



DHBW

Duale Hochschule
Baden-Württemberg

Lörrach

GREEDY ALGORITHMS

Präsentation am 26. Mai 2023 erarbeitet von Gruppe 2, WDS/WWI22A:

*Mihabat Aeido, Samuel Butler, Tjark Gerken, Eric Harter, Jacob Ruhnau, Tom Warscheit
im Rahmen der Vorlesung „Algorithmen und Datenstrukturen“ bei Max Bergau*

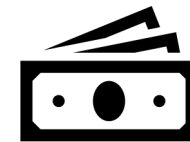
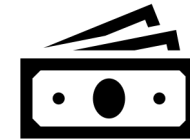
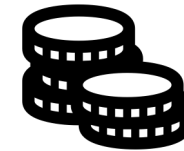
GREEDY ALGORITHMS



**Kunde möchte 40€
abheben**

**Zur Verfügung stehen
25€ 20€ 10€ und 5€ Noten**

*Gewünscht ist die
geringstmögliche
Anzahl an Scheinen*



AGENDA

I. Grundlagen

- Prinzipien
- Vor- / Nachteile

II. Beispiele, Probleme & Analyse

- Longest Path
- Traveling Sales Man
- Making Change

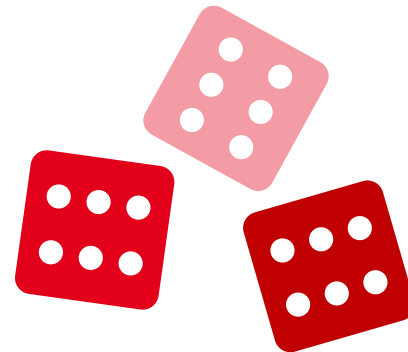
III. Fazit

PRINZIPIEN

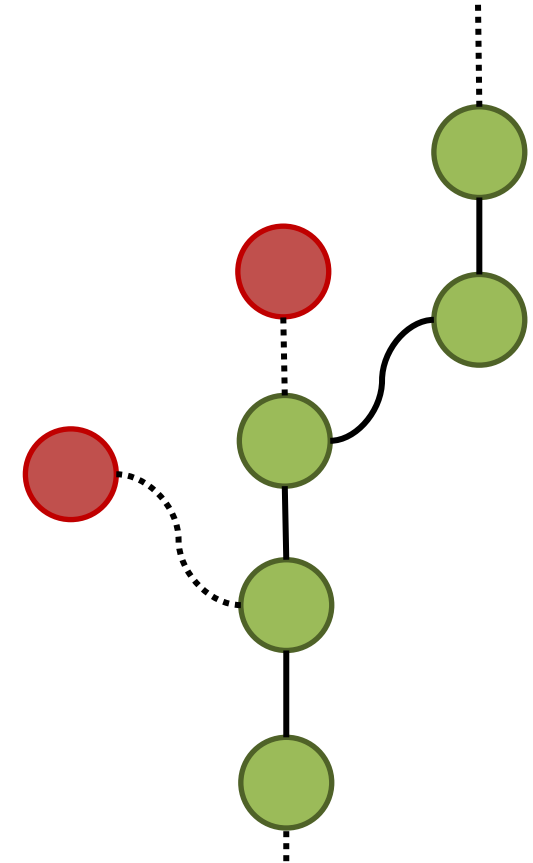
greedy ['gri:di] *engl.; gierig, raffsüchtig*



