10.

Backtracking

Buch Mark Weiss „Data Structures & Problem Solving Using Java“ siehe S. 333-336

Lernziele Kapitel 10 Backtracking

- Prinzip des Backtrackings verstehen
- Backtracking auf typische dafür geeignete Probleme anwenden

Thema / Inhalt

Backtracking bezeichnet eine wichtige algorithmische Problemlösungsmethode, die auf dem **systematischen Durchmustern** eines grossen Zustandsraums beruht. Dabei wird versucht, eine erreichte Teillösung schrittweise weiter zu einer Gesamtlösung auszubauen. Sobald jedoch absehbar ist, dass eine Teillösung nicht zu einer endgültigen Lösung führen kann, wird der letzte Schritt zurückgenommen und es werden (falls vorhanden) alternative Wege probiert; ansonsten muss noch weiter zurückgegangen werden. Auf diese Weise ist sichergestellt, dass alle in Frage kommenden Lösungswege ausprobiert werden können.

Man kann dies auch als eine **Depth-first-Traversierung** eines Zustandsgraphen bzw. des Entscheidungsbaums ansehen, dessen Knoten Teillösungen repräsentieren, die evtl. weiter in Richtung einer Gesamtlösung vervollständigt werden können. Ein Blatt stellt dabei entweder das Ende einer Sackgasse oder aber eine Gesamtlösung dar. Mittels Heuristiken wird versucht, frühzeitig nicht-zielführende Unterbäume zu erkennen und zu vermeiden bzw. erfolgreich aussehende Alternativen zuerst auszuprobieren. Da man oft nicht an allen Zuständen oder denkbaren Lösungen interessiert ist, sondern nur an einer bestimmten (oder ersten) Lösung, z.B. wenn man den Ausgang aus einem Labyrinth sucht, können solche Heuristiken die erfolgreiche Suche wesentlich beschleunigen.

Thema / Inhalt (2)

Als Beispiele betrachten wir das **Labyrinth-Problem** (den Goldtopf im Inneren finden – oder vielleicht auch den Minotauros, wenn man Theseus heisst – dann aber auch wieder zügig einen Ausgang suchen), **Edge-Matching-Puzzles** als frustrierende Geduldsspiele und schliesslich das prominente **8-Damen-Problem** („es handelt sich darum, auf ein Schachbrett 8 Königinnen aufzustellen, derart, dass keine irgend eine der andern zu schlagen im Stande ist").

Das 8-Damen-Problem (verallgemeinert auch das n-Damen-Problem) gehen wir in einem **Java-Programm** an. Für die Repräsentation der Spielzustände braucht man keine Nachbildung des Spielbretts (wie z.B. ein zweidimensionales Array), das würde nur unnötigen Aufwand erzeugen. Es genügt dafür eine Permutation der Zahlen 1 bis 8, die in einem eindimensionalen Array der Länge 8 gespeichert werden kann. Die i-te Stelle der Permutation symbolisiert dabei die Höhe (über der Grundlinie) der Dame der i-ten Spalte. Aus Effizienzgründen ist es zweckmässig, die jeweils bedrohten Zeilen und Diagonalen in gesonderten Booleschen Variablen nachzuhalten, deren Werte aus dem Zustand abgeleitet sind. Für den eigentlichen Backtracking-Algorithmus nutzen wir in unserem Java-Programm eine Methode, die in einer bestimmten Spalte (unter Berücksichtigung von Bedrohungen aus kleineren Spalten) eine Dame zu platzieren versucht und bei Erfolg rekursiv auf die nachfolgende Spalte angewendet wird. Eine Hilfsmethode sorgt für eine ansprechende zweidimensionale Ausgabe einer gefundenen Lösung. Optional kann ein Trace ausgegeben werden, der die einzelnen Schritte des versuchsweisen Setzens einer Dame sowie die Backtrack-Schritte visualisiert.

Backtracking als allgemeines Prinzip wurde nicht einfach irgendwann entdeckt; wiederholt ist es bei der Lösung konkreter Probleme quasi nebenbei „erfunden" worden. **Robert John Walker** hatte 1960 in seinem Aufsatz „An enumerative technique for a class of combinatorial problems" das algorithmische Backtracking-Prinzip formal beschrieben und als ein allgemeines Lösungs-

Thema / Inhalt (3)

verfahren propagiert. Allerdings wurde die Methode des Backtrackings, ohne es so zu nennen, zuvor schon angewendet, beispielsweise beim Labyrinth-Problem. **Gaston Tarry** gab 1895 dafür einen Backtrack-Algorithmus an, der eine Verallgemeinerung des Depth-First-Prinzips darstellt.

Als erster ist aber wohl C.F. Gauß zu nennen. Er beschrieb 1850 das Backtracking-Prinzip am Beispiel des damals noch neuen 8-Damen-Problems; wir zitieren dazu Ausschnitte aus seinen Briefen. Er sprach von „Tatonnements" und nutzte dafür ein „schicklich präparirtes Quadratnetz". Am liebsten hätte er (und andere Mathematiker) wohl eine Art Formel (analog zu seiner Osterformel) gehabt, die das Problem löst. Oder wenn schon Algorithmus, dann eher so ein Straight-Forward-Verfahren wie beim Sieb des Eratosthenes zur Primzahlbestimmung, das ohne Versuch und Irrtum auskommt. Aber im Laufe der folgenden Jahrzehnte dämmerte den Mathematikern, dass man es möglicherweise mit einer unangenehmen Situation zu tun habe: Vielleicht muss man bei solcherart Problemen einfach im wesentlichen alles durchprobieren (und das dauert dann seine Zeit!). Anfangs wurde der Verdacht noch etwas zaghaft geäussert („die Aufgabe scheint nicht ganz ohne Probiren löslich zu sein"), der Logiker **Kurt Gödel** formuliert Mitte der 1950er-Jahre in einem Brief an John von Neumann dies in Verallgemeinerung als konkrete Frage („wie stark im allgemeinen bei finiten kombinatorischen Problemen die Anzahl der Schritte gegenüber dem blossen Probieren verringert werden kann"). Dies wurde später als „**P-NP-Problem**" formaler ausgedrückt und führte in den 1960er-Jahren zur Begründung der **Komplexitätstheorie**. Die starke Vermutung heute: Es geht tatsächlich nicht prinzipiell besser als durch systematisches Probieren (mit exponentiellem Aufwand). Aber bewiesen ist es noch nicht.

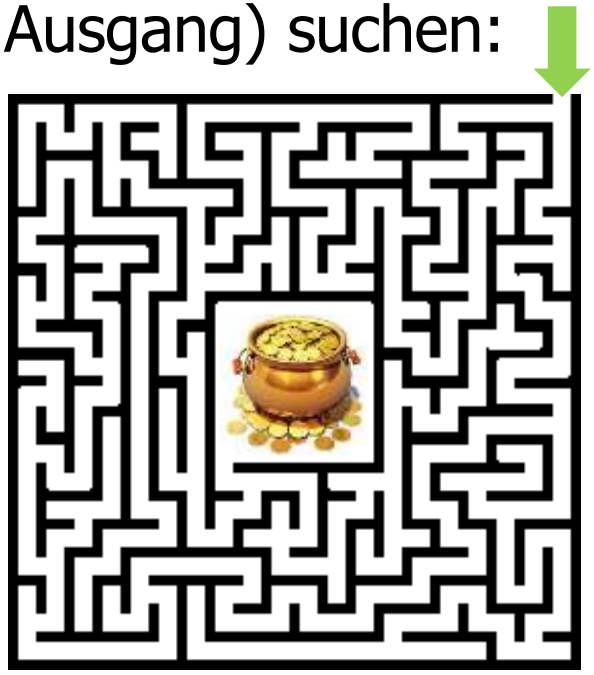
Backtracking [„Rücksetzverfahren”]

- Systematisches Durchmustern eines grossen Zustandsraumes, um zu einem (kombinatorischen) Problem eine Lösung zu finden
- Systematisches „trial and error”
- Beispiel: Mittels systematischer Tiefensuche („depth first“) in einem Labyrinth den Goldtopf (oder den Ausgang) suchen:
  - Weggabelung: eine (neue) Richtung wählen
  - In diese Richtung weitergehen
  - Wenn „letztendlich“ erfolglos: Einen Schritt zurückgehen und andere Richtung wählen
  - Idee: Um Mehrfachbesuche zu vermeiden, bereits ausprobierter Richtungen markieren

passende, gute, optimale; alle Lösungen...

Backtracking

Falls bereits alle Richtungen ausprobiert → noch weiter zurück



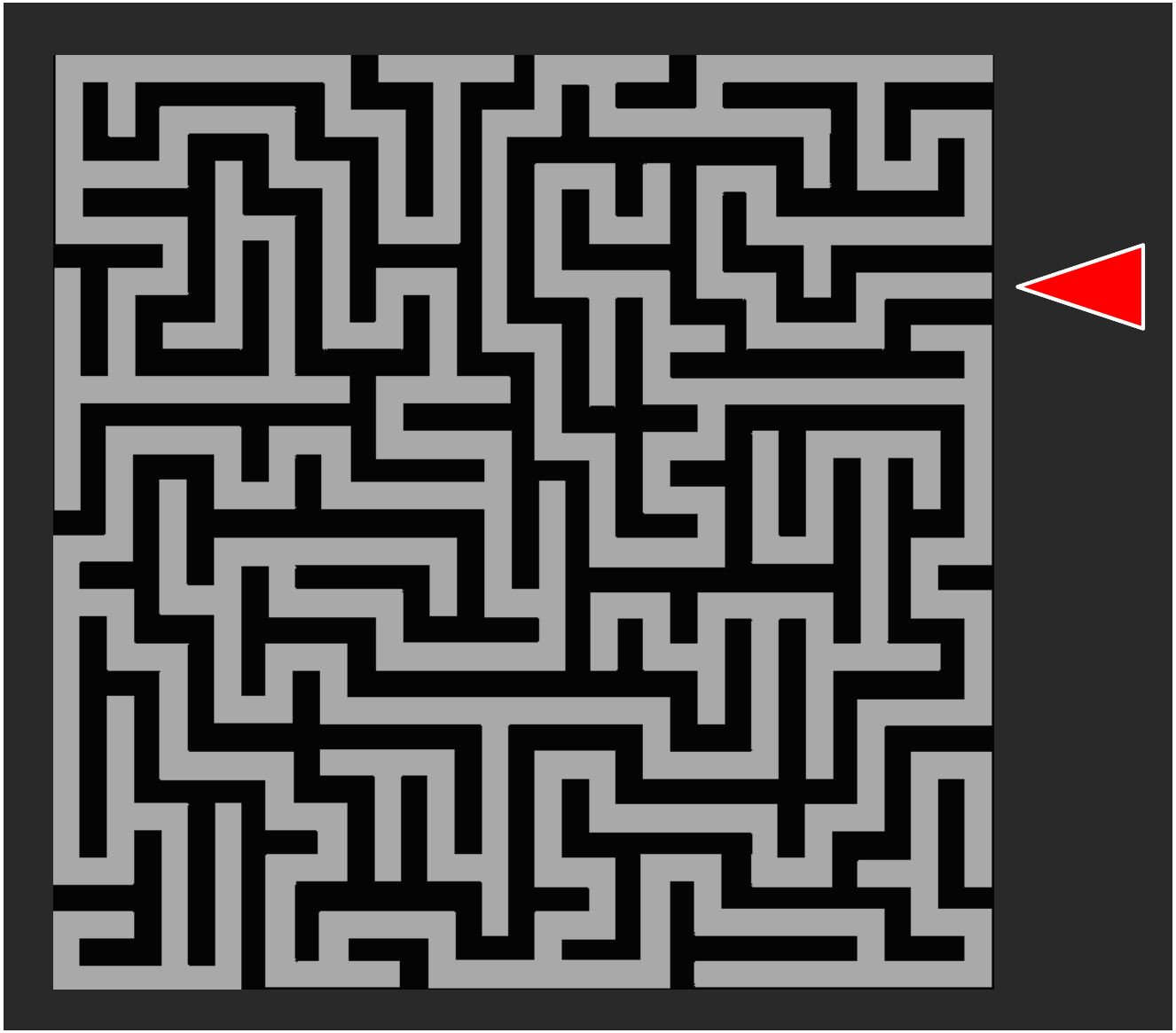
Tiefensuche mit Backtracking (Animation)

Tiefensuche rekursiv: Führe nacheinander für jeden benachbarten Knoten, falls man ihn noch nie gesehen hat, eine Tiefensuche durch.

Exploration in die Tiefe: **weiss**

Backtracking aus Sackgassen: **blau**

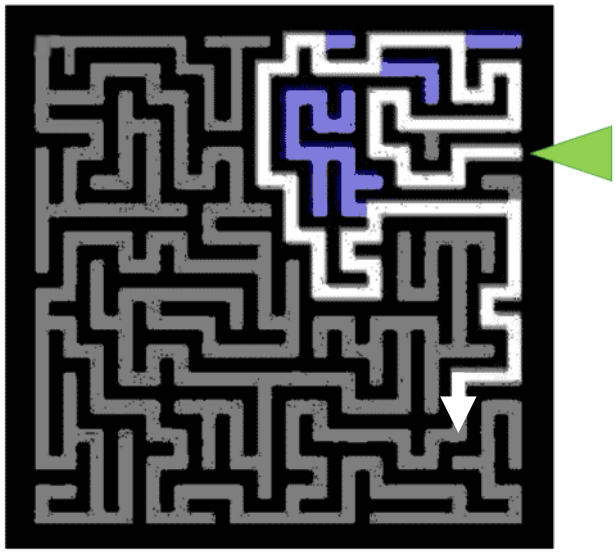
Alles scheint eine einzige rekursive Sackgasse zu sein!



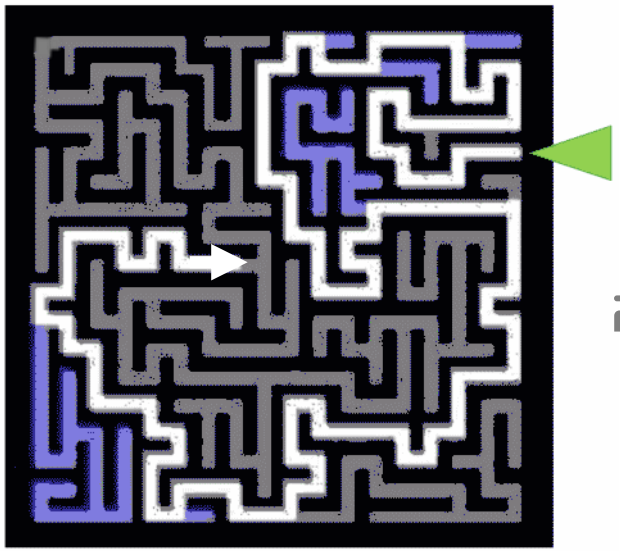
Tiefensuche mit Backtracking: 4 Schnappschüsse

Animation siehe https://en.wikipedia.org/wiki/File:Depth-First_Search_Animation.ogv

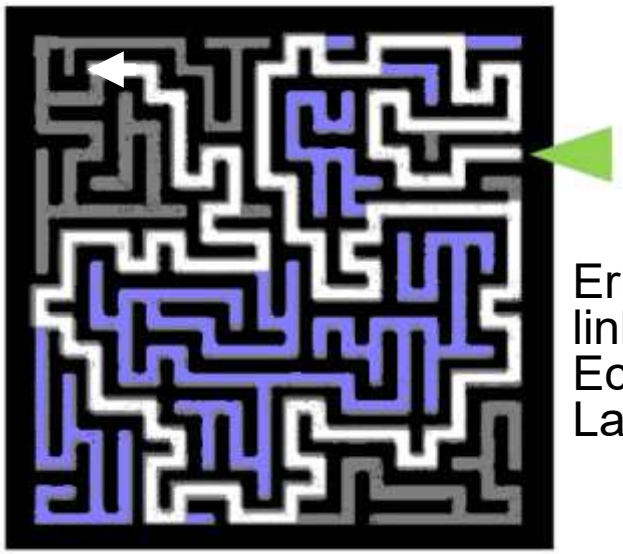
1)
05:29



2)
09:77

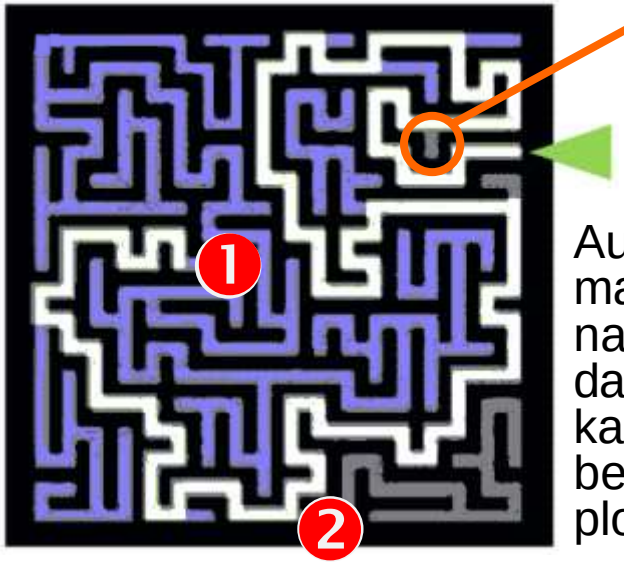


3)
18:43



Erkunden der linken oberen Ecke des Labyrinths

4)
24:08



Allerletzte zu explorierende Sackgasse

Auf dem Rückmarsch von (1) nach (2), um das noch unbekannte Gebiet bei (2) zu explorieren

Tiefensuche im Labyrinth

Das eben geschieht den Menschen, die in einem Irrgarten hastig werden: Eben die Eile führt immer tiefer in die Irre.
-- Seneca

Denkübung: Unser mit Backtracking explorier-tes Labyrinth hatte eine baumartige Struktur. Was wäre zu tun, wenn Zyklen enthalten sind?

http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/39375521.jpg

„Die Tiefensuche geht immer tiefer ins Laby-rinth, bis sie in einen Saal kommt, der keine abgehenden Gänge hat oder dessen sämtliche Gänge auf Knoten füh-ren, die sie schon be-sucht hat. Dann geht sie den Gang, den sie gerade gekommen ist, bis zum vorigen Knoten zurück und weiss, dass sie alles erledigt hat, was dahinterliegt. Klar, dass sie so durch jeden Gang höchstens einmal hin und einmal wieder zurückgehen muss.“ [Sebastian Stiller]

Das Labyrinth von Versailles

A Der Eingang des Irrgartens.
B Die Abbildung Æsopi.
C Die Abbildung der Liebe.
1 Die Eule und die Vögel.
2 Die Hahnen und das Feldhuhn.
3 Der Hahn und der Fuchs.
4 Der Hahn und Demärstein.
5 Die hangende Katz und Ratten.
6 Der Adler und Fuchs.
7 Die Pfauen und Krähe.
8 Der Hahn und Kalkunische Hahn.
9 Der Pfau und die Alster.
10 Der Drach und die Feile.
11 Der Affe und seine Jungen.
12 Streit der Thiere.
13 Der Fuchs und Kranichvogel
14 Der Kranichvogel und Fuchs
15 Die Henne mit ihre Kichen.
16 Der Pfau und Nachtigal.
17 Der Papegay und Affe.
18 Der Affe Richter.
19 Die Rattemauß und Frosch.
20 Der Hase und Schildkröthe.
21 Der Wolf und Kranichvogel.
22 Der Stoßgeyer und Vögel.
23 Der Affe König.
24 Der Fuchs und Beck.
25 Kaht der Rattenmäuse.
26 Die Frösche und Jupiter.
27 Der Affe und die Katz.
28 Der Fuchs und die Trauben.
29 Der Adler / das Conijn und Pferdwurm.
30 Der Wolf und das Stachelschwein.
31 Die Schlange mit vielen Köpfen.
32 Die Mauß / die Katz / und der kleine Hahn.
33 Der Stoßgeyer und die Tauben.
34 Der Delphin und Affe.
35 Der Fuchs und Rabe.
36 Der Schwan / und der Kranichvogel.
37 Der Wolff und das Haupt.
38 Die Schlange und das Stachelschwein.
39 Die Enten / und das junge Wasserhündlein.

Echelle de 40 toises

A Esope
B la prudance sous la figure dun Ange
C le Duc et les Oiseaux
1 les Cogs et la Perdrix
2 le Coq et le Renard
3 le Coq et le Diamant
4 le Chat pendu et les Rats
5 les Paons et le Geay
6 le Coq et le Coq dinde
7 le Paon et la Pie
8 le Serpent et la Lime
9 le Singe et ses Petits
10 le Combat des Animaux
11 l'Aigle et le Renard
12 le Renard et la Cigogne
13 la Cigogne et le Renard
14 la Poule et les Poussins
15 le Paon et le Rossignol
16 le Perroquet et le Singe
17 le Singe juge
18 le Rat et la Grenouille
19 le Lievre et la Tortue
20 le Loup et la Grue
21 le Milan et les Oiseaux
22 le Singe Roy
23 le Renard et le Bouc
24 le Conseil des Rats
25 les Grenouilles et la Cigogne
26 le Renard et les Raisins
27 le Singe et le Chat
28 l'Aigle, le Lapin, et le Scarbot
29 le Loup et le Porc-épic
30 le Serpent à plusieurs testes
31 la Souris, le Chat et le petit Coq
32 le Milan et les Colombes
33 le Dauphin et le Singe
34 le Renard et le Corbeau
35 le Cigne et la Grue
36 le Loup et la Teste
37 le Serpent et le Porc-epic
38 les Cannes et le Barbet
K Est un Cabinet couvert dont le dessein du Plafond fait plaisir a voir

Se vend à Paris chez Demortain sur le Pont Nôtre Dame à l'enseigne des belles Estampes.

