Fachhochschule Nordwestschweiz
Hochschule für Wirtschaft

Problemlösen als Suche

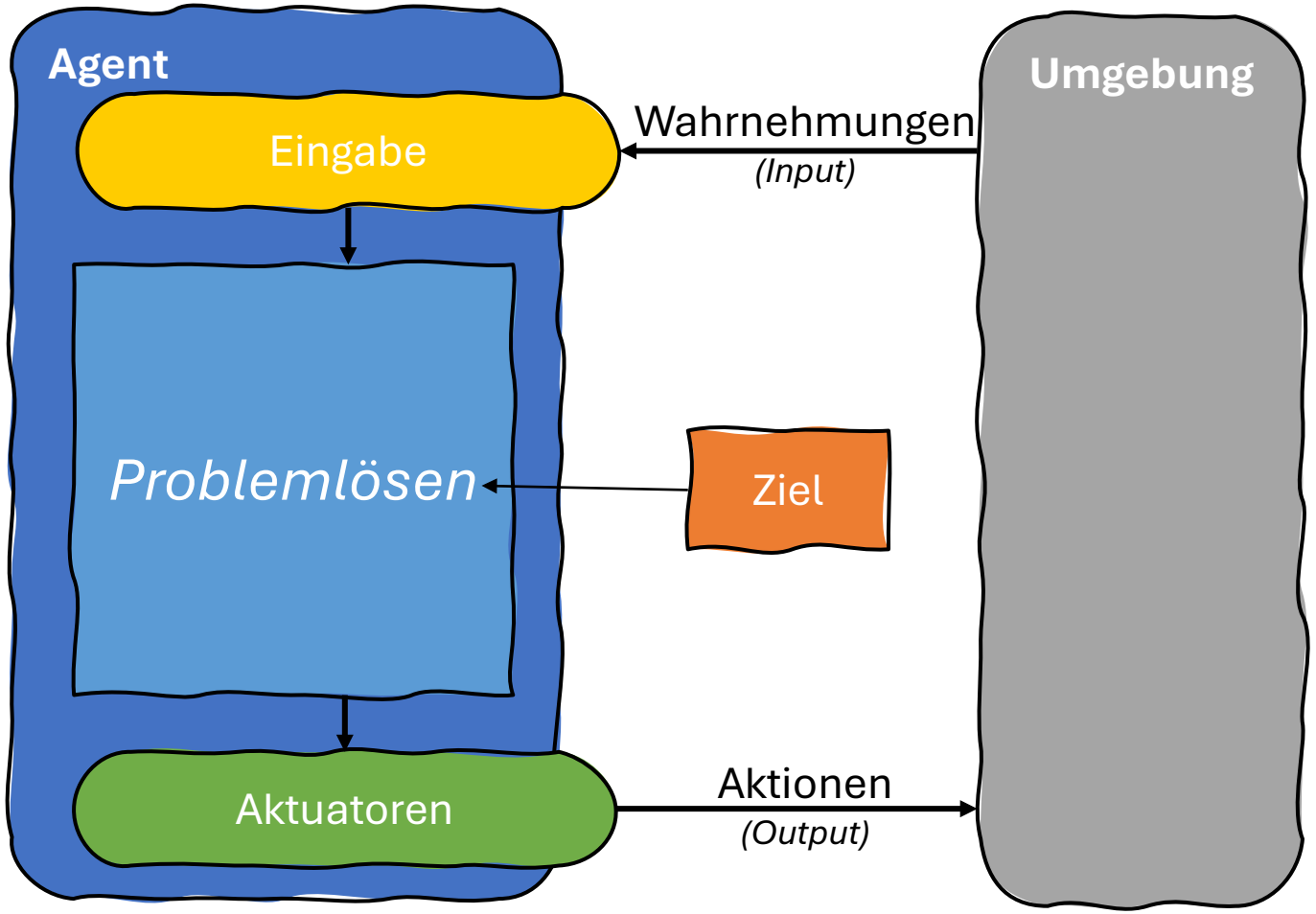
Künstliche Intelligenz | BSc BAI

Knut Hinkelmann

Künstliche Intelligente Agenten – Grundmodell

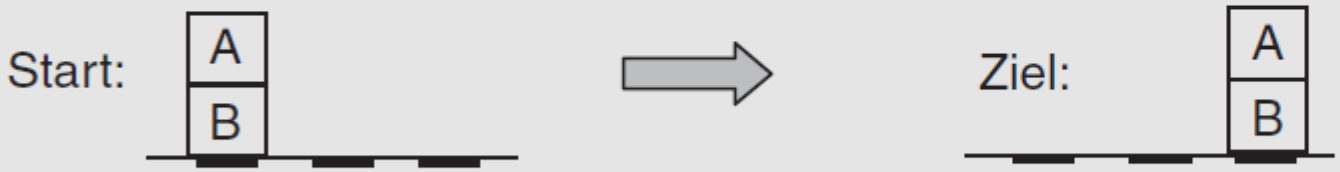
- Das Grundmodell zeigt, wie ein intelligenter Agent wahrnimmt und handelt.
- Wir sehen nun, wie das Handeln des Agenten durch ein Ziel bestimmt wird.

- Der Agent ist in der Lage einen Weg zu finden, um von seinem aktuellen Zustand zu einem Zustand zu gelangen, der das Ziel erfüllt.



Problemlösen: Beispiel Containerlager

In einem Containerlager werden Container umgestapelt. Dafür stehen drei Stellplätze zur Verfügung. Die Container A und B sollen vom linken auf den rechten Stellplatz transportiert werden, wobei der mittlere benutzt werden kann. Die Container können immer nur einzeln transportiert werden.



Wie kann ein Agent, beispielsweise eine automatische Containerbrücke, das Problem lösen?

Lämmel und Cleve (2023), Künstliche Intelligenz.Seite 117

Problemlösen als Suche

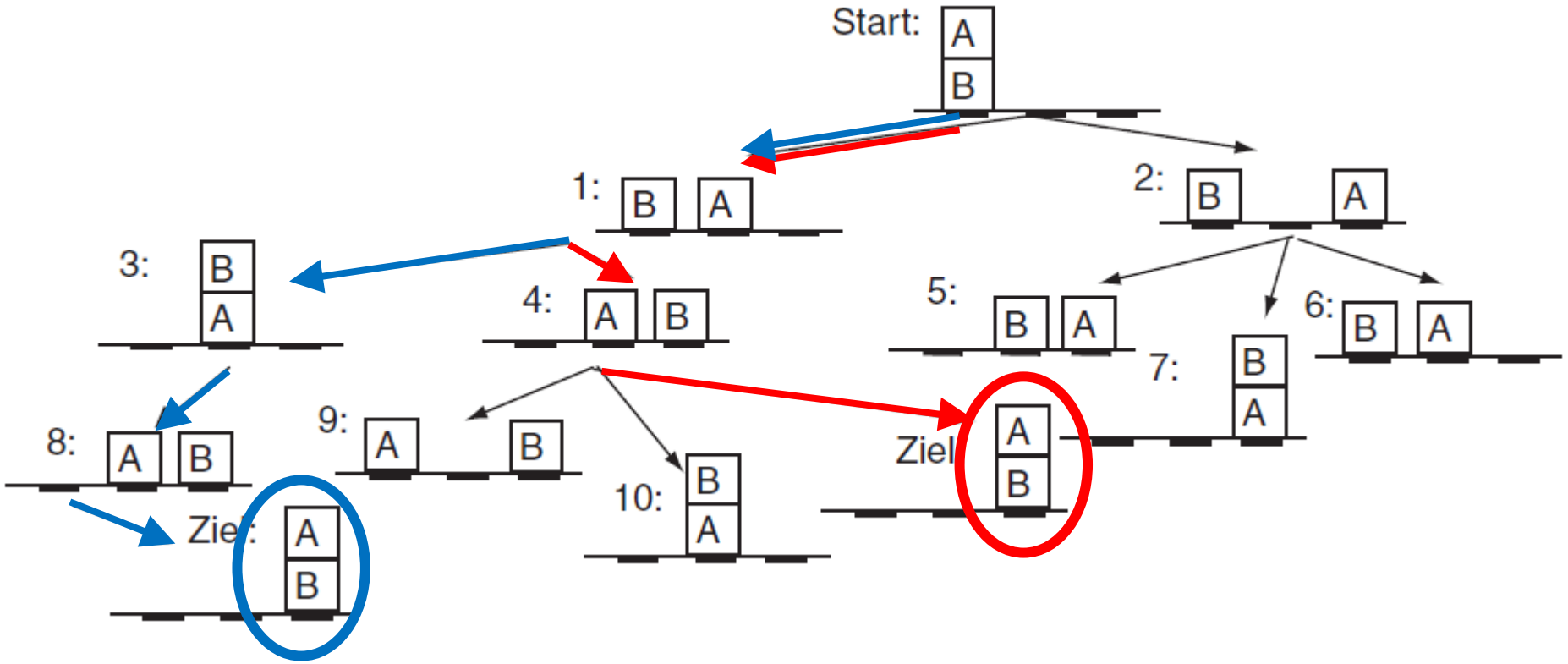
Probleme, für die kein Lösungsalgorithmus bekannt ist, können oft als Suchproblem dargestellt werden:

