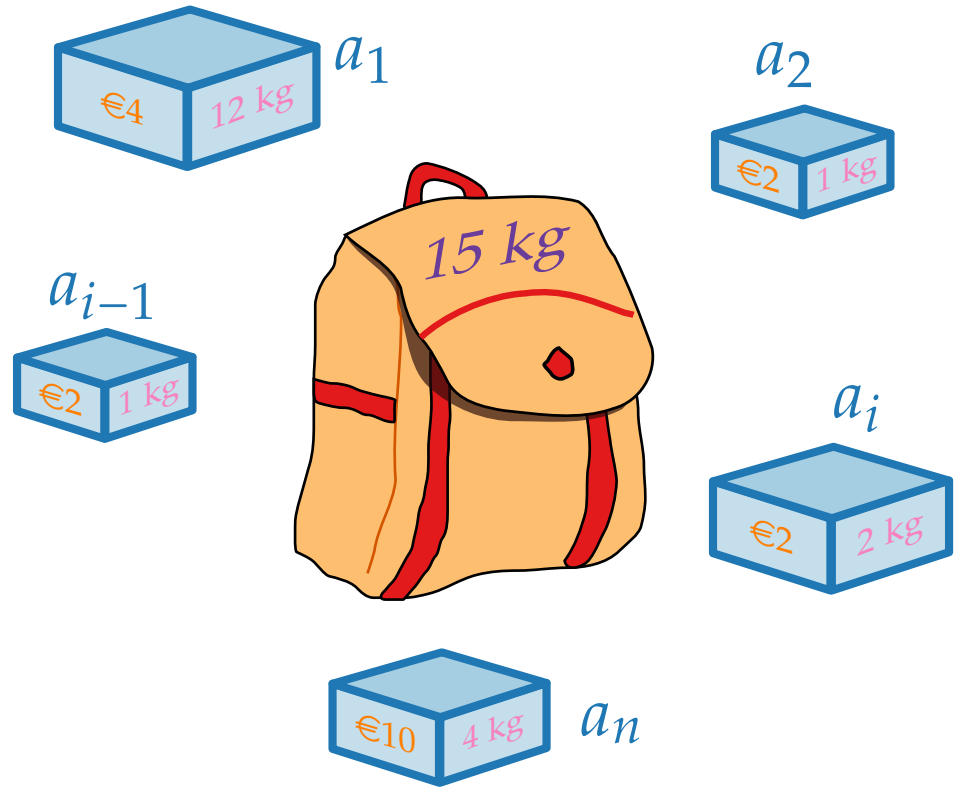
Sei Π ein Optimierungsproblem dessen Instanzen durch **Objekte** (z.B. Mengen oder Graphen) und **Zahlen** (z.B. Kosten, Gewichte, Profite) repräsentiert sind.



Pseudopolynomieller Algorithmus

Sei Π ein Optimierungsproblem dessen Instanzen durch **Objekte** (z.B. Mengen oder Graphen) und **Zahlen** (z.B. Kosten, Gewichte, Profite) repräsentiert sind.

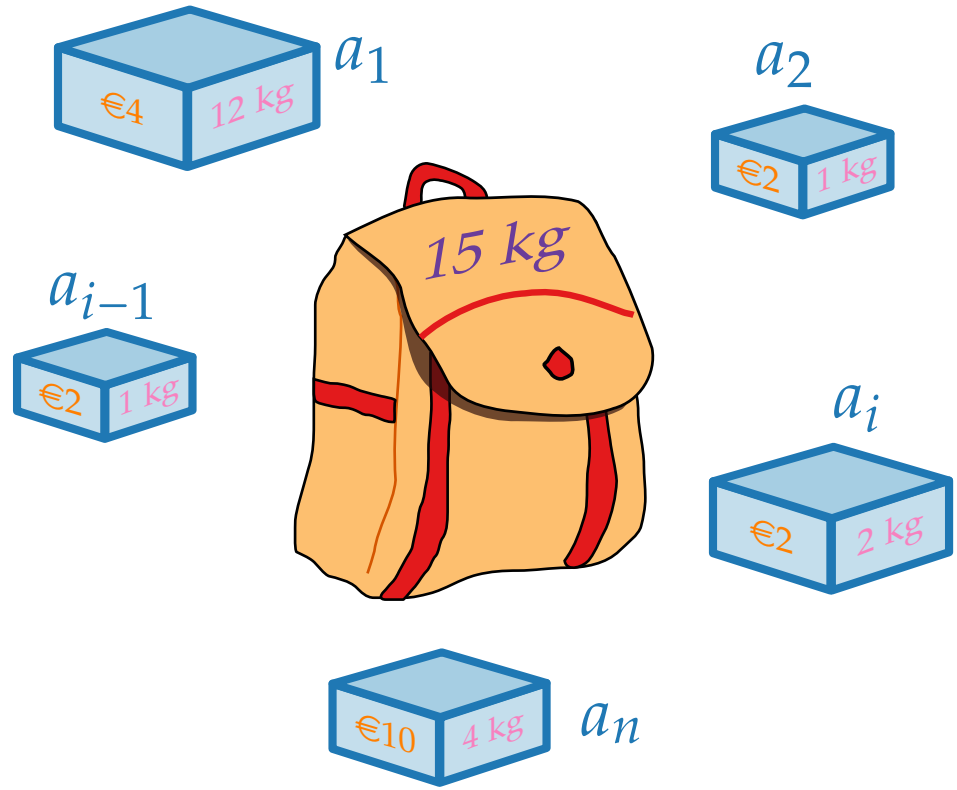
$|I|$: Die Größe einer Instanz $I \in D_\Pi$, wobei alle Zahlen in I **binär** kodiert sind.



Pseudopolynomieller Algorithmus

Sei Π ein Optimierungsproblem dessen Instanzen durch **Objekte** (z.B. Mengen oder Graphen) und **Zahlen** (z.B. Kosten, Gewichte, Profite) repräsentiert sind.

$|I|$: Die Größe einer Instanz $I \in D_\Pi$, wobei alle Zahlen in I **binär** kodiert sind. $(5 \mathrel{\hat{=}} 101 \Rightarrow |I| = 3)$

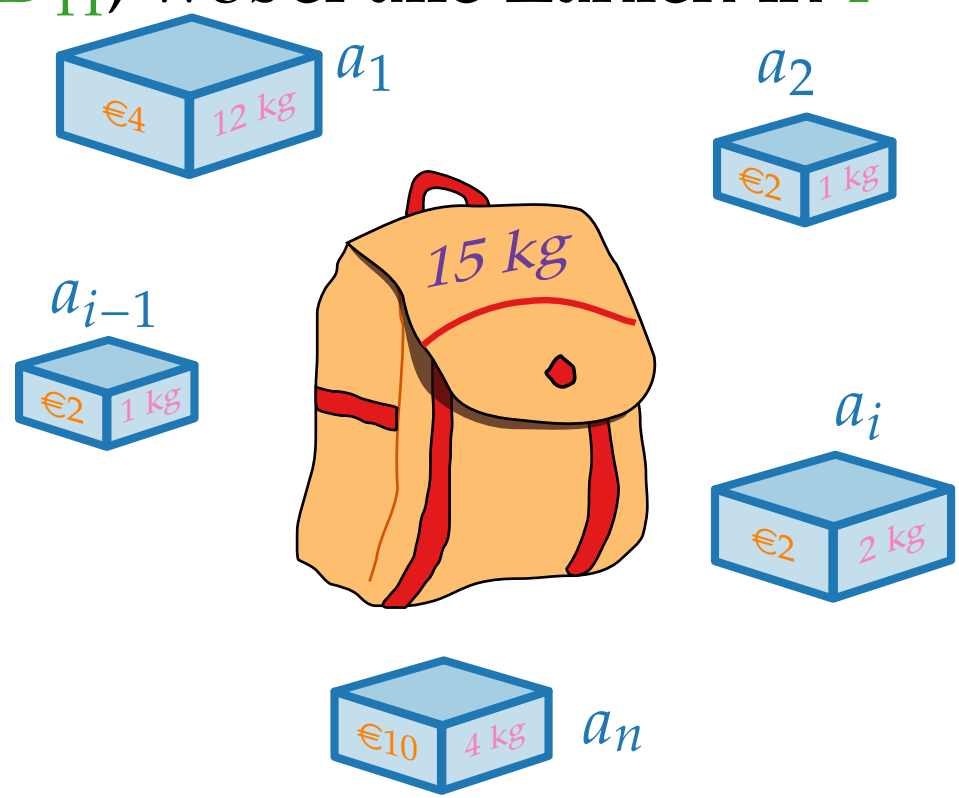


Pseudopolynomieller Algorithmus

Sei Π ein Optimierungsproblem dessen Instanzen durch **Objekte** (z.B. Mengen oder Graphen) und **Zahlen** (z.B. Kosten, Gewichte, Profite) repräsentiert sind.

$|I|$: Die Größe einer Instanz $I \in D_\Pi$, wobei alle Zahlen in I **binär** kodiert sind. $(5 \mathrel{\hat{=}} 101 \Rightarrow |I| = 3)$

$|I|_u$: Die Größe einer Instanz $I \in D_\Pi$, wobei alle Zahlen in I **unär** kodiert sind.



Pseudopolynomieller Algorithmus

Sei Π ein Optimierungsproblem dessen Instanzen durch **Objekte** (z.B. Mengen oder Graphen) und **Zahlen** (z.B. Kosten, Gewichte, Profite) repräsentiert sind.

$|I|$: Die Größe einer Instanz $I \in D_\Pi$, wobei alle Zahlen in I **binär** kodiert sind. $(5 \mathrel{\hat{=}} 101 \Rightarrow |I| = 3)$

$|I|_\mathrm{u}$: Die Größe einer Instanz $I \in D_\Pi$, wobei alle Zahlen in I **unär** kodiert sind. $(5 \mathrel{\hat{=}} 11111 \Rightarrow |I|_\mathrm{u} = 5)$

Pseudopolynomieller Algorithmus

Sei Π ein Optimierungsproblem dessen Instanzen durch **Objekte** (z.B. Mengen oder Graphen) und **Zahlen** (z.B. Kosten, Gewichte, Profite) repräsentiert sind.

$|I|$: Die Größe einer Instanz $I \in D_\Pi$, wobei alle Zahlen in I **binär** kodiert sind. $(5 \mathrel{\hat{=}} 101 \Rightarrow |I| = 3)$
$|I|_u$: Die Größe einer Instanz $I \in D_\Pi$, wobei alle Zahlen in I **unär** kodiert sind. $(5 \mathrel{\hat{=}} 11111 \Rightarrow |I|_u = 5)$

Die Laufzeit eines polynomiellen Algorithmus für Π ist polynomiell in $|I|$.

Pseudopolynomieller Algorithmus

Sei Π ein Optimierungsproblem dessen Instanzen durch **Objekte** (z.B. Mengen oder Graphen) und **Zahlen** (z.B. Kosten, Gewichte, Profite) repräsentiert sind.

$|I|$: Die Größe einer Instanz $I \in D_\Pi$, wobei alle Zahlen in I **binär** kodiert sind. $(5 \mathrel{\hat{=}} 101 \Rightarrow |I| = 3)$
$|I|_u$: Die Größe einer Instanz $I \in D_\Pi$, wobei alle Zahlen in I **unär** kodiert sind. $(5 \mathrel{\hat{=}} 11111 \Rightarrow |I|_u = 5)$

Die Laufzeit eines polynomiellen Algorithmus für Π ist polynomiell in $|I|$.

Die Laufzeit eines **pseudo-polynomiellen Algorithmus** ist polynomiell in $|I|_u$.

Pseudopolynomieller Algorithmus

Sei Π ein Optimierungsproblem dessen Instanzen durch **Objekte** (z.B. Mengen oder Graphen) und **Zahlen** (z.B. Kosten, Gewichte, Profite) repräsentiert sind.

$|I|$: Die Größe einer Instanz $I \in D_\Pi$, wobei alle Zahlen in I **binär** kodiert sind. $(5 \mathrel{\hat{=}} 101 \Rightarrow |I| = 3)$
$|I|_u$: Die Größe einer Instanz $I \in D_\Pi$, wobei alle Zahlen in I **unär** kodiert sind. $(5 \mathrel{\hat{=}} 11111 \Rightarrow |I|_u = 5)$

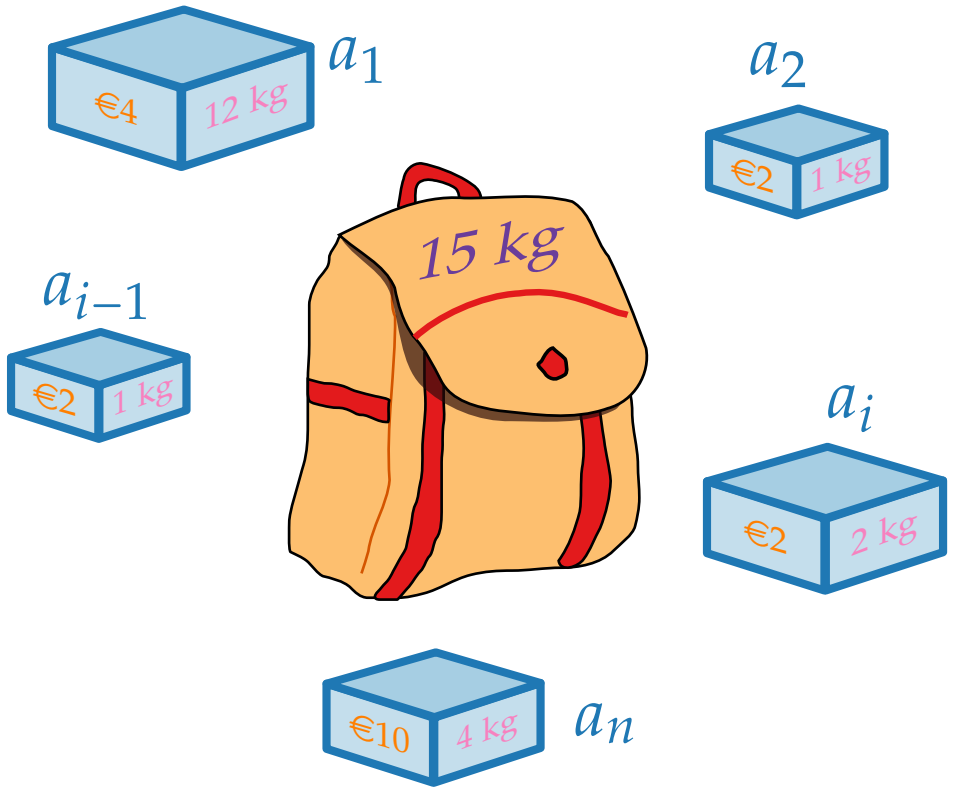
Die Laufzeit eines polynomiellen Algorithmus für Π ist polynomiell in $|I|$.

