Kapitel 3

Informatisches Problemlösen

Skript Informatik · Klasse 10 · MINT

Ein Paketdienst will die kürzeste Route durch alle Orte finden. Beim Packen für eine Wanderung soll der Rucksack nicht zu schwer werden, aber möglichst viel Nützliches enthalten. Solche Aufgaben heißen **Optimierungsprobleme**. In diesem Kapitel lernst du, Probleme genau zu beschreiben, alle Möglichkeiten systematisch auszuprobieren und clevere Abkürzungen zu finden, wenn das Ausprobieren zu lange dauert. Du arbeitest dabei vor allem mit Papier und Stift: mit Tabellen, Graphen und Listen. Programmieren musst du in diesem Kapitel nicht.

Das erwartet dich in diesem Kapitel:

1. **Problem und Probleminstanz** – Aufgaben genau beschreiben und als Graph, Tabelle oder Liste darstellen.
2. **Optimale Lösung und Brute-Force** – alle Möglichkeiten ausprobieren und die beste sicher finden.
3. **Heuristiken** – mit einer cleveren Faustregel schnell eine gute Lösung finden.
4. **Das Rucksackproblem** – Brute-Force und Greedy an einem zweiten Problem vergleichen.
5. **Aufwand** – warum das Ausprobieren schon bei wenigen Orten oder Gegenständen viel zu lange dauert.
6. **Heuristiken bewerten** – wie gut eine Faustregel wirklich ist, über viele Probleminstanzen hinweg.

So kannst du es sagen

Mit diesen Satzanfängen beschreibst und begründest du deine Ergebnisse in diesem Kapitel:

- *„Es handelt sich um ein Problem/eine Probleminstanz, weil …“*
- *„Der Ansatz prüft/wählt … und benötigt dabei …“*
- *„Die Heuristik ist hier geeignet/nicht geeignet, weil …“*

Am Ende hilft dir ein **Rückblick** mit den wichtigsten Begriffen und einem Kompetenz-Check beim Wiederholen.

3.1 Problem und Probleminstanz: Aufgaben genau beschreiben

Leitidee

Eine Paketbotin startet morgens im Depot, fährt zu mehreren Adressen und kehrt am Ende zurück. Welche Reihenfolge ist die kürzeste? Morgen hat sie andere Adressen – die Frage bleibt aber dieselbe. In der Informatik unterscheidet man deshalb genau zwischen der **allgemeinen Fragestellung** und dem **konkreten Fall** mit festen Zahlen. Außerdem lässt man alles weg, was für die Lösung keine Rolle spielt, und stellt den Fall übersichtlich dar: als Tabelle, als Graph oder als Liste. In diesem Abschnitt lernst du diese Werkzeuge kennen. Damit untersuchst du ein berühmtes Problem der Informatik: das Problem des Handlungsreisenden.

Lernziele

Nach diesem Abschnitt kannst du

- zwischen einem Problem und einer Probleminstanz unterscheiden und deine Entscheidung begründen,
- eine Probleminstanz als Tabelle, als Graph oder als Liste darstellen,
- die Begriffe Knoten, Kante und Kantengewicht verwenden,
- alle Rundreisen einer kleinen Probleminstanz systematisch auflisten,
- an einer Entfernungstabelle eine Symmetrie erkennen und erklären, was sie für die Anzahl der Rundreisen bedeutet.

3.1.1 Allgemeine Aufgabe oder konkreter Fall?

Aus dem Matheunterricht kennst du Aufgaben wie „Löse eine quadratische Gleichung“. Das ist eine allgemeine Aufgabe. Welche Gleichung gemeint ist, steht noch nicht fest. Erst bei „Löse $x^2 - 5x + 6 = 0$“ sind alle Zahlen bekannt, und du kannst loslegen.

Genauso ist es in der Informatik. Eine allgemeine Aufgabe hat noch **Leerstellen** – wie ein Formular, in dem die Felder noch leer sind. Man nennt sie ein **Problem**. Füllt man alle Leerstellen mit festen Werten aus, entsteht eine **Probleminstanz**. Zu einem Problem gibt es also sehr viele Probleminstanzen.

Definition: Problem

Ein **Problem** ist eine allgemeine Fragestellung mit Leerstellen. Es beschreibt, *was* gegeben ist und *was* gesucht wird, aber noch nicht mit festen Werten.

Definition: Probleminstanz

Eine **Probleminstanz** (kurz: Instanz) ist ein konkreter Fall eines Problems. Alle Leerstellen sind mit festen Werten gefüllt.

Ein Beispiel: „Finde die kürzeste Route zwischen beliebigen Städten“ ist ein Problem. Welche Städte gemeint sind und wie weit sie voneinander entfernt sind, bleibt offen. „Finde die kürzeste Route zwischen genau diesen fünf Städten mit diesen Entfernungen“ ist eine Probleminstanz.

Merke

- Ein **Problem** ist eine allgemeine Fragestellung mit Leerstellen (z. B. „Finde die kürzeste Route zwischen beliebigen Städten“).
- Eine **Probleminstanz** ist eine konkrete Ausprägung dieses Problems mit festen Werten (z. B. „Finde die kürzeste Route zwischen genau diesen 5 Städten mit diesen Entfernungen“).

So erkennst du den Unterschied: Frage dich, ob du **sofort anfangen könntest zu rechnen**. Fehlen noch Angaben („beliebige Städte“, „eine begrenzte Tragkraft“), ist es ein Problem. Stehen alle Werte fest („diese fünf Gegenstände“, „höchstens 10 kg“), ist es eine Probleminstanz.

Das Problem des Handlungsreisenden

Durch das ganze Kapitel begleitet dich ein berühmtes Problem. Ein Handlungsreisender (ein Vertreter, der Kunden besucht) will mehrere Städte besuchen und am Ende wieder zu Hause ankommen. Er möchte dabei möglichst wenig fahren. Heute stehen Paketdienste, Pflegedienste oder Müllabfuhr vor derselben Frage. Auf Englisch heißt es *Traveling Salesman Problem*, kurz **TSP**.

Beispiel: Problem des Handlungsreisenden

Problem: Finde für eine beliebige Menge von Städten mit gegebenen paarweisen Entfernungen die kürzeste **Rundreise**. Eine Rundreise besucht jede Stadt genau einmal und kehrt zum Ausgangspunkt zurück.

Probleminstanz: Vier Städte A, B, C und D mit diesen Entfernungen (in km):

	A	B	C	D
A	–	10	15	20
B	10	–	35	25
C	15	35	–	30
D	20	25	30	–

Die Tabelle liest du so: Suche die Zeile des einen Ortes und die Spalte des anderen. Wo sich Zeile und Spalte kreuzen, steht die Entfernung. In der Zeile **B** und der Spalte **D** steht zum Beispiel 25: Von B nach D sind es 25 km. Auf der Diagonale stehen Striche, denn von A nach A muss man nicht fahren.

Diese Probleminstanz wird dir in den nächsten Abschnitten immer wieder begegnen. Merke sie dir gut – oder markiere dir die Seite.

Arbeitsauftrag 1: Problem oder Probleminstanz?

Entscheide für jede Aussage, ob es sich um ein **Problem** (allgemein) oder um eine **Probleminstanz** (konkret) handelt. Begründe jeweils kurz.

1. „Packe möglichst wertvolle Gegenstände in einen Rucksack mit maximal 10 kg Tragkraft, wobei diese fünf Gegenstände zur Verfügung stehen: …“
2. „Packe möglichst wertvolle Gegenstände in einen Rucksack mit begrenzter Tragkraft.“
3. „Finde den kürzesten Weg von Saarbrücken nach Trier im gegebenen Straßennetz.“
4. „Finde in einem beliebigen Graphen den kürzesten Weg zwischen zwei beliebigen Knoten.“

Deine Antworten mit Begründung:

So kannst du es sagen

So begründest du deine Entscheidung – vom einfachen zum fachsprachlichen Satz:

- *„Da stehen Zahlen drin."* → *„Es handelt sich um eine Probleminstanz, weil alle Werte (Tragkraft und Gegenstände) festgelegt sind."*
- *„Das ist allgemein."* → *„Es handelt sich um ein Problem, weil die Tragkraft und die Gegenstände nicht festgelegt sind."*
- *„Aufgabe 3 gehört zu Aufgabe 4."* → *„Aussage 3 ist eine Instanz des Problems aus Aussage 4."*

3.1.2 Abstrahierte Darstellung: nur das Wichtige behalten

Denk an einen Netzplan von Bussen oder U-Bahnen. Dort fehlen Häuser, Parks und Straßennamen. Die Linien sind gerade gezeichnet, obwohl die echten Strecken Kurven haben. Trotzdem – oder gerade deshalb – findest du darauf schnell heraus, wo du umsteigen musst. Der Plan lässt alles weg, was für diese Frage unwichtig ist.

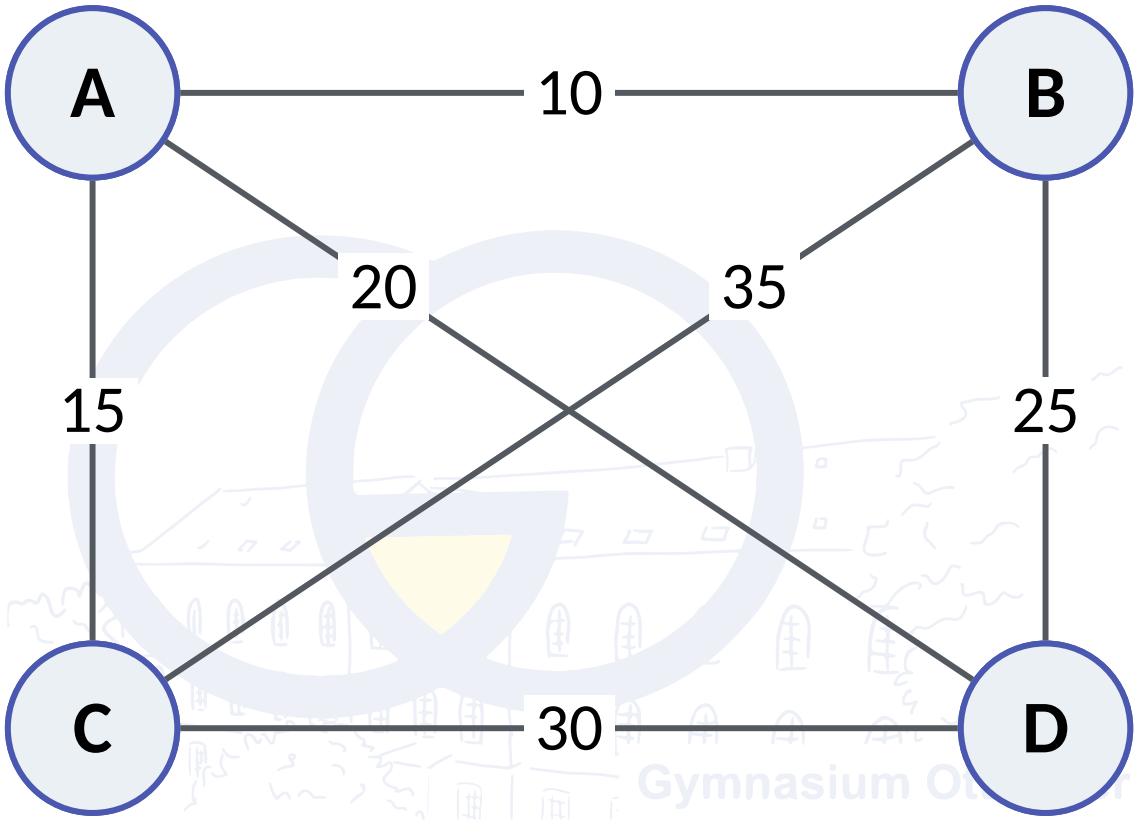
Dieses Weglassen nennt man **Abstrahieren**. Für das Problem des Handlungsreisenden sind nur die Orte und ihre Entfernungen wichtig. Straßennamen, Ampeln oder die Farbe der Häuser spielen keine Rolle. Die Entfernungstabelle oben ist deshalb eine **abstrahierte Darstellung** der Probleminstanz.

Definition: Abstrahierte Darstellung

Eine **abstrahierte Darstellung** enthält nur die Informationen, die für das Problem wichtig sind. Alles andere wird weggelassen. Häufige Formen sind Tabellen, Graphen und Listen.

Darstellung als Graph

Dieselbe Probleminstanz kannst du auch zeichnen. Jeder Ort wird ein Kreis, und zwischen zwei Orten zieht man eine Linie. An die Linie schreibt man die Entfernung. Eine solche Zeichnung heißt in der Informatik **Graph** (nicht zu verwechseln mit dem Funktionsgraphen aus Mathe).



Die Probleminstanz mit vier Orten als Graph. Die Zahlen an den Kanten sind die Entfernungen in km (Kantengewichte)

Definition: Graph

Ein **Graph** besteht aus **Knoten** (den Objekten, hier den Orten) und **Kanten** (den Verbindungen zwischen zwei Knoten). Steht an jeder Kante eine Zahl, zum Beispiel eine Entfernung, nennt man diese Zahl **Kantengewicht** und den Graphen **gewichtet**.

Im Graphen oben gibt es 4 Knoten und 6 Kanten, denn jeder Ort ist mit jedem anderen direkt verbunden. Die Kante zwischen A und D hat das Gewicht 20, die Kante zwischen B und C das Gewicht 35. Diese beiden Kanten kreuzen sich in der Mitte. Das ist nur eine Folge der Zeichnung: Es gibt dort keinen Knoten, man kann also an der Kreuzung nicht „abbiegen“.

Stolperstein

In einem Graphen zählen **nur die Zahlen an den Kanten**, nicht die gezeichnete Länge. Im Bild oben ist die Kante A–B (10 km) länger gezeichnet als die Kante A–C (15 km). Das ist kein Fehler. Wo die Knoten auf dem Papier liegen, darfst du frei wählen – Hauptsache, sie sind übersichtlich angeordnet.

Darstellung als Liste

Nicht jedes Problem hat mit Orten zu tun. Beim Packen eines Rucksacks sind die Gegenstände wichtig, und von jedem Gegenstand nur sein Gewicht und sein Wert. Farbe, Marke oder Form lässt man weg. Eine Probleminstanz kann man dann als **Liste** aufschreiben, zum Beispiel so:

Gegenstand	Gewicht (kg)	Wert (€)
Zelt	5	60
Schlafsack	4	40
Kocher	2	30

Diese Art von Problem lernst du im Abschnitt über das Rucksackproblem genauer kennen.

Merke

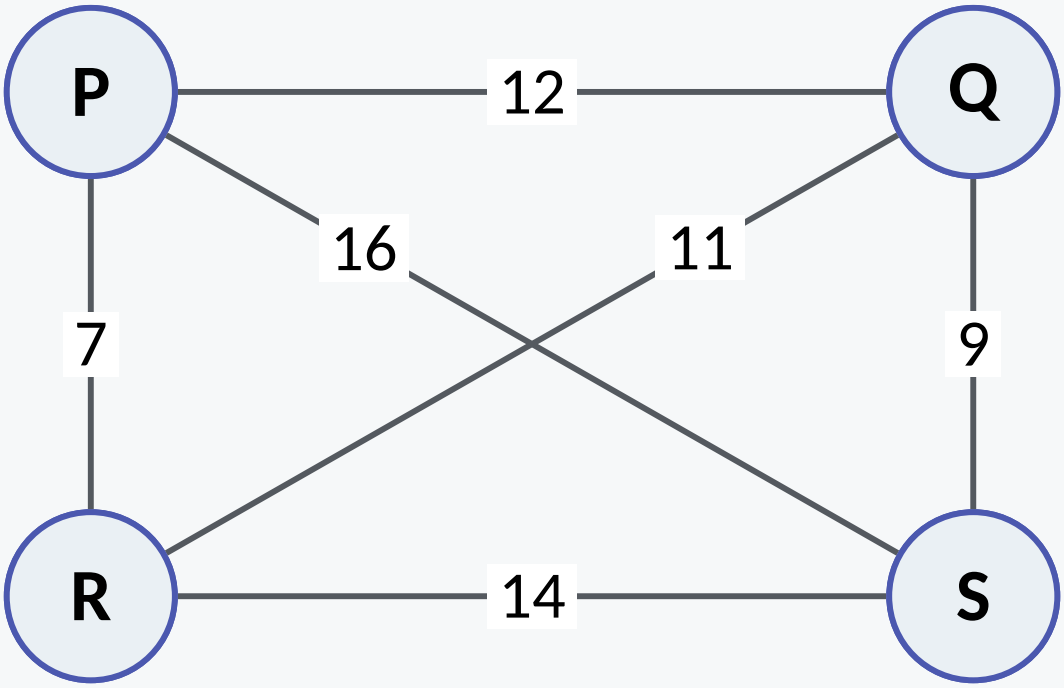
Probleminstanzen werden häufig **abstrahiert dargestellt**, um sie handhabbar zu machen, z. B. als

- **Tabelle** (z. B. Entfernungstabelle zwischen Orten),
- **Graph** (Knoten = Objekte, Kanten = Beziehungen oder Entfernungen),
- **Liste** (z. B. Liste von Gegenständen mit Gewicht und Wert).

Arbeitsauftrag 2: Vom Graphen zur Tabelle

Ein Sportverein plant eine Fahrt zu vier Turnierorten P, Q, R und S. Der Graph zeigt die Entfernungen in km.

Gymnasium Ottweiler



Vier Turnierorte mit Entfernungen in km

1. Übertrage die Entfernungen in die Tabelle.
2. Nenne die Anzahl der Knoten und der Kanten.

	P	Q	R	S
P	–			
Q		–		
R			–	
S				–

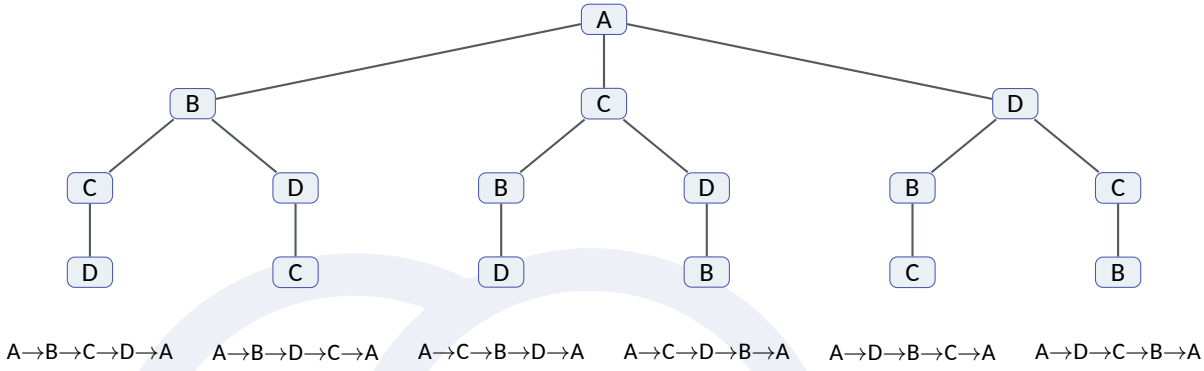
Anzahl der Knoten und Kanten:

3.1.3 Rundreisen systematisch auflisten

Bevor man die kürzeste Rundreise suchen kann, muss man wissen, welche Rundreisen es überhaupt gibt. Wir bleiben bei den vier Orten A, B, C und D und legen fest: Die Reise beginnt und endet in A. Eine Rundreise schreiben wir mit Pfeilen auf, zum Beispiel A → B → C → D → A.

Damit du keine Rundreise vergisst und keine doppelt zählst, gehst du **systematisch** vor. Das gelingt mit einem Baum, wie in der Abbildung unten:

- Von A aus gibt es **3** Möglichkeiten für den zweiten Ort: B, C oder D.
- Danach sind noch **2** Orte übrig. Für den dritten Ort gibt es also 2 Möglichkeiten.
- Für den vierten Ort bleibt nur noch **1** Möglichkeit. Danach geht es zurück nach A.



Alle Rundreisen mit Start in A. Jeder Weg von oben nach unten ist eine Rundreise. Es gibt $3 \cdot 2 \cdot 1 = 6$ Stück

Mit Start in A gibt es also $3 \cdot 2 \cdot 1 = 6$ Rundreisen. Die Schreibweise $3 \cdot 2 \cdot 1$ kürzt man in Mathe mit $3!$ ab (gelesen: „drei Fakultät“). Mehr dazu im nächsten Abschnitt.

Versuch: Andere Reihenfolge im Baum

Zeichne den Baum noch einmal, aber beginne in Ort B. Welche Rundreisen erhältst du? Vergleiche sie mit den sechs Rundreisen oben.

3.1.4 Symmetrien erkennen

Wer eine Probleminstanz genau anschaut und dabei auch kleine **Varianten** ausprobiert (anderer Startort, andere Zahlen), entdeckt oft Regelmäßigkeiten. Solche Regelmäßigkeiten heißen **strukturelle Eigenschaften** eines Problems. Sie helfen später, Arbeit zu sparen.

Eine wichtige strukturelle Eigenschaft ist die **Symmetrie**. Im Alltag kennst du Symmetrie von Figuren: Eine symmetrische Figur sieht gespiegelt genauso aus. Bei Tabellen ist es ähnlich. Bei einer symmetrischen Tabelle kannst du an der Diagonale (den Strichen) spiegeln – es kommen dieselben Zahlen heraus.

Merke

Das Untersuchen mehrerer **Varianten** einer Probleminstanz hilft, **Symmetrien** und Muster zu erkennen, die spätere Lösungsstrategien vereinfachen können.

Arbeitsauftrag 3: Symmetrien erkennen

Betrachte die Entfernungstabelle der vier Orte A, B, C und D (im Beispiel *Problem des Handlungsreisenden*).

1. Welche strukturelle Eigenschaft erkennst du an der Tabelle, wenn du die Entfernung von A nach B mit der Entfernung von B nach A vergleichst? Prüfe auch die anderen Paare. Was bedeutet diese Eigenschaft für die Anzahl der Rundreisen, die du tatsächlich unterscheiden musst?
2. Wie viele **verschiedene** Rundreisen gibt es bei vier Orten, wenn du diese Eigenschaft berücksichtigst? Zähle systematisch. Tipp: Suche in der Liste der sechs Rundreisen nach Paaren.

Deine Antworten:

Die Symmetrie spart also die Hälfte der Arbeit. Bei vier Orten musst du statt 6 nur 3 Rundreisen untersuchen.

Stolperstein

Die Symmetrie gilt nur, wenn Hin- und Rückweg wirklich gleich lang sind. Das ist nicht immer so. Gibt es **Einbahnstraßen**, kann der Weg von A nach B kürzer sein als der Weg von B nach A. Misst man statt der Entfernung die **Fahrzeit**, kann ein Weg bergab schneller sein als bergauf. Dann ist die Tabelle nicht symmetrisch, und du musst alle 6 Rundreisen einzeln untersuchen.

Eine zweite Symmetrie steckt im **Startort**. Die Rundreise B → D → C → A → B besucht die Orte in derselben Reihenfolge wie A → B → D → C → A. Sie beginnt nur an einer anderen Stelle des Kreises. Beide benutzen dieselben vier Kanten und sind gleich lang. Deshalb darf man den Startort einfach festlegen – wir nehmen immer A.

Arbeitsauftrag 4: Gleiche Rundreise?

Gib bei jedem Paar an, ob es sich um dieselbe Rundreise handelt (also dieselben Kanten benutzt werden). Begründe kurz.

1. A → C → D → B → A und C → D → B → A → C
2. A → B → C → D → A und B → A → D → C → B
3. A → B → C → D → A und A → C → B → D → A

Gymnasium Ottweiler

Deine Antworten:

3.1.5 Für Interessierte: Rundreisen mit Python auflisten

Dieser Unterabschnitt ist freiwillig. Für das Kapitel musst du nicht programmieren. Wenn du Lust hast, kannst du mit Python überprüfen, ob du alle Rundreisen gefunden hast.