- Es gibt einen definierten **Startzustand (= Ausgangszustand)**.
- Der gewünschte **Zielzustand** ist vorgegeben.
- Die möglichen **Aktionen** zum Übergang von einem Zustand zum nächsten sind bekannt.

- Ein Suchproblem kann mal als gerichteten **Graphen** darstellen:
  - Die Knoten des Graphen sind Zustände
  - Die Wurzel des Graphen ist der Startzustand,
  - die Kanten sind die Zustandsübergänge (= Aktionen).

- **Problemlösen**: Suche einen Pfad von dem Startzustand zu einem Zielzustand

Suchbaum für das Container-Problem



Problemlösen: Beispiel Wassergefässe

Wir sind im Urlaub auf einem Campingplatz an der Ostsee. Heute sind wir mit der Zubereitung der Vorsuppe dran. Doch bereits beim Abmessen des Wassers gibt es das erste Problem. Laut Beschreibung benötigen wir exakt 2 Liter. Wir haben aber nur 2 kleine Kanister, einer fasst 3 Liter, der andere 4 Liter. Beide haben keine Skalen. Können wir trotzdem exakt 2 Liter bekommen?

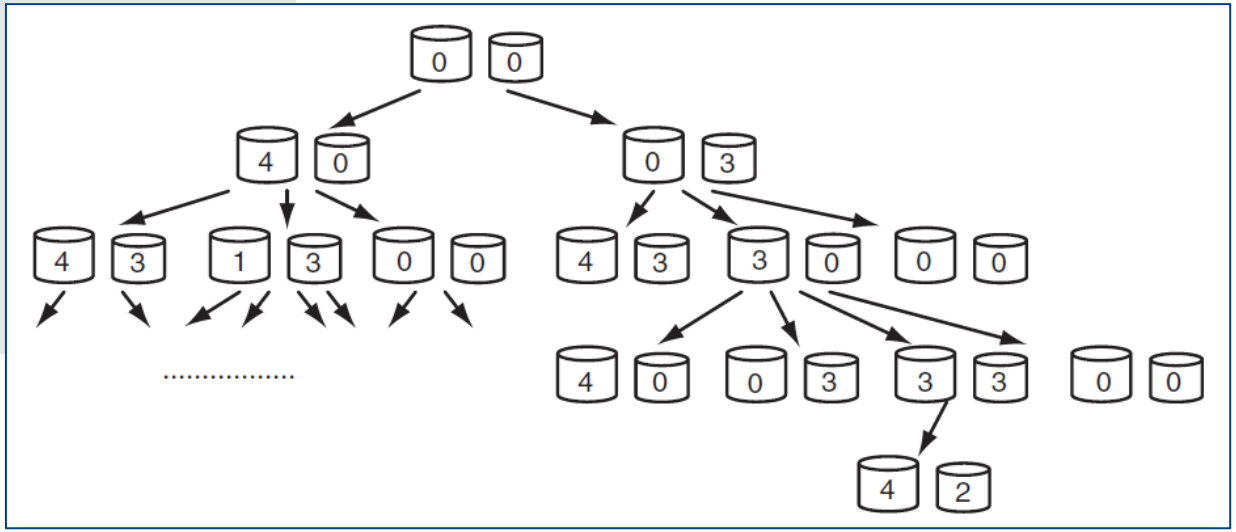
Dieses Problem können wir durch Suchen lösen.

Der **Ausgangszustand**: Beide Gefäße sind leer.

Der **Zielzustand**: In mindestens einem Gefäß sind exakt 2 Liter.

Es gibt nur drei mögliche **Aktionen**, die man ausführen kann:

1. Auffüllen eines Gefäßes
2. Ausschütten eines Gefäßes
3. Umfüllen des Inhaltes eines Gefäßes in das andere



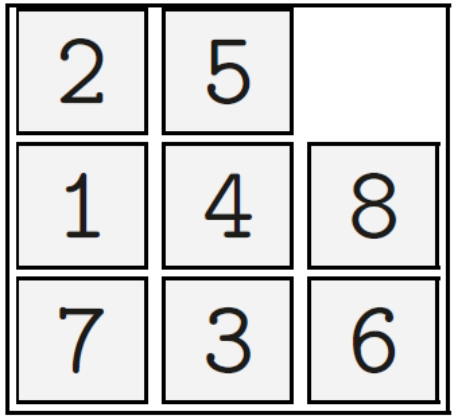
Lämmel und Cleve (2023), Künstliche Intelligenz.Seite 119

Suchproblem

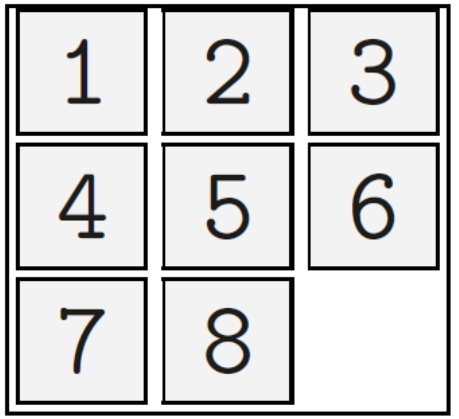
- Ein Suchproblem wird definiert durch folgende Grössen
  - **Zustand**: Beschreibung des Zustands, in dem sich ein Suchagent befindet.
  - **Startzustand**: Der Initialzustand, in dem der Agent gestartet wird.
  - **Zielzustand**: Zustand, in dem der Agent terminiert und die Lösung ausgibt.
  - **Aktionen**: Alle erlaubten Aktionen des Agenten.
  - **Lösung**: Der Pfad im Suchbaum vom Startzustand zum Zielzustand.
  - **Kostenfunktion**: Ordnet jeder Aktion einen Kostenwert zu.
  - **Zustandsraum**: Menge aller Zustände.

(Ertel 2021, S. 107)

Problemlösen: Beispiel



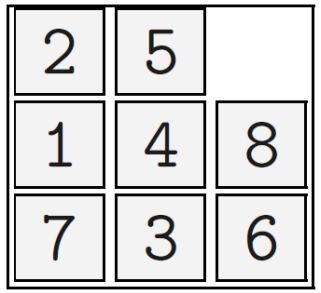
Möglicher Startzustand



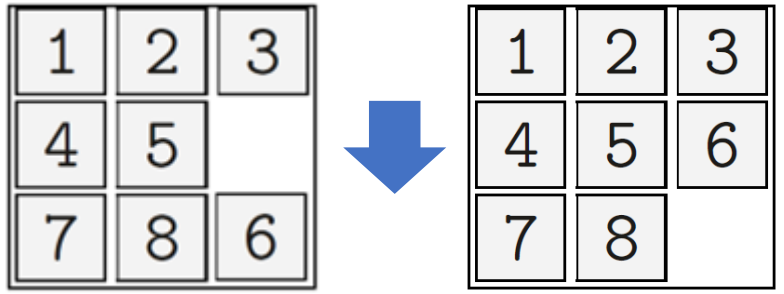
Zielzustand

Das 8-Puzzle als Suchproblem

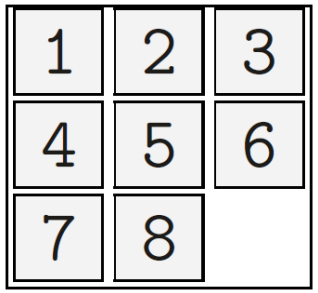
- **Zustand**: 3 ×3 Matrix mit den Werten 1, 2, 3, 4, 5, 6, 7, 8 (je einmal) und einem leeren Feld.
- **Startzustand**: Ein beliebiger Zustand.
- **Zielzustand**: Ein Zustand, in dem die Felder geordnet sind und das leere Feld unten rechts ist.
- **Aktionen**: Bewegung des leeren Feldes nach links, rechts, oben oder unten.
- **Lösung**: Eine Folge von Bewegungen des leeren Feldes, die den Startzustand in den Zielzustand überführen
- **Kostenfunktion**: Die konstante Funktion 1, da alle Aktionen gleich aufwändig sind.