Die Laufzeit eines **pseudo-polynomiellen Algorithmus** ist polynomiell in $|I|_u$.

Laufzeit eines pseudo-polynomiellen Algorithmus ist nicht immer polynomiell in $|I|$.

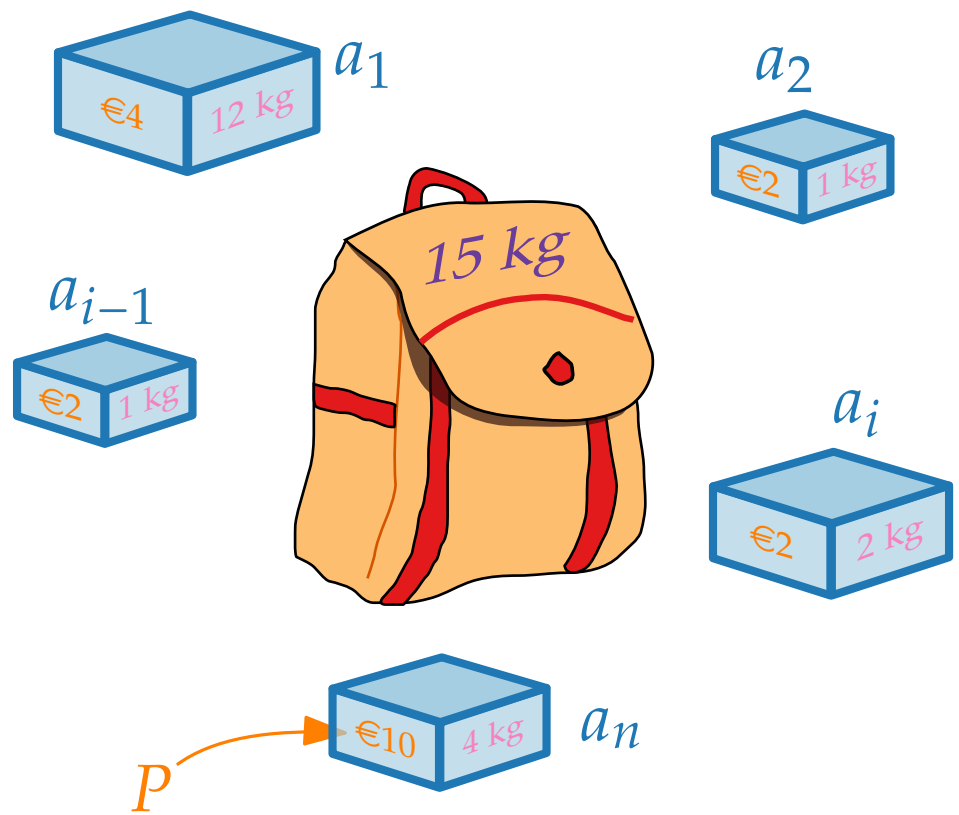
Pseudopolynomieller Alg. für Rucksack

Sei $P := \max_i \text{profit}(a_i)$



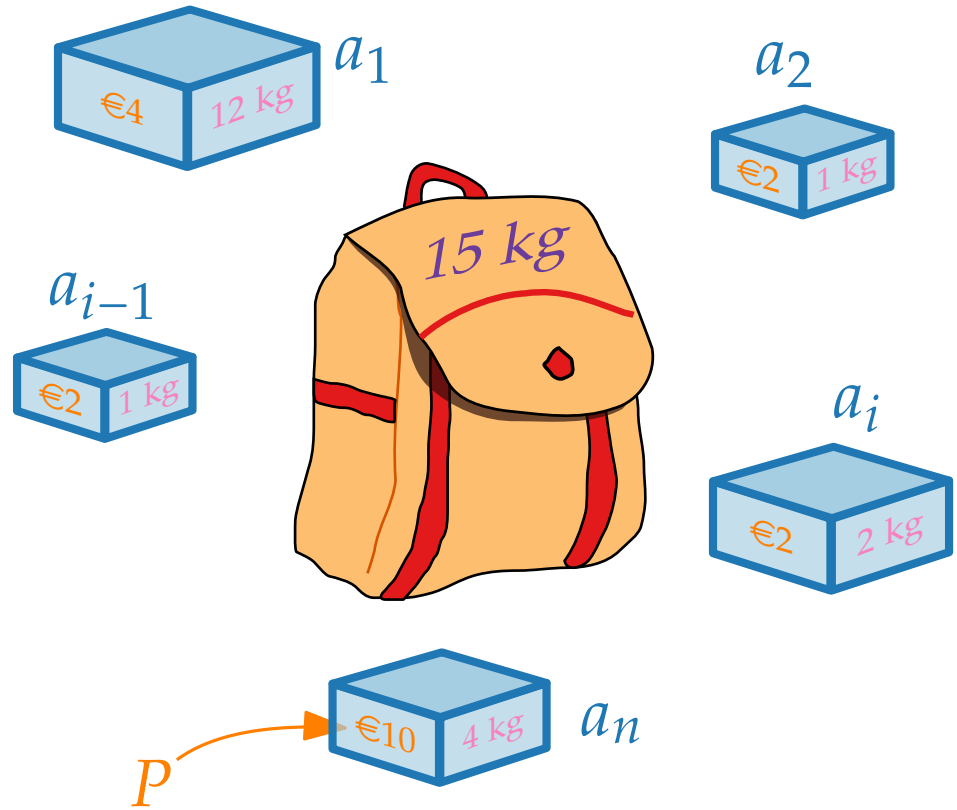
Pseudopolynomieller Alg. für Rucksack

Sei $P := \max_i \text{profit}(a_i)$



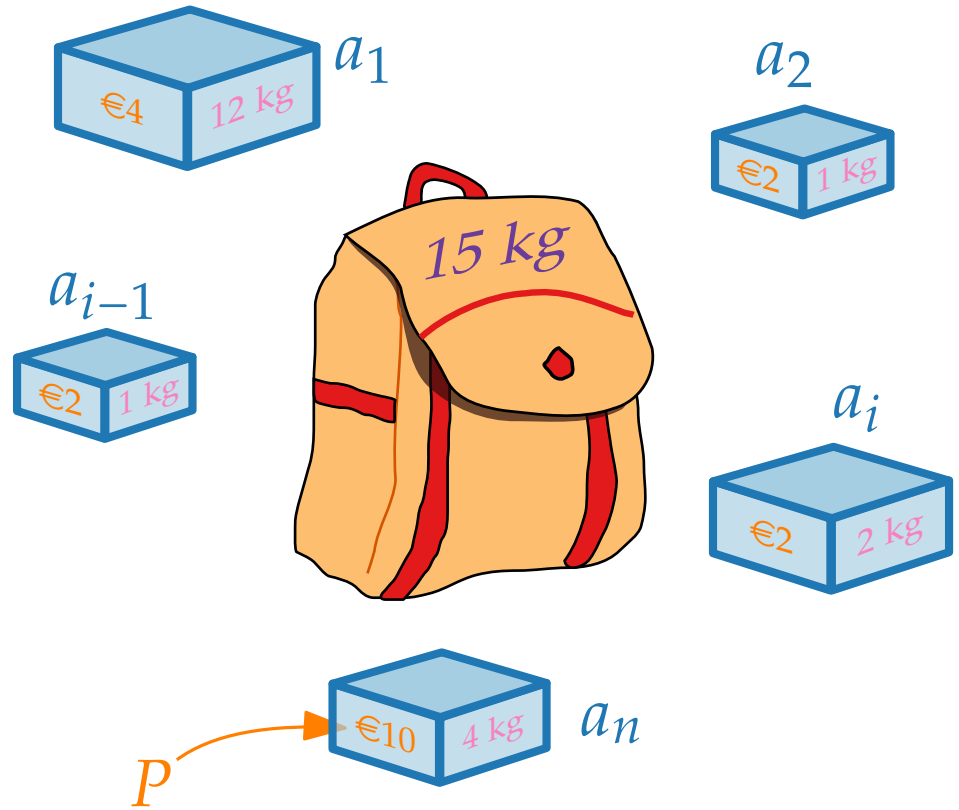
Pseudopolynomieller Alg. für Rucksack

Sei $P := \max_i \text{profit}(a_i) \quad \Rightarrow \qquad \leq \text{OPT} \leq$



Pseudopolynomieller Alg. für Rucksack

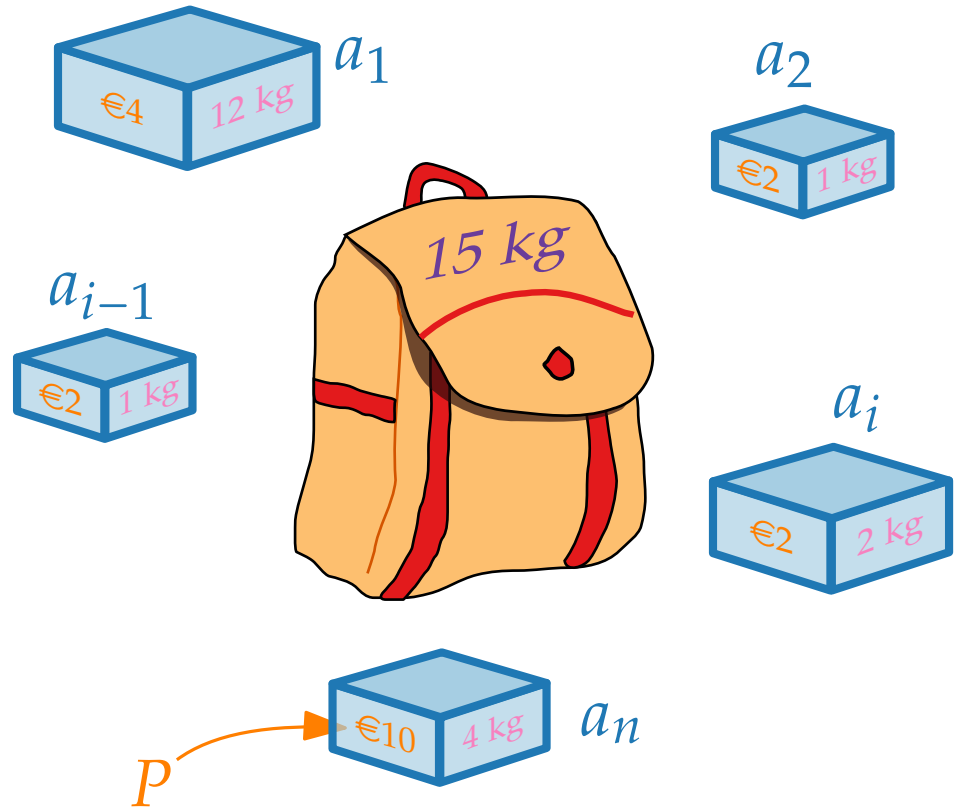
Sei $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$



Pseudopolynomieller Alg. für Rucksack

Sei $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

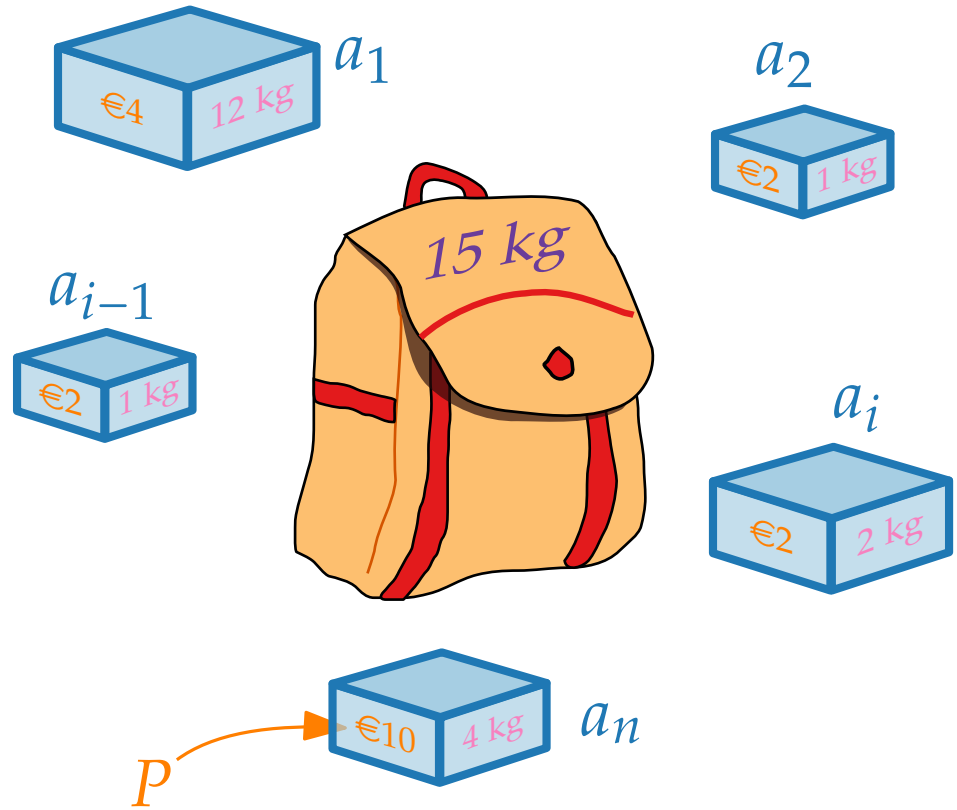
Für jedes $i = 1, \ldots, n$ und jedes $p \in \{1, \ldots, nP\}$