16

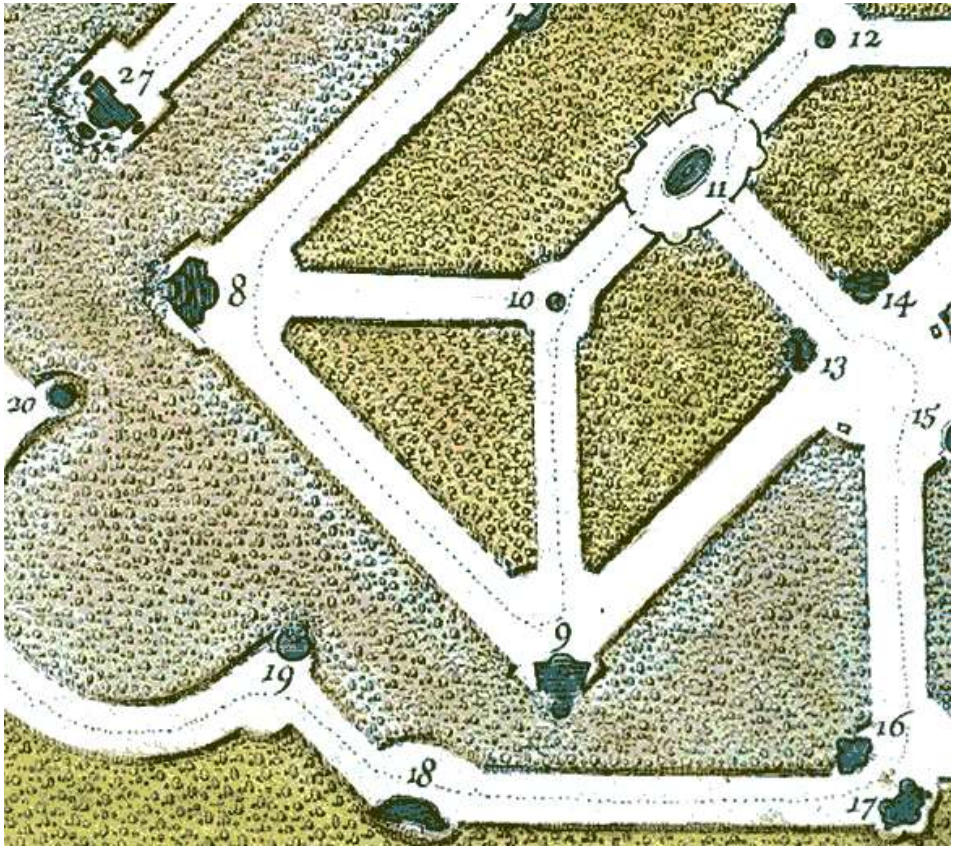
Textabbildungen aus dem zeitgenöss. Labyrinthführer; kolorierter Plan nach Gilles Demortain als Separatdruck

Das Labyrinth von Versailles (2)

Das berühmte Labyrinth im Schlosspark von Versailles gibt es heute nicht mehr. Nachdem es entsprechend den Plänen von André Le Nôtre 1680 fertiggestellt wurde, existierte es über 100 Jahre, dann wurden die Unterhaltsarbeiten auch für den französischen König Ludwig XVI zu teuer, ausserdem änderte sich die Mode: Lustwandeln unter exotischen Bäumen war dann mehr angesagt als Versteckspiel im Wäldchen des Labyrinths, wo man sich an den in Versform angebrachten Inschriften zu den Fabeln des Äsop delektieren sollte.

Das Besondere am Versailler Labyrinth war, dass an jeder Weggabelung ein Springbrunnen aufgestellt war (39 insgesamt), an dem farbig bemalte Bleigussfiguren jeweils ein Motiv aus den Tierfabeln Äsops (nach Jean de La Fontaine) illustrierten.

Es handelte sich nicht um einen üblichen Irrgarten, bei dem es einen Zielplatz zu finden galt, um dann möglichst schnell den Ausgang zu erreichen, vielmehr bestand die Aufgabe darin, den Weg so zu wählen, dass alle Brunnen genau einmal erreicht wurden. In aller Regel sieht man von einem Brunnen (mindestens) einen anderen.

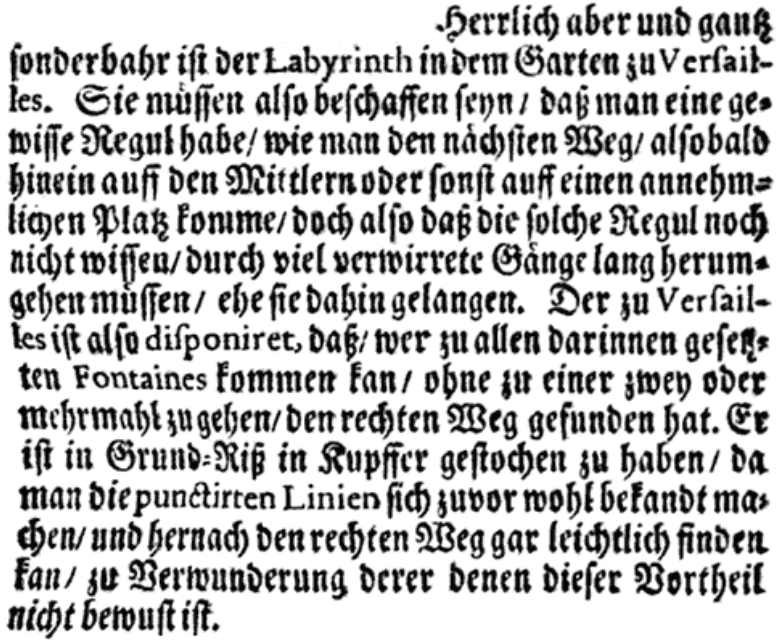


Detail aus dem Plan: *« Le filet ponctué dans les allées, marque la route pour parcourir les fontaines sans s'égarer ni repasser par les mêmes endroits. »*

Das Labyrinth von Versailles (3)

In gewisser Weise war also eine Art optimale Graphtraversierung gesucht. Obwohl das Labyrinth nicht allgemein zugänglich war, verkauften Buchhändler auf den Seine-Brücken bei der Notre-Dame in Paris Pläne mit der Lösung („que tiennent les compagnies depuis 1 jusqu'à 39").

Der deutscher Architekturtheoretiker, Baumeister, Theologe und Professor für Mathematik Leonhard Christoph Sturm (1669 – 1719) schreibt 1706 dazu in einem seiner Bücher mit dem gefälligen Titel *Die zum Vergnügen der Reisenden Geöffnete Baumeister-Academie, Oder Kurtzer Entwurf derjenigen Dinge, die einem galant-homme zu wissen nöhtig sind, dafern er Gebäude mit Nutzen besehen, und vernünfftig davon urtheilen will: Alles nach denen besten Reguln, und heut zu Tage üblichen Manieren der geschicktesten Baumeister, jedoch in möglichster Kürtze vorgestellet, und mit nöthigen Figuren erläutert*:

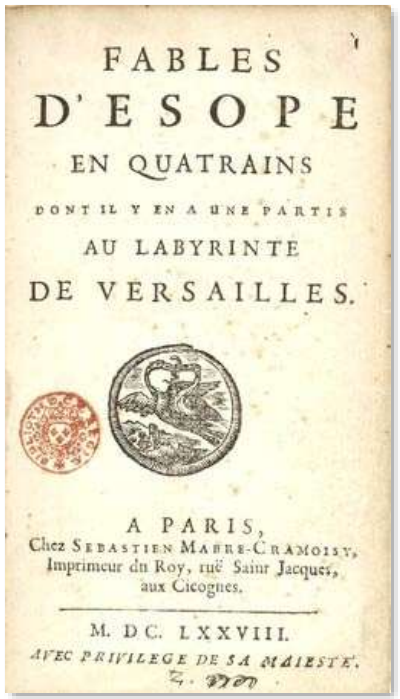


Herrlich aber und gantz
ſonderbahr iſt der Labyrinth in dem Garten zu Verſail-
les. Sie müſſen alſo beſchaffen ſeyn / daß man eine ge-
wiſſe Regul habe / wie man den nächſten Weg / alſobald
hinein auff den Mittlern oder ſonſt auff einen annehm-
lichen Platz komme / doch alſo daß die ſolche Regul noch
nicht wiſſen / durch viel verwirrete Gänge lang herum-
gehen müſſen / ehe ſie dahin gelangen. Der zu Verſail-
les iſt alſo diſponiret, daß / wer zu allen darinnen geſetz-
ten Fontaines kommen kan / ohne zu einer zwey oder
mehrmahl zu gehen / den rechten Weg gefunden hat. Er
iſt in Grund-Riß in Kupffer geſtochen zu haben / da
man die punctirten Linien ſich zuvor wohl bekandt ma-
chen / und hernach den rechten Weg gar leichtlich finden
kan / zu Verwunderung derer denen dieſer Vortheil
nicht bewuſt iſt.

Herrlich aber und ganz sonderbahr ist der *Labyrinth* in dem Garten zu *Versailles*. Sie müssen also beschaffen seyn / daß man eine gewisse Regul habe / wie man den nächsten Weg / alsobald hinein auff den Mittlern oder sonst auff einen annehmlichen Platz komme / doch also daß die solche Regul noch nicht wissen / durch viel verwirrete Gänge lang herumgehen müssen / ehe sie dahin gelangen. Der zu *Versailles* ist also *disponiret* daß / wer zu allen darinnen gesetzten *Fontaines* kommen kan / ohne zu einer zwey oder mehrmahl zu gehen / den rechten Weg gefunden hat. Er ist in Grund-Riß in Kupffer gestochen zu haben / da man die *punctirten Linien* sich zuvor wohl bekandt machen / und hernach den rechten Weg gar leichtlich finden kan / zu Verwunderung derer denen dieser Vortheil nicht bewust ist.

Das Labyrinth von Versailles (4)

Den Eingang des Labyrinths zierten zwei besondere Figuren: Der Dichter Äsop steht rechts, der Liebesgott Cupido links. Letzterer hält einen Knäuel Garn in einer Hand, aus dem er den „Ariadnefaden" herauszieht. Diese Szene findet sich auch auf dem Bild von Jacques Bailly (ca. 1675), welches den mehrsprachigen (französisch, deutsch, englisch, niederländisch) Labyrinthführer in Buchform illustriert.


FABLES
D'ESOPE
EN QUATRAINS
DONT IL Y EN A UNE PARTIE
AU LABYRINTE
DE VERSAILLES.

A PARIS,
Chez SEBASTIEN MABRE-CRAMOISY,
Imprimeur du Roy, rue Saint Jacques,
aux Cicognes.

M. DC. LXXVIII.
AVEC PRIVILEGE DE SA MAJESTE'.

Die Inschrift bei Cupido lautet:

Oui, je peux désormais fermer les yeux et rire;
Avec ce peloton, je saurai me conduire.

Äsops Antwort ist eher warnend:

Amour, ce faible fil pourrait bien t'égarer;
Au moindre choc, il peut casser.

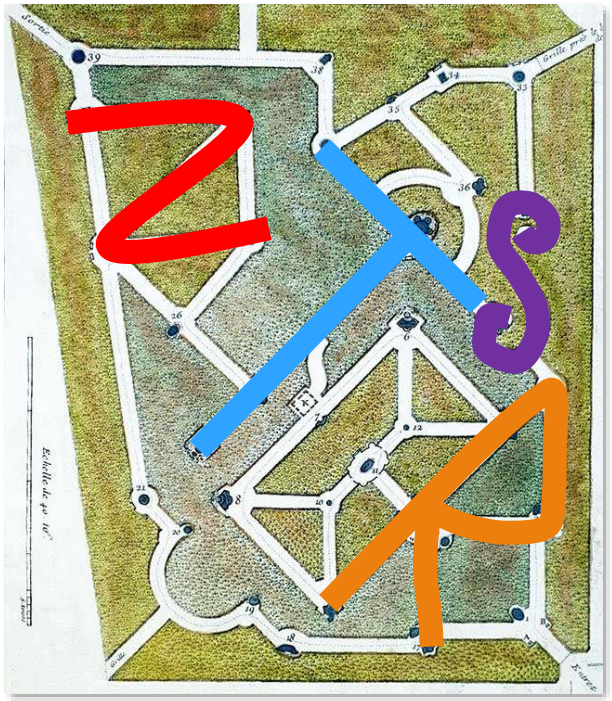
Das Labyrinth von Versailles (5)

Cupido (alias Amor) mit Ariadnefaden – passt das zusammen? Es geht wohl auch um die Verliebten, die im Labyrinth der Liebe gefangen, verirrt und verwirrt sind. Normalerweise hilft in diesem Zustand nichts, keine anderen Götter können demjenigen helfen, den Amors Pfeil trifft: *Omnia vincit Amor*. Nun also der Ariadnefaden: Die Liebe macht zwar blind, aber mit ihm kann Amor seine Augen mit gutem Gewissen schliessen; diesmal weiss er sich zu benehmen – „conduire" auf französisch, wo das Lateinische „ducere" drinsteckt: hinführen, hinausziehen etc. Allerdings ist so ein Faden doch dünn und rissig… Aber wie dem auch sei: Der Ariadnefaden steht für die Klugheit; und solange Amor bzw. die Liebe davon begleitet wird, ist man nicht auf ewig verloren… Vermutlich ist dies die Moral der Geschichte!

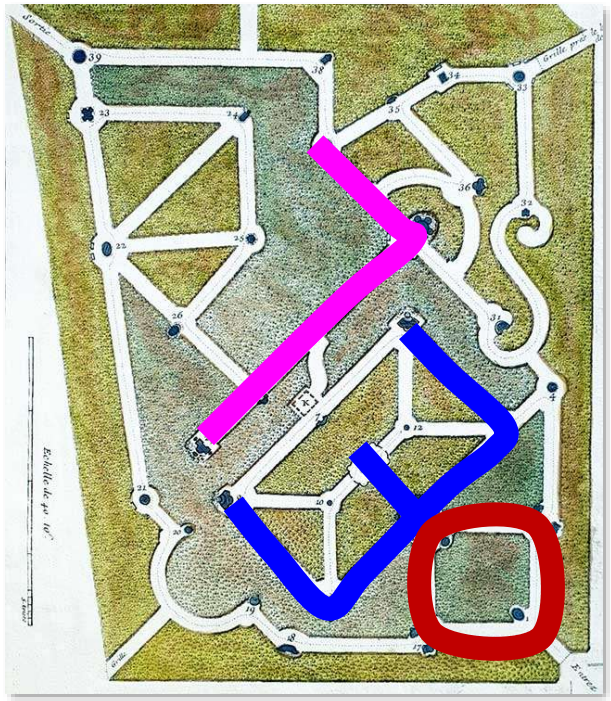
Das Labyrinth von Versailles (6)

Labyrinthe sind traditionell anfällig für Verschwörungstheorien. Beim Versailler Labyrinth fiel auf, dass es im Vergleich zu anderen Labyrinthen relativ viele gerade Wegstrecken gibt, aber auch ein paar wenige sonstige ungewöhnliche Formen. Es tauchte der Verdacht auf, dass der Gartenarchitekt Le Nôtre („NOSTRE") darin heimlich seinen Namen versteckt hatte. Tatsächlich kann man die Buchstaben finden, wenn man will. Wahrheit oder Einbildung / Überinterpretation? Er wäre jedenfalls nicht der einzige: Der deutsche Landschaftsarchitekt Hermann Mattern (1902 – 1971, u.a. Professor für Landschaftsbau und Gartenkunst an der TU Berlin) beispielsweise modellierte durch Geländeaufschüttungen oft seine Initiale „M" in den Boden, was jedoch meist nur aus der Luft zu erkennen ist. Dies gilt heute als „besonderes Gestaltungselement" und ist daher als Denkmal geschützt, was gelegentlich für Ärger mit Bauunternehmern sorgt…

Übungsaufgabe: Man suche weitere „easter eggs" in Le Nôtres Labyrint!



N, T, S, R



O, E, L

Wie man sich aus einem Labyrinthe herausfindet

Wie findet man aus einem Labyrinth wieder heraus? Im 19. Jahrhundert befassten sich einige Mathematiker – eher spielerisch und nebenbei – mit diesem Problem. Hier Auszüge aus einer Veröffentlichung mit einem Verfahren, das 1873 der Geheime Hofrat und Professor Christian Wiener (1826 – 1896) von der Grossherzoglichen Technischen Hochschule zu Karlsruhe, dem heutigen KIT, veröffentlichte [*Ueber eine Aufgabe aus der Geometria situs*, Math. Ann. 6(1), 29–30]:

Die Aufgabe, welche im Folgenden gelöst werden soll, und welche, wie ich glaube, bisher nicht behandelt ist, besteht darin, ein Verfahren anzugeben, *wie man sich aus einem Labyrinthe herausfindet*.

...

Daher die Regel: *Man wähle beim Eintritt in das Labyrinth eine jener beiden nach aussen führenden Randlinien des Weges und verfolge sie nach innen; dieselbe muss auch wieder nach aussen führen.*

In dem Falle, dass man sich im Innern des Labyrinthes befindet, ohne eine Randlinie von aussen verfolgt zu haben, kann man sich ebenfalls wieder herausfinden unter der Voraussetzung, dass man den Weg, den man zurücklegt nebst dem Sinne, in welchem man ihn beschreibt, im Gedächtniss zu behalten oder mit Marken zu bezeichnen vermag. Man verfolgt dabei zweckmässig die Axe des Weges statt seiner Randlinie. Jene Möglichkeit beruht auf der Wahrheit, dass so lange man den Ausgang noch nicht erreicht hat, ein bereits durchlaufenes Stück der Wegeaxe nothwendig von noch nicht beschriebenen Theilen derselben getroffen werden muss, weil sonst jenes Stück in

Wie man sich aus einem Labyrinthe herausfindet

sich abgeschlossen wäre und mit dem Eingangswege nicht zusammenhinge. Man markire sich daher den Weg, den man zurücklegt nebst dem Sinne, in welchem es geschieht.

Sobald man auf einen schon markirten Weg stösst, kehre man um und durchschreite den schon beschriebenen Weg in umgekehrtem Sinne. Da man, wenn man nicht ablenkte, denselben hierbei in seiner ganzen Ausdehnung nochmals zurücklegen würde, so muss man nothwendig hierbei auf einen noch nicht markirten einmündenden Weg treffen, den man dann verfolge, bis man wieder auf einen markirten trifft. Hier kehre man wieder um und verfahre wie vorher. Es werden dadurch stets neue Wegtheile zu den beschriebenen zugefügt, so dass man nach einer endlichen Zeit das ganze Labyrinth durchwandern würde und so jedenfalls den Ausgang fände, wenn er nicht schon vorher erreicht worden wäre.

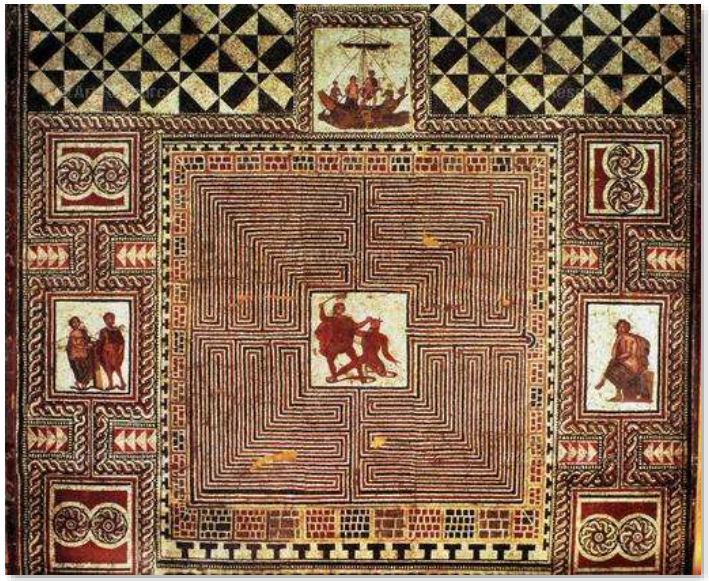
Labyrinth vs. Irrgarten: Im engeren (bzw. ursprünglichen) Sinn wird unter „Labyrinth" ein verschlungener, eng geführter und vor allem verzweigungsfreier Weg verstanden, der in vielen Windungen zwangsläufig zum Ziel führt. Im allgemeinen Sinn versteht man darunter aber ein System mit Verzweigungen, das auch Sackgassen und Zyklen enthält, bei dem man also „in die Irre" gehen kann – daher auch die Bezeichnung „Irrgarten" (engl.: „maze") für entsprechend angelegte Gärten mit meist überkopfhohen und blickdichten Hecken.

Theseus im Labyrinth

And with his finger following her thread – He issued forth to see the heavens once more. (Edward Robeson Taylor)

http://4.bp.blogspot.com/-ipTHJ1bHC_k/UnpDNN8LMCI/AAAAAAAAMec/y41R_xe4vNY/s1600/theseus.jpg

Edward Burne-Jones, *Theseus in the labyrinth*, 1861



Theseus-Mosaik der Römischen Villa in Loig (bei Salzburg), 3. Jh. n. Chr., 6.36m x 5.50m. Schwarze Linien symbolisieren Wände, die rote Linie den Ariadne-Faden. Links übergibt Ariadne ihrem geliebten Theseus den Faden, im Mittelbild besiegt dieser das Ungeheuer, oben das Fluchtschiff der beiden. Rechts trauert Ariadne, nachdem Theseus sie auf der Insel Naxos zurücklies; ihr spätes Glück fand sie dann dort mit Dionysos, dem Gott des Weines.

Exkurs: Theseus, Ariadne und das traurige Schicksal des Hofingenieurs Daidalos

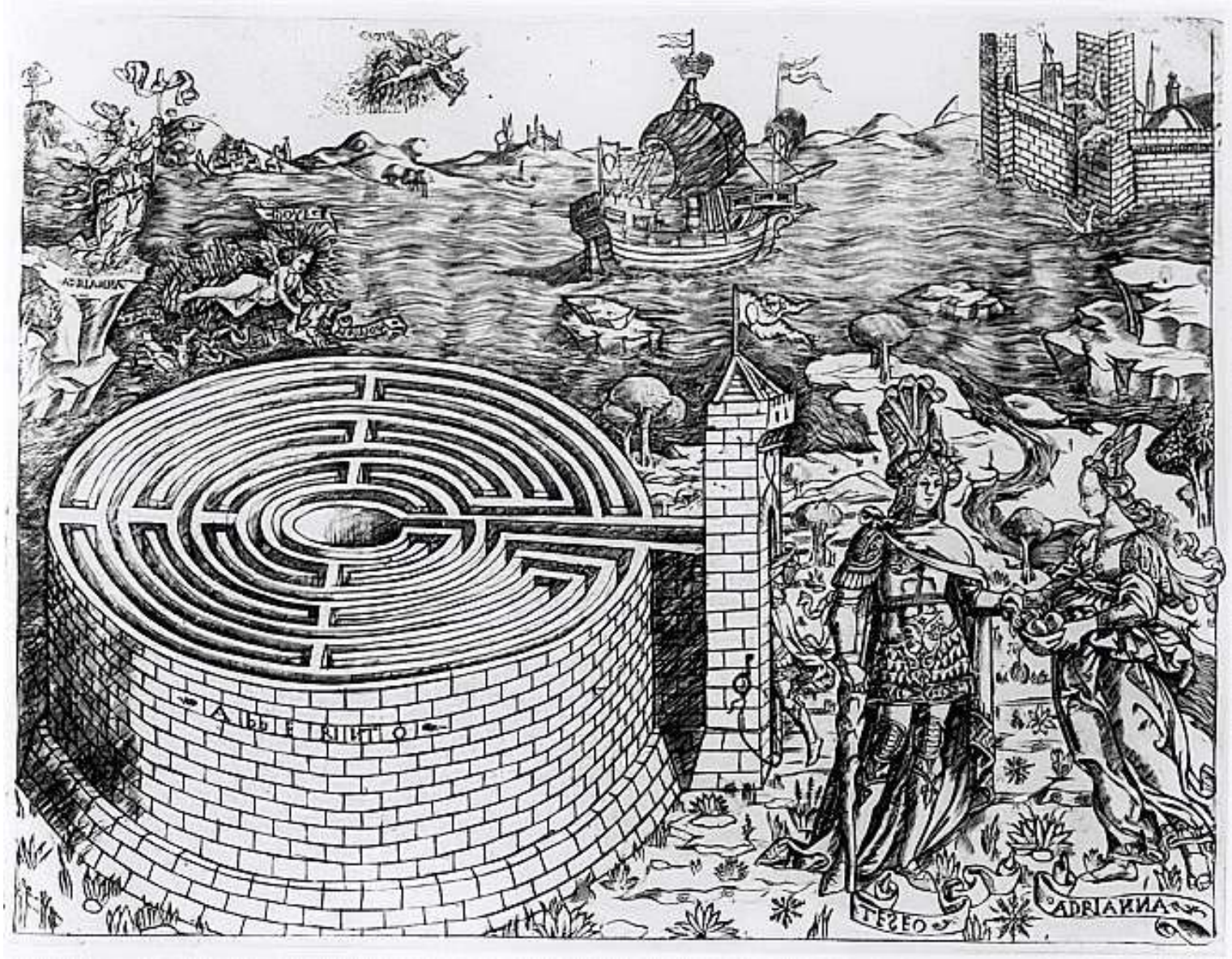
Der österreichische Informatikpionier Heinz Zemanek erzählt die Geschichte dazu:*)

...sei mit einem Einschub über den antiken Hintergrund (der Theseus-Sage) bedacht, der den Leser an seine frühen Kenntnisse griechischer Mythologie erinnern wird.