Divide and Conquer
/ Betrachtung des
lokalen Optimums



Randomisierung &
Wahrscheinlichkeiten



Backtracking

VOR- UND NACHTEILE



**Einfache
Implementierung**

**Time & Space
Complexity**

**Näherungs-
lösungen**

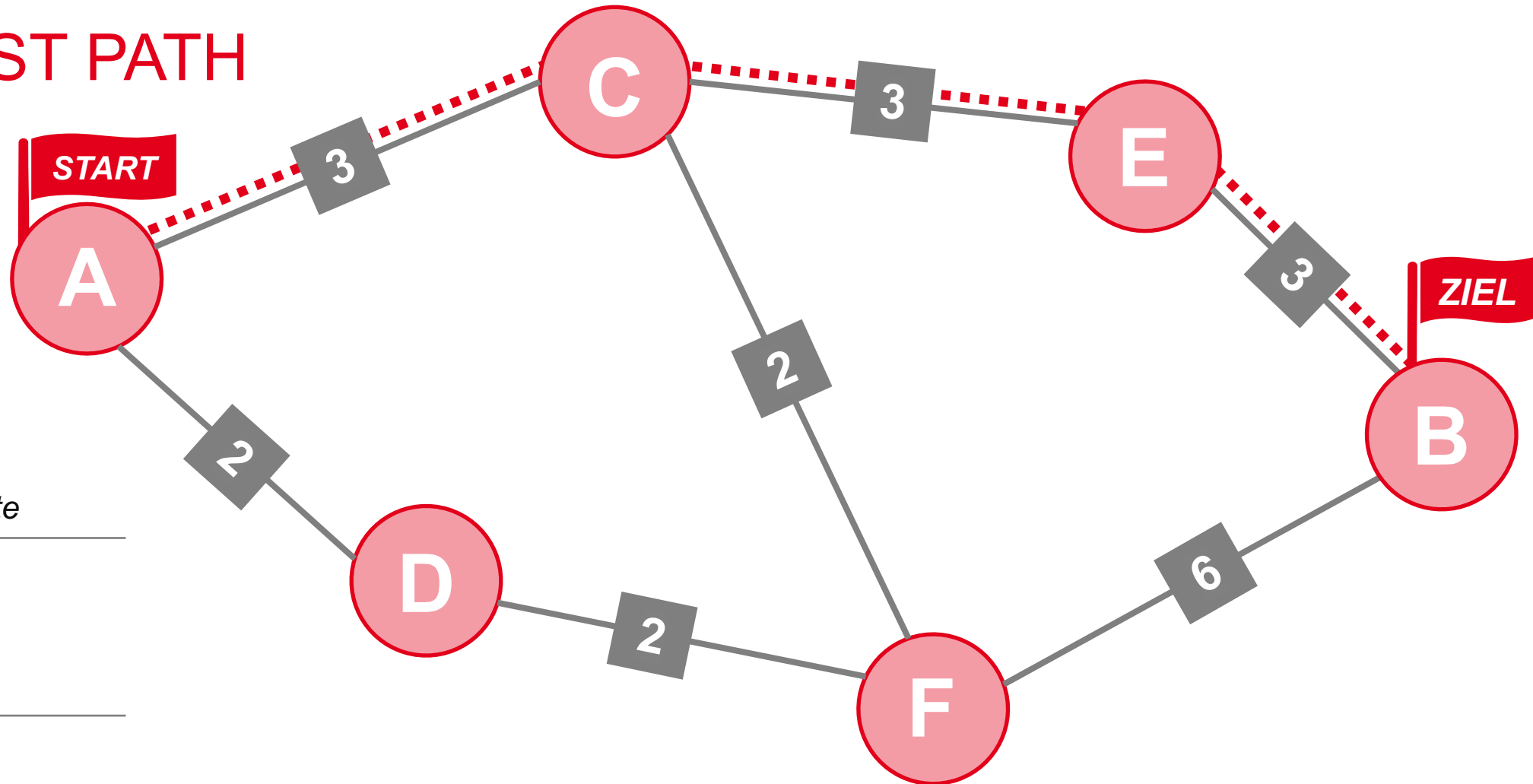


**Blickt nicht
zurück**

**Kennt kein
globales Optimum**

**Schwierigkeiten
bei komplexen
Problemen**

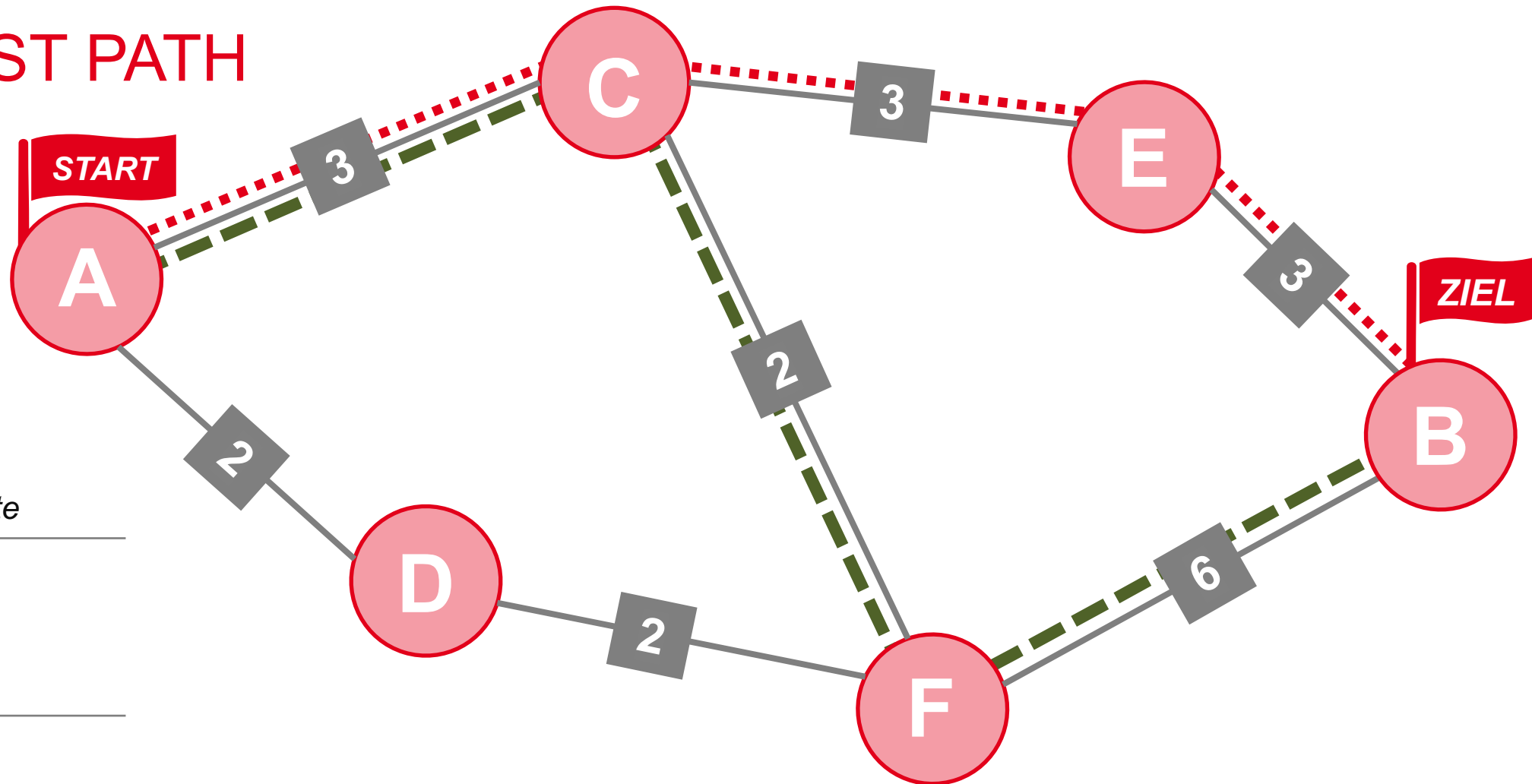