Pseudopolynomieller Alg. für Rucksack

Sei $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

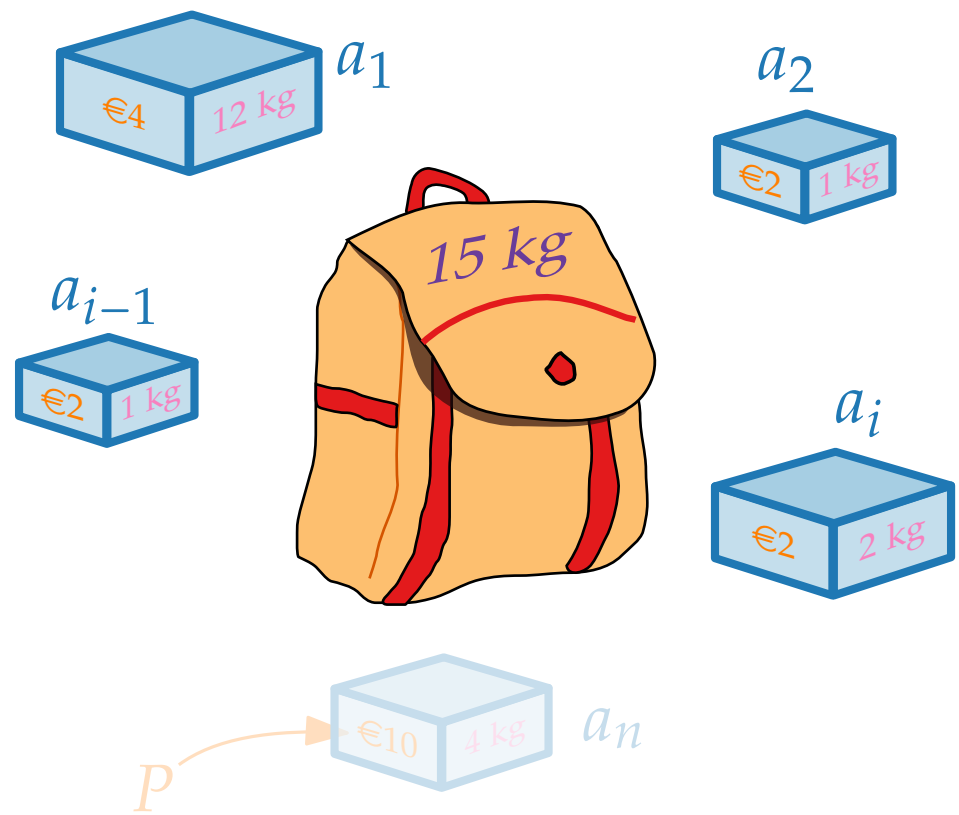
Für jedes $i = 1, \ldots, n$ und jedes $p \in \{1, \ldots, nP\}$
sei $S_{i,p}$ eine Teilmenge von $\{a_1, \ldots, a_i\}$ deren Profit genau p ist



Pseudopolynomieller Alg. für Rucksack

Sei $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

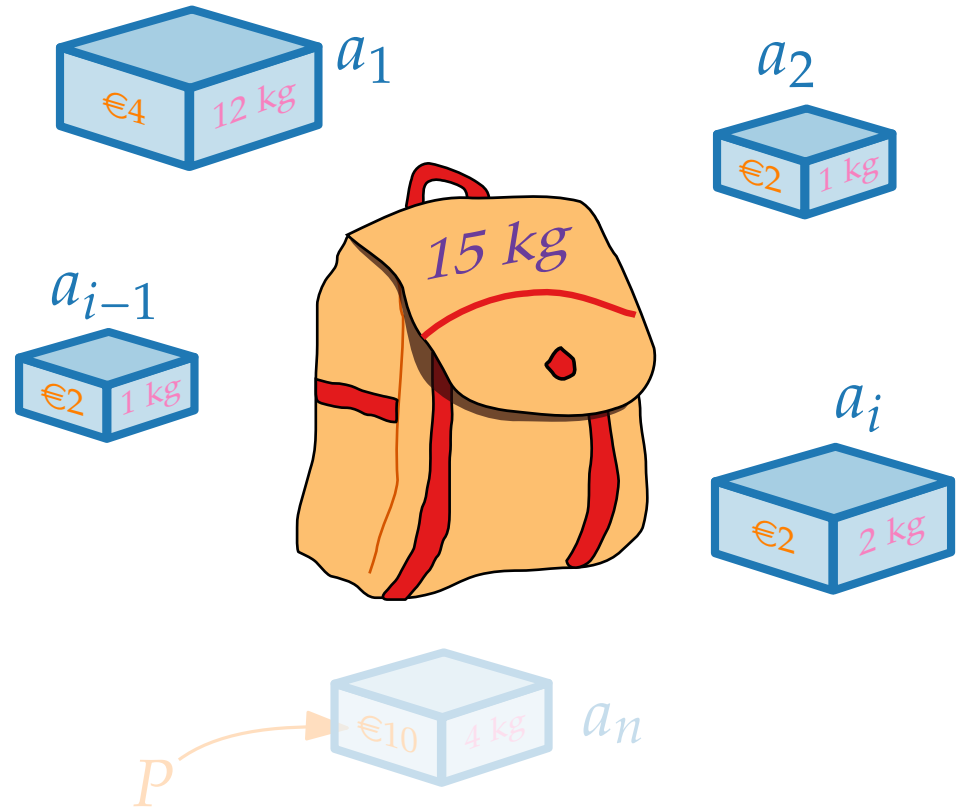
Für jedes $i = 1, \ldots, n$ und jedes $p \in \{1, \ldots, nP\}$
sei $S_{i,p}$ eine Teilmenge von $\{a_1, \ldots, a_i\}$ deren Profit genau p ist



Pseudopolynomieller Alg. für Rucksack

Sei $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

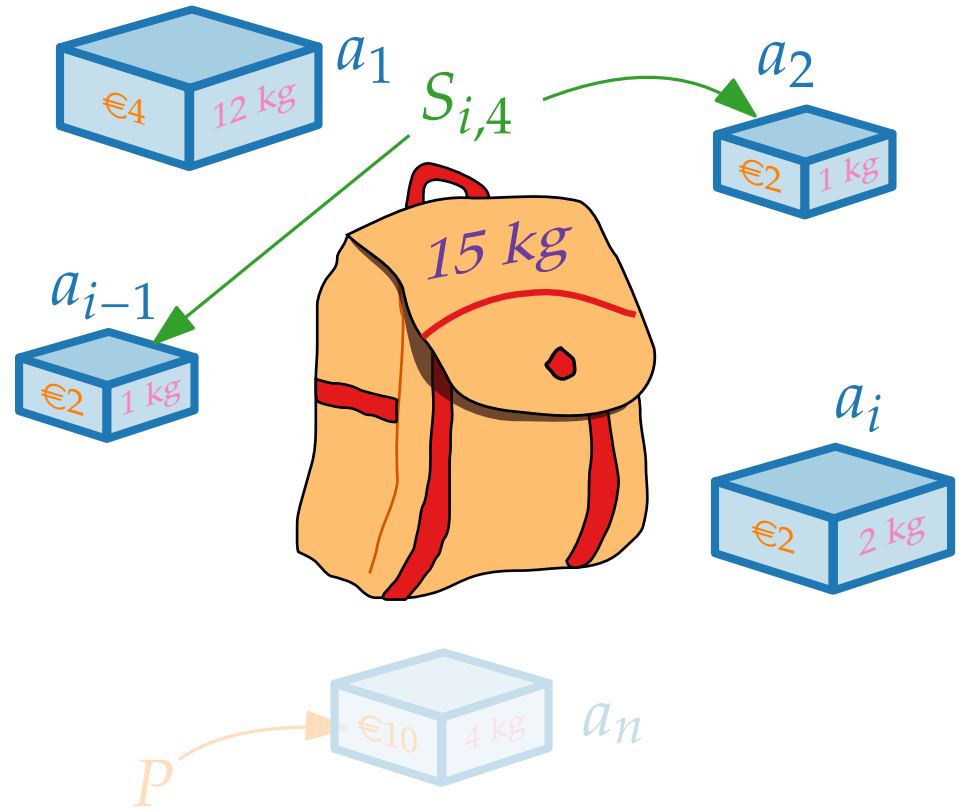
Für jedes $i = 1, \ldots, n$ und jedes $p \in \{1, \ldots, nP\}$ sei $S_{i,p}$ eine Teilmenge von $\{a_1, \ldots, a_i\}$ deren Profit genau p ist und deren Gesamtgröße minimal unter allen Teilmengen mit den geforderten Eigenschaften ist.



Pseudopolynomieller Alg. für Rucksack

Sei $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

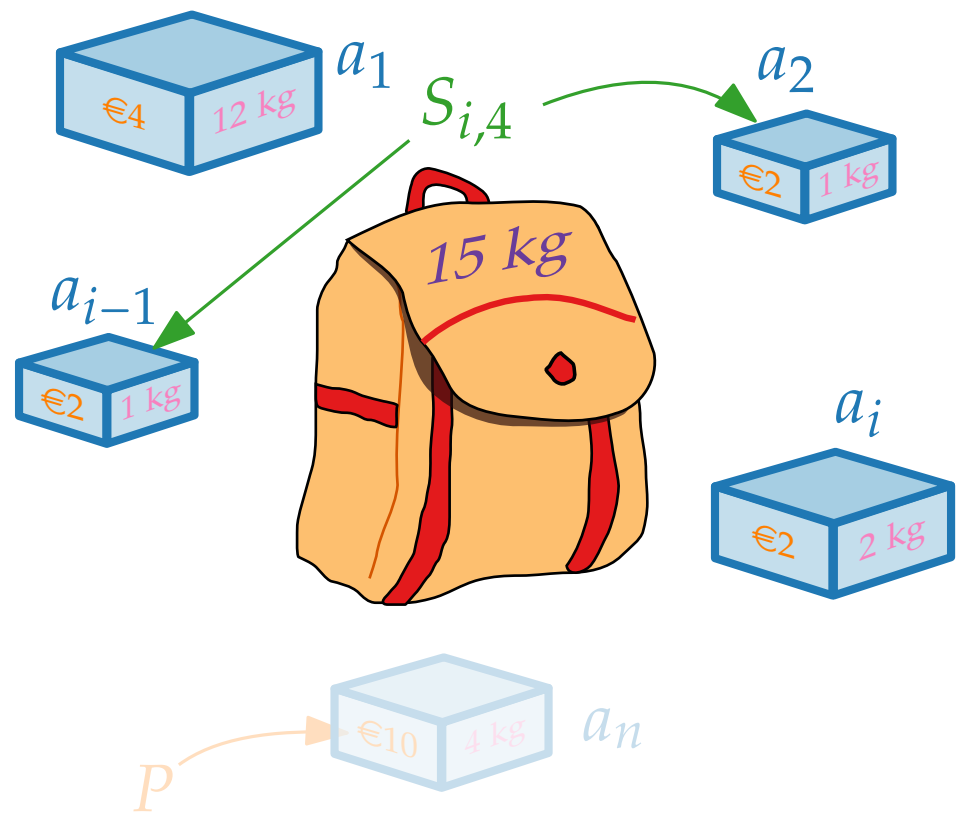
Für jedes $i = 1, \ldots, n$ und jedes $p \in \{1, \ldots, nP\}$ sei $S_{i,p}$ eine Teilmenge von $\{a_1, \ldots, a_i\}$ deren Profit genau p ist und deren Gesamtgröße minimal unter allen Teilmengen mit den geforderten Eigenschaften ist.



Pseudopolynomieller Alg. für Rucksack

Sei $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

Für jedes $i = 1, \ldots, n$ und jedes $p \in \{1, \ldots, nP\}$ sei $S_{i,p}$ eine Teilmenge von $\{a_1, \ldots, a_i\}$ deren Profit genau p ist und deren Gesamtgröße minimal unter allen Teilmengen mit den geforderten Eigenschaften ist. So eine Menge muss nicht existieren.

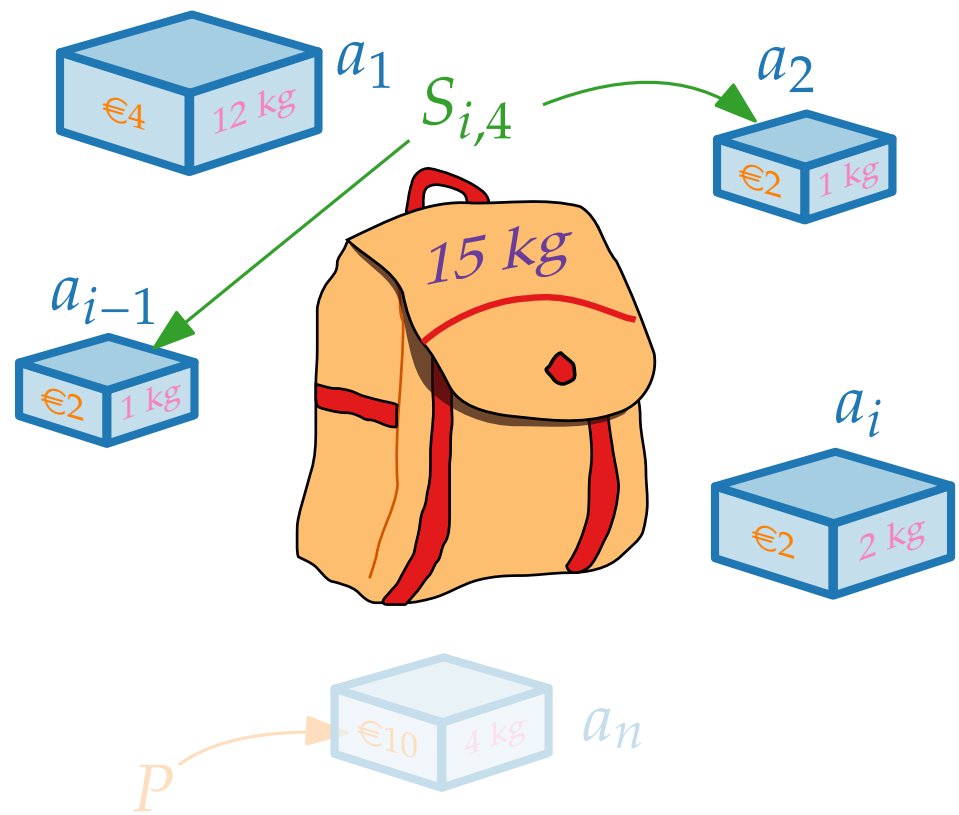


Pseudopolynomieller Alg. für Rucksack

Sei $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

Für jedes $i = 1, \ldots, n$ und jedes $p \in \{1, \ldots, nP\}$ sei $S_{i,p}$ eine Teilmenge von $\{a_1, \ldots, a_i\}$ deren Profit genau p ist und deren Gesamtgröße minimal unter allen Teilmengen mit den geforderten Eigenschaften ist. So eine Menge muss nicht existieren.

Sei $A[i,p]$ die Gesamtgröße der Menge $S_{i,p}$ (setzen $A[i,p] = \infty$ falls keine solche Menge existiert).

