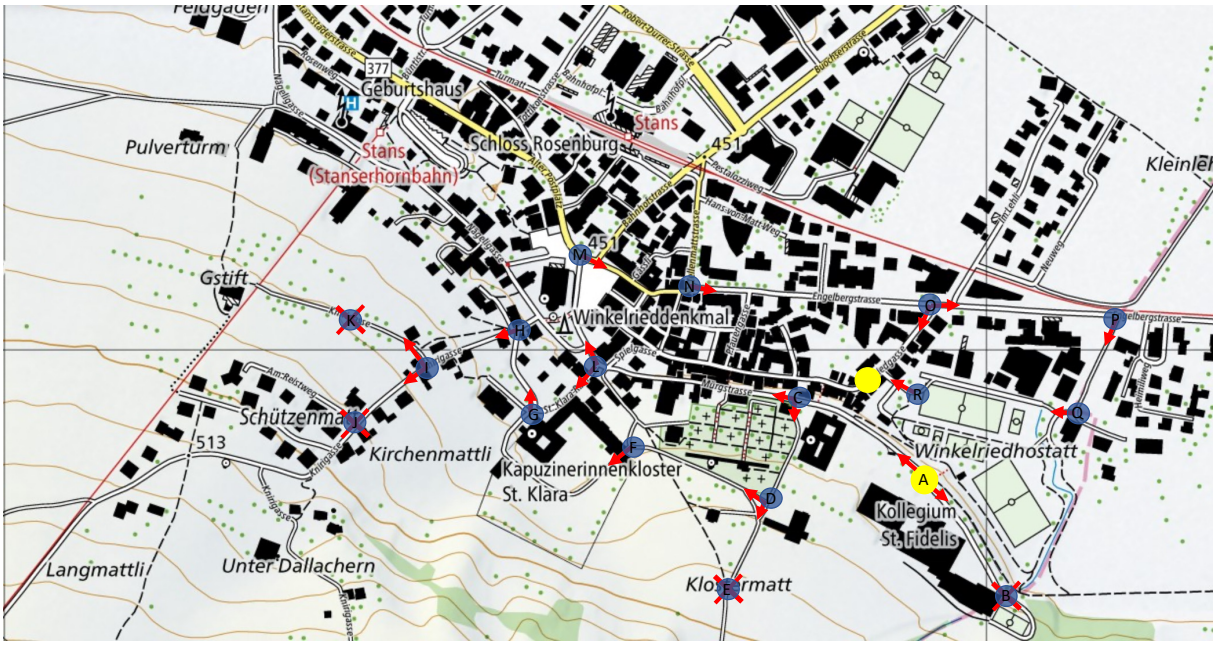
Algorithmen-Entwurf anhand des Gedankenspiels: SCHNITZELJAGD DURCH STANS



0 Einleitung

Schon seit jeher sind Algorithmen wichtige Instrumente der Automatisierung des Problemlösens. Die ersten bekannten Algorithmen wurden bereits 2500 v. Chr. von den Babyloniern angewandt und heutzutage bildet die Algorithmik und Algorithmen-Theorie einen der Grundpfeiler der Informatik. Dabei können Algorithmen als Berechnungsvorschrift für ein spezifisches Problem betrachtet werden. Es existieren so viele Algorithmen, wie es andersartige Probleme zu lösen gilt. Dennoch gibt es einige generelle Strategien zur Suche nach Algorithmen, die für mehr als ein Problem funktionieren.

Diese sind:

- Greedy
- Divide and Conquer
- Dynamische Programmierung
- Lokale Suche
- Backtracking

In dieser Arbeit steht der Entwurf eines Algorithmus im Zentrum: Lernende der 11. Klasse werden schrittweise und entdeckend an den Algorithmus eines Problems herangeführt. Grundlage der Lektionen stellt die Modellierung von Suchbäumen für verschiedene Probleme dar. Dadurch wird erkannt, dass für viele Probleme das Wachstum des Suchbaums zu speicherintensiv wird, um ihn zuerst ganzheitlich zu erstellen.
Damit dieser limitierende Faktor umgangen werden kann, wird das Backtracking-Verfahren erlernt und angewandt. Backtracking-Algorithmen beruhen auf dem Prinzip von Trial-and-Error. Dabei suchen sie anhand des Schemas einer Tiefensuche im Suchbaum nach einer Lösung. Dies geschieht rekursiv und somit muss nicht der ganze Suchbaum im Speicher erstellt werden.

Das Backtracking beruht auf der systematischen Auflistung aller Lösungen in folgender Weise:

- Solange die Möglichkeit eine Lösung zu finden besteht, führe einen möglichen Schritt in Richtung Lösung aus.
- Falls die Lösung nicht erreicht werden kann (Sackgasse), gehe zurück zu einem der Schritte zuvor, die über noch ungeprüfte Möglichkeiten verfügen, und wiederhole das Vorgehen.