Die Idee: Für den zweiten, dritten und vierten Ort probiert das Programm jeweils alle drei Orte B, C und D aus. Dafür braucht es drei ineinander geschachtelte `for`-Schleifen. Eine Reihenfolge ist aber nur dann eine Rundreise, wenn kein Ort doppelt vorkommt. Das prüft eine Bedingung.

```
1 orte = ["B", "C", "D"]
2 anzahl = 0
3 for ort2 in orte:
4     for ort3 in orte:
5         for ort4 in orte:
6             if ort2 != ort3 and ort2 != ort4 and ort3 != ort4:
7                 anzahl += 1
8                 print("A ->", ort2, "->", ort3, "->", ort4, "-> A")
9 print("Anzahl der Rundreisen:", anzahl)
```

AUSGABE

```
A -> B -> C -> D -> A
A -> B -> D -> C -> A
A -> C -> B -> D -> A
A -> C -> D -> B -> A
A -> D -> B -> C -> A
A -> D -> C -> B -> A
Anzahl der Rundreisen: 6
```

- **Zeile 1:** Die Liste enthält die Orte, die nach dem Start in A besucht werden müssen.
- **Zeile 2:** Ein Zähler für die gefundenen Rundreisen beginnt bei 0.
- **Zeilen 3–5:** Drei Schleifen ineinander. Die äußere wählt den zweiten Ort (`ort2`), die mittlere den dritten (`ort3`), die innere den vierten (`ort4`). Insgesamt werden $3 \cdot 3 \cdot 3 = 27$ Reihenfolgen erzeugt – auch unsinnige wie A → B → B → B → A.
- **Zeile 6:** Nur wenn alle drei Orte verschieden sind, ist es eine Rundreise. Von den 27 Reihenfolgen bleiben 6 übrig.
- **Zeilen 7–8:** Der Zähler wird erhöht, und die Rundreise wird ausgegeben.
- **Zeile 9:** Nach allen Schleifen wird die Anzahl ausgegeben.

Die Ausgabe zeigt dieselben sechs Rundreisen wie der Baum – sogar in derselben Reihenfolge.

3.1.6 Übungen

Arbeitsauftrag 5: Probleminstanz als Liste

Eine Lehrerin will 6 Schülerinnen und Schüler so auf zwei Teams aufteilen, dass beide Teams möglichst gleich stark sind. Jede Person hat eine Spielstärke von 1 bis 10: Ali 7, Ben 4, Cem 9, Dana 6, Ela 5, Finn 3.

1. Formuliere das allgemeine **Problem**, zu dem diese Situation gehört.
2. Stelle die **Probleminstanz** abstrahiert als Liste dar. Welche Angaben lässt du weg?

Deine Antworten:

Gymnasium Ottweiler

Arbeitsauftrag 6 (Zusatz): Eigenes Problem

Formuliere selbst ein **Problem** und eine dazu passende **Probleminstanz** aus einem Bereich deiner Wahl, zum Beispiel Stundenplanung, Einkaufstour oder Gruppeneinteilung. Gib an, wie man die Probleminstanz abstrahiert darstellen kann (Tabelle, Graph oder Liste).

Dein Problem und deine Probleminstanz:

3.1.7 Weitere Übungen

Hier kannst du das Gelernte weiter üben. Die Punkte im Titel zeigen, wie schwer eine Aufgabe ist: ●○○ leicht, ●●○ mittel, ●●● knifflig. Fang mit den leichten an.

Arbeitsauftrag 7: Problem oder Probleminstanz? Noch einmal ●○○

Entscheide bei jeder Aussage, ob es sich um ein **Problem** oder um eine **Probleminstanz** handelt. Notiere in der Tabelle auch das Wort oder die Angabe, an der du es erkennst.

1. „Plane für beliebig viele Freundinnen und Freunde eine Abholreihenfolge, sodass die Fahrt möglichst kurz ist."
2. „Mia startet zu Hause und will zur Bäckerei, zum Getränkemarkt und zum Supermarkt. Die Entfernungen stehen in dieser Tabelle: …"
3. „Teile eine Klasse so in zwei Gruppen auf, dass beide Gruppen gleich groß sind."
4. „Packe in einen Koffer mit höchstens 20 kg möglichst nützliche Dinge aus dieser Liste: Jacke 2 kg, Schuhe 1 kg, …"
5. „Finde die kürzeste Rundreise durch die Landeshauptstädte Saarbrücken, Mainz, Wiesbaden und Stuttgart."
6. „Finde in einem Straßennetz den kürzesten Weg zwischen zwei Orten."

Nr.	Problem oder Probleminstanz?	daran erkenne ich es
1		
2		
3		
4		
5		
6		

Arbeitsauftrag 8: Die Entfernungstabelle lesen ●○○

Mia kauft für eine Klassenparty ein. Sie startet zu Hause (Z) und fährt mit dem Fahrrad zur Bäckerei (B), zum Getränkemarkt (G) und zum Supermarkt (S). Die Tabelle zeigt die Entfernungen in km.

	Z	B	G	S
Z	–	3	7	6
B	3	–	9	5
G	7	9	–	4
S	6	5	4	–

1. Wie weit ist es von der Bäckerei zum Supermarkt?
2. Welches Geschäft liegt am nächsten an Mias Zuhause?
3. Welche zwei Orte liegen am weitesten auseinander?
4. Die Tabelle hat 12 Zahlen. Wie viele **verschiedene** Strecken musste man trotzdem nur messen? Begründe.

Deine Antworten:

Gymnasium Ottweiler

Arbeitsauftrag 9: Von der Tabelle zum Graphen ●○○

Skizziere die Probleminstanz aus dem Auftrag *Die Entfernungstabelle lesen* als Graph. Zeichne für jeden Ort einen Knoten und für jedes Paar von Orten eine Kante. Schreibe die Entfernungen als Kantengewichte an die Kanten. Gib an, wie viele Knoten und Kanten dein Graph hat.

Dein Graph:

Anzahl der Knoten und Kanten:

Arbeitsauftrag 10: Nur das Wichtige behalten ●●○

Lies den Text.

Jonas trägt jeden Samstag Prospekte aus. Er startet mit seinem roten Fahrrad am Kiosk (K) und muss in drei Straßen: Ahornweg (A), Birkenstraße (B) und Lindenplatz (L). Im Ahornweg stehen 12 Häuser, in der Birkenstraße wohnt ein Hund, der immer bellt. Vom Kiosk sind es 400 m zum Ahornweg, 700 m zur Birkenstraße und 500 m zum Lindenplatz. Vom Ahornweg zur Birkenstraße sind es 300 m, vom Ahornweg zum Lindenplatz 600 m und von der Birkenstraße zum Lindenplatz 400 m. Die Prospekte wiegen zusammen 6 kg. Am Ende bringt Jonas die leere Tasche zum Kiosk zurück. Er möchte möglichst wenig fahren.

1. Nenne die Angaben, die für Jonas' Frage wichtig sind, und mindestens drei Angaben, die du weglassen kannst.
2. Stelle die Probleminstanz abstrahiert als Entfernungstabelle (in m) dar.

	K	A	B	L
K	–			
A		–		
B			–	
L				–

Wichtige und unwichtige Angaben:

Arbeitsauftrag 11: Rundreisen beim Partyeinkauf ●●○

Mia aus dem Auftrag *Die Entfernungstabelle lesen* startet und endet zu Hause (Z) und besucht B, G und S in irgendeiner Reihenfolge.

1. Liste alle Rundreisen systematisch auf. Tipp: Zeichne dir zuerst einen Baum wie im Abschnitt *Rundreisen systematisch auflisten*.
2. Schreibe zu jeder Rundreise die Nummer ihrer Rückwärts-Partnerin.
3. Wie viele **verschiedene** Rundreisen muss Mia wirklich vergleichen? Begründe mit der Symmetrie der Tabelle.

Nr.	Rundreise	rückwärts gefahren ist das Nr.
1		
2		
3		
4		
5		
6		

Anzahl der verschiedenen Rundreisen und Begründung:

Arbeitsauftrag 12: Bergauf und bergab ●●●

Noah wohnt unten im Tal (H). Die Schule (S) und die Musikschule (M) liegen weiter oben am Hang, das Freibad (F) ganz oben. Die Tabelle zeigt, wie viele Minuten Noah mit dem Fahrrad braucht. Lies sie so: **Zeile = von, Spalte = nach**. Von H nach S braucht er also 8 Minuten.

von \ nach	H	S	M	F
H	–	8	10	15
S	6	–	7	9
M	8	7	–	8
F	9	6	5	–

1. Prüfe, ob die Tabelle symmetrisch ist. Nenne alle Paare von Orten, bei denen Hin- und Rückweg gleich lange dauern.
2. Erkläre, warum die Tabelle hier nicht symmetrisch ist.
3. Noah plant eine Rundreise mit Start und Ende bei H durch S, M und F. Wie viele Rundreisen muss er vergleichen, wenn er die Fahrzeit möglichst klein halten will? Begründe, warum es nicht wie beim Partyeinkauf weniger sind.

Deine Antworten:

Gymnasium Ottweiler

3.1.8 Das Wichtigste in Kürze

- Ein **Problem** ist eine allgemeine Fragestellung mit Leerstellen. Eine **Probleminstanz** ist ein konkreter Fall mit festen Werten.
- Beim **Problem des Handlungsreisenden** (TSP) sucht man die kürzeste Rundreise, die jeden Ort genau einmal besucht und zum Start zurückkehrt.
- Eine **abstrahierte Darstellung** enthält nur das Wichtige: als **Tabelle**, als **Graph** oder als **Liste**.
- Ein **Graph** besteht aus **Knoten** und **Kanten**. Die Zahlen an den Kanten heißen **Kantengewichte**. Die gezeichnete Länge einer Kante spielt keine Rolle.
- Bei vier Orten und festem Start gibt es $3 \cdot 2 \cdot 1 = 6$ Rundreisen. Ein Baum hilft, sie systematisch aufzulisten.
- Ist die Entfernungstabelle **symmetrisch**, sind eine Rundreise und ihre Rückwärts-Rundreise gleich lang. Dann muss man nur die Hälfte untersuchen.

Selbstcheck

- ☐ Ich kann erklären, was ein Problem und was eine Probleminstanz ist, und Beispiele zuordnen.
- ☐ Ich kann eine Entfernungstabelle lesen.
- ☐ Ich kann eine Probleminstanz als Graph zeichnen und umgekehrt aus einem Graphen eine Tabelle erstellen.
- ☐ Ich kenne die Begriffe Knoten, Kante und Kantengewicht.
- ☐ Ich kann alle Rundreisen durch vier Orte mit einem Baum auflisten.
- ☐ Ich kann erklären, warum es bei einer symmetrischen Tabelle nur 3 verschiedene Rundreisen durch vier Orte gibt.

Gymnasium Ottweiler

3.2 Optimale Lösung und Brute-Force: alle Möglichkeiten ausprobieren

Leitidee

Im letzten Abschnitt hast du alle Rundreisen durch die vier Orte A, B, C und D aufgelistet. Jetzt geht es um die eigentliche Frage: Welche davon ist die kürzeste? Die sicherste Methode klingt fast zu einfach: Man rechnet für **jede** Rundreise die Länge aus und nimmt die kürzeste. Diese Methode heißt Brute-Force. Sie findet garantiert die beste Lösung. In diesem Abschnitt führst du sie von Hand durch – und merkst dabei, wo ihr Haken liegt.

Lernziele

Nach diesem Abschnitt kannst du

- erklären, was eine zulässige Lösung ist, und prüfen, ob ein Vorschlag zulässig ist,
- eine optimale Lösung als zulässige Lösung definieren, die bezüglich eines festgelegten Kriteriums die beste ist,
- die Länge einer Rundreise mit einer Entfernungstabelle berechnen,
- den Brute-Force-Ansatz beschreiben und an einer kleinen Probleminstanz von Hand durchführen,
- die Anzahl der Rundreisen bei n Orten mit $(n-1)!$ berechnen und beschreiben, wie schnell sie wächst.

3.2.1 Lösung, zulässige Lösung, optimale Lösung

Stell dir vor, du suchst einen Weg von zu Hause zur Schule. Es gibt viele Wege. Manche kommen aber gar nicht in Frage, zum Beispiel einer, der durch einen gesperrten Park führt. Von den erlaubten Wegen suchst du den besten. Aber was heißt „der beste“? Der kürzeste? Der schnellste? Der mit den wenigsten Ampeln? Erst wenn du das festlegst, kannst du sagen, welcher Weg gewinnt.

Genau diese drei Stufen unterscheidet man auch in der Informatik:

1. Ein **Lösungsvorschlag** ist irgendeine Antwort auf die Frage, zum Beispiel irgendeine Reihenfolge der Orte.
2. Eine **zulässige Lösung** hält alle Bedingungen des Problems ein.
3. Eine **optimale Lösung** ist unter allen zulässigen Lösungen die beste – gemessen an einem vorher festgelegten **Kriterium**.

Definition: Zulässige Lösung

Eine **zulässige Lösung** ist eine Lösung, die alle Bedingungen des Problems erfüllt.

Gymnasium Ottweiler

Definition: Optimale Lösung

Eine **optimale Lösung** ist eine zulässige Lösung, die in Bezug auf ein festgelegtes Kriterium die beste ist.

Beim Problem des Handlungsreisenden heißt die Bedingung: Jeder Ort wird **genau einmal** besucht, und die Reise endet am Startort. Das Kriterium ist die **Gesamtlänge**. Die beste Rundreise ist also die mit der kleinsten Gesamtlänge.

Vorschlag	zulässig?	Grund
A → B → C → D → A	✓	jeder Ort genau einmal, zurück in A
A → B → C → A	✗	D wird nicht besucht
A → B → C → B → D → A	✗	B wird zweimal besucht
A → B → C → D	✗	die Reise endet nicht in A

Merke

- Eine **zulässige Lösung** erfüllt alle Bedingungen des Problems (z. B. „besucht jede Stadt genau einmal“).
- Eine **optimale Lösung** ist unter allen zulässigen Lösungen diejenige, die bezüglich eines festgelegten Kriteriums (z. B. Gesamtlänge) **am besten** ist.

Stolperstein

„Optimal“ ergibt nur Sinn, wenn das Kriterium feststeht. Die kürzeste Rundreise muss nicht die schnellste sein, wenn auf einer Strecke Stau ist. Achte außerdem darauf: Eine optimale Lösung muss nicht die einzige sein. Haben zwei Rundreisen dieselbe kleinste Länge, sind beide optimal.

Arbeitsauftrag 13: Zulässig oder nicht?

Für die vier Orte A, B, C und D liegen diese Vorschläge vor. Entscheide jeweils, ob sie zulässige Lösungen für das Problem des Handlungsreisenden mit Start in A sind. Begründe, wenn ein Vorschlag nicht zulässig ist.

1. A → D → C → B → A
2. A → C → D → A
3. A → D → B → D → C → A
4. A → C → B → D → A

Deine Antworten:

3.2.2 Die Länge einer Rundreise berechnen

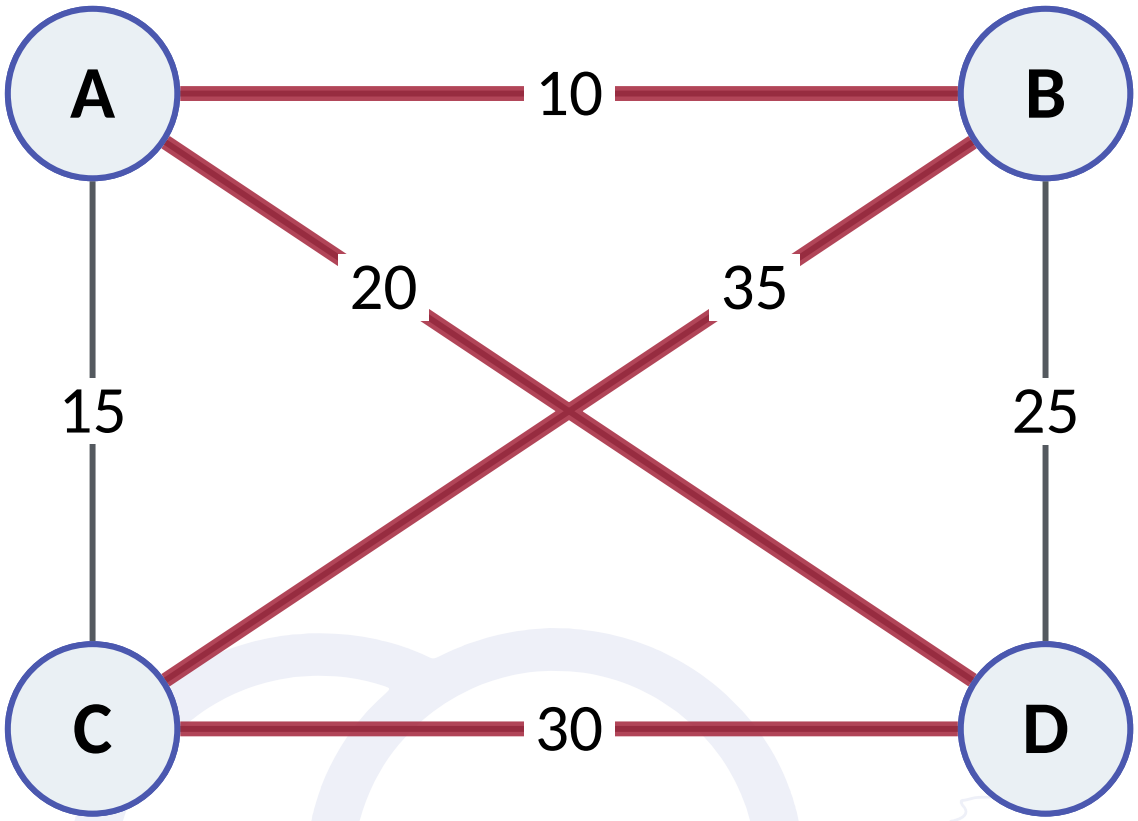
Um Rundreisen zu vergleichen, brauchst du ihre Gesamtlänge. Du addierst dafür die Entfernungen aller Teilstrecken. Nimm die Entfernungstabelle der vier Orte zur Hand:

	A	B	C	D
A	–	10	15	20
B	10	–	35	25
C	15	35	–	30
D	20	25	30	–

Für die Rundreise A → B → C → D → A gibt es vier Teilstrecken. Die Rückfahrt von D nach A gehört dazu – sie wird leicht vergessen.

```
A-B:  10
B-C:  35
C-D:  30
D-A:  20
----------
      95
```

Die Rundreise A → B → C → D → A ist also 95 km lang. Kurz schreibt man: $10 + 35 + 30 + 20 = 95$. Im Graphen sind die vier benutzten Kanten rot eingezeichnet:



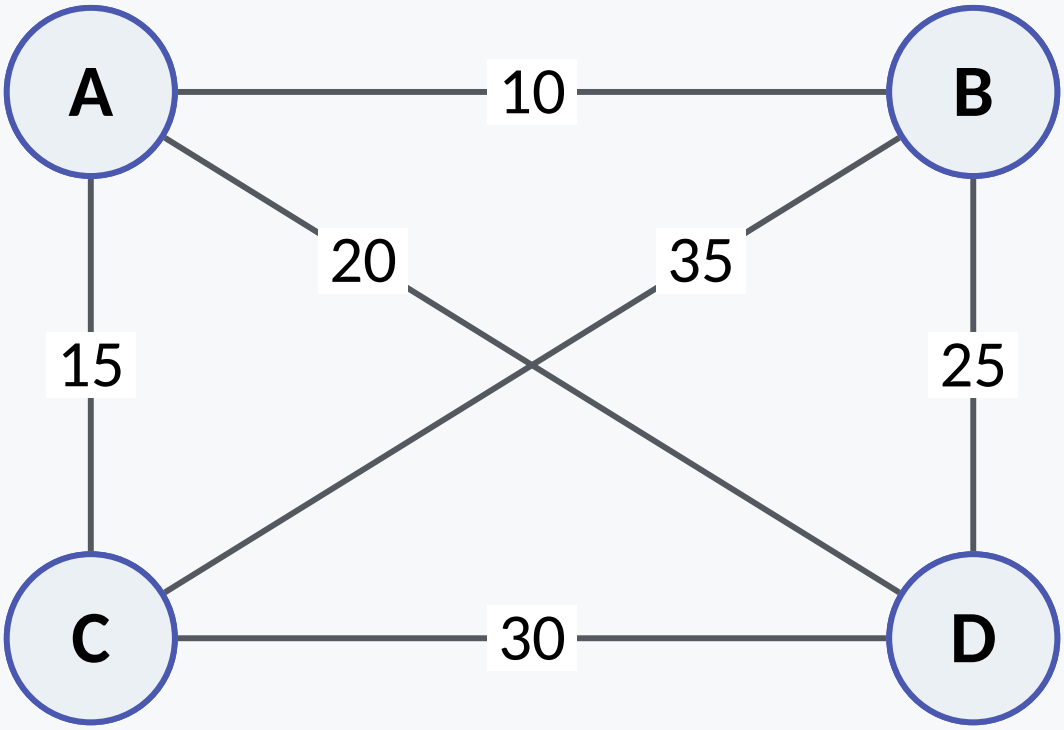
Die Rundreise A → B → C → D → A (rot) ist $10 + 35 + 30 + 20 = 95$ km lang

Tipp

Schreibe bei jeder Teilstrecke die beiden Orte dazu, wie oben (A-B, B-C, ...). So siehst du sofort, ob eine Strecke fehlt. Eine Rundreise durch n Orte hat immer genau n Teilstrecken.

Arbeitsauftrag 14: Rundreisen vergleichen

1. Berechne die Längen der Rundreisen A → B → D → C → A und A → C → B → D → A. Schreibe die Rechnung mit auf.
2. Vergleiche mit der Rundreise A → B → C → D → A (95 km). Welche der drei Rundreisen ist am kürzesten?
3. Zeichne die kürzeste der drei Rundreisen in den Graphen ein.



Vorlage: Zeichne die kürzeste Rundreise ein

Deine Rechnungen und dein Vergleich:

Gymnasium Ottweiler

3.2.3 Der Brute-Force-Ansatz

Stell dir ein Fahrradschloss mit drei Zahlenrädern vor. Du hast die Zahl vergessen. Was tun? Du kannst einfach **alle** Zahlen von 000 bis 999 durchprobieren. Das ist mühsam, aber du findest die richtige Zahl **garantiert** – spätestens nach 1000 Versuchen.

Diese Methode heißt **Brute-Force** (englisch, gesprochen „brut fors“, wörtlich „rohe Gewalt“). Gemeint ist: Man denkt nicht lange nach, sondern probiert stur alles aus.

Definition: Brute-Force-Ansatz

Beim **Brute-Force-Ansatz** werden **alle** zulässigen Lösungen systematisch erzeugt und bewertet. Die beste davon ist die optimale Lösung.

