

# Backtracking mit Heuristiken

Thomas Dübendorfer

[thomas@duebendorfer.ch](mailto:thomas@duebendorfer.ch)

14. September 2000

# Inhalt und Ablauf

<b>1. Einführendes Beispiel</b>	(4 min)
<b>2. Konzepte zu Backtracking</b>	(8 min)
<b>3. Eingesperrt im Labyrinth</b>	
a) Der Backtracking-Ansatz	(4 min)
b) Der Computer zeigt den Weg	(10 min)
<b>4. Laufzeit</b>	(3 min)
<b>5. Springerwege</b>	
a) Lösungsansatz mit Backtracking	(2 min)
b) Heuristiken zur Optimierung	(3 min)
<b>6. Zusammenfassung</b>	(1 min)
	<hr/>
	(35 min)

# 1. Einführendes Beispiel

**Wann wurde über die Initiative  
„Schweiz ohne Armee“ abgestimmt?**

Die Antwort steht im Internet!

Suche auf den Seiten der Bundesverwaltung:

**[www.admin.ch](http://www.admin.ch)**

Confoederatio Helvetica: Schweizerische Eidgenossenschaft - Confédération Suisse - Confederazione Sv - Netscape

File Edit View Go Communicator Help

Back Forward Reload Home Search Netscape Print Security Shop Stop

Bookmarks Location: <http://www.admin.ch/> What's Related

# www.admin.ch



**Link: „deutsch“**

## CONFOEDERATIO HELVETICA

*deutsch* Willkommen bei den Behörden der Schweizerischen Eidgenossenschaft

*français* Bienvenue sur le site des autorités fédérales suisses

*italiano* Benvenuti presso le autorità della Confederazione Svizzera

*rumantsch* Bainvegni tar las autoritads da la confederaziun svizra

*english* Welcome to the Authorities of the Swiss Confederation

**Frage:** Wann wurde über die Initiative „Schweiz ohne Armee“ abgestimmt?


Document: Done

Einstiegsseite der Schweizerischen Bundesbehörden - Netscape

File Edit View Go Communicator Help

Back Forward Reload Home Search Netscape Print Security Shop Stop

Bookmarks Location: <http://www.admin.ch/ch/index.de.html> What's Related

 **CONFOEDERATIO HELVETICA**  
Die Bundesbehörden der Schweizerischen Eidgenossenschaft

rumantsch  
english  
français  
italiano

[Grussadresse der Bundeskanzlerin](#)

[Bundesrat \(Exekutive\)](#)

[Bundesverwaltung \(Übersicht\)](#)

**BK** Bundeskanzlei

**EDA** Departement für auswärtige Angelegenheiten

**EDI** Departement des Innern

**EJPD** Justiz- und Polizeidepartement

**VBS** Departement für Verteidigung, Bevölkerungsschutz und Sport

**EFD** Finanzdepartement

**EVD** Volkswirtschaft

**UVEK** Departement für Energie und Klima

[Parlament \(Legislative\)](#)

[Bundesgericht \(Judikative\)](#)

[Kantone online](#)

[Kontaktieren Sie uns per E-Mail oder](#)

Suche

[Erweiterte Suche](#)

[Neues in unserem Informationsangebot](#)

[Neuste Pressemitteilungen](#) | [Archiv](#)

[Ausgewählte Themen](#) | [Weitere Themen](#)

[Asyl Schweiz](#)

[Aussenpolitik](#)

[Beziehungen Schweiz - Europäische Union](#)

[Bundesblatt](#)

Bundesrecht: [Systematische Rechtssammlung](#) | [Amtliche Sammlung](#)

[Bundesverfassung](#) (Inkrafttreten 1. Januar 2000)

[Politische Rechte im Bund](#)

[Rechtsprechung \(Verwaltungspraxis, VPB\)](#)

[Referendumsvorlagen](#)

[Regierungs- und Verwaltungsreform RVR](#)

[Schweiz - Suisse - Svizzera - Svizra - Switzerland](#)

[Stellenanzeiger des Bundes](#) | [Lehrstellenangebot](#)

[Umweltfragen](#)

[Vernehmlassungen](#)

[Volksabstimmungen](#)

[Volksabstimmung vom 24. September 2000](#)

[Volksabstimmung vom 26. November 2000](#)

**Link: „Volksabstimmungen“**

Document: Done

Politische Rechte im Bund - Netscape

File Edit View Go Communicator Help

Back Forward Reload Home Search Netscape Print Security Shop Stop

Bookmarks Location: <http://www.admin.ch/ch/d/pore/index.html#3> What's Related

## Abstimmungen

Für alle Änderungen der [Bundesverfassung](#) sowie für den Beitritt zu bestimmten internationalen Organisationen gilt das obligatorische Referendum: das heisst, darüber muss eine Volksabstimmung stattfinden. Zur Annahme einer solchen Vorlage müssen die Stimmberechtigten die Vorlage angenommen haben.

**Link: „Daten der eidg. Volksabstimmungen“**

Geänderte oder neue Gesetze und ähnliche Beschlüsse des Parlaments sowie bestimmte völkerrechtliche Verträge kommen nur dann zur Abstimmung, wenn dies mit dem fakultativen [Referendum](#) verlangt wird. Zur Annahme einer derartigen Vorlage genügt das Volksmehr.