Möglicher Startzustand



Beispiel für eine Aktion

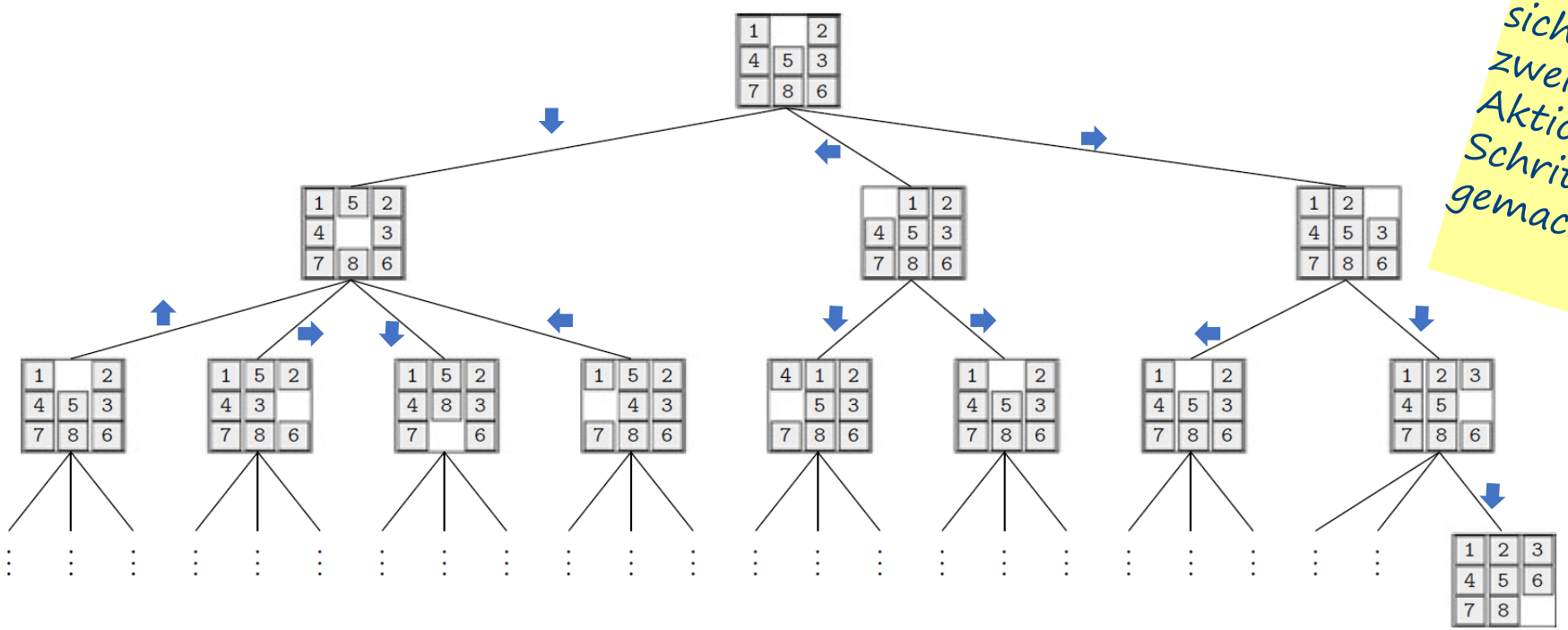


Zielzustand

(Ertel 2021, S. 107)

Suchbaum

- Ein Suchproblem kann als Suchbaum dargestellt werden:
  - Die Wurzel ist der Startzustand
  - Aktionen sind durch Pfeile dargestellt
  - Die Nachfolgeknoten sind Zustände, die durch Anwendung einer Aktion erreicht werden können.

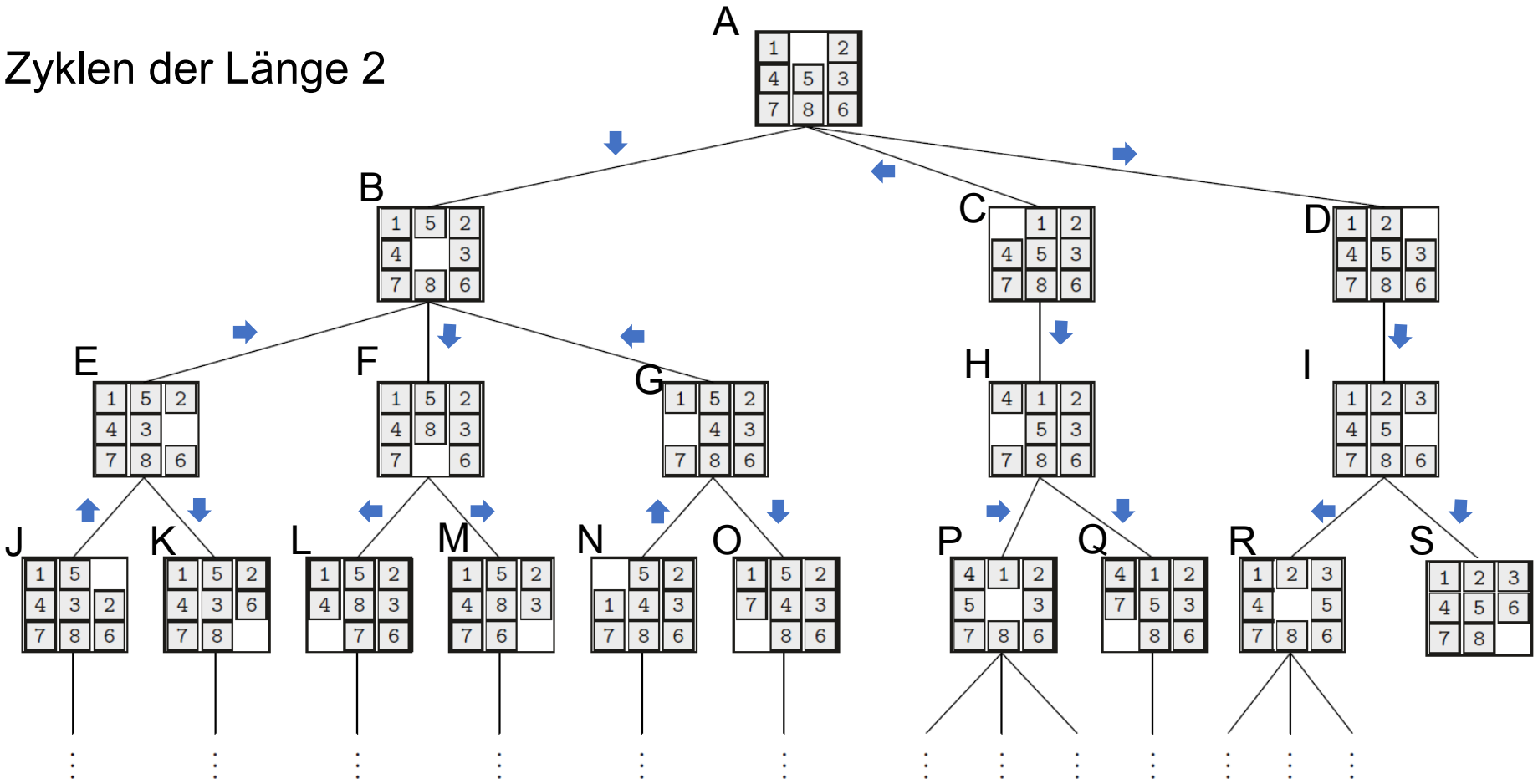


In diesem Beispiel wiederholt sich jeder Zustand mehrfach zwei Ebenen tiefer, denn jede Aktion kann im nächsten Schritt wieder rückgängig gemacht werden

