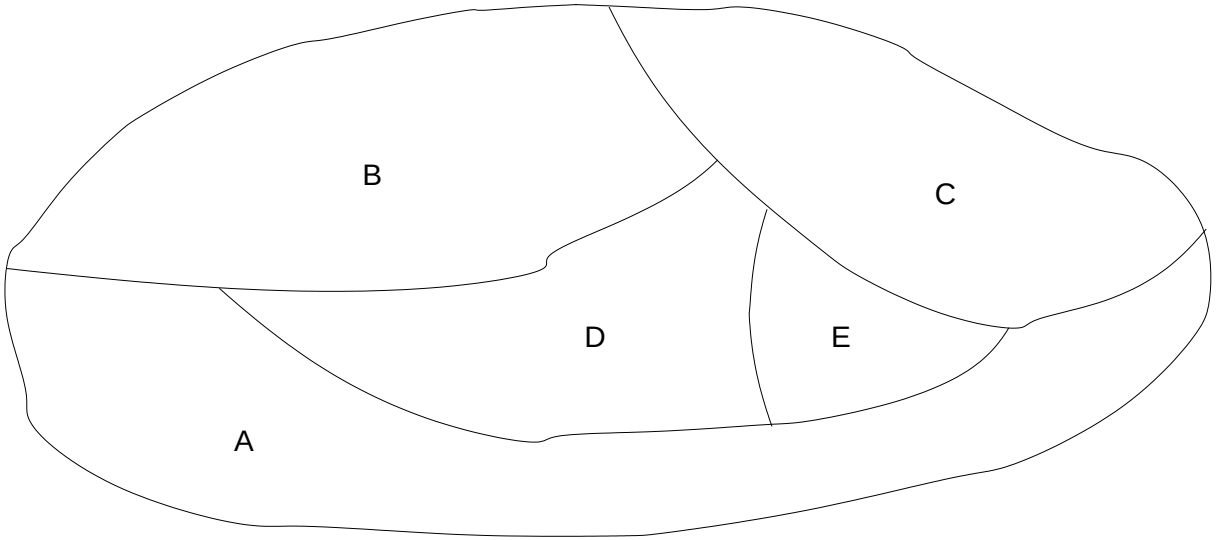
2. Suchverfahren

- Viele Probleme der Entscheidungsunterstützung lassen sich auf ein Suchproblem zurückführen.
- Die Eigenschaften und Lösungsverfahren von Suchproblemen sind daher von grundlegender Bedeutung für dieses Gebiet.
- Suchverfahren sind ein klassisches Thema innerhalb der Wissensverarbeitung.

Färbeproblem



Beispiel 2.1. Die angegebene Landkarte mit den Ländern A, B, C, D und E ist so mit den Farben rot, blau, gelb und orange zu färben, daß keine zwei benachbarten Länder die gleiche Farbe haben.

Färbeproblem (2)

- Ein naives *generate-and-test* Verfahren würde 4^5 mögliche Farbkonstellationen prüfen.
- Allgemein sind $\mathfrak{m}^{\mathfrak{n}}$ Farbkonstellationen zu prüfen, mit $\mathfrak{m} :=$ Anzahl der Farben und $\mathfrak{n} :=$ Anzahl der Länder.

☞ Ineffizient!

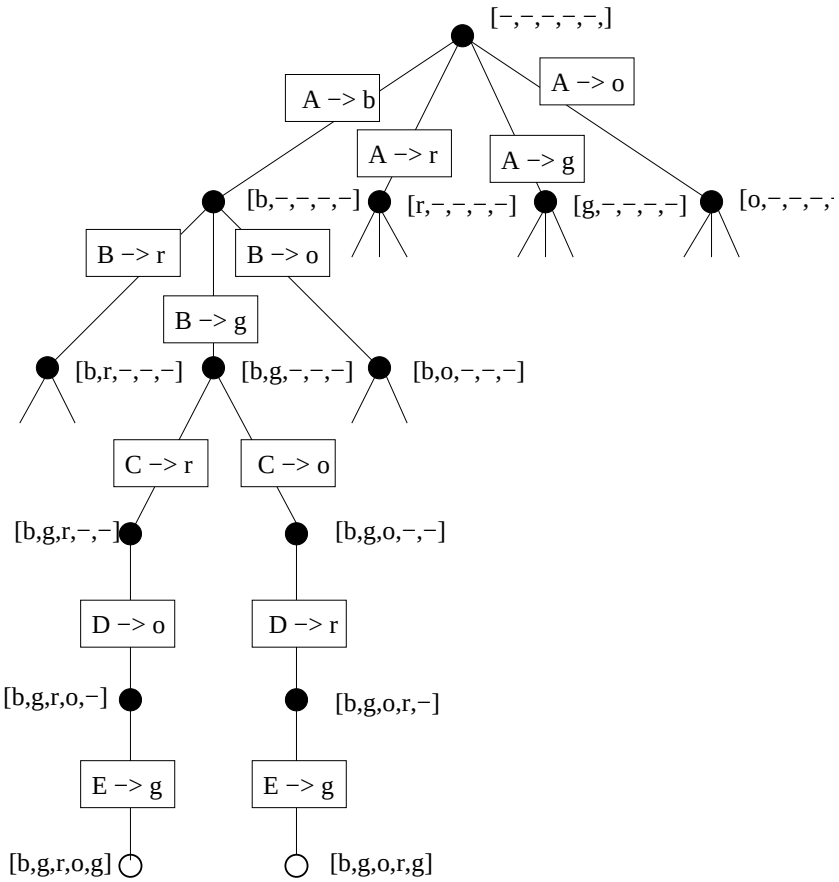
Färbeproblem (3)