Zur Zeit des Königs Aigeus war Athen der Insel Kreta tributpflichtig. Alljährlich mussten sieben Jünglinge und sieben Jungfrauen von Athen nach Kreta versandt werden, um dem Minotauros zum Frasse vorgeworfen zu werden. Dieser Minotauros, ein gewaltiger, mit den Göttern verwandter Stier, wohnte im Labyrinth, einem Palast von verwirrender Weiträumigkeit und unübersichtlicher Anlage, in dem sich die Opfer jedes Mal hoffnungslos verirrten.

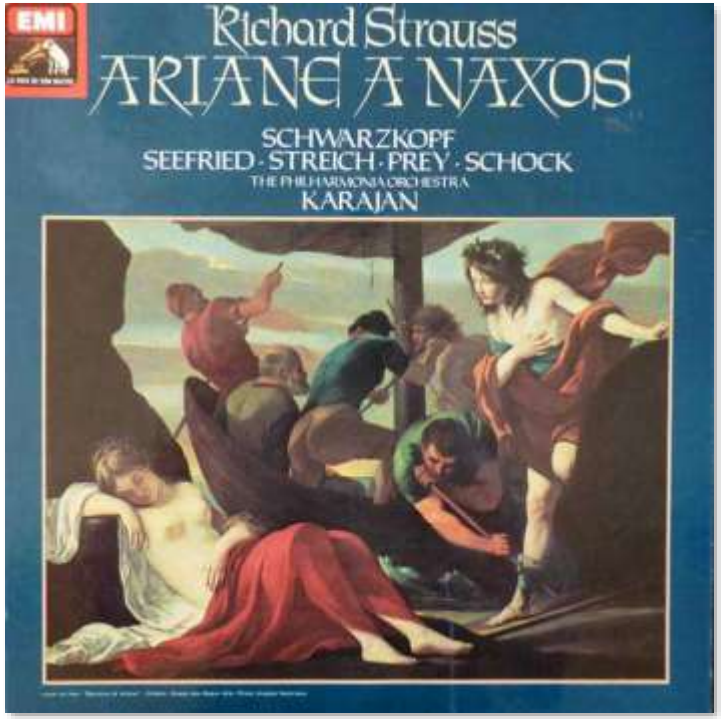
Theseus, der Sohn des Königs Aigeus von Athen, meldete sich freiwillig unter die sieben Jünglinge, in der Absicht, Athen vom Tribut zu befreien. Auf Kreta angekommen, macht er sich an Ariadne, die Tochter des Königs Minos von Kreta, heran und verführt sie dazu, etwas für seine Rettung zu unternehmen. Ariadne erhält vom Hofingenieur, der das Labyrinth erbaut hatte, einem gewissen Daidalos, den Lösungsalgorithmus – nicht als Computerprogramm, sondern in Form weiblicher Hardware, nämlich eines Knäuels, das als Ariadnefaden sprichwörtlich geworden ist. Wo der Faden nicht liegt, war man noch nicht. Wo er liegt, ist – vorläufig – der Lösungsweg. Wo er doppelt liegt, ist eine Sackgasse (da musste man wieder zurück). Und wo sich drei Fäden treffen würden, entstünde ein Kreisweg: Man drehe um, ehe man diese Stelle betritt. Mit diesen vier Regeln kann man jedes La-byrinth lösen, d.h. den Weg vom Start zum Ziel finden. Theseus ist daher erfolgreich und findet den Minotauros. Es fällt auf, dass der allen Beschreibungen nach gewiss nicht einfache Kampf mit dem Ungeheuer in den Schilderungen verblasst gegen die erste Auflösung eines Labyrinths. Geistige Leistungen haben eben schon im Altertum die Umwelt beeindruckt, wenn sie auch damals mitunter mit einem unrichtigen Erfindernamen verbunden wurden.

**) In: 50 Jahre Kybernetik in Österreich, e&i elektrotechnik und informationstechnik, Mai 2004, 121. Jg., H. 5, S. 171-179*



Theseus and Ariadne by the Cretan labyrinth; in the foreground the labyrinth (lettered: 'AbbE RINTO') with the two figures named: 'TESEO' and 'ADRIANNA'; in the background from the left Ariadne is seen abandoned on a rocky island, making signals to Theseus's ship, plunging into the sea, being rescued by Jupiter and carried to heaven; on the right Aegeus, father of Theseus, is shown throwing himself into the sea. Anonymous, print made by Baccio Baldini, c. 1460-70, britishmuseum.org

Man versteht, dass König Minos ärgerlich wird, sobald man ihm den Verrat des Lösungsalgorithmus hinterbringt. Da seine Tochter indessen mit Theseus auf dem Athener Schiff durchgebrannt ist, rächt sich Minos wenigstens an seinem Hofingenieur, indem er diesen samt dessen Sohn Ikaros kurzerhand in das Labyrinth einmauern lässt. Dabei unterschätzt er aber den Erfindungsreichtum des guten Ingenieurs. Aus Federn abgeschossener Vögel und mit Wachs von im Labyrinth nistenden Bienen baut Daidalos zwei Paar Flügel. Vater und Sohn verlassen das Labyrinth und Kreta auf dem Luftwege. Leider schlägt der junge Mann die Mahnungen des Vaters in den Fahrtwind, ja nicht von der vorgeschriebenen Luftstrasse abzuweichen. Ikaros geht aus purer Lust auf zu grosse Flughöhe und gerät in die Wärmestrahlung des Gegenverkehrs, wo fahrplanmässig Gott Helios sein Sonnengefährt steuert. Das Wachs der Flügel schmilzt, und der hoffnungsvolle, aber undisziplinierte Jüngling stürzt ab. Immerhin wurde zu seinem Angedenken das entsprechende Meer nach ihm benannt.



Daidalos hingegen landet glücklich in Ägypten und ersucht um politisches Asyl, das für einen Experten natürlich rasch gewährt wird. Ein Auslieferungsbegehren von König Minos wird mangels bestehender Abkommen aus prinzipiellen Gründen abgelehnt. Es kann nicht ausgeschlossen werden, dass die Ingenieurkunst eines Heron von Alexandrien aus einer Tradition erwuchs, die auf Daidalos zurückgeht. Dass Theseus Ariadne auf Naxos allein zurücklässt, ist eine andere Geschichte, zu der man Näheres in der Oper erfahren kann.

Zemaneks automatisierter Ariadnefaden

Zemaneks automatisierter Ariadnefaden geht auf ein Projekt von Claude Shannon zurück, der Anfang der 1950er-Jahre eine „maze-solving machine" konstruierte. Dabei sucht sich eine Robotermaus „Theseus", geführt von kupfernen Schnurrhaaren, einen Weg durch ein Metall-Labyrinth. Die Zeitschrift „Popular Science" beschreibt dies im März 1952 stakkatoartig so: „Mouse solves maze by trial and error on first attempt. On second try, mouse ‚remembers' correct route, avoids blind alleys, and heads straight for ‚cheese'."

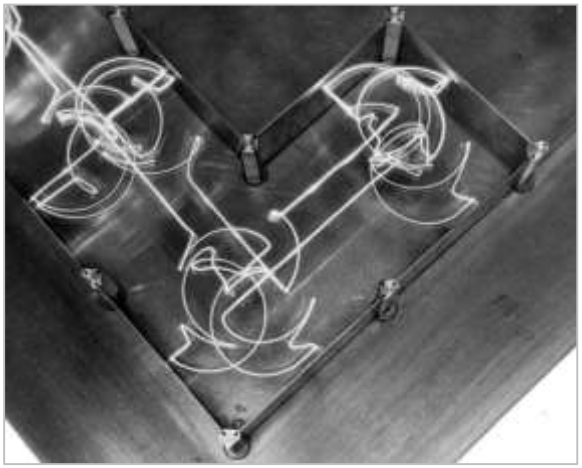
Claude Shannon mit seiner Robotermaus Theseus

Das Artefakt wird heute vom MIT-Museum verwaltet. In einer Beschreibung dazu heisst es dort:

„In December 1949, shortly after their marriage, Betty Shannon gave Claude Shannon a Meccano set for Christmas. In short order he had built a mechanical turtle with an antenna, which would run around a room and back away after encountering obstacles. The original mouse for Theseus was a descendent of that turtle, about an inch and a half square. Parts of the top maze for Theseus were machined at Bell Labs, but the circuitry was built at home on the living room floor. Betty Shannon, who was working as a technician at Bell Labs' s microwave research lab, did most of the wiring. A metal maze with walls whose position can be changed sits on top of a box; inside the box, a plotter-like electromagnet controls the movement of the mouse, and a series of relays calculate its position and remember successful and unsuccessful paths.

Zemaneks automatisierter Ariadnefaden (2)

After its demonstrations at Bell Labs functions, the machine continued to be used for experiments: Andrew Shannon recalled using it in elementary school science fair projects with real mice. Since the walls of the maze were low enough for live mice to jump over, the metal maze had to be chilled before use, so that the mice would prefer to keep moving instead of jumping over the sides."
[https://webmuseum.mit.edu/detail.php?module=objects&type=related&kv=76066]



Lichtspur der sich durch das Labyrinth tastenden Maus „Theseus" von Shannon
http://cyberneticzoo.com/wp-content/uploads/Shannon_Maze_Timelapse_Corner-x640.jpg

Zemanek greift zusammen mit seinem Diplomanden Richard Eier diese Idee auf, verbessert aber die Labyrinthsuche durch einen „automatisierten Ariadnefaden". Die Autoren beschreiben dies 1960 in einem Aufsatz „Automatische Orientierung im Labyrinth" (Elektron. Rechenanlagen 2(1), 23-31). Die Idee sei hier aus dem Anfangsteil der Veröffentlichung zitiert:

„Das Absuchen eines Labyrinths ohne System wird im allgemeinen erfolglos sein. Ein Teil der Gänge wird immer wieder betreten werden, während ein anderer völlig unbeachtet bleiben mag. Auch die Vorschrift, an jedem Verzweigungspunkt den nächsten rechten (oder linken) Ausgang zu versuchen, bringt keine Lösung, wenn in sich geschlossene Wege bestehen. Wird nämlich einer davon einmal durchlaufen, so ermöglicht es diese Vorschrift nicht, aus der Kreisbahn auszubrechen. Sie allein gestattet nicht einmal die Feststellung, dass dieser Fall eingetreten ist. Aber selbst die Erkenntnis, dass es so ist, muss nicht zum Ziel führen, denn ein systemloses Ausbrechen schützt nicht vor beliebig vielen Wiederholungen, von Kreisbahn zu Kreisbahn ausbrechend.

Um mit Sicherheit alle möglichen Wege eines Labyrinths zu durchlaufen und dabei das gegebene Ziel zu finden, müssen alle abgesuchten Verzweigungspunkte markiert werden. Der Ariadnefaden ist für die Markierung bestens geeignet. Er zeigt an, wo man noch nicht war, welchen Weg man zweimal

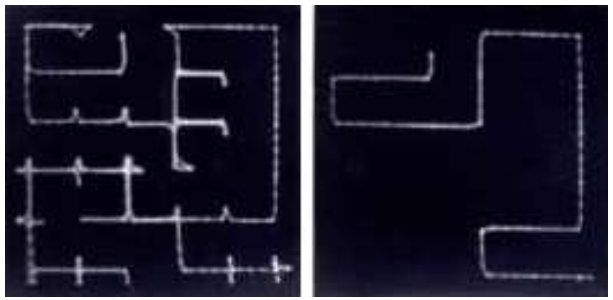
Zemaneks automatisierter Ariadnefaden (3)

durchlaufen hat und wie man zum Ausgangspunkt zurückfindet. Eine geringe Verschärfung der Aussagen und Definitionen erlaubt eine Systematik, die sich für die Automatisierung eignet. […]

Beim Absuchen des Labyrinths wird der Faden ausgelegt; ein Ende ist beim Eingang und das andere am jeweiligen Standort, wo es den Rückweg anzeigt. Im allgemeinen wird ein Verbindungsgang nur einmal durchlaufen, so dass ein einfacher Faden darin liegen bleibt. Nur in Sackgassen ist man gezwungen, bis zur Möglichkeit einer anderen Abzweigung zurückzukehren, so dass in der Sackgasse ein doppelter Faden liegen bleibt und von einem dritten Betreten abhält. Bei der Abzweigung wird man nur unmarkierte Verbindungsgänge betreten; erst wenn alle möglichen Fortschreitungsrichtungen ebenfalls als Sackgassen gekennzeichnet sind, wird man sich entschliessen müssen, in der Richtung des einfachen Fadens zurückzulaufen bis zu einer anderen Verzweigung, wodurch der ganze abgesuchte Teilkomplex als Sackgasse gekennzeichnet erscheint.

Backtracking

Es kann aber auch der Fall eintreten, dass ein unmarkierter Verbindungsgang zu einer bereits einmal markierten Verzweigung zurückführt und damit also einen Kreisweg schliesst. In einer solchen Verzweigung liegen dann drei einzelne Fäden; zwei davon führen zurück zum Eingang (wobei einer der beiden an dieser Verzweigung noch einmal vorbeiführt), der dritte weist in den Kreisweg hinein. Wenn der Faden keine besonderen Merkmale trägt, weiss man mit Sicherheit nur, dass der Gang, den man gerade gekommen ist, der eine Rückweg sein muss, während eine Aussage über die beiden anderen Richtungen, in denen noch einfache Fäden liegen, nicht möglich ist. Darum wird in dieser Situation der Gang, den man gerade gekommen ist, zurückverfolgt, wodurch er mit doppeltem Faden als Sackgasse gekennzeichnet ist. Diese Suchvorschrift erlaubt es, Gänge, die mit doppelten Fäden belegt sind, vom wei-



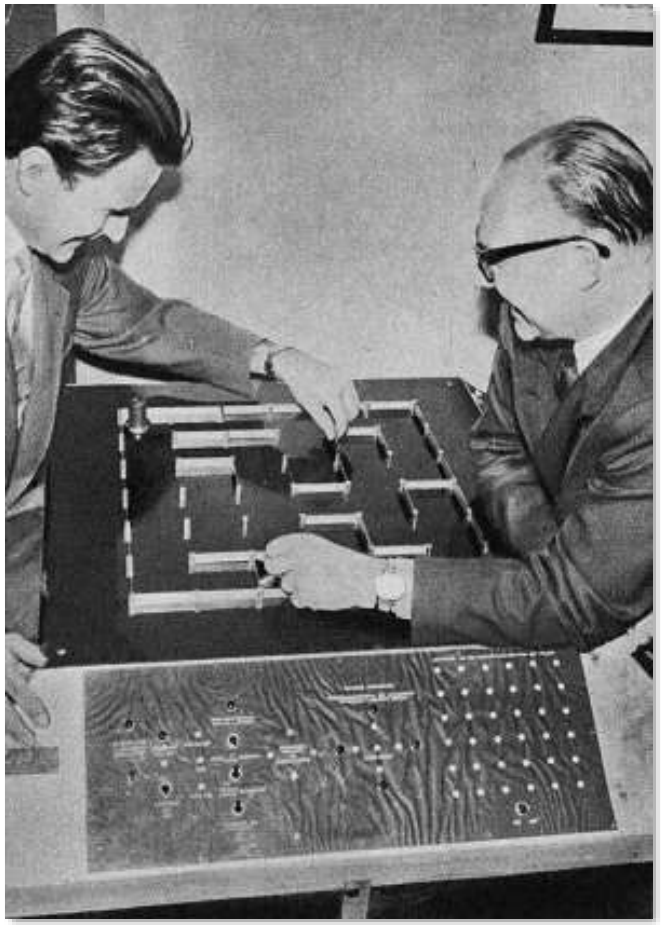
Initialer Weg und der gelernte sackgassenfreie kürzeste Weg vom Start- zum Zielpunkt

Zemaneks automatisierter Ariadnefaden (4)

teren Durchsuchen auszuschliessen und Kreiswege in Sackgassen umzuwandeln. Es werden alle noch nicht besuchten Gänge erkannt und nach dem Ziel abgesucht. Am Ende weist dann ein einfacher Faden den Weg vom Eingang zum Ziel und umgekehrt."

Richard Eier (der 1972 Professor und Direktor des neu gegründeten Instituts für Datenverarbeitung an der TU Wien wurde) baute 1958 im Rahmen seiner Staatsprüfungsarbeit „Gedächtnissteuerung zur Orientierung in einem Labyrinth" am Wiener Institut für Schwachstromtechnik ein Modell, das auf dem beschriebenen Verfahren beruht. Er schreibt dazu:

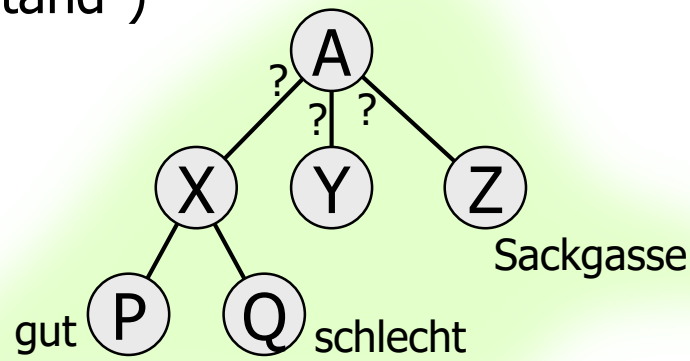
„Das Labyrinth wird auf einer Platte in einem Feld bestehend aus sechs mal sechs Quadraten durch Aufstellen von Trennwänden an Quadratseiten abgesteckt. Jedes dieser Quadrate entspricht einem Knoten, dem bis zu vier Ausgänge eingeräumt werden können, so dass innerhalb dieser 36 Quadrate eine große Anzahl verschiedener Labyrinthe absteckbar ist. Die Richtungen der möglichen Ausgänge können nach den vier Himmelsrichtungen mit N, O, S und W bezeichnet werden. In dem ausgeführten Modell wurden zwei Drehwähler und 284 Relais verwendet."



Richard Eier und Heinz Zemanek am Labyrinthmaus-Demonstrationsmodell [Bild: Urania-Universum VII/1961, via blog.hnf.de/heinz-zemanek-1920-2014/]

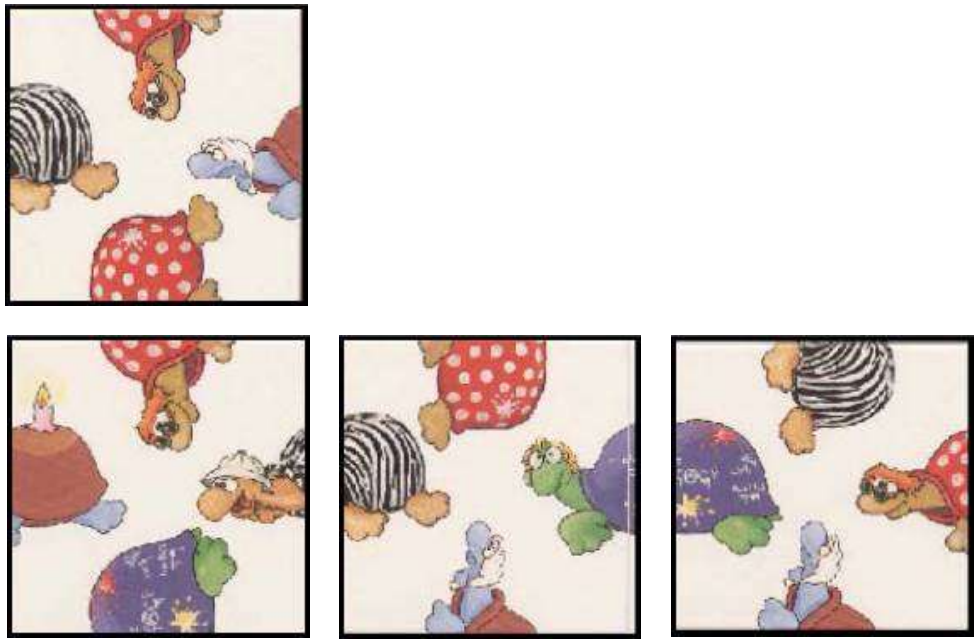
Durchmustern eines Entscheidungsbaums

- Idee: Den Baum aller möglichen Entscheidungen systematisch (rekursiv, depth first) durchlaufen:
  - Knoten = Entscheidungssituation („Zustand“)
  - Kante = Entscheidung
  - Nachfolgeknoten = Situation nach der Entscheidung
  - Blatt = Endsituation / Sackgasse



- Nützliche Strategie: Sackgassen möglichst früh entdecken
  - Sackgassen-Vorhersagbarkeit ist abhängig vom konkreten Problem
  - Spart evtl. viel Suchaufwand (ganze Unterbäume!)
  - Beispiel: Analyse eines Spielbaums: „Dieser Zug erscheint hoffnungslos“

Ein Puzzle



- Gegeben seien 9 derartige Quadrate mit verschiedenen Hälften von Figuren
- Aufgabe: Diese so zu einem 3×3-Feld zusammensetzen, dass aneinanderstossende Ränder zueinander passen


Das verflixte
Schildkröten-
Spiel

BILLION BUBBLE BEVERAGES

MYSTERY PUZZLE

DIRECTIONS

The object is to arrange the nine pieces in a square - three rows of three each so that the upper and lower halves of twelve "VESS" bottles are correctly matched and each of these bottles completely within the same color. For example, if yellow is the background color of the upper half of a "VESS" bottle, then yellow must be the background color of the lower half of that bottle.