Für das Problem des Handlungsreisenden besteht der Brute-Force-Ansatz aus drei Schritten:

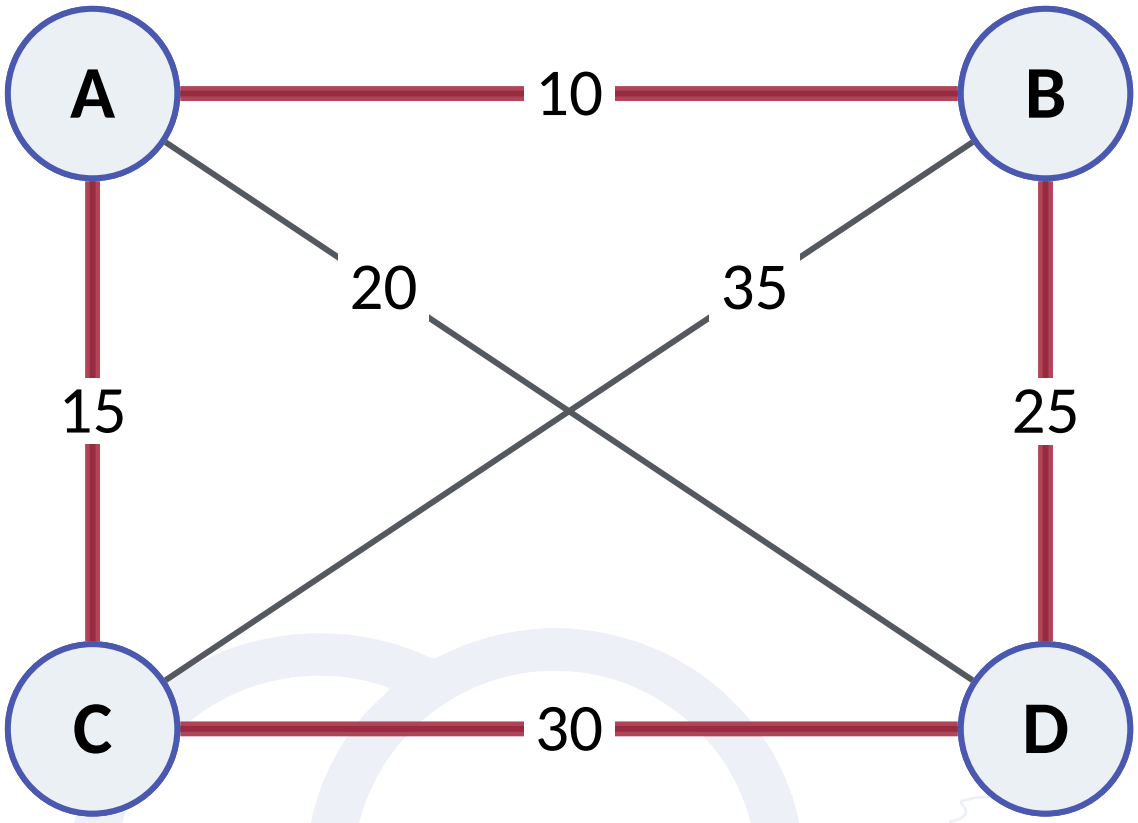
1. **Erzeugen:** Liste alle Rundreisen systematisch auf, zum Beispiel mit dem Baum aus dem letzten Abschnitt.
2. **Bewerten:** Berechne für jede Rundreise die Gesamtlänge.
3. **Auswählen:** Wähle die Rundreise mit der kleinsten Gesamtlänge.

Beispiel: Brute-Force für die vier Orte A, B, C, D

Rundreise	Rechnung	Länge (km)
A → B → C → D → A	$10 + 35 + 30 + 20$	95
A → B → D → C → A	$10 + 25 + 30 + 15$	**80**
A → C → B → D → A	$15 + 35 + 25 + 20$	95
A → C → D → B → A	$15 + 30 + 25 + 10$	**80**
A → D → B → C → A	$20 + 25 + 35 + 15$	95
A → D → C → B → A	$20 + 30 + 35 + 10$	95

Die kleinste Länge ist 80 km. Die optimale Rundreise ist A → B → D → C → A. Ihre Rückwärts-Variante A → C → D → B → A ist ebenfalls optimal.

Gymnasium Ottweiler



Die optimale Rundreise A → B → D → C → A (rot) ist $10 + 25 + 30 + 15 = 80$ km lang

Warum kann man sicher sein, dass es keine kürzere Rundreise gibt? Weil **jede** Rundreise in der Tabelle steht. Eine bessere kann es also nicht geben. Das ist die große Stärke von Brute-Force.

Merke

- Ein **Brute-Force-Ansatz** („rohe Gewalt“) ermittelt die optimale Lösung, indem er **alle** zulässigen Lösungen systematisch erzeugt, bewertet und die beste auswählt.
- Vorteil: Brute-Force liefert garantiert die **optimale** Lösung.
- Nachteil: Die Anzahl der zu prüfenden Lösungen wächst bei vielen Problemen sehr schnell (z. B. **faktoriell** beim Problem des Handlungsreisenden). Brute-Force wird für größere Instanzen schnell unbrauchbar.

Was „faktoriell“ bedeutet, erfährst du gleich im Abschnitt über die Anzahl der Rundreisen.

So kannst du es sagen

So beschreibst du das Vorgehen in Fachsprache:

- *„Ich hab alles ausprobiert.“ → „Ich habe alle zulässigen Lösungen systematisch erzeugt und ihre Gesamtlängen berechnet.“*
- *„80 ist die beste.“ → „Die Rundreise A → B → D → C → A ist optimal bezüglich der Gesamtlänge, weil keine andere zulässige Rundreise kürzer ist.“*

3.2.4 Brute-Force mit fünf Orten

Mit vier Orten war Brute-Force schnell erledigt. Was passiert mit einem Ort mehr? Das probierst du jetzt selbst aus.

Wie viele Rundreisen gibt es bei fünf Orten mit Start in A? Für den zweiten Ort gibt es 4 Möglichkeiten, dann 3, dann 2, dann 1. Das sind $4 \cdot 3 \cdot 2 \cdot 1 = 24$ Rundreisen. Wegen der Symmetrie musst du aber nur die Hälfte berechnen, also 12.

Tipp

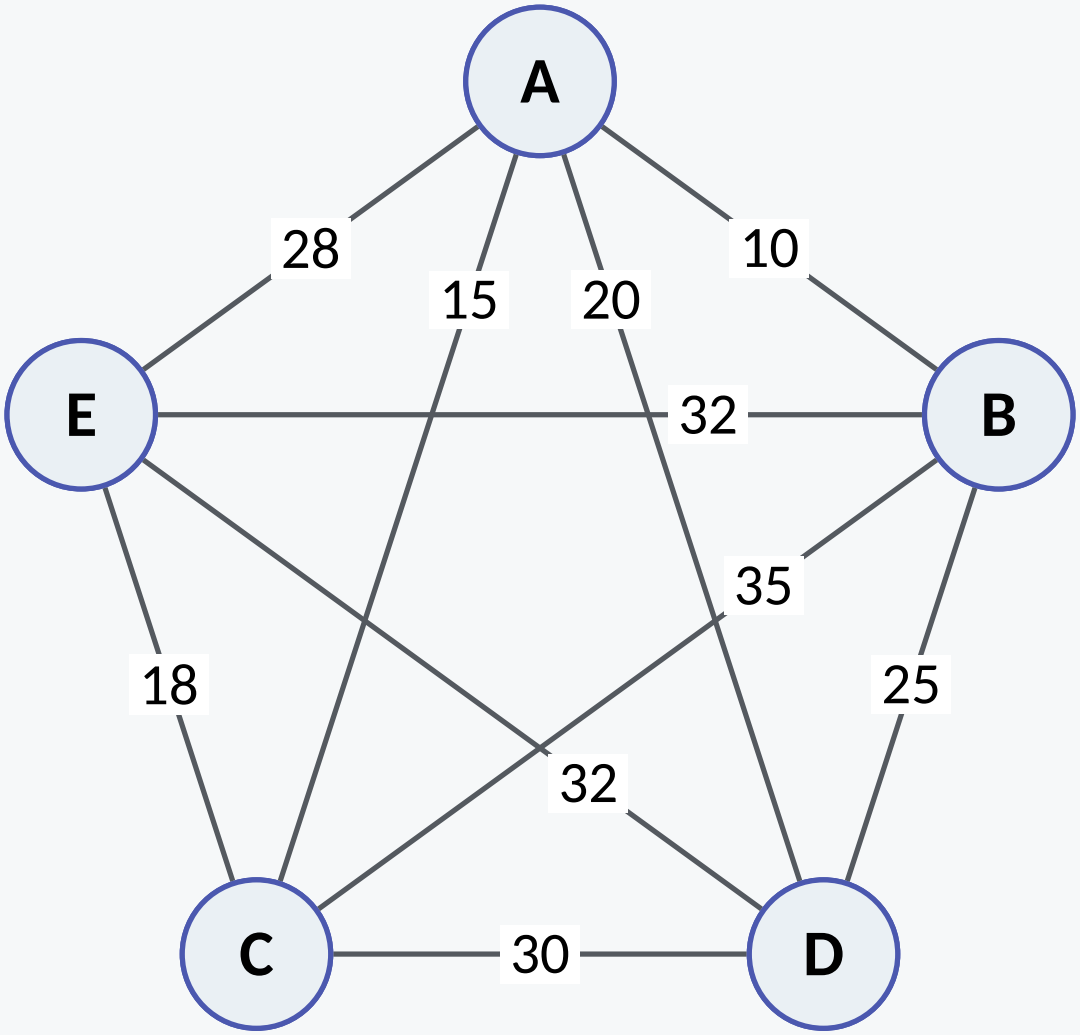
So findest du die 12 verschiedenen Rundreisen, ohne eine doppelt zu zählen: Gehe vor wie im Baum (zweiter Ort B, C, D, E der Reihe nach) und schreibe nur die Rundreisen auf, bei denen der **zweite** Ort im Alphabet **vor** dem Ort steht, der **direkt vor der Rückkehr** nach A besucht wird. A → B → C → D → E → A schreibst du auf (B vor E), A → E → D → C → B → A nicht (E nach B) – sie ist ja dieselbe Reise rückwärts.

Arbeitsauftrag 15: Neue Probleminstanz mit fünf Orten

1. Stelle eine eigene Entfernungstabelle für **fünf Orte** A bis E auf. Die Werte darfst du frei erfinden, sie sollten aber plausibel sein. Wenn du keine eigenen Werte erfinden möchtest, nimm die Tabelle und den Graphen unten: Die Orte A bis D sind die bekannten, Ort E ist neu dazugekommen.
2. Bestimme mit dem Brute-Force-Ansatz die optimale Rundreise: Liste die 12 verschiedenen Rundreisen auf, berechne ihre Längen und wähle die kürzeste.

	A	B	C	D	E
A	–	10	15	20	28
B	10	–	35	25	32
C	15	35	–	30	18
D	20	25	30	–	32
E	28	32	18	32	–

Gymnasium Ottweiler



Vorschlag für eine Probleminstanz mit fünf Orten (Entfernungen in km)

Trage deine Rundreisen hier ein:

Nr.	Rundreise	Rechnung	Länge
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			

Optimale Rundreise und ihre Länge:

Du hast es sicher gemerkt: Mit fünf Orten ist Brute-Force schon richtig Arbeit. Statt 3 Rundreisen waren es 12. Und das bei nur einem Ort mehr!

3.2.5 Wie viele Rundreisen gibt es?

Die Anzahl der Rundreisen kannst du ausrechnen, ohne sie aufzuschreiben. Bei n Orten legst du den Startort fest. Dann gibt es

- für den zweiten Ort $n-1$ Möglichkeiten,
- für den dritten Ort $n-2$ Möglichkeiten,
- und so weiter, bis für den letzten Ort nur noch 1 Möglichkeit bleibt.

Insgesamt sind es $(n-1)\cdot(n-2)\cdot\ldots\cdot 2\cdot 1$ Rundreisen. Für so ein Produkt gibt es in der Mathematik eine Abkürzung, die **Fakultät**: Man schreibt die Zahl mit einem Ausrufezeichen dahinter.

Definition: Fakultät

Die **Fakultät** einer natürlichen Zahl k ist das Produkt aller natürlichen Zahlen von 1 bis k. Man schreibt $k!$ (gelesen: „k Fakultät“). Beispiele: $3! = 3 \cdot 2 \cdot 1 = 6$ und $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$. Für $k = 1$ ist $1! = 1$. Außerdem legt man fest: $0! = 1$.

Bei n Orten gibt es mit festem Startort also $(n-1)!$ Rundreisen. Ist die Entfernungstabelle symmetrisch, sind davon nur $(n-1)! : 2$ verschieden (ab drei Orten).

Stolperstein

Die Klammer in $(n-1)!$ ist wichtig. Bei $n = 5$ Orten ist $(5-1)! = 4! = 24$. Ohne Klammer stünde $n - 1! = 5 - 1 = 4$ da – das ist etwas ganz anderes. Rechne immer zuerst die Klammer aus und dann die Fakultät.

Arbeitsauftrag 16: Wachstum beobachten

1. Berechne die Anzahl der Rundreisen bei 4, 5 und 6 Orten (Startort fest) und trage sie in die Tabelle ein. Ergänze auch die Anzahl der verschiedenen Rundreisen, wenn man die Symmetrie berücksichtigt.
2. Was fällt dir am Wachstum auf? Um welchen Faktor wächst die Anzahl, wenn ein Ort dazukommt?

Orte n	Rechnung	Rundreisen $(n-1)!$	verschieden (mit Symmetrie)
4			
5			
6			

Deine Beobachtung:

Ein solches Wachstum heißt **faktorielles Wachstum**. Schon bei 10 Orten gibt es $9! = 362\,880$ Rundreisen. Wie lange selbst ein Computer dafür braucht, untersuchst du im Abschnitt über den Aufwand.

3.2.6 Für Interessierte: Brute-Force mit Python

Dieser Unterabschnitt ist freiwillig. Mit einem kleinen Programm kannst du deine Handrechnung kontrollieren.

Zuerst muss die Entfernungstabelle ins Programm. Dafür benutzen wir eine Liste, deren Einträge selbst wieder Listen sind – für jede Zeile der Tabelle eine.

Werkzeug

Eine **Tabelle als Liste von Listen:** In `entfernung = [[0, 10], [10, 0]]` ist `entfernung[0]` die erste Zeile, also die Liste `[0, 10]`. Mit einem zweiten Index wählst du darin die Spalte: `entfernung[0][1]` ist `10`. Die Orte bekommen Nummern: A ist 0, B ist 1, C ist 2, D ist 3. Die Entfernung von B nach C steht dann in `entfernung[1][2]`. Auf der Diagonale steht 0 statt eines Strichs.

```
namen = ["A", "B", "C", "D"]
entfernung = [[0, 10, 15, 20],
              [10, 0, 35, 25],
              [15, 35, 0, 30],
              [20, 25, 30, 0]]

def laenge(reise):
    summe = 0
    for i in range(len(reise) - 1):
        summe += entfernung[reise[i]][reise[i + 1]]
    return summe

def als_text(reise):
    text = namen[reise[0]]
    for i in range(1, len(reise)):
        text = text + " -> " + namen[reise[i]]
    return text

kuerzeste = 1000000
beste_reise = []
for ort2 in [1, 2, 3]:
    for ort3 in [1, 2, 3]:
        for ort4 in [1, 2, 3]:
            if ort2 != ort3 and ort2 != ort4 and ort3 != ort4:
                reise = [0, ort2, ort3, ort4, 0]
                strecke = laenge(reise)
                print(als_text(reise), "Länge:", strecke)
                if strecke < kuerzeste:
                    kuerzeste = strecke
                    beste_reise = reise

print("Optimal:", als_text(beste_reise), "mit Länge", kuerzeste)
```

AUSGABE

```
A -> B -> C -> D -> A Länge: 95
A -> B -> D -> C -> A Länge: 80
A -> C -> B -> D -> A Länge: 95
A -> C -> D -> B -> A Länge: 80
A -> D -> B -> C -> A Länge: 95
A -> D -> C -> B -> A Länge: 95
Optimal: A -> B -> D -> C -> A mit Länge 80
```

- **Zeilen 1–5:** die Namen der Orte und die Entfernungstabelle als Liste von Listen.
- **Zeilen 7–11:** Die Funktion `laenge` bekommt eine Rundreise als Liste von Ortsnummern, zum Beispiel `[0, 1, 2, 3, 0]`. Sie addiert die Entfernungen aller Teilstrecken von `reise[i]` nach `reise[i + 1]`. Weil die Liste mit 0 beginnt und endet, ist die Rückfahrt nach A schon dabei.
- **Zeilen 13–17:** Die Funktion `als_text` macht aus den Nummern einen lesbaren Text wie `A -> B -> C -> D -> A`.
- **Zeilen 19–20:** `kuerzeste` startet mit einer riesigen Zahl. So ist die erste gefundene Rundreise auf jeden Fall kürzer und wird übernommen.
- **Zeilen 21–24:** Wie im Abschnitt über das Auflisten erzeugen drei Schleifen alle Reihenfolgen. Nur solche ohne doppelte Orte sind Rundreisen (**Erzeugen**).
- **Zeilen 25–27:** Die Rundreise wird gebaut, ihre Länge berechnet (**Bewerten**) und ausgegeben.
- **Zeilen 28–30:** Ist die neue Rundreise kürzer als die bisher beste, wird sie gemerkt (**Auswählen**).
- **Zeile 32:** Am Ende steht die optimale Rundreise fest.

Die Rückwärts-Variante A → C → D → B → A ist ebenfalls 80 km lang. Sie wird nicht übernommen, weil `80 < 80` falsch ist. Das Programm merkt sich also die erste optimale Rundreise, die es findet.

Arbeitsauftrag 17 (Zusatz): Programm für fünf Orte

Passe das Programm für deine Probleminstanz mit fünf Orten an (oder für die vorgeschlagene Tabelle). Das Programm soll nur die optimale Rundreise und die Anzahl der geprüften Rundreisen ausgeben. Tipp: Statt der langen Bedingung kannst du eine Funktion `ist_rundreise(reise)` schreiben, die prüft, ob jeder Ort 1 bis 4 in der Liste vorkommt (mit `in`).

3.2.7 Weitere Übungen

Hier kannst du das Gelernte weiter üben. Die Punkte im Titel zeigen, wie schwer eine Aufgabe ist: ●○○ leicht, ●●○ mittel, ●●● knifflig. Fang mit den leichten an.

In mehreren Aufträgen geht es um Mias Partyeinkauf. Sie startet und endet zu Hause (Z) und fährt zur Bäckerei (B), zum Getränkemarkt (G) und zum Supermarkt (S). Die Entfernungen in km:

	Z	B	G	S
Z	–	3	7	6
B	3	–	9	5
G	7	9	–	4
S	6	5	4	–

Arbeitsauftrag 18: Länge einer Rundreise ●○○

Berechne die Länge der beiden Rundreisen. Schreibe zu jeder Teilstrecke die beiden Orte dazu.

1. Z → G → S → B → Z
2. Z → S → B → G → Z

Deine Rechnungen:

Arbeitsauftrag 19: Zulässige Lösung beim Partyeinkauf? ●○○

Mia muss zu Hause starten, jedes Geschäft genau einmal besuchen und wieder nach Hause fahren. Entscheide bei jedem Vorschlag, ob er eine zulässige Lösung ist. Begründe, wenn er es nicht ist.

Vorschlag	zulässig?	Begründung
Z → B → G → S → Z		
Z → B → S → Z		
Z → G → S → G → B → Z		
Z → S → G → B		
Z → S → B → G → Z		

Arbeitsauftrag 20: Fakultäten berechnen ●○○

1. Berechne die Anzahl der Rundreisen (Startort fest) für 3, 7 und 8 Orte. Ergänze auch die Anzahl der verschiedenen Rundreisen bei einer symmetrischen Tabelle.
2. Bei wie vielen Orten gibt es genau 720 Rundreisen?
3. Bei wie vielen Orten gibt es mit Symmetrie genau 60 verschiedene Rundreisen?

Orte n	Rechnung	Rundreisen $(n-1)!$	verschieden (mit Symmetrie)
3			
7			
8			

Deine Antworten zu 2 und 3:

Arbeitsauftrag 21: Die Fachbegriffe ●○○

Ergänze die Sätze mit den passenden Fachbegriffen.

Satzanfang	Ergänzung
Eine Lösung, die alle Bedingungen des Problems erfüllt, heißt …	
Die beste zulässige Lösung bezüglich eines Kriteriums heißt …	
Ein Ansatz, der alle zulässigen Lösungen erzeugt, bewertet und die beste auswählt, heißt …	
Beim Problem des Handlungsreisenden ist das Kriterium meistens …	
Der große Vorteil von Brute-Force ist, dass …	
Der große Nachteil von Brute-Force ist, dass …	

Arbeitsauftrag 22: Welche Verbindung ist optimal? ●●○

Lara will mit der Bahn zum Freizeitpark. Die App zeigt drei Verbindungen. Lara hat höchstens 20 € für die Fahrt.

Verbindung	Fahrzeit	Umstiege	Preis
V1	1 h 40 min	2	18 €
V2	2 h 10 min	0	12 €
V3	1 h 25 min	3	22 €

1. Welche Verbindungen sind zulässige Lösungen? Begründe.

2. Bestimme die optimale Verbindung, wenn das Kriterium (a) die Fahrzeit, (b) die Anzahl der Umstiege, (c) der Preis ist.
3. Erkläre an diesem Beispiel, warum man bei „optimal“ immer das Kriterium angeben muss.

Deine Antworten:

Arbeitsauftrag 23: Brute-Force beim Partyeinkauf ●●○

Bestimme mit dem Brute-Force-Ansatz die optimale Rundreise für Mias Partyeinkauf.

1. Schreibe alle 6 Rundreisen mit Start in Z auf (oder nimm deine Liste aus dem Auftrag *Rundreisen beim Partyeinkauf*).
2. Berechne ihre Längen. Zwei kennst du schon aus dem Auftrag *Länge einer Rundreise*.
3. Wähle die kürzeste Rundreise aus. Ist sie die einzige optimale Lösung?

Rundreise	Rechnung	Länge (km)

Optimale Rundreise:

Arbeitsauftrag 24: Fehler in der Brute-Force-Tabelle ●●○

Tom hat Brute-Force für Mias Partyeinkauf so aufgeschrieben:

Rundreise	Toms Rechnung	Toms Länge
Z → B → G → S → Z	$3 + 9 + 4 + 6$	22
Z → B → S → G → Z	$3 + 5 + 4$	12
Z → G → B → S → Z	$7 + 9 + 5 + 6$	27
Z → G → S → B → Z	$7 + 4 + 5 + 3$	18
Z → B → G → S → Z	$3 + 9 + 4 + 6$	22
Z → S → G → B → Z	$6 + 4 + 9 + 3$	22

Toms Ergebnis: „Optimal ist Z → B → S → G → Z mit 12 km.“

Finde **drei** Fehler in Toms Tabelle. Erkläre jeweils, was falsch ist, und gib das richtige Ergebnis an.

Toms Fehler:

Gymnasium Ottweiler

Arbeitsauftrag 25: Radtour mit fünf Zielen ●●●

Eine Klasse übernachtet in einer Jugendherberge (J). Mit dem Fahrrad will sie die Burg (B), den See (S), den Turm (T) und den Wasserfall (W) besuchen und dann zurückfahren. Die Entfernungen in km:

	J	B	S	T	W
J	–	9	11	10	13
B	9	–	6	14	4
S	11	6	–	15	9
T	10	14	15	–	12
W	13	4	9	12	–

1. Wie viele Rundreisen mit Start in J gibt es? Wie viele davon sind verschieden?
2. Die Tabelle unten enthält die 12 verschiedenen Rundreisen. Sechs Längen sind schon berechnet. Berechne die übrigen sechs.
3. Bestimme die optimale Rundreise. Wie viel länger ist die längste Rundreise?

Nr.	Rundreise	Rechnung	Länge (km)
1	J → B → S → T → W → J	$9+6+15+12+13$	55
2	J → B → S → W → T → J	$9+6+9+12+10$	46
3	J → B → T → S → W → J	$9+14+15+9+13$	60
4	J → B → T → W → S → J	$9+14+12+9+11$	55
5	J → B → W → S → T → J	$9+4+9+15+10$	47
6	J → B → W → T → S → J	$9+4+12+15+11$	51
7	J → S → B → T → W → J		
8	J → S → B → W → T → J		
9	J → S → T → B → W → J		
10	J → S → W → B → T → J		
11	J → T → B → S → W → J		
12	J → T → S → B → W → J		

Deine Antworten zu 1 und 3:

3.2.8 Das Wichtigste in Kürze

- Eine **zulässige Lösung** erfüllt alle Bedingungen des Problems. Beim Handlungsreisenden: jeder Ort genau einmal, zurück zum Start.
- Eine **optimale Lösung** ist eine zulässige Lösung, die bezüglich eines festgelegten **Kriteriums** (z. B. Gesamtlänge) die beste ist. Es kann mehrere optimale Lösungen geben.
- Die Länge einer Rundreise ist die Summe aller Teilstrecken – die Rückfahrt zum Start gehört dazu.
- Der **Brute-Force-Ansatz** erzeugt alle zulässigen Lösungen, bewertet sie und wählt die beste. Er findet garantiert eine optimale Lösung.
- Bei n Orten gibt es mit festem Start $(n-1)!$ Rundreisen, mit Symmetrie $(n-1)! : 2$ verschiedene. Diese Anzahl wächst **faktoriell**, also sehr schnell.
- Für die vier Orte A, B, C, D ist A → B → D → C → A mit 80 km optimal.