- Es scheint sinnvoller zu sein, die Länder der Reihe nach zu färben.
- So kann man Zwischenzustände bei der Problemlösung durch Teilfärbungen beschreiben, etwa

$$(\mathrm{A} \leftarrow \mathrm{rot}, \mathrm{B} \leftarrow \mathrm{blau}, \mathrm{C} \leftarrow \mathrm{gelb})$$

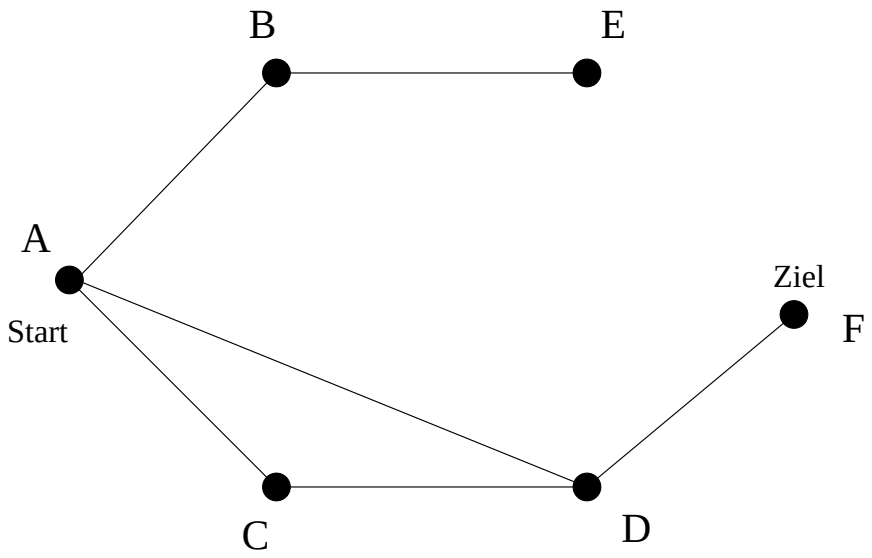
- Nach der Zuordnung $(\mathrm{A} \leftarrow \mathrm{blau}, \mathrm{B} \leftarrow \mathrm{blau})$ kann man direkt abbrechen.
- Die Problemlösung startet mit der leeren Färbung $()$.
- Ziel ist es, eine komplette zulässige Färbung zu erreichen.
- Die Schritte im Laufe der Problemlösung lassen sich durch Zustandsübergangsoperatoren beschreiben.

Suchbaum



- Die Lösung des Färbeproblems läßt sich als Suchbaum darstellen.
- Die Knoten des Suchbaums entsprechen den Zuständen (zulässige Teilfärbungen).
- Die Kanten entsprechen den Operatoren.

Routenproblem



Beispiel 2.2. Gegeben ist eine Karte mit Städten und Straßen, die die Städte miteinander verbinden.

Gesucht ist eine Route von einem Startort zu einem Zielort.

Suchbaum: Tafel ✎.

Zustandsraum

Ein *Problem* wird repräsentiert durch Wissen, das ein DSS nutzen kann, um zu entscheiden, welche Aktionen ausführt werden sollen.

Für Suchprobleme läßt sich das Wissen repräsentieren durch:

- Ein *Zustand* stellt das Wissen zu einem bestimmten Zeitpunkt der Lösungsfindung dar.
- Der *Zustandsraum* ist die Menge aller Zustände.
- *Zustandsübergangsoperatoren* beschreiben, wie ausgehend von einem Zustand andere Zustände des Zustandsraums erreicht werden können.
- Der *Startzustand* ist der Zustand, der zu Beginn der Lösungsfindung vorliegt. Er läßt sich explizit angeben.
- Die Menge der *Zielzustände* charakterisiert die Lösungen des Problems. Zielzustände lassen sich in der Regel nur implizit angeben, z.B. über ein Testprädikat.

