Westfälische Hochschule

Rucksackproblem

Simon Hünting

Algorithmen und Datenstrukturen

Prof. Dr. Manfred Meyer

Inhaltsverzeichnis

Westfälische Hochschule

Definition

- Kombinatorisches Optimierungsproblem
  - ➢Nicht die eine Lösung, sondern die Beste
  - ➢Nicht alle Lösungen durchprobieren

- Menge von n Objekten mit Gewicht w und Nutzen v
- Maximale Tragkraft T, darf nicht überschritten werden

→Auswahl von Objekten S mit

1. $\sum_{i \in s} \omega[i] \leq T$

2. $\sum_{i \in s} v[i]$ ist maximal

Beispiel: 7 vs. Wild

Westfälische Hochschule

Objekt s	Gewicht w	Nutzen v
	2 KG	7
	1 KG	4
	9 KG	2
	7 KG	3
	5 KG	5

?

Lösungsansatz: Erschöpfende Suche

- Algorithmische Prinzip der rohen Gewalt
- Alle Optionen werden durchprobiert

algorithm ES(w[1…n],v[1…n],T)
G←0 *//maximaler Gesamtwert*
for each $S \subseteq \{1, \dots, n\}$
    weight←$\sum_{i \in S} w[i]$
    value← $\sum_{i \in S} v[i]$
    if weight≤T *and* value>G
        G←value
return G

Laufzeitkomplexität:

$$o(2^n \cdot n)$$

Beispiel: 7 vs. Wild

Westfälische Hochschule

Objekt s	Gewicht w	Nutzen v
	2 KG	7
	1 KG	4
	9 KG	2
	7 KG	3
	5 KG	5

?