LONGEST PATH



Strecke	Punkte
A → C	3
C → E	3
E → B	3
Gesamt	9

→ Nicht optimale Lösung

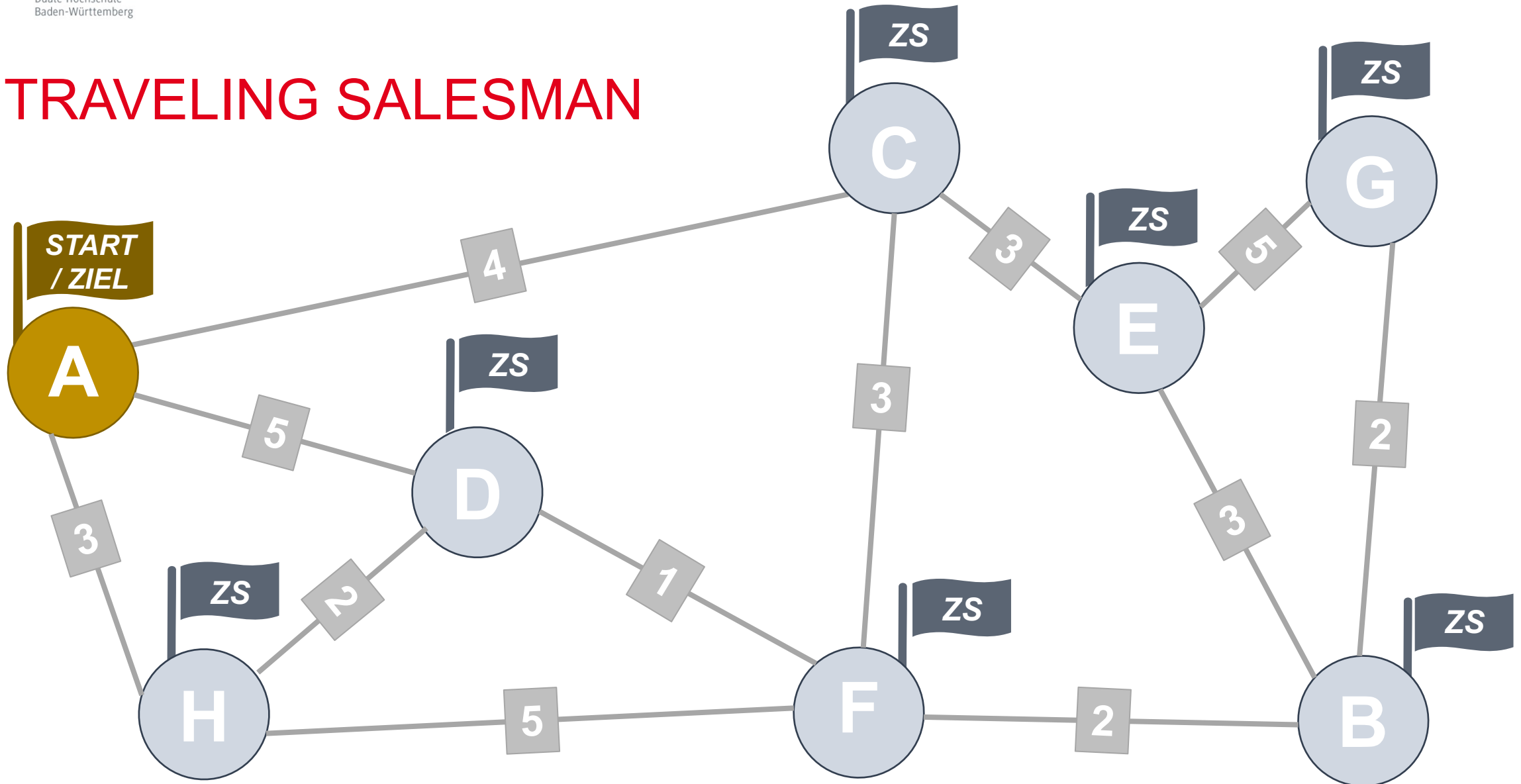
LONGEST PATH



Strecke	Punkte
A→C	3
C→F	2
F→B	6
Gesamt	11

→ Optimale Lösung

TRAVELING SALESMAN



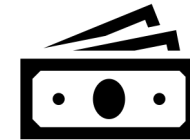
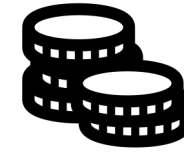
MAKING CHANGE



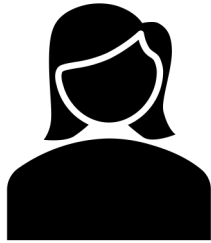
**Kunde möchte 40€
abheben**

**Zur Verfügung stehen
25€ 20€ und 5€ Noten**

*Gewünscht ist die
geringstmögliche
Anzahl an Scheinen*



HERANGEHENSWEISE



Menschen

Bei kleineren Mengen können wir schnell intuitiv
Erkennung

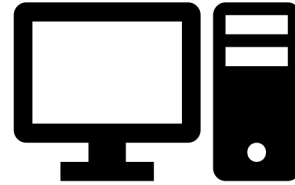
Wie würdet Ihr bei größeren Mengen vorgehen?



Greedy Change Making Algorithm

Es wird immer die höchste noch mögliche Denomination
herausgegeben.

