# Das Haus des Nikolaus

> **Exkurs: Quelle**
> Diese Aufgabe stammt von der Website oberstufeninformatik.de. Die Materialien sind nicht Freeware, sondern Beerware. Voraussetzung für den Einsatz ist, dass der Benutzer ein Bier zum Wohle des Autors Horst Gierhardt trinkt .

Hier ist das *„Haus des Nikolaus“* mit einer bestimmten Nummerierung der Eckpunkte vorgegeben. Es sollen alle Lösungen zum Zeichnen der Figur in einem Zug gefunden werden. Eine Lösung könnte dann in der Form 123451352 ausgegeben werden. Das Programm soll eine einfache Anpassung an andere Graphen ermöglichen. Der Ausschluss von gespiegelten Lösungen ist nicht gefordert.

Eine Lösung lässt sich nach dem Prinzip *Versuch und Testen* ermitteln. Eine vermutete Teillösung muss wieder verworfen werden, wenn ein Test ihre Ungültigkeit nachgewiesen hat. Man nennt diesen Ansatz deshalb auch *Rückverfolgen* oder *Backtracking*. Mit diesem Ansatz lassen sich eine ganze Reihe von Problemen in der Informatik sehr elegant formulieren und lösen. Hier eine kleine Auswahl (Genaueres dazu später):

**Acht-Damen-Problem:** Acht Damen sollen so auf ein Schachbrett gestellt werden, dass keine Dame eine andere bedroht.

**Vier-Farben-Problem:** Eine Landkarte soll mit vier Farben so gefärbt werden, dass benachbarte Länder immer unterschiedliche Farben bekommen.

**Labyrinth-Problem:** Ein Labyrinth mit Sackgassen und Verzweigungen ist zu durch- laufen, um den Ausgang zu finden.

Konkreter:

(a) Man versucht, eine Kante (Verbindungsstrecke) zu zeichnen, wenn sie zulässig ist oder noch nicht gezeichnet wurde.

(b) Ist das nicht möglich, muss die zuletzt gezeichnete Kante gelöscht werden.

(c) Ist es möglich, dann hat man das Problem um eine Stufe vereinfacht.

(d) Hat man durch dieses Verfahren insgesamt 8 Kanten zeichnen können, hat man eine Lösung gefunden. Jetzt löscht man wieder die zuletzt gezeichnete Kante und sucht nach weiteren Lösungen.

## Realisierung des Programms

### Datenstrukturen

Die folgende Tabelle gibt an, welche Verbindungslinien zulässig sind (durch X markiert). Die erste Zeile bedeutet also, dass von Punkt 1 zu den Punkten 2, 3 und 5 Strecken gezeichnet werden dürfen. Eine solche Tabelle heißt auch Adjazenzmatrix (von adjazieren; lat.: anwohnen, anliegen). Eine solche Tabelle lässt sich durch `boolean[][] kanteZulaessig;` in einem zweidimensionalen Feld

speichern. Eine entsprechende Tabelle `boolean[][] kanteGezeichnet;` erfasst dann die schon gezeichneten Kanten. In einem weiteren eindimensionalen Feld wird jeweils eine Lösung erfasst.

### Methoden

Es bieten sich folgende Methoden zur Strukturierung des Programmes an:

(a) `void initArrays()`

(b) `void zeichneKante(int von, int nach)`

(c) `void loescheKante(int von, int nach)`

(d) `void loesungAusgeben()`

(e) `void versucheKanteZuZeichnen(int start)`: Die rekursive Methode soll vom Punkt start weitere Kanten zeichen.