Optimierte Lösung: Branch and Bound

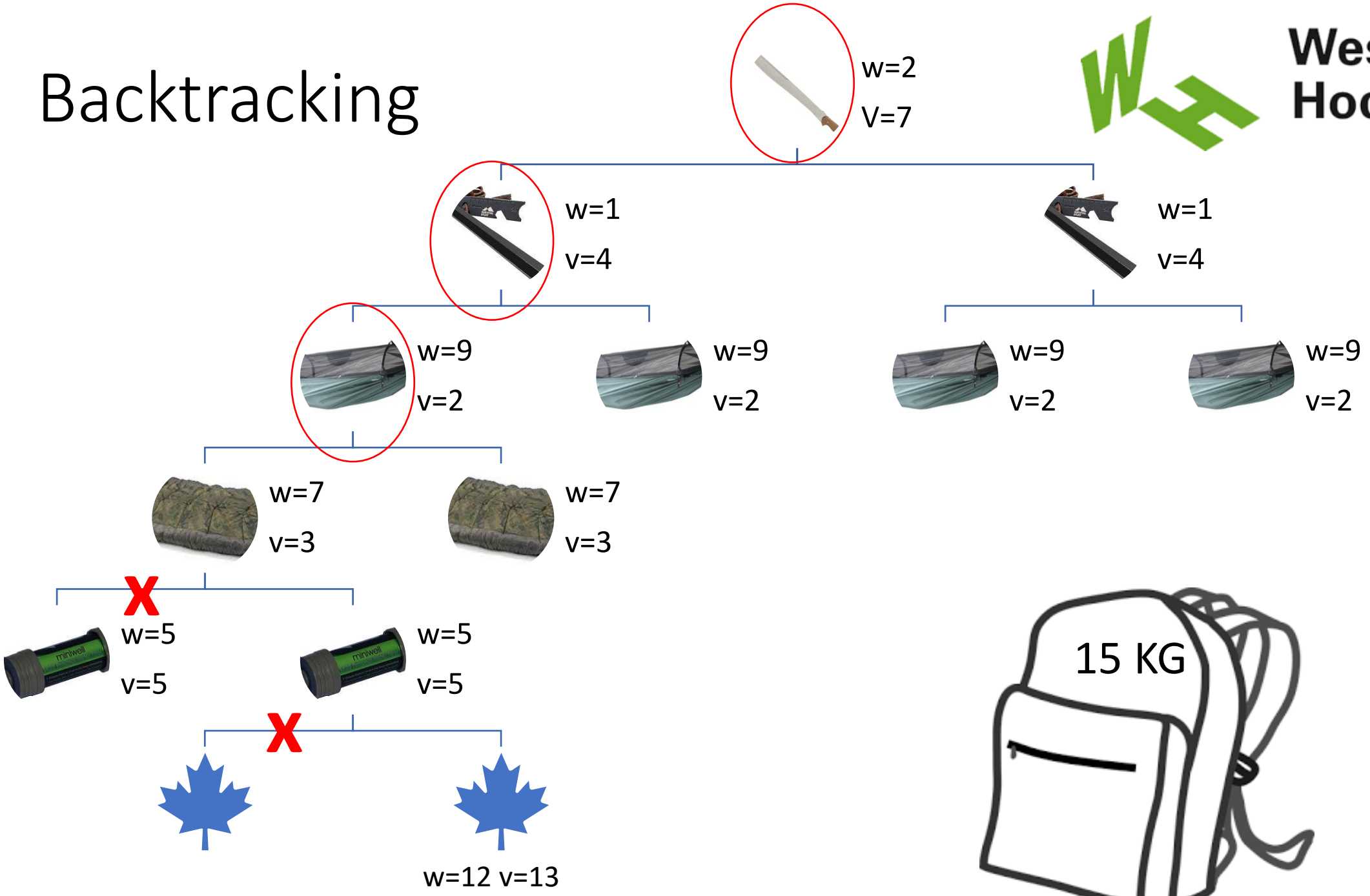
Westfälische Hochschule

- „Verzweigen und Begrenzen“
- Sicherstellen, dass optimale Lösung gefunden wird (Optimierungsproblem)
- Optimal: Zielfunktion wird maximiert/minimiert hinsichtlich Nebenbedingungen
- Führt auf Entscheidungsbaum zurück
- Backtracking: Äste werden „abgeschnitten“(*pruning*), falls keine optimale Lösung vorliegt
- Begrenzen: Restvalue angucken

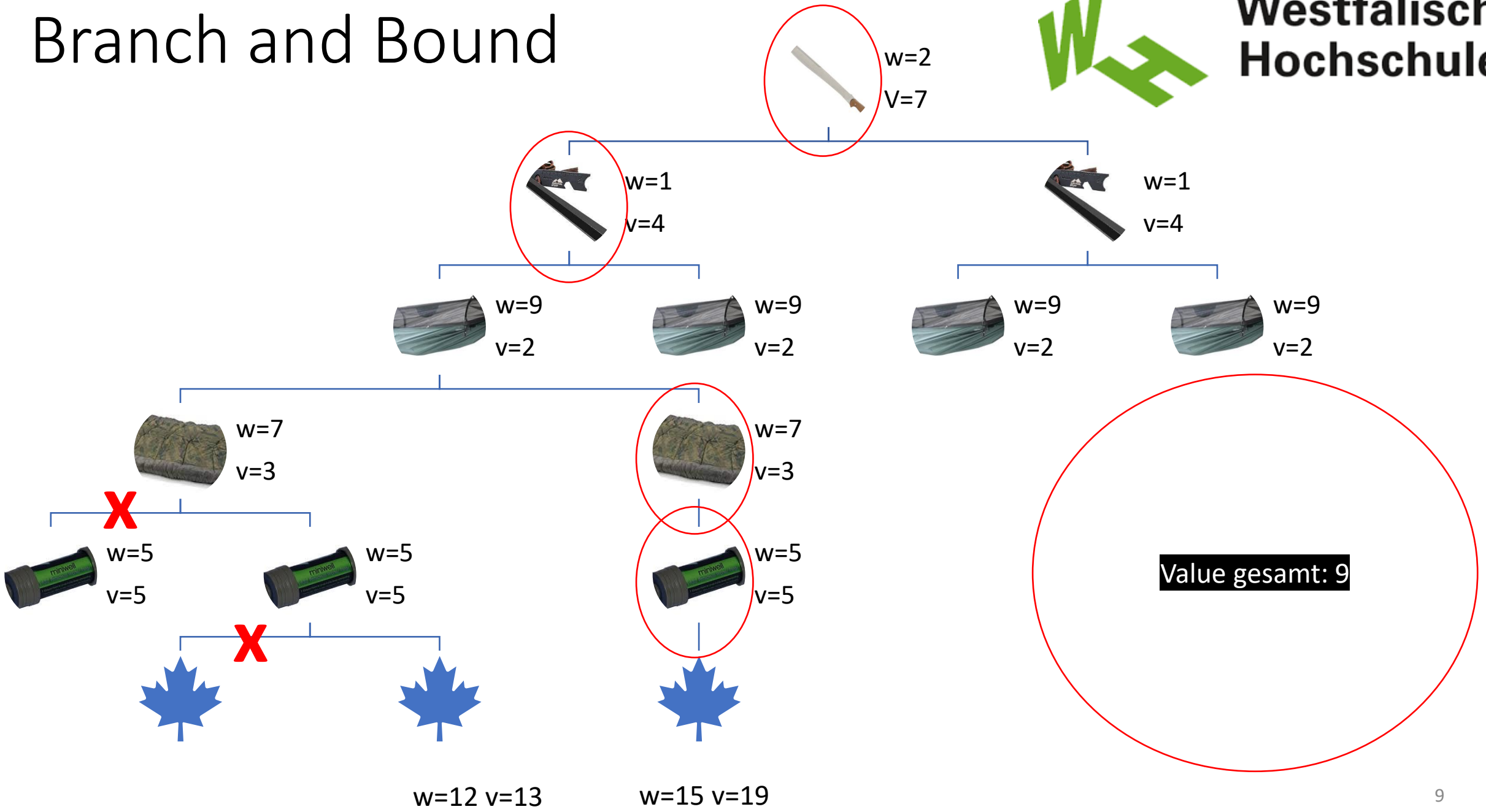
Laufzeitkomplexität:

$$o(2^n)$$

Backtracking



Branch and Bound



Praktikumsaufgabe

Westfälische Hochschule

- Rakete startet zur Raumstation
- Maximallast T von 645 KG
- Liste von Objekten mit Objekt Nummer, Gewicht in KG und Profit in €