- [Daten der Eidgenössischen Volksabstimmungen \(Chronologische Übersicht\)](#)
- [Ergebnisse der Vorlagen an eidgenössischen Volksabstimmungen](#)
- [Eidgenössische Volksabstimmungen \(nach Abstimmungsergebnis geordnet\)](#)

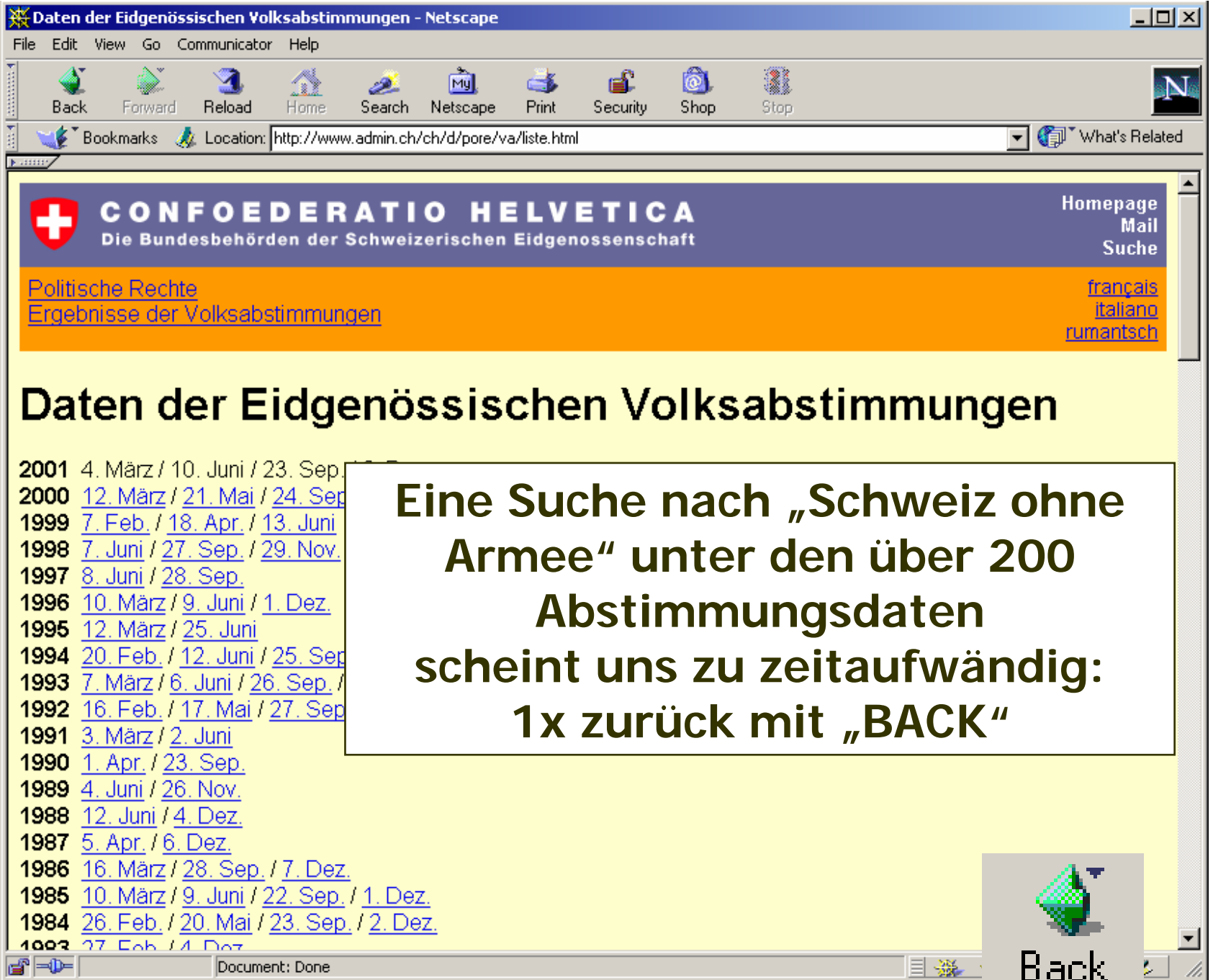
## Volksinitiative

Bürgerinnen und Bürger können einen Volksentscheid über eine von ihnen gewünschte Änderung der Verfassung verlangen. Damit eine Initiative zustande kommt, braucht es innert einer Sammelfrist von 18 Monaten die Unterschriften von 100 000 Stimmberechtigten.

Das Parlament prüft die Initiative und entscheidet, ob sie angenommen werden soll. Im Fall der Annahme wird die Initiative dem Volk zur Abstimmung vorgelegt. Im anderen Fall ist - als fertig ausgearbeitet - die Initiative zurückzuziehen. (Die Initiative kann auch zurückgezogen werden, wenn sie nicht weit gehenden Gegenstand hat.)

**Frage: Wann wurde über die Initiative „Schweiz ohne Armee“ abgestimmt?**

Document: Done



## Daten der Eidgenössischen Volksabstimmungen

- 2001 [4. März](#) / [10. Juni](#) / [23. Sep.](#)
- 2000 [12. März](#) / [21. Mai](#) / [24. Sep.](#)
- 1999 [7. Feb.](#) / [18. Apr.](#) / [13. Juni](#)
- 1998 [7. Juni](#) / [27. Sep.](#) / [29. Nov.](#)
- 1997 [8. Juni](#) / [28. Sep.](#)
- 1996 [10. März](#) / [9. Juni](#) / [1. Dez.](#)
- 1995 [12. März](#) / [25. Juni](#)
- 1994 [20. Feb.](#) / [12. Juni](#) / [25. Sep.](#)
- 1993 [7. März](#) / [6. Juni](#) / [26. Sep.](#)
- 1992 [16. Feb.](#) / [17. Mai](#) / [27. Sep.](#)
- 1991 [3. März](#) / [2. Juni](#)
- 1990 [1. Apr.](#) / [23. Sep.](#)
- 1989 [4. Juni](#) / [26. Nov.](#)
- 1988 [12. Juni](#) / [4. Dez.](#)
- 1987 [5. Apr.](#) / [6. Dez.](#)
- 1986 [16. März](#) / [28. Sep.](#) / [7. Dez.](#)
- 1985 [10. März](#) / [9. Juni](#) / [22. Sep.](#) / [1. Dez.](#)
- 1984 [26. Feb.](#) / [20. Mai](#) / [23. Sep.](#) / [2. Dez.](#)
- 1983 [27. Feb.](#) / [4. Dez.](#)

**Eine Suche nach „Schweiz ohne Armee“ unter den über 200 Abstimmungsdaten scheint uns zu zeitaufwändig: 1x zurück mit „BACK“**



Politische Rechte im Bund - Netscape

File Edit View Go Communicator Help

Back Forward Reload Home Search Netscape Print Security Shop Stop

Location: <http://www.admin.ch/ch/d/pore/index.html#3> What's Related

## Abstimmungen

Für alle obligatorischen Vorlagen, die die Vorlage Stimmern...