Im Anschluss werden wir das Konzept von *Forward-Checking* als Erweiterung des naiven Backtrackings vorstellen. Mit diesem werden anhand von Einschränkungen (Constraints), welche je nach betrachtetem Problem bestehen, bereits früher Sackgassen erkannt und der Backtrack-Schritt somit früher eingeleitet. Dies führt zu einer Laufzeiteinsparung des Backtracking-Algorithmus.

Lernziele

Die Lernenden

- erstellen Suchbäume;
- verstehen das Konzept von Backtracking und wenden es auf Problembeispiele an;
- verstehen das Prinzip von Look-Ahead-Strategien.

Jahrgangsstufe

Ergänzungs- oder Grundlagenfach 11. Klasse

Vorwissen ***(ist unentbehrlich für die Entwicklung des folgenden Algorithmus)***

- *Die Lernenden wissen, wie Suchbäume erstellt werden; sie vertiefen dieses Wissen jedoch durch Üben.*
- *Die Lernenden sind mit ersten Algorithmen (wie Tiefen- und Breitensuche) vertraut.*

Ablauf

1. Die Lernenden werden zu einem Gedankenspiel animiert und vor folgendes Problem gestellt: Sie sollen bei einer Schnitzeljagd einen Weg finden.
2. Sie werden aufgefordert als Gruppe, ausgehend vom Startpunkt, einige mögliche Optionen durchzuspielen und einen ihre Planung («der Jagd») darstellenden Graphen zu zeichnen.
3. In einer anschliessenden Klassendiskussion werden die «Jagdwege» verglichen und erkannt, dass es mehrere Möglichkeiten und Sackgassen gibt.
4. Erste Aufgaben repetieren die Erstellung eines Suchbaumes.
5. Im Anschluss an die Schnitzeljagd wird das Backtracking erklärt. Das n *x* n-Damenproblem stellt eine weitere diesbezügliche Aufgabe dar.
6. Über die Auseinandersetzung mit dem Damenproblem wird das Backtracking mit dem Look-Ahead-Zusatz ergänzt. Es folgen weitere Übungsaufgaben.

Zeitaufwand

Circa 4 Lektionen, wovon 2 Übungslektionen sind und 2 mit Gruppenaufträgen und Theorie gespickt sind.

1 Inhaltsverzeichnis

2 Einstieg in den Algorithmen-Entwurf

2.1 Lehrerinput: Auftragserteilung

Die Lernenden werden in Dreiergruppen eingeteilt und ihnen wird ein Kartenausschnitt von Stans vorgelegt. Die Karte ist jedoch mit einem weissen Blatt bedeckt, das ein Loch enthält. Letzteres befindet sich beim Kollegium St.Fidelis, dem Ausgangspunkt dieser imaginären Schnitzeljagd.
Die Lehrperson fordert die Lernenden auf, das weisse Blatt noch nicht zu verschieben und erteilt jeder Gruppe folgenden schriftlichen Auftrag (siehe die dazugehörige Druckvorlage im Anhang).

2.2 Gruppenauftrag

Schnitzeljagd durch Stans

Auftrag
Eine imaginäre Schnitzeljagd führt dich durch Stans. Finde möglichst geschickt zum Ziel, also zum Versteck der gegnerischen Gruppe.

Hinweise:

- Dies ist eine Gruppenarbeit, tausche dich also mit zwei Mitschüler*innen aus.
- Du hast einen Dorfplan von Stans, auf dem ein Papier mit kreisrundem Ausschnitt liegt (siehe unten). Der kreisrunde Ausschnitt liegt exakt auf dem Startpunkt der Schnitzeljagd und zeigt zwei in entgegengesetzte Richtung weisende Pfeile.
- Du hast 15 Minuten Zeit bis zum Vorlegen deiner Lösung.

Durchführung:

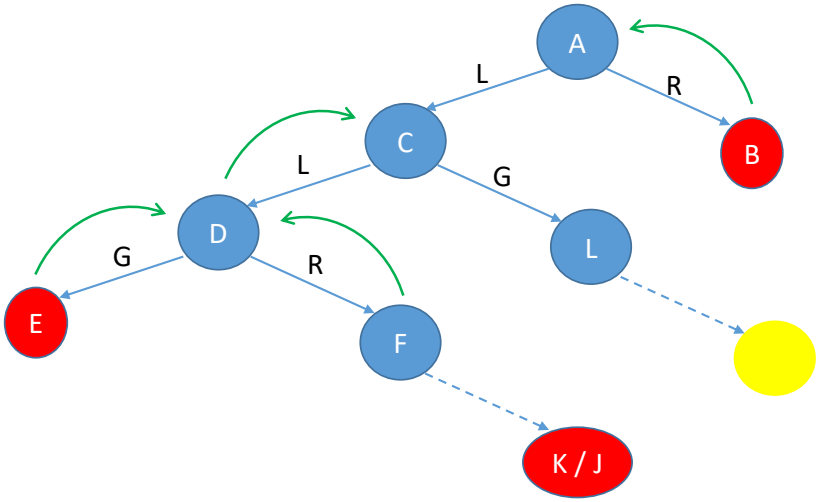
1. Verschiebe das Papier in eine Pfeilrichtung entlang der Strasse auf der Karte – niemand will sich die Schuhe dreckig machen ;-)
2. Skizziere mit den erreichten Zwischenstationen einen Graphen und schreibe auf die Kanten deine jeweilige Entscheidung (L = links, R = rechts, G = geradeaus).
3. Schreibe deine Entscheidungsfolge, die zum Ziel führt, auf.