→Optimale Lösung

→Implementierung in Java

Westfälische Hochschule

class Item

- Erstellen der Klasse mit Datentypen
- Konstruktor für Items
- getter/setter
- Methode zur Ausgabe der Integer

```
13 class Item {
14
15     private int objectNr;  //@param objectNr -> Objektnummer für Weltraummaterial
16     private int value;     //@param value -> Profit in 1000€
17     private int weight;    //@param weight -> Gewicht in Kg
18
19     //Konstruktor mit Methoden um einzelne Items aufzurufen
20     public Item(int objectNr, int weight, int value) {
27
28     int getValue() {
29         return value;
30     }
31
32     int getWeight() {
33         return weight;
34     }
35     //@return wirft Itemdetails zurück
36     public String toString() {
39 }
```

Statische Attribute & method knapsack

Westfälische Hochschule

- Statische Werte für beste Items und value
- Methode mit Parametern

```
static int bestValue;
static List<Item> bestSelection;


 * Operation knapsack, zum Herausfinden des bestmöglichen Values unter maximalem Gewicht
public static void knapsack(List<Item> selectedItems,
                            int selectedValue,
                            List<Item> availableItems,
                            int availableValue,
                            int currentWeight,
                            int weightLimit)
```

Methode knapsack

- Beruht auf backtracking Ansatz
- Verarbeitung aller Items und neues Optimum
  →statische Attribute werden aktualisiert
- Kann neuer Gegenstand in aktuelle Liste aufgenommen werden?
  →Beschränkungen eingehalten?
  →Optimum verbessert?
- Add, Backtracking oder nächster Gegenstand aus availableItems

```
if (availableItems.isEmpty())        //alle Items sind verarbeitet, Ergebnis wird als Bound/Messlatte gemerkt
{
    if (selectedValue > bestValue)
    {
        bestValue = selectedValue;
        bestSelection.clear();
        bestSelection.addAll(selectedItems);
    }
} // Verschiebung des Else-Parts auf äußeres IF, added by me
else
    {
        Item item = availableItems.remove(0);
        int weight = item.getWeight();
        int value = item.getValue();
        if (selectedValue + availableValue > bestValue)
            {
            //neues Optimum nicht möglich, Item wird genommen und weiter nach verfügbaren Items geguckt
            if (currentWeight + weight <= weightLimit)
                {
                selectedItems.add(item);
                knapsack(selectedItems, selectedValue+value,
                         availableItems, availableValue-value,
                         currentWeight+weight, weightLimit);
                selectedItems.remove(item);  //Entfernen des Items
                }
            knapsack(selectedItems, selectedValue,
                     availableItems, availableValue,
                     currentWeight, weightLimit);
            }
        availableItems.add(item);  //verfügbare Items zu Liste hinzufügen
      }
  }
```

Items aus Praktikum

- Erstellen und Befüllen der Liste

```
98  public static void main(String[] args) {
99
100     List<Item> items = new ArrayList<Item>();
101     bestSelection = new ArrayList<Item>();
102
103     //@param items Liste mit Wert aus Aufgabenstellung befüllen
104     items.add(new Item(1,201,191));
105     items.add(new Item(2,141,239));
106     items.add(new Item(3,50,66));
107     items.add(new Item(4,38,249));
108     items.add(new Item(5,79,137));
109     items.add(new Item(6,73,54));
110     items.add(new Item(7,232,153));
111     items.add(new Item(8,48,148));
112
```

Westfälische Hochschule

Main Class

- Setzen der Rahmenbedingungen
- Ausgabe optimale Packliste
- Ausgabe Werte

```
114     bestValue = 0;
115     int availableValue = 0;
116       for(Item item : items)
117           availableValue += item.getValue();
118
119     knapsack(new ArrayList<Item>(),0,items,availableValue,0,645);
120     System.out.println("Die optimale Packliste:");
121     int value = 0;
122     int weight = 0;
123     //optimale Packliste bestimmen
124     for (Item item : bestSelection)
125     {
126         System.out.println(item);
127         value += item.getValue();
128         weight += item.getWeight();
129     }
130     System.out.println("Wert: "+value+" Gewicht: "+weight);
```

Lösung der Aufgabe

Westfälische Hochschule

```
Die optimale Packliste:
1 191 201
2 239 141
3 66 50
4 249 38
5 137 79
6 54 73
8 148 48
Wert: 1084 Gewicht: 630
```