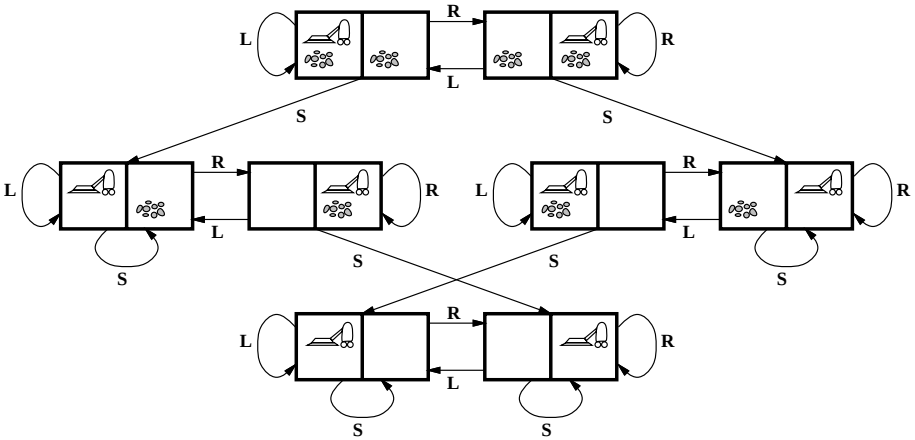
- Um verschiedene Lösungen vergleichen zu können, können Folgen von Aktionen *Kosten* zugewiesen werden.

  Typischerweise definiert man Kosten abhängig von einem Zustand und einer Aktion. Die Gesamtkosten für eine Aktionenfolge ergeben sich aus den Einzelkosten.

Beispiel 2.3.

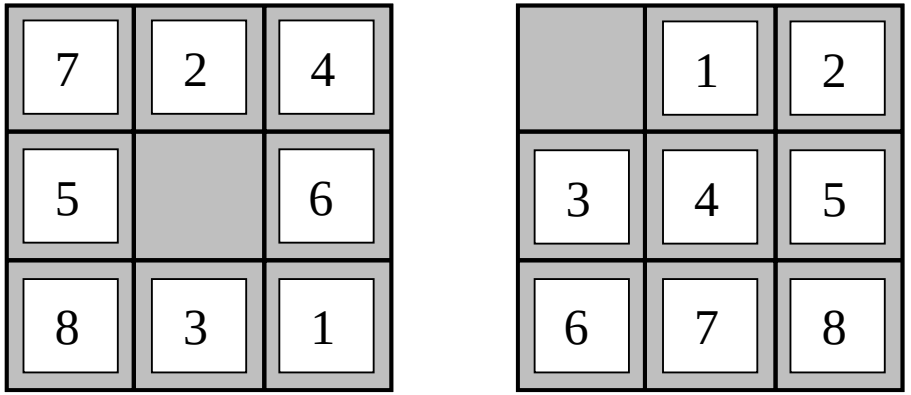
Staubsaugerwelt:

- Operationen: rechts, links, saugen
- Ziel: alle Räume müssen sauber sein



8-Puzzle:

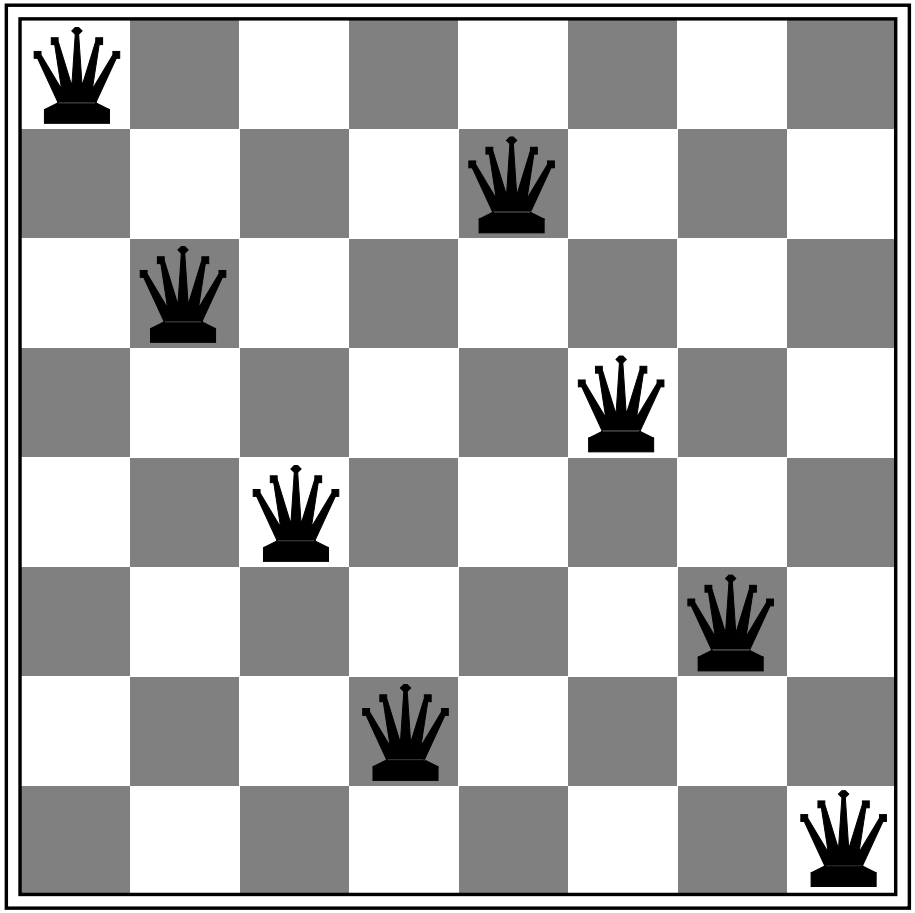
- Operationen: rechts, links, oben, unten
- Ziel: Endkonfiguration