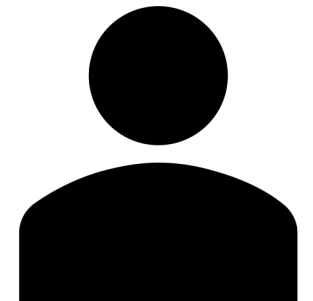
STEP BY STEP



Greedy Change Making Algorithm



Restsumme: **40€**



25€ 20€ 5€

STEP BY STEP

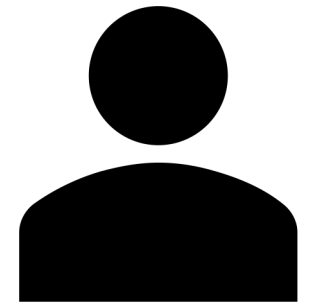


Greedy Change Making Algorithm



25€ 20€ 5€

Restsumme: **0€**



5€ 5€ 5€ 25€

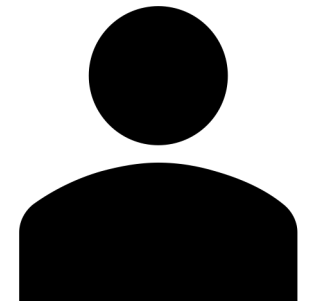
STEP BY STEP



Greedy Change Making Algorithm

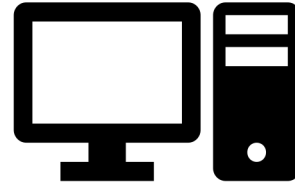


Restsumme: **45€**



25€ 20€ 5€

STEP BY STEP

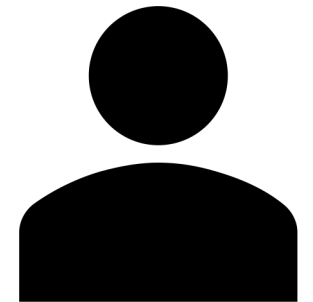


Greedy Change Making Algorithm



25€ 20€ 5€

Restsumme: **0€**



20€ 25€

greedy_algos.ipynb — 22-2_alg_ry-data

greedy_algos.ipynb M • helper_methods.py

06-presentations > greedy_algos.ipynb > M*Implementation algorithms to make change

+ Code + Markdown | ▶ Alle ausführen ✕ Alle Ausgaben löschen ↺ Neu starten [x] Variablen ≡ Gliederung ...

home (Python 3.10.11)