(Ertel, 2021, S. 106)

Suchbaum

– Suchbaum ohne Zyklen der Länge 2



Probleme der realen Welt

Es gibt eine ganze Reihe von Problemen, die sich als Suchprobleme darstellen lassen

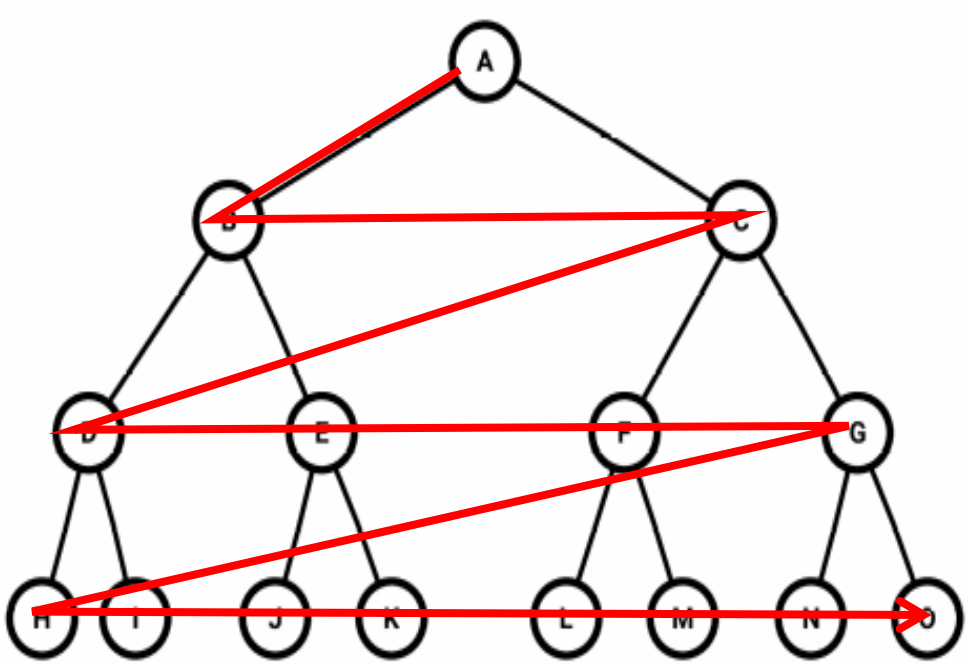
- Routenplanung
  - Navigationssysteme
  - Flugplanung
  - Routing von Daten durch das Internet
- Tourenplanung (Travelling Salesman Problem)
- Montageplanung
- Kombinatorische Spiele

Uninformierte Suche

Uninformierte Suche

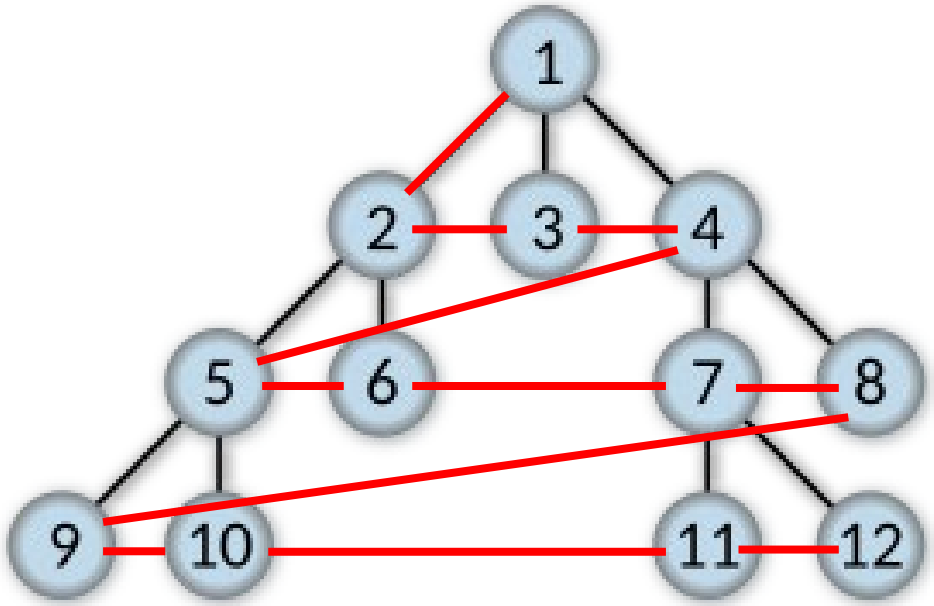
- Eine uninformierte Suche ist ein Suchalgorithmus, der keine zusätzlichen Informationen über den Zustand oder das Ziel nutzt, um effizient zum Ziel zu gelangen. Der Algorithmus kann also nicht beurteilen, welche Zustände "vielversprechender" sind als andere.
- Die uninformierte Suche entspricht Strategien zum Durchlaufen des Suchbaums. Zu den gängigen uninformierten Suchmethoden gehören:
  - **Breitensuche** (**Breadth-First Search**):
    - Untersucht zunächst alle Knoten einer Ebeneund geht dann in die nächste Ebene.
    - Sie findet garantiert den kürzesten Weg in einem ungewichteten Graphen zu finden, ist jedoch speicherintensiv.
  - **Tiefensuche** (**Depth-First Search**):
    - Geht so tief wie möglich in einen Pfad, bevor sie zurückverfolgt und einen neuen Pfad versucht.
    - Sie ist speichereffizienter als Breitensuche, findet jedoch nicht immer eine Lösung und nicht immer die beste Lösung.
  - **Iterative Tiefensuche** (**Iterative Deepening Depth-First Search**):
    - Kombiniert Eigenschaften von Breiten- und Tiefensuche
    - Erhöht schrittweise die maximale Tiefe, die die Tiefensuche erkundet, bis das Ziel gefunden wird.
- Uninformierte Suche kann in Situationen verwendet werden, in denen keine zusätzliche Information über die Zielzustände oder die Kosten bekannt ist.

Breitensuche - Prinzip



- Bei der Breitensuche wird der Suchbaum *Ebene für Ebene* exploriert.
  - Ist der aktive Knoten ein Zielzustand, ist das Problem gelöst
  - Ist der aktive Knoten kein Zielzustand, wird der Knoten erweitert und die Nachfolgeknoten werden gespeichert

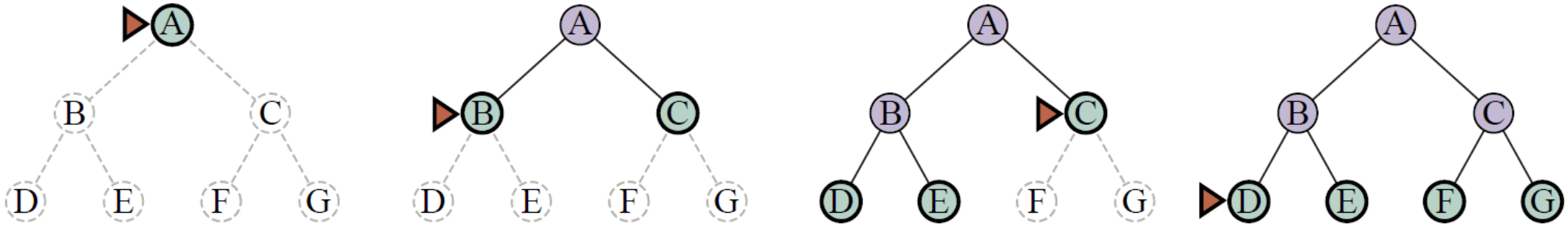
Breitensuche - Prinzip



- Bei der Breitensuche wird der Suchbaum *Ebene für Ebene* exploriert.
  - Ist der aktive Knoten ein Zielzustand, ist das Problem gelöst
  - Ist der aktive Knoten kein Zielzustand, wird der Knoten erweitert und die Nachfolgeknoten werden gespeichert

Breitensuche

- Die Abbildung veranschaulicht die Breitensuche in einem einfachen Binärbaum.
- In jeder Phase wird der als nächstes zu erweiternde Knoten durch die dreieckige Markierung angezeigt.