Start State Goal State

8-Damen-Problem:

- Zustand: Plazierung zwischen 0 und 8 Damen
- Operationen: eine Dame plazieren
- Ziel: alle Damen so plaziert, daß sie sich nicht bedrohen



Beispiel 2.4. Suchprobleme in der realen Welt:

- Routenberechnung, Navigation
- Travelling Salesman Problem, Vehicle Routing: Tourenplanung, Fahrzeugeinsatzplanung, Gebietsplanung
- VLSI Layout: Cell Layout und Channel Routing
- Robot Navigation: virtuell und physisch, wie findet der Roboter/Avatar sein Ziel?
- Assembly Sequencing: Finde eine Reihenfolge zum Zusammenbau komplexer Werkstücke aus Einzelteilen

Zustandsraum und Suchbaum

- Die Knoten eines Suchbaums stellen die Zustände dar.
- Die Kanten entsprechen den Zustandsübergangsoperatoren.
- Die Wurzel entspricht dem Startzustand.
- Die Zielknoten sind die Zielzustände.
- Die Berechnung der Nachfolger eines Knotens s wird als *Expansion* des Knotens s bezeichnet.

Zustandsraum und Suchbaum (2)

- Der Zustandsraum beschreibt nur, wie man prinzipiell zu einer Lösung gelangen kann,
- aber nicht, wie man effizient zu dieser kommt.
- Ganz wesentlich für eine effiziente Problemsösung sind:
  - ☞ das Verfahren, das festlegt, in welcher Reihenfolge die Zustände untersucht bzw. expandiert werden sowie
  - ☞ die Bewertung der einzelnen Zustände.

Uninformierte Suchverfahren

- Suchverfahren, die über die Beschreibung des Zustandsraums hinaus keine Zusatzinformation benutzen, heißen *uninformierte Suchverfahren*.
- Insbesondere findet keine Bewertung der einzelnen Zustände statt.
- Dementsprechend unterscheiden sich die Verfahren im wesentlichen darin, in welcher Reihenfolge die Zustände expandiert werden.
- Die wichtigsten Vertreter der uninformierten Suchverfahren sind die *Breitensuche* und die *Tiefensuche*.

Uninformierte Suchverfahren (2)

- Ausgehend von der Wurzel des Suchbaums (Startzustand) werden die Knoten sukzessive expandiert.
- Später wird man von den Nachfolgern des expandierten Knotens weiterarbeiten, solange bis man einen Zielknoten gefunden hat.
- Die Liste der Knoten, die gerade in Bearbeitung sind, heißt *Agenda (open list)*.
- Knoten der Agenda sind generiert, aber noch nicht expandiert.
- Expandierte Knoten werden auch als *closed* bezeichnet.