Selbstcheck

- ☐ Ich kann prüfen, ob ein Vorschlag eine zulässige Lösung ist.
- ☐ Ich kann erklären, was eine optimale Lösung ist, und weiß, dass dazu ein Kriterium gehört.
- ☐ Ich kann die Länge einer Rundreise mit einer Entfernungstabelle berechnen.
- ☐ Ich kann den Brute-Force-Ansatz in drei Schritten beschreiben.
- ☐ Ich kann mit Brute-Force die optimale Rundreise durch vier oder fünf Orte bestimmen.
- ☐ Ich kann die Anzahl der Rundreisen mit $(n-1)!$ berechnen und erklären, warum Brute-Force bei vielen Orten unbrauchbar wird.

Gymnasium Ottweiler

3.3 Heuristiken: schnell eine gute Lösung finden

Leitidee

Brute-Force findet immer die beste Rundreise – aber schon bei wenigen Orten wird das Ausprobieren sehr mühsam. Eine Paketbotin mit 100 Adressen kann unmöglich alle Reihenfolgen durchrechnen. Was macht sie stattdessen? Vermutlich etwas ganz Naheliegendes: Sie fährt immer zur nächstgelegenen Adresse, die sie noch nicht beliefert hat. So eine Faustregel heißt in der Informatik **Heuristik**. In diesem Abschnitt lernst du, wie solche Faustregeln funktionieren, wendest eine davon von Hand an und prüfst, wie gut ihr Ergebnis ist.

Lernziele

Nach diesem Abschnitt kannst du

- den Begriff Heuristik definieren und an einem Beispiel erklären,
- beschreiben, was eine Greedy-Heuristik auszeichnet,
- die Nearest-Neighbor-Heuristik von Hand auf eine Probleminstanz anwenden,
- das Ergebnis einer Heuristik mit der optimalen Lösung vergleichen und die Abweichung in Prozent berechnen,
- Vor- und Nachteile von Heuristik und Brute-Force gegenüberstellen.

3.3.1 Faustregeln statt Ausprobieren

Im Alltag benutzt du ständig Faustregeln. An der Supermarktkasse stellst du dich an die kürzeste Schlange. Das geht schnell und klappt meistens gut. Eine Garantie gibt es aber nicht: Vielleicht hat die Person vor dir einen riesigen Einkauf, und an der Nachbarkasse wärst du schneller gewesen. Um sicherzugehen, müsstest du alle Einkaufswagen genau ansehen – das dauert länger als das Warten selbst.

Genau solche Faustregeln nennt man in der Informatik **Heuristiken**. Das Wort kommt vom griechischen *heuriskein*, „finden“. Eine Heuristik verzichtet darauf, alle Möglichkeiten zu prüfen. Dafür ist sie schnell.

Definition: Heuristik

Eine **Heuristik** ist eine Lösungsstrategie, die **schnell** eine zulässige Lösung liefert, ohne alle Möglichkeiten zu prüfen. Die Lösung ist meist gut, aber **nicht garantiert optimal**.

Merke

Eine **Heuristik** ist eine Lösungsstrategie, die **schnell** eine (meist gute, aber nicht garantiert optimale) Lösung liefert, ohne alle Möglichkeiten zu prüfen.

3.3.2 Greedy: immer das, was gerade am besten aussieht

Eine besonders einfache Art von Heuristik geht so vor: Sie baut die Lösung Schritt für Schritt auf. In jedem Schritt nimmt sie das, was **im Moment** am besten aussieht. Was später passiert, bedenkt sie nicht. Eine einmal getroffene Entscheidung wird nie mehr geändert.

Solche Heuristiken heißen **Greedy-Heuristiken** (englisch *greedy*, gesprochen „griedi“, heißt „gierig“). Denk an ein Kind am Buffet, das sich zuerst das größte Stück Kuchen nimmt – ohne zu überlegen, ob danach noch Platz für den Nachtisch bleibt.

Definition: Greedy-Heuristik

Eine **Greedy-Heuristik** baut eine Lösung schrittweise auf. In jedem Schritt trifft sie die Entscheidung, die **im Moment** am besten erscheint, ohne die Auswirkungen auf spätere Schritte zu berücksichtigen.

Eine Greedy-Strategie hast du übrigens schon in Klasse 9 kennengelernt: Beim Lernen eines Entscheidungsbaums wählt man in jedem Schritt das Merkmal, das die Daten im Moment am besten aufteilt. Ob eine andere Wahl später zu einem besseren Baum geführt hätte, prüft man nicht.

Merke

Eine **Greedy-Heuristik** („gierig“) trifft in jedem Schritt die Entscheidung, die im **Moment** am besten erscheint, ohne spätere Auswirkungen zu berücksichtigen.

3.3.3 Die Nearest-Neighbor-Heuristik

Für das Problem des Handlungsreisenden gibt es eine naheliegende Greedy-Heuristik. Sie heißt **Nearest Neighbor** (englisch für „nächster Nachbar“, gesprochen „nierest neiber“). Ihre Regel lautet: Fahre immer zum nächstgelegenen Ort, an dem du noch nicht warst.

So gehst du Schritt für Schritt vor:

1. Beginne im Startort. Er ist der **aktuelle Ort**.
2. Suche unter allen **noch nicht besuchten** Orten den, der vom aktuellen Ort am wenigsten weit entfernt ist. Sind zwei Orte gleich weit entfernt, nimm den, der im Alphabet zuerst kommt.
3. Fahre dorthin. Dieser Ort ist jetzt der aktuelle Ort und gilt als besucht.
4. Wiederhole die Schritte 2 und 3, bis alle Orte besucht sind.
5. Fahre zurück zum Startort.

Merke

Beim Problem des Handlungsreisenden wählt die Greedy-Heuristik **Nearest Neighbor** von der aktuellen Stadt aus immer die **nächstgelegene, noch nicht besuchte** Stadt.

Beispiel: Nearest Neighbor von Hand, Start in A

Wir benutzen wieder die Entfernungstabelle der vier Orte:

	A	B	C	D
A	–	10	15	20
B	10	–	35	25
C	15	35	–	30
D	20	25	30	–

Schritt	aktueller Ort	noch nicht besucht (Entfernung)	Wahl
1	A	B 10, C 15, D 20	B (10 km)
2	B	C 35, D 25	D (25 km)
3	D	C 30	C (30 km)
4	C	alle besucht	zurück nach A (15 km)

Rundreise: A → B → D → C → A. Länge: $10 + 25 + 30 + 15 = 80$ km.

Lies die Tabelle Zeile für Zeile:

- **Schritt 1:** Von A aus ist B am nächsten (10 km). C und D sind weiter weg.
- **Schritt 2:** In B schaust du nur noch auf C und D, denn A ist schon besucht. D ist mit 25 km näher als C mit 35 km.
- **Schritt 3:** Von D aus bleibt nur noch C übrig. Hier gibt es nichts zu entscheiden.
- **Schritt 4:** Alle Orte sind besucht. Zum Schluss geht es zurück zum Start.

Vergleiche mit dem Ergebnis von Brute-Force aus dem letzten Abschnitt: Die optimale Rundreise ist A → B → D → C → A mit 80 km. Die Heuristik hat hier also mit nur vier kurzen Blicken in die Tabelle die optimale Lösung gefunden. Ist das immer so? Das untersuchst du jetzt.

Stolperstein

Die drei häufigsten Fehler bei Nearest Neighbor:

- Du wählst einen Ort, der **schon besucht** ist (meistens den Startort). Streiche besuchte Orte am besten in der Tabelle durch.
- Du vergisst am Ende die **Rückfahrt** zum Start.
- Du schaust „voraus“ und wählst einen Ort, weil es danach günstiger weitergeht. Das ist zwar klug, aber nicht mehr die Nearest-Neighbor-Heuristik. Sie schaut nur auf den aktuellen Schritt.

Arbeitsauftrag 26: Anderer Startpunkt

Wende die Nearest-Neighbor-Heuristik **von Hand** an, aber starte diesmal in Ort **C**. Lege dazu eine Tabelle an wie im Beispiel. Vergleiche Rundreise und Länge mit dem Ergebnis für den Start in A. Was fällt dir auf?

Deine Tabelle und dein Vergleich:

Arbeitsauftrag 27: Start in D

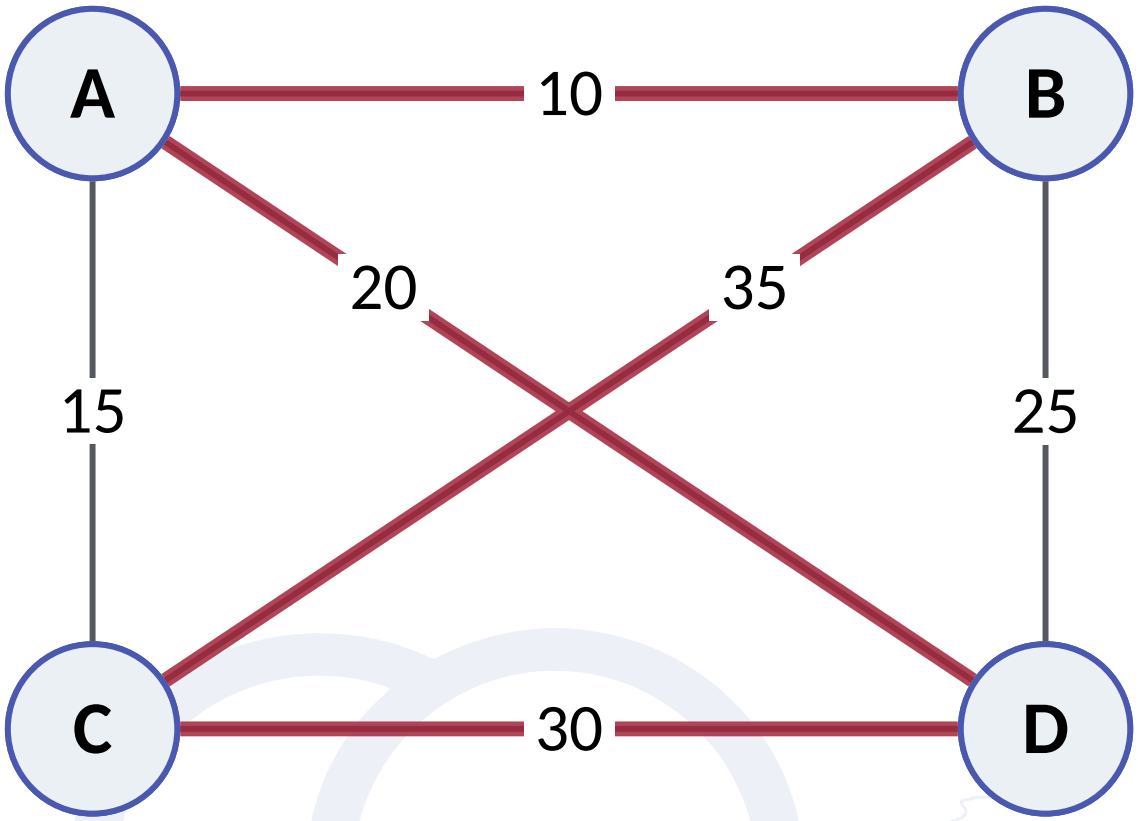
Wende die Nearest-Neighbor-Heuristik noch einmal an, jetzt mit Start in **D**. Ist das Ergebnis optimal? Erkläre, an welcher Stelle die Heuristik „in eine Falle läuft“.

Gymnasium Ottweiler

Deine Tabelle und deine Erklärung:

Die Heuristik liefert also je nach Startort unterschiedlich gute Ergebnisse. Bei Start in A, B oder C kommt sie auf 80 km, bei Start in D auf 95 km. Im Graphen siehst du, woran es liegt: Die Rundreise vom Start D aus benutzt die teure Kante B–C.

Gymnasium Ottweiler



Nearest Neighbor mit Start in D liefert D → A → B → C → D (rot) mit $20 + 10 + 35 + 30 = 95$ km. Optimal wären 80 km

Merke

Heuristiken sind **schneller** als Brute-Force, liefern aber **nicht immer** die optimale Lösung – manchmal führt eine „im Moment beste“ Entscheidung in eine Sackgasse.

3.3.4 Wie gut ist die Heuristik? Die Abweichung in Prozent

„95 km statt 80 km“ – ist das viel oder wenig? Um das zu beurteilen, rechnet man aus, um **wie viel Prozent** die Lösung der Heuristik schlechter ist als die optimale Lösung. Das kennst du aus Mathe als prozentuale Änderung:

$$\text{Abweichung} = \frac{\text{Länge der Heuristik} - \text{optimale Länge}}{\text{optimale Länge}} \cdot 100\,\%$$

Für den Start in D:

$$\text{Abweichung} = \frac{95 - 80}{80} \cdot 100\,\% = \frac{15}{80} \cdot 100\,\% = 18{,}75\,\% \approx 18{,}8\,\%$$

Die Rundreise der Heuristik ist also knapp 19 % länger als nötig. Bei einer Abweichung von 0 % hat die Heuristik die optimale Lösung getroffen.

Stolperstein

Teile immer durch die **optimale** Länge, nicht durch die Länge der Heuristik. $\frac{15}{95} \approx 15{,}8\,\%$ wäre falsch. Die Frage lautet ja: Um wie viel ist die Heuristik schlechter **als das Optimum**?

Arbeitsauftrag 28: Heuristik und Optimum

Nimm deine eigene Probleminstanz mit fünf Orten aus dem Auftrag *Neue Probleminstanz mit fünf Orten* (oder die dort vorgeschlagene Tabelle).

1. Wende die Nearest-Neighbor-Heuristik mit Start in A an. Gib die Rundreise und ihre Länge an.
2. Vergleiche mit der optimalen Rundreise, die du mit Brute-Force gefunden hast. Berechne die **prozentuale Abweichung**.

Deine Tabelle und deine Rechnung:

Gymnasium Ottweiler

3.3.5 Heuristik oder Brute-Force?

Warum benutzt man überhaupt Heuristiken, wenn sie nicht immer optimal sind? Weil sie **viel weniger Arbeit** machen. Zähle einmal, wie oft man bei Nearest Neighbor eine Entfernung ansehen muss: Bei vier Orten sind es im ersten Schritt 3 Entfernungen, im zweiten 2 und im dritten 1, zusammen also 6. Bei 10 Orten sind es $9+8+7+\ldots+1 = 45$. Brute-Force müsste bei 10 Orten dagegen $9! = 362\,880$ Rundreisen durchrechnen.

	Brute-Force	**Nearest Neighbor**
Ergebnis	garantiert optimal	zulässig, meist gut, nicht immer optimal
Arbeit bei 4 Orten	6 Rundreisen berechnen	6 Entfernungen ansehen
Arbeit bei 10 Orten	362 880 Rundreisen berechnen	45 Entfernungen ansehen
geeignet für	kleine Probleminstanzen	auch große Probleminstanzen

So kannst du es sagen

So vergleichst du die beiden Ansätze in Fachsprache:

- *„Die Heuristik ist schneller, aber nicht so gut."* → *„Die Nearest-Neighbor-Heuristik benötigt deutlich weniger Rechenschritte als der Brute-Force-Ansatz, liefert aber nicht immer eine optimale Lösung."*
- *„Hier war sie 15 km schlechter."* → *„Bei Start in D weicht die Lösung der Heuristik um 18,75 % von der optimalen Lösung ab."*

3.3.6 Für Interessierte: Nearest Neighbor mit Python

Dieser Unterabschnitt ist freiwillig. Die Entfernungstabelle und die Funktionen `laenge` und `als_text` kennst du aus dem Abschnitt über Brute-Force. Neu ist die Funktion `nearest_neighbor`.

```
1  namen = ["A", "B", "C", "D"]
2  entfernung = [[0, 10, 15, 20],
3                [10, 0, 35, 25],
4                [15, 35, 0, 30],
5                [20, 25, 30, 0]]
6
7  def laenge(reise):
8      summe = 0
9      for i in range(len(reise) - 1):
10         summe += entfernung[reise[i]][reise[i + 1]]
11     return summe
12
13 def als_text(reise):
14     text = namen[reise[0]]
15     for i in range(1, len(reise)):
16         text = text + " -> " + namen[reise[i]]
17     return text
18
19 def nearest_neighbor(start):
20     reise = [start]
21     aktuell = start
22     while len(reise) < len(namen):
23         naechster = -1
```

```
24          kleinste = 1000000
25          for ort in range(len(namen)):
26              if ort not in reise:
27                  if entfernung[aktuell][ort] < kleinste:
28                      kleinste = entfernung[aktuell][ort]
29                      naechster = ort
30          reise.append(naechster)
31          aktuell = naechster
32      reise.append(start)
33      return reise
34
35  for start in range(len(namen)):
36      reise = nearest_neighbor(start)
37      print(als_text(reise), "Länge:", laenge(reise))
```

AUSGABE

```
A -> B -> D -> C -> A Länge: 80
B -> A -> C -> D -> B Länge: 80
C -> A -> B -> D -> C Länge: 80
D -> A -> B -> C -> D Länge: 95
```

- **Zeilen 19–21:** Die Funktion bekommt die Nummer des Startorts. Die Rundreise beginnt als Liste mit nur dem Startort. Er ist auch der aktuelle Ort.
- **Zeile 22:** Solange noch nicht alle Orte in der Liste stehen, wird ein weiterer Ort gesucht.
- **Zeilen 23–29:** Das ist die Suche nach dem nächsten Nachbarn. `kleinste` startet mit einer riesigen Zahl. Die Schleife geht alle Orte durch. `ort not in reise` ist wahr, wenn der Ort noch nicht besucht wurde. Ist er außerdem näher als der bisher nächste, merkt sich das Programm ihn. Bei gleicher Entfernung bleibt der zuerst gefundene Ort gemerkt, denn `<` ist nur bei einer echt kleineren Entfernung wahr. Das ist der Ort, der im Alphabet zuerst kommt – wie in der Regel oben.
- **Zeilen 30–31:** Der nächste Nachbar wird an die Reise angehängt und zum aktuellen Ort (**Greedy-Entscheidung**, die nie mehr geändert wird).
- **Zeilen 32–33:** Zum Schluss wird der Startort noch einmal angehängt (Rückfahrt), und die fertige Rundreise wird zurückgegeben.
- **Zeilen 35–37:** Die Heuristik wird für jeden Startort ausgeführt. Die Ausgabe bestätigt deine Handrechnungen.

Beachte: `in` funktioniert bei Listen genauso wie bei Zeichenketten. `3 in [0, 1, 3]` ist `True`.

Versuch: Ändere die Tabelle

Trage im Programm die Tabelle mit fünf Orten ein (eine fünfte Zeile, in jeder Zeile eine fünfte Zahl, und `"E"` in `namen`). Die Funktion `nearest_neighbor` musst du nicht ändern. Vergleiche die Ausgabe mit deiner Lösung zum Auftrag *Heuristik und Optimum*.

3.3.7 Übungen

Arbeitsauftrag 29: Nearest Neighbor von allen Startorten

Wende die Nearest-Neighbor-Heuristik auf die vier Orte A, B, C und D mit Start in **B** an. Fasse danach die Ergebnisse für alle vier Startorte in einer Tabelle zusammen (Startort, Rundreise, Länge, Abweichung vom Optimum 80 km).

Deine Rechnung und deine Tabelle:

Arbeitsauftrag 30 (Zusatz): Was würdest du empfehlen?

Ein Lieferdienst muss jeden Tag 40 Kundinnen und Kunden beliefern. Die Disponentin überlegt, ob sie mit einem Computer die optimale Route per Brute-Force berechnen lassen soll oder ob die Nearest-Neighbor-Heuristik reicht. Beurteile die beiden Möglichkeiten und gib eine begründete Empfehlung.

Gymnasium Ottweiler

Deine Beurteilung:

3.3.8 Weitere Übungen

Hier kannst du das Gelernte weiter üben. Die Punkte im Titel zeigen, wie schwer eine Aufgabe ist: ●○○ leicht, ●●○ mittel, ●●● knifflig. Fang mit den leichten an.

Arbeitsauftrag 31: Ausprobieren oder Faustregel? ●○○

Entscheide bei jeder Vorgehensweise, ob es sich um einen **Brute-Force-Ansatz** oder um eine **Heuristik** handelt. Gib bei den Heuristiken an, ob sie **greedy** sind, also in jedem Schritt das nehmen, was im Moment am besten aussieht.

Gymnasium Ottweiler

Vorgehensweise	Brute-Force oder Heuristik?	greedy?
Ben hat die Zahl seines Zahlenschlosses vergessen und probiert alle Zahlen von 000 bis 999 der Reihe nach aus.		
Beim Kartenspiel spielt Ida in jeder Runde ihre höchste Karte.		
Elif schreibt alle Reihenfolgen auf, in denen sie ihre vier Hausaufgaben machen kann, und wählt die, die am wenigsten Zeit kostet.		
Beim Wandern nimmt Paul an jeder Kreuzung den Weg, der am steilsten bergauf führt, um möglichst schnell zum Gipfel zu kommen.		
Eine Kassiererin gibt Wechselgeld, indem sie immer die größte Münze nimmt, die noch passt.		

Arbeitsauftrag 32: Nearest Neighbor beim Partyeinkauf ●○○

Mia startet zu Hause (Z) und fährt zur Bäckerei (B), zum Getränkemarkt (G) und zum Supermarkt (S). Die Entfernungen in km:

	Z	B	G	S
Z	–	3	7	6
B	3	–	9	5
G	7	9	–	4
S	6	5	4	–

Wende die Nearest-Neighbor-Heuristik mit Start in **Z** an. Fülle die Tabelle aus und gib die Rundreise und ihre Länge an.

Schritt	aktueller Ort	noch nicht besucht (Entfernung)	Wahl
1	Z		
2			
3			
4		alle besucht	

Rundreise und Länge:

Arbeitsauftrag 33: Start in der Bäckerei ●○○

Mias Freundin Ela wohnt direkt neben der Bäckerei. Sie startet deshalb in **B** und fährt am Ende wieder nach B zurück. Wende die Nearest-Neighbor-Heuristik mit Start in B an (Tabelle aus dem Auftrag *Nearest Neighbor beim Partyeinkauf*). Vergleiche die Länge mit der optimalen Rundreise (19 km). Um wie viele km ist Elas Rundreise länger?

Deine Tabelle und dein Vergleich:

Arbeitsauftrag 34: Stimmt das? ●○○

Entscheide, ob die Aussage richtig oder falsch ist. Begründe kurz.

Aussage	richtig oder falsch?	Begründung
Eine Heuristik prüft alle Möglichkeiten.		
Nearest Neighbor kann die optimale Rundreise finden.		
Nearest Neighbor liefert immer eine zulässige Rundreise.		
Eine Greedy-Heuristik ändert eine Entscheidung später noch, wenn sie sich als schlecht herausstellt.		
Heuristiken sind meistens schneller als Brute-Force.		

Arbeitsauftrag 35: Abweichung in Prozent ●●○

1. Berechne für jede Zeile die prozentuale Abweichung der Heuristik vom Optimum. Runde auf eine Stelle nach dem Komma.
2. Paul hat für die erste Zeile $\frac{22-19}{22} \cdot 100\,\% \approx 13{,}6\,\%$ gerechnet. Erkläre seinen Fehler.

