Rucksack packen mit TS

Vom greedy Verfahren
zur Tiefensuche

beim einfachen
Rucksackproblem

(Pythonversion, flache Liste)

Rucksack packen mit TS

- Ausgangspunkt:

30	30	30	30
20	20	20	20

Eine Stückeliste, aus der Stücke in einen Behälter (Rucksack, Container) gegebener Größe gefüllt werden sollen.

```
stuecke=[30, 30, 30, 30, 20, 20, 20, 20]
container= [80]
```

Rucksack packen mit TS

- Stand **`fuelle`** (greedy) Version:

```
def fuelle(stuecke, container):
    if ist_voll(container):
        return container
    elif len(stuecke)==0:
        return False
    elif wird_zu_voll(stuecke[0],container):
        return fuelle(stuecke[1:], container)
    else:
        return fuelle(stuecke[1:],
                      fuelle_ein(stuecke[0], container))
```

eine Bedingung

zugehöriger ja-Fall

else-Fall am Ende

Mehrfachverzweigung

Rucksack packen mit TS

- Hilfsfunktionen **fuelle_ein**

```
def fuelle_ein(stueck, container):
    inhalt=[]+container[1]
    # echte Kopie erzeugen!
    inhalt.append(stueck)
    return [container[0], inhalt]
```

Rucksack packen mit TS

- Hilfsfunktionen **`ist_voll`** und **`wird_zu_voll`**:

```
def ist_voll(container):
    return container[0]==sum(container[1])

def wird_zu_voll(stueck, container):
    return container[0]<
                        stueck+sum(container[1])
```

Rucksack packen mit TS

- Stand **`fuelle`** (greedy) Version:

```
def fuelle(stuecke, container):
    if ist_voll(container):
        return container
    elif len(stuecke)==0:
        return False
    elif wird_zu_voll(stuecke[0],container):
        return fuelle(stuecke[1:], container)
    else:
        return fuelle(stuecke[1:],
                      fuelle_ein(stuecke[0], container))
```

Erfolgsfall

Misserfolgsfall

Einfüllen verhindern

Rekursionsschritt

Rucksack packen mit TS

- Problem:

```
fuelle(stuecke, [100, []])
```

liefert `False`

- Obwohl das Problem mit den gegebenen Stücken durchaus lösbar wäre. Wir wollen als Lösung bekommen:

```
fuelle(stuecke, [100, []])
```

liefert `[100, [30, 30, 20, 20]]`

Rucksack packen mit TS

- Grund:

  Das dritte 30-er-Stück kann eingefüllt werden, verhindert aber die mögliche Füllung mit zwei 20-er-Stücken.

- Lösungsidee:

  Wir müssen zu einem vorigen Zustand zurück gehen können und von dort aus die Alternativen prüfen.

  → Idee des *backtracking*

Rucksack packen mit TS

- Lösungsansatz:

  Der Rekursionsschritt in die Tiefe darf nur bedingt erfolgen.

- „*Normale*“ Realisierung wäre eine interne Verzweigung im bisherigen else-Fall.

Rucksack packen mit TS

- Erweiterung **`fuelle`** Tiefensuche-Version:

```
def fuelle(stuecke, container):
    if ist_voll(container):
        return container
    elif len(stuecke)==0:
        return False
    elif wird_zu_voll(stuecke[0], container):
        return False
    else:
        ergebnis=fuelle(stuecke[1:],
                        fuelle_ein(stuecke[0], container))
        if ergebnis:
            return ergebnis
        else:
            return fuelle(stuecke[1:], container)
```

Erfolgsfall

Misserfolgsfall

Einfüllen verhindern

Versuch des Rekursionsschritts

Rucksack packen mit TS

- Lösungsansatz:

  Der Rekursionsschritt in die Tiefe darf nur bedingt erfolgen.

- *„Normale“* Realisierung wäre eine interne Verzweigung im bisherigen else-Fall.

- Bei Python muss die Logik nicht voll in der Mehrfachverzweigung umgesetzt werden, da die Funktion immer nach einem `return` einen Rücksprung auslöst. Daher lässt sich das Gewollte auch in einer nachfolgenden gesonderten Verzweigung realisieren.

Rucksack packen mit TS

- Alternative **`fuelle`** Tiefensuche-Version:

```
def fuelle(stuecke, container):
    if ist_voll(container):
        return container
    if len(stuecke)==0:
        return False
    if wird_zu_voll(stuecke[0], container):
        return False
    ergebnis=fuelle(stuecke[1:],
                    fuelle_ein(stuecke[0], container))
    if ergebnis:
        return ergebnis
    else:
        return fuelle(stuecke[1:], container)
```

Erfolgsfall

Misserfolgsfall

Einfüllen verhindern

Versuch des Rekursionsschritts

Rucksack packen mit TS

Ablauf:

- nicht exakt voll, noch Stücke da, nicht zu voll
  → erstes 30-er-Stück wird versucht
- nicht exakt voll, noch Stücke da, nicht zu voll
  → zweites 30-er-Stück wird versucht
- nicht exakt voll, noch Stücke da, nicht zu voll
  → drittes 30-er-Stück wird versucht
- nicht exakt voll, noch Stücke da, aber zu voll
  → viertes 30-er-Stück wird verworfen
- nicht exakt voll, noch Stücke da, aber zu voll
  → erstes 20-er-Stück wird verworfen

bis

- nicht exakt voll, keine Stücke mehr da
  → #f zurück geben über vier Stufen und mit weiteren 20-er Versuchen
- else ohne das dritte 30-er-Stück versuchen
- nicht exakt voll, noch Stücke da, nicht zu voll
  → erstes 20-er-Stück wird versucht
- nicht exakt voll, noch Stücke da, nicht zu voll
  → zweites 20-er-Stück wird versucht
- exakt voll
  → (100 20 20 30 30) zurück geben