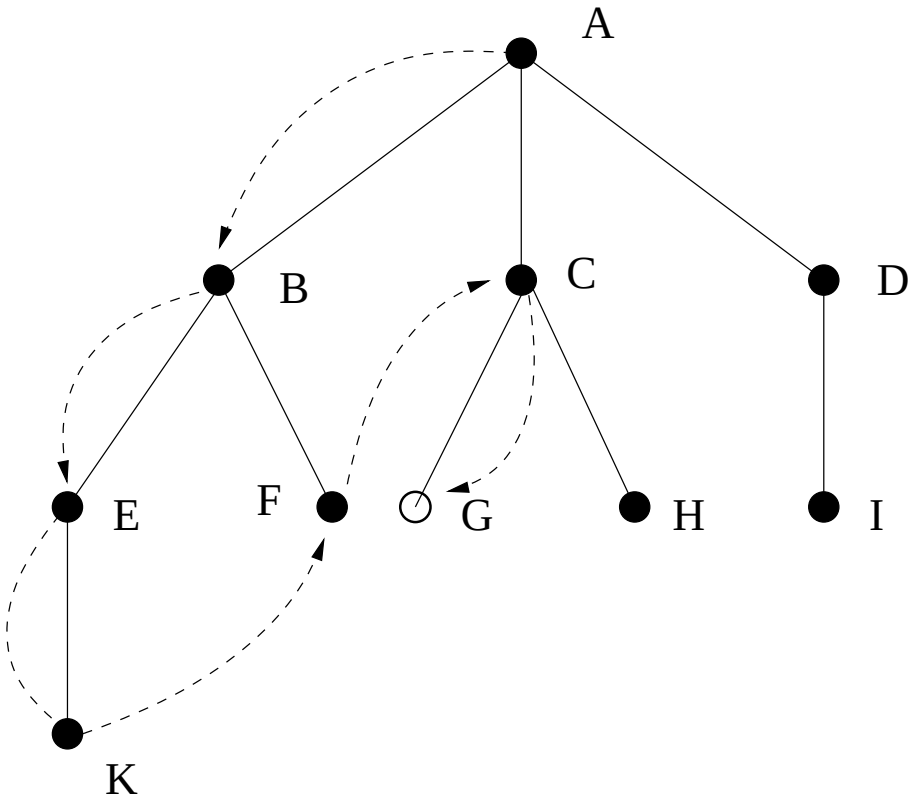
Uninformierte Suchverfahren (3)

- Breitensuche und Tiefensuche laufen nach dem gleichen Schema ab.
- Zu Beginn der Suche besteht die Agenda aus dem Startzustand.
- In einer beliebigen Iteration wird der erste Knoten s_{akt} aus der Agenda genommen.
- Wenn s_{akt} ein Zielzustand ist, hat man eine Lösung gefunden.
- Ist s_{akt} kein Zielzustand, so wird s_{akt} expandiert, d.h. alle Nachfolger von s_{akt} werden in die Agenda eingefügt.
- Breiten- und Tiefensuche unterscheiden sich darin, wo die Nachfolger in die Agenda eingefügt werden.

Tiefensuche

☞ Bei der Tiefensuche werden die Nachfolger eines expandierten Knotens s_{akt} an den Anfang der Agenda eingefügt.

- Die Agenda entspricht einem Kellerspeicher (Stack).
- Liefert ein Knoten, der kein Zielknoten ist, keine neuen Knoten, so wird die Suche fortgesetzt an dem nächstgelegenen Knoten, für den noch nicht alle Nachfolger expandiert wurden.
- Dies entspricht einem Backtracking.