Implementation algorithms to make change

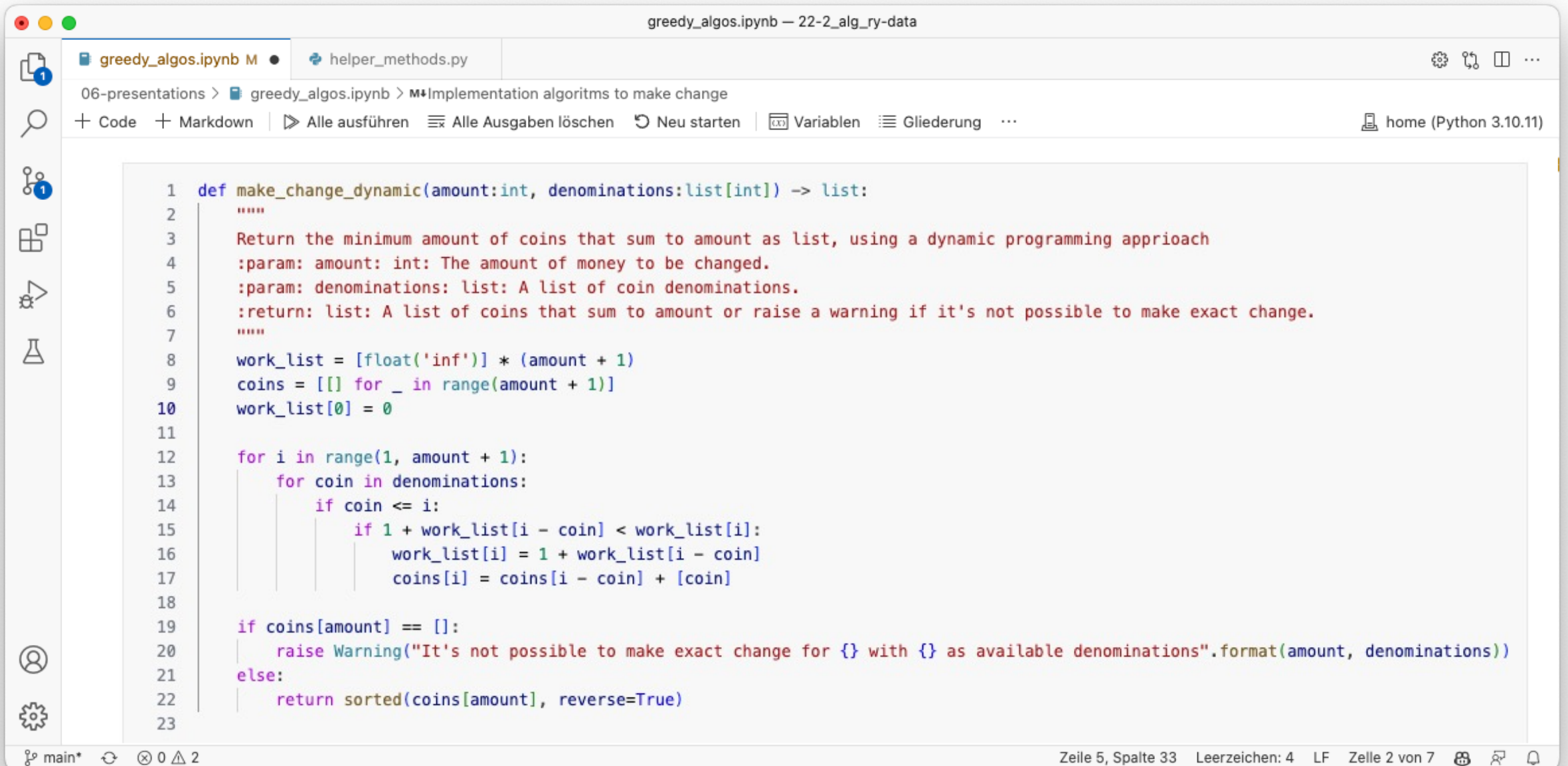
1. Greedy algorithm

```
1 def make_change_greedy(amount:int, denominations:list[int]) -> list:
2     """
3     Return a list of coins that sum to amount, using the greedy algorithm
4     :param: amount: int: The amount of money to be changed.
5     :param: denominations: list: A list of coin denominations.
6     :return: list: A list of coins that sum to amount.
7     """
8     denominations.sort(reverse=True)
9     coins = []
10    for coin in denominations:
11        while coin <= amount:
12            coins.append(coin)
13            amount -= coin
14    return coins
15
```

[8] ✓ 0.0s Python

main* ↺ 0 2

Zeile 5, Spalte 33 Leerzeichen: 4 LF Zelle 2 von 7



greedy_algos.ipynb — 22-2_alg_ry-data

greedy_algos.ipynb M • helper_methods.py

06-presentations > greedy_algos.ipynb > M*Implementation algorithms to make change

+ Code + Markdown | ▶ Alle ausführen | ✖ Alle Ausgaben löschen | ↺ Neu starten | [x] Variablen | ≡ Gliederung | ...

home (Python 3.10.11)

```
1 def make_change_dynamic(amount:int, denominations:list[int]) -> list:
2     """
3     Return the minimum amount of coins that sum to amount as list, using a dynamic programming approach
4     :param: amount: int: The amount of money to be changed.
5     :param: denominations: list: A list of coin denominations.
6     :return: list: A list of coins that sum to amount or raise a warning if it's not possible to make exact change.
7     """
8     work_list = [float('inf')] * (amount + 1)
9     coins = [[] for _ in range(amount + 1)]
10    work_list[0] = 0
11
12    for i in range(1, amount + 1):
13        for coin in denominations:
14            if coin <= i:
15                if 1 + work_list[i - coin] < work_list[i]:
16                    work_list[i] = 1 + work_list[i - coin]
17                    coins[i] = coins[i - coin] + [coin]
18
19    if coins[amount] == []:
20        raise Warning("It's not possible to make exact change for {} with {} as available denominations".format(amount, denominations))
21    else:
22        return sorted(coins[amount], reverse=True)
23
```

main* | 0 2 | Zeile 5, Spalte 33 | Leerzeichen: 4 | LF | Zelle 2 von 7

greedy_algos.ipynb — 22-2_alg_ry-data

greedy_algos.ipynb M • helper_methods.py

06-presentations > greedy_algos.ipynb > M*Implementation algorithms to make change