**Link „Daten der eidg. Volksabstimmungen“ als **besucht** markiert**

Geänderte oder neue Gesetze und ähnliche Beschlüsse des Parlaments sowie bestimmte völkerrechtliche Verträge kommen nur dann zur Abstimmung, wenn dies mit dem fakultativen [Referendum](#) verlangt wird. Zur Annahme einer derartigen Vorlage genügt das Volksmehr.

- [Daten der Eidgenössischen Volksabstimmungen \(Chronologische Übersicht\)](#)
- [Ergebnisse der Vorlagen an eidgenössischen Volksabstimmungen](#)
- [Eidgenössische Volksabstimmungen \(nach Abstimmungsresultat geordnet\)](#)

---

## Volksinitiative

Bürgerinnen und Bürgerinnen verlangen, von 100 000...

**Link: „Ergebnisse der eidg. Volksabstimmungen“**

Das Volksbegehren kann als allgemeine Anregung formuliert sein oder - was viel häufiger der Fall ist - als fertig ausgearbeiteter Text vorliegen, dessen Wortlaut Parlament und Regierung nicht mehr verändern können.

Die Behörden reagieren auf eine eingereichte Initiative manchmal mit einem (meist nicht so weit gehenden) Gegenvorschlag - in der Hoffnung, dieser werde von Volk und Ständen eher angenommen.

Document: Done



Ergebnisse der Vorlagen an eidgenössischen Volksabstimmungen - Netscape

File Edit View Go Communicator Help

Back Forward Reload Home Search Netscape Print Security Shop Stop

Bookmarks Location: <http://www.admin.ch/ch/d/pore/va/index.html> What's Related

**CONFOEDERATIO HELVETICA**  
Die Bundesbehörden der Schweizerischen Eidgenossenschaft

Homepage  
Mail  
Suche

[Politische Rechte](#)  
[Abstimmungsdaten](#)  
[Eidgenössische Volksabstimmungen \(nach Abstimmungsresultat geordnet\)](#)

[français](#)

## Ergebnisse der Vorlagen an eidgenössischen Volksabstimmungen

<a href="#">6. Juni 1848</a>	<a href="#">Totalrevision vom 12. September 1848</a>	
<a href="#">14. Jan. 1866</a>	<a href="#">Festsetzung von Mass und Gewicht</a>	
<a href="#">4. Juni 1899</a>	<a href="#">Eidgenössische Volksinitiative 'natu...' gegen Tiertabriken (Klembauern-Initiative)'</a>	
<a href="#">26. Nov. 1989</a>	<a href="#">Eidgenössische Volksinitiative 'für eine Schweiz ohne Armee und für eine umfassende Friedenspolitik'</a>	<a href="#">Die Initiative wurde verworfen</a>
<a href="#">26. Nov. 1989</a>	<a href="#">Eidgenössische Volksinitiative 'pro Tempo 130/100'</a>	<a href="#">Die Initiative wurde verworfen</a>
<a href="#">1. A...</a>		<a href="#">...ive wurde verworfen</a>
<a href="#">1. A...</a>		<a href="#">...ive wurde verworfen</a>

**Gefunden:**  
**Eidg. Volksinitiative**  
**„Schweiz ohne Armee“**  
**vom**  
**26. Nov. 1989**

**Frage:** Wann wurde über die Initiative „Schweiz ohne Armee“ abgestimmt?

Document: Done

## 2. Konzepte zu Backtracking

### Definition:

**Backtracking** ist eine

- *systematische Art der Suche* in einem
- *vorgegebenen Lösungsraum*.

Wenn eine Teillösung in eine

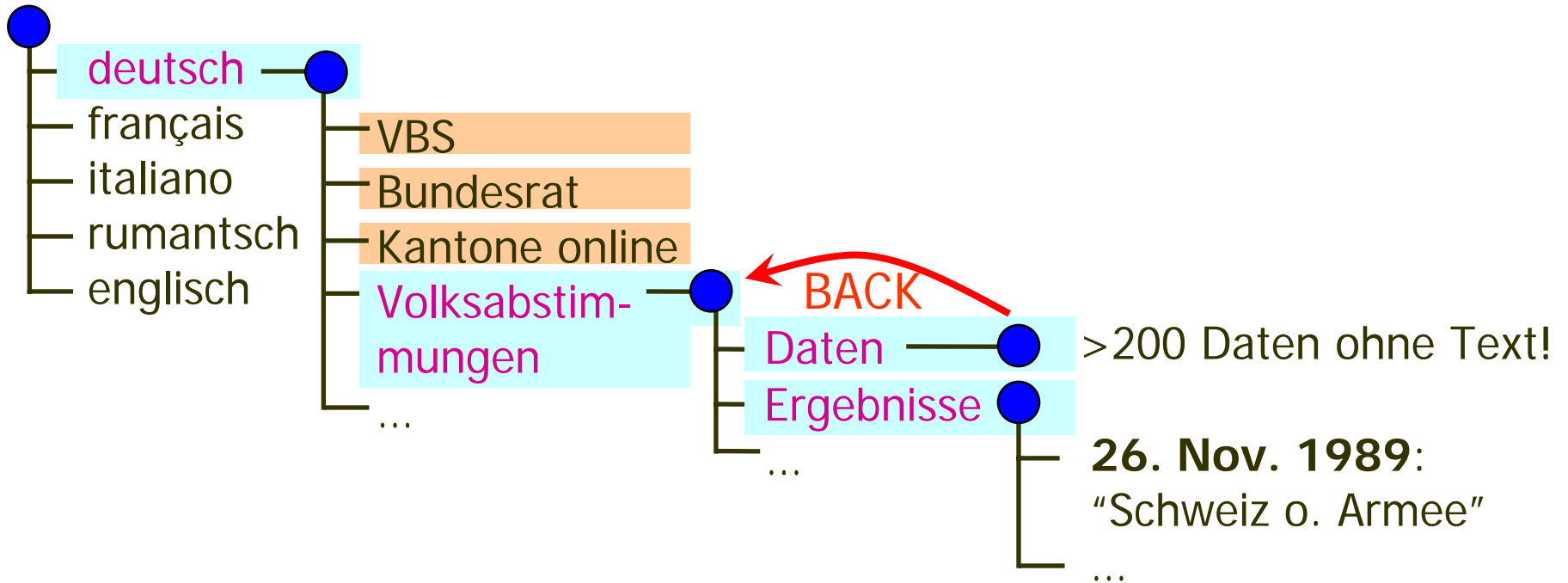
- *Sackgasse* führt, dann wird der
- *jeweils letzte Schritt rückgängig* gemacht („back-tracking“).