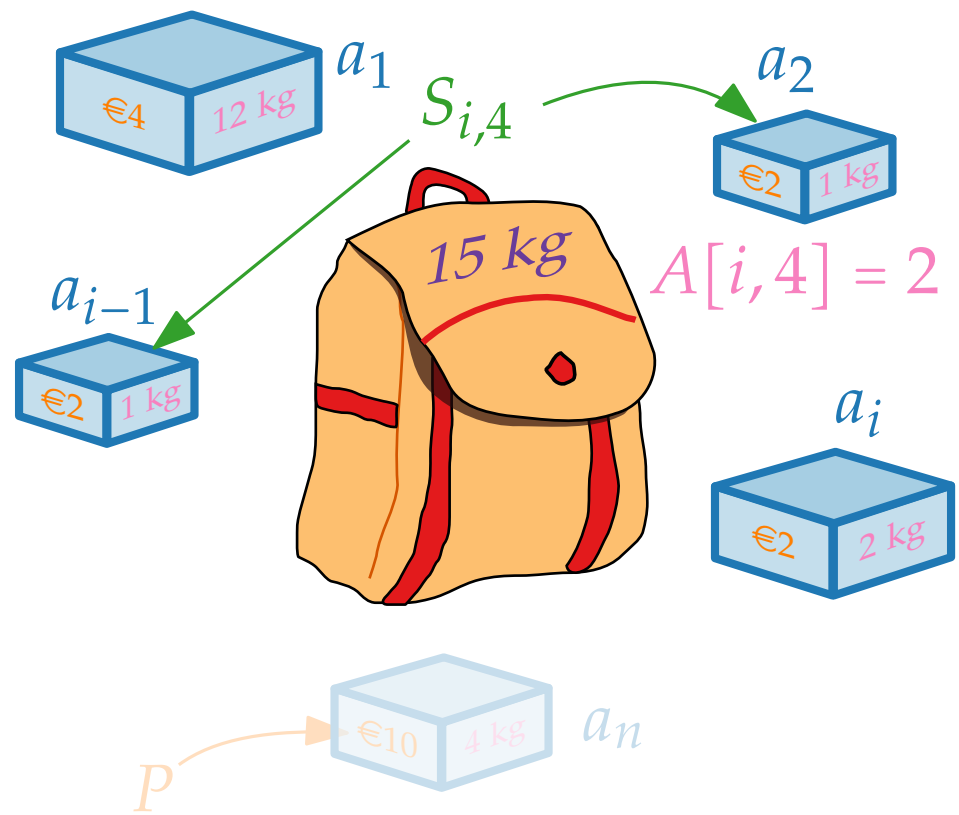


Pseudopolynomieller Alg. für Rucksack

Sei $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

Für jedes $i = 1, \ldots, n$ und jedes $p \in \{1, \ldots, nP\}$ sei $S_{i,p}$ eine Teilmenge von $\{a_1, \ldots, a_i\}$ deren Profit genau p ist und deren Gesamtgröße minimal unter allen Teilmengen mit den geforderten Eigenschaften ist. So eine Menge muss nicht existieren.

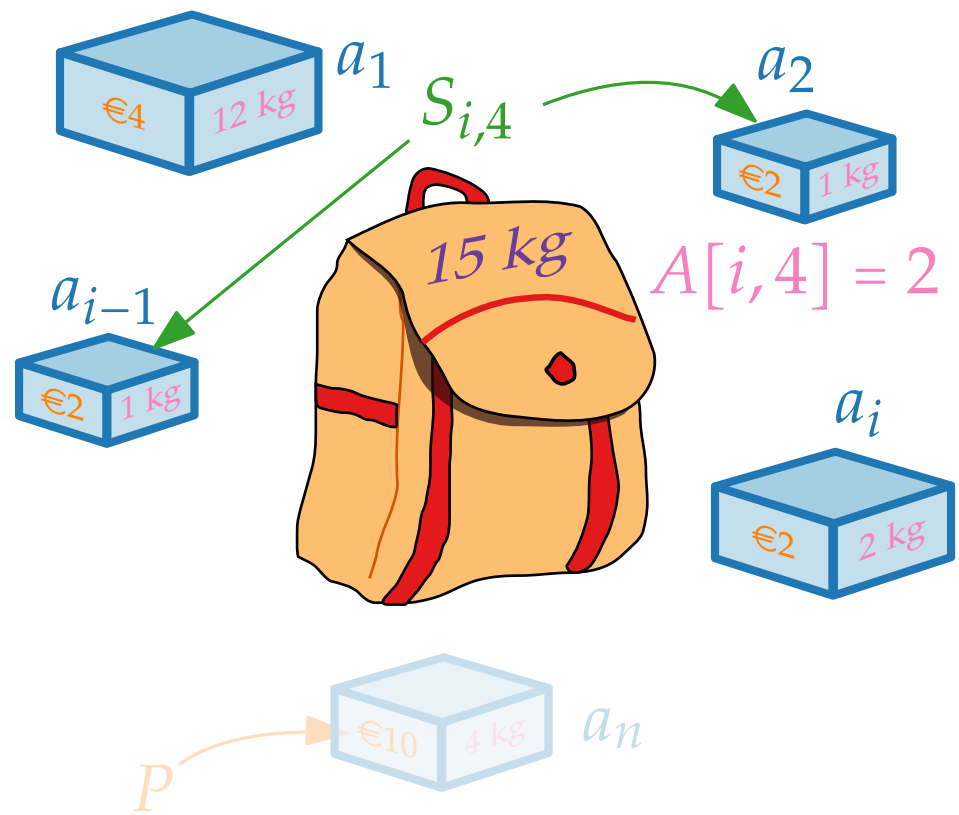
Sei $A[i,p]$ die Gesamtgröße der Menge $S_{i,p}$ (setzen $A[i,p] = \infty$ falls keine solche Menge existiert).



Pseudopolynomieller Alg. für Rucksack

Sei $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

Für jedes $i = 1, \ldots, n$ und jedes $p \in \{1, \ldots, nP\}$ sei $S_{i,p}$ eine Teilmenge von $\{a_1, \ldots, a_i\}$ deren Profit genau p ist und deren Gesamtgröße minimal unter allen Teilmengen mit den geforderten Eigenschaften ist. So eine Menge muss nicht existieren.



Sei $A[i,p]$ die Gesamtgröße der Menge $S_{i,p}$ (setzen $A[i,p] = \infty$ falls keine solche Menge existiert).

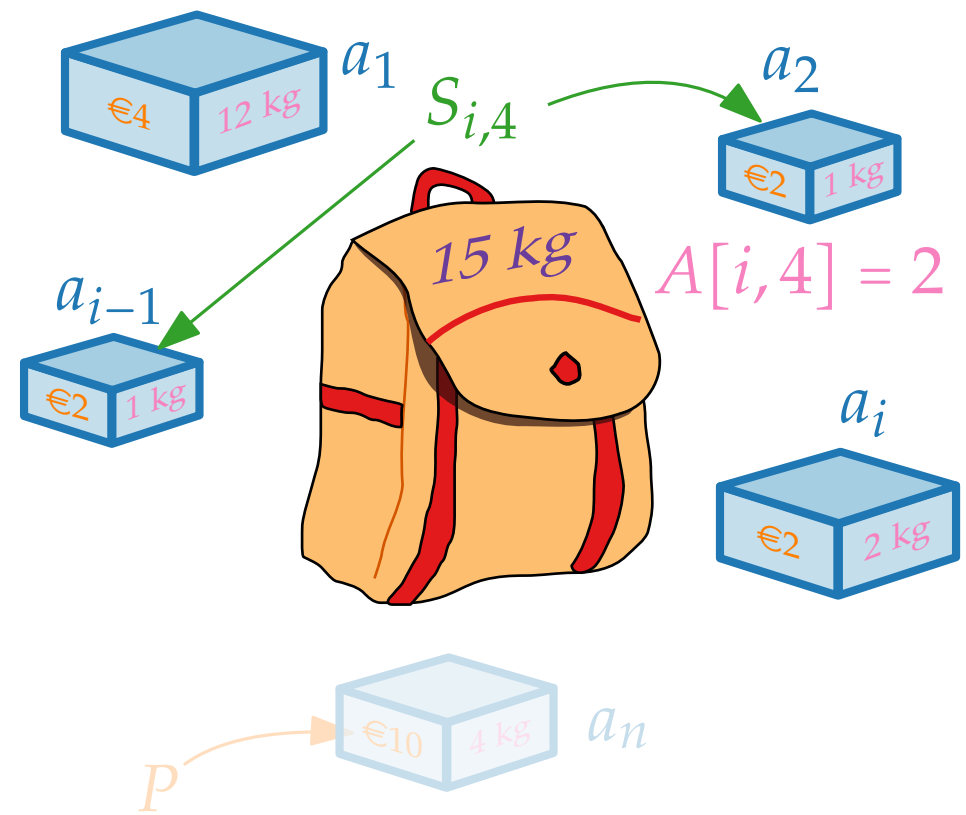
Falls alle $A[i,p]$ bekannt, dann ergibt sich

$\text{OPT} =$

Pseudopolynomieller Alg. für Rucksack

Sei $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

Für jedes $i = 1, \ldots, n$ und jedes $p \in \{1, \ldots, nP\}$ sei $S_{i,p}$ eine Teilmenge von $\{a_1, \ldots, a_i\}$ deren Profit genau p ist und deren Gesamtgröße minimal unter allen Teilmengen mit den geforderten Eigenschaften ist. So eine Menge muss nicht existieren.

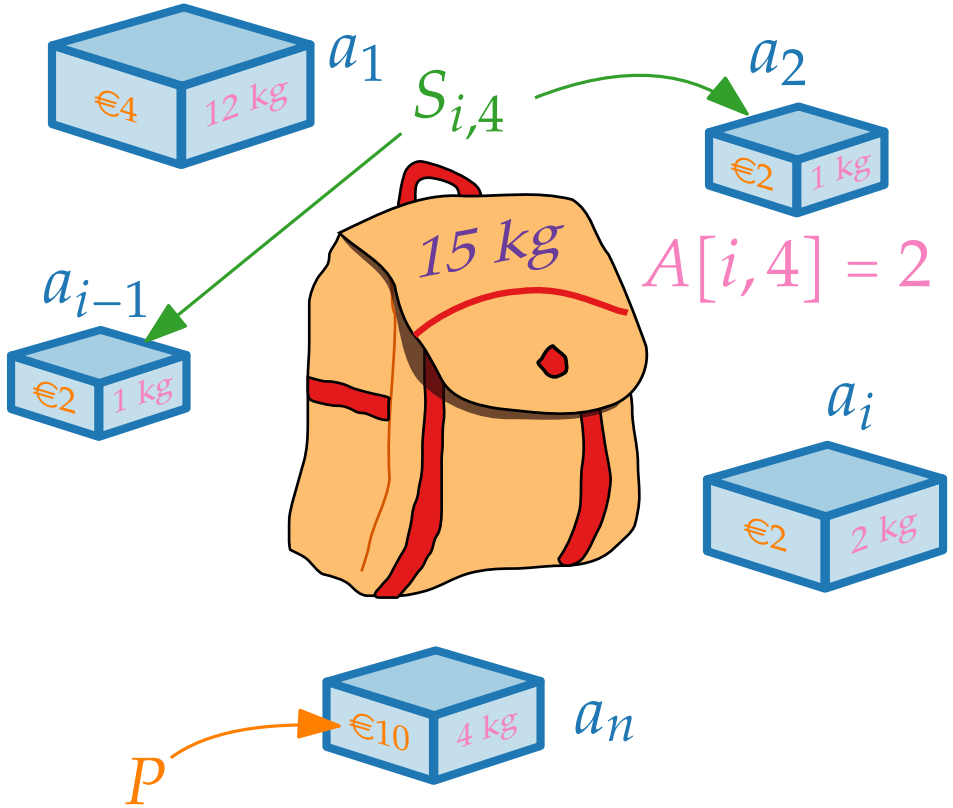


Sei $A[i,p]$ die Gesamtgröße der Menge $S_{i,p}$ (setzen $A[i,p] = \infty$ falls keine solche Menge existiert).

Falls alle $A[i,p]$ bekannt, dann ergibt sich
$\text{OPT} = \max\{\, p \mid A[n,p] \leq B \,\}$

Pseudopolynomieller Alg. für Rucksack

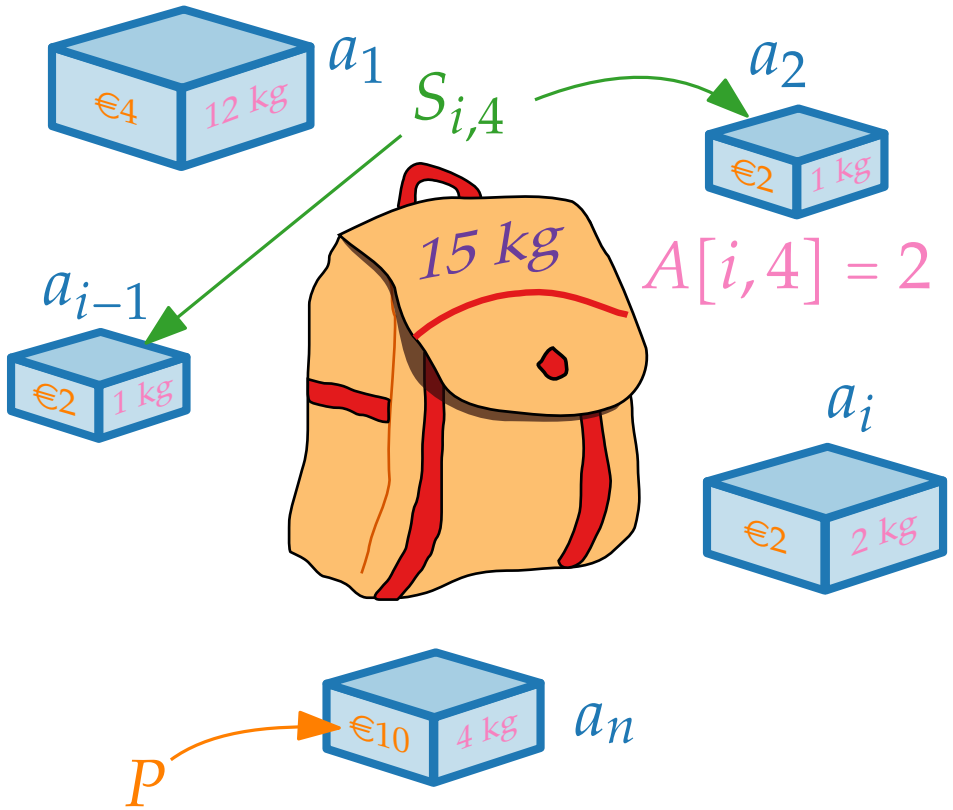
$A[1,p]$ lässt sich für alle $p \in \{0,\ldots,nP\}$ berechnen