Als billig herzustellende Werbemittel gab es solche „edge-matching puzzles“ schon vor rund 80 Jahren

www.robspuzzlepage.com/images/vess-billion-bubbles-pcs1.jpg

Naive Puzzle-Lösung mit „brute force“

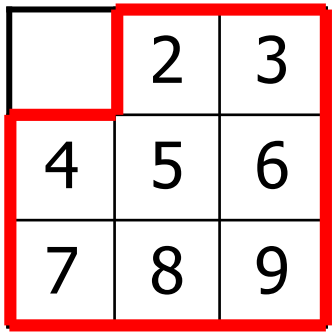
- Das Problem ist schwierig (frustrierend und reizvoll zugleich!), da man evtl. erst recht spät merkt, dass ein Ansatz nicht „aufgeht“ und man grosse Teile zurückbauen muss!
  - Der Puzzle-Designer hat es wohl absichtlich so gestaltet, dass viele partielle und wenig vollständige Lösungen existieren
- Naiver Ansatz („brute force“): Es gibt nur endlich viele verschiedene Anordnungen der Quadrate in einem 3×3-Feld; → generiere alle systematisch und teste jeweils auf Korrektheit

Davon gibt es $9 \times 4 \times 8 \times 4 \times 7 \times 4 \times \ldots 1 \times 4^9 = 4^9 \times 9! \approx 100\,000\,000\,000$

Aber sollte das nicht effizienter gehen? **?**

Das ist einfach! (Für die Kryptographie ist es übrigens entscheidend, dass es kombinatorische Probleme gibt, für die man nur sehr schwer Lösungen findet, diese aber einfach überprüfen kann)

Ein rekursiver Ansatz

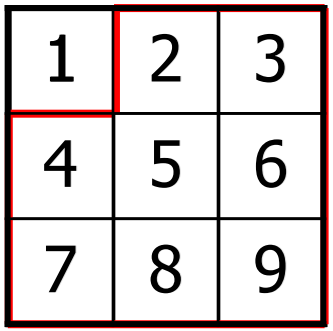


Problem:
- Gibt es eine Lösung?
- Wenn ja, wie lautet sie?

1) Bestimme zunächst (rekursiv) die Lösungen für ein Spielfeld, welches um 1 Einzelfeld kleiner ist
2) Versuche für eine solche Lösung (falls existent) das zusätzliche Einzelfeld so mit einem der restlichen Spielsteine zu besetzen, dass eine Gesamtlösung entsteht

Oder andersherum?

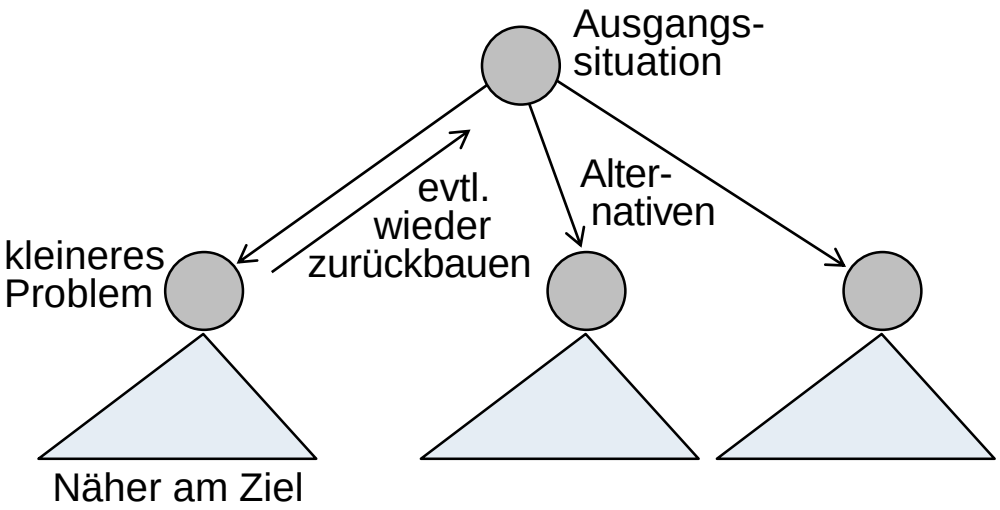
- Solange noch nicht alle Spielsteine (in allen 4 Orientierungen) für Position 1 ausprobiert worden sind:
  1) Belege Position 1 auf eine neue Weise
  2) Löse rekursiv das einfachere Problem „Gibt es mit den restlichen Spielsteinen eine *dazu passende* Lösung für das Spielfeld der Positionen 2,...,9 ?"
  3) „ja": → Gesamtlösung zurückliefern



- „Nein" melden, wenn erfolglos alle Möglichkeiten probiert

- Welcher der beiden Ansätze ist besser?
- Sollte man nicht die Mitte (Position 5) recht früh belegen?
  - Diese induziert sehr viele „Constraints" → Sackgassen früh erkennbar?

Der Backtracking-Baum



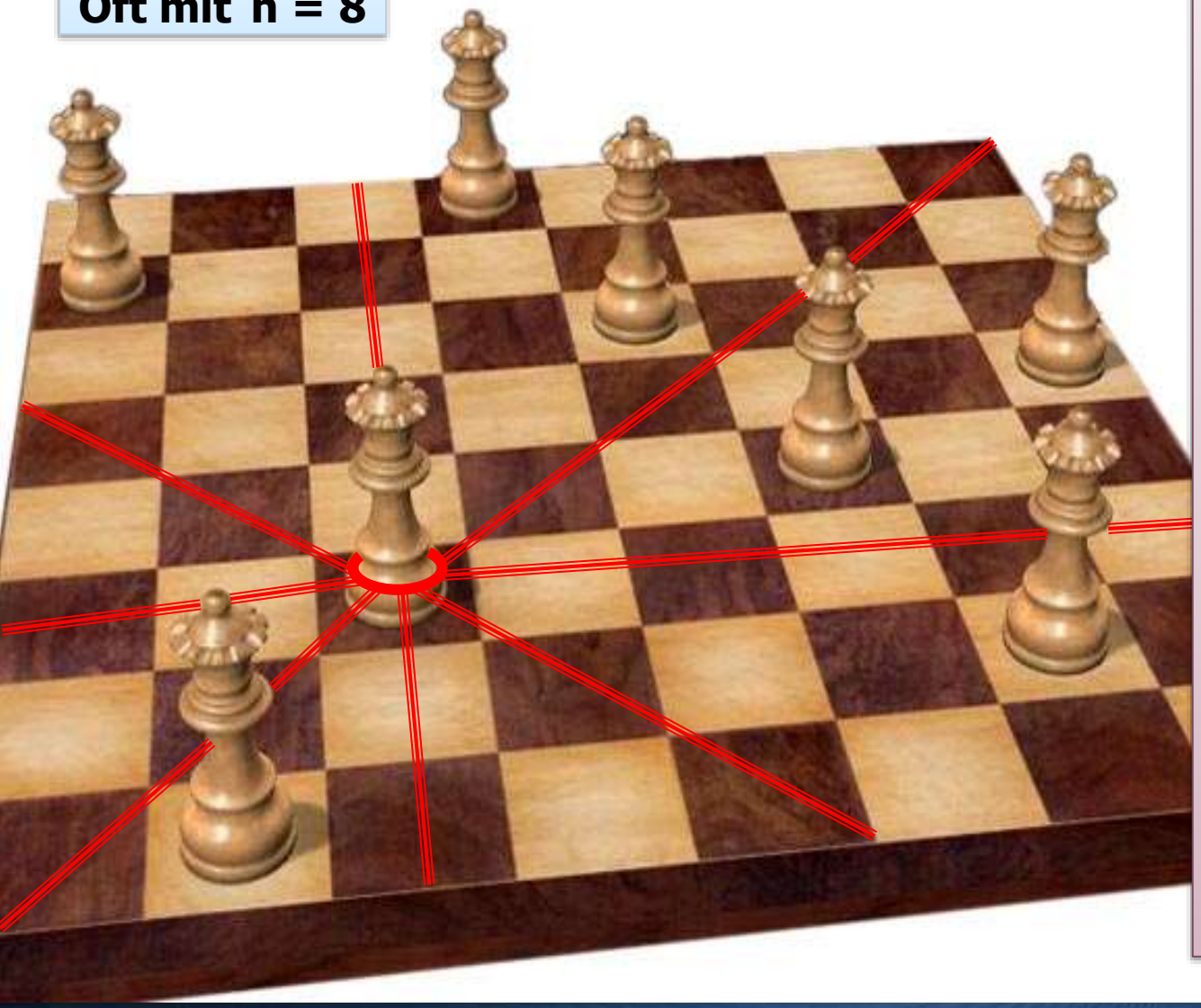
Je nach Problem kann man einem Zwischenzustand (= Teilproblem) eventuell ansehen, dass er keinesfalls zu einer Gesamtlösung beitragen kann → „tree pruning".

Gelegentlich wird ein Unterbaum auch dann gekappt, wenn man zwar nicht sicher ist, aber vermutet, dass sich in ihm keine (gute) Lösung befindet. Hierfür verwendet man problembezogene Heuristiken.

- Der Baum wird „depth first" durchlaufen: Zunächst wird ganz links abgestiegen...
- Illegale partielle Situationen werden nicht weiter ausgebaut (→ Sackgasse)
  - Im Gegensatz zum „Brute-Force-Ansatz", der alle Situationen betrachtet
  - Um den exponentiellen Aufwand möglichst stark abzuschwächen, sollten frühzeitig so viele Alternativen wie möglich ausgeschlossen werden

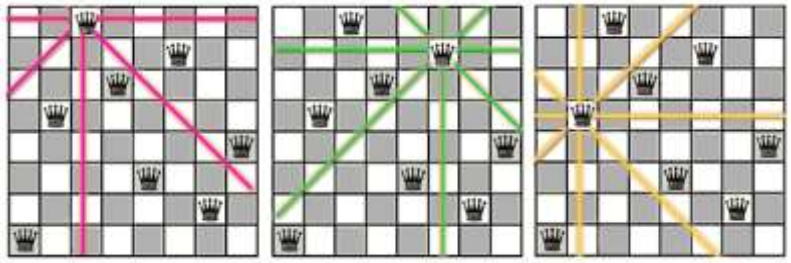
Das n-Damen-Problem

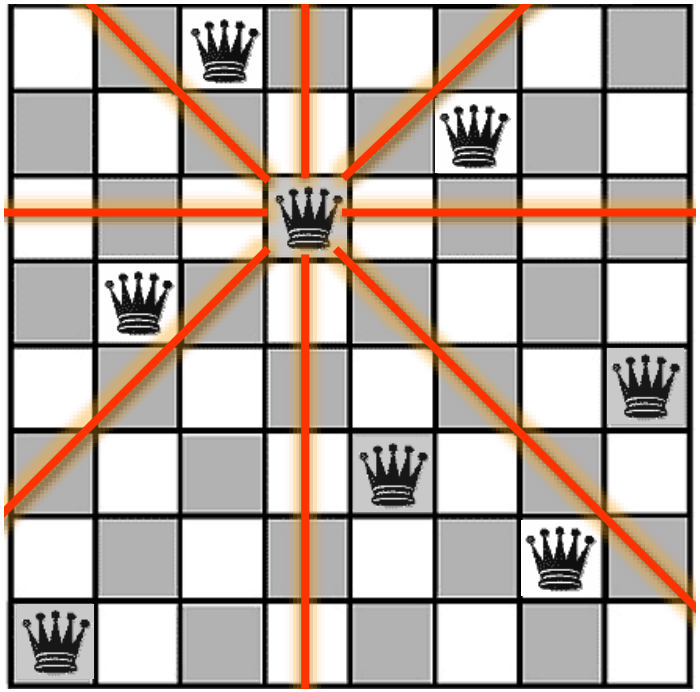
Oft mit n = 8



Das 8-Damen-Problem wurde von Max Bezzel 1848 in der Berliner „Schachzeitung" veröffentlicht („*Wie viele Steine mit der Wirksamkeit der Dame können auf das im Uebrigen leere Brett in der Art aufgestellt werden, dass keiner den andern angreift und deckt, und wie müssen sie aufgestellt werden?*"), blieb zunächst aber unbeachtet. Erst als die Aufgabe 1850 vom Schachexperten Dr. Franz Nauck in der Leipziger „Illustrirten Zeitung" erneut zur Diskussion gestellt wurde („*Man kann 8 Schachfiguren, von denen jede den Rang einer Königin hat, auf dem Brett so aufstellen, dass keine von einer anderen geschlagen werden kann*"), fand sie breites Echo. Nauck selbst publizierte in der gleichen Zeitschrift einen Monat später 60, und drei Monate danach 92 Lösungen, allerdings ohne Argument, wieso es nicht noch mehr geben könnte. Erst 1874 konnte gezeigt werden, dass 92 maximal ist.

Das n-Damen-Problem





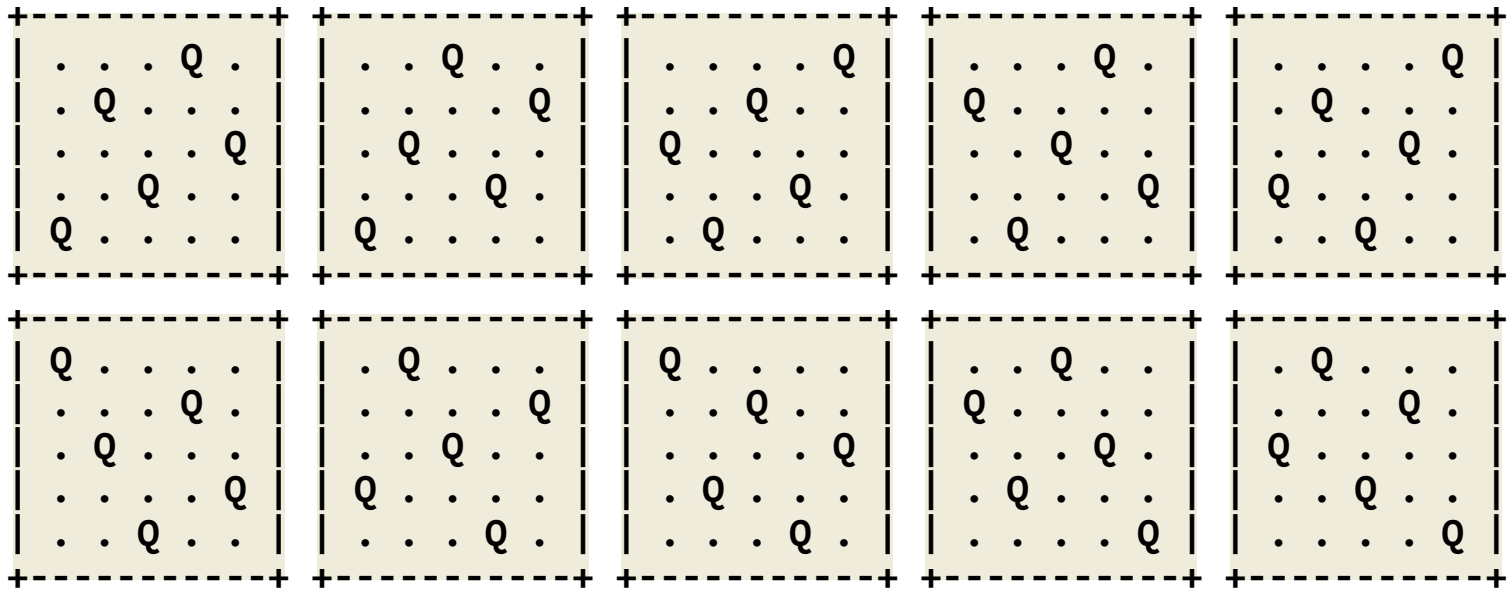
- Keine der n „Damen“ auf dem n×n Schachbrett darf andere bedrohen
  - D.h. keine zwei Damen stehen in der selben Zeile, Spalte, Diagonale
- Wie viele verschiedene derartige Aufstellungen gibt es (z.B. für n=8)?
  - Beachte Dreh- / Spiegelsymmetrien

Hier kann man es interaktiv selbst ausprobieren:
www.murderousmaths.co.uk/GAMES/queens/queens.htm

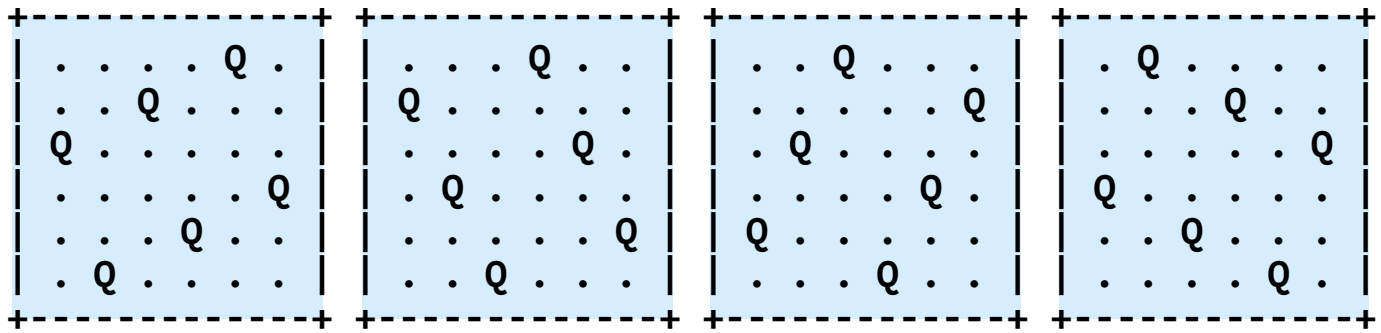
Die Dame ist die stärkste Figur im Schachspiel. Im Englischen wird sie als „Queen“ bezeichnet; im Deutschen, Französischen, Niederländischen und einigen anderen Sprachen ist für die Gattin des Schachkönigs (analog zu den Spielkarten) die Bezeichnung „Dame“ (ital.: „donna“) üblich. Ursprünglich symbolisierte die Figur neben dem König dessen Berater oder Minister bzw. Wesir, im Arabischen heisst sie heute noch so („wazīr“, وزير).

Alle Lösungen für n=5 und n=6

a(5) = 10:



a(6) = 4:



https://oeis.org/A000170/

Das 8-Damen-Problem

Beschreibung des Problems im „Mathematischen Wörterbuch“, Band VI (Berlin, 1867), herausgegeben von Ludwig Hoffmann und Leopold Natani, S. 348-350, unter dem Stichwort „Rösselsprung“:

Es handelt sich darum, auf ein Schachbrett 8 Königinnen aufzustellen, derart, dass keine irgend eine der andern, nach dem Gang, welchem die Königin auf dem Schachbrette folgt, zu schlagen im Stande ist. Es lässt sich dieser Aufgabe ein rein mathematischer Ausdruck geben.

„Es sind acht Felder gegeben, deren Reihenfolge durch eine darüber geschriebene Zahl angezeigt ist, welche wir Ordnungszahl nennen. Es soll in jedes der acht Felder eine andere der ersten acht natürlichen Zahlen derart geschrieben werden, dass die Differenz zweier darunter nicht gleich der Differenz ihrer Ordnungszahlen ist. Steht also im dritten Felde eine 4, so darf z. B. im fünften weder eine 6 noch eine 2 stehen, weil $4-2 = 6-4 = 5-3$ ist.“

Bedeuten dann die Ordnungszahlen die Felder einer Columne des Schachbrettes, und die Zahlen der Felder selbst die Stelle, welche die betreffende Königin in ihrer Horizontalreihe einnimmt, so gibt die nach diesem Gesetze gebildete Zahlenreihe die Stellung der 8 Königinnen.

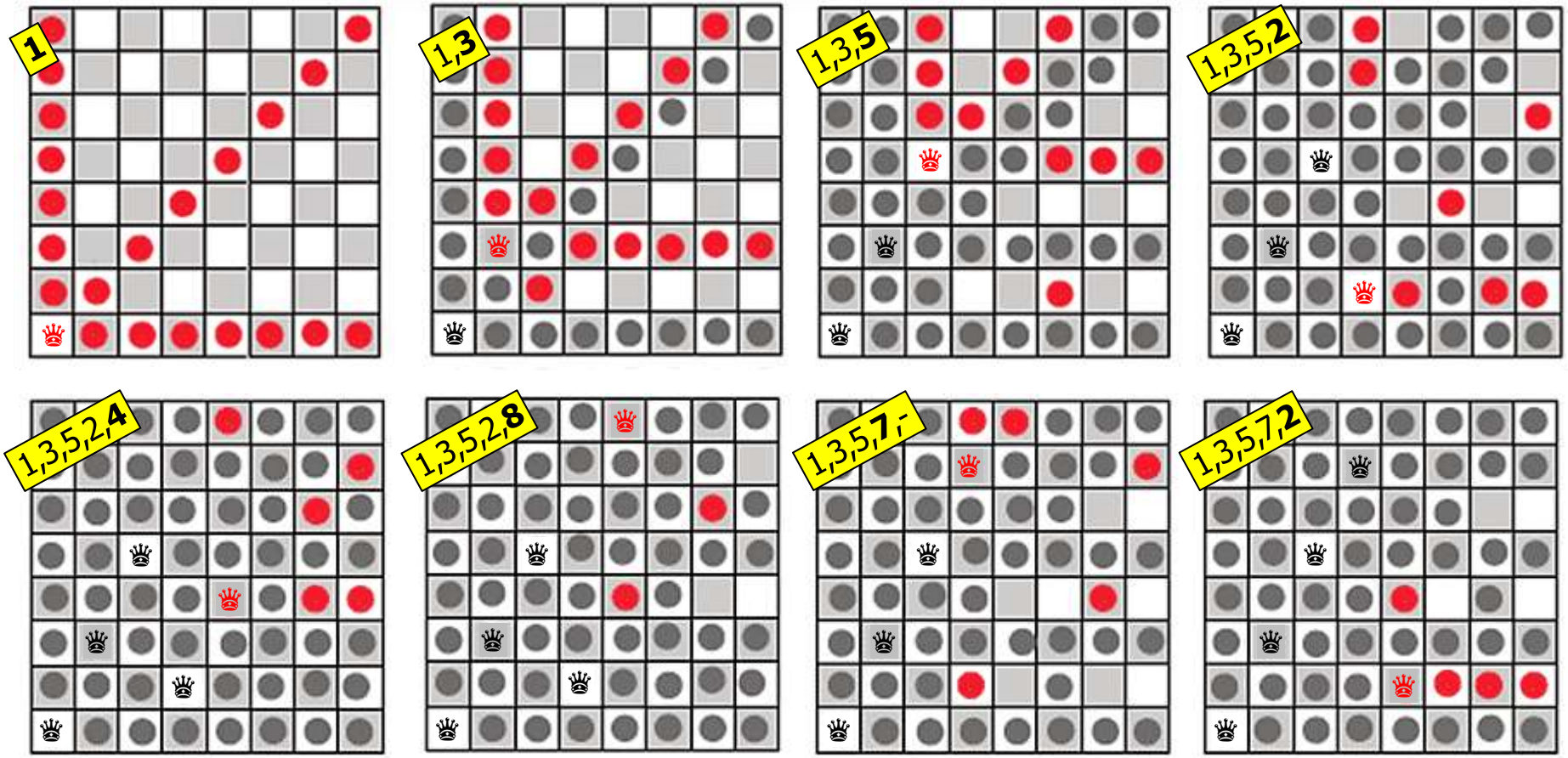
Ist eine Auflösung dieser Aufgabe gefunden, so ergeben sich aus ihr im Allgemeinen sieben andere, da man das Schachbrett viermal um einen rechten Winkel drehen und statt jeder dieser vier Stellungen die symmetrische (das Spiegelbild) nehmen kann. [...]