Länge der Heuristik	optimale Länge	Rechnung	Abweichung
22 km	19 km		
60 km	50 km		
47 km	43 km		
30 km	30 km		

Pauls Fehler:

Arbeitsauftrag 36: Fehler im Protokoll ●●○

Lina hat Nearest Neighbor für die Radtour mit fünf Zielen (Auftrag *Radtour mit fünf Zielen* im Abschnitt über Brute-Force) mit Start in J angewendet. Hier ist die Tabelle noch einmal:

	J	B	S	T	W
J	–	9	11	10	13
B	9	–	6	14	4
S	11	6	–	15	9
T	10	14	15	–	12
W	13	4	9	12	–

Linas Protokoll:

Schritt	aktueller Ort	noch nicht besucht (Entfernung)	Wahl
1	J	B 9, S 11, T 10, W 13	B (9 km)
2	B	J 9, S 6, T 14, W 4	W (4 km)
3	W	S 9, T 12	T (12 km), weil es danach zum See geht
4	T	S 15	S (15 km)

Linas Ergebnis: J → B → W → T → S, Länge $9 + 4 + 12 + 15 = 40$ km.

1. Finde **drei** Fehler in Linas Protokoll und erkläre sie.
2. Führe die Heuristik richtig durch. Vergleiche mit der optimalen Rundreise (43 km) und berechne die Abweichung.

Linas Fehler und deine richtige Lösung:

Arbeitsauftrag 37: Die Falle selbst bauen ●●●

In dieser Entfernungstabelle fehlt eine Zahl: Die Entfernung zwischen R und S heißt x.

	P	Q	R	S
P	–	4	6	8
Q	4	–	7	5
R	6	7	–	x
S	8	5	x	–

1. Setze $x = 5$ ein. Wende Nearest Neighbor mit Start in P an. Bestimme mit Brute-Force die optimale Rundreise (3 verschiedene Rundreisen genügen). Ist die Heuristik optimal?
2. Setze $x = 20$ ein und wiederhole beides. Berechne die Abweichung.
3. Untersuche allgemein: Welche Rundreise wählt Nearest Neighbor, egal wie groß x ist? Ab welchem Wert von x ist sie nicht mehr optimal? Begründe.

Deine Rechnungen und deine Begründung:

Gymnasium Ottweiler

3.3.9 Das Wichtigste in Kürze

- Eine **Heuristik** liefert schnell eine zulässige, meist gute Lösung, ohne alle Möglichkeiten zu prüfen. Eine optimale Lösung ist nicht garantiert.
- Eine **Greedy-Heuristik** trifft in jedem Schritt die Entscheidung, die im Moment am besten aussieht, und ändert sie nie wieder.
- Die **Nearest-Neighbor-Heuristik** fährt immer zum nächstgelegenen, noch nicht besuchten Ort und am Ende zurück zum Start.
- Das Ergebnis kann vom **Startort** abhängen: Bei den vier Orten liefert der Start in D 95 km statt der optimalen 80 km.
- Die **Abweichung** berechnet man als (Länge der Heuristik − optimale Länge) : optimale Länge · 100 %.
- Heuristiken machen viel weniger Arbeit als Brute-Force. Deshalb verwendet man sie bei großen Probleminstanzen.

Selbstcheck

- ☐ Ich kann den Begriff Heuristik definieren und ein Beispiel aus dem Alltag nennen.
- ☐ Ich kann erklären, was „greedy“ bei einer Heuristik bedeutet.
- ☐ Ich kann die Nearest-Neighbor-Heuristik von Hand mit einer Tabelle durchführen.
- ☐ Ich kann die Abweichung einer Heuristik vom Optimum in Prozent berechnen.
- ☐ Ich kann an einem Beispiel erklären, warum eine Greedy-Entscheidung in eine Sackgasse führen kann.
- ☐ Ich kann begründen, wann man eine Heuristik statt Brute-Force einsetzt.

Gymnasium Ottweiler

3.4 Das Rucksackproblem: Brute-Force und Greedy im Vergleich

Leitidee

Du planst eine Wanderung mit Übernachtung. Zelt, Schlafsack, Kocher, Proviant und Erste-Hilfe-Set liegen bereit – aber alles zusammen ist zu schwer. Dein Rucksack soll höchstens 10 kg wiegen. Was nimmst du mit, damit der Rucksack möglichst nützlich gefüllt ist? Diese Frage heißt **Rucksackproblem**. Es ist ganz anders aufgebaut als das Problem des Handlungsreisenden: Hier gibt es keine Orte und keine Reihenfolge, sondern nur die Frage „mitnehmen oder nicht?“. Trotzdem helfen dieselben Werkzeuge: Brute-Force findet garantiert die beste Auswahl, eine Greedy-Heuristik findet schnell eine gute. In diesem Abschnitt wendest du beide Ansätze an und vergleichst sie.

Lernziele

Nach diesem Abschnitt kannst du

- das Rucksackproblem beschreiben und zulässige und optimale Lösungen dafür angeben,
- erklären, warum es bei n Gegenständen 2^n mögliche Auswahlen gibt,
- den Brute-Force-Ansatz für eine kleine Rucksack-Instanz von Hand durchführen,
- das Wert-Gewicht-Verhältnis berechnen und die Greedy-Heuristik von Hand anwenden,
- Brute-Force und Greedy-Heuristik nach Ergebnis und Aufwand vergleichen,
- eine Probleminstanz konstruieren, bei der die Greedy-Heuristik nicht die optimale Lösung findet.

3.4.1 Das Problem: Was kommt in den Rucksack?

Beim Rucksackproblem hat jeder Gegenstand zwei Eigenschaften: ein **Gewicht** und einen **Wert**. Der Wert gibt an, wie nützlich oder kostbar der Gegenstand ist. Hier messen wir ihn in Euro. Der Rucksack hat eine **Tragkraft**, also ein Höchstgewicht. Gesucht ist eine Auswahl, die möglichst viel wert ist und trotzdem nicht zu schwer.

Auf Englisch heißt das Problem *Knapsack Problem* (gesprochen „näpsäck“, *knapsack* ist ein altes Wort für Rucksack). Es steckt in vielen Alltagsfragen: Welche Pakete lädt man in einen Lieferwagen? Welche Projekte finanziert eine Firma mit ihrem Budget? Welche Fotos passen noch auf die Speicherkarte?

Merke

Beim **Rucksackproblem** (*Knapsack Problem*) stehen mehrere Gegenstände mit jeweils einem **Gewicht** und einem **Wert** zur Verfügung. Gesucht ist die Auswahl von Gegenständen mit **maximalem Gesamtwert**, deren **Gesamtgewicht** eine vorgegebene Tragkraft nicht übersteigt.

Beispiel: Problem und Probleminstanz

Problem: Gegeben sind beliebige Gegenstände mit Gewicht und Wert und ein Rucksack mit einer Tragkraft. Finde eine Auswahl der Gegenstände mit möglichst großem Gesamtwert, deren Gesamtgewicht die Tragkraft nicht übersteigt.

Probleminstanz: Tragkraft **10 kg** und diese fünf Gegenstände:

Gegenstand	Gewicht (kg)	Wert (€)
Zelt	5	60
Schlafsack	4	40
Kocher	2	30
Proviant	3	25
Erste-Hilfe-Set	1	20

Die Probleminstanz ist als **Liste** abstrahiert dargestellt: Farbe, Marke oder Größe der Gegenstände spielen keine Rolle. Die Begriffe aus den letzten Abschnitten passen auch hier:

- Eine **zulässige Lösung** ist eine Auswahl, deren Gesamtgewicht höchstens 10 kg beträgt.
- Das **Kriterium** ist der Gesamtwert. Eine **optimale Lösung** ist eine zulässige Auswahl mit dem **größten** Gesamtwert.

Beachte den Unterschied zum Handlungsreisenden: Dort war die kleinste Länge am besten, hier ist der größte Wert am besten. Beides sind **Optimierungsprobleme** – einmal wird etwas möglichst klein gemacht, einmal möglichst groß.

Arbeitsauftrag 38: Erste Kombinationen

Finde durch Ausprobieren **mindestens zwei** zulässige Kombinationen (Gesamtgewicht höchstens 10 kg) für die Probleminstanz oben. Berechne jeweils Gesamtgewicht und Gesamtwert. Welche deiner Kombinationen ist die beste?

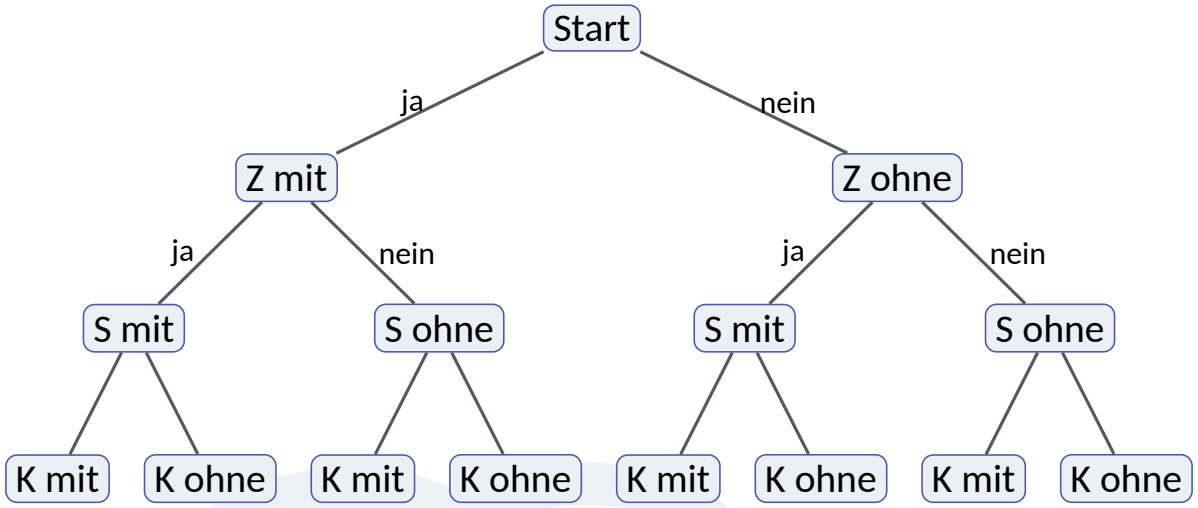
Kombination	Gesamtgewicht (kg)	Gesamtwert (€)

Gymnasium Ottweiler

Meine beste Kombination:

3.4.2 Brute-Force: alle Auswahlen ausprobieren

Wie viele verschiedene Auswahlen gibt es überhaupt? Gehe die Gegenstände der Reihe nach durch. Bei jedem Gegenstand gibt es genau **zwei** Möglichkeiten: Er kommt mit, oder er bleibt da. Bei drei Gegenständen kannst du das als Baum zeichnen:



Bei drei Gegenständen (Z = Zelt, S = Schlafsack, K = Kocher) gibt es $2 \cdot 2 \cdot 2 = 8$ Möglichkeiten. Jeder Weg von oben nach unten ist eine Auswahl

Mit jedem weiteren Gegenstand verdoppelt sich die Anzahl der Wege: Jeder bisherige Weg teilt sich in „mit“ und „ohne“. Bei vier Gegenständen sind es $2 \cdot 2 \cdot 2 \cdot 2 = 2^4 = 16$, bei fünf Gegenständen $2^5 = 32$ Auswahlen.

Eine Auswahl von Gegenständen nennt man in der Mathematik eine **Teilmenge**. Dazu zählen auch die leere Auswahl (nichts einpacken) und die volle Auswahl (alles einpacken).

Definition: Teilmenge

Eine **Teilmenge** einer Menge von Gegenständen ist eine beliebige Auswahl dieser Gegenstände. Auch die leere Auswahl und die Auswahl aller Gegenstände sind Teilmengen. Eine Menge mit n Gegenständen hat 2^n Teilmengen.

Merke

Brute-Force-Ansatz beim Rucksackproblem: Alle möglichen **Teilmengen** der Gegenstände werden gebildet und bewertet – bei n Gegenständen gibt es 2^n Teilmengen (**exponentieller Aufwand**).

Exponentiell heißt: Die Anzahl der Gegenstände steht im **Exponenten**. Jeder zusätzliche Gegenstand verdoppelt die Arbeit. Bei 10 Gegenständen sind es schon $2^{10} = 1024$ Teilmengen.

Teilmengen systematisch aufschreiben

Damit keine Teilmenge fehlt, schreibst du jede Auswahl als Folge von Nullen und Einsen. Die Stellen gehören der Reihe nach zu Zelt (Z), Schlafsack (S), Kocher (K), Proviant (P) und Erste-Hilfe-Set (E). Eine **1** bedeutet „kommt mit“, eine **0** bedeutet „bleibt da“.

Zum Beispiel bedeutet `1 1 0 0 1`: Zelt, Schlafsack und Erste-Hilfe-Set kommen mit. Gewicht: $5+4+1 = 10$ kg, Wert: $60 + 40 + 20 = 120$ €.

Vielleicht erkennst du das Muster wieder: Es sind genau die fünfstelligen Binärzahlen aus dem Kapitel über Schaltungen. Wenn du von `1 1 1 1 1` rückwärts bis `0 0 0 0 0` zählst, bekommst du jede Teilmenge genau einmal.

Tipp

Du sparst Arbeit, wenn du zuerst das Gewicht berechnest. Liegt es über 10 kg, ist die Auswahl nicht zulässig. Dann musst du den Wert nicht mehr ausrechnen – trage einfach ✗ ein.

Arbeitsauftrag 39: Brute-Force von Hand

Führe den Brute-Force-Ansatz für die Probleminstanz mit der Tragkraft 10 kg durch.

1. Berechne für jede der 32 Teilmengen das Gesamtgewicht. Entscheide, ob sie zulässig ist (✓) oder nicht (✗).
2. Berechne für die zulässigen Teilmengen den Gesamtwert.
3. Markiere die optimale Lösung.

Teilmengen **mit** Zelt:

Gymnasium Ottweiler

Nr.	Z S K P E	Gewicht (kg)	Wert (€)	zulässig?
1	1 1 1 1 1			
2	1 1 1 1 0			
3	1 1 1 0 1			
4	1 1 1 0 0			
5	1 1 0 1 1			
6	1 1 0 1 0			
7	1 1 0 0 1			
8	1 1 0 0 0			
9	1 0 1 1 1			
10	1 0 1 1 0			
11	1 0 1 0 1			
12	1 0 1 0 0			
13	1 0 0 1 1			
14	1 0 0 1 0			
15	1 0 0 0 1			
16	1 0 0 0 0			

Teilmengen **ohne** Zelt:

Gymnasium Ottweiler

Nr.	Z S K P E	Gewicht (kg)	Wert (€)	zulässig?
17	0 1 1 1 1			
18	0 1 1 1 0			
19	0 1 1 0 1			
20	0 1 1 0 0			
21	0 1 0 1 1			
22	0 1 0 1 0			
23	0 1 0 0 1			
24	0 1 0 0 0			
25	0 0 1 1 1			
26	0 0 1 1 0			
27	0 0 1 0 1			
28	0 0 1 0 0			
29	0 0 0 1 1			
30	0 0 0 1 0			
31	0 0 0 0 1			
32	0 0 0 0 0			

Optimale Auswahl, Gesamtgewicht und Gesamtwert:

Gymnasium Ottweiler

Arbeitsauftrag 40: Anzahl der Teilmengen

1. Wie viele Teilmengen musstest du im Auftrag *Brute-Force von Hand* insgesamt prüfen? Berechne die Anzahl zuerst mit der Formel 2^n. Überprüfe sie dann durch Abzählen in deiner Tabelle.
2. Wie viele davon waren zulässig?
3. Wie viele Teilmengen müsstest du prüfen, wenn noch eine Trinkflasche als sechster Gegenstand dazukäme?

Deine Antworten:

Mit Brute-Force weißt du jetzt sicher: Die optimale Lösung ist **Zelt, Schlafsack und Erste-Hilfe-Set** mit 10 kg und 120 €. Keine andere zulässige Auswahl ist so viel wert.

Für Interessierte: Brute-Force mit Python

Dieser Unterabschnitt ist freiwillig. Die 32 Teilmengen erzeugst du im Programm genauso wie die Zeilen einer Wahrheitstabelle: mit fünf ineinander geschachtelten Schleifen über `[0, 1]`.

```
namen = ["Zelt", "Schlafsack", "Kocher", "Proviant", "Erste-Hilfe-Set"]
gewichte = [5, 4, 2, 3, 1]
werte = [60, 40, 30, 25, 20]
tragkraft = 10

def gesamt(auswahl, zahlen):
    summe = 0
    for i in range(len(auswahl)):
        if auswahl[i] == 1:
            summe += zahlen[i]
    return summe

anzahl = 0
bester_wert = 0
beste_auswahl = []
for z in [0, 1]:
    for s in [0, 1]:
        for k in [0, 1]:
            for p in [0, 1]:
                for e in [0, 1]:
                    auswahl = [z, s, k, p, e]
                    anzahl += 1
                    gewicht = gesamt(auswahl, gewichte)
                    wert = gesamt(auswahl, werte)
                    if gewicht <= tragkraft and wert > bester_wert:
                        bester_wert = wert
                        beste_auswahl = auswahl

print("Geprüfte Teilmengen:", anzahl)
print("Bester Wert:", bester_wert, "Euro")
print("Gewicht:", gesamt(beste_auswahl, gewichte), "kg")
for i in range(len(namen)):
```

```
33      if beste_auswahl[i] == 1:
34          print("-", namen[i])
```

AUSGABE

```
Geprüfte Teilmengen: 32
Bester Wert: 120 Euro
Gewicht: 10 kg
- Zelt
- Schlafsack
- Erste-Hilfe-Set
```

- **Zeilen 1–4:** Die Probleminstanz als drei Listen. An derselben Stelle (demselben Index) stehen Name, Gewicht und Wert desselben Gegenstands.
- **Zeilen 6–11:** Die Funktion `gesamt` bekommt eine Auswahl wie `[1, 1, 0, 0, 1]` und eine Liste von Zahlen. Sie addiert die Zahlen an den Stellen, an denen in der Auswahl eine 1 steht. Mit `gewichte` liefert sie das Gesamtgewicht, mit `werte` den Gesamtwert.
- **Zeilen 13–15:** Ein Zähler und die bisher beste Lösung. Zu Beginn ist der beste Wert 0 (leerer Rucksack).
- **Zeilen 16–21:** Fünf Schleifen über `[0, 1]` erzeugen alle $2^5 = 32$ Auswahlen, für jeden Gegenstand „ohne“ (0) oder „mit“ (1).
- **Zeilen 22–24:** Der Zähler wird erhöht, Gesamtgewicht und Gesamtwert werden berechnet.
- **Zeilen 25–27:** Ist die Auswahl zulässig **und** besser als die bisher beste, wird sie gemerkt.
- **Zeilen 29–34:** Ausgabe der Anzahl und der optimalen Auswahl.

3.4.3 Greedy: Was ist pro Kilogramm am meisten wert?

Brute-Force ist bei fünf Gegenständen noch machbar. Bei 30 Gegenständen wären es aber 2^{30}, also über eine Milliarde Teilmengen. Gibt es eine schnelle Faustregel?

Eine naheliegende Idee: Pack zuerst den wertvollsten Gegenstand ein. Das ist aber nicht schlau, wenn er sehr schwer ist. Besser ist die Frage: **Wie viel Wert bekomme ich pro Kilogramm?** Das kennst du vom Einkaufen. Auf jedem Preisschild im Supermarkt steht neben dem Preis auch der Grundpreis, zum Beispiel „1 kg = 4,20 €“. Damit kann man Packungen verschiedener Größe vergleichen.

Definition: Wert-Gewicht-Verhältnis

Das **Wert-Gewicht-Verhältnis** eines Gegenstands ist sein Wert geteilt durch sein Gewicht. Es gibt an, wie viel Wert ein Kilogramm dieses Gegenstands bringt.

Für das Zelt ist es zum Beispiel $60 : 5 = 12$, also 12 € pro kg. Je größer das Verhältnis, desto „lohnender“ ist der Gegenstand pro Kilogramm.

Die Greedy-Heuristik für das Rucksackproblem geht so vor:

1. Berechne für jeden Gegenstand das Wert-Gewicht-Verhältnis.
2. Sortiere die Gegenstände nach diesem Verhältnis, den größten Wert zuerst. Haben zwei Gegenstände dasselbe Verhältnis, kommt der zuerst, der in der Liste weiter oben steht.
3. Gehe die sortierte Liste von oben nach unten durch. Passt der Gegenstand noch in den Rucksack, pack ihn ein. Passt er nicht mehr, lass ihn weg und mach mit dem nächsten weiter.

Das ist eine Greedy-Heuristik: Sie nimmt in jedem Schritt den Gegenstand, der im Moment am lohnendsten aussieht. Eine Entscheidung wird nie zurückgenommen.

Merke

Greedy-Heuristik beim Rucksackproblem: Die Gegenstände werden nach ihrem **Wert-Gewicht-Verhältnis** (Wert : Gewicht) sortiert und der Reihe nach eingepackt, solange sie noch hineinpassen – bis kein weiterer Gegenstand mehr hineinpasst.

Beispiel: Greedy beim Tagesausflug

Für einen Tagesausflug darf die Tasche höchstens **4 kg** wiegen. Zur Auswahl stehen:

Gegenstand	Gewicht (kg)	Wert (€)	Verhältnis (€ pro kg)
Fernglas	2	30	$30 : 2 = 15$
Kamera	3	36	$36 : 3 = 12$
Trinkflasche	1	10	$10 : 1 = 10$
Regenjacke	1	8	$8 : 1 = 8$

Die Liste ist schon nach dem Verhältnis sortiert. Jetzt wird gepackt:

Gegenstand	bisheriges Gewicht	passt?	Gewicht danach	Wert danach
Fernglas	0 kg	ja: $0 + 2 = 2 \leq 4$	2 kg	30 €
Kamera	2 kg	nein: $2 + 3 = 5 > 4$	2 kg	30 €
Trinkflasche	2 kg	ja: $2 + 1 = 3 \leq 4$	3 kg	40 €
Regenjacke	3 kg	ja: $3 + 1 = 4 \leq 4$	4 kg	48 €

Ergebnis: Fernglas, Trinkflasche und Regenjacke, 4 kg, **48 €**.

Ein Vergleich mit Brute-Force (bei vier Gegenständen gibt es nur $2^4 = 16$ Teilmengen) zeigt: 48 € ist hier tatsächlich der größte mögliche Wert. Die Heuristik hat also die optimale Lösung gefunden.

Stolperstein

Hör nicht beim ersten Gegenstand auf, der nicht mehr passt! Im Beispiel passt die Kamera nicht, aber die leichteren Gegenstände danach schon. Wer bei der Kamera aufhört, hat nur 30 € statt 48 € im Rucksack. Gehe die Liste immer **bis zum Ende** durch.

Arbeitsauftrag 41: Greedy-Heuristik von Hand

Wende die Greedy-Heuristik auf die Probleminstanz mit Zelt, Schlafsack, Kocher, Proviant und Erste-Hilfe-Set (Tragkraft 10 kg) an.

1. Berechne für jeden Gegenstand das Wert-Gewicht-Verhältnis und sortiere die Gegenstände absteigend danach.
2. Packe nach der Greedy-Regel ein. Lege dazu eine Tabelle an wie im Beispiel *Greedy beim Tagesausflug*.
3. Vergleiche das Ergebnis mit der optimalen Lösung aus dem Brute-Force-Ansatz.

Gegenstand	Gewicht (kg)	Wert (€)	Verhältnis (€ pro kg)
Zelt	5	60	
Schlafsack	4	40	
Kocher	2	30	
Proviant	3	25	
Erste-Hilfe-Set	1	20	

Sortierte Liste, Packtabelle und Vergleich:

Gymnasium Ottweiler

Warum findet Greedy hier nicht das Optimum?