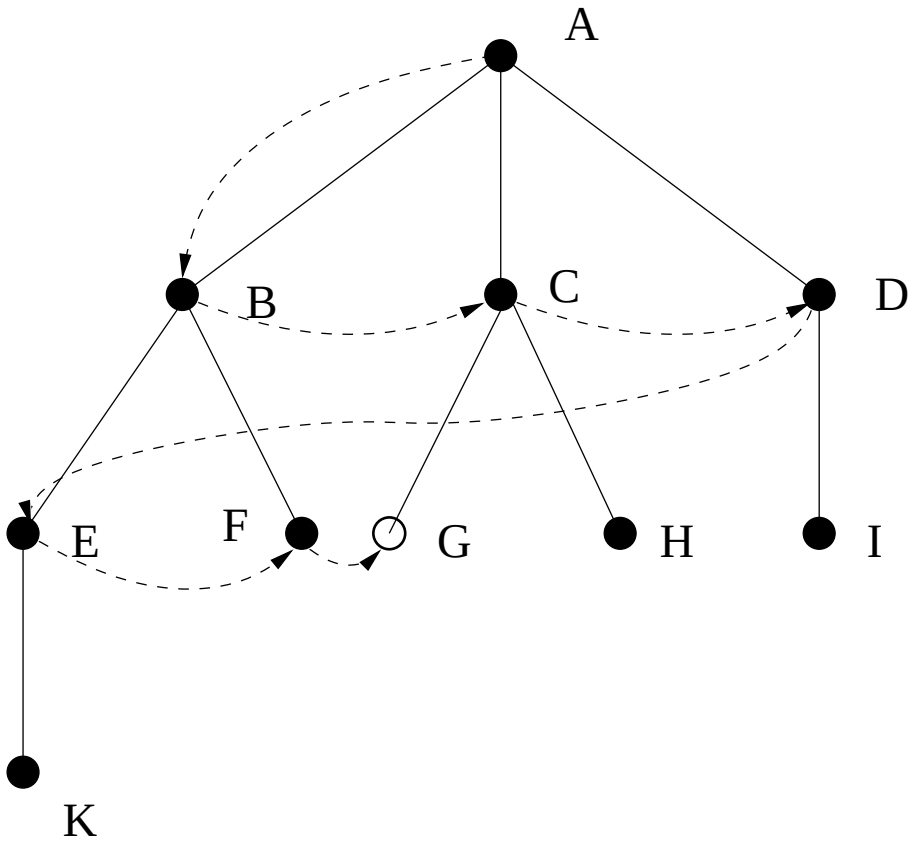


Schritt	Agenda	s_{akt}
1	(A)	A
2	(B,C,D)	B
3	(E,F,C,D)	E
4	(K,F,C,D)	K
5	(F,C,D)	F
6	(C,D)	C
7	(G,H,D)	G

Breitensuche

☞ Bei der Breitensuche werden die Nachfolger eines expandierten Knotens s_{akt} an das Ende der Agenda eingefügt.

- Die Agenda entspricht einer Warteschlange (Queue).



Schritt	Agenda	s_{akt}
1	(A)	A
2	(B,C,D)	B
3	(C,D,E,F)	C
4	(D,E,F,G,H)	D
5	(E,F,G,H,I)	E
6	(F,G,H,I,K)	F
7	(G,H,I,K)	G

Algorithmen

Algorithmus 2.1. [Tiefensuche]

$Agenda := (Startknoten)$;
`while` $Agenda \neq ()$ `do`
    $s_{akt} := first(Agenda)$;
    Entferne s_{akt} aus der Agenda;
    `if` s_{akt} ist Zielknoten `then` s_{akt} ist Lösung; STOP;
    $Agenda := Nachfolger(s_{akt}) + Agenda$;
`end`
Problem hat keine Lösung; STOP;

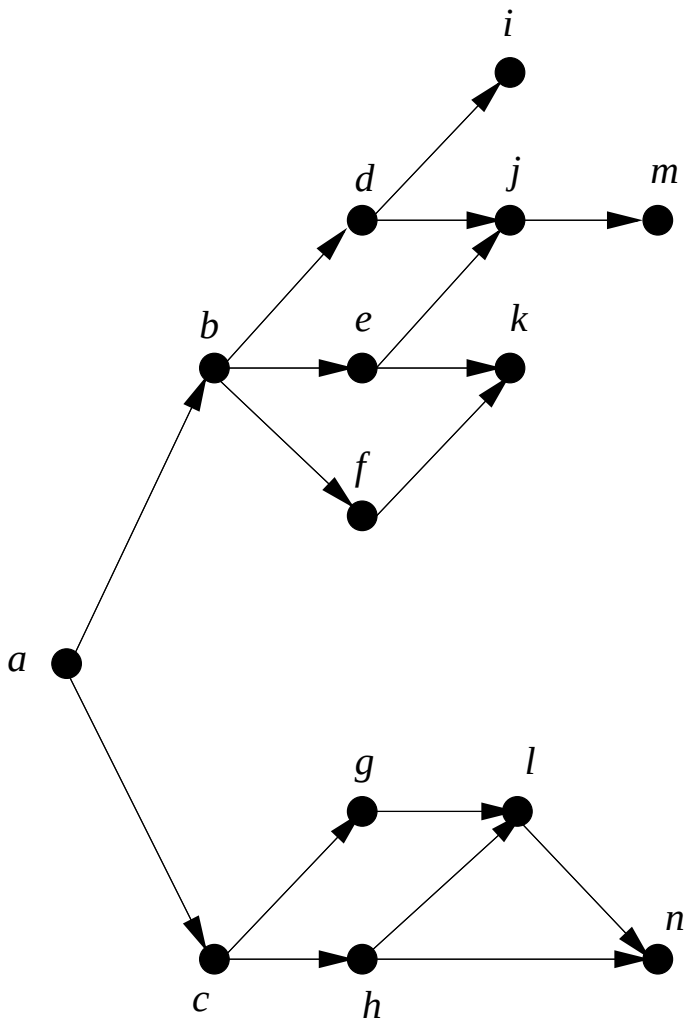
Algorithmen (2)

Algorithmus 2.2. [Breitensuche]

$Agenda := (Startknoten)$;
`while` $Agenda \neq ()$ `do`
    $s_{akt} := first(Agenda)$;
    Entferne s_{akt} aus der Agenda;
    `if` s_{akt} ist Zielknoten `then` s_{akt} ist Lösung; STOP;
    $Agenda := Agenda + Nachfolger(s_{akt})$;
`end`
Problem hat keine Lösung; STOP;

Beispiel 2.5. Suche einen Weg von a nach e mit Tiefensuche bzw. Breitensuche.