2.3 Lehrpersonenhinweis

Weiteres Vorgehen:
Besprechung der Graphen und Lösungen der Lernenden in einer Klassendiskussion

Im Anschluss an die Gruppenarbeit von ca. 15 min werden die Lösungsvarianten und -verfahren der einzelnen Gruppen in der Klasse diskutiert. Ziel der Diskussion ist die gemeinsame Ausarbeitung eines Suchbaums, der aus Lösungsgraphen von einigen Lernenden, versehen mit Ergänzungen durch die Lehrperson, entsteht.
In der Folge wird zunächst darauf Wert gelegt, die Darstellung von Suchbäumen zu üben, bevor man später diesen gemeinsam erstellten Suchbaum wieder aufgreift, um Verfahren zu betrachten, mit denen Suchbäume nach Lösungen abgesucht werden können (siehe Backtracking).

Dies würde gemeinsam mit den Lernenden aufgezeichnet. (Bei den von L und bei F ausgehenden Kanten gibt es natürlich weitere Knoten dazwischen.)



3 Suchbaum-Modellierung

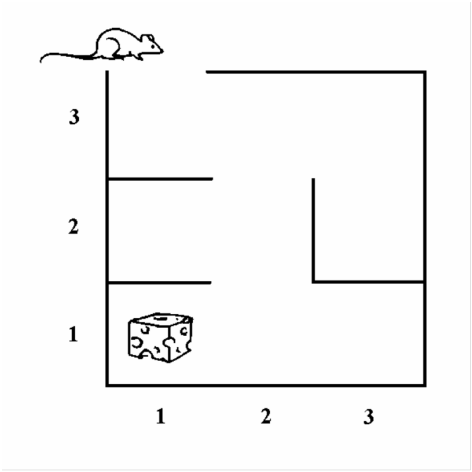
Einen Suchbaum kann man verstehen als eine abstrakte Darstellung von Daten bzw. Lösungen oder des Prozesses der Lösungssuche, bei der die Menge von Elementen bzw. Lösungsschritten in einer Baumstruktur abgebildet wird. Dabei entspricht jeder Schritt der Bewegung entlang einer Kante zwischen zwei Knoten (Elementen).
Bei der Lösungssuche erfordert jeder einzelne Schritt eine Entscheidung, ob und wie man bei der Suche fortfährt. Dadurch wird der Suchbaum, der alle Lösungskandidaten des Ausgangsproblems enthält, modelliert.

«Der wesentliche fachdidaktische Mehrwert der Modellierung besteht darin, dass beim Problemlösen mit dem Computer eine planerische, konzeptionelle Modellierungsphase explizit durchlaufen wird, die zwischen dem zu lösenden Ausgangsproblem und der algorithmischen Lösung liegt.» (*Quelle: Suchbaum-Modellierung, Gerhard Röhner*)

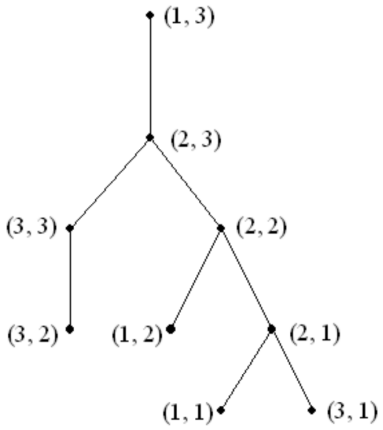
3.1 Lehrerinput: Beispiel

Im Lehrgespräch können die Sätze zur Suchbaum-Modellierung mit den Lernenden erörternd repetiert werden und/oder in einem nachfolgenden schriftlichen Schülerauftrag behandelt werden.

Die Lehrperson entfaltet im Folgenden auf einer Projektionsfläche ein kleines Beispielproblem und zeigt, wie man einen Suchbaum erstellt:



Lösung



Die Knoten (X-/ Y-Koordinaten) des Suchbaums entsprechen der Position der Maus im Labyrinth. Der Schritt, den die Maus von einem auf ein benachbartes Feld macht, wird durch eine Kante zwischen dem bisherigen und dem neuen Knoten repräsentiert. Der maximale Verzweigungsgrad bei einem Labyrinth beträgt drei (links, rechts sowie geradeaus), weil man logischerweise nicht zu dem Feld zurückgeht, von dem man gerade gekommen ist. Einzig die Wurzel eines Labyrinth-Suchbaums kann sich in alle vier Richtungen verzweigen, wenn die bewegende Figur mittendrin startet.