Pseudopolynomieller Alg. für Rucksack

$A[1,p]$ lässt sich für alle $p \in \{0,\ldots,nP\}$ berechnen

Setze $A[i,p] := \infty$ für $p < 0$

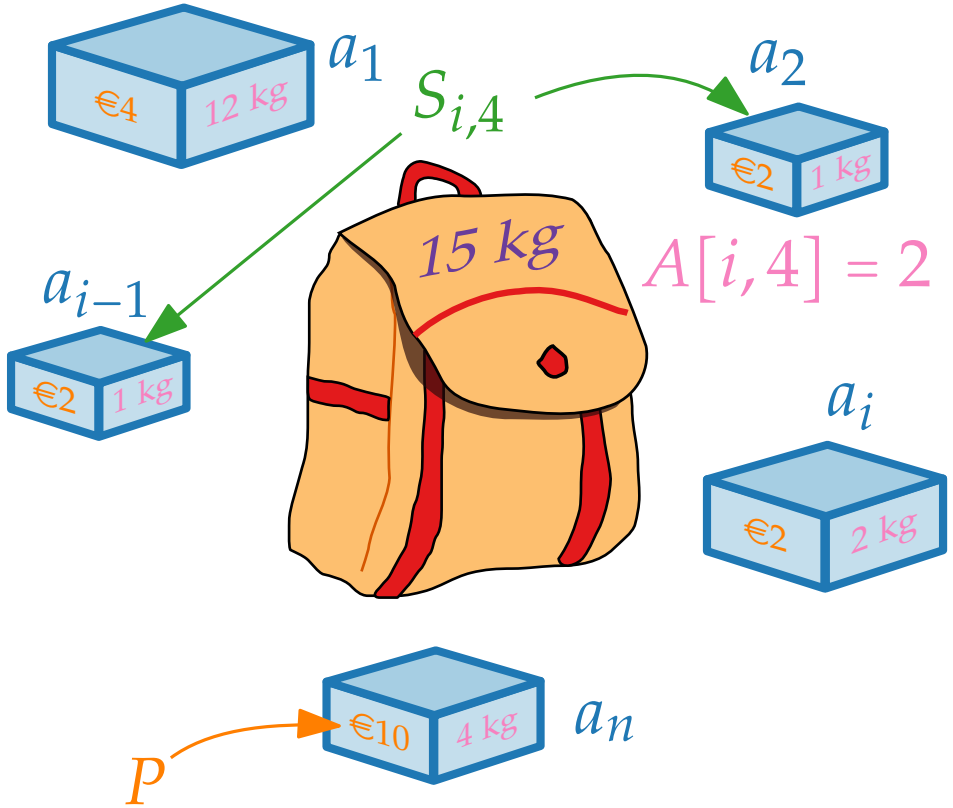


Pseudopolynomieller Alg. für Rucksack

$A[1,p]$ lässt sich für alle $p \in \{0,\ldots,nP\}$ berechnen

Setze $A[i,p] := \infty$ für $p < 0$

$A[i+1,p] =$

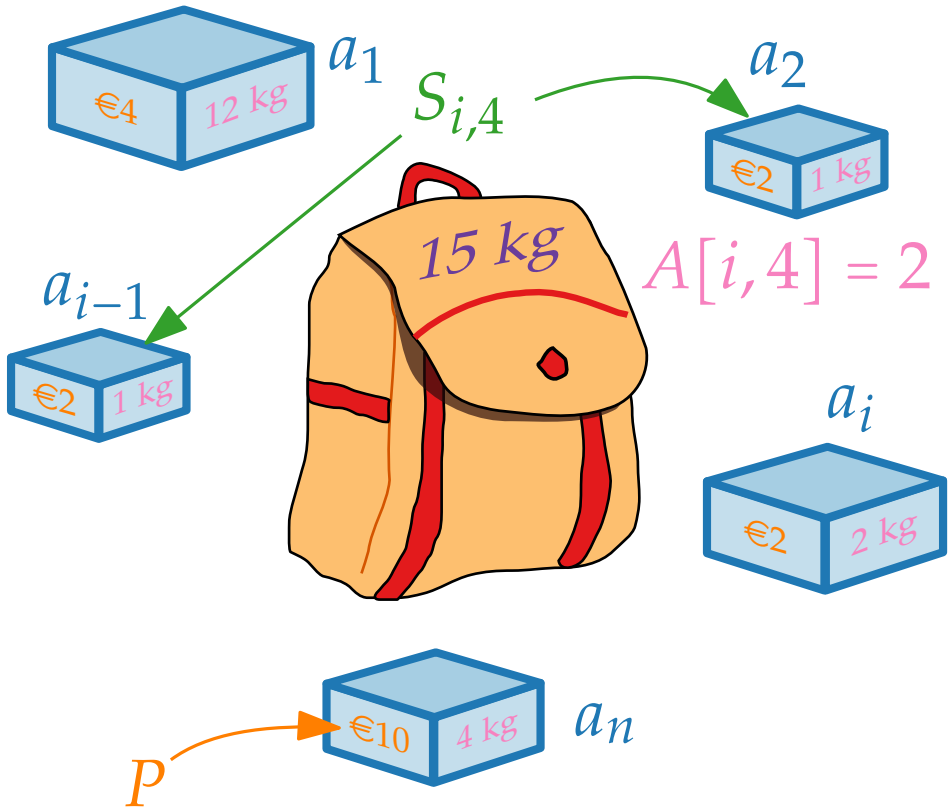


Pseudopolynomieller Alg. für Rucksack

$A[1,p]$ lässt sich für alle $p \in \{0,\ldots,nP\}$ berechnen

Setze $A[i,p] := \infty$ für $p < 0$

$A[i+1,p] = \min\{ \qquad \}$

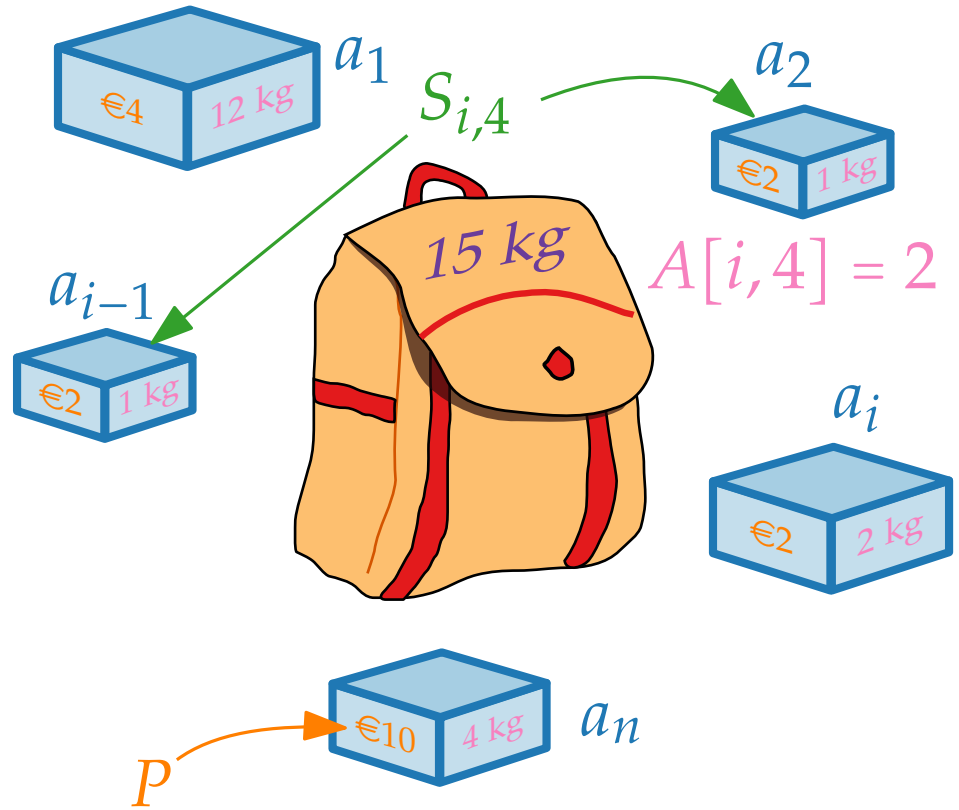


Pseudopolynomieller Alg. für Rucksack

$A[1,p]$ lässt sich für alle $p \in \{0,\ldots,nP\}$ berechnen

Setze $A[i,p] := \infty$ für $p < 0$

$A[i+1,p] = \min\{A[i,p],$ $\}$

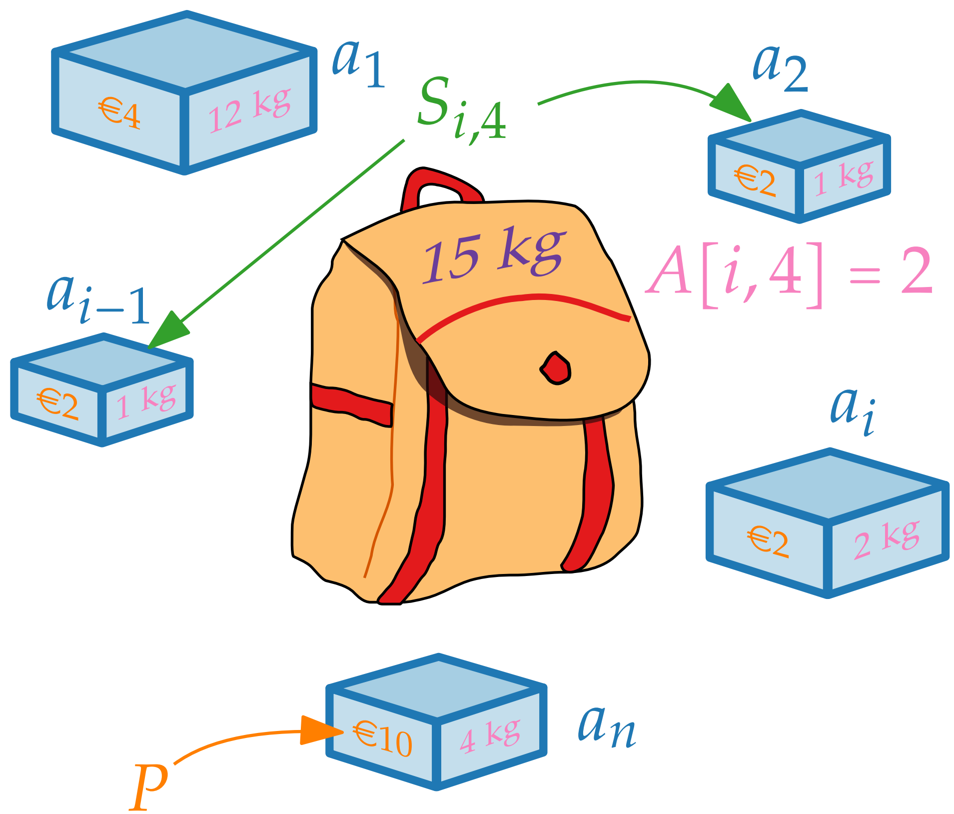


Pseudopolynomieller Alg. für RUCKSACK

$A[1,p]$ lässt sich für alle $p \in \{0,\dots,nP\}$ berechnen

Setze $A[i,p] := \infty$ für $p < 0$

$A[i+1,p] = \min\{A[i,p], \text{size}(a_{i+1}) + \qquad\qquad\qquad \}$

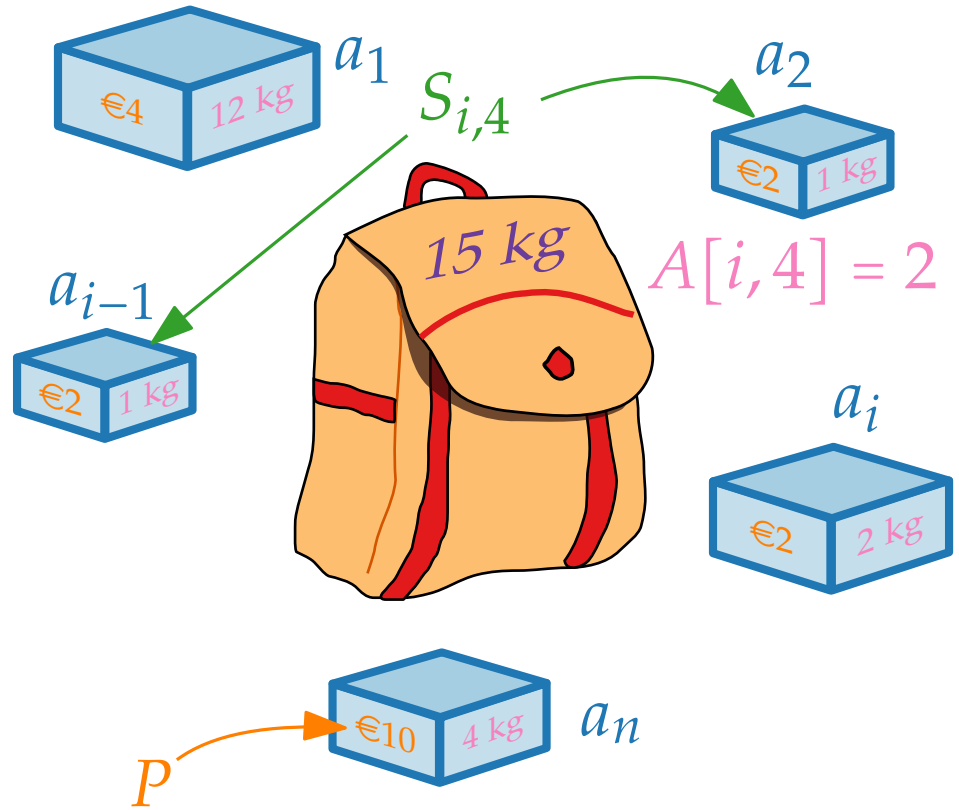


Pseudopolynomieller Alg. für Rucksack

$A[1,p]$ lässt sich für alle $p \in \{0,\ldots,nP\}$ berechnen

Setze $A[i,p] := \infty$ für $p < 0$

$A[i+1,p] = \min\{A[i,p], \text{size}(a_{i+1}) + A[i, p - \text{profit}(a_{i+1})]\}$