+ Code + Markdown | ▶ Alle ausführen | ☒ Alle Ausgaben löschen | ↺ Neu starten | [x] Variablen | ☰ Gliederung | ...

home (Python 3.10.11)

```
1 execute_change_algos(amount=42, denominations=[1, 5, 10, 25])
```

[5] ✓ 0.0s Python

```
... ##### Change for 42 #####
Denomination: 25, 10, 5, 1

Greedy: [25, 10, 5, 1, 1]
- Optimal: [25, 10, 5, 1, 1]
Greedy and optimal change are equal: True
```

```
1 execute_change_algos(amount=40, denominations=[5, 10, 20, 25])
```

[6] ✓ 0.0s Python

```
... ##### Change for 40 #####
Denomination: 25, 20, 10, 5

Greedy: [25, 10, 5]
- Optimal: [20, 20]
Greedy and optimal change are equal: False
```

main* | 0 | 2

Zeile 5, Spalte 33 | Leerzeichen: 4 | LF | Zelle 2 von 7

FAZIT



*Einfache
Implementierung*

*Time & Space
Complexity*

*Näherungs-
lösungen*



*Kennt kein
globales
Optimum*

Blickt nicht zurück

*Schwierigkeiten
bei komplexen
Problemen*



Anwendungsfälle bedürfen gründlicher Analyse.

Abhängig von den Anforderungen kann der Einsatz hoch effektiv sein.

VERWEISE & WEITERFÜHRENDE LINKS



Repository mit den Ressourcen dieser Präsentation

https://github.com/jacobrhn/22_2-AlgoDat-presentation-greedy_algos

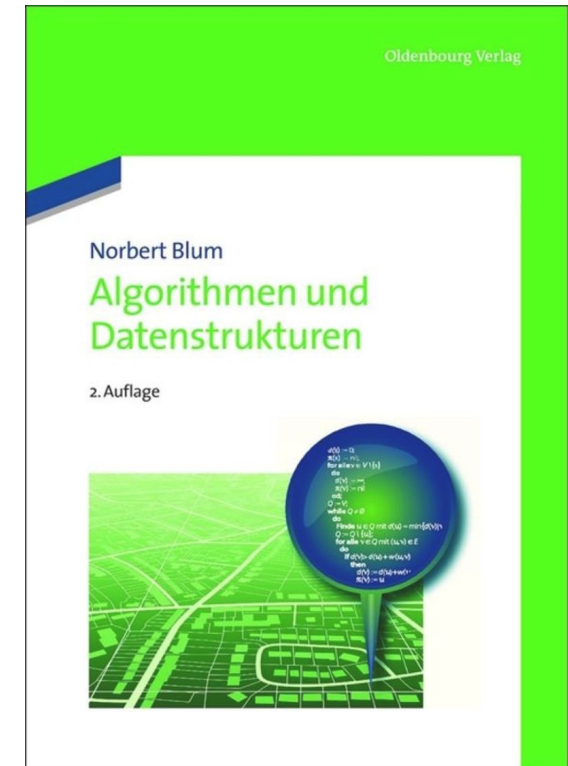
GeeksForGeeks über Greedy Algorithms

<https://www.geeksforgeeks.org/greedy-algorithms/>



CodeCrucks über das Making Change Problem

<https://codecrucks.com/making-change-problem-using-dynamic-programming/>



<https://www.degruyter.com/document/doi/10.1524/9783486719666/html>



DHBW

Duale Hochschule
Baden-Württemberg

Lörrach

GREEDY ALGORITHMS

Präsentation am 26. Mai 2023 erarbeitet von Gruppe 2, WDS/WWI22A:

*Mihabat Aeido, Samuel Butler, Tjark Gerken, Eric Harter, Jacob Ruhnau, Tom Warscheit
im Rahmen der Vorlesung „Algorithmen und Datenstrukturen“ bei Max Bergau*