Schau dir die beiden Ergebnisse genau an:

	Greedy-Heuristik	optimale Lösung
Auswahl	Erste-Hilfe-Set, Kocher, Zelt	Zelt, Schlafsack, Erste-Hilfe-Set
Gesamtgewicht	8 kg	10 kg
Gesamtwert	110 €	120 €

Die Greedy-Heuristik nimmt den Kocher, weil er pro Kilogramm sehr lohnend ist. Danach bleiben aber nur noch 2 kg frei, und dafür gibt es keinen passenden Gegenstand mehr. **2 kg Tragkraft bleiben ungenutzt.** Die optimale Lösung lässt den Kocher weg und nutzt die 10 kg mit dem Schlafsack vollständig aus. Die Heuristik betrachtet jeden Gegenstand einzeln. Wie gut die Gegenstände **zusammen** den Rucksack füllen, sieht sie nicht.

Wie groß ist die Abweichung? Beim Rucksack ist ein **größerer** Wert besser. Deshalb rechnet man hier „optimaler Wert minus Wert der Heuristik“:

$$\text{Abweichung} = \frac{\text{optimaler Wert} - \text{Wert der Heuristik}}{\text{optimaler Wert}} \cdot 100\,\% = \frac{120 - 110}{120} \cdot 100\,\% \approx 8{,}3\,\%$$

Die Greedy-Lösung ist also etwa 8,3 % weniger wert als die optimale. Dafür musste die Heuristik nur fünf Verhältnisse berechnen, sie sortieren und fünfmal prüfen, ob etwas passt – statt 32 Teilmengen durchzurechnen.

So kannst du es sagen

So beschreibst du das Ergebnis in Fachsprache:

- *„Greedy war schlechter.“ → „Die Greedy-Heuristik liefert eine zulässige Lösung mit dem Gesamtwert 110 €. Die optimale Lösung hat den Gesamtwert 120 €.“*
- *„Da war noch Platz.“ → „Die Heuristik nutzt die Tragkraft nicht vollständig aus, weil nach der Auswahl der Gegenstände mit dem besten Wert-Gewicht-Verhältnis kein weiterer Gegenstand mehr passt.“*

Gymnasium Ottweiler

3.4.4 Grenzen der Heuristik

Im Beispiel mit dem Tagesausflug war die Greedy-Lösung optimal, bei der Wanderung nicht. Um zu zeigen, dass eine Heuristik **nicht immer** die optimale Lösung findet, genügt ein einziges Beispiel, bei dem sie daneben liegt. So ein Beispiel heißt **Gegenbeispiel**.

Merke

Um zu zeigen, dass eine Heuristik nicht immer optimal ist, genügt **ein Gegenbeispiel**: eine Probleminstanz, bei der die Heuristik eine schlechtere als die optimale Lösung liefert.

Arbeitsauftrag 42: Ein Gegenbeispiel konstruieren

Denke dir eine eigene Probleminstanz mit **drei Gegenständen** aus, bei der die Greedy-Heuristik **nicht** die optimale Lösung findet. Teste deine Instanz mit beiden Ansätzen: Wende die Greedy-Heuristik an, und bestimme die optimale Lösung mit Brute-Force (bei drei Gegenständen gibt es $2^3 = 8$ Teilmengen).

Tipp: Sorge dafür, dass der Gegenstand mit dem besten Verhältnis so schwer ist, dass danach nichts anderes mehr passt.

Deine Probleminstanz und deine Tests:

Gymnasium Ottweiler

Arbeitsauftrag 43: Brute-Force oder Greedy?

Vergleiche den Brute-Force-Ansatz und die Greedy-Heuristik für das Rucksackproblem. Ergänze die Tabelle und beurteile anschließend: Wann würdest du welchen Ansatz verwenden?

	Brute-Force	Greedy-Heuristik
Ist das Ergebnis immer optimal?		
Was muss man rechnen?		
Anzahl Prüfungen bei 5 Gegenständen		
Anzahl Prüfungen bei 30 Gegenständen		

Deine Beurteilung:

3.4.5 Für Interessierte: die Greedy-Heuristik mit Python

Dieser Unterabschnitt ist freiwillig. Statt die Liste zu sortieren, sucht das Programm in jeder Runde den Gegenstand mit dem besten Verhältnis, der noch nicht geprüft wurde. Das ergibt dieselbe Reihenfolge wie das Sortieren.

```python
namen = ["Zelt", "Schlafsack", "Kocher", "Proviant", "Erste-Hilfe-Set"]
gewichte = [5, 4, 2, 3, 1]
werte = [60, 40, 30, 25, 20]
tragkraft = 10

def verhaeltnis(i):
    return werte[i] / gewichte[i]

geprueft = []
gewicht = 0
wert = 0
for runde in range(len(namen)):
    bester = -1
    for i in range(len(namen)):
        if i not in geprueft:
            if bester == -1 or verhaeltnis(i) > verhaeltnis(bester):
                bester = i
    geprueft.append(bester)
    if gewicht + gewichte[bester] <= tragkraft:
        gewicht += gewichte[bester]
        wert += werte[bester]
        print("eingepackt:", namen[bester])
    else:
        print("passt nicht mehr:", namen[bester])

print("Gesamtgewicht:", gewicht, "kg")
print("Gesamtwert:", wert, "Euro")
```

AUSGABE

```
eingepackt: Erste-Hilfe-Set
eingepackt: Kocher
eingepackt: Zelt
passt nicht mehr: Schlafsack
passt nicht mehr: Proviant
Gesamtgewicht: 8 kg
Gesamtwert: 110 Euro
```

- **Zeilen 6–7:** Die Funktion liefert das Wert-Gewicht-Verhältnis des Gegenstands mit dem Index `i`.
- **Zeilen 9–11:** Die Liste `geprueft` merkt sich, welche Gegenstände schon an der Reihe waren. Gewicht und Wert des Rucksacks beginnen bei 0.
- **Zeile 12:** Es gibt so viele Runden wie Gegenstände.
- **Zeilen 13–17:** In jeder Runde wird unter den noch nicht geprüften Gegenständen der mit dem größten Verhältnis gesucht. `bester == -1` bedeutet: Es wurde in dieser Runde noch keiner gefunden.
- **Zeile 18:** Der gefundene Gegenstand gilt jetzt als geprüft.
- **Zeilen 19–24:** Passt er noch, wird er eingepackt. Sonst wird er weggelassen, und die nächste Runde beginnt (**nicht** aufhören!).
- **Zeilen 26–27:** Das Ergebnis stimmt mit deiner Handrechnung überein.

Versuch: Teste dein Gegenbeispiel

Ändere die Listen im Programm so, dass sie dein Gegenbeispiel aus dem Auftrag *Ein Gegenbeispiel konstruieren* enthalten, und ändere die Tragkraft. Liefert das Programm denselben Wert wie deine Handrechnung? Lass dann auch das Brute-Force-Programm mit deiner Instanz laufen. Dort musst du die Anzahl der Schleifen an die Anzahl der Gegenstände anpassen.

3.4.6 Weitere Übungen

Hier kannst du das Gelernte weiter üben. Die Punkte im Titel zeigen, wie schwer eine Aufgabe ist: ●○○ leicht, ●●○ mittel, ●●● knifflig. Fang mit den leichten an.

In mehreren Aufträgen geht es um diese Probleminstanz: Ben will auf dem Flohmarkt verkaufen. In seinen Fahrradanhänger passen höchstens **7 kg**. Er schätzt, wie viel er für jede Sache bekommt.

Gegenstand	Gewicht (kg)	Wert (€)
Comics (C)	1	11
Bücherkiste (K)	4	32
Inline-Skates (I)	2	18
Lautsprecherbox (L)	5	50

Arbeitsauftrag 44: Ein Rucksackproblem im Alltag ●○○

Tim will Videos auf einen USB-Stick kopieren, um sie bei einem Filmabend zu zeigen. Auf den Stick passen 16 GB. Jedes Video hat eine Dateigröße (in GB) und eine Zahl von Likes. Tim möchte Videos mit möglichst vielen Likes zusammen mitnehmen. Seine Liste: „Skate-Tricks“ (4 GB, 120 Likes, 12 Minuten), „Klassenfahrt“ (7 GB, 300 Likes, 25 Minuten), „Katzenvideo“ (2 GB, 90 Likes, 3 Minuten), „Konzert“ (9 GB, 350 Likes, 40 Minuten).

1. Erkläre, was in Tims Situation den **Gegenständen**, dem **Gewicht**, dem **Wert** und der **Tragkraft** entspricht.
2. Stelle die Probleminstanz abstrahiert als Liste dar. Welche Angabe lässt du weg?
3. Handelt es sich um ein Problem oder eine Probleminstanz? Begründe.

Deine Antworten:

Arbeitsauftrag 45: Was passt in den Anhänger? ●○○

Prüfe für Bens Flohmarkt-Instanz jede Auswahl: Berechne das Gesamtgewicht und entscheide, ob sie zulässig ist. Berechne für die zulässigen Auswahlen auch den Gesamtwert.

Auswahl	Gesamtgewicht (kg)	zulässig?	Gesamtwert (€)
Comics, Bücherkiste, Inline-Skates			
Bücherkiste, Lautsprecherbox			
Inline-Skates, Lautsprecherbox			
Comics, Inline-Skates, Lautsprecherbox			
Comics, Lautsprecherbox			

Arbeitsauftrag 46: Wert pro Kilogramm ●○○

1. Berechne für jeden Gegenstand aus Bens Flohmarkt-Instanz das Wert-Gewicht-Verhältnis.
2. Sortiere die Gegenstände absteigend nach diesem Verhältnis.

Gegenstand	Gewicht (kg)	Wert (€)	Verhältnis (€ pro kg)
Comics	1	11	
Bücherkiste	4	32	
Inline-Skates	2	18	
Lautsprecherbox	5	50	

Sortierte Reihenfolge:

Arbeitsauftrag 47: Teilmengen zählen ●○○

1. Lea darf für den Schulausflug drei Dinge einpacken oder weglassen: Apfel (A), Buch (B) und Cap (C). Schreibe alle möglichen Auswahlen auf. Vergiss die leere Auswahl nicht.
2. Ergänze die Tabelle.
3. Erkläre, warum sich die Anzahl verdoppelt, wenn ein Gegenstand dazukommt.

Gegenstände n	**1**	**2**	**3**	**4**	**6**	**8**
Teilmengen 2^n						

Deine Auswahlen und deine Erklärung:

Arbeitsauftrag 48: Brute-Force am Flohmarkt ●●○

Bestimme mit dem Brute-Force-Ansatz die optimale Auswahl für Bens Anhänger (Tragkraft 7 kg). Die Buchstaben stehen für Comics (C), Bücherkiste (K), Inline-Skates (I) und Lautsprecherbox (L).

1. Wie viele Teilmengen musst du prüfen?
2. Fülle die Tabelle aus. Berechne zuerst das Gewicht. Ist eine Teilmenge zu schwer, trage ✗ ein und lass den Wert weg.
3. Markiere die optimale Lösung.

Nr.	Auswahl	Gewicht (kg)	zulässig?	Wert (€)
1	keine			
2	L			
3	I			
4	I, L			
5	K			
6	K, L			
7	K, I			
8	K, I, L			
9	C			
10	C, L			
11	C, I			
12	C, I, L			
13	C, K			
14	C, K, L			
15	C, K, I			
16	C, K, I, L			

Anzahl der Teilmengen und optimale Lösung:

Gymnasium Ottweiler

Arbeitsauftrag 49: Greedy am Flohmarkt ●●○

1. Wende die Greedy-Heuristik auf Bens Flohmarkt-Instanz an. Die sortierte Reihenfolge kennst du aus dem Auftrag *Wert pro Kilogramm*. Lege eine Packtabelle an wie im Beispiel *Greedy beim Tagesausflug*.
2. Vergleiche mit der optimalen Lösung aus dem Auftrag *Brute-Force am Flohmarkt*.
3. Erkläre, warum die Heuristik hier nicht die optimale Lösung findet.

Gegenstand	bisheriges Gewicht	passt?	Gewicht danach	Wert danach

Dein Vergleich und deine Erklärung:

Arbeitsauftrag 50: Fehler beim Packen ●●○

Für den Schulbasar transportiert Elif Gebäck in einem Bollerwagen, der höchstens **9 kg** trägt:

Gebäck	Gewicht (kg)	Wert (€)
Waffeln	2	16
Muffins	4	28
Torte	5	30
Kekse	2	10
Limonade	1	4

Elif wendet die Greedy-Heuristik so an:

- Verhältnisse: Waffeln $16 : 2 = 8$, Muffins $28 : 4 = 7$, Torte $30 : 5 = 6$, Kekse $2 : 10 = 0{,}2$, Limonade $4 : 1 = 4$.
- Sortiert: Waffeln, Muffins, Torte, Limonade, Kekse.
- Waffeln passen (2 kg), Muffins passen (6 kg). Die Torte passt nicht mehr ($6 + 5 = 11 > 9$). Fertig.
- Ergebnis: Waffeln und Muffins, 6 kg, 44 €.

1. Finde Elifs **zwei** Fehler und erkläre sie.
2. Wende die Heuristik richtig an. Gib Auswahl, Gesamtgewicht und Gesamtwert an.

Elifs Fehler und deine richtige Lösung:

Arbeitsauftrag 51: Gegenbeispiel oder nicht? ●●●

Drei Personen behaupten, ein Gegenbeispiel für die Greedy-Heuristik gefunden zu haben. Prüfe jede Behauptung: Wende die Greedy-Heuristik an und bestimme mit Brute-Force die optimale Lösung (8 Teilmengen). Entscheide dann, ob wirklich ein Gegenbeispiel vorliegt.

Person	Tragkraft	Gegenstand X	Gegenstand Y	Gegenstand Z
Emma	5 kg	3 kg, 15 €	2 kg, 8 €	2 kg, 7 €
Noah	6 kg	4 kg, 20 €	3 kg, 13 €	3 kg, 12 €
Lea	4 kg	1 kg, 6 €	4 kg, 20 €	3 kg, 12 €

Deine Prüfung:

Gymnasium Ottweiler

3.4.7 Das Wichtigste in Kürze

- Beim **Rucksackproblem** sucht man eine Auswahl von Gegenständen mit **maximalem Gesamtwert**, deren Gesamtgewicht die Tragkraft nicht übersteigt.
- Eine Auswahl ist eine **Teilmenge**. Bei n Gegenständen gibt es 2^n Teilmengen, denn jeder Gegenstand kommt mit oder nicht. Jeder weitere Gegenstand verdoppelt die Anzahl (**exponentielles Wachstum**).
- **Brute-Force** prüft alle 2^n Teilmengen und findet garantiert die optimale Lösung. In der Wanderungs-Instanz: Zelt, Schlafsack und Erste-Hilfe-Set mit 120 €.
- Das **Wert-Gewicht-Verhältnis** ist Wert : Gewicht.
- Die **Greedy-Heuristik** sortiert nach dem Wert-Gewicht-Verhältnis und packt der Reihe nach ein, was noch passt. Sie ist schnell, aber nicht immer optimal (hier 110 € statt 120 €).
- Ein **Gegenbeispiel** genügt, um zu zeigen, dass eine Heuristik nicht immer die optimale Lösung findet.

Selbstcheck

- ☐ Ich kann das Rucksackproblem beschreiben und zulässige Lösungen angeben.
- ☐ Ich kann erklären, warum es bei n Gegenständen 2^n Teilmengen gibt.
- ☐ Ich kann alle Teilmengen einer kleinen Instanz systematisch aufschreiben und die optimale Lösung bestimmen.
- ☐ Ich kann das Wert-Gewicht-Verhältnis berechnen und die Greedy-Heuristik von Hand anwenden.
- ☐ Ich kann erklären, warum die Greedy-Heuristik bei der Wanderung nicht die optimale Lösung findet.
- ☐ Ich kann ein Gegenbeispiel konstruieren, bei dem die Greedy-Heuristik versagt.
- ☐ Ich kann beurteilen, wann Brute-Force und wann die Greedy-Heuristik sinnvoller ist.

Gymnasium Ottweiler

3.5 Aufwand: Wie lange dauert das Ausprobieren?

Leitidee

Eine alte Legende erzählt vom Erfinder des Schachspiels. Als Belohnung wünschte er sich Reis: ein Korn auf das erste Feld des Schachbretts, zwei auf das zweite, vier auf das dritte – auf jedes Feld doppelt so viele wie auf das vorige. Der König hielt das für einen bescheidenen Wunsch. Doch allein auf dem letzten Feld müssten 2^{63}, also über 9 Trillionen Körner liegen. Das ist mehr Reis, als heute auf der ganzen Welt in hundert Jahren geerntet wird. Genauso überraschend wächst die Anzahl der Möglichkeiten, die Brute-Force ausprobieren muss. In diesem Abschnitt rechnest du nach, wie schnell sie wächst, und vergleichst den **Aufwand** von Brute-Force und Heuristik.

Lernziele

Nach diesem Abschnitt kannst du

- erklären, was man anschaulich unter dem Aufwand eines Lösungsansatzes versteht,
- die Anzahl der Möglichkeiten für Brute-Force beim Handlungsreisenden ($(n-1)!$) und beim Rucksackproblem (2^n) berechnen,
- abschätzen, wie lange ein Computer für Brute-Force bei größeren Probleminstanzen brauchen würde,
- faktorielles und exponentielles Wachstum beschreiben und vergleichen,
- den Aufwand von Brute-Force und Greedy-Heuristiken miteinander vergleichen,
- begründen, warum man in der Praxis oft Heuristiken verwendet.

3.5.1 Was ist Aufwand?

Wenn du ein Zimmer aufräumst, kannst du den Aufwand auf zwei Arten angeben: „Ich musste 40 Dinge wegräumen“ oder „Ich habe 25 Minuten gebraucht“. Die erste Angabe hängt nur vom Zimmer ab. Die zweite hängt auch davon ab, wie schnell du bist.

In der Informatik ist es ähnlich. Den **Aufwand** eines Lösungsansatzes beschreibt man anschaulich so:

- durch die **Anzahl der Möglichkeiten**, die geprüft werden müssen (zum Beispiel Rundreisen oder Teilmengen), oder
- durch die **gemessene Rechenzeit** auf einem Computer.

Die Rechenzeit hängt vom Computer ab: Ein neuer Rechner ist schneller als ein alter. Die Anzahl der Möglichkeiten hängt dagegen nur von der Probleminstanz und vom Lösungsansatz ab. Deshalb rechnen wir in diesem Abschnitt vor allem mit ihr.

Definition: Aufwand

Der **Aufwand** eines Lösungsansatzes gibt an, wie viel Arbeit nötig ist, um eine Probleminstanz zu lösen. Anschaulich misst man ihn als Anzahl der zu prüfenden Möglichkeiten oder als Rechenzeit.

Merke

Der **Aufwand** eines Lösungsansatzes lässt sich anschaulich als **Anzahl der zu prüfenden Möglichkeiten** oder als **tatsächlich gemessene Rechenzeit** beschreiben.

3.5.2 Brute-Force beim Handlungsreisenden: faktorielles Wachstum

Ein Programm hat alle Rundreisen für 2 bis 8 Orte erzeugt und gezählt (Startort fest, ohne Berücksichtigung der Symmetrie). Das ist das Ergebnis:

Orte n	Anzahl der Rundreisen
2	1
3	2
4	6
5	24
6	120
7	720
8	5 040

Das sind genau die Werte von $(n-1)!$, die du im Abschnitt über Brute-Force hergeleitet hast. Bei 8 Orten ist es zum Beispiel $7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 5040$.

Merke

Beim Problem des Handlungsreisenden (Brute-Force) wächst der Aufwand **faktoriell** mit der Anzahl der Städte: $(n-1)!$

Wie lange braucht ein Computer dafür? Nehmen wir einen sehr schnellen Computer an, der **eine Milliarde** ($1\,000\,000\,000$) Rundreisen pro Sekunde prüfen kann. Das ist sehr großzügig gerechnet. Dann gilt:

$$\text{Zeit in Sekunden} = \frac{\text{Anzahl der Rundreisen}}{1\,000\,000\,000}$$

Bei 8 Orten sind das $5040 \,:\, 1\,000\,000\,000 \;=\; 0{,}000\,005\,04$ Sekunden – nur ein paar Millionstel Sekunden. Kein Grund zur Sorge also? Rechne selbst weiter.

Arbeitsauftrag 52: Tabelle interpretieren

Betrachte die Tabelle mit den Anzahlen der Rundreisen für 2 bis 8 Orte.

1. Um welchen Faktor wächst die Anzahl der Rundreisen jeweils von n zu $n+1$ Orten? Beschreibe das Muster.
2. Schätze anhand des Musters (ohne die Fakultät direkt auszurechnen), wie viele Rundreisen es bei **12 Orten** gibt. Wie lange würde der Computer mit einer Milliarde Rundreisen pro Sekunde dafür brauchen?
3. Und bei **20 Orten**? Berechne die Anzahl mit dem Taschenrechner (Taste $x!$) und die Zeit in Jahren. Ein Jahr hat $365 \cdot 24 \cdot 60 \cdot 60 = 31\,536\,000$ Sekunden.

Deine Antworten:

Die folgende Tabelle setzt die Rechnung fort. Die Zeiten gelten für den Computer mit einer Milliarde Rundreisen pro Sekunde.

Orte n	Rundreisen $(n-1)!$	Rechenzeit
10	362 880	weniger als eine Tausendstelsekunde
12	39 916 800	etwa 0,04 Sekunden
15	87 178 291 200	etwa 87 Sekunden (knapp 1,5 Minuten)
18	355 687 428 096 000	etwa 4,1 Tage
20	121 645 100 408 832 000	etwa 3,9 Jahre
25	620 448 401 733 239 439 360 000	etwa 20 Millionen Jahre

Und die Symmetrie? Sie halbiert die Anzahl der Rundreisen, die man wirklich prüfen muss. Bei 25 Orten wären es dann „nur“ noch etwa 10 Millionen Jahre. Das ändert am Ergebnis nichts: Brute-Force ist für 25 Orte hoffnungslos – dabei hat ein Paketdienst oft über 100 Adressen am Tag.

Stolperstein

Ein doppelt so schneller Computer hilft kaum. Er schafft in derselben Zeit doppelt so viele Rundreisen. Aber schon **ein** zusätzlicher Ort vervielfacht die Anzahl: Von 20 auf 21 Orte wächst sie auf das Zwanzigfache. Schnellere Computer verschieben die Grenze nur um wenige Orte.

3.5.3 Brute-Force beim Rucksack: exponentielles Wachstum

Beim Rucksackproblem verdoppelt jeder zusätzliche Gegenstand die Anzahl der Teilmengen. Bei n Gegenständen sind es 2^n. Das ist das Muster aus der Schachbrett-Legende.

Merke

Beim Rucksackproblem (Brute-Force) wächst der Aufwand **exponentiell** mit der Anzahl der Gegenstände: 2^n

Arbeitsauftrag 53: Brute-Force und Greedy beim Rucksack

1. Berechne für eine wachsende Anzahl von Gegenständen die Anzahl der Teilmengen, die Brute-Force prüfen muss, und trage sie in die Tabelle ein.
2. Vergleiche gedanklich mit dem Aufwand der Greedy-Heuristik: Wie viele Gegenstände muss sie jeweils prüfen, nachdem die Liste sortiert ist?

Gegenstände n	Teilmengen 2^n (Brute-Force)	Prüfungen der Greedy-Heuristik
5		
10		
15		
18		

Gymnasium Ottweiler

Dein Vergleich:

Auch hier lohnt sich ein Blick auf größere Instanzen. Wieder gilt: eine Milliarde Teilmengen pro Sekunde.

Gegenstände n	**Teilmengen 2^n**	**Rechenzeit**
20	1 048 576	etwa eine Tausendstelsekunde
30	1 073 741 824	etwa 1 Sekunde
40	1 099 511 627 776	etwa 18 Minuten
50	1 125 899 906 842 624	etwa 13 Tage
60	1 152 921 504 606 846 976	etwa 37 Jahre

Merkst du das Muster? Zehn Gegenstände mehr bedeuten jeweils etwa das Tausendfache, denn $2^{10} = 1024$.