3.2 Schülerauftrag

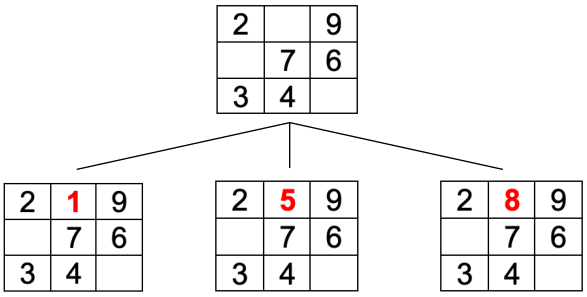
Darstellung von Lösungskandidaten mittels Suchbäumen

Übungsaufgaben
Du betrachtest nun zwei weitere Probleme, die sich mit Suchbäumen modellieren lassen.

1. **Sudoku**
   Du hast eines der 9 Quadrate eines Sudokus vor dir. Die möglichen Kandidaten für die leeren Felder sind die Zahlen 1, 5 und 8.

2		9
	7	6
3	4	

Beginne wie folgt:



2. Das n *x* n-Damenproblem

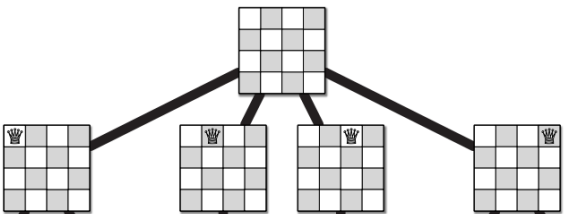
Auf dem leeren Schachbrettfeld gibt es 64 Möglichkeiten eine Dame zu setzen. Möchtest du eine zweite Dame setzen, bleiben dafür noch 63 Möglichkeiten. Das heisst, mit der Annahme 8 Damen zu setzen, bei denen sich je zwei gegenseitig nicht schlagen, erhält man auf der Suche nach Lösungen mit diesem Brute-Force-Ansatz genau 64! viele Lösungskandidaten, unter welchen man die tatsächlichen Lösungen bestimmen muss. Effizienter und pfiffiger, aber immer noch Brute-Force, erzeugt man weniger zu berücksichtigende Lösungskandidaten, in dem man alle Möglichkeiten betrachtet, um *in jeder Zeile genau eine* Dame zu platzieren. So sind es nur noch, aber doch noch zu viele, nämlich $8^8 = 2^{24}$, also rund 16.7 Millionen mögliche Platzierungen als Lösungskandidaten. Man kann es auf 8! reduzieren, wenn man zusätzlich nur Lösungskandidaten zulässt, die *auch in jeder Spalte* genau eine Dame haben. Das ergibt nun 40‘320. Für Probleme dieser Grösse können wir keine gesamten Suchbäume mehr erstellen und werden später mit dem Backtracking eine möglich Lösung kennenlernen, um dies zu umgehen. Bevor wir uns dem Backtracking widmen, möchten wir aber ein Gefühl für einen solchen Suchbaum mit kleineren Schachbrettern bekommen.

a) Deine Aufgabe ist es nun, das Damenproblem auf einem 4 *x* 4-Schachbrett als Suchbaum darzustellen.

   Hinweis I: Die Damen können sich im Schachspiel geradeaus, sowie auf den diagonalen Feldern beliebig weit bewegen, sprich andere Figuren „schlagen“.

   Hinweis II: Es gibt im Baum Äste, die zu keiner Lösung führen, in diesem Fall kannst du abbrechen und vom letzten Knoten ausgehend den nächsten Ast ansetzen.

b) *Zusatzaufgabe*: Versuche dasselbe mit einem Schachbrett der Grösse 6 *x* 6.



4 Backtracking

4.1 Lehrerinput: Theorie Backtracking

Überleitung vom n x n-Damenproblem zum Backtracking

Wir haben die Grundlagen gelegt: Der Suchbaum ist wieder präsent. Nun stellt sich die Frage; Wie können wir eine oder mehrere Lösungen im Suchbaum finden. Hier ist der Speicheraufwand von grossen Suchbäumen (oftmals exponentielles Wachstum) die Knacknuss. Sprich, für viele Probleme kann nicht zuerst der gesamte Suchbaum erstellt werden (das gilt zum Beispiel für das 8 *x* 8-Damenproblem).

An diesem Punkt kommt nun das Backtracking ins Spiel, welches eine sehr speichereffiziente Möglichkeit bietet, und wir greifen zurück auf unser Gedankenspiel «Schnitzeljagd in Stans».

Zusätzlich haben die Lernenden vermutlich gemerkt (was schon in Hinweis II kommuniziert wurde), führen gewisse Äste nicht zu einer Lösung des Damenproblems. Sie können sich auch vorstellen, dass das Erstellen des gesamten Suchbaumes für ein grösseres Schachfeld relativ lange dauern kann.