Um die Auflösungen der Aufgabe zu ermitteln, gibt es wohl keine andere Verfahrungsweise als Ausschliessung der Anordnungen, welche der Aufgabe nicht entsprechen, wobei sich allerdings mancherlei Erleichterungen ergeben.

Als selbstständige Lösungen bezeichnen wir alle, die nicht unter die acht gehören, welche aus einander entstehen. Es gibt im Ganzen zwölf selbstständige Lösungen, worunter eine symmetrische, aus denen sich also im Ganzen 92 ergeben.

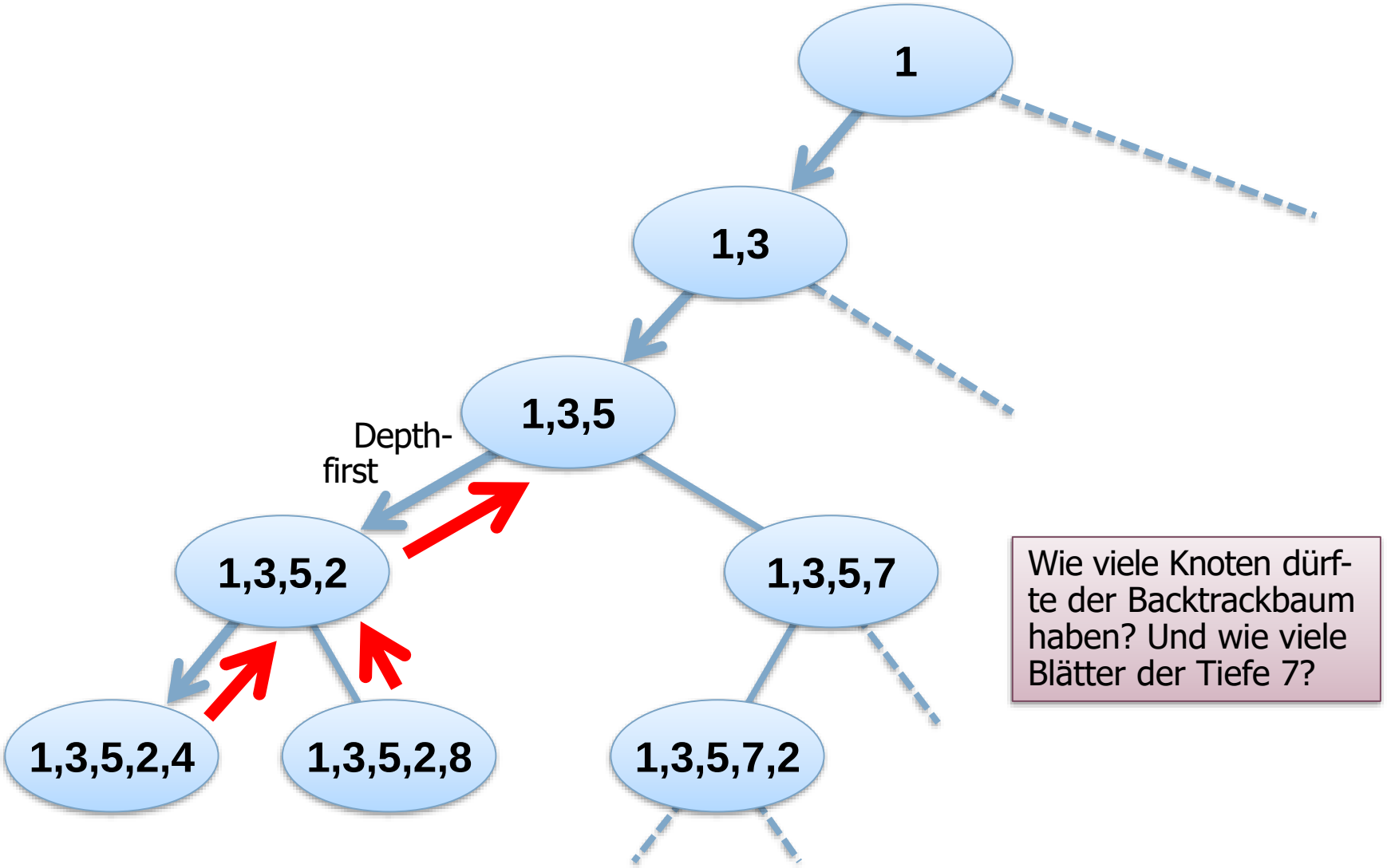
Eine Rückbausituation beim 8-Damen-Problem

In jede Spalte (beginnend links) eine Dame setzen (jew. von unten nach oben durchprobieren)



Setzt man sukzessive bzgl. der Spalten eine Dame in die unterste, noch nicht bedrohte Zeile, so sind schon nach 5 Schritten alle Felder bedroht, eine sechste Dame kann nicht mehr gesetzt werden. Nun beginnt der Rückbau, die zuletzt gesetzte Dame der 5. Spalte wird entfernt und neu platziert; allerdings wird dadurch die 6. Spalte noch nicht frei. Daher wird jetzt die Dame der 4. Spalte neu platziert...

Eine Rückbausituation beim 8-Damen-Problem



Wie viele Knoten dürfte der Backtrackbaum haben? Und wie viele Blätter der Tiefe 7?

Bau und Rückbau bis zur ersten Lösung

Gesetzt (1,1)
Gesetzt (2,3)
Gesetzt (3,5)
Gesetzt (4,2)
Gesetzt (5,4)
Entfernt (5,4)
Gesetzt (5,8)
Entfernt (5,8)
Entfernt (4,2)
Gesetzt (4,7)
Gesetzt (5,2)
Gesetzt (6,4)
Gesetzt (7,6)
Entfernt (7,6)
Entfernt (6,4)
Entfernt (5,2)
Gesetzt (5,4)
Entfernt (5,4)
Entfernt (4,7)
Gesetzt (4,8)
Gesetzt (5,2)
Gesetzt (6,4)
Gesetzt (7,6)
Entfernt (7,6)
Entfernt (6,4)
Entfernt (5,2)
Gesetzt (5,4)
Entfernt (5,4)
Entfernt (4,8)
Entfernt (3,5)
Gesetzt (3,6)
Gesetzt (4,2)
Gesetzt (5,7)
Gesetzt (6,5)
Entfernt (6,5)
Entfernt (5,7)
Entfernt (4,2)
Gesetzt (4,8)
Gesetzt (5,2)

Gesetzt (6,4)
Entfernt (6,4)
Gesetzt (6,5)
Entfernt (6,5)
Entfernt (5,2)
Entfernt (4,8)
Entfernt (3,6)
Gesetzt (3,7)
Gesetzt (4,2)
Gesetzt (5,4)
Gesetzt (6,8)
Entfernt (6,8)
Entfernt (5,4)
Gesetzt (5,8)
Gesetzt (6,5)
Entfernt (6,5)
Entfernt (5,8)
Entfernt (4,2)
Entfernt (3,7)
Gesetzt (3,8)
Gesetzt (4,2)
Gesetzt (5,4)
Entfernt (5,4)
Entfernt (5,7)
Entfernt (5,7)
Entfernt (4,2)
Gesetzt (4,6)
Gesetzt (5,2)
Entfernt (5,2)
Gesetzt (5,4)
Gesetzt (6,2)
Gesetzt (7,5)
Entfernt (7,5)
Entfernt (6,2)
Entfernt (5,4)
Entfernt (4,6)
Entfernt (3,8)
Entfernt (2,3)
Gesetzt (2,4)

Gesetzt (3,2)
Gesetzt (4,5)
Gesetzt (5,3)
Entfernt (5,3)
Gesetzt (5,8)
Entfernt (5,8)
Entfernt (4,5)
Gesetzt (4,7)
Gesetzt (5,3)
Entfernt (5,3)
Entfernt (4,7)
Gesetzt (4,8)
Gesetzt (5,3)
Gesetzt (6,7)
Entfernt (6,7)
Entfernt (5,3)
Gesetzt (5,6)
Gesetzt (6,3)
Entfernt (6,3)
Entfernt (5,6)
Entfernt (4,8)
Entfernt (3,2)
Gesetzt (3,6)
Gesetzt (4,3)
Entfernt (4,3)
Gesetzt (4,8)
Gesetzt (5,2)
Gesetzt (6,5)
Gesetzt (7,3)
Entfernt (7,3)
Entfernt (6,5)
Gesetzt (6,7)
Gesetzt (7,3)
Entfernt (7,3)
Entfernt (6,7)
Entfernt (5,2)
Gesetzt (5,3)
Gesetzt (6,5)
Entfernt (6,5)

Gesetzt (6,7)
Entfernt (6,7)
Entfernt (5,3)
Entfernt (4,8)
Entfernt (3,6)
Gesetzt (3,7)
Gesetzt (4,3)
Gesetzt (5,6)
Gesetzt (6,2)
Gesetzt (7,5)
Entfernt (7,5)
Entfernt (6,2)
Entfernt (5,6)
Gesetzt (5,8)
Gesetzt (6,2)
Gesetzt (7,5)
Entfernt (7,5)
Entfernt (6,2)
Entfernt (5,8)
Entfernt (4,3)
Gesetzt (4,5)
Gesetzt (5,2)
Entfernt (5,2)
Gesetzt (5,3)
Entfernt (5,3)
Gesetzt (5,8)
Gesetzt (6,2)
Entfernt (6,2)
Entfernt (5,8)
Entfernt (4,5)
Entfernt (3,7)
Gesetzt (3,8)
Gesetzt (4,3)
Entfernt (4,3)
Gesetzt (4,5)
Gesetzt (5,2)
Entfernt (5,2)
Gesetzt (5,3)
Entfernt (5,3)

Entfernt (4,5)
Entfernt (3,8)
Entfernt (2,4)
Gesetzt (2,5)
Gesetzt (3,2)
Gesetzt (4,6)
Gesetzt (5,3)
Gesetzt (6,7)
Gesetzt (7,4)
Entfernt (7,4)
Entfernt (6,7)
Entfernt (5,3)
Entfernt (4,6)
Gesetzt (4,8)
Gesetzt (5,3)
Gesetzt (6,7)
Gesetzt (7,4)
Entfernt (7,4)
Entfernt (6,7)
Entfernt (5,3)
Gesetzt (5,6)
Gesetzt (6,3)
Entfernt (6,3)
Gesetzt (6,4)
Entfernt (6,4)
Entfernt (5,6)
Entfernt (4,8)
Entfernt (3,2)
Gesetzt (3,7)
Gesetzt (4,2)
Gesetzt (5,4)
Gesetzt (6,8)
Entfernt (6,8)
Entfernt (5,4)
Gesetzt (5,6)
Gesetzt (6,3)
Entfernt (6,3)
Gesetzt (6,8)
Entfernt (6,8)

Entfernt (5,6)
Entfernt (4,2)
Entfernt (3,7)
Gesetzt (3,8)
Gesetzt (4,2)
Gesetzt (5,4)
Gesetzt (6,7)
Gesetzt (7,3)
Entfernt (7,3)
Entfernt (6,7)
Entfernt (5,4)
Gesetzt (5,7)
Gesetzt (6,3)
Gesetzt (7,6)
Entfernt (7,6)
Entfernt (6,3)
Entfernt (5,7)
Entfernt (4,2)
Gesetzt (4,6)
Gesetzt (5,3)
Gesetzt (6,7)
Gesetzt (7,2)
Gesetzt (8,4)

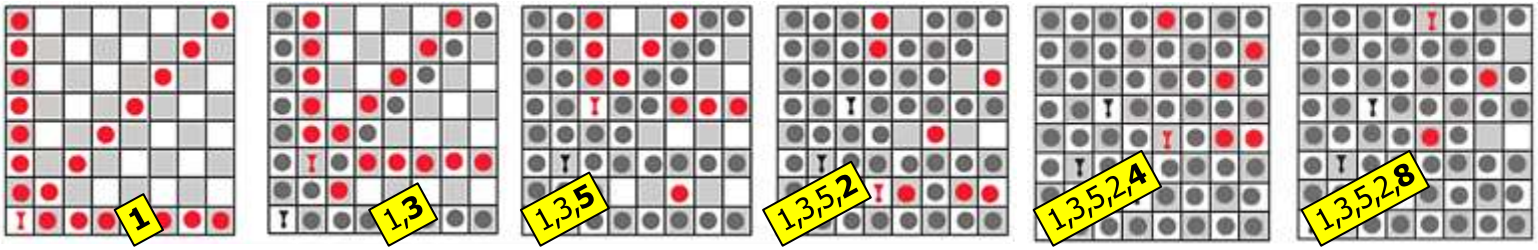
Lösung gefunden: 15863724

Bis die *erste* Lösung auf diese Weise gefunden wird, dauert es eine ziemliche Zeit – dafür wurde das einige Slides weiter hinten befindliche Programm benutzt. Der dabei aufgebaute Backtrackingbaum hat 113 Knoten; bei Ermittlung *aller* 92 Lösungen besteht er aus 2056 Knoten.

Gauß und sein schicklich präparirtes Quadratnetz

Schon Carl Friedrich Gauß nutzte 1850 eine wie oben skizzierte Darstellung bei seinen Ansätzen, das 8-Damen-Problem zu lösen. Er hatte bereits 72 Lösungen gefunden, aber keine Geduld, dies zu Ende zu führen, als er im September 1850 an seinen „theuersten Freund", den Astronomen und Geodäten Heinrich Christian Schumacher (3. 9. 1780 – 28. 12. 1850), schrieb:

„[...] Auf einem schicklich präparirten Quadratnetz gehen die Tatonnements schneller. Sobald ein Platz besetzt wird, etwa mit einem ⊕, fallen schon von allen übrigen 63 Plätzen viele aus, die durch ein Zeichen ◯ als cassirt betrachtet werden. Besetzt man von den übrigen einen zweiten, so fallen wieder eine grosse Menge aus, und man gelangt bald dahin, entweder alle Plätze theils mit ⊕, theils mit ◯ besetzt zu finden, oder zu einer wahren Auflösung zu gelangen."



Er räsoniert im Sinne der oben betrachteten Backtracking-Situation: „Das Tatonniren ist nun sehr leicht. Z.B. ich versuche den Anfang 1,3,..... zu completiren. Vermöge jener zwei Bedingungen wird in der dritten Reihe nicht 2 und nicht 4 stehen dürfen, also nur 5,6,7 oder 8. Es müssen also die Anfänge 1,3,5,..... 1,3,6,..... 1,3,7,..... 1,3,8..... durchprobirt werden. Ich fange an mit 1,3,5. Vermöge jener Bedingungen darf am 4ten Platz nicht 4 und nicht 6 stehen. Es bleiben also bloss übrig 2,7,8 [...] Es bleiben also bloss die Anfänge: 1, 3, 5, 2, 4 und 1, 3, 5, 2, 8. Die Berücksichtigung obiger Bedingungen ergibt, dass bei dem Anfange 1, 3, 5, 2, 4 [...] Es fällt also dieser Anfang weg. Eben so darf auch für Anfang 1, 3, 5, 2, 8 [...] Es fällt also auch dieser Anfang weg. Der Anfang 1, 3, 5, 2 ist also überhaupt unzulässig." Und

Gauß und sein schicklich präparirtes… (2)

weiter: „Es liesse sich leicht über diese Gegenstände noch 1 oder ein Paar Bogen vollschreiben, aber man muss aufzuhören wissen. Am elegantesten ist es, die Sachen so einzukleiden, dass sie den complexen Zahlen angehören. Es heisst dann, man soll 8 verschiedene complexe Zahlen finden a + bi, so dass [...]".

Es ist verständlich, dass Gauß im Schachbrett seine „gaußsche Zahlenebene" sah und die Koordinaten der Stellungen als komplexe Zahl auffasste; bei der Lösung des kombinatorischen Problems scheint diese Darstellung allerdings keine besonderen Vorteile zu bringen, man muss letztlich wohl die „Tâtonnements" algorithmisch systematisieren – was Wilhelm Ahrens (1872 – 1927) in seinem Buch „Mathematische Unterhaltungen und Spiele" von 1901 als „Gestaltung des planmässigen Tatonnierens bezeichnete".

Der Mathematikhistoriker Siegmund Günther (1848 – 1923) meinte 1874 dazu: „Es ist eine ganz combinatorische Operation, bei der successive alles Untaugliche ausgeschieden wird, etwa in der Art des Siebes von Eratosthenes." Interessant ist auch seine Anmerkung zur Mechanisierung der Backtracking-Methode: „Es wäre nur noch nötig, sie dahin zu vervollkommnen, dass bei ihrer Anwendung gar keine besondere Genauigkeit mehr nötig, vielmehr das ganze Tatonnement völlig mechanisch wäre." Die reine Anwendung wäre jedenfalls kinderleicht, meint Ahrens: „So einfach, dass nach ihm der Franzose Laquière, wie beiläufig bemerkt sein mag, die 92 Lösungen für das gewöhnliche Schachbrett durch ein Kind in einem Nachmittag bestimmen lassen konnte, wobei nur 3 Fehler vorkamen."

Dass es keine einfache Beschreibung und keine geschlossene Formel für die Zahl der Lösungen gibt, sondern dass man „probieren" muss, wurde den Experten im Laufe der Zeit jedenfalls immer deutlicher – das wird auch in folgendem Text von Emil Pauls als Vermutung geäussert.

Das Maximalproblem der Damen auf dem Schachbrete.

(Studie aus dem Gebiete des mathematischen Schachs.)

Von E. Pauls.

Deutsche Schachzeitung, 29. Jg, (1874), Nr. 5, S. 129–134 und Nr. 9, S. 257–267

I.

Zu den Problemen, welche seit Jahrzehnten in der Schachwelt bekannt sind, ohne eine endgültige Lösung gefunden zu haben, gehört das Maximalproblem der Damen. Meist wird diese Aufgabe so aufgefasst, dass gefordert wird, 8 Damen auf dem gewöhnlichen Brete von 64 Feldern so aufzustellen, dass keine Dame die andere schlagen kann. In diesem beschränkten Sinne mag v. Jänisch in seinem vor 12 Jahren erschienenen „Traité“ die Aufgabe gelöst haben. Der russische Meister entwickelte die 92 möglichen Stellungen, muss indess nicht ganz mit seinen Beweisen befriedigt haben. Denn die Recension S. 88 — 1862 — dieser Zeitung erklärt, die Aufgabe scheine nicht ganz ohne Probiren löslich zu sein, und auch die ausführlichere Darlegung (1863 — S. 364) scheint indirect diese Angabe zu bestätigen.

„(Thesis II.) Die mathematische Analysis besitzt bis jetzt keine allgemeine Methode um ähnliche Fragen, wie die vorliegende, zu lösen.“

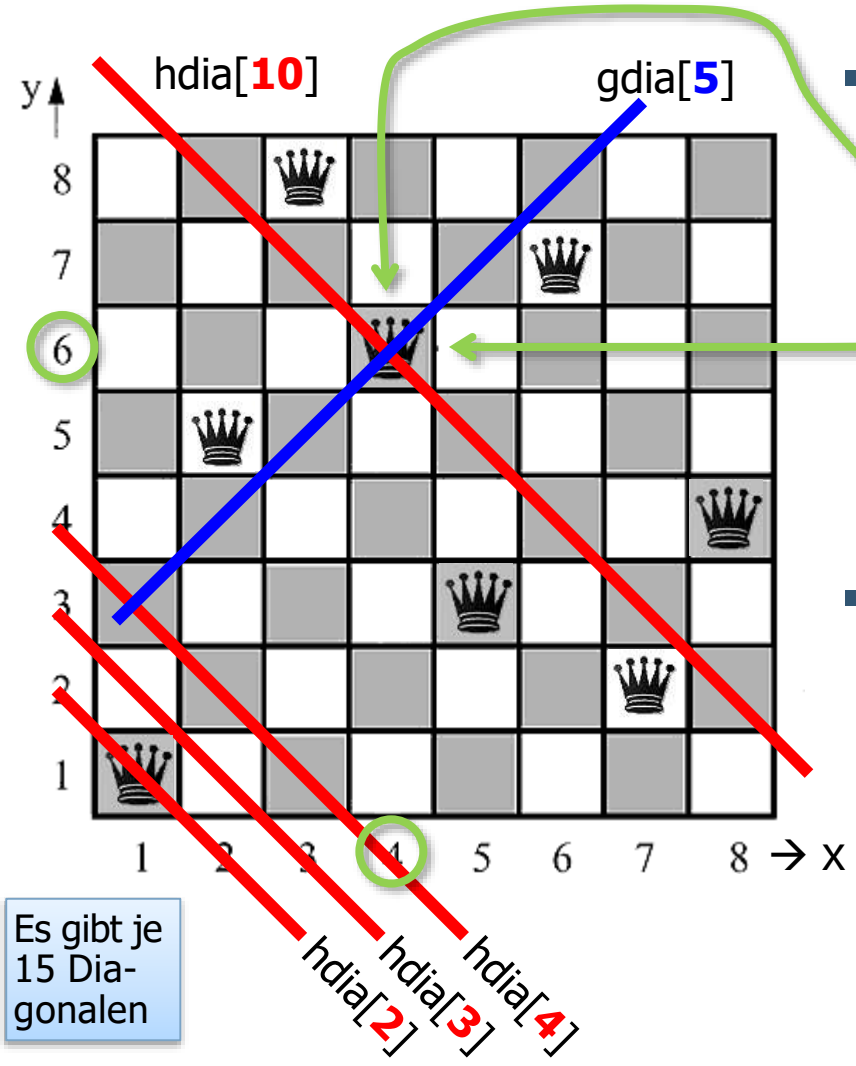
Emil Pauls erwähnt den 1862/63 publizierten „Traité des applications de l'analyse mathématique au jeu des échecs, précédé d'une introduction à l'usage des lecteurs soit étrangers aux échecs, soit peu versés dans l'analyse" des russischen Schachmeisters und -theoretikers Carl Ferdinand v. Jänisch (1813 – 1872). Seine Problemformulierung im nachfolgenden Textauszug ist für die Konstruktion der Hilfsgrössen *hdia* und *gdia* der nächsten Slide interessant:

si l'on exige que huit *dames*, dont chacune supposée *prenable*, occupent l'échiquier de manière qu'aucune ne se trouve en prise. Cette question, considérée au point de vue *mathématique*, est tout-à-fait du même genre que le *problème du cavalier* (voyez les n^os^ 80 et 91 du Livre II), quoique plus simple, ce qui la rend très-digne de l'attention des géomètres. Il s'agit uniquement de répartir les huit premiers nombres naturels (*sans répétition*) aux places laissées vides dans les parenthèses

(1,), (2,), (3,), (4,), (5,), (6,), (7,), (8,),

mais de telle façon que le nouveau chiffre *ajouté* à l'ancien fournisse, pour chaque parenthèse, une somme *différente*, et, qu'en même temps, le nouveau chiffre *déduit* de l'ancien laisse, chaque fois, un reste *différent*.

