

# Rucksack Problem

## Algorithmen und Datenstrukturen II

---

Sebastian Baumann, Korbinian Karl, Ehsan Moslehi

June 23, 2019

Hochschule für Angewandte Wissenschaften München

1. Beschreibung des Problems
2. Lösungsansätze

# Beschreibung des Problems

---

# Rucksack Problem



## Gegeben:

- Gegenstände  $1, 2, 3, \dots, n$ 
  - $w_i$  : Wert vom Gegenstand  $i$
  - $v_i \in \mathbb{N}$  : Volumen vom Gegenstand  $i$
- Rucksack mit dem Volumen  $V \in \mathbb{N}$

## Gesucht:

Eine Rucksackfüllung mit maximalen Gesamtwert, wobei das Volumen  $V$  nicht überschritten werden darf.

$$\max \left\{ \sum_{i=1}^n w_i t_i \mid \sum_{i=1}^n v_i t_i \leq V, \forall i : t_i \in \{0, 1\} \right\}$$

Ganzzahliges Lineares Optimierungsproblem

Ganzzahliges Lineares Optimierungsproblem

**NP-Vollständig**

# Lösungsansätze

---

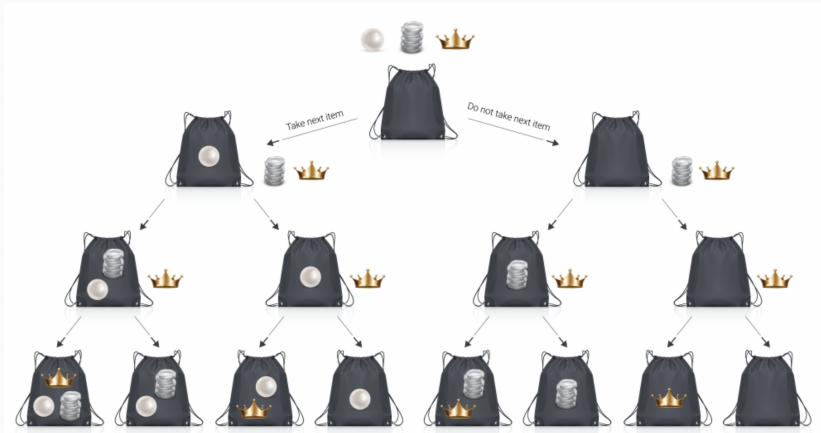


1. Brute Force
2. Greedy
3. Dynamische Programmierung

# Brute Force

---

# Brute Force



**Figure 2:** Probiere alle Teilmengen!

Optimale globale Lösung wird gefunden.

Optimale globale Lösung wird gefunden.

Exponentielle Laufzeit  $O(2^n)$

# Greedy Algorithmus

---

**Strategien:**

1. **Absteigende Sortierung nach Wert**

**Packe solange Gegenstände in den Rucksack, bis kein Gegenstand mehr rein passt!**

**Strategien:**

1. **Absteigende Sortierung nach Wert**
2. **Aufsteigende Sortierung nach Volumen**

**Packe solange Gegenstände in den Rucksack, bis kein Gegenstand mehr rein passt!**



## Strategien:

1. Absteigende Sortierung nach Wert
2. Aufsteigende Sortierung nach Volumen
3. Absteigende Sortierung nach Wertdichte  $d_i = \frac{w_i}{v_i}$

Packe solange Gegenstände in den Rucksack, bis kein Gegenstand mehr rein passt!

Optimale globale Lösung wird **NICHT** gefunden.  
Optimale lokale Lösung wird gefunden.

Optimale globale Lösung wird **NICHT** gefunden.  
Optimale lokale Lösung wird gefunden.

Laufzeit  $O(n \cdot \log n)$

# Dynamische Programmierung

---

1. Teile das Problem in gleichartige Teilprobleme!

1. Teile das Problem in gleichartige Teilprobleme!
2. Speichere die Resultate der Teilprobleme systematisch, um wiederholte Berechnungen zu vermeiden!

1. Teile das Problem in gleichartige Teilprobleme!
2. Speichere die Resultate der Teilprobleme systematisch, um wiederholte Berechnungen zu vermeiden!
3. Setze die optimale Lösung aus den optimalen Lösungen der Teilprobleme zusammen!

## Idee:

1. Löse das Problem für eine Menge von Gegenständen mit einem Gegenstand und einem Rucksack mit Volumen eins. Speichere das Ergebnis in einer Tabelle.



## Idee:

1. Löse das Problem für eine Menge von Gegenständen mit einem Gegenstand und einem Rucksack mit Volumen eins. Speichere das Ergebnis in einer Tabelle.
2. Wiederhole das für verschiedene Rucksäcke mit volumen  $2, \dots, V$  und benutze dabei die vorherige Ergebnisse.

## Idee:

1. Löse das Problem für eine Menge von Gegenständen mit einem Gegenstand und einem Rucksack mit Volumen eins. Speichere das Ergebnis in einer Tabelle.
2. Wiederhole das für verschiedene Rucksäcke mit volumen 2, ...,  $V$  und benutze dabei die vorherige Ergebnisse.
3. Wiederhole die vorherigen Schritte für Mengen mit zwei, drei, ...,  $n$  Gegenständen.

# Dynamische Programmierung

```
1  for i := 1; i <= numItems; i++ {
2      for j := 1; j <= capacity; j++ {
3          if is[i-1].volume <= j {
4              valueOne := float64(matrix[i-1][j])
5              valueTwo := float64(is[i-1].worth + matrix[i-1][j-
is[i-1].volume])
6              matrix[i][j] = int(math.Max(valueOne, valueTwo))
7          } else {
8              matrix[i][j] = matrix[i-1][j]
9          }
10     }
11 }
12
```

## Beispiel:

Rucksack mit Volumen 5.

Gegenstände :

1.  $v_1 = 3, w_1 = 5$
2.  $v_2 = 2, w_2 = 3$
3.  $v_3 = 1, w_3 = 4$

**Fülle die Tabelle!**

<div>V: I:</div>	1	2	3	4	5
1 (3, 5)	0	0	5	5	5
2 (2, 3)	0	3	5	5	8
3 (1, 4)	4	4	7	9	9

# Dynamische Programmierung

- Fange unten Rechts an.
- Vergleiche den Wert mit dem Wert in der Zelle drüber.
  1. Die Werte sind gleich  $\Rightarrow$  Der Gegenstand ist nicht genommen.
  2. Die Werte sind nicht gleich  $\Rightarrow$  Der Gegenstand genommen. So gehe eine Zeile hoch und um Volumen vom Gegenstand nach links.
- Wiederhole bis der Zeiger aus der Tabelle raus geht.

<div>V: I:</div>	1	2	3	4	5
1 (3, 5)	0	0	5	5	5
2 (2, 3)	0	3	5	5	8
3 (1, 4)	4	4	7	9	9

**Fragen?**