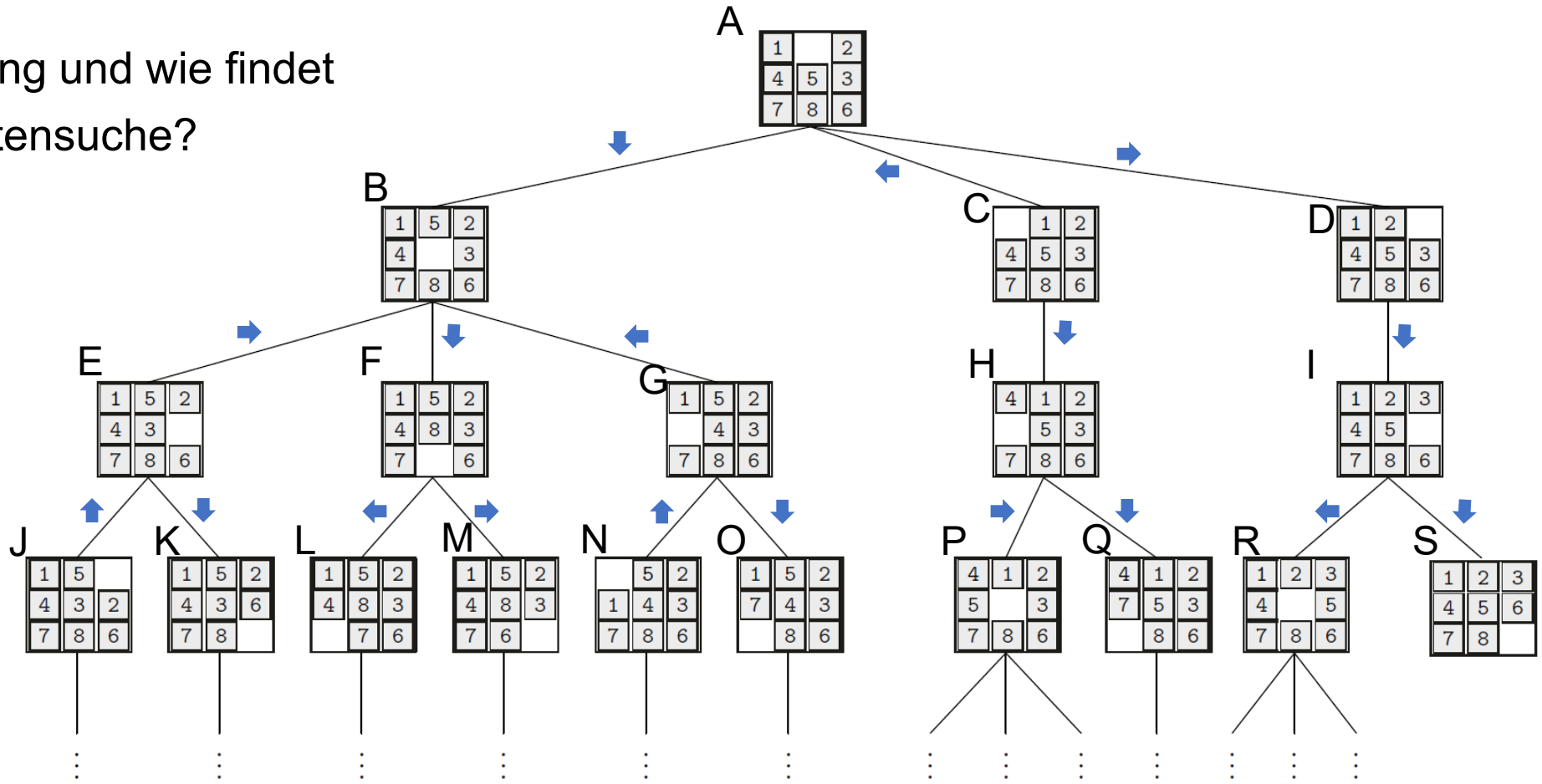


Die Bearbeitung der Knoten erfolgt nach der FIFO-Strategie: First In First Out (Schlange – Queue)

Bild aus (Russell & Norvig, 2022, S. 94)

Suchbaum

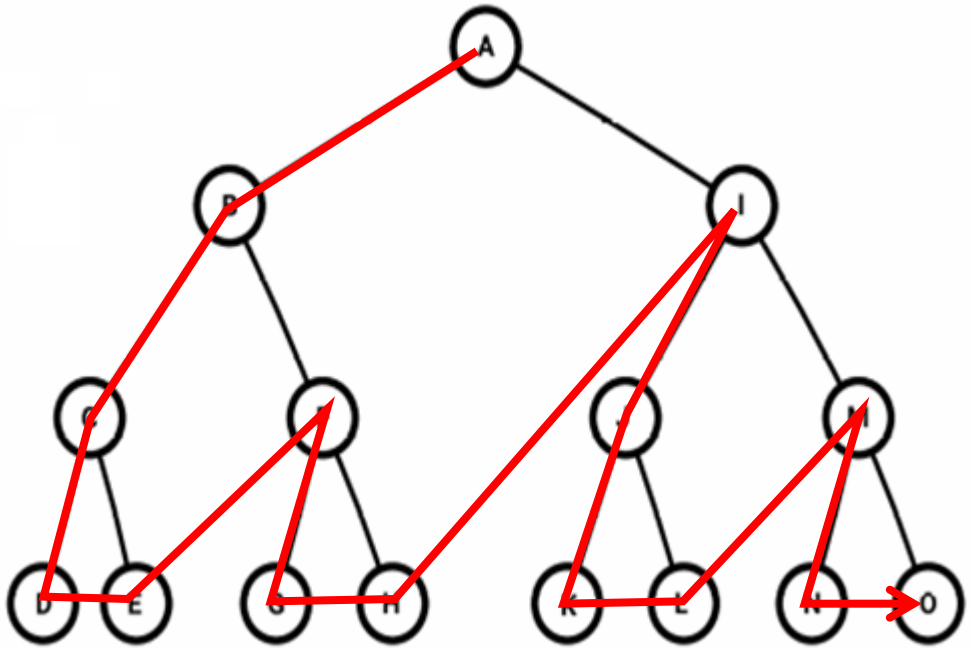
– Was ist die Lösung und wie findet man sie mit Breitensuche?



Algorithmus für die Breitensuche

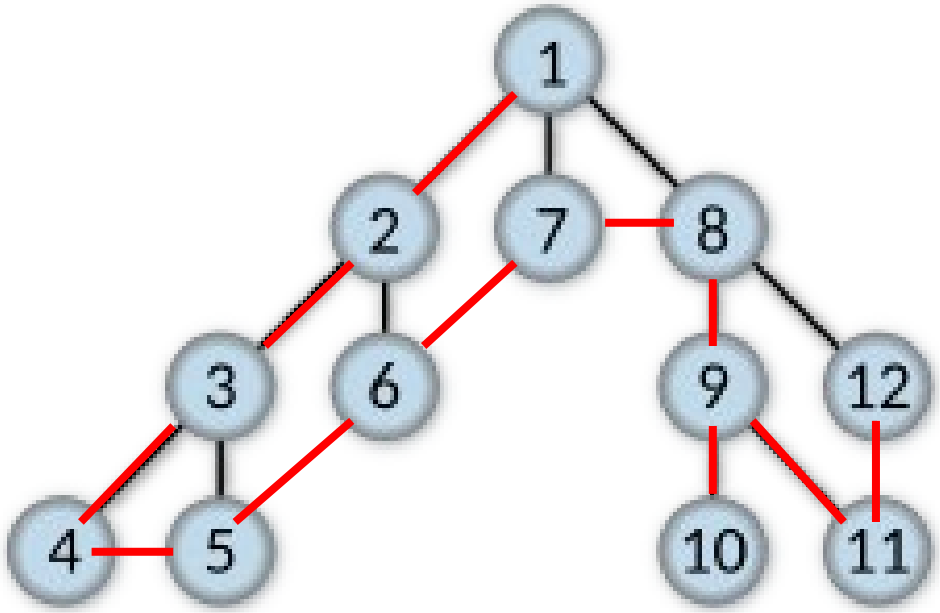
```
BREITENSUCHE (Knotenliste, Ziel)
   NeueKnoten = ∅
   FOR ALL Knoten ∈ Knotenliste
      IF Ziel erreicht (Knoten, Ziel)
         THEN Return (”Lösung gefunden“, Knoten); STOP
      NeueKnoten = Append (NeueKnoten, Nachfolger(Knoten))
      IF NeueKnoten = ∅
         THEN Return(Breitensuche (NeueKnoten, Ziel))
         ELSE Return (”keine Lösung“)
```