Repräsentation der Spielsituation in Java



Es gibt je 15 Diagonalen

- Darstellung der Spielsituation durch ein (globales) int-Array dame[0..n]:

x	1	2	3	4	5	6	7	8	← Spalte x
dame[x]	1	5	8	6	3	7	2	4	← y-Koordinate

Man braucht also gar kein (aufwendigeres) 2-dimensionales Array für den Spielzustand!

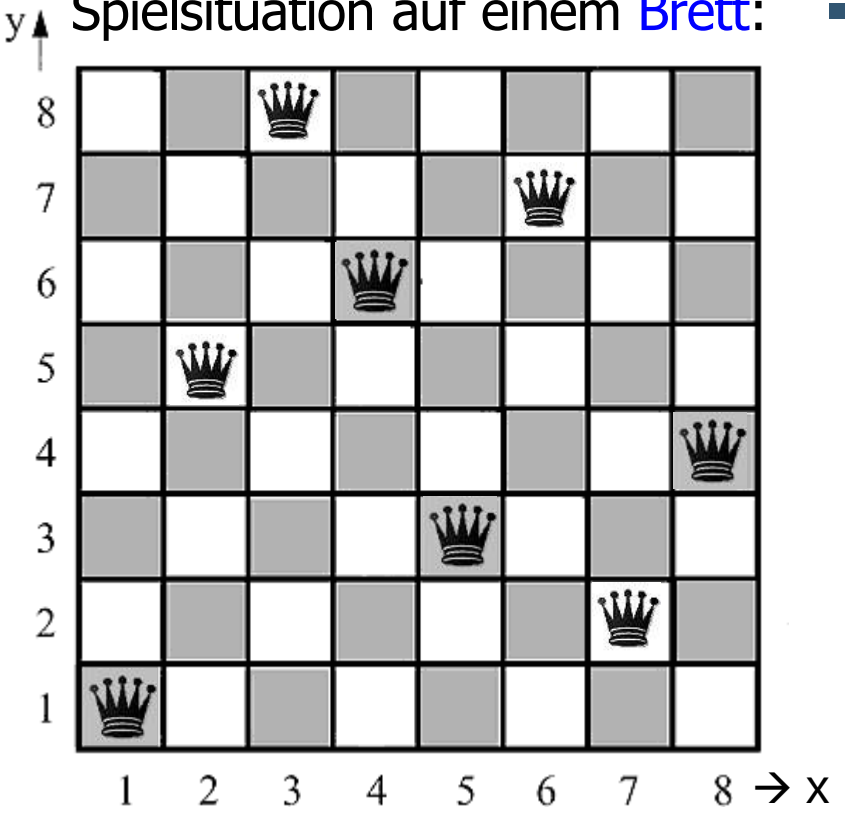
- Zweckmässig sind ferner 3 aus der Spielsituation „abgeleitete Grössen" in Form (globaler) boolean-Arrays:
  - zeile[y] == true: Zeile y ist bedroht
  - hdia[k] == true: Die Hauptdiagonale mit x+y=k ist bedroht (k=2,…,16)
  - gdia[k] == true: Die Gegendiagonale mit x-y+7=k ist bedroht (k=0,…,14)

∀ Damen gilt: Ihre jew. x- und y-Koordinaten, Haupt- und Gegendiagonalnummern müssen eindeutig sein

Und, nein: Wir brauchen kein Array „spalte[]" als symmetrisches Gegenstück zu „zeile[]"!

Repräsentation der Spielsituation in Java

Spielsituation auf einem Brett:



- Darstellung der Spielsituation durch ein (globales) int-Array dame[0..n]:

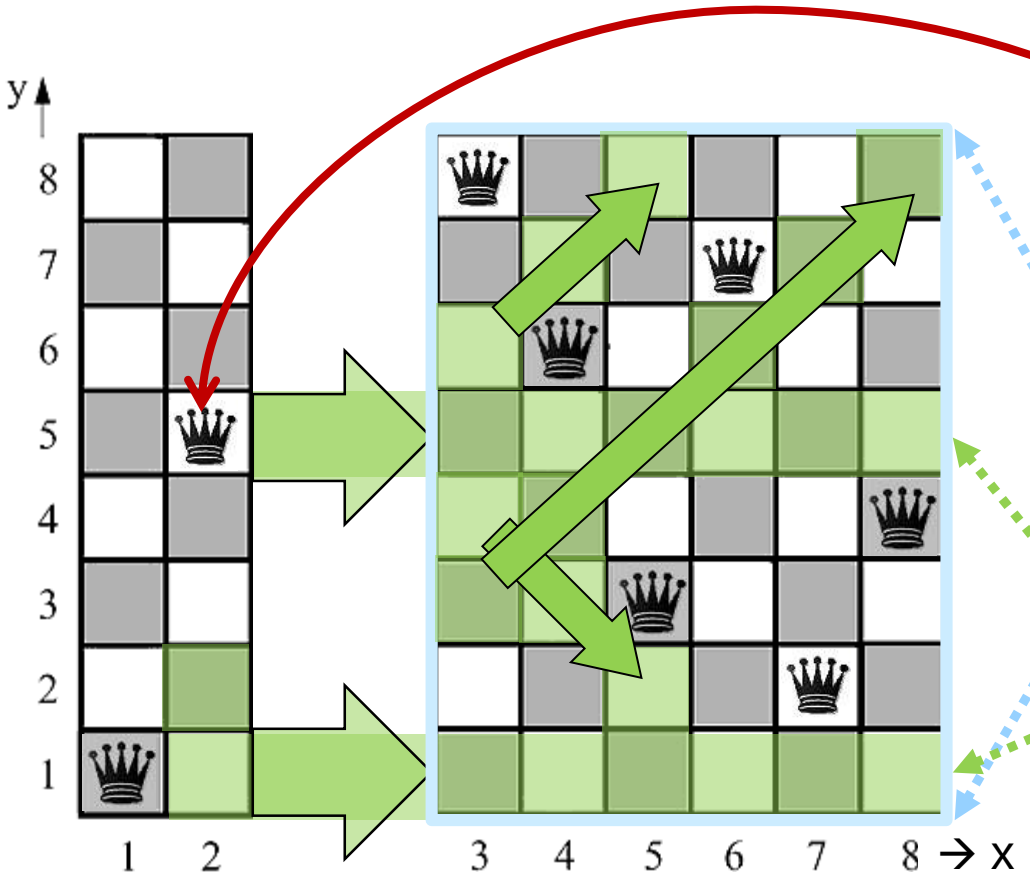
x	1	2	3	4	5	6	7	8	← Spalte x
dame[x]	1	5	8	6	3	7	2	4	← y-Koordinate

Beides sind nur verschiedene Repräsentationen der gleichen abstrakten Situation!

Man wähle die für den jeweiligen Zweck adäquate!

- Die beiden Darstellungen bedeuten das gleiche und sind im Prinzip gleichwertig (aber je nach Zweck unterschiedlich gut geeignet)
  - Man muss sie jeweils richtig deuten bzw. interpretieren
  - Sie lassen sich systematisch ineinander umwandeln

Eine Lösung mit Backtracking – die Idee



Ansatz:

Setze eine (einzige) Dame in Spalte x an geeignete Position.

Löse dann das Problem rekursiv für das Brett aus den Spalten x+1,..., n.

Dabei alle Bedrohungen der Damen aus Spalten 1,...,x berücksichtigen!

Empfehlung: Man lese hierzu den nett geschriebenen Beitrag von E.W. Dijkstra "Notes on Structured Programming", hier insbesondere das 17. Teilkapitel "*The problem of the eight queens*" auf den Seiten 72-82 in: O.J. Dahl, E.W. Dijkstra, C.A.R. Hoare (Eds.), *Structured Programming*. Academic Press Ltd., London, UK, 1972; online frei zugänglich bei http://dl.acm.org/citation.cfm?id=1243380; siehe auch E.W. Dijkstra: EWD 316 - A Short Introduction to the Art of Programming (9. The problem of eight queens), Aug. 1971, www.cs.utexas.edu/users/EWD/transcriptions/EWD03xx/EWD316.9.html

Eine Lösung mit Backtracking – das Programm

`setze(1)`

```
static void setze(int x) { // Spalte Nummer x
  for (int y=1; y<=8; y++) // von unten nach oben
    if (!(zeile[y] || hdia[x+y] || gdia[x-y+7])) {
      dame[x] = y;
      zeile[y] = true;
      hdia[x+y] = true;
      gdia[x-y+7] = true;
      if (x<8)
        setze(x+1);
      else { // x == 8
        for (int i=1; i<=8; i++)
          System.out.print(dame[i]);
        System.out.println();
      }
      zeile[y] = false;
      hdia[x+y] = false;
      gdia[x-y+7] = false;
    }
}
```

Prüfe, ob Position (x,y) bedroht ist

Hier wird der neue Zustand (inklusive neuer Bedrohungen) global gesetzt

Rekursiver Aufruf! (Nächste Spalte x+1)

Ausgabe einer Lösung bei x=8

Hier wird der alte Zustand (bezüglich der Bedrohungen) wieder restauriert

Wieso wird dame[x] nicht zurückgesetzt?

Backtracking durch Verlassen des rekursiven Aufrufs (nach Ende der for-Schleife)

Ansatz: Setze eine (einzige) Dame in Spalte x. Löse dann das Problem rekursiv für das Brett aus den Spalten x+1,..., n. Dabei alle Bedrohungen der Damen aus Spalten 1,...,x berücksichtigen!

Beachte: y ist eine lokale Variable jeder Methodeninstanz einer Spalte; die Zustandsvariablen zeile, hdia und gdia seien global (in der Klasse) definiert.

Eine Lösung mit Backtracking – das Resultat

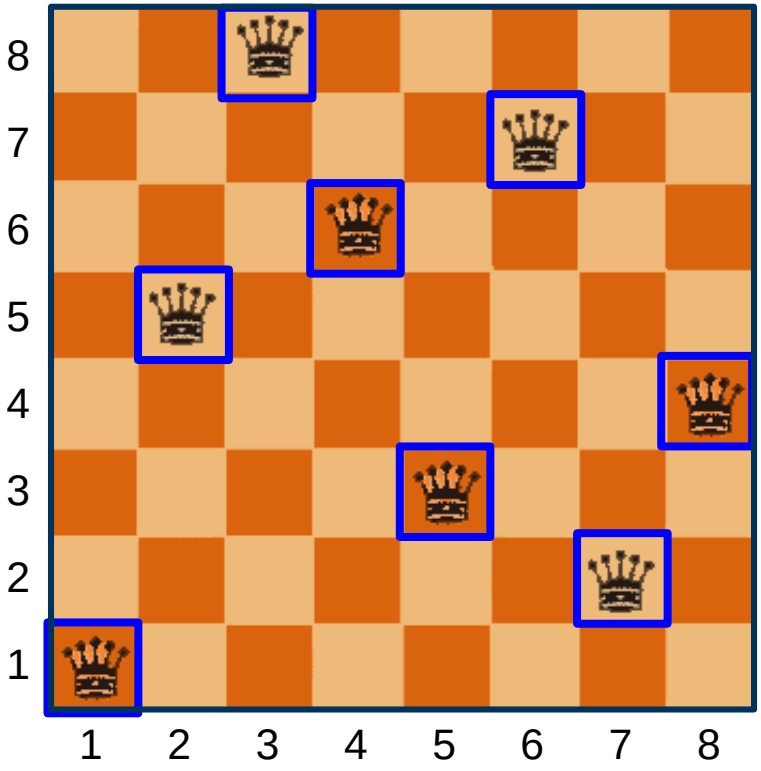
Der Aufruf erfolgt in main mit setze(1); dies erzeugt in Nullkommanix 92 Lösungen (davon 80 „symmetrische“):

1) 15863724
2) 16837425
3) 17468253
4) 17582463
5) 24683175
...
92) 84136275

Ist das eigentlich eine monoton wachsende Zahlenfolge?

Wie wäre es, wenn wir einfach mit Zahlen zur Basis 8 (und den Ziffern 1...8 statt 0...7) *zählen* würden und jede Zahl prüfen würden, ob sie eine Interpretation einer gültigen Stellung ist?

→ Positionen (1,1), (2,5), (3,8), (4,6), (5,3), (6,7), (7,2), (8,4)



n	Q(n) [Anzahl Lösungen]
1	1
2	0
3	0
4	2
5	10
6	4
7	40
8	**92**
9	352
10	724
11	2680
12	14200
13	73712
14	365596
15	2279184
16	14772512
...	...
27	234907967154122528

A priori ist gar nicht klar, ob Q(n) > 0 ist für alle n > 3

Alle 92 Lösungen (für n = 8) von Édouard Lucas

Tableau des 92 solutions du problème des huit reines.

1	1586 3724	24	3681 5724	47	5146 8273	70	6318 5247
2	1683 7425	25	3682 4175	48	5184 2736	71	6357 1428
3	1746 8253	26	3728 5146	49	5186 3724	72	6358 1427
4	1758 2463	27	3728 6415	50	5246 8317	73	6372 4815
5	2468 3175	28	3847 1625	51	5247 3861	74	6372 8514
6	2571 3864	29	4158 2736	52	5261 7483	75	6374 1825
7	2574 1863	30	4158 6372	53	5281 4736	76	6415 8273
8	2617 4835	31	4258 6137	54	5316 8247	77	6428 5713
9	2683 1475	32	4273 6815	55	5317 2864	78	6471 3528
10	2736 8514	33	4273 6851	56	5384 7162	79	6471 8253
11	2758 1463	34	4275 1863	57	5713 8642	80	6824 1753
12	2861 3574	35	4285 7136	58	5714 2863	81	7138 6425
13	3175 8246	36	4286 1357	59	5724 8136	82	7241 8536
14	3528 1746	37	4615 2837	60	5726 3148	83	7263 1485
15	3528 6471	38	4682 7135	61	5726 3184	84	7316 8524
16	3571 4286	39	4683 1752	62	5741 3862	85	7382 5164
17	3584 1726	40	4718 5263	63	5841 3627	86	7425 8136
18	3625 8174	41	4738 2516	64	5841 7263	87	7428 6135
19	3627 1485	42	4752 6138	65	6152 8374	88	7531 6824
20	3627 5184	43	4753 1682	66	6271 3584	89	8241 7536
21	3641 8572	44	4813 6275	67	6271 4853	90	8253 1746
22	3642 8571	45	4815 7263	68	6317 5824	91	8316 2574
23	3681 4752	46	4853 1726	69	6318 4275	92	8413 6275

On peut construire ce tableau par un procédé systématique, dont l'application est très simple et qui a été imaginé par Gauss, puis retrouvé par M. Laquière, en 1881. On place d'abord une reine dans la case la moins élevée de la première colonne à gauche ; on place ensuite une seconde reine dans la seconde colonne, sur la case la moins élevée qu'il soit possible, et ainsi de suite, en cherchant toujours à placer une reine dans une nouvelle colonne à droite, le plus bas qu'il soit possible, d'après les conditions du problème, c'est-à-dire en ayant égard aux positions des reines déjà placées à gauche. Lorsqu'il arrive un moment ou l'on ne peut plus placer aucune reine dans sa colonne, on élève celle de la colonne précédente de une, deux,..., cases, et l'on continue toujours, d'après le même principe, de n'élever une reine que lorsqu'il n'y a plus de positions admissibles pour l'ensemble des reines à placer à la droite. Chaque fois qu'une solution est trouvée, on l'inscrit d'après la notation convenue, et les solutions se trouvent ainsi rangées dans l'ordre numérique de la notation.

En suivant cette méthode, M. Laquière a fait effectuer par un enfant, dans une après-midi, le tableau des 92 solutions de l'échiquier de 64 cases. Ce tableau, facile à vérifier, ne contenait que trois erreurs provenant d'une seule omission et de deux solutions inexactes.

Aus dem Buch „Récréations mathématiques, vol. 1", von Édouard Lucas, erschienen 1882.

2D-Prettyprinting

Das Spielbrett kann auch 2-dimensional ausgeben werden, z.B. indem die zwei Zeilen

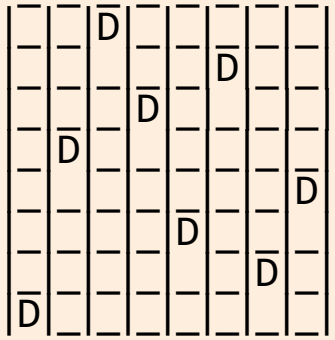
```
for (int i=1; i<=8; i++)
          System.out.print(dame[i]);
```

ersetzt werden durch:

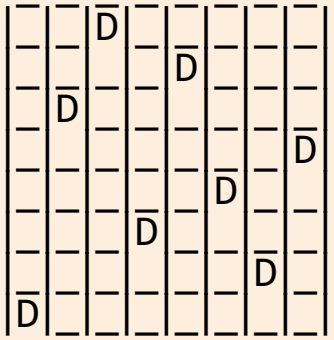
```
System.out.println("Lösung "+(++z)+":");
System.out.println(" _ _ _ _ _ _ _ _");
for (int i=8; i>=1; i--) {
  for (int j=1; j<=8; j++) {
    if (dame[j]==i) System.out.print("|D");
               else System.out.print("|_");
  }
  System.out.println("|");
}
```

Wobei z mit „static int z = 0;“ global deklariert sein sollte. Rechts ein Teil der Ausgabe. Für noch hübschere Ausgaben könnte man die Zellen hell bzw. dunkel färben und die Unicode-Zeichen ♕, ♛ (U+2655, U+265B) für die Dame benutzen.

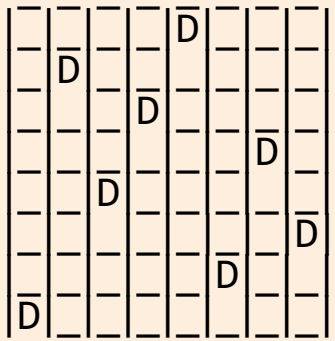
Lösung 1:



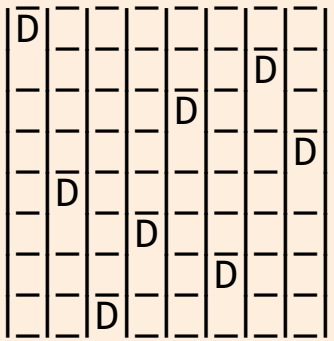
Lösung 2:



Lösung 3:



Lösung 92:



Eine Variante – Zählen bei beliebiger Zahlenbasis

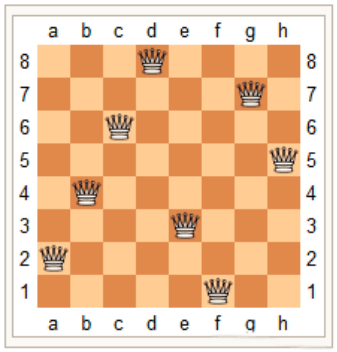
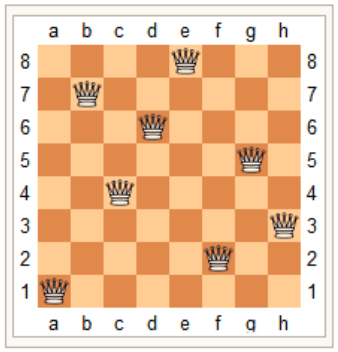
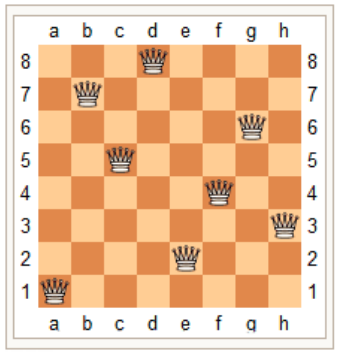
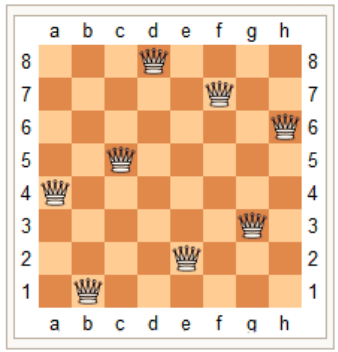
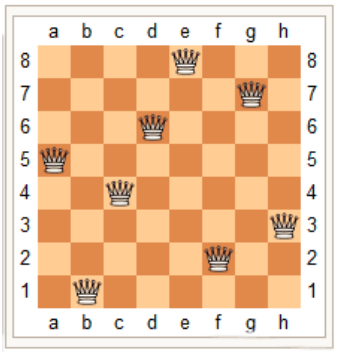
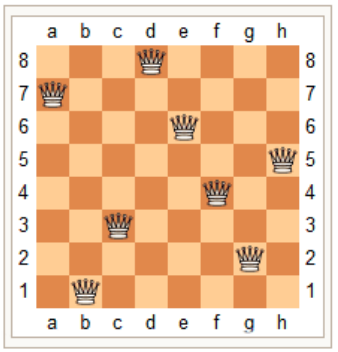
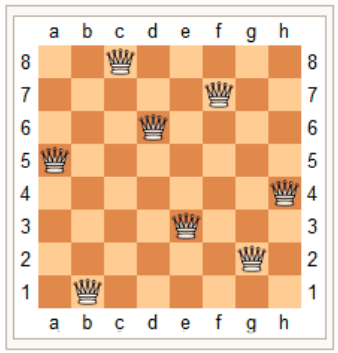
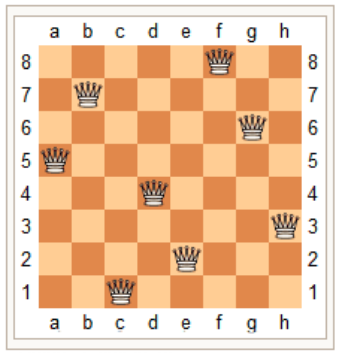
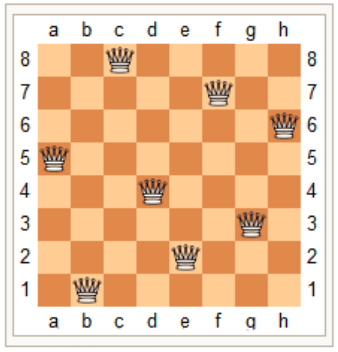
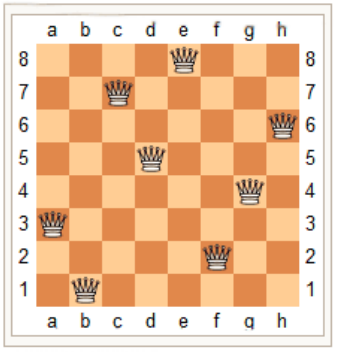
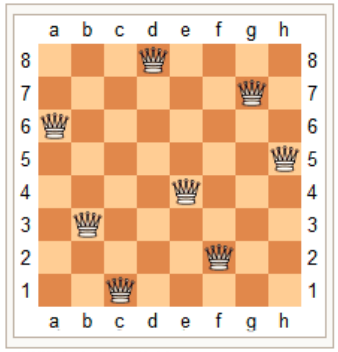
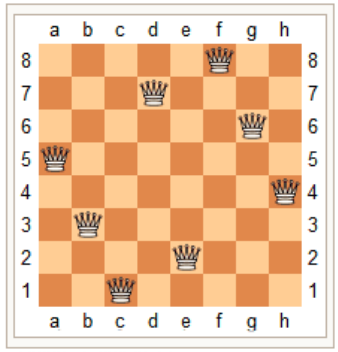
```
static void setze(int x) {
  for (int y=0; y < basis; y++)
    dame[x] = y;
    if (x < stellen)
      setze(x+1);
    else {
      for (int i=1; i<=stellen; i++)
        if (dame[i] < 10)
          System.out.print(dame[i]);
        else System.out.print((char)
          (dame[i]-10+(int)'A'));
      System.out.println();
    }
}
```

```
00000
00001
00010
00011
00100
00101
00110
00111
01000
01001
01010
01011
01100
01101
01110
01111
10000
10001
10010
10011
10100
10101
10110
10111
11000
11001
11010
11011
11100
11101
11110
11111
```

```
00 13
01 14
02 15
03 16
04 17
05 18
06 19
07 1A
08 1B
09 1C
0A ...
0B F8
0C F9
0D FA
0E FB
0F FC
10 FD
11 FE
12 FF
```