Tafel ✎.



Beispiel 2.6. Ein Weinhändler hat drei Krüge, einen von 9 Liter, einen von 7 Liter und einen von 4 Liter Inhalt.

Auf den Krügen sind keine Litermarkierungen angebracht.

Der 9-Liter-Krug ist gefüllt, die anderen sind leer.

Die Krüge sollen so umgefüllt werden, daß der 9-Liter-Krug sechs Liter und der 4-Liter-Krug drei Liter enthält.

Tafel ✎.

Eigenschaften von Suchverfahren

Definition 2.1. Ein Suchverfahren heißt *vollständig*, wenn für jeden Suchbaum jeder Knoten expandiert werden könnte, solange noch kein Zielknoten gefunden wurde.

- Ein vollständiges Suchverfahren ist fair in dem Sinne, daß jeder Knoten die Chance hat, expandiert zu werden.
- Ein vollständiges Sucherfahren findet auch bei unendlichen Suchbäumen stets eine Lösung, falls eine existiert.

☞ Breitensuche ist vollständig.

☞ Tiefensuche ist nur bei endlichen Suchbäumen vollständig.

Eigenschaften von Suchverfahren (2)

Definition 2.2. Für ein uninformiertes Suchverfahren heißt eine Lösung *optimal*, wenn sie unter allen Lösungen die geringste Tiefe im Suchbaum aufweist.

- ☞ Breitensuche findet eine optimale Lösung (falls existent).
- ☞ Tiefensuche findet i.a. keine optimale Lösung.

Eigenschaften von Suchverfahren (3)

Komplexitäten:

- Für Breiten- und Tiefensuche ist der ungünstigste Fall, daß die Lösung in der "äußersten rechten Ecke" des Suchbaums liegt.
- $\Longrightarrow$ Zeitkomplexität $\mathrm{O}(\mathrm{b}^{\mathrm{t}})$, mit $\mathrm{b} =$ Verzweigungsrate und $\mathrm{t} =$ Tiefe des Zielknotens.
- Bei der Tiefensuche enthält die Agenda die Knoten des aktuellen Suchpfades sowie deren Nachfolger $\Longrightarrow$ Platzkomplexität $\mathrm{O}(\mathrm{bt})$.
- Bei der Breitensuche kann die Agenda eine komplette Ebene des Suchbaums enthalten $\Longrightarrow$ Platzkomplexität $\mathrm{O}(\mathrm{b}^{\mathrm{t}})$.

Varianten von uninformierten Suchverfahren

Tiefensuche mit Begrenzung der Suchtiefe:

- Versucht die Nachteile der Tiefensuche auszuschalten.
- Eine *maximale Suchtiefe* wird vorgegeben.
- Zweige jenseits der maximalen Suchtiefe werden abgeschnitten.
- Hierdurch wird garantiert, daß eine Lösung gefunden wird, wenn eine innerhalb der Suchtiefe existiert.
- Nachteil: Wie groß soll man die maximale Suchtiefe wählen?