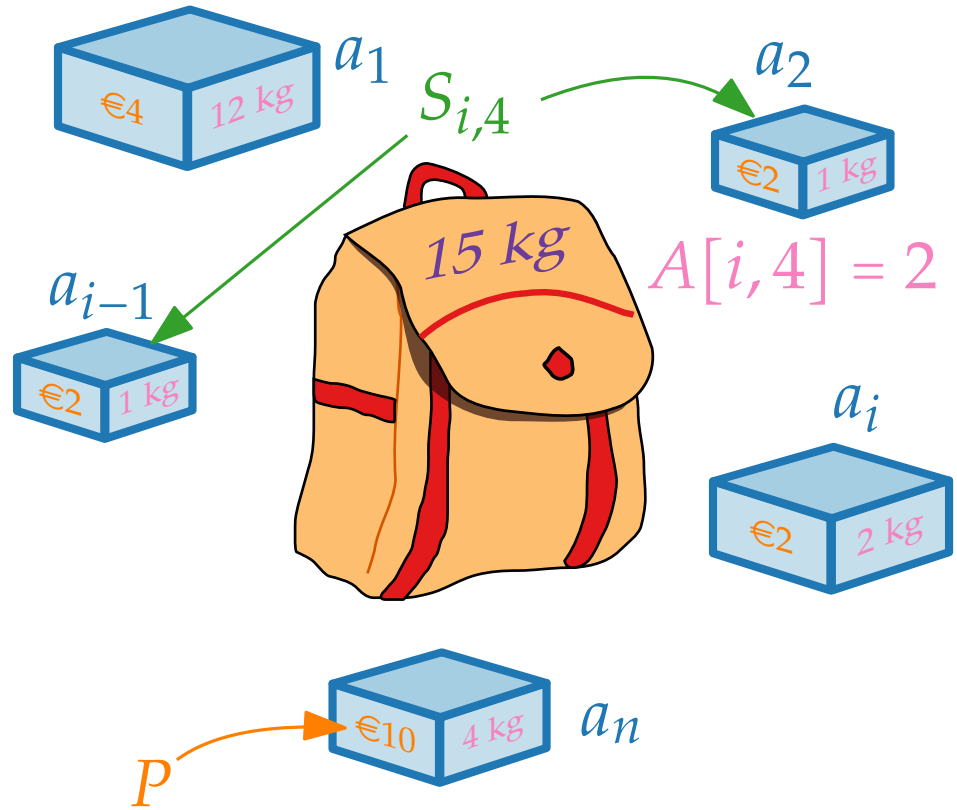
Pseudopolynomieller Alg. für Rucksack

$A[1,p]$ lässt sich für alle $p \in \{0,\ldots,nP\}$ berechnen

Setze $A[i,p] := \infty$ für $p < 0$

$A[i+1,p] = \min\{A[i,p], \text{size}(a_{i+1}) + A[i, p - \text{profit}(a_{i+1})]\}$

$\Rightarrow$ Alle $A[i,p]$-Werte lassen sich in $O(n^2 P)$ Zeit ermitteln



Pseudopolynomieller Alg. für Rucksack

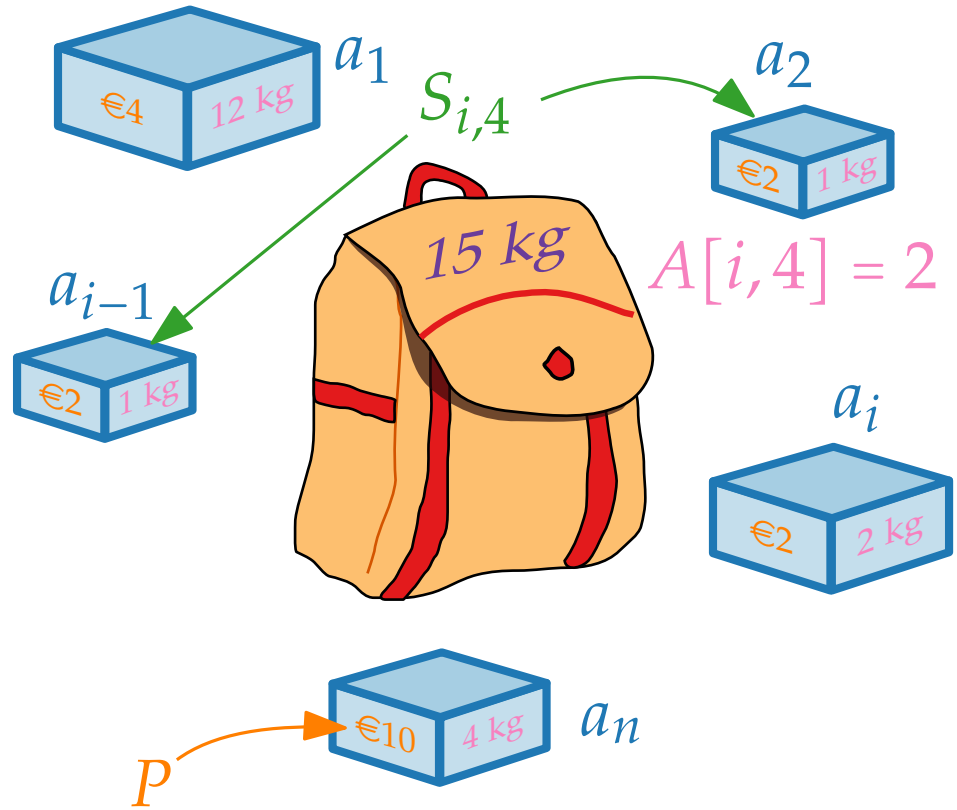
$A[1,p]$ lässt sich für alle $p \in \{0,\ldots,nP\}$ berechnen

Setze $A[i,p] := \infty$ für $p < 0$

$A[i+1,p] = \min\{A[i,p], \text{size}(a_{i+1}) + A[i, p - \text{profit}(a_{i+1})]\}$

$\Rightarrow$ Alle $A[i,p]$-Werte lassen sich in $O(n^2 P)$ Zeit ermitteln

$\Rightarrow$ OPT lässt sich in $O(n^2 P)$ Zeit ermitteln



Pseudopolynomieller Alg. für Rucksack

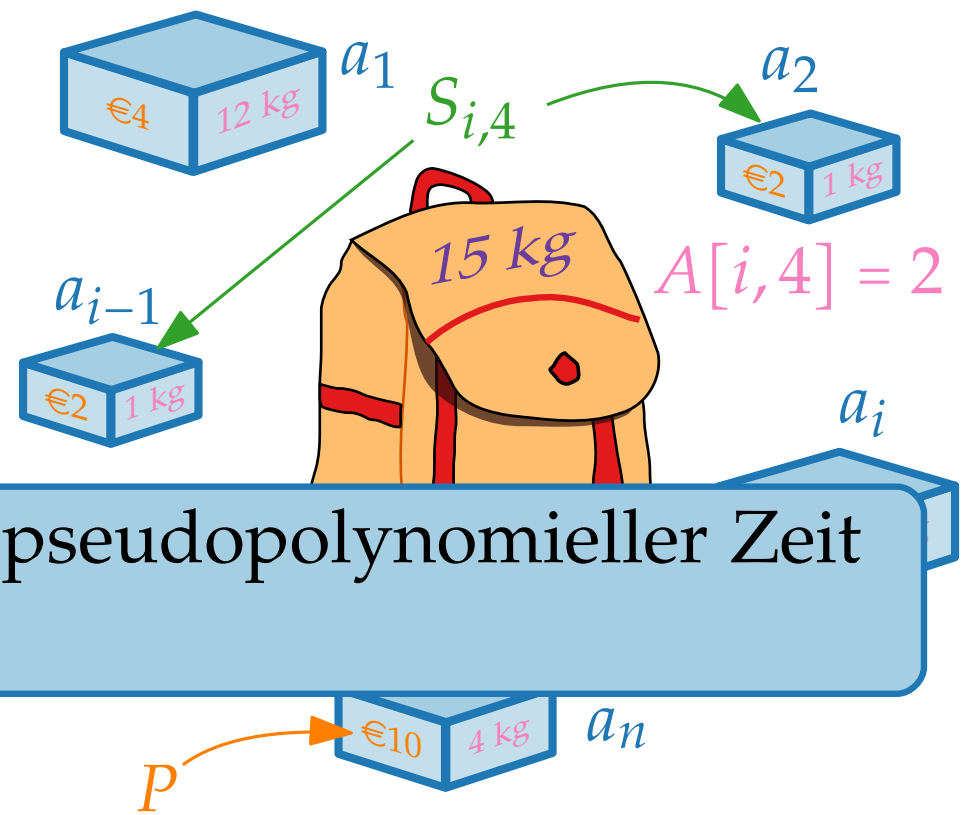
$A[1,p]$ lässt sich für alle $p \in \{0,\ldots,nP\}$ berechnen

Setze $A[i,p] := \infty$ für $p < 0$

$A[i+1,p] = \min\{A[i,p], \text{size}(a_{i+1}) + A[i, p - \text{profit}(a_{i+1})]\}$

$\Rightarrow$ Alle $A[i,p]$-Werte lassen sich in $O(n^2 P)$ Zeit ermitteln

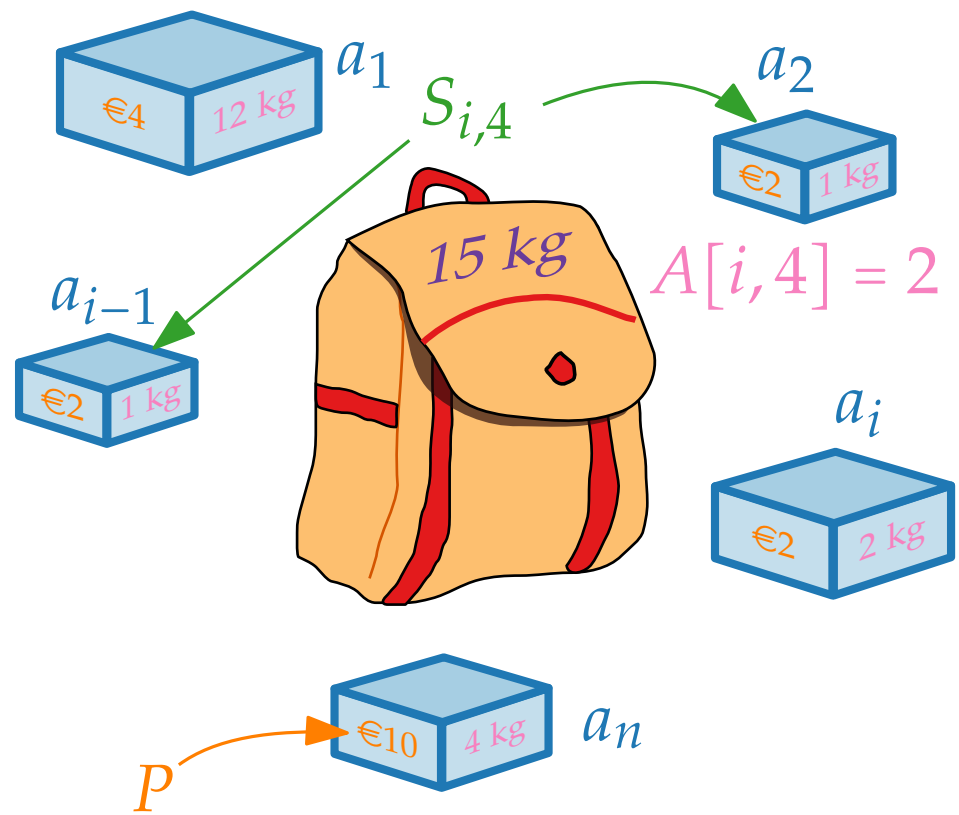
$\Rightarrow$ OPT lässt sich in $O(n^2 P)$ Zeit ermitteln



Satz. Rucksack lässt sich in pseudopolynomieller Zeit $O(n^2 P)$ optimal lösen.

FPTAS für Rucksack durch Skalierung

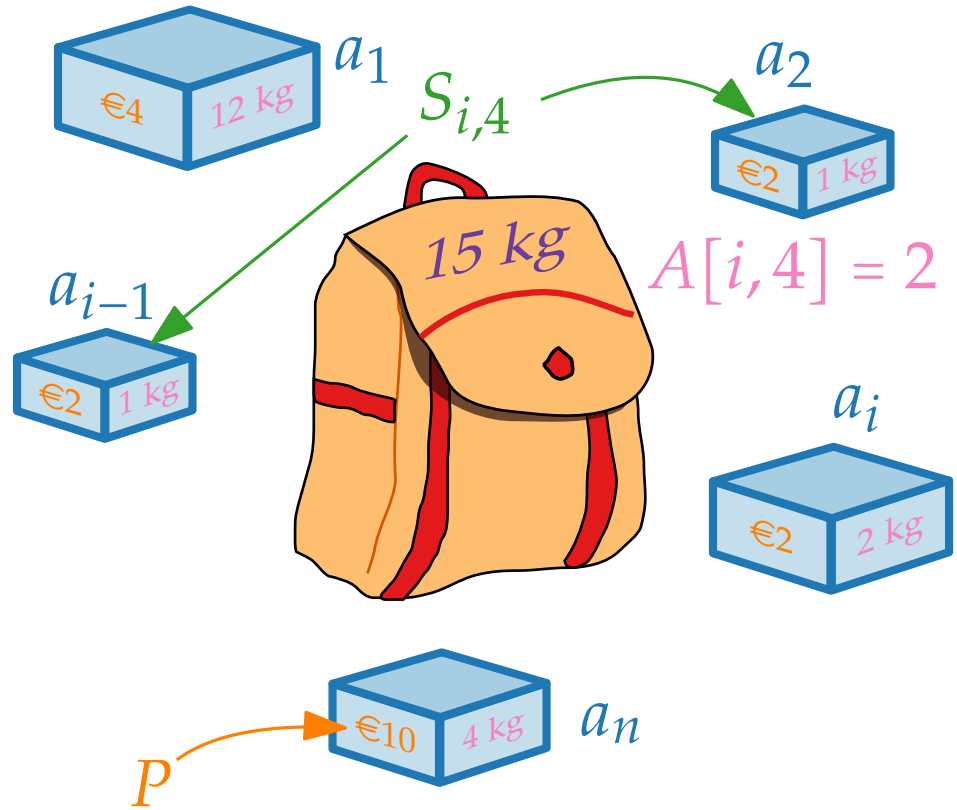
Satz. Rucksack lässt sich in pseudopolynomieller Zeit $O(n^2 P)$ optimal lösen.



FPTAS für Rucksack durch Skalierung

Satz. Rucksack lässt sich in pseudopolynomieller Zeit $O(n^2 P)$ optimal lösen.

