Das 8 – Damenproblem

Das Problem, Damen so u positionieren, dass sie einander schlagen, ist ja im Prinzip ein Problem jedes Casanova. Und das Problem wird größer, wenn mehr Damen im Spiel sind. Die wirkliche Leistung war es, dieses Problem zu erkennen. Lösungen zu finden ist mit Fleiß leicht möglich, wenn man sogenanntes Backtracking verwendet. In zahlreichen Programmiersprachen wurden Programme verfasst, die diese 92 Lösungen ganz schnell finden.

Als ich vor etwa 40 Jahren auf einem Sharp-Minicomputer ein Programm in Basic geschrieben hatte, piepste das Gerät ungefähr alle 15 Minuten, um eine Lösung anzuzeigen. Diese Lösung konnte man weder speichern noch ausdrucken, sondern musste sie abschreiben. Es dauerte also fast 24 Stunden, um alle Lösungen zu bekommen!

Nach der Lösung eines Problems folgt der nächste Schritt: Die Verallgemeinerung! Hier sind mehrere Möglichkeiten gefunden worden. Größere Spielfelder, Spielfelder in anderer Form, insbesondere die Oberfläche eines Zylinders oder Ringes, Figuren mit anderen Schlageigenschaften. Wählt man etwa die Eigenschaft „Turm“, dann erhält als Lösungen alle Permutationen.

Der erste Schritt zu den Lösungen könnte sein, einen Code zu finden, wie man die Lösungen notieren kann. Eine einfache Möglichkeit besteht darin, die Spalten des Schachbretts von links weg aufsteigend zu nummerieren. Dann betrachtet man die Dame in der ersten Spalte und trägt ihre Reihenzahl auf Platz 1 einer achtziffrigen Zahl ein. Die Platzierung der Dame in Spalte zwei folgt an zweiter Stelle usw.

Die kleinste Lösungszahl lautet 15863724

		8					
					7		
			6				
	5						
							4
				3			
						2	
1							

Man kann aus einer gegeben Lösung 7 weitere Lösungen generieren, indem man sie an den 4 Symmetrieachsen spiegelt, bzw. um 90° 180° oder 270° dreht. Die Lösung 35281746 ist allerdings punktsymmetrisch und liefert daher nur 3 weitere Lösungen.

Vorausgesetzt, man hat bereits alle 92 Lösungen in aufsteigender Reihenfolge vorliegen, dann nimmt man die kleinste Lösung, die keine Verwandte von 1 ist, dies ist die Lösung 2!, und wendet wieder Spiegelungen und Drehungen an. Setzt man dieses Verfahren konsequent fort, dann erhält man die zwölf kleinsten Basiszahlen, aus denen man alle Lösungen ableiten kann. Alles dies kann man aus der später folgenden Exel-Liste ersehen werden.

k	n
1	1
2	5
3	8
4	6
5	3
6	7
7	2
8	4

Wenn man eine Lösung wie links anschreibt, dann gibt es einfache Formeln, um Spiegelungen und Drehungen abzuleiten.

Ich habe der Ausgangszahl den Buchstaben a gegeben. Die Spiegelung an der Querachse nenne ich b, die Formel lautet n ---> 9-n , also b = 84136275, Lösung 92!

c: Spiegelung an der Vertikalachse, Platztauschen nach der Formel k / n -- > 9-k / n

42736851 (man kann die Spalte n auch von unten nach oben lesen!, Lösung 33

d: Drehung nach rechts um 90°. Formel k / n ---> n / 9-k

e: Drehung um 180°. Formel k / n ---> 9-k / 9-n

f: Drehung nach rechts um 270°. Formel k / n --->9-n / k

g: Spiegelung an der Diagonale von links unten nach rechts oben: Formel k / n --> n / k

h: Spiegelung an der zweiten Diagonale. Formel: k / n ---> 9-n/ 9-k.

Zur raschen Bearbeitung der Formeln lohnt es sich, zwei weitere Tabellen anzulegen.

n	9-n

Die Tabelle links fertigt man auf einem Papierstreifen an, den man längs faltet. Nach dem Eintragen der Lösung n, bzw. -n, legt man diesen Streifen entsprechend der jeweiligen Formel mit der linken oder rechten Seite an die passende Spalte von Tabelle 2 und kann dann sofort die Lösung erkennen.