Faktoriell oder exponentiell – was wächst schneller?

Vergleiche die beiden Wachstumsarten für kleine Werte von n:

n	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$(n-1)!$	1	1	2	6	24	120	720	5 040	40 320	362 880
2^n		2	4	8	16	32	64	128	256	512	1 024

Bei $n = 1$ steht in der Tabelle $(1-1)! = 0! = 1$ (zur Erinnerung: $0! = 1$ ist so festgelegt). Bis $n = 5$ ist 2^n größer. Ab $n = 6$ überholt $(n-1)!$ und lässt 2^n weit hinter sich. Der Grund: Bei 2^n wird bei jedem Schritt mit 2 multipliziert, bei der Fakultät mit einer immer größeren Zahl. Beide Wachstumsarten sind aber so schnell, dass Brute-Force schon bei mittelgroßen Instanzen unbrauchbar wird.

3.5.4 Und die Heuristiken?

Wie viel Arbeit macht die Nearest-Neighbor-Heuristik? Bei n Orten muss sie im ersten Schritt $n-1$ Entfernungen ansehen, im zweiten $n-2$ und so weiter bis 1. Bei 20 Orten sind das

$$19 + 18 + 17 + \ldots + 2 + 1 = 190$$

Entfernungen. Brute-Force bräuchte dagegen $19!$, also etwa $1{,}2 \cdot 10^{17}$ Rundreisen.

Bei der Greedy-Heuristik für den Rucksack mit 20 Gegenständen: 20 Verhältnisse berechnen, die Liste sortieren und 20-mal prüfen, ob ein Gegenstand passt. Auch mit einem einfachen Sortierverfahren sind das nur wenige hundert Schritte statt über einer Million Teilmengen.

bei $n = 20$	Brute-Force	Greedy-Heuristik
Handlungsreisender	$19! \approx 1{,}2 \cdot 10^{17}$ Rundreisen	190 Entfernungen (Nearest Neighbor)
Rucksack	$2^{20} = 1\,048\,576$ Teilmengen	wenige hundert Schritte (Wert-Gewicht-Verhältnis)

Merke

- Greedy-Heuristiken haben dagegen einen **deutlich geringeren** Aufwand. Er wächst nur langsam mit der Größe der Probleminstanz. Beim Rucksack bestimmt vor allem das **Sortieren** den Aufwand.
- Schon bei mäßig größeren Probleminstanzen wird der Unterschied zwischen Brute-Force und Heuristik **drastisch** spürbar.

Deshalb setzt man in der Praxis bei großen Probleminstanzen fast immer Heuristiken ein. Man nimmt in Kauf, dass die Lösung vielleicht nicht optimal ist – dafür bekommt man überhaupt eine Lösung, und zwar rechtzeitig.

So kannst du es sagen

So vergleichst du den Aufwand in Fachsprache:

- *„Brute-Force dauert ewig.“* → *„Der Aufwand des Brute-Force-Ansatzes wächst beim Handlungsreisenden faktoriell mit der Anzahl der Orte.“*
- *„Bei einem Gegenstand mehr wird es doppelt so viel.“* → *„Beim Rucksackproblem verdoppelt jeder zusätzliche Gegenstand die Anzahl der zu prüfenden Teilmengen; der Aufwand wächst exponentiell.“*
- *„Greedy geht schneller.“* → *„Die Greedy-Heuristik hat einen deutlich geringeren Aufwand, liefert aber nicht immer eine optimale Lösung.“*

3.5.5 Wachstum im Diagramm

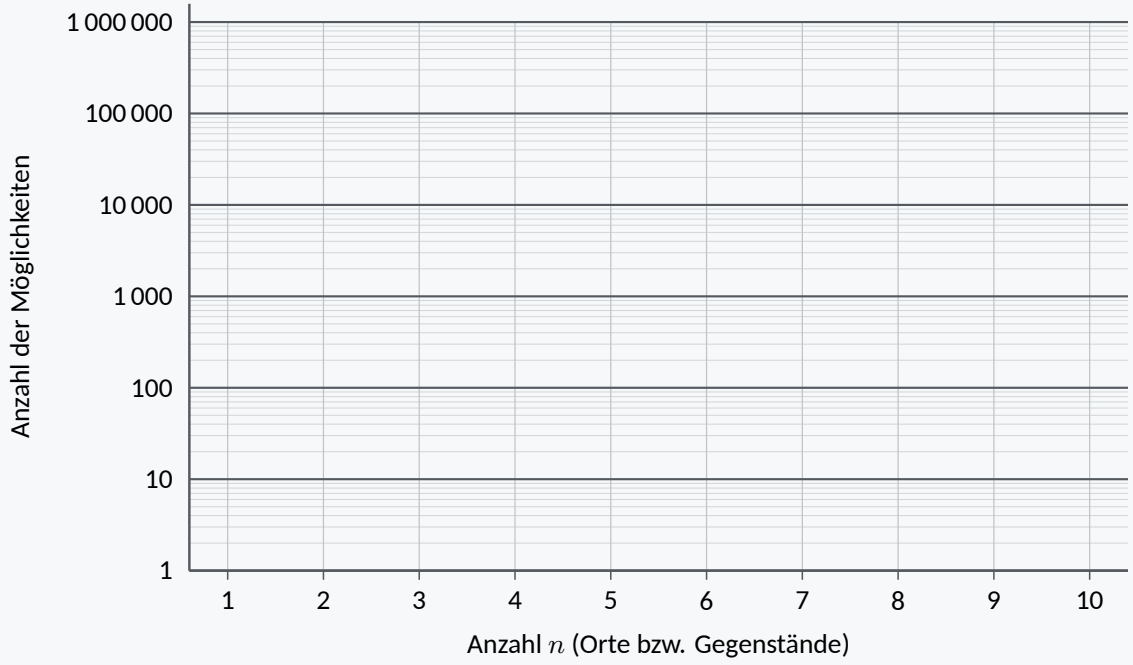
Die Zahlen werden so schnell so groß, dass sie in ein normales Diagramm nicht hineinpassen. Wenn die Hochachse bis 400 000 reicht, verschwinden alle Werte bis 1000 als Punkte auf der Rechtsachse. Man benutzt deshalb eine besondere Einteilung.

Werkzeug

Bei einer **logarithmischen Skala** bedeutet jeder Schritt auf der Hochachse **das Zehnfache**: Auf 1 folgt 10, dann 100, 1000 und so weiter. Die dünnen Linien dazwischen stehen für 2, 3, …, 9 (bzw. 20, 30, …, 90 usw.). Sie liegen nicht gleichmäßig, sondern rücken nach oben immer enger zusammen. Ein Wert wie 24 liegt also zwischen der Linie für 20 und der für 30, etwas näher an 20. So passen sehr kleine und sehr große Zahlen in ein Diagramm.

Arbeitsauftrag 54 (Zusatz): Diagramm zum Wachstum

Stelle das faktorielle Wachstum (Handlungsreisender, $(n-1)!$) und das exponentielle Wachstum (Rucksack, 2^n) für $n = 1$ bis 10 in dem Diagramm mit logarithmischer Skala gegenüber. Nimm die Werte aus der Tabelle im Abschnitt *Faktoriell oder exponentiell*. Zeichne die Punkte ein, verbinde sie und kennzeichne die beiden Kurven. Beschreibe, was du erkennst.



Leeres Diagramm mit logarithmischer Hochachse

Das erkenne ich:

3.5.6 Für Interessierte: Die Tabelle mit Python berechnen

Dieser Unterabschnitt ist freiwillig. Python kann mit beliebig großen ganzen Zahlen rechnen. Die Fakultät berechnest du, indem du bei jedem Schritt mit der nächsten Zahl multiplizierst – genau wie im Auftrag *Tabelle interpretieren*.

```
1 rundreisen = 1
2 for n in range(2, 13):
3     if n > 2:
4         rundreisen = rundreisen * (n - 1)
5     print(n, rundreisen, 2 ** n)
```

AUSGABE

```
2 1 4
3 2 8
4 6 16
5 24 32
6 120 64
7 720 128
8 5040 256
9 40320 512
10 362880 1024
11 3628800 2048
12 39916800 4096
```

- **Zeile 1:** Bei $n = 2$ Orten gibt es $(2-1)! = 1$ Rundreise. Damit beginnt die Variable.
- **Zeilen 2–4:** Für jedes weitere n wird die bisherige Anzahl mit $n - 1$ multipliziert. Aus $(n-2)!$ wird so $(n-1)!$.
- **Zeile 5:** Ausgegeben werden n, die Anzahl der Rundreisen und die Anzahl der Teilmengen 2^n (mit `**` für „hoch“).

Versuch: Weiter rechnen

Ändere `range(2, 13)` in `range(2, 26)`. Vergleiche die letzte Zeile mit der Tabelle im Abschnitt über das faktorielle Wachstum. Ergänze dann eine Ausgabe der Rechenzeit in Sekunden: `rundreisen / 1000000000`. Python schreibt sehr kleine Kommazahlen in einer Kurzform: `1e-09` bedeutet $1 \cdot 10^{-9}$, also eine Milliardstel.

3.5.7 Weitere Übungen

Hier kannst du das Gelernte weiter üben. Die Punkte im Titel zeigen, wie schwer eine Aufgabe ist: ●○○ leicht, ●●○ mittel, ●●● knifflig. Fang mit den leichten an.

Arbeitsauftrag 55: Sätze zum Aufwand ●○○

Ergänze die Sätze.

Satzanfang	Ergänzung
Den Aufwand eines Lösungsansatzes kann man angeben als Anzahl der …	
… oder als gemessene …	
Beim Handlungsreisenden prüft Brute-Force bei n Orten so viele Rundreisen: …	
Dieses Wachstum nennt man …	
Beim Rucksackproblem prüft Brute-Force bei n Gegenständen so viele Teilmengen: …	
Dieses Wachstum nennt man …	
Eine Greedy-Heuristik prüft im Vergleich zu Brute-Force …	

Arbeitsauftrag 56: Zweierpotenzen ●○○

1. Ergänze die Tabelle. Tipp: Jede Zahl ist doppelt so groß wie die davor.
2. Wie viele Teilmengen muss Brute-Force bei 12 Gegenständen prüfen, wie viele bei 13?

n	5	6	7	8	9	10	11	12
2^n	32							

Deine Antwort zu 2:

Gymnasium Ottweiler

Arbeitsauftrag 57: Rundreisen oder Teilmengen? ●○○

Entscheide bei jeder Situation, ob sie zum Problem des Handlungsreisenden oder zum Rucksackproblem passt. Gib an, wie viele Möglichkeiten Brute-Force prüfen muss, und ob das Wachstum faktoriell oder exponentiell ist.

Situation	Problem	Anzahl der Möglichkeiten	Wachstum
Ein Bäcker beliefert morgens 6 Cafés und kehrt zur Backstube zurück (7 Orte).			
Für eine Wanderung stehen 7 Gegenstände zur Auswahl, der Rucksack hat eine Tragkraft.			
Auf dem Handy ist Speicher frei. Von 10 Apps soll Lina einige installieren, die zusammen möglichst nützlich sind.			
Ein Schulbus holt Kinder an 9 Haltestellen ab und fährt zur Schule zurück, von der er gestartet ist (10 Orte).			

Arbeitsauftrag 58: Wer muss mehr prüfen? ●○○

Vergleiche den Aufwand von Brute-Force beim Handlungsreisenden mit n Orten und beim Rucksack mit n Gegenständen. Ergänze die Tabelle und gib an, bei welchem Problem Brute-Force mehr prüfen muss.

n	**Rundreisen** $(n-1)!$	**Teilmengen** 2^n	**mehr muss prüfen …**
4			
5			
6			
8			

Gymnasium Ottweiler

Das fällt mir auf:

Arbeitsauftrag 59: Wie lange rechnet der Computer? ●●○

Ein Laptop prüft **eine Million** Rundreisen pro Sekunde.

1. Wie lange braucht er für Brute-Force bei 10, 12 und 14 Orten? Nutze die Werte $9! = 362\,880$, $11! = 39\,916\,800$ und $13! = 6\,227\,020\,800$.
2. Eine Lieferfirma muss ihre Route jeden Morgen in höchstens 5 Minuten berechnen. Für wie viele Orte reicht Brute-Force auf diesem Laptop? Begründe.

Deine Rechnungen:

Arbeitsauftrag 60: Wie viele Gegenstände schafft der Laptop? ●●○

Derselbe Laptop prüft **eine Million** Teilmengen pro Sekunde.

1. Wie lange braucht er bei 20, 30 und 40 Gegenständen? Nutze $2^{20} = 1\,048\,576$, $2^{30} = 1\,073\,741\,824$ und $2^{40} = 1\,099\,511\,627\,776$.
2. Der Laptop darf **eine Stunde** rechnen. Das sind 3 600 000 000 Teilmengen. Wie viele Gegenstände kann er höchstens untersuchen? Tipp: Beginne bei 2^{30} und verdopple.

Deine Rechnungen:

Arbeitsauftrag 61: Ein zehnmal schnellerer Computer ●●●

Der Laptop aus den letzten beiden Aufträgen braucht für Brute-Force beim Handlungsreisenden mit 12 Orten etwa 40 Sekunden und beim Rucksack mit 20 Gegenständen etwa 1 Sekunde. Ein neuer Computer ist **zehnmal so schnell**.

1. Wie viele Orte kann der neue Computer in 40 Sekunden untersuchen? Begründe mit dem Faktor, um den die Anzahl der Rundreisen wächst.
2. Wie viele Gegenstände kann er in 1 Sekunde untersuchen? Begründe ebenso.
3. Was folgt daraus: Lässt sich das Problem des großen Aufwands mit schnelleren Computern lösen?

Gymnasium Ottweiler

Deine Antworten:

3.5.8 Das Wichtigste in Kürze

- Der **Aufwand** eines Lösungsansatzes wird anschaulich als Anzahl der zu prüfenden Möglichkeiten oder als Rechenzeit angegeben.
- Brute-Force beim **Handlungsreisenden** prüft $(n-1)!$ Rundreisen: **faktorielles** Wachstum. Von n auf $n+1$ Orte wächst die Anzahl um den Faktor n.
- Brute-Force beim **Rucksackproblem** prüft 2^n Teilmengen: **exponentielles** Wachstum. Jeder weitere Gegenstand verdoppelt die Anzahl.
- Selbst bei einer Milliarde Möglichkeiten pro Sekunde dauert Brute-Force bei 20 Orten etwa 3,9 Jahre und bei 25 Orten etwa 20 Millionen Jahre.
- **Greedy-Heuristiken** haben einen viel geringeren Aufwand, der nur langsam wächst (z. B. 190 Entfernungen bei Nearest Neighbor mit 20 Orten).
- Deshalb verwendet man bei großen Probleminstanzen Heuristiken – auch wenn ihre Lösung nicht immer optimal ist.

Selbstcheck

- ☐ Ich kann erklären, was man unter dem Aufwand eines Lösungsansatzes versteht.
- ☐ Ich kann die Anzahl der Rundreisen $(n-1)!$ und der Teilmengen 2^n für ein gegebenes n berechnen.
- ☐ Ich kann abschätzen, wie lange ein Computer für Brute-Force braucht.
- ☐ Ich kann faktorielles und exponentielles Wachstum beschreiben und vergleichen.
- ☐ Ich kann den Aufwand von Brute-Force und Greedy-Heuristik an einem Beispiel vergleichen.
- ☐ Ich kann begründen, warum man in der Praxis oft Heuristiken einsetzt.

3.6 Heuristiken bewerten und Rückblick

Leitidee

Du hast gesehen: Die Nearest-Neighbor-Heuristik findet bei den vier Orten manchmal die optimale Rundreise und manchmal nicht. Die Greedy-Heuristik für den Rucksack war beim Tagesausflug optimal, bei der Wanderung nicht. Wie gut ist eine Heuristik denn nun? Mit einem einzigen Test kann man das nicht sagen – so wie man ein Navigationsgerät nicht nach einer einzigen Fahrt beurteilt. In diesem Abschnitt lernst du, wie man Heuristiken an **vielen** Probleminstanzen prüft und das Ergebnis in Zahlen fasst. Zum Schluss blickst du auf das ganze Kapitel zurück und bereitest dich auf einen möglichen Test vor.

Lernziele

Nach diesem Abschnitt kannst du

- die Abweichung einer Heuristik von der optimalen Lösung in Prozent berechnen, für Probleme mit kleinster und mit größter Zielgröße,
- erklären, warum man eine Heuristik an mehreren, möglichst unterschiedlichen Probleminstanzen testen muss,
- die durchschnittliche Abweichung berechnen und deuten,
- beurteilen, ob eine Heuristik für den praktischen Einsatz geeignet ist,
- die wichtigsten Begriffe des Kapitels erklären und an Beispielen anwenden.

3.6.1 Die Abweichung als Maßstab

Um eine Heuristik zu bewerten, vergleicht man ihre Lösung mit der optimalen Lösung. Dafür muss die optimale Lösung bekannt sein. Bei kleinen Probleminstanzen kann man sie mit Brute-Force berechnen. Üblich ist es, den Unterschied in **Prozent** anzugeben. So kann man auch Instanzen mit ganz unterschiedlichen Zahlen vergleichen: 10 km Umweg sind bei einer 50-km-Tour viel, bei einer 500-km-Tour wenig.

Du kennst schon zwei Formeln. Beim Handlungsreisenden ist eine **kleinere** Länge besser:

$$\text{Abweichung} = \frac{\text{Länge der Heuristik} - \text{optimale Länge}}{\text{optimale Länge}} \cdot 100\,\%$$

Beim Rucksackproblem ist ein **größerer** Wert besser:

$$\text{Abweichung} = \frac{\text{optimaler Wert} - \text{Wert der Heuristik}}{\text{optimaler Wert}} \cdot 100\,\%$$

In beiden Fällen gilt: Man teilt durch das Optimum, und das Ergebnis ist nie negativ. 0 % bedeutet, dass die Heuristik die optimale Lösung getroffen hat. Je größer die Abweichung, desto schlechter.

Merke

Um eine Heuristik zu **bewerten**, vergleicht man ihre Lösung mit der **optimalen** Lösung (sofern diese bekannt bzw. berechenbar ist). Üblich ist die **prozentuale Abweichung**.

3.6.2 Eine Heuristik, viele Probleminstanzen

Ein einziger Test sagt wenig. Vielleicht hatte die Heuristik Glück, vielleicht Pech. Das hast du schon bei den vier Orten A, B, C und D erlebt: Mit Start in A war Nearest Neighbor optimal, mit Start in D um 18,75 % zu lang.

Merke

- Eine Heuristik kann bei **manchen** Probleminstanzen die optimale Lösung treffen und bei **anderen** deutlich schlechter abschneiden – deshalb reicht ein einzelner Test nicht aus.
- Zur verlässlichen Bewertung testet man eine Heuristik an **mehreren, möglichst unterschiedlichen** Probleminstanzen und betrachtet z. B. die **durchschnittliche** Abweichung.

Den **Durchschnitt** (das arithmetische Mittel) kennst du aus Mathe: Man addiert alle Werte und teilt durch ihre Anzahl.

Gymnasium Ottweiler

Beispiel: Nearest Neighbor mit allen Startorten

Für die vier Orte A, B, C, D (optimal: 80 km) und die fünf Orte A bis E (optimal: 100 km) wurde Nearest Neighbor von jedem Startort aus durchgeführt:

Startort	A	B	C	D	E	Durchschnitt
vier Orte: Länge	80	80	80	95	–	
vier Orte: Abweichung	0 %	0 %	0 %	18,75 %	–	4,7 %
fünf Orte: Länge	111	100	100	110	100	
fünf Orte: Abweichung	11 %	0 %	0 %	10 %	0 %	4,2 %

Rechnung für vier Orte: $(0+0+0+18{,}75) : 4 \approx 4{,}7$ (in Prozent). Für fünf Orte: $(11+0+0+10+0) : 5 = 4{,}2$ (in Prozent).

Im Durchschnitt liegt die Heuristik also weniger als 5 % über dem Optimum. Im ungünstigsten Fall waren es aber fast 19 %. Beides gehört zu einer ehrlichen Bewertung: der **Durchschnitt** und der **schlechteste Fall**.

3.6.3 Ein Computerexperiment

Mit vier oder fünf Orten kann man nicht viel über eine Heuristik lernen. Für ein aussagekräftigeres Ergebnis hat ein Computer das folgende Experiment durchgeführt:

- Er hat **10 Probleminstanzen** mit je **6 Orten** erzeugt. Die Entfernungen wurden zufällig zwischen 5 und 50 km gewählt (symmetrisch, also Hinweg = Rückweg).
- Für jede Instanz hat er mit **Brute-Force** die optimale Rundreise berechnet (bei 6 Orten sind das $5! = 120$ Rundreisen).
- Außerdem hat er **Nearest Neighbor** angewendet, jeweils mit dem ersten Ort als Start. Bei gleich weit entfernten Orten hat er den im Alphabet früheren genommen.
- Zum Schluss hat er die Abweichung in Prozent und deren Durchschnitt berechnet.

Instanz	optimal (km)	Nearest Neighbor (km)	Abweichung
1	119	119	0,0 %
2	92	92	0,0 %
3	79	79	0,0 %
4	122	124	1,6 %
5	81	81	0,0 %
6	113	113	0,0 %
7	78	109	39,7 %
8	67	75	11,9 %
9	83	87	4,8 %
10	114	135	18,4 %
Durchschnitt			**7,7 %**

Du kannst einzelne Zeilen nachrechnen. Für Instanz 4 zum Beispiel: $\frac{124-122}{122} \cdot 100\,\% \approx 1{,}6\,\%$.

Arbeitsauftrag 62: Auswertung des Experiments

Beantworte anhand der Tabelle:

1. Gab es Instanzen, bei denen die Heuristik die optimale Lösung getroffen hat (Abweichung 0 %)? Wie viele?
2. Wie groß war die größte beobachtete Abweichung, und bei welcher Instanz trat sie auf?
3. Ist die Nearest-Neighbor-Heuristik deiner Einschätzung nach für den praktischen Einsatz geeignet? Begründe.

Deine Antworten:

Gymnasium Ottweiler

Stolperstein

Der Durchschnitt allein kann täuschen. 7,7 % klingt harmlos, aber bei Instanz 7 waren es fast 40 %. Nenne bei einer Bewertung deshalb immer auch den **schlechtesten Fall**. Und: Zehn Instanzen sind schon besser als eine, aber für eine wirklich sichere Aussage testet man viel mehr – und auch größere Instanzen.

3.6.4 Die Ausreißer-Instanz genauer untersucht

Warum hat Nearest Neighbor bei Instanz 7 so schlecht abgeschnitten? Das kannst du selbst herausfinden. Die Orte heißen hier A bis F.

Arbeitsauftrag 63: Instanz 7 von Hand

Hier ist die Entfernungstabelle von Instanz 7 (in km):

	A	B	C	D	E	F
A	–	41	10	36	21	7
B	41	–	5	14	47	42
C	10	5	–	35	28	25
D	36	14	35	–	6	22
E	21	47	28	6	–	36
F	7	42	25	22	36	–

1. Wende die Nearest-Neighbor-Heuristik mit Start in **A** an. Gib die Rundreise und ihre Länge an.
2. Der Computer hat als optimale Rundreise A → C → B → D → E → F → A gefunden. Berechne ihre Länge.
3. Berechne die Abweichung. Erkläre, an welcher Stelle die Heuristik „in die Falle läuft“.
4. Wende Nearest Neighbor noch einmal an, diesmal mit Start in **C**. Was stellst du fest?

Gymnasium Ottweiler

Deine Tabellen und Rechnungen:

3.6.5 Auch die Rucksack-Heuristik auf dem Prüfstand

Dieselbe Bewertungsmethode funktioniert für jede Heuristik. Eine einfache Art, mehrere Probleminstanzen zu bekommen: Man nimmt dieselben Gegenstände, ändert aber die **Tragkraft**. Jede Tragkraft ergibt eine neue Probleminstanz.