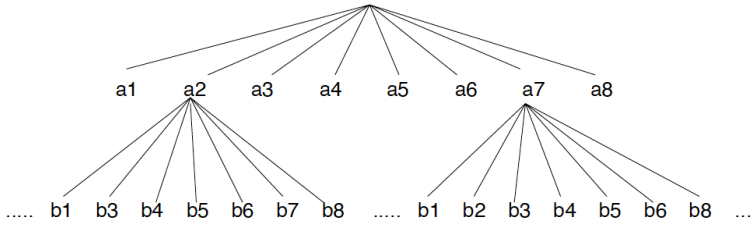
Dies wird im zweiten Schritt mit dem optimierten Look-Ahead-Backtracking behandelt.

Folgende Theorieblätter werden den Lernenden ausgehändigt, die jedoch zuerst mündlich an einem Beispiel erarbeitet werden.

Die Schnitzeljagd – Beispiel eines Backtracking Verfahrens

Ein Algorithmus zur systematischen Suche nach Lösungen im Suchbaum ist das Backtracking. Backtracking basiert auf dem Prinzip der Tiefensuche, wobei ein Suchbaum von links nach rechts abgesucht wird. Führt ein Suchpfad in eine Sackgasse, dann wird der jeweils letzte Schritt rückgängig gemacht (backtrack) und vom letzten Verzweigungspunkt (Knoten des Baums) aus die Suche zu einem noch unbesuchten Nachbarknoten fortgesetzt.
Das Zurückgehen und erneute Voranschreiten in eine andere «Richtung» wird so lange wiederholt, bis eine Lösung des vorliegenden Problems gefunden ist oder man erkennt, dass das Problem keine Lösung besitzt.

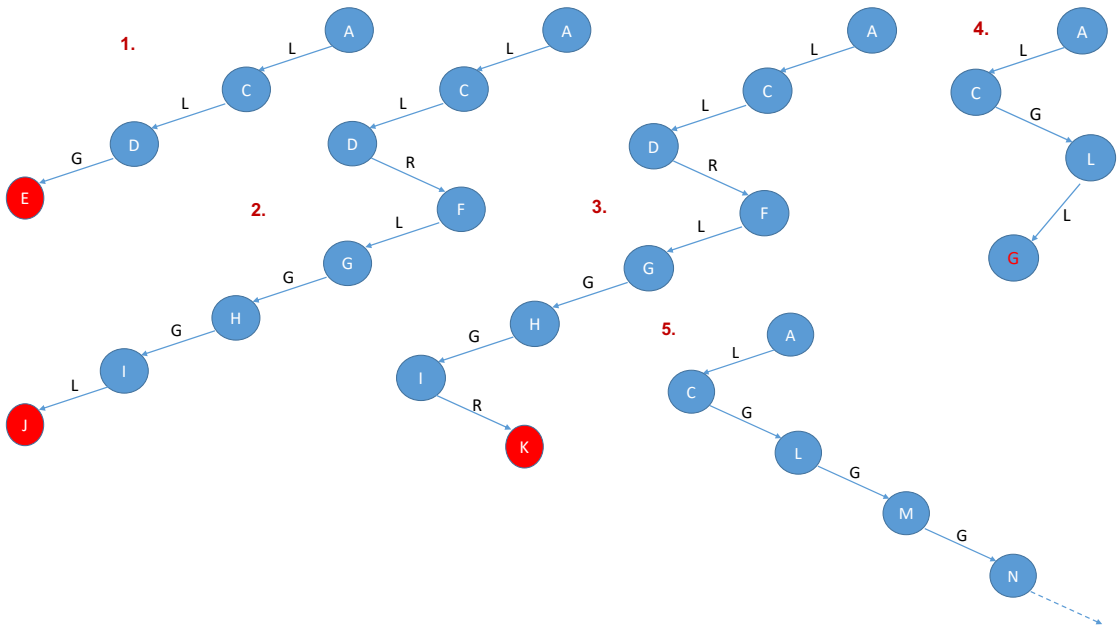
Der Algorithmus löst Probleme also mittels systematischer Durchmusterung über die Trial-and-Error-Methode, also über die Methode Versuch und Irrtum. Er ist besonders nützlich, da das Backtracking-Verfahren für das Durchsuchen des Suchbaums diesen nicht zuerst komplett erstellen muss. Somit können wir auch Suchbäume von Problemen durchsuchen, welche einen viel zu hohen Speicherplatz bzw. Zeichnungsplatz beanspruchen würden (*siehe Abb. 1: 8 x 8-Damenproblem*). Der Speicherplatz, um den ganzen Suchbaum abzuspeichern, wüchse exponentiell mit der Länge der Lösungsdarstellung (wie wir es in den vorherigen Beispielen gesehen haben). Das Ziel des Backtrackings ist es, während der Suche nach einer / allen Lösung/en höchstens einen Weg aus der Wurzel zu einem Baumknoten abzuspeichern. Der Weg kann also höchstens die Länge der Darstellung von Lösungen haben.



Schematischer Aufbau des Suchbaums für das 8 *x* 8-Damenproblem. Es müsste ein Suchbaum mit 8^8 (16'777'216) möglichen Stellungen erstellt werden.

