

Algorithmen und Datenstrukturen

SS 2019

Übungsblatt 12: Paradigmen

Aufgabe 12-1 Backtracking

In der Vorlesung haben Sie als Beispiel für ein Backtracking Problem das 4 Damenproblem kennengelernt.

- (a) Geben Sie nun eine Lösung für das 5 Damenproblem an. Berechnen Sie diese mittels Backtracking. Sie müssen nicht alle Schritte explizit darstellen. Sie können die Damen mit x kennzeichnen und in die folgende Tabelle eintragen:

	1	2	3	4	5
A					
B					
C					
D					
E					

Lösungsvorschlag:

Eine mögliche Lösung ist zum Beispiel:

	1	2	3	4	5
A		X			
B				X	
C	X				
D			X		
E					X

- (b) Wie groß ist die Komplexität für das n Damenproblem, wenn ein naiver Algorithmus für die Berechnung der Lösung verwendet wird?

Es reicht eine obere Abschätzung in O -Notation anzugeben. Dabei kann man vernachlässigen, dass Damen auch diagonal gehen können.

Lösungsvorschlag:

Die Komplexität ist eine Teilmenge von $n!$ und damit $O(n!)$. Sie berechnet sich durch die Anzahl der möglichen Positionierungen der Damen. Für die 1. Zeile sind es n mögliche Positionen, für die 2. $n - 1$ mögliche Positionen (etwas weniger eigentlich, allerdings vernachlässigen wir hier die Diagonalen, da es reicht, die Obermenge anzugeben), \dots , für die n -te Zeile gibt es nur noch (maximal) eine Position. Damit $n \cdot (n - 1) \cdot \dots \cdot 1 = n!$. Mittels verschiedener mathematischer Eigenschaften lassen sich jedoch effizientere Algorithmen finden.

- (c) Weiterhin Sei eine modifizierte Version eines Sudokus mit jeweils nur 4 anstelle von 9 Feldern gegeben. Wie bei dem originalen Sudoku gilt, dass in jeder Zeile, in jeder Spalte und in jedem Block (2x2 Unterquadrat) die Zahlen 1 bis 4 nur einmal vorkommen dürfen.

Gegeben Sei nun folgendes Sudoku:

1	3		
2	4	3	
3	2	1	4
4			

- (i) Geben Sie generell grob einen naiven Algorithmus in Pseudocode an, der mittels Backtracking eine Lösung berechnen kann. Tipp: Es muss nicht über alle Spalten und Zeilen iteriert werden, sondern es reicht eine Betrachtung der freien Felder.

Lösungsvorschlag:

Ein möglicher Brute-Force Algorithmus (alle Möglichkeiten durchprobieren) ist folgender:

```

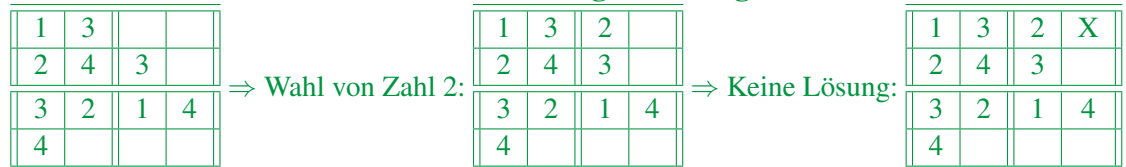
Algorithmus sudoku(int[][] sudokuBrett, Feld[] freieFelder, int freiesFeldZahl)
    if istZahlAufBrettDoppelt(sudokuBrett)
        return false
    if freiesFeldZahl == freieFelder.length
        return true
    for zahl = 1 bis 4
        freiesFeld = freieFelder[freiesFeldZahl]
        sudokuBrett[freiesFeld.zeile][freiesFeld.spalte] = zahl
        if sudoku(sudokuBrett, freiesFeldZahl + 1)
            return true
    return false