Tiefensuche - Prinzip



- Bei der Tiefensuche wird immer zuerst der tiefste Knoten expandiert
- Wenn ein Knoten einen Zielzustand darstellt, ist man fertig
- Wenn ein Knoten keinen Zielzustand darstellt, gibt es zwei Möglichkeiten.
  - Hat der Knoten noch Nachfolgeknoten, wird der erste Nachfolgeknoten zum aktiven Knoten
  - Hat der Knoten keine Nachfolger, wird mittels Backtracking rückwärts bei der letzten Verzweigung der nächste offene Knoten expandiert

Tiefensuche - Prinzip



- Bei der Tiefensuche wird immer zuerst der tiefste Knoten expandiert
- Wenn ein Knoten einen Zielzustand darstellt, ist man fertig
- Wenn ein Knoten keinen Zielzustand darstellt, gibt es zwei Möglichkeiten.
  - Hat der Knoten noch Nachfolgeknoten, wird der erste Nachfolgeknoten zum aktiven Knoten
  - Hat der Knoten keine Nachfolger, wird mittels Backtracking rückwärts bei der letzten Verzweigung der nächste offene Knoten expandiert

Tiefensuche

- Die Abbildung veranschaulicht die Tiefensuche in einem einfachen Binärbaum.
- In jeder Phase wird der als nächstes zu erweiternde Knoten durch die dreieckige Markierung angezeigt.
- Wenn kein Nachfolgeknoten existiert, wird der noch nicht expandierte Knoten des Vorgängerknotens expandiert (Backtracking)

Die Bearbeitung der Knoten erfolgt nach der LIFO-Strategie – Last In First Out (Stapel – Stack)

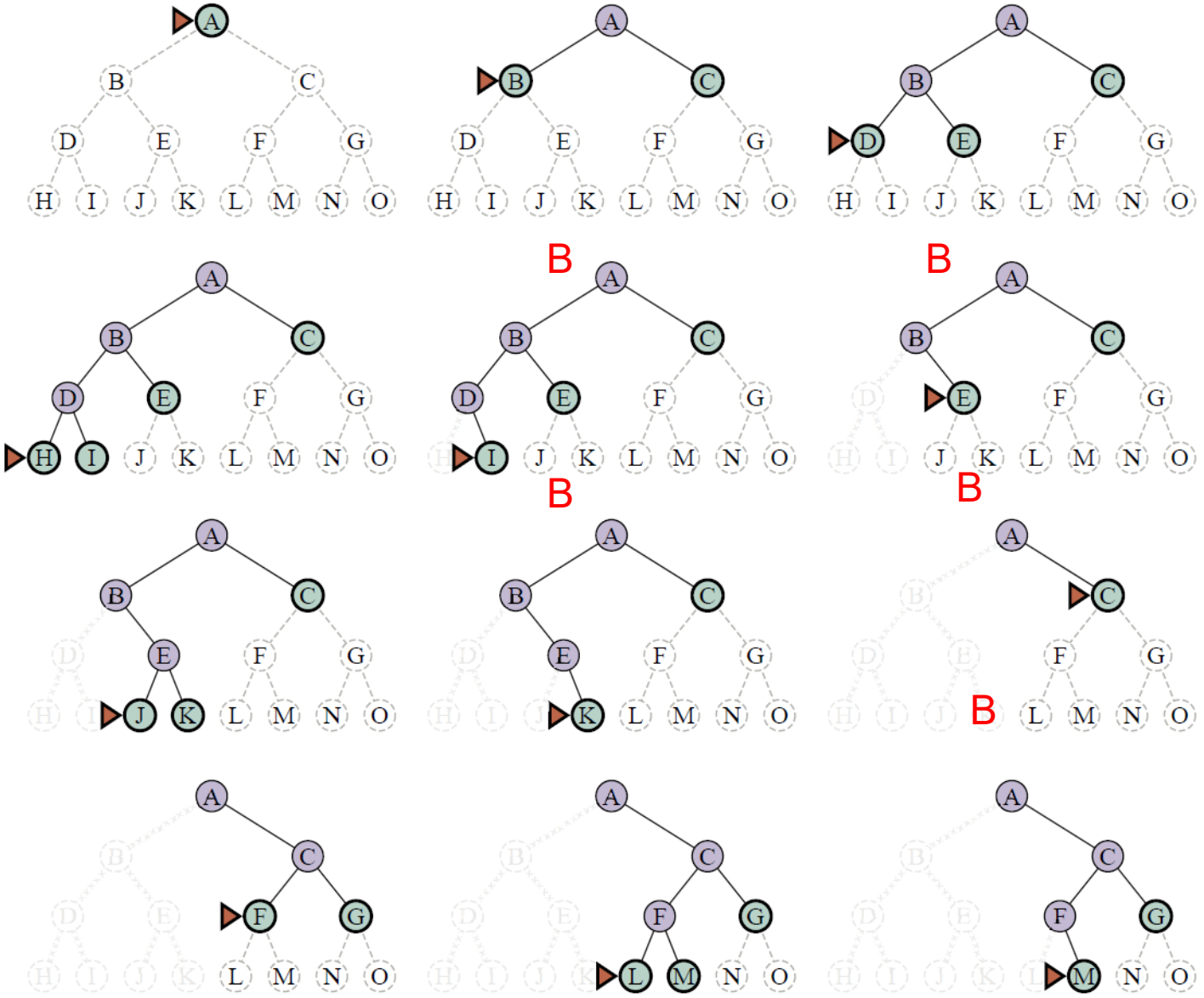


Bild aus (Russel & Norvig, 2022, S. 97)

Algorithmus für die Tiefensuche

```
TIEFENSUCHE (Knoten, Ziel)
IF ZielErreicht(Knoten, Ziel)
   THEN Return (”Lösung gefunden“)
NeueKnoten = Nachfolger (Knoten)
WHILE NeueKnoten =/= ø
   Ergebnis = TIEFENSUCHE (Erster(NeueKnoten), Ziel)
   IF Ergebnis = ”Lösung gefunden“
      THEN Return (”Lösung gefunden“)
   NeueKnoten = Rest (NeueKnoten)
Return (”keine Lösung“)
```