K	9-k
1	8
2	7
3	6
4	5
5	4
6	3
7	2
8	1

Will man rein mechanisch arbeiten, dann kann man für die Spiegelungen einen Spiegel verwenden. Für die Drehungen dreht man das Schachbrett jeweils um den entsprechenden Winkel und schreibt dann die Lösungszahl ab.

Es gibt aber noch einen weiteren Weg, um aus einer gegebenen Lösung zu weiteren 5 Lösungen zu gelangen. Allerdings funktioniert das nur bei 48 Lösungen. Man schiebt die Figuren entweder nach links, nach rechts, nach oben oder nach unten jeweils um ein Feld weiter. Figuren, die dabei aus der Spielfläche herausgeschoben werden, kommen auf der anderen Seite wieder auf das Spielfeld zurück.

Die Formel für die Rechtsschiebung: k / n ---> k+1 / n , k = 9 wird durch k = 1 ersetzt.

Formel für die Linksschiebung: k / n ---> k-1 / n, K = 0 wird durch K = 8 ersetzt.

Schiebung nach oben: k / n ---> k / n+1 n = 9 wird durch n = 1 ersetzt.

Schiebung nach unten: k / n ---> k / n-1 n = 0 wird durch n = 8 ersetzt.

Wenn zwei aufeinander folgende Schiebungen, z. Bsp. zuerst, nach rechts und dann nach oben, erfolgreich zu einer Lösung geführt haben, dann kann man dasselbe Resultat auch durch eine diagonale Schiebung erreichen.

↗ Formel: k / n ---> k+1 /n+1, 9 ist durch 1 zu ersetzen

↖↘ Formel: k / n ---> k-1 / n+1, k = 0 wird durch k = 8, n = 9 durch n = 1 ersetzt.

↘ Formel: k / n ---> k+1 / n-1, k = 9 wird durch k = 1, n = 0 wird durch n = 8 ersetzt.

↙ Formel: k / n ---> k-1 / n-1, k = 0 bzw. n = 0, werden durch k = 8 bzw. n = 8 ersetzt.

Es entstehen dabei Ringstrukturen, die wie eine liegende oder stehende Acht in rechteckiger Ausführung aussehen. Rechts in untenstehender Tabelle sind die Mitglieder der 8Ringe R1, R4, R5, R6, R10, R17, R19 und R22 aufgelistet. Lösungen mit N gehören keinem Ring an. Jeder Ring enthält jeweils genau ein Elementaus den „Familien“ B1, B3, B4, B6, B8 und B11. Die 8 Ringe erzeugen also genau jene Elemente aus Schiebungen, die in den „Familien“ durch Spiegelungen und Drehungen erzeugt werden.

Nr.	Lösung	O	1	2	3	4	5	6	7	8	9	10	11	12	N	1	4	5	6	##	##	##	##
1	15863724		B1a													R1							
2	16837425			B2a											N								
3	17468253			2g											N								
4	17582463		B1g														R4						
5	24683175				B3a													R5					
6	25713864					B4a													R6				
7	25741863						B5a								N								
8	26174835							B6a								R1							
9	26831475								B7a						N								
10	27368514									B8a									R10				
11	27581463										B9a				N								
12	28613574									8d							R4						
13	31758246							6g									R4						
14	35281746			Drehsymmetrisch								B10a			N								
15	35286471			2f											N								
16	35714286							6h										R5					
17	35841726												B11a								R17		
18	36258174													B12	N								
19	36271485					4f																R19	
20	36275184						5h								N								
21	36418572	S1									9c				N								
22	36428571		B1d																				R22
23	36814752	S2					5c								N								
24	36815724													12d	N								
25	36824175												11f						R6				
26	37285146		B1ahh										11e			R1							
27	37286415								7g						N								
28	38471625				3d														R10				
29	41582736					4g													R10				
30	41586372	S3	B1ar							8c						R1							
31	42586137								7e						N								
32	42736815										9f				N								
33	42736851	S4	B1c																R10				
34	42751863													12h	N								
35	42857136												11d										R22
36	42861357				3e												R4						
37	46152837							6e													R17		
38	46827135						5f					10f			N								
39	46831752	S5				4c												R5					
40	47185263	S6												12d	N								
41	47382516	S7			3f																	R19	
42	47526138			2e											N								
43	47531682									8h													R22
44	48136275										9d				N								
45	48157263						5d								N								
46	48531726								7d						N								