(f) Das Hauptprogramm:

```
1  public static void main(String[] arg) {
2    initArrays();
3    for (int punktNr=1; punktNr<=maxPunktAnz; punktNr++) {
4      loesungsWeg[0] = punktNr; // Startpunkt eintragen
5      versucheKanteZuZeichnen(punktNr);
6    } // for punktNr
7    Out.println();
8    Out.println("Es ergaben sich " + loesungsAnzahl + " Loesungen.");
9  } // main
```

```
3  public class Nikolaus {
4
5    static final int maxPunktAnz = 5;
6    static final int maxKantenAnz = 8;
7    static boolean[][] kanteZulaessig;
8    static boolean[][] kanteGezeichnet;
9    static int[] loesungsWeg;
10   static int aktKantenAnzahl = 0;
11   static int loesungsAnzahl = 0;
12
13   static void initArrays() {
14     kanteZulaessig = new boolean[maxKantenAnz + 1][maxKantenAnz + 1];
15     kanteGezeichnet = new boolean[maxKantenAnz + 1][maxKantenAnz + 1];
16     loesungsWeg = new int[maxKantenAnz + 2]; // mit Startpunkt
17     // Erst mal alles auf false ;
18     for (int i = 1; i <= maxPunktAnz; i++) {
19       for (int k = 1; k <= maxPunktAnz; k++) {
20         kanteZulaessig[i][k] = false;
21         kanteGezeichnet[i][k] = false;
22       }
23     }
24     /*
25      * Zulaessige Kanten fuer das " Haus des Nikolaus " eintragen . Der
    ↪ Nummerierung
26      * liegt das Bild in main zu Grunde . Eine Anpassung 2an andere Graphen ist
27      * leicht moeglich .
28      */
29     kanteZulaessig[1][2] = true; // von 1 nach 2 zulaessig
30     kanteZulaessig[1][3] = true;
```

```
31      kanteZulaessig[1][5] = true;
32      kanteZulaessig[2][1] = true;
33      kanteZulaessig[2][3] = true;
34      kanteZulaessig[2][5] = true;
35      kanteZulaessig[3][1] = true;
36      kanteZulaessig[3][2] = true;
37      kanteZulaessig[3][4] = true;
38      kanteZulaessig[3][5] = true;
39      kanteZulaessig[4][3] = true;
40      kanteZulaessig[4][5] = true;
41      kanteZulaessig[5][1] = true;
42      kanteZulaessig[5][2] = true;
43      kanteZulaessig[5][3] = true;
44      kanteZulaessig[5][4] = true;
45      for (int i = 0; i <= maxKantenAnz; i++) {
46        loesungsWeg[i] = 0;
47      }
48    }
49
50    static void zeichneKante(final int von, final int nach) {
51      kanteGezeichnet[von][nach] = true;
52      kanteGezeichnet[nach][von] = true;
53      aktKantenAnzahl++;
54      // Anzahl bereits gezeichneter Kanten erhoehen
55      loesungsWeg[aktKantenAnzahl] = nach; // neuen Wegpunkt in Loesung aufnehmen
56    }
57
58    static void loescheKante(final int von, final int nach) {
59      kanteGezeichnet[von][nach] = false;
60      kanteGezeichnet[nach][von] = false;
61      aktKantenAnzahl--;
62    }
63
64    static boolean fertig() {
65      return (aktKantenAnzahl == maxKantenAnz);
66    }
67
68    static void loesungAusgeben() {
69      for (int i = 0; i <= maxKantenAnz; i++) {
70        System.out.print(loesungsWeg[i]);
71        System.out.print(" ");
72        loesungsAnzahl++;
73        if (loesungsAnzahl % 5 == 0) {
74          System.out.println();
75        }
76      }
77    }
78
79    static void versucheKanteZuZeichnen(final int start) {
80      for (int ziel = 1; ziel <= maxPunktAnz; ziel++) {
81        if (kanteZulaessig[start][ziel] && !kanteGezeichnet[start][ziel]) {
82          zeichneKante(start, ziel);
83          if (!fertig()) {
84            versucheKanteZuZeichnen(ziel);
85          } else {
86            loesungAusgeben();
87          }
88          loescheKante(start, ziel);
89        }
90      }
91    }
92
```

```
93    public static void main(final String[] arg) {
94      initArrays();
95      System.out.println(
96
            ↪  " Das Programm bestimmt alle Loesungen des Problems, das Haus des Nikolaus in einem Zug zu ze
97      System.out.println("      4      ");
98      System.out.println("     . .     ");
99      System.out.println("    .   .    ");
100     System.out.println("   5-----3   ");
101     System.out.println("   |.   .|   ");
102     System.out.println("   | . . |   ");
103     System.out.println("   | . . |   ");
104     System.out.println("   |.   .|   ");
105     System.out.println("   1-----2   ");
106     for (int punktNr = 1; punktNr <= maxPunktAnz; punktNr++) {
107       loesungsWeg[0] = punktNr;
108       versucheKanteZuZeichnen(punktNr);
109     }
110     System.out.println();
111     System.out.println(" Es ergaben sich " + loesungsAnzahl + " Loesungen.");
112   }
113 }
```