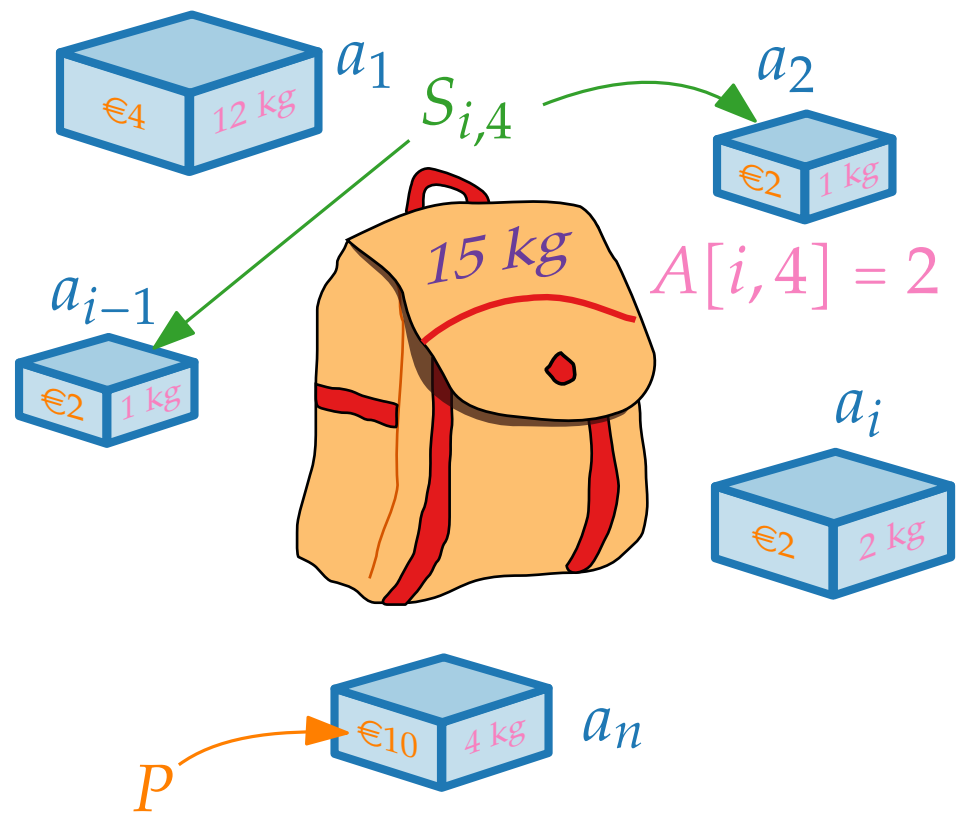
Beispiele. $P = n^5$



FPTAS für Rucksack durch Skalierung

Satz. Rucksack lässt sich in pseudopolynomieller Zeit $O(n^2 P)$ optimal lösen.

Beispiele. $P = n^5 \quad \Rightarrow$ Laufzeit $O(n^7)$

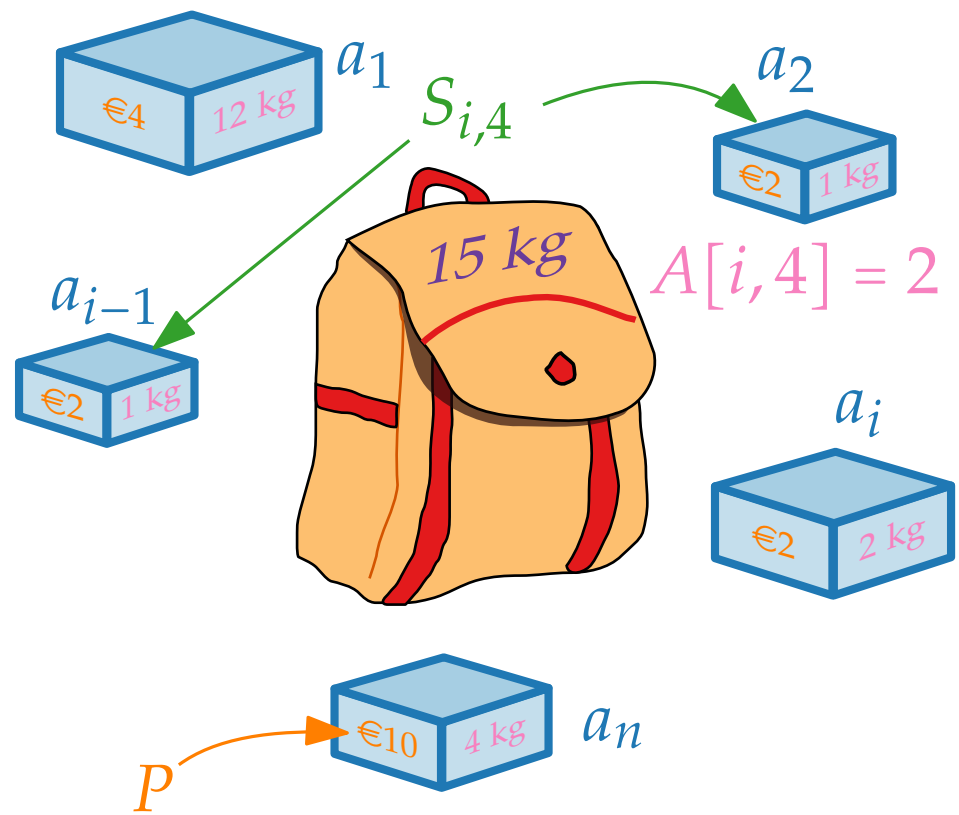


FPTAS für Rucksack durch Skalierung

Satz. Rucksack lässt sich in pseudopolynomieller Zeit $O(n^2 P)$ optimal lösen.

Beispiele. $P = n^5$ $\Rightarrow$ Laufzeit $O(n^7)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n \log n)$



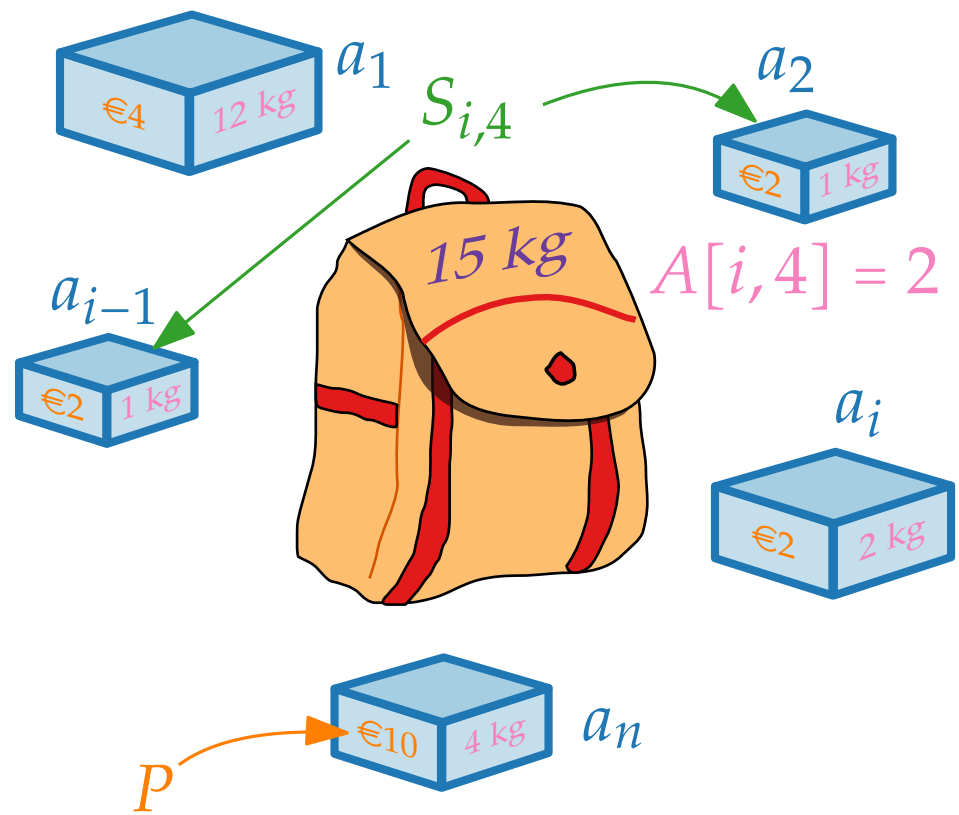
FPTAS für Rucksack durch Skalierung

Satz. Rucksack lässt sich in pseudopolynomieller Zeit $O(n^2 P)$ optimal lösen.

Beispiele. $P = n^5$ $\Rightarrow$ Laufzeit $O(n^7)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$



FPTAS für Rucksack durch Skalierung

Satz. Rucksack lässt sich in pseudopolynomieller Zeit $O(n^2 P)$ optimal lösen.

Beispiele. $P = n^5$ $\Rightarrow$ Laufzeit $O(n^7)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ Laufzeit $O(|I|^7 / \log^7 |I|) = \text{poly}(|I|)$

FPTAS für Rucksack durch Skalierung

Satz. Rucksack lässt sich in pseudopolynomieller Zeit $O(n^2 P)$ optimal lösen.

Beispiele. $P = n^5$ $\Rightarrow$ Laufzeit $O(n^7)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ Laufzeit $O(|I|^7 / \log^7 |I|) = \text{poly}(|I|)$

$P = 2^n$

FPTAS für Rucksack durch Skalierung

Satz. Rucksack lässt sich in pseudopolynomieller Zeit $O(n^2 P)$ optimal lösen.

Beispiele. $P = n^5$ $\Rightarrow$ Laufzeit $O(n^7)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ Laufzeit $O(|I|^7 / \log^7 |I|) = \text{poly}(|I|)$

$P = 2^n$ $\Rightarrow$ Laufzeit $O(n^2 2^n)$

FPTAS für Rucksack durch Skalierung

Satz. Rucksack lässt sich in pseudopolynomieller Zeit $O(n^2 P)$ optimal lösen.

Beispiele. $P = n^5$ $\Rightarrow$ Laufzeit $O(n^7)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ Laufzeit $O(|I|^7 / \log^7 |I|) = \text{poly}(|I|)$

$P = 2^n$ $\Rightarrow$ Laufzeit $O(n^2 2^n)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n^2)$

FPTAS für Rucksack durch Skalierung

Satz. Rucksack lässt sich in pseudopolynomieller Zeit $O(n^2 P)$ optimal lösen.

Beispiele. $P = n^5$ $\Rightarrow$ Laufzeit $O(n^7)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ Laufzeit $O(|I|^7 / \log^7 |I|) = \text{poly}(|I|)$

$P = 2^n$ $\Rightarrow$ Laufzeit $O(n^2 2^n)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n^2)$

$\Rightarrow$ Laufzeit $O(|I| 2^{\sqrt{|I|}}) \neq \text{poly}(|I|)$

FPTAS für Rucksack durch Skalierung

Satz. Rucksack lässt sich in pseudopolynomieller Zeit $O(n^2 P)$ optimal lösen.

Beispiele. $P = n^5$ $\Rightarrow$ Laufzeit $O(n^7)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ Laufzeit $O(|I|^7 / \log^7 |I|) = \text{poly}(|I|)$

$P = 2^n$ $\Rightarrow$ Laufzeit $O(n^2 2^n)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n^2)$

$\Rightarrow$ Laufzeit $O(|I| 2^{\sqrt{|I|}}) \neq \text{poly}(|I|)$

(Un.) Instanzgröße $|I|_{\text{u}} \leq nP = O(n 2^n)$

FPTAS für Rucksack durch Skalierung

Satz. Rucksack lässt sich in pseudopolynomieller Zeit $O(n^2 P)$ optimal lösen.

Beispiele. $P = n^5$ $\Rightarrow$ Laufzeit $O(n^7)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ Laufzeit $O(|I|^7 / \log^7 |I|) = \text{poly}(|I|)$

$P = 2^n$ $\Rightarrow$ Laufzeit $O(n^2 2^n)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n^2)$

$\Rightarrow$ Laufzeit $O(|I| 2^{\sqrt{|I|}}) \neq \text{poly}(|I|)$

(Un.) Instanzgröße $|I|_\text{u} \leq nP = O(n 2^n)$

$\Rightarrow n \in O(\log |I|_\text{u} - \log \log |I|_\text{u})$

FPTAS für Rucksack durch Skalierung

Satz. Rucksack lässt sich in pseudopolynomieller Zeit $O(n^2 P)$ optimal lösen.

Beispiele. $P = n^5$ $\Rightarrow$ Laufzeit $O(n^7)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ Laufzeit $O(|I|^7 / \log^7 |I|) = \text{poly}(|I|)$

$P = 2^n$ $\Rightarrow$ Laufzeit $O(n^2 2^n)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n^2)$

$\Rightarrow$ Laufzeit $O(|I| 2^{\sqrt{|I|}}) \neq \text{poly}(|I|)$

(Un.) Instanzgröße $|I|_u \leq nP = O(n 2^n)$

$\Rightarrow n \in O(\log |I|_u - \log \log |I|_u)$

$\Rightarrow$ Laufzeit $O(|I|_u \log |I|_u) = \text{poly}(|I|_u)$

FPTAS für Rucksack durch Skalierung

Satz. Rucksack lässt sich in pseudopolynomieller Zeit $O(n^2 P)$ optimal lösen.

Beispiele. $P = n^5$ $\Rightarrow$ Laufzeit $O(n^7)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ Laufzeit $O(|I|^7 / \log^7 |I|) = \text{poly}(|I|)$

$P = 2^n$ $\Rightarrow$ Laufzeit $O(n^2 2^n)$

(Bin.) Instanzgröße $|I| \leq n \log P = O(n^2)$

$\Rightarrow$ Laufzeit $O(|I| 2^{\sqrt{|I|}}) \neq \text{poly}(|I|)$

(Un.) Instanzgröße $|I|_u \leq nP = O(n 2^n)$

$\Rightarrow n \in O(\log |I|_u - \log \log |I|_u)$

$\Rightarrow$ Laufzeit $O(|I|_u \log |I|_u) = \text{poly}(|I|_u)$

Beobachte. Laufzeit $O(n^2 P)$ poly in n, wenn P poly in n.