Man vergleiche dies mit dem ursprünglichen Programm: „Bedrohungen“ gibt es hier nicht mehr, daher fällt die if-Abfrage am Anfang der äusseren for-Schleife (und alles was an Variablen dazugehört) weg. Statt nur Werte zwischen 1 und 8 zu berücksichtigen, sind wir hier flexibler und erlauben alles zwischen 0 und basis-1. Und anstelle eines Arrays mit 8 Plätzen (und dem ungenutzten 0. Platz) sind wir hier auch flexibler; das Array wird (ausserhalb der Methode „setze“) mit „int [] dame = new int [stellen+1]” gegründet; entsprechend läuft hier der Index i von 1 bis „stellen”. Schliesslich sorgen wir dafür, dass ein Wert in „dame“, der grösser als 9 ist, mit einer „Buchstabenziffer“ A, B, C etc. ausgegeben wird. Als Test verwenden wir basis = 2, stellen = 5 sowie basis = 16, stellen = 2.

Die 12 Fundamentallösungen für n = 8
(die anderen 80 sind dreh- / spiegelsymmetrisch dazu)

Wie viele Damen stehen jeweils auf einem schwarzen Feld; wie viele auf einem weissen? Ist das Zufall?

Das vollständige Programm mit Demo-Ausgabe

```
public class NQueensBacktracking {                                    1
   static int n = 8;
   static int[ ] dame = new int[n+1];
   static boolean[ ] zeile = new boolean[n+1];
   static boolean[ ] hdia = new boolean[2*n+1];
   static boolean[ ] gdia = new boolean[2*n+1];
   static boolean demo = true;
   public static void main(String[ ] args)  { setze(1); }  2
   // Hier die anderen Methoden 2, 3, 4
}
```

```
static void sleepABit() {                                             3
   try {
      Thread.sleep(1000);
   }  catch (InterruptedException e)
         { e.printStackTrace(); }
}
```

```
static void printBoard(int spalte) {                                  4
   for (int i = n; i > 0; i--) {
      for (int j = 1; j <= n; j++) {
         String symbol = "_ ";
         if (zeile[i]  ||  hdia[i+j]  ||  gdia[j-i+(n-1)]  ||  j <= spalte)
             symbol = "x ";
         if (dame[j] == i) symbol = "D ";
         System.out.print(symbol);
      }
      System.out.println();
   }
      System.out.println();
}
```

```
static void setze(int x) {                                            2
   for (int y=1; y<=n; y++) {
      if (!(zeile[y] || hdia[x+y] || gdia[x-y+(n-1)])) {

         // Setzen der Dame und der bedrohten Felder
         dame[x] = y;  zeile[y] = true;
         hdia[x+y] = true;  gdia[x-y+(n-1)] = true;

         if (demo) {
            sleepABit();  3
            System.out.println("Dame gesetzt auf (" + x + "," + y + ")");
            printBoard(x);  4
         }

         if (x<n) {   // Noch freie Spalten - Rekursion
            setze(x+1);
         } else {   // Alle Spalten besetzt - Lösung gefunden
            System.out.print("Lösung gefunden: ");
            for (int i=1; i<=n; i++) System.out.print(dame[i]);
            System.out.println();
         }   // Backtracking - Freigabe bedrohter Felder
         dame[x] = 0;  zeile[y] = false;
         hdia[x+y] = false;  gdia[x-y+(n-1)] = false;

         if (demo) {
            sleepABit();  3
            System.out.println("Dame entfernt von (" + x + "," + y + ")");
            printBoard(x-1);  4
         }
      }
   }
}
```

Backtrack-Beispiel für n = 4

```
   1 2 3 4
4  _ _ _ _
3  _ _ _ _
2  _ _ _ _
1  _ _ _ _
```

Dame gesetzt auf (1,1)

```
   1 2 3 4
4  x _ _ x
3  x _ x _
2  x x _ _
1  D x x x
```

Dame gesetzt auf (2,3)

```
   1 2 3 4
4  x x x x
3  x D x x
2  x x x _
1  D x x x
```

Dame entfernt von (2,3)

```
   1 2 3 4
4  x _ _ x
3  x _ x _
2  x x _ _
1  D x x x
```

Dame gesetzt auf (2,4)

```
   1 2 3 4
4  x D x x
3  x x x _
2  x x _ x
1  D x x x
```

Dame gesetzt auf (3,2)

```
   1 2 3 4
4  x D x x
3  x x x x
2  x x D x
1  D x x x
```

Dame entfernt von (3,2)

```
   1 2 3 4
4  x D x x
3  x x x _
2  x x _ x
1  D x x x
```

Dame entfernt von (2,4)

```
   1 2 3 4
4  x _ _ x
3  x _ x _
2  x x _ _
1  D x x x
```

Dame entfernt von (1,1)

```
   1 2 3 4
4  _ _ _ _
3  _ _ _ _
2  _ _ _ _
1  _ _ _ _
```

Dame gesetzt auf (1,2)

```
   1 2 3 4
4  x _ x _
3  x x _ _
2  D x x x
1  x x _ _
```

Dame gesetzt auf (2,4)

```
   1 2 3 4
4  x D x x
3  x x x _
2  D x x x
1  x x _ _
```

Dame gesetzt auf (3,1)

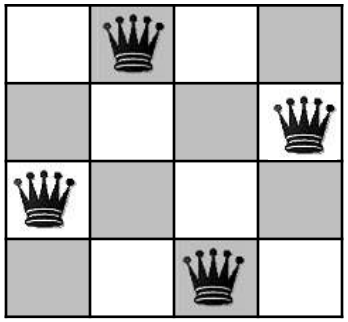
```
   1 2 3 4
4  x D x x
3  x x x _
2  D x x x
1  x x D x
```

Backtrack-Beispiel für n=4

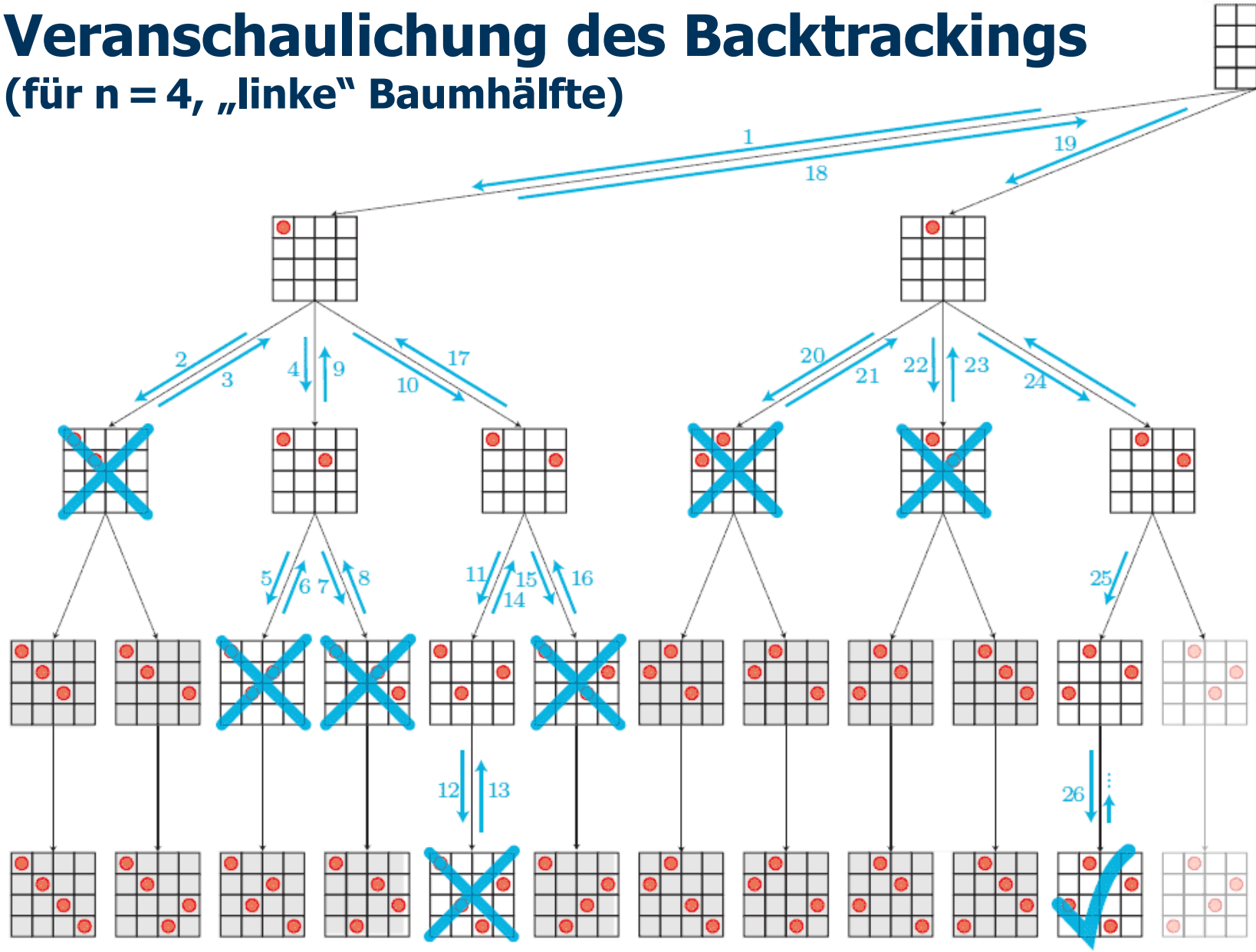
Dame gesetzt auf (4,3)

	1	2	3	4
4	x	D	x	x
3	x	x	x	D
2	D	x	x	x
1	x	x	D	x

Lösung gefunden: 2413

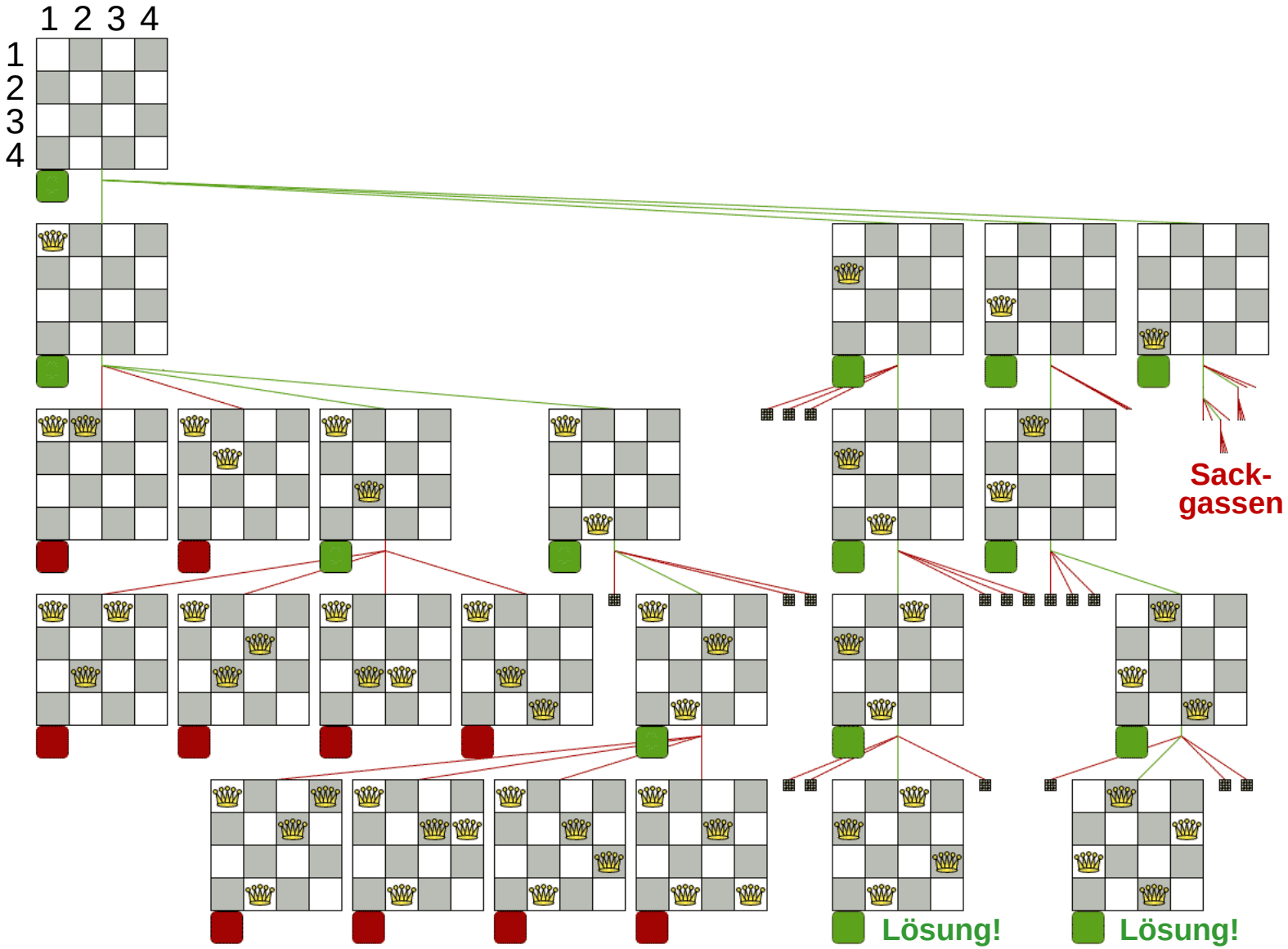


Veranschaulichung des Backtrackings
(für n = 4, „linke“ Baumhälfte)

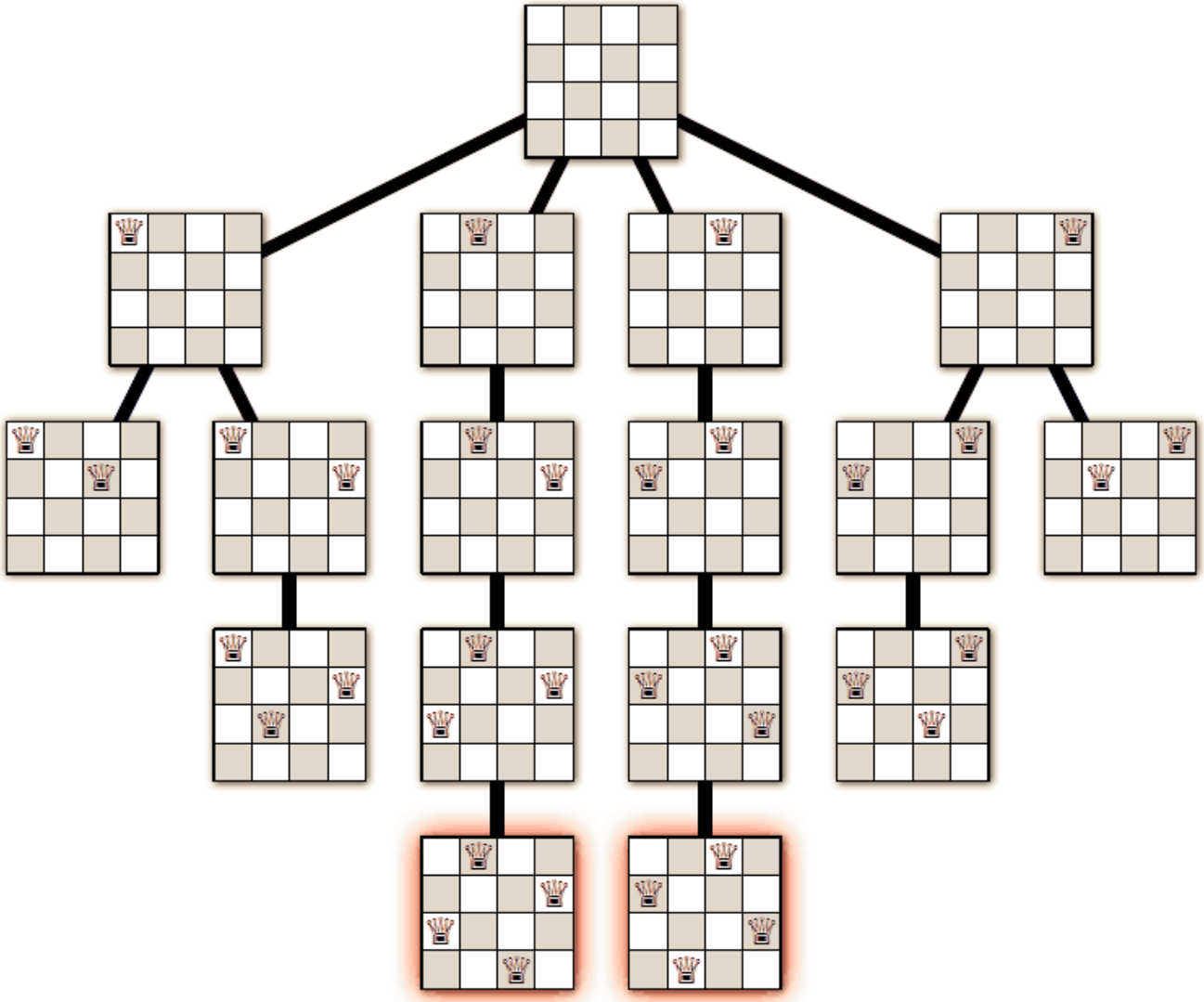


www.educ.ethz.ch/unterrichtsmaterialien/informatik/baeume-und-backtracking.html

Der Backtrack-Baum in alternativer Darstellung



Der Backtrack-Baum verkürzt



Lässt man die „illegalen“ Zustände von vornherein weg, dann entsteht ein entschlackter und übersichtlicherer Baum

http://jeffe.cs.illinois.edu/teaching/algorithms/book/02-backtracking.pdf

Lösungen für n = 1,..., 18 in Fortran

FORmula TRANslation; Ende der 1950er-Jahre von IBM konzipiert, war die Sprache im wissenschaftlich-technischen Bereich weit verbreitet

- So hatte man noch in den 1970er-Jahren programmiert: Programme in Fortran auf Lochkarten gestanzt; den Lochkartenstapel im Rechenzentrum abgegeben; Ausgabe (Resultat oder Compiler-Fehlermeldung) Stunden später auf Zebrapapier
- Dies hier, Fortran 77, war seinerzeit sehr modern (implicit-Statement, print-Statement!); ältere Dialekte, wie FORTRAN IV, waren nicht so bequem

```
C N QEENS WITH BACKTRACKING.
C AS IS, THE PROGRAM ONLY
C PRINTS THE NUMBER OF N
C QUEENS CONFIGURATIONS.
C TO PRINT ALSO THE
C CONFIGURATIONS, UNCOMMENT
C THE LINE AFTER LABEL 80.
C
PROGRAM QUEENS
IMPLICIT INTEGER(A-Z)
PARAMETER(L=18)
DIMENSION A(L),S(L),U(4*L-2)
DO 10 I=1,L
10 A(I)=I
DO 20 I=1,4*L-2
20 U(I)=0
DO 110 N=1,L
M=0
I=1
R=2*N-1
GO TO 40
30 S(I)=J
U(P)=1
U(Q+R)=1
I=I+1
40 IF(I.GT.N) GO TO 80
J=I
50 Z=A(I)
Y=A(J)
P=I-Y+N
Q=I+Y-1
A(I)=Y
A(J)=Z
IF((U(P).EQ.0).AND.(U(Q+R).EQ.0)) GO TO 30
60 J=J+1
IF(J.LE.N) GO TO 50
70 J=J-1
IF(J.EQ.I) GO TO 90
Z=A(I)
A(I)=A(J)
A(J)=Z
GO TO 70
80 M=M+1
C PRINT *,(A(K),K=1,N)
90 I=I-1
IF(I.EQ.0) GO TO 100
P=I-A(I)+N
Q=I+A(I)-1
J=S(I)
U(P)=0
U(Q+R)=0
GO TO 60
100 PRINT *,N,M
110 CONTINUE
END
```

Q: Why were early programming languages so "SHOUTY"? Did keyboards have caps-lock on by default? A: There was no lower case. You wrote your program on punched cards and the key-punch only provided upper case. Likewise, the printers only provided upper case. [www.quora.com]

Lösungen für n = 1,..., 18 in Fortran

IBM

FORTRAN Coding Form

X28-7327-5
Printed in U.S.A.