## 2. Konzepte zu Backtracking

# Suchweg zur Antwort auf die Frage:

Wann wurde über die Initiative „Schweiz ohne Armee“ abgestimmt?

www.admin.ch



● Webseite

Besucher Link

BACK

Back

Link, der wohl nicht zur Lösung führt

## 2. Konzepte zu Backtracking

### **Definition:**

**Heuristiken** sind „Strategien, die *mit höherer Wahrscheinlichkeit* (jedoch *ohne Garantie*) das Auffinden einer *Lösung beschleunigen* sollen.“

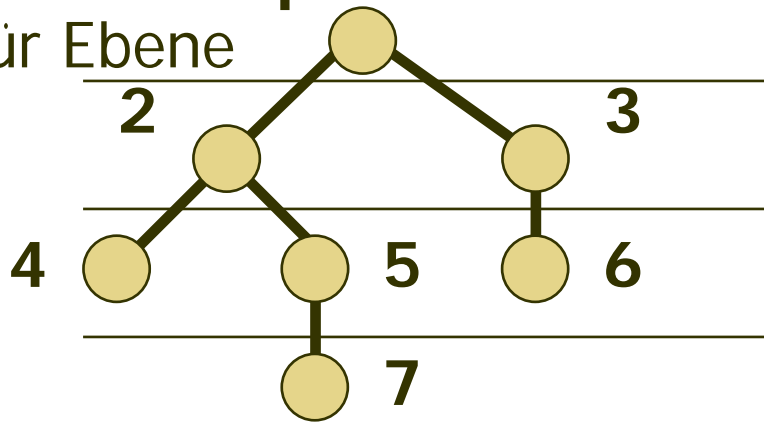
(Quelle: Schüler-Duden „Die Informatik“, S. 236, Bibliograph. Institut, Mannheim/Wien/Zürich, 1986)

## 2. Konzepte zu Backtracking

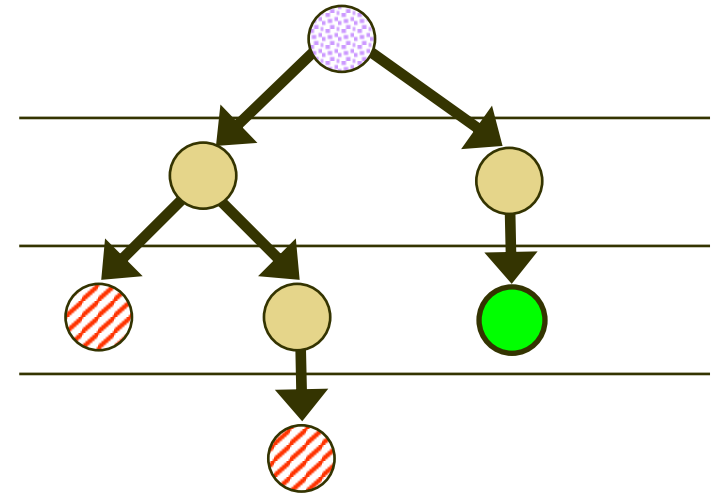
### Suche im Lösungsbaum:

#### Breitensuche: 1

Ebene für Ebene

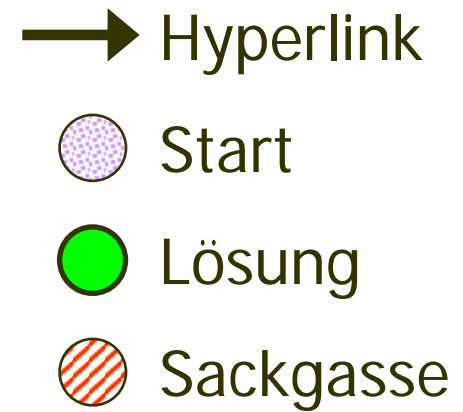
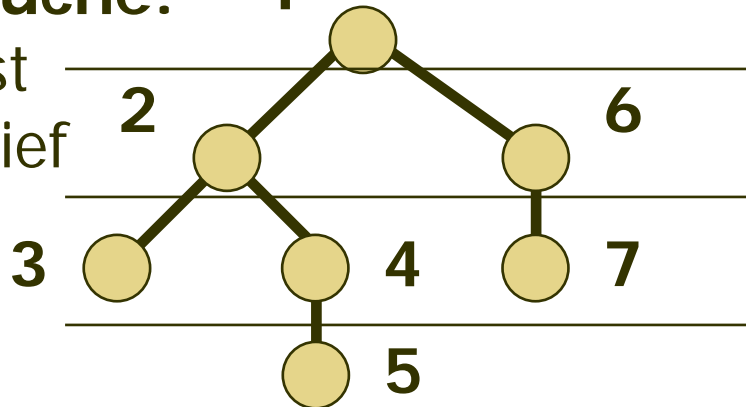


#### Suchbaum:



#### Tiefensuche: 1

Möglichst  
schnell tief  
suchen



**Backtracking ist Tiefensuche!**

## Ein allgemeiner **Backtracking Algorithmus:**

```
boolean FindeLoesung(int index, Lsg loesung, ...) {
```

```
    // index ist die aktuelle Schrittzahl
```

```
    // Teillösungen loesung werden als Referenz übergeben.
```