Backtracking graphisch am Beispiel eines Schnitzeljagd-Ausschnitts

Dies würde gemeinsam mit den Lernenden konzipiert.
Angenommen man würde immer zuerst links abbiegen, entstehen nacheinander folgende nummerierten Bilder. Mit den Lernenden würde man die Variante 1, 2, 3, 4 einfach nach dem Erstellen wieder durchstreichen, um zu zeigen, dass der Speicherplatz (im Cache) frei wird.



Der Algorithmus ist jedoch nicht immer die beste Wahl, da er bei grossen Suchräumen und komplexen Problemen sehr ineffizient sein kann, sprich laufzeitintensiv ist. Die Zeit des Durchlaufens des ganzen Baumes entspricht der Grösse des Suchbaumes. Der Vorteil des Algorithmus ist jedoch, dass er durch den rekursiven Aufruf nicht speicherintensiv ist.

Ein wichtiger Bestandteil des Backtracking-Algorithmus ist die Heuristik[1], die bestimmt, welcher Pfad als Nächstes ausprobiert wird. Eine gut gewählte Heuristik (wie im Beispiel der Schnitzeljagd «Immer zuerst nach rechts abbiegen») kann die Effizienz des Algorithmus verbessern, indem sie die Anzahl der zu durchsuchenden möglichen Lösungswege reduziert.

4.2 Look-Ahead-Backtracking

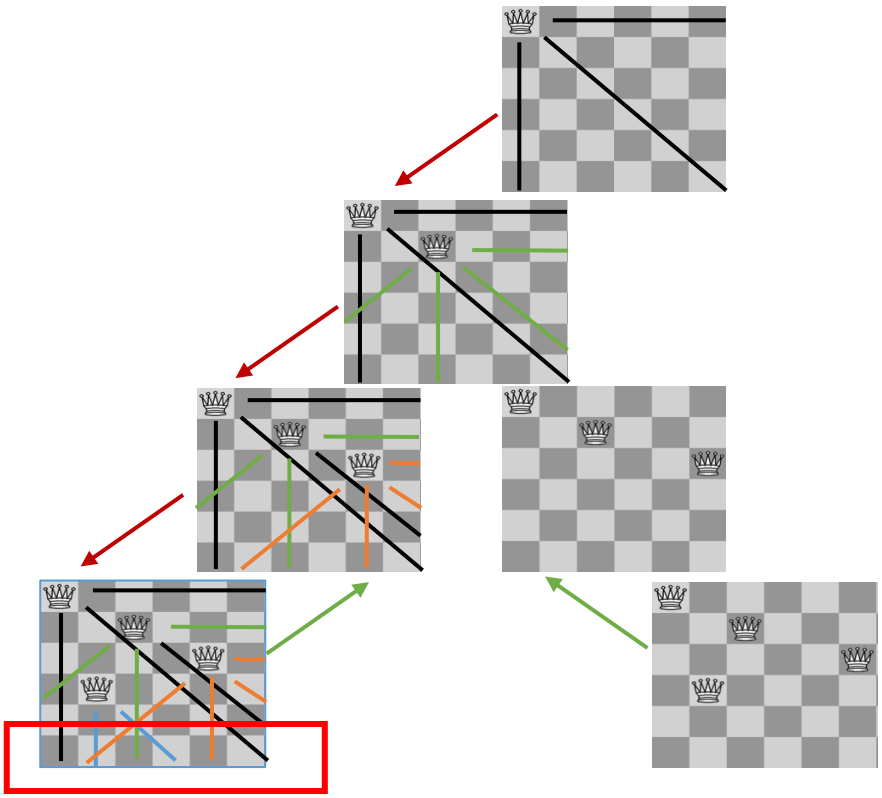
Das n *x* n -Damenproblem – Look-Ahead-Backtracking

Beim 6x6-Schachbrettspielfeld und grösseren ist das Backtracking sehr zeitaufwendig und mühsam. Du kannst dir ausmalen, wie ineffizient es für ein 8x8 Felder Brett ist. Trotzdem wirst du dich gleich dieser Aufgabe widmen.

Wenden wir uns zunächst einem 6 *x* 6-Schachspielfeld zu.

Tipp: Streiche alle Felder durch, deren Spielfiguren die Königinnen schlagen könnten. Wirst du in jeder Zeile eine Königin setzen können?

Mit den Lernenden wird dieses Bild erarbeitet und der «Look-Ahead» (rot eingerahmt) erläutert.



[1] Definition Heuristik: Heuristiken sind Methoden, welche die Möglichkeiten für Lösungen einschränken.

Obwohl man in der fünften Zeile eine Dame setzen könnte, ist dies in der sechsten Zeile bereits nicht mehr möglich. Mit einem Look-Ahead-Blick sieht man die sechste Zeile, bevor man in der fünften eine Dame setzt, und somit kann bereits hier abgebrochen werden. Der Backtrack-Schritt wird eingeleitet.