47	51468273								8g				N											
48	51842736	q				5g							N											
49	51863724												N											
50	52468317								8f							R5								
51	52473861	S8	2c										N											
52	52617483	B1ahr		3h										R1										
53	52814736												12e	N										
54	53168247				4h																			R22
55	53172864										10g			N										
56	53847162	S9					6c											R10						
57	57138642			3c													R6							
58	57142863											11g				R5								
59	57248136												12f	N										
60	57263148	B1e																	R17					
61	57263184									9h				N										
62	57413862	S10					7c							N										
63	58413627								8e											R19				
64	58417263				4d														R17					
65	61528374			3g															R17					
66	62713584						7h							N										
67	62714853	S11										11c								R19				
68	63175824											11h			R4									
69	63184275	B1h											12g	N										
70	63185247					5e								N										
71	63571428	B1f														R5								
72	63581427									9f				N										
73	63724815					5f								N										
74	63728514	B1ahhr			4e										R1									
75	63741825												12b	N										
76	64158273											11b						R10						
77	64285713						6f																	R22
78	64713528		2h											N										
79	64718253	S12									10b			N										
80	68241753						6d										R6							
81	71386425								8h								R6							
82	72418536									9b				N										
83	72631485								8b										R17					
84	73168524						7b							R4										
85	73825164						6b													R19				
86	74258136					5b								N										
87	74286135				4b										R4									
88	75316824			3b																				R22
89	82417536	1d															R6							
90	82531746		2d											N										
91	83162574		2b											N										
92	84136275	B1b																		R19				
														44	6	6	6	6	6	6	6	6		

Ein Pythonprogramm, um die Lösungen zu generieren.

```
# Erzeuge eine Liste von Lösung auf einem Brett mit Reihen und Spalten.
# Eine Lösung wird durch eine Liste der Spaltenpositionen,
# indiziert durch die Reihennummer, angegeben.
# Die Indizes beginnen mit Null.
```

```
def damenproblem(reihen, spalten):
    if reihen <= 0:
        return [[]] # Problem nicht definiert, eine Dame auf dem nicht-
existierenden Brett
    else:
        return eine_dame_dazu(reihen - 1, spalten, damenproblem(reihen - 1,
spalten))

# Probiere alle Spalten, in denen für eine gegebene Teillösung
# eine Dame in "neue_reihe" gestellt werden kann.
# Wenn kein Konflikt mit der Teillösung auftritt,
# ist eine neue Lösung des um eine Reihe erweiterten
# Bretts gefunden.
def eine_dame_dazu(neue_reihe, spalten, vorherige_loesungen):
    neue_loesungen = []
    for loesung in vorherige_loesungen:
        # Versuche, eine Dame in jeder Spalte von neue_reihe einzufügen.
        for neue_spalte in range(spalten):
            # print('Versuch: %s in Reihe %s' % (neue_spalte, neue_reihe))
            if kein_konflikt(neue_reihe, neue_spalte, loesung):
                # Kein Konflikte, also ist dieser Versuch eine Lösung.
                neue_loesungen.append(loesung + [neue_spalte])
    return neue_loesungen

# Kann eine Dame an die Position "neue_spalte"/"neue_reihe" gestellt
werden,
# ohne dass sie eine der schon stehenden Damen schlagen kann?
def kein_konflikt(neue_reihe, neue_spalte, loesung):
    # Stelle sicher, dass die neue Dame mit keiner der existierenden
    # Damen auf einer Spalte oder Diagonalen steht.
    for reihe in range(neue_reihe):
        if loesung[reihe]         == neue_spalte              or  # Gleiche
Spalte
           loesung[reihe] + reihe == neue_spalte + neue_reihe or  # Gleiche
Diagonale
           loesung[reihe] - reihe == neue_spalte - neue_reihe:    # Gleiche
Diagonale
                return False
    return True

for loesung in damenproblem(8, 8):
    print(loesung)
```

Dieser rekursiv programmierte Algorithmus kann leicht in einen nicht-rekursiven (iterativen) umgewandelt werden.