Arbeitsauftrag 64: Greedy bei verschiedenen Tragkräften

Nimm die Gegenstände der Wanderung (Zelt 5 kg/60 €, Schlafsack 4 kg/40 €, Kocher 2 kg/30 €, Proviant 3 kg/25 €, Erste-Hilfe-Set 1 kg/20 €). Die sortierte Reihenfolge nach dem Wert-Gewicht-Verhältnis kennst du schon: Erste-Hilfe-Set, Kocher, Zelt, Schlafsack, Proviant.

1. Wende die Greedy-Heuristik für die Tragkräfte 6 kg, 8 kg und 12 kg an. (Für 10 kg kennst du das Ergebnis schon.)
2. Ein Computer hat mit Brute-Force die optimalen Werte berechnet. Trage die Abweichungen ein und berechne die durchschnittliche Abweichung.
3. Beurteile kurz, wie gut die Greedy-Heuristik bei diesen Instanzen funktioniert.

Tragkraft	optimaler Wert	Greedy-Auswahl	Greedy-Wert	Abweichung
6 kg	80 €			
8 kg	110 €			
10 kg	120 €	Erste-Hilfe-Set, Kocher, Zelt	110 €	
12 kg	150 €			

Durchschnitt und Beurteilung:

So kannst du es sagen

So formulierst du eine Bewertung in Fachsprache:

- *„Die Heuristik ist ganz gut.“ → „Die Heuristik weicht im Durchschnitt um 7,7 % von der optimalen Lösung ab.“*
- *„Manchmal geht es schief.“ → „Im schlechtesten untersuchten Fall beträgt die Abweichung 39,7 %.“*
- *„Ich finde sie brauchbar.“ → „Wegen des geringen Aufwands und der meist kleinen Abweichung ist die Heuristik für große Probleminstanzen geeignet.“*

3.6.6 Rückblick: das Kapitel im Überblick

Zum Abschluss lohnt sich ein Blick auf das ganze Kapitel. Eine Übersicht über alle Fachbegriffe mit Beispielen findest du im Rückblick am Ende des Kapitels. Wenn du bei einem Begriff unsicher bist, lies im passenden Abschnitt nach.

Arbeitsauftrag 65: Zusammenfassung des Kapitels

Beantworte in eigenen Worten. Die Fragen sind eine gute Vorbereitung auf einen Test.

1. Was ist der Unterschied zwischen einem Problem und einer Probleminstanz?
2. Wann liefert ein Brute-Force-Ansatz garantiert die optimale Lösung, und wo liegt sein Nachteil?
3. Was zeichnet eine Greedy-Heuristik aus, und welches Risiko birgt sie?
4. Nenne je ein Beispiel für ein Problem mit faktoriellem und eines mit exponentiellem Brute-Force-Aufwand.

Gymnasium Ottweiler

Deine Antworten:

Gymnasium Ottweiler

3.6.7 Ausblick: weitere Probleme und warum Optimieren nachhaltig ist

Das Problem des Handlungsreisenden und das Rucksackproblem sind nur zwei Beispiele. Mit denselben Werkzeugen untersucht man viele weitere Probleme:

- **Kürzeste Wege** (*Shortest Path Problem*): Ein Navigationsgerät sucht den kürzesten Weg von A nach B in einem Straßengraphen. Hier gibt es mit dem **Dijkstra-Algorithmus** ein Verfahren, das schnell **und** garantiert optimal ist. Das ist eine Ausnahme.
- **Behälterproblem** (*Bin Packing*): Gegenstände sollen in möglichst wenige Kisten gleicher Größe gepackt werden, zum Beispiel Pakete in Lieferwagen. Eine einfache Heuristik heißt **First Fit**: Lege jeden Gegenstand in die erste Kiste, in die er noch passt.
- **Aufgabenplanung** (*Scheduling*): In welcher Reihenfolge erledigt man Aufträge, damit alle möglichst früh fertig sind? Eine Heuristik lautet: „Kürzeste Aufgabe zuerst."

Optimieren spart nicht nur Zeit, sondern auch **Ressourcen**. Kürzere Touren von Paketdiensten und Müllfahrzeugen sparen Kraftstoff und vermeiden CO_2-Ausstoß. Gut gepackte Lieferwagen und Container bedeuten weniger Fahrten. Ein guter Plan für Busse oder Maschinen spart Energie. Auch die Rechenzeit selbst verbraucht Strom: Eine Heuristik, die in Sekunden eine gute Lösung findet, kann sinnvoller sein als eine Rechnung, die tagelang einen Großrechner beschäftigt. Optimierungsprobleme sind deshalb ein wichtiges Werkzeug für **nachhaltiges Ressourcenmanagement**.

Arbeitsauftrag 66 (Zusatz): Kisten packen mit First Fit

Kisten tragen höchstens **10 kg**. Fünf Pakete wiegen 6 kg, 5 kg, 4 kg, 3 kg und 2 kg. Lege jedes Paket der Reihe nach in die **erste** Kiste, in die es noch passt. Passt es in keine, öffne eine neue Kiste.

1. Wende First Fit in der Reihenfolge 6, 5, 4, 3, 2 an. Wie viele Kisten brauchst du?
2. Wende First Fit in der Reihenfolge 2, 3, 4, 5, 6 an. Wie viele Kisten brauchst du jetzt?
3. Wie viele Kisten braucht man mindestens? Begründe. Was folgt daraus für die Heuristik?

Gymnasium Ottweiler

Deine Kisten und deine Begründung:

3.6.8 Weitere Übungen

Hier kannst du das Gelernte weiter üben. Die Punkte im Titel zeigen, wie schwer eine Aufgabe ist: ●○○ leicht, ●●○ mittel, ●●● knifflig. Fang mit den leichten an.

Arbeitsauftrag 67: Abweichungen berechnen ●○○

Eine Heuristik für das Problem des Handlungsreisenden wurde an fünf Probleminstanzen getestet. Berechne jeweils die prozentuale Abweichung vom Optimum. Runde auf eine Stelle nach dem Komma.

Instanz	Länge der Heuristik	optimale Länge	Abweichung
1	36 km	30 km	
2	44 km	40 km	
3	25 km	20 km	
4	58 km	58 km	
5	52 km	48 km	

Arbeitsauftrag 68: Durchschnitt und schlechtester Fall ●○○

Eine Heuristik wurde an fünf Instanzen getestet. Die Abweichungen vom Optimum waren: 0 %, 5 %, 12 %, 0 %, 3 %.

1. Berechne die durchschnittliche Abweichung.
2. Nenne den schlechtesten Fall.
3. Bei wie vielen Instanzen war die Heuristik optimal?

Deine Antworten:

Arbeitsauftrag 69: Richtig bewertet? ●○○

Entscheide, ob die Aussage richtig oder falsch ist. Begründe kurz.

Aussage	richtig oder falsch?	Begründung
Um eine Heuristik zu bewerten, vergleicht man ihre Lösung mit der optimalen Lösung.		
Hat eine Heuristik bei einer Instanz 0 % Abweichung, ist sie immer optimal.		
Man sollte eine Heuristik an möglichst unterschiedlichen Instanzen testen.		
Eine Abweichung von 0 % bedeutet, dass die Heuristik die optimale Lösung gefunden hat.		
Die Abweichung berechnet man, indem man durch die Länge der Heuristik teilt.		

Arbeitsauftrag 70: Zwei Navi-Apps im Test ●●○

Zwei Apps berechnen Lieferrouten mit unterschiedlichen Heuristiken. Beide wurden an denselben fünf Probleminstanzen getestet. Die optimalen Längen wurden mit Brute-Force bestimmt.

Instanz	optimal	App X	App Y
1	50 km	55 km	50 km
2	40 km	40 km	40 km
3	80 km	84 km	80 km
4	20 km	21 km	20 km
5	100 km	110 km	130 km

1. Berechne für beide Apps die Abweichung bei jeder Instanz.
2. Berechne für beide Apps die durchschnittliche Abweichung und nenne den schlechtesten Fall.
3. Welche App würdest du einer Lieferfirma empfehlen? Begründe.

Deine Rechnungen und deine Empfehlung:

Arbeitsauftrag 71: Die Rucksack-Heuristik bewerten ●●○

Die Greedy-Heuristik für das Rucksackproblem wurde an vier Instanzen aus diesem Kapitel getestet: an der Wanderung, an Bens Flohmarkt (Auftrag *Greedy am Flohmarkt*) und an zwei Instanzen aus dem Auftrag *Gegenbeispiel oder nicht?*. Beim Rucksack ist ein größerer Wert besser.

Instanz	optimaler Wert	Greedy-Wert	Abweichung
Wanderung	120 €	110 €	
Flohmarkt	68 €	61 €	
Noahs Gegenbeispiel	25 €	20 €	
Leas Gegenbeispiel	20 €	18 €	

1. Berechne die Abweichungen und ihren Durchschnitt. Runde auf eine Stelle nach dem Komma.
2. Mira sagt: „Die Greedy-Heuristik ist im Durchschnitt 12 % schlechter als das Optimum.“ Beurteile diese Aussage. Achte darauf, wie die vier Instanzen ausgewählt wurden.

Deine Rechnungen und deine Beurteilung:

Arbeitsauftrag 72: Ein Test von Hand ●●●

Teste die Nearest-Neighbor-Heuristik selbst an drei Probleminstanzen mit je vier Orten A, B, C, D (Entfernungen in km).

Instanz 1	A	B	C	D
A	–	10	4	6
B	10	–	8	4
C	4	8	–	10
D	6	4	10	–

Instanz 2	A	B	C	D
A	–	4	12	8
B	4	–	9	8
C	12	9	–	9
D	8	8	9	–

Instanz 3	A	B	C	D
A	–	4	2	9
B	4	–	6	7
C	2	6	–	7
D	9	7	7	–

1. Wende bei jeder Instanz Nearest Neighbor mit Start in A an.
2. Bestimme bei jeder Instanz mit Brute-Force die optimale Länge. Wegen der Symmetrie genügen die drei Rundreisen A → B → C → D → A, A → B → D → C → A und A → C → B → D → A.
3. Berechne die Abweichungen, ihren Durchschnitt und nenne den schlechtesten Fall.
4. Beurteile: Reichen drei Instanzen aus, um die Heuristik zu bewerten?

Gymnasium Ottweiler

Deine Rechnungen und deine Beurteilung:

Gymnasium Ottweiler

3.6.9 Das Wichtigste in Kürze

- Eine Heuristik **bewertet** man, indem man ihre Lösung mit der optimalen Lösung vergleicht und die **prozentuale Abweichung** berechnet. Geteilt wird immer durch das Optimum.
- Eine Heuristik kann bei manchen Instanzen optimal sein und bei anderen deutlich schlechter. Ein einzelner Test reicht deshalb nicht.
- Zuverlässig wird eine Bewertung erst mit **vielen, möglichst unterschiedlichen** Instanzen. Man betrachtet die **durchschnittliche** Abweichung und den **schlechtesten Fall**.
- Im Computerexperiment mit 10 zufälligen Instanzen (6 Orte) lag Nearest Neighbor im Durchschnitt 7,7 % über dem Optimum, im schlechtesten Fall 39,7 %.
- Kapitelüberblick: Problem und Probleminstanz → abstrahierte Darstellung → zulässige und optimale Lösung → Brute-Force (garantiert optimal, großer Aufwand) → Heuristik/Greedy (schnell, nicht immer optimal) → Aufwand vergleichen → Heuristik bewerten.
- Optimierung hilft, Ressourcen wie Kraftstoff, Material und Energie zu sparen.

Selbstcheck

- ☐ Ich kann die Abweichung einer Heuristik in Prozent berechnen – beim Handlungsreisenden und beim Rucksack.
- ☐ Ich kann erklären, warum man eine Heuristik an vielen Probleminstanzen testen muss.
- ☐ Ich kann eine durchschnittliche Abweichung berechnen und zusammen mit dem schlechtesten Fall deuten.
- ☐ Ich kann begründet beurteilen, ob eine Heuristik für den praktischen Einsatz geeignet ist.
- ☐ Ich kann die Begriffe Problem, Probleminstanz, zulässige und optimale Lösung, Brute-Force, Heuristik, Greedy und Aufwand erklären.
- ☐ Ich kann je ein Problem mit faktoriellem und mit exponentiellem Brute-Force-Aufwand nennen.

Gymnasium Ottweiler

3.7 Kapitel 3 kompakt: der Lernzettel

Leitidee

Dieser Lernzettel fasst das Kapitel auf wenigen Seiten zusammen. Nutze ihn zum Wiederholen vor einem Test. Ausführliche Erklärungen, Beispiele und Übungen findest du in den einzelnen Abschnitten.

3.7.1 Problem, Probleminstanz, Darstellung (Abschnitt 3.1)

- **Problem:** allgemeine Fragestellung mit Leerstellen. *„Finde die kürzeste Rundreise durch beliebige Orte."*
- **Probleminstanz:** konkreter Fall, alle Werte stehen fest. *„… durch genau diese 4 Orte mit diesen Entfernungen."*
- Test: Könnte ich **sofort losrechnen**? Ja → Instanz. Signalwörter für ein Problem: „beliebig", „begrenzt", „eine".
- **Abstrahierte Darstellung:** nur das, was für das Problem wichtig ist. Formen: **Tabelle** (Entfernungstabelle), **Graph** (Knoten = Orte/Objekte, Kanten = Verbindungen, Kantengewicht = Entfernung), **Liste** (Gegenstände mit Gewicht und Wert).
- **Problem des Handlungsreisenden:** kürzeste **Rundreise**, die jeden Ort genau einmal besucht und zum Start zurückkehrt.
- **Symmetrie:** Ist Hinweg = Rückweg (Tabelle spiegelbildlich zur Diagonale), sind eine Rundreise und ihre Rückwärts-Rundreise gleich lang. Man muss nur die Hälfte untersuchen.

	A	B	C	D
A	–	10	15	20
B	10	–	35	25
C	15	35	–	30
D	20	25	30	–

Lesen: Zeile B, Spalte D → 25 km. Jede Zahl steht zweimal (symmetrisch).

Typische Fehler: die gezeichnete Länge einer Kante statt des Kantengewichts verwenden; Symmetrie annehmen, obwohl Hin- und Rückweg verschieden sind (z. B. Fahrzeiten bergauf und bergab).

3.7.2 Optimale Lösung und Brute-Force (Abschnitt 3.2)

- **Zulässige Lösung:** erfüllt alle Bedingungen (jeder Ort genau einmal, zurück zum Start).
- **Optimale Lösung:** die beste zulässige Lösung bezüglich eines festgelegten **Kriteriums** (z. B. Gesamtlänge). Ohne Kriterium kein „optimal". Es kann mehrere optimale Lösungen geben.
- **Länge einer Rundreise:** alle Teilstrecken addieren, **die Rückfahrt gehört dazu**. A → B → D → C → A: $10 + 25 + 30 + 15 = 80$ km.
- **Brute-Force:** alle zulässigen Lösungen **erzeugen**, **bewerten**, die beste **auswählen**. Vorteil: garantiert optimal. Nachteil: Die Anzahl der Lösungen wächst sehr schnell.
- **Anzahl der Rundreisen** bei n Orten (Start fest): $(n-1)! = (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1$; mit Symmetrie $(n-1)! : 2$ verschiedene.

Typische Fehler: Rückfahrt vergessen; eine Rundreise doppelt zählen, eine andere vergessen; bei Brute-Force zu früh aufhören; $(n-1)!$ ohne Klammer rechnen.

3.7.3 Heuristiken und Greedy (Abschnitt 3.3)

- **Heuristik:** Lösungsstrategie, die **schnell** eine (meist gute, aber **nicht garantiert optimale**) Lösung liefert, ohne alle Möglichkeiten zu prüfen.
- **Greedy-Heuristik** („gierig"): wählt in jedem Schritt, was **im Moment** am besten aussieht; spätere Folgen werden nicht bedacht, Entscheidungen nie zurückgenommen.
- **Nearest Neighbor** (Handlungsreisender): vom aktuellen Ort immer zum **nächstgelegenen, noch nicht besuchten** Ort; am Ende zurück zum Start.

Schritt	aktueller Ort	noch nicht besucht	Wahl
1	A	B 10, C 15, D 20	B (10)
2	B	C 35, D 25	D (25)
3	D	C 30	C (30)
4	C	–	zurück nach A (15)

Ergebnis: 80 km, hier optimal. Mit Start in D: 95 km, nicht optimal. Das Ergebnis kann vom Startort abhängen.

- **Abweichung in Prozent** (kleiner ist besser): $\frac{\text{Heuristik} - \text{Optimum}}{\text{Optimum}} \cdot 100\,\%$, z. B. $\frac{95-80}{80} \cdot 100\,\% \approx 18{,}8\,\%$.

Typische Fehler: einen schon besuchten Ort wählen; „vorausschauen" (das macht Nearest Neighbor nicht); durch die Länge der Heuristik statt durch das Optimum teilen.

3.7.4 Das Rucksackproblem (Abschnitt 3.4)

- Gegenstände mit **Gewicht** und **Wert**, ein Rucksack mit **Tragkraft**. Gesucht: Auswahl mit **maximalem Gesamtwert**, deren Gesamtgewicht die Tragkraft **nicht übersteigt** (genau gleich ist erlaubt).
- **Brute-Force:** alle **Teilmengen** bilden und bewerten. Jeder Gegenstand: mit oder ohne → bei n Gegenständen 2^n Teilmengen (die leere Auswahl zählt mit).
- **Greedy-Heuristik: Wert-Gewicht-Verhältnis** (Wert : Gewicht) berechnen, absteigend sortieren, der Reihe nach einpacken, was noch passt – **bis zum Ende der Liste**.
- Beispiel (Tragkraft 10 kg): Zelt 5 kg/60 €, Schlafsack 4 kg/40 €, Kocher 2 kg/30 €, Proviant 3 kg/25 €, Erste-Hilfe-Set 1 kg/20 €. Greedy: Erste-Hilfe-Set, Kocher, Zelt → 8 kg, 110 €. Optimal (Brute-Force): Zelt, Schlafsack, Erste-Hilfe-Set → 10 kg, 120 €. Greedy lässt 2 kg ungenutzt.
- **Gegenbeispiel:** Eine einzige Instanz, bei der die Heuristik schlechter ist als das Optimum, zeigt: Die Heuristik ist nicht immer optimal.

Typische Fehler: Gewicht : Wert statt Wert : Gewicht; beim ersten Gegenstand, der nicht passt, aufhören; nach dem Wert statt nach dem Verhältnis sortieren; die leere Teilmenge beim Zählen vergessen.

3.7.5 Aufwand (Abschnitt 3.5)

- **Aufwand** anschaulich: **Anzahl der zu prüfenden Möglichkeiten** oder **gemessene Rechenzeit**.
- Brute-Force beim Handlungsreisenden: $(n-1)!$, **faktorielles** Wachstum. Von n auf $n+1$ Orte: Faktor n.
- Brute-Force beim Rucksack: 2^n, **exponentielles** Wachstum. Jeder weitere Gegenstand: Faktor 2.
- Greedy-Heuristiken prüfen **wesentlich weniger** Möglichkeiten (vor allem Sortieren und einmal Durchgehen).
- Schon bei mäßig großen Instanzen ist der Unterschied **drastisch**.

n	**4**	**5**	**6**	**8**	**10**	**12**	**20**
$(n-1)!$	6	24	120	5 040	362 880	39 916 800	$\approx 1{,}2 \cdot 10^{17}$
2^n	16	32	64	256	1 024	4 096	1 048 576

Typische Fehler: faktoriell und exponentiell verwechseln (Rundreisen → Fakultät, Teilmengen → Zweierpotenz); bei Zeitabschätzungen die Einheiten (Sekunden, Minuten, Tage) durcheinanderbringen.

3.7.6 Heuristiken bewerten (Abschnitt 3.6)

- Bewerten = Lösung der Heuristik mit der **optimalen** Lösung vergleichen (wenn diese bekannt bzw. berechenbar ist), üblich als **prozentuale Abweichung**.
- Beim Rucksack ist größer besser: $\frac{\text{Optimum} - \text{Heuristik}}{\text{Optimum}} \cdot 100\,\%$, z. B. $\frac{120-110}{120} \cdot 100\,\% \approx 8{,}3\,\%$.
- Eine Heuristik kann bei manchen Instanzen optimal sein (0 %) und bei anderen deutlich schlechter. **Ein einzelner Test reicht nicht.**
- Verlässlich: an **mehreren, möglichst unterschiedlichen** Instanzen testen, **durchschnittliche** Abweichung berechnen und die **größte** Abweichung nennen.
- Durchschnitt: alle Abweichungen addieren, durch ihre Anzahl teilen. $(0+0+0+18{,}75) : 4 \approx 4{,}7\,\%$.

Typische Fehler: nur den Durchschnitt nennen (ein großer Ausreißer geht unter); nur Instanzen testen, bei denen die Heuristik gut oder nur schlecht abschneidet.

So kannst du es sagen

- *„Es handelt sich um ein Problem/eine Probleminstanz, weil …“*
- *„Der Ansatz prüft/wählt … und benötigt dabei …“*
- *„Die Heuristik ist hier geeignet/nicht geeignet, weil …“*

3.8 Rückblick auf Kapitel 3

Leitidee

Am Ende des Kapitels lohnt sich ein Blick zurück: Welche Fachbegriffe kennst du jetzt? Was kannst du schon sicher, was solltest du noch einmal üben?

3.8.1 Die Basisbegriffe

Diese Begriffe solltest du mit eigenen Worten erklären und an einem Beispiel aus dem Kapitel zeigen können.

Begriff	Bedeutung	Beispiel
Problem	allgemeine Fragestellung mit Leerstellen	kürzeste Rundreise durch beliebige Orte
Probleminstanz	konkreter Fall mit festen Werten	die vier Orte A, B, C, D mit ihrer Entfernungstabelle
abstrahierte Darstellung	nur das Wichtige, als Tabelle, Graph oder Liste	Graph mit Knoten, Kanten, Kantengewichten
zulässige Lösung	erfüllt alle Bedingungen	jede Rundreise; jede Auswahl bis 10 kg
optimale Lösung	beste zulässige Lösung bezüglich eines Kriteriums	A → B → D → C → A (80 km); Zelt, Schlafsack, Erste-Hilfe-Set (120 €)
Brute-Force-Ansatz	alle zulässigen Lösungen erzeugen, bewerten und die beste auswählen	alle $(n-1)!$ Rundreisen; alle 2^n Teilmengen
Heuristik	schnelle Lösungsstrategie, liefert meist eine gute, aber nicht garantiert optimale Lösung	Faustregel „immer zum nächsten Ort“
Greedy-Heuristik	wählt in jedem Schritt das, was im Moment am besten aussieht	Nearest Neighbor; Wert-Gewicht-Verhältnis
Aufwand	Anzahl der Möglichkeiten oder Rechenzeit	faktoriell beim Handlungsreisenden, exponentiell beim Rucksack
Bewertung	Abweichung vom Optimum über viele Instanzen	Durchschnitt 7,7 %, schlechtester Fall 39,7 %

Gymnasium Ottweiler

3.8.2 Kompetenz-Check

Ich kann ...	sicher	teilweise	noch nicht
zwischen einem Problem und einer Probleminstanz unterscheiden.			
kleine Probleminstanzen als Tabelle, Graph oder Liste darstellen und untersuchen.			
strukturelle Eigenschaften eines Problems erkennen (z. B. Symmetrie).			
definieren, was eine optimale Lösung ist.			
einen Brute-Force-Ansatz beschreiben und an kleinen Instanzen durchführen.			
den Begriff Heuristik definieren.			
eine Heuristik beschreiben und auf Probleminstanzen anwenden.			
den Aufwand verschiedener Lösungsansätze vergleichen.			
beurteilen, wie gut eine Heuristik bei verschiedenen Instanzen funktioniert.			

Gymnasium Ottweiler