Look-Ahead-Backtracking bedeutet, dass zusätzlich zur Methode des Backtracking vorausgeschaut wird. Es wird/werdeb eine oder mehrere Schrittweite/n mehr in Betracht gezogen, bevor eine Entscheidung getroffen wird. Somit vermeidet man die Generierung einiger aussichtsloser Kandidaten.

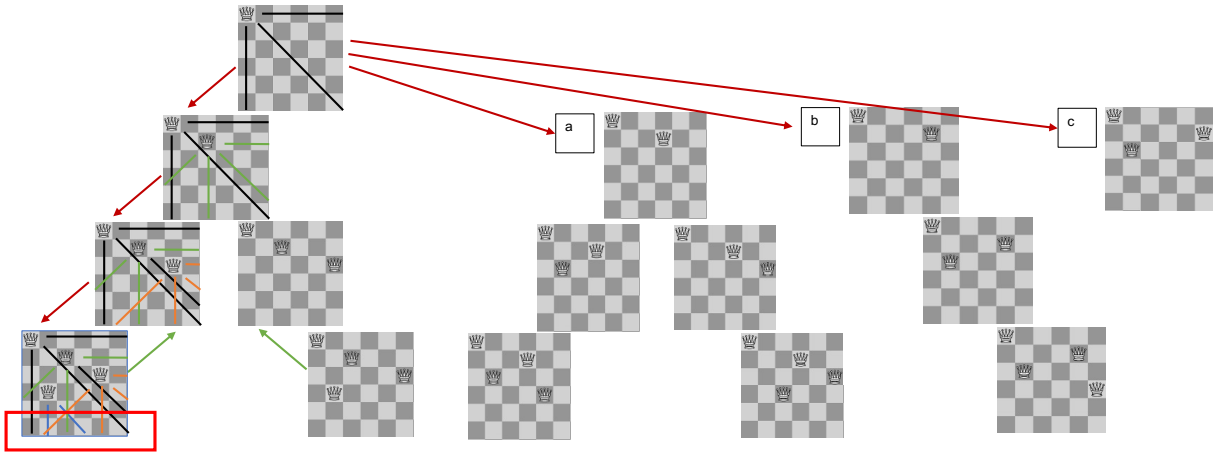
Im Grunde genommen sind diese vorausschauenden Entscheidungen nichts anderes als sogenannte Constraints (Einschränkungen). Ein Constraint ist eine Bedingung, die erfüllt sein muss, damit ein Lösungskandidat eine zulässige Lösung ist. Beim Backtracking-Algorithmus werden diese Einschränkungen verwendet, um das Durchschauen von Lösungskandidaten zu reduzieren und damit die Effizienz des Algorithmus zu erhöhen. Jeder Schritt im Backtracking-Algorithmus prüft, ob die aktuelle Teillösung gegen die bestehenden Constraints verstösst. Wenn dies so ist, wird der Algorithmus zurückgehen und eine andere Möglichkeit ausprobieren. Constraints können in vielen Problemen verwendet werden, wie z.B. der Erstellung von Zeitplänen oder dem Design von Schaltkreisen.

Auf das n *x* n-Damenproblem angewendet, lautet der Constraint, dass in jeder Zeile, Spalte und Diagonale höchstens eine Dame stehen kann.

4.3 Übungsaufgaben

1. **6 *x* 6-Damenproblem**
   Finde mittels Look-Ahead-Backtracking ***alle*** Lösungskandidaten dieses Problems.

   Lösung



2. Das 8 *x* 8-Damenproblem

Führe den Backtracking-Algorithmus so lange aus (mit Hilfe von Look-Ahead), bis du ***einen*** Lösungskandidaten des 8 *x* 8-Damenproblems gefunden hast.

3. Sum of Subsets

Das Subset Sum - Problem lautet wie folgt:

Bestimme aus einer gegebenen Menge von Zahlen eine Teilmenge, die in ihrer Summe die gegebene Zielnummer ergibt.

Betrachten wir ein Beispiel:

Zielnummer = 33 *Gegebene Menge von Zahlen = {2, 4, 5, 12, 17, 22}*
Lösung: L = {4, 12, 17}, da 4 + 12 + 17 = 33

Dies kann wiederum mit einem Backtracking-Verfahren gelöst werden. Zusätzlich gibt es mehrere Einschränkungen, die für eine Look-Ahead-Optimierung in Frage kämen.

a) Löse folgendes Subset Sum - Problem mithilfe eines Suchbaums, der mit der Backtracking Methode aufgestellt wird.

M = {2, 5, 17, 25, 42, 56, 68, 86, 104}; Zielnummer = 116

b) Überlege dir mögliche Look-Ahead-Strategien, welche es dir ermöglichen, die Grösse des abzusuchenden Suchbaums zu verringern. Zeichne für deine Strategien jeweils einen Ast, bei dem frühzeitig erkannt werden kann, dass er zu keiner Lösung führen wird.