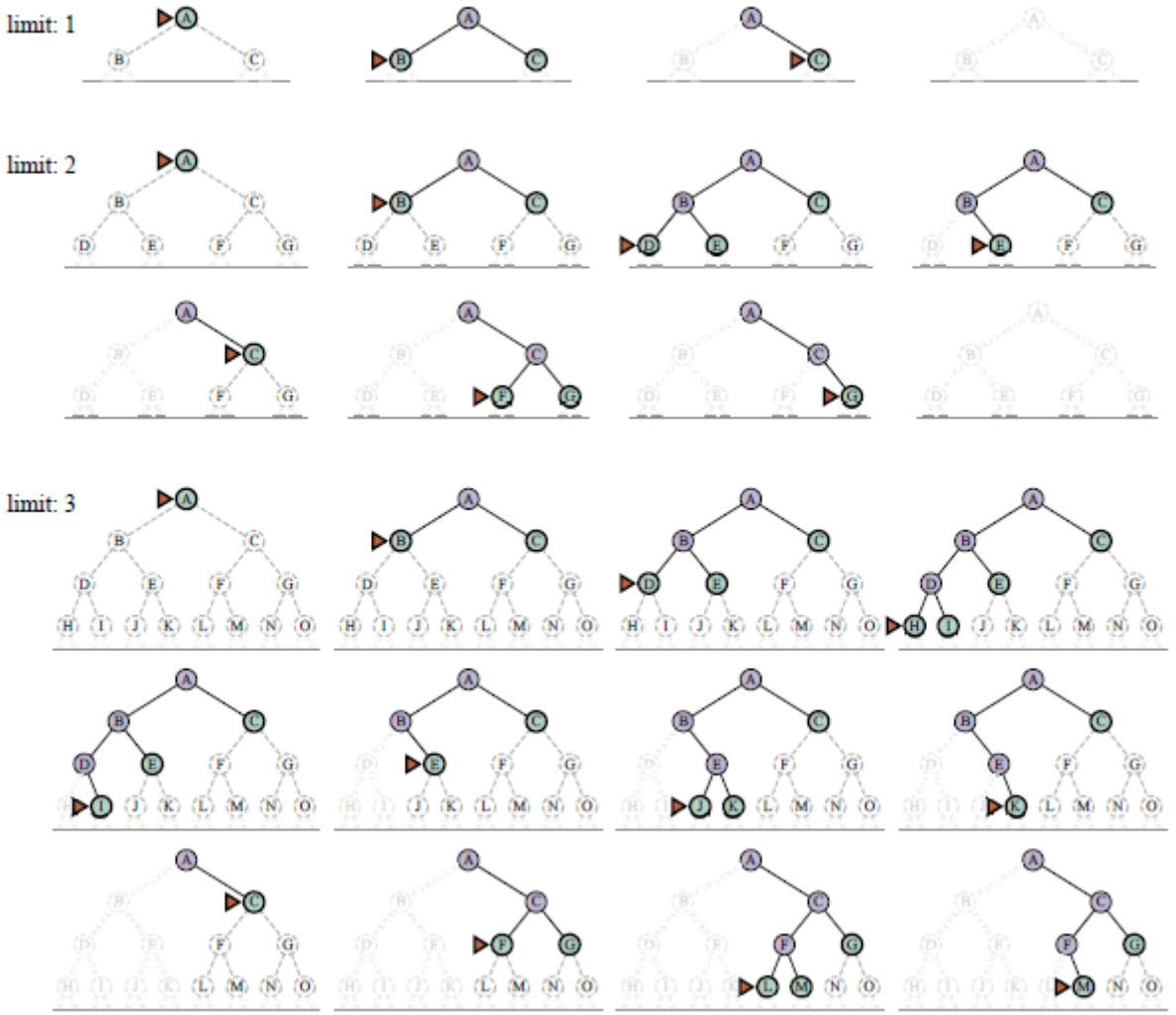
Breitensuche vs. Tiefensuche

Breitensuche	Tiefensuche
Die Breitensuche ist vollständig.	Tiefensuche ist nicht vollständig bei unendlich tiefen Bäumen, z.B. wenn die Tiefensuche in eine Endlosschleife läuft
Die Breitensuche findet immer die optimale Lösung (wenn die Kosten aller Aktionen gleich sind)	Die Tiefensuche findet nicht immer die optimale Lösung
Rechenzeit und Speicherplatz wachsen exponentiell mit der Tiefe des Baumes	Die Tiefensuche benötigt viel weniger Speicherplatz als die Breitensuche, denn in jeder Tiefe werden maximal b Knoten gespeichert, wobei b = Anzahl der Nachfolger pro Knoten (Verzweigungsfaktor)

Ein Kompromiss ist die Iterative Tiefensuche

Iterative Deepening

- Die iterative Tiefensuche vermeidet unendliche Suche:
  - Man startet die Tiefensuche mit Tiefenschranke 1
  - Falls keine Lösung gefunden wird, erhöht man die Schranke um 1 und startet die Suche erneut
- Analyse: Iterative Deepening …
  - … braucht gleich wenig Speicherplatz wie Tiefensuche
  - … ist vollständig wie Breitensuche



Heuristische Suche

Heuristische Suche

- Breitensuche und Tiefensuche heissen auch uninformierte Suche
  - Sie durchlaufen den ganzen Suchbaum, was sehr aufwändig sein kann
- In der Praxis wird eine schnell gefundene gute Lösung einer optimalen, aber nur mit grossem Aufwand herbeigeführten, Entscheidung vorgezogen.
- Heuristische Suche hat das Ziel, mit «wenig» Aufwand eine «gute» Lösung zu finden
- **Heuristiken** sind Problemlösungsstrategien, die in vielen Fällen zu einer schnelleren Lösung führen als die uninformierte Suche
- Zur mathematischen Modellierung einer Heuristik wird eine **heuristische Bewertungsfunktion** f(s) für Zustände verwendet

Heuristische Bewertungsfunktionen

- Zur mathematischen Modellierung einer Heuristik wird eine **heuristische Bewertungsfunktion** f(s) für Zustände verwendet
- Idee:
  - Die Heuristik ist eine Vereinfachung der Aufgabenstellung, die mit wenig Rechenaufwand zu lösen ist
  - Die Kosten für das vereinfachte Problem dienen als Abschätzung der Kosten für das eigentliche Problem

Algorithmus für die Heuristische Suche

„Einsortieren(X,Y)“ fügt die Elemente aus der Liste X in die aufsteigend sortierte Liste Y ein. Als Sortierschlüssel wird die heuristische Bewertung verwendet.

```
HEURISTISCHESUCHE (Start, Ziel)
Knotenliste = [Start]
WHILE True
   IF Knotenliste = ø
      THEN Return(”keine Lösung“)
   Knoten = Erster (Knotenliste)
   Knotenliste = Rest (Knotenliste)
   IF Ziel erreicht (Knoten, Ziel)
      THEN Return (”Lösung gefunden“, Knoten)
   Knotenliste = Einsortieren (Nachfolger(Knoten),Knotenliste)
```

A*-Suche

- Beim A*-Algorithmus berücksichtigt die heuristische Bewertungsfunktion zwei Kosten

  f(s) = g(s) + h(s)

- Dabei sind

  g(s) = Summe der vom Start bis zum aktuellen Knoten angefallenen Kosten

  h(s) = Abschätzung der Kosten vom aktuellen Knoten zum Zielzustand

Zulässige Heuristik und optimale Lösung

Eine heuristische Kostenschätzfunktion h(s), welche die tatsächlichen Kosten vom Zustand s zum Ziel nie überschätzt, heißt **zulässig** (engl. **admissible**).

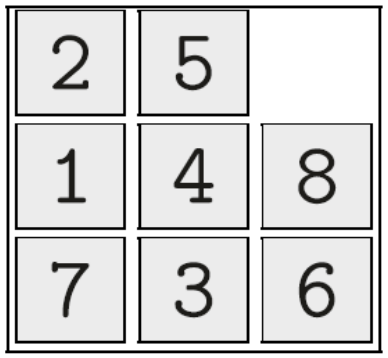
Der A*-Algorithmus ist **optimal**, d.h. er findet immer die Lösung mit den niedrigsten Gesamtkosten, wenn die Heuristik h **zulässig** ist.

(Ertel 2021, S. 121)

Heuristische Bewertung für 8-Puzzle

- Für das 8-Puzzle gibt es zwei zulässige heuristische Bewertungsfunktionen
  - Die Heuristik h_1 zählt die Anzahl der Plättchen, die nicht an der richtigen Stelle liegen
  - Die Heuristik h_2 misst den Manhattan-Abstand. Für jedes Plättchen werden horizontaler und vertikaler Abstand zum gleichen Plättchen im Zielzustand addiert. Dieser Wert wird dann über alle Plättchen aufsummiert, was den Wert $h_2(s)$ ergibt

- Beispiel: heuristische Bewertung des abgebildeten Zustands:



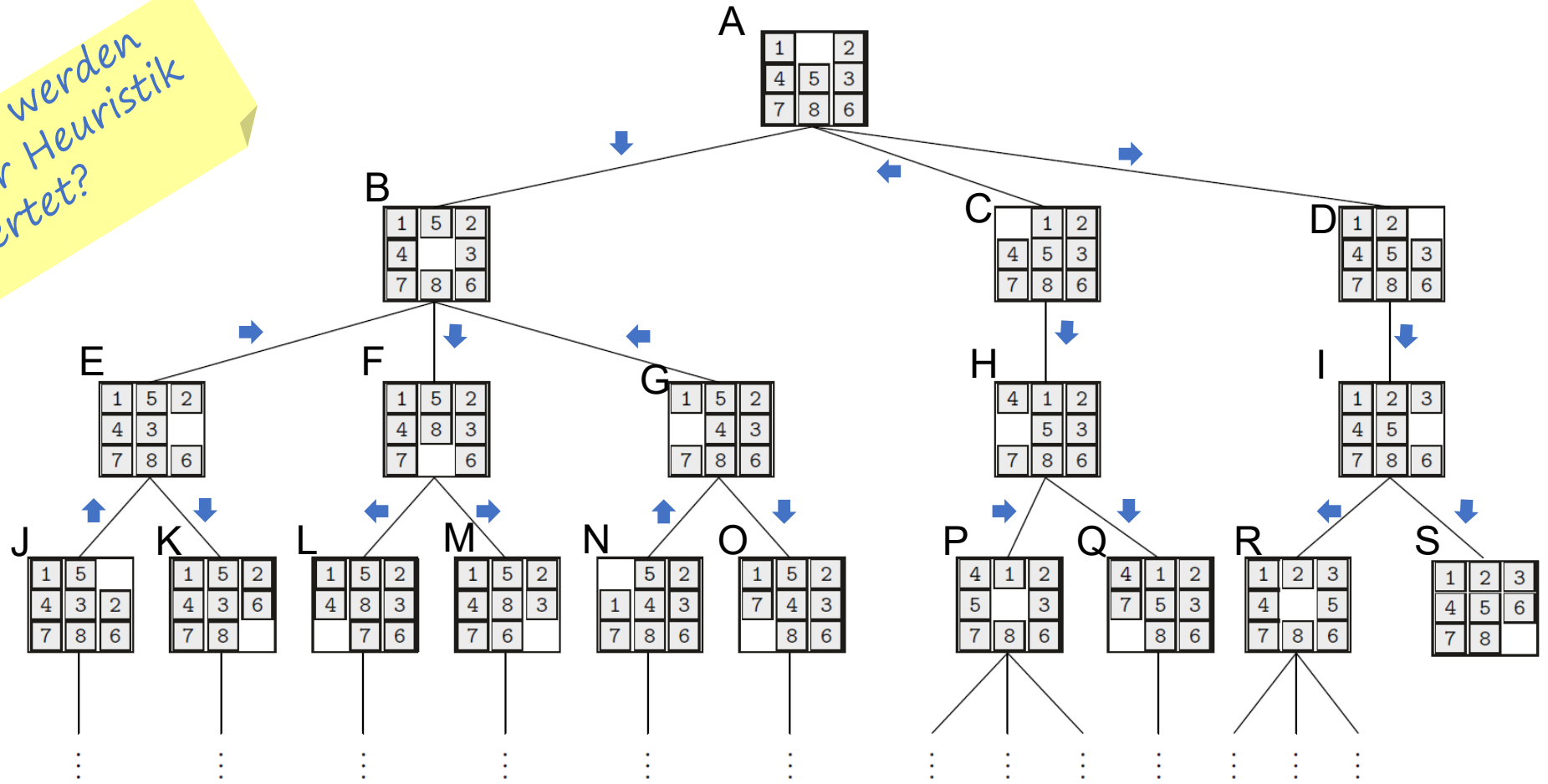
$h_1(s) = 7$

$h_2(s) = 1 + 1 + 1 + 1 + 2 + 0 + 3 + 1 = 10$

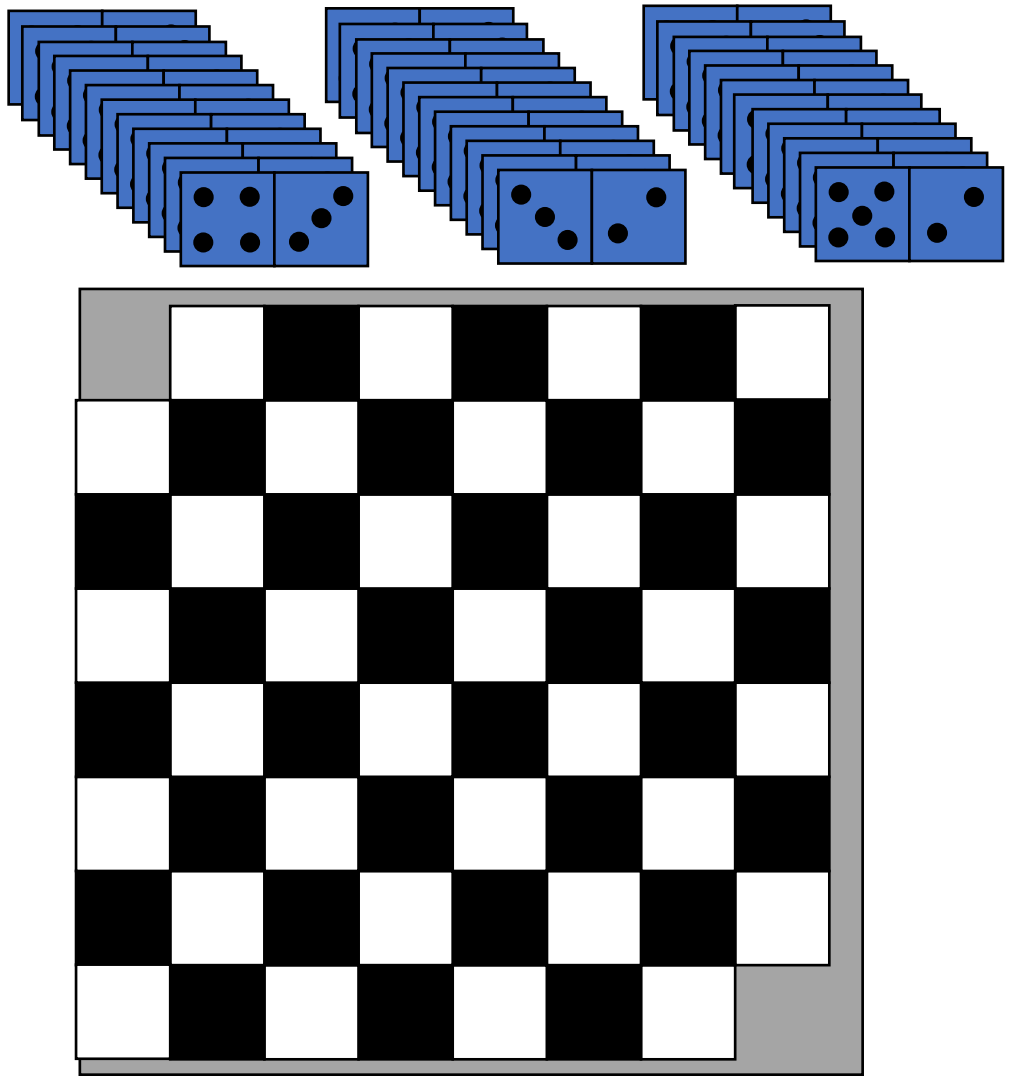
Beispiel: Anwendung der Heuristik h1 für 8-Puzzle

Suchbaum

Welche Unterbäume werden bei Anwendung der Heuristik H1 nicht ausgewertet?



Problemlösen: Suche vs. Wissen



- Auf einem Schachbrett wurden zwei gebenüberliegende Ecken herausgesägt.
- Ein Dominostein bedeckt genau zwei Felder

- Ist es möglich, mit Dominosteinen alle Felder des Schachbretts abzudecken?

Fazit

- Uninformierte Suche scheitert bei schwierigen kombinatorischen Suchproblemen meist an der Grösse des Suchraumes
- Heuristische Suche reduziert den effektiven Verzweigungsfaktors
  - Heuristiken basieren auf Wissen über die Anwendung
  - Die eigentliche Aufgabe des Entwicklers beim Lösen schwieriger Suchprobleme besteht daher im Entwurf von Heuristiken, die den effektiven Verzweigungsfaktor stark verkleinern.
- Heuristiken bringen nur Gewinn bei lösbaren Problem. Um zu erkennen, dass ein Problem nicht lösbar ist, muss man den ganzen Suchraum durchsuchen
  - Ausnahme: Man hat zusätzliches Wissen, um unlösbare Probleme vorab zu erkennen (vgl. Dominosteine auf dem Schachbrett)