PROGRAM

PROGRAMMER DATE

STATEMENT NUMBER | CONT | FORTRAN STAT

```
C       *** N QEENS/BACKTRACKING ***
        PROGRAM QUEENS
        IMPLICIT INTEGER(A-Z)
        PARAMETER(L=18)
        DIMENSION A(L),S(L),U(4*L-2)
        DO 10 I=1,L
   10   A(I)=I
        DO 20 I=1,4*L-2
   20   U(I)=0
        DO 110 N=1,L
        M=0
        I=1
        R=2*N-1
        GO TO 40
   30   S(I)=J
        U(P)=1
        U(Q+R)=1
        I=I+1
   40   IF(I.GT.N) GO TO 80
        J=I
   50   Z=A(I)
        Y=A(J)
        P=I-Y+N
        Q=I+Y-1
```

* A standard card form, IBM electro 888167, is available for punching statements from this form

Als es noch keine interaktiven Computerterminals oder gar PCs gab, also bis in die 1970er-Jahre, wurde ein Programm offline auf Lochkarten aus 0.17mm-dünnem flexiblem Karton gebracht, die dann stapelweise an einer Lochkartenlesestation im Rechenzentrum eingelesen wurden. Das Erstellen der Lochkarten zu einem Programm geschah allerdings oft nicht durch die Programmierer selbst, sondern durch sogenannte „Locherinnen" – ein typisch weiblicher Anlernberuf – mittels schreibmaschinenähnlicher Lochkartenstanzer. Die Programmierer füllten dazu per Hand vorgedruckte „Ablochformulare" (coding forms) aus; jede Zeile des Formulars entsprach dabei einer zu lochenden Karte; jede Spalte

einer Karte codierte (durch mehrere rechteckige Löcher in dieser Spalte) ein bestimmtes Zeichen – Grossbuchstaben, Ziffern und einige wenige Sonderzeichen umfasste der Zeichensatz. Der Lochcode konnte in der Lesestation beim Einlesen des Kartenstapels durch elektrische Kontakte oder Photozellen entschlüsselt werden, so dass ein Programm mit 10 bis 20 Lochkarten pro Sekunde zeichenweise in den Speicher des „Grossrechners" geladen wurde.

Lochkarten waren übrigens keine Errungenschaft der ab den 1950er-Jahren aufkommenden „elektronischen" Datenverarbeitung (EDV) mit Digitalrechnern; als Datenträger wurden sie bereits seit dem Ende des 19. Jahrhunderts für die Verarbeitung von Daten mit mechanischen und elektromechanischen Auswertemaschinen (Sortier-, Stanz- und Tabelliermaschinen) benutzt, etwa zur Erstellung von Statistiken (Volkszählung) oder im Grosshandel sowie Banken- und Versicherungsbereich. Diese etablierte Gerätetechnik wurde mit dem Aufkommen der elektronischen Grossrechner als Peripheriegeräte für die Ein- und Ausgabe adaptiert.

Locherinnen zur Datenerfassung gab es somit schon länger, der Bedarf dieser Arbeitskräfte stieg allerdings mit der Verbreitung der Digitalrechner, einer ersten Digitalisierungswelle in der Wirtschaft Mitte des 20. Jahrhunderts, stark an. Ihre Arbeit war recht monoton, erforderte aber doch hohe Konzentration. Oft waren mehrere „Erfassungsplätze" in einem grösseren schallgedämpften Saal angeordnet. Dennoch galt die Arbeit in der EDV als modern und begehrenswert, die Locherinnen gaben dafür oft ihre Berufe in anderen Branchen als Verkäuferin, Serviererin oder Arbeiterin auf. Mit der Zeit wurden Lochkarten durch andere Datenträger ersetzt (Magnetbandkassetten oder Disketten – später kamen auch online-angebundene Arbeitsstationen mit „Datensichtgeräten" auf), womit sich auch die Berufsbezeichnung zu „Datentypistin" änderte. Tatsächlich ging es ab den 1970er-Jahren vor allem darum, die immer umfangreicher werdenden auf Papier vorliegenden Datenmengen zu erfassen; die Programme hingegen wurden von den Entwicklern zunehmend selbst an interaktiven Terminals eingegeben – meist wie bei einem Fernschreiber („teletype", daher noch heute die Abkürzung „tty") zeichen- und zeilenweise ohne gute Korrekturmöglichkeit; für sogenannte „full screen editors" bei Bildschirmarbeitsplätzen („glass terminals") genügten die Hardwareressourcen anfangs noch nicht. Ein leitender Angestellter einer Versicherung kommentierte 1975 den Bezeichnungswandel von der Locherin zur Datentypistin so: „Wie auch vor langer Zeit die Putzfrau den Sprung zur

Raumpflegerin schaffte, die Schreibkraft sich zur Steno- oder Phonotypistin mauserte, erstaunte es keinen, dass sich die Locherin zu einer Datentypistin entwickelt."

Aber auch die modernere Bezeichnung änderte nichts daran, dass der Beruf niedriger qualifiziert war als etwa Büroschreibkräfte, Stenotypistinnen oder gar Sekretärinnen, denn Vertrautheit mit der Orthographie und Grammatik der Schriftsprache war beim reinen Abtippen der Programmformulare oder Datenbelege kaum relevant.

Aus heutiger Sicht ist es allerdings irritierend, mit welchen Argumenten seinerzeit dieser Beruf als besonders geeignet für Frauen angesehen wurde. Auch Wissenschaftler reproduzierten dabei alte diskriminierende Vorurteile, wie die folgenden etwas peinlichen Textpassagen aus dem Buch „Industriebürokratie" des Soziologen und Göttinger Lehrstuhlinhabers Hans Paul Bahrdt (1918-1994) aus dem Jahr 1958 zeigen:

„Kontinuierlich folgt Lochkarte auf Lochkarte. Die inhaltliche Individualität der einzelnen Karte darf in der Regel nicht ins Bewusstsein eindringen, wenn die Arbeit flüssig vonstatten gehen soll. Die Locherin vergegenwärtigt sich gewöhnlich nicht, was sie locht. Habitualisierung und Verschlüsselung schirmen sie gegen die Gefahr des Mitdenkens ab. […]

Das notwendige Sicheinschmiegen in einen vorgegebenen Ablauf von Verrichtungen, ohne sich von ihm völlig einlullen zu lassen, gelingt offensichtlich Frauen sehr viel besser als Männern. Männer, die zu vergleichbaren Routinearbeiten herangezogen werden, haben viel eher die Tendenz, entweder abzustumpfen oder auszubrechen. Oder aber sie zwingen sich ständig zur Arbeit, was dann auf die Dauer zu Verkrampfung und zu nervlichen und psychischen Schaden führen kann. Deshalb ist Locharbeit wie kaum eine andere Arbeit für Frauen eine Tatsache, die seltsamerweise erst spät erkannt worden ist. […]

Und nichts wäre falscher, als wenn man in der Absicht, ein qualifiziertes Locherinnenteam heranzubilden, besonders intelligente Mädchen auswählen würde. Überdurchschnittlich intelligente Mädchen sind im allgemeinen zu nervös für diese Tätigkeit oder werden nervös, weil sie den Lärm und die Gleichförmigkeit der Tätigkeit nicht vertragen, vor allem aber, weil sie ständig in Versuchung geraten, sich etwas dabei zu denken, was der Zügigkeit der Arbeit, im Ganzen gesehen, abträglich ist."

Explicit Solutions to the N–Queens Problem for all N

Eine Lösung ohne Computer
(ACM SIGART Bulletin 2(1), 1991, S. 7)

Bo Bernhardsson
Department of Automatic Control
Land Institute of Technology
P.O. Box 118
S–22100 Lund SWEDEN
bob@control.lth.se

The n–queens problem is often used as a benchmark problem for AI research and in combinatorial optimization. An example is the recent article [1] in this magazine that presented a polynomial time algorithm for finding a solution. Several CPU–hours were spent finding solutions for some n up to 500,000.

I would like to draw the readers' attention to the less known fact that *explicit* solutions for this problem exist for all $n \geq 4$, see below. This result was published 1969 in [2]. The construction is quite simple and requires no combinatorial search or computer time whatsoever. To find one solution to the n–queens problem is therefore completely trivial. The n–Queens problem is therefore a bad benchmark problem. A lot of CPU–hours can be saved in the future noting this.

A verification of the construction below is given in [2]. Let (i, j) denote the square on row i and column j. Depending on n there are three cases:

(A) n even but not of the form $6k+2$: Place queens on elements

$$(j, 2j)$$
$$(n/2+j, 2j-1), \quad j = 1, 2, \ldots, n/2$$

(B) n even but not of the form $6k$: Place queens on elements

$$(j, 1 + [2(j-1) + n/2 - 1 \text{ modulo } n])$$
$$(n+1-j, n - [2(j-1) + n/2 - 1 \text{ modulo } n]),$$
$$j = 1, 2, \ldots n/2$$

(C) n odd: Use case A or case B on $n-1$ and extend with a queen at

$$(n, n)$$

It could finally be noted that the related problem of finding *all* solutions to the n–queens problem is non–trivial. It was introduced by Gauss with $n = 8$.

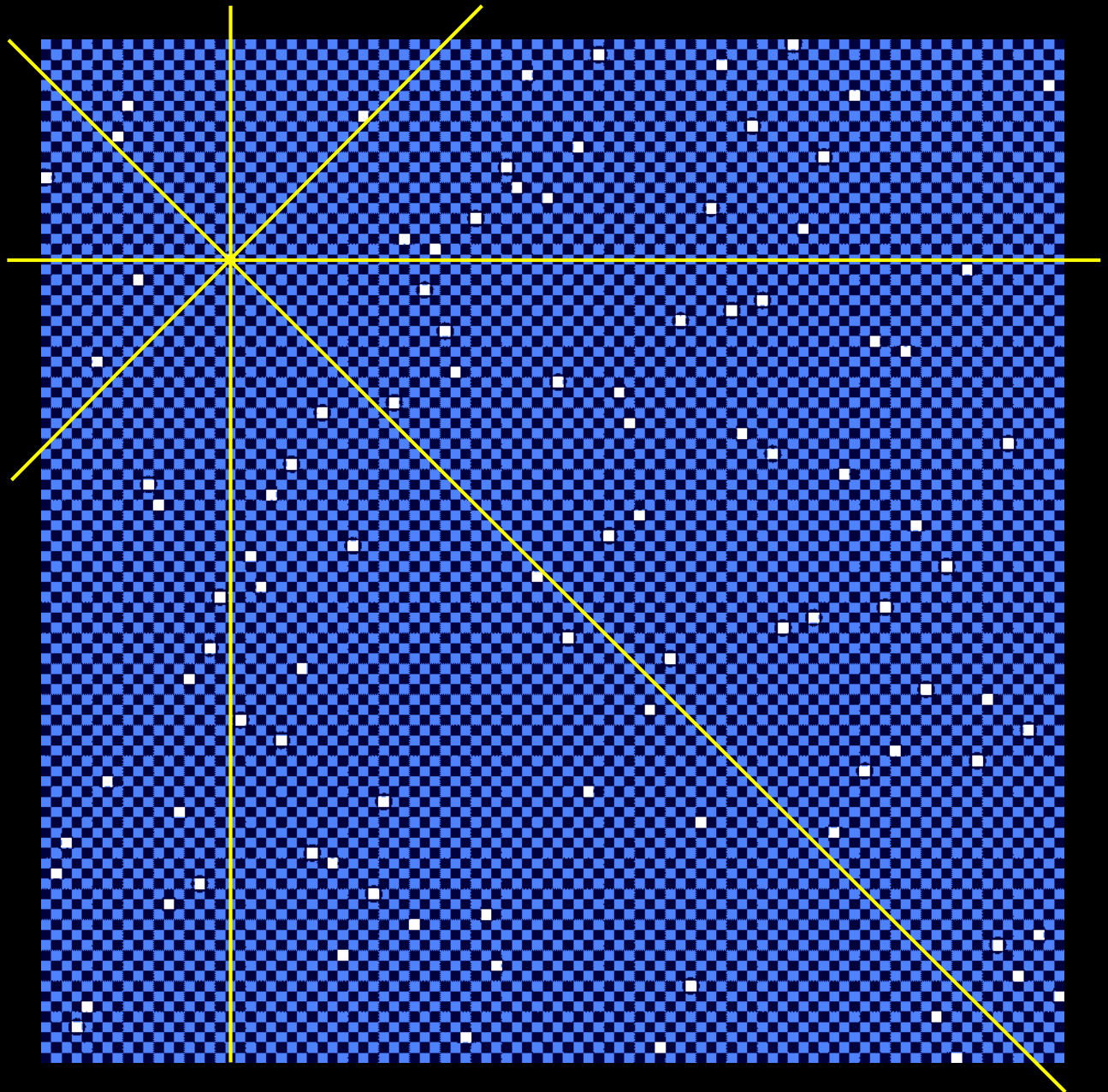
References

[1] Sosic, R. and Gu, J. (1990): A Polynomial Time Algorithm for the N-Queens Problem, *SIGART Bulletin*, 1, p. 7–11.

[2] Hoffman, E.J., Loessi, J.C. and Moore, R.C. (1969): Constructions for the Solution of the m Queens Problem, *Mathematics Magazine*, p. 66–72.

Nun ja: Eine (!) Lösung für das n-Damen-Problem lässt sich so finden, das ist nett. Eine Konstruktionsvorschrift für eine spezielle Lösung bei beliebigem n wurde allerdings schon 1874 von Emil Pauls angegeben. (*Das Maximalproblem der Damen auf dem Schachbrete.* Deutsche Schachzeitung 29(5), Mai 1874, S. 129–134; und 29(9), Sep. 1874, S. 257–267.) Wenn man alle Lösungen sucht, kommt man aber wohl um Backtracking nicht herum. Für viele andere kombinatorische Probleme ist Backtracking auch zum Suchen einer einzigen Lösung sowieso unverzichtbar.

100 konfliktfreie Damen in einer 100^2-Matrix



www.ronybc.com/img/n-queens-100-a.png

Backtracking formal

The author once waited all night for the output from a back-track program, only to discover that the answers would not be forthcoming for about 10^6 centuries. – Donald Knuth

Robert John Walker (1909 – 1992), der an der Cornell University die Etablierung der Informatik betrieb, hat 1960 das Backtracking-Prinzip in mathematischer Weise so beschrieben:

Many combinatorial problems involve the construction of one or more (possibly all) ordered sets $A = \{a_1, a_2, .. ., a_n\}$, where the a_i are elements of a finite set U and the elements of a set A are subject to certain restrictions. The most common example of such a set A is a permutation; here n is the number of elements in U and the restriction is that $a_i \neq a_j$ if $i \neq j$. More complicated examples are the following:

(i) *The problem of the queens.* Place eight queens on a chess board so that no two attack each other. Since there must be one queen in each file (column) let the queen in the ith file be on the a_ith rank (row). Then $U = \{1, 2,\ldots, 8\}$, $n = 8$, and the restrictions are

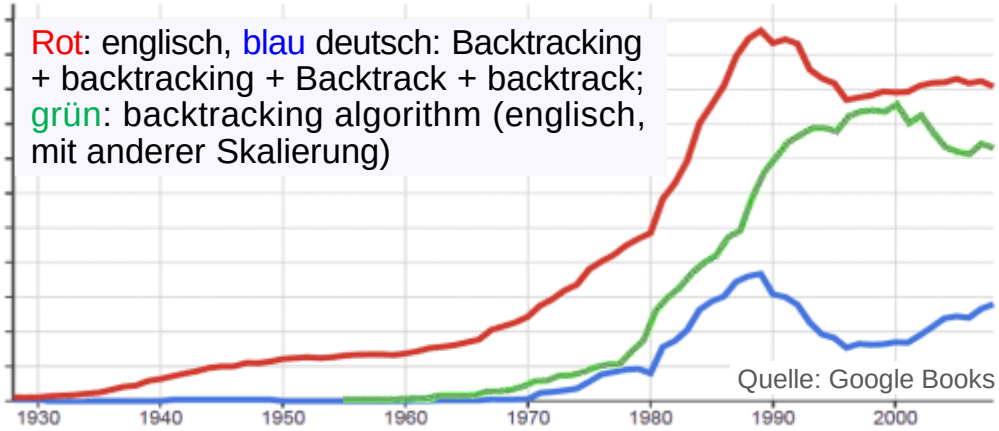
$a_i \neq a_j$ if $i \neq j$ $\quad a_i - i \neq a_j - j$ if $i \neq j$ $\quad a_i + i \neq a_j + j$ if $i \neq j$ […].

Nachfolgende Paraphrase von Walkers generellem Backtracking-Ansatz stammt von James Bitner und Edward Reingold:

Assume that the solution to a problem consists of a vector $(a_1, a_2,\ldots)$ of undetermined length. This vector satisfies certain constraints on the components, which makes it a solution. Each a_i is a member of a finite, linearly ordered set A_i. Thus the exhaustive search must consider the elements of $A_1 \times A_2 \times \ldots \times A_i$ for $i = 0, 1, 2,\ldots$ as potential solutions. Initially we start with the null vector as our partial solution, and the constraints tell us which of the members of A_1 are candidates for a_1; we call this subset S_1. We choose the least element of S_1 as a_1, and now we have the partial solution (a_1). In general, the various constraints which describe solutions tell us which subset S_k of A_k comprises candidates for the extension of the partial solution $(a_1, a_2, \ldots, a_{k-1})$ to $(a_1, a_2, \ldots, a_{k-1}, a_k)$. If the partial solution $(a_1, a_2, \ldots, a_{k-1})$ admits no possibilities for a_k, then $S_k = \emptyset$, so we backtrack and make a new choice for a_{k-1}. If there are no new choices for a_{k-1} we backtrack still further and make a new choice for a_{k-2}, and so on.

Backtracking – historische Anmerkungen

Backtracking als allgemeines Prinzip wurde nicht einfach irgendwann entdeckt; wiederholt ist es bei der Lösung konkreter Probleme quasi nebenbei „erfunden“ worden. Im Englischen tauchte vor seiner algorithmischen Bedeutung die Redewendung „to backtrack“ bereits um 1870 für „den gleichen Weg noch einmal zurückgehen“ auf. In den 1950er-Jahren soll der US-amerikanische Mathematiker Derrick Lehmer (1905 – 1991) den Begriff dann erstmalig auf Algorithmen bezogen haben. (Lehmer, der u.a. zusammen mit seiner Frau Emma 1945/46 Teil des Nutzer- und Bedienteams des ENIAC-Rechners war, leistete Bedeutendes in der Primzahltheorie, er entwickelte auch die ersten Pseudozufallszahlengeneratoren.) Wie oben ausgeführt, hatte R.J. Walker in seinem Aufsatz „An enumerative technique for a class of combinatorial problems“ das Backtracking 1960 als ein allgemeines Lösungsprinzip propagiert.



> *The characteristic property of these problems is that they defy analytic solutions. Instead, they require large amounts of exacting labor, patience, and accuracy. Such algorithms have therefore gained relevance almost exclusively through the automatic computer, which possesses these properties to a much higher degree than people, and even geniuses, do.*
> -- Niklaus Wirth

Zuvor wurde das Backtracking-Prinzip, ohne es so zu nennen, beispielsweise beim Labyrinth-Problem angewendet. Dabei wird ein Labyrinth als Graph aus Kanten und Knoten aufgefasst, und es geht im Sinne einer Graphtraversierung darum, ausgehend von einem Startknoten (der z.B. für den Labyrintheingang stehen kann) alle Knoten des Graphen zu besuchen (um so auf jeden Fall den Goldtopf im Inneren oder auch einen anderen Ausgang zu finden) und

Backtracking – historische Anmerkungen (2)

sicher wieder zum Startknoten zurückzugelangen – und zwar mit möglichst wenig Aufwand und ohne sich in Zyklen zu verlieren. Ein Verfahren dazu, das eine leichte Verallgemeinerung der Tiefensuche darstellt, wurde 1895 vom französischen Mathematiker und in Algerien lebenden Finanzinspekteur Gaston Tarry (1843–1913) vorgestellt. Tarry eröffnet seinen Aufsatz „Le problème des labyrinthes“ folgendermassen:

Tout labyrinthe peut être parcouru en une seule course, en passant deux fois en sens contraire par chacune des allées, sans qu'il soit nécessaire d'en connaître le plan. Pour résoudre ce problème, il suffit d'observer cette règle unique : *Ne reprendre l'allée initiale qui a conduit à un carrefour pour la première fois que lorsqu'on ne peut pas faire autrement.*

Diese einzige Regel von Tarry muss man etwas wohlwollend mit dem nötigen Kontext interpretieren; man findet heute in der Literatur etwas konkreter gefasste Regeln etwa der folgenden Art:

(1) Wenn man einen Gang betritt, markiere man den Eingang mit „stopp“. Man betrete nie einen Gang, der mit „stopp“ markiert ist. (2) Betritt man das erste Mal eine Kreuzung, markiere man den eben verlassenen Gang mit „zuletzt“. (3) Gibt es an einer Kreuzung Gänge, die keine Markierung besitzen, wähle man einen beliebigen davon. Sollte es keine unmarkierten Gänge mehr geben, betrete man den mit „zuletzt“ markierten Gang.

Gaston Tarry

Betritt man eine Kreuzung (an der mehr als drei Gänge münden) zum wiederholten Mal, so findet man dort evtl. mehrere unmarkierte Gänge vor. Die Depth-first-Strategie (Tiefensuche) ergibt sich aus dem Tarry-Algorithmus dadurch als Spezialisierung, dass in diesem

Backtracking – historische Anmerkungen (3)

Fall die Wahlfreiheit eingeschränkt wird durch eine weitere Regel: *(4) Falls man eine Kreuzung wiederholt betritt, kehre man sofort um, sofern dieser Rückweg nicht durch „stopp“ gesperrt ist.*

Tarry schliesst seinen Aufsatz mit einem fast moralisierenden Resümee:

En suivant cette marche pratique, un voyageur perdu dans un labyrinthe ou dans des catacombes, retrouvera forcément l’entrée avant d’avoir parcouru toutes les allées et sans passer plus de deux fois par la même allée. Ce qui démontre qu’un labyrinthe n’est jamais inextricable, et que le meilleur fil d’Ariane est le fil du raisonnement.

Wie weiter oben dargelegt, hatte allerdings C.F. Gauß bereits knappe 50 Jahre früher das Backtracking-Prinzip am Beispiel des 8-Damen-Problems skizzenhaft beschrieben. Er dürfte der erste gewesen sein. Sein Kollege Schumacher bedankte sich am 24. September 1850 übrigens artig: „Nehmen Sie, mein theuerster Freund, meinen besten Dank für die Belehrung über die 8 Damen...“

Resümee des Kapitels

- Backtracking
  - Systematisches Durchmustern eines Problembereichs
  - Relevante Prinzipien: Rekursion; Depth-first-Baumtraversierung

- Beispiele für Backtracking-Probleme:
  - Labyrinth
  - Puzzle
  - n-Damen-Problem