```

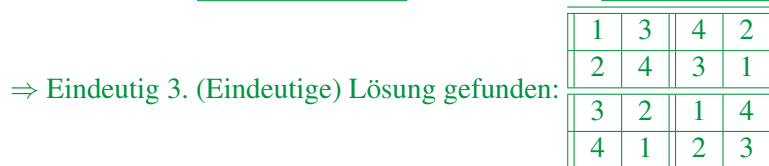
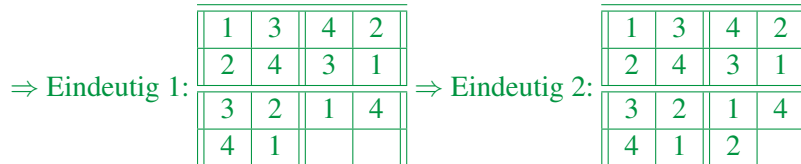
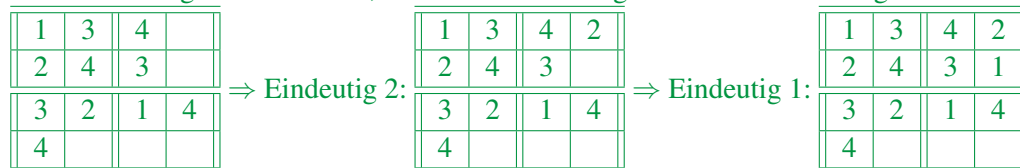
Die Ausführung startet dann mit `sudoku(sudokuBrett, freieFelder, 0)`

- (ii) Berechnen Sie die (eindeutige) Lösung des obigen Sudokus mit Hilfe des erstellten Algorithmus. Geben Sie dabei alle Zwischenschritte an, wobei es reicht, die anfangs noch leeren Felder anzugeben. Im folgenden sollen die freien Felder von links nach rechts und dann von oben nach unten betrachtet werden, also wie in gewohnter deutscher Schreib-/Leserichtung (wobei jede beliebige Abfolge dieser Felder möglich wäre).

Lösungsvorschlag:

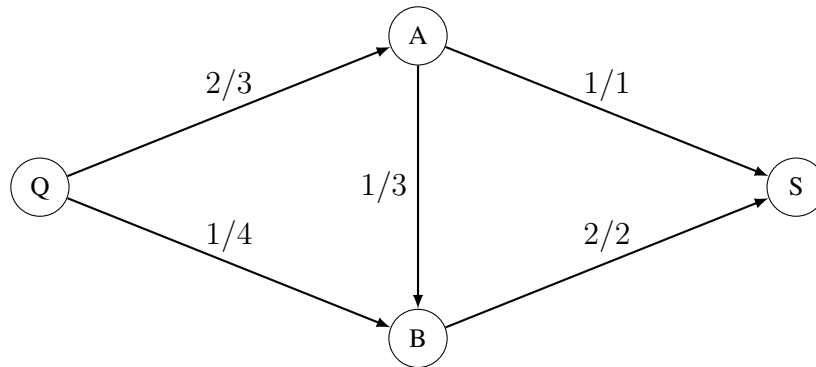


⇒ Keine Möglichkeit für X, daher Backtracking zu letzter Wahlmöglichkeit und einsetzen von 4:

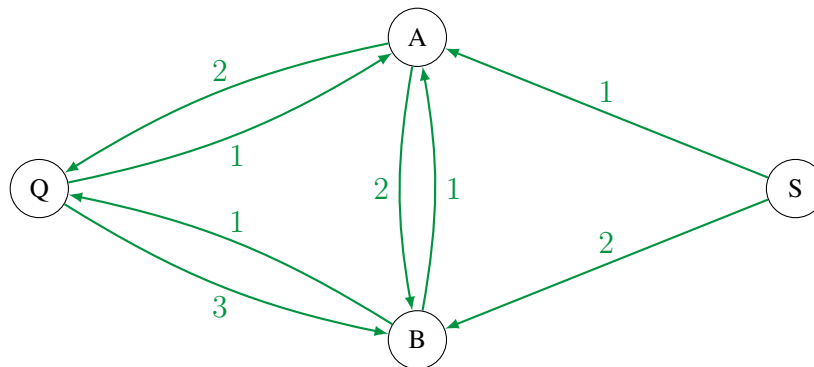


Aufgabe 12-2 Flussnetzwerke

Gegeben sei das folgende Flussnetzwerk:



- (a) Zeichnen Sie das Residualnetzwerk zu dem oben gegebenen Flussnetzwerk in die Vorlage ein. Residualnetzwerk:



- (b) Bestimmen Sie den maximalen Fluss des obigen Flussnetzwerkes mit Hilfe des Min-Cut-Max-Flow-Theorems und kennzeichnen Sie den Schnitt des maximalen Flusses deutlich in dem Flussnetzwerk.