Lösung: L = {5, 25, 86}

Dies sind mögliche Look-Ahead-Strategien:

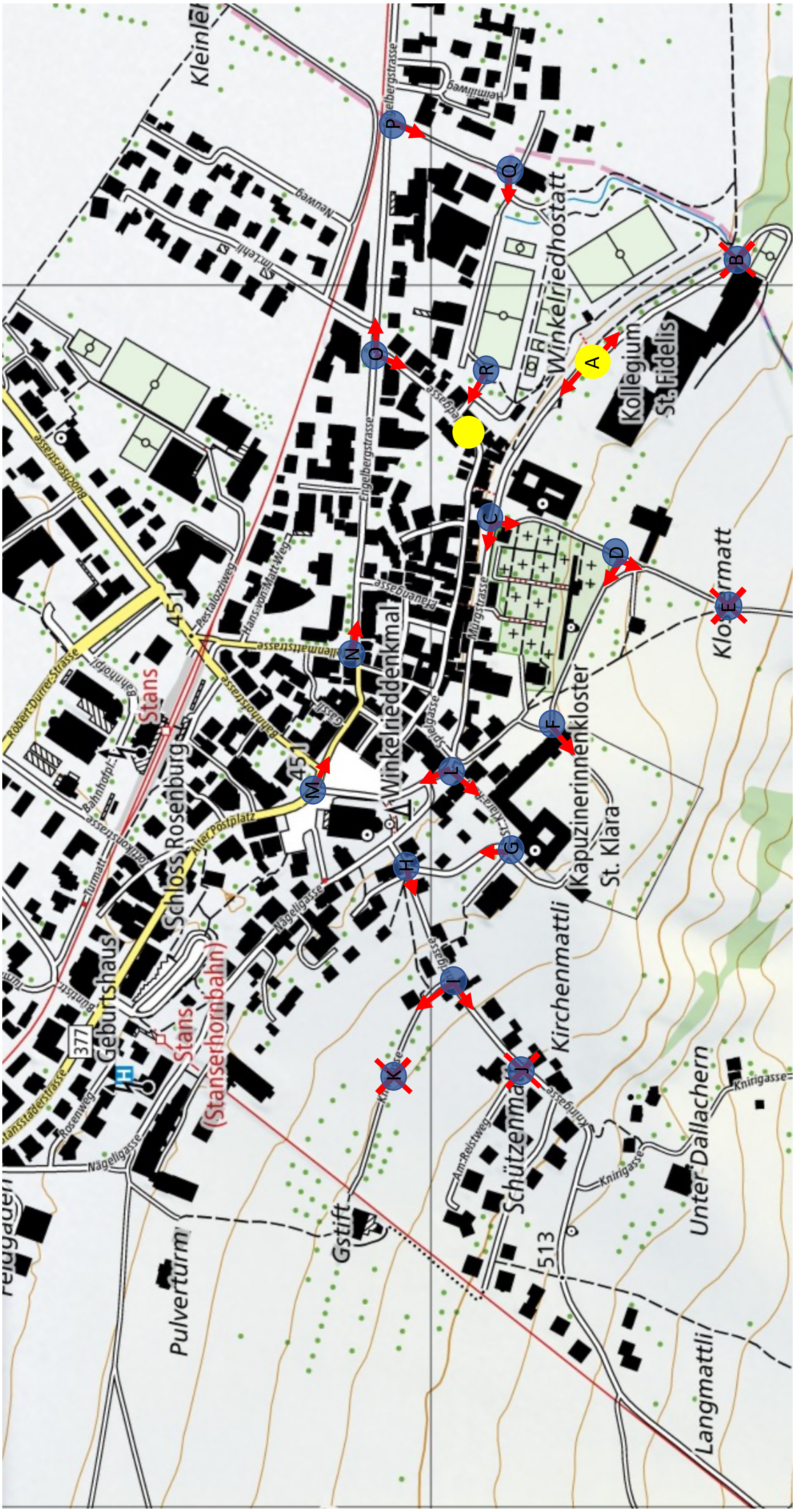
- Die Zielnummer ist gerade; falls nur noch gerade Zahlen vorhanden sind und die momentane Summe ungerade ist, kann abgebrochen werden.

- Man berechne zuerst die Gesamtsumme (405) und ziehe davon jeweils die bereits verwendeten Zahlen ab. Sobald die Restsumme zu klein ist, um die Zielnummer zu erreichen, kann abgebrochen werden.

4. **Zusatz:**
   Wähle eine andere Schachspielfigur, von der du weisst, wie sie laufen darf. Führe wiederum das Look-Ahead-Backtracking aus, um eine Lösung zu erhalten.

5 Anhang

Jede Druckvorlage ist aus Platzgründen auf einer eigenen Seite bereitgestellt.


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
Winkelriedhostatt
Kollegium St. Fidelis
Klostermatt
Kapuzinerinnenkloster St. Klara
Winkelrieddenkmal
Kirchenmattli
Schützenmattli
Unter Dallachern
Langmattli
Gstift
Pulverturm
Schloss Rosenburg
Stans
Stans (Stanserhornbahn)
Geburtshaus
Kleinfeld
513
451
377