1. Solange es noch neue Teil-Lösungsschritte gibt:

a) Wähle einen neuen Teil-Lösungsschritt **schritt**; // **Heuristik**

b) Falls **schritt** gültig ist:

I) Erweitere **loesung** um **schritt**;

II) Falls **loesung** vollständig ist, *return true*, sonst:

```
    if (FindeLoesung(index+1,loesung)) { // rekursiv
```

```
        return true; // Lösung gefunden
```

```
    } else { // Wir sind in einer Sackgasse
```

```
        Mache schritt rückgängig; // Backtracking
```

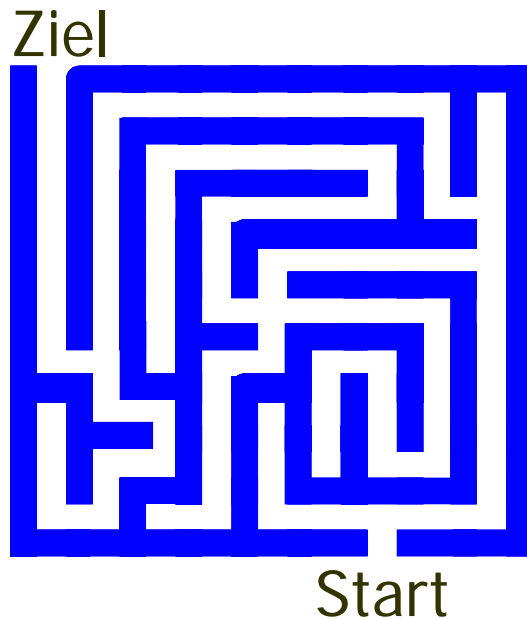
```
    }
```

2. Gibt es keine neuen Teil-Lösungsschritte mehr, so: *return false*

```
}
```

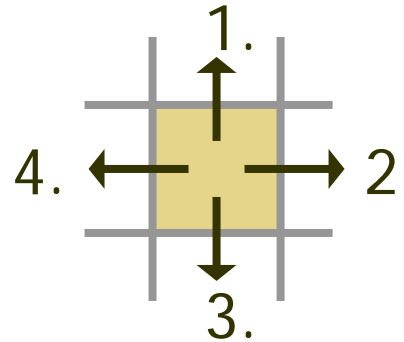
# 3. Eingesperrt im Labyrinth

## a) Der Backtracking-Ansatz



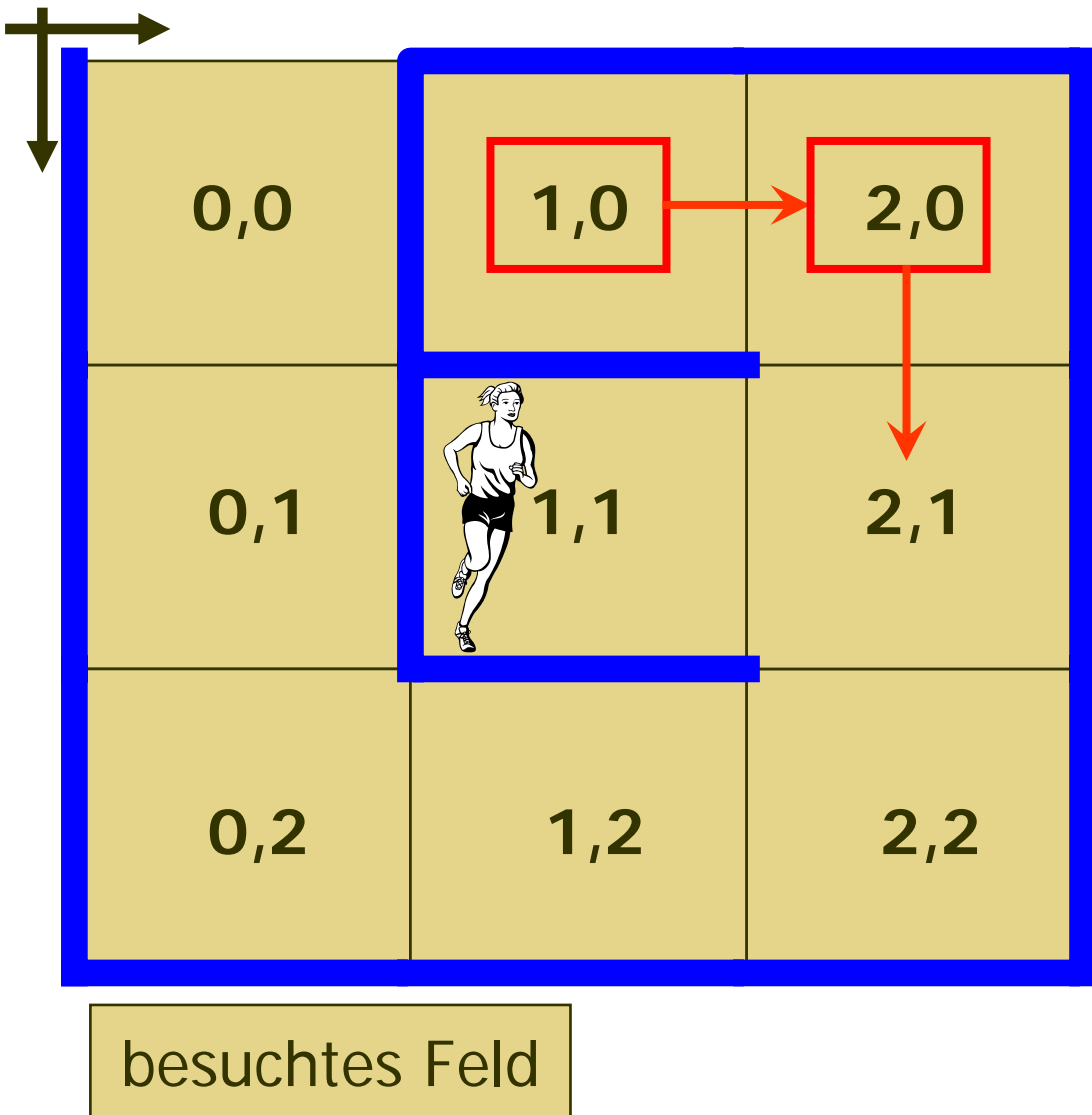
### Lösungsstrategie:

- **Systematisch** vom aktuellen Feld im Labyrinth nach 1. oben, 2. rechts, 3. unten und 4. links abzweigen

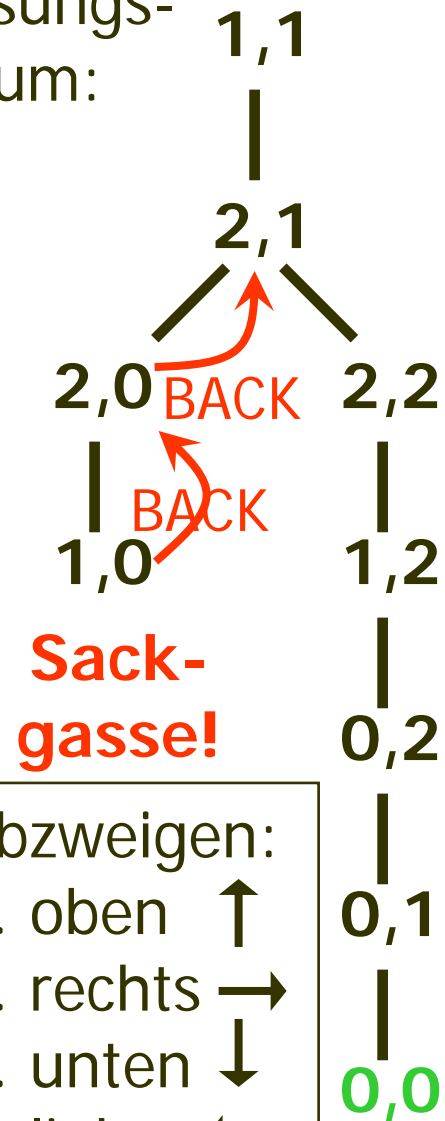


- **Besuchte** Felder **markieren**
- In **Sackgassen** die Züge zurücknehmen (**Backtracking**)

### 3. Eingesperrt im Labyrinth - a) Der Backtracking-Ansatz



Lösungsbaum:



Abzweigen:

- 1. oben ↑
- 2. rechts →
- 3. unten ↓
- 4. links ←



# Allgemeiner **Backtracking Algorithmus** (Java-Pseudocode)

```
boolean FindeLoesung(int index, Lsg loesung, ...) {  
    // index = Schrittzahl, loesung = Referenz auf Teillösung  
  
    while (es gibt noch neue Teil-Lösungsschritte) {  
        Wähle einen neuen Teil-Lösungsschritt schritt; // Heuristik  
        if (schritt ist gültig) {  
            Erweitere loesung um schritt;  
            if (loesung noch nicht vollständig) {  
                // rekursiver Aufruf von FindeLoesung  
                if (FindeLoesung(index+1,loesung,...)) {  
                    return true; // Lösung gefunden  
                } else { // wir sind in einer Sackgasse  
                    Mache schritt rückgängig; // Backtracking  
                }  
            } else return true; // Lösung gefunden -> fertig  
        }  
    }  
    } return false;  
} // Bei true als Rückgabewert steht die Lösung in loesung
```

### 3. Eingesperrt im Labyrinth

b) Der Computer zeigt den Weg

Wir schreiben eine Backtracking Prozedur

boolean **FindeLoesung**(int **index**, Lsg **loesung**, int aktX, int aktY),  
um in einem Labyrinth mit KxL Feldern vom **Start** zum **Ziel** einen Weg zu finden.

// **index** = aktuelle Schrittzahl

// **loesung** = Referenz auf Teillösung

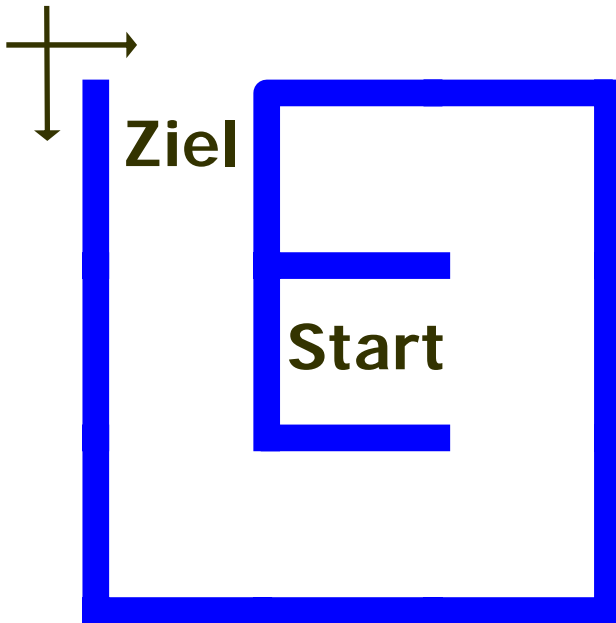
// (aktX, aktY) = aktuelle Feldposition

Der Weg vom Start zum Ziel:

- Der **Start** wird beim Aufruf als (aktX, aktY) übergeben
- Während der Suche wird bei den rekursiven Aufrufen in (aktX, aktY) das aktuelle Feld übergeben.
- Am **Ziel** liefert **ausgangGefunden**(aktX, aktY) „true“ und in **loesung** steht der gefundene Weg.