FPTAS für Rucksack durch Skalierung

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) :=$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\leq K \cdot \text{profit}'(o_i) \leq$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - kK \geq$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - kK \geq \text{OPT} - nK =$$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - kK \geq \text{OPT} - nK = \text{OPT} - \varepsilon P$$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - kK \geq \text{OPT} - nK = \text{OPT} - \varepsilon P$$

Beob. 2. $\geq K \cdot \sum_i \text{profit}'(o_i)$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - kK \geq \text{OPT} - nK = \text{OPT} - \varepsilon P$$

Beob. 2. $\geq K \cdot \text{profit}'(S') \geq K \cdot \sum_i \text{profit}'(o_i)$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - kK \geq \text{OPT} - nK = \text{OPT} - \varepsilon P$$

Beob. 2. $\text{profit}(S') \geq K \cdot \text{profit}'(S') \geq K \cdot \sum_i \text{profit}'(o_i)$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - kK \geq \text{OPT} - nK = \text{OPT} - \varepsilon P$

Beob. 2. $\text{profit}(S') \geq K \cdot \text{profit}'(S') \geq K \cdot \sum_i \text{profit}'(o_i)$

$\Rightarrow \text{profit}(S') \geq$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - kK \geq \text{OPT} - nK = \text{OPT} - \varepsilon P$

Beob. 2. $\text{profit}(S') \geq K \cdot \text{profit}'(S') \geq K \cdot \sum_i \text{profit}'(o_i)$

$\Rightarrow \text{profit}(S') \geq \text{OPT} - \varepsilon P \geq$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - kK \geq \text{OPT} - nK = \text{OPT} - \varepsilon P$

Beob. 2. $\text{profit}(S') \geq K \cdot \text{profit}'(S') \geq K \cdot \sum_i \text{profit}'(o_i)$

$\Rightarrow \text{profit}(S') \geq \text{OPT} - \varepsilon P \geq \text{OPT} - \varepsilon\, \text{OPT}$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - kK \geq \text{OPT} - nK = \text{OPT} - \varepsilon P$$

Beob. 2. $\text{profit}(S') \geq K \cdot \text{profit}'(S') \geq K \cdot \sum_i \text{profit}'(o_i)$

$$\Rightarrow \text{profit}(S') \geq \text{OPT} - \varepsilon P \geq \text{OPT} - \varepsilon\,\text{OPT} = (1-\varepsilon) \cdot \text{OPT}$$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - kK \geq \text{OPT} - nK = \text{OPT} - \varepsilon P$

Beob. 2. $\text{profit}(S') \geq K \cdot \text{profit}'(S') \geq K \cdot \sum_i \text{profit}'(o_i)$

$\Rightarrow \text{profit}(S') \geq \text{OPT} - \varepsilon P \geq \text{OPT} - \varepsilon\,\text{OPT} = (1-\varepsilon) \cdot \text{OPT}$ $\square$

FPTAS-Idee: **Skaliere** Profite auf polynomielle Größe (je nach geforderter Fehlerschranke ε).

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - kK \geq \text{OPT} - nK = \text{OPT} - \varepsilon P$

Beob. 2. $\text{profit}(S') \geq K \cdot \text{profit}'(S') \geq K \cdot \sum_i \text{profit}'(o_i)$

$\Rightarrow \text{profit}(S') \geq \text{OPT} - \varepsilon P \geq \text{OPT} - \varepsilon\,\text{OPT} = (1-\varepsilon) \cdot \text{OPT}$ $\square$

Satz. RucksackSkalierung ist ein FPTAS für Rucksack mit Laufzeit $O(\quad)$.

[Ibarra & Kim, '75]

FPTAS für Rucksack durch Skalierung

RucksackSkalierung (I, ε)

$K \leftarrow \varepsilon P/n$ // Skalierungsfaktor

$\text{profit}'(a_i) := \lfloor \text{profit}(a_i)/K \rfloor$

berechne optimale Lösung S' für I bezüglich $\text{profit}'(\cdot)$

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Beweis. Sei $\text{OPT} = \{o_1, \ldots, o_k\}$.

Beob. 1. Für $i = 1, \ldots, k$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - kK \geq \text{OPT} - nK = \text{OPT} - \varepsilon P$

Beob. 2. $\text{profit}(S') \geq K \cdot \text{profit}'(S') \geq K \cdot \sum_i \text{profit}'(o_i)$

$\Rightarrow \text{profit}(S') \geq \text{OPT} - \varepsilon P \geq \text{OPT} - \varepsilon\,\text{OPT} = (1-\varepsilon) \cdot \text{OPT}$ $\square$

Satz. RucksackSkalierung ist ein FPTAS für Rucksack mit Laufzeit $O(n^3/\varepsilon)$.

[Ibarra & Kim, '75]

Starke NP-Schwere

Ein Optimierungsproblem heißt **stark NP-schwer**, wenn es auch mit unärkodierten Zahlen NP-schwer bleibt.

Starke NP-Schwere

Ein Optimierungsproblem heißt **stark NP-schwer**, wenn es auch mit unärkodierten Zahlen NP-schwer bleibt.

Ist das Rucksack Problem stark NP-schwer?

Starke NP-Schwere

Ein Optimierungsproblem heißt **stark NP-schwer**, wenn es auch mit unärkodierten Zahlen NP-schwer bleibt.

Ist das RUCKSACK Problem stark NP-schwer?

Satz. Ein stark NP-schweres Problem besitzt keinen pseudopolynomiellen Algorithmus (außer P = NP).

FPTAS und pseudopoly. Algorithmen

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem

FPTAS und pseudopoly. Algorithmen

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion

FPTAS und pseudopoly. Algorithmen

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ für alle Instanzen I von Π.

FPTAS und pseudopoly. Algorithmen

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ für alle Instanzen I von Π. Wenn Π einen FPTAS besitzt,

FPTAS und pseudopoly. Algorithmen

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ für alle Instanzen I von Π. Wenn Π einen FPTAS besitzt, dann gibt es auch einen pseudopolynomiellen Algorithmus für Π.

FPTAS und pseudopoly. Algorithmen

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ für alle Instanzen I von Π. Wenn Π einen FPTAS besitzt, dann gibt es auch einen pseudopolynomiellen Algorithmus für Π.

Beweis.

Angenommen es gibt einen FPTAS für Π (in $q(|I|, 1/\varepsilon)$ Zeit).

FPTAS und pseudopoly. Algorithmen

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ für alle Instanzen I von Π. Wenn Π einen FPTAS besitzt, dann gibt es auch einen pseudopolynomiellen Algorithmus für Π.

Beweis.

Angenommen es gibt einen FPTAS für Π (in $q(|I|, 1/\varepsilon)$ Zeit).

Setze $\varepsilon =$

FPTAS und pseudopoly. Algorithmen

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ für alle Instanzen I von Π. Wenn Π einen FPTAS besitzt, dann gibt es auch einen pseudopolynomiellen Algorithmus für Π.

Beweis.

Angenommen es gibt einen FPTAS für Π (in $q(|I|, 1/\varepsilon)$ Zeit).

Setze $\varepsilon = 1/p(|I|_{\mathrm{u}})$.

FPTAS und pseudopoly. Algorithmen

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ für alle Instanzen I von Π. Wenn Π einen FPTAS besitzt, dann gibt es auch einen pseudopolynomiellen Algorithmus für Π.

Beweis.

Angenommen es gibt einen FPTAS für Π (in $q(|I|, 1/\varepsilon)$ Zeit).

Setze $\varepsilon = 1/p(|I|_{\mathrm{u}})$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} \;\; <$

FPTAS und pseudopoly. Algorithmen

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ für alle Instanzen I von Π. Wenn Π einen FPTAS besitzt, dann gibt es auch einen pseudopolynomiellen Algorithmus für Π.

Beweis.

Angenommen es gibt einen FPTAS für Π (in $q(|I|, 1/\varepsilon)$ Zeit).

Setze $\varepsilon = 1/p(|I|_{\mathrm{u}})$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} \;<\; \mathrm{OPT} + \varepsilon p(|I|_{\mathrm{u}}) =$

FPTAS und pseudopoly. Algorithmen

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ für alle Instanzen I von Π. Wenn Π einen FPTAS besitzt, dann gibt es auch einen pseudopolynomiellen Algorithmus für Π.

Beweis.

Angenommen es gibt einen FPTAS für Π (in $q(|I|, 1/\varepsilon)$ Zeit).

Setze $\varepsilon = 1/p(|I|_{\mathrm{u}})$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} \;<\; \mathrm{OPT} + \varepsilon p(|I|_{\mathrm{u}}) = \mathrm{OPT} + 1.$

FPTAS und pseudopoly. Algorithmen

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_u)$ für alle Instanzen I von Π. Wenn Π einen FPTAS besitzt, dann gibt es auch einen pseudopolynomiellen Algorithmus für Π.

Beweis.

Angenommen es gibt einen FPTAS für Π (in $q(|I|, 1/\varepsilon)$ Zeit).

Setze $\varepsilon = 1/p(|I|_u)$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} \;<\; \mathrm{OPT} + \varepsilon p(|I|_u) = \mathrm{OPT} + 1.$

$\Rightarrow \mathrm{ALG} = \mathrm{OPT}.$

FPTAS und pseudopoly. Algorithmen

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_u)$ für alle Instanzen I von Π. Wenn Π einen FPTAS besitzt, dann gibt es auch einen pseudopolynomiellen Algorithmus für Π.

Beweis.

Angenommen es gibt einen FPTAS für Π (in $q(|I|, 1/\varepsilon)$ Zeit).

Setze $\varepsilon = 1/p(|I|_u)$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} \;<\; \mathrm{OPT} + \varepsilon p(|I|_u) = \mathrm{OPT} + 1.$

$\Rightarrow \mathrm{ALG} = \mathrm{OPT}.$

Laufzeit:

FPTAS und pseudopoly. Algorithmen

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_u)$ für alle Instanzen I von Π. Wenn Π einen FPTAS besitzt, dann gibt es auch einen pseudopolynomiellen Algorithmus für Π.

Beweis.

Angenommen es gibt einen FPTAS für Π (in $q(|I|, 1/\varepsilon)$ Zeit).

Setze $\varepsilon = 1/p(|I|_u)$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} \;<\; \mathrm{OPT} + \varepsilon p(|I|_u) = \mathrm{OPT} + 1.$

$\Rightarrow \mathrm{ALG} = \mathrm{OPT}.$

Laufzeit: $q(|I|, p(|I|_u))$

FPTAS und pseudopoly. Algorithmen

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_u)$ für alle Instanzen I von Π. Wenn Π einen FPTAS besitzt, dann gibt es auch einen pseudopolynomiellen Algorithmus für Π.

Beweis.

Angenommen es gibt einen FPTAS für Π (in $q(|I|, 1/\varepsilon)$ Zeit).

Setze $\varepsilon = 1/p(|I|_u)$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} \;<\; \mathrm{OPT} + \varepsilon p(|I|_u) = \mathrm{OPT} + 1.$

$\Rightarrow \mathrm{ALG} = \mathrm{OPT}.$

Laufzeit: $q(|I|, p(|I|_u))$, also

FPTAS und pseudopoly. Algorithmen

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_u)$ für alle Instanzen I von Π. Wenn Π einen FPTAS besitzt, dann gibt es auch einen pseudopolynomiellen Algorithmus für Π.

Beweis.

Angenommen es gibt einen FPTAS für Π (in $q(|I|, 1/\varepsilon)$ Zeit).

Setze $\varepsilon = 1/p(|I|_u)$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} \;<\; \mathrm{OPT} + \varepsilon p(|I|_u) = \mathrm{OPT} + 1.$

$\Rightarrow \mathrm{ALG} = \mathrm{OPT}.$

Laufzeit: $q(|I|, p(|I|_u))$, also $\mathrm{poly}(|I|_u)$.

FPTAS und starke NP-Schwere

Satz. Ein stark NP-schweres Problem besitzt keinen pseudopolynomiellen Algorithmus (außer P = NP).

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ für alle Instanzen I von Π. Wenn Π einen FPTAS besitzt, dann gibt es auch einen pseudopolynomiellen Algorithmus für Π.

FPTAS und starke NP-Schwere

Satz. Ein stark NP-schweres Problem besitzt keinen pseudopolynomiellen Algorithmus (außer P = NP).

Satz. Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ für alle Instanzen I von Π. Wenn Π einen FPTAS besitzt, dann gibt es auch einen pseudopolynomiellen Algorithmus für Π.

Korollar. Sei Π ein NP-schweres Optimierungsproblem mit den oben genannten Restriktionen. Wenn Π stark NP-schwer ist, dann gibt es keinen FPTAS für Π (außer P = NP).

FPTAS und starke NP-Schwere

Satz. Ein stark NP-schweres Problem besitzt keinen pseudopolynomiellen Algorithmus (außer P = NP).

Korollar. Sei p ein Polynom und Π ein stark NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ für alle Instanzen I von Π. Dann gibt es keinen FPTAS für Π (außer P = NP).

FPTAS und starke NP-Schwere

Satz. Ein stark NP-schweres Problem besitzt keinen pseudopolynomiellen Algorithmus (außer P = NP).

Korollar. Sei p ein Polynom und Π ein stark NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ für alle Instanzen I von Π. Dann gibt es keinen FPTAS für Π (außer P = NP).

Gibt es ein stark NP-schweres Problem, für das es einen FPTAS gibt?

FPTAS und starke NP-Schwere

Satz. Ein stark NP-schweres Problem besitzt keinen pseudopolynomiellen Algorithmus (außer P = NP).

Korollar. Sei p ein Polynom und Π ein stark NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ für alle Instanzen I von Π. Dann gibt es keinen FPTAS für Π (außer P = NP).

Gibt es ein stark NP-schweres Problem, für das es einen FPTAS gibt?

Ja! Z.B. Rucksack mit *rationalen* Größen und Profiten!

FPTAS und starke NP-Schwere

Satz. Ein stark NP-schweres Problem besitzt keinen pseudopolynomiellen Algorithmus (außer P = NP).

Korollar. Sei p ein Polynom und Π ein stark NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ für alle Instanzen I von Π. Dann gibt es keinen FPTAS für Π (außer P = NP).

Gibt es ein stark NP-schweres Problem, für das es einen FPTAS gibt?

Ja! Z.B. Rucksack mit *rationalen* Größen und Profiten!

Warum?

FPTAS und starke NP-Schwere

Satz. Ein stark NP-schweres Problem besitzt keinen pseudopolynomiellen Algorithmus (außer P = NP).

Korollar. Sei p ein Polynom und Π ein stark NP-schweres Minimierungsproblem mit ganzzahliger Zielfunktion und $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ für alle Instanzen I von Π. Dann gibt es keinen FPTAS für Π (außer P = NP).

Gibt es ein stark NP-schweres Problem, für das es einen FPTAS gibt?

Ja! Z.B. Rucksack mit *rationalen* Größen und Profiten!

Warum?

Müssten um Produkt der Nenner skalieren, um ganzzahlig zu werden.