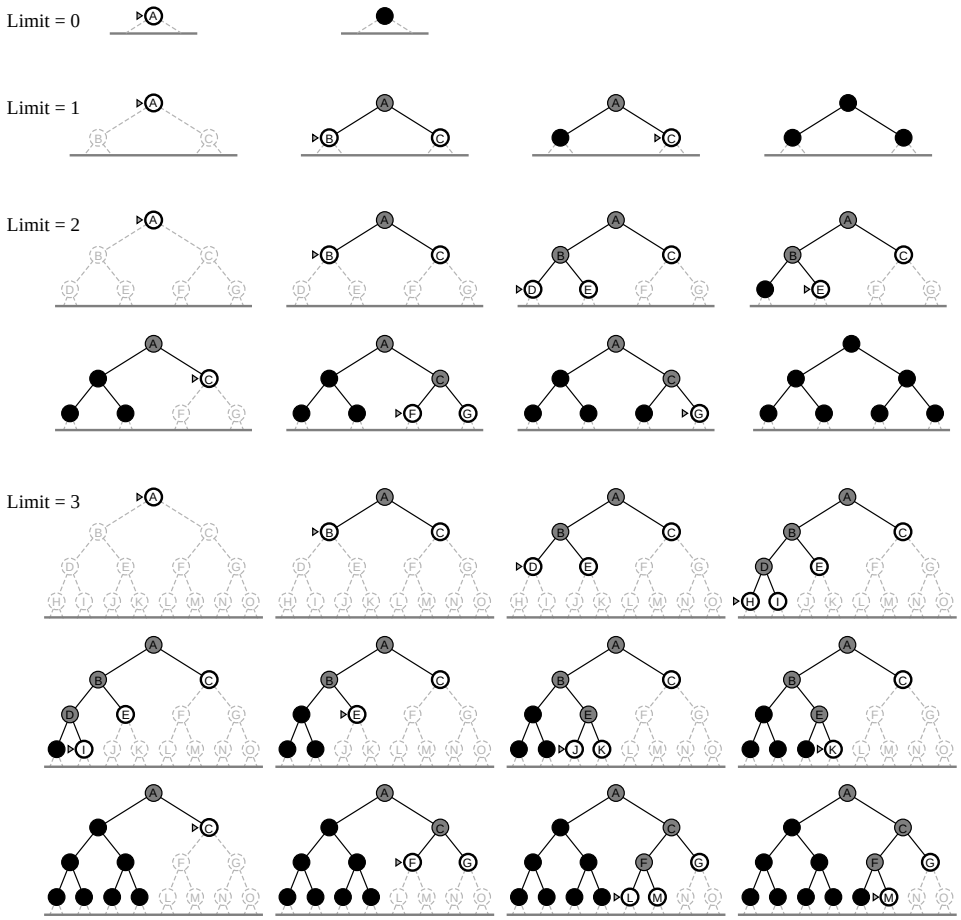
Iterative Deepening:

- Da man nicht weiß, wie groß man die maximale Suchtiefe wählen soll, probiert man dies einfach systematisch durch.
- D.h. man führt immer wieder Tiefensuche mit Begrenzung der Suchtiefe durch und erhöht die maximale Suchtiefe jeweils um 1.
- Bei einer maximalen Suchtiefe t und einem regulären Suchbaum mit Verzweigungsgrad b werden $O(b^t)$ Knoten erzeugt.

☞ Wegen

$$\sum_{d=0}^{t} b^d = \frac{b^{t+1} - 1}{b - 1} < \frac{b}{b - 1} b^t = O(b^t)$$

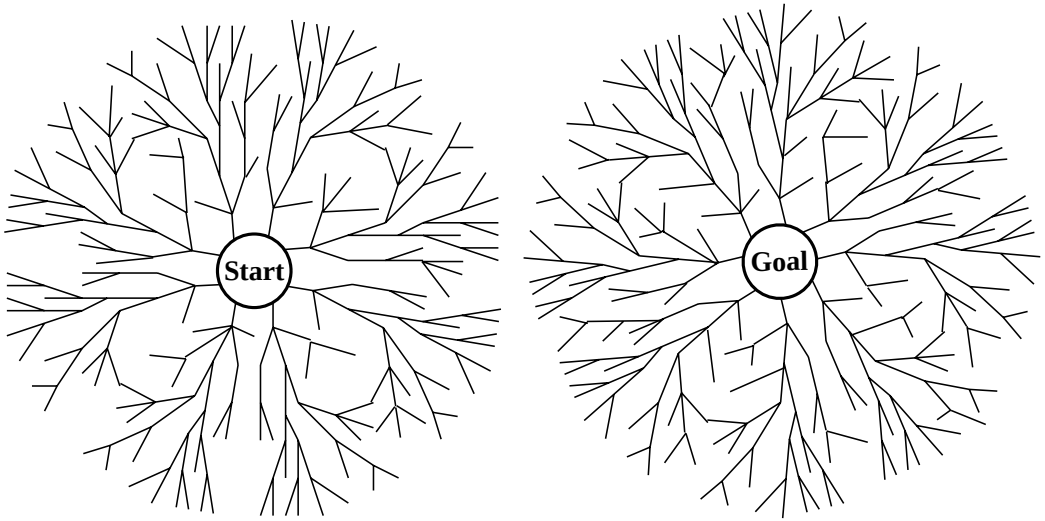
verliert man nur einen konstanten Faktor.



☞ Iterationen bei Iterative Deepening und einem binären Suchbaum

Bidirektionale Suche:

- Gleichzeitige Suche von Start zum Ziel und vom Ziel zum Start.
- Lösungen ergeben sich dort, wo sich die Suchpfade treffen.



- Durch Halbierung der Länge der Suchpfade ergibt sich ein Aufwand von $O(2b^{\frac{d}{2}}) = O(b^{\frac{d}{2}})$.

  ☞ drastische Reduzierung

- Probleme:
  - Zielzustände sind oft nicht explizit bekannt.
  - Zielzustände müssen nicht eindeutig sein.
  - Operatoren sind nicht unbedingt umkehrbar.