### 3. Eingesperrt im Labyrinth: b) Der Computer zeigt den Weg

Beispiel:  $K=3$ ,  $L=3$



Start=(1,1) Ziel=(0,0)

Lösungsweg in  
`loesung.feld[x][y]`:

The diagram shows a 3x3 grid with a coordinate system. The path sequence is as follows:

0,0	7	1,0	-1	2,0	-1
0,1	6	1,1	1	2,1	2
0,2	5	1,2	4	2,2	3

Weg als Besuchsreihenfolge

**Feldwerte:** -1 (Sackgasse), 0 (unbesucht), >0 (Weg)

3. **Eingesperrt im Labyrinth:** b) Der Computer zeigt den Weg

## **Vorgehen:**

1. Demonstration mit Animation des Algorithmus:

### **Backtracking im Labyrinth**

(Java Programm „Labyrinth“)

2. Besprechung der

### **Implementierung von FindeLoesung**

als Java-Pseudocode

3. Erneute Demonstration

### **Backtracking im Labyrinth**

(Java Programm „Labyrinth“)

4. Im Anschluss an den Vortrag:

Studium der Java-Implementierung Labyrinth.java

3. **Eingesperrt im Labyrinth:** b) Der Computer zeigt den Weg  
**Backtracking Algorithmus zum Labyrinth** implementiert  
in Java-Pseudocode:

```
boolean FindeLoesung(int index, Lsg loesung, int aktX, int aktY) {  
    // index           = Schrittzahl  
    // loesung        = Referenz auf Teillösung  
    // aktX, aktY      = aktuelle Feldposition im Labyrinth
```

```
    while („wir haben noch nicht alle Richtungen probiert“) {  
        Wähle als neuen Teil-Lösungsschritt schritt eine Richtung:  
        „Wir gehen nach 1. oben, 2. rechts, 3. unten und 4. links  
        und merken uns die aktuelle Richtung in schritt .“
```

```
        boolean ok := schritt ist gültig;
```

```
        „Der schritt ist gültig, wenn er
```

- innerhalb des Labyrinth-Spielfeldes bleibt,
- nicht durch eine Wand führt und
- nicht auf ein bereits besuchtes Feld führt“

3. **Eingesperret im Labyrinth:** b) Der Computer zeigt den Weg

```

if (ok) { // d.h. schritt ist gültig
    Erweitere loesung um schritt:
    „(neuX,neuY) = (aktX,aktY) + deltaXY[schritt]“
    „Markiere Feld (neuX,neuY) mit aktueller Schrittzahl“

    if (loesung noch nicht vollständig) {
        // d.h. ausgangGefunden(aktX,aktY) liefert „false“
        // rekursiver Aufruf von FindeLoesung
        if (FindeLoesung(index+1,loesung,neuX,neuY)) {
            return true; // Lösung gefunden
        } else { // wir sind in einer Sackgasse
            Mache schritt rückgängig mit Backtracking:
            „Markiere neues Feld als Sackgasse.“
        }
    } else return true; // Lösung gefunden -> fertig
}
} return false; // Ende des while(...) Blocks
} // Bei true als Rückgabewert steht die Lösung in loesung

```

# 4. Laufzeitbetrachtungen

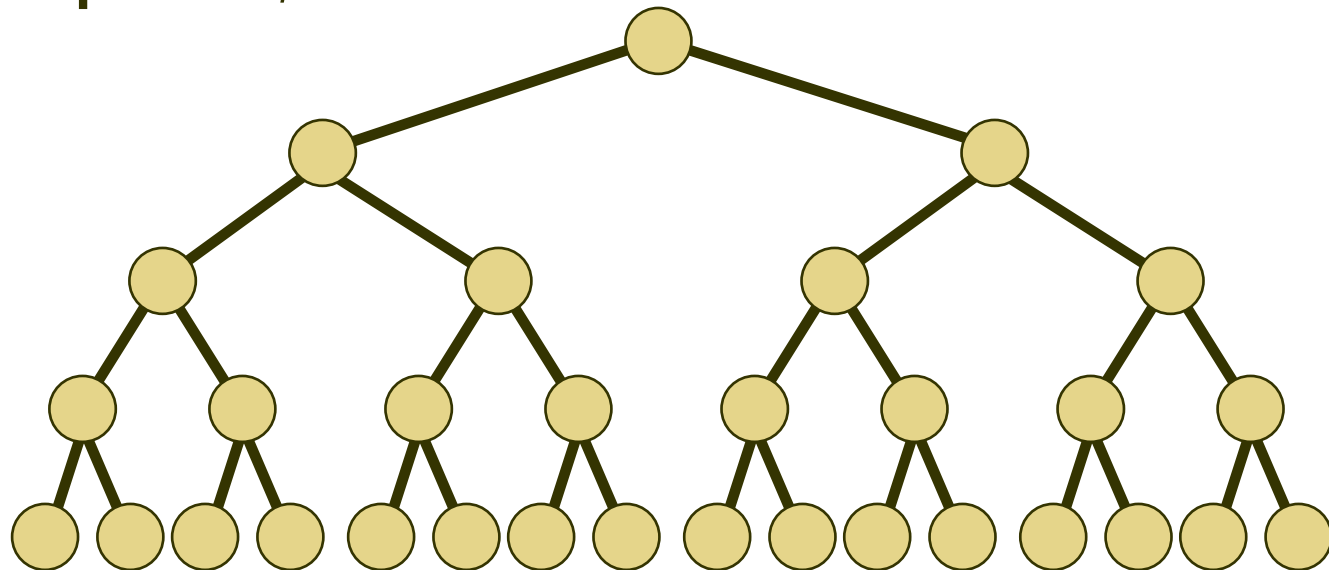
Bei der Tiefensuche werden bei

- max.  $k$  möglichen Verzweigungen von jeder Teillösung aus
  - einem Lösungsbaum mit maximaler Tiefe von  $n$
- im schlechtesten Fall

$$1 + k + k^2 + k^3 + \dots + k^n = (k^{n+1} - 1) / (k - 1) = \mathbf{O(k^n)}$$

Knoten im Lösungsbaum erweitert.

**Bsp.:**  $k=2, n=4$



Tiefe	Knoten
0	1
1	$k = 2$
2	$k^2 = 4$
3	$k^3 = 8$
$n$	$k^n = 16$

## 4. Laufzeitbetrachtungen

Die Tiefensuche und somit auch Backtracking haben im schlechtesten Fall mit  $O(k^n)$  eine **exponentielle Laufzeit**.

Bei grosser Suchtiefe  $n$  und Verzweigungsgrad  $k > 1$  dauert die Suche somit oft sehr lange.

**Beispiel:**  $k=10$ , Rechenleistung: 1000 Knoten/s = 1 ms/Knoten

Tiefe $n$	Knoten	Zeit (s)
0	1	1 ms
2	111	0.1 s
4	11'111	11 s
6	1'111'111	19 min
8	$10^8$	31 h
10	$10^{10}$	129 d
12	$10^{12}$	35 y

Gute **Heuristiken** können die Suche nach einer Lösung beschleunigen.

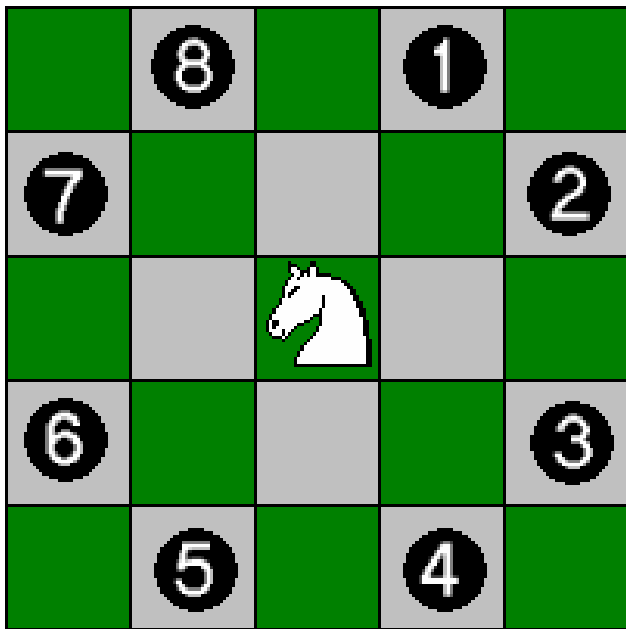


## 5. Springerwege

### Das Problem:

Ein **Springerweg** ist ein Pfad aus Springerzügen auf einem Schachbrett, der jedes Feld genau einmal besucht.

Ein **Springer** kann max. **8 mögliche Orte** anspringen.



**Zug** ist **gültig**, wenn das neue Feld

- innerhalb des Spielfeldes liegt
- und
- unbesucht ist

Wir merken uns den bisherigen Pfad in einem 2D-Array of Integers, wo wir die Schrittzahl als „**Besuchsreihenfolge**“ speichern.

## 5. Springerwege

### Backtracking Algorithmus:

- Systematisch gemäss obiger Reihenfolge springen
- Bei „Sackgassen“ Sprünge zurücknehmen (Backtracking)
- Springerweg, sobald Weglänge = Anzahl Felder

### Beispiellösung (6x6):

36	31	22	19	4	29
23	18	3	30	9	20
32	35	24	21	28	5
17	2	33	8	13	10
34	25	12	15	6	27
1	16	7	26	11	14

Die Abbildung zeigt einen Springerweg auf einem Schachbrett der Grösse 6x6.

Begonnen wurde unten links.

## 5. Springerwege

### Zeitprobleme beim Backtracking:

**Feststellung:** Bei dieser Methode kann das Finden einer Lösung schon für ein 8x8 Schachbrett **mehrere Tage** dauern!

**Grund:** Wir suchen zwar systematisch, aber wenig intelligent!

Bsp.:

	10	
	5	18
9		11
14	17	6
	12	15
16	7	18

Unser Algorithmus springt auf dem 8x8 Schachbrett von Schritt 17 aus zur roten Position 18.

-> Dadurch erzwingen wir unbewusst, dass die Ecke mit der grünen 18 als letztes Feld besucht werden muss.

**Bessere Strategie:** Immer **in die Ecke springen**, wenn möglich!

-> Schneller

## 5. Springerwege

### **Heuristik von Warnsdorf (1823):**

Es muss immer auf das Feld gesprungen werden, welches am schlechtesten erreichbar ist.

**Mass** für schlechte Erreichbarkeit eines Feldes:

Anzahl Felder, welche man von diesem in einem Sprung erreichen kann. Je weniger, um so schlechter erreichbar.

### **Wir bauen diese Regel ein:**

Anstatt in der vorgegebenen Reihenfolge zu springen, berechnen wir jeweils die Erreichbarkeit aller vom aktuellen Ort anspringbarer Felder und springen zuerst zum am schlechtest erreichbaren.

Diese Heuristik macht unser Programm „intelligenter“ und dadurch schneller.

## 5. Springerwege

**Auswirkungen** der Heuristik von Warnsdorf:

Springerweg für 8x8 in unter 1 Sekunde.

Springerweg für 50x50 in wenigen Sekunden.

Springerweg ab ca. 60x60 ziemlich langsam.

### **Grund:**

Bis 56x56 Felder brauchen wir dank der Heuristik von Warnsdorf keinen einzigen Backtracking-Schritt zu machen.

Der Aufwand zur Bestimmung der Erreichbarkeiten der Felder macht sich also mehr als bezahlt.

### **Bemerkung:**

Es gibt nicht nur einen möglichen Springerweg auf dem 8x8-Schachbrett, sondern **33'439'123'484'294**, um genau zu sein.

(Quelle: ECCC TR95-047, M. Löbbing, I. Wegener)

# 6. Zusammenfassung

## Backtracking

- ist eine **systematische** Suchstrategie und findet deshalb immer eine **optimale Lösung**, sofern vorhanden, und sucht höchstens einmal in der gleichen „Sackgasse“
- ist einfach zu implementieren mit **Rekursion**
- macht eine **Tiefensuche** im Lösungsbaum
- hat im schlechtesten Fall eine exponentielle **Laufzeit  $O(k^n)$**  und ist deswegen primär für kleine Probleme geeignet
- erlaubt Wissen über ein Problem in Form einer **Heuristik** zu nutzen, um den Suchraum einzuschränken und die Suche